

DP32G030 Reference Manual

32-bit Microcontrollers

Version. 1.23



Action Dynamic Tech.(HK) Trading Co.

Shenzhen Dynamic Century Technology Co.

Address : Room 1111, Block B, Phase 6, International
Innovation Valley, Taishi 1st Road, Nanshan District,
Shenzhen, China Tel : 0755-83134419

Fax: 0755-82519160

Company website:

www.dnsj88.com

EMAIL:dnsj@dn-ic.com

Zip code: 518031

catalogs

Table of Contents	1
Figure Catalog	13
Table of Contents	18
Revised Record	19
1. Introduction	20
1.1 OVERVIEW	20
1.2 Product Characteristics	21
2. Selection Guide	26
3. Chip Structure Block Diagram	27
4. Pin Definitions	28
4.1 Package Type	28
4.2 Multiplexed Pins	30
4.3 Pin Multiplexing Function	36
5. Functional Description	39
5.1 Address Space Mapping	39
5.2 Memory Division and Privilege Control	41
5.3 Interrupt Vector Table	41
5.4 ARM Cortex-M0 Core	43
5.4.1 Overview	43
5.4.2 Characterization	43
5.4.3 System Timer (SYSTICK)	45
SYST_CTRL Status register (0x00)	47
SYST_LOAD Reload register (0x04)	47
SYST_VAL Current Value Register (0x08)	48
5.5 Power Management (PMU)	49
5.5.1 Overview	49
5.5.2 Characteristics	49
5.5.3 Module Structure Block Diagram	49
5.5.4 Functional Description	51
Power supply	51
reset (a dislocated joint)	52
Low Power Mode	55
Register Mapping	62
Register Description	63
LPOW_MD register (0x00)	63
LPMD_WKEN register (0x04)	64
LPMD_WKST register (0x08)	64
CHIP_RST_ST register (0x0C)	65
SRC_CFG register (0x10)	65
TRIM_POW0 register (0x20)	66
TRIM_POW1 register (0x24)	66

TRIM_POW2 register (0x28)	67
TRIM_POW3 register (0x2C)	67
TRIM_RCHF register (0x30)	67
TRIM_RCLF register (0x34)	68
TRIM_OPA register (0x38)	68
TRIM_PLL register (0x3C)	69
TRIM_LOCK register (0x80)	69
DATA_BAK0 register (0x100)	69
DATA_BAK1 register (0x104)	69
DATA_BAK2 register (0x108)	70
DATA_BAK3 register (0x10C)	70
5.6 System Control (SYSCON)	71
5.6.1 Overview	71
5.6.2 Characteristics	71
5.6.3 Module Structure Block Diagram	71
5.6.4 Functional Description	73
RCHF Clock	RCHF Clock
PLL Clock	75
XTAL Clock	76
RCLF Clock	77
System clock (sys_clk) selection	78
RTC Clock	78
Independent Watchdog (IWDT) Clock	79
Window Watchdog (WWDT) Clock	79
SARADC Sampling Clock	79
Low Power Mode Wake-Up Clock	79
128-bit Chip Unique Identifier	79
Register Mapping	80
Register Description	81
CLK_SEL register (0x00)	81
DIV_CLK_GATE register (0x04)	82
DEV_CLK_GATE register (0x08)	82
RC_FREQ_DELTA register (0x78)	84
VREF_VOLT_DELTA Register (0x7C)	84
CHIP_ID0 register (0x80)	85
CHIP_ID1 register (0x84)	85
CHIP_ID2 register (0x88)	85
CHIP_ID3 register (0x8C)	85
PLL_CTRL register (0x180)	85
PLL_ST register (0x184)	86
5.7 IO Function Configuration (PORTCON)	87
5.7.1 Overview	87
5.7.2 Characteristics	87
5.7.3 Module Structure Block Diagram	88

5.7.4 Functional Description	89
Register Mapping Register Mapping	94
Register Description	95
PORTA_SEL0 register (0x00)	95
PORTA_SEL1 register (0x04)	97
PORTB_SEL0 register (0x08)	99
PORTB_SEL1 register (0x0C)	100
PORTC_SEL0 register (0x10)	102
PORTA_IE register (0x100)	104
PORTB_IE register (0x104)	104
PORTC_IE register (0x108)	104
PORTA_PU register (0x200)	105
PORTB_PU register (0x204)	105
PORTC_PU register (0x208)	105
PORTA_PD register (0x300)	105
PORTB_PD register (0x304)	106
PORTC_PD register (0x308)	106
PORTA_OD register (0x400)	106
PORTB_OD register (0x404)	107
PORTC_OD register (0x408)	107
PORTA_WKE register (0x500)	107
PORTB_WKE register (0x504)	107
PORTC_WKE register (0x508)	108
PORT_CFG register (0x600)	108
PORTA_WK_SEL register (0x700)	109
PORTB_WK_SEL register (0x704)	109
PORTC_WK_SEL register (0x708)	109
5.8 General Purpose IO (GPIO)	110
5.8.1 Overview	110
5.8.2 Characteristics	110
5.8.3 Module Structure Block Diagram	111
5.8.4 Functional Description	111
Register Map GPIO DATA Register (0x00)	119
GPIO DIR register (0x04)	119
INTLVLTRG register (0x08)	119
INTBE register (0x0C)	119
INTRISEEN register (0x10)	120
INTEN register (0x14)	120
INTRAWSTAUS register (0x18)	120
INTSTAUS Register (0x1C)	121
INTCLR register (0x20)	121
5.9 Basic Timer (TIMERBASE)	122
5.9.1 Overview	122
5.9.2 Characteristics	122

5.9.3 Block Diagram of Module Structure.....	123
5.9.4 Functional Description	124
Register Mapping.....	129
Register Description	130
TIMERBASE_EN register (0x00)	130
TIMERBASE_DIV register (0x04)	130
TIMERBASE_IE register (0x10).....	130
TIMERBASE_IF register (0x14).....	131
HIGH_LOAD register (0x20).....	131
HIGH_CNT register (0x24).....	131
LOW_LOAD register (0x30)	131
LOW_CNT register (0x34).....	132
5.10 Advanced Timer (TIMERPLUS)	133
5.10.1 Overview	133
5.10.2 Characteristics	134
5.10.3 Module Structure Block Diagram.....	135
5.10.4 Functional Description	137
Counting Clock Source Selection and Prescaling.....	137
Timer Mode.....	138
Counting Mode.....	140
Input Capture Mode.....	142
HALL mode.....	144
disruptions.....	146
Operation Process.....	147
Register Mapping.....	158
Register Description	159
TIMERPLUS_EN register (0x00)	159
TIMERPLUS_DIV Register (0x04)	159
TIMERPLUS_CTR register (0x08).....	159
TIMERPLUS_IE register (0x10).....	161
TIMERPLUS_IF register (0x14)	162
TIMERPLUS_HIGH_LOAD Register (0x20).....	163
TIMERPLUS_HIGH_CNT Register (0x24)	163
TIMERPLUS_HIGH_CVAL register (0x28)	164
TIMERPLUS_LOW_LOAD register (0x30)	164
TIMERPLUS_LOW_CNT register (0x34)	164
TIMERPLUS_LOW_CVAL register (0x38)	164
HALL_VAL register (0x40)	165
5.11 Independent Watchdog Clock (IWDT)	165
5.11.1 OVERVIEW	165
5.11.2 Characteristics	166
5.11.3 Module Structure Block Diagram.....	167
5.11.4 Functional Description	167
Register Mapping.....	171

Register Description	171
IWDT_LOAD register (0x00)	171
IWDT_CTRL register (0x08)	172
IWDT_IF register (0x0C)	172
IWDT_FEED register (0x10)	172
5.12 Window Watchdog Clock (WWDT)	173
5.12.1 Overview	173
5.12.2 Characteristics	173
5.12.3 Module Structure Block Diagram	174
5.12.4 Functional Description	175
WWDT interrupt generation and dog feeding interval	175
WWDT Feed Dog and Reset Generation	176
Register Mapping	177
Register Description	177
WWDT_LOAD register (0x00)	177
WWDT_VALUE register (0x04)	177
WWDT_CTRL register (0x08)	178
WWDT_IF Register (0x0C)	178
WWDT_FEED register (0x10)	179
5.13 Basic Pulse Width Modulation Generator (PWMBASE)	180
5.13.1 Overview	180
5.13.2 Characteristics	180
5.13.3 Module Structure Block Diagram	181
5.13.4 Functional Description	183
Register Mapping	189
Register Description	189
PWMBASE_EN register (0x00)	189
PWMBASE_DIV register (0x04)	190
PWMBASE_CON register (0x08)	190
PWMBASE_PERIOD register (0x0C)	191
PWMBASE_INTEN register (0x10)	191
PWMBASE_IF register (0x14)	192
PWMBASE_CNT register (0x18)	192
PWMBASE_CH0_COMP register (0x20)	192
PWMBASE_CH1_COMP register (0x30)	192
PWMBASE_CH2_COMP register (0x40)	193
5.14 Advanced Pulse Width Modulation Generator (PWMPLUS)	194
5.14.1 Overview	194
5.14.2 Characteristics	194
5.14.3 Block Diagram of Module Structure	195
5.14.4 Functional Description	198
Register Description	221
PWMPLUS_CFG register (0x00)	221
PWMPLUS_GEN register (0x04)	222

PWMPLUS_CLKSRC register (0x08)	224
PWMPLUS_BRAKE_CFG register (0x0c).....	225
PWMPLUS_MASKLEV register (0x10).....	227
PWMPLUS_PERIOD Register (0x1C).....	228
PWMPLUS_CH0_COMP register (0x20).....	228
PWMPLUS_CH1_COMP register (0x24).....	229
PWMPLUS_CH2_COMP register (0x28).....	229
PWMPLUS_CH0_DT register (0x30)	229
PWMPLUS_CH1_DT register (0x34)	230
PWMPLUS_CH2_DT register (0x38)	230
PWMPLUS_TRIG_COMP register (0x40).....	230
PWMPLUS_TRIG_CFG register (0x44)	231
PWMPLUS_IE Register (0x60)	231
PWMPLUS_IF register (0x64)	232
PWMPLUS_SWLOAD register (0x84).....	234
PWMPLUS_MASK_EN register (0x88)	235
PWMPLUS_CNT_ST register (0xE0)	235
PWMPLUS_BRAKE_ST Register (0xE4)	236
5.15 Real Time Clock (RTC)	237
5.15.1 Overview	237
5.15.2 Characteristics.....	237
5.15.3 Module Structure Diagram.....	238
5.15.4 Functional Description	239
Count Clock Source Selection.....	239
Second marking function	239
Calendar function	240
Alarm Clock Function.....	241
Interrupt Function.....	241
Low Power Wake-Up Function.....	242
Operation Process.....	242
Register Mapping.....	244
Register Description	244
RTC_CFG register (0x00).....	244
RTC_IE register (0x04)	245
RTC_IF register (0x08)	246
RTC_PRE register (0x10).....	247
RTC_TR register (0x14)	247
RTC_DR register (0x18)	248
RTC_AR register (0x1C)	249
RTC_TSTR register (0x20)	250
RTC_TSDR register (0x24).....	251
RTC_CNT register (0x28)	251
RTC_VALID register (0x2C)	251
5.16 UART Controller (UART)	253

5.16.1 Overview	253
5.16.2 Characteristics	254
5.16.3 Module Structure Block Diagram.....	255
Functional Description	256
Receive timing and associated status signals.....	256
Transmit Timing and Related Status Signals.....	257
Baud Rate Calculation	257
Automatic Baud Rate Detection.....	258
Data Sampling.....	259
Hardware Autoflow Control.....	259
Receive Timeout Description	261
Interruptions	262
Register Mapping.....	263
Register Description	264
UART_CTRL register (0x00)	264
UART_BAUD register (0x04).....	265
UART_TDR register (0x08).....	266
UART_RDR register (0x0C)	266
UART_IE register (0x10)	266
UART_IF register (0x14)	267
UART_FIFO register (0x18)	269
UART_FC register (0x1C)	270
UART_RXTO register (0x20).....	271
5.17 SPI Bus Controller (SPI)	272
5.17.1 Overview	272
5.17.2 Characteristics	272
5.17.3 Module Structure Block Diagram.....	273
5.17.4 Functional Description	274
SPI Interface Timing	274
I/O Configuration	275
Data Transfer Configuration.....	277
Interrupt Generation.....	278
DMA Control.....	284
Operation Procedure.....	285
Register Mapping.....	286
Register Description	286
SPICR register (0x00)	286
SPIWDR register (0x04).....	288
SPIRDR register (0x08)	289
SPIIE register (0x10).....	289
SPIIF register (0x14)	289
SPIFIFO register (0x18)	290
5.18 IIC Controller (IIC)	292
5.18.1 Overview	292

5.18.2 Characteristics	293
5.18.3 Block Diagram of Module Structure	294
5.18.4 Functional Description	294
Introduction to Protocols	294
SCL and SDA	298
Interrupt Function	299
Operation Process	300
Register Mapping	312
Register Description	312
IIC_CCFG register (0x00)	312
IIC_CST register (0x04)	313
IIC_CTRANS register (0x08)	314
IIC_RXDATA register (0x0C)	315
IIC_TXDATA register (0x10)	315
IIC_IE register (0x14)	316
IIC_IF register (0x18)	316
IIC_MCTRL register (0x20)	318
IIC_MSOPC Register (0x24)	319
IIC_SCTRL register (0x30)	320
IIC_SADDR register (0x34)	321
5.19 Analog-to-digital converter (ADC)	323
5.19.1 Overview	323
5.19.2 Characterization	324
5.19.3 Block Diagram of Module Structure	326
5.19.4 Functional Description	327
Channel Selection	327
Sampling once vs. averaging multiple samples	327
Single vs. Continuous Sampling	328
Channel Storage with FIFO	333
Internal vs. External Sample Clock Methods	334
CPU Trigger vs. External Signal Trigger	334
Interruptions	335
Operating Procedures	336
Register Mapping	337
Register Description	339
ADC_CFG register (0x00)	339
ADC_START register (0x04)	341
ADC_IE register (0x08)	342
ADC_IF register (0x0C)	343
ADC_CHx_STAT registers	343
ADC_CHx_DATA registers	343
ADC_FIFO_STAT register (0xA0)	344
ADC_FIFO_DATA register (0xA4)	345
ADC_EXTTRIG_SEL register (0xB0)	345

ADC_CALIB_OFFSET register (0xF0).....	345
ADC_CALIB_KD register (0xF4)	346
5.20 Comparator (COMP).....	347
5.20.1 Overview	347
5.20.2 Characteristics.....	347
5.20.3 Module Structure Block Diagram.....	348
5.20.4 Functional Description	349
Comparator Input Pins and Internal Output Signals.....	349
hysteresis	349
Filtering.....	350
Interruptions	351
Register Mapping.....	351
Register Description	352
CMP_CFG register (0x120)	352
CMP_ST register (0x124)	353
5.21 Operational Amplifiers (OPAMP).....	354
5.21.1 Overview	354
5.21.2 Characteristics.....	354
5.21.3 Module Structure Diagram.....	355
Register Mapping.....	355
Register Description	356
OPA_CFG	356
5.22 FLASH Controller (FLASHCTRL)	357
5.22.1 Overview	357
5.22.2 Characteristics.....	357
5.22.3 Module Structure Block Diagram.....	358
5.22.4 Functional Description	359
Low Power Mode.....	359
Read Rate Configuration.....	359
NVR zone and MAIN zone selection	360
Operation Mode	360
FLASH Operating Address	360
Sector Size	361
START operation start control	361
Operating Locks	361
MASK Function	361
Erase Time and Program Time Configuration.....	362
FLASH Initialization Busy Flag	362
Controller Busy Flag.....	362
Programming Data Cache Register Empty Status Flag	362
FLASH Program Initialization Flow	363
FLASH Sector Erase Procedure	363
FLASH Single Word Programming Procedure.....	365
FLASH Multi-Word Continuous Programming Procedure	367

Register Mapping.....	369
Register Description	369
FLASH_CFG Register (0x00).....	369
FLASH_ADDR register (0x04)	370
FLASH_WDATA register (0x08).....	371
FLASH_START register (0x10).....	371
FLASH_ST Register (0x14).....	371
FLASH_LOCK register (0x18).....	372
FLASH_UNLOCK Register (0x1C)	372
FLASH_MASK register (0x20).....	372
FLASH_ERASETIME register (0x24)	373
FLASH_PROGTIME register (0x28)	373
5.23 Cyclic Redundancy Check (CRC)	375
5.23.1 Overview	375
5.23.2 Characteristics.....	375
5.23.3 Module Structure Diagram.....	376
5.23.4 Functional Description	377
CRC code generation rules	377
CRC Polynomials.....	378
Operation Process.....	378
Register Mapping.....	379
Register Description.....	379
CRC_CR register (0x00)	379
CRC_IV register (0x04)	380
CRC_DATAIN register (0x08)	381
CRC_DATAOUT register (0x0C)	381
5.24 DMA Controller (DMA)	382
5.24.1 Overview	382
5.24.2 Characteristics.....	382
5.24.3 Module Structure Schematic	383
5.24.4 Functional Description	384
DMA Processing Flow	384
DMA Arbiter Processing.....	385
DMA Channel Configuration Information	386
Source and Destination Address Data Format Description	388
Interrupt Description	390
DMA Request Mapping Relationships.....	390
DMA Workflow	393
Register Mapping.....	395
Register Description	396
DMA_CTR register (0x00)	396
DMA_INTEN register (0x04)	396
DMA_INTST register (0x08)	397
DMA_CHnCTR register (0x100 + 0x20*(n))	398

DMA_CHnMOD register (0x100 + 0x20*(n) + 0x04).....	399
DMA_CHnMSADDR register (0x100 + 0x20*(n) + 0x08)	401
DMA_CHnMDADDR register (0x100 + 0x20*(n) + 0x0C).....	401
DMA_CHn_ST register (0x100 + 0x20*(n) + 0x10)	401
5.25 AES Operational Unit (AES).....	402
5.25.1 Overview	402
5.25.2 Characteristics.....	403
5.25.3 Module Structure Block Diagram.....	403
5.25.4 Function Description.....	404
Electronic Code Book (ECB)	404
Cipher Block Link (CBC)	406
Counter mode (CTR).....	407
Data Mode	409
Operation Process.....	411
Register Mapping.....	414
Register Description.....	414
AES_CR register (0x00)	414
AES_SR register (0x04).....	416
AES_DINR register (0x08)	416
AES_DOUTR register (0x0C)	417
AES_KEYR0 register (0x10)	417
AES_KEYR1 register (0x14)	418
AES_KEYR2 register (0x18)	418
AES_KEYR3 register (0x1C)	418
AES_IVR0 register (0x20)	419
AES_IVR1 register (0x24)	419
AES_IVR2 register (0x28)	419
AES_IVR3 register (0x2C)	420
6. Electrical Characteristics	421
6.1 Test Conditions	421
6.2 Absolute Maximum Ratings	421
6.2.1 Voltage Characteristics.....	421
6.2.2 Current Characteristics	421
6.2.3 Temperature Characteristics	422
6.3 Working conditions	422
6.3.1 General working conditions	422
6.3.2 Power-up and power-down operating conditions.....	422
6.3.3 Reset and Power Control Module Features.....	422
6.3.4 Supply Current Characteristics	423
6.3.5 Embedded Reference Voltage.....	424
6.3.6 Low power mode wake-up time.....	424
6.3.7 Internal High Frequency Clock Source Characteristics	425
6.3.8 Internal Low Frequency Clock Source Characteristics	425
6.3.9 External High Frequency Clock Source Characteristics	425

6.3.10 External Low Frequency Clock Source Characteristics.....	426
6.3.11 PLL Characterization.....	426
6.3.12 FLASH Memory Characterization.....	426
6.3.13 EMC Characterization.....	427
6.3.14 IO Port Characterization.....	427
6.3.15 EXTRST Port Characteristics.....	428
6.3.16 ADC Characterization.....	429
6.3.17 ADC Accuracy	429
6.3.18 Temperature Sensor Characteristics.....	430
6.3.19 Operational Amplifier Characteristics.....	430
6.3.20 Comparator Characteristics	431

catalog of diagrams

Figure 5-1 Cortex-M0 Processor Functional Block Diagram	43
Figure 5-2 SysTick Module Structure	45
Figure 5-3 sysTick Count Timing Diagram.....	46
Figure 5-4 Power Module Block Diagram.....	50
Figure 5-5 Power-on Reset and Power-off Reset Waveform Schematics	53
Figure 5-6 External Pin Reset Waveform Schematic.....	54
Figure 5-7 SLEEP Mode Entry and Exit Schematics	59
Figure 5-8 DEEPSLEEP Mode Entry and Exit Schematic	61
Figure 5-9 STOP Mode Entry and Exit Schematics	62
Figure 5-10 Chip Clock Structure.....	72
Figure 5-11 Hardware Configuration Diagram for External High Frequency Crystals	
73	
Figure 5-12 External Clock Related Hardware Configuration.....	74
Figure 5-13 Hardware Configuration Diagram Related to External Low Frequency Crystals	
77	
Figure 5-14 PORTCON Module Architecture Block Diagram	88
Figure 5-15 IO Input Function Selection Diagram	89
Figure 5-16 IO Output Function Selection Diagram	90
Figure 5-17 Analog Signal Connection IO Schematic	91
Figure 5-18 Functional Schematic of IO Multiplexing to ana_signal_b Analog Signal	91
Figure 5-19 IO Push-Pull Output Mode Schematic.....	92
Figure 5-20 IO Open-Drain Output Mode Schematic.....	93
Figure 5-21 GPIO Module System Block Diagram	110
Figure 5-22 GPIO Module Block Diagram	Figure 5-22 GPIO
Module Block Diagram	
Figure 5-23 GPIO Output Timing Chart.....	112
Figure 5-24 GPIO High Level Interrupt Timing Chart	113
Figure 5-25 GPIO Low Level Trigger Interrupt Timing Chart	114
Figure 5-26 GPIO Rising Edge Triggered Interrupt Timing Chart	Figure 5-26 GPIO
Rising Edge Triggered Interrupt Sequence Diagram	
Figure 5-27 GPIO Falling Edge Triggered Interrupt Timing Chart.....	Figure 5-27 GPIO
Falling Edge Triggered Interrupt Sequence Diagram	
Figure 5-28 GPIO Double Edge Trigger Interrupt Timing Chart	117
Figure 5-29 GPIO Operation Flowchart.....	118
Figure 5-30 TIMERBASE Module System Block Diagram.....	122
Figure 5-31 TIMERBASE Module Structure Block Diagram	123
Figure 5-32 TIMERBASE High Counter Count Timing Chart	124
Figure 5-33 TIMERBASE Low Counter Timing Chart.....	125
Figure 5-34 TIMERBASE Counter Crossover Count Timing Chart.....	126
Figure 5-35 TIMERBASE Interrupt Flag and Interrupt Diagram	127
Figure 5-36 TIMERBASE Operation Flow	128

Figure 5-37 TIMERPLUS System Block Diagram.....	Reference Manual	133
Figure 5-38 TIMERPLUS Structure Block Diagram		136
Figure 5-39 TIMERPLUS Clock Source Selection Diagram		138
Figure 5-40 Counter Timing Diagram with TIMERPLUS pclk Crossover Clock.....		138
Figure 5-41 Counter Timing Diagram in TIMERPLUS Timing Mode.....		139

Figure 5-42 TIMERPLUS Count Mode Rising Edge Valid Timing Chart	140
Figure 5-43 TIMERPLUS Count Mode Falling Edge Valid Timing Chart	141
Figure 5-44 TIMERPLUS Count Mode Double Edge Valid Timing Diagram	142
Figure 5-45 Rising Edge Valid Timing Chart under TIMERPLUS Input Capture	142
Figure 5-46 TIMERPLUS Input Capture Lower Falling Edge Valid Timing Chart	143
Figure 5-47 TIMERPLUS Input Capture Lower Double-Edge Valid Timing Chart	144
Figure 5-48 TIMERPLUS Timing Diagram in HALL Mode	145
Figure 5-49 TIMERPLUS Interrupt Flags and Interrupt Schematics	146
Figure 5-50 TIMERPLUS Timer Mode Operation Flowchart	148
Figure 5-51 TIMERPLUS Counting Mode Operation Flowchart	151
Figure 5-52 TIMERPLUS Input Capture Mode Operation Flowchart	154
Figure 5-53 TIMERPLUS HALL Mode Operation Flowchart	157
Figure 5-54 IWDT System Block Diagram	166
Figure 5-55 IWDT Module Structure Block Diagram	167
Figure 5-56 IWDT Interrupt Generation Diagram	168
Figure 5-57 IWDT Feed Dog and Reset Before First Interrupt	168
Figure 5-58 IWDT Feed Dog and Reset Before Second Interrupt	169
Figure 5-59 IWDT Interrupt Generation Operation Flowchart	170
Figure 5-60 WWDT Module System Block Diagram	173
Figure 5-61 WWDT Module Structure Block Diagram	174
Figure 5-62 WWDT Interrupt Generation and Dog Feeding Interval	175
Figure 5-63 WWDT Dog Feed and Reset	176
Figure 5-64 PWMBASE Module System Block Diagram	180
Figure 5-65 PWMBASE Module Architecture Block Diagram	Figure 5-65 PWMBASE Module Block Diagram
Figure 5-66 PWMBASE Timing Diagram for Different Cycles and Flip Points	184
Figure 5-67 PWMBASE 1 Frequency Division Timing Diagram	184
Figure 5-68 PWMBASE 2 Frequency Division Timing Diagram	185
Figure 5-69 PWMBASE 6 Division Timing Chart	185
Figure 5-70 PWMBASE Output Enable Timing Chart	185
Figure 5-71 PWMBASE Output Flip-Flop Timing Chart	186
Figure 5-72 PWMBASE Output Interrupt Timing Chart	187
Figure 5-73 PWMBASE Operation Flowchart	188
Figure 5-74 PWMPLUS System Block Diagram	194
Figure 5-75 PWMPLUS Structure Block Diagram	197
Figure 5-76 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Single Output, Start Level 0	199
Figure 5-77 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Single Output, Start Level 1	200
Figure 5-78 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Cyclic Output, Start Level 0	201
Figure 5-79 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Cyclic Output, Start Level 1	201
Figure 5-80 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Single Output, Start Level 0	

Figure 5-81 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Single Output, Start Level 1	202
	202
Figure 5-82 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Cyclic Output, Start Level 0	203
Figure 5-83 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Cyclic Output, Start Level 1	203
Figure 5-84 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Single Output, Start Level 0	204
Figure 5-85 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Single Output, Start Level 1	205

Figure 5-86 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Cyclic Output, Start Level 0	205
Figure 5-87 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Cyclic Output, Start Level 0	206
Figure 5-88 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Single Output, Start Level 0	207
Figure 5-89 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Single Output, Start Level 1	207
Figure 5-90 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Cyclic Output, Start Level 0	208
Figure 5-91 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Cyclic Output, Start Level 1	208
Figure 5-92 PWMPLUS Complementary Output Timing Diagram with Deadband Insertion	209
Figure 5-93 PWMPLUS Deadband Length Greater Than Positive Pulse Less Than Negative Pulse Timing Chart	209
Figure 5-94 PWMPLUS Deadband Length Greater Than Negative Pulse Less Than Positive Pulse Timing Chart	210
Figure 5-95 PWMPLUS Edge Alignment Mode Brake Case Timing Chart	210
Figure 5-96 PWMPLUS Center Alignment Mode Brake Case Timing Diagram	211
Figure 5-97 PWMPLUS Edge Alignment Mode MASK Case Timing Chart	211
Figure 5-98 PWMPLUS Center Alignment Mode MASK Case Timing Chart	212
Figure 5-99 PWMPLUS Edge Alignment Mode, Count Up, Flip Point and Cycle Timing Diagram with Start Level 0	213
Figure 5-100 PWMPLUS Edge Alignment Mode, Count Up, Flip Point and Cycle Timing Diagram with Start Level 1	213
Figure 5-101 PWMPLUS Edge Aligned Mode, Count Down, Start Level 0 Flip Point and Cycle Timing Diagram	214
Figure 5-102 PWMPLUS Edge Aligned Mode, Count Down, Start Level 1 Flip Point and Cycle Timing Diagram	214
Figure 5-103 PWMPLUS Center Aligned Mode, Count Up, Flip Point and Cycle Timing with Start Level 0	215
Figure 5-104 PWMPLUS Center Aligned Mode, Count Up, Flip Point and Cycle Timing with Start Level 1	215
Figure 5-105 PWMPLUS Center Aligned Mode, Count Down, Flip Point and Cycle Timing with Start Level 0	216
Figure 5-106 PWMPLUS Center Aligned Mode, Count Down, Flip Point and Cycle Timing with Start Level 1	216
Figure 5-107 PWMPLUS Output Priority Relationship Diagram	217
Figure 5-108 PWMPLUS Brake Interrupt Timing Diagram	217
Figure 5-109 PWMPLUS Flip-Flop, Specific Trigger Points, Cycle Overflow Interrupt, and Trigger Signal	

DP32G030

Timing Diagram	Reference Manual	218
Figure 5-110 PWMPLUS Module Operation Flowchart.....		220
Figure 5-111 RTC Module System Block Diagram		237
Figure 5-112 RTC Module Architecture Block Diagram		238
Figure 5-113 RTCCLK Selection Schematic.....		239
Figure 5-114 RTC Interrupt Flags and Interrupt Schematics		241
Figure 5-115 UART Module System Block Diagram		253
Figure 5-116 UART Module Block Diagram		255
Figure 5-117 UART Receive Timing and Related Status Signal Diagrams		256
Figure 5-118 UART Transmit Timing and Related Status Signal Diagrams		257
Figure 5-119 UART Automatic Baud Rate Calculation Schematic		258
Figure 5-120 UART Data Sampling Schematic.....		259
Figure 5-121 UART Auto Flow Control Connection Diagram.....		260

Figure 5-122 UART Flow Control Receive Schematic.....	Reference Manual	260
Figure 5-123 UART Flow Control Transmission Schematic	Figure 5-123 UART Flow Control	
Transmission Schematic		
Figure 5-124 UART First Receive Timeout Interrupt.....		261
Figure 5-125 UART Second Receive Timeout Interrupt.....		262
Figure 5-126 UART Interrupt Flags and Interrupt Schematics.....		263
Figure 5-127 SPI Module System Block Diagram.....		272
Figure 5-128 SPI Module Architecture Block Diagram.....		273
Figure 5-129 SPI Data/Clock Timing Diagram (CPHA=0)		274
Figure 5-130 SPI Data/Clock Timing Diagram (CPHA=1)		275
Figure 5-131 SPI SSN Timing Diagram (CPHA=0)		275
Figure 5-132 SPI Master/SPI Slave Interconnection		277
Figure 5-133 SPI Interrupt Flags and Interrupt Schematics.....		278
Figure 5-134 SPI Receive FIFO Full Interrupt.....		279
Figure 5-135 SPI Receive FIFO Half Full Interrupt.....		281
Figure 5-136 SPI Send FIFO Half Full Interrupt.....		282
Figure 5-137 SPI Transmit FIFO Air Break.....		283
Figure 5-138 SPI Operation Flow.....		285
Figure 5-139 IIC Module Architecture Block Diagram.....		294
Figure 5-140 IIC Data Transfer Diagram.....		295
Figure 5-141 IIC Host Transmit with 7bit Address Addressing Slave Receive ...		296
Figure 5-142 IIC Master Reads Slave Immediately After First Byte		297
Figure 5-143 IIC Combination Format		297
Figure 5-144 IIC Host Transmit Addressing Slave Receive with 10-Bit Addresses		297
Figure 5-145 IIC Host Receives Slave Transmission with 10-Bit Address Addressing		298
Figure 5-146 IIC SCL Open-Drain Diagram		298
Figure 5-147 IIC Interrupt Flags and Interrupt Schematics.....		299
Figure 5-148 IIC Host Transmit Operation Flowchart.....		301
Figure 5-149 IIC Host Receive Operation Flowchart.....		303
Figure 5-150 IIC Slave Transmit Operation Flowchart.....		306
Figure 5-151 IIC Slave Receiving Operation Flowchart.....		309
Figure 5-152 SARADC System Block Diagram		324
Figure 5-153 SARADC Module Architecture Diagram.....		326
Figure 5-154 SARADC Channel Selection Schematic.....		327
Figure 5-155 SARADC Single Channel Single Sample Schematic		328
Figure 5-156 SARADC Single-Channel Continuous Sampling Schematic		329
Figure 5-157 SARADC Multi-Channel Single Sampling Schematic		330
Figure 5-158 SARADC Multi-Channel Continuous Sampling Schematic.....		332
Figure 5-159 SARADC External Trigger Selection.....		335
Figure 5-160 SARADC Interrupt Flag and Interrupt Signal Schematic		335
Figure 5-161 SARADC Operation Flowchart.....		336
Figure 5-162 Comparator Structure Block Diagram		348
Figure 5-163 Comparator Hysteresis Schematic.....		350
Figure 5-164 Schematic of the comparator's filter function for burr elimination		350

Figure 5-166 Operational Amplifier Block Diagram	Reference Manual
Figure 5-167 FLASH Controller Structure Diagram.....	358
Figure 5-168 Sector Erase Operation Flowchart.....	364
Figure 5-169 Single Word Programming Operation Flowchart.....	366
Figure 5-170 Multi-Word Continuous Programming Operation Flowchart	369
Figure 5-171 CRC Module System Block Diagram	375
Figure 5-172 CRC Module Architecture	376
Figure 5-173 CRC Code Composition Diagram	377
Figure 5-174 CRC Operation Flowchart.....	379
Figure 5-175 DMA Controller System Connection Diagram.....	382
Figure 5-176 DMA Module Structure Diagram.....	384
Figure 5-177 Data Format Configuration Table for DMA Source and Destination Addresses	
389	
Figure 5-178 DMA Interrupt Description Table.....	390
Figure 5-179 Peripheral Request Mapping for DMA Channels.....	390
Figure 5-180 DMA Source-Side Peripheral Request Mapping Relationships....	392
Figure 5-181 DMA Destination Side Peripheral Request Mapping Relationship Diagram	
393	
Figure 5-182 DMA Workflow Diagram.....	394
Figure 5-183 AES System Block Diagram.....	402
Figure 5-184 AES Module Architecture Block Diagram.....	404
Figure 5-185 Encryption Schematic of the AES Electronic Codebook Algorithm	405
Figure 5-186 Decryption Schematic for the AES Electronic Codebook Algorithm	405
Figure 5-187 Encryption Schematic for AES Cipher Block Linking Algorithm	406
Figure 5-188 Schematic diagram of the AES cipher block linking algorithm for decryption.	
407	
Figure 5-189 Encryption Schematic for AES Counter Mode Linking Algorithm	408
Figure 5-190 Decryption Schematic for AES Counter Mode Linking Algorithm	409
Figure 5-191 AES Bit Shift Schematic.....	411

table of contents

Table 2-1 32G030 Series MCU Selection Table	26
Table 4-1 Pin Definitions	30
Table 4-2 Pin Multiplexing Functions.....	36
Table 5-1 Address Space Mapping	39
Table 5-6 32G030 Interrupt Vector Table.....	41
Table 5-8 GPIO High Level Trigger Interrupt Timing Parameters	113
Table 5-9 GPIO Low Level Trigger Interrupt Timing Parameters.....	114
Table 5-10 GPIO Rising Edge Triggered Interrupt Timing Parameters.....	115
Table 5-11 GPIO Falling Edge Triggered Interrupt Timing Parameters	116
Table 1 Voltage Characteristics	421
Table 2 Current Characteristics.....	421
Table 3 Temperature Characteristics.....	422
Table 4 Table of Shared Working Conditions	422
Table 5 Power-up and power-down operating conditions	422
Table 6 Reset and Power Control Module Characteristics	422
Table 7 Supply Current Characteristics	423

revised record

releas es	dates	author	note
1.0	2021/11/20	Zhou Liye	first edition
1.1	2021/12/15	Zhou Liye	Optimize the instructions for use
1.2	2022/01/20	Yang Weijia	Further optimization of instructions for use and structural drawings
1.21	2022/02/16	Wu Wei Nan	Formatting Amendments Content Review
1.22	2022/02/21	Sun Haojun	Corrected formatting and updated CMP-related content
1.23	2022/02/21	Sun Haojun	Update the correct register table

1. Summary

1.1 summarize

Embedded with ARM Cortex M0 core, it can operate up to 72MHz, supports PLL, has built-in high-speed memory, abundant enhanced IO ports and various peripherals. The product includes up to 40 IOs, one (14-channel) 12-bit ADC, four 16-bit advanced timers with input capture and cycle pulse output, six 16-bit basic timers, one 20-bit standalone watchdog, one 7-bit windowed watchdog, six standalone basic PWM waveform generators, six standalone advanced PWM waveform generators, with support for deadband and complementary functions, 2 SPIs, 2 IICs, 3 UARTs, 1 standard RTC, 1 CRC, 3 comparators, 1 temperature sensor, 2 operational amplifiers, 1 128bit AES.

The operating voltage of this product series is 2.0V-3.6V and the operating temperature is -40°C-105°C. Multiple power-saving operating modes ensure the requirements of low-power applications.

This product is suitable for the following applications: Internet of Things, Smart Home, Industrial Control, Instrumentation, Wearable Devices, Small Household Appliances, Motor Control, Medical and Handheld Devices and other products.

1.2 Product Characteristics

- Kernel and System
 - 32-bit ARM Cortex M0 Processor Core
 - Maximum operating frequency 72MHz
 - 24-bit System Tick Timer
 - Integrated Nested Vector Interrupt Controller (NVIC) for up to 32 interrupts
 - Program burning via SWD interface
- memory (unit)
 - Built-in 64K bytes FLASH memory for program storage area
 - Built-in 16K bytes of RAM as data storage area
- Clock, Reset and Power Management
 - 2.0V-3.6V Supply Voltage
 - Power **On/Power** Off Reset (POR/PDR), Watchdog Reset, Off-chip Dedicated Pin Reset (EXTRST)
 - Built-in 4-32MHz high-frequency crystal oscillator driver
 - Built-in 32768Hz low frequency crystal oscillator driver
 - Built-in factory-tuned 48MHz high-frequency RC oscillator
 - Built-in factory-tuned low-frequency 32768Hz RC oscillator
 - Built-in PLL circuitry to support frequencies up to 72MHz
- low power

Supports SLEEP mode, DEEPSLEEP mode, and STOP mode, totaling three system low-power modes.
- SARADC
 - 14-channel 12bit SARADC
 - Sample rates up to 2.4M
 - For supply voltage, temperature and external signal sampling
 - Supports single-shot mode and continuous mode

- Supports hardware averaging, which can be configured as 1, 2, 4 or 8 times averaging.
 - Supports ADC clock configurability, selectable system clock 1, 2, 4, 8 divisions
 - Software triggering and hardware triggering optional
 - Supports multiple interrupts
-
- GPIO
 - Up to 40 IO ports
 - Configurable for the following modes: float input, pull-up input, pull-down input, push-pull output, open-drain output, analog
- IO
- Flexible interrupt configuration, configurable as level-triggered and edge-triggered, level-triggered for low and high levels, and edge-triggered for rising, falling, and double edges
 - Each GPIO supports a key wake-up function that can be configured as a rising edge wake-up and a falling edge wake-up.
 - Most IOs support 5V level input tolerance.
-
- CRC
 - Supports CRC operations on 8-, 16-, and 32-bit data.
 - Supports multiple data formats for input data and output CRC values.
 - CRC polynomials are supported:
 x^8+x^2+x+1 , $x^{16}+x^{12}+x^5+1$, $x^{16}+x^{15}+x^2+1$, $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$
-
- AES
 - 128bit AES, supports ECB, CBC and CTR modes
-
- DMA
 - Supports up to 4 channels
 - The source and destination devices can be configured to transmit data

widths of 8, 16, and 32 bits, respectively, and the source addresses must be aligned according to the data transmission width.

- Each channel can be configured to transmit up to 4096 data strokes. When each channel finishes transmitting one data entry, the channel releases the bus, and the circuit will re-judge the priority of each channel, with the channel with the highest priority transmitting first.
- Channels can be configured in incremental or fixed address mode
- Supports memory to memory, memory to peripheral, peripheral to memory
- Supports cyclic restart DMA
- Supports configurable priority for each channel, independent interrupts for each channel
- comparator
 - 3-way comparator
 - Rail to Rail
 - Different hysteresis voltages can be configured
 - Can be combined into a window comparator
- operational amplifier
 - 2 Operational Amplifiers
 - Rail to Rail
 - 3MHz bandwidth
 - Low Offset Voltage
 - Output pins can optionally be configured to ADC channels for direct acquisition
- communications interface
 - 2 SPI interfaces, configurable master-slave mode, programmable clock polarity and phase, configurable master mode rate up to 4 divisions of the system clock, configurable data transfer order, independent read/write

DP32G030

data registers, 8 levels of FIFO caching for receiving Reference Manual
respectively, DMA transfer capability

- 3 UART interfaces, support full-duplex mode, support 8/9-bit data format selection, configurable parity bit parity parity even parity constant 0 constant 1, support 1-bit stop bit, configurable transmit delay time, support transmit completion interrupt, receive completion interrupt, receive timeout interrupt, support automatic baud rate detection, hardware flow control, 8-level FIFO buffer mechanism, DMA transmission Function
- 2 IIC interfaces, support master-slave mode, support 3 modes: Standard-mode(100kbps) Fast-mode (400kbps) Fast-mode Plus (1Mbps) host mode support for clock synchronization, SCL Configurable clock duty cycle, slave address mask support, 7-bit and 10-bit address mode support
- timers
 - 6 16-bit basic timers, counting clock supports 1-65536 divider frequency, only support timing function
 - 4 16-bit advanced timers, counting clock supports 1-65536 divider frequency, timing, counting, input capture and cycle pulse output functions, support HALL function
 - 1 x 20-bit Standalone Watchdog Timer, 1 x 7-bit Window Watchdog Timer
 - 6 independent 16-bit basic PWM waveform generators, count clock supports 1-65536 divider frequency, support flip-flop interrupt and cycle overflow interrupt
 - 6 independent 16-bit advanced PWM waveform generators, count clock support 1-65536 frequency division, support deadband, complementary and brake functions, support for rising edge count or falling edge count, support for edge-aligned or center-aligned waveform outputs, support for idle level, count start level, and output level flip-flop can be configured, support for flip-flop interrupts, cycle overflow interrupts, and specific trigger point interrupts
 - 1 standard RTC with real-time clock, alarm and calendar functions and

automatic leap year resolution, outputs half-second, second, minute, hour, day and alarm interrupts synchronized with the RTC, and RTC counting clocks with a choice of two clock sources, RCLF and XTAL.

- UUID 128-bit Chip Unique Identifier
- matrix
 - Working temperature: -40°C-105°C

- Storage temperature: -50°C-150°C
- Humidity class: MSL3
- seal inside
- LQFP48, LQFP32, TSSOP20, etc.

2. Selection Guide

Table 2-1 32G030 Series MCU Selection Table

Part Number	Voltage (V)	FLASH (KB)	RAM (KB)	IO	TIMERB ASE	TIMERP LUS	PWMB ASE	PWMPL US	IWDT	ADC	UART	SPI	IIC	Package
DP32G030LQ48	2.0V-3.6V	64	16	40	3	2	2	2	1	1	3	2	2	LQFP48
DP32G030LQ32	2.0V-3.6V	64	16	26	3	2	2	2	1	1	3	2	2	LQFP32
DP32G030TS20	2.0V-3.6V	64	16	16	3	2	2	2	1	1	3	2	2	TSSOP20

3. Chip structure block diagram

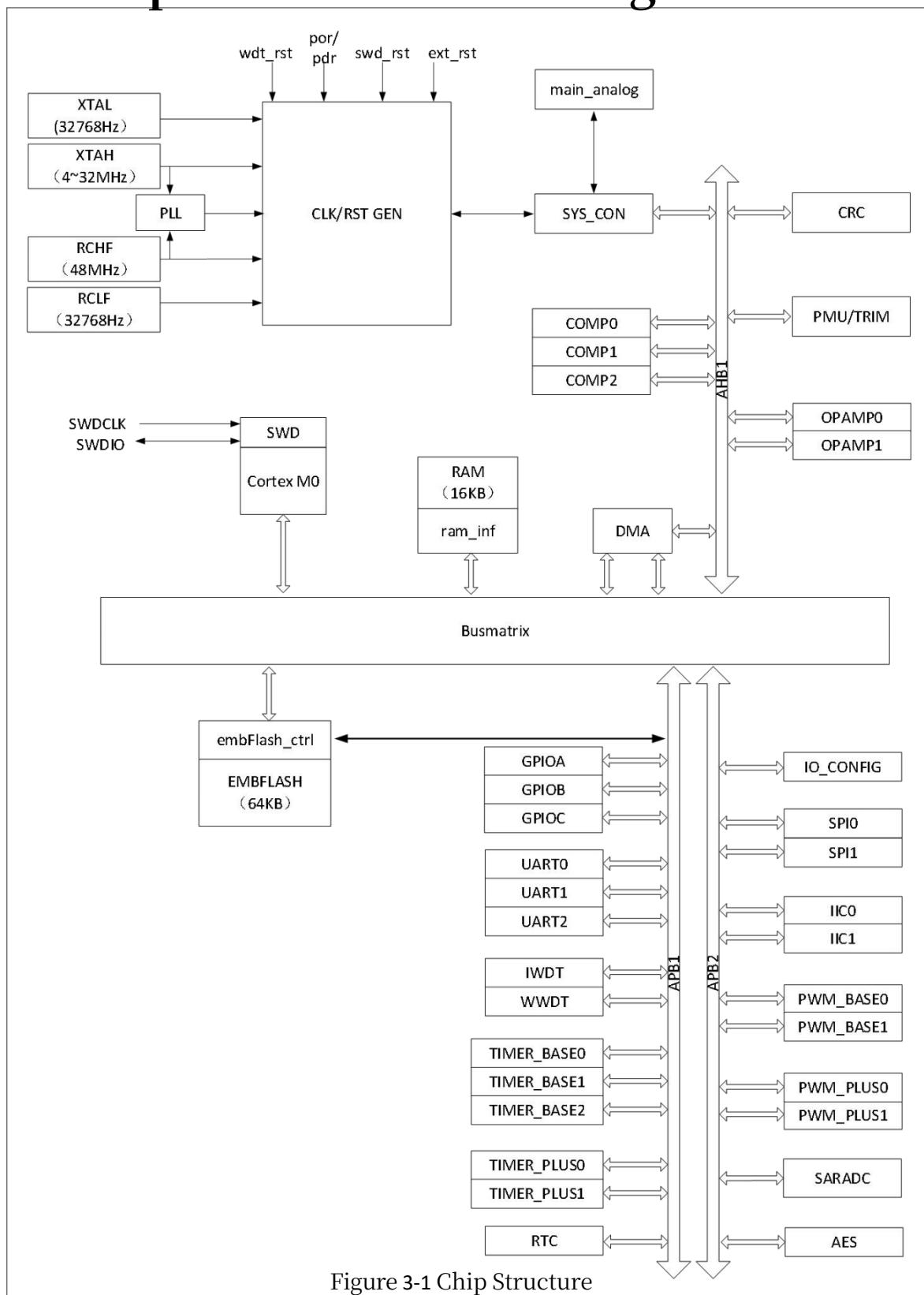
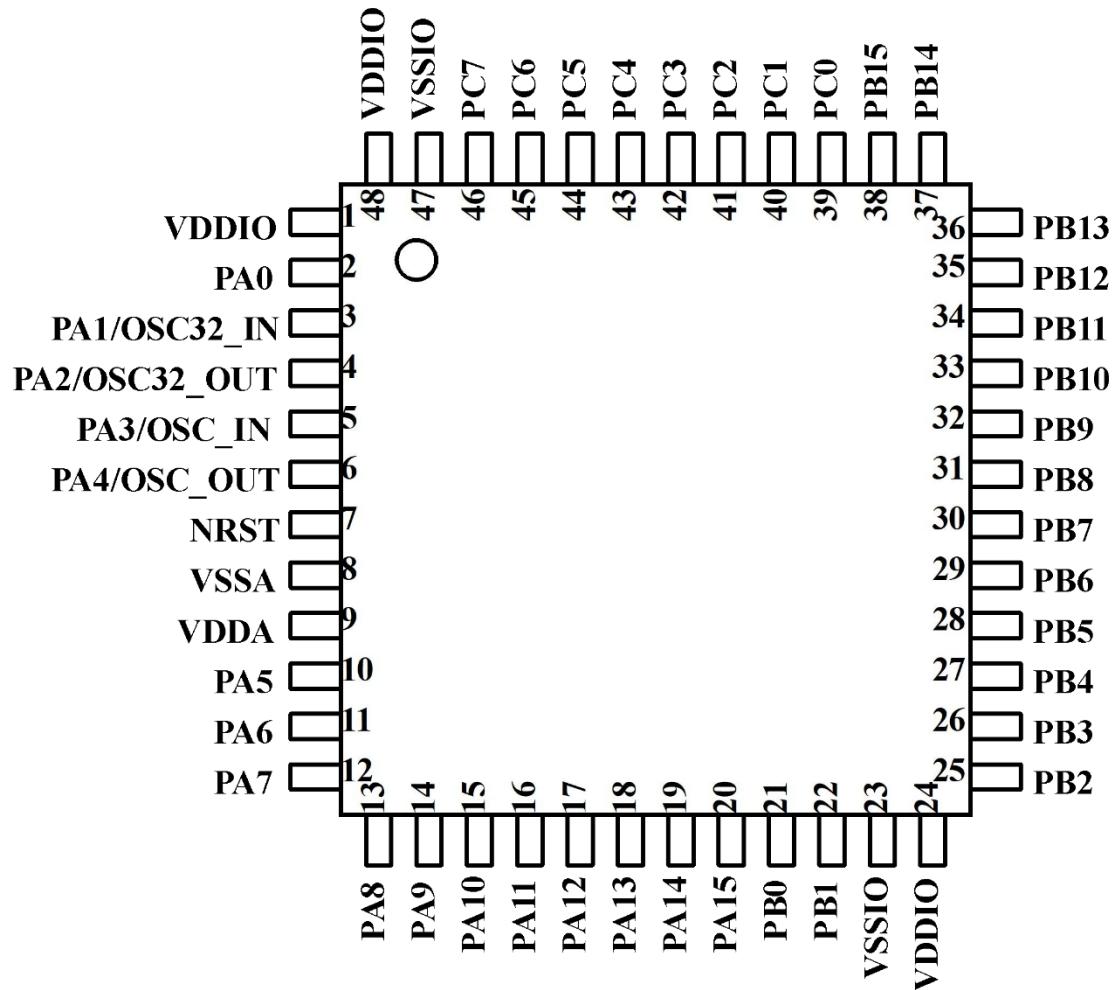


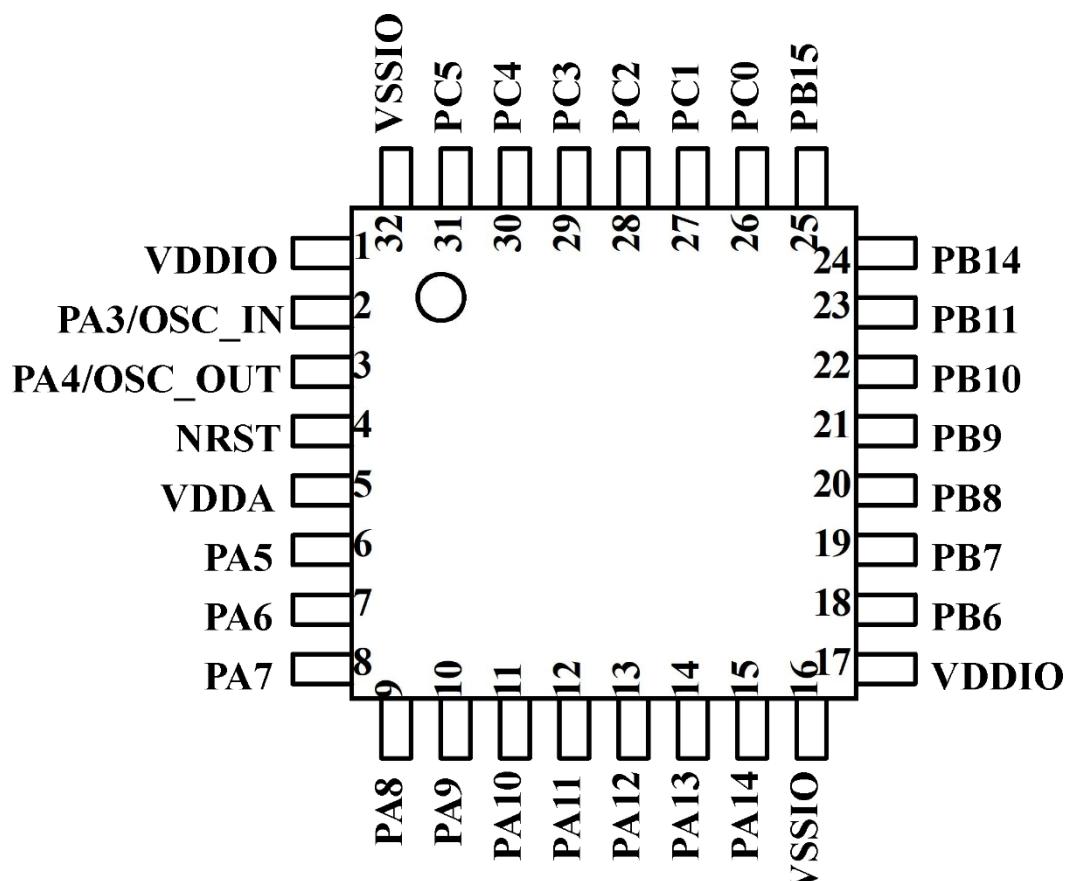
Figure 3-1 Chip Structure

4. Pin Definitions

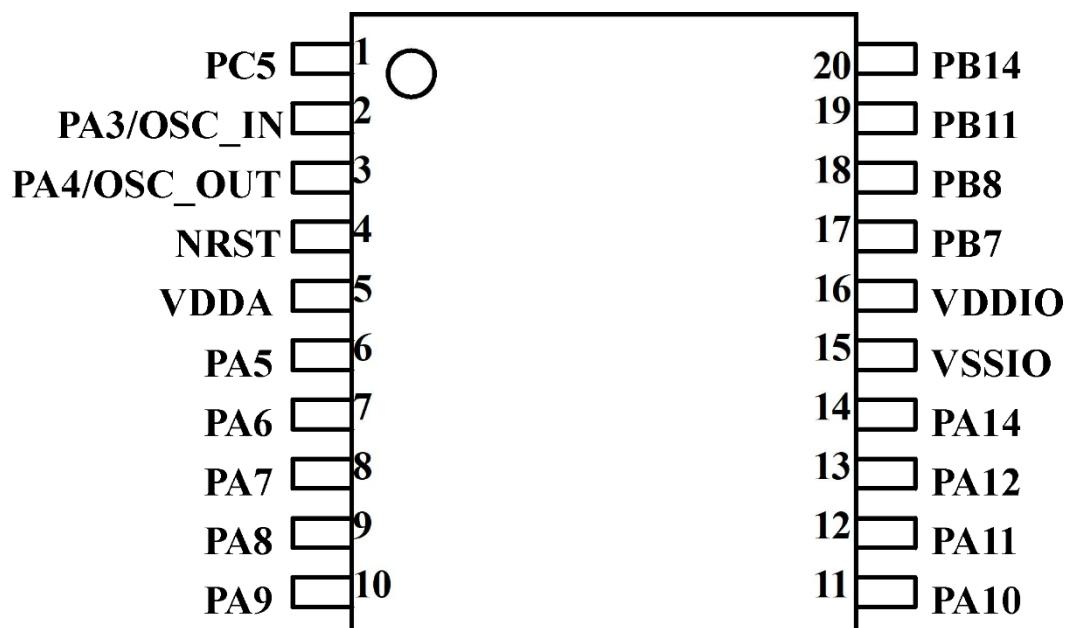
4.1 Package form



LQFP48(top view)



LQFP32(top view)



TSSOP20(top view)

4.2 multiplexed pin

Table 4-1 Pin Definitions

Pin Nu mb er LQ48	Pin Nu mb er LQ32	Pin Nu mb er TS20	Pin Defi niti ons	ty p ol o g y	multiplexing function	descriptive
1 24 48	1 17	16	VDDIO	S		2.0V~3.6 V power input pin
23 47	16 32	15	VSSIO	S		POWER GROUND
9	5	5	VDDA	S		2.3V~3.6 V power input pin
8			VSSA	S		POWER GROUND
7	4	4	NRST	I/O		Hardware Reset Pin
2		PA0	I/O	GPIO_PA0	PA0: Digital GPIO function pin	
				PWMP1_PLUS0	PWMP1_PLUS0: Advanced PWM1 input pulse pin 0	
				PWMP0_PLUS1	PWMP0_PLUS1: Advanced PWM0 input pulse pin 1	
				TM	TM: RTC 1/2 second marker output pin	
				WAKEUP0	WAKEUP0: Wake-up pin 0	
3		PA1	I/O	GPIO_PA1	PA1: Digital GPIO Function Pin	
				XTAL_XI	XTAL_XI: Input pin for low frequency crystal oscillator	
4		PA2	I/O	GPIO_PA2	PA2: Digital GPIO Function Pin	
				XTAL_XO	XTAL_XO: Output pin for low frequency crystal oscillator	
5	2	PA3	I/O	GPIO_PA3	PA3: Digital GPIO Function Pins	
				CMP0_VN	CMP0_VN: N input pin for comparator 0	
				XTAH_XI	XTAH_XI: Input pin for high-frequency crystal oscillator	
				GPIO_PA4	PA4: Digital GPIO Function Pins	

DP32G030

					CMP0_VP	CMP0_VP: P input pin for op amp 0
					XTAH_XO	XTAH_XO: Output pin for high frequency crystal oscillator
10	6	6	PA5	I/O	GPIO_PA5	PA5: Digital GPIO Function Pins
					UART1_CTS	UART1_CTS: CTS pin for serial port 1
					PWMP1_PLUS1	PWMP1_PLUS1: Advanced PWM1 input pulse pin 1
					TIMERP1_IN0	TIMERP1_IN0: Input 0 pin for advanced timer 1
					TIMERP1_OUT_L	TIMERP1_OUT_L: Advanced Timer 1 L output pin
					WAKEUP1	WAKEUP1: Wake-up pin 1
					SARADC_CH0	SARADC_CH0: Channel 0 pin of SARADC
11	7	7	PA6	I/O	GPIO_PA6	PA6: Digital GPIO Function Pins
					UART1 RTS	UART1 RTS: RTS pin for serial port 1
					TIMERP1_IN1	TIMERP1_IN1: Input 1 pin for Advanced Timer 1
					TIMERP1_OUT_H	TIMERP1_OUT_H: Advanced Timer 1 H output pin
					SARADC_CH1	SARADC_CH1: Channel 1 pin of SARADC
					OPA0_OUT	OPA0_OUT: Operational amplifier 0 output pin
12	8	8	PA7	I/O	GPIO_PA7	PA7: Digital GPIO Function Pins
					UART1_TX	UART1_TX: transmit pin of serial port 1
					TIMERP0_IN0	TIMERP0_IN0: Advanced Timer 0 Input 0 Pin
					TIMERP0_OUT_L	TIMERP0_OUT_L: Advanced Timer 0 L output pin
					SARADC_CH2	SARADC_CH2: Channel 2 pin of SARADC
					OPA0_VP	OPA0_VP: P input pin for op amp 0
13	9	9	PA8	I/O	GPIO_PA8	PA8: Digital GPIO Function Pin
					UART1_RX	UART1_RX: Receive pin for serial port 1
					TIMERP0_IN1	TIMERP0_IN1: Advanced Timer 0 input 1 pin
					TIMERP0_OUT_H	TIMERP0_OUT_H: Advanced Timer 0 H output pin
					SARADC_CH3	SARADC_CH3: Channel 3 pin of SARADC
					OPA0_VN	OPA0_VN: N input pin of op amp 0
14	10	10	PA9	I/O	GPIO_PA9	PA9: Digital GPIO Function Pin
					SPI0_SSN	SPI0_SSN: SPI0 chip select pin
					TIMERP1_IN0	TIMERP1_IN0: Input 0 pin for advanced timer 1
					TIMERP1_OUT_L	TIMERP1_OUT_L: Advanced Timer 1 L output pin
					TM	TM: RTC 1/2 second marker output pin
					SARADC_CH4	SARADC_CH4: Channel 4 pin of SARADC
					CMP1_VN	CMP1_VN: N input pin for comparator 1



Digital GPIO Function Pin Reference Manual					
15	11	11	PA10	I/O	GPIO_PA10
					SPI0_CLK
					SARADC_CH5
					CMP1_VP
16	12	12	PA11	I/O	GPIO_PA11
					SPI0_MISO
					PWMB0_CH0
					PWMP0_BRK0
					TIMERP1_IN1
					TIMERP1_OUT_H
					SARADC_CH6
17	13	13	PA12	I/O	GPIO_PA12
					SPI0_MOSI
					PWMB0_CH1
					PWMP0_CH0N
					TIMERP0_IN0
					TIMERP0_OUT_L
					SARADC_CH7
					GPIO_PA13
18	14		PA13	I/O	PWMB0_CH2
					PWMP0_CH1N
					TIMERP0_IN1
					TIMERP0_OUT_H
					SARADC_CH8
					GPIO_PA14
19	15	14	PA14	I/O	PWMB1_CH0
					PWMP0_CH2N
					TIMERP1_IN0
					TIMERP1_OUT_L
					SARADC_CH9
					GPIO_PA15
20			PA15	I/O	PWMB1_CH1
					PWMP0_CH0

					TIMERP1_IN1	TIMERP1_IN1: Reference to Advanced Timer 1
					TIMERP1_OUT_H	TIMERP1_OUT_H: Advanced Timer 1 H output pin
21		PB0	I/O	GPIO_PB0	PB0: Digital GPIO function pin	
				UART2_TX	UART2_TX: Transmit pin of serial port 2	
				IIC0_SCL	IIC0_SCL: Clock pin of IIC0	
				PWMB1_CH2	PWMB1_CH2: Channel 2 pin of basic PWM1	
				PWMP0_CH1	PWMP0_CH1: Channel 1 pin for advanced PWM0	
22		PB1	I/O	GPIO_PB1	PB1: Digital GPIO Function Pin	
				UART2_RX	UART2_RX: Receive pin for serial port 2	
				IIC0_SDA	IIC0_SDA: Data pin of IIC0	
				PWMP0_CH2	PWMP0_CH2: Channel 2 pin for advanced PWM0	
25		PB2	I/O	GPIO_PB2	PB2: Digital GPIO Function Pin	
				SPI1_SSN	SPI1_SSN: SPI1 chip select pin	
				PWMP0_BRK1	PWMP0_BRK1: BRAKE1 pin of advanced PWM0	
				TIMERP1_HALL0	TIMERP1_HALL0: HALL0 pin for advanced timer 1	
26		PB3	I/O	GPIO_PB3	PB3: Digital GPIO Function Pins	
				SPI1_CLK	SPI1_CLK: SPI1 clock pin	
				IIC1_SDA	IIC1_SDA: Data pin of IIC1	
				PWMP0_CH0N	PWMP0_CH0N: Channel 0N pin for advanced PWM0	
				TIMERP1_HALL1	TIMERP1_HALL1: HALL1 pin for advanced timer 1	
27		PB4	I/O	GPIO_PB4	PB4: Digital GPIO Function Pins	
				SPI1_MISO	SPI1_MISO: Host receive pin for SPI1	
				IIC1_SCL	IIC1_SCL: Clock pin of IIC1	
				PWMP1_CH0	PWMP1_CH0: Channel 0 pin for advanced PWM1	
				PWMP0_CH1N	PWMP0_CH1N: Channel 1N pin for advanced PWM0	
				TIMERP1_HALL2	TIMERP1_HALL2: Advanced Timer 1 HALL2 pin	
28		PB5	I/O	GPIO_PB5	PB5: Digital GPIO Function Pins	
				SPI1_MOSI	SPI1_MOSI: Host transmit pin for SPI1	
				PWMP1_CH0N	PWMP1_CH0N: Channel 0N pin for advanced PWM1	
				PWMP0_CH2N	PWMP0_CH2N: Channel 2N pin for advanced PWM0	
				TIMERP0_IN0	TIMERP0_IN0: Advanced Timer 0 Input 0 Pin	
				TIMERP0_OUT_L	TIMERP0_OUT_L: Advanced Timer 0 L output pin	

29	18	PB6	I/O	GPIO_PB6	PB6: Digital GPIO Function Pin
				PWMP0_CH0	PWMP0_CH0: Channel 0 pin for advanced PWM0
				TIMERP0_IN1	TIMERP0_IN1: Advanced Timer 0 input 1 pin
				TIMERP0_OUT_H	TIMERP0_OUT_H: Advanced Timer 0 H output pin
30	19	17	PB7	GPIO_PB7	PB7: Digital GPIO Function Pin
				SPI0_SSN	SPI0_SSN: SPI0 chip select pin
				UART0_TX	UART0_TX: transmit pin of serial port 0
				IIC0_SCL	IIC0_SCL: Clock pin of IIC0
				PWMP1_BRK0	PWMP1_BRK0: BRAKE0 pin for advanced PWM1
				PWMP0_CH1	PWMP1_BRK0: BRAKE0 pin for advanced PWM1
31	20	18	PB8	GPIO_PB8	PB8: Digital GPIO Function Pin
				SPI0_CLK	SPI0_CLK: SPI0 clock pin
				UART0_RX	UART0_RX: Receive pin for serial port 0
				IIC0_SDA	IIC0_SDA: Data pin of IIC0
				PWMB0_CH0	PWMB0_CH0: Channel 0 pin of basic PWM0
				PWMP1_BRK1	PWMP1_BRK1: BRAKE1 pin for advanced PWM1
				PWMP0_CH2	PWMP0_CH2: Channel 2 pin for advanced PWM0
32	21	PB9	I/O	GPIO_PB9	PB9: Digital GPIO Function Pin
				SPI0_MISO	SPI0_MISO: Host receive pin for SPI0
				UART0_CTS	UART0_CTS: CTS pin for serial port 0
				PWMB0_CH1	PWMB0_CH1: Channel 1 pin of basic PWM0
				PWMP1_CH0	PWMP1_CH0: Channel 0 pin for advanced PWM1
				TIMERP1_IN1	TIMERP1_IN1: Input 1 pin for Advanced Timer 1
				TIMERP1_OUT_H	TIMERP1_OUT_H: Advanced Timer 1 H output pin
33	22	PB10	I/O	GPIO_PB10	PB10: Digital GPIO Function Pin
				SPI0_MOSI	SPI0_MOSI: Host transmit pin for SPI0
				UART0_RTS	UART0_RTS: RTS pin for serial port 1
				PWMB0_CH2	PWMB0_CH2: Channel 2 pin of basic PWM0
				PWMP1_CH1	PWMP1_CH1: Channel 1 pin for advanced PWM1
				PWMP0_PLUS0	PWMP0_PLUS0: Advanced PWM0 input pulse pin 0
				TIMERP1_IN0	TIMERP1_IN0: Input 0 pin for advanced timer 1
				TIMERP1_OUT_L	TIMERP1_OUT_L: Advanced Timer 1 L output pin



34	23	19	PB11	I/O	GPIO_PB11 SWDIO	PB11: Digital GPIO Function Pin SWDIO: Data pin of the chip's SW port
					PWMP1_CH2 PWMP0_BRK2	PWMP1_CH2: Channel 2 pin for advanced PWM1 PWMP0_BRK2: BRAKE2 pin of advanced PWM0
					GPIO_PB12 UART1_TX IIC1_SCL PWMP1_CH0N	GPIO_PB12: Digital GPIO Function Pin UART1_TX: transmit pin of serial port 1 IIC1_SCL: Clock pin of IIC1 PWMP1_CH0N: Channel 0N pin for advanced PWM1
35		PB12	I/O	GPIO_PB13 UART1_RX IIC1_SDA PWMP1_CH1N	GPIO_PB13 UART1_RX IIC1_SDA PWMP1_CH1N	PB13: Digital GPIO Function Pins UART1_RX: Receive pin for serial port 1 IIC1_SDA: Data pin of IIC1 PWMP1_CH1N: Channel 1N pin for advanced PWM1
				GPIO_PB14 SWCLK UART2_TX PWMP1_CH2N	GPIO_PB14 SWCLK UART2_TX PWMP1_CH2N	PB14: Digital GPIO Function Pin SWCLK: Clock pin on the SW port of the chip UART2_TX: transmit pin of serial port 1 PWMP1_CH2N: Channel 2N pin for advanced PWM1
				GPIO_PB15 SPI1_SS_N UART2_RX	GPIO_PB15 SPI1_SS_N UART2_RX	PB15: Digital GPIO Function Pin SPI1_SS_N: SPI1 chip select pin UART2_RX: Receive pin for serial port 2
				GPIO_PC0 SPI1_CLK UART2_CTS PWMB1_CH0	GPIO_PC0 SPI1_CLK UART2_CTS PWMB1_CH0	PC0: Digital GPIO function pin SPI1_CLK: SPI1 clock pin UART2_CTS: CTS pin for serial port 2 PWMB1_CH0: Channel 0 pin of basic PWM1
40	27	PC1	I/O	GPIO_PC1 SPI1_MISO UART2_RTS PWMB1_CH1 TIMERP0_IN0 TIMERP0_OUT_L	GPIO_PC1 SPI1_MISO UART2_RTS PWMB1_CH1 TIMERP0_IN0 TIMERP0_OUT_L	PC1: Digital GPIO Function Pin SPI1_MISO: Host receive pin for SPI1 UART2_RTS: RTS pin for serial port 2 PWMB1_CH1: Channel 1 pin of basic PWM1 TIMERP0_IN0: Advanced Timer 0 Input 0 Pin TIMERP0_OUT_L: Advanced Timer 0 L output pin
				GPIO_PC2 SPI1_MOSI PWMB1_CH2 PWMP1_BRK2 TIMERP0_IN1 TIMERP0_OUT_H	GPIO_PC2 SPI1_MOSI PWMB1_CH2 PWMP1_BRK2 TIMERP0_IN1 TIMERP0_OUT_H	PC2: Digital GPIO Function Pin SPI1_MOSI: Host transmit pin for SPI1 PWMB1_CH2: Channel 2 pin of basic PWM1 PWMP1_BRK2: BRAKE2 pin for advanced PWM1 TIMERP0_IN1: Advanced Timer 0 input 1 pin TIMERP0_OUT_H: Advanced Timer 0 H output pin
				GPIO_PC3	GPIO_PC3	PC3: Digital GPIO Function Pin



DP32G030

42	29		PC3	I/O	UART0_TX	UART0_TX: transmit pin port 0
					IIC0_SCL	IIC0_SCL: Clock pin of IIC0
					PWMP1_CH1N	PWMP1_CH1N: Channel 1N pin for advanced PWM1
					TIMERP0_HALL0	TIMERP0_HALL0: Advanced Timer 0 HALL0 pin
					CMP2_VN	CMP2_VN: N input pin for Comparator 2
43	30		PC4	I/O	GPIO_PC4	PC4: Digital GPIO Function Pin
					UART0_RX	UART0_RX: Receive pin for serial port 0
					IIC0_SDA	IIC0_SDA: Data pin of IIC0
					PWMP1_CH2N	PWMP1_CH2N: Channel 2N pin for advanced PWM1
					TIMERP0_HALL1	TIMERP0_HALL1: Advanced Timer 0 HALL1 pin
					CMP2_VP	CMP2_VP: P input pin for comparator 2
44	31	1	PC5	I/O	GPIO_PC5	PC5: Digital GPIO Function Pin
					TIMERP0_HALL2	TIMERP0_HALL2: HALL2 pin for Advanced Timer 0
					OPA1_VP	OPA1_VP: P input pin of op amp 1
45			PC6	I/O	GPIO_PC6	PC6: Digital GPIO Function Pin
					IIC1_SCL	IIC1_SCL: Clock pin of IIC1
					PWMP1_CH1	PWMP1_CH1: Channel 1 pin for advanced PWM1
					TIMERP1_IN1	TIMERP1_IN1: Input 1 pin for Advanced Timer 1
					TIMERP1_OUT_H	TIMERP1_OUT_H: Advanced Timer 1 H output pin
					OPA1_VN	OPA1_VN: N input pin of op amp 1
46			PC7	I/O	GPIO_PC7	PC7: Digital GPIO Function Pins
					IIC1_SDA	IIC1_SDA: Data pin of IIC1
					PWMP1_CH2	PWMP1_CH2: Channel 2 pin for advanced PWM1
					TIMERP1_IN0	TIMERP1_IN0: Input 0 pin for advanced timer 1
					TIMERP1_OUT_L	TIMERP1_OUT_L: Advanced Timer 1 L output pin
					OPA1_OUT	OPA1_OUT: Operational amplifier 1 output pin

4.3 Pin Multiplexing Function

Table 4-2 Pin Multiplexing Functions

LQ48	pino ut nam e (of a thing)	SEL000	SEL001	SEL010	SEL011	SEL100	SEL101	SEL110	SEL111
2	PA0	GPIOA0	PWMP1_PLUS0	PWMP0_PLUS1	TM	WAKEUP0		--	
3	PA1	GPIOA1	XTAL_XI				--		
4	PA2	GPIOA2	XTAL_XO				--		
5	PA3	GPIOA3	CMP0_VN	XTAH_XI	--	--	--		
6	PA4	GPIOA4	CMP0_VP	XTAH_XO	--	--	--		
10	PA5	GPIOA5	UART1_CTS	PWMP1_PLUS1	TIMERP1_IN0	TIMERP1_OUT_L	WAKEUP1	SARADC_CH0	
11	PA6	GPIOA6	UART1_RTS	TIMERP0_IN0	TIMERP1_OUT_H	SARADC_CH1	OPA0_OUT		
12	PA7	GPIOA7	UART1_TX	BREAK_IN1	TIMERP0_OUT_L	SARADC_CH2	OPA0_VP		

DP32G030

13	PA8	GPIOA8	UART1_RX	TIMERP0_IN1	TIMERP0_OUT_L	Reference Manual	SARADC_CH3	OPA0_VN	--	
14	PA9	GPIOA9	SPI0_SS_N	TIMERP1_IN0	TIMERP1_OUT_L	TM	SARADC_CH4	CMP1_VN		
15	PA10	GPIOA10	SPI0_CLK	SARADC_CH5	CMP1_VP			--		

16	PA11	GPIOA11	SPI0_MISO	PWMB0_CH0	PWMP0_BRAKE0	TIMERP1_IN1	TIMERP1_OUT_H	SARADC_CH6	
17	PA12	GPIOA12	SPI0_MOSI	PWMB0_CH1	PWMP0_CH0N	TIMERP0_IN0	TIMERP0_OUT_L	SARADC_CH7	
18	PA13	GPIOA13	PWMB0_CH2	PWMP0_CH1N	TIMERP0_IN1	TIMERP0_OUT_H	SARADC_CH8	--	
19	PA14	GPIOA14	PWMB1_CH0	PWMP0_CH2N	TIMERP1_IN0	TIMERP1_OUT_L	SARADC_CH9		
20	PA15	GPIOA15	PWMB1_CH1	PWMP0_CH0	TIMERP1_IN1	TIMERP1_OUT_H	--		
21	PB0	GPIOB0	UART2_TX	IIC0_SCL	PWMB1_CH2	PWMP0_CH1			
22	PB1	GPIOB1	UART2_RX	IIC0_SDA	PWMP0_CH2				
25	PB2	GPIOB2	SPI1_SSN	PWMP0_BRAKE1	TIMERP1_HALL0				
26	PB3	GPIOB3	SPI1_CLK	IIC1_SDA	PWMP0_CH0N	TIMERP1_HALL1			
27	PB4	GPIOB4	SPI1_MISO	IIC1_SCL	PWMP1_CH0	PWMP0_CH1N	TIMERP1_HALL2		
28	PB5	GPIOB5	SPI1_MOSI	PWMP1_CH0N	PWMP0_CH2N	TIMERP0_IN0	TIMERP0_OUT_L		
29	PB6	GPIOB6	PWMP0_CH0	TIMERP0_IN1	TIMERP0_OUT_H				
30	PB7	GPIOB7	SPI0_SSN	UART0_TX	IIC0_SCL	PWMP1_BRAKE0	PWMP0_CH1		
31	PB8	GPIOB8	SPI0_CLK	UART0_RX	IIC0_SDA	PWMB0_CH0	PWMP1_BRAKE1	PWMP0_CH2	
32	PB9	GPIOB9	SPI0_MISO	UART0_CTS	PWMB0_CH1	PWMP1_CH0	TIMERP1_IN1	TIMERP1_OUT_H	
33	PB10	GPIOB10	SPI0_MOSI	UART0_RTS	PWMB0_CH2	PWMP1_CH1	PWMP0_PLUS0	TIMERP1_IN0	TIMERP1_OUT_L
34	PB11	GPIOB11	SWDIO	PWMP1_CH2	PWMP0_BRAKE2				

35	PB12	GPIOB12	UART1_TX	IIC1_SCL	PWMP1_CH0N				
36	PB13	GPIOB13	UART1_RX	IIC1_SDA	PWMP1_CH1N				
37	PB14	GPIOB14	SWCLK	UART2_TX	PWMP1_CH2N				
38	PB15	GPIOB15	SPI1_SS_N	UART2_RX					
39	PC0	GPIOC0	SPI1_CLK	UART2_CTS	PWMB1_CH0				
40	PC1	GPIOC1	SPI1_MISO	UART2_RTS	PWMB1_CH1	TIMERP0_IN0	TIMERP0_OUT_L		
41	PC2	GPIOC2	SPI1_MOSI	PWMB1_CH2	PWMP1_BRAKE2	TIMERP0_IN1	TIMERP0_OUT_H		
42	PC3	GPIOC3	UART0_TX	IIC0_SCL	PWMP1_CH1N	TIMERP0_HALL0	CMP2_VN		
43	PC4	GPIOC4	UART0_RX	IIC0_SDA	PWMP1_CH2N	TIMERP0_HALL1	CMP2_VP		
44	PC5	GPIOC5	TIMERP0_HALL2	OPA1_VP					
45	PC6	GPIOC6	IIC1_SCL	PWMP1_CH1	TIMERP1_IN1	TIMERP1_OUT_H	OPA1_VN		
46	PC7	GPIOC7	IIC1_SDA	PWMP1_CH2	TIMERP1_IN0	TIMERP1_OUT_L	OPA1_OUT		

5. Functional Description

5.1 address space mapping

The 32G030 controller is a 32-bit general-purpose controller that provides 4G bytes of addressing space, as shown in the following table. The data format only supports the small end mode, and the specific register layout and operation description of each module are described in detail in the following sections.

Table 5-1 Address Space Mapping

originate	close	module (in software)
memory (unit)		
0x00000000	0x0000FFFF	FLASH Zone
0x20000000	0x20003FFF	RAM area
AHB Bus Peripheral		
0x40000000	0x400007FF	SYSCON
0x40000800	0x40000FFF	PMU
0x40001000	0x400017FF	DMA
0x40003000	0x400037FF	CRC
APB1 Bus Peripheral		
0x40060000	0x400607FF	GPIOA
0x40060800	0x40060FFF	GPIOB
0x40061000	0x400617FF	GPIOC
0x40064000	0x400647FF	TIMER_BASE0
0x40064800	0x40064FFF	TIMER_BASE1
0x40065000	0x400657FF	TIMER_BASE2
0x40067000	0x400677FF	TIMER_PLUS0
0x40067800	0x40067FFF	TIMER_PLUS1
0x40069000	0x400697FF	RTC
0x4006A000	0x4006A7FF	IWDT
0x4006A800	0x4006AFFF	WWDT
0x4006B000	0x4006B7FF	UART0

0x4006B800

0x4006BFFF

UART1

0x4006C000	0x4006C7FF	UART2	Reference Manual
0x4006F000	0x4006F7FF	FLASH_CTRL	
APB2 Bus Peripheral			
0x400B0000	0x400B07FF	PORTCON	
0x400B1000	0x400B17FF	PWM_BASE0	
0x400B1800	0x400B1FFF	PWM_BASE1	
0x400B4000	0x400B47FF	PWM_PLUS0	
0x400B4800	0x400B4FFF	PWM_PLUS1	
0x400B8000	0x400B87FF	SPI0	
0x400B8800	0x400B8FFF	SPI1	
0x400B9000	0x400B97FF	IIC0	
0x400B9800	0x400B9FFF	IIC1	
0x400BA000	0x400BA7FF	SARADC	
0x400BD000	0x400BD7FF	AES128	

5.2 Memory division and authority control

There are two types of memory areas: 64KB FLASH and 16KB RAM.

64KB FLASH is used as the program storage area with 2KB NVR area and 64KB MAIN area, which is mainly used for storing some of our company-specific data, such as factory code information, TRIM data, and product configuration information, etc. The MAIN area is used for storing the user's program and is completely open to the user's use.

All 16KB of RAM is used as data area.

5.3 interrupt vector table (computing)

The interrupt vector table is shown below:

Table 5-6 32G030 Interrupt Vector Table

source of interruption	peripheral interrupt
0	WWDT
1	IWDT
2	RTC
3	DMA
4	SARADC
5	TIMER_BASE0
6	TIMER_BASE1
7	TIMER_PLUS0
8	TIMER_PLUS1
9	PWM_BASE0
10	PWM_BASE1
11	PWM_PLUS0
12	PWM_PLUS1
13	UART0
14	UART1
15	UART2
16	SPI0

17	SPI1
18	IIC0
19	IIC1
20	CMP
21	TIMER_BASE2
22	GPIOA5
23	GPIOA6
24	GPIOA7
25	GPIOB0
26	GPIOB1
27	GPIOC0
28	GPIOC1
29	GPIOA
30	GPIOB
31	GPIOC

5.4 ARM Cortex-M0 core

5.4.1 summarize

The Cortex-M0 processor is a 32-bit multi-level configurable RISC processor. It has an AMBA AHB-Lite interface and a Nested Vector Interrupt Controller (NVIC) has optional hardware debugging, executes Thumb instructions, and is compatible with the rest of the Cortex-M family. The processor supports two modes of operation, Thread mode and Handler mode. When an exception occurs, the processor enters Handler mode. Exception returns can only occur in Handler mode. The processor enters Thread mode when a reset occurs, and the processor can also enter Thread mode when an exception returns. The following figure shows the functional modules of the processor core.

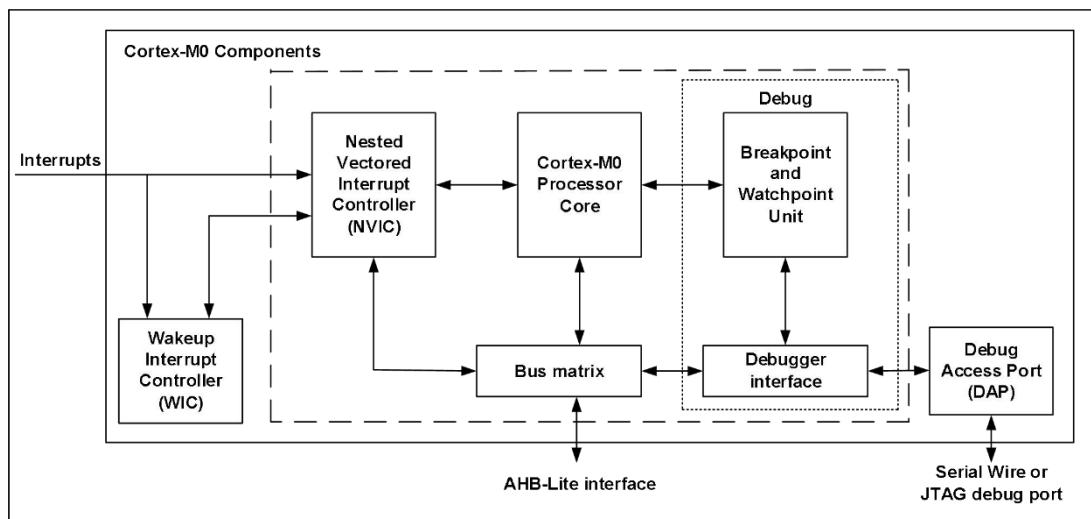


Figure 5-1 Cortex-M0 Processor Functional Block Diagram

5.4.2 characterization

- Processor Characteristics.
 - Thumb instruction set
 - Thumb-2 technology
 - 24-bit SYSTICK Timer

DP32G030
Reference Manual

- 32-bit Hardware Multiplier

- The system interface supports little-endian data access.
- Deterministic, fixed-latency interrupt handling capability
- Multiple load/store and multi-cycle multiply instructions can be discarded and restarted to ensure fast interrupt handling
- C Application Binary Interface Compatible Exception Mode (C-ABI)

ARMv6-M (C-ABI) Compatible Exception Mode allows users to implement interrupt handling using pure C functions
- Low-power hibernation mode can be entered by using the sleep-on-exit feature, which goes to sleep directly when the wait-for-interrupt (WFI) returns from an interrupt.
- NVIC Characteristics.
 - 32 external interrupt inputs, each with 4 levels of priority
 - Non-maskable interrupt input (NMI)
 - Supports level and pulse triggered interrupts
 - Wake on Interrupt Controller (WIC) supports very low power sleep mode
- adjust components during testing
 - Four hardware breakpoints
 - Two observation points
 - Program Count Sample Register (PCSR) for non-intrusive code
 - Single-step and vector capture capabilities
- Bus Interface.
 - Single 32-bit AMBA-3 AHB-Lite system interface provides easy integration of all system peripherals and memories
 - Single 32-bit slave with DAP (Debug Access Port) support

5.4.3 System timer (SYSTICK)

summarize

The Cortex-M0 core provides an internal 24-bit system timer. This timer loads the current value register when enabled.

(The value in the counter status register (`SYST_VAL`) is decremented down to 0 and the value in the reload register (`SYST_LOAD`) is reloaded on the next clock edge. When the counter decrements to 0 again, the identification bit `COUNTFLAG` in the counter status register (`SYST_CTRL`) is set and read to clear it.

After reset, both `SYST_VAL` register and `SYST_LOAD` register values are unknown, so it is necessary to initialize them before use by writing any value to `SYST_VAL`, clearing the zero and resetting the status register at the same time to ensure that the loaded value is the value in `SYST_LOAD` register. When the value of `SYST_LOAD` register is 0, the timer stays at 0 after reloading and stops reloading.

characterization

- 24-bit System Timer
- in descending order
- Write

Clear

Module

Block

Diagram

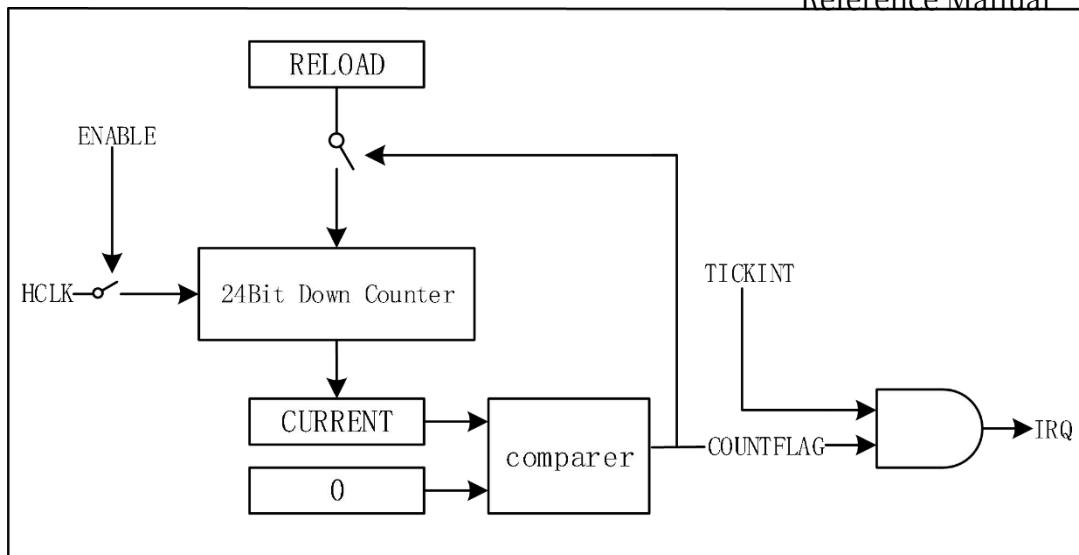


Figure 5-2 Systick Module Architecture Diagram

Functional Description

When this timer is enabled, the value in the current value register (**SYST_VAL**) is loaded and decremented down to 0, and the value in the reload register (**SYST_LOAD**) is reloaded on the next clock. When the counter decrements to 0 again, the flag bit **COUNTFLAG** in the counter status register (**SYST_CTRL**) is set and can be read to clear it.

After reset, both **SYST_VAL** register and **SYST_LOAD** register values are unknown, so it is necessary to initialize them before use by writing any value to **SYST_VAL**, clearing the zero and resetting the status register at the same time to ensure that the loaded value is the value in **SYST_LOAD** register. When the value of **SYST_LOAD** register is 0, the timer stays at 0 after reloading and stops reloading.

The counter can be used as a tick timer for real-time systems or as a simple counter.

The SysTick count timing diagram is shown below:

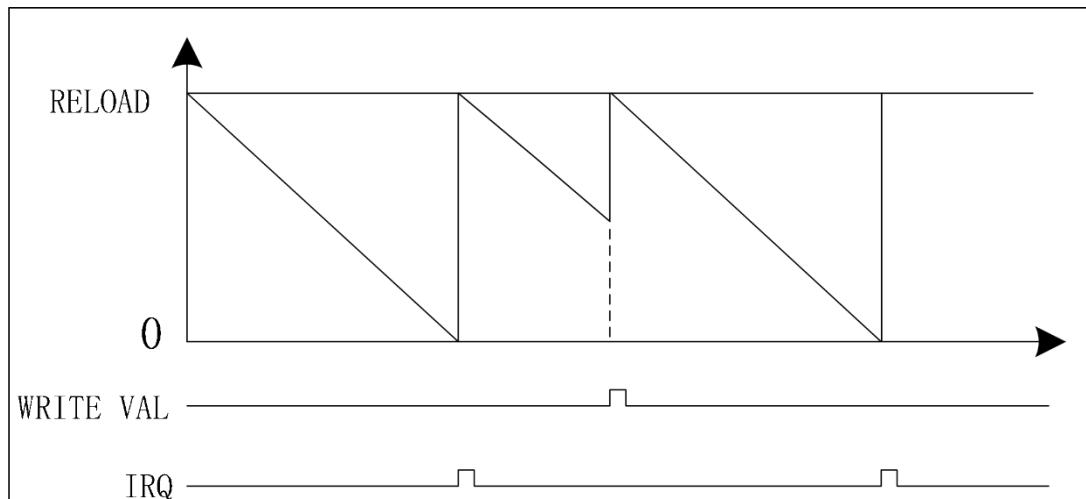


Figure 5-3 systick Count Timing Chart

register map

name (of a thing)	misalign ment	typology	reset value	descriptive

SYSTICK BASE:0xE000E010				
SYST_CTRL	0x00	R/W	0x00	status register
SYST_LOAD	0x04	R/W	0x00	Overloaded Registers
SYST_VAL	0x08	R/W	0x00	Current Value Register

name (of a thing)	misalignment	typology	reset value	descriptive
SYSTICK	BASE:0xE000E010			
SYST_CTRL	0x00	R/W	0x00	status register
SYST_LOAD	0x04	R/W	0x00	Overloaded Registers
SYST_VAL	0x08	R/W	0x00	Current Value Register

register description

SYST_CTRL Status register (0x00)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:17	RESERVED	R	0	reserved bit
16	COUNTFLAG	R	0	Counter counts to 0, this position 1
15:2	RESERVED	R	0	reserved bit
1	TINKINT	R/W	0	interrupt enable bit 0: Forbidden Energy 1: Enabling
0	ENABLE	R/W	0	Timer Enable Bit 0: Forbidden Energy 1: Enabling

SYST_LOAD Reload register (0x04)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:24	RESERVED	R	0	reserved bit
23:0	RELOAD	R/W	0	Counter counts to 0 Load the value of this register to restart counting

SYST_VAL Current Value Register (0x08)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:24	RESERVED	R	0	reserved bit
23:0	VAL	R/W	0	A read operation returns the current count value and a write operation clears the register and the COUNTFLAG flag bit.

5.5 Power Management (PMU)

5.5.1 summarize

The chip's supply voltage (AVDD) is 2.0-3.6 V. The internal digital circuits are supplied with a 1.2 V power supply through a built-in voltage regulator.

The internal A/D converter can be powered by an independent AVDD_ADC with a supply voltage of 2.0-3.6V, and an independent VREF reference voltage can be provided off-chip. For different packages, the A/D converter power supply may be double bonded to the main power supply (AVDD) on the same PIN, see the pinout of each package. The VREF reference voltage of the A/D converter may also be bonded to the same PIN as the AVDD for different packages, please refer to the pin description of each package.

The chip also provides a variety of ways to reduce power consumption, including: lowering the system clock frequency, peripheral clock control, and multiple low-power modes.

The chip contains watchdog (WDT) reset, pin reset, and CPU soft reset in addition to power-on reset and power-off reset.

5.5.2 characterization

- Operating voltage is 2.0V-3.6V
- ADC with separate power supply and reference supply
- Power-On Reset (POR) and Power-Down Reset (PDR)
- External power-on reset pin
- watchdog reset

Operating modes are: normal, WFI, sleep, and deep sleep.
(deepsleep) stop mode (stop)

5.5.3 Block Diagram of Module Structure

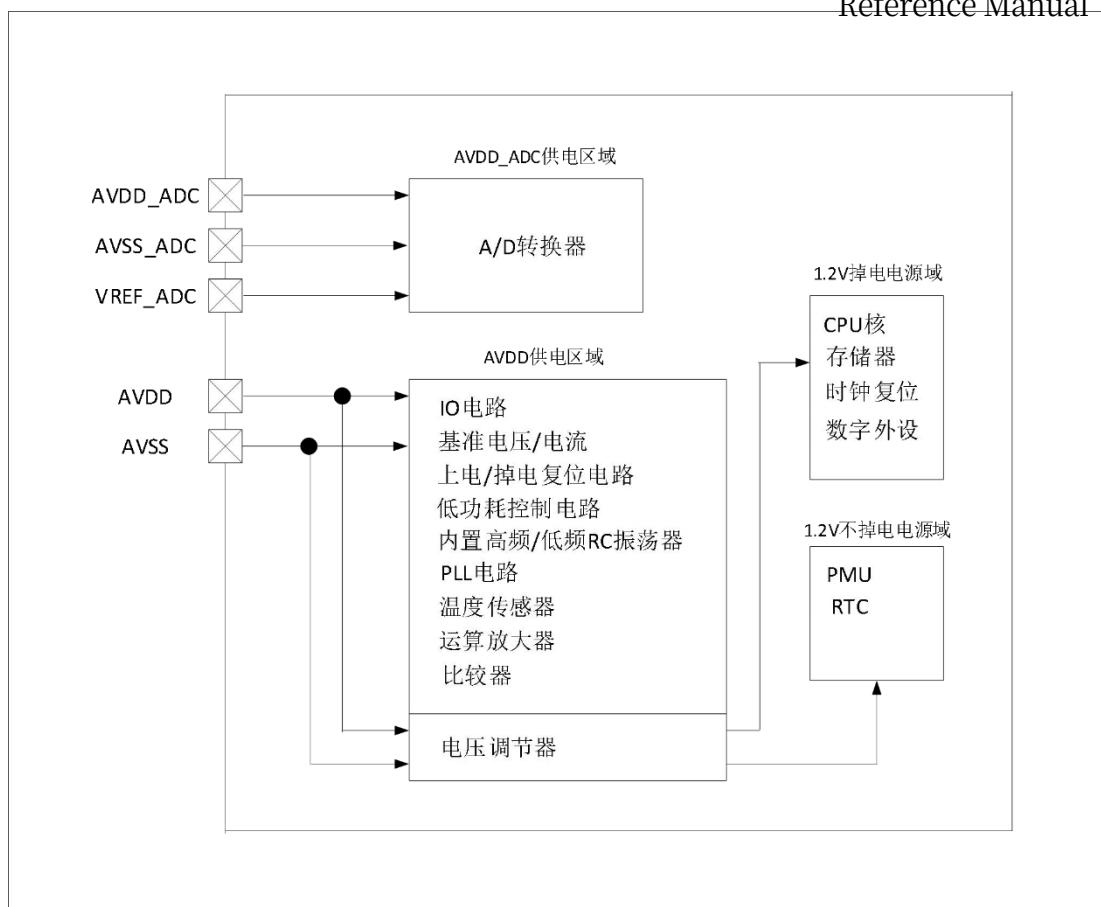


Figure 5-4 Power Module Structure Block Diagram

As shown in the figure above, there are two sets of power/grounds: one is the main power/ground (AVDD/AVSS) that powers the chip's analog circuitry, and the other is the power/ground (AVDD_ADC/AVSS_ADC) that powers the A/D converter.

Depending on the package, it is possible to connect AVDD and AVDD_ADC to the same PIN.

A/D converter with separate power supply (AVDD_ADC/AVSS_ADC) and separate external voltage reference inputs

PAD (VREF_ADC)

Most of the analog circuits inside the chip are supplied by AVDD, including: multiple voltage regulators, IO circuits, reference voltage/current, power-on/power-off reset circuits, low-power control circuits, built-in high-frequency/low-frequency RC oscillator circuits, PLLs, temperature sensors, operational amplifiers, and comparators.

The on-chip digital circuits all operate at 1.2V and are divided into power domains: the 1.2V power-down domain and the 1.2V non-dropout power domain, and the operating voltage can be adjusted by a voltage regulator to meet different low-power application scenarios.

The 1.2V power-down domain consists mainly of the CPU core, memory, clock reset control circuitry, and numerous digital peripherals, while the

The 1.2V non-dropout power domain mainly consists of PMU modules, RTC modules, and so on.

5.5.4 Function

n

Description

Power

The HM1030 chip operates from 2.0 V to 3.6 V. The built-in voltage regulator provides a 1.2 V supply to the internal digital circuitry. To improve conversion accuracy, the ADC utilizes a separate power supply and reference supply.

Separate A/D converter power supply and reference voltage

In order to improve the conversion accuracy of the ADC, the ADC uses a separate power supply and an off-chip reference voltage. The purpose of the separate power supply is to filter and shield interference and noise from the PCB board. The separate off-chip reference voltage is provided to provide a flexible reference voltage during use to meet more application scenarios.

The ADC power pin is

AVDD_ADC. The ADC ground

pin is AVSS_ADC.

The off-chip reference voltage pin for the ADC is VREF_ADC.

If VREF_ADC is connected to the same PIN as AVDD_ADC, the current reference voltage can be inverted by connecting a fixed level internally.

voltage regulator

The chip's internal voltage regulator is used to supply power to all digital circuits. It works in five different ways depending on the operating mode of the chip:

- 1) Normal Operation Mode: The voltage regulator provides 1.2V power to the digital circuits in normal power consumption mode, when all digital circuits can work normally;
- 2) WFI mode: Only the CPU core is stopped, other clocks, power supply, and all peripherals including the CPU core's peripherals such as NVIC, SysTick, etc. are still running in their original state.

- 3) SLEEP operation mode: The voltage regulator provides 1.2V power supply to the digital circuits in low-power mode to ensure that the digital circuits in the non-dropout power domain work normally, and the digital circuits in the dropout power domain save the current state.
- 4) DEEPSLEEP operation mode: The voltage regulator provides 1.2V power supply (this voltage can be further regulated by TRIM_LPLDO register to step down) to the digital circuits in a low-power mode to ensure that the digital circuits in the power domain do not drop out and operate normally.
- 5) STOP mode: The voltage regulator stops supplying power. All digital circuit contents are lost.

reset (a dislocated joint, an electronic device etc)

There are six reset sources: power-on reset, power-down reset, pin reset, watchdog reset and CPU soft reset, among which there are two kinds of watchdog reset: independent watchdog (IWDT) reset and window watchdog (WWDT) reset.

Power-On Reset (POR) and Power-Down Reset (PDR)

The chip consists of a complete set of power-on reset (POR) and power-down reset (PDR) circuits internally. When the chip supply voltage rises to a specific threshold voltage, the system can operate normally; when it falls below this specific threshold voltage, the chip maintains the reset state without the need for external reset circuitry.

The power-on reset and power-down reset waveforms are shown schematically below:

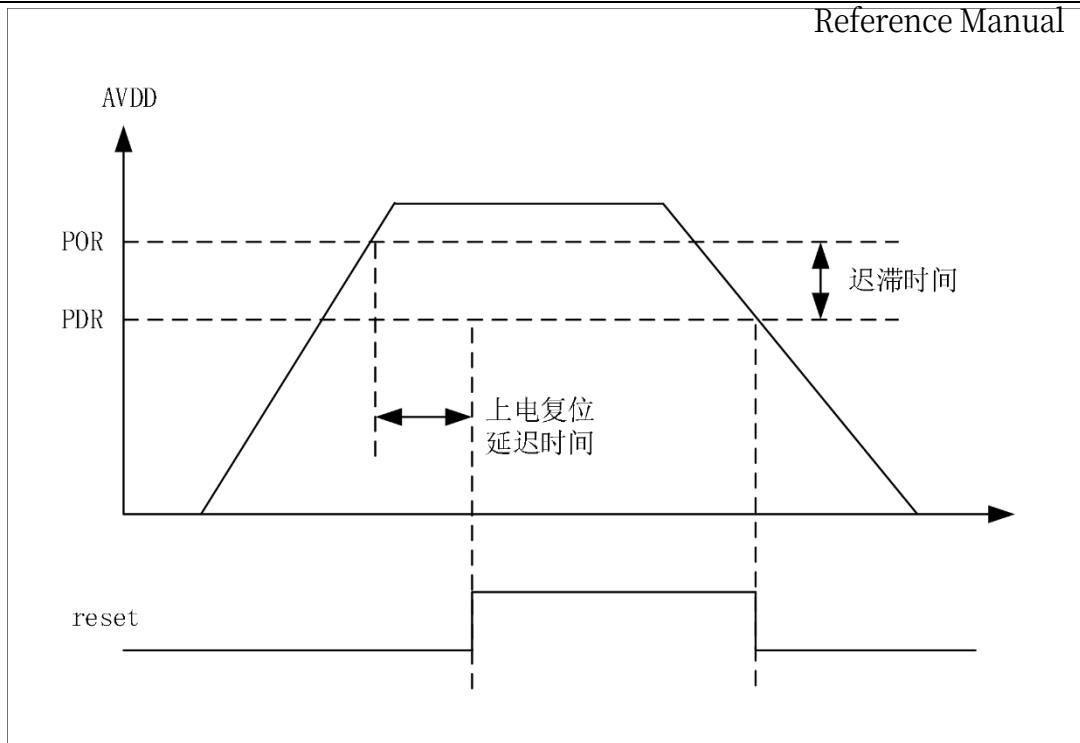


Figure 5-5 Power-on Reset and Power-off Reset Waveform Schematics

where **reset** is the total reset signal for digital circuits, resetting all digital circuits.

