Linko Semiconductor Co., Ltd.

LKS32MC07x User Manual

Catalog

TABLE OF CONTENTS I

PICTURE DIRECTORY I

1 DOCUMENT CONVENTION 1

1.1 Register Read/Write Permissions 1

1.2 Glossary of abbreviations 1

2 OVERVIEW 3

2.1 Brief description 3

2.2 Characteristic 3

2.2.1 Memory 3

2.2.2 Clock 3

2.2.3 Peripheral module 3

2.2.4 Simulation module 4

2.3 Resource Diagram 5

3 ADDRESS SPACE 6

4 INTERRUPT 8

5 ANALOG CIRCUIT 9

5.1 Introduction 9

5.1.1 Power management system 10

5.1.2 Clock system 11

5.1.3 Bandgap Voltage Reference (BGP) 11

5.1.4 Analog-to-digital Converter (ADC) 12

5.1.5 Operational Amplifier (OPA) 12

5.1.6 Comparator 13

5.1.7 Temperature sensor 14

5.1.8 DAC module 15

5.2 Register 17

5.2.1 Address assignment 17

5.2.2 SYS_AFE_INFO 17

5.2.3 SYS_AFE_DBG 18

5.2.4 SYS_AFE_REG0 18
5.2.5 SYS_AFE_REG1 19
5.2.6 SYS_AFE_REG2 Analog Configuration 2 20
5.2.7 SYS_AFE_REG3 Analog Configuration Register 3 21
5.2.8 SYS_AFE_REG4 Analog Configuration 4 22
5.2.9 SYS_AFE_REG5 Analog Configuration 5 23
5.2.10 SYS_AFE_REG6 Analog Configuration 6 24
5.2.11 SYS_AFE_REG7 Analog Configuration Register 7 25
5.2.12 SYS_AFE_DAC_CTRL DAC control register 25
5.2.13 SYS_AFE_DAC0 DAC0 Digital Register 26
5.2.14 SYS_AFE_DAC1 DAC1 digital register 27
5.2.15 SYS_AFE_DAC0_AMC DAC0 Gain Correction Register 27
5.2.16 SYS_AFE_DAC0_DC DAC0 DC offset register 27
5.2.17 SYS_AFE_DAC1_AMC DAC1 Gain Correction Register 28
5.2.18 SYS_AFE_DAC1_DC DAC1 DC offset register 28
6 SYSTEM CONTROL AND CLOCK RESET 30
6.1 CLOCK 30
6.1.1 Clock source 30
6.2 RESET 32
6.2.1 Reset source 32
6.2.1.1 Hardware reset 32
6.2.1.1.1 Hardware reset architecture 33
6.2.1.1.2 Hardware reset logging 33
6.2.1.2 Software reset 33
6.2.2 Resets the scope 34
6.3 REGISTER 35
6.3.1 Address assignment 35
6.3.2 SYS_CLK_CFG Clock Control Register 35
6.3.3 SYS_IO_CFG IO control register 36
6.3.4 SYS_DBG_CFG Debug control register 37

6.3.5 SYS_CLK_DIV0 peripheral clock divide register 0 38
6.3.6 SYS_CLK_DIV2 Peripheral Clock Divide Register 2 39
6.3.7 SYS_CLK_FEN Peripheral Clock Gating Register 39
6.3.8 SYS_SFT_RST soft reset register 41
6.3.9 SYS_PROTECT Write Protect Register 43
6.3.10 SYS_FLSE Erase Protection Register 44
6.3.11 SYS_FLSP Program Protection Register 44
7 NON-VOLATILE MEMORY 45
7.1 INTRODUCTION 45
7.2 FUNCTIONAL FEATURES 47
7.2.1 Functional description 47
7.2.1.1 Reset operation 48
7.2.1.2 Sleep operation 48
7.2.1.3 Read operation 48
7.2.1.4 FLASH programming operation 49
7.2.1.5 FLASH erase operation 51
7.2.1.6 FLASH prefetch operation 52
7.2.1.7 Non-volatile memory bank protection 52
7.2.1.8 FLASH Online Upgrade (IAP) 54
7.2.1.8.1 Turn on interrupted online upgrade 54
7.2.1.8.2 Turn Off an Interrupted Online Upgrade 55
7.2.1.8.3 Location of Online Upgrade Function 55
7.3 REGISTER 55
7.3.1 Address assignment 55
7.3.2 FLASH_CFG configuration register (recommended to read back first and modify by or/and) 56
7.3.3 FLASH_ADDR Address Register 57
7.3.4 FLASH_WDATA Write Data Register 57
7.3.5 FLASH_RDATA Read Data Register 58
7.3.6 FLASH_ERASE Erase Control Register 58
7.3.7 FLASH_PROTECT Protection Status Register 58

7.3.8 FLASH_READY 59

7.3.9 FLASH_SIZE capacity register 59

7.3.10 FLASH_NCFG configuration register (recommended to read back first and modify by or/and) .. 60

7.3.11 FLASH_NADDR Address Register 61

7.3.12 FLASH_NWDATA Write Data Register 61

7.3.13 FLASH_NRDATA Read Data Register 62

7.3.14 FLASH_NERASE Erase Control Register 62

7.3.15 FLASH_NKEY Key Register 63

8 SPI 67

8.1 OVERVIEW 67

8.2 MAIN CHARACTERISTICS 67

8.3 FUNCTIONAL DESCRIPTION 67

8.3.1 Functional block diagram 67

8.3.2 Functional description 68

8.3.2.1 Full duplex mode 68

8.3.2.2 Half-duplex mode 69

8.3.2.3 Chip select signal 71

8.3.2.4 Communication format 71

8.3.2.5 Data format and length 71

8.3.2.6 DMA transfer 72

8.3.2.7 MCU transmission 72

8.3.2.8 External event transmission 73

8.3.2.1 Interrupt Handling 73

8.3.2.2 Baud rate setting 74

8.4 REGISTER 74

8.4.1 Address assignment 74

8.4.2 SPI_CFG SPI control register 74

8.4.3 SPI_IE SPI Interrupt Register 75

8.4.4 SPI_BAUD SPI Baud Rate Register 76

8.4.5 SPI_TXDATA SPI data transmit register 77

8.4.6 SPI_RXDATA SPI data receive register 77

8.4.7 SPI_SIZE SPI Data Transfer Length Register 77

9 I2C 79

9.1 OVERVIEW 79

9.2 MAIN CHARACTERISTICS 79

9.3 FUNCTIONAL DESCRIPTION 79

9.3.1 Functional block diagram 79

9.3.2 Functional description 80

9.3.2.1 Mode selection 80

9.3.2.2 Slave mode 82

9.3.2.2.1 Slave Mode Transmission 82

9.3.2.2.2 Slave Mode Transport 83

9.3.2.2.3 Receive from Mode Single Byte 83

9.3.2.3 Slave Mode DMA Transfer 84

9.3.2.3.1 Slave Mode DMA Transmit 84

9.3.2.3.2 Slave Mode DMA Receive 85

9.3.2.4 Master mode 85

9.3.2.4.1 Main mode single byte transfer 86

9.3.2.4.2 Master Mode Single Byte Send 86

9.3.2.4.3 Master Mode Single Byte Receive 87

9.3.2.4.4 Master mode DMA transfer 87

9.3.2.4.5 Master Mode DMA Transmit 87

9.3.2.4.6 Master mode DMA receive 88

9.3.2.5 DMA transfer 89

9.3.2.6 I2C bus exception handling 90

9.3.2.7 Interrupt Handling 90

9.3.2.8 Communication speed setting 91

9.4 REGISTER 91

9.4.1 Address assignment 91

9.4.2 I2C_ADDR Address Register 91

9.4.3 I2C_CFG system control register 92

9.4.4 I2C_SCR status control register 93

9.4.5 I2C_DATA Data Register 94

9.4.6 I2C_MSCR Master Mode Register 94

9.4.7 I2C_BCR I2C transfer control register 95

9.4.8 I2C_B SIZE I2C Transfer Length Register ..... 错误!未定义书签。

10 CMP 96

10.1 OVERVIEW 96

10.2 REGISTER 96

10.2.1 Address assignment 96

10.2.2 CMP_IE Interrupt enable register 96

10.2.3 CMP_IF Interrupt Flag Register 97

10.2.4 CMP_TCLK Divided clock control register 97

10.2.5 CMP_CFG Control Register 99

10.2.6 CMP_BLCWIN Windowing control register 102

10.2.7 CMP_DATA Output Data Register 103

11 HALL 104

11.1 OVERVIEW 104

11.2 IMPLEMENTATION DESCRIPTION 104

11.2.1 Signal source 104

11.2.2 Working clock 104

11.2.3 Signal filtering 104

11.2.4 Capture 105

11.2.5 Interrupt 106

11.2.6 Data flow 106

11.3 REGISTER 106

11.3.1 Address assignment 106

11.3.2 HALL_CFG HALL module configuration register 106

11.3.3 HALL_INFO HALL module information register 108

11.3.4 HALL_WIDTH HALL Width Meter Value Register 108

11.3.5 HALL_TH HALL module counter threshold register *109*

11.3.6 HALL_CNT HALL count register *109*

12 ADC **110**

12.1 OVERVIEW 110

12.1.1 Functional block diagram *110*

12.1.2 ADC trigger mode *111*

12.1.3 ADC Channel Select *112*

12.1.4 ADC Interrupt *112*

12.1.5 ADC output number system *113*

12.1.6 ADC range *114*

12.1.7 ADC correction *114*

12.1.8 ADC Threshold Monitoring (Analog Watchdog) *115*

12.1.9 Oversampling *115*

12.2 REGISTER 117

12.2.1 Address assignment *117*

12.2.2 Sample Data Register *118*

12.2.2.1 ADCx_DAT0(x = 0,1) 118

12.2.2.2 ADCx_DAT1(x = 0,1) 118

12.2.2.3 ADCx_DAT2(x = 0,1) 118

12.2.2.4 ADCx_DAT3(x = 0,1) 119

12.2.2.5 ADCx_DAT4(x = 0,1) 119

12.2.2.6 ADCx_DAT5(x = 0,1) 119

12.2.2.7 ADCx_DAT6(x = 0,1) 120

12.2.2.8 ADCx_DAT7(x = 0,1) 120

12.2.2.9 ADCx_DAT8(x = 0,1) 121

12.2.2.10 ADCx_DAT9(x = 0,1) 121

12.2.2.11 ADCx_DAT10(x = 0,1) 121

12.2.2.12 ADCx_DAT11(x = 0,1) 122

12.2.2.13 ADCx_DAT12(x = 0,1) 122

12.2.2.14 ADCx_DAT13(x = 0,1) 122

12.2.2.15 ADCx_IDAT0(x = 0,1) 123

12.2.2.16 ADCx_IDAT1(x = 0,1) 123

12.2.3 Signal source register 123

12.2.3.1 ADCx_CHN0(x = 0,1) 123

12.2.3.2 ADCx_CHN1(x = 0,1) 124

12.2.3.3 ADCx_CHN2(x = 0,1) 124

12.2.3.4 ADCx_CHN3(x = 0,1) 125

12.2.3.5 ADCx_ICHN(x = 0,1) 125

12.2.3.6 ADC channel signal selection 125

12.2.4 Number of sampling channels register 126

12.2.4.1 ADCx_CHNT(x = 0,1) 126

12.2.5 Configuration register 127

12.2.5.1 ADCx_GAIN(x = 0,1) 127

12.2.5.2 ADCx_CFG (x = 0,1) 128

12.2.5.3 ADC0_TRIG 129

12.2.5.4 ADC1_TRIG 130

12.2.6 Software trigger register 131

12.2.6.1 ADCx_SWT(x = 0,1) 131

12.2.7 Calibration register 132

12.2.7.1 ADCx_DC0(x = 0,1) 132

12.2.7.2 ADCx_AMC0(x = 0,1) 133

12.2.7.3 ADCx_DC1(x = 0,1) 133

12.2.7.4 ADCx_AMC1(x = 0,1) 134

12.2.8 Interrupt register 134

12.2.8.1 ADCx_IE(x = 0,1) 134

12.2.8.2 ADCx_IF(x = 0,1) 135

12.2.9 Analog watchdog 135

12.2.9.1 ADC0_LTH 135

12.2.9.2 ADC0_HTH 136

12.2.9.3 ADC0_GEN 136

12.3 APPLICATION GUIDE 137

12.3.1 ADC Sampling Trigger Mode 137

12.3.1.1 Single-segment trigger mode 138

12.3.1.2 Two-stage trigger mode 139

12.3.2 Interrupt 141

12.3.2.1 Single Segment Triggered Sampling Complete Interrupt 141

12.3.2.2 Two-segment triggered sample complete interrupt 141

12.3.3 Configuration modification 141

12.3.4 Select the corresponding analog channel 142

13 TIMER GENERAL TIMER 143

13.1 OVERVIEW 143

13.1.1 Functional block diagram 143

13.1.1.1 Register module 143

13.1.1.2 IO filter module 143

13.1.1.3 General Timer Module 143

13.1.1.4 Clock divider module 144

13.1.2 Functional features 144

13.2 CHARACTERISTIC 144

13.2.1 Clock Division 144

13.2.2 The interrupt flag is cleared 144

13.2.3 Filtering 144

13.2.4 Mode 145

13.2.4.1 Counter 145

13.2.4.2 Capture mode 145

13.2.4.3 Compare mode (edge-aligned PWM) 147

13.2.4.4 Single trigger 147

13.2.4.5 Center Mode (Complementary PWM) 148

13.2.5 External events 149

13.2.5.1.1 External start signal 149

13.2.5.1.2 External clock signal 150

13.2.5.1.3 External reset signal ... 151

13.2.5.1.4 External gate signal ... 151

13.2.6 ADC trigger ... 152

13.2.7 Encoder ... 152

13.2.7.1 Quadrature coded signal ... 152

13.2.7.2 Sign plus pulse signal ... 153

13.2.7.3 CCW/CW double pulse signal ... 154

13.3 REGISTER ... 155

13.3.1 Address assignment ... 155

13.3.2 TIMER0 register ... 157

13.3.2.1 TIMER0_CFG Timer0 configuration register ... 157

13.3.2.2 TIMER0_TH Timer0 threshold register ... 160

13.3.2.3 TIMER0_CNT Timer0 count register ... 160

13.3.2.4 TIMER0_CMP0 Timer0 Channel 0 Compare Capture Register ... 161

13.3.2.5 TIMER0_CMP1 Timer0 Channel 1 Compare Capture Register ... 161

13.3.2.6 TIMER0_EVT Timer0 external event select register ... 162

13.3.2.7 TIMER0_FLT Timer0 filter control register ... 164

13.3.2.8 TIMER0_IE Timer0 Interrupt Enable Register ... 164

13.3.2.9 TIMER0_IF Timer0 Interrupt Flag register ... 165

13.3.2.10 TIMER0_IO Timer0 IO control register ... 166

13.3.3 TIMER1 register ... 167

13.3.3.1 TIMER1_CFG Timer1 configuration register ... 167

13.3.3.2 TIMER1_TH Timer 1 threshold register ... 169

13.3.3.3 TIMER1_CNT Timer1 count register ... 170

13.3.3.4 TIMER1_CMP0 Timer1 Channel 0 Compare Capture Register ... 170

13.3.3.5 TIMER1_CMP1 Timer1 Channel 1 Compare Capture Register ... 171

13.3.3.6 TIMER1_EVT Timer1 external event select register ... 171

13.3.3.7 TIMER1_FLT Timer1 filter control register ... 173

13.3.3.8 TIMER1_IE Timer1 Interrupt Enable Register ... 174

13.3.3.9 TIMER1_IF Timer1 Interrupt Flag register ... 174

13.3.4 TIMER2 register ... 175

13.3.4.1 TIMER2_CFG Timer2 configuration register ... 175

13.3.4.2 TIMER2_TH Timer2 threshold register ... 178

13.3.4.3 TIMER2_CNT Timer2 count register ... 178

13.3.4.4 TIMER2_CMP0 Timer2 Channel 0 Compare Capture Register ... 179

13.3.4.5 TIMER2_CMP1 Timer2 Channel 1 Compare Capture Register ... 179

13.3.4.6 TIMER2_EVT Timer2 external event select register ... 180

13.3.4.7 TIMER2_FLT Timer2 filter control register ... 182

13.3.4.8 TIMER2_IE Timer2 Interrupt Enable Register ... 182

13.3.4.9 TIMER2_IF Timer2 Interrupt Flag register ... 183

13.3.5 TIMER3 register ... 184

13.3.5.1 TIMER3_CFG Timer3 configuration register ... 184

13.3.5.2 TIMER3_TH Timer3 threshold register ... 187

13.3.5.3 TIMER3_CNT Timer3 count register ... 187

13.3.5.4 TIMER3_CMP0 Timer3 Channel 0 Compare Capture Register ... 188

13.3.5.5 TIMER3_CMP1 Timer3 Channel 1 Compare Capture Register ... 188

13.3.5.6 TIMER3_EVT Timer3 external event select register ... 188

13.3.5.7 TIMER3_FLT Timer3 filter control register ... 190

13.3.5.8 TIMER3_IE Timer3 Interrupt Enable Register ... 191

13.3.5.9 TIMER3_IF Timer3 Interrupt Flag register ... 192

13.3.6 QEP0 register ... 193

13.3.6.1 QEP0_CFG QEP0 Configuration Register ... 193

13.3.6.2 QEP0_TH QEP0 Count Threshold Register ... 194

13.3.6.3 QEP0_CNT QEP0 count register ... 194

13.3.6.4 QEP0_IE QEP0 Interrupt Enable Register ... 194

13.3.6.5 QEP0_IF QEP0 Interrupt Flag register ... 195

13.3.7 QEP1 register ... 195

13.3.7.1 QEP1_CFG QEP1 Configuration Register ... 195

13.3.7.2 QEP 1_TH QEP1 Count Threshold Register ... 196

13.3.7.3 QEP1_CNT QEP1 Count Register ... 197

13.3.7.4 QEP1_IE QEP1 Interrupt Enable Register 197
13.3.7.5 QEP1_IF QEP1 Interrupt Flag register 197
14 MCPWM 199
14.1 OVERVIEW 199
14.1.1 Base Counter module 200
14.1.2 FAIL signal processing 202
14.1.3 MCPWM Special Output Status 204
14.1.4 IO DRIVER module 204
14.1.4.1 MCPWM Wave-form Output: Center-aligned Mode 205
14.1.4.2 MCPWM Wave-form Control: Center-aligned Push-pull Mode 206
14.1.4.3 MCPWM Wave-form Output: Edge-aligned Mode 207
14.1.4.4 MCPWM Waveform Control-Edge Aligned Push-Pull Mode 207
14.1.4.5 MCPWM IO : Dead-zone Control 208
14.1.4.6 MCPWM IO: Polarity Settings 209
14.1.4.7 MCPWM IO: Auto- Protection 209
14.1.5 ADC Trigger Timer module 209
14.1.6 Digital power applications 210
14.1.7 Interrupt 212
14.2 REGISTER 213
14.2.1 Address assignment 213
14.2.2 MCPWM0_TH00 215
14.2.3 MCPWM0_TH01 216
14.2.4 MCPWM0_TH10 216
14.2.5 MCPWM0_TH11 216
14.2.6 MCPWM0_TH20 217
14.2.7 MCPWM0_TH21 217
14.2.8 MCPWM0_TH30 218
14.2.9 MCPWM0_TH31 218
14.2.10 MCPWM0_TH40 219
14.2.11 MCPWM0_TH41 219

14.2.12 MCPWM0_TH50 ... 219

14.2.13 MCPWM0_TH51 ... 220

14.2.14 MCPWM0_TMR0 ... 220

14.2.15 MCPWM0_TMR1 ... 221

14.2.16 MCPWM0_TMR2 ... 221

14.2.17 MCPWM0_TMR3 ... 222

14.2.18 MCPWM0_TH0 ... 222

14.2.19 MCPWM0_TH1 ... 222

14.2.20 MCPWM0_CNT0 ... 223

14.2.21 MCPWM0_CNT1 ... 223

14.2.22 MCPWM0_UPDATE ... 224

14.2.23 MCPWM0_FCNT ... 226

14.2.24 MCPWM0_EVT0 ... 226

14.2.25 MCPWM0_EVT1 ... 227

14.2.26 MCPWM0_DTH00 ... 227

14.2.27 MCPWM0_DTH01 ... 228

14.2.28 MCPWM0_DTH10 ... 228

14.2.29 MCPWM0_DTH11 ... 229

14.2.30 MCPWM0_FLT ... 229

14.2.31 MCPWM0_SDCFG ... 229

14.2.32 MCPWM0_AUEN ... 230

14.2.33 MCPWM0_TCLK ... 231

14.2.34 MCPWM0_IE0 ... 232

14.2.35 MCPWM0_IF0 ... 233

14.2.36 MCPWM0_IE1 ... 234

14.2.37 MCPWM0_EIE ... 235

14.2.38 MCPWM0_EIF ... 236

14.2.39 MCPWM0_RE ... 236

14.2.40 MCPWM0_PP ... 237

14.2.41 MCPWM0_IO01 ... 238

14.2.42 MCPWM0_IO23 239

14.2.43 MCPWM0_IO45 240

14.2.44 MCPWM0_IF1 241

14.2.45 MCPWM0_FAIL012 242

14.2.46 MCPWM0_FAIL345 243

14.2.47 MCPWM0_CH_DEF 245

14.2.48 MCPWM0_CH_MSK 245

14.2.49 MCPWM0_PRT 246

14.2.50 MCPWM0_SWAP 247

14.2.51 MCPWM0_STT_HYST 247

14.2.52 MCPWM0_ZCS_DELAY 247

15 GPIO 248

15.1 OVERVIEW 248

15.1.1 Functional block diagram 248

15.1.2 Product features 248

15.2 REGISTER 249

15.2.1 Address assignment 249

15.2.2 GPIOx_PIE 250

15.2.3 GPIOx_POE 251

15.2.4 GPIOx_PDI 251

15.2.5 GPIOx_PDO 252

15.2.6 GPIOx_PUE 252

15.2.7 GPIOx_PODE 253

15.2.8 GPIOx_PFLT 254

15.2.9 GPIOx_F3210 254

15.2.10 GPIOx_F7654 255

15.2.11 GPIOx_FBA98 255

15.2.12 GPIOx_FFEDC 256

15.2.13 GPIOx_BSRR 256

15.2.14 GPIOx_BRR 258

15.2.15 External events ... 258
15.2.15.1 EXTI_CR0 ... 258
15.2.15.2 EXTI_CR1 ... 259
15.2.15.3 EXTI_IE ... 261
15.2.15.4 EXTI_IF ... 262
15.2.15.5 CLKO_SEL ... 263
15.2.15.6 PWM_SWAP ... 263
15.3 IMPLEMENTATION DESCRIPTION ... 265
15.3.1 Pull-up implementation ... 265
15.3.2 Filter implementation ... 265
15.3.3 Pull-up implementation ... 265
15.4 APPLICATION GUIDE ... 266
15.4.1 External Interrupt ... 266
15.4.2 Analog function using GPIO ... 266
16 CRC ... 267
16.1 OVERVIEW ... 267
16.2 BASIC PRINCIPLES ... 267
16.3 BASIC CONCEPTS ... 267
16.3.1 Correspondence ... 267
16.3.2 Generator polynomial ... 268
16.3.3 Number of check code bits ... 268
16.3.4 Generation step ... 268
16.4 REGISTER ... 270
16.4.1 Address assignment ... 270
16.4.2 Register description ... 270
16.4.2.1 CRC0_DR CRC information code register ... 270
16.4.2.2 CRC0_CR CRC Control Register ... 271
16.4.2.3 CRC0_INIT CRC Initial Code Register ... 272
16.4.2.4 CRC0_POL CRC generation code register ... 272
17 UART ... 273

17.1 OVERVIEW 273

17.2 FUNCTIONAL DESCRIPTION 273

17.2.1 Transport(TX) 273

17.2.2 Receive(RX) 274

17.2.3 Uart frame format 274

17.2.4 Baud rate configuration 275

17.2.5 Transmit/receive port interchange (TX/RX interchange) 276

17.2.6 Multi-machine communication 276

17.2.7 Check digit 278

17.3 REGISTER 278

17.3.1 Address assignment 278

17.3.2 UARTx_CTRL:UARTx Control Register (X = 0,1) 279

17.3.3 UARTx_DIVH:UARTx Baud Rate Setting High Byte Register (X = 0,1) 279

17.3.4 UARTx_DIVL:UARTx Baud Rate Setting Low Byte Register (X = 0,1) 280

17.3.5 UARTx_BUFF:UARTx Transmit and Receive Buffer Register (X = 0,1) 280

17.3.6 UARTx_ADR:UARTx Address Match Register (X = 0, 1) 281

17.3.7 UARTx_STT:UARTx Status Register (X = 0,1) 281

17.3.8 UARTx_RE:UARTx DMA Request Enable Register (X = 0,1) 282

17.3.9 UARTx_IE:UARTx Interrupt Enable Register (X = 0,1) 282

17.3.10 UARTx_IF:UARTx Interrupt Flag Register (X = 0, 1) 283

17.3.11 UARTx_IOC:UARTx IO Control Register (X = 0,1) 284

18 DMA 286

18.1 OVERVIEW 286

18.2 REQUEST 291

18.3 PRIORITY 292

18.4 ARBITRATION 293

18.5 INTERRUPT 293

18.6 REGISTER 293

18.6.1 Address assignment 293

18.6.2 DMA_CTRL DMA control register 294

18.6.3 DMA_IE DMA Interrupt Enable Register ... 294

18.6.4 DMA_IF DMA interrupt flag register ... 295

18.6.5 DMA Channel Configuration Register ... 295

18.6.5.1 DMA_CCRx (where x =0,1,2,3) ... 295

18.6.5.2 DMA_RENx (where x = 0,1,2,3) ... 296

18.6.5.3 DMA_CTMSx (where x = 0,1,2,3) ... 297

18.6.5.4 DMA_SADRx (where x = 0,1,2,3) ... 298

18.6.5.5 DMA_DADRx (where x = 0,1,2,3) ... 298

19 DSP ... 300

19.1 OVERVIEW ... 300

19.1.1 Functional block diagram ... 301

19.1.2 DSP Core Registers ... 301

19.1.3 Bit width ... 302

19.1.4 Instruction Cycle ... 302

19.1.5 Address space ... 303

19.1.6 Interaction with CPU, GPIO, CL module ... 304

19.2 REGISTER ... 305

19.2.1 Address assignment ... 305

19.2.2 DSP0_SC ... 305

19.2.3 DSP sin/cos Related Register ... 307

19.2.3.1 DSP0_THETA ... 307

19.2.3.2 DSP0_SIN ... 307

19.2.3.3 DSP0_COS ... 308

19.2.4 DSP arctan Related Register ... 308

19.2.4.1 DSP0_X ... 308

19.2.4.2 DSP0_Y ... 308

19.2.4.3 DSP0_MOD ... 309

19.2.4.4 DSP0_ARCTAN ... 309

19.2.5 DSP Division Related Register ... 310

19.2.5.1 DSP0_DID ... 310

19.2.5.2 DSP0_DIS 310

19.2.5.3 DSP0_QUO 311

19.2.5.4 DSP0_REM 311

19.2.6 DSP Root Correlation Register 312

19.2.6.1 DSP0_RAD 312

19.2.6.2 DSP0_SQRT 312

19.2.7 DSP0_PC 313

19.2.8 DSP0_PDI 313

19.2.9 DSP0_BSRR 314

19.2.10 DSP0_BRR 315

19.2.11 DSP0_CL 315

19.2.12 DSP0_OP 316

19.2.13 DSP0_RES 316

19.3 DSP INSTRUCTION SET 317

19.3.1 Instruction Set Summary 317

19.3.2 ADD 318

19.3.2.1 Coding 318

19.3.2.2 Assembly Syntax 318

19.3.2.3 Pseudo-code 318

19.3.3 AND 318

19.3.3.1 Instruction Code 318

19.3.3.2 Assembly Syntax 318

19.3.3.3 Pseudo-code 318

19.3.4 OR 318

19.3.4.1 Instruction Code 318

19.3.4.2 Assembly Syntax 319

19.3.4.3 Pseudo-code 319

19.3.5 SUB 319

19.3.5.1 Instruction Code 319

19.3.5.2 Assembly Syntax 319

19.3.5.3 Pseudo-code 319

19.3.6 ASR 319

19.3.6.1 Instruction Code 319

19.3.6.2 Assembly Syntax 319

19.3.6.3 Pseudo-code 320

19.3.7 ASRI 320

19.3.7.1 Instruction Code 320

19.3.7.2 Assembly Syntax 320

19.3.7.3 Pseudo-code 320

19.3.8 LSL 320

19.3.8.1 Instruction Code 320

19.3.8.2 Assembly Syntax 320

19.3.8.3 Pseudo-code 320

19.3.9 LSR 321

19.3.9.1 Instruction Code 321

19.3.9.2 Assembly Syntax 321

19.3.9.3 Pseudo-code 321

19.3.10 LSLI 321

19.3.10.1 Instruction Code 321

19.3.10.2 Assembly Syntax 321

19.3.10.3 Pseudo-code 321

19.3.11 MAC 321

19.3.11.1 Instruction Code 321

19.3.11.2 Assembly Syntax 322

19.3.11.3 Pseudo-code 322

19.3.12 MACI (reserved) 322

19.3.12.1 Instruction Code 322

19.3.12.2 Assembly Syntax 322

19.3.12.3 Pseudo-code 322

19.3.13 DIV 323

19.3.13.1 Instruction Code 323
19.3.13.2 Assembly Syntax 323
19.3.13.3 Pseudo-code 323
19.3.14 SAT 323
19.3.14.1 Instruction Code 323
19.3.14.2 Assembly Syntax 323
19.3.14.3 Pseudo-code 323
19.3.15 SATI (reserved) 324
19.3.15.1 Instruction Code 324
19.3.15.2 Assembly Syntax 324
19.3.15.3 Pseudo-code 324
19.3.16 SIN_COS 324
19.3.16.1 Instruction Code 324
19.3.16.2 Assembly Syntax 324
19.3.16.3 Pseudo-code 324
19.3.17 ARCTAN 324
19.3.17.1 Instruction Code 324
19.3.17.2 Assembly Syntax 325
19.3.17.3 Pseudo-code 325
19.3.18 SQRT 325
19.3.18.1 Instruction Code 325
19.3.18.2 Assembly Syntax 325
19.3.18.3 Pseudo-code 325
19.3.19 LDRWI 325
19.3.19.1 Instruction Code 325
19.3.19.2 Assembly Syntax 325
19.3.19.3 Pseudo-code 326
19.3.20 LDRDHI 326
19.3.20.1 Instruction Code 326
19.3.20.2 Assembly Syntax 326

19.3.20.3 Pseudo-code....326

19.3.21 STRWI....326

19.3.21.1 Instruction Code....326

19.3.21.2 Assembly Syntax....327

19.3.21.3 Pseudo-code....327

19.3.22 STRDHI (reserved)....327

19.3.22.1 Instruction Code....327

19.3.22.2 Assembly Syntax....327

19.3.22.3 Pseudo-code....327

19.3.23 JUMPI....328

19.3.23.1 Instruction Code....328

19.3.23.2 Assembly Syntax....328

19.3.23.3 Pseudo-code....328

19.3.24 JLE....328

19.3.24.1 Instruction Code....328

19.3.24.2 Assembly Syntax....328

19.3.24.3 Pseudo-code....328

19.3.25 JEI....328

19.3.25.1 Instruction Code....328

19.3.25.2 Assembly Syntax....328

19.3.25.3 Pseudo-code....329

19.3.26 JLEI....329

19.3.26.1 Instruction Code....329

19.3.26.2 Assembly Syntax....329

19.3.26.3 Pseudo-code....329

19.3.27 IRQ....329

19.3.27.1 Instruction Code....329

19.3.27.2 Assembly Syntax....329

19.3.27.3 Pseudo-code....329

19.3.28 R (simulator only)....329

19.3.28.1 Instruction Code 329
19.3.28.2 Assembly Syntax 330
19.3.28.3 Pseudo-code 330
19.3.29 BREAK (simulator only) 330
19.3.29.1 Instruction Code 330
19.3.29.2 Assembly Syntax 330
19.3.29.3 Pseudo-code 330
19.3.30 END (simulator only) 330
19.3.30.1 Instruction Code 330
19.3.30.2 Assembly Syntax 330
19.3.30.3 Pseudo-code 330
19.4 APPLICATION GUIDE 331
19.4.1 Memory access addressing 331
19.4.2 Delayed commit of multi-cycle instructions 331
19.4.3 Software-simulated serial interface 332
20 IWDG INDEPENDENT WATCHDOG 333
20.1 OVERVIEW 333
20.2 REGISTER 334
20.2.1 Address assignment 334
20.2.2 IWDG_PSW Independent Watchdog Password Register 335
20.2.3 IWDG_CFG Independent Watchdog Configuration Register 335
20.2.4 IWDG_CLR Independent Watchdog Clear Register 335
20.2.5 IWDG_WTH Independent Watchdog Timed Wake-up Threshold 336
20.2.6 IWDG_RTH Independent Watchdog Timeout Reset Threshold 336
20.2.7 IWDG_CNT Independent Watchdog Current Count Register 337
21 PMU POWER MANAGEMENT MODULE 338
21.1 PERIPHERAL CLOCK GATING 338
21.2 PERIPHERAL CLOCK DIVIDE 338
21.3 LOW POWER CONSUMPTION MODE 338
21.4 SLEEP MODE 338

21.4.1 Mode entry *339*

21.4.2 Mode exit *339*

21.5 DEEP SLEEP MODE 339

21.5.1 Mode entry *339*

21.5.2 Mode exit *340*

21.6 REGISTER 340

21.6.1 Address assignment *340*

21.6.2 AON_PWR_CFG Power Management Configuration Register *340*

21.6.3 AON_EVT_RCD Event Logging Register *341*

21.6.4 AON_IO_WAKE_POL IO wake-up polarity register *341*

21.6.5 AON_IO_WAKE_EN IO wake-up enable register *342*

22 SIF MODULE 343

22.1 BRIEF DESCRIPTION 343

22.2 MAIN CHARACTERISTICS 343

22.3 FUNCTIONAL DESCRIPTION 343

22.3.1 Functional block diagram *343*

22.3.2 Functional description *344*

22.3.2.1 SIF format description 344

22.3.2.2 To SC time base 344

22.3.2.3 Synchronize the time configuration 344

22.3.2.4 End time configuration 345

22.3.2.5 DMA transfer 345

22.4 REGISTER 345

22.4.1 Address assignment *345*

22.4.2 SIF0_CFG configuration register *345*

22.4.3 SIF0_TOSC time base register *346*

22.4.4 SIF0_TSTH1 synchronization time register *347*

22.4.5 SIF0_TDTH1 end time register *347*

22.4.6 SIF0_IRQ Interrupt Register *348*

22.4.7 SIF0_WDATA data register *348*

23 CL0 CONFIGURABLE LOGIC 350

23.1 OVERVIEW 350

23.2 MAIN CHARACTERISTICS 350

23.3 FUNCTIONAL DESCRIPTION 350

23.3.1 Functional block diagram 350

23.3.2 Functional description 351

23.3.2.1 Configure the process 351

23.3.2.2 Input multiplexer 352

23.3.2.3 Output configuration 353

23.3.2.4 LUT configuration 354

23.4 REGISTER 354

23.4.1 Address assignment 354

23.4.2 CL0_EN0 CL0 enable register 355

23.4.3 CL0_IE0 CL0 Interrupt Enable Register 355

23.4.4 CL0_IF0 CL0 Interrupt Flag register 356

23.4.5 CL0_OUT0 CL0 Output Register 357

23.4.6 CL0_MX0 CL0 Input Mux Register 357

23.4.7 CL0_FN0 CL0 function select register 358

23.4.8 CL0_CF0 CL0 configuration register 358

24 CAN 361

24.1 MAIN CHARACTERISTICS 361

24.2 FUNCTIONAL DESCRIPTION 361

24.2.1 Functional block diagram 361

24.2.2 Functional description 362

24.2.2.1 Working mode 362

24.2.2.2 Transmission 362

24.2.2.3 Interrupt processing 362

24.2.2.3.1 Receive interrupt 363

24.2.2.3.2 Transmit interrupt 363

24.2.2.3.3 Error interrupt 363

24.2.2.3.4 Data overflow interrupt 364

24.2.2.3.5 Arbitration lost interrupt 364

24.2.2.3.6 Bus error interrupt 364

24.2.2.4 Baud rate setting 364

24.2.2.5 ID filter 366

24.2.2.6 Reception of can frames 367

24.2.2.7 Transmission of CAN frame 369

24.2.2.8 Error Management 371

24.2.2.9 Error count 372

24.2.2.10 Active error state 372

24.2.2.11 Passive error state 372

24.2.2.12 Shutdown state and shutdown recovery 372

24.2.2.13 Arbitration 373

24.2.3 Register 373

24.2.3.1 Address assignment 373

24.2.3.2 Register description 374

24.2.3.2.1 CAN_RBUF0/1/2/3 Register 374

24.2.3.2.2 CAN_TBUF0/1/2/3 register 374

24.2.3.2.3 CAN_CFG_STAT configuration and status register 375

24.2.3.2.4 CAN_TCMD transmit command register 376

24.2.3.2.5 CAN_TCTRL transmit control register 377

24.2.3.2.6 CAN_RCTRL receive control register 378

24.2.3.2.7 CAN_RTIE Transmit Receive Interrupt Control Register 379

24.2.3.2.8 CAN_RTIF Transmit Receive Interrupt Flag Register 380

24.2.3.2.9 CAN_ERRINT Error Interrupt Enable and Flag Register 381

24.2.3.2.10 CAN_LIMIT Error & Warning Threshold Register 382

24.2.3.2.11 CAN_SBAUD Baud Rate Configuration Register 382

24.2.3.2.12 CAN_EALCAP Error Message and Missing Arbitration Message Record Register 383

24.2.3.2.13 CAN_RECNT receive error counter register 383

24.2.3.2.14 CAN_TECNT transmit error counter register 384

24.2.3.2.15 CAN_ACFCTRL ID Filter Control Register 384

24.2.3.2.16 CAN_ACFEN ID Filter Enable Register 385

24.2.3.2.17 CAN_ACF ID filter select register 385

25 VERSION HISTORY 387

Table of contents

Table 3-1 System Address Space Allocation 6

Table 4-1 Interrupt Number List 8

Table 5-1 System Control Register 17

Table 5-2 Chip Version Information Register(SYS_AFE_INFO) 17

表 5-3 SYS_AFE_DBG 18

Table 5-4 AFE Register 0 (SYS_AFE_REG0) 18

Table 5-5 SYS_AFE_REG1 Analog Configuration 1 19

Table 5-6 SYS_AFE_REG2 Analog Configuration 2 20

Table 5-7 SYS_AFE_REG3 Analog Configuration Register 3 21

Table 5-8 SYS_AFE_REG4 Analog Configuration 4 22

Table 5-9 SYS_AFE_REG5 Analog Configuration 5 23

Table 5-10 SYS_AFE_REG6 Analog Configuration 6 24

Table 5-11 SYS_AFE_REG7 Analog Configuration Register 7 25

Table 5-12 SYS_AFE_DAC_CTRL DAC control register 26

Table 5-13 SYS_AFE_DAC0 DAC0 digital register 26

Table 5-14 SYS_AFE_DAC1 DAC1 digital register 27

Table 5-15 SYS_AFE_DAC0_AMC DAC0 Gain Correction Register 27

Table 5-16 SYS_AFE_DAC0_DC DAC0 DC offset register 27

Table 5-17 SYS_AFE_DAC1_AMC DAC1 Gain Correction Register 28

Table 5-18 SYS_AFE_DAC1_DC DAC1 DC offset register 28

Table 6-1 System clock source 30

Table 6-2 Divide Configuration with PLL as MCLK Clock 31

Table 6-3 System reset source 32

Table 6-4 Resets the scope 34

Table 6-5 System Control Register 35

Table 6-6 SYS_CLK_CFG Clock Control Register 35

Table 6-7 SYS_IO_CFG IO control register 36

Table 6-8 SYS_DBG_CFG Debug control register 37

Table 6-9 SYS_CLK_DIV0 peripheral clock divide register 0 38

Table 6-10 SYS_CLK_DIV2 Peripheral Clock Divide Register 2........ 39

Table 6-11 SYS_CLK_FEN Peripheral Clock Gating Register........ 39

Table 6-12 SYS_SFT_RST soft reset register 41

Table 6-13 SYS_PROTECT Write Protect Register........ 43

Table 6-14 Erase protection register SYS_FLSE......... 44

Table 6-15 Program protection register SYS_FLSE. 44

Table 7-1 FLASH Access Space Allocation Table 48

Table 7-2 FLASH Sector Address Allocation Table 51

Table 7-3 IAP VTOR register description........ 54

Table 7-4 FLASH Controller Module Register List........ 55

Table7-5 FLASH_CFG configuration register 56

Table 7-6 FLASH_ADDR Address Register 57

Table7-7 FLASH_WDATA Write Data Register........ 57

Table7-8 FLASH_RDATA Read Data Register 58

Table7-9 FLASH_ERASE Erase Control Register........ 58

Table7-10 FLASH_PROTECT Protection Status Register........ 59

Table7-11 FLASH_READY operating status register........ 59

Table7-12 FLASH_SIZE capacity register 59

Table7-13 FLASH_NCFG configuration register 60

Table7-14 FLASH_NADDR Address Register........ 61

Table7-15 FLASH_NWDATA Write Data Register........ 61

Table7-16 FLASH_NRDATA Read Data Register 62

Table7-17 FLASH_NERASE Erase Control Register........ 62

Table7-18 FLASH_NKEY Key Register........ 错误!未定义书签。

Table 8-1 SPI Module Control Register List........ 74

Table 8-2 System Control Register SPI_CFG. 75

Table 8-3 SPI_IE Interrupt Register 75

Table 8-4 SPI_BAUD control register........ 76

Table 8-5 SPI_TXDATA data transmit register........ 77

Table 8-6 SPI_RXDATA data receive register 77

Table 8-7 SPI_SIZE Data Transfer Length Register 77

Table9-1 I2C register address allocation table 91

Table 9-2 Address register I2C_ADDR 91

Table 9-3 System control register I2C_CFG 92

Table 9-4 Status control register I2C_SCR 93

Table 9-5 Data register I2C_DATA 94

Table 9-6 Master Mode Register I2C_MSCR 94

Table 9-7 DMA transfer control register I2C_BCR 95

Table9-8 DMA transfer length register I2C_BSIZE .... 错误!未定义书签。

Table 10-1 List of comparator registers 96

Table 10-2 CMP_IE Comparator Interrupt Enable Register 96

Table 10-3 CMP_IF Comparator Interrupt Flag register 97

Table 10-4 CMP_TCLK comparator divided clock control register 97

Table 10-5 CMP_CFG Comparator Control Register 99

Table 10-6 CMP_BLCWIN Comparator Windowing Control Register 102

Table 10-7 CMP_DATA Comparator Output Data Register 103

Table 11-1 HALL module register address assignment 106

Table11-2 HALL_CFG HALL Module Configuration Register 107

Table 11-3 HALL_INFO HALL Module Information Register 108

Table 11-4 HALL_WIDTH HALL width count register 108

Table 11-5 HALL_TH HALL Module Counter Threshold Register 109

Table 11-6 HALL_CNT HALL count register 109

Table 12-1 Mapping ADC Channel Selection to Register Configuration 112

Table 12-2 ADC output digital quantity number-system conversion 113

Table 12-3 ADCx register list 117

Table 12-4 Sample Data Register ADCx_DAT0 118

Table 12-5 Sample Data Register ADCx_DAT1 118

Table 12-6 Sample Data Register ADCx_DAT2 118

Table 12-7 Sample Data Register ADCx_DAT3 119

Table 12-8 Sample Data Register ADCx_DAT4 119

Table 12-9 Sample Data Register ADCx_DAT5 120

Table 12-10 Sample Data Register ADCx_DAT6......120

Table 12-11 Sample Data Register ADCx_DAT7......120

Table 12-12 Sample Data Register ADCx_DAT8......121

Table 12-13 Sample Data Register ADCx_DAT9......121

Table 12-14 Sample Data Register ADCx_DAT10 121

Table 12-15 Sample Data Register ADCx_DAT11 122

Table 12-16 Sample Data Register ADCx_DAT12 122

Table 12-17 Sample Data Register ADCx_DAT13 122

Table 12-18 Idle Sample Data Register ADCx_IDAT0 123

Table 12-19 Idle Sample Data Register ADCx_IDAT1 123

Table 12-20 Signal source register ADCx_CHN0 123

Table 12-21 Signal source register ADCx_CHN1 124

Table 12-22 Signal source register ADCx_CHN2 124

Table 12-23 Signal source register ADCx_CHN3 125

Table 12-24 Signal source register ADCx_ICHN......125

Table12-25 ADC Analog Signal Source Distribution......125

Table 12-26 Sample Channel Count Register ADCx_CHNT......126

Table 12-27 Gain select register ADCx_GAIN......127

Table 12-28 Mode Configuration Register ADCx_CFG......128

Table 12-29 Sample trigger configuration register ADC0_TRIG......129

Table 12-30 Sample trigger configuration register ADC1_TRIG......130

Table 12-31 Software trigger register ADCx_SWT......131

Table 12-32 0 gain DC offset register ADCx_DC0 132

Table 12-33 0 Gain Gain Correction Register ADCx_AMC0......133

Table 12-34 1st gain DC offset register ADCx_DC1 133

Table 12-35 1st gain gain correction register ADCx_AMC1......134

Table 12-36 Interrupt Enable Register ADCx_IE 134

Table12-37Interrupt Flags Register ADCx_IF. 135

Table 12-38 Lower threshold register ADC0_LTH....136

Table 12-39 Upper threshold register ADC0_HTH....136

Table 12-40 Monitor enable register ADC0_GEN....136

Table 12-41 ADC Sampling Trigger Mode....138

Table 13-1 TIMER0 shadow register mapping....148

Table 13-2 Encoder orthogonal coding mode of operation....153

Table 13-3 Coder symbol plus pulse mode of operation....153

Table 13-4 Encoder CCW/CW Double Pulse Operation Mode....154

Table 13-5 Timer0 register address assignment....155

Table 13-6 Timer1 register address assignment....155

Table 13-7 Timer2 register address assignment....156

Table 13-8 Timer3 register address assignment....156

Table 13-9 QEP0 register address assignment....156

Table 13-10 QEP1 register address assignment....157

Table 13-11 Timer0 configuration register TIMER0_CFG....157

Table 13-12 Timer0 threshold register TIMER0_TH....160

Table 13-13 Timer0 count register TIMER0_CNT....161

Table 13-14 Timer0 Channel 0 Compare Capture Register TIMER0_CMP0....161

Table 13-15 Timer0 Channel 1 Compare Capture Register TIMER0_CMP1....161

Table 13-16 Timer0 external event select register TIMER0_EVT....162

Table 13-17 Timer0 filter control register TIMER0_FLT....164

Table13-18 Timer0 Interrupt Enable Register TIMER0_IE....164

Table13-19 Timer0 interrupt flag register TIMER0_IF....165

Table 13-20 Timer0 IO control register TIMER0_IO....166

Table13-21 Timer1 configuration register TIMER1_CFG....167

Table 13-22 Timer1 threshold register TIMER 1_TH....170

Table 13-23 Timer1 count register TIMER1_CNT....170

Table 13-24 Timer1 Channel 0 Compare Capture Register TIMER1_CMP0....170

Table 13-25 Timer1 channel 1 compare capture register TIMER1_CMP1....171

Table 13-26 Timer1 external event select register TIMER1_EVT....171

Table 13-27 Timer1 filter control register TIMER1_FLT 173

Table 13-28 Timer1 Interrupt Enable Register TIMER1_IE 174

Table 13-29 Timer1 interrupt flag register TIMER1_IF 174

Table 13-30 Timer2 configuration register TIMER2_CFG....175

Table 13-31 Timer2 threshold register TIMER2_TH....178

Table 13-32 Timer2 count register TIMER2_CNT 178

Table 13-33 Timer2 Channel 0 Compare Capture Register TIMER2_CMP0....179

Table 13-34 Timer2 channel 1 compare capture register TIMER2_CMP1....179

Table 13-35 Timer2 external event select register TIMER2_EVT....180

Table 13-36 Timer2 filter control register TIMER2_FLT 182

Table 13-37 Timer2 Interrupt Enable Register TIMER2_IE....182

Table 13-38 Timer2 interrupt flag register TIMER2_IF 183

Table 13-39 Timer3 configuration register TIMER3_CFG....184

Table13-40 Timer3 threshold register TIMER3_TH....187

Table13-41 Timer3 count register TIMER3_CNT 187

Table13-42 Timer3 Channel 0 Compare Capture Register TIMER3_CMP0....188

Table13-43 Timer3 channel 1 compare capture register TIMER3_CMP1....188

Table13-44 Timer3 external event select register TIMER3_EVT....189

Table 13-45 Timer3 filter control register TIMER3_FLT 191

Table 13-46 Timer3 Interrupt Enable Register TIMER3_IE....191

Table 13-47 Timer3 interrupt flag register TIMER3_IF 192

Table 13-48 QEP0 configuration register QEP0_CFG 193

Table 13-49 QEP0 count threshold register QEP0_TH 194

Table 13-50 QEP0 count register QEP0_CNT....194

Table 13-51 QEP0 Interrupt Enable Register QEP0_IE 194

Table 13-52 QEP0 interrupt flag register QEP0_IF....195

Table 13-53 QEP1 configuration register QEP1_CFG 195

Table13-54 QEP1 count threshold register QEP1_TH 196

Table 13-55 QEP1 count register QEP1_CNT....197

Table 13-56 QEP1 Interrupt Enable Register QEP1_IE 197

Table 13-57 QEP1 interrupt flag register QEP1_IF....197
Table14-1 MCPWM Counter Threshold and Event Mapping Table....209
Table14-2 MCPWM Module Register List....213
Table 14-3 Register protected by MCPWM0_PRT....214
Table 14-4 Registers where shadow registers exist....215
Table14-5 MCPWM0_TH00 Configuration Register....215
Table14-6 MCPWM0_TH00 Configuration Register....216
Table14-7 MCPWM0_TH10 Configuration Register....216
Table14-8 MCPWM0_TH11 Configuration Register....217
Table14-9 MCPWM0_TH20 Configuration Register....217
Table14-10 MCPWM0_TH21 Configuration Register....217
Table14-11 MCPWM0_TH30 configuration register....218
Table14-12 MCPWM0_TH31 configuration register....218
Table14-13 MCPWM0_TH40 configuration register....219
Table14-14 MCPWM0_TH41 Configuration Register....219
Table14-15 MCPWM0_TH50 configuration register....220
Table14-16 MCPWM0_TH51 Configuration Register....220
Table14-17 MCPWM0_TMR0 Configuration Register....220
Table14-18 MCPWM0_TMR1 Configuration Register....221
Table14-19 MCPWM0_TMR2 Configuration Register....221
Table14-20 MCPWM0_TMR3 Configuration Register....222
Table14-21 MCPWM0_TH0 Time Base 0 Register....222
Table14-22 MCPWM0_TH1 Time Base 1 Register....223
Table14-23 MCPWM0_CNT0 Register....223
Table14-24 MCPWM0_CNT1 Register....223
Table14-25 MCPWM0_UPDATE MCPWM Manual Update Register....224
Table14-26 MCP WM0_FCNT Register....226
Table14-27 MCPWM0_EVT0 MCPWM Time Base 0 External Trigger Register....226
Table14-28 MCPWM0_EVT1 MCPWM Time Base 1 External Trigger Register....227
Table14-29 MCPWM0_DTH00 Configuration Register....228

Table14-30 MCPWM0_DTH01 Configuration Register........228

Table14-31 MCPWM0_DTH10 Configuration Register........228

Table14-32 MCPWM0_DTH11 Configuration Register........229

Table14-33 MCPWM0_FLT MCPWM Filter Clock Divide Register........229

Table14-34 MCPWM0_SDCFG Configuration Register........230

Table 14-35 MCPWM0_AUEN MCPWM Auto Update Enable Register........231

Table14-36 MCPWM0_TCLK configuration register........232

Table14-37 MCPWM0_IE0 MCPWM Time Base 0 Interrupt Control Register........233

Table14-38 MCPWM0_IF0 MCPWM Time Base 0 Interrupt Flag Register........233

Table14-39 MCPWM0_IE1 MCPWM Time Base 1 Interrupt Control Register........234

Table14-40 MCPWM0_EIE Configuration Register........235

Table14-41 MCPWM0_EIF Configuration Register........236

Table14-42 MCPWM0_RE Configuration Register........236

Table14-43 MCP WM0_PP Configuration Register........237

Table14-44 MCPWM0_IO01 Configuration Register........238

Table14-45 MCPWM0_IO23 configuration register........239

Table14-46 MCPWM0_IO45 configuration register........240

Table14-47 MCPWM0_IF1 MCPWM Time Base 1 Interrupt Flag Register........241

Table14-48 MCPWM0_FAIL012 configuration register........242

Table14-49 MCPWM0_FAIL345 Configuration Register........243

Table14-50 MCPWM 0_CH_DEF Short-Circuit Protection Channel Output Value Register........245

Table14-51 MCPWM0_CH_MSK Channel Mask Bit Register........246

Table14-52 MCP WM0_PRT Register........246

Table14-53 MCPWM 0_STT_HYST dwell delay register........247

Table14-54 MCPWM0_ZCS_DELAY Status Dwell Time Register........247

Table 15-1 GPIO secondary function........249

Table15-2 List of GPIOx registers........250

Table15-3 GPIO Interrupt Module Register List........250

Table15-4 GPIOx Input Enable Register GPIOx_PIE........250

Table15-5 GPIOx Output Enable Register GPIOx_POE........251

Table15-6 GPIOx Input Data Register GPIOx_PDI.....251
Table15-7 GPIOx Output Data Register GPIOx_PDO.....252
Table15-8 GPIOx pull-up enable register GPIOx_PUE.....252
Table15-9 GPIOx open-drain enable register GPIOx_PODE.....253
Table15-10 GPIOx Filter Register GPIOx_PFLT.....254
Table15-11 GPIOx function select register GPIOx_F3210.....254
Table15-12 GPIOx Function Select Register GPIOx_F7654.....255
Table15-13 GPIOx function select register GPIOx_FBA98.....255
Table15-14 GPIOx function select register GPIOx_FFEDC......256
Table15-15 GPIOx Bits Operation Register GPIOx_BSRR......256
Table15-16 GPIOx bits clear register GPIOx_BRR.....258
Table15-17 EXTI_CR0 external trigger configuration register 0.....259
Table15-18 EXTI_CR1 External Trigger Configuration Register 1.....260
Table15-19 GPIO Interrupt/DMA Request Event Resource Distribution Table.....260
Table15-20 EXTI_IE GPIO Interrupt Enable Register.....261
Table15-21 EXTI_IF external interrupt flag register.....262
Table15-22 CLKO_SEL GPIO output clock select register.....263
Table15-23 PWM_SWAP register.....263
Table15-24 MCPWM default output table.....264
Table15-25 MCPWM Output Mapping Table when PWM_SWAP = 1.....264
Table15-26 MCPWM output mapping table when PWM_SWAP = 2.....264
Table15-27 GPIO pull-up resource table.....265
Table15-28 GPIO filter resource table.....265
Table15-27 GPIO open-drain resource table.....265
Table16-1 List of CRC registers.....270
Table16-2 CRC0_DR CRC Data Register.....270
Table16-3 CRC0_CR CRC Control Register.....271
Table16-4 CRC0_INIT CRC Initial Code Register.....272
Table16-5 CRC0_POL CRC generation code register.....272
Table17-1 Example of Uart baud rate configuration.....275

Table17-2 Uart frame format......278

Table17-3 UART address allocation list......278

Table17-4 UART control register UARTx_CTRL.......279

Table17-5 Uart Baud Rate Setting High Byte Register UARTx_DIVH.......280

Table17-6 Uart Baud Rate Set Low Byte Register UART_DIVL.......280

Table17-7 UART Transmit Buffer Register UART_BUFF.......280

Table17-8 UART Address Match Register UART_ADR......281

Table17-9 UART status register UART_STT.......281

Table17-10 UART DMA Request Enable Register UART_RE.......282

Table17-11 UART Interrupt Enable Register UART_IE.......282

Table17-12 UART Interrupt Enable Register UART_IF.......283

Table17-13 UARTIO control register UART_IOC.......284

Table18-1 DMA request......291

Table18-2 DMA register list......293

Table18-3 DMA Control Register DMA_CTRL.......294

Table18-4 DMA Interrupt Enable Register DMA_IE......294

Table18-5 DMA Interrupt Flags Register DMA_IF......295

Table18-6 DMA channel configuration register DMA_CCRx.......295

Table18-7 DMA request enable register DMA_RENx......296

Table18-8 DMA transfer count register DMA_CTMSx.......297

Table18-9 DMA Source Address Register DMA_SADRx.......298

Table18-10 DMA destination address register DMA_DADRx.......299

Table19-1 DSP Core Registers......301

Table 19-2 Table of special operands of division......302

Table19-3 DSP address space......303

Table 19-4 DSP data memory space read-write access content......304

Table19-5 DSP register list......305

Table19-6 DSP status control register DSP0_SC......305

Table19-7 DSP sin/cos angle input register......307

Table19-8 DSP sin/cos sine result register......307

Table19-9 DSP sin/cos Cosine Result Register 308

Table19-10 DSP arctan/module Coordinate X Input Register 308

Table19-11 DSP arctan/module Compute Coordinate Y Input Register......309

Table19-12 DSP arctan vector magnitude result sqrt (X^2+ Y^2) register......309

Table19-13 DSP arctan angle result arctan (Y/X) angle register 309

Table19-14 DSP divide dividend register......310

Table19-15 DSP Divide Divisor Register 310

Table19-16 DSP Division Quotient Register......311

Table19-17 DSP Division Remainder Register......311

Table19-18 DSP radicand register......312

Table19-19 DSP Square Root Register......312

Table19-20 DSP Program Pointer Register DSP0_PC......313

Table19-21 DSP0_PDI Register......313

Table19-22 DSP0_BSRR register......314

Table19-23 DSP0_BSRR register......315

Table19-24 DSP0_CL register......315

Table19-25 DSP0_OP Register 316

Table19-26 DSP0_RES Register......316

Table 19-27 DSP instruction set......317

Table20-1Independent watchdog register 334

Table20-2 IWDG_PSW Independent Watchdog Password Register......335

Table20-3 IWDG_CFG Independent Watchdog Configuration Register......335

Table20-4 IWDG_CLR watchdog clear register 336

Table20-5 IWDG_WTH Independent Watchdog Timeout Reset Threshold......336

Table20-6 IWDG_RTH Independent Watchdog Timeout Reset Threshold 336

Table20-7 IWDG_CNT Independent Watchdog Current Count Register......337

Table 21-1 Low Power Modes Summary......338

Table21-2Power Management Module Address Space 340

Table21-3 AON_PWR_CFG Power Management Configuration Register 340

Table21-4 AON_EVT_RCD Event Logging Register......341

Table21-5 AON_IO_WAKE_POL IO wake-up source polarity configuration register....341
Table21-6 AON_IO_WAKE_EN IO wakeup source enable register....342
Table22-1 SIF module control register list....345
Table22-2Address register SIF 0_CFG....345
Table22-3 SIF 0_TOSCL SIF Time Base Low Byte Register....346
Table22-4Synchronization time register SIF 0_TSTH1L....347
Table22-5End time register SIF0_TDTH1....347
Table22-6Interrupt register SIF0_IRQ....348
Table22-7Data register SIF0_WDATA....348
Table23-1 CLUnA input selection....352
Table23-2 CLUnB input selection....352
Table 23-3 Timing of the CLU clock/external signal....354
Table23-4 LUT truth table....354
Table23-5 CL0 module register address assignment....354
Table23-6 CL0_EN0 CL0 enable register....355
Table23-7 CL0_IE0 CL0 Interrupt Enable Register....355
Table23-8 CL0_IF0 CL0 Interrupt Flag register....356
Table23-9 CL0_OUT0 CL0 output register....357
Table23-10 CL0_MX0 CL0 Input Mux Register....357
Table23-11 CL0_FN0 CL0 function select register....358
Table23-12 CL0_CF0 CL0 configuration register....358
Table24-1The structure of the received frame....365
Table24-2The structure of the received frame....368
Table24-3The structure of the transmit frame....370
Table24-4 CAN Error Count and Receive Error Count Rules....372
Table24-5 CAN register address assignment....373
Table24-6Read Register CAN_RBUF0/1/2/3....374
Table24-7Write to Register CAN_TBUF0/1/2/3....374
Table24-8Configuration and status register CAN_CFG_STAT....375
Table24-9Transmit Command Register CAN_TCMD....376

Table24-10Transmit control register CAN_TCTRL......377

Table24-11Receive control register CAN_RCTRL......378

Table24-12Transmit Receive Interrupt Control Register CAN_RTIE......379

Table24-13Transmit Receive Interrupt Flag Register CAN_RTIF. 380

Table24-14Error Interrupt Enable and Flag Register CAN_ERRINT......381

Table24-15Error & Warning Threshold Register CAN_LIMIT. 382

Table24-16Baud rate configuration register CAN_SBAUD. 382

Table24-17Error Information and Missing Arbitration Information Record Register CAN_EALCAP.383

Table24-18Receive error counter register CAN_RECNT. 383

Table24-19Transmit error counter register CAN_TECNT. 384

Table24-20 ID Filter Control Register CAN_ACFCTRL. 384

Table24-21 ID Filter Enable Register CAN_ACFEN......385

Table24-22 ID Filter Select Register CAN_ACF......386

Table25-1Document version history 387

Picture directory

Figure 2-1 LKS32MC07x system diagram 5

Figure 5-1 Functional Block Diagram of Analog Circuit 10

Figure 5-2 Amplifier Block Diagram 12

Figure 5-3 BEMFx_MID Signal 14

Figure 6-1 Clock architecture 32

Figure 6-2 Reset the architecture 33

Figure7-1 Non-volatile memory space division block diagram and model 46

Figure 7-2 FLASH control state transition diagram 47

Figure 7-3 FLASH Indirect Read Operation Flowchart 49

Figure 7-4 Flow chart of FLASH module programming operation 49

Figure 7-5 Flow chart of FLASH module programming operation 50

Figure 7-6 FLASH Module Erase Operation Flowchart 52

Figure 8-1 Structure block diagram of SPI module 68

Figure 8-2 SPI Interface Full Duplex Interconnect Block Diagram 69

Figure 8-3 SPI Interface Interconnect Block Diagram in Half-Duplex Mode 70

Figure 8-4 SPI module Slave mode chip select signal selection 71

Figure 8-5 SPI module Master mode chip select signal selection 71

Figure 8-6 SPI module interrupt selection signal generation diagram 73

Figure 9-1 I2C Module Top Level Functional Block Diagram 80

Figure 9-2 Basic I2C Transfer Timing Diagram 81

Figure 9-3 Schematic diagram of slave mode transmission 83

Figure 9-4 Schematic diagram of multi-byte sending in slave mode 85

Figure 9-5 Schematic diagram of multi-byte receiving in slave mode 85

Figure 9-6 Schematic diagram of single byte transmission in master mode 86

Figure 9-7 Schematic diagram of multi-byte transmission in master mode 88

Figure 9-8 Schematic diagram of multi-byte reception in master mode 89

Figure 10-1 Comparator filter clock generation 99

Figure 10-2 Comparator control and interrupt generation logic 100

Figure 10-3 Linkage of CMP and MCPWM 101

Note: CMP_CFG.CMP0_PWM_POL and CMP_CFG.CMP1_PWM_POL will also affect the comparator signal sent to the MCPWM module as a fail signal, as shown in Figure10-4. The MCPWM FAIL signal from the comparator is the original signal output by the analog comparator, which is not filtered by the comparator digital interface module, but can be windowed by the MCPWM channel signal. See the comparator digital interface module for windowing control settings. After the FAIL signal enters the MCPWM block, it can be filtered by setting the MCPWM_TCLK. 101

Figure 10-5 Illustration of comparator windowing function 102

Figure 11-1 Block diagram of 7/5 filter module 105

Figure 11-2Data flow block diagram 106

Figure 12-1 ADC Acquisition Module Functional Block Diagram 111

Figure 12-2 ADC interrupt generated 113

Figure 12-3 ADC Analog-to-Digital Conversion Scale at 2/3 Gain Settings 114

Figure12-4 Oversampling Conversion Timing Example 1 116

Figure12-5 Oversampling Conversion Timing Example 2 116

Figure 12-6 ADC Single Segment Sampling State Transition Diagram 139

Take two software-triggered two-segment sampling as an example, the state transition is as shown in the figure12-7As shown 140

Take the two-round sampling triggered by two software trigger as an example, the state transition is shown in Figure12-8. 140

Figure 12-9 ADC Two-Segment Sampling State Transition Diagram 141

Figure 13-1 TIMER Module Top Level Functional Block Diagram 143

Figure 13-2 Schematic diagram of TIMER filtering 145

Figure 13-3 TIMER General counter 145

Figure 13-4 TIMER capture mode 146

Figure 13-5 TIMER comparison mode 147

Figure 13-6 TIMER single trigger 148

Figure13-7 TIMER Center Count Mode 149

Figure 13-8 TIMER External Event Trigger Start 150

Figure13-9 TIMER external event triggers single (single cycle) count 150

Figure 13-10 TIMER counts with an external clock 151

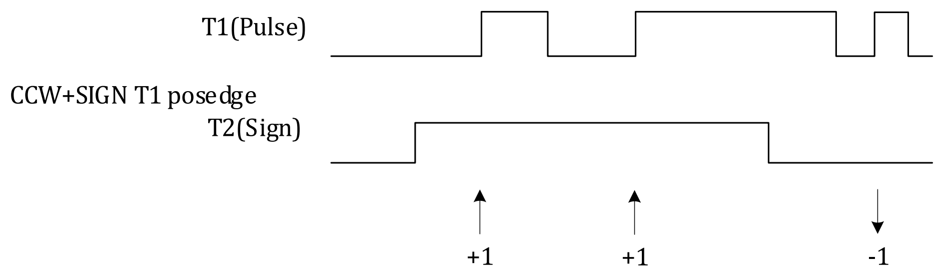
Figure 13-11 TIMER external signal reset function 151

Figure13-12 TIMER external signal gating function 151

Figure 13-13 Quadrature Encoded Signal Count Case where Encoder Counts Only at Time T1......153

Figure 13-14 Quadrature Encoded Signal Count Case Counted by Encoder at Time T1 or T2......153

Figure 13-15 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts both on T1 Rising and Falling Edges



......154

Figure 13-16 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts only on T1 Rising Edges......154

Figure 13-17 CCW/CW Double Pulse Signal Counting when Encoder Counts only on the T1 and T2 Rising Edge......155

Figure 13-18 CCW/CW Double Pulse Signal Counting when Encoder Counts both on the Rising and Falling Edge T1 and T2......155

Figure14-1 Block diagram of MCPWM module 200

Figure14-2 Base Counter t0/T1 Timing......201

Figure14-3 MCPWM update mechanism......202

Figure14-4 MCPWM FAIL Logic Diagram......203

Filter is the clock of the filter module, which comes from the gated clock FCLK of the system master clock MCLK, as shown in the SYS_CLK_FEN, and is divided by two optional stages. The first stage is controlled by the MCPWM0_TCLK.CLK_DIV to divide by 1/2/4/8. The second stage of frequency division can realize the frequency division of 1 to 256 times. The MCPWM0_FLT.FLT_CLKDIV[7:0] is used as the division factor of the second stage,as shown inFigure 14-5. 203

Figure 14-6 MCPWM Fail Signal Filter Clock Generation Logic......204

Figure14-7 IO Driver Module Data Flow Diagram......205

Figure14-8MCPWM Timings TH < n > 0 and TH < n > 1-Center Alignment Mode......206

Figure14-9 MCPWM Timings TH < n > 0 and TH < n > 1-Center Aligned Push-Pull Mode......206

Figure14-10MCPWM Timing Edge Alignment Mode......207

Figure14-11MCPWM Timing TH < n > 0 and TH < n > 1 Edge-Aligned Push-Pull Mode......208

Figure14-12MCPWM IO Control Schematic......209

Figure14-13 Timing Diagram of MCPWM Channel 0 Controlled by Comparator 1 for State Action ..212

Figure14-14 MCPWM Channel 0 Action Timing When Comparator 2 Triggers ZCS Event 212

Figure15-1 GPIO Functional Block Diagram 248

Figure17-1 Uart frame format..........275

Figure17-2 UART multi-machine communication interconnection topology 276

Figure17-3 UART Multi-machine Communication Example 278

Figure18-1 Multi-layer AHB bus architecture 286

Figure18-2 RMODE = 0 DMA transfer 287

Figure18-3 RMODE = 1 DMA transfer 287

Figure18-4 DMA address increment control 289

Figure18-5 DMA transfer with CIRC = 1 and RMODE = 1..........290

Figure18-6 DMA transfer with CIRC = 1 and RMODE = 0..........290

Figure18-7 DMA channel priority 292

Figure19-1 Functional block diagram of DSP module 301

Figure20-1 Example of an IWDG countdown event occurring 333

Figure22-1 Structure block diagram of SIF module 343

Figure22-2 SIF Module Sync (SYNC) Waveform..........344

Figure22-3 SIF module data (DATA) waveform..........344

As shown in the figure23-1As shown, the configurable logic CL0 mainly includes four independent configurable logic units CLU. Each CLU can be independently configured as a user-defined asynchronous or synchronous digital logic function without CPU intervention. A number of internal and external signals are available as inputs to each CLU, and the outputs of the CLUs can also be connected to IO outputs or directly to selected peripheral inputs..........350

Figure23-2 CL Module Top Level Functional Block Diagram..........351

Figure23-3 Functional block diagram of independent CLU module..........351

Figure24-1 CAN Module Top Level Functional Block Diagram..........361

Figure24-2 CAN module Bit cycle introduction diagram 364

Figure24-3 CAN module ID filtering logic..........367

Figure24-4 CAN module error management..........371

1 Document Convention

1.1 Register Read/Write Permissions

RW	Read/write, available for software read and write.
RO	Read-only, software can only read.
WO	Write-only, software can only write. The default value will be returned when reading this bit.
RW1C	Read and Write, 1 to Clear

1.2 Glossary of abbreviations

Word: 32-bit data/instruction.

Halfword: 16-bit data/instruction.

Byte: 8-bit data.

Double word: 64-bit data.

ADC: Analog-Digital Converter

DAC: Digital-Analog Converter

BGP: Bandgap. Bandgap voltage reference

WDT: Watch dog

LSI: Low Speed Internal Clock, the 32KHz RC oscillator

HSI: High Speed Internal Clock, the 4MHz RC oscillator

HSE: High Speed External Clock, the 4~8MHz external crystal clock

PLL: Phase Lock Loop Clock, the 96MHz PLL clock, which usually used as high-speed system clock

POR: Power-On Reset. Reset signal generated when the chip system is powered on

NVR: Non-Volatile Register. A storage area in the flash that is different from the main area.

IAP (In-Application Programming): IAP means that the Flash of the microcontroller can be reprogrammed while the user program is running.

ICP (In-Circuit Programming): ICP means that you can use the JTAG protocol, SWD protocol or bootloader to program the Flash of the microcontroller when the device is installed on the Subscriber Circuit Board.

CW: Clockwise

CCW: Counterclockwise

Option bytes: Option byte, the MCU configuration byte saved in Flash.

2 Overview

2.1 Brief description

The LKS32MC07x is enhanced mainline MCU with a 96MHz ARM Cortex-M0 core and the necessary analog and peripheral resources.

2.2 Characteristic

- 96 MHz 32-bit Cortex-M0 core, 32-bit hardware division coprocessor
- 4-channel DMA
- 10 uA Low Power Sleep Mode
- -40 ~ 105 °C industrial temperature range
- 2.5~5.5V power supply, with an integrated internal 5V LDO for partial power supply for internal MCU of chip
- High ESD and group pulse reliability

2.2.1 Memory

- 64/128kB Flashwith data anti-theft
- 12kB RAM

2.2.2 Clock

- 8MHz built-in high-precision RC oscillator, with an accuracy of ± 1% at -40 ~ 105 ℃
- 32KHz built-in low-speed clock for low-power mode
- Internal PLL up to 96 MHz

2.2.3 Peripheral module

- Two UARTs
- One SPI
- One IIC
- 16/32-bit standard timers, support capture and edge-aligned PWM function, Timer0 supports center-aligned PWM

- Motor control PWM module, supports 12 channels/6 pairs of PWM waveform output, independent dead-band control
- Hall signal interface with speed measurement and debouncing function
- Hardware watchdog
- Supports up to 64 GPIOs

2.2.4 Simulation module

- Two 12bit SAR ADC, simultaneous double sampling, 3Msps sampling and conversion rate, 14 analog signal channels
- Four operational amplifiers. Differential PGA mode is available.
- Three comparators. Hysteresis mode is available.
- Two 12bit digital-to-analog converter (DAC)
- 1.2V 0.5% built-in linear regulator
- Low-power LDO and power monitoring circuit
- RC oscillator with high precision and low temperature drift

2.3 Resource Diagram

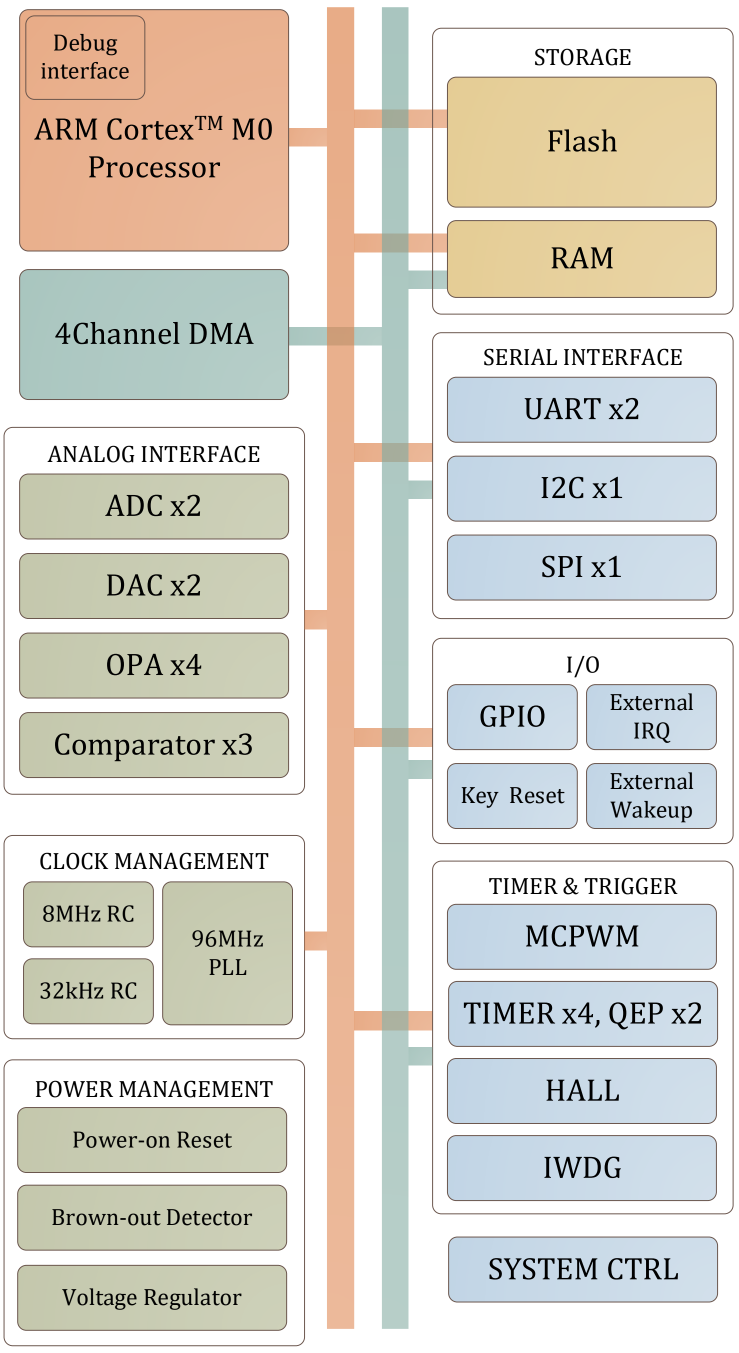


Figure 2-1 LKS32MC07x system diagram

3 Address space

The data bytes are stored in memory in little-endian format. The lowest address byte in a word is considered the least significant byte of the word, and the highest address byte is the most significant byte. All other unallocated on-chip memory and external memory are reserved address spaces.

Table 3-1 System Address Space Allocation

Type	Peripheral Modules	Start Address	End Address	Size	Clock/Soft Reset
CODE	FLASH	0x0000_0000	0x0001_FFFF	128kB	Same as bus
	FLSCR	0x0002_0000	0x0002_00FF	256B	Same as bus
RAM	RAM	0x2000_0000	0x2000_2FFF	12kB	Same as bus
	SYS	0x4000_0000	0x4000_00FF	256B	Same as bus
	SPI0	0x4001_0000	0x4001_00FF	256B	SYS_CLK_FEN[0] SYS_SFT_RST[0]
	I2C0	0x4001_0100	0x4001_01FF	256B	SYS_CLK_FEN[1] SYS_SFT_RST[1]
	CMP	0x4001_0200	0x4001_02FF	256B	SYS_CLK_FEN[2] SYS_SFT_RST[2]
	HALL0	0x4001_0300	0x4001_03FF	256B	SYS_CLK_FEN[3] SYS_SFT_RST[3]
	ADC0	0x4001_0400	0x4001_04FF	256B	SYS_CLK_FEN[12] SYS_SFT_RST[12]
	ADC1	0x4001_0500	0x4001_05FF	256B	SYS_CLK_FEN[13] SYS_SFT_RST[13]
	TIMER0	0x4001_0600	0x4001_06FF	256B	SYS_CLK_FEN[4] SYS_SFT_RST[4]
	TIMER1	0x4001_0700	0x4001_07FF	256B	SYS_CLK_FEN[5] SYS_SFT_RST[5]
	TIMER2	0x4001_0800	0x4001_08FF	256B	SYS_CLK_FEN[6] SYS_SFT_RST[6]
	TIMER3	0x4001_0900	0x4001_09FF	256B	SYS_CLK_FEN[7] SYS_SFT_RST[7]
	QEP0	0x4001_0A00	0x4001_0AFF	256B	SYS_CLK_FEN[8] SYS_SFT_RST[8]
	QEP1	0x4001_0B00	0x4001_0BFF	256B	SYS_CLK_FEN[9] SYS_SFT_RST[9]
	MCPWM0	0x4001_0C00	0x4001_0CFF	256B	SYS_CLK_FEN[10] SYS_SFT_RST[10]
	GPIO	0x4001_0D00	0x4001_0EFF	512B	SYS_CLK_FEN[11] SYS_SFT_RST[11]
	CRC0	0x4001_0F00	0x4001_0FFF	256B	SYS_CLK_FEN[16] SYS_SFT_RST[16]
	UART0	0x4001_1000	0x4001_10FF	256B	SYS_CLK_FEN[14] SYS_SFT_RST[14]
	UART1	0x4001_1100	0x4001_11FF	256B	SYS_CLK_FEN[15] SYS_SFT_RST[15]
	DMA0	0x4001_1200	0x4001_12FF	256B	SYS_CLK_FEN[18] SYS_SFT_RST[18]
	CAN0	0x4001_1300	0x4001_14FF	512B	SYS_CLK_FEN[19] SYS_SFT_RST[19]
	SIF0	0x4001_1500	0x4001_15FF	256B	SYS_CLK_FEN[20] SYS_SFT_RST[20]

	Resv.	0x4001_1600	0x4001_16FF	256B	Same as bus
	AON	0x4001_1700	0x4001_17FF	256B	Same as bus
	CL0	0x4001_1800	0x4001_18FF	256B	SYS_CLK_FEN[21] SYS_SFT_RST[21]
	DSP0	0x4001_2000	0x4001_3FFF	8kB	SYS_CLK_FEN[17] SYS_SFT_RST[17]

4 Interrupt

The nested vectored interrupt controller is inside the CPU core. When an interrupt event occurs, it will notify the CPU to suspend the execution of the main program, and enter the interrupt service function according to priority setting.

The LKS32MC07x support up to 25 external interrupt sources.

It supports up to four interrupt priority levels for programming.

Table 4-1 Interrupt Number List

Interrupt No.	Description	Interrupt No.	Description
-14	NMI		
-13	HardFault		
-12	Reserved		
-11			
-10			
-9			
-8			
-7			
-6			
-5	SVCall		
-4	Reserved		
-3			
-2	PendSV		
-1	SysTick		
0	TIMER0_IRQn	16	MCPWM0_IRQn
1	TIMER1_IRQn	17	MCPWM1_IRQn
2	TIMER2_IRQn	18	DMA0_IRQn
3	TIMER3_IRQn	19	CAN0_IRQn
4	QEP0_IRQn	20	SIF0_IRQn
5	QEP1_IRQn	21	System wake-up interrupt, WAKE_IRQn
6	I2C0_IRQn	22	Software interrupt, SW_IRQn
7	SPI0_IRQn	23	AVDD Voltage low, PWRDN_IRQn
8	GPIO_IRQn	24	CL0_IRQn
9	HALL0_IRQn	25	Reserved
10	UART0_IRQn	26	Reserved
11	UART1_IRQn	27	Reserved
12	DSP0_IRQn	28	Reserved
13	CMP_IRQn	29	Reserved
14	ADC0_IRQn	30	Reserved
15	ADC1_IRQn	31	Reserved

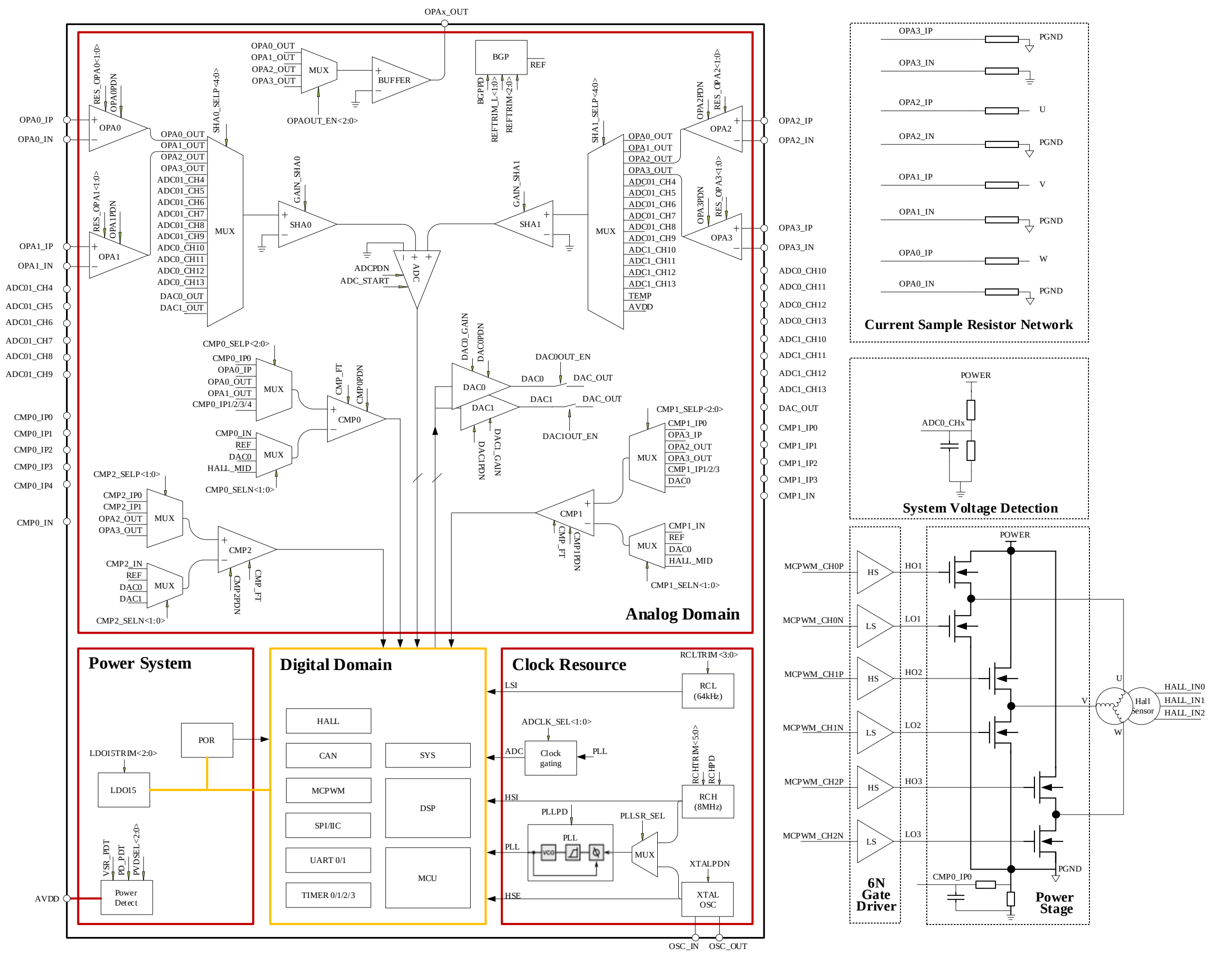
5 Analog circuit

5.1 Introduction

The analog circuit contains the following modules:

- Built-in 2 12bit SAR ADC, 3Msps sampling rate, and each sampling circuit supports up to 16 channels, including 4 OPA outputs and 10 external ADC channels for a total of 14 optional ADC channel signals.
- Built-in 4 operational amplifiers. Differential PGA mode is available.
- Built-in 3 comparators. Hysteresis mode is available.
- Built-in 12bit digital-to-analog converter (DAC)
- ± 2 °C built-in temperature sensor
- Built-in high-precision reference sources

The interrelationship between the modules and the control register of each module (see the "Analog Register Table" below for register description) are shown in the figure below.



* ADC 01_CH4 ~ ADC 01_CH9 are common channels for ADC0 and ADC 1

Figure 5-1 Functional Block Diagram of Analog Circuit

5.1.1 Power management system

The power management system is composed of LDO15 module and power-on/power-off reset module (POR), power detection module (PVD).

The chip is powered by single 2.5~5V power supply to save off-chip power costs, and all internal digital circuits and PLL modules are powered by an internal LDO15.

The LDO automatically turns on after power-on, without software configuration.

The LPOR module monitors the voltage of the LDO15 and provides a reset signal for the digital circuit to avoid faults occurred when the voltage of LDO15 is lower than 1.25V (e.g.: power-on or power-off).

The PVD module monitors the 5V input power. If it is below a certain set threshold, it will remind the MCU by sending an alarm (interrupt) signal. The interrupt reminder threshold can be set to different voltages by the PVDSEL [9:8] registers, and the PVD module can be turned off by setting PD_PDT ="1".When the low-voltage event occurs, the system only generates a low-voltage flag and does not cause a chip reset.

When the voltage is low, a low voltage interrupt will be triggered, and the interrupt number is 23. See Chapter 4 for details.

For the description of PVDSEL[9:8]/ PD_PDT, see the analog register SYS_AFE_REG6.

5.1.2 Clock system

The clock system consists of an internal 32 kHz RC clock, an internal 8 MHz RC clock and a PLL circuit.

The 32kHz RC clock LSI is mainly used for the watchdog module and reset signal filters in the system, and can also be used as the main clock of the MCU. The 8MHz RC clock can be used as the MCU master clock. The PLL provides a clock of up to 96 MHz, which is typically used by MCUs as the system master clock.

Both the 32kHz and 8MHz RC clocks have been through the correct calibration procedure at the factory, and they can achieve ± 5% and ± 1% accuracy respectively at room temperature. The accuracy of 32 kHz RC clock is ± 50% in the range of -40 ~ 105 ℃, and the accuracy of 8 MHz RC clock is ± 1% in this temperature range.

The 8MHz RC clock is turned on by setting RCHPD = '0' (ON by default, turn off when set to "1"). The RC clock requires the BGP voltage reference module to provide the reference voltage and current. Therefore, when the RC clock is turned on, the BGP module is actually turned on(BGPPD=’0’). By default, both the 8 MHz RC clock and the BGP block are turned on when the chip is powered on. The 32kHz RC clock is always on and cannot be turned off.

The PLL multiplies the 8MHz RC clock to provide a higher working clock for the MCU, ADC and other modules. The maximum clock for the MCU and PWM modules is 96 MHz, and the maximum clock for the ADC module is 48 MHz.

PLL is turned on by setting PLLPDN = ’1’ (OFF by default, turn on when set to '1'). Before turning on the PLL module, the BGP (Bandgap) module should be turned on first. After the PLL is turned on, it needs a settling time of 6us to achieve a stable frequency output. When the chip is powered on, the RCH clock and BGP module are both turned on. PLL is OFF by default and enabled by software.

The crystal oscillator circuit has a built-in amplifier but no oscillator capacitor. Connect a crystal between IO OSC_IN and OSC_OUT, and a 15pF capacitor to ground at each pin of OSC_IN and OSC_OUT. Set XTALPDN = ’1’ to start the oscillation. A 8MHz square wave signal could be connected to OSC_IN and take the place of oscillator.

BGPPD/PLLPDN is described in analog register SYS_AFE_REG5

5.1.3 Bandgap Voltage Reference (BGP)

Bandgap Voltage Reference (BGP) provides reference voltage and current for ADC, DAC, RC clock, PLL, temperature sensor, operational amplifier, comparator and FLASH. Turn on the Bandgap before using any of the above modules.

When the chip is powered on, the BGP module is turned on automatically. The voltage reference is turned on by setting BGPPD = ’0’. From OFF to ON, BGP needs about 6us to stabilize. BGP output voltage is about 1.2V, and accuracy is ± 0.5%

For the description of BGPPD, see the analog register SYS_AFE_REG5

5.1.4 Analog-to-digital Converter (ADC)

Refer to Chapter 12 ADC.

5.1.5 Operational Amplifier (OPA)

The 4-channel of rail-to-rail OPAs is integrated, with a built-in feedback resistor, and an external resistor R_0 connected to the signal source on the pin. The resistance of feedback resistors R2: R1 can be adjusted by register RES_OPAx [1:0] to achieve different gains.

For the description of RES_OPAx<1:0>, see the analog register SYS_AFE_REG0

The schematic diagram of the amplifier is as follows:

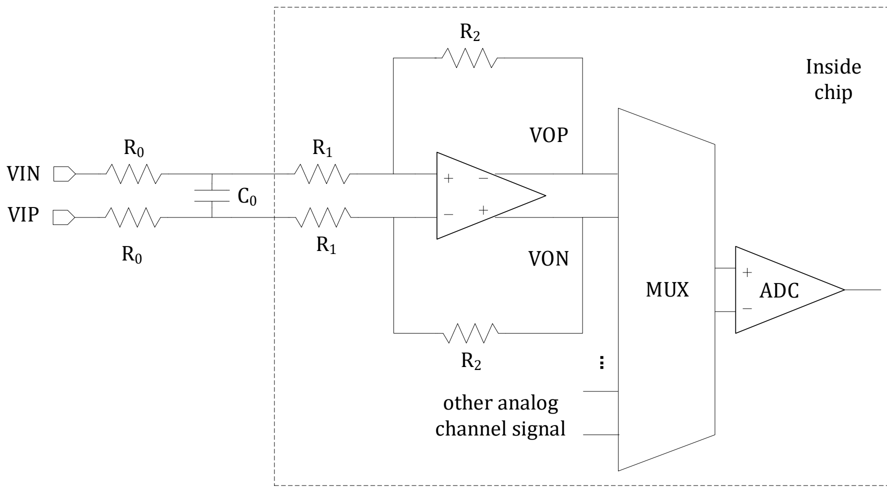


Figure 5-2 Amplifier Block Diagram

The two R0 in the figure are the resistance of the external resistor, so the resistance must be equal. The close-loop gain of OPA is $R_2/(R_1+R_0)$.

For the application of MOS resistance direct sampling, it is recommended to connect an external resistance of >20kΩ to reduce the current flowing into the chip pin to avoid the voltage signal rises to tens of volts when lower MOS tube is turned off, and the upper tube is turned on;

For the application of shunt resistance sampling, it is recommended to connect an external resistor of 100 ~ 2KΩ. C_0 is the signal filter capacitor; it forms a first-order RC filter circuit with R_0, and the specific resistance of R_0 can be determined based on the filter constant of R_0* C_0. The filtering is not necessary if the Signal to Noise Ratio (SNR) is small, or C_0 can be omitted when the signal requires a large bandwidth (faster response speed).

The OPA can select one of the output signals of the 4-channels amplifiers by setting OPAOUT_EN <2:0>, and send it to the P2.7 IO port through a buffer for measurement (see the corresponding relationship in the datasheet 'Pin Function Description"). Because of the BUFFER, the OPAMP is able to send one output signal in the normal working mode. The signal sent to the IO port is the positive output signal of the OPA (the detailed block diagram and description of the corresponding amplifier can be seen in the Wiki: the difference between the op amp difference and single-ended operation mode)

For the description of OPAOUT_EN<2:0>, see the analog register SYS_AFE_REG2.

When the chip is powered on, the OPA module is OFF by default. It can be turned on by setting OPAxPDN = 1 (x=0, 1, 2, 3), and turn on the BGP module before turning on the amplifier.

For the description of OPAxPDN, see the analog register SYS_AFE_REG5.

For built-in clamp diodes are integrated between the positive and negative OPA inputs, the motor phase line could be directly connected to the OPA input through a matching resistor, thereby simplifying the external circuit for MOSFET current sampling.

5.1.6 Comparator

Built-in 3-channel rail-to-rail comparators with programmable comparator speed, hysteresis voltage, and signal source.

The comparison delay can be set to < 50nS/200nS using Register CMP_FT, and the hysteresis voltage can be set to 20mV/0mV by CMP_HYS.

The signal sources of the positive and negative input of the comparator can be set by the registers CMP2_SELP[1:0]/ CMP1_SELP[2:0]/ CMP0_SELP[2:0] and CMPx_SELN[1:0] (x=0/1/2 for comparators CMP0/CMP1/CMP2).

It should be noted that The BEMFx_MID signals at the negative input terminals of the two comparators are the average of the CMPx_IP1/CMPx_IP2/CMPx_IP3 signals at the positive input terminals of the comparator. The specific connection method is shown in Fig. 5-3. Among them, the resistance R=8.2k ohms, the switch in the picture will be turned on only after the negative input signal of the comparator is selected as BEMFx_MID, otherwise the switches are in the off state.

BEMFx_MID is mainly used for BLDC square wave mode control, the virtual motor phase line center point voltage, used for back-EMF zero-crossing detection. After the three phase lines are divided, connect to CMPx_IP1, CMPx_IP2, and CMPx_IP3 respectively. The MCU controls the negative end of the comparator to select BEMFx_MID, and the multiplexer at the positive end of the comparator selects CMPx_IP1, CMPx_IP2, and CMPx_IP3 in a time-division multiplexing manner to compare the zero-crossing point of the back EMF.

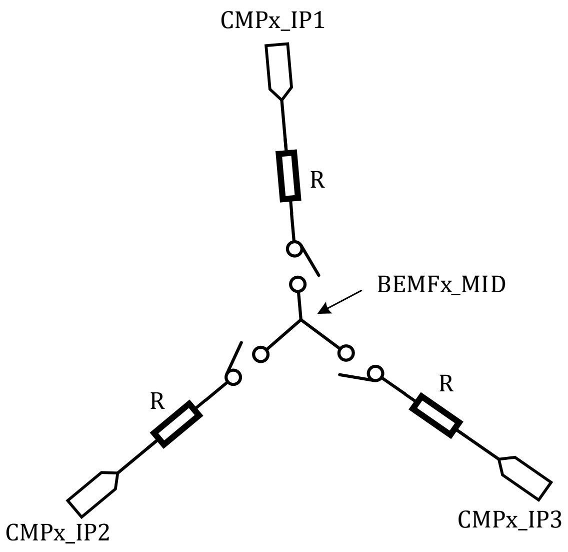


Figure 5-3 BEMFx_MID Signal

The output of the comparator can be read through the register SYS_AFE_CMP.

CMP_FT is described in analog register SYS_AFE_REG1.

CMPx_SELN<1:0>/CMPx_SELP<2:0>/CMP_HYS are described in analog register SYS_AFE_REG2 / SYS_AFE_REG3.

When the chip is powered on, the comparator module is OFF by default. The comparator is turned on by setting CMPxPDN=1 (x=0,1,2), and turn on the BGP module before turning on the comparator.

CMPxPDN is described in analog register SYS_AFE_REG5.

5.1.7 Temperature sensor

The chip has a built-in temperature sensor with an accuracy of 2°C in the range of -40°C to 85°C. The accuracy is 3°C in the range of 85°C to 105°C.

When measuring, an internal reference should be selected. Choosing an external reference will cause a significant deviation in the results.

The operating temperature of chips will be corrected before leaving the factory, and the corrected value is saved in the flash info area.

When the chip is powered on, the temperature sensor module is OFF by default. Turn on the BGP module before turning on the temperature sensor.

The temperature sensor is turned on by setting TMPPDN = '1', and it takes about 2us to be stable after turning on. Thus, it should be turned on at least 2us ahead before the ADC measures the sensor output.

The temperature sensor signal is connected to channel 14 of the ADC.

For the ADC settings, please refer to Chapter Analog to Digital Converter (ADC)

For the description of TMPPDN,, see the analog register SYS_AFE_REG5.

The typical curve of the temperature sensor is shown in the figure below:

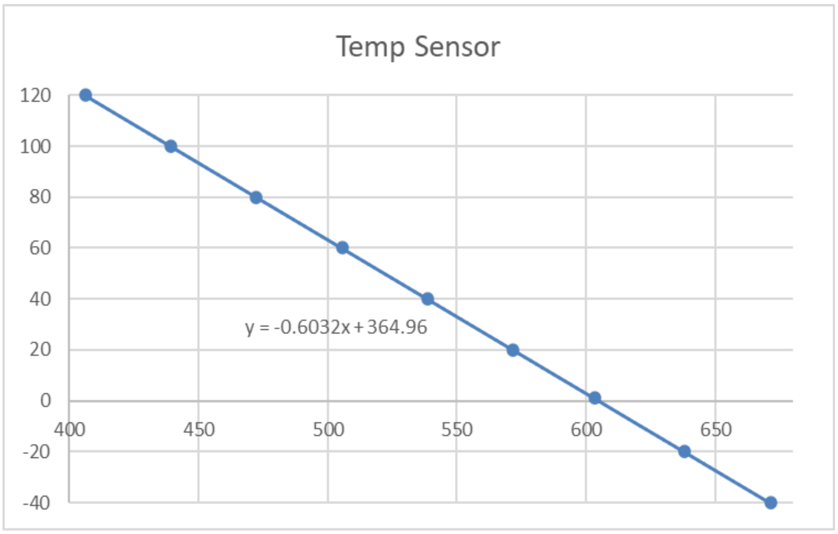


Figure5-4 Temperature Sensor Curve

The X axis in the figure is the ADC value corresponding to the temperature signal of the temperature sensor, and the Y axis is the temperature of the sensor. When measuring temperature, configure the sensor-related registers according to the above requirements, and put the ADC value as X into the formula after the value obtained:

y=-0.6032x+364.96

The calculated value Y is the current temperature.

There are two coefficients in the formula, a = -0.6032, b = 364.96, and the value of the coefficient b differs from different chips. The temperature sensor will be calibrated in the factory. Write the coefficient b into the Flash info area, and the address is 0x000014D4. The decimal point of the coefficient b will be shifted to the right by one digit (multiplied by 10) and stored in the info area. The second digit after the decimal point will not be saved.

For the convenience of customers, the coefficient a will also be stored in the Flash info area, and the address is 0x000014D0. The decimal point of the coefficient a will be shifted to the right by four digits (multiplied by 10000) and stored in the info area.

In actual use, the coefficient a and b should be read first from the address in different Flash info area, and then put the measured ADC value into the formula to calculate the current temperature. The unit is degrees Celsius. When calculating, pay attention to the digits of the decimal points of coefficient a and b, that is, the coefficient a should be divided by 10000, and the coefficient b should be divided by 10.

Please note that the above calculation formula is implemented based on ADC right-aligned mode. If the calculation adopts left-aligned mode, the ADC sampling value should be shifted right by four bits before putting into the above formula.

5.1.8 DAC module

The chip contains two 12-bit DACs, and the maximum range of the output signal can be set to 1.2

V/4.85 V through the DAC0_GAIN and DAC1_GAIN registers.

The DAC0 can be output via IO port P0.0 by setting register DAC0OUT_EN = 1, which can drive a load resistance of over 5kΩ and a load capacitance of 50pF. The DAC1 can be output via IO port P0.0 by setting register DAC1OUT_EN = 1. Normally, DAC0 and DAC1 are not output at the same time to avoid signal contention.

The maximum output bit rate of DAC is 1MHz.

By default, the DAC block is off when the chip is powered on. DAC0 can be turned on by setting DAC0PDN = 1, and DAC1 can be turned on by setting DAC1PDN = 1. Before turning on the DAC module, you need to turn on the BGP module.

The input digital signal register of DAC is SYS_AFE_DACx, low 12-bit is effective. The signal range is 0x000 ~ 0xFFF. The zero analog output corresponding to the signal range 0x000 is 0V, and the full-scale analog output corresponding to 0xFFF is DAC_{fs}. As mentioned above, the value of DAC_{fs} can be set by the DACx_GAIN register. The analog signal amplitude corresponding to each gear signal (LSB) is $\frac{DAC_{fs}}{4096}$. If the digital value of SYS_AFE_DACx is Din, the analog signal of the DAC output corresponding to the digital signal is$\frac{DAC_{fs}}{4096} * Din$.

There are manufacturing deviations in the DAC by using different chips. Thus, the DAC has a calibration hardware module to offset the deviations. The DAC output formula is: y=ax+b. x is the value filled in SYS_AFE_DACx (ideal digital quantity). a is the value filled in SYS_AFE_DACx_AMC register and b is the value filled in SYS_AFE_DACx_DC register. The hardware multiplies and adds SYS_AFE_DACx, SYS_AFE_DACx_AMC and SYS_AFE_DACx_DC to obtain the corrected digital quantity, and sends it to the DACn input terminal, so that the final analog value of the DACn output is the ideal digital quantity. After the system is powered on, the calibration value of 4.85V is loaded by default. If it is changed to another range, the software will read the flash info area and reload it into the corresponding register. Where X = 0, 1, corresponding to DAC0 and DAC1.

DAC0:

Address 0x00001450 is the parameter a of the 4.85V range, and address 0x00001454 is the parameter b of the 4.85V range.

Address 0x00001458 is the parameter a of the 1.2V range, and the address 0x0000145C is the parameter b of the 1.2V range.

DAC1:

Address 0x00001460 is the parameter a of the 4.85V range, and address 0x00001464 is the parameter b of the 4.85V range.

Address 0x00001468 is the parameter a of the 1.2V range, and the address 0x0000146C is the parameter b of the 1.2V range.

Except for external module usage via IO port, the analog signal output by the DAC can also be used as a reference signal for the comparator by connecting the configuration register to the negative side of the two-channel CMP inside the chip.. See the section Comparator (CMP) for details.

DAC0OUT_EN/DAC1OUT_EN are described in Analog RegisterSYS_AFE_REG3

DAC 0_GAIN/DAC 1_GAIN are described in Analog RegisterSYS_AFE_REG1

The DAC0PDN/DAC1PDN are described in Analog RegisterSYS_AFE_REG5

The SYS_AFE_DAC0/SYS_AFE_DAC1 are described in the registerSYS_AFE_DAC0, SYS_AFE_DAC1

5.2 Register

5.2.1 Address assignment

Address space of 0x40000010 ~ 0x40000028 is a register open to users, among which the reserved registers (Res) must all be set as 0 (it will be reset to 0 after power on). Other registers will be configured according to the actual working situation.

The base address of the emulation register is 0x 4000 0000.

