珠海泰为电子有限公司
Zhuhai Tai-Action Electronics CO., LTD.

TAE32F5600 系列

芯片手册
Specification

版本号：Rev 2.3
修订日期：2022 年 5 月 10 日

目录

1 文档说明 1

1.1 缩写词 1
1.2 词汇表 1
1.3 相关文档 2

2 特性列表 3

2.1 特性描述 3
2.2 资源汇总 4

3 引脚定义 5

3.1 引脚框图 5
3.2 引脚描述 7

4 存储器和总线架构 13

4.1 系统架构 13
4.2 总线矩阵 13
4.3 存储器地址映射 15
4.4 外设寄存器映射 15
4.4.1 SRAM 17
4.4.2 FLASH 17
4.5 自举配置 17

5 嵌入式 FLASH（FLASH） 18

5.1 简介 18
5.2 结构框图 18
5.3 主要特性 19
5.4 功能描述 20
5.4.1 实时加速器 20
5.4.2 擦除和编程操作 22
5.5 寄存器描述 29
5.5.1 寄存器列表 29
5.5.2 寄存器详细描述 30

6 数据存储 FLASH（DFLASH） 39

6.1 简介 39
6.2 结构框图 39
6.3 主要特性 39
6.4 功能描述 40
6.4.1 DFLASH 擦除和编程操作 40
6.5 寄存器描述 43
6.5.1 寄存器列表 43

6.5.2 寄存器描述 44
7 电源控制器（PWR） 49
7.1 电源 49
7.1.1 LDO（线性调压器） 50
7.2 电源监控器 50
7.2.1 上电复位（POR）/掉电复位（PDR） 50
7.2.2 低电压检测器（LVD） 51
7.3 低功耗模式 51
7.3.1 降低系统时钟速度 52
7.3.2 外设时钟门控 52
7.3.3 睡眠模式 52
7.3.4 停止模式 53
7.4 PWR 电源控制寄存器 54
7.5 LVD 低电压监测寄存器 55
7.5.1 寄存器列表 55
7.5.2 寄存器详细描述 56
8 复位和时钟控制（RCU） 60
8.1 复位 60
8.1.1 系统复位 60
8.1.2 电源复位 61
8.2 时钟 62
8.2.1 架构图 62
8.2.2 系统时钟（SYSCLK）选择 62
8.2.3 LSI 振荡器 63
8.2.4 HSI 振荡器 63
8.2.5 HSE 晶振 64
8.2.6 PLL 配置 64
8.2.7 时钟安全系统（CSS） 65
8.2.8 看门狗时钟 66
8.2.9 时钟输出功能 66
8.3 寄存器描述 67
8.3.1 寄存器列表 67
8.3.2 寄存器详细描述 68
9 系统配置控制器（SYSCTRL） 90
9.1 寄存器描述 90
9.1.1 寄存器列表 90
9.1.2 寄存器详细描述 91
10 通用 I/O（GPIO） 95
10.1 简介 95
10.2 结构框图 95
10.3 主要特性 96

10.4 功能描述 97
10.4.1 I/O 复位状态 97
10.4.2 通用 I/O 97
10.4.3 引脚复用 98
10.4.4 I/O 控制寄存器 99
10.4.5 I/O 数据寄存器 99
10.4.6 I/O 数据位操作 99
10.4.7 I/O 复用功能 100
10.4.8 外部中断功能 100
10.4.9 输入配置 100
10.4.10 输出配置 102
10.4.11 复用功能配置 105
10.5 寄存器描述 106
10.5.1 寄存器列表 106
10.5.2 寄存器描述 107

11 基础定时器（TMR0~3） 115

11.1 简介 115
11.2 结构框图 115
11.3 主要特性 116
11.4 功能描述 116
11.4.1 时基单元 116
11.4.2 计数器模式 118
11.4.3 定时器同步 120
11.4.4 调试模式 120
11.4.5 事件与中断 120
11.5 寄存器描述 122
11.5.1 寄存器列表 122
11.5.2 寄存器描述 123

12 通用定时器（TMR4~7） 128

12.1 简介 128
12.2 结构框图 128
12.3 主要特性 129
12.4 功能描述 129
12.4.1 时基单元 129
12.4.2 计数器模式 131
12.4.3 时钟选择 134
12.4.4 捕获/比较通道 135
12.4.5 输入捕获 135
12.4.6 输出比较 136
12.4.7 定时器同步 138
12.4.8 调试模式 138
12.4.9 事件与中断 138
12.5 寄存器描述 142

12.5.1 寄存器列表 142
12.5.2 寄存器描述 143
13 独立看门狗（IWDG） 154
13.1 简介 154
13.2 结构框图 154
13.3 主要特性 154
13.4 功能描述 155
13.4.1 IWDG 键值功能 155
13.4.2 IWDG 寄存器访问保护 155
13.4.3 IWDG 调试模式 155
13.4.4 IWDG 中断模式和复位模式 155
13.4.5 IWDG 计时 156
13.4.6 IWDG 中断和状态 156
13.5 寄存器描述 157
13.5.1 寄存器列表 157
13.5.2 寄存器描述 158
14 窗口看门狗（WWDG） 162
14.1 简介 162
14.2 结构框图 162
14.3 主要特性 162
14.4 功能描述 163
14.4.1 使能看门狗 163
14.4.2 控制递减计数器 163
14.4.3 看门狗中断高级特性 163
14.4.4 复位使能和调试模式 164
14.4.5 看门狗设置 164
14.5 寄存器描述 165
14.5.1 寄存器列表 165
14.5.2 寄存器描述 166
15 DMA 控制器（DMA） 169
15.1 简介 169
15.2 结构框图 169
15.3 主要特性 170
15.4 功能描述 170
15.4.1 DMA 传输 170
15.4.2 DMA 事务 170
15.4.3 DMA 通道 170
15.4.4 DMA 仲裁 171
15.4.5 DMA 源、目标和传输模式 172
15.4.6 DMA 地址控制 175
15.4.7 DMA 中断 175
15.5 寄存器描述 176

15.5.1 寄存器列表 176
15.5.2 寄存器详细描述 177

16 中断和事件 182

16.1 嵌套向量中断控制器（NVIC） 182
16.1.1 NVIC 特性 182
16.1.2 Sys Tick 校准值寄存器 182
16.1.3 中断和异常向量 182
16.1.4 唤醒事件 184

17 CORDIC 运算单元（CORDIC） 185

17.1 简介 185
17.2 主要特性 185
17.3 功能描述 185
17.3.1 CORDIC 方程 185
17.3.2 CORDIC 运算单元操作 186
17.3.3 数据格式 186
17.3.4 溢出标志及幅度调节 187
17.3.5 操作模式和对应结果数据 187
17.4 寄存器描述 189
17.4.1 寄存器列表 189
17.4.2 寄存器描述 190

18 IQ 除法单元（IQDIV） 195

18.1 简介 195
18.2 主要特性 195
18.3 功能描述 195
18.3.1 IQ 格式除法 195
18.3.2 注意事项 196
18.4 寄存器描述 197
18.4.1 寄存器列表 197
18.4.2 寄存器描述 198

19 模数转换器（ADC） 203

19.1 简介 203
19.2 结构框图 203
19.3 主要特性 204
19.4 功能描述 205
19.4.1 SOC 工作原理 205
19.4.2 单次转换支持 208
19.4.3 ADC 转换优先级 208
19.4.4 同步采样模式 210
19.4.5 EOC 和中断 211
19.4.6 上电顺序 212
19.4.7 ADC 校准 212

19.4.8 内部/外部参考电压选择 213
19.4.9 内部温度传感器 213
19.5 寄存器描述 215
19.5.1 寄存器列表 215
19.5.2 寄存器详细描述 217

20 比较器（DACCMP） 293

20.1 简介 293
20.2 结构框图 293
20.3 主要特性 293
20.4 功能描述 294
20.4.1 比较器功能 294
20.4.2 DAC 基准电压 294
20.4.3 斜坡发生器输入 295
20.4.4 初始化 296
20.4.5 数字域操作 296
20.5 寄存器描述 297
20.5.1 寄存器列表 297
20.5.2 寄存器详细描述 298

21 增强型脉宽调制器（EPWM） 305

21.1 简介 305
21.2 结构框图 305
21.3 主要特性 308
21.4 功能描述 309
21.4.1 EPWM 子模块 309
21.4.2 电源拓扑的应用 349
21.5 寄存器描述 361
21.5.1 寄存器列表 361
21.5.2 寄存器详细描述 363

22 脉冲密度调制（PDM） 403

22.1 简介 403
22.2 结构框图 403
22.3 主要特性 404
22.4 功能描述 405
22.4.1 SPI 控制单元 405
22.4.2 SINC 滤波器 405
22.4.3 主滤波器 406
22.4.4 比较滤波器 409
22.5 寄存器描述 411
22.5.1 寄存器列表 411
22.5.2 寄存器描述 412

23 内部集成接口（I2C） 423

23.1 简介 423
23.2 结构框图 423
23.3 主要特性 424
23.4 功能描述 424
23.4.1 I2C 协议 424
23.4.2 操作模式 430
23.4.3 I2C_CLK 频率配置 432
23.4.4 SDA 保持时间 ..... 错误！未定义书签。
23.4.5 I2C 中断 ..... 错误！未定义书签。
23.4.6 DMA 控制接口 433
23.5 寄存器描述 437
23.5.1 寄存器列表 437
23.5.2 寄存器详细描述 438

24 串行外设接口（SPI） 452

24.1 简介 452
24.2 结构框图 452
24.3 主要特性 453
24.4 功能描述 454
24.4.1 SPI 协议 454
24.4.2 SPI 主设备 456
24.4.3 SPI 从设备 457
24.4.4 DMA 模式 458
24.5 SPI 寄存器描述 459
24.5.1 寄存器列表 459
24.5.2 寄存器详细描述 460

25 通用异步收发器（UART） 468

25.1 简介 468
25.2 结构框图 468
25.3 主要特性 469
25.4 功能描述 469
25.4.1 UART 协议 469
25.4.2 RS485 协议 470
25.4.3 9bit 数据传输 470
25.4.4 小数波特率支持 472
25.4.5 DMA 472
25.5 寄存器描述 473
25.5.1 寄存器列表 473
25.5.2 寄存器详细描述 474

26 控制器局域网络（CAN） 486

26.1 简介 486
26.2 结构框图 486
26.3 主要特性 487

26.4 功能描述 488
26.4.1 时钟 488
26.4.2 Bus-Off 状态 488
26.4.3 Acceptance Filters（ACF） 489
26.4.4 接收报文 489
26.4.5 发送报文 490
26.4.6 扩展状态和错误报告 491
26.4.7 扩展功能 492
26.4.8 软件复位 494
26.5 寄存器定义 497
26.5.1 寄存器列表 497
26.5.2 寄存器描述 498

27 正交编码器接口（QEI） 514

27.1 简介 514
27.2 结构框图 514
27.3 主要特性 515
27.4 功能描述 515
27.4.2 位置计数器及控制单元 517
27.4.3 边沿捕获单元 519
27.4.4 中断结构 520
27.5 寄存器描述 521
27.5.1 寄存器列表 521
27.5.2 寄存器详细描述 522

28 封装信息 536

29 订购信息 539

版本历史 540

1 文档说明

1.1 缩写词

缩写词	说明
读写(R/W)	软件可以读写这些位
只读(R)	软件只能读取这些位
只写(W)	软件只能写入该位，读取该位时将返回复位值
读清 0(RC/W)	软件可以读写该位，读取该位时，将自动清零
写清 0(R/WAC)	软件可以读写该位，写入 0 或 1 时，将自动清零
写 1 清 0(R/W1C)	软件可以读写该位，写入 1 时，将自动清零
写 0 清 0(R/W0C)	软件可以读写该位，写入 0 时，将自动清零

1.2 词汇表

本节简要介绍本文档中所用首字母缩略词和缩写词的定义：

- 在本文档中，将 Cortex™-M3 内核称为 M3
- CPU 内核集成了 SWD 调试端口：
  - SWD 调试端口(SWD-DP)提供基于串行线调试(SWD)协议的 2 引脚（时钟和数据）接口。

    有关 SWD 协议的信息，请参见《Cortex™-M3 技术参考手册》。
- 字：32 位数据/指令。
- 半字：16 位数据/指令。
- 字节：8 位数据。
- 双字：64 位数据。
- IAP（在应用中编程）：IAP 是指可以在用户程序运行期间对微控制器的 FLASH 进行重新编程。
- ICP（在线编程）：ICP 是指可以在器件安装于用户应用电路板上时使用 SWD 协议或自举程序对微控制器的 FLASH 进行编程。
- I-Code：此总线用于将 CPU 内核的指令总线连接到 FLASH 指令接口，通过此总线可执行预取操作。
- D-Code：此总线用于将 CPU 的 D-Code 总线（数据加载和调试访问）连接到 FLASH 数据接口。
- 选项字节：存储于 FLASH 中的产品配置位。
- AHB：高级高性能总线。
- CPU：指 Cortex™-M3 内核。
- FMC：FLASH Memory Controller 的简称。
- FLASH：Embedded FLASH 的简称。

1.3 相关文档

- Cortex™-M3 技术参考手册，可按下述链接下载：
  http://infocenter.arm.com/help/topic/com.arm.doc.100165_0201_00_en/arm_cortexm3_processor_trm_100165_0201_00_en.pdf

- Cortex™-M3 Device 通用用户指南(含架构、指令集、核内外设等)，可按下述链接下载：
  http://infocenter.arm.com/help/topic/com.arm.doc.dui0552a/DUI0552A_cortex_m3_dgug.pdf

2 特性列表

2.1 特性描述

- 采用 ARM Cortex™-M3 32 位内核
  - 系统工作频率 90MHz
  - 内核内置 32 位硬件乘/除法
- 硬件加速器
  - 内置 CORDIC 运算单元，支持正余弦及反正切的计算
  - 内置 IQ 除法单元，支持硬件 32 位 IQ 格式除法
- 存储资源
  - 内置 150KB 容量的 FLASH，支持代码零延迟执行、ECC 纠错及加密和读写擦保护功能
  - 内置 20KB 系统 SRAM
- 时钟、复位和电源管理
  - 支持单电源输入，输入范围为 3.1V-3.6V
  - 支持上下电复位及欠压复位
  - 内置独立看门狗(IWDG)和窗口看门狗(WWDG)
  - 内置高频 RC 振荡器：8MHz
  - 支持带晶振和无晶振方案，支持 6M-26MHz 晶体振荡器
  - 内置 2 套 PLL，支持 SSC 展频功能
  - 内置时钟安全系统，支持晶振异常监测功能
  - 内置温度传感器
- DMA 控制器
  - 内置 4 个独立通道的 DMA
  - 支持内存与外设之间任意组合传输
- 通用 I/O(GPIO)
  - 支持 I/O 功能复用映射，支持位操作
  - 所有 I/O 内建上拉/下拉电阻
  - 支持外部中断输入，支持上下边沿触发，可用于产生中断、事件唤醒，支持一路 NMI 中断
  - 内置 2 档驱动能力可调，最大 24mA 驱动能力
- 定时器(TIMER)
  - 内置 4 套基础定时器（TMR0~3）和 4 套通用定时器（TMR4~7）
  - 支持定时、PWM 输出及 Capture 捕获功能
  - 支持自动重载功能
- 增强型脉宽调制器(EPWM)
  - 支持互补/独立输出模式，支持对称/不对称波形输出
  - 内置死区及斩波插入机制，支持 3 路故障和事件输入，内置故障及异常保护功能
  - 支持多路 EPWM 同步机制
- 模数转换器(ADC)
  - 内置 2 个完全独立高速模数转换器(ADC)
  - 支持 4.4M SPS 采样速率，ADC 分辨率达到 13 位，有效位数 11 位
  - 支持 16 路模拟采样通道(每个 ADC 8 路采样通道)
  - 支持多种 SOC 触发机制，支持增益补偿与偏置补偿机制
  - 支持内置/外置 ADC 基准电压
- 数模转换器(DAC)及比较器(CMP)
  - 内置 3 路 DAC 和 CMP
  - DAC 分辨率为 12 位，INL/DNL 小于 5 个 LSB
  - 支持斜坡补偿，支持方向与幅度可编程
  - CMP 转换延迟为最小值/典型值/最大值为 15ns/18ns/21ns
- 通讯接口外设
  - 内置 2 套 UART 接口，支持 RS485 通信
  - 内置 1 套 I2C 接口，支持 10 位寻址
  - 内置 2 套 SPI 接口，支持主从模式
  - 内置 2 套 PDM 接口，支持四种滤波选择
  - 内置 1 套 CAN 控制器，支持 CAN2.0B
  - 内置 2 套 QEI 接口，支持 3 个输入通道，即 2 个相位信号和 1 个索引信号
- LVD 欠压监测功能(4 档可调)
- 128 位芯片唯一标识码

- 两种低功耗模式: 睡眠模式和停机模式
- 支持 2 线 SWD 调试接口
- 工作结温范围：-40℃ - 125℃
- 封装：LQFP80（12*12）、LQFP64（10*10）、VQFN56（7*7）

2.2 资源汇总

TAE32F5600 系列提供 56PIN、64PIN 和 80PIN 封装。

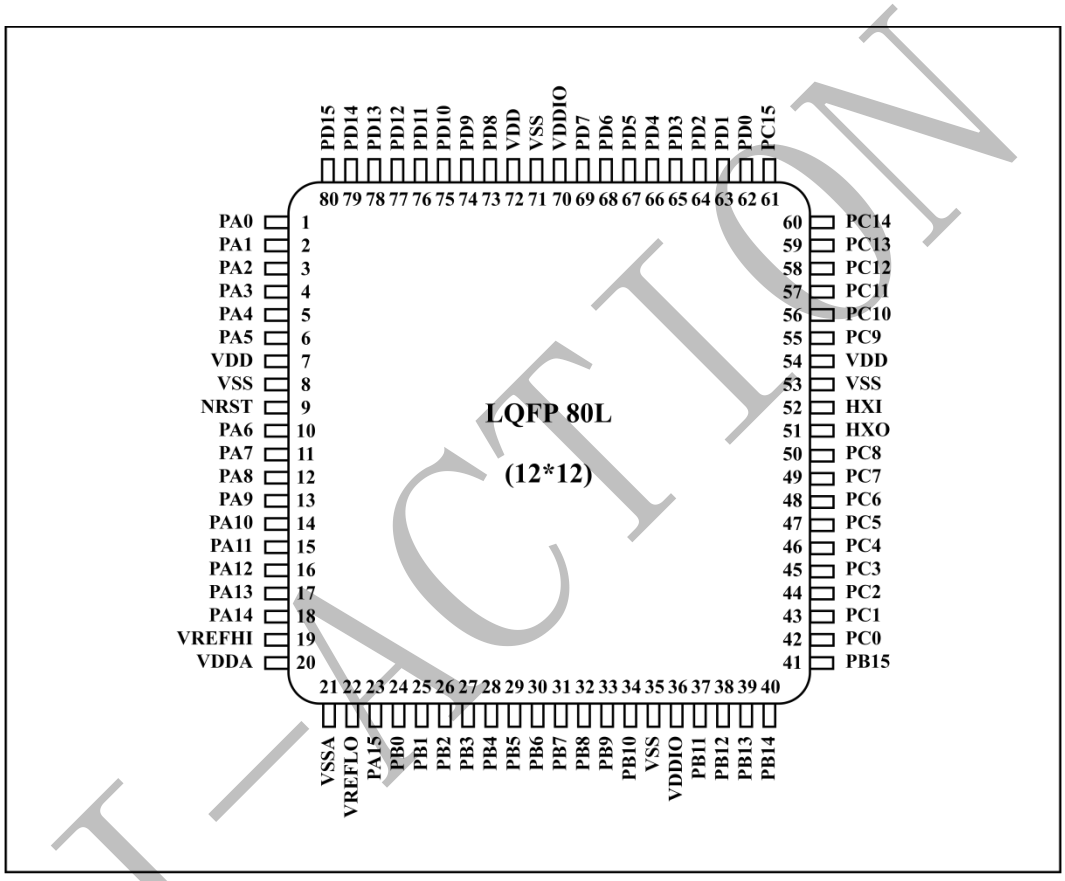
表 2-1 TAE32F5600 资源汇总

Peripheral		TAE32F5600 AVE150	TAE32F5600 ALF150	TAE32F5600 ALH150
Flash memory (Kbytes)		150	150	150
System SRAM (Kbytes)		20	20	20
Timer	Timer	8	8	8
	SysTick timer	1	1	1
	Watchdog	2	2	2
Comm. interfaces	EPWM	4	6	7
	UART	2	2	2
	I2C	1	1	1
	CAN	1	1	1
	SPI	1	1	2
	PDM	2	2	2
	QEI	2	2	2
GPIO	Normal I/Os	42	50	64
DMA channels		4	4	4
12-bit ADCs Number of channels		13	15	16
12-bit DAC channels		3	3	3
Ultra-fast analog Comparator		3	3	3
CPU frequency		90Mhz		
Operating voltage		3.1 to 3.6 V		
Operating temperature		Ambient operating temperature: - 40 to 105 °C Junction temperature: - 40 to 125 °C		
Package		VQFN 56L	LQFP 64L	LQFP 80L

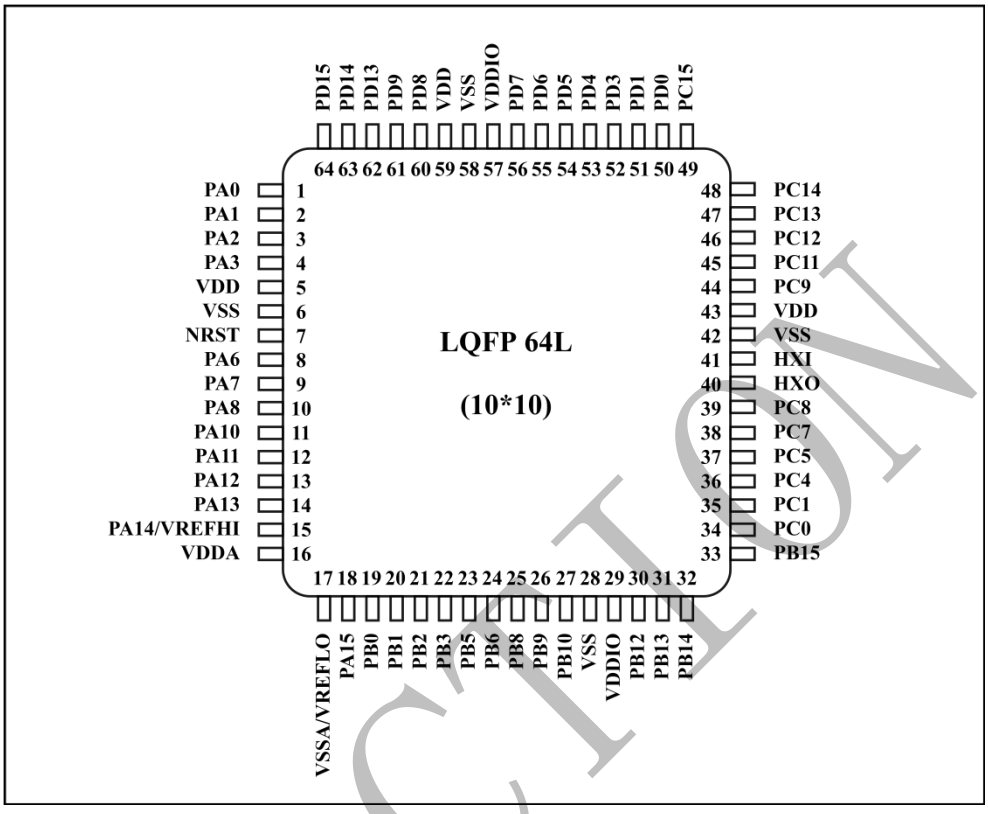
3 引脚定义

3.1 引脚框图

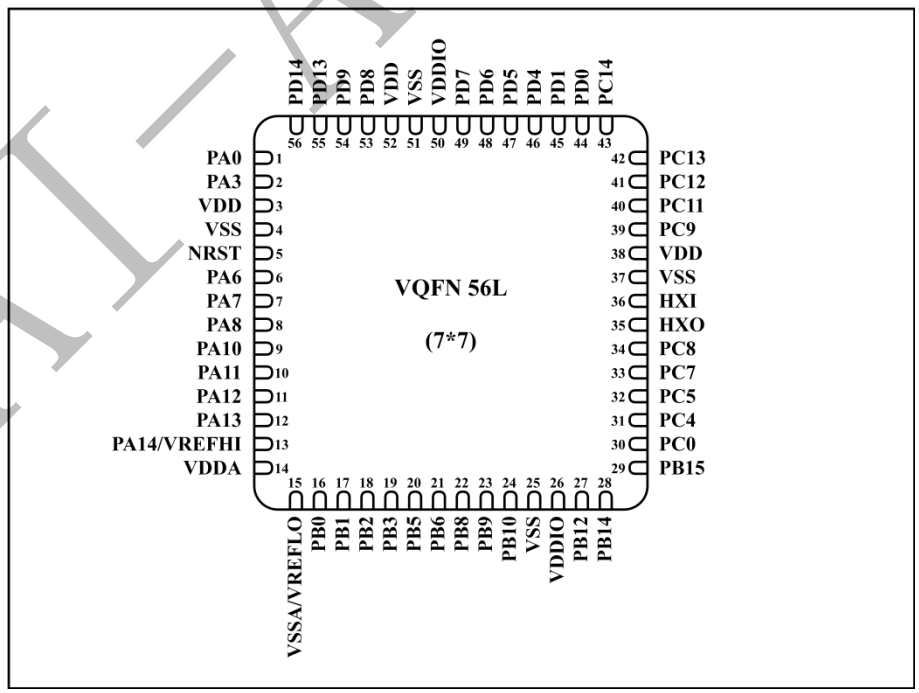
LQFP 80L:



LQFP 64L:



VQFN 56L:



3.2 引脚描述

表 3-1 引脚定义缩写说明

名称	缩写	定义
引脚名称	除非在引脚名称下使用括号特别说明，否则在复位期间和复位之后的引脚功能均与实际引脚名称相同。	
引脚类型	S	电源引脚
	SO	电源输出引脚
	I	仅输入引脚
	I/O	输入/输出引脚
引脚架构	TT	3.3V 容忍 I/O
	B	专用 BOOT 引脚
	RST	嵌入了弱上拉电阻的复位引脚
注释	除非通过注释特别说明，否则所有 I/O 在复位期间和复位之后均设置为悬空状态	
复用功能	通过 GPIOx_MUX 寄存器选择的功能	

表 3-2 引脚定义

Package				Pin type	I/O structure	Pin Function Description	
LQFP80	LQFP64	VQFN56	Pin name			Digital Functions	Analog Functions
1	1	1	PA0	I/O	TT	TMR7, I2C_SMA,UART1_TX	-
2	2	–	PA1	I/O	TT	I2C_SDA, EPWMSYNCI	-
3	3	–	PA2	I/O	TT	I2C_SCL, EPWMSYNCO	-
4	4	2	PA3	I/O	TT	QEI0_Z, UART1_RX	-
5	–	–	PA4	I/O	TT	QEI1_A, CMP0_OUT	-
6	–	–	PA5	I/O	TT	QEI1_B, CMP1_OUT	-
7	5	3	VDD	SO	-	-	-
8	6	4	VSS	S	-	-	-
9	7	5	NRST	I	RST	-	-
10	8	6	PA6	I/O	TT	QEI1_Z, TMR5	
11	9	7	PA7	I/O	TT	-	ADC0_IN7
12	10	8	PA8	I/O	TT	-	ADC0_IN6/CMP2_INP
13	-	-	PA9	I/O	TT	-	ADC0_IN5
14	11	9	PA10	I/O	TT	-	ADC0_IN4/CMP1_INP
15	12	10	PA11	I/O	TT	-	ADC0_IN3
16	13	11	PA12	I/O	TT	-	ADC0_IN2/CMP0_INP
17	14	12	PA13	I/O	TT	PDM0_DAT	ADC0_IN1
18	15	13	PA14	I/O	TT	PDM0_CLK	ADC0_IN0
19	15	13	V_{REFHI}	S	-	-	-
20	16	14	VDDA	S	-	-	-
21	17	15	VSSA	S	-	-	-
22	17	15	V_{REFLO}	SO	-	-	-

23	18	-	PA15	I/O	TT	PDM1_CLK	ADC1_IN0
24	19	16	PB0	I/O	TT	PDM1_DAT	ADC1_IN1/CMP0_INM0
25	20	17	PB1	I/O	TT	-	ADC1_IN2/CMP0_INM1
26	21	18	PB2	I/O	TT	-	ADC1_IN3/CMP1_INM0
27	22	19	PB3	I/O	TT	-	ADC1_IN4/CMP1_INM1
28	–	–	PB4	I/O	TT	-	ADC1_IN5/CMP2_INM0
29	23	20	PB5	I/O	TT	-	ADC1_IN6/CMP2_INM1
30	24	21	PB6	I/O	TT	-	ADC1_IN7
31	–	–	PB7	I/O	TT	TMR6, SPI1_CS	-
32	25	22	PB8	I/O	TT	QEI1_A, CAN_TX	-
33	26	23	PB9	I/O	TT	QEI1_B, CAN_RX	-
34	27	24	PB10	I/O	TT	I2C_SCL, UART0_TX, TZ2	-
35	28	25	VSS	S	-	-	-
36	29	26	VDDIO	S	-	-	-
37	–	–	PB11	I/O	TT	TMR5, SPI1_CLK	-
38	30	27	PB12	B	TT	-	-
39	31	–	PB13	I/O	TT	TMR5, UART1_TX, EPWM4B	-
40	32	28	PB14	I/O	TT	I2C_SDA, UART0_RX, TZ1	-
41	33	29	PB15	I/O	TT	MCO, SPI0_CLK, UART1_TX	-
42	34	30	PC0	I/O	TT	SPI0_MISO, TZ2	-
43	35	–	PC1	I/O	TT	EPWM4A	-
44	–	–	PC2	I/O	TT	SPI1_MISO	-
45	–	–	PC3	I/O	TT	QEI1_Z	-
46	36	31	PC4	I/O	TT	SPI0_MOSI, TZ1	-
47	37	32	PC5	I/O	TT	UART0_TX, SPI1_MOSI, TZ0	-
48	–	–	PC6	I/O	TT	EPWM6B	-
49	38	33	PC7	I/O	TT	UART0_RX, EPWM3B	-
50	39	34	PC8	I/O	TT	EPWMSYNCI, EPWMSYNCO, EPWM3A	-
51	40	35	HXO	O	-	-	-
52	41	36	HXI	I	-	-	-
53	42	37	VSS	S	-	-	-
54	43	38	VDD	SO	-	-	-
55	44	39	PC9	I/O	TT	CLKIN, TMR4, SPI0_CS, UART1_RX	-
56	–	–	PC10	I/O	TT	UART0_DE, UART1_DE	-
57	45	40	PC11	I/O	TT	SWCLK, CLKIN	-
58	46	41	PC12	I/O	TT	SWO	-
59	47	42	PC13	I/O	TT	TMR6	-
60	48	43	PC14	I/O	TT	SWDIO, TMR7	-

61	49	–	PC15	I/O	TT	TMR6, UART1_RX, EPWM5B	-
62	50	44	PD0	I/O	TT	TMR4, SPI0_MOSI, EPWM2B	-
63	51	45	PD1	I/O	TT	EPWM2A	-
64	–	–	PD2	I/O	TT	EPWM6A	-
65	52	–	PD3	I/O	TT	EPWM5A	-
66	53	46	PD4	I/O	TT	SPI0_MISO, CMP1_OUT, EPWM1B	-
67	54	47	PD5	I/O	TT	EPWM1A	-
68	55	48	PD6	I/O	TT	CMP0_OUT, EPWM0B	-
69	56	49	PD7	I/O	TT	EPWM0A	-
70	57	50	VDDIO	S	-	-	-
71	58	51	VSS	S	-	-	-
72	59	52	VDD	SO	-	-	-
73	60	53	PD8	I/O	TT	TRACE_CK, TMR5, QEI1_Z, TZ0	-
74	61	54	PD9	I/O	TT	TRACE_D0, UART0_DE, UART1_DE, CMP1_OUT, CMP2_OUT	-
75	–	–	PD10	I/O	TT	TRACE_D1, UART1_RX, SPI1_CS, TZ0	-
76	–	–	PD11	I/O	TT	TRACE_D2, SPI1_MISO, TZ1	-
77	–	–	PD12	I/O	TT	TRACE_D3, UART1_TX, SPI1_CLK, TZ2	-
78	62	55	PD13	I/O	TT	QEI0_A, CMP1_OUT	-
79	63	56	PD14	I/O	TT	QEI0_B, CMP2_OUT	-
80	64	-	PD15	I/O	TT	TMR4, SPI1_MOSI	-

**注意：(1) PC11/PC14 为调试引脚，PC14 数据接口内置上拉，PC11 时钟接口内置下拉。*

(2) PB12 作为BOOT 脚，默认内置下拉。

表 3-3 引脚功能复用表

Pin name	Functions							
	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
	INPUT	OUTPUT	SYS_AF	TMR/QEI	I2C/UART	UART/CAN/SPI/PDM/CMP	PWM/CMP/UART	ANALOG
PA0	-	-	-	TMR7	I2C_SMA	UART1_TX	-	-
PA1	-	-	-	-	I2C_SDA	-	EPWMSYNCI	-
PA2	-	-	-	-	I2C_SCL	-	EPWMSYNCO	-
PA3	-	-	-	QEI0_Z	-	UART1_RX	-	-
PA4	-	-	-	QEI1_A	-	CMP0_OUT	-	-
PA5	-	-	-	QEI1_B	-	CMP1_OUT	-	-
PA6	-	-	-	QEI1_Z	TMR5	-	-	
PA7	-	-	-	-	-	-	-	ADC0_IN7
PA8	-	-	-	-	-	-	-	ADC0_IN6/CMP2_INP
PA9	-	-	-	-	-	-	-	ADC0_IN5
PA10	-	-	-	-	-	-	-	ADC0_IN4/CMP1_INP
PA11	-	-	-	-	-	-	-	ADC0_IN3
PA12	-	-	-	-	-	-	-	ADC0_IN2/CMP0_INP
PA13	-	-	-	-	-	PDM0_DAT	-	ADC0_IN1
PA14	-	-	-	-	-	PDM0_CLK	-	ADC0_IN0
PA15	-	-	-	-	-	PDM1_CLK	-	ADC1_IN0
PB0	-	-	-	-	-	PDM1_DAT	-	ADC1_IN1/CMP0_INM0
PB1	-	-	-	-	-	-	-	ADC1_IN2/CMP0_INM1
PB2	-	-	-	-	-	-	-	ADC1_IN3/CMP1_INM0
PB3	-	-	-	-	-	-	-	ADC1_IN4/CMP1_INM1
PB4	-	-	-	-	-	-	-	ADC1_IN5/CMP2_INM0
PB5	-	-	-	-	-	-	-	ADC1_IN6/CMP2_INM1
PB6	-	-	-	-	-	-	-	ADC1_IN7
PB7	-	-	-	TMR6	-	SPI1_CS	-	-
PB8	-	-	-	QEI1_A	-	CAN_TX	-	-

PB9	-	-	-	QEI1_B	-	CAN_RX	-	-
PB10	-	-	-	-	I2C_SCL	UART0_TX	TZ2	-
PB11	-	-	-	TMR5	-	SPI1_CLK	-	-
PB12	-	-	-	-	-	-	-	-
PB13	-	-	-	TMR5	-	UART1_TX	EPWM4B	-
PB14	-	-	-	-	I2C_SDA	UART0_RX	TZ1	-
PB15	-	-	MCO	-	-	SPI0_CLK	UART1_TX	-
PC0	-	-	-	-	-	SPI0_MISO	TZ2	-
PC1	-	-	-	-	-	-	EPWM4A	-
PC2	-	-	-	-	-	SPI1_MISO	-	-
PC3	-	-	-	QEI1_Z	-	-	-	-
PC4	-	-	-	-	-	SPI0_MOSI	TZ1	-
PC5	-	-	-	-	UART0_TX	SPI1_MOSI	TZ0	-
PC6	-	-	-	-	-	-	EPWM6B	-
PC7	-	-	-	-	UART0_RX	-	EPWM3B	-
PC8	-	-	-	-	EPWMSYNCI	EPWMSYNCO	EPWM3A	-
PC9	-	-	CLKIN	TMR4	-	SPI0_CS	UART1_RX	-
PC10	-	-	-	UART0_DE	UART1_DE	-	-	-
PC11[(1)]	-	-	SWCLK	CLKIN	-	-	-	-
PC12	-	-	SWO	-	-	-	-	-
PC13	-	-	-	TMR6	-	-	-	-
PC14[(1)]	-	-	SWDIO	TMR7	-	-	-	-
PC15	-	-	-	TMR6	-	UART1_RX	EPWM5B	-
PD0	-	-	-	TMR4	-	SPI0_MOSI	EPWM2B	-
PD1	-	-	-	-	-	-	EPWM2A	-
PD2	-	-	-	-	-	-	EPWM6A	-
PD3	-	-	-	-	-	-	EPWM5A	-
PD4	-	-	-	-	SPI0_MISO	CMP1_OUT	EPWM1B	-
PD5	-	-	-	-	-	-	EPWM1A	-

PD6	-	-	-	-	-	CMP0_OUT	EPWM0B	-
PD7	-	-	-	-	-	-	EPWM0A	-
PD8	-	-	TRACE_CK	TMR5	QEI1_Z	-	TZ0	-
PD9	-	-	TRACE_D0	UART0_DE	UART1_DE	CMP1_OUT	CMP2_OUT	-
PD10	-	-	TRACE_D1	-	UART1_RX	SPI1_CS	TZ0	-
PD11	-	-	TRACE_D2	-	-	SPI1_MISO	TZ1	-
PD12	-	-	TRACE_D3	-	UART1_TX	SPI1_CLK	TZ2	-
PD13	-	-	-	QEI0_A	-	-	CMP1_OUT	-
PD14	-	-	-	QEI0_B	-	-	CMP2_OUT	-
PD15	-	-	-	TMR4	-	SPI1_MOSI	-	-

**注意：(1) PC11/PC14 为调试引脚，PC14 数据接口内置上拉，PC11 时钟接口内置下拉。*

(2) PB12 作为BOOT 脚，默认内置下拉。

4 存储器和总线架构

4.1 系统架构

芯片主系统由多层 AHB 总线矩阵互联，如下图所示：

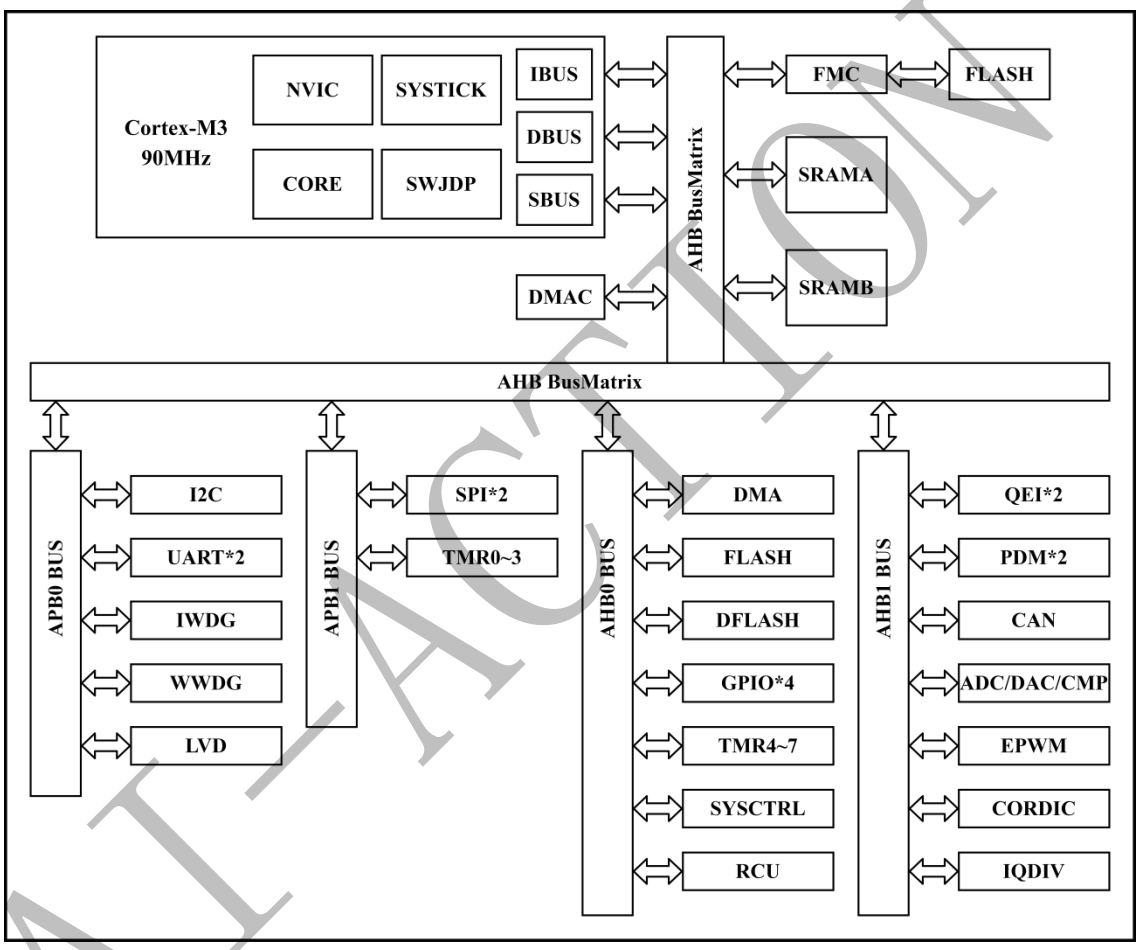


图 4-1 总线架构

4.2 总线矩阵

主系统由 32 位多层 AHB 总线矩阵构成，可实现以下部分的互连：

- 五条主控总线：
  - Cortex™-M3 内核 I 总线、D 总线和 S 总线
  - DMA 总线 0
  - DMA 总线 1
- 八条被控总线：
  - FLASH ICode 总线
  - FLASH DCode 总线

- SRAMA(10KB)
- SRAMB(10KB)
- AHB0 外设
- AHB1 外设
- APB0 外设
- APB1 外设

通过总线矩阵可以实现主控总线到被控总线的访问，这样即使在多个高速外设同时运行期间，系统也可以实现并发访问和高效运行，此架构如图 4-2 所示。

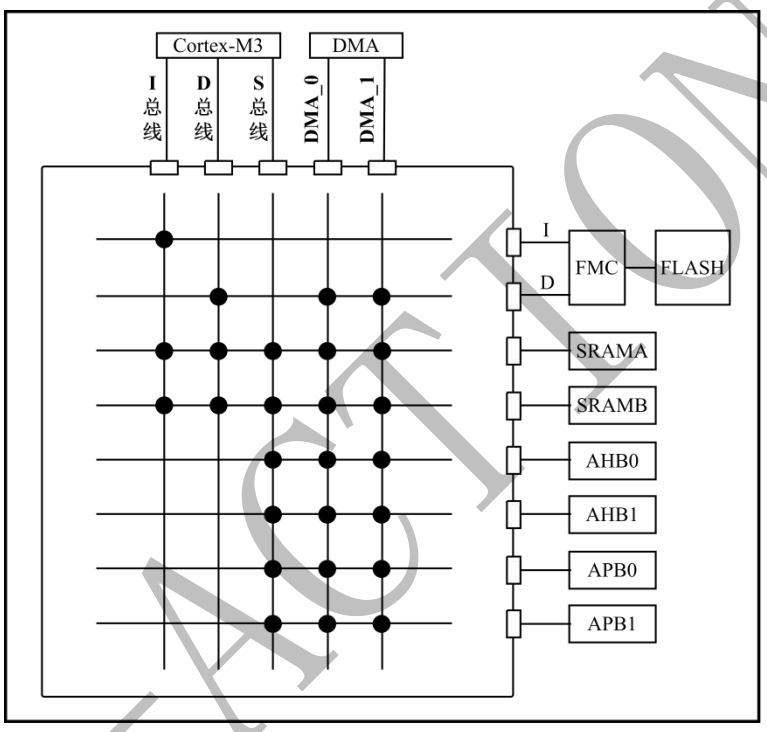


图 4-2　系统总线互连矩阵

I 总线：此总线用于将 Cortex™-M3 内核的指令总线连接到总线矩阵，内核通过此总线获取指令。此总线访问的对象是包含代码的存储器（内部 FLASH/SRAM）。

D 总线：此总线用于将 Cortex™-M3 数据总线连接到总线矩阵，内核通过此总线进行立即数加载和调试访问。此总线访问的对象是包含代码或数据的存储器（内部 FLASH/SRAM）。

S 总线：此总线用于 Cortex™-M3 内核的系统总线连接到总线矩阵。此总线用于访问位于外设或 SRAM 中的数据，也可通过此总线获取指令（效率低于 ICode）。此总线访问的对象是内部 SRAMA/B 及 APB0/1 和 AHB0/1 外设。

DMA 总线：此总线用于将 DMA 模块访问接口连接到总线矩阵，DMA 通过此总线时序存储器与存储器之间、存储器与外设之间或者外设与外设之间的数据互传。此总线访问的对象是数据存储器：包括内部 FLASH/SRAM、APB0/1 及 AHB0/1 外设。

总线矩阵：总线矩阵用于主控总线之间的访问仲裁管理。

4.3 存储器地址映射

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个顺序的 4GB 地址空间内。

各字节按小端格式在存储器中编码，字中编号最低的字节被视为该字的最低有效字节，而编号最高的字节被视为最高有效字节。

未分配给片上存储器和外设的所有存储区域均视为“保留区”，有关外设寄存器映射的详细信息，请参见“4.4 外设寄存器映射”章节。

遵循 ARM® Cortex™-M3 对存储器的规定，存储器基本组织架构如下：

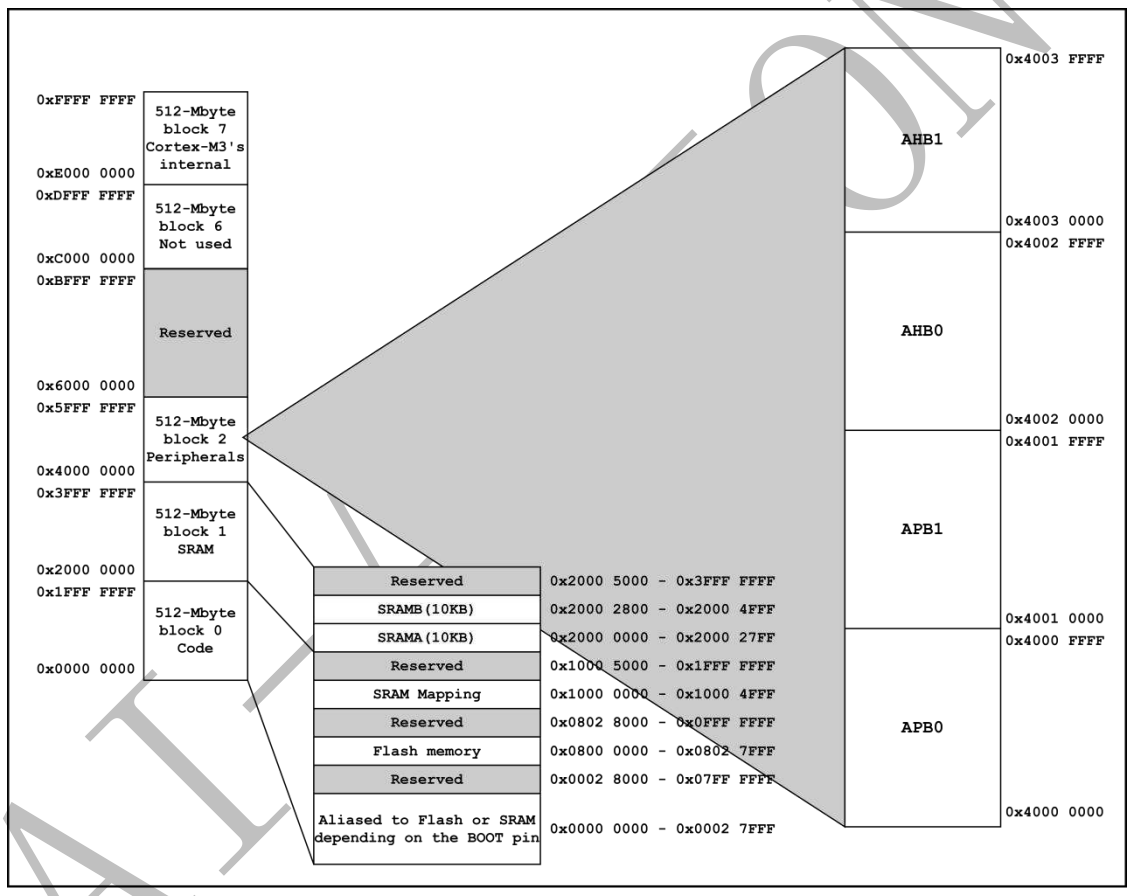


图 4-3 存储映射图

4.4 外设寄存器映射

下表提供了芯片外设地址边界划分，详细请参考表 4-1：

表 4-1 芯片外设地址映射表

总线	边界地址	外设
Reserved	0xE010 0000 - 0xFFFF FFFF	Reserved
Cortex™-M3	0xE000 0000 - 0xE00F FFFF	Cortex™-M3 internal peripherals
AHB1	0x4003 F000 - 0x4003 FFFF	IQDIV
	0x4003 E000 - 0x4003 EFFF	CORDIC