For details on the threshold voltage thresholds for power-on reset and power-off reset, hysteresis for power-on reset and power-off reset, and power-on reset delay time, refer to the contents of the Electrical Characteristics related section of the datasheet.

Pin Reset

The chip supports a specific external pin
reset, where a low level generates a reset. The
waveform schematic for external pin reset is
shown below:

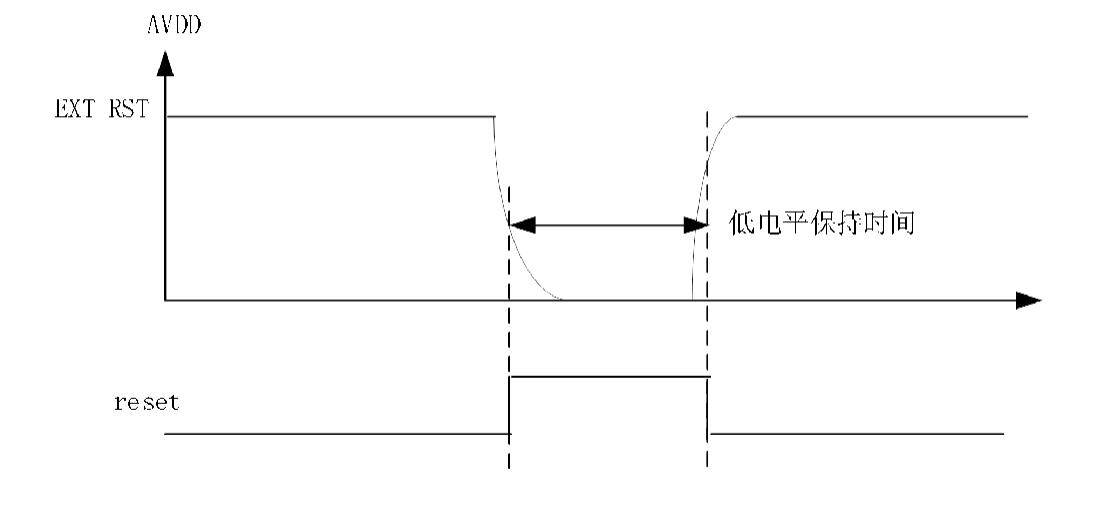


Figure 5-6 External Pin Reset Waveform Schematic

where **reset** is the total reset signal for digital circuits, resetting all digital circuits.

Refer to the Electrical Characteristics section of the datasheet for details such as pin reset low hold time.

watchdog reset

This chip has two watchdogs: an independent watchdog (**IWDT**) and a window watchdog (**WWDT**). Both can generate a watchdog reset and can reset all digital circuits on the chip, including the CPU core, buses, memory controllers, peripheral circuits, and so on.

CPU soft reset

CPU by writing 0x05fa0004 to address 0xe000ed0c via ARM CORTEX M0's own soft reset instruction.

The core issues a software reset operation.

This reset resets the vast majority of digital circuitry. Instead of the PMU/RTC module in the power-down domain; **VREF_VOLT_DELTA, RC_FREQ_DELTA, DEVICE_ID0, DEVICE_ID1, DEVICE_ID2, DEVICE_ID3** within the **SYSCON** module in the power-down domain.

DP32G030
Reference Manual

registers, these circuits will not be reset.

Reset Source Reset Range

Table 1-1 Reset Source Range

reset source	PMU/RTC	most digital circuits	Special Circuitry Not Affected by CPU Soft Reset
Power-on/power-off reset	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)
Pin Reset	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)
watchdog reset	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)
CPU soft reset	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)	non-reset

Among them, the circuits that are specifically not affected by the CPU soft reset refer to the circuits described in the contents of the CPU soft reset chapter. Most of the digital circuits refer to CPU cores, buses, memory controllers, clock reset control circuits, and peripheral devices.

Low Power Mode

After the system power supply of the chip has been powered up and reset, the chip is in normal operation. At this time, the system clock defaults to the internal high-frequency RC oscillator clock, the output frequency is 24MHz, all the circuits can work normally, the CPU begins to take the finger, the program runs normally.

Various low-power modes are available to save chip power. The chip does not need to run continuously, such as when waiting for an external event or when waiting for a long period of time. Users can select an appropriate low-power mode based on requirements such as minimum power consumption, boot time requirements, and wake-up sources.

There are four main low-power operating modes: **WFI** mode, **SLEEP** mode, **DEEPSLEEP** mode, and nuclear **STOP**.

Mode.

- **WFI** mode: only the **CPU** core is stopped, other clocks, power supply, and all peripherals including the **CPU** core's peripherals such as **NVIC**, **SYSTICK**, etc. are still running in their original state.
- **SLEEP** mode: All digital circuits are not powered down. The voltage regulator (**LDO**) will also work in low-power mode, and the output voltage can be adjusted to further reduce the overall power consumption. At this time, the internal high-frequency **RC** oscillator clock source is turned off, the system clock is turned off, and only the internal low-frequency **RC** oscillator clock is left operating for the **RTC** and wake-up circuits.

- DEEPSLEEP mode: the digital circuits in the 1.2V power-down domain are powered down, including digital circuits such as the CPU core, memory, and most of the peripherals, etc. The digital circuits in the 1.2V non-power-down domain are in the powered state, and the supply voltage regulator at that time (LDO) will also work in low-power mode, and the output voltage can be adjustable section, can do to further reduce the overall power consumption. At this time, the internal high-frequency RC oscillator clock source is turned off, and only the internal low-frequency RC oscillator clock operates for RTC and wake-up circuits.
- STOP mode: 1.2V power off, all clock sources off.

In addition, power consumption can be reduced under normal operating conditions in the following ways:

- Lower the system clock.
 - Turns off unused peripheral
- clocks on the bus. The following table gives detailed information in low power mode:

[Table Low Power Mode Detailed Information](#)

paradigm	go into	awakens	Impact on the digital domain	Impact on the clock source	Impact on voltage regulators
WFI mode	WFI	CPU interrupt	CPU clock off, no effect on system clock and other peripheral	not have	not have

			clocks	Reference Manual	
SLEEP	Set the LPOW_MD SLEEP of the register Position 1	All IO or RTC	Turn off the system clock and periphera l clocks	RCHF Off	Enables low- power mode and adjustable output voltage (TRIM_LPLDO)
DEEPSLEEP	Set the LPOW_MD registers DEEPSLEEP Bit 1	Specific IO or RTC	1.2V The power- down domain is dead. Turn off system clock and peripheral clocks	RCHF Off	Enables low- power mode and adjustable output voltage (TRIM_LPLDO)
STOP	Set the LPOW_MD STOP register Position 1	Specific IO	All digital domains are dead.	RCHF and RCLF Close	cloture

Each low-power mode is described in detail below:

Reduced system clock

In normal operating mode, the frequency of the system clock can be reduced by programming `DIV_CLK_SEL` in the `CLK_SEL` register. See Section 11.6.1: Clock Selection Register (`CLK_SEL`) for details.

Peripheral Clock Control

The chip provides independent peripheral clock control for each peripheral, and power consumption can be reduced by turning off the peripheral clock at any time during normal operating mode.

The clock of each peripheral module is switched on and off by programming the respective peripheral clock control bits in the `DEV_CLK_GATE` register.

WFI mode

By executing the `WFI` instruction of the ARM CPU core, the CPU core can be directly put into standby mode. In this mode, only the CPU is stopped and all other IO pins and peripheral modules remain in their normal operating mode. Any peripheral interrupt responded to by the Nested Vector Interrupt Controller can wake up the CPU core from standby and continue to execute instructions normally.

This mode takes the shortest amount of time to wake up, which can be accomplished in just a few system clock cycles since there is no time lost on the entry or exit of interrupts.

SLEEP mode

The chip can be actively put into SLEEP mode by software by setting `SLEEP` position 1 in the `LPOW_MD` register. In order to keep the overall power consumption of the chip as low as possible in SLEEP mode, modules that do not

need to work (e.g., ADC module, operational amplifier module, peripherals that do not need to work, etc.) can be shut down before entering to save power consumption, and the output voltage of the voltage regulator can be lowered under low-power mode by using the LPLDO voltage adjusting bit (TRIM_LPLDO) in the TRIM_POW3 register, thus further reducing the power consumption mode through the LPLDO voltage adjustment bit (TRIM_LPLDO) of the TRIM_POW3 register, thus further reducing power consumption.

In SLEEP mode, the analog power control circuit turns off all analog modules that consume large amounts of power (e.g., RCHF, etc.) and retains only the basic BG, voltage regulator, and built-in low-frequency RC oscillator (RCLF) clocks, and keeps the voltage regulator in low-power mode, and the high-frequency clocks and system clocks in the chip are also low-power due to the built-in high-frequency RC oscillator.

(RCHF) is turned off and stopped. At this time, only the circuits in the PMU module and RTC module that are working under the RCLF clock are working normally, and all other digital circuits are stopped due to the absence of the clock.

SLEEP mode can be exited by external IO signal or RTC timing signal, PMU will clear the SLEEP_MODE signal to 0 to exit SLEEP mode when it detects the corresponding IO or RTC valid wake-up signal, and the analog power control circuit will start the wake-up process when it detects that the SLEEP_MODE signal becomes 0 to return to normal operation mode from SLEEP mode. When the analog power control circuit detects that the SLEEP_MODE signal has changed to 0, it will start the wake-up process to restore the chip from SLEEP mode to normal operation mode.

All IOs support SLEEP mode wakeup. The

wake-up time for this mode is about

100us.

The SLEEP mode entry and exit schematic is shown below:

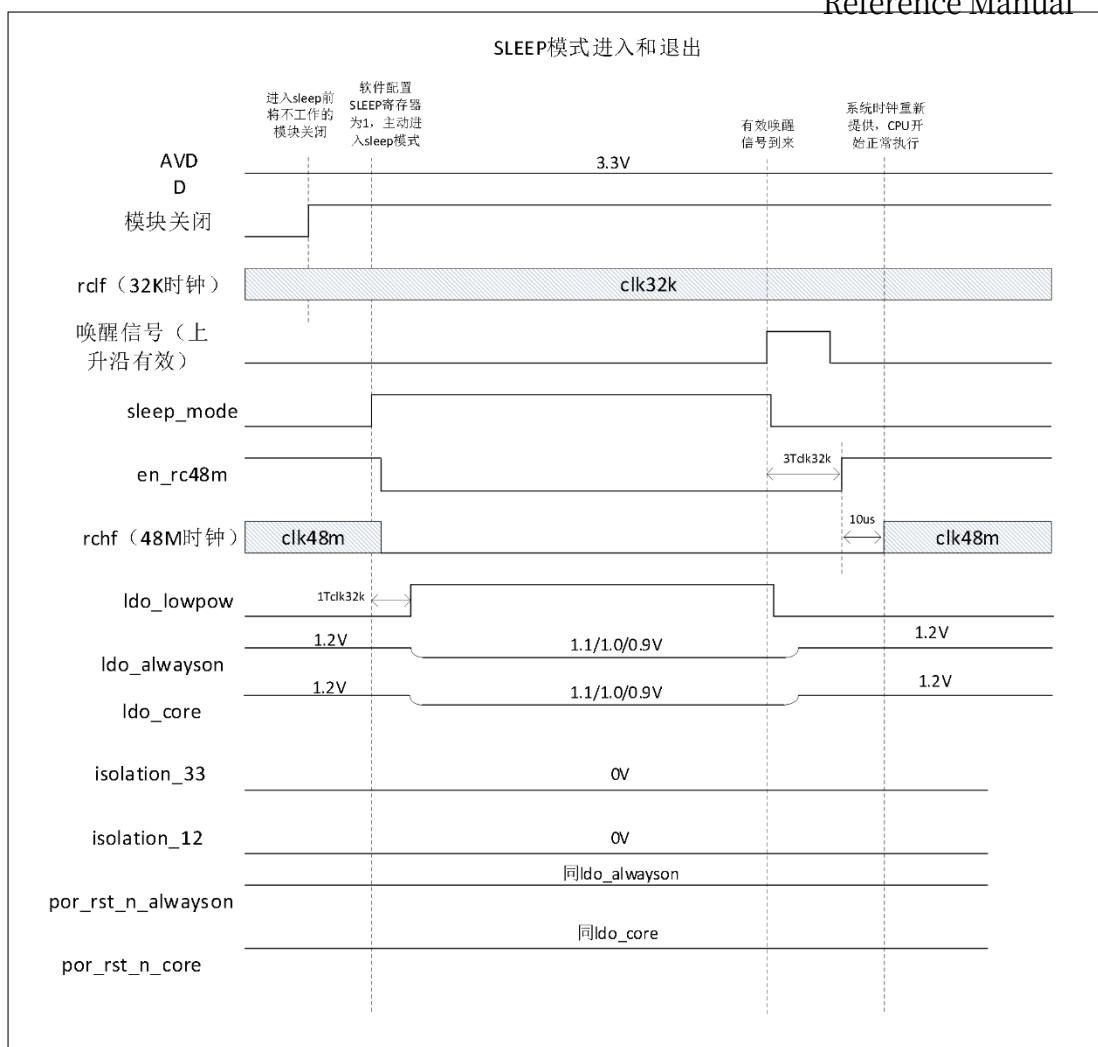


Figure 5-7 SLEEP Mode Entry and Exit Schematics

DEEPSLEEP mode

The chip can be actively put into DEEPSLEEP mode by software by setting DEEPSLEEP position 1 in the LPOW_MD register. In order to keep the overall power consumption of the chip as low as possible in DEEPSLEEP mode, modules that do not need to work (e.g., ADC module, op-amp module, and digital function peripherals that do not need to work) can be shut down before entering to save power consumption, and the output voltage of the voltage regulator in low-power mode can be adjusted down through the TRIM_LPLDO register, which further reduces the power consumption. In addition, the output voltage of the voltage regulator in low-power mode can be lowered through the TRIM_LPLDO register to further

DP32G030
Reference Manual

reduce power consumption.

In DEEPSLEEP mode, the analog power control circuit turns off all analog modules with high power consumption (e.g., RCHF, etc.) leaving only the basic BG, voltage regulator, and built-in low-frequency RC oscillator (RCLF) clock, and

The voltage regulator is operated in low-power mode, and the high-frequency clock in the chip as well as the system clock are also stopped due to the shutdown of the built-in high-frequency RC oscillator (**RCHF**).

The power supply to the digital circuits in the 1.2V power-down domain will also be disconnected, causing all circuits in the power-down domain to be powered down for further power savings.

The 1.2V non-dropout domain is still powered by the voltage regulator, and except for the circuits in the **PMU** and **RTC** modules in this power domain that are operating under the **RCLF** clock, which are working normally, the other digital circuits are stopped due to the absence of the clock.

With this chip, **RAM** is in the power-down domain and data is not saved due to power-down.

DEEPSLEEP mode can be exited by external **IO** signal or **RTC** timing signal, **PMU** will clear the **DEEPSLEEP_MODE** signal to 0 when it detects the corresponding **IO** or **RTC** valid wake-up signal to indicate the exit from **DEEPSLEEP** mode, and the analog power control circuit will start the wake-up process when it detects that the **DEEPSLEEP_MODE** signal has changed to 0 to enable the chip to resume the normal operation mode from **DEEPSLEEP** mode. When the analog power control circuit detects that the **DEEPSLEEP_MODE** signal turns to 0, it will start the wake-up process to restore the chip from **DEEPSLEEP** mode to normal operation mode.

Only certain **IOs** on this chip support wake-up in **DEEPSLEEP** mode. The wake-up time for this mode is about 140us.

The **DEEPSLEEP** mode entry and exit schematic is shown below:

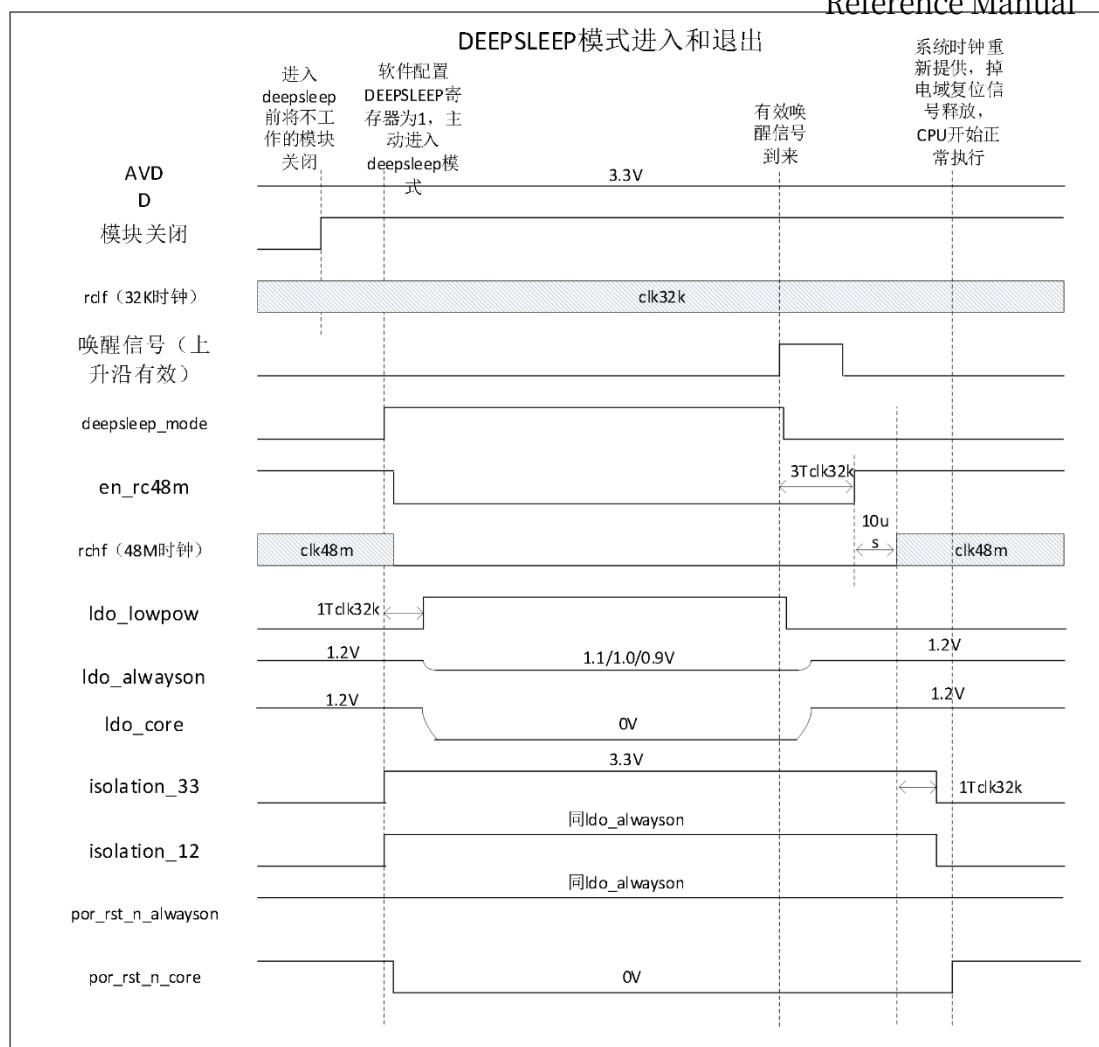


Figure 5-8 DEEPSLEEP Mode Entry and Exit Schematic

STOP mode

The chip can be put into STOP mode actively by software by setting the STOP position 1 in the LPOW_MD register.

In STOP mode, the voltage regulator is turned off, all 1.2V digital circuits are powered down, all analog modules in the AVDD power domain are turned off, and only a very small portion of the analog wake-up circuits are left operating.

STOP mode can be exited by external IO signals to wake up. When the analog wake-up circuit detects the corresponding wake-up signal, it will start the wake-up process to restore the chip from STOP mode to normal operation mode.

DP32G030

This chip supports the wake-up function in STOP mode on [Reference Manual](#)

This mode takes about 350us to wake up.

The STOP mode entry and exit schematic is shown below:

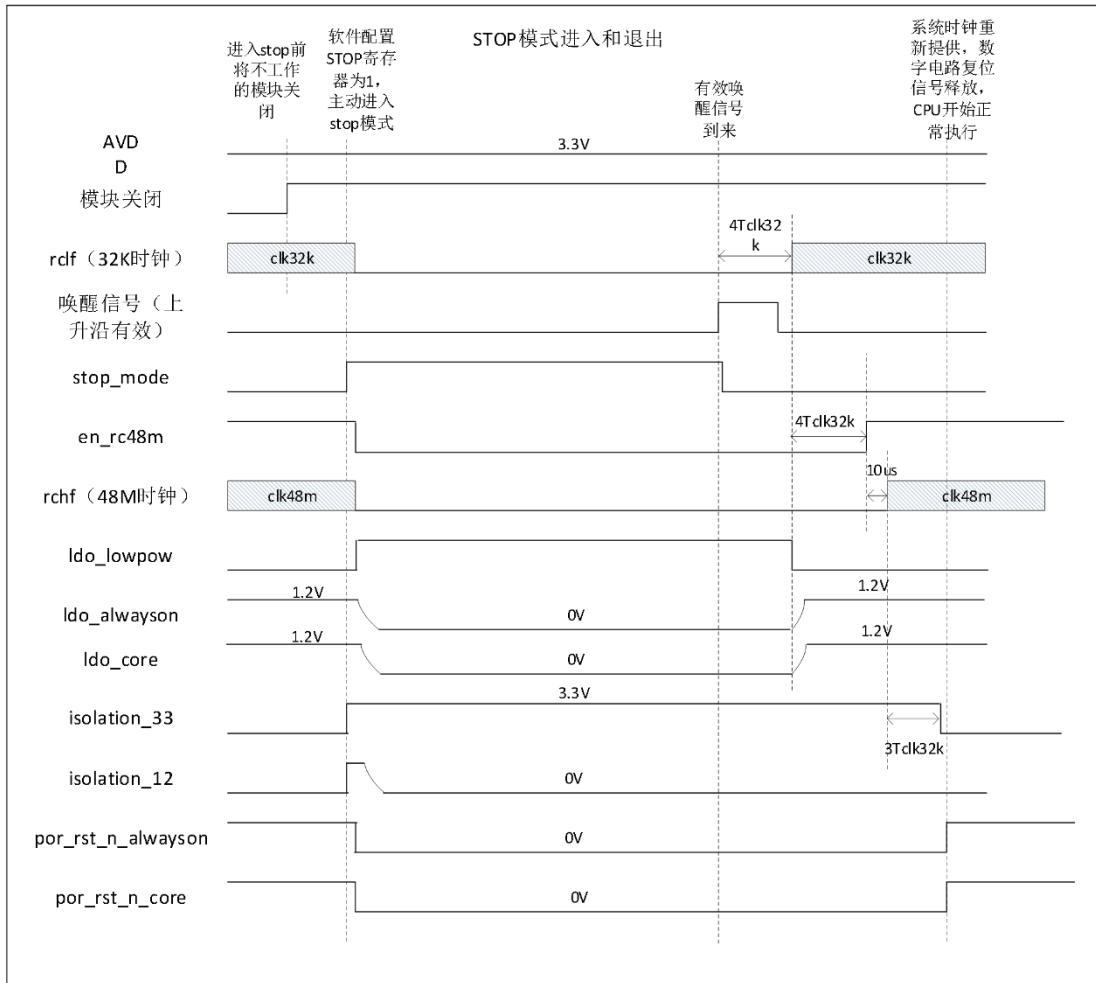


Figure 5-9 STOP Mode Entry and Exit Schematic

register map

Controls which low-power modes the entire chip enters: sleep, deepsleep, and stop.

This module is a power consumption control module, including the generation of various entry low-power control circuits, wake-up circuits, and data (including analog TRIM values) that need to be held at power-down.

name (of a thing)	misalign ment	typology	reset value	descriptive
PMU	BASE:0x40000800			
LPOW_MD	0x00	R/W	0x00	Low Power Mode Selection Register
LPMD_WKEN	0x04	R/W	0x00	Low Power Wake-Up Source Enable Registers
LPMD_WKST	0x08	R/W	0x00	Low-power wake-up source status registers
CHIP_RST_ST	0x0C	R/W	0x01	Chip Reset Status Register
SRC_CFG	0x10	R/W	0x03	Clock Source Configuration Register
TRIM_POW0	0x20	R/W	0x00	POW0 Related Analog Module TRIM Registers
TRIM_POW1	0x24	R/W	0x00	POW1 Related Analog Module TRIM Registers
TRIM_POW2	0x28	R/W	0x00	POW2 Related Analog Module TRIM Registers
TRIM_POW3	0x2C	R/W	0x00	POW3 Related Analog Module TRIM Registers
TRIM_RCHF	0x30	R/W	0x808	RCHF Clock Module TRIM Register
TRIM_RCLF	0x34	R/W	0x810	RCLF Clock Module TRIM Register
TRIM_OPA	0x38	R/W	0x00	OPA Module TRIM Register
TRIM_PLL	0x3c	R/W	0x00	PLL Module TRIM Register
TRIM_LOCK	0x80	R/W	0x00	TRIM Locks the registers
DATA_BAKE0	0x100	R/W	0x00	No power-down domain data backup register 0
DATA_BAKE1	0x104	R/W	0x00	No power-down domain data backup register 1
DATA_BAKE2	0x108	R/W	0x00	No power-down domain data backup register 2
DATA_BAKE3	0x10C	R/W	0x00	No power-down domain data

register description

LPOW_MD register (0x00)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:4	RESERVED	R	0	reserved bit	Reference Manual
3	STOP	R/W	0	Write 1 to this register, the chip enters STOP mode.	
2	DEEPSLEEP	R/W	0	Write 1 to this register, the chip enters DEEPSLEEP mode Software writes 1, hardware clears it automatically	
1	SLEEP	R/W	0	Write 1 to this register, the chip enters SLEEP mode Software writes 1, hardware clears it automatically	
0	RESERVED	R	0	reserved bit	

Note: The chip enters the low-power mode from the normal operating mode, and can only enter one kind of low-power mode each time, and after waking up, it will exit from the low-power mode and return to the normal operating mode, and then be configured by the software to enter another kind of low-power mode.

LPMD_WKEN register (0x04)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:3	RESERVED	R	0	reserved bit
2	IO_WKEN	R/W	0	IO wake-up enable in low-power mode 0: disable 1: Enable Note 1: Which IO has the wake-up function can be configured through the PORTA_WKE, PORTB_WKE, PORTC_WKE registers.
1	rtc_tim_wk en	R/W	0	RTC time signal wake-up enable in low-power mode 1: RTC time signal with low-power wake-up function
0	RTC_ALA_WK EN	R/W	0	RTC alarm signal wake-up enable in low-power mode 1: RTC alarm signal with low-power wake-up function

LPMD_WKST register (0x08)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:3	RESERVED	R	0	reserved bit

Reference Manual				
2	IO_WKST	R/W	0	IO wake-up flag in low-power mode 1: Wake on IO event 0: no IO event wakeup hardware set to 1, software write 1 clear
1	RTC_TIM_WKST	R/W	0	RTC time to wake up flag in low power mode 1: RTC time event wakeup hardware set to 1, software write 1 clear
0	RTC_ALA_WKST	R/W	0	RTC alarm clock wake-up flag in low-power mode 1: RTC alarm event wakeup hardware set to 1, software write 1 clear

CHIP_RST_ST register (0x0C)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:3	RESERVED	R	0	reserved bit
2	WWDT_RST_ST	R/W	0	WWDT Reset Status Flag Register 0: Indicates that no WWDT reset has occurred 1: Indicates a WWDT reset Write 1 Clear
1	IWDT_RST_ST	R/W	0	IWDT Reset Status Flag Register 0: Indicates that no IWDT reset has occurred. 1: Indicates an IWDT reset Write 1 Clear

Reference Manual Power-on reset status flag register				
0	POR_RST_ST	R/W	1	0: Indicates no power-on reset 1: Indicates a power-on reset Write 1 clear

SRC_CFG register (0x10)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:5	RESERVED	R	0	reserved bit	Reference Manual
4	RTC_CLK_SEL	R/W	0	RTC Clock Selection 0: RCLF 1: XTAL	
3	XTAL_EN	R/W	0	XTAL enable control bit 0: Close XTAL 1: Turn on XTAL	
2	XTAH_EN	R/W	0	XTAH Enable Control Bit 0: Close XTAH 1: Turn on XTAH	
1	RCHF_FSEL	R/W	1	RCHF Frequency selection control bit 0: 48MHz 1: 24 MHz	
0	RCHF_EN	R/W	1	RCHF enable control bit 0: Close RCHF 1: Enable RCHF	

TRIM_POW0 register (0x20)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:11	RESERVED	R	0	reserved bit
10:8	TRIM_TEMPC O_HPBG	R/W	0	HPBG temperature trim bit
7:4	TRIM_I_HP	R/W	0	HPBG current trim bit
3:0	TRIM_V_HP	R/W	0	HPBG Voltage trim bit

TRIM_POW1 register (0x24)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:8	RESERVED	R	0	reserved bit
------	----------	---	---	--------------

7:4	TRIM_V_LP	R/W	0	LPBG Voltage trim bit	Reference Manual
3:0	TRIM_TEMPC O_LPBG	R/W	0	LPBG temperature trim bit	

TRIM_POW2 register (0x28)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:0	RESERVED	R	0	reserved bit

TRIM_POW3 register (0x2C)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:4	RESERVED	R	0	reserved bit
3	TRIM_HPLDO _H	R/W	0	HPLDO voltage adjusted to 1.264v 0: unchanged 1: Adjust upward to 1.264v
2:1	TRIM_LPLDO	R/W	0	LPLDO Voltage output trim bit 00: 1.1V 01: 1.0V 10: 0.9V 11: 0.8V
0	TRIM_PD_UV LO	R/W	0	UVLO33 trim bit 0: Under SLEEP, the chip resets when the supply voltage drops to 1.8V; the analog circuitry consumes an additional 0.6uA of power in this case; 1: Under SLEEP, the chip resets when the supply voltage drops to 1.3V (+500mV deviation); the analog circuitry saves 0.6uA of power consumption;

TRIM_RCHF register (0x30)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:12	RESERVED	R	0	reserved bit	Reference Manual
11:8	TRIM_N	R/W	0x8	RCHF N trim bit	
7:4	RESERVED	R	0	reserved bit	
3:0	TRIM_P	R/W	0x8	RCHF P trim bit	

TRIM_RCLF register (0x34)

bitfield (math.)	name (of a thing)	typolog ogy	reset value	descriptive
31:12	RESERVED	R	0	reserved bit
11:8	TRIM_CS	R/W	0x8	RCLF CS trim bit (coarse trim bit)
7:5	RESERVED	R	0	reserved bit
4:0	TRIM_FINE	R/W	0x10	RCLF FINE trim bit (fine trim bit)

TRIM_OPA register (0x38)

bitfield (math.)	name (of a thing)	typolog ogy	reset value	descriptive
31:20	RESERVED	R	0	reserved bit
19:15	OPA1_TRIMP	R/W	0	P-terminal TRIM bit of OPA1
14:10	OPA1_TRIMN	R/W	0	N-terminal TRIM bit of OPA1
9:5	OPA0_TRIMP	R/W	0	P-terminal TRIM bit of OPA0
4:0	OPA0_TRIMN	R/W	0	N-terminal TRIM bit of OPA0

TRIM_PLL register (0x3C)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:4	RESERVED	R	0	reserved bit
3:0	PLL_R_TRSIM	R/W	0	R value of PLL TRIM bit

TRIM_LOCK register (0x80)

bitfield Id (mat h.)	name (of a thing)	typolog y	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:8	TRIM_UNLOCK	W	0	The corresponding TRIM configuration register can be rewritten by writing this register to 0xAA.
7:0	TRIM_LOCK	W	0	Writing this register to 0x55 prevents the corresponding TRIM configuration register from being rewritten, and is used to protect against accidental rewriting of TRIM registers.

DATA_BAK0 register (0x100)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:0	DATA_BAK0	R/W	0	Data backup 0

DATA_BAK1 register (0x104)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:0	DATA_BAK1	R/W	0	Data backup 1

DATA_BAK2 register (0x108)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:0	DATA_BAK2	R/W	0	Data backup 2

DATA_BAK3 register (0x10C)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:0	DATA_BAK3	R/W	0	Data backup 3

5.6 System Control (SYSCON)

5.6.1 summarize

The chip manages and controls the clock network of the overall chip system through the system control unit, providing clocks for each module and function in the chip, including the system clock, peripheral clocks of each module, and clocks of each special function. The unit can also independently control the clock on or off of each module, the selection and frequency division of the system clock source, and the frequency division of the special function clock to achieve reasonable power consumption control.

5.6.2 characterization

- System clock source selection, frequency division and control
- Separate switch for each peripheral
- With 128bit chip unique identifier

5.6.3 Block Diagram of Module Structure

There are 5 clock sources: RCHF (48MHz internal RC oscillator) RCLF (32768Hz internal RC oscillator)
PLL (up to 72MHz) XTAH (4-32MHz crystal) XTAL (32768Hz crystal)

The chip clock network is structured as follows

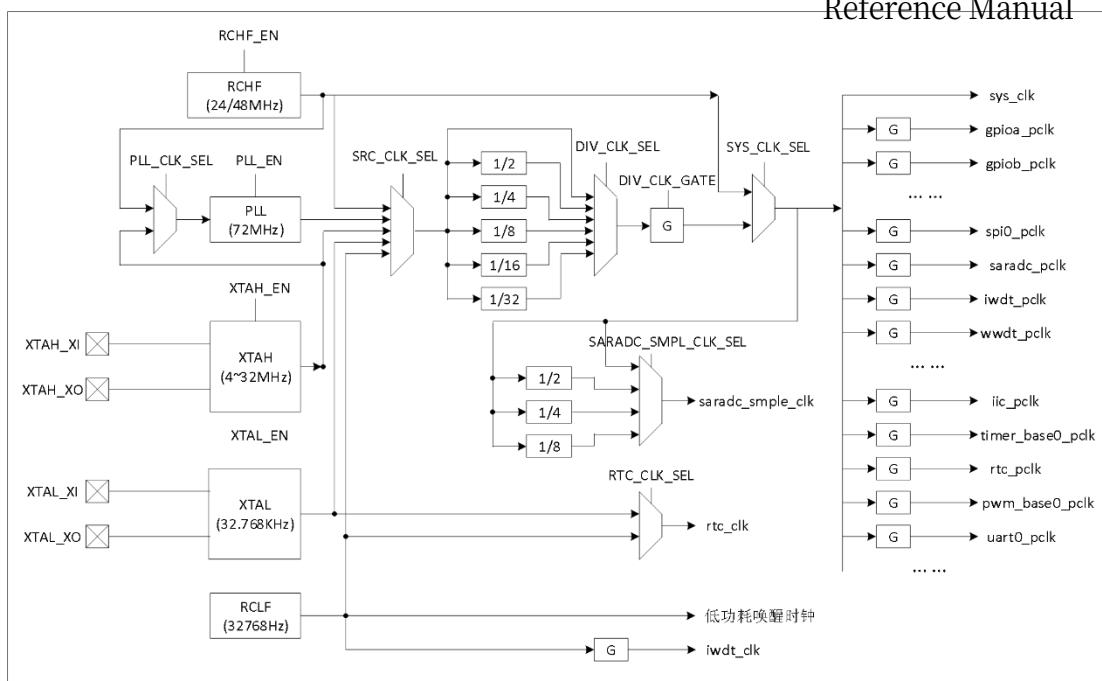


Figure 5-10 Chip Clock Structure

`sys_clk` (system clock) selects one of five clock sources and can also be divided 1/2/4/8/16/32

Frequency.

The CPU Cortex-M0's FCLK, HCLK, and SCLK are all clocked by `sys_clk`, so the M0's internal SYSTICK

The clock source is `sys_clk`.

All peripheral device bus clocks have their own clock control registers to control their on or off, so that their clocks can be turned off when the module is not in use, to achieve the purpose of saving power consumption; and their bus clock frequency is consistent with the system clock frequency.

The sampling clock of the SARADC is selectable as `sys_clk` through the `SARADC_SMPL_CLK_SEL` bit in the `CLK_SEL` register.

The 1/2/4/8 division frequency of the sampling clock can be selected depending on the usage scenario of the sampled signal.

The independent watchdog (IWDT) has a low-frequency clock source for counting use that is a fixed RCLF clock supply and is controlled by the `IWDT_CLK_GATE` bit of the `DEV_CLK_GATE` register to

turn this clock on or off.

The window watchdog (WWDT) uses the system clock for counting purposes and is subject to the **DEV_CLK_GATE** register's

The **WWDT_CLK_GATE** bit controls whether this clock is turned on or off.

The counting clock of the RTC can be selected from either **RCLF** or **XTAL** clock sources through the **DEV_CLK_GATE** register.

RTC_CLK_SEL bit to configure the selection.

SLEEP mode and DEEPSLEEP mode fix the use of RCLF as the wake-up clock, and the RCLF clock cannot be turned off.

5.6.4 Functional Description

The external high-frequency crystal clock (XTAH) can be generated from two clock sources: an external crystal/ceramic resonator or the user's external clock.

To minimize clock output distortion and shorten startup stabilization time, the crystal/ceramic resonator and load capacitance must be placed as close as possible to the oscillator pins. And the load capacitance must be matched and adjusted to the selected oscillator.

1. External crystal/ceramic resonator

The XTAH supports an external 4-32MHz crystal/ceramic resonator to provide a very accurate clock for the chip. The hardware configuration can be seen in the figure below and further information can be found in the Electrical Characteristics section of the datasheet.

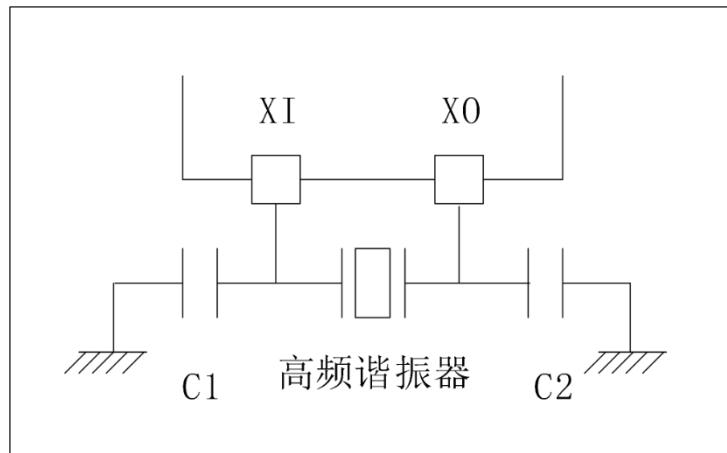


Figure 5-11 Hardware Configuration Diagram for External High Frequency Crystals

2. External clock

The XTAH also supports the connection of an external clock directly into the XTAH with a frequency of up to 48 MHz. the external clock signal must be connected to the XI pin and the XO pin must be left free. The hardware configuration is shown in the following figure.

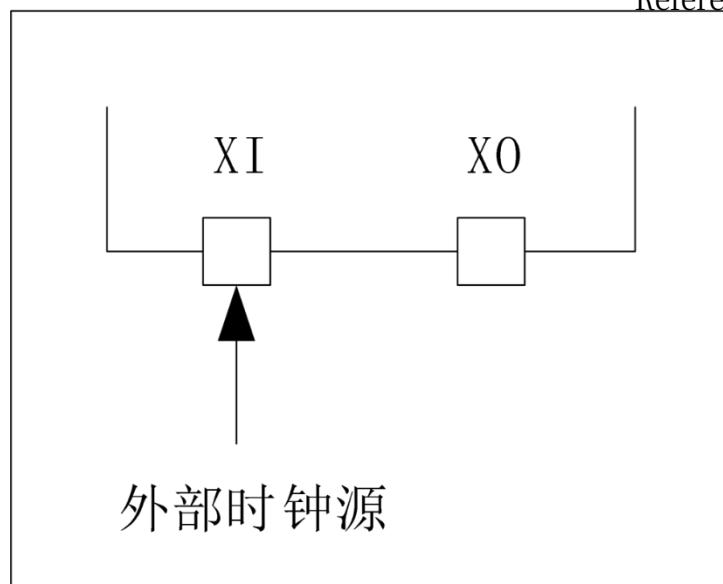


Figure 5-12 External Clock Related Hardware Configuration

The XTAH clock can be enabled or disabled by configuring the XTAH_EN bit in the SRC_CFG register in the PMU. After startup, wait at least 10ms for normal use.

RCHF Clock

The RCHF clock signal is generated by the chip's internal high-frequency RC oscillator (48MHz). It can be turned on or off by configuring the RCHF_EN bit in the SRC_CFG register in the PMU, and the 48MHz output or 24MHz output can be selected by configuring the RCHF_FSEL bit in the SRC_CFG register in the PMU.

The RCHF RC oscillator can provide the clock required by the system without the need for any external devices. It has a shorter start-up time than the XTAH crystal oscillator, but even after calibration its clock frequency accuracy is still poor. Further information can be found in the Electrical Characteristics section of the datasheet.

calibrations

The manufacturing process determines that the RC oscillator frequency will

vary from chip to chip, which is why the RCHF clock frequency has been calibrated to within $\pm 1\%$ (25°C) before leaving the factory. After a system reset, the calibration value written to a specific address in the memory during the production test can be loaded into the corresponding bit of the TRIM_RCHF register by software, and it is also possible to

The actual frequency value written to a specific address in the memory during the production test is loaded into the RCHF_DELTA and RCHF_SIG bits of the RC_FREQ_DELTA register, so that the deviation of the current frequency from the ideal frequency can be known at the time of use, thus obtaining a more accurate clock frequency.

If the user bases the usage scenario on a different voltage or ambient temperature, this will affect the accuracy of the RC oscillator. The user can also adjust the RCHF frequency by using the TRIM_P bit in the TRIM_RCHF register.

PLL Clock

The PLL can use one of two clock sources (XTAH or RCHF) as a reference clock input to produce a multiplied clock output. See the clock network structure diagram and the PLL_CLK_SEL bit in the CLK_SEL register for a description.

Before PLLs can be enabled, the configuration of the PLLs must be completed (clock source selection, prescaler and multiplier factors, etc.) and should not be enabled until their input clocks have stabilized. These parameters cannot be changed once the PLL is enabled.

When you need to change the configuration of a PLL, you must first turn off the activated PLL and then change the corresponding configuration before re-enabling the PLL.

When PLL is enabled, the PLL_LOCK status bit of the PLL_ST register can be used to determine whether the PLL has completed locking, and the PLL clock can only be used after locking. It takes about 30us for the PLL to lock after it is enabled. The PLL clock can only be used after it is locked. It takes about 30us for the PLL to lock after being enabled.

The PLL clock frequency is mainly obtained from the input clock by using the multiplier frequency through the parameters PLL_M bit and PLL_N bit of the PLL_CTRL register. The formula is as follows:

$$f_{pll} = \frac{f_{in}}{M} \times N$$

Where f_{PLL} is the PLL output clock frequency and $f_{Reference}$ is the reference clock frequency. m is the register **PLL_M** configuration value and n is the register **PLL_N** configuration value.

The PLL module can select two input clock sources, RCHF or XTAH, through the **PLL_CLK_SEL** bit of register **CLK_SEL**, where the reference clock frequency of the PLL is obtained through the input **clock/PLL_M**, and the reference clock frequency ranges from 3MHz to 6MHz.

The maximum frequency of PLL output on this chip is 72MHz, and the following table is recommended for the configuration of the input clock, PLL_M and PLL_N:

Input Signal Source	Input frequency	PLL_M	PLL Reference Clock Frequency	PLL_N	PLL Clock
RCHF	24	8	3	24	72
RCHF	24	6	4	18	72
RCHF	24	4	6	12	72
RCHF	24	6	4	16	64
RCHF	24	6	4	14	56
RCHF	24	6	4	12	48
XTAH	4	1	4	18	72
XTAH	8	2	4	18	72
XTAH	12	2	6	12	72
XTAH	12	4	3	24	72
XTAH	16	4	4	18	72
XTAH	24	4	6	12	72
XTAH	24	6	4	18	72
XTAH	24	8	3	24	72
XTAH	32	8	4	18	72
XTAH	32	8	4	16	64

XTAL Clock

The external low frequency crystal clock (XTAL) can be plugged into a 32768Hz crystal/ceramic resonator. It provides a low power and accurate 32768Hz clock source for real time clocks or other circuits.

The XTAL clock can be started up or shut down by configuring the XTAL_EN bit in the SRC_CFG. After startup, wait for at least 2s to work properly.

The XTAL supports an external 32768Hz crystal/ceramic resonator to provide a very accurate clock for the chip. The hardware configuration is shown in the figure below and further information can be found in the Electrical Characteristics section of the datasheet.

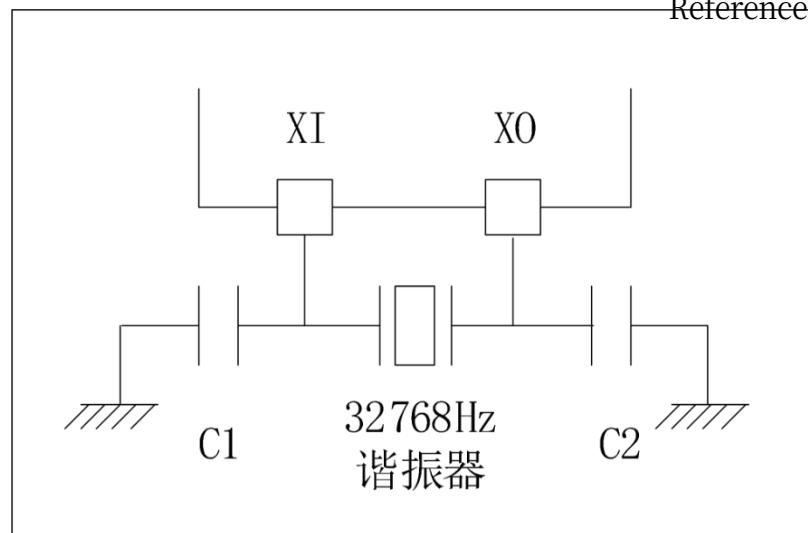


Figure 5-13 Hardware Configuration Diagram for External Low Frequency Crystals

RCLF Clock

The RCLF clock signal is generated by the chip's internal low-frequency RC oscillator (32768Hz), acting as a low-power, low-frequency clock source, which can be kept running in SLEEP and DEEPSLEEP modes to provide clocks for the IWDT, real-time clock, and wake-up circuits.

The RCLF clock frequency is 32768 Hz and this clock cannot be switched off. Further information can be found in the Electrical Characteristics section of the datasheet.

calibrations

The manufacturing process determines that the RC oscillator frequency will vary from chip to chip, which is why the RCLF clock frequency of each chip has been calibrated to within $\pm 1\%$ (25°C) before leaving the factory. After a system reset, the software can load the calibration value written to a specific address in the memory during the production test into the corresponding bit of the TRIM_RCLF register, and also load the actual frequency value written to a specific address in the memory during the production test into

the RCLF_DELTA and RCLF_SIG bits of the RC_FREQ_DELTA Register, so the deviation of the current frequency from the ideal frequency can be obtained at the time of use. When used, the deviation of the current frequency from the ideal frequency can be known, so that a more accurate clock frequency can be obtained.

If the user bases the usage scenario on a different voltage or ambient temperature, this will affect the accuracy of the RC oscillator. The user can also adjust the RCLF frequency with the TRIM_FINE bit in the TRIM_RCLF register.

System clock (sys_clk) selection

After system reset, the system clock (sys_clk) uses RCHF as the default clock source after power-up, and the RCHF clock output frequency is 24MHz.

Referring to the clock network structure, the system clock can be selected at any time by switching between the RCHF clock and the DIV_CLK clock via the SYS_CLK_SEL bit in the CLK_SEL register.

If the clock source and crossover frequency of the DIV_CLK clock are changed, the system clock must be configured as 0 and the system clock selected as RCHF clock to ensure that the system clock will not be affected during the switching of the clock source. Next, turn off the DIV_CLK clock by configuring the DIV_CLK_GATE register to 0. Then select the frequency to be output from the DIV_CLK clock by using the SRC_CLK_SEL bit and the DIV_CLK_SEL bit in the CLK_SEL register and select DIV_CLK_GATE to 1 to turn on the DIV_CLK clock output, and finally configure the SYS_CLK_SEL to 0 to ensure that the system clock will not be affected during the clock source switching process. After the selection, configure DIV_CLK_GATE to 1 to turn on the DIV_CLK clock output, and finally configure SYS_CLK_SEL to 1 to select the system clock as the DIV_CLK clock output.

RTC Clock

The RTCCLK clock source can be provided by either the XTAL or RCLF clock by setting the RTC_CLK_SEL bit in the SRC_CFG register.

Before the RTC is enabled, all configurations of the RTC (clock source selection, etc.) must be completed, and the RTC must be configured in a way that

allows the RTC to be used when changing the
The `RTC_CLK_SEL` bit must be enabled for at least 70us.

Whether in normal operating mode, SLEEP mode, or DEEPSLEEP mode, the selection is flexible depending on the application.

Independent Watchdog Dog (IWDT) Clock

The independent watchdog count clock source is provided by the RCLF clock, which cannot be turned off and is controlled by the IWDT_CLK_GATE bit in the DEV_CLK_GATE register, which is only available when configured to one.

Window Watchdog Dog (WWDT) Clock

The window watchdog's count clock is at the same frequency as the module's bus clock and system clock, and is subject to the DEV_CLK_GATE Controlled by the WWDT_CLK_GATE bit in the register, WWDTCLK will only function properly when this bit is configured to 1.

SARADC Sampling Clock

The SARADC sampling clock can be selected from the following four frequencies via the SARADC_SMPL_CLK_SEL bit in the CLK_SEL register: system clock, system clock 2-division, system clock 4-division, and system clock 8-division.

Low-power mode wake-up clock

In the two low-power modes, SLEEP and DEEPSLEEP, only the RCLF clock operates and the wake-up for low-power modes is driven through this clock.

128-bit Chip Unique Identifier

This chip has a 128bit Chip Unique Identifier, which

is unique per piece. The unique identification of the product is very suitable:

- Used as a serial number (e.g. USB character serial number or other terminal applications);
- Used as a password, this unique identifier is used in conjunction with software encryption and decryption algorithms when writing flash memory to improve the security of the code within the flash memory storage;
- is used to activate the bootstrap process with a safety mechanism;

The 128bit identification code can be read out and used in the four registers CHIP_ID0, CHIP_ID1, CHIP_ID2 and CHIP_ID3.

register map

name (of a thing)	offset	typology	reset value	descriptive
SYSCONBASE: 0x40000000				
CLK_SEL	0x00	R/W	0x02	Clock Select Register
DIV_CLK_GATE	0x04	R/W	0x01	Divided Clock Gating Register
DEV_CLK_GATE	0x08	R/W	0x00	Peripheral Clock Gating Register
RC_FREQ_DELTA	0x78	R/W	0x00	RCHF/RCLF Real Frequency Value Difference Registers
VREF_VOLT_DELTA	0x7C	R/W	0x00	VREF Real Voltage Value Difference Register
CHIP_ID0	0x80	R/W	0x00	Device ID Register 0
CHIP_ID1	0x84	R/W	0x00	Device ID Register 1
CHIP_ID2	0x88	R/W	0x00	Device ID Register 2
CHIP_ID3	0x8C	R/W	0x00	Device ID Register 3
PLL_CTRL	0x180	R/W	0x2c8	PLL Control Register
PLL_ST	0x184	R/W	0x00	PLL Status Register

register description

CLK_SEL register (0x00)

bitfield d (math .)	name (of a thing)	typolo gy	reset value	descriptive
31:12	RESERVED	R	0	reserved bit
11	PLL_CLK_SEL	R	0	PLL Input Clock Selection 0: RCHF 1: XTAH
11:10	SARADC_SMPL_ CLK_SEL	W	0	SARADC Sampling Clock Selection 00: 1 division of the system clock 01: 2 divisions of the system clock 10: 4 divisions of the system clock 11: 8-division frequency of the system clock
10:9	SARADC_SMPL_ CLK_SEL	R	0	SARADC Sampling Clock Selection 00: 1 division of the system clock 01: 2 divisions of the system clock 10: 4 divisions of the system clock 11: 8-division frequency of the system clock
8	RESERVED	R	0	reserved bit
7	RESERVED	R	0	reserved bit
7	PLL_CLK_SEL	W	0	PLL Input Clock Selection 0: RCHF 1: XTAH

Reference Manual Selection				
6:4	SRC_CLK_SEL	R/W	0	Source clock (SRC_CLK) selection 000: RCHF 001: RCLF 010: XTAH 011: XTAL 100: PLL Other: reserved
3:1	DIV_CLK_SEL	R/W	01	Divided frequency clock (DIV_CLK) selection 000: 1-division frequency of SRC_CLK 001: 2 divisions of SRC_CLK 010: 4-division frequency of SRC_CLK 011: 8-division frequency of SRC_CLK 100: 16 divisions of SRC_CLK 101: 32 divisions of SRC_CLK Other: Reserved
0	SYS_CLK_SEL	R/W	0	System Clock Selection 0: RCHF clock 1: DIV_CLK Clock

DIV_CLK_GATE register (0x04)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:1	RESERVED	R	0	reserved bit

0	DIV_CLK_GATE	R/W	0x1	Fractional Clock Gating Reference Manual 1: Divided Clock Output 0: Frequency division clock disabled Note: When it is necessary to change the DIV_CLK_SEL or SRC_CLK_SEL register, it is necessary to switch the system clock to RCHF first, then set this register to 0 to turn off DIV_CLK, and then finally change the value of DIV_CLK_SEL or SRC_CLK_SEL, so as to ensure the reliability of the clock.
---	--------------	-----	-----	--

DEV_CLK_GATE register (0x08)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)				
31:29	RESERVED	R	0	reserved bit
28	AES_CLK_GATE	R/W	0	AES128 Module Clock Gating

Reference Manual				
		R/W	0	CRC Module Clock Gating
27	CRC_CLK_GATE	R/W	0	CRC Module Clock Gating
26	RESERVED	R	0	reserved bit
25	SARADC_CLK_GATE	R/W	0	SARADC_CTRL Module Clock Gating
24	WWDT_CLK_GATE	R/W	0	WWDT Module Clock Gating
23	IWDT_CLK_GATE	R/W	0	IWDT Module Clock Gating
22	RTC_CLK_GATE	R/W	0	RTC Module Clock Gating
21	PWM_PLUS1_CLK_GATE	R/W	0	PWM_PLUS1 Module Clock Gating
20	PWM_PLUS0_CLK_GATE	R/W	0	PWM_PLUS0 Module Clock Gating
19	RESERVED	R	0	reserved bit
18	PWM_BASE1_CLK_GATE	R/W	0	PWM_BASE1 Module Clock Gating
17	PWM_BASE0_CLK_GATE	R/W	0	PWM_BASE0 Module Clock Gating
16	TIMER_PLUS1_CLK_GATE	R/W	0	TIMER_PLUS1 Module Clock Gating
15	TIMER_PLUS0_CLK_GATE	R/W	0	TIMER_PLUS0 Module Clock Gating
14	TIMER_BASE2_CLK_GATE	R/W	0	TIMER_BASE2 Module Clock Gating
13	TIMER_BASE1_CLK_GATE	R/W	0	TIMER_BASE1 Module Clock Gating
12	TIMER_BASE0_CLK_GATE	R/W	0	TIMER_BASE0 Module Clock Gating
11	SPI1_CLK_GATE	R/W	0	SPI1 Module Clock Gating
10	SPI0_CLK_GATE	R/W	0	SPI0 Module Clock Gating
9	RESERVED	R	0	reserved bit
8	UART2_CLK_GATE	R/W	0	UART2 Module Clock Gating
7	UART1_CLK_GATE	R/W	0	UART1 Module Clock Gating
6	UART0_CLK_GATE	R/W	0	UART0 Module Clock Gating
5	IIC1_CLK_GATE	R/W	0	IIC1 Module Clock Gating

IIC0 Module Clock Gating				
4	IICO_CLK_GATE	R/W	0	IICO Module Clock Gating
3	RESERVED	R	0	reserved bit
2	GPIOC_CLK_GATE	R/W	0	GPIOC Module Clock Gating
1	GPIOB_CLK_GATE	R/W	0	GPIOB Module Clock Gating
0	GPIOA_CLK_GATE	R/W	0	GPIOA Module Clock Gating

RC_FREQ_DELTA register (0x78)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31	RCHF_SIG	R/W	0	1: Indicates that RCHF_DELTA is positive. 0: Indicates that RCHF_DELTA is negative.
30:11	RCHF_DELTA	R/W	0	RCHF Difference between actual test frequency and 48MHz Note: The true frequency is the sum of 48MHz and the difference.
10	RCLF_SIG	R/W	0	1: Indicates that RCLF_DELTA is positive. 0: Indicates that RCLF_DELTA is negative.
9:0	RCLF_DELTA	R/W	0	RCLF Difference between actual test frequency and 32.768KHz Note: The true frequency is the sum of 32.768KHz and the difference.

VREF_VOLT_DELTA register (0x7C)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:7	RESERVED	R	0	reserved bit

6	VREF_SIG	R/W	0	1: Indicates that VREF_DELTA is positive. 0: Indicates that VREF_DELTA is negative.
5:0	VREF_DELTA	R/W	0	VREF Difference between actual test reference voltage value and theoretical value (in mv) Note: The true voltage value is the sum of the theoretical value and the difference.

CHIP_ID0 register (0x80)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:0	CHIP_ID0	R/W	0	Device ID Register 0

CHIP_ID1 register (0x84)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:0	CHIP_ID1	R/W	0	Device ID Register 1

CHIP_ID2 register (0x88)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:0	CHIP_ID2	R/W	0	Device ID Register 2

CHIP_ID3 register (0x8C)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:0	CHIP_ID3	R/W	0	Device ID Register 3

PLL_CTRL register (0x180)

DP32G030
Reference Manual

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:11	RESERVED	R	0	reserved bit	Reference Manual
10:6	PLL_M	R/W	0xb	PLL Reference Clock Divider 00000:1 crossover frequency 00001: 2 crossover frequency 00010: 3 crossover frequency 00011: 4 crossover frequency 11110:31 Crossover frequency 11111: 32 crossover frequency	
5:1	PLL_N	R/W	0x4	PLL Feedback Clock Division 00000: 2 crossover frequency 00001: 4 crossover frequency 00010: 6 crossover frequency 00011: 8 crossover frequency 11110: 62 crossover frequency 11111: 64 crossover frequency	
0	PLL_EN	R/W	0	PLL enable control bit 0: Turn off PLL 1: Enable PLL Note: The PLL will not lock until at least 30us after it is turned on.	

PLL_ST register (0x184)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:1	RESERVED	R	0	reserved bit
0	PLL_LOCK	R	0	PLL Lock Status Bit 0: not locked 1: Locked, PLL clock can be used

5.7 IO Function Configuration (PORTCON)

5.7.1 summarize

Each IO can be independently configured with its corresponding function, and the PORTx_SELx register can be used to select which function the IO is multiplexed to, and the corresponding registers can be configured into multiple modes by software settings according to the specific hardware characteristics of each IO port listed in the manual.

5.7.2 characterization

- Each IO can be configured for a specific digital function depending on which digital function it is multiplexing
- Each IO can be configured for a specific analog function depending on which analog function it is multiplexed with
- Independent input pull-up enable control
- Independent input pull-down enable control
- Independent open-drain or push-pull output modes
- Independent input enable control
- Independent IO wake-up enable
- Specific IO wake-up active edge selection control
- Input hysteresis selection control
- Output drive capability selection control
- Input pull-up resistor resistance value selection control

Most of the above functions can be freely programmed independently for each IO port, while some function controls are global and work for all IOs, so select and

DP32G030

use them according to the corresponding function register description in the [Reference Manual](#)
configuration.

5.7.3 Block Diagram of Module Structure

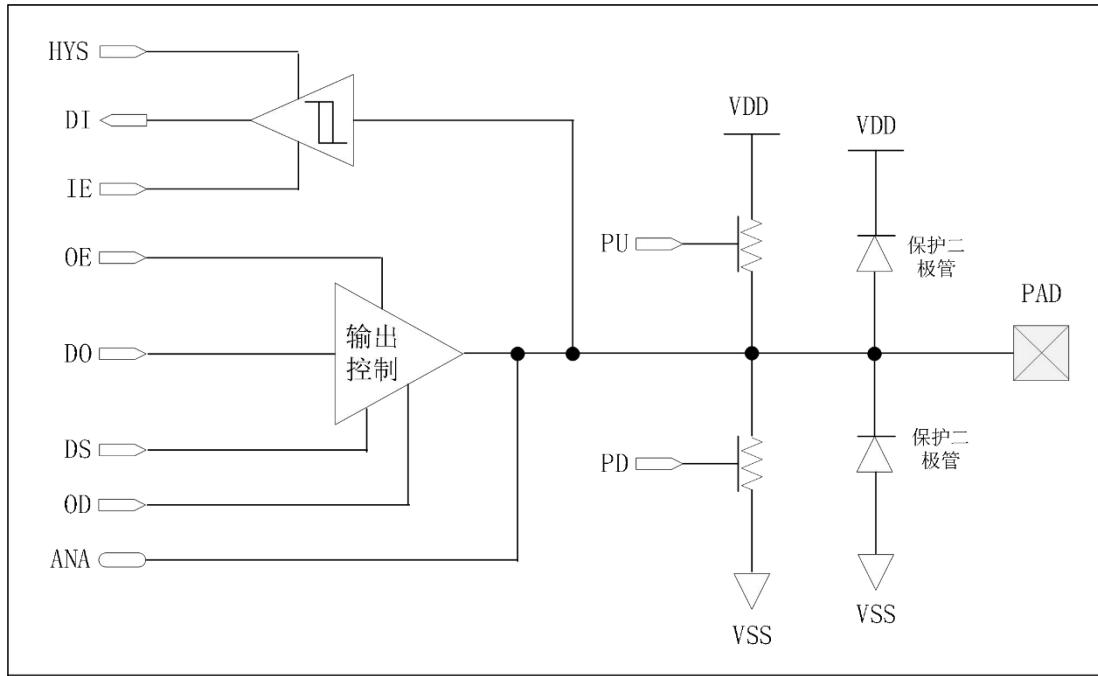


Figure 5-14 PORTCON Module Block Diagram

Among them:

HYS is an input hysteresis selection console that can be configured uniformly from specific registers;

IE is the input enable control terminal, which can be configured directly from the separate registers of each IO;

DS is the output drive capability control terminal, which can be uniformly configured by specific registers;

OD is the output mode control terminal, which can be directly configured by the separate registers of each IO;

PU is the input pull-up enable control, which can be directly configured by each independent register of IO; PD

is the input pull-down enable control, which can be directly configured by each independent register of IO; ANA

is the analog signal channel, which is directly connected to the analog signal defined by each IO;

DI is the input signal terminal which is input to the chip [Reference Manual](#) used as an input signal for the corresponding function by digital function multiplexing selection;

DO is the output signal terminal which is output to PAD from the chip. It can be used as the output signal of the corresponding function by digital function multiplexing selection;

OE is the output enable console. It is co-located with DO, and when this IO is configured to function as an output, then the hardware will The OE is turned on automatically, otherwise the OE is turned off;

5.7.4 Functional Description

Pin Input Enable

If this chip is used as an input or a peripheral that requires an input, the input enable register of the corresponding IO must be set.

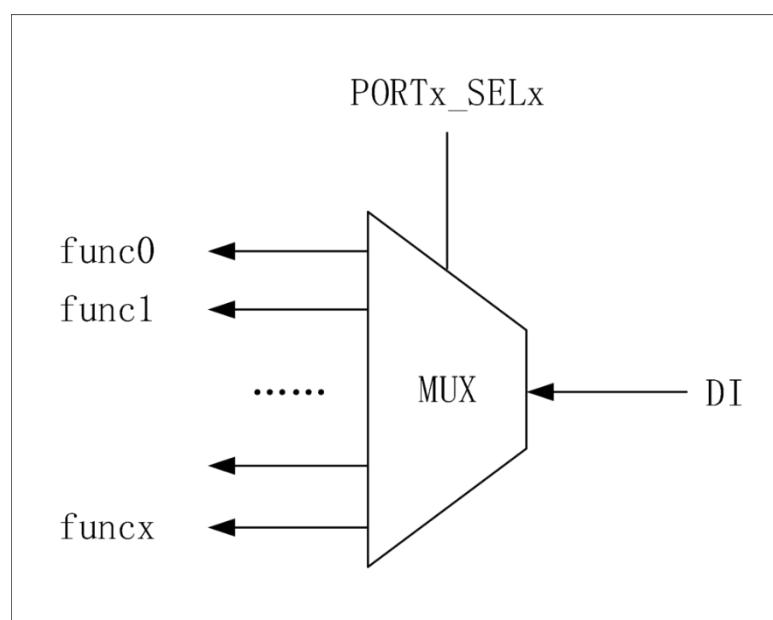
(PORTx_IE) is configured as valid. When the input enable register corresponding to IO is configured to 1, i.e., IE is valid, the input enable is turned on, and the level state of PAD can be input into the internal chip to the DI terminal to obtain the current external state of the chip.

Digital multiplexing function selection

Different IOs can be multiplexed into different digital functions, such as GPIO, SPI, UART, etc. The direction of the IOs depends on the specific functional requirements of the multiplexing. The direction of IOs depends on the specific function requirements of multiplexing. The function to be realized by each IO can be directly configured by PORTx_SELx register.

- Enter a description of the function:

The input function selection schematic is shown below:



func0, func1,, funcx, etc. are digital signals input to the internal chip.

Configure an IO as a specific digital function through **PORTx_SELx** register, if the function is an input, then use **PORTx_SELx** to select the DI signal to be input to one of the func signals, and the other func signals will be at level 0. Through **PORTx_SELx**, the path from DI to the specific digital signal can be opened, and when the IE of the corresponding IO is opened, the input signal from the PAD terminal will directly enter into the chip to give the selected digital signal.

- Outputs a description of the function:

The output function selection schematic is shown below:

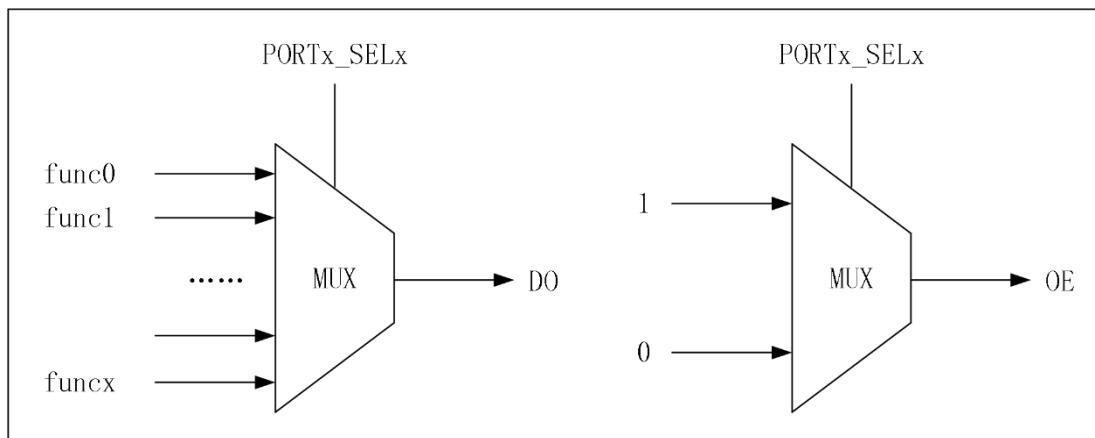


Figure 5-16 IO Output Function Selection Diagram

func0, func1,, funcx, etc. are internal digital signals of the chip to be output.

Configure an IO as a specific digital function through **PORTx_SELx** register, if the function is output, the selected digital signal will be output to DO through **PORTx_SELx** selection, other func signals will not be selected for output, and the OE terminal of the corresponding IO is configured to be active. It can realize to output the internal digital signal level to the off-chip PAD for off-chip acquisition.

Analog multiplexing function selection

When a specific IO is selected as analog via the **PORTx_SELx** register, the OE side of the IO is directly turned off, and the IE side of the IO needs to be turned off by configuring the corresponding Input Enable register (**PORTx_IE**) to zero. In

addition, in order to ensure the integrity of the analog signal ~~Referenced Manual~~ ~~PAD~~ and not to be interfered, the software needs to ensure that the pull-up and pull-down resistors of the IO are invalidated.

(Controlled by turning off the pull-up enable (PORTx_PU) and pull-down enable (PORTx_PD) of the corresponding IO.

The following is an example to illustrate the connection and control relationship between analog signals and IOs.
The schematic diagram of the analog signal connection to the IO is shown below:

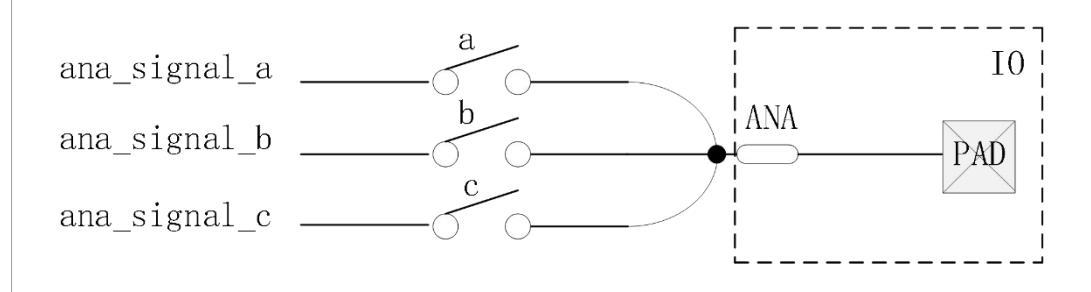
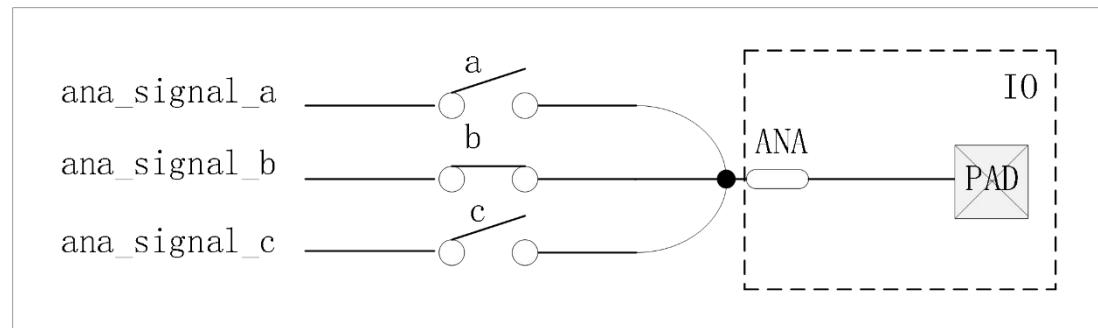


Figure 5-17 Analog Signal Connection IO Schematic

In the figure above, `ana_signal_a`, `ana_signal_b`, and `ana_signal_c` are three analog signals multiplexed to the same `IO`. The three analog signals are controlled by their respective analog switches `a`, `b`, and `c`, which are controlled by different modules and are not uniformly controlled by the module. When this `IO` does not multiplex the analog signals, the switches are disconnected and the analog signals are not connected to the `ANA` side, so that the levels and behavior on the `PAD` do not affect the analog signals inside the chip.

When it is necessary to multiplex the `IO` to one of the analog functions, the corresponding switch is closed so that this analog signal can be connected to the



`ANA` side of the `IO`. As shown in the figure below, the `IO` is multiplexed to the function of the `ana_signal_b` analog signal.

Figure 5-18 Functional Schematic of IO Multiplexing to `ana_signal_b` Analog Signal

Pull-up/down resistor enable

All `IOs` on this chip have independent input pull-ups or input pull-downs that can be configured.

As shown in the basic structure schematic of the `IO` port, the pull-up resistor

is controlled through the PU port and the pull-down resistor [Refer to the Datasheet](#) through the PD port.

When the IE terminal of an IO is configured to be valid through the input enable register (PORTx_IE), the PU port of that IO is set to 1 by configuring the corresponding pull-up resistor enable register (PORTx_PU) to be 1, then the DI terminal is at logic level 1 even if the PAD is in the suspended state.

When the IE side of an IO is configured to be valid through the input enable register (**PORTx_IE**), and the PD port of that IO is set to 1 by configuring the corresponding pull-down resistor enable register (**PORTx_PD**) to 1, the DI side will be at logic level 0 even if the PAD is in the suspended state.

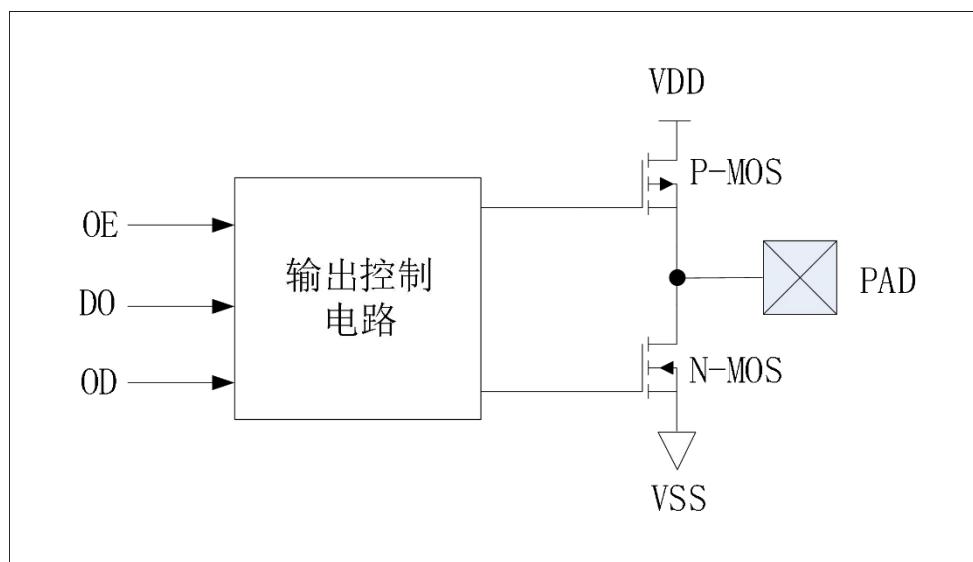
Note 0: The pull-up enable and pull-down enable of the same IO cannot be configured as active at the same time. In this chip, if both IOs are active at the same time, the pull-down resistor will work and the pull-up resistor will be invalid.

Output Mode Selection

All IOs of this chip have independent output mode selection control, which can be independently configured as output open-drain mode or output push-pull mode.

When the IO is used as an output, OE is high, and the output state of the PAD can be configured to either push-pull or open-drain mode by configuring the open-drain enable register (**PORTx_OD**) to be 0 or 1.

When the IO's open drain enable register (**PORTx_OD**) is configured to 0, it is in push-pull output mode, where the IO has the ability to pull/pour current. This is



shown in the figure below:

Figure 5-19 IO Push-Pull Output Mode Schematic

DP32G030
Reference Manual

When **OD = 0** and **OE = 1**, PAD is strong 0 when DO is 0 and strong 1 when DO is 1.

When the IO's open-drain enable register (**PORTx_OD**) is configured to 1, it is in open-drain output mode, in which case the IO only has the ability to pour current, not to pull current. As shown in the figure below:

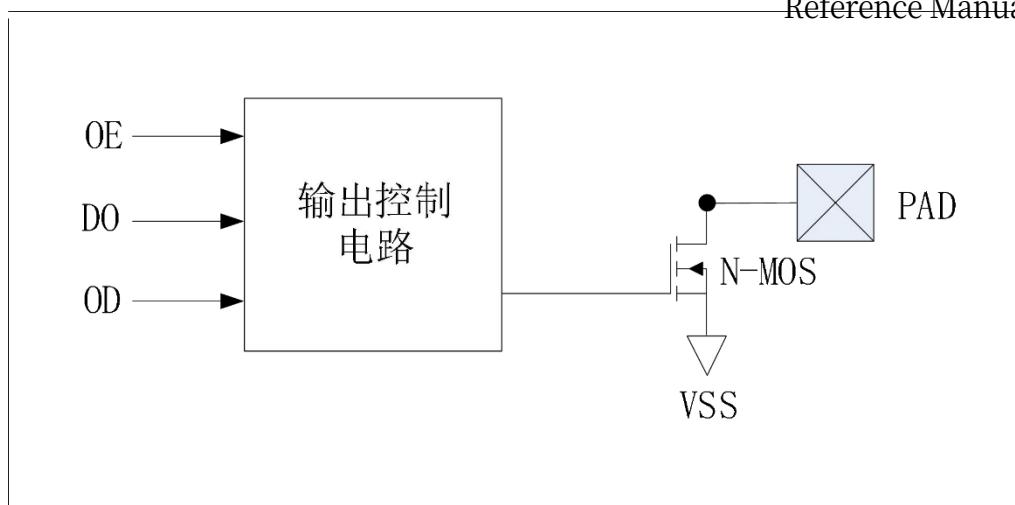


Figure 5-20 IO Open-Drain Output Mode Schematic

When $OD = 1$ and $OE = 1$, when DO is 0, PAD is strong 0, when DO is 1, PAD is high resistance state, if you need to output high level, you need to connect the external pin pull-up resistor to the power supply, and realize the high level output through external pull-up.

Pin Wake-Up Function

The chip supports pin wake-up function in low-power mode. Independent wake-up control is provided for each IO, which is used to provide flexible and convenient IO wake-up methods in the use of the chip.

Each IO can be independently configured to act as a wake-up pin or not through the wake-up enable register (`PORTx_WKE`). And can support wake-up edge or level configurability.

All IOs can be independently configured for rising edge wakeup or falling edge wakeup through the `PORT_RIS` register.

Note 0: In this chip, RIS is defined to be active low when configured as 0 and active rising edge when configured as 1;

Output drive capability selection

Different groups of IOs have independent output drive capability configuration control, which can be selected through their respective `PORT_DS` registers.

DP32G030

Generally speaking, the output drive capability of IOs can Reference Manual the following four modes: 5mA, 14mA, 22mA and 30mA. Note: This register is the global control register of IOs.

Input hysteresis selection

This IO provides two kinds of input hysteresis block selection control, users can choose and configure reasonably according to the system condition and the level of external signal.

Input hysteresis can be controlled through the PORT_HYS register. When this register is configured to 0, a low input hysteresis configuration can be selected; when this register is configured to 1, a high input hysteresis configuration can be selected.

Note: This register is the global control register for the IO. **Pull-up**

Resistor Value Selection

For customer's convenience, this IO provides the option of configurable pull-up resistor resistance value so that the user can choose a more reasonable pull-up resistor.

The pull-up resistor resistance value is configurable through the PORT_PUR register. Three resistance values are available: 32kΩ, 40kΩ, and 150kΩ.

Note: This register is the global control register for the IO.

Register Mapping Register Mapping

name (of a thing)	offset	bit width	typology	reset value	descriptive
PORTBASE: 0x400B0000					
PORTA_SEL0	0x00	32	R/W	0x00	PORTA Function Selection Register 0
PORTA_SEL1	0x04	32	R/W	0x00	PORTA Function Selection Register 1

DP32G030

PORTB_SEL0	0x08	32	R/W	0xffff0000	PORTB Reference Manual Register 0
PORTB_SEL1	0x0C	32	R/W	0x01001000	PORTB Function Selection Register 1
PORTC_SEL0	0x10	32	R/W	0x00	PORTC function selection register 0
PORTA_IE	0x100	32	R/W	0x00	PORTA Input Enable Register
PORTB_IE	0x104	32	R/W	0x4800	PORTB Input Enable Register
PORTC_IE	0x108	32	R/W	0x20	PORTC Input Enable Register
PORTA_PU	0x200	32	R/W	0x00	PORTA Pull-up Enable Register
PORTB_PU	0x204	32	R/W	0x00	PORTB Pull-up Enable Register
PORTC_PU	0x208	32	R/W	0x00	PORTC Pull-up Enable Register
PORTA_PD	0x300	32	R/W	0x00	PORTA Pull-down Enable Register
PORTB_PD	0x304	32	R/W	0x00	PORTB Pull-down Enable Register
PORTC_PD	0x308	32	R/W	0x20	PORTC Pull-down Enable Register
PORTA_OD	0x400	32	R/W	0x00	PORTA Open Drain Enable Register
PORTB_OD	0x404	32	R/W	0x00	PORTB Open Drain Enable Register
PORTC_OD	0x408	32	R/W	0x00	PORTC Open Drain Enable Register
PORTA_WKE	0x500	32	R/W	0x00	PORTA Wake-up Enable Register
PORTB_WKE	0x504	32	R/W	0x00	PORTB Wake-up Enable Register
PORTC_WKE	0x508	32	R/W	0x00	PORTC Wake-up Enable Register

					Register Reference Manual
PORT_CFG	0x600	32	R/W	0x15	PORT Configuration Register
PORTA_WK_SEL	0x700	32	R/W	0x00	PORTA Wake-up IO Along Select Register
PORTB_WK_SEL	0x704	32	R/W	0x00	PORTB Wake-up IO Along Select Register
PORTC_WK_SEL	0x708	32	R/W	0x00	PORTC Wake-Up IO Along Select Register

register description

PORTA_SEL0 register (0x00)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:28	PORTA7	R/W	0	0000: GPIOA7 0001: UART1_TX 0010: timerp0_in0 0011: timerp0_out_l 0100: SARADC_CH2 0101: OPA0_VP Other: Reserved

				Reference Manual
27:24	PORTA6	R/W	0	0000: GPIOA6 0001: UART1_RTS 0010: TIMERP1_IN1 0011: timerp1_out_h 0100: SARADC_CH1 0101: OPA0_OUT Other: Reserved
23:20	PORTA5	R/W	0	0000: GPIOA5 0001: UART1_CTS 0010: PWMP1_PLUS1 0011: TIMERP1_IN0 0100: timerp1_out_l 0101: WAKEUP1 0110: SARADC_CH0 Other: Reserved
19:16	PORTA4	R/W	0	000: GPIOA4 001: CMPO_VP 010: XTAH_XO Other: Reserved
15:12	PORTA3	R/W	0	000: GPIOA3 001: CMPO_VN Other: Reserved
11:8	PORTA2	R/W	0	000: GPIOA2 001: XTAL_XO Other: Reserved
7:4	PORTA1	R/W	0	000: GPIOA1 001: XTAL_XI Other: Reserved

3:0	PORTAO	R/W	0	0000: GPIOA0 0001: PWMP1_PLUS0 0010: PWMP0_PLUS1 0011: TM 0100: WAKEUP0 Other: Reserved	Reference Manual
-----	--------	-----	---	--	------------------

PORTA_SEL1 register (0x04)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:28	PORTA15	R/W	0	0000: GPIOA15 0001: PWMB1_CH1 0010: PWMP0_CH0 0011: TIMERP1_IN1 0100: timerp1_out_h Other: Reserved
27:24	PORTA14	R/W	0	0000: GPIOA14 0001: PWMB1_CH0 0010: PWMP0_CH2N 0011: TIMERP1_IN0 0100: timerp1_out_l 0101: SARADC_CH9 Other: Reserved
23:20	PORTA13	R/W	0	0000: GPIOA13 0001: PWMB0_CH2 0010: PWMP0_CH1N 0011: timerp0_in1 0100: timerp0_out_h 0101: SARADC_CH8 Other: Reserved

				Reference Manual
19:16	PORTA12	R/W	0	0000: GPIOA12 0001: SPI0_MOSI 0010: PWMBO_CH1 0011: PWMPO_CH0N 0100: TIMERPO_IN0 0101: timerp0_out_l 0110: SARADC_CH7 Other: Reserved
15:12	PORTA11	R/W	0	0000: GPIOA11 0001: SPI0_MISO 0010: PWMBO_CH0 0011: PWMPO_BRAKE0 0100: TIMERP1_IN1 0101: timerp1_out_h 0110: SARADC_CH6 Other: Reserved
11:8	PORTA10	R/W	0	0000: GPIOA10 0001: SPI0_CLK 0010: SARADC_CH5 0011: CMP1_VP Other: Reserved
7:4	PORTA9	R/W	0	0000: GPIOA9 0001: SPI0_SSN 0010: TIMERP1_IN0 0011: timerp1_out_l 0100: TM 0101: SARADC_CH4 0110: CMP1_VN Other: Reserved

3:0	PORTA8	R/W	0	0000: GPIOA8 0001: UART1_RX 0010: timerp0_in1 0011: timerp0_out_h 0100: SARADC_CH3 0101: OPA0_VN Other: Reserved	Reference Manual
-----	--------	-----	---	--	------------------

PORTB_SEL0 register (0x08)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:28	PORTB7	R/W	0xf	0000: GPIOB7 0001: SPI0_SS_N 0010: UART0_TX 0011: IIC0_SCL 0100: PWMP1_BRAKE0 0101: PWMPO_CH1 Other: Reserved
27:24	PORTB6	R/W	0xf	0000: GPIOB6 0001: PWMPO_CH0 0010: timerp0_in1 0011: timerp0_out_h Other: Reserved
23:20	PORTB5	R/W	0xf	0000: GPIOB5 0001: SPI1_MOSI 0010: PWMP1_CH0_N 0011: PWMPO_CH2_N 0100: TIMERPO_IN0 0101: timerp0_out_l Other: Reserved

					Reference Manual
19:16	PORTB4	R/W	0xf		0000: GPIOB4 0001: SPI1_MISO 0010: IIC1_SCL 0011: PWMP1_CH0 0100: PWMPO_CH1N 0101: timerp1_hall2 Other: Reserved
15:12	PORTB3	R/W	0		0000: GPIOB3 0001: SPI1_CLK 0010: IIC1_SDA 0011: PWMPO_CH0N 0100: TIMERP1_HALL1 Other: Reserved
11:8	PORTB2	R/W	0		0000: GPIOB2 0001: SPI1_SS_N 0010: PWMPO_BRAKE1 0011: TIMERP1_HALL0 Other: Reserved
7:4	PORTB1	R/W	0		0000: GPIOB1 0001: UART2_RX 0010: IIC0_SDA 0011: PWMPO_CH2 Other: Reserved
3:0	PORTB0	R/W	0		0000: GPIOB0 0001: UART2_TX 0010: IIC0_SCL 0011: PWMB1_CH2 0100: PWMPO_CH1 Other: Reserved

PORTE_SEL1 Register (0x0C)

bitfield	name (of)	typolo	reset	descriptive

DP32G030

(math.)	a thing)	gy	value	
---------	----------	----	-------	--

					Reference Manual
31:28	PORTB15	R/W	0	0000: GPIOB15 0001: SPI1_SS_N 0010: UART2_RX Other: Reserved	
27:24	PORTB14	R/W	0x1	0000: GPIOB14 0001: SWCLK 0010: UART2_TX 0011: PWMP1_CH2N Other: Reserved	
23:20	PORTB13	R/W	0	0000: GPIOB13 0001: UART1_RX 0010: IIC1_SDA 0011: PWMP1_CH1N Other: Reserved	
19:16	PORTB12	R/W	0	0000: GPIOB12 0001: UART1_TX 0010: IIC1_SCL 0011: PWMP1_CH0N Other: Reserved	
15:12	PORTB11	R/W	0x1	0000: GPIOB11 0001: SWDIO 0010: PWMP1_CH2 0011: PWMPO_BRAKE2 Other: Reserved	
11:8	PORTB10	R/W	0	0000: GPIOB10 0001: SPI0_MOSI 0010: UART0_RTS 0011: PWMBO_CH2 0100: PWMP1_CH1 0101: PWMPO_PLUS0 0110: timerp1_in0 0111: timerp1_out_l Other: Reserved	

				Reference Manual
7:4	PORTB9	R/W	0	0000: GPIOB9 0001: SPI0_MISO 0010: UART0_CTS 0011: PWMB0_CH1 0100: PWMP1_CH0 0101: timerp1_in1 0110: timerp1_out_h Other: Reserved
3:0	PORTB8	R/W	0	0000: GPIOB8 0001: SPI0_CLK 0010: UART0_RX 0011: IIC0_SDA 0100: PWMB0_CH0 0101: PWMP1_BRAKE1 0110: PWMP0_CH2 Other: Reserved

PORTC_SEL0 register (0x10)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:28	PORTC7	R/W	0	0000: GPIOC7 0001: IIC1_SDA 0010: PWMP1_CH2 0011: TIMERP1_IN0 0100: timerp1_out_l 0101: OPA1_OUT Other: Reserved

27:24	PORTC6	R/W	0	0000: GPIOC6 0001: IIC1_SCL 0010: PWMP1_CH1 0011: TIMERP1_IN1 0100: timerp1_out_h 0101: OPA1_VN Other: Reserved
-------	--------	-----	---	---

				Reference Manual
23:20	PORTC5	R/W	0	0000: GPIOC5 0001: timerp0_hall2 0010: TM 0011: OPA1_VP Other: Reserved
19:16	PORTC4	R/W	0	0000: GPIOC4 0001: UART0_RX 0010: IIC0_SDA 0011: PWMP1_CH2N 0100: TIMERPO_HALL1 0101: CMP2_VP Other: Reserved
15:12	PORTC3	R/W	0	0000: GPIOC3 0001: UART0_TX 0010: IIC0_SCL 0011: PWMP1_CH1N 0100: TIMERPO_HALLO 0101: CMP2_VN Other: Reserved
11:8	PORTC2	R/W	0	0000: GPIOC2 0001: SPI1_MOSI 0010: PWMB1_CH2 0011: PWMP1_BRAKE2 0100: TIMERPO_IN1 0101: timerp0_out_h Other: Reserved
7:4	PORTC1	R/W	0	0000: GPIOC1 0001: SPI1_MISO 0010: UART2_RTS 0011: PWMB1_CH1 0100: TIMERPO_IN0 0101: timerp0_out_l Other: Reserved

3:0	PORTC0	R/W	0	0000: GPIOC0 0001: SPI1_CLK 0010: UART2_CTS 0011: PWMB1_CH0 Other: Reserved	Reference Manual
-----	--------	-----	---	---	------------------

PORTA_IE register (0x100)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTA_IE	R/W	0x00	PORTA Input Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to A0, bit1 corresponds to A1)

PORTB_IE register (0x104)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTB_IE	R/W	0x4800	PORTB Input Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to B0, bit1 corresponds to B1)

PORTC_IE register (0x108)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit

7:0	PORTE_IE	R/W	0x20	PORTE Input Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to C0, bit1 corresponds to C1)
-----	----------	-----	------	---

PORTA_PU register (0x200)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTA_PU	R/W	0	PORTA Pull-up Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to A0, bit1 corresponds to A1)

PORTB_PU register (0x204)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTB_PU	R/W	0	PORTB Pull-up Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to B0, bit1 corresponds to B1)

PORTC_PU register (0x208)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	PORTC_PU	R/W	0	PORTC Pull-up Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to C0, bit1 corresponds to C1)

PORTA_PD register (0x300)

DP32G030

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit

15:0	PORTA_PD	R/W	0	PORTA Pull-down Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to A0, bit1 corresponds to A1)
------	----------	-----	---	---

PORTB_PD register (0x304)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTB_PD	R/W	0	PORTB Pull-down Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to B0, bit1 corresponds to B1)

PORTC_PD register (0x308)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	PORTC_PD	R/W	0x20	PORTC Pull-down Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to C0, bit1 corresponds to C1)

PORTA_OD register (0x400)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit

PORTA Open Drain Enable Register

0: disable 1: Enable

(each bit corresponds to 1 IO, bit0 corresponds to A0, bit1 corresponds to A1)

15:0	PORATA_OD	R/W	0	
------	-----------	-----	---	--

PORTB_OD register (0x404)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTB_OD	R/W	0	PORTB Open Drain Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to B0, bit1 corresponds to B1)

PORTC_OD register (0x408)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	PORTC_OD	R/W	0	PORTC Open Drain Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to C0, bit1 corresponds to C1)

PORTA_WKE register (0x500)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTA_WKE	R/W	0	PORTA Wake-up Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to A0, bit1 corresponds to A1)

PORTB_WKE register (0x504)

DP32G030

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit

PORTB Wake-up Enable Register Reference Manual				
15:0	PORTB_WKE	R/W	0	0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to B0, bit1 corresponds to B1)

PORTC_WKE register (0x508)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	PORTC_WKE	R/W	0	PORTC Wake-up Enable Register 0: disable 1: Enable (each bit corresponds to 1 IO, bit0 corresponds to C0, bit1 corresponds to C1)

PORT_CFG register (0x600)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:11	RESERVED	R	0	reserved bit
10	PORT_HYS	R/W	0	PORT Input hysteresis level selection 0: low input hysteresis (input signal greater than 0.7VDD and less than 0.3VDD) 1: High input hysteresis (input signal greater than 0.85VDD and less than 0.15VDD)
9:6	RESERVED	R	0	reserved bit
5:4	PORTC_DS	R/W	0x01	PORTC Driveability Selection Register 00: 5mA 01: 10mA 10: 15mA 11: 20mA

DP32G030

PORTB Drive capability selection register Preference Manual				
3:2	PORTB_DS	R/W	0x01	PORTB Drive capability selection register Preference Manual

PORTA Driveability Selection Register Reference Manual				
1:0	PORTA_DS	R/W	0x01	PORTA Driveability Selection Register
				00: 5mA
				01: 10mA
				10: 15mA
				11: 20mA

PORTA_WK_SEL register (0x700)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTA_WK_SEL	R/W	0	PORTA wake-up function is configured along 0: PORTA wake-up function active on falling edge 1: PORTA Wake-up function rising edge active

PORTB_WK_SEL register (0x704)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PORTB_WK_SEL	R/W	0	PORTB wake-up function is configured along 0: PORTB wake-up function active on falling edge 1: PORTB Wake-up function rising edge active

PORTC_WK_SEL register (0x708)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit

PORTC wake-up function configuration				
7:0	PORTC_WK_SEL	R/W	0	PORTC wake-up function configured
				0: PORTC wake-up function active on falling edge
				1: PORTC wake-up function active on rising edge

5.8 General Purpose IO (GPIO)

5.8.1 summarize

The GPIO module realizes a general-purpose programmable IO interface, supports two modes of configurable input and output, and can be used for serial data communication. The clock of the corresponding GPIO module must be enabled before use. The system schematic is shown below:

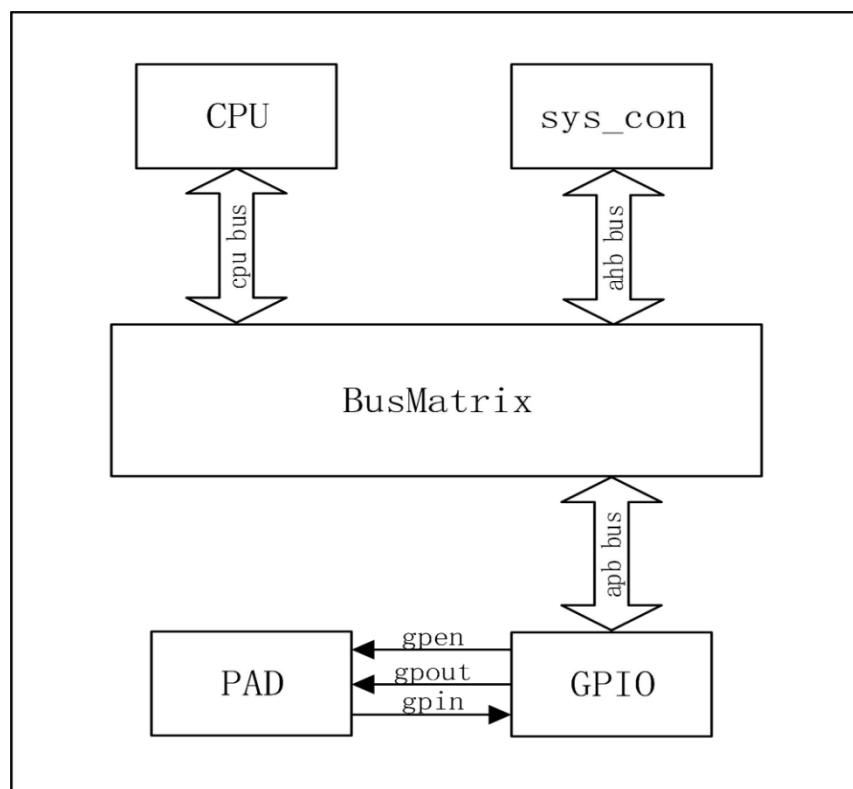


Figure 5-21 GPIO Module System Block Diagram

5.8.2 characterization

- Up to 40 independent IOs
- Interrupt entry for each IO
- Configurable interrupt trigger conditions, supports level trigger and edge trigger
- Level trigger support for high and low levels

- Edge triggering supports rising edge, falling edge and dual edge triggering.

5.8.3 Block Diagram of Module Structure

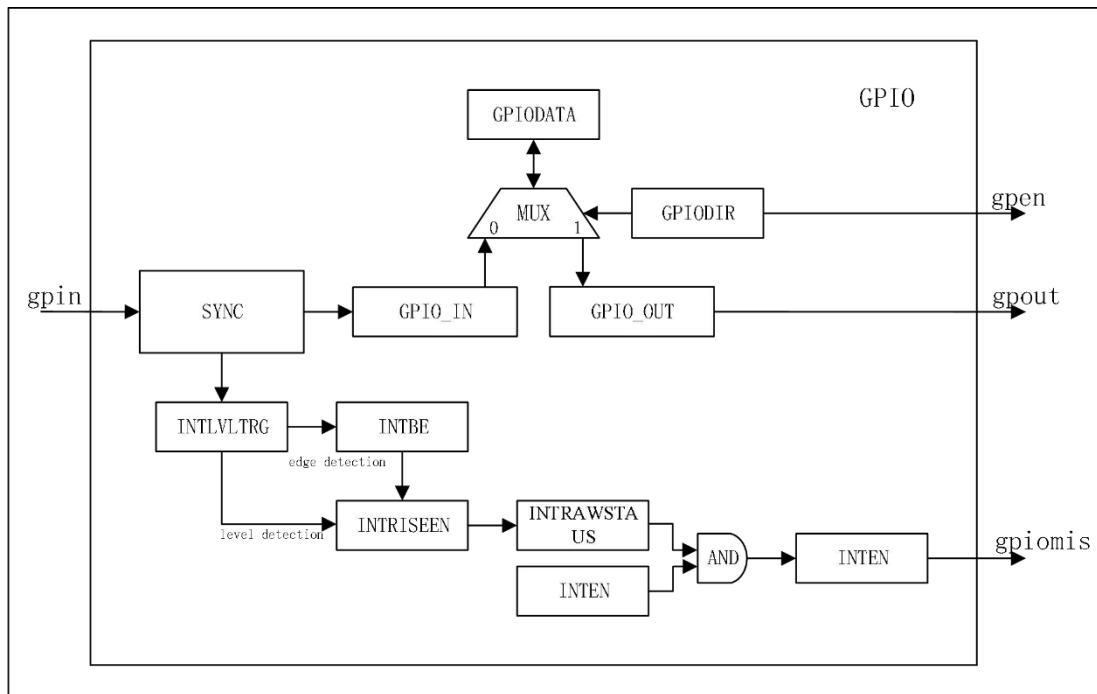


Figure 5-22 GPIO Module Block Diagram

The above figure shows the internal structure of the GPIO module. As shown in the above figure, the external input data `gpin` is processed by the synchronization circuit `sync` to detect along the trigger interrupt and also used for reading the input data from the APB bus. `gpin` signal generates the interrupt status `INTRAWSTAUS` through different interrupt settings. `INTRAWSTAUS` signal outputs the interrupt flag signal `gpiomis` to the system after controlled by the interrupt enable register. The `gpiomis` signal is controlled by the interrupt enable register and outputs the interrupt flag signal to the system. The clock source in this module is `pclk`, which is the same source as the system clock, and can be configured through the `DEV_CLK_GATE` register in the `SYSCON` module to enable the clock of this module.

5.8.4 Functional Description

directional control

- The default state of all pins after power-up is **GPIO float**. Refer to the **GPIO Reference Manual** for the SWD pin.
- The **GPIO Direction Register (DIRx)** is used to configure each individual pin to input mode or output mode
- When the data direction is set to 0, the corresponding **GPIO** pin is configured as input.

Get the current status value of the specified **GPIO port** by reading the corresponding bit of the corresponding data register (**DATA**).

- When the data direction is set to 1, the corresponding **GPIO** pin is configured as output.

The specified pin output is changed by writing a value to the corresponding bit of the corresponding port data register (**DATA**). 0 outputs a low level.