Table 5-1 System Control Register

Name	Offset	Explain
SYS_AFE_INFO	0x04	Chip version information register
SYS_AFE_DBG	0x08	Debug register
SYS_AFE_REG0	0x10	Analog register 0
SYS_AFE_REG1	0x14	Analog register 1
SYS_AFE_REG2	0x18	Analog register 2
SYS_AFE_REG3	0x1C	Analog register 3
SYS_AFE_REG4	0x20	Analog register 4
SYS_AFE_REG5	0x24	Analog register 5
SYS_AFE_REG6	0x28	Analog register 6
SYS_AFE_REG7	0x2C	Analog register 7
SYS_AFE_DAC_CTRL	0x5C	DAC Control Register
SYS_AFE_DAC0	0x60	DAC0 digital register
SYS_AFE_DAC1	0x64	DAC1 digital register
SYS_AFE_DAC0_AMC	0x68	DAC0 Digital Gain Correction Register
SYS_AFE_DAC0_DC	0x6C	DAC0 digital DC offset register
SYS_AFE_DAC1_AMC	0x70	DAC1 Digital Gain Correction Register
SYS_AFE_DAC1_DC	0x74	DAC1 digital DC offset register

5.2.2 SYS_AFE_INFO

Address: 0x4000 0004

Reset value: different according to wafer version

Table 5-2 Chip Version Information Register(SYS_AFE_INFO)

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

Location	Bit name	Description
[31:8]		Not used
[7:4]	Series	Chip model information
[3:0]	Version	Chip version information

This register is user independent.

5.2.3 SYS_AFE_DBG

Address: 0x4000_0008

Reset value:0x0

表 5-3 SYS_AFE_DBG

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

Location	Bit name	Explain
[31:8]		Not used
[15]	PWR_WEAK	The power supply is lower than the power failure monitoring threshold
[14:0]		Not used

The PWR_WEAK flag is set when the power supply voltage is lower than the threshold set by the power failure monitoring circuit, and a CPU power failure interrupt (interrupt number 23) is generated. When the power supply is restored and the value is higher than the threshold, it is cleared. This flag is a read-only bit and cannot be cleared by software.

For details about how to use power failure monitoring, see 5.1.1 Power Management System.

5.2.4 SYS_AFE_REG0 Analog Configuration 0

Address: 0x4000 0010

Reset value: 0x0

Table 5-4 AFE Register 0 (SYS_AFE_REG0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.						IT_OPA		RES_OPA3		RES_OPA2		RES_OPA1		RES_OPA0	
RW						RW		RW		RW		RW		RW	
0						0		0		0		0		0	

Location	Bit name	Explain
[31:16]		Not used
[15:10]		Reserved bit, must be 0

[9:8]	IT_OPA	Bias current adjustment for OPA, default configuration 00: ×1, recommended if RES_OPAx != 00 01: ×1.2 10: ×1.5, recommended if RES_OPAx == 00 11: ×2
[7:6]	RES_OPA3	OPA3 feedback resistor 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[5:4]	RES_OPA2	OPA2 feedback resistor 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[3:2]	RES_OPA1	OPA1 feedback resistor 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k
[1:0]	RES_OPA0	OPA0 feedback resistor 00: 320k:10k 01: 160k:10k 10: 80k:10k 11: 40k:10k

5.2.5 SYS_AFE_REG1 Analog Configuration 1

Address: 0x4000 0014

Reset value: 0x0

Table 5-5 SYS_AFE_REG1 Analog Configuration 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.				DAC1_GAIN	DAC0_GAIN	Res.	REF2VDD	Res.			CMP_FT
RW	RW	RW	RW	RW				RW	RW	RW	RW	RW			RW
0	0	0	0	0				0	0	0	0	0			0

Location	Bit name	Description
[31:16]		Unused
[15]	Reserved	Reserved bit, must be 0
[14]	Reserved	Reserved bit, must be 0
[13]	Reserved	Reserved bit, must be 0
[12]	Reserved	Reserved bit, must be 0
[11:8]	Reserved	Reserved bit, must be 0
[7]	DAC1_GAIN	DAC1 output configuration 0: full scale is 4.85V; 1: 1.2 V full scale
[6]	DAC0_GAIN	DAC0 output configuration

		0: full scale is 4.85V; 1: 1.2 V full scale
[5]	Reserved	Reserved bit, must be 0
[4]	REF2VDD	ADC Reference Select 1: Use the external input power supply as the ADC reference; 0: Use default internal REF 2.4V as ADC reference
[3:1]	Reserved	Reserved bit, must be 0
[0]	CMP_FT	Enable comparator for quick comparison 1: Enabled, the comparison speed of the comparator is less than 50ns 0: Disabled, the comparison speed of the comparator is less than 200ns.

When REF2VDD is set to 1 and an external power supply is used as the ADC reference, if ADCx GAIN=0 is selected, the range changes to 7.5V. If ADC GAIN=1 is selected, the range changes to 15V.

In the DAC output gear configuration, the 4.85V range is saturated in advance when approaching 4.45V. The 1.2V range is pre-saturated when approaching 1.13V.

5.2.6 SYS_AFE_REG2 Analog Configuration 2

Address: 0x4000 0018

Reset value: 0x0

Table 5-6 SYS_AFE_REG2 Analog Configuration 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP2_SELP		Res.					XTRSEL	Res.			REF_AD_EN	LDOOUT_EN	OPAOUT_EN		
RW		RW					RW	RW			RW	RW	RW		
0		0					0	0			0	0	0		

Location	Bit name	Description
[31:16]		Not used
[15:14]	CMP2_SELP	Positive Selection of Comparator 2 Signal 00: Connect to CMP2_IP0 01: Connect to CMP2_IP 1 10: Connect to OPA2_OUT 11: Connect to OPA3_OUT
[13:9]	Reserved	Reserved bit, must be 0
[8]	XTRSEL	Crystal start-up circuit resistance adjustment. XTRSEL = 1: P terminal resistance is doubled to slow down crystal oscillating
[7:5]	Reserved	Reserved bit, must be 0
[4]	REF_AD_EN	ADC Reference output to P2.7
[3]	LDOOUT_EN	LDO15 output to P2.7

[2:0]	OPAOUT_EN	Enable OPAx output signal to IO port P2.7
		000: no output;
		001: Output OPA0 signal to IO port;
		010: Output OPA1 signal to IO port;
		011: Output OPA2 signal to IO port;
		100: Output OPA3 signal to IO port;
		101 ~ 111: Prohibited

5.2.7 SYS_AFE_REG3 Analog Configuration Register 3

Address: 0x4000 001C

Reset value: 0x0

Table 5-7 SYS_AFE_REG3 Analog Configuration Register 3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC1OUT_EN	CMP1_SELP			DAC0OUT_EN	CMP0_SELP			CMP_HYS	Res.	CMP1_SELN		CMP0_SELN		CMP2_SELN	
RW	RW			RW	RW			RW	RW	RW		RW		RW	
0	0			0	0			0	0	0		0		0	

Location	Bit name	Description
[31:16]		Not used
[15]	DAC1OUT_EN	DAC1 Output and IO Enable 0: Disabled 1: Enable output via IO P0[0]
[14:12]	CMP1_SELP	Positive Selection of Comparator 1 Signal 000: CMP1_IP0 001: OPA3_IP 010: OPA2_OUT 011: OPA3_OUT 100: CMP1_IP1 101: CMP1_IP2 110: CMP1_IP3 111: DAC0 output Note: The above are pin names except AVSS/OPA2_OUT/OPA3_OUT. For the definition of the pins, please refer to the specific section in the DATASHEET.
[11]	DAC0OUT_EN	DAC0 Output and IO Enable 0: Disabled 1: Enable output via IO P0[0]
[10:8]	CMP0_SELP	Positive Selection of Comparator 0 Signal 000: CMP0_IP0 001: OPA0_IP 010: OPA0_OUT 011: OPA1_OUT 100: CMP0_IP1 101: CMP0_IP2 110: CMP0_IP3 111: CMP0_IP4 Note: The above are pin names except AVSS/OPA2_OUT/OPA3_OUT. For the definition of the pins, please refer to the specific section in the DATASHEET.
[7]	CMP_HYS	Comparator hysteresis selection with default configuration 0: 20mv; 1: 0mv
[6]	Reserved	Reserved bit, must be 0

[5:4]	CMP1_SELN	Negative Selection of Comparator 1 Signal 00: CMP1_IN 01: REF 10: DAC0 output 11: BEMF1_MID Note: The above CMP1_IN is the pin name, For the pin definition, please refer to the specific section in the DATASHEET; REF is the 1.2V BANDGAP reference inside the chip; The DAC0 output is the analog signal output by the DAC module inside the chip; BEMF1_MID is the average value obtained by connecting the CMP1_IP1, CMP1_IP2, and CMP1_IP3 signal through a star connection.
[3:2]	CMP0_SELN	Comparator 0 signal negative terminal selection 00: CMP0_IN 01: REF 10: DAC0 output 11: BEMF0_MID Note: The above CMP0_IN is the pin name, For the pin definition, please refer to the specific section in the DATASHEET; REF is the 1.2V BANDGAP reference inside the chip; The DAC0 output is the analog signal output by the DAC module inside the chip; BEMF0_MID is the average value obtained by connecting the CMP1_IP1, CMP1_IP2, and CMP1_IP3 signal through a star connection.
[1:0]	CMP2_SELN	Comparator 0 signal negative terminal selection 00: CMP2_IN 01: REF 10: DAC0 output 11: DAC1 output

5.2.8 SYS_AFE_REG4 Analog Configuration 4

Address: 0x4000 0020

Reset value: 0x0

Table 5-8 SYS_AFE_REG4 Analog Configuration 4

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

Location	Bit name	Description
[31:16]		
[15]		Not used
[14]	Reserved	Reserved bit, must be 0
[13]		Not used
[12]	Reserved	Reserved bit, must be 0
[11:8]		Not used
[7]		Not used
[6]	Reserved	Reserved bit, must be 0
[5]		Not used
[4]	Reserved	Reserved bit, must be 0

[3:0]		Not used

5.2.9 SYS_AFE_REG5 Analog Configuration 5

Address: 0x4000 0024

Reset value: 0x0

Table 5-9 SYS_AFE_REG5 Analog Configuration 5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLPDN	XTALPDN	TMPPDN	DAC0PDN	CMP2PDN	RCHPD	DAC1PDN	BGPPD	CMP1PDN	CMP0PDN	OPA3PDN	OPA2PDN	OPA1PDN	OPA0PDN	Res.	ADCPDN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]	PLLPDN	PLL Off Control 0: Turn off PLL (default) 1: Turn on PLL
[14]	XTALPDN	Crystal Oscillator Enable 0: Off (default) 1: On
[13]	TMPPDN	Temperature Sensor Enable 0: Off 1: On
[12]	DAC0PDN	DAC0 Enable 0: Off 1: On
[11]	CMP2PDN	CMP2 Enable 0: Off 1: On
[10]	RCHPD	RCH Clock Off 0: On 1: Off
[9]	DAC1PDN	DAC1 Enable 0: Off 1: On
[8]	BGPPD	BGP Enable 0: On 1: Off
[7]	CMP1PDN	CMP1 Enable 0: Off 1: On
[6]	CMP0PDN	CMP0 Enable 0: Off 1: On
[5]	OPA3PDN	OPA3 Enable 0: Off 1: On

[4]	OPA2PDN	OPA2 Enable 0: Off 1: On
[3]	OPA1PDN	OPA1 Enable 0: Off 1: On
[2]	OPA0PDN	OPA0 Enable 0: Off 1: On
[1]	Reserved	Reserved bits, all must be '0'
[0]	ADCPDN	ADC Enable 0: Off 1: On

If SYS_CLK_CFG selects PLL clock, PLLPDN is hardware-controlled, and software configuration of PLLPDN to disable PLL is invalid. Disabling PLL requires PLLPDN=0, and SYS_CLK_CFG does not select PLL as the chip master clock. Both conditions must be satisfied.

Similarly, if SYS_CLK_CFG selects HRC clock, then RCHPD is controlled by hardware. It is invalid to configure RCHPD to disable RCH directly by software. Turning off PLL requires RCHPD=1 and the chip goes to sleep.

If the chip master clock is a PLL clock and the HRC is a PLL reference clock, then the RCH is also controlled by the hardware.

Since RCH and PLL depend on BGP, BGPPD is also hardware-controlled. When RCH or PLL is used on a chip, it is invalid to configure BGPPD in software to disable BGP. To disable BGP, disable the PLL and RCH in sequence and the chip goes to sleep.

5.2.10 SYS_AFE_REG6 Analog Configuration 6

Address: 0x4000 0028

Reset value: 0x0

Table 5-10 SYS_AFE_REG6 Analog Configuration 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLSR_SEL	Res.					PVDSEL		Res.				Res.		VSR_PDT	PD_PDT
RW	RW					RW		RW				RW		RW	RW
0	0					0		0				0		0	0

Location	Bit name	Description
[31:16]		Not used
[15]	PLLSR_SEL	PLL clock source selection 0: Use RCH as the input clock source; 1: Use XTAL OSC as the input clock source
[14:10]	Reserved	Reserved bit, must be 0
[9:8]	PVDSEL	Power failure monitoring threshold V_{th} selection

		00: 4.00V 01: 3.75V 10: 3.50V 11: 3.25V (SYS_AFE_INFO.VERSION<=3) 2.50V (SYS_AFE_INFO.VERSION=4) The above threshold is set value when VSR_PDT = 0, that is, the setting value when the low-power reference source V_{BGP} is selected; If VSR_PDT = 1 and the DAC output is selected as the reference, the threshold is adjusted accordingly to V_{th}* DAC0_OUT/V_{BGP}.
[7:2]	Reserved	Reserved bit, must be 0
[1]	VSR_PDT	Brown-out detection reference selection where the low power reference V_{BGP} is around 1.3V and the DAC output is software configurable 0: Select low-power reference source V_{BGP}; 1: Select DAC0 output
[0]	PD_PDT	Power Down Supply Voltage Detection Circuit 0: On 1: Off

5.2.11 SYS_AFE_REG7 Analog Configuration Register 7

Address: 0x4000 002C

Reset value: 0x0

Table 5-11 SYS_AFE_REG7 Analog Configuration Register 7

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

Location	Bit name	Description
[31:16]		Not used
[15:5]	Reserved	Reserved bit, must be 0
[4]	ADCLKSEL	ADC Clock Frequency Selection 0: 48MHz 1: 24MHz
[3:0]	Reserved	Reserved bits, all must be '0'

5.2.12 SYS_AFE_DAC_CTRL DAC control register

Address: 0x4000_005C

Reset value: 0x0

Table 5-12 SYS_AFE_DAC_CTRL DAC control register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DAC1_STEP			TIMER3_TRIG_DAC1	TIMER2_TRIG_DAC1	TIMER1_TRIG_DAC1	TIMER0_TRIG_DAC1		DAC0_STEP			TIMER3_TRIG_DAC0	TIMER2_TRIG_DAC0	TIMER1_TRIG_DAC0	TIMER0_TRIG_DAC0
	RW			RW	RW	RW	RW		RW			RW	RW	RW	RW
	0			0	0	0	0		0			0	0	0	0

Location	Bit name	Description
[31:15]		Not used
[14:12]	DAC1_STEP	SYS_AFE_DAC1 step value, 3 bit signed number, range -4 ~ 3
[11]	TIMER3_TRIG_DAC1	TIMER3 zero-crossing event triggers DAC1 step enable. Highly effective.
[10]	TIMER2_TRIG_DAC1	TIMER2 zero-crossing event triggers DAC1 step enable. Highly effective.
[9]	TIMER1_TRIG_DAC1	TIMER1 zero-crossing event triggers DAC1 step enable. Highly effective.
[8]	TIMER0_TRIG_DAC1	TIMER0 zero-crossing event triggers DAC1 step enable. Highly effective.
[7]		Not used
[6:4]	DAC0_STEP	SYS_AFE_DAC0 step value, 3 bit signed number, range -4 ~ 3
[3]	TIMER3_TRIG_DAC0	TIMER3 zero-crossing event triggers DAC0 step enable. Highly effective.
[2]	TIMER2_TRIG_DAC0	TIMER2 zero-crossing event triggers DAC0 step enable. Highly effective.
[1]	TIMER1_TRIG_DAC0	TIMER1 zero-crossing event triggers DAC0 step enable. Highly effective.
[0]	TIMER0_TRIG_DAC0	TIMER0 zero-crossing event triggers DAC0 step enable. Highly effective.

The DACx allows the value of the SYS_AFE_DACx to be updated using the zero-crossing event of the Timer as a trigger. Each time the trigger occurs, the setting of the DAC is stepped (incremented or decremented) from the previous value. The step value is set by the SYS_AFE_CTRL. DACx_STEP, and x is 0/1.

5.2.13 SYS_AFE_DAC0 DAC0 Digital Register

Address: 0x4000 0060

Reset value: 0x0

Table 5-13 SYS_AFE_DAC0 DAC0 digital register

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

Location	Bit name	Description
[31:12]		Not used
[11:0]	DAC_IN	DAC Digital Input to be Converted

5.2.14 SYS_AFE_DAC1 DAC1 digital register

Address: 0x4000 0064

Reset value: 0x0

Table 5-14 SYS_AFE_DAC1 DAC1 digital register

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

Location	Bit name	Description
[31:12]		Not used
[11:0]	DAC_IN	DAC Digital Input to be Converted

5.2.15 SYS_AFE_DAC0_AMC DAC0 Gain Correction Register

Address: 0x4000 0068

Reset value: 0x0

Table 5-15 SYS_AFE_DAC0_AMC DAC0 Gain Correction Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AMC									
						RW									
						0x200									

Location	Bit name	Description
[31:10]		Not used
[9:0]	AMC	DAC0 gain correction value, 10 bit unsigned fixed-point number, B[9] is integer, B[8:0] is fractional

5.2.16 SYS_AFE_DAC0_DC DAC0 DC offset register

Address: 0x4000 006C

Reset value: 0x0

Table 5-16 SYS_AFE_DAC0_DC DAC0 DC offset register

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

	0

Location	Bit name	Description
[31:10]		Not used
[9:0]	DC	DAC0 DC offset, 10 bit signed, B[9] is the sign bit

5.2.17 SYS_AFE_DAC1_AMC DAC1 Gain Correction Register

Address: 0x4000 0070

Reset value: 0x0

Table 5-17 SYS_AFE_DAC1_AMC DAC1 Gain Correction Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AMC									
						RW									
						0x200									

Location	Bit name	Description
[31:10]		Not used
[9:0]	AMC	DAC1 gain correction value, 10 bit unsigned fixed-point number, B[9] is integer, B[8:0] is fractional

5.2.18 SYS_AFE_DAC1_DC DAC1 DC offset register

Address: 0x4000 0074

Reset value: 0x0

Table 5-18 SYS_AFE_DAC1_DC DAC1 DC offset register

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

Location	Bit name	Description
[31:10]		Not used
[9:0]	DC	DAC1 DC offset, 10 bit signed, B [9] is the sign bit

DAC gain correction, the 12-bit DAC value of the analog output is the DAC_raw, and the corrected 12-bit DAC value is the DAC_cali.

DAC_cali = DAC_raw * AMC –DC;

Where AMC is the DAC gain correction factor, which is a 10-bit fixed-point unsigned number, B[9] is an integer, B[8:0] is a decimal fraction, and the size is about 1, for example, AMC = 10'b10_0001_0000 = 1 + 1/32, orAMC = 10’b01_1110_1100 = 1-5/128

DC is DAC DC offset, 10 bit signed integer.

The calculation result of gain correction is truncated and retained, and the final DAC_cali is still a 12-bit integer.

And the value after gain correction and DC offset correction will be saturated, the maximum is 0xfff, and the minimum is 0x000.

It should be noted that the DAC has two output ranges. After the system is powered on, the DAC calibration value of the default gear is loaded. If you switch to other gears, please use the library function provided by the factory.

6 System Control and Clock Reset

6.1 Clock

6.1.1 Clock source

As shown in the table below, the system includes four clock sources. The internal low-speed RC oscillation clock LSI/the internal high-speed RC oscillation clock HSI will not stop vibration

Table 6-1 System clock source

Clock source	Frequency	Source	Error	Explain
LSI/LRC	32kHz	Internal RC oscillator	Full temperature range error < 50%	Internal system clock, used for watchdog module, and filtering and widening of reset signal, and can also be used as the master clock for MCU operation.
HSI/HRC	8MHz	Internal RC oscillator	Full temperature range error < 1%	As a PLL source clock
HSE	8MHz	External crystal	Related to the crystal	External crystal
PLL	96MHz	PLL clock	Depends on PLL reference clock. When using the internal HRC as the PLL reference clock, the full temperature range error is <1%	PLL output clock, with HSI as the reference clock input, output PLL clock as the system master clock after 12 times frequency multiplication.
JTAG/SWD	<10MHz	Debugger	-	JTAG/SWD clock, depending on host computer settings

The SWD clock rate is related to the host computer or downloader settings.

The system can use the internal high-speed RC clock HSI as the reference clock for the PLL. The PLL multiplies the 8 MHz reference clock, HSI, by 12 to 96 MHz.

Typically, after the PLL is divided by n/8, a high-speed clock of 96 MHz × n/8 can be obtained as the system master clock MCLK. The system master clock, MCLK, can be divided by n/8 by the SYS_CLK_CFG. CLK_DIV, and provide 12, 24, and 48 MHz.

When the system is reset, the PLL is turned off by default and the HSI is turned on by default, that is, the HSI clock is selected when the system is powered on, and 8 MHz is used as the main clock of the system to work, so as to ensure that the power consumption of the system at the beginning of power-on is at a low level.

LSI and crystal clock HSE can also be used as the system master clock, which is selected by SYS_CLK_CFG. CLK_SEL[1:0]. The system clock can be switched to LSI clock by setting SYS_CLK_CFG. CLK_SEL[1:0] = 2. At this time, the system works at 32kHz clock, and analog modules such as PLL/HRC/BGP can be turned off to reduce power consumption.

The SYS_CLK_CFG. CLK_DIV is a division factor for the PLL. When the SYS_CLK_CFG. CLK_SEL is not 1, the system clock IHS/HSE or LSI is used as the operation clock, and the SYS_CLK_CFG. CLK_DIV is no longer used for frequency division.

Table 6-2 Divide Configuration with PLL as MCLK Clock

SYS_CLK_CFG	Division factor	Frequency/MHz	Whether it is uniform
0x0101	1/8	12	Yes
0x0111	2/8	24	Yes
0x0155	4/8	48	Yes
0x01FF	8/8	96	Yes

The system high speed clock MCLK is supplied to the peripheral after being switched by the SYS_CLK_FEN register. The I2C clock can be further divided by the SYS_CLK_DIV0 register, and the UART clock can be further divided by the SYS_CLK_DIV2 register.

The clock from the PLL is divided by 2 and sent to the ADC(typical frequency is 48MHz), which is ACLK.

The internal 32kHz RC generates an LSI clock LCLK, which is mainly used for the working clock of the watchdog WDT, as well as part of the system control, reset filtering and broadening, etc. When the system has no excessive working tasks and the power consumption of the system is reduced, the SYS_CLK_CFG. CLK_SEL = 2 can be set to switch the system clock to the LSI.

The crystal clock can be used as the system operating clock or as a reference clock source for the PLL. The system clock can be switched to the crystal clock HSE by setting the SYS_CLK_CFG. CLK_SEL = 3. If you want to use the crystal oscillator clock, you need to enable the crystal oscillator startup circuit by setting the SYS_AFE_REG5.XTALPDN = 1. The crystal oscillator startup circuit is off by default. If you want to use the crystal clock as the PLL reference clock, you need to set the SYS_AFE_REG6. PLLSR_SEL = 1. The PLL default reference clock is the internal 8 MHz HRC clock.

It should be noted that the external crystal may fail to oscillate in some cases. There is a crystal oscillator stop detection circuit inside the chip. When the external crystal stops oscillating, it will automatically switch the module using the crystal oscillator clock to the internal HRC clock.

The working state of the crystal can be read through the SYS_CLK_CFG. HSE_FAIL and the SYS_CLK_CFG. HSE_FAILED.

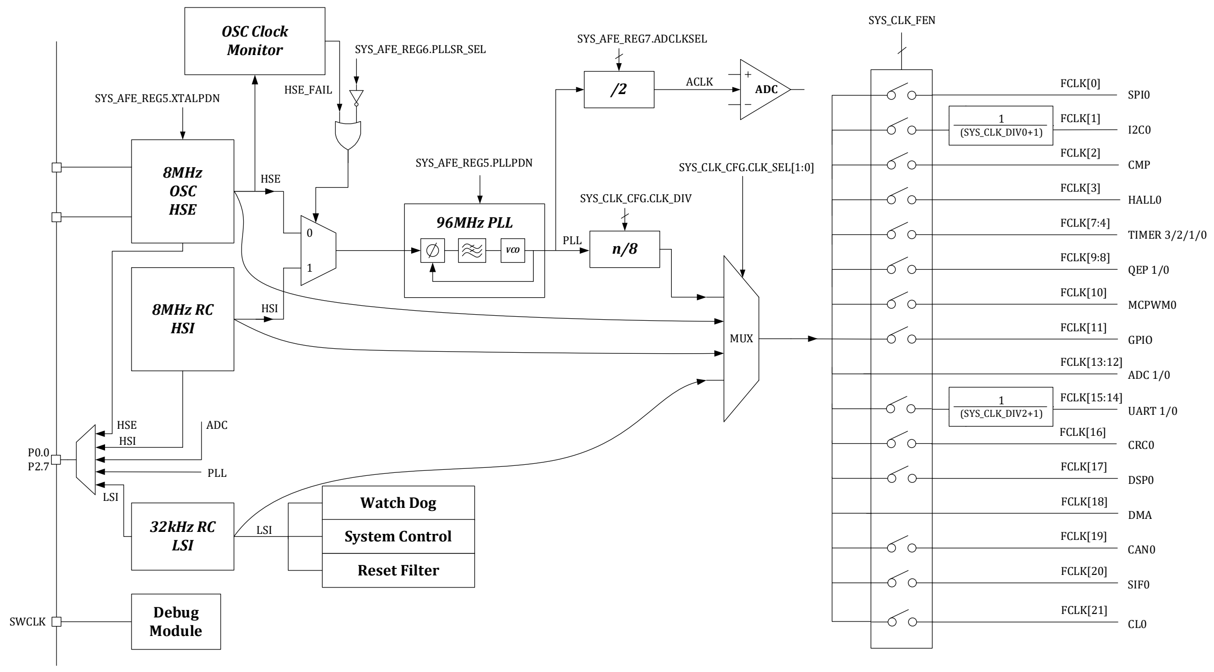


Figure 6-1 Clock architecture

In order to ensure the reliable operation of the system, the clock system has a mechanism to prevent the clock from being shut down by misoperation. For example, when the PLL is used as the master clock for system operation, the PLL circuit itself cannot be turned off, and the HSI used as the PLL reference clock cannot be turned off by software; The 32kHz LSI clock is operational on power-up and cannot be turned off. SWCLK is provided by the debugger, and the frequency can be selected on the upper computer debugging interface.

To facilitate debugging and factory calibration, the high-speed RC clock HSI and the low-speed clock LSI can be output through chip pins by configuring the second function of the GPIO.

6.2 Reset

6.2.1 Reset source

The reset sources of the chip include hardware reset and software reset.

6.2.1.1 Hardware reset

As shown in Table 6-3 Hardware Reset Sources, the system includes three hardware reset sources. The reset generated is a chip global reset. After the reset is generated, the processor program counter (PC) returns to the 0 address, and all registers are restored to the default values.

Table 6-3 System reset source

Name	Source	Description
PORn	Internal power management	Monitor both the 1.5V digital supply and the 5V supply, with a reset occurring when the 1.5V supply is below 1.26V and a reset occurring when the 5V supply is below 2.2V
RSTn	External keys	External key reset circuit, active low

WDTn	Hardware watchdog	If not feed the watchdog, then it will reset the CPU at a regular time, and the reset interval is configurable.

6.2.1.1.1 Hardware reset architecture

As shown in the figure below, PORn comes from the internal analog circuit and RSTn comes from the external key.

WDTn is a signal with a width of 1 LSI clock cycle, and is an internal digital signal. The WDTn signal can be masked in Debug mode by setting SYS_DBG_CFG.

After pre-filtering and broadening the reset signal, a stable and reliable reset signal is output through AND algorithm.

The reset level and scope of the three reset signals are the same, and they are all global reset.

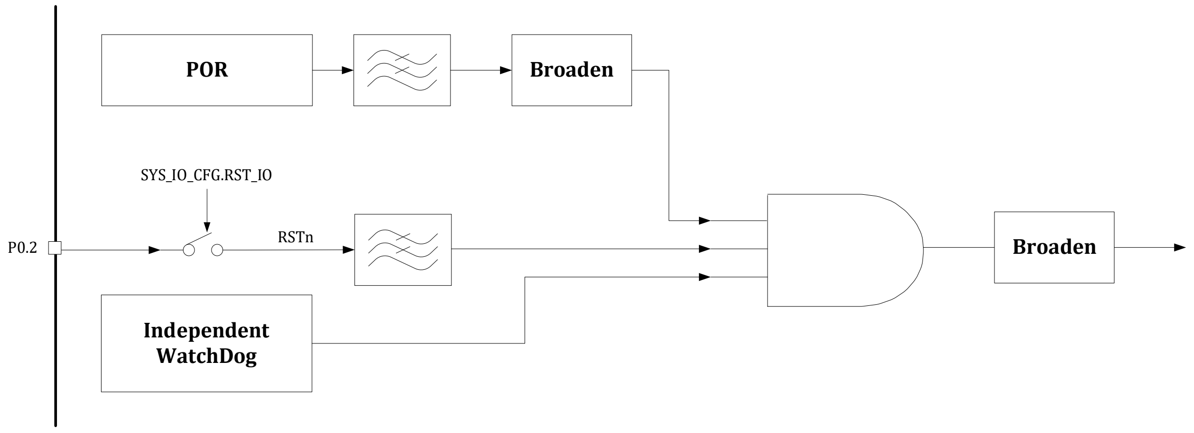


Figure 6-2 Reset the architecture

6.2.1.1.2 Hardware reset logging

The AON_EVT_RCD register is used to save hardware reset events. When a hardware reset occurs, the corresponding bit of AON_EVT_RCD is set. The AON_EVT_RCD register itself be reset by the reset signal; it can only clear the record by writing 0xCA40 to the AON_EVT_RCD register. With the reset record, we can easily understand whether and what kind of reset has occurred.

6.2.1.2 Software reset

CPU soft reset can return the PC (PC: Program Counter) to address 0. Whether the soft reset resets the peripheral registers depends on the SYS_DBG_CFG. SFT_RST_PERI setting.

In the IDE (IDE: Integrated Development Environment) debug mode, clicking "Reset" has the same effect as "CPU Soft Reset", which will only return the PC to address 0, and do not affect the registers in all peripherals. However, if a soft reset of the peripheral module is performed in the bootloader, the peripheral register will be reset to the default value. Please consult the chip supplier for the specific bootloader implementation. If the SYS_DBG_CFG.SFT_RST_PERI = 1, the peripheral registers can be reset at the same time.

Some peripheral modules have a soft reset, which is performed by using the SYS_SFT_RST register. Write the corresponding bit to the register, restore the module state machine to its initial state, and restore the module register to the default value, see 6.3.8 for details.

6.2.2 Resets the scope

Table 6-4 Resets the scope

Reset source	Scope
PORn	Internal power management, global reset
RSTn	External key, global reset, except very few registers
WDTn	Hardware watchdog, global reset, except very few registers
SYS_SFT_RST.SPI0_ SFT_RST	SPI0
SYS_SFT_RST.I2C0_SFT_RST	I2C0
SYS_SFT_RST.CMP_SFT_RST	CMP digital interface block (does not include analog comparato)
SYS_SFT_RST.HALL0_SFT_RST	HALL0
SYS_SFT_RST.TIMER0_ SFT_RST	TIMER0
SYS_SFT_RST.TIMER1_ SFT_RST	TIMER1
SYS_SFT_RST.TIMER2_ SFT_RST	TIMER2
SYS_SFT_RST.TIMER3_ SFT_RST	TIMER3
SYS_SFT_RST.QEP0_ SFT_RST	QEP0
SYS_SFT_RST.QEP1_ SFT_RST	QEP1
SYS_SFT_RST.MCPWM0_ SFT_RST	MCPWM
SYS_SFT_RST.GPIO_ SFT_RST	GPIO
SYS_SFT_RST.ADC0_ SFT_RST	ADC0 digital interface module
SYS_SFT_RST.ADC1_ SFT_RST	ADC1 Digital Interface Module
SYS_SFT_RST.UART0_SFT_RST	UART0
SYS_SFT_RST.UART1_SFT_RST	UART1
SYS_SFT_RST.CRC0_ SFT_RST	CRC0
SYS_SFT_RST.DSP0_ SFT_RST	DSP0
SYS_SFT_RST.DMA0_ SFT_RST	DMA0
SYS_SFT_RST.CAN0_ SFT_RST	CAN0
SYS_SFT_RST.SIF0_ SFT_RST	SIF0
SYS_SFT_RST.CL0_ SFT_RST	CL0
NVIC_SystemReset();	CPU soft reset, which resets only the CPU core, resets the PC to 0. If SYS_DBG_CFG. SFT_RST_PERI = 0, all peripheral register values remain, and SYS_DBG_CFG. SFT_RST_PERI = 1, all peripheral registers are reset simultaneously.

The SYS_IO_CFG. RST_IO used to control whether P0 [2] is used as a GPIO or an external reset pin, and the AON_EVT_RCD of the reset record register is only reset by POR.

A global reset resets all on-chip registers, including the CPU core registers and all peripheral registers, except for the very few registers mentioned above.

If the CPU soft reset is set to reset only the CPU core and not the peripheral registers. It is recommended to reset the peripheral registers by power-down or external reset after reburning the program.

Flash stores contents, and SRAM stores contents are not affected by reset.

6.3 Register

6.3.1 Address assignment

The system module register base address is 0x 4000_0000.

Table 6-5 System Control Register

Name	Offset	Description
SYS_CLK_CFG	0x80	Clock Control Register
SYS_IO_CFG	0x84	IO control register
SYS_DBG_CFG	0x88	Debug control register
SYS_CLK_DIV0	0x90	Peripheral Clock Divide Register 0
SYS_CLK_DIV2	0x98	Peripheral Clock Divide Register 2
SYS_CLK_FEN	0x9C	Peripheral Clock Gating Register
SYS_SFT_RST	0xA4	Soft Reset Register
SYS_PROTECT	0xA8	Write protect register
SYS_FLSE	0xD0	Flash Erase Protection Register
SYS_FLSP	0xD4	Flash Program Protection Register

6.3.2 SYS_CLK_CFG Clock Control Register

Address: 0x4000 0080

Reset value: 0x0

Table 6-6 SYS_CLK_CFG Clock Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		XTAL_FAILED	XTAL_FAIL			CLK_SEL		CLK_DIV							
		RW1C	RO			RW		RW							
		1	0			0		0							

Location	Bit name	Description
[31:14]		Not used
[13]	HSE_FAILED	Does the crystal oscillator clock stop oscillating after being turned on? If the crystal stops oscillating, this bit is set and remains recorded as 1 even if the crystal subsequently resumes oscillation. Write 1 to clear
[12]	HSE_FAIL	Is the crystal oscillator clock currently stopped? 0: The external crystal is currently working normally 1: The external crystal has stopped vibrating currently.
[11:10]		Not used
[9:8]	CLK_SEL	Source select signal for system clock MCLK. HRC is selected by default 0: HRC 1: PLL 2: LRC 3: XTAL Note that the PLL/XTAL is off by default after power-up and requires software to turn it on.

[7:0]	CLK_DIV	PLL output divide control, default 0x00: 1/8 frequency division 0x01: 1/8 frequency division 0x11: 1/4 frequency division 0x55: 1/2 frequency division 0xFF: 1/1 frequency division Other configuration values are not recommended.

When the CLK_SEL = 0, the MCLK selects the HSI clock (8 MHz), and the division factor of the SYS_CLK_CFG [7:0] has no effect. The resulting output is a system clock of 8 MHz.

When the CLK_SEL = 2, MCLK selects the LRC clock (32kHz), and the division factor of the SYS_CLK_CFG [7:0] is invalid. The final output system clock frequency is 32 kHz.

When the CLK_SEL = 3, the HSE clock (8 MHz) is selected by MCLK, and the division factor of the SYS_CLK_CFG [7:0] has no effect. The resulting output is a system clock of 8 MHz.

6.3.3 SYS_IO_CFG IO control register

Address: 0x4000 0084

Reset value: 0x40

Table 6-7 SYS_IO_CFG IO control register

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

Location	Bit name	Description
[31:7]		Not used
[6]	SWDMUX	SWD multiplex control signal, default configuration is SWD 0: P2.14 is multiplexed to SWCLK and P2.15 is multiplexed to SWDIO 1: P2.14, P2.15 as normal GPIO
[5]	RST_IO	RSTn/P0.2 multiplex control signal, default configuration is RSTn 0:RSTn 1:P0.2 Note that the default is RSTn after power-up, and subsequent software can enable this bit to disable the RSTn function.
[4:0]		Not used

For safety reasons, the IO of the SWD cannot be switched to the GPIO function within 16 ms after power-up, and can only be used as the SWD. Even if the software overwrites the SYS_IO_CFG[6], it will not work and the SYS_IO_CFG[6] needs to be overwritten again after 16 ms. The behavior is that software can write 1'b1 to the SYS_IO_CFG[6] within 16 ms, but the bit is still read back as 1'b0. If 1'b1 is written again after 16 ms, 1'b1 can be read back. That is, writing 1'b1 to the software SYS_IO_CFG[6] is invalid within 16 ms after power-on, and writing 1'b1 to the SYS_IO_CFG[6] by the software is valid 16 ms after power-on. This is to prevent the SWD function from being switched to GPIO when the application is powered on, so that the chip cannot erase and burn the flash through the SWD subsequently.

Note that after SWD IO is switched to GPIO function, the user can no longer debug the chip through SWD. Therefore, it is recommended to leave a certain time window after power-on and switch SWD to GPIO function after a certain delay. It is also recommended to leave other design considerations in the software so that SWD IO can be switched back to SWD.

6.3.4 SYS_DBG_CFG Debug control register

Address: 0x4000 0088

Reset value: 0x40

Table 6-8 SYS_DBG_CFG Debug control register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_IRQ_TRIG															
W															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_IRQ	SFT_RST_PERI			DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_TIM0_STOP			DBG_IWDG_STOP				DBG_STOP	DBG_SLP
RW1C	RW			RW	RW	RW	RW			RW				RW	RW
0	0			0	0	0	0			0				0	0

Location	Bit name	Description
[31:16]	SW_IRQ_TRIG	Writing 0x5AA5 to this bit triggers a software interrupt and sets the SW_IRQ to 1
[15]	SW_IRQ	Software interrupt flag, corresponding to interrupt number 30, write 1 to clear.
[14]	SFT_RST_PERI	Whether the Debug soft reset resets peripheral registers other than the Flash/SYS_AFE.
[13:12]		Not used
[11]	DBG_TIM3_STOP	Timer3 stop in CPU halt state of debug model 1: Timer3 stops counting in CPU halt state 0: Timer3 continues counting in CPU halt state
[10]	DBG_TIM2_STOP	Timer2 stop in CPU halt state of debug model 1: Timer2 stops counting in CPU halt state 0: Timer2 continues counting in CPU halt state

[9]	DBG_TIM1_STOP	Timer1 stop in CPU halt state of debug model 1: Timer1stops counting in CPU halt state 0: Timer1 continues counting while in CPU halt state
[8]	DBG_TIM0_STOP	Timer0 stop in CPU halt state of debug model 1: Timer0 stops counting in CPU halt state 0: Timer0 continues counting in CPU halt state
[7:6]		Not used
[5]	DBG_IWDG_STOP	Independent watchdog stop in CPU halt state of debug mode 1: Independent watchdog stops counting in CPU halt state 0: Independent watchdog continues counting in CPU halt state
[4:2]		Not used
[1]	DBG_STOP	Debug STOP mode 0: (FCLK = Off, HCLK = Off) In STOP mode, the clock management module turns off HCLK and FCLK, the processor clock and all peripheral clocks. When exiting the STOP mode, the clock management module does not undergo reset, maintains the original configuration, and can be restored to the clock setting before the STOP mode. In STOP mode, all peripheral registers are held, so there is no need to reconfigure after exit. 1: (FCLK = On, HCLK = On) If the DBG_STOP is set to 1, both HCLK and FCLK are not turned off after entering the STOP mode. Enter the STOP mode by setting SCB- > SCR = (1UL < < 2) and then using the__ WFI ()/___ WFE () command.
[0]	DBG_SLP	Debug Sleep (SLEEP) Mode 0: (FCLK = On, HCLK = Off) In sleep mode, FCLK is used as all peripheral clocks and is not turned off; HCLK, as the CPU clock, is turned off. In the sleep mode, only the CPU clock is temporarily suspended, and all peripherals including the system clock management module are kept in the original configuration state. Therefore, software does not need to reconfigure the peripheral registers when exiting sleep mode. 1: (FCLK = On, HCLK = On) If the configuration DBG_SLP is 1, HCLK is not turned off when entering sleep mode. The processor can be put into sleep mode by a__ WFI ()/___ WFE () instruction

6.3.5 SYS_CLK_DIV0 peripheral clock divide register 0

Address: 0x4000 0090

Reset value: 0x0

Table 6-9 SYS_CLK_DIV0 peripheral clock divide register 0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV0															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DIV0	I2C operation clock = MCLK/ (CLK_DIV0 + 1). Where MCLK is determined by

		the SYS_CLK_CFG division factor

6.3.6 SYS_CLK_DIV2 Peripheral Clock Divide Register 2

Address: 0x4000 0098

Reset value: 0x0

Table 6-10 SYS_CLK_DIV2 Peripheral Clock Divide Register 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV2															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DIV2	UART operation clock = MCLK/ (CLK_DIV2 + 1). The UART0/UART1 share this divide configuration, and the baud rate is further divided according to the Uart baud rate register, where MCLK is determined by the SYS_CLK_CFG divide factor.

6.3.7 SYS_CLK_FEN Peripheral Clock Gating Register

Address: 0x4000 009C

Reset value: 0x0

Table 6-11 SYS_CLK_FEN Peripheral Clock Gating Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										CL0_CLK_EN	SIF0_CLK_EN	CAN0_CLK_EN	Res.	DSP0_CLK_EN	CRC0_CLK_EN
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART1_CLK_EN	UART0_CLK_EN	Res.	Res.	GPIO_CLK_EN	MCPWM0_CLK_EN	QEP1_CLK_EN	QEP0_CLK_EN	TIMER3_CLK_EN	TIMER2_CLK_EN	TIMER1_CLK_EN	TIMER0_CLK_EN	HALL0_CLK_EN	CMP_CLK_EN	I2C0_CLK_EN	SPI0_CLK_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:21]		Not used

[21]	CL0_CLK_EN	CL0 module clock control signal, CL0 module clock off by default 1: Enable CL0 module clock 0: CL0 module clock off
[20]	SIF0_CLK_EN	SIF0 module clock control signal, SIF0 module clock off by default 1: Enable SIF0 module clock 0: Turn off SIF0 module clock
[19]	CAN0_CLK_EN	CAN0 module clock control signal, CAN0 module clock off by default 1: Enable CAN0 module clock 0: Turn off the CAN0 module clock
[18]		Reserved
[17]	DSP0_CLK_EN	DSP0 module clock control signal, DSP0 module clock off by default 1: Enable DSP 0 module clock 0: Turn off the DSP0 module clock
[16]	CRC0_CLK_EN	CRC0 module clock control signal, CRC0 module clock is off by default 1: Enable CRC0 module clock 0: Turn off the clock of CRC0 module
[15]	UART1_CLK_EN	UART1 module clock control signal, the default is to turn off the UART1 module clock 1: Enable UART1 module clock 0: Turn off UART1 module clock
[14]	UART0_CLK_EN	Clock control signal of UART0 module, which turns off the clock of UART0 module by default 1: Enable UART0 module clock 0: Turn off UART0 module clock
[13]		Reserved
[12]		Reserved
[11]	GPIO_CLK_EN	GPIO module clock control signal, GPIO module clock is off by default 1: Enable GPIO module clock 0: Turn off GPIO module clock
[10]	MCPWM0_CLK_EN	MCPWM0 module clock control signal, MCPWM0 module clock off by default 1: Enable MCPWM0 module clock 0: MCPWM0 module clock off
[9]	QEP1_CLK_EN	QEP1 module clock control signal, QEP1 module clock is off by default 1: Enable QEP1 module clock 0: Turn off the clock of QEP1 module
[8]	QEP0_CLK_EN	QEP0 module clock control signal, QEP0 module clock off by default 1: Enable QEP0 module clock 0: Turn off the clock of QEP0 module
[7]	TIMER3_CLK_EN	TIMER3 module clock control signal, TIMER3 module clock is off by default 1: Enable TIME R3 module clock 0: Turn off TIMER3 module clock
[6]	TIMER2_CLK_EN	TIMER2 module clock control signal, TIMER2 module clock is off by default 1: Enable TIMER2 module clock 0: Turn off TIMER2 module clock
[5]	TIMER1_CLK_EN	TIMER1 module clock control signal, TIMER1 module clock is off by default 1: Enable TIMER1 module clock

		0: turn off TIMER1 module clock
[4]	TIMER0_CLK_EN	TIMER0 module clock control signal, TIMER0 module clock is off by default 1: Enable TIMER0 module clock 0: turn off TIMER0 module clock
[3]	HALL0_CLK_EN	HALL0 module clock control signal, HALL0 module clock is off by default 1: Enable HALL0 module clock 0: Turn off the clock of HALL0 module
[2]	CMP_CLK_EN	CMP module clock control signal, CMP module clock is off by default 1: Enable CMP module clock 0: Turn off CMP module clock
[1]	I2C0_CLK_EN	I2C0 module clock control signal, I2C0 module clock is off by default 1: Enable I2C0 module clock 0: Turn off I2C0 module clock
[0]	SPI0_CLK_EN	SPI0 module clock control signal, SPI0 module clock off by default 1: Enable SPI0 module clock 0: Turn off SPI0 module clock

Note that the clock of each module mentioned above is the working clock of the internal circuit of the respective module. Even if the clock of the respective module is not turned on, it will not affect the software to access the registers of the respective modules.

6.3.8 SYS_SFT_RST soft reset register

Address: 0x4000 00 A4

Reset value: 0x0

Table 6-12 SYS_SFT_RST soft reset register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
										CL0_SFT_RST	SIF0_SFT_RST	CAN0_SFT_RST	DMA0_SFT_RST	DSP0_SFT_RST	CRC0_SFT_RST
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

15	4	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART1_SFT_RST	UART0_SFT_RST	ADC1_SFT_RST	ADC0_SFT_RST	GPIO_SFT_RST	MCPWM0_SFT_RST	QEP1_SFT_RST	QEP0_SFT_RST	TIMER3_SFT_RST	TIMER2_SFT_RST	TIMER1_SFT_RST	TIMER0_SFT_RST	HALL0_SFT_RST	CMP0_SFT_RST	I2C0_SFT_RST	SPI0_SFT_RST
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:21]		Not used
[21]	CL0_SFT_RST	CL0 module soft reset signal, the default is not to reset the CL0 module

		1: reset CL0 module 0: Release CL0 module
[20]	SIF0_SFT_RST	SIF0 module soft reset signal, default not to reset SIF0 module 1: Reset SIF0 module 0: Release SIF0 module
[19]	CAN0_SFT_RST	CAN0 module soft reset signal, the default is not to reset the CAN0 module 1: Reset CAN0 module 0: CAN0 module is released
[18]	DMA0_SFT_RST	DMA0 module soft reset signal, default not to reset DMA0 module 1: Reset DMA0 module 0: Release DMA0 module
[17]	DSP0_SFT_RST	DSP0 module soft reset signal, default not to reset DSP0 module 1: Reset DSP0 module 0: release DSP0 module
[16]	CRC0_ SFT_RST	Soft reset signal of CRC0 digital interface module, the default is not to reset the CRC0 module 1: Reset CRC0 digital interface module 0: Release CRC0 digital interface module
[15]	UART1_SFT_RST	UART1 module soft reset signal, default not to reset UART1 module 1: Reset UART1 module 0: Release UART1 module
[14]	UART0_SFT_RST	UART0 module soft reset signal, default not to reset UART0 module 1: Reset UART0 module 0: Release UART0 module
[13]	ADC1_ SFT_RST	ADC1 digital interface module soft reset signal, ADC1 module is not reset by default 1: reset ADC 1 digital interface module 0: release ADC 1 digital interface module
[12]	ADC0_ SFT_RST	ADC0 digital interface module soft reset signal, ADC0 module is not reset by default 1: Reset ADC0 digital interface module 0: Release ADC0 digital interface module
[11]	GPIO_ SFT_RST	GPIO module soft reset signal, GPIO module is not reset by default 1: Reset GPIO module 0: Release GPIO module
[10]	MCPWM0_ SFT_RST	MCPWM0 module soft reset signal, default not to reset MCPWM0 module 1: Reset MCPWM0 module 0: Release the MCPWM0 module
[9]	QEP1_ SFT_RST	Soft reset signal of QEP1 module, the default is not to reset QEP1 module 1: Reset QEP1 module 0: Release QEP1 module
[8]	QEP0_ SFT_RST	Soft reset signal of QEP0 module, the default is not to reset QEP0 module 1: Reset QEP0 module 0: Release QEP0 module
[7]	TIMER3_ SFT_RST	TIMER3 module soft reset signal, default not to reset TIMER3 module 1: Reset TIMER3 module 0: Release TIMER3 module

[6]	TIMER2_ SFT_RST	TIMER2 module soft reset signal, default not to reset TIMER2 module 1: Reset TIMER2 module 0: Release TIMER2 module
[5]	TIMER1_ SFT_RST	TIMER1 module soft reset signal, default not to reset TIMER1 module 1: Reset TIMER1 module 0: Release TIMER1 module
[4]	TIMER0_ SFT_RST	TIMER0 module soft reset signal, default not to reset TIMER0 module 1: Reset TIMER0 module 0: Release the TIMER0 module
[3]	HALL0_SFT_RST	HALL0 module soft reset signal, default not to reset HALL0 module 1: Reset HALL0 module 0: Release HALL0 module
[2]	CMP_ SFT_RST	CMP module soft reset signal, default not to reset CMP module 1: Reset CMP module 0: Release CMP module
[1]	I2C_0SFT_RST	I2C0 module soft reset signal, default not to reset I2C0 module 1: Reset I2C0 module 0: Release I2C0 module
[0]	SPI0_ SFT_RST	SPI0 module soft reset signal, SPI0 module is not reset by default 1: Reset SPI0 module 0: Release SPI0 module

Note that a soft reset of the module remains in the reset state after 1 is written to the corresponding bit of the SYS_SFT_RST, and 0 needs to be written again to release the reset state.

6.3.9 SYS_PROTECT Write Protect Register

Address: 0x4000_00 A8

Reset value: 0x0

Table 6-13 SYS_PROTECT Write Protect Register

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	PSW	In addition to SYS_AFE_DAC0, SYS_AFE_DAC0_AMC, SYS_AFE_DAC0_DC, In addition to the SYS_AFE_DAC1, the SYS_AFE_DAC1_AMC, and the SYS_AFE_DAC1_DC, other system registers (the registers at the beginning of SYS_, including the clock reset management and the analog registers) are write-protected and need to be written with a password to remove the write-protection before writing Write 0x7A83 to remove the write protection and enable the register write Write other values, enable write protection, and disable register writes

		Readback is always 0

6.3.10 SYS_FLSE Erase Protection Register

Address: 0x4000 00D0

Reset value: 0x0

Table 6-14 Erase protection register SYS_FLSE.

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	FLSE	**FLASH erase protection register. The erase function of FLASH is finally enabled by writing 0x8FCA to this register. The erase function of FLASH cannot take effect when other values are written.To prevent false erasure, it is recommended that this register is not activated when the erase function is not used.**

6.3.11 SYS_FLSP Program Protection Register

Address: 0x4000 00D4

Reset value: 0x0

Table 6-15 Program protection register SYS_FLSE.

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	FLSP	**FLASH program protection register. Write 0x8F35 to this register for the FLASH programming function to finally take effect. The programming function of FLASH cannot be effective when other values are written.To prevent misprogramming, it is recommended that this register is not activated when the programming function is not used.**

7 Non-volatile memory

7.1 Introduction

Non-volatile memory consists of two parts: FLASH and ROM.

The FLASH consists of two parts: NVR and MAIN. NVR size is 1.5KB; MAIN comes in three sizes: 32KB, 64KB, and 128KB.

- The main flash memory area (MAIN) includes application programs and user data area.
- FLASH information storage area (NVR), reserved for user data area

ROM, fixed in the factory for specific programs, has two sizes: 32KB and 64KB.

LKS32MC07x series chips, non-volatile memorys include five models:

- Model 1: 1.5K B NVR, 32KB FLASH
- Model 2: 1.5K B NVR, 32KB FLASH, 32KB ROM
- Model 3: 1.5K B NVR, 64KB FLASH
- Model 4: 1.5K B NVR, 64KB FLASH, 64KB ROM
- Model 5: 1.5K B NVR, 128KB FLASH

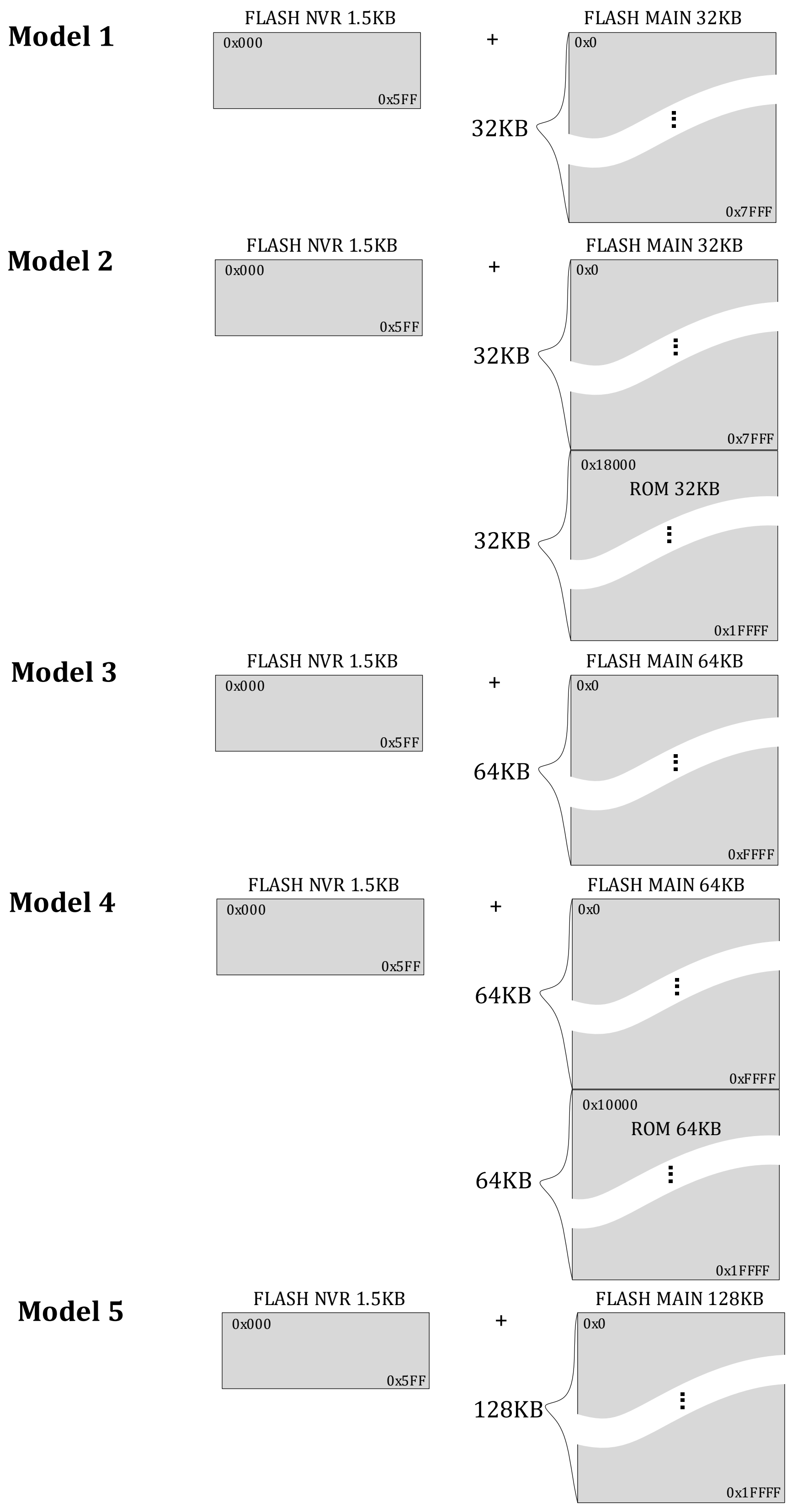


Figure7-1 Non-volatile memory space division block diagram and model

7.2 Functional features

The nonvolatile memory controller is mainly used for realizing relevant operations on the FLASH memory; The ROM contains only the execution operations. These include:

- FLASH reading, including reading operations of the NVR and MAIN.
- FLASH writing, including writing operations to the NVR and MAIN.
- FLASH erase, including CHIP erase and SECTOR erase. SECTOR erase only available in NVR, and the MAIN part supports both CHIP erase and SECTOR erase.
- FLASH deep sleep, reducing the sleep power consumption of the chip.
- FLASH memory content encryption.
- FLASH reading acceleration, which improves the overall operation efficiency of the chip.
- FLASH control register access.
- ROM area, executable only, not copyable.
- In FLASH protected state, Sector 2 of NVR (counting from 0) supports read, program and erase.

7.2.1 Functional description

The nonvolatile memory bank controller realizes operations on the FLASH memory such as reset/read/write/erase/ sleep. The following is the control state transition diagram:

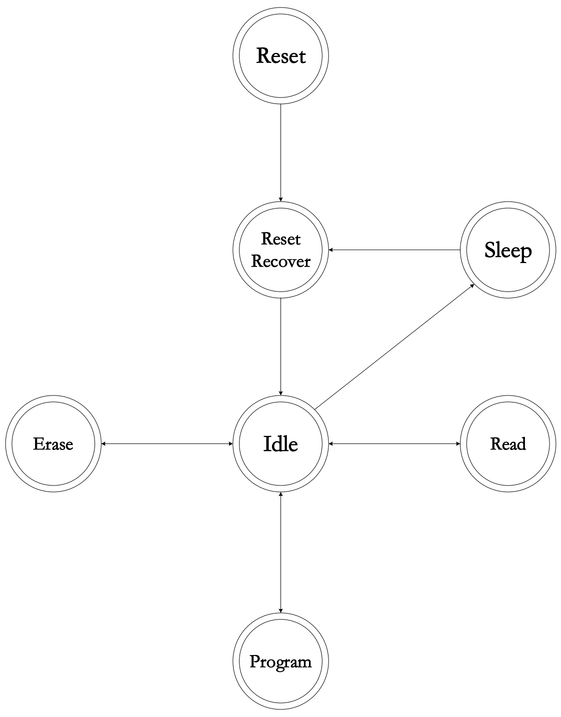


Figure 7-2 FLASH control state transition diagram

7.2.1.1 Reset operation

After the system is reset, it takes some times for the FLASH recovery, thus to ensure the internal circuit stability of the FLASH memory. After then, other operations could be performed on FLASH. This recovery operation is automatically realized by hardware without software intervention.

7.2.1.2 Sleep operation

The sleep operation of FLASH is divided into two parts: Standby and Deep Sleep. When no operations will be performed on FLASH, FLASH enters the StandBy state automatically (if prefetching is turned on, this function is disabled). When the system executes Deep Sleep operation, it will trigger FLASH to enter Deep Sleep, to further reduce power consumption. The operation of FLASH entering Deep Sleep is completed by hardware automatically without software intervention.

When system is waked up by the outside world, the FLASH will be waked up at the same time. After a period of recovery, FLASH operations can be performed normally. This wake-up recovery operation is automatically completed by hardware without software intervention.

7.2.1.3 Read operation

The read operation is the basic operation of FLASH. The system can access the data inside FLASH through two paths.

- The MCU fetches instructions and accesses data directly on the FLASH through the AHB bus. The fetch width is 32bit, and can only access data in the MAIN space. The hardware also provides an acceleration function to speed up the MCU to fetch instructions and access data.
- The MCU accesses the register of the controller through the AHB bus to read FLASH internal data indirectly. Both the data in the MAIN and NVR spaces can be accessed; if a continuous reading is required, the hardware will accumulate the addresses automatically without updating the address register value each time.

  The ROM cannot be read and only can be performed. The system has only one way to access the data inside the ROM

- Through the AHB bus, the MCU directly fetches instructions and data from the ROM, with a width of 32 bits.

The FLASH_CFG. REGION bit indicates which space is currently being accessed. The specific table is as follows:

Table 7-1 FLASH Access Space Allocation Table

FLASH_CFG.REGION	Access area
0	MAIN area
1	NVR area

The process for reading the internal data indirectly on FLASH by accessing the register of the controller is as follows:

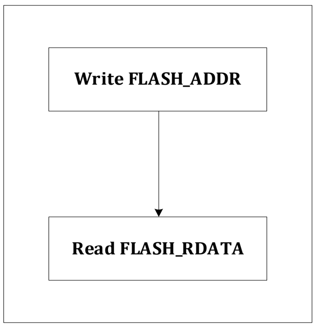


Figure 7-3 FLASH Indirect Read Operation Flowchart

7.2.1.4 FLASH programming operation

FLASH Programming refers to programming operations on the FLASH memory bank. Generally, an erase operation should be performed before data programming. And, the programming can only be performed by accessing the registers of the FLASH controller. The specific process is:

- SYS_FLSP Register, write 0x8F35 to turn on Program Enable Switch 1
- Write 1 to the FLASH_CFG.PRG_EN to turn on the program enable switch 2
- FLASH_ADDR, write programming addr
- FLASH_WDATA, write programming data

The process for FLASH programming by accessing the register of this controller is as follows:

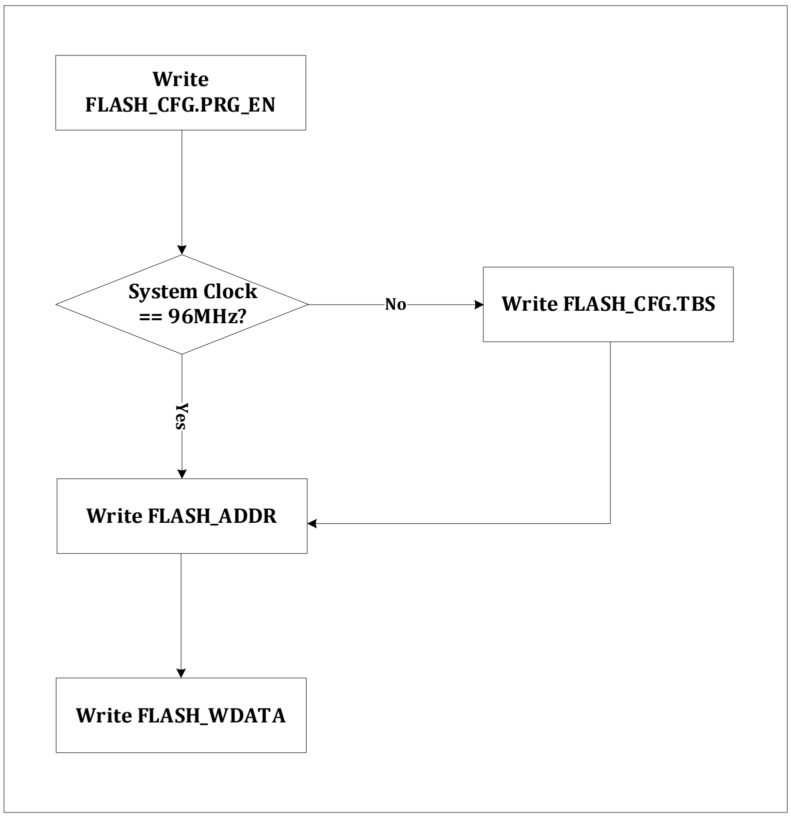


Figure 7-4 Flow chart of FLASH module programming operation

In order to prevent the FLASH region from being programmed by mistake, an additional set of programming enable switches, SYS_FLSP, is added. The SYS_FLSP protects the FLASH from being programmed by mistake. Write 0x8F35, enable the programming enable, configure the FLASH_CFG.PRG_EN, and finally enable the programming function of FLASH.

For the selection of operating frequency, please refer to the configuration of SYS_CLK_CFG. The absolute time of the FLASH write/erase operation is fixed, and the count value corresponding to these absolute times should be saved in the FLASH controller. The default value of FLASH_CFG.TBS is the count value at 96MHz clock frequency; When the chip works at other frequencies, the value of FLASH_CFG.TBS should be set to achieve the count values of 48MHz and 24MHz (other frequencies are not supported). In this way, the value obtained by multiplying the count value by the clock frequency is equal to a constant time. The corresponding FLASH_CFG. TBS values at different frequencies are listed in the FLASH_CFG register. Please note that only the values provided in the register description could be set to the FLASH_CFG.TBS, and other values are not available for writing into; otherwise, it may cause FLASH programming/erasing failure. It is recommended to read first on the FLASH_CFG, and then perform other operations by follow the OR/AND method. Besides, the CPU will stop temporarily when the FLASH program/erase operation is performing until the operation is finished.

Figure 7-4 only shows the flow of one programming. If continuous programming is required, set the FLASH_CFG.ADR_INC before writing to the FLASH_ADDR register to enable the address auto-increment mode. After then, repeatedly write into the FLASH_WDATA register, and the address will be added by 0x4 automatically each time when writing data into the FLASH_ADDR. The operation of continuous reading is similar to this. The continuous programming process is as follows:

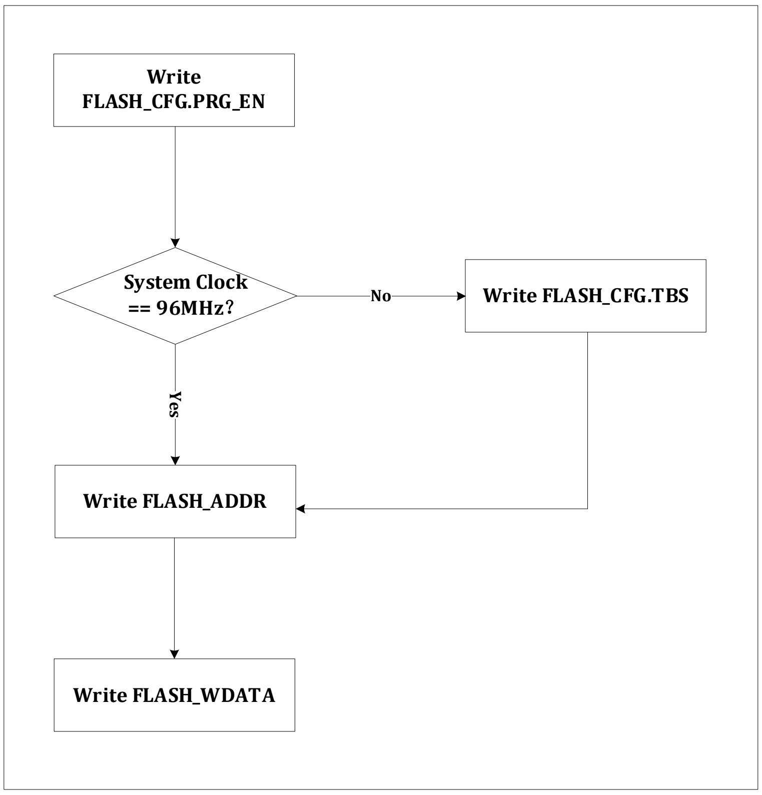


Figure 7-5 Flow chart of FLASH module programming operation

7.2.1.5 FLASH erase operation

The erase operation is the basic operation of FLASH. The system can only be implemented by accessing the registers of the FLASH controller. The specific process is as follows:

- SYS_FLSE Register, Write 0x8FCA, Erase Enable Switch 1
- FLASH_CFG. ERS_EN Write 1, Erase Enable Switch 2
- FLASH_ADDR, write erase addr
- FLASH_ERASE, triggering an erase operation

In order to prevent the FLASH area from being erased by mistake, a group of erase enable switches, namely SYS_FLSE, are additionally added. The SYS_FLSE protects the FLASH from being erased by mistake, writes 0x8FCA, enables the erase enable, configures the FLASH_CFG.ERS_EN, and finally enables the erase function of the FLASH.

Perform flash erase on the FLASH memory bank. The erase operation is divided into Sector erasure and FullChip erasure, corresponding to 512Byte erasure and 32KB/64kB/128KB erasure respectively. Which type of erase operation is performed can be determined by setting the FLASH control register.

The following table allocates space for Secotor addresses.

Table 7-2 FLASH Sector Address Allocation Table

Name	Addresses	Size(Bytes)
Sector 0	0x0000 0000 - 0x0000 01FF	512
Sector1	0x0000 0200 - 0x0000 03FF	512
Sector2	0x0000 0400 - 0x0000 05FF	512
...	...	...
Sector255	0x0001 FE00 - 0x0001 FFFF	512

The NVR area can only realize Sector erasure, and the NVR area only has three sectors (0/1/2); the MAIN area can realize Sector erasure and Full erasure. The specific table is as follows:

NVR（FLASH_CFG .REGION）	Sector Erase	Full Erase
0	Main area	Main area
1	NVR area	Main area

The FLASH erase operation flow is as follows.

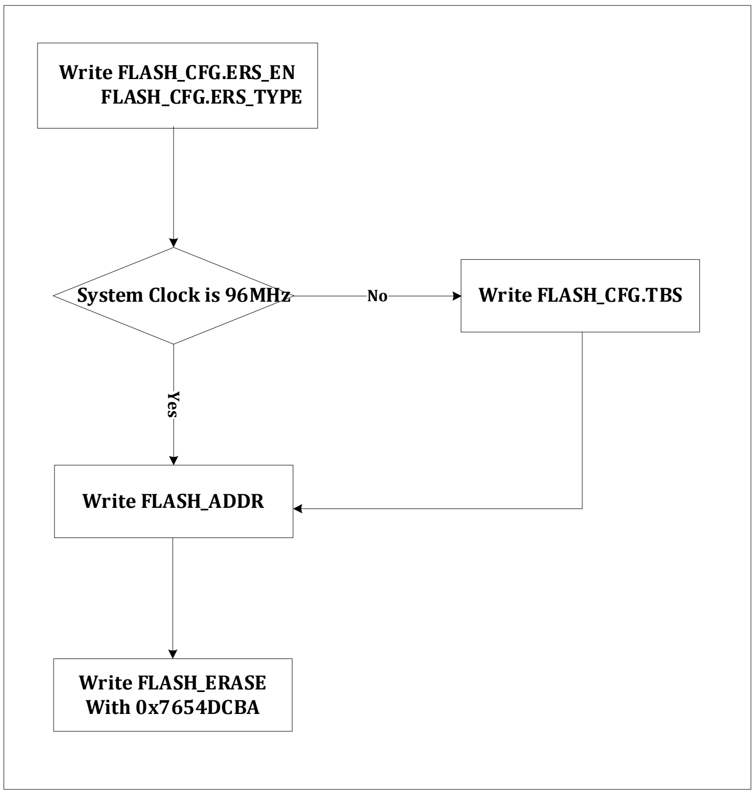


Figure 7-6 FLASH Module Erase Operation Flowchart

If you select Secotor erase, you need to determine which Secotor is erased through the FLASH_ADDR. If it is Full mode, the value of the FLASH_ADDR will be invalid. Writing 0x7654DCBA to the FLASH_ERASE triggers an erase operation.

7.2.1.6 FLASH prefetch operation

Due to the speed limitation of FLASH memory bank, the speed of 96MHz cannot be achieved. When reading the FLASH, a clock cycle longer than 96 MHz is required to complete the data reading. In order to speed up the data read out, the FLASH controller adds a prefetch function. After the FLASH controller completes the current reading operation, the data of the next WORD is prefetched sequentially without affecting the normal program execution. You only need to set the FLASH_CFG. PREF to open and close the prefetch operation.

When the system frequency is reduced to 48 MHz and below, the FLASH is read without waiting for two clock cycles. In this case, you can configure the FLASH_CFG.WAIT to be 0, removing an additional wait period. Remember, when adjusting from high frequency to low frequency, first slow down the clock, and then change the FLASH_CFG.WAIT to 0; When adjusting from low frequency to high frequency, first change the FLASH_CFG.WAIT to 1, and then set the clock fast.

7.2.1.7 Non-volatile memory bank protection

Non volatile storage supports encryption protection, which can protect FLASH and ROM. Because the ROM can only be executed and cannot be copied, it is already in a protected state and cannot be changed; This section mainly explains the protection of the FLASH section.

The purpose of FLASH protection is to prevent the outside world from illegally obtaining FLASH content.

If the data in the FLASH memory bank is in a protection state, the user can perform a de-protection operation; Conversely, if the data in the FLASH bank is in an unprotected state, the user can perform a protection operation. By default, the data in the FLASH store is protected. After the power-on reset of the chip is completed, the hardware automatically executes a state update operation, if the chip is in a protection state, the chip maintains the protection state, otherwise, the chip becomes a non-protection state.

FLASH memory banks have 32KB, 64KB and 128KB, and the last WORD of each is designed as a protected word (32KB corresponds to 0x7FFC, 64KB corresponds to 0xFFFC, and 128KB corresponds to 0X1FFFC). When the content of this WORD is all "1", it means that the content of FLASH does not need to be protected, and it is enough to read the FLASH_PROTECT register and complete the status update. When the content of this WORD is written as non-all "1", it means that the content of FLASH needs to be protected. If protection is required, only the content of the last WORD needs to be written into a value other than all "1", the FLASH_PROTECT register needs to be read, the status update is completed (it is meaningless to read the return value of the FLASH_PROTECT), and the protection is started.

Deprotection is divided into two situations. If the last WORD has not executed the operation of writing not all "1", read the FLASH_PROTECT register to complete the status update (it is meaningless to read the return value of the FLASH_PROTECT), and remove the protection. If the last WORD is not all "1", you need to perform an erase operation to remove the protection. Erase the FLASH to restore the last WORD to all "1", and then read the FLASH_PROTECT register to complete the status update and cancel the protection (it is meaningless to read the FLASH_PROTECT return value).

When FLASH is in the protection state, it is convenient for customers to realize special application requirements such as secondary burning. The LKS32MC07x releases only Sector 2 of the NVR area for customer access operations, including read, program, and erase. Note that this function is valid only when FLASH is in the protected state. Meanwhile, when performing this operation, we can only realize this function by accessing the following registers FLASH_NCFG/NADDR/NWDATA/NRDATA/NERASE/NKEY.

The specific process is as follows:

- SYS_FLSE Register, Write 0x8FCA, Erase Enable Switch 1
- SYS_FLSP Register, write 0x8F35 to turn on Program Enable Switch 2
- Read the FLASH_PROTECT register to confirm that FLASH is in the protected state
- FLASH_NKEY, write the special value (0x0000_000F) to enable the operation of NVR Sector 2 in the protection state
- FLASH_NCFG, turn on the program and erase switches (if required)
- FLASH_NADDR, write to program or erase addr
- FLASH_NERASE, enable erasure of NVR Sector 2 (if required)
- FLASH_NWDATA, enable programming of NVR Sector 2 (if required)

- FLASH_NRDATA, read NVR Sector 2

Remember that this path can only be accessed to NVR Sector 2, and after the general access is complete, a hard reset is performed.

7.2.1.8 FLASH Online Upgrade (IAP)

IAP mode to remap the interrupt vector table. In the LKS32MC07x series chip, the system register VTOR is included, and its address is 0xE000_ED08. Used to remap the interrupt vector table entry address.

Table 7-3 IAP VTOR register description

Name	Reset value	Offset	Location	Permissions	Description
VTOR	0x0		[31:7]	RW	Execute the write operation and write the entry address of the interrupt vector table

The default value is 0x0, and the interrupt vector table entry address is 0x0. When a non-zero value is written, the interrupt vector table entry address is mapped to the address corresponding to the VTOR register and takes effect immediately.

In LKS32MC07x series chips, because of the VTOR register. Users can update the content of the whole FLASH according to their own needs. Interrupts can be used or turned off during an online upgrade.

7.2.1.8.1 Turn on interrupted online upgrade

Recommended software configuration process:

Turn off the interrupt controller of the MCU to stop receiving the new interrupt responses temporarily;

Place the interrupt processing function code at the new interrupt entry address;

Write the new interrupt entry address to the VTOR register;

Turn on the interrupt controller of the MCU to enable interrupts;

Jump to the online upgrade function to start the online upgrade;

Turn off the interrupt controller of the MCU after upgraded and set VTOR as the default value of 0.

Perform a MCU soft reset, and the PC restarts the upgraded program from address 0.

7.2.1.8.2 Turn Off an Interrupted Online Upgrade

Turn off the interrupt controller of the MCU to stop receiving the new interrupt responses temporarily;

Jump to the online upgrade function to start the online upgrade; If the online upgrade uses UART-like peripheral communication, the MCU shoul poll the UART interrupt flag bit.

Perform a MCU soft reset, and the PC restarts the upgraded program from address 0.

7.2.1.8.3 Location of Online Upgrade Function

If the flash should be erased, place the online upgrade function in RAM; if an interrupt is required, place the new interrupt vector entry address in the RAM address space.

If only part of the flash area occupied by the application should be erased, place the online upgrade function in the free area of the high flash address, and then use the block to erase the old flash application and write the new application.

7.3 Register

7.3.1 Address assignment

The base address of the FLASH controller module registers is 0x0002_0000 and the register list is as follows:

Table 7-4 FLASH Controller Module Register List

Name	Offset	Description
FLASH_CFG	0x00	FLASH configuration register
FLASH_ADDR	0x04	Address Register
FLASH_WDATA	0x08	Write data register
FLASH_RDATA	0x0C	Read Data Register
FLASH_ERASE	0x10	Erase Control Register
FLASH_PROTECT	0x14	FLASH protection status register
FLASH_READY	0x18	FLASH working status register
FLASH_SIZE	0x1C	FLASH Capacity Status Register
FLASH_NCFG	0x20	FLASH configuration register (for NVR operation in protection state)
FLASH_NADDR	0x24	Address register (for NVR operation in protection state)
FLASH_NWDATA	0x28	Write Data Register (for NVR operation in protected state)
FLASH_NRDATA	0x2C	Read data register (for NVR operation in protection state)
FLASH_NERASE	0x30	Erase control register (for NVR operation in protection state)
FLASH_NKEY	0x34	FLASH key register (for NVR operation in protected state)

7.3.2 FLASH_CFG configuration register (recommended to read back first and modify by or/and)

Address: 0x0002 0000

Reset value: 0xE0

Table7-5 FLASH_CFG configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERS_EN				PRG_EN				ADR_INC				PREF			
RW				RW				RW				RW			
0				0				0				0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS_TYPE				REGION				WAIT		TBS					
RW				RW				RW		RW					
0				0				1		0x60					

Location	Bit name	Description
[31]	ERS_EN	FLASH erasure enable. The default value is 0. 0: Erasure off 1: Erasure on
[27]	PRG_EN	FLASH programming enable. The default value is 0. 0: programming off 1: Programming on
[23]	ADR_INC	FLASH address increment enable. The default value is 0. 0: Address increment off 1: Address increment on When performing FLASH continuous read and write access, enable this function can reduce the operation of the address.
[19]	PREF	FLASH prefetch acceleration enable. The default value is 0. 0: Acceleration off 1: Acceleration on
[15]	ERS_TYPE	FLASH erasure type selection. The default value is 0. 0: Sector 1: FULL
[11]	REGION	Access FLASH area selection. The default value is 0. 0: MAIN 1: NVR
[7]	WAIT	Read the FLASH data and wait for the switch. The default is 1. 0: Read, wait one cycle 1: Read, wait two cycles
[6:0]	TBS	Program/erase time base register. Default value is 0x60. **Only the following values can be set:** 0x60: FLASH program/erase time base configuration value at 96Mhz system frequency. 0x30: FLASH program/erase time base configuration value at 48Mhz system frequency.

		0x17: FLASH program/erase time base configuration value at 24Mhz system frequency.

7.3.3 FLASH_ADDR Address Register

Address: 0x0002 0004

Reset value: 0x0

Table 7-6 FLASH_ADDR Address Register

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

Location	Bit name	Description
[31:15]		Not used
[14:0]	ADDR	Address register. Address register corresponding to read/write/erase operation. The lowest two digits will be ignored by the FLASH controller because of the WORD operation. When performing the erase operation, the addresses should be aligned according to the erase type. One Sector is 512-Byte. If performing the Sector erase, the address should be an integer multiple of 512 (if offset, the offset will be ignored) If performing a full chip erase, the value of this register is not used for reference.

7.3.4 FLASH_WDATA Write Data Register

Address: 0x0002 0008

Reset value: 0x0

Table7-7 FLASH_WDATA Write Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
WO															
0															

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

Location	Bit name	Description
[31:0]	WDATA	Perform a write operation to write the value of FLASH

7.3.5 FLASH_RDATA Read Data Register

Address: 0x0002 000C

Reset value: 0x0

Table7-8 FLASH_RDATA Read Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA															
RO															
0															

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

Location	Bit name	Description
[31:0]	RDATA	Perform a read operation to read the value of FLASH

7.3.6 FLASH_ERASE Erase Control Register

Address: 0x0002 0010

Reset value: 0x0

Table7-9 FLASH_ERASE Erase Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERASE															
WO															
0															

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

Location	Bit name	Description
[31:0]	ERASE	Write 0x7654DCBA to trigger an erase operation

7.3.7 FLASH_PROTECT Protection Status Register

Address: 0x0002 0014

Reset value: 0x0

Table7-10 FLASH_PROTECT Protection Status Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PROTECT															
RO															
0															

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

Location	Bit name	Description
[31:0]	PROTECT	Read this register and update the encryption/decryption status. The read return value has no reference value.

7.3.8 FLASH_READY

Address: 0x0002 0018

Reset value: 0x0

Table7-11 FLASH_READY operating status register

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

Location	Bit name	Description
[31:1]		Not used
[0]	READY	1: FLASH is idle; 0: FLASH is busy

7.3.9 FLASH_SIZE capacity register

Address: 0x0002 001C

Reset value: 0x00

Table7-12 FLASH_SIZE capacity register

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

Location	Bit name	Description
[31:8]	--	Not used
[7:6]	FLASH_SIZE	FLASH capacity register, read only 0：32KB

		1： 64KB 2： 128KB 3： 128KB
[5:2]	--	Not used
[3:2]	ROM_SIZE	ROM capacity register, read only 0： 0KB 1： 32KB 2： 64KB 3： 0KB

7.3.10 FLASH_NCFG configuration register (recommended to read back first and modify by or/and)

Address: 0x0002 0020

Reset value: 0xB0

Table7-13 FLASH_NCFG configuration register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERS_EN				PRG_EN				ADR_INC				PREF			
RW				RW				RO				RW			
0				0				0				0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS_TYPE				REGION				WAIT		TBS					
RO				RW				RO		RO					
0				0				1		60					

Location	Bit name	Description
[31]	ERS_EN	FLASH erase enable. The default is 0. 0: Erasure off 1: Erasure on
[27]	PRG_EN	FLASH programming enabled. The default is 0. 0: programming off 1: Programming on
[23]	ADR_INC	FLASH address increment enabled. The default is 0. This register can only be read but not modified. 0: Turn off increment enable 1: Turn on increment enable When FLASH continuous read and write access is performed, this function can be turned on to reduce the operation on the address.
[19]	PREF	FLASH prefetch acceleration is enabled. The default is 0. 0: Acceleration off 1: Acceleration on
[15]	ERS_TYPE	FLASH erase type selection. The default is 0. This register can only be read but not modified. 0:Sector

		1:FULL
[11]	REGION	Access the FLASH area selection. The default is 0. Modifying this register, regardless of whether a value of 0 or 1 is written, can only be forced to write a 1 in the end. 0:MAIN 1:NVR
[7]	WAIT	Read the FLASH data and wait for the switch. The default is 1. This register can only be read but not modified. 0: Read, wait one cycle 1: Read, wait two cycles
[5:0]	TBS	Program/erase time base register. Default value is 0x60. Only the following values can be configured. This register can only be read but not modified. 0x60: FLASH program/erase time base configuration value at 96Mhz system frequency. 0x30: FLASH program/erase time base configuration value at 48Mhz system frequency. 0x17: FLASH program/erase time base configuration value at 24Mhz system frequency.

7.3.11 FLASH_NADDR Address Register

Address: 0x0002 0024

Reset value: 0x0

Table7-14 FLASH_NADDR Address Register

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

Location	Bit name	Description
[31:15]		Not used
[14:0]	ADDR	Address register. Address register corresponding to read/write/erase operation. The lowest two digits will be ignored by the FLASH controller because of the WORD operation. When performing the erase operation, the addresses should be aligned according to the erase type. One Sector is 512-Byte. If performing the Sector erase, the address should be an integer multiple of 512 (if offset, the offset will be ignored) If performing a full chip erase, the value of this register is not used for reference.

7.3.12 FLASH_NWDATA Write Data Register

Address: 0x0002 0028

Reset value: 0x0

Table7-15 FLASH_NWDATA Write Data Register

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

WDATA
WO
0

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

Location	Bit name	Description
[31:0]	WDATA	Perform a write operation to write the value of FLASH

7.3.13 FLASH_NRDATA Read Data Register

Address: 0x0002 002C

Reset value: 0x0

Table7-16 FLASH_NRDATA Read Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA															
RO															
0															

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

Location	Bit name	Description
[31:0]	RDATA	Perform a read operation to read the value of FLASH

7.3.14 FLASH_NERASE Erase Control Register

Address: 0x0002 0030

Reset value: 0x0

Table7-17 FLASH_NERASE Erase Control Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERASE															
WO															
0															

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

Location	Bit name	Description
[31:0]	ERASE	Write 0x7654DCBA to trigger an erase operation

7.3.15 FLASH_NKEY Key Register

Address: 0x0002 0034

Reset value: 0x0

Table7-18 FLASH_NKEY Key Register

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

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

Location	Bit name	Description
[31:4]	--	Not used
[3:0]	KEY	The FLASH cannot be read, programmed and erased by the SWD when the FLASH is in the protected state. At this time, the user can write a special value (0x0000_000F) to this register, enable the switch, configure the FLASH_NCFG/NADDR/NWDATA/NRDATA/NERASE registers, and read, program, and erase the NVR Sector 2.

7.3.16 NVR Calibration Parameter Address Information

LKS32MC07x product memory chip calibration parameters:

Calibration parameters, each product is independently calibrated, each product does not support the mixing of calibration parameters;

Calibration parameters are written before the factory, after the factory does not support programming and erasing, only support reading;

Calibration parameters are accessed via library functions provided by LKS;

Calibrate parameters. It is recommended to turn off the system interrupt before performing the access operation.

The lks32mc07x VR. o file contains the Read function of calibration parameters:

Read function: uint32 t Read Trim(uint32 t adr);

Table7-19 LKS32MC07x Address For Storing the Calibration Calue

地址	内容
0x00001420	ADC0 DC0 calibration, 3.6V range OFFSET
0x00001424	ADC0 AMC0 calibration value, 3.6V range GAIN
0x00001428	ADC0 DC1 calibration, 7.2V range OFFSET
0x0000142C	ADC0 AMC1 calibration value, 7.2V range GAIN
0x00001430	ADC1 DC0 calibration, 3.6V range OFFSET
0x00001434	ADC1 AMC0 calibration value, 3.6V range GAIN
0x00001438	ADC1 DC1 calibration, 7.2V range OFFSET
0x0000143C	ADC1 AMC1 calibration value, 7.2V range GAIN
0x00001450	DAC0 select 4.85V gear, SYS AFE DAC AMC calibration value (512 times enlarged result)
0x00001454	DAC0 select 4.85V gear, SYS AFE DAC DC calibration value
0x00001458	DAC0 select 1.20V gear, SYS AFE DAC AMC calibration value (512 times enlarged result)
0x0000145C	DAC0 select 1.20V gear, SYS AFE DAC DC calibration value
0x00001460	DAC1 select 4.85V gear, SYS AFE DAC AMC calibration value (512 times enlarged result)
0x00001464	DAC1 select 4.85V gear, SYS AFE DAC DC calibration value
0x00001468	DAC1 select 1.20V gear, SYS AFE DAC AMC calibration value (512 times enlarged result)
0x0000146C	DAC1 select 1.20V gear, SYS AFE DAC DC calibration value
0x00001480	OPA0, 320K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x00001484	OPA0, 160K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x00001488	OPA0, 80K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x0000148C	OPA0, 40K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x00001490	OPA1, 320K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x00001494	OPA1, 160K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x00001498	OPA1, 80K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x0000149C	OPA1, 40K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014A0	OPA2, 320K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014A4	OPA2, 160K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014A8	OPA2, 80K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014AC	OPA2, 40K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014B0	OPA3, 320K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014B4	OPA3, 160K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014B8	OPA3, 80K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014BC	OPA3, 40K ohms VS 10K ohms, GAIN calibration value (1000-fold enlargement result),R0 is 0 ohms.
0x000014C0	OPA0, common-mode voltage (10000 times enlarged result)
0x000014C4	OPA1, common-mode voltage (10000 times enlarged result)
0x000014C8	OPA2, common-mode voltage (10000 times enlarged result)
0x000014CC	OPA3, common-mode voltage (10000 times enlarged result)

0x000014D0	Temperature sensor, slope calibration value
0x000014D4	Temperature sensor, bias calibration value
0x00001500	OPA0, 320K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001504	OPA0, 160K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001508	OPA0, 80K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x0000150C	OPA0, 40K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001510	OPA1, 320K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001514	OPA1, 160K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001518	OPA1, 80K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x0000151C	OPA1, 40K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001520	OPA2, 320K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001524	OPA2, 160K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001528	OPA2, 80K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x0000152C	OPA2, 40K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001530	OPA3 320K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001534	OPA3, 160K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001538	OPA3, 80K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x0000153C	OPA3, 40K ohm :10K ohm gear. The high 16 bits store the actual resistance value of R2, the low 16 bits store the actual resistance value of R1 (enlarged by 100 times)
0x00001560	Offset value under OPA0 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 4x gain for OPA0 Bit[15:0]: Offset value at 8x gain for OPA0
0x00001564	Offset value under OPA0 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 16x gain for OPA0 Bit[15:0]: Offset value at 32x gain for OPA0
0x00001568	Offset value under OPA1 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 4x gain for OPA1 Bit[15:0]: Offset value at 8x gain for OPA1
0x0000156C	Offset value under OPA1 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 16x gain for OPA1 Bit[15:0]: Offset value at 32x gain for OPA1
0x00001570	Offset value under OPA2configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 4x gain for OPA2 Bit[15:0]: Offset value at 8x gain for OPA2
0x00001574	Offset value under OPA2 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 16x gain for OPA2 Bit[15:0]: Offset value at 32x gain for OPA2
0x00001578	Offset value under OPA3configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 4x gain for OPA3 Bit[15:0]: Offset value at 8x gain for OPA3

0x0000157C	Offset value under OPA3 configuration (signed type, unit: V, scaled by 10,000) Bit[31:16]: Offset value at 16x gain for OPA3 Bit[15:0]: Offset value at 32x gain for OPA3

8 SPI

8.1 Overview

The SPI interface is mainly used in application scenarios where the external design uses the SPI protocol. SPI working mode software is optional, the default is SPI Motorola mode. The SPI interface supports full-duplex transmission and half-duplex transmission. When the interface is set in Master mode, it can send clock signals for use by external Slave devices.

8.2 Main characteristics

- Support Master and Slave mode
- Support full-duplex transmission. Three or four signal lines can be used according to the application.
- Support half-duplex transmission. Twi signal lines can be used according to the application.
- Programmable clock polarity and phase
- Programmable data sequence: MSB or LSB
- The fastest transmission speed is 1/8 of the system's highest clock frequency
- Chip select signals are optional. In the Master mode, the chip select signal can be controlled by software or generated by hardware; in the Slave mode, the chip select signal can be constant and effective, or it can come from an external device
- No local FIFO, support DMA operation, including overflow detection and chip select signal anomaly detection
- Data length adjustable, 8-Bit ~ 16-Bit

8.3 Functional description

8.3.1 Functional block diagram

This interface uses a synchronous serial design to achieve SPI transmission between the MCU and external devices. and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

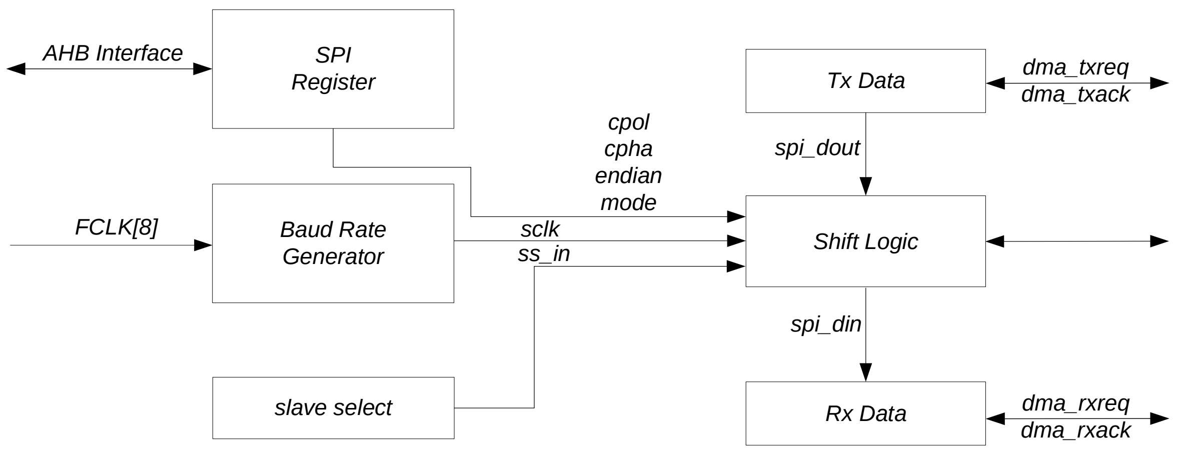


Figure 8-1 Structure block diagram of SPI module

Interface signals include spi_din, spi_dout, sclk_in, sclk_out, ss_in and ss_out.

Spi_din: data signal received by the interface. Compared with the SPI protocol, when the interface is configured in Master mode, it is equivalent to MISO; when the interface is configured in Slave mode, it is equivalent to MOSI.

spi_dout: data signal sent by the interface. Compared with the SPI protocol, when the interface is configured in Master mode, it is equivalent to MOSI; when the interface is configured in Slave mode, it is equivalent to MISO.

sclk_in: clock signal received by the interface. The working mode of the interface is Slave, and this signal input is invalid in non-Slave mode.

sclk_out: clock signal sent by the interface. The working mode of the interface is Master, and in non-Master mode, the signal output is always 0.

ss_in: chip select signal received by the interface. The working mode of the interface is Slave, and this signal input is invalid in non-Slave mode.

ss_out: chip select signal sent by the interface. The working mode of the interface is Master, and this signal output is always 1 in non-Master mode.

8.3.2 Functional description

8.3.2.1 Full duplex mode

The SPI interface is configured in full-duplex mode by default. Thus, two data lines are required for data transmission. The change of the data signal occurs on the edge of the clock signal, which is synchronized with the clock signal.

When the interface is in Master mode:

- spi_din is the data input, connected to the MISO of the external Slave device
- spi_dout is the data output, connected to the MOSI of the external Slave device
- spi_ss_out is a chip select signal, choose whether to use this signal or software to control

other GPIO implementation according to the application

When the interface is in Slave mode:

- spi_din is the data input, connected to the MOSI of the external master device
- spi_dout is the data output, connected to the MISO of the external master device
- spi_ss_in is the chip select signal, depending on whether the signal is used or the chip select is always valid

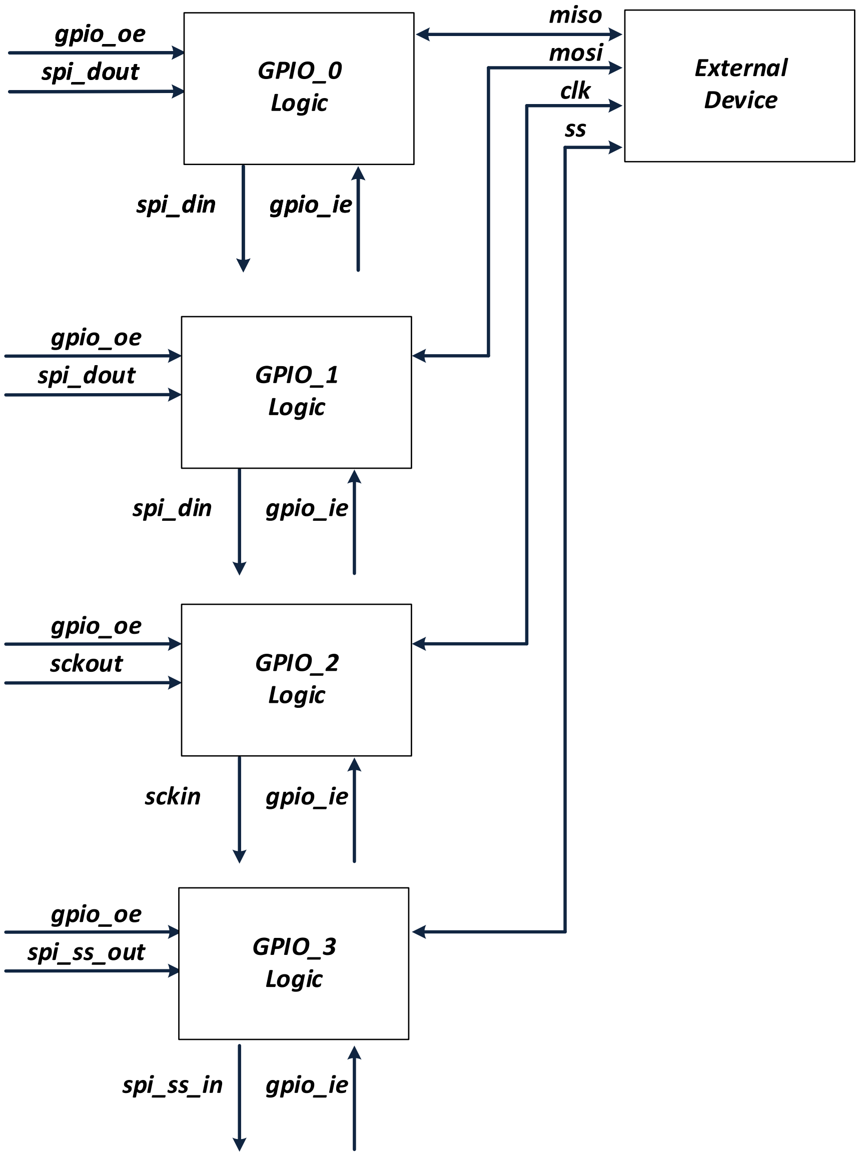


Figure 8-2 SPI Interface Full Duplex Interconnect Block Diagram

As can be seen from the above figure, if the GPIO is configured as an output, the SPI interface can send data; If the GPIO is configured as an input, the SPI interface can receive data.

8.3.2.2 Half-duplex mode

The SPI interface can be set in half-duplex mode. Thus, only one data line is needed for data transmission. The change of the data signal occurs on the edge of the clock signal, which is synchronized with the clock signal. A transmission can only be in one direction, either transmitting or receiving.

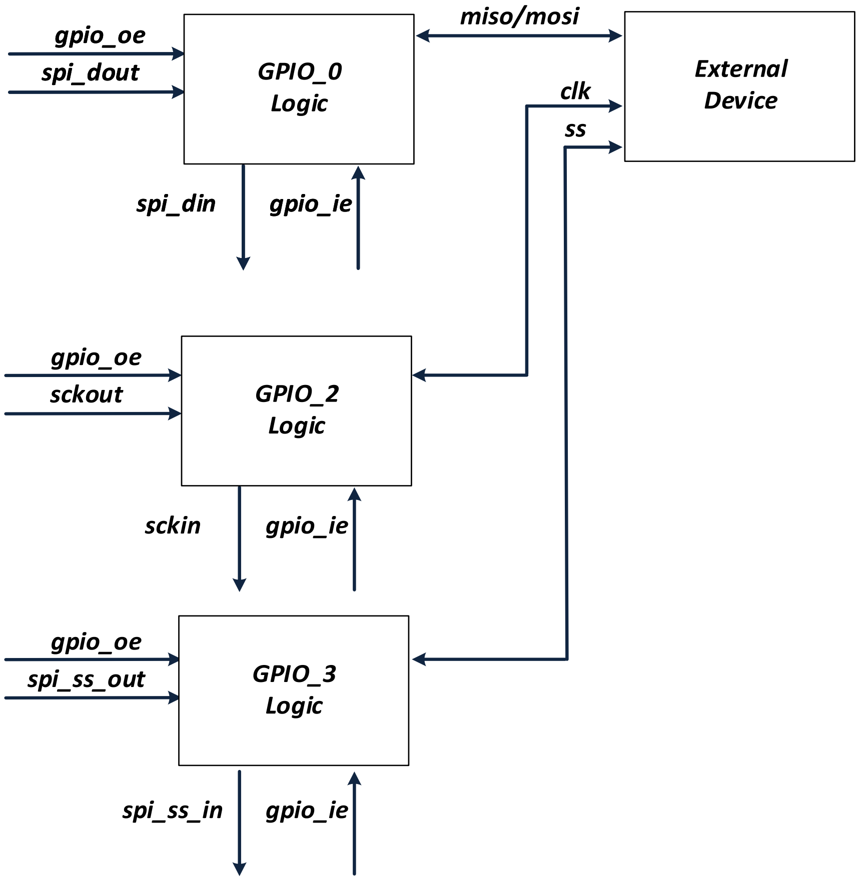


Figure 8-3 SPI Interface Interconnect Block Diagram in Half-Duplex Mode

Note that if the interface is a Master in the above figure, CLK is the output signal of the interface; if the interface is the Slave, CLK is the input signal of the interface.

Transmit only

SPI_CFG. DUPLEX is set to 2, and half-duplex transmission mode is valid. This interface can only transmit data. GPIO_0's oe is enabled, sending spi_dout data to the outside world; GPIO_0's ie is off, and the spi_din constant input is 0. It supports DMA transmission and supports sending in Master/Slave mode.

Receive only

SPI_CFG. DUPLEX is set to 3, and half-duplex reception mode is valid. This interface can only receive data. GPIO_0's oe is off, spi_dout cannot send data to the outside world; GPIO_0's ie is on, and spi_din receives data from the outside. In this mode, it supports DMA transmission and supports reception in Master/Slave mode.

Note that in full-duplex mode, two GPIOs are used for data transmission. In half-duplex mode, one GPIO can be selected for data transmission.

8.3.2.3 Chip select signal

When this interface is in Slave mode, the chip select signal is optional, and SPI_CFG. CS determines the chip select source. ss is the strobe enable signal sent by the master device. Active low.

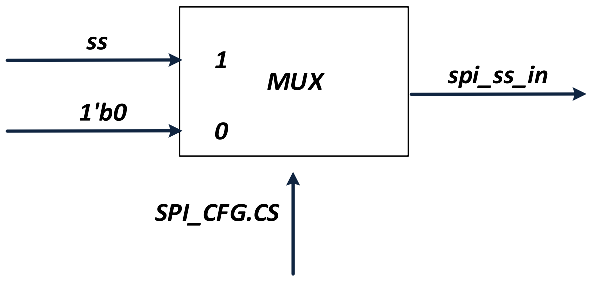


Figure 8-4 SPI module Slave mode chip select signal selection

When this interface is in Master mode, the chip select signal is also selectable. The module hardware generates a standard chip-select signal, which can be shielded by actual application by software operating additional GPIO.