	0x4003 A600 - 0x4003 A6FF	EPWM6
	0x4003 A500 - 0x4003 A5FF	EPWM5
	0x4003 A400 - 0x4003 A4FF	EPWM4
	0x4003 A300 - 0x4003 A3FF	EPWM3
	0x4003 A200 - 0x4003 A2FF	EPWM2
	0x4003 A100 - 0x4003 A1FF	EPWM1
	0x4003 A000 - 0x4003 A0FF	EPWM0
	0x4003 9200 - 0x4003 92FF	DACCMP2
	0x4003 9100 - 0x4003 91FF	DACCMP1
	0x4003 9000 - 0x4003 90FF	DACCMP0
	0x4003 8000 - 0x4003 8FFF	ADC
	0x4003 3000 - 0x4003 30FF	CAN
	0x4003 2100 - 0x4003 21FF	PDM1
	0x4003 2000 - 0x4003 20FF	PDM0
	0x4003 1000 - 0x4003 1FFF	QEI1
	0x4003 0000 - 0x4003 0FFF	QEI0
AHB0	0x4002 F000 - 0x4002 FFFF	SYSCTRL
	0x4002 E000 - 0x4002 EFFF	RCU
	0x4002 7400 - 0x4002 74FF	TMRxG[x=4~7]
	0x4002 7300 - 0x4002 73FF	TMR7
	0x4002 7200 - 0x4002 72FF	TMR6
	0x4002 7100 - 0x4002 71FF	TMR5
	0x4002 7000 - 0x4002 70FF	TMR4
	0x4002 6000 - 0x4002 6FFF	GPIOD
	0x4002 5000 - 0x4002 5FFF	GPIOC
	0x4002 4000 - 0x4002 4FFF	GPIOB
	0x4002 3000 - 0x4002 3FFF	GPIOA
	0x4002 2000 - 0x4002 2FFF	DFLASH
	0x4002 1000 - 0x4002 1FFF	FLASH
	0x4002 0000 - 0x4002 0FFF	DMA
APB1	0x4001 4400 - 0x4001 44FF	TMRxG[x=0~3]
	0x4001 4300 - 0x4001 43FF	TMR3
	0x4001 4200 - 0x4001 42FF	TMR2
	0x4001 4100 - 0x4001 41FF	TMR1
	0x4001 4000 - 0x4001 40FF	TMR0
	0x4001 1000 - 0x4001 1FFF	SPI1
	0x4001 0000 - 0x4001 0FFF	SPI0
APB0	0x4000 A000 - 0x4000 AFFF	LVD
	0x4000 9000 - 0x4000 9FFF	WWDG
	0x4000 8000 - 0x4000 8FFF	IWDG
	0x4000 3000 - 0x4000 3FFF	UART1
	0x4000 2000 - 0x4000 2FFF	UART0
	0x4000 0000 - 0x4000 0FFF	I2C0

4.4.1 SRAM

系统 SRAM 可按字节、半字（16 位）或全字（32 位）访问，读写操作以 CPU 速度执行，且执行等待周期为 0，系统 SRAM（SRAMA/B）映射在地址 0x2000 0000 和地址 0x1000 0000 上，地址 0x1000 0000 可通过 I/D-Code 总线访问；地址 0x2000 0000 可供 CPU 系统总线及 DMA 访问。

如果选择从 SRAM 自举，则 CPU 可通过系统总线或 I/D-Code 总线访问系统 SRAM。

4.4.2 FLASH

Embedded FLASH Controller 接口用于管理 CPU 通过 I 总线(I-Code Bus)及 D 总线(D-Code Bus)对 Embedded FLASH 的访问。通过 Embedded FLASH Controller 可以对 FLASH 存储体执行擦除与编程操作，并对 FLASH 存储体实现读/写保护机制及存储体数据的加/解密功能。

FLASH 结构如下：

- 主存储器块分为多个扇区。
- FLASH 作为程序存储器，芯片可在 FLASH 自举启动。
- 选项字节，用于配置读保护、写保护及加密等功能。

4.5 自举配置

存储器采用固定的存储器映射，代码区域起始地址为 0x0000 0000（通过 ICode/DCode 总线访问），而数据区域起始地址为 0x2000 0000（通过系统总线访问）。Cortex™-M3 CPU 复位启动时通过 ICode 总线获取复位向量，这就意味着只有代码区域（通常为 FLASH）可以提供自举空间，也可通过修改芯片自举配置，让芯片从其它存储器（如内部 SRAM）进行自举。

在芯片中，可通过 BOOT 引脚选择两种不同的自举模式，如表 4-2 所示。

表 4-2 自举模式

自举模式选择引脚	自举模式	自举空间
BOOT（PB12）		
0	FLASH	选择 FLASH 作为自举空间
1	SRAM	选择 SRAM 作为自举空间

复位后，在 FCLK 的第四个上升沿锁存 BOOT 引脚的值。上电时，用户可以通过设置 BOOT 引脚的电平来选择需要的自举模式。

BOOT 引脚只有在芯片重新上电时，才会对 BOOT 引脚重新采样并锁存。上电启动结束后，CPU 将从 0x0000 0000 地址获取栈顶值，然后从 0x0000 0004 地址的自举存储器开始执行代码。

注意：如果器件从 SRAM 自举，在应用程序初始化代码中，需要使用 NVIC 程序及中断向量表和偏移寄存器来重新分配 SRAM 中的向量表。

5 嵌入式 FLASH（FLASH）

5.1 简介

FLASH 控制器用于管理 CPU 通过 I 总线(I-Code Bus)及 D 总线(D-Code Bus)对 FLASH 的访问。通过 FLASH 控制器可以对 FLASH 存储体执行编程与擦除操作、对 FLASH 存储体实现读/写保护机制及对 FLASH 存储体中数据的加/解密功能。

5.2 结构框图

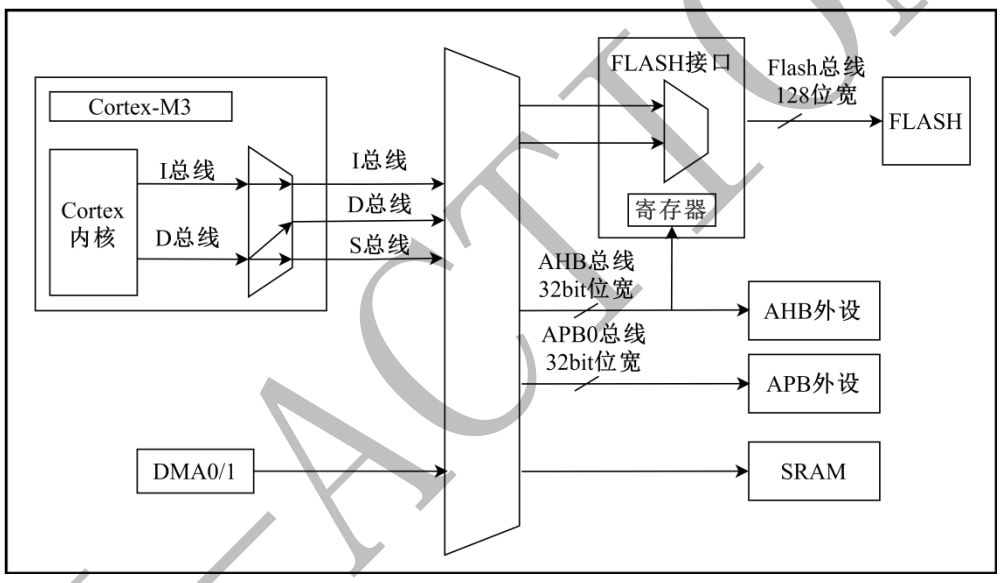


图 5-1 FLASH 顶层结构框图

5.3 主要特性

FLASH 主要具有以下特性：

- 150KByte 存储空间
- 高达 320 扇区，每个扇区 480 字节
- 支持按整片/扇区擦除(Chip/Sector Erase)
- FLASH 作为启动代码存储空间(默认的自举存储)和执行用户代码
- FLASH 支持 128 位宽编程功能，支持按字节/半字/字读取 FLASH
- 增强安全功能
  - FLASH 读保护功能(RDP)
  - FLASH 写保护功能(WRP)
  - FLASH 存储体加密存储，实现对用户代码的保护
- FLASH I/D 总线 128Bit 位宽预取、缓存功能
  - FLASH I-Code Bus 上 512 Byte 缓存
  - FLASH D-Code Bus 上 256 Byte 缓存
- 误码校验(ECC)，支持纠一检错
- 支持低功耗模式

5.4 功能描述

5.4.1 实时加速器

为了高效地发挥处理器的性能，通过加速器实现指令预取队列及分支缓存机制，减少 FLASH 读取指令时等待的情况，从而提高了 128 位 FLASH 程序的执行速度。

指令预取

FLASH 每次读操作可读取 128 位数据，即可以是 4 条 32 位指令，也可以是 8 条 16 位指令，具体取决于烧写在 FLASH 中的程序。因此对于顺序执行的代码，至少需要 4 个 CPU 周期来执行前一次读取的 128 位指令行。

在 CPU 读取当前指令行时，可使用 I-Code 总线的预取操作读取 FLASH 中的下一个连续存放的 128 位指令行。可通过 FLASH_ECR 寄存器中的 DBPE/ IBPE 位置 1 来使能预取功能。

图 5-2、图 5-3 为需要 3 个等待周期访问 FLASH 时连续 32 位指令的执行过程，分别介绍了不使用和使用预取操作两种情况。

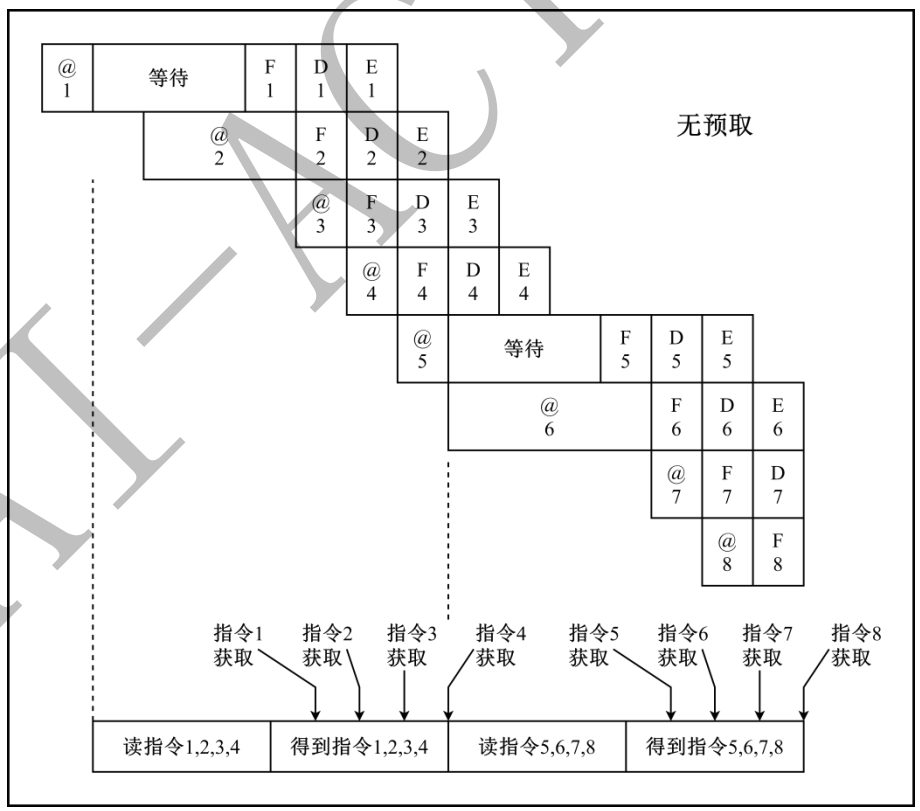


图 5-2　连续 32 位指令的执行过程（不使用预取操作）

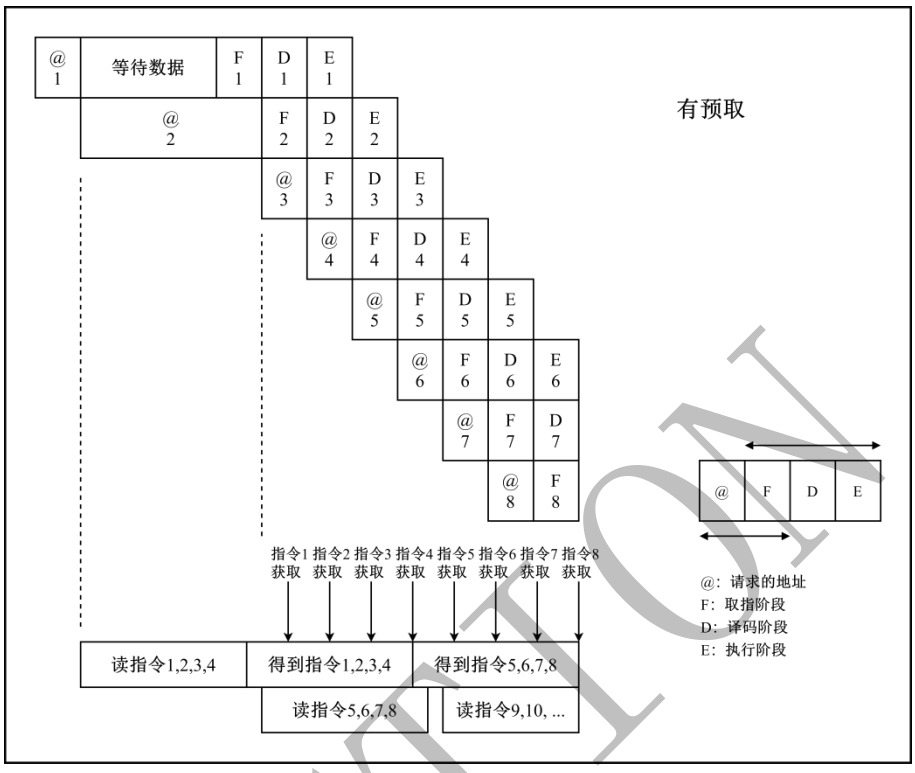


图 5-3　连续 32 位指令的执行过程（使用预取操作）

处理非顺序执行的代码（有分支）时，指令可能并不存在于当前使用的或预取的指令行中。 这种情况下，CPU 等待时间为 FLASH 重新取值的周期数。

指令缓存存储器

为了减少因指令跳转而产生的时间消耗，可将 32 行 128 位(512 Byte)的指令保存到指令缓存(I-Cache)存储器中。每当出现指令缺失（即请求的指令未存在于当前使用的指令行、预取指令行或指令缓存存储器中）时，系统会将新读取的行复制到指令缓存存储器中。如果 CPU 请求的指令已存在于指令缓存区中，则无需任何延时即可立即获取。指令缓存存储器存满后，可采用最近最少使用策略确定指令缓存存储器中待替换的指令行。此特性非常适用于包含循环的代码。

数据管理

在 CPU 流水线执行阶段，将通过 D-Code 总线访问 FLASH 中的数据缓冲池。因此，直到提供了请求的数据后，CPU 流水线才会继续执行。为了减少因此而产生的时间损耗，通过 D-Code 数据总线的访问优先于通过 I-Code 指令总线的访问。

如果频繁使用某些数据，同样添加了数据缓存(D-Cache)功能，此特性的工作原理与指令缓存存储器类似，但保留的数据大小限制在 16 行 128 位(256 Byte)以内。

5.4.2 擦除和编程操作

执行任何 FLASH 编程操作（擦除或编程）时，FLASH 的工作时钟频率 (HCLK)不能低于 1MHz。如果在 FLASH 操作期间发生器件复位，无法保证 FLASH 中的内容，建议将系统时钟频率配置为 90MHz。

在对 FLASH 执行写入或擦除操作期间，任何读取 FLASH 的操作都会导致总线阻塞。只有在完成编程操作后，才能正确处理读操作。这意味着，写入/擦除操作进行期间不能从 FLASH 中执行代码或数据获取操作。

注意：FLASH 所有操作均为串行执行，确保上一笔操作完成后再发起下一笔操作。

5.4.2.1 FLASH 控制寄存器解锁

FLASH 中用于解锁控制寄存器的 Key 值有以下几种：

- KEY1 = 0x7654_3210
- KEY2 = 0xFEDC_BA98
- OPTKEY= 0xF7BC_4A12

FLASH 控制寄存器(FLASH_CR)解锁

复位后，FLASH 控制寄存器(FLASH_ECR)不允许执行写/擦操作（读不需要解锁），以防因电气干扰等原因出现对 FLASH 的意外操作。当需要执行 FLASH 的编程/擦写操作时，必须先执行解锁，然后发起相应的操作。

- 解锁方式

向 FLASH 密钥寄存器(FLASH_KEY)中依次写入 KEY1 和 KEY2，成功解锁后，FLASH 控制寄存器的解锁标志位(FLASH_CR.LOCK)将会自动清零，即可对 FLASH 进行编程/擦写操作。

- 加锁方式

完成编程/擦除操作后，通过软件将 FLASH 控制寄存器的解锁标志位(FLASH_CR.LOCK)位置 1，将重新锁定 FLASH，实现加锁功能，下次再进行编程/擦写操作前，需要重新进行解锁操作。

注意：

- *进行写/擦除操作前，必须对 FLASH 进行解锁，否则将导致写/擦除操作无效；*
- *解锁过程必须严格按照解锁顺序，如果顺序错误，则会解锁失败，*寄存器的解锁标志位*(FLASH_CR.LOCK)将不会清除，解锁无效；*
- *建议在完成目标操作后，对 FLASH 进行加锁，避免意外操作。*

FLASH 读/写保护寄存器(FLASH_RDPR/FLASH_WRPR)解锁

在对 FLASH 进行读/写保护修改前，需要写入特定的密钥进行操作解锁，避免意外的读/写保护产生。

- 解锁方式
  向 FLASH 密钥寄存器(FLASH_KEY)写入 OPTKEY 值即可解锁。
- 加锁方式
  向 FLASH 密钥寄存器(FLASH_KEY)写入其他非 OPTKEY 值即重新加锁。

注意：读/写保护解锁只要 FLASH 密钥寄存器(FLASH_KEY)保持 OPTKEY 值，即可对读/写保护进行修改，直至 FLASH_KEY 写入其他非 OPTKEY 值。因此建议在完成读/写保护的修改后，对 OPTKEY 写入其他值(推荐写 0)加锁，避免意外操作。

FLASH 低功耗寄存器（FLASH_LPR）解锁

FLASH 支持低功耗模式，可以按需进入/退出低功耗模式，降低系统功耗。操作前需要对低功耗操作进行解锁，避免意外的操作。

- 解锁方式
  向 FLASH 低功耗寄存器的解锁标志位(FLASH_LPR.LOCK)写 1 即可解锁操作。
- 加锁方式
  向 FLASH 低功耗寄存器的低功耗控制位(FLASH_LPR.WKUP/STDBY)写 1 发起进入/退出低功耗操作，操作完成后将自动加锁。

5.4.2.2 FLASH 擦除

FLASH 擦除操作可针对扇区擦除(Sector Erase)或整个闪存擦除 (或称整片擦除，Chip Erase)执行。

注意：擦除操作，不会解除 FLASH 的读/写保护功能。

扇区擦除(Sector Erase)

扇区擦除的具体步骤如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 将 FLASH_ECR 寄存器中的 EMODE 位置 0；
3. 将目标扇区号写入 FLASH_ECR[8:0]中；
4. 将 FLASH_CR 寄存器中的 ES 位置 1；
5. 等待 BSY 位清零，擦除动作完成。

注意：扇区擦除的扇区号，不得超过 FLASH 的扇区范围，否则状态寄存器(FLASH_ISR)中的操作错误标志位(OPTEIF)和写保护错误标志位(WPEIF)将置 1 (如果使能中断，将导致触发错误中断)

闪存擦除(Chip Erase)

要执行批量擦除，建议采用以下步骤：

1. 检查 FLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 在 FLASH_ECR 寄存器中，将 EMODE 位置 1；
3. 将 FLASH_CR 寄存器中的 ES 位置 1；
4. 等待 BSY 位清零，擦除动作完成。

5.4.2.3 FLASH 编程

FLASH 的编程顺序如下：

1. 检查 FLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 向 FLASH_DRx(x = 0 … 3) 写入需要编程的 128bit 数据；
3. 向 FLASH_ADDR 中写入编程起始地址；
4. 将 FLASH_CR 寄存器中的 PS 位置 1；
5. 等待 BSY 位清零，编程操作完成。

技巧：把 FLASH 的单元从逻辑“1”写为逻辑“0”时，可以无需执行擦除操作即可进行写操作，这样可以减少 FLASH 擦写次数。但把 FLASH 单元从逻辑“0”写为逻辑“1”时，则必须先执行 FLASH 擦除操作。

注意：

- *如果同时发出擦除和编程操作请求时(ES 和 PS 同时置位)，将认为无效操作，不会执行任何操作，需重新发起对应操作；*
- *当 FLASH_SR 寄存器中的 BSY 位为 1 时，将不能对 FLASH 再次发起其他操作，否则会导致 FLASH 异常，必须等到 BSY 位清零才能发起下一笔操作；*
- *FLASH 写操作位宽必须按照 128 位对齐，若发起非对齐操作将不会执行，并且状态寄存器(FLASH_ISR)中的操作错误标志位(OPTEIF)将置 1，并引发中断；*
- *FLASH 写操作必须处于有效的地址范围内，不能越界。否则操作将不会执行，并且状态寄存器(FLASH_ISR)中的操作错误标志位(OPTEIF)和写保护错误标志位(WPEIF) 将置 1，并引发中断。*

编程与缓存(Cache)

如果 FLASH 编程操作涉及数据缓存(Data Cache)中的某些数据，FLASH 的写操作将修改 FLASH 存储中的数据，同时更新数据缓存中的数据。

如果 FLASH 中的擦除操作也涉及数据或指令缓存(D-Cache/I-Cache)中的数据，则也会同样实现缓存更新操作，将缓存进行初始化，保证与 FLASH 存储数据一致。

5.4.2.4 FLASH ECC 校验

由于 FLASH 存储数据受电子干扰或其他因素导致数据错误，这样会导致系统工作异常。为了提高系统稳定性及芯片可靠性，对 FLASH 加入了 ECC 校验功能，采用 8bit 纠 128bit 的 ECC

策略，可以实现 128bit 纠 1bit。

对 FLASH 发起 Program 写操作时，ECC 功能模块会自动对 128bit 写入数据进行 ECC 编码计算，生成 ECC 的校验码，同时将 Program 数据和 ECC 校验码写入 FLASH 中；当进行读操作时，将读出数据和 ECC 校验码同时读回，并进行 ECC 校验，如果存在 1bit 错误会自动纠正，同时产生 ECC 错误(置位 FLASH_ISR 寄存器中的 ECCEIF)，并触发中断(如果使能 EIE 中断)。

注意：内部 ECC 功能只能纠正 1bit 错误，超过 1bit 错误无法纠回。

5.4.2.5 FLASH 加密功能

为了提高芯片的安全特性，加强对用户数据的保护，增加了 FLASH 加密特性，并对指令代码实现 0 等待获取，因而不会对 FLASH 执行代码效率造成影响。

当对 FLASH 发起 Program 写操作时，加密单元会对写入数据执行加密操作，最终将加密后的数据写入 FLASH 存储体；当 CPU 读取 FLASH 时，读返回的数据也会经过加密单元进行解密操作，然后返回到 CPU。

5.4.2.6 FLASH 中断

FLASH 有下述几种情况将会引发中断，FLASH_ISR 寄存器中相应的中断标志位将会置位，通过对相应的位置进行写 1，将清除相应的中断标志位。

表 5-1 引发 FLASH 中断的情况

中断事件	中断使能	中断标志	备注
操作完成	DIE	DIF	编程完成、擦除完成、低功耗操作完成
操作错误	EIE	OPTEIF	编程地址不对齐、扇区擦除编号超过扇区范围、编程地址超过闪存地址范围
读保护错误	EIE	RPEIF	读保护模式，并在调试模式下对 FLASH 进行擦除/编程操作；读操作还将同时产生 BusFault 异常中断
写保护错误	EIE	WPEIF	扇区擦除/编程区域命中带写保护的区域
ECC 错误	EIE	ECCEIF	ECC 校验到错误

5.4.2.7 FLASH 读保护(RDP)

对 FLASH 中的用户代码区域实施读保护机制，防止用户代码泄露。读保护分三个级别。

级别 0(RDP Level 0)：无读保护

将 0xAA 写入读保护寄存器 (FLASH_RDPR[7:0]) 时，读保护级别即设为 0。此时，在所有自举配置（FLASH 用户自举、调试或从 RAM 自举）中，均可执行对 FLASH 的读取/编程操作（如果未设置其他保护）。

级别 1(RDP Level 1)：存储器读保护(调试功能受限)

将任意值(分别用于设置级别 0 和级别 2 的 0xAA 和 0xCC 除外) 写入读保护寄存器 (FLASH_RDPR[7:0])时，即激活读保护级别 1。设置读保护级别 1 后：

- 在连接调试功能或从 RAM 自举时，则不允许 FLASH 进行访问(读取、擦除、编程)。如果发起读操作请求，将导致总线错误(BusFault 异常)。
- 从 FLASH 自举时(未连接调试器)，允许通过用户代码对 FLASH 进行访问(读取、擦除、编程)。
- 激活级别 1 后，如果修改读保护寄存器回退到级别 0(RDP 降级)，则将对 FLASH 执行闪存擦除（在读保护降级为级别 0 前，进行擦除动作）。

注意：

- *如果通过代码执行闪存擦除(Chip Erase)操作，将仅擦除闪存区域，读/写保护配置将不受影响，闪存擦除动作之后，依然保持原来的设置。*
- *只有在已激活级别 1 并回退到级别 0 时，才会执行闪存擦除，当提高保护级别时，不会执行擦除。*

级别 2(RDP Level 2)：禁止调试/芯片读保护

将 0xCC 写入 RDP 选项字节时，可激活读保护级别 2。设置读保护级别 2 后：

- 级别 1 提供的所有保护均有效
- 调试口(SW)处于禁止状态，RAM 自举也将无法通过调试口载入代码
- 从 FLASH 自举时，允许用户代码对 FLASH 进行读取、擦除、编程操作
- 读保护级别 2 的设置不可逆，无法再降级回到级别 0 或级别 1

不同读保护级别下的访问限制

表 5-2　不同读保护级别下的访问限制

存储区	保护级别	从 FLASH 自举			从 SRAM 自举或连接调试器		
		读	写	擦除	读	写	擦除
FLASH Main 区域	级别 1	Y			N		
	级别 2	Y			N（1）		

注意：（1）读保护级别设置为级别 2 时，调试口将被禁止。

不同保护级别之间的转换方案

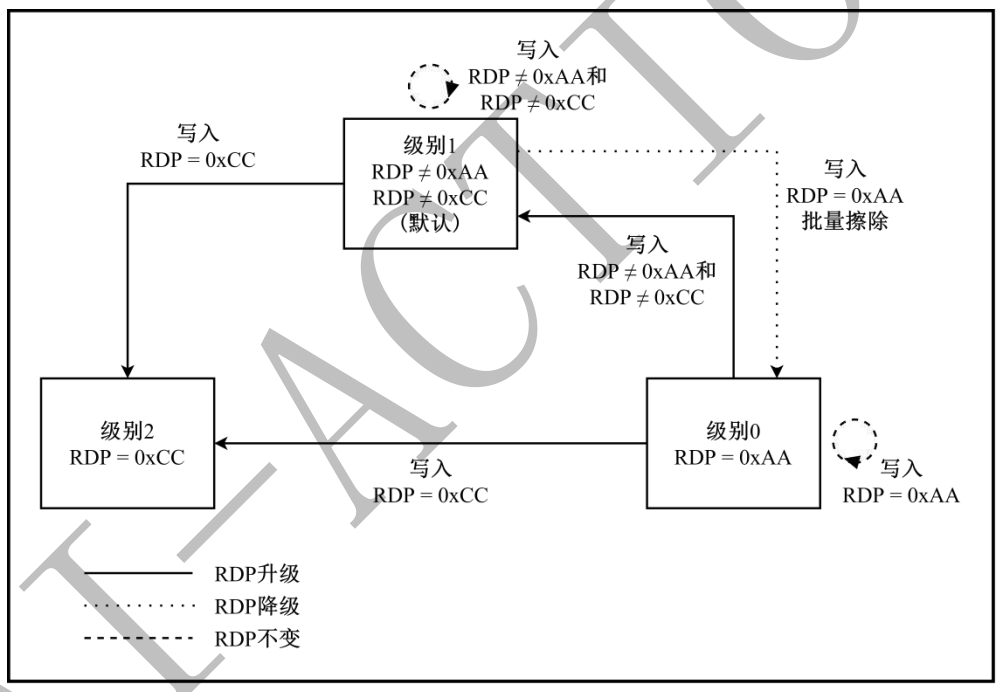


图 5-4　RDP 级别切换

图 5-4 为 RDP 级别切换的过程。

注意：

- *只有级别 1 才能回退到级别 0，并导致闪存擦除(Chip Erase)*
- *级别 2 不能回退到级别 1 或级别 0*

5.4.2.8 FLASH 写保护(WRP)

FLASH Main 区域都具备写保护功能，用于避免 FLASH 的非易失性代码、数据被意外修改。FLASH 16 个扇区为一组，写保护寄存器（FLASH_WRPR[19 : 0]）中每个 bit 对应一组 FLASH 存储扇区的写保护功能，对该寄存器的写保护位的清 0/置 1，可以为相应的扇区执行/去除写保护功能。

受到写保护的扇区，既不能对保护扇区进行扇区擦除、也不能编程。如果对 FLASH 中处于写保护状态的区域执行扇区擦除/编程操作，则 FLASH_ISR 寄存器中的写保护错误标志位(WPEIF)以及操作错误标志位(OPTEIF)将置 1。

注意：

- *如果执行闪存擦除(Chip Erase)，将无视写保护功能，对整个闪存执行擦除。*
- *如果设置了读保护级别 1，并连接了调试器进行调试，即使指定扇区未设置写保护，也无法对相应的区域进行编程/擦除(包括扇区擦除和闪存擦除)操作。*
- *如果当前处于读保护级别 1，并执行读保护级别 1 退回到级别 0，即使设置了写保护，也同样发起闪存擦除操作，将整片 FLASH 擦除。*
- *在读保护处于级别 2 时，将与级别 1 的一致，区别为读保护级别 2 不能通过调试器访问芯片。*

写保护错误标志

如果对 FLASH 的写保护区域执行擦除/编程操作，则 FLASH_ISR 寄存器中的写保护错误标志位 WPEP 将置 1。

如果请求执行擦除操作，则以下情况下 WPE/OEP 位置 1：

- 配置扇区擦除（SER = 1）导致 WPE 置 1
- 请求执行扇区擦除但扇区编号字段无效导致 OEP 置 1
- 请求针对写保护扇区执行扇区擦除
- 当 FLASH 处于读保护状态，但企图发起擦除操作

如果请求执行编程操作，则以下情况下 WPE/OEP 位置 1：

- 针对系统存储器或用户特定扇区的保留区域执行写操作
- 针对用户配置扇区执行写操作
- 针对通过选项位实施写保护的扇区执行写操作
- 当 FLASH 处于读保护状态，但企图发起写操作

5.5 寄存器描述

5.5.1 寄存器列表

FLASH 基地址：**0x4002 1000**

偏移	实例地址	名称	默认值	描述
0x00	0x40021000	FLASH_CR	0x80000300	FLASH 控制寄存器
0x04	0x40021004	FLASH_LPR	0x00000000	FLASH 低功耗寄存器
0x08	0x40021008	FLASH_ISR	0x00000000	FLASH 中断状态寄存器
0x0C	0x4002100C	FLASH_SR	0x00000000	FLASH 状态寄存器
0x10	0x40021010	FLASH_DR0	0x00000000	FLASH 数据寄存器 0
0x14	0x40021014	FLASH_DR1	0x00000000	FLASH 数据寄存器 1
0x18	0x40021018	FLASH_DR2	0x00000000	FLASH 数据寄存器 2
0x1C	0x4002101C	FLASH_DR3	0x00000000	FLASH 数据寄存器 3
0x20	0x40021020	FLASH_AR	0x00000000	FLASH 编程地址寄存器
0x28	0x40021028	FLASH_ECR	0x00000000	FLASH 擦除控制寄存器
0x50	0x40021050	FLASH_KEYR	0x00000000	FLASH 操作键寄存器
0x60	0x40021060	FLASH_RDPR	0x000000FF	FLASH 读保护寄存器
0x70	0x40021070	FLASH_WRPR	0xFFFFFFFF	FLASH 写保护寄存器
0x80/ 0x84/ 0x88/ 0x8C	0x40021080/ 0x40021084/ 0x40021088/ 0x4002108C	FLASH_UIDx	0xFFFFFFFF	FLASH 用户 IDx 寄存器

5.5.2 寄存器详细描述

5.5.2.1 FLASH 控制寄存器（FLASH_CR）

- 名称：FLASH Control Register
- 偏移地址：0x00
- 默认值：0x80000300
- 返回寄存器列表

位	31	30:18	17	16	15:13	12	11:10	9	8	7:2	1	0
名称	LOCK		EIE	DIE		LAU		DEP	IPE		ES	PS
访问	R/W		R/W	R/W		R/W1C		R/W	R/W		R/WAC	R/WAC
复位值	0x1		0x0	0x0		0x0		0x0	0x0		0x0	0x0

字段	说明
[31] LOCK	**FLASH 锁定标志（FLASH Lock Flag）** 仅写 1 有效，用于对 FLASH 进行加锁。 读取该位： 0：指示当前 FLASH 控制器已解锁，可以进行编程/擦除功能。 1：指示当前 FLASH 控制器已加锁，编程/擦除功能被屏蔽。 注意：解锁后是否能成功进行编程/擦除功能，还将受读/写保护设定限制，解锁详请参考“5.4.2.1FLASH 控制寄存器解锁”章节。
[17] EIE	**FLASH 错误中断使能控制（FLASH Error Interrupt Enable）** 0：Error 中断关闭 1：Error 中断使能
[16] DIE	**FLASH 操作完成中断使能控制（FLASH Operation Done Interrupt Enable）（默认关闭）** 0：Done 中断关闭 1：Done 中断使能
[12] LAU	**FLASH Launch Enable** 0:关闭 1:开启 注意：1. 当解除读写保护后，可以通过对 Launch bit 写 1 来手动更新保护级别，该 bit 写 1 自清。 2. 受读/写保护设定限制，详细请参考“5.4.2.7FLASH 读保护（RDP），5.4.2.8 FLASH 写保护（WRP）”章节
[9] DEP	**FLASH D 总线预取使能（Flash Dbus Prefetch Enable）** 0:关闭 1:开启 预取开关使能，当 CPU 频率工作在 40M 以下，关闭预取功能
[8] IEP	**FLASH I 总线预取使能（Flash Ibus Prefetch Enable）** 0:关闭 1:开启 预取开关使能，当 CPU 频率工作在 40M 以下，关闭预取功能

[1] ES	**FLASH 擦除开始控制位（FLASH Erase Start）** 0：无效 1：执行擦写操作
[0] PS	**FLASH 编程开始控制位（FLASH Program Start）** 0：无效 1：执行编程操作

5.5.2.2 FLASH 低功耗寄存器（FLASH_LPR）

- **名称：FLASH Lowpower Register**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31	30:2	1	0
名称	LOCK		WKUP	STDBY
访问	R/W		R/WAC	R/WAC
复位值	0x0		0x0	0x0

字段	说明
[31] LOCK	**FLASH 唤醒/待机解锁标志位（FLASH Wakeup/Standby Unlock Flag）** 用于解锁低功耗的控制功能，在配置低功耗 Standby/Wakeup 前，需要向 LOCK 位写 1 完成解锁，否则操作无效。
[1] WKUP	**FLASH 唤醒模式启动位（FLASH Wakeup Start）** 1：启动 Wakeup 0：无效 启动 FLASH Wakeup 操作后，会向 FLASH 发起 Wakeup 命令，退出 Standby 状态。
[0] STDBY	**FLASH 待机模式启动位（FLASH Standby Start）** 1：启动 Standby 0：无效 启动 FLASH Standby 操作后，会向 FLASH 发起进入 Standby 的命令。

5.5.2.3 FLASH 中断状态寄存器（FLASH_ISR）

- 名称：FLASH Interrupt Status Register
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:5	4	3	2	1	0
名称		DIF	RPEIF	ECCEIF	WPEIF	OPTEIF
访问		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[4] DIF	**FLASH 操作完成标志（FLASH Operation Done Pending）** 1：Pending 0：No Pending 操作完成标志，详见“5.4.2.6 FLASH 中断”。
[3] RPEIF	**FLASH 读保护错误标志（FLASH Read Protect Error Pending）** 1：读保护下非法执行 FLASH 写/擦操作，该位置 1；若执行读操作则会产生 BusFault 中断； 0：No Pending 读保护错误标志，详见“5.4.2.6 FLASH 中断”。
[2] ECCEIF	**FLASH ECC 标志（FLASH ECC Pending）** 1：ECC 纠错中发现错误（FLASH 中存在 bit 错误） 0：ECC 正常 ECC 错误标志，详见“5.4.2.6 FLASH 中断”。
[1] WPEIF	**FLASH 写保护错误标志（FLASH Write Protect Error Pending）** 1：Pending 0：No Pending 写保护错误标志，详见“5.4.2.6 FLASH 中断”。
[0] OPTEIF	**FLASH 操作错误标志（FLASH Operation Error Pending）** 1：Pending 0：No Pending 操作错误标志，详见“5.4.2.6 FLASH 中断”。

5.5.2.4 FLASH 状态寄存器（FLASH_SR）

- 名称：FLASH Status Register
- 偏移地址：0x0C
- 默认值：0x00000000
- 返回寄存器列表

位	31:1	0
名称		BSY
访问		R
复位值		0x0

字段	说明
[0]	**FLASH 忙碌状态（FLASH Busy Status）**
BSY	FLASH 发起了 Program/Erase/Read 或者其他操作，操作还未完成为 1，当操作完成后会自动清 0
	1：Busy
	0：Idle

5.5.2.5 FLASH 数据寄存器 0（FLASH_DR0）

- 名称：FLASH Data Register0
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DR0
访问	R/W
复位值	0x0

字段	说明
[31:0]	**FLASH 编程数据 0（FLASH Program Data0）**
DR0	FLASH 编程数据 128bit 的第一个 Word
	注意：128Bit 数据如果是全 F，硬件会屏蔽该写操作，产生操作异常（OPTEIF 置位并触发中断）。

5.5.2.6 FLASH 数据寄存器 1（FLASH_DR1）

- 名称：FLASH Data Register1
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DR1
访问	R/W
复位值	0x0

字段	说明
[31:0]	**FLASH 编程数据 1（FLASH Program Data1）**
DR1	FLASH 编程数据 128bit 的第二个 Word 注意：128Bit 数据如果是全 F，硬件会屏蔽该写操作，产生操作异常（OPTEIF 置位并触发中断）。

5.5.2.7 FLASH 数据寄存器 2（FLASH_DR2）

- 名称：FLASH Data Register2
- 偏移地址：0x18
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DR2
访问	R/W
复位值	0x0

字段	说明
[31:0]	**FLASH 编程数据 2（FLASH Program Data2）**
DR2	FLASH 编程数据 128bit 的第三个 Word 注意：128Bit 数据如果是全 F，硬件会屏蔽该写操作，产生操作异常（OPTEIF 置位并触发中断）。

5.5.2.8 FLASH 数据寄存器 3（FLASH_DR3）

- 名称：FLASH Data Register3
- 偏移地址：0x1C
- 默认值：0x00000000

返回寄存器列表

位	31:0
名称	DR3
访问	R/W
复位值	0x0

字段	说明
[31:0]	**FLASH 编程数据 3（FLASH Program Data3）**
DR3	FLASH 编程数据 128bit 的第四个 Word 注意：128Bit 数据如果是全 F，硬件会屏蔽该写操作，产生操作异常（OPTEIF 置位并触发中断）。

5.5.2.9 FLASH 编程地址寄存器（FLASH_AR）

- 名称：FLASH Program Address Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	ADDR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**FLASH 编程地址（FLASH Program Address）**
ADDR	FLASH 编程地址，地址必须 128bit 对齐，否则报错退出（128bit->0x10 对齐）； Main 区域地址:实际 150K(0x0 - 0x25800) (Note：一个 Sector 空间为 480Byte，即一个 Sector 以 0x1e0 对齐，一共 320 个 Sector)

5.5.2.10 FLASH 擦除控制寄存器（FLASH_ECR）

- 名称：FLASH Erase Control Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31	30:9	8:0
名称	EM		EA
访问	R/W		R/W
复位值	0x0		0x0

字段	说明
[31] EM	**FLASH 擦除模式（FLASH Erase Mode）** 1：Chip Erase 0：Sector Erase
[8:0] EA	**FLASH 擦除扇区号选择（FLASH Erase Sector Number select）** Sector 擦除选择，Main 区域一共 320 个 Sector（0-319），每个 Sector 空间为 480Byte，其中 Sector 以 0x1e0 对齐，Main 区域(程序区域)地址范围为 0x0-0x25800；

5.5.2.11 FLASH 键寄存器（FLASH_KEYR）

- 名称：FLASH Key Register
- 偏移地址：0x50
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	KEY
访问	R/W
复位值	0x0

字段	说明
[31:0] KEY	**FLASH 解锁键寄存器（FLASH Unlock Key Reigster）** 用于解锁 FLASH 的擦除/编程控制、读/写保护操作操作。 详请参考“5.4.2.1FLASH 控制寄存器解锁”章节。

5.5.2.12 FLASH 读保护寄存器（FLASH_RDPR）

- 名称：FLASH Read Protect Register
- 偏移地址：0x60
- 默认值：0x000000FF
- 返回寄存器列表

位	31:8	7:0
名称		RDP
访问		R/W
复位值		0xFF

字段	说明
[7:0] RDP	**FLASH 读保护级别（FLASH Read Protect Level）** FLASH 读保护控制，详细请参考“5.4.2.7FLASH 读保护（RDP）”章节 。 Level0：0xAA，此 Level 下无读保护功能，允许调试接口对 FLASH 的读写及擦除操作； Level1：其他（除了 0xAA 和 0xCC），此 Level 下有读保护功能，禁止 RAM Boot 调试接口对 FLASH 的读写及擦除操作； Level2：0xCC，此 Level 下有读保护功能，而且禁用调试接口，并且保护级别不能回退。 Note：设置读保护之前需先解锁，解锁详请参考“5.4.2.1FLASH 控制寄存器解锁”章节。

5.5.2.13 FLASH 写保护寄存器（FLASH_WRPR）

- 名称：FLASH Write Protect Register
- 偏移地址：0x70
- 默认值：0xFFFFFFFF
- 返回寄存器列表

位	31:20	19:0
名称		WRP
访问		R/W
复位值		0xFFFFFFFF

字段	说明
[19:0] WRP	**FLASH 用户写权限控制（FLASH User Write Permission Control）** FLASH 写权限控制，FLASH 一个 Sector 480Byte 空间，每个 bit 对应 FLASH 16 个 Secotr 区域，详情请参考“5.4.2.8 FLASH 写保护（WRP）”章节。 1：存在写权限 0：关闭写权限 Note：设置写保护之前需先解锁，解锁详请参考“5.4.2.1FLASH 控制寄存器解锁”章节

5.5.2.14 FLASH 用户 IDx 寄存器（FLASH_UIDx）

- 名称：FLASH User IDx Register（x = 0 ... 3）
- 偏移地址：0x80/0x84/0x88/0x8C
- 默认值：0xFFFFFFFF
- 返回寄存器列表

位	31:0
名称	UIDx
访问	R
复位值	0xFFFFFFFF

字段	说明
[31:0]	**FLASH 用户 ID 寄存器（FLASH User ID Register）**
UIDx	存储芯片独一无二身份码

6 数据存储 FLASH（DFLASH）

6.1 简介

DFLASH 控制器用于管理 CPU 通过 AHB S-Code Bus 对 DFLASH 的访问。可以通过 DFLASH 控制器对 DFLASH 存储执行擦除与编程操作。

6.2 结构框图

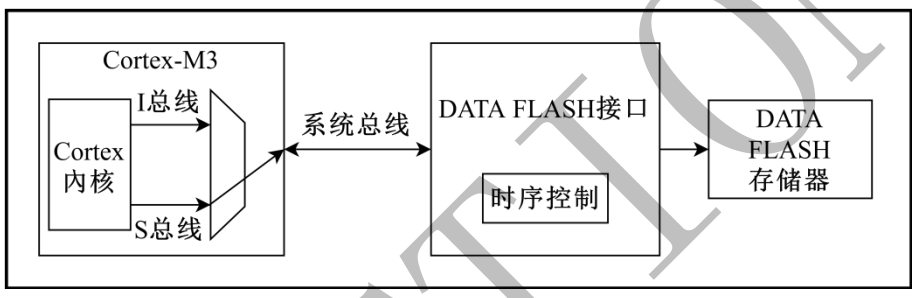


图 6-1 DFLASH 结构框图

6.3 主要特性

DFLASH 主要具有以下特性：

- 18KByte 存储空间
- 包括 1 个主存储区(Main Memory)+ 1 个副存储区(Secondary Memory)
  - 主存储区包含 32 个扇区，每个扇区 512Byte，共计 16 KByte
  - 副存储区包含 4 个扇区，每个扇区 512Byte，共计 2KByte
  - 主存储区支持扇区擦除(Sector Erase)和闪存擦除(或称整片擦除，Chip Erase)
  - 副存储区仅支持扇区擦除(Sector Erase)，不受闪存擦除影响
- 支持寄存器方式的读取/编程/擦除操作
- 32Bit 总线位宽，支持按字节(Byte)或按字(Word)两种操作模式
- 支持加锁保护，避免数据误操作
- 支持低功耗模式

6.4 功能描述

6.4.1 DFLASH 擦除和编程操作

执行任何 DFLASH 编程/擦除操作时，FLASH 工作时钟频率不能低于 1MHz，建议将系统时钟频率配置为 90M。如果在 DFLASH 编程/擦除操作期间发生器件复位，将无法保证 DFLASH 中的内容。

注意：所有操作均为串行执行，需确保上一笔操作完成后再发起新的操作。完成状态可通过查询 DFLASH 状态寄存器(DFLASH_SR)中的 BSY 标志位进行判断，等待操作完成后 BSY 标志将自动清 0。BSY 为 1 时不允许发起新的操作，否则无法保证 DFLASH 的工作状态。

6.4.1.1 DFLASH 控制寄存器解锁

芯片复位完成后，DFLASH 控制寄存器处于加锁状态，DFLASH 控制寄存器(DFLASH_CR)不允许执行写/擦/读操作，以防因电气干扰等原因出现对 DFLASH 的意外操作：

1. 对 DFLASH 密钥寄存器(DFLASH_KEYR)中写入 KEY1 = 0x76543210
2. 对 DFLASH 密钥寄存器(DFLASH_KEYR)中再写入 KEY2 = 0xFEDCBA98

解锁过程必须严格按照上述顺序，解锁成功后，DFLASH_CR 寄存器中的 LOCK 标志位将会自动清 0。如解锁错误，LOCK 标志位不会清 0，解锁无效。

当解锁完成并执行完对应操作后，也可通过软件将 DFLASH_CR 寄存器中的 LOCK 位置 1，重新对 DFLASH 进行加锁。

DFLASH 写操作支持 Byte 及 Word 位宽操作，不允许非对齐操作，操作中必须按照指定 Size 对齐，否则会提示 Error 错误。

6.4.1.2 DFLASH 擦除

擦除操作分为扇区擦除(Sector Erase)和闪存擦除(或称整片擦除，Chip Erase)：

- DFLASH 的主存储区支持扇区擦除和闪存擦除两种擦除方式
- DFLASH 的副存储区仅支持扇区擦除，不受闪存擦除影响

扇区擦除（Sector Erase）

扇区擦除的具体步骤如下：

1. 检查 DFLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 将 DFLASH_ECR 寄存器中的 EMODE 位置 0；
3. 将目标扇区号写入 DFLASH_ECR[4:0]中；
4. 将 DFLASH_CR 寄存器中的 ES 位置 1；
5. 等待 DFLASH_CR 寄存器中 BSY 位清零，擦除动作完成。

注意：扇区编号必须满足 DFLASH 的扇区范围：主存储区为 0 - 31 扇区号，副存储区为 32 - 35 扇区号。否则操作不执行，并且 DFLASH_ISR 寄存器中的 EIF 错误标志位将置 1，并触发中断(如果使能)。

闪存擦除（Chip Erase）

要执行闪存擦除，建议采用以下步骤：

1. 检查 DFLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 将 DFLASH_ECR 寄存器中的 EMODE 位置 1；
3. 将 DFLASH_CR 寄存器中的 ES 位置 1；
4. 等待 DFLASH_CR 寄存器中 BSY 位清零，擦除动作完成。

6.4.1.3 DFLASH 编程/读取

DFLASH 编程的步骤如下：

1. 检查 DFLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 配置 DFLASH_CR 寄存器内的 PDS 位选择需要编程的数据位宽：8/32 bit；
3. 向 DFLASH_DR 寄存器写入相应长度的数据；
4. 向 DFLASH_ADDR 寄存器写入编程起始地址；
5. 将 DFLASH_CR 寄存器内的 PS 位置 1；
6. 等待 DFLASH_CR 寄存器中 BSY 位清零，擦除动作完成。

技巧：把 DFLASH 的单元从逻辑“1”写为逻辑“0”时，可以无需执行擦除操作即可进行写操作，这样可以减少 DFLASH 擦写次数。但 DFLASH 单元从逻辑“0”写为逻辑“1”时，则必须先执行 DFLASH 擦除操作。

DFLASH 读取数据的步骤如下：

1. 检查 DFLASH_SR 寄存器中的 BSY 位是否为 0，否则等待上一笔操作完成；
2. 配置 DFLASH_CR 寄存器内的 PDS 位选择需要读取的数据位宽：8/32 bit；
3. 向 DFLASH_ADDR 寄存器写入读取起始地址，向 DFLASH_DR 寄存器内写入 0(可选操作，用于避免数据残留)；
4. 将 DFLASH_CR 寄存器内的 RS 位置 1；
5. 等待 DFLASH_CR 寄存器中 BSY 位清零，读取动作完成。
6. 从 DFLASH_DR 寄存器读取数据。

注意：

- *DFLASH_ADDR 内写入的编程起始地址，必须按 DFLASH_CR 的 PDS 位所选择编程数据位宽对齐（例如数据位宽选择 32bit 位宽，则编程起始地址需按 32bit 对齐）。否则该笔非对齐操作将不会被执行，并且 DFLASH_ISR 寄存器中的 EIF 错误标志位将置 1，并触发中断(如果使能)。*
- *DFLASH_ADDR 内写入的地址也必须满足有效的地址范围内，不得越界。否则操作不执行，并且 DFLASH_ISR 寄存器中的 EIF 错误标志位将置 1，并触发中断(如果使能)。*
- *DFLASH_SR 寄存器中的 BSY 位自动清 0 前，不允许对 DFLASH 发起新的编程操作，否*