1 Output high.

The output timing of the **GPIO** is shown below:

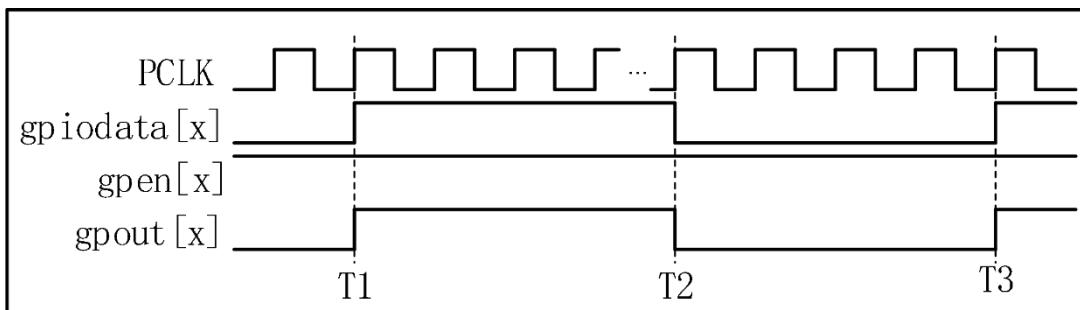


Figure 5-23 GPIO Output Timing Chart

Set the corresponding **GPIO** bit of the **GPIO** module as an output pin.

T1 time, write **gpiodata[x]** value is 1, **GPIO** output **gpout[x]** becomes high. T2 time, write **gpiodata[x]** value is 0, **GPIO** output **gpout[x]** becomes low. At time T3, write **gpiodata[x]** value to 1, and **GPIO** output **gpout[x]** becomes high. **GPIO** output **gpout[x]** changes with the change of **gpiodata[x]**.

Interrupt Configuration and Clearing

The corresponding pins of **GPIO port** can be configured as interrupt mode according to the requirement, and the interrupt polarity and trigger mode can be configured through the related registers.

Trigger mode is divided into two modes: edge trigger and level trigger.

- For edge-triggered interrupts, they can be set as rising-edge-triggered, falling-edge-triggered or double-edge-triggered. After an interrupt occurs, the flag bit has a hold characteristic and must be cleared by software.
- For level-triggered interrupts, the interrupt occurs when the external pin input

is the specified level. When the level is flipped, the interrupt signal polarity does not change without software clearing. When using level-triggered interrupts, it is necessary to ensure that the external signal source remains at a stable level so that the valid interrupt level can be recognized by the port.

Use the following registers to define the generation interrupt trigger method and polarity:

- GPIO Interrupt Trigger Mode Register (**INTLVLTRG**) to configure level triggering or edge triggering

- GPIO Interrupt Trigger Polarity Register (**INTRISEEN**) to configure the level or edge trigger polarity
- GPIO Interrupt Edge Trigger Configuration Register (**INTBE**) used to configure single-edge trigger or dual-edge trigger when selected as edge trigger

The GPIO interrupt enable register (**INTEN**) can enable or disable the interrupt of the corresponding bit of the corresponding port, and the GPIO raw interrupt state (**INTRAWSTATUS**) is not affected by the enable bit. The GPIO raw interrupt state (**INTRAWSTATUS**) is not affected by the enable bits.

(**RAWINTSTATUS**) gets the status of the interrupt signal. When the corresponding bit of the interrupt enable register (**INTEN**) is 1, the corresponding interrupt signal can be read from the interrupt status(**INTSTATUS**) register, and the interrupt signal will enter the interrupt configuration module and the NVIC module to execute the interrupt program.

The corresponding bit interrupt can be cleared by writing a 1 to the bit specified in the GPIO Interrupt Clear Register (**INTCLR**).

1. High level trigger interrupt timing

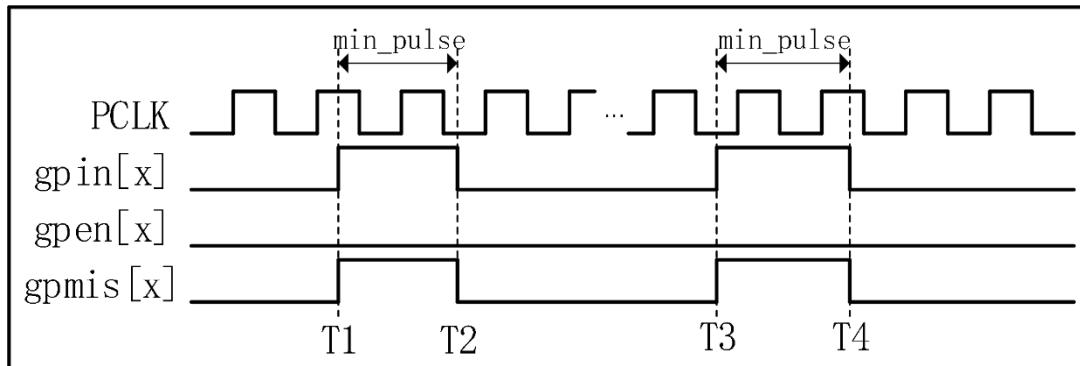


Figure 5-24 GPIO High Level Interrupt Timing Chart

The time parameters in the figure are described below:

Table 5-8 GPIO High Level Trigger Interrupt Timing Parameters

Parameter	descriptive	min
min_PLUS	Minimum pulse width of interrupt signal INTRAWSTATUS that can be handled by the interrupt module	>=1pclk

Set the corresponding GPIO bits of the GPIO module as input and enable interrupt. When τ_1 time, `gpin[x]` inputs high level signal, `INTSTATUS [x]` of `INTSTATUS` register will be set to 1 by hardware, and at the same time, `INTSTATUS [x]` interrupt signal will be output to the system. When τ_2 time, `gpin[x]` inputs a low level signal, `INTSTATUS [x]` of `INTSTATUS` register is cleared 0 by hardware, and hardware clears `INTSTATUS [x]` interrupt signal.

Note that since the interrupt signal INTSTAUS needs to be processed by the interrupt module in the system, there is a minimum limit on the width of the high level pulse of the gpin signal: min_PLUS width requires at least one pclk clock cycle. If an interrupt is masked when a high level occurs, the interrupt is ignored and the corresponding interrupt flag signal INTSTAUS is not generated. in addition, since the GPIO module does not latch on interrupts triggered by a high level, the interrupt flag is ignored if the interrupt signal generated by a high level interrupt event disappears at high level before the interrupt module can recognize it.

2. Low level trigger interrupt timing

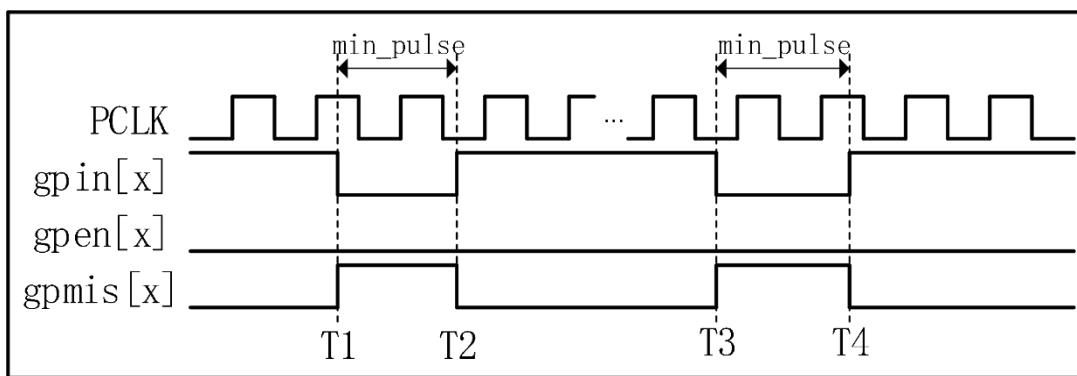


Figure 5-25 GPIO Low Level Trigger Interrupt Timing Chart

The time parameters in the figure are described below:

Table 5-9 GPIO Low Level Trigger Interrupt Timing Parameters

Parameter	descriptive	min
min_PLUS	Minimum pulse width of interrupt signal INTSTAUS that can be handled by the interrupt module	>=1pclk

Set the corresponding GPIO bits of the GPIO module as input pins, detect low level and enable interrupt. When T1 time, gpin[x] inputs low level signal, INTSTAUS [x] of INTSTAUS register will be set to 1 by hardware, and at the same time, INTSTAUS[x] interrupt signal will be output to the system. When T2 time, gpin[x] inputs a high level signal, INTSTAUS [x] of INTSTAUS register is cleared 0 by hardware, and hardware clears INTSTAUS[x] interrupt signal.

Note that since the interrupt signal INTSTAUS needs to be handled by the interrupt module in the system, there is a minimum limit on the width of the high level

pulse of the `gpin` signal: `min_PLUS` width requires at least one [Reference Manual](#) an interrupt is masked when a low level occurs, the interrupt is ignored and the corresponding interrupt flag signal `INTSTAUS` is not generated. in addition, since the `GPIO` module does not latch on interrupts triggered by a low level, the interrupt flag is ignored if the interrupt signal generated by a low level interrupt event disappears from a low level before the interrupt module is able to recognize it.

3. Rising edge triggered interrupt timing

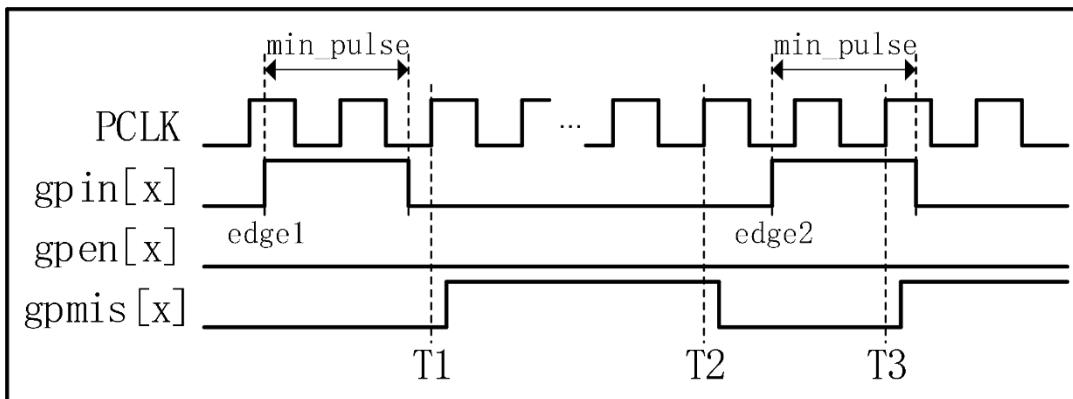


Figure 5-26 GPIO Rising Edge Trigger Interrupt Timing Diagram

The time parameters in the figure are described below:

Table 5-10 GPIO Rising Edge Triggered Interrupt Timing Parameters

Parameter	descriptive	min
min_PLUS	Minimum pulse width of interrupt signal INTSTATUS that can be handled by the interrupt module	>=1pclk

Set the corresponding GPIO bit of GPIO module as input pin, rising edge detection and enable interrupt. **edge1** time, **gp in[x]** inputs a rising edge, hardware detects the rising edge and processes it, and outputs **INTSTATUS[x]** interrupt signal after at least 2 **pclk** cycles away from the rising edge of **edge1**, see **T1** time. The interrupt is cleared by software at the **T2** time. At the **edge2** time, **gp in[x]** inputs another rising edge, the hardware detects the rising edge and processes it, and after at least 2 **pclk** cycles from the **edge2** rising edge, it outputs the **INTSTATUS[x]** interrupt signal generated by the rising edge event at the **edge2** time, see **T3** time.

Note that since the GPIO module requires rising edge detection, there is a minimum limit on the width of the high level pulse of the **gp in** signal: the **min_PLUS** width needs to be at least one **pclk** clock cycle to ensure that the GPIO module can detect the rising edge. If the interrupt is masked when the rising edge occurs, the interrupt will be ignored and the corresponding interrupt flag signal **INTSTATUS** will not be generated. in addition, since the current interrupt is cleared at **T2**, no new interrupt event (rising edge)can occur before **T2-2*Tpclk**, otherwise the interrupt flag signal **INTSTATUS** generated by the new interrupt event will be cleared at the

same time as the new interrupt flag signal INTSTAUS is cleared at Reference Manual the new interrupt flag signal INTSTAUS will be generated at the same time. cleared at the same time, and the new interrupt flag signal will be lost.

4. Falling edge triggered interrupt timing

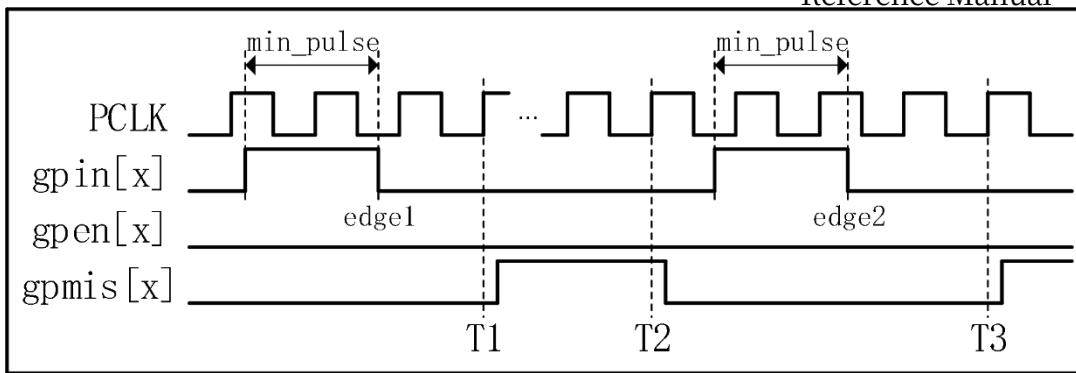


Figure 5-27 GPIO Falling Edge Triggered Interrupt Timing Diagram

The time parameters in the figure are described below:

Table 5-11 GPIO Falling Edge Triggered Interrupt Timing Parameters

Parameter	descriptive	min
min_PLUS	Minimum pulse width of interrupt signal INTSTATUS that can be handled by the interrupt module	$\geq 1\text{pclk}$

Set the corresponding **GPIO** bit of **GPIO** module as input pin, falling edge detection and enable interrupt. **edge1** time, **gpin[x]** inputs a falling edge, hardware detects the falling edge and processes it, and outputs **INTSTATUS[x]** interrupt signal after at least 2 **pclk** cycles away from the rising edge of **edge1**, see **T1** time. The interrupt is cleared by software at the **T2** time. At the **edge2** moment, **gpin[x]** inputs another falling edge, hardware detects the falling edge and processes it, and after at least 2 **pclk** cycles from the **edge2** rising edge, it outputs the **INTSTATUS[x]** interrupt signal generated by the falling edge event at the **edge2** moment, see **T3** moment.

Note that since the **GPIO** module requires falling edge detection, there is a minimum limit on the width of the low level pulse of the **gpin** signal: the **min_PLUS** width needs to be at least one **pclk** clock cycle to ensure that the **GPIO** module can detect the falling edge. If the interrupt is masked when the falling edge occurs, the interrupt will be ignored and the corresponding interrupt flag signal **INTSTATUS** will not be generated. In addition, since the current interrupt is cleared at the **T2** time, a new interrupt event (falling edge) cannot occur before the **T2-2*Tpclk** time, or the interrupt flag signal **INTSTATUS** generated by the new interrupt event will be cleared at the same time at the **T2** time, and the new interrupt flag will be cleared at the **T2** time, and the new interrupt flag will

be generated at the T2 time, cleared at the same time, and the new interrupt will be lost.

5. Double-edge triggered interrupt timing

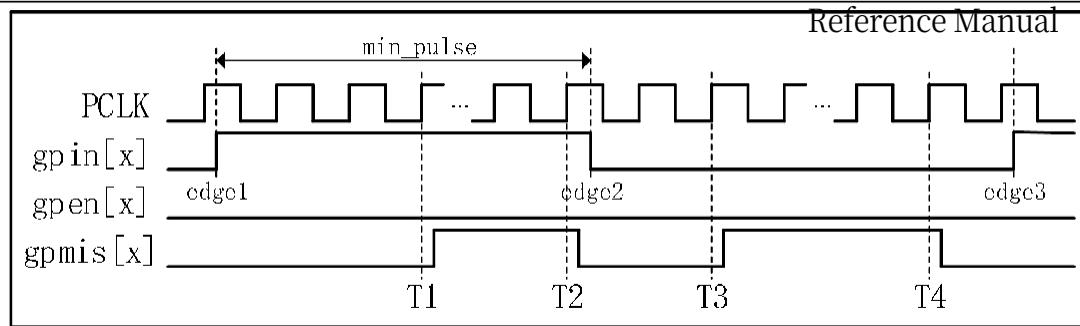


Figure 5-28 GPIO Double-Edge Trigger Interrupt Timing Chart

Set the corresponding **GPIO** bits of the **GPIO** module to input pins, double-edge detection, and enable interrupts.

At the **edge1** moment, **gpin[x]** inputs a rising edge, hardware detects the rising edge and processes it, and after at least 2 **pclk** cycles from the **edge1** rising edge, it outputs the **INTSTATUS[x]** interrupt flag signal at the **T1** moment. The system software clears the interrupt generated by the rising edge at the moment of **edge1** at the time of **T2**.

At the **edge2** moment, **gpin[x]** inputs a falling edge, hardware detects the falling edge and processes it, and after at least 2 **pclk** cycles from the **edge2** falling edge, it outputs the **INTSTATUS[x]** interrupt flag signal at the **T3** moment. The system clears the interrupt generated by the falling edge at the time of **edge2** by software at the time of **T4**.

At the **edge3** moment, another rising edge is input to **gpin[x]**, which is processed by the **GPIO** module in the same way as above and generates an interrupt.

Note that if an interrupt is masked when an interrupt event (rising or falling edge) occurs, the interrupt will be ignored and the corresponding interrupt flag signal **INTSTATUS** will not be generated. In addition, as shown in the timing diagram, since the interrupt generated by **edge1** is not cleared until the **T2** time, a new interrupt event (falling edge) can not occur until the **T2-2*Tpclk** time, otherwise the new interrupt flag signal **INTSTATUS** generated by the new interrupt event will also be cleared at the same time at the **T2** time, and the new interrupt flag signal will be lost. The interrupt flag signal **INTSTATUS** generated by the new interrupt event will also be cleared at the same time at **T2**, and the new interrupt flag signal will

be lost. For the interrupt flag signal generated by the edge2 event only at the T4 time, so no new interrupt event (rising edge) can occur before the T4-2*Tpclk moment otherwise the interrupt flag signal INTSTATUS generated by the new interrupt event will be cleared at the same time at the T4 time, and the new interrupt flag signal will be lost.

workflow

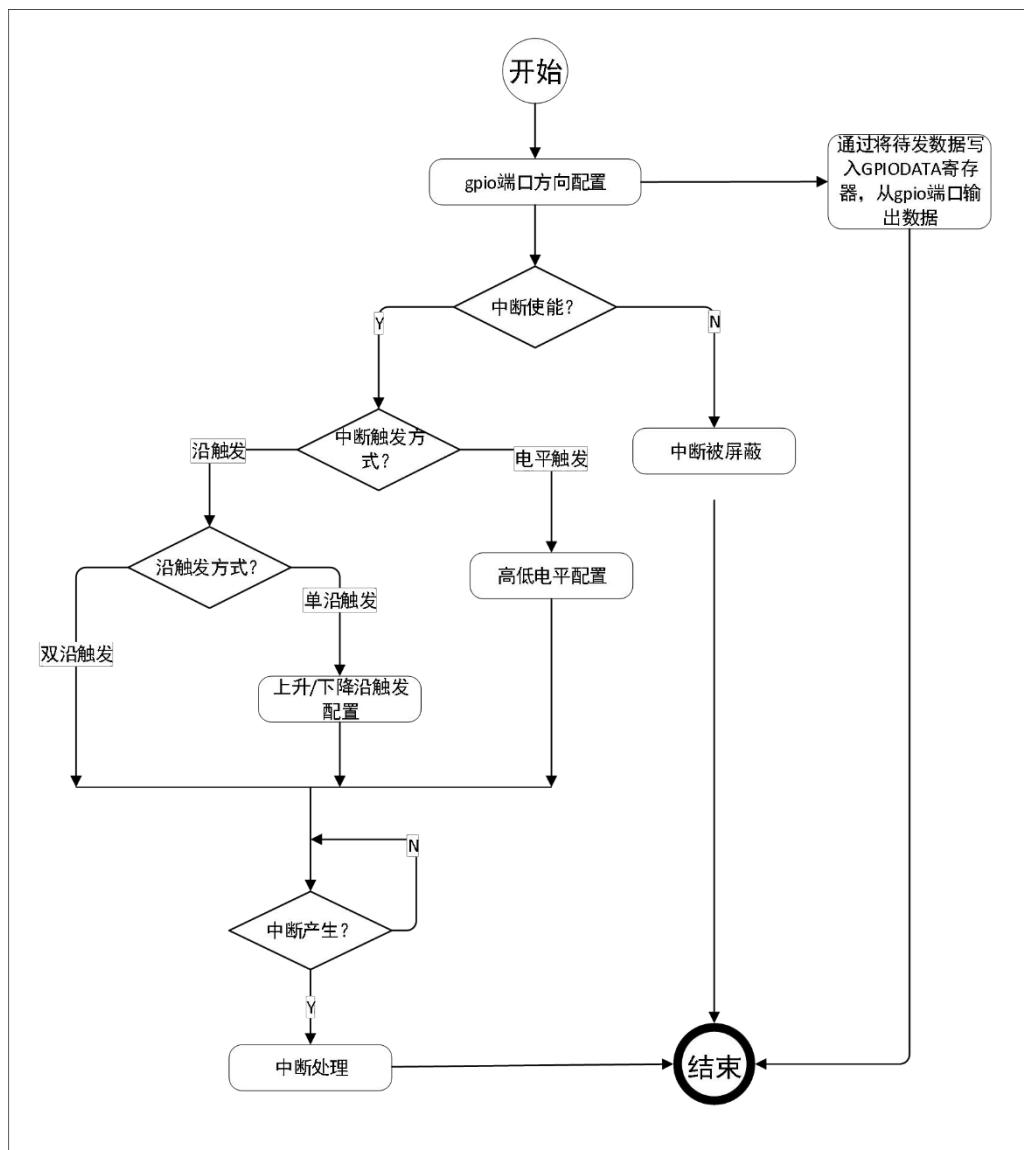


Figure 5-29 GPIO Operation Flowchart

- GPIO Module Clock Enable
- PORT port configured for GPIO function
- Configuration Port Direction (**GPIODIR**) Registers
- If the port is configured for output, data is output from the gpio port by writing the pending data to the **GPIODATA** register
- If the port is configured as an input, configure the interrupt enable and the interrupt trigger method, wait for an interrupt and handle the interrupt

Register Map GPIO DATA Register (0x00)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	GPIODATA	R/W	0	data register

GPIO DIR register (0x04)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	GPIO DIR	R/W	0	Sets the GPIO pin direction: 1: Set the GPIO pin of the corresponding bit as output pin 0: Set the GPIO pin of the corresponding bit as input pin

INTLVLTRG register (0x08)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTLVLTRG	R/W	0	Sets the GPIO pin interrupt sensitivity condition: 1: Set the GPIO pin of the corresponding bit as level detection 0: Set the GPIO pin of the corresponding bit as Along Detect

INTBE register (0x0C)

bitfield (math.)	name (of a	typol	reset	descriptive
------------------	------------	-------	-------	-------------

	thing)	ogy	value	
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit

GPIO_WIDTH	INTBE	R/W	0	Sets the GPIO pin edge trigger method: 1: Set the GPIO pin of the corresponding bit to double-edge triggered interrupt, i.e., both the rising and falling edges will trigger the interrupt 0: Set the GPIO pin of the corresponding bit to single-edge trigger interrupt, the corresponding bit of INTRISEEN register determines the rising/falling edge trigger. <small>Reference Manual</small>
------------	-------	-----	---	--

INTRISEEN register (0x10)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTRISEEN	R/W	0	Sets the GPIO pin interrupt event mode: 1: Set the GPIO pin of the corresponding bit to trigger an interrupt with a rising edge/high level. 0: Set the GPIO pin of the corresponding bit as falling edge/low level trigger interrupt

INTEN register (0x14)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTEN	R/W	0	Sets the GPIO pin interrupt enable: 1: Set the GPIO pin interrupt enable for the corresponding bit. 0: Set the GPIO pin interrupt disable

INTRAWSTAUS register (0x18)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTRAWSTAUS	R	0	<p>When the GPIO is configured in input mode, the interrupt flag is generated according to the set trigger condition, independent of the interrupt enable register. Set by hardware and cleared by software by writing 1 to GPIOIC.</p> <p>1: Indicates that the GPIO interrupt trigger condition for the corresponding bit is detected (raw, before mask) 0: Indicates that no GPIO interrupt trigger condition is detected for the corresponding bit.</p>

INTSTAUS Register (0x1C)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTSTAUS	R	0	<p>When the GPIO is configured in input mode and the interrupt of the corresponding bit is enabled, an interrupt flag is generated according to the set trigger condition. It is set by hardware and cleared by software by writing 1 to the GPIOIC.</p> <p>1: Indicates that an interrupt generated by the GPIO pin of the corresponding bit is detected</p>

0: Indicates that no interrupt generated by the GPIO pin of the corresponding bit is detected

INTCLR register (0x20)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:GPIO_WIDTH	RESERVED	RO	0	reserved bit
GPIO_WIDTH	INTCLR	W	0	<p>Write 1: Clear the corresponding bit of GPIO Pin Along Trigger Interrupt Flag. INTRAWSTATUS and INTSTATUS.clear interrupts After the flag, the corresponding bit of INTCLR is automatically restored to 0 by the hardware.</p> <p>Write 0: no effect.</p> <p>A read operation on this register returns 0.</p>

5.9 Basic Timer (TIMERBASE)

5.9.1 summarize

The basic timer module is equipped with timing function, has a 16-bit prescaler, supports interrupt, and needs to enable the timer module clock before use. The system block diagram of the basic timer module is shown below:

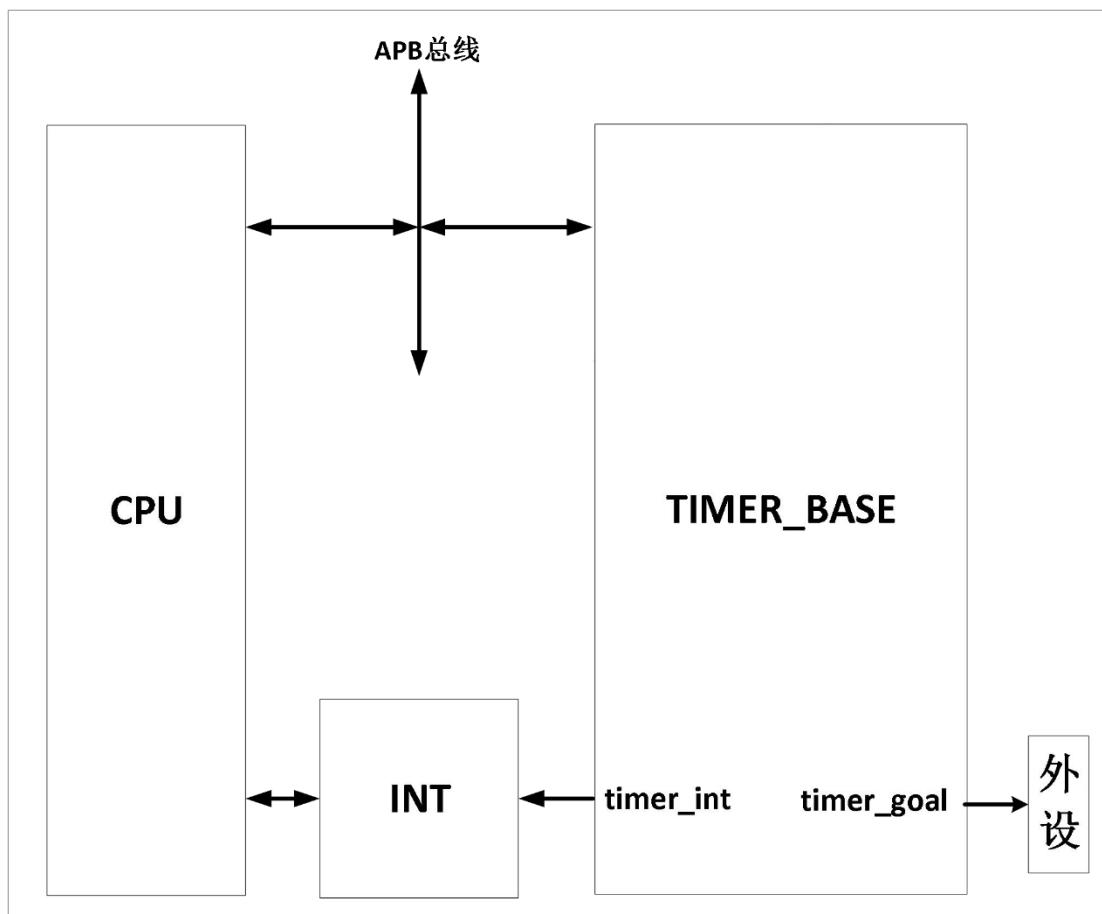


Figure 5-30 TIMERBASE Module System Block Diagram

5.9.2 characterization

- Supports 2 independent upward counting 16bit counters (HIGH, LOW)
- Supports 16bit prescaling

- Output of interrupt and target value flag signals

5.9.3 Block Diagram of Module Structure

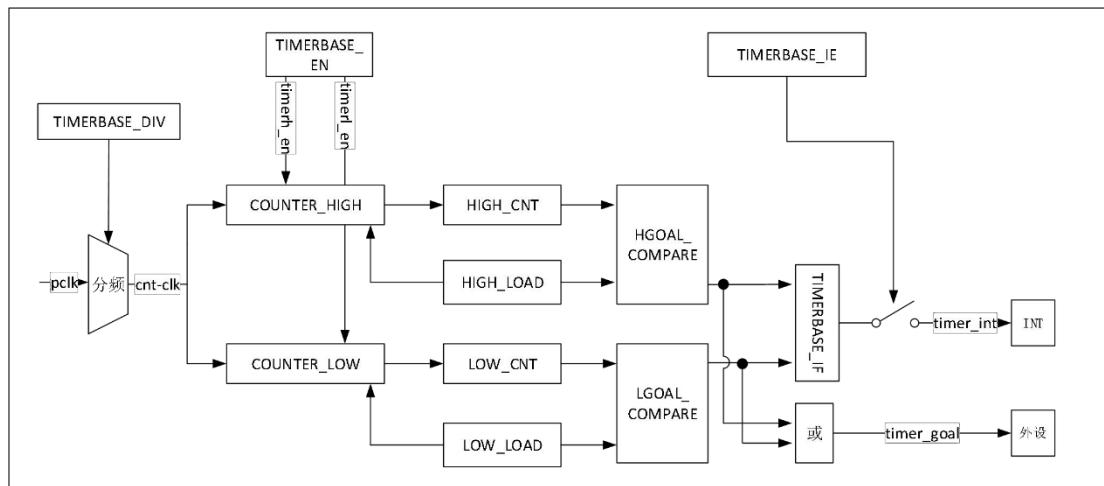


Figure 5-31 TIMERBASE Module Block Diagram

In this module, the two independent high and low counters share a 16bit divided frequency counting clock with a range of 1-65536, and the counting clock is provided for the counters through the configuration of the divided frequency register `TIMERBASE_DIV`. Through the high and low timer target registers `HIGH_LOAD/LOW_LOAD` can be configured for the high and low counters counting target value, and then turn on the counting enable after the counter starts counting, when the counter count value reaches the target value will generate the corresponding interrupt state, and output the counter target value flag signal `timer_goal`, when the interrupt enable is turned on, it will also generate the corresponding interrupt signal `timer_int`. signal `timer_int`.

5.9.4 Functional Description

High Counter Count

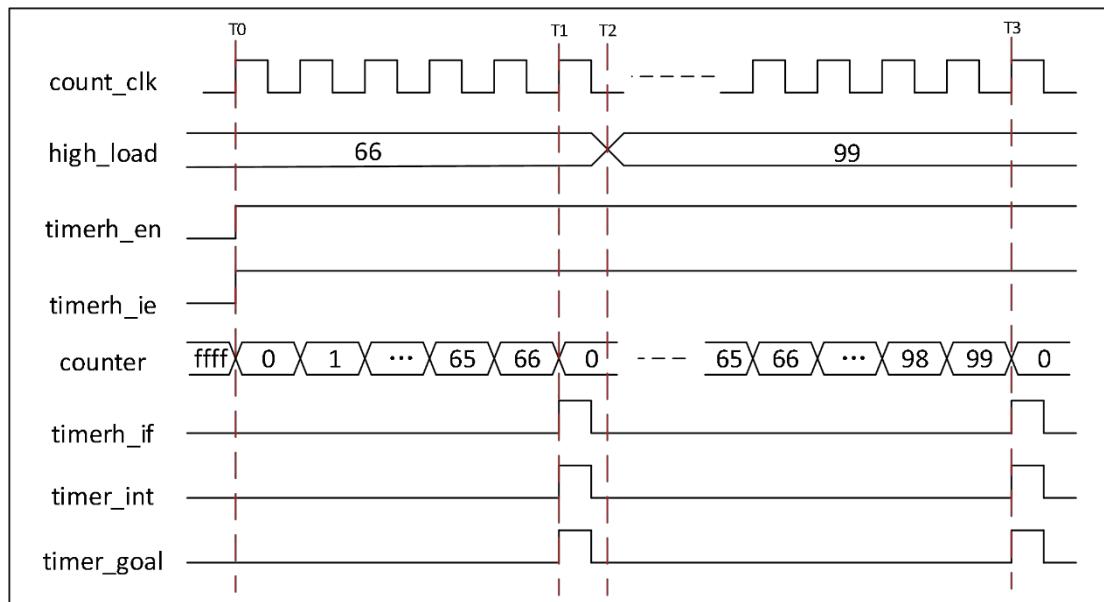


Figure 5-32 TIMERBASE High Counter Timing Chart

As shown in the above figure for the high counter counting timing diagram, when configured to count the target value, turn on the high counter counting enable and interrupt enable at T_0 time, after the counter reaches the target value at T_1 time, it generates the high counter interrupt state, the interrupt signal and the target value flag signal. change the target value at T_2 time, the counter will count to the new target value and generates the high counter interrupt state, interrupt signal and The clock source in this module is p The clock source in this module is `pclk`, which is the same source as the system clock, and the clock enable of this module can be configured through the `DEV_CLK_GATE` register in the `SYSCON` module.

Low Counter Count

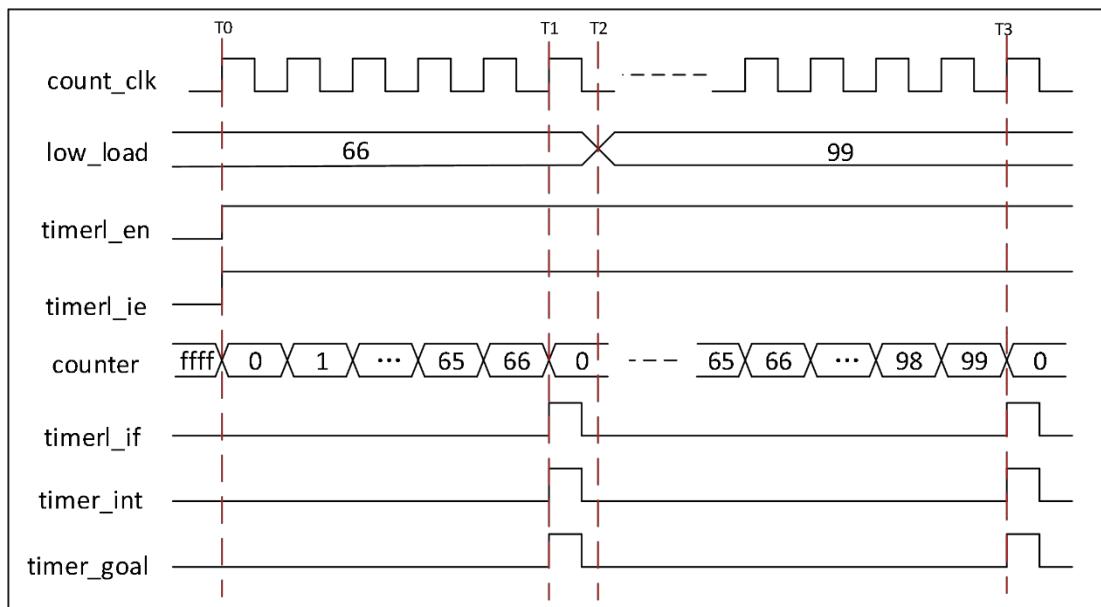


Figure 5-33 TIMERBASE Low Counter Timing Chart

As shown in the above figure for the low counter counting timing diagram, when configuring the counting target value, turn on the low counter counting enable and interrupt enable at T_0 time, after the counter reaches the target value at T_1 time, it generates the low counter interrupt state, the interrupt signal and the target value flag signal. change the target value at T_2 time, the counter will count to the new target value and generates the low counter interrupt state, interrupt signal and target value flag signal.

crossover frequency counting

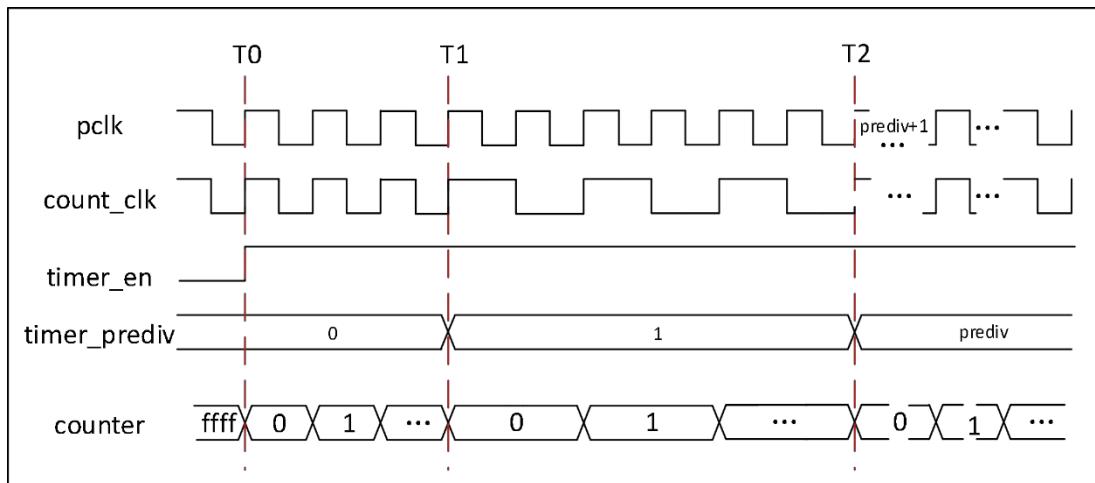


Figure 5-34 TIMERBASE Counter Frequency Division Count Timing Chart

The above figure shows the timings of the counter when the divider register `TIMERBASE_DIV` is configured as 0, 1 and `prediv` respectively.

Timed Duration Calculation Formula

When the basic timer is used as two independent 16-bit timers, it counts upward and the counting source is the system clock `Sys`. The timing duration `Tout` is calculated as follows:

$$Tout = (T_{pre} + 1) * (T_{load} + 1) / Sys.$$

Note: `Tpre` is the crossover coefficient, `Tload` is the high or low 16 bits of the loaded value, and `Sys` is the system clock.

disruptions

TIMERBASE provides 2 types of interrupt sources, and their relationship is shown below:

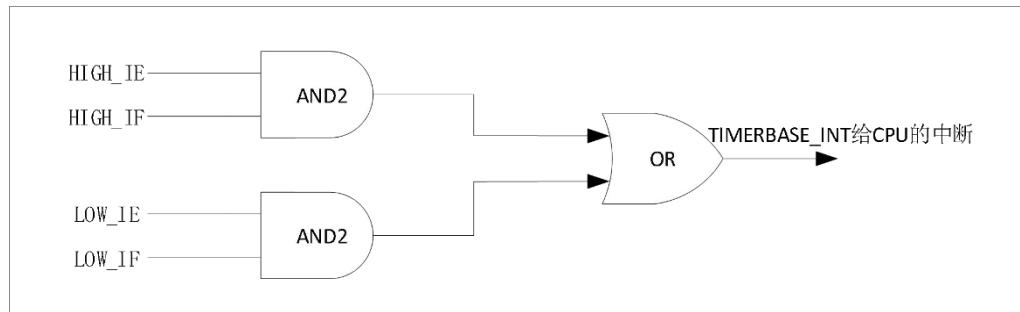


Figure 5-35 TIMERBASE Interrupt Flag and Interrupt Diagram

workflow

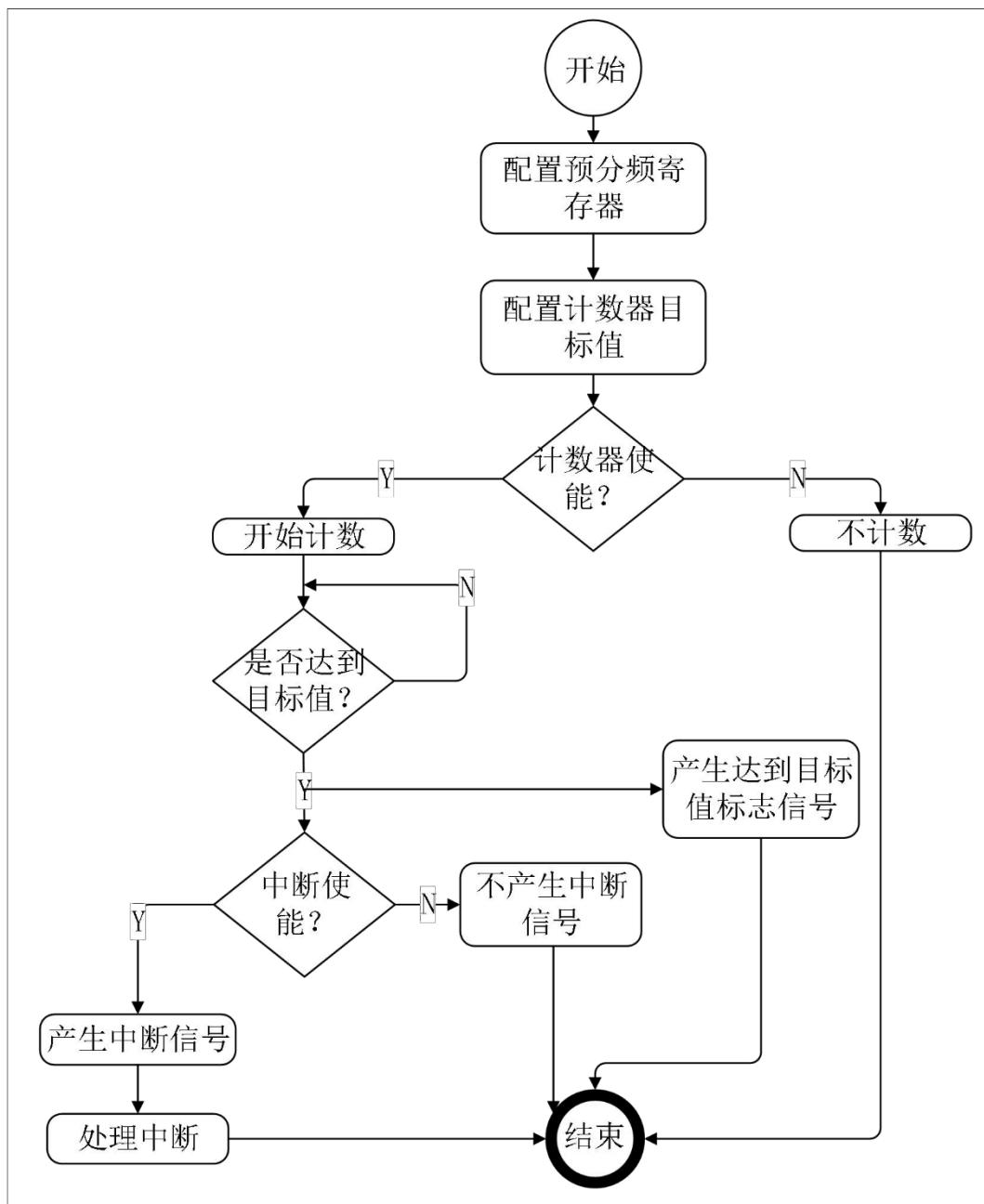


Figure 5-36 TIMERBASE Operation Flow

- The system clock is divided by setting the pre-division target value (1-65536) through the pre-division register (TIMERBASE_DIV).

- The count target value (16 bit)s set via the load value register (HIGH_LOAD or LOW_LOAD). There are two timers, HIGH_LOAD and LOW_LOAD, which can be configured according to the corresponding requirements.
- Interrupt enable is configured through the interrupt enable register (TIMERBASE_IE).
- Enabling of the corresponding timer is performed via the enable register (TIMERBASE_EN).
- The corresponding timer starts counting from 0 upwards, and when it reaches the loaded value, an interrupt is generated and the timer starts counting from 0 upwards again.
Starts counting and goes to the next cycle of counting.
- Interruptions can be queried (with interrupts enabled)through the interrupt status register (TIMERBASE_IF), and the interrupt status can be cleared by a write 1 operation to the register.
- During counting, the current count value can be obtained by reading the current count value register (HIGH_CNT or LOW_CNT).

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
timerbase0base: 0x40064000					
timerbase1base: 0x40064800					
TIMERBASE_EN	0x00	32	R/W	0x00	TIMER Enable Register
TIMERBASE_DIV	0x04	32	R/W	0x00	TIMER Count Clock Divider Register
TIMERBASE_IE	0x10	32	R/W	0x00	TIMER Interrupt Enable Register
TIMERBASE_IF	0x14	32	R/W	0x00	TIMER Interrupt status register

DP32G030

HIGH_LOAD	0x20	32	R/W	0xffff	TIMER HIGH Target Register Manual Configuration Register
HIGH_CNT	0x24	32	R	0x00	TIMER HIGH Current count value register
LOW_LOAD	0x30	32	R/W	0xffff	TIMER LOW Target Configuration Registers
LOW_CNT	0x34	32	R	0x00	TIMER LOW Current count value register

register description

TIMERBASE_EN register (0x00)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:2	RESERVED	R	0	reservations
1	HIGH_EN	R/W	0	TIMERBASE HIGH Timer Enable Register
0	LOW_EN	R/W	0	TIMERBASE LOW Timer Enable Register

TIMERBASE_DIV register (0x04)

bitfield d (math .)	name (of a thing)	typolo gy	reset value	descriptive
31:16	RESERVED	R	0	reservations
15:0	DIV	R/W	0	TIMERBASE Count Clock Prescaler Register 0x0000: indicates 1 division 0x0001: indicates 2 divisions 0xFFFF: indicates 65536 division frequency

TIMERBASE_IE register (0x10)

bitfield	name (of a thing)	typol	reset	descriptive

DP32G030

d (math .)		o gy	v a l u r e	
31:2	RESERVED	R	0	reserved bit
1	HIGH_IE	R/W	0	TIMERBASE HIGH Timer Interrupt Enable
0	LOW_IE	R/W	0	TIMERBASE LOW Timer Interrupt Enable

TIMERBASE_IF register (0x14)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:2	RESERVED	R	0	reserved bit
1	HIGH_IF	R/W	0	TIMERBASE HIGH Timer Interrupt Status Write 1 to clear the status.
0	LOW_IF	R/W	0	TIMERBASE LOW Timer Interrupt Status Write 1 to Clear

HIGH_LOAD register (0x20)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	HIGH_LOAD	R/W	0xffff	TIMERBASE HIGH Timer Target Configuration Register When the high 16bit counter counts up to the set value, the corresponding status signal is generated.

HIGH_CNT register (0x24)

bitfield	name (of a thing)	typol	reset	descriptive
----------	-------------------	-------	-------	-------------

DP32G030

d (math .)		ogy	value	
31:16	RESERVED	R	0	reserved bit
15:0	HIGH_CNT	R	0	TIMERBASE HIGH Timer Current Count Value

LOW_LOAD register (0x30)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive

31:16	RESERVED	R	0	reserved bit	Reference Manual
15:0	LOW_LOAD	R/W	0xffff	TIMERBASE LOW Timer Target Configuration Register When the low 16bit counter counts up to the set value, the corresponding status signal is generated.	

LOW_CNT register (0x34)

bitfield	name (of a thing)	typology	reset value	descriptive
d (math .)				
31:16	RESERVED	R	0	reserved bit
15:0	LOW_CNT	R	0	TIMERBASE LOW Timer Current Count Value

5.10 Advanced Timer (TIMERPLUS)

5.10.1 summarize

The advanced timer module is equipped with timing, counting, capturing, cycle pulse output, etc. It has a 16-bit prescaler, supports interrupts, and two independent 16-bit timers (HIGH and LOW) of which the low 16-bit timer also supports the HALL function, which is required to enable the clock of the advanced timer module before use.

The system block diagram of the advanced timer module is shown below:

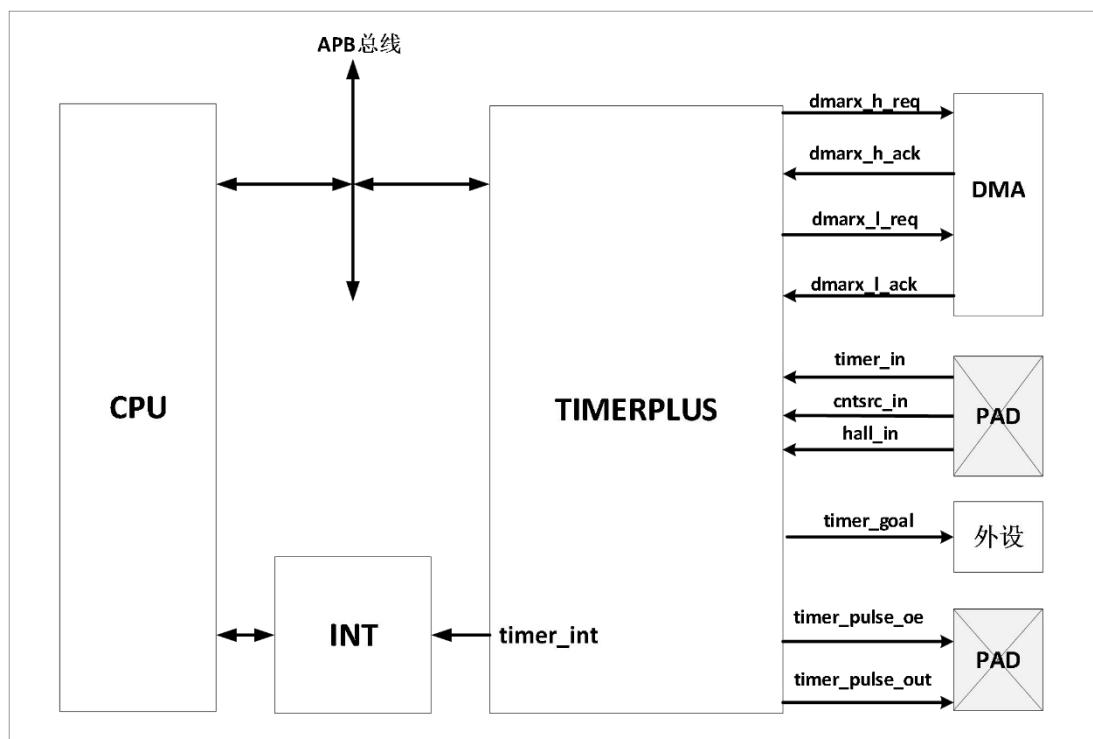


Figure 5-37 TIMERPLUS System Block Diagram

As shown in the figure above, the **TIMERPLUS** module in this chip completes the communication with the **CPU** through the **APB** bus and supports the communication with the **DMA**, which can generate many kinds of interrupt signals, and it can also output the cycle pulse output signal, cycle pulse signal output enable output to the **PAD** port, and transmit the counter counter to reach the target value flag signal to other peripheral devices; at the same time, **TIMERPLUS** can also input off-chip

timer_in, cntsrc_in, hall_in, and other signals through the [Reference Manual](#). At the same time, TIMERPLUS can also input off-chip timer_in, cntsrc_in, hall_in and other signals through the PAD port.

5.10.2 characterization

- 2 independent 16bit counters (HIGH and LOW)
- With 16bit prescaler counter
- Two timers with timing, counting, input capture and periodic pulse output functions respectively
- The LOW counter supports the HALL function
- Separate support for count clock selection
- Supports multiple interrupt functions
- Both timers have their own target value shadow registers, and the new target value will take effect next cycle
- Supports DMA interface to read input pulse capture value

5.10.3 Block Diagram of Module Structure

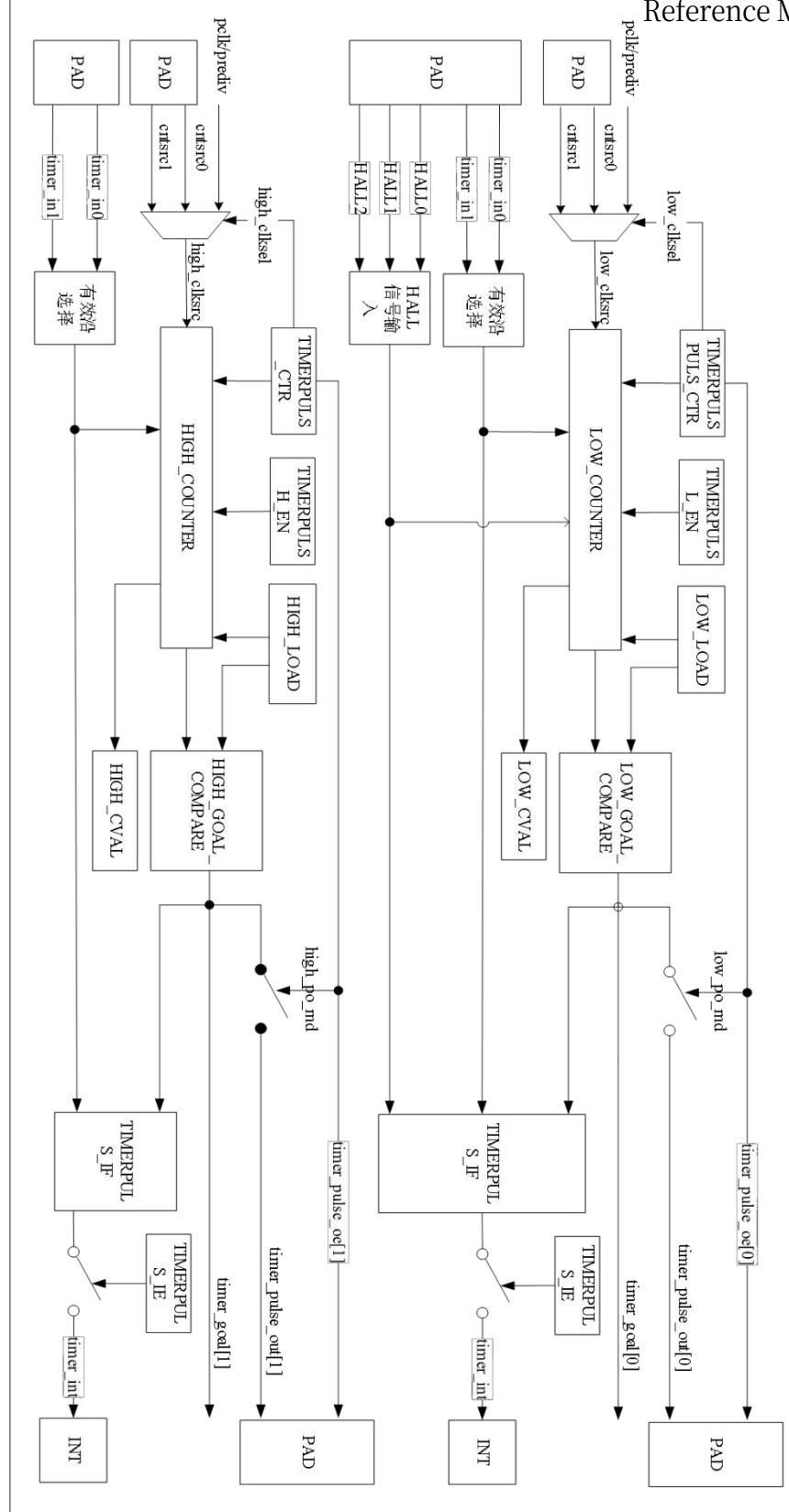


Figure 5-38 TIMERPLUS Structure Block Diagram

The above figure shows the structure of TIMERPLUS module, this module contains two independent 16bit counters, both counters can choose internal prescaler clock or off-chip counter clock cntsrc, where the internal prescaler clock can be prescalerized through the pclk clock configuration prescaler register TIMERPLUS_PREDIV, the PCLK_PREDIV bit, and the prescaler value range is 1-65536. The internal pre-divided clock can be divided by the PCLK_PREDIV bit of TIMERPLUS_PREDIV in the pclk clock configuration pre-divided register, with a value range of 1-65536. The clock source of this module is pclk, which is the same source as the system clock, and can be configured through the DEV_CLK_GATE register in the SYSCON module to enable the clock of this module.

Both high counter and low counter support timing function, counting function and input capture function, and low counter also supports HALL function. The functions of the high and low counters and the related configurations can be configured through the register TIMERPLUS_CTR.

The interrupt signals of different states in this module are controlled by the interrupt enable register TIMERPLUS_IE, and the interrupt signals are generated only when the corresponding interrupt state is generated with the corresponding interrupt enable turned on.

This module also supports DMA read of input capture, in input capture mode, when DMA read enable is 1, the DMA will read the capture value from the Counter Capture Value Register instead of the CPU.

The two signals timer_goal[1:0] generated by this module can be used as the trigger source for the SARADC module.

5.10.4 Functional Description

Counting Clock Source Selection and Prescaling

Both internal timers in this module can select the counting clock source

separately and share a common prescaler. The counter [Reference Manual](#) diagram is as follows, in which the off-chip clocks `cntsrc0` and `cntsrc1` are inputted through the `PAD` port, which `shares` the same `PAD port` with the off-chip input signal `timer_in`, and the `PAD` port can also be used as the input port of the external clocks `cntsrc0` and `cntsrc1` when the `PAD` port is selected to be `timer_in` in the `IO` configuration. When `timer_in` is selected in the `IO` function configuration, the `PAD` port can also be used as an input port for external clocks `cntsrc0` and `cntsrc1`.

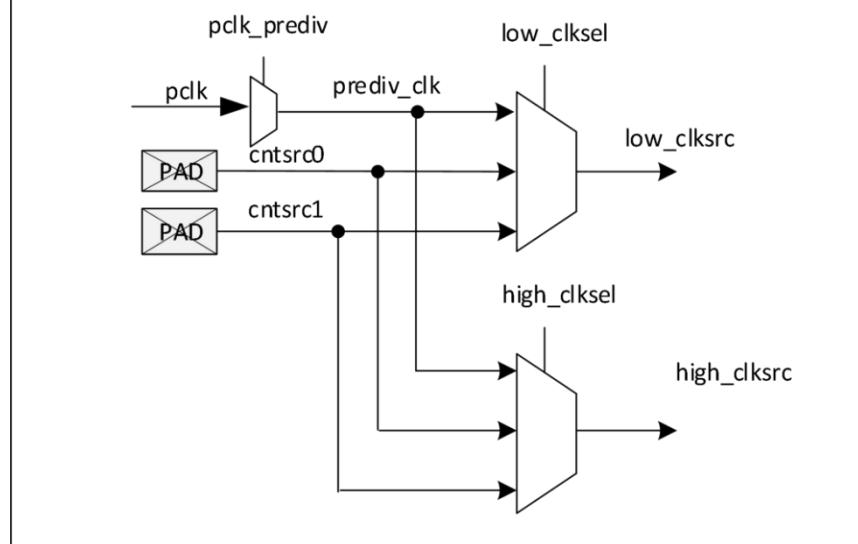


Figure 5-39 TIMERPLUS Clock Source Selection Schematic

The prescaler is used only for `pclk`. The prescaler can divide the `pclk` clock under the control of the `PCLK_PREDIV` bit of the prescaler register, with the value of 1-65536. The divided clock and the off-chip clocks, `cntsrc0` and `cntsrc1`, provide the counting clock for the counter through the selection of the counter clock register. The counter timing with `pclk` divider clock is shown below:

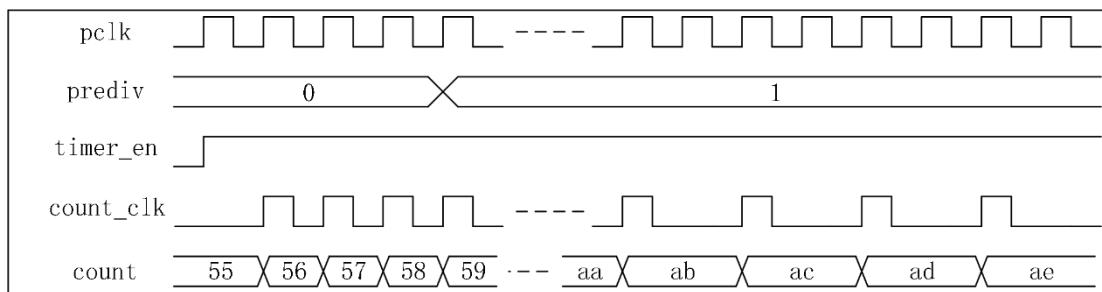


Figure 5-40 Counter Timing Diagram with TIMERPLUS `pclk` Crossover Clock

Timer mode

The timing mode of this module is when the counting clock is selectable as internal

DP32G030

prescaled clock, off-chip cntsrc[0] Reference Manual [1].

In the three clock source cases, the counter counts and the timing time is configured through the counter target value configuration register when the counting

The counter reaches target value flag signal is generated after counting to the target value. The timer mode can be configured with a cycle pulse output to the pin, which can be used as a simple square wave output. Modifying the cycle value during timer operation does not take effect immediately, but takes effect the next cycle after the end of the cycle.

The advanced timer counts up when used as two independent 16-bit timers to count the clock source as the system clock `pclk`

For example, the formula for calculating the timing time `Tout` is as follows:

$$Tout = (T_{pre} + 1) * (T_{load} + 1) / pclk.$$

Note: `Tpre` is the crossover coefficient, `Tload` is the high or low 16 bits of the load value, and `pclk` is the system clock.

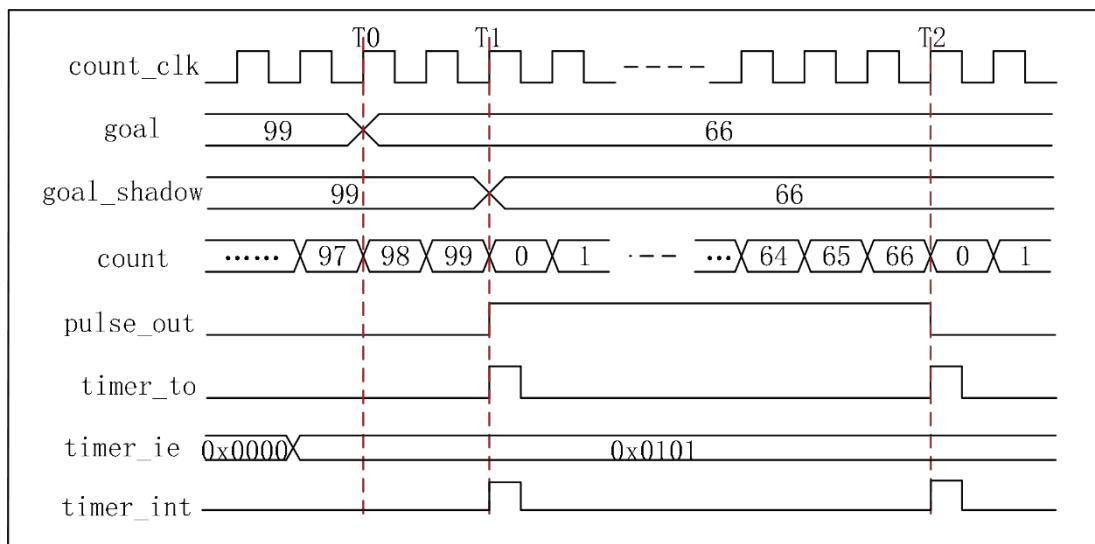


Figure 5-41 Counter Timing Diagram in TIMERPLUS Timing Mode

Where `count_clk` is the count clock of the counter, `goal` is the target value configured by the CPU, `goal_shadow` is the shadow register value, `count` is the count value, `PLUS_out` is the original value of the periodic pulse output, and `timer_to` is the flag signal when the count value reaches the target value. When the target value is changed at `T0`, the target value of the shadow register is not changed immediately,

but is changed only after the counter reaches the previous target value and after the target value is reached at T_1 time, it will cause the cycle pulse output signal to flip over, generate the flag signal of reaching the target value, and the counter will start counting from zero again. After the time counter reaches the new target value again, it will generate the target value flag signal again, the cycle pulse output signal will flip again, the counter will start counting again from 0. When the time counter reaches the new target value again, it will generate the target value interrupt signal again. The high and low counter cycle pulse output enable bits **HIGH_PO_MD** and **LOW_PO_MD** of the **TIMERPLUS_CTR** register respectively are

Controls the output of different bit bits of the cycle pulse signal to the PADs, and the cycle pulse signal will be output to the corresponding PAD port only when the enable is turned on.

counting mode

The counting mode of this module refers to counting the rising edge, falling edge or double edge of the off-chip `timer_in[0]` or off-chip `timer_in[1]` signal, and generating the counter reach target flag signal when the count reaches the target value. The counter is counted by `pclk` detecting the rising edge, falling edge or double edge, and the specific timing diagram is shown below:

Rising edge valid

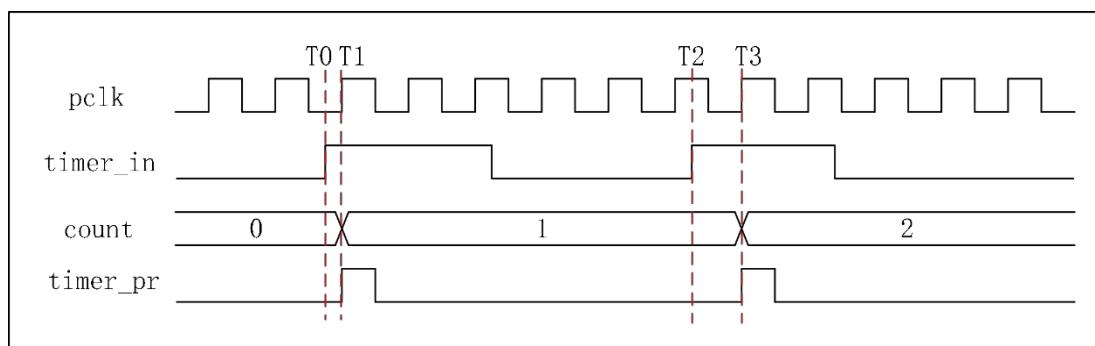


Figure 5-42TIMERPLUS Count Mode Rising Edge Valid Timing Chart

In the counting mode, `pclk` is used as the counter clock, `timer_in` is the off-chip input signal, `count` is the count value, and `timer_pr` is the rising edge flag signal detected by `timer_in`. The input signal `timer_in` generates a rising edge at `T0`, the counter is incremented at the next rising edge of the counting clock at `T1`, and a rising edge flag is generated, as well as at `T2` and `T3`.

Effective along the descending edge

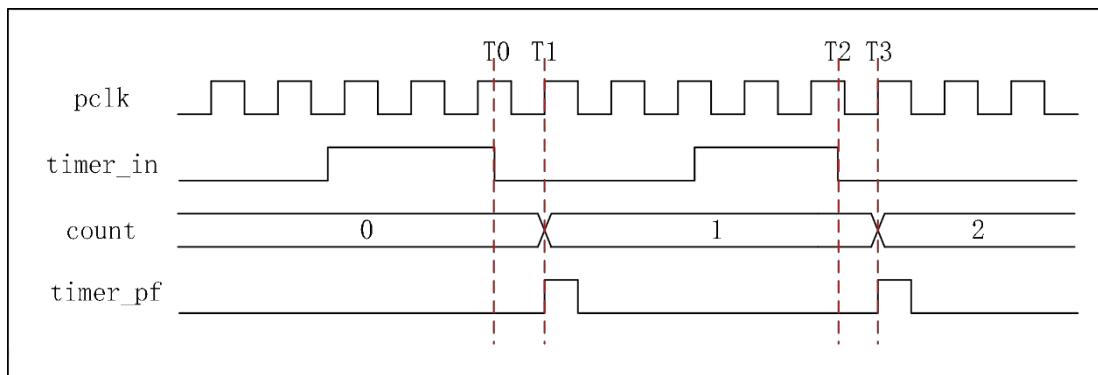


Figure 5-43 TIMERPLUS Count Mode Falling Edge Valid Timing Chart

Where **pclk** is the counter clock, **timer_in** is the off-chip input signal, **count** is the count value, and **timer_pf** is the detection of **timer_in** falling edge flag signal. Input signal **timer_in** at **T0** generates a falling edge, in the next rising edge of the counting clock **T1** time counter plus one, and generates a falling edge flag signal, **T2** and **T3** time is also the same.

double-edged validity

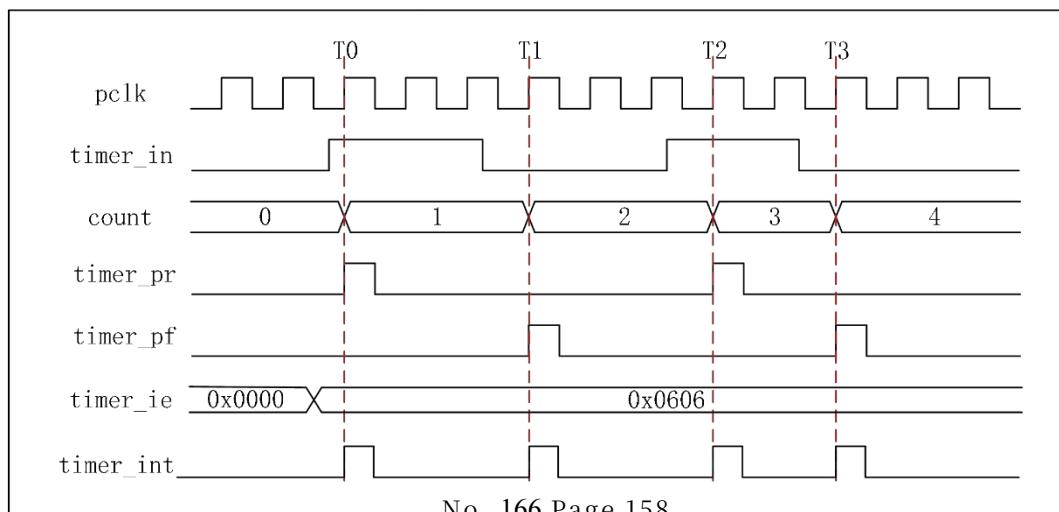


Figure 5-44 TIMERPLUS Count Mode Double Edge Valid Timing Diagram

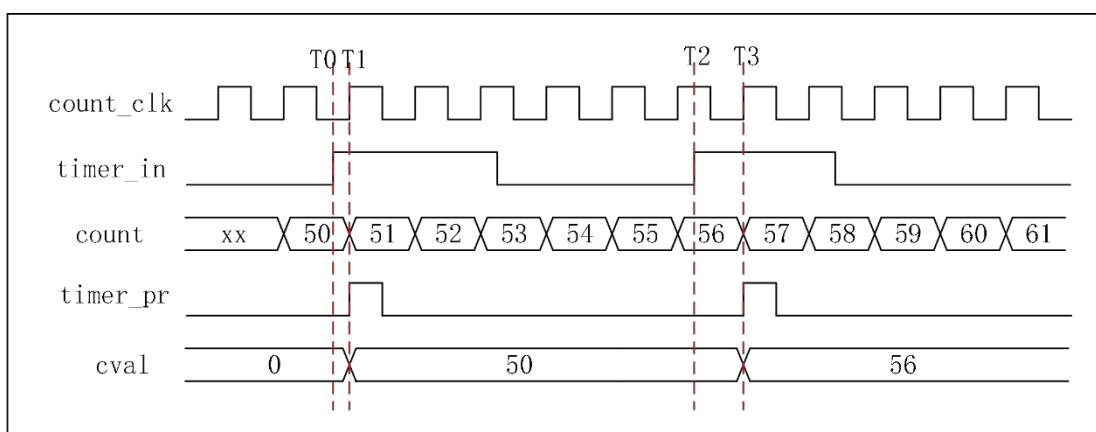
Where `pclk` is the counter clock, `timer_in` is the off-chip input signal, `count` is the count value, `timer_pr` is the detected `timer_in` rising edge flag signal, `timer_pf` is the detected `timer_in` falling edge flag signal. The case of double-edge validity is the same as the counter counting in the case of rising and falling edge validity, i.e., after the valid edge arrives, the counter counting will be valid on the next rising edge of the counting clock, and the corresponding interrupt signals will be generated when the corresponding interrupt enable is turned on, such as the `T0`, `T1`, `T2` and `T3` moments in the above figure.

Input Capture Mode

Input capture mode means that the counter counts when the counting clock can be selected as internal pre-scaled clock, off-chip `cntsrc[0]` and off-chip `cntsrc[1]` clock sources. The current counter value is saved in register `cval` when the rising or falling edge of the off-chip `timer_in[0]` or off-chip `timer_in[1]` signal is detected.

The timing diagram for different effective edges is shown below:

Rising edge valid



DP32G030
Figure 5-45 Rising Edge Valid Timing Diagram under TIMERPLUS Input Capture Reference Manual

count_clk is the counter clock, timer_in is the off-chip input signal, count is the count value, timer_pf is the detected falling edge flag signal of timer_in, and cval is the captured counter count value. In the case of falling edge validity, when the falling edge of the off-chip input signal timer_in arrives at T0, the current count value of the counter is saved to register CVAL at T1, and the corresponding detected falling edge flag signal is generated. When the falling edge of the off-chip input signal timer_in arrives again at T2, the value of the counter will be saved to register CVAL again at T3.

Effective along the descending edge

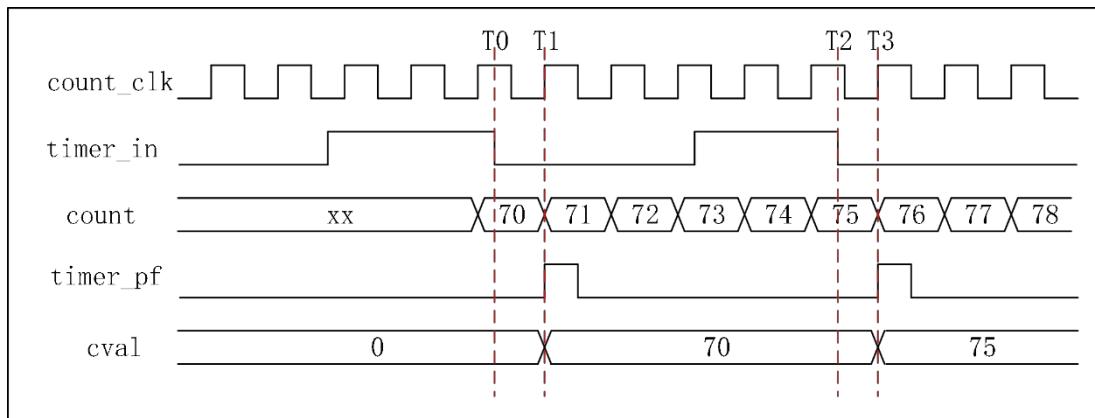


Figure 5-46 TIMERPLUS Input Capture Lower Falling Edge Valid Timing Chart

count_clk is the counter clock, timer_in is the off-chip input signal, count is the count value, timer_pf is the detected falling edge flag signal of timer_in, and cval is the captured counter count value. In the case of falling edge validity, when the falling edge of the off-chip input signal timer_in arrives at T0, the current count value of the counter is saved to register cval at T1, and the corresponding detected falling edge flag signal is generated. When the falling edge of the off-chip input signal timer_in arrives again at T2, the value of the counter is saved to register cval again at T3.

double-edged validity

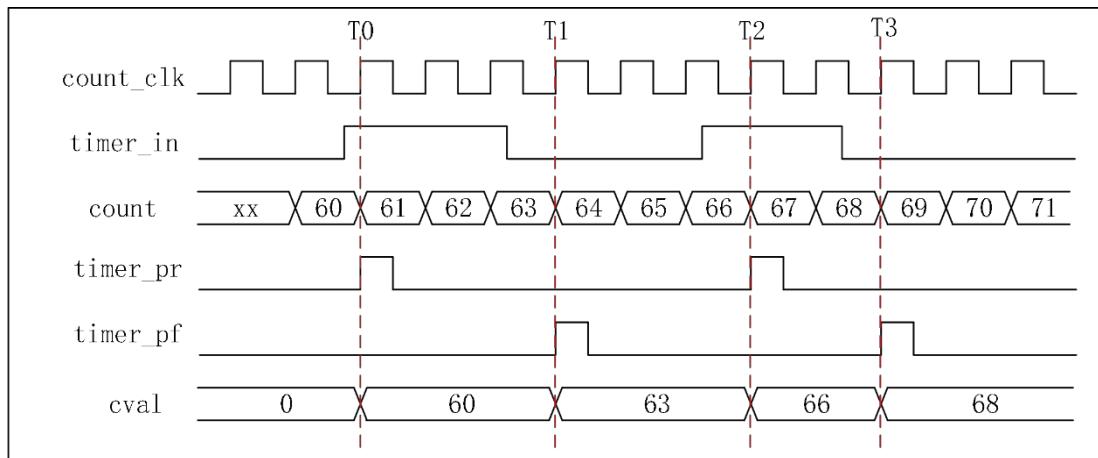


Figure 5-47 TIMERPLUS Input Capture Lower Double-Edge Valid Timing Chart

`count_clk` is the counter clock, `timer_in` is the off-chip input signal, `count` is the count value, `timer_pr` is the signaled rising edge of `timer_in`, `timer_pf` is the signaled falling edge of `timer_in`, and `cval` is the captured count value of the counter. In the case of double-edge validity, when the rising or falling edge arrives, the current count value of the counter will be saved to register `cval` and the corresponding flag signal will be generated, as shown in the figure for `T0`, `T1`, `T2` and `T3` moments.

HALL mode

In this module, only the `LOW` timer has the `HALL` mode function, which can be used for Hall signal acquisition. This mode means that the counter will count when the counting clock can be selected from internal pre-divided clock, off-chip `cntsrc[0]` and off-chip `cntsrc[1]` clock sources. When the rising and falling edges of the off-chip `hall_in[0]`, off-chip `hall_in[1]` and off-chip `hall_in[2]` signals are detected, the current counter count value is saved via register `cval` and the `LOW` counter is cleared to zero.

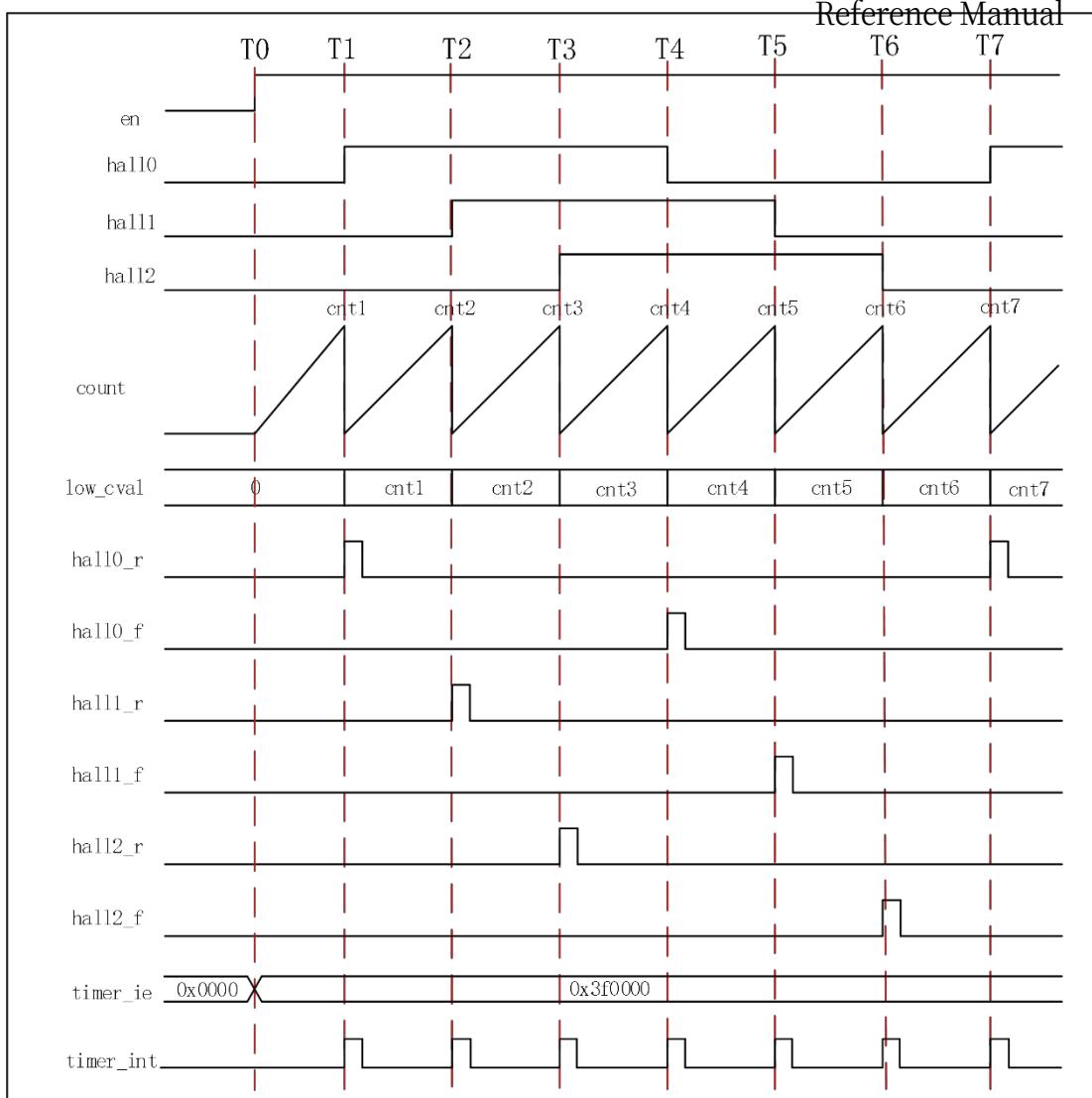


Figure 5-48 TIMERPLUS Timing Diagram in HALL Mode

hall0, **hall1** and **hall2** are the three input HALL signals, **count** is the count value, **hall0_r**, **hall1_r** and **hall2_r** are the signals to detect the rising edge of HALL signal, **hall0_f**, **hall1_f** and **hall2_f** are the signals to detect the falling edge of HALL signal, **low_cval** is the count value of the captured counter, **timer_ie** is the interrupt enable signal, **timer_int** is the generated interrupt signal. **cval** is the captured counter count value, **timer_ie** is the interrupt enable signal, and **timer_int** is the generated interrupt signal. The circuit clears the counter every time it detects a rising and falling edge, and saves the count value before clearing to the **low_cval** register, and generates the

DP32G030

rising or falling edge flag signal of each HALL signal. When Reference Monitoring interrupt is turned on, the corresponding interrupt signal will be generated after the corresponding status flag signal is generated, as shown in the above figure for T1-T7.

disruptions

TIMERPLUS provides 12 interrupt sources as shown below:

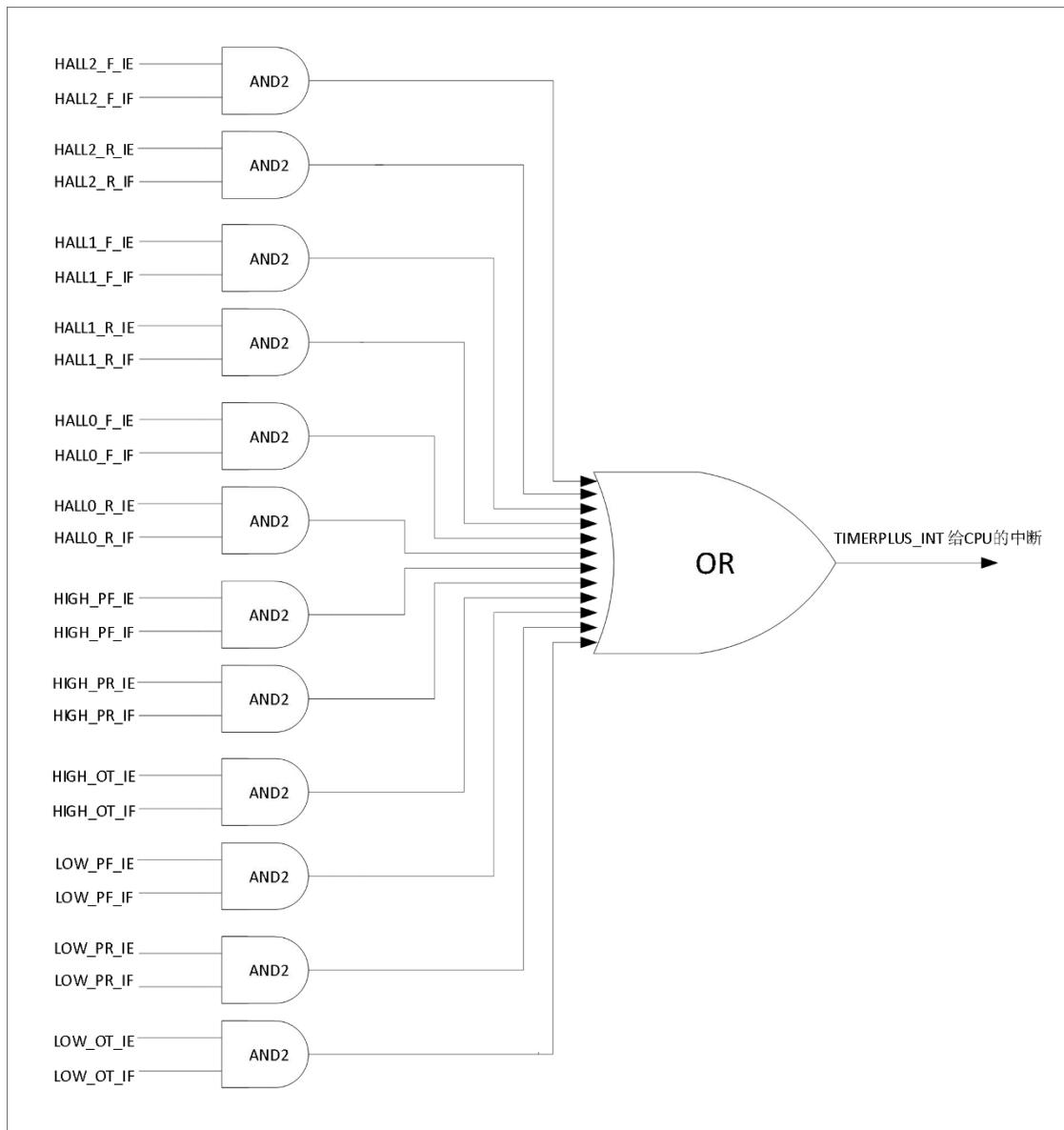


Figure 5-49 TIMERPLUS Interrupt Flag and Interrupt Diagram



DP32G030

Reference Manual

workflow

Timed Mode Operation Procedure

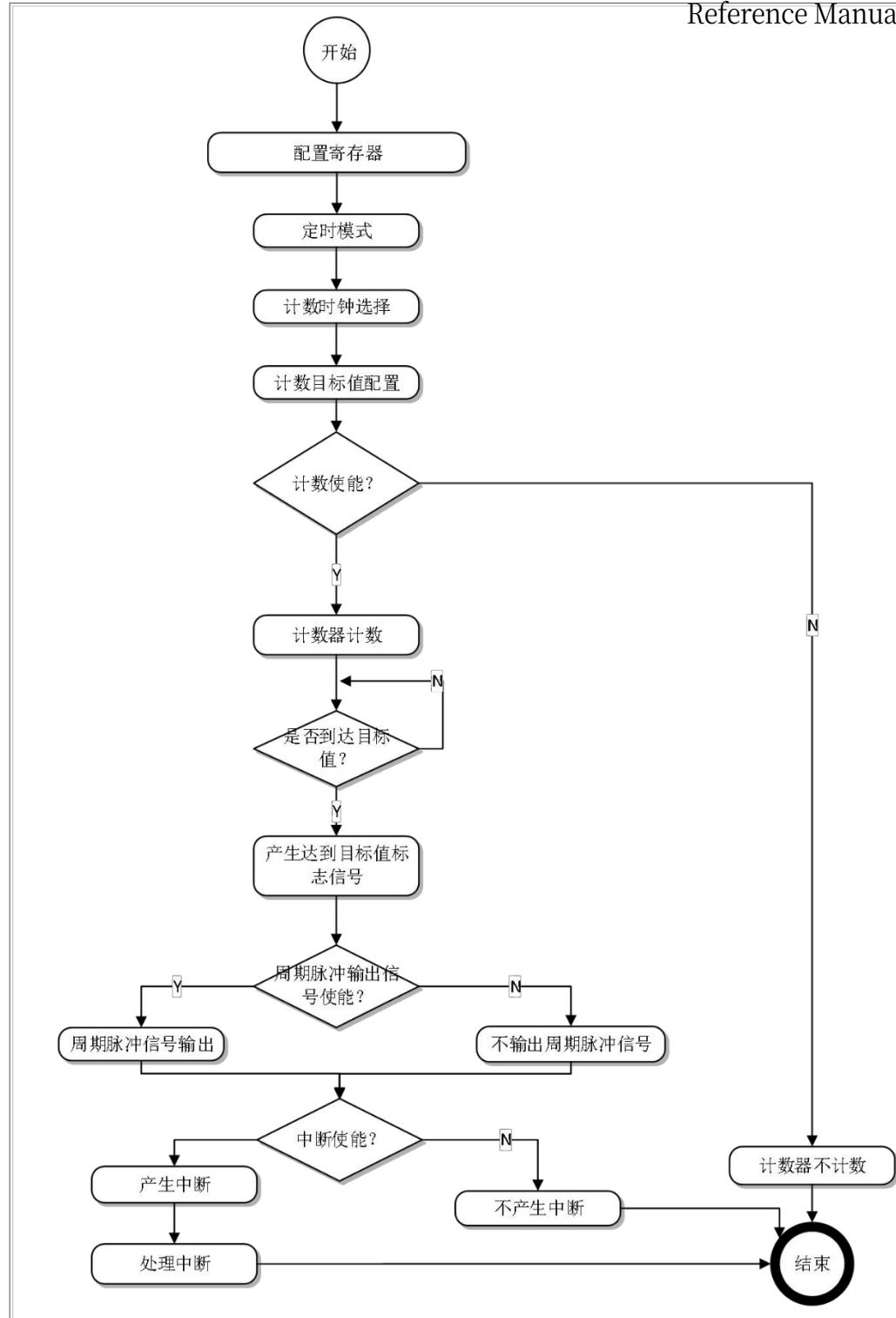


Figure 5-50 TIMERPLUS Timer Mode Operation Flowchart

- TIMER PLUS Clock Enable
- The PORT port is configured for the TIMER PLUS function.
- Configure the counter operating mode register to select the timing mode
- Configuring the Counter Clock
- If the internal `pclk` clock is selected as the counter clock, the pre-divider register must be configured to divide the clock.
- Configure the counter count target value
- Configure Interrupt Enable
- Configuration cycle pulse signal enable
- Configure counter enable to start counting



DP32G030

Reference Manual

Counting Mode Operation Procedure

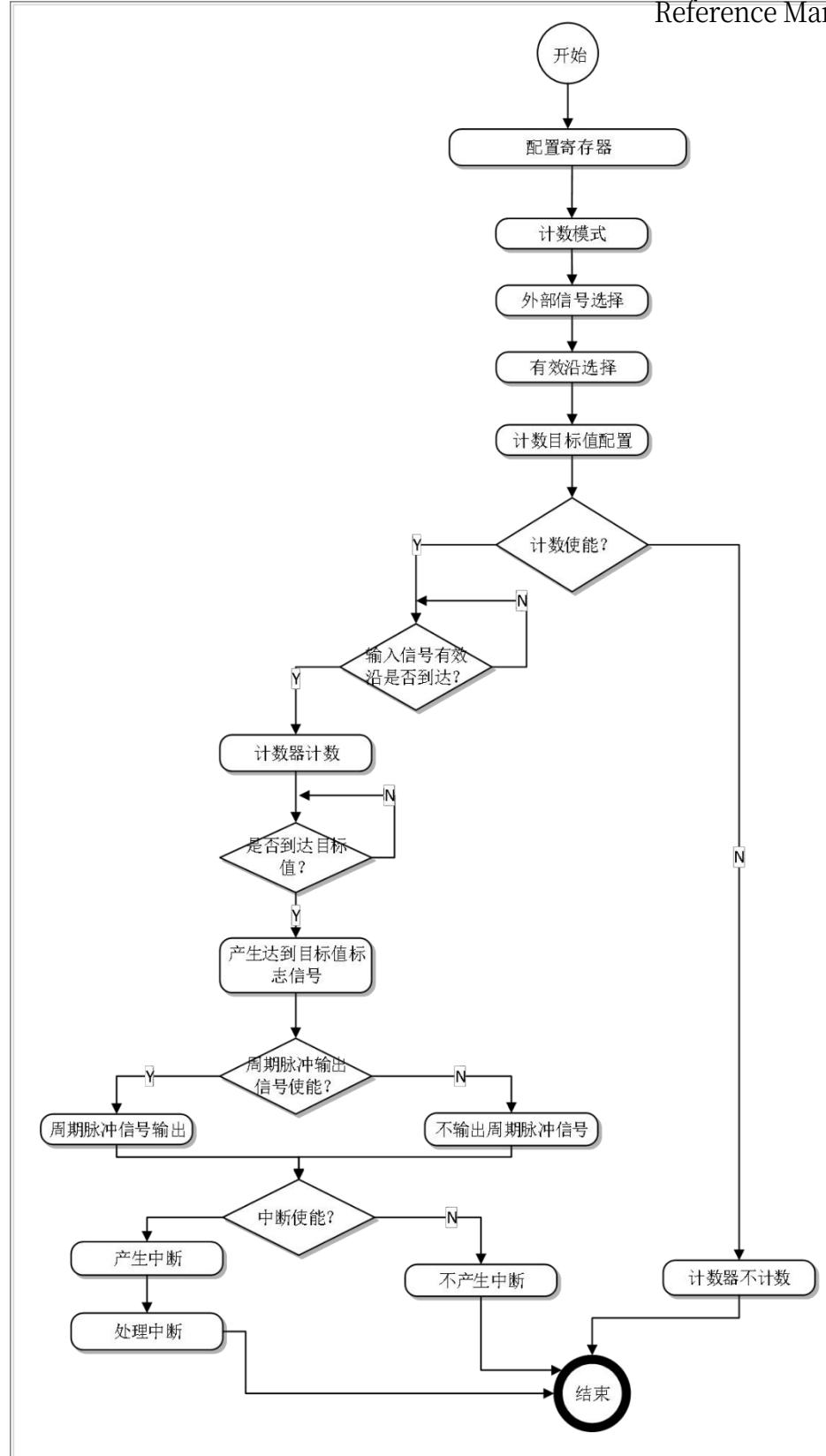


Figure 5-51 TIMERPLUS Count Mode Operation Flowchart

- TIMERPLUS Clock Enable
- PORT port configured for TIMERPLUS function
- Configure the counter operating mode register to select the counting mode
- Configure off-chip input signals and active edge
- Configure the preshunt register to divide the clock.
- Configure the counter count target value
- Configure Interrupt Enable
- Configuration cycle pulse signal enable
- Configure counter enable to start counting



DP32G030

Reference Manual

Input Capture Mode Operation Flow

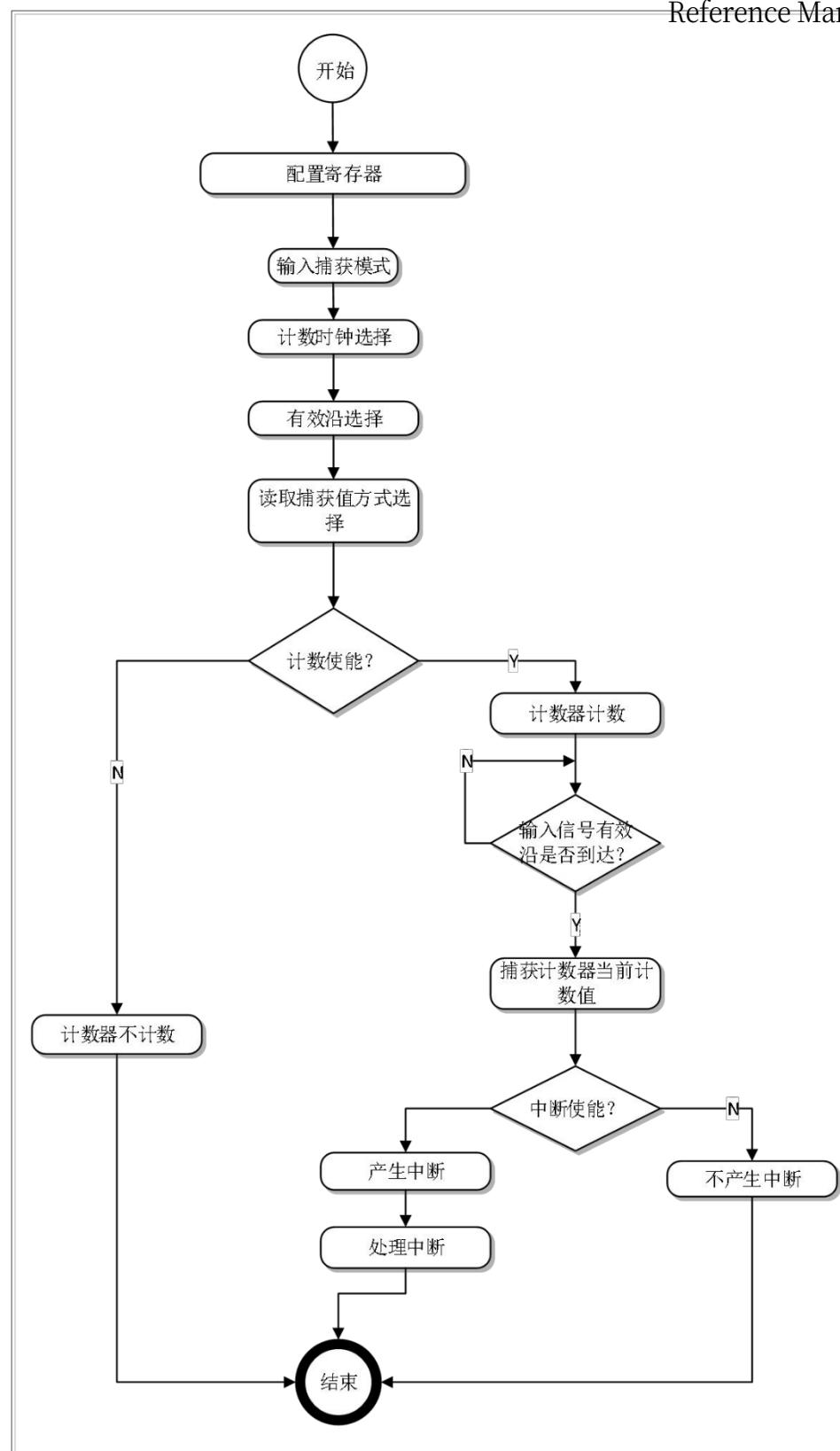


Figure 5-52 TIMERPLUS Input Capture Mode Operation Flowchart

- TIMER PLUS Clock Enable
- The PORT port is configured for the TIMER PLUS function.
- Configure the counter operating mode register to select the input capture mode
- Configuring the Counter Clock
- If the internal `pclk` clock is selected as the counter clock, a prescaler counter must be configured to divide the clock
- Configure the input signal and the active edge of the input signal
- Configure Interrupt Enable
- Configure counter enable to start counting



DP32G030

Reference Manual

HALL Mode Operation Procedure

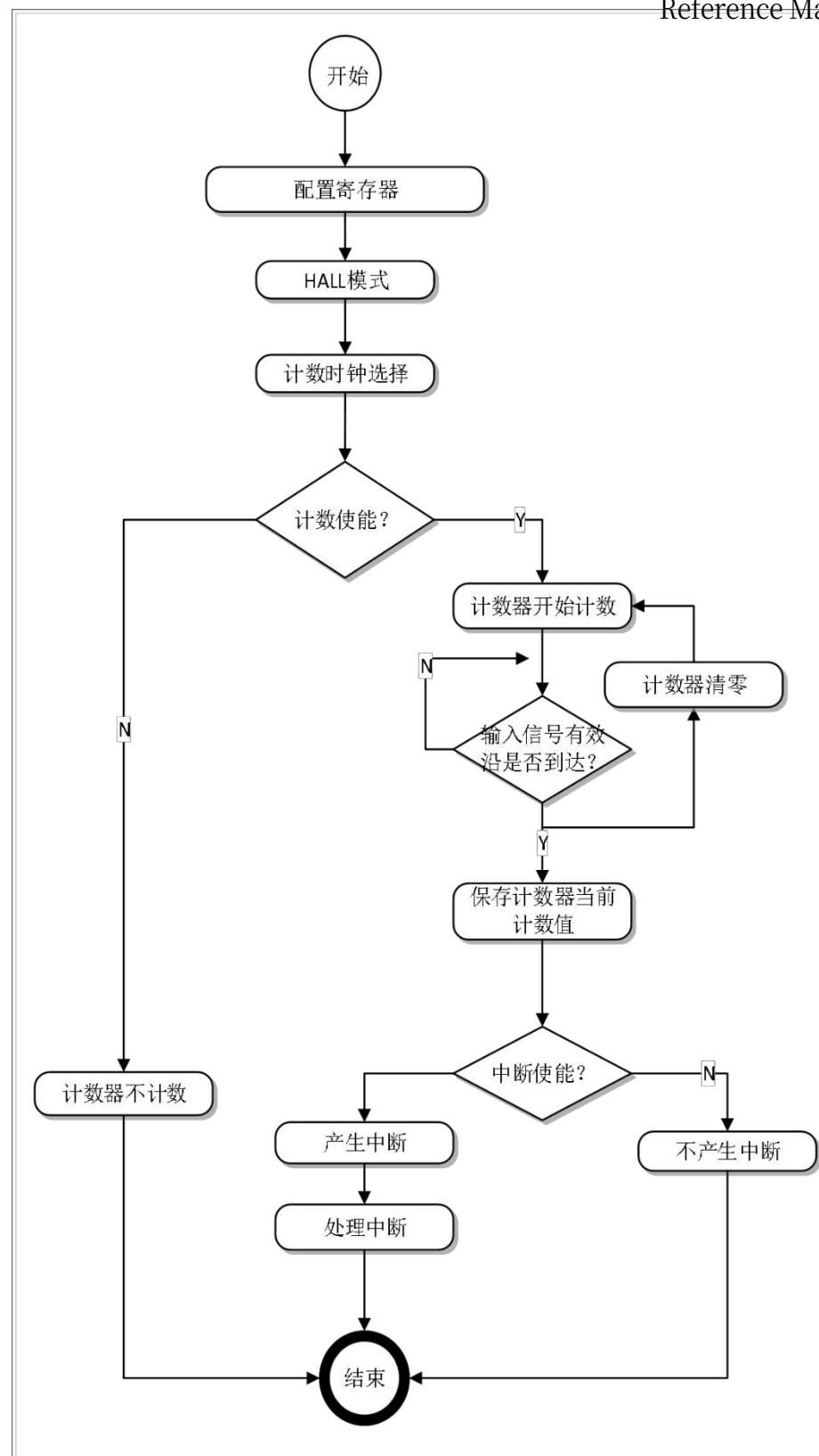


Figure 5-53 TIMERPLUS HALL Mode Operation Flowchart

- TIMER PLUS Clock Enable
- The PORT port is configured for the TIMER PLUS function.
- Selects the low 16-bit counter
- Configure the counter operating mode register to select HALL mode.
- Configuring the Counter Clock
- If the internal `pclk` clock is selected as the counter clock, a prescaler counter must be configured to divide the clock
- Configure Interrupt Enable
- Configure counter enable to start counting

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
timerplus0base: 0x40067000					
timerplus1base: 0x40067800					
TIMERPLUS_EN	0x00	32	R/W	0x00	TIMERPLUS Enable Register
TIMERPLUS_DIV	0x04	32	R/W	0x00	TIMERPLUS Count Clock Prescaler Register
TIMERPLUS_CTR	0x08	32	R/W	0x600060	TIMERPLUS Configuration Register
TIMERPLUS_IE	0x10	32	R/W	0x00	TIMERPLUS Interrupt Enable Register
TIMERPLUS_IF	0x14	32	R/W	0x00	TIMERPLUS Interrupt Status Register
HIGH_GOAL	0x20	32	R/W	0xffff	TIMERPLUS HIGH Target Configuration Registers

HIGH_CNT	0x24	32	R	0x00	TIMERPLUS HIGH Current count value register
HIGH_CVAL	0x28	32	R	0x00	TIMERPLUS HIGH Capture Value Register
LOW_GOAL	0x30	32	R/W	0xffff	TIMERPLUS LOW Target Configuration Registers
LOW_CNT	0x34	32	R	0x00	TIMERPLUS LOW Current count value register
LOW_CVAL	0x38	32	R	0x00	TIMERPLUS HIGH Capture Value Register
HALL_VAL	0x40	32	R	0x00	HALL Signal Raw Value Register

register description

TIMERPLUS_EN register (0x00)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:2	RESERVED	R	0	reserved bit
1	timerplus_hi gh_en	R/W	0	TIMERPLUS High 16bit Timer Enable Register 0: Forbidden Energy 1: Enabling
0	TIMERPLUS_LO W_EN	R/W	0	TIMERPLUS Low 16bit Timer Enable Register 0: Forbidden Energy 1: Enabling

TIMERPLUS_DIV register (0x04)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	TIMERPLUS_ DIV	R/W	0	TIMERPLUS Count Clock Divider Register 0x0000: indicates 1 division 0x0001: indicates 2 divisions 0xFFFF: indicates 65536 division frequency

TIMERPLUS_CTR register (0x08)

bitfie	name (of a	typol	reset	descriptive

DP32G030

ld (mat h.)	thing)	ogy	value	
31:25	RESERVED	R	0	reservations

Reference Manual				
24	HIGH_DMA_EN	R/W	0	DMA Read TIMER HIGH Capture Value Enable 0: CPU reads the captured value 1: DMA Read Capture Value
23	HIGH_PO_MD	R/W	0	TIMER HIGH Periodic Pulse Output Enable 0: Output off 1: Output enable
22:21	HIGH_EXT_EDGE	R/W	0x3	TIMER HIGH Count mode or input capture mode input signal active edge selection 00: Rising edge active 01: Falling edge active 10: Rising or falling edge valid 11: Reservations
20	HIGH_EXT_SEL	R/W	0	TIMER HIGH Count mode or input capture mode input signal selection 0: timer_in0 1: timer_in1
19:18	HIGH_CLKSEL	R/W	0	TIMER HIGH Count Clock Source Selection 00: PCLK/PREDIV (selects pclk pre-divided clock) 01: CNTSRC0 10: CNTSRC1 11: Reservations
17:16	HIGH_MODE	R/W	0	TIMER HIGH Operating Mode Register 00: Timing mode (can generate periodic pulse output signal) 01: Count mode (only pclk can be selected for count clock) 10: Input capture mode 11: Reservations
15:9	RESERVED	R	0	reservations
8	LOW_DMA_EN	R/W	0	DMA Read TIMER LOW Capture Value Enable 0: CPU reads the captured value 1: DMA Read Capture Value

7	LOW_PO_MD	R/W	0	TIMER LOW Periodic Pulse Output Enable 0: Output off 1: Output enable
---	-----------	-----	---	---

				Reference Manual TIMER LOW Count mode or input capture mode Input signal active edge selection 00: Rising edge active 01: Falling edge active 10: Rising or falling edge valid 11: Reservations
6:5	LOW_EXT_EDGE	R/W	0x3	TIMER LOW Count Mode or Input Capture Mode Input Signal Selection 0: timer_in0 1: timer_in1
4	LOW_EXT_SEL	R/W	0	TIMER LOW Count Clock Source Selection 00: PCLK/PREDIV (selects pclk pre-divided clock) 01: CNTSRC0 10: CNTSRC1 11: Reservations
3:2	LOW_CLKSEL	R/W	0	TIMER LOW Operating Mode Register 00: Timing mode (can generate periodic pulse output signal) 01: Count mode (only pclk can be selected for count clock) 10: Input capture mode 11: HALL mode

TIMERPLUS_IE register (0x10)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)				
31:22	RESERVED	R	0	reserved bit
21	HALL2_F_IE	R/W	0	HALL2 Falling edge interrupt enable
20	HALL2_R_IE	R/W	0	HALL2 rising edge interrupt enable

DP32G030

19	HALL1_F_IE	R/W	0	HALL1 Falling edge interrupt enable
18	HALL1_R_IE	R/W	0	HALL1 rising edge interrupt enable
17	HALLO_F_IE	R/W	0	HALLO Falling edge interrupt enable

Reference Manual				
16	HALLO_R_IE	R/W	0	HALLO rising edge interrupt enable
15:11	RESERVED	R	0	reserved bit
10	HIGH_PF_IE	R/W	0	TIMERPLUS HIGH Input pulse falling edge interrupt enable
9	HIGH_PR_IE	R/W	0	TIMERPLUS HIGH Input pulse rising edge interrupt enable
8	HIGH_TO_IE	R/W	0	TIMERPLUS HIGH Reach target value interrupt enable
7:3	RESERVED	R	0	reserved bit
2	LOW_PF_IE	R/W	0	TIMERPLUS LOW Input pulse falling edge interrupt enable
1	LOW_PR_IE	R/W	0	TIMERPLUS LOW Input pulse rising edge interrupt enable
0	LOW_TO_IE	R/W	0	TIMERPLUS LOW Reach target value interrupt enable

TIMERPLUS_IF register (0x14)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)				
31:22	RESERVED	R	0	reserved bit
21	HALL2_F_IF	R/W	0	HALL2 Falling edge interrupt status write 1 clear
20	HALL2_R_IF	R/W	0	HALL2 Rising edge interrupt status write 1 clear
19	HALL1_F_IF	R/W	0	HALL1 Falling edge interrupt status

				write 1 clear Reference Manual
18	HALL1_R_IF	R/W	0	HALL1 Rising edge interrupt status write 1 clear
17	HALLO_F_IF	R/W	0	HALLO Falling edge interrupt status write 1 clear
16	HALLO_R_IF	R/W	0	HALLO Rising edge interrupt status write 1 clear
15:11	RESERVED	R	0	reserved bit

Reference Manual				
10	HIGH_PF_IF	R/W	0	TIMERPLUS HIGH Input pulse falling edge interrupt status write 1 clear
9	HIGH_PR_IF	R/W	0	TIMERPLUS HIGH Input pulse rising edge interrupt status write 1 clear
8	HIGH_TO_IF	R/W	0	TIMERPLUS HIGH Reach target value Interrupt status write 1 clear
7:3	RESERVED	R	0	reserved bit
2	LOW_PF_IF	R/W	0	TIMERPLUS LOW Input pulse falling edge interrupt status write 1 clear
1	LOW_PR_IF	R/W	0	TIMERPLUS LOW Input pulse rising edge interrupt status write 1 clear
0	LOW_TO_IF	R/W	0	TIMERPLUS LOW Reach target value Interrupt status write 1 clear

TIMERPLUS_HIGH_LOAD register (0x20)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (math.)	RESERVED	R	0	reserved bit
15:0	HIGH_LOAD	R/W	0xffff	<p>TIMERPLUS HIGH Timer Target Configuration Register</p> <p>When the high 16bit counter counts up to the set value, the corresponding status signal is generated.</p>

Note: The HIGH counter can generate the corresponding interrupt status (HIGH_TO_IF) and external trigger signals (e.g., the timer_goal[1] signal in the block diagram of the module structure) when it reaches the value configured in this register.