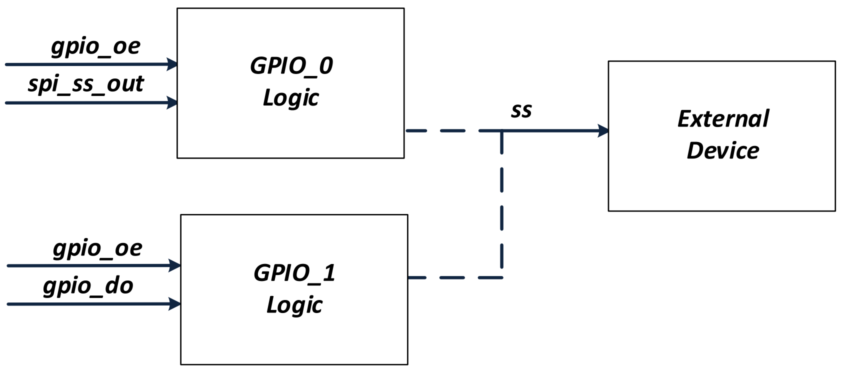


Figure 8-5 SPI module Master mode chip select signal selection

Note that the dotted line in Figure 8-5 only indicates uncertainty. If spi_ss_out is used as the source of ss, then GPIO_0 is interconnected with external devices; if software is used to operate GPIO, GPIO_1 can be interconnected with external devices.

8.3.2.4 Communication format

In the SPI communication process, the sending or receiving operation is based on the SPI clock. The communication format is controlled by SPI_CFG. SAMPLE and SPI_CLK_POL. SPI_CFG. SAMPLE is Phase control bit and SPI_CLK_POL is Polarity control bit.

Polarity controls the level status of the SPI clock signal by default. When Polarity is 0, the default clock level is low; when Polarity is 1, the default level is high.

Phase controls the transmission/reception time of SPI data. When Phase is 0, the clock transitions from the default level to the first transition edge is the time to sample data, and when Phase is 1, the clock transitions from the default level to the first transition edge is the time to transmit data.

8.3.2.5 Data format and length

SPI data transmission format is divided into two types: MSB and LSB. The data transmission format is controlled by CFG [1]. Note that the hardware automatically converts the transmission format during data transmission without software conversion.

The SPI data length is configurable from 8-Bit to 16-Bit. SPI_SIZE. BIT SIZE control length.

8.3.2.6 DMA transfer

In the application of large-capacity data transmission, the SPI interface supports DMA transmission, reducing the burden on the MCU. The maximum transmission volume of a transmission is 255 bytes, and the minimum transmission volume is one byte. In full-duplex mode, DMA transmission can be achieved for both reception and transmission; in half-duplex mode, only DMA transmission can be achieved for reception or transmission.

After receiving the new data, the hardware generates a DMA request automatically and moves the data to RAM through the DMA module. Before sending new data, the hardware automatically generates a DMA request, and moves the data from the RAM to the SPI interface through the DMA module. Since SPI has no FIFO, SPI transmits a DMA request, and DMA can only move a byte of data. To achieve multi-byte transmission, the DMA should be set for multiple rounds of transmission, with one byte transferred for each round.

Corresponding register of the DMA module should be set for DMA transmission.

Since the SPI interface supports DMA transmission, it also supports MCU transmission. The difference between the two is that the data sent by DMA transmission comes from the movement of DMA; the data sent by MCU transmission comes from the movement of MCU.

The recommended software configuration process for DMA transmission is as follows:

- Initialize the DMA module, set the data source sent this time, the destination of the received data, and the transmission length.
- Initialize the GPIO module and set the GPIO multiplexed with SPI.
- Initialize the SPI interface, and set SPI_IE/SPI_CFG/SPI_BAUD/SPI_SIZE and other registers.
- Trigger the SPI interface and enter the send/receive state. The trigger condition is that the MCU performs a write operation to the SPI_TXDATA register. Since the data finally sent is from the DMA, the data written by the MCU this time will not be mixed into the SPI transmission process.

8.3.2.7 MCU transmission

MCU transmission can only send/receive one byte at a time, and should judge whether the transfer is completed by interruption or polling after each completion. In either master or slave mode, write to the SPI_TXDATA register triggers a transfer. The master mode is active transmission, and the slave mode is to load data to the transmission queue and wait for the master mode to send a clock signal to start transmission. The recommended software configuration process is as follows:

- Initialize the GPIO module and set the GPIO multiplexed with SPI.
- The SPI interface is initialized, and the SPI_IE/SPI_CFG/SPI_BAUD/SPI_SIZE registers are configured.

- The MCU writes to the SPI_TXDATA register to trigger the SPI interface to enter the transmission process. In the slave mode, the data is loaded to the internal state machine to wait for the master device to initiate a read operation; In master mode, the trigger is sent.

Note that if continuous transmission is required, the SPI_TXDATA register needs to be reconfigured. The transmission completion information is obtained by means of interruption or polling.

8.3.2.8 External event transmission

Under the application of large-capacity data transmission, the SPI interface supports DMA transmission. During transmission, it may or may not be interrupted. The so-called interruption refers to the completion of the current byte transmission, and it needs to wait for an external event before starting the transmission of the next byte; the non-interruption refers to the completion of the current byte transmission and the transmission of the next byte. Interrupt mode should be used with other modules. For example, the timer module. The timer can trigger the transmission of the next byte. Interrupt mode is only valid in Master mode. SPI_IE.TRANS_TRIG controls whether to use interrupt mode.

8.3.2.1 Interrupt Handling

The SPI interface contains three types of interrupt events, including data transmission completion event, abnormal event and overflow event.

- Data transmission completion event. Current data has been transferred. Active high, write 1 to SPI_IE. CMPLT_IF to clear.
- For abnormal events, the SPI interface is in Slave mode. If the chip select signal is disturbed during transmission and is pulled high, a chip select abnormal event will occur. Active high, write 1 to SPI_IE. AB_IF to clear.
- Overflow event. If the data in SPI_RX_DATA is not returned, an overflow event will occur. Active high, write 1 to SPI_IE.OV_IF to clear.

The above events do not trigger the SPI interrupt by default, but can set SPI_IE[7:4] to enable the event to generate an interrupt.

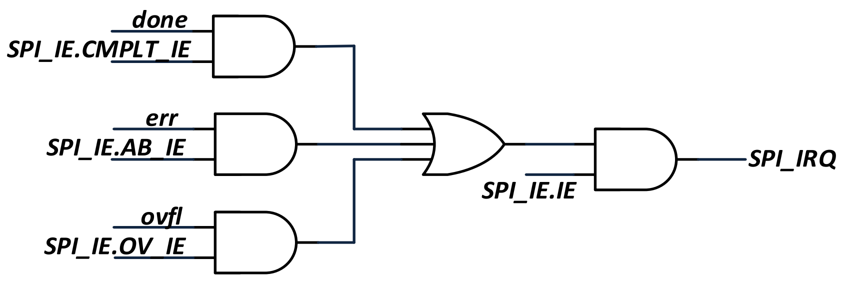


Figure 8-6 SPI module interrupt selection signal generation diagram

After the data transfer is completed, the end can be judged by DMA interrupt or SPI interrupt.

- In the transmission mode, the DMA first completes the moving operation, and the SPI transmits afterwards. After the SPI is sent, it can be used as the completion mark of this transmission.
- In the receiving mode, the SPI reception is completed, triggering the DMA move. The completion of DMA transfer can be used as the completion mark of this transfer.

Overflow event. In full-duplex mode, the DMAs sent and received are valid, and generally no overflow event will be sent; In half-duplex mode, there is only sending (or receiving), at this time the hardware blocks the receiving (or sending) overflow judgment.

8.3.2.2 Baud rate setting

The SPI interface clock is obtained by dividing the system clock by the frequency division factor from SPI_BAUD. BAUD. The SPI transmission baud rate configuration is calculated as:

SPI transmit baud rate = system clock/ (2 * (BAUD + 1))

The SPI protocol is a half-shot protocol. The rising edge sends data and the falling edge collects data; or the falling edge sends data and the rising edge uses data.

The SPI interface adopts a synchronous design, and the signals of external devices need to be synchronously sampled. The synchronous clock is the system clock. Synchronization of data and clock signals (Slave mode) requires two beats of the system clock. Considering the clock phase shift, the redundancy of the system clock is required at this time. It is deduced from this that the fastest BAUD rate is 1/8 of the system clock, the high-level period is four-beat system clock, and the low-level period is four-beat system clock. Therefore, the configuration value of the SPI_BAUD. BAUD cannot be less than 3.

8.4 Register

8.4.1 Address assignment

The base address of the SPI module registers is 0x4001_0000. The module registers are listed below.

Table 8-1 SPI Module Control Register List

Name	Offset	Description
SPI_CFG	0x00	SPI Configuration Register
SPI_IE	0x04	SPI Interrupt Register
SPI_BAUD	0x08	SPI baud rate register
SPI_TXDATA	0x0C	SPI Transmit Data Register
SPI_RXDATA	0x10	SPI Receive Data Register
SPI_SIZE	0x14	SPI Transfer Data Length Register

8.4.2 SPI_CFG SPI control register

Address: 0x4001 0000

Reset value: 0x0

Table 8-2 System Control Register SPI_CFG.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX		CS	MS	CPHA	CPOL	ENDIAN	EN
								RW		RW	RW	RW	RW	RW	RW
								0		1	0	0	1	0	0

Location	Bit name	Description
[31:8]		Not used
[7:6]	DUPLEX	Half-duplex mode setting 0X: turn off half-duplex mode 10: Turn on half-duplex mode, transmit only 11: Turn on half-duplex mode, receive only
[5]	CS	Source of chip select signal under SPI slave device. The default value is 1. 1: The chip select signal in Slave mode comes from the Master device 0: The chip select signal in Slave mode is always a valid value --0
[4]	MS	SPI master-slave mode selection. The default value is 0. 1: Master mode 0: Slave mode
[3]	CPHA	SPI phase selection. The default value is 0. 1: Phase is 1 0: Phase is 0
[2]	CPOL	SPI polarity selection. The default value is 0. 1: Polarity is 1 0: Polarity is 0
[1]	ENDIAN	SPI module transmission sequence. The default value is 0. 1: LSB, low bit is transmitted first 0: MSB, high bit is transmitted first
[0]	EN	SPI module enable signal. The default value is 0. 1: turn on the SPI module 0: turn off the SPI module

8.4.3 SPI_IE SPI Interrupt Register

Address: 0x4001 0004

Reset value: 0x0

Table 8-3 SPI_IE Interrupt Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	CMPLT_IE	AB_IE	OV_IE	TRANS_TRIG	CMPLT_IF	AB_IF	OV_IF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description

[31:8]		Not used
[7]	IE	SPI interrupt enable switch. The default value is 0. 1: enable SPI interrupt 0: disable SPI interrupt
[6]	CMPLT_IE	SPI transmission, complete event interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[5]	AB_IE	SPI transmission, abnormal event interrupt enable signal. 1: enable this interrupt source 0: disable this interrupt source
[4]	OV_IE	SPI transmission, interrupt enable signal for overflow event. The default value is 0. 1: enable this interrupt source 0: disable this interrupt source
[3]	TRANS_TRIG	Transmission trigger selection. 1: external trigger 0: internally executed automatically. Only the master mode is valid
[2]	CMPLT_IF	SPI transmission, complete event. Active high, write 1 to clear.
[1]	AB_IF	SPI transmission, abnormal events. In Slave mode, the transmission is not completed, and the chip select signal invalid event occurs. Active high, write 1 to clear.
[0]	OV_IF	SPI transfer, overflow event. The old data received last time has not been taken away, and the new data received this time has arrived. Active high, clear by writing 1.

8.4.4 SPI_BAUD SPI Baud Rate Register

Address: 0x4001 0008

Reset value: 0x0

Table 8-4 SPI_BAUD control register

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

Location	Bit name	Description
[31:9]		Not used
[15]	TRANS_MODE	SPI data transfer method. The default is 0, DMA mode. 0: The SPI interface supports DMA to move data to the SPI interface to complete data transfer and receiving. 1: The SPI interface supports the MCU to move data to the SPI interface to complete data transfer and receiving.
[14:12]		Unused
[11:0]	BAUD	SPI transmission baud rate configuration. SPI actual transmission speed calculation formula is: SPI transmission speed = system clock/(2*(BAUD+1))

		Remember, the set value of BAUD cannot be less than 3.

8.4.5 SPI_TXDATA SPI data transmit register

Address: 0x4001 000C

Reset value: 0x0

Table 8-5 SPI_TXDATA data transmit register

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	TX_DATA	SPI data transmit register

8.4.6 SPI_RXDATA SPI data receive register

Address: 0x4001 0010

Reset value: 0x0

Table 8-6 SPI_RXDATA data receive register

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	RX_DATA	SPI Data Receive Register

8.4.7 SPI_SIZE SPI Data Transfer Length Register

Address: 0x4001 0014

Reset value: 0x0

Table 8-7 SPI_SIZE Data Transfer Length Register

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

	0

Location	Bit name	Description
[31:5]		Not used
[4:0]	BITSIZE	Byte length register. 0x00: illegal value 0x07: illegal value 0x08:8-Bit 0x09:9-Bit ... 0x0E:14-Bit 0x0F:15-Bit 0x10:16-Bit

9 I2C

9.1 Overview

The I2C bus interface connects the microcontroller and the serial I2C bus. It provides multi-master functions to control the specific timing, protocol, arbitration and timing of all I2C buses. Besides, it supports standard and fast modes.

9.2 Main characteristics

- Multi-master function: this module can be used as both master and slave.

  I2C master device function: generate clock, START and STOP events.

  I2C slave device functions: programmable I2C hardware address comparison (only supports 7-bit hardware address), stop bit detection.

- Provide different communication speeds depending on the frequency division.
- Status flags: transmitter/receiver mode flag, byte transmission end flag, I2C bus busy flag.
- Error flags: Loss of arbitration in master mode, acknowledgment (ACK) error after address/data transmission, start or stop condition where misalignment was detected.
- An interrupt vector contains five interrupt sources: bus error interrupt source, completion interrupt source, NACK interrupt source, hardware address matching interrupt source, and transfer completion interrupt source.
- DMA with single byte buffer.

9.3 Functional description

9.3.1 Functional block diagram

This interface adopts a synchronous serial design to achieve I2C transmission between the MCU and external devices, and supports polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

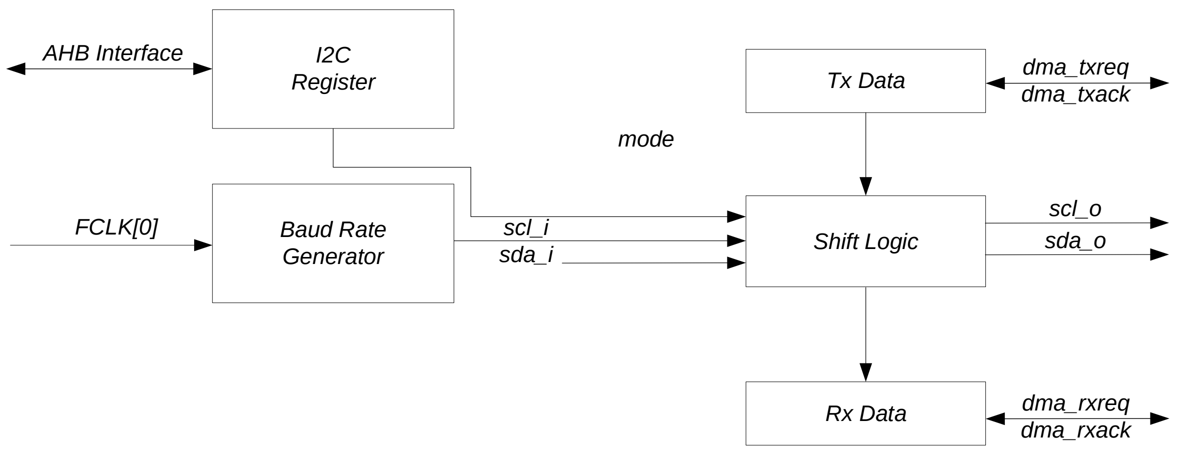


Figure 9-1 I2C Module Top Level Functional Block Diagram

The I2C interface communicates with the outside world only with two signal lines, SCL and SDA. SDA is a bidirectional multiplexed signal line, controlled by sda_oe. Module-level I2C interface signals include scl_i, sda_i, scl_o, sda_o, and sda_oe.

scl_i: clock signal. When the I2C interface is set as slave mode, this is the clock input signal for the I2C bus.

sda_i: data signal. When the I2C interface receives data (regardless of master mode or slave mode), this is the data input signal of the I2C bus.

scl_o: clock signal. When the I2C interface is set as the main mode, this is the clock output signal of the I2C bus.

sda_o: data signal. When the I2C interface sends data (regardless of master mode or slave mode), this is the data output signal of the I2C bus.

sda_oe: data enable signal. When sda_o is output, sda_oe is valid; when sda_i is input, sda_oe is invalid.

9.3.2 Functional description

The I2C module receives and sends data, and converts the data from serial to parallel, or parallel to serial , and can enable or disable interrupts. The interface is connected to the I2C bus via data pins (SDA) and clock pins (SCL).

9.3.2.1 Mode selection

The interface can operate in one of the following four modes:

- Slave tx mode
- Slave rx mode
- Master tx mode
- Master rx mode

The I2C interface is not enabled by default. The interface enters master mode or slave mode according to the configuration. When arbitration is lost or a stop signal is generated, the master mode releases the bus automatically and generates an abnormal interrupt. Multiple host functions is available.

In master mode, the I2C interface starts data transmission and generates a clock signal. Serial data transmission always starts with a start condition and ends with a stop condition. The start condition and stop condition are generated by software control in the master mode.

In slave mode, the I2C interface can recognize its own address (7 bits). The software can control to enable or disable the hardware address comparison function, which can reduce the burden on the MCU. Only when the addresses match, the MCU is notified to perform relevant processing.

The data and address are transmitted in 8 bits/byte, with the high order first. The one byte following the start condition is the address, and the address is only sent in master mode.

During the ninth clock after the eight clocks for one byte transmission, the receiver must send back an acknowledge bit (ACK) to the transmitter.

The software can enable or disable acknowledgement (ACK), and can set the address of the I2C interface.

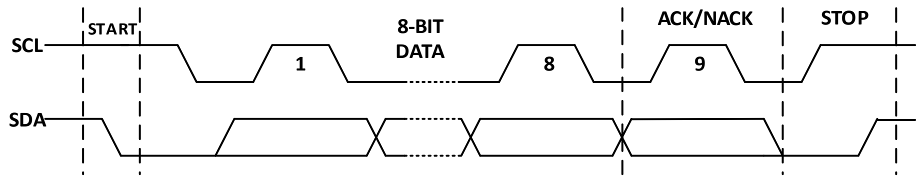


Figure 9-2 Basic I2C Transfer Timing Diagram

The I2C interface has no FIFO. If a large amount of data is sent at once, DMA cooperation is required to reduce the burden on the MCU. The I2C interface supports DMA transfer (multi-byte transfer) and non-DMA transfer (single-byte transfer). The above four transmission modes are further expanded to:

- Single-byte transmission in slave mode, DMA transmission in slave mode
- Single-byte reception in slave mode, DMA reception in slave mode
- Single-byte transmission in main mode, DMA transmission in main mode
- Single-byte reception in master mode, DMA reception in master mode

In general, one byte at a time is transmitted in non-DMA mode (single transmission can be repeated, and the data should be provided by software). A continuous transmission can be multi-byte in DMA mode. The maximum is no more than 16 bytes. In extreme cases, one byte is transmitted at a time. Since there is no FIFO, only one byte is transmitted per DMA request, and the data transmission is completed in multiple rounds.

All the above modes follow the basic principles below:

- Single byte transmission. An interrupt will be generated after the 8-bit data is sent and the response is received (either ACK/NACK).

- Single byte reception. An interrupt will be generated after the 8-bit data is received.
- DMA transmission. Normally, an interrupt will be generated after the data is sent and the response is received (either ACK/NACK).
- DMA reception. Normally, an interrupt will be generated after the data is received.
- When the I2C interface is set as the mastermode, the I2C interface will release the bus after detecting an error, restore to the initial state and generate an interrupt signal.

9.3.2.2 Slave mode

Both the master mode and slave mode of the I2C interface are turned off by default. If operating in slave mode, the slave mode should be enabled. In order to generate the correct timing, the operating clock frequency of the I2C interface must be set by the register SYS_CLK_DIV0. The I2C interface clock is divided based on the system high-speed main clock, and SYS_CLK_DIV0 is the division factor of the I2C interface working clock.

- In slave mode, the I2C interface monitors the signals on the bus at all times. Once the start condition is detected, it will save the address bit data and read-write bit data.
- In slave mode, if the hardware address matching function is enabled, an interrupt will be generated and the MCU will be notified for subsequent processing only when the addresses match. If the function is not enabled, an interrupt will be generated each time the address and read/write bit data is received.
- Single-byte reception in slave mode. Each time a byte of data is received, an interrupt is generated. The I2C interface can pull SCL low until the interrupt is completed and continue the subsequent operations.
- Single-byte transmission in slave mode. After each byte is sent and a response (ACK/NACK) is received, an interrupt is generated. The I2C interface can pull SCL low until the interrupt is completed and continue the subsequent operations.
- DMA reception in slave mode. Each time the data after the SIZE agreement is received, an interrupt is generated, and the I2C interface can pull the SCL low until the interrupt is completed.
- DMA transmission in slave mode. Each time the data after the SIZE agreement is sent and a response (ACK/NACK) is received, an interrupt is generated. The I2C interface can pull the SCL low until the interrupt is completed.

9.3.2.2.1 Slave Mode Transmission

Single-byte transmission does not mean that only one byte of data is transmitted. It means that after each byte of data is transmitted, an interrupt will be generated to determine whether to continue the transmission. The extreme case of single-byte transmission is to transmit only one byte of data. The following figure is a schematic diagram of the bus for single-byte transmission. As can be seen from the figure, the general single byte transmission process is as follows:

- Address match, generate address match interrupt, ready to start transmission.
- In the rx mode, an interrupt is generated after a byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In the tx mode, an interrupt is generated when receving the reponse (ACK/NACK) after a byte is sent, and subsequent operations are judged based on the response.
- Obtain the bus STOP event, this transmission is completed.

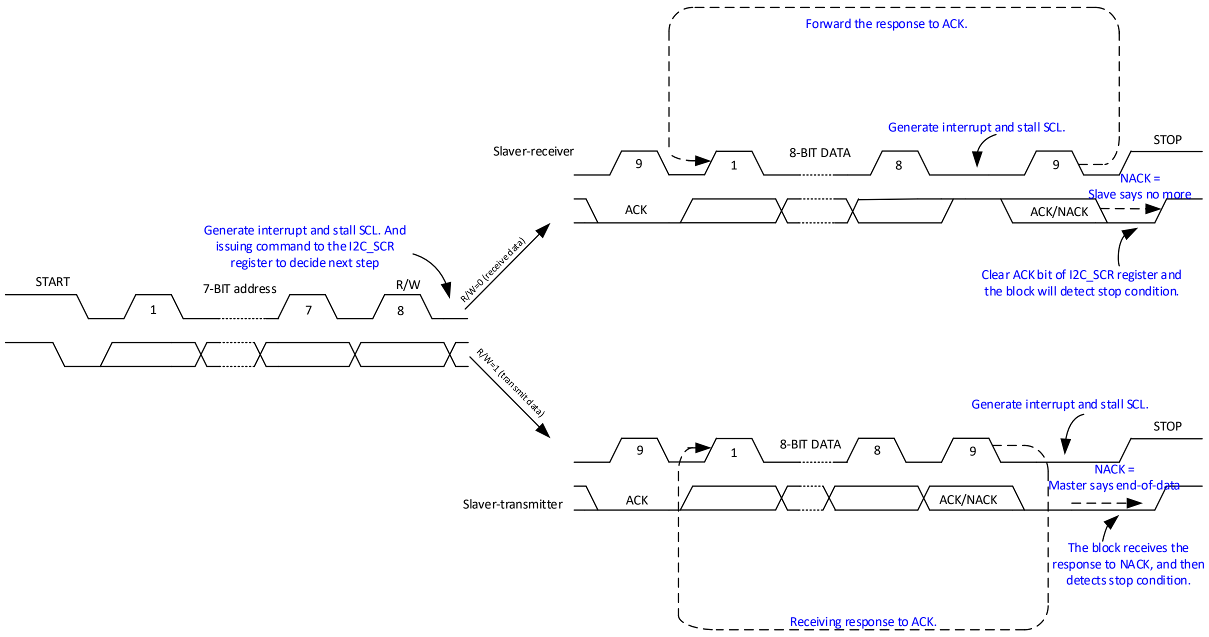

Forward the response to ACK.
Generate interrupt and stall SCL.
STOP
Slaver-receiver
9
1
8-BIT DATA
8
9
NACK =
Slave says no more
ACK
ACK/NACK
Generate interrupt and stall SCL. And issuing command to the I2C_SCR register to decide next step
R/W=0 (received data)
Clear ACK bit of I2C_SCR register and the block will detect stop condition.
START
7-BIT address
R/W
1
7
8
R/W=1 (transmit data)
Generate interrupt and stall SCL.
STOP
9
1
8-BIT DATA
8
9
NACK =
Master says end-of-data
Slaver-transmitter
ACK
ACK/NACK
The block receives the response to NACK, and then detects stop condition.
Receiving response to ACK.

Figure 9-3 Schematic diagram of slave mode transmission

9.3.2.2.2 Slave Mode Transport

After the address matches, the slave sends the byte from the I2C_DATA register to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the slave device can lower the SCL until the data to be sent has been written to the I2C_DATA register. The I2C interface performs the following operations after sending each byte:

- If the ACK bit is received, the next byte of data is loaded and the transmission continues. The SCL can be lowered during the loading process.
- If the NACK bit is received, stop loading the next byte.
- Wait for the STOP event to end this transmission.

9.3.2.2.3 Receive from Mode Single Byte

After the address matches, the slave receiver stores the data received from the SDA line through the internal shift register into the I2C_DATA register. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an ACK response pulse is generated after a byte is received.
- If the ACK bit is cleared, a NACK response pulse is generated after a byte is received.
- Wait for the STOP event to end this transmission.

9.3.2.3 Slave Mode DMA Transfer

Generally, only I2C clock, slave address and hardware address matching are enabled in slave mode. After waiting for an access request from the bus, determine whether to respond to this transmission request in situations. DMA transfer, which means that after each transfer of multiple bytes of data, an interrupt will be generated to determine whether to continue the transfer. The extreme case of DMA transfer is to transfer only one byte of data. Hardware address comparison function, NACK interrupt, and transfer completion interrupt are recommended to be enabled for DMA transmission. The general DMA transfer process is as follows:

- Set I2C slave address, enable I2C interrupt (enable hardware address comparison interrupt). Address matching, generate I2C address matching interrupt, set DMA in the interrupt processing function, ready to send data or ready to receive address. Then write I2C_SCR, ready to start transmission or end this transmission.
- In rx mode, an interrupt is generated after I2C_BSIZE.SIZE agreed byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In tx mode, wait for the response (ACK/NACK) after sending the agreed byte of I2C_BSIZE.SIZE, an interrupt is generated after receiving the response, and the subsequent operation is judged based on the response.
- Obtain the bus completion flag, this transmission is completed.

9.3.2.3.1 Slave Mode DMA Transmit

After the addresses match, the DMA is configured. Send a DMA request to move the byte from RAM to the I2C_DATA register, and then send it to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the slave device can pull the SCL until the data to be sent has been written to the I2C_DATA register. Sending data in slave mode requires software assistance to trigger the first DMA move, and set I2C_BCR.BYTE_CMPLT to 1. The I2C interface performs the following operations after sending the byte data agreed by I2C_BSIZE.SIZE:

- If the ACK bit is received, set the DMA, prepare for the next batch of data, and continue the transmission. SCL can be lowered during preparation.
- If the NACK bit is received, stop preparing the next batch of data and stop this transmission.

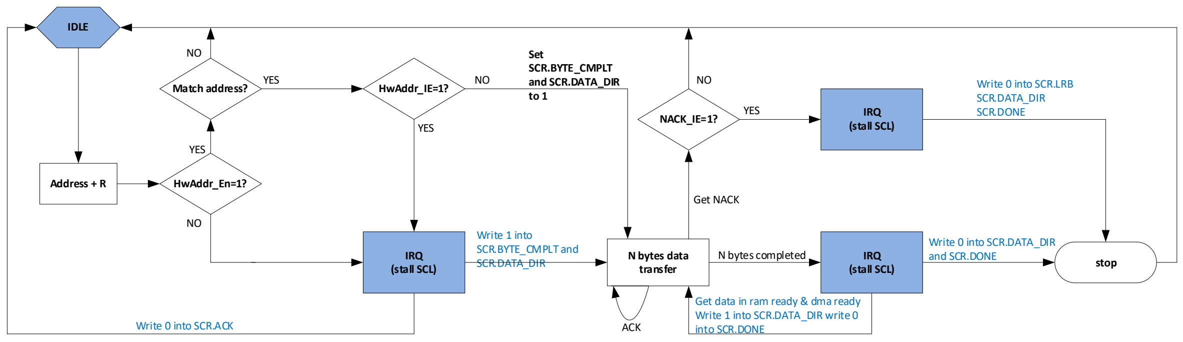

IDLE
Address + R
NO
Match address?
YES
YES
HwAddr_En=1?
NO
HwAddr_IE=1?
NO
YES
Set SCR.BYTE_CMPLT and SCR.DATA_DIR to 1
IRQ (stall SCL)
Write 1 into SCR.BYTE_CMPLT and SCR.DATA_DIR
Write 0 into SCR.ACK
N bytes data transfer
ACK
NACK_IE=1?
NO
YES
Get NACK
N bytes completed
IRQ (stall SCL)
Get data in ram ready & dma ready Write 1 into SCR.DATA_DIR write 0 into SCR.DONE
Write 0 into SCR.LRB SCR.DATA_DIR SCR.DONE
Write 0 into SCR.DATA_DIR and SCR.DONE
stop

Figure 9-4 Schematic diagram of multi-byte sending in slave mode

9.3.2.3.2 Slave Mode DMA Receive

After the addresses match, the DMA is configured, and the data received from the SDA line is stored in the I2C_DATA register, and then moved to the RAM through the DMA. The I2C interface will execute a DMA request after receiving a byte. After completing the data transfer agreed by I2C_BCR.SIZE, perform the following operations:

- If the ACK bit is set, an ACK response pulse is generated after I2C_BSIZE.SIZE agreed data is received.
- If the ACK bit is cleared, a NACK response pulse is generated after the I2C_BSIZE.SIZE agreed data is received.

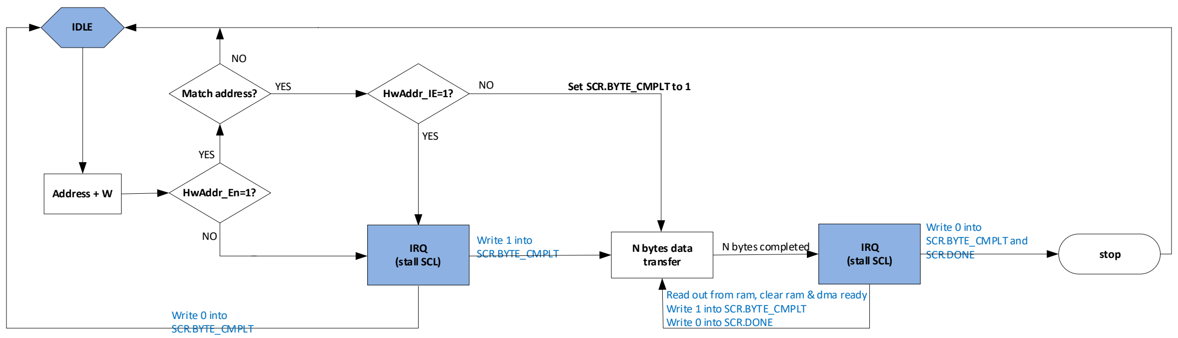

IDLE
Address + W
NO
Match address?
YES
YES
HwAddr_En=1?
NO
HwAddr_IE=1?
NO
YES
Set SCR.BYTE_CMPLT to 1
IRQ (stall SCL)
Write 1 into SCR.BYTE_CMPLT
Write 0 into SCR.BYTE_CMPLT
N bytes data transfer
N bytes completed
IRQ (stall SCL)
Read out from ram, clear ram & dma ready Write 1 into SCR.BYTE_CMPLT Write 0 into SCR.DONE
Write 0 into SCR.BYTE_CMPLT and SCR.DONE
stop

Figure 9-5 Schematic diagram of multi-byte receiving in slave mode

9.3.2.4 Master mode

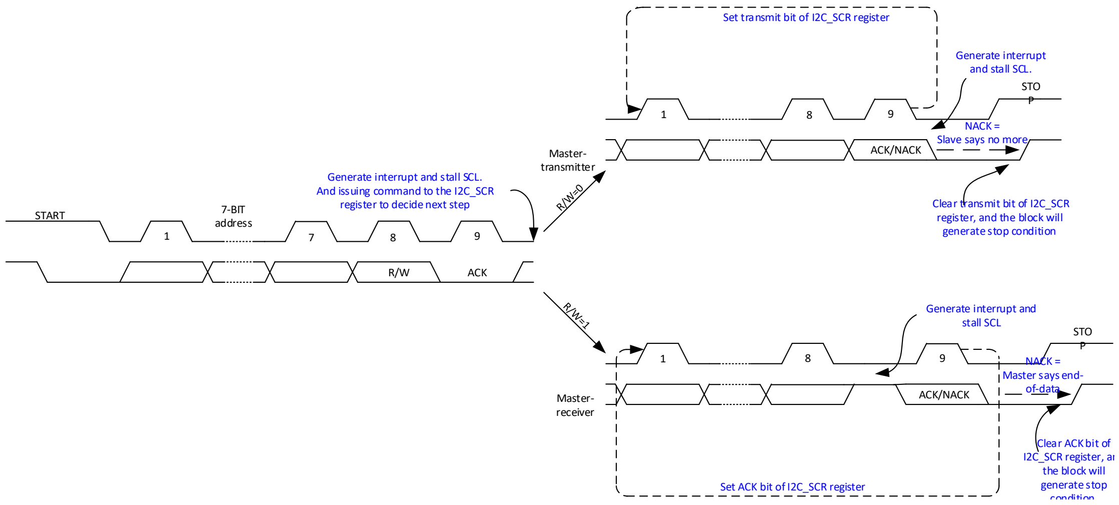
Both the master mode and slave mode of the I2C interface are turned off by default. If operating in the master mode, the master mode should be enabled. In order to generate the correct timing, the working clock of the I2C interface must be set in the register CLK_DIV0.

Judge whether the bus is idle before performing the transmission via I2C interface in master mode. Read BIT3 of the I2C_MSCR register to query the current bus status. If the bus is busy, turn on the I2C interrupt and determine whether the bus is idle by receiving the STOP interrupt event. Only in the idle state can the START state and subsequent data be sent normally.

9.3.2.4.1 Main mode single byte transfer

Single-byte transmission does not mean that only one byte of data is transmitted. It means that after each byte of data is transmitted, an interrupt will be generated to determine whether to continue the transmission. The extreme case of single-byte transmission is to transmit only one byte of data. Fig.9-6 is a schematic diagram of a bus for single-byte transmission. As can be seen from the figure, the general single byte transmission process is as follows:

- Determine whether the bus is idle, if it is idle, prepare to start transmission.
- In the rx mode, an interrupt is generated after a byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In the tx mode, an interrupt is generated when receving the reponse (ACK/NACK) after a byte is sent, and subsequent operations are judged based on the response.
- Send the bus STOP event, this transmission is completed.



START
1
7-BIT address
7
8
9
R/W
ACK
Generate interrupt and stall SCL. And issuing command to the I2C_SCR register to decide next step
R/W=0
R/W=1
Master-transmitter
Set transmit bit of I2C_SCR register
1
8
9
Generate interrupt and stall SCL.
STOP
NACK = Slave says no more
ACK/NACK
Clear transmit bit of I2C_SCR register, and the block will generate stop condition
Master-receiver
1
8
9
Generate interrupt and stall SCL
STOP
NACK = Master says end-of-data
ACK/NACK
Clear ACK bit of I2C_SCR register, and the block will generate stop condition
Set ACK bit of I2C_SCR register

Figure 9-6 Schematic diagram of single byte transmission in master mode

9.3.2.4.2 Master Mode Single Byte Send

After the transmission starts, the I2C interface stores the data received from the SDA line through the internal shift register in the I2C_DATA register. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an ACK response pulse is generated after a byte is received.
- If the ACK bit is cleared, a NACK response pulse is generated after a byte is received.
- A STOP event is generated to end this transmission.

9.3.2.4.3 Master Mode Single Byte Receive

After the transmission starts, the I2C interface stores the data received from the SDA line through the internal shift register in the I2C_DATA register. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an ACK response pulse is generated after a byte is received.
- If the ACK bit is cleared, a NACK response pulse is generated after a byte is received.
- A STOP event is generated to end this transmission.

9.3.2.4.4 Master mode DMA transfer

DMA transfer, which means that after each transfer of multiple bytes of data, an interrupt will be generated to determine whether to continue the transfer. The extreme case of DMA transfer is to transfer only one byte of data. NACK interrupt and the transmission completion interrupt are recommended to be enabled for DMA transmission. The general DMA transfer process is as follows:

- The bus is idle and ready to start transmission.
- In rx mode, an interrupt is generated after I2C_BSIZE.SIZE agreed byte is received, the software determines whether to continue receiving, and returns an ACK/NACK response.
- In tx mode, wait for the response (ACK/NACK) after sending the agreed byte of I2C_BSIZE.SIZE, an interrupt is generated after receiving the response, and the subsequent operation is judged based on the response.
- Send STOP event, this transmission is completed.

9.3.2.4.5 Master Mode DMA Transmit

The bus is idle and the DMA is configured. Send a DMA request to move the byte from RAM to the I2C_DATA register, and then send it to the SDA line via the internal shift register. Before the I2C_DATA data is ready, the master device may not generate the SCL clock until the data to be sent has been written to the I2C_DATA register. The I2C interface performs the following operations after sending the byte data agreed by SIZE:

- If the ACK bit is received, set the DMA, prepare for the next batch of data, and continue the transmission. SCL can be lowered during preparation.
- If the NACK bit is received, stop loading the next batch of data.
- If the data transmission is completed, stop the subsequent transmission.
- A STOP event is generated to end this transmission.

The exception is:

- If the slave device address does not match or the slave device is not ready, the slave device will return NACK.
- The master device generates a STOP event to stop this transmission.
- After waiting for a period of time, reset the I2C register, turn off the channel enable signal corresponding to the DMA, reset the DMA register, and send the transfer request again. The corresponding channel of DMA is closed since I2C has prefetch data, and DMA is not the initial state.

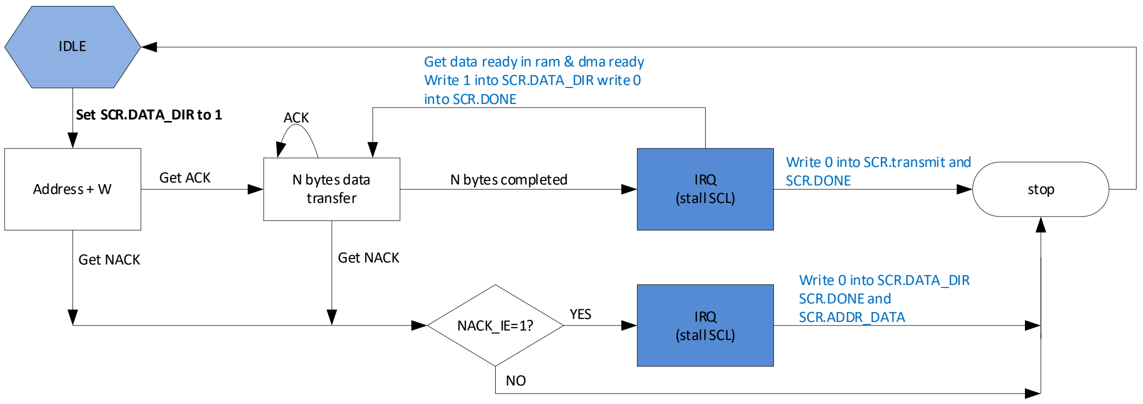

IDLE
Set SCR.DATA_DIR to 1
Address + W
Get ACK
ACK
N bytes data transfer
Get data ready in ram & dma ready
Write 1 into SCR.DATA_DIR write 0 into SCR.DONE
N bytes completed
IRQ (stall SCL)
Write 0 into SCR.transmit and SCR.DONE
stop
Get NACK
Get NACK
NACK_IE=1?
YES
NO
IRQ (stall SCL)
Write 0 into SCR.DATA_DIR SCR.DONE and SCR.ADDR_DATA

Figure 9-7 Schematic diagram of multi-byte transmission in master mode

9.3.2.4.6 Master mode DMA receive

The bus is idle, and the data received by the DMA from the SDA line is stored in the I2C_DATA register, and then moved to the RAM through DMA. The I2C interface will execute a DMA request after receiving a byte. After completing the data transfer agreed by I2C_BCR.BURST_SIZE, perform the following operations:

- If the ACK bit is set, an ACK response pulse is generated after I2C_BCR.BURST_SIZE agreed data is received.
- If the ACK bit is cleared, a NACK response pulse is generated after the I2C_BCR.BURST_SIZE agreed data is received.
- A STOP event is generated to end this transmission.

The exception is:

- If the slave device address does not match or the slave device is not ready, then the slave device will return NACK.
- The master device generates a STOP event to stop this transmission.
- After waiting for a period of time, reset the I2C register and send the transfer request again. Since no valid data was received last time, DMA had no operation, so there is no need to reset DMA.

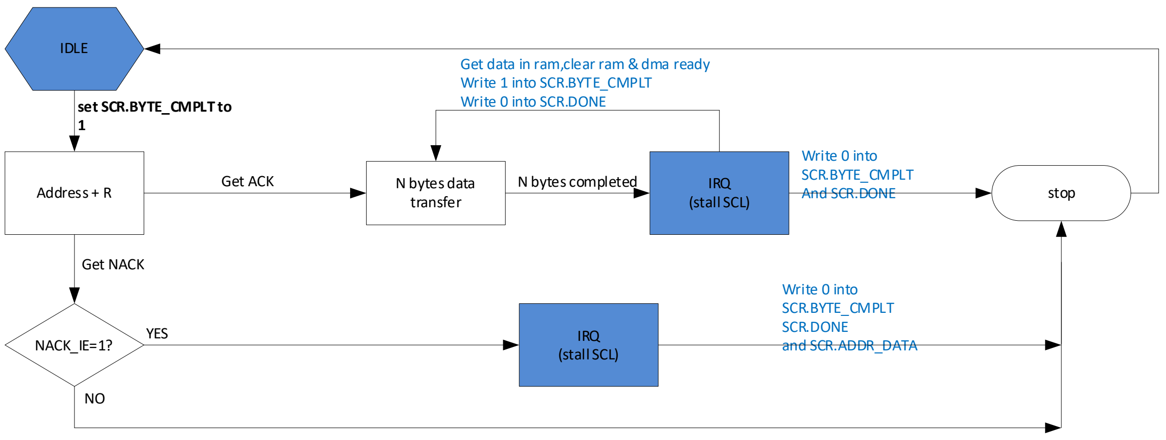


Figure 9-8 Schematic diagram of multi-byte reception in master mode

9.3.2.5 DMA transfer

Under the application of large-capacity data transmission, the I2C interface supports DMA transmission, thus reducing the burden on the MCU. The maximum transmission volume of a transmission is 16 bytes, and the minimum transmission volume is one byte. Since I2C has no FIFO, DMA can only move one byte of data after receiving the DMA request sent by I2C. To achieve multi-byte transmission, the DMA should be set for multiple rounds of transmission, with one byte transferred for each round.

After receiving the new data, the hardware generates a DMA request automatically and moves the data to RAM through the DMA module. Before sending new data, the hardwaregenerates a DMA request automatically and moves the data from RAM to the I2C interface through the DMA module.

Corresponding register of the DMA module should be set for DMA transmission.

The I2C interface supports both DMA transmission and MCU transmission. The difference between the two is that the data sent by DMA transmission comes from the movement of DMA, and the data sent by MCU transmission comes from the movement of MCU.

The recommended software configuration process for DMA transmission is as follows:

- Initialize the DMA module, set the data source sent this time, the destination of the received data, and the transmission length.
- Initialize the GPIO module and set the GPIO multiplexed with I2C.
- Initialize the I2C interface, set I2C_CFG, I2C_BCR and other registers.
- In master mode, trigger the I2C interface to enter the sending state; in slave mode, wait for the master to send a transmission request.

If the I2C interface is in slave mode for transmission, data prefetching should be considered, and I2C_BCR.SLV_DMA_PREF is the prefetching switch. Generally, the hardware address comparison (I2C_ADDR. ADDR_CMP) can be turned on during transmission in the slave mode, and choose whether to enable the hardware address comparison interrupt (I2C_BCR.BURST_ADDR_CMP).

- Turn on the hardware address comparison interrupt. After the address received from the device matches itself successfully, an interrupt will be generated. Inform the software that the master device requests the slave device data, and the software determines whether to receive it.
- If decide to receive, respond an ACK, and the software should set I2C_BCR. SLV_DMA_PREF to "1" to assist the hardware in prefetching the first transmitted data; otherwise, it should respond a NACK.
- Turn off the hardware address comparison interrupt. Once the START event occurs on the bus, the slave device hardware prefetches the first transmitted data, regardless of whether the slave device matches the address successfully. If matched successfully, the address match interrupt will not be generated and the data transmission will start immediately. If the match fails, the slave device will not transmit data. Since I2C has a prefetch operation, if the match fails, clear the DMA for the subsequent transmission if the next match is successful.

9.3.2.6 I2C bus exception handling

During an address or data byte transmission, a bus error is generated when the I2C interface detects an external stop or start condition. Generally speaking, the bus error is caused by interference on the bus, and some I2C devices are not synchronized with the I2C network, resulting in a START event/STOP event sent automatically. According to the I2C protocol, when a bus error occurs, the interface logic of this I2C device must be reset after receiving a START event/STOP event. For the slave device, this operation is OK; for the master device, a bus error will force it to release the bus and reset its I2C interface logic. Since the master device does not respond to external START and STOP events, an interrupt handler function is required to handle this exception after a bus error occurred, and instruct the master device to continue to monitor the bus situation, so as to perform subsequent I2C bus transmission.

For I2C interface: In master mode, bus errors can be detected and bus error interrupts can also be generated; in slave mode, bus errors will trigger address data to be received, while allowing the I2C interface to return to an idle state and generate interrupts.

9.3.2.7 Interrupt Handling

The I2C interface contains three types of interrupt events, including data transmission completion event, bus error event, STOP event, NACK event, and hardware address matching event.

- Data transmission completion event. The current data transmission is completed. Active high, write 0 to clear BIT0 of I2C_SCR.
- Bus error event. During the transmission process, the bus generates an erroneous START event/STOP event. Active high, write 0 to clear BIT7 of I2C_SCR.
- STOP event. When the current data transmission is completed, the master device sends a STOP event. The slave device receives a STOP event and generates a corresponding interrupt. Active high, write 0 to clear BIT5 of I2C_SCR.
- NACK event. The sender receives a NACK response, indicating that the receiver cannot continue subsequent transmissions. Active high, write 0 to clear BIT1 of I2C_SCR.

- Hardware address matching event. The address received in slave mode matches the address of this device, and a corresponding interrupt is generated. Active high, write 0 to clear BIT3 of I2C_SCR.

9.3.2.8 Communication speed setting

The operating clock of the I2C interface comes from the frequency division of the system clock, and the frequency division register is CLK_DIV0 of the SYS module.

I2C interface adopts synchronous design and needs to sample the signal of external equipment synchronously, and the synchronous clock is the working clock of I2C interface.

- I2C module operating clock frequency = system frequency/ (CLK_DIV0 + 1)
- The Serial data(SDA) and Serial clock line(SCL) clock frequency = I2C module operating clock frequency/17
- I2C baud rate = I2C module operating clock frequency/17

9.4 Register

9.4.1 Address assignment

The base address of the I2C module registers is 0x4001_0100, and the register list is as follows:

Table9-1 I2C register address allocation table

Name	Offset	Description
I2C_ADDR	0x00	I2C address register
I2C_CFG	0x04	I2C configuration register
I2C_SCR	0x08	I2C Status Register
I2C_DATA	0x0C	I2C Data Register
I2C_MSCR	0x10	I2C Master Mode Register
I2C_BCR	0x14	I2C transfer control register
I2C_BSIZE	0x18	I2C transfer length register

9.4.2 I2C_ADDR Address Register

Address: 0x4001 0100

Reset value: 0x0

Table 9-2 Address register I2C_ADDR

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

	0	0

Location	Bit name	Description
[31:8]		Not used
[7]	ADDR_CMP	I2C hardware address compare enable switch. Default value is 0. 0: Off 1: On
[6:0]	ADDR	For slave mode only, the I2C device hardware address. In master mode, the slave device address is written to the I2C_DATA register.

9.4.3 I2C_CFG system control register

Address: 0x4001 0104

Reset value: 0x0

Table 9-3 System control register I2C_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IE	TC_IE	BUS_ERR_IE	STOP_IE			MST_MODE	SLV_MODE
								RW	RW	RW	RW			RW	RW
								0	0	0	0			0	0

Location	Bit name	Description
[31:8]		Unused
[7]	IE	I2C interrupt enable signal. The default value is 0. 1: enable I2C interrupt 0: disable I2C interrupt
[6]	TC_IE	I2C data transfer completion interrupt enable signal. The default value is 0. 1: enable this interrupt source 0: disable this interrupt source
[5]	BUS_ERR_IE	I2C bus error event interrupt enable signal. The default value is 0. 1: enable this interrupt source 0: disable this interrupt source
[4]	STOP_IE	I2C STOP event interrupt enable signal. The default value is 0. 1: enable this interrupt source 0: disable this interrupt source
[3:2]		NA
[1]	MST_MODE	I2C master mode enable signal. The default value is 0. 1: enable master mode 0: disable master mode
[0]	SLV_MODE	I2C slave mode enable signal. The default value is 0. 1: enable slave mode 0: disable slave mode

9.4.4 I2C_SCR status control register

Address: 0x4001 0108

Reset value: 0x0

Table 9-4 Status control register I2C_SCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								STT_ERR	LOST_ARB	STOP_EVT	BYTE_CMPLT	ADDR_DATA	DATA_DIR	RX_ACK	Done
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Not used
[7]	STT_ERR	The bus error status flag bit is used for master mode transmit /master mode receive, and is cleared by writing 0. 0: No START/STOP bus error 1: START/STOP bus error
[6]	LOST_ARB	The bus arbitration lost status flag bit is used for master mode transmit/master mode receive. This bit is set to 1 when a bus arbitration lost event occurs. No interrupt event is generated. This bit is checked during the byte complete interrupt. Any START event on the bus will cause this bit to be cleared by hardware. 0: No bus arbitration lost error occurred 1: Bus arbitration lost error occurred
[5]	STOP_EVT	STOP event status flag bit for master mode transmit/slave mode transmit/master mode receive/slave mode receive. Clear by writing 0. 0: No STOP event 1: There is STOP event
[4]	BYTE_CMPLT	ACK control bit for master mode receive /slave mode receive. The receiver's response to the sender's completion of sending the current byte. In the case of the sender, this bit retains the value 0. The receiver is configured according to the actual situation. 0: The byte is sent completely, and a NACK response is returned, indicating that the receiver cannot receive more data 1: The byte is sent completely, and an ACK response is returned, indicating that the receiver can continue to receive data.
[3]	ADDR_DATA	The address data flag bit is used for master mode transmit /slave mode transmit /master mode receive /slave mode receive. After START, the first byte is the address data and this bit is a hint bit. Clear by writing 0. 0: The data sent or received is not Address data 1: The data sent or received is Address data
[2]	DATA_DIR	Transmit or receive control bit, master mode transmit/slave mode transmit, this bit is set to 1, trigger transmit, and the hardware is automatically cleared; Master mode receive/slave mode receive. This bit is set to 0, waiting for receive. 0: Receive 1: triggered sent

[1]	RX_ACK	The receiving response flag bit is used for sending in the master mode/sending in the slave mode to inform the sender and the receiver of the feedback. After receiving the feedback, the sender clears the bit. 0: The I2C interface sends data and receives ACK response. 1: The I2C interface sends data and receives a NACK response.
[0]	Done	The transmission completion status flag bit is used for master mode transmit /slave mode transmit /master mode receive /slave mode receive. Clear by writing 0. 0: transmission undone 1: transmission done

Generally, after entering the interrupt, read the I2C_SCR register to get the current I2C bus status and what stage the current transmission is in; then, write different values to the I2C_SCR, and the software informs the hardware what to do next.

9.4.5 I2C_DATA Data Register

Address: 0x4001 010C

Reset value: 0x0

Table 9-5 Data register I2C_DATA

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

Location	Bit name	Description
[31:8]		Not used
[7:0]	DATA	Data register for master mode transmit/slave mode transmit/master mode receive/slave mode receive. The sender writes data to trigger transmission; The receiver reads the received data. Note that address data is also data, and the main mode can only write the address data to be sent to this register.

9.4.6 I2C_MSCR Master Mode Register

Address: 0x4001 0110

Reset value: 0x0

Table 9-6 Master Mode Register I2C_MSCR

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

	RW	RW	RW	RW
	0	0	0	0

Location	Bit name	Description
[31:4]		Not used
[3]	BUSY	I2C-bus, idle busy state. 0: STOP event detected, idle. 1: START event detected, busy.
[2]	MST_CHECK	Master mode scrambles for the bus flag. If the bus is scrambled, set to 1; if the STOP event or bus collision occurs, the module releases the bus and sets to 0
[1]	RESTART	Trigger the START event again and write 1 is valid. After sending START, the hardware is cleared to 0. Set I2C_CFG [1] to 1 to achieve write "1" operation.
[0]	START	Trigger START event and send address data to the bus, write 1 is valid. Set I2C_CFG [1] to 1 to achieve write "1" operation.

9.4.7 I2C_BCR I2C transfer control register

Address: 0x4001 0114

Reset value: 0x0

Table 9-7 DMA transfer control register I2C_BCR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BURST_NACK	BURST_ADDR_CMP	BUSRT_EN	SLV_DMA_PREF				
								RW	RW	RW	RW				
								0	0	0	0				

Location	Bit name	Description
[31:8]		Not used
[7]	BURST_NACK	I2C transfer, NACK event interrupt enable signal. 1: Enable this interrupt source 0: Mask this interrupt source
[6]	BURST_ADDR_CMP	I2C transfer, hardware address match interrupt enable signal. 1: Enable this interrupt source 0: Mask this interrupt source
[5]	BUSRT_EN	I2C multiple data transmission is enabled. DAM is required. 1: enable 0: disable
[4]	SLV_DMA_PREF	I2C multiple data transmission. Perform DMA transmission in slave mode, triggering the hardware to prefetch the first byte. The hardware is automatically cleared. 1: enable 0: disable
[3:0]		Not used

10 CMP

10.1 Overview

The comparator signal processing module (Hereinafter referred to as CMP module. For better distinguish, the analog comparator in the following figure is represented by Comparator, and the digital CMP module is represented by CMP) is used to process the output signals generated by the three analog rail-to-rail comparators and consists of a series of digital circuits such as enable, polarity control, and filtering. The signal processing clock is obtained by dividing the system master clock, and this module is also used to generate the comparator interrupt to the CPU.

CMP0/1 can use the 4-way P channel of MCPWM for windowing control.

The unfiltered original output value of the analog comparator can be obtained by reading the CMP_DATA value. The unfiltered original output value of the analog comparator can also be sent out by configuring the function AF1 of GPIO. For specific GPIO second function configuration and introduction location, please refer to the device datasheet.

For more information about the analog comparator, including the selection of its input signals and the configuration of its hysteresis, please refer to Section0。

10.2 Register

10.2.1 Address assignment

The base address of the CMP module registers is 0x4001_0200, and the register list is as follows:

Table 10-1 List of comparator registers

Name	Offset address	Description
CMP_IE	0x00	Comparator interrupt enable register
CMP_IF	0x04	Comparator interrupt flag register
CMP_TCLK	0x08	Comparator divider clock control register
CMP_CFG	0x0C	Comparator control register
CMP_BLCWIN	0x10	Comparator windowing control register 0
CMP_DATA	0x14	Comparator output value register

10.2.2 CMP_IE Interrupt enable register

Address: 0x4001 0200

Reset value: 0x0

Table 10-2 CMP_IE Comparator Interrupt Enable Register

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

	CMP2_RE	CMP1_RE	CMP0_RE		CMP2_IE	CMP1_IE	CMP0_IE
	RW	RW	RW		RW	RW	RW
	0	0	0		0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	CMP2_RE	Comparator 2 DMA request enable, active high
[9]	CMP1_RE	Comparator 1 DMA request enable, active high
[8]	CMP0_RE	Comparator 0 DMA request enable, active high
[7:3]		Not used
[2]	CMP2_IE	Comparator 2 interrupt enable, active high
[1]	CMP1_IE	Comparator 1 interrupt enable, active high
[0]	CMP0_IE	Comparator 0 interrupt enable, active high

10.2.3 CMP_IF Interrupt Flag Register

Address: 0x4001 0204

Reset value: 0x0

Table 10-3 CMP_IF Comparator Interrupt Flag register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													CMP2_IF	CMP1_IF	CMP0_IF
													RW	RW	RW
													0	0	0

Location	Bit name	Description
[31:2]		Not used
[2]	CMP2_IF	Comparator 2 interrupt flag, active high, write 1 to clear
[1]	CMP1_IF	Comparator 1 interrupt flag, active high, write 1 to clear
[0]	CMP0_IF	Comparator 0 interrupt flag, active high, write 1 to clear

When the CMP DMA request is enabled, that is, CMP_IE. CMPx_RE = 1, the DMA automatically clears the corresponding CMPx_IF bit when the DMA receives the corresponding request and starts to move data, and software clearing is no longer required.

10.2.4 CMP_TCLK Divided clock control register

Address: 0x4001 0208

Reset value: 0x0

Table 10-4 CMP_TCLK comparator divided clock control register

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

	COMM_DIV	FIL2_CLK_DIV16	CLK2_EN	FIL2_CLK_DIV2
	RW	RW	RW	RW
	3’h4	0	0	0

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

FIL1_CLK_DIV16	CLK1_EN	FIL1_CLK_DIV2	FIL0_CLK_DIV16	CLK0_EN	FIL0_CLK_DIV2
RW	RW	RW	RW	RW	RW
0	0	0	0	0	0

Location	Bit name	Description
[31:27]		Not used
[26:24]	COMM_DIV	Comparators 0/1/2 share the filter clock division factor, which affects the time to enter the comparator interrupt and supports up to 4
[23:20]	FIL2_CLK_DIV16	Comparator 2 filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[19]	CLK2_EN	Comparator 2 Filter Clock Enable, Active High
[18:16]	FIL2_CLK_DIV2	Comparator 2 Filter Clock Divide 0: 1 frequency division 1: 2 frequency division 2: 4 frequency division 3: 8 frequency division 4: 16 frequency division 5: 32 frequency division 6: 64 frequency division 7: 128 frequency division
[15:12]	FIL1_CLK_DIV16	Comparator 1 filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[11]	CLK1_EN	Comparator 1 Filter Clock Enable, Active High
[10:8]	FIL1_CLK_DIV2	Comparator 1 Filter Clock Divide 0: 1 frequency division 1: 2 frequency division 2: 4 frequency division 3: 8 frequency division 4: 16 frequency division 5: 32 frequency division 6: 64 frequency division 7: 128 frequency division
[7:4]	FIL0_CLK_DIV16	Comparator 0 filter clock frequency division, based on MCLK to divide by 1 to 16, affecting the time to enter the comparator interrupt
[3]	CLK0_EN	Comparator 0 Filter Clock Enable, Active High
[2:0]	FIL0_CLK_DIV2	Comparator 0 Filter Clock Divide

		0: 1 frequency division 1: 2 frequency division 2: 4 frequency division 3: 8 frequency division 4: 16 frequency division 5: 32 frequency division 6: 64 frequency division 7: 128 frequency division

Taking comparator 0 as an example, the filter time width of CMP0 is calculated by the following formula

Filtering time=Period(FCLK)*(2$^{(\text{COMM\_DIV+CMP\_TCLK.FIL0\_CLK\_DIV2})}$*(CMP_TCLK.FIL0_CLK_DIV16+1)-1), where FCLK is the master clock of the CMP module and is controlled by the SYS_CLK_FEN. Note that the CMP_TCLK.CLK_EN bit should be enabled to generate the CMP filter clock.

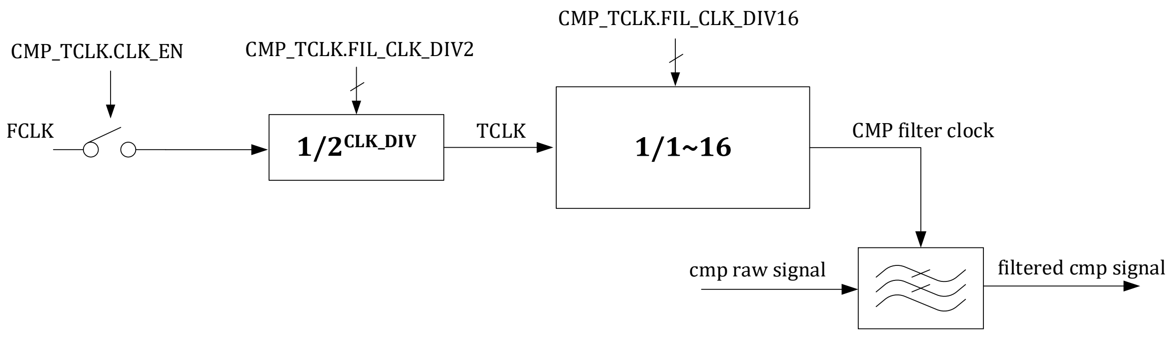


Figure 10-1 Comparator filter clock generation

The CMP module uses this filtering clock to filter the output signal of the analog comparator, that is, only when the signal stability time exceeds the set filtering time width can it pass through the filter, and the filtered signal output by the CMP module will change and generate a corresponding interrupt. And if that stable time of the input signal is less than the filter time width, the filtered signal output by the CMP module keep the original value unchanged.

Since the filter clock can be divided down, the filter width of the CMP signal ranges from 0 to 128 * 16 * 16-1 bus cycles, i.e., 1 to 32767 system master clock cycles.

10.2.5 CMP_CFG Control Register

Address: 0x4001_020C

Reset value: 0x0

Table 10-5 CMP_CFG Comparator Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					CMP2_IRQ_TRIG	CMP2_IN_EN	CMP2_POL	CMP1_W_PWM_POL	CMP1_IRQ_TRIG	CMP1_IN_EN	CMP1_POL	CMP0_W_PWM_POL	CMP0_IRQ_TRIG	CMP0_IN_EN	CMP0_POL
					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
					0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	CMP2_IRQ_TRIG	Comparator 2 interrupt trigger type, 0: level triggered, 1: edge triggered
[9]	CMP2_IN_EN	Comparator 2 signal input enabled
[8]	CMP2_POL	Comparator 2 polarity selection, 0: active high; 1: active low
[7]	CMP1_W_PWM_POL	Comparator 1 windowed PWM signal polarity selection, used when CMP_BLCWIN1 is enabled
[6]	CMP1_IRQ_TRIG	Comparator 1 interrupt trigger type, 0: level triggered, 1: edge triggered
[5]	CMP1_IN_EN	Comparator 1 signal input enabled
[4]	CMP1_POL	Comparator 1 polarity selection, 0: active high; 1: active low
[3]	CMP0_W_PWM_POL	Comparator 0 windowed PWM signal polarity selection, used when CMP_BLCWIN0 is enabled
[2]	CMP0_IRQ_TRIG	Comparator 0 interrupt trigger type, 0: level triggered, 1: edge triggered
[1]	CMP0_IN_EN	Comparator 0 signal input enabled
[0]	CMP0_POL	Comparator 0 polarity selection, 0: active high; 1: active low

The polarity and enable control of the comparator are as follows:

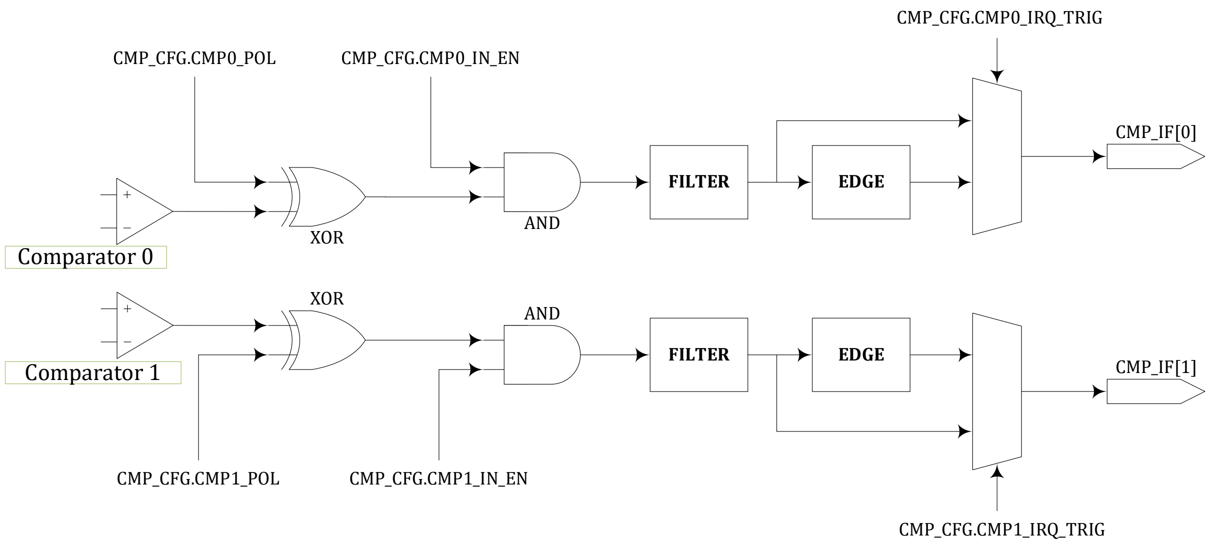


Figure 10-2 Comparator control and interrupt generation logic

The comparator 0/1 and the MCPWM module can work together, and the P-tube control signal of the MCPWM module can be used as the control signal for comparator windowing. However, the interrupt signal of the comparator itself is generated regardless of the window control and is only affected by the CMP_CFG register. Comparator 2 does not support windowing control.

The fail signal of MCPWM can come from GPIO or from the comparator module, and is controlled by the MCPWM_FAIL register. If the fail signal of MCPWM comes from the comparator, it is controlled by the window inside the comparator module. After the fail signal enters MCPWM, it will also be processed with polarity enable and filtering. It is similar to the comparator module but completely independent, and is controlled by the register inside MCPWM. The error interrupt signal related to fail in MCPWM is affected by the polarity-enable filter control register in MCPWM. For details, please refer to the MCPWM chapter.

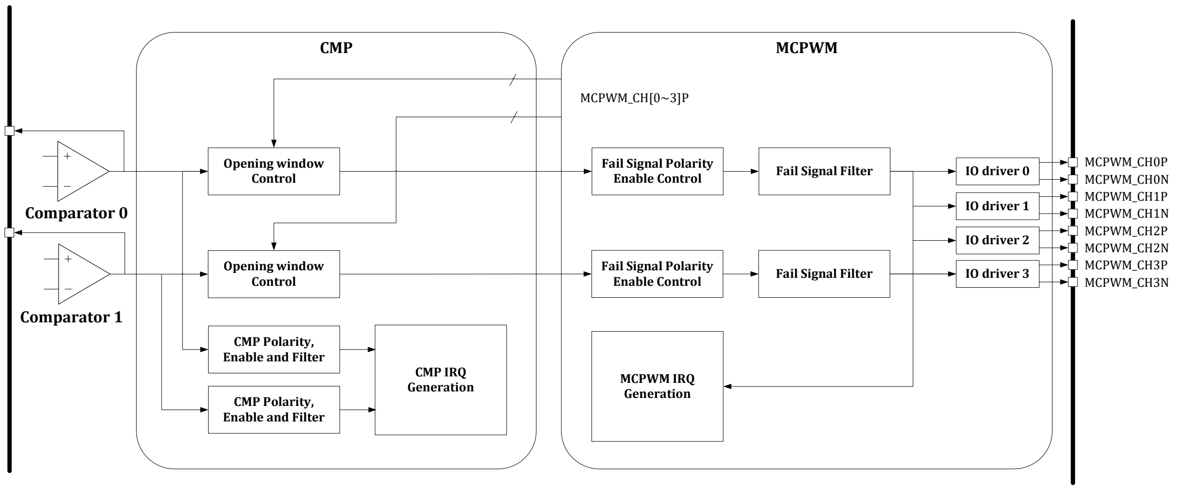


Figure 10-3 Linkage of CMP and MCPWM

For the windowing function of the comparator, if CMP_CFG.CMP0_PWM_POL = 1, then when the corresponding MCPWM CHNx_P signal is 1, the comparator 0 can generate a comparison signal output, and the comparison signal is 0 at other times; Conversely, if CMP_CFG.CMP0_PWM_POL = 0, then when the corresponding MCPWM CHNx_P signal is 0, the comparator 0 can generate a comparison signal output, and the comparison signal is 0 at other times. The window control signal polarity of the comparator 1 is controlled by the CMP_CFG.CMP1_PWM_POL bit, and the logic is the same.

Note: CMP_CFG.CMP0_PWM_POL and CMP_CFG.CMP1_PWM_POL will also affect the comparator signal sent to the MCPWM module as a fail signal, as shown in Figure10-4. The MCPWM FAIL signal from the comparator is the original signal output by the analog comparator, which is not filtered by the comparator digital interface module, but can be windowed by the MCPWM channel signal. See the comparator digital interface module for windowing control settings. After the FAIL signal enters the MCPWM block, it can be filtered by setting the MCPWM_TCLK.

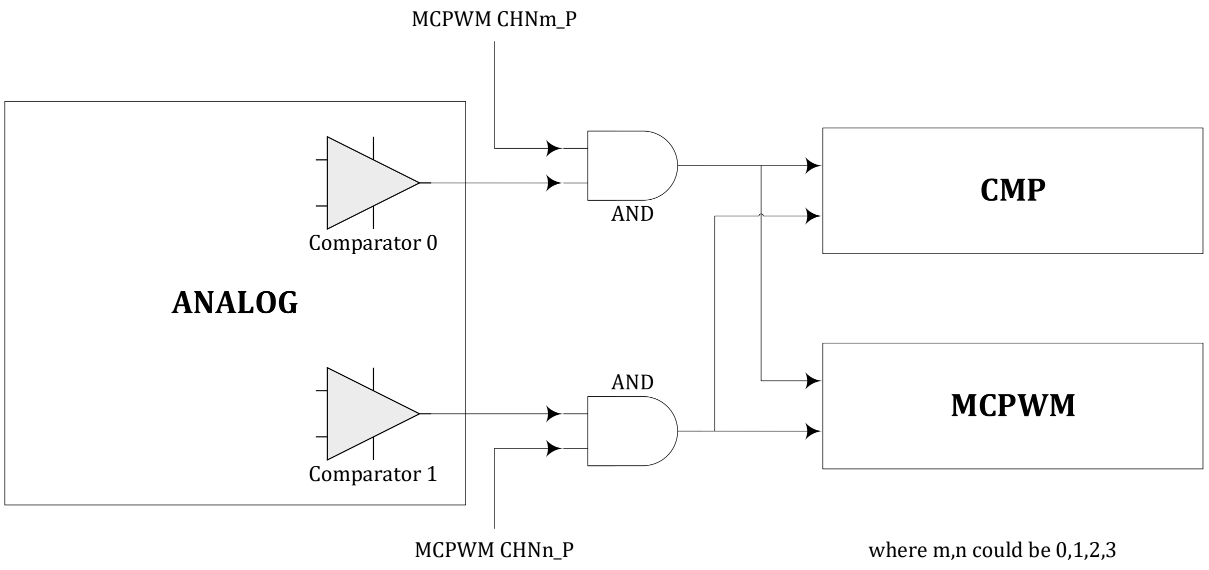


Figure 10-5 Illustration of comparator windowing function

10.2.6 CMP_BLCWIN Windowing control register

Address: 0x4001 0210

Reset value: 0x0

Table 10-6 CMP_BLCWIN Comparator Windowing Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CMP1_CHN3P_WIN_EN	CMP1_CHN2P_WIN_EN	CMP1_CHN1P_WIN_EN	CMP1_CHN0P_WIN_EN	CMP0_CHN3P_WIN_EN	CMP0_CHN2P_WIN_EN	CMP0_CHN1P_WIN_EN	CMP0_CHN0P_WIN_EN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:8]		Reserved
[7]	CMP1_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM module as the comparator 1 window enable
[6]	CMP1_CHN2P_WIN_EN	Use the P tube switch control signal output from the CHN2_P channel of the CHPWM module as the comparator 1 window enable
[5]	CMP1_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM module as the comparator 1 window enable
[4]	CMP1_CHN0P_WIN_EN	Use the P-tube switch control signal output from the CHN0_P channel of the MCPWM module as the comparator 1 window enable

[3]	CMP0_CHN3P_WIN_EN	Use the P-tube switch control signal output from the CHN3_P channel of the MCPWM module as the comparator 0 window enable
[2]	CMP0_CHN2P_WIN_EN	Use the P tube switch control signal output from the CHN2_P channel of the CHPWM module as the comparator 0 window enable
[1]	CMP0_CHN1P_WIN_EN	Use the P-tube switch control signal output from the CHN1_P channel of the MCPWM module as the comparator 0 window enable
[0]	CMP0_CHN0P_WIN_EN	Use the P-tube switch control signal output by the CHN0_P channel of the MCPWM module as the comparator 0 window enable

10.2.7 CMP_DATA Output Data Register

Address: 0x4001 0214

Reset value: 0x0

Table 10-7 CMP_DATA Comparator Output Data Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					CMP2_FLT_DATA	CMP1_FLT_DATA	CMP0_FLT_DATA						CMP2_RAW_DATA	CMP1_RAW_DATA	CMP0_RAW_DATA
					R	R	R						R	R	R
					0	0	0						0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	CMP2_FLT_DATA	Comparator 2 filtered signal
[9]	CMP1_FLT_DATA	Comparator 1 filtered signal
[8]	CMP0_FLT_DATA	Comparator 0 filtered signal
[7:3]		Not used
[2]	CMP2_RAW_DATA	Comparator 2 raw output signal, directly from Analog Comparator 2
[1]	CMP1_RAW_DATA	Comparator 1 raw output signal, directly from Analog Comparator 1
[0]	CMP0_RAW_DATA	Comparator 0 raw output signal, directly from analog Comparator 0

11 HALL

11.1 Overview

The chip supports three HALL signal inputs.

The processing of the input HALL sensor signal includes:

Filtering. Eliminate the effect of HALL signal glitches.

Capture. When the HALL input changes, record the current timer value and output an interrupt.

Overflow. When the HALL signal remains changed for a long time, resulting in the counter overflows, an interrupt is output.

11.2 Implementation description

11.2.1 Signal source

The HALL signal comes from GPIO, and the chip has two IOs that can be used as the source of each HALL signal. Users can choose to use one of the GPIO input signals as the HALL signal by setting the GPIO register.

Please see DATASHEET for detailed pin location.

11.2.2 Working clock

The working frequency of HALL module is adjustable. Users can select the 1/2/4/8 frequency division of the main clock as the operating frequency of the HALL module by setting the HALL_CFG.CLK_DIV register. Both filtering and counting work at this frequency.

11.2.3 Signal filtering

The filter module is mainly used to remove glitches on the HALL signal.

The filtering includes two stages of filters:

In the first stage, the "5 in 7" rule is used for filtering. That is, if five "1" is reached while filtering the seven consecutive sampling points, output as "1"; if reached or over five "0", output as "0"; otherwise, the output result would remain unchanged as the last filtering. Whether the first stage filter is enabled or not can be selected by configuring the HALL_CFG. FIL_75. The details are shown below:

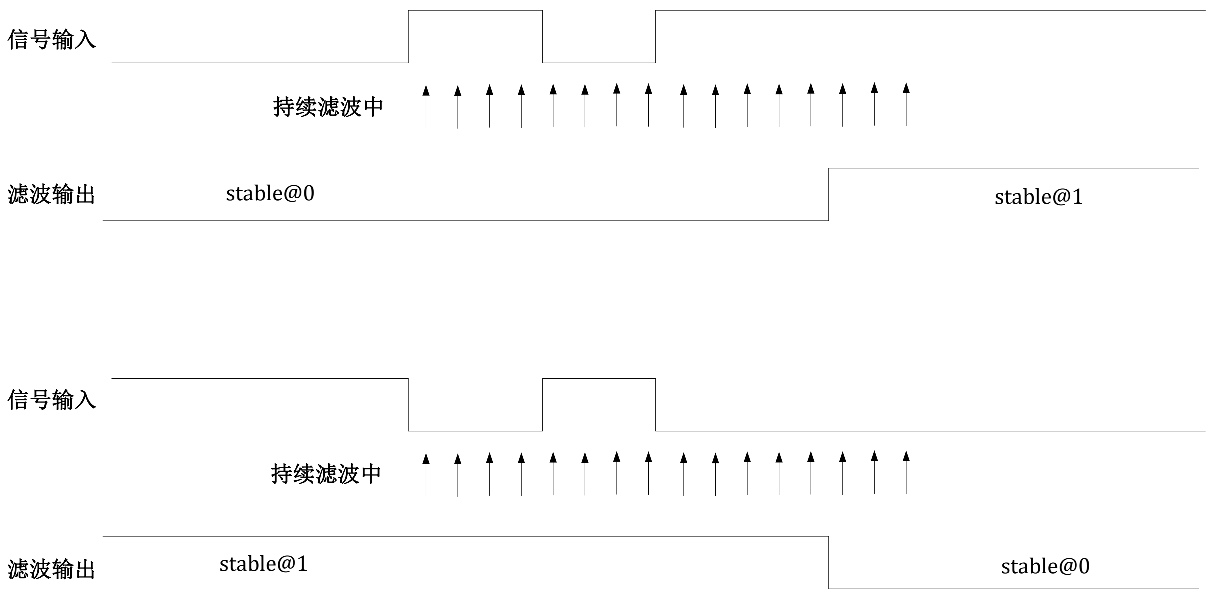


Figure 11-1 Block diagram of 7/5 filter module

In the second stage, continuous filtering is adopted. If all results filtered are zero while filtering N consecutive sampling points, output as "0"; if all results filtered are "1", output as "1"; otherwise, the output result would remain unchanged as the last filtering.

Select the filtering depth of the second-stage filter by setting HALL_CFG.FIL_LEN, that is, the number of consecutive samples. The maximum number of consecutive samples is 2^{15}, and the calculation formula of the filter time constant is as follows:

$$T_{fit} = T_{clk*}(\text{HALL\_CFG.FIL\_LEN}[14:0]+1)$$

For example, at a 96MHz operating frequency, the period T_{clk} is 10.4ns, the maximum register configuration is 32767, and the longest filter width is about 10.4ns × 32768≈340us.

Capture the filtered HALL signal by accessing HALL_INFO.FIL_DATA [2: 0]; HALL_INFO.RAW_DATA [2: 0] is the original HALL input signal before filtering, see 11.3.3 for details.

11.2.4 Capture

The capture module is used to measure the time between two HALL signal changes, with a 24-bit counter as its core. At 96 MHz, if the Hall clock is divided by 8 by the HALL_CFG. CLK_DIV = 3, it can record a maximum of about 2 ^ 24 * 8/96e6 = 1. The time width is 4 s, and the time resolution is 10.42 ns.

HALL_CNT starts counting from 0. When the HALL signal changes, the current HALL_CNT value will be saved to the HALL_WIDTH register, and the current HALL signal is saved to HALL_INFO.FIL_DATA. Then, a HALL signal change interrupt is output, and HALL_CNT starts counting from 0 again.

When the counter count value reaches HALL_TH, the HALL counter overflow interrupt is output, and the counter starts counting from 0 again.

11.2.5 Interrupt

Capture and overflow events trigger interrupts. The interrupt enable control bits are in HALL_CFG.CHG_IE and HALL_CFG.OV_IE, and the interrupt flag bits are in HALL_INFO.CHG_IF and HALL_INFO.OV_IF. The terminal flag can be cleared by writing 1 to HALL_INFO.CHG_IF and HALL_INFO.OV_IF.

The hall interrupt event is available as a DMA request, but requires that the CHG_RE, OV_RE, or SW_RE be enabled in the HALL_CFG. When the DMA receives the transfer request and starts the data transfer, the interrupt flag is cleared, and software clearing is no longer required.

11.2.6 Data flow

The data flow of the Hall block is shown in the figure below. FCLK is the Hall clock controlled by the SYS_CLK_FEN. HALL_CLK_EN gating signal, which is a PLL clock of 96 MHz after it is turned on.

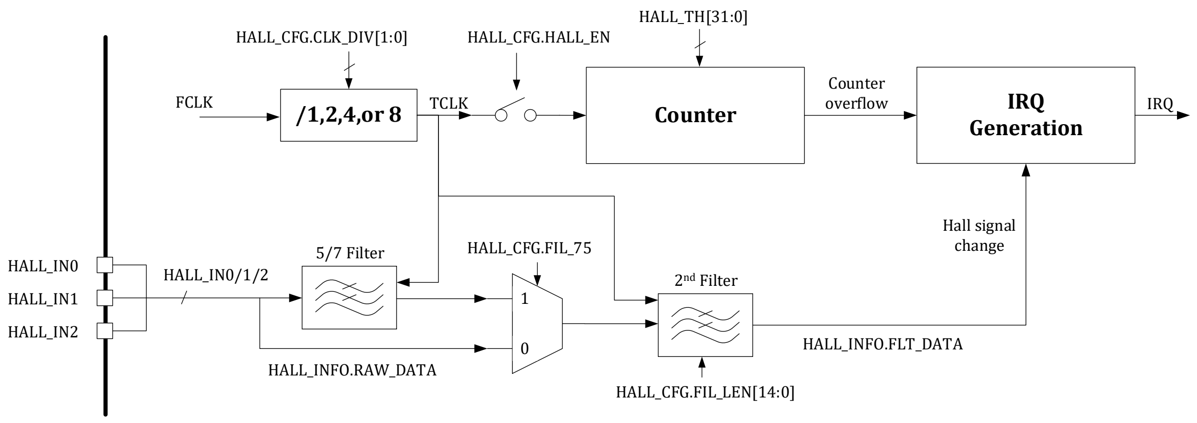


Figure 11-2Data flow block diagram

11.3 Register

11.3.1 Address assignment

The base address of the HALL module register is 0x4001_0300, and the register list is as follows:

Table 11-1 HALL module register address assignment

Name	Offset	Description
HALL_CFG	0x00	HALL module configuration register
HALL_INFO	0x04	HALL module information register
HALL_WIDTH	0x08	HALL Width Meter Value Register
HALL_TH	0x0C	HALL module counter threshold register
HALL_CNT	0x10	HALL count register

11.3.2 HALL_CFG HALL module configuration register

Address: 0x4001 0300

Reset value: 0x0

Table11-2 HALL_CFG HALL Module Configuration Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SW_IE	OV_IE	CHG_IE		OV_RE	CHG_RE	HALL_EN				FIL_75			CLK_DIV	
	RW	RW	RW		RW	RW	RW				RW			RW	
	0	0	0		0	0	0				0			0	

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

Location	Bit name	Description
[31]		Not used
[30]	SW_IE	Software-triggered HALL signal change interrupt enable. Active high. After this bit is valid, writing 1 to HALL_INFO.SW_IF, a HALL signal change interrupt will be generated manually and set the HALL_INFO. CHG_IF by writing a 1 to the HALL_INFO. SW_IF.
[29]	OV_IE	HALL counter overflow interrupt enable switch. Off by default. 1: Enable 0: Disable
[28]	CHG_IE	HALL signal change interrupt enable switch. Off by default. 1: Enable 0: Disable
[27]		Reserved
[26]	OV_RE	HALL counter overflow DMA request enable switch. Off by default. 1: Enable 0: Disable
[25]	CHG_RE	HALL signal change DMA request enable switch. Off by default. 1: Enable 0: Disable
[24]	HALL_EN	HALL block enable switch. Off by default. 1: Enable 0: Disable
[23:21]		Not used
[20]	FIL_75	7/5 filter switch (sequential sampling for seven times, and five results should be the same). Off by default. 1: Enable 0: Disable
[19:18]		Not used
[17:16]	CLK_DIV	HALL clock division factor 00: No frequency division 01: Two-divided frequency 10: Four-divided frequency 11: Eight-divided frequency
[15]		Unused
[14:0]	FIL_LEN	Filter width. Signals below the corresponding pulse width will be automatically filtered by the hardware. The calculation formula of the filter width is [14: 0]+1.

11.3.3 HALL_INFO HALL module information register

Address: 0x4001 0304

Reset value: 0x0

Table 11-3 HALL_INFO HALL Module Information Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													SW_IF	OV_IF	CHG_IF
													WO	RW1C	RW1C
													0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RAW_DATA			Reserved					FLT_DATA		
RW					RO			RW					RO		
0					0			0					0		

Location	Bit name	Description
[31:19]		Unused
[18]	SW_IF	Software-triggered HALL signal change interrupt. Trigger by writing 1, and clear automatically.
[17]	OV_IF	HALL counter overflow event flag. Write 1 to clear
[16]	CHG_IF	HALL signal change event flag. Write 1 to clear
[15:11]		Reserved bit. Write 0, and read 0
[10:8]	RAW_DATA	HALL value. Unfiltered result
[7:3]		Reserved bit. Write 0, and read 0
[2:0]	FLT_DATA	HALL value. Flter result

11.3.4 HALL_WIDTH HALL Width Meter Value Register

Address: 0x4001 0310

Reset value: 0x0

Table 11-4 HALL_WIDTH HALL width count register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CAP_CNT							
								RO							
								0							

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

Location	Bit name	Description
[31:24]		Not used
[23:0]	CAP_CNT	HALL Width Counter Value

11.3.5 HALL_TH HALL module counter threshold register

Address: 0x4001 030C

Reset value: 0x0

Table 11-5 HALL_TH HALL Module Counter Threshold Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								TH							
								RW							
								0							

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

Location	Bit name	Description
[31:24]		Not used
[23:0]	TH	HALL counter threshold

11.3.6 HALL_CNT HALL count register

Address: 0x4001 0308

Reset value: 0x0

Table 11-6 HALL_CNT HALL count register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								CNT							
								RW							
								0							

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

Location	Bit name	Description
[31:24]		Not used
[23:0]	CNT	HALL count value, write any value to clear

12 ADC

12.1 Overview

LKS32MC07x series chips integrate two 12-bit SAR-ADC conversion cores, and 14 analog IO input signals/4 OPA outputs/2 DAC outputs/temperature sensor outputs/AVDD can be used as the sampled signals of ADC. The main features are as follows:

- 2 ADC digital interfaces for 2 ADCs
- 3Msps sample rate, 48 MHz operation
- Each ADC interface supports 16 channel selection
- Each ADC interface has an analog watchdog that detects both upper and lower thresholds.
- Oversampling is supported
- Support software and hardware triggering function
- It can be linked with MCPWM/Timer/CL unit, and the trigger indication signal can be sent out for debugging through GPIO
- Customized sampling sequence is supported, and the sequence times and channel number can be flexibly configured
- Support left and right alignment modes
- Supports differential input analog signal
- Supports single-segment and two-segment sampling
- Hardware oversampling is supported, and the effective number of bits after oversampling can reach more than 16 bits.

The terminology about ADC conventions are:

Single-time sampling: complete the sampling conversion of the corresponding analog signal quantity to the data signal quantity of one channel, and store the digital quantity to the ADC_DATx register.

Single-stage sampling: may contain one or several samples. The analog channels for several samples can be the same or different. Sampling is usually triggered by MCPWM, UTimer, or software. One trigger signal could start one stage sampling. After sampling is completed, a corresponding stage sampling completion interrupt is generated.

12.1.1 Functional block diagram

Each ADC interface includes 16 data registers (the ADC samples the digital quantity corresponding to the analog quantity of each channel for 16 times), as well as several control registers, 14 regular trigger sampling registers, and 2 idle sampling trigger registers.

Two channels can be sampled at the same time.

Taking ADC0 as an example, the data register ADC0_DATx is used to store the digital quantity obtained by the xth sampling of ADC0. The source of the converted analog signal is selected by one of the 4 bits in the ADC 0_CHNx register (see12.2.3)。 Taking ADC 0_CHN0 as an example, Bits [3:0] select the analog signal source for the 0th sample, ADC01_CH4 to ADC01_CH9, (channels 4-9 shared by ADC0 and ADC1) ADC 0_CH10 to ADC0_CH13 are optional, if ADC0_CHN0[3:0] = 0, ADC0_CHN0[7:4] = 4, then the analog quantity positive end signal of the 0th sample is from OPA0_OUT, the analog quantity positive end signal of the 1st sample corresponds to IO ADC01_CH4, and so on.

The ADC_CHNT of the sampling channel number register controls the number of times of sampling in each segment, 1 ~ 15 corresponds to 1 ~ 15 times, and cannot be set to 0 times. The sum of sampling times of two sections shall not exceed 16 times.

The control logic selects the trigger signal from MCPWM or UTimer according to trigger configuration register ADC_TRIG to start sampling or initiates the sampling through the software trigger. MCPWM/UTimer will send timed trigger signal.

The ADC conversion complete interrupt is triggered when a segment conversion (all channels within a segment are sampled and converted) is complete.

Figure 12-1 ADC Acquisition Module Functional Block Diagram

Source selection allows the user to flexibly configure the sampling sequence and the source of the sampled signal, even for the purpose of multiple sampling of a single signal. The control register allows the user to configure the number of samples to increase the sampling frequency/reduce the sampling power.

Both the positive and negative signal sources can be configured, so that the user can flexibly use the differential signal source of each sampling and the IO resources.

Oversampling allows the user to average the signal through multiple samples to obtain a higher signal-to-noise ratio in scenarios where the sampling rate is not required to be high but the accuracy is required.

12.1.2 ADC trigger mode

- Support single-stage triggering and two-stage triggering
- Interval triggering can be achieved by setting the number of times the sampling trigger event occurs
- Can be triggered by software
- The completion of each sampling trigger generates a corresponding interrupt flag, and if the interrupt request is enabled, an interrupt service request is generated to the NVIC
- Trigger indication signal can be sent out through GPIO for debugging
- Support idle trigger sampling

There are two types of ADC triggers: Normal Trigger (NT) and Idle Trigger (IT). A regular trigger has a higher priority than an idle trigger. If an idle trigger occurs while the ADC is in the middle of a regular sample conversion, the ADC continues to complete the normally triggered channel sample conversion, and then completes the idle trigger sample conversion. If the ADC is in the middle of an idle sample conversion and a regular trigger occurs, the regular trigger interrupts the idle sample conversion and waits for the regular trigger sample to complete before the idle trigger occurs.

Caution: For chips whose SYS_AFE_INFO.VERSION<=3, idle trigger sampling and normal trigger sampling cannot be used simultaneously. For chips whose SYS_AFE_INFO.VERSION=4, idle trigger sampling and normal trigger sampling can be used simultaneously.

12.1.3 ADC Channel Select

Each ADC module has five channel signal source registers, which control the signal selection of sampling sequence 0 ~ 15. ADC_CHN0 controls the 0th to 3rd samplings, ADC_CHN1 controls the 4th to 7th samplings, ADC_CHN2 controls the 8th to 11th samplings, and ADC_CHN3 controls the 12th to 13th samplings. The ADC_ICHN controls the 0th to 1st idle samplings. Each sampling channel value selection range is 0 ~ 15, corresponding to channel 0 ~ 15, that is, a certain channel can be sampled multiple times. Each sample corresponds to a result register, and after the conversion, the ADC sample result can be obtained directly from the corresponding result register.

Table 12-1 Mapping ADC Channel Selection to Register Configuration

ADC sample sequence	Corresponding sample data register	Corresponding signal source register
Routine Sample 0	ADC_DAT0	ADC_CHN0[3:0]
1st routine sampling	ADC_DAT1	ADC_CHN0[7:4]
2nd routine sampling	ADC_DAT2	ADC_CHN0[11:8]
3rd routine sampling	ADC_DAT3	ADC_CHN0[15:12]
4th routine sampling	ADC_DAT4	ADC_CHN1[3:0]
	……	
13th routine sampling	ADC_DAT13	ADC_CHN3[7:4]
0th idle sampled	ADC_IDAT0	ADC_ICHN[3:0]
1st idle sampled	ADC_IDAT1	ADC_ICHN[7:4]

12.1.4 ADC Interrupt

The ADC interrupt signal is set high after each stage is sampled.

If a software or hardware trigger event occurs while the ADC is working, an abnormal trigger interrupt will be generated.

ADC_DAT0 has a threshold interrupt, and upper threshold/lower threshold settings are available. Set the threshold value by ADC_CFG [1], and an interrupt occurs when ADC_DAT0 exceeds ADC_DAT0_TH.

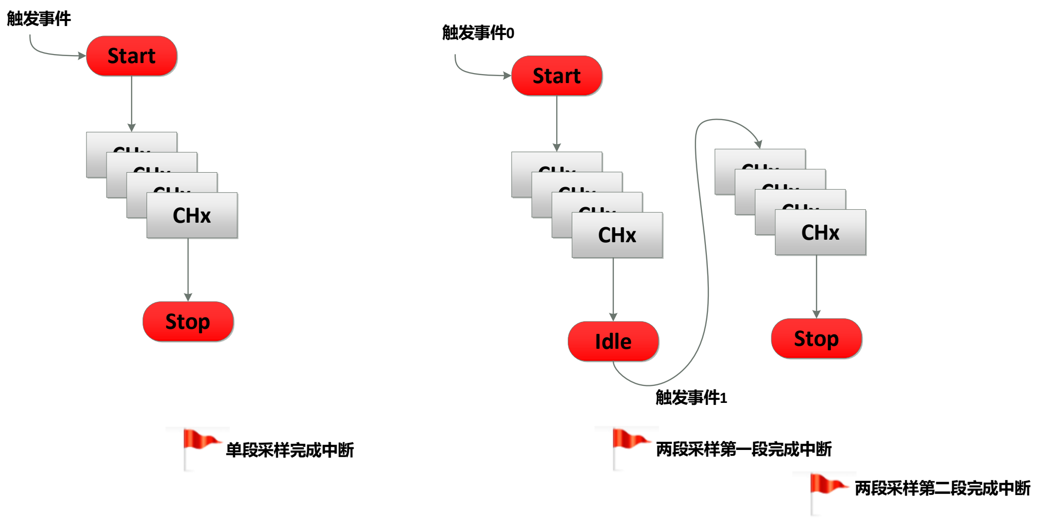


Figure 12-2 ADC interrupt generated

12.1.5 ADC output number system

The ADC output data is 12 bits complement and the input signal 0 corresponds to 12'b0000_0000_0000.

The ADC has two range settings, 3.6 V and 7.2 V. Using the 3.6 V range as an example, an input signal of -3.6 V corresponds to 12'b1000_0000_0000 and an input signal of + 3.6 V corresponds to 12'b011 1_1111_1111. The 12-bit complementary code after ADC conversion should be expanded to 16-bit and stored in the sampling data register of 16-bit width. Left or right alignment can be set by the configuration register. Taking 12'h1000_0000_1101 as an example, if the configuration is left-aligned, the right side is filled with four "0", and the value stored in ADCx_DAT is 16'h1000_0000_1101_0000; If the configuration is right-aligned, sign expansion is performed on the left, and the value stored in ADCx_DAT is 16'h1111_1000_0000_1101. Left alignment is recommended.

Because the chip is powered at 5 V, the 7.2 V range actually supports a differential signal range of ± 5 V. Therefore, the 7.2 V range cannot be used for the entire ADC range.

Table 12-2 ADC output digital quantity number-system conversion

Different input value/V at ADC 3.6 V range	Different input value/V at ADC 7.2 V range	Numeric value after conversion to signed number
3.6	7.2	12'b0111_1111_1111
0	0	12'b0000_0000_0000
-3.6	-7.2	12'b1000_0000_0000

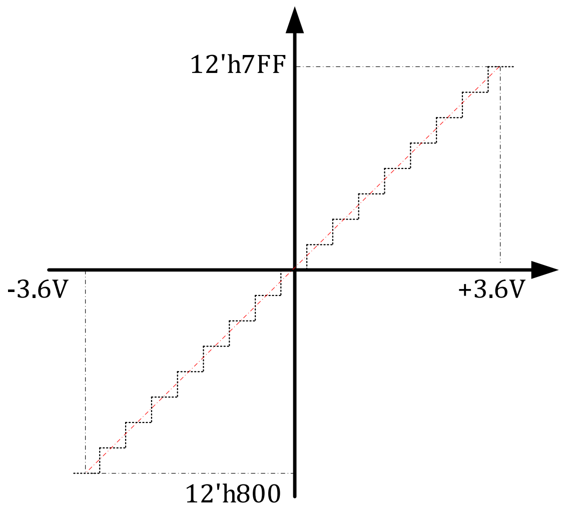


Figure 12-3 ADC Analog-to-Digital Conversion Scale at 2/3 Gain Settings

12.1.6 ADC range

The ADC has two range settings, 3.6 V and 7.2 V, with a default reference configuration of 2.4 V. This corresponds to a maximum input signal amplitude of ± 5 V at the 7.2 V range and ± 3.6 V at the 3.6 V range.

When the ADC sampling channel is set as the output signal of the OPA (i.e. OPA0 ~ OPA3), the appropriate OPA gain should be selected. This will allow the maximum signal in a specific application to be amplified to a level close to +/- 3.3V, while setting the ADC to a gain of 2/3. For example, the maximum phase current is 100A (effective value of sine wave), and the MOS internal resistance (assuming as MOS internal resistance sampling) is 5mR, then the maximum input signal amplitude of the OPA is +/- 707mV. Then, the OPA gain should be selected to be 4.5 times, and the amplified signal is about +/- 3.18V.

If the output signal of the OPA is amplified and the maximum signal is still less than +/- 2.4V due to objective reasons, the ADC should be configured for a 3.6 V scale.

When the ADC sampling channel is set as the input signal of the GPIO multiplex port, the ADC gain is also selected according to the maximum amplitude of the signal. Due to the limitation of the IO port, the signal range of the GPIO multiplex port can only be between -0.3V ~ AVDD+0.3V.

12.1.7 ADC correction

The ADC hardware interface module can perform DC offset calibration and gain calibration.

The $AMP_{correction}$, which is a gain calibration factor, stored by ADC_AMC is a 10-bit unsigned fixed-point number. ADC_AMC [9] is the integer, and ADC_AMC [8: 0] is the decimal, which can represent a fixed-point number whose value is around "1".

Record that the digital quantity output by the ADC is D_{ADC}, the true value corresponding to D_{ADC} is D, and D_0 is 0 in the coding system, then

$D = (D_{ADC}-D_0-DC_{offset})*AMP_{correction}$

Finally, the hardware will store the corrected D into the corresponding sampling data register. The ADC interface hardware circuit will automatically select $AMP_{correction}$ and DC_{offset} according to the gain configuration of each channel (ADCx_GAIN).

12.1.8 ADC Threshold Monitoring (Analog Watchdog)

Each ADC interface module is equipped with a set of threshold monitoring circuits for both upper and lower threshold monitoring. The data register of the ADC can be configured separately to enable or disable a set of threshold monitoring. A single data register can enable multiple sets of threshold monitoring at the same time, but this is not usually the case. If threshold monitoring is enabled, when the ADC completes a conversion and the corrected data is written to the ADCx_DAT register, a threshold comparison is performed and the threshold overrun interrupt flag is set if the data written is greater than the upper threshold or less than the lower threshold.

The threshold is a 12-bit signed number, so the threshold comparison is independent of the left-right alignment of the data. The ADCx_DATx [15:4] is compared to the threshold for left justification, and the ADCx_DATx [11:0] is compared to the threshold for right justification.

Note:

DC threshold monitoring logic treats the values of ADC DATA and ADC LTH and ADC HTH as unsigned numbers for threshold monitoring. Therefore, the usage scenarios of the threshold detection function are limited. The specific usage scenarios are as follows:

1. When the values of ADC LTH and ADC HTH converted to signed numbers are both positive, that is, 0<ADC LTH<ADC HTH, the threshold monitoring function can be used normally.

2. When the values of ADC LTH and ADC HTH converted into signed numbers are negative, that is, ADC LTH<ADC HTH<0, the threshold monitoring function can be used normally.

Except in the preceding scenarios, the threshold monitoring function is unavailable. For example, when the values of ADC HTH and ADC LTH are converted to a signed number, the symbol bit is inconsistent, and ADC HTH>ADC LTH in the case of a signed number, then the conversion to an unsigned number will result in ADC HTH<ADC LTH, and the threshold sampling interrupt will be continuously generated.

12.1.9 Oversampling

The ADC allows the signal to be oversampled and averaged before being written to the data register. The oversampling ratio can range from 1 to 128 times, but only in powers of 2, and is configured through the ADCx_CFG. OVSR. By default, one sample is written to the data register, that is, no oversampling is performed. If an oversampling multiple is configured, all signals specified by the ADCx_CHNx are averaged over multiple samples before being written to the data register. The value written to the data register is related to the ADC multiple conversion value by the following equation.

$$\text{ADCx\_DAT} = \frac{1}{OVSR}\sum_{i=0}^{OVSR-1}\text{ADCx\_DAT\_RAW}_i$$

Oversampling can be used in conjunction with threshold monitoring, where a threshold overrun event occurs only when the averaged value exceeds the threshold range.

When the oversampling mode is configured, the per-sample conversion time is multiplied and an interrupt is generated after the sample conversion is complete.

The ADCx_CFG.TROVS can be configured to sample and average data multiple times ADCx_CFG. OVSR in a single trigger, as shown in Figure 12-4. ADCx_CFG. TROVS = 0, ADCx_CFG. OVSR = 1, that is, the oversampling ratio is 2. After triggering, the ADC samples each signal twice and averages the results before writing the results to the corresponding ADCx_DAT register. If ADCx_CFG. TROVS = 1, it takes several ADCx_CFG. OVSR triggers to accumulate enough data to average, as shown in Figure 12-5, if the ADCx_CFG. OVSR = 1, that is, the oversampling ratio is still 2, the accumulated data is averaged and written to the data register only twice every two triggers. When multiple triggers are required for accumulation, only one channel can be sampled for each trigger, that is, the ADCx_CHNT is required to be 1. If multiple different signals are sampled, the data of different channels will be accumulated together to obtain incorrect converted data when the channels are switched.

Oversampling can be used in conjunction with continuous sampling, where the ADC samples the signal repeatedly, accumulates the number of oversamplings, and stores the data in the data register.