则可能引发未知异常。

- *不得同时设置擦除/编程/读取操作，否则将判定为无效操作，DFLASH 不执行任何实际动作。*

6.4.1.4 DFLASH 中断

DFLASH 有下述几种情况将会引发中断，DFLASH_ISR 寄存器中相应的中断标志位将会置位，通过对相应的位置进行写 1，将清除相应的中断标志位。

表 6-1 引发 DFLASH 中断的情况

中断事件	中断使能	中断标志	备注
操作完成	DIE	DIF	读取完成、编程完成、擦除完成、低功耗操作完成
操作错误	EIE	EIF	编程地址不对齐、扇区擦除编号超过扇区范围、编程地址超过闪存地址范围、编程全 FF 的数据

6.5 寄存器描述

6.5.1 寄存器列表

DFLASH 基地址：**0x4002 2000**

偏移	实例地址	名称	默认值	描述
0x00	0x40022000	DFLASH_CR	0x80000000	DFLASH 控制寄存器
0x04	0x40022004	DFLASH_LPR	0x00000000	DFLASH 低功耗寄存器
0x08	0x40022008	DFLASH_ISR	0x00000000	DFLASH 中断状态寄存器
0x0C	0x4002200C	DFLASH_SR	0x00000000	DFLASH 状态寄存器
0x10	0x40022010	DFLASH_DR	0x00000000	DFLASH 数据寄存器
0x14	0x40022014	DFLASH_AR	0x00000000	DFLASH 地址寄存器
0x18	0x40022018	DFLASH_ECR	0x00000000	DFLASH 擦除控制寄存器
0x30	0x40022030	DFLASH_KEYR	0x00000000	DFLASH 操作键寄存器

6.5.2 寄存器描述

6.5.2.1 DFLASH 控制寄存器（DFLASH_CR）

- 名称：DFLASH Control Register
- 偏移地址：0x00
- 默认值：0x80000000
- 返回寄存器列表

位	31	30:18	17	16	15:5	4	3	2	1	0
名称	LOCK		EIE	DIE		DS		ES	RS	PS
访问	R/W		R/W	R/W		R/W		R/WAC	R/WAC	R/WAC
复位值	0x1		0x0	0x0		0x0		0x0	0x0	0x0

字段	说明
[31] LOCK	**DFLASH 编程/擦除锁定标志位（DFLASH Program/Erase Lock Flag）** 在编程/读取/擦除之前需要解锁。 仅写 1 有效，用于对 DFLASH 进行加锁。读取该位含义： 1：已加锁，编程/读取/擦除功能被屏蔽。 0：已解锁，可以进行编程/读取/擦除功能。 注意：解锁操作参看“6.4.1.1DFLASH 控制寄存器解锁”章节。
[17] EIE	**DFLASH 错误中断使能（DFLASH Error Interrupt Enable）** 1：开启操作 Error 中断使能 0：关闭操作 Error 中断使能
[16] DIE	**DFLASH 操作完成中断使能（DFLASH Operation Done Interrupt Enable）** 1：开启操作 Done 中断使能 0：关闭操作 Done 中断使能
[4] DS	**编程数据位宽选择（Program DataSize）** 1：32Bit 0：8Bit 注意：必须在读取/编程操作启动前配置，中途不允许切换。
[2] ES	**擦除开始控制位（Erase Start）** 1：开始擦写操作 0：无效
[1] RS	**读取开始控制位（Read Start）** 1：开始读取操作 0：无效
[0] PS	**编程开始控制位（Program Start）** 1：开始编程操作 0：无效 写 1 开始编程操作，硬件会自动清 0，写 0 无效。

6.5.2.2 DFLASH 低功耗寄存器（DFLASH_LPR）

- 名称：DFLASH Lowpower Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31	30:2	1	0
名称	LOCK		WS	SS
访问	R/W		R/WAC	R/WAC
复位值	0x0		0x0	0x0

字段	说明
[31] LOCK	**DFLASH 唤醒/待机解锁标志位（DFLASH Wakeup/Standby Unlock Flag）** 解锁方式： 1、先向 31bit 位写 1 完成解锁 2、再启动 Wakeup/Standby 操作 必须按照上述操作流程，否则不允许发起操作，解锁后，自动清除解锁操作；
[1] WS	**启动唤醒模式控制位（Wakeup Start）** 1：启动 Wakeup 0：无效 启动 DFLASH Wakeup 操作后，会向 DFLASH 发起 Wakeup 命令，退出 Standby 状态； 写 1 开始擦写操作，硬件会自动清 0，写 0 无效
[0] SS	**启动待机模式控制位（Standby Start）** 1：启动 Standby 0：无效 启动 DFLASH Standby 操作后，会向 DFLASH 发起进入 Standby 的命令； 写 1 开始擦写操作，硬件会自动清 0，写 0 无效；

6.5.2.3 DFLASH 中断状态寄存器（DFLASH_ISR）

- **名称：DFLASH Interrupt Status Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:2	1	0
名称		DIF	EIF
访问		R/W1C	R/W1C
复位值		0x0	0x0

字段	说明
[1] DIF	**DFLASH 完成中断标志位（DFLASH Done Interrupt Flag）** 1：Pending 0：No Pending 操作完成中断标志位，详请参考“6.4.1.4DFLASH 中断”章节
[0] EIF	**操作错误中断标志位（DFLASH Error Interrupt Flag）** 1：Pending 0：No Pending DFLASH 操作错误标志，包括地址不对齐/地址或扇区越界/写全 FF 数据。详请参考“6.4.1.4DFLASH 中断”章节

6.5.2.4 DFLASH 状态寄存器（DFLASH_SR）

- **名称：DFLASH Status Register**
- **偏移地址：0x0C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:1	0
名称		BSY
访问		R
复位值		0x0

字段	说明
[0] BSY	**DFLASH 繁忙状态位（DFLASH Busy Status）** DFLASH 发起了 Program/Erase/Read 或者其他操作，操作还未完成，当操作完成后会自动清 0 1：操作忙 0：空闲

6.5.2.5 DFLASH 数据寄存器（DFLASH_DR）

- 名称：DFLASH Data Register
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DATA
访问	R/W
复位值	0x0

字段	说明
[31:0]	**DFLASH 编程/读取数据寄存器（DFLASH Program/Read Data Register）**
DATA	DFLASH 编程 32bit 数据，如果 Size 为 8Bit，则取前 8Bit 数据； 需要注意：32Bit 数据如果是全 F，硬件会屏蔽该写操作。

6.5.2.6 DFLASH 地址寄存器（DFLASH_AR）

- 名称：DFLASH Address Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	ADDR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**DFLASH 编程/读取地址（DFLASH Program/Read Address）：Byte 地址**
ADDR	DFLASH 编程地址，地址必须根据配置的 Size 对齐； 详请参看“6.4.1.3DFLASH 编程/读取”章节。

6.5.2.7 DFLASH 擦除控制寄存器（DFLASH_ECR）

- 名称：DFLASH Erase Control Register
- 偏移地址：0x18
- 默认值：0x00000000
- 返回寄存器列表

位	31	30:6	5:0
名称	EM		SEA
访问	R/W		R/W
复位值	0x0		0x0

字段	说明
[31] EM	**DFLASH 擦除模式选择（DFLASH Erase Mode）** 1：闪存擦除 0：扇区擦除
[5:0] SEA	**擦除目标扇区号选择（DFLASH Erase Sector Number select）** 仅当 EMODE=0 时有效。 主存储区扇区号范围（共 32 个扇区）：0 to 31 副存储区扇区号范围（共 4 个扇区）：32 to 35

6.5.2.8 DFLASH 操作键寄存器（DFLASH_KEYR）

- 名称：DFLASH Option Key Register
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	KEY
访问	R/W
复位值	0x0

字段	说明
[31:0] KEY	**DFLASH Unlock Key Register** 用于解锁 FLASH 的擦除/编程控制、读/写保护操作操作。 详请参考“6.4.1.1DFLASH 控制寄存器解锁”章节

7 电源控制器（PWR）

7.1 电源

芯片的工作电压(VDDA/VDDIO) 要求介于 3.1 V 到 3.6V 之间，通过内部 LDO（线性调压器）提供内部 1.5V 数字电源。

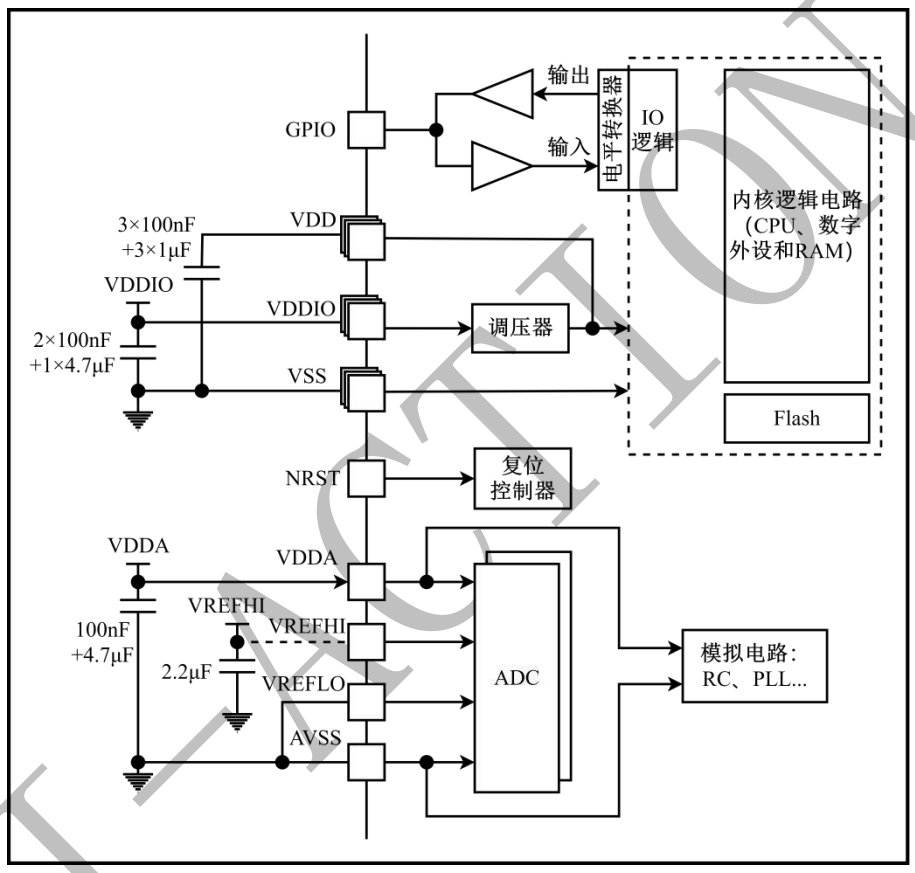


图 7-1 电源框架图

7.1.1 LDO（线性调压器）

嵌入式线性调压器为所有数字电路供电，调压器输出电压约为 1.5V。

此调压器需要将两个外部电容连接到专用引脚 VDD 和 VSS 之间，所有封装都配有 VDD 引脚，具体引脚与封装有关。

调压器在复位后始终处于使能状态，该模块由 VDDIO 供电，每一个 VDD 引脚上需要连接一个 1uF 和一个 100nF 电容来稳定电压，也可以使用更大的电容，但应该注意更大的电容会引起更长的上电时间。这些电容应该被放置在尽可能靠近 VDD 引脚的位置。不建议使用 VDD 驱动外部负载。

- 运行模式，调压器为 1.5 V 域（内核、存储器和数字外设）提供全功率，在此模式下，可将调压器的输出电压通过软件配置调整为 1.5V 的电压值（PMUCR 寄存器的位[7:4]）。
- 待机模式，调压器为 1.5 V 域（内核、存储器和数字外设）提供低功率，可以保留寄存器和内部 SRAM 中的内容。在此模式下，将调压器的输出电压通过软件配置调整为 1.1V（PMUCR 寄存器的位[7:4]）。

注意：当 VDD 电压配置为 1.1V 时，需注意 LVD 中阈值电压的选择与配置(避免高于 1.1V)，防止误入复位状态。

7.2 电源监控器

7.2.1 上电复位（POR）/掉电复位（PDR）

本芯片内部集成有 POR/PDR 电路，可以从 2.8V 开始正常工作。

当 VDDIO/VDDA 低于指定阈值 $V_{POR/PDR}$ 时，器件无需外部复位电路便会保持复位状态。

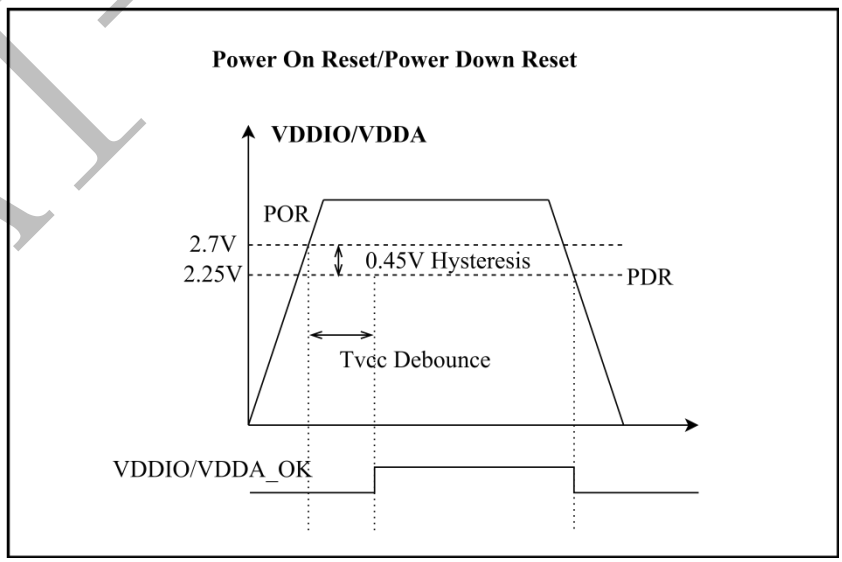


图 7-2 上电复位/掉电复位波形

7.2.2 低电压检测器（LVD）

可以使用 LVD 单元监视 VDDIO/VDD 的电源电压，对 VDDIO/VDD 低电压和过流监测，将其与芯片的 LVD 低电压检测模块中的电压阈值控制寄存器(LACR)所选的阈值进行比较。

VDDIO/VDD 低电压及过流监测功能分别通过设置 LVD 模块中的电压阈值控制寄存器(LVD_LACR)的 Enable 来使能 LVD 对应的功能。

LVD 单元中的电源控制/状态寄存器(LCR)中提供了 VDDIO/VDD 欠压或过流标志，用于指示电源电压是否小于 LVD 阈值或者电源电流是否大于 LVD 阈值。该事件内部连接到 NMI 中断，如果中断使能，则可以产生不可屏蔽中断；该事件内部连接到系统复位，如果复位使能，则可以产生系统复位；该事件内部连接到事件系统，该功能的用处是可以在异常情况下执行紧急关闭的任务。

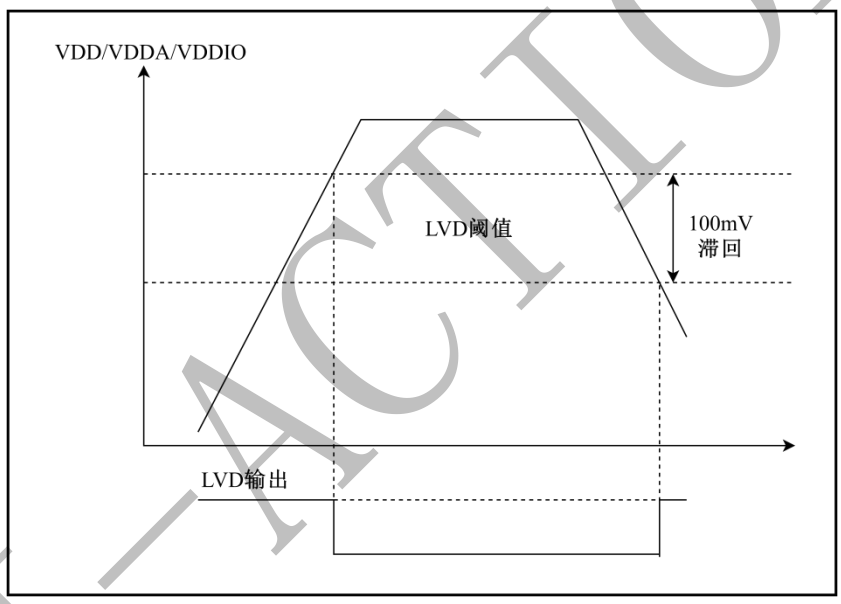


图 7-3 LVD 阈值

7.3 低功耗模式

默认情况下，系统复位或上电复位后，微控制器进入运行模式。在运行模式下，CPU 通过 HCLK 提供时钟，并执行程序代码。系统提供了多个低功耗模式，可在 CPU 不需要运行时（例如等待外部事件时）节省功耗。由用户根据应用选择具体的低功耗模式，以在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。

芯片有两个低功耗模式：

- 睡眠模式（Cortex™-M3 内核停止，外设保持运行）
- 停止模式（除 LSI 时钟外，所有时钟都停止）

此外，可通过下列方法之一降低运行模式的功耗：

- 降低系统时钟速度
- 不使用 APB 和 AHB 外设时，将对应的外设时钟关闭

表 7-1 低功耗模式汇总

模式名称	进入	唤醒	对 VDD 域时钟的影响	调压器
睡眠（立即休眠或退出时休眠）	WFI	任意中断	CPU CLK 关闭对其它时钟或模拟时钟源无影响	开启
	WFE	唤醒事件		
停止	SLEEPDEEP 位 + WFI 或 WFE	任意中断	除 LSI 时钟外，所有时钟都停止	开启

7.3.1 降低系统时钟速度

在运行模式下，可通过对预分频寄存器编程来降低系统时钟（FCLK、HCLK、PCLK0 和 PCLK1）速度。
在进入睡眠模式之前，也可以使用这些预分频器来降低外设运行速度。
有关详细信息，请参见第 8.3.2.5 节：RCU 系统时钟控制寄存器（RCU_SCLKCR）。

7.3.2 外设时钟门控

在运行模式下，可随时停止各外设和存储器的 HCLK 和 PCLK 以降低功耗。要进一步降低睡眠模式的功耗，可在执行 WFI 或 WFE 指令之前禁止外设时钟。

AHB0 外设时钟门控由 AHB0 外设时钟使能寄存器(RCU_AHB0CCR)。参见第 8.3.2.12 节：RCU AHB0 时钟控制寄存器（RCU_AHB0CCR）。

AHB1 外设时钟门控由 AHB1 外设时钟使能寄存器(RCU_AHB1CCR)。参见第 8.3.2.13 节：RCU AHB1 时钟控制寄存器（RCU_AHB1CCR）。

APB0 外设时钟门控由 APB0 外设时钟使能寄存器(RCU_APB0CCR)。参见第 8.3.2.10 节：RCU APB0 时钟控制寄存器（RCU_APB0CCR）。

APB1 外设时钟门控由 APB1 外设时钟使能寄存器(RCU_APB1CCR)。参见第 8.3.2.11 节：RCU APB1 时钟控制寄存器（RCU_APB1CCR）。

7.3.3 睡眠模式

进入睡眠模式

执行 WFI（等待中断）或 WFE（等待事件）指令即可进入睡眠模式。根据 Cortex™-M3 系统控制寄存器中 SLEEPONEXIT 位的设置，可以通过两种方案选择睡眠模式进入机制：

- 立即休眠：如果 SLEEPONEXIT 位清零，MCU 将在执行 WFI 或 WFE 指令时立即进入睡眠模式。
- 退出时休眠：如果 SLEEPONEXIT 位置 1，MCU 将在退出优先级最低的 ISR 时立即进入睡眠模式。

有关如何进入睡眠模式的详细信息，请参见表 7-2 和表 7-3。

退出睡眠模式

如果使用 WFI 指令进入睡眠模式，则嵌套向量中断控制器(NVIC)确认的任意外设中断都会将器件从睡眠模式唤醒。

如果使用 WFE 指令进入睡眠模式，CPU 将在有事件发生时立即退出睡眠模式。唤醒事件可通过以下方式产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能 Cortex™-M3 系统控制寄存器中的 SEVONPEND 位。当 CPU 从 WFE 恢复时，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）。
- 配置一个外部或内部中断为事件模式。当 CPU 从 WFE 恢复时，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

由于没有在进入/退出中断时浪费时间，此模式下的唤醒时间最短。

有关如何退出睡眠模式的详细信息，请参见表 7-2 和表 7-3。

表 7-2　进入和退出立即休眠

立即休眠模式	说明
进入模式	WFI（等待中断）或 WFE（等待事件），且： ➢ SLEEPDEEP = 0 及 ➢ SLEEPONEXIT = 0 请参见 Cortex™-M3 系统控制寄存器。
退出模式	如果使用 WFI 进入： 中断：请参见表 16-1（芯片的中断向量表） 如果使用 WFE 进入： 唤醒事件：请参见第 16.1.4 节：唤醒事件

表 7-3　进入和退出退出时休眠

退出时休眠	说明
进入模式	WFI（等待中断），且： ➢ SLEEPDEEP = 0 及 ➢ SLEEPONEXIT = 1 请参见 Cortex™-M3 系统控制寄存器。
退出模式	中断：请参见表 16-1（芯片的中断向量表）

7.3.4　停止模式

停止模式基于 Cortex™-M3 深度睡眠模式与外设时钟门控。在停止模式下，1.5V 域中除了 32K 外的所有时钟都会停止，PLL、HSI 和 HSE RC 振荡器也被禁止。内部 SRAM 和寄存器内容将保留。

进入停止模式

有关如何进入停止模式的详细信息，请参见表 7-4。

要进一步降低停止模式的功耗，可将内部调压器设置为低功耗模式。通过芯片电源控制寄存器(PMUCR)的 VDDS 位进行设置，将电压设置为 1.1V。

在停止模式下，ADC 或 DAC 也会产生功耗，在进入停止模式前将其禁止。

退出停止模式

有关如何退出停止模式的详细信息，请参见表 7-4。

通过发出中断或唤醒事件退出停止模式时，将选择 HSI RC 振荡器作为系统时钟。

在停止模式下，调压器处于低功耗模式下工作，当从停止模式唤醒，可将 LDO 输出电压设置为 1.5V。

表7-4 进入和退出停止模式

停止模式	说明
进入模式	WFI（等待中断）或WFE（等待事件），且： ➢ 将电源控制寄存器(PMUCR)中的VDDS位调整为1.1V； ➢ 将Cortex™-M3系统控制寄存器中的SLEEPDEEP位置1； 注意：要进入停止模式之前，所有中断需处于挂起状态，否则将忽略进入停止模式这一过程，继续执行程序。
退出模式	如果使用 WFI 进入： 所有配置为中断模式的中断，必须在NVIC中使能对应的中断向量，请参见表16-1：芯片的中断向量表。 如果使用 WFE 进入： 所有配置为事件模式的中断，请参见第16.1.4节：唤醒事件。

调试模式

默认情况下，如果使用调试功能时应用程序将 CPU 置于停止模式，调试连接将中断。这是因为 Cortex™-M3 内核时钟停止工作。

7.4 PWR 电源控制寄存器

参考 SYSCTRL_PMUCR 寄存器描述。

7.5 LVD 低电压监测寄存器

7.5.1 寄存器列表

LVD 基地址：**0x4000 A000**

偏移	实例地址	名称	默认值	描述
0x00	0x4000A000	LVD_ACR	0x00001A77	PWR 模拟部分控制寄存器
0x04	0x4000A004	LVD_CR	0x00000000	PWR 数字部分控制寄存器
0x08	0x4000A008	LVD_SR	0x00000000	PWR 异常状态寄存器
0x0C	0x4000A00C	LVD_DBCR	0x00000700	PWR 异常消抖寄存器

7.5.2 寄存器详细描述

7.5.2.1 LVD 模拟部分控制寄存器（LVD_ACR）

- 名称：**LVD Analog Contorl Register**
- 偏移地址：**0x00**
- 默认值：**0x00001A77**
- 返回寄存器列表

位	31:14	13:12	11:10	9:8	7
名称		VDDOCS	VDDALVS	VDDIOLVS	
访问		R/W	R/W	R/W	
复位值		0x1	0x2	0x2	
位	6:4	3	2	1	0
名称	VDDLVS	VDDOCE	VDDALVE	VDDIOLVE	VDDLVE
访问	R/W	R/W	R/W	R/W	R/W
复位值	0x7	0x0	0x1	0x1	0x1

字段	说明
[13:12] VDDOCS	**VDD 过流检测档位设置（VDD Overcurrent Detection Gear Setting）** 00：300mA　01：350mA 10：400mA　11：450mA
[11:10] VDDALVS	**VDDA 低电压阈值设置（VDDA Low Voltage Threshold Setting）** 00：2.4　01：2.55 10：2.7　11：3
[9:8] VDDIOLVS	**VDDIO 低电压阈值设置（VDDIO Low Voltage Threshold Setting）** 00：2.4　01：2.55 10：2.7　11：3
[6:4] VDDLVS	**VDD 低电压阈值设置（VDD Low Voltage Threshold Setting）** 000：0.8　001：0.9　010：1.0　011：1.1　100：1.2　101：1.3 110：1.35 (FLASH 低于 1.35V 工作不正常，低电压阈值设为 1.35V)　111：1.4
[3] VDDOCE	**VDD 过流异常功能使能（VDD Overcurrent Abnormal Function Enable）** 0：不使能 1：使能
[2] VDDALVE	**VDDA 低电压异常功能使能（VDDA Low Voltage Abnormal Function Enable）** 0：不使能 1：使能
[1] VDDIOLVE	**VDDIO 低电压异常功能使能（VDDIO Low Voltage Abnormal Function Enable）** 0：不使能 1：使能

[0] VDDLVE	**VDD 低电压异常功能使能（VDD Low Voltage Abnormal Function Enable）** 0：不使能 1：使能

7.5.2.2 LVD 数字部分控制寄存器（LVD_CR）

- **名称：LVD Contorl Register**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15	14	13	12	11	10
名称		VDDOCBE	VDDALVBE	VDDIOLVBE	VDDLVBE	VDDOCRE	VDDALVRE
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0
位	9	8	7:4	3	2	1	0
名称	VDDIOLVRE	VDDLVRE		VDDOCIE	VDDALVIE	VDDIOLVIE	VDDLVIE
访问	R/W	R/W		R/W	R/W	R/W	R/W
复位值	0x0	0x0		0x0	0x0	0x0	0x0

字段	说明
[15] VDDOCBE	**VDD 过流异常刹车使能（VDD Overcurrent Abnormal Brake Enable）** 0：不使能 1：使能
[14] VDDALVBE	**VDDA 低电压异常刹车使能（VDDA Low Voltage Abnormal Brake Enable）** 0：不使能 1：使能
[13] VDDIOLVBE	**VDDIO 低电压异常刹车使能（VDDIO Low Voltage Abnormal Brake Enable）** 0：不使能 1：使能
[12] VDDLVBE	**VDD 低电压异常刹车使能（VDD Low Voltage Abnormal Brake Enable）** 0：不使能 1：使能
[11] VDDOCRE	**VDD 过流异常复位使能（VDD Overcurrent Abnormal Reset Enable）** 0：不使能 1：使能
[10] VDDALVRE	**VDDA 低电压异常复位使能（VDDA Low Voltage Abnormal Reset Enable）** 0：不使能 1：使能
[9] VDDIOLVRE	**VDDIO 低电压异常复位使能（VDDIO Low Voltage Abnormal Reset Enable）** 0：不使能 1：使能

[8] VDDLVRE	**VDD 低电压异常复位使能（VDD Low Voltage Abnormal Reset Enable）** 0：不使能 1：使能
[3] VDDOCIE	**VDD 过流异常中断使能（VDD Overcurrent Abnormal Interrupt Enable）** 0：不使能 1：使能
[2] VDDALVIE	**VDDA 低电压异常中断使能（VDDA Low Voltage Abnormal Interrupt Enable）** 0：不使能 1：使能
[1] VDDIOLVIE	**VDDIO 低电压异常中断使能（VDDIO Low Voltage Abnormal Interrupt Enable）** 0：不使能 1：使能
[0] VDDLVIE	**VDD 低电压异常中断使能（VDD Low Voltage Abnormal Interrupt Enable）** 0：不使能 1：使能

7.5.2.3 LVD 异常状态寄存器（LVD_SR）

- **名称：LVD Status Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:4	3	2	1	0
名称		VDDOCST	VDDALVST	VDDIOLVST	VDDLVST
访问		R	R	R	R
复位值		0x0	0x0	0x0	0x0

字段	说明
[3] VDDOCST	**VDD 过流异常状态位（VDD Overcurrent Abnormal Status）** 0：正常 1：低电压 Note：该 Bit 位只读；
[2] VDDALVST	**VDDA 低电压异常状态位（VDDA Low Voltage Abnormal Status）** 0：正常 1：低电压 Note：该 Bit 位只读；
[1] VDDIOLVST	**VDDIO 低电压异常状态位（VDDIO Low Voltage Abnormal Status）** 0：正常 1：低电压 Note：该 Bit 位只读；

[0] VDDLVST	**VDD 低电压异常状态位（VDD Low Voltage Abnormal Status）** 0：正常 1：低电压 Note：该 Bit 位只读；

7.5.2.4 LVD 异常消抖寄存器（LVD_DBCR）

- **名称：LVD Debounce Register**
- **偏移地址：0x0C**
- **默认值：0x00000700**
- 返回寄存器列表

位	31:16	15:8	7:4	3	2	1	0
名称		DW		VDDOCDBE	VDDALVDBE	VDDIOLVDBE	VDDLVDBE
访问		R/W		R/W	R/W	R/W	R/W
复位值		0x7		0x0	0x0	0x0	0x0

字段	说明
[15:8] DW	**模拟输入信号消抖窗口宽度设置（Debounce Window）** 0:1 1:2 … N:N+1
[3] VDDOCDBE	**VDD 过流消抖 Bypass 使能（VDD Overcurrent Debounce Bypass Enable）** 0:不使能 1:使能
[2] VDDALVDBE	**VDDA 低电压消抖 Bypass 使能（VDDA Low Voltage Debounce Bypass Enable）** 0:不使能 1:使能
[1] VDDIOLVDBE	**VDDIO 低电压消抖 Bypass 使能（VDDIO Low Voltage Debounce Bypass Enable）** 0:不使能 1:使能
[0] VDDLVDBE	**VDD 低电压消抖 Bypass 使能（VDD Low Voltage Debounce Bypass Enable）** 0:不使能 1:使能

8 复位和时钟控制（RCU）

8.1 复位

芯片共有二种类型的复位，分别为系统复位、电源复位。

8.1.1 系统复位

除了 SYSCTRL 模块中复位标志位、FLASH 自检参数外，系统复位将复位所有寄存器至复位状态，当发生以下任一事件时，将产生一个系统复位：

- NRST 引脚上的低电平(外部复位)
- 独立看门狗计数完成(IWDG 复位)
- 窗口看门狗计数完成(WWDG 复位)
- 软件复位

可通过查看 SYSCTRL 内部 SRSTSR 状态寄存器中的复位状态标志位识别复位事件来源。若要对芯片进行软件复位，必须将 Cortex™-M3 应用中断和复位控制寄存器中的 SYSRESETREQ 位置 1。有关详细信息，请参见 Cortex™-M3 技术参考手册。

8.1.1.1 引脚复位

芯片具有一个独立的复位引脚 NRST，默认上拉，拉低该引脚将引起系统复位。

8.1.1.2 独立看门狗复位

详见独立看门狗模块介绍。

8.1.1.3 窗口看门狗复位

详见窗口看门狗模块介绍。

8.1.1.4 软件复位

通过将 Cortex™-M3 内核的“应用中断与复位控制寄存器”中的 SYSRESETREQ 位置 1，可实现软件复位（内核和外设）；该寄存器中的另一个控制位 VECTRESET 则只复位 Cortex™-M3 内核，不复位外设。

8.1.2 电源复位

以下事件之一发生时，将产生电源复位：

- 上电复位（POR 复位）
- 低电压复位（LVD 复位）

电源复位将复位所有寄存器。

8.1.2.1 上电复位

芯片内部有一个完整的上电复位（POR）电路，当供电电压达到 V_{POR} 时系统即能正常工作。当 VDD 低于指定的限位电压 V_{POR} 时，系统保持为复位状态。

8.1.2.2 低电压复位

低电压复位是一种强制性保护复位，用户可以通过配置使能位开启，当供电电压低于检测阈值时，系统将产生复位。

8.2 时钟

8.2.1 架构图

芯片内部时钟包括 32KHz 低频 RC 振荡器(LSI)，8MHz 高频 RC 振荡器(HSI)及两路 PLL，分别为系统及外设提供时钟，具体架构图如下：

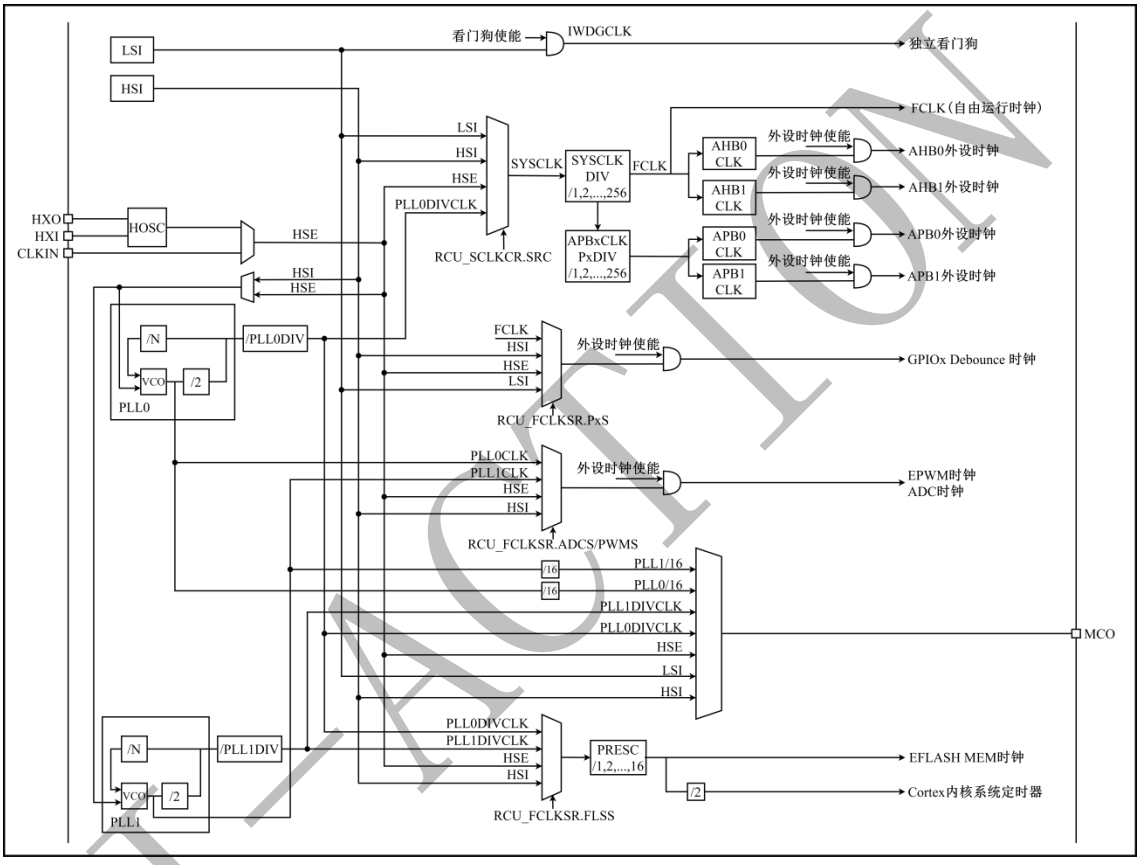


图 8-1　时钟电路简图

8.2.2 系统时钟（SYSCLK）选择

在系统复位后，默认系统时钟为 HSI，若需要切换时钟，需要在目标时钟源已准备就绪（时钟已启动或 PLL 已锁相），才可从一个时钟源切换到另一个。

其中系统时钟（SYSCLK）可使用以下四种不同的时钟源来驱动：

- HSI 振荡器时钟
- LSI 振荡器时钟
- PLL0DIVCLK 时钟
- HSE 振荡器时钟

对于每个时钟源来说，在未使用时都可单独打开或者关闭（LSI 除外），以降低功耗。

时钟控制器（RCU）为应用带来了高度的灵活性，用户在运行内核和外设时可选择使用外部晶振或者使用内部振荡器及 PLL 时钟，既可灵活地配置系统工作的时钟频率，也可为 EPWM、ADC 等需要特定外设保证合适的时钟频率。

通过多个预分频器来配置 AHB0/1、低速 APB(APB0)和高速 APB(APB1)的时钟频率。其中 AHB 域最高时钟频率为 90MHz，APB0 域最高时钟频率为 20MHz，APB1 域最高时钟频率为 40MHz。

除了来自于 PLL1 或 PLL0 输出的 ADC 及 EPWM 工作时钟外，所有其他外设时钟均由系统时钟直接或者分频后来提供。

时钟控制器（RCU）送出一个 5MHz 的 SysTick 时钟到 Cortex 内核的系统定时器(SysTick)，SysTick 可使用此时钟作为时钟源，也可使用 HCLK 作为时钟源，具体可在 SysTick 控制和状态寄存器中配置。

内部 FCLK 充当 Cortex™-M3 的自由运行时钟，有关详细信息，请参见 Cortex™-M3 技术参考手册。

如将 HSE 或 PLL0（PLL0 参考源选择为由 HSE 提供）用作系统时钟输入，则在 HSE 发生故障时，两套 PLL 也将随即停止工作。为了避免因 HSE 故障带来的影响，用户可自由配置选择时钟安全切换方案：第一种是故障发生时将系统时钟直接切换至内部 HSI；第二种是故障发生时将 PLL 时钟参考源切换至 HSI，详细参考 8.2.7 时钟安全系统（CSS）。

8.2.3 LSI 振荡器

LSI 时钟信号由内部 32KHz RC 振荡器生成，可直接用作系统时钟。

LSI 可作为低功耗的时钟源，在停机和待机模式下保持运行。

供独立看门狗(IWDG)使用，时钟频率在 32 kHz 左右，LSI 低频振荡器不可关闭。

8.2.4 HSI 振荡器

HSI 时钟信号由内部 8 MHz RC 振荡器生成，可直接用作系统时钟，或者用作 PLL 参考时钟输入。

HSI RC 振荡器的优点是成本较低（无需使用外部组件），此外，其启动速度也要比 HSE 晶振快。但是 RC 振荡器即使校准后，其精度也不及外部晶振或陶瓷谐振器。

HSI 振荡器频率不做校准情况下，振荡频率为 8MHz 左右，具体频率值保存在 FLASH 中。当 HSI 用做 PLL 的参考时钟源时，可读取该实际频率来计算相应分频比，配置 PLL 以得到准确的 PLL 输出时钟频率。

HSI RC 振荡器可通过 RCU 中晶振控制寄存器 RCU_XOSCCR.HIE 位来打开或关闭。

HSI 信号还可作为备份时钟源（辅助时钟）使用，以防 HSE 晶振发生故障。请参见第 8.2.7 节：时钟安全系统(CSS)。

8.2.5 HSE 晶振

外部晶振谐振器和负载电容必须尽可能地靠近振荡器的引脚，以尽量减小输出失真和起振稳定时间。负载电容值必须根据所选振荡器的不同做适当调整。

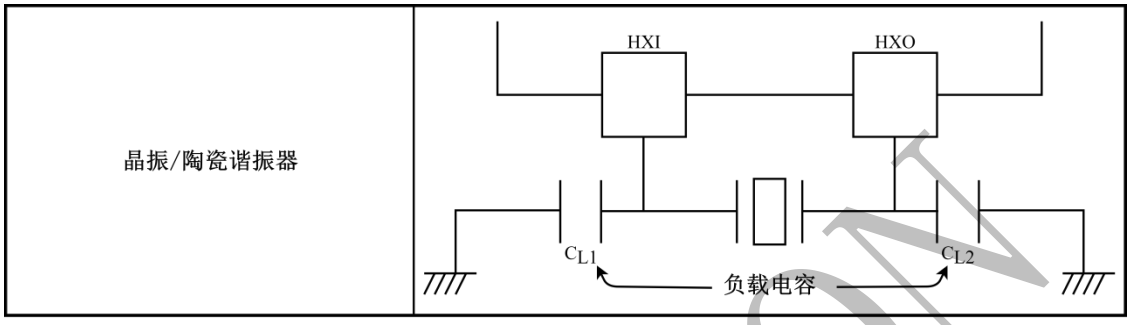


图 8-2　HSE 时钟源

外部晶振/陶瓷谐振器

HSE 的特点是精度非常高。

相关的硬件配置如图 8-2 所示。

在芯片上电完成后，晶振起振电路使能默认为 1，起振电路可以直接使用。

晶振起振电路可通过 RCU 中晶振控制寄存器 RCU_XOSCCR.XEN 位来打开或关闭。

8.2.6 PLL 配置

芯片内内置两套 PLL，两套 PLL 均可以使用 HSE 或 HSI 作为参考时钟：

- 主 PLL（PLL0）用于生成系统时钟 90MHz，主要用于以下三种不同的输出时钟：
  - 第一个输出用于生成高速系统时钟（最高达 90MHz）
  - 第二个输出用于生成 FLASH 工作时钟（固定为 10MHz）
  - 第三个输出直接用于 ADC/EPWM 模块工作时钟；
- 专用 PLL（PLL1）用于生成高速时钟 160M，直接用于 ADC 及 EPWM 模块工作时钟；

如果动态修改 PLL 参数会导致 PLL 需重新锁定，避免 PLL 锁定过程中的频率过冲带来不确定影响，建议在 PLL 使能后不可更改 PLL 的配置参数，所以先对 PLL 进行配置，然后再使能。如需调整 PLL 的配置参数，需先切换时钟源，然后关闭 PLL 使能，调整参数并使能。

PLL 配置流程

首先确定 PLL 参考时钟的选择，其中可以使用 HSI 或 HSE 时钟作为 PLL 的参考时钟输入。

- 参考源配置
  通过配置 RCU 模块中 PLLx 控制寄存器 RCU_PLLxCR.SRC 位来选择 PLLx 的参考时钟。
  - 00：HSE
  - 01：HSI

- 10：Reserved
- 11：Reserved

- PLLxCLK 频率

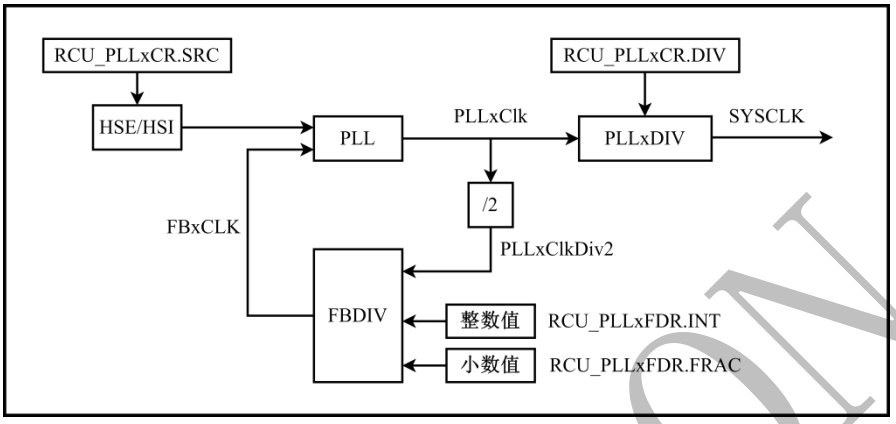


图 8-3　PLL 的配置流程

参考上图 8-3，SYSCLK 系统时钟频率*（乘以）PLLx 分频系数(PLLxDIV)可以得到 PLLx 的输出时钟频率（PLLxClk）。

- PLL 的 Feedback 分频：包含小数部分和整数部分

1. PLLx 的时钟频率（PLLxClk）经过硬件 2 分频后送到 Feedback 分频模块(FBDIV)。
2. PLLx 反馈时钟频率要求 FBDIV 的输出时钟频率(FBxCLK)应等于 PLLx 的输入参考源频率（参考第一步配置）。将 PLLxClkDiv2 时钟频率/(除以)输入参考源频率即可得到 FBDIV 的分频值。根据上述公式的计算结果，将整数部分存入 PLLx 分频寄存器 RCU_PLL0FDR.INT 位内，将小数部分*（乘以）2^16 再取整后存入 PLLx 分频寄存器 RCU_PLL0FDR.FRAC 位内。

Note：

1. HSE 晶振频率范围支持 6~26Mhz，推荐晶振频率为 8Mhz；
2. PLLx 的输出时钟频率 PLLxClk 也等于输入参考源频率*FBDIV*2；
3. 系统时钟频率 SYSCLK 等于 PLLxClk / PLLxCLK_DIV；

8.2.7　时钟安全系统（CSS）

为了增强系统的可靠性，防止因时钟失效造成系统出错甚至死机的严重后果， 芯片内部增加了一个时钟安全监控（CSS）模块。

时钟安全监控（CSS）模块用于监测 HSE 的振荡情况，包含晶振、陶振、或外部输入时钟的运行情况，一旦发生停振或振荡异常（频率低于正常值或高于正常值），则发送异常标志，该异常标志可作为中断输入或 PWM 异常事件信号输入。

用户可通过配置 RCU 中 CSS 控制寄存器 RCU_CSSCR 来使能 CSS 模块，当发生异常时，用户可以选择系统时钟或 PLL 参考时钟是否自动切换内部 HSI。

注意：当 CSS 功能使能后，外部 HSE 振荡器出现异常，CSS 监测模块产生异常标志，该标志信号会触发 NMI 中断，NMI 中断不可屏蔽并且直到 CSS 中断标志位被清除。

如果直接或间接使用 HSE 振荡器作为系统时钟（间接是指该振荡器用作 PLL 的参考时钟源，并且该 PLL 时钟用作为系统时钟），当 HSE 振荡器出现故障，用户可选择将系统时钟切换到 HSI 振荡器。

如果 HSE 振荡器用作为 PLL0 的参考时钟源，则在 HSE 振荡器发生故障时，PLL 也会被禁止，用户可选择将 PLL 的参考时钟源切换到 HSI 振荡器，由于内部 HSI 振荡器时钟会导致 PLL 时钟输出出现稍微偏差。

8.2.8 看门狗时钟

如果独立看门狗(IWDG)通过软件设置的方式启动，则 LSI 振荡器将为独立看门狗(IWDG)提供运行计数时钟，并且 LSI 振荡器不可关闭。

8.2.9 时钟输出功能

芯片时钟输出(MCO)引脚：
用户可配置向 MCO 引脚(PB15)输出以下不同的时钟源：

- LSI
- HSI
- HSE
- FCLK
- PLL0DivClk
- PLL0/16
- PLL1DivClk
- PLL1/16

所需的时钟源选择可通过 RCU 模块中 RCU_SDBGCR.SRC 位来选择。

对于 MCO 引脚，必须将相应的 GPIO 端口的复用功能配置为 AF2 模式(复用为 MCO 功能)。

建议 MCO 输出时钟频率不要超过 80 MHz，具体速率与 I/O 负载电容、I/O 驱动能力以及 Slew Rate 配置相关。

8.3 寄存器描述

8.3.1 寄存器列表

RCU 基地址：0x4002 E000

地址	偏移	寄存器名称	默认值	寄存器描述
0x4002E000	0x00	RCU_PLL0CR	0x00001400	RCU PLL0 控制寄存器
0x4002E004	0x04	RCU_PLL0FDR	0x000A0000	RCU PLL0 分频寄存器
0x4002E008	0x08	RCU_PLL1CR	0x00001400	RCU PLL1 控制寄存器
0x4002E00C	0x0C	RCU_PLL1FDR	0x000A0000	RCU PLL1 分频寄存器
0x4002E020	0x20	RCU_SCLKCR	0x00000000	RCU 系统时钟控制寄存器
0x4002E024	0x24	RCU_BCLKCR	0x000F0103	RCU 总线时钟控制寄存器
0x4002E028	0x28	RCU_FCLKSR	0x00000000	RCU 功能时钟源寄存器
0x4002E030	0x30	RCU_FCLKDR0	0x00000000	RCU 功能时钟分频寄存器 0
0x4002E034	0x34	RCU_FCLKDR1	0x00000000	RCU 功能时钟分频寄存器 1
0x4002E038	0x38	RCU_APB0CCR	0x00000000	RCU APB0 时钟控制寄存器
0x4002E03C	0x3C	RCU_APB1CCR	0x00000000	RCU APB1 时钟控制寄存器
0x4002E040	0x40	RCU_AHB0CCR	0x00000002	RCU AHB0 时钟控制寄存器
0x4002E044	0x44	RCU_AHB1CCR	0x00000000	RCU AHB1 时钟控制寄存器
0x4002E048	0x48	RCU_FUNCCR	0x00000001	RCU 功能时钟控制寄存器
0x4002E050	0x50	RCU_SYSRCR	0x00000F0F	RCU 系统复位控制寄存器
0x4002E054	0x54	RCU_APB0RCR	0x0000000D	RCU APB0 复位控制寄存器
0x4002E058	0x58	RCU_APB1RCR	0x00000013	RCU APB1 复位控制寄存器
0x4002E05C	0x5C	RCU_AHB0RCR	0x000000FF	RCU AHB0 复位控制寄存器
0x4002E060	0x60	RCU_AHB1RCR	0x0000DF1F	RCU AHB1 复位控制寄存器
0x4002E070	0x70	RCU_XOSCCR	0x0001000B	RCU 晶振控制寄存器
0x4002E074	0x74	RCU_CSSCR	0x00007812	RCU 时钟安全寄存器
0x4002E078	0x78	RCU_PLLCFG	0x00006D6D	RCU 时钟配置寄存器
0x4002E080	0x80	RCU_DBGCR	0x00000000	RCU 时钟控制寄存器
0x4002E084	0x84	RCU_SCFGR	0x0000000B	RCU 系统配置寄存器
0x4002E088	0x88	RCU_RSTSR	0x00000000	RCU 复位状态寄存器
0x4002E090	0x90	RCU_KEY	0x00000000	RCU 键寄存器

8.3.2 寄存器详细描述

8.3.2.1 RCU PLL0 控制寄存器（RCU_PLL0CR）

- 名称：RCU PLL0 Control Register
- 偏移地址：0x00
- 默认值：0x00001400
- 返回寄存器列表

位	31:16	15	14	13:8	7:4	3:2	1:0
名称		EN	LOCK		DIV		SRC
访问		R/W	R		R/W		R/W
复位值		0x0	0x0		0x0		0x0