TIMERPLUS_HIGH_CNT register (0x24)

DP32G030
Reference Manual

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	HIGH_CNT	R	0	TIMERPLUS HIGH Timer Current Count Value

TIMERPLUS_HIGH_CVAL register (0x28)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	HIGH_CVAL	R	0	TIMERPLUS HIGH Capture value count value

TIMERPLUS_LOW_LOAD register (0x30)

bitfield ld (mat h.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	LOW_LOAD	R/W	0xffff	<p>TIMERPLUS LOW Timer Target Configuration Register</p> <p>When the low 16bit counter counts up to the set value, the corresponding status signal is generated.</p>

Note: The LOW counter can generate the corresponding interrupt status (LOW_TO_IF) and external trigger signals (e.g., the timer_goal[0] signal in the block diagram of the module structure) when it reaches the value configured in this register.

TIMERPLUS_LOW_CNT register (0x34)

bitfield d	name (of a thing)	typol ogy	reset value	descriptive

(math.)				
31:16	RESERVED	R	0	reserved bit
15:0	LOW_CNT	R	0	TIMERPLUS LOW Timer Current Count Value

TIMERPLUS_LOW_CVAL register (0x38)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	LOW_CVAL	R	0	TIMERPLUS LOW Capture value count value

HALL_VAL register (0x40)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:3	RESERVED	R	0	reserved bit
2	HALL2_VAL	R	0	Raw signal level of HALL2
1	HALL1_VAL	R	0	Raw signal level of HALL1
0	HALLO_VAL	R	0	Raw signal level of HALLO

5.11 Independent Watchdog Clock (IWDT)

5.11.1 summarize

The Independent Watchdog Timer (IWDT) is mainly used to control the correct program flow and reset the chip if the program flow does not execute the specified program according to the established flow for a long time. The corresponding IWDT module clock needs to be enabled before use. The system block diagram is shown below:

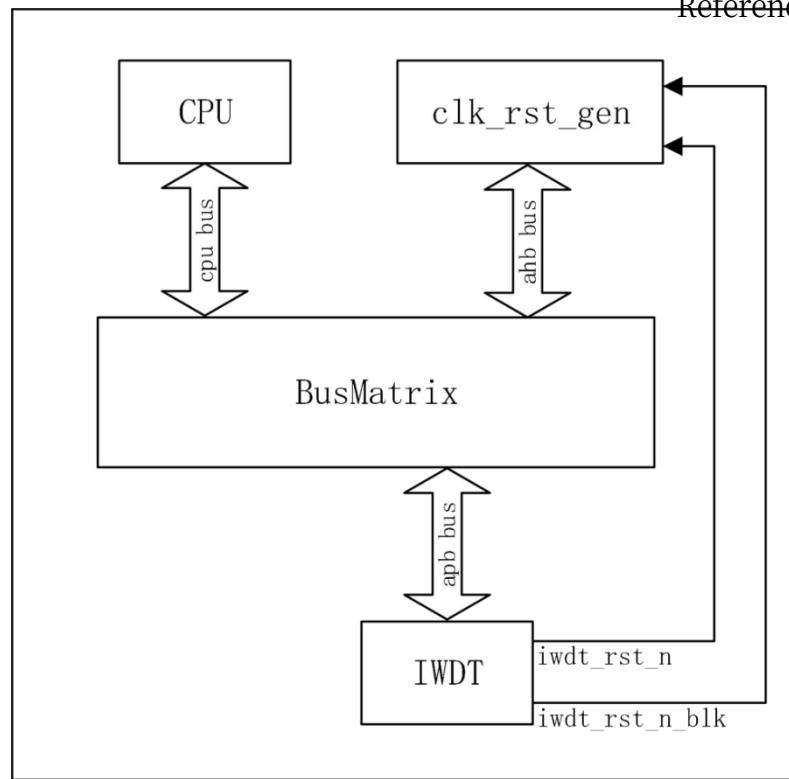


Figure 5-54 IWDT System Block Diagram

5.11.2 characterization

- Generate counter overflow reset signal, reset signal enable can be configured
- Flexible, wide-range overflow cycles configurable with 32-bit count bit widths
- Has an interrupt function that provides a periodic count overflow interrupt signal that generates a Watchdog reset signal if the interrupt signal is not cleared before the next interrupt signal is generated.
- With dog feeding function

5.11.3 Block Diagram of Module Structure

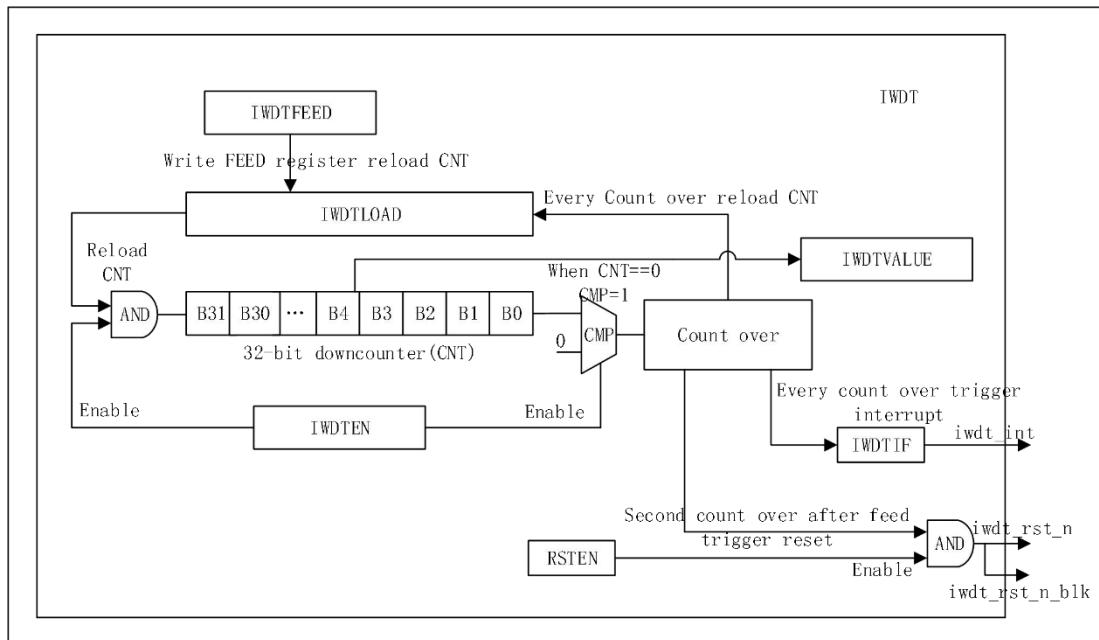


Figure 5-55 IWDT Module Block Diagram

The above figure shows the structure block diagram of the independent watchdog clock module, this module has a 32bit downward counting counter, the initial value of the counter can be configured through the register **IWDTLOAD**, and through the counter enable **IWDTE** to control the start and stop of the counter, when the counter counts to 0, it will generate an interrupt signal, **iwdt_int** interrupt signal is generated, if there has been no response, then when the counter counts to 0 again, if the **RSTEN** bit in the **IWDTControl** register is 1, the reset signal is valid, otherwise when **RSTEN** is 0, the reset signal is invalid. After the counter reaches 0, an interrupt signal will be generated. **iwdt_int** interrupt signal is generated, if there has been no response after the interrupt signal is generated, then when the counter counts to 0 again, the reset signal will be valid if the **RSTEN** bit in the **IWDTControl** register is 1. Otherwise, the reset signal will not be valid when the **RSTEN** is 0, and the signal can be used to generate a system reset.

5.11.4 Functional Description

disruption generation

DP32G030
Reference Manual

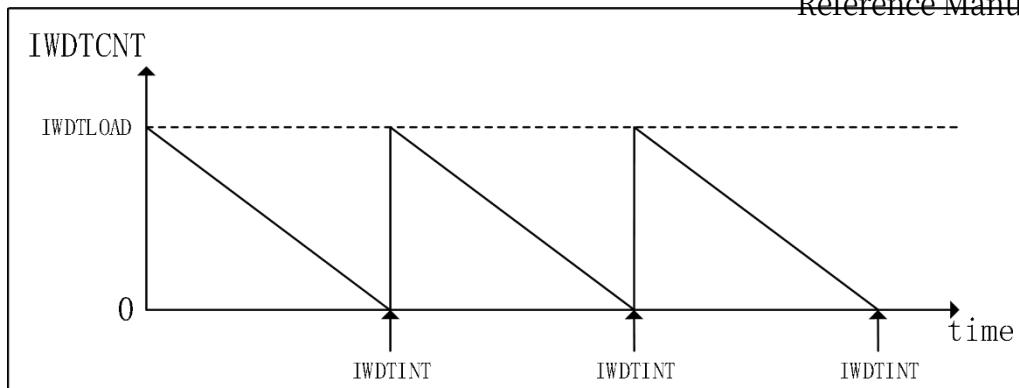


Figure 5-56 IWDT Interrupt Generation Diagram

IWDT built-in 32BIT counter IWDTCNT, when the watchdog enable IWDTEN is active, IWDTCNT will load the set IWDTLOAD value and start counting down, whenever the IWDTCNT count value reaches 0, it will generate IWDT interrupt status WDT_IF signal, and it will generate system interrupt directly.

Note that the interrupt status is generated only with respect to the current count value of the IWDTCNT, and is generated whenever the count value reaches zero.

When the interrupt is not cleared until the count value reaches 0 again, if the RSTEN bit is valid at that time, a watchdog reset will be generated and IWDTCNT will be cleared.

IWDT Feed Dog and Reset Generation

1. Feed the dog before the first interruption

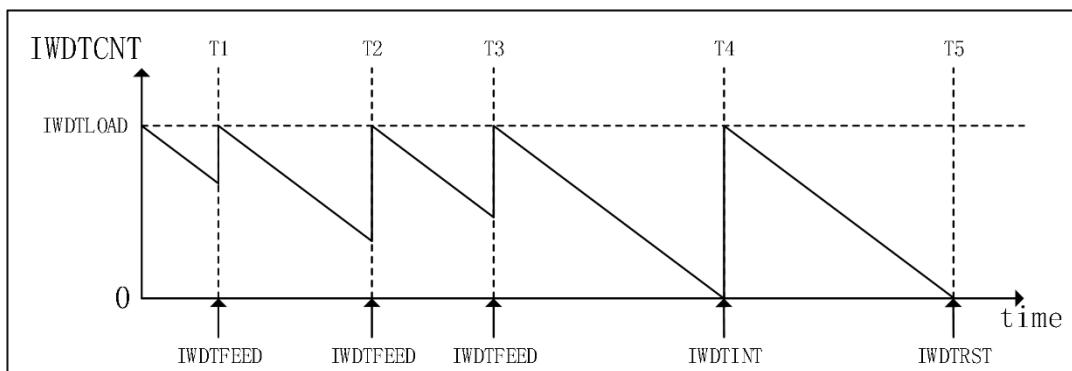


Figure 5-57 IWDT Feed Dog and Reset Before First Interrupt

As shown above, when the IWDTCNT counter has not reached 0 for the first time for a watchdog feed, the counter will start counting again from IWDTLOAD (T1, T2, T3 moments) When the IWDTCNT generates two interrupts (T4, T5 moments) after the last dog feed (T3 moment) and RSTEN is valid, a watchdog reset IWDTRST is generated (T5 moment)

The watchdog reset is generated only when the second interrupt is generated after feeding the dog, and the number of interrupts is re-recorded if another dog feeding operation is performed before the second interrupt is generated.

2. Feed the dog before the second interruption

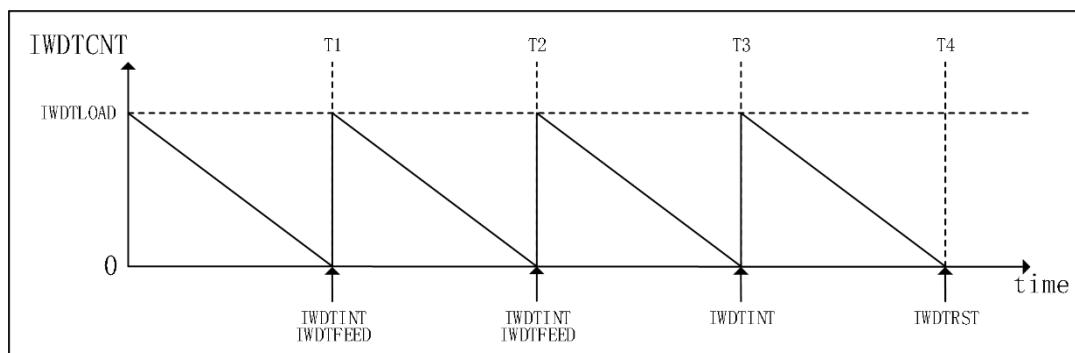


Figure 5-58 IWDT Feed Dog and Reset Before Second Interrupt

As shown in the figure above, when the IWDTCNT counter reaches 0 once for a watchdog feed, the counting will start again from IWDTLOAD (T1, T2 moments) When the IWDTCNT generates two interrupts (T3 and T4 moments) after the last dog feed (T2 moment) and RSTEN is valid, a watchdog reset IWDTRST is generated (T4 moment)

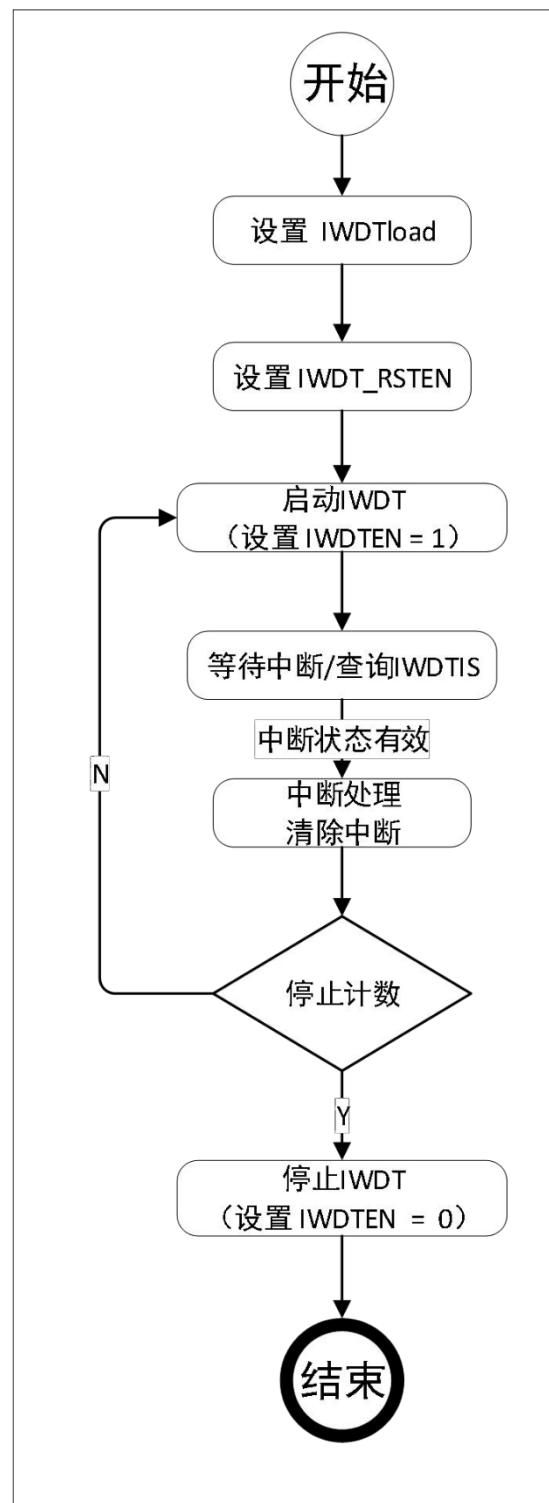


Figure 5-59 IWDT Interrupt Generation Operation Flowchart

- IWDT Clock Enable
- Configuration initial value register (IWDTLOAD)
- Configuration Reset Enable Register (RSTEN)
- Configuration Enable Register (IWDTEN)
- Waiting for interrupt generation and handling interrupts
- If the interrupt is not handled by the time the IWDTValue counts to 0 again, a watchdog reset is generated

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
IWDTBASE: 0x4006A000					
IWDT_LOAD	0x00	32	R/W	0xfffff	IWDT Initial Value Register
IWDT_CTRL	0x08	32	R/W	0x00	IWDT Control Register
IWDT_IF	0x0C	32	R/W	0x00	IWDT Status Register
IWDT_FEED	0x10	32	R/W	0x00	IWDT Feed Dog Register

register description

IWDT_LOAD register (0x00)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:20	RESERVED	RO	0	reserved bit

19:0	IWDTLOAD	R/W	0xfffff	Initial value configuration register for the IWDT counter. When the IWDT starts, the counter is automatically loaded with the IWDTLOAD value and starts counting down. When the counter value reaches 0, the hardware automatically reloads the value in the IWDTLOAD register into the counter to continue the decremental count. When the first count reaches 0, an interrupt is generated. If the dog is not fed, the When the count is reduced to 0 again, a reset is generated. This register must be configured when IWDTEN is invalid.
------	----------	-----	---------	---

IWDT_CTRL register (0x08)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:2	RESERVED	RO	0	reserved bit
1	INTEN	R/W	0	IWDT interrupt enable bit 1: Enabling 0: Prohibited
0	IWDTEN	R/W	0	IWDT start bit 1: Initiate IWDT counting 0: stop counting

IWDT_IF register (0x0C)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:1	RESERVED	RO	0	reserved bit
0	IWDT_IF	R/W	0	IWDT status bit, count to 0, high valid hardware set, software write 1 clear

IWDT_FEED register (0x10)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	RO	0	reserved bit

DP32G030

IWDT Restart Counter Registers Reference Manual

7:0	FEED	R/W	0	IWDT Restart Counter Registers Writing 0x55 to this register restarts the IWDT counter(feed dog operation)
-----	------	-----	---	--

5.12 Window Watchdog Clock (WWDT)

5.12.1 summarize

The Window Watchdog Timer (WWDT) is mainly used to control the correct program flow and reset the chip if the program flow does not execute the specified program according to the established flow for a long time. The corresponding WWDT module clock needs to be enabled before use.

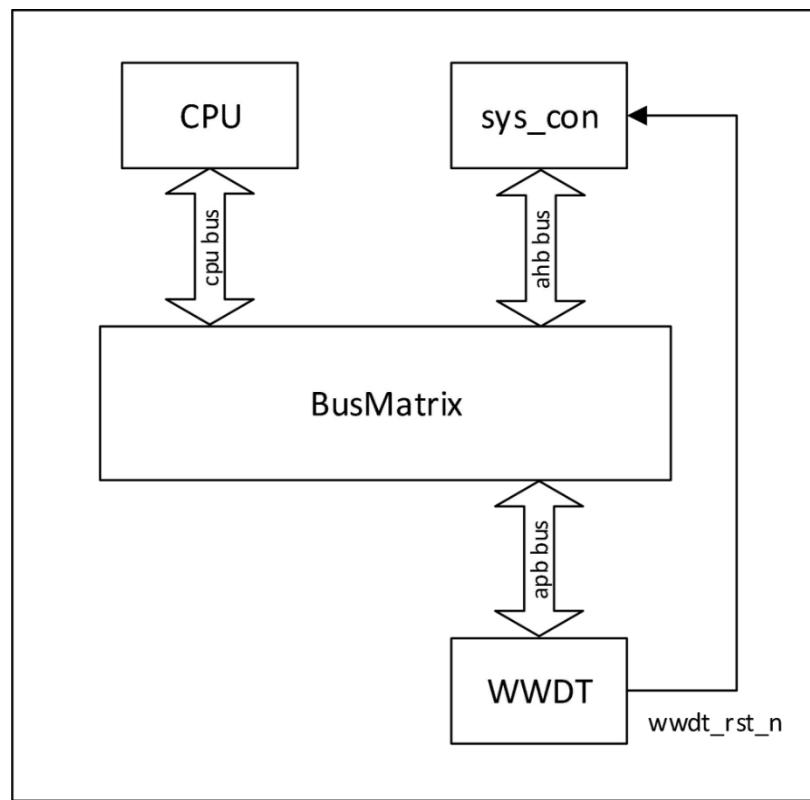


Figure 5-60WWDT Module System Block Diagram

5.12.2 characterization

- The watchdog count clock source is the 4096 division of the system clock.
- Functional features: Support 7-bit decrement count, provide window interrupt and pre-reset interrupt, generate reset signal if the dog feeding operation is still not performed within one watchdog counting cycle after the pre-reset

interrupt.

- Supports independently configurable interrupt window value and reset window value respectively
- Watchdog count clock frequency is configurable

- Supports pre-reset interrupt alarm function

5.12.3 Block Diagram of Module Structure

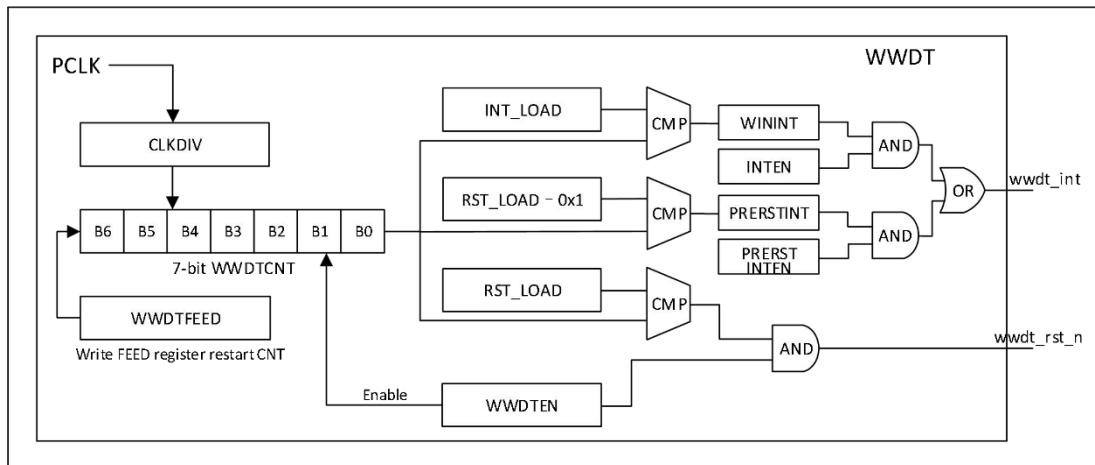


Figure 5-61 WWDT Module Block Diagram

WWDT has a built-in 7-bit counter, and the count clock is the clock after PCLK has been divided by CLKDIV.

1, enable the counter, the counter loads the count initial value 0x7f from the register, turn on the decrement count.

When the count value reaches the INT_LOAD setting, WWDT generates the WININT interrupt state, and if the interrupt enable is enabled, WWDT generates the WININT interrupt state.

INTEN, a system interrupt wwdt_int occurs.

When the count value reaches the RST_LOAD-1 setting, the WWDT generates the PRERSTINT interrupt state, and if the interrupt enable PRERSTINTEN is turned on, the system interrupt wwdt_int occurs.

When the count value reaches the RST_LOAD setting, a watchdog reset wwdt_rst_n is generated.

A watchdog feed occurs by writing to WWDTFEED register 0x55, causing the counter to turn counting back on from value 0x7f.

5.12.4 Functional Description

WWDT interrupt generation and dog feeding interval

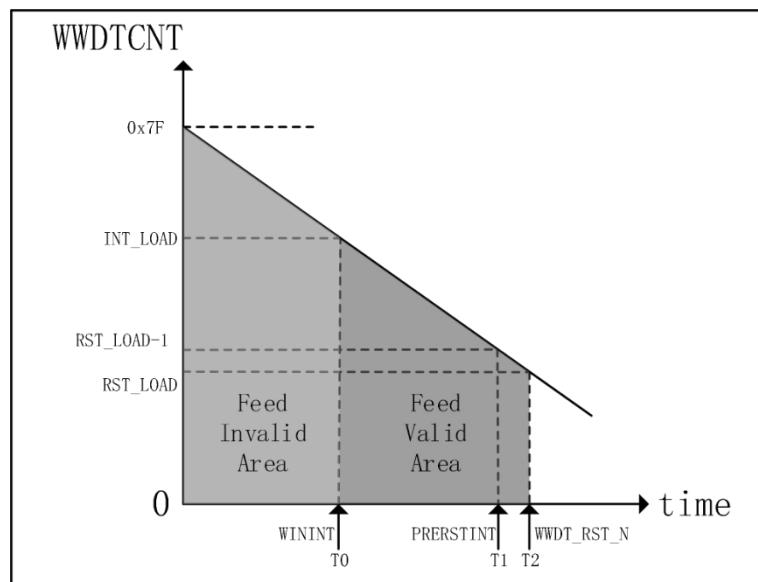


Figure 5-62WWDT Interrupt Generation and Dog Feeding Interval

As shown above:

When the WWDTCNT count value reaches INT_LOAD (T0 time) the WININT interrupt state is generated;

When the WWDTCNT count value reaches RST_LOAD-1 (T1 time) the PRERSTINT interrupt state is generated;

A WWDTCNT watchdog reset is generated when the WWDTCNT count value reaches RST_LOAD (T2 time)

WWDT interrupt generation is only related to the current count value.

From time 0 to time T0 is the invalid dog feeding interval, if the dog is fed at this time, a watchdog reset will be generated immediately;

From T0 time to T2 time is the valid dog feed interval, if the dog feed at this time will reset the WWDTCNT watchdog counter.

WWDT Feed Dog and Reset Generation

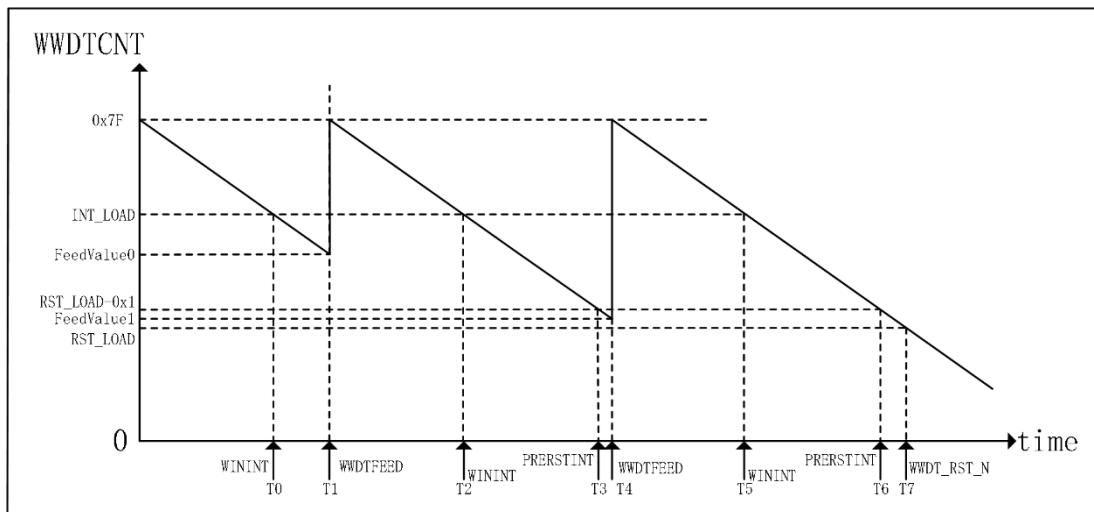


Figure 5-63 WWDT Dog Feed and Reset

As shown above:

When the WWDTcnt count value reaches INT_LOAD (T0 time) the WININT interrupt state is generated;

When the WWDTcnt count value reaches FeedValue0 (T1 time) watchdog feeding is performed and the counter starts counting again;

When the WWDTcnt count value reaches INT_LOAD (T2 time) the WININT interrupt state is generated;

When the WWDTcnt count value reaches RST_LOAD-1 (T3 time) the PRERSTINT interrupt state is generated;

When the WWDTcnt count value reaches FeedValue1 (T4 time) watchdog feeding is performed and the counter starts counting again;

When the WWDTcnt count value reaches INT_LOAD (T5 time) the WININT interrupt state is generated;

When the WWDTcnt count value reaches RST_LOAD-1 (T6 time) the PRERSTINT interrupt state

DP32G030
Reference Manual

is generated;

When the WWDTCNT count value reaches RST_LOAD (T7 time) a WWDT_RST_N watchdog reset is generated.

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
WWDTBASE: 0x4006A800					
WWDT_LOAD	0x00	32	R/W	0x40	WWDT Initial Value Register
WWDT_VALUE	0x04	32	R/W	0x7f	WWDT Current Count Register
WWDT_CTRL	0x08	32	R/W	0x00	WWDT Control Register
WWDT_IF	0x0C	32	R/W	0x00	WWDT Interrupt Status Register
WWDT_FEED	0x10	32	R/W	0x00	WWDT Feed Dog Register

register description

WWDT_LOAD register (0x00)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:14	RESERVED	RO	0	reserved bit
13:8	RST_LOAD	R/W	0x0	Window Reset Comparison Value Register Note 1: A reset can be generated when the watchdog counts down to the value Note 2: The watchdog can generate an interrupt when it counts down to this value plus one.
7	RESERVED	RO	0	reserved bit

6:0	INT_LOAD	R/W	0x40	Window Interrupt Compare Value Register Note 1: When configuring, the window interrupt comparison value must be greater than the window reset comparison value. Note 2: The watchdog can generate an interrupt when it counts down to this value.
-----	----------	-----	------	---

WWDT_VALUE register (0x04)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:7	RESERVED	RO	0	reserved bit

6:0	VALUE	R	0x7f	This register is a read only register with a reset value of 0x7f. reading this register returns the current count value of the counter.
-----	-------	---	------	---

WWDT_CTRL register (0x08)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:5	RESERVED	RO	0	reserved bit
4	PRERSTINTEN	R/W	0	Pre-reset interrupt enable bit 1: Enabling 0: Prohibited
3	INTEN	R/W	0	Window interrupt enable bit 1: Enabling 0: Prohibited
2:1	CLKDIV	R/W	0	Watchdog Counter Count Clock Presharing 00: Watchdog count clock 1 division 01: Watchdog count clock 2 divisions 10: Watchdog Count Clock 4 divisions 11: Watchdog Count Clock 8 divisions Note: The watchdog count clock is a 4096 division of the system clock.
0	EN	R/W	0	WWDT start bit 1: Initiate WWDT counting 0: stop counting

WWDT_IF register (0x0C)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive

31:2	RESERVED	RO	0	reserved bit Reference Manual
1	PRERSTINT	R/W	0	Pre-reset interrupt flag bit (i.e. VALUE = RST_LOAD + 1) time) high effective Hardware set, software write 1 clear
0	WININT	R/W	0	Window interrupt flag bit (i.e., when VALUE = INT_LOAD) valid high Hardware set, software write 1 clear

WWDT_FEED register (0x10)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	RO	0	reserved bit
7:0	FEED	W	0	WWDT Restart Counter Registers Writing 0x55 to this register will restart the WWDT counter (dog feeding operation) Note: Feeding the dog will only take effect during the window period (INT_LOAD > VALUE > RST_LOAD), otherwise the dog feeding operation will generate a watchdog reset.

5.13 Basic Pulse Width Modulation Generator (PWMBASE)

5.13.1 summarize

PWMBASE is a basic pulse width modulation generator. The PWMBASE module provides 3 independent channels (CH0, CH1, CH2), supports pre-scaler function, output level flip-flop, flip-flop interrupt and end-of-cycle interrupt. Before using the PWMBASE module, it is necessary to enable the PWMBASE clock, and the PWMNASE module is connected to the CPU through the APB bus, and can generate interrupts and output PWM waveforms to the PAD port, and its system block diagram is shown in the following figure:

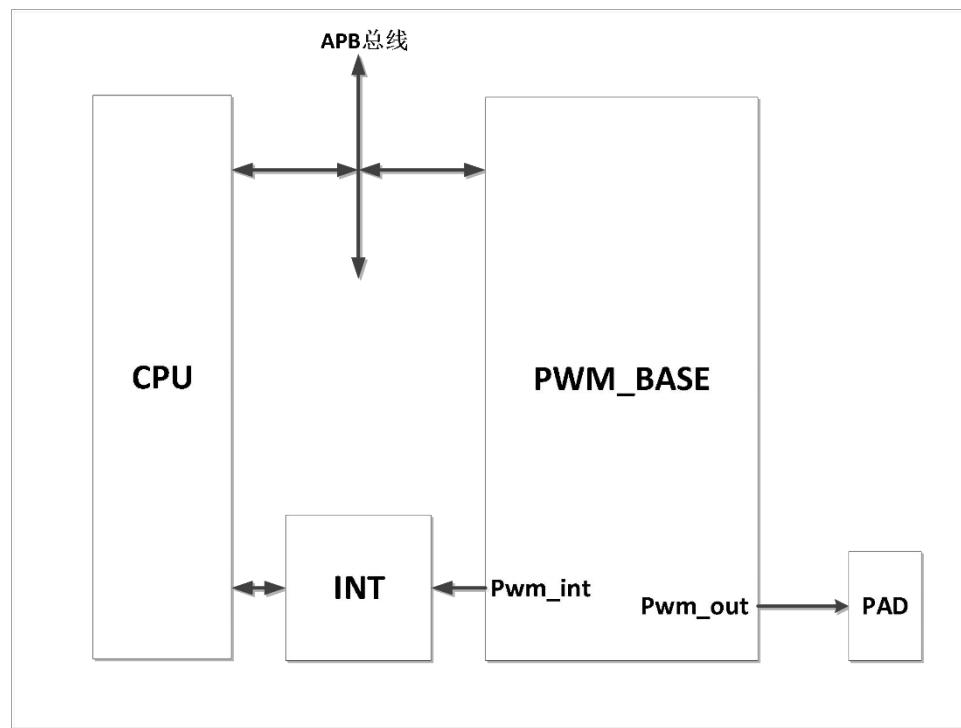


Figure 5-64 PWMBASE Module System Block Diagram

5.13.2 characterization

- 3 16bit PWM, can output PWM waveforms with different duty cycles
- 8bit prescaler counter
- Whether the output level is flipped or not

- Supports reach flip point interrupt and end of cycle interrupt

- Internal down counter

5.13.3 Block Diagram of Module Structure

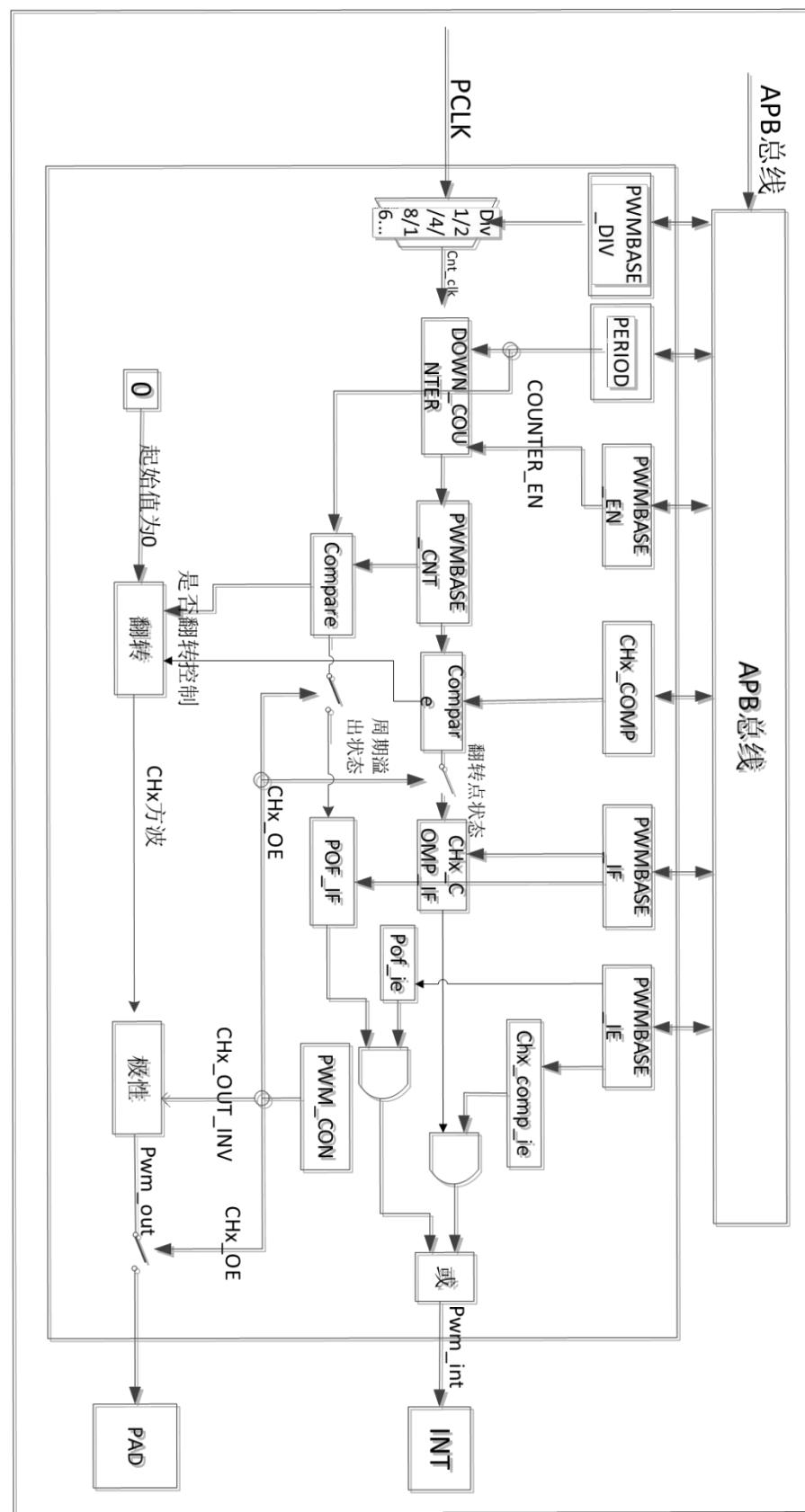


Figure 5-65 PWMBASE Module Block Diagram

The above figure shows the framework of PWMBASE module structure, from which it can be seen that the system clock `pclk` can be divided by 8bit `div` pre-divided frequency register, and then provide counting clock for the downward counting counter after dividing frequency. The counter `period` is configured through the period configuration register, and the counter enable `counter_en` controls the start and end of the count. After the counting starts, the current counting value will exist in `PWM_CNT` register, and the current counting value will be compared with the configured rollover point and period value, because the starting level of the channel output is 0, and it is a downward counting counter, so when the counting value is less than the rollover point value, the output is 1, and greater than or equal to the rollover point value, the output is 0, and when the counting is full, the waveform will be flipped again. This module configures the PWMBASE waveform output through the PWMBASE output configuration register, in which `chx_out_inv` controls the polarity of the channel waveform, when `chx_out_inv` is 1, the output is opposite to the original waveform, and when `chx_out_inv` is 0, the output is the same as the original waveform; `chx_oe` controls the channel waveform output enable, and when `chx_oe` is 1, the PWMBASE waveform output is 1, the PWMBASE waveform output is 1, the PWMBASE waveform output is 1. `chx_oe` controls the channel waveform output enable, when `chx_oe` is 1, PWMBASE outputs channel waveform, when `chx_oe` is 0, PWMBASE outputs high resistance state.

The generation of interrupt signals is controlled by the interrupt state and interrupt enable. The interrupt state is divided into the flip-point state and the cycle overflow state, and the interrupt state can only be generated when `chx_oe` is 1. In the case of interrupt state generation, the corresponding interrupt enable is turned on, i.e., the flip-point interrupt enable and the cycle overflow interrupt enable of each channel, and the interrupt signals are generated only when the interrupt state is generated.

5.13.4 Functional Description

Periodic values and rollover points

When the pass-through output enable register `chx_oe` is 1, different period values and flip-point values of the output waveform can be configured through registers `PWMBASE_PERIOD` and `PWM_CHX_COMP`, as shown in the following figure with the period value configured as 7 and the timing diagram of different flip-point values.

Example of configuring the relationship between the cycle value and the flip point value (idle start level is 0)

- 1) The user wants to generate a PWM waveform with period 8 and duty cycle 25%, then it needs to be configured as: `PERIOD=0x07, CHx_COMP=0x02`.
- 2) The PWM waveform that the user wants to generate is: period 8, duty cycle is 87.5%, then it needs to be configured as: `PERIOD =0x07, CHx_COMP=0x07`.
- 3) The user wants to generate a PWM waveform with period 8 and duty cycle 100%, then it needs to be configured as: `PERIOD =0x07, CHx_COMP=0x08` (greater than 7 is fine)

- 4) The user wants to generate a PWM waveform with period 8 and duty cycle 0%, then it needs to be configured as: PERIOD =0x07, CHx_COMP=0x00.

The following schematic example realizes the output waveform for the above relationship between the period value and the flip point value.

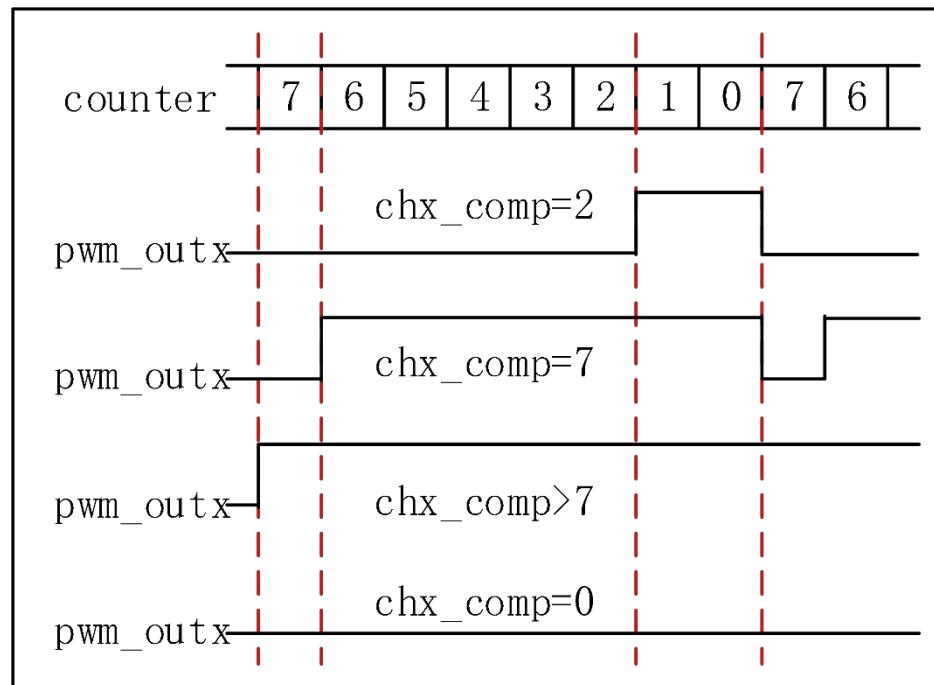
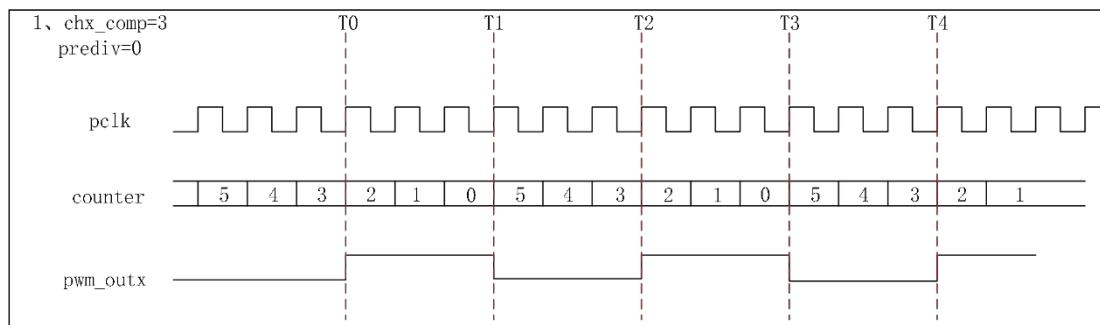


Figure 5-66 PWMBASE Timing Diagram for Different Cycles and Flip Points

preshared count

This module can be configured with different clock division values through the 8bit prescaler counter register `pwmbase_div` to realize the division frequency counting, and the division frequency value range is 1-256. In the following figure, the timing diagram is given as an example when the flip-flop value is 3, the period value is 6, and the division frequency value is 1, 2, 6.



DP32G030

Figure 5-67 PWMBASE 1 Frequency Division Timing Diagram
Reference Manual

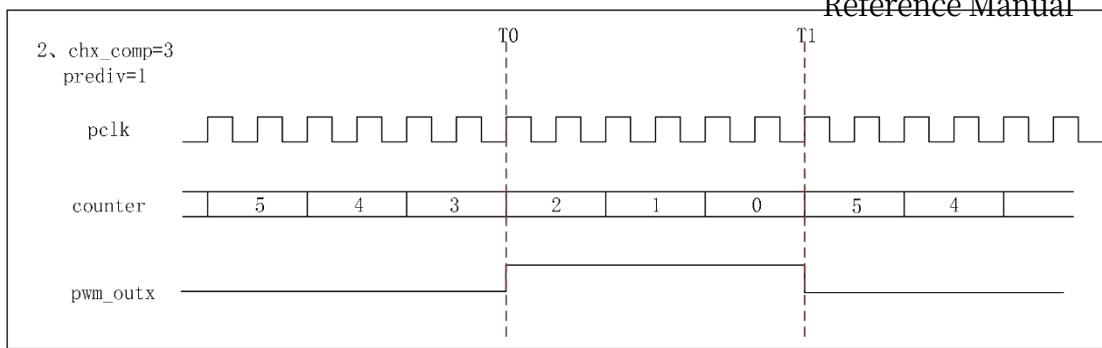


Figure 5-68 PWMBASE 2 Frequency Division Timing Diagram

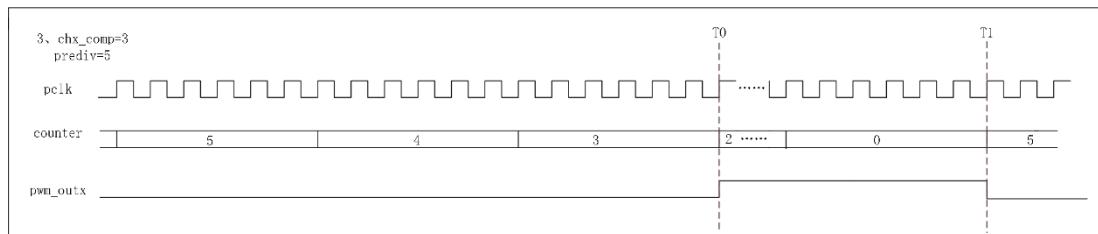


Figure 5-69 PWMBASE 6 Division Timing Chart

output enable (computing)

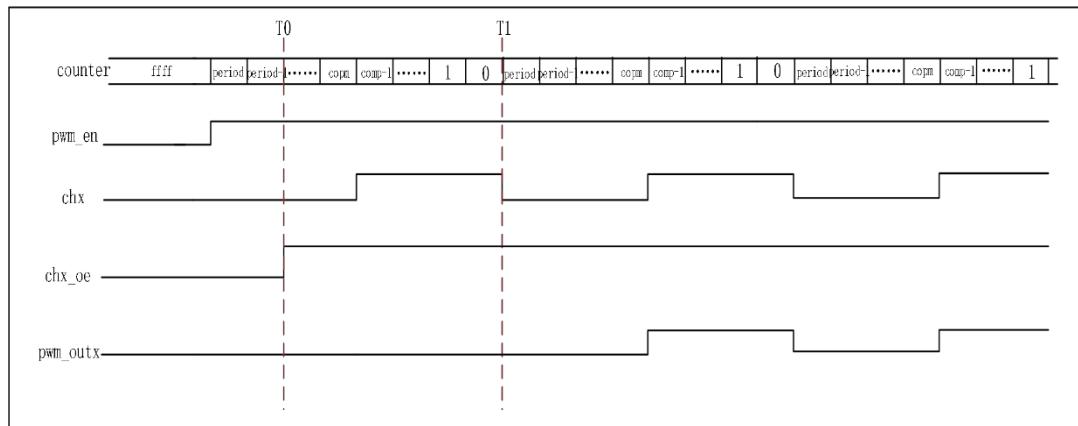


Figure 5-70 PWMBASE Output Enable Timing Chart

The above figure shows the output enable timing diagram of PWMBASE, in which **chx** is the original waveform, **chx_oe** is the channel waveform output enable, and **pwm_outx** is the waveform output from **pwm**. Before the channel waveform enable, i.e., before the **T0** time, **pwm** outputs the starting level of 0. When **chx_oe** of

the channel waveform enable register is changed to 1 in the T
output the waveform immediately, but Output pass

The initial level of the channel is 0, and the channel waveform will be output only from the next T1 moment to ensure the complete waveform output. It should be noted that before the channel waveform enable is turned on, the output of the corresponding PAD port is in high resistance state, and only after the channel waveform enable is turned on, the corresponding PAD port outputs the waveform of `pwm_outx`.

output flip-flop

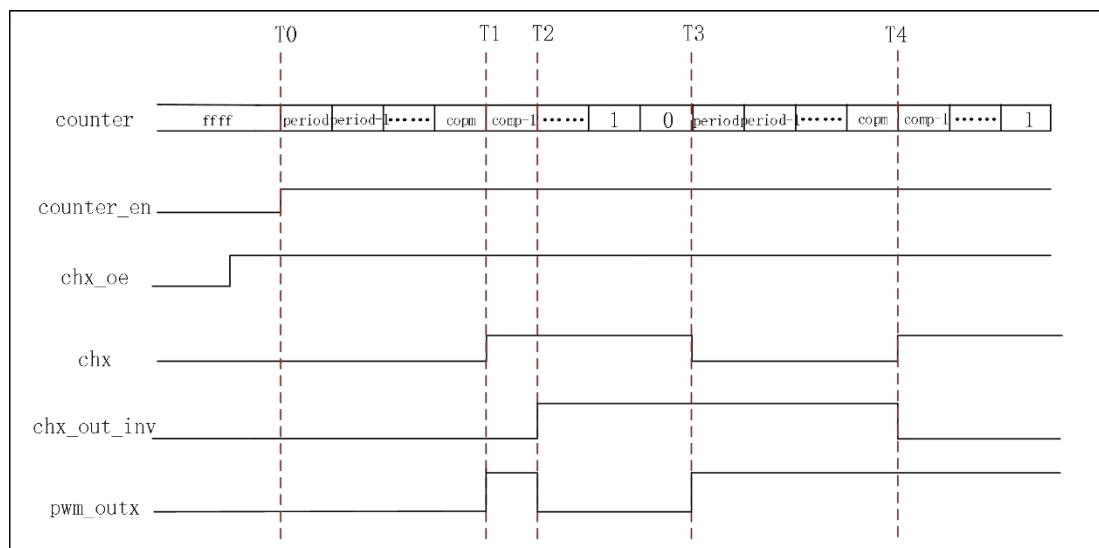


Figure 5-71 PWMBASE Output Flip-Flop Timing Chart

As shown in the above figure, when the counter enable and channel waveform output enable are turned on at the same time, when `chx_out_inv` is configured to be 0, we can see that the waveform output from the `pwm` is the same as the original waveform in the time period of T0-T2, and the waveform output from the `pwm` will be the same as the original waveform when `chx_out_inv` is configured to be 1 in the time period of T2, and the waveform output from the `pwm` will be the same as the original waveform with the opposite polarity. On the contrary, when `chx_out_inv` is configured to 0 at T4, the waveform of `pwm` output will be consistent with the original waveform again.

disruptions

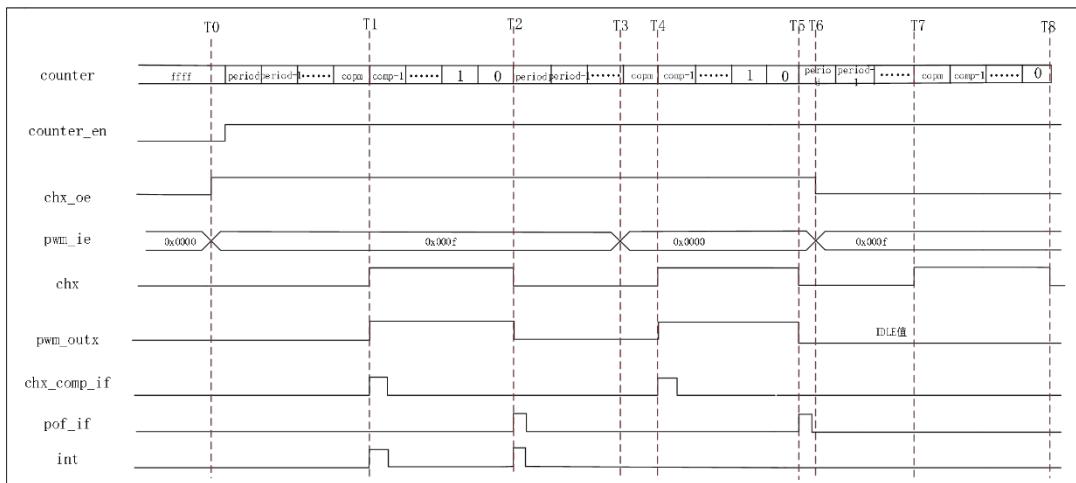


Figure 5-72 PWMBASE Output Interrupt Timing Chart

This module supports many kinds of interrupts, as shown in the above figure, at T_0 time, configure the channel waveform output enable chx_oe to 1, and at the same time, configure the interrupt enable register pwm_ie to $0x000f$, which is to turn on the flip-point interrupt and cycle overflow interrupt of each channel. Turn on the counter enable, at T_1 time, when the count value reaches the flip-flop, the flip-flop interrupt status register chx_comp_if will be pulled high, and at the same time generate interrupt signals, and then process the interrupt, and the interrupt status register will be cleared by writing 1; when the count value is 0, it means that the counting is one cycle, and at this time, the cycle overflow status will be changed to 1, such as the T_2 time in the figure, and generate interrupt signals, and then process the interrupt, and the interrupt status will be cleared by writing 1. Clear the interrupt status. When pwm_ie is configured to $0x0000$ at T_3 , the flip-flop and cycle overflow interrupt enable is turned off, and only the corresponding interrupt status will be generated at T_4 and T_5 , and no interrupt signal will be generated. If the flip-flop and cycle overflow interrupts are turned on again at T_6 and the channel waveform output enable chx_oe is configured to 0, even though the interrupt enable is turned on, the flip-flop and cycle overflow interrupts will not be generated at T_7 and T_8 , and therefore no interrupt signal will be generated.

workflow

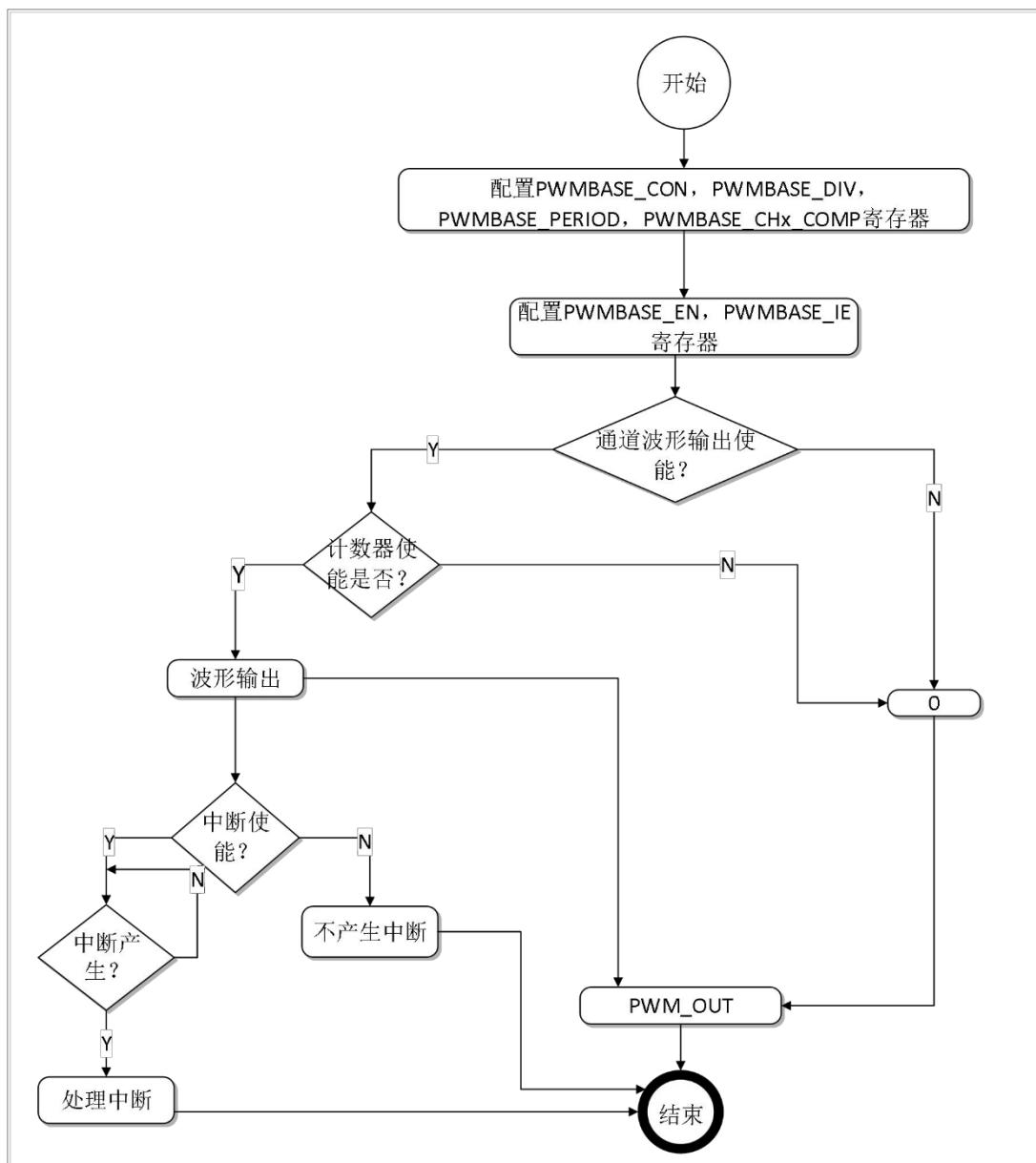


Figure 5-73 PWMBASE Operation Flowchart

- Configuring PWMBASE Clock Enable
- The PORT port is configured for PWMBASE functionality.
- Configuring the prescaler (PWMBASE_DIV) registers
- Configure the output flip-flop (PWMBASE_CON) registers
- Configuration Interrupt Enable (PWMBASE_INT_EN) registers

- Configuration cycle (PWMBASEx_PERIOD) and flip-flop (PWMBASEx_CHx_COMP) registers
- If interrupts are configured, enable PWMBASE interrupts
- Configuration Waveform Output Enable (PWMBASE_CON) Registers
- Configure the PWMBASE enable bit (PWMBASE_EN) to enable PWMBASE

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
PWMBASE0BASE: 0x400B1000					
PWMBASE1BASE: 0x400B1800					
PWMBASE_EN	0x00	32	R/W	0x00	PWMBASE Enable Register
PWMBASE_DIV	0x04	32	R/W	0x00	PWMBASE Clock prescaler register
PWMBASE_CON	0x08	32	R/W	0x00	PWMBASE Output Configuration Register
PWMBASE_PERIOD	0x0C	32	R/W	0x00	PWMBASE Cycle Configuration Register
PWMBASE_IE	0x10	32	R/W	0x00	PWMBASE Interrupt Enable Register
PWMBASE_IF	0x14	32	R/W	0x00	PWMBASE Interrupt Status Register
PWMBASE_CNT	0x18	32	R/W	0xffff	PWMBASE Current Count Register
PWMBASE_CH0_COMP	0x20	32	R/W	0x00	PWMBASE Channel 0 Flip Point Configuration Register
PWMBASE_CH1_COMP	0x30	32	R/W	0x00	PWMBASE Channel 1 Flip Point Configuration Register
PWMBASE_CH2_COMP	0x40	32	R/W	0x00	PWMBASE Channel 2 Flip Point Configuration Register

register description

PWMBASE_EN register (0x00)

bitfield	name (of a	typolo	reset	descriptive
----------	------------	--------	-------	-------------

(math.)	thing)	gy	value	
31:1	RESERVED	R	0	reserved bit

PWMBASE Counter Enable Register				
0	COUNTER_EN	R/W	0	PWMBASE Counter Enable Register When this bit is set to 1, it means that the counter starts counting, and CH0/CH1/CH2 will generate the PWM output waveform of the corresponding channel according to the pre-configured period value and comparison value. When this bit is configured to 0, it means the counter stops counting, and the output waveform level returns to the initial 0 level state after one full counting cycle.

PWMBASE_DIV register (0x04)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PWMBASE_DIV	R/W	0	PWMBASE Count Clock Prescaler Register 0x0000: indicates 1 division 0x0001: indicates 2 divisions 0x0002: Indicates 3 divisions 0xFFFF: indicates 65536 division frequency

PWMBASE_CON register (0x08)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:7	RESERVED	R	0	reserved bit
6	CH2_OE	R/W	0	CH2 Channel waveform output enable 0: Output off, high resistance state on pin 1: Output CH2 square wave

5	CH1_OE	R/W	0	CH1 Channel waveform output enable 0: Output off, high resistance state on pin 1: Output CH1 square wave
4	CH0_OE	R/W	0	CH0 Channel waveform output enable 0: Output off, high resistance state on pin 1: Output CH0 square wave
3	RESERVED	RO	0	reserved bit

Reference Manual				
2	CH2_OUT_INV	R/W	0	CH2 Output polarity flip-flop or not register 0: No change, CH2 output waveform is the original waveform 1: Polarity flip, CH2 output waveform is original waveform flip
1	CH1_OUT_INV	R/W	0	CH1 Output polarity flip-flop or not register 0: No change, CH1 output waveform is the original waveform 1: Polarity flip, CH1 output waveform is original waveform flip
0	CH0_OUT_INV	R/W	0	CH0 Output polarity flip-flop or not register 0: no change, CH0 output waveform is the original waveform 1: Polarity flip, CH0 output waveform is original waveform flip

PWMBASE_PERIOD register (0x0C)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PWMx_PERIOD	R/W	0	PWMx Output Period Configuration Register The actual counting period is the value of the period configured for this register plus 1. Note 0: The period cannot be configured as 0. For example, if the configuration is decimal 199, the PWM waveform period is considered to be 200.

PWMBASE_INTEN register (0x10)

bitfield ld	name (of a thing)	typology	reset value	descriptive
----------------	----------------------	----------	----------------	-------------

(mat h.)				
31:4	RESERVED	R	0	reserved bit
3	POF_IE	R/W	0	Cycle overflow interrupt enable
2	CH2_COMP_IE	R/W	0	CH2 Reach flip point interrupt enable
1	CH1_COMP_IE	R/W	0	CH1 Reach flip point interrupt enable
0	CH0_COMP_IE	R/W	0	CH0 Reach flip point interrupt enable

PWMBASE_IF register (0x14)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:4	RESERVED	R	0	reserved bit
3	POF_IF	R/W	0	Cycle Overflow Interrupt Status Write 1 Clear
2	CH2_COMP_IF	R/W	0	CH2 Reach Flip Point Status Write 1 Zero Clear
1	CH1_COMP_IF	R/W	0	CH1 Reach flip point status write 1 clear
0	CH0_COMP_IF	R/W	0	CH0 Reach flip point status write 1 clear

PWMBASE_CNT register (0x18)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	PWMBASE_CNT	R	0xffff	PWMBASE Counter Current Count Value Register

PWMBASE_CH0_COMP register (0x20)

bitfield d (math)	name (of a thing)	typol ogy	reset value	descriptive

.)				
31:16	RESERVED	R	0	reserved bit
15:0	CH1_COMP	R/W	0	CH0 Flip Point Configuration Register Note: If the count value is less than the rollover point value, output 1; if it is greater than or equal to the rollover point value, output 0.

PWMBASE_CH1_COMP register (0x30)

bitfield	name (of a thing)	typology	reset value	descriptive
d (math .)				
31:16	RESERVED	R	0	reserved bit

					Reference Manual CH1 Flip Point Configuration Register
bitfield d (math .)	name (of a thing)	typology	reset value	descriptive	
15:0	CH1_COMP	R/W	0		Note: If the count value is less than the rollover point value, output 1; if it is greater than or equal to the rollover point value, output 0.

PWMBASE_CH2_COMP register (0x40)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	CH2_COMP	R/W	0	CH2 Flip Point Configuration Register Note: If the count value is less than the rollover point value, output 1; if it is greater than or equal to the rollover point value, output 0.

5.14 Advanced Pulse Width Modulation Generator (PWMPLUS)

5.14.1 summarize

PWMPLUS is an advanced PWM module. The PWMPLUS supports various functions such as deadband length, brake, shield, polarity flip, counting and symmetry, etc. The PWMPLUS can output flexible waveforms with different duty cycles to control external devices. The PWMPLUS clock needs to be enabled before using the PWMPLUS module.

Its system framework diagram is shown below:

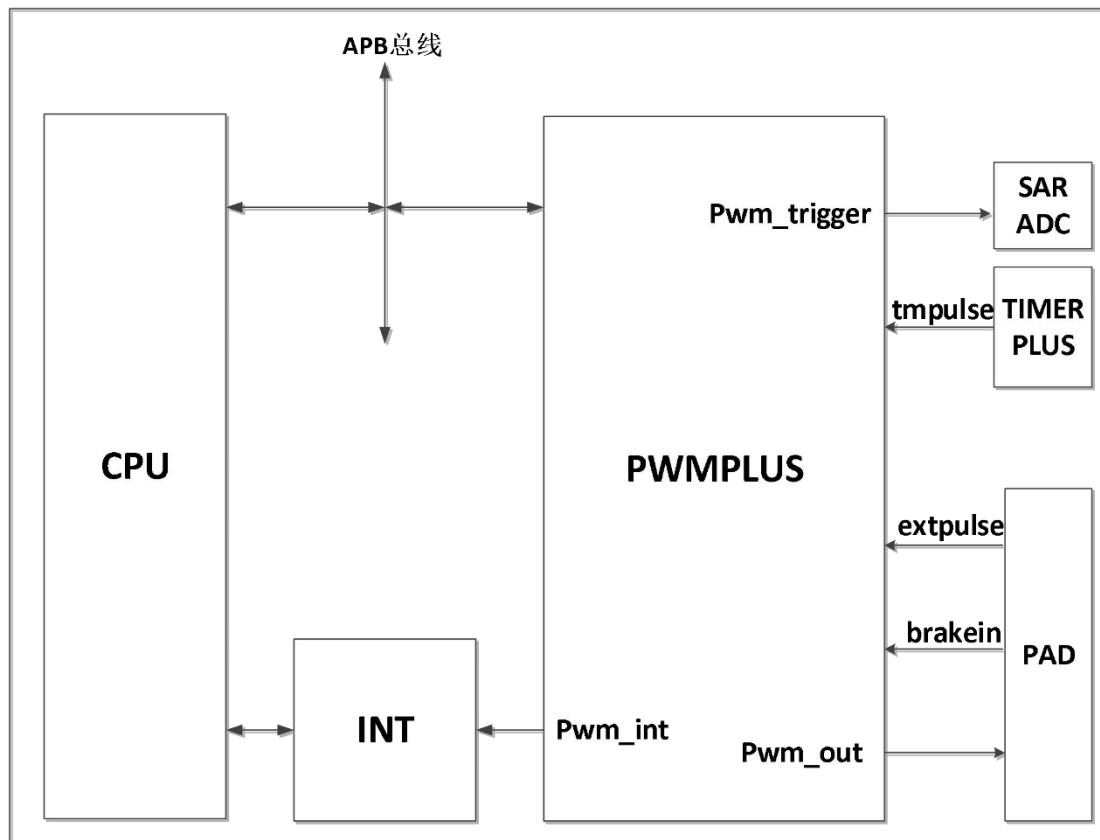


Figure 5-74 PWMPLUS System Block Diagram

5.14.2 characterization

- Supports 3 independent channels of 16bit PWM channel outputs (CH0/CH1/CH2) which can

DP32G030
Reference Manual

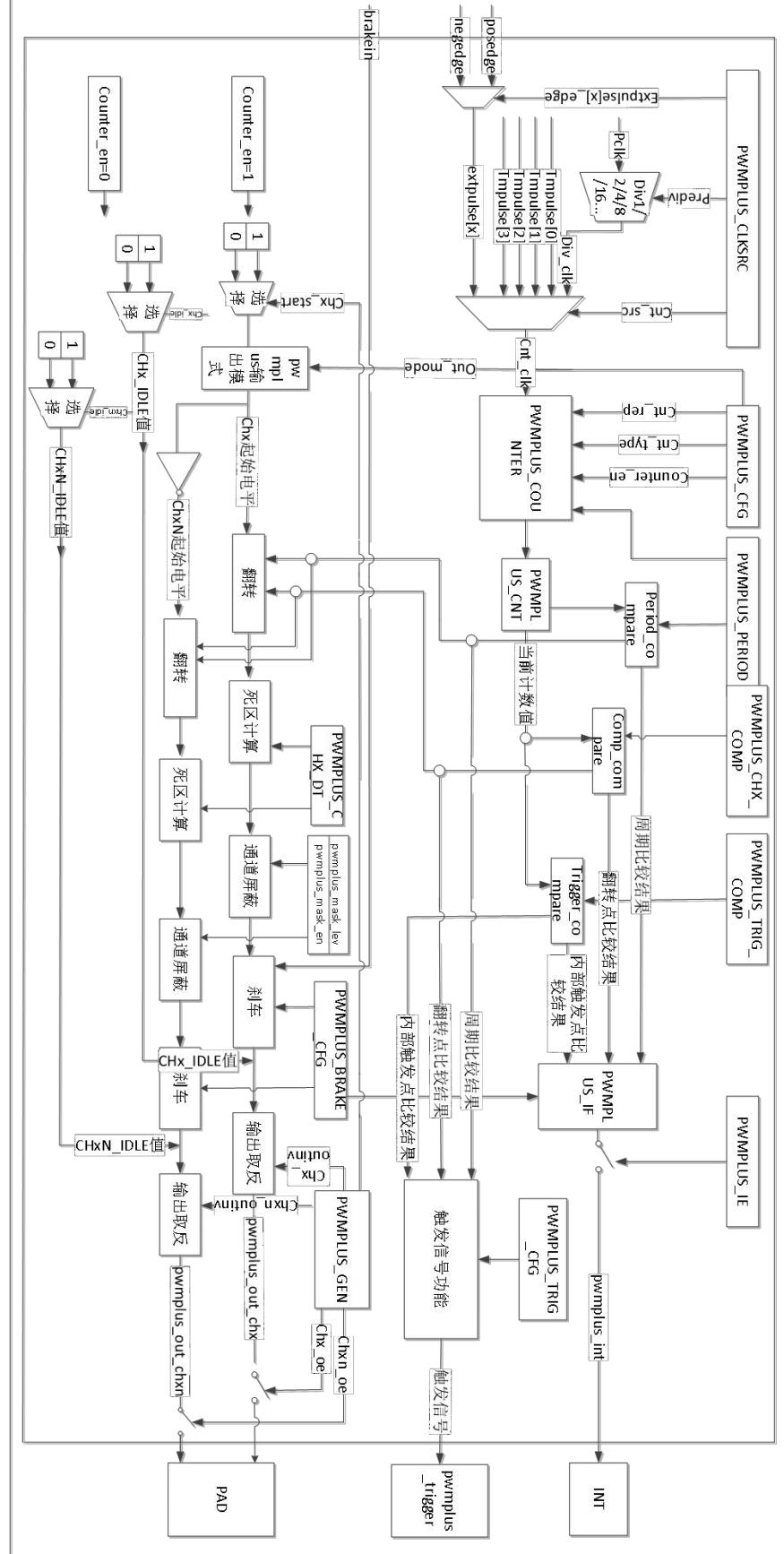
output PWM with different duty cycles.

waveform

- 16bit pre-scaler, supports on-chip timer or external signal as counting clock function

- Configurable deadband length per channel support
- Each channel PWM supports its own complementary output (CH0/CH0N, CH1/CH1N, CH2/CH2N)
- PWM counter supports counting up or counting down, edge-aligned mode or center-aligned mode configurable
- All channels support configuration of idle level, count start level, and whether outputs are flipped or not
- Support brake function, brake effective level configurable
- Support software forced output fixed level function, level polarity can be configured
- Supports internal specific trigger mechanism, which can generate three kinds of pulse signals: end-of-cycle, channel flip-flop, and specific trigger point.
- Supports single or cyclic mode selection
- Configurable to generate edge-aligned or center-aligned waveforms
- Support cycle value, channel flip point value, specific trigger point value fixed cycle automatic loading or software loading function

5.14.3 Block Diagram of Module Structure



As shown in the above figure, the structure of the PWMPLUS module is shown in the frame diagram, and its input signals mainly include clock `pclk`, external brake input signal `brakein`, external input pulse signal `extPLUS`, and internal timer input pulse signal `tmPLUS`, of which the three signals of `pclk`, `extPLUS`, and `tmPLUS` can be used as the counting clock of PWMPLUS. The three signals `pclk`, `extPLUS` and `tmPLUS` can be selected as PWMPLUS counting clock. The output signals are `pwmplus_int`, `pwmplus_trigger`, and `pwmplus_out`, which is the output of the channel waveform.

There are three kinds of counting clocks in this module, namely internal system clock `pclk`, external input pulse signal `extPLUS` and internal timer input pulse signal `tmPLUS`. The three clocks can be configured through the register `pwmplus_clksrc`, in which the internal clock `pclk` can be prescored with a frequency range of 1-256, while the external input pulse signal `extPLUS` can be selected as either rising edge valid or falling edge valid. The internal clock `pclk` can be prescaled in the range 1-256, while the external input pulse signal `extPLUS` can be selected as either rising or falling edge valid. In addition, the output mode of the `pwmplus`, the counting behavior of the counter, the counting cycle and the counting enable can be configured via the `pwmplus_cfg` register.

The output of the channel waveform is controlled by the channel waveform enable register. When `chx_oe` is 1, the channel waveform is output to the port of the PAD, and when `chx_oe` is 0, the high resistance state is output to the port of the PAD. The output of channel waveform is divided into idle state output and channel square wave output. When `counter_en` is 0, `pwm` outputs idle state `idle` value, which can be configured as 1 or 0 through `pwmplus_gen`, and can be further controlled to flip or not when outputting. When the count enable `counter_en` is 1, the `pwm` outputs channel square wave, and the output mode of PWMPLUS can be configured through the register `pwmplus_cfg`, and the starting voltage of the channel square wave can be configured as 1 or 0. Then, the waveform flip can be controlled through the comparison of the software configured count period value, flip point

value and the current count value, and the auto-load register `pwmplus_reload` to indicate the cycle overflow. You can also configure the Auto-Load register to indicate how many times the cycle overflow will automatically load the cycle value, the comparison value, the deadband value and the `TRIGGER` value. After that, you can configure the deadband length, channel mask, brake and flip-flop to generate the desired waveform.

The generation of interrupt signals is controlled by the interrupt enable and the interrupt state. When the interrupt enable `pwmplus_ie` is 1, the corresponding interrupt signal will be generated after the corresponding interrupt state is generated; when the interrupt enable `pwmplus_ie` is 0, the interrupt signal will not be generated even if there is an interrupt state. The trigger signal is controlled by the internal trigger configuration register `pwmplus_trig_cfg`, which can be configured to select the function of the trigger signal. The trigger signal is generated only when the corresponding trigger function is configured and there is a corresponding trigger state, and the trigger signal is a high level for one clock cycle.

This module also supports cycle value, channel flip-flop value, specific trigger point value fixed cycle auto-load or software load function, where the auto-load function indicates how many times the cycle overflow is automatically loaded once after the cycle value, the comparison value, the deadband value and the `TRIGGER` value, and the number of times the cycle overflow is configured through the auto-reload register `auto_reload` (`auto_reload+1` times). The software load function can be realized through the `pwmplus` configuration register software load control bit. In this module, `pwm_period`,

The 8 registers `pwm_ch0_comppwm_ch1_comppwm_ch2_comppwm_ch0_dt`, `pwm_ch1_dt`, `pwm_ch2_dt`, and `trig_comp` have their own shadow registers. After the software writes a 1 to this bit, the hardware loads the latest values stored in these 8 registers into their respective shadow registers at the end of the current cycle, which will take effect in the following cycle. The module can also read the operating status of the counter via the counter operating status register and the current status of the brake input signal via the brake input signal status register.

5.14.4 Functional Description

Edge aligned mode output

The PWM output mode of the counter of this module can be configured as two modes: edge-aligned mode and center-aligned mode. Firstly, in the edge-aligned output mode, the counter counts upward or downward, and the timings for single output, cyclic output, and different starting levels are shown below.

- Counter Up

Upward counting means that the counter starts counting from zero and adds one to each clock cycle counted, until it reaches the configured cycle value. The meanings of `chx`, `chxn`, `chx_inv`, `chxn_inv` and the gray part in the following diagrams are as follows: `chx` indicates the `pwmplus` output waveform of the `chx` channel when `chx_outinv` of the output status register is 0, `chx_inv` indicates the `pwmplus` output waveform of the `chx` channel when `chx_outinv` of the output status register is 1, and the output waveform of the channel at this time is opposite to the original waveform of the channel, `chxn` and `chxn` are the same as the original waveform of the channel, `chxn` and `chxn` are the same as the original waveform of the channel. output waveform, at this time the output waveform of the channel is opposite to the original waveform of the channel, and `chxn` and `chxn_inv` indicate its complementary output. The gray part of the figure represents the dead zone, and the dead zone length is assignable.

DP32G030
Reference Manual

1、 Single output, start=0 case

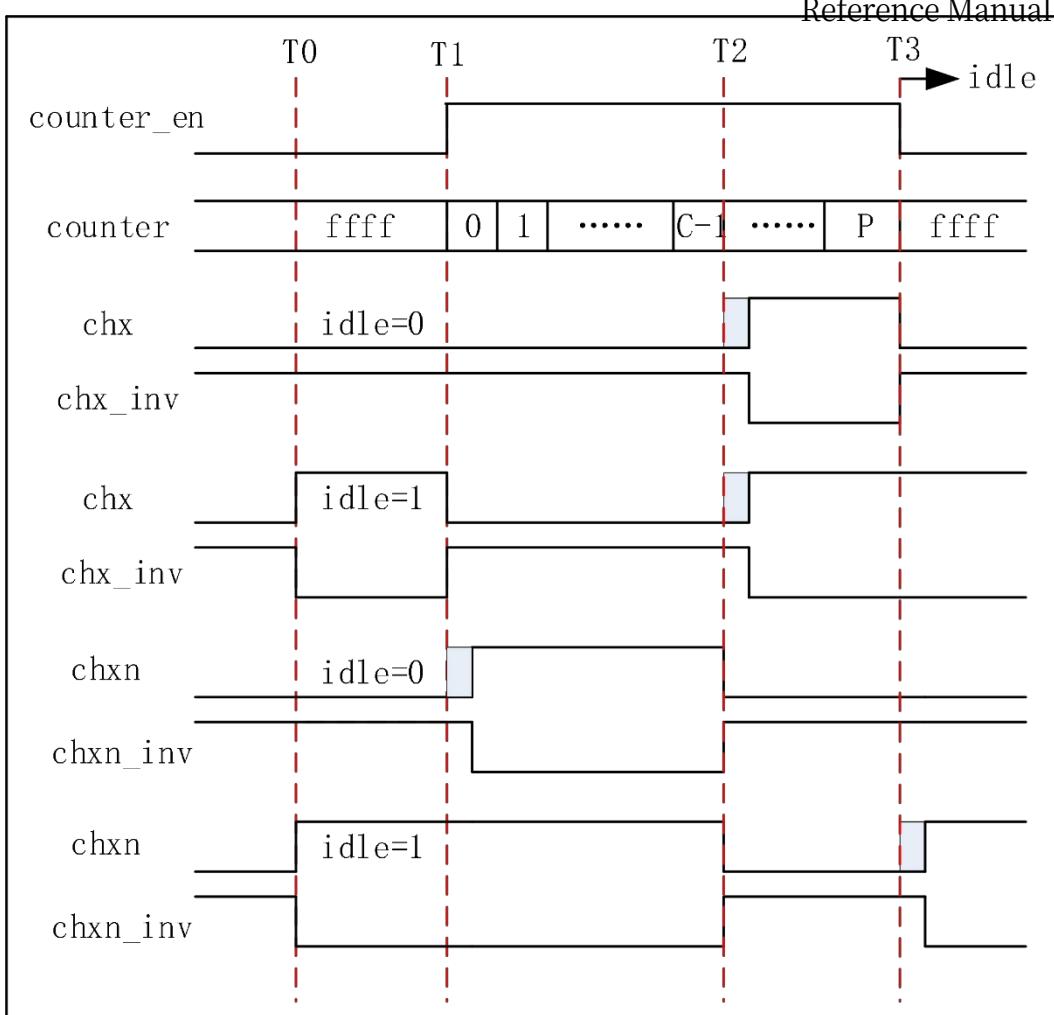


Figure 5-76 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Single Output, Start Level 0

As shown in the figure above, before turning on the counting enable `counter_en` (before T_1 time) `pwmplus` outputs IDLE value, when T_0 time changes IDLE value `pwmplus` outputs change with IDLE value, when T_1 time turns on the counter counting enable, `pwmplus` outputs channel waveform. When the corresponding interrupt enable is configured, a flip-flop interrupt is generated at T_2 and a cycle overflow interrupt is generated at T_3 .

2、Single output, `start=1` case

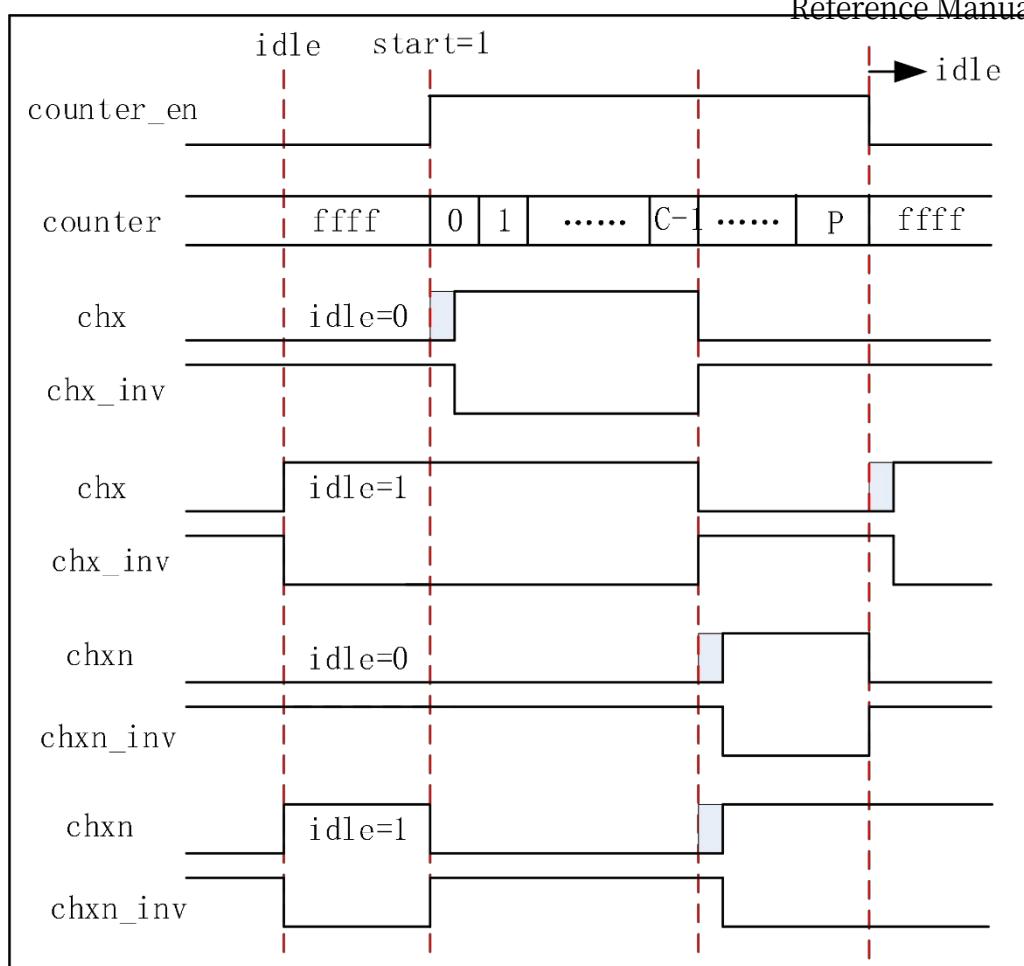


Figure 5-77 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Single Output, Start Level 1

As shown above, configuring the flip-point interrupt enable and end-of-cycle interrupt enable generates a flip-point interrupt at the third vertical line and an end-of-cycle interrupt at the fourth vertical line.

3、Loop output, start=0 case

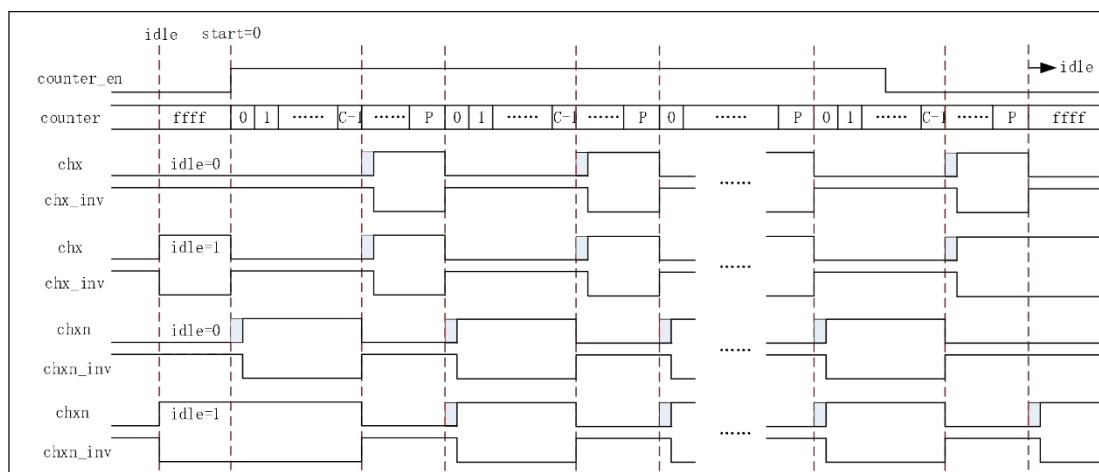


Figure 5-78 Timing Diagram of PWMPLUS Edge Alignment Mode, Count Up, Cyclic Output, Start Level 0

As shown in the above figure, configuring the flip-flop interrupt enable and the end-of-cycle interrupt enable, the flip-flop interrupt can be generated at each "C-1" position and the end-of-cycle interrupt can be generated at each "P" position.

4. Loop output, start=1 case

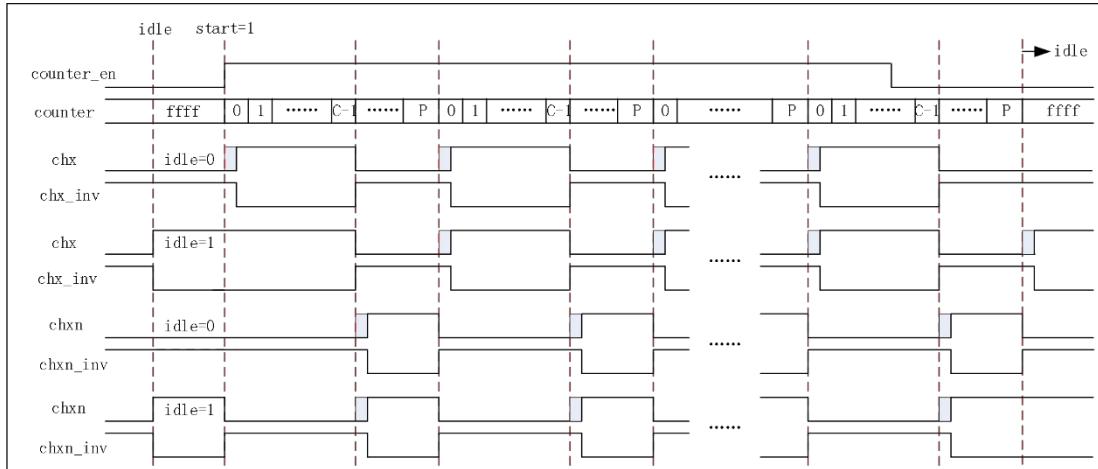


Figure 5-79 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Up, Cyclic Output, Start Level 1

As shown in the above figure, configuring the flip-flop interrupt enable and the end-of-cycle interrupt enable, the flip-flop interrupt can be generated at each "C-1" position and the end-of-cycle interrupt can be generated at each "P" position.

- Counter Down

Counting down means that the counter starts counting from the configured cycle value and decrements by one for each counting clock cycle, all the way down to zero. The meanings of **chx**, **chxn**, **chx_inv**, **chxn_inv** and the gray part in the following figure are as follows: **chx** indicates the **pwm** output waveform of the **chx** channel when **chx_outinv** of the output status register is 0, **chx_inv** indicates the **pwmplus** output waveform of the **chx** channel when **chx_outinv** of the output status register is 1, at this time, the output waveform of the channel is opposite to the original waveform of the channel, **chxn** and **chxn_inv** indicate the output waveform of the channel. When **chx_outinv** is 1, **chx** channel **pwmplus** outputs waveform, the output waveform of the channel at this time is opposite to the original waveform.

of the channel, and `chxn` and `chxn_inv` indicate its complement. The gray part of the figure represents the dead zone, and the length of the dead zone can be configured.