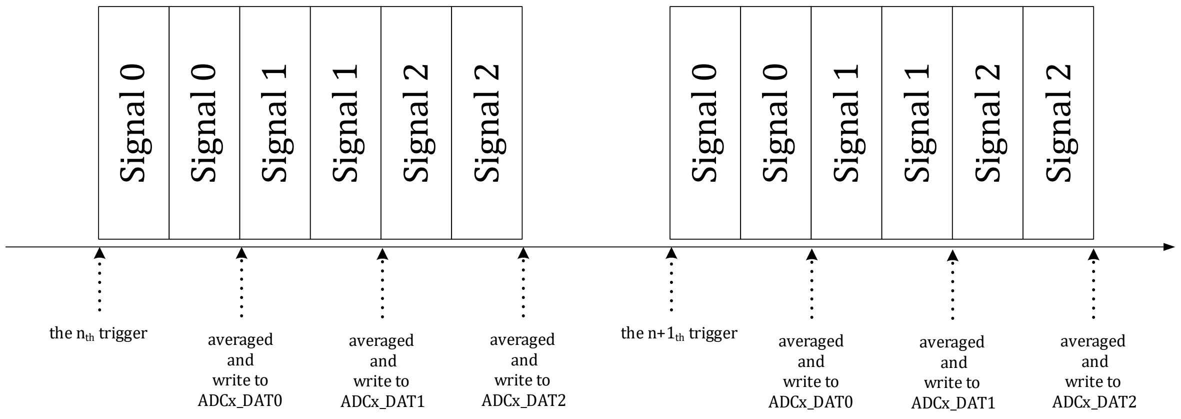

Signal 0
Signal 0
Signal 1
Signal 1
Signal 2
Signal 2
Signal 0
Signal 0
Signal 1
Signal 1
Signal 2
Signal 2
the nth trigger
averaged and write to ADCx_DAT0
averaged and write to ADCx_DAT1
averaged and write to ADCx_DAT2
the n+1th trigger
averaged and write to ADCx_DAT0
averaged and write to ADCx_DAT1
averaged and write to ADCx_DAT2

Figure12-4 Oversampling Conversion Timing Example 1

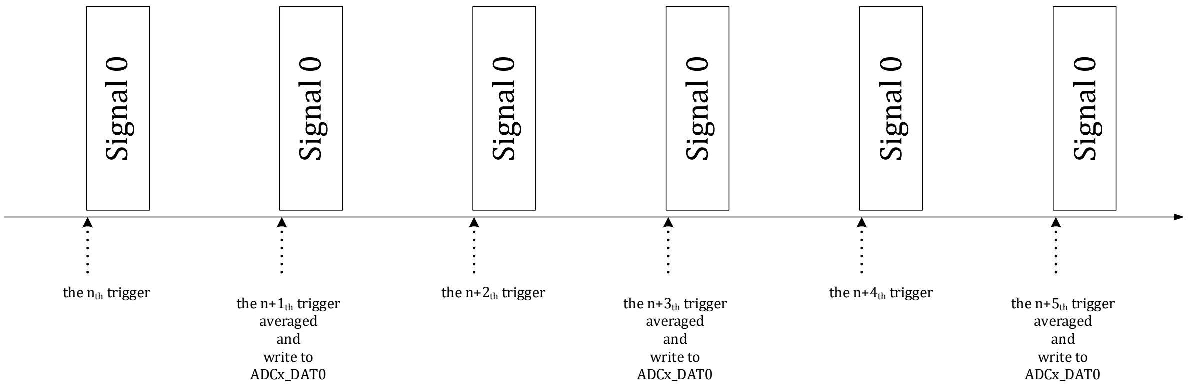

Signal 0
Signal 0
Signal 0
Signal 0
Signal 0
Signal 0
the nth trigger
the n+1th trigger averaged and write to ADCx_DAT0
the n+2th trigger
the n+3th trigger averaged and write to ADCx_DAT0
the n+4th trigger
the n+5th trigger averaged and write to ADCx_DAT0

Figure12-5 Oversampling Conversion Timing Example 2

Oversampling is supported only for regular sampling, and the occurrence of idle sampling during regular sampling does not affect the accumulation of oversampled data.

12.2 Register

12.2.1 Address assignment

The base address of ADC0 in the chip is 0x4001_0400;

The base address of ADC1 in the chip is 0x4001_0500.

Table 12-3 ADCx register list

Name	Offset address	Description
ADCx_DAT0	0x00	ADCx 0th Regular Sample Data
ADCx_DAT1	0x04	ADCx 1st Regular Sample Data
ADCx_DAT2	0x08	ADCx 2nd Regular Sample Data
ADCx_DAT3	0x0C	ADCx 3rd Regular Sample Data
ADCx_DAT4	0x10	ADCx 4th Regular Sample Data
ADCx_DAT5	0x14	ADCx 5th Regular Sample Data
ADCx_DAT6	0x18	ADCx 6th Regular Sample Data
ADCx_DAT7	0x1C	ADCx 7th Regular Sample Data
ADCx_DAT8	0x20	ADCx 8th Regular Sample Data
ADCx_DAT9	0x24	ADCx 9th Regular Sample Data
ADCx_DAT10	0x28	ADCx 10th Regular Sample Data
ADCx_DAT11	0x2C	ADCx 11th Regular Sample Data
ADCx_DAT12	0x30	ADCx 12th Regular Sample Data
ADCx_DAT13	0x34	ADCx 13th Regular Sample Data
ADCx_IDAT0	0x38	ADCx 0th Idle Sample Data
ADCx_IDAT1	0x3C	ADCx 1st Idle Sample Data
ADCx_ICHN	0x40	ADCx Idle Sample Channel Configuration
ADCx_CHN0	0x50	Input signal selection for the 0th to 3rd normal sampling of ADCx
ADCx_CHN1	0x54	ADCx 4th-7th normal sampling input signal selection
ADCx_CHN2	0x58	ADCx 8th to 11th normal sample input signal selection
ADCx_CHN3	0x5C	ADCx 12th to 15th Regular Sample Input Signal Selection
ADCx_CHNT	0x60	ADCx Number of Samples in Various Trigger Mode
ADCx_GAIN	0x64	ADCx Gain Select
ADCx_CFG	0x74	ADCx mode configuration
ADCx_TRIG	0x78	ADCx Sampling Trigger Configuration
ADCx_SWT	0x7C	ADCx software trigger
ADCx_DC0	0x80	DC offset when ADCx gain is 0
ADCx_AMC0	0x84	Gain correction factor when ADCx gain is 0
ADCx_DC1	0x88	DC offset when ADCx gain is 1
ADCx_AMC1	0x8C	Gain correction factor when ADCx gain is 1
ADCx_IE	0x90	ADCx Interrupt Enable
ADCx_IF	0x94	ADCx interrupt flag
ADCx_LTH	0xC4	ADCx Data Low Threshold
ADCx_HTH	0xC8	ADCx Data High Threshold
ADCx_GEN	0xCC	ADCx threshold monitoring enabled

12.2.2 Sample Data Register

12.2.2.1 ADCx_DAT0(x = 0,1)

Addresses are: 0x4001_0400, 0x4001_0500

Reset value: 0x0

Table 12-4 Sample Data Register ADCx_DAT0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT0															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT0	ADCx 0th sample data

12.2.2.2 ADCx_DAT1(x = 0,1)

Addresses are: 0x4001_0404, 0x4001_0504

Reset value: 0x0

Table 12-5 Sample Data Register ADCx_DAT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT1															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT1	ADCx 1st sample data

12.2.2.3 ADCx_DAT2(x = 0,1)

Addresses are: 0x4001_0408, 0x4001_0508

Reset value: 0x0

Table 12-6 Sample Data Register ADCx_DAT2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT2															

RW
0

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT2	ADCx 2nd Sample Data

12.2.2.4 ADCx_DAT3(x = 0,1)

Addresses are: 0x4001_040 C, 0x4001_050C

Reset value: 0x0

Table 12-7 Sample Data Register ADCx_DAT3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT3															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT3	ADCx 3rd sample data

12.2.2.5 ADCx_DAT4(x = 0,1)

Addresses are: 0x4001_0410, 0x4001_0510

Reset value: 0x0

Table 12-8 Sample Data Register ADCx_DAT4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT4															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT4	ADCx 4th sample data

12.2.2.6 ADCx_DAT5(x = 0,1)

Addresses are: 0x4001_0414, 0x4001_0514

Reset value: 0x0

Table 12-9 Sample Data Register ADCx_DAT5

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT5															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT5	ADCx 5th Sample Data

12.2.2.7 ADCx_DAT6(x = 0,1)

Address: 0x4001_0418, 0x4001_0518

Reset value: 0x0

Table 12-10 Sample Data Register ADCx_DAT6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT6															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT6	ADCx 6th Sample Data

12.2.2.8 ADCx_DAT7(x = 0,1)

Addresses are: 0x4001_041C, 0x4001_051C

Reset value: 0x0

Table 12-11 Sample Data Register ADCx_DAT7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT7															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT7	ADCx 7th sample data

12.2.2.9 ADCx_DAT8(x = 0,1)

Address: 0x4001_0420, 0x4001_0520

Reset value: 0x0

Table 12-12 Sample Data Register ADCx_DAT8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT8															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT8	ADCx 8th sample data

12.2.2.10 ADCx_DAT9(x = 0,1)

Addresses are: 0x4001_0424, 0x4001_0524

Reset value: 0x0

Table 12-13 Sample Data Register ADCx_DAT9

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT9															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT9	ADCx 9th sample data

12.2.2.11 ADCx_DAT10(x = 0,1)

Addresses are: 0x4001_0428, 0x4001_0528

Reset value: 0x0

Table 12-14 Sample Data Register ADCx_DAT10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT10															
RW															
0															

Location	Bit name	Description

[31:16]		Not used
[15:0]	DAT10	ADCx 10th sample data

12.2.2.12 ADCx_DAT11(x = 0,1)

Addresses are: 0x4001_042C, 0x4001_052C

Reset value: 0x0

Table 12-15 Sample Data Register ADCx_DAT11

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT11															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT11	ADCx 11th Sample Data

12.2.2.13 ADCx_DAT12(x = 0,1)

Addresses are: 0x4001_0430, 0x4001_0530

Reset value: 0x0

Table 12-16 Sample Data Register ADCx_DAT12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT12															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT12	ADCx 12th Sample Data

12.2.2.14 ADCx_DAT13(x = 0,1)

Addresses are: 0x4001_0434, 0x4001_0534

Reset value: 0x0

Table 12-17 Sample Data Register ADCx_DAT13

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAT13															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	DAT13	ADCx 13th Sample Data

12.2.2.15 ADCx_IDAT0(x = 0,1)

Address: 0x4001_0438, 0x4001_0538

Reset value: 0x0

Table 12-18 Idle Sample Data Register ADCx_IDAT0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDAT0															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	IDAT0	ADCx 0th Idle Sample Data

12.2.2.16 ADCx_IDAT1(x = 0,1)

Addresses are: 0x4001_043C, 0x4001_053C

Reset value: 0x0

Table 12-19 Idle Sample Data Register ADCx_IDAT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDAT1															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	IDAT15	ADCx 1st Idle Sample Data

12.2.3 Signal source register

12.2.3.1 ADCx_CHN0(x = 0,1)

Addresses are: 0x4001_0450, 0x4001_0550

Reset value: 0x0

Table 12-20 Signal source register ADCx_CHN0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS3				PDS2				PDS1				PDS0			

RW	RW	RW	RW
0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15:12]	PDS3	ADCx 3rd Sample Positive Side Input Signal Select
[11:8]	PDS2	ADCx 2nd Sample Positive Side Input Signal Select
[7:4]	PDS1	ADCx 1st Sample Positive Side Input Signal Select
[3:0]	PDS0	ADCx 0-th Sample Positive Side Input Signal Select

12.2.3.2 ADCx_CHN1(x = 0,1)

Addresses are: 0x4001_0454, 0x4001_0554

Reset value: 0x0

Table 12-21 Signal source register ADCx_CHN1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS7				PDS6				PDS5				PDS4			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Description
[31:16]		Not used
[15:12]	PDS7	ADCx 7th Sample Positive Side Input Signal Select
[11:8]	PDS6	ADCx 6th Sample Positive Side Input Signal Select
[7:4]	PDS5	ADCx 5th Sample Positive Side Input Signal Select
[3:0]	PDS4	ADCx 4th Sample Positive Side Input Signal Select

12.2.3.3 ADCx_CHN2(x = 0,1)

Address: 0x4001_0458, 0x4001_0558

Reset value: 0x0

Table 12-22 Signal source register ADCx_CHN2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDS11				PDS10				PDS9				PDS8			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Description
[31:16]		Not used
[15:12]	PDS11	ADCx 11th Sample Positive Side Input Signal Select
[11:8]	PDS10	ADCx 10th Sample Positive Side Input Signal Select
[7:4]	PDS9	ADCx 9th Sample Positive Side Input Signal Select

[3:0]	PDS8	ADCx 8th Sample Positive Side Input Signal Select

12.2.3.4 ADCx_CHN3(x = 0,1)

Addresses are: 0x4001_045C, 0x4001_055C

Reset value: 0x0

Table 12-23 Signal source register ADCx_CHN3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								PDS13				PDS12			
								RW				RW			
								0				0			

Location	Bit name	Description
[31:8]		Not used
[7:4]	PDS13	ADCx 13th Sample Positive Side Input Signal Select
[3:0]	PDS12	ADCx 12th Sample Positive Side Input Signal Select

12.2.3.5 ADCx_ICHN(x = 0,1)

Addresses are: 0x4001_0440, 0x4001_0540

Reset value: 0x0

Table 12-24 Signal source register ADCx_ICHN.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IDS1				IDS0			
								RW				RW			
								0				0			

Location	Bit name	Description
[31:8]		Not used
[7:4]	IDS1	ADCx 1st Idle Sample Input Signal Select
[3:0]	IDS0	ADCx 0th Idle Sample Input Signal Select

12.2.3.6 ADC channel signal selection

The acquired analog signals that can be selected by the two ADC interfaces are not exactly the same, as shown in the following Table 12-25.

Table12-25 ADC Analog Signal Source Distribution

Register	Channel signal corresponding to different configuration values
ADC0_CHNx	0x0: OPA0 output 0x1: OPA1 output 0x2: OPA2 output 0x3: OPA3 output 0x4: ADC01_CH4

	0x5: ADC01_CH5 0x6: ADC01_CH6 0x7: ADC01_CH7 0x8: ADC01_CH8 0x9: ADC01_CH9 0xA: ADC0_CH10 0xB: ADC0_CH11 0xC: ADC0_CH12 0xD: ADC0_CH13 0xE: DAC0_OUT 0xF: DAC1_OUT
ADC1_CHNx	0x0: OPA0 output 0x1: OPA1 output 0x2: OPA2 output 0x3: OPA3 output 0x4: ADC01_CH4 0x5: ADC01_CH5 0x6: ADC01_CH6 0x7: ADC01_CH7 0x8: ADC01_CH8 0x9: ADC01_CH9 0xA: ADC1_CH10 0xB: ADC1_CH11 0xC: ADC1_CH12 0xD: ADC1_CH13 0xE: Temperature sensor 0xF: Internal power supply AVDD

* ADC 01_CH4 ~ ADC 01_CH9 are common channels for ADC0 and ADC1

Normally, ADC samples the op-amp channel with differential signals.

12.2.4 Number of sampling channels register

12.2.4.1 ADCx_CHNT(x = 0,1)

Addresses are: 0x4001_0460, 0x4001_0560

Reset value: 0x0

Table 12-26 Sample Channel Count Register ADCx_CHNT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						IS1		S2				S1			
						RW		RW				RW			
						0		0				0			

Location	Bit	Description

	name	
[31:10]		Not used
[9:8]	IS1	Number of idle samples
[7:4]	S2	Regular sampling times of the second section
[3:0]	S1	Regular sampling times of the first section

Note: Sampling times are not allowed to be configured as 0. 1 means 1 channel, 2 means 2 channels, 16 means 16 channels. If regular sampling uses two-segment sampling, the sum of the channels for the two segments should not exceed 16.

Idle sampling supports only single-segment sampling, and the sampling times can be configured as 1 or 2.

12.2.5 Configuration register

12.2.5.1 ADCx_GAIN(x = 0,1)

Addresses are: 0x4001_0464, 0x4001_0564

Reset value: 0x0

Table 12-27 Gain select register ADCx_GAIN.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IG1	IG0	G13	G12	G11	G10	G9	G8	G7	G6	G5	G4	G3	G2	G1	G0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]	IG1	IDAT1 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[14]	IG0	IDAT0 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[13]	G13	DAT13 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[12]	G12	DAT12 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[11]	G11	DAT11 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[10]	G10	DAT10 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[9]	G9	DAT9 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[8]	G8	DAT8 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[7]	G7	DAT7 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[6]	G6	DAT6 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[5]	G5	DAT5 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[4]	G4	DAT4 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[3]	G3	DAT3 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[2]	G2	DAT2 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[1]	G1	DAT1 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range
[0]	G0	DAT0 gain select, 0: ± 3.6 V range, 1: ± 7.2 V range

Limited to a 5 V supply, the 7.2 V range can only be used up to ± 5 V.

12.2.5.2 ADCx_CFG (x = 0,1)

Address: 0x4001_0474, 0x4001_0574

Reset value: 0x0

Table 12-28 Mode Configuration Register ADCx_CFG.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			NSMP	FSM_RS	DATA_ALIGN		CSMP	TCNT				TROVS	OVSR		
			RW	RW	RW		RW	RW				RW	RW		
			0	0	0		0	0				0	0		

Location	Bit name	Description
[31:13]		Not used
[12]	NSMP	0: Single-segment sampling, 1: Two-segment sampling
[11]	FSM_RS	State machine reset. After the software is written 1, the state machine returns to the idle state, and is automatically cleared upon completion. This reset control does not affect the configuration values of the other ADC registers. Reading this bit returns the current state of the ADC, with a 1 indicating that the ADC is currently sampling and converting, and a 0 indicating idle.
[10]	DATA_ALIGN	ADC0_DAT alignment 0: Left aligned, with 4'h0 at the right, 1: right-aligned, with 4bit sign bit on the left
[9]		Not used
[8]	CSMP	Continuous sampling mode 0: Disable 1: Enable the continuous sampling mode. When the trigger arrives, the ADC starts sampling conversion. After the conversion of all signals is completed, a new round of sampling conversion starts from the 0th signal without waiting for the trigger.
[7:4]	TCNT	The number of events required to trigger a sample. 0: Indicates that 1 event is required to trigger a sampling 1: Indicates that 2 events are required to trigger a sampling …… 15: Indicates that 16 events are required to trigger a sampling
[3]	TROVS	Oversampling trigger model 0: One trigger means continuous oversampling to OVSR for several times, and the data is averaged and stored. Multiple channels can be configured for sampling, and each channel samples OVSR for times before sampling the next channel. 1: Only one sample conversion is performed in one trigger, and it takes OVSR triggers to collect and average enough data before it is stored in the data register. Due to the limitation of the number of accumulators, only one channel can be configured for sampling ADCx_CHNT in this configuration.

[2:0]	OVSR	Oversampling rate 0:1, the default is to store the data after one sampling 1: Store data after 2 times of sampling 2: Store data after 4 times of sampling 3: Store data after 8 times of sampling 4: Store data after 16 times of sampling 5: Store data after 32 times of sampling 6: Store data after 64 times of sampling 7: Store data after 128 times of sampling When the oversampling rate is set to a non-zero value, it must be ensured that ADC0 and ADC1 do not operate simultaneously. That is, during the sampling and conversion process of one ADC, the sampling trigger event of the other ADC must not occur.

12.2.5.3 ADC0_TRIG

Address: 0x4001_0478

Reset value: 0x0

Table 12-29 Sample trigger configuration register ADC0_TRIG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CL_IT3_EN	CL_IT2_EN	CL_IT1_EN	CL_IT0_EN	CL_NT3_EN	CL_NT2_EN	CL_NT1_EN	CL_NT0_EN	UTIME2_CMP1_EN	UTIME2_CMP0_EN	UTIME0_CMP1_EN	UTIME0_CMP0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]	CL_IT3_EN	CL_OUTPUT [3] triggers ADC idle sampling enable, active high
[14]	CL_IT2_EN	CL_OUTPUT [2] triggers ADC idle sampling enable, active high
[13]	CL_IT1_EN	CL_OUTPUT [1] triggers ADC idle sampling enable, active high
[12]	CL_IT0_EN	CL_OUTPUT [0] triggers ADC idle sampling enable, active high
[11]	CL_NT3_EN	CL_OUTPUT [3] Triggers ADC Normal Sampling Enable, Active High
[10]	CL_NT2_EN	CL_OUTPUT [2] triggers ADC normal sampling enable, active high
[9]	CL_NT1_EN	CL_OUTPUT [1] triggers ADC normal sampling enable, active high
[8]	CL_NT0_EN	CL_OUTPUT [0] triggers ADC normal sampling enable, active high
[7]	TIMER2_CMP1_EN	TIMER2 Comparison Event 1 Triggers ADC Normal Sampling Enable, Active High
[6]	TIMER2_CMP0_EN	TIMER2 Comparison Event 0 Triggers ADC Normal

		Sampling Enable, Active High
[5]	TIMER0_CMP1_EN	TIMER0 Comparison Event 1 Triggers ADC Normal Sampling Enable, Active High
[4]	TIMER0_CMP0_EN	TIMER0 Comparison Event 0 Triggers ADC Normal Sampling Enable, Active High
[3]	MCPWM0_T3_EN	MCPWM0 T3 Event Triggers ADC Normal Sampling Enable, Active High
[2]	MCPWM0_T2_EN	MCPWM0 T2 Event Triggers ADC Normal Sampling Enable, Active High
[1]	MCPWM0_T1_EN	MCPWM0 T1 Event Triggers ADC Normal Sampling Enable, Active High
[0]	MCPWM0_T0_EN	MCPWM0 T0 Event Triggers ADC Normal Sampling Enable, Active High

12.2.5.4 ADC1_TRIG

The addresses are: 0x4001 0578

Reset value: 0x0

Table 12-30 Sample trigger configuration register ADC1_TRIG.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CL_IT3_EN	CL_IT2_EN	CL_IT1_EN	CL_IT0_EN	CL_NT3_EN	CL_NT2_EN	CL_NT1_EN	CL_NT0_EN	UTIME3_CMP1_EN	UTIME3_CMP0_EN	UTIME1_CMP1_EN	UTIME1_CMP0_EN	MCPWM0_T3_EN	MCPWM0_T2_EN	MCPWM0_T1_EN	MCPWM0_T0_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]	CL_IT3_EN	CL_OUTPUT [3] triggers ADC idle sampling enable, active high
[14]	CL_IT2_EN	CL_OUTPUT [2] triggers ADC idle sampling enable, active high
[13]	CL_IT1_EN	CL_OUTPUT [1] triggers ADC idle sampling enable, active high
[12]	CL_IT0_EN	CL_OUTPUT [0] triggers ADC idle sampling enable, active high
[11]	CL_NT3_EN	CL_OUTPUT [3] Triggers ADC Normal Sampling Enable, Active High
[10]	CL_NT2_EN	CL_OUTPUT [2] triggers ADC normal sampling enable, active high
[9]	CL_NT1_EN	CL_OUTPUT [1] triggers ADC normal sampling enable, active high
[8]	CL_NT0_EN	CL_OUTPUT [0] triggers ADC normal sampling enable, active high
[7]	TIMER3_CMP1_EN	TIMER3 Comparison Event 1 Triggers ADC Normal Sampling Enable, Active High
[6]	TIMER3_CMP0_EN	TIMER3 Comparison Event 0 Triggers ADC Normal Sampling Enable, Active High

[5]	TIMER1_CMP1_EN	TIMER1 Comparison Event 1 Triggers ADC Normal Sampling Enable, Active High
[4]	TIMER1_CMP0_EN	TIMER1 Comparison Event 0 Triggers ADC Normal Sampling Enable, Active High
[3]	MCPWM0_T3_EN	MCPWM0 T3 Event Triggers ADC Normal Sampling Enable, Active High
[2]	MCPWM0_T2_EN	MCPWM0 T2 Event Triggers ADC Normal Sampling Enable, Active High
[1]	MCPWM0_T1_EN	MCPWM0 T1 Event Triggers ADC Normal Sampling Enable, Active High
[0]	MCPWM0_T0_EN	MCPWM0 T0 Event Triggers ADC Normal Sampling Enable, Active High

The trigger source of ADC is MCPWM or TIMER. Usually, and MCPWM and TIMER are not used to trigger at the same time, but only one of them is selected.

The trigger signal of TIMER to ADC can be observed by configuring GPIO as Timer function (7th/8th function) and outputting Timer channel signal.

The trigger signal of MCPWM to ADC can be sent by setting GPIO as the ninth function, that is, ADC_TRIGGER function, which is used to capture and debug. Every time an ADC trigger occurs, the ADC_TRIGGER signal flips over for capture output.

When using two-segment trigger, only the trigger signal of BIT0/2/4/6 ... /16/18/... can be used as the trigger signal of the first segment, only the trigger signal of BIT1/3/5/... /17/19/...can be used as the trigger signal of the second segment.

For example, when both the MCPWM0_T1_EN and the MCPWM0_T0_EN are enabled, the first segment sampling is triggered by the MCPW_M0_T0_EN, but the second segment sampling is not triggered. MCPWM0_T1_EN triggers the second segment but not the first segment.

12.2.6 Software trigger register

12.2.6.1 ADCx_SWT(x = 0,1)

Addresses are: 0x4001_047 C, 0x4001_057 C

Reset value: 0x0

Table 12-31 Software trigger register ADCx_SWT.

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	SWT	When the write data is 0x5AA5, a general trigger is generated When the write data is 0xF00F, an idle trigger is generated

Note that the software trigger acquisition register is a write-only register, and the software trigger event is generated only when the write data is 0x5AA5/0xF00F. A bus write generates a software trigger, and the register is automatically cleared after a data write generates a software trigger. The next software trigger needs to writ 0x5AA5/0xF00F again.

12.2.7 Calibration register

One of the signal sources in the optional analog channel of the ADC is GND, and the DC bias of the system can be obtained by measuring this channel.

Generally, after the system is initialized, the GND signal will be measured once, and the measured value will be stored in the DC offset register. The DC offset will be automatically subtracted from the sampling value of the signals of other channels each time, and then stored in the converted digital quantity register.

Due to signal errors, the signal with the DC offset removed may overflow, and the overflowing data is saturated so that the signal range is between -1.2 V and + 1.2 V or between -2.4 V and + 2.4 V.

The ADC is available in two ranges: 3.6 V and 7.2 V. The 3.6 V range corresponds to a maximum input signal amplitude of ± 3.6 V. The 7.2 V range corresponds to a maximum input signal amplitude of ± 5 V. The two ranges use different DC offset and gain correction factors. Therefore, there are two sets of correction registers, ADCx_DC0, ADCx_AMC0 and ADCxDC1, ADCxAMC1. The chip has been calibrated in the factory, and the calibration data is stored in Flash info. When the chip is powered on, the loading of calibration parameters is automatically completed. When the ADC module is initialized, the DC offset needs to be configured according to the data left-right alignment mode. Please refer to the library function provided by the chip supplier. When the ADC module is soft reset using the SYS_SFT_RST, the registers inside the ADC are reset. Calibration parameters such as DC/AMC need to be read from the NVR using the ADC initialization function again.

12.2.7.1 ADCx_DC0(x = 0,1)

Addresses are: 0x4001_0480, 0x4001_0580

Reset value: 0x0

Table 12-32 0 gain DC offset register ADCx_DC0

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	DC_OFFSET	ADC sampling circuit DC offset

Considering that the DC offset of the ADC is close to 0, only the lower 10 bits of the register are meaningful here, and the upper 6 bits are only the sign extension of ADCx_DC[9], and the ADCx_DC register is also sign extended when it is involved in ADC data correction.

That is, if 0x12B0 is written to the ADCx_DC; What is actually written is 0x2B0, and when read out, 0xFFB0 is read out.

The ADC_DC value stored here should correspond to the offset value that is right-justified. When the ADCx_CFG register is configured to be left-justified, the hardware will automatically adjust the ADCx_DC to participate in ADC calibration according to the alignment setting. Specifically, when right-aligned, ADCx_DC[15:0] is directly involved in the calibration operation, and when left-aligned, ADCx_DC[15:0] is shifted left by 4 bits and then involved in the ADC calibration operation.

12.2.7.2 ADCx_AMC0(x = 0,1)

The addresses are: 0x4001_0484, 0x4001_0584

Reset value: 0x200

Table 12-33 0 Gain Gain Correction Register ADCx_AMC0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AM_CALI									
						RW									
						0x200									

Location	Bit name	Description
[31:10]		Not used
[9:0]	AM_CALI	ADC sample circuit gain correction coefficient

The gain correction factor of the ADC is a value close to 1. 0×200 is calibrated as 1. For example, 0×1F0 is less than 1, and 0×205 is greater than 1.

12.2.7.3 ADCx_DC1(x = 0,1)

Address: 0x4001_0488, 0x4001_0588

Reset value: 0x0

Table 12-34 1st gain DC offset register ADCx_DC1

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	DC_OFFSET	ADC sampling circuit DC offset

12.2.7.4 ADCx_AMC1(x = 0,1)

The addresses are: 0x4001_048C, 0x4001_058C

Reset value: 0x200

Table 12-35 1st gain gain correction register ADCx_AMC1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						AM_CALI									
						RW									
						0x200									

Location	Bit name	Description
[31:10]		Not used
[9:0]	AM_CALI	ADC sample circuit gain correction coefficient

12.2.8 Interrupt register

12.2.8.1 ADCx_IE(x = 0,1)

Address: 0x4001_0490, 0x4001_0590

Reset value: 0x0

Table 12-36 Interrupt Enable Register ADCx_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF_RE	HERR_RE	SERR_RE			AWD0_RE	SF2_RE	SF1_RE	ISF_IE	HERR_IE	SERR_IE			AWD0_IE	SF2_IE	SF1_IE
RW	RW	RW			RW	RW	RW	RW	RW	RW			RW	RW	RW
0	0	0			0	0	0	0	0	0			0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]	ISF_RE	Idle sample complete DMA request enabled
[14]	HERR_RE	A hardware trigger occurs in a non-idle state DMA request enable
[13]	SERR_RE	Software trigger occurs in non-idle state DMA request enable
[12:11]		Not used
[10]	AWD0_RE	Threshold monitoring 0 out of limit DMA request enable
[9]	SF2_RE	The second segment of sampling is completed DMA request enable

[8]	SF1_RE	The first segment of sampling is completed DMA request enable
[7]	ISF_IE	Idle sampling complete interrupt enabled
[6]	HERR_IE	Hardware trigger occurs in non-idle state interrupt enable
[5]	SERR_IE	Software trigger occurs in non-idle state interrupt enable
[4:3]		Not used
[2]	AWD0_IE	Threshold monitoring 0 overrun interrupt enable
[1]	SF2_IE	Second-segment normal sample complete interrupt enable
[0]	SF1_IE	First segment normal sample complete interrupt enable

12.2.8.2 ADCx_IF(x = 0,1)

The addresses are: 0x4001_0494, 0x4001_0594.

Reset value: 0x0

Table12-37Interrupt Flags Register ADCx_IF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								ISF_IF	HERR_IF	SERR_IF			AWD0_IF	SF2_IF	SF1_IF
								RW1C	RW1C	RW1C			RW1C	RW1C	RW1C
								0	0	0			0	0	0

Location	Bit name	Description
[31:8]		Not used
[7]	ISF_IF	Interrupt flag for idle sampling complete
[6]	HERR_IF	Interrupt flag for hardware trigger occurs in non-idle state
[5]	SERR_IF	Interrupt flag for software trigger occurs in non-idle state
[4:3]		Not used
[2]	AWD0_IF	Interrupt flag for threshold monitoring 0 overrun
[1]	SF2_IF	Interrupt flag for the second-segment sampling completion
[0]	SF1_IF	Interrupt flag for the first-segment sampling completion

Only when the software/hardware triggers the ADC to do normal sampling and the ADC is doing normal sampling at that time; Or software/hardware triggers the ADC to idle sampling, and the trigger error interrupt flags ADC_IF[6:5] are generated when the ADC is idle sampling at that time.

For the above ADCx_IF flag bits, 0 indicates that no interrupt has occurred, and 1 indicates that an interrupt has occurred. Write 1 to clear.

12.2.9 Analog watchdog

12.2.9.1 ADC0_LTH

Address: 0x4001 04C4

Reset value: 0x0000_F800

Table 12-38 Lower threshold register ADC0_LTH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sign				LTH											
RO				RW											
0xF				0x800											

Location	Bit name	Description
[31:16]		Not used
[15:12]		Symbol extension
[11:0]	LTH	ADC Analog Watchdog 0 Lower Threshold

Only [11:0] can be written to the ADC0_LTH, and [15:12] are the sign extension bits when the ADC 0_LTH is read out. The ADC0_LTH uses bit [11] as the sign bit.

12.2.9.2 ADC0_HTH

Address: 0x4001 04C8

Reset value: 0x0000 07FF

Table 12-39 Upper threshold register ADC0_HTH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				HTH											
				RW											
				0x7FF											

Location	Bit name	Description
[31:16]		Not used
[15:12]		Symbol extension
[11:0]	HTH	ADC Analog Watchdog 0 Upper Threshold

Only [11:0] can be written to ADC0_HTH. When reading out, [15:12] are the sign extension bits, and the ADC0_HTH uses bit [11] as the sign bit.

12.2.9.3 ADC0_GEN

Address: 0x400 1_04CC

Reset value: 0x0

Table 12-40 Monitor enable register ADC0_GEN

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	GEN	Enable bit for ADC analog watchdog 0

		BIT0: DAT0 watchdog monitoring enabled BIT1: DAT1 watchdog monitoring enabled …… BIT13: DAT13 watchdog monitoring enabled BIT14: IDAT0 watchdog monitoring enabled BIT15: IDAT1 watchdog monitoring enabled

Since there is only one analog watchdog, there is only one set of high and low thresholds. Threshold monitoring is typically enabled for only one ADC_DAT register at a time. If multiple groups are enabled at the same time, multiple groups of ADC_DAT registers are monitored by the same high and low thresholds.

For example, if the ADC_GEN [1] = 1 is set, the watchdog 0 monitors the value of the ADC X_DAT1 and generates a corresponding interrupt if the value is greater than the ADC X_TH0.HTH or less than ADCx_TH0.LTH.

12.3 Application Guide

12.3.1 ADC Sampling Trigger Mode

The ADC supports one-segment and two-segment sampling modes. Each segment requires a specific external event to trigger the start. Each segment supports different sampling times and sampling signal channel configurations.

First trigger

ADC sampling can be triggered by the timing event TADC[0]/TADC[1]/TADC[2]/TADC[3] from MCPWM/UTimer, any one or several trigger samples of four trigger sources are available for selection, and it can also be triggered by writing command words to ADCx_SWT using 16'h5AA5 software.

The first segment is sampled

Judge whether it is a segment of sampling.

Yes: When the sampling times reaches the preset value ADCx_CHNT[3:0], the ADC will return to the idle state 0; if sampling times has not reached the preset value, the sampling continues.

No: When the sampling times reaches the preset value ADCx_CHNT [3:0], the ADC will enter the idle state 1 (the first round of two-round sampling is completed, waiting for the second round to be triggered); if the sampling times has not reached the preset value, the first round of sampling continues.

The second segment triggers

The second segment is sampled

When the sampling times of the second section reaches the preset value ADC_CHNT[7:4], this sampling is ended, and the ADC will return to the idle state 0..

Trigger conditions for various hardware trigger modes are summarized as followTableAs shown. The single-segment sampling mode is relatively special, and it can be set by the ADC_CFG register whether the sampling is triggered by one TADC event or multiple TADC events. The 2-segment and 4-segment sampling modes allow only one corresponding TADC event to trigger one segment of sampling.

In addition, the ADC module also supports triggering sampling by writing special values through software, and software triggering only supports triggering after writing once.

The trigger conditions of various hardware trigger modes are summarized in 错误!未找到引用源。. The single-round sampling mode is different. It can determine by the ADC_CFG register that whether the sampling is triggered by one TADC event or multiple TADC events, while the two-round sampling modes only support the sampling to be triggered by one MCPWM/UTimer timing event.

Besides, the ADC module also supports the sampling to be triggered by writing 0x5AA5 to ADC0_SWT through software. The software trigger also only supports trigger sampling by writing once.

Table 12-41 ADC Sampling Trigger Mode

	Single-segment trigger	Two-stage trigger
TADC trigger	None (TADC trigger is not enabled)	The first round of TADC[0] The second round of TADC[1]
	C times of TADC [0]	
	C times of TADC [1]	
	C times of TADC [2]	
	C times of TADC [3]	
	C times * TADC[0]/TADC[1]/ TADC[2]/TADC[3]	
Software trigger	Write 16'h5aa5 to the ADC_SWT.	First round: Write 0x5AA5 to ADC_SWT Second round: Write 0x5AA5 to ADC_SWT

* C is setted by ADC_CFG.SINGLE_TCNT. The ADC_CFG.SINGLE_TCNT is only used in single-segment triggering. If TADC[3:0] is enabled at the same time, all four trigger sources are counted and one ADC sample conversion is triggered by one to SINGLE_TCNT.

12.3.1.1 Single-segment trigger mode

Single-segment trigger mode means that the ADC receives one trigger to complete a segment of sampling action. A segment of sampling may include multiple sampling of analog signals. The number of times is configured by the segment sampling times register with ADCx_CHNT. When the register value is 1 to 15, the corresponding number of times is 1 to 15.

Assuming that the number of channels in a single-stage sampling configuration is 4, the sampled and converted data is filled in ADC_DAT0, ADC_DAT1, ADC_DAT2, and ADC_DAT3 in that order.

Trigger events can be the MCPWM/TIMER timing signals TADC [0], TADC [1], TADC [2], and TAAC [3] from external, occur a preset number of times, or are software triggers.

The signal source for each sample is configured through the signal source register ADC_CHN0/1/2/3. The signal source must be selected before triggering and should not be changed until a sample is completed.

After a period of sampling is completed, it enters the idle state and generates the sampling completion interrupt.

Take MCPWM triggering single-segment sampling as an example, set ADC_CFG. TCNT = 4 and ADC X_TRIG. MCPWM0_T2_EN = 1, that is, MCPWM0 is triggered after TADC [2] occurs 4 times, and the state transition is as followFigureAs shown.

Single-round triggering completes a sampling action when a trigger is received. One round of sampling may include multiple samplings of the analog signal, and the sampling times are set by the round register ADCx_CHNT; When the register value is 1 ~ 25, the corresponding sampling times are 1 ~ 15.

Assuming that the number of sampling channels for the single-round sampling is "4", the converted data will be filled into ADC_DAT0, ADC_DAT1, ADC_DAT2, and ADC_DAT3 in turn.

The trigger event can be triggered by the external MCPWM/UTimer timing signals TADC [0], TADC [1], TADC [2], TADC [3] to a preset number of times, or triggered by software.

Each sampled signal source is set and selected by the signal source register ADC_CHN0/1/2/3. The signal source should be selected before the trigger and shall remain unchanged before the sampling completed.

It will enter the idle state and generate a sampling completion interrupt upon the completion of one-round sampling.

Take the single-round sampling triggered by MCPWM as an example, set ADC_CFG.TCNT=4，ADCx_TRIG.MCPWM0_T2_EN =1,this is, the event is triggered when the TADC[2] occurs four times. The state transition is shown in **错误!未找到引用源。**.

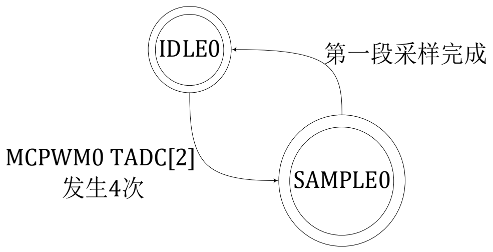


Figure 12-6 ADC Single Segment Sampling State Transition Diagram

12.3.1.2 Two-stage trigger mode

Two-segment triggering requires two triggers to complete a full round of sampling. The first trigger arrives with the first segment of sampling and the second trigger arrives with the second segment of sampling.

Assuming that the number of channels in the two-segment sampling configuration is 2 and 3 respectively, the converted data of the first segment will be filled into ADC0_DAT and ADC_DAT1 in sequence, and the converted data of the second segment will be filled into ADC_DAT2, ADC_DAT3 and ADC_DAT4 in sequence.

The trigger events can be the external MCPWM timing signals TADC [0] and TADC [1] or two software triggers.

After a TADC [0] or software trigger occurs, sample ADC_CHNT [3:0] times. After that, enter the idle state and wait for the next trigger signal. After a TADC [1] or software trigger occurs as the second trigger signal, sample ADC_CHNT [7:4] times. The sample count is configured using the Split Sample Count Register ADC_CHNT.

The signal source for each sample is selected through the register configuration. The selection of the signal source must be completed before triggering and should not be changed until a sample is completed.

The software trigger has a lower priority than the hardware trigger. When the software trigger occurs during the hardware trigger sampling, the state machine does not process it and generates an error interrupt. That is, software-triggered sampling requests are processed only when the state machine is idle. If you need to use software-triggered sampling, you need to make sure that hardware triggering is turned off. A software trigger is then generated by writing 0x5AA5 to the ADC_SWT register.

Take two software-triggered two-segment sampling as an example, the state transition is as shown in the figure12-7As shown.

Two-round trigger requires two triggers to complete one round of sampling. The first round is sampled when the first trigger arrives and the second round is sampled when the second trigger arrives.

Assuming that the number of channels in the two-segment sampling configuration is 2 and 3 respectively, the converted data of the first segment will be filled into ADC_DAT0 and ADC_DAT1 in sequence, and the converted data of the second segment will be filled into ADC_DAT2, ADC_DAT3 and ADC_DAT4 in sequence.

The trigger event can be triggered by external MCPWM/UTimer timing signals TADC[0] and TADC[1] or triggered twice by software.

When TADC [0] or software trigger occurs, the ADC_CHNT[3:0] sampling starts. And then, it will enter the idle state upon sampling completion and wait for the next trigger signal; When TADC[1] or software trigger occurs as the second trigger signal, the ADC_CHNT[7:4] sampling starts. The sampling times are set by the ADC_CHNT round register.

Each sampled signal source is set and selected by the register. The signal source should be selected before the trigger and shall remain unchanged before the sampling completed.

Software trigger has a lower priority than hardware trigger. If a software trigger occurs during the hardware-triggered sampling, the state machine will not process it, but generates an error interrupt. That is, the sampling request triggered by the software will be processed only when the state machine is in the idle state. If software trigger is required, the hardware trigger should be turned off in advance. Then write a 0x5AA5 to the ADC_SWT register to generate a software trigger.

Take the two-round sampling triggered by two software trigger as an example, the state transition is shown in Figure12-8.

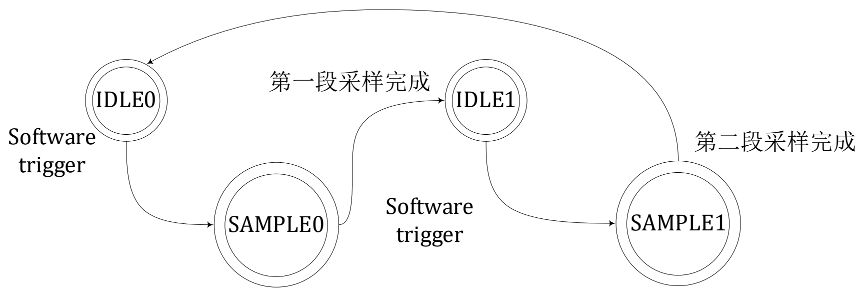


Figure 12-9 ADC Two-Segment Sampling State Transition Diagram

12.3.2 Interrupt

12.3.2.1 Single Segment Triggered Sampling Complete Interrupt

A done interruption is triggered when sampling is completed.

12.3.2.2 Two-segment triggered sample complete interrupt

A done interruption is triggered when the first-round sampling is completed, and another done interruption will be triggered when the second-round sampling is completed.

12.3.2.3 ADC_IF trigger Time

The CPU software runs in the PLL clock domain and the ADC module runs in the ADC clock domain. Typically, the ADC clock cycle is twice the PLL clock cycle (bus clock cycle). Clearing ADC IF requires operating registers in the ADC clock domain and takes up to 20 bus cycles. It is recommended that the ADC IF flag be read clear at the interrupt function entrance. If the clear statement is at the end of the interrupt function, the flag may not be cleared after the interrupt function exits, causing the interrupt to enter again.

Similarly, IF the software queries the ADC IF immediately after clearing the ADC IF, the ADC IF may remain the original value.

12.3.3 Configuration modification

It is recommended to configure and modify ADC_CHNx in the ADC interrupt. After entering the ADC interrupt, it means that the ADC has completed one round of sampling and is now in an idle state. Since the ADC operating status cannot be confirmed in the main program, the ADC trigger should be turned off in advance if the ADC_CHNx and ADC_CHNT registers should be modified in the main program, and then write "1" to ADC_CFG[11] to reset the ADC interface circuit state machine, thus ensuring that the ADC is not in a working state. If the ADC settings change during operation, the subsequent behavior will be unpredictable.

The example program is as follows

```
ADCx_CFG_temp = ADCx_CFG;         //Save ADCx_CFG
ADCx_CFG = 0x0000;                //Disable ADC trigger
ADCx_CFG = 0x0800;                  //Reset ADC state machine
/*
        Add your code below, like:
ADCx_CHNT = 0x0005
ADCx_CHN0 = 0x3210;
ADCx_CHN1 = 0x7654;
*/
ADCx_CFG = ADCx_CFG_temp;           //restore ADCx_CFG
```

12.3.4 Select the corresponding analog channel

ADC sampling signal source can be referred to Table 12.25 Pin Function Selection in DATASHEET. Turn off IE and OE of the corresponding IO to use its simulation function.

For the channel corresponding to the signal sampled by the ADC, please refer to Table 12.25 and Table 2-2 Pin Function Selection in DATASHEET. Turn off the corresponding IO IE and OE to us function.

13 TIMER General timer

13.1 Overview

13.1.1 Functional block diagram

As shown in Fig.13-1, The universal timer TIMER mainly includes four independent Timers, which can be independently configured to run the count clock and filter constant. Each Timer can be used to output a waveform with a specific duty cycle, or it can capture an external waveform to detect the duty cycle.

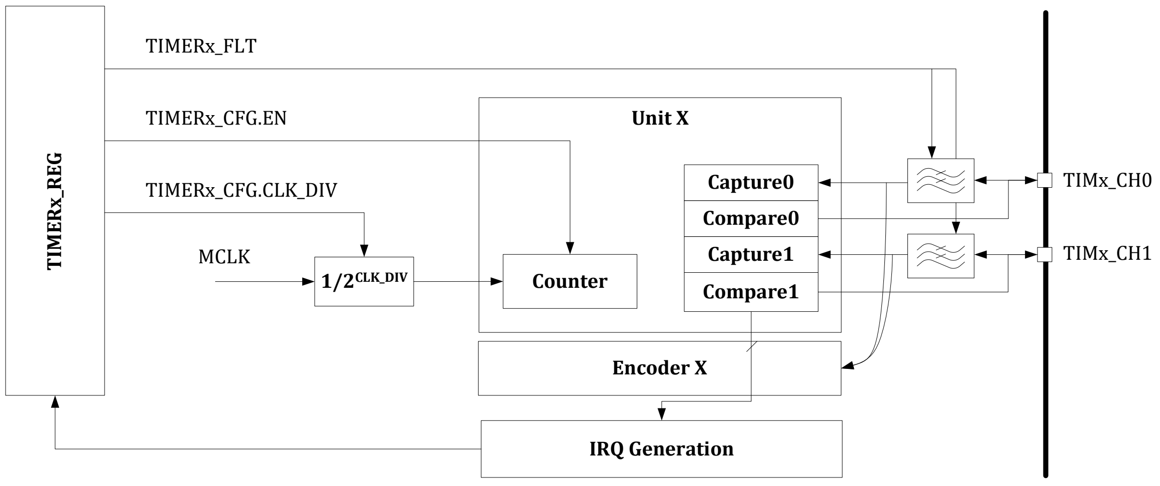


Figure 13-1 TIMER Module Top Level Functional Block Diagram

13.1.1.1 Register module

Read and write control registers of each submodule.

Access to the status and result registers of each submodule.

Interrupt signal processing and interrupt generation for each submodule.

13.1.1.2 IO filter module

The IO filter module filters the input signal from outside the chip to reduce the effect of glitches on the timer.

13.1.1.3 General Timer Module

The general timer module implements general timer functions, including compare and capture modes which can process two external input signals or generate two pulse signals to be transmitted outside the chip.

General timer module, supporting external events to trigger the start of counting. The external event source can be configured. When triggered by an external event, the timer begins to increment itself. Supports counting using an external signal as the timer clock.

13.1.1.4 Clock divider module

The clock frequency dividing module is used to generate various signals of clock frequency dividing.

13.1.2 Functional features

The timer module has the following features:

- Available for working in at different frequencies independently
- Timer0 and Timer1 are 16-bit general timers
- Timer2 and Timer3 are 32bit general timers
- Each general timer processes two external input signals (capture mode) or generates two output signals (comparison mode)
- Available for filtering each input signal of up to 2040 main clocks, that is, when the chip works at a clock frequency of 96MHz, it can filter out glitches below 21uS width.

13.2 Characteristic

13.2.1 Clock Division

Timer can reduce the counting frequency of the counter by dividing with the TIMERx_CFG. CLK_DIV, and the division factor can be set as 1/2/4/8/16/32/64/128.

13.2.2 The interrupt flag is cleared

A design is adopted in which the flag bit is cleared by writing 1 to each interrupt flag bit.

13.2.3 Filtering

Each timer block has two channels and two channels share a single filter coefficient when operating in input capture mode.

By configuring the filter register TIMERx_FLT, the filter width can be adjusted to the clock width of 0 ~ 2040 Timer counts.

Input signal filtering uses the frequency division clock of Timer count, and the frequency division of Timer count clock by TIMERx_CFG. CLK_DIV has an impact on the filtering clock, that is, the filtering width becomes larger after Timer frequency division.

Such as Figure13-2, the original input signal is flipped at several times from T1 to t 6, and the filter width is configured as T. It can be seen that only the inversions occurring at times T3 and T6 last longer than T, so the signal is seen to have only two inversions at the output of the filter.

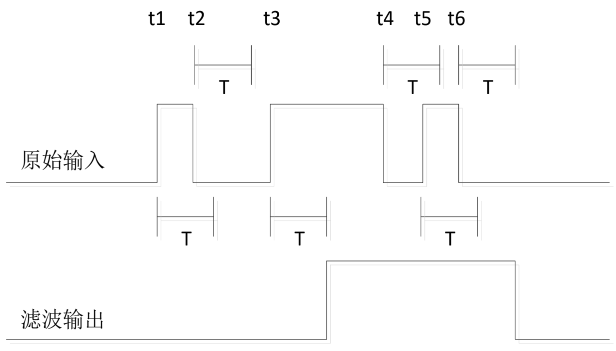


Figure 13-2 Schematic diagram of TIMER filtering

13.2.4 Mode

13.2.4.1 Counter

The counter in the Timer can be selected to count up or down, and the default is to count up.

If counting up, the counter counts from 0 to the value of th, and then returns to 0 to start counting again. When the counter returns to 0, a return-to-zero interrupt is generated. If counting down, the counter counts from TH to 0, and then returns to TH to start counting again. When the counter returns to 0, a return-to-zero interrupt is generated. The actual count period is CLK_freq * (TH + 1).

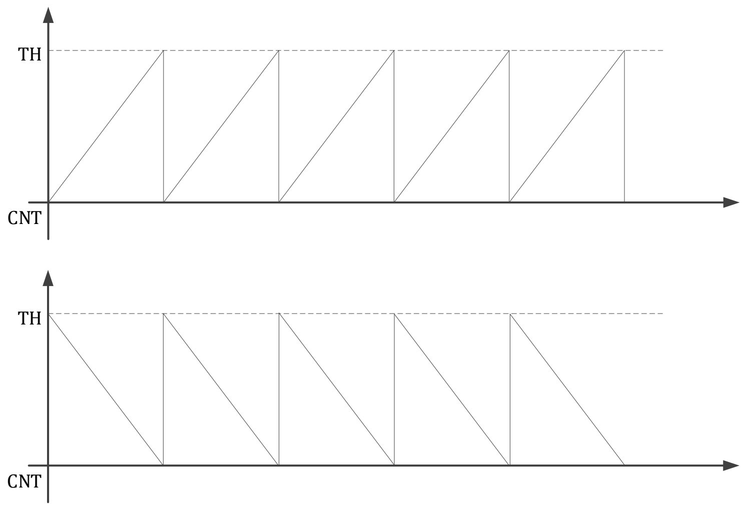


Figure 13-3 TIMER General counter

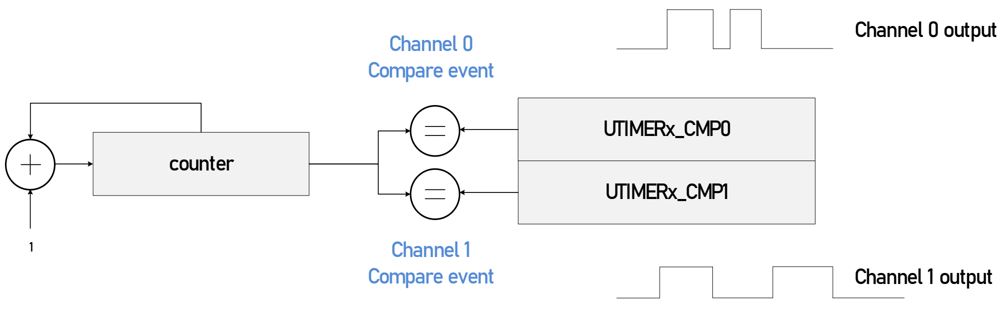
The counter's count clock can be configured through the TIMERx_CFG. XCLK_EN, using either the on-chip system clock (typically the on-chip 96MHz PLL clock) or the Timer Channel 0/1 signal as an external clock.

The counting clock of the counter can be divided by the TIMERx_CFG. CLK_DIV to reduce the counting frequency of the counter.

13.2.4.2 Capture mode

Timer can detect the rising/falling or double edge of the input signal in the capture mode. When a capture event (that is, the input signal level changes) occurs, the timer count value is stored in the

UTIMER_UNTx_CMP register and a capture interrupt is generated. When the counter returns to zero, the zero return interrupt will still be generated.



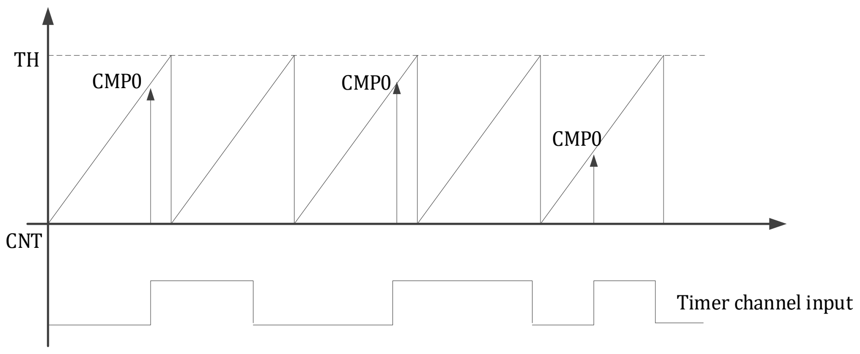


Figure 13-4 TIMER capture mode

As shown in Figure 13-4, the timer is set to capture on the rising edge. When changes in the rising edge of the input signal is captured at timing CAP0, CAP1, and CAP2, the timer count value at the corresponding time will be stored in the UTIMER_UNTx_CMP register.

In the capture mode, the signal source captured by Channel 0/1 can be set through the TIMERx_CFG. SRC0 and TIMERx_SRC1. The signal source can be set as the channel signal from the IO, or from the analog comparator or from the exclusive of the two IO channel signals.

The Timer can be automatically cleared by setting the TIMERx_CFG.CAP0_CLR_EN or the TIMERx_CFG.CAP1_CLR_EN, that is, when a capture event occurs on channel 0/1, the TIMER_CNT automatically returns to zero and starts counting again. This feature allows the Timer to calculate the period and duty cycle of the captured signal.

For example, if Channel 0/1 with Timer0 set to capture the same signal at the same time, Channel 0 captures the rising edge and Channel 1 captures the falling edge. And enable that CAP0_CLR_EN, that is, the TIMER_CNT automatically returns to zero when a channel 0 capture event occur. When the capture event of channel 0 (CAP0) occurs, the value recorded by TIMERx_CMP0 is the period of the previous signal cycle, and the value recorded by TIMERx_CMP1 is the high level duty cycle of the previous signal cycle.

Similarly, if channels 0/1 with Timer0 set capture the same signal at the same time, channel 0 captures the rising edge and channel 1 captures the falling edge. And enables the CAP1_CLR_EN, which automatically zeroes the TIMER_CNT when a Channel 1 capture event occurs. When the capture event of channel 1 (CAP1) occurs, the value recorded by TIMERx_CMP1 is the period of the previous cycle of the signal, and the value recorded by TIMERx_CMP0 is the low level duty cycle of the previous cycle.

13.2.4.3 Compare mode (edge-aligned PWM)

In compare mode, a compare pulse can be driven, and a compare interrupt is generated when the counter counts to the UTIMER_UNTx_CMP value. When returning to zero, it will output a level to the IO port (polarity can be configured); When the comparison event occurs, the level is inverted, and another level is output to the IO port. When the counter returns to zero, the zero return interrupt will still be generated. Set TIMERx_CMP0 = 0 to make Timer X channel 0 constant 1. Set TIMERx_CMP0 = TIMERx_TH + 1 to make Timer channel 0 constant 0.

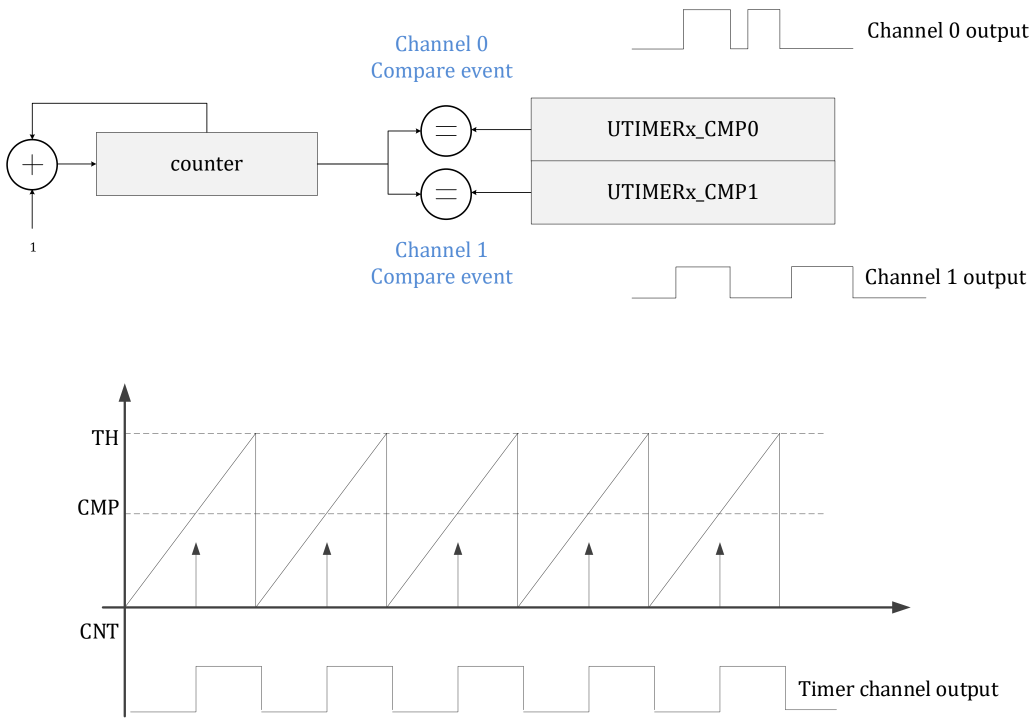


Figure 13-5 TIMER comparison mode

13.2.4.4 Single trigger

In compare mode, when the TIMERx_CFG.EN = 0, that is, the Timer does not start counting, the Timer can be triggered to emit a waveform with a specific duty cycle by writing a 1 to the TIMERx_CFG. ONE_TRIG. After that, the Timer returns to the idle state, and the channel no longer acts. If TIMER single trigger is still in the counting period, TIMER will not respond and continue to complete the counting of this period. That is, software one-shots that occur too frequently during one-shot counting are not accepted by TIMER. As shown in the following figure, writing 1 to the ONE_TRIG for the third time is invalid, and TIMER will not be triggered to restart counting.

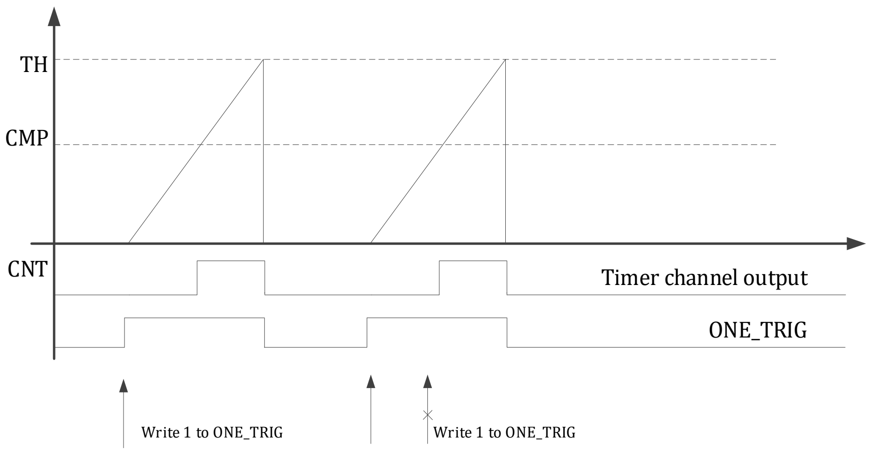


Figure 13-6 TIMER single trigger

13.2.4.5 Center Mode (Complementary PWM)

In addition to counting up and down, the counter can also be set to center counting mode, i.e., TIMERx_CFG.CENTER = 1. At this time, the counter counts up from 0 to TH and then counts down back to 0, starting the center count of the next cycle. The actual count period is CLK_freq*(2*TH + 1). The center count mode can be used to output complementary PWM waveforms. By setting TIMERx_CMP0/1 properly, the dead time of the upper and lower transistors can be generated. The dead time is TIMERx_CMP1-TIMERxCMP0. In addition, the initial values of the channels need to be set appropriately to generate the complementary waveforms. Timer has a TIMERx_CNT = TIMER_CMP0/1 during both the count up and count down phases, but only the count up phase generates an interrupt event.

In addition, Timer0 supports timely shutdown of the PWM output by a FAIL event. When the FAIL event occurs, the output value of Timer channel 0/1 is designated by TIMER0_IO. CH0_DEFAULT and TIMER0_IO. CH1_DEFAULT, and the TIMER 0_IO. MOE is cleared by the FAIL event. Timer outputs the PWM waveform by restoring.

Timer0 is also equipped with the shadow registers TIMER0_STH/TIMER0_SCMP0/TIER0_SCMP1. When the_CFG. SHADOW of TIMER0 = 1, it indicates that the shadow register is used as the action indication of the counter, and TIMER0_TH/TIMER0_CMP0/TIER0_CMP1 are only used as the preload registers. Writes to TIMER0_TH/TIMER0_CMP0/TIMERO_CMP1 modify only the preload values, not the shadow register values. The shadow registers of TIMER0 can be software updated by writing a 1 to the TIMER0_CFG.UPDATE; Or you can wait for the timer zero-crossing event to update automatically.

When the TIMER0_CFG.SHADOW = 0, just like the common Timer, TIMER0_TH/ TIMER0_CMP0/ Timer0_CMP1 is still used as the action indication of the counter.

The shadow register values TIMER0_STH/TIMER0_SCMP0/Timer0_SCMP1 are always returned when reading TIMERO_TH/TIMERO_CMP0/Timer0_CMP1.

Table 13-1 TIMER0 shadow register mapping

TIMER0_CFG. SHADOW Setpoint	0	1
TIMER0_CNT hit TH	TIMER0_TH	TIMER0_STH

TIMER0_CNT hit CMP0	TIMER0_CMP0	TIMER0_SCMP0
TIMER0_CNT hit CMP1	TIMER0_CMP1	TIMER0_SCMP1

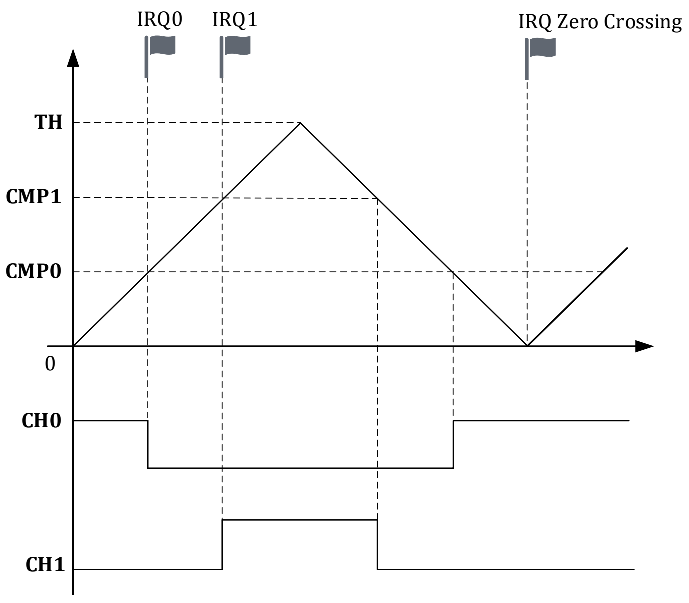


Figure13-7 TIMER Center Count Mode

TIMER0/1/2/3 all support center count mode for complementary PWM output; But only TIMER0 is equipped with a shadow register and a FAIL shutdown mechanism. That is, only TIMER0 exists in the TIMER0_IO register.

13.2.5 External events

Timer supports linkage with other Timer channels, MCPWM ADC trigger events, and CLU outputs. The external signal can be used as an external start signal, an external clock signal, an external reset signal and an external gate signal of the Timer.

External events have only one interpretation inside the Timer, so only one external event function can be used, and the above four functions cannot be used at the same time, that is, TIMERx_CFG. ETON, TIMERx_CFG. XCLK_EN, and TIMERx_CFG. RL_EN. Only one of TIMERx_CFG. GATE_EN four set bits can be 1.

The following description takes TIMER's positive count-up as an example, and the same applies to the count-down.

External events are selected through the TIMERx_EVT.

13.2.5.1.1 External start signal

When TIMERx_CFG. EN = 0 and TIMERx_CFG. ETON = 1, TIMERx stays at 0 until an external event occurs and starts counting when an external event occurs. After the external start signal is processed into a pulse signal, the start trigger of TIMER is carried out.

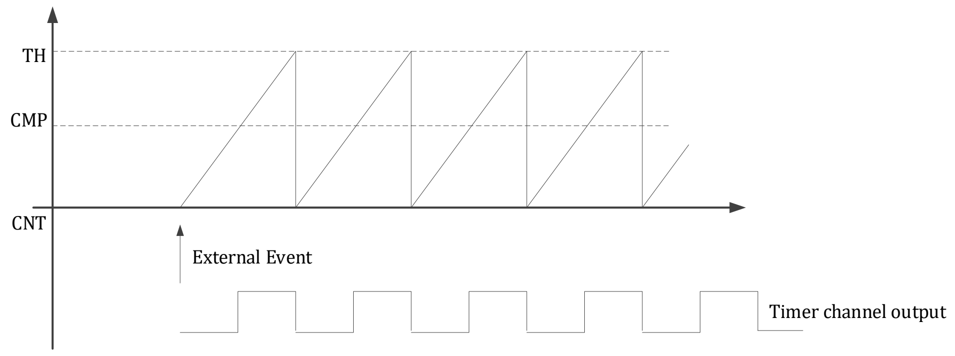


Figure 13-8 TIMER External Event Trigger Start

External start can also be used in combination with single trigger. At this time, the software needs to configure ETON = 1 first, and then configure ONE_TRIG = 1. After the external event is triggered, TIMER counts for a cycle. After the cycle is over, the ONE_TRIG is cleared by the hardware.

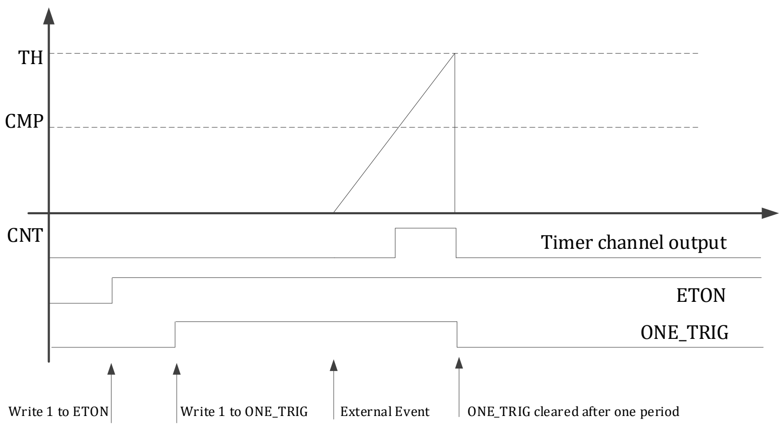


Figure13-9 TIMER external event triggers single (single cycle) count

13.2.5.1.2 External clock signal

By setting TIMERx_CFG. XCLK_EN = 1, the TIMER can use an external signal as the count clock. The selection of the external clock is set by the TIMERx_EVT.

The TIMERx_CNT is incremented (or decremented) when a rising edge occurs on the external signal. In particular, if the channel input of TIMERm is used as an external clock for TIMERx, the channel input of TIMERm is the signal filtered by TIMERm.

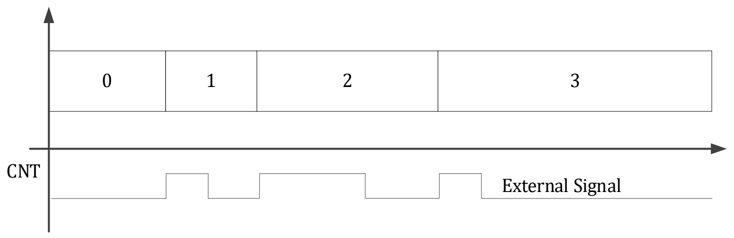


Figure 13-10 TIMER counts with an external clock

13.2.5.1.3 External reset signal

By setting TIMERx_CFG. RL_EN = 1, the TIMER can use an external signal as a reset signal. The selection of the external reset signal is set by the TIMERx_EVT. If TIMER is set to count up or center, the TIMERx_CNT is reset to 0 when an external reset occurs. If TIMER is set to count down, the TIMERx_CNT is reset to the TIMERX_TH when an external reset occurs. The external reset signal is processed into a pulse signal to reset the TIMER.

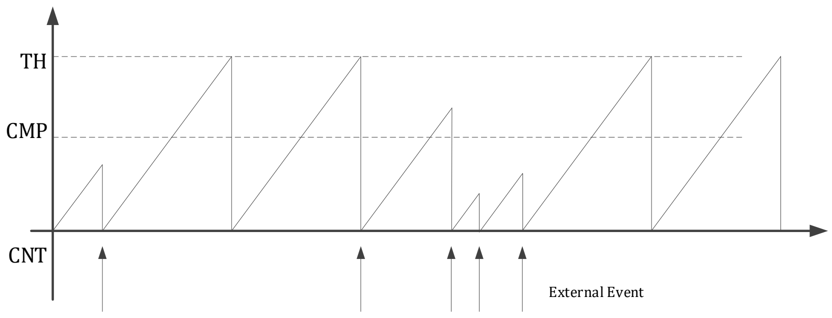


Figure 13-11 TIMER external signal reset function

13.2.5.1.4 External gate signal

By setting TIMERx_CFG. GATE_EN = 1, the TIMER can use an external signal as a gating signal. The selection of the external gate signal is set by the TIMERx_EVT. When the gate signal level is high, the TIMER counts; when the level is low, the TIMER stops counting, and the TIMERx_CNT remains unchanged. Because the gate signal needs to be a level signal, the ADC trigger signal of the MCPWM cannot be used as an external gate signal. In particular, if the channel input of TIMERm is used as the gating signal for TIMERx counting, the channel input signal is affected by the filtering of TIMERm.

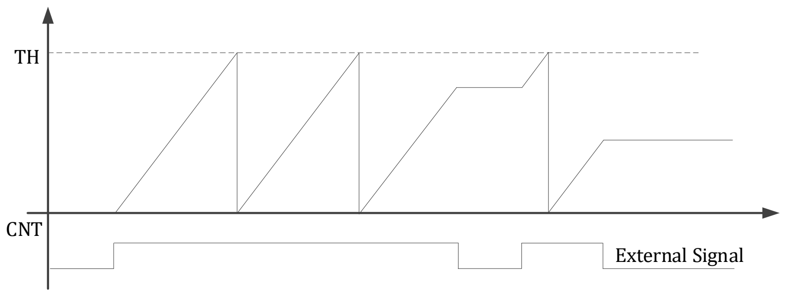


Figure13-12 TIMER external signal gating function

Take TIMER2 channel 1 as the external event signal of TIMER3 for example. You need to use TIMER2_CFG.SRC1 to set the signal of channel 1 of TIMER2 to come from IO TIM2_CH1, and use TIMER2_CFG.CH1_MODE to set the capture mode of channel 1 of TIMER2. In addition, it should be noted that the setting of TIMER2_FLT still has filtering effect on the input signal of TIMER2 channel 1. GPIO needs to be set up to allow signals to enter TIMER2 channel 1 through GPIO. Finally, channel 1 of TIMER2 is set through TIMER3_EVT to act as the external signal of TIMER3. Usually, only one of the four setting bits TIMER3_CFG.ETON, TIMER3_CFG.XCLK_EN, TIMER3_CFG.RL_EN, TIMER3_CFG.GATE_EN can be 1.

13.2.6 ADC trigger

The comparison event of Timer0/1 (CMP0/1) can be used as the ADC sampling trigger event.

13.2.7 Encoder

The encoder interface supports three modes: quadrature encoded signal, sign plus pulse signal, and CW/CCW double pulse signal.

The two input signals T1/T2 of QEP0 are respectively from the GPIO input corresponding to Timer2 Channel0/1, and are filtered inside Timer; The T1/T2 signals of QEP1 come from the corresponding GPIO inputs of Timer3 Channel0/1, respectively, and are filtered inside the Timer. When the encoder function is enabled, it does not affect the normal use of the Timer function.

The input T1/T2/Z signals of the encoder are controlled by the corresponding Timer filter coefficients. For example, the Z signal of QEP0 is controlled by TIMER2_FLT filter time, and the Z signal of QEP1 is controlled by TIMER3_FLT filter time.

Because the encoder multiplexes the Timer channel input and is affected by the Timer channel filtering, the corresponding Timer clock enable in the corresponding SYS_CLK_FEN needs to be turned on when using the encoder. If QEP0 is used, Time2 peripheral clock enable needs to be enabled. If QEP1 is used, Time3 peripheral clock enable needs to be enabled.

13.2.7.1 Quadrature coded signal

The quadrature coded signal is mostly used to count the number of encoder turns. The input signals are T1 and T2, which support the two modes in the following table.

Generally speaking, the transition edge of T1 and T2 will cause the counter to increment or decrement. The counter counting direction (increasing or decreasing) is determined by the level of another steady-state signal other than the transition signal.

If T1 has a rising edge transition, then check whether T2 is high or low level. If it is a high level, the counter decrements, if it is a low level, the counter increments; The change of T1 falling edge counter is just the opposite.

If T2 has a rising edge transition, then check whether T1 is high or low level. If it is a high level, the counter decrements, if it is a low level, the counter increments; The change of T2 falling edge counter is just the opposite.

The formula is as follows:

Counter Up = (T1 !=T2) @(T1 triggering edges) | (T1 == T2) @ (T2 triggering edges)

Counter Down = (T1 == T2) @ (T1 triggering edges) | (T1 != T2) @ (T2 triggering edges)

Table 13-2 Encoder orthogonal coding mode of operation

Counting mode	T1/T2 level status (steady state signal)	T1 change edge state		T2 Change Edge State	
		Rising edge	Falling edge	Rising edge	Falling edge
T1-only count	T2 high	Decreasing	Increment	Do not count	Do not count
	T2 is low	Increment	Decreasing	Do not count	Do not count
T1/T2 both count	T2 high	Decreasing	Increment	Do not count	Do not count
	T2 is low	Increment	Decreasing	Do not count	Do not count
	T1 is high	Do not count	Do not count	Increment	Decreasing
	T1 is low	Do not count	Do not count	Decreasing	Increment

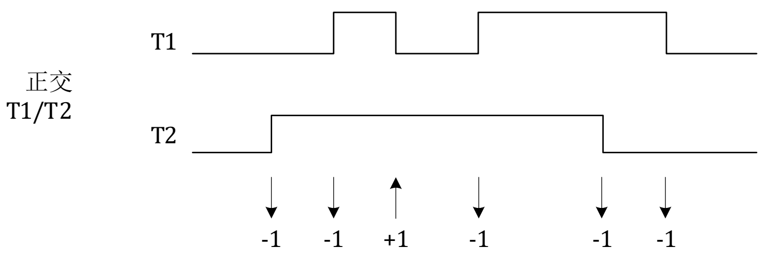


Figure 13-13 Quadrature Encoded Signal Count Case where Encoder Counts Only at Time T1

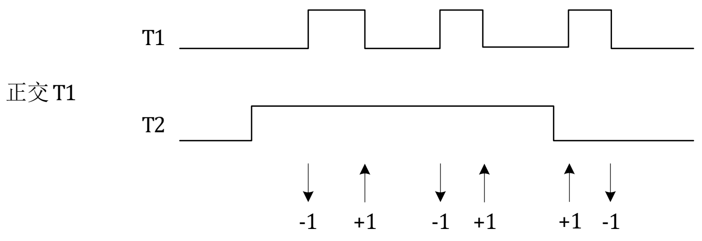


Figure 13-14 Quadrature Encoded Signal Count Case Counted by Encoder at Time T1 or T2

13.2.7.2 Sign plus pulse signal

In this mode of operation, T1 is a pulse signal, and T2 is a signed signal. The edge of T1 triggers counting and the level of T2 controls the counting direction; if it is a high level, it increases, and if it is a low level, it decreases. It can also be set to count only T1 rising edges or both T1 rising and falling edges.

Counter Up = (T2==1) @ (T1 triggering edges)

Counter Down = (T2==0) @ (T1 triggering edges)

Table 13-3 Coder symbol plus pulse mode of operation

Counting mode	T2 level status (steady state signal)	T1 change edge state	
		Rising edge	Falling edge
T1 rising edge	High	Increment	Do not count

only	Low	Decreasing	Do not count
T1 rising falling ed	High	Increment	Decreasing
	Low	Decreasing	Increment

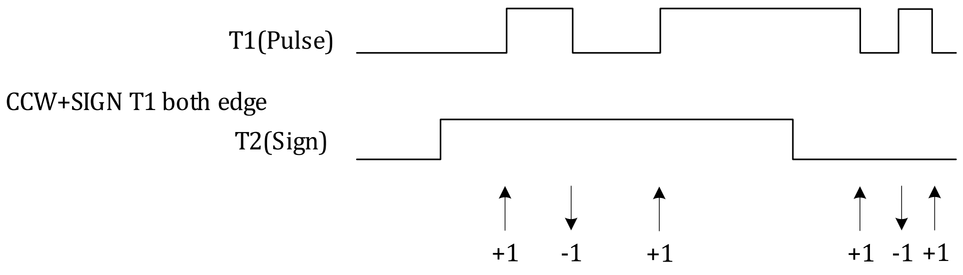


Figure 13-15 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts both on T1 Rising and Falling Edges

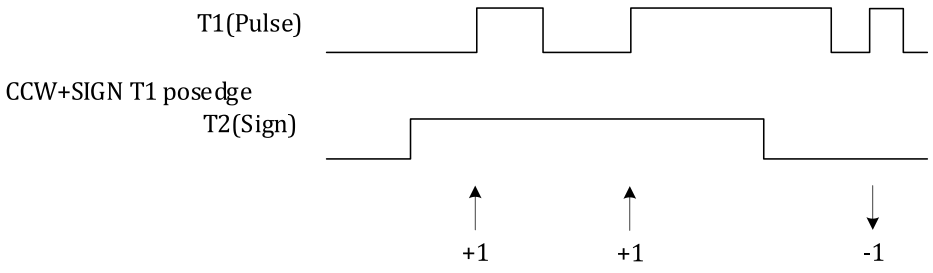


Figure 13-16 The Counts of Pulse Signal with Symbolic Data Type when the Encoder Counts only on T1 Rising Edges

13.2.7.3 CCW/CW double pulse signal

The counter increments when T1 transitions, and decrements when T2 transitions. It can also be set to count only the rising edges or both the rising and falling edges. The formula is as follows:

Counter Up	= 1 @ (T1 triggering edges)
Counter Down	= 1 @ (T2 triggering edges)

Table 13-4 Encoder CCW/CW Double Pulse Operation Mode

Counting mode	Change the edge state			
	T1 rising edge	T1 falling edge	T2 rising edge	T2 falling edge
T1/T2 rising edge	Increment	Do not count	Decreasing	Do not count
T1/T2 rising/falling edges	Increment	Increment	Decreasing	Decreasing

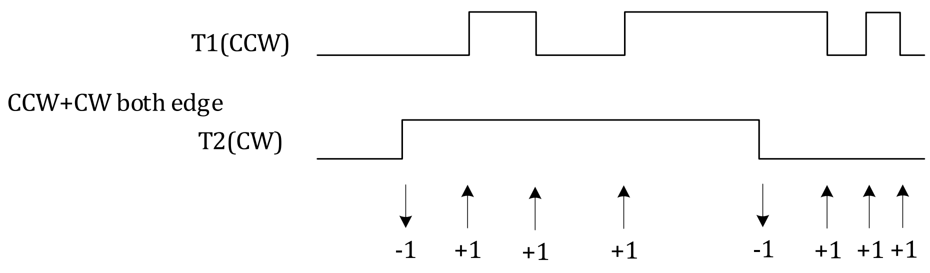


Figure 13-17 CCW/CW Double Pulse Signal Counting when Encoder Counts only on the T1 and T2 Rising Edge

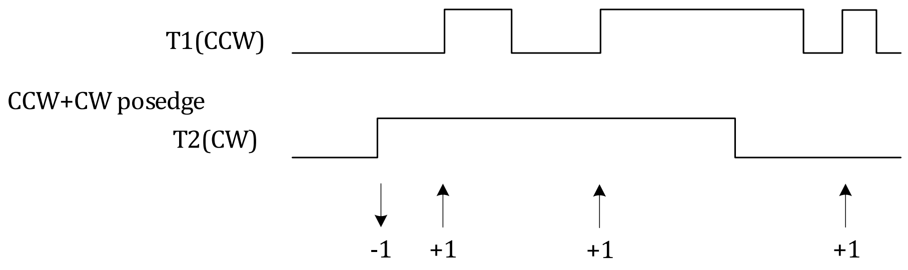


Figure 13-18 CCW/CW Double Pulse Signal Counting when Encoder Counts both on the Rising and Falling Edge T1 and T2

13.3 Register

13.3.1 Address assignment

The base address of Timer0 in the chip is 0x4001_0600

Table 13-5 Timer0 register address assignment

Name	Offset	Description
TIMER0_CFG	0x00	Timer0 configuration register
TIMER0_TH	0x04	Timer0 count threshold register
TIMER0_CNT	0x08	Timer0 count register
TIMER0_CMP0	0x0C	Timer0 Compare/Capture Register 0
TIMER0_CMP1	0x10	Timer0 Compare/Capture Register 1
TIMER0_EVT	0x14	Timer0 external event select register
TIMER0_FLT	0x18	Timer0 Filter Control Register
TIMER0_IE	0x1C	Timer0 Interrupt Enable Register
TIMER0_IF	0x20	Timer0 Interrupt Flag register
TIMER0_IO	0x24	Timer0 IO Control Register

The base address of Timer1 in the chip is 0x4001_0700

Table 13-6 Timer1 register address assignment

TIMER1_CFG	0x00	Timer1 configuration register
TIMER1_TH	0x04	Timer1 count threshold register
TIMER1_CNT	0x08	Timer1 count register
TIMER1_CMP0	0x0C	Timer1 Compare/Capture Register 0

TIMER1_CMP1	0x10	Timer1 Compare/Capture Register 1
TIMER1_EVT	0x14	Timer1 external event select register
TIMER1_FLT	0x18	Timer1 Filter Control Register
TIMER1_IE	0x1C	Timer1 Interrupt Enable Register
TIMER1_IF	0x20	Timer1 Interrupt Flag register

The base address of Timer2 in the chip is 0x4001_0800.

Table 13-7 Timer2 register address assignment

TIMER2_CFG	0x00	Timer2 configuration register
TIMER2_TH	0x04	Timer2 count threshold register
TIMER2_CNT	0x08	Timer2 count register
TIMER2_CMP0	0x0C	Timer2 Compare/Capture Register 0
TIMER2_CMP1	0x10	Timer2 Compare/Capture Register 1
TIMER2_EVT	0x14	Timer2 external event select register
TIMER2_FLT	0x18	Timer2 filter control register
TIMER2_IE	0x1C	Timer2 Interrupt Enable Register
TIMER2_IF	0x20	Timer2 Interrupt Flag register

The base address of Timer3 in the chip is 0x4001_0900

Table 13-8 Timer3 register address assignment

TIMER3_CFG	0x00	Timer3 configuration register
TIMER3_TH	0x04	Timer3 count threshold register
TIMER3_CNT	0x08	Timer3 count register
TIMER3_CMP0	0x0C	Timer3 Compare/Capture Register 0
TIMER3_CMP1	0x10	Timer3 Compare/Capture Register 1
TIMER3_EVT	0x14	Timer3 external event select register
TIMER3_FLT	0x18	Timer3 Filter Control Register
TIMER3_IE	0x1C	Timer3 Interrupt Enable Register
TIMER3_IF	0x20	Timer3 Interrupt Flag register

The difference between Timer0/1/2/3 is that the Timer0/1 counter-related registers are 16 bits wide, while the Timer2/3 counter-related registers are 32 bits wide, that is, the TIMERx_TH, TIMERx_CMP0, and TIMERx_CMP1 are different bits wide.

And there is a shadow register in TH/CMP0/CMP1 of Timer0, you can choose whether to enable the shadow register, if not, Timer0 and Timer1 are the same. If the shadow register is enabled, only the preload value is written to the above three registers, and the preload value is loaded into the shadow register when a zero-crossing event occurs in Timer0; Read the value of the shadow register read by TIMER0_TH/CMP0/CMP1.

The base address of QEP0 in the chip is 0x4001_0A00

Table 13-9 QEP0 register address assignment

QEP0_CFG	0x00	QEP0 Configuration Register
QEP0_TH	0x04	QEP0 Count Threshold Register
QEP0_CNT	0x08	QEP0 Count Register
QEP0_IE	0x0C	QEP0 Interrupt Enable Register

QEP0_IF	0x10	QEP0 Interrupt Flag register

The base address of QEP1 in the chip is 0x4001_0B00

Table 13-10 QEP1 register address assignment

QEP1_CFG	0x00	QEP1 configuration register
QEP1_TH	0x04	QEP1 Count Threshold Register
QEP1_CNT	0x08	QEP1 Count Register
QEP1_IE	0x0C	QEP1 Interrupt Enable Register
QEP1_IF	0x10	QEP1 Interrupt Flag register

13.3.2 TIMER0 register

13.3.2.1 TIMER0_CFG Timer0 configuration register

Address: 0x4001 0600

Reset value: 0x0

Table 13-11 Timer0 configuration register TIMER0_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN	UPDATE	SHADOW	ONE_TRIG	CENTER	DIR	CLK_DIV			ETON	GATE_EN	RL_EN	XCLK_EN
RW		RW	RW	WO	RW	RW	RW	RW	RW			RW	RW	RW	RW
0		0	0	0	0	0	0	0	0			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	CH0_MODE	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Location	Bit name	Description
[31]	EN	Timer block overall enable, active high
[30]		Not used
[29]	CAP1_CLR_EN	When a CAP1 capture event occurs, the Timer counter is cleared, active high
[28]	CAP0_CLR_EN	When a CAP0 capture event occurs, the Timer counter is cleared, active high
[27]	UPDATE	Software updates When the shadow registers are enabled, a 1 is written to UPDATE to load the preload value of TH/CMP0/CMP1 into the shadow registers STH/SCMP0/SCMP1. It is automatically cleared, and the readback is always 0. Writing 0 has no effect.

		Manually update the shadow register TIMER0_CFG | = BIT27; is recommended to prevent other configurations from being reset by mistake during manual update.
[26]	SHADOW	0: The shadow register is not enabled, and the corresponding shadow register is directly updated when the software writes to TH/CMP0/CMP1. 1: Enable the shadow register. When software writes to TH/CMP0/CMP1, only the preload value is updated. The preload value is written to the shadow register only when the Timer has a zero-crossing event.
[25]	ONE_TRIG	Single transmit mode, this bit needs to be used in Timer compare mode and EN needs to be set to 0 When ETON is 0, software writes a 1 to the ONE_TRIG to trigger the Timer to send a pulse with a specific duty cycle for one cycle, and the Timer stops counting after one cycle. This bit is read as 1 during the current cycle of the Timer count and is automatically cleared when the Timer is stopped. When ETON is 1, even if the software configuration ONE_TRIG = 1, it is necessary to wait for an external event to trigger, and the Timer stops after counting one cycle. The ONE_TRIG bit is cleared by hardware after the externally triggered Timer stops counting one cycle.
[24]	CENTER	Center count mode enabled 0: Timer counts up from 0 to TH and back to 0, or Timer counts down from TH to 0 and back to TH 1: Timer counts up from 0 to TH, then down to 0
[23]	DIR	0:0-> TH count up, 1: TH-> count down This bit is read only when CENTER = 1 and is used to indicate the current count direction
[22:20]	CLK_DIV	Timer counter frequency configuration. The counter count frequency is divided by 2^{CLK_DIV} of the system master clock frequency. The default value is 0 and does not divide. 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division
[19]	ETON	Timer counter external start enabled 0: Run automatically 1: Wait for external event trigger counting, and external trigger signal is selected according to TIMER0_EVT. EVT_SRC. Note that using an external trigger also requires setting TIMERx_CFG. EN = 1, unless it is a single trigger from an external signal, that is, TIMERx_CFG. ONE_TRIG = 1.
[18]	GATE_EN	Timer pause enabled 0: No pause 1: When the external signal is low, the Timer stops counting, and the external signal is selected according to the TIMER0_EVT. EVT_SRC.
[17]	RL_EN	Timer reload enabled 0: External event reload is disabled. Timer counts up to TH and returns to 0, or counts down to 0 and returns to TH. 1: external event reloading is enabled, and the reloading signal is selected according to the TIMER0_EVT. EVT_SRC. When counting up,

		an external event occurs, and the Timer is reloaded to 0; When counting down, an external event occurs and the Timer is reloaded to TH.
[16]	XCLK_EN	Timer clock source 0: Chip internal clock 1: external clock. The clock source is selected according to the TIMER0_EVT. EVT_SRC. When the external clock is used, the CLK_DIV no longer works, that is, the clock is no longer divided.
[15:12]	SRC1	Timer capture mode channel 1 signal source. The default is 3'h1. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (seeData sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[11]	CH1_POL	Output polarity control of Timer channel 1 in comparison mode, output value when counter CNT < CMP1.
[10]	CH1_MODE	Operation mode selection for Timer channel 1. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 1 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 1 input signal, the counter count is stored in the Timer Channel 1 Compare Capture Register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Timer capture mode channel 0 signal source. The default is 3'h0. 0: Timer channel 0, from chip GPIO (see Datasheet and application configuration) 1: Timer channel 1, from chip GPIO (see Datasheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[3]	CH0_POL	Output polarity control of Timer channel 0 in comparison mode: output value when counter CNT < CMP0. The output value at.
[2]	CH0_MODE	The working mode selection of Timer channel 0. The default value is 0.

		0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 0 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 0 input signal, the counter count is stored in the Timer Channel 0 Compare Capture Register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

When counting with an external clock, it is also necessary to set the TIMERx_CFG. TON = 1 and turn on the enable for the Timer clock in the SYS_CLK_FEN. The TIMERx_TH also needs to be set when counting with an external clock.

Generally, the external event is only used for external start, external gate, external clock and external reset, that is, only 1 bit is set to 1 in the TIMERx_CFG[19:16].

If SRC0 is set as 4'h8 to capture the Xor values of the two channels (usually capture the two orthogonal channels of the encoder), it is necessary to set CMP0_CLR_EN = 1, CH0_FE_CAP_EN = 1, CH0_RE_CAP_EN = 1 and CH0_MODE = 1 at the same time, that is, to capture the rising and falling edges. And clear that counter each time there is an edge of the signal. CMP0 is the count between two captured signal edges.

13.3.2.2 TIMER0_TH Timer0 threshold register

Address: 0x400 1_0604

Reset value: 0x0

Table 13-12 Timer0 threshold register TIMER0_TH

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	TH	Timer counter count threshold. Count up counts from 0 to the value of TH and back to 0, or count down counts from TH to 0 and back to TH

13.3.2.3 TIMER0_CNT Timer0 count register

Address: 0x4001 0608

Reset value: 0x0

Table 13-13 Timer0 count register TIMER0_CNT

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	CNT	Timer 0 counter current count value. A write operation can write a new count value.

Note that the Timer0 clock needs to be turned on by the SYS_CLK_FEN. TIMER0_CLK_EN before writing to TIMER0_CNT.

13.3.2.4 TIMER0_CMP0 Timer0 Channel 0 Compare Capture Register

Address: 0x4001 060C

Reset value: 0x0

Table 13-14 Timer0 Channel 0 Compare Capture Register TIMER0_CMP0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	CMP0	When Timer Channel 0 works in compare mode, a compare event occurs when the counter value equals CMP0. When Timer Channel 0 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP0 register.

Set CMP0 = 0 to make channel 0 constant! CH0_POL. Set CMP0 = TH + 1 to make channel 0 CH0_POL.

13.3.2.5 TIMER0_CMP1 Timer0 Channel 1 Compare Capture Register

Address: 0x4001 0610

Reset value: 0x0

Table 13-15 Timer0 Channel 1 Compare Capture Register TIMER0_CMP1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Location	Bit name	Description

[31:16]		Not used
[15:0]	CMP1	When Timer channel 1 operates in compare mode, a compare event occurs when the counter value equals CMP1. When Timer Channel 1 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP1 register.

Setting CMP 1 = 0 makes the channel constant! CH1_POL. Set CMP1 = TH + 1 to make the channel CH1_POL.

13.3.2.6 TIMER0_EVT Timer0 external event select register

Address: 0x4001 0614

Reset value: 0x0

Table 13-16 Timer0 external event select register TIMER0_EVT

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

Location	Bit name	Description
[31:5]		Not used
[3:0]	EVT_SRC	Timer external event selection register. This register needs to be used in conjunction with TIMER0_CFG. ETON, TIMER0_CFG. GATE_EN, TIMER0_CFG. RL_EN, and TIMER1_CFG. XCLK_EN. Normally, the 4 set bits are not used at the same time. When ETON = 1 is set, this register selects the event that triggers the Timer count. It should be noted that the Timer's own compare event cannot trigger the Timer to count. When ETON = 1, the rising edge of the trigger signal triggers the Timer to start counting. The Timer used as the trigger source needs to work in the comparison mode without inputting a signal from the outside to the Timer channel. 0: Not available 1: Not available 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output rising edge 9: CLU1 output rising edge 10: CLU2 output rising ed 11: CLU3 output rising ed 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event The Timer counts when the GATE_EN = 1 and the external signal is high, the Timer pauses when the external signal is low, and the Timer uses the level signal corresponding to the external signal as a count gate when

		the GATE_EN = 1. The GATE_EN can only be used in conjunction with external events from 0 to 11. When the EVT_SRC is selected as 0 to 7, the external signal is the input signal of the Timer channel after being filtered by the corresponding Timer. The Timer channel used as the gating signal needs to be configured as an input enable. 0: TIMER0 channel 0 filtered input signal 1: TIMER0 channel 1 filtered input signal 2: TIMER1 channel 0 filtered input signal 3: TIMER1 Channel 1 filtered input signal 4: TIMER2 channel 0 filtered input signal 5: TIMER2 Channel 1 filtered input signal 6: TIMER3 channel 0 filtered input signal 7: TIMER3 Channel 1 filtered input signal 8: CLU0 output signal 9: CLU1 output signal 10: CLU2 output signal 11: CLU3 output signal When RL_EN is high, the Timer uses the external signal as the reload signal Timer channels that are reloaded signals need to be configured as input enabled 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event When the XCLK_EN is high, the Timer uses an external signal as the external clock of the Timer, and the Timer counts on the rising edge of the clock. If a Timer channel is used as an external clock, the channel signal is affected by the corresponding Timer filter setting. 0: TIMER0 channel 0 input signal 1: TIMER0 channel 1 input signal 2: TIMER1 channel 0 input signal 3: TIMER1 channel 1 input signal 4: TIMER2 channel 0 input signal 5: TIMER2 channel 1 input signal 6: TIMER3 channel 0 input signal 7: TIMER3 channel 1 input signal 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event

		13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event

13.3.2.7 TIMER0_FLT Timer0 filter control register

Address: 0x4001 0618

Reset value: 0x0

Table 13-17 Timer0 filter control register TIMER0_FLT

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

Location	Bit name	Description
[31:8]		Not used
[7:0]	FLT	Channel 0/1 signal filter width selection. Value range: 0 ~ 255. When FLT is 0, the channel is not filtered. When FLT is not 0, filter the channel signal: the filter width is FLT × 8. When the channel signal level is stable and exceeds the width of FLT * 8 system clock cycles, the filter output is updated; Otherwise, the filter keeps the current output unchanged.

Note that the working clock of the above filter is the same as the working clock of the corresponding Timer after frequency division, and is controlled by the frequency division factor of the TIMERx_CFG. CLK_DIV.

Assume TIMERx_FLT = 0x6; TIMERx_CFG.CLK_DIV = 0x2; The running clock of the Timer is divided by 4 with respect to the system clock. The channel input signal needs to be filtered by an 8 × 6 Timer running clock, that is, 8 × 6 × 4 system clock.

13.3.2.8 TIMER0_IE Timer0 Interrupt Enable Register

Address: 0x4001 061C

Reset value: 0x0

Table13-18 Timer0 Interrupt Enable Register TIMER0_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				FAIL_RE	ZC_RE	CH1_RE	CH0_RE					FAIL_IE	ZC_IE	CH1_IE	CH0_IE
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

Location	Bit name	Description

[31:12]		Not used
[11]	FAIL_RE	Timer FAIL event DMA request enabled, active high.
[10]	ZC_RE	Timer counter 0 DMA request enabled, active high.
[9]	CH1_RE	Timer channel 1 compare/capture DMA request enable, active high.
[8]	CH0_RE	Timer Channel 0 Compare/Capture DMA request Enable, active High.
[7:4]		Not used
[3]	FAIL_IE	Timer FAIL event interrupt enabled, active high.
[2]	ZC_IE	Timer counter over-zero interrupt enabled, active high.
[1]	CH1_IE	Timer Channel 1 Compare/Capture Interrupt Enable, active High.
[0]	CH0_IE	Timer Channel 0 Compare/Capture Interrupt Enabled, active High.

The DMA request event is the same event flag as the interrupt. After the DMA request is accepted by the DMA, the interrupt flag will be automatically cleared by the DMA hardware without CPU software intervention. It is important to note that a DMA request that turns on an event typically turns off the corresponding interrupt enable.

13.3.2.9 TIMER0_IF Timer0 Interrupt Flag register

Address: 0x4001 0620

Reset value: 0x0

Table13-19 Timer0 interrupt flag register TIMER0_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												FAIL_IF	ZC_IF	CH1_IF	CH0_IF
												RW1C	RW1C	RW1C	RW1C
												0	0	0	0

Location	Bit name	Description
[31:4]		Not used
[3]	FAIL_IF	Timer Fail Event Interrupt Flag. Active high, write 1 to clear
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear
[1]	CH1_IF	Timer Channel 1 Compare/Capture Interrupt Flag. Active high, write 1 to clear
[0]	CH0_IF	Timer Channel 0 Compare/Capture Interrupt Flag. Active high, write 1 to clear

In compare mode, when Timer is set to center count (0- > TH- > 0), that is, TIMERx_CFG. CENTER = 1. CH0_IF and CH1_IF are set only when the TIMERx_CNT = TIMERx_CMP0/1 while the Timer is counting up from 0 to TH; The bit is not set while the Timer is counting down from TH to 0.

When TIMERx_CFG. DIR = 0

Write 1 to clear the interrupt flag. It is generally not recommended to use the following | = method to clear, because | = reads the interrupt flag first, changes the corresponding bit to 1, and then writes it to clear. If other interrupt flags are set at the same time, they will be cleared together, which is usually not expected by the software. For example, the following writing method is intended to clear the ZC_IF, but if CH0_if is set to 1 before writing, the software first reads back the TIMERx_IF value as 0x24, then executes the or operation 0x4 | 0x1 = 0x5, and then writes, clearing both CH0_if and the ZC_IF. May cause the Timer to enter one less interrupt due to capture.

TIMERx_IF|=0x4;

If you want to clear the ZC_IF flag, write a 1. Directly to BIT2 as follows.

TIMERx_IF=0x4;

13.3.2.10 TIMER0_IO Timer0 IO control register

Address: 0x4001 0624

Reset value: 0x0

Table 13-20 Timer0 IO control register TIMER0_IO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CH1_DEFAULT	CH0_DEFAULT	HALT_PRT	MOE				FAIL_SEL	FAIL_POL	FAIL_EN
						RW	RW	RW	RW				RW	RW	RW
						0	0	0	0				0	0	0

Location	Bit name	Description
[31:10]		Not used
[9]	CH1_DEFAULT	CH1 channel output value after MOE is cleared in the Fail event
[8]	CH0_DEFAULT	CH0 channel output value after MOE is cleared in the Fail event
[7]	HALT_PRT	The MCU enters the HALT state and the Timer output value is selected. 1: Output CH1_DEFAULT and CH0_DEFAULT; 0: Normal output
[6]	MOE	Timer Channel 0/1 output enabled when TIMER0_CFG. CENTER = 1 1: Output normal signal 0: Output CH0_DEFAULT and CH1_DEFAULT defaults A change of 1 in the TIMER0_IF. FAIL_IF causes the trigger MOE to become 0, and the channel outputs the default value. To restore the normal output signal, first clear the FAIL_IF and then set MOE = 1 by software
[5:3]		Not used
[2]	FAIL_SEL	FAIL signal selection 0: TIMER0_FAIL from GPIO 1: CLU0 output
[1]	FAIL_POL	FAIL signal polarity 0: High FAIL 1: Low FAIL
[0]	FAIL_EN	FAIL signal enabled 0: Disable FAIL 1: Enable FAIL

13.3.3 TIMER1 register

13.3.3.1 TIMER1_CFG Timer1 configuration register

Address: 0x4001 0700

Reset value: 0x0

Table13-21 Timer1 configuration register TIMER1_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN			ONE_TRIG	—	—	CLK_DIV			—	—	RL_EN	XCLK_EN
RW		RW	RW			RW	RW	RW	RW			RW	RW	RW	RW
0		0	0			0	0	0	0			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	—	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Location	Bit name	Description
[31]	EN	Timer block overall enable, high effective
[30]		Not used
[29]	CAP1_CLR_EN	When a CAP1 capture event occurs, the Timer counter is cleared, active high
[28]	CAP0_CLR_EN	When a CAP0 capture event occurs, the Timer counter is cleared, active high
[27:26]		Not used
[25]	ONE_TRIG	Single transmit mode, this bit needs to be used in Timer compare mode and EN needs to be set to 0 When ETON is 0, software writes a 1 to the ONE_TRIG to trigger the Timer to send a pulse with a specific duty cycle for one cycle, and the Timer stops counting after one cycle. This bit is read as 1 during the current cycle of the Timer count and is automatically cleared when the Timer is stopped. When ETON is 1, even if the software configuration ONE_TRIG = 1, it is necessary to wait for an external event to trigger, and the Timer stops after counting one cycle. The ONE_TRIG bit is cleared by hardware after the externally triggered Timer stops counting one cycle.
[24]	CENTER	Center count mode enabled 0: Timer counts up from 0 to TH and back to 0, or Timer counts down from TH to 0 and back to TH 1: Timer counts up from 0 to TH, then down to 0
[23]	DIR	0:0-> TH count up, 1: TH-> count down This bit is read only when CENTER = 1 and is used to indicate the current count direction

[22:20]	CLK_DIV	Timer counter frequency configuration. The counter count frequency is divided by $2^{CLK_DIV}$of the system master clock frequency. The default value is 0 and does not divide. 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division
[19]	ETON	Timer counter count external start enabled 0: Run automatically 1: Wait for external event trigger counting, and external trigger signal is selected according to TIMER1_EVT. EVT_SRC. Note that using an external trigger also requires setting TIMER1_CFG. EN = 1, unless it is a single trigger from an external signal, i.e. TIMER1_CFG. ONE_TRIG = 1.
[18]	GATE_EN	Timer pause enabled 0: No pause 1: When the external signal is low, the Timer stops counting, and the external signal is selected according to the TIMER1_EVT. EVT_SRC.
[17]	RL_EN	Timer reload enabled 0: External event reload is disabled. Timer counts up to TH and returns to 0, or counts down to 0 and returns to TH. 1: External event reloading is enabled, and the reloading signal is selected according to the_EVT. EVT_SRC of TIMER1. When counting up, an external event occurs, and the Timer is reloaded to 0. When counting down, an external event occurs and the Timer is reloaded to TH.
[16]	XCLK_EN	Timer clock source 0: Chip internal clock 1: external clock. The clock source is selected according to the_EVT. EVT_SRC of TIMER1. When the external clock is used, the CLK_DIV does not work, that is, the clock is no longer divided.
[15:12]	SRC1	Timer capture mode channel 1 signal source. The default is 3'h1. 0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[11]	CH1_POL	Output polarity control of Timer channel 1 in comparison mode, output value when counter CNT < CMP1.
[10]	CH1_MODE	Operation mode selection for Timer channel 1. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 1 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel

		1 input signal, the counter count is stored in the Timer Channel 1 Compare Capture Register.
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Timer capture mode channel 0 signal source. The default is 3'h0. 0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[3]	CH0_POL	Output polarity control of Timer channel 0 in comparison mode: output value when counter CNT < CMP0.
[2]	CH0_MODE	The working mode selection of Timer channel 0. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 0 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 0 input signal, the counter count is stored in the Timer Channel 0 Compare Capture Register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

When counting with an external clock, it is also necessary to set the TIMERx_CFG. TON = 1 and turn on the enable for the Timer clock in the SYS_CLK_FEN. The TIMERx_TH also needs to be set when counting with an external clock.