字段	说明
[15] EN	**PLL 使能（PLL Enable）** 0：关闭 1：使能 PLL
[14] LOCK	**PLL 锁定标志位（PLL Lock Status）** 0：PLL 未锁定 1：PLL 已锁定
[7:4] DIV	**PLL 输出分频设置（PLL Divide）** 0：1 分频　　1：2 分频 …　　N：N+1 分频
[1:0] SRC	**PLL 参考时钟源设置（PLL Refer Clock）** 00：HSE　　01：HSI 10：关闭　　11：Reserved

8.3.2.2 RCU PLL0 分频寄存器（RCU_PLL0FDR）

- 名称：RCU PLL0 Feedback Divisor Register
- 偏移地址：0x04
- 默认值：0x000A0000
- 返回寄存器列表

位	31:30	29:16	15:0
名称		INT	FRAC
访问		R/W	R/W
复位值		0xa	0x0

字段	说明
[29:16] INT	**Feedback 分频系数整数部分（Feedback Division Integer Part）** 参考下面小数部分描述
[15:0] FRAC	**Feedback 分频系数小数部分（Feedback Division Fraction Part）** Note: 1、PLLxClk 的频率经过硬件二分频后作为 Feedback 输入时钟； 2、Feedback 输入时钟频率经过该寄存器配置(Int+Frac)分频后得到 PLLx 的 FeedBack 输出时钟，需确保 FeedBack 输出时钟频率与 PLLx 的参考时钟源频率相等； 例如: 如果选取 8M 晶振为 PLLx 的参考时钟源，需要 PLLx 输出频率为 160M，即 PLLxClk(160M)时钟先进行硬件 2 分频得到除频时钟 80M；然后 80M 时钟频率除以参考时钟源 8M 就可得到除频系数，即可得到 Feedback 除频系数为 Int=10，Frac=0；

8.3.2.3 RCU PLL1 控制寄存器（RCU_PLL1CR）

- **名称：RCU PLL1 Control Register**
- **偏移地址：0x08**
- **默认值：0x00001400**
- 返回寄存器列表

位	31:16	15	14	13:8	7:4	3:2	1:0
名称		EN	LOCK		DIV		SRC
访问		R/W	R		R/W		R/W
复位值		0x0	0x0		0x0		0x0

字段	说明
[15] EN	**PLL 使能（PLL Enable）** 0：关闭 1：使能 PLL
[14] LOCK	**PLL 锁定标志位（PLL Lock Status）** 0：PLL 未锁定 1：PLL 已锁定
[7:4] DIV	**PLL 输出分频设置（PLL Divide）** 0：1 分频　　1：2 分频 …　　N：N+1 分频
[1:0] SRC	**PLL 参考时钟源设置（PLL Refer Clock）** 00：HSE　　01：HSI 10：关闭　　11：Reserved

8.3.2.4 RCU PLL1 分频寄存器（RCU_PLL1FDR）

- 名称：RCU PLL1 Feedback Divisor Register
- 偏移地址：0x0C
- 默认值：0x000A0000
- 返回寄存器列表

位	31:30	29:16	15:0
名称		INT	FRAC
访问		R/W	R/W
复位值		0xa	0x0

字段	说明
[29:16] INT	**Feedback 分频系数整数部分（Feedback Division Integer Part）** 参考下面小数部分描述
[15:0] FRAC	**Feedback 分频系数小数部分（Feedback Division Fraction Part）** Note: 1、PLLxClk 的频率经过硬件二分频后作为 Feedback 输入时钟； 2、Feedback 输入时钟频率经过该寄存器配置(Int+Frac)分频后得到 PLLx 的 FeedBack 输出时钟，需确保 FeedBack 输出时钟频率与 PLLx 的参考时钟源频率相等； 例如：如果选取 8M 晶振为 PLLx 的参考时钟源，需要 PLLx 输出频率为 160M，即 PLLxClk(160M)时钟先进行硬件 2 分频得到除频时钟 80M；然后 80M 时钟频率除以参考时钟源 8M 就可得到除频系数，即可得到 Feedback 除频系数为 Int=10，Frac=0；

8.3.2.5 RCU 系统时钟控制寄存器（RCU_SCLKCR）

- 名称：RCU SYSCLK Control Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:24	23:16	15:2	1:0
名称		DIV		SRC
访问		R/W		R/W
复位值		0x0		0x0

字段	说明
[23:16] DIV	**SYSCLK 时钟分频设置（SYSCLK Clock Divider）** 0：1 分频（不分频） 1：2 分频 … N：N+1 分频

[1:0]	**SYSCLK 时钟源选择（SYSCLK Clock Source Selection）**
SRC	00：HSI　　01：LSI
	10：PLL0/N 分频　11：XOSC

8.3.2.6 RCU 总线时钟控制寄存器（RCU_BCLKCR）

- **名称：RCU BUSCLK Control Register**
- **偏移地址：0x24**
- **默认值：0x000F0103**
- 返回寄存器列表

位	31:20	19	18	17	16	15:8	7:0
名称		P1EN	P0EN	H1EN	H0EN	P1DIV	P0DIV
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x1	0x1	0x1	0x1	0x1	0x3

字段	说明
[19]	**APB1 总线时钟使能（APB1 Bus Clock Enable）**
P1EN	0：关闭
	1：启动
[18]	**APB0 总线时钟使能（APB0 Bus Clock Enable）**
P0EN	0：关闭
	1：启动
[17]	**AHB1 总线时钟使能（AHB1 Bus Clock Enable）**
H1EN	0：关闭
	1：启动
[16]	**AHB0 总线时钟使能（AHB0 Bus Clock Enable）**
H0EN	0：关闭
	1：启动
[15:8]	**APB1 时钟分频设置(FCLK/N - 默认 2 分频)（APB1 Clock Divider）**
P1DIV	1：2 分频
	…
	N：N+1 分频
[7:0]	**APB0 时钟分频设置(FCLK/N - 默认 4 分频)（APB0 Clock Divider）**
P0DIV	3：4 分频
	…
	N：N+1 分频

8.3.2.7 RCU 功能时钟源寄存器（RCU_FCLKSR）

- 名称：RCU FUNCLK Source Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:24	23:22	21:20	19:18	17:16	15:8	7:6	5:4	3:2	1:0
名称		PDS	PCS	PBS	PAS		DFLSS	FLSS	ADCS	PWMS
访问		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0		0x0	0x0	0x0	0x0

字段	说明
[23:22]	**GPIOD 消抖时钟源设置（GPIOD Debounce Clock Source）**
PDS	00：LSI 01：XOSC
	10：FCLK 11：HSI
[21:20]	**GPIOC 消抖时钟源设置（GPIOC Debounce Clock Source）**
PCS	00：LSI 01：XOSC
	10：FCLK 11：HSI
[19:18]	**GPIOB 消抖时钟源设置（GPIOB Debounce Clock Source）**
PBS	00：LSI 01：XOSC
	10：FCLK 11：HSI
[17:16]	**GPIOA 消抖时钟源设置（GPIOA Debounce Clock Source）**
PAS	00：LSI 01：XOSC
	10：FCLK 11：HSI
[7:6]	**DFLASH 功能时钟源设置（DFLASH Memory Clock Source）**
DFLSS	00：HSI 01：PLL0/N
	10：PLL1/N 11：Reserved
[5:4]	**FLASH 功能时钟源设置（FLASH Memory Clock Source）**
FLSS	00：HSI 01：PLL0/N
	10：PLL1/N 11：Reserved
[3:2]	**ADC 功能时钟源设置（ADC Function Clock Source）**
ADCS	00：HSI 01：XOSC
	10：PLL0 11：PLL1
[1:0]	**EPWM 功能时钟源设置（EPWM Function Clock Source）**
PWMS	00：HSI 01：XOSC
	10：PLL0 11：PLL1

8.3.2.8 RCU 功能时钟分频寄存器 0（RCU_FCLKDR0）

- 名称：RCU FUNCLK Divide Register 0
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:8	7:4	3:2	1:0
名称		DFLSD	FLSD	ADCD	PWMD
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0

字段	说明
[11:8]	**DFLASH 功能时钟分频设置（DFLASH Function Clock Divide）**
DFLSD	0：1 分频（不分频） 1：2 分频
	… N：N+1 分频
[7:4]	**FLASH 功能时钟分频设置（FLASH Function Clock Divide）**
FLSD	0：1 分频（不分频） 1：2 分频
	… N：N+1 分频
[3:2]	**ADC 功能时钟分频设置（ADC Function Clock Divide）**
ADCD	0：1 分频（不分频） 1：2 分频
	… N：N+1 分频
[1:0]	**EPWM 功能时钟分频设置（EPWM Function Clock Divide）**
PWMD	0：1 分频（不分频） 1：2 分频
	… N：N+1 分频

8.3.2.9 RCU 功能时钟分频寄存器 1（RCU_FCLKDR1）

- 名称：RCU FUNCLK Divide Register 1
- 偏移地址：0x34
- 默认值：0x00000000
- 返回寄存器列表

位	31:24	23:16	15:8	7:0
名称	PDD	PCD	PBD	PAD
访问	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0

字段	说明
[31:24]	**GPIOD 消抖时钟分频设置（GPIOD Debounce Clock Divide）**
PDD	0：1 分频（不分频）　1：2 分频
	…　N：N+1 分频
[23:16]	**GPIOC 消抖时钟分频设置（GPIOC Debounce Clock Divide）**
PCD	0：1 分频（不分频）　1：2 分频
	…　N：N+1 分频
[15:8]	**GPIOB 消抖时钟分频设置（GPIOB Debounce Clock Divide）**
PBD	0：1 分频（不分频）　1：2 分频
	…　N：N+1 分频
[7:0]	**GPIOA 消抖时钟分频设置（GPIOA Debounce Clock Divide）**
PAD	0：1 分频（不分频）　1：2 分频
	…　N：N+1 分频

8.3.2.10 RCU APB0 时钟控制寄存器（RCU_APB0CCR）

- 名称：RCU APB0CLK Control Register
- 偏移地址：0x38
- 默认值：0x00000000
- 返回寄存器列表

位	31:4	3	2	1	0
名称		UART1E	UART0E		I2C0E
访问		R/W	R/W		R/W
复位值		0x0	0x0		0x0

字段	说明
[3] UART1E	**UART1 时钟使能（UART1 Clock Enable）** 0：关闭 1：启动
[2] UART0E	**UART0 时钟使能（UART0 Clock Enable）** 0：关闭 1：启动
[0] I2C0E	**I2C0 时钟使能（I2C0 Clock Enable）** 0：关闭 1：启动

8.3.2.11 RCU APB1 时钟控制寄存器（RCU_APB1CCR）

- 名称：RCU APB1CLK Control Register
- 偏移地址：0x3C
- 默认值：0x00000000
- 返回寄存器列表

位	31:5	4	3:2	1	0
名称		TMRE		SPI1E	SPI0E
访问		R/W		R/W	R/W
复位值		0x0		0x0	0x0

字段	说明
[4] TMRE	**TMR0~3 时钟使能（TMR0~3 Clock Enable）** 0：关闭 1：启动

[1] SPI1E	**SPI1 时钟使能（SPI1 Clock Enable）** 0：关闭 1：启动
[0] SPI0E	**SPI0 时钟使能（SPI0 Clock Enable）** 0：关闭 1：启动

8.3.2.12 RCU AHB0 时钟控制寄存器（RCU_AHB0CCR）

- **名称：RCU AHB0CLK Control Register**
- **偏移地址：0x40**
- **默认值：0x00000002**
- 返回寄存器列表

位	31:10	7	6	5	4	3	2	1	0
名称		TMRE	PDE	PCE	PBE	PAE	EPRE	FLSE	DMAE
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x0

字段	说明
[7] TMRE	**TMR4~7 时钟使能（TMR4~7 Clock Enable）** 0：关闭 1：启动
[6] PDE	**GPIOD 时钟使能（GPIOD Clock Enable）** 0：关闭 1：启动
[5] PCE	**GPIOC 时钟使能（GPIOC Clock Enable）** 0：关闭 1：启动
[4] PBE	**GPIOB 时钟使能（GPIOB Clock Enable）** 0：关闭 1：启动
[3] PAE	**GPIOA 时钟使能（GPIOA Clock Enable）** 0：关闭 1：启动
[2] DFLSE	**DFLASH 时钟使能（DFLASH Clock Enable）** 0：关闭 1：启动
[1] FLSE	**FLASH 时钟使能（FLASH Clock Enable）** 0：关闭 1：启动

[0]	**DMA 时钟使能（DMA Clock Enable）**
DMAE	0：关闭
	1：启动

8.3.2.13 RCU AHB1 时钟控制寄存器（RCU_AHB1CCR）

- **名称：RCU AHB1CLK Control Register**
- **偏移地址：0x44**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16		15	14	13	12	11	10
名称			IQDIVE	CORDICE		PWME	DACCMP2E	DACCMP1E
访问			R/W	R/W		R/W	R/W	R/W
复位值			0x0	0x0		0x0	0x0	0x0
位	9	8	7:5	4	3	2	1	0
名称	DACCMP0E	ADCE		CANE	PDM1E	PDM0E	QEI1E	QEI0E
访问	R/W	R/W		R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0		0x0	0x0	0x0	0x0	0x0

字段	说明
[15]	**IQDIV 时钟使能（IQDIV Clock Enable）**
IQDIVE	0：关闭
	1：启动
[14]	**CORDIC 时钟使能（CORDIC Clock Enable）**
CORDICE	0：关闭
	1：启动
[12]	**EPWM 时钟使能（EPWM Clock Enable）**
PWME	0：关闭
	1：启动
[11]	**DACCMP2 时钟使能（DACCMP2 Clock Enable）**
DACCMP2E	0：关闭
	1：启动
[10]	**DACCMP1 时钟使能（DACCMP1 Clock Enable）**
DACCMP1E	0：关闭
	1：启动
[9]	**DACCMP0 时钟使能（DACCMP0 Clock Enable）**
DACCMP0E	0：关闭
	1：启动

[8] ADCE	**ADC 时钟使能（ADC Clock Enable）** 0：关闭 1：启动
[4] CANE	**CAN 时钟使能（CAN Clock Enable）** 0：关闭 1：启动
[3] PDM1E	**PDM1 时钟使能（PDM1 Clock Enable）** 0：关闭 1：启动
[2] PDM0E	**PDM0 时钟使能（PDM0 Clock Enable）** 0：关闭 1：启动
[1] QEI1E	**QEI1 时钟使能（QEI1 Clock Enable）** 0：关闭 1：启动
[0] QEI0E	**QEI0 时钟使能（QEI0 Clock Enable）** 0：关闭 1：启动

8.3.2.14 RCU 功能时钟控制寄存器（RCU_FUNCCR）

- **名称：RCU FUNCLK Control Register**
- **偏移地址：0x48**
- **默认值：0x00000001**
- 返回寄存器列表

位	31:4	3	2	1	0
名称		PWME	ADCE	DFLSE	FLSE
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x1

字段	说明
[3] PWME	**EPWM 功能时钟使能（EPWM Function Clock Enable）** 0：关闭 1：启动
[2] ADCE	**ADC 功能时钟使能（ADC Function Clock Enable）** 0：关闭 1：启动
[1] DFLSE	**DFLASH 功能时钟使能（DFLASH Function Clock Enable）** 0：关闭 1：启动

[0]	**FLS 功能时钟使能（FLASH Function Clock Enable）**
FLSE	0：关闭
	1：启动

8.3.2.15 RCU 系统复位控制寄存器（RCU_SYSRCR）

- **名称：RCU SYSRST Control Register**
- **偏移地址：0x50**
- **默认值：0x00000F0F**
- 返回寄存器列表

位	31:12	11	10	9	8	7:4	3	2	1	0
名称		PDR	PCR	PBR	PAR		P1R	P0R	H1R	H0R
访问		R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W
复位值		0x1	0x1	0x1	0x1		0x1	0x1	0x1	0x1

字段	说明
[11]	**GPIOD 消抖复位控制（GPIOD Debounce Reset）**
PDR	0：发起复位
	1：释放复位
[10]	**GPIOC 消抖复位控制（GPIOC Debounce Reset）**
PCR	0：发起复位
	1：释放复位
[9]	**GPIOB 消抖复位控制（GPIOB Debounce Reset）**
PBR	0：发起复位
	1：释放复位
[8]	**GPIOA 消抖复位控制（GPIOA Debounce Reset）**
PAR	0：发起复位
	1：释放复位
[3]	**APB1 总线复位控制（APB1 Bus Reset）**
P1R	0：发起复位
	1：释放复位
[2]	**APB0 总线复位控制（APB0 Bus Reset）**
P0R	0：发起复位
	1：释放复位
[0]	**AHB1 总线复位控制（AHB1 Bus Reset）**
H1R	0：发起复位
	1：释放复位
[0]	**AHB0 总线复位控制（AHB0 Bus Reset）**
H0R	0：发起复位
	1：释放复位

8.3.2.16 RCU APB0 复位控制寄存器（RCU_APB0RCR）

- 名称：RCU APB0RST Control Register
- 偏移地址：0x54
- 默认值：0x0000000D
- 返回寄存器列表

位	31:4	3	2	1	0
名称		UART1RE	UART0RE		I2C0RE
访问		R/W	R/W		R/W
复位值		0x1	0x1		0x1

字段	说明
[3] UART1RE	**UART1 软复位控制（UART1 Reset）** 0：发起复位 1：释放复位
[2] UART0RE	**UART0 软复位控制（UART0 Reset）** 0：发起复位 1：释放复位
[0] I2C0RE	**I2C0 软复位控制（I2C0 Reset）** 0：发起复位 1：释放复位

8.3.2.17 RCU APB1 复位控制寄存器（RCU_APB1RCR）

- 名称：RCU APB1RST Control Register
- 偏移地址：0x58
- 默认值：0x00000013
- 返回寄存器列表

位	31:5	4	3:2	1	0
名称		TMRRE		SPI1RE	SPI0RE
访问		R/W		R/W	R/W
复位值		0x1		0x1	0x1

字段	说明
[4] TMRRE	**TMR0~3 软复位控制（TMR0~3 Reset）** 0：发起复位 1：释放复位

[1] SPI1RE	**SPI1 软复位控制（SPI1 Reset）** 0：发起复位 1：释放复位
[0] SPI0RE	**SPI0 软复位控制（SPI0 Reset）** 0：发起复位 1：释放复位

8.3.2.18 RCU AHB0 复位控制寄存器（RCU_AHB0RCR）

- **名称：RCU AHB0RST Control Register**
- **偏移地址：0x5C**
- **默认值：0x000000FF**
- 返回寄存器列表

位	31:10	7	6	5	4	3	2	1	0
名称		TMRRE	PDRE	PCRE	PBRE	PARE	DFLSRE	FLSRE	DMARE
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1

字段	说明
[7] TMRRE	**TMR4~7 软复位控制（TMR4~7 Reset）** 0：发起复位 1：释放复位
[6] PDRE	**GPIOD 软复位控制（GPIOD Reset）** 0：发起复位 1：释放复位
[5] PCRE	**GPIOC 软复位控制（GPIOC Reset）** 0：发起复位 1：释放复位
[4] PBRE	**GPIOB 软复位控制（GPIOB Reset）** 0：发起复位 1：释放复位
[3] PARE	**GPIOA 软复位控制（GPIOA Reset）** 0：发起复位 1：释放复位
[2] DFLSRE	**DFLASH 软复位控制（DFLASH Reset）** 0：发起复位 1：释放复位
[1] FLSRE	**FLASH 软复位控制（FLASH Reset）** 0：发起复位 1：释放复位

[0]	**DMA 软复位控制（DMA Reset）**
DMARE	0：发起复位
	1：释放复位

8.3.2.19 RCU AHB1 复位控制寄存器（RCU_AHB1RCR）

- **名称：RCU AHB1RST Control Register**
- **偏移地址：0x60**
- **默认值：0x0000DF1F**
- 返回寄存器列表

位	31:16		15	14	13	12	11	10
名称			IQDIVRE	CORDICRE		PWMRE	DACCMP2RE	DACCMP1RE
访问			R/W	R/W		R/W	R/W	R/W
复位值			0x1	0x1		0x1	0x1	0x1
位	9	8	7:5	4	3	2	1	0
名称	DACCMP0RE	ADCRE		CANRE	PDM1RE	PDM0RE	QEI1RE	QEI0RE
访问	R/W	R/W		R/W	R/W	R/W	R/W	R/W
复位值	0x1	0x1		0x1	0x1	0x1	0x1	0x1

字段	说明
[15]	**IQDIV 软复位控制（IQDIV Reset）**
IQDIVRE	0：发起复位
	1：释放复位
[14]	**CORDIC 软复位控制（CORDIC Reset）**
CORDICRE	0：发起复位
	1：释放复位
[12]	**EPWM 软复位控制（EPWM Reset）**
PWMRE	0：发起复位
	1：释放复位
[11]	**DACCMP2 软复位控制（DACCMP2 Reset）**
DACCMP2RE	0：发起复位
	1：释放复位
[10]	**DACCMP1 软复位控制（DACCMP1 Reset）**
DACCMP1RE	0：发起复位
	1：释放复位
[9]	**DACCMP0 软复位控制（DACCMP0 Reset）**
DACCMP0RE	0：发起复位
	1：释放复位

[8] ADCRE	**ADC 软复位控制（ADC Reset）** 0：发起复位 1：释放复位
[4] CANRE	**CAN 软复位控制（CAN Reset）** 0：发起复位 1：释放复位
[3] PDM1RE	**PDM1 软复位控制（PDM1 Reset）** 0：发起复位 1：释放复位
[2] PDM0RE	**PDM0 软复位控制（PDM0 Reset）** 0：发起复位 1：释放复位
[1] QEI1RE	**QEI1 软复位控制（QEI1 Reset）** 0：发起复位 1：释放复位
[0] QEI0RE	**QEI0 软复位控制（QEI0 Reset）** 0：发起复位 1：释放复位

8.3.2.20 RCU 晶振控制寄存器（RCU_XOSCCR）

- **名称：RCU XOSC Control Register**
- **偏移地址：0x70**
- **默认值：0x0001000B**
- 返回寄存器列表

位	31:18	17	16	15:4	3:1	0
名称		EHS	HIE		XDR	XEN
访问		R/W	R/W		R/W	R/W
复位值		0x0	0x1		0x5	0x1

字段	说明
[17] EHS	**内部 XOSC 时钟源选择（XOSC Source Select）** 0: 来源于 XOSC 起振电路 1: 来源于外部 CLKIN
[16] HIE	**内部高频振荡器使能（HSI Enable）** 0：关闭 1：启动
[3:1] XDR	**外部晶振启动电流配置（HSE Startup Current Configuration）** 000：0.25mA 001：0.375mA 010：0.5mA 011：0.625mA 100：0.75mA 101：0.875mA 110：1.0mA 111：1.125mA

[0]	**外部晶振使能控制（HSE Enable）**
XEN	0：关闭
	1：启动

8.3.2.21 RCU 时钟安全寄存器（RCU_CSSCR）

- **名称：RCU CSS Control Register**
- **偏移地址：0x74**
- **默认值：0x00007812**
- 返回寄存器列表

位	31:30	29	28	27	26	25	24	23:0
名称		XLPD	XLPE		SSE	PSE	SWE	
访问		R/W1C	R/W		R/W	R/W	R/W	
复位值		0x0	0x0		0x0	0x0	0x0	

字段	说明
[29]	**外部晶振异常状态位（HSE Abnormal Status）**
XLPD	0：无异常
	1：发生复位
	注意：该位为状态位，写 1 清 0；
[28]	**XOSC 中断使能控制（XOSC Interrupt Enable）**
XLPE	0:关闭
	1:启动
[26]	**系统时钟自动切换使能（SYSCLK Automatic Switch Enable ）**
SSE	0：关闭
	1：使能自动切换功能
	注意：使能该功能后，当外部晶振出现异常，系统时钟将自动切换到 HSI（内部高频振荡器）；
[25]	**PLL 参考时钟自动切换使能（PLL Reference Clock Automatic Switch Enable）**
PSE	0：关闭
	1：使能自动切换功能
	注意：使能该功能后，当外部晶振出现异常，PLL 参考时钟将自动切换到 HSI（内部高频振荡器）；
[24]	**时钟安全监测功能使能（Clock Security Monitoring Function Enable）**
SWE	0：关闭
	1：使能时钟安全监测

8.3.2.22 RCU PLL 配置寄存器（RCU_PLLCFG）

- 名称：RCU PLLx Config Register
- 偏移地址：0x78
- 默认值：0x00006D6D
- 返回寄存器列表

位	31:16	15	14	13:12	11:10	9	8
名称		VCO1	OPA1	GVCO1	P1GCP	P1LPF	P1PD
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x1	0x2	0x3	0x0	0x1
位	7	6	5:4	3:2	1	0	
名称	VCO0	OPA0	GVCO0	P0GCP	P0LPF	P0PD	
访问	R/W	R/W	R/W	R/W	R/W	R/W	
复位值	0x0	0x1	0x2	0x3	0x0	0x1	

字段	说明
[15] VCO1	**PLL1 VCO 测试电压使能控制（PLL1 VCO Test Voltage Enable）** 0:关闭 1:启动
[14] OPA1	**PLL1 VI 转换器输入控制（PLL1 VI Converter Input）** 0:PMOS 对输入 1:轨对轨输入
[13:12] GVCO1	**PLL1 VCO 频率调节增益控制（PLL1 VCO Frequency Adjustment Gain）** 00:最小值 11:最大值
[11:10] P1GCP	**PLL1 Charge Pump 电流控制（PLL1 Charge Pump Current）** 00:1uA 01:2uA 10:3uA 11:4uA
[9] P1LPF	**PLL1 参考时钟频率（PLL1 Reference Clock Frequency）** 1:26M 0:8M
[8] P1PD	**PLL1 参考时钟预分频控制（PLL1 Reference Clock Prescaler）** 0:2 分频 1:不分频
[7] VCO0	**PLL0 VCO 测试电压使能控制（PLL0 VCO Test Voltage Enable）** 0:关闭 1:启动
[6] OPA0	**PLL0 VI 转换器输入控制（PLL0 VI Converter Input）** 0:PMOS 对输入 1:轨对轨输入

[5:4] GVCO0	**PLL0 VCO 频率调节增益控制（PLL0 VCO Frequency Adjustment Gain）** 00:最小值 11:最大值
[3:2] P0GCP	**PLL0 Charge Pump 电流控制（PLL0 Charge Pump Current）** 00:1uA 01:2uA 10:3uA 11:4uA
[1] P0LPF	**PLL0 参考时钟频率（PLL0 Reference Clock Frequency）** 1:26M 0:8M
[0] P0PD	**PLL0 参考时钟预分频控制（PLL0 Reference Clock Prescaler）** 0:2 分频 1:不分频

8.3.2.23 RCU 调试控制寄存器（RCU_ DBGCR）

- **名称：RCU Debug Control Register**
- **偏移地址：0x80**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:7	6:4	3:2	1	0
名称		SRC		CFE	TIE
访问		R/W		R/W	R/W
复位值		0x0		0x0	0x0

字段	说明
[6:4] SRC	**内部时钟 Fanout 源选择（Clock Fanout Source）** 000:LSI 001:HSI 010:XOSC 011:FCLK 100:PLL0DivClk 101:PLL0/16 110:PLL1DivClk 111:PLL1/16
[1] CFE	**内部时钟源 FanOut 输出使能(具体时钟源见上)（Clock Fanout Enable）** 0:关闭 1:启动
[0] TIE	**TEST (ATE) 时钟输入使能（TEST(ATE) CLK Input Enable）** 0:关闭 1:启动

8.3.2.24 RCU 系统配置寄存器（RCU_ SCFGR）

- 名称：RCU System Config Register
- 偏移地址：0x84
- 默认值：0x0000000B
- 返回寄存器列表

位	31:4	3	2	1	0
名称		SREQRE	LKUPRE	WWDGRE	IWDGRE
访问		R/W	R/W	R/W	R/W
复位值		0x1	0x0	0x1	0x1

字段	说明
[3] SREQRE	**CPU SYSTEMREQ 复位使能控制（CPU SYSTEMREQ Reset Enable）** 0:关闭 1:启动
[2] LKUPRE	**CPU LOCKUP 复位使能控制（CPU LOCKUP Reset Enable）** 0:关闭 1:启动
[1] WWDGRE	**WWDG 复位使能控制（WWDG Reset Enable）** 0:关闭 1:启动
[0] IWDGRE	**IWDG 复位使能控制（IWDG Reset Enable）** 0:关闭 1:启动

8.3.2.25 RCU 复位状态寄存器（RCU_RSTSR）

- 名称：**RCU Reset Status Register**
- 偏移地址：**0x88**
- 默认值：**0x00000000**
- 返回寄存器列表

位	31:6	5	4	3	2	1	0
名称		WWRST	IWRST	LKRST	SQRST	LPRST	MCRST
访问		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[5] WWRST	**WWDG 复位状态位（WWDG RESET Status）** 0:无 Reset 1: WWDG 产生 Reset
[4] IWRST	**IWDG 复位状态位（IWDG RESET Status）** 0:无 Reset 1: IWDG 产生 Reset
[3] LKRST	**LKUP 复位状态位（LKUP RESET Status）** 0:无 Reset 1: LKUP 产生 Reset
[2] SQRST	**SystemREQ 复位状态位（SystemREQ RESET Status）** 0:无 Reset 1: SystemREQ 产生 Reset
[1] LPRST	**LowPower 复位状态位（LowPower RESET Status）** 0:无 Reset 1: LowPower 产生 Reset
[0] MCRST	**MCLR 复位状态位（MCLR RESET Status）** 0:无 Reset 1: MCLR 产生 Reset

8.3.2.26 RCU 键寄存器（RCU_KEY）

- 名称：RCU LockKey Register
- 偏移地址：0x90
- 默认值：0x00000000
- 返回寄存器列表

位	31:1	0
名称		KEY
访问		R/W
复位值		0x0

字段	说明
[0]	**系统寄存器写操作的密码保护配置（Key Protection）**
KEY	1、向 KEY 寄存器写入 Key 值(0x3fac87e4)，然后进行相关的写寄存器操作； 2、向 KEY 寄存器写入非 Key 值 0x3fac87e4 时，将进行加锁；

9 系统配置控制器（SYSCTRL）

系统配置控制器主要用于管理系统时钟、复位及调试相关功能，具体参考下节寄存器描述。

9.1 寄存器描述

9.1.1 寄存器列表

SYSCTRL 基地址：**0x4002 F000**

偏移	实例地址	名称	默认值	描述
0x00	0x4002F000	SYSCTRL_SYSCR	0x000000C0	SYSCTRL 系统控制寄存器
0x04	0x4002F004	SYSCTRL_TCCR	0x00000000	SYSCTRL 测试控制寄存器
0x08	0x4002F008	SYSCTRL_PMUCR	0x00000682	SYSCTRL PMU 控制寄存器
0x40	0x4002F040	SYSCTRL_KEY	0x00000000	SYSCTRL 键寄存器
0x44	0x4002F044	SYSCTRL_CIDCR	0x00000000	SYSCTRL CHIPID 寄存器

9.1.2 寄存器详细描述

9.1.2.1 SYSCTRL 系统控制寄存器（SYSCTRL_SYSCR）

- 名称：SYSCTRL SYSCFG Control Register
- 偏移地址：0x00
- 默认值：0x000000C0
- 返回寄存器列表

位	31:17	16	15:8	7	6	5	4	3:1	0
名称		NMIE		WWDGDE	IWDGDE	TMR1DE	TMR0DE		PMUDE
访问		R/W		R/W	R/W	R/W	R/W		R/W
复位值		0x0		0x1	0x1	0x0	0x0		0x0

字段	说明
[16] NMIE	**GPIO NMI 中断使能控制（GPIO NMI Interrupt Enable）** 0:关闭 1:启动
[7] WWDGDE	**WWDG 调试使能控制（WWDG Debug Enable）** 0:关闭 1:启动 Note: 当内核进入调试模式，WWDG 内部计数器是否自动关闭；
[6] IWDGDE	**IWDG 调试使能控制（IWDG Debug Enable）** 0:关闭 1:启动 Note: 当内核进入调试模式，IWDG 内部计数器是否自动关闭；
[5] TMR1DE	**TMR4~7 调试使能控制（TMR4~7 Debug Enable）** 0:关闭 1:启动 Note: 当内核进入调试模式，TMR4~7 内部计计数器是否自动关闭；
[4] TMR0DE	**TMR0~3 调试使能控制（TMR0~3 Debug Enable）** 0:关闭 1:启动 Note: 当内核进入调试模式，TMR0~3 内部计数器是否自动关闭；
[0] PMUDE	**PMU 测试信号输出使能控制（PMU Test Signal Output Enable）** 0:关闭 1:启动

9.1.2.2 SYSCTRL 测试控制寄存器（SYSCTRL_TCCR）

- 名称：SYSCTRL Test Control Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:10	9	8	7:0
名称		ABFBYP	ABFEN	
访问		R/W	R/W	
复位值		0x0	0x0	

字段	说明
[9] ABFBYP	**ADC 缓存 Bypass 使能（ADC Buffer Bypass Enable）** 0:关闭 1:启动
[8] ABFEN	**ADC 缓存使能（ADC Buffer Enable）** 0:关闭 1:启动

9.1.2.3 SYSCTRL PMU 控制寄存器（SYSCTRL_PMUCR）

- 名称：SYSCTRL PMU Control Register
- 偏移地址：0x08
- 默认值：0x00000682
- 返回寄存器列表

位	31:11	10	9:8	7:4	3:2	1	0
名称		AVDD_DRD	AVDD_SET	VDD_SET		AVDD_EN	TSE
访问		R/W	R/W	R/W		R/W	R/W
复位值		0x1	0x2	0x8		0x1	0x0

字段	说明
[10] AVDD_DRD	**AVDD 的无电容 LDO 下拉配置选择（AVDD Drop Down）** 0：不发生下拉 1：0.5mA 下拉 启动时，如果打开下拉，可以帮助稳定和启动电路，当 26M 晶振开始工作时关闭下拉；
[9:8] AVDD_SET	**AVDD 电压控制，默认为 1.5V（AVDD Setting）** 00：1.3V　01：1.4V 10：1.5V　11：1.6V

[7:4]	**VDD(校准器)电压粗调控制，默认为 1.5V（VDD Setting）**
VDD_SET	0000：1.1V 0001：1.15V 0010：1.2V 0011：1.25V
	0100：1.3V 0101：1.35V 0110：1.4V 0111：1.45V
	1000：1.5V 1001：1.55V 1010：1.6V 1011：1.65V
	1100：1.7V 1101：1.75V 1110：1.8V 1111：1.85V
[1]	**AVDD 使能控制（AVDD Enable）**
AVDD_EN	0：关闭
	1：启动
[0]	**温度传感器使能控制（Temperature Sensor Enable）**
TSE	0：关闭
	1：启动

9.1.2.4 SYSCTRL 键寄存器（SYSCTRL_KEY）

- **名称：SYSCTRL LockKey Register**
- **偏移地址：0x40**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:1	0
名称		KEY
访问		R/W
复位值		0x0

字段	说明
[0]	**系统寄存器写操作的密码保护配置（Key Protection）**
KEY	内部寄存器保护 Key 为 0x3fac87e4，写寄存器之前必须写 Key 进行解锁；

9.1.2.5 SYSCTRL CHIPID 寄存器（SYSCTRL_CIDCR）

- 名称：SYSCTRL CHIPID Control Register
- 偏移地址：0x44
- 默认值：0x00000000
- 返回寄存器列表

位	31:24	23:8	7:0
名称		ID	DCN
访问		R	R
复位值		0x0	0x0

字段	说明
[23:8] ID	**CHIP ID**
[7:0] DCN	**CHIP DCN**

10 通用 I/O（GPIO）

10.1 简介

GPIO 是可编程的通用 I/O 外设（Programmable General Purpose I/O），芯片内共包含 4 组 GPIO（GPIOA~D），其中每组 GPIO 均有 16 个独立的 I/O。

10.2 结构框图

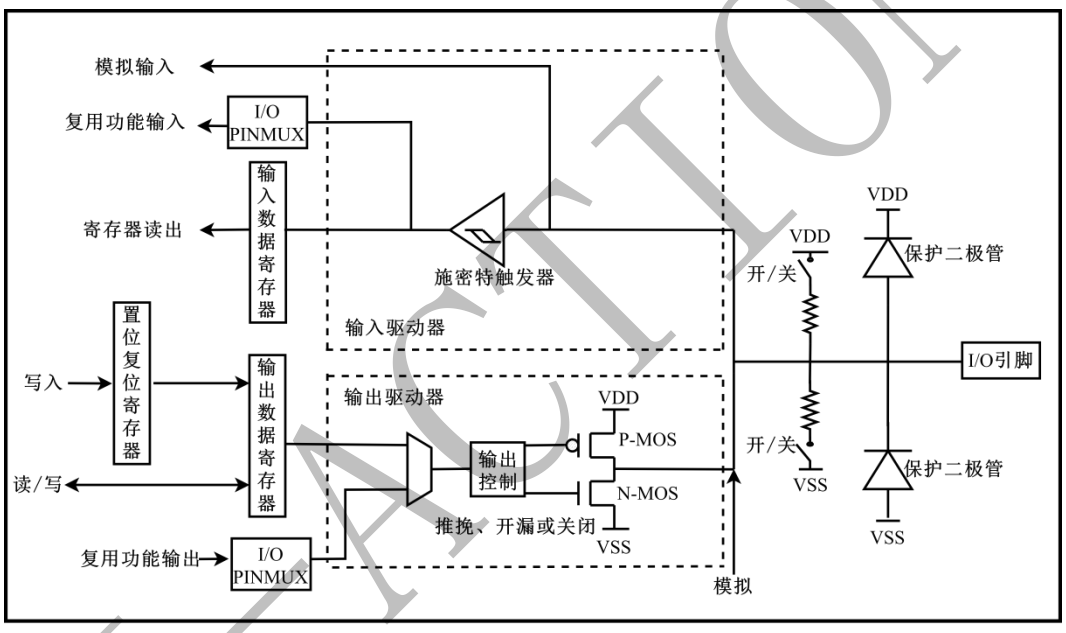


图 10-1　GPIO 结构框图

10.3 主要特性

GPIO 主要具有以下特性：

- 芯片内受控 I/O 为 64 个
- 输出状态：推挽或开漏 + 上拉/下拉
- 输入状态：浮空、上拉/下拉、模拟
- 输入功能：I/O 输入数据寄存器
- 输出功能：I/O 输出数据寄存器
- 复用功能：允许将 I/O 配置为 GPIO 或各种外设功能
- 模拟功能
- 置位和复位寄存器，可实现对输出数据寄存器按位操作
- 可选的同步功能和消抖功能
- 可配的电流驱动能力
- 每个 I/O 均可独立配置输出压摆率(Slew Rate)和输入迟滞(Hysteresis)
- 所有 I/O 均可独立配置外部输入触发中断使能，上升沿/下降沿/双边沿触发，独立的中断挂起标志

10.4 功能描述

通过软件可将 GPIO 各个端口分别配置为以下多种模式：

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟功能
- 具有上拉或下拉的开漏输出
- 具有上拉或下拉的推挽输出
- 具有上拉或下拉的复用功能推挽
- 具有上拉或下拉的复用功能开漏

每个 I/O 端口的工作模式均可自由配置，GPIO 内部寄存器均按 32 位(Word)进行访问。其中可以通过 GPIO_BSR 寄存器实现对 GPIO_ODR 输出数据寄存器的按位操作。

10.4.1 I/O 复位状态

芯片复位期间：

- PB4/PB6 在芯片复位期间为下拉输入状态，复位完成后为浮空状态。
- PC11/PC14 将默认复用为调试功能(AF2)，PC11 处于下拉状态，PC14 处于上拉状态。

复位完成后：

- PC11/PC14 将默认复用为调试功能(AF2)
  - PC14：SWDAT 处于上拉状态
  - PC11：SWCLK 处于下拉状态
- 其他端口为浮空高阻模式

10.4.2 通用 I/O

当引脚功能配置为输出后，写入到输出数据寄存器(GPIO_ODR)的值将输出到 I/O 引脚上。

当引脚功能配置为输入后，输入数据寄存器(GPIO_IDR)每隔 1 个 CPU 时钟周期(FCLK)捕获一次 I/O 引脚的数据。

所有 GPIO 引脚都具有内部弱上/下拉功能，可通过配置 GPIOx_PUR/GPIOx_PDR 寄存器中的值来选择打开/关闭上下拉功能。

10.4.3 引脚复用

I/O 引脚通过一个复用逻辑单元连接到不同外设模块，该复用逻辑单元一次仅允许一个外设复用到 I/O 引脚上，这样可以确保共用一个 I/O 引脚的外设之间不会发生冲突。

芯片内部每个 I/O 引脚都有一个复用逻辑单元，其支持 8 路复用功能输入(AF0 至 AF7)，可通过 GPIOx_MR0(针对 I/O 引脚 0~7)和 GPIOx_MR1(针对 I/O 引脚 8~15)寄存器对复用功能进行配置：

- AF0 为 I/O 输入模式，AF1 为 I/O 输出模式
- 外设的复用功能映射到 AF2 至 AF6
- AF7 为模拟功能(Analog function)
- 芯片复位完成后，除 PC11/PC14 外(详见 10.4.1 I/O 复位状态)，其他 I/O 都会映射到系统的复用功能(AF7)

I/O 复用架构除了具有灵活性，各外设还可以将复用功能映射到不同 I/O 引脚上，这可以优化在小型封装中可使用外设的数量。

复用功能配置说明

系统及调试功能：

- SWD：复位后，会将 PC11/14 引脚指定为 SWD 功能，供片上调试使用。
- MCO：复位后，默认为 AF7，若要使用 MCO 复用功能必须配置为对应的复用功能模式。

GPIO：

- 在 GPIOx_MR0/GPIOx_MR1 寄存器中将所需 I/O 配置为输出或输入复用功能。

外设复用功能：

- 对于 ADC/DAC/CMP 模拟功能，在 GPIOx_MR0/GPIOx_MR1 寄存器中将所需 I/O 配置为模拟通道复用(AF7)。
- 其他外设复用功能，在 GPIOx_MR0/GPIOx_MR1 寄存器中将所需 I/O 配置为对应的复用功能。引脚的特定复用功能分配请参见 3.2 引脚描述。

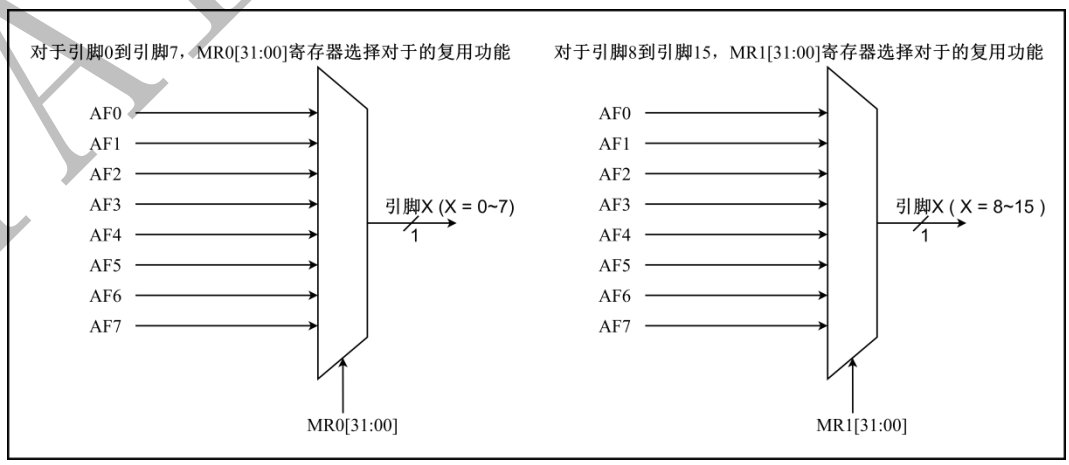


图 10-2 GPIO 选择复用功能示意图

10.4.4 I/O 控制寄存器

每组 GPIO 端口有 9 个控制寄存器(GPIOx_MR0、GPIOx_MR1、GPIOx_IDR、GPIOx_ODR、GPIOx_PUR、GPIOx_PDR、GPIOx_OSR、GPIOx_IHR、GPIOx_DHR)，每组 GPIO 可配置的 I/O 数为 16 个。

其中 GPIOx_MR0、GPIOx_MR1 寄存器用于选择 I/O 复用功能(输入、输出、外设引脚复用、模拟功能)；GPIOx_ODER、GPIOx_PUR、GPIOx_PDR 寄存器分别用于选择 I/O 输出类型(推挽或开漏)和上下拉功能；GPIOx_OSR、GPIOx_DHR、GPIOx_IHR 寄存器用于配置 I/O 输出摆率、输出驱动能力及输入迟滞。

注意：I/O 上下拉功能不区分输入和输出方向，都可以配置，有关寄存器说明的详细信息，请参见下面 10.5 寄存器描述。

10.4.5 I/O 数据寄存器

每组 GPIO 都包括一个输入数据寄存器和一个输出数据寄存器，其中输入数据寄存器（GPIOx_IDR）用于存储输入数据，输出数据寄存器（GPIOx_ODR）用于存储输出数据。

注意：有关寄存器说明的详细信息，请参见下面 10.5 寄存器描述。

10.4.6 I/O 数据位操作

每组 GPIO 都具有 1 个 32 位的置位/复位寄存器 (GPIOx_BSR)，其允许用户对输出数据中的单个数据位执行置位和复位操作。置位/复位寄存器(GPIOx_BSR)是一个 32 位寄存器，其中低 16 位用于 Bit Set 置位功能，而高 16 位用于 Bit Reset 复位功能。当向 GPIOx_BSR 寄存器的低 16 位中某一位写 1 时，其对应位 I/O 将置位；当向 GPIOx_BSR 寄存器的高 16 位中某一位写 1 时，其对应位 I/O 将清零。

注意：

- *当向 GPIOx_BSR 寄存器写入 0 对 Bit Set 和 Reset 都不会产生任何影响。*
- *当向 GPIOx_BSR 寄存器中 Bit Set 和 Reset 同时写入 1 时，将发起 Set 置位操作，Set 优先级更高。*
- *通过 GPIOx_BSR 寄存器可以实现 AHB 总线对 I/O 的原子写操作，实现寄存器单 Bit 或多 Bit 修改操作。*

10.4.7 I/O 复用功能

每组 GPIO 都具有 2 个 32 位的 I/O 功能复用寄存器，共 16 个 I/O，每个 I/O 对应 4Bit，用来选择 8 个复用功能。根据用户的需求可将功能复用寄存器 GPIOx_MRn(n = 0~1)配置为所需的复用功能，并将其功能映射到对应的引脚上。

使用 GPIOx_MRn 复用功能寄存器，可以在任意一个 I/O 上灵活选择一个功能复用。

10.4.8 外部中断功能

所有 GPIO 端口都具有外部中断功能，要使用外部中断功能，必须将端口配置为输入模式，同时配置 GPIOx_IER 寄存器开启 GPIO 端口的中断使能，再配置 GPIOx_ICR 触发使能寄存器选取所指定引脚的触发模式，可通过对 GPIOx_ISR 挂起标志位寄存器的检查来判断对应引脚是否触发了中断。

10.4.9 输入配置

当 I/O 端口配置为输入时：

- 输出寄存器被禁止
- 输入迟滞可配
- 根据配置 GPIOx_PUR /GPIOx_PDR 寄存器来决定 I/O 是否打开上拉和下拉电阻
- 输入数据寄存器每隔 1 个 CPU 时钟周期(F_{CLK})对 I/O 引脚上的数据进行一次采样
- 读取输入数据寄存器可获得对应 I/O 的状态

输入模式

注意：所有 GPIO 引脚都具有一个上拉和下拉电阻，通过对应寄存器配置可以打开或关闭。

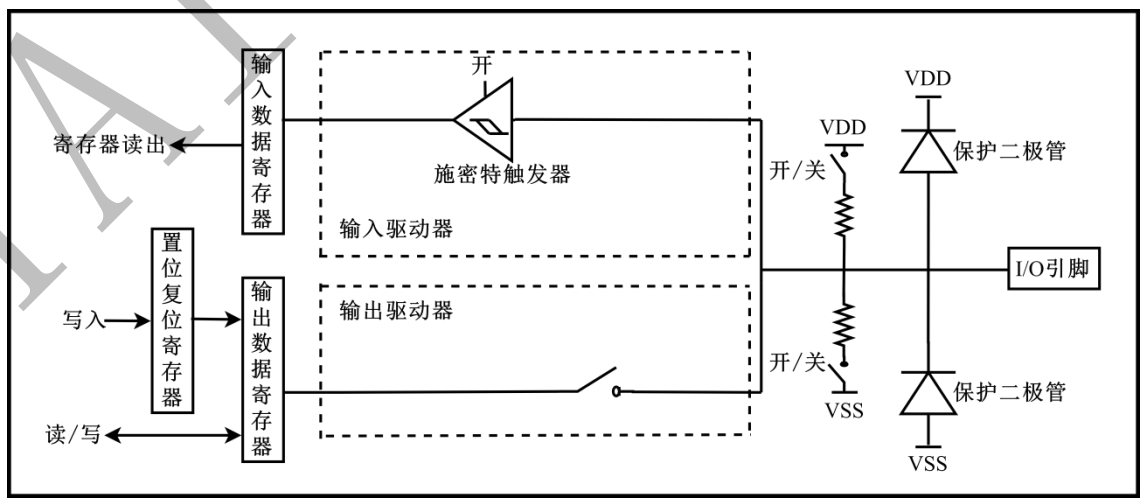


图 10-3 GPIO 输入上拉/下拉配置

浮空状态

当 I/O 引脚的上拉和下拉电阻都处于断开状态，此时 I/O 引脚处于悬空状态，容易收到外部干扰。

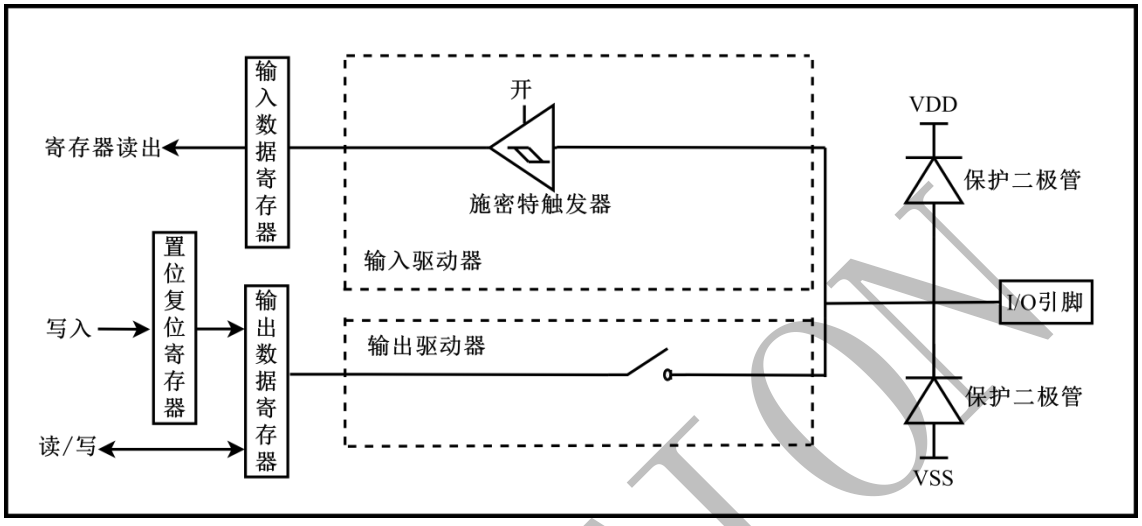


图 10-4　GPIO 输入浮空配置

输入上拉

打开 I/O 引脚的上拉电阻，将 I/O 引脚的不确定信号通过一个电阻嵌到高电平，电阻起到限流作用。

通常可以给 I/O 引脚外接一个开关，开关一端接 I/O 引脚，一端接地。当按下按钮时，读出低电平；松开按钮时，读出高电平。也可外接一个强上拉电阻，一直读取到高电平。