1、 Single output, `start=0` case

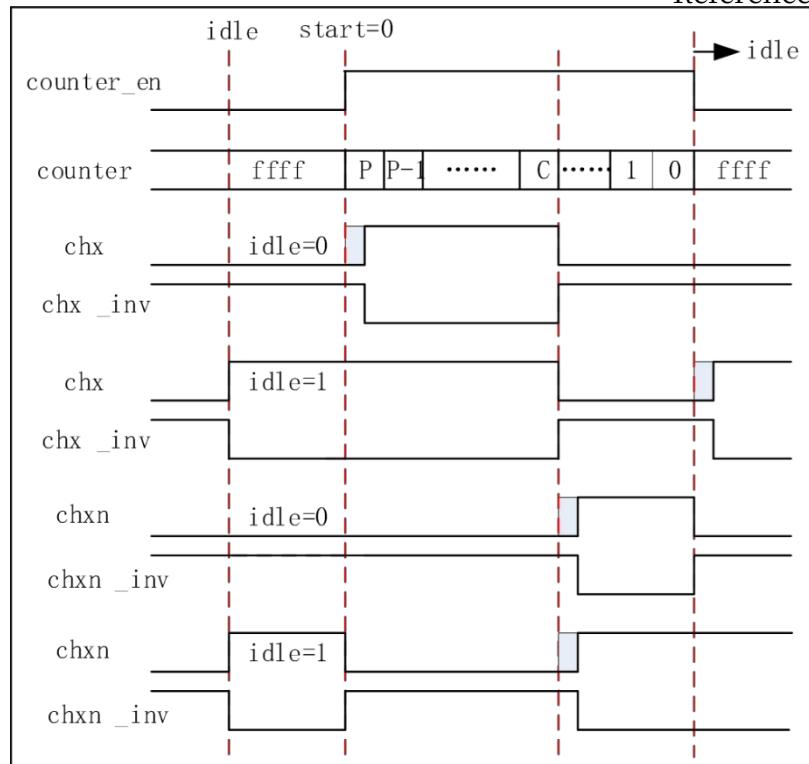


Figure 5-80 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Single Output, Start Level 0

2、Single output, start=1 case

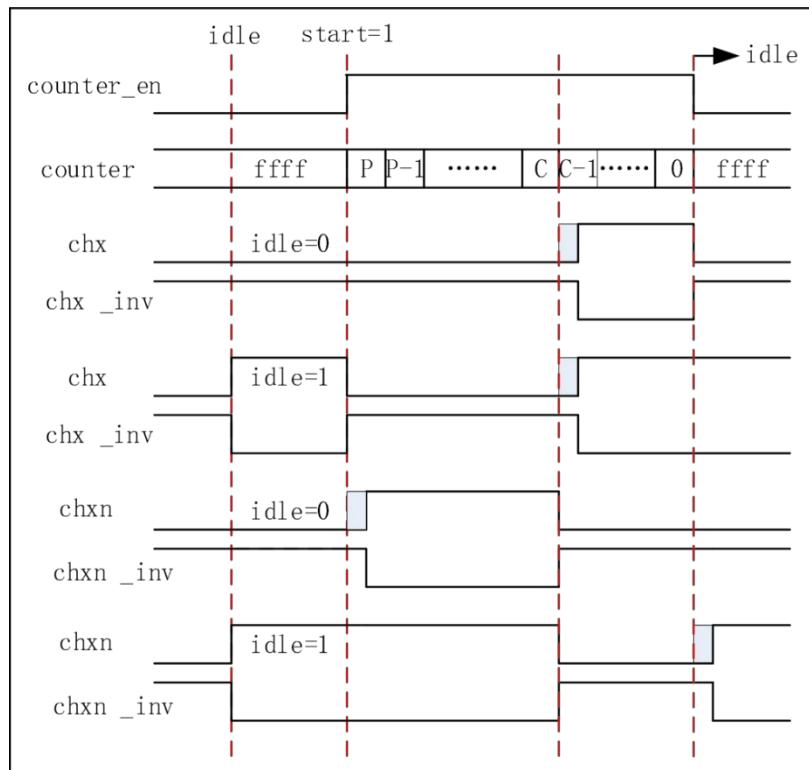


Figure 5-81 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Single Output, Start Level 1

3、Loop output, start=0 case

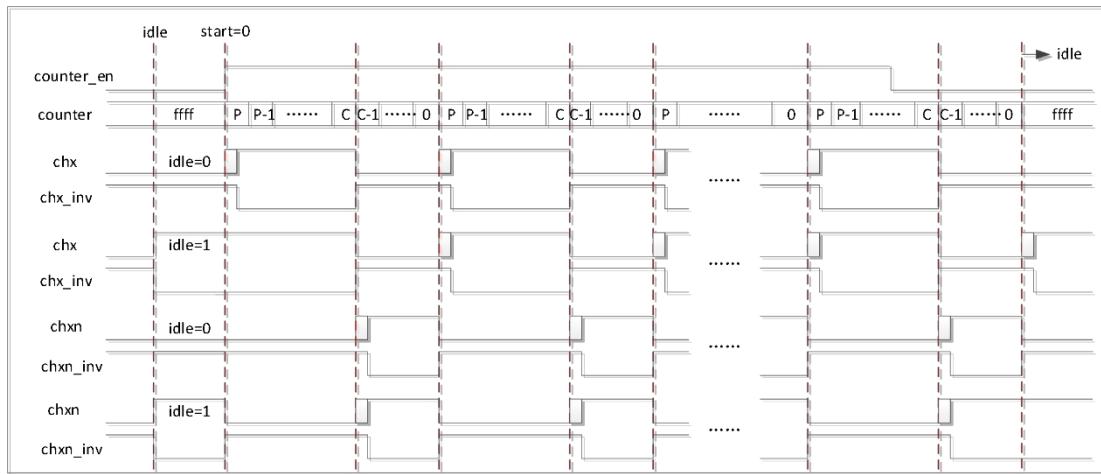


Figure 5-82 Timing Diagram of PWMPLUS Edge Alignment Mode, Count Down, Cyclic Output, Start Level 0

4、Loop output, start=1 case

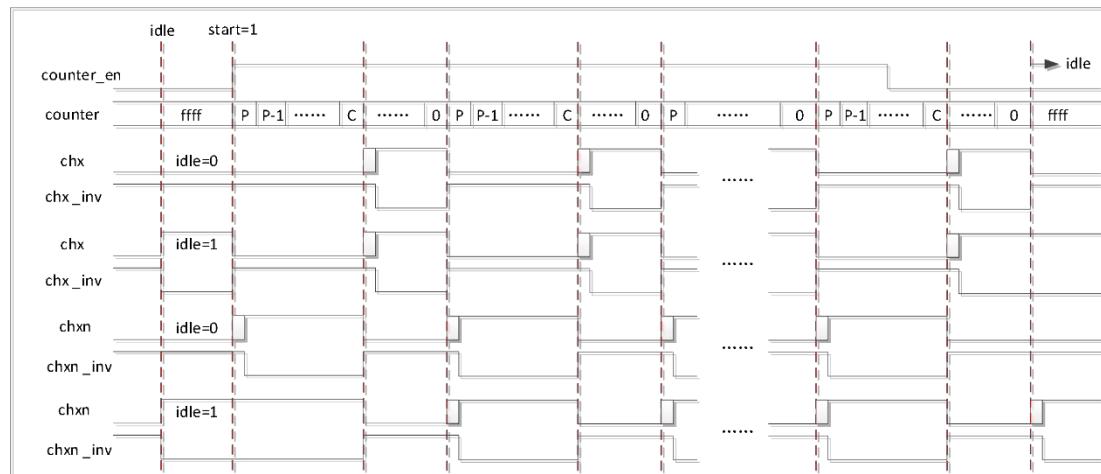


Figure 5-83 Timing Diagram for PWMPLUS Edge Alignment Mode, Count Down, Cyclic Output, Start Level 1

Center-aligned mode output

The timing schematic for the cases of counter counting up or down, single output, cyclic output and different starting levels in the center aligned mode is shown below:

- Counter Up

Upward counting means that the counter starts counting from zero and adds one to each counting clock cycle until it reaches the configured cycle value. The

diagrams are as follows: chx indicates the output state registers

When `chx_outinv` is 0, the `pwmplus` output waveform of `chx` channel, `chx_inv` indicates the output status register `chx_outinv` is 1, the `pwmplus` output waveform of `chx` channel, at this time, the output waveform of the channel is opposite to the original waveform of the channel, and `chxn` and `chxn_inv` indicate its complementary output. The gray part of the graph represents the dead zone, and the length of the dead zone can be configured.

1、Single output, start=0 case

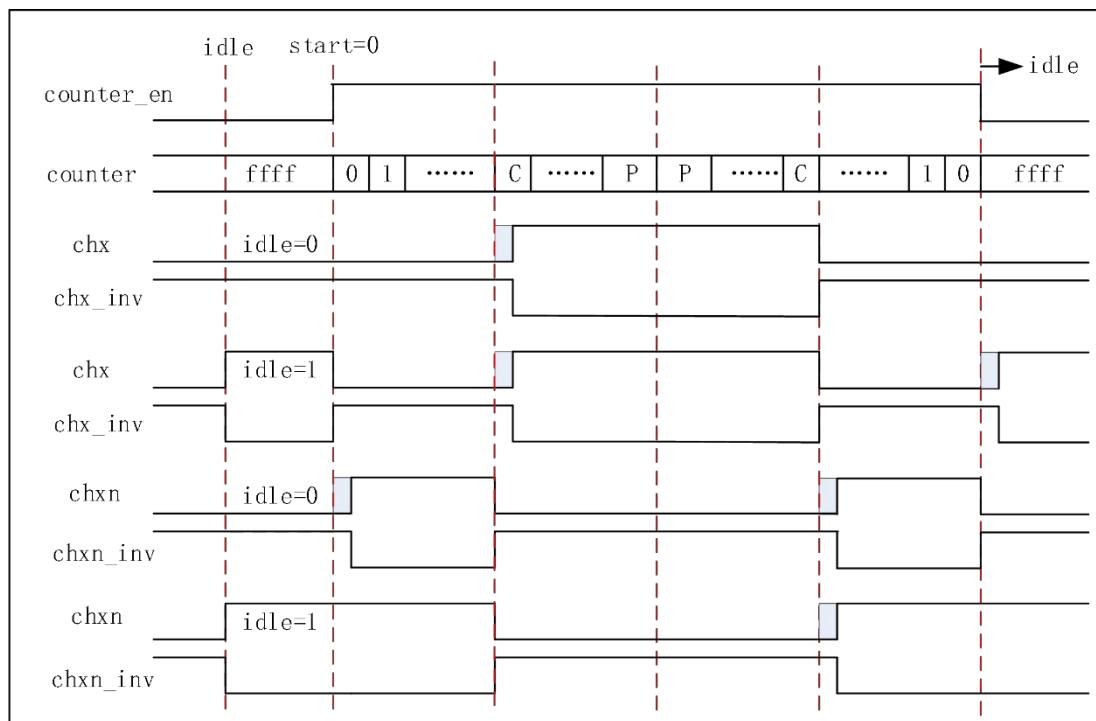


Figure 5-84 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Single Output, Start Level 0

As can be seen in the figure above, the center-aligned mode `pwmplus` output waveforms are center-aligned at the position where the p-dot ends, with the first half of the cycle counting up and the second half counting down. It can be configured to generate flip-flop interrupts at the third and fifth vertical lines and end-of-cycle interrupts at the fourth and sixth vertical lines.

2、Single output, start=1 case

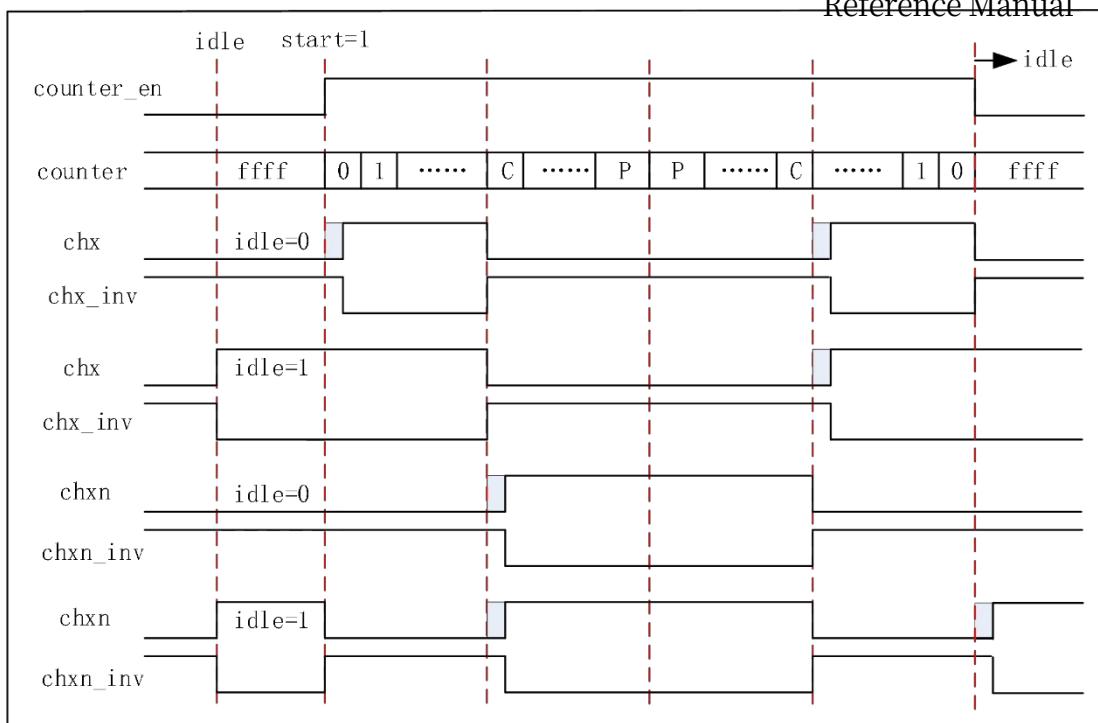


Figure 5-85 Timing Diagram for PWMPLUS Center Aligned Mode, Count Up, Single Output, Start Level 1

As shown above, configuring the flip-point interrupt enable and end-of-cycle interrupt enable generates flip-point interrupts at the third and fifth vertical lines and end-of-cycle interrupts at the fourth and sixth vertical lines.

3、Loop output, start=0 case

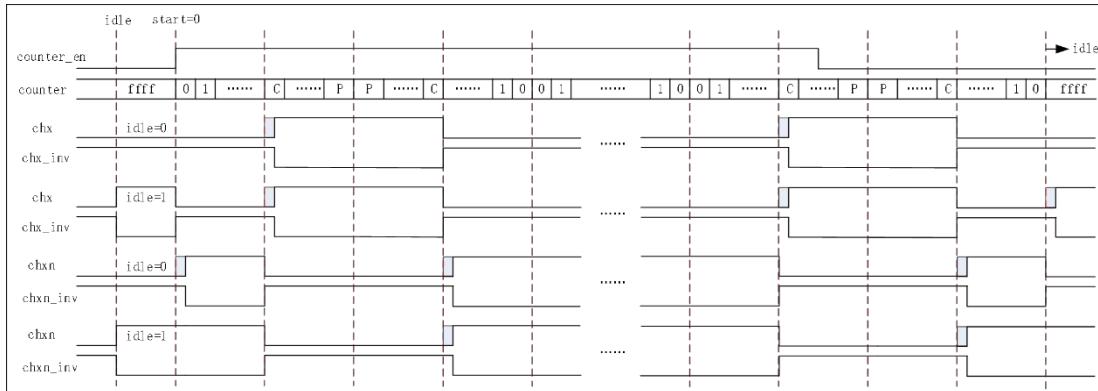


Figure 5-86 Timing Diagram of PWMPLUS Center Aligned Mode, Count Up, Cyclic Output, Start Level 0

As shown in the above figure, configuring the flip-flop interrupt enable and the end-of-cycle interrupt enable can generate a flip-flop interrupt at every "C" position, and an end-of-cycle interrupt at every two "P" positions or every two "0" positions. The end-of-cycle interrupt can be generated at every "C" position, at every two "P" positions or at every two "0" positions.

4. Loop output, start=1 case

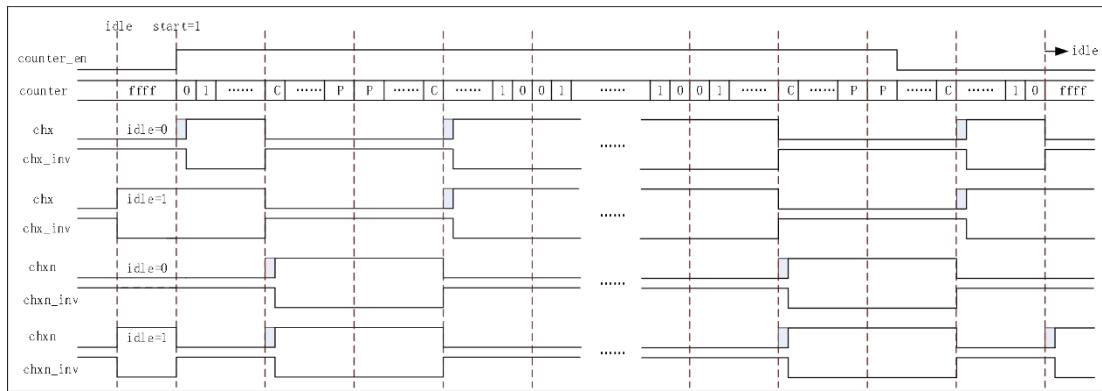


Figure 5-87 Timing Diagram of PWMPLUS Center Aligned Mode, Count Up, Cyclic Output, Start Level 0

As shown in the above figure, configuring the flip-flop interrupt enable and end-of-cycle interrupt enable can generate a flip-flop interrupt at every "C" position, and an end-of-cycle interrupt at every two "P" positions or every two "0" positions. The end-of-cycle interrupt can be generated at every "C" position, at every two "P" positions or at every two "0" positions.

- Counter Down

Counting down means that the counter starts counting from the configured cycle value and decrements by one for each counting clock cycle, all the way down to zero. The meanings of `chx`, `chxn`, `chx_inv`, `chxn_inv` and the gray part in the following diagrams are as follows: `chx` indicates the `pwmplus` output waveform of the `chx` channel when the output status register `chx_outinv` is 0, `chx_inv` indicates the `pwmplus` output waveform of the `chx` channel when the output status register `chx_outinv` is 1, and the output waveform of the `chx` channel is the opposite of the original waveform of the channel, `chxn` and `chxn_inv` are the opposite of the original waveform of the channel. The output waveform of the channel at this time is opposite to the original waveform of the channel, and `chxn` and `chxn_inv` indicate its complementary output. The gray part of the graph represents the dead zone, and the length of the dead zone can be configured.

1. Single output, start=0 case

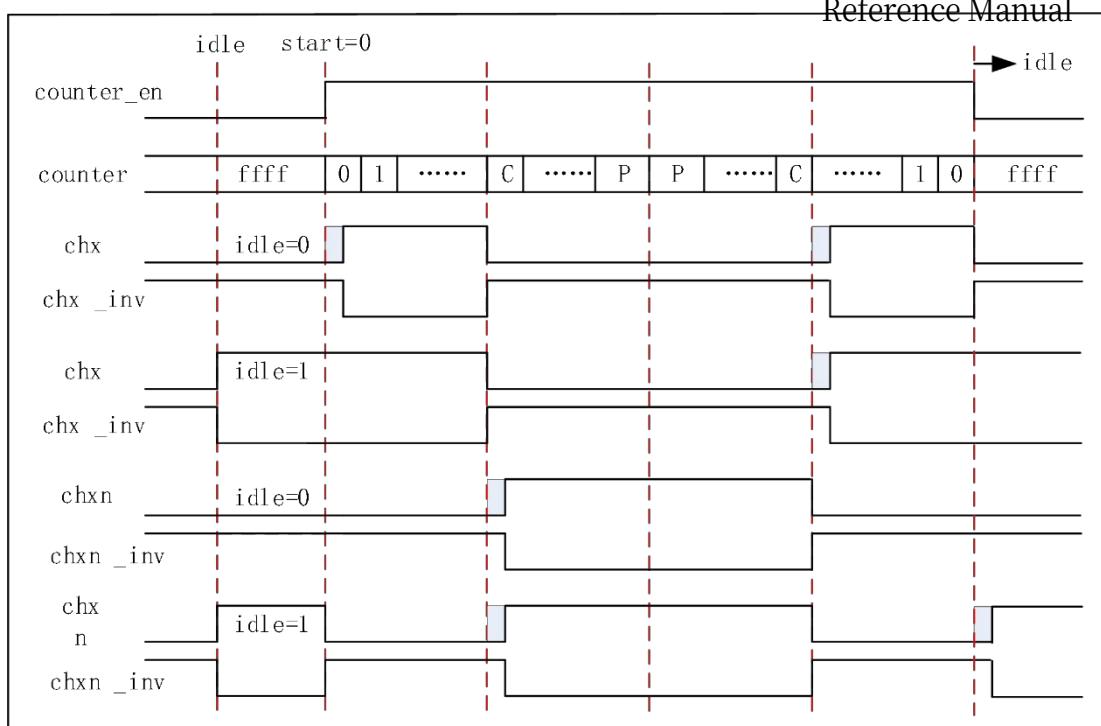


Figure 5-88 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Single Output, Start Level 0

2、Single output, start=1 case

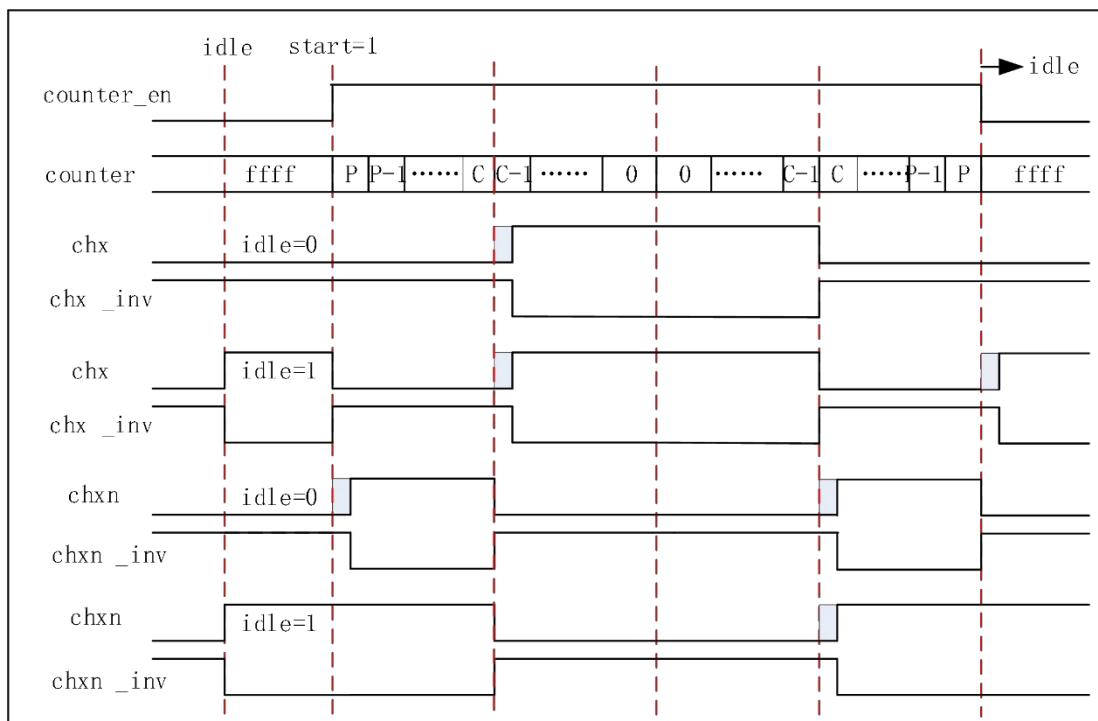


Figure 5-89 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Single Output, Start Level 1

3、Loop output, start=0 case

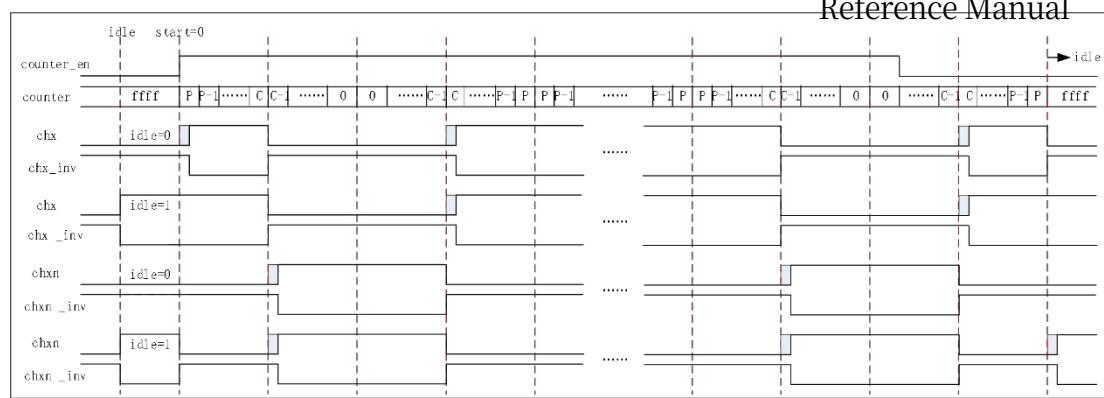


Figure 5-90 Timing Diagram of PWMPLUS Center Aligned Mode, Count Down, Cyclic Output, Start Level 0

4. Loop output, start=1 case

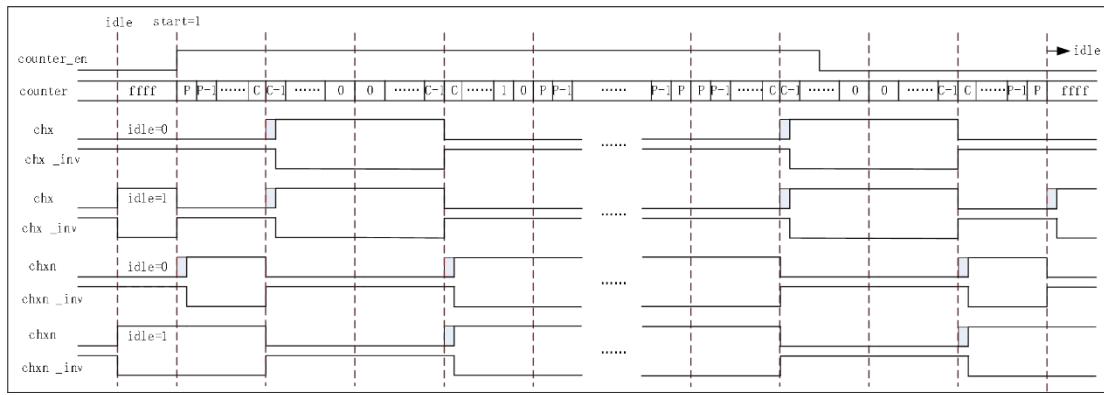


Figure 5-91 Timing Diagram for PWMPLUS Center Aligned Mode, Count Down, Cyclic Output, Start Level 1

Complementary output with deadband insertion

PWMPLUS is capable of outputting two complementary signals and managing the instantaneous turn-off and turn-on of the outputs, which is often referred to as the dead time, and which should be adjusted by the user according to the connected outputs and their characteristics (level-shifting delays, power-switching delays, etc.).

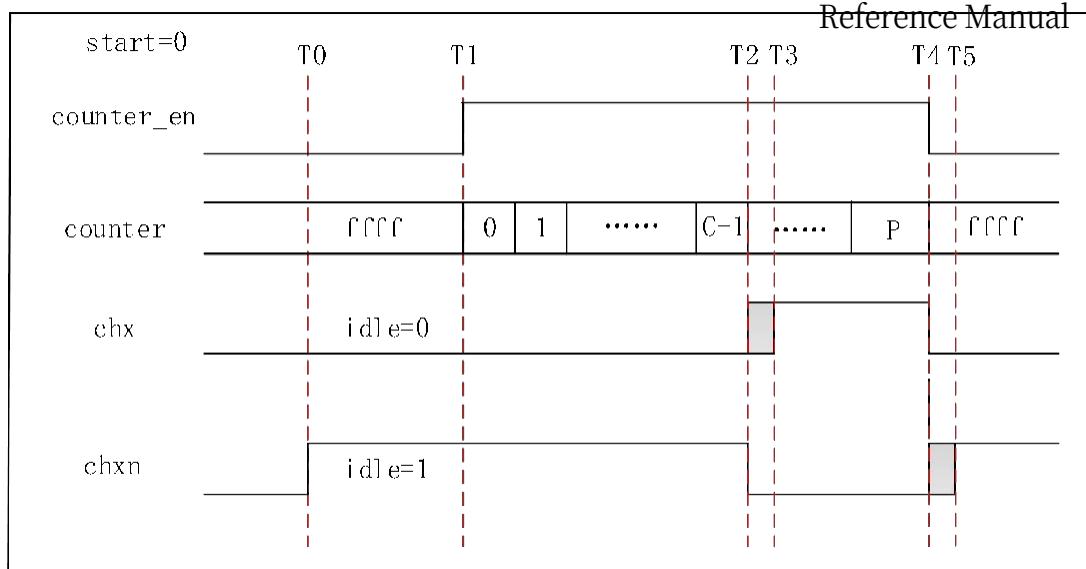


Figure 5-92 PWMPLUS Complementary Output Timing Diagram with Deadband Insertion

The above figure shows the complementary output of PWMPLUS with deadband insertion, where chxn is the complementary output of chx. The gray part between T2-T3 and T4-T5 in the figure is the deadband, and the length of deadband can be configured through the CHx_DT register, and the configuration of the deadband length needs to match with the value of the period, the value of CHx_START, and the value of CHx_COMP, or the output waveform may not reach the Otherwise, the output waveform may not achieve the expected effect.

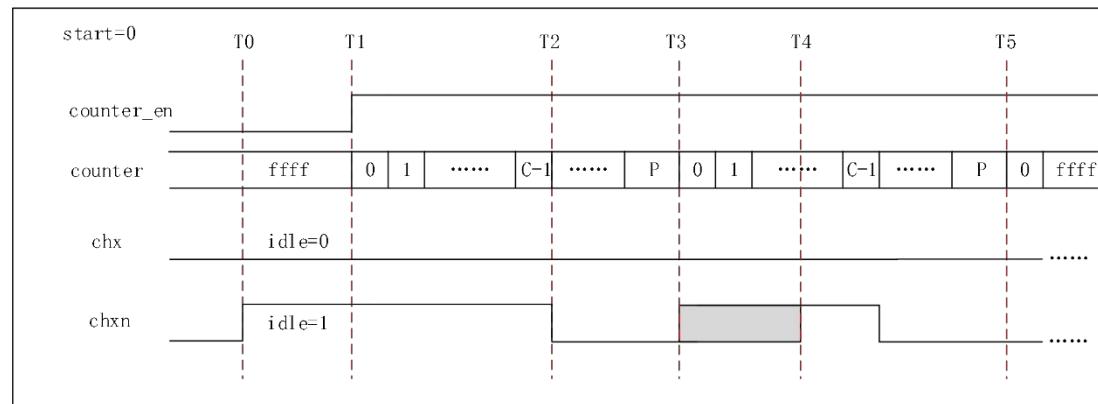


Figure 5-93 PWMPLUS Deadband Length Greater Than Positive Pulse Less Than Negative Pulse Timing Chart

The above figure shows the timing diagram when the PWMPLUS start level is 0, when the configured deadband length is greater than the positive pulse and less than the negative pulse, and the gray part of T3-T4 in the figure is the deadband length.

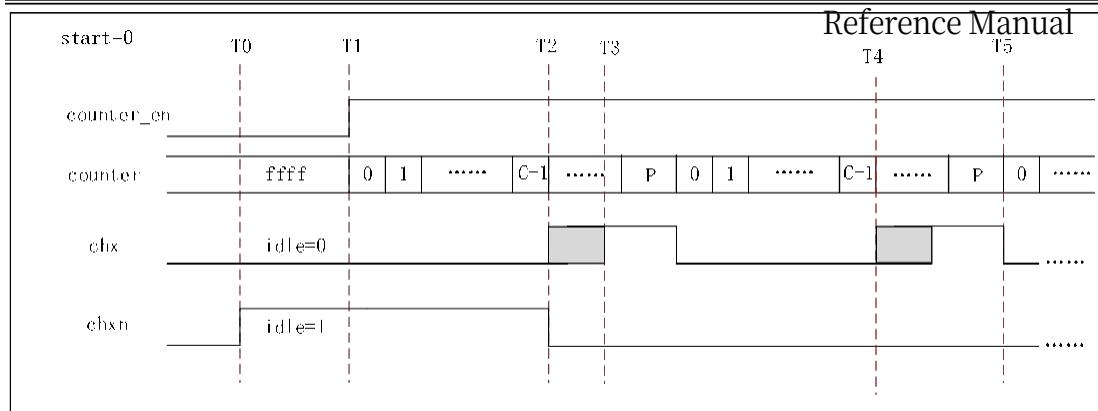


Figure 5-94 PWMPLUS Deadband Length Greater Than Negative Pulse Less Than Positive Pulse Timing Chart

The above figure shows the timing diagram when the PWMPLUS start level is 0, when the configured deadband length is greater than the positive pulse and less than the negative pulse, and the gray part of T2-T3 in the figure is the deadband length.

Output under braking

This module supports the selection of different brake signals for different channels, configurable effective brake level, and configurable level value during braking for each channel. During braking, the counter counts normally and is not affected by braking.

When the brake signal arrives, it is effective on the channel output signal immediately, and after the brake is withdrawn, the normal waveform will not be output until the beginning of the next cycle.

Take the case when the counter is counting up and the start level is 0 as an example:

1. Edge alignment mode

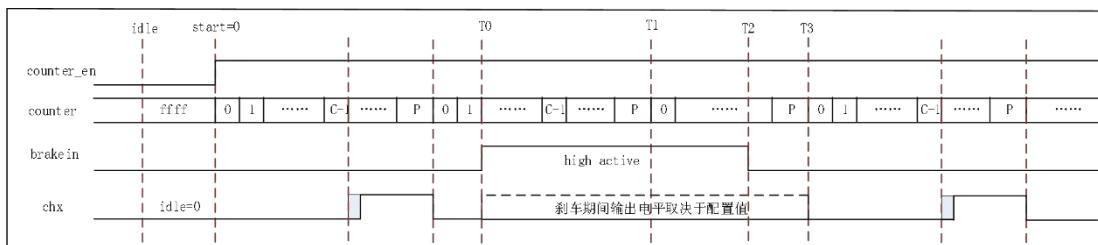


Figure 5-95 PWMPLUS Edge Alignment Mode Brake Case Timing Chart

As shown in the above figure, in the edge alignment mode, take the high effective brake level as an example, the input brake level becomes high in T0

moment, the signal is effective immediately and enters into **Feedback Map**, during the braking period, the channels are not outputting normal waveforms, as shown in the figure, the output waveforms in the τ_1 moment will not be flipped and the channel waveforms outputs depend on the configured value; when the braking is finished in the τ_2 moment, the channel waveforms do not return to normal immediately, but When τ_2 moment is over, the channel waveforms do not return to normal immediately, but will output normal waveforms only at the beginning of the next cycle (τ_3 moment).

2. Center alignment

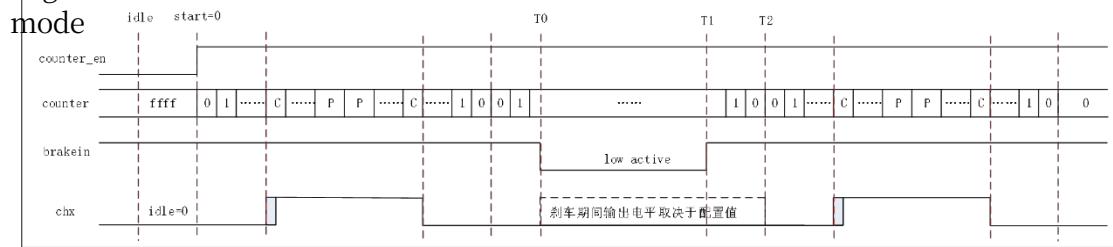


Figure 5-96 PWMPLUS Center Alignment Mode Brake Condition Timing Chart

As shown in the above figure, in the center-aligned mode, take the low effective brake level as an example, the input brake level becomes low in τ_0 moment, the signal is effective immediately and enters into the braking period, during the braking period, the channels are not outputting normal waveforms, and the channel waveform outputs are dependent on the configuration value; when the braking is finished in τ_1 moment, the channel waveforms don't return to normal immediately, but will only be outputting the normal waveforms at the beginning of the next cycle (τ_2 moment). Normal waveforms are output.

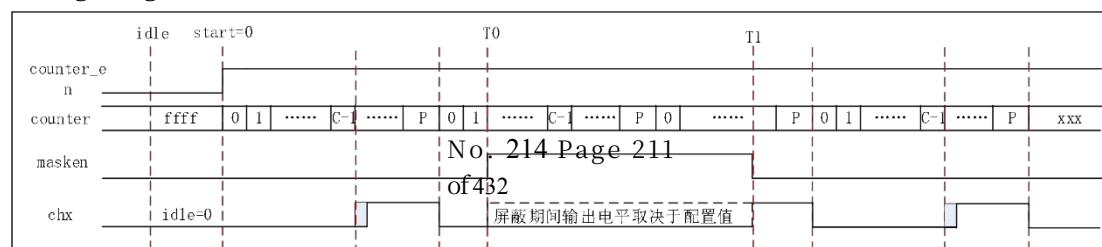
In addition, this chip also supports the brake filter function, which can control the digital filtering of the brake signal with 2, 4 and 8 internal pre-scaler clock filters, which are controlled by the register BRAKE_FILTER.

Output in case of MASK

This module supports the configuration of independent MASK function for each channel, and the level value during the MASK period can be configured for each channel, during the MASK period, the counter counts normally and is not affected.

When the MASK function is effective, it is immediately effective for the channel output signal, and when the MASK function is revoked, the normal waveform will be output immediately. Take the case when the counter is counting up and the start level is 0 as an example:

1. Edge alignment mode



DP32G030
Reference Manual

2. Center

alignment Figure 5-97 PWMPLUS Edge Alignment MASK Case Timing Diagram
mode

As shown in the above figure, in the edge alignment mode, the shielding enable becomes high at the τ_0 time, the shielding enable is effective immediately and enters into the shielding period, during the shielding period, the channels are not outputting normal waveforms, and the channel waveform output depends on the configured value; when the shielding ends at the τ_1 time, the channel waveforms immediately return to normal, and normal waveforms are output.

2. Center

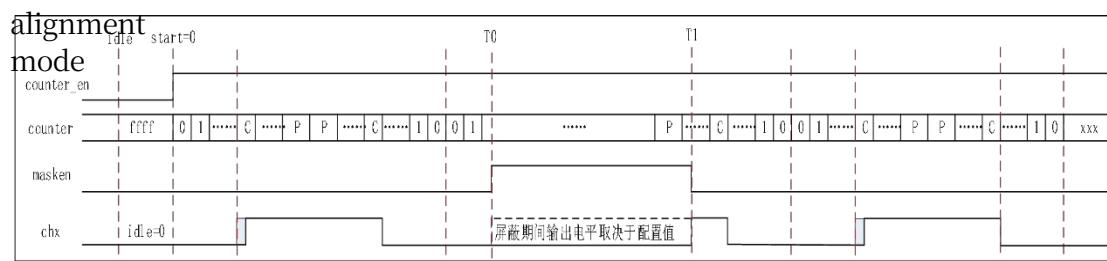


Figure 5-98 PWMPLUS Center Alignment MASK Case Timing Diagram

As shown in the above figure, in the center-aligned mode, the shielding enable becomes high at the T_0 time, the shielding enable is effective immediately and enters into the shielding period, during the shielding period, the channels are not outputting normal waveforms, and the channel waveform output depends on the configured value; when the shielding ends at the T_1 time, the channel waveforms immediately return to normal, and normal waveforms are output.

Periodic values and rollover points

Example of the relationship between the configuration cycle value and the flip point value (IDLE is 0)

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 0, duty cycle is 75%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x2.

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 0, duty cycle is 12.5%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x7.

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 0, duty cycle is 0%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x8 (greater than 7 is fine)

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 0, duty cycle is 100%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x0.

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 1, duty cycle is 25%, then it needs to be configured as: PERIOD=0x7,

2、Center

~~CHx_COMP=0x2.~~
mode

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 1, duty cycle is 87.5%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x7.

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 1, duty cycle is 100%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x8 (greater than 7 is fine)

The PWMPLUS waveform that the user wants to generate is: period 8, count start level is 1, duty cycle is 0%, then it needs to be configured as: PERIOD=0x7, CHx_COMP=0x0.

The cycle value is configured as

7 in the following schematic

example. 1. edge-aligned mode,

count-up case:

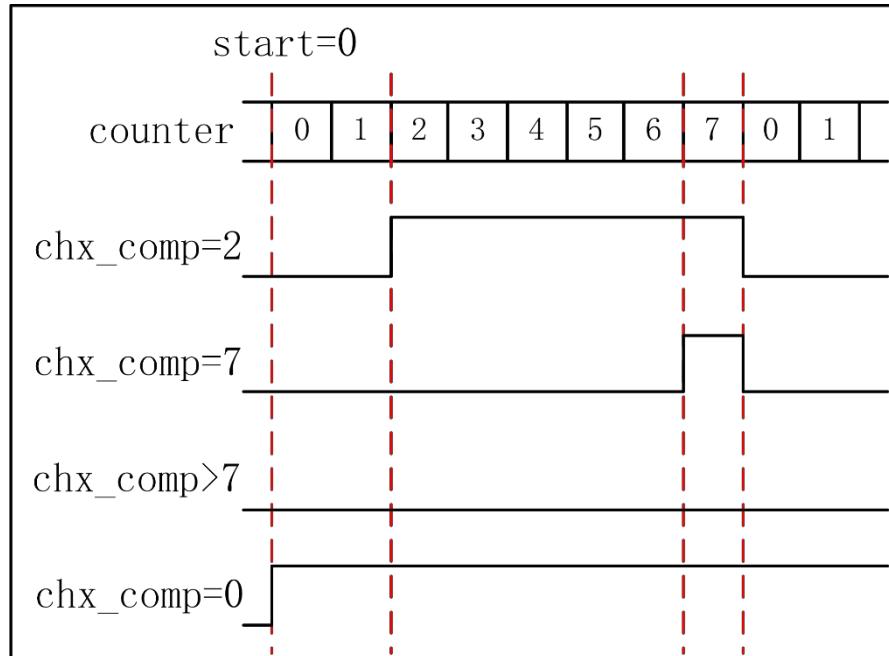


Figure 5-99 PWMPLUS Edge Alignment Mode, Count Up, Flip Point and Cycle Timing Diagram with Start Level 0

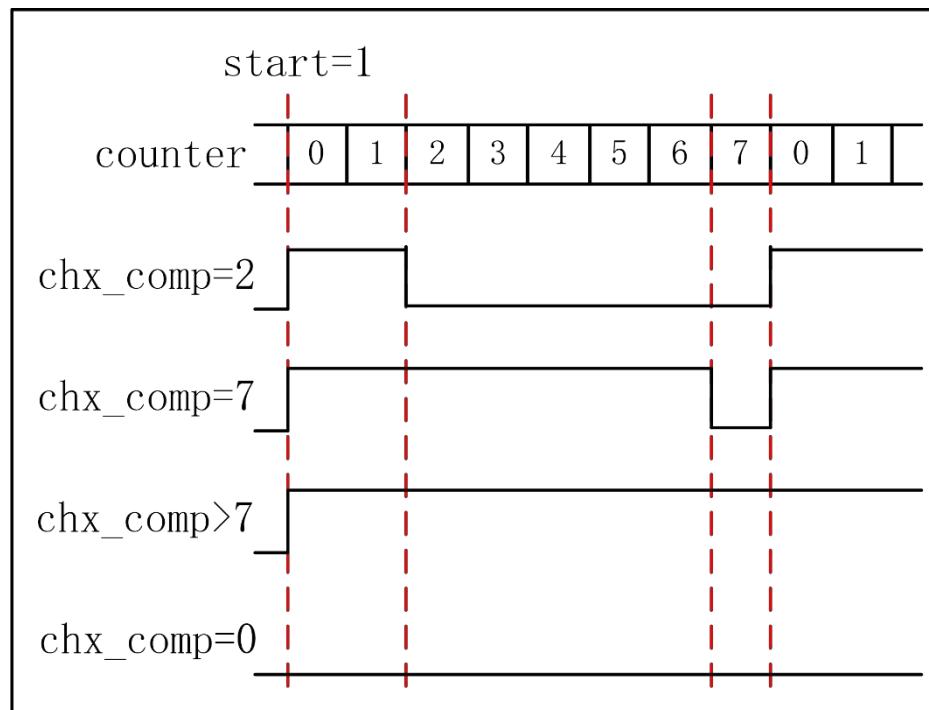


Figure 5-100 PWMPLUS Edge Alignment Mode, Count Up, Flip Point and Cycle Timing Diagram with Start Level 1

DP32G030
Reference Manual

2. Edge-aligned mode, downward counting case:

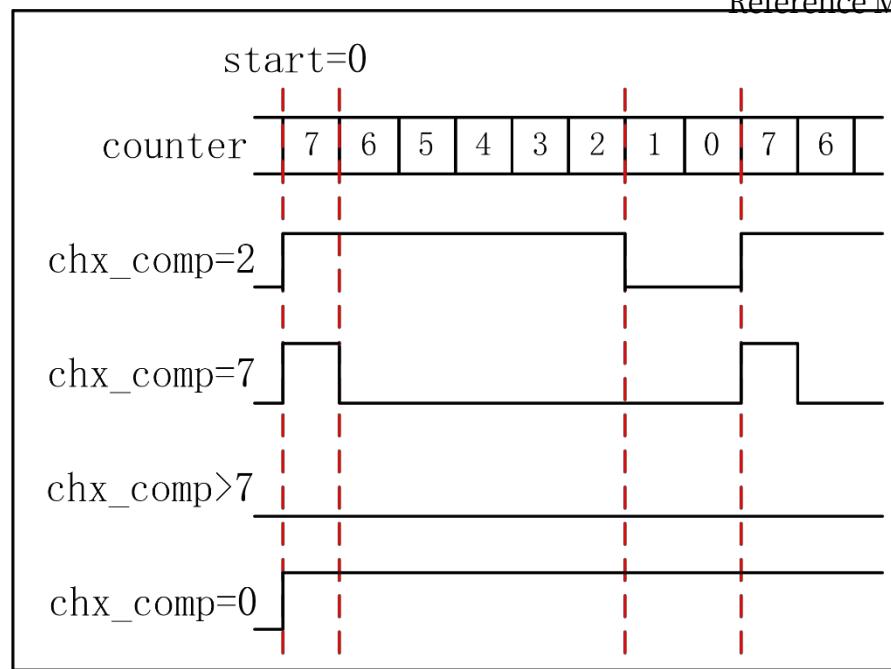


Figure 5-101 PWMPLUS Edge Aligned Mode, Count Down, Start Level 0 Flip Point and Cycle Timing Diagram

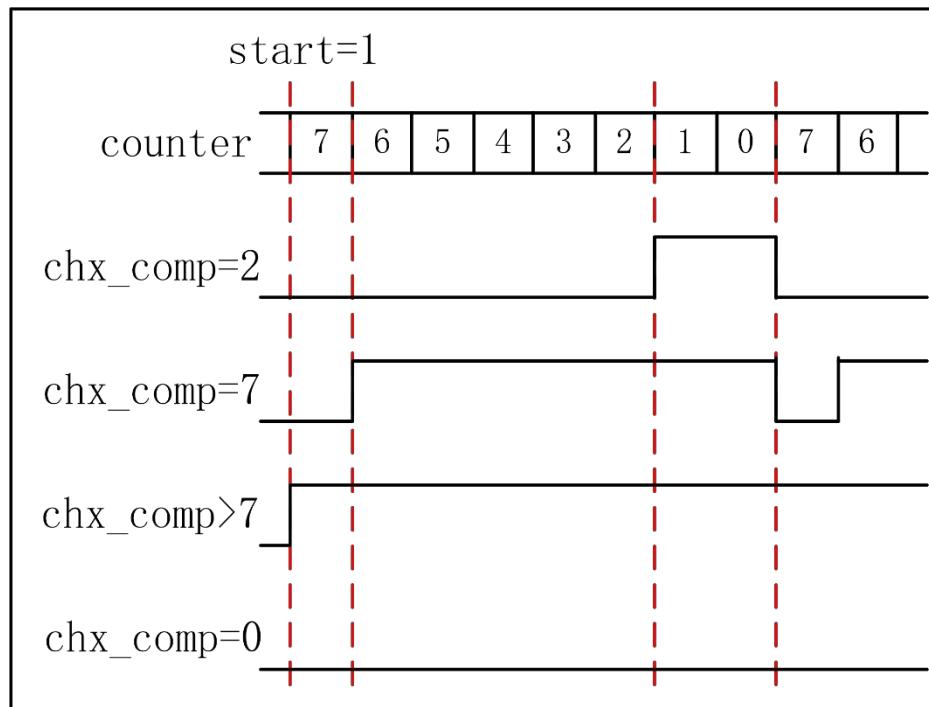


Figure 5-102 PWMPLUS Edge Aligned Mode, Count Down, Start Level 1 Flip Point and Cycle Timing Diagram

3. Center alignment mode, counting up situation:

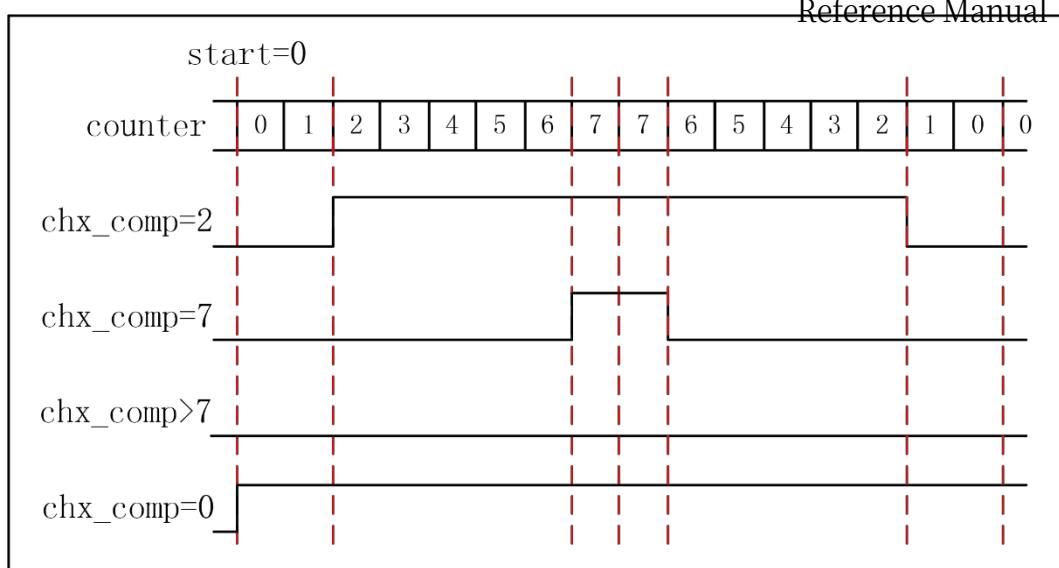


Figure 5-103 PWMPLUS Center Aligned Mode, Count Up, Flip Point and Cycle Timing Diagram with Start Level 0

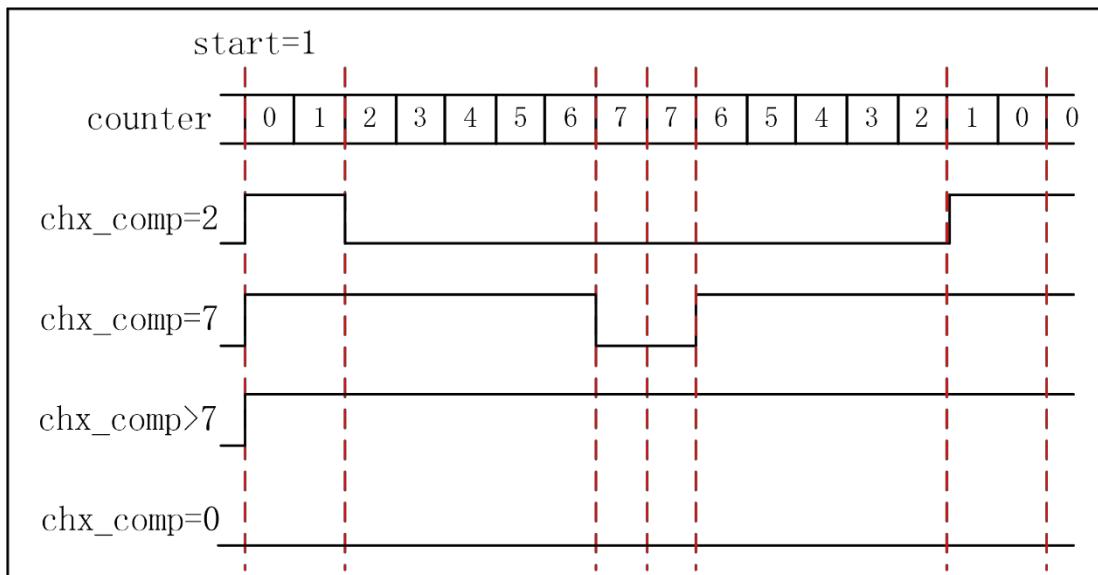


Figure 5-104 PWMPLUS Center Aligned Mode, Count Up, Flip Point and Cycle Timing Chart with Start Level 1

4. Center-aligned mode, counting down situation:

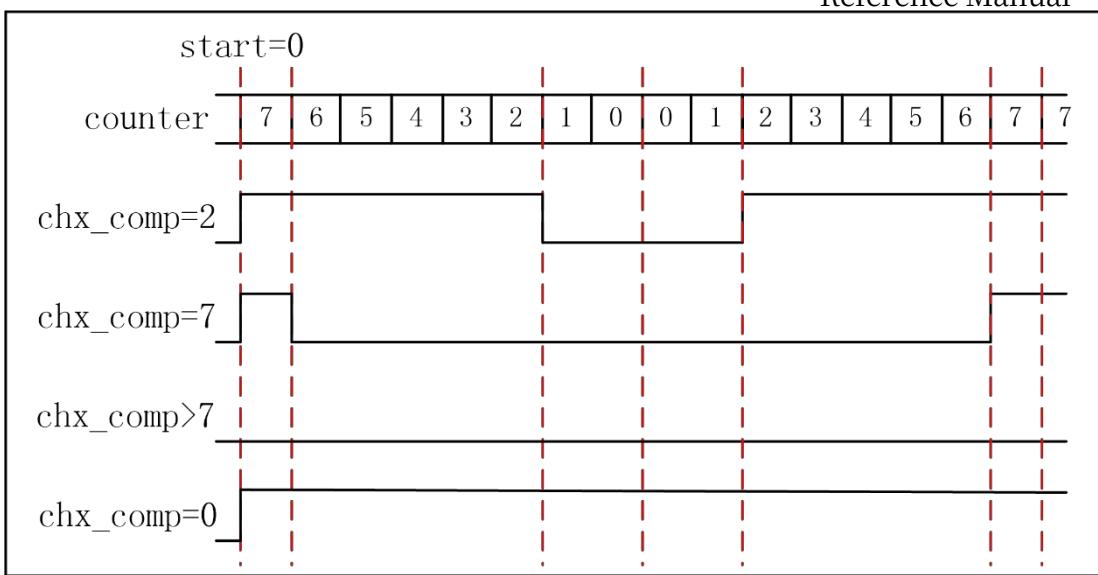


Figure 5-105 PWMPLUS Center Aligned Mode, Count Down, Start Level 0 Flip Point and Cycle Timing Chart

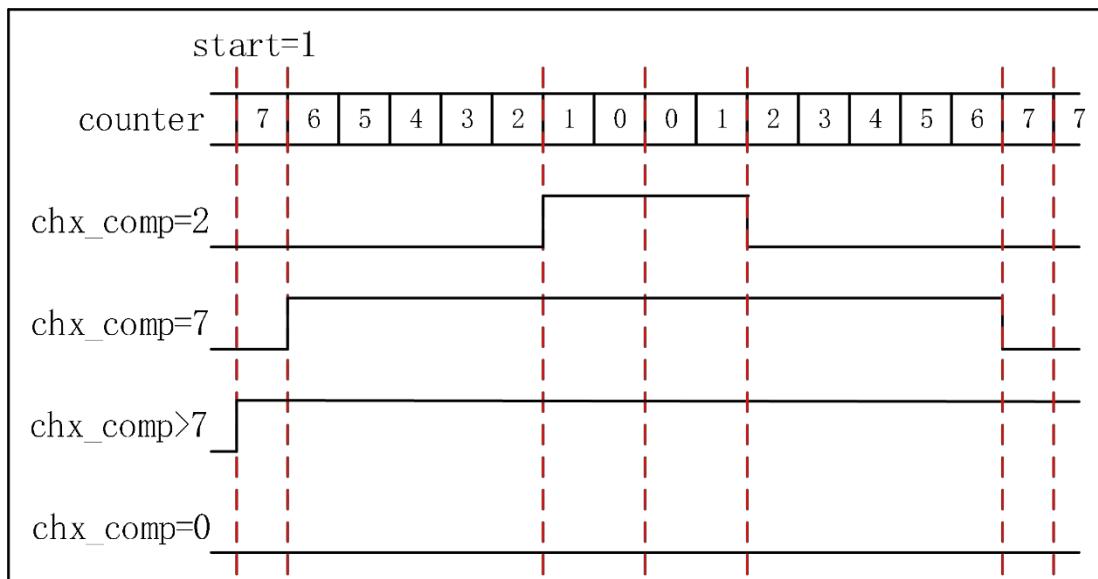


Figure 5-106 PWMPLUS Center Aligned Mode, Count Down, Flip Point and Cycle Timing Diagram with Start Level 1

Output Priority Relationship

The five conditions of deadband value calculation, channel mask, brake signal, output flip-flop, and output enable affect each channel PWM waveform output, the output waveform priority relationship is shown below:

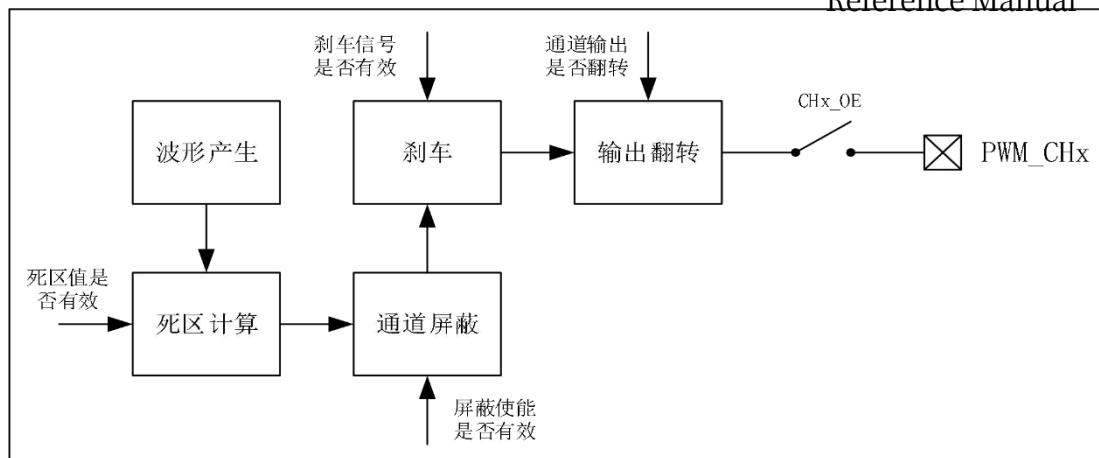


Figure 5-107 PWMPLUS Output Priority Relationship Diagram

Trigger signals and interrupts

This module can realize the trigger signals of three cases, namely, three channel flip-flops, end-of-cycle and specific trigger point, and the trigger signal is the high level of 1 pclk clock; it can also realize the generation of flip-flop, cycle overflow, characteristic trigger point and brake interrupt signals under different counting modes, among which the specific trigger point can only generate the corresponding trigger and interrupt states, and it won't change the waveforms of the PWM outputs. The following figure shows the generation of trigger signals and interrupts under various conditions, using the edge alignment mode with the starting value of 0 and counting up as an example.

1. Brake interruption

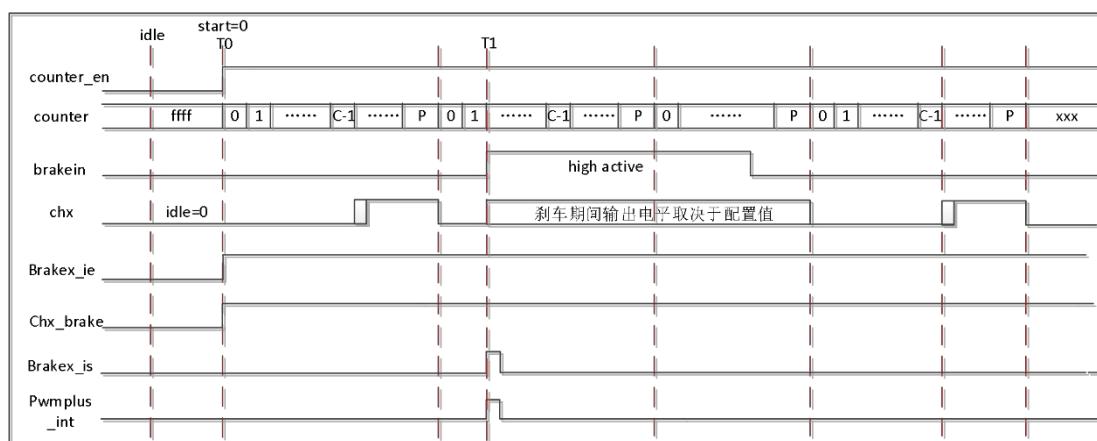


Figure 5-108 PWMPLUS Brake Interrupt Timing Diagram

DP32G030
Reference Manual

As shown in the figure above, the brake interrupt signal is controlled by the brake interrupt enable register `brake_ie` and the brake signal control register `ch_brake`, only if any channel is controlled by any bit of the input brake signal and when the corresponding bit of the input brake signal is used, the channel will be controlled by `ch_brake`.

The corresponding interrupt state can only be generated when the valid level of the car signal arrives, and when the interrupt enable is 1, the corresponding interrupt signal will be generated. As shown in the above figure, at T_0 time, turn on the brake interrupt enable, configure the brake signal control register `ch_brake`, so that any channel is controlled by the brake signal, and when the valid level of the input brake signal arrives at T_1 time, the corresponding interrupt state and interrupt signal will be generated.

2. Flip points, specific trigger points, cycle overflow interrupts and trigger signals

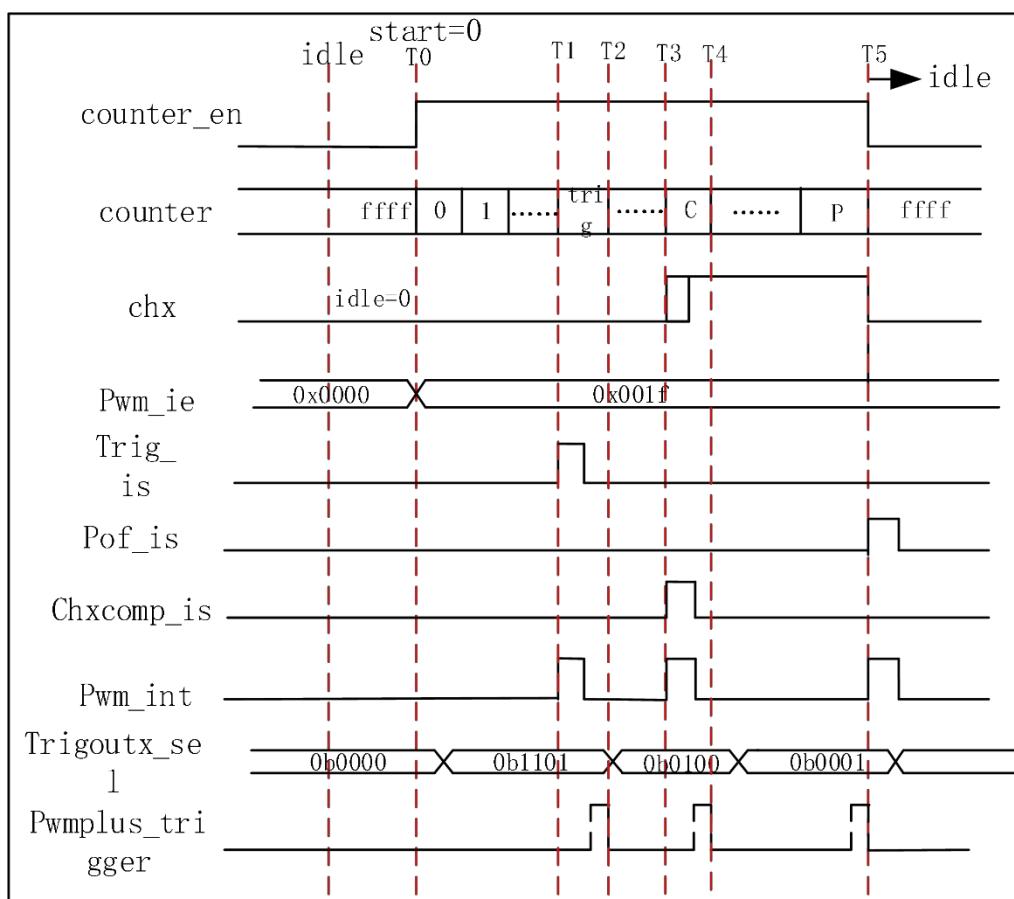


Figure 5-109 PWMPLUS Flip-Flop, Specific Trigger Points, Cycle Overflow Interrupt, and Trigger Signal Timing Diagram

As shown in the above figure, the corresponding interrupt enables are turned on at T_0 , i.e., internal trigger point interrupt, flip-flop interrupt and cycle overflow interrupt enable under up count. When the internal trigger point is reached at T_1 ,

the internal trigger point interrupt state will be generated, and the trigger signal will be generated at the same time, in which the internal trigger point only generates the interrupt state and does not affect the PWM waveform generation. Similarly, when the flip-flop and cycle overflow are reached at T3 and T5, the corresponding flip-flop interrupt state and cycle overflow interrupt state will be generated and the corresponding interrupt signal will be generated. The output function of the trigger signal can be selected through the configuration register PWMPLUS_TRIG_CFG, and the corresponding trigger will be generated when the corresponding state is reached.

Timing diagram under different trigger signal function, after configuring the corresponding output function, the output signal will be generated one system clock cycle before T2, T4, T5 time. However, the trigger signals of different functions cannot be selected at the same time, so only one output function can be selected at one time for each bit of trigger signal, and different output functions can be configured by different bit bits, and there are 4 bits in total in this module `pwm_trigger`.

Autoloading

The PWMPLUS module supports the auto-load function, which can be realized by configuring the auto-load register `AUTO_RELOAD` to indicate the number of cycle overflows after which the cycle value, the comparison value, the deadband value and the `TRIGGER` value are automatically loaded once. The auto-load action is performed after (`AUTO_RELOAD+1`) cycle overflows.

Software loading

PWMPLUS module supports software loading function, `PWM_PERIOD`, `PWM_CH0_COMP`, `PWM_CH1_COMP`, `PWM_CH2_COMP`, `PWM_CH0_DT`, `PWM_CH1_DT`, `PWM_CH2_DT` in this module,

`TRIG_COMP` These 8 registers have their own shadow registers. After software writes a 1 to the software LOAD control bit `SWLOAD`, the hardware loads the latest values stored in these 8 registers into their respective shadow registers at the end of the current cycle, which will take effect in the following cycle.

workflow

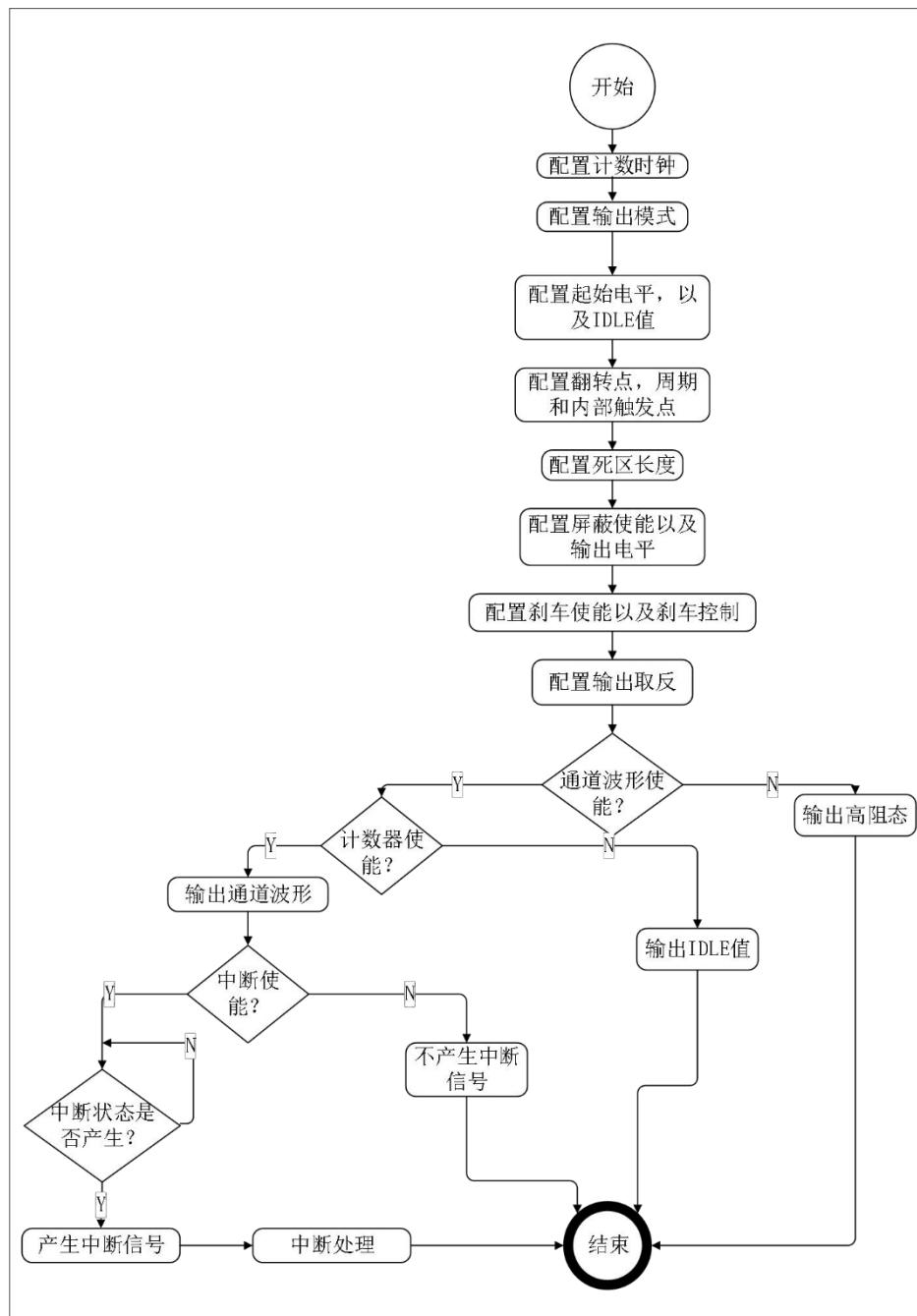


Figure 5-110 PWMPLUS Module Operation Flowchart

- Configuring PWMPLUS Clock Enable
- PORT Port Configuration for PWMPLUS Function
- Configuring the Count Clock

- Configure the output mode, count cycle mode, count behavior mode
- Configure the start level and IDLE value
- Configure cycle values, flip point values, and internal trigger point values
- Configuring the Autoload Register
- Configuring Deadband Length
- Configure mask enable and output level
- Configure brake enable, brake filter, brake signal active edge and brake output level.
- Configuring Output Flip
- If interrupts are configured, enable PWMPLUS interrupts
- Configure the channel waveform output enable, if the enable is not turned on then the PAD port outputs a high resistance state
- Configure counter enable, output IDLE value if enable is not turned on

register description

PWMPLUS_CFG register (0x00)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reservations
15:8	AUTO_RELOAD	R/W	0	Auto-Load Register, which indicates how many times after a cycle overflow the cycle value, compare value, deadband value and TRIGGER value are automatically loaded once. The autoload action performs an autoload after (AUTO_RELOAD+1) cycle overflow.
7:4	RESERVED	R	0	reservations

DP32G030

3	OUT_MODE	R/W	0	PWM output mode Reference Manual 0: Edge aligned mode output 1: Center-aligned mode output
2	CNT REP	R/W	0	PWM Counter Cycle Method 0: Single, stop after one counting cycle 1: Loop, start and count in a loop until the software stops

Reference Manual PWM Counter Behavior				
1	CNT_TYPE	R/W	0	<p>0: count up 1: Counting down If center-aligned mode, it indicates the counting behavior of the first half cycle of the counter.</p>
0	COUNTER_EN	R/W	0	<p>Counter Enable Register When this bit is set to 1, it means that the counter starts counting, and CH0/CH1/CH2 generate the PWM output waveforms of the corresponding channels according to the pre-configured period value, comparison value and deadband value. When configured for the one-shot method, this bit is automatically cleared by the hardware after one clock cycle. When configured for cyclic mode, this bit needs to be cleared to zero by the software to stop the counter counting, and the output waveform level returns to the IDLE state after one full counting cycle.</p>

Note: CH0, CH1, and CH2 share one cycle configuration register.

PWMPLUS_GEN register (0x04)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:30	RESERVED	R	0	reservations
29	CH2N_OE	R/W	0	CH2N Channel waveform output enable 0: Output off, high resistance on pin 1: Output CH2N square wave
28	CH2_OE	R/W	0	CH2 Channel waveform output enable 0: Output off, high resistance on pin 1: Output CH2 square wave

27	CH1N_OE	R/W	0	CH1N Channel waveform output enable 0: Output off, high resistance on pin 1: Output CH1N square wave
26	CH1_OE	R/W	0	CH1 Channel waveform output enable 0: Output off, high resistance on pin 1: Output CH1 square wave

				Reference Manual
25	CH0N_OE	R/W	0	CH0N Channel waveform output enable 0: Output off, high resistance state on pin 1: Output CH0N square wave
24	CH0_OE	R/W	0	CH0 Channel waveform output enable 0: Output off, high resistance state on pin 1: Output CH0 square wave
23:22	RESERVED	R	0	reservations
21	CH2N_OUTINV	R/W	0	CH2N Channel Output Status Selection 0: No change in level 1: Output level reversal
20	CH2_OUTINV	R/W	0	CH2 Channel output state selection 0: No change in level 1: Output level reversal
19	CH1N_OUTINV	R/W	0	CH1N Channel output state selection 0: No change in level 1: Output level reversal
18	CH1_OUTINV	R/W	0	CH1 Channel output state selection 0: No change in level 1: Output level reversal
17	CH0N_OUTINV	R/W	0	CH0N Channel output state selection 0: No change in level 1: Output level reversal
16	CH0_OUTINV	R/W	0	CH0 Channel output state selection 0: No change in level 1: Output level reversal
15:11	RESERVED	R	0	reservations
10	CH2_START	R/W	0	Raw CH2 Output status value at start of channel counting 0: Raw CH2 channel output 0 level 1: The original CH2 channel outputs a level of 1. When counting starts, CH2N is the inverse of CH2.

				Reference Manual
9	CH1_START	R/W	0	Raw CH1 Output status value at start of channel counting 0: Raw CH1 channel output 0 level 1: The original CH1 channel outputs 1 level After counting starts, CH1N is the inverse of CH1.
8	CH0_START	R/W	0	Output status value when raw CH0 channel starts counting 0: Raw CH0 channel output 0 level 1: The original CH0 channel outputs a level of 1. When counting starts, CH0N is the inverse of CH0.
7:6	RESERVED	R	0	reservations
5	CH2N_IDLE	R/W	1	Raw CH2N output status value when channel is idle 0: raw CH2N channel output 0 level 1: Raw CH2N channel output 1 level
4	CH2_IDLE	R/W	0	Raw CH1 Output status value when channel is idle 0: Raw CH1 channel output 0 level 1: Raw CH1 channel output 1 level
3	CH1N_IDLE	R/W	1	Raw CH1N Output status value when channel is idle 0: raw CH1N channel output 0 level 1: Raw CH1N channel output 1 level
2	CH1_IDLE	R/W	0	Raw CH1 Output status value when channel is idle 0: Raw CH1 channel output 0 level 1: Raw CH1 channel output 1 level
1	CH0N_IDLE	R/W	1	Raw CH0N Output status value when channel is idle 0: raw CH0N channel output 0 level 1: Raw CH0N channel output 1 level

0	CH0_IDLE	R/W	0	Raw CH0 Output status value when channel is idle 0: Raw CH0 channel output 0 level 1: Raw CH0 channel output 1 level
---	----------	-----	---	--

PWMPLUS_CLKSRC register (0x08)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive

				Reference Manual
31:16	PREDIV	R/W	0	Internal prescaled clock frequency selection. Use the pclk of this module as the clock source of the prescaled clock. 0x0000: indicates 1 division of pclk 0x0001: Indicates the 2-division frequency of pclk. 0x0002: Indicates the 3-division frequency of pclk 0xFFFF: indicates the 65536 division frequency of pclk
15:6	RESERVED	R	0	reserved bit
5	EXTPLUS1_EDGE	R/W	0	ExtPLUS1 Edge Selection Control for Count Clock 0: Falling edge trigger count 1: Rising edge trigger count
4	EXTPLUS0_EDGE	R/W	0	ExtPLUS0 Edge selection control when used as count clock 0: Falling edge trigger count 1: Rising edge trigger count
3	RESERVED	R	0	reserved bit
2:0	CNT_SRC	R/W	0	PWM Counter Count Clock Selection 000: Select internal pre-divided clock as count clock 001: Select extPLUS[0] as the count clock 010: Select extPLUS[1] as the count clock 011: Select tmPLUS[0] as the count clock 100: Select tmPLUS[1] as count clock 101: Select tmPLUS[2] as count clock 110: Select tmPLUS[3] as the count clock 111: Reservations

PWMPLUS_BRAKE_CFG register (0x0c)

bitfield d (mat h.)	name (of a thing)	typol ogy	reset value	descriptive
31:26	RESERVED	R	0	reserved bit

Reference Manual Digital filtering control of brake signals				
25:24	BRAKE_FILTER	R/W	0	00: No filtering 01: 2 internal pre-scaler clock filters performed 10: 4 internal pre-scaler clock filters performed 11: 8 internal pre-scaler clock filtering
23:22	RESERVED	R	0	reserved bit
21	BRAKE_CH2NPOL	R/W	0	CH2N output level selection during braking 0: Output when braking 0 1: Output when braking 1
20	BRAKE_CH2POL	R/W	0	CH2 output level selection during braking 0: Output when braking 0 1: Output when braking 1
19	BRAKE_CH1NPOL	R/W	0	CH1N output level selection when braking 0: Output when braking 0 1: Output when braking 1
18	BRAKE_CH1POL	R/W	0	CH1 output level selection during braking 0: Output when braking 0 1: Output when braking 1
17	BRAKE_CHONPOL	R/W	0	CH0N output level selection when braking 0: Output when braking 0 1: Output when braking 1
16	BRAKE_CHOPOL	R/W	0	CH0 output level selection during braking 0: Output when braking 0 1: Output when braking 1
15	RESERVED	R	0	reserved bit
14:12	BRAKE_LEV	R/W	0	Brake effective level selection Bit2 of BRAKE_LEV corresponds to brakein[2], and Bit1 corresponds to brakein[1], Bit0 corresponds to brakein[0]. 0: Indicates that the brake input is active low 1: Indicates that the brake input is active

				high	
11:9	RESERVED	R	0	reserved bit	

				Reference Manual CH2/CH2N Brake Control Selection
8:6	CH2_BRAKE	R/W	0	Bit2 of CH2_BRAKE corresponds to brakein[2], and Bit1 corresponds to brakein[1], Bit0 corresponds to brakein[0]. 0: Indicates not controlled by brake signal 1: Indicates controlled by brake signal
5:3	CH1_BRAKE	R/W	0	CH1/CH1N Brake control selection Bit2 of CH1_BRAKE corresponds to brakein[2], Bit1 corresponds to brakein[1], Bit0 corresponds to brakein[0]. 0: Indicates not controlled by brake signal 1: Indicates controlled by brake signal
2:0	CHO_BRAKE	R/W	0	CHO/CH0N Brake control selection Bit2 of CHO_BRAKE corresponds to brakein[2], Bit1 corresponds to brakein[1], Bit0 corresponds to brakein[0]. 0: Indicates not controlled by brake signal 1: Indicates controlled by brake signal

Note: When braking starts, it takes effect immediately on the output waveform;
during braking, the counter counts normally; after braking is withdrawn, each
channel outputs the waveform at the end of the complete cycle.

PWMPLUS_MASKLEV register (0x10)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:6	RESERVED	R	0	reserved bit
5	CH2N_MASKLEV	R/W	0	CH2N Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1

4	CH2_MASKLEV	R/W	0	CH2 Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1
3	CH1N_MASKLEV	R/W	0	CH1N Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1

				Reference Manual CH1 Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1
2	CH1_MASKLEV	R/W	0	CH0N Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1
1	CH0N_MASKLEV	R/W	0	CH0 Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1
0	CH0_MASKLEV	R/W	0	CH0 Channel Mask Level Selection 0: Masking period forced output 0 1: Masking period mandatory output 1

PWMPLUS_PERIOD register (0x1C)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	RO	0	reserved bit
15:0	PERIOD	R/W	0xffff	<p>PWMPLUS Cycle Configuration Register.</p> <p>The actual counting period is the value of the period configured for this register plus 1. Note 0: The period cannot be configured as 0.</p> <p>For example, if the configuration is decimal 199, the PWM waveform period is considered to be 200.</p> <p>NOTE 1: In center-aligned mode, the actual period is 2 times the configured value plus 1.</p>

PWMPLUS_CH0_COMP register (0x20)

bitfield d (math)	name (of a thing)	typology	reset value	descriptive

.				
31:16	RESERVED	R	0	reserved bit
15:0	CH0_COMP	R/W	0	CH0/CH0N Flip Point Configuration Register. Note: If the count value is less than the flip-point value, output start[0]; if it is greater than or equal to the flip-point value, output start[0] not.

PWMPLUS_CH1_COMP register (0x24)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	CH1_COMP	R/W	0	CH1/CH1N Flip Point Configuration Register. Note: If the count value is less than the flip point value, output start[1]; if it is greater than or equal to the flip point value, output start[1] not.

PWMPLUS_CH2_COMP register (0x28)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	CH2_COMP	R/W	0	CH2/CH2N Flip Point Configuration Register. Note: If the count value is less than the flip point value, output start[2]; if it is greater than or equal to the flip point value, output start[2] not.

PWMPLUS_CH0_DT register (0x30)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive

31:10	RESERVED	R	0	reserved bit	Reference Manual
9:0	CH0_DT	R/W	0	<p>CH0/CH0N Deadband Length Configuration Registers</p> <p>Note 0: A configuration of 0 means no deadband; a configuration of 1 means a deadband length of 1, a configuration of 2 means a deadband length of 2, and so on.</p> <p>Note 1: When configuring this value, it is necessary to match with the cycle value, CH0_START value and CH0_COMP value, otherwise the output waveform may not achieve the expected effect.</p>	

PWMPLUS_CH1_DT register (0x34)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:10	RESERVED	R	0	reserved bit
9:0	CH1_DT	R/W	0	<p>CH1/CH1N Deadband Length Configuration Registers</p> <p>Note 0: A configuration of 0 means no deadband; a configuration of 1 means a deadband length of 1, a configuration of 2 means a deadband length of 2, and so on.</p> <p>Note 1: When configuring this value, it is necessary to match with the period value, CH1_START value and CH1_COMP value, otherwise the output waveform may not achieve the expected effect.</p>

PWMPLUS_CH2_DT register (0x38)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:10	RESERVED	R	0	reserved bit
9:0	CH2_DT	R/W	0	<p>CH2/CH2N Deadband Length Configuration Registers</p> <p>Note 0: A configuration of 0 means no deadband; a configuration of 1 means a deadband length of 1, a configuration of 2 means a deadband length of 2, and so on.</p> <p>Note 1: When configuring this value, it is necessary to match with the period value, CH2_START value and CH2_COMP value, otherwise the output waveform may not achieve the expected effect.</p>

--	--	--	--	--

PWMPLUS_TRIG_COMP register (0x40)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	TRIG_COMP	R/W	0	Internal Trigger Point Configuration Register Note: The internal trigger point value must be less than the cycle value

PWMPLUS_TRIG_CFG register (0x44)

bitfield d (math. .)	name (of a thing)	typolo gy	reset value	descriptive
31:4	RESERVED	R	0	reserved bit
3:0	TIRGOUT_L_SEL	R/W	0	<p>Output trigger0 signal function selection</p> <p>0000: No signal output;</p> <p>0001: Up count cycle overflow point;</p> <p>0010: Down count cycle overflow point;</p> <p>0011: Up or down count cycle overflow point;</p> <p>0100: CH0 Counts up the flip point;</p> <p>0101: CH0 Count down the flip point;</p> <p>0110: CH0 Flip point up or down;</p> <p>0111: CH1 Counts up the flip point;</p> <p>1000: CH1 Counts down the flip point;</p> <p>1001: CH1 Flip point up or down;</p> <p>1010: CH2 Up Count Flip Point;</p> <p>1011: CH2 Count down flip point;</p> <p>1100: CH2 Flip point up or down;</p> <p>1101: Count up internal trigger points;</p> <p>1110: Count down internal trigger points;</p> <p>1111: Up or down internal trigger point;</p>

PWMPLUS_IE register (0x60)

bitfield d (math. .)	name (of a thing)	typolo gy	reset value	descriptive
-------------------------------	-------------------	--------------	----------------	-------------

Reference Manual				
31:20	RESERVED	R	0	reserved bit
19	AUTORELOAD_IE	R/W	0	Autoload Interrupt Enable
18	BRAK2_IE	R/W	0	Brake 2 Interrupt Enable
17	BRAK1_IE	R/W	0	Brake 1 Interrupt enable

Reference Manual				
16	BRAKO_IE	R/W	0	Brake 0 Interrupt enable
15:13	RESERVED	R	0	reserved bit
12	DOWN_TRIG_IE	R/W	0	Count down to TRIGGER Trigger Point Interrupt Enable
11	DOWN_POF_IE	R/W	0	Down count cycle overflow interrupt enable
10	DOWN_CH2COMP_IE	R/W	0	Count Down CH2 Reach Flip Point Interrupt Enable
9	DOWN_CH1COMP_IE	R/W	0	Count Down CH1 Reach Flip Point Interrupt Enable
8	DOWN_CH0COMP_IE	R/W	0	Count Down CH0 Reach Flip Point Interrupt Enable
7:5	RESERVED	R	0	reserved bit
4	UP_TRIG_IE	R/W	0	Count up to TRIGGER Trigger Point Interrupt Enable
3	UP_POF_IE	R/W	0	Up count cycle overflow interrupt enable
2	UP_CH2COMP_IE	R/W	0	Count Up CH2 Reach Flip Point Interrupt Enable
1	UP_CH1COMP_IE	R/W	0	Count Up CH1 Reach Flip Point Interrupt Enable
0	UP_CH0COMP_IE	R/W	0	Count Up CH0 Reach Flip Point Interrupt Enable

PWMPLUS_IF register (0x64)

bitfield d (math)	name (of a thing)	typolo gy	reset value	descriptive

DP32G030

.				
31:20	RESERVED	R	0	reserved bit
19	AUTORELOAD_IF	R/W	0	Auto Load Interrupt Status Write 1 Clear
18	BRAK2_IF	R/W	0	Brake 2 Interrupt status Write 1 Clear Note: This state is only generated if any channel is controlled by the Brake 2 signal.

				Reference Manual
17	BRAK1_IF	R/W	0	Brake 1 Interrupt status Write 1 Clear Note: This state is only generated if any channel is controlled by the Brake 1 signal.
16	BRAKO_IF	R/W	0	Brake 0 Interrupt status Write 1 Clear Note: This state is only generated if any channel is controlled by the Brake 0 signal.
15:13	RESERVED	R	0	reserved bit
12	DOWN_TRIG_IF	R/W	0	Count down to TRIGGER Trigger Point Interrupt Status Write 1 Zeroed Note: Edge-aligned mode, indicates that the internal trigger point is reached when configured for counting down; center-aligned mode, indicates that the internal trigger point is reached in the first half-cycle when configured for counting down or the internal trigger point is reached in the second half-cycle when configured for counting up
11	DOWN_POF_IF	R/W	0	Down Count Cycle Overflow Interrupt Status Write 1 Clear Note: Edge-aligned mode, indicates that the cycle overflow point is reached when configured for counting down; center-aligned mode, indicates that the cycle overflow point is reached in the first half-cycle when configured for counting down or in the second half-cycle when configured for counting up
10	DOWN_CH2COM_P_IF	R/W	0	Count down CH2 Reach flip-flop interrupt status write 1 clear
9	DOWN_CH1COM_P_IF	R/W	0	Count down CH1 Reach flip-flop interrupt status write 1 clear

				Count down CH0 Reference Manual flop interrupt status write 1 clear
8	DOWN_CH0COM P_IF	R/W	0	Note: Edge-aligned mode means that the flip point is reached when configured to count down; center-aligned mode means that the flip point is reached in the first half cycle when configured to count down or in the second half cycle when configured to count up. The other channel behaviors are the same.
7:5	RESERVED	R	0	reserved bit

				Reference Manual Trigger Point
4	UP_TRIG_IF	R/W	0	Count up to TRIGGER Trigger Point Interrupt Status Write 1 Zeroed Note: Edge-aligned mode means that the internal trigger point is reached when configured to count up; center-aligned mode means that the internal trigger point is reached in the first half-cycle when configured to count up or in the second half-cycle when configured to count down
3	UP_POF_IF	R/W	0	Up Count Cycle Overflow Interrupt Status Write 1 Clear Note: Edge-aligned mode means that the cycle overflow point is reached when configured for counting up; center-aligned mode means that the cycle overflow point is reached in the first half of the cycle when configured for counting up or in the second half of the cycle when configured for counting down
2	UP_CH2COMP_IF	R/W	0	Count up CH2 Reach flip-flop interrupt status write 1 clear
1	UP_CH1COMP_IF	R/W	0	Count up CH1 Reach flip-flop interrupt status write 1 clear
0	UP_CH0COMP_IF	R/W	0	Count up CH0 Reach flip-flop interrupt status write 1 clear Note: Edge-aligned mode means that the flip point is reached when configured to count up; center-aligned mode means that the flip point is reached in the first half-cycle when configured to count up or the flip point is reached in the second half-cycle when configured to count down. The other channel behaviors are the same.

PWMPLUS_SWLOAD Register (0x84)

bitfield d (math .)	name (of a thing)	typolo gy	reset value	descriptive
31:1	RESERVED	R	0	reserved bit
0	SWLOAD	R/W	0	<p>PWM Configuration Register Software LOAD Control Bit</p> <p>In this module, the eight registers PWM_PERIOD, PWM_CH0_COMP, PWM_CH1_COMP, PWM_CH2_COMP, PWM_CH0_DT, PWM_CH1_DT, PWM_CH2_DT, TRIG_COMP, PWM_CH0_DT, PWM_CH1_DT, PWM_CH2_DT, and TRIG_COMP are used.</p> <p>The memory has its own shadow register. After software writes a 1 to this bit, the hard</p> <p>The latest values stored in these 8 registers will be used by the device at the end of the current cycle.</p> <p>load into their respective shadow registers and take effect on the next cycle.</p>

PWMPLUS_MASK_EN register (0x88)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:6	RESERVED	R	0	reserved bit
5	CH2N_MASK_EN	R/W	0	CH2N Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.
4	CH2_MASK_EN	R/W	0	CH2 Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.
3	CH1N_MASK_EN	R/W	0	CH1N Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.
2	CH1_MASK_EN	R/W	0	CH1 Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.
1	CH0N_MASK_EN	R/W	0	CH0N Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.
0	CH0_MASK_EN	R/W	0	CH0 Channel Mask Enable Writing this bit to 1 activates the forced output mask function, and the forced output level is selected via MASK_CFG.

Note: When the blocking function of each channel is enabled, the forced output takes effect immediately. When the masking function is deactivated, the output will be normalized according to the original waveform immediately.

PWMPLUS_CNT_ST register (0xE0)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:18	RESERVED	R	0	reserved bit

				Reference Manual PWM Counter Operating Status
17	CNT_ST	R	0	0: Indicates that the counter is not working 1: Indicates that the counter is counting
16	CNT_DIR	R	0	PWM Counter Current Count Direction 0: Indicates that the counter is currently counting up 1: Indicates that the counter is currently counting down
15:0	PWMPLUS_CNT	R	0	PWM Counter Current Count Value Register

PWMPLUS_BRAKE_ST register (0xE4)

bitfield d (math)	name (of a thing)	typology	reset value	descriptive
31:2	RESERVED	R	0	reserved bit
1:0	BRAKE_ST	R	0	Current status of brake input signal BRAKE_ST[1] corresponds to brake1 and BRAKE_ST[0] corresponds to brake0.

5.15 Real Time Clock (RTC)

5.15.1 summarize

The real-time clock is an independent timer. the RTC module has a real-time clock function that automatically solves the leap year problem. the input clock source can be selected as RCLF or XTAL, which can output 1/2 second and its interrupt. the corresponding clock needs to be turned on before using the RTC. The system block diagram is shown below:

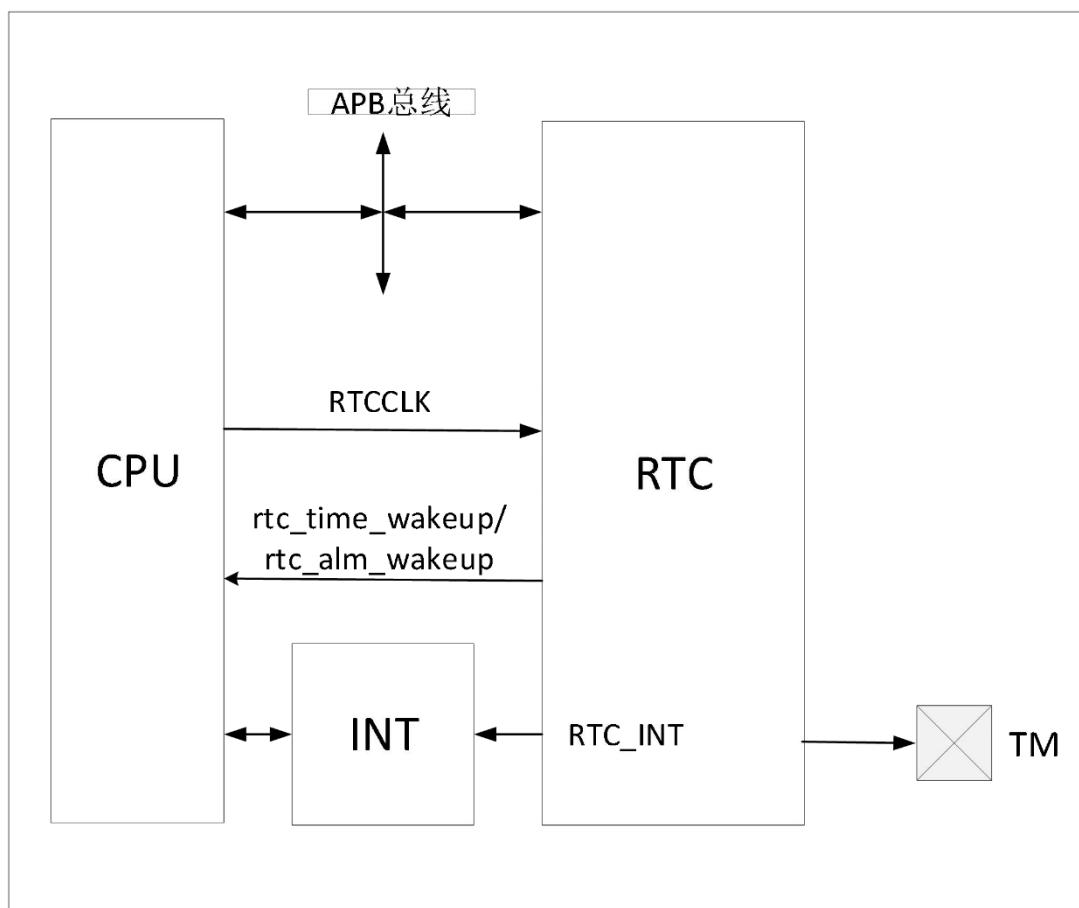


Figure 5-111 RTC Module System Block Diagram

5.15.2 characterization

- Real-time clock function

DP32G030
Reference Manual

- Alarm clock function

- Calendar function, chronograph range 0-99 years
- Automatic resolution of leap year issues
- Outputs seconds, minutes, hours, days, and alarm interrupts synchronized with the RTC
- RTC counting clock can be selected from two clock sources
- Outputs 1/2 second and its interrupt
- Outputs RTC Wake-on-Time and Wake-on-Alarm signals for low-power wake-up modules
- Data validity check for clock setting and alarm setting
- Alarm register with BCD coding
- Time register with BCD coding

5.15.3 Module Structure Diagram

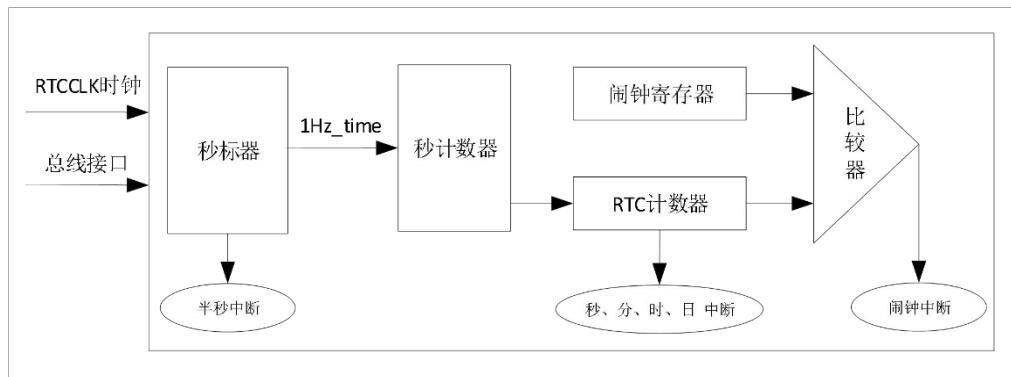


Figure 5-112 RTC Module Block Diagram

5.15.4 Function Description

Count Clock

Source

Selection

Two clock sources can be selected for the RTC counter clock (RTCCLK) of this chip: an on-chip low-frequency RC oscillator (RCLF) and an off-chip low-frequency crystal oscillator (XTAL).

The RTCCLK clock source can be provided by either the XTAL or RCLF clock by setting the RTC_CLK_SEL bit in the SRC_CFG register. The RTC configuration (clock source selection, etc.) must be completed before the RTC can be enabled.

The RTCCLK selection schematic is shown below:

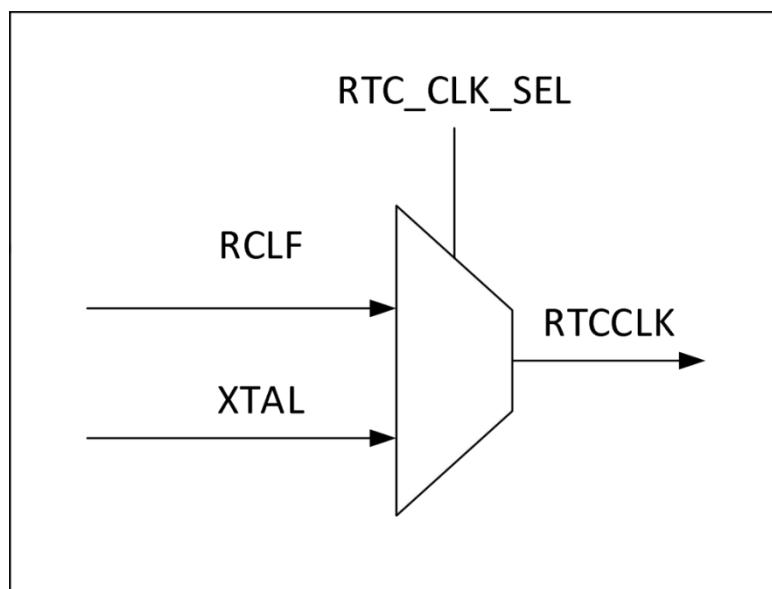


Figure 5-113 RTCCLK Selection Diagram

second marking function

The second marker generates the second marker time (1Hz_time) used to update the

calendar, and can also be generated by configuring the RTC register to a 1/2 second time.

The `PRE_ROUND` bit of the `RTC_PRE` register is the integer portion of the prescaler, `PRE_PERIOD` refers to the number of seconds for each fractional portion calculation, and `PRE_DECIMAL` is the fractional portion of the prescaler. The specific configuration value is determined according to which accuracy is better when comparing `PRE_PERIOD` and `PRE_DECIMAL`, with an error range of $\pm 1\text{ppm}$ for 1 second.

For example 1:

RTC clock frequency is 32767.62Hz, then PRE_ROUND is configured as 32766; the decimal part is calculated as $0.62*8=4.96$ and $0.62*16=9.92$ respectively, then selecting 8 seconds needs to be configured as 5, selecting 16 seconds needs to be configured as 10, it can be seen that the two methods have the same accuracy, and the priority is to select 8 seconds, and the PRE_DECIMAL is configured as 5, and the PRE_PERIOD is configured as 0. PRE_DECIMAL is configured as 5 and PRE_PERIOD is configured as 0.

The error for 1 second is $(32767+5/8)/32767.62) - 1 = 0.15\text{ppm}$

For example 2:

RTC clock frequency is 32767.71Hz, then PRE_ROUND is configured as 32766; the decimal part is calculated as $0.71*8=5.68$ and $0.71*16=11.36$ respectively, then select 8 seconds to be configured as 6 and 16 seconds to be configured as 11, which shows that selecting 16 seconds is more accurate, and priority is given to selecting 16 seconds, PRE_DECIMAL is 11, PRE_PERIOD is 1. PRE_DECIMAL is configured as 11 and PRE_PERIOD is configured as 1.

The error for 1 second is $(32767+11/16)/32767.71)-1 = -0.68\text{ppm}$

For example, 3:

RTC clock frequency is 32770.094Hz, then PRE_ROUND is configured as 32769; the decimal part is calculated as $0.094*8=0.752$ and $0.094*16=1.504$ respectively, then selecting 8 seconds needs to be configured as 1, selecting 16 seconds needs to be configured as 1, which shows that the precision of selecting 8 seconds is higher, and priority is given to selecting 8 seconds, and the PRE_DECIMAL is configured as 1, and the PRE_PERIOD is configured as 0. PRE_DECIMAL is configured as 1 and PRE_PERIOD is configured as 0.

Then the error for 1 second is $(32770+1/8)/32770.094)-1 = 0.946\text{ppm}$

calendar function

The real-time clock accurately times the year, month, day, hour, minute, and second based on the `1Hz_time` generated by the second scaler and outputs the timing time. The RTC accepts the time setting and can set the time to start timing.