If SRC0 is set as 4'h8 to capture the Xor values of the two channels (usually capture the two orthogonal channels of the encoder), it is necessary to set CMP0_CLR_EN = 1, CH0_FE_CAP_EN = 1, CH0_RE_CAP_EN = 1 and CH0_MODE = 1 at the same time, that is, to capture the rising and falling edges. And clear that counter each time there is an edge of the signal. CMP0 is the count between two captured signal edges.

13.3.3.2 TIMER1_TH Timer 1 threshold register

Address: 0x4001 0704

Reset value: 0x0

Table 13-22 Timer1 threshold register TIMER 1_TH

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	TH	Timer counter count threshold. Count up counts from 0 to the value of TH and back to 0, or count down counts from TH to 0 and back to TH

13.3.3.3 TIMER1_CNT Timer1 count register

Address: 0x4001 0708

Reset value: 0x0

Table 13-23 Timer1 count register TIMER1_CNT

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	CNT	Timer 0 counter current count value. A write operation can write a new count value.

Note that the Timer1 clock needs to be turned on by the SYS_CLK_FEN. TIMER1_CLK_EN before writing TIMER1_CNT.

13.3.3.4 TIMER1_CMP0 Timer1 Channel 0 Compare Capture Register

0x4001 070C

Reset value: 0x0

Table 13-24 Timer1 Channel 0 Compare Capture Register TIMER1_CMP0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Location	Bit name	Description

[31:16]		Not used
[15:0]	CMP0	When Timer Channel 0 works in compare mode, a compare event occurs when the counter value equals CMP0. When Timer Channel 0 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP0 register.

Set CMP0 = 0 to make channel 0 constant! CH0_POL. Set CMP0 = TH + 1 to make channel 0 CH0_POL.

13.3.3.5 TIMER1_CMP1 Timer1 Channel 1 Compare Capture Register

Address: 0x4001 0710

Reset value: 0x0

Table 13-25 Timer1 channel 1 compare capture register TIMER1_CMP1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Location	Bit name	Description
[31:16]		Not used
[15:0]	CMP1	When Timer channel 1 operates in compare mode, a compare event occurs when the counter value equals CMP1. When Timer Channel 1 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP1 register.

Setting CMP 1 = 0 makes the channel constant! CH1_POL. Set CMP1 = TH + 1 to make the channel CH1_POL.

13.3.3.6 TIMER1_EVT Timer1 external event select register

Address: 0x4001 0714

Reset value: 0x0

Table 13-26 Timer1 external event select register TIMER1_EVT

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

Location	Bit name	Description
[31:5]		Not used
[3:0]	EVT_SRC	Timer external event selection register. This register needs to be used in conjunction with TIMER1_CFG. ETON, TIMER1_CFG. GATE_EN, TIMER1_CFG. RL_EN, and TIER1_CFG. XCLK_EN. Normally, the 4 set bits are not used at the same time. When ETON = 1 is set, this register selects the event that triggers the Timer count. It should be noted that the Timer's own compare event cannot trigger the Timer to count. When ETON = 1, the rising edge of the trigger signal

		triggers the Timer to start counting. The Timer used as the trigger source needs to work in the comparison mode without inputting a signal from the outside to the Timer channel. 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: Not available 3: Not available 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output rising edge 9: CLU1 output rising edge 10: CLU2 output rising ed 11: CLU3 output rising ed 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event The Timer counts when the GATE_EN = 1 and the external signal is high, the Timer pauses when the external signal is low, and the Timer uses the level signal corresponding to the external signal as a count gate when the GATE_EN = 1. The GATE_EN can only be used in conjunction with external events from 0 to 11. When the EVT_SRC is selected as 0 to 7, the external signal is the input signal of the Timer channel after being filtered by the corresponding Timer. The Timer channel used as the gating signal needs to be configured as an input enable. 0: TIMER0 channel 0 filtered input signal 1: TIMER0 channel 1 filtered input signal 2: TIMER1 channel 0 filtered input signal 3: TIMER1 Channel 1 filtered input signal 4: TIMER2 channel 0 filtered input signal 5: TIMER2 Channel 1 filtered input signal 6: TIMER3 channel 0 filtered input signal 7: TIMER3 Channel 1 filtered input signal 8: CLU0 output signal 9: CLU1 output signal 10: CLU2 output signal 11: CLU3 output signal When RL_EN is high, the Timer uses the external signal as the reload signal Timer channels that are reloaded signals need to be configured as input enabled 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output 9: CLU1 output 10: CLU2 output

		11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event When the XCLK_EN is high, the Timer uses an external signal as the external clock of the Timer, and the Timer counts on the rising edge of the clock. If a Timer channel is used as an external clock, the channel signal is affected by the corresponding Timer filter setting. 0: TIMER0 channel 0 input signal 1: TIMER0 channel 1 input signal 2: TIMER1 channel 0 input signal 3: TIMER1 channel 1 input signal 4: TIMER2 channel 0 input signal 5: TIMER2 channel 1 input signal 6: TIMER3 channel 0 input signal 7: TIMER3 channel 1 input signal 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event

13.3.3.7 TIMER1_FLT Timer1 filter control register

Address: 0x4001 0718

Reset value: 0x0

Table 13-27 Timer1 filter control register TIMER1_FLT

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

Location	Bit name	Description
[31:8]		Not used
[7:0]	FLT	Channel 0/1 signal filter width selection. Value range: 0 ~ 255. When FLT is 0, the channel is not filtered. When FLT is not 0, filter the channel signal: the filter width is FLT × 8. When the channel signal level is stable and exceeds the width of FLT * 8 system clock cycles, the filter output is updated; Otherwise, the filter keeps the current output unchanged.

Note that the working clock of the above filter is the same as the working clock of the corresponding Timer after frequency division, and is controlled by the frequency division factor of the TIMERx_CFG. CLK_DIV.

Assume TIMERx_FLT = 0x6; TIMERx_CFG.CLK_DIV = 0x2; The running clock of the Timer is divided by 4 with respect to the system clock. The channel input signal needs to be filtered by an 8 × 6 Timer running clock, that is, 8 × 6 × 4 system clock.

13.3.3.8 TIMER1_IE Timer1 Interrupt Enable Register

Address: 0x4001 071C

Reset value: 0x0

Table 13-28 Timer1 Interrupt Enable Register TIMER1_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	ZC_RE	Timer counter 0 DMA request enabled, active high.
[9]	CH1_RE	Timer channel 1 compare/capture DMA request enable, active high.
[8]	CH0_RE	Timer channel 0 compare/capture DMA request enable, active High.
[7:3]		Not used
[2]	ZC_IE	Timer counter over-zero interrupt enabled, active high.
[1]	CH1_IE	Timer channel 1 compare/capture interrupt enable, active High.
[0]	CH0_IE	Timer channel 0 compare/capture interrupt enable, active High.

The DMA request event is the same event flag as the interrupt. After the DMA request is accepted by the DMA, the interrupt flag will be automatically cleared by the DMA hardware without CPU software intervention. It is important to note that a DMA request that turns on an event typically turns off the corresponding interrupt enable.

13.3.3.9 TIMER1_IF Timer1 Interrupt Flag register

Address: 0x4001 0720

Reset value: 0x0

Table 13-29 Timer1 interrupt flag register TIMER1_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF

	RW1C	RW1C	RW1C
	0	0	0

Location	Bit name	Description
[31:3]		Not used
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear 0
[1]	CH1_IF	Timer Channel 1 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0
[0]	CH0_IF	Timer Channel 0 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0

In compare mode, when Timer is set to center count (0- > TH- > 0), that is, TIMERx_CFG. CENTER = 1. CH0_IF and CH1_IF are set only when the TIMERx_CNT = TIMERx_CMP0/1 while the Timer is counting up from 0 to TH; The bit is not set while the Timer is counting down from TH to 0.

Write 1 to clear the interrupt flag. It is generally not recommended to use the following | = method to clear, because | = reads the interrupt flag first, changes the corresponding bit to 1, and then writes it to clear. If other interrupt flags are set at the same time, they will be cleared together, which is usually not expected by the software. For example, the following writing method is intended to clear the ZC_IF, but if CH0_if is set to 1 before writing, the software first reads back the TIMERx_IF value as 0x24, then executes the or operation 0x4 | 0x1 = 0x5, and then writes, clearing both CH0_if and the ZC_IF. May cause the Timer to enter one less interrupt due to capture.

TIMERx_IF|=0x4;

If you want to clear the T0_ZC_IF flag, write a 1. Directly to BIT2 as follows.

TIMERx_IF=0x4;

13.3.4 TIMER2 register

13.3.4.1 TIMER2_CFG Timer2 configuration register

Address: 0x4001 0800

Reset value: 0x0

Table 13-30 Timer2 configuration register TIMER2_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN			ONE_TRIG	—	—	CLK_DIV			—	—	RL_EN	XCLK_EN
RW		RW	RW			RW	RW	RW	RW			RW	RW	RW	RW
0		0	0			0	0	0	0			0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SRC1	CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0	CH0_POL	—	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0001	0	0	0	0	0	0	0	0	0

Location	Bit name	Description
[31]	EN	Timer block overall enable, high effective
[30]		Not used
[29]	CAP1_CLR_EN	When a CAP1 capture event occurs, the Timer counter is cleared, active high
[28]	CAP0_CLR_EN	When a CAP0 capture event occurs, the Timer counter is cleared, active high
[27:26]		Not used
[25]	ONE_TRIG	Single transmit mode, this bit needs to be used in Timer compare mode and EN needs to be set to 0 When ETON is 0, software writes a 1 to the ONE_TRIG to trigger the Timer to send a pulse with a specific duty cycle for one cycle, and the Timer stops counting after one cycle. This bit is read as 1 during the current cycle of the Timer count and is automatically cleared when the Timer is stopped. When ETON is 1, even if the software configuration ONE_TRIG = 1, it is necessary to wait for an external event to trigger, and the Timer stops after counting one cycle. The ONE_TRIG bit is cleared by hardware after the externally triggered Timer stops counting one cycle.
[24]	CENTER	Center count mode enabled 0: Timer counts up from 0 to TH and back to 0, or Timer counts down from TH to 0 and back to TH 1: Timer counts up from 0 to TH, then down to 0
[23]	DIR	0:0-> TH count up, 1: TH-> count down This bit is read only when CENTER = 1 and is used to indicate the current count direction
[22:20]	CLK_DIV	Timer counter frequency configuration. The counter count frequency is divided by $2^{CLK_DIV}$of the system master clock frequency. The default value is 0 and does not divide. 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division
[19]	ETON	Timer counter external start enabled 0: Run automatically 1: Wait for external event trigger counting, and external trigger signal is selected according to TIMER2_EVT. EVT_SRC. Note that using an external trigger also requires setting TIMER2_CFG. EN = 1, unless it is a single trigger from an external signal, i.e. TIMER2_CFG. ONE_TRIG = 1.
[18]	GATE_EN	Timer pause enabled

		0: No pause 1: When the external signal is low, the Timer stops counting, and the external signal is selected according to the TIMER2_EVT. EVT_SRC.
[17]	RL_EN	Timer reload enabled 0: External event reload is disabled. Timer counts up to TH and returns to 0, or counts down to 0 and returns to TH. 1: external event reloading is enabled, and the reloading signal is selected according to the_EVT. EVT_SRC of TIMER2. When counting up, an external event occurs, and the Timer is reloaded to 0; When counting down, an external event occurs and the Timer is reloaded to TH.
[16]	XCLK_EN	Timer clock source 0: Chip internal clock 1: external clock. The clock source is selected according to the TIMER2_EVT. EVT_SRC. When the external clock is used, the CLK_DIV no longer works, that is, the clock is no longer divided.
[15:12]	SRC1	Timer capture mode channel 1 signal source. The default is 3'h1. 0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[11]	CH1_POL	Output polarity control of Timer channel 1 in comparison mode, output value when counter CNT < CMP1.
[10]	CH1_MODE	Operation mode selection for Timer channel 1. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 1 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 1 input signal, the counter count is stored in the Timer Channel 1 Compare Capture Register
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Timer capture mode channel 0 signal source. The default is 3'h0. 0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0

		7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[3]	CH0_POL	Output polarity control of Timer channel 0 in comparison mode: output value when counter CNT < CMP0.
[2]	CH0_MODE	The working mode selection of Timer channel 0. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 0 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 0 input signal, the counter count is stored in the Timer Channel 0 Compare Capture Register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

13.3.4.2 TIMER2_TH Timer2 threshold register

Address: 0x4001 0804

Reset value: 0x0

Table 13-31 Timer2 threshold register TIMER2_TH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TH															
RW															
0															

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

Location	Bit name	Description
[31:0]	TH	Timer counter count threshold. Count up counts from 0 to the value of TH and back to 0, or count down counts from TH to 0 and back to TH

13.3.4.3 TIMER2_CNT Timer2 count register

Address: 0x4001 0808

Reset value: 0x0

Table 13-32 Timer2 count register TIMER2_CNT

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

0

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

Location	Bit name	Description
[31:0]	CNT	Timer2 counter current count value. A write operation can write a new count value.

Note that the Timer2 clock needs to be turned on by the SYS_CLK_FEN. TIMER2_CLK_EN before writing to TIMER2_CNT.

13.3.4.4 TIMER2_CMP0 Timer2 Channel 0 Compare Capture Register

Address: 0x4001 080C

Reset value: 0x0

Table 13-33 Timer2 Channel 0 Compare Capture Register TIMER2_CMP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP0															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Location	Bit name	Description
[31:0]	CMP0	When Time channel 0 operates in compare mode, a compare event occurs when the counter value equals CMP0. When Timer Channel 0 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP0 register.

Set CMP0 = 0 to make channel 0 constant! CH0_POL. Set CMP0 = TH + 1 to make channel 0 CH0_POL.

13.3.4.5 TIMER2_CMP1 Timer2 Channel 1 Compare Capture Register

Address: 0x4001 0810

Reset value: 0x0

Table 13-34 Timer2 channel 1 compare capture register TIMER2_CMP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															

RW
0

Location	Bit name	Description
[31:0]	CMP1	When Timer channel 1 operates in compare mode, a compare event occurs when the counter value equals CMP1. When Timer Channel 1 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP1 register.

Setting CMP 1 = 0 makes the channel constant! CH1_POL. Set CMP1 = TH + 1 to make the channel CH1_POL.

13.3.4.6 TIMER2_EVT Timer2 external event select register

Address: 0x4001 0814

Reset value: 0x0

Table 13-35 Timer2 external event select register TIMER2_EVT

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

Location	Bit name	Description
[31:3]		Not used
[3:0]	EVT_SRC	Timer external event selection register. This register needs to be used in conjunction with TIMER2_CFG. ETON, TIMER2_CFG. GATE_EN, TIMER2_CFG. RL_EN, and TIMER2_CFG. XCLK_EN. Normally, the 4 set bits are not used at the same time. When ETON = 1 is set, this register selects the event that triggers the Timer count. It should be noted that the Timer's own compare event cannot trigger the Timer to count. When ETON = 1, the rising edge of the trigger signal triggers the Timer to start counting. The Timer used as the trigger source needs to work in the comparison mode without inputting a signal from the outside to the Timer channel. 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: Not available 5: Not available 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output rising edge 9: CLU1 output rising edge 10: CLU2 output rising ed 11: CLU3 output rising ed 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event

		14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event The Timer counts when the GATE_EN = 1 and the external signal is high, the Timer pauses when the external signal is low, and the Timer uses the level signal corresponding to the external signal as a count gate when the GATE_EN = 1. The GATE_EN can only be used in conjunction with external events from 0 to 11. When the EVT_SRC is selected as 0 to 7, the external signal is the input signal of the Timer channel after being filtered by the corresponding Timer. The Timer channel used as the gating signal needs to be configured as an input enable. 0: TIMER0 channel 0 filtered input signal 1: TIMER0 channel 1 filtered input signal 2: TIMER1 channel 0 filtered input signal 3: TIMER1 Channel 1 filtered input signal 4: TIMER2 channel 0 filtered input signal 5: TIMER2 Channel 1 filtered input signal 6: TIMER3 channel 0 filtered input signal 7: TIMER3 Channel 1 filtered input signal 8: CLU0 output signal 9: CLU1 output signal 10: CLU2 output signal 11: CLU3 output signal When RL_EN is high, the Timer uses the external signal as the reload signal Timer channels that are reloaded signals need to be configured as input enabled 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event When the XCLK_EN is high, the Timer uses an external signal as the external clock of the Timer, and the Timer counts on the rising edge of the clock. If a Timer channel is used as an external clock, the channel signal is affected by the corresponding Timer filter setting. 0: TIMER0 channel 0 input signal 1: TIMER0 channel 1 input signal 2: TIMER1 channel 0 input signal 3: TIMER1 channel 1 input signal 4: TIMER2 channel 0 input signal 5: TIMER2 channel 1 input signal 6: TIMER3 channel 0 input signal

		7: TIMER3 channel 1 input signal 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event

13.3.4.7 TIMER2_FLT Timer2 filter control register

Address: 0x4001 0818

Reset value: 0x0

Table 13-36 Timer2 filter control register TIMER2_FLT

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

Location	Bit name	Description
[31:8]		Not used
[7:0]	FLT	Channel 0/1 signal filter width selection. Value range: 0 ~ 255. When FLT is 0, the channel is not filtered. When FLT is not 0, filter the channel signal: the filter width is FLT × 8. When the channel signal level is stable and exceeds the width of FLT * 8 system clock cycles, the filter output is updated; Otherwise, the filter keeps the current output unchanged.

Note that the working clock of the above filter is the same as the working clock of the corresponding Timer after frequency division, and is controlled by the frequency division factor of the TIMERx_CFG. CLK_DIV.

Assume TIMERx_FLT. FLT = 0x6; TIMERx_CFG.CLK_DIV = 0x2; The running clock of the Timer is divided by 4 with respect to the system clock. The channel input signal needs to be filtered by an 8 × 6 Timer running clock, that is, 8 × 6 × 4 system clock.

13.3.4.8 TIMER2_IE Timer2 Interrupt Enable Register

Address: 0x4001 081C

Reset value: 0x0

Table 13-37 Timer2 Interrupt Enable Register TIMER2_IE

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

	ZC_RE	CH1_RE	CH0_RE		ZC_IE	CH1_IE	CH0_IE
	RW	RW	RW		RW	RW	RW
	0	0	0		0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	ZC_RE	Timer counter 0 DMA request enabled, active high.
[9]	CH1_RE	Timer channel 1 compare/capture DMA request enable, active high.
[8]	CH0_RE	Timer Channel 0 Compare/Capture DMA Request Enable, Active High.
[7:3]		Not used
[2]	ZC_IE	Timer counter over-zero interrupt enabled, active high.
[1]	CH1_IE	Timer Channel 1 Compare/Capture Interrupt Enable, Active High.
[0]	CH0_IE	Timer Channel 0 Compare/Capture Interrupt Enabled, Active High.

The DMA request event is the same event flag as the interrupt. After the DMA request is accepted by the DMA, the interrupt flag will be automatically cleared by the DMA hardware without CPU software intervention. It is important to note that a DMA request that turns on an event typically turns off the corresponding interrupt enable.

13.3.4.9 TIMER2_IF Timer2 Interrupt Flag register

Address: 0x4001 0820

Reset value: 0x0

Table 13-38 Timer2 interrupt flag register TIMER2_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Location	Bit name	Description
[31:3]		Not used
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear 0
[1]	CH1_IF	Timer Channel 1 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0
[0]	CH0_IF	Timer Channel 0 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0

In compare mode, when Timer is set to center count (0- > TH- > 0), that is, TIMERx_CFG. CENTER

= 1. CH0_IF and CH1_IF are set only when the TIMERx_CNT = TIMERx_CMP0/1 while the Timer is counting up from 0 to TH; The bit is not set while the Timer is counting down from TH to 0.

Write 1 to clear the interrupt flag. It is generally not recommended to use the following | = method to clear, because | = reads the interrupt flag first, changes the corresponding bit to 1, and then writes it to clear. If other interrupt flags are set at the same time, they will be cleared together, which is usually not expected by the software. For example, the following writing method is intended to clear the ZC_IF, but if CH0_if is set to 1 before writing, the software first reads back the TIMERx_IF value as 0x24, then executes the or operation 0x4 | 0x1 = 0x5, and then writes, clearing both CH0_if and the ZC_IF. May cause the Timer to enter one less interrupt due to capture.

TIMERx_IF|=0x4;

If you want to clear the T0_ZC_IF flag, write a 1. Directly to BIT2 as follows.

TIMERx_IF=0x4;

13.3.5 TIMER3 register

13.3.5.1 TIMER3_CFG Timer3 configuration register

Address: 0x4001 0900

Reset value: 0x0

Table 13-39 Timer3 configuration register TIMER3_CFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN		CAP1_CLR_EN	CAP0_CLR_EN			ONE_TRIG	—	—	CLK_DIV			—	—	RL_EN	XCLK_EN
RW		RW	RW			RW	RW	RW	RW			RW	RW	RW	RW
0		0	0			0	0	0	0			0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRC1				CH1_POL	CH1_MODE	CH1_FE_CAP_EN	CH1_RE_CAP_EN	SRC0				CH0_POL	—	CH0_FE_CAP_EN	CH0_RE_CAP_EN
RW				RW	RW	RW	RW	RW				RW	RW	RW	RW
0001				0	0	0	0	0				0	0	0	0

Location	Bit name	Description
[31]	EN	Timer block overall enable, high effective
[30]		Not used
[29]	CAP1_CLR_EN	When a CAP1 capture event occurs, the Timer counter is cleared, active high
[28]	CAP0_CLR_EN	When a CAP0 capture event occurs, the Timer counter is cleared, active high

[27:26]		Not used
[25]	ONE_TRIG	Single transmit mode, this bit needs to be used in Timer compare mode and EN needs to be set to 0 When ETON is 0, software writes a 1 to the ONE_TRIG to trigger the Timer to send a pulse with a specific duty cycle for one cycle, and the Timer stops counting after one cycle. This bit is read as 1 during the current cycle of the Timer count and is automatically cleared when the Timer is stopped. When ETON is 1, even if the software configuration ONE_TRIG = 1, it is necessary to wait for an external event to trigger, and the Timer stops after counting one cycle. The ONE_TRIG bit is cleared by hardware after the externally triggered Timer stops counting one cycle.
[24]	CENTER	Center count mode enabled 0: Timer counts up from 0 to TH and back to 0, or Timer counts down from TH to 0 and back to TH 1: Timer counts up from 0 to TH, then down to 0
[23]	DIR	0:0-> TH count up, 1: TH-> count down This bit is read only when CENTER = 1 and is used to indicate the current count direction
[22:20]	CLK_DIV	Timer counter frequency configuration. The counter count frequency is divided by $2^{CLK_DIV}$of the system master clock frequency. The default value is 0 and does not divide. 0:1 frequency division 1:2 frequency division 2:4 frequency division 3:8 frequency division 4:16 frequency division 5:32 frequency division 6:64 frequency division 7:128 frequency division
[19]	ETON	Timer counter external start enabled 0: Run automatically 1: Wait for external event trigger counting, and external trigger signal is selected according to TIMER3_EVT. EVT_SRC. Note that using an external trigger also requires setting TIMER3_CFG. EN = 1, unless it is a single trigger from an external signal, i.e. TIMER3_CFG. ONE_TRIG = 1.
[18]	GATE_EN	Timer pause enabled 0: No pause 1: When the external signal is low, the Timer stops counting, and the external signal is selected according to the TIMER3_EVT. EVT_SRC.
[17]	RL_EN	Timer reload enabled 0: External event reload is disabled. Timer counts up to TH and returns to 0, or counts down to 0 and returns to TH. 1: External event reloading is enabled, and the reloading signal is selected according to the TIMER3_EVT. EVT_SRC. When counting up, an external event occurs, and the Timer is reloaded to 0. When counting down, an external event occurs and the Timer is reloaded to TH.
[16]	XCLK_EN	Timer clock source 0: Chip internal clock 1: external clock. The clock source is selected according to the TIMER3_EVT. EVT_SRC. When the external clock is used, the CLK_DIV no longer works, that is, the clock is no longer divided.
[15:12]	SRC1	Timer capture mode channel 1 signal source. The default is 3'h1.

		0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[11]	CH1_POL	Output polarity control of Timer channel 1 in comparison mode, output value when counter CNT < CMP1.
[10]	CH1_MODE	Operation mode selection for Timer channel 1. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 1 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 1 input signal, the counter count is stored in the Timer Channel 1 Compare Capture Register
[9]	CH1_FE_CAP_EN	Channel 1 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 1 input signal is considered a capture event. Falling edge event enable can coexist with rising edge event enable.
[8]	CH1_RE_CAP_EN	Channel 1 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 1 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.
[7:4]	SRC0	Timer capture mode channel 0 signal source. The default is 3'h0. 0: Timer channel 0, from chip GPIO (see Data sheet and application configuration) 1: Timer channel 1, from chip GPIO (see Data sheet and application configuration) 2: CLU0 output 3: CLU1 output 4: CLU2 output 5: CLU3 output 6: Output of comparator 0 7: Output of comparator 1 8: Output of comparator 2 9: Xor of Timer channels 0 and 1
[3]	CH0_POL	Output polarity control of Timer channel 0 in comparison mode: output value when counter CNT < CMP0.
[2]	CH0_MODE	The working mode selection of Timer channel 0. The default value is 0. 0: Comparison mode. Outputs a square wave, the IO is flipped when the Timer Channel 0 counter counts to 0 or to the Timer Compare Capture register value. 1: Capture mode. When a capture event occurs on the Timer Channel 0 input signal, the counter count is stored in the Timer Channel 0 Compare Capture Register.
[1]	CH0_FE_CAP_EN	Channel 0 falling edge capture event enable. 1: enable; 0: disable. A 1→0 transition on the channel 0 input signal is considered a capture event. Falling edge event enable can coexist with rising edge

		event enable.
[0]	CH0_RE_CAP_EN	Channel 0 rising edge capture event enable. 1: enable; 0: disable. A 0→1 transition on the channel 0 input signal is considered a capture event. Rising edge event enable can coexist with falling edge event enable.

13.3.5.2 TIMER3_TH Timer3 threshold register

Address: 0x400 1_0904

Reset value: 0x0

Table13-40 Timer3 threshold register TIMER3_TH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TH															
RW															
0															

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

Location	Bit name	Description
[31:0]	TH	Timer counter count threshold. Count up counts from 0 to the value of TH and back to 0, or count down counts from TH to 0 and back to TH

13.3.5.3 TIMER3_CNT Timer3 count register

Address: 0x4001 0908

Reset value: 0x0

Table13-41 Timer3 count register TIMER3_CNT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT															
RW															
0															

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

Location	Bit name	Description
[31:0]	CNT	Timer3 counter current count value. A write operation can write a new count value.

Note that the Timer3 clock needs to be turned on by the SYS_CLK_FEN. TIMER3_CLK_EN before writing to TIMER3_CNT.

13.3.5.4 TIMER3_CMP0 Timer3 Channel 0 Compare Capture Register

Address: 0x4001 090C

Reset value: 0x0

Table13-42 Timer3 Channel 0 Compare Capture Register TIMER3_CMP0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP0															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0															
RW															
0															

Location	Bit name	Description
[31:0]	CMP0	When Time channel 0 operates in compare mode, a compare event occurs when the counter value equals CMP0. When Timer Channel 0 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP0 register.

Set CMP0 = 0 to make channel 0 constant! CH0_POL. Set CMP0 = TH + 1 to make channel 0 CH0_POL.

13.3.5.5 TIMER3_CMP1 Timer3 Channel 1 Compare Capture Register

Address: 0x4001 0910

Reset value: 0x0

Table13-43 Timer3 channel 1 compare capture register TIMER3_CMP1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1															
RW															
0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP1															
RW															
0															

Location	Bit name	Description
[31:0]	CMP1	When Timer channel 1 operates in compare mode, a compare event occurs when the counter value equals CMP1. When Timer Channel 1 is operating in capture mode, the counter value at the time of the capture event is stored in the CMP1 register.

Setting CMP 1 = 0 makes the channel constant! CH1_POL. Set CMP1 = TH + 1 to make the channel CH1_POL.

13.3.5.6 TIMER3_EVT Timer3 external event select register

Address: 0x4001 0914

Reset value: 0x0

Table13-44 Timer3 external event select register TIMER3_EVT

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

Location	Bit name	Description
[31:3]		Not used
[3:0]	EVT_SRC	Timer external event selection register. This register needs to be used in conjunction with TIMER3_CFG. ETON, TIMER3_CFG. GATE_EN, TIMER3_CFG. RL_EN, and TIMER3_CFG. XCLK_EN. Normally, the 4 set bits are not used at the same time. When ETON = 1 is set, this register selects the event that triggers the Timer count. It should be noted that the Timer's own compare event cannot trigger the Timer to count. When ETON = 1, the rising edge of the trigger signal triggers the Timer to start counting. The Timer used as the trigger source needs to work in the comparison mode without inputting a signal from the outside to the Timer channel. 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: Not available 7: Not available 8: CLU0 output rising edge 9: CLU1 output rising edge 10: CLU2 output rising ed 11: CLU3 output rising ed 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event The Timer counts when the GATE_EN = 1 and the external signal is high, the Timer pauses when the external signal is low, and the Timer uses the level signal corresponding to the external signal as a count gate when the GATE_EN = 1. The GATE_EN can only be used in conjunction with external events from 0 to 11. When the EVT_SRC is selected as 0 to 7, the external signal is the input signal of the Timer channel after being filtered by the corresponding Timer. The Timer channel used as the gating signal needs to be configured as an input enable. 0: TIMER0 channel 0 filtered input signal 1: TIMER0 channel 1 filtered input signal 2: TIMER1 channel 0 filtered input signal 3: TIMER1 Channel 1 filtered input signal 4: TIMER2 channel 0 filtered input signal 5: TIMER2 Channel 1 filtered input signal

		6: TIMER3 channel 0 filtered input signal 7: TIMER3 Channel 1 filtered input signal 8: CLU0 output signal 9: CLU1 output signal 10: CLU2 output signal 11: CLU3 output signal When RL_EN is high, the Timer uses the external signal as the reload signal Timer channels that are reloaded signals need to be configured as input enabled 0: TIMER0 Channel 0 Compare/Capture Event 1: TIMER0 Channel 1 Compare/Capture Event 2: TIMER1 Channel 0 Compare/Capture Event 3: TIMER1 Channel 1 Compare/Capture Event 4: TIMER2 Channel 0 Compare/Capture Event 5: TIMER2 Channel 1 Compare/Capture Event 6: TIMER3 Channel 0 Compare/Capture Event 7: TIMER3 Channel 1 Compare/Capture Event 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event When the XCLK_EN is high, the Timer uses an external signal as the external clock of the Timer, and the Timer counts on the rising edge of the clock. If a Timer channel is used as an external clock, the channel signal is affected by the corresponding Timer filter setting. 0: TIMER0 channel 0 input signal 1: TIMER0 channel 1 input signal 2: TIMER1 channel 0 input signal 3: TIMER1 channel 1 input signal 4: TIMER2 channel 0 input signal 5: TIMER2 channel 1 input signal 6: TIMER3 channel 0 input signal 7: TIMER3 channel 1 input signal 8: CLU0 output 9: CLU1 output 10: CLU2 output 11: CLU3 output 12: MCPWM TADC [0] Compare Event 13: MCPWM TADC [1] Compare Event 14: MCPWM TADC [2] Compare Event 15: MCPWM TADC [3] Compare Event

13.3.5.7 TIMER3_FLT Timer3 filter control register

Address: 0x4001 0918

Reset value: 0x0

Table 13-45 Timer3 filter control register TIMER3_FLT

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

Location	Bit name	Description
[31:8]		Not used
[7:0]	FLT	Channel 0/1 signal filter width selection. Value range: 0 ~ 255. When FLT is 0, the channel is not filtered. When FLT is not 0, filter the channel signal: the filter width is FLT × 8. When the channel signal level is stable and exceeds the width of FLT * 8 system clock cycles, the filter output is updated; Otherwise, the filter keeps the current output unchanged.

Note that the working clock of the above filter is the same as the working clock of the corresponding Timer after frequency division, and is controlled by the frequency division factor of the TIMERx_CFG. CLK_DIV.

Assume TIMERx_FLT. FLT = 0x6; TIMERx_CFG.CLK_DIV = 0x2; The running clock of the Timer is divided by 4 with respect to the system clock. The channel input signal needs to be filtered by an 8 × 6 Timer running clock, that is, 8 × 6 × 4 system clock.

13.3.5.8 TIMER3_IE Timer3 Interrupt Enable Register

Address: 0x4001 091C

Reset value: 0x0

Table 13-46 Timer3 Interrupt Enable Register TIMER3_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ZC_RE	CH1_RE	CH0_RE						ZC_IE	CH1_IE	CH0_IE
					RW	RW	RW						RW	RW	RW
					0	0	0						0	0	0

Location	Bit name	Description
[31:11]		Not used
[10]	ZC_RE	Timer counter 0 DMA request enabled, active high.
[9]	CH1_RE	Timer channel 1 compare/capture DMA request enable, active high.
[8]	CH0_RE	Timer channel 0 compare/capture DMA request enable, active high.
[7:3]		Not used
[2]	ZC_IE	Timer counter over-zero interrupt enabled, active high.
[1]	CH1_IE	Timer channel 1 compare/capture interrupt enable, active high.
[0]	CH0_IE	Timer channel 0 compare/capture interrupt enabled, active high.

The DMA request event is the same event flag as the interrupt. After the DMA request is accepted by the DMA, the interrupt flag will be automatically cleared by the DMA hardware without CPU software intervention. It is important to note that a DMA request that turns on an event typically turns off the corresponding interrupt enable.

13.3.5.9 TIMER3_IF Timer3 Interrupt Flag register

Address: 0x4001 0920

Reset value: 0x0

Table 13-47 Timer3 interrupt flag register TIMER3_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ZC_IF	CH1_IF	CH0_IF
													RW1C	RW1C	RW1C
													0	0	0

Location	Bit name	Description
[31:3]		Not used
[2]	ZC_IF	Timer counter over-zero interrupt flag. Active high, write 1 to clear 0
[1]	CH1_IF	Timer Channel 1 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0
[0]	CH0_IF	Timer Channel 0 Compare/Capture Interrupt Flag. Active high, write 1 to clear 0

In compare mode, when Timer is set to center count (0- > TH- > 0), that is, TIMERx_CFG. CENTER = 1. CH0_IF and CH1_IF are set only when the TIMERx_CNT = TIMERx_CMP0/1 while the Timer is counting up from 0 to TH; The bit is not set while the Timer is counting down from TH to 0.

Write 1 to clear the interrupt flag. It is generally not recommended to use the following | = method to clear, because | = reads the interrupt flag first, changes the corresponding bit to 1, and then writes it to clear. If other interrupt flags are set at the same time, they will be cleared together, which is usually not expected by the software. For example, the following writing method is intended to clear the ZC_IF, but if CH0_if is set to 1 before writing, the software first reads back the TIMERx_IF value as 0x24, then executes the or operation 0x4 | 0x1 = 0x5, and then writes, clearing both CH0_if and the ZC_IF. May cause the Timer to enter one less interrupt due to capture.

TIMERx_IF|=0x4;

If you want to clear the T0_ZC_IF flag, write a 1. Directly to BIT2 as follows.

TIMERx_IF=0x4;

13.3.6 QEP0 register

13.3.6.1 QEP0_CFG QEP0 Configuration Register

QEP 0_CFG Address: 0x4001_0A00

Reset value: 0x0

Table 13-48 QEP0 configuration register QEP0_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

Location	Bit name	Description
[31:16]		Not used
[15]		Enabling of whole QEP module
[14:11]		Not used
[10]	FE_CNT_EN	Whether to count on the falling edge in CCW + SIGN/CCW + CW mode (the rising edge is always counted)
[9:8]	MODE	Encoder mode selection 00:counting on T1 , 01:counting on T1 & T2 Both of the above modes are the quadrature encoded signal counting modes 10: CCW + SIGN, sign plus pulse signal counting mode 11: CCW + CW, CCW + CW double-pulse signal counting mode
[7:4]		Not used
[3]	ZNC	Z Signal Clear Polarity Selection: Low/Falling Edge Clear Enable
[2]	ZPC	Z signal clear polarity selection: high/Rising edge clear enabled
[1]	ZLC	Z signal level clears QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC = 1, the jumping edge of the Z signal is used to clear the QEP counter; when ZNC is 1 at the same time, the falling edge of the Z signal clears the QEP counter; when ZPC is 1 at the same time, the rising edge of Z signal clears the QEP count; ZNC and ZPC can be 1 at the same time, then the QEP counter is cleared when the Z signal changes, and the reset is released immediately after clearing, and the QEP counter is ready for subsequent counting.

When ZLC = 1, the active level of the Z signal is used to clear the QEP count; when ZNC is 1 at the same time, the low level of the Z signal clears the QEP counter; when ZPC is 1 at the same time, the high level of Z signal clears the QEP counter; In general, when the active level of the Z signal is used to clear the QEP counter, ZNC and ZPC will not be 1 at the same time, otherwise the counter will always be cleared. When the active level is used to clear zero, the QEP counter resumes counting only after the Z signal has flipped.

13.3.6.2 QEP0_TH QEP0 Count Threshold Register

QEP 0 TH Address: 0x4001 0A04

Reset value: 0x0

Table 13-49 QEP0 count threshold register QEP0_TH

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	TH	Counting threshold TH. After the encoder has counted up (incremented) to the value of th, counting up again causes the counter to return to 0. After the encoder has counted down (decremented) to a value of 0, counting down again causes the counter to return to TH.

13.3.6.3 QEP0_CNT QEP0 count register

QEP 0 CNT Address: 0x4001 0A08

Reset value: 0x0

Table 13-50 QEP0 count register QEP0_CNT

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	CNT	QEP0 count value.

13.3.6.4 QEP0_IE QEP0 Interrupt Enable Register

Address: 0x4001 0 A0C

Reset value: 0x0

Table 13-51 QEP0 Interrupt Enable Register QEP0_IE

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

Location	Bit name	Description
[31:2]		Not used
[1]	OF_IE	Overflow interrupt enabled, active high.
[0]	UF_IE	Underflow interrupt enabled, active high.

13.3.6.5 QEP0_IF QEP0 Interrupt Flag register

Address: 0x4001 0 A10

Reset value: 0x0

Table 13-52 QEP0 interrupt flag register QEP0_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														OF_IF	UF_IF
														RW1C	RW1C
														0	0

Location	Bit name	Description
[31:2]		Not used
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the counting threshold, the up-counting event triggers the up-overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches 0, the down-count event triggers the underflow interrupt.

13.3.7 QEP1 register

13.3.7.1 QEP1_CFG QEP1 Configuration Register

QEP 1_CFG Address: 0x4001_0B00

Reset value: 0x0

Table 13-53 QEP1 configuration register QEP1_CFG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN					FE_CNT_EN	MODE						ZNC	ZPC	ZLC	ZEC
RW					RW	RW						RW	RW	RW	RW
0					0	0						0	0	0	0

Location	Bit name	Description

[31:11]		Not used
[10]	FE_CNT_EN	Whether to count on the falling edge in CCW + SIGN/CCW + CW mode (the rising edge is always counted)
[9:8]	MODE	Encoder mode selection 00:counting on T1 , 01:counting on T1 & T2 Both of the above modes are the quadrature encoded signal counting modes 10: CCW + SIGN, sign plus pulse signal counting mode 11: CCW + CW, CCW + CW double-pulse signal counting mode
[7:4]		Not used
[3]	ZNC	Z Signal Clear Polarity Selection: Low/Falling Edge Clear Enable
[2]	ZPC	Z signal clear polarity selection: high/rising edge clear enabled
[1]	ZLC	Z signal level clears QEP counter enable
[0]	ZEC	Z signal edge clear QEP counter enable

When ZEC = 1, the jumping edge of the Z signal is used to clear the QEP counter; when ZNC is 1 at the same time, the falling edge of the Z signal clears the QEP counter; when ZPC is 1 at the same time, the rising edge of Z signal clears the QEP count; ZNC and ZPC can be 1 at the same time, then the QEP counter is cleared when the Z signal changes, and the reset is released immediately after clearing, and the QEP counter is ready for subsequent counting.

When ZLC = 1, the active level of the Z signal is used to clear the QEP count; when ZNC is 1 at the same time, the low level of the Z signal clears the QEP counter; when ZPC is 1 at the same time, the high level of Z signal clears the QEP counter; In general, when the active level of the Z signal is used to clear the QEP counter, ZNC and ZPC will not be 1 at the same time, otherwise the counter will always be cleared. When the active level is used to clear zero, the QEP counter resumes counting only after the Z signal has flipped.

13.3.7.2 QEP 1_TH QEP1 Count Threshold Register

QEP 1_TH Address: 0x4001_0B04

Reset value: 0x0

Table13-54 QEP1 count threshold register QEP1_TH

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	TH	QEP counting threshold TH. After the encoder has counted up (incremented) to the value of th, counting up again causes the counter to return to 0. After the encoder has counted down (decremented) to a value of 0, counting down again causes the counter to return to TH.

13.3.7.3 QEP1_CNT QEP1 Count Register

QEP 1_CNT Address: 0x4001_0B08

Reset value: 0x0

Table 13-55 QEP1 count register QEP1_CNT

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

Location	Bit name	Description
[31:16]		Not used
[15:0]	CNT	QEP count value.

13.3.7.4 QEP1_IE QEP1 Interrupt Enable Register

Address: 0x4001 0 B0C

Reset value: 0x0

Table 13-56 QEP1 Interrupt Enable Register QEP1_IE

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

Location	Bit name	Description
[31:2]		Not used
[1]	OF_IE	Overflow interrupt enabled, active high.
[0]	UF_IE	Underflow interrupt enabled, active high.

13.3.7.5 QEP1_IF QEP1 Interrupt Flag register

Address: 0x4001 0B10

Reset value: 0x0

Table 13-57 QEP1 interrupt flag register QEP1_IF

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

	RW1C	RW1C
	0	0

Location	Bit name	Description
[31:2]		Not used
[1]	OF_IF	Overflow interrupt flag. Active high, write 1 to clear. When the counter reaches the counting threshold, the up-counting event triggers the up-overflow interrupt.
[0]	UF_IF	Underflow interrupt flag. Active high, write 1 to clear. When the counter reaches 0, the down-count event triggers the underflow interrupt.

14 MCPWM

14.1 Overview

The MCPWM module is a module that precisely controls the output of the motor drive waveform.

Two 16-bit up counters are included to provide two counting cycles (hereinafter referred to as two time bases). The clock frequency division of the counter has four options of 1/2/4/8, and the generated clock frequency division is 96MHz/48MHz/24MHz/12MHz respectively. Channels 0/1/2 are fixed to time base 0 and channels 3/4/5 are fixed to time base 1.

It contains 6 groups of PWM generation modules.

-Able to produce 6 pairs (complementary signals) or 12 independent (edge mode) non-overlapping PWM signals;

-Edge-aligned PWM support

-Center Align PWM

-Phase-shifted PWM

Besides, it can generate 4 channels of timing information at the same time as MCPWM, which is used to trigger the synchronous sampling of the ADC module for linkage with MCPWM.

It also contains a set of emergency stop protection modules for quickly shutting down the output of the MCPWM module without relying on CPU software processing. The MCPWM module can input four emergency stop signals, two of which come from IO and two from the output of the on-chip comparator. When an emergency stop event occurs (supports effective level polarity selection), disable the corresponding PWM channel based on the MCPWM0_FAIL012/MCPWM0_FAIL345/ MCPWM0_CH_MSK settings to avoid short circuit.

Moreover, there is an independent filter module for the emergency stop signal.

Each output IO of MCPWM supports two control modes: PWM hardware control or software direct control (for EABS soft brake, or BLDC square-wave commutation control).

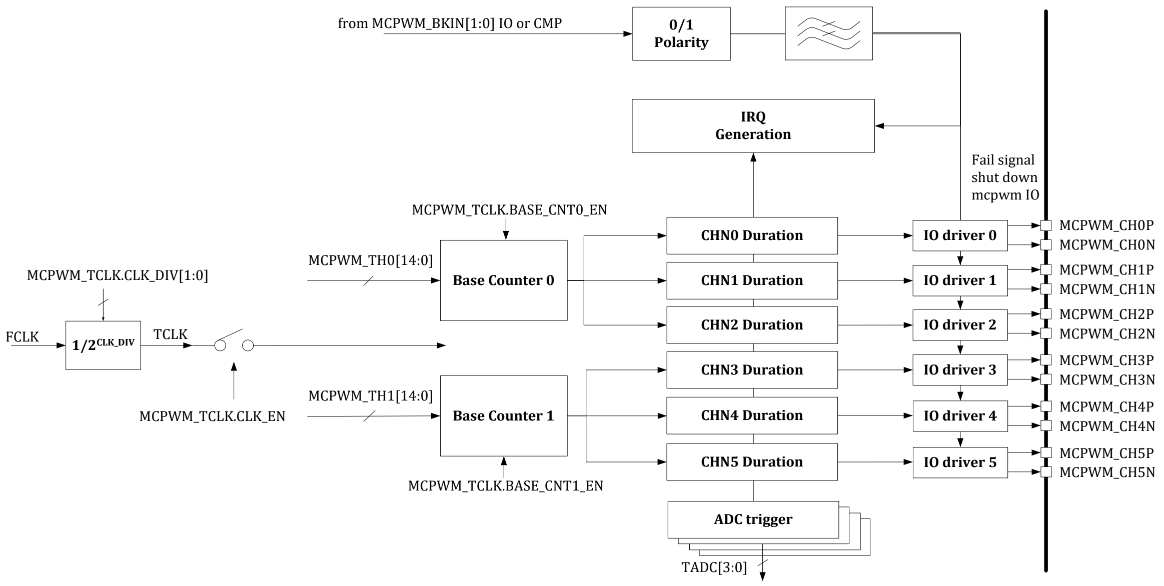


Figure14-1 Block diagram of MCPWM module

Usually, a 96MHz clock is used as the operating frequency of the MCPWM module to ensure the timing accuracy.

14.1.1 Base Counter module

The module is mainly composed of two count-up counter, and the count threshold are MCPWM0_TH0/MCPWM0_TH1. The counter starts at time t0, and counts up from -TH. It passes the time zero at time t1, counts at time t2 to TH to complete a counting cycle, and then returns to -TH to restart counting. The count period is (TH×2+1) times the count clock period.

Timed event interrupt can be generated at t0/t1 (current time t0 is the previous t2), MCPWM0_IFx.T0_IF and the MCPWM0_IFx.T1_IF will be set.

The start and stop of the Base Counter can be controlled by register configuration MCPWM0_TCLK. BASE_CNT0_EN/MCPWM0_TCLK. BASE_CNT1_EN.

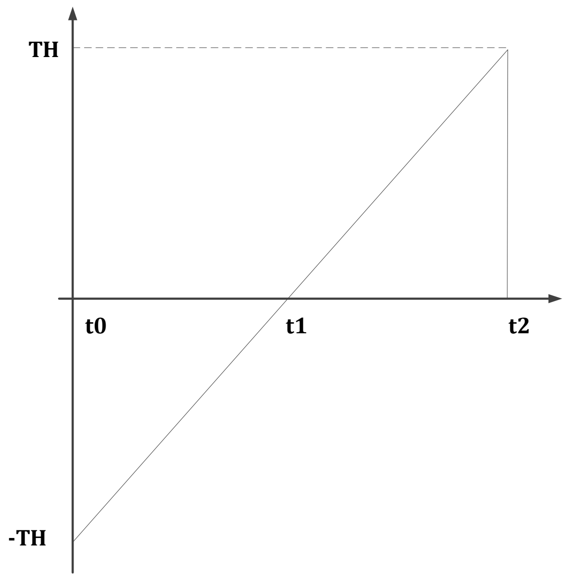


Figure14-2 Base Counter t0/T1 Timing

Before running the MCPWM module, users should set the corresponding comparison thresholds (MCPWM_TH00 ~ MCWM_TH51), dead-zone registers (MCPWM0_DTH00/ MCPWM0_DTH01/ MCPWM0_DTH10/ MCPWM0_DTH11) in advance. In the actual operation process, the comparison threshold value and the PWM period register can also be changed. In the actual running process, the comparison threshold and the dead zone register can also be changed dynamically.The hardware update can only generate update events at time t0 and t1 (update t0 or t1 and update both t0 and t1 at all times), and the hardware loads the value of the load register into the running register. The occurrence frequency of the update event can be set, that is, the update occurs every N time t0 and t1. Regardless of whether an update occurs, a corresponding interrupt can be generated at t0 and t1. If the hardware loads the value of the load register into the running register, a load interrupt is generated.

Update manually by writing to the MCPWM0_UPDATE register, or complete hardware auto-update by setting MCPWM_SDCFG.T1_UPDATE_EN and MCPWM_SDCFG.T0_UPDATE_EN. The hardware update can only generate update events at time t0 and t1 (update t0 or t1 and update both t0 and t1 at all times), and the hardware loads the value of the load register into the running register. The occurrence frequency of the update event can be set, that is, the update occurs every N time t0 and t1. Regardless of whether an update occurs, a corresponding interrupt can be generated at t0 and t1. If the hardware loads the value of the load register into the running register, a load interrupt is generated.

Select whether the update occurs at t0 or t1 or both by setting the MCPWM0_SDCFG register, and set the update interval number as 1 ~ 16. The most frequent update configuration is that updates occur at t0 and t1, which occur continuously. The lowest speed update configuration is that the update occurs at t1, and updates every sixteen t1.

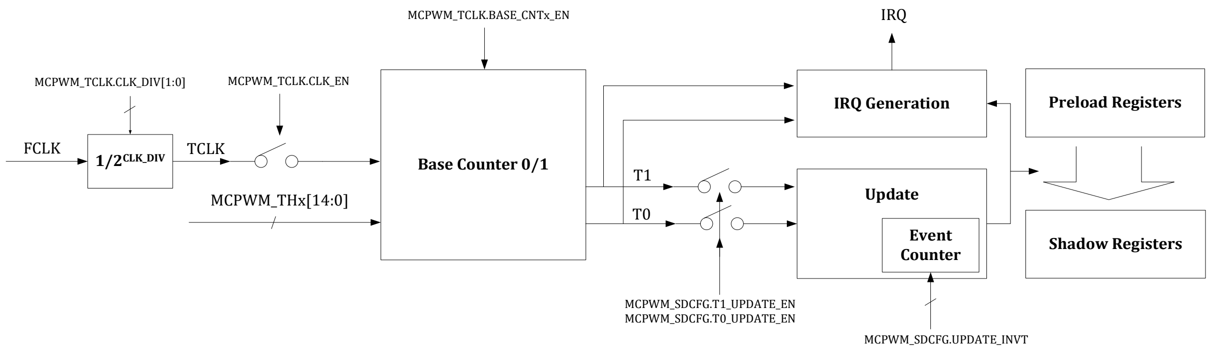


Figure14-3 MCPWM update mechanism

The TH register MCPWM0_THx and the count register MCPWM0_CNTx (X = 0, 1) corresponding to time base 0 and time base 1 have shadow registers and both support manual update (update by software writing the corresponding bit to MCPWm0_UPDATE) and automatic update (update triggered by the occurrence of a specific hardware event). The shadow registers can be updated without turning on the MCPWM TCLK clock (the MCPWM0_TCLK.TCLK_EN= 0). MCPWM0_TCLK.BASE_CNT0_EN and MCPWM0_TCLK.BASE_CNT1_EN can control the count of time base 0 and time base 1, respectively. When MCPWM0_TCLK.BASE_CNT0_EN=0, the CNT of time base 0 remains unchanged after the update, and the CNT of time base 1 is the same with the same implementation.

In addition, to accurately control the count of the first MCPWM cycle, it is recommended that MCPWM0_THx and MCPWM0_CNTx be updated simultaneously. Of course, separate updates of MCPWM0_THx and MCPWM0_CNTx are also supported. The start of counting is then triggered by a software write or an external event, which sets the MCWPM_TCLk. BASE_CNT0_EN/MCWPM_TCLk. BASE_CNT1_EN to enable CNT counting for time base 0 and time base 1, respectively.

14.1.2 FAIL signal processing

The FAIL signal is an emergency stop signal primarily used to quickly shut down power transistors in the event of an anomaly, thereby preventing irreversible hardware damage. This signal processing module configures emergency stop events based on actual conditions to achieve rapid shutdown of PWM output. Two FAIL signal inputs are provided to the MCPWM: FAIL0 and FAIL1, which can originate from either the chip's I0 MCPWM_BKIN[1:0] or the internal comparator's output CMP[1:0]. Among these, the P/N channels 0/1/2 of MCPWM can choose to use CMP[0] or MCPWM_BKIN[0], while the P/N channel 3 of MCPWM can choose to use CMP[1] or MCPWM_BKIN[1].

07x MCU FAIL Signal distribution table

	MCPWM CH0/1/2		MCPWM CH3/4/5	
	Fail0	Fail1	Fail2	Fail3
MCPWM_BKIN0	√		√	
MCPWM_BKIN1		√		√
CMP0	√		√	

CMP1		√		√
CLUOUT0	√		√	
CLUOUT1	√		√	
CLUOUT2		√		√
CLUOUT3		√		√

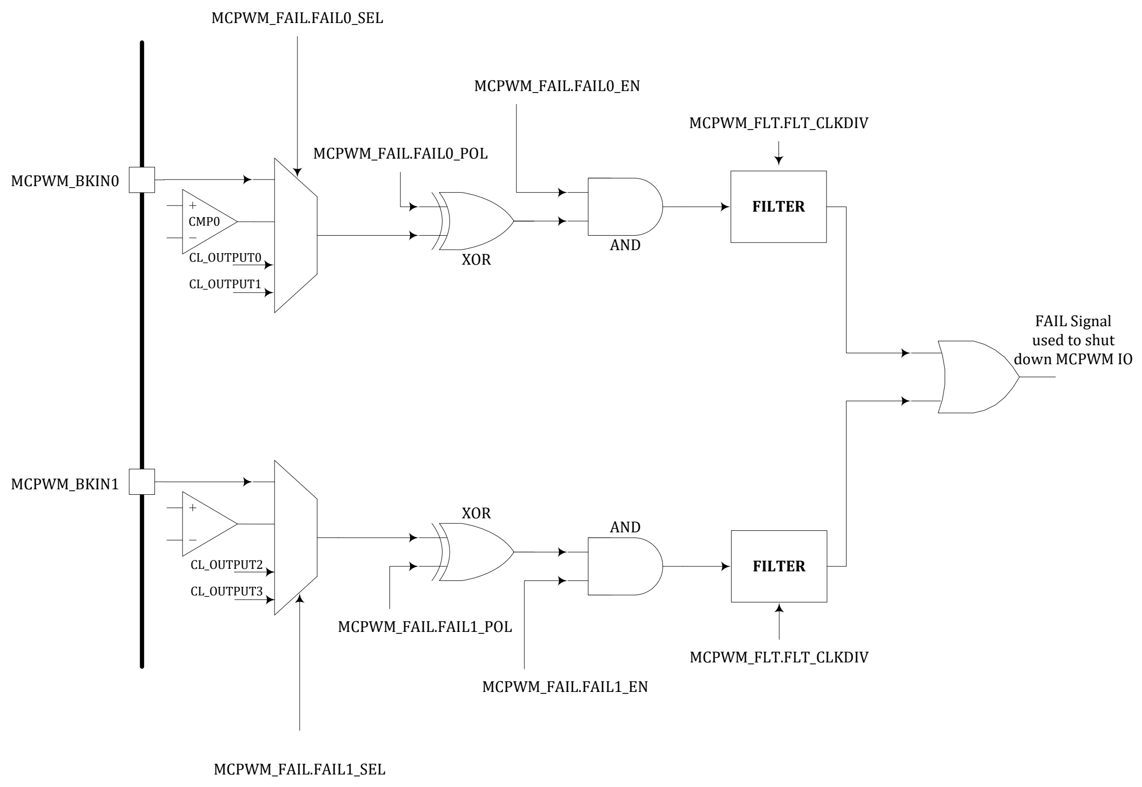


Figure14-4 MCPWM FAIL Logic Diagram

Filter is the clock of the filter module, which comes from the gated clock FCLK of the system master clock MCLK, as shown in the SYS_CLK_FEN, and is divided by two optional stages. The first stage is controlled by the MCPWM0_TCLK.CLK_DIV to divide by 1/2/4/8. The second stage of frequency division can realize the frequency division of 1 to 256 times. The MCPWM0_FLT.FLT_CLKDIV[7:0] is used as the division factor of the second stage,as shown inFigure 14-5.

The MCPWM module uses the divided clock to filter the Fail signal, and the filter width is fixed to 16 cycles, that is, the input signal must remain stable for at least 16 divided clock cycles (the clock after two stages of division) before the hardware determines it as a valid input signal. The equation for the filter time constant is, where $T_{MCLK}$is the clock period of MCLK/FCLK [4], and 96 MHz corresponds to 10.42 ns.

$$T = T_{MCLK}\times(MCPWM0\_TCLK.CLK\_DIV)\times(FLT\_CLKDIV+1)\times16$$

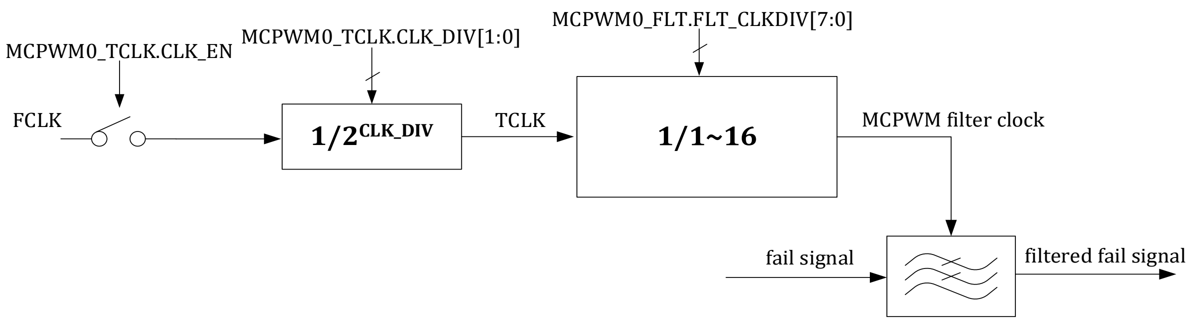


Figure 14-6 MCPWM Fail Signal Filter Clock Generation Logic

When a FAIL event occurs, the hardware forces the IO outputs to the fault defaults specified by the MCPWM0_FAILx.CHxN_DEFAULT and MCPWM0_FAILx.CHxP_DEFAULT registers. At this time, the values of the MCPWM0_FAILx.CHxN_DEFAULT and the MCPWM0_FAILx.CHxP_DEFAULT are directly output to the IO port, and are no longer affected by the polarity control such as the MCPWM0_FAILx. FAIL_POL. Where MCPWM0_FAIL012 is used to control the fault defaults for channels 0/1/2, and MCPWM0_FAIL345 is used to control the fault defaults for channels 3/4/5.

The MCPWM FAIL signal from the comparator is the original signal output by the analog comparator, which is not filtered by the comparator digital interface module, but are not controlled by the comparator filter, see the comparator digital interface module for windowing control settings. After the FAIL signal enters the MCPWM block, it can be filtered by setting the MCPWM_TCLK.

14.1.3 MCPWM Special Output Status

The all-0 and all-1 output States are often used in motor control, and the following complementary mode settings can produce the desired output.

1. If THn0≥THn1, the chip is in a constant 0 state (CH <n> P off, CH <n> N on), no dead-zone
2. If THn0 = -TH, THn1 = TH, the chip is in a constant 1 state (CH <n> P is on, CH <n> P is off), no dead-zone

14.1.4 IO DRIVER module

This module sets IO to the corresponding level according to the actual MCPWM register configuration. The overall data flow chart of the IO Driver module is as follows:

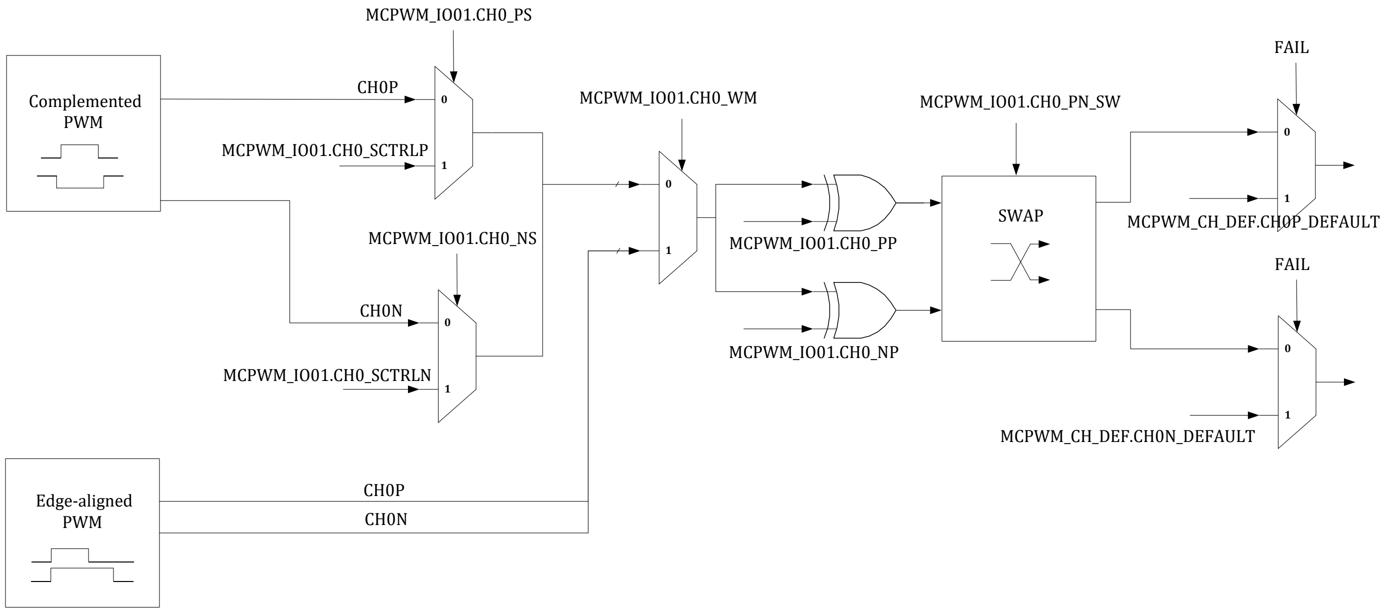


Figure14-7 IO Driver Module Data Flow Diagram

14.1.4.1 MCPWM Wave-form Output: Center-aligned Mode

The four MCPWM IO Drivers use independent control thresholds and independent dead-zone widths (each pair of complementary IO dead-zones need to be configured independently, that is, four dead-zone configuration registers) and share data update events.

TH <n> 0 and TH <n> 1 are used to control the start and shutdown of the <n> MCPWM IO, n is 1, 2, 3, and 4.

When the counter CNT counts up to TH <n> 0, CH <n> N is turned off at time t3, and DTH0 is delayed after dead-zone delay, and CH <n> P is turned on.

When the counter CNT value counts up to reach TH <n> 1, CH <n> P is turned off at time t4, after dead-zone delay DTH1, CH <n> N is turned on.

The independent startup and shutdown time control is adopted to provide phase control.

The dead-zone delay guarantees that CH <n> P/CH <n> N will not be high at the same time to avoid short circuit.

Both t3 and t4 will generate corresponding interrupts.

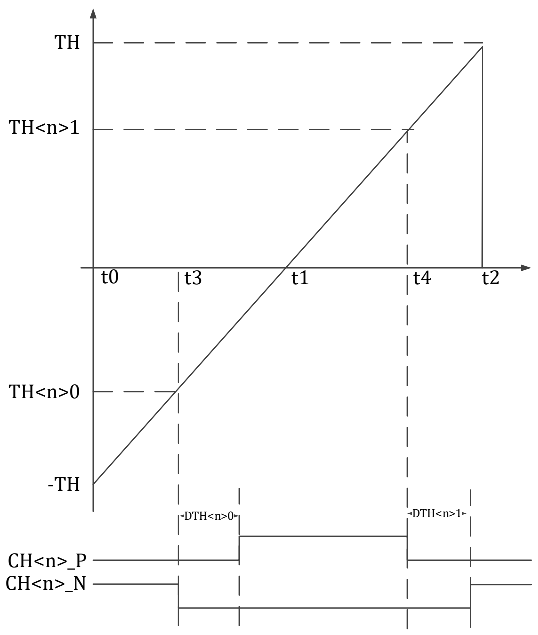


Figure14-8MCPWM Timings TH < n > 0 and TH < n > 1-Center Alignment Mode

14.1.4.2 MCPWM Wave-form Control: Center-aligned Push-pull Mode

Complementary push-pull mode. In the first cycle, CH <n> P is set to 1 at time t3, and CH <n> P goes low at time t4. In the second cycle, CH <n> N is set to 1 at time t3, and CH <n> N goes low at time t4.

Both t3 and t4 will generate corresponding interrupts.

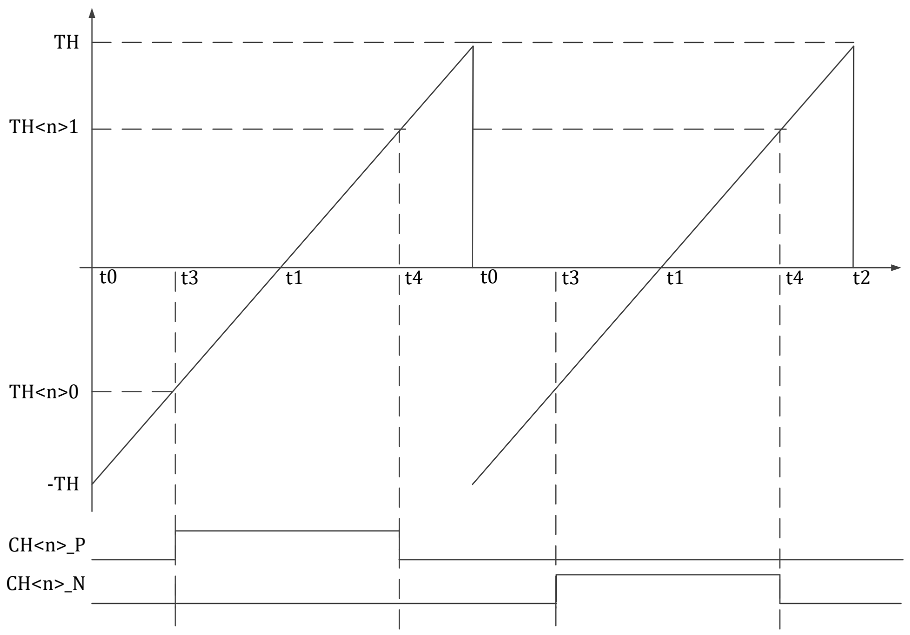


Figure14-9 MCPWM Timings TH < n > 0 and TH < n > 1-Center Aligned Push-Pull Mode

14.1.4.3 MCPWM Wave-form Output: Edge-aligned Mode

In edge-aligned mode, CH <n> P and CH <n> N are reset to 0 at time t0 at the same time, then CH<n>N goes high at time t3, and CH<n>P goes high at time t4.

Both t3 and t4 will generate corresponding interrupts.

In edge-aligned mode, CH <n> P and CH <n> N don't need dead-zone protection.

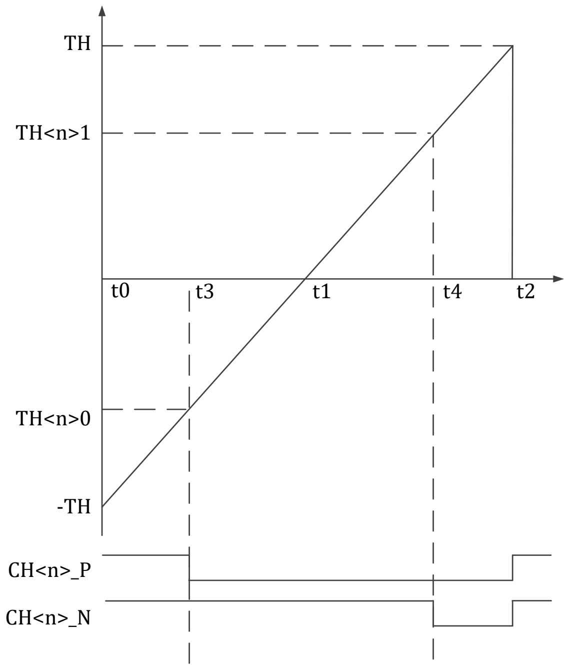


Figure14-10MCPWM Timing Edge Alignment Mode

14.1.4.4 MCPWM Waveform Control-Edge Aligned Push-Pull Mode

Edge-aligned push-pull mode. In the first cycle, CH <n> P is set to 1 at time t0, and CH <n> P goes low at time t3. In the second cycle, CH <n> N is set to 1 at time t0, and CH <n> N goes low at time t3.

Both t0 and t3 will generate corresponding interrupts.

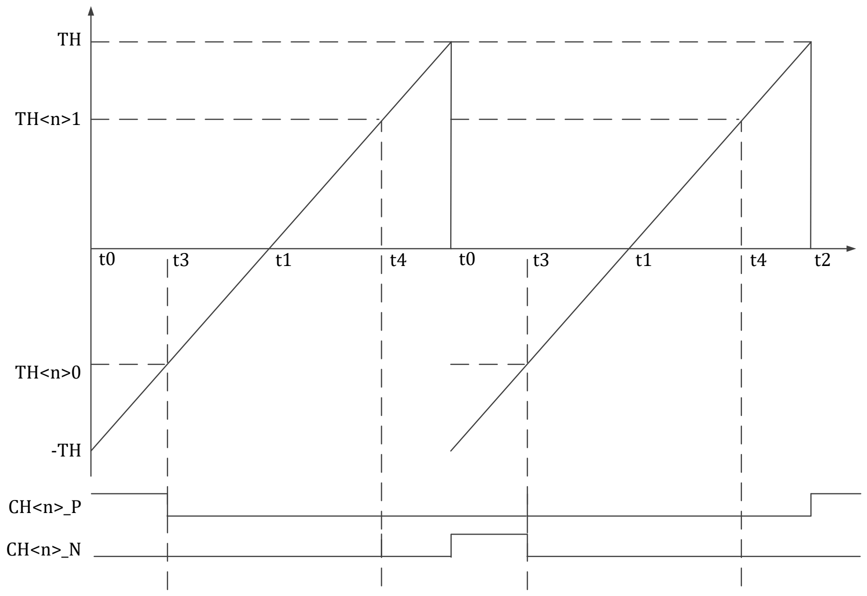


Figure14-11MCPWM Timing TH < n > 0 and TH < n > 1 Edge-Aligned Push-Pull Mode

14.1.4.5 MCPWM IO : Dead-zone Control

MCPWM IO is a pair of mutually exclusive control signals CH <n> P/CH <n> N, which controls the circuit shown in the figure below,

When CH <n> P is high and CH <n> N is low, Vout output is high (VDD);

When CH <n> P is low and CH <n> N is high, Vout output is low (VSS);

When CH <n> P is high and CH <n> N is high, Vout output is undefined, but a short circuit from VDD to VSS will occur accordingly;

When CH <n> P is low and CH <n> _ is low, the Vout output is undefined.

It is necessary to avoid the situation where CH <n> P and CH <n> N are both high. The introduction of dead-zone can avoid the short circuit from VDD to VSS effectively.

The dead-zone width of the four groups of MCPWM IO can be adjusted independently.

For complementary mode, MCPWM IO is automatically inserted into the dead-zone.

For edge-aligned mode, MCPWM IO has no dead-zone.

Added conflict detection for CH <n> P and CH <n> N in the IO Driver module. When a conflict occurs, it will pull IO low automatically and output an error interrupt (The error interrupt flag bit can be cleared by software or hardware automatically).

MCPWM IO can also be output by software configuration, that is, the dead-zone control is controlled by software. If the MCPWM module is configured in center-aligned mode, the hardware short-circuit protection mechanism is still effective to ensure that P and N are not turned on at the same time.

The position of CH <n> P and CH <n> N output to IO can be interchanged.

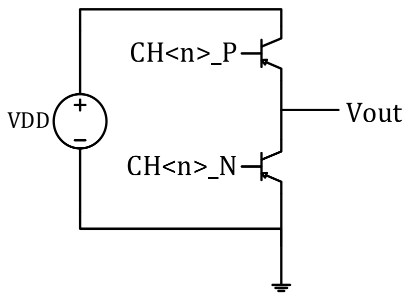


Figure14-12MCPWM IO Control Schematic

14.1.4.6 MCPWM IO: Polarity Settings

The effective levels of CH <n> P and CH <n> N can be set as high and low, and the effective level of each IO can be set individually. The position of CH <n> P and CH <n> N output to IO can be interchanged by software configuration.

14.1.4.7 MCPWM IO: Auto- Protection

When an emergency stop event (Fail event) occurs, CH < n > P/CH < n > N shall be automatically switched to the default safe level immediately. Note the default level configuration (the MCPWM0_CH_DEF. CHxN_DEFAULT and the MCPWM0_CH_DEF. CHxP_DEFAULT control the default level).

- After the chip works normally, the default output level of IO is the value specified by the MCPWM0_CH_DEF. CHxN_DEFAULT and MCPWM0_CH_DEF. CHxP_DEFAULT. Configure the MCPWM0_FAILx.MCPWM0_OE (MOE) to 1. The IO output level is controlled by the MCPWM IO module.
- When a FAIL short circuit condition occurs, the hardware immediately switches to the IO default output level.
- When the CPU is suspended by the upper computer software during chip debugging, the MCPWM output can be suspended and the default level value can be output instead..

14.1.5 ADC Trigger Timer module

MCPWM can provide ADC sampling control. When the counter counts to MCPWM_TMR0/MCPWM_TMR1/MCPWM_TMR2/MCPWM_TMR3, a timing event can be generated to trigger ADC sampling. Besides, the trigger signal can be output to GPIO for debugging. For the specific GPIO output, please refer to the datasheet of the corresponding device.MCPWM0_TMR0/ MCPWM0_TMR1 uses time base 0 permanently, while MCPWM0_TMR2/ MCPWM0_TMR3 can use time base 0/1 optionally, see 14.2.33 MCPWM0_TCLK.

Table14-1 MCPWM Counter Threshold and Event Mapping Table

MCPWMx Time base0 t0 Event	MCPWMx_CNT0 = -MCPWMx_TH0
MCPWMx Time base0 T1 Event	MCPWMx_CNT0 = 0
MCPWMx Time base1 t0 Event	MCPWMx_CNT1 = -MCPWMx_TH1

MCPWMx Time base1 T1 Event	MCPWMx_CNT1 = 0
MCPWMx TIO0 [0] Event	MCPWMx_CNT0 = MCPWMx_TH00
MCPWMx TIO0 [1] Event	MCPWMx_CNT0 = MCPWMx_TH01
MCPWMx TIO1 [0] Event	MCPWMx_CNT0 = MCPWMx_TH10
MCPWMx TIO1 [1] Event	MCPWMx_CNT0 = MCPWMx_TH11
MCPWMx TIO2 [0] Event	MCPWMx_CNT0 = MCPWMx_TH20
MCPWMx TIO2 [1] Event	MCPWMx_CNT0 = MCPWMx_TH21
MCPWMx TIO3 [0] Event	MCPWMx_CNT1 = MCPWMx_TH30
MCPWMX TIO3 [1] event	MCPWMx_CNT1 = MCPWMx_TH31
MCPWMx TIO4 [0] Event	MCPWMx_CNT1 = MCPWMx_TH40
MCPWMx TIO4 [1] Event	MCPWMx_CNT1 = MCPWMx_TH41
MCPWMx TIO5 [0] Event	MCPWMx_CNT1 = MCPWMx_TH50
MCPWMx TIO5 [1] Event	MCPWMx_CNT1 = MCPWMx_TH51
MCP WMx TADC [0] event	TMR0
MCP WMx TADC [1] Event	TMR1
MCP WMx TADC [2] Event	TMR2
MCP WMx TADC [3] Event	TMR3

14.1.6 Digital power applications

In the digital power supply application, the MCPWM can be linked with the comparator, Timer, and DAC. In this configuration, the state machine inside the MCPWM is completely controlled by the comparison value output by the comparator. The MCPWM0_CNT0/1 is no longer automatically increasing counting, that is, the output level of MCPWM channel is controlled by the output value of the comparator.

The timing of the zero-crossing event of the Timer triggers the DAC to update its value, which causes the DAC to output an incrementing or decrementing ramp signal that is connected to the negative terminal of a comparator whose positive terminal is connected to an external sample signal. When the DAC signal rises or falls to a certain value, the comparator is triggered to flip, thereby further triggering the action of the MCPWM internal state machine. The SYS_AFE_DAC_CTRL. TIMERm_TRIG_DACx allows you to set which Timer's zero-crossing event is used as the DAC value update event, m = 0, 1, 2, 3 for TIMER0/1/2/3, X = 0, 1 for DAC0 or DAC1. The update step of the DAC value is set by the SYS_AFE_DAC_CTRL.DACx_STEP.

Digital power applications usually require only one of the three MCPWM channels to be used, for example, the MCPWM0_TCLK.CMP_CTRL_CNT0 = 1 can be set to use the CH0N/P channel, and the MCPWM0_TCLK.CMP_CTRL_CNT1 = 1 can be set to use the CH3N/P channel. Channels 1/2/4/5 do not support linkage with comparators.

MCPWM is divided into HS_ON, LS_ON and ALL_OFF in such applications. The state switching between HS_ON and LS_ON has a minimum switching time, or can be considered as a state minimum staying time, that is, when the MCPWM switches from the HS_ON to the LS_ON state, it stays for at least a period of time before switching back to the HS_ON state, and vice versa. The state minimum dwell time is set by the MCPWM0_STT_HYST.

When the state switches, the MCPWM generates a t0 event (MCPWM0_CNTx = -MCPWM0_THx) or a t1 event (MCPWm0_CNTx = 0) synchronously. If CH0N/P is used for digital power supply application, it is generally required to synchronously set MCPWM0_RE.TR0_T0_RE = 1 and MCPWM0_RE.TR0_T1_RE = 1 to enable DMA triggering of t0/t1 event of MCPWM time base 0; Similarly, if CH3N/P is used for digital power supply applications, it is generally necessary to synchronously set MCPWM0_RE.TR1_T0_RE = 1 and MCPWM0_RE.TR1_T1_RE = 1 to enable the DMA trigger of the t0/T1 event of MCPWM time base 1. Uch that shortly after the MCPWM enters a new state, the DMA stores the new DAC reference value in the SYS_AFE_DAC register, and the subsequent Timer timing triggers the DAC output to generate a ramp signal starting from the new DAC reference value.

Due to the limitation of the comparator resources, the MCPWM0_TCLK.CMP_CTRL_CNT0 = 1 and the MCPWM0_TCLK.CMP_CTRL_CNT0 = 1 are usually not set at the same time. Typically, only one phase channel in CH0N/P or CH3N/P is used for digital power applications at the same time.

HS_ON status: PWM upper tube output is high, lower tube output is low.

Timer is cleared to start counting, DAC0 is triggered at regular time to input digital quantity to update SYS_AFE_DAC0, and DAC0 generates a slope signal.

When the output of the CMP1 is high, and the minimum dwell time of the state is satisfied, it is switched to the LS_ON state. Close the upper tube first, and open the lower tube after the dead time. At the same time, the zero-crossing event of the MCPWM triggers the DMA to load the SYS_AFE_DAC0 as the LS_ON reference value.

When overcurrent occurs, the MCPWM high side and low side channels are all closed. Enter the ALL_OFF state.

LS_ON status: PWM upper tube output is low, lower tube output is high.

Timer is cleared to start counting, DAC0 is triggered to input digital quantity update, and DAC0 generates slope signal.

When the output of the CMP1 is low and the state minimum dwell time is satisfied, the state is switched to the HS_ON state. Close the lower tube first, and open the upper tube after the dead time. At the same time, the zero-crossing event of the MCPWM triggers the DMA to load the SYS_AFE_DAC0 as the HS_ON reference value.

When overcurrent occurs, the high-side and low-side switches are fully closed. Enter the ALL_OFF state.

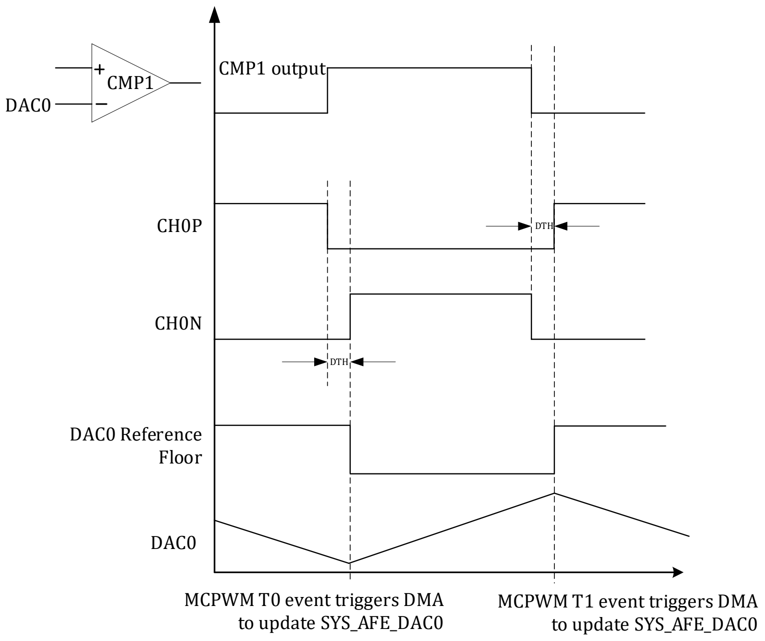

CMP1
DAC0
CMP1 output
CH0P
DTH
CH0N
DTH
DAC0 Reference Floor
DAC0
MCPWM T0 event triggers DMA to update SYS_AFE_DAC0
MCPWM T1 event triggers DMA to update SYS_AFE_DAC0

Figure14-13 Timing Diagram of MCPWM Channel 0 Controlled by Comparator 1 for State Action

In particular, in the HS_ON state, if the output of the current polarity signal CMP2 is low, it is a ZCS event, when ZCS occurs, the upper transistor and the lower transistor need to be turned off simultaneously, and then after a time delay Tzcs, the time Tzcs can be set by the MCPWM0_ZCS_DELAY to enter the LS_ON state. In the LS_ON state, if the output of the current polarity signal CMP2 is high, it is also a ZCS event, and the upper and lower transistors need to be turned off at the same time, and then enter the HS_ON state after a delay of Tzcs.

The Tzcs time can be set via the MCPWM0_ZCS_DELAY.

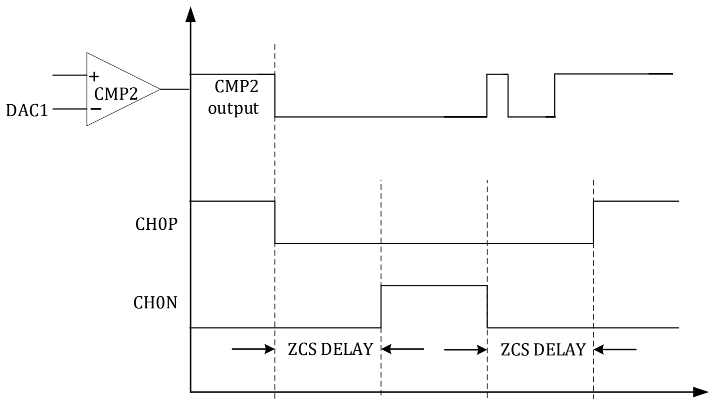


Figure14-14 MCPWM Channel 0 Action Timing When Comparator 2 Triggers ZCS Event

14.1.7 Interrupt

When the MCPMW_IF0 and MCPWM0_EIF [5:4], MCPWM0_EIF [8] flags are set and enabled, the MCPWM0_IRQ0 interrupt is generated.

When the MCPMW_IF1 and MCPWM0_EIF [7:6], MCPWM0_EIF [9] flags are set and enabled, the MCPWM0_IRQ1 interrupt is generated.

14.2 Register

14.2.1 Address assignment

The base address of the MCPWM module register is 0x4001_0C00.

The register list is as follows:

Table14-2 MCPWM Module Register List

Name	Offset address	Explain
MCPWM0_TH00	0x00	MCPWM CH0_P Compare Threshold Register
MCPWM0_TH01	0x04	MCPWM CH0_N Compare Threshold Register
MCPWM0_TH10	0x08	MCPWM CH1_P Compare Threshold Register
MCPWM0_TH11	0x0C	MCPWM CH1_N Compare Threshold Register
MCPWM0_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWM0_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWM0_TH30	0x18	MCPWM CH3_P Compare Threshold Register
MCPWM0_TH31	0x1C	MCPWM CH3_N Compare Threshold Register
MCPWM0_TH40	0x20	MCPWM CH4_P Compare Threshold Register
MCPWM0_TH41	0x24	MCPWM CH4_N Compare Threshold Register
MCPWM0_TH50	0x28	MCPWM CH5_P Compare Threshold Register
MCPWM0_TH51	0x2C	MCPWM CH5_N Compare Threshold Register
MCPWM0_TMR0	0x30	ADC Sample Timer Compare Threshold 0 Register
MCPWM0_TMR1	0x34	ADC Sample Timer Compare Threshold 1 Register
MCPWM0_TMR2	0x38	ADC Sample Timer Compare Threshold 2 Register
MCPWM0_TMR3	0x3C	ADC Sample Timer Compare Threshold 3 Register
MCPWM0_TH0	0x40	MCPWM Time Base 0 Threshold Register
MCPWM0_TH1	0x44	MCPWM Time Base 1 Threshold Register
MCPWM0_CNT0	0x48	MCPWM Time Base 0 Counter Register
MCPWM0_CNT1	0x4C	MCPWM Time Base 1 Counter Register
MCPWM0_UPDATE	0x50	MCPWM Load Control Register
MCPWM0_FCNT	0x54	CNT value at MCPWM FAIL
MCPWM0_EVT0	0x58	MCPWM Time Base 0 External Trigger
MCPWM0_EVT1	0x5C	MCPWM Time Base 1 External Trigger
MCPWM0_DTH00	0x60	MCPWM CH0/1/2 N-Channel dead-zone Width Control Register
MCPWM0_DTH01	0x64	MCPWM CH0/1/2 P Channel Dead Width Control Register
MCPWM0_DTH10	0x68	MCPWM CH3/4/5 N-Channel dead-zone Width Control Register
MCPWM0_DTH11	0x6C	MCPWM CH3/4/5 P Channel Dead Width Control Register
MCPWM0_FLT	0x70	MCPWM Filter Clock Divide Register
MCPWM0_SDCFG	0x74	MCPWM Load Configuration Register
MCPWM0_AUEN	0x78	MCPWM Auto Update Enable Register
MCPWM0_TCLK	0x7C	MCPWM Clock Divide Control Register
MCPWM0_IE0	0x80	MCPWM Time Base 0 Interrupt Control Register
MCPWM0_IF0	0x84	MCPWM Time Base 0 Interrupt Flag Register

MCPWM0_IE1	0x88	MCPWM Interrupt Control Register
MCPWM0_IF1	0x8C	MCPWM Interrupt Flag
MCPWM0_EIE	0x90	MCPWM Abnormal Interrupt Control Register
MCPWM0_EIF	0x94	MCPWM Abnormal Interrupt Flag
MCPWM0_RE	0x98	MCPWM DMA Request Control register
MCPWM0_PP	0x9C	MCPWM Push-Pull Mode Enable Register
MCPWM0_IO01	0xA0	MCPWM CH0 CH1 IO Control Register
MCPWM0_IO23	0xA4	MCPWM CH2 CH3 IO Control Register
MCPWM0_IO45	0xA8	MCPWM CH4 CH5 IO Control Register
MCPWM0_FAIL012	0xB0	MCPWM CH0 CH1 CH2 short circuit control register
MCPWM0_FAIL345	0xB4	MCPWM CH3 CH4 CH5 short circuit control register
MCPWM0_CH_DEF	0xB8	MCPWM Short Circuit Protection Channel Output Default
MCPWM0_CH_MSK	0xBC	MCPWM Channel Mask Register
MCPWM0_PRT	0xC0	MCPWM protection register
MCPWM0_STT_HYST	0xC8	MCPWM Dwell Delay Register
MCPWM0_ZCS_DELAY	0xCC	MCPWM ZCS Status Delay Register

Table 14-3 Register protected by MCPWM0_PRT

Name	Offset address	Explain
MCPWM0_TH0	0x30	MCPWM Threshold Register
MCPWM0_TH1	0x34	MCPWM Threshold Register
MCPWM0_DTH00	0x60	MCPWM CH0/1/2 N-Channel dead-zone Width Control Register
MCPWM0_DTH01	0x64	MCPWM CH0/1/2 P Channel Dead Width Control Register
MCPWM0_DTH10	0x68	MCPWM CH3/4/5 N-Channel dead-zone Width Control Register
MCPWM0_DTH11	0x6C	MCPWM CH3/4/5 P Channel Dead Width Control Register
MCPWM0_FLT	0x70	MCPWM Filter Clock Divide Register
MCPWM0_SDCFG	0x74	MCPWM Load Configuration Register
MCPWM0_AUEN	0x78	MCPWM autoload enable register
MCPWM0_TCLK	0x7C	MCPWM Clock Divide Control Register
MCPWM0_IE0	0x80	MCPWM Time Base 0 Interrupt Control Register
MCPWM0_IE1	0x88	MCPWM Time Base1 Interrupt Control Register
MCPWM0_EIE	0x90	MCPWM Abnormal Interrupt Control Register
MCPWM0_RE	0x98	MCPWMDMA Request Control
MCPWM0_PP	0x9C	MCPWM Push-Pull Mode Enable Register
MCPWM0_IO01	0xA0	MCPWM CH0 CH1 IO Control Register
MCPWM0_IO23	0xA4	MCPWM CH2 CH3 IO Control Register
MCPWM0_IO45	0xA8	MCPWM CH4 CH5 IO Control Register
MCPWM0_FAIL012	0xB0	MCPWM CH0 CH1 CH2 short circuit control register
MCPWM0_FAIL345	0xB4	MCPWM CH3 CH4 CH5 short circuit control register
MCPWM0_CH_DEF	0xB8	Output value of MCPWM short-circuit protection channel
MCPWM0_CH_MSK	0xBC	MCPWM Channel Mask Register
MCPWM0_STT_HYST	0xC8	MCPWM Dwell Delay Register
MCPWM0_ZCS_DELAY	0xCC	MCPWM ZCS Status Delay Register

Table 14-4 Registers where shadow registers exist

Name	Offset address	Explain
MCPWM0_TH00	0x00	MCPWM CH0_P Compare Threshold Register
MCPWM0_TH01	0x04	MCPWM CH0_N Compare Threshold Register
MCPWM0_TH10	0x08	MCPWM CH1_P Compare Threshold Register
MCPWM0_TH11	0x0C	MCPWM CH1_N Compare Threshold Register
MCPWM0_TH20	0x10	MCPWM CH2_P compare threshold register
MCPWM0_TH21	0x14	MCPWM CH2_N compare threshold register
MCPWM0_TH30	0x18	MCPWM CH3_P Compare Threshold Register
MCPWM0_TH31	0x1C	MCPWM CH3_N Compare Threshold Register
MCPWM0_TH40	0x20	MCPWM CH4_P Compare Threshold Register
MCPWM0_TH41	0x24	MCPWM CH4_N Compare Threshold Register
MCPWM0_TH50	0x28	MCPWM CH5_P Compare Threshold Register
MCPWM0_TH51	0x2C	MCPWM CH5_N Compare Threshold Register
MCPWM0_TMR0	0x30	Compare threshold 0 register for ADC sampling timer
MCPWM0_TMR1	0x34	Compare threshold 1 register for ADC sampling timer
MCPWM0_TMR2	0x38	Compare threshold 2 register for ADC sampling timer
MCPWM0_TMR3	0x3C	Compare threshold 3 register for ADC sampling timer
MCPWM0_TH0	0x40	MCPWM Time Base 0 Threshold Register
MCPWM0_TH1	0x44	MCPWM Time Base 1 Threshold Register
MCPWM0_CNT0	0x48	MCPWM Time Base 0 Counter Register
MCPWM0_CNT1	0x4C	MCPWM Time Base 1 Counter Register

For all MCPWM configuration registers with shadow registers, the preload register is written when writing, the preload register value is written to the shadow register when an update event occurs, and the shadow register value is read when reading.

14.2.2 MCPWM0_TH00

Register without write protection

Address: 0x4001_0C00

Reset value: 0x0

Table14-5 MCPWM0_TH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH00															
RW															
0															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TH00	MCPWM CH0_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM. The preload register is written, the preload register value is written to the shadow register only when an update event occurs, and the shadow register value is read when reading.

14.2.3 MCPWM0_TH01

Register without write protection

Address: 0x4001 0C04

Reset value: 0x0

Table14-6 MCPWM0_TH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH01															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH01	MCPWM CH0_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.4 MCPWM0_TH10

Register without write protection

Address: 0x4001 0C08

Reset value: 0x0

Table14-7 MCPWM0_TH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH10															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
15:0]	TH10	MCPWM CH1_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.5 MCPWM0_TH11

Register without write protection

Address: 0x4001 0C0C

Reset value: 0x0

Table14-8 MCPWM0_TH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH11															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH11	MCPWM CH1_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.6 MCPWM0_TH20

Register without write protection

Address: 0x4001 0C10

Reset value: 0x0

Table14-9 MCPWM0_TH20 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH20															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH20	MCPWM CH2_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.7 MCPWM0_TH21

Register without write protection

Address: 0x4001 0C14

Reset value: 0x0

Table14-10 MCPWM0_TH21 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH21															
RW															
0															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TH21	MCPWM CH2_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.8 MCPWM0_TH30

Register without write protection

Address: 0x4001 0C18

Reset value: 0x0

Table14-11 MCPWM0_TH30 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH30															
RW															
0															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TH30	MCPWM CH3_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.9 MCPWM0_TH31

Register without write protection

Address: 0x4001_0C1C

Reset value: 0x0

Table14-12 MCPWM0_TH31 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH31															
RW															
0															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TH31	MCPWM CH3_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.10 MCPWM0_TH40

Register without write protection

Address: 0x4001_0C20

Reset value: 0x0

Table14-13 MCPWM0_TH40 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH30															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH40	MCPWM CH4_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.11 MCPWM0_TH41

Register without write protection

Address: 0x4001 0C24

Reset value: 0x0

Table14-14 MCPWM0_TH41 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH31															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH41	MCPWM CH4_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.12 MCPWM0_TH50

Register without write protection

Address: 0x4001 0C28

Reset value: 0x0

Table14-15 MCPWM0_TH50 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH30															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH50	MCPWM CH5_P comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.13 MCPWM0_TH51

Register without write protection

Address: 0x4001_0C2C

Reset value: 0x0

Table14-16 MCPWM0_TH51 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TH31															
RW															
0															

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	TH51	MCPWM CH5_N comparison threshold, 16-bit signed number; When an update event occurs, this register is loaded into the actual operating system of the MCPWM.

14.2.14 MCPWM0_TMR0

Register without write protection

Address: 0x4001 0C30

Reset value: 0x0

Table14-17 MCPWM0_TMR0 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR0															
RW															
0x7FFF															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TMR0	ADC sample timer compare threshold 0 register, 16-bit signed number; A TADC [0] event generated when MCPWM0_CNT0 = TMR0 triggers the ADC to sample. After an MCPWM update event, this register is loaded into the system where the MCPWM is actually running.

14.2.15 MCPWM0_TMR1

Register without write protection

Address: 0x4001 0C34

Reset value: 0x0

Table14-18 MCPWM0_TMR1 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR1															
RW															
0x7FFF															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TMR1	ADC Sample Timer Compare Threshold 1 register, 16-bit signed number; A TADC [1] event generated when MCPWM0_CNT = TMR1 triggers the ADC to sample. After an MCPWM update event, this register is loaded into the system where the MCPWM is actually running.

14.2.16 MCPWM0_TMR2

Register without write protection

Address: 0x4001 0C38

Reset value: 0x0

Table14-19 MCPWM0_TMR2 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR2															
RW															
0x7FFF															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TMR2	ADC Sample Timer Compare Threshold 2 register, 16-bit signed number; After an update event, this register is loaded into the actual operating

		system of the MCPWM.

14.2.17 MCPWM0_TMR3

Register without write protection

Address: 0x4001_0C3C

Reset value: 0x0

Table14-20 MCPWM0_TMR3 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMR3															
RW															
0x7FFF															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	TMR3	ADC sampling timer comparison threshold 3 register, 16-bit signed number; After an update event, this register is loaded into the actual operating system of the MCPWM.

14.2.18 MCPWM0_TH0

Write-protected register

Address: 0x4001 0C40

Reset value: 0x0

Table14-21 MCPWM0_TH0 Time Base 0 Register

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

Location	Bit name	Explain
[31:15]		Not used
[14:0]	TH	The threshold value of the MCPWM time base 0 counter is a 15-bit unsigned number, and the time base 0 counter in the MCPWM actual operation system counts from -TH to TH; After an update event, this register is loaded into the actual operating system of the MCPWM.

14.2.19 MCPWM0_TH1

Write-protected register

Address: 0x4001 0C44

Reset value: 0x0

Table14-22 MCPWM0_TH1 Time Base 1 Register

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

Location	Bit name	Explain
[31:15]		Not used
[14:0]	TH	The threshold value of the MCPWM time base 1 counter is a 15-bit unsigned number, and the time base 1 counter in the MCPWM actual operation system counts from -TH to TH; After an update event, this register is loaded into the actual operating system of the MCPWM.

14.2.20 MCPWM0_CNT0

Register without write protection

Address: 0x4001 0C48

Reset value: 0x8000

Table14-23 MCPWM0_CNT0 Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
RW															
0x8000															

Location	Bit name	Explain
[31:16]		Not used
[15:0]	CNT	Write to this register to change the setting of the time base 0 counter. After an update event, this register is loaded into the time base 0 CNT of the system where the MCPWM is actually running. The data read out is the value of the time base 0 counter in the actual operating system of the MCPWM. The actual readout counts range from -TH to +TH.

14.2.21 MCPWM0_CNT1

Register without write protection

Address: 0x4001_0C4C

Reset value: 0x8000

Table14-24 MCPWM0_CNT1 Register

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

CNT
RW
0x8000

Location	Bit name	Explain
[31:16]		Not used
[15:0]	CNT	Write to this register to change the setting of the time base1 counter. After an update event, this register is loaded into the time base1 CNT of the system where the MCPWM is actually running. The data read out is the value of the time base 1 counter in the actual operating system of the MCPWM. The actual readout counts range from -TH to +TH.

14.2.22 MCPWM0_UPDATE

Register without write protection

Address: 0x4001 0C50

Reset value: 0x0

Table14-25 MCPWM0_UPDATE MCPWM Manual Update Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				CNT1_UPDATE	TH1_UPDATE	TMR3_UPDATE	TMR2_UPDATE			TH51_UPDATE	TH50_UPDATE	TH41_UPDATE	TH40_UPDATE	TH31_UPDATE	TH30_UPDATE
				WO	WO	WO	WO			WO	WO	WO	WO	WO	WO
				0	0	0	0			0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CNT0_UPDATE	TH0_UPDATE	TMR1_UPDATE	TMR0_UPDATE			TH21_UPDATE	TH20_UPDATE	TH11_UPDATE	TH10_UPDATE	TH01_UPDATE	TH00_UPDATE
				WO	WO	WO	WO			WO	WO	WO	WO	WO	WO
				0	0	0	0			0	0	0	0	0	0

Location	Bit name	Explain
[31:28]		Not used
[27]	CNT1_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_CNT1 from the preload register into the shadow register used by the MCPWM operation
[26]	TH1_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH1 from the preload register into the shadow register used by the MCPWM operation
[25]	TMR3_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TMR3 from the preload register into the shadow register used by the MCPWM operation
[24]	TMR2_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TMR2

		from the preload register into the shadow register used by the MCPWM operation
[23:22]		Not used
[21]	TH51_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH51 from the preload register into the shadow register used by the MCPWM operation
[20]	TH50_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH50 from the preload register into the shadow register used by the MCPWM operation
[19]	TH41_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH41 from the preload register into the shadow register used by the MCPWM operation
[18]	TH40_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH40 from the preload register into the shadow register used by the MCPWM operation
[17]	TH31_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH31 from the preload register into the shadow register used by the MCPWM operation
[16]	TH30_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH30 from the preload register into the shadow register used by the MCPWM operation
[15:12]		Not used
[11]	CNT0_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_CNT0 from the preload registers into the shadow registers used by the MCPWM operation
[10]	TH0_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH0 from the preload register into the shadow register used by the MCPWM operation
[9]	TMR1_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TMR1 from the preload register into the shadow register used by the MCPWM operation
[8]	TMR0_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TMR0 from the preload registers into the shadow registers used by the MCPWM operation
[7:6]		Not used
[5]	TH21_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH21 from the preload register into the shadow register used by the MCPWM operation
[4]	TH20_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH20 from the preload register into the shadow register used by the MCPWM operation
[3]	TH11_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH11 from the preload register into the shadow register used by the MCPWM operation
[2]	TH10_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH10 from the preload register into the shadow register used by the MCPWM operation
[1]	TH01_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH01 from the preload register into the shadow register used by the MCPWM operation
[0]	TH00_UPDATE	Writing a 1 causes a software (manual) trigger to load MCPWM0_TH00 from the preload registers into the shadow registers used by the MCPWM operation

Writing a 1 to the corresponding bit of MCPWM0_UPDATE triggers the register value to be written from the preload register to the shadow register. MCPWM0_UPDATE is automatically cleared after being written, and no software clearing is required. Perform a software/manual trigger every time 1 is written. Updating to MCPWM0_UPDATE [15:14] causes MCPWM0_CNT0/1 to be updated to the preload value. Note whether CNT needs to be updated.

Writing 0 to MCPWM0_UPDATE does not work. Writing 0 is not recommended.