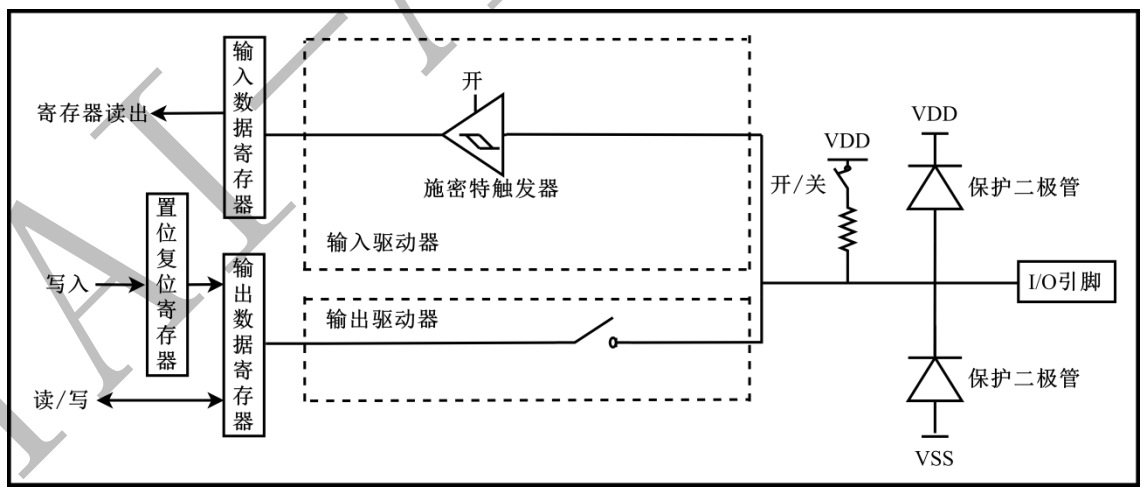


图 10-5　GPIO 输入上拉配置

输入下拉

与上拉输入原理一样，将电位拉到 GND，读到低电平。

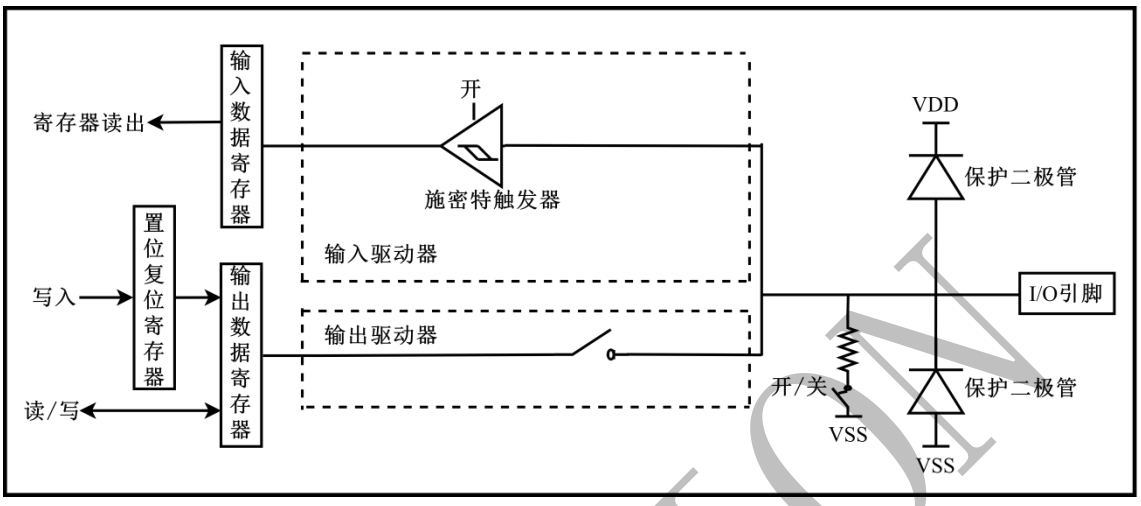


图 10-6　GPIO 输入下拉配置

10.4.10　输出配置

当 I/O 端口被配置为输出时：

- 输出数据寄存器被使能，输入数据寄存器同时使能，实时采样输出数据
  - 开漏模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”会使端口保持高阻状态(P-MOS 始终不激活)。
  - 推挽模式：输出寄存器中的“0”可激活 N-MOS，而输出寄存器中的“1”可激活 P-MOS。
- 根据配置 GPIOx_PUR /PDR 寄存器来决定 I/O 是否打开上拉和下拉电阻

输出模式

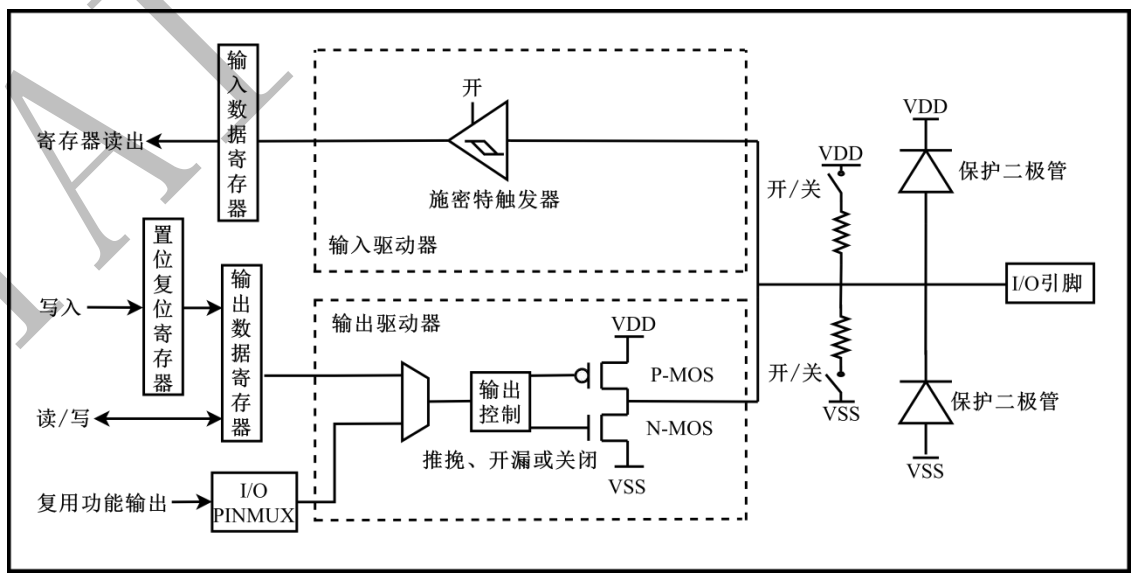


图 10-7　GPIO 输出推挽/开漏或关闭配置

推挽输出

可以输出高、低电平，连接数字器件；推挽结构一般是指两个三极管分别受两互补信号的控制，总是在一个三极管导通的时候另一个截止。

推挽输出，写 1

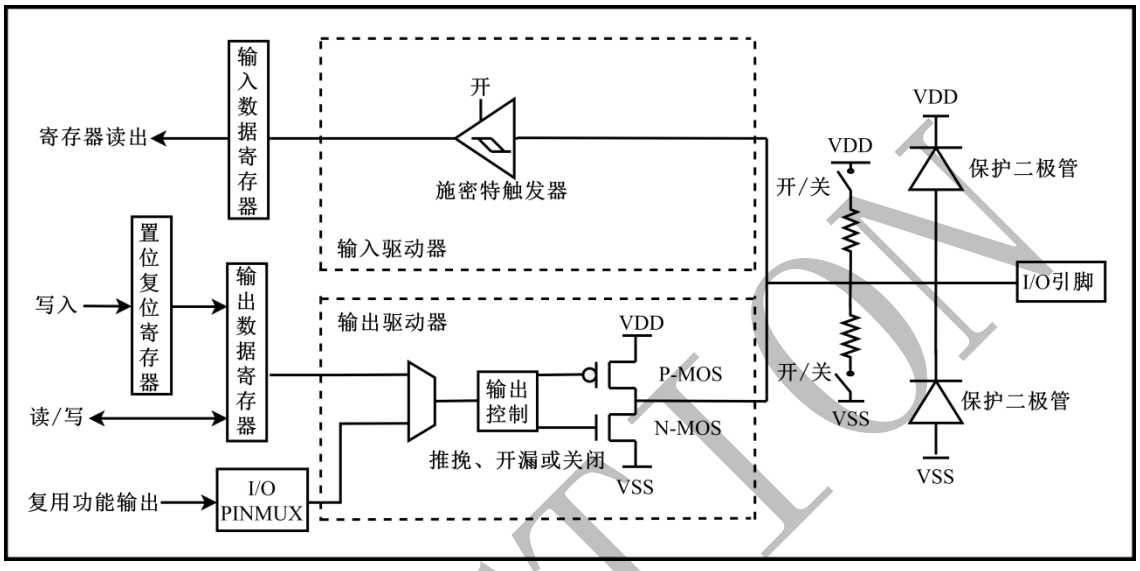


图 10-8　GPIO 推挽输出配置（写 1）

P-MOS 管激活，连接 VDD，I/O 引脚输出高电平，输入数据寄存器读到高电平。

推挽输出，写 0

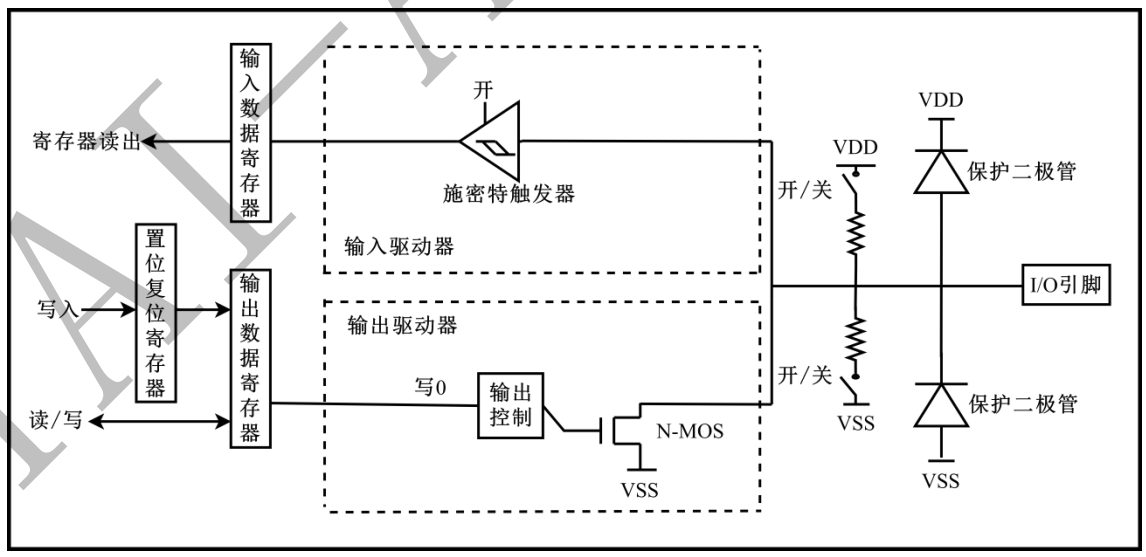


图 10-9　GPIO 推挽输出配置（写 0）

N-MOS 管被激活，连接到 VSS 接地，I/O 引脚输出低电平，输入数据寄存器读到低电平。

开漏输出

输出端相当于三极管的集电极，要得到高电平状态需要上拉电阻才行，适合于做电流型的驱动，其吸收电流的能力相对强。

开漏输出，写 1

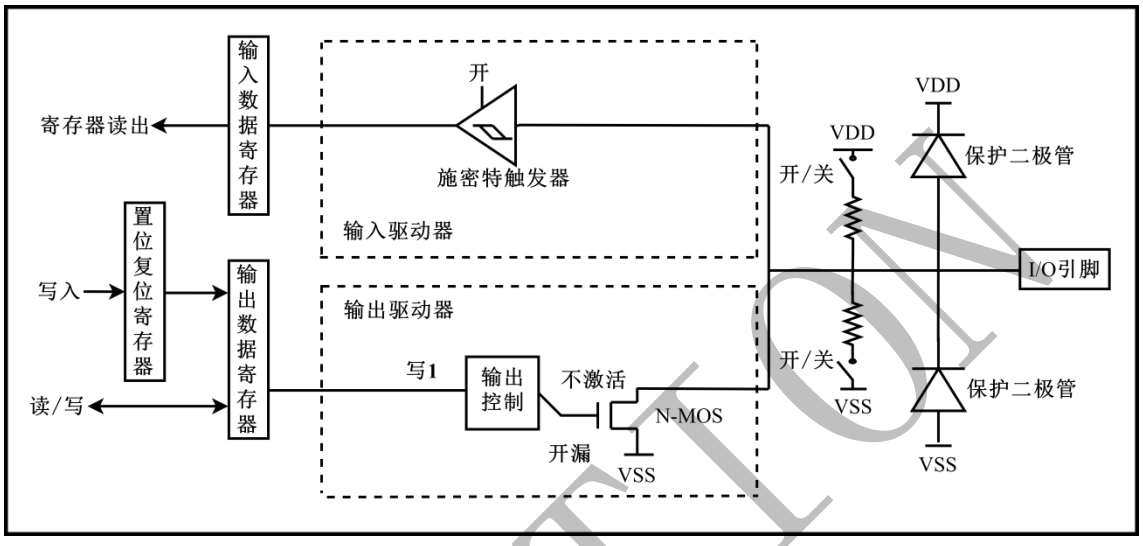


图 10-10　GPIO 开漏输出配置（写 1）

输出数据寄存器写入“1”，N-MOS 管不激活，引脚相当于浮空状态，引脚电平由外接电阻决定，接上拉电阻则为高电平，接下拉电阻相当于低电平。输出数据寄存器的写“1”不会使得引脚输出高电平 ，写“0”会使得引脚输出低电平。

开漏输出，写 0

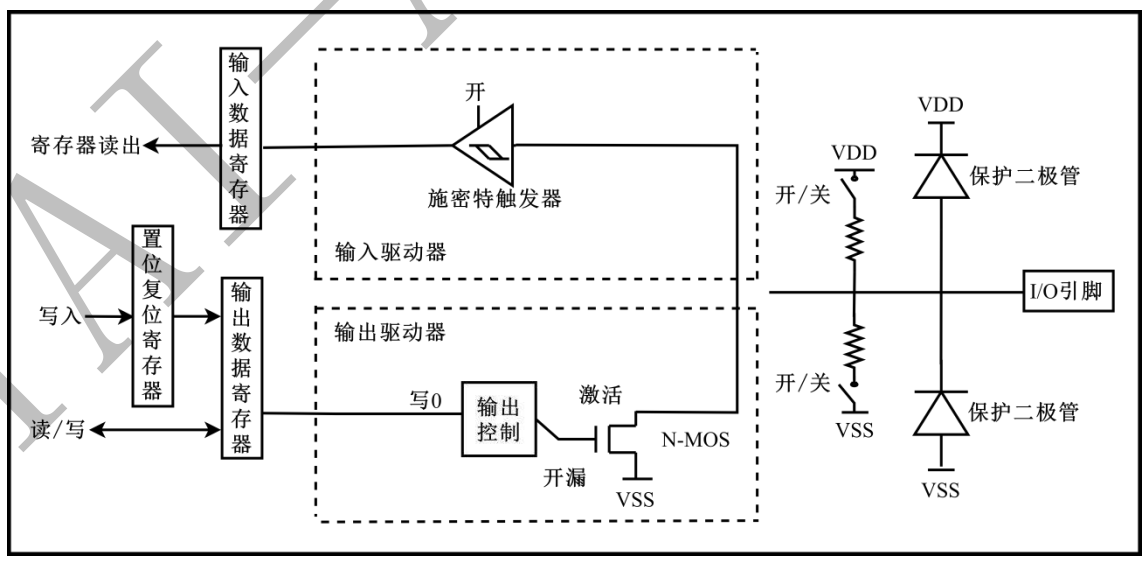


图 10-11　GPIO 开漏输出配置（写 0）

N-MOS 管激活，连接 VSS 接地，I/O 引脚输出低电平，输入数据寄存器也可以读到低电平。

10.4.11 复用功能配置

当 I/O 端口被配置为复用功能时：

- 可将输出配置为开漏或推挽式模式
- 输出缓冲器可由内置外设的信号驱动(复用功能输出)
- 根据配置 GPIOx_PUR/GPIOx_PDR 寄存器来决定 I/O 是否打开上拉和下拉电阻

复用推挽/开漏可以理解为 GPIO 口被用作第二功能时的配置情况，即非通用 I/O 口，GPIOx_MRn 寄存器允许用户把一些复用功能重新映射到不同的引脚。

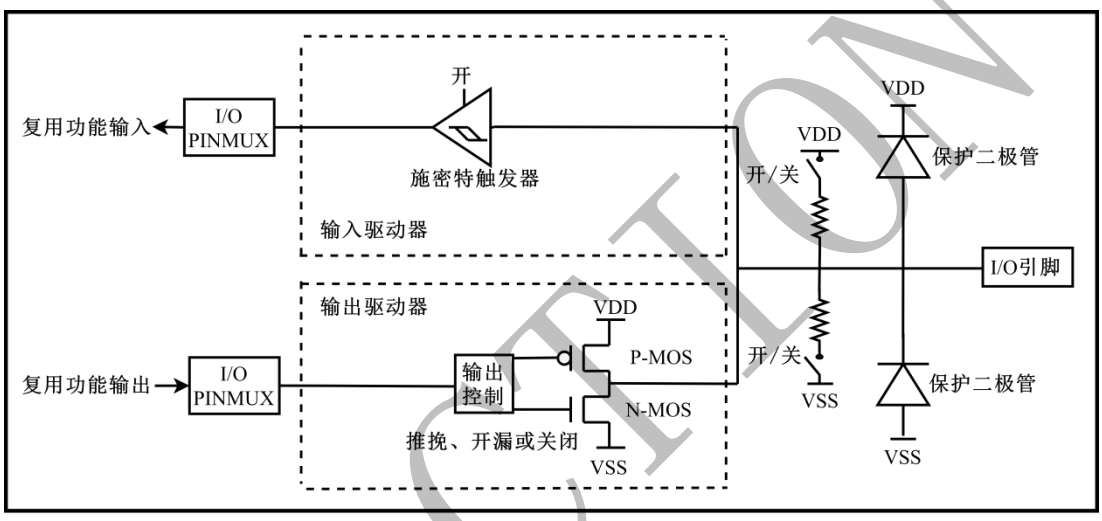


图 10-12 GPIO 复用功能配置

模拟输入配置

- 输出缓冲器被禁止
- 禁止施密特触发输入，实现了每个模拟 I/O 引脚上的零消耗。施密特触发输出值被强置为“0”
- 弱上拉和下拉电阻被禁止

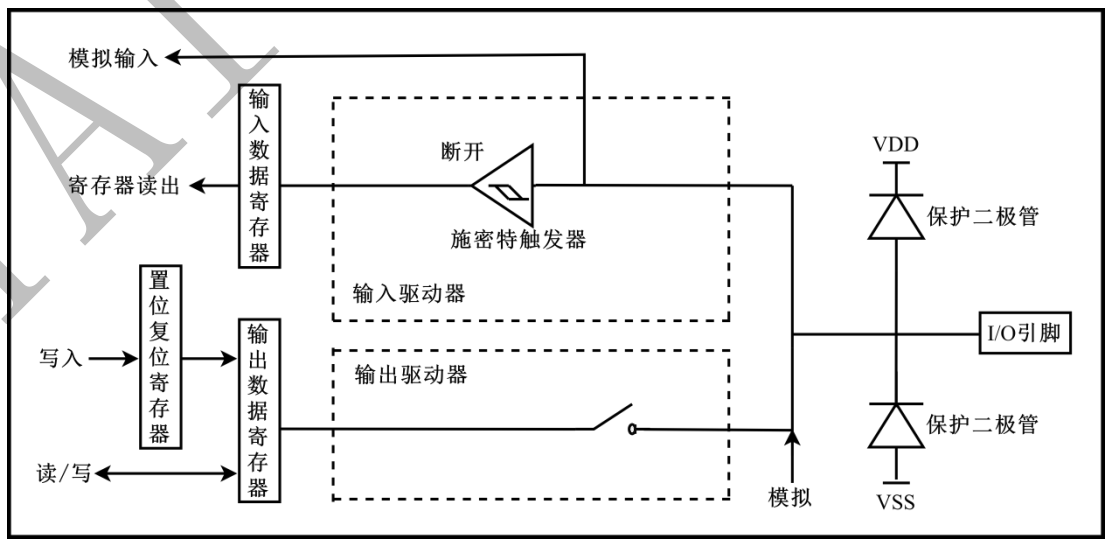


图 10-13 GPIO 模拟输入配置

注意：模拟输入，读到的不是高低电平，而是准确的电压值。

10.5 寄存器描述

10.5.1 寄存器列表

GPIOA 基地址：**0x4002 3000**
GPIOB 基地址：**0x4002 4000**
GPIOC 基地址：**0x4002 5000**
GPIOD 基地址：**0x4002 6000**

偏移	实例地址	名称	默认值	描述
0x00	0x40023000	GPIOx_BSR	0x00000000	GPIOx 端口置位/复位寄存器
0x04	0x40023004	GPIOx_IDR	0x00000000	GPIOx 输入数据寄存器
0x08	0x40023008	GPIOx_ODR	0x00000000	GPIOx 输出数据寄存器
0x10	0x40023010	GPIOx_IER	0x00000000	GPIOx 中断使能寄存器
0x14	0x40023014	GPIOx_ICR	0x00000000	GPIOx 中断模式寄存器
0x18	0x40023018	GPIOx_ISR	0x00000000	GPIOx 中断状态寄存器
0x20	0x40023020	GPIOx_MR0	0x77777777	GPIOx 功能复用寄存器 0
0x24	0x40023024	GPIOx_MR1	0x77777777	GPIOx 功能复用寄存器 1
0x28	0x40023028	GPIOx_SER	0x00000000	GPIOx 同步使能寄存器
0x2C	0x4002302C	GPIOx_DER	0x00000000	GPIOx 去抖使能寄存器
0x30	0x40023030	GPIOx_PUR	0x00000000	GPIOx 上拉模式寄存器
0x34	0x40023034	GPIOx_PDR	0x00000000	GPIOx 下拉状态寄存器
0x38	0x40023038	GPIOx_ODER	0x00000000	GPIOx 开漏使能寄存器
0x40	0x40023040	GPIOx_DHR	0x00000000	GPIOx 驱动寄存器
0x44	0x40023044	GPIOx_IHR	0x00000000	GPIOx 输入迟滞寄存器
0x48	0x40023048	GPIOx_OSR	0x00000000	GPIOx 输出速率配置寄存器

【说明】上表中，x=A、B、C、D。

10.5.2 寄存器描述

10.5.2.1 GPIOx 端口置位/复位寄存器（GPIOx_BSR）

- 名称：GPIOx Bit Set/Reset Register（x = A to D）
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称	BR[y]	BS[y]
访问	W1C	W1C
复位值	0x0	0x0

字段	说明
[31:16] BR[y]	**GPIO 端口 y 复位: (y=0~15)（GPIO PIN Reset）** Note：Bit Set/Reset 同时有效时，Set 优先级高；
[15:0] BS[y]	**GPIO 端口 y 置位: (y=0~15)（GPIO PIN Set）** Note：Bit Set/Reset 同时有效时，Set 优先级高；

10.5.2.2 GPIOx 输入数据寄存器（GPIOx_IDR）

- 名称：GPIOx Data Input Register（x = A to D）
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		DI[y]
访问		R
复位值		0x0

字段	说明
[15:0] DI[y]	**GPIO 端口 y 输入数据寄存器（y=0~15）（GPIO PIN Input Data）** Note: 当 Pinmux 配置为 AF0 输入模式时，将端口数据采集到寄存器内，可通过该寄存器来读取 I/O 上的数据输入；

10.5.2.3 GPIOx 输出数据寄存器（GPIOx_ODR）

- 名称：GPIOx Data Output Register（x = A to D）
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		DO[y]
访问		R/W
复位值		0x0

字段	说明
[15:0]	**GPIO 端口 y 输出数据寄存器（y=0~15）（GPIO PIN Output Data）**
DO[y]	Note：当 Pinmux 配置为 AF1 输出模式时，将数据寄存器内数据映射到 I/O 端口上输出；

10.5.2.4 GPIOx 中断使能寄存器（GPIOx_IER）

- 名称：GPIOx Interrupt Enable Register（x = A to D）
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		IE[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 中断使能位（y=0~15）（GPIO PIN Interrupt Enable）**
IE[y]	0：关闭
	1：启动

10.5.2.5 GPIOx 中断模式寄存器（GPIOx_IMR）

- **名称：GPIOx Interrupt Mode Register（x = A to D）**
- **偏移地址：0x14**
- **默认值：0x00000000**
- 返回寄存器列表

位	2y+1:2y
名称	IM[y]
访问	R/W
复位值	0x0

字段	说明
[2y+1:2y]	**GPIO 端口 y 中断模式选择位（y=0~15）（GPIO PIN Interrupt Mode Select）**
IM[y]	00：关闭　　01：上升沿
	10：下降沿　　11：上升沿/下降沿

10.5.2.6 GPIOx 中断状态寄存器（GPIOx_ISR）

- **名称：GPIOx Interrupt Status Register（x = A to D）**
- **偏移地址：0x18**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		PS[y]
访问		R/W1C
复位值		0x0

字段	说明
[15:0]	**GPIO 端口 y 中断状态位（y=0~15）（GPIO PIN Interrupt Status）**
PS[y]	0：No IRQ pending
	1：Clear IRQ pending
	Note：每一个 Bit 对应一个 IO 的 PENDING，通过写“1”清 0，写 0 无效

10.5.2.7 GPIOx 功能复用寄存器 0（GPIOx_MR0）

- 名称：GPIOx Pinmux Register0（x = A to D）
- 偏移地址：0x20
- 默认值：0x77777777
- 返回寄存器列表

位	4y+3:4y
名称	PM[y]
访问	R/W
复位值	0x7

字段	说明
[4y+3:4y] PM[y]	GPIO 端口 y 功能复用配置寄存器（GPIO0~GPIO7）：（y=0~7）（GPIO PIN Function Reuse Setting） 0000:AF0 …… 0111:AF7 1xxx:Reserve

10.5.2.8 GPIOx 功能复用寄存器 1（GPIOx_MR1）

- 名称：GPIOx Pinmux Register1（x = A to D）
- 偏移地址：0x24
- 默认值：0x77777777
- 返回寄存器列表

位	4y+3:4y
名称	PM[y]
访问	R/W
复位值	0x7

字段	说明
[4y+3:4y] PM[y]	GPIO 端口 y 功能复用配置寄存器（GPIO8~GPIO15）：（y=0~7）（GPIO PIN Function Reuse Setting） 0000:AF0 …… 0111:AF7 1xxx:Reserve

10.5.2.9 GPIOx 同步使能寄存器（GPIOx_SER）

- 名称：GPIOx Sync Enable Register（x = A to D）
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		SE[y]
访问		R/W
复位值		0x0

字段	说明
[15:0]	**GPIO 端口 y 同步使能位（y=0~15）（GPIO PIN Synchronization Enable）**
SE[y]	0：关闭
	1：使能输入同步功能

10.5.2.10 GPIOx 去抖使能寄存器（GPIOx_DER）

- 名称：GPIOx Debounce Enable Register（x = A to D）
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		DE[y]
访问		R/W
复位值		0x0

字段	说明
[15:0]	**GPIO 端口 y 去抖使能位（y=0~15）（GPIO PIN Debounce Enable）**
DE[y]	0：关闭
	1：使能输入去抖功能

10.5.2.11 GPIOx 上拉使能寄存器（GPIOx_PUR）

- 名称：GPIOx Pullup Enbale Register（x = A to D）
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		PU[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 上拉使能位（y=0~15）（GPIO PIN Pull Up Enable）**
PU[y]	0：无上拉
	1：上拉

10.5.2.12 GPIOx 下拉使能寄存器（GPIOx_PDR）

- 名称：GPIOx Pulldown Enable Register（x = A to D）
- 偏移地址：0x34
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		PD[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 下拉使能位（y=0~15）（GPIO PIN Pull Down Enable）**
PD[y]	0：无下拉
	1：下拉

10.5.2.13 GPIOx 开漏使能寄存器（GPIOx_ODER）

- 名称：GPIOx Open Drain Enable Register（x = A to D）
- 偏移地址：0x38
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		OD[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 开漏使能（y=0~15）（GPIO PIN Open Drain Enable）**
OD[y]	0：推挽输出
	1：开漏输出

10.5.2.14 GPIOx 驱动寄存器（GPIOx_DHR）

- 名称：GPIOx Driver Register（x = A to D）
- 偏移地址：0x40
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		DR[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 驱动能力配置（y=0~15）（GPIO PIN Drive Capability Setting）**
DR[y]	0：8mA
	1：24mA

10.5.2.15 GPIOx 输入迟滞寄存器（GPIOx_IHR）

- 名称：GPIOx Hysteresis Register（x = A to D）
- 偏移地址：0x44
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		IH[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 输入迟滞配置（y=0~15）（GPIO PIN Input Hysteresis Setting）**
IH[y]	0：关闭
	1：使能输入迟滞

10.5.2.16 GPIOx 输出速率配置寄存器（GPIOx_OSR）

- 名称：GPIOx Output Slew Register（x = A to D）
- 偏移地址：0x48
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15: 0
名称		OS[y]
访问		R/W
复位值		0x0

字段	说明
[15: 0]	**GPIO 端口 y 输出速率配置（y=0~15）（GPIO PIN Output Slew Setting）**
OS[y]	0：正常压摆率
	1：增强压摆率
	注意：增强压摆率可以提高 IO 口输出速率，但相应会带来高的 EMI 干扰。

11 基础定时器（TMR0~3）

11.1 简介

芯片内置 4 套基础定时器(TMR0~3)，每个定时器内包含一个 32 位自动重载计数器，计数器由可编程的 16 位预分频器驱动，并都包含 16 位起始/终止寄存器，用于配置计数器的计数起始/终止值。

TMR0/1/2/3 四个定时器为一组，具有独立可配的寄存器同步功能，详情请参考“11.4.3 定时器同步”章节。

11.2 结构框图

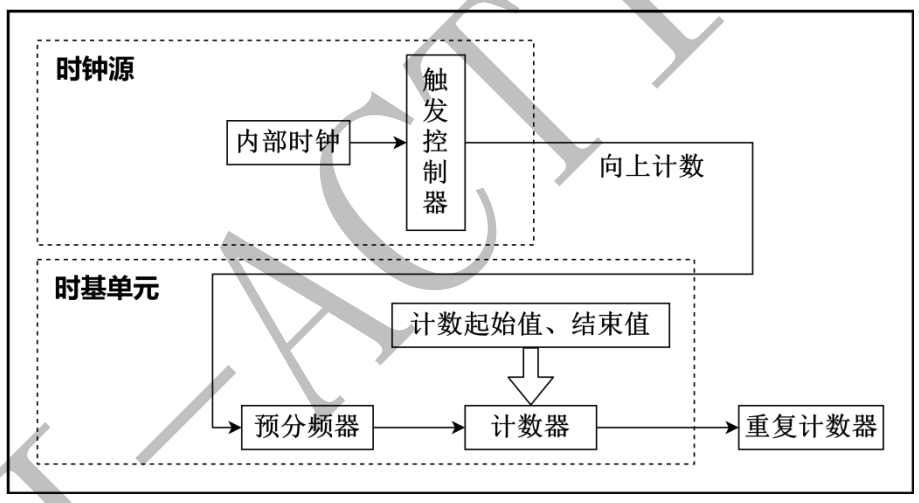


图 11-1 TMR 结构框图

11.3 主要特性

基础定时器 TMRx(x = 0 ... 3)具有以下特性：

- APB0CLK 作为时钟源，内置 16 位的预分频器，用于对计数器的时钟源进行预分频。
- 32 位的自动重载向上计数器。
- 16 位的计数起始值和终止值寄存器，用于灵活设定计数器的开始计数值和终止计数值。
- 两种工作模式：
  - 循环模式，计数完成后自动重载并继续计数
  - 单次模式，计数完成后自动关闭定时器的使能
- 多组定时器同步功能

11.4 功能描述

11.4.1 时基单元

基础定时器的主要模块由一个 32 位计数器及其相关的自动重载寄存器组成，计数器采用递增方式计数。计数器的时钟可通过预分频器进行分频。

通用定时器的自动重载计数器(TMRx_CNTR)、计数起始值寄存器(TMRx_CSVR)、计数终止值寄存器(TMRx_CEVR)和预分频寄存器(TMRx_PSCR)可通过软件进行读写，即使在计数器运行中也可执行读写操作。

时基单元包含：

- 自动重载计数器(TMRx_CNTR)
- 预分频寄存器(TMRx_PSCR)
- 计数起始值寄存器(TMRx_CSVR)
- 计数终止值寄存器(TMRx_CEVR)

其中预分频寄存器(TMRx_PSCR)、计数起始值寄存器(TMRx_CSVR)及计数终止值寄存器(TMRx_CEVR)都是预装载的。对预装载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器立即生效，也可以在每次发生更新事件时更新到影子寄存器，这取决于定时器控制寄存器(TMRx_CR)中的自动重载预装载使能位(ARPE)。

定时器计数器达到上溢值，将产生更新事件。

更新事件相关的详细介绍，请参看“11.4.5.1 更新事件”章节。

定时器的计数器由时钟源经过预分频器分频后提供的时钟驱动，且仅当控制寄存器(TMRx_CR)中的计数器使能位(CEN)置 1 时，才会启动计数器计数。

注意：计数器将在控制寄存器(TMRx_CR)中的计数器使能位(CEN)置 1 时刻后的一个时钟周期开始计数。

预分频器说明

预分频器用于对计数器的计数时钟频率进行分频，通过配置预分频器寄存器(TMRx_PSCR)设定预分频的分频系数。计数器计数频率公式如下：

$$f_{CLK\_CNT} = f_{CLK\_SRC} / (PSC + 1)$$

该寄存器具有预加载缓存功能，因此可以对预分频器实现实时修改，而新的预分频比将在下一个更新事件发生时被采用。

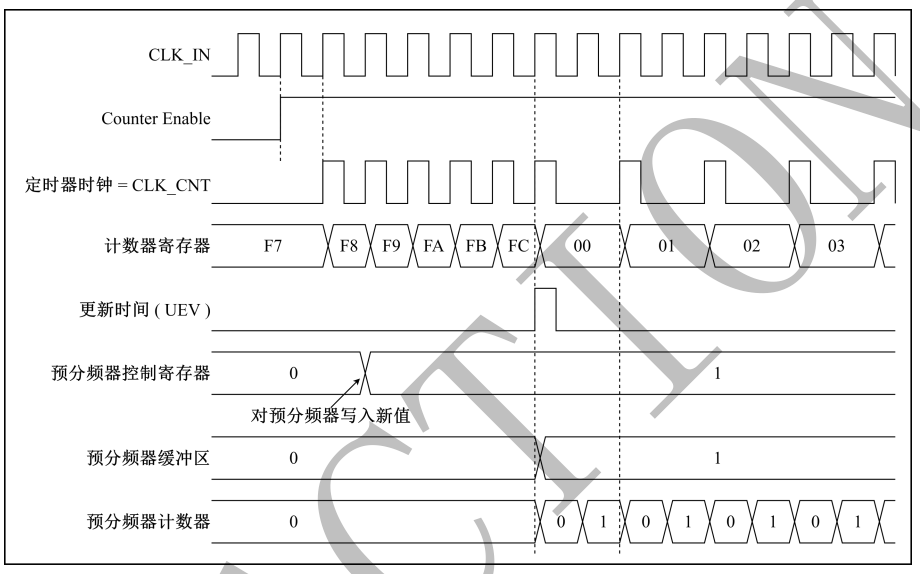


图 11-2　实时修改预分频器的分频值从 1 变为 2 的时序图

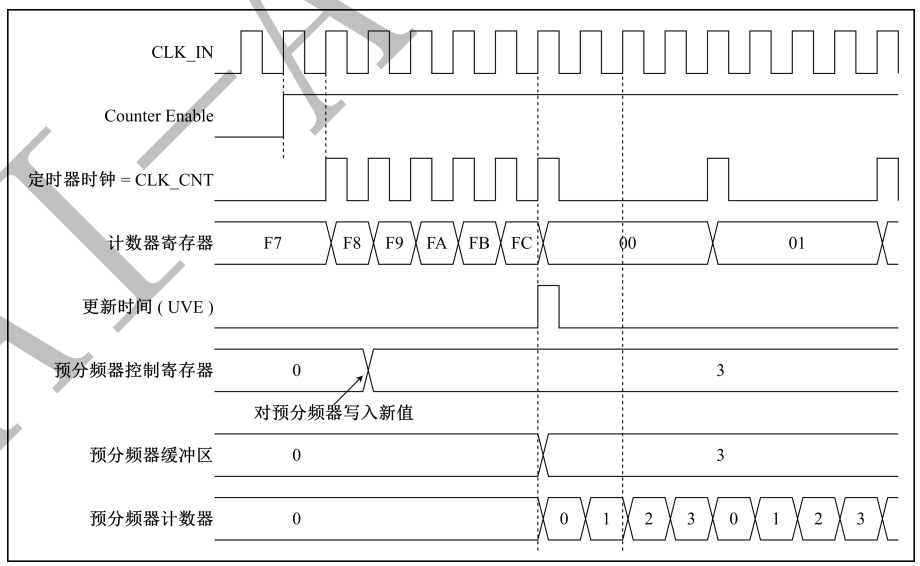


图 11-3　实时修改预分频器的分频值从 1 变为 4 的时序图

11.4.2 计数器模式

通用定时器仅支持向上计数模式，计数器从起始值(TMRx_CSVR)计数到结束值(TMRx_CEVR-1)，支持单次和连续计数模式：单次模式下，计数完成后将自动关闭计数器使能(CEN 自动置 0)停止计数；连续模式下，每次计数完成后，自动从起始值(TMRx_CSVR)开始重新计数，并生成计数器更新事件 UEV。

定时器发生更新事件(UEV)时，将更新相关寄存器的值且将更新中断标志位(UIF)置 1：

- 自动重载影子寄存器将更新为预装载寄存器(TMRx_CEVR)的值
- 预分频器影子寄存器将更新为预装载寄存器(TMRx_PSCR)的值
- 捕获/比较寄存器更新为预装载寄存器(TMRx_CR)的值(非通过软件设置 UG 位更新事件时)

以下图 11-4 ~图 11-9 将通过一些示例说明当计数器等于 0x36 时，不同时钟频率下表现的行为。

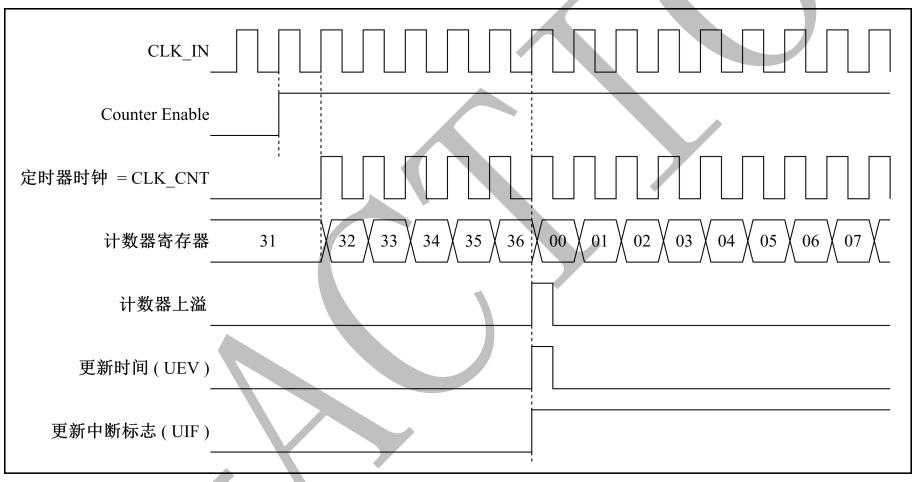


图 11-4　计数器时序图，1 分频内部时钟

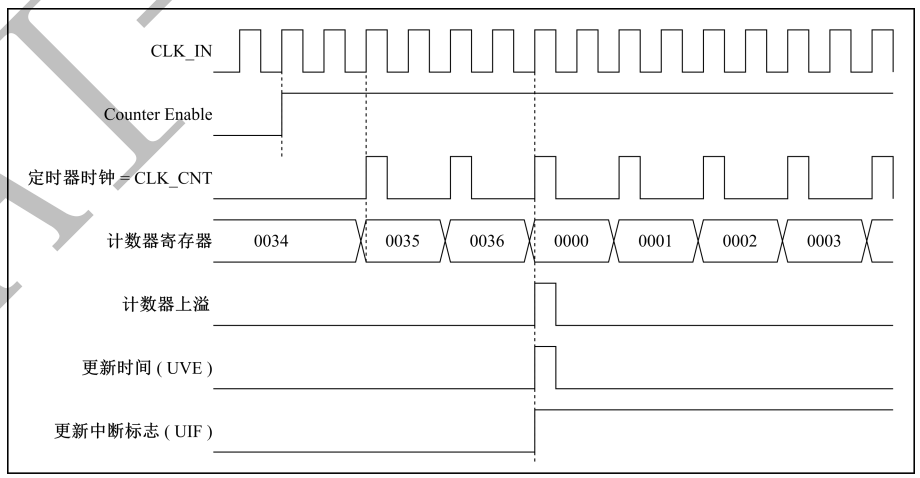


图 11-5　计数器时序图，2 分频内部时钟

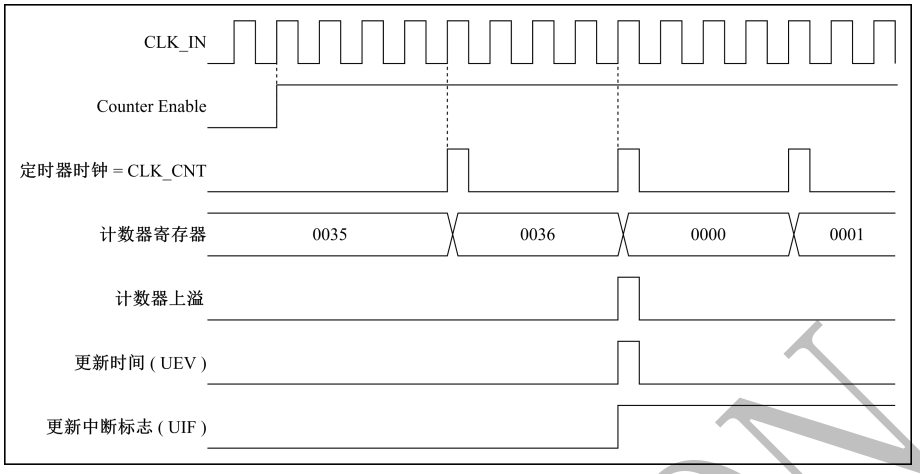


图 11-6　计数器时序图，4 分频内部时钟

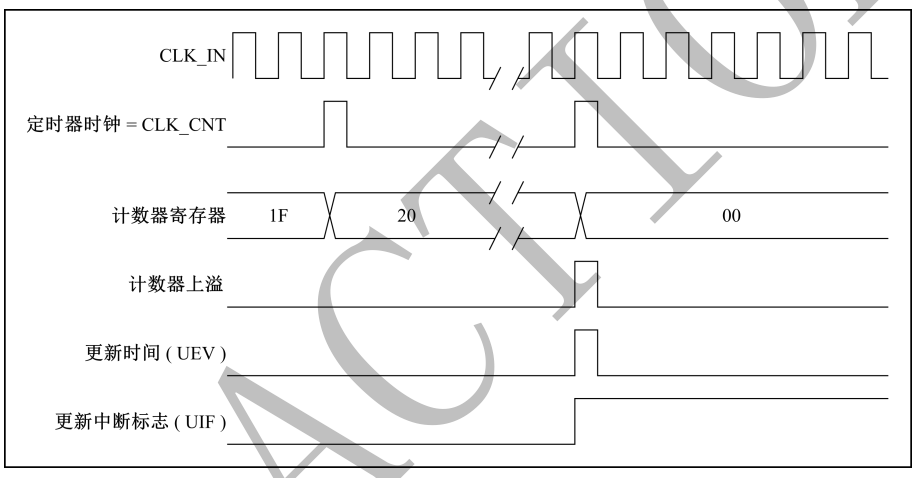


图 11-7　计数器时序图，N 分频内部时钟

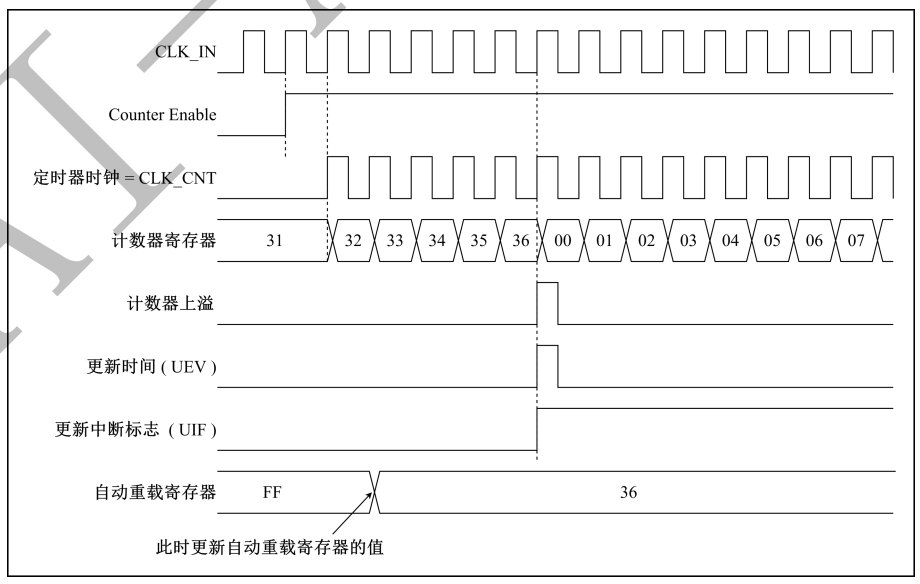


图 11-8　计数器时序图，运行中更新重载值（自动重载使能关闭-实时生效）

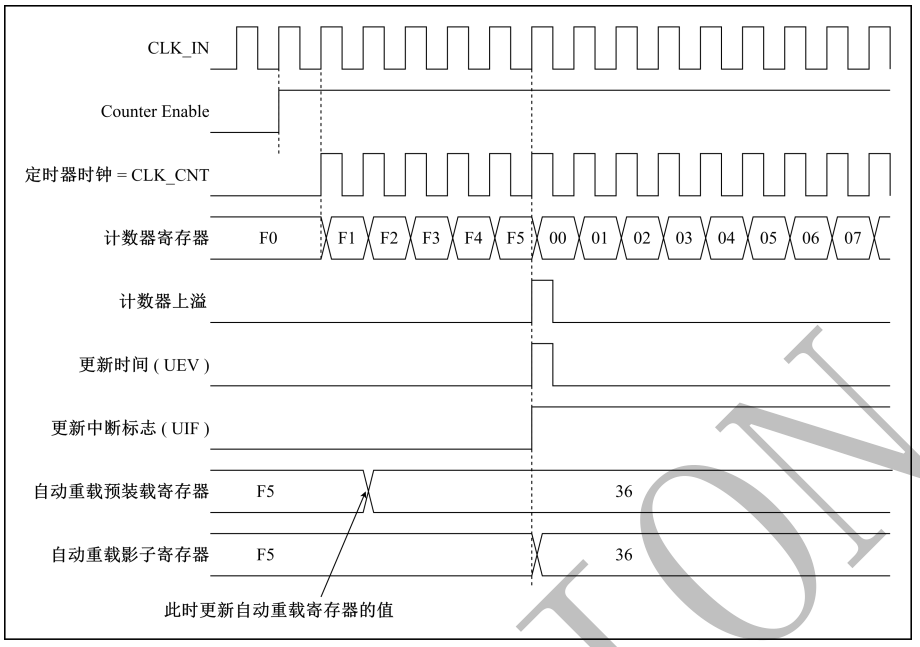


图 11-9　计数器时序图，运行中更新重载值（自动重载使能使能-更新事件生效）

11.4.3　定时器同步

每个定时器都是完全独立的，定时器之间没有共享任何资源。

为了实现定时器同步启动功能，4 套基础定时器之间增加同步启动的功能。

通过定时器组同步控制寄存器(TMRG_SYNCR)，实现同组定时器的同步启动/关闭功能。

11.4.4　调试模式

当内核进入调试模式（Cortex™-M3 内核停止）时，定时器内部计数器会根据 SYSCTRL 模块中系统控制寄存器的 TMR0 调试使能位(SYSCTRL_SYSCR.TMR0DE)来选择继续正常工作或者停止工作。