The calendar function has a set of configuration time registers and current time registers in BCD code format. The configuration time register is equipped with an automatic time format checking function (except for the day of the week) which is able to determine whether the set time is legal or not according to the value written by the user. If the time format is not legal (e.g., January 34, February 30, 24:01, etc.) the `TIME_ERR` bit in the status register `RTC_IF` will indicate that the set time data format is incorrect. The configured time value will not take effect even if `LOAD_EN` is turned on.

Alarm clock function

The RTC module provides an alarm function that can generate an interrupt when the counter reaches the alarm setting.

Alarm clock setting is performed through the `RTC_AR` register, which only supports day of the week, hour, minute and second. The alarm register has an automatic time format checking function (except for day of the week) which is able to judge whether the set time is legal or not according to the value written by the user. If the format written to the alarm register is not legal (e.g. 20:20:63, 24:01, etc.), the alarm format error will be queried in the `ALM_ERR` bit in the status register `RTC_IF`, and then even if the alarm function enable bit is turned on, the alarm will not take effect. When the day of the week is set to all 0, the alarm will respond only once, otherwise the alarm will respond cyclically until it is turned off.

interrupt function

This module provides six interrupt functions, including seconds, minutes, hours, days, alarms, and half-second interrupts. The interrupts can be accessed through the `RTC_IE`

Configure the corresponding interrupt enable bits in the `RTC_IF`

register, and view the status bits of each corresponding

interrupt through the `RTC_IF` register. The interrupt flags and

interrupt schematic are as follows:

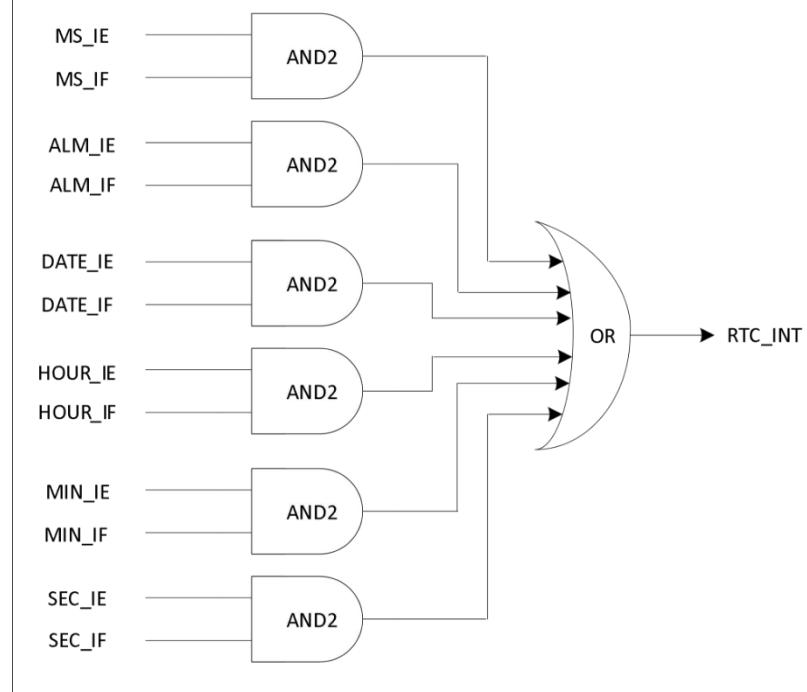


Figure 5-114 RTC Interrupt Flag and Interrupt Diagram

Low-power wake-up function

This module provides two low-power wake-up functions to support waking up the system in SLEEP mode and DEEPSLEEP mode. They are Wake on Time and Wake on Alarm, respectively.

By configuring the time and alarm of RTC, the corresponding wake-up signal can be generated to the PMU wake-up circuit, and by configuring the LPMD_WKEN register through the PMU wake-up circuit, the corresponding wake-up enable can be turned on, and the wake-up status can be queried through the LPMD_WKST register.

workflow

Initialization settings

1. Configure the RTC_PRE register to output 1Hz_time according to RTCCLK;
2. Configure the RTC_TR and RTC_DR registers;
3. Read the TIME_ERR bit of the status register RTC_IF to confirm the correct format;
4. Configure the LOAD_EN bit of RTC_CFG to 1 to turn on the setup time function;
5. Configure the RTC_AR register;
6. Read the ALM_ERR bit of the status register RTC_IF to confirm the correct format;
7. Configure the ALM_EN bit of RTC_CFG to 1 to turn on the alarm clock function;
8. Configure the RTC_EN bit of RTC_CFG to be 1, open RTC and start timing; Note: If you don't need the time setting

DP32G030
Reference Manual

function, skip 2-4; if you don't need the alarm clock

function, skip 5-7.

time setting

1. Configure the RTC_TR and RTC_DR registers;
2. Read the TIME_ERR bit of the status register RTC_IF to confirm the correct format;
3. Configure the LOAD_EN bit of RTC_CFG to 1 to turn on the setup time function;
4. The RTC timer loads the new value and starts timing at the next RTCCLK clock cycle.

Alarm settings

1. Configure the RTC_AR register;
2. Read the ALM_ERR bit of the status register RTC_IF to confirm the correct format;
3. Configure the ALM_EN bit of RTC_CFG to 1 to turn on the alarm clock function;
4. Wait for the alarm to respond.

retrieve time

1. Read and judge RTC_VALID to be 1;
2. Read registers RTC_TSTR, RTC_TSDR to get the current time value;
3. Read RTC_CNT to get the current count value of the second scale;

Interrupt Response Setting

1. Configure RTC_IE, there are 6 kinds of interrupts can be configured separately;

2. Wait for the interrupt response;
3. Read the RTC_IF register and query the current interrupt state
4. Write 1 to the corresponding bit of RTC_IF register to clear the corresponding interrupt.

register map

name (of a thing)	offset	bit width	typolog	reset value	descriptive
RTCBASE: 0x40069000					
RTC_CFG	0x00	32	R/W	0x00	RTC Configuration Register
RTC_IE	0x04	32	R/W	0x00	RTC Interrupt Enable Register
RTC_IF	0x08	32	R/W	0x00	RTC Status Register
RTC_PRE	0x10	32	R/W	0x7fff	RTC prescaler register
RTC_TR	0x14	32	R/W	0x00	RTC Time Register
RTC_DR	0x18	32	R/W	0x101	RTC Date Register
RTC_AR	0x1C	32	R/W	0x00	RTC Alarm Clock Register
RTC_TSTR	0x20	32	RO	0x00	RTC Current Time Register
RTC_TSDR	0x24	32	RO	0x101	RTC Current Date Register
RTC_CNT	0x28	32	RO	0x00	RTC Second Scale Current Count Value
RTC_VALID	0x2C	32	RO	0x00	RTC Current Time Valid Flag Register

register description

RTC_CFG register (0x00)

bitfiel	name (of a	typolo	reset	descriptive
---------	------------	--------	-------	-------------

DP32G030

d (math.)	thing)	gy	value	
31:3	RESERVED	RO	0x0	reserved bit

RTC Load Register RTC_TR and RTC_DR Time Reference Manual				
2	LOAD_EN	R/W, AC	0x0	<p>RTC Load Register RTC_TR and RTC_DR Time Reference Manual</p> <p>Setting Value</p> <p>1: Load the value set in the registers, hardware auto zeroing</p> <p>0: The value set in the register is not loaded</p> <p>Note: It is recommended to check RTC_IF after RTC_TR and RTC_DR have been configured.</p> <p>TIME_ERR bit to 0, then turn this bit on again</p>
1	ALM_EN	R/W	0x0	<p>RTC Alarm Clock Function Enable Bit</p> <p>1: Turn on the alarm clock function, only when ALM_WEEKDAY is 0, this bit is automatically cleared by the hardware</p> <p>0: Turn off the alarm clock function</p> <p>Note: It is recommended to check that the ALM_ERR bit of RTC_IF is 0 after RTC_AR is configured, and then turn this bit on again.</p>
0	RTC_EN	R/W	0x0	<p>RTC enable bit</p> <p>1: Open the RTC</p> <p>0: Turn off RTC</p>

RTC_IE register (0x04)

bitfield Id (mat h.)	name (of a thing)	typology	reset value	descriptive
31:6	RESERVED	RO	0x0	reserved bit
5	MS_IE	R/W	0x0	<p>1/2 second interrupt enable bit</p> <p>1: Enabling</p> <p>0: not enabled</p>
4	ALM_IE	R/W	0x0	<p>Alarm clock interrupt enable bit</p> <p>1: Enabling</p> <p>0: not enabled</p>

3	DATE_IE	R/W	0x0	Day Interrupt Enable Bit 1: Enabling 0: not enabled	Reference Manual
2	HOUR_IE	R/W	0x0	Hourly Interrupt Enable Bit 1: Enabling 0: not enabled	

Minute Interrupt Enable Bit Reference Manual				
1	MIN_IE	R/W	0x0	Minute Interrupt Enable Bit 1: Enabling 0: not enabled
0	SEC_IE	R/W	0x0	Seconds Interrupt Enable Bit 1: Enabling 0: not enabled

RTC_IF register (0x08)

bitfield d (math.)	name (of a thing)	typolog y	reset value	descriptive
31:10	RESERVED	RO	0x0	reserved bit
9	ALM_ERR	RO	0x0	Alarm Setting Valid Flag Bits 1: Wrong alarm setting 0: Alarm clock set correctly Note: This bit does not judge the alarm clock ALM_WEEKDAY
8	TIME_ERR	RO	0x0	Time Setting Valid Flag Bit 1: Time setting error 0: Time set correctly Note: This bit is also valid if BCD_WEEKDAY is configured as 7.
7:6	RESERVED	RO	0x0	reserved bit
5	MS_IF	R, W1C	0x0	1/2 sec interrupt response, active high, write 1 clear
4	ALM_IF	R, W1C	0x0	Alarm Clock Interrupt Response, Active High, Write 1 Clear
3	DATE_IF	R, W1C	0x0	Day Interrupt Response, Active High, Write 1 Clear

Reference Manual				
2	HOUR_IF	R, W1C	0x0	Hourly Interrupt Response, High Active, Write 1 Clear
1	MIN_IF	R, W1C	0x0	Minute Interrupt Response, Active High, Write 1 Clear
0	SEC_IF	R, W1C	0x0	Interrupt Response in seconds, active high, write 1 clear

RTC_PRE register (0x10)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:25	RESERVED	RO	0x0	reserved bit
24	PRE_PERIOD	R/W	0x0	Fractional calculation cycle selection 0:8 seconds 1:16 seconds Note: Decimal part calculations are performed at intervals of seconds
23:20	PRE_DECIMAL	R/W	0x0	prescaled decimal fraction 0: indicates no decimal 1: indicates 1/8 or 1/16 ... 7: indicates 7/8 or 7/16 8: indicates 8/16 ... 15: indicates 15/16
19:0	PRE_ROUND	R/W	0x7fff	prescaled integer part The count value is the configured value plus 1

RTC_TR register (0x14)

bitfield d (math.)	name (of a thing)	typolog y	reset value	descriptive
31:27	RESERVED	RO	0x0	reserved bit

26:24	BCD_WEEK	R/W	0x0	Setting the day of the week for the time, 0-6 is legal data. 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
23:22	RESERVED	RO	0x0	reserved bit
21:20	BCD_HOUR_DEC	R/W	0x0	Ten digits (decimal)of the hour to which the set time belongs, 0-2 valid
19:16	BCD_HOUR	R/W	0x0	Set the number of digits (decimal)of the hour to which the time belongs, 0-9 is valid
15	RESERVED	RO	0x0	reserved bit
14:12	BCD_MIN_DEC	R/W	0x0	Ten digits (decimal)of the minute to which the set time belongs, 0-5 valid
11:8	BCD_MIN	R/W	0x0	Set the number of digits (decimal)of the minute to which the time belongs, 0-9 valid
7	RESERVED	RO	0x0	reserved bit
6:4	BCD_SEC_DEC	R/W	0x0	Ten digits (decimal)of the second to which the set time belongs, 0-5 valid
3:0	BCD_SEC	R/W	0x0	Setting the number of digits (decimal)of the second to which the time belongs, 0-9 is valid

RTC_DR register (0x18)

bitfield d (math)	name (of a thing)	typolo gy	reset value	descriptive

DP32G030

.)				
31:24	RESERVED	R/W	0x0	reserved bit
23:20	BCD_YEAR_DEC	R/W	0x0	Setting the ten digits (decimal) of the year to which the time belongs, 0-9 is valid
19:16	BCD_YEAR	R/W	0x0	Sets the number of digits (decimal) of the year to which the time belongs, 0-9 is valid
15:13	RESERVED	RO	0x0	Reserved Bits.

12	BCD_MONTH_DEC	R/W	0x0	Set the ten digits (decimal) of the month to which the time belongs, 0-1 is valid <small>Reference Manual</small>
11:8	BCD_MONTH	R/W	0x1	Sets the number of digits (decimal) of the month to which the time belongs, 0-9 is valid
7:6	RESERVED	RO	0x0	Reserved Bits.
5:4	BCD_DATE_DEC	R/W	0x0	Set the ten digits (decimal) of the date to which the time belongs, 0-3 valid
3:0	BCD_DATE	R/W	0x1	Set the number of digits (decimal) of the date to which the time belongs, 0-9 is valid

RTC_AR register (0x1C)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
31	RESERVED	RO	0x0	reserved bit
30:24	ALM_WEEKDAY	R/W	0x0	<p>The week of the alarm time, each bit indicates one of the days of the week.</p> <p>bit0=1: Sunday</p> <p>bit1=1: Monday</p> <p>bit2=1: Tuesday</p> <p>bit3=1: Wednesday</p> <p>bit4=1: Thursday</p> <p>bit5=1: Friday</p> <p>bit6=1: Saturday</p> <p>Note: With all zeros, the alarm responds only once; with other values, the alarm</p>

DP32G030

Referentiel MM EN

				responds multiple times until MM EN is configured to shut off
23:22	RESERVED	RO	0x0	reserved bit
21:20	ALM_HOUR_DEC	R/W	0x0	Ten digits of the hour to which the alarm time belongs (decimal) 0-2 Valid
19:16	ALM_HOUR	R/W	0x0	Number of digits (decimal)of the hour to which the alarm time belongs, 0-9 valid
15	RESERVED	RO	0x0	reserved bit
14:12	ALM_MIN_DEC	R/W	0x0	Ten digits of the minute to which the alarm time belongs (decimal) 0-5 valid
11:8	ALM_MIN	R/W	0x0	Single digit (decimal)of the minute to which the alarm time belongs, valid from 0 to 9
7	RESERVED	RO	0x0	reserved bit

6:4	ALM_SEC_DEC	R/W	0x0	Reference Manual Ten digits of the second to which the alarm time belongs (decimal) 0-5 valid
3:0	ALM_SEC	R/W	0x0	Number of digits (decimal) of the second to which the alarm time belongs, 0-9 valid

RTC_TSTR register (0x20)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:27	RESERVED	RO	0x0	Reserved Bits.
26:24	WEEKDAY	RO	0x0	The week of the current time, 0-6 are legal data. 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
23:22	RESERVED	RO	0x0	Reserved Bits.
21:20	HOUR_DEC	RO	0x0	Ten digits of the hour to which the current time belongs (decimal) 0-2 valid
19:16	HOUR	RO	0x0	The number of digits (decimal) of the hour to which the current time belongs, 0-9 is valid
15	RESERVED	RO	0x0	Reserved Bits.
14:12	MIN_DEC	RO	0x0	Ten digits (decimal) of the minute to which the current time belongs, 0-5 valid
11:8	MIN	RO	0x0	The number of digits (decimal) of the minute to

9

				which the current time belongs, valid 0 to 9
7	RESERVED	RO	0x0	Reserved Bits.
6:4	SEC_DEC	RO	0x0	Ten digits of the second to which the current time belongs (decimal) 0-5 valid
3:0	SEC	RO	0x0	Number of digits (decimal) of the second to which the current time belongs, 0-9 valid

RTC_TSDF register (0x24)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:25	RESERVED	RO	0x0	reserved bit
24	LEAPYEAR	RO	0x0	
23:20	YEAR_DEC	RO	0x0	Ten digits (decimal) of the year to which the current time belongs, 0-9 valid
19:16	YEAR	RO	0x0	Number of digits (decimal) of the chronological year to which the current time belongs, 0-9 valid
15:13	RESERVED	RO	0x0	reserved bit
12	MONTH_DEC	RO	0x0	Ten digits of the month to which the current time belongs (decimal) 0-1 valid
11:8	MONTH	RO	0x1	The number of digits (decimal) of the month to which the current time belongs, 0-9 Valid
7:6	RESERVED	RO	0x0	reserved bit
5:4	DATE_DEC	RO	0x0	Ten digits (decimal) of the date to which the current time belongs, 0-3 valid
3:0	DATE	RO	0x1	Number of digits (decimal) of the date to which the current time belongs, 0-9 valid

RTC_CNT register (0x28)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive

31:20	RESERVED	RO	0x0	reserved bit	Reference Manual
19:0	CNT_20	RO	0x0	20bit counting bits	

RTC_VALID register (0x2C)

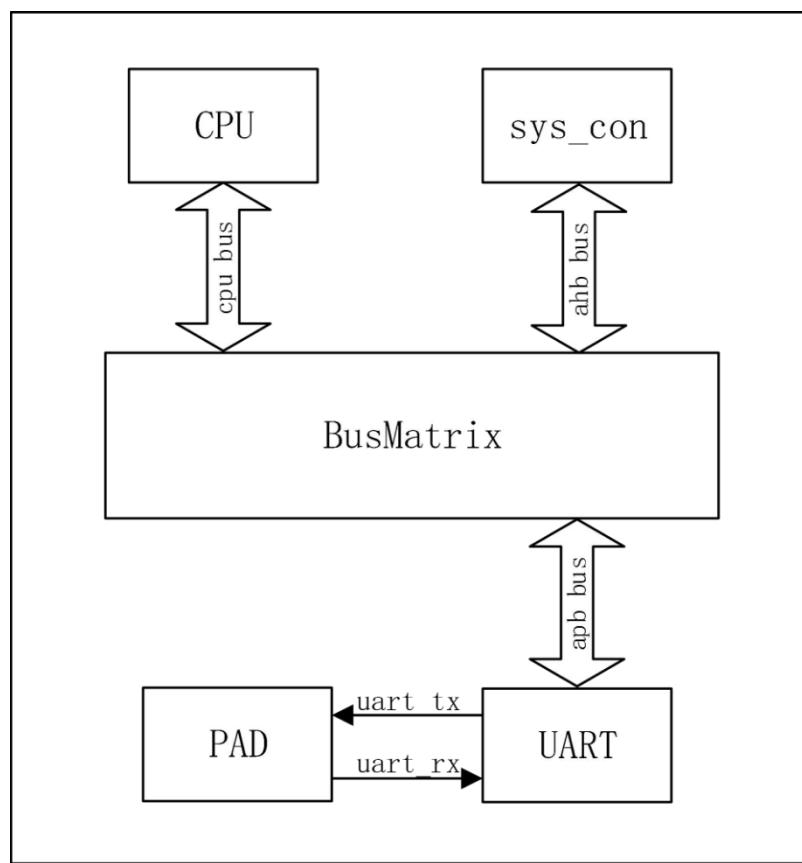
bitfield d (math.)	name (of a thing)	typology	reset value	descriptive

31:1	RESERVED	RO	0x0	reserved bit	Reference Manual
0	CUR_VALID	RO	0x0	Current time valid flag When this bit is judged to be 1, the current time register can be read. (RTC_TSTR, RTC_TSDR, RTC_CNT)	

5.16 UART Controller (UART)

5.16.1 summarize

Universal Asynchronous Receiver/Transmitter (UART) is a communication technology for serial communication, commonly used in board-level communication between microcontrollers and computers, as well as between microcontrollers and microcontrollers, which is asynchronous communication, and the port is able to send data on one line while receiving data on another line. The most important parameters for serial communication are baud rate, data bits, stop bits and parity. This chip has 3 serial ports, supports baud rate configuration, data length, parity bit, stop bit can be configured to support a variety of interrupts, support for DMA transmission mechanism, with hardware automatic flow control function, receive and transmit respectively independent with a depth of 8 FIFO, and FIFO water level can be configured to use the corresponding UART clock needs to be enabled before. The module system connection diagram is shown below:



DP32G030

Figure 5-115 UART Module System Block Diagram Reference Manual

The above figure shows the system connection diagram of UART module. The CPU can directly control the UART module through the APB bus interface.

The UART module realizes the communication between the chip and external devices via PAD.

5.16.2 characterization

- Supports standard UART protocol
- Supports full duplex mode
- Supports baud rate configuration
- Programmable data bit length (8bit or 9bit)
- Configurable parity bit Odd parity Even parity constant 0 Constant 1
- Supports 1-bit stop bit and configurable transmit delay time
- Support baud rate auto-detection function
- With hardware automatic flow control
- Supports DMA transfer mechanism
- Separate FIFO depths of 8 for receive and transmit, with configurable FIFO levels
- Supports send completion interrupt, receive completion interrupt, receive timeout interrupt function

5.16.3 Block Diagram of Module Structure

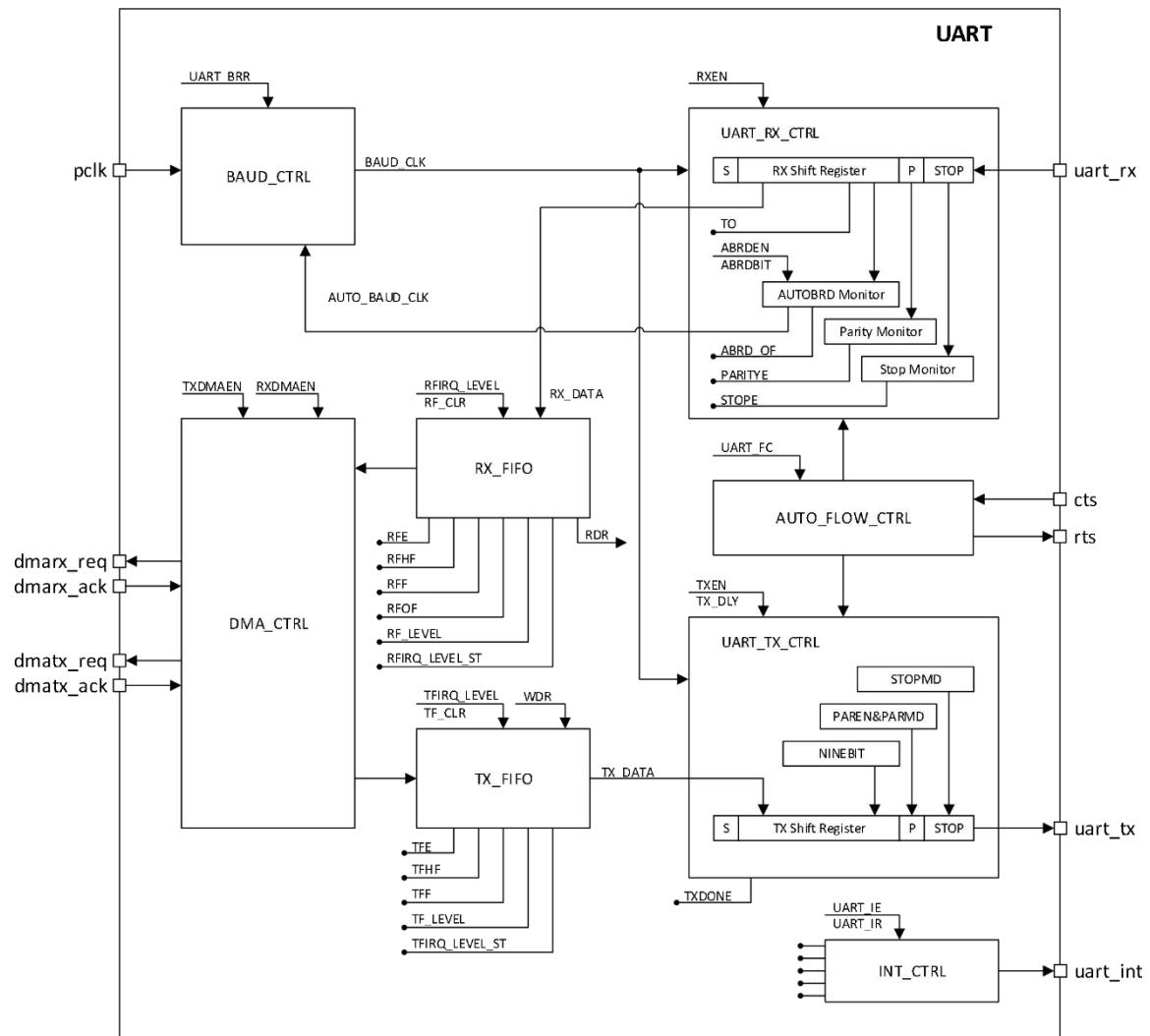


Figure 5-116 UART Module Block Diagram

The UART generates the sample clock BAUD_CLK through the internal baud rate controller BAUD_CTRL according to the baud rate parameter set by UART_BAUD, which can be used for data sampling by UART_RX_CTRL on the receiving side and data transmission by UART_TX_CTRL on the transmitting side.

The UART has two built-in FIFOs with a depth of 8. The RX_FIFO is used to store valid data sampled by the UART_RX_CTRL. and TX_FIFO for storing data to be sent written by the CPU or DMA.

The data receiver UART_RX_CTRL samples the data of UART_RX synchronously and simultaneously, analyzes the sampled data, checks whether the data parity bit and

stop bit are the same as the set mode, thus generating the correct bits, and saves the valid data `RX_DATA`, which is free from parity bit error and stop bit error, into `RX_FIFO`.

The data transmitter **UART_RX_CTRL** receives the data to be sent from **RX_FIFO**, and sends it one **BIT** at a time according to **BAUD_CLK** with the data bit, parity bit, and stop bit set.

The **UART** supports the **DMA** function, and the **DMA** control of the **UART** is realized through the internal **DMA_CTRL** and the corresponding interface of **DMA**.

Data reception and transmission.

The **UART** implements hardware flow control via the internal **AUTO_FLOW_CTRL**.

Functional Description

Receive timing and related status signals

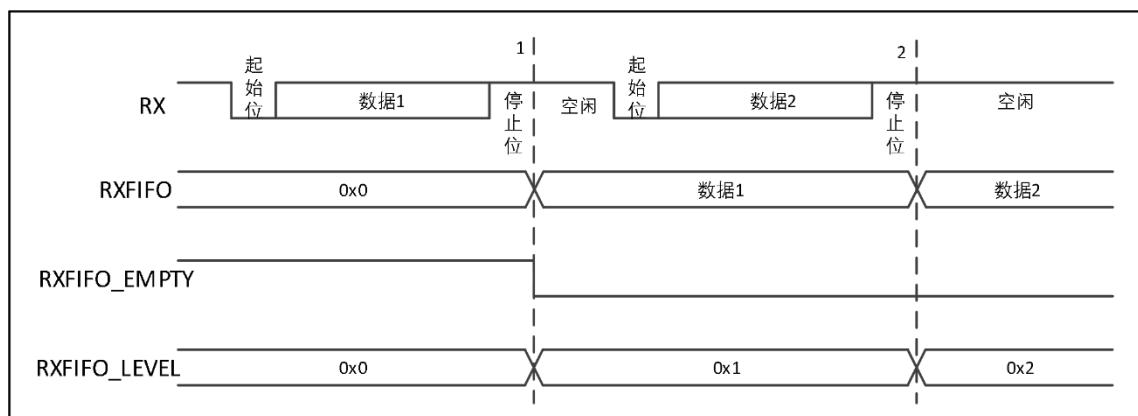


Figure 5-117 UART Receive Timing and Related Status Signal Diagrams

The above figure shows the simple timing diagram of **UART** receiving. At flag 1, **UART** saves the received data into **RXFIFO**, **RXFIFO** level increases, and **RXFIFO** null is set low; at flag 2, **UART** saves the received data into **RXFIFO**, and **RXFIFO** level increases.

Transmit timing and related status signals

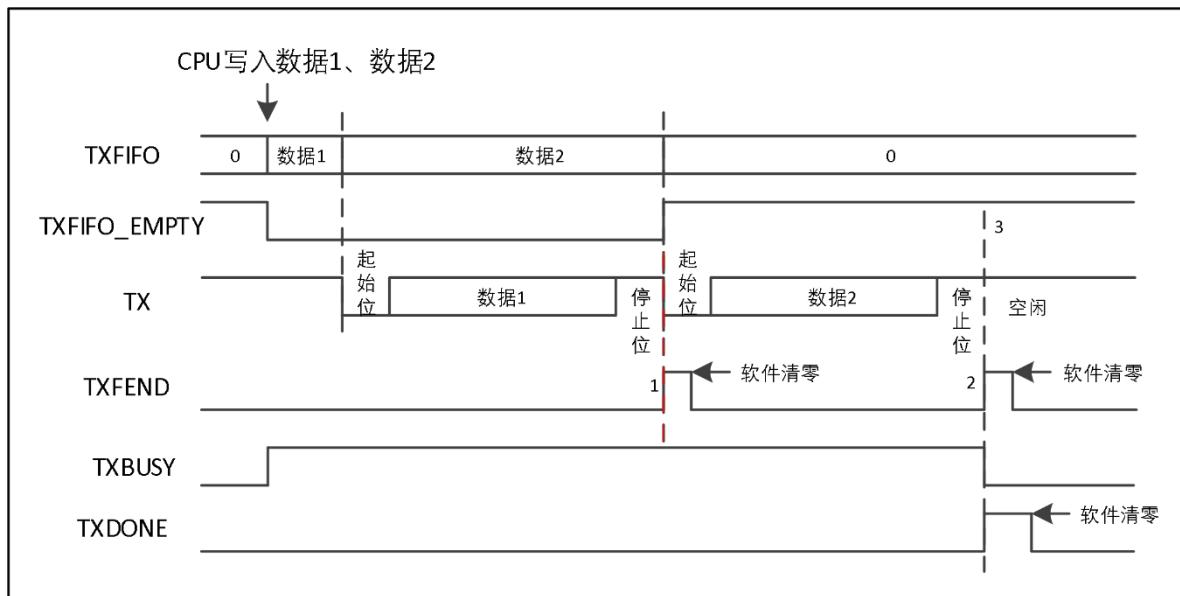


Figure 5-118 UART Transmission Timing and Related Status Signal Diagrams

The above figure shows the simple timing diagram of UART transmission, every time a group of data is sent, TXFEND will be set to 1, see figures 1 and 2 above.

The following is an example of the TXDONE function. TXDONE is set to 1 when all data is sent.

TXBUSY is active when there is pending data in TXFIFO and the transmit shift register is in transit.

baud rate calculation

The baud rate formula is as follows:

$$f_{baudrate} = \frac{f_{pclk}}{UARTDIV}$$

For example: $f_{pclk} = 48M$, baud rate set to 115200, then $UARTDIV = 48000000/115200 = 416.6$, the

DP32G030
Reference Manual

417 can be selected based on rounding.

Automatic baud rate detection

The auto baud rate detection function automatically measures the data input from the `uart_rx` pin to calculate the baud rate. When the automatic baud rate measurement is finished, the measurement result is put into the `UART_BAUD` register.

In automatic baud rate detection, the `uart_rx` data will be detected for the time from the beginning of the falling edge of the start bit to the end of the first rising edge, which can be determined by `ABRDBIT`.

The automatic baud rate detection function can be enabled by configuring `ABRDEN`. In the initial stage, `uart_rx` is kept as 1. When a falling edge is detected, i.e. the start bit is received, the auto baud rate counter is activated and starts counting, and the auto baud rate counter will stop counting when the first rising edge is detected. Then, the result of dividing the auto baud rate counter value by `ABRDBIT` is automatically stored in the `UART_BAUD` register, and `ABRDEN` is cleared to zero, and the auto baud rate detection end flag is generated.

The automatic baud rate calculation is illustrated below:

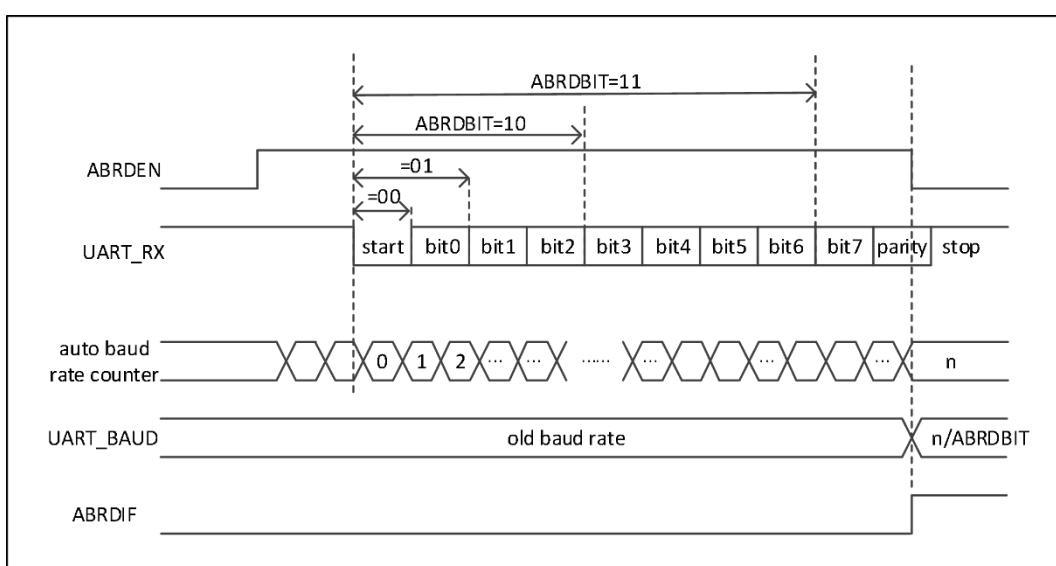


Figure 5-119 UART Automatic Baud Rate Calculation Diagram

data sampling

The UART is an asynchronous communication interface and does not have an independent operation clock, so it is necessary to synchronize the received data. In order to correctly obtain the received character data, the UART samples the UART_RX with a clock that is 16 times the baud rate of the data, and there are 16 sampling clocks for each bit of data, and the data sampled in the middle of the 8th, 9th, and 10th samples are used as the actual data received.

The schematic of data sampling is shown below:

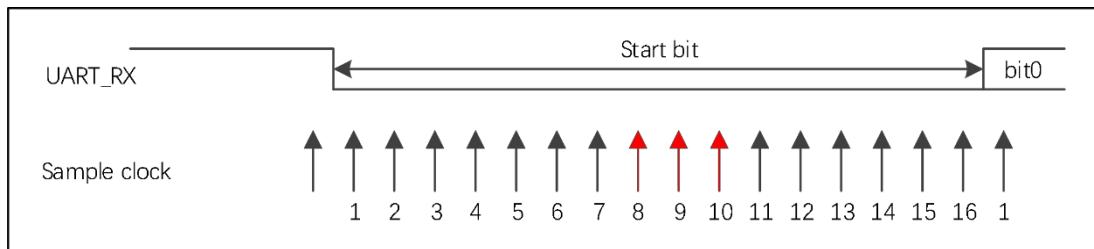


Figure 5-120 UART Data Sampling Schematic

For each bit of data, 16 samples are taken according to the baud rate setting, and the 8th, 9th and 10th samples are taken for judgment: if the three samples contain at least two zeros, the bit bit is 0; if the three samples contain at least two ones, the bit is 1.

Hardware automatic flow control

The UART controls the asynchronous serial data flow between the two devices via the CTS input and RTS output. As shown in the figure below

Show:

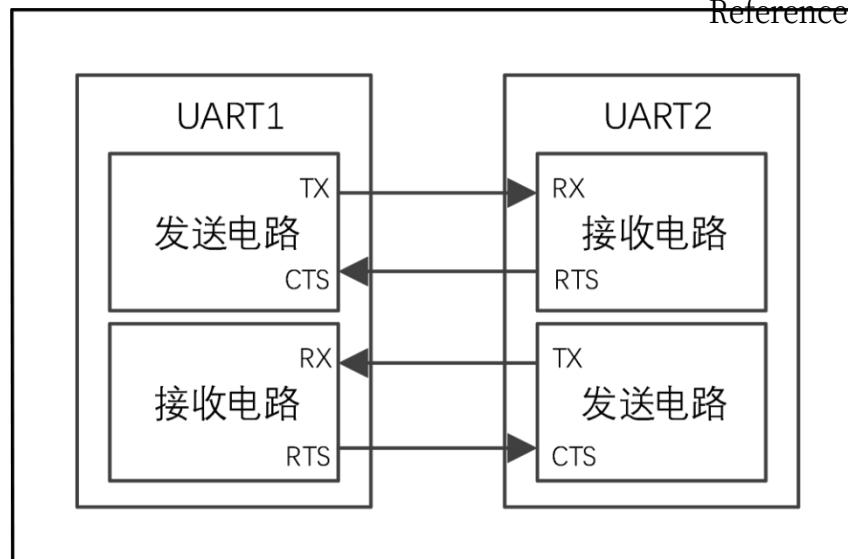


Figure 5-121 UART Auto Flow Control Connection Diagram

RTS flow control:

When RTSEN is configured to 1, RTS outputs an active level (level is configurable) when there is room in the data receiving FIFO. When the data receiving FIFO is full, RTS outputs an invalid level to indicate that data transfer is desired to stop at the end of the current set of data.

An example of starting RTS flow control communication is shown below:

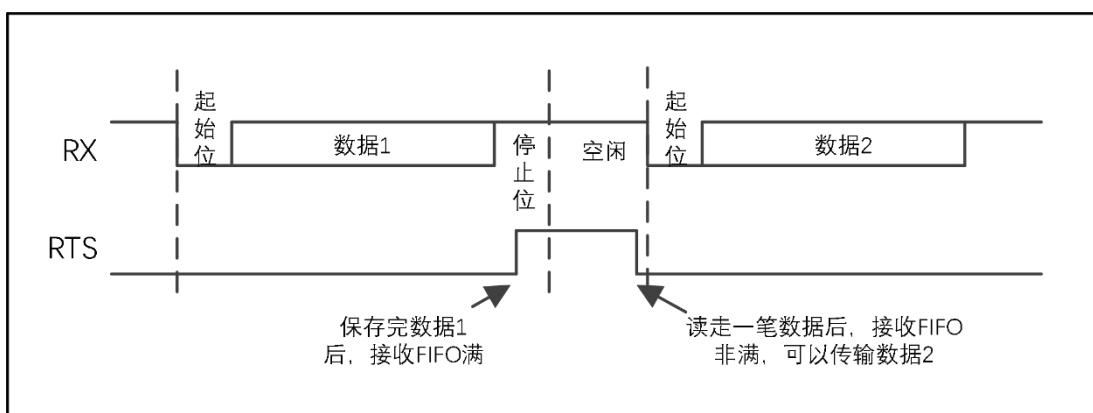


Figure 5-122 UART Flow Control Receive Schematic

CTS Flow Control:

When CTSEN is configured to 1, the transmitter circuit checks the input CTS level before sending the next set of data. If CTS is valid (level is configurable) the next data set in the transmit FIFO can be sent, otherwise a data set cannot be sent. If the CTS becomes invalid during data sending, the next set of data is stopped after the current transmission is completed.

An example of starting CTS flow control communication is shown below:

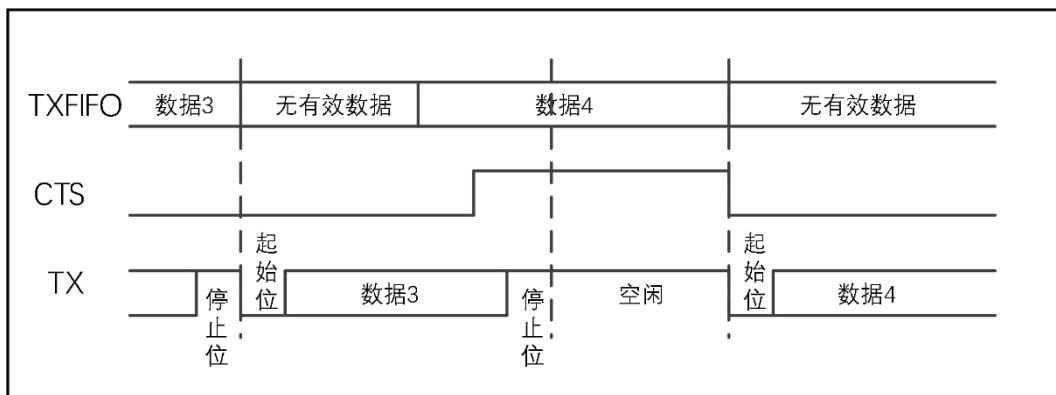
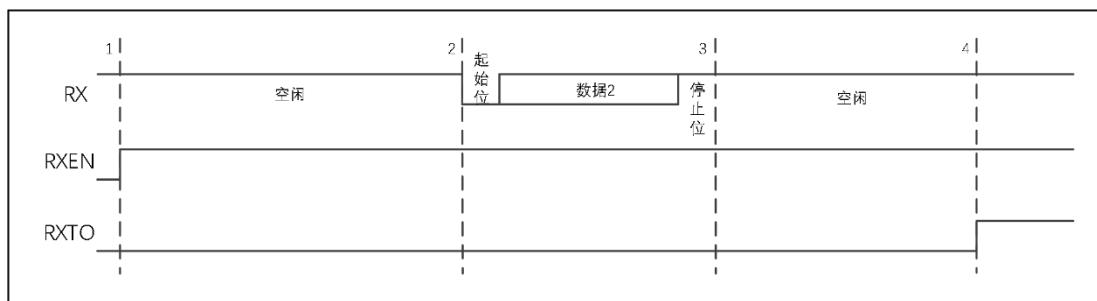


Figure 5-123 UART Flow Control Transmission Schematic

Receive Timeout Description

Two situations generate a receive timeout:

1. When the receive enable is active, the timeout interrupt will not take effect until at least one byte of data is received (when the timeout interrupt enable is turned on). When the time exceeds the configured timeout time, a timeout interrupt is generated, which can be handled by the user in the interrupt service function to clear the timeout interrupt flag.



DP32G030
Reference Manual

Figure 5-124 UART First Receive Timeout Interrupt

As shown above:

The UART receive enable is turned on at 1 moment, and the UART receive is idle until 2 moments, and no timeout interrupt will be generated in this interval; the UART receives the first data at 2 moments, and the data is received until 3 moments, after which it enters into the idle time, and then the timeout counter will start counting, and a timeout interrupt will be generated until 4 moments.

2. When a receive timeout interrupt has occurred, the timeout interrupt can take effect again only after at least one byte of data has been received again (when the timeout interrupt enable is turned on)

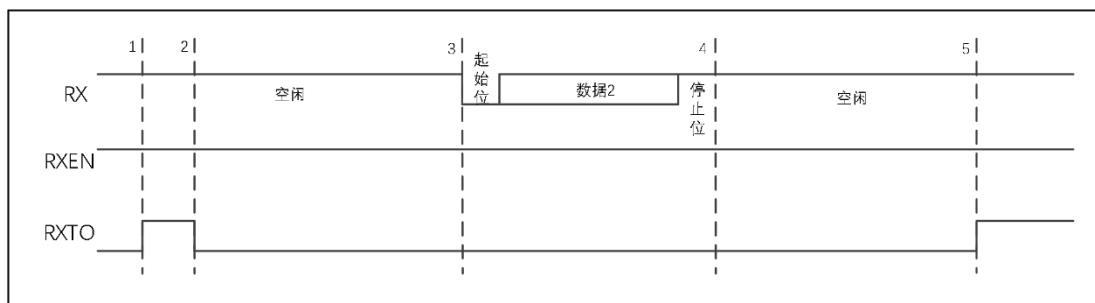


Figure 5-125 UART Second Receive Timeout Interrupt

As shown above:

The UART generates RXTO receive timeout interrupt at time 1 and is cleared by the CPU at time 2. The UART receives idle until time 3, but no timeout interrupt will be generated in this interval; the UART receives the first data at time 3 until the data is received at time 4, and then enters into idle at this time when the timeout counter starts counting until time 5 when a timeout interrupt will be generated.

disruptions

The UART provides 8 types of interrupt sources and their relationship is shown below:

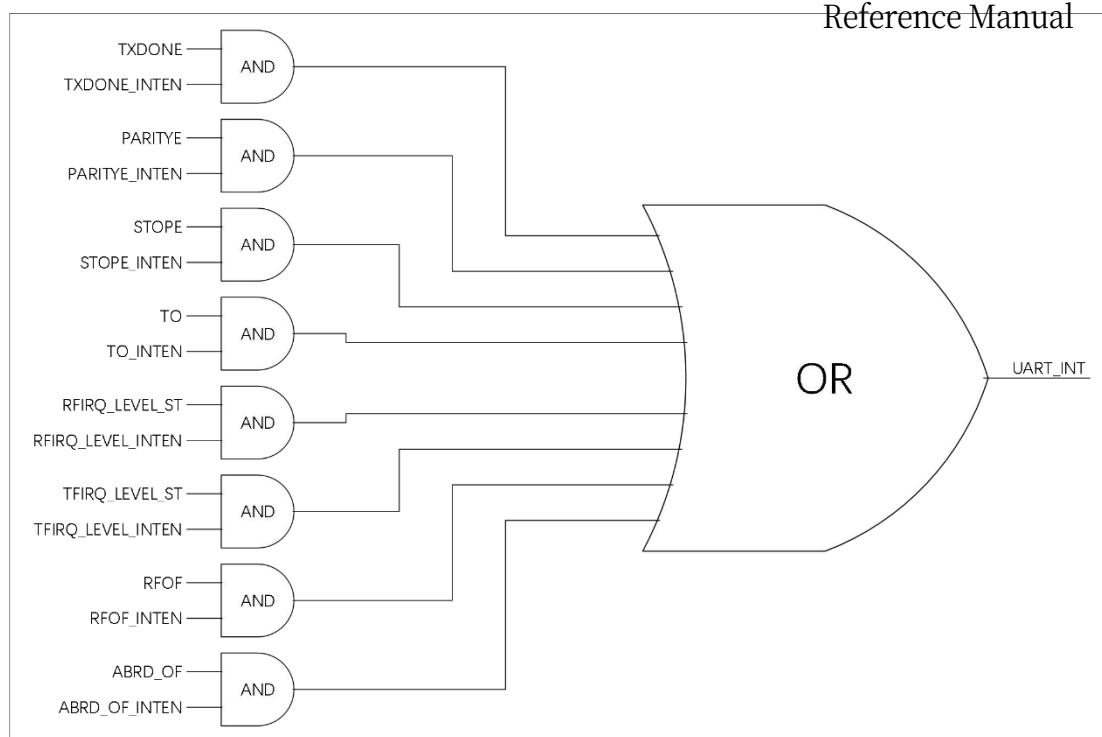


Figure 5-126 UART Interrupt Flag and Interrupt Diagram

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
UART0: BASE: 0x4006B000					
UART1: BASE: 0x4006B800					
UART2: BASE: 0x4006C000					
UART_CTRL	0x00	32	R/W	0x00	UART Control Register
UART_BAUD	0x04	32	R/W	0x00	UART Baud Rate Configuration Register
UART_TDR	0x08	32	W	0x00	UART Write Data Register
UART_RDR	0x0c	32	R	0x00	UART Read Data Register
UART_IE	0x10	32	R/W	0x00	UART Interrupt Enable Register
UART_IF	0x14	32	R/W	0x2400	UART Interrupt Status Register
UART_FIFO	0x18	32	R/W	0x07	UART FIFO Control Register
UART_FC	0x1c	32	R/W	0x00	UART Flow Control Configuration Register

UART_RXTO	0x20	32	R/W	0xff	UART Receiver Timeout Configuration Register
-----------	------	----	-----	------	---

register description

UART_CTRL register (0x00)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:17	RESERVED	R	0	reserved bit
16:14	TX_DLY	R/W	0	<p>Transmit delay time setting (used to set the transmission delay time between the last stop bit and the next start bit)</p> <p>000: No delay</p> <p>001: 1bit</p> <p>delay 010:</p> <p>2bit delay</p> <p>.....</p> <p>111: 7bit delay</p>
13:12	ABRDBIT	R/W	0	<p>Automatic baud rate detection bit length configuration</p> <p>11: 8-bit duration from the start bit to the first rising edge. Input data is 0x80.</p> <p>10: 4-bit duration from the start bit to the first rising edge. The input data is 0x08.</p> <p>01: 2-bit duration from the start bit to the first rising edge. Input data is 0x02.</p> <p>00: 1-bit duration from the start bit to the first rising edge. The input data is 0x01.</p>
11	ABRDEN	R/W	0	<p>Auto baud rate detection enable</p> <p>1: Automatic baud rate detection function enabled</p> <p>0: Automatic baud rate detection function disabled</p> <p>This bit will be automatically cleared after the auto-detect is finished.</p>

10:9	RESERVED	R	0	reserved bit
------	----------	---	---	--------------

				Parity mode selection 11: Standing 0 10: Standing 1 01: Even Check 00: odd calibration	Reference Manual
8:7	PARMD	R/W	0	Parity bit enable 1: with parity bit 0: without parity bit	
6	PAREN	R/W	0	9bit data mode enable 1: 9bit data mode 0: 8bit data mode	
5	NINEBIT	R/W	0	Send DMA Transmit Enable 1: Indicates the transmit data register for DMA operation of the UART. 0: Indicates that the CPU operates the transmit data register of the UART.	
4	TXDMAEN	R/W	0	Receive DMA Transmit Enable 1: Indicates the receive data register for DMA operation of the UART. 0: Indicates that the CPU operates the receive data register of the UART.	
3	RXDMAEN	R/W	0	transmit enable bit 1: Send open. Sends the data stored in tx_fifo through uart_tx. 0: Transmit off. No data is sent. uart_tx signal is held at 1.	
2	TXEN	R/W	0	receive enable bit (computing) 1: Receive open. External data can be received via uart_rx 0: Receive off. Do not receive data from uart_rx	
1	RXEN	R/W	0	UART Enable Bit 1: Enable UART Module 0: Disable UART module	
0	UARTEN	R/W	0		

UART_BAUD register (0x04)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:16	RESERVED	R	0	reserved bit
15:0	BAUD	R/W	0	Baud rate configuration data

UART_TDR register (0x08)

bitfield ld (mat h.)	name (of a thing)	typology	reset value	descriptive
31:9	RESERVED	R	0	reserved bit
8:0	TDR	W	0	Write Data Register The data to be sent is written to the transmit FIFO through this register. When the data is ready to be sent, the shift register reads the transmit FIFO directly to send the data.

UART_RDR register (0x0C)

bitfield ld (mat h.)	name (of a thing)	typology	reset value	descriptive
31:9	RESERVED	R	0	reservations

8:0	RDR	R	0	<p>Read Data Register Reference Manual</p> <p>The received data is read out through this register. The shift register stores data into the receive FIFO after each transmission is completed, and the data in the receive FIFO is read out through this register.</p> <p>Note: If a parity bit error or a stop bit error occurs, the set of data will not be written to the RXFIFO. If the RXFIFO is not empty, the data in the RXFIFO is valid data.</p>
-----	-----	---	---	---

UART_IE register (0x10)

bitfield d (math .)	name (of a thing)	typology	reset value	descriptive
31:10	RESERVED	R	0	reserved bit
9	ABRD_OVF	R/W	0	Auto baud rate detection function Counter overflow interrupt enable
8	RXFIFO_OVF	R/W	0	Receive FIFO overflow interrupt enable
7	TXFIFO	R/W	0	Send the data stored in FIFO to reach the set water level Interrupt Enable
6	RXFIFO	R/W	0	Receive data received in FIFO reaches the set water level Interrupt enable
5	RXTO	R/W	0	Receive timeout interrupt enable
4	STOPE	R/W	0	Stop bit error interrupt enable for receive data
3	PARITYE	R/W	0	Receive data with parity error interrupt enable
2	TXDONE	R/W	0	All data sent interrupt enable (indicates that the data in the send shift register has been sent and there is no more data to be sent in

1	RESERVED	R/W	0	reserved bit
0	RESERVED	R/W	0	reserved bit

UART_IF register (0x14)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:23	RESERVED	R	0	reserved bit
22:20	TF_LEVEL	R	0	Send FIFO water level flag signal 000: If the FIFO is not full, it means there is 0 data in the FIFO, and if it is full, it means The FIFO has 8 data; 001: Indicates that the FIFO has 1 data; 111: Indicates that the FIFO has 7 data;

Reference Manual				
19:17	RF_LEVEL	R	0	Receive FIFO water level flag signal 000: If the FIFO is not full, it means there is 0 data in the FIFO, and if it is full, it means The FIFO has 8 data; 001: Indicates that the FIFO has 1 data; 111: Indicates that the FIFO has 7 data;
16	TXBUSY	R	0	Data transmission busy flag 1: Transmit FIFO is not empty, or there is data being sent. 0: Transmit FIFO is empty and there is no data being sent.
15	TXFIFO_HFULL	R	0	Send FIFO half full flag
14	TXFIFO_FULL	R	0	Send FIFO full flag
13	TXFIFO_EMPTY	R	0x1	Send FIFO Empty Flag
12	RXFIFO_HFULL	R	0	Receive FIFO half-full flag
11	RXFIFO_FULL	R	0	Receive FIFO full flag
10	RXFIFO_EMPTY	R	0x1	Receive FIFO empty flag
9	ABRD_OVF	R/W	0	Automatic baud rate detection function Counter overflow flag 1: Counter overflow, detection failure 0: Counter not overflowed write 1 Clear
8	RXFIFO_OVF	R/W	0	Receive FIFO Overflow Flag Write 1 Clear Note: Data after overflow will be discarded
7	TXFIFO	R	0	When the data stored in the transmit FIFO reaches the set water level, the bit is 1, otherwise 0
6	RXFIFO	R	0	When the data received in the receive FIFO reaches the set water level, the bit 1, otherwise 0

5	RXTO	R/W	0	Receive Timeout Flag 1: Send Receive Timeout Write 1 Clear
---	------	-----	---	--

4	STOPE	R/W	0	Stop bit error in receive data Write 1 Clear The main point is that the stop bit is not received or recognized within the expected time period, then it is considered a stop bit error	Reference Manual
3	PARITYE	R/W	0	Parity error in received data 1 clear	
2	TXDONE	R/W	0	All data transmission is complete (indicates that data transmission in the transmit shift register is complete and there is no pending data in the transmit data FIFO) Write 1 to clear the data.	
1	RESERVED	R/W	0	reserved bit	
0	RESERVED	R/W	0	reserved bit	

UART_FIFO register (0x18)

bitfield ld (mat h.)	name (of a thing)	typology	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7	TF_CLR	R/W	0	TXFIFO clear enable Software Write 1 Clear Transmit FIFO, Hardware Auto Clear
6	RF_CLR	R/W	0	RXFIFO clear enable Software Write 1 Clear Receive FIFO, Hardware Auto Clear

5:3	TF_LEVEL	R/W	0	Water level setting for TXFIFO generating interrupts 000: 0 001: 1 010: 2 111: 7 Indicates that the number of data to be sent in the sending FIFO is not more than the water level setting. For example, if the value 011 is set, the corresponding signal is generated when the number of written data in TXFIFO is less than or equal to 3.
-----	----------	-----	---	---

2:0	RF_LEVEL	R/W	0x7	<p>Water level setting for RXFIFO generating interrupts</p> <p>000:1 001: 2 010:3 111:8</p> <p>Indicates that the number of received data in the receive FIFO has reached at least the water level setting.</p> <p>For example, if the value 011 is set, the corresponding signal is generated when there are at least 4 data in the RXFIFO.</p>
-----	----------	-----	-----	--

UART_FC register (0x1C)

bitfield ld (mat h.)	name (of a thing)	typology	reset value	descriptive
31:6	RESERVED	R	0	reserved bit
5	RTS_SIGNAL	R	0	<p>Indicates RTS status on the line</p> <p>1: RTS is high 0: RTS is low</p>
4	CTS_SIGNAL	R	0	<p>Indicates online CTS status</p> <p>1: CTS is high 0: CTS is low</p>
3	RTSPOL	R/W	0	<p>RTS Signal Polarity Configuration</p> <p>1: When RTS signal is high, UART can receive data; when RTS signal is low, UART receive FIFO is full and can not receive data.</p> <p>0: When RTS signal output is low, UART can receive data; when RTS signal output is high, UART receive FIFO is full and no more data can be received.</p>

				CTS Signal Polarity Configuration
2	CTSPOL	R/W	0	1: When the CTS signal input is high, the UART can send data; when the CTS signal input is low, the UART does not send data. 0: When CTS signal input is low, UART can send data; when CTS signal input is high, UART does not send data.

Reference Manual				
1	RTSEN	R/W	0	RTS flow control enable 1: RTS signaling plays a flow control role 0: RTS signal does not work
0	CTSEN	R/W	0	CTS flow control enable 1: CTS signaling plays a flow control role 0: CTS signal does not work

UART_RXTO register (0x20)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	RXTO	R/W	0xff	Receive Data Timeout Trigger Comparison Value When the RXFIFO receives a new piece of data, the timer is cleared and timing restarts (the timing clock is the duration of a piece of data) A receive timeout flag is generated if a valid start bit has not been received by the time the timer has timed out past this configured value.

5.17 SPI Bus Controller (SPI)

5.17.1 summarize

Serial Peripheral Interface (SPI) is a serial synchronous communication means for external devices to exchange 8-bit data over 2 wires. The chip provides an SPI interface module, which can be configured as a master or a slave device to realize SPI communication with the outside.

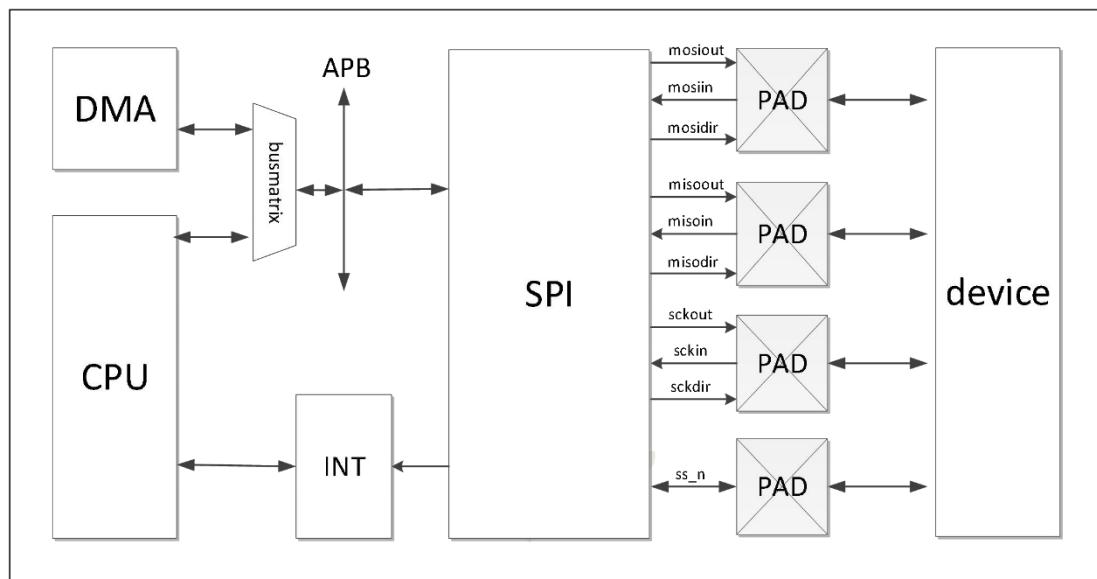


Figure 5-127 SPI Module System Block Diagram

The above diagram shows the system connection of SPI module. The CPU can directly control the SPI module through the APB bus interface, and the SPI module realizes the communication between the chip and external devices through PADs. Depending on the configured mode, the SPI module can configure the direction of each PAD.

5.17.2 characterization

- Supports host mode and slave mode
- Programmable clock polarity and phase
- Master mode rate assignable up to 4 divisions of the system clock

- Configurable data transfer sequence

- End of transmission interrupt flag
- Separate read data registers and write data registers
- 8-level FIFO caching mechanism for receive and transmit respectively
- With DMA transfer interface

5.17.3 Block Diagram of Module Structure

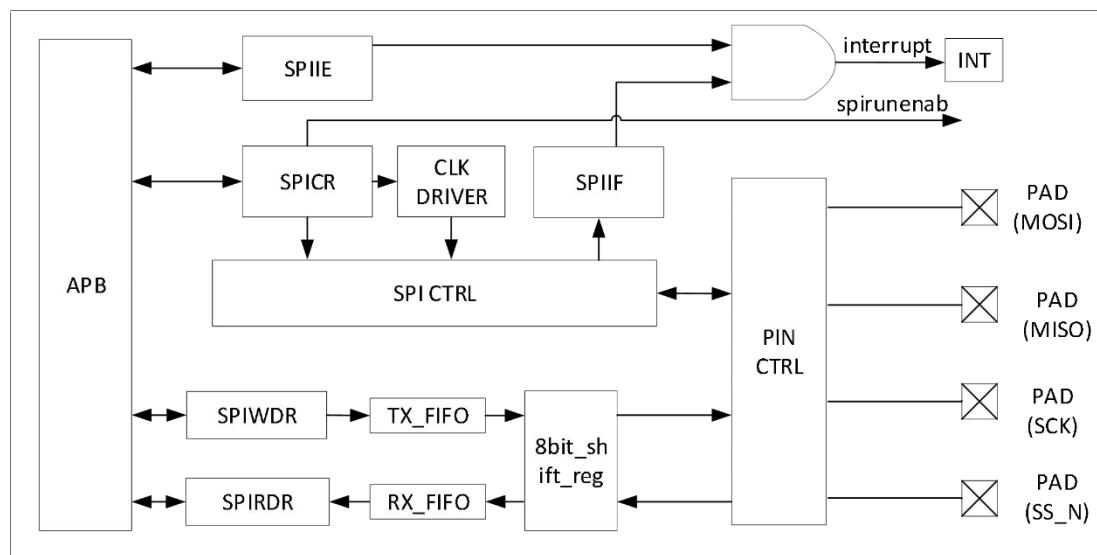


Figure 5-128 SPI Module Block Diagram

As shown in the above figure, it is the structure schematic of the SPI module, and the SPI control register SPICR can be used to configure the module's master-slave mode, transmission mode, FIFO clearing and other functions. Data writing is completed through the operation of SPIWDR, when preparing to send data, SPI will write the data to be sent into the send FIFO through the SPIWDR register, and the shift register will read the value of FIFO to send the data; data reading is completed through the operation of SPIRDR, the shift register will store the data into the receive FIFO after each transmission is completed, and read the receive FIFO through the SPIRDR register. The reading of data is accomplished by the operation of SPIRDR. The shift register stores the data into the receiving FIFO after each transmission, and the data in the receiving FIFO is read by the SPIRDR register. The interrupt signal of the module is controlled by the interrupt enable

register SPIIE, turn on the interrupt enable, and the interrupt will be generated when there is a corresponding interrupt state.

5.17.4 Function

n

Description

SPI Interface

Timing

For different SPI peripherals, the timing of the SPI serial clock can be generated in 4 different combinations by the clock phase select bit (**SPICR.CPHA**) and clock polarity select bit (**SPICR.CPOL**) settings. The timing configuration of the master and slave devices must be consistent in order to ensure correct data transfer.

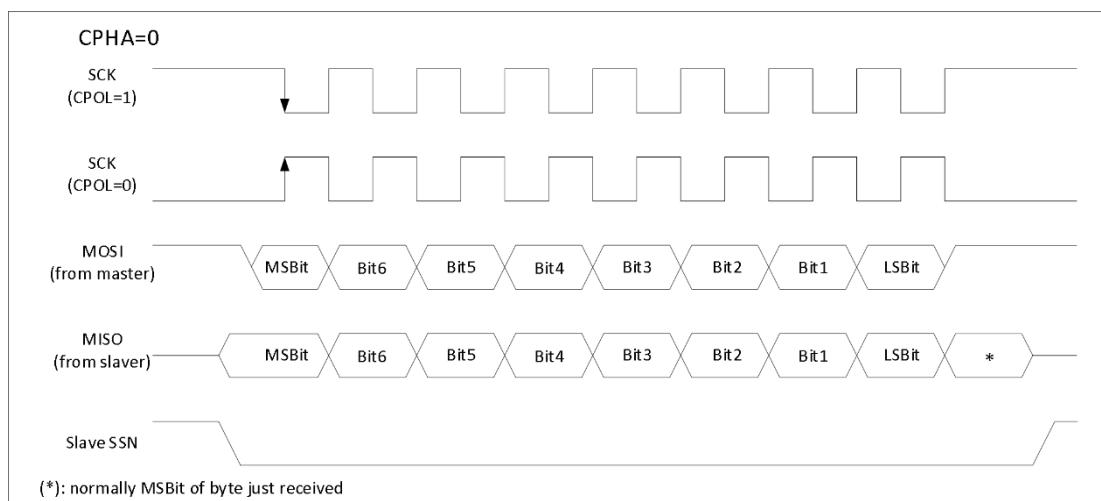
There is no serial clock output from the SCK pin of the SPI when the device mode or the SPI System Enable Bit (**SPICR.SPE**) bit is zero.

1: When CPHA=0, the SPI module samples data at the first

hopping edge of the serial clock, i.e., if CPOL=1, it samples

data at the falling edge of the serial clock;

If CPOL=0, sample the data on the rising edge of the serial clock. As shown in the figure below:



(CPHA=0) 2: When CPHA=1, the SPI module samples data at the second hopping edge of the serial clock, i.e., if CPOL=1, it samples data at the rising edge of the serial clock; If CPOL=0, sample the data on the falling edge of the serial clock. As shown in the figure below:

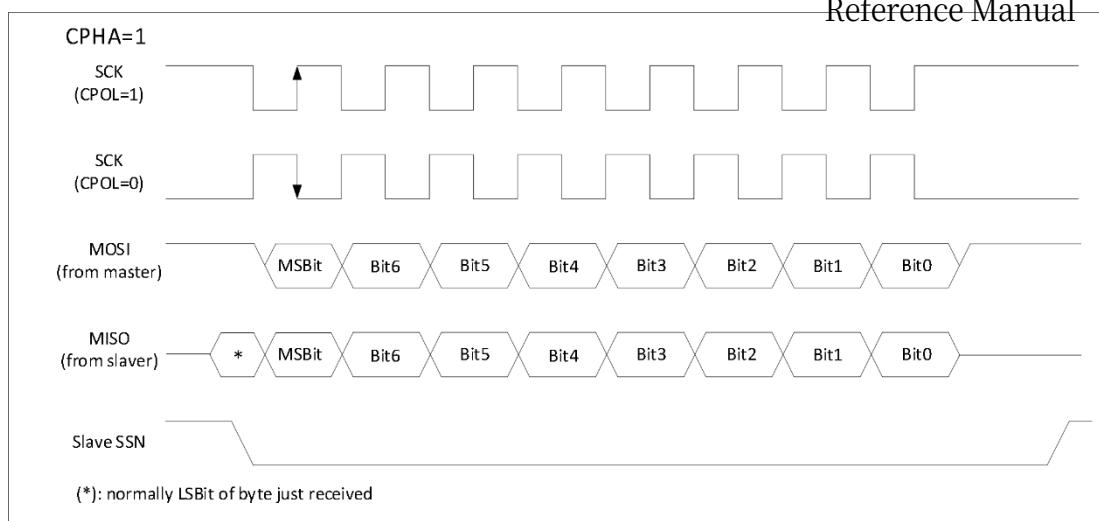


Figure 5-130 SPI Data/Clock Timing Diagram (CPHA=1)

3: Slave SSN

If the SPI is a slave device, the SSN pin of the slave device can remain low for continuous data transfers. This is shown in the following figure:

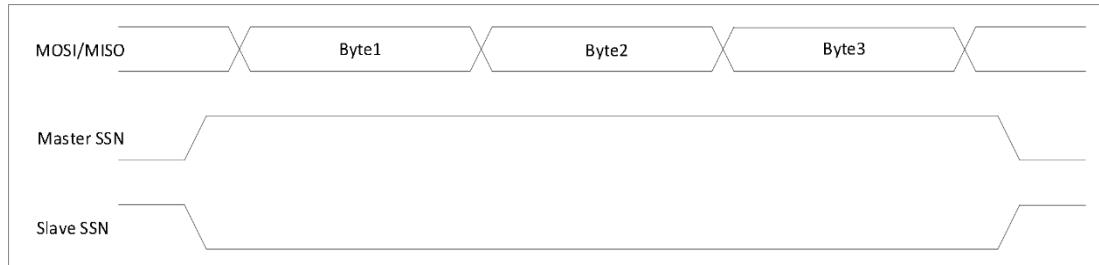


Figure 5-131 SPI SSN Timing Diagram (CPHA=0)

I/O Configuration

Master Output, Slave Input (MOSI)

The Master Out Slave In (MOSI) pin is the output of the master device and the input of the slave device and is used for serial data transfer from the master to the slave device. This pin is an output when SPI is configured as a master device and an input when SPI is configured as a slave device.

Master Input, Slave Output (MISO)

The Master In Slave Out (MISO) pin is an output from the slave device and an input from the master device, and is used for serial data transfer from the slave device to the master device. This pin is an input when SPI is configured as a master device and an output when SPI is configured as a slave device.

Serial Clock (SCK)

The Serial Clock (SCK) pin is an output from the master device and an input from the slave device, and is used to synchronize serial data transfers between the master and slave devices on the MOSI and MISO lines. This pin outputs the clock when SPI is configured as a master device and is an input when SPI is configured as a slave device.

Selection from (SSN)

The Slave Select (SSN) pin is used to control slave device selection. When SPI is configured as a master device, the SSN pin can be used to control whether the slave device is selected or not by means of registers, and when SPI is configured as a slave device, the SSN pin originates from the control signal of the master device.

The SPI master-slave device connections are shown below:

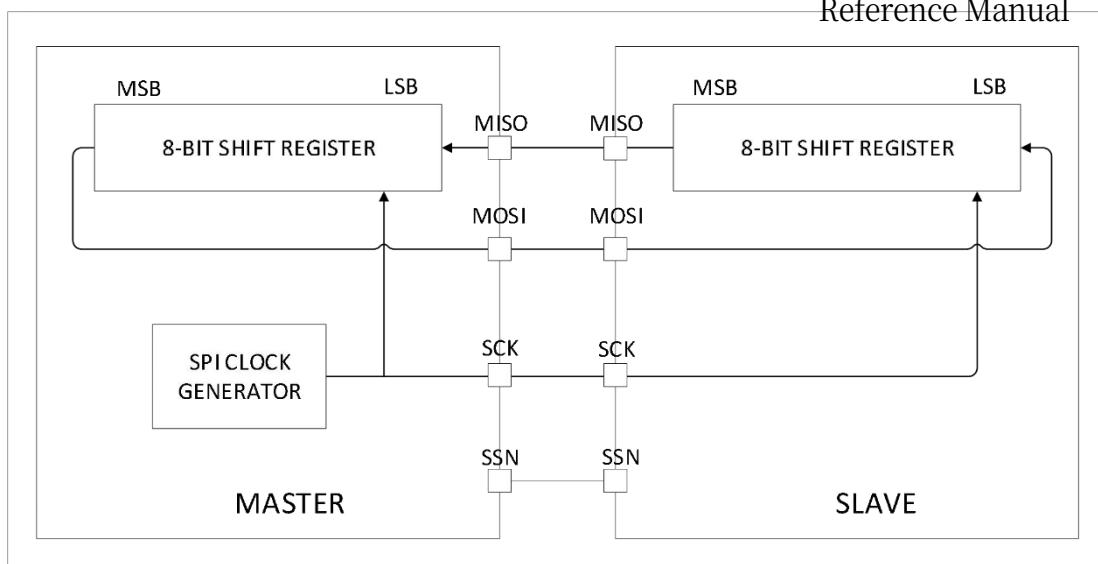


Figure 5-132 SPI Master/SPI Slave Interconnection

The **MOSI**, **MISO**, **SCK** and **SSN** of the master and slave devices are connected together. The master and slave devices are connected into a loop through **MOSI** and **MISO**, the master device outputs the clock, and during data transfer, the master device outputs the data through **MOSI** and the slave device outputs the data through **MISO**. When one byte of data has been transferred, the master and slave will exchange 8-bit shift register values.

Data Transfer Configuration

- 1: Configure the **SPICR.SPE** and **SPICR.MSTR** bits to enable SPI and set the master-slave mode before data transfer.
Style.
 - 2: Configure the **SPICR.CPHA** bit and **SPICR.CPOL** bit to set the serial clock phase and polarity (master and slave devices need to be identical)
 - 3: Configure the **SPICR.SPR[2:0]** bits to set the serial clock baud rate (not necessary if in slave mode, the serial clock rate is determined by the master device)
 - 4: Configure the **SPICR.LSB** bit to set the transmission order and

SPICR.CPHA_DATAHOLD_S to set the number of transmission data Reference Manual slave mode.

Configure interrupts and configure the SPIIE and SPIIR bits when needed.

The SSN pin of the slave device needs to be pulled low before data transfer in master device mode. MCU Write SPIWDR in Master Mode

The data transfer is initiated by the action of the register and completed by the setting of the interrupt flag SPIIR.SPIF.

Slave mode handling is more specific, when CPHA=0, the SSN pin of the slave device pulls low to start the data transfer, and the SSN pin of the slave device pulls high to end the data transfer (even if the SPIIR.SPIF interrupt has been generated before that) because the slave device doesn't know when the transfer is going to start, and when the SSN pin pulls low, the MISO pin starts the data MSB transfer immediately.

When CPHA=1, the slave device starts the data transfer at the first edge of the serial clock and ends the data transfer after SPIIR.SPIF is set.

disruption generation

The SPI module provides 5 types of interrupt sources, as shown in the following diagram.

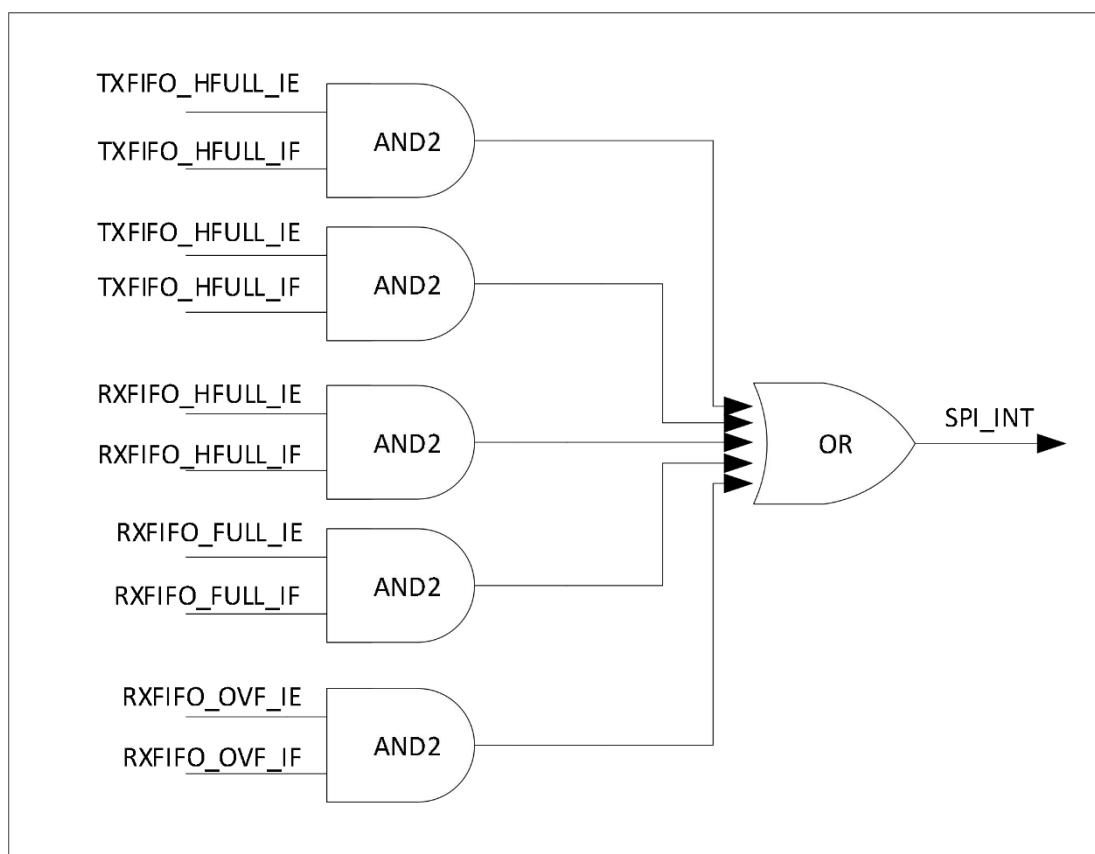


Figure 5-133 SPI Interrupt Flag and Interrupt Diagram

Receive FIFO full interrupt

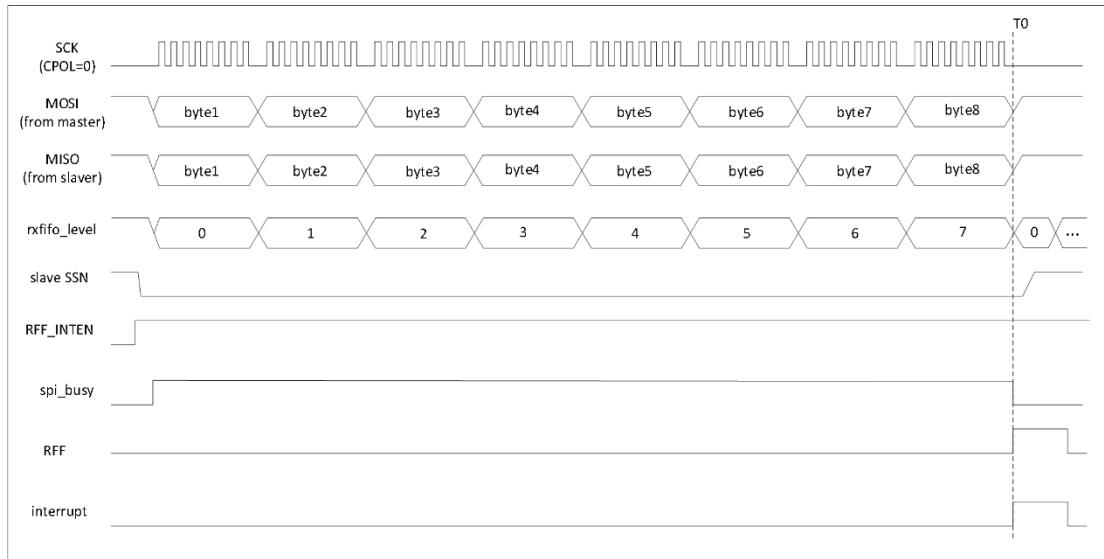
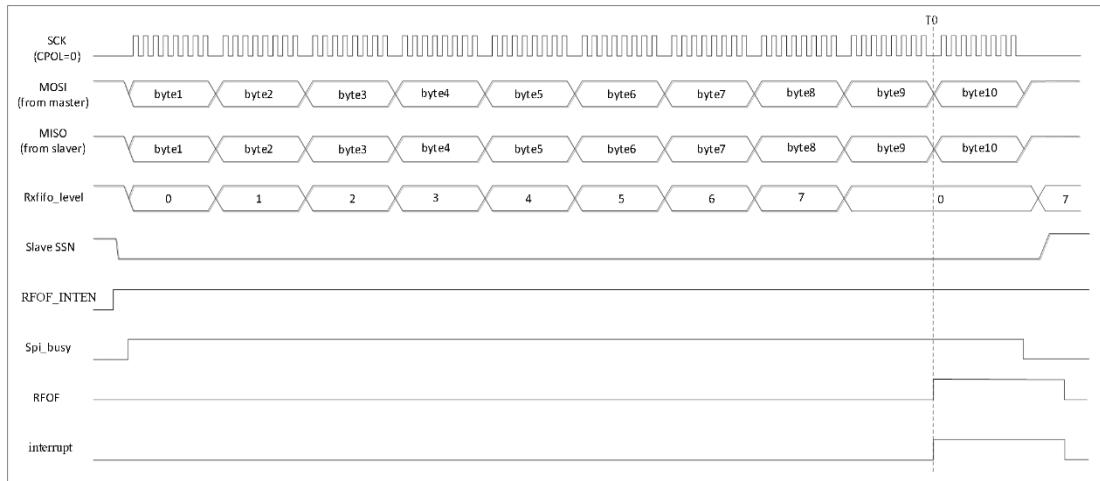


Figure 5-134 SPI Receive FIFO Full Interrupt

During data transmission, the received data will be saved into the receive FIFO through the shift register, enable the receive FIFO full interrupt enable register **RFF_INTEN**, the FIFO depth of this module is 8, so when the data in the FIFO reaches 8, the receive FIFO full status signal **RFF** will be set to 1, that is, the **T0** time in the above figure, and generate an interrupt signal. It should be noted that when the data in the FIFO is not read away and the FIFO is always full, the receive full status signal will be kept high, even if the status is cleared to zero, it will be set to 1 again, and the full status will be cleared only when the data in the FIFO is read away and the FIFO is not full, and the SPI transfer busy flag will be kept high during the SPI data transfer.

Receive FIFO overflow interrupt



During the data transmission process, the received data will be saved into the receive FIFO through the shift register, enable the receive FIFO overflow interrupt enable register RFOF_INTEN, the depth of the FIFO of this module is 8, so when the data in the FIFO reaches 8, continue to transmit the data to the receive FIFO, the receive FIFO overflow status signal RFOF will be set to 1 and generate an interrupt signal in the T0 time. The receive FIFO overflow status signal RFOF will be set to 1 at T0 and an interrupt signal will be generated. It should be noted that when the data in the FIFO is not read away and the FIFO is always full, continue to transfer data to the FIFO, the data will not be written and will be discarded, the receive FIFO overflow status signal will be kept high even after the status is cleared, it will be set to 1 again, and during the SPI data transfer process, the SPI transfer busy flag will be kept high.

Receive FIFO half full interrupt

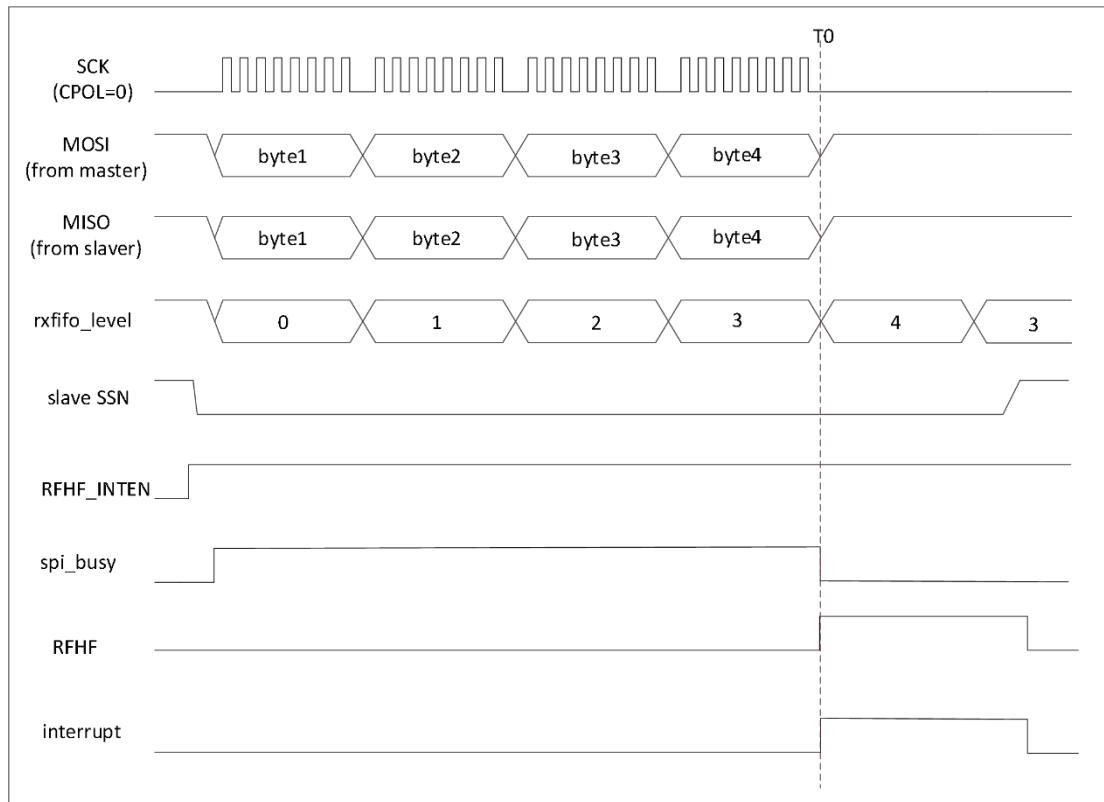


Figure 5-135 SPI Receive FIFO Half Full Interrupt

During data transmission, the received data will be saved into the receive FIFO through the shift register, enable the receive FIFO half-full interrupt enable register **RFHF_INTEN**, the FIFO depth of this module is 8, so when the data in the FIFO reaches 4, the receive FIFO half-full status signal **RFF** will be set to 1, i.e., T_0 time in the figure above, and generate an interrupt signal. It should be noted that when the number of data in the FIFO is greater than or equal to 4, the receive half-full status signal will remain high, even if the status is cleared, it will be set to 1 again, only when the data in the FIFO is read away, the data in the FIFO is less than half-full, the half-full status will be cleared, and in the process of SPI data transfer, the SPI transfer busy flag will remain high.

Send FIFO half full interrupt

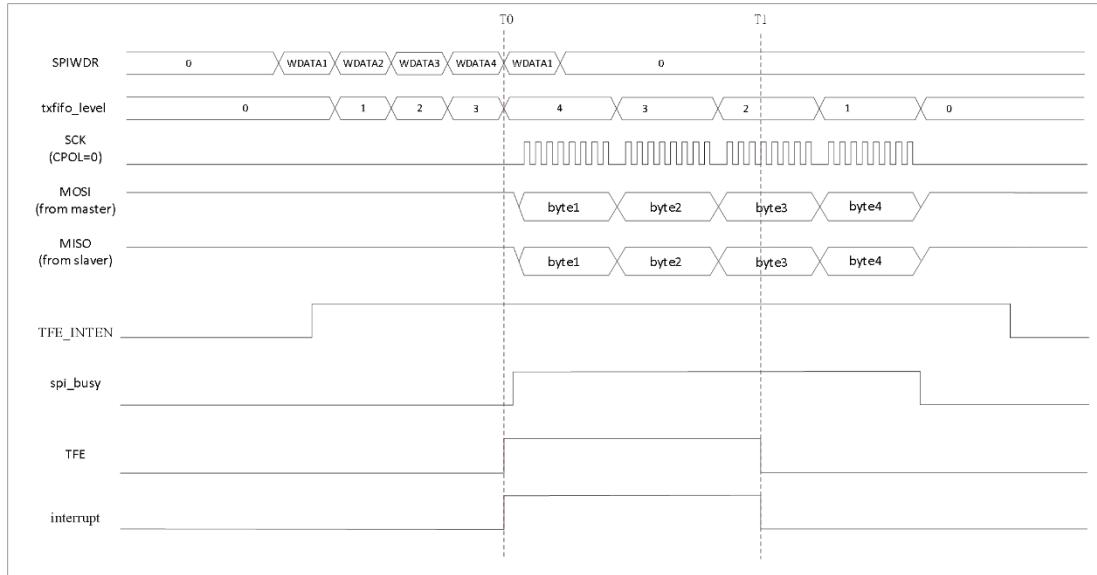


Figure 5-136 SPI Send FIFO Half Full Interrupt

During data transmission, the data sent by this module will be saved in the transmit FIFO through the SPIWDR register, enable the transmit FIFO half-full interrupt enable register TFE_INTEN, the depth of the FIFO of this module is 8, so when the amount of data in the transmit FIFO reaches 4, the transmit FIFO half-full status signal TFE will be set to 1, i.e., T0 in the above figure and generate an interrupt signal. An interrupt signal is generated. It should be noted that when the data in the transmit FIFO is not read away and the FIFO is always greater than or equal to half-full, the transmit full status signal will be kept high, even if it is set to 1 again after the status is cleared to zero, only when the data in the FIFO is read away and the FIFO is less than half-full, the full status is cleared, such as the T1 time in the figure, and during the SPI data transmission, the SPI Busy flag will remain high during the SPI data transfer.

Send FIFO Air Break

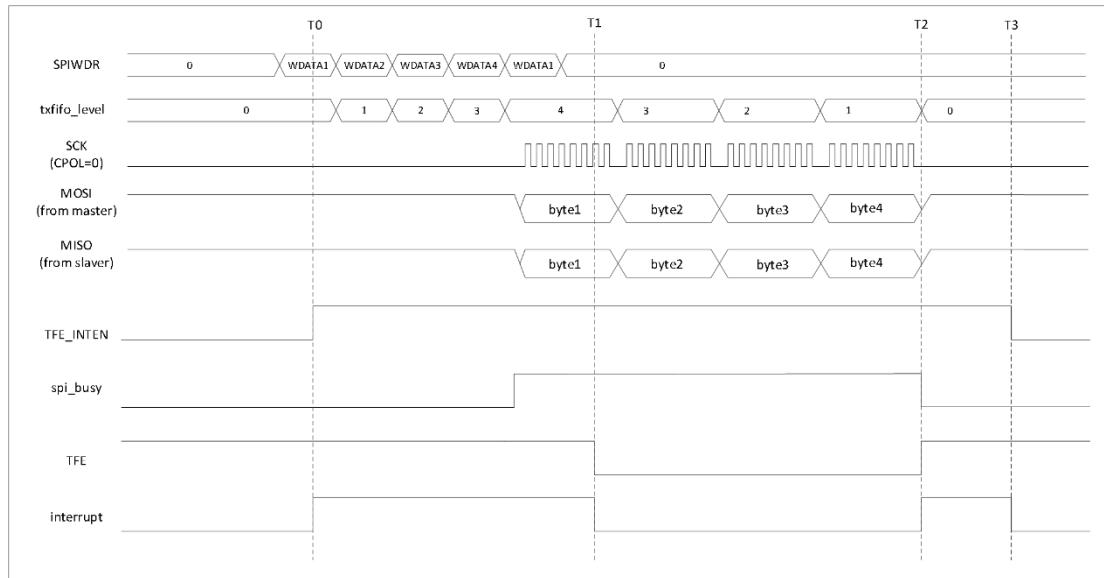


Figure 5-137 SPI Transmit FIFO Air Break

In the process of data transmission, the data sent by this module will be saved to the sending FIFO through the SPIWDR register, and there is no data in the sending FIFO at the beginning of transmission, so the sending empty state signal TFE is 1, and the interrupt signal will be generated by enabling the half-full interrupt enable register TFE_INTEN of the sending FIFO in the T0 time. When data is written to the transmit FIFO and the empty state is cleared, the empty state and the transmit empty interrupt signal will be set to 0 at T1 time; at T2 time, when the data in the transmit FIFO is read empty again, the transmit empty state will be set to 1 again and an interrupt signal will be generated; at T3 time, turn off the transmit empty interrupt enable, and the interrupt signal will not be generated again. It should be noted that, when there is no data in the transmit FIFO, and the FIFO is always empty, the transmit full status signal will be kept high, and it will be set to 1 again even after the status is cleared, and only when there is data written in the FIFO, and the FIFO is in the non-empty state, the empty status will be cleared, and in the process of SPI data transmission, the SPI transmission busy flag will be kept high.

DMA Control

DMA can realize the control of this module through the DMA control enable registers TXDMAEN and RXDMAEN. When TXDMAEN is 1, it means DMA operates the transmit data register of SPI; when RXDMAEN is 1, it means DMA operates the receive data register of SPI.

workflow

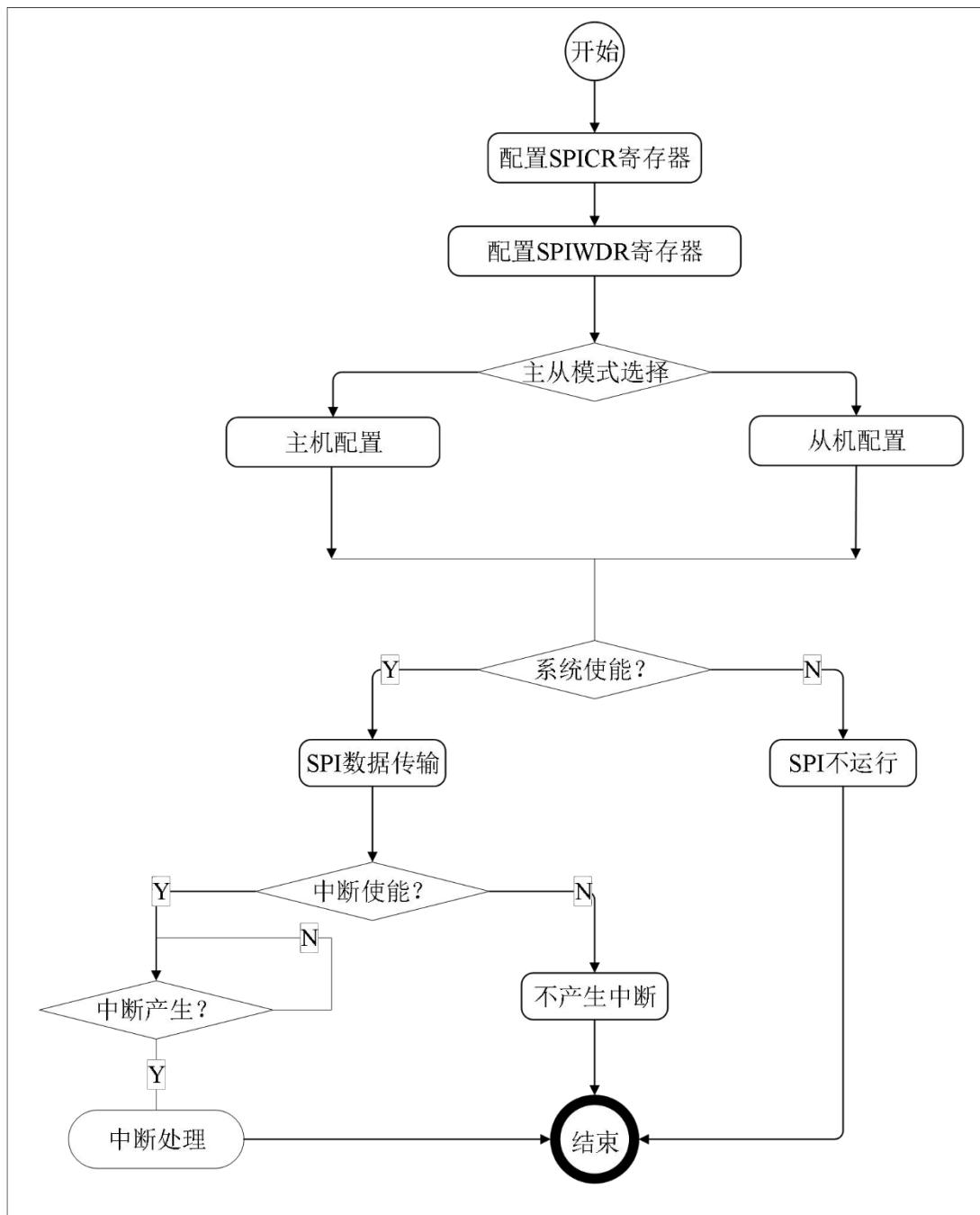


Figure 5-138 SPI Operation Flow

- Configuring the SPI Module Clock
- PORT port configured for SPI functionality

- Configuring SPI Master-Slave Mode
- Configuring the Phase and Polarity of the SPI Clock
- Configuring the transfer order
- Configure the SPI baud rate if it is the master mode.
- Configure Interrupt Enable
- Configure SPI system enable, turn on SPI, and start transmitting data

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
SPI0:	BASE: 0x400B8000				
SPI1:	BASE: 0x400B8800				
SPICR	0x00	32	R/W	0x1010	SPI Control Register
SPIWDR	0x04	32	R/W	0x00	SPI Write Data Register
SPIRDR	0x08	32	R	0x00	SPI Read Data Register
SPIIE	0x10	32	RW	0x00	SPI Interrupt Enable Register
SPIIF	0x14	32	R/W	0x8	SPI Interrupt Status Register
SPIFIFOST	0x18	32	R/W	0x9	SPI FIFO Status Register

register description

SPICR register (0x00)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:17	RESERVED	R	0	reserved bit
16	TF_CLR	R/W	0	Send FIFO Clear Bit Software Write 1 Clear Transmit FIFO, Hardware Auto-Clear
15	RF_CLR	R/W	0	Receive FIFO Clear Bit Software Write 1 Clear Receive FIFO, Hardware Auto Clear
14	TXDMAEN	R/W	0	Send DMA control enable bit 1: Indicates the transmit data register for DMA operation SPI. 0: Indicates that the CPU operates the SPI transmit data register.
13	RXDMAEN	R/W	0	Receive DMA control enable bit 1: Indicates the receive data register for DMA operation SPI. 0: Indicates that the CPU operates the receive data register of the SPI.
12	MSR_SSN	R/W	1	SSN output in master mode, default output 1 This register is only valid in master mode
11:8	CPHA_DATA HOLD_S	R/W	0	Data Hold Time Configuration Register when CPHA is 1 in Slave Mode 0000: 1 pclk 0001: 2 pclk 1111: 16 pclk
7	LSB	R/W	0	Data transfer sequence selection 0: MSB 1: LSB

6	MSTR	R/W	0	Master-Slave mode selection Reference Manual 0 = SPI system configured in slave mode 1 = SPI system configured as master device mode
5	CPOL	R/W	0	Clock Polarity Selection 0 = Serial clock idle state is low, active level is high 1 = Serial clock idle state is high, active level is low
4	CPHA	R/W	1	Clock phase selection 0 = sample data at the first edge of the serial clock jump 1 = Sample data on the second edge of the serial clock jump

Reference Manual				
3	SPE	R/W	0	SPI System Enable 0 = SPI system shutdown 1 = SPI system enable
2	SPR2	R/W	0	SPI baud rate select bit 2
1	SPR1	R/W	0	SPI baud rate select bit 1
0	SPR0	R/W	0	SPI baud rate select bit 0

SPR0, SPR1, SPR2 Indicates the baud rate selection:

SPR2	SPR1	SPR0	Fsck	Fsck(Fcpu=48MHz)
0	0	0	Fpclk/4	12MHz
0	0	1	Fpclk/8	6MHz
0	1	0	Fpclk/16	3MHz
0	1	1	Fpclk/32	1.5MHz
1	0	0	Fpclk/64	750KHz
1	0	1	Fpclk/128	375KHz
1	1	0	Fpclk/256	187.5KHz
1	1	1	Fpclk/512	93.75 KHz

SPIWDR register (0x04)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
7:0	SPIWDR	R/W	0	SPI writes the data to be sent to the transmit FIFO through the SPIWDR register. When the data is ready to be sent, the shift register reads the transmit FIFO directly to send the data.

SPIRDR register (0x08)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
7:0	SPIRDR	R	0	SPI reads the received data through the SPIRDR register. The shift register stores the data in the receive FIFO after each transmission is completed and reads the data in the receive FIFO through the SPIRDR register.

SPIIE register (0x10)

bitfie ld (mat h.)	name (of a thing)	typolog y	reset value	descriptive
31:5	RESERVED	R	0	reserved bit
4	TXFIFO_HFULL	R/W	0	Send FIFO half-full interrupt enable
3	TXFIFO_EMPTY	R/W	0	Transmit FIFO air-break enable
2	RXFIFO_HFULL	R/W	0	Receive FIFO half-full interrupt enable
1	RXFIFO_FULL	R/W	0	Receive FIFO full interrupt enable
0	RXFIFO_OVF	R/W	0	Receive FIFO overflow interrupt enable

SPIIF register (0x14)

bitfie ld (math.)	name (of a thing)	typol ogy	reset value	descriptive

)				
31:5	RESERVED	R	0	reserved bit
4	TXFIFO_HFULL	R/W	0	Send FIFO Half Full Flag Write 1 Clear
3	TXFIFO_EMPTY	R/W	1	Send FIFO Empty Flag Write 1 Clear
2	RXFIFO_HFULL	R/W	0	Receive FIFO half-full flag write 1 clear

1	RXFIFO_FULL	R/W	0	Receive FIFO full flag write 1 clear	Reference Manual
0	RXFIFO_OVF	R/W	0	Receive FIFO Overflow Flag Write 1 Clear Note: Data after overflow will be discarded.	