14.2.23 MCPWM0_FCNT

Register without write protection

Address: 0x4001 0C54

Reset value: 0x0

Table14-26 MCP WM0_FCNT Register

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

Location	Bit name	Explain
[31:16]		Not used
[15:0]	FCNT	If MCPWM0_FAIL012 [15] = 1, when the fail0/1 event occurs, record the value of MCPWM0_CNT0 and store it into MCPWM0_FCNT; If MCPWM0_FAIL345 [15] = 1, when the fail2/3 event occurs, record the MCPWM0_CNT1 value and store it into MCPWM0_FCNT

14.2.24 MCPWM0_EVT0

Register without write protection

Address: 0x4001 0C58

Reset value: 0x0

Table14-27 MCPWM0_EVT0 MCPWM Time Base 0 External Trigger Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM3_CMP1	TIM3_CMP0	TIM2_CMP1	TIM2_CMP0	TIM1_CMP1	TIM1_CMP0	TIM0_CMP1	TIM0_CMP0					PWM0_TMR3	PWM0_TMR2	PWM0_TMR1	PWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW
0	0	0	0	0	0	0	0					0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	TIM3_CMP1	TIME R3 CMP1 event triggers time base 0 to start counting

[14]	TIM3_CMP0	TIMER3 CMP0 event triggers time base 0 to start counting
[13]	TIM2_CMP1	TIMER2 CMP1 event triggers time base 0 to start counting
[12]	TIM2_CMP0	TIMER2 CMP0 event triggers time base 0 to start counting
[11]	TIM1_CMP1	TIMER1 CMP1 event triggers time base 0 to start counting
[10]	TIM1_CMP0	TIMER1 CMP0 event triggers time base 0 to start counting
[9]	TIM0_CMP1	TIMER0 CMP1 event triggers time base 0 to start counting
[8]	TIM0_CMP0	TIMER0 CMP0 event triggers time base 0 to start counting
[7:4]		Not used
[3]	PWM0_TMR3	MCPWM0 TMR3 event triggers time base0 to start counting
[2]	PWM0_TMR2	MCPWM0 TMR2 event triggers time base0 to start counting
[1]	PWM0_TMR1	MCPWM0 TMR1 event triggers time base0 to start counting
[0]	PWM0_TMR0	MCPWM0 TMR0 event triggers time base0 to start counting

14.2.25 MCPWM0_EVT1

Register without write protection

Address: 0x4001_0C5C

Reset value: 0x0

Table14-28 MCPWM0_EVT1 MCPWM Time Base 1 External Trigger Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM3_CMP1	TIM3_CMP0	TIM2_CMP1	TIM2_CMP0	TIM1_CMP1	TIM1_CMP0	TIM0_CMP1	TIM0_CMP0					PWM0_TMR3	PWM0_TMR2	PWM0_TMR1	PWM0_TMR0
RW	RW	RW	RW	RW	RW	RW	RW					RW	RW	RW	RW
0	0	0	0	0	0	0	0					0	0	0	0

Location	Bit name	Explain
[31:12]		Not used
[15]	TIM3_CMP1	TIMER3CMP1 event triggers time base 1 to start counting
[14]	TIM3_CMP0	TIMER3 CMP0 event triggers time base 1 to start counting
[13]	TIM2_CMP1	TIMER2 CMP1 event triggers time base 1 to start counting
[12]	TIM2_CMP0	TIMER2 CMP0 event triggers time base 1 to start counting
[11]	TIM1_CMP1	TIMER1 CMP1 event triggers time base 1 to start counting
[10]	TIM1_CMP0	TIMER1 CMP0 event triggers time base 1 to start counting
[9]	TIM0_CMP1	TIMER0 CMP1 event triggers time base 1 to start counting
[8]	TIM0_CMP0	TIMER0 CMP0 event triggers time base 1 to start counting
[7:4]		Not used
[3]	PWM0_TMR3	MCPWM0 TMR3 event triggers time base 1 to start counting
[2]	PWM0_TMR2	MCPWM0 TMR2 event triggers time base 1 to start counting
[1]	PWM0_TMR1	MCPWM0 TMR1 event triggers time base 1 to start counting
[0]	PWM0_TMR0	MCPWM0 TMR0 event triggers time base1 to start counting

14.2.26 MCPWM0_DTH00

Write-protected register

Address: 0x4001 0C60

Reset value: 0x0

Table14-29 MCPWM0_DTH00 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DTH00									
						RW									
						0									

Location	Bit name	Explain
[31:10]		Not used
[9:0]	DTH00	MCPWM CH0/1/2 N-channel dead-zone width control register, 10-bit unsigned

14.2.27 MCPWM0_DTH01

Write-protected register

Address: 0x4001 0C64

Reset value: 0x0

Table14-30 MCPWM0_DTH01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DTH01									
						RW									
						0									

Location	Bit name	Explain
[31:10]		Not used
[9:0]	DTH01	MCPWM CH0/1/2 P-Channel dead-zone width control register, 10-bit unsigned

14.2.28 MCPWM0_DTH10

Write-protected register

Address: 0x4001 0C68

Reset value: 0x0

Table14-31 MCPWM0_DTH10 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DTH10									
						RW									
						0									

Location	Bit name	Explain
[31:10]		Not used

[9:0]	DTH10	MCPWM CH3/4/5 N-channel dead-zone width control register, 10-bit unsigned

14.2.29 MCPWM0_DTH11

Write-protected register

Address: 0x4001 0C6C

Reset value: 0x0

Table14-32 MCPWM0_DTH11 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						DTH11									
						RW									
						0									

Location	Bit name	Explain
[31:10]		Not used
[9:0]	DTH11	MCPWM CH3/4/5 P-Channel dead-zone width control register, 10-bit unsigned

14.2.30 MCPWM0_FLT

Write-protected register

Address: 0x4001 0C70

Reset value: 0x0

Table14-33 MCPWM0_FLT MCPWM Filter Clock Divide Register

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

Location	Bit name	Explain
[31:8]		Not used
[7:0]	FLT_CLKDIV	Filter clock divide register for GPIO inputs, based on system clock divide, affects MCPWM0_FAIL [1:0]. The calculation formula is as follows: System Clock/ (FLT_CLKDIV + 1). The frequency division range is 1-256.

The MCPWM uses the divided clock to perform 16 cycles of filtering on the FAIL signal. When 256 frequency division is used, the filter width is 4096 TCLK clock widths.

14.2.31 MCPWM0_SDCFG

Write-protected register

Address: 0x4001 0C74

Reset value: 0x0

Table14-34 MCPWM0_SDCFG Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TR1_AEC	TR1_T1_UEN	TR1_T0_UEN	TR1_UP_INTV					TR0_AEC	TR0_T1_UEN	TR0_T0_UEN	TR0_UP_INTV			
	RW	RW	RW	RW					RW	RW	RW	RW			
	0	0	0	0					0	0	0	0			

Location	Bit name	Explain
[31:15]		Not used
[14]	TR1_AEC	Whether the update event automatically clears MCPWM0_EIF [7:6] and sets MCPWM0_FAIL345.MOE to restore the MCPWM Channel 3/4/5 signal output. 1: Enable automatic fault clearing function; 0: Turn off the automatic fault clearing function.
[13]	TR1_T1_UEN	Time base 1 T1 (zero-crossing) event update enable. 1: enable; 0: off.
[12]	TR1_T0_UEN	time base1 t0 (start) event update enabled. 1: enable; 0: off.
[11:8]	TR1_UP_INTV	Time base 1 update interval. The MCPWM system automatically triggers the loading of the MCPWM0_TH1, TH30 to TH51, and MCPWM0_TMR registers into the MCPWM operating system once the t0 and T1 events occur the same number of times as the TR1_UP_INTV. If both TR1_T1_UEN and TR1_T0_UEN are closed, this type of loading will not be triggered and can only be triggered manually.
[7]		Not used
[6]	TR0_AEC	Whether the update event automatically clears MCPWM0_EIF [5:4] and sets MCPWM0_FAIL012.MOE to restore the MCPWM channel 0/1/2 signal output. 1: Enable automatic fault clearing function; 0: Turn off the automatic fault clearing function.
[5]	TR0_T1_UEN	Time base 0 t1(zero-crossing) event update enabled. 1: enable; 0: off.
[4]	TR0_T0_UEN	time base0 t0 (start) event update enabled. 1: enable; 0: off.
[3:0]	TR0_UP_INTV	Time base 0 update interval. Once the number of t0 and t1 events is equal to TR0_UP_INTV + 1,the MCPWM system automatically triggers the loading of the MCPWM0_TH0, MCPWM0_TH00 to TH21, and MCPWM0_TMR registers into the MCPWM operating system . If both TR0_T1_UEN and TR0_T0_UEN are closed, this type of loading will not be triggered and can only be triggered manually.

14.2.32 MCPWM0_AUEN

Write-protected register

Address: 0x4001 0C78

Reset value: 0x0

Table 14-35 MCPWM0_AUEN MCPWM Auto Update Enable Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
				CNT1_AUPDATE	TH1_AUPDATE	TMR3_AUPDATE	TMR2_AUPDATE			TH51_AUPDATE	TH50_AUPDATE	TH41_AUPDATE	TH40_AUPDATE	TH31_AUPDATE	TH30_AUPDATE
				RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
				0	1	1	1			1	1	1	1	1	1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CNT0_AUPDATE	TH0_AUPDATE	TMR1_AUPDATE	TMR0_AUPDATE			TH21_AUPDATE	TH20_AUPDATE	TH11_AUPDATE	TH10_AUPDATE	TH01_AUPDATE	TH00_AUPDATE
				RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
				0	1	1	1			1	1	1	1	1	1

Location	Bit name	Explain
[31:28]		Not used
[27]	CNT1_AUPDATE	MCPWM0_CNT1 autoload enable. 1: Load; 0: Not loaded.
[26]	TH1_AUPDATE	MCPWM0_TH1 autoload enable. 1: Load; 0: Not loaded.
[25]	TMR3_AUPDATE	MCP WM0_TMR3 autoload enable. 1: Load; 0: Not loaded.
[24]	TMR2_AUPDATE	MCP WM0_TMR2 autoload enable. 1: Load; 0: Not loaded.
[23:22]		Not used
[21]	TH51_AUPDATE	MCPWM0_TH51 autoload enable. 1: Load; 0: Not loaded.
[20]	TH50_AUPDATE	MCPWM0_TH50 autoload enable. 1: Load; 0: Not loaded.
[19]	TH41_AUPDATE	MCPWM0_TH41 autoload enable. 1: Load; 0: Not loaded.
[18]	TH40_AUPDATE	MCPWM0_TH40 autoload enable. 1: Load; 0: Not loaded.
[17]	TH31_AUPDATE	MCPWM0_TH31 autoload enable. 1: Load; 0: Not loaded.
[16]	TH30_AUPDATE	MCPWM0_TH30 autoload enable. 1: Load; 0: Not loaded.
[15:12]		Not used
[11]	CNT0_AUPDATE	MCPWM0_CNT0 autoload enable. 1: Load; 0: Not loaded.
[10]	TH0_AUPDATE	MCPWM0_TH0 autoload enable. 1: Load; 0: Not loaded.
[9]	TMR1_AUPDATE	MCPWM0_TMR1 autoload enable. 1: Load; 0: Not loaded.
[8]	TMR0_AUPDATE	MCP WM0_TMR0 autoload enable. 1: Load; 0: Not loaded.
[7:6]		Not used
[5]	TH21_AUPDATE	MCPWM0_TH21 autoload enable. 1: Load; 0: Not loaded.
[4]	TH20_AUPDATE	MCPWM0_TH20 autoload enable. 1: Load; 0: Not loaded.
[3]	TH11_AUPDATE	MCPWM0_TH11 autoload enable. 1: Load; 0: Not loaded.
[2]	TH10_AUPDATE	MCPWM0_TH10 autoload enable. 1: Load; 0: Not loaded.
[1]	TH01_AUPDATE	MCPWM0_TH01 autoload enable. 1: Load; 0: Not loaded.
[0]	TH00_AUPDATE	MCPWM0_TH00 autoload enable. 1: Load; 0: Not loaded.

14.2.33 MCPWM0_TCLK

Write-protected register

0x4001 077C

Reset value: 0x0

Table14-36 MCPWM0_TCLK configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CMP_CTRL_CNT1	CMP_CTRL_CNT0	EVT_CNT1_EN	EVT_CNT0_EN	BASE_CNT1_EN	BASE_CNT0_EN	TMR3_TB	TMR2_TB	ZCS_EN	CLK_EN	CLK_DIV	
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
				0	0	0	0	0	0	0	0	0	0	0	

Location	Bit name	Explain
[31:12]		Not used
[11]	CMP_CTRL_CNT1	The comparator controls the MCPWM0_CNT1 count
[10]	CMP_CTRL_CNT0	The comparator controls the MCPWM0_CNT0 count
[9]	EVT_CNT1_EN	time base1 external trigger enabled
[8]	EVT_CNT0_EN	time base0 external trigger enabled
[7]	BASE_CNT1_EN	MCPWM time base 1 Counter Enable Switch. 1: Enable; 0: Off.
[6]	BASE_CNT0_EN	MCPWM time base 0 Counter Enable Switch. 1: Enable; 0: Off.
[5]	TMR3_TB	TMR3 time base selection, 0: time base 0, 1: time base 1
[4]	TMR2_TB	TMR2 time base selection, 0: time base 0, 1: time base 1
[3]	ZCS_EN	Enable ZCS event detection, active high. When the ZCS_EN is equal to 1 and a ZCS event occurs, the P/N channels are simultaneously turned off, and the state is switched after the MCPWM0_ZCS_DELAY is delayed; When the ZCS_EN is equal to 0, the occurrence of the ZCS event is not delayed and the state is not switched.
[2]	CLK_EN	MCPWM operation clock is enabled. 1: Enable; 0: Off.
[1:0]	CLK_DIV	MCPWM Operating Clock Divide Register. 0: system clock 1: system clock/2 2: system clock/4 3: system clock/8

The MCPWM0_TCLK. CLK_EN must be enabled to complete the shadow register update. After configuring the shadow registers, turn on the MCPWM0_TCLK. BASE_CNT0/1_EN at the same time so that time base 0 and time base 1start counting synchronously. If TH0 and TH1 are set to the same value, time base 0 and time base 1 are at exactly the same frequency.

When the external trigger MCPWM is used to start counting, it is necessary to configure the BASE_CNTx_EN to 0 and the EVT_CNTx_EN to 1, and set MCPWM0_EVTx to select the appropriate external trigger source. After the trigger event occurs, the BASE_CNTx_EN is set to 1 by the hardware, and the MCPWM counter starts counting.

14.2.34 MCPWM0_IE0

Write-protected register

Address: 0x4001 0C80

Reset value: 0x0

Table14-37 MCPWM0_IE0 MCPWM Time Base 0 Interrupt Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2IE	TMR1_IE	TMR0_IE			TH21_IE	TH20_IE	TH11_IE	TH10_IE	TH01_IE	TH00_IE	T1_IE	T0_IE
	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0			0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:15]		Not used
[14]	UP_IE	time base0 update event interrupt source enabled. 1: enable; 0: off.
[13]	TMR3_IE	MCPWM0_CNT0 equals MCPWM0_TMR3 interrupt source enable. 1: enable; 0: off.
[12]	TMR2_IE	MCPWM0_CNT0 equals MCPWM0_TMR2 interrupt source enable. 1: enable; 0, off.
[11]	TMR1_IE	MCPWM0_CNT0 equals MCPWM0_TMR1 interrupt source enable. 1, enable; 0: off.
[10]	TMR0_IE	MCPWM0_CNT0 equals MCPWM0_TMR0 interrupt source enable. 1: enable; 0: off.
[9:8]		Not used
[7]	TH21_IE	MCPWM0_CNT0 equals MCPWM0_TH21 interrupt source enable. 1: enable; 0: off.
[6]	TH20_IE	MCPWM0_CNT0 equals MCPWM0_TH20 interrupt source enable. 1: enable; 0: off.
[5]	TH11_IE	MCPWM0_CNT0 equals MCPWM0_TH11 interrupt source enable. 1: enable; 0: off.
[4]	TH10_IE	MCPWM0_CNT0 equals MCPWM0_TH10 interrupt source enable. 1: enable; 0: off.
[3]	TH01_IE	MCPWM0_CNT0 equals MCPWM0_TH01 interrupt source enable. 1: enable; 0: off.
[2]	TH00_IE	MCPWM0_CNT0 equals MCPWM0_TH00 interrupt source enable. 1: enable; 0: off.
[1]	T1_IE	Time base 0 T1 event, the count value of MCPWM0_CNT0 returns to 0, and the interrupt source is enabled. 1: enable; 0: off.
[0]	T0_IE	Time base 0 T0 event, the count value of MCPWM0_CNT0 is equal to MCPWM0_TH0 interrupt source enable. 1: enable; 0: off.

14.2.35 MCPWM0_IF0

Register without write protection

Address: 0x4001 0C84

Reset value: 0x0

Table14-38 MCPWM0_IF0 MCPWM Time Base 0 Interrupt Flag Register

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

	UP_IF	TMR3_IF	TMR2_IF	TMR1_IF	TMR0_IF		TH21_IF	TH20_IF	TH11_IF	TH10_IF	TH01_IF	TH00_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C	RW1C	RW1C		RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0	0	0		0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:15]		Not used
[14]	UP_IF	Time base 0 update event The MCPWM0_TH0/MCPWM0_TH00 to MCPWM0_TH21/MCPWM_TMR registers are updated to the interrupt event of the actual running system of the MCPWM. 1: occur; 0: didn't happen. Write 1 to clear.
[13]	TMR3_IF	MCPWM0_CNT0 equals the MCPWM0_TMR3 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[12]	TMR2_IF	MCPWM0_CNT0 equals the MCPWM0_TMR2 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[11]	TMR1_IF	MCPWM0_CNT0 equals the MCPWM0_TMR1 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[10]	TMR0_IF	MCPWM0_CNT0 equals MCPWM0_TMR0 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[9:8]		Not used
[7]	TH21_IF	MCPWM0_CNT0 equals the MCPWM0_TH21 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[6]	TH20_IF	MCPWM0_CNT0 equals the MCPWM0_TH20 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[5]	TH11_IF	MCPWM0_CNT0 equals the MCPWM0_TH11 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[4]	TH10_IF	MCPWM0_CNT0 equals the MCPWM0_TH10 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[3]	TH01_IF	MCPWM0_CNT0 equals the MCPWM0_TH01 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[2]	TH00_IF	MCPWM0_CNT0 equals the MCPWM0_TH00 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[1]	T1_IF	T1 event, MCPWM0_CNT0 equals 0 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[0]	T0_IF	T0 event, MCPWM0_CNT0 equals MCPWM0_TH interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.

14.2.36 MCPWM0_IE1

Write-protected register

Address: 0x4001 0C88

Reset value: 0x0

Table14-39 MCPWM0_IE1 MCPWM Time Base 1 Interrupt Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IE	TMR3_IE	TMR2_IE					TH51_IE	TH50_IE	TH41_IE	TH40_IE	TH31_IE	TH30_IE	T1_IE	T0_IE

	RW	RW	RW		RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0		0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:15]		Not used
[14]	UP_IE	Time base 1 update event The MCPWM0_TH1/MCPWM0_TH30 to MCPWM0_TH51/MCPWM0_TMR2 to MCPWM0_TMR3 registers are updated to the interrupt enable of the system where the MCPWM is actually running. 1: enable; 0: off.
[13]	TMR3_IE	MCPWM0_CNT1 equals MCPWM0_TMR3 interrupt enable. 1: enable; 0: off.
[12]	TMR2_IE	MCPWM0_CNT1 equals MCPWM0_TMR2 interrupt enable. 1: enable; 0: off.
[11:8]		Not used
[7]	TH51_IE	MCPWM0_CNT1 equals MCPWM0_TH51 interrupt enable. 1: enable; 0: off.
[6]	TH50_IE	MCPWM0_CNT1 equals MCPWM0_TH50 interrupt enable. 1: enable; 0: off.
[5]	TH41_IE	MCPWM0_CNT1 equals MCPWM0_TH41 interrupt enable. 1: enable; 0: off.
[4]	TH40_IE	MCPWM0_CNT1 equals MCPWM0_TH40 interrupt enable. 1: enable; 0: off.
[3]	TH31_IE	MCPWM0_CNT1 equals MCPWM0_TH31 interrupt enable. 1: enable; 0: off.
[2]	TH30_IE	MCPWM0_CNT1 equals MCPWM0_TH30 interrupt enable. 1: enable; 0: off.
[1]	T1_IE	T1 event, MCPWM0_CNT1 equals 0 interrupt enable. 1: enable; 0: off.
[0]	T0_IE	T0 event, MCPWM0_CNT1 equals MCPWM0_TH1 interrupt enable. 1: enable; 0: off.

14.2.37 MCPWM0_EIE

Write-protected register

Address: 0x4001 0C90

Reset value: 0x0

Table14-40 MCPWM0_EIE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						ZCS1_IE	ZCS0_IE	FAIL3_IE	FAIL2_IE	FAIL1_IE	FAIL0_IE				
						RW	RW	RW	RW	RW	RW				
						0	0	0	0	0	0				

Location	Bit name	Explain
[31:10]		Not used
[9]	ZCS1_IE	Time base 1 ZCS event interrupt source enable. 1: enable; 0: off.
[8]	ZCS0_IE	time base0 ZCS event Interrupt Source Enable. 1: enable; 0: off.
[7]	FAIL3_IE	FAIL3 interrupt source enable. 1: enable; 0: off.
[6]	FAIL2_IE	FAIL2 interrupt source enable. 1: enable; 0: off.
[5]	FAIL1_IE	FAIL1 interrupt source enable. 1: enable; 0: off.
[4]	FAIL0_IE	FAIL0 interrupt source enable. 1: enable; 0: off.

[3:0]		Not used

14.2.38 MCPWM0_EIF

Register without write protection

Address: 0x4001 0C94

Reset value: 0x0

Table14-41 MCPWM0_EIF Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						ZCS1_IF	ZCS1_IF	FAIL3_IF	FAIL2_IF	FAIL1_IF	FAIL0_IF				
						RW1C	RW1C	RW1C	RW1C	RW1C	RW1C				
						0	0	0	0	0	0				

Location	Bit name	Explain
[31:10]		Not used
[9]	ZCS1_IF	Time base 1 ZCS interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[8]	ZCS0_IF	Time base 0 ZCS interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[7]	FAIL3_IF	FAIL3 interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[6]	FAIL2_IF	FAIL2 interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[5]	FAIL1_IF	FAIL 1 interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[4]	FAIL0_IF	FAIL0 interrupt source event. 1: occur; 0: didn't happen. Write 1 to clear.
[3:0]		Not used

14.2.39 MCPWM0_RE

Write-protected register

Address: 0x4001 0C98

Reset value: 0x0

Table14-42 MCPWM0_RE Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR1_T1_RE1	TR1_T0_RE1	TR0_T1_RE1	TR0_T0_RE1	TMR3_RE1	TMR2_RE1	TMR1_RE1	TMR0_RE1	TR1_T1_RE0	TR1_T0_RE0	TR0_T1_RE0	TR0_T0_RE0	TMR3_RE0	TMR2_RE0	TMR1_RE0	TMR0_RE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	TR1_T1_RE1	Time Base 1 T1 event DMA Request 1 Enable. 1: Enable; 0: Off.
[14]	TR1_T0_RE1	Time Base 1 T0 event DMA Request 1 Enable. 1: Enable; 0: Off.
[13]	TR0_T1_RE1	Time base 0 T1 event DMA request 1 enabled. 1: Enable; 0: Off.
[12]	TR0_T0_RE1	Time base 0 T0 event DMA request 1 enabled. 1: Enable; 0: Off.
[11]	TMR3_RE1	The MCPWM counter hits TMR3 and DMA request 1 is enabled. 1: Enable; 0: Off.
[10]	TMR2_RE1	The MCPWM counter hits TMR2 and DMA request 1 is enabled. 1: Enable; 0: Off.
[9]	TMR1_RE1	The MCPWM counter hits TMR1 and DMA request 1 is enabled. 1: Enable; 0: Off.
[8]	TMR0_RE1	The MCPWM counter hits TMR0 and DMA request 1 is enabled. 1: Enable; 0: Off.
[7]	TR1_T1_RE0	Time Base 1 T1 event DMA Request 0 Enable. 1: Enable; 0: Off.
[6]	TR1_T0_RE0	Time base 1 T0 event DMA request 0 enabled. 1: Enable; 0: Off.
[5]	TR0_T1_RE0	Time base 0 T1 event DMA request 0 enabled. 1: Enable; 0: Off.
[4]	TR0_T0_RE0	Time Base 0 T0 event DMA Request 0 Enable. 1: Enable; 0: Off.
[3]	TMR3_RE0	The MCPWM counter hits TMR3 and DMA request 0 is enabled. 1: Enable; 0: Off.
[2]	TMR2_RE0	The MCPWM counter hits TMR2 and DMA request 0 is enabled. 1: Enable; 0: Off.
[1]	TMR1_RE0	The MCPWM counter hits TMR1 and DMA request 0 is enabled. 1: Enable; 0: Off.
[0]	TMR0_RE0	The MCPWM counter hits TMR0 and DMA request 0 is enabled. 1: Enable; 0: Off.

The MCPWM may generate two DMA request signals. If RE0 in the MCPWM0_RE register is enabled and the corresponding event flag is set, DMA request 0 is generated. If RE1 in the MCPWM0_RE register is enabled and the corresponding event flag is set, DMA request 1 is generated. The two request signals can respectively trigger two DMA channels to carry.

14.2.40 MCPWM0_PP

Write-protected register

Address: 0x4001 0C9C

Reset value: 0x0

Table14-43 MCP WM0_PP Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
										IO5_PPE	IO4_PPE	IO3_PPE	IO2_PPE	IO1_PPE	IO0_PPE
										RW	RW	RW	RW	RW	RW
										0	0	0	0	0	0

Location	Bit name	Explain

[31:6]		Not used
[5]	IO5_PPE	IO5 Push-Pull Mode Enable Signal. 1 : enable; 0 : close.
[4]	IO4_PPE	IO4 Push-Pull Mode Enable Signal. 1 : enable; 0 : close.
[3]	IO3_PPE	IO3 Push-Pull Mode Enable Signal. 1 : enable; 0 : close.
[2]	IO2_PPE	IO2 Push-Pull Mode Enable Signal. 1 : enable; 0 : close.
[1]	IO1_PPE	IO1 push-pull mode enable signal. 1 : enable; 0 : close.
[0]	IO0_PPE	IO0 push-pull mode enable signal. 1 : enable; 0 : close.

Push-pull mode enable signal. Depending on the mode of operation. An edge mode that turns on a push-pull mode of the edge mode; Align Center, turn on the push-pull mode of center alignment.

14.2.41 MCPWM0_IO01

Write-protected register

Address: 0x4001 0 CA0

Reset value: 0x0

Table14-44 MCPWM0_IO01 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1_WM	CH1_PN_SW	CH1_SCTRLP	CH1_SCTRLN	CH1_PS	CH1_NS	CH1_PP	CH1_NP	CH0_WM	CH0_PN_SW	CH0_SCTRLP	CH0_SCTRLN	CH0_PS	CH0_NS	CH0_PP	CH0_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	CH1_WM	CH 1 operating mode selection. 1: Edge mode; 0: Complementary mode.
[14]	CH1_PN_SW	The P and N channel outputs of CH1 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 1: interchange; 0: Not interchangeable.
[13]	CH1_SCTRLP	Value output to CH1 P channel when CH1_PS = 1.
[12]	CH1_SCTRLN	Value output to CH1 N channel when CH1_NS = 1.
[11]	CH1_PS	CH1 P source. 1: from CH1_SCTRLP; 0: MCPWM internal counter generation.
[10]	CH1_NS	CH1 N source. 1: from CH1_SCTRLN; 0: MCPWM internal counter generation.
[9]	CH1_PP	CH1 P polarity selection. 1: CH1 P signal inversion output; 0: normal output of CH1 P signal.
[8]	CH1_NP	CH1 N polarity selection. 1: CH1 N signal inversion output; 0: normal output of CH1 N signal.
[7]	CH0_WM	CH0 operating mode selection. 1: Edge mode; 0: Complementary mode.
[6]	CH0_PN_SW	The P and N channel outputs of CH0 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel.

		1: interchange; 0: Not interchangeable.
[5]	CH0_SCTRLP	Value output to the CH0 P channel when CH0_PS = 1.
[4]	CH0_SCTRLN	Value output to CH0 N channel when CH0_NS = 1.
[3]	CH0_PS	CH0 P source. 1: from CH0_SCTRLP; 0: The counter is generated in the MCPWM actual operation system.
[2]	CH0_NS	CH0 N source. 1: from CH0_SCTRLN; 0: The counter is generated in the MCPWM actual operation system.
[1]	CH0_PP	CH0 P polarity selection. 1: CH0 P signal inversion output; 0: CH0 P signal is output normally.
[0]	CH0_NP	**CH0 N polarity selection. 1: CH0 N signal inversion output; 0: normal output of CH0 N signal. The polarity selection follows the channel exchange. For example, if CH0 N is selected as the negated output and the channel exchange is selected at the same time, the exchanged CH0 N is still the negated output.**

14.2.42 MCPWM0_IO23

Write-protected register

Address: 0x4001 0 CA4

Reset value: 0x0

Table14-45 MCPWM0_IO23 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3_WM	CH3_PN_SW	CH3_SCTRLP	CH3_SCTRLN	CH3_PS	CH3_NS	CH3_PP	CH3_NP	CH2_WM	CH2_PN_SW	CH2_SCTRLP	CH2_SCTRLN	CH2_PS	CH2_NS	CH2_PP	CH2_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	CH3_WM	CH3 operating mode selection. 1: Edge mode; 0: Complementary mode.
[14]	CH3_PN_SW	The P and N channel outputs of CH3 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 1: interchange; 0: Not interchangeable.
[13]	CH3_SCTRLP	Value output to the CH3 P channel when CH3_PS = 1.
[12]	CH3_SCTRLN	When CH3_NS = 1, the value output to the CH3 N channel.
[11]	CH3_PS	CH3 P source. 1: from CH3_SCTRLP; 0: The counter is generated in the MCPWM actual operation system.
[10]	CH3_NS	CH3 N source. 1: from CH3_SCTRLN; 0: The counter is generated in the MCPWM actual operation system.
[9]	CH3_PP	CH3 P polarity selection. 1: CH3 P signal inversion output; 0: CH3 P signal is output normally.

[8]	CH3_NP	CH3 N polarity selection. 1: CH3 N signal inversion output; 0: CH3 N signal is output normally.
[7]	CH2_WM	CH2 operating mode selection. 1: Edge mode; 0: Complementary mode.
[6]	CH2_PN_SW	The P and N channel outputs of CH2 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 1: interchange; 0: Not interchangeable.
[5]	CH2_SCTRLP	Value output to the CH2 P channel when CH2_PS = 1.
[4]	CH2_SCTRLN	Value output to the CH2 N channel when CH2_NS = 1.
[3]	CH2_PS	CH2 P source. 1: from CH2_SCTRLP; 0: The counter is generated in the MCPWM actual operation system.
[2]	CH2_NS	CH2 N source. 1: from CH2_SCTRLN; 0: The counter is generated in the MCPWM actual operation system.
[1]	CH20_PP	Ch2 P polarity selection. 1: CH2 P signal inversion output; 0: CH2 P signal is output normally.
[0]	CH2_NP	CH2 N polarity selection. 1: CH2 N signal inversion output; 0: CH2 N signal is output normally. **The polarity selection follows the channel exchange. For example, CH2 N selects the negated output. If the channel exchange is selected at the same time, the exchanged CH2 N is still the negated output.**

14.2.43 MCPWM0_IO45

Write-protected register

Address: 0x4001 0CA8

Reset value: 0x0

Table14-46 MCPWM0_IO45 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH5_WM	CH5_PN_SW	CH5_SCTRLP	CH5_SCTRLN	CH5_PS	CH5_NS	CH5_PP	CH5_NP	CH4_WM	CH4_PN_SW	CH4_SCTRLP	CH4_SCTRLN	CH4_PS	CH4_NS	CH4_PP	CH4_NP
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	CH5_WM	CH5 operating mode selection. 1: Edge mode; 0: Complementary mode.
[14]	CH5_PN_SW	The P and N channel outputs of CH5 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 1: interchange; 0: Not interchangeable.
[13]	CH5_SCTRLP	Value output to CH5 P channel when CH5_PS = 1.
[12]	CH5_SCTRLN	Value output to CH5 N channel when CH5_NS = 1.
[11]	CH5_PS	CH5 P source.

		1: from CH5_SCTRLP; 0: The counter is generated in the MCPWM actual operation system.
[10]	CH5_NS	CH5 N source. 1: from CH5_SCTRLN; 0: The counter is generated in the MCPWM actual operation system.
[9]	CH5_PP	CH5 P polarity selection. 1: CH5 P signal inversion output; 0: normal output of CH5 P signal.
[8]	CH5_NP	CH5 N polarity selection. 1: CH5 N signal inversion output; 0: normal output of CH5 N signal.
[7]	CH4_WM	CH4 operating mode selection. 1: Edge mode; 0: Complementary mode.
[6]	CH4_PN_SW	The P and N channel outputs of CH4 are selected interchangeably. That is, the P channel signal is finally output from the N channel, and the N channel signal is finally output from the P channel. 1: interchange; 0: Not interchangeable.
[5]	CH4_SCTRLP	Value output to the CH4 P channel when CH4_PS = 1.
[4]	CH4_SCTRLN	The value output to the CH4 N channel when CH4_NS = 1.
[3]	CH4_PS	CH4 P source. 1: from CH4_SCTRLP; 0: The counter is generated in the MCPWM actual operation system.
[2]	CH4_NS	CH4 N source. 1: from CH4_SCTRLN; 0: The counter is generated in the MCPWM actual operation system.
[1]	CH40_PP	CH4 P polarity selection. 1: CH4 P signal inversion output; 0: CH4 P signal is output normally.
[0]	CH4_NP	CH4 N polarity selection. 1: CH4 N signal inversion output; 0: CH4 N signal is output normally. **The polarity selection follows the channel exchange. For example, CH4 N selects the negated output. If the channel exchange is selected at the same time, the exchanged CH4 N is still the negated output.**

14.2.44 MCPWM0_IF1

Register without write protection

Address: 0x4001 0C8C

Reset value: 0x0

Table14-47 MCPWM0_IF1 MCPWM Time Base 1 Interrupt Flag Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	UP_IF	TMR3_IF	TMR2_IF					TH51_IF	TH50_IF	TH41_IF	TH40_IF	TH31_IF	TH30_IF	T1_IF	T0_IF
	RW1C	RW1C	RW1C					RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
	0	0	0					0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:15]		Not used
[14]	UP_IF	Time base 1 update event The MCPWM0_TH1/MCPWM0_TH30 to MCPWM0_TH51/MCPWM_TMR

		registers are updated to the interrupt event of the actual running system of the MCPWM. 1: occur; 0: didn't happen. Write 1 to clear.
[13]	TMR3_IF	MCPWM0_CNT1 equals the MCPWM0_TMR3 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[12]	TMR2_IF	MCPWM0_CNT1 equals the MCPWM0_TMR2 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[11:8]		Not used
[7]	TH51_IF	MCPWM0_CNT1 equals MCPWM0_TH51 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[6]	TH50_IF	MCPWM0_CNT1 equals MCPWM0_TH50 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[5]	TH41_IF	MCPWM0_CNT1 equals the MCPWM0_TH41 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[4]	TH40_IF	MCPWM0_CNT1 equals MCPWM0_TH40 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[3]	TH31_IF	MCPWM0_CNT1 equals the MCPWM0_TH31 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[2]	TH30_IF	MCPWM0_CNT1 equals the MCPWM0_TH30 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[1]	T1_IF	T1 event, MCPWM0_CNT1 equals 0 interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.
[0]	T0_IF	T0 event: MCPWM0_CNT1 equals MCPWM0_TH interrupt event. 1: occur; 0: didn't happen. Write 1 to clear.

14.2.45 MCPWM0_FAIL012

Write-protected register

Address: 0x4001 0CB0

Reset value: 0x0

Table14-48 MCPWM0_FAIL012 configuration register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_0CAP				FAIL1_SEL		FAIL0_SEL		HALT_PRT	MOE	FAIL1_EN	FAIL0_EN	FAIL1_POL	FAIL0_POL		
RW				RW		RW		RW	RW	RW	RW	RW	RW		
0				0		0		0	0	0	0	0	0		

Location	Bit name	Explain
[31:16]		Not used
[15]	FAIL_0CAP	When a fail0/1 event occurs, the MCPWM0_CNT0 value is stored in MCPWM0_FCNT
[14:12]		Not used
[11:10]	FAIL1_SEL	FAIL 1 Source Selection 0: MCPWM0_BKIN1 1: original output of comparator 1

		2: CLUOUT2 3: CLUOUT3
[9:8]	FAIL0_SEL	FAIL0 source selection 0: MCPWM0_BKIN0 1: original output of comparator 0 2: CLUOUT0 3: CLUOUT1
[7]	HALT_PRT	The MCU enters the HALT state and the MCPWM output value is selected. 1: Normal output; 0: Force MCPWM to output the protection value.
[6]	MOE	MOE controls the MCPWM CH0/CH1/CH2 channel P and N output values. 1: output the normal signal generated by MCPWM 0: Output the default value of CHxN_DEFAULT and CHxP_DEFAULT set by the MCPWM0_CH_DEF. This default value is not controlled by polarity/channel selection, etc. A change in any of the MCPWM0_EIF. FAIL1_if and the MCPWM0_EIF. FAIL0_if will trigger the MOE to become 0 and output the default value.
[5]	FAIL1_EN	FAIL1 input enable. 1: Enable; 0: Off.
[4]	FAIL0_EN	FAIL0 input enable. 1: Enable; 0: Off.
[3]	FAIL1_POL	FAIL1 polarity selection. 1: The signal polarity is reversed and input, and the signal input low is the effective level; 0: The signal polarity is input normally, and the signal input high is the effective level.
[2]	FAIL0_POL	FAIL0 polarity selection. 1: The signal polarity is reversed and input, and the signal input low is the effective level; 0: The signal polarity is input normally, and the signal input high is the effective level.
[1:0]		Not used

The FAIL0/1 signal is used for MCPWM channels 0/1/2 operating on time base 0. When FAIL0/1 is active, the counter MCPWM0_CNT0 corresponding to MCPWM time base 0 is suspended and the corresponding error interrupt flag MCPWM0_EIF will be set.

14.2.46 MCPWM0_FAIL345

Write-protected register

Address: 0x4001 0CB4

Reset value: 0x0

Table14-49 MCPWM0_FAIL345 Configuration Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAIL_1CAP				FAIL3_SEL		FAIL2_SEL		HALT_PRT	MOE	FAIL1_EN	FAIL0_EN	FAIL3_POL	FAIL2_POL		
RW				RW		RW		RW	RW	RW	RW	RW	RW		
0				0		0		0	0	0	0	0	0		

Location	Bit name	Explain
[31:16]		Not used
[15]	FAIL_1CAP	When a fail2/3 event occurs, the MCPWM0_CNT1 value is stored in MCPWM0_FCNT
[14:12]		Not used
[11:10]	FAIL3_SEL	FAIL3 source selection 0: MCPWM0_BKIN1 1: Result of comparator 1 2: CLUOUT2 3: CLUOUT3
[9:8]	FAIL2_SEL	FAIL2 source selection 0: MCPWM0_BKIN0 1: Result of comparator 0 2: CLUOUT0 3: CLUOUT1
[7]	HALT_PRT	The MCU enters the HALT state and the MCPWM output value is selected. 1: Normal output; 0: Force MCPWM to output the protection value.
[6]	MOE	MOE controls the MCPWM CH3/CH4/CH5 channel P and N output values. 1: output the normal signal generated by MCPWM 0: Output CH3N_DEFAULT and CH3P_DEFAULT defaults, which are not controlled by polarity/channel selection, etc. A change in any of the MCPWM0_EIF. FAIL2_if and the MCPWM0_EIF. FAIL3_if will trigger the MOE to become 0 and output the default value.
[5]	FAIL3_EN	FAIL3 input enable. 1: Enable; 0: Off.
[4]	FAIL2_EN	FAIL2 input enable. 1: Enable; 0: Off.
[3]	FAIL3_POL	FAIL3 polarity selection. 1: The signal polarity is reversed and input, and the signal input low is the effective level; 0: The signal polarity is input normally, and the signal input high is the effective level.
[2]	FAIL2_POL	FAIL2 polarity selection. 1: The signal polarity is reversed and input, and the signal input low is the effective level; 0: The signal polarity is input normally, and the signal input high is the effective level.
[1:0]		Not used

The FAIL2/3 signals are for MCPWM channels 3/4/5 operating on time base 1. When FAIL2/3 is valid, the counter MCPWM0_CNT1 corresponding to MCPWM time base 1 will be suspended, and the corresponding error interrupt flag MCPWM0_EIF will be set.

MCPWM0_FAIL can be used to set the emergency shutdown event and block the signal output of MCPWM. There are two FAIL0 and FAIL1 E-STOP events for channels 0/1/2 and two FAIL2 and FAIL3 E-STOP events for channels 3/4/5. A total of 3 signal sources are available for each FAIL signal, comparator output , MCPWM0_BKIN and CLUOUT.

The input signal to the FAIL can be digitally filtered, and the first division of the filtered clock is set by the MCPWM0_TCLK.CLK_DIV register, the second division of the filtered clock is set by the MCPWM0_FLT.FLT_CLKDIV[7:0] register.

Finally, the filter circuit will filter the FAIL signal with 16 filtering clocks, that is, only the signal stability time exceeds 16 filtering cycles can pass the filter. That is, filtering width = filtering clock period *16.

Refer to FAIL Signal Processing for more information.

14.2.47 MCPWM0_CH_DEF

Write-protected register

Address: 0x4001 0CB8

Reset value: 0x0

Table14-50 MCPWM 0_CH_DEF Short-Circuit Protection Channel Output Value Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CH5P_DEFAULT	CH5N_DEFAULT	CH4P_DEFAULT	CH4N_DEFAULT	CH3P_DEFAULT	CH3N_DEFAULT	CH2P_DEFAULT	CH2N_DEFAULT	CH1P_DEFAULT	CH1N_DEFAULT	CH0P_DEFAULT	CH0N_DEFAULT
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:12]		Not used
[11]	CH5P_DEFAULT	CH5_P Channel Default
[10]	CH5N_DEFAULT	CH5_N Channel Default
[9]	CH4P_DEFAULT	CH4_P Channel Default
[8]	CH4N_DEFAULT	CH4_N Channel Default
[7]	CH3P_DEFAULT	CH3 P Channel Default
[6]	CH3N_DEFAULT	**CH3 N channel default. When a FAIL event occurs or MOE is 0, the corresponding channel outputs a default level. The default level output is not affected by the BIT0, BIT1, BIT8, BIT9, and BIT6 of MCPWM0_IO23 channel exchange and polarity control, and directly controls the channel output.**
[5]	CH2P_DEFAULT	CH2_P Channel Default
[4]	CH2N_DEFAULT	CH2_N Channel Default
[3]	CH1P_DEFAULT	CH1_P Channel Default
[2]	CH1N_DEFAULT	CH1_N Channel Default
[1]	CH0P_DEFAULT	CH0_P Channel Default
[0]	CH0N_DEFAULT	**CH0_N channel default. When a FAIL event occurs or MOE is 0, the corresponding channel outputs a default level. The default level output is not affected by the BIT0, BIT1, BIT8, BIT9, and BIT6 of MCPWM0_IO01 and MCPWM0_IO23 channel exchange and polarity control, and directly controls the channel output.**

14.2.48 MCPWM0_CH_MSK

Write-protected register

Address: 0x4001 0 CBC

Reset value: 0x0

Table14-51 MCPWM0_CH_MSK Channel Mask Bit Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CH5P_FAIL_EN	CH5N_FAIL_EN	CH4P_FAIL_EN	CH4N_FAIL_EN	CH3P_FAIL_EN	CH3N_FAIL_EN	CH2P_FAIL_EN	CH2N_FAIL_EN	CH1P_FAIL_EN	CH1N_FAIL_EN	CH0P_FAIL_EN	CH0N_FAIL_EN
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				1	1	1	1	1	1	1	1	1	1	1	1

Location	Bit name	Explain
[31:12]		Not used
[11]	CH5P_FAIL_EN	CH5_P channel FAIL event enable, active high, on by default
[10]	CH5N_FAIL_EN	CH5_N Channel FAIL event Enable, Active High, On by Default
[9]	CH4P_FAIL_EN	CH4_P channel FAIL event enable, active high, on by default
[8]	CH4N_FAIL_EN	CH4_N channel FAIL event enable, active high, on by default
[7]	CH3P_FAIL_EN	CH3_P channel FAIL event enable, active high, on by default
[6]	CH3N_FAIL_EN	CH3_N channel FAIL event enable, active high, on by default
[5]	CH2P_FAIL_EN	CH2_P channel FAIL event enable, active high, on by default
[4]	CH2N_FAIL_EN	CH2_N channel FAIL event enable, active high, on by default
[3]	CH1P_FAIL_EN	CH1_P channel FAIL event enable, active high, on by default
[2]	CH1N_FAIL_EN	CH1_N channel FAIL event enable, active high, on by default
[1]	CH0P_FAIL_EN	CH0_P channel FAIL event enable, active high, on by default
[0]	CH0N_FAIL_EN	CH0_N channel FAIL event enabled, 1: when FAIL event occurs or MCPWM0_FAIL012.MOE = 0, CH0_N channel level output is the default value, 0: when FAIL event occurs or MCPWM0_FAIL012.MOE = 0, CH0_n channel level is not affected, It is still controlled by the MCPWM internal hardware. FAIL enable on by default

14.2.49 MCPWM0_PRT

Register without write protection

Address: 0x4001 0CC0

Reset value: 0x0

Table14-52 MCP WM0_PRT Register

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

Location	Bit name	Explain
[31:16]		Not used
[15:0]	PRT	Write 0xDEAD to remove the write protection of the MCPWM register.

		Write another value to put the MCPWM register into write protection. The readout of this register is always 0.

14.2.50 MCPWM0_SWAP

Move to the PWM_SWAP register in the GPIO module.

14.2.51 MCPWM0_STT_HYST

Write-protected register

Address: 0x4001 0CC8

Reset value: 0x0

Table14-53 MCPWM 0_STT_HYST dwell delay register

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

Location	Bit name	Explain
[31:12]		Not used
[11:0]	STT_HYST	The state switching delay only works when the MCPWM0_TCLK. CMP_CTRL_CNT0 = 1 or the MCPWM0_TCLK.CMP_CTRL_CNT1 = 1. 12-bit unsigned number

14.2.52 MCPWM0_ZCS_DELAY

Write-protected register

Address: 0x4001_0CCC

Reset value: 0x0

Table14-54 MCPWM0_ZCS_DELAY Status Dwell Time Register

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

Location	Bit name	Explain
[31:15]		Not used
[14:0]	ZCS_DELAY	The ZCS state dwell time only works when the MCPWM0_TCLK. CMP_CTRL_CNT0 = 1 or the MCPWM0_TCLK.CMP_CTRL_CNT1 = 1. 15bit unsigned number. The ZCS_DELAY must be greater than the STT_HYST.

15 GPIO

15.1 Overview

LSK32MC07x series chips integrate 4 groups of GPIOs, each group contains 16 GPIOs. Some GPIOs can be used as a sleep mode wakeup source for the system. Some GPIOs can be used as external interrupt source inputs.

15.1.1 Functional block diagram

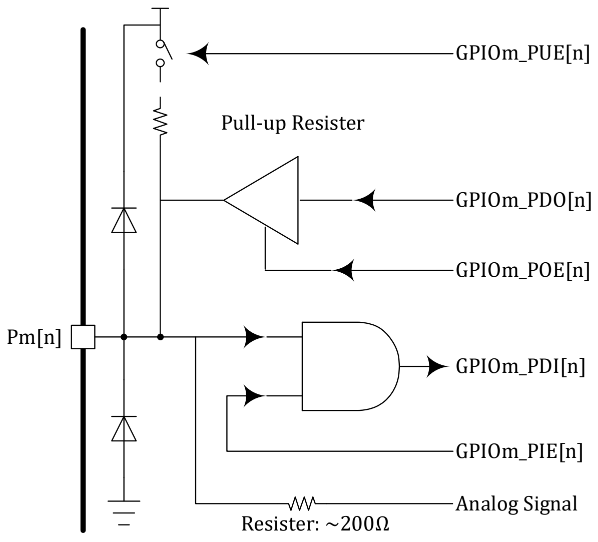


Figure15-1 GPIO Functional Block Diagram

As shown in the above figure, Pm [n] is the chip PAD, m can be 0 to 3, representing any one of the four groups of GPIOs, and n can be 0 to 15, representing one of the 16 GPIOs in a group. The analog signal is connected directly to the PAD through a series resistor of approximately 200 Ω. When the output enable GPIOm_POE [n] = 0, the buffer outputs a high impedance state, otherwise the buffer outputs the same level as the GPIOm_PDO [n]. When the GPIOm_PIE [n] = 0, the GPIOm_PDI [n] is always 0. When the GPIOm_PIE [n] = 1, that is, the input enable is turned on, the level of the GPIOm_PDI [n] is the same as that of Pm [n]. Chip PADs can be configured with pull-up. The P0 [2] pin is multiplexed as an external reset pin. The RSTN pull-up resistor is 300 kΩ. Note that not all PADs are equipped with pull-up resistors. For details about which pads have pull-up resistance resources, see chapter 15.2.6. PADs without pull-up resistors can also be configured with the GPIOm_PUE [n] register, but for no practical purpose.

15.1.2 Product features

- 64-channel GPIO
- Push-pull open-drain mode configurable
- Some GPIOs support external interrupts

- Some GPIOs can be used as sleep wakeup sources
- Some GPIOs support input signal filtering

Specific GPIO that support external interrupt, please refer to the 15.2.15.4

Specific GPIO that can be used as sleep wakeup, please refer to the sources21.6.5

Specific GPIO that can be used for input signal filtering,please refer to the sources15.3.2

The GPIO can be configured for up to 13 external functions. Refer to the corresponding device datasheet for the distribution of GPIO functions.

Table 15-1 GPIO secondary function

GPIO secondary function value	Functional description
0	Analog function. When the GPIO is used as an analog function, it is necessary to turn off the output enable and pull-up enable of the corresponding IO. The IO output signal or the on-chip pull-up resistor interferes with the analog signal.
1	CMP_OUT/CLKO, Analog Comparator Direct Output/Clock Output
2	HALL, HALL signal input
3	MCPWM, MCPWM channel output or shutdown signal input
4	UART, serial port RXD or TXD
5	SPI, SPI Clock, Chip Select, Data In, Data Out
6	I2C, I2C clock, I2C data
7	TIMER0/1, Channel 0/1 of TIMER0/1, input for capture mode or external clock source, output for compare mode, stop signal input of TIMER0
8	TIMER2/3, channel 0/1 of TIMER2/3, input for capture mode or external clock source, output for comparison mode, Z-axis clear signal input of QEP0/1
9	ADC_TRIGGER0/1, ADC 0/1 sampling trigger signal output, the ADC_TRIGGER signal is inverted once every ADC sampling trigger
10	CAN, CAN RX or TX
11	SIF, 1-wire serial communication interface
12	CL, Configurable Logic Array UNIT Output

15.2 Register

15.2.1 Address assignment

The GPIO 0 block's base address on the chip is 0x4001_0D00.

The GPIO 1 block's base address on the chip is 0x4001_0D40.

The GPIO 2 block's base address on the chip is 0x4001_0D80.

The GPIO 3 block's base address on the chip is 0x4001_0 DC0.

The register definitions for GPIO 0/1/2/3 are identical, differing only by the base address.

Table15-2 List of GPIOx registers

Name	Offset address	Explain
GPIOx_PIE	0x00	GPIO X Input Enable
GPIOx_POE	0x04	GPIO X output enabled
GPIOx_PDI	0x08	GPIO X input data
GPIOx_PDO	0x0C	GPIO X Output Data
GPIOx_PUE	0x10	GPIO X pull-up enabled
GPIOx_PODE	0x18	GPIO X open-drain enabled
GPIOx_PFLT	0x1C	GPIO X filtering enabled
GPIOx_F3210	0x20	GPIO X [3:0] function selection
GPIOx_F7654	0x24	GPIO X [7:4] function selection
GPIOx_FBA98	0x28	GPIO X [11:8] function selection
GPIOx_FFEDC	0x2C	GPIO X [15:12] function selection
GPIOx_BSRR	0x30	GPIO X Bit Operation Register
GPIOx_BRR	0x34	GPIO X Bit Clear Register

The base address of the GPIO interrupt block is 0x4001_0E00.

Table15-3 GPIO Interrupt Module Register List

Name	Offset address	Explain
EXTI_CR0	0x00	External Interrupt Configuration Register 0
EXTI_CR1	0x04	External Interrupt Configuration Register 1
EXTI_IE	0x08	GPIO Interrupt/DMA Enable
EXTI_IF	0x0C	GPIO Interrupt Flag
CLKO_SEL	0x10	Output clock selection signal
PWM_SWAP	0x14	MCPWM channel rearrangement

15.2.2 GPIOx_PIE

The addresses are: 0x4001_0 D00, 0x4001_0D40, 0x4001_0D80, 0x4001 0 DC0

Reset value: 0x0

Table15-4 GPIOx Input Enable Register GPIOx_PIE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIE15	PIE14	PIE13	PIE12	PIE11	PIE10	PIE9	PIE8	PIE7	PIE6	PIE5	PIE4	PIE3	PIE2	PIE1	PIE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	PIE15	GPIO X [15]/Px [15] input enable
[14]	PIE14	GPIO X [14]/Px [14] input enabled
[13]	PIE13	GPIO X [13]/Px [13] input enabled
[12]	PIE12	GPIO X [12]/Px [12] input enabled
[11]	PIE11	GPIO X [11]/Px [11] input enabled
[10]	PIE10	GPIO X [10]/Px [10] input enabled
[9]	PIE9	GPIO X [9]/Px [9] input enabled
[8]	PIE8	GPIO X [8]/Px [8] input enabled

[7]	PIE7	GPIO X [7]/Px [7] input enabled
[6]	PIE6	GPIO X [6]/Px [6] input enabled
[5]	PIE5	GPIO X [5]/Px [5] input enabled
[4]	PIE4	GPIO X [4]/Px [4] input enabled
[3]	PIE3	GPIO X [3]/Px [3] input enabled
[2]	PIE2	GPIO X [2]/Px [2] input enabled
[1]	PIE1	GPIO X [1]/Px [1] input enabled
[0]	PIE0	GPIO X [0]/Px [0] input enabled

15.2.3 GPIOx_POE

The addresses are: 0x4001_0 D04, 0x4001_0D44, 0x4001_0D84, and 0x400 1_0 DC4

Reset value: 0x0

Table15-5 GPIOx Output Enable Register GPIOx_POE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POE15	POE14	POE13	POE12	POE11	POE10	POE9	POE8	POE7	POE6	POE5	POE4	POE3	POE2	POE1	POE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	POE15	GPIO X [15]/Px [15] output enable
[14]	POE14	GPIO X [14]/Px [14] output enabled
[13]	POE13	GPIO X [13]/Px [13] output enabled
[12]	POE12	GPIO X [12]/Px [12] output enabled
[11]	POE11	GPIO X [11]/Px [11] output enabled
[10]	POE10	GPIO X [10]/Px [10] output enabled
[9]	POE9	GPIO X [9]/Px [9] output enabled
[8]	POE8	GPIO X [8]/Px [8] output enabled
[7]	POE7	GPIO X [7]/Px [7] output enabled
[6]	POE6	GPIO X [6]/Px [6] output enabled
[5]	POE5	GPIO X [5]/Px [5] output enabled
[4]	POE4	GPIO X [4]/Px [4] output enabled
[3]	POE3	GPIO X [3]/Px [3] output enabled
[2]	POE2	GPIO X [2]/Px [2] output enabled
[1]	POE1	GPIO X [1]/Px [1] output enabled
[0]	POE0	GPIO X [0]/Px [0] output enabled

15.2.4 GPIOx_PDI

The addresses are: 0x4001_0 D08, 0x4001_0D48, 0x4001_0D88, and 0x400 1_0 DC8

Reset value: 0x0

Table15-6 GPIOx Input Data Register GPIOx_PDI.

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

PDI
RO
0

Location	Bit name	Explain
[31:16]		Not used
[15:0]	PDI	GPIO X input data

When GPIOx PIE=0, GPIOx PDI read-back is 0.

15.2.5 GPIOx_PDO

The addresses are: 0x4001_0 D0C, 0x4001_0 D4C, 0x400_0 D8C, 0x400_1 0 DCC

Reset value: 0x0

Table15-7 GPIOx Output Data Register GPIOx_PDO.

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

Location	Bit name	Explain
[31:16]		Not used
[15:0]	PDO	GPIO X Output Data

15.2.6 GPIOx_PUE

The addresses are: 0x4001_0D10, 0x4001_0D50, 0x4001 — 0D90, 0x4001 — 0DD0

Reset value: 0x0

Table15-8 GPIOx pull-up enable register GPIOx_PUE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUE15	PUE14	PUE13	PUE12	PUE11	PUE10	PUE9	PUE8	PUE7	PUE6	PUE5	PUE4	PUE3	PUE2	PUE1	PUE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	PUE15	GPIO X [15]/Px [15] pull-up enabled
[14]	PUE14	GPIO X [14]/Px [14] pull-up enabled
[13]	PUE13	GPIO X [13]/Px [13] pull-up enabled
[12]	PUE12	GPIO X [12]/Px [12] pull-up enabled

[11]	PUE11	GPIO X [11]/Px [11] pull-up enabled
[10]	PUE10	GPIO X [10]/Px [10] pull-up enabled
[9]	PUE9	GPIO X [9]/Px [9] pull-up enabled
[8]	PUE8	GPIO X [8]/Px [8] pull-up enabled
[7]	PUE7	GPIO X [7]/Px [7] pull-up enabled
[6]	PUE6	GPIO X [6]/Px [6] pull-up enabled
[5]	PUE5	GPIO X [5]/Px [5] pull-up enabled
[4]	PUE4	GPIO X [4]/Px [4] pull-up enabled
[3]	PUE3	GPIO X [3]/Px [3] pull-up enabled
[2]	PUE2	GPIO X [2]/Px [2] pull-up enabled
[1]	PUE1	GPIO X [1]/Px [1] pull-up enabled
[0]	PUE0	GPIO X [0]/Px [0] pull-up enabled

Note: Not all IOs have the pull-up function. Please refer to which IOs have the pull-up function.15.3.1。 The PUE registers for IOs without pull-up are also not implemented, so writing a 1 to these locations is invalid and reading back is always a 0.

15.2.7 GPIOx_PODE

The addresses are: 0x4001_0D18, 0x4001_0D58, 0x400_0D98, 0x4001 0DD8

Reset value: 0x0

Table15-9 GPIOx open-drain enable register GPIOx_PODE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PODE15	PODE14	PODE13	PODE12	PODE11	PODE10	PODE9	PODE8	PODE7	PODE6	PODE5	PODE4	PODE3	PODE2	PODE1	PODE0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	PODE15	GPIO X [15]/Px [15] open-drain enabled
[14]	PODE14	GPIO X [14]/Px [14] open-drain enabled
[13]	PODE13	GPIO X [13]/Px [13] open-drain enabled
[12]	PODE12	GPIO X [12]/Px [12] open-drain enabled
[11]	PODE11	GPIO X [11]/Px [11] open-drain enabled
[10]	PODE10	GPIO X [10]/Px [10] open-drain enabled
[9]	PODE9	GPIO X [9]/Px [9] open-drain enabled
[8]	PODE8	GPIO X [8]/Px [8] open-drain enabled
[7]	PODE7	GPIO X [7]/Px [7] open-drain enabled
[6]	PODE6	GPIO X [6]/Px [6] open-drain enabled
[5]	PODE5	GPIO X [5]/Px [5] open-drain enabled
[4]	PODE4	GPIO X [4]/Px [4] open-drain enabled
[3]	PODE3	GPIO X [3]/Px [3] open-drain enabled
[2]	PODE2	GPIO X [2]/Px [2] open-drain enabled
[1]	PODE1	GPIO X [1]/Px [1] open-drain enabled
[0]	PODE0	GPIO X [0]/Px [0] open-drain enabled

15.2.8 GPIOx_PFLT

The addresses are: 0x4001_0 D1C, 0x4001_0 D5C, 0x400 1_0D9C, 0x400 1_0DDC

Reset value: 0x0

Table15-10 GPIOx Filter Register GPIOx_PFLT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFLT 15	PFLT 14	PFLT 13	PFLT 12	PFLT 11	PFLT 10	PFLT 9	PFLT 8	PFLT 7	PFLT 6	PFLT 5	PFLT 4	PFLT 3	PFLT 2	PFLT1	PFLT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	PFLT15	GPIO X [15]/Px [15] filtering enabled
[14]	PFLT14	GPIO X [14]/Px [14] filtering enabled
[13]	PFLT13	GPIO X [13]/Px [13] filtering enabled
[12]	PFLT12	GPIO X [12]/Px [12] filtering enabled
[11]	PFLT11	GPIO X [11]/Px [11] filtering enabled
[10]	PFLT10	GPIO X [10]/Px [10] filtering enabled
[9]	PFLT9	GPIO X [9]/Px [9] filtering enabled
[8]	PFLT8	GPIO X [8]/Px [8] filtering enabled
[7]	PFLT7	GPIO X [7]/Px [7] filtering enabled
[6]	PFLT6	GPIO X [6]/Px [6] filtering enabled
[5]	PFLT 5	GPIO X [5]/Px [5] filtering enabled
[4]	PFLT 4	GPIO X [4]/Px [4] filtering enabled
[3]	PFLT 3	GPIO X [3]/Px [3] filtering enabled
[2]	PFLT 2	GPIO X [2]/Px [2] filtering enabled
[1]	PFLT1	GPIO X [1]/Px [1] filtering enabled
[0]	PFLT0	GPIO X [0]/Px [0] filtering enabled

Only some GPIOs support input signal filtering. Refer to 15.3.2 for details on which GPIOs support filtering.

15.2.9 GPIOx_F3210

The addresses are: 0x4001_0D20, 0x4001_0D60, 0x4001 0DA0, 0x4001 0DE0

Reset value: 0x0

Table15-11 GPIOx function select register GPIOx_F3210

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F3				F2				F1				F0			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Explain
[31:16]		Not used
[15:12]	F3	GPIO X [3]/Px [3] function selection
[11:8]	F2	GPIO X [2]/Px [2] function selection
[7:4]	F1	GPIO X [1]/Px [1] function selection
[3:0]	F0	GPIO X [0]/Px [0] function selection

The second GPIO function of F3/F2/F1/F0 can be set as shown in Table 15-1, the GPIO functions are sparsely mapped, that is, each GPIO can only be configured to use part of the second function. If the distribution of GPIO functions is required, please refer to the corresponding device datasheet. The following GPIOx_F7654, GPIOx_FBA98, and GPIOx_FFEDC are the same as those of GPIOx_F3210.

15.2.10 GPIOx_F7654

The addresses are: 0x4001_0D24, 0x4001_0D64, 0x4001_0 DA4, 0x4001_0 DE4

Reset value: 0x0

Table15-12 GPIOx Function Select Register GPIOx_F7654

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F7				F6				F5				F4			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Explain
[31:16]		Not used
[15:12]	F7	GPIO X [7]/Px [7] function selection
[11:8]	F6	GPIO X [6]/Px [6] function selection
[7:4]	F5	GPIO X [5]/Px [5] function selection
[3:0]	F4	GPIO X [4]/Px [4] function selection

15.2.11 GPIOx_FBA98

The addresses are: 0x4001_0D28, 0x4001_0D68, 0x400l_0DA8, 0x400l_0DE8

Reset value: 0x0

Table15-13 GPIOx function select register GPIOx_FBA98

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F11				F10				F9				F8			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit	Explain

	name	
[31:16]		Not used
[15:12]	F11	GPIO X [11]/Px [11] function selection
[11:8]	F10	GPIO X [10]/Px [10] function selection
[7:4]	F9	GPIO X [9]/Px [9] function selection
[3:0]	F8	GPIO X [8]/Px [8] function selection

15.2.12 GPIOx_FFEDC

The addresses are: 0x4001_0 D2C, 0x4001_0 D6C, 0x400 1_0 DAC, and 0x400 1_0DEC

Reset value: 0x0

Table15-14 GPIOx function select register GPIOx_FFEDC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F15				F14				F13				F12			
RW				RW				RW				RW			
0				0				0				0			

Location	Bit name	Explain
[31:16]		Not used
[15:12]	F15	GPIO X [15]/Px [15] function selection
[11:8]	F14	GPIO X [14]/Px [14] function selection
[7:4]	F13	GPIO X [13]/Px [13] function selection
[3:0]	F12	GPIO X [12]/Px [12] function selection

For the detailed list of GPIO function reuse, please refer to the corresponding product DATASHEET pin distribution section.

15.2.13 GPIOx_BSRR

The addresses are: 0x4001_0D30, 0x4001_0D70, 0x4001_0 DB0, 0x4001 — 0DF0

Reset value: 0x0

Table15-15 GPIOx Bits Operation Register GPIOx_BSRR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31]	CLR15	Writing 1 clears GPIO X [15]. Writing 0 has no effect.
[30]	CLR14	Writing 1 clears GPIO X [14]. Writing 0 has no effect.
[29]	CLR13	Writing 1 clears GPIO X [13]. Writing 0 has no effect.
[28]	CLR12	Writing 1 clears GPIO X [12]. Writing 0 has no effect.
[27]	CLR11	Writing 1 clears GPIO X [11]. Writing 0 has no effect.
[26]	CLR10	Writing 1 clears GPIO X [10]. Writing 0 has no effect.
[25]	CLR9	Writing 1 clears GPIO X [9]. Writing 0 has no effect.
[24]	CLR8	Writing 1 clears GPIO X [8]. Writing 0 has no effect.
[23]	CLR7	Writing 1 clears GPIO X [7]. Writing 0 has no effect.
[22]	CLR6	Writing 1 clears GPIO X [6]. Writing 0 has no effect.
[21]	CLR5	Writing 1 clears GPIO X [5]. Writing 0 has no effect.
[20]	CLR4	Writing 1 clears GPIO X [4]. Writing 0 has no effect.
[19]	CLR3	Writing 1 clears GPIO X [3]. Writing 0 has no effect.
[18]	CLR2	Writing 1 clears GPIO X [2]. Writing 0 has no effect.
[17]	CLR1	Writing 1 clears GPIO X [1]. Writing 0 has no effect.
[16]	CLR0	Writing 1 clears GPIO X [0]. Writing 0 has no effect.
[15]	SET15	Writing 1 sets GPIO X [15] to 1. Writing 0 has no effect.
[14]	SET14	Writing 1 sets GPIO X [14] to 1. Writing 0 has no effect.
[13]	SET13	Writing 1 sets GPIO X [13] to 1. Writing 0 has no effect.
[12]	SET12	Writing 1 sets GPIO X [12] to 1. Writing 0 has no effect.
[11]	SET11	Writing 1 sets GPIO X [11] to 1. Writing 0 has no effect.
[10]	SET10	Writing 1 sets GPIO X [10] to 1. Writing 0 has no effect.
[9]	SET9	Writing 1 sets GPIO X [9] to 1. Writing 0 has no effect.
[8]	SET8	Writing 1 sets GPIO X [8] to 1. Writing 0 has no effect.
[7]	SET7	Writing 1 sets GPIO X [7] to 1. Writing 0 has no effect.
[6]	SET6	Writing 1 sets GPIO X [6] to 1. Writing 0 has no effect.
[5]	SET5	Writing 1 sets GPIO X [5] to 1. Writing 0 has no effect.
[4]	SET4	Writing 1 sets GPIO X [4] to 1. Writing 0 has no effect.
[3]	SET3	Writing 1 sets GPIO X [3] to 1. Writing 0 has no effect.
[2]	SET2	Writing 1 sets GPIO X [2] to 1. Writing 0 has no effect.
[1]	SET1	Writing 1 sets GPIO X [1] to 1. Writing 0 has no effect.
[0]	SET0	Writing 1 sets GPIO X [0] to 1. Writing 0 has no effect.

This bit is cleared if the same GPIO bit is both set and cleared by the upper 16 bits and the lower 16 bits of the BSRR.

15.2.14 GPIOx_BRR

The addresses are: 0x4001_0D34, 0x4001_0D74, 0x4001_0 DB4, 0x400 1_0 DF4

Reset value: 0x0

Table15-16 GPIOx bits clear register GPIOx_BRR.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	CLR15	Writing 1 clears GPIO X [15]. Writing 0 has no effect.
[14]	CLR14	Writing 1 clears GPIO X [14]. Writing 0 has no effect.
[13]	CLR13	Writing 1 clears GPIO X [13]. Writing 0 has no effect.
[12]	CLR12	Writing 1 clears GPIO X [12]. Writing 0 has no effect.
[11]	CLR11	Writing 1 clears GPIO X [11]. Writing 0 has no effect.
[10]	CLR10	Writing 1 clears GPIO X [10]. Writing 0 has no effect.
[9]	CLR9	Writing 1 clears GPIO X [9]. Writing 0 has no effect.
[8]	CLR8	Writing 1 clears GPIO X [8]. Writing 0 has no effect.
[7]	CLR7	Writing 1 clears GPIO X [7]. Writing 0 has no effect.
[6]	CLR6	Writing 1 clears GPIO X [6]. Writing 0 has no effect.
[5]	CLR5	Writing 1 clears GPIO X [5]. Writing 0 has no effect.
[4]	CLR4	Writing 1 clears GPIO X [4]. Writing 0 has no effect.
[3]	CLR3	Writing 1 clears GPIO X [3]. Writing 0 has no effect.
[2]	CLR2	Writing 1 clears GPIO X [2]. Writing 0 has no effect.
[1]	CLR1	Writing 1 clears GPIO X [1]. Writing 0 has no effect.
[0]	CLR0	Writing 1 clears GPIO X [0]. Writing 0 has no effect.

15.2.15 External events

EXTI_CR0 and EXTI_CR1 are used to select the rising-edge/falling-edge trigger type for an interrupt or DMA request generated by a GPIO external signal.

15.2.15.1 EXTI_CR0

Address: 0x4001 0E00

Reset value: 0x0

Table15-17 EXTI_CR0 external trigger configuration register 0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
T7	T6	T5	T4	T3	T2	T1	T0
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15:14]	T7	GPIO 0 [15]/P0 [15] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[13:12]	T6	GPIO 0 [14]/P0 [14] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[11:10]	T5	GPIO 0 [11]/P0 [11] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[9:8]	T4	GPIO 0 [6]/P0 [6] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[7:6]	T3	GPIO 0 [3]/P0 [3] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[5:4]	T2	GPIO 0 [2]/P0 [2] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[3:2]	T1	GPIO 0 [1]/P0 [1] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[1:0]	T0	GPIO 0 [0]/P0 [0] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered

15.2.15.2 EXTI_CR1

Address: 0x4001 0E04

Reset value: 0x0

Table15-18 EXTI_CR1 External Trigger Configuration Register 1

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
T15	T14	T13	T12	T11	T10	T9	T8
RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15:14]	T15	GPIO 2 [15]/P2 [15] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[13:12]	T14	GPIO 2 [12]/P2 [12] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[11:10]	T13	GPIO 2 [7]/P2 [7] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[9:8]	T12	GPIO 2 [4]/P2 [4] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[7:6]	T11	GPIO 2 [3]/P2 [3] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[5:4]	T10	GPIO 1 [10]/P1 [10] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[3:2]	T9	GPIO 1 [1]/P1 [1] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered
[1:0]	T8	GPIO 1 [0]/P1 [0] external trigger type selection 00: No trigger 01: Falling edge trigger 10: Rising edge trigger 11: Both rising edge and falling edge are triggered

Table15-19 GPIO Interrupt/DMA Request Event Resource Distribution Table

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

P0	√	√			√					√			√	√	√	√
P1						√									√	√
P2	√			√					√			√	√			
P3																

15.2.15.3 EXTI_IE

Address: 0x4001 0E08

Reset value: 0x0

Table15-20 EXTI_IE GPIO Interrupt Enable Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				EXIT3_RE	EXIT2_RE	EXIT1_RE	EXTI0_RE					EXIT3_IE	EXIT2_IE	EXIT1_IE	EXTI0_IE
				RW	RW	RW	RW					RW	RW	RW	RW
				0	0	0	0					0	0	0	0

Location	Bit name	Explain
[31:12]		Not used
[11]	EXTI3_RE	GPIO3 DMA request enable, required with EXTI_CR0/1
[10]	EXTI2_RE	GPIO2 DMA request enable, required with EXTI_CR0/1
[9]	EXTI1_RE	GPIO1 DMA request enable required with EXTI_CR0/1
[8]	EXTI0_RE	GPIO0 DMA request enable required with EXTI_CR0/1
[7:4]		Not used
[3]	EXTI3_IE	GPIO3 interrupt enable, required with EXTI_CR0/1
[2]	EXTI2_IE	GPIO2 interrupt enable, required with EXTI_CR0/1
[1]	EXTI1_IE	GPIO1 interrupt enable, required with EXTI_CR0/1
[0]	EXTI0_IE	GPIO0 interrupt enable, required with EXTI_CR0/1

Each set of GPIO external request signals can be used collectively as an interrupt request, or a DMA request. Typically, however, each set of signals is not used as both an interrupt and a DMA request. The DMA request signal is also generated from the interrupt flag setting. When the interrupt occurs, if the EXTIx_RE is enabled, the DMA will clear all the interrupt flags of the group of GPIOs after responding to the request. In GPIO0, for example, only one of the 16 GPIOs P0.0 to P0.15 is typically used as an external event source, either as an interrupt request or as a DMA request. If both P0.0 and P0.1 external event requests are enabled, both IOs generate external events. If it is a DMA request, the DMA will clear all external event flags from P0.0 to P0.15 when any one of P0.0 and P0.1 external events occurs.

However, one IO in GPIO0 is allowed as an interrupt event request, while one IO in GPIO2 is enabled as a DMA request. When the IO of GPIO2 has a DMA request, the DMA will only clear the external event flag of GPIO2 in hardware, and will not affect the external event flag of GPIO0. The external event of GPIO0 will generate an interrupt request, which will be processed by the interrupt service routine.

15.2.15.4 EXTI_IF

Address: 0x4001 0E0C

Reset value: 0x0

Table15-21 EXTI_IF external interrupt flag register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IF15	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7	IF6	IF5	IF4	IF3	IF2	IF1	IF0
RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	IF15	GPIO 2 [15]/P2 [15] external interrupt flag. Interrupt flag active high, write 1 to clear
[14]	IF14	GPIO 2 [12]/P2 [12] external interrupt flag. Interrupt flag active high, write 1 to clear
[13]	IF13	GPIO 2 [7]/P2 [7] external interrupt flag. Interrupt flag active high, write 1 to clear
[12]	IF12	GPIO 2 [4]/P2 [4] external interrupt flag. Interrupt flag active high, write 1 to clear
[11]	IF11	GPIO 2 [3]/P2 [3] external interrupt flag. Interrupt flag active high, write 1 to clear
[10]	IF10	GPIO 1 [10]/P1 [10] external interrupt flag. Interrupt flag active high, write 1 to clear
[9]	IF9	GPIO 1 [1]/P1 [1] external interrupt flag. Interrupt flag active high, write 1 to clear
[8]	IF8	GPIO 1 [0]/P1 [0] external interrupt flag. Interrupt flag active high, write 1 to clear
[7]	IF7	GPIO 0 [15]/P0 [15] external interrupt flag. Interrupt flag active high, write 1 to clear
[6]	IF6	GPIO 0 [14]/P0 [14] external interrupt flag. Interrupt flag active high, write 1 to clear
[5]	IF5	GPIO 0 [11]/P0 [11] external interrupt flag. Interrupt flag active high, write 1 to clear
[4]	IF4	GPIO 0 [6]/P0 [6] external interrupt flag. Interrupt flag active high, write 1 to clear
[3]	IF3	GPIO 0 [3]/P0 [3] external interrupt flag. Interrupt flag active high, write 1 to clear
[2]	IF2	GPIO 0 [2]/P0 [2] external interrupt flag. Interrupt flag active high, write 1 to clear
[1]	IF1	GPIO 0 [1]/P0 [1] external interrupt flag. Interrupt flag active high, write 1 to clear
[0]	IF0	GPIO 0 [0]/P0 [0] external interrupt flag. Interrupt flag active high, write 1 to clear

Even if the EXTI_IE is not enabled, the EXTI_IF will still be set, but no interrupt or DMA request is generated.

15.2.15.5 CLKO_SEL

Address: 0x4001 0E10

Reset value: 0x0

Table15-22 CLKO_SEL GPIO output clock select register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
											HSE_OE	ADC_OE	PLL_OE	HSI_OE	LSI_OE
											RW	RW	RW	RW	RW
											0	0	0	0	0

Location	Bit name	Explain
[31:5]		Not used
[4]	HSE_OE	Crystal clock output enable. 1: Enable; 0: Disabled.
[3]	ADC_OE	ADC Clock Output Enable. 1: Enable; 0: Disabled.
[2]	PLL_OE	PLL Output Enable. 1: Enable; 0: Disabled.
[1]	HSI_OE	HRC Output Enable. 1: Enable; 0: Disabled.
[0]	LSI_OE	LRC Output Enable. 1: Enable; 0: Disabled.

Each clock signal of the chip can be observed from P0.0/P2.7. Note that because the ADC/PLL is clocked at a high frequency, it may not be able to drive large loads.

If clock output is required, configure the second function of P0.0 as 1, that is, GPIO0_F3210 = 0x0001, and enable P0.0 output to enable GPIO0_POE = 0x0001, and configure the CLKO_SEL to select a certain clock to output through P0.0.

15.2.15.6 PWM_SWAP

Address: 0x4001 0E14

Reset value: 0x0

Table15-23 PWM_SWAP register

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

Location	Bit name	Explain
[31:2]		Not used
[1:0]	SWAP	Writing 0x67 to this register sets BIT [1:0] to 1, writing 0x69 sets BIT [1:0] to 2, and writing any other value sets it to 0

When the value of MCPWM0_SWAP is 0, the MCPWM channel output is related to the GPIO as follows:

Table15-24 MCPWM default output table

MCPWM output sequence	GPIO correspondence order	
MCPWM0_CH0P	P1.4	P0.15
MCPWM0_CH0N	P1.5	P1.0
MCPWM0_CH1P	P1.6	P2.11
MCPWM0_CH1N	P1.7	P2.12
MCPWM0_CH2P	P1.8	P2.4
MCPWM0_CH2N	P1.9	P2.5
MCPWM0_CH3P	P1.10	P3.2
MCPWM0_CH3N	P1.11	P3.4
MCPWM0_CH4P	P3.10	P0.3
MCPWM0_CH4N	P3.11	P0.4
MCPWM0_CH5P	P2.9	P0.5
MCPWM0_CH5N	P2.10	P0.6

In order to facilitate the multi-die package SIP design of MCU chip and Gate Driver chip, the PWM channel output in GPIO module can be remapped. In the following table, the leftmost column of MCPWM channels is rearranged, and the two columns on the right are unchanged, which is still consistent with the IO arrangement order of the chip.

When the value of the PWM_SWAP is 1, the correspondence is as follows:

Table15-25 MCPWM Output Mapping Table when PWM_SWAP = 1

MCPWM output sequence	GPIO correspondence order	
MCPWM0_CH0N	P1.4	
MCPWM0_CH1N	P1.5	
MCPWM0_CH2N	P1.6	
MCPWM0_CH0P	P1.7	
MCPWM0_CH1P	P1.8	
MCPWM0_CH2P	P1.9	
MCPWM0_CH3N	P1.10	P3.2
MCPWM0_CH4N	P1.11	P3.4
MCPWM0_CH5N	P3.10	P0.3
MCPWM0_CH3P	P3.11	P0.4
MCPWM0_CH4P	P2.9	P0.5
MCPWM0_CH5P	P2.10	P0.6

When the value of the PWM_SWAP is 2, the corresponding relationship is as follows:

Table15-26 MCPWM output mapping table when PWM_SWAP = 2

MCPWM output sequence	GPIO correspondence order	
MCPWM0_CH0N	P1.13	
MCPWM0_CH1N	P1.14	
MCPWM0_CH2N	P1.15	
MCPWM0_CH0P	P2.0	
MCPWM0_CH1P	P1.4	
MCPWM0_CH2P	P1.5	
MCPWM0_CH3N	P1.6	
MCPWM0_CH4N	P1.7	

MCPWM0_CH5N	P1.8	
MCPWM0_CH3P	P1.9	
MCPWM0_CH4P	P1.10	
MCPWM0_CH5P	P1.11	

15.3 Implementation description

15.3.1 Pull-up implementation

LKS32MC07x series chips, some GPIOs are equipped with pull-up resistors. GPIOs with the pull-up function are as follows:

Table15-27 GPIO pull-up resource table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	√	√					√		√	√		√	√	√		√
P1					√	√							√			√
P2	√	√	√			√	√	√	√	√	√	√				
P3							√									

15.3.2 Filter implementation

Some GPIOs of LKS32MC07x series chips support the filtering operation of input signals. The filtering time width is 4 LSI clock cycles, about 120 us, that is, changes less than 120 us will be filtered. Note that since the filter uses an LSI clock and the RC itself has limited accuracy, there is an individual variation in the specific filter time constant.

Table15-28 GPIO filter resource table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	√	√	√	√	√				√	√				√		√
P1																
P2										√	√	√	√			
P3																

15.3.3 Pull-up implementation

LKS32MC07x series chips, if an IO is configurated as general-purpose IO, it could be open drained. But if an IO is configurated as serial interface or other functions than GPIO, only some of them could be open-drained.

Table15-29 GPIO open-drain resource table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0	√	√							√	√		√	√			√
P1					√	√										
P2	√	√	√	√		√	√	√	√	√	√	√				
P3																

15.4 Application Guide

15.4.1 External Interrupt

Examples are as follows:

```
GPIO0_PIE = 0x0001;                //Enable P0 [0] input
NVIC_EnableIRQ(GPIO_IRQn);         //Enable GPIO interrupt
__enable_irq();                    //enable interrupt
i = 1000;
while(i--);
                                   //P0 [7] External square wave signal on IO
EXTI_CR0 = 0x0002;                 //enable the rising edge of P0 [7] to trigger an external interrupt
    while(irq_flag != 2);          //external signal flips twice, causing two interrupts IRQ_flag two increments in the GPIO
interrupt handler
EXTI_CR0 = 0x0001;                 //enable the falling edge of P0 [7] to trigger an external interrupt
    while(irq_flag != 4);
EXTI_CR0 = 0x00003;                //enable both rising and falling edge triggers on P0 [7] to generate an external interrupt.
    while(irq_flag != 8);
EXTI_CR0 = 0x0000;                 //If the upper and lower edge triggers of P0 [7] are disabled at the same time, the external
interrupt cannot be generated
```

15.4.2 Analog function using GPIO

With IE and OE of the GPIO turned off, the emulation function is available. In this case, the PAD is connected directly to the analog module through an internal resistor.

16 CRC

16.1 Overview

CRC is Cyclic Redundancy Check Code(Cyclic Redundancy Check): It is one of the most commonly used error checking codes in the field of data communication. It is characterized in that the length of the information field and the check field can be selected arbitrarily. Cyclic Redundancy Check (CRC) is a data transmission error detection feature that performs a polynomial calculation on the data and attaches the result to the frame. The receiving device performs a similar algorithm to ensure the correctness and integrity of the data transmission.

The process of using CRC for error detection can be simply described as follows: at the sending end, according to the K-bit binary code sequence to be transmitted, an R-bit check code (CRC code) is generated according to a certain rule, which is attached to the original information to form a new binary code sequence number K + R bits, and then sent out. At the receiving end, a check is made according to the rules followed between the information code and the CRC code to determine whether an error was made in the transmission. This rule is called "generator polynomial" in error control theory.

16.2 Basic principles

The basic principle of Cyclic Redundancy Check (CRC) is that the R-bit check code is spliced after the K-bit information code, and the length of the whole code is N bits, so this code is also called (N, K) code. For a given (N, K) code, it can be shown that there exists a polynomial G(x) with the highest power N-K=R. A check code of K bits of information can be generated from G(X), and G(X) is called the generator polynomial of this CRC code. The specific generation process of the check code is as follows: assuming that the information to be sent is represented by a polynomial C(X), shift C(X) to the left by R bits (represented as $C(X)*2^R$, so that R bits will be left on the right side of C(X), which is the position of the check code. The remainder obtained by dividing $C(X)*2^R$ by the generator polynomial G(X) is the check code.

Any code consisting of a string of binary bits can be matched one-to-one with a polynomial whose coefficients are only values of '0' and' 1 '. For example, The polynomial corresponding to code 1010111 is $x^6+x^4+x^2+x+1$, and the polynomial is $x^5+x^3+x^2+x+1$ corresponds to code 101111.

16.3 Basic concepts

16.3.1 Correspondence

There is a direct correspondence between a polynomial and a binary number: the highest power of X corresponds to the highest bit of the binary number, the following bits correspond to the powers of the polynomial,any power of x corresponds to 1, and no power of X corresponds to 0. It can be seen that the highest power of X is R, and the corresponding binary number has R + 1 bits.

Polynomial include the generator polynomial G(X) and the information polynomial C(X).

If the generator polynomial is $G(X) = X^4 + X^3 + X + 1$, it can be converted into a binary digital 11011.

The transmission information is 101111, which can be converted into a data polynomial $C(X) = X^5+X^3+X^2+X+1$.

16.3.2 Generator polynomial

The generator polynomial is a contract between the receiver and the sender, that is a binary number. This number remains constant throughout the transmission.

At the sender, the generator polynomial is used to divide the information polynomial modulo 2 to generate a check code. At the receiver, the generator polynomial is used to perform modulo 2 division on the received encoded polynomial to detect and determine the error location.

The following conditions shall be met:

A. The highest and lowest bits of the generator polynomial must be 1.

B. When an error occurs in any bit of the transmitted information (CRC code), the remainder shall not be 0 after division by the generator polynomial.

C. When errors occur in different bits, the remainders should be different.

D. RightRemainderTo continue the division, the remainder should be cycled.

16.3.3 Number of check code bits

Number of bits of CRC check code = number of bits of generator polynomial-1. Note that the most significant bit 1 of the generator polynomial is omitted from the shorthand notation for some generator polynomials.

16.3.4 Generation step

1. Convert the generator polynomial G(X) whose highest power of X is R into a corresponding R + 1-bit binary number.

2. Shift the information code left by R bits, which is equivalent to the corresponding information polynomial $C(X)*2^R$.

3. Divide the information code with the generator polynomial (binary number) to obtain the remainder of R bits (note: the remainder obtained by binary division here is actually the remainder obtained by modulo 2 division, not equal to the remainder obtained from the division of its corresponding decimal number) .

4. Spell the remainder to the position vacated after the information code is shifted to the left to obtain a complete CRC code.

Suppose the generator polynomial used is G(X) = X^3 + X + 1. The 4-bit original message is 1010. Find the encoded message.

Solution:

1. Convert the generator polynomial G(X) = X^3 + X + 1 to the corresponding binary divisor 1011.

2. The generator polynomial of this question has 4 bits (R + 1) (note: the check code obtained from the calculation of the 4-bit generator polynomial is 3 bits, and R is the number of bits of the check code). The original message C(X) should be shifted left by 3 (R) bits to 1010 000.

3. Use the binary number corresponding to the generator polynomial to perform modulo 2 division (high-order alignment) on the original message that has been shifted 3 bits to the left, which is equivalent to bitwise Xor:

```
1010000

1011

------------------

0001000

0001011

------------------

0000011
```

The remaining bits are 011, so the final code is 1010 011.

POL=0x13， data=0x77

```
011101110000000

 10010011

 01111101000000

  10010011

  0110100100000

   10010011

   010000010000

        10010011
        00010001000
    10010011
            00011011
```

16.4 Register

16.4.1 Address assignment

The base address of the CRC is 0x4001_0F00, and the register list is as follows:

Table16-1 List of CRC registers

Name	Offset address	Explain
CRC0_DR	0x00	CRC data (input information code/output code) register
CRC0_CR	0x04	CRC control register
CRC0_INIT	0x08	CRC initial code register
CRC0_POL	0x0C	Binary code register corresponding to CRC generator polynomial

16.4.2 Register description

16.4.2.1CRC0_DR CRC information code register

Address: 0x4001 0F00

Reset value: 0x0

Table16-2 CRC0_DR CRC Data Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR															
RW															
0															

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

Location	Bit name	Explain
[31:0]	DR	Toring the information code to be coded and the code after CRC check

The CRC0_DR register is used both to place the data to be verified and to return the verification result. Writing to the CRC0_DR register triggers a CRC calculation. The data to be encoded should be written last after the CR and other registers have been configured to trigger the start of the CRC calculation.

16.4.2.2CRC0_CR CRC Control Register

Address: 0x4001 0F04

Reset value: 0x0

Table16-3 CRC0_CR CRC Control Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			REV_OUT_TYPE			REV_IN_TYPE				POLY_SIZE					RESET
			RW			RW				RW					WO
			0			0				0					0

Location	Bit name	Explain
[31:13]		Not used
[12]	REV_OUT_TYPE	Whether to output the code after CRC check after inversion, i.e. B [31] = B [0], B [30] = B [1], ... [b0]=b[31]
[11:10]		Not used
[9:8]	REV_IN_TYPE	Type of data inversion to be encoded 00: No reversal 01: Byte reversal, i.e. B [31] = B [24], B [30] = B [25], ... , b[24]=b[31], ... , b[7]=b[0], b[6]=b[1], ... , b[0]=b[7] 10: Invert by halfword (16 bit), i.e. B [31] = B [16], B [30] = B [17], ... , b[16]=b[31], ... , b[15]=b[0], b[14]=b[1], ... , b[0]=b[15] 11: Reverse by word, i.e. B [31] = B [0], B [30] = B [1], ... [b0]=b[31]
[7:6]		Not used
[5:4]	POLY_SIZE	Output Encoding (Polynomial) Bit Width 00: 32bits 01: 16bits 10: 8bits 11: 7bits
[3:1]		Not used
[0]	RESET	Data source for CRC calculation with input information code 0: from the previous calculation result 1: from CRC0_INIT Write 1 to reset the CRC data and clear it automatically, and read it back to 0.

It is also important to note that writing a 1 to the CRC0_CR. RESET resets the CRC0_INIT register to 0xFFFFFFFF.

If the result of the CRC calculation needs to be cleared, a 1 should be written to the CRC0_CR. RESET, otherwise subsequent CRC calculations will start with the previous result.

16.4.2.3CRC0_INIT CRC Initial Code Register

Address: 0x4001 0F08

Reset value: 0x0

Table16-4 CRC0_INIT CRC Initial Code Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INIT															
RW															
0xFFFFFFFF															

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

Location	Bit name	Explain
[31:0]	INIT	Store the initial code

CRC check calculation starts after initial CRC0_DR Xor CRC0_INIT .

16.4.2.4CRC0_POL CRC generation code register

Address: 0x4001 0F0C

Reset value: 0x0

Table16-5 CRC0_POL CRC generation code register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL															
RW															
0x04C11DB7															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL															
RW															
0x04C11DB7															

Location	Bit name	Explain
[31:0]	POL	Store the generator code corresponding to the generator polynomial

17 UART

17.1 Overview

A Universal Asynchronous Receiver Transmitter (Universal Asynchrounous Receiver/ Transmitter), commonly referred to as a UART, is an asynchronous receiver/transmitter.

The main features of UART are as follows:

- Support full-duplex operation
- Supports single-wire half-duplex operation
- Support for 8/9 data bit
- Support 1/2 stop bit
- Support odd/even/no check mode
- With 1-byte send buffer
- With 1-byte receive buffer
- Support LIN mode break character transceiving
- Idle frame detection is supported
- Multi-drop Slave/Master mode supporting one master and multiple slaves

17.2 Functional description

17.2.1 Transport(TX)

The UART includes a byte transmit buffer. When there is data in the transmit buffer, the UART shifts the data in the transmit buffer into the serial shift register and transmits it out through the UART_TXD port. The UART will automatically transmit as long as there is data in the UART transmit buffer.

After the loading is completed, the send buffer empty interrupt is generated. At this time, the user can fill the next byte to be sent in the send buffer. In this way, after the sending is completed, the UART will load this byte and send it.

The transmit complete interrupt is generated when the transmit is complete.

Sending process:

Set SYS_CLK_FEN.UART_CLK_EN = 1 turns on the UART clock

Set the UART_CTRL.BYTE_LEN to select the 8/9 bit data byte length

Set the UART_CTRL.CK_EN and select whether to enable data byte check

Set the UART_CTRL.CK_TYPE and select whether to use odd parity or even parity

Sets the UART_CTRL.BIT_ORDER to select whether LSB first or MSB first is to be transmitted

Set the UART_CTRL.STOP_LEN and select the stop bit length as 1bit/2bit

If DMA is used to transfer data, set the UART_RE.TX_BUF_EMPTY_RE = 1, that is, when the TX buffer is empty, the DMA is triggered to carry new data to the UART; If software polling or interrupts is used,set the UART_IE.TX_BUF_EMPTY_IE=1

If DMA is used to carry data to UART for sending, the DMA needs to be set accordingly.

If software is used to carry data to the UART for sending, the software is required to write data to the UART_BUFF, which can trigger the UART to start sending data through the UART_TXD port.

17.2.2 Receive(RX)

The UART includes a receive buffer of one byte, when the reception of one byte is completed, a receive interrupt will be generated and the received byte will be stored in the receive buffer. The user should finish reading this byte before the UART receives the next byte, otherwise the buffer will be written to the newly received byte. In the receiving part, a high-speed clock is used to over-sample the UART_RX signal to determine the steady-state data signal, so as to prevent erroneous reception caused by noise interference.

17.2.3 Uart frame format

The format of data transmitted and received on the UART signal is generally as follows:

The signal line is idle;

Start bit (1 bit Start bit: 1 bit Zero)

Data bytes (data word, 8bits or 9bits, LSB first or MSB first)

Stop Bit (1/2 bit Stop bit: 1/2 Bit Ones)

The length of the data byte can be selected by the UART_CTRL.BYTE_LEN, 8 bits (UART_CTRL.BYTE_LEN = 0) or 9 bits (UART_CTRL.BYTE_LEN = 1). The start bit is 1 bit zero, the TX signal line is low, and the TX signal line is high during the stop bit.

Note that when the check bit is enabled (UART_CTR.CK_EN = 1), the check bit replaces the most significant bit, the MSB, of the data byte, so that the 8 bit data byte is actually 7 bit data + 1 bit check word and the 9 bit data byte is actually 8 bit data + 1 bit check word.

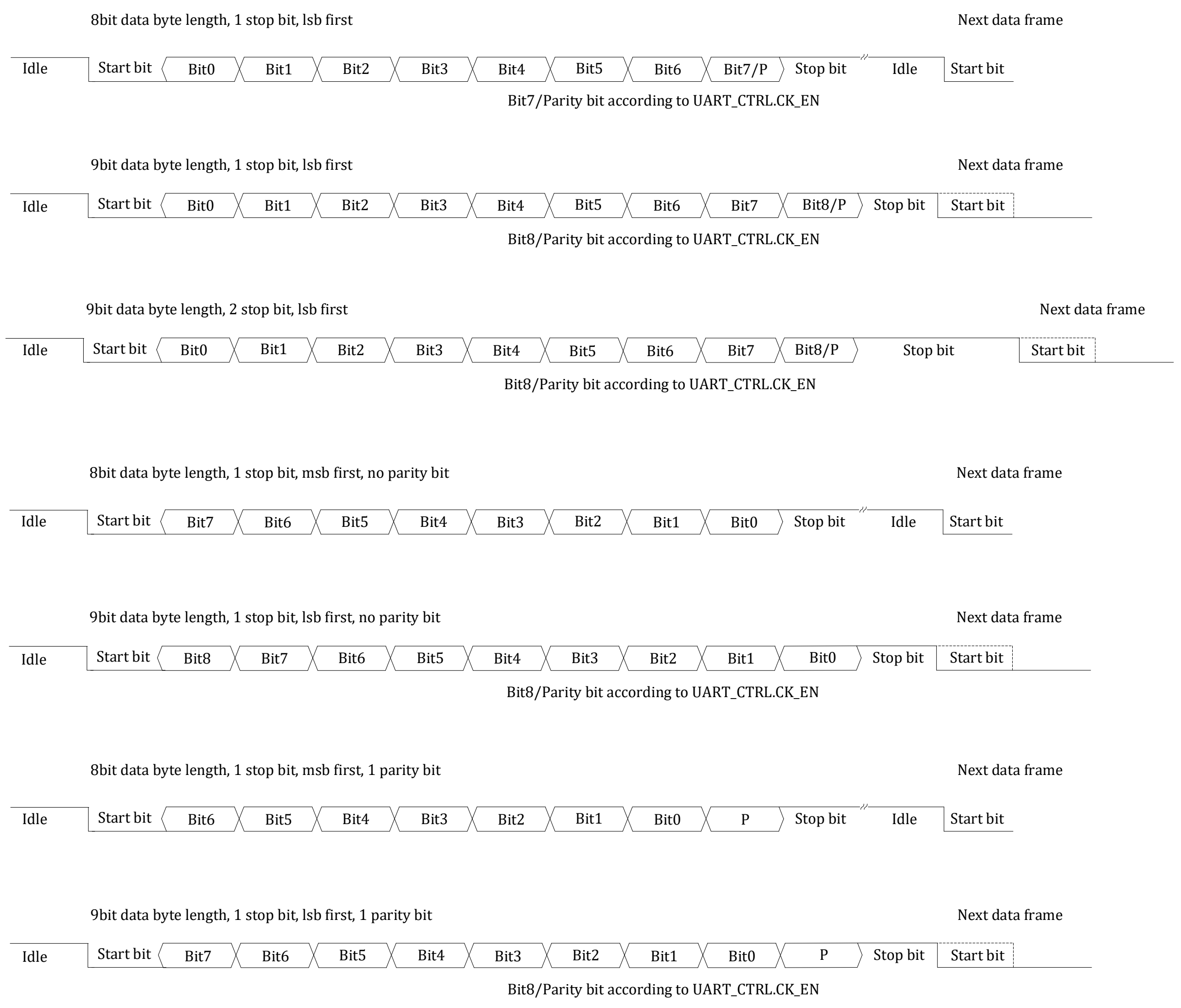

8bit data byte length, 1 stop bit, lsb first
Next data frame
Idle
Start bit
Bit0
Bit1
Bit2
Bit3
Bit4
Bit5
Bit6
Bit7/P
Stop bit
Idle
Start bit
Bit7/Parity bit according to UART_CTRL.CK_EN
9bit data byte length, 1 stop bit, lsb first
Next data frame
Idle
Start bit
Bit0
Bit1
Bit2
Bit3
Bit4
Bit5
Bit6
Bit7
Bit8/P
Stop bit
Start bit
Bit8/Parity bit according to UART_CTRL.CK_EN
9bit data byte length, 2 stop bit, lsb first
Next data frame
Idle
Start bit
Bit0
Bit1
Bit2
Bit3
Bit4
Bit5
Bit6
Bit7
Bit8/P
Stop bit
Start bit
Bit8/Parity bit according to UART_CTRL.CK_EN
8bit data byte length, 1 stop bit, msb first, no parity bit
Next data frame
Idle
Start bit
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
Stop bit
Idle
Start bit
9bit data byte length, 1 stop bit, lsb first, no parity bit
Next data frame
Idle
Start bit
Bit8
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
Stop bit
Start bit
Bit8/Parity bit according to UART_CTRL.CK_EN
8bit data byte length, 1 stop bit, msb first, 1 parity bit
Next data frame
Idle
Start bit
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
P
Stop bit
Idle
Start bit
9bit data byte length, 1 stop bit, lsb first, 1 parity bit
Next data frame
Idle
Start bit
Bit7
Bit6
Bit5
Bit4
Bit3
Bit2
Bit1
Bit0
P
Stop bit
Start bit
Bit8/Parity bit according to UART_CTRL.CK_EN

Figure17-1 Uart frame format

17.2.4 Baud rate configuration

The UART input clock is the system master clock, and the baud rate is achieved by two stages of frequency division.

Baud rate = UART module clock/ (256 * DIVH + DIVL + 1)

The UART module clock can be divided by SYS_CLK_DIV2.

UART module clock = system master clock/ (1 + SYS_CLK_DIV2)

Table17-1 Example of Uart baud rate configuration

Uart baud rate	SYS_CLK_DIV2	UART_DIVH	UART_DIVL
300	0x0007	0x9C	0x3F
600	0x0003	0x9C	0x3F
1200	0x0001	0x9C	0x3F
2400	0x0000	0x9C	0x3F
4800	0x0000	0x4E	0x1F
9600	0x0000	0x27	0x0F
19200	0x0000	0x13	0x87
38400	0x0000	0x09	0xC3

43000	0x0000	0x08	0xB8
56000	0x0000	0x06	0xB1
57600	0x0000	0x06	0x82
115200	0x0000	0x03	0x40

Note that the configuration factor may not be unique for the same baud rate.

17.2.5 Transmit/receive port interchange (TX/RX interchange)

The UART module supports the interchange of Tx and Rx ports. The Tx and Rx ports can be interchanged by configuring the corresponding GPIO of the Tx as the input and the corresponding GPIO of the Rx as the output. At this time, the second function of GPIO is still selected as UART, and the configuration of UART itself does not need to be modified.

In addition, if you want to use a GPIO as Tx and Rx at the same time, you need to time-multiplex the IO as input or output, corresponding to Rx or Tx, to achieve single-port half-duplex logic.

17.2.6 Multi-machine communication

In a multi-machine communication scenario, one device is usually used as a master device, and there are several slave devices. The UARTm_TXD port of the master device is connected to the UARTs_RXD port of all the slave devices, and the UARTs_TXD of the slave device is connected to the UARTm_RXD port of the master device. Such as Figure 17-2 Shown is the interconnection of a master device and three slave devices (addresses 0x51, 0x52, and 0x53).

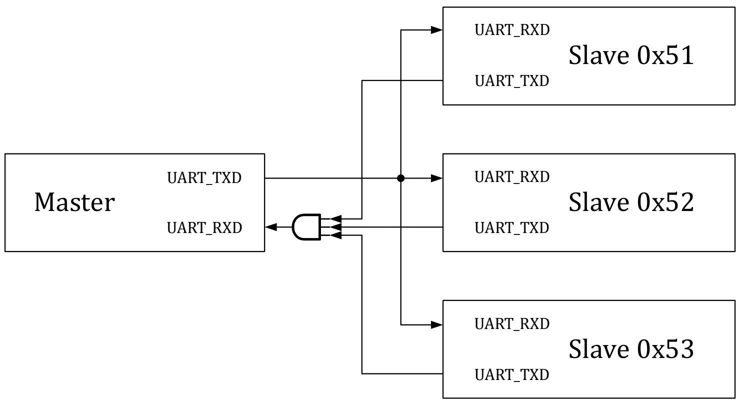


Figure17-2 UART multi-machine communication interconnection topology

In order to reduce the message processing load of the slave device, the slave device should only process the UART data sent to it and ignore the data sent to other slave devices. In a multi-machine communication scenario, each slave machine has an 8-bit device address, UART_ADR. After the slave device sets the UART_CTRL.MD_EN = 1, the multi-machine communication filtering mode is enabled, and the host does not need to set the MD_EN. Both the master and the slave need to use the 9bit mode to send data, that is, set the UART_CTRL.BYTE_LEN = 1. When the transmit data byte MSB (BIT8) is 1, the lower 8 bits (BIT7: BIT0) are the slave address issued by the master, that is, the address byte; When the transmit data byte MSB (BIT8) is 0, the lower 8 bits (BIT7: BIT0) are the data that the master sends to a specific slave, that is, the data byte.

After receiving the data byte with MSB (BIT8) = 1, all slave devices determine whether the address of the lower 8 bits (BIT7: BIT0) is equal to its own UART_ADR configuration value. If they are equal, it means that the subsequent data is sent to the slave, otherwise, the slave enters a silent state, ignoring all data sent by the subsequent master. When the slave device receives a data byte with MSB (BIT8) = 1 again and the lower 8 bits (BIT7: BIT0) of the address are equal to its own UART_ADR configuration value, it indicates that the slave device is selected and exits the silent state. The slave computer sets the UART_CTRL.MD_EN = 1 to enter the silent state immediately after the multi-computer communication filtering mode is enabled. If the master sends a data byte without sending an address byte with MSB = 1, all slaves are silent and do not receive any data.

In the multi-machine communication mode, the master device does not need to set the UART_CTRL.MD_EN = 1, and the slave device needs to set the UART_CTRL.MD_EN = 1. If there is no need for address filtering, the slave device may not set the UART_CTRL.MD_EN = 1 and interpret the received data with software. At this time, the slave device will receive and process all message data.

In the multi-machine communication mode, if the slave device sets the UART_CTRL.MD_EN = 1, the UART_IF.RX_DONE_IF flag is not set whether the address byte hits the UART_ADR or not; If the addresses do not match, that is, if the slave is silent, the UART_IF.RX_DONE_IF is not asserted even though the master is sending data. If the address matches, the UART_IF.RX_DONE_IF flag of the slave device is set to 1 when the data byte sent by the master device is received, indicating that the slave device has received a data byte.