11.4.5　事件与中断

基础定时器在计数完成时产生更新事件，事件主要用于更新相应的影子寄存器以及产生相应的中断。

基础定时器内只有以下一种事件：

- 定时器上溢事件，简称上溢事件 OVEV(Overflow Event)

中断包含以下一种中断：

- 定时器上溢中断，发生上溢事件后，上溢中断标志位(OVIF)同时被置位

11.4.5.1 更新事件（Update Event）

更新事件(UEV)由以下情况产生：

- 计数器寄存器(TMRx_CNTR)计数值到达计数终止值寄存器(TMRx_CEVR)设定的值。

更新事件的作用

计数器到达上溢后，计数器寄存器会立即重新加载计数值起始寄存器(TMRx_CSVR)的值，重新开始计数(如果为连续模式)。

事件产生后，中断状态寄存器(TMRx_ISR)中的上溢中断标志位(OVIF)将会置 1，并且会触发定时器中断(如果 OVIE = 1)。

11.4.5.2 中断

中断标志位与中断使能之间的关系如下表描述所示，详细内容请参考“11.4.5 事件与中断”章节描述：

表 11-1 中断标志位与中断使能之间的关系

中断名称	中断使能	中断标志	清除方式	备注
上溢中断	OVIE	OVIF	W1C	计数器计数上溢后产生

11.5 寄存器描述

11.5.1 寄存器列表

TMRx 基地址：**0x4001 4000 + 0x100*x [x=0~3]**
TMRG 基地址：**0x4001 4400**

偏移	实例地址	名称	默认值	描述
0x00	0x40014000	TMRx_CR	0x00000000	TMRx 控制寄存器
0x10	0x40014010	TMRx_ISR	0x00000000	TMRx 中断状态寄存器
0x20	0x40014020	TMRx_CSVR	0x00000000	TMRx 计数起始值寄存器
0x24	0x40014024	TMRx_CEVR	0xFFFFFFFF	TMRx 计数终止值寄存器
0x2C	0x4001402C	TMRx_PSCR	0x00000000	TMRx 预分频寄存器
0x30	0x40014030	TMRx_CNTR	0x00000000	TMRx 计数寄存器
0x00	0x40014400	TMRxG_SYNCR	0x00000000	定时器组同步寄存器

【说明】上表中，x=0~3。

11.5.2 寄存器描述

11.5.2.1 TMRx 控制寄存器（TMRx_CR）

- 名称：TMRx Control Register（x = 0~3）
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:9	8	7:5	4	3	2:1	0
名称		OVIE		ARPE	MS		CEN
访问		R/W		R/W	R/W		R/W
复位值		0x0		0x0	0x0		0x0

字段	说明
[8] OVIE	**计数器上溢中断使能控制（Timer Counter Overflow Interrupt Enable）** 1：使能中断 0：禁止中断
[4] ARPE	**定时器自动重载预装载使能（Timer Auto-Reload Preload Enable）** 此位用于开启/关闭影子寄存器预加载功能。 0：寄存器不进行预加载 1：寄存器进行预加载 注意： 1. 预加载功能涉及的寄存器有： ➢ 计数终止值寄存器（TMRx_CEVR） ➢ 预分频寄存器（TMRx_PSCR） 但他们的更新事件分别由不同的情况产生。 2. 如果不开启预加载功能，则相关寄存器写入新值将立刻生效 3. 如果开启预加载功能，则相关寄存器写入新值后，在下一个更新事件（UEV）到来后生效。 更详细信息请参看“11.4.5.1 更新事件（Update Event）”章节。
[3] MS	**定时器工作模式选择（Timer Mode Selection）** 0：连续模式 1：单次模式 注意：配置为单次模式后，计数器溢出后，将自动停止计数。
[0] CEN	**计数器使能位（Timer Counter Enable）** 1：开始计数器 0：关闭计数器

11.5.2.2 TMRx 中断状态寄存器（TMRx_ISR）

- 名称：TMRx Interrupt Status Register（x = 0~3）
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:5	4	3:0
名称		OVIF	
访问		R/W1C	
复位值		0x0	

字段	说明
[4]	**计数器上溢中断标志位（Timer Counter Overflow Interrupt Flag）**
OVIF	0：未发生上溢
	1：计数器上溢

11.5.2.3 TMRx 计数起始值寄存器（TMRx_CSVR）

- 名称：TMRx Counter Start Value Register（x = 0~3）
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CSVR
访问		R/W
复位值		0x0

字段	说明
[15:0]	**计数起始值（Timer Counter Start Value）**
CSVR	计数起始值寄存器，通过配置此寄存器设置定时器开始计数的值。

11.5.2.4 TMRx 计数终止值寄存器（TMRx_CEVR）

- **名称：TMRx Counter End Value Register（x = 0~3）**
- **偏移地址：0x24**
- **默认值：0xFFFFFFFF**
- 返回寄存器列表

位	31:16	15:0	
名称		CEVR	
访问		R/W	
复位值		0xFFFFFFFF	

字段	说明
[15:0] CEVR	**计数终止值（Timer Counter End Value）** 通过配置此寄存器设置定时器结束计数的值。此外，该寄存器支持自动预装载（Auto-Reload）功能： ● **不使能自动预装载功能（ARPE=0）** 对该寄存器的设置将立即作用到内部影子寄存器内，本次计数周期内即生效； ● **使能自动预装载功能（ARPE=1）** 对该寄存器的设置将在更新事件（UEV）产生后，再更新至内部影子寄存器内生效。 更详细介绍请参考“11.4.1 时基单元”和“11.4.5.1 更新事件（Update Event）”章节。

11.5.2.5 TMRx 预分频寄存器（TMRx_PSCR）

- 名称：TMRx Prescaler Register（x = 0~3）
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		PSC
访问		R/W
复位值		0x0

字段	说明
[15:0] PSC	**定时器预分频值（Timer Prescaler value）** 定时器计数器的计数频率（CK_CNT）通过此预分频器对时钟源的频率（CK_SRC）进行分频得到： $f_{CK\_CNT} = f_{CK\_SRC} /（PSC + 1）$ 此外，该寄存器支持自动预装载（Auto-Reload）功能： ● **不使能自动预装载功能（ARPE=0）** 对该寄存器的设置将立即作用到内部影子寄存器内，本次计数周期内即生效； ● **使能自动预装载功能（ARPE=1）** 对该寄存器的设置将在更新事件（UEV）产生后，再更新至内部影子寄存器内生效。 更详细介绍请参考“11.4.1 时基单元”和“11.4.5.1 更新事件（Update Event）”章节。

11.5.2.6 TMRx 计数寄存器（TMRx_CNTR）

- 名称：TMRx Counter Register（x = 0~3）
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CNT
访问		R/W
复位值		0x0

字段	说明
[15:0] CNT	**定时器计数值（Timer Counter Value）** 通过该寄存器可对定时器计数器的值进行读取/写入。需注意，因为系统与外设之间存在一定的总线延迟，对该寄存器的读取/写入操作可能存在一定的偏差。

11.5.2.7 定时器组同步寄存器（TMRx_SYNCR）

- 名称：Timer Group Sync Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7	6	5	4	3	2	1	0
名称		SYNC3DIS	SYNC2DIS	SYNC1DIS	SYNC0DIS	SYNC3EN	SYNC2EN	SYNC1EN	SYNC0EN
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] SYNC3DIS	**定时器 TMR3 的同步暂停位（Timer3 Counter Disable）** 1：暂停计数器 0：无效
[6] SYNC2DIS	**定时器 TMR2 的同步暂停位（Timer2 Counter Disable）** 1：暂停计数器 0：无效
[5] SYNC1DIS	**定时器 TMR1 的同步暂停位（Timer1 Counter Disable）** 1：暂停计数器 0：无效
[4] SYNC0DIS	**定时器 TMR0 的同步暂停位（Timer0 Counter Disable）** 1：暂停计数器 0：无效
[3] SYNC3EN	**定时器 TMR3 的同步使能位（Timer3 Counter Enable）** 1：开始计数器 0：关闭计数器
[2] SYNC2EN	**定时器 TMR2 的同步使能位（Timer2 Counter Enable）** 1：开始计数器 0：关闭计数器
[1] SYNC1EN	**定时器 TMR1 的同步使能位（Timer1 Counter Enable）** 1：开始计数器 0：关闭计数器
[0] SYNC0EN	**定时器 TMR0 的同步使能位（Timer0 Counter Enable）** 1：开始计数器 0：关闭计数器

12 通用定时器（TMR4~7）

12.1 简介

芯片内置 4 套通用定时器(TMR4~7)，每个定时器内包含一个 32 位自动重载计数器，计数器由可编程的 32 位预分频器驱动，并都包含 32 位起始/终止寄存器，用于配置计数器的计数起始/终止值。

通用定时器可用于多种用途，包括最基本的定时功能(基础计时)以及测量输入信号的脉冲宽度(输入捕获)、生成 PWM 输出波形(输出比较)、外部脉冲计数(ETR)等多种灵活配置的功能。

TMR4/5/6/7 四个定时器为一组，具有独立可配的寄存器同步功能。详情请参考“12.4.7 定时器同步”章节。

12.2 结构框图

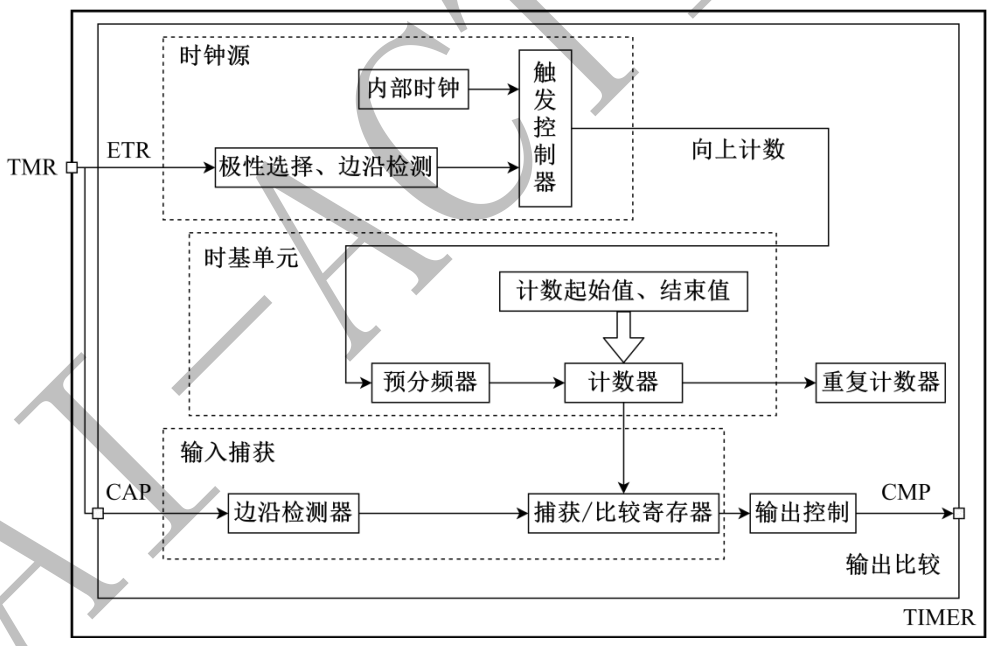


图 12-1 TMR 结构框图

12.3 主要特性

通用定时器 TMRx(x = 4~7)具有以下特性：

- 32 位的自动重载向上计数器。
- 32 位的预分频器，用于对计数器的时钟源进行预分频。
- 以 AHB0CLK 作为时钟源，32 位的计数起始值和终止值寄存器，用于灵活设定计数器的开始计数值和终止计数值。
- 两种工作模式：
  - 循环模式，计数完成后自动重载并继续计数
  - 单次模式，计数完成后自动关闭定时器的使能
- 独立通道，可用于：
  - 输入捕获
  - 输出比较
  - PWM 生成
  - ETR 外部触发输入
- 多组定时器同步功能
- 发生如下事件时产生中断：
  - 计数器上溢(Overflow Event)
  - 更新事件(Update Event) (需使能)：计数器上溢产生更新事件，也可软件产生(UG)
  - 输入捕获(Capture Event)、重复捕获(Over Capture Event)
  - 输出比较(Compare Event)

12.4 功能描述

12.4.1 时基单元

通用定时器的主要模块由一个 32 位计数器及其相关的自动重载寄存器组成，计数器采用递增方式计数。计数器的时钟可通过预分频器进行分频。

通用定时器的自动重载计数器(TMRx_CNTR)、计数起始值寄存器(TMRx_CSVR)、计数终止值寄存器(TMRx_CEVR)和预分频寄存器(TMRx_PSCR)可通过软件进行读写，即使在计数器运行中也可执行读写操作。

时基单元包含：

- 自动重载计数器(TMRx_CNTR)
- 预分频寄存器(TMRx_PSCR)
- 计数起始值寄存器(TMRx_CSVR)
- 计数终止值寄存器(TMRx_CEVR)

其中预分频寄存器(TMRx_PSCR)、计数起始值寄存器(TMRx_CSVR)及计数终止值寄存器(TMRx_CEVR)都是预装载的。对预装载寄存器执行写入或读取操作时会访问预装载寄存器。预装载寄存器的内容既可以直接传送到影子寄存器立即生效，也可以在每次发生更新事件时更新到影子寄存器，这取决于定时器控制寄存器自动重载预装载使能位(TMRx_CR.ARPE)。

定时器计数器达到上溢值，且控制寄存器(TMRx_CR.UDIS)为 0 时，将产生更新事件。此外，该更新事件也可通过软件发起，通过对定时器事件生成寄存器(TMRx_EGR.UG)位置 1。

更新事件相关的详细介绍，请参看“12.4.9.2 更新事件”章节。

定时器的计数器由时钟源经过预分频器分频后提供的时钟驱动，且仅当控制寄存器计数器使能位(TMRx_CR.CEN)置 1 时，才会启动计数器计数。

注意：控制寄存器计数器使能位(TMRx_CR.CEN)置 1 时刻后的一个时钟周期开始计数。

预分频器说明

预分频器用于对计数器的计数时钟频率进行分频，通过配置预分频器寄存器(TMRx_PSCR)设定预分频的分频系数。计数器计数频率公式如下：

$$f_{CLK\_CNT} = f_{CLK\_SRC} / (PSC + 1)$$

该寄存器具有预加载缓存功能，因此可以对预分频器实现实时修改，而新的预分频比将在下一个更新事件发生时被采用。也可通过软件设置 UG 位，立即产生一个更新事件（UEV）。

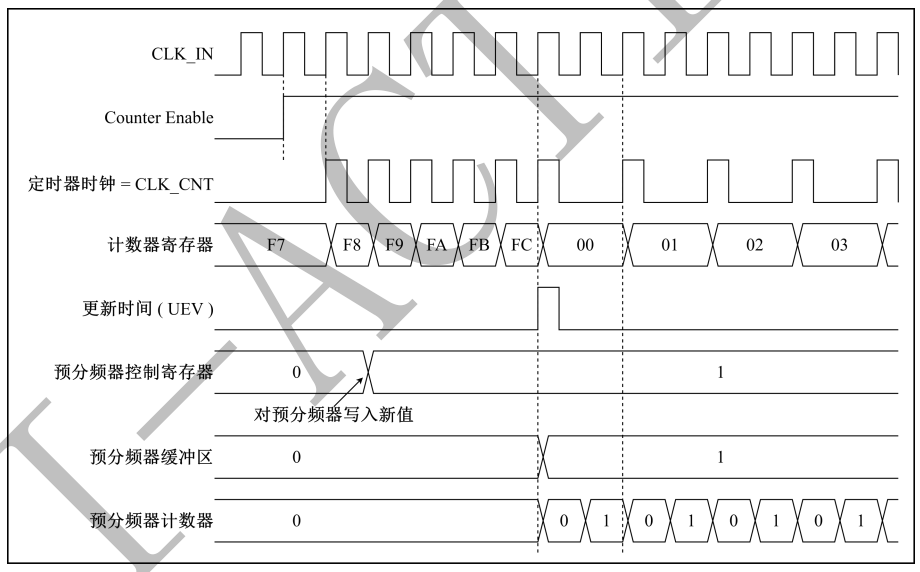


图 12-2　实时修改预分频器的分频值从 1 变为 2 的时序图

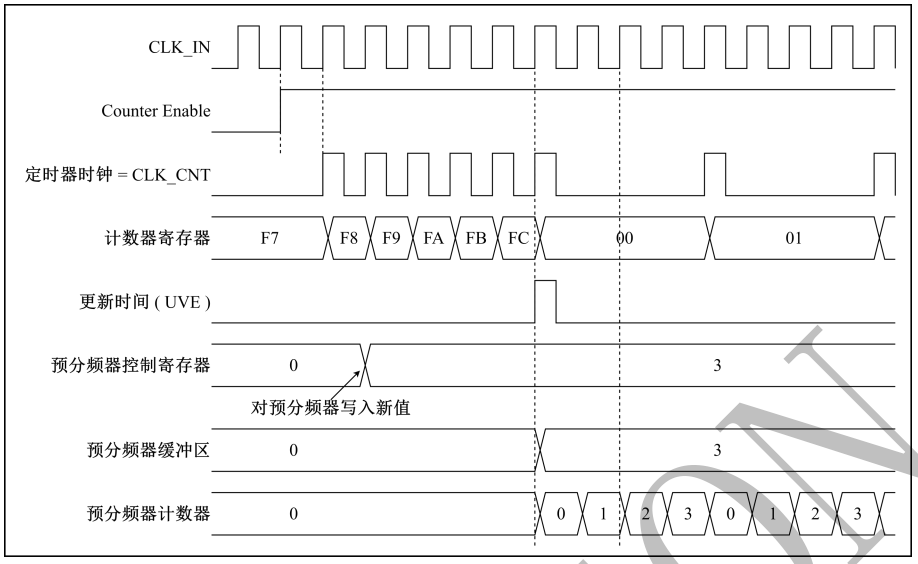


图 12-3　实时修改预分频器的分频值从 1 变为 4 的时序图

12.4.2　计数器模式

通用定时器仅支持向上计数模式，计数器从起始值(TMRx_CSVR)计数到结束值(TMRx_CEVR-1)，支持单次和连续计数模式：单次模式下，计数完成后将自动关闭计数器使能(CEN 自动置 0)停止计数；连续模式下，每次计数完成后，自动从起始值(TMRx_CSVR)开始重新计数，并生成计数器上溢事件 OVEV 和更新事件 UEV(如果使能更新事件)。

通过设置定时器控制寄存器(TMRx_CR)中的禁止更新位(UDIS)置 1 可禁止更新事件 UEV，禁止之后将不会产生任何的更新事件，直到禁止更新位重新置 0。这样可避免向预装载寄存器写入新值时更新影子寄存器，不过计数仍然能够在上溢事件 OVEV 后，重新从起始值开始重新计数。

此外，如果定时器控制寄存器(TMRx_CR)中更新请求源位(URS)置 1，禁止更新位(UDIS)置 0，然后通过软件将定时器事件产生寄存器(TMRx_EGR)中的更新产生位(UG)置 1，则只会生成更新事件(UEV)，但是不会将更新中断标志位(UIF)置 1，因此不会发起任何中断。

定时器发生更新事件(UEV)时，将更新相关寄存器的值且将更新中断标志位(UIF)置 1(这同时取决于 URS 位)：

- 自动重载影子寄存器将更新为预装载寄存器(TMRx_CEVR)的值
- 预分频器影子寄存器将更新为预装载寄存器(TMRx_PSCR)的值
- 捕获/比较寄存器更新为预装载寄存器(TMRx_CCR)的值(非通过软件设置 UG 位更新事件时)

以下图 12-4 ~图 12-9 将通过一些示例说明当计数器等于 0x36 时，不同时钟频率下表现的行为。

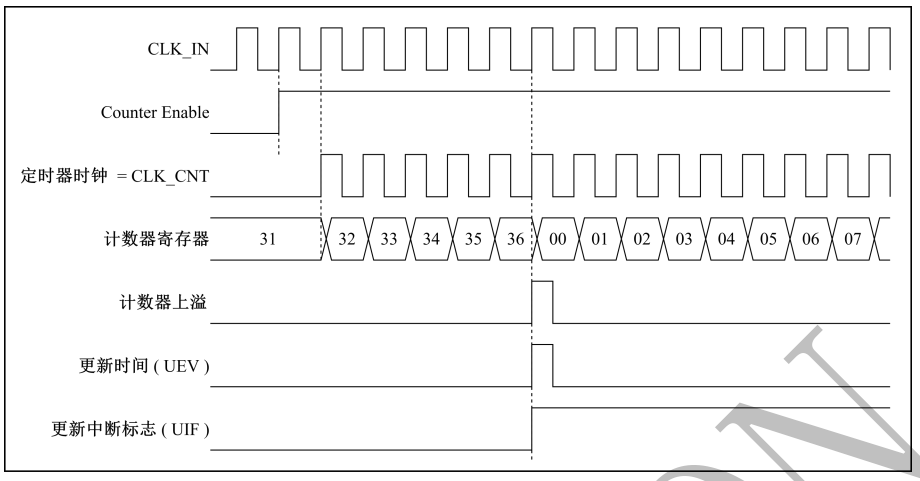


图 12-4　计数器时序图，1 分频内部时钟

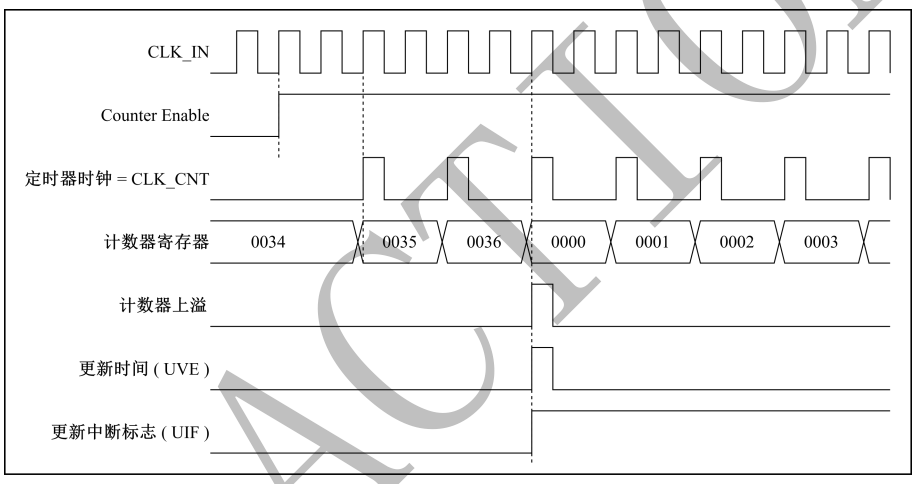


图 12-5　计数器时序图，2 分频内部时钟

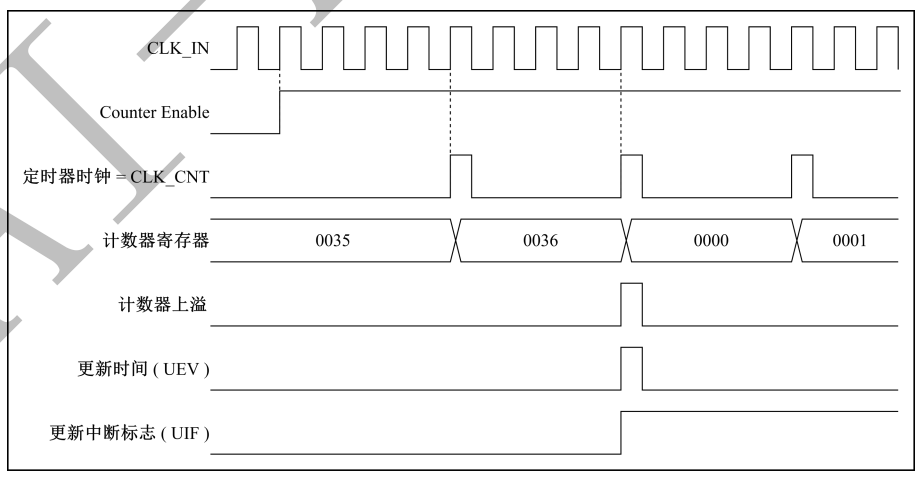


图 12-6　计数器时序图，4 分频内部时钟

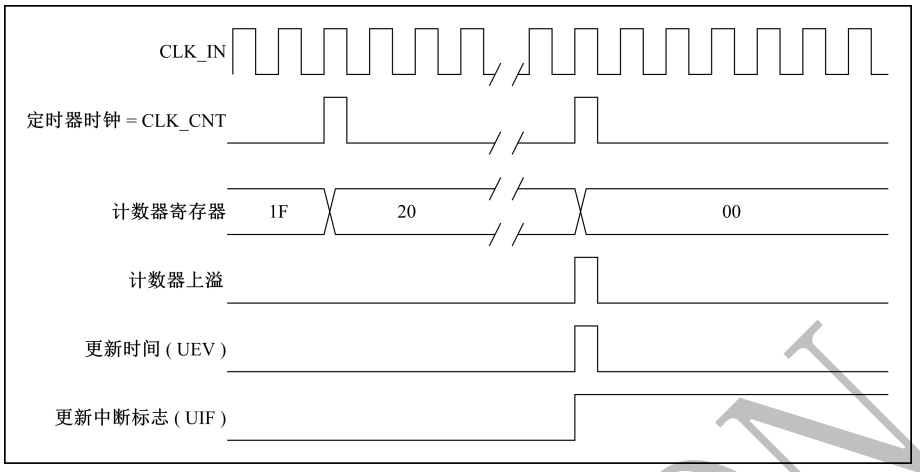


图 12-7　计数器时序图，N 分频内部时钟

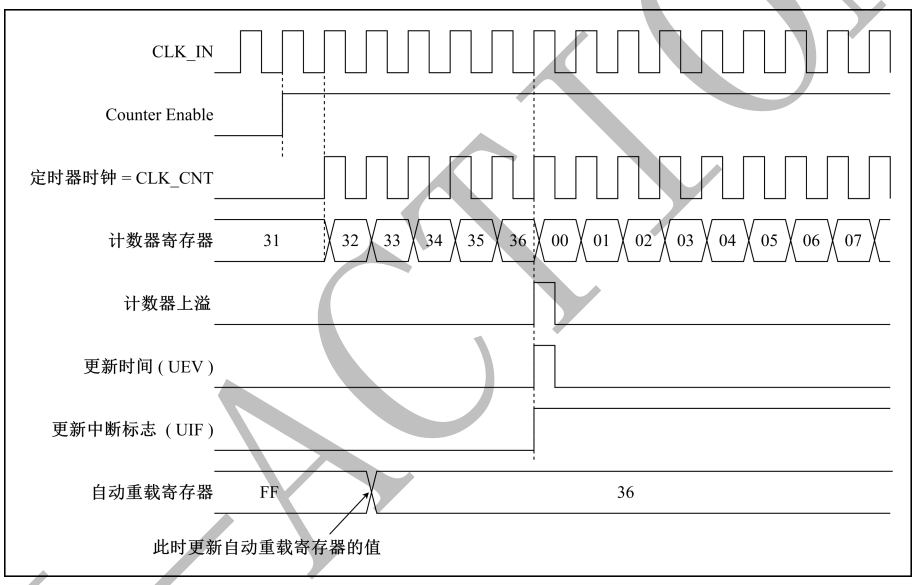


图 12-8　计数器时序图，运行中更新重载值（自动重载使能关闭-实时生效）

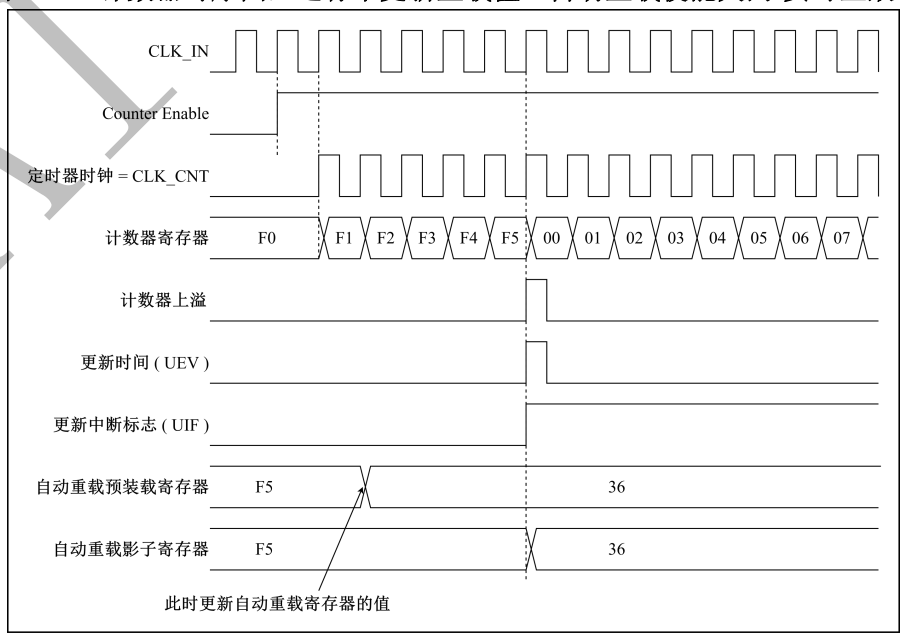


图 12-9　计数器时序图，运行中更新重载值（自动重载使能使能-更新事件生效）

12.4.3 时钟选择

定时器的计数器时钟源分为以下两类：

- 内部时钟源（CLK_IN）：系统时钟上升沿
- 外部时钟（ETR）：通过外部引脚输入时钟，并可配置触发边沿（上升/下降/双边沿）

内部时钟源（CLK_IN）

下图为正常模式下内部控制逻辑及计数器计数的行为（预分频设置为 0，即在没有预分频情况下）。

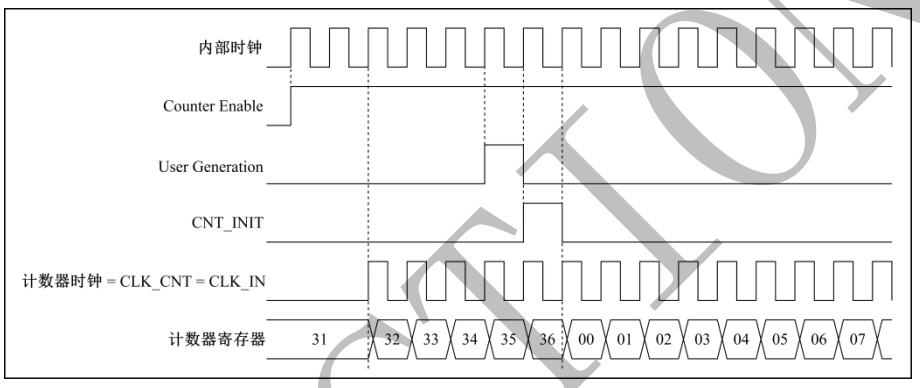


图 12-10　正常模式下的控制时序图（没有预分频情况）

外部时钟源（ETR）

定时器可选取外部时钟源作为计数器的时钟源。外部时钟源由定时器外设的功能引脚灌入，通过内部边沿检测电路触发计数器计数，可利用外部时钟源分为三种边沿，作为计数器时钟的触发：

- 上升沿计数
- 下降沿计数
- 双边沿计数

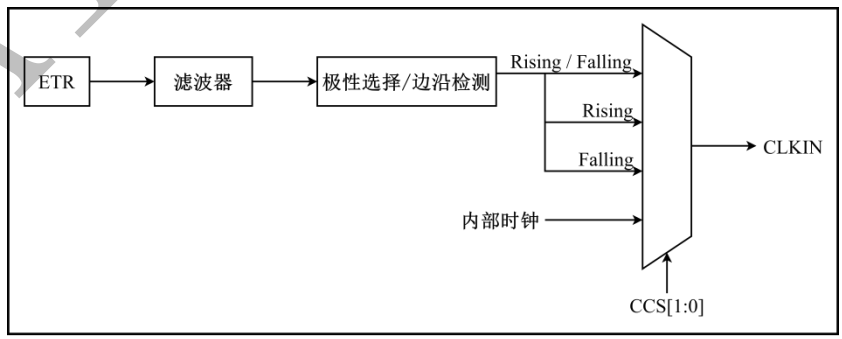


图 12-11　外部时钟连接示意图

注意：外部输入时钟源从功能引脚输入会经过一个滤波器消抖，该滤波器的消抖使能在 GPIO 模块中的“去抖使能寄存器(GPIOx_DER)”中设置。开启滤波器功能后，只有当输入电平宽度至少达到 4 个时钟周期（时钟源可配，具体参考 GPIO 文档）才会被送出，否则会被当作信号抖动被滤波器滤掉。

12.4.4 捕获/比较通道

每个通用定时器均带有一个独立的捕获/比较通道，捕获/比较通道均基于一个捕获/比较寄存器(包括一个影子寄存器)、一个捕获输入(滤波器、极性判断和预分频器)和一个比较输出(比较器和输出控制)组成。

输入捕获，是对相应的端口输入信号进行采样，经过滤波器处理后输出，再通过极性选择和边沿检测生成一个触发信号作为输入捕获触发信号，用于及时捕获到指定边沿发生时，计数器的准确数值。通过这样可实现对外部信号的精准捕获和测量，由此可实现用于对信号进行脉宽测量、周期测量、计算占空比等功能。

输出比较，是利用计数器计数，并与预先配置好的比较值进行对比，当计数值与比较值匹配时，根据所设定的比较输出模式，相应的端口进行相应的输出，从而实现例如 PWM、反转、强制极性输出等功能。

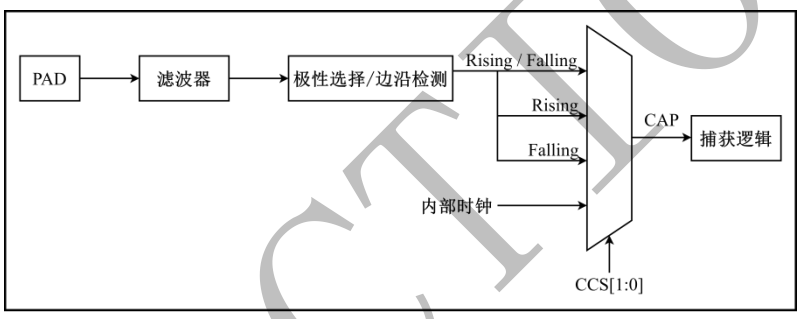


图 12-12　输入捕获通道示意图

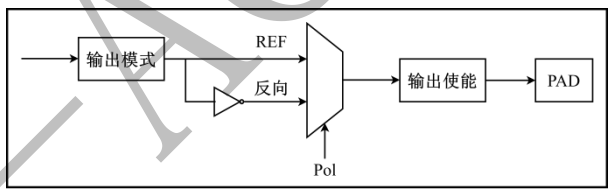


图 12-13　比较输出通道示意图

- 在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容复制到预装载寄存器中。
- 在比较模式下，预装载寄存器的内容将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

12.4.5 输入捕获

定时器可通过配置捕获/比较控制寄存器(TMRx_CCCR)中的模式选择位(CCS=1)，并设置使能捕获/比较功能(CCE=1)，使定时器工作在输入捕获模式下。

此外，通过配置捕获/比较控制寄存器(TMRx_CCCR)中的极性选择位域(CCP[1:0])可以选择捕获模式下的触发边沿，配置输入捕获源选择位域(ICSRC[2:0])可选择捕获的目标功能引脚或使用比较器模块(CMPx)的内部信号。

定时器处于输入捕获模式下后，当检测到捕获源产生了目标跳变沿之后，定时器立刻将当前计数器的值锁存捕获到捕获/比较寄存器(TMRx_CCR)内，同时设置输入捕获中断标志位(ICIF)

置 1，并触发定时器中断(如果 ICIE=1)，软件上需要及时对输入捕获中断标志位进行清除(ICIF 写 1 清 0，或通过读取捕获/比较寄存器，硬件将自动清除 ICIF 位)。如果在清除该标志位之前，再次捕获到目标跳变沿，会将新的计数值锁存并覆盖捕获/比较寄存器(TMRx_CCR)内的值，同时将会产生重复采样导致输入捕获重复采样标志位(ICOIF)将被置 1，并触发定时器中断(如果 ICOIE=1)。因此在处理输入捕获时，建议在捕获状态标志位置起后应当及时读取数据。否则一旦发生重复捕获，新捕获的数据将覆盖掉之前捕获的数据，导致捕获信息丢失。

12.4.6 输出比较

定时器可通过配置捕获/比较控制寄存器(TMRx_CCCR)中的模式选择位(CCS=0)，并设置使能捕获/比较功能(CCE=1)，使定时器工作在输出比较模式下。

此外，通过配置捕获/比较控制寄存器(TMRx_CCCR)中的比较输出模式位域(OCM[2:0])选择输出比较的工作模式、极性选择位域(CCP[1:0])可以选择比较模式下有效电平的极性、设置输出比较预装载使能位(OCPE)可使能输出比较的预装载功能。

当定时器的计数器的值(TMRx_CNTR)与捕获/比较寄存器(TMRx_CCR)的值相匹配时，将根据目前设置的输出比较工作模式对功能脚输出相应的信号，同时输出比较中断标志位(OCIF)将被置 1，并触发定时器中断(如果 OCIE=1)，软件上可通过对输出比较中断标志位进行清除（OCIF 写 1 清 0）。

输出比较工作模式

目前定时器支持的输出比较工作模式有以下几种，通过 OCM[2:0]位域进行设置：

- 000 ：冻结，比较匹配后不对功能脚产生变化，保持原有输出。
- 001 ：匹配后输出有效电平（有效电平极性通过 CCP[1:0]进行设置）
- 010 ：匹配后输出无效电平（与有效电平相反）
- 011 ：匹配后翻转电平
- 100 ：强制变为无效电平
- 101 ：强制变为有效电平
- 110 ：PWM 模式 1。当计数器值 ≤ 比较值时，输出有效电平，否则为无效电平。
- 111 ：PWM 模式 2。当计数器值 ≤ 比较值时，输出无效电平，否则为有效电平。

配合定时器中的单次/连续计数模式，还可实现更多组合功能，例如利用单次计数模式+输出比较 PWM 模式可实现单脉冲输出功能。更多特别模式说明将在后续章节进行更详细介绍。

输出比较的预装载功能

在输出比较的过程中，软件可通过修改捕获/比较寄存器(TMRx_CCR)来修改用于比较的值，通过设置输出比较预装载使能控制位(OCPE)，可配置是否开启上述寄存器的预装载功能。预装载的作用如下：

- 不开启预装载功能，向 TMRx_CCR 写入新值，将立即更新到内部影子寄存器内并立即生效。
- 开启预装载功能，向 TMRx_CCR 写入新值，并不会立即生效，而是在得到比较成功或软件产生捕获/比较事件(CCEV)之后，再将数值更新至内部影子寄存器内生效。更详细的捕

获/比较事件描述，请参考“12.4.9.3 捕获/比较事件”章节说明。

12.4.6.1 强制输出

在输出模式下，可直接通过软件将输出比较信号强制设置为有效电平或无效电平，而无需考虑输出比较寄存器和计数器之间的比较结果。

配置强制输出模式，只需将 TMRx_CCCR 寄存器中比较输出模式选择位域(OCM[2:0])配置为 100 或 101，在配置极性位域(CCP[1:0])选择合适的有效电平。

在配置为强制输出模式后，虽然定时的功能引脚输出将会根据配置强制输出指定的电平，但定时器的计数器值与比较值的匹配行为依然在执行，匹配时依然会引发相应的标志位(OCIF)并触发中断处理。

- 配置强制输出无效电平模式(OCM=100)时：
  - CCP=0（有效电平为高电平），将比较强制输出为低电平
  - CCP=1（有效电平为低电平），将比较强制输出为高电平
- 配置强制输出有效电平模式(OCM=101)时：
  - CCP=0（有效电平为高电平），将比较强制输出为高电平
  - CCP=1（有效电平为低电平），将比较强制输出为低电平

12.4.6.2 PWM 输出

输出比较的 PWM 模式，可以用于生成一个调制信号，该信号频率周期由计数器配置的基础定时周期决定(也就是由 TMRx_PSCR、TMRx_CSVR、TMRx_CEVR 寄存器共同决定)，而占空比则由捕获/比较寄存器(TMRx_CCR)设定的值决定。PWM 模式有两种：

- PWM 模式 1（OCM=110）

该模式下，当计数器值 < 比较值时输出有效电平，否则输出无效电平。

- PWM 模式 2（OCM=111）

该模式下，当计数器值 < 比较值时输出无效电平，否则输出有效电平。

有效电平通过捕获/比较控制寄存器(TMRx_CCCR)中的极性设置位域(CCP[1:0])选择合适的有效电平。

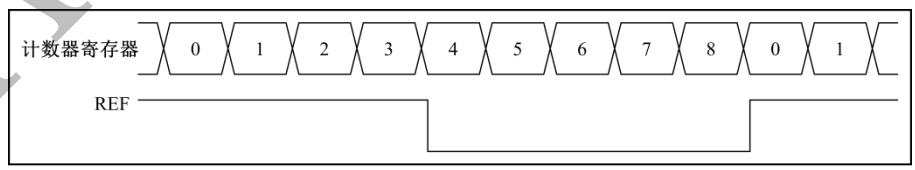


图 12-14 PWM 输出（TMRx_CEVR=8，TCCRR=4，Polarity=0 高电平有效）

12.4.6.3 单脉冲模式

单脉冲模式 (One-Pulse Mode) 是单次计数模式+PWM 模式的一个特例。在这种模式下，可以配置输出一个可控脉宽的单脉冲信号。

首先将定时器模块配置为单次计数模式(MS=1)，配置输出比较工作模式为 PWM 模式 1/2，配置捕获/比较寄存器(TMRx_CCR)写入合适的比较值，再根据电平需求配置有效电平的极性(CCP[1:0])。这样启动定时器计数后，将会输出一段可控宽度的单脉冲。

12.4.7 定时器同步

每个定时器都是完全独立的，定时器之间没有共享任何资源。

为了实现定时器同步启动功能，4 套基础定时器之间增加同步启动的功能。

通过定时器组同步控制寄存器(TMRG_SYNCR)，实现同组定时器的同步启动/关闭功能。

12.4.8 调试模式

当内核进入调试模式（Cortex™-M3 内核停止）时，定时器内部计数器会根据 SYSCTRL 模块中系统控制寄存器的 TMR1 调试使能位(SYSCTRL_SYSCR.TMR1DE)来选择继续正常工作或者停止工作。

12.4.9 事件与中断

通用定时器在多种情况下会产生相应的事件，事件主要用于更新相应的影子寄存器、执行特定功能以及产生相应中断。

通用定时器内主要有以下几种事件：

- 定时器上溢事件，简称上溢事件 OVEV(Overflow Event)
- 定时器更新事件，简称更新事件 UEV (Update Event)
  需要使用更新事件，必须将定时器控制寄存器(TMRx_CR)中的禁止更新位(UDIS)置 0
- 定时器输入捕获/输出比较事件，简称捕获/比较事件 CCEV (Capture/Compare Event)

中断主要包含以下五种中断触发：

- 定时器上溢中断，发生上溢事件后，上溢中断标志位(OVIF)同时被置位。
- 定时器更新中断，发生更新事件后，更新中断标志位(UIF)同时被置位。
  特别说明：如果更新请求源选择位(URS)置 1，即仅上溢事件会产生更新事件，那么软件上如果通过设置更新产生位(UG)，将只会产生更新事件用于更新预加载相关的寄存器，但不会触发定时器更新中断。
- 定时器输入捕获中断，当定时器为输入捕获模式并捕获到相应的边沿时，输入捕获中断标志位（ICIF）将被置位。
- 定时器重复捕获中断，当输入捕获中断标志位已经置位(ICIF=1)，再次捕获到相应边沿，则输入捕获重复捕获中断标志位（ICOIF）将被置位。
- 定时器输出比较中断，当定时器位输出比较模式，并且计数值与比较值相匹配时，输出比较中断标志位(OCIF)将被置位。

12.4.9.1 上溢事件（Overflow Event）

上溢事件(OVEV)由以下情况产生：

- 计数器寄存器(TMRx_CNTR)计数值到达计数终止值寄存器(TMRx_CEVR)设定的值。

上溢事件的作用

计数器到达上溢后，计数器寄存器会立即重新加载计数值起始寄存器(TMRx_CSVR)的值，重新开始计数(如果为连续模式)。

事件产生后，中断状态寄存器(TMRx_ISR)中的上溢中断标志位(OVIF)将会置 1，并且会触发定时器中断(如果 OVIE = 1)。

12.4.9.2 更新事件（Update Event）

更新事件(UEV)可以由以下两种情况产生：

- 计数器上溢：计数器上溢事件产生的同时会产生更新事件。
- 软件置位：软件将定时器事件产生寄存器(TMRx_EGR)中的更新标志位(UG) 置 1。

更新事件使用的注意事项

- 如果需要使用更新事件(UEV)，必须将定时器控制寄存器(TMRx_CR)中的更新禁止位(UDIS)置 0，使能更新事件功能，否则更新事件将被禁止。
- 如果需要软件置位功能，还必须要配置定时器控制寄存器(TMRx_CR)中的更新请求位(URS)置 0。否则只有计数器上溢才会产生更新事件。

更新事件的作用

事件产生后，中断状态寄存器(TMRx_ISR)中的更新中断标志位(UIF)将会置 1，并且会触发定时器中断 (如果 UIE = 1)。

特别的，如果在计数器计数的过程中(未达到计数器上溢)，通过软件设置 UG 位产生更新事件后，计数器寄存器也会立即重新加载计数值起始寄存器(TMRx_CSVR)的值，重新开始计数(如果为连续模式)。不同之处在于：通过软件更新标志位(UG)设置产生的更新事件，不会触发上溢事件的中断。

更新事件对自动装载预加载(Auto-Reload Preload)功能的作用

当定时器开启自动装载预加载功能（TMRx_CR 寄存器内 ARPE=1），更新事件产生后，才会将以下几个寄存器的值，更新到内部影子寄存器内使之生效：

- 计数终止值寄存器(TMRx_CEVR)
- 预分频寄存器(TMRx_PSCR)
- 捕获/比较寄存器(TMRx_CCR)

注意：软件置位(UG=1)产生的更新事件，不会影响捕获/比较寄存器(TMRx_CCR)。

如果定时器未开启自动装载预加载功能（TMRx_CR 寄存器内 ARPE=0），则上述寄存器写入新值将立即生效，更新事件将不影响上述寄存器的配置。

12.4.9.3 捕获/比较事件（Capture/Compare Event）

捕获/比较事件(CCEV)，可以由以下三种情况产生：

- 比较成功：计数器寄存器(TMRx_CNTR)的计数值与捕获/比较寄存器(TMRx_CCR)的值比较成功时(仅针对输出比较模式下)。
- 捕获成功：外部输入一个与捕获/比较控制寄存器(TMRx_CCCR)中所设置的极性(CCP 位)相符的边沿信号时（仅针对输入捕获模式下）。
- 软件产生：定时器事件产生寄存器(TMRx_EGR)中相应的事件产生标志位(CCG)置 1。

捕获/比较事件的作用

在不同模式下捕获/比较事件的作用不相同：

- 对于输出比较模式：
  - 捕获/比较事件产生后，中断状态寄存器(TMRx_ISR)中的比较中断(OCIF)将会置 1，并触发定时器中断(如果比较中断使能，OCIE 位置 1)。
  - 如果使能了比较器预加载功能(捕获/比较控制寄存器 TMRx_CCCR 中的 OCPE 位置 1)，事件产生后(这里特指比较成功所产生事件)，定时器将会执行以下动作：
    - 定时器的输出，将根据捕获/比较控制寄存器(TMRx_CCCR)中设置的比较输出模式(OCM[2:0])，对输出状态进行相应的改变；
    - 将捕获/比较寄存器(TMRx_CCR)的值重新载入内部影子寄存器，使之生效。
    - 将定时器的计数器值(TMRx_CNTR)重新加载为计数起始值寄存器(TMRx_CSVR)的值，并重新开始计数(循环模式)。
  - 如果没有使能比较器预加载功能(捕获/比较控制寄存器 TMRx_CCCR 中的 OCPE 位置 0)，事件产生后，定时器的动作与上述情况不同之处在于：对捕获/比较寄存器(TMRx_CCR)写入新值将立即生效，捕获/比较事件不会影响该寄存器值何时生效。

注意：如果在到达比较值之前(未达到比较成功)，通过软件对 CCG 置 1 产生事件，与上述描述基本一致，不同之处在于：不会导致比较输出模式(OCM[2:0])所设定的输出变化，因为本次计数器值并没有与触发捕获/比较事件前所设定的比较值相匹配。

- 对于输入捕获模式：
  - 捕获/比较事件产生后，定时器将当前计数器寄存器(TMRx_CNTR)的值捕获到捕获/比较寄存器(TMRx_CCR)内；
  - 接着中断状态寄存器(TMRx_ISR)中的捕获中断标志(ICIF)将会置 1，并触发定时器中断(如果 ICIE=1)。如果本次捕获成功产生捕获/比较事件时，前一次捕获中断标志未及时清除(ICIF 未清 0)，则中断状态寄存器(TMRx_ISR)中的重复捕获中断标志(ICOIF)将会置 1，并且触发定时器中断(如果使能 ICOIE 置 1)。

注意：如果通过软件对 CCG 置 1 产生的捕获/比较事件，则立刻执行上述动作，而不用外部产生符合捕获成功的信号。

12.4.9.4 中断

中断标志位与中断使能之间的关系如下表描述所示，详细内容请参考“12.4.9 事件与中断”章节描述：

表 12-1 中断标志位与中断使能之间的关系

中断名称	中断使能	中断标志	清除方式	备注
上溢中断	OVIE	OVIF	W1C	计数器计数上溢后产生
更新中断	UIE	UIF	W1C	计数器更新事件触发更新中断
输入捕获中断	ICIE	ICIF	W1C 或读 CCR	输入捕获模式下，捕获到目标边沿
重复捕获中断	ICOIE	ICOIF	W1C	输入捕获中断未及时清除，再次发生捕获
输出比较中断	OCIE	OCIF	W1C	输出比较模式下，计数值与比较值匹配