SPIFIFO register (0x18)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:12	Reserved	R	0	reserved bit
11:9	TF_LEVEL	R	0	<p>Send FIFO water level status</p> <p>000: If the FIFO is not full, it means that there is 0 data in the FIFO, and if it is full, it means that there is 0 data in the FIFO.</p> <p>8 data;</p> <p>001: Indicates that the FIFO has 1 data;</p> <p>010: Indicates that the FIFO has 2 data;</p> <p>011: Indicates that the FIFO has 3 data;</p> <p>100: Indicates that the FIFO has 4 data;</p> <p>101: Indicates that the FIFO has 5 data;</p> <p>110: Indicates that the FIFO has 6 data;</p> <p>111: Indicates that the FIFO has 7 data;</p>

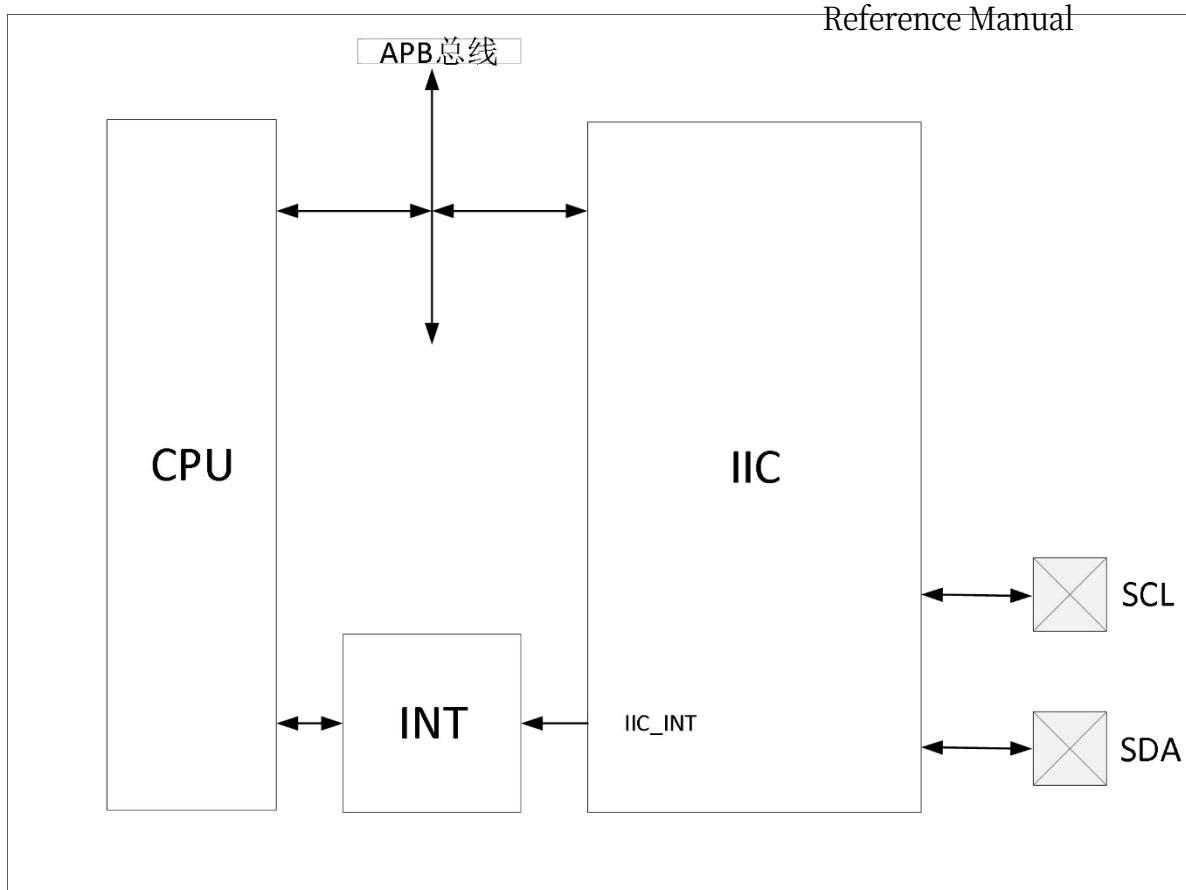
				Receive FIFO water level status Reference Manual 000: If the FIFO is not full, it means that there is 0 data in the FIFO, and if it is full, it means that there is 0 data in the FIFO. 8 data; 001: Indicates that the FIFO has 1 data; 010: Indicates that the FIFO has 2 data; 011: Indicates that the FIFO has 3 data; 100: Indicates that the FIFO has 4 data; 101: Indicates that the FIFO has 5 data; 110: Indicates that the FIFO has 6 data; 111: Indicates that the FIFO has 7 data;
5	TFHF	R	0	Send FIFO half full flag
4	TFF	R	0	Send FIFO full flag
3	TFE	R	1	Send FIFO Empty Flag
2	RFHF	R	0	Receive FIFO half-full flag
1	RFF	R	0	Receive FIFO full flag
0	RFE	R	1	Receive FIFO empty flag

5.18 IIC Controller (IIC)

5.18.1 summarize

IIC (Inter-Integrated Circuit) is a serial communication bus that uses a multi-master-slave architecture. The IIC bus is very simple in physical connection, consisting of SDA (serial data line) and SCL (serial clock line) and pull-up resistors. The communication principle is to control the high and low level timing of the SCL and SDA lines to generate the signals required by the IIC bus protocol for data transfer. When the bus is idle, these two lines are generally pulled up by the pull-up resistor connected to the top and kept at a high level. The IIC communication mode is half-duplex, there is only one SDA line, and only one-way communication is possible at the same time. The IIC bus data transmission rate can be as high as 100kbit/s in the standard mode, and 400kbit/s in the fast mode. The transmission rate can be adjusted by the programmable clock of the IIC bus interface. The transfer rate is usually adjusted by the programmable clock of the IIC bus interface and is also related to the resistance value of the connected pull-up resistor. The chip provides an IIC interface module to communicate with external IIC devices.

The following figure shows the system schematic of the IIC module. This module is connected to the CPU through the APB bus and can generate interrupts.



IIC Module System Schematic

5.18.2 characterization

- Supports master and slave modes
- Supports digital filtering of IIC input signals
- Supports 3 modes: ~~Slow (100kbps)~~ and ~~Fast-mode Plus (1Mbps)~~
- Readable data on SCL/SDA line
- Supports multiple interrupts

Master mode characteristics:

- Support SCL LOW timeout alarm
- The maximum supported SCL clock period is $(2^{17})^*$ system clock.
- SCL clock duty cycle is configurable

Slave mode characteristics:

- Supports 7-bit and 10-bit address modes
- Supports address masking, allowing a slave device to occupy multiple addresses, up to 128 addresses for a slave device in 7-bit address mode, and up to 256 addresses for a slave device in 10-bit address mode
- Supports clock stretching, where a slave device can hold the bus by pulling down the SCL.

5.18.3 Block Diagram of Module Structure

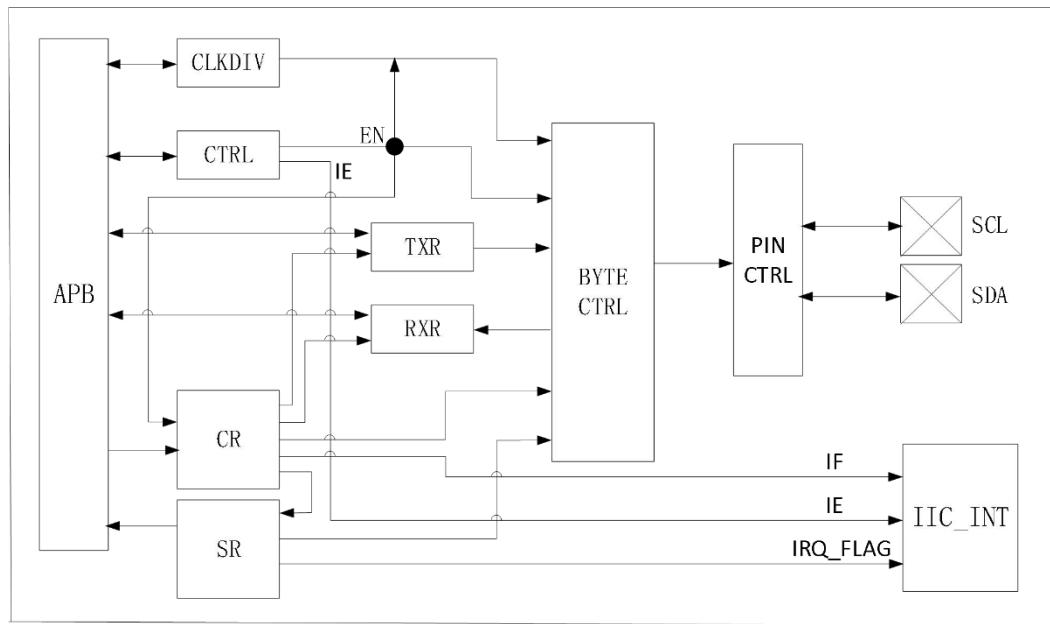


Figure 5-139 IIC Module Structure Block Diagram

The structure of IIC module is shown in the figure above. REG module implements register read/write and generates interrupt; IO module performs synchronization processing and digital filtering for SDA and SCL inputs, and selects SDA and SCL outputs; MASTER module implements the host function; SLAVE module implements the slave function; and TIMER implements the timing function, including START SCL hold time, TIMER implements timing functions, including START SCL hold time, SCL low level length timing, SCL high level length timing, SDA data hold time timing when sending data, SCL low level length timeout timing, etc.

5.18.4 Function

nal

Description

Protocol

Introduction

The IIC bus uses a serial data line (SDA) and a serial clock line (SCL) to transfer data.

Data is synchronized byte-by-byte between the master and slave devices via SCL clock signals on the SDA data lines. One bit of data is sent per SCL clock pulse, with the high bit coming first. Each byte of data sent generates an answer signal. Each bit of data is sampled during a high level on the clock line SCL. The data line SDA changes when the clock line SCL is low and remains stable when the clock line SCL is high.

Typically, a standard communication consists of four parts: start signal, slave address, data transfer, and stop signal. This is shown in the figure below:

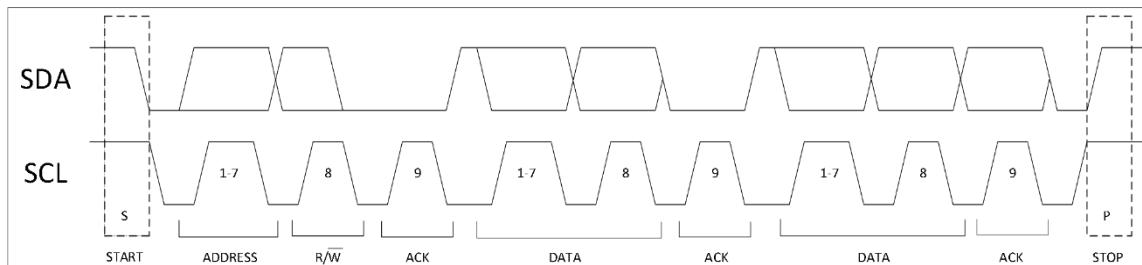


Figure 5-140 IIC Data Transfer Diagram

Start signaling

When the bus is idle, indicating that no host device is occupying the bus (both SCL and SDA are held high) the host can initiate a transfer by sending a start signal. The start signal, often referred to as the S bit, jumps SDA from high to low when SCL is high. The start signal indicates the beginning of a new data transfer.

Slave Address Transmission

The first byte of data transferred from the master after the start signal is the slave address. Contains the 7-bit slave address and the 1-bit R/W indicator bit, which signals the direction of data transfer to the slave, with 0 indicating a write operation and 1 indicating a read operation. Slaves in the system cannot have the same address. An answer bit (pulling SDA low on the ninth clock cycle) is generated in response only if the slave address matches the address sent by the master.

data transmission

Once the slave address has been successfully obtained, the host can send data byte-by-byte controlled by the R/W bits. Each byte transmitted requires an answer bit to be generated on the ninth clock cycle.

If the slave signal is invalid or the slave returns a NACK signal, the master can generate a stop signal to abort the data transfer.

If the master, as the receiving device, does not answer the slave, the slave releases the SDA and the master generates a stop signal.

Stop signal transmission

The host can terminate communication by generating a stop signal. The stop signal is often referred to as the P bit and is defined as SCL SDA jumps from low to high when it is high.

7bit address data transfer

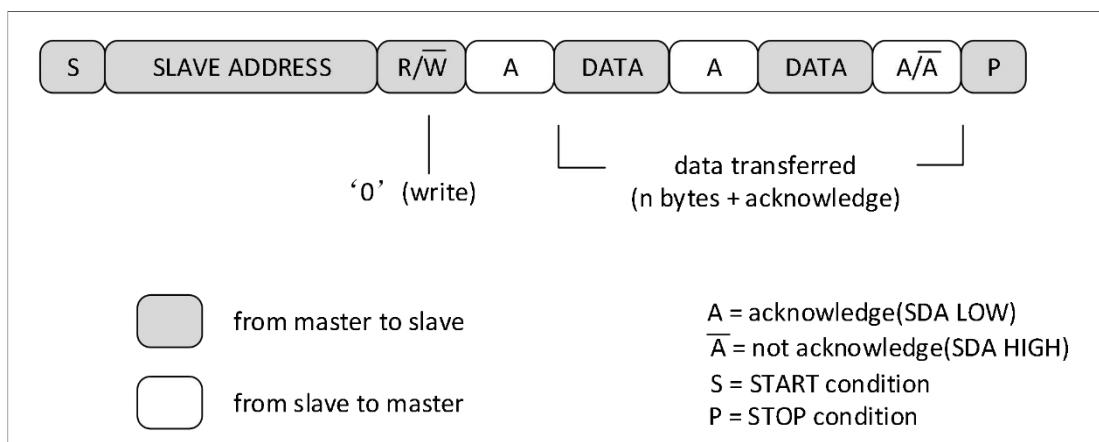


Figure 5-141 I²C Host Transmit Addressing Slave Receive with 7-Bit Addressing

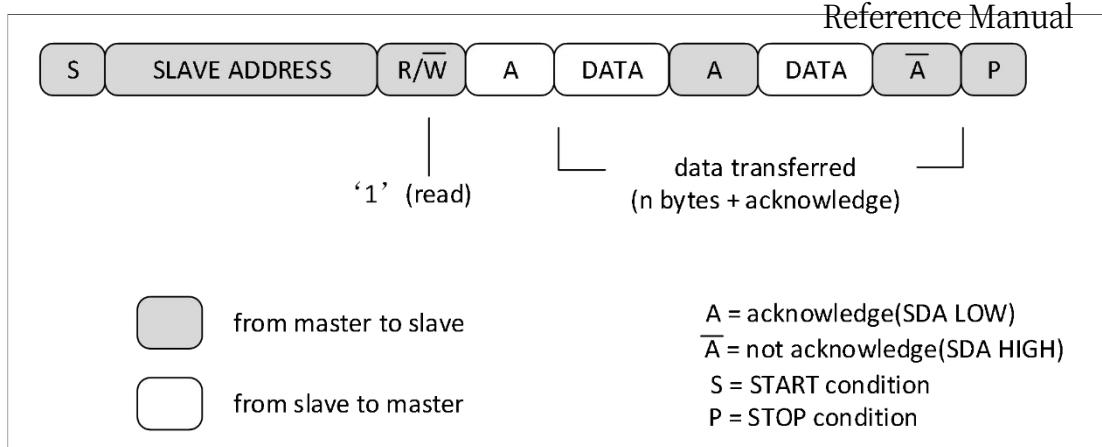


Figure 5-142 IIC Master Reads Slave Immediately After First Byte

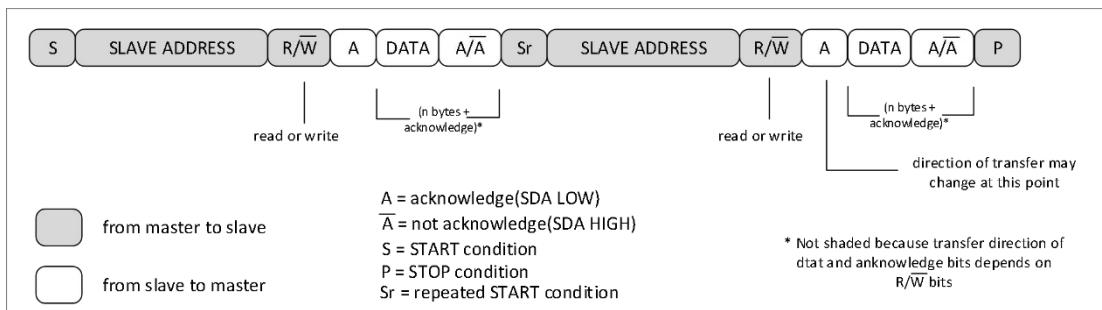


Figure 5-143 IIC Combination Format

10bit address data transfer

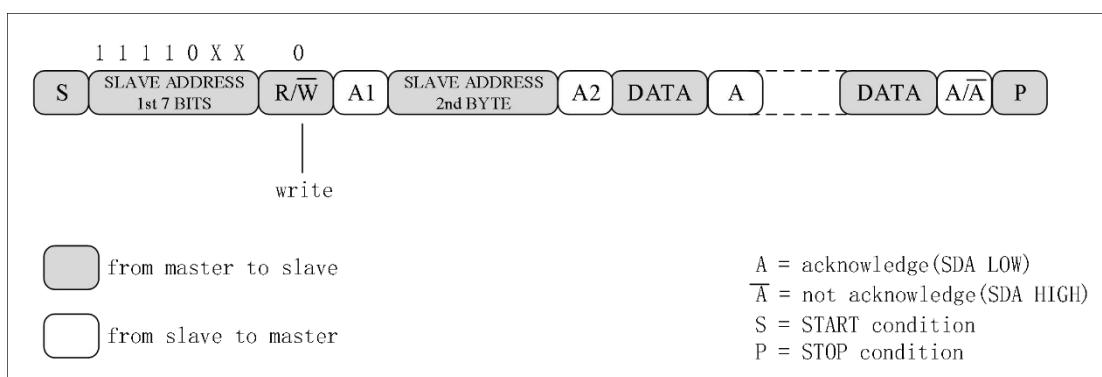


Figure 5-144 IIC Host Transmit Addressing Slave Receive with 10-Bit Addresses

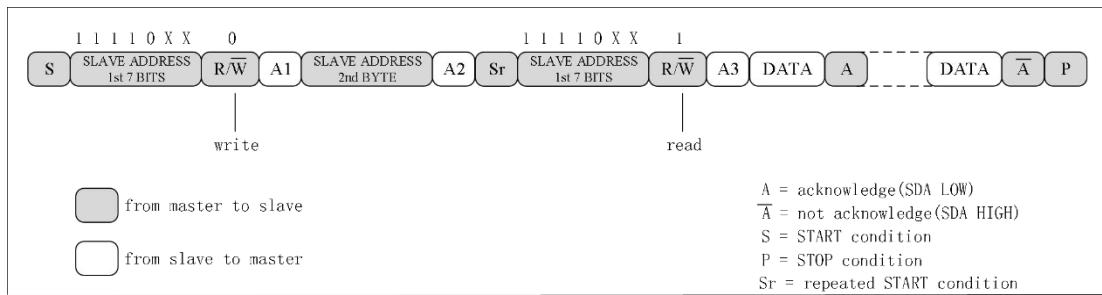


Figure 5-145 IIC Host Receives Slave Transmission with 10-Bit Address Addressing

SCL and SDA

In the IIC protocol, the port signals SCL and SDA are bidirectional. For Standard-mode, Fast-mode, SCL and SDA are bi-directional open-drain IO, the signals are pulled up to high level by a resistor with a slow rising edge.

The IOs used by SCL and SDA are normal bi-directional IOs, and the bi-directional open-drain IOs are realized by processing the enable signals. The bi-directional open-drain IOs of SCL and SDA are realized in the following way:

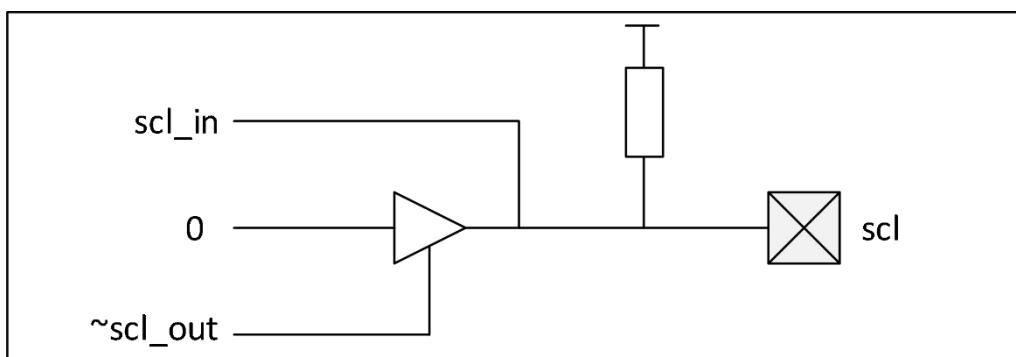


Figure 5-146 IIC SCL Open-Drain Diagram

interrupt function

This module provides six kinds of interrupt functions, including Receive Data Overflow, Send Complete, Receive Complete, SLAVE Detected START, SLAVE Detected STOP, and MASTER SCL LOW timeout interrupt. The corresponding interrupt enable bits can be configured through the IIC_IE register, and the corresponding interrupt status bits can be viewed through the IIC_IF register.

The interrupt flag and interrupt schematic are shown below:

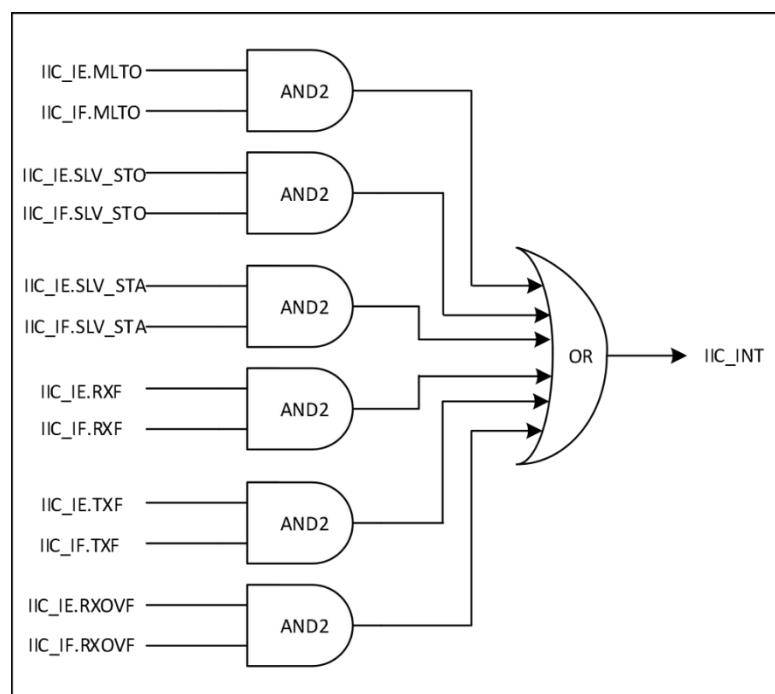
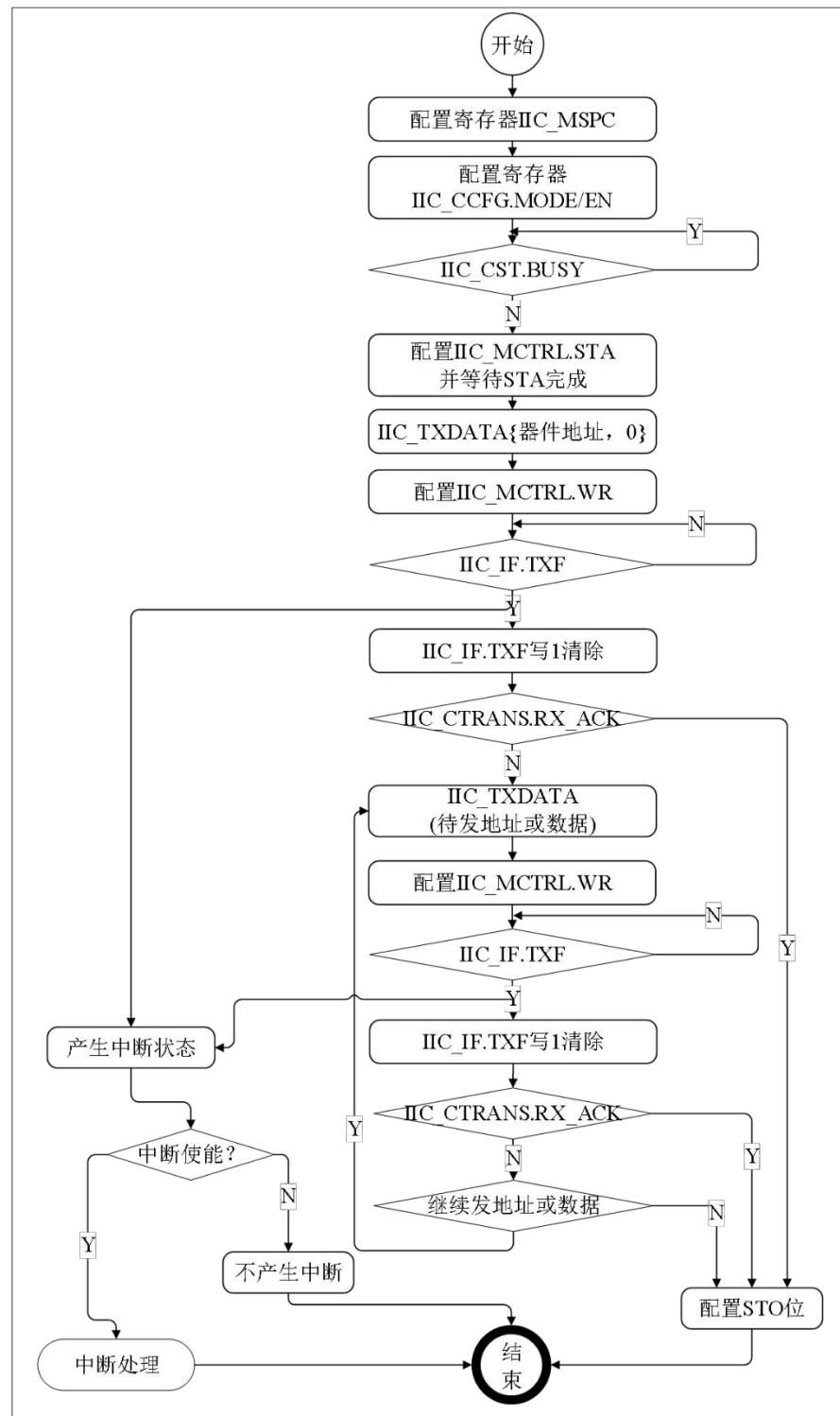


Figure 5-147 IIC Interrupt Flag and Interrupt Diagram

workflow

master-transmitter



1, set register IIC_MSPC. suppose pclk=60M, hope IIC work in Standard-mode (100kbps) speed, then each SCL 600 pclk, then tLOW is 400 pclk, tHIGH is 200 pclk, so you can set SCL_LOW=0xC4, SCL_HITH=0x60, CPD=0x01. HITH=0x60, CPD=0x01.

2. Set the MODE bit of register IIC_CCFG to 1 and the EN bit of register IIC_CCFG to 1.

3、Query the BUSY bit of register IIC_CST, if it is 1, then wait until it becomes 0; if it is 0, then proceed to the next step.

Set the STA bit of register IIC_MCTRL to 1. Query this bit until it becomes 0. 4.

5. Send the slave address byte, the specific steps are as follows:

Set register IIC_TXDATA, where bit7~bit1 is the slave address and bit0 is 0 i.e. write command;

Set the WR bit of register IIC_MCTRL to 1 and query the bit until it becomes (or query to register IIC_IF).

TXF bit to 1 (transmit successful) and write 1 to clear)

Read the RX_ACK bit of register IIC_CTRANS, if the bit is 0, it means the slave address match is successful and you can proceed to the next step, if the bit is 1, go to step 8.

6. Send the address or data to be written to the slave as follows:

Set register IIC_TXDATA to prepare the address or data to be written to the slave;

Set the WR bit of register IIC_MCTRL to 1 and query the bit until it becomes (or query to register IIC_IF).

TXF bit to 1 and write 1 to clear)

Read the RX_ACK bit of register IIC_CTRANS, if the bit is 0, it indicates that the write address or data is successful and you can proceed to the next step, if the bit is 1 go to step 8.

7. Repeat step 6 if you want to continue sending addresses or data, otherwise proceed to

the next step.

8. send STOP. set the STO bit of register IIC_MCTRL to 1, query the bit until it becomes 0.

If interrupt enable is set, the interrupt status can be checked when TXF transmission is completed and interrupt processing can be performed.



DP32G030

Reference Manual

master-receiver

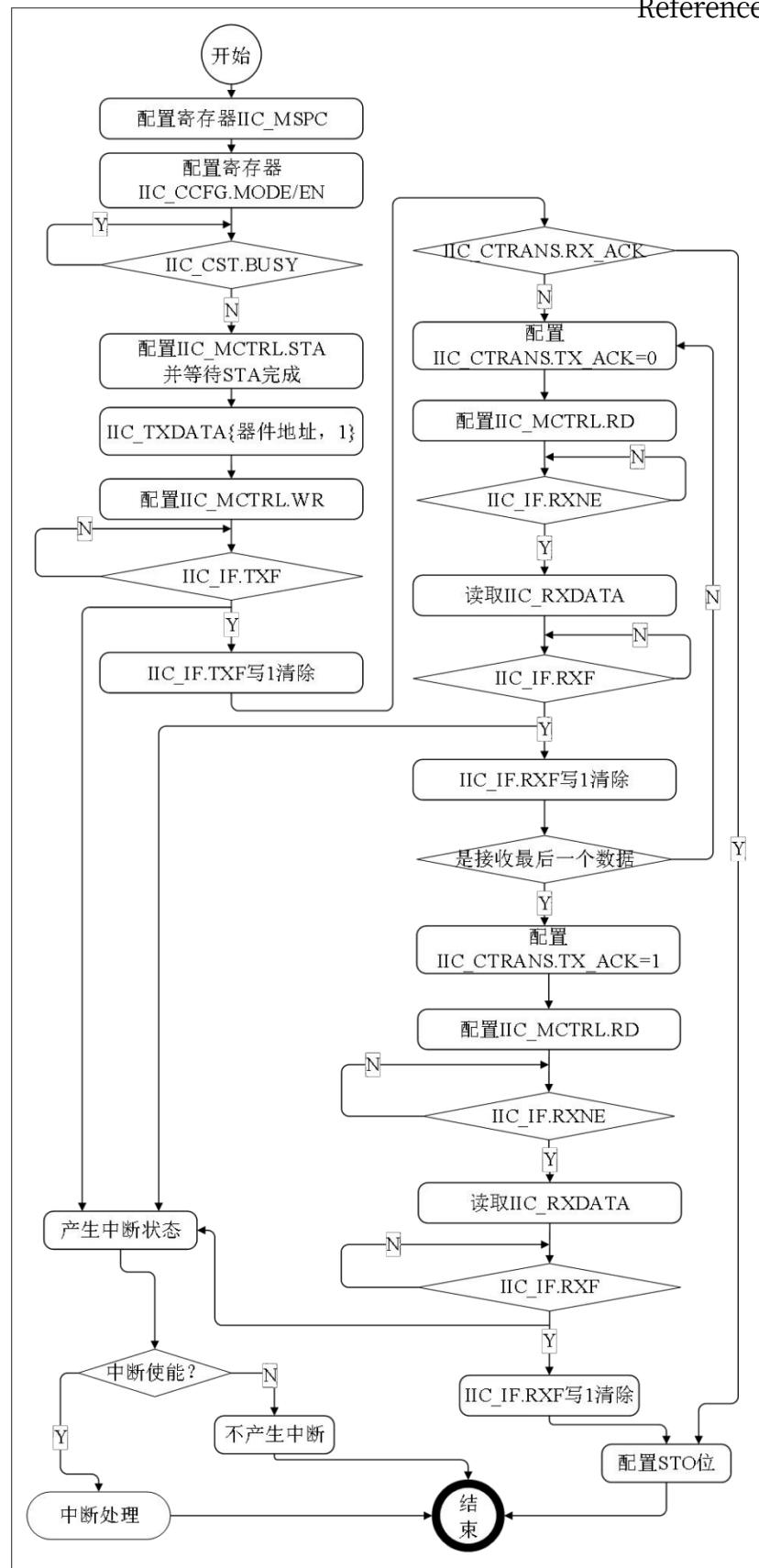


Figure 5-149 IIC Host Receiving Operation Flowchart

1, set register IIC_MSPC. suppose pclk=60M, hope IIC work in Standard-mode (100kbps) speed, then each SCL 600 pclk, then tLOW is 400 pclk, tHIGH is 200 pclk, so you can set SCL_LOW=0xC4, SCL_HITH=0x60, CPD=0x01. HITH=0x60, CPD=0x01.

2. Set the MODE bit of register IIC_CCFG to 1 and the EN bit of register IIC_CCFG to 1.

3、Query the BUSY bit of register IIC_CST, if it is 1, then wait until it becomes 0; if it is 0, then proceed to the next step.

Set the STA bit of register IIC_MCTRL to 1. Query this bit until it becomes 0.4.

5. Send the slave address byte, the specific steps are as follows:

Set register IIC_TXDATA, where bit7~bit1 is slave address and bit0 is 1 i.e. read command;

Set the WR bit of register IIC_MCTRL to 1 and query the bit until it becomes 0 (or query to register IIC_IF).

TXF bit to 1 (transmit successful) and write 1 to clear)

Read the RX_ACK bit of register IIC_CTRANS, if the bit is 0, it means the slave address match is successful and you can proceed to the next step, if the bit is 1, go to step 8.

6. Read data from the slave, the specific steps are as follows:

Sets the TX_ACK bit of register IIC_CTRANS to 0;

Set the RD bit of register IIC_MCTRL to 1, query until the RXNE bit of register IIC_IF is 1; read register IIC_RXDATA to get the slave data;

Query the RD bit of register IIC_MCTRL until it becomes 0 (or query register IIC_IF.RXF bit to 1 and write 1 to clear)

7, continue to receive and is not the last data received, repeat step 6; if it is the last data received, set the TX_ACK bit of register IIC_CTRANS to 1, and the others are the same as step 6, and then proceed to the next step after completion.

8. send STOP. set the STO bit of register IIC_MCTRL to 1, query the bit until it becomes 0.

If interrupt enable is set, the interrupt status can be checked when TXF

DP32G030

transmission is completed and RXF reception is completed, [Reference Manual](#)
processing can be performed.



DP32G030

Reference Manual

slave-transmitter

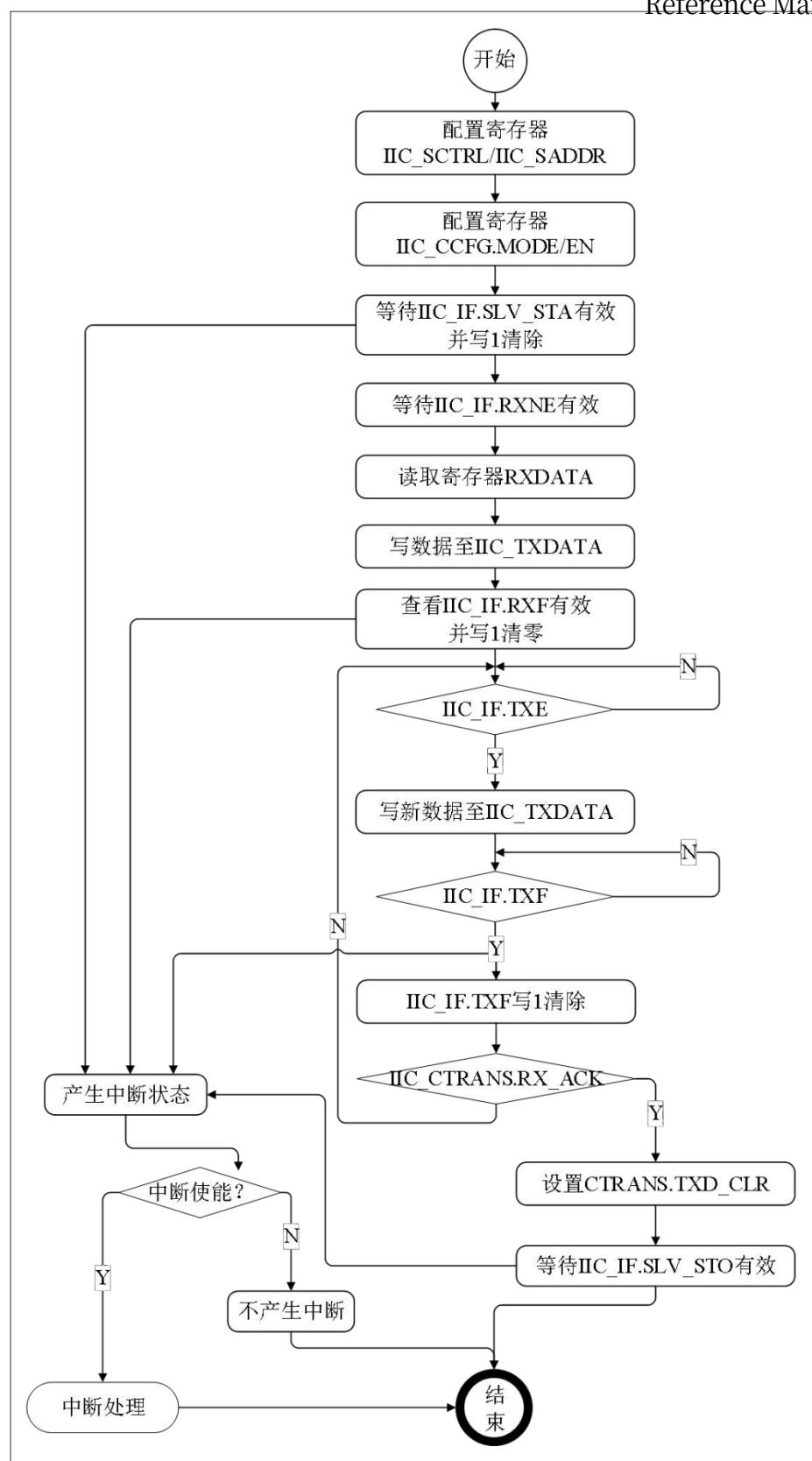


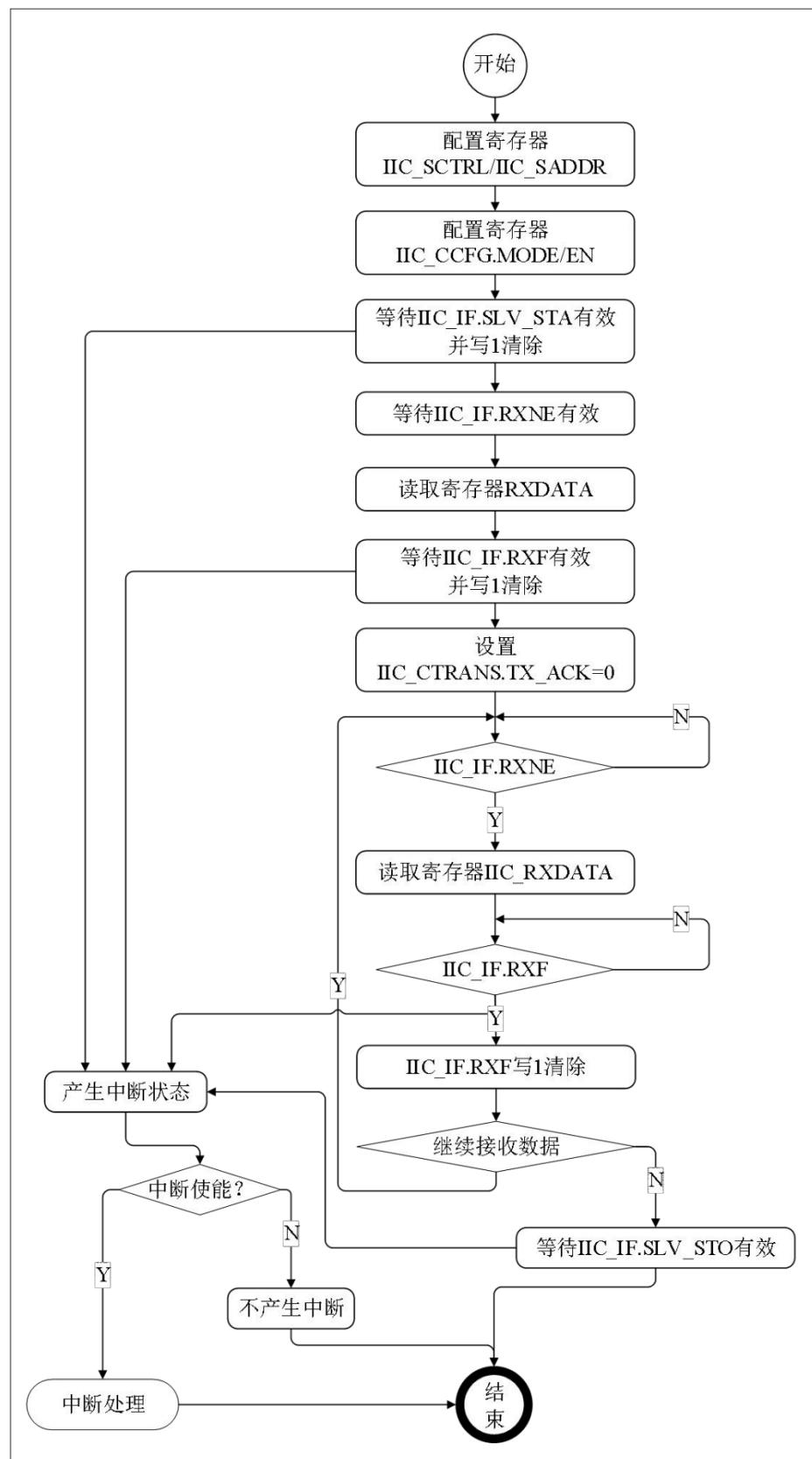
Figure 5-150 IIC Slave Transmit Operation Flowchart

1. Set slave address mode. The ADMD bit of register IIC_SCTRL is 0.
2. Set the slave address register IIC_SADDR.
3. Set the MODE bit of register IIC_CCFG to 0 and the EN bit of register IIC_CCFG to 1.
4. Query the SLV_STA bit of the IIC_IF register to indicate that a start has been detected on the IIC bus, and write 1 Zero.
5. Query until the RXNE bit of IIC_IF is 1, indicating that the device is selected by a master.
 6. Read register IIC_RXDATA, if the address mask is set in register IIC_SADDR, judge master The actual address to send to.
 7. Prepare data and write register IIC_TXDATA.
 8. Query until the RXF bit of register IIC_IF is 1, indicating that the return ACK ends after the previous address match.
 9. Query until the TXE bit of register IIC_IF is 1, then you can write new data to TXDATA.
 10. query until the TXF bit of register IIC_IF is 1, indicating that data transmission is completed. Then write 1 to clear.
 - 11, query the RX_ACK bit of register IIC_CTRANS, if it is 0, it means that the master wants to continue to receive data, repeat 9~11; if the RX_ACK bit of register IIC_CTRANS is 1, it means that the master wants to end the read operation, set the TXD_CLR bit of register IIC_CTRANS to clear the last data that was prepared into register IIC_TXDATA before. Set the TXD_CLR bit of register IIC_CTRANS to clear the last data previously prepared into register IIC_TXDATA. Proceed to the next step.
- 12、Querying the SLV_STO bit of register IIC_IF indicates that a STOP is detected on the IIC bus. This session ends.
13. If interrupt enable is set, the interrupt can be set when SLAVE detects STA, RXF receive completion, TXF transmit completion, and

DP32G030

SLAVE checks the interrupt status when STO is detected, and [Reference Manual](#) can be performed.

slave-receiver



1. Set slave address mode. The ADMD bit of register IIC_SCTRL is 0.
2. Set the slave address register IIC_SADDR.
3. Set the MODE bit of register IIC_CCFG to 0 and the EN bit of register IIC_CCFG to 1.
4. Query the SLV_STA bit of the IIC_IF register to indicate that a start has been detected on the IIC bus, and write 1 Zero.

Wait until the RXNE bit of IIC_IF is 1. This indicates that a master has selected the device.

6. Read register IIC_RXDATA, if the address mask is set in register IIC_SADDR, judge master The actual address to send to.

7, query until the RXF bit of register IIC_IF is 1, indicating that the previous address is matched, and then return to the end of ACK. Then write 1 to clear.

8. Set the TX_ACK bit of register IIC_CTRANS to zero.

9. Query until the RXNE bit of IIC_IF is 1, indicating that the slave receives new data and reads the register.

IIC_RXDATA.

10, query until the RXF bit of register IIC_IF is 1, indicating that after receiving data before, return ACK end. Then write 1 to clear.

11、Repeat 9~10, continue to receive data until the SLV_STO bit of register IIC_IF is queried, indicating the end of this session.

12. If interrupt enable is set, the interrupt can be detected when SLAVE detects STARXF reception is completed and SLAVE detects STO.

The interrupt status is checked at the time, and interrupt processing can be performed.

clock-stretching

Take master-receiver as an example, as follows:

- 1, set register IIC_MSPC. suppose pclk=60M, hope IIC work in Standard-mode (100kbps) speed, then each SCL 600 pclk, then tLOW is 400 pclk, tHIGH is 200 pclk, so you can set SCL_LOW=0xC4, SCL_HITH=0x60, CPD=0x01. HITH=0x60, CPD=0x01.
2. Set the MODE bit of register IIC_CCFG to 1 and the EN bit of register IIC_CCFG to 1.
- 3、Query the BUSY bit of register IIC_CST, if it is 1, then wait until it becomes 0; if it is 0, then proceed to the next step.
Set the STA bit of register IIC_MCTRL to 1. Query this bit until it becomes 0. 4.

5. Send the slave address byte, the specific steps are as follows:
Set register IIC_TXDATA, where bit7~bit1 is slave address and bit0 is 1 i.e. read command;
Set the WR bit of register IIC_MCTRL to 1 and query the bit until it becomes (or query to register IIC_IF).
TXF bit to 1 (transmit successful) and write 1 to clear)

Read the RX_ACK bit of register IIC_CTRANS, if the bit is 0, it means the slave address match is successful and you can proceed to the next step, if the bit is 1, go to step 8.

6, read data from slave: if the slave prepares the data for a long time, the master's SCL will be held by the slave until the slave data is ready, if the hold time is too long, it will cause the master to generate an interrupt of the MLTO bit of the IIC_IF register, the specific steps are as follows:

Sets the TX_ACK bit of register IIC_CTRANS to 0;
Set the RD bit of register IIC_MCTRL to 1, query until the RXNE bit of register IIC_IF is 1; read register IIC_RXDATA to get the slave data;
Query the RD bit of register IIC_MCTRL until it becomes 0 (or query register

DP32G030
Reference Manual

IIC_IF.RXF bit to 1 and write 1 to clear)

7, continue to receive and is not the last data received, repeat step 6; if it is the last data received, set the TX_ACK bit of register IIC_CTRANS to 1, and the others are the same as step 6, and then proceed to the next step after completion.

8. send STOP. set the STO bit of register IIC_MCTRL to 1, query the bit until it becomes 0.

If interrupt enable is set, the interrupt status can be checked when TXF transmission is completed and RXF reception is completed, and interrupt processing can be performed.

Take slave-transmitter as an example, as follows:

1. Set the slave address mode. The ADMD bit of register IIC_SCTRL is 0 and the STRETCH bit is 1.
2. Set the slave address register IIC_SADDR.
3. Set the MODE bit of register IIC_CCFG to 0 and the EN bit of register IIC_CCFG to 1.
4. Query the SLV_STA bit of the IIC_IF register to indicate that a start has been detected on the IIC bus, and write 1 Zero.
5. Query until the RXNE bit of IIC_IF is 1, indicating that the device is selected by a master.
 6. Read register IIC_RXDATA, if the address mask is set in register IIC_SADDR, judge master The actual address to send to.
 7. prepare data, write register IIC_TXDATA, if the time to prepare data is long, the slave hardware will automatically pull down the SCL, preventing the master from issuing a new SCL. when the slave completes the action of writing register IIC_TXDATA, the slave hardware will wait for the SCL_LOW time set in register IIC_MSPC again, then release the SCL to proceed to the next step. The slave hardware will wait for the SCL_LOW time set by IIC_MSPC before releasing the SCL for the next step.
 8. Query until the RXF bit of register IIC_IF is 1, indicating that the return ACK ends after the previous address match.

9. Query until the TXE bit of register IIC_IF is 1, then you can write data to TXDATA.

10, query until the TXF bit of register IIC_IF is 1, indicating that data transmission is completed. Then write 1 to clear.

11, query the RX_ACK bit of register IIC_CTRANS, if it is 0, it means that the master wants to continue to receive data, repeat 9~11; if the RX_ACK bit of register IIC_CTRANS is 1, it means that the master wants to end the read operation, set the TXD_CLR bit of register IIC_CTRANS to clear the last data that was prepared into register IIC_TXDATA before. Set the TXD_CLR bit of register IIC_CTRANS to clear the last data previously prepared into register IIC_TXDATA. Proceed to the next step.

12、Querying the SLV_STO bit of register IIC_IF indicates that a STOP is detected on the IIC bus. This session ends.

13. If interrupt enable is set, the interrupt can be set when SLAVE detects STA, RXF receive completion, TXF transmit completion, and SLAVE checks the interrupt status when STO is detected, and interrupt processing can be performed.

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
IICOBASE: 0x400B9000					
IIC1BASE: 0x400B9800					
IIC_CCFG	0x0	32	R/W	0x18	General Configuration Register
IIC_CST	0x4	32	RO	0x06	General Status Register
IIC_CTRANS	0x8	32	R/W	0x02	General Purpose Transfer Register
IIC_RXDATA	0xC	32	RO	0x00	Receive Data Register
IIC_TXDATA	0x10	32	R/W	0x00	Transmit Data Register
IIC_IE	0x14	32	R/W	0x00	Interrupt Enable Register
IIC_IF	0x18	32	R/W	0x01	Interrupt Flag Register
IIC_MCTRL	0x20	32	R/W	0x00	Host Mode Control Register
IIC_MSFC	0x24	32	R/W	0x33f7f	Timing Configuration Register
IIC_SCTRL	0x30	32	R/W	0x08	Slave Mode Control Register
IIC_SADDR	0x34	32	R/W	0x00	Slave Mode Address Register

register description

IIC_CCFG register (0x00)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:7	RESERVED	RO	0	reserved bit

6:3	DNF	R/W	0x3	Receive SDA, SCL Digital Noise Filter 0000: Filter not enabled 0001: Filter enable and filter capacity up to 1 system clock 1111: Filter enable and filter capacity up to 15 system clocks
2	RESERVED	RO	0	reserved bit
1	MODE	R/W	0x0	Mode Control 0: slave mode 1: master mode
0	EN	R/W	0	IIC Bus Enable 0: not enabled 1: Enabling

IIC_CST register (0x04)

bitfield	name (of a thing)	typology	reset value	descriptive
31:14	RESERVED	R	0	reserved bit
13:12	SLV_RXDT	RO	0	The type of data received by the Slave. Valid only in Slave mode. 00: RXDATA is empty. 01: The address is received. 10: Data is received. 11: Master code is received. valid only when MCDE=1.
11	SLV_STRETCH_BUSY	RO	0	Slave clock stretching busy state. Valid only in slave mode. 0: Without clock stretching. 1: With

				clock stretching.	
10	SLV_WR	RO	0	Slave write status. Valid only in slave mode. 1: Valid after the slave receives a write request from the master. 0: The slave is cleared automatically after receiving a read request from the master or STOP.	

Reference Manual				
9	SLV_RD	RO	0	Slave read state. Valid only in slave mode. 1: Valid after the slave receives a read request from the master. 0: The slave is cleared automatically after receiving a write request from the master or STOP.
8	SLV_ACTIVE	RO	0	Slave active state. Valid only in slave mode. 0: slave device is inactive 1: The slave device is active. This bit is active after a successful address match; it is automatically cleared after receiving a STOP, or an unsuccessful address match after Start_repeat.
7:3	RESERVED	RO	0	Reserved Bits.
2	SDA	RO	1	IIC SDA Status. Not affected by IIC bus enable 0: IIC SDA is low 1: IIC SDA is high
1	SCL	RO	1	IIC SCL Status. Not affected by IIC bus enable 0: IIC SCL is low 1: IIC SCL is high
0	BUSY	RO	0	Bus Busy Status. This bit is not controlled by the IIC_CCFG.EN bit and still detects the bus busy state when EN is not enabled 0: Bus not busy 1: Bus busy, IIC bus active from START to STOP

IIC_CTRANS register (0x08)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)				
31:3	RESERVED	R	0	reserved bit

2	TXD_CLR	W, AC	0	Transmit data register is cleared. Hardware clears it automatically. 0: Not cleared. 1: Clear the IF.TXE bit.
1	RX_ACK	RO	1	When acting as a sender, received ACK/NACK. hardware set, IF.TXF is valid to query this bit; received Start_repeat or STOP will set this bit 1. 0: ACK received 1: NACK received

				Sends ACK/NACK when acting as receiver. 0: Sends ACK. 1: Send NACK.	Reference Manual
0	TX_ACK	R/W	0	<p>Note: In the following cases, ACK/NACK is not determined by this bit:</p> <ul style="list-style-type: none"> A. The hardware automatically sends an ACK/NACK when the slave receives the address. B. slave MCDE is valid, hardware automatically returns NACK when master code is received. C. The hardware automatically sends a NACK when the slave receives an overflow. 	

IIC_RXDATA register (0x0C)

bitfield Id (mat h.)	name (of a thing)	typology	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	RXDATA	R	0	<p>Receive data register. IF.RXNE is 1, indicating that valid data exists in this register.</p> <p>Note: This register is updated at the moment of completion of data reception (not including ACK/NACK transmission).</p> <p>See the RXF bit description for slave receive address byte conditions.</p>

IIC_TXDATA register (0x10)

bitfield Id (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	TXDATA	R/W	0	Transmit Data Register. IF.TXE is 0, indicating that there is data to be sent in this register.

IIC_IE register (0x14)

bitfield Id (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:18	RESERVED	R	0	reserved bit
17	MLTO	R/W	0	Master SCL LOW Timeout interrupt enable. 0: Not enabled. 1: Enable.
16:10	RESERVED	RO	0	Reserved Bits.
9	SLV_STO	R/W	0	Slave detects STOP interrupt enable. 0: Not enabled. 1: Enable.
8	SLV_STA	R/W	0	Slave detects START interrupt enable. 0: Not enabled. 1: Enable.
7:5	RESERVED	RO	0	Reserved Bits.
4	RXF	R/W	0	End of receive data interrupt enable. 0: Not enabled. 1: Enable.
3	TXF	R/W	0	End-of-send data interrupt enable. 0: Not enabled. 1: Enable.
2	RXOVF	R/W	0	Receive data register overflow interrupt enable. 0: Not enabled. 1: Enable.
1:0	RESERVED	RO	0	Reserved Bits.

IIC_IF register (0x18)

bitfie ld (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:18	RESERVED	R	0	reserved bit

17	MLTO	R, W1C	0	Master SCL LOW timeout. Write 1 Clear. Valid only in master mode. 0: No timeout. 1: Timeout. the SCL LOW time exceeds 1024 SCL LOW times set by the MSPC registers.
16:10	RESERVED	RO	0	reserved bit
9	SLV_STO	R, W1C	0	Slave detected STOP. write1 clear. Valid only in slave mode. 0: slave did not detect STOP. 1: slave detected STOP.
8	SLV_STA	R, W1C	0	Slave detects START. write 1 clear. Valid only in slave mode. 0: slave does not detect START. 1: slave detects START.
7:5	RESERVED	RO	0	reserved bit
4	RXF	R, W1C	0	End of reception. Write 1 Clear, contains ACK/NACK time. 0: Receiving is not finished. 1: End of reception. Note: The slave receives the situation description: 1. In 7-bit address mode of Slave device, the slave address byte (inclusive) is used as the address of the slave device. (R/W bits) Receipt is complete and this interrupt is generated if the address matches. 2. In 10-bit address mode of Slave device, when the second byte of slave address (ADDR[7:0]) is received, this interrupt will be generated if the 10-bit address matches; the first byte of the slave address following repeat START, this interrupt will be generated if the address of 8 bits matches; the first byte following repeat START will not generate this interrupt even if ADDR[9:8] matches after the first byte of the slave address has been received. After the first byte following START is received, even if ADDR[9:8] matches, this interrupt will not be

				<p>generated.</p> <p>Reference Manual</p> <p>3. Slave mode, SCTRL.MCDE=1, receive master code</p> <p>This interrupt is generated when</p>
3	TXF	R, W1C	0	<p>End of transmission. Write 1 Clear, contains ACK/NACK time.</p> <p>0: Sending is not finished, or not sent. 1: End of sending.</p>

2	RXOVF	R, W1C	0	<p>Receive data register overflow Software Write 1 Clear.(Updated point in time, not including ACK/NACK transmissions)</p> <p>0: No overflow. 1: When RXDATA is non-null and a new byte is received, an overflow occurs. When the overflow occurs, the new data is lost.</p> <p>Note: For slave mode, if the STRETCH bit is valid, when the receive data register is non-empty and a new byte is received, the slave device pulls down the SCL signal until the old data in RXDATA is read away, and then stores the new data in RXDATA, this case will not generate an overflow.</p>
1	RXNE	RO	0	<p>The receive data register is not empty. 0: Receive data register empty, no unread receive data exists. 1: The receive data register is not empty and there is unread receive data. Note: This bit is updated at the moment when the data is finished being received (excluding the ACK/NACK transmit time)</p> <p>If the old data is not read in time when the reception of new data is completed, it is handled in the following cases:</p> <p>Master mode: New data is lost. RXOVF bit is set at the same time.</p> <p>Slave mode:</p> <ul style="list-style-type: none"> A. STRETCH=0: New data is lost. Simultaneously set IF. RXOVF bit, the hardware automatically sends a NACK. B. STRETCH=1: Normal return to ACK, then before the master sends the next byte, slave holds SCL low until the old data is read away and then updates the new data into the RXDATA register. Finally release SCL.

0	TXE	RO	1	<p>Send data register empty Reference Manual</p> <p>0: The transmit data register is non-empty and writing the TXDATA register is not allowed.</p> <p>1: The transmit data register is empty, allowing the TXDATA register to be written.</p> <p>Note: This bit is updated to 1 after the transmit data is read away by the hardware at the moment the transmit data is started (when IF.TXF is still 0)</p> <p>Writing new data to the TXDATA register clears this bit.</p>
---	-----	----	---	--

IIC_MCTRL register (0x20)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)		gy		

31:4	RESERVED	RO	0	reserved bit	Reference Manual
3	STO	W,AC	0	Write 1 to generate STOP, which is automatically cleared upon completion.	
2	WR	W,AC	0	<p>Write 1 to send the data in TXDATA, which is automatically cleared when finished (including ACK/NACK time).</p> <p>Before writing 1 to this bit, TXDATA must not be empty. Otherwise, this bit cannot be set.</p> <p>Note: The WR and RD bits cannot be written to 1 at the same time.</p>	
1	RD	W,AC	0	<p>Write 1, receive data into RXDATA, and when finished (with ACK/NACK), write 1, receive data into RXDATA.</p> <p>time) is automatically cleared.</p>	
0	STA	W,AC	0	<p>Write 1 to generate START, which is automatically cleared upon completion.</p> <p>Note: Allow STA and WR to be set at the same time to prioritize sending START.</p>	

IIC_MSPC register (0x24)

bitfield Id (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:28	RESERVED	RO	0	reserved bit
27:24	DAT_HD	R/W	0x00	<p>SDA Data Hold Time Configuration. (Valid for Master and Slave) For master: tHD;DAT = (DAT_HD + 4) * Tpclk</p> <p>For slave: tHD;DAT = (DAT_HD + DNF + 6) * Tpclk</p> <p>Note: If the application environment is harsh, it should be noted that a burr that occurs during the SDA data hold has the potential to cause the SDA to change along the time that advances the width of the burr (if there is no burr on the SCL at this time, an unanticipated</p>

				STA, STOP will occur on the bus. In this case, DAT_HD should be set so that tHD;DAT is greater than the maximum burr width.
23:16	CPD	R/W	0x03	Clock prescaling, see SCL_HI and SCL_LOW descriptions for details.(Only for (Master mode active))
15:8	SCL_HI	R/W	0x3f	SCL Clock High Time Configuration. (available only in Master mode) $tHIGH = (SCL_HI + 1) * (CPD + 1) + DNF + 6) * Tpclk$

7:0	SCL_LOW	R/W	0x7f	<p>SCL Clock Low Time Configuration. (Valid for Master mode; in slave mode, this register needs to be configured if the STRETCH function is enabled and SCTRL.ASDS is configured to 0.) After slave writes TXDATA, delay the time set in this register before releasing SCL)</p> <p>$t_{LOW} = (SCL_LOW + 1) * (CPD + 1) + DAT_HD + 5) * T_{pclk}$</p> <p>The period of SCL is $t_{HIGH} + t_{LOW}$.</p> <p>The recommended ratio of t_{HIGH} to t_{LOW} is 1:2.</p>
-----	---------	-----	------	--

IIC_SCTRL register (0x30)

bitfield ld (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:4	RESERVED	RO	0	reserved bit
3	ASDS	R/W	1	<p>Adaptive Data Setup Time Enable after Stretching. (Adaptive Stretching Data Setup)</p> <p>0: Adaptive is not enabled. Set by MSPC.</p> <p>1: Adaptive Enable. Automatically detects SCL when receiving master address</p> <p>Low level time as data build-up time after stretching.</p> <p>Note: Slave-transmitter, when the STRETCH register is set to valid and stretching occurs, the slave will continue to pull down the SCL for a period of time after the new data is ready to ensure that the SDA line meets the data setup time requirement.</p>

				Clock stretching enable control. 0: Clock stretching is not enabled. 1: Clock stretching is enabled.
2	STRETCH	R/W	0	<p>Note: When the slave acts as a receiver, when new data is received but the old data is not read in time (IF.RXNE=1) CTRANS. STRETCH_BUSY becomes effective, and after returning to ACK, hold SCL at low level until the old data is read, then update the new data to RXDATA, and at the same time, STRETCH_BUSY becomes ineffective, and then release SCL to start the next data reception. BUSY becomes invalid, then release SCL to start the next data reception.</p> <p>When the slave acts as a transmitter, when the transmission is finished (IF.TXF=1, including the time of receiving ACK/NACK) but the new data is not ready (TXE=1) CTRANS. STRETCH_BUSY becomes active, holds SCL at low level until the new data is ready, and after delaying the SCL_LOW time. STRETCH_BUSY becomes invalid, and then release SCL to start sending new data).</p>
1	MCDE	R/W	0	<p>Master Code Detect Enable. 0: Master code is not detected. 1: Master code is detected.</p> <p>When this bit is valid, slave detects master code after START and generates IF.RXF interrupt and hardware sets CST.SLV_RXDT to 11. software should make sure that slave address setting does not conflict with master code.</p>
0	ADMD	R/W	0	<p>slave Address mode control.</p> <p>0: 7-bit address mode</p> <p>1: 10-bit address mode</p>

IIC_SADDR register (0x34)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat)				

h.)				
31:24	RESERVED	RO	0	reserved bit

Reference Manual				
23:17	MASK_ADDR[7:1]	R/W	0	Slave Corresponding Address Bit Mask. 0: No masking. 1: The mask corresponds to the bit address. After masking, the masked address bits are ignored when hardware matches the slave address. For 10-bit address mode, RXDATA only holds ADDR[7:0], so masking of ADDR[9:8] is not supported.
16	MASK_ADDR0	R/W	0	Slave Corresponding Address Bit Mask.
15:10	RESERVED	RO	0	Reserved Bits.
9:8	ADDR[9:8]	R/W	0	7-bit address mode: don't care 10-bit address mode: address bit9~bit8
7:1	ADDR[7:1]	R/W	0	Address bit7~bit1
0	ADDR0	R/W	0	7-bit address mode: don't care 10-bit address mode: address bit0

5.19 Analog-to-digital converter (ADC)

5.19.1 summarize

The SARADC adopts the successive approximation structure, which can sample the analog signal of 14 active channels and realize A/D Conversion.

Channels 0 - 10 are used for sampling external analog input signals. Channel 1 can optionally sample the OPA0 analog output and channel 10 can optionally sample the OPA1 analog output.

Channels 11 and 12 are reserved for non-use.

Channel 13 is used to sample the chip temperature.

Channel 14 is used to sample the 1.2V

reference voltage. Channel 15 samples

the AVDD/3 voltage. The system block

diagram is shown below:

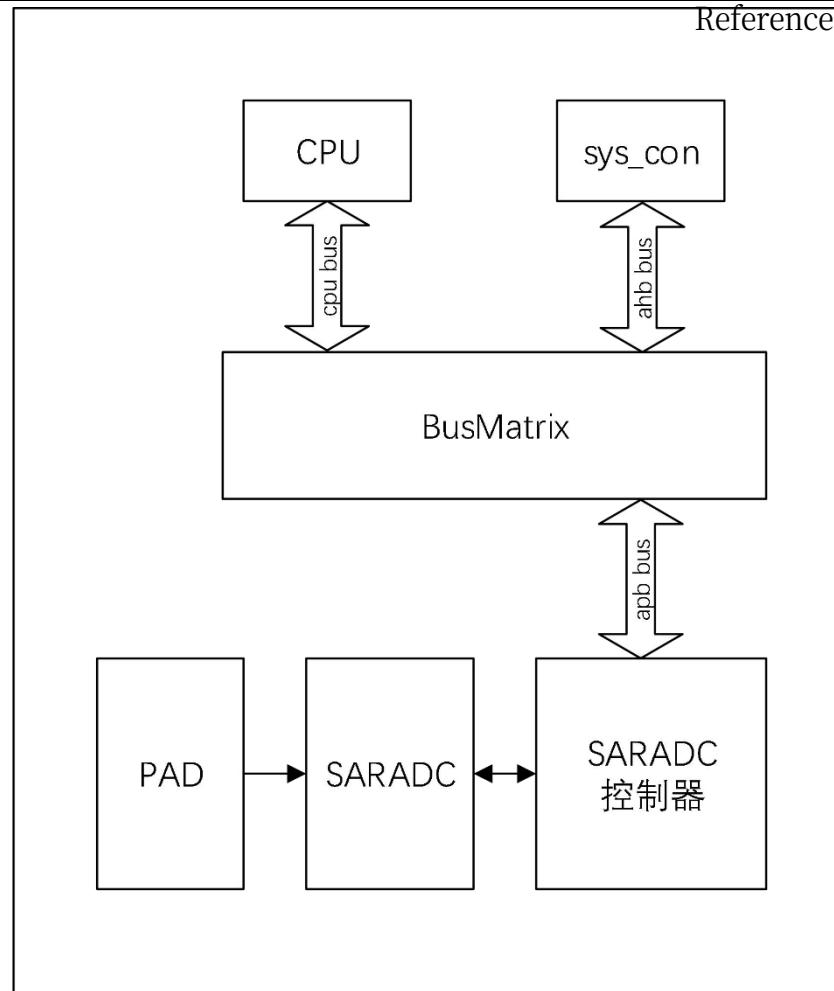


Figure 5-152 SARADC System Block Diagram

5.19.2 characterization

- 11 analog channel inputs from external IO ports;
- Sample rates up to 2.4M;
- The 0-10 channel is a high-speed channel, and the 13-15 channel is a low-speed channel; the high-speed channel has a short setup time and requires a short sampling window, while the low-speed channel has a long setup time and requires an appropriate increase in the sampling window to ensure the accuracy of the sampled voltage;
- Channel 13 is the on-chip temperature sensor voltage input;
- 14 channels are 1.2V reference voltage inputs;

- Channel 15 is the AVDD/3 voltage input;

- External ADC reference voltage PAD_VREF, some packages PAD_VREF is connected to AVDD_ADC;
- Separate channel data registers, channel-wide common 16-level FIFO available;
- Software trigger and hardware trigger, hardware trigger can select the overflow signal of the TIMERPLUS module and the PMWPLUS trigger signal;
- Support for sampling once or sampling multiple times for averaging is configurable;
- Supports single and continuous sampling configurable;
- Single Sample traverses all selected channels once from low channel to high channel after startup;
- Continuous sampling, after startup, repeatedly traverses all selected channels from low channel to high channel without interruption until the software stops sampling;
- Support internal sampling clock, external sampling clock can be configured;
- Support for sampling windows is configurable;
- Interrupts can be generated for each channel conversion completion state, FIFO full state, and FIFO half-full state;
- ADC channel input range: $0 \leq V_{IN} \leq V_{ref} \leq AVDD_ADC$;
- Supports DMA reading of FIFO sample data;

5.19.3 Block Diagram of Module Structure

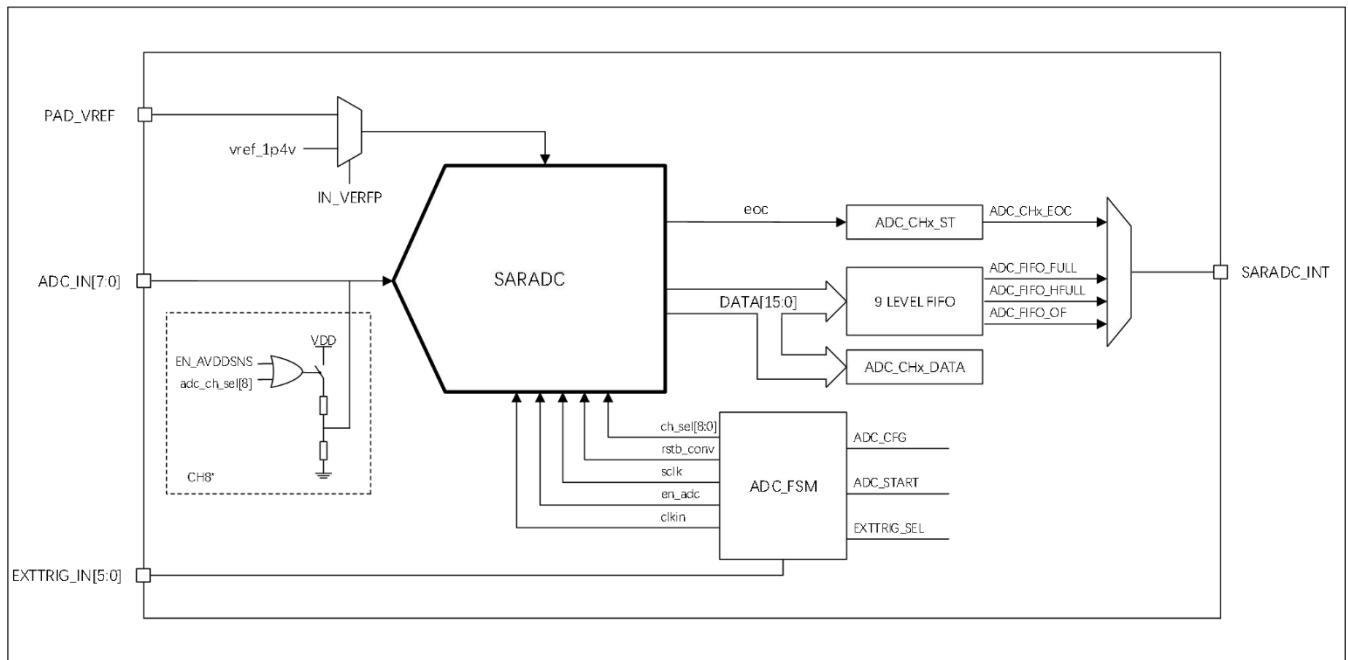


Figure 5-153 SARADC Module Structure Diagram

The SARADC has 11 external ADC sampling channels, and the voltage of the external ADC sampling channels should be less than the reference voltage **PAD_VREF**. The ADC control unit generates the corresponding control timing according to the ADC configuration, and controls the ADC to perform sampling. Sampling results can be stored in each channel register or in a FIFO, and generate a sample complete state or FIFO half-full and full states.

The SARADC supports 12 external trigger sources, **TIMER_PLUS1_GOAL[1:0]** from the **TIMERPLUS** module.

(High Counter Overflow Signal and Low Counter Overflow Signal for **TIMERPLUS1** Module)

and **TIMER_PLUS0_GOAL[1:0]**

(the high counter overflow signal and the low counter overflow signal from the **TIMERPLUS0** module) and from the **PWMPLUS** module.

PWM_PLUS1_TRIGGER[3:0] (trigger signal of **PWMPLUS1** module) and **PWM_PLUS0_TRIGGER[3:0]** (trigger signal of **PWMPLUS0** module)

5.19.4 Function

n

Description

Channel

Selection

The SARADC has a total of 16 sampling channels, of which 14 are active and 2 are inactive (channels 11 and 12) and should not be used. The ADC_CH_SEL bit in the control register ADC_CFG controls which channels are valid for the next sample.

After initiating the sampling, the control unit will traverse all the selected active channels from low channel to high channel to complete the sampling.

Kind.

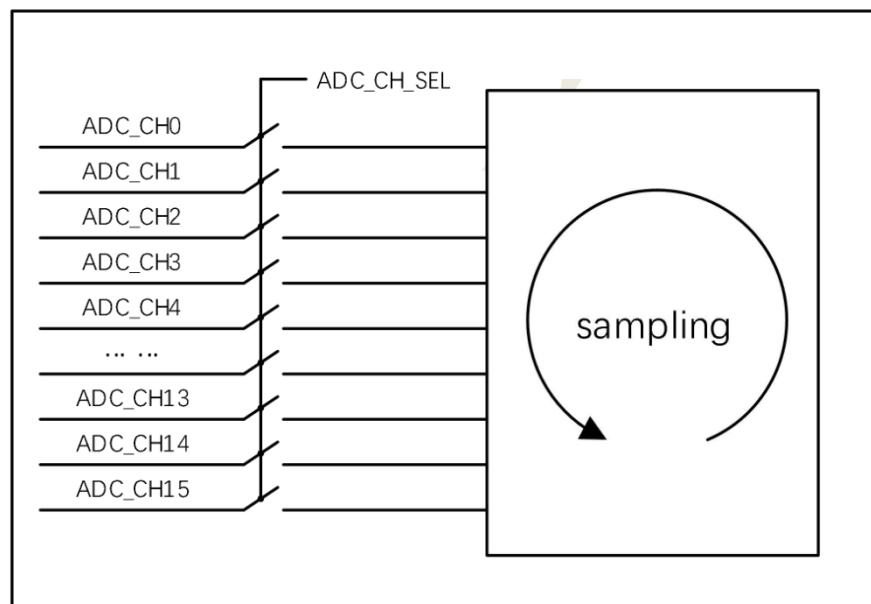


Figure 5-154 SARADC Channel Selection Schematic

Sampling once vs. averaging multiple samples

The configuration register `SPL_NUM` is used to set how many samples are averaged for each channel. For example, if `SPL_NUM` is set to average 8 samples, then each channel will take 8 complete samples each time it samples, and `SARADC_CTRL` will average the results of the 8 samples and store them into the channel registers or FIFOs.

The channel sampling completion flag is generated after averaging.

Single vs. continuous sampling

The SARADC sampling mode can be set through the configuration register ADC_MD. The SARADC has two modes: single sampling and continuous sampling. In single sampling mode, the SARADC will traverse the active channels to sample once and then stop waiting for the next sample; in continuous sampling mode, the SARADC will traverse the active channels to sample in a loop and continue to sample until the software stops sampling.

Single channel single sample

Single channel single sampling: the effective channel is only one single channel, i.e., only one bit of ADC_CH_SEL is high, and the SARADC sampling mode is single sampling.

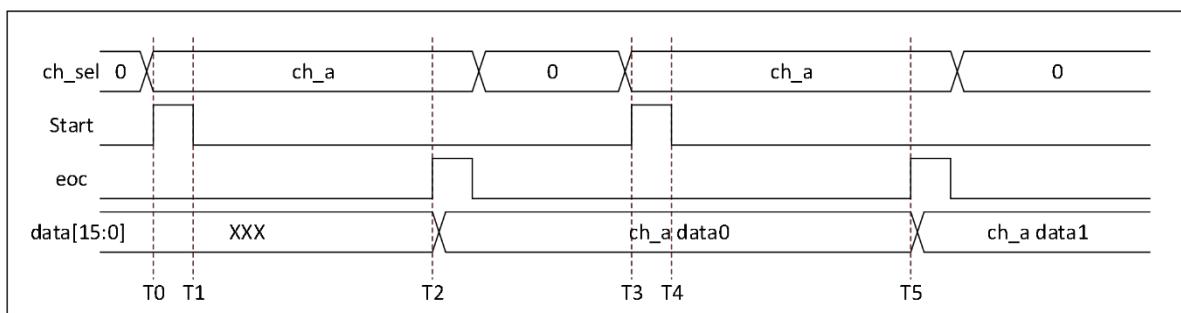


Figure 5-155 SARADC Single Channel Single Sample Schematic

The above figure shows the single sample timing diagram for channel a:

- The CPU programs the ADC_START bit in the ADC_START register to be 1 at time T0, initiating the SARADC to sample channel a. The SARADC will then be able to sample channel a at time T0;

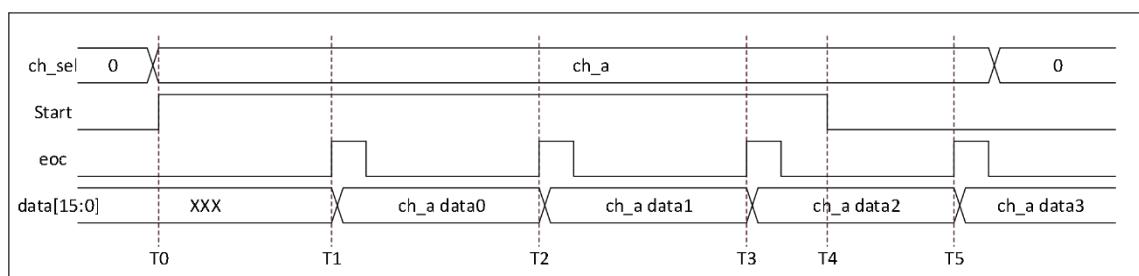
- The ADC_START bit is cleared by the digital logic hardware after a conversion is completed.

- At the time of T_2 , SARADC completes sampling once, outputs channel a sampling data `ch_a data0`, and the `eoc` flag is pulled high, at this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- To resample the channel, the CPU needs to write 1 to the `ADC_START` bit in the `ADC_START` register again, as shown in the above figure at time T_3 , initiating the SARADC to resample channel a. The CPU must write 1 to the `ADC_START` bit in the `ADC_START` register again;
- The `ADC_START` register is cleared by the digital logic hardware at time T_4 ;
- At the time of T_5 , SARADC completes sampling once, outputs channel a sampling data `ch_a data1`, and the `eoc` flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;

When a single channel is sampled once, the CPU starts the `ADC_START` bit to sample the channel once, and `ADC_START` will be cleared by hardware directly, after the sampling is completed, SARADC stops sampling and waits for the CPU to initiate the next sampling.

Single-channel continuous sampling

Single-channel continuous sampling: the effective channel is only one single channel, i.e., only one bit of `ADC_CH_SEL` is high, and the SARADC sampling mode is continuous sampling.



DP32G030
Reference Manual

Figure 5-156 SARADC Single-Channel Continuous Sampling Schematic

The above figure shows the sequential sampling timing diagram for channel a:

- The CPU programs the ADC_START bit in the ADC_START register to be 1 at time T0, initiating the SARADC to sample channel a. The SARADC will then be able to sample channel a at time T0;

- At T_1 time, SARADC completes sampling once, outputs channel a sampling data `ch_a data0`, and the `eoc` flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- At the time of T_2 , SARADC completes sampling once, outputs channel a sampling data `ch_a data1`, and the `eoc` flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- At the time of T_3 , SARADC completes sampling once, outputs channel a sampling data `ch_a data2`, and the `eoc` flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- If the CPU expects to acquire N samples, it is necessary to stop the CPU from continuing to sample after $N-1$ samples have been acquired by programming the `ADC_START` bit in the `ADC_START` register to be zero. See T_4 above, clearing `ADC_START` to 0 will stop the SARADC from sampling after the current sampling cycle is completed;
- At the time of T_5 , the SARADC completes sampling once and outputs the channel a sampling data `ch_a data3` with the `eoc` flag pulled high. At this time, the CPU can acquire the last sample data;

When sampling a single channel continuously, after `ADC_START` is set to one, the SARADC samples this channel in a cycle until `ADC_START` is cleared by the program, after which the SARADC stops sampling after the current sample is completed.

Multi-Channel Single Sampling

Multi-Channel Single Sampling: The valid channels are [Reference Manual](#), i.e., ADC_CH_SEL has multiple bits as high, and the SARADC sampling mode is single sampling.

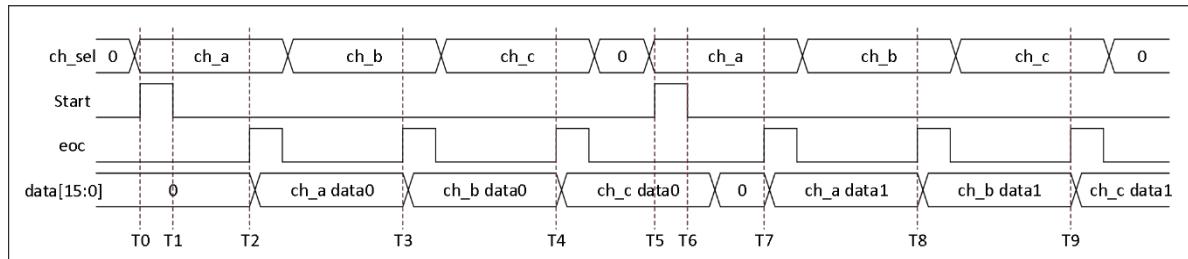


Figure 5-157 SARADC Multi-Channel Single Sampling Schematic

The above diagram shows the sequential sampling timing of channel a, channel b, and channel c, where channel number a<b<c, e.g., channel 0<channel 5<channel 8:

- At T0, the CPU programs the ADC_START bit in the ADC_START register to be 1, and the hardware initiates the SARADC to sample channels a, b, and c sequentially;
- The ADC_START register is cleared by the digital logic hardware at time T1;
- T2 time SARADC sampling channel a is completed, output channel a sampling data ch_a data0, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- T3 time SARADC sampling channel b is completed, output channel b sampling data ch_b data0, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- At T4 time, SARADC sampling channel c is completed, output channel c sampling data ch_c data0, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- To resample the sequence channel, the CPU needs to resample the ADC_START register again. The SARADC is initiated to re-sample channels a, b, and c at time T5 as shown above;
- The ADC_START register is cleared by the digital logic hardware at time T6;
- At T7 time, SARADC sampling channel a is completed, output channel a sampling data ch_a data1, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by

reading the corresponding status register;

- At t_8 time SARADC sampling channel b is completed, output channel b sampling data ch_b data1, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- At t_9 time, SARADC sampling channel c is completed, output channel c sampling data ch_c data1, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;

For multi-channel single sampling, starting ADC_START will sample all the selected channels once, ADC_START will be cleared directly and each channel will be sampled once, after all the sampling is completed, SARADC stops sampling and waits for the next sampling to be initiated.

Multi-Channel Continuous Sampling

Multi-Channel Continuous Sampling: The valid channels are multiple channels, i.e., ADC_CH_SEL has multiple high and SARADC sampling mode is continuous sampling.

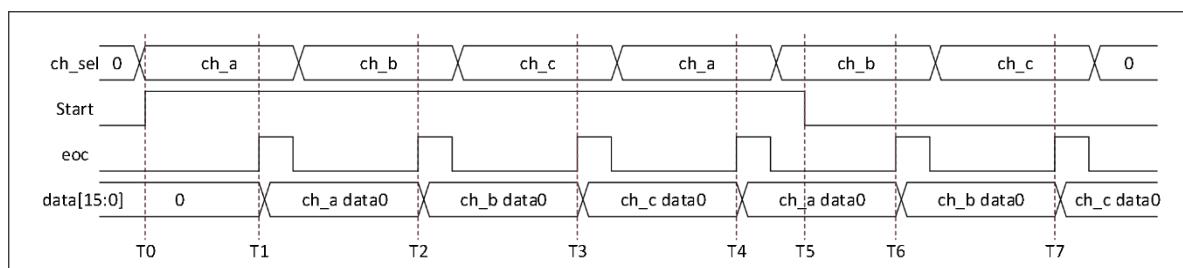


Figure 5-158 SARADC Multi-Channel Continuous Sampling Schematic

The above figure shows the sequential sampling timing diagram for channel a, channel b, and channel c, where channel number $a < b < c$, for example, channel 0< Channel 5 < Channel 8:

- The CPU programs the ADC_START bit in the ADC_START register to be 1 at the time of T0, initiating the SARADC's sequence of channel a, b, c Sampled;
- T1 time SARADC sampling channel a is completed, output channel a sampling data ch_a data0, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the

corresponding status register;

- T2 time SARADC sampling channel b is completed, output channel b sampling data ch_b data0, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;

- At T_3 time, SARADC sampling channel c is completed, output channel c sampling data $ch_c\ data0$, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- If the CPU expects to acquire N samples, it needs to stop the CPU from continuing sampling by programming the ADC_START bit in the ADC_START register to be 0 after $N-1$ samples have been acquired and before all channel data conversions are completed for the N th time. For example, if the ADC_START bit is cleared to 0 at time T_5 in the above figure, the hardware circuitry will stop the SARADC sampling after all valid channel conversions are completed in the current sampling cycle;
- At T_4 time, SARADC sampling channel a is completed, output channel a sampling data $ch_a\ data1$, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- T_6 time SARADC sampling channel b is completed, output channel b sampling data $ch_b\ data1$, eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;
- At T_7 time, SARADC sampling channel c is completed, output channel c sampling data $ch_c\ data1$, and the eoc flag is pulled high. At this time, the CPU can obtain the sampling data by reading the corresponding channel data register or FIFO data register, and obtain the sampling completion flag by reading the corresponding status register;

For multi-channel continuous sampling, after ADC_START is set to one, SARADC will cycle through all the selected channels until ADC_START is cleared by the

program, after which SARADC will stop sampling after [Reference Manual](#) of sampling is completed.

Channel Storage and FIFO

The SARADC is capable of storing the sample results in the Sample Data Register for each channel, or using the built-in FIFO with a depth of 16 bytes to store the sample results for all channels. The sample data storage location is set by the configuration register ADC_MEM_MD.

When using the channel registers, the channel register of each channel stores the sampled value of its own channel, and if the data is not read out in time, the new sampled data will overwrite the old data.

When using FIFO storage, the FIFO stores the sample values of the pass-through channel, and if the data is not read out in time, the new sample data will be lost.

Internal and External Sampling Clock Methods

The SARADC has two sampling modes: an internal sample clock mode and an external sample clock mode, which is selected through the configuration register ADC_IN_SMPL.

Different sampling clock methods require corresponding window settings. The window setting of internal sampling clock mode can be programmed through the IN_SMPL_WIN bit in the ADC_CFG register, and the configurable parameters are 1/3/5/7/9/11/13/15 with eight choices; the window setting of external sampling clock mode can be programmed through the SMPL_SETUP bit in the ADC_CFG register, and the configurable parameters are 1/2/4/8/16/32/64/128 with eight choices. SMPL_SETUP bit in the ADC_CFG register can be programmed by software, and the configurable parameters are 1/2/4/8/16/32/64/128.

CPU Trigger and External Signal Trigger

The SARADC provides selectable external signal trigger sources, and the trigger method can be selected through the configuration register ADC_TRIG. There are 12 external trigger source signals.

where ADC_EXTTRIG_SEL[3:0] are used to control PWM_PLUS0_TRIGGER[3:0] respectively; ADC_EXTTRIG_SEL[7:4] are used to control PWM_PLUS1_TRIGGER[3:0] respectively; ADC_EXTTRIG_SEL[9:8] are used to control TIMER_PLUS0_GOAL[1:0]; ADC_EXTTRIG_SEL[11:10] are used to control TIMER_PLUS1_GOAL[1:0] respectively;

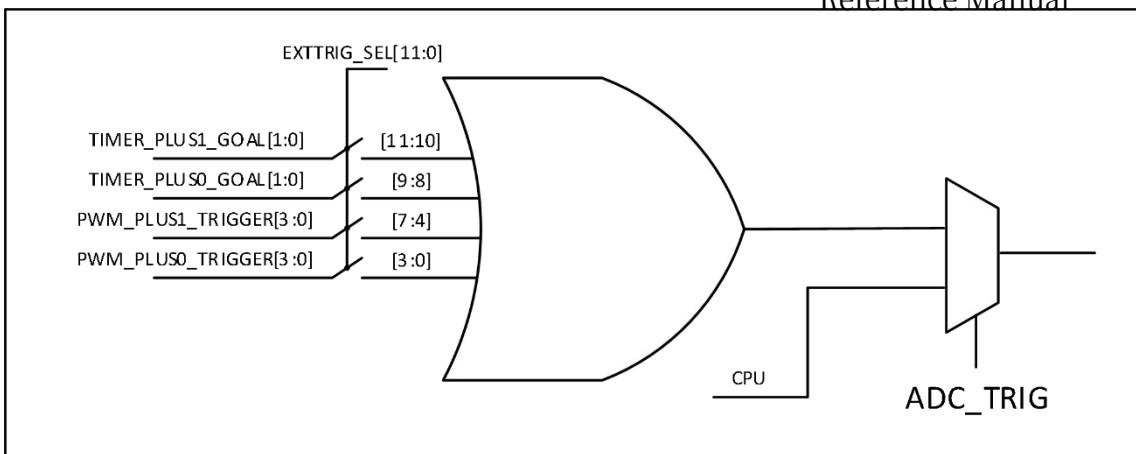


Figure 5-159 SARADC External Trigger Selection

disruptions

The SARADC provides 18 interrupts as shown below.

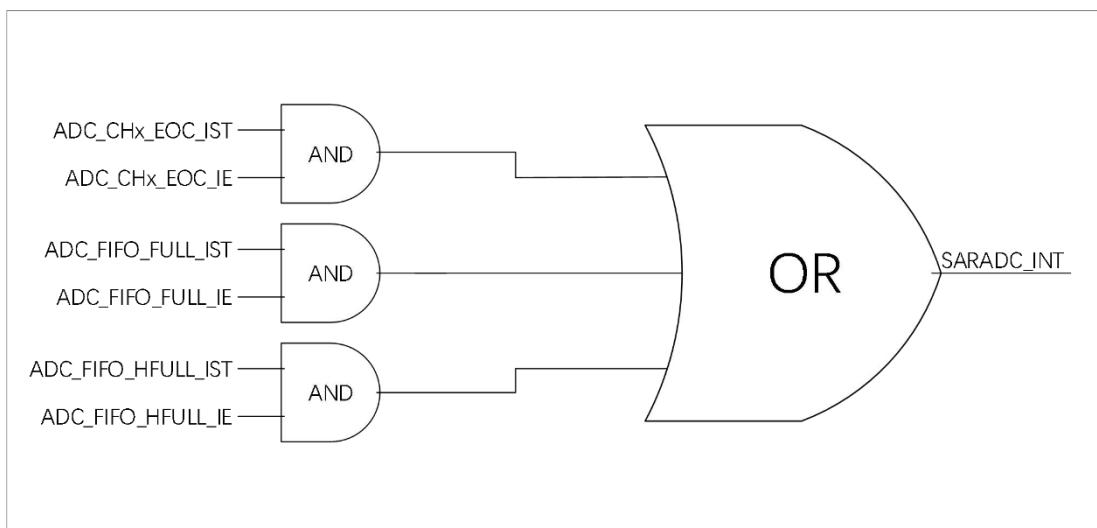


Figure 5-160 SARADC Interrupt Flag and Interrupt Signal Schematic

workflow

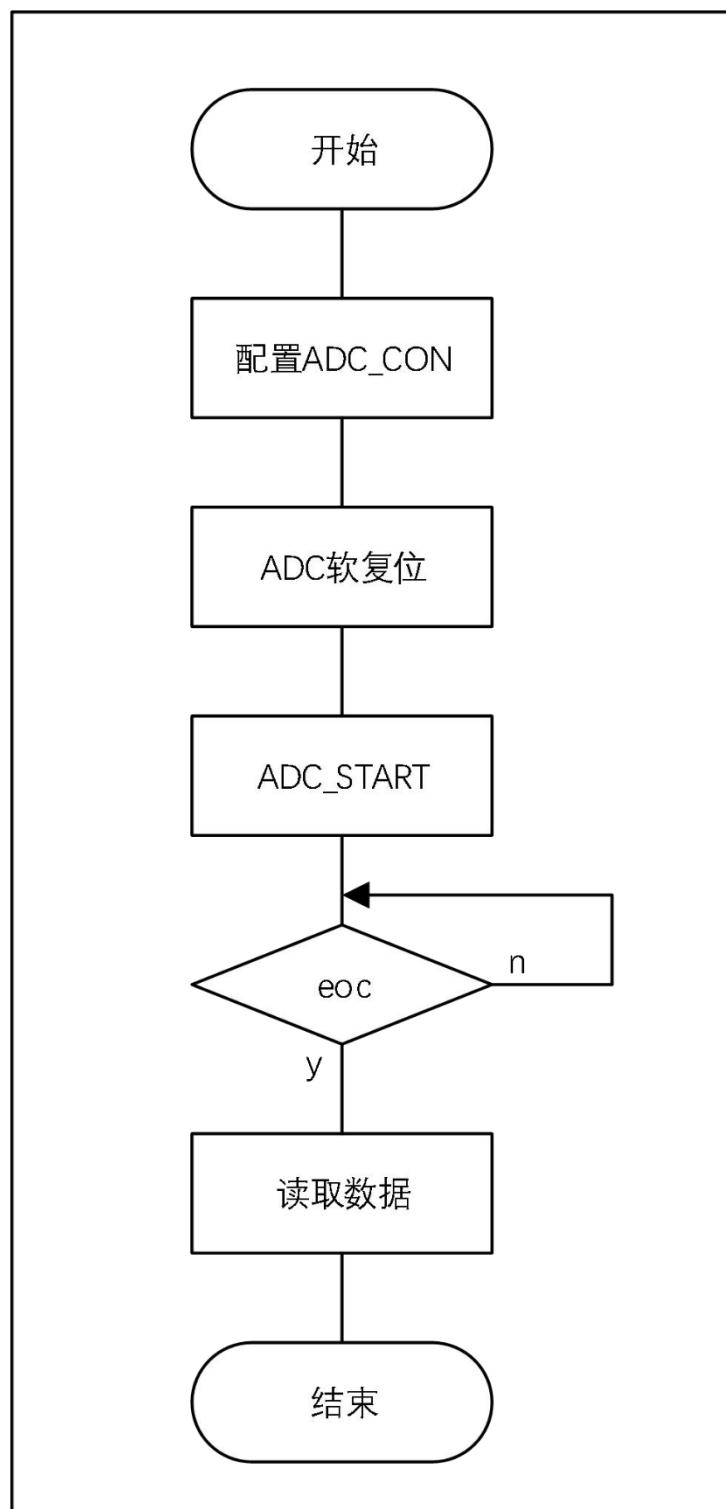


Figure 5-161 SARADC Operation Flowchart

- Turn on SARADC Module Clock
- Configure the pin for ADC functionality
- Configure the ADC's conversion clock division value
- Configuring the ADC's Conversion Channels
- Configure Sample Averaging
- Configure whether the transition mode is single or continuous mode
- Configure data storage as FIFO or channel
- Configure whether the sample clock is internal or external.
- Configure internal or external sampling windows
- Configure whether **KD** data is valid
- Configure whether **offset** data is valid
- Configure whether the channel transition completion interrupt is enabled
- Configure whether the FIFO full interrupt is enabled
- Configure whether the FIFO half-full interrupt is enabled or not.
- Initiate ADC soft reset
- Start ADC sampling
- Waiting for the end of sampling
- Read out the sampling data

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
SARADCBASE: 0x400BA000					
ADC_CFG	0x00	32	R/W	0x2100 000	ADC Configuration Register
ADC_START	0x04	32	R/W	0x00	ADC Startup Register
ADC_IE	0x08	32	R/W	0x00	ADC Interrupt Enable Register
ADC_IF	0x0c	32	RW	0x00	ADC Interrupt Status Register
ADC_CH0_STAT	0x10	32	R/W	0x00	ADC Channel 0 Status Register
ADC_CH0_DATA	0x14	32	R/W	0x00	ADC Channel 0 Data Register
ADC_CH1_STAT	0x18	32	R/W	0x00	ADC Channel 1 Status Register
ADC_CH1_DATA	0x1C	32	R/W	0x00	ADC Channel 1 Data Register
ADC_CH2_STAT	0x20	32	R/W	0x00	ADC Channel 2 Status Register
ADC_CH2_DATA	0x24	32	R/W	0x00	ADC Channel 2 Data Register
ADC_CH3_STAT	0x28	32	R/W	0x00	ADC Channel 3 Status Register
ADC_CH3_DATA	0x2C	32	R/W	0x00	ADC Channel 3 Data Register
ADC_CH4_STAT	0x30	32	R/W	0x00	ADC Channel 4 Status Register
ADC_CH4_DATA	0x34	32	R/W	0x00	ADC Channel 4 Data Register
ADC_CH5_STAT	0x38	32	R/W	0x00	ADC Channel 5 Status Register

ADC_CH5_DATA	0x3C	32	R/W	0x00	ADC Channel 5 Data Register
ADC_CH6_STAT	0x40	32	R/W	0x00	ADC Channel 6 Status Register
ADC_CH6_DATA	0x44	32	R/W	0x00	ADC Channel 6 Data Register
ADC_CH7_STAT	0x48	32	R/W	0x00	ADC Channel 7 Status Register
ADC_CH7_DATA	0x4C	32	R/W	0x00	ADC Channel 7 Data Register
ADC_CH8_STAT	0x50	32	R/W	0x00	ADC Channel 8 Status Register
ADC_CH8_DATA	0x54	32	R/W	0x00	ADC Channel 8 Data Register
ADC_CH9_ST	0x58	32	R/W	0x00	ADC Channel 9 Status Register
ADC_CH9_DATA	0x5C	32	R/W	0x00	ADC Channel 9 Data Register
ADC_CH10_ST	0x60	32	R/W	0x00	ADC Channel 10 Status Register
ADC_CH10_DATA	0x64	32	R/W	0x00	ADC Channel 10 Data Register
ADC_CH11_ST	0x68	32	R/W	0x00	ADC Channel 11 Status Register
ADC_CH11_DATA	0x6C	32	R/W	0x00	ADC Channel 11 Data Register
ADC_CH12_ST	0x70	32	R/W	0x00	ADC Channel 12 Status Register
ADC_CH12_DATA	0x74	32	R/W	0x00	ADC Channel 12 Data Register
ADC_CH13_ST	0x78	32	R/W	0x00	ADC Channel 13 Status Register
ADC_CH13_DATA	0x7C	32	R/W	0x00	ADC Channel 13 Data

					Register Reference Manual
ADC_CH14_ST	0x80	32	R/W	0x00	ADC Channel 14 Status Register
ADC_CH14_DATA	0x84	32	R/W	0x00	ADC Channel 14 Data Register
ADC_CH15_ST	0x88	32	R/W	0x00	ADC Channel 15 Status Register
ADC_CH15_DATA	0x8c	32	R/W	0x00	ADC Channel 15 Data Register
ADC_FIFO_STAT	0xa0	32	R/W	0x04	ADC FIFO Status Register
ADC_FIFO_DATA	0xa4	32	R/W	0x00	ADC FIFO Data Register
EXTTRIG_SEL	0xb0	32	R/W	0x00	External Signal Trigger ADC Select Register
ADC_CALIB_OFFSET	0xf0	32	R/W	0x00	ADC Calibration OFFSET Register
ADC_CALIB_KD	0xf4	32	R/W	0x00	ADC Calibration KD Register

register description

ADC_CFG register (0x00)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:30	RESERVED	R	0	reserved bit
29	DMA_EN	R/W	0	DMA Read FIFO Enable 0: CPU reads FIFO 1: DMA Read FIFO

28	ADC_TRIGGER	R/W	0	ADC Trigger Source Selection 0: Select CPU to trigger ADC sampling 1: Select an external signal to trigger ADC sampling
----	-------------	-----	---	---

27	ADC_EN	R/W	0	ADC enable bit 0: disable 1: Enable	Reference Manual
26:24	IN_SMPL_WI N	R/W	010	ADC Internal Sample Clock Method Sample Window Selection 000: Sample setup time is held for 1 ADC clock cycle 001: Sample build time is held for 3 ADC clock cycles 010: Sample build time is held for 5 ADC clock cycles 011: Sample build time is held for 7 ADC clock cycles 100: Sample build time is held for 9 ADC clock cycles 101: Sample setup time is held for 11 ADC clock cycles 110: Sample build time is held for 13 ADC clock cycles 111: Sample build time is held for 15 ADC clock cycles	
23	ADC_SMPL_C LK	R/W	0	ADC Sampling Mode Selection 1: Indicates that the ADC uses the internal sampling clock method. 0: Indicates that the ADC uses an external sampling clock.	
22	ADC_MEM_ MODE	R/W	0	ADC data storage method selection: 0: ADC data storage is in FIFO mode; 1: ADC data is stored in channel mode;	

				ADC External Sampling Clock Method Sampling Window Selection
21:19	SMPL_SETUP	R/W	010	<p>000: Sample setup time is held for 1 ADC clock cycle</p> <p>001: Sample setup time is held for 2 ADC clock cycles</p> <p>010: Sample build time is held for 4 ADC clock cycles</p> <p>011: Sample build time hold for 8 ADC clock cycles</p> <p>100: Sample setup time is held for 16 ADC clock cycles</p> <p>101: Sample build time is held for 32 ADC clock cycles</p> <p>110: Sample build time is held for 64 ADC clock cycles</p> <p>111: 128 ADC clock cycles of sample setup time retention</p>
18	CONT	R/W	0	<p>ADC Sampling Mode</p> <p>0: single sample</p> <p>1: Continuous sampling</p>
17:16	AVG	R/W	0	<p>One-start ADC Sample Averaging Configuration Register</p> <p>00: 1 sample averaged</p> <p>01: 2 samples averaged</p> <p>10: 4 samples averaged</p> <p>11: 8 samples averaged</p>

ADC Channel Select Register Reference Manual				
15:0	ADC_CH_SEL	R/W	0	Bit15-bit0 corresponds to channel 15-channel 0. If the corresponding bit is configured as 1, it means the channel is valid.