Because the check bit needs to occupy 1 bit data bit, and the multi-machine communication needs to use 9 bit byte length. Therefore, the check bit is not supported in the multi-machine communication scenario. When setting the UART_CTRL.MD_EN = 1, do not set the UART_CTRL. CK_EN = 1. Both the master and the slave need to set the UART_CTRL.BYTE_LEN = 1, use the 9 bit mode, and set the UART_CTRL.MD_EN. Multi-machine communication supports MSB first, that is, UART_CTRL.BIT_ORDER = 1.

Such as Figure 17-3show, the master sends data 0x03 first, but there is no slave address hit at this time, so none of the 3 slave devices receive 0x03. The master sends the address byte 0x151 and the slave 0x51 is selected. However, after that, the master device does not send data, but directly sends the address byte 0x153. The slave device 0x53 is selected, and the master device sends three consecutive data bytes 0x03, 0x53, 0x04, which are received by the slave device 0x53. The master then sends the address byte 0x152, followed by the data byte 0x07, which is received by the slave 0x52.

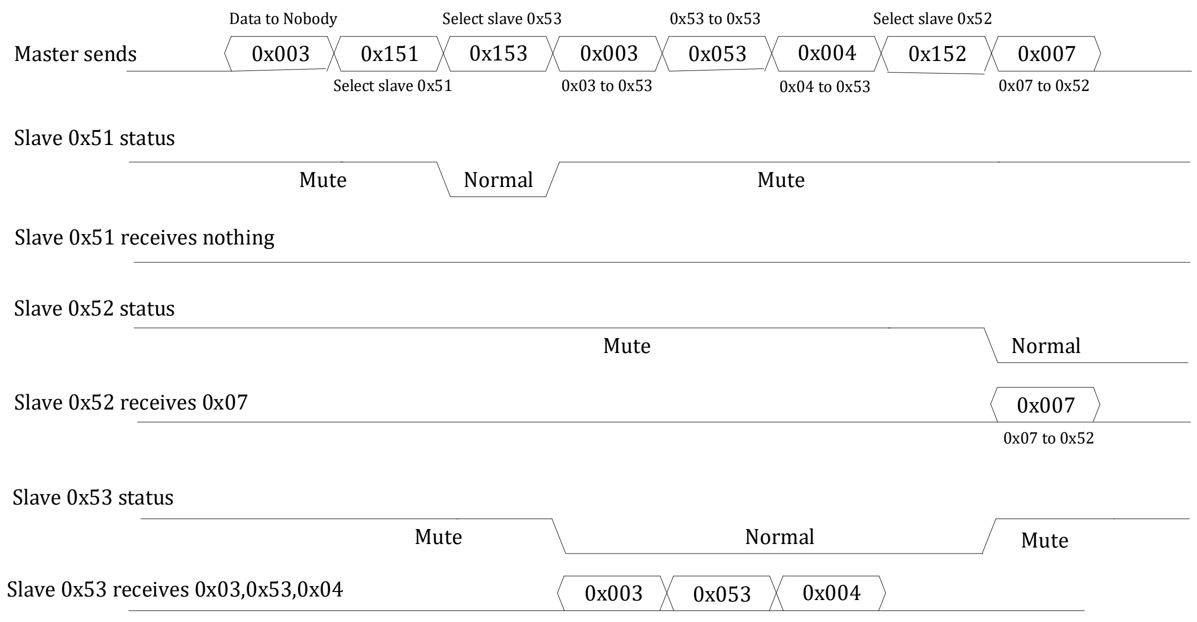


Figure17-3 UART Multi-machine Communication Example

17.2.7 Check digit

The parity bit can be enabled by setting the UART_CTRL.CK_EN = 1. At the same time, according to the different UART_CTRL.BYTE_LEN of byte length, the frame format of UART has four cases.

Table17-2 Uart frame format

BYTE_LEN	CK_EN	UART frame
0	0	| StartBit | 8bit data | StopBit |
0	1	| StartBit | 7bit data | Parity | StopBit |
1	0	| StartBit | 9bit data | StopBit |
1	1	| StartBit | 8bit data | Parity | StopBit |

17.3 Register

17.3.1 Address assignment

The UART0 and UART1 implementations are identical.

UART0 Base Address 0x4001_1000.

UART1 Base Address 0x4001_1100.

Table17-3 UART address allocation list

Name	Offset address	Explain
UARTx_CTRL	0x00	UART control register
UARTx_DIVH	0x04	Uart Baud Rate Setting High Byte Register
UARTx_DIVL	0x08	Uart Baud Rate Set Low Byte Register

UARTx_BUFF	0x0C	UART Transceiver Buffer Register
UARTx_ADR	0x10	485 Communication Address Match Register
UARTx_STT	0x14	UART status register
UARTx_RE	0x18	UART DMA Request Enable Register
UARTx_IE	0x1C	UART Interrupt Enable Register
UARTx_IF	0x20	UART Interrupt Flag register
UARTx_IOC	0x24	UART IO control

17.3.2 UARTx_CTRL:UARTx Control Register (X = 0,1)

The addresses are: 0x4001_1000, 0x4001_1100

Reset value: 0x0

Table17-4 UART control register UARTx_CTRL.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								DUPLEX	LIN_EN	MD_EN	CK_EN	CK_TYPE	BIT_ORDER	STOP_LEN	BYTE_LEN
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:8]		Not used
[7]	DUPLEX	Duplex. The default is 0. 0: Full duplex; 1: Half duplex
[6]	LIN_EN	LIN mode is enabled, default value is 0. 0: Close; 1: On
[5]	MD_EN	Multi-drop is enabled and the default value is 0. 0: Close; 1: On
[4]	CK_EN	Data validation switch. The default value is 0. 0: Close; 1: On
[3]	CK_TYPE	Parity configuration. Default is 0. 0: EVEN; 1: ODD
[2]	BIT_ORDER	Data sending order configuration. The default value is 0. 0:LSB； 1:MSB
[1]	STOP_LEN	Stop bit length configuration. The default value is 0. 0:1-Bit； 1:2-Bit
[0]	BYTE_LEN	Data length configuration. The default value is 0. 0:8-Bit； 1:9-Bit

17.3.3 UARTx_DIVH:UARTx Baud Rate Setting High Byte Register (X = 0,1)

The addresses are: 0x4001_1004, 0x4001_1104

Reset value: 0x0

Table17-5 Uart Baud Rate Setting High Byte Register UARTx_DIVH.

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

Location	Bit name	Explain
[31:8]		Not used
[7:0]	DIVH	Baud Rate Setting High Byte BAUDRATE = master clock/ (1 + 256 * UART_DIVH + UART_DIVL)

17.3.4 UARTx_DIVL:UARTx Baud Rate Setting Low Byte Register (X = 0,1)

The addresses are: 0x4001_1008, 0x4001_1108

Reset value: 0x0

Table17-6 Uart Baud Rate Set Low Byte Register UART_DIVL.

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

Location	Bit name	Explain
[31:8]		Not used
[7:0]	DIVL	Baud rate setting low byte BAUDRATE = master clock/ (1 + 256 * UART_DIVH + UART_DIVL)

17.3.5 UARTx_BUFF:UARTx Transmit and Receive Buffer Register (X = 0,1)

The addresses are: 0x4001_100C, 0x4001_110C

Table17-7 UART Transmit Buffer Register UART_BUFF.

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

Location	Bit name	Explain
[31:9]		Not used
[8:0]	BUFF	Write: send data cache; Read: Receive Data Register

The Tx_buffer and the Rx_buffer of the UART share the UART_BUFF. Where Tx_buffer is write-only and Rx_buffer is read-only. Thus, a read UART_BUFF register is an access UART_RX_BUFF, and a write UART_BUFF register is an access UART_TX_BUFF.

When the UARTx_CTRL.BYTE_LEN = 0, only the lower 8 bits of buffer are used for UART transceiver, that is, the UARTx_BUFF [7:0];

When UARTx_CTRL.BYTE_LEN = 1, all 9 bits of buffer are used for UART transceiver, that is, UARTx_BUFF [8:0].

If the UARTx_CTRL.CK_EN is enabled, that is, the check bit is used, the most significant bit of the UARTx_BUFF (BIT8 when UARTx_CTRL.BYTE_LEN = 1 or BIT7 UARTx_CTRL.BYTE_LEN = 0) has no effect. It is replaced with a check digit when transmitted and received as a check digit without being written to the UARTx_BUFF.

17.3.6 UARTx_ADR:UARTx Address Match Register (X = 0, 1)

The addresses are: 0x4001_1010, 0x4001_1110

Reset value: 0x0

Table17-8 UART Address Match Register UART_ADR.

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

Location	Bit name	Explain
[31:8]		Not used
[7:0]	ADR	Slave address in case of multi-machine communication

17.3.7 UARTx_STT:UARTx Status Register (X = 0,1)

The addresses are: 0x4001_1014, 0x4001_1114

Reset value: 0x0

Table17-9 UART status register UART_STT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												RX_BUSY	ADR_MATCH	TX_DONE	TX_BUF_EMPTY
												R	R	R	R
												0	0	1	1

Location	Bit name	Explain
[31:4]		Not used
[3]	RX_BUSY	1: The UART has detected the start symbol and is receiving 0: UART receiving end is idle This status bit is read-only and is set and cleared by hardware
[2]	ADR_MATCH	Address match flag in Multi-drop mode.

		1: match; 0: Not matched.
[1]	TX_DONE	Send complete flag bit. 1: Complete; 0: Incomplete.
[0]	TX_BUF_EMPTY	Transmit cache status bits. 1: null; 0: not empty.

17.3.8 UARTx_RE:UARTx DMA Request Enable Register (X = 0,1)

The addresses are: 0x4001_1018, 0x4001_1118

Reset value: 0x0

Table17-10 UART DMA Request Enable Register UART_RE.

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

Location	Bit name	Explain
[31:3]		Not used
[2]	TX_BUF_EMPTY	Transmit buffer empty DMA request switch. Default is 0. 0: Close; 1: On.
[1]	RX_DONE	Receive completion DMA request switch. Default value is 0. 0: Close; 1: On.
[0]	TX_DONE	Transmit complete DMA request switch. Default value is 0. 0: Close; 1: On.

17.3.9 UARTx_IE:UARTx Interrupt Enable Register (X = 0,1)

The addresses are: 0x4001_101C, 0x4001_111C

Reset value: 0x0

Table17-11 UART Interrupt Enable Register UART_IE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE	LBD	RX_OV	TX_OV	CK_ERR	STOP_ERR	TX_BUF_EMPTY	RX_DONE	TX_DONE
							RW	RW	RW	RW	RW	RW	RW	RW	RW
							0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:9]		Not used
[8]	IDLE	Idle frame interrupt enabled 0: Close; 1: On
[7]	LBD	LIN break character detect interrupt enabled 0: Close; 1: On
[6]	RX_OV	Receive buffer overflow interrupt enabled 0: Close; 1: On
[5]	TX_OV	Transmit buffer overflow interrupt enabled 0: Close; 1: On
[4]	CK_ERR	Checksum error interrupt switch. Default value is 0. 0: Close; 1: On
[3]	STOP_ERR	Stop bit error interrupt switch. Default value is 0. 0: Close; 1: On
[2]	TX_BUF_EMPTY	Transmit buffer air break switch. Default value is 0. 0: Close; 1: On
[1]	RX_DONE	Receive completion interrupt switch. The default value is 0. 0: Close; 1: On
[0]	TX_DONE	Transmit completion interrupt switch. The default value is 0. 0: Close; 1: On

17.3.10 UARTx_IF:UARTx Interrupt Flag Register (X = 0, 1)

The addresses are: 0x4001_1020, 0x4001_1120

Reset value: 0x0

Table17-12 UART Interrupt Enable Register UART_IF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							IDLE	LBD	RX_OV	TX_OV	CK_ERR	STOP_ERR	TX_BUF_EMPTY	RX_DONE	TX_DONE
							RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C	RW1C
							0	0	0	0	0	0	1	0	1

Location	Bit name	Explain
[31:9]		Not used
[8]	IDLE	Idle frame interrupt flag, active high, clear by writing 1 When the UART detects an idle frame, the hardware sets this flag. If the UARTx_IE.IDLE_IE = 1, an interrupt request is generated. After being cleared by writing 1, the IDLE_IF flag is set only after the RX_DONE_IF flag is set again. In a multi-drop scenario, the IDLE_IF is set only when the slave address hits, that is, when the UARTx_STT.ADR_MATCH = 1, indicating that the master wants to communicate with the slave at this time. Otherwise, even if the UARTx_RX signal is normally high, the IDLE_IF is not set.
[7]	LBD	LIN break character detect interrupt flag, active high, write 1 to clear

[6]	RX_OV	Receive buffer overflow interrupt flag, active high, write 1 to clear
[5]	TX_OV	Transmit buffer overflow interrupt flag, active high, write 1 to clear
[4]	CK_ERR	Check error interrupt flag, active high, write 1 to clear.
[3]	STOP_ERR	Stop bit error interrupt flag, active high, write 1 to clear
[2]	TX_BUF_EMPTY	Transmit buffer empty interrupt flag, active high, write 1 to clear
[1]	RX_DONE	Receive completion interrupt flag, active high, write 1 to clear
[0]	TX_DONE	Transmit completion interrupt flag, active high, write 1 to clear

Write 1 to clear the interrupt flag. It is generally not recommended to use the following method | = to clear, because | = reads the interrupt flag first, changes the corresponding bit to 1, and then writes it to clear. If other interrupt flags are set at the same time, they will be cleared together, which is usually not expected by the software. For example, the following writing method is intended to clear the TX_DONE_IF, but if the RX_DONE_IF is set to 1 before writing, the software will first read back the UART_IF value as 0x2, then execute the or operation 0x2 | 0x1 = 0x3, and then write. At the same time, the RX_DONE_IF and TX_DONE_IF are cleared, which may cause the UART to enter one less interrupt generated by receiving data, thus receiving one less byte of data.

```
UART_IF|=0x1;
```

If you want to clear the TX_DONE_IF flag, write a 1 directly to BIT0 as follows.

```
UART_IF=0x1;
```

17.3.11 UARTx_IOC:UARTx IO Control Register (X = 0,1)

The addresses are: 0x4001_1024, 0x4001_1124

Reset value: 0x0

Table17-13 UARTIO control register UART_IOC.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SBK	LBDL		AUTO			TXD_INV	RXD_INV
								RW	RW		RW			RW	RW
								0	0		0			0	0

Location	Bit name	Explain
[31:8]		Not used
[7]	SBK	In LIN mode, write 1 to send a break character once, that is, 13 consecutive 0s, which will be cleared automatically after completion. If it is not completed, it will be read back as 1. Before writing 1 to this bit, you need to configure the UARTx_CTRL.LIN_EN = 1, otherwise the break character cannot be sent.
[6]	LBDL	LIN break character detection length, 10/11 0s 0:10bits， 1:11bits
[5]		Not used

[4]	AUTO	Baud Rate Adaptive IO Port Enable Switch. 0: Close; 1: On
[3:2]		Not used
[1]	TXD_INV	TXD output polarity enable switch. Default value is 0. 0: normal output; 1: negated output. Normal output polarity, that is, software sends 1 and hardware sends 1; Invert the output polarity, that is, the application sends a 1 and the hardware sends a 0.
[0]	RXD_INV	RXD input polarity enable switch. Default value is 0. 0: normal input; 1: Negative input. Normal input polarity, that is, the hardware receives 1 and the software receives 1; Input polarity is reversed, that is, hardware receives a 1 and software receives a 0.

18 DMA

18.1 Overview

DMA and CPU are the master devices of the chip bus.

As Figure 18-1show. Some of the devices do not need to be accessed by DMA and are only mounted on the bus connected to the CPU. Devices such as ADC/DAC/SPI/I2C/MCPWM/UART/ Timer/GPIO/Hall/SRAM can be shared and accessed by CPU and DMA.

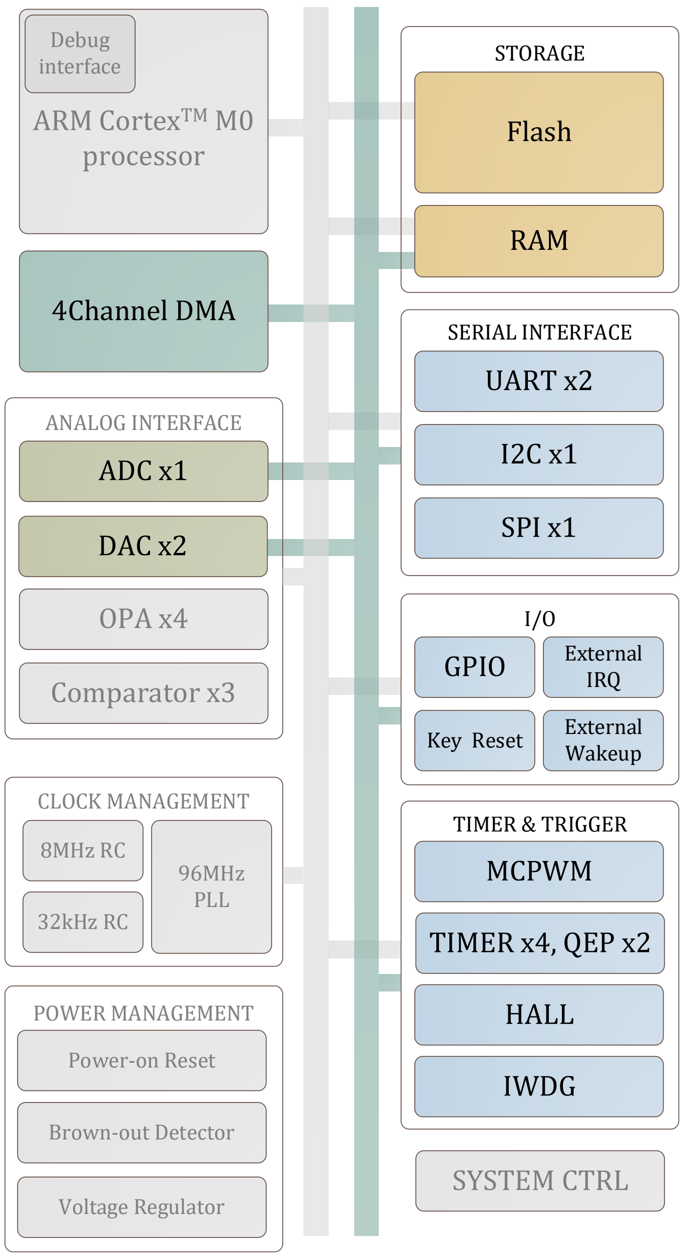


Figure18-1 Multi-layer AHB bus architecture

The DMA has 4 channels and shares a single transport engine. When the DMA is idle and multiple channels have requests at the same time, arbitration is performed according to the priority, but preemption does not occur in the transfer process, that is, a certain channel must complete the transfer, and after the DMA is idle, arbitration is initiated to determine which channel obtains the request next.

There are two ways of DMA transmission: DMA_CCRx.RMODE = 1, multiple rounds, one transmission in each round; DMA_CCR X.RMODE = 0, one round, multiple consecutive transmissions.

If multiple data transfers are configured in one round, i.e. DMA_CCRx.RMODE = 0, the DMA continuously sends DMA_CTMSx times of data for one DMA request, and then the hardware clears the DMA_CCRx.EN bit and sets the interrupt flag.

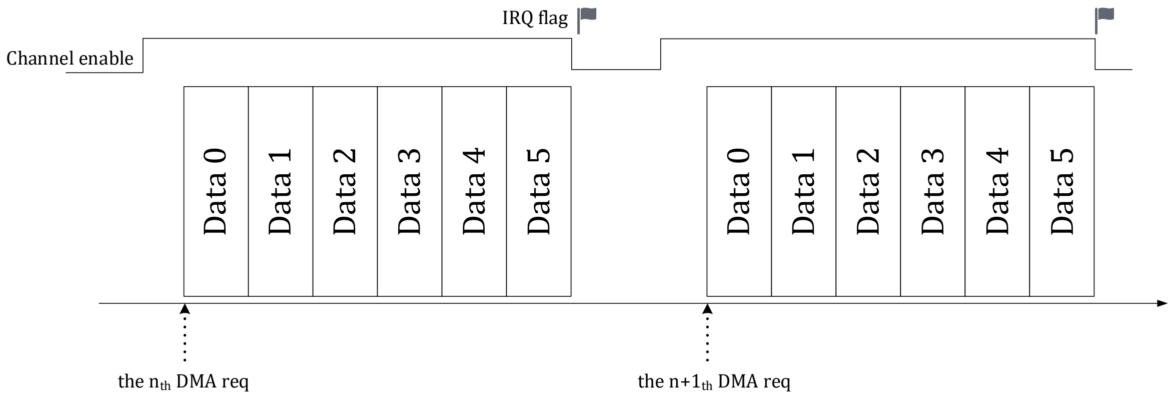


Figure18-2 RMODE = 0 DMA transfer

If a channel is configured for multiple rounds of traffic, the DMA may respond to traffic requests from other channels between rounds of traffic.Each round of multi-round transmission carries data only once. After carrying DMA_CTMSx rounds, the hardware clears the DMA_CCRx.EN bit and sets the interrupt flag.

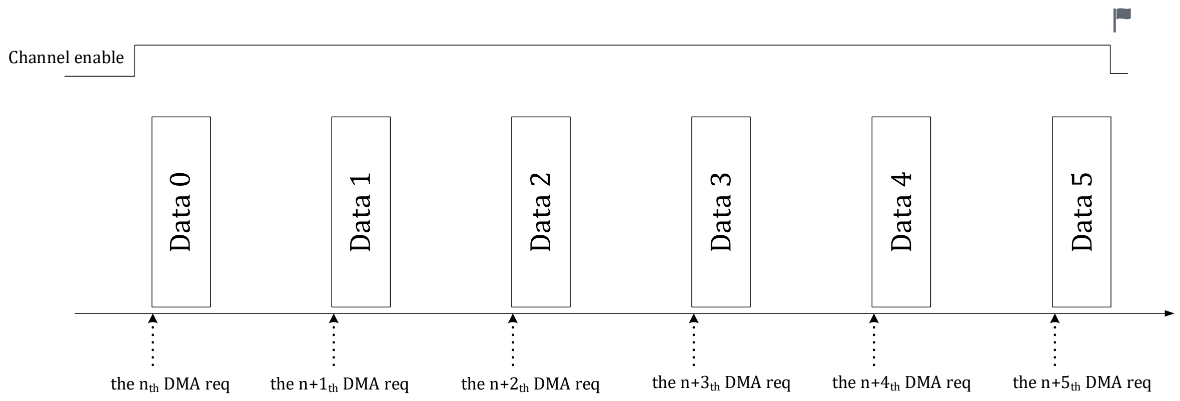


Figure18-3 RMODE = 1 DMA transfer

When a DMA request occurs, the DMA starts a round of transmission, and after the round of transmission is completed, the DMA responds to the next request according to the request condition of the current DMA channel. The next request can still be the request of the current channel, or can be the request of other channels, or enter an idle state under the condition of no request. At the same time, the number of rounds to be transmitted in the current DMA channel is reduced by 1. When the round number of the DMA to be transmitted is 0, the transmission of the current channel is completely completed, the DMA generates an interrupt, and meanwhile, the DMA channel enable DMA_CCRx.EN is automatically cleared by hardware. During the transfer, the number of rounds to be transferred can be read from the DMA_CTMSx register. The number of pending transfers in one transfer does not support reads because the DMA transfers several times in a row, so the number changes quickly.

For power consumption control, the DMA module can be disabled by setting the DMA_CTRL. EN bit to 0 (it is required to turn off the DMA_CCRx.EN corresponding to the channel before turning off the DMA enable). In this case, the DMA clock is gated off.

DMA supports 8, 16, or 32 bit (byte, half-word, or word) transmission operations, and the bit width of source and destination address access is selected by configuring the DMA_CCRx.SBTW and DMA_CCRx.DBTW. The bit width of the source address access may be different from the bit width of the destination address access.

Each time the DMA completes a transfer, the address is automatically incremented based on the DMA_CCRx.SINC and the DMA_CCRx.DINC. All peripheral register addresses are word aligned, so the peripheral address increment should be configured as 0/4. For example, for UARTs/SPI/I2C, because the fixed address of the UART_DATA or the SPI/I2C FIFO interface is accessed each time, there is no need to increment the address between rounds; To access the ADC data register, the address usually needs to be automatically incremented by 4, which can be programmed to increment in a round. For the memory, if in-round increment is set, the value of each address increment is set according to the memory data bit width (DMA_CCRx.SBTW/DBTW). For the case that the bit width of memory access is byte, the address is automatically increased by 1, for half-word, the address is automatically increased by 2, and for word, 4.

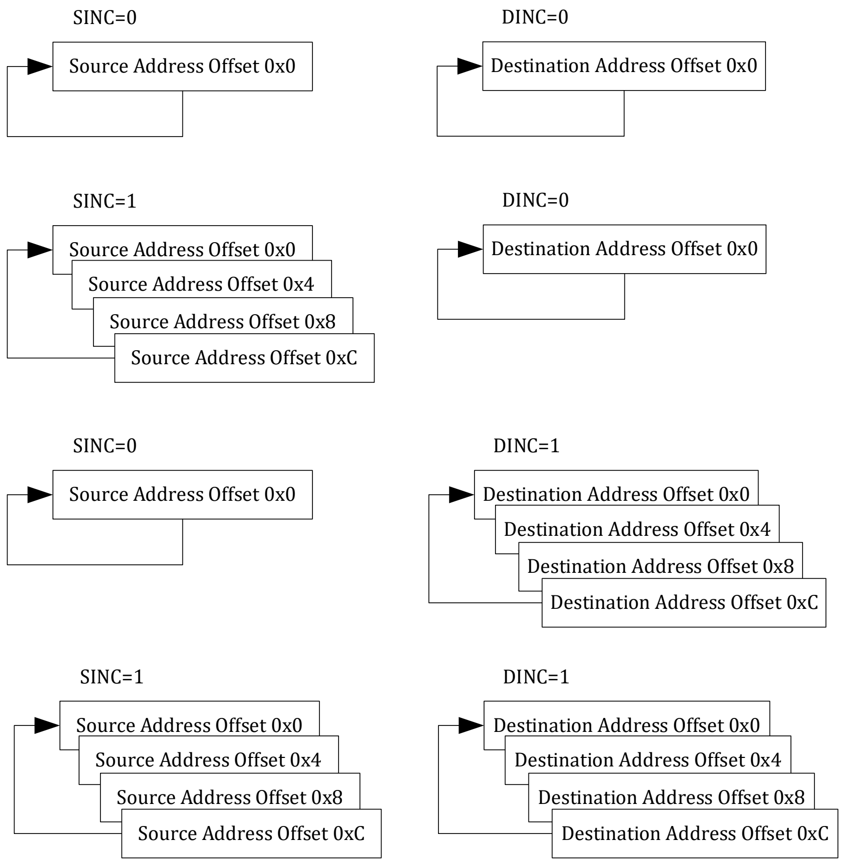

SINC=0
Source Address Offset 0x0
DINC=0
Destination Address Offset 0x0
SINC=1
Source Address Offset 0x0
Source Address Offset 0x4
Source Address Offset 0x8
Source Address Offset 0xC
DINC=0
Destination Address Offset 0x0
SINC=0
Source Address Offset 0x0
DINC=1
Destination Address Offset 0x0
Destination Address Offset 0x4
Destination Address Offset 0x8
Destination Address Offset 0xC
SINC=1
Source Address Offset 0x0
Source Address Offset 0x4
Source Address Offset 0x8
Source Address Offset 0xC
DINC=1
Destination Address Offset 0x0
Destination Address Offset 0x4
Destination Address Offset 0x8
Destination Address Offset 0xC

Figure18-4 DMA address increment control

Figure 18-4as an example of address increment only, it is configured to transmit 4 times in one round, or to transmit 4 rounds, and the data size of the source address and the destination address is word. In practice, if the configuration data size is byte or halfword, the address offset is incremented by 1/2.

In general, SINC = 0, DINC = 0, the source address and destination address are not incremented, which may be the transfer from peripheral to peripheral.

SINC = 1, DINC = 0, the source address is incremented, the destination address is not incremented, which may be the transfer of the memory array to the peripheral register interface.

SINC = 0, DINC = 1, the source address is not incremented, and the destination address is incremented, which may be the transfer of the peripheral register interface to the memory array.

SINC = 1, DINC = 1, both the source address and the destination address are incremented, which is the transfer of multiple groups of peripheral data (such as ADC_DATx) to the memory array.

If the DMA_CCRx.CIRC = 1 and DMA_CCRx.RMODE = 1, that is cyclic multi-round mode. One DMA request signal is triggered, and only one data transfer is performed (for example, UART data is transferred to RAM, and one round of transfer is completed to wait for the next request). In this mode, the DMA_CTMSx can be configured to a larger value, with the DMA_CTMSx decremented by 1 for each pass. Because that interrupt mark is not generate when the circular mode DMA carries out a plurality of round of transportation, the CPU can judge the data amount which is transmitted at present by reading the DMA_CTMSx, and if the data amount is enough, the processing is star. When the

DMA_CTMSx round is transmitted, the address returns to the original address. The DMA_CTMSx is reloaded to the default value and a new round of decrements begins. In round-robin mode, the DMA's channel enable DMA_CCRx.EN bit is set by software and remains high even when the DMA_CTMSx is decremented to 0. If you want to exit the transmission, you can clear the DMA_CCRx.EN by software.

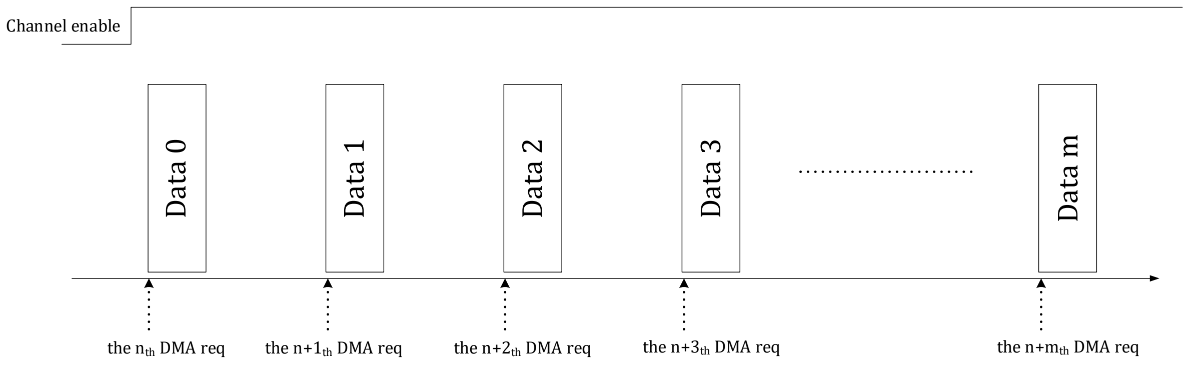


Figure18-5 DMA transfer with CIRC = 1 and RMODE = 1

If the DMA_CCRx.CIRC = 1 and the DMA_CCRx.RMODE = 0, that is, the cyclic single-pass mode, the DMA will continuously carry out data transfer for DMA_CTMSx times after receiving a request, and generate an interrupt flag. In this mode, the DMA_CCRx.EN does not need to be set by software. It is automatically set when the DMA channel receives a request, and is automatically cleared after one round of transmission. To exit this mode, clear the DMA_CCRx.CIRC bit and clear the DMA_CCRx.EN bit.

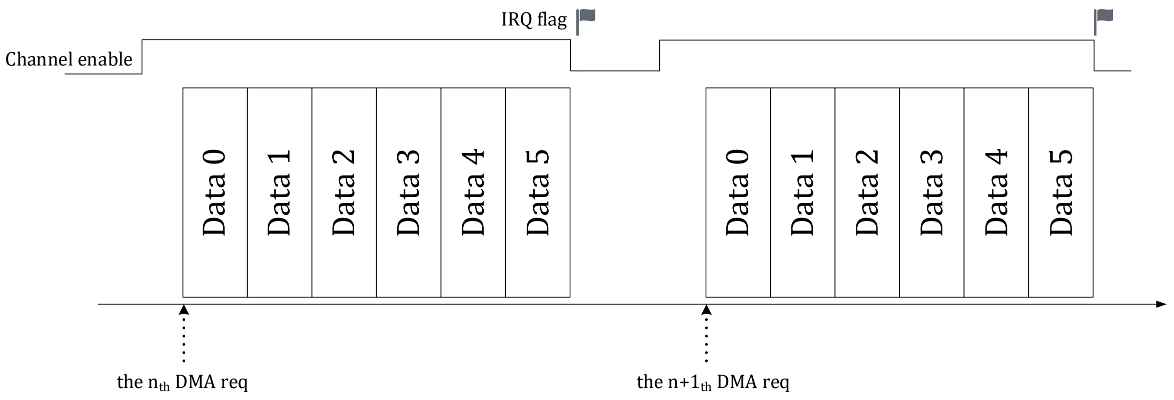


Figure18-6 DMA transfer with CIRC = 1 and RMODE = 0

In cyclic mode, the DMA_CTMSx registers are reloaded with the initial configuration values after being decremented to zero by the transport. If the data is transferred to the memory, the data previously transferred to the memory is overwritten; If it is carried to the peripheral, one round of data transmission is repeated. Taking the transfer of ADC data to the memory as an example, if the data block size is set to 16 bits × 12 channels × 8 times, the DMA will restart the next round of transfer after completing a round of 96 half word transfers in the cyclic mode, and overwrite the previously used memory address without setting the DMA completion interrupt flag bit. In the single mode, the DMA operation is terminated after the DMA completes the transfer of a data block of a certain size, a DMA completion interrupt flag bit of the current channel is set, and the hardware automatically closes the corresponding DMA channel at the same time, that is, the hardware circuit automatically sets the DMA_CCRx.EN to 0 after the transmission of the channel is completed. The number of DMA passes required is controlled by the DMA_CTMS register.

18.2 Request

DMA requests are divided into software requests and hardware requests. Software requests are generated by setting the DMA_RENx.SW_TRIG of the corresponding DMA channel to 1. After the software trigger bit is set, it needs to be cleared by software. A hardware request is usually an interrupt event of a peripheral. When a specific peripheral interrupt event is used as a DMA transfer request, the interrupt response of the corresponding event is usually disabled, that is, the CPU is no longer required to respond to the interrupt, and the response is only used as a DMA transfer request. Moreover, the hardware DMA request signal will be cleared by the DMA hardware after being accepted by the DMA channel, and there is no need for software to clear the peripheral interrupt flag that raises the request.

Table18-1 DMA request

Trigger source	Description
Software	The transfer operation to be performed when triggered by software is specified by the configuration register DMA_CCRx. When the DMA_RENx.SW_TRIG is set, the DMA operation starts as soon as the channel is enabled.
ADC	In the single-stage trigger mode, the ADC generates an interrupt request after completing the sampling of several channels at a time, and the DMA carries the value converted by the ADC to the SRAM. The ADC single-segment sampling completion interrupt event is used as a DMA request signal, and the request signal is cleared by the DMA after receiving a DMA response without software clearing; Note that this requires software to also disable the ADC sample completion interrupt so that the interrupt is no longer responded to by the CPU.
DSP	Interrupt event flags generated by DSP IRQ instructions can be used as DMA request
Timer	The Timer can use the zero-crossing/compare/capture event as a DMA request, and the specific DMA operation is set according to the configuration register, usually as a timing event (such as triggering a DMA operation every 10 ms).
SPI	The SPI module can use the receive buffer full event as the DMA request signal. Since the SPI is sent and received at the same time, the receive buffer full is an event flag for both receiving and sending completion. Read the SPI FIFO auto clear event flag.
MCPWM	The MCPWM module can use the zero-crossing/count cycle end/4 ADC triggers as DMA requests, and the DMA operation is set according to the configuration register.
I2C	The I2C module can use the I2C0_SCR.BYTE_CMPLT, that is, the completion of byte transmission, as a trigger for the DMA request, and the DMA automatically clears the I2C request flag. Other I2C interrupt events are still responded to by the CPU.
UART	The serial port module can use a UART_IF to trigger a DMA request, and if the transmission direction configured by the DMA is from the memory to the serial port, the UART is used to send a buffer empty flag as a DMA request signal; If the transfer direction is serial port to memory, the DMA request signal should be generated using the UART receive completion event. The event flag is automatically cleared by the DMA. The UART shall also disable the corresponding interrupt when operated by the DMA to prevent it from being responded by the CPU.
HALL	HALL interrupt flag can be used as DMA request
SIF	SIF interrupt flag can be used as DMA request*
Comparator	CMP interrupt flag can be used as DMA request

GPIO	GPIO interrupt flag can be used as DMA request

For chips whose SYS_AFE_INFO.VERSION<=3, this request is available; For chips whose SYS_AFE_INFO.VERSION=4, this request is not available.

18.3 Priority

The priority of the DMA is a fixed priority, as shown inFigure 18-7. In order to avoid the situation that it is too late to respond to some peripheral requests, the real-time of the task should be considered when designing the application software, and each channel should not be configured to carry too much data, so that other channels can not respond in time.

Such as Figure18-7, The priority decreases from top to bottom. Of the four DMA channels, the priority relationship is channel 0 > channel 1 > channel 2 > channel 3 (the > sign indicates a higher priority). Within each DMA channel, there are usually multiple hardware request events and one software request event, with hardware requests having a higher priority than software requests. Multiple hardware request events have the same priority. Typically, a DMA channel is configured with one hardware request event enable, and multiple hardware requests should not occur simultaneously within a channel.

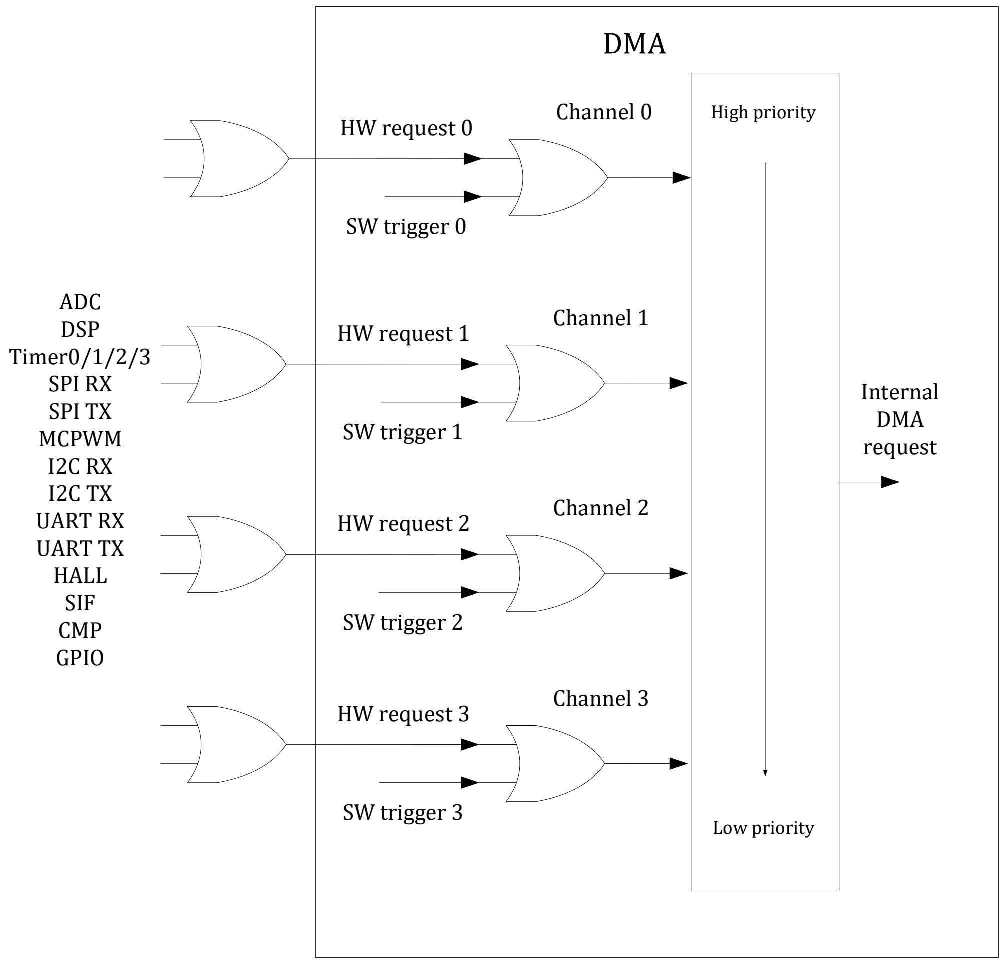


Figure18-7 DMA channel priority

18.4 Arbitration

When the DMA is in an idle state or has just completed the DMA transfer of a certain channel, if one or more DMA requests occur at this time, the DMA will arbitrate according to the priority setting, and the peripheral request with higher priority will be served by the DMA first. For example, in the continuous mode of ADC, the sampling completion event flag of the ADC is cleared by the DMA every time a round of ADC data transfer is completed, and the DMA returns to the idle state or turns to serve other peripheral requests; For UART/SPI/I2C, arbitration is performed again after each byte is transferred.

In order to prevent CPU or DMA from occupying the peripheral/SRAM for a long time, a time slice mechanism is added in a port arbitration module of the peripheral/SRAM, that is, a master device releases the access right after occupying for a period of time, and the arbitration module observes whether another master device requests access, if so, the other master device is allowed to access, otherwise, the access which is not completed by the current master device is continued.

18.5 Interrupt

A DMA interrupt is generated after one of the DMA channels completes a DMA operation. When a DMA channel finishes the operation, it will automatically turn off the enable of the channel through the DMA_CCRx.EN.

18.6 Register

18.6.1 Address assignment

The base address of the DMA controller module register is 0x4001_1200, and the register list is as follows:

Table18-2 DMA register list

Name	Offset address	Explain
DMA_CCR0	0x00	DMA Channel 0 Channel Configuration Register
DMA_REN0	0x04	DMA Channel 0 Request Enable Register
DMA_CTMS0	0x08	DMA Channel 0 Transfer Count Register
DMA_SADR0	0x0C	DMA Channel 0 Source Address Register
DMA_DADR0	0x10	DMA Channel 0 Destination Address Register
DMA_CCR1	0x20	DMA channel 1 channel configuration register
DMA_REN1	0x24	DMA Channel 1 Request Enable Register
DMA_CTMS1	0x28	DMA Channel 1 Transfer Count Register
DMA_SADR1	0x2C	DMA Channel 1 Source Address Register
DMA_DADR1	0x30	DMA Channel 1 Destination Address Register
DMA_CCR2	0x40	DMA Channel 2 Channel Configuration Register
DMA_REN2	0x44	DMA Channel 2 Request Enable Register
DMA_CTMS2	0x48	DMA Channel 2 Transfer Count Register
DMA_SADR2	0x4C	DMA Channel 2 Source Address Register

DMA_DADR2	0x50	DMA Channel 2 Destination Address Register
DMA_CCR3	0x60	DMA Channel 3 Channel Configuration Register
DMA_REN3	0x64	DMA Channel 3 Request Enable Register
DMA_CTMS3	0x68	DMA Channel 3 Transfer Count Register
DMA_SADR3	0x6C	DMA Channel 3 Source Address Register
DMA_DADR3	0x70	DMA Channel 3 Destination Address Register
DMA_CTRL	0x80	DMA control register
DMA_IE	0x84	DMA Interrupt Enable Register
DMA_IF	0x88	DMA Interrupt Flag register

18.6.2 DMA_CTRL DMA control register

Address: 0x4001_1280

Reset value: 0x0

Table18-3 DMA Control Register DMA_CTRL.

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

Location	Bit name	Explain
[31:1]		Not used
[0]	EN	DMA global enable

18.6.3 DMA_IE DMA Interrupt Enable Register

Address: 0x4001_1284

Reset value: 0x0

Table18-4 DMA Interrupt Enable Register DMA_IE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												CH3_FIE	CH2_FIE	CH1_FIE	CH0_FIE
												RW	RW	RW	RW
												0	0	0	0

Location	Bit name	Explain
[31:4]		Not used
[3]	CH3_FIE	Channel 3 done interrupt enable
[2]	CH2_FIE	Channel 2 done interrupt enable
[1]	CH1_FIE	Channel 1 done interrupt enable
[0]	CH0_FIE	Channel 0 done interrupt enable

18.6.4 DMA_IF DMA interrupt flag register

Address: 0x4001_1288

Reset value: 0x0

Table18-5 DMA Interrupt Flags Register DMA_IF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												CH3_FIF	CH2_FIF	CH1_FIF	CH0_FIF
												RW1C	RW1C	RW1C	RW1C
												0	0	0	0

Location	Bit name	Explain
[31:4]		Not used
[3]	CH3_FIF	Channel 3 completion interrupt flag, active high, write 1 to clear
[2]	CH2_FIF	Channel 2 completion interrupt flag, active high, write 1 to clear
[1]	CH1_FIF	Channel 1 completion interrupt flag, active high, write 1 to clear
[0]	CH0_FIF	Channel 0 completion interrupt flag, active high, write 1 to clear

18.6.5 DMA Channel Configuration Register

18.6.5.1 DMA_CCRx (where x =0,1,2,3)

The addresses are: 0x4001_1200, 0x4001_1220, 0x4001_1240, 0x4001_1260

Reset value: 0x0

Table18-6 DMA channel configuration register DMA_CCRx.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				SBTW		DBTW			SINC		DINC	CIRC		RMODE	EN
				RW		RW			RW		RW	RW		RW	RW
				0		0			0		0	0		0	0

Location	Bit name	Explain
[31:12]		Not used
[11:10]	SBTW	Source Address Access Bit Width 0: Byte 1: Halfword 2: Word 3: Reserved

[9:8]	DBTW	Destination address access bit width 0: Byte 1: Halfword 2: Word 3: Reserved
[7]		Not used
[6]	SINC	Source Address Increment Mode 0: No increment 1: The address increases by 1/2/4 according to the corresponding size of SBTW for each transmission
[5]		Not used
[4]	DINC	Destination Address Increment Mode 0: No increment 1: The address increases by 1/2/4 according to the corresponding size of DBTW for each transmission
[3]	CIRC	Cycle mode, high efficiency
[2]		Not used
[1]	RMODE	0: Single round of transmission, one round of continuous transmission for many times, every time a DMA request is received, a DMA request is transmitted. 1: Multiple rounds, one data transmission is performed in each round, and multiple DMA requests are transmitted after one round of DMA request transmission is received Multiple rounds × multiple transfers are not supported
[0]	EN	Channel enable, high active, software set 1 to open the channel for DMA transfer operation, DMA hardware will clear this bit after the transfer is completed

18.6.5.2 DMA_RENx (where x = 0,1,2,3)

The addresses are: 0x4001_1204, 0x4001_1224, 0x4001_1244, 0x4001_1264

Reset value: 0x0

Table18-7 DMA request enable register DMA_RENx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SW_REN		GPIO_REN			CMP_REN	SIF_REN	HALL_REN					UART1_TX_REN	UART1_RX_REN	UART0_TX_REN	UART0_RX_REN
RW		RW			RW	RW	RW					RW	RW	RW	RW
0		0			0	0	0					0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C0_TX_REN	I2C0_RX_REN	MCPWM1 REN	MCPWM0_REN	SPI TX_REN	SPI RX_REN			TIMER3_REN	TIMER2_REN	TIMER1_REN	TIMER0_REN	DSP_REN		ADC1_REN	ADC0_REN
RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW		RW	RW
0	0	0	0	0	0			0	0	0	0	0		0	0

Generally, the memory and peripheral address configured in the DMA channel shall correspond to the enabled peripheral interrupt request, which shall be guaranteed by the application software. The software request is always enabled, that is, the software writes a 1 to the DMA_RENx.SW_TRIG to start a DMA transfer. A hardware request from a peripheral enters the DMA to form a request signal through the or logic, and each DMA channel generally enables only one hardware DMA request at a time. The software trigger flag DMA_RENx.SW_TRIG is automatically cleared by the hardware after the request is accepted by the DMA.

Since the CMP signal may be a slow level signal, it is recommended to select the interrupt trigger type as edge triggered when using the CMP triggered DMA.

The DMA_RENx register can be overwritten if the DMA channel is enabled.

18.6.5.3 DMA_CTMSx (where x = 0,1,2,3)

The addresses are: 0x4001_1208, 0x4001_1228, 0x4001_1248, 0x4001 — 1268

Reset value: 0x0

Table18-8 DMA transfer count register DMA_CTMSx.

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

Location	Bit name	Explain
[7:0]	TMS	DMA channel X number of data transfers. This register becomes read-only when the channel is enabled.

The DMA_CTMSx register can be written to only after the channel is disabled, that is, DMA_CCRx.EN = 0.

When the DMA_CTRL = 1 and the DMA_CCRx.EN = 0, refill the CTMSx value could clear the DMA internal carried rounds counter.

When the DMA_CCRx.RMODE = 0, it indicates that one round of data is transmitted, the number of transmissions per round is DMA_CTMS.

When the DMA_CCR X.RMODE = 1, it indicates that the DMA_CTMS rounds are transmitted, and one data is transmitted in each round;

Multi-round transfers with multiple data transfers per round are not supported.

For example, if the peripheral data width is 16 and the memory data width is 32, that is, the DMA_CTMSx = 16 and the DMA_CCR X.RMODE = 0, the DMA needs to read peripheral data 16bit = 2byte and write memory data 32bit = 4byte in each round. A total of 16 rounds of handling are required, that is, 32 bytes of peripherals are read and 64 bytes are stored in the memory;

Even if only one round is carried, it is necessary to set the CTMS.ROUND = 1 and not let it be 0.

When the DMA_CCRx.CIRC = 1 is set (i.e., the circulation mode), the DMA_CTMSx no longer works, which is equivalent to an infinite wheel; In other cases, the DMA_CTMSx needs to be set accordingly, for example, when DMA_CTMSx = 1 and DMA_CCRx.RMODE = 1, it is used to carry a round of data.

When the DMA_CCRx.RMODE = 1, the DMA_CTMSx reads the current number of remaining rounds, and the number of rounds is reset to the configured value after the DMA transfer is completed. The number of times is always the configured value when it is read. For example, if the configuration DMA_CTMSx =4, when reading, you may see that the DMA_CTMSx decreases from 4 to 3, 2, 1 And then it becomes 0. When the current DMA channel needs to perform four rounds of transmission again, the round value needs to be written to the DMA_CTMSx again. When the DMA_CCRx.RMODE = 0, the DMA_CTMSx reads out the number of the remaining non-transportation times of the current round. However, because the data in a round is continuously carried by DMA, the DMA_CTMSx read by software will change rapidly.

Note that the DMA_CTMSx register is reconfigured each time the DMA channel is restarted, and the DMA_CTMSx register has been decremented to 0 in the last transfer.

18.6.5.4 DMA_SADRx (where x = 0,1,2,3)

The addresses are: 0x4001_120C, 0x4001_122C, 0x4001_124C, 0x4001 — 126C

Reset value: 0x0

Table18-9 DMA Source Address Register DMA_SADRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															
0															

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

Location	Bit name	Explain
[31:0]	ADDR	DMA channel X source addr

When the DMA_CCRx.SBTW = 2'b01, the source data is transferred in units of 16 bits. The DMA_SADRx.ADDR[0] value is invalid, and the peripheral address is increments by 2.

When the DMA_CCRx.SBTW = 2'b10, the peripheral data is configured to be transferred in units of 32 bits. The DMA_SADRx.ADDR [1:0] value is not valid. the peripheral address is increments by 4.

Note that only after the channel is disabled, that is, DMA_CCRx.EN = 0,data can be written to the DMA_SADRx register!!!

18.6.5.5 DMA_DADRx (where x = 0,1,2,3)

The addresses are: 0x4001_1210, 0x4001_1230, 0x4001_1250, 0x4001_1270

Reset value: 0x0

Table18-10 DMA destination address register DMA_DADRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
RW															
0															

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

Location	Bit name	Explain
[31:0]	ADDR	DMA channel X destination addr

When the DMA_CCR X.DBTW = 2'b01, memory data is transferred in 16-bit units. DADRx. ADDR[0] value is invalid, the memory address is incremented by 2.

When the DMA_CCR X.DBTW = 2'b10, memory data is transferred in 32-bit units. DADRx.ADDR [1:0] is not valid, the memory address is incremented by 4..

Note that only after the channel is disabled, that is, DMA_CCRx.EN = 0,data can be written to the DMA_DADRx register !!!

19 DSP

19.1 Overview

The DSP module uses a DSP instruction set designed independently, and can carry out single-cycle arithmetic instructions such as addition and subtraction, or, multiplication and accumulation, shift, saturation,etc, and multi-cycle arithmetic operation instructions such as division, square root, trigonometric function,etc; Memory access instructions such as load/store, branch instructions such as unconditional jump and conditional jump, and miscellaneous instructions such as interrupt raising. There are pseudo instructions such as breakpoint instruction and register assignment that can be used for debugging on the simulator.

DSP has two operation modes: autonomous operation and passive call.

The so-called autonomous operation means that the DSP reads the instructions in the CODE MEM and the data in the DATA MEM to execute the DSP program, which is independent of the ARM Cortex M0. At this time, the DSP0_SC.PAUSED = 0, that is, the DSP is in operation. CODE MEM and DATA MEM allow DSP access but not CPU access to overwrites.

Passive calling means that the DSP is called by ARM Cortex M0 as a peripheral module, and the CPU directly accesses the arithmetic operation resources inside the DSP, such as division, square root, trigonometric function, etc. At this time, DSP0_SC.PAUSED = 1, that is, the DSP does not run the DSP program and is in the pause state. CODE MEM and DATA MEM allow the CPU to access and rewrite. For users who do not develop DSP programs, it is recommended to use this mode to directly call the arithmetic unit of the DSP through the software run by the CPU.

The DSP is equipped with separate program memory (CODE MEM) and data memory (DATA MEM). The two independent storage areas can be accessed through the CPU when the DSP is paused, that is, the DSP0_SC.PAUSED= 1, and during the initialization of the DSP, a program run by the DSP and initial data are required to be respectively written into a CODEMEM and a DATAMEM of the DSP by the CPU. The DSP has an instruction to raise the interrupt. After the interrupt is set, the DSP enters the pause state at the same time. At this time, the CPU is allowed to access the DATA MEM through the bus interface to interact with the DSP, including reading the DSP operation results and writing the data required for the subsequent operation of the DSP.

In addition, in order to make full use of the DSP, the CPU is allowed to directly access the DSP divider, square root, trigonometric function and other operation modules through the DSP register interface when the DSP is suspended, that is, the CPU is allowed to use the DSP as a simple operation co-processing module.

19.1.1 Functional block diagram

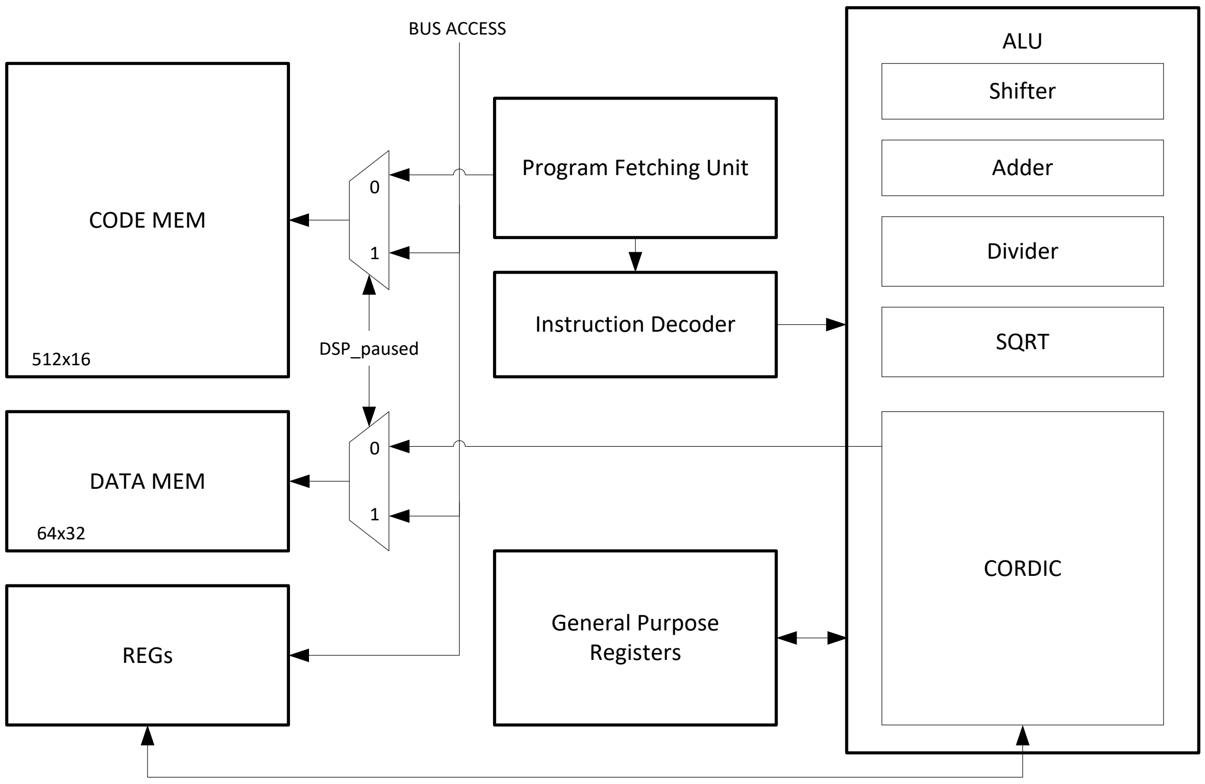


Figure19-1 Functional block diagram of DSP module

19.1.2 DSP Core Registers

Table19-1 DSP Core Registers

Register	Bit Width	Usage
R0	32	Always read as 0
R1	32	
R2	32	
R3	32	
R4	32	
R5	32	
R6	32	ARCTAN module destination
R7	32	MAC result / DIV dividend
PC	16	Program Counter

The R0 register is a constant 0 register, data cannot be written, and the read back is constant 0. The R0 register can be used as a special operand to construct some pseudo-instructions. Such as

ADD R0 R0 R0 is equivalent to NOP

MAC R1 R2 R0 is equivalent to MUL R1 R2, that is, the number accumulated in the multiplication accumulation is 0, which becomes the multiplication operation.

The R6 register is fixed in the ARCTAN instruction to hold the magnitude of the vector

The R7 register is fixed in the MAC instruction to hold the result of the calculation and in the DIV instruction to hold the dividend operand.

The above convention is due to the limitation of fixed-length instruction encoding, which requires a fixed operand in the 4-operand instruction to use the convention register.

19.1.3 Bit width

The dividend and quotient of the division are both 32-bit signed numbers, and the divisor and remainder are 32-bit signed numbers.

The radicand is a 32-bit unsigned number and the square root is a 16-bit unsigned number.

Both multipliers of the multiply accumulation are 32-bit signed numbers, and the sum of the addends is a 32-bit signed number.

The bit width of the trigonometric function CORDIC module is 16 bits, in Q15 fixed-point number format.

Note: When using DSP arithmetic operation resources, whether it is called by the CPU or through DSP arithmetic instructions, be sure that the operands do not exceed the representation range, otherwise a calculation exception will occur.

When the divisor is 0, the quotient is always 0 and the remainder is the dividend, regardless of the value of the dividend.

When the dividend is 0x8000_0000 and the divisor is -1, the quotient exceeds the representation range of the 32bit signed number, and the saturation is 0x7fff_ffff.

Table 19-2 Table of special operands of division

Dividend DSP0_DID		Divisor DSP0_DIS		Quotient DSP0_QUO		Remainder DSP0_REM	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0x7FFF_FFFF	2^{31}-1	0x8000_0000	-2^{31}	0x0	0	0x7FFF_FFFF	2^{31}-1
0x7FFF_FFFF	2^{31}-1	0x7FFF_FFFF	2^{31}-1	0x1	1	0x0	0
0x8000_0000	-2^{31}	0x7FFF_FFFF	2^{31}-1	0xFFFF_FFFF	-1	0xFFFF_FFFF	-1
0x8000_0000	-2^{31}	0x8000_0000	-2^{31}	0x1	1	0x0	0
0x8000_0000	-2^{31}	0xFFFF_FFFF	-1	0x7FFF_FFFF	2^{31}-1	0x0	0
0x7FFF_FFFF	2^{31}-1	0xFFFF_FFFF	-1	0x8000_0001	1-2^{31}	0x0	0
0x1234	4660	0x0	0	0x0	0	0x1234	4660
-0x1234	-4660	0x0	0	0x0	0	-0x1234	-4660

In addition, because the CORDIC operation is based on the successive approximation of multiple rotations, the calculation error of the calculation results is not more than 0.1%.

19.1.4 Instruction Cycle

The divide instruction requires 12 bus cycles (96 MHz) to complete.

The root instruction requires eight bus cycles (96 MHz) to complete.

A trigonometric function instruction requires 20 bus cycles (96 MHz) to complete.

The remaining instructions are single-cycle instructions.

19.1.5 Address space

Table19-3 DSP address space

Duan	Module	CPU software-view address spaces	DSP software-view address spaces	Actual bank size/description
CODE	DSP Code Memory	0x4001_2000 ~ 0x4001_27FF	CODE space: 0x0 ~ 0x1FF	512 x 16bit
DATA	Reserved	0x4001_2800 ~ 0x4001_2EFF		Reserved
	DSP Data Memory	0x4001_2F00 ~ 0x4001_2FEF	DATA space: 0x0 ~ 0x3B	60x32bit
	DSP0_PDI	0x4001_2FF0	DATA space: 0x3C	GPIO_PDI register
	DSP0_BSRR	0x4001_2FF4	DATA space: 0x3D	GPIO_PDO set register
	DSP0_BRR/DSP0_OP	0x4001_2FF8	DATA space: 0x3E	GPIO_PDO Clear Register/DSP Operand Register
	DSP0_CL/DSP0_RES	0x4001_2FFC	DATA space: 0x3F	CL_OUTPUT register/DSP result register
REG	DSP register	0x4001_3000 ~ 0x4001_37FF		
		0x4001_3800 ~ 0x4001_3FFF		Reserved

The address space of DSP is divided into four segments: program segment, data segment, register segment and reserved segment. Each segment occupies 2kB address space, but the actual CODE MEM and DATA MEM storage space is less than 2kB. The program segment (CODE MEM) is used to store the program code required for DSP operation, which is a single-port SRAM and can complete the read-write operation in a single cycle; The data segment (DATA MEM) is used to store the data required for DSP operation. It is a single-port SRAM that completes the read-write operation in a single cycle. The register section is a DSP register allowing the CPU to access through the bus; The reserved segment is not used yet.

The CODE MEM bit width is 16, but it is still addressed by word, that is, the address is incremented by 4 each time.

CODE MEM and DATA MEM only support addressing by word, that is, writing 32 bits and reading 32 bits at a time.

The DSP address space needs to be accessed at the appropriate time. The state control register of the DSP can be accessed at any time; The CODE MEM, the DATA MEM of the DSP, and the CODIC trigonometric function module, the DIV divider, and the SQRT extractor in the register section can access registers only when the DSP is paused, that is, when the DSP0_SC.PAUSED = 1. Because all the arithmetic units may be used by the DSP during the operation of the DSP, the access of the CPU through the register interface at the same time will cause access conflicts. Therefore, during DSP operation,that is, DSP0_SC.PAUSED = 0, access to the arithmetic unit of the DSP via the register interface is prohibited.

When the DSP encounters an interrupt instruction, it will wait for the CPU to process, and at the same time, the DSP enters the pause state, and the DSP0_SC.PAUSED will be set to 1. In addition, the software can also write DSP0_SC.PAUSED = 1 to make the DSP enter the pause state at any time. This mechanism is mainly to prevent the DSP from running permanently without power failure because there is no IRQ instruction in the program written by the DSP.

The following table lists the actual content accessed when the CPU and DSP read and write access the DSP data Memory.

Table 19-4 DSP data memory space read-write access content

DSP Addressing address	DSP		CPU Addressing address	CPU	
	Read	Write		Read	Write
0	Data[0]	Data[0]	0x4001_2F00	Data[0]	Data[0]
1	Data[1]	Data[1]	0x4001_2F04	Data[1]	Data[1]
2	Data[2]	Data[2]	0x4001_2F08	Data[2]	Data[2]
⋮	⋮	⋮	⋮	⋮	⋮
0x3B	Data[59]	Data[59]	0x4001_2FEC	Data[59]	Data[59]
0x3C	DSP0_PDI		0x4001_2FF0		
0x3D		DSP0_BSRR	0x4001_2FF4		
0x3E	DSP0_OP	DSP0_BRR	0x4001_2FF8	DSP0_OP	DSP0_OP
0x3F	DSP0_CL	DSP0_RES	0x4001_2FFC	DSP0_RES	DSP0_RES

Where Data [n] represents the nth word in the DSP DATA Memory.

The DSP0_RES register is the word of the DSP data memory at 0x3F, the last word, which is a copy of Data [63]. For the CPU software, the read and write operations 0x4001_2FFC operate on the DSP0_RES register, and the CPU software can directly read and write DSP0_RES, even when the DSP is running. For DSP, DSP reads 0x3F to get the value of DSP0_CL, and writes 0x3F address to overwrite the value of DSP0_RES register.

DSP0_OP is the word whose DSP data memory is located at 0x3E, that is, the penultimate word. For the CPU software, the read and write operations 0x4001_2FF8 operate on the DSP0_OP register. The CPU software can directly read and write DSPO_OP, even when the DSP is running. For the DSP, the DSP reads 0x3D to get the value of DSP0_OP, and the DSP writes 0x3D address to write the DSP0_BRR register, that is, the GPIO bit is cleared.

19.1.6 Interaction with CPU, GPIO, CL module

The last four words in the DSP data segment are used for special purposes and are no longer used as DATA Memory.

When DSP reads DATA Memory at 0x3C through assembly instruction, it actually reads DSP0_PDI;

When the DSP writes data to the DATA MEM Address 0x3D or 0x3E via an assembly instruction, it is actually writing data to the GPIOx_BSRR, GPIOx_BRR register for bit manipulation of the GPIO_PDO. Refer to the DSP Registers section for specific GPIO mapping.

When the DSP reads DATA MEM Address 0x3F via an assembly instruction, it is actually reading the CL_OUTPUT.

In addition, the 0x3E and 0x3F addresses allow the CPU to interact with the DSP while the DSP is running. DSP0_OP is generally used for CPU to write data and to read data, for example, when DSP is used to simulate serial port transmission; DSP0_RES is generally used when DSP writes data and CPU reads data, for example, when DSP is used to simulate serial port reception. While the DMA accepts the transfer request of the DSP, the DSP0_SC.IF and the DSP0_SC. PAUSED will be cleared and restarts the operation of the DSP, the data transfer is performed after the DSP accepts the request, a read-write channel is required to allow data interaction during the period when the DSP allows.

19.2 Register

19.2.1 Address assignment

The base address of the DSP module in the chip is 0x4001_2000. The base address of the DSP registers in the chip is 0x4001_3000.

Table19-5 DSP register list

Name	Offset	Explain
DSP0_SC	0x00	DSP Status Control Register
DSP0_THETA	0x04	DSP sin/cos Input Angle Register
DSP0_X	0x08	DSP arctan/module Compute Input Coordinate X Register
DSP0_Y	0x0C	DSP arctan/module Compute Input Coordinate Y Register
DSP0_SIN	0x10	DSP sin/cos calculation result sin register
DSP0_COS	0x14	DSP sin/cos calculation result cos register
DSP0_MOD	0x18	DSP arctan calculation result sqrt (X^2+ Y^2) register
DSP0_ARCTAN	0x1C	DSP arctan calculation result arctan (Y/X) angle register
DSP0_DID	0x20	DSP division operation dividend
DSP0_DIS	0x24	DSP Division Operation Divisor
DSP0_QUO	0x28	DSP division quotient
DSP0_REM	0x2C	DSP division operation remainder
DSP0_RAD	0x30	Radicand of DSP Rooting Operation
DSP0_SQRT	0x34	DSP square root operation
DSP0_PC	0x38	DSP Program Pointer (Program Count)

19.2.2 DSP0_SC

Address: 0x4001 3000

Reset value: 0x2

Table19-6 DSP status control register DSP0_SC

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					DMA_ACC_EN	RE	IE					RESET_PC	CORDIC_MODE	PAUSED	IF
					RW	RW	RW					WO	RW	RW	RWIC
					0	0	0					0	0	1	0

Location	Bit name	Explain

[31:11]		Not used
[10]	DMA_ACC_EN	When RE = 1, DMA clears PAUSED and RESET_PC when clearing if. 0: Disabled, DMA only clears IF after receiving the request 1: Enable, DMA clears if, PAUSED and RESET_PC after receiving the request. DMA_ACC_EN = 1 is usually used for DSP software to simulate the transmission and reception of serial communication interface.
[9]	RE	DSP DMA request enable, active high, generates DMA request when RE = 1 and IF = 1 The DMA receives the request and accepts the request to start the data transfer, and clears the IF flag. The software is no longer required to clear the IF.
[8]	IE	DSP interrupt enable, active high. Generates an interrupt request when IE = 1 and IF = 1. IF is asserted when DSP reaches the IRQ instruction, even if IE = 0.
[7:4]		Not used
[3]	RESET_PC	Write 1 to reset DSP PC to address 0 when DSP is paused
[2]	CORDIC_MODE	CORDIC mode, 0: arctan, 1: sin/cos
[1]	PAUSED	Indicates whether the DSP is in a halt state. This bit is set when the DSP reaches an IRQ instruction. Software writes can set this bit to 1. DSP can be started by clearing this bit through software. 0: DSP is reading CODE MEM and DATA MEM to run DSP program autonomously 1: DSP pauses instruction fetching, allowing ARM software to access DSP internal arithmetic unit through register, write operand to register to trigger operation, and read register to obtain operation result For users who do not write DSP CODE MEM, PASUED should remain at 1
[0]	IF	DSP interrupt flag, write 1 to clear

Note that the DSP is in the pause state after the reset, that is, when DSP0_SC.PAUSED = 1; The DSP0_SC.CORDIC_MODE is used to select the sin/cos mode and arctan mode when the CORDIC block is accessed by the CPU through the register interface. The CORDIC block uses the same hardware circuit to calculate sin/cos or arctan. Therefore, before performing a certain calculation, select the appropriate mode by configuring the DSP0_SC calculator.

The CPU can call the DSP through the register interface only when the DSP is in the halted state. When the DSP is running autonomously, the CPU cannot access the internal resources of the DSP through the register interface.

The DSP0_SC.CORDIC_MODE bit needs to be set only when the CPU calls the DSP CORDIC unit through the register interface. The DSP program can switch modes directly according to the SIN_COS or ARCTAN instruction, and the DSP0_SC.CORDIC_MODE is no longer effective.

When the software calls the CORDIC module to calculate sin/cos, the angle DSP0_THETA is used as the input, and the sin/cos result is calculated and output to the DSP0_SIN/DSPO_COS register; Calculates and outputs the angle theta = arctan (y/X) and module = sqrt (X^2+ y^2) to the DSP0_ARCTAN and DSP0_MOD registers with the coordinates DSPO_X/DSPO_Y as input for the arctan calculation.

19.2.3 DSP sin/cos Related Register

The CORDIC module uses the same data path to calculate sin/cos and arctan. Therefore, when the CORDIC module is used by the CPU to calculate sin/cos, it is necessary to write the DSP0_SC.CORDIC_MODE as 1 first, so that CORDIC enters the sin/cos mode.

19.2.3.1 DSP0_THETA

Address: 0x4001_3004

Reset value: 0x0

Table19-7 DSP sin/cos angle input register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_THETA [15] } }
[15:0]	THETA	DSP sin/cos Input Angle Register

DSP0_THETA is a 16-bit signed fixed-point number indicating that the range (-32768 to 32767) corresponds to (– π to π).

19.2.3.2 DSP0_SIN

Address: 0x4001 3010

Reset value: 0x0

Table19-8 DSP sin/cos sine result register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_SIN [15] } }
[15:0]	SIN	DSP sin/cos calculation result sin register

DSP0_SIN is a 16-bit signed fixed-point number, in which 1bit is the sign bit and 15bit is the decimal bit; Indicates the range (– 1 ~ 1).

19.2.3.3 DSP0_COS

Address: 0x4001 3014

Reset value: 0x0

Table19-9 DSP sin/cos Cosine Result Register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_COS [15] } }
[15:0]	COS	DSP sin/cos calculation result cos register

DSP0_COS is a 16-bit signed fixed-point number, in which 1bit is the sign bit and 15bit is the decimal bit; Indicates the range (– 1 ~ 1).

19.2.4 DSP arctan Related Register

19.2.4.1 DSP0_X

Address: 0x4001 3008

Reset value: 0x0

Table19-10 DSP arctan/module Coordinate X Input Register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_X [15] } }
[15:0]	X	DSP arctan/module Compute Input Coordinate X Register

DSP0_X is a 16-bit signed fixed-point number in Q15 format, where 1 bit sign bit and 15 bit integer bits represent the range (– 32768 to 32767).

19.2.4.2 DSP0_Y

Address: 0x4001_300C

Reset value: 0x0

Table19-11 DSP arctan/module Compute Coordinate Y Input Register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_Y [15] } }
[15:0]	Y	DSP arctan/module Compute Input Coordinate Y Register

DSP0_Y is a 16-bit signed fixed-point number, where 1 bit of sign and 15 bit of integer indicates the range (– 32768 to 32767).

19.2.4.3 DSP0_MOD

Address: 0x4001 3018

Reset value: 0x0

Table19-12 DSP arctan vector magnitude result sqrt (X^2+ Y^2) register

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

Location	Bit name	Explain
[31:16]		Reserved, always read as 0
[15:0]	MOD	DSP arctan calculation result sqrt (X^2+ Y^2) register

DSP0_MOD is a 16-bit unsigned fixed-point number, and the 16 bits are all integer bits, indicating the range (0 to 65535).

19.2.4.4 DSP0_ARCTAN

0x4001_301C

Reset value: 0x0

Table19-13 DSP arctan angle result arctan (Y/X) angle register

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

Location	Bit name	Explain
[31:16]		Reserved, read out with sign extension, i.e. { 16 { DSP0_ARCTAN [15] } }
[15:0]	ARCTAN	DSP arctan calculation result arctan (Y/X) angle register

DSP0_ARCTAN is a 16-bit signed fixed-point number indicating that the range (-32768 to 32767) corresponds to (– π to π).

19.2.5 **DSP Division Related Register**

19.2.5.1 DSP0_DID

Address: 0x4001 3020

Reset value: 0x0

Table19-14 DSP divide dividend register

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

Location	Bit name	Explain
[31:0]	DID	DSP divide dividend register

19.2.5.2 DSP0_DIS

0x4001_3024

Reset value: 0x0

Table19-15 DSP Divide Divisor Register

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

Location	Bit	Explain

	name	
[31:0]	DIS	DSP Divide Divisor Register

19.2.5.3 DSP0_QUO

0x4001_3028

Reset value: 0x0

Table19-16 DSP Division Quotient Register

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

Location	Bit name	Explain
[31:0]	QUO	DSP quotient register

19.2.5.4 DSP0_REM

Address: 0x4001 302C

Reset value: 0x0

Table19-17 DSP Division Remainder Register

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

Location	Permissions	Explain
[31:0]	REM	DSP Division Remainder Register

When the CPU needs to use the DSP divider, first ensure that the DSP is in a pause state. Write the dividend to the DSP first, and then write the divisor; Writing to the divisor triggers a division operation. A 32/32-bit division requires 12 cycles to complete. Reading the result DSP0_QUO or DSP0_REM during the 32 cycles causes the CPU to wait for the division to complete and return the result via the bus.

19.2.6 DSP Root Correlation Register

19.2.6.1 DSP0_RAD

Address: 0x4001 3030

Reset value: 0x0

Table19-18 DSP radicand register

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

Location	Bit name	Explain
[31:0]	RAD	DSP radicand register

19.2.6.2 DSP0_SQRT

0x4001_3034

Reset value: 0x0

Table19-19 DSP Square Root Register

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

Location	Bit name	Explain
[31:16]		Reserved, always 0 when read out
[15:0]	SQRT	DSP Square Root Register

When the CPU needs to use the DSP square extractor, the DSP shall be ensured to be in the pause state first. Writing the radicand to the DSP; Writing the radicand can trigger a root operation, which takes 8 cycles to complete the 32-bit root, during which reading the root result DSP0_SQRT will cause the CPU to enter the waiting state, waiting for the root calculation to complete and return the calculation result through the bus.

19.2.7 DSP0_PC

0x4001_3038

Reset value: 0x0

Table19-20 DSP Program Pointer Register DSP0_PC

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

Location	Bit name	Explain
[31:9]		Reserved
[8:0]	PC	Allows software to write directly to the PC value when the DSP is in a halted state. The PC registers are readable whether or not the DSP is in a halted state

19.2.8 DSP0_PDI

Through DSP assembler access, software cannot directly access, corresponding DSP DATA Memory address: 0x3C

Reset value: 0x0

Table19-21 DSP0_PDI Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P2.14	P2.7	P2.6	P2.5	P2.4	P2.3	P1.0	P0.15	P0.14	P0.13	P0.12	P0.11	P0.4	P0.3	P0.2	P0.0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	P2.14	GPIO2_PDI[14]
[14]	P2.7	GPIO2_PDI[7]
[13]	P2.6	GPIO2_PDI[6]
[12]	P2.5	GPIO2_PDI[5]
[11]	P2.4	GPIO2_PDI[4]
[10]	P2.3	GPIO2_PDI[3]
[9]	P1.0	GPIO1_PDI[0]

[8]	P0.15	GPIO0_PDI[15]
[7]	P0.14	GPIO0_PDI[14]
[6]	P0.13	GPIO0_PDI[13]
[5]	P0.12	GPIO0_PDI[12]
[4]	P0.11	GPIO0_PDI[11]
[3]	P0.4	GPIO0_PDI[4]
[2]	P0.3	GPIO0_PDI[3]
[1]	P0.2	GPIO0_PDI[2]
[0]	P0.0	GPIO0_PDI[0]

19.2.9 DSP0_BSRR

Through DSP assembler access, software cannot directly access, corresponding DSP DATA Memory address: 0x3D

Reset value: 0x0

Table19-22 DSP0_BSRR register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P2.14	P2.7	P2.6	P2.5	P2.4	P2.3	P1.0	P0.15	P0.14	P0.13	P0.12	P0.11	P0.4	P0.3	P0.2	P0.0
WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	P2.14	GPIO2_BSRR[14]
[14]	P2.7	GPIO2_BSRR[7]
[13]	P2.6	GPIO2_BSRR[6]
[12]	P2.5	GPIO2_BSRR[5]
[11]	P2.4	GPIO2_BSRR[4]
[10]	P2.3	GPIO2_BSRR[3]
[9]	P1.0	GPIO1_BSRR[0]
[8]	P0.15	GPIO0_BSRR[15]
[7]	P0.14	GPIO0_BSRR[14]
[6]	P0.13	GPIO0_BSRR[13]
[5]	P0.12	GPIO0_BSRR[12]
[4]	P0.11	GPIO0_BSRR[11]
[3]	P0.4	GPIO0_BSRR[4]
[2]	P0.3	GPIO0_BSRR[3]
[1]	P0.2	GPIO0_BSRR[2]
[0]	P0.0	GPIO0_BSRR[0]

19.2.10 DSP0_BRR

Through DSP assembler access, software cannot directly access, corresponding DSP DATA Memory address: 0x3E

Reset value: 0x0

Table19-23 DSP0_BSRR register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P2.14	P2.7	P2.6	P2.5	P2.4	P2.3	P1.0	P0.15	P0.14	P0.13	P0.12	P0.11	P0.4	P0.3	P0.2	P0.0
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:16]		Not used
[15]	P2.14	GPIO2_BRR[14]
[14]	P2.7	GPIO2_BRR[7]
[13]	P2.6	GPIO2_BRR[6]
[12]	P2.5	GPIO2_BRR[5]
[11]	P2.4	GPIO2_BRR[4]
[10]	P2.3	GPIO2_BRR[3]
[9]	P1.0	GPIO1_BRR[0]
[8]	P0.15	GPIO0_BRR[15]
[7]	P0.14	GPIO0_BRR[14]
[6]	P0.13	GPIO0_BRR[13]
[5]	P0.12	GPIO0_BRR[12]
[4]	P0.11	GPIO0_BRR[11]
[3]	P0.4	GPIO0_BRR[4]
[2]	P0.3	GPIO0_BRR[3]
[1]	P0.2	GPIO0_BRR[2]
[0]	P0.0	GPIO0_BRR[0]

19.2.11 DSP0_CL

DSP0_CL is accessed through DSP assembler, and cannot be accessed directly by software. The corresponding DSP DATA Memory address is 0x3F.

Reset value: 0x0

Table19-24 DSP0_CL register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												CL_OUTPUT3	CL_OUTPUT2	CL_OUTPUT1	CL_OUTPUT0
												RO	RO	RO	RO
												0	0	0	0

Location	Bit name	Explain
[31:4]		Not used
[3]	CL_OUTPUT3	GPIO0_BRR[4]
[2]	CL_OUTPUT2	GPIO0_BRR[3]
[1]	CL_OUTPUT1	GPIO0_BRR[2]
[0]	CL_OUTPUT0	GPIO0_BRR[0]

19.2.12 DSP0_OP

Address: 0x4001_2FF8

Reset value: 0x0

Table19-25 DSP0_OP Register

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

Location	Bit name	Explain
[31:0]	OPERAND	DSP Operand Register

DSP0_op is the content of the DSP DATA memory at address 0x3E, the penultimate word content. Readable and writable to the CPU software and allows the DSP to read and write during operation. For the DSP, DSP0_OP is read-only, and the DSP writes the 0x3E address to the DSP0_BRR register.

19.2.13 DSP0_RES

Address: 0x4001_2FFC

Reset value: 0x0

Table19-26 DSP0_RES Register

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

Location	Bit name	Explain
[31:0]	RESULT	DSP Operation Result Register

DSP0_RES is the contents of the DSP DATA memory at Address 0x3F, a copy of the contents of the last word, which is read-only for CPU software and allows reading during DSP operation. For the DSP, DSP0_RES is only writable, and what the DSP reads from the 0x3F address is actually the contents of DSP0_CL.

19.3 DSP instruction set

19.3.1 Instruction Set Summary

Table 19-27 DSP instruction set

Operation	Description	Assembler		Cycles
Add		ADD	Rd1 Rs1 Rs2	1
Subtract		SUB	Rd1 Rs1 Rs2	1
And		AND	Rd1 Rs1 Rs2	1
Or		OR	Rd1 Rs1 Rs2	1
Shift	Arithmetic right shift	ASR	Rd1 Rs1 Rs2	1
	5bit Immediate	ASRI	Rd1 Rs1 #<Imm>	1
	Logical left shift	LSL	Rd1 Rs1 Rs2	1
	Logical right shift	LSR	Rd1 Rs1 Rs2	1
	5bit Immediate	LSLI	Rd1 Rs1 #<Imm>	1
Multiply and accumulation		MAC	Rs1 Rs2 Rs3	1
	5bit Immediate	MACI	Rs1 Rs2 #<Imm>	1
Divide		DIV	Rd1 Rs1 Rs2	10
Saturation		SAT	Rd1 Rs1 Rs2	1
	4bit Immediate	SATI	Rd1 #<Imm1> #<Imm2>	1
Cordic	SIN/COS	SIN_COS	Rd1 Rd2 Rs1	8
	Arctan/Module	ARCTAN	Rd1 Rs1 Rs2	8
Square root	Square root	SQRT	Rd1 Rs1	8
Memory access	Load word	LDRWI	Rd1 #<Imm>	1
	Load double half words	LDRDHI	Rd1 Rd2 #<Imm>	1
	Store word	STRWI	Rs1 #<Imm>	1
	Store double half words	STRDHI	Rs1 Rs2 #<Imm>	1
Branch	Unconditional Jump	JUMP	Rs1	2
	Immediate	JUMPI	#<Imm>	2
	Jump if less than or equal to	JLE	Rs1 Rs2 Rs3	2
	Jump if equal to	JEI	Rs1 Rs2 Rs3	2
	Immediate	JLEI	Rs1 Rs2 #<Imm>	2
Miscellaneous	Generate IRQ and Pause DSP	IRQ		1

The DSP uses 16-bit fixed-length encoding instructions, and because there are 8 general-purpose registers, 3 bits are required for register encoding. Most of the instructions are 3-operand instructions, including two source operand registers and one destination operand register. Some instructions contain an immediate value; Some instructions involve 4 operands. Take MAC operation as an example, Rd = Rs1 * Rs2 + Rs3. Since the instruction length is not enough to represent 4 registers, Rd is fixed to R7, which is not displayed in the instruction code. The remaining 4-operand instructions also have ARCTAN/DIV. See the detailed explanation of the instruction below for the specific operand assignment.

19.3.2 ADD

19.3.2.1 Coding

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	0	0	0	0	0	Rs2	Rd1	Rs1

19.3.2.2 Assembly Syntax

ADD Rd1 Rs1 Rs2

19.3.2.3 Pseudo-code

Rd1 = Rs1 + Rs2

As a result of that anti-overflow protection,

Rd 1 = 0x7FFF_FFFF after overflow,

Rd 1 = 0x8000 0000 after underflow

19.3.3 AND

19.3.3.1 Instruction Code

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	0	0	1	0	0	Rs2	Rd1	Rs1

19.3.3.2 Assembly Syntax

AND Rd1 Rs1 Rs2

19.3.3.3 Pseudo-code

Rd1 = Rs1&Rs2

19.3.4 OR

19.3.4.1 Instruction Code

15	14	13	12	11	10	9	8 7 6	5 4 3	2 1 0
0	0	0	0	1	1	0	Rs2	Rd1	Rs1

19.3.4.2 Assembly Syntax

```
OR        Rd1
```

19.3.4.3 Pseudo-code

Rd1 = Rs1|Rs2

19.3.5 SUB

19.3.5.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	Rs2			Rd1			Rs1		

19.3.5.2 Assembly Syntax

```
SUB       Rd1  Rs1  Rs2
```

19.3.5.3 Pseudo-code

Rd1 = Rs1 – Rs2

As a result of that anti-overflow protection,

Rd 1 = 0x7FFF_FFFF after overflow,

Rd 1 = 0x8000 0000 after underflow

19.3.6 ASR

19.3.6.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	Rs2			Rd1			Rs1		

19.3.6.2 Assembly Syntax

```
ASR       Rd1  Rs1  Rs2
```

19.3.6.3 Pseudo-code

Rd1 = Rs1 >> Rs2

Arithmetic right shift instruction only supports right shift of 0 ~ 31 bits.

19.3.7 ASRI

19.3.7.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	Imm					Rd1			Rs1		

The immediate value is a 5-bit unsigned number, indicating that the value range is 0 to 31.

19.3.7.2 Assembly Syntax

ASRI   Rd1 Rs1 Imm

19.3.7.3 Pseudo-code

Rd1 = Rs1 >> Imm

Arithmetic right shift instruction with immediate value only supports right shift of 0 ~ 31 bits.

19.3.8 LSL

19.3.8.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	Rs2			Rd1			Rs1		

19.3.8.2 Assembly Syntax

LSL Rd1 Rs1 Rs2

19.3.8.3 Pseudo-code

Rd1 = Rs1 << Rs2

Logic shift left only supports 0 ~ 31 bit shift left.

19.3.9 LSR

19.3.9.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	Rs2			Rd1			Rs1		

19.3.9.2 Assembly Syntax

Rd1 Rs1 Rs2

19.3.9.3 Pseudo-code

Rd1 = Rs1 >> Rs2

Logical right shift only supports 0-31 bit left shift.

19.3.10 LSLI

19.3.10.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	Imm					Rd1			Rs1		

The immediate value is a 5-bit unsigned number, indicating that the value range is 0 to 31.

19.3.10.2 Assembly Syntax

Rd1 Rs1 Imm

19.3.10.3 Pseudo-code

Rd1 = Rs1 << Imm

Logical left shift with immediate only supports 0 ~ 31 bit left shift.

19.3.11 MAC

19.3.11.1 Instruction Code

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

0	1	0	0	0	0	0	Rs3	Rs2	Rs1

19.3.11.2 Assembly Syntax

MAC Rs1 Rs2 Rs3

19.3.11.3 Pseudo-code

Rd7 = Rs1 × Rs2 + Rs3

As a result of that anti-overflow protection,

Rd7 = 0x7FFF_FFFF after overflow,

Rd 7 = 0x8000 0000 after underflow

Where Rs1 and Rs2 are 32-bit signed numbers, and Rs3 is a 32-bit signed number. When Rs3 is R0, MAC can be used as the multiplication instruction MUL.

19.3.12 MACI (reserved)

The multiply-add instruction with immediate is reserved in this version of DSP and is not implemented.

19.3.12.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	Imm					Rs2			Rs1		

The immediate value is a 5-bit signed number, indicating that the value range is -16 ~ 15.

19.3.12.2 Assembly Syntax

▍ Rs1 Rs2 Imm

19.3.12.3 Pseudo-code

Rd7 = Rs1 × Rs2 + Imm

As a result of that anti-overflow protection,

Rd7 = 0x7FFF_FFFF after overflow,

Rd 7 = 0x8000 0000 after underflow

19.3.13 DIV

19.3.13.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0		Rd2			Rd1			Rs1	

19.3.13.2 Assembly Syntax

Rd2 Rd1 Rs1

19.3.13.3 Pseudo-code

Rd1 = R7/Rs1, Rd2 = R7 % Rs1

It takes 12 cycles for the division instruction to finish submitting. In the process of division calculation, the DSP should not issue other multi-cycle instructions, that is, only one multi-cycle instruction can be in the long-pipeline at the same time. Other multi-cycle instructions include trigonometric function instructions and square root instructions. Multi-cycle instructions can run in the background, that is, the DSP can continue to execute other single-cycle instructions while the multi-cycle instructions are operating. However, only one multi-cycle instruction can run in the background at a time. The DSP can still use the destination operand of the multi-cycle instruction during the calculation of the multi-cycle instruction, but it should be noted that the destination operand register will be overwritten when the multi-cycle calculation is completed and the result is submitted.

Wherein R7, Rs1, Rd1 and Rd2 are 32-bit signed numbers. The dividend of the divide instruction is fixed at R7.

19.3.14 SAT

19.3.14.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0		Rs2			Rd1			Rs1	

19.3.14.2 Assembly Syntax

Rd1 Rs1 Rs2

19.3.14.3 Pseudo-code

If (Rd1<Rs1) Rd1=Rs1; else if (Rd1>Rs2) Rd1=Rs2

19.3.15 SATI (reserved)

The saturation instruction with an immediate value is reserved in this version of the DSP and is not implemented.

19.3.15.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	Imm2				Imm1				Rs1		

19.3.15.2 Assembly Syntax

SATI Rd1 Imm1 Imm2

19.3.15.3 Pseudo-code

If (Rd1<Imm1) Rd1=Imm1; else if (Rd1>Imm2) Rd1=Imm2

19.3.16 SIN_COS

19.3.16.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	Rd2			Rd1			Rs1		

19.3.16.2 Assembly Syntax

SIN_COS Rd1 Rd2 Rs1

19.3.16.3 Pseudo-code

Sin/cos instruction 20 cycles out and commits, and the DSP should not issue another multi-cycle instruction during its execution.

Rd1=cos (Rs1); Rd2=sin (Rs1)

19.3.17 ARCTAN

19.3.17.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	Rs2			Rd1			Rs1		

19.3.17.2 Assembly Syntax

ARCTAN Rd1 Rs1 Rs2

19.3.17.3 Pseudo-code

The ARCTAN instruction 20 cycles out and commits, and the DSP should not issue another multi-cycle instruction during its execution.

Rd1= arctan(Rs2/Rs1); R6 = sqrt(Rs1^2+Rs2^2)

19.3.18 SQRT

19.3.18.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	0	0	0	Rd1			Rs1		

19.3.18.2 Assembly Syntax

SQRT Rd1 Rs1

19.3.18.3 Pseudo-code

The 8-cycle open instruction is completed and committed, and the DSP should not issue another multi-cycle instruction during its execution.

Rd1 = sqrt(Rs1)

19.3.19 LDRWI

19.3.19.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	Imm						Rd1			0	0	0

19.3.19.2 Assembly Syntax

LDRWI Rd1 Imm

19.3.19.3 Pseudo-code

Rd1=word(SRAM[imm])

Because the load instructions are immediate instructions, the data address to be accessed can be generated in the decode stage, so the load operation can be completed in one cycle. For a similar load instruction, the CPU needs to access a register to calculate the address, so it takes 2 cycles.

The expression range of immediate IMM was 0 ~ 63. Because that DSP data mem is compose of 64 32-bit words.

19.3.20 LDRDHI

19.3.20.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	Imm						Rd1			Rd2		

19.3.20.2 Assembly Syntax

```
LDRDHI        Rd1  Rd2  Imm
```

19.3.20.3 Pseudo-code

```
LDRDHI   Rd1  Rd2
{Rd1,Rd2}= word(SRAM[Imm]),
```

The upper 16 bits of the 32 bit data retrieved from the data mem are sign extended to 32 bits and assigned to Rd1.

The lower 16 bits are sign extended to 32 bits and assigned to Rd2.

The expression range of immediate IMM was 0 ~ 63. Because that DSP data mem is compose of 64 32-bit words.

19.3.21 STRWI

19.3.21.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	Imm						0	0	0	Rs1		

19.3.21.2 Assembly Syntax

STRWI   Rs1 Imm

19.3.21.3 Pseudo-code

word{SRAM[imm]}=Rs1

The data for the Store instruction comes from a register, so even if the address to be written is generated in the decode stage, the Store operation cannot be completed immediately. The address needs to be latched for one cycle and then sent to the data memory interface with the written data. Therefore, if the load instruction is connected immediately after the Store instruction, there will be an access violation. The assembler should be designed to avoid such instruction sequences. If the STR instruction must be followed by an LDR instruction, an ADD R0 R0 R0 instruction can be inserted between the two as an instruction bubble.

The expression range of immediate IMM was 0 ~ 63. Because that DSP data mem is compose of 64 32-bit words.

19.3.22 STRDHI (reserved)

The store double halfword instruction with an immediate is reserved in this version of the DSP and is not implemented.

19.3.22.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	Imm						Rs1			Rs2		

19.3.22.2 Assembly Syntax

STRDHI   Rs1 Rs2 Imm

19.3.22.3 Pseudo-code

word{SRAM[imm]}={Rs1, Rs2}

The expression range of immediate IMM was 0 ~ 63. Because that DSP data mem is compose of 64 32-bit words.

19.3.23 JUMPI

19.3.23.1 Instruction Code

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

19.3.23.2 Assembly Syntax

JUMPI   Imm

19.3.23.3 Pseudo-code

PC = PC + 1 + IMM

19.3.24 JLE

19.3.24.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	Rs3			Rs2			Rs1		

19.3.24.2 Assembly Syntax

JLE   Rs1 Rs2 Rs3

19.3.24.3 Pseudo-code

PC = PC + 1 + Rs3, if (Rs1 <= Rs2)

19.3.25 JEI

19.3.25.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	Imm					Rs2			Rs1		

19.3.25.2 Assembly Syntax

JEI Rs1 Rs2 Imm

19.3.25.3 Pseudo-code

PC = PC + 1 + IMM, if (Rs1 == Rs2)

19.3.26 JLEI

19.3.26.1 Instruction Code

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	Imm					Rs2			Rs1		

19.3.26.2 Assembly Syntax

JLEI Rs1 Rs2 Imm

19.3.26.3 Pseudo-code

PC = PC + 1 + IMM, if (Rs1 <= Rs2)

19.3.27 IRQ

19.3.27.1 Instruction Code

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

19.3.27.2 Assembly Syntax

IRQ

19.3.27.3 Pseudo-code

An IRQ instruction generates an interrupt, waits for the CPU to process it, and suspends the DSP.

19.3.28 R (simulator only)

19.3.28.1 Instruction Code

None

19.3.28.2 Assembly Syntax

```
R5 0x5555        # Assign 0x5555 to R5
R1 20            # Assign 20 to R1
```

19.3.28.3 Pseudo-code

Debug instructions, used only in the DSP Emulator. It can be used to set R1 to R7 to any specified value at any position in the DSP program.

19.3.29 BREAK (simulator only)

19.3.29.1 Instruction Code

None

19.3.29.2 Assembly Syntax

```
BREAK
```

19.3.29.3 Pseudo-code

Debug instructions, used only in the DSP Emulator. It can be used to insert breakpoints anywhere in the DSP program and print the R1 to R7 register values. After the breakpoint is hit, press Enter to continue the program.

19.3.30 END (simulator only)

19.3.30.1 Instruction Code

None

19.3.30.2 Assembly Syntax

```
END
```

19.3.30.3 Pseudo-code

Similar to the breakpoint instruction, used only in the DSP Emulator. When the instruction is encountered in the instruction simulation, the operation of the simulator is interrupted and the core register value is printed directly.

19.4 Application Guide

19.4.1 Memory access addressing

DSP uses immediate addressing. Due to the limited address space of DATA MEM, the 6-bit immediate in STR and LDR instructions can directly represent all address offsets of DATA MEM.

Taking the following DATA MEM content as an example, the first row of data 0x01000100 corresponds to the address of 0x0, the second row of data 0x30005000 corresponds to the address of 0x1, and the third row of data 0x0003FFF8 corresponds to address of 0x2.

Use LDRDHI R1 R2 0x1 to assign R1, R2 R1 = 0x3000, R2 = 0x5000

Use STRWI R3 0x3 to write 32 bits of data from R3 to the 0x3 address and overwrite the 0xFFFFE000 data.

It should be noted that although DSP memory addressing increases by 4 Bytes per 1, for the CPU, it increases by 4 Bytes per 4 addresses. Therefore, when the CPU accesses the DSP DATA MEM, it should be calculated according to the_DATA_MEM_BASE of DSP0 + offset * 4. Taking the following DATA MEM content as an example, the CPU addressing address corresponding to the second row of data 0x30005000 is 0x 4001_4804, and the CPU addressing address corresponding to the third row of data 0x0003FFF8 is 0x4001_4808.

DATA MEM：

```
0x00100010
0x30005000
0x0003FFF8
0xFFFFE000
0x30004000
0x7FFFFFFF
0x7FFFFFFF
0xF0000003
0x00007FFF  # 8
0x00008000
0x80000000
0x00000000
```

19.4.2 Delayed commit of multi-cycle instructions

DSP's multi-cycle arithmetic instructions include division, square root, and trigonometric functions.

The divide instruction requires 10 bus cycles (96 MHz) to complete.

The root instruction requires eight bus cycles (96 MHz) to complete.

A trigonometric function instruction requires 20 bus cycles (96 MHz) to complete.

In order to take full advantage of the DSP performance, multi-cycle instructions are allowed to execute in the background, i.e., the DSP can still execute other instructions during the computation of multi-cycle instructions without blocking the pipeline. This requires some consideration of the instruction sequence at the time of programming, inserting other unrelated instructions between the issue of the multi-cycle instruction and the use of the result. The following instruction sequence, where ADD R0 R0 R0 acts like NOP and can be replaced with other instructions in practice.

```
SIN_COS R1 R2 R3

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

ADD R0 R0 R0

LDRWI R4 0x10

MAC R1 R4 R0
```

19.4.3 Software-simulated serial interface

The DSP0_PDI and the DSP0_BSRR, DSPO_BRR registers allow the DSP program to interact with the GPIOs. Unlike CPU programs, which frequently enter the interrupt handling program and cause poor real-time performance of software, DSP programs run in a relatively closed program environment and have strong time accuracy in program execution. DSP analog serial interface can be realized by using this characteristic of DSP program. Taking UART transmission as an example, the frame structure of UART is usually 1bit start bit + 1 byte data + 1bit stop bit. The DSP can set or clear the GPIO_PDO by operating the DSP0_BSRR and DSP0_BRR registers according to a certain time interval, send the serial data bit by bit, and notify the CPU/DMA that the sending is completed by an interrupt instruction after the sending is completed. If UART reception is performed, the DSP program keeps polling the DSP0_PDI register. If 0 is detected, it indicates that the UART start bit is detected, and the DSP0_PDI register is read regularly at a certain time interval to read in a complete frame of UART data.

Similarly, other serial interfaces such as SPI or I2C are also simulated by DSP software in the above way.

20 IWDG Independent Watchdog

20.1 Overview

The independent watchdog operates with a 32 kHz low-speed RC (LRC/LSI) clock, which is guaranteed to work even when the system master clock is stopped.

The independent watchdog contains a 21-bit down counter, which starts counting down from the configured value after startup. A separate watchdog may be configure to generate a system sleep wakeup source as a timed wakeup source. A full chip reset signal may also be generate in that event of a timeout. When the MCU enters the deep sleep state, the system clock including PLL/HRC is turned off, while the LRC clock is always present, so the independent watchdog can continue to work normally.

Generally, the watchdog's timed wake-up threshold is less than the timeout reset threshold, but greater than 0x8, that is, the independent watchdog counts down from the IWDG_RTH after reloading, and generates a wake-up signal when the count reaches the IWDG_WTH. Generally, if the IWDG is to be used for sleep timed wake-up, the IWDG is refreshed and reset before entering sleep, the IWDGcounts down from the IWDG_RTH, and an IWDG wake-up signal is generated after a period of time. If the IWDG sleep timed wake-up is not enabled, the chip can also be released from sleep when the IWDG generates a full chip reset, but all MCU register configurations are reset at the same time.

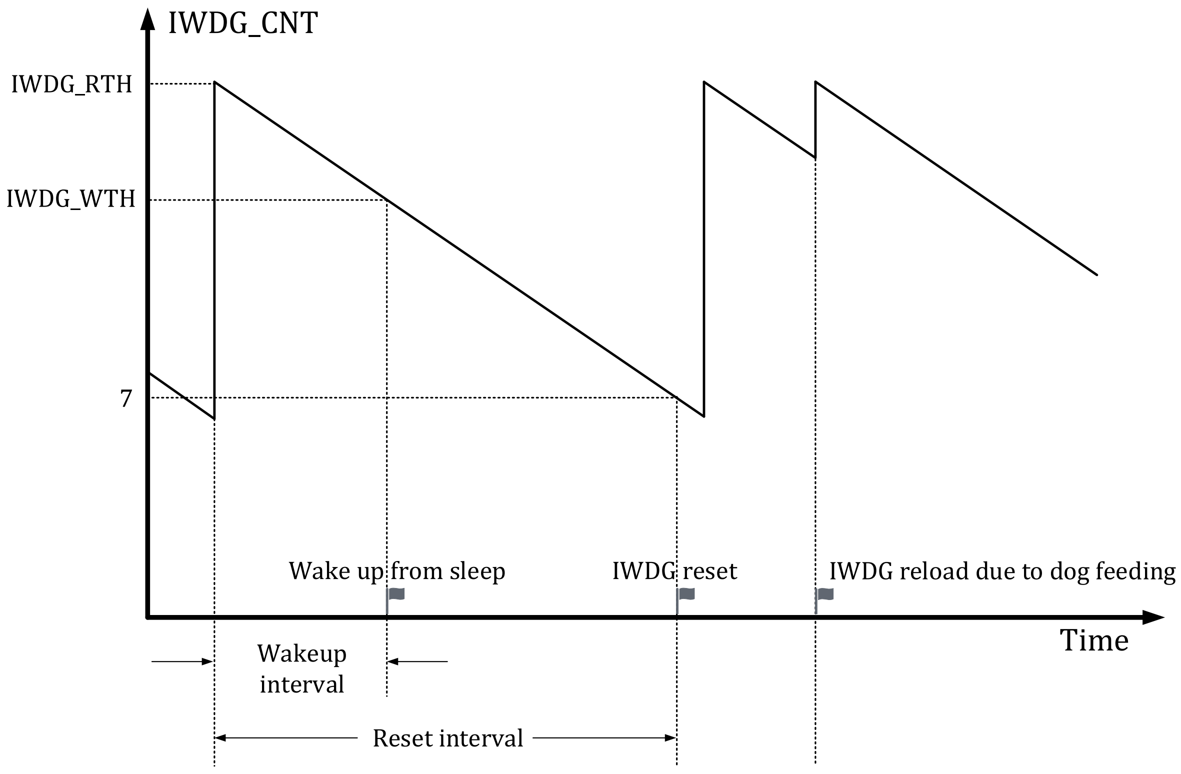


Figure20-1 Example of an IWDG countdown event occurring

The independent watchdog timeout reset time is calculated by the following formula

$Timeout_{IWDG}=t_{LRC}\times(IWDG\_RTH-7)$

The independent watchdog wake-up time is calculated by the following formula

$\text{Wakeup}_{\text{IWDG}}=t_{\text{LRC}}\times(\text{IWDG\_RTH}-\text{IWDG\_WTH})$

Where $\text{Timeout}_{\text{IWDG}}$is the independent watchdog timeout reset time, $\text{Wakeup}_{\text{IWDG}}$is the independent watchdog wakeup time, $t_{\text{LRC}}$is the LRC clock cycle, 1/32kHz = 31.25uS, and IWDG_RTH is the Independent Watchdog timeout reset threshold. The IWDG_WTH is the independent watchdog wake-up threshold.