12.5 寄存器描述

12.5.1 寄存器列表

TMRx 基地址：**0x4002 7000 + 0x100*x [x=0~3]**

TMRG 基地址：**0x4002 7400**

偏移	实例地址	名称	默认值	描述
0x00	0x40027000	TMRx_CR	0x00000000	TMRx 控制寄存器
0x04	0x40027004	TMRx_CCCR	0x00000000	TMRx 捕获/比较控制寄存器
0x08	0x40027008	TMRx_EGR	0x00000000	TMRx 事件生成寄存器
0x0C	0x4002700C	TMRx_ICFR	0x00000000	TMRx 输入捕获滤波寄存器
0x10	0x40027010	TMRx_ISR	0x00000000	TMRx 中断状态寄存器
0x20	0x40027020	TMRx_CSVR	0x00000000	TMRx 计数起始值寄存器
0x24	0x40027024	TMRx_CEVR	0x0000FFFF	TMRx 计数终止值寄存器
0x28	0x40027028	TMRx_CCR	0x00000000	TMRx 捕获/比较寄存器
0x2C	0x4002702C	TMRx_PSCR	0x00000000	TMRx 预分频寄存器
0x30	0x40027030	TMRx_CNTR	0x00000000	TMRx 计数寄存器
0x00	0x40027400	TMRx_SYNCR	0x00000000	定时器组同步寄存器

【说明】上表中，x=4~7。

12.5.2 寄存器描述

12.5.2.1 TMRx 控制寄存器（TMRx_CR）

- 名称：TMRx Control Register（x = 4~7）
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:14	13:12	11:10	9	8	7:5	4	3	2	1	0
名称		CKSRC		UIE	OVIE		ARPE	MS	URS	UDIS	CEN
访问		R/W		R/W	R/W		R/W	R/W	R/W	R/W	R/W
复位值		0x0		0x0	0x0		0x0	0x0	0x0	0x0	0x0

字段	说明
[13:12] CKSRC	**定时器时钟源（Timer Clock Source）** 00：系统时钟上升沿　　01：外部时钟上升沿 10：外部时钟下降沿　　11：外部时钟上下沿
[9] UIE	**更新事件中断使能控制（Timer Update Interrupt Enable）** 1：使能中断 0：禁止中断 注意：如果需要使用更新事件，则要配置更新事件功能开启（TMRx_CR 寄存器的 UDIS 位置 0）。更详细介绍请参看“12.4.9.2 更新事件（Update Event）”章节。
[8] OVIE	**计数器上溢中断使能控制（Timer Counter Overflow Interrupt Enable）** 1：使能中断 0：禁止中断
[4] ARPE	**定时器自动重载预装载使能（Timer Auto-Reload Preload Enable）** 此位用于开启/关闭影子寄存器预加载功能。 0：寄存器不进行预加载 1：寄存器进行预加载 注意： 1. 预加载功能涉及的寄存器有： ➢ 计数终止值寄存器（TMRx_CEVR） ➢ 预分频寄存器（TMRx_PSCR） ➢ 捕获/比较寄存器（TMRx_CCR） 但他们的更新事件分别由不同的情况产生。 2. 如果不开启预加载功能，则相关寄存器写入新值将立刻生效 3. 如果开启预加载功能，则相关寄存器写入新值后，在下一个更新事件（UEV）到来后生效。 更详细信息请参看“12.4.9.2 更新事件（Update Event）”章节。

[3] MS	**定时器工作模式选择（Timer Mode Selection）** 0：连续模式 1：单次模式 注意：配置为单次模式后，计数器溢出后，将自动停止计数。
[2] URS	**定时器更新事件源选择（Timer Update Request Source）** 0：使能后，所有以下事件都会生成更新事件： ➢ 计数器上溢 ➢ 将 UG 位置 1 1：使能后，只有计数器上溢会生成更新事件。
[1] UDIS	**定时器更新事件禁止位（Timer Update Disable）** 此位由软件置 1 和清零，用以使能/禁止更新事件生成。 0：更新使能，更新（UEV）事件可通过以下事件之一生成： ➢ 计数器上溢 ➢ 将 UG 位置 1 1：更新禁止 注意：配置为 1 不会生成更新事件（不会产生 Counter 更新中断），各影子寄存器的值保持不变，如果 UserGenerate 位置 1，则会重新初始化计数器和预分频器。
[0] CEN	**计数器使能位（Timer Counter Enable）** 1：开始计数器 0：关闭计数器

12.5.2.2 TMRx 捕获/比较控制寄存器（TMRx_CCCR）

- 名称：TMRx Capture/Compare Control Register（x=4~7）
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:15	14:12	11	10	9	8	7:5	4	3	2:1	0
名称		OCM		OCIE	ICOIE	ICIE	ICSRC	OCPE	CCS	CCP	CCE
访问		R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[14:12] OCM	**定时器输出比较器模式（Timer Output Compare Mode）** 000：冻结，输出比较寄存器与计数器进行比较不会对输出造成任何影响（保持原有的输出状态） 001：当计数器与捕获/ 比较寄存器匹配后，输出有效电平—参考有效电平极性 010：当计数器与捕获/ 比较寄存器匹配后，输出无效电平—参考有效电平极性 011：当计数器与捕获/ 比较寄存器匹配后，发生翻转 100：强制变为无效电平—参考有效电平极性 101：强制变为有效电平—参考有效电平极性 110：PWM 模式 1 ➢ 当计数 Counter 值≤ Compare 值时 输出有效状态，否则输出无效状态（有效电平取决于极性寄存器） 111：PWM 模式 2 ➢ 当计数 Counter 值≤ Compare 值时 输出无效状态，否则输出有效状态（有效电平取决于极性寄存器）
[10] OCIE	**比较器中断使能位（ Timer Compare Interrupt Enable）** 1：开启中断使能 0：关闭中断使能
[9] ICOIE	**定时器重复捕获中断使能控制（Timer OverCapture Interrupt Enable）** 1：开启中断使能 0：关闭中断使能
[8] ICIE	**定时器捕获中断使能控制（Timer Capture Interrupt Enable）** 1：开启中断使能 0：关闭中断使能

[7:5] ICSRC	**TMRx 的输入捕获源选择（Timer Input Capture Source Selection）** 0x0：TMR4 的功能引脚；　0x1：TMR5 的功能引脚； 0x2：TMR6 的功能引脚；　0x3：TMR7 的功能引脚； 0x4：比较器模块 0 的内部输出（CMP0 Internal Output） 0x5：比较器模块 1 的内部输出（CMP1 Internal Output） 0x6：比较器模块 2 的内部输出（CMP2 Internal Output） 0x7：PWM Sync Output 注意：功能引脚的使用，请参考数据手册中的引脚复用（Pinmux）介绍。
[4] OCPE	**输出比较器预加载使能控制（Timer Output Compare Preload Enable）** 0：禁止比较器预加载功能 1：使能比较器预加载功能 更详细介绍请参考“12.4.9.3 捕获/比较事件（Capture/Compare Event）”章节
[3] CCS	**定时器捕获/比较模式选择（Timer Capture /Compare Select）** 0：配置为输出比较模式（Output Compare） 1：配置为输入捕获模式（Input Capture）
[2:1] CCP	**捕获/比较极性配置位（Timer Capture /Compare Polarity）** ● **配置为输出比较时（CCS=0）：** ➢ 0：高电平有效 ➢ 1：低电平有效 ● **配置为输入捕获时（CCS=1）：** ➢ 00：上升沿触发 ➢ 01：下降沿触发 ➢ 1x：上升沿和下降沿均触发
[0] CCE	**定时器捕获/比较使能控制（Timer Capture/Compare Enable）** ● **配置为输出比较时（CCS=0）：** ➢ 0：关闭输出—引脚将处于悬空状态 ➢ 1：开启输出—在相应引脚上输出 Compare 信号 ● **配置为输入捕获时（CCS=1）：** ● 该位将决定在捕获发生后，计数器值是否能被存储到捕获/比较寄存器内（TMRx_CCR）： ➢ 0：禁止捕获 ➢ 1：使能捕获

12.5.2.3 TMRx 事件生成寄存器（TMRx_EGR）

- 名称：TMRx Event Generation Register（x = 4~7）
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:9	8	7:1	0
名称		CCG		UG
访问		R/W		R/W
复位值		0x0		0x0

字段	说明
[8] CCG	**定时器捕获/比较事件生成（Timer Capture/Compare Generation）** 此位由软件置 1 以生成一次软件的捕获/比较事件，并由硬件自动清零。 0：不执行任何操作 1：软件重生捕获/比较事件 详细说明请参考“12.4.9.3 捕获/比较事件（Capture/Compare Event）”章节。
[0] UG	**定时器计数更新事件生成（Timer Counter Update Generation）** 此位由软件置 1 以生成一次更新事件，并由硬件自动清零 0：不执行任何操作 1：软件重生更新事件 详细说明请参考“12.4.9.2 更新事件（Update Event）”章节。

12.5.2.4 TMRx 输入捕获滤波寄存器（TMRx_ICFR）

- **名称：TMRx Input Capture Filter Register（x = 4~7）**
- **偏移地址：0x0C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:8	7:0
名称		ICF
访问		R/W
复位值		0x0

字段	说明
[7:0] ICF	**定时器输入捕获滤波器值（Timer Input Capture Filter value）** 输入捕获滤波器，该位域可用户定义外部输入信号的数字滤波带宽。通过设定该位域，可实现每 N 个连续事件才视为一个有效的边沿输出，其中： N =滤波器值（ICF）+ 1 例如：输入捕获模式下，配置极性为上升沿捕获（CCP=00），滤波器值为 3（ICF=3），则 N = 3 + 1 = 4，意味着每 4 个连续的上升沿才视为一个有效的上升沿，触发捕获/比较事件。 更详细介绍可参考“12.4.5 输入捕获”和“12.4.9.3 捕获/比较事件（Capture/Compare Event）”章节。

12.5.2.5 TMRx 中断状态寄存器（TMRx_ISR）

- **名称：TMRx Interrupt Status Register（x = 4~7）**
- **偏移地址：0x10**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:5	4	3	2	1	0
名称		OVIF	ICOIF	ICIF	OCIF	UIF
访问		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[4] OVIF	**计数器上溢中断标志位（Timer Counter Overflow Interrupt Flag）** 0：未发生上溢 1：计数器上溢

[3] ICOIF	**定时器输入捕获重复捕获中断标志位（Timer Input Capture Over-capture Interrupt Flag）** 输入捕获模式下有效（CCS=1），当捕获标志位（ICIF）已经置 1，再次发生输入捕获事件，则称为重复捕获，重复捕获标志位将被置 1。 0：未检测到重复捕获 1：发生重复捕获 更详细描述请参考“12.4.5 输入捕获”章节。
[2] ICIF	**定时器输入捕获标志位（Timer Input Capture Interrupt Flag）** 输入捕获模式下有效（CCS=1）： 0：未发生输入捕获 1：发生输入捕获（检测到与极性配置位 CCP 设置匹配的边沿），同时当前计数器的值将会被捕获到捕获/比较寄存器（TMRx_CCR）内。 注意：读取捕获/比较寄存器（TMRx_CCR）同样导致该位自动清 0。 更详细描述请参考“12.4.5 输入捕获”章节。
[1] OCIF	**输出比较匹配标志位（Timer Output Compare Interrupt Flag）** 输出比较模式下有效（CCS=0）： 0：输出比较未匹配 1：输出比较匹配（当前计数器的值与捕获/比较寄存器 TMRx_CCR 内设定的值相匹配）。 更详细描述请参考“12.4.6 输出比较”章节
[0] UIF	**定时器计数更新中断标志位（Timer Counter Update Interrupt Flag）** 0：未发生更新事件 1：产生更新事件 注意：使用更新事件还需要将定时器控制寄存器（TMRx_CR）中的禁止更新位（UDIS）置 0，否则更新事件将被屏蔽。 更多详细描述请参考“12.4.9.2 更新事件（Update Event）”章节。

12.5.2.6 TMRx 计数起始值寄存器（TMRx_CSVR）

- 名称：TMRx Counter Start Value Register（x = 4~7）
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CSVR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**计数起始值（Timer Counter Start Value）**
CSVR	计数起始值寄存器，通过配置此寄存器设置定时器开始计数的值。

12.5.2.7 TMRx 计数终止值寄存器（TMRx_CEVR）

- 名称：TMRx Counter End Value Register（x = 4~7）
- 偏移地址：0x24
- 默认值：0x0000FFFF
- 返回寄存器列表

位	31:0
名称	CEVR
访问	R/W
复位值	0xFFFF

字段	说明
[31:0]	**计数终止值（Timer Counter End Value）**
CEVR	通过配置此寄存器设置定时器结束计数的值。此外，该寄存器支持自动预装载（Auto-Reload）功能： ● **不使能自动预装载功能（ARPE=0）** 对该寄存器的设置将立即作用到内部影子寄存器内，本次计数周期内即生效； ● **使能自动预装载功能（ARPE=1）** 对该寄存器的设置将在更新事件（UEV）产生后，再更新至内部影子寄存器内生效。 更详细介绍请参考“12.4.1 时基单元”和“12.4.9.2 更新事件（Update Event）”章节。

12.5.2.8 TMRx 捕获/比较寄存器（TMRx_CCR）

- 名称：TMRx Capture/Compare Register（x = 4~7）
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CCR
访问	R/W
复位值	0x0

字段	说明
[31:0] CCR	**定时器捕获/比较值（Timer Capture/Compare value）** 不同的工作模式下该寄存器的功能描述如下： ● **配置为输出比较时（CCS=0）** 该寄存器写入计数比较值，用于当前的定时器计数器值（TMRx_CNTR）与该寄存器比较，比较一致时产生捕获/比较事件（CCEV）。在该工作模式下时，该寄存器支持输出比较预装载功能： ➢ **不使能输出比较预装载功能时（OCPE=0）** 对该寄存器写入的值将立即生效，作为新的比较值用于输出比较。 ➢ **使能输出比较预装载功能时（OCPE=1）** 对该寄存器写入的值将在以下几种事件触发后，才会使新的值真正加载进入内部影子寄存器生效： – 下一次更新事件（UEV，且仅限于计数器上溢产生的更新事件） – 之前的比较值所产生的捕获/比较事件（CCEV） – 软件通过设置 CCG 位产生的捕获/比较事件（CCEV） 更详细描述请参考“12.4.6 输出比较”、“12.4.9.2 更新事件（Update Event）”和“12.4.9.3 捕获/比较事件（Capture/Compare Event）”等相关章节。 ● **配置为输入捕获时（CCS=1）** 该寄存器用于当外部输入信号产生与极性（CCP）所设置一样的边沿时，将当前定时器计数器的值捕获到该寄存器内，产生捕获/比较事件（CCEV）。 此外，通过软件设置 CCG 位，可以立即产生捕获/比较事件（CCEV），而无需外部输入信号产生相应的边沿。 更多描述请参考“12.4.5 输入捕获”和“12.4.9.3 捕获/比较事件（Capture/Compare Event）”等相关章节。

12.5.2.9 TMRx 预分频寄存器（TMRx_PSCR）

- 名称：TMRx Prescaler Register（x = 4~7）
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	PSC
访问	R/W
复位值	0x0

字段	说明
[31:0] PSC	**定时器预分频值（Timer Prescaler value）** 定时器计数器的计数频率（CK_CNT）通过此预分频器对时钟源的频率（CK_SRC）进行分频得到： $f_{CK\_CNT} = f_{CK\_SRC} /（PSC + 1）$ 此外，该寄存器支持自动预装载（Auto-Reload）功能： ● **不使能自动预装载功能（ARPE=0）** 对该寄存器的设置将立即作用到内部影子寄存器内，本次计数周期内即生效； ● **使能自动预装载功能（ARPE=1）** 对该寄存器的设置将在更新事件（UEV）产生后，再更新至内部影子寄存器内生效。 更详细介绍请参考“12.4.1 时基单元”和“12.4.9.2 更新事件”章节。

12.5.2.10 TMRx 计数寄存器（TMRx_CNTR）

- 名称：TMRx Counter Register（x = 4~7）
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CNT
访问	R/W
复位值	0x0

字段	说明
[31:0] CNT	**定时器计数值（Timer Counter Value）** 通过该寄存器可对定时器计数器的值进行读取/写入。需注意，因为系统与外设之间存在一定的总线延迟，对该寄存器的读取/写入操作可能存在一定的偏差。

12.5.2.11 定时器组同步寄存器（TMRx_SYNCR）

- 名称：Timer Group Sync Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7	6	5	4	3	2	1	0
名称		SYNC7DIS	SYNC6DIS	SYNC5DIS	SYNC4DIS	SYNC7EN	SYNC6EN	SYNC5EN	SYNC4EN
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] SYNC7DIS	**定时器 TMR7 的同步暂停位（Timer7 Counter Disable）** 1：暂停计数器 0：无效
[6] SYNC6DIS	**定时器 TMR6 的同步暂停位（Timer6 Counter Disable）** 1：暂停计数器 0：无效
[5] SYNC5DIS	**定时器 TMR5 的同步暂停位（Timer5 Counter Disable）** 1：暂停计数器 0：无效
[4] SYNC4DIS	**定时器 TMR4 的同步暂停位（Timer4 Counter Disable）** 1：暂停计数器 0：无效
[3] SYNC7EN	**定时器 TMR7 的同步使能位（Timer7 Counter Enable）** 1：开始计数器 0：关闭计数器
[2] SYNC6EN	**定时器 TMR6 的同步使能位（Timer6 Counter Enable）** 1：开始计数器 0：关闭计数器
[1] SYNC5EN	**定时器 TMR5 的同步使能位（Timer5 Counter Enable）** 1：开始计数器 0：关闭计数器
[0] SYNC4EN	**定时器 TMR4 的同步使能位（Timer4 Counter Enable）** 1：开始计数器 0：关闭计数器

13 独立看门狗（IWDG）

13.1 简介

独立看门狗具有安全性高、使用灵活的特点。可用于检测并解决软件错误导致的故障，在定时器到达指定的超时值时触发系统复位。

独立看门狗(IWDG)由其专用低速时钟(LSI)驱动，因此即便在主时钟发生故障时仍然保持工作状态。

IWDG 最适合应用于那些需要看门狗作为一个在主程序之外，能够完全独立工作，并且对时间精度要求较低的场合。

13.2 结构框图

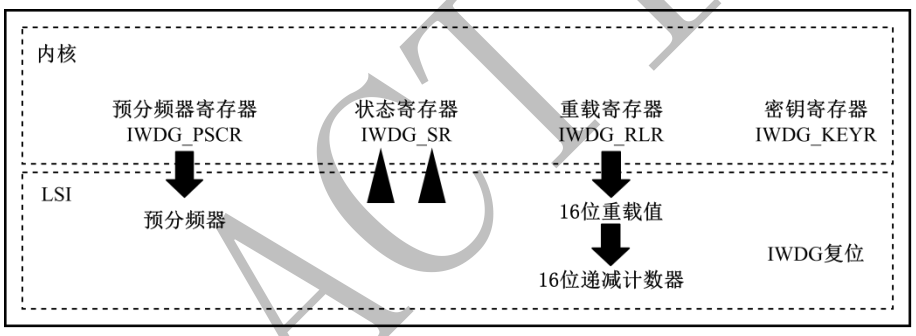


图 13-1 IWDG 架构示意图

13.3 主要特性

IWDG 主要具有以下特性：

- 自由运行递减计数器
- 16 位重载计数值寄存器，计数时钟支持 4 到 512 分频
- 时钟由独立 RC 振荡器(LSI)提供（可在待机模式下独立运行）
- 当递减计数器值达到 0x0 时产生复位（如果看门狗已激活）

13.4 功能描述

13.4.1 IWDG 键值功能

将键值 0xCCCC 写到键寄存器(IWDG_KEYR.KEY)中，将启动独立看门狗。看门狗内部计数器开始从复位值 0xFFFF 递减计数，当内部计数器计数到终值(0x0000) 时将会产生一个系统复位或触发中断。

任何时候将键值 0xAAAA 写到键寄存器(IWDG_KEYR.KEY)中，重载寄存器(IWDG_RLR）的值就会被重载到计数器，从而避免产生看门狗复位。

将键值 0xDDDD 写到键寄存器(IWDG_KEYR.KEY)中，将停止独立看门狗。

将键值 0x5003 写到键寄存器(IWDG_KEYR.KEY)中，将解除 IWDG 的寄存器访问保护，详见“13.4.2IWDG 寄存器访问保护”说明。

注意：重载操作(将键值 0xAAAA 写到 IWDG_KEYR.KEY*)必须在启动 IWDG 后，且 IWDG_SR.RLVUPD/PSCUPD 位都为 0 时才可以进行。*

13.4.2 IWDG 寄存器访问保护

IWDG 预分频寄存器(IWDG_PSCR)、重加载寄存器(IWDG_RLR)及控制寄存器(IWDG_CR)具有写访问保护。

若要修改上述寄存器，首先需要将 0x5003 写入键寄存器(IWDG_KEYR.KEY)解除上述寄存器的访问保护；写入其他值，则使寄存器访问保护重新生效，下次修改则需要重新解除访问保护。这意味着重装载操作（写 0xAAAA）也会启动写保护功能。

状态寄存器指示预分频值和递减计数器是否正在被更新。

13.4.3 IWDG 调试模式

当内核进入调试模式时（Cortex™-M3 内核停止），IWDG 计数器会根据 SYSCTRL 模块中系统配置寄存器中 IWDG 调试使能位(SYSCTRL_SYSCR.IWDGDE)选择继续正常工作或者停止工作。

13.4.4 IWDG 中断模式和复位模式

IWDG 支持两种工作模式：中断模式和复位模式。通过 IWDG_CR.MODE 位选择 IWDG 的工作模式。

中断模式下，IWDG 在未重载导致超时后，IWDG_SR.TOIF 将会置位，并触发中断(如果使能 IWDG 中断)，且不会导致系统复位。

复位模式下，IWDG 在未重载导致超时后，触发一个系统复位信号，如果使能 RCU 模块中系统配置寄存器中 IWDG 复位使能位(RCU_SCFGR.IWDGRE)，IWDG 复位则引发系统复位；如果 RCU_SCFGR.IWDGRE 未使能，则不会导致系统复位，直到该位使能。

13.4.5 IWDG 计时

表 13-1 32 KHz 频率条件下 IWDG 超时周期的最小值/最大值

预分频器	IWDG_PSCR.PSC	最短超时（ms） IWDG_RLR = 0x0	最长超时（ms） IWDG_RLR = 0xFFFF
/4	0	0.125	8192
/8	1	0.25	16384
/16	2	0.5	32768
/32	3	1	65536
/64	4	2	131072
/128	5	4	262144
/256	6	8	524288
/512	7	16	1048576

注意：

- *这些时间均针对 32kHz 时钟给出。实际上芯片内部的 RC 频率会在 30kHz 到 60kHz 之间变化。此外，即使 RC 振荡器的频率是精确的，确切的时序仍然依赖于 APB 接口时钟 RC 振荡器时钟之间的相位差，因此总会有一个完整的 RC 周期是不确定的；*
- *配置 IWDG_RLR 及 IWDG_PSCR 都需要从 APB 时钟域同步到 LSI 时钟域下，会一定程度影响 IWDG 的精度。*

13.4.6 IWDG 中断和状态

IWDG_SR 寄存器中定义了下述几个中断标志位和状态标志位。

标志名称	中断使能	标志位	备注
超时中断	TOIE	TOIF	IWDG 处于中断模式下时，未重载导致看门狗超时。
重载值更新状态	-	RLVUPD	置位时表示重载值正在更新，完成后硬件自动清 0。
分频值更新状态	-	PSCUPD	置位时表示分频值正在更新，完成后硬件自动清 0。

注意：IWDG 配置重载值寄存器(IWDG_RLR)和预分频寄存器(IWDG_PSCR)后，应当检查 RLVUPD 和 PSCUPD 标志，只有更新动作完成后，才可对看门狗进行重载操作。

13.5 寄存器描述

13.5.1 寄存器列表

IWDG 基地址：**0x4000 8000**

偏移	实例地址	名称	默认值	描述
0x00	0x40008000	IWDG_KEYR	0x00000000	IWDG 键寄存器
0x04	0x40008004	IWDG_CR	0x00000000	IWDG 控制寄存器
0x08	0x40008008	IWDG_RLR	0x0000FFFF	IWDG 重载寄存器
0x0C	0x4000800C	IWDG_PSCR	0x00000000	IWDG 预分频寄存器
0x10	0x40008010	IWDG_SR	0x00000000	IWDG 状态寄存器

13.5.2 寄存器描述

13.5.2.1 IWDG 键寄存器（IWDG_KEYR）

- 名称：IWDG Key Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		KEY
访问		R/W
复位值		0x0

字段	说明
[15:0]	**IWDG 键值（IWDG key value）**
KEY	详细请参看“13.4.1IWDG 键值功能”。

13.5.2.2 IWDG 控制寄存器（IWDG_CR）

- 名称：WDG Control Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:2	1	0
名称		TOIE	MODE
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[1]	**IWDG 超时中断使能位（IWDG Timeout Int Enable）**
TOIE	需要由软件解锁，写访问前对 0x00 寄存器写入值 0x5003； 0：关闭 1：使能 注意：仅中断模式下有效，详请参考“13.4.4 IWDG 中断模式和复位模式”。

[0]	**IWDG 模式配置位（IWDG Mode config）**
MODE	0：复位模式 1：中断模式 注意：详请参考“13.4.4 IWDG 中断模式和复位模式”章节。

13.5.2.3 IWDG 重载寄存器（IWDG_RLR）

- **名称：IWDG Reload Register**
- **偏移地址：0x08**
- **默认值：0x0000FFFF**
- 返回寄存器列表

位	31:16	15:0
名称		RLV
访问		R/W
复位值		0xFFFF

字段	说明
[15:0]	**IWDG 重载值（IWDG Reload Value）**
RLV	● 对 IWDG 进行重载操作（IWDG_KEYR 寄存器写 0xAAAA） 将 RLV 值重载至内部计数器重新开始计数。 注意： 1. 若要对 IWDG 进行重载，IWDG_SR 寄存器的 RLVUPD 和 PSCUPD 位必须都为 0。 2. 若要对 IWDG 的重载值进行修改，RLVUPD 位同样必须为 0。

13.5.2.4 IWDG 预分频寄存器（IWDG_PSCR）

- 名称：**IWDG Prescaler Register**
- 偏移地址：**0x0C**
- 默认值：**0x00000000**
- 返回寄存器列表

位	31:3	2:0
名称		PSC
访问		R/W
复位值		0x0

字段	说明
[2:0]	**IWDG 预分频器（IWDG Prescaler Divider）**
PSC	000：4 分频　001：8 分频　010：16 分频　011：32 分频 100：64 分频　101：128 分频　110：256 分频　111：512 分频 注意： 1. 若要对 IWDG 的分频进行修改，PSCUPD 位同样必须为 0。

13.5.2.5 IWDG 状态寄存器（IWDG_SR）

- 名称：**IWDG Status Register**
- 偏移地址：**0x10**
- 默认值：**0x00000000**
- 返回寄存器列表

位	31:3	2	1	0
名称		TOIF	RLVUPD	PSCUPD
访问		R/W1C	R	R
复位值		0x0	0x0	0x0

字段	说明
[2]	**IWDG 超时中断标志位（IWDG Timeout Interrupt Flag）**
TOIF	0：No Pending 1：Irq Pending 软件写 1 清 0，写 0 无效。

[1] RLVUPD	**IWDG 计数器重载值更新（IWDG Reload Value Update）** 通过硬件将该位置 1 以指示重载值正在更新，更新完成后会通过硬件将该位复位。 注意：重载值只有在该位为 0 时才可更新。
[0] PSCUPD	**IWDG 预分频器值更新（IWDG Prescaler Update）** 通过硬件将该位置 1 以指示预分频器值正在更新，更新完成后会通过硬件将该位复位。 注意：预分频器值只有在该位为 0 时才可更新。

14 窗口看门狗（WWDG）

14.1 简介

窗口看门狗(WWDG)时钟由 APB0 时钟经预分频后提供，窗口看门狗通常被用来监测，通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。除非递减计数器的值递减成 0 前被刷新，看门狗电路在达到预置的时间周期时，会产生一个系统复位。如果在递减计数器达到窗口寄存器值之前刷新控制寄存器中的递减计数器值，也会产生复位，这意味着必须在限定的时间窗口内刷新计数器。

14.2 结构框图

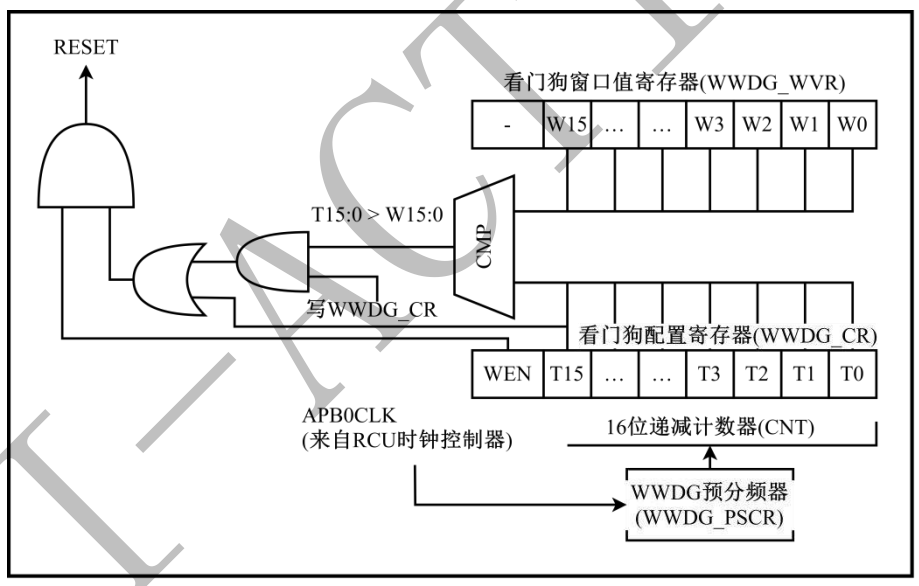


图 14-1 WWDG 架构框图

14.3 主要特性

WWDG 主要具有以下特性：

- 自由运行递减计数器
- 复位条件(若 RCU_SCFGR.WWDGRE 位置 1，则允许 WWDG 系统复位，详见 14.4.4 复位使能和调试模式介绍)
  - 当递减计数器值小于 0x40 时复位（如果看门狗已使能）
  - 在窗口之外重载递减计数器时复位（如果看门狗已使能）
- 提前唤醒中断(EWI)：当递减计数器减至 0x40 时触发（如果已使能且看门狗已使能），可用于在中断内重载计数器以避免 WWDG 复位

14.4 功能描述

如果使能看门狗(WWDG_CR.WEN 位置 1)，当 16 位递减计数器从 0x40 滚动到 0x3F 时会引发复位信号；如果软件在计数器值(WWDG_CVR)大于窗口寄存器(WWDG_WVR)中所存储的值时重载计数器，也会产生复位信号。

若 RCU_SCFGR.WWDGRE 位置 1，则 WWDG 的复位信号最终导致系统复位。

应用程序在正常运行过程中必须定期地写 WWDG_CVR 寄存器以防止发生复位。只有当计数器值低于窗口寄存器值时，才能执行此操作。存储在 WWDG_CVR 寄存器中的值须介于 0xFFFF 和 0x40 之间(如果设置为 0x40 将无法触发提前唤醒中断 Early Wakeup Interrupt)，用户应当根据实际应用需求合理配置。

14.4.1 使能看门狗

在系统复位后，看门狗处于关闭状态。可通过设置窗口看门狗控制寄存器(WWDG_CR)中的 WEN 位来使能看门狗，也可以按需随时关闭窗口看门狗功能。

14.4.2 控制递减计数器

在使能窗口看门狗之后递减计数器(WWDG_CVR.CV)就开始运行，每个计数周期执行一次递减操作。使能看门狗前，用户需要根据自己的应用需求确保以下限制：

- 递减计数器的值不应当配置 0x40 以下，否则启动窗口狗后将立即引发系统复位信号。
- 如果需要启用提前唤醒中断(EWI)功能，递减计数器的值不应当配置在 0x41 以下(仅在递减计数器从 0x41 递减至 0x40 时触发提前唤醒中断)

在 WWDG 开始工作后，为了防止系统复位，当递减计数器的值低于窗口寄存器(WWDG_WVR.WV)的值，且大于 0x3F 时，必须对递减计数器进行重载（向 WWDG_CVR.CV 写入新值）。重载后递减计数器从新值重新开始按计数周期递减。

14.4.3 看门狗中断高级特性

如果在产生实际复位之前必须执行特定的安全操作或数据记录，则可使用提前唤醒中断(EWI)。通过设置控制寄存器 WWDG_CR.EWIE 位使能该中断。当递减计数器的值从 0x41 递减至 0x40 时，将生成中断。在 WWDG 递减计数器继续递减到 0x3F 之前，可以使用相应的中断服务程序来触发特定操作（例如通信或数据记录）或重载递减计数器避免复位。

在某些应用中，可以使用 EWI 中断来管理软件系统检查和/或系统恢复/功能退化，而不会生成 WWDG 复位。在这种情况下，相应的中断服务程序可用来重载 WWDG 递减计数器以避免 WWDG 复位，然后再触发所需操作。

通过对中断状态寄存器 WWDG_ISR.EWIF 写 1 清 0，即清除提前中断。

注意：在提前唤醒中断的处理中，如果处理时间超过一个计数周期未重载递减计数器，计数

器从 0x40 递减至 0x3F，WWDG 将发出复位信号，引发系统复位(如果使能复位)。因此所有操作必须在这一个计数周期内完成，否则应当先关闭WWDG 或重载递减计数器再执行其他操作。

14.4.4 复位使能和调试模式

除了开启 WWDG 模块使能，还需将 RCU_SCFGR.WWDGRE 位置 1，才可使能 WWDG 的系统复位功能。否则 WWDG 发出的复位信号并不会引发实际的系统复位，直到使能 RCU_SCFGR.WWDGRE。

还可以通过将 SYSCTRL_SYSCR.WWDGDE 位置 1，开启 WWDG 的调试模式。开启后，当内核进入调试模式（Cortex™-M3 内核停止），WWDG 的递减定时器也会自动停止工作，直到退出调试模式。

14.4.5 看门狗设置

图 14-2 为 WWDG 计数时序图。

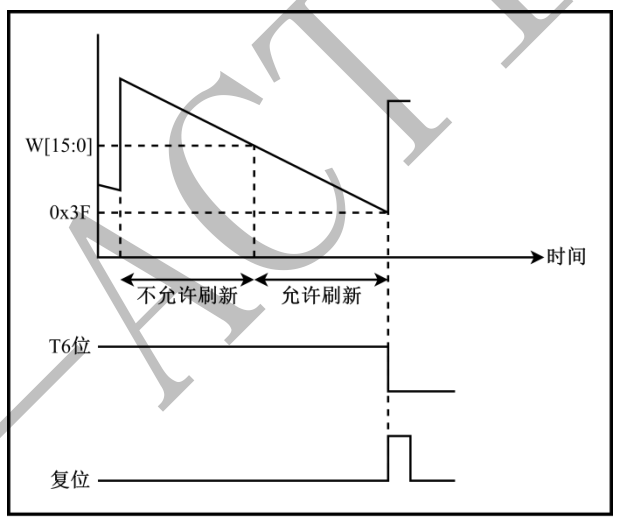


图 14-2　WWDG 计数时序图

超时值的计算公式如下：

$$t_{WWDG} = t_{APB0CLK} \times (PSC+1) \times (CVR+1) \quad (ms)$$

其中：

PSC　：WWDG_PSCR 窗口看门狗分频寄存器

CVR　：WWDG_CVR 窗口看门狗递减计数器

t_{WWDG}　：WWDG 超时时间

$t_{APB0CLK}$　：APB0 时钟周期，以 ms 为测量单位

例如，假设 APB0 频率为 50MHz（$t_{APB0CLK}$=1/50000 ms），PSC 值为 99，CVR 值为 999：

$$t_{WWDG} = (1/50000) \times (99 + 1) \times (999 + 1) = 2 \text{ ms}$$

14.5 寄存器描述

14.5.1 寄存器列表

IWDG 基地址：**0x4000 9000**

偏移	实例地址	名称	默认值	描述
0x00	0x40009000	WWDG_CR	0x00000000	WWDG 控制寄存器
0x04	0x40009004	WWDG_WVR	0x0000FFFF	WWDG 窗口值寄存器
0x08	0x40009008	WWDG_CVR	0x0000FFFF	WWDG 计数器值寄存器
0x0C	0x4000900C	WWDG_PSCR	0x00000000	WWDG 预分频寄存器
0x10	0x40009010	WWDG_ISR	0x00000000	WWDG 中断状态寄存器

14.5.2 寄存器描述

14.5.2.1 WWDG 控制寄存器（WWDG_CR）

- 名称：WWDG Control Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:2	1	0
名称		EWIE	WEN
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[1] EWIE	**WWDG 提前唤醒中断使能位（WWDG Early Wakeup Interrupt Enable）** 使能提前唤醒中断，在计数减至 0x40 时产生中断，用于提醒即将系统复位，需及时喂狗或执行复位前的必要操作。 0：关闭 1：使能
[0] WEN	**WWDG 使能位（WWDG Enable）** 0：关闭看门狗 1：使能看门狗

14.5.2.2 WWDG 窗口值寄存器（WWDG_WVR）

- 名称：WWDG Window Value Register
- 偏移地址：0x04
- 默认值：0x0000FFFF
- 返回寄存器列表

位	31:16	15:0
名称		WV
访问		R/W
复位值		0xFFFF

字段	说明
[15:0] WV	**WWDG 窗口比较值（WWDG Window Value）** 注意：用于与递减计数器进行比较的窗口值，只有当计数器值低于窗口寄存器值时，才能重载计数器值，否则也会产生复位操作。

14.5.2.3 WWDG 计数器值寄存器（WWDG_CVR）

- 名称：WWDG Counter Value Register
- 偏移地址：0x08
- 默认值：0x0000FFFF
- 返回寄存器列表

位	31:16	15:0
名称		CV
访问		R/W
复位值		0xFFFF

字段	说明
[15:0] CV	**WWDG 递减计数值（WWDG Counter Value）** 注意：每个计数周期递减一次，当它从 0x41 递减到 0x40 时引发提前唤醒中断；从 0x40 递减到 0x3F 时会产生复位。 正常运行过程中必须定期地写该寄存器以防止 MCU 发生复位，只有当计数器值低于窗口比较值时，才能执行此操作，否则也会产生复位操作。

14.5.2.4 WWDG 预分频寄存器（WWDG_PSCR）

- 名称：WWDG Prescaler Register
- 偏移地址：0x0C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		PSC
访问		R/W
复位值		0x0

字段	说明
[15:0] PSC	**WWDG 预分频器（WWDG Prescaler divider）** 用于对递减计数器时钟源（APB0CLK）分频。 分频关系 = PSC + 1 分频 0：不分频　　1：2 分频 2：3 分频　　3：4 分频 ……　　65535：65536 分频

14.5.2.5 WWDG 中断状态寄存器（WWDG_ISR）

- 名称：WWDG Interrupt Status Register
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:1	0
名称		EWIF
访问		R/W1C
复位值		0x0

字段	说明
[0]	**WWDG 提前唤醒中断标志位（WWDG Early Wakeup Interrupt Flag）**
EWIF	0：No Pending
	1：Pending

15 DMA 控制器（DMA）

15.1 简介

直接存储器访问(DMA)用于在外设与存储器之间及存储器与存储器之间提供高速数据传输，可以在无需任何 CPU 操作的情况下通过 DMA 实现数据快速搬运，这样可以很大程度地节省 CPU 资源，减少中断进入次数，最终提高整体系统的性能。

DMA 控制器基于复杂的总线矩阵架构，DMA 控制器包含独立的 FIFO 和双 AHB 主总线架构，优化了系统带宽，有效地实现数据传输。

DMA 控制器有两个通道，每个通道总共可以有多达 8 个请求，两个通道可以分配给一个或多个特定的外围设备进行数据传输，每个通道都有一个仲裁器，用于处理 DMA 请求间的优先级。

15.2 结构框图

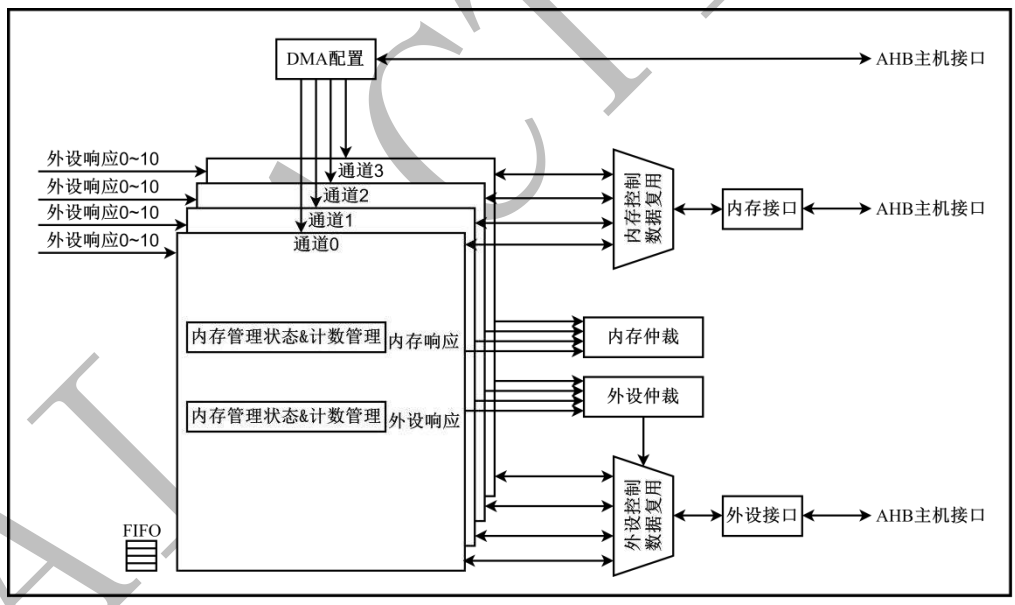


图 15-1　DMA 结构框图

15.3 主要特性

DMA 主要具有以下特性：

- 包含 4 个通道，每个通道可连接 16 个特定的外设请求
- 存储器和外围设备支持单一传输，4 拍、8 拍和 16 拍增量突发传输
- 支持外设 DMA：2 个 PDM，2 个 SPI，2 个 UART，1 个 I2C
- 当外围设备向存储器发送数据时支持改变存储器
- 支持对所有内部存储的 DMA 访问
- 支持优先级(通道数越低，优先级越高)
- 存储器和外设的数据传输宽度可配置为：字节/半字/字
- 存储器和外设的数据传输支持固定和增量寻址
- 每个通道有 3 种类型的事件及独立中断

15.4 功能描述

15.4.1 DMA 传输

DMA 控制器执行直接存储传输：DMA 传输接口采用 AHB 总线，可以控制 AHB 总线矩阵来启动 DMA 传输事务。

DMA 传输事务主要包括以下几项：

- 外设到存储器的传输
- 存储器到外设的传输
- 存储器到存储器的传输
- 外设到外设的传输

DMA 控制器提供两个 AHB 端口：AHB 存储器端口(连接存储器 SRAMA/B)和 AHB 外设端口。

15.4.2 DMA 事务

一个 DMA 事务(Burst)由一串数据传输序列组成，需传输数据项的数目及宽度(8 位、16 位或 32 位)可由软件配置。

每个 DMA 传输都包含以下操作：

- 通过 DMA_SAR 寄存器寻址，从外设数据寄存器或存储器单元中加载数据；
- 通过 DMA_DAR 寄存器寻址，将加载的数据存储到外设数据存储器或存储器单元。

15.4.3 DMA 通道

DMA 模块共包含 4 个通道，软件可以自由选择通道作为 DMA 传输通道。

DMA 每个通道可软件自由配置源端/目标端的外设握手接口。通过配置相应通道的 DMA 控制

寄存器的接口索引位(DMA_CTR.SHF/DHF)，选择源外设/目标外设的外设握手接口源。

表 15-1 给出了 DMA 握手接口映射。

表 15-1　DMA 握手接口映射表

DMA_CTR.SHF/DHF	连接外设请求信号	说明
DMA HS Interface 0	I2C0_TX_REQ	-
DMA HS Interface 1	I2C0_RX_REQ	-
DMA HS Interface 2	UART0_TX_REQ	-
DMA HS Interface 3	UART0_RX_REQ	-
DMA HS Interface 4	UART1_TX_REQ	-
DMA HS Interface 5	UART1_RX_REQ	-
DMA HS Interface 6	SPI0_TX_REQ	-
DMA HS Interface 7	SPI 0_RX_REQ	-
DMA HS Interface 8	SPI 1_TX_REQ	-
DMA HS Interface 9	SPI 1_RX_REQ	-
DMA HS Interface 11	PDM0_RX_REQ	-
DMA HS Interface 13	PDM1_RX_REQ	-

说明：如果源端/目标端配置的是 Memory，则该配置无效。

15.4.4　DMA 仲裁

仲裁器为两个 AHB 主端口（存储器和外设端口）提供基于请求优先级的 4 个 DMA 数据请求管理，并启动外设/存储器访问序列。

优先级管理分为两种：

- 软件：每个数据流优先级都可以在相应通道的 DMA 控制寄存器的优先级配置位(DMA_CTR.PRI)进行配置，4 个通道分为 4 个级别：
  - 高优先级(0x3)
  - 低优先级(0x0)
- 硬件：如果两个请求具有相同的软件优先级，则编号低的通道优先于编号高的通道。例如，通道 0 的优先级高于通道 1。

15.4.5 DMA 源、目标和传输模式

源传输和目标传输在整个 4GB 区域(地址在 0x0000 0000 和 0xFFFF FFFF 之间)都可以寻址外设和存储器。

传输方向可在相应通道的 DMA 控制寄存器的传输类型选择位(DMA_CTR.TC)进行配置，有四种可能的传输方向：存储器到外设、外设到存储器、存储器到存储器、外设到外设，下表介绍了四种传输类型。

表 15-2　传输类型

DMA_CTR.TC	传输类型
00	存储器到存储器(M2M)
01	存储器到外设(M2P)
10	外设到存储器(P2M)
11	外设到外设(P2P)

当数据宽度（由相应通道的 DMA 控制寄存器的传输数据位宽(DMA_CTR.SDS/DDS)来控制）分别是半字或字时，写入该 DMA 通道的 DMA_SAR 或 DMA_DAR 寄存器的外设或存储器地址必须分别按字或半字地址的边界对齐。

存储器到存储器模式

当使能这种模式时，每次产生外设请求，DMA 都会启动数据源到 FIFO 的传输。达到 FIFO 的阈值级别时，将 FIFO 内容移出并存储到目标中。如果 DMA 使能位由软件清零或传输完成即会停止。

只有获得仲裁的通道才有权访问 AHB 的源或目标端口，软件使用可以通过配置相应通道的 DMA 控制寄存器的优先级配置位(DMA_CTR.PRI)为相应的通道定义优先级。

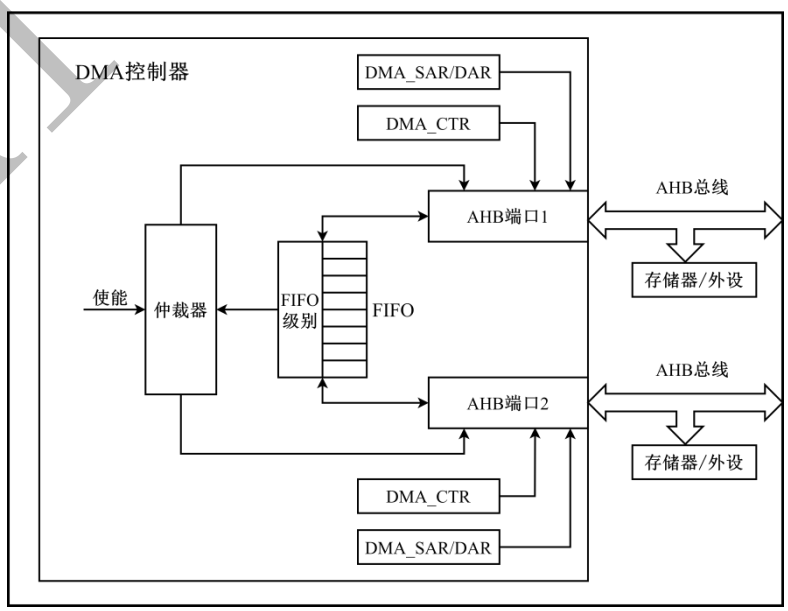


图 15-2　存储器到存储器模式示意图

外设到存储器模式

当使能这种模式时，每次产生外设请求，DMA 都会启动数据源到 FIFO 的传输。达到 FIFO 的阈值级别时，将 FIFO 内容移出并存储到目标中。如果 DMA 使能位由软件清零或传输完成即会停止。

只有获得仲裁的通道才有权访问 AHB 的源或目标端口，软件使用可以通过配置相应通道的 DMA 控制寄存器的优先级配置位(DMA_CTR.PRI)为相应的通道定义优先级。

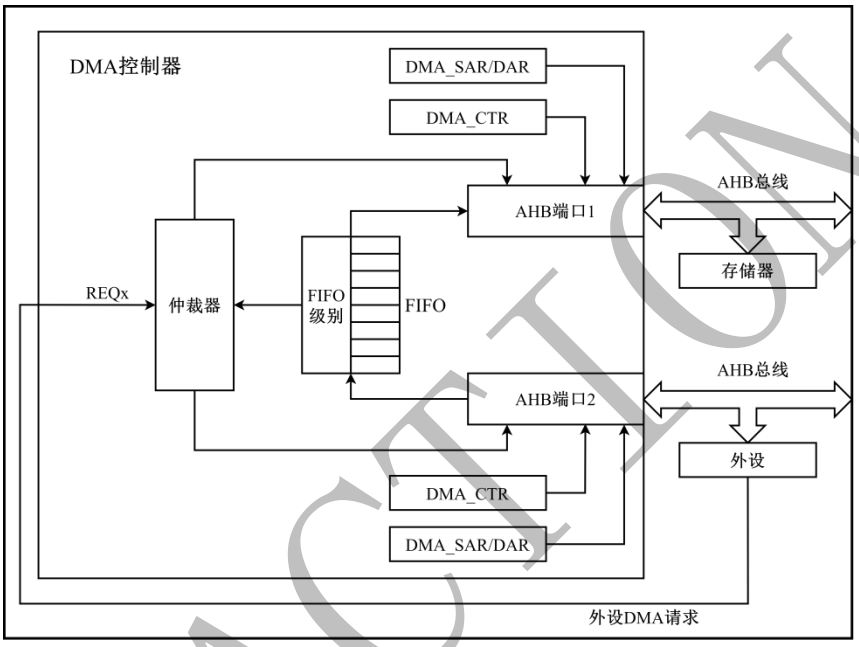


图 15-3　外设到存储器模式示意图

存储器到外设模式

当使能这种模式时，每次产生外设请求，DMA 都会启动数据源到 FIFO 的传输。达到 FIFO 的阈值级别时，将 FIFO 内容移出并存储到目标中。如果 DMA 使能位由软件清零或传输完成即会停止。