ADC_START register (0x04)

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
30:4	RESERVED	R	0	reserved bit
3	FIFO_CLR	R/W	0	FIFO Clear Enable 0: disable 1: enable write 1 clear FIFO
2	SOFT_RESET	R/W	0	ADC Soft Reset 0 Reset active After the ADC is enabled, the ADC must be reset once before startup
1	BUSY	R	0	ADC Operating Status 1 Indicates busy

				ADC start signal 0: disable 1: enable A write of 1 to this bit initiates a conversion. Can be used with ADC_CONT If ADC_CONT is in one-shot sampling mode, after this position 1, the valid channels will be polled sequentially for sample conversion and the converted data will be saved in the FIFO or register of the corresponding channel. The hardware will automatically clear the data after the conversion is completed. If ADC_CONT is in continuous sampling mode, the position 1 means start ADC conversion, and clearing it means stop ADC conversion. After starting ADC conversion, the valid channels will be polled sequentially for sampling and conversion, and the converted data will be stored in the FIFO or register of the corresponding channel. After each conversion, it judges whether the bit is 1, if it is 1, the conversion will continue, if it is 0, the conversion will stop.	Reference Manual
0	START	R/W	0		

ADC_IE register (0x08)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive

30:18	RESERVED	R	0	reserved bit
17	ADC_FIFO_HFULL_IE	R/W	0	ADC FIFO Half Full Interrupt Enable 0: disable 1: Enable
16	ADC_FIFO_FULL_IE	R/W	0	ADC FIFO full interrupt enable 0: disable 1: Enable
15:0	ADC_CHx_EOC_IE	R/W	0	ADC Channel x Data Conversion Complete Interrupt Enable

ADC_IF register (0x0C)

bitfield d (math.)	name (of a thing)	typo logy	reset value	descriptive
30:18	RESERVED	R	0	reserved bit
17	ADC_FIFO_HF ULL_IF	R/W	0	ADC FIFO half-full interrupt status write 1 clear
16	ADC_FIFO_FU LL_IF	R/W	0	ADC FIFO Full Interrupt Status Write 1 Zero Clear
15:0	ADC_CHx_EO C_IST	R/W	0	ADC channel x data conversion complete interrupt status write 1 clear

ADC_CHx_STAT Register

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
30:1	RESERVED	R	0	reserved bit
0	ADC_CH_E OC	R	0	ADC Channel x Data Conversion Completion Flag 1: Indicates that the ADC has completed one sample conversion for channel x. The ADC can be cleared by writing 1 to the bit corresponding to ADC_IF.

ADC_CHx_DATA Register

DP32G030
Reference Manual

bitfield (math.)	name (of a thing)	typolo gy	reset value	descriptive
30:16	RESERVED	R	0	reserved bit
15:12	ADC_CH_ NUM	R	0	Channel number corresponding to ADC data
11:0	ADC_CH_ DATA	R	0	ADC Channel x Data Register Note: After the overflow, the data converted again will overwrite the old data

ADC_FIFO_STAT register (0xA0)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
30:8	RESERVED	R	0	reserved bit
7:4	ADC_FIFO_LEVEL	R	0	ADC Data FIFO Water Level 0000: When it is not full, it means the FIFO has 0 data, when it is full, it means the FIFO has 16 data; 0001: Indicates that the FIFO has 1 data; 0010: Indicates that the FIFO has 2 data; 0011: Indicates that the FIFO has 3 data; 0100: Indicates that the FIFO has 4 data; 0101: Indicates that the FIFO has 5 data; 0110: Indicates that the FIFO has 6 data; 0111: Indicates that the FIFO has 7 data; 1000: Indicates that the FIFO has 8 data; 1001: Indicates that the FIFO has 9 data; 1010: Indicates that the FIFO has 10 data; 1011: Indicates that the FIFO has 11 data; 1100: Indicates that the FIFO has 12 data; 1101: Indicates that the FIFO has 13 data; 1110: Indicates that the FIFO has 14 data; 1111: Indicates that the FIFO has 15 data;
3	RESERVED	R	0	reserved bit
2	ADC_FIFO_EMPTY	R	1	ADC Data FIFO Empty Flag 1: Indicates that the FIFO is empty 0: Indicates that the FIFO is not empty
1	ADC_FIFO_HFULL	R	0	ADC Data FIFO Half Full Flag 1: Indicates that the FIFO is half full 0: Indicates that the FIFO is not half full

0	ADC_FIFO_FULL	R	0	ADC data FIFO full flag 1: Indicates that the FIFO is full 0: Indicates that the FIFO is not full
---	---------------	---	---	---

ADC_FIFO_DATA register (0xA4)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
30:16	RESERVED	R	0	reserved bit
15:12	ADC_FIFO_NUM	R	0	Channel number corresponding to ADC data
11:0	ADC_FIFO_DATA	R	0	ADC Data FIFO Register Note: After an overflow, data that is converted again is discarded

ADC_EXTTRIG_SEL register (0xB0)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
30:12	RESERVED	R	0	reserved bit
11:0	EXTTRIG_SEL	R/W	0	External Trigger ADC Sample Signal Selection Control Bit11-bit0: Indicates that they are used to control exttrig_in[11:0], respectively. Whether to trigger the control bit of ADC 0 means not valid; 1 means valid

ADC_CALIB_OFFSET register (0xF0)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
30:17	RESERVED	R	0	reserved bit

DP32G030

16	OFFSET_VALID	R/W	0	Reference Manual OFFSET Whether the data is valid or not
15:8	RESERVED	R	0	reserved bit
7:0	OFFSET	R/W	0	OFFSET value for ADC data calibration. The calculated OFFSET needs to be stored in this register. Used for calibration when using ADC.

ADC_CALIB_KD register (0xF4)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
30:17	RESERVED	R	0	reserved bit
16	KD_VALID	R/W	0	Validity of KD data
15:10	RESERVED	R	0	reserved bit
9:0	KD	R/W	0	ADC data calibrates the fractional part of the K value. The fractional part of the calculated K value needs to be stored in this register. Used for calibration when using ADC. If K is greater than 1, the sign bit is 1. If K is less than 1, the sign bit is 0.

5.20 Comparator (COMP)

5.20.1 summarize

This chip provides three general-purpose comparator units, each of which can be independently configured for use. Specific IO ports are used as inputs as shown in the block diagram of the structure of this module. They have a variety of functions including rail-to-rail comparators, independently programmable configuration of hysteresis voltages and independently programmable configuration of output filtering functions.

5.20.2 characterization

- Rail-to-Rail Comparator
- Each comparator can be programmed independently to configure hysteresis voltages: 0mv hysteresis, 24mv hysteresis, 40mv hysteresis, and 60mv hysteresis in four stops.
- Each comparator can be programmed independently to configure the output filtering function: no filtering, 2 system clock cycles filtering, 4 system clock cycles filtering and 8 system clock cycles filtering.
- Each comparator has interrupt generation capability
- Can be combined into a window comparator

5.20.3 Block Diagram of Module Structure

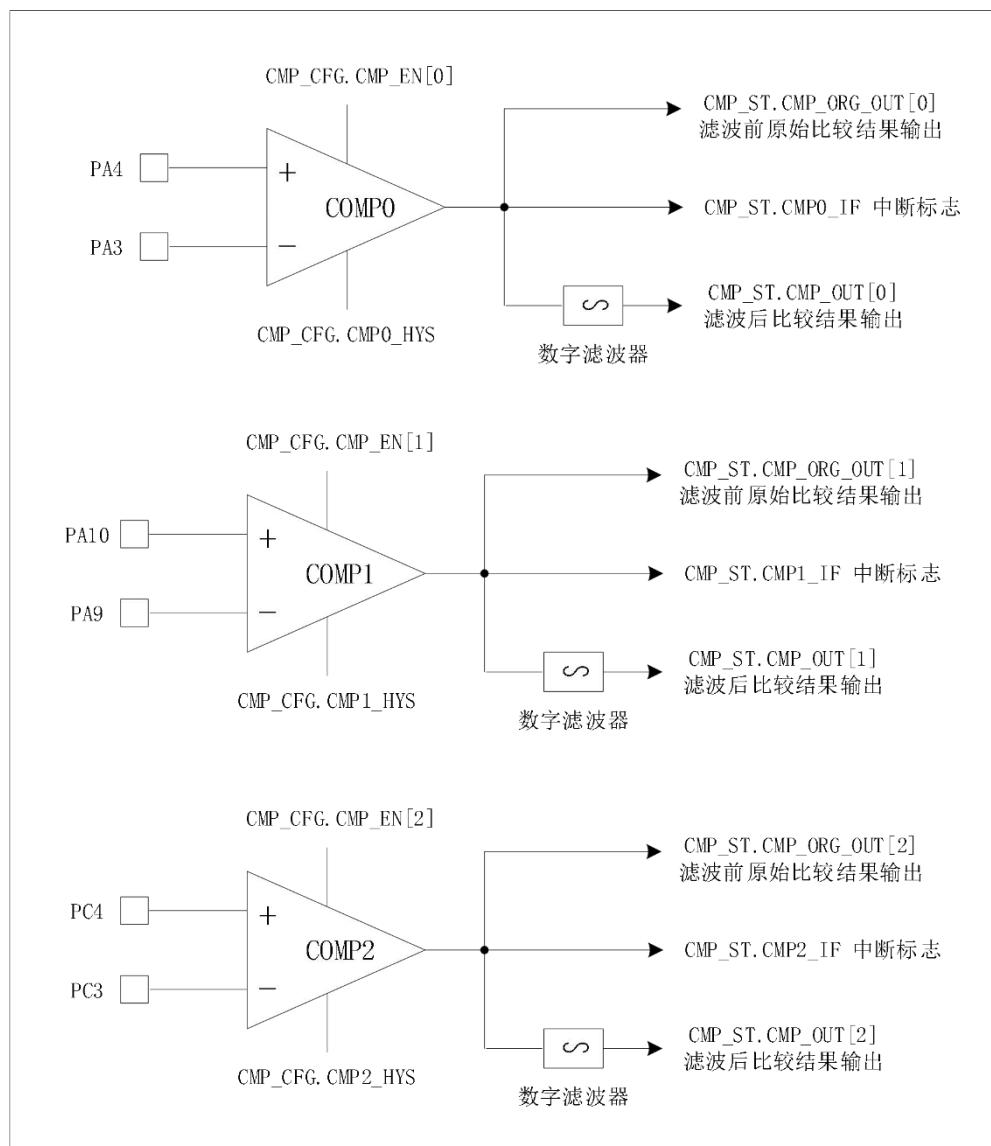


Figure 5-162 Comparator Block Diagram

As shown above:

The VP side of Comparator 0 is input from the PA4 pin and the VN side is input from the PA3 pin; and the comparator is directly programmable from its separate enable register, CMP_CFG.CMP_EN[0].

The VP side of Comparator 1 is input from the PA10 pin and the VN side is input from the PA9 pin; and the comparator is directly programmable from its separate enable register, CMP_CFG.CMP_EN[1].

The VP side of Comparator 2 is input from the PC4 pin and the VN side is input from the PC3 pin; and the comparator is directly programmable from its separate enable register, CMP_CFG.CMP_EN[2].

5.20.4 Functional Description

Comparator input pins and internal output signals

Each of the three independent comparators has independent inputs and outputs.

The positive end of comparator 0 (COMP0) is input through PA4 and the negative end is input through PA3. The raw comparison result can be queried through the CMP_ORG_OUT[0] bit of the CMP_ST register, the filtered comparison result can be queried through the CMP_OUT[0] bit of the CMP_ST register, and the interrupt flag can be queried through the CMP_IF[0] bit of the CMP_ST register.

The positive end of comparator 1 (COMP1) is input through PA10 and the negative end is input through PA9. The raw comparison result can be queried through the CMP_ORG_OUT[1] bit of the CMP_ST register, the filtered comparison result can be queried through the CMP_OUT[1] bit of the CMP_ST register, and the interrupt flag can be queried through the CMP_IF[1] bit of the CMP_ST register

The positive side of Comparator 2 (COMP2) is input through PC4 and the negative side is input through PC3. The raw comparison result can be queried through the CMP_ORG_OUT[2] bit of the CMP_ST register, the filtered comparison result can be queried through the CMP_OUT[2] bit of the CMP_ST register, the interrupt flag can be queried through the CMP_IF[2] bit of the CMP_ST register

hysteresis

The hysteresis function of the comparators can be configured independently

via the respective hysteresis registers `CMPx_HYS` and is used to reduce output errors due to disturbances.

The programmable hysteresis register allows four hysteresis stops to be selected: 0mV, 24mV, 40mV, and 60mV. The comparator hysteresis is shown below:

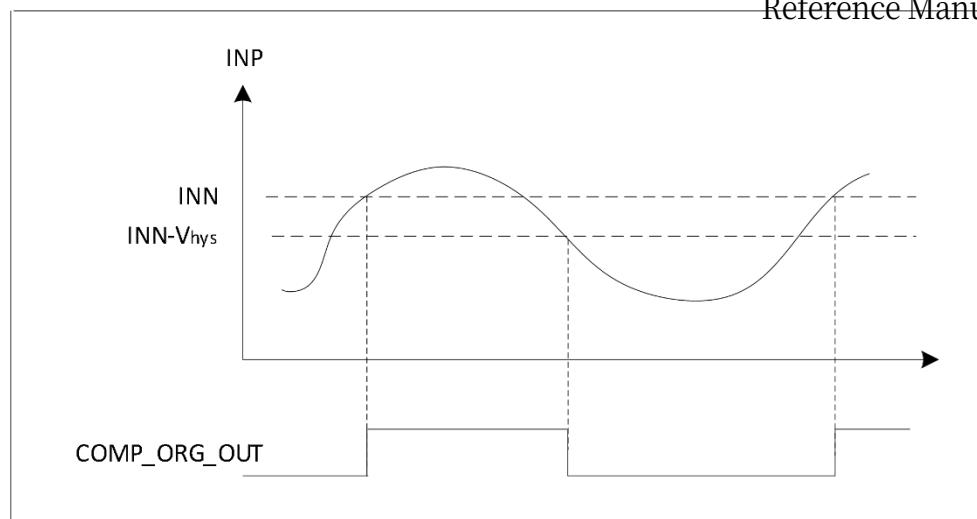


Figure 5-163 Comparator Hysteresis Schematic

filtering radio waves (i.e. pick out one frequency)

In order to improve the reliability of the comparator and reduce the misjudgment due to interference, the output of the comparator is digitally filtered

Wave.

The programmable filter register `CMP_CFG.CMP_FILTER` allows the selection of four filter stops: 0 system clock cycle filter, 2 system clock cycle filter, 4 system clock cycle filter, and 8 system clock cycle filter.

A schematic diagram of the filtering function used to eliminate burrs is shown below:

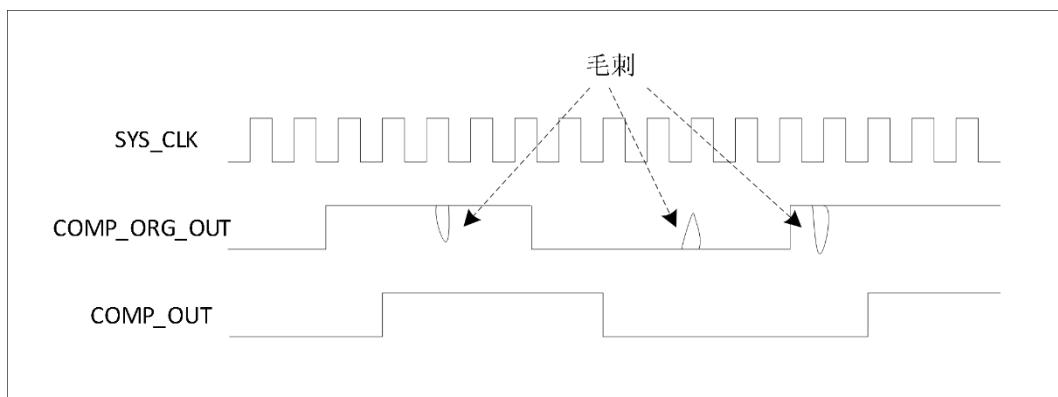


Figure 5-164 Comparator Filtering Function Schematic for Burr Elimination

This configuration is valid for all comparators.

disruptions

Each comparator has its own two comparator output status flags: a status flag indicating a change from 0 to 1, and a status flag indicating a change from 0 to 1. flag; the other is a change state flag that indicates a change from 1 to 0.

The three independent comparators are controlled by their respective interrupt enables and together output to the CPU's interrupt controller to generate the chip's comparator interrupt source.

The interrupt flag and interrupt schematic are shown below:

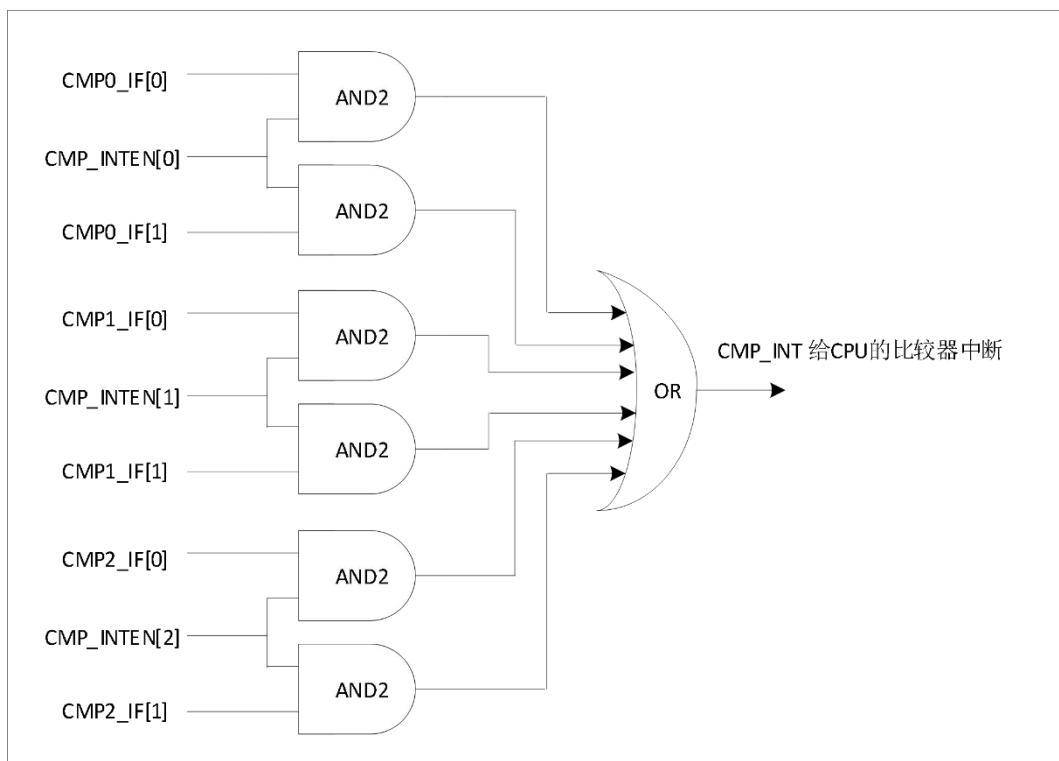


Figure 5-165 Comparator Interrupt Flag and Interrupt Diagram

register map

name (of a thing)	offset	typolo	reset	descriptive
-------------------	--------	--------	-------	-------------

DP32G030

		gy	value	
COMPARATORBASE: 0x40000000				

CMP_CFG	0x120	R/W	0x200000	Reference Manual Comparator Configuration Register
CMP_ST	0x124	R/W	0x00	Comparator Status Register

register description

CMP_CFG register (0x120)

bitfield (math.)	name (of a thing)	typolog y	reset value	descriptive
31:26	RESERVED	R	0	reserved bit
25:24	CMP_FILTER	R/W	10	Comparator Output Filter Control 00: No filtering 01: 2 system clock cycles of filtering 10: 4 system clock cycles of filtering 11: 8 system clock cycles of filtering
23:22	RESERVED	R	0	reserved bit
21:20	CMP2_HYS	R/W	0	CMP2 hysteresis selection 00:24mv 01:40mv 10:60mv 11:0mv
19:18	CMP1_HYS	R/W	0	CMP1 Hysteresis Selection 00:24mv 01:40mv 10:60mv 11:0mv
17:16	CMP0_HYS	R/W	0	CMP0 Hysteresis Selection 00:24mv 01:40mv 10:60mv 11:0mv
15:11	RESERVED	R	0	reserved bit
10:8	CMP_INTEN	R/W	0	Comparator interrupt enable control bit 1: Enabling 0: Closed Bit10-bit8 controls CMP2-CMP0 respectively.
7:3	RESERVED	R	0	reserved bit

Reference Manual Comparator Enable Control Bit				
2:0	CMP_EN	R/W	0	1: Enabling 0: Closed Bit2-bit0 controls CMP2-CMP0 respectively.

CMP_ST register (0x124)

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:22	RESERVED	R	0	reserved bit
21:20	CMP2_IF	R/W	0	Comparator 2 interrupt flag A Bit21 of 1 indicates that the output of comparator 2 changes from 0 to 1. A Bit20 of 1 indicates that the output of Comparator 2 changes from 1 to 0. Both are written 1 and cleared
19:18	CMP1_IF	R/W	0	Comparator 1 interrupt flag A Bit19 of 1 indicates a change from 0 to 1 on the Comparator 1 output. A Bit18 of 1 indicates a change from 1 to 0 on the Comparator 1 output. Both are written 1 and cleared
17:16	CMP0_IF	R/W	0	Comparator 0 interrupt flag A Bit17 of 1 indicates a change from 0 to 1 on the Comparator 0 output. A Bit16 of 1 indicates a change from 1 to 0 on the Comparator 0 output. Both are written 1 and cleared
15:11	RESERVED	R	0	reserved bit
10:8	CMP_ORG_ OUT	R	0	Comparator result output (before filtering) 1: P-terminal > N-terminal 0: P-terminal < N-terminal Bit10~bit8 represent the output of CMP2~CMP0 respectively.
7:3	RESERVED	R	0	reserved bit

2:0	CMP_OUT	R	0	Comparator result output 1: P-terminal > N-terminal 0: P-terminal < N-terminal Bit2~bit0 represent the output of CMP2~CMP0 respectively.
-----	---------	---	---	---

5.21 Operational amplifiers (OPAMP)

5.21.1 summarize

This chip provides two independent operational amplifier units, each of which can be independently configured for use. Each operational amplifier

The outputs of the amplifiers are connected to specific ADC input channels inside the chip, enabling direct ADC sampling measurements. Enables

Use specific IO ports as inputs as shown in the block diagram of this module.

5.21.2 characterization

- Rail-to-rail input/output
- Positive/Negative Input
- 2 Operational Amplifiers
- 3MHz bandwidth
- Low Offset Voltage
- Output signals can optionally be configured to the ADC channel for direct acquisition

5.21.3 Module Structure Diagram

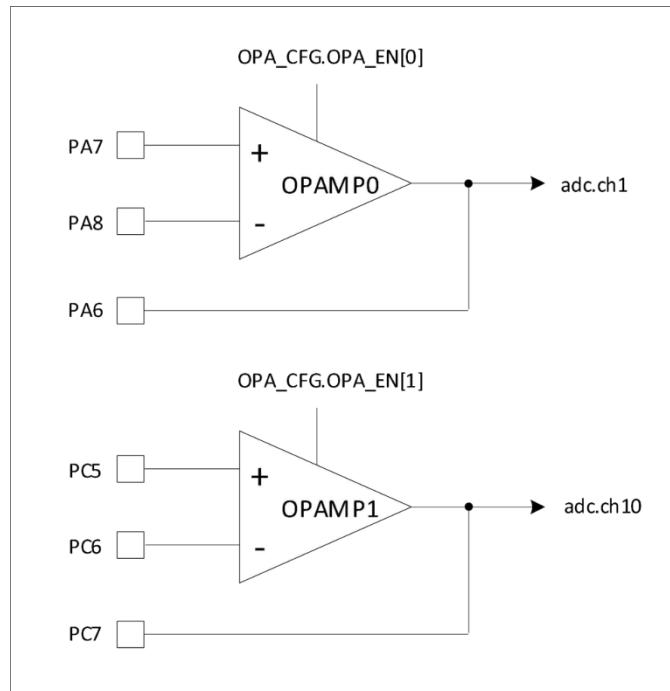


Figure 5-166 Operational Amplifier Block Diagram

As shown above:

The VP terminal of OPAMP0 is inputted from PA7, the VN terminal is inputted from PA8, and the output can be directly output to PA6 or to channel 1 of the SARADC; and the op amp can be directly programmed by its independent enable register OPA_CFG.OPA_EN[0].

The VP terminal of OPAMP0 is inputted from PC5, the VN terminal is inputted from PC6, and the output can be directly output to PC7 or to channel 10 of the SARADC; and this operational amplifier can be directly programmed by its independent enable register OPA_CFG.OPA_EN[1].

register map

name (of a thing)	offset	typology	reset value	descriptive

OPA_CFG	0x140	R/W	0x00	Reference Manual Operational Amplifier Configuration Register
---------	-------	-----	------	---

register description

OPA_CFG

bitfield (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:2	RESERVED	R	0	reserved bit
1:0	OPA_EN	R	0	OPA Enable Control Bit 1: Enabling 0: Closed Bit1-bit0 controls OPA1-OPA0 respectively.

5.22 FLASH Controller (FLASHCTRL)

5.22.1 summarize

The chip has a 64K-byte embedded FLASH memory for storing application programs and parameter data related to chip performance tuning. FLASH has two areas: one is the main array area for storing application programs or data, and the other is the NVR area for storing parameter data related to the performance tuning of the chip, etc. The NVR area is used for storing application programs or data.

The controller circuit for controlling the operation of the internal FLASH is located between the CPU and the FLASH, and read, program, and erase operations of the FLASH are realized through the read, write, and erase operations of the CPU.

5.22.2 characterization

- Supports 64K bytes of application storage
- Supports entering and exiting FLASH low-power mode
- The read rate is configurable. When the system clock frequency is less than or equal to 56MHz, the read rate is 1 system clock cycle wait; when the system clock frequency is greater than 56MHz and less than or equal to 84MHz, the read rate is 2 system clock cycles wait
- Configuration area (NVR area) domain total 4 sectors, each sector 512 bytes, the first sector is stored in the FLASH configuration information, the user can not be operated. The other 3 sectors can be erased, read or written.
- Operation modes support read, program, and sector erase operations.
- Operation lock can be configured to prohibit programming and erasing operations on FLASH to protect FLASH data from being modified.

- Supports address masking, which can be configured to 2¹² bytes, 8K bytes.
- Each sector is 512 bytes in size and supports sector erase operations.
- Supports sequential programming of up to 64 words (256 bytes) within the same sector

5.22.3 Block Diagram of Module Structure

The FLASH controller includes AHB slave interface, APB slave interface, FLASH control registers, and FLASH initialization controller,

It consists of a FLASH operation controller and on-chip FLASH.

The schematic block diagram of the FLASH memory controller structure is shown below:

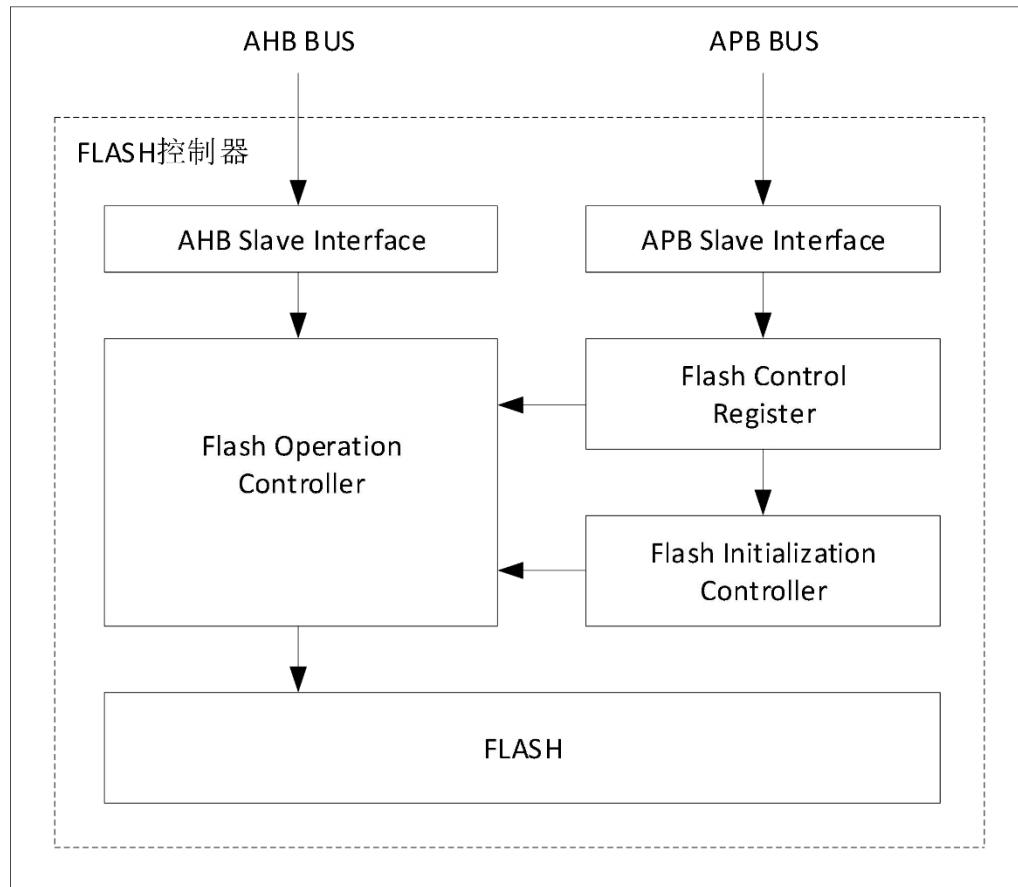


Figure 5-167 FLASH Controller Structure Diagram

AHB Slave Interface: Used for CPU instruction and data reading.

APB Slave Interface: Used for CPU to program and configure FLASH controller related registers.

FLASH control registers: used to realize the logic of each function register and the logic of related state generation in each operation mode required in the FLASH

DP32G030
Reference Manual

controller module.

FLASH Initialization Controller: When the chip is powered on or reset, the FLASH Initialization Controller circuit will start to access the FLASH automatically and read the FLASH parameter data and load the user configuration information into the FLASH registers to complete the initialization of the FLASH. After the initialization of the FLASH is completed, the FLASH can start to perform normal operations.

FLASH operation control: According to the functional requirements of CPU to operate FLASH via bus, it realizes the relevant operations on FLASH, including FLASH erase, FLASH programming and FLASH reading, and generates corresponding status signals to be fed back to the control register logic.

On-chip FLASH Memory: The on-chip FLASH memory is mainly used for storing user programs and parameters, which includes a 2K-byte configuration area and a 64K-byte program area, each sector size is 512 bytes, with a total of 128 sectors, and the minimum programmable data bit size is 32 bits.

5.22.4 Function Description

Low Power Mode

Mode

FLASH supports to enter the low power mode, after entering the low power mode, the working current of the chip can be greatly reduced. To enter the low power mode, configure the DEEP_PD position 1 in the FLASH_CFG register in RAM. To exit the low power mode, configure the DEEP_PD position 0 in the FLASH_CFG register in RAM.

Read Rate Configuration

When the system clock is configured to different frequencies, it is necessary to match the corresponding read rate, otherwise there will be problems with FLASH reading. When the system clock frequency is less than or equal to 56MHz, only 1 system clock wait can be used, that is, configure the READ_MD position 0 in the FLASH_CFG register in RAM; when the system clock frequency is greater than 56MHz and less than or equal to 84MHz, 2 system clock wait are needed, that is, configure the READ_MD position 1 in the FLASH_CFG register in RAM. When the system clock frequency is greater than 56MHz and less than 84MHz, 2 system clock waits are required, that is, configure the READ_MD in FLASH_CFG register in RAM to be 1.

NVR zone and MAIN zone selection

The TRIM values such as the chip's voltage and clock are stored in the NVR area, and the program needs to transfer the TRIM values from the NVR area during initialization.

The values are read and written to the appropriate registers.

The NVR area can be accessed by configuring NVR_SEL position 1 in the FLASH_CFG register in RAM. The NVR area consists of 4 sectors, each sector is 512 bytes, the first sector stores the FLASH configuration information, and the user can not. The next three sectors can be erased, read and written by the user.

operating mode

The operation modes of FLASH support the following three methods: read operation, programmed operation, and sector erase operation. Configuration of the operation mode must be performed in RAM.

Programming operation: Program the data to be programmed into the address starting from the address in FLASH_ADDR, and program up to half a sector (256 bytes) at a time, and can only be programmed in half a sector.

Sector erase operation: Each sector is 512 bytes and the address of the corresponding erased sector is specified by FLASH_ADDR. After each programming or sector erase operation, the operation mode must be configured back to read operation.

Read operation: Read operation of FLASH can be realized by directly accessing the FLASH address by CPU.

FLASH operation address

FLASH_ADDR register: address register for programming/sector erase operation.

Starting address for initiating the programming operation, **Reference Manual**. The maximum programming data for each consecutive programming is half a sector, i.e., the maximum programming data is 64 words, and programming can only be performed in half a sector.

When the sector erase operation is started, bit7-bit13 indicates the sector number to be erased; that is, 1 sector is 512 Byte, and the address is 512-byte aligned.

Sector Size

Each sector of FLASH of this chip is 512Byte, the main program area (MAIN area) is 64KBytes, that is, 128 sectors, and the configuration area (NVR area) is 2KBytes, that is, 4 sectors. The chip supports sector erase, programming operation can only program up to half a sector (256 bytes), and can only be programmed in half a sector.

START operation start control

When the START bit in the FLASH_START register is set to 1, FLASH will start the command operation configured in the MODE register (Read, Sector Erase, Programming) and the bit will be cleared automatically when the current command operation is completed.

operation lock

Write 0x55 to FLASH_LOCK register, then operation start is locked and START cannot be written 1.
Used for protection of FLASH
of misuse, the power-up defaults to a locked state.

Writing 0xAA to the FLASH_UNLOCK register causes the operation start to be unlocked and START can be written 1.

MASK Function

The MASK address can be configured as unmasked, 2KBytes, 4KBytes, and 8KBytes.

The MASK selector lock can be configured so that when the MASK_LOCK bit is
No. 369 Page 351
of 432

configured to 1, MASK_SEL cannot be modified. When the Reference Manual is configured to 0, MASK_SEL can be modified.

Erase Time and Program Time Configuration

Depending on the system clock frequency, the erase time and programming time need to be configured to meet the corresponding requirements of the FLASH timing.

The specific configuration can be found in the corresponding register descriptions.

FLASH Initialization Busy Flag

When the INIT_BUSY bit of the FLASH_ST register is 1, it indicates that flash initialization is in progress, and 0 indicates that initialization is complete.

After power-up, reset and exit from low-power mode, the program needs to judge this bit, and only after the FLASH initialization is completed, normal operation can be performed.

Controller Busy Flag

When the BUSY bit of the FLASH_ST register is 1, it indicates that the controller is in the process of the current command operation, and is 0.

When the current operation is completed, the controller is in the READY state, waiting for a new command to be executed.

Programming Data Cache Register Empty Status Flag

When the PROG_BUF_EMPTY bit of FLASH_ST register is 1, it means that the programming data cache register is empty and the next word data can be written; when it is 0, it means that the programming data cache register is not empty and the next word data cannot be written. If the next word data is written during the current word programming process (16us), continuous programming will be

DP32G030

performed, otherwise, the continuous programming operation is terminated.
Refer to the Manual.

FLASH Program Initialization Flow

Configure the FLASH to enter normal operation mode, configure it to read mode, configure the read rate, erase time, and programming time parameters according to the system clock, and lock the FLASH to prohibit the programming operation of the FLASH.

These operations must be performed in RAM.

FLASH Sector Erase Procedure

- 1) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 2) Configure the mode as a sector erase operation;
- 3) Write sector erase start address in words;
- 4) Configure FLASH to unlock and initiate the START command;
- 5) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 6) Configure the mode as read operation, lock the FLASH, and complete the sector erase operation. The sector erase operation flow diagram is shown below:

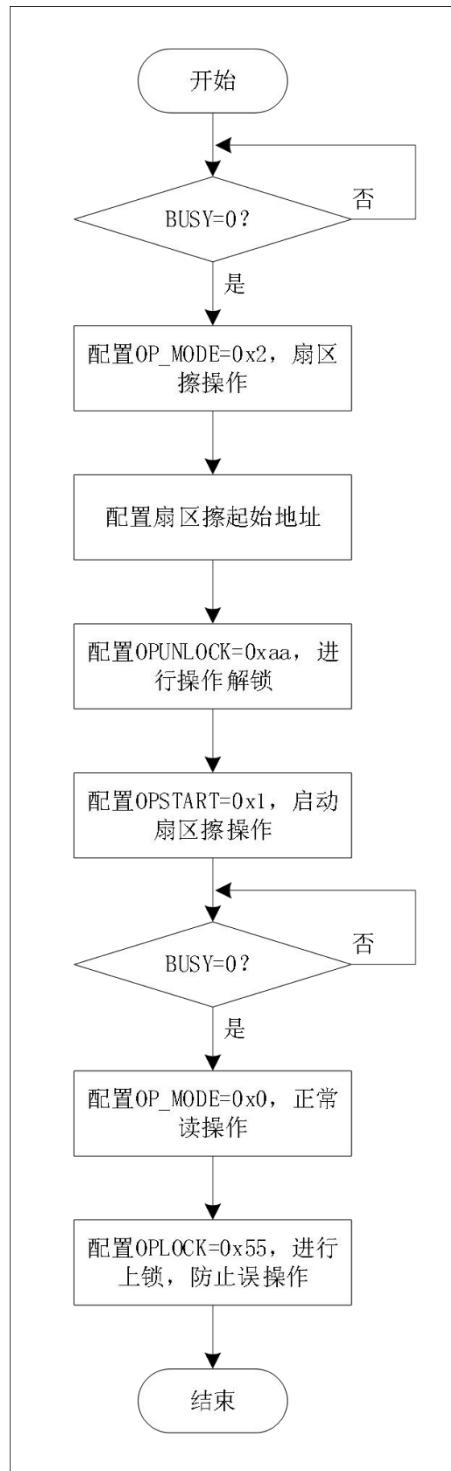


Figure 5-168 Sector Erase Operation Flowchart

These operations must be performed in RAM.

FLASH Single Character Programming Procedure

- 1) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 2) Configuration mode is programmed operation;
- 3) Write to the programmed address in words;
- 4) Put the data to be programmed into the data register;
- 5) Configure FLASH to unlock and initiate the START command;
- 6) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 7) Configuration mode is read operation, and the FLASH is locked to complete the programming operation for the word data. The programming operation flow is shown below:

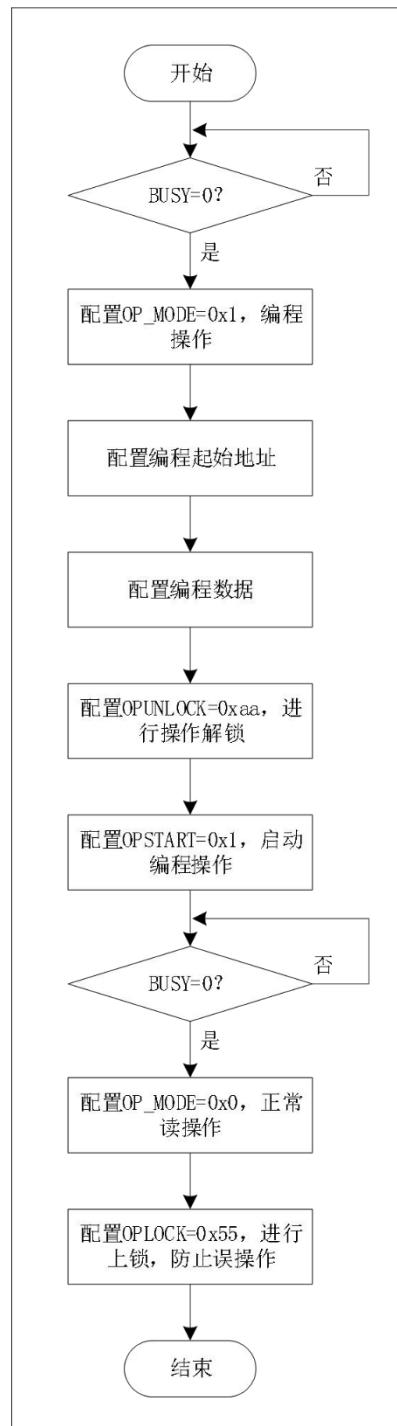


Figure 5-169 Single-word Programming Operation Flowchart

These operations must be performed in RAM.

FLASH Multi-Word Continuous Programming Procedure

- 1) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 2) Configuration mode is programmed operation;
- 3) Write to the programmed address in words;
- 4) Writes the first word of the programmed data to the data register;
- 5) Configure FLASH to unlock and initiate the START command;
- 6) If there is still data to be programmed, judge whether the PROG_BUF_EMPTY bit is 0. If it is 0, the next word to be programmed can be written (if it is 1, wait and cannot write) until all the data to be programmed have been written;
- 7) Queries the FLASH controller busy flag and waits for the controller to be in the READY state;
- 8) Configuration mode is read operation, FLASH lock, complete all the data to be programmed for this operation, in which the maximum amount of multi-word continuous programming data is 64 words, and can only be programmed in half a sector.

A schematic diagram of the programming operation flow is shown below:

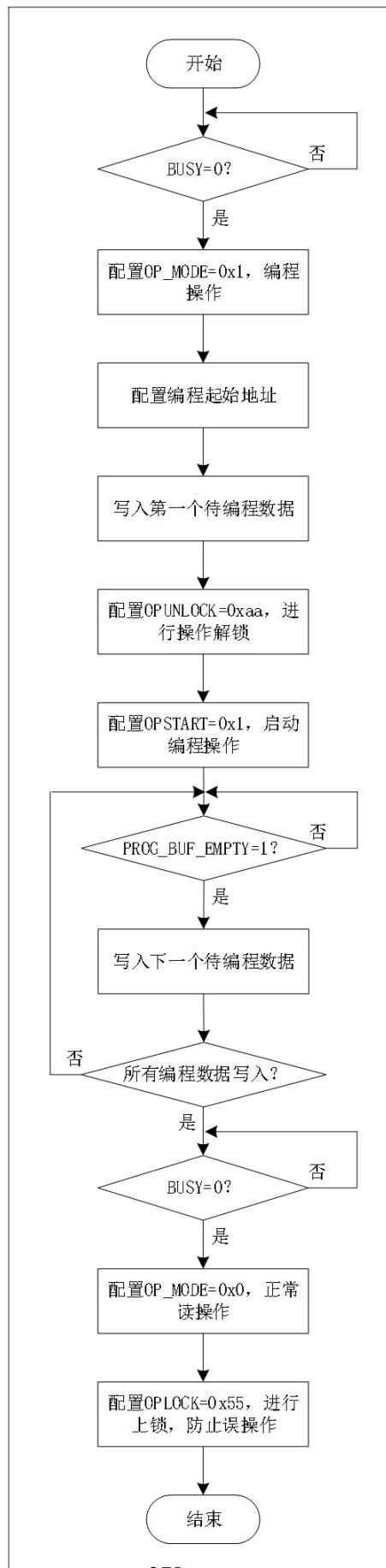


Figure 5-170 Multi-word Continuous Programming Operation Flowchart

These operations must be performed in RAM.

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
flash	_ctrlbase: 0x4006f000				
FLASH_CFG	0x00	32	R/W	0x80000000	configuration register
FLASH_ADDR	0x04	32	R/W	0x0	Address Configuration Register
FLASH_WDATA	0x08	32	W	0x0	Write Data Register
FLASH_START	0x10	32	R/W	0x0	Operation startup register
FLASH_ST	0x14	32	R	0x5	status register
FLASH_LOCK	0x18	32	W	0x0	Operation Lock Control Register
FLASH_UNLOCK	0x1c	32	W	0x0	Operation Unlock Control Register
FLASH_MASK	0x20	32	R/W	0x2	Mask Control Register
FLASH_ERASETIME	0x24	32	R/W	0x7532a31b	Erase Time Parameter Configuration Register
FLASH_PROGTIME	0x28	32	R/W	0x1f4360	Programming Time Parameter Configuration Register

register description

FLASH_CFG register (0x00)

bitfield	name (of a	typol	reset	descriptive
----------	------------	-------	-------	-------------

d (math.)	thing)	ogy	value	
31	DEEP_PD	R/W	1	<p>Configure FLASH to enter low-power mode</p> <p>1: Enter low power mode</p> <p>0: Normal operating mode</p> <p>Note: This operation can only be performed in RAM.</p>

30:5	RESERVED	R	0	reserved bit	Reference Manual
4:2	MODE	R/W	0	<p>Operating Mode Configuration Register</p> <p>000: Normal read operation. Used for AHB read operation mode.</p> <p>001: Programming operation. Program the data to be programmed into the address starting from the address in FLASH_ADDR. The maximum number of sectors to be programmed is half a sector (256 bytes) at a time, and only half a sector can be programmed.</p> <p>010: Sector erase operation (512 bytes per sector) The address of the corresponding erased sector is specified by FLASH_ADDR.</p> <p>Other: Reserved</p> <p>Note 1: All the above operations can only be performed in RAM.</p> <p>Note 2: MODE must be configured back to 0 every time other operation modes (other than normal read operation mode) are completed.</p>	
1	NVR_SEL	R/W	0	<p>NVR Zone Selection</p> <p>0: Main Array selected (128 sectors, 512 bytes per sector)</p> <p>1: Selects the NVR sector (4 sectors, 512 bytes each)</p> <p>Note: This register change must be performed in RAM.</p>	
0	READ_MD	R/W	0x0	<p>Read Rate Mode Selection</p> <p>1: 2 system clock cycles wait (56MHz<sys_clk<84MHz)</p> <p>0: 1 system clock cycle wait (sys_clk<=56MHz)</p> <p>Note: This register change must be performed in RAM.</p>	

FLASH_ADDR register (0x04)

bitfield	name (of a thing)	typology	reset value	descriptive

(math.)				
31:14	RESERVED	R	0	reserved bit
13:0	ADDR	R/W	0	<p>Address register for program/sector erase operations</p> <p>The starting address, in words, at which the programming operation is initiated. The maximum programming data for each consecutive programming is half a sector, i.e., the maximum programming data is 64 words, and programming can only be performed in half a sector;</p> <p>When the sector erase operation is started, bit7-bit13 indicates the sector number to be erased; that is, 1 sector 512Byte.</p>

FLASH_WDATA register (0x08)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:0	WDATA	R/W	0	Data registers for programmed operations When a programming operation is initiated, the data in this register is programmed to be written to the corresponding address

FLASH_START register (0x10)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive
31:1	RESERVED	R	0	reserved bit
0	START	R/W	0	Operational startup control bits Writing this bit to 1 initiates the command operation configured in the MODE register. This bit is automatically cleared when the current command operation is completed.

FLASH_ST register (0x14)

bitfield d (math.)	name (of a thing)	typology	reset value	descriptive

)				
31:3	RESERVED	R	0	reserved bit
2	PROG_BUF_EMPTY	R	1	<p>Programming Data Cache Register Empty Status Flag</p> <p>1: Indicates that the Programming Data Cache Register is empty. The next word of data can be written.</p> <p>0: Indicates that the Programming Data Cache Register is non-empty and the next word of data cannot be written.</p> <p>Writing the next word data during the current word programming is programmed consecutively, otherwise Terminates continuous programming operation.</p>
1	BUSY	R	0	<p>Controller Busy Flag</p> <p>1: Indicates that the controller is in the process of the current command operation.</p> <p>0: Indicates that the controller is in the READY state, waiting for the command operation to be executed.</p>
0	INIT_BUSY	R	1	<p>FLASH Initialization Busy Flag</p> <p>1: Indicates that flash initialization is in progress.</p> <p>0: Indicates initialization is complete</p>

FLASH_LOCK register (0x18)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	LOCK	W	0x0	<p>Operator lock control</p> <p>Writing 0x55 to this register locks the operation start and START cannot be written 1. Used to protect against misoperation of FLASH.</p> <p>Power-up is locked by default.</p>

FLASH_UNLOCK Register (0x1C)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive
31:8	RESERVED	R	0	reserved bit
7:0	UNLOCK	W	0	Operation Unlock Controls Writing 0xAA to this register unlocks operation start and START can be written 1.

FLASH_MASK register (0x20)

bitfield d (math.)	name (of a thing)	typolo gy	reset value	descriptive

31:3	RESERVED	R	0	reserved bit	Reference Manual
2	MASK_LOCK	R/W	1	<p>MASK Select Lock Control</p> <p>When this bit is configured as 1, MASK_SEL cannot be changed. When this bit is set to 0, MASK_SEL can be modified.</p>	
1:0	MASK_SEL	R/W	0	<p>MASK Selection</p> <p>00: Unshielded</p> <p>01: Minimum 2KB (i.e., minimum 4 sectors) blocked</p> <p>10: Minimum 4KB (i.e., minimum 8 sectors) blocked</p> <p>11: Minimum 8KB (i.e., minimum 16 sectors) blocked</p>	

FLASH_ERASETIME register (0x24)

bitfield Id (mat h.)	name (of a thing)	typology	reset value	descriptive
31	RESERVED	R	0	reserved bit
30:19	TRCV	R/W	0xea6	<p>FLASH Sector Erase Time, TRCV Time Register</p> <p>Depending on the frequency of the system clock, the TRCV needs to be configured appropriately so that its timing is at least greater than the 50us requirement.</p> <p>Sector erase operation can be operated with a system clock of up to 72MHz.</p> <p>Note 1: The default value is the configured value for the sector erase operation when the system clock is 48MHz.</p>

18:0	TERASE	R/W	0x2a31 b	FLASH Sector Erase Time, TERASE Time Register Depending on the frequency of the system clock, the TERASE needs to be configured appropriately so that the TERASE time is in the range of 3.2ms-4ms. The sector erase operation works with a system clock of up to 72MHz; Note 1: The default value is the configured value for the sector erase operation when the system clock is 48MHz.
------	--------	-----	-------------	--

FLASH_PROGTIME register (0x28)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive

31:22	RESERVED	R	0	reserved bit	Reference Manual
21:11	TPGS	R/W	0x3e8	<p>TPGS Time Control Register for FLASH programming operations.</p> <p>Depending on the frequency of the system clock, the TPGS needs to be configured appropriately so that its timing is at least greater than the 20us requirement.</p> <p>Note: Default value is for system clock at 48MHz.</p>	
10:0	TPROG	R/W	0x360	<p>TPROG time control register for FLASH programming operation. The TPROG time needs to be in the range of 16us-20us when programming;</p> <p>Depending on the frequency of the system clock, the TPROG needs to be configured appropriately so that it is within the proper range.</p> <p>Note: Default value is for system clock at 48MHz.</p>	

5.23 Cyclic Redundancy Check (CRC)

5.23.1 summarize

This chip CRC is mainly used to reduce the BER of communication lines during data communication. Through the AHB bus interface, the transmitted data can be calculated by this module, and the reliability of data transmission can be ensured by the cyclic redundancy check result.

The block diagram of the system is shown below:

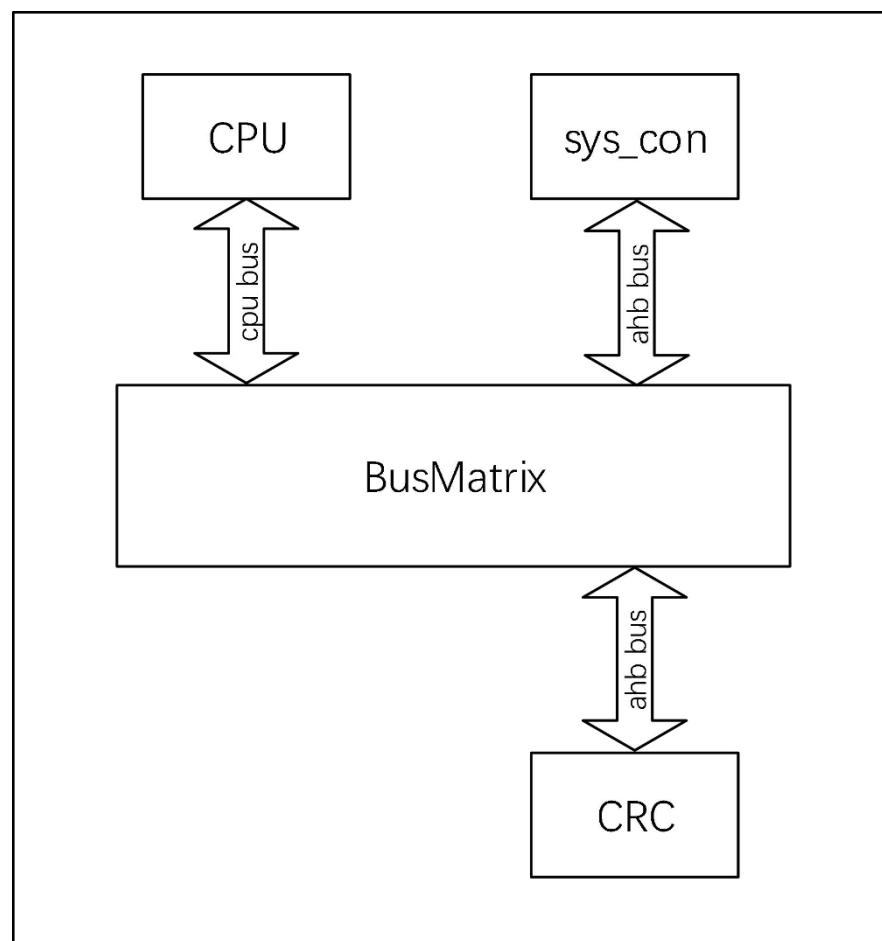


Figure 5-171 CRC Module System Block Diagram

5.23.2 characterization

- Supports CRC operations on 8-, 16-, and 32-bit data.

- Supports input data and output data inversion;
- Supports input data and output data rollover;
- The following four CRC polynomials are supported:

$x^{16}+x^{12}+x^5+1,$

$x^8+x^2+x+1,$

$x^{16}+x^{15}+x^2+1,$

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1.$

5.23.3 Module Structure Diagram

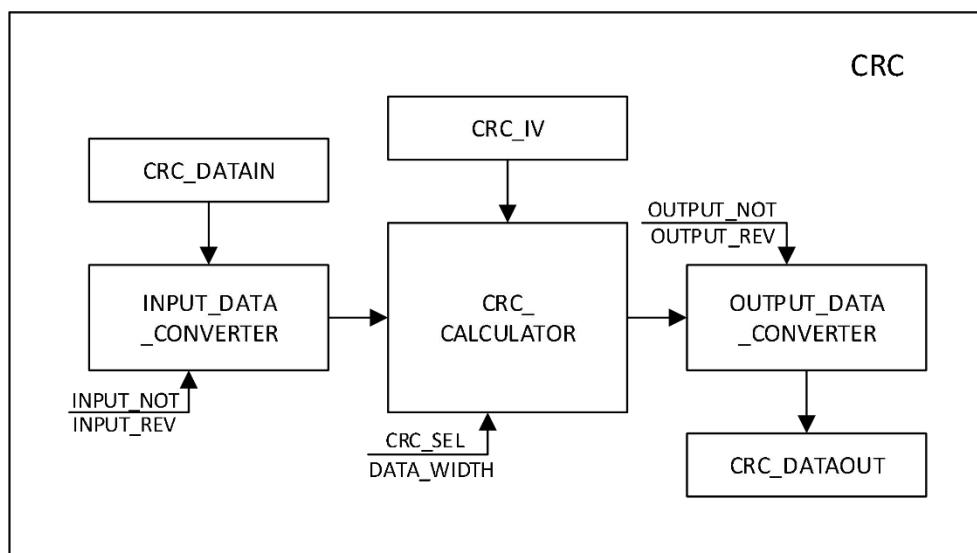


Figure 5-172 CRC Module Structure

The above figure shows the internal structure of the CRC module. The input data `CRC_DATAIN` is inverted and flipped into the CRC calculator, which performs CRC calculation according to the CRC initial value, valid data bit setting and selected CRC polynomial, and outputs the result of the calculation to `CRC_DATAOUT`.

DP32G030
Reference Manual

after inverting and flipping.

5.23.4 Functional Description

CRC code generation rules

The CRC code consists of two parts, the first part is the information code, which is the information to be checked, and the second part is the check code. At the sending end (memory write), according to the k -bit binary code sequence to be transmitted, an r -bit supervisory code (CRC code) for checksum is generated by a certain rule, attached to the original information, constituting a new binary code sequence with a total of $k+r$ bits, which is then sent out. At the receiving end (memory read) a check is performed based on the rules followed between the message code and the CRC code to determine whether an error has occurred in the transmission.

If the CRC code T is n bit long and the message code D is k bit long, the remaining r bit of F is the check bit. For example, (7,3) code:

110 1001, the first three digits of 110 are information code and 1001 is check code. The following is a schematic diagram of a CRC code.

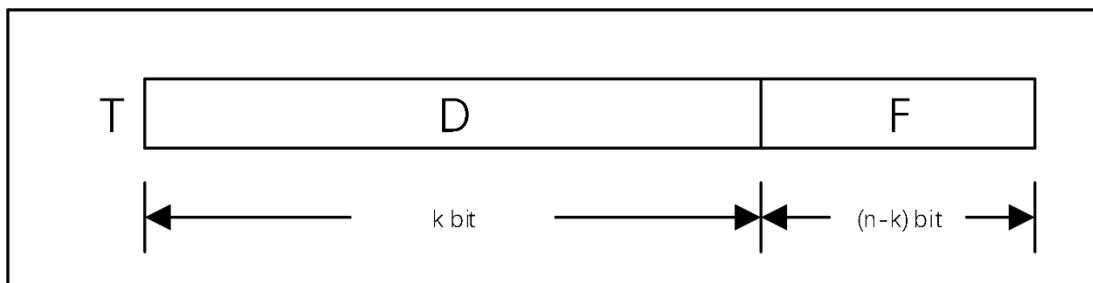


Figure 5-173 Diagram of CRC Code Composition

The rules for generating CRC checksums can be summarized in the following steps:

1. Shift the original message code left by CRC polynomial bit, right by zero;
e.g.: binary data 110 for 4-bit CRC checksum generation -> 110 0000;
2. Divide the CRC polynomial by 110 0000 (note that modulo 2 division is used) and the remainder is the CRC checksum;

3. The check digit is continued to the end of the message code.

CRC polynomial

CRC mode	Name of statement	CRC Checksum Bits	the corresponding polynomial
CRC-CCITT	CRC_CCITT	16	$x^{16}+x^{12}+x^5+1$
CRC-8	CRC_8	8	x^8+x^2+x+1
CRC-16	CRC_16	16	$x^{16}+x^{15}+x^2+1$
CRC-32	CRC_32	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$

workflow

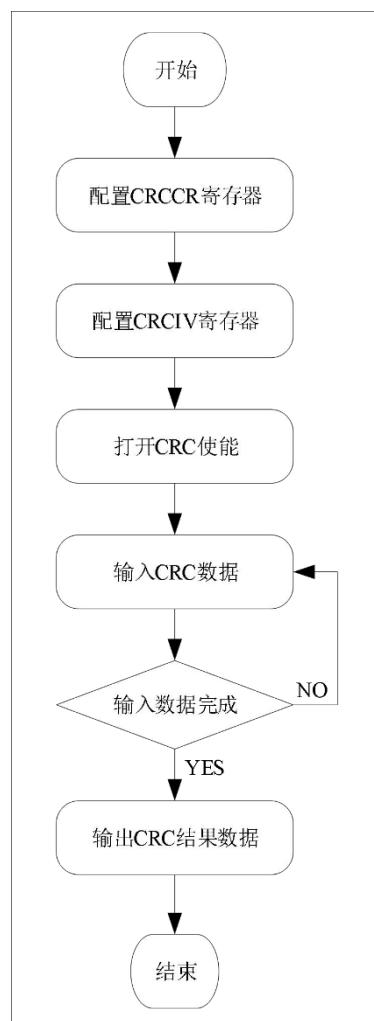


Figure 5-174 CRC Operation Flowchart

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
CRCBASE: 0x40003000					
CRC_CR	0x00	32	R/W	0x00	CRC Control Register
CRC_IV	0x04	32	W	0x00	CRC Initial Value Register
CRC_DATAIN	0x08	32	R/W	0x00	CRC Input Data Register
CRC_DATAOUT	0x0c	32	R	0x00	CRC Output Data Register

register description

CRC_CR register (0x00)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:11	RESERVED	RO	0	reserved bit
10:9	CRC_SEL	R/W	0	CRC Algorithm Selection Register 00: $x^{16}+x^{12}+x^5+1$ 01: x^8+x^2+x+1 10: $x^{16}+x^{15}+x^2+1$ 11: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

				CRC Input Data Valid Bits Register
8:7	DATA_WIDTH	R/W	0	
				00: 32-bit input data valid
				01: Low 16-bit input data valid
				10: Low 8-bit input data valid
				11: Reservations

6:5	OUTPUT_INV	R/W	0	Output Data Flip-Flop Register Flip-Flop Rule Same as INPUT_INV Reference Manual
4	OUTPUT_REV	R/W	0	Whether the output data is inverted or not 1: Inversion of output data 0: Output data is not inverted
3:2	INPUT_INV	R/W	0	Input data flip-flop register 00: bit order unchanged 01: bit order completely flipped 32-bit data width 31:0 -> 0:31; 16-bit data width 15:0 -> 0:15; 8-bit data width 7:0 -> 0:7 10: bit order flipped within byte range 32-bit data width 31:0 -> 24:31, 16:23, 8:15, 0:7; 16-bit data width 15:0 -> 8:15, 0:7; 8-bit data width same as 01 11: Byte-Only Sequential Flip 32-bit data width 31:0 -> 7:0,15:8,23:16,31:24; 16-bit data width 15:0 -> 7:0,15:8; 8-bit data width same as 00
1	INPUT_REV	R/W	0	Whether the input data is inverted or not 1: Input data inversion 0: Input data is not inverted
0	CRC_EN	R/W	0	CRC enable bit 1: CRC Enable 0: CRC not enabled

CRC_IV register (0x04)

bitfield	name (of a thing)	typol	reset	descriptive
----------	-------------------	-------	-------	-------------

d (math .)		ogy	value	
31:0	CRC_IV	R/W	0	CRC Initial Value Register

CRC_DATAIN register (0x08)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:0	CRC_DATAIN	W	0	<p>CRC Input Data</p> <p>Register Note: When DATA_WIDTH is 00: 32 bits are valid.</p> <p>01: Low 16-bit valid</p> <p>10: Lower 8 bits are valid</p> <p>11: Reservations</p>

CRC_DATAOUT register (0x0C)

bitfield d (math .)	name (of a thing)	typol ogy	reset value	descriptive
31:0	CRC_DATAOUT	W	0	<p>CRC Output Data</p> <p>Register Note:</p> <p>When CRC_SEL is 00: the lower 16 bits are valid.</p> <p>01: Lower 8 bits valid</p> <p>10: Low 16-bit valid</p> <p>11: 32-bit valid</p>

5.24 DMA Controller (DMA)

5.24.1 summarize

This module is a DMA controller, which is located between two peripherals and is mainly used to realize data handling between system storage devices and peripherals. The system connection diagram is shown below:

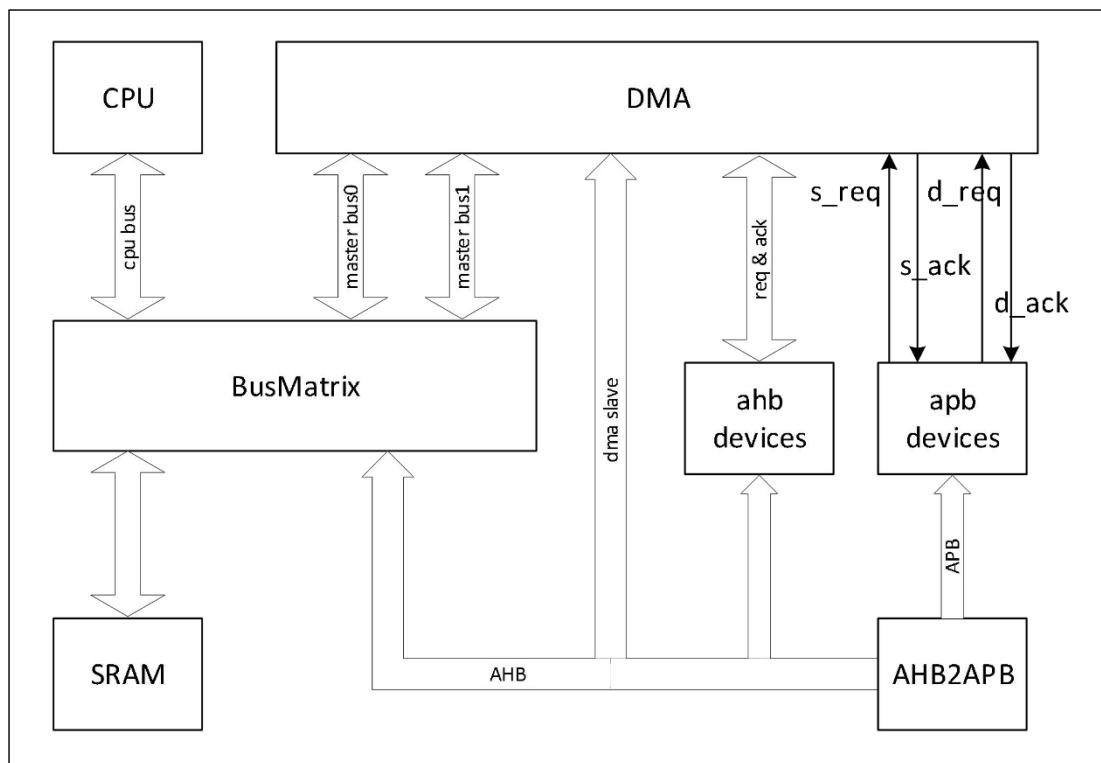


Figure 5-175 DMA Controller System Connection Diagram

5.24.2 characterization

- The number of DMA channels on this chip is 4.
- Has two AHB master console interfaces: one for reading data from the source address device and the other for writing data to the destination address device;

- The source and destination address devices can be configured to transmit data widths of 8, 16, and 32bits, respectively, and the source and destination addresses must be aligned according to the data transmission width;
- Each channel can be configured to transmit up to 4096 data strokes. When each channel finishes transmitting one data entry, the channel releases the bus and the circuit will re-judge the priority of each channel, with the channel with the higher priority transmitting first.
- Support channels can be configured in incremental or fixed address mode.
- Supports memory to memory, memory to peripheral, and peripheral to memory.
- Supports cyclic restart DMA
- Supports configurable per-channel prioritization
- Supports independent interrupts for each channel

5.24.3 Module Structure Schematic

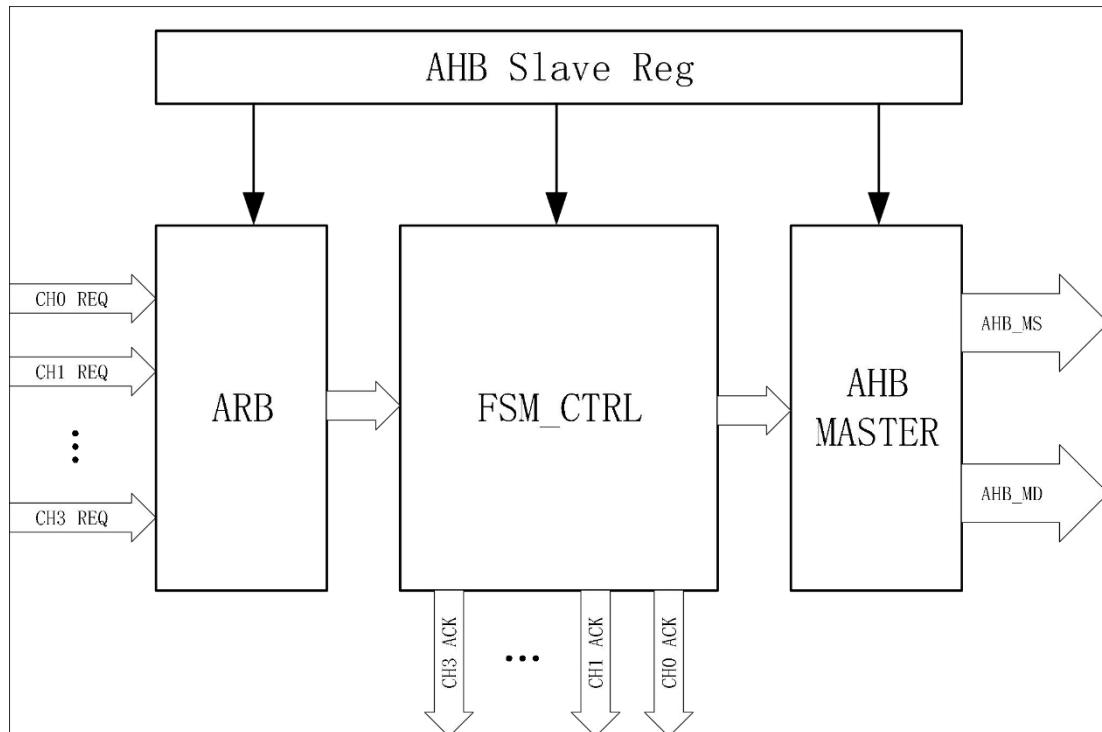


Figure 5-176 DMA Module Structure Diagram

5.24.4 Functional Description

The DMA controller shares the system bus matrix with the CPU core and can directly perform memory-to-memory, memory-to-peripheral, and peripheral-to-memory data transfers. When the CPU and DMA access the same target (RAM or peripheral) at the same time, the DMA request suspends the CPU access to the bus for a number of cycles, and the bus arbiter performs an arbitration operation to ensure that at least half of the system bus bandwidth is available to the CPU.

DMA Processing Flow

Memory (RAM) to memory (RAM)

If the DMA is expected to use the CH_n channel to carry data from one space in RAM to another space in RAM. Configure the source side of the channel to configure the CH_nMOD.MS_SEL register to 0, and the destination side to configure the CH_nMOD.MD_SEL register to 0, and both the source and destination addresses are configured as the address space of the RAM, and after the other configuration items of the channel have been configured, configure the CH_nCON.SWREQ to 1, and then start the data handling, and each data is read out from the source address of RAM through the master controller on the source side according to the number of configured SIZE size DMA. After the other configuration items of the channel are configured, CH_nCON.SWREQ is configured to 1, then the data handling starts, and each data is read out from the source address of RAM through the source-side master controller according to the number of configured SIZE size DMA, and written into the destination address of RAM through the destination-side master controller after the DMA processing.

Memory (RAM) to Peripheral

If you want DMA to use the CH_n channel to carry data from RAM to the peripheral.

Then configure the source side of the channel to configure the CH_nMOD.MS_SEL register to 0, and the destination side to configure the CH_nMOD.MD_SEL register to the corresponding peripheral, and after the other configuration items of the channel have been configured, configure the CH_nCON.SWREQ to 1, and then start to carry the data, and each data according to the number of configured SIZE size DMA through the source side. Main Controller

After reading from the source address of RAM and DMA processing, the processed data is sent to the peripheral through the bus after waiting for the request signal from the destination side, and the answer signal corresponding to the peripheral request is sent to the peripheral for the peripheral to clear the valid request this time in order to initiate the next request normally.

Peripheral to memory (RAM)

If you want DMA to use the CHn channel to carry data from the peripheral to RAM.

MS_SEL register is configured as the corresponding peripheral, and the destination side configures CHnMOD.MD_SEL register as 0. After the other configuration items of the channel are configured, the channel is enabled and waits for the request signal from the source side peripheral, and after the request signal is received, the data handling will be activated, and each data is read out from the source address of peripheral through the master controller of the source side, and is written into the destination address of RAM through the master controller of the destination side after DMA processing. SIZE size DMA is read out from the source address of the peripheral through the master controller of the source side, and after DMA processing, it is written into the destination address of the RAM through the master controller of the destination side, and at the same time, an answer signal of this request is returned to the peripheral of the source side, which is used for the peripheral of the source side to clear the current request, so that the next request can be initiated normally.

DMA arbiter processing

Each channel can be independently configured with a priority level, and the arbiter initiates memory or peripheral access transfers based on the request priority of each channel.

The priority of each channel of the DMA is configured through the Reference Manual. There are 4 levels:

- Low priority
- Medium priority
- High priority
- Highest priority

If two or more channels both initiate a request at the same moment of arbitration, this data transfer carries data to the channel with the higher priority first, and if the priority is the same, the channel with the lower number has priority over the channel with the higher number. For example, channel 2 has priority over channel 3.

When requests are received from other channels while the DMA is in the middle of a data carry for a particular channel, these requests are put on hold until the current data carry for that channel is complete, then control of the DMA is released once. The next transfer selects which channel is being carried by repunching.

DMA Channel Configuration Information

Each channel has its own channel-related configuration registers and status registers, and can independently perform DMA data transfers between fixed-address peripheral registers and memories. The amount of data transferred by DMA is programmable, and can support up to 4096 transfers, with the size of each data bit programmed. A data transfer count register is available to know how many transfers have been performed and is incremented after each transfer is completed.

source address

The CHnMSADDR register allows you to configure the source address of the peripheral or memory for DMA transfers. When a data transfer occurs, this address will be used as the source of the data transfer.

destination address

The CHnMDADDR register allows you to configure the destination address of the peripheral or memory for DMA transfers. When a data transfer occurs, this address will be used as the destination for the data transfer.

channel prioritization

The priority level of the current channel can be configured through the CHnCON.PRI register, and there are 4 levels to choose from. The priority level of each channel is compared at each arbitration cycle, and the higher level is transmitted first.

recurrent mode

The cyclic mode of the current channel is configured through the CHnCON.CIRMD register. When this register is configured as 0, the cyclic mode is not turned on, and the transmission will be stopped after all the configured data volume of the channel has been transmitted; when this register is configured as 1, the cyclic mode is turned on, and after all the configured data volume of the channel has been transmitted, the DMA will restart the data transmission according to the configured information of the channel automatically, until the channel is shut down.

Amount of data transferred

The amount of transmitted data for the current channel can be configured through the CHnCON.LENGTH register, which can support a maximum of 4096 data strokes. Write data format size of the destination side bus is configured through CHnMOD.MD_SIZE register, and read data format size of the source side bus is configured through CHnMOD.MS_SIZE register, which can be programmed as 8bit, 16bit and 32bit respectively.

channel enable (computing)

The current channel can be configured to be turned on or off through the CHnCON.CH_EN register. When the relevant configuration information of the channel is written to the relevant configuration register and the register is configured to 1, the configuration information takes effect and waits for a valid request signal from the source side.

When this register is configured to 1, if the cyclic mode is not turned on, when a valid request is received from the source side, the data transmission will start, and when the configured amount of data is all transmitted, the hardware will automatically clear the CHnCON.CH_EN register to 0 to shut down the channel, and at this time, the channel will not be activated even if a valid request is received from the source side again. If you want to enable the channel transmission again, you need to program the register to 1 by software.

When this register is configured to 1, if the cyclic mode is on, when a valid request is received from the source side, the data transfer will start, and when the configured amount of data has been transferred in full, the transfer data amount counter will start counting again, and the DMA will perform the data transfer again according to the previously configured information. If data transfer is no longer desired, software programming is required to configure this register CHnCON.CH_EN to 0 to close the channel, then no further data transfer will take place even if a request is received from the source side. Note that when software clears the CHnCON.CH_EN register to 0, the channel is not shut down immediately, and the hardware will not automatically shut down the channel until the entire amount of data currently being transferred has been transferred.

Channel mapping selection

Each channel can independently select the corresponding memory or peripheral on the source and destination sides.

The source side can be programmatically selected via the CHnMOD.MS_SEL register, and the destination side can be programmatically selected via the CHnMOD.MD_SEL register. Both registers can each select multiple devices (memory or peripherals) where a configuration of 0x00 selects for memory, usually the on-chip RAM memory. They can also be used for other peripherals that can be accessed without handshaking or waiting. Configuration 0x01-0x07 selects a peripheral that has a function that requires handshaking.

Channel address change method

Each channel can independently select the address change method for the source and destination sides. There are two address change methods to choose from: address incremental and address constant.

The source side can be programmed through the CHnMOD.MD_ADDMOD register, and the destination side can be programmed through the CHnMOD.MS_ADDMOD register. CHnMOD.MD_ADDMOD Register Programming Selection.

Source and Destination Address Data Format Description

The data format of the source and destination addresses can be configured separately, and the correspondence is shown in the following example:

F0/F1/F2/F3/F4/F5/F6/F7/F8/F9/FA/FB/FC/FD/FE/FF all represent one byte of data.

Figure 5-177 Data Format Configuration Table for DMA Source and Destination Addresses

s o u rc e w id th	T ar g et w id th	tra ns mis sio n len gth	Source (address/data)	transport operation	Destination (address/dat a)
8	8	4	0x0/F0 0x1/F1 0x2/F2 0x3/F3	1: Read F0[7:0] at 0x0, write F0[7:0] at 0x0. 2: read F1[7:0] at 0x1, write F1[7:0] at 0x1 3: Read F2[7:0] at 0x2, write F2[7:0] at 0x2. 4: Read F3[7:0] at 0x3, write F3[7:0] at 0x3.	0x0/F0 0x1/F1 0x2/F2 0x3/F3
8	16	4	0x0/F0 0x1/F1 0x2/F2 0x3/F3	1: Read F0[7:0] at 0x0, write 00F0[15:0] at 0x0. 2:Read F1[7:0] at 0x1, write 00F1[15:0] at 0x2 3: Read F2[7:0] at 0x2, write 00F2[15:0] at 0x4. 4: Read F3[7:0] at 0x3, write 00F3[15:0] at 0x6.	0x0/00F0 0x2/00F1 0x4/00F2 0x6/00F3
8	32	4	0x0/F0 0x1/F1 0x2/F2 0x3/F3	1: Read F0[7:0] at 0x0, write 000000F0[31:0] at 0x0. 2: Read F1[7:0] at 0x1, write 000000F1[31:0] at 0x4. 3: Read F2[7:0] at 0x2, write 000000F2[31:0] at 0x8. 4: Read F3[7:0] at 0x3, write 000000F3[31:0] at 0xC.	0x0/000000F0 0x4/000000F1 0x8/000000F2 0xC/000000F3
16	8	4	0x0/F1F0 0x2/F3F2 0x4/F5F4 0x6/F7F6	1: Read F1F0[15:0] at 0x0, write F0[7:0] at 0x0. 2: Read F3F2[15:0] at 0x2, write F2[7:0] at 0x1. 3: Read F5F4[15:0] at 0x4, write F4[7:0] at 0x2. 4:Read F7F6[15:0] at 0x6, write F6[7:0] at 0x3.	0x0/F0 0x1/F2 0x2/F4 0x3/F6
16	16	4	0x0/F1F0 0x2/F3F2 0x4/F5F4 0x6/F7F6	1:Read F1F0[15:0] at 0x0, write F1F0[15:0] at 0x0. 2: Read F3F2[15:0] at 0x2, write F3F2[15:0] at 0x2. 3: Read F5F4[15:0] at 0x4, write F5F4[15:0] at 0x4. 4:Read F7F6[15:0] at 0x6, write F7F6[15:0] at 0x6.	0x0/F1F0 0x2/F3F2 0x4/F5F4 0x6/F7F6
16	32	4	0x0/F1F0 0x2/F3F2 0x4/F5F4 0x6/F7F6	1: Read F1F0[15:0] at 0x0, write 0000F1F0[31:0] at 0x0. 2: Read F3F2[15:0] at 0x2, write 0000F3F2[31:0] at 0x4. 3: Read F5F4[15:0] at 0x4, write 0000F5F4[31:0] at	0x0/0000F1F0 0x4/0000F3F2 0x8/0000F5F4 0xC/0000F7F6

			0x8. 4: Read F7F6[15:0] at 0x6, write 0000F7F6[31:0] at 0xC.	
32	8	4	0x0/ F3F2F1F0 0x4/ F7F6F5F4 0x8/FBFAF9F8 0xC/FFFEFDFO	1: Read F3F2F1F0[31:0] at 0x0, write F0[7:0] at 0x0. 2: Read F7F6F5F4[31:0] at 0x4, write F4[7:0] at 0x1 3:Read FBFAF9F8 [31:0] at 0x8, write F8[7:0] at 0x2 4: read FFFEFDFC [31:0] at 0xc, write FC[7:0] at 0x3
32	16	4	0x0/ F3F2F1F0 0x4/ F7F6F5F4 0x8/FBFAF9F8 0xC/FFFEFDFO	1: Read F3F2F1F0[31:0] at 0x0, write F1F0[15:0] at 0x0. 2: Read F7F6F5F4[31:0] at 0x4, write F5F4[15:0] at 0x2. 3:Read FBFAF9F8 [31:0] at 0x8, write F9F8[15:0] at 0x4 4:Read FFFEFDFC[31:0] at 0xc, write FDFO[15:0] at 0x6
32	32	4	0x0/ F3F2F1F0 0x4/ F7F6F5F4 0x8/FBFAF9F8 0xC/FFFEFDFO	1: Read F3F2F1F0[31:0] at 0x0, write F3F2F1F0[31:0] at 0x0. 2: Read F7F6F5F4[31:0] at 0x4, write F7F6F5F4[31:0] at 0x4. 3:Read FBFAF9F8 [31:0] at 0x8, write FBFAF9F8 [31:0] at 0x8 4:Read FFFEFDFC [31:0] at 0xc, write FFFEFDFC [31:0] at 0xc.

Interrupt Description

Each channel can generate interrupts at the halfway point of a DMA transfer and at the completion of the transfer. To meet application flexibility and practicality considerations, these interrupts can be turned on by programming different bits in the registers.

Figure 5-178 DMA Interrupt Description Table

disruption event	Enable Control Bit	event marker
more than halfway through transmission	HTC_INTEN	HTC_INTST
Transmission complete	TC_INTEN	TC_INTST

DMA Request Mapping Relationship

The DMA of this chip has four channels, channel 0-channel 3, which support requests from eight peripherals: UART0, UART1, UART2, SPI0, SPI1, ADC, TIMER_PLUS0 and TIMER_PLUS1.

Which peripheral requests correspond to each channel can be found in the following table and the corresponding relationships shown in Fig.

The mapping relationship of the corresponding peripheral request for each channel is as follows:

Figure 5-179 Mapping of Peripheral Requests for Each DMA Channel

Corresponding bit	Channel 0		Channel 1	
	source side	target side	source side	target side
000	UART0_RX	UART2_TX	UART1_RX	UART0_TX
001	UART1_RX	UART0_TX	UART2_RX	UART1_TX
010	SPI0_RX	SPI1_TX	SPI1_RX	SPI0_TX
011	SARADC	N/A	SARADC	N/A
100	TIMER_PLUS0_L	N/A	TIMER_PLUS0_H	N/A
101	TIMER_PLUS1_L	N/A	TIMER_PLUS1_H	N/A
Corresponding bit	Channel 2		Channel 3	
	source side	target side	source side	target side
000	UART2_RX	UART1_TX	UART0_RX	UART2_TX
001	UART0_RX	UART2_TX	UART1_RX	UART0_TX
010	SPI1_RX	SPI0_TX	SPI0_RX	SPI1_TX
011	SARADC	N/A	SARADC	N/A
100	TIMER_PLUS1_L	N/A	TIMER_PLUS1_H	N/A
101	TIMER_PLUS0_L	N/A	TIMER_PLUS0_H	N/A

The source-side peripheral request mapping relationship is shown below:

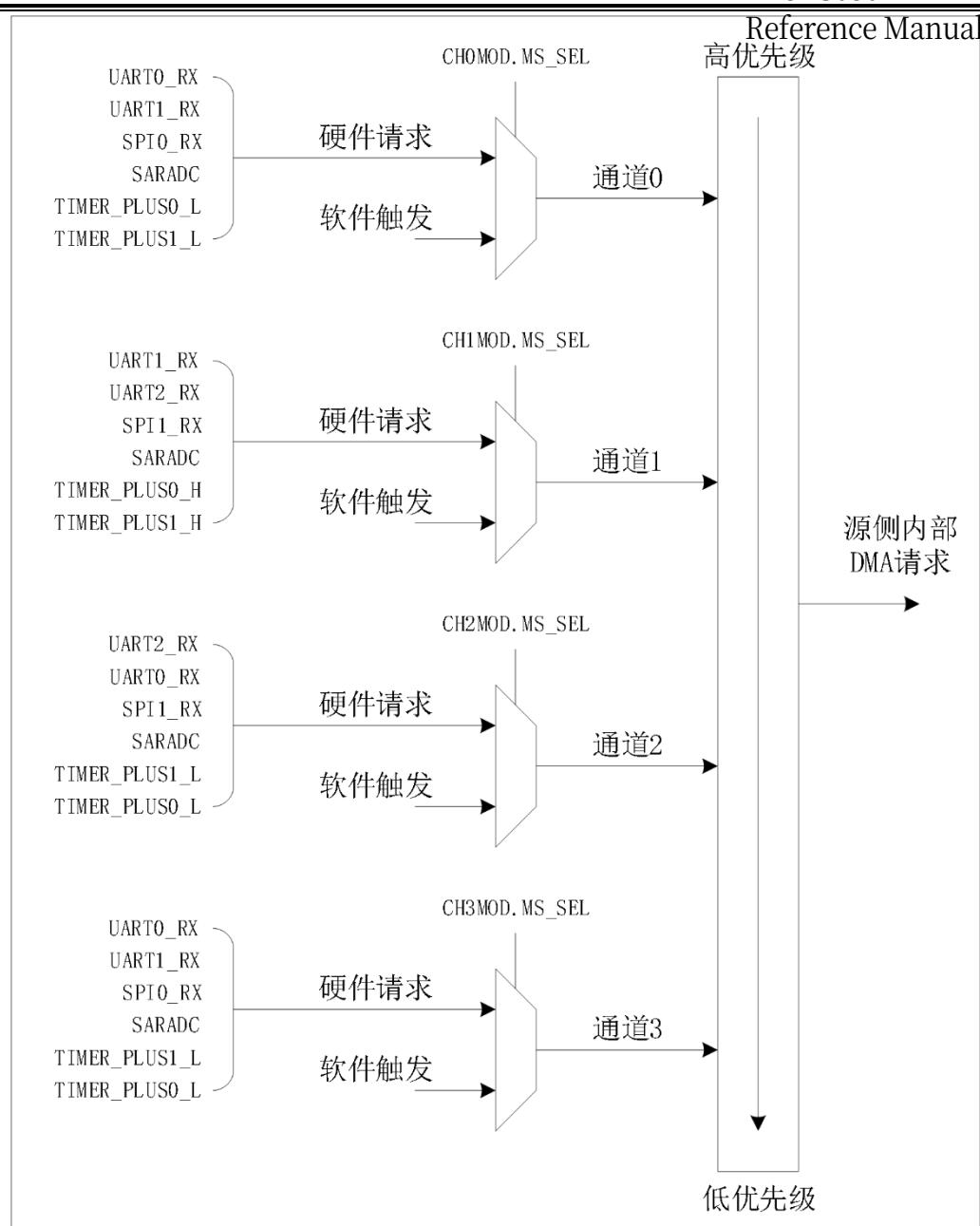


Figure 5-180 DMA Source-Side Peripheral Request Mapping Relationship Diagram

The destination-side peripheral request mapping relationship diagram is as follows:

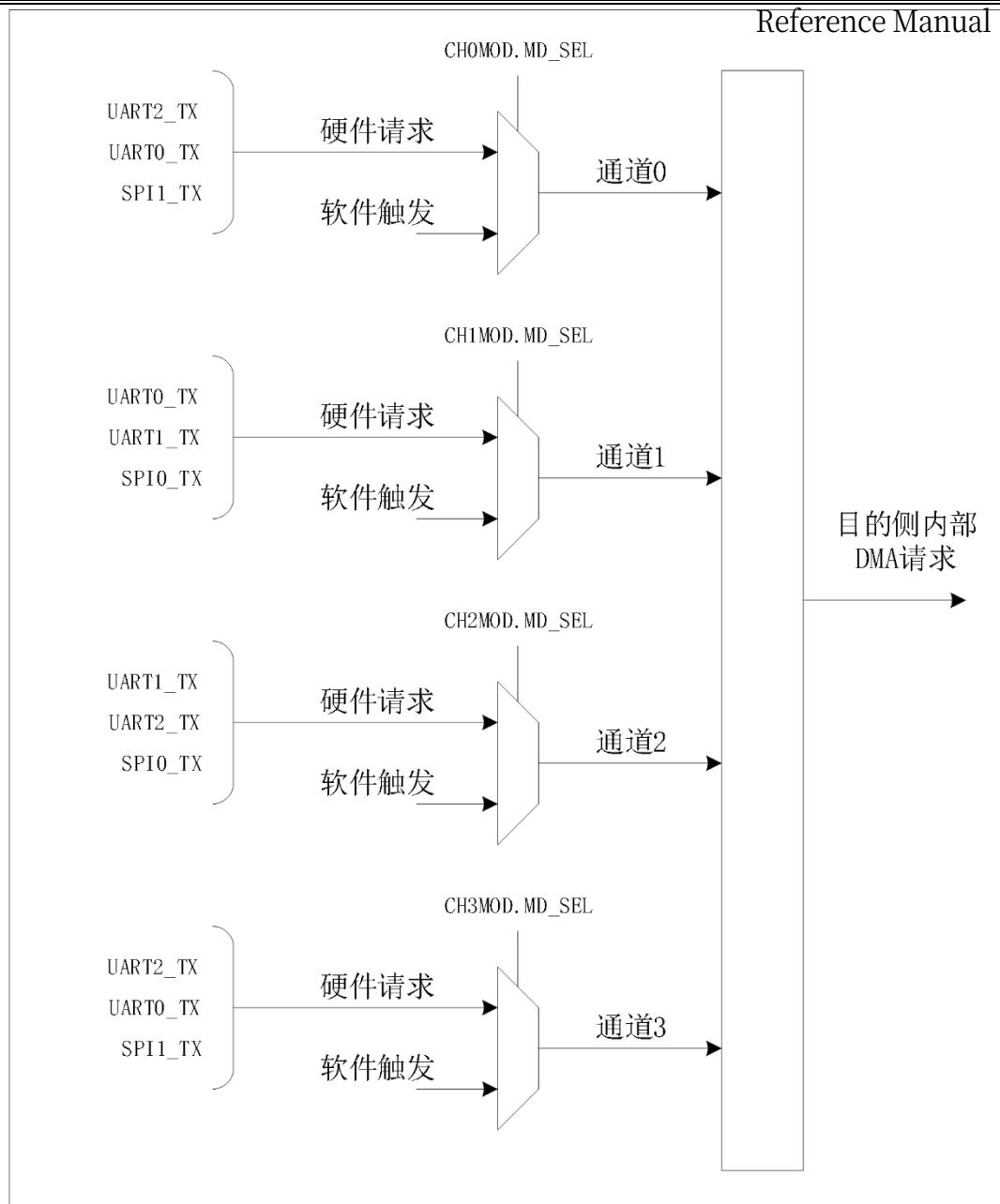


Figure 5-181 DMA Destination Side Peripheral Request Mapping Relationship Diagram

DMA Workflow

The workflow of DMA is shown in the figure below:

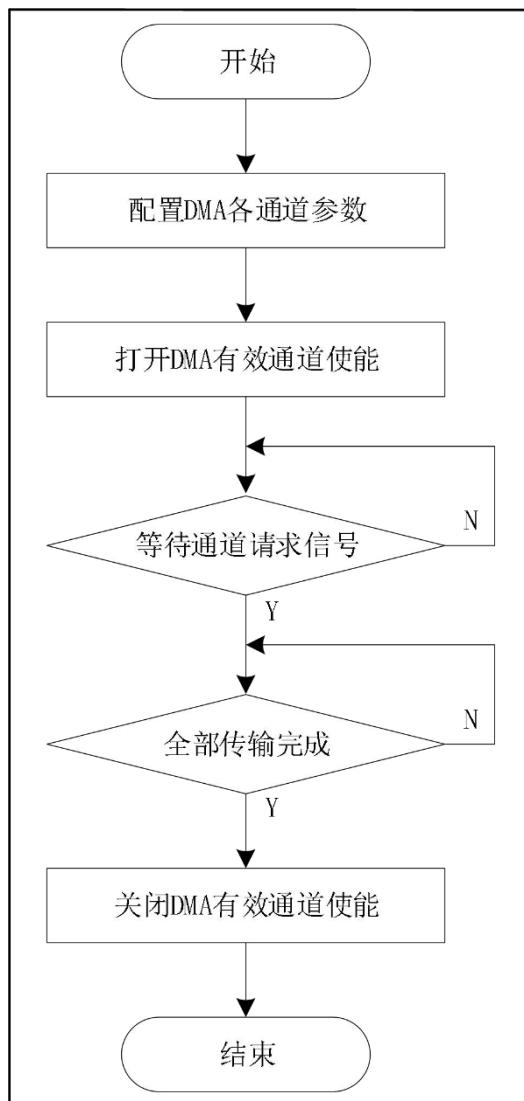


Figure 5-182 DMA Workflow Diagram

The above figure briefly describes the working process of DMA. There are some main operations as follows:

- 1) Configure the DMA_EN bit in register DMA_CON to 1 to turn DMA on;
- 2) Turn on the corresponding interrupt enable for each channel;
- (3) The DMA_CHnCON, DMA_CHnMOD, DMA_CHnMSADDR, and DMA_CHnMSADDR of each channel are passed through the registers of each channel.
DMA_CHnMADDR programs the parameter information to be configured for the respective channel into the corresponding register bits;

- 4) Configure the corresponding channel DMA_CHnCON.CH_EN register in the Reference Manual; the channel;

- 5) Waiting for a request signal from the source side of the channel, the DMA receives the request signal from the source side, reads data from the source address set by the corresponding peripheral via the bus, and saves it for processing;
- 6) After reading back the required data from the source side, it waits for the request signal from the destination side. When the DMA receives the request signal from the destination side, it writes the processed data to the destination address set by the corresponding peripheral device via the bus;
- 7) Depending on the configured amount of data to be transferred, the transfer half interrupt flag or transfer complete interrupt flag is generated when half or all of all data has been transferred;
- 8) When all data has been transferred, configure this channel CH_EN to 0 to turn off the channel. At this point, you can modify the configuration register value of this channel and wait for the next data transfer to use it.

register map

name (of a thing)	offset	bit width	typology	reset value	descriptive
DMABASE: 0x40001000					
DMA_CTR	0x00	32	R/W	0x00	DMA Control Register
DMA_INTEN	0x04	32	R/W	0x00	DMA Interrupt Enable Register
DMA_INTST	0x08	32	R/W	0x00	DMA Interrupt Status Register
DMA_CH0CTR	0x100	32	R/W	0x1ffe	Channel 0 Control Register
DMA_CH0MOD	0x104	32	R/W	0x00	Channel 0 Mode Register
DMA_CH0MSADDR	0x108	32	R/W	0x00	Channel 0 Source Address Register

DMA_CH0MDADDR	0x10c	32	R/W	0x00	Channel 0 Destination Address Register
DMA_CH0_ST	0x110	32	R	0x00	Channel 0 Status Register
DMA_CH1CTR	0x120	32	R/W	0x1ffe	Channel 1 Control Register
DMA_CH1MOD	0x124	32	R/W	0x00	Channel 1 Mode Register
DMA_CH1MSADDR	0x128	32	R/W	0x00	Channel 1 Source Address Register
DMA_CH1MDADDR	0x12c	32	R/W	0x00	Channel 1 Destination Address Register

DMA_CH1_ST	0x130	32	R	0x00	Reference Manual Channel 1 Status Register
DMA_CH2CTR	0x140	32	R/W	0x1ffe	Channel 2 Control Register
DMA_CH2MOD	0x144	32	R/W	0x00	Channel 2 Mode Register
DMA_CH2MSADDR	0x148	32	R/W	0x00	Channel 2 Source Address Register
DMA_CH2MDADDR	0x14c	32	R/W	0x00	Channel 2 Destination Address Register
DMA_CH2_ST	0x150	32	R	0x00	Channel 2 Status Register
DMA_CH3CTR	0x160	32	R/W	0x1ffe	Channel 3 Control Register
DMA_CH3MOD	0x164	32	R/W	0x00	Channel 3 Mode Register
DMA_CH3MSADDR	0x168	32	R/W	0x00	Channel 3 Source Address Register
DMA_CH3MDADDR	0x16c	32	R/W	0x00	Channel 3 Destination Address Register
DMA_CH3_ST	0x170	32	R	0x00	Channel 3 Status Register

register description

DMA_CTR register (0x00)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31: 1	RESERVED	R	0	reserved bit
0	DMA_EN	R/W	0	DMA Enable 0: DMA off 1: DMA Enable

DMA_INTEN register (0x04)

bitfield d (math.)	name (of a thing)	typol ogy	reset value	descriptive
31:12	RESERVED	R	0	reserved bit

Reference Manual				
11	CH3 THC_INTEN	R/W	0	Channel 3 Transmission Half Completed Interrupt Enable Register
10	CH2 THC_INTEN	R/W	0	Channel 2 Transmit Half Completed Interrupt Enable Register
9	CH1 THC_INTEN	R/W	0	Channel 1 Transmission Half Completed Interrupt Enable Register
8	CH0 THC_INTEN	R/W	0	Channel 0 Transmit Half Completed Interrupt Enable Register
7:4	RESERVED	R	0	reserved bit
3	CH3_TC_INTEN	R/W	0	Channel 3 Transmission Completion Interrupt Enable Register
2	CH2_TC_INTEN	R/W	0	Channel 2 Transmission Completion Interrupt Enable Register
1	CH1_TC_INTEN	R/W	0	Channel 1 Transmission Completion Interrupt Enable Register
0	CH0_TC_INTEN	R/W	0	Channel 0 Transmission Completion Interrupt Enable Register

DMA_INTST register (0x08)

bitfield (math.)	name (of a thing)	typology	reset value	descriptive
31:12	RESERVED	R	0	reserved bit
11	CH3 THC_INTST	R/W	0	Channel 3 Transmission Half Completed Interrupt Status Register Write 1 to clear.
10	CH2 THC_INTST	R/W	0	Channel 2 Transmission Half Completed Interrupt Status Register Write 1 to clear.

9	CH1_THC_INTST	R/W	0	Channel 1 Transmission Half Completed Interrupt Status Register Write 1 to clear.
8	CH0_THC_INTST	R/W	0	Channel 0 Transmit Half Completed Interrupt Status Register Write 1 to clear.
7:4	RESERVED	R	0	reserved bit
3	CH3_TC_INTST	R/W	0	Channel 3 Transmission completion interrupt status register Write 1 to clear.
2	CH2_TC_INTST	R/W	0	Channel 2 Transmission Completion Interrupt Status Register Write 1 to clear.

Reference Manual				
1	CH1_TC_INTST	R/W	0	Channel 1 Transmission completion interrupt status register Write 1 to clear.
0	CH0_TC_INTST	R/W	0	Channel 0 Transmission completion interrupt status register Write 1 to clear.

DMA_CHnCTR register (**0x100 + 0x20*(n)**)

bitfield ld (mat h.)	name (of a thing)	typolo gy	reset value	descriptive
31:17	RESERVED	R	0	reserved bit
16	SWREQ	R/W	0	<p>Requesting this channel to start transmission via software</p> <p>Write 1 Starting transmission</p> <p>Note 1: Only used for requests initiated when the source address side is configured as a storage device. Note 2: If LOOP is configured as 0, it will be automatically cleared by hardware upon completion of a transfer; if LOOP is configured as 1, it will be cleared by software control.</p>
15:14	PRI	R/W	0	<p>channel prioritization</p> <p>00: Low</p> <p>01: Medium</p> <p>10: High</p> <p>11: Maximum</p> <p>Note: If different channels are configured with the same priority, the one with the smaller channel number has the higher priority.</p>

				Reference Manual
13	LOOP	R/W	0	<p>Cycle mode control bit</p> <p>0: Cyclic mode is not executed. Stop when transmission is completed</p> <p>1: Execute the cyclic method. Automatically re-transmits according to the original configuration after the transmission is completed</p>
12:1	LENTH	R/W	0xffff	<p>Transmission Count Register, which indicates the number of requests transmitted on this channel. The actual number of transmissions is (LENTH+1).</p> <p>The unit of data for each transmission is: the unit of data on the source address side is determined by the MS_SIZE is determined; the data unit on the destination address side is determined by MD_SIZE.</p>
0	CH_EN	R/W	0	<p>Channel Enable Control</p> <p>0: Channel closed</p> <p>1: Channel effective</p> <p>Note 1: Channel enable is written 1 by software to initiate a transmission. If configured as 0, a transmission is completed and automatically cleared by hardware; If configured as 1, it is cleared by software control.</p> <p>Note 2: All configurations need to be done with enable off. When this bit is 1, the relevant configuration cannot be changed.</p>

DMA_CHnMOD register (**0x100 + 0x20*(n) + 0x04**)

bitfield	name (of a thing)	typology	reset value	descriptive
ld (mat h.)				

31:14	RESERVED	R	0	reserved bit
13:11	MD_SEL	R/W	0	<p>MD Side Peripheral Selection</p> <p>000: Select storage peripheral</p> <p>001: Select the peripheral that generates hsreq_md[0]</p> <p>010: Select the peripheral that generates hsreq_md[1].</p> <p>011: Select the peripheral that generates hsreq_md[2].</p> <p>100: Selects the peripheral that generates hsreq_md[3]</p> <p>101: Select the peripheral corresponding to the generation of hsreq_md[4]</p> <p>110: Selects the peripheral that generates hsreq_md[5]</p> <p>111: Select the peripheral that generates hsreq_md[6]</p> <p>Note 1: A configuration of 000 indicates the selection of a memory peripheral, which usually indicates on-chip SRAM. can also be used for other peripherals that are accessible without handshaking or waiting.</p> <p>Note 2: When the configuration is 001-111, the peripherals connected to different chips are different, and the corresponding peripherals need to be decided according to the actual peripherals connected to the specific chip.</p>