IWDG_RTH = 0 × 001000 corresponds to a minimum 4096/32kHz ≈ 128ms independent watchdog reset interval.

IWDG_RTH = 0x1FF000 corresponds to a maximum independent watchdog reset interval of 511 × 4096/32kHz ≈ 64s.

It should be noted that due to the limited accuracy of the low-speed RC clock, there is a certain deviation from chip to chip. Usually, the deviation of the low-speed RC clock does not exceed ± 50% over the full temperature range.

About IWDG feed dog cross-clock synchronization issues:

Since IWDG counters work in the LRC clock domain, the software runs in the system master clock domain, which is usually a PLL clock. When the software is fed to the dog by writing to the IWDG RTH or writing to the IWDG CLR, the watchdog counter is reset from the time the software writes to the watchdog, requiring up to 3 LRC clock cycles.

Therefore, if the software needs to read the IWDG CNT after feeding the dog, it needs to delay at least 3 LRC clock cycles to read the IWDG CNT and find that the counter has been reset.

Software write modify IWDG RTH/IWDG WTH/IWDG CFG/IWDG PSW and other registers, write immediately take effect without waiting. Only the reset of the IWDG CNT takes effect after a delay.

20.2 Register

20.2.1 Address assignment

The IWDG base address is 0x40011700

Table20-1Independent watchdog register

Name	Offset	Explain
IWDG_PSW	0x00	Independent watchdog password register
IWDG_CFG	0x04	Independent watchdog configuration register
IWDG_CLR	0x08	Independent watchdog clear register
IWDG_WTH	0x0C	Independent Watchdog Counter Timed Wake-up Threshold Register
IWDG_RTH	0x10	Independent watchdog counter timeout reset threshold register
IWDG_CNT	0x14	Independent watchdog counter current count register

20.2.2 IWDG_PSW Independent Watchdog Password Register

Address: 0x4001_1700

Reset value: 0x0

Table20-2 IWDG_PSW Independent Watchdog Password Register

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

Locatio n	Bit name	Explain
[31:16]		Not used
[15:0]	PSW	Write 0xA6B4 to write to IWDG_CLR/IWDG_RTH, etc. A write to IWDG_CLR or IWDG_RTH clears the password, so the password needs to be rewritten each time before the Watchdog IWDG_CLR/IWDG_RTH register is written

20.2.3 IWDG_CFG Independent Watchdog Configuration Register

Address: 0x4001 1704

Reset value: 0x0

Table20-3 IWDG_CFG Independent Watchdog Configuration Register

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

Locatio n	Bit name	Explain
[31:5]		Not used
[4]	DWK_EN	Deep Sleep Timed Wakeup Enable, 0: Disabled, 1: Enabled
[3:1]		Not used
[0]	WDG_EN	Independent watchdog enabled, 0: disabled, 1: enabled. Enabled by default, set by writing a 1 and cleared by writing a 0 while writing 0x3C to [15:8]. When the watchdog is disabled, the reset signal is no longer generated, but the timer wake-up signal can still be counted and generated

Writes to the IWDG_CFG do not require the password to be written to the IWDG_PSW.

20.2.4 IWDG_CLR Independent Watchdog Clear Register

Address: 0x4001 1708

Reset value: 0x0

Table20-4 IWDG_CLR watchdog clear register

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

Location	Bit name	Explain
[31:16]	IWDG_CLR	Not used
[15:0]		Write byte 16'b0111_1001_1000_110B$_0$. The upper 15 bits are the password. When the password is correct, B [0] can be written. If 1 is written in B [0], the WDT counter is reset to TH, and the bit is automatically cleared after being written. Writing 0 is invalid.

Writing to the IWDG_CLR requires first writing the password to the IWDG_PSW.

20.2.5 IWDG_WTH Independent Watchdog Timed Wake-up Threshold

Address: 0x4001 170C

Reset value: 0x001F_F000

Table20-5 IWDG_WTH Independent Watchdog Timeout Reset Threshold

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_WTH				
											RW				
											0x1F				

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

Location	Bit name	Explain
[31:21]	IWDG_WTH	Not used
[20:12]		Watchdog timer wake-up threshold. Watchdog uses 32kHz LRC clock to start counting from IWDG_RTH and count down to IWDG_WTH to generate wake-up signal.
[11:0]		Is always 0

Writes to the IWDG_WTH do not require the password to be written to the IWDG_PSW.

20.2.6 IWDG_RTH Independent Watchdog Timeout Reset Threshold

Address: 0x4001 1710

Reset value: 0x001F_F000

Table20-6 IWDG_RTH Independent Watchdog Timeout Reset Threshold

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

	IWDG_RTH
	RW
	0x1F

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

Location	Bit name	Explain
[31:21]	IWDG_RTH	Not used
[20:12]		The watchdog timeout reset threshold is also a reload value. The watchdog starts counting from the IWDG_RTH using a 32 kHz LRC clock, and a count to 0x7 generates a reset. Writing 0x0 to this register is hardware-forced to 0x1000. Write the correct password to the IWDG_PSW before overwriting the IWDG_RTH register. Rewriting the IWDG_RTH also has the effect of resetting the watchdog counter, which starts counting from the new IWDG_RTH.
[11:0]		Is always 0

To prevent the IWDG_RTH from being written to 0, when the software writes a value of all 0s, the hardware forces a write to 0x 1000, and the lower 12 bits of the register are always 0. Writing to the IWDG_RTG requires writing a password to the IWDG_PSW.

20.2.7 IWDG_CNT Independent Watchdog Current Count Register

Address: 0x4001_1714

Reset value: 0x0000 0000

Table20-7 IWDG_CNT Independent Watchdog Current Count Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											IWDG_CNT				
											R				
											0x00				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_CNT															
R															
0x0000															

Location	Bit name	Explain
[31:21]	IWDG_CNT	Not used
[20:0]		Watchdog current count value, this value ⩽ IWDG_TH

21 PMU power management module

The chip can achieve the purpose of reducing power consumption by reducing the running clock frequency and shutting down part of the circuit clock.

21.1 Peripheral clock gating

The peripheral clock is obtained by gating or frequency dividing the system high-speed clock MCLK; When the peripheral is no longer needed, the corresponding peripheral clock can be turned off by configuring the SYS_CLK_FEN register. There is a clock gate for each peripheral's operating clock, as described in6.3.7。 **After the peripheral clock is powered on, it is turned off by default, and it needs to be turned on by software before using the corresponding peripheral module.**

21.2 Peripheral Clock Divide

An independent clock frequency division module is arranged outside the part, so that the module can work at an appropriate lower clock frequency.

The I2C uses SYS_CLK_DIV0 as the division factor, and the UART uses SYS_CLK_DIV2 as the division factor, as shown in6.3.3And6.3.6。 The baud rate of the UART has an additional divider inside the UART module.

21.3 Low power consumption mode

Table 21-1 Low Power Modes Summary

Pattern	Mode entry	Mode exit	PLL/HSI	LSI
Sleep Sleep	WFI/WFE	Arbitrary interrupt Debug operation External reset IWDG reset	PLL/HSI On, CPU Clock Off, NVIC/Peripheral Clock On	On
Deep sleep Deep Sleep	SLEEPDEEP + WFI/WFE	Timed wake-up of IWDG IO wake-up Debug operation External reset IWDG reset	PLL/HSI Off, CPU/Peripheral Clock Off	

21.4 Sleep Mode

In sleep mode, the CPU clock is turned off, but the NVIC module continues to work, and all peripheral modules and IOs work normally.

Hibernation mode has no effect on circuit power, and all register States and memory data remain normally powered during hibernation.

You can refer to the sleep routine to turn off the clock of each module in the digital part and turn off the analog ADC/OPA/CMP/DAC module before entering sleep.

21.4.1 Mode entry

Sleep mode can be entered by executing the WFI/WFE command. There are two different ways to enter sleep mode depending on how the SLEEPO NEXIT bit is set in the CPU core.

Sleep-now: If SLEEPONEXIT is 0, the CPU enters Sleep mode immediately after executing the WFI/WFE instruction

Sleep-on-exit: If SLEEPONEXIT is set to 1, the CPU enters Sleep mode after all interrupts have been processed

21.4.2 Mode exit

If WFI is used to enter hibernation, any interrupt can wake up the CPU.

If WFE is used to enter sleep, the peripheral interrupt flag or the IO wake-up event can be used as the wake-up event. When the peripheral interrupt flag is used as a wake-up event, the peripheral raises an interrupt signal to the CPU, but the corresponding interrupt is not enabled in the NVIC, and SEVONPEND needs to be set to 1. The peripheral interrupt flag and the NVIC interrupt pending bit need to be cleared by software after wake-up.

Using this approach for sleep wakeup results in the shortest wakeup time because there is no interrupt entry and exit involved.

The Debug operation wakes the chip from sleep mode.

21.5 Deep Sleep Mode

Deep sleep mode normally turns off all system high-speed clocks, including the PLL/HSI. Before entering deep sleep, set the SYS_AFE_REG5 = 0x0500 to turn off the PLL/HRC/BGP. The 32kHz RC clock LSI still works. At the same time, the LDO enters a low power mode and the BGP module is turned off.

Deep sleep mode can further reduce power consumption compared to sleep mode.

Deep sleep mode has no effect on circuit power, and all register States and memory data remain normally powered in deep sleep mode.

If a peripheral needs to use a high-speed clock to complete an in-progress operation, such as a flash erase write, deep sleep delays waiting for the operation to complete before entering.

21.5.1 Mode entry

Set the CPU core System Control Register SLEEPDEEP to 1, and then enter deep sleep by executing the WFI/WFE instruction.

21.5.2 Mode exit

An IO wake-up, CL output wake-up, or IWDG timed wake-up signal can wake up the chip from deep sleep.

An external reset or an IWDG reset resets the full chip and releases the deep sleep state.

Debug operations can wake up the chip from deep sleep mode.

21.6 Register

21.6.1 Address assignment

The PMU base address is 0x40011720

Table21-2Power Management Module Address Space

Name	Offset	Explain
AON_PWR_CFG	0x0	
AON_EVT_RCD	0x4	Event Log Register
AON_IO_WAKE_POL	0x8	IO Wake-up Polarity Register
AON_IO_WAKE_EN	0xC	IO Wake-up Enable Register

21.6.2 AON_PWR_CFG Power Management Configuration Register

Address: 0x4001 1720

Reset value: 0x0

Table21-3 AON_PWR_CFG Power Management Configuration Register

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

Location	Bit name	Explain
[31:2]		Not used
[1]	IOWK_FLT	IO wake-up signal filtering enabled, enabled by default
[0]		Not used

The software can choose to filter the IO wake-up signal or not. If the on-chip filtering is enabled, the IO signal is sent to the PMU after being filtered with a time constant of 120us. The use of on-chip IO signal filtering can replace board-level filtering in certain application scenarios, but it will introduce a wake-up delay of about 120us. If the on-chip IO wake-up signal filtering is not enabled, the application scenario needs to ensure that the IO wake-up input signal is stable and interference-free, and board-level filtering can be considered to avoid unnecessary power waste caused by false wake-up of the chip.

21.6.3 AON_EVT_RCD Event Logging Register

Address: 0x4001 1724

Reset value: 0x0

Table21-4 AON_EVT_RCD Event Logging Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		DEEPSLEEP	SLEEP			IWDG_WK	EVT_WK					IWDG_RST	KEY_RST		POR_RST
		RO	RO			RO	RO					RO	RO		RO
		0	0			0	0					0	0		1

Location	Bit name	Explain
[31:14]		Not used
[13]	DEEPSLEEP	Deep sleep record, high indicates occurred
[12]	SLEEP	Sleep record, high indicates that it occurred
[11:10]		Not used
[9]	IWDG_WK	IWDG timed wake-up record, high indicates that Deep Sleep sleep is timed wake-up by IWDG
[8]	EVT_WK	Record of IO wake-up or CLU wake-up. High indicates that Deep Sleep is woken up by the corresponding enable signal in the AON_IO_WAKE_EN.
[7:4]		Not used
[3]	IWDG_RST	Record of occurrence of independent watchdog reset. High indicates that it occurred.
[2]	KEY_RST_RCD	Record the occurrence of key reset. High indicates that it has occurred.
[1]		Not used
[0]	POR_RST_RCD	POR reset occurrence record, high indicates that it has occurred

Clear all event logs by writing 0xCA40 to the AON_EVT_RCD register. Because the EVT_RCD works in the LSI clock domain, it takes 32 us from the time the software writes the password to the time it clears.

21.6.4 AON_IO_WAKE_POL IO wake-up polarity register

Address: 0x4001_1728

Reset value: 0x0

Table21-5 AON_IO_WAKE_POL IO wake-up source polarity configuration register

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

Location	Bit name	Explain
[31:1]		Not used
[0]	WK_POL	IO external wake-up trigger level selection. 1: high level; 0: Low level

Use the same polarity selection for all IO wake-up signals.

21.6.5 AON_IO_WAKE_EN IO wake-up enable register

Address: 0x4001_172C

Reset value: 0x0

Table21-6 AON_IO_WAKE_EN IO wakeup source enable register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				CLUOUT3_EN	CLUOUT2_EN	CLUOUT1_EN	CLUOUT0_EN	P2_15_EN	P2_7_EN	P2_4_EN	P0_14_EN	P0_11_EN	P0_6_EN	P0_2_EN	P0_0_EN
				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
				0	0	0	0	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:12]		Not used
[11]	CLUOUT3_EN	CLUOUT3 as wakeup enabled
[10]	CLUOUT2_EN	CLUOUT2 as wakeup enabled
[9]	CLUOUT1_EN	CLUOUT1 as wakeup enabled
[8]	CLUOUT0_EN	CLUOUT0 as wakeup enabled
[7]	P2_15_EN	P2 [15] external wake-up enable
[6]	P2_7_EN	P2 [7] external wake-up enabled
[5]	P2_4_EN	P2 [4] external wake-up enabled
[4]	P0_14_EN	P0 [14] external wake-up enabled
[3]	P0_11_EN	P0 [11] external wake-up enabled
[2]	P0_6_EN	P0 [6] external wake-up enabled
[1]	P0_2_EN	P0 [2] external wake-up enabled
[0]	P0_0_EN	P0 [0] external wake-up enabled

22 SIF module

22.1 Brief description

The SIF module is suitable for single-wire, one-way communication applications, such as on-board instrument communication. This section describes the implementation and use of SIF.

22.2 Main characteristics

- Single line communication
- One-way communication, output only
- Support synchronization signal, end signal on/off, configurable
- Support synchronous signal and end signal with configurable length
- Support synchronization signal and end signal level to be configured
- Support MSB/LSB configurable
- The data duty cycle can be configured in 2:1 and 3:1 modes.
- DMA transfers are supported

22.3 Functional description

22.3.1 Functional block diagram

This interface adopts synchronous serial design to realize SIF transmission between MCU and external equipment. Support polling and interrupt mode to obtain transmission status information. The main function modules of this interface are shown in the figure below:

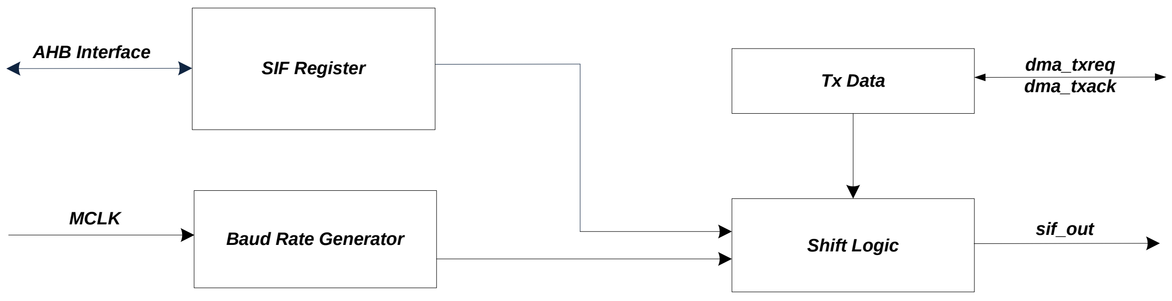


Figure22-1 Structure block diagram of SIF module

SIF Register Module. Registers associated with the SIF module.

Baud Rate Generator module, Tosc time base generation circuit.

Tx Data module, transmit data buffer.

Shift Logic module, data sending module.

22.3.2 Functional description

22.3.2.1 SIF format description

The SIF communication protocol is divided into several parts: synchronization (SYNC), data transmission, Done, and default level.

Sync (SYNC): The waveform at the start of a frame transmission. Some applications require a sync marker, some do not.

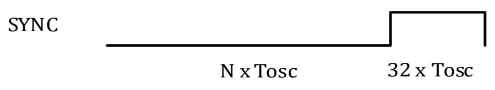


Figure22-2 SIF Module Sync (SYNC) Waveform

Output transmission: transmission 0 and transmission 1, two States.

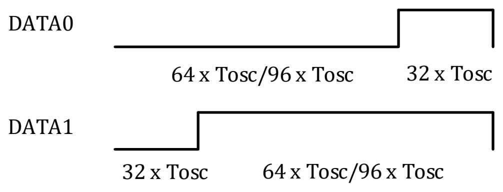


Figure22-3 SIF module data (DATA) waveform

Done: The end waveform of a frame transmission. Some applications require an end marker, and some do not. The end signal does not have a rollover waveform and outputs a high/low level as required by the application.

22.3.2.2 To SC time base

Tosc time base module to generate basic time count unit. The Tosc time base is clocked from the system clock division (up to 3m after division), with a minimum Tosc time base of 333 ns (decimated) and a maximum time base of approximately 1.364 ms (333 ns * 4096). This is achieved by configuring the SIF. TOSCL/H register.

22.3.2.3 Synchronize the time configuration

Configuration of the synchronization (SYNC) signal time, related to the SIF.STH1 register. The default level of the synchronization (SYNC) signal is configured in the SIF. CFG register.

Synchronous signal, if the default is low level, the minimum low level time is 1023 * Tosc, and the high level is fixed to 32 * Tosc.

Synchronous signal, if the default is high level, the maximum time of high level is 1023 * Tosc, and the low level is fixed to 32 * Tosc.

22.3.2.4 End time configuration

Configuration of the Done signal time, related to the SIF.DTH1 register. The default level of the Done signal is configured in the SIF. CFG register. End signal, the default level can be configured, the step is 1ms, from 1ms to 16 ms optional. The end time is the interval between the current output transmission and the next data transmission.

22.3.2.5 DMA transfer

The SIF module supports DMA transfers. The data received by the SIF module is one byte wide (8-bit), so the DMA moves one byte a round to the SIF module. The number of transmission rounds is controlled by the DMA module. If a SIF communication packet consists of N bytes, the DMA round number register can be configured with N.

22.4 Register

22.4.1 Address assignment

The base address of the SIF module register is 0x 4001_1500. The module register is listed below.

Table22-1 SIF module control register list

Name	Offset	Explain
SIF0_CFG	0x00	SIF configuration register
SIF0_TOSC	0x04	SIF TOSC Time Base Register
SIF0_STH1	0x08	SIF Sync Time register
SIF0_DTH1	0x0C	SIF end signal time register
SIF0_IRQ	0x10	SIF Interrupt Register
SIF0_WDATA	0x14	SIF transmit data register

22.4.2 SIF0_CFG configuration register

Address: 0x4001_1500

Reset value: 0x0

Table22-2Address register SIF 0_CFG

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

	DONE	SYNC	IDLE	DONE_VLD	SYNC_VLD	RATIO	MSB	EN
	RW	RW	RW	RW	RW	RW	RW	RW
	0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:8]		Not used
[7]	DONE	SIF module data transmission, complete signal default level 1: default high level 0: default low level
[6]	SYNC	SIF module data transmission, sync signal default level 1: default high level 0: default low level
[5]	SYNC_PULSE	SIF module data transmission, whether the synchronous signal generates pulse signal 1: Synchronous signal has pulse signal 0: no pulse signal for synchronous signal
[4]	DONE_VLD	SIF module data transmission, end signal control bit 1: There is an end signal 0: No end signal
[3]	SYNC_VLD	SIF module data transmission, sync signal control bit 1: There is a synchronization signal 0: no synchronization signal
[2]	RATIO	SIF module data duty cycle control bit 1： 3:1 0： 2:1
[1]	MSB	SIF module data header control bit 1： MSB 0： LSB
[0]	EN	SIF block enable bit 1: SIF module enabled 0: SIF module off

22.4.3 SIF0_TOSC time base register

Address: 0x4001_1504

Reset value: 0x0

Table22-3 SIF 0_TOSCL SIF Time Base Low Byte Register

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

	0

Location	Bit name	Explain
[31:12]		Not used
[11:0]	OSC_TH	SIF Module Time Base Settings T_{osc}= (OSC_TH+1)×(system clock period × 32)

22.4.4 SIF0_TSTH1 synchronization time register

Address: 0x4001 1508

Reset value: 0x0

Table22-4 Synchronization time register SIF 0_TSTH1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						TSTH1									
						RW									
						0									

Location	Bit name	Explain
[31:10]		Not used
[9:0]	T_{STH1}	SIF Module Sync Signal Period When using sync pulse, this period is the time before sync pulse. When using sync without pulse, this period is the whole time of sync. Time = (T_{STH1}+1)×Tosc

22.4.5 SIF0_TDTH1 end time register

Address: 0x4001 150C

Reset value: 0x0

Table22-5End time register SIF0_TDTH1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												TDTH1			
												RW			
												0			

Location	Bit name	Explain
[31:4]		Not used
[3:0]	T_{DTH1}	SIF Module End Signal Cycle Time = $(T_{DTH1}+1)\times$Tosc

22.4.6 SIF0_IRQ Interrupt Register

Address: 0x4001 1510

Reset value: 0x0

Table22-6Interrupt register SIF0_IRQ

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

Location	Bit name	Explain
[31:8]		Not used
[7:5]		Not used
[4]	IRQ_IF	Write 1 to clear the interrupt flag bit of SIF module. 1: SIF interrupt flag bit is available 0: No SIF interrupt flag bit
[3:2]		Not used
[1]	DMA_EN	SIF module DMA enable control bit 1: Enable 0: Off
[0]	IRQ_EN	SIF block interrupt enable control bit 1: Enable interrupt 0: turn off interrupt

22.4.7 SIF0_WDATA data register

Address: 0x4001 1514

Reset value: 0x0

Table22-7Data register SIF0_WDATA

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

	0

Location	Bit name	Explain
[31:8]		Not used
[7:0]	WDATA	SIF Module Data Register

23 CL0 configurable logic

23.1 Overview

Configurable Logic Unit (CLU for Configurable Logic Unit) provides user-configurable digital logic operations that operate without CPU intervention. The Configurable Logic (CL) module consists of four independent Configurable Logic Units (CLUs) that support user-programmable asynchronous and synchronous logic operations. Because of its flexible logic function, CL can be applied to many digital functions, such as replacing the glue logic of the system, generating special waveforms, or synchronizing the triggering of system events.

23.2 Main characteristics

The main features of CL0 are as follows:

- Contains four CLUs with IO pins and internal logic connections
- Each CLU supports 256 different combinational logic functions (AND, OR, XOR, MUX, etc.) and contains a clock-controlled D flip-flop for synchronous operation
- Each CLU may be used for synchronous or asynchronous logic, respectively
- CLUs can be cascaded to implement more complex logic functions.
- Can operate with SPI, I2C, or TIMER and MCPWM
- Can be used to synchronize or trigger multiple on-chip resources (ADC, UTIMTER, etc.)
- Asynchronous logic outputs can be used for wake-up operation

23.3 Functional description

23.3.1 Functional block diagram

As shown in the figure23-1As shown, the configurable logic CL0 mainly includes four independent configurable logic units CLU. Each CLU can be independently configured as a user-defined asynchronous or synchronous digital logic function without CPU intervention. A number of internal and external signals are available as inputs to each CLU, and the outputs of the CLUs can also be connected to IO outputs or directly to selected peripheral inputs.

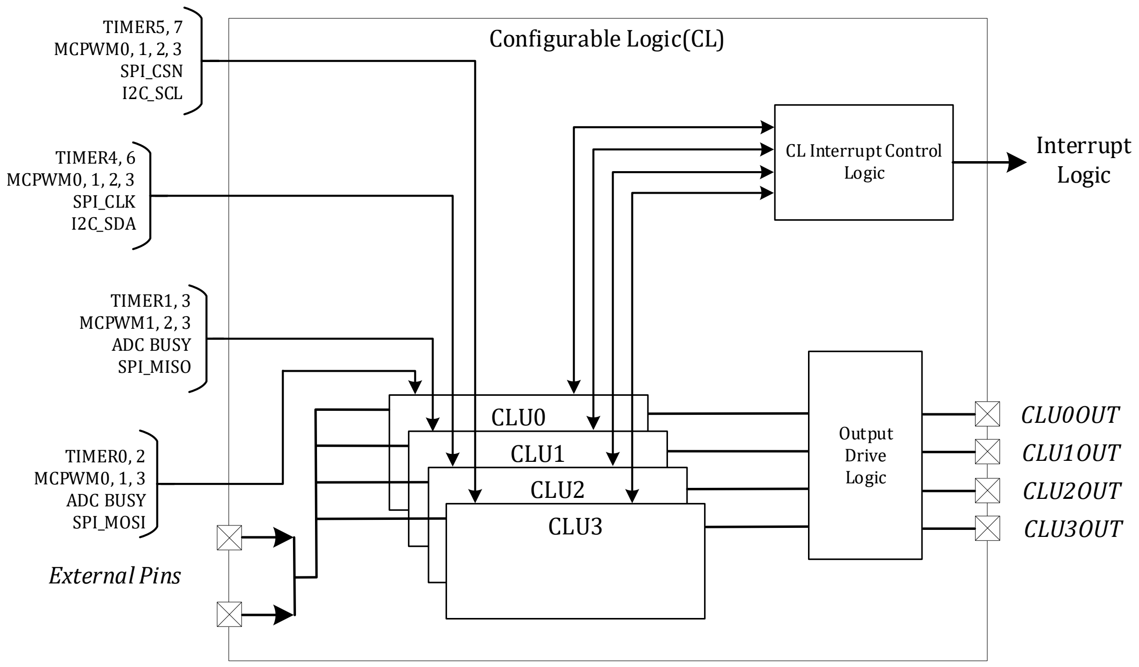


Figure23-2 CL Module Top Level Functional Block Diagram

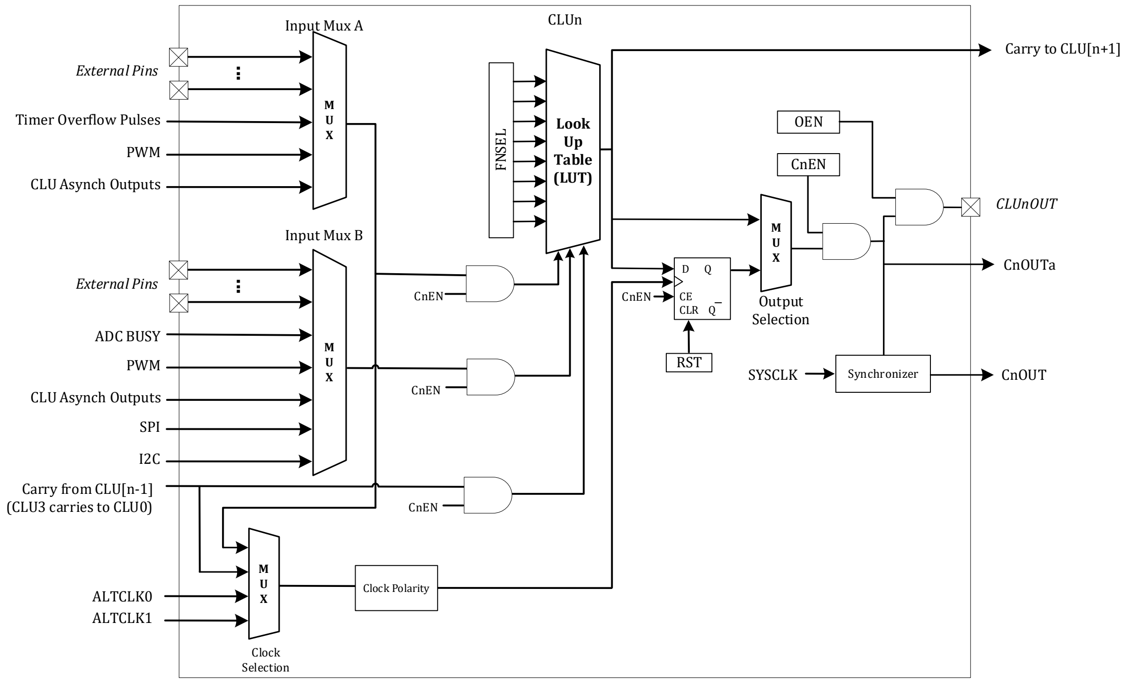


Figure23-3 Functional block diagram of independent CLU module

From the above and the functional block diagram of CL module, it can be seen that the function of CL is to input a number of internal and external signals to each CLU without CPU intervention, obtain the desired logic value result through the "LUT (lookup table)" method, and output it to the IO port or directly use it for the input of peripheral equipment.

23.3.2 Functional description

23.3.2.1 Configure the process

Function configuration, input multiplexing selection and output function configuration should be performed before enabling the independent CLU. For CLU initialization, refer to the following steps:

1. Selects the A and B inputs to the LUT in C0_MX0
2. Configure CL0_FN0 to select LUT function
3. Configure CL0_CF0 configuration register for CLU
4. If the D trigger output of the CLU is selected (OUTPSEL = 1), RST = 1 is configured in advance to reset the D triggered output to 0
5. Configure CL0_IE0 to enable the required interrupt signal. Rising and falling edge interrupts are enabled via the CnFIE and CnRIE bits, respectively
6. Configure CnEN of CL0_EN0 to enable the corresponding CLUn to enable multiple CLUs at the same time
7. If output to the IO port is required, enable the corresponding CnOEN bit of CL0_CF0

23.3.2.2 Input multiplexer

Each CLU has two main logic inputs (A and B) and a carry input (C). The A and B inputs are selected using CnMXA and CnMXB in Register CL0_MX0. One of each of the internal and external signals can be selected as an input. When the output of another CLU is selected as the input, the asynchronous output of the CLU is used, and the cascade can realize more complex logic functions.

Notes: When Timer overflow is used as an input, one clock overflow event is generated as a high level for one system clock cycle, and the rest of the time is a low level.

The carry input C is the asynchronous output of the preceding CLU. For example, the carry input C of CLU1 is the asynchronous output of CLU0, and the carry input C of CLU0 is the asynchronous output of CLU3.

Table23-1 CLUnA input selection

CLUnMX.MXA	CLU0A	CLU1A	CLU2A	CLU3A
0000	CLU0OUT	CLU0OUT	CLU0OUT	CLU0OUT
0001	CLU1OUT	CLU1OUT	CLU1OUT	CLU1OUT
0010	CLU2OUT	CLU2OUT	CLU2OUT	CLU2OUT
0011	CLU3OUT	CLU3OUT	CLU3OUT	CLU3OUT
0100	Timer0 ch0 ovfl	Timer0 ch1 ovfl	Timer2 ch0 ovfl	Timer2 ch1 ovfl
0101	Timer1 ch0 ovfl	Timer1 ch1 ovfl	Timer3 ch0 ovfl	Timer3 ch1 ovfl
0110	PWM0	PWM2	PWM0	PWM1
0111	PWM1	PWM3	PWM2	PWM3
1000	P0.0	P0.0	P2.8	P2.8
1001	P0.3	P0.2	P1.13	P3.9
1010	P0.5	P0.5	P1.15	P1.15
1011	P0.7	P0.6	P1.10	P2.0
1100	P2.12	P2.12	P2.1	P2.1
1101	P0.12	P0.11	P2.4	P2.3
1110	P0.14	P0.14	P2.6	P2.6
1111	P1.0	P0.15	P2.14	P2.13

Table23-2 CLUnB input selection

CLUnMX.MXB	CLU0B	CLU1B	CLU2B	CLU3B

0000	CLU0OUT	CLU0OUT	CLU0OUT	CLU0OUT
0001	CLU1OUT	CLU1OUT	CLU1OUT	CLU1OUT
0010	CLU2OUT	CLU2OUT	CLU2OUT	CLU2OUT
0011	CLU3OUT	CLU3OUT	CLU3OUT	CLU3OUT
0100	ADBUSY0	ADBUSY0	PWM1	PWM0
0101	ADBUSY1	ADBUSY1	PWM3	PWM2
0110	PWM3	PWM1	I2CSDA	I2CSCL
0111	SPIMOSI	SPIMISO	SPICLK	SPICSN
1000	P0.2	P0.3	P3.9	P1.13
1001	P0.4	P0.4	P1.14	P1.14
1010	P0.6	P0.7	P2.0	P1.10
1011	P2.11	P2.11	P1.11	P1.11
1100	P0.11	P0.12	P2.3	P2.4
1101	P0.13	P0.13	P2.5	P2.5
1110	P0.15	P1.0	P2.13	P2.14
1111	P2.7	P2.7	P2.15	P2.15

Note:

1. CLU0OUT, CLU1OUT, CLU2OUT and CLU3OUT respectively represent 4 output ports of CLU;

2. ADBUSY0 represents the ADC start signal, indicating that the ADC is sampling;

3. PWM0, PWM1, PWM2 and PWM3 respectively represent the TADC event trigger signals of the four PWM modules (trigger ADC sampling);

4. SPIMOSI, SPIMISO, SPICLK and SPICSN respectively represent the MOSI, MISO, CLK and CS ports of the SPI module;

5. I2CSDA and I2CSCL respectively represent the data signal SDA and the clock signal SCL of the I2C module;

6、 TIMER0 ch0 ovfl、TIMER0 ch1 ovfl、TIMER1 ch0 ovfl、TIMER1 ch1 ovfl、TIMER2 ch0 ovfl、TIMER2 ch1 ovfl、TIMER3 ch0 ovfl、TIMER3 CH1 ovfl represents the ADC sampling event triggered by utimer (utimer comparison event or capture event of utimer)

23.3.2.3 Output configuration

Each CLU has synchronous and asynchronous outputs. The synchronous output can be output at any time through the register CLOUT0. The CLU output is controlled by the OUTSEL of CLUNCF whether it is output directly from the LUT or from the DFF. When the CLU is closed (CLEN0 in CnEN is equal to 0), all outputs are held at 0.

The DFF clock is selected through CLKSEL of CLUnCF, and there are four sources. The DFF clock can be inverted by bit CLKINV. Each CLU has the following four DFF clock options:

1. CARRY_IN: the carry output C of the preceding CLU, and the first CLU uses the carry of the last CLU;
2. MXA_INPUT: A-input of CLU, selected by register MXA;
3. Altclk0: Timer3 overflows for clu0, Timer0 overflows for clu1, Timer1 overflows for clu2, and Timer2 overruns for clu3.
4. ALTCLK1: Timer1 of CLU0 overflows, Timer2 of CLU1 overflows, Timer3 of CLU2 overflows and Timer0 of CLU3 overflows.

Table 23-3 Timing of the CLU clock/external signal

Parameter	Symbol	Test conditions	Min	Typ	Max	Unit
Transmission delay	t_{DLY}	Single CLU, connected to external pin	-	-	50	ns
		Single CLU, internal connection	-	4	6	ns
Clock frequency	f_{CLK}	Cascade of 1, 2, or 3 CLUs	-	-	96	MHz
		4 CLU cascades	-	-	48	MHz

23.3.2.4 LUT configuration

The logic function of each CLU is determined by the LUT configuration, which can be selected by configuring CnFNSEL of CL0_FN0. Each bit of CnFNSEL is mapped to a combination of 8 multiplexer inputs, that is, the output of the LUT is selected by the combination of the A, B, and C inputs.

Table23-4 LUT truth table

A Input	B Input	C Input	LUT output
0	0	0	FNSEL.0
0	0	1	FNSEL.1
0	1	0	FNSEL.2
0	1	1	FNSEL.3
1	0	0	FNSEL.4
1	0	1	FNSEL.5
1	1	0	FNSEL.6
1	1	1	FNSEL.7

With reference to the truth table shown in the table above, all logic functions of the three inputs can be implemented using the LUT. For example, to implement and logic (A AND B), the output of the truth table is 1 only when both a and B are 1, that is, only FNSEL.7 and FNSEL.6 are 1, so FNSEL should be written to 11000000 B, that is, 0xC0.

23.4 Register

23.4.1 Address assignment

The base address of CL0 in the chip is 0x40011800, and the register list is as follows:

Table23-5 CL0 module register address assignment

Name	Offset	Description
CL0_EN0	0x00	CL0 enable register
CL0_IE0	0x04	CL0 Interrupt Enable Register
CL0_IF0	0x08	CL0 Interrupt Flag register
CL0_OUT0	0x0C	CL0 Output Register
CL0_MX0	0x10	CL0 Input Mux Register
CL0_FN0	0x14	CL0 function select register
CL0_CF0	0x18	CL0 configuration register

23.4.2 CL0_EN0 CL0 enable register

Address: 0x400 11800.

Reset value: 0x0

Table23-6 CL0_EN0 CL0 enable register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												C3EN	C2EN	C1EN	C0EN
												RW	RW	RW	RW
												0	0	0	0

Location	Bit name	Explain
[31:4]		Not used
[3]	C3EN	CLU3 enabled. Off by default. 0: Off 1: Enable
[2]	C2EN	CLU2 enabled. Off by default. 0: Off 1: Enable
[1]	C1EN	CLU1 enabled. Off by default. 0: Off 1: Enable
[0]	C0EN	CLU0 enabled. Off by default. 0: Off 1: Enable

23.4.3 CL0_IE0 CL0 Interrupt Enable Register

Address: 0x40011804

Reset value: 0x0

Table23-7 CL0_IE0 CL0 Interrupt Enable Register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								C3RIE	C3FIE	C2RIE	C2FIE	C1RIE	C1FIE	C0RIE	C0FIE
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:8]		Not used
[7]	C3RIE	CLU3 rising edge triggered interrupt enable. Off by default. 0: Off 1: Enable
[6]	C3FIE	CLU3 falling edge triggered interrupt enable. Off by default. 0: Off

		1: Enable
[5]	C2RIE	CLU2 rising edge trigger interrupt enable. Off by default. 0: Off 1: Enable
[4]	C2FIE	CLU2 falling edge trigger interrupt enable. Off by default. 0: Off 1: Enable
[3]	C1RIE	The CLU1 rising edge triggers an interrupt enable. Off by default. 0: Off 1: Enable
[2]	C1FIE	CLU1 falling edge trigger interrupt enable. Off by default. 0: Off 1: Enable
[1]	C0RIE	CLU0 rising edge trigger interrupt enable. Off by default. 0: Off 1: Enable
[0]	C0FIE	CLU0 falling edge trigger interrupt enable. Off by default. 0: Off 1: Enable

23.4.4 CL0_IF0 CL0 Interrupt Flag register

Address: 0x40011808

Reset value: 0x0

Table23-8 CL0_IF0 CL0 Interrupt Flag register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								C3RIF	C3FIF	C2RIF	C2FIF	C1RIF	C1FIF	C0RIF	C0FIF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[31:8]		Not used
[7]	C3RIF	CLU3 rising edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[6]	C3FIF	CLU3 falling edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[5]	C2RIF	CLU2 rising edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[4]	C2FIF	CLU2 falling edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[3]	C1RIF	CLU1 rising edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[2]	C1FIF	CLU1 falling edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[1]	C0RIF	CLU0 rising edge interrupt flag bit. Interrupt flag active high, clear by writing 1.
[0]	C0FIF	CLU0 falling edge interrupt flag bit. Interrupt flag active high, clear by writing 1.

23.4.5 CL0_OUT0 CL0 Output Register

Address: 0x4001180C

Reset value: 0x0

Table23-9 CL0_OUT0 CL0 output register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												C3OUT	C2OUT	C1OUT	C0OUT
												R	R	R	R
												0	0	0	0

Location	Bit name	Explain
[31:4]		Not used
[3]	C3OUT	CLU3 output status, synchronized with the system clock.
[2]	C2OUT	CLU2 output status, synchronized with the system clock.
[1]	C1OUT	CLU1 output status, synchronized with the system clock.
[0]	C0OUT	CLU0 output status, synchronized with the system clock.

23.4.6 CL0_MX0 CL0 Input Mux Register

Address: 0x40011 810

Reset value: 0x0

Table23-10 CL0_MX0 CL0 Input Mux Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C3MXA				C3MXB				C2MXA				C2MXB			
RW				RW				RW				RW			
0x0				0x0				0x0				0x0			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1MXA				C1MXB				C0MXA				C0MXB			
RW				RW				RW				RW			
0x0				0x0				0x0				0x0			

Location	Bit name	Explain
[31:28]	C3MXA	CLU3 A input mux select
[27:24]	C3MXB	CLU3 B Input Mux Select
[23:20]	C2MXA	CLU2 A input mux select
[19:16]	C2MXB	CLU2 B Input Mux Select
[15:12]	C1MXA	CLU1 A input mux select
[11:8]	C1MXB	CLU1 B Input Mux Select
[7:4]	C0MXA	CLU0 A input mux select

[3:0]	C0MXB	CLU0 B Input Mux Select

23.4.7 CL0_FN0 CL0 function select register

Address: 0x40011 814

Reset value: 0x0

Table23-11 CL0_FN0 CL0 function select register

31-24	23-16
C3FN	C2FN
RW	RW
0x00	0x00

15-8	7-0
C1FN	C0FN
RW	RW
0x00	0x00

Location	Bit name	Explain
[31:24]	C3FN	CLU3 lookup table function selection.
[23:16]	C2FN	CLU2 lookup table function selection.
[15:8]	C1FN	CLU1 lookup table function selection.
[7:0]	C0FN	CLU0 lookup table function selection.

23.4.8 CL0_CF0 CL0 configuration register

Address: 0x40011 818

Reset value: 0x0

Table23-12 CL0_CF0 CL0 configuration register

31	30	29-28	27	26	25-24	23	22	21-20	19	18	17-16
C3OUTSEL	C3OEN		C3RST	C3CLKINV	C3CLKSEL	C2OUTSEL	C2OEN		C2RST	C2CLKINV	C2CLKSEL
RW	RW		RW	RW	RW	RW	RW		RW	RW	RW
0	0		0	0	0x0	0	0		0	0	0x0

15	14	13-12	11	10	9-8	7	6	5-4	3	2	1-0
C1OUTSEL	C1OEN		C1RST	C1CLKINV	C1CLKSEL	C0OUTSEL	C0OEN		C0RST	C0CLKINV	C0CLKSEL
RW	RW		RW	RW	RW	RW	RW		RW	RW	RW
0	0		0	0	0x0	0	0		0	0	0x0

Location	Bit name	Explain

[31]	C3OUTSEL	CLU3 output selection 0: D trigger output 1: LUT output
[30]	C3OEN	CLU3 output enable. Off by default. 0: Off 1: Enable
[29:28]		Not used
[27]	C3RST	CLU3 D Trigger Reset. This bit is reset by writing a 1. The reading is always 0.
[26]	C3CLKINV	CLU3 D-trigger clock level is inverted. 0: Off 1: Enable
[25:24]	C3CLKSEL	CLU3 D Trigger Clock Select. 0x0: carry input 0x1: MXA input 0x2: ALTCLK0 0x3: ALTCLK1
[23]	C2OUTSEL	CLU2 output selection 0: D trigger output 1: LUT output
[22]	C2OEN	CLU2 output enabled. Off by default. 0: Off 1: Enable
[21:20]		Not used
[19]	C2RST	CLU2 D trigger reset. This bit is reset by writing a 1. The reading is always 0.
[18]	C2CLKINV	The CLU2 D trigger clock level is inverted. 0: Off 1: Enable
[17:16]	C2CLKSEL	CLU2 D trigger clock select. 0x0: carry input 0x1: MXA input 0x2: ALTCLK0 0x3: ALTCLK1
[15]	C1OUTSEL	CLU1 output selection 0: D trigger output 1: LUT output
[14]	C1OEN	CLU1 output enable. Off by default. 0: Off 1: Enable
[13:12]		Not used
[11]	C1RST	CLU1 D trigger reset. This bit is reset by writing a 1. The reading is always 0.
[10]	C1CLKINV	CLU1 D trigger clock level is inverted. 0: Off 1: Enable
[9:8]	C1CLKSEL	CLU1 D trigger clock select. 0x0: carry input 0x1: MXA input 0x2: ALTCLK0 0x3: ALTCLK1
[7]	C0OUTSEL	CLU0 output selection 0: D trigger output 1: LUT output
[6]	C0OEN	CLU0 output enabled. Off by default.

		0: Off 1: Enable
[5:4]		Not used
[3]	C0RST	CLU0 D trigger reset. This bit is reset by writing a 1. The reading is always 0.
[2]	C0CLKINV	CLU0 D flip-flop clock level is inverted. 0: Off 1: Enable
[1:0]	C0CLKSEL	CLU0 D Trigger Clock Select. 0x0: carry input 0x1: MXA input 0x2: ALTCLK0 0x3: ALTCLK1

24 CAN

Controller Area Network (CAN) bus is a kind of bus standard which can realize the communication between microprocessors or devices without a host.

24.1 Main characteristics

The CAN controller supports BOSCH 2.0 A and 2.0 B protocols following the CAN bus. 2.0 A is equivalent to CAN1.2 and includes an 11-bit ID format; 2.0 B contains an 11-bit ID and a 29-bit ID.

The CAN bus controller can process the data transmission and reception on the bus. In this product, the CAN controller has 4 groups of filters. Filters are used to select messages to be received for an application.

In the CAN controller, the Tx Buffer includes two groups of buffers: a high-priority primary Tx Buffer (Primary Transmit Buffer, hereinafter referred to as PTB) and a low-priority secondary Tx Buffer (Secondary Transmit Buffer, STB hereafter). PTB contains one Slot, STB contains two Slots, and the sending order is determined by hardware scheduling. A Receive Buffer (hereinafter referred to as Rx Buffer) acquires bus data, and the Rx Buffer contains ten Slots. Slot is the unit of CAN storage (13-byte)

24.2 Functional description

24.2.1 Functional block diagram

The interface adopts synchronous serial design to realize CAN transmission between MCU and external equipment. Support polling and interrupt mode to obtain transmission status information. The main functional modules of this interface are shown in the figure below.

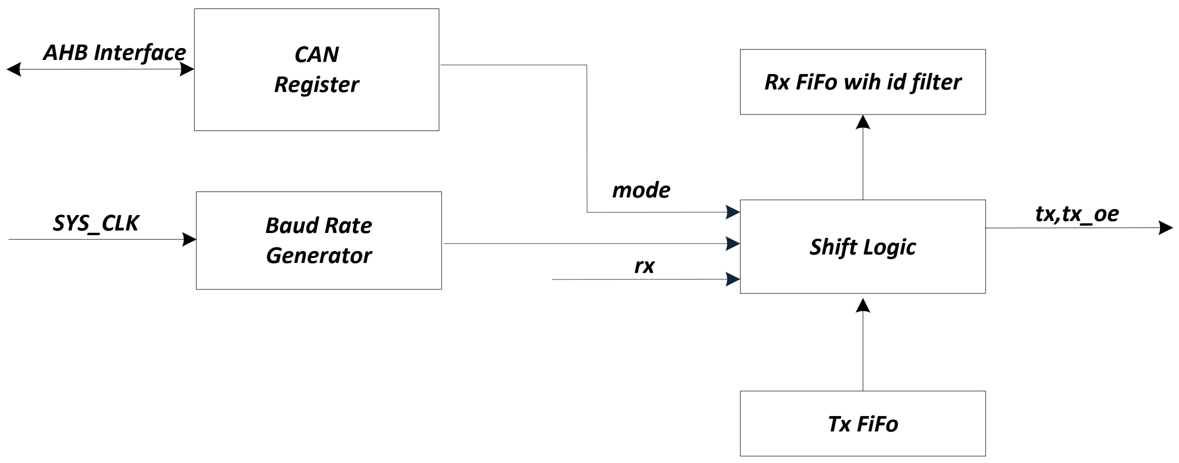


Figure24-1 CAN Module Top Level Functional Block Diagram

The CAN interface has only two signal lines, tx and rx, to communicate with the outside world. When tx sends valid data, tx_OE is valid. When tx does not send valid data, tx_OE is invalid.

Rx: data signal. Receives CAN data from outside.

Tx: data signal. Sends CAN data to the CAN bus.

Tx_OE: Data enable signal. When tx is output, tx_OE is valid; When tx has no data output, tx_OE is invalid.

24.2.2 Functional description

The CAN module receives and transmits data and converts data from serial to parallel or from parallel to serial. Interrupts can be enabled or disabled. The interface is connected to the PHY chip of the CAN bus through a data output pin (TX) and a data input pin (RX).

24.2.2.1 Working mode

The CAN module mainly consists of two operating modes: normal operating mode and reset mode.

Reset mode. Timing parameters, ID configuration, error statistics, etc. Are set in this mode. CAN_CFG_STAT. RESET is 1, which is the reset mode. After the hardware reset, the CAN module is in reset mode.

Normal operating mode, CAN_CFG_STAT.RESET is 0. At this time, the CAN bus request can be responded normally.

In the above two modes, the Listen Only and Self Test modes are extended. The former, like a collector, only receives data on the CAN bus and does not send any data. The latter is an internal self-test, in which the transmitted data is received by itself at the same time to check whether the internal function is correct.

24.2.2.2 Transmission

Move 11 bytes or 13 bytes according to SFF (Standard Frame Format) or EFF (Extended Frame Format); After completion, it is necessary to judge whether the transmission is completed by means of interruption or polling.

For MCU transmission, the recommended software configuration process is as follows:

Initialize the GPIO module and complete the configuration of the GPIO for CAN multiplexing.

The CAN interface is initialized and the control register is configured.

Write to PTB/STB via the CAN_TBUF register

Trigger the CAN interface to enter the sending process

24.2.2.3 Interrupt processing

The CAN-interface contains many interrupts, which are described in the CAN_RTIE and CAN_ERRINT register descriptions. Turn on the corresponding interrupt event enable switch according to the actual use.

The CAN controller provides the following seven interrupts:

- Receive interrupt
- Transmit interrupt
- Error interrupt
- Data overflow interrupt
- Passive error interrupt
- Arbitration loss interrupt
- Bus error interrupt

24.2.2.3.1 Receive interrupt

Receive interrupt CAN_RTIE.RIE.When the CAN Rx Buffer receives a valid frame, the CAN_RTIF. RIF is set. The receive interrupt is further refined, and there are three interrupt sources CAN_RTIE. ROIE, CAN_RTIE.RFIE, and CAN_RTIE.RAFIE. CAN Rx Buffer Receive Overflow Interrupt, CAN Rx Buffer Receive Full Interrupt, and CAN Rx BufferReceive Near Full Interrupt. Therefore, when a receive interrupt is generated, several flag bits may be set.

The flag bit is not set until the interrupt is enabled.

24.2.2.3.2 Transmit interrupt

CAN Tx Buffer has two sources, one is PTB and the other is STB. There are also two corresponding interrupts, the CAN_RTIE. TPIE and the CAN_RTIE. TSIE. In addition, because this CAN device supports sending a terminated transmission, there is a CAN_RTIF. AIF for the terminated transmission status bit (no interrupt enable flag).

The flag bit is not set until the interrupt is enabled.

24.2.2.3.3 Error interrupt

During CAN transmission, bus error events occur, including receive error and transmit error. Generally, when an error occurs, the corresponding error counter is incremented, and different types of error interrupts are generated according to the value of the counter.

Error interrupt CAN_RTIE.EIE. If the value of the CAN_RECNT/CAN_TECNT error counter goes from being less than the value specified by the CAN_LIMIT.EWL register to exceeding the value specified by the CAN_LIMIT.EWL register, or vice versa; Entering the BusOff state from the non-BusOff state, or vice versa, triggers this interrupt, and the interrupt flag is the CAN_RTIE.EIF.

Passive Error Interrupt CAN_ERRINT.EPIE. This interrupt is triggered when the CAN_RECNT/CAN_TECNT error counter value exceeds the specified value (> 127), the CAN device goes from an active error state to a passive error state, or vice versa (where the error counter value is < 127). The interrupt flag is a CAN_ERRINT.EPIF.

The flag bit is not set until the interrupt is enabled.

24.2.2.3.4 Data overflow interrupt

When the CAN Rx Buffer is fully received, the data is not taken away by the MCU in time, and if a valid frame is received again, the receiving overflow interrupt CAN_RTIE.ROIE is triggered, and the interrupt flag bit is a CAN_RTIE.ROIF.

24.2.2.3.5 Arbitration lost interrupt

Arbitration Lost Interrupt CAN_ERRINT.ALIE. The CAN bus supports the arbitration mechanism. If the CAN device loses the bus right during the bus competition, the arbitration loss interrupt will be triggered. The interrupt flag is a CAN_ERRINT.ALIF.

The flag bit is not set until the interrupt is enabled.

24.2.2.3.6 Bus error interrupt

Bus error interrupt CAN_ERRINT.BEIE. Bus error interrupt will be triggered when bus error event occurs during CAN transmission. The interrupt flag is a CAN_ERRINT.BEIF. The difference between a bus error interrupt and an error interrupt is that the error interrupt is triggered only after the bus error has accumulated to a certain amount.

The flag bit is not set until the interrupt is enabled.

24.2.2.4 Baud rate setting

The setting of the baud rate mainly depends on the CAN_SBAUD register. The CAN_SBAUD is mainly used to configure the TQ parameter (see the CAN_SBAUD register description for the calculation of TQ), the sampling point configuration, and the tolerance range information.

The CAN_SBAUD.PRESC sets the transmission basic time unit parameter TQ: TQ = Tclk * (CAN_SBAUD.PRESC + 1)

The LKS07x has a maximum clock speed of 96M, corresponding to a Tclk of 10.4 ns and a maximum TQ of 2.6624 us.

The CAN_SBAUD sets the baud rate; Meanwhile, the width of each part (TSEG1 and TSEG2) in the BIT information can be adjusted to find the ideal sampling point.

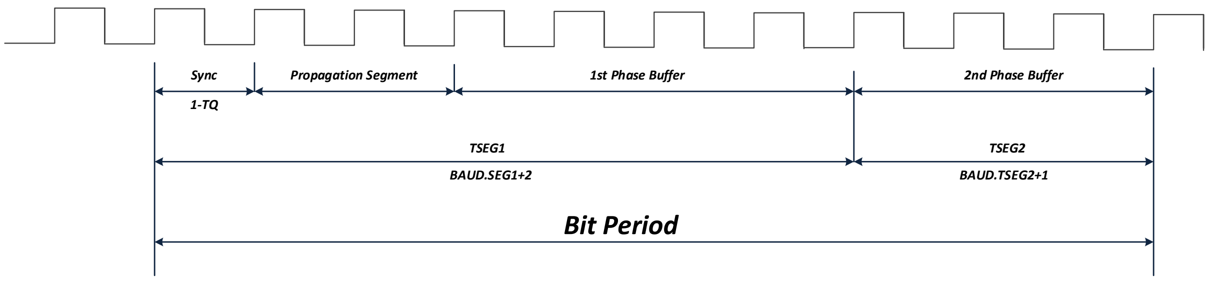


Figure24-2 CAN module Bit cycle introduction diagram

As shown in the figure above, TSEG1 consists of three parts: Sync, Propagation Segment, and 1stPhase Buffer. Sync is the synchronization of this BIT, with 1 TQ fixed. Propagation Segment is the propagation delay. 1stPhase Buffer is the first segment phase buffer. In the specific CAN network, it is necessary to pay attention to the parameter of Propagation Segment, which determines how long a data needs to be transmitted at least. Propagation Segment also directly affects the configuration of SEG1. In this design, the above three parameters are merged into one parameter: TSEG1. TSEG2 contains only one part of 2ndPhase Buffer, which is the second phase buffer that compensates for errors in the edge (Sync) phase and is lengthened or shortened by resynchronization. The length of TSEG1 and TSEG2 determines the sampling point of CAN data, which allows a certain range of data transmission delay and crystal error. The TSEG1 is used to adjust the error caused by the data transmission delay time, and the TSEG2 is used to adjust the error of the crystal frequency at different nodes.

SEG1 segment time calculation formula: TSEG1 = TQ * (CAN_SBAUD.SEG1 + 2)

SEG2 section time calculation formula: TSEG2 = TQ * (CAN_SBAUD.SEG2 + 1)

The baud rate is calculated as: **CAN Baudrate = 1/(TSEG1 + TSEG2)**

CAN communication belongs to asynchronous communication, which requires continuous resynchronization to ensure accurate sampling of the receiving and transmitting nodes. Therefore, SJW (synchronous jump width) determines whether the receiving node has good compatibility. The CAN_SBAUD. SJW is the SJW bit, which triggers the synchronization jump register. How much communication time error is allowed for each device communicating at a certain baud rate.

Bus lower bound tolerance < bus baud rate < bus upper bound tolerance

The tolerance is calculated as: TQ * (SJW + 1)

To configure the CAN_SBAUD register, generally refer to the following:

- TSEG1 is slightly larger than TSEG2, and the sampling point will be slightly later than the middle point to ensure the stability of the signal. It can also be adjusted according to the actual situation, so that the sampling point can be moved back and forth. TSEG2 > = 2 in general (i.e. SEG2 > = 1). BOSCH specifies that TSEG1 + TSEG2 > = 8. Generally, when the chip clock is fast but the baud rate is relatively low, it can be realized by combining the CAN_SBAUD.PRESC, the CAN_SBAUD.SEG1 and the CAN_SBAUD.SEG2.
- The value of SJW should not be greater than SEG2, and it is generally reasonable to be half of the latter.
- For the same CAN network, the baud rate configuration is as close as possible or consistent as possible, which is a better choice. Due to the different frequency of each chip, the CAN_SBAUD. PRESC is different. The baud rate configuration is as consistent as possible, and the TQ generated by dividing the CAN_SBAUD.PRESC is as consistent as possible. If TQ is consistent, TSEG1, TSEG2 and SJW can be basically consistent.

Based on the LKS07x chip, some baud rate index values are as follows:

Table24-1The structure of the received frame

Can baud rate	CAN_SBAUD.PRESC[7:0]	CAN_SBAUD.SJW[6:0]	CAN_SBAUD.SEG2[6:0]	CAN_SBAUD.SEG1[7:0]	Sample Point
1Mbps	0x05	0x02	0x05	0x08	63%
800Kbps	0x0B	0x01	0x03	0x04	60%
666Kbps	0x0B	0x01	0x03	0x06	67%
500Kbps	0x0B	0x02	0x05	0x08	63%
400Kbps	0x0B	0x02	0x06	0x0B	65%
250Kbps	0x17	0x02	0x05	0x08	63%
200Kbps	0x17	0x02	0x06	0x0B	65%
125Kbps	0x2F	0x02	0x05	0x08	63%
100Kbps	0x2F	0x02	0x06	0x0B	65%
80Kbps	0x35	0x02	0x06	0x0B	65%
50Kbps	0x5F	0x02	0x06	0x0B	65%
40Kbps	0x77	0x02	0x06	0x0B	65%
25Kbps	0xBF	0x02	0x06	0x0B	65%
20Kbps	0xEF	0x02	0x06	0x0B	65%
10Kbps	0xEF	0x06	0x0D	0x18	65%
05Kbps	0xEF	0x0D	0x1B	0x32	65%

24.2.2.5 ID filter

The CAN bus can be attached to many devices. Through the ID number, different devices can know whether the frame sent on the current bus needs to be received by themselves, or whether the frame sent out has a response. This CAN device has 4 sets of ID filters (ACF0/1/2/3). The MCU can configure the ID filter only when the CAN device is in reset mode. The ID number of the CAN frame has two lengths -- 11 bits and 29 bits. The former corresponds to SFF (Standard Frame Format), and the latter corresponds to EFF (Extended Frame Format). The ID filtering length of the CAN device is 29 bits.

If the currently received frame passes one of the ID filters, it will be accepted by the CAN device. If accepted, the data for this frame is stored in the Rx Buffer and RIF is asserted if RIE is enabled. If the frame? Is not accepted, the RIF is not set and the Rx Buffer pointer is not incremented. Frames that are not accepted are discarded and overwritten with the next frame. Any stored valid frames will not be overwritten by any unaccepted frames.

The ID range that can be received by the current CAN equipment is determined by the CAN_ACR and the CAN_AMR. The CAN_ACR lists a specific ID, and the CAN_AMR is the corresponding MASK register, which identifies whether the corresponding bit data in the CAN_ACR needs to be matched with the bit data corresponding to the received ID. A bit of the CAN_AMR is 0, indicating that this bit needs to match the CAN_ACR; A bit of the CAN_AMR is 1, indicating that this bit does not need to match the CAN_ACR. The extremes are matching every bit of the CAN_ACR, or not matching every bit.

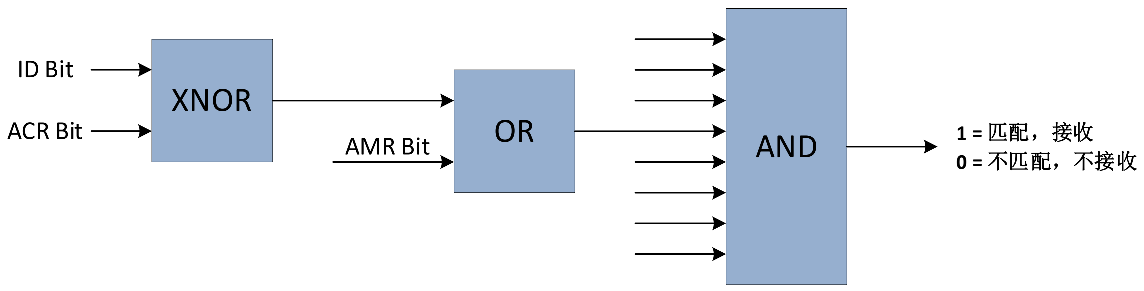


Figure24-3 CAN module ID filtering logic

The CAN equipment realizes the configuration of CAN_ACR0/1/2/3 and CAN_AMR0/1/2/3 through three groups of registers including CAN_ACFCTRL, CAN_ACFEN and CAN_ACF. The CAN_ACRx and CAN_AMRx are grouped into four sets of ID filters (ACF0/1/2/3). The CAN_ACFCTRL is a selection register for selecting which group of filters will be selected by the MCU; the CAN_ACFEN is an enable register for selecting which group of filters will be enabled/closed; The CAN_ACRCTRL selects ACRx/AMRx, and the next step is to read and write ACRx/AMRx by accessing the CAN_ACF. By default, ACF0 is enabled with CAN_ACR0 all 0s and CAN_AMR0 all 1s.

The CAN_ACRx and CAN_AMRx of this CAN equipment are designed according to the maximum 29-bit. If it is a standard frame, the ID is only 11-bit. At this time, the high bit corresponding to the CAN_AMRx can be shielded (write 1).

- Standard frame: write the ID value to be filtered in the CAN_ACRx [10:00]; write all 1s in the CAN_AMRx [28:11], and write the corresponding value in the CAN_AMRx [10:00].
- Extension frame: write the ID value to be filtered in the CAN_ACRx [28:00]; write the corresponding value in the CAN_AMRx [28:00].

24.2.2.6 Reception of can frames

Valid and accepted frames received are stored in the Rx Buffer. One frame of data is stored in one Slot, and the CAN device and Rx Buffer provide 10 Slots. The Rx Buffer is stored and read out according to the FIFO mode, that is, the frame stored in the Rx Buffer is read by the MCU first.

- If CAN_RTIE.RIE is enabled, the CAN_RTIF.RIF is set to 1 each time a valid and accepted frame is received.
- If the CAN_RTIE.RAFIE is enabled and the number of valid and accepted frames received reaches the value specified by the CAN_LIMIT.AFWL, the CAN_RTIF.RAFIF will be set to 1.
- If the CAN_RTIE.RFIE is enabled, the Rx Buffer is full of received valid and accepted frames, and the CAN_RTIF.RFIF is set to 1.

The frame that is the oldest stored in the Rx Buffer Slot and that has not been taken is read through the CAN_RBUF register. After the MCU finishes reading, set the CAN_RCTRL.RREL to 1 (the hardware automatically returns to zero), and release the Rx Buffer Slot that has been read. The CAN_RBUF register automatically points to the next Rx Buffer Slot.

The CAN device can continue to receive frames even if the Rx Buffer is full. Once a frame is received that is required by the CAN device (through the ID filter), it is determined how to process it according to the configuration of the CAN_RCTRL.ROM. When the CAN_RCTRL.ROM is 1, the newly received frame is discarded; when the CAN_RCTRL.ROM is 0, the frame that enters the Rx Buffer earliest is discarded.

The structure of the can frame is divided into an ID part and a data part. The first byte contains information such as frame classification to determine whether it is an SFF (standard) frame or an EFF (extended) frame; Determining whether it is a remote frame or a data frame; And determine that length of the data. The ID for SFF is 2 bytes long and the ID for EFF is 4 bytes long. The maximum data length is 8 bytes.

Table24-2The structure of the received frame

SFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_RBUF0[31:24]	/							
CAN_RBUF0[23:16]	/							
CAN_RBUF0[15:08]	/					ID[10:08]		
CAN_RBUF0[07:00]	ID[07:00]							
CAN_RBUF1[31:24]	/							
CAN_RBUF1[23:16]	/							
CAN_RBUF1[15:08]	KOER[2:0]			TX	/			
CAN_RBUF1[07:00]	0	RTR	/		DLC[3:0]			
CAN_RBUF2[31:24]	DATA3							
CAN_RBUF2[23:16]	DATA2							
CAN_RBUF2[15:08]	DATA1							
CAN_RBUF2[07:00]	DATA0							
CAN_RBUF3[31:24]	DATA7							
CAN_RBUF3[23:16]	DATA6							
CAN_RBUF3[15:08]	DATA5							
CAN_RBUF3[07:00]	DATA4							
EFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_RBUF0[31:24]	/			ID[28:24]				
CAN_RBUF0[23:16]	ID[23:16]							
CAN_RBUF0[15:08]	ID[15:08]							
CAN_RBUF0[07:00]	ID[07:00]							
CAN_RBUF1[31:24]	/							
CAN_RBUF1[23:16]	/							
CAN_RBUF1[15:08]	KOER[2:0]			TX	/			
CAN_RBUF1[07:00]	1	RTR	/		DLC[3:0]			
CAN_RBUF2[31:24]	DATA3							
CAN_RBUF2[23:16]	DATA2							
CAN_RBUF2[15:08]	DATA1							
CAN_RBUF2[07:00]	DATA0							
CAN_RBUF3[31:24]	DATA7							
CAN_RBUF3[23:16]	DATA6							
CAN_RBUF3[15:08]	DATA5							
CAN_RBUF3[07:00]	DATA4							

IDE (IDentifier Extension): 0: standard frame format; 1: extended frame format

RTR (Remote Transmission Request): 0: data frame; 1: remote frame

DLC (Data Length Code). The setting range is 0 ~ 8, and the corresponding data length is 0 Byte ~ 8 Byte

KOER: Same as CAN_EALCAP.KOER.

TX: This bit is set to 1 when receiving its own transmitted data in loopback mode.

The CAN frame is received as follows:

- Configure the ID filter
- Configure CAN_RTIE.RFIE, CAN_RTIE.RAFIE, and CAN_LIMIT.AFWL
- Wait for CAN_RTIF.RFIF or CAN_RTIF.RAFIF (either interrupt or poll)
- Reading the earliest received can frame from the Rx Buffer through the CAN_RBUF; Setting the CAN_RCTRL.RREL to 1 (hardware auto-zeroing) releases the Rx Buffer Slot that has been read, and the CAN_RBUF register automatically points to the next Rx Buffer Slot. This is repeated until the Rx Buffer is confirmed to be empty by the CAN_RCTRL.RSTAT.

24.2.2.7 Transmission of CAN frame

The CAN module must be in normal operating mode to send data. At least one frame is written into the transmit buffer (Primary Transmit Buffer or Secondary Transmit Buffer) before any transmission begins. The CAN_TBUF register provides access to the Primary Transmit Buffer (PTB) and the Secondary Transmit Buffer (STB). This CAN device preferentially sends the CAN frame of PTB. In case of idle transmission, the CAN_TCMD.TPE is configured as 1 to trigger the PTB to transmit data. At this time, the STB can only transmit data after the PTB is completely transmitted.

The storage depth of PTB is one, that is, only one frame can be stored, and STB can store two frames. When the CAN_TCMD.TBSEL is 1, select STB; when it is 0, select PTB. When the CAN_TCMD. TBSEL is 1, the MCU can write to the STB. Because two frames of data can be stored, the CAN_TBUF sets the CAN_TCTRL.TSNEXT after writing the current frame, and the hardware automatically jumps to the next STB Slot. Data from all CAN_TBUF can be written in any order. The CAN_TCTRL.TSSTAT may know the state of the STB.

The CAN frame is sent as follows:

- Configure the CAN_TCMD.TBSEL and select which Tx Buffer to send data (0: PTB; 1: STB)
- Writes the transmitted data to the CAN_TBUF register
- If STB is selected, set TSNEXT = 1 to finish loading the STB Slot
- The CAN_TCMD.TPE writes 1 to trigger the transmission of the can frame of the PTB; Writing 1 in the CAN_TCMD.TSALL or CAN_TCMD.TSONE triggers the STB to send the can frame
- Send completion status confirmation. When the CAN_RTIE.TPIE is enabled and the can frame of PTB is sent, the CAN_RTIF.TPIF is set; when the CAN_TCMD.TSONE is in single frame mode and the CAN_RTIE.TSIE is enabled and the can frame of STB is sent, the CAN_RTIF.TSIF is set; CAN_TCMD.TSALL multi-frame mode, CAN_RTIE.TSIE enable, all can frames of STB, after transmission is completed, the CAN_RTIF.TSIF is set

- Note that if both STB and PTB trigger the sending request, under the same conditions, PTB sends first.

This CAN device sends CAN frames and supports CAN arbitration. Once the arbitration fails, the frame can be retransmitted when the bus is free, and the frame can not be retransmitted again. CAN_CFG_STAT.TPSS configures whether to retransmit PTB (0: retransmit; 1: single), CAN_CFG_STAT. The TSSS configures whether the STB retransmits (0: retransmit; 1: single). When configured in single transmit mode, the CAN_RTIF.TPIF/CAN_RTIF.TSIF is set and the corresponding Slot is cleared (data stale) regardless of whether the last frame was successfully transmitted. In this case, two other status bits are required to determine:

- When a bus error occurs, the CAN_EALCAP.KOER is updated and the CAN_ERRINT.BEIF is set (CAN_ERRINT.BEIE enabled).
- Arbitration failed, CAN_ERRINT.ALIF set (CAN_ERRINT.ALIE enabled)

The maximum payload length of a can frame is 8 bytes. The data length per frame is defined by the DLC. For remote frames (bit RTR), the value of DLC is meaningless because the data length of a remote frame is always 0 bytes.

Table24-3The structure of the transmit frame

SFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_TBUF0[31:24]	/							
CAN_TBUF0[23:16]	/							
CAN_TBUF0[15:08]	/					ID[10:08]		
CAN_TBUF0[07:00]	ID[07:00]							
CAN_TBUF1[31:24]	/							
CAN_TBUF1[23:16]	/							
CAN_TBUF1[15:08]	/							
CAN_TBUF1[07:00]	0	RTR	/		DLC[3:0]			
CAN_TBUF2[31:24]	DATA3							
CAN_TBUF2[23:16]	DATA2							
CAN_TBUF2[15:08]	DATA1							
CAN_TBUF2[07:00]	DATA0							
CAN_TBUF3[31:24]	DATA7							
CAN_TBUF3[23:16]	DATA6							
CAN_TBUF3[15:08]	DATA5							
CAN_TBUF3[07:00]	DATA4							
EFF								
	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
CAN_TBUF0[31:24]	/			ID[28:24]				
CAN_TBUF0[23:16]	ID[23:16]							
CAN_TBUF0[15:08]	ID[15:08]							
CAN_TBUF0[07:00]	ID[07:00]							
CAN_TBUF1[31:24]	/							
CAN_TBUF1[23:16]	/							

CAN_TBUF1[15:08]	/			
CAN_TBUF1[07:00]	1	RTR	/	DLC[3:0]
CAN_TBUF2[31:24]	DATA3			
CAN_TBUF2[23:16]	DATA2			
CAN_TBUF2[15:08]	DATA1			
CAN_TBUF2[07:00]	DATA0			
CAN_TBUF3[31:24]	DATA7			
CAN_TBUF3[23:16]	DATA6			
CAN_TBUF3[15:08]	DATA5			
CAN_TBUF3[07:00]	DATA4			

Data transmission that has been requested but not yet performed can be canceled by configuring the CAN_TCMD.TPA or the CAN_TCMD.TSA. Attention shall be paid to the following situations:

- In arbitration. If the node arbitration fails, the data transmission is canceled; If the node arbitration is successful, continue to send
- Data is being sent. If the data is successfully sent and ACK is received, the corresponding flag and status are set normally, and the data sending is not cancelled; Data is successfully sent but no ACK is received, data transmission is canceled, and the error counter is incremented. CAN_TCMD.TSALL sending, the STB Slot being sent is sent normally, and the STB Slot that has not started sending is cancelled

24.2.2.8 Error Management

The CAN protocol requires a transmit error count and a receive error count in each node. The values of these two error counts determine the current error state of the controller (e.g., active error, passive error, offline). The controller stores the values in the CAN_RECNT and the CAN_TECNT respectively, and the MCU can read the values at any time. In addition to the error state, the controller also provides the function of error alarm limitation (CAN_LIMIT.EWL), which can warn the user of the current serious bus error before the controller enters the passive error state.

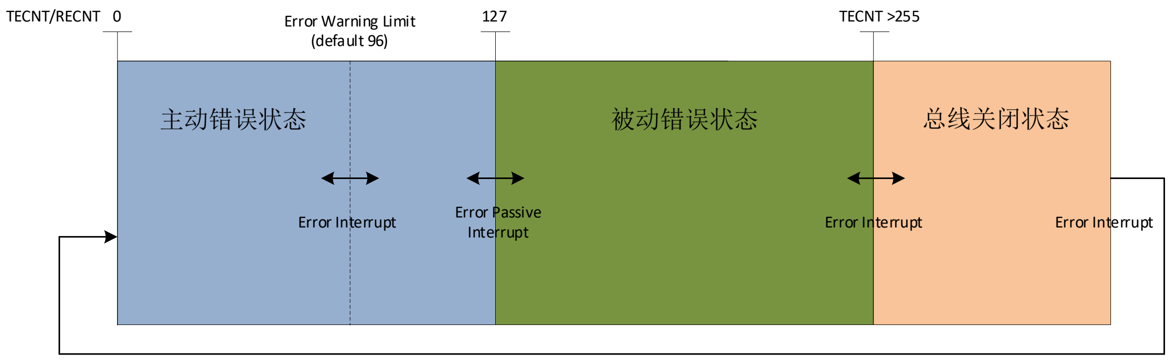


Figure24-4 CAN module error management

24.2.2.9 Error count

The transmit error count and the receive error count are incremented/decremented according to the following table. Note that more than one rule can be applied in a frame transmission.

Table24-4 CAN Error Count and Receive Error Count Rules

	Variation condition of receive and transmit error count values	Send Error Count (TECNT)	Receive Error Count (RECNT)
1	When the receiving unit detects an error Exception: When the receiving unit detects a "bit error" in the transmit error flag or the overload flag, the receive error count is not incremented	-	+1
2	The first bit detected by the receiving unit after sending the error flag is a dominant level	-	+8
3	When the sending unit outputs the error flag,	+8	-
4	The sending unit detects a bit error when sending the active error flag or the overload flag	+8	-
5	The receiving unit detects a bit error when sending the active error flag or the overload flag	-	+8
6	Each unit detects the dominant level of continuous 14 bits from the beginning of the active error flag and the overload flag, and then detects the dominant level of continuous 8 bits every time	When sending +8	When receiving +8
7	When 8 consecutive dominant bits added after the passive error flag are detected	When sending +8	When receiving +8
8	When the transmission unit normally transmits data (when ACK is returned and no error is detected by the end of the frame)	TECNT = 0 when TECNT = 0 TECNT-1	-

9	When the receiving unit normally receives data (until the CRC detects no error and normally returns ACK)	-	Recnt = 0 when Recnt = 0
			RECNT -1 when 1 < = RECNT < = 127
			RECNT = 127 for RECNT > 127

10	Cell in bus off state, 11 consecutive recessive bits detected 128 times	=0	=0

24.2.2.10 Active error state

When the values of the transmit error counter (CAN_TECNT) and the receive error counter (CAN_RECNT) are both < = 127, it is an active error state; CAN equipment in active error state can normally participate in CAN bus communication, and can send active error signals (6 consecutive dominant signals) when an error is detected.

24.2.2.11 Passive error state

When the value of the transmit error counter (CAN_TECNT) or the receive error counter (CAN_RECNT) is > 127 but < 256, it is a passive error state; CAN equipment in passive error state cannot send active error signal (6 consecutive dominant signals) when an error is detected; Only a passive error signal (recessive signal) can be issued, waiting for the error to be detected by other CAN devices.

24.2.2.12 Shutdown state and shutdown recovery

When the value of the transmit error counter (CAN_TECNT) or the receive error counter (CAN_RECNT) is greater than 255, the present CAN device automatically enters a shutdown state so as not to participate in the communication of the CAN bus until it returns to an active error state. The status of this CAN device can be confirmed by the CAN_CFG_STAT.BUSOFF bit. When the BUSOFF bit is set, a CAN_RTIF.EIF interrupt is generated.

There are two ways to return the CAN from the OFF state to the active error state:

- Power-on reset
- A recessive bit sequence (recovery sequence) of 11 consecutive bits is received 128 consecutive times.

When the node is closed, the CAN_TECNT value remains unchanged, and the CAN_RECNT is used to count the recovery sequence. The CAN_TECNT and CAN_RECNT are reset to 0 upon recovery from the node off state.

24.2.2.13 Arbitration

This device can accurately capture the location of the arbitration failure and reflect it in the CAN_EALCAP.ALC register. The CAN_EALCAP.ALC register holds the location of the last arbitration failure, and the CAN_EALCAP.ALC bit is not updated if the device wins arbitration.

The CAN_ALC values are defined as follows:

After the SOF bit, the CAN_EALCAP.ALC of the first ID data bit is 0, the CAN_EALCAP.ALC of the second ID data bit is 1, and so on. Because arbitration occurs only in arbitration fields, the maximum CAN_EALCAP.ALC is 31. For example, for a standard format remote frame and an extended frame arbitration, the extended frame will fail to compete for the bus in the IDE bit, and the CAN_EALCAP. ALC = 12.

24.2.3 Register

24.2.3.1 Address assignment

The CAN module register base address is 0x4001_1300. The register list is shown below.

Table24-5 CAN register address assignment

Name	Offset	Explain
CAN_RBUF0	0x00	CAN Rx Buffer 0
CAN_RBUF1	0x04	CAN Rx Buffer Register 1
CAN_RBUF2	0x08	CAN Rx Buffer 2
CAN_RBUF3	0x0C	CAN Rx Buffer Register 3
CAN_TBUF0	0x50	CAN Tx Buffer Register 0
CAN_TBUF1	0x54	CAN Tx Buffer Register 1
CAN_TBUF2	0x58	CAN Tx Buffer 2
CAN_TBUF3	0x5C	CAN Tx Buffer Register 3
CAN_CFG_STAT	0xA0	CAN configuration and status register
CAN_TCMD	0xA1	CAN Transmit Command Register
CAN_TCTRL	0xA2	CAN Transmit Control Register
CAN_RCTRL	0xA3	CAN receive control register
CAN_RTIE	0xA4	CAN Transmit Receive Interrupt Control Register
CAN_RTIF	0xA5	CAN Transmit Receive Interrupt Flag register
CAN_ERRINT	0xA6	CAN Error Interrupt Enable and Flag Register
CAN_LIMIT	0xA7	CAN warning register
CAN_SBAUD	0xA8	CAN baud rate configuration register
CAN_EALCAP	0xB0	CAN Error Message and Missing Arbitration Message Record Register

CAN_RECNT	0xB2	CAN receive error counter register
CAN_TECNT	0xB3	CAN transmit error counter register
CAN_ACFCTRL	0xB4	CAN ID Filter Control Register
CAN_ACFEN	0xB6	CAN ID Filter Enable Register
CAN_ACF	0xB8	CAN ID Filter Select Register

24.2.3.2 Register description

24.2.3.2.1 CAN_RBUF0/1/2/3 Register

0x4001_1300/4/8/C

Reset value: X

Table24-6Read Register CAN_RBUF0/1/2/3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA															
RO															
X															

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

Location	Bit name	Explain
[31:0]	RDATA	Receive register for reading the data of the CAN-frame from the Rx Buffer Slot.

The data contents of CAN_RBUF0/1/2/3 are different due to different frame types and data lengths. Analyze the structure of the received frame according to the document.

24.2.3.2.2 CAN_TBUF0/1/2/3 register

0x4001_1350/4/8/C

Reset value: X

Table24-7Write to Register CAN_TBUF0/1/2/3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDATA															
WO															
X															

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

Location	Bit name	Explain
[31:0]	WDATA	Transmit register, write data to Tx Buffer Slot (PTB or STB)

The data contents of CAN_TBUF0/1/2/3 are different due to different frame types and data lengths. Analyze the structure of the sending frame according to the document. Do not fill in the bit, fill in 0.

24.2.3.2.3 CAN_CFG_STAT configuration and status register

Address: 0x4001_13 A0

Reset value: 0x0

Table24-8Configuration and status register CAN_CFG_STAT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
								RW	RW	RW	RW	RW	RO	RO	RW
								1	0	0	0	0	0	0	0

Location	Bit name	Explain
[7]	RESET	1: CAN module, reset mode 0: CAN module, normal operating mode In the reset mode, some circuits of the CAN module are in the reset state, and some registers can only be configured in the reset mode; In normal operating mode, can frames can be sent and received. Note that when switching from the reset mode to the normal working mode, it is necessary to wait for 11 CAN Bit Time (for example, when the baud rate is 500K, one Bit Time is 2us, and it is necessary to wait for 22us to send or receive a can frame).
[6]	LBME	External loopback mode enable bit 1: Enable 0: Off Note that this bit is disabled during normal communication
[5]	LBMI	Internal loopback mode enable bit 1: Enable 0: Off Note that this bit is disabled during normal communication
[4]	TPSS	PTB Single Transfer Mode Enable Bit 1: Enable 0: Off
[3]	TSSS	STB Single Transfer Mode Enable Bit 1: Enable 0: Off
[2]	RACTIVE	Receive status flag bit

		0: This CAN device, receiving idle, bus idle 1: This CAN device is receiving CAN frame
[1]	TACTIVE	Send status flag bit 0: This CAN device, send idle 1: This CAN device is sending CAN frame
[0]	BUSOFF	This CAN device bus off status bit (Bus Off) 0: This CAN device is not in the bus off status bit. 1: This CAN device is in the bus off status bit. Note: Writing a 1 to BUSOFF clears the TECNT and RECNT registers, but only for project debugging

24.2.3.2.4 CAN_TCMD transmit command register

Address: 0x4001_13A1

Reset value: 0x0

Table24-9Transmit Command Register CAN_TCMD.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								TBSEL	LOM	--	TPE	TPA	TSONE	TSALL	TSA
								RW	RW	--	RO	RO	RO	RO	RO
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[7]	TBSEL	Tx Buffer selection bit, which is written into the corresponding Buffer through the CAN_TBUF. 1：STB 0：PTB Note that the value of TBSEL cannot be changed while the CAN_TBUF is being written; Meanwhile, when the STB is written, TSNEXT needs to be updated to complete the writing of a frame.
[6]	LOM	Snoop Mode Enable Bit 1: Enable 0: Off LOM cannot be set if TPE, TSONE, or TSALL is set. If LOM is enabled and LBME is disabled, the transfer cannot be started. LOM = 1 and LBME = 0 prohibit all transmissions. When LOM = 1 and LBME = 1, it is prohibited to reply to the corresponding received frame and error frame, but the frame of this CAN device can be sent. Note that this bit is disabled during normal communication
[5]	--	Keep the default values
[4]	TPE	PTB Transmit Enable Bit 1: Trigger PTB sending 0: Do not trigger PTB transmission TPE is configured, and in the next window that can be sent, the STB

		sends the PTB data first (unless the STB is currently sending, the PTB is sent first). Once written, this bit remains set to 1 until the PTB transmission is complete or the transmission is canceled by the TPA. Software cannot clear this bit by writing a 0. There are several other situations in which a TPE can be cleared: 1. CAN_CFG_STAT.RESET=1 2. CAN_TCMD.LOM = 1 and CAN_CFG_STAT.LBME = 0
[3]	TPA	PTB sends a cancel bit 1: Cancel the PTB sending that has been requested through TPE setting 1 but has not started yet 0: Do not cancel Set by software and reset by hardware automatically. Setting TPA will automatically clear TPE. Generally, TPA and TPE are not set at the same time. CAN_CFG_STAT.RESET = 1 clears TPA.
[2]	TSONE	Transmitting a frame of STB data control bits (Transmit Secondary ONE Frame) 1: Send a frame of STB data 0: Do not send When TSONE is configured as 1, triggering transmission, successful transmission or TSA termination transmission can realize hardware zero clearing, otherwise, it will remain 1. CAN_CFG_STAT. RESET = 1, can also be cleared.
[1]	TSALL	Transmit multiframe STB data control bit (Transmit Secondary All Frames) 1: transmit multi-frame STB data 0: Do not send When TSALL is configured as 1, the transmission is triggered. If the transmission is successful or the TSA terminates the transmission, the hardware can be cleared. Otherwise, it remains 1. CAN_CFG_STAT. RESET = 1, can also be cleared.
[0]	TSA	The STB sends a cancel bit 1: Cancel the STB transmission that has been requested through TSONE or TSALL but has not been started, write 1 in software and reset in hardware. Writing a 1 clears the TSONE or TSALL bit 0: Do not cancel General TSA and TSONE/TSALL are not set at the same time. CAN_CFG_STAT. RESET = 1 clears TSA

24.2.3.2.5 CAN_TCTRL transmit control register

Address: 0x4001_13A2

Reset value: 0x0

Table24-10Transmit control register CAN_TCTRL.

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

		RW		RW
		0		0

Location	Bit name	Explain
[7]	-	Keep the default values
[6]	TSNEXT	Update STB Slot Location 1: Update 0: Do not update When the current STB Slot is filled with data, the software sets TSNEXT, and the hardware automatically points to the next STB Slot. At this time, TSONE/TSALL is set, and the filled STB Slot can be sent. You can set both TSNEXT and TSONE or TSALL. If TBSEL = 0, there is no point in setting TSNEXT, and the configuration of TSNEXT is ignored and cleared automatically. If all Slots of the STB are filled, TSNEXT will remain set until the Slot is free, and will not be cleared by the hardware.
[5:2]	-	Keep the default values
[1:0]	TSSTAT	STB status bit 3: STB is full 2: STB is more than half full 1: STB is less than or equal to half full 0: STB is empty

24.2.3.2.6 CAN_RCTRL receive control register

Address: 0x4001 13 A3

Reset value: 0x0

Table24-11Receive control register CAN_RCTRL.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								SACK	ROM	ROV	RREL	RBALL		RSTAT	
								RW	RW	RW	RW	RW		RO	
								0	0	0	0	0		0	

Location	Bit name	Explain
[7]	SACK	Self-answering 1: When LBME = 1, the self-acknowledge function is enabled 0: No self-answer
[6]	ROM	Rx Buffer full, overflow control bit 1: Newly received data is not stored 0: The oldest received data is overwritten
[5]	ROV	Rx Buffer overflow bit 1: overflow, at least one frame of data is lost. Clear by writing RREL to 1

		0: No overflow
[4]	RREL	Release the Rx Buffer Slot. 1: The software has taken the current Rx Buffer Slot data, the current Rx Buffer Slot can be released, and the hardware points to the next Rx Buffer Slot 0: No release operation
[3]	RBALL	Receive Rx Buffer stores frames filtered by ID 1: Store all CAN frames, including error frames 0: Normal mode, only correct CAN frames are stored
[2]	--	Keep the default values
[1:0]	RSTAT	Rx Buffer status bit 3: Full (in overflow state, keep this value) 2: The number of frames that have been received by the Rx Buffer is greater than or equal to the value of AFWL, but is not full 1: The number of frames already received by the Rx Buffer is less than the value of AFWL 0: The number of frames already received by Rx Buffer is 0, null

24.2.3.2.7 CAN_RTIE Transmit Receive Interrupt Control Register

Address: 0x4001_13 A4

Reset value: 0x0

Table24-12Transmit Receive Interrupt Control Register CAN_RTIE.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[7]	RIE	Receive Interrupt Enable (Receive Interrupt Enable) 1: Enable 0: Forbidden
[6]	ROIE	Receive overrun interrupt enable (Receive Overrun Interrupt Enable) 1: Enable 0: Forbidden
[5]	RFIE	Rx Buffer Full Interrupt Enable (Rx Buffer Full Interrupt Enable) 1: Enable 0: Forbidden
[4]	RAFIE	Rx Buffer Full Interrupt Enable (Rx Buffer Almost Full Interrupt Enable) 1: Enable 0: Forbidden
[3]	TPIE	PTB transmit interrupt enable (Transmission Primary Interrupt Enable) 1: Enable 0: Forbidden
[2]	TSIE	STB transmit interrupt enable (Transmission Secondary Interrupt Enable) 1: Enable 0: Forbidden
[1]	EIE	Error Interrupt Enable (Error Interrupt Enable)

		1: Enable 0: Forbidden
[0]	TSFF	Tx Buffer flag bit 1: STB Slot is fully filled 0: STB Slot is not fully filled

24.2.3.2.8 CAN_RTIF Transmit Receive Interrupt Flag Register

Address: 0x4001_13A5

Reset value: 0x0

Table24-13Transmit Receive Interrupt Flag Register CAN_RTIF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
								RW	RW	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[7]	RIF	Receive interrupt flag bit (Receive Interrupt Flag) 1: When a valid frame (data frame or remote frame) is received, write 1 to clear. 0: No valid frame received
[6]	ROIF	Receive overflow interrupt flag bit (Receive Overrun Interrupt Flag) 1: At least one frame of unread data in Rx Buffer is overwritten 0: Rx Buffer No Overwrite In case of overflow, set ROIF and RFIF to 1 at the same time, and write 1 to clear.
[5]	RFIF	Rx Buffer Full Interrupt Flag (Rx Buffer Full Interrupt Flag) 1: Rx Buffer is full, write 1 to clear 0: Rx Buffer is not full
[4]	RAFIF	Rx Buffer Full Interrupt Flag Bit (Rx Buffer Almost Full Interrupt Flag) 1: The number of Rx Buffer Slot filled is greater than or equal to the AFWL setting value 0: Number of Rx Buffer Slot filled is less than AFWL setpoint
[3]	TPIF	PTB transmit interrupt flag (Transmission Primary Interrupt Flag) 1: Trigger PTB sending, and the sending is completed successfully. Write 1 to clear 0: No PTB sending request, no completion flag
[2]	TSIF	STB sends interrupt flag (Transmission Secondary Interrupt Flag) 1: The STB is triggered to transmit, and the transmission is completed successfully. Write 1 to clear 0: No STB sending request, no completion flag
[1]	EIF	Error Interrupt Flag (Error Interrupt Flag) 1: The value of the error counter changes to be greater than or less than the value set in the error warning register. Write 1 to clear 0: None An interrupt flag is triggered when the value of the error counter changes from less than the setting of the error warning register to greater than the setting, or from greater than to less than the setting. Also, entering or exiting from Bus Off triggers.

[0]	AIF	Cancel transmit interrupt flag (Abort Interrupt Flag) 1: Send message requested via TPA and TSA was successfully canceled. Write 1 to clear 0: Sending data was not canceled AIF does not have a corresponding enable register

Note: After the corresponding enable register is enabled, the interrupt flag bit will be set; AIF is the exception.

24.2.3.2.9 CAN_ERRINT Error Interrupt Enable and Flag Register

Address: 0x4001_13 A6

Reset value: 0x0

Table24-14Error Interrupt Enable and Flag Register CAN_ERRINT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
								RW	R	RW	RW	RW	RW	RW	RW
								0	0	0	0	0	0	0	0

Location	Bit name	Explain
[7]	EWARN	Error count value overrun interrupt flag bit 1: RECNT or TECNT is greater than or equal to EWL set value, write 1 to clear 0: RECNT or TECNT is less than EWL setting value
[6]	EPASS	CAN device in passive error state 1: CAN device is in passive error state 0: CAN device in active error state
[5]	EPIE	Passive Error Interrupt Enable (Error Passive Interrupt Enable) 1: Enable 0: Forbidden
[4]	EPIF	Passive Error Interrupt Flag (Error Passive Interrupt Flag) 1: Change from active error to passive error (or vice versa), write 1 to clear 0: Not occurred
[3]	ALIE	Arbitration failure interrupt enable (Arbitration Lost Interrupt Enable) 1: Enable 0: Forbidden
[2]	ALIF	Arbitration failure interrupt flag bit (Arbitration Lost Interrupt Flag) 1: arbitration failed, write 1 to clear 0: Arbitration succeeded
[1]	BEIE	Bus Error Interrupt Enable (Bus Error Interrupt Enable) 1: Enable 0: Forbidden
[0]	BEIF	Bus Error Interrupt Flag (Bus Error Interrupt Flag) 1: Bus error, write 1 to clear 0: No bus error

24.2.3.2.10 CAN_LIMIT Error & Warning Threshold Register

Address: 0x4001_13 A7

Reset value: 0x1B

Table24-15Error & Warning Threshold Register CAN_LIMIT.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								AFWL				EWL			
								RW				RW			
								0x1				0xB			

Location	Bit name	Explain
[7:4]	AFWL	Rx Buffer Almost Full Warning Configuration Value AFWL is compared with the quantity actually received by Rx Buffer. If the actual quantity exceeds AFWL, RAFIF is triggered. AFWL = 0, meaningless, hardware is forced to be configured as 1; AFWL exceeds the actual capacity of Rx Buffer, which is meaningless. The hardware is forced to configure the actual capacity of Rx Buffer.
[3:0]	EWL	Programmable Error Warning Limit = (EWL + 1) * 8. Possible limits: 8, 16, ··· 128。 The value of EWL controls the EIF.

24.2.3.2.11 CAN_SBAUD Baud Rate Configuration Register

Address: 0x4001_13 A8

Reset value: 0x0

Table24-16Baud rate configuration register CAN_SBAUD.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_PRESC									S_SJW						
RW									RW						
0									0						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S_SEG_2							S_SEG_1							
	RW							RW							
	0							0							

Location	Bit name	Explain
[31:24]	S_PRESC	TQ prescaler setting value (S_Prescaler) is mainly used to configure the size of TQ. TQ = system clock period *(S_PRESC+1)
[23]	--	Keep the default values
[22:16]	S_SJW	Resynchronization compensation width time setting (Bit Timing Segment 2)

		Resynchronization compensation width time = (S_SJW + 1) * TQ
[15]	--	Keep the default values
[14:08]	S_SEG_2	Segment 2 Time Unit Setting (Bit Timing Segment 2) Segment 2 Time = (S_SEG_2 + 1) * TQ
[07:00]	S_SEG_1	Bit Timing Segment 1 Segment 1 Time = (S_SEG_2 + 2) * TQ

24.2.3.2.12 CAN_EALCAP Error Message and Missing Arbitration Message Record Register

Address: 0x4001_13B0

Reset value: 0x0

Table24-17Error Information and Missing Arbitration Information Record Register CAN_EALCAP.

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

Location	Bit name	Explain
[7:5]	KOER	Kind Of Error 0: No error 1: Bit error 2: Formal error 3: Fill error 4: response error 5: CRC error 6: Other errors 7: Reserved The KOER bit is updated when there is an error, and the KOER bit remains unchanged during normal transmission and reception.
[4:0]	ALC	Arbitration Lost Capture When the arbitration fails, ALC records the position of the frame data when the arbitration fails.

24.2.3.2.13 CAN_RECNT receive error counter register

Address: 0x4001 13B2

Reset value: 0x0

Table24-18Receive error counter register CAN_RECNT.

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

Location	Bit name	Explain
[7:0]	RECNT	Receive Error Count The receive error counter is incremented or decremented according to the error count specified by the CAN protocol. This counter does not overflow. 255 is the maximum

24.2.3.2.14 CAN_TECNT transmit error counter register

Address: 0x4001_13B3

Reset value: 0x0

Table24-19Transmit error counter register CAN_TECNT.

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

Location	Bit name	Explain
[7:0]	TECNT	Send Error Counter (Transmit Error Count) The transmit error counter is incremented or decremented according to the error count specified by the CAN protocol. This counter does not overflow. 255 is the maximum

24.2.3.2.15 CAN_ACFCTRL ID Filter Control Register

Address: 0x4001_13B4

Reset value: 0x0

Table24-20 ID Filter Control Register CAN_ACFCTRL.

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

Location	Bit name	Explain

[7:6]	--	Keep the default values
[5]	SELMASK	Select mask register for ID filter (Select Acceptance MASK) 1: ACF points to ID filter AMR register (MASK) 0: ACF points to ID filter ACR register Select a specific set of filter registers via ACFADR
[4]	--	Keep the default values
[3:0]	ACFADR	ID Filter Address (Acceptance Filter Address) ACFADR points to a specific ID filter to distinguish between ACR and AMR through SELMASK. 0: Point to ACF 0 1: Point to ACF_1 2: Point to ACF_2 3: Point to ACF_3 Other values, invalid

24.2.3.2.16 CAN_ACFEN ID Filter Enable Register

Address: 0x4001_13B6

Reset value: 0x0

Table24-21 ID Filter Enable Register CAN_ACFEN.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												AE_3	AE_2	AE_1	AE_0
												RW	RW	RW	RW
												0	0	0	0

Location	Bit name	Explain
[31:8]		Not used
[3]	AE_3	ACF_3 Enable (Acceptance Filter 4 Enable) 1: Enable 0: Forbidden
[2]	AE_2	ACF_2 Enable (Acceptance Filter 3 Enable) 1: Enable 0: Forbidden
[1]	AE_1	ACF_1 Enable (Acceptance Filter 2 Enable) 1: Enable 0: Forbidden
[0]	AE_0	ACF 0 Enable (Acceptance Filter 1 Enable) 1: Enable 0: Forbidden

24.2.3.2.17 CAN_ACF ID filter select register

Address: 0x4001_13B8

Reset value: 0x0

Table24-22 ID Filter Select Register CAN_ACF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IDCMP	IDMASK	ID ACR [28:16] or AMR [28:16]												
	RW	RW	RW												
	0	0	0												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID ACR [15:00] or AMR [15:00]															
RW															
0															

Location	Bit name	Explain
[31]	--	Keep the default values
[30]	IDCMP	Valid when SELMASK = 1, ID AMR (MASK) selects the scope 1: that MASK receive by the ID filter is proces according to a standard frame format or an extended frame format. 0: The ID filter receives two formats of MASK compatible standard frame format and extended frame format.
[29]	IDMASK	When the IDCMP bit is 1, MASK selects which frame format 1: ID filter receives only extended frame format 0: ID filter receives only standard frame format
[28:00]	ID CODE/MASK	ACFADR selects which set of ID filters SELMASK = 0 indicates that ID ACR is selected for the ID filter SELMASK = 1 indicates that AMR is selected for the ID filter The range of ID ACR/AMR is Bit10-Bit0 in the standard frame format and Bit28-Bit0 in the extended frame format

25 Version history

Table25-1Document version history

Date	Version No.	Description
2025.07.31	1.33	07x oversampling setting notes
2025.07.31	1.32	Revised IO Driver Module Data Flow Diagram Internal Register Description
2025.07.30	1.31	Revise FAIL signal description and add FAIL signal distribution table.
2025.05.12	1.30	Add the description of no reset event when PVD undervoltage
2025.04.16	1.29	Modify the position of ROM_SIZE to [3:2]
2025.03.21	1.28	Delete 10k resistance value
2025.01.25	1.27	Update SIF SIF0_TOSC register description
2025.01.02	1.26	Update the FLASH NVR address for OPA Offset
2024.12.17	1.25	Add the FLASH NVR address for OPA Offset
2024.12.02	1.24	Add description of the time required for ADC IF to trigger
2024.9.29	1.23	The CLU input multiple check section is supplemented
2024.08.21	1.22	DSP, temperature sensor part errata
2024.08.17	1.21	Modify IT_OPA description
2024.08.08	1.20	ADC threshold monitoring function Corrigendum
2024.06.17	1.19	Add NVR calibration parameter address information
2024.06.04	1.18	Modified the description of PLL clock accuracy and deleted the description of BSIZE register
2024.05.29	1.17	Modify IIC module's decription of transmit and receive flow
2024.05.09	1.16	Modify DMA SIF request, ADC idle trigger, GPIO open-drain SYS_AFE_REG6.PVDSEL description.
2024.01.19	1.15	Modify the IIC's description of clock frequency division, added the description of sleep wake, added the ADC module channel number description
2023.11.20	1.14	Add a range description for DAC Gain and a single-ended output mode description for OPA
2023.09.25	1.13	Added notes on IWDG feeding dogs clock issues and GPIO PDI
2023.09.11	1.12	Remove the description of OPAHFLF EN, add the description of external benchmark time range gear for ADC, and add ESD diode to GPIO function block diagram
2023.08.10	1.11	Modify errors in the DAC calibration address and GPIO sections
2023.07.24	1.1	Modified the reset value of the FLASH CFG configuration register
2023.07.05	1.09	Minor revision, modify CMP_FT description
2023.07.04	1.08	The bit5 in SYS_AFE_REG2 is modified to the reserved bit,modify power supply range
2023.05.07	1.07	Modified the description of PWM Filter clock frequency division description
2023.04.20	1.06	Add power detection symbol (PWR_WEAK) description
2023.04.07	1.05	Add OPAHFLF_EN description
2023.02.18	1.04	Modified the description of MCPWM_SDCFG
2023.02.15	1.03	Add some descrition for TIMER external event
2023.02.10	1.02	Add the description of PLLPDN, BGPPD, BGPPD
2023.02.06	1.01	Added description of SYS_FLSE/SYS_FLSP register
2022.10.16	1.0	The official version is released
2022.01.02	0.9	Update the description of the CAN module, split the CMP_TCLK filter configuration, and add the MCPWM ZCS_IF.
2021.12.29	0.8	Added description of CAN module
2021.12.22	0.7	Added description of CL0 module

2021.12.17	0.6	Update SYS_AFE_REG.
2021.12.14	0.5	Revise ADC channel, add description of GPIO and CL access for DSP
2021.12.11	0.4	Revise drawings in all chapters except SIF, SPI and I2C
2021.03.18	0.3	Modify the HRC/LRC clock frequency
2021.02.02	0.2	All sections modified except flash/I2C/SPI
2020.12.01	0.1	Initial version, the build containing the instructions associated with the test

Disclaimer

LKS and LKO are registered trademarks of Linko.

Linko tries its best to ensure the accuracy and reliability of this document, but reserves the right to change, correct, enhance, modify the product and/or document at any time without prior notice. Users can obtain the latest information before placing an order.

Customers should select the appropriate Linko product for their application needs and design, validate and test your application in detail to ensure that it meets the appropriate standards and any safety, security or other requirements. The customer is solely responsible for this.

Linko hereby acknowledges that no intellectual property licenses, express or implied, are granted to Linko or to third parties.

Resale of Linko products on terms other than those set forth herein shall void any warranty warranties made by Linko for such products.

For earlier versions, please refer to this document.