只有获得仲裁的通道才有权访问 AHB 的源或目标端口，软件使用可以通过配置相应通道的 DMA 控制寄存器的优先级配置位(DMA_CTR.PRI)为相应的通道定义优先级。

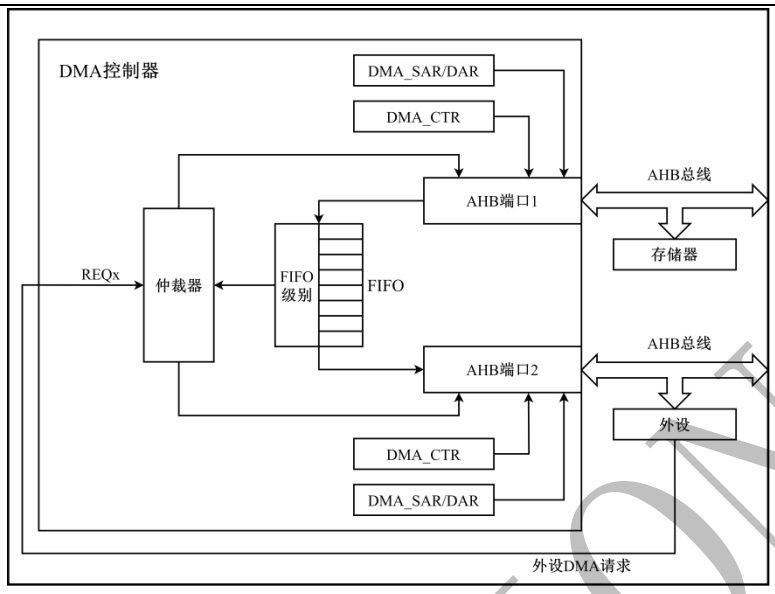


图 15-4　存储器到外设模式示意图

外设到外设模式

当使能这种模式时，每次产生外设请求，DMA 都会启动数据源到 FIFO 的传输。达到 FIFO 的阈值级别时，将 FIFO 内容移出并存储到目标中。如果 DMA 使能位由软件清零或传输完成即会停止。

只有获得仲裁的通道才有权访问 AHB 的源或目标端口，软件使用可以通过配置相应通道的 DMA 控制寄存器的优先级配置位(DMA_CTR.PRI)为相应的通道定义优先级。

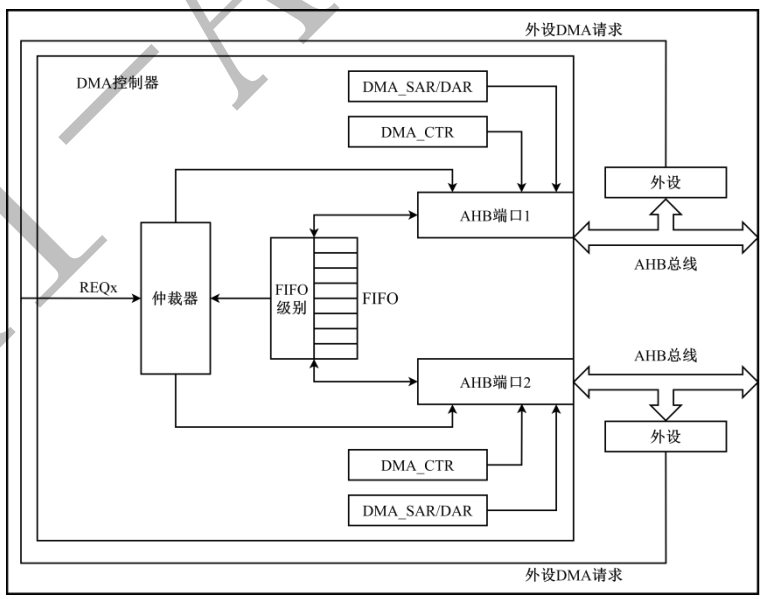


图 15-5　外设到外设模式示意图

15.4.6 DMA 地址控制

根据 DMA 控制寄存器中地址控制位(DMA_CTR.SAI/DAI)的状态，外设和存储器指针在每次传输后可以自动向后递增、递减或保持常量，如果只通过单个寄存器访问外设源或目标数据时，可以禁止递增模式。

如果使能了递增模式，则根据 DMA 控制寄存器的传输数据位宽(DMA_CTR.SDS/DDS)中配置的数据宽度，下一次传输的地址将是前一次传输的地址递增 1(对于字节)、2(对于半字)或 4(对于字)。

为了优化传输操作，可以不管 AHB 外设端口数据位宽，将外设地址的增量偏移大小固定下来。例如通过 DMA 控制寄存器的传输数据位宽(DMA_CTR.SDS/DDS)将增量偏移大小与外设 AHB 端口固定为 32 位地址，此时地址递增 4，需确保源/目的地址按 32 位进行对齐。

FIFO 用于存储在源数据传输到目标之前的临时数据，每个通道都有一个独立的 FIFO，当 FIFO 内部数据达到一笔传输阈值时即启动传输。

15.4.7 DMA 中断

DMA 中每个通道都有 3 种类型的状态/中断：

- Transfer Finish：在 DMA 完成所有数据传输后，产生传输完成标志，挂起相应的中断请求标志，并关闭 DMA 的通道使能。
- Transfer Half Finish：在 DMA 块传输完成一半数据传递到目标端后，产生块传输完成标志，并挂起相应的中断请求标志。
- Transfer Error：在传输过程中由于传输出现异常时，产生传输错误标志，挂起相应的请求标志并且取消 DMA 传输、关闭通道使能。

15.5 寄存器描述

15.5.1 寄存器列表

DMA 基地址：**0x4002 0000**

偏移	实例地址	名称	默认值	描述
0x00	0x40020000+N*0x20[N=0-3]	DMA_SAR	0x00000000	DMA 通道 n 源地址寄存器
0x04	0x40020004+N*0x20[N=0-3]	DMA_DAR	0x00000000	DMA 通道 n 目标地址寄存器
0x08	0x40020008+N*0x20[N=0-3]	DMA_DBL	0x00000000	DMA 通道 n 传输数量长度寄存器
0x0C	0x4002000C+N*0x20[N=0-3]	DMA_CTR	0x00000000	DMA 通道 n 控制寄存器
0x10	0x40020010+N*0x20[N=0-3]	DMA_CER	0x00000000	DMA 通道 n 通道使能寄存器
0x14	0x40020014+N*0x20[N=0-3]	DMA_STR	0x00000000	DMA 通道 n 状态寄存器

15.5.2 寄存器详细描述

15.5.2.1 DMA 源地址寄存器（DMA_SAR）

- **名称：DMA Source Address Register**
- **偏移地址：0x00+N*0x20[N=0-3]**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:0
名称	SAR
访问	R/W
复位值	0x0

字段	说明
[31:0] SAR	**当前 DMA 传输的源地址（DMA Source Address）** 注意： ● DMA 传输地址是否进行递增、或保持不变由寄存器 DCTRL[10:9] (Source Address Increase)位决定 ● SAR 地址必须与 Source Data Width 保持对齐 ● DMA 通道使能由软件配置源地址

15.5.2.2 DMA 目标地址寄存器（DMA_DAR）

- **名称：DMA Destination Address Register**
- **偏移地址：0x04+N*0x20[N=0-3]**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:0
名称	DAR
访问	R/W
复位值	0x0

字段	说明
[31:0] DAR	**当前 DMA 传输的目标地址（DMA Destination Address）** 注意： ● DMA 传输地址是否进行递增、递减或者保持不变由寄存器 DCTRL[8:7] (Destination Address Increase)位决定； ● DAR 地址必须与 Destination Data Width 保持对齐； ● DMA 通道使能由软件配置源地址

15.5.2.3 DMA 传输数量长度寄存器（DMA_DBL）

- 名称：DMA Block Length Register
- 偏移地址：0x08+N*0x20[N=0-3]
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:0
名称		BL
访问		R/W
复位值		0x0

字段	说明
[11:0]	**DMA 传输数量长度（DMA Block Length）**
BL	Note：总数据量=BL* DataSize；

15.5.2.4 DMA 控制寄存器（DMA_CTR）

- 名称：DMA Control Register
- 偏移地址：0x0C+N*0x20[N=0-3]
- 默认值：0x00000000
- 返回寄存器列表

位	31:30	29	28	27:26	25:24	23:20	19:16	15:14	13:12
名称		DMS	SMS	DBL	SBL	DHF	SHF	DDS	SDS
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	11	10	9	8	7	6	5:4	3	2:0
名称		CIE	HCIE	EIE	DAI	SAI	TC		PRI
访问		R/W	R/W	R/W	R/W	R/W	R/W		R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0		0x0

字段	说明
[29]	**DMA 目的端设备接口控制位（DMA Destination Master Select）**
DMS	参考目标设备所在的总线； 0:AHB Master1 1:AHB Master2

[28] SMS	**DMA 源端设备接口控制位（DMA Source Master Select）** 参考目标设备所在的总线； 0:AHB Master1 1:AHB Master2
[27:26] DBL	**DMA 目的端 Burst 长度设置（DMA Destination Burst Length）** 0: 1　1: 4 2: 8　3: 16
[25:24] SBL	**DMA 源端 Burst 长度设置（DMA Source Burst Length）** 0: 1　1: 4 2: 8　3: 16
[23:20] DHF	**DMA 目的端硬件握手接口索引（DMA Destination Handshake Index）** (参考下面描述)
[19:16] SHF	**DMA 源端硬件握手接口索引（DMA Source Handshake Index）** 0:I2C0Tx　1:I2C0RX　2:UART0Tx　3:UART0Rx　4:UART1Tx 5:UART1RX　6:SPI0Tx　7:SPI0Rx　8:SPI1Tx　9:SPI1RX 10:xx　11:PDM0Rx　12:xx　13:PDM1RX
[15:14] DDS	**DMA 目的端传输位宽选择（DMA Destination Data Length Select）** 0: Byte　1: HalfWord 2: Word　x: Reseved
[13:12] SDS	**DMA 源端传输位宽选择（DMA Source Data Length Select）** 0: Byte　1: HalfWord　2: Word　x: Reseved Note:单次传输数据长度为 BL*SDS；
[10] CIE	**DMA 传输完成中断使能位（DMA Transfer Complete Interrupt Enable）** 0:不使能 1:使能
[9] HCIE	**DMA 传输一半完成中断使能位（DMA Transfer half Completed Interrupt Enable）** 0:不使能 1:使能
[8] EIE	**DMA 错误中断使能位（DMA Error Interrupt Enable）** 0:不使能 1:使能
[7] DAI	**DMA 目的端地址控制位（DMA Destination Address Control）** 0:递增 1:不变 Note: 该位指示每次传输完成时目的地址是否递增或者保持不变；如果设备正在向固定地址或者外设 FIFO 写数据，此字段设置为“不变”。
[6] SAI	**DMA 源端地址控制位（DMA Source Address Control）** 0:递增 1:不变 Note: 该位指示每次传输完成时源地址是否递增或者保持不变；如果设备正在向固定地址或者外设 FIFO 取数据，此字段设置为“不变”。
[5:4] TC	**DMA 传输类型控制（DMA Transfer Class Control）** 0:Memory 到 Memory　1:Memory 到外设 2:外设到 Memory　3:外设到外设

[2:0]	**DMA 通道优先级设置（DMA Channel Priority Control）(根据通道数目)**
PRI	0: 低优先级
	7: 高优先级
	Note:
	1、7 是最高优先级，0 是最低优先级。
	2、字段必须在 0~7 范围内设置，如果超出这个范围会出错；
	3、复位值是通道值；例如，通道 0 的情况下复位值为 0；通道 1 的情况下复位值为 1。

15.5.2.5 DMA 通道使能寄存器（DMA_CER）

- **名称：DMA Channel Enable Register**
- **偏移地址：0x10+N*0x20[N=0-3]**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:1	0
名称		CEN
访问		R/W
复位值		0x0

字段	说明
[0]	**DMA 传输使能（DMA Enable）**
CEN	0:关闭
	1:使能

15.5.2.6 DMA 状态寄存器（DMA_STR）

- 名称：DMA Status Register
- 偏移地址：0x14+N*0x20[N=0-3]
- 默认值：0x00000000
- 返回寄存器列表

位	31:3	2	1	0
名称		CS	HCS	ES
访问		R/W1C	R/W1C	R/W1C
复位值		0x0	0x0	0x0

字段	说明
[2] CS	**DMA 传输完成中断状态位（DMA Transfer Complete Interrupt Status）** 0: DMA 传输未完成 1: DMA 传输完成
[1] HCS	**DMA 传输一半完成中断状态位（DMA Transfer half Completed Interrupt Status）** 0: DMA 传输一半未完成 1: DMA 传输一半完成
[0] ES	**DMA 错误中断状态位（DMA Error Interrupt Status）** 0: DMA 未出现错误 1: DMA 出现错误

16 中断和事件

16.1 嵌套向量中断控制器（NVIC）

16.1.1 NVIC 特性

嵌套向量中断控制器 NVIC 包含以下特性：

- 芯片具有 57 个可屏蔽中断通道（不包括 Cortex™-M3 的 16 根中断线）
- 16 个可编程优先级（使用了 4 位中断优先级）
- 低延迟异常和中断处理
- 电源管理控制
- 系统控制寄存器的实现

嵌套向量中断控制器(NVIC)和处理器内核接口紧密配合，可以实现低延迟的中断处理和晚到中断的高效处理。

包括内核异常在内的所有中断均通过 NVIC 进行管理。更多关于异常和 NVIC 编程的说明，请参考《ARM Cortex™-M3 技术参考手册》中的第 16.1.3 节：中断和异常向量和第 16.1 节：嵌套向量中断控制器。

16.1.2 Sys Tick 校准值寄存器

SysTick 校准值设置为 200000，当 SysTick 时钟设置为 5MHz，会产生 40 ms 时间基准，SysTick 校准值的默认值为 0，需要用户写入。

16.1.3 中断和异常向量

请参见表 16-1，了解芯片的中断向量表。

表 16-1 芯片的中断向量表

位置	优先级	优先级类型	名称	说明	地址
	-	-	-	保留	0x0000 0000
	-3	固定	Reset	复位	0x0000 0004
	-2	固定	NMI	不可屏蔽中断。RCU 时钟安全系统(CSS)连接到 NMI 向量。	0x0000 0008
	-1	固定	HardFault	所有类型的错误	0x0000 000C
	0	可设置	MemManage	存储器管理	0x0000 0010
	1	可设置	BusFault	预取指失败，存储器访问失败	0x0000 0014
	2	可设置	UsageFault	未定义的指令或非法状态	0x0000 0018

	-	-	-	保留	0x0000 001C - 0x0000 002B
	3	可设置	SVCall	通过 SWI 指令调用的系统服务	0x0000 002C
	4	可设置	Debug Monitor	调试监控器	0x0000 0030
	-	-	-	保留	0x0000 0034
	5	可设置	PendSV	可挂起的系统服务	0x0000 0038
	6	可设置	SysTick	系统嘀嗒定时器	0x0000 003C
0	7	可设置	I2C0	I2C0 全局中断	0x0000 0040
1	8	可设置	UART0	UART0 全局中断	0x0000 0044
2	9	可设置	UART1	UART1 全局中断	0x0000 0048
3	10	可设置	SPI0	SPI0 全局中断	0x0000 004C
4	11	可设置	SPI1	SPI1 全局中断	0x0000 0050
5	12	可设置	CAN	CAN 全局中断	0x0000 0054
6	13	可设置	DMA	DMA 全局中断	0x0000 0058
7	14	可设置	TMR0	TMR0 全局中断	0x0000 005C
8	15	可设置	TMR1	TMR1 全局中断	0x0000 0060
9	16	可设置	TMR2	TMR2 全局中断	0x0000 0064
10	17	可设置	TMR3	TMR3 全局中断	0x0000 0068
11	18	可设置	TMR4	TMR4 全局中断	0x0000 006C
12	19	可设置	TMR5	TMR5 全局中断	0x0000 0070
13	20	可设置	TMR6	TMR6 全局中断	0x0000 0074
14	21	可设置	TMR7	TMR7 全局中断	0x0000 0078
15	22	可设置	IWDG	IWDG 全局中断	0x0000 007C
16	23	可设置	WWDG	WWDG 全局中断	0x0000 0080
17	24	可设置	LVD	可编程低电压检测中断	0x0000 0084
18	25	可设置	GPIOA	GPIOA 全局中断	0x0000 0088
19	26	可设置	GPIOB	GPIOB 全局中断	0x0000 008C
20	27	可设置	GPIOC	GPIOC 全局中断	0x0000 0090
21	28	可设置	GPIOD	GPIOD 全局中断	0x0000 0094
22	29	可设置	EFLASH	FLASH 全局中断	0x0000 0098
23	30	可设置	DFLASH	DFLASH 全局中断	0x0000 009C
24	31	可设置	QEI0	QEI0 全局中断	0x0000 00A0
25	32	可设置	QEI1	QEI1 全局中断	0x0000 00A4
26	33	可设置	DAC0	DAC0 全局中断	0x0000 00A8
27	34	可设置	DAC1	DAC1 全局中断	0x0000 00AC
28	35	可设置	DAC2	DAC2 全局中断	0x0000 00B0
29	36	可设置	CMP0	CMP0 全局中断	0x0000 00B4
30	37	可设置	CMP1	CMP1 全局中断	0x0000 00B8
31	38	可设置	CMP2	CMP2 全局中断	0x0000 00BC
32	39	可设置	PDM0	PDM0 全局中断	0x0000 00C0
33	40	可设置	PDM1	PDM1 全局中断	0x0000 00C4
34	41	可设置	EPWM0	EPWM0 全局中断	0x0000 00C8
35	42	可设置	EPWM1	EPWM1 全局中断	0x0000 00CC

36	43	可设置	EPWM2	EPWM2 全局中断	0x0000 00D0
37	44	可设置	EPWM3	EPWM3 全局中断	0x0000 00D4
38	45	可设置	EPWM4	EPWM4 全局中断	0x0000 00D8
39	46	可设置	EPWM5	EPWM5 全局中断	0x0000 00DC
40	47	可设置	EPWM6	EPWM6 全局中断	0x0000 00E0
41	48	可设置	EPWM_TZ0	EPWM0 故障中断	0x0000 00E4
42	49	可设置	EPWM_TZ1	EPWM1 故障中断	0x0000 00E8
43	50	可设置	EPWM_TZ2	EPWM2 故障中断	0x0000 00EC
44	51	可设置	EPWM_TZ3	EPWM3 故障中断	0x0000 00F0
45	52	可设置	EPWM_TZ4	EPWM4 故障中断	0x0000 00F4
46	53	可设置	EPWM_TZ5	EPWM5 故障中断	0x0000 00F8
47	54	可设置	EPWM_TZ6	EPWM6 故障中断	0x0000 00FC
48	55	可设置	ADCINT0	ADCINT0 中断	0x0000 0100
49	56	可设置	ADCINT1	ADCINT1 中断	0x0000 0104
50	57	可设置	ADCINT2	ADCINT2 中断	0x0000 0108
51	58	可设置	ADCINT3	ADCINT3 中断	0x0000 010C
52	59	可设置	ADCINT4	ADCINT4 中断	0x0000 0110
53	60	可设置	ADCINT5	ADCINT5 中断	0x0000 0114
54	61	可设置	ADCINT6	ADCINT6 中断	0x0000 0118
55	62	可设置	ADCINT7	ADCINT7 中断	0x0000 011C
56	63	可设置	ADCINT8	ADCINT8 中断	0x0000 0120

16.1.4 唤醒事件

芯片能够处理外部或内部事件来唤醒内核(WFE)。唤醒事件可通过以下方式产生：

- 在外设的控制寄存器使能一个中断，但不在 NVIC 中使能，同时使能 Cortex™-M3 系统控制寄存器中的 SEVONPEND 位。当 CPU 从 WFE 恢复时，需要清除相应外设的中断挂起位和外设 NVIC 中断通道挂起位（在 NVIC 中断清除挂起寄存器中）。
- 配置一个外部或内部中断为事件模式，当 CPU 从 WFE 恢复时，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或 NVIC 中断通道挂起位。

17 CORDIC 运算单元（CORDIC）

17.1 简介

芯片内置一个 32 位 CORDIC 运算单元，可通过其实现基本 CORDIC 算法。CORDIC 算法是一个迭代过程，每次计算进行 N 次迭代，(N/2) + 2 个周期内完成，包含预处理后处理时间（N=16），可用来解圆函数（三角函数）等。
CORDIC 单元可实现基于圆函数的旋转与向量运算，尤其可快速完成高精度的正余弦运算。

17.2 主要特性

- 基于 CORDIC 算法（圆函数）实现旋转模式和向量模式
- CORDIC 计算采样预处理及后处理逻辑，可支持圆函数的全范围取值
- CORDIC 计算支持 IQ26/IQ24/IQ15 三种预置数据格式
- CORDIC 支持运算结果的 K 因子自动消除
- 一个计算引擎，两套控制和操作数寄存器

17.3 功能描述

17.3.1 CORDIC 方程

一般 CORDIC 算法对应如下 CORDIC 方程：

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}$$

$$z_{i=1} = z_i - d_i \cdot e_i$$

旋转模式下 $d_i = sign(z_i), z_i \to 0$

向量模式下 $d_i = sign(y_i), y_i \to 0$

$$e_i = \arctan(2^{-i})$$

17.3.2 CORDIC 运算单元操作

CORDIC 运算单元可工作在旋转模式（Rotate）或向量模式（Vector）下，用来计算圆函数（三角函数），可通过 CORDIC 控制寄存器设置旋转或向量模式。

根据需要执行的函数将相应初始值复制到 X、Y、Z 数据寄存器中后，启动 CORDIC 计算（RUN=1），CORDIC 将数据寄存器内容加载到内核计算寄存器中，通过置位 RUN 启动运算，在计算过程中 RUN 位一直为 1，计算结束后由硬件清零，并自动把计算结果输出到结果计算器。在计算过程中如果强制清除 RUN 位，则当前计算立刻停止，计算结果也不输出到结果寄存器（保持前值）。

计算结束后，若有数据溢出，则 OVF 标志位被置位。

由于 CORDIC 计算结果带有算法固有的 K 因子，如希望消除该因子，可开启 K 因子后处理，也可由用户软件自行计算处理。

CORDIC 计算引擎只有一套，但有两套完全独立的控制和操作数寄存器（组 0 和组 1），两套寄存器可以独立配置工作模式。置位两组寄存器的 RUN 控制位（RUN0 和 RUN1 在两个寄存器地址，无法同时置位，此处假设 RUN0 先置位），则组 0 先运行，组 1 处于等待状态，在组 0 运行结束后立刻切换到组 1 运行。

输入数据寄存器在运行期间无法修改，保证运算数据不会被篡改，但 CORDIC_CRx.MODE 控制位没有做屏蔽，运行期间如果更改 MODE 位会得到错误的结果，需用户软件保证。

一种典型应用是在主流程中使用寄存器组 0 进行运算一项运算（旋转或向量），在中断中使用寄存器组 1 进行运算另一项运算（旋转或向量），由于硬件对两次运算有自动衔接处理，可以减少状态查询和数据输入输出的开销。

17.3.3 数据格式

CORDIC 运算单元的初始数据 X，Y 和 Z 输入都是有符号数 2 补码格式。结果数据也是有符号数 2 补码格式，有一种例外是在向量模式中的 X 结果将会按无符号数存储，这样可防止潜在的 X 数据结果溢出，这种情况下 MSB（最高位）也是数据位。只有工作在向量模式时，X 操作结果始终为正。

X 和 Y 输入数据格式可以是整数或定点数，但在任何计算中 X 和 Y 数据必须保持一致，如果是定点小数，则必须拥有相同个数的小数位数；而对于 Z 数据始终是归一化整数值，在旋转模式下 Z 作为输入数据时代表角度，用 $[-2^N,(2^N-1)]$ 代表 $[-\pi,\frac{(2^N-1)}{2^N}\pi]$，而在向量模式下作为输出数据时也是代表角度，角度的分辨率为 $\frac{\pi}{2^N}$。

- 输入 Z 的初始值=实数 Z 初始值（弧度）$\times\frac{2^N}{\pi}$

- 实数 Z 的结果数据（弧度）=Z 结果数据×$\frac{\pi}{2^N}$

注意：

- *N 可取 15 和 26，分别对应 16 位和 27 位 CORDIC*
- *CORDIC 的计算结果包含一个固有的，有旋转模式或向量模式引起的增益因子 K，在圆函数中 K=1.6467602579*

17.3.4 溢出标志及幅度调节

若在 CORDIC 计算过程中发生了溢出，则会产生 OVF 标志，有一种例外是在圆函数向量模式中的 X 结果数据将固定会按无符号数存储，如计算结果在[0,2^(N+1)]范围内则不会产生溢出标志。另外 CORDIC_CRx.XYMRS 位可用来设置 CORDIC 结果数据的幅度调节，若 XYMRS 置 1 则 CORDIC 迭代结束后，将 X 和 Y 的计算结果除以 2 以后存入 X 和 Y 寄存器中，这样可有效防止结果数据溢出。

17.3.5 操作模式和对应结果数据

CORDIC 方程如下，运算单元操作模式和对应的结果数据见下表，X_{result}，Y_{result}，Z_{result} 为最终数据，X，Y，Z 表示初始值。

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot e_i$$

表 17-1　CORDIC 操作模式和对应结果数据

	旋转模式	**向量模式**
圆函数 $e_i = \arctan(2^{-i})$	$d_i = sign(z_i), z_i \to 0$	$d_i = sign(y_i), y_i \to 0$
	$X_{result} = K[X\cos(Z) - Y\sin(Z)]$ $Y_{result} = K[X\sin(Z) + Y\cos(Z)]$ $Z_{result} = 0$ 其中 K=1.6467602579	$X_{result} = K\sqrt{X^2 + Y^2}$ $Y_{result} = 0$ $Z_{result} = Z + \arctan(Y/X)$ 其中 K=1.6467602579

	设置 $X=1/K, Y=0$ 则 $X_{result}=\cos(Z), Y_{result}=\sin(Z)$ 由于 CORDIC 内部采用了预处理逻辑，所以 X，Y 和 Z 可全范围（$[-2^N,(2^N-1)]$）取值（N 可取 15 和 26）	设置 $X=X/K, Y=Y/K$ 则向量幅值 $X_{result}=\sqrt{X^2+Y^2}$ 由于 CORDIC 内部采用了预处理和后处理逻辑，所以 X，Y 可全范围（$[-2^N,(2^N-1)]$）取值（N 可取 15 和 26） 设置 $Z=0$ 则 $Z_{result}=\arctan(Y/X)$ 由于 CORDIC 内部采用了预处理和后处理逻辑，所以 X，Y 可全范围（$[-2^N,(2^N-1)]$）取值（N 可取 15 和 26），X=0 除外

17.4 寄存器描述

17.4.1 寄存器列表

CORDIC 基地址：**0x4003 E000**

偏移	实例地址	名称	默认值	描述
0x00	0x4003E000	CORDIC_CR0	0x00000000	CORDIC 配置寄存器 0
0x04	0x4003E004	CORDIC_X0	0x00000000	CORDIC 操作数 X/结果数 X 寄存器 0
0x08	0x4003E008	CORDIC_Y0	0x00000000	CORDIC 操作数 Y/结果数 Y 寄存器 0
0x0C	0x4003E00C	CORDIC_Z0	0x00000000	CORDIC 操作数 Z/结果数 Z 寄存器 0
0x20	0x4003E020	CORDIC_CR1	0x00000000	CORDIC 配置寄存器 1
0x24	0x4003E024	CORDIC_X1	0x00000000	CORDIC 操作数 X/结果数 X 寄存器 1
0x28	0x4003E028	CORDIC_Y1	0x00000000	CORDIC 操作数 Y/结果数 Y 寄存器 1
0x2C	0x4003E02C	CORDIC_Z1	0x00000000	CORDIC 操作数 Z/结果数 Z 寄存器 1

17.4.2 寄存器描述

17.4.2.1 CORDIC 配置寄存器 0（CORDIC_CR0）

- 名称：CORDIC Config Register 0
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7	6:5	4	3	2	1	0
名称		MODE	FORMAT		XYMRS	KADJ	OVF	RUN
访问		R/W	R/W		R/W	R/W	R	R/WAC
复位值		0x0	0x0		0x0	0x0	0x0	0x0

字段	说明
[7] MODE	**CORDIC 模式选择位（CORDIC Mode Select）** 0：旋转模式 1：向量模式
[6:5] FORMAT	**CORDIC 运算数据格式（CORDIC Data Format Select）** 0：IQ26 格式　　1：IQ24 格式 2：IQ15 格式　　3：Reserved 注：IQ26/IQ24 都使用 27 位 CORDIC 进行运算，IQ15 使用 16 位 CORDIC 进行运算 运算数据格式并不限于上述三种，用户可以使用其他格式输入，但需要对输入和输出进行调整
[3] XYMRS	**CORDIC 运算 XY 幅度调节选择位（CORDIC XY Amplitude Adjust Select）** 0：最后一次迭代后，X 和 Y 不变作为最终结果 1：最后一次迭代后，X 和 Y 除以 2 作为最终结果 注：正确设置 XYMRS 位，可避免结果数据溢出
[2] KADJ	**CORDIC 运算 K 因子调节选择位（CORDIC K Adjust Select）** 0：运算结果不进行 1/K 调整 1：运算结果乘以 1/K，消除 K 因子 注：K=1.64676
[1] OVF	**CORDIC 运算溢出标志位（CORDIC Overflow Flag）** 由硬件置 1 与清 0 0：无溢出的情况发生 1：有溢出的情况发生 注：运算完成后，该 Flag 保持，直到模块复位或下次运算启动才清除
[0] RUN	**CORDIC 运算启动标志位（CORDIC Run Flag）** 软件置 1，硬件或软件清 0 0：运算已结束或未启动 1：启动 CORDIC 运算，运算过程中标志位 1，运算完成后标志由硬件清 0 注：允许运算过程中清零 RUN 以强行结束运算。RUN 启动时会把 CORDIC 内部 Flag 清零

17.4.2.2 CORDIC 操作数 X/结果数 X 寄存器 0（CORDIC_X0）

- 名称：CORDIC Operand X Register 0
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	OPRDX
访问	R/W
复位值	0x0

字段	说明
[31:0]	**CORDIC X 操作数/X 结果（CORDIC Operand X）**
OPRDX	运算前：存放 CORDIC 运算 X 操作数
	运算完成后：存放 CORDIC 运算 X 结果

17.4.2.3 CORDIC 操作数 Y/结果数 Y 寄存器 0（CORDIC_Y0）

- 名称：CORDIC Operand Y Register 0
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	OPRDY
访问	R/W
复位值	0x0

字段	说明
[31:0]	**CORDIC Y 操作数/Y 结果（CORDIC Operand Y）**
OPRDY	运算前：存放 CORDIC 运算 Y 操作数
	运算完成后：存放 CORDIC 运算 Y 结果

17.4.2.4 CORDIC 操作数 Z/结果数寄存器 0（CORDIC_Z0）

- 名称：CORDIC Operand Z Register 0
- 偏移地址：0x0C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	OPRDZ
访问	R/W
复位值	0x0

字段	说明
[31:0] OPRDZ	**CORDIC Z 操作数/Z 结果（CORDIC Operand Z）** 运算前：存放 CORDIC 运算 Z 操作数 运算完成后：存放 CORDIC 运算 Z 结果

17.4.2.5 CORDIC 配置寄存器 1（CORDIC_CR1）

- 名称：CORDIC Config Register 1
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7	6:5	4	3	2	1	0
名称		MODE	FORMAT		XYMRS	KADJ	OVF	RUN
访问		R/W	R/W		R/W	R/W	R	R/WAC
复位值		0x0	0x0		0x0	0x0	0x0	0x0

字段	说明
[7] MODE	**CORDIC 模式选择位（CORDIC Mode Select）** 0：旋转模式 1：向量模式
[6:5] FORMAT	**CORDIC 运算数据格式（CORDIC Data Format Select）** 0：IQ26 格式　　1：IQ24 格式 2：IQ15 格式　　3：Reserved 注：IQ26/IQ24 都使用 27 位 CORDIC 进行运算，IQ15 使用 16 位 CORDIC 进行运算 运算数据格式并不限于上述三种，用户可以使用其他格式输入，但需要对输入和输出进行调整

[3] XYMRS	**CORDIC 运算 XY 幅度调节选择位（CORDIC XY Amplitude Adjust Select）** 0：最后一次迭代后，X 和 Y 不变作为最终结果 1：最后一次迭代后，X 和 Y 除以 2 作为最终结果 注：正确设置 XYMRS 位，可避免结果数据溢出
[2] KADJ	**CORDIC 运算 K 因子调节选择位（CORDIC K Adjust Select）** 0：运算结果不进行 1/K 调整 1：运算结果乘以 1/K，消除 K 因子 注：K=1.64676
[1] OVF	**CORDIC 运算溢出标志位（CORDIC Overflow Flag）** 由硬件置 1 与清 0 0：无溢出的情况发生 1：有溢出的情况发生 注：运算完成后，该 Flag 保持，直到模块复位或下次运算启动才清除
[0] RUN	**CORDIC 运算启动标志位（CORDIC Run Flag）** 软件置 1，硬件或软件清 0 0：运算已结束或未启动 1：启动 CORDIC 运算，运算过程中标志位 1，运算完成后标志由硬件清 0 注：允许运算过程中清零 RUN 以强行结束运算。RUN 启动时会把 CORDIC 内部 Flag 清零

17.4.2.6 CORDIC 操作数 X/结果数 X 寄存器 1（CORDIC_X1）

- **名称：CORDIC Operand X Register 1**
- **偏移地址：0x24**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:0
名称	OPRDX
访问	R/W
复位值	0x0

字段	说明
[31:0] OPRDX	**CORDIC X 操作数/X 结果（CORDIC Operand X）** 运算前：存放 CORDIC 运算 X 操作数 运算完成后：存放 CORDIC 运算 X 结果

17.4.2.7 CORDIC 操作数 Y/结果数 Y 寄存器 1（CORDIC_Y1）

- 名称：CORDIC Operand Y Register 1
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	OPRDY
访问	R/W
复位值	0x0

字段	说明
[31:0]	**CORDIC Y 操作数/Y 结果（CORDIC Operand Y）**
OPRDY	运算前：存放 CORDIC 运算 Y 操作数
	运算完成后：存放 CORDIC 运算 Y 结果

17.4.2.8 CORDIC 操作数 Z/结果数 Z 寄存器 1（CORDIC_Z1）

- 名称：CCORDIC Operand Z Register 1
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	OPRDZ
访问	R/W
复位值	0x0

字段	说明
[31:0]	**CORDIC Z 操作数/Z 结果（CORDIC Operand Z）**
OPRDZ	运算前：存放 CORDIC 运算 Z 操作数
	运算完成后：存放 CORDIC 运算 Z 结果

18 IQ 除法单元（IQDIV）

18.1 简介

芯片除了具有 Cortex-M3 内核具有的硬件 32 位除法器外，更扩展了专用的 IQ 除法单元，用于加快高精度定点除法运算。

IQDIV 除法器可用硬件快速完成 32 位以内的有/无符号定点除法运算，较之 CPU 软件实现的 IQ 除法运算有极大速度提升。

18.2 主要特性

IQ 格式除法是用整数定标的方法确定小数，从而使浮点运算转换为定点运算，是电机控制中的常用方法。本运算单元直接提供 IQ 格式的除法，提高除法运算的运行效率。

- 支持两组独立的寄存器、可以同时启动
- 支持 Q0~31 格式，根据需要的精度选择合适的 IQ 格式
- 支持有符号运算及无符号运算
- 支持对运算结果自动做四舍五入或截尾处理
- 支持对运算结果的自动饱和处理

18.3 功能描述

18.3.1 IQ 格式除法

进行 IQ 格式除法时，IQDIV_DNDx 用于存放除法操作的被除数：IQDIV_SORx 用于存放除法操作的除数。IQDIV_CRx.QFMT 用于存放除法的格式（即结果的精度，范围为 0~31），结果为一个 32 位的商存放于 IQDIV_RLTx 中。

此除法单元共有两组寄存器，两组寄存器独立设置并启动，但实际处理是串行的，即同时只能处理一组计算配置，另一组必须等上一组处理完后才会执行。假设组 0 启动时正在计算组 1，则组 0 不会马上执行，而是等待组 1 处理完后再执行。在此期间组 0 的 RUN 状态一直为 1。反之如果组 1 启动时正在计算组 0，则组 1 不会马上执行，而是等待组 0 处理完后再执行。在此期间组 1 的 RUN 状态一直为 1。

写除数到寄存器 IQDIV_SORx，写被除数到 IQDIV_DNDx，数据格式到 IQDIV_CRx.QFMT。

寄存器 IQDIV_CRx.SIGN=0 选择无符号除法操作，操作数用原码表示：IQDIV_CRx.SIGN=1 选择有符号除法操作，操作数用补码表示，最高位表示符号位；写 1 到 IQDIV_CRx.RUN 启动运算，此位在运行期间保持为 1 直到运行结束，软件只能写 1 不能写 0（写 0 忽略）。IQDIV_CRx.SAT=0 位为饱和位（即当运算溢出或除 0 后，由硬件自动将结果设为饱和值且将此 FLAG 置为 1），使用中由软件负责清 0。

设置操作参数与启动可在一条指令中完成。

由于除法操作需要多个系统周期执行时间（10 个周期），启动除法后，IQDIV_CRx.RUN 保持为 1 表示未完成，直到结果寄存器有效后自动变为 0，所以软件可以利用此位作为运算是否结束的判断标志。

当硬件完成操作后、会自动将 IQDIV_CRx.RUN 清 0。此时，可以从相应寄存器中读取结果。

读取操作不会改变寄存器中值。在 IQDIV_CRx.RUN 为 1 时任何对寄存器的写入动作都无效，由硬件保证。

运算时由于输入的被除数是 32 位，所以要将输入的 32 位被除数作为 64 位除法的高 32 位，低 32 位补 0。然后除以 32 位除数，得到 64 位的商。然后根据指定的数据格式 IQDIV_CRx.QFMT 对结果进行处理。

对于有符号除法运算，输入的 32 位数是有符号数（包括被除数及除数），以补码表示，结果也应是带符号数。

对于无符号除法运算，输入的 32 位数是无符号数（包括被除数及除数），以原码表示，结果也应是无符号数。

如果结果溢出就置 IQDIV_CRx.SAT 为 1。IQDIV_CRx.RUN 位清 0，结果为饱和值。

如果除数为 0 就置 IQDIV_CRx.SAT 为 1。IQDIV_CRx.RUN 位清 0，结果为饱和值。IQDIV_CRx.CMOD 用作指定在选取结果时是否作四舍五入。如果为 1，会直接丢弃多余的尾数，如果为 0 需要作四舍五入以提高数据的精度。

18.3.2 注意事项

在硬件执行除法操作没有结束时读取运算结果（即读取寄存器 IQDIV_RLTx），会得到不可预知的结果。但是，读取动作不会影响硬件继续运算并得到正确结果（当硬件完成运算后，仍然可以读取到正确的结果）。

如果除法操作中除数为 0，则不会执行除法操作，操作数寄存器保持原有值不变。硬件自动将 IQDIV_CRx.RUN 位清 0 并置位 IQDIV_CRx.SAT 标志，其余位保持不变，不会影响用户程序继续运行。IQDIV_CRx.SAT 异常标志位由软件清 0。

在一组运算未完成时（IQDIV_CRx.RUN=1），任何对于该组寄存器的写入操作都是无效的、另一组不受影响。例如，组 0 的 RUN Flag 为 1 时，对组 0 寄存器的任何写操作都无效。但是可以正常读写组 1 的寄存器。

两组寄存器在使用时是完全独立的，但是执行时还是顺序执行的，即必须等一组计算完后才进行另一组计算，这个等待由硬件完成，无需软件参与。

18.4 寄存器描述

18.4.1 寄存器列表

IQDIV 基地址：**0x4003 F000**

偏移	实例地址	名称	默认值	描述
0x00	0x4003F000	IQDIV_CR0	0x00000000	IQDIV 配置寄存器 0
0x04	0x4003F004	IQDIV_DND0	0x00000000	IQDIV 被除数寄存器 0
0x08	0x4003F008	IQDIV_SOR0	0x00000000	IQDIV 除数寄存器 0
0x0C	0x4003F00C	IQDIV_RLT0	0x00000000	IQDIV 计算结果寄存器 0
0x20	0x4003F020	IQDIV_CR1	0x00000000	IQDIV 配置寄存器 1
0x24	0x4003F024	IQDIV_DND1	0x00000000	IQDIV 被除数寄存器 1
0x28	0x4003F028	IQDIV_SOR1	0x00000000	IQDIV 除数寄存器 1
0x2C	0x4003F02C	IQDIV_RLT1	0x00000000	IQDIV 计算结果寄存器 1

18.4.2 寄存器描述

18.4.2.1 IQDIV 配置寄存器 0（IQDIV_CR0）

- **名称：IQDIV Config Register 0**
- **偏移地址：0x00**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:9	8	7:3	2	1	0
名称		CMOD	QFMT	SIGN	SAT	RUN
访问		R/W	R/W	R/W	R	R/WAC
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[8] CMOD	**结果处理方式选择位（IQDIV Round Mode）** 0：小数直接舍弃 1：小数做四舍五入处理
[7:3] QFMT	**IQ 格式设置（IQDIV IQ Format）** 最大为 31
[2] SIGN	**有符号除法选择（IQDIV Sign Mode）** 0：无符号除法(操作数为原码) 1：有符号除法(操作数为补码)
[1] SAT	**饱和标志（IQDIV Saturation Flag）** ● 如果结果为饱和值就置此位 ● 当遇到除 0，符号位溢出 ● 当 32 位结果溢出，结果寄存器就设为饱和值、且此位设为 1，否则为 0 ● 无符号数除法时饱和值为 0xFFFFFFF ● 当有符号数除法时看结果的符号位（被除数或除数只有一个是负数时，结果为负） ➢ 负数饱和值为 0x80000000 ➢ 正数饱和值为 0x7FFFFFFF
[0] RUN	**IQDIV 运算启动标志位（IQDIV Run Flag）** 软件置 1，硬件或软件清 0 0：运算已结束或未启动 1：启动 IQDIV 运算，运算过程中标志为 1，运算完成后标致由硬件清 0

18.4.2.2 IQDIV 被除数寄存器 0（IQDIV_DND0）

- 名称：IQDIV Dividend Register 0
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DIVDND
访问	R/W
复位值	0x0

字段	说明
[31:0]	**IQDIV 被除数（IQDIV Dividend）**
DIVDND	32 位，当有符号数除法时对应补码，无符号除法时对应原码

18.4.2.3 IQDIV 除数寄存器 0（IQDIV_SOR0）

- 名称：IQDIV Divisor Register 0
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DIVSOR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**IQDIV 除数（IQDIV Divisor）**
DIVSOR	32 位，当有符号数除法时对应补码，无符号除法时对应原码

18.4.2.4 IQDIV 计算结果寄存器 0（IQDIV_RLT0）

- 名称：IQDIV Result Register 0
- 偏移地址：0x0C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	RESULT
访问	R/W
复位值	0x0

字段	说明
[31:0]	**IQDIV 计算结果（IQDIV Result）**
RESULT	32 位，当有符号数除法时对应补码，无符号除法时对应原码

18.4.2.5 IQDIV 配置寄存器 1（IQDIV_CR1）

- 名称：IQDIV Config Register 1
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:9	8	7:3	2	1	0
名称		CMOD	QFMT	SIGN	SAT	RUN
访问		R/W	R/W	R/W	R	R/WAC
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[8]	**结果处理方式选择位（IQDIV Round Mode）**
CMOD	0：小数直接舍弃
	1：小数做四舍五入处理
[7:3]	**IQ 格式设置（IQDIV IQ Format）**
QFMT	最大为 31
[2]	**有符号除法选择（IQDIV Sign Mode）**
SIGN	0：无符号除法(操作数为原码)
	1：有符号除法(操作数为补码)

[1] SAT	**饱和标志（IQDIV Saturation Flag）** ● 如果结果为饱和值就置此位 ● 当遇到除 0，符号位溢出 ● 当 32 位结果溢出，结果寄存器就设为饱和值、且此位设为 1，否则为 0 ● 无符号数除法时饱和值为 0xFFFFFFF ● 当有符号数除法时看结果的符号位（被除数或除数只有一个是负数时，结果为负） ➢ 负数饱和值为 0x80000000 ➢ 正数饱和值为 0x7FFFFFFF
[0] RUN	**IQDIV 运算启动标志位（IQDIV Run Flag）** 软件置 1，硬件或软件清 0 0：运算已结束或未启动 1：启动 IQDIV 运算，运算过程中标志为 1，运算完成后标致由硬件清 0

18.4.2.6 IQDIV 被除数寄存器 1（IQDIV_DND1）

- **名称：IQDIV Dividend Register 1**
- **偏移地址：0x24**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:0
名称	DIVDND
访问	R/W
复位值	0x0

字段	说明
[31:0] DIVDND	**IQDIV 被除数（IQDIV Dividend）** 32 位，当有符号数除法时对应补码，无符号除法时对应原码

18.4.2.7 IQDIV 除数寄存器 1（IQDIV_SOR1）

- 名称：IQDIV Divisor Register 1
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	DIVSOR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**IQDIV 除数（IQDIV Divisor）**
DIVSOR	32 位，当有符号数除法时对应补码，无符号除法时对应原码

18.4.2.8 IQDIV 计算结果寄存器 1（IQDIV_RLT1）

- 名称：IQDIV Result Register 1
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	RESULT
访问	R/W
复位值	0x0

字段	说明
[31:0]	**IQDIV 计算结果（IQDIV Result）**
RESULT	32 位，当有符号数除法时对应补码，无符号除法时对应原码

19 模数转换器（ADC）

19.1 简介

ADC 模块包含一个 13 位循环（Cyclic）型的 ADC。ADC 由内核和外围电路组成。ADC 内核由模拟电路组成，包括前端模拟多路复用器（MUX），采样保持（S/H）电路，采样和保持电路可以单独/连续采样，这些信号由 16 个模拟输入通道传输；ADC 外围电路由配置和控制 ADC 的数字电路组成。

19.2 结构框图

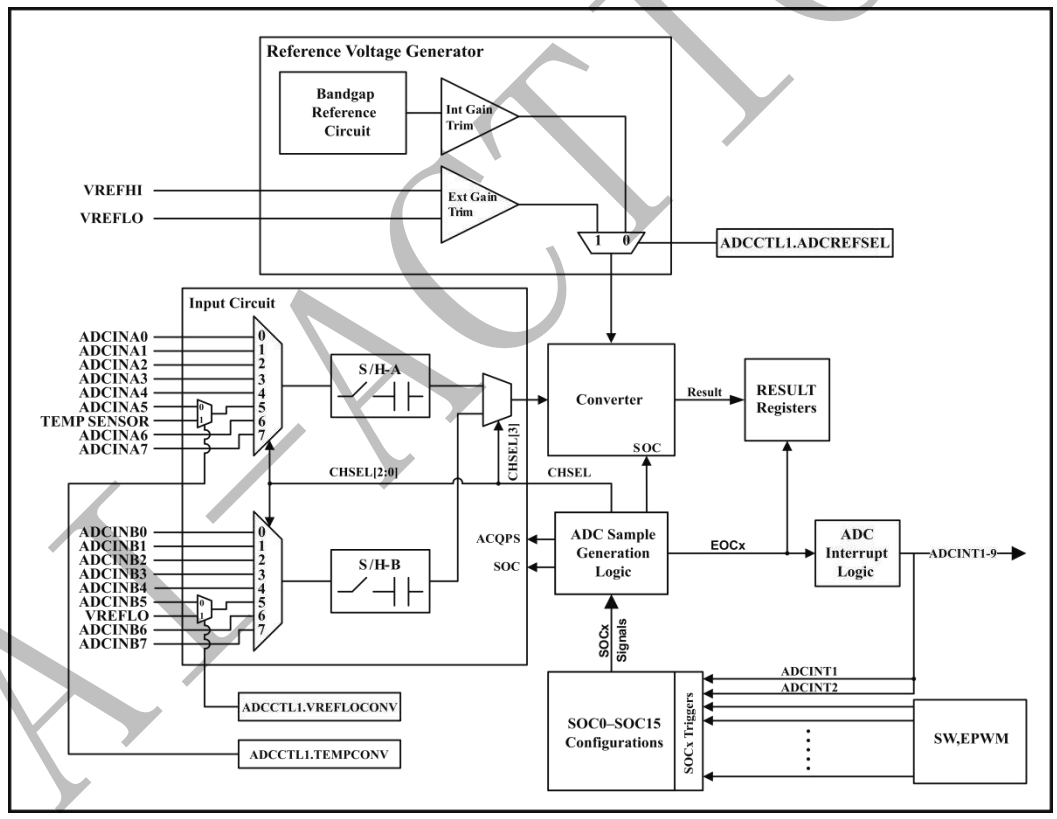


图 19-1　ADC 结构框图

19.3 主要特性

ADC 的主要功能如下：

- 内置双采样保持（S / H）的 13 位 ADC 内核
- 同步采样或连续采样模式
- 全量程模拟输入：固定 0 V 到 3 V，或 VREFLO 到 VREFHI
- 多达 16 通道的多路复用输入
- 16 个 SOC，可配置触发事件、采样窗口和通道
- 16 个结果寄存器(可单独寻址)来存储转换值
- 多个触发源
  - S/W——软件立即启动
  - EPWM 0-6
  - ADCINT 1/2
- 9 个灵活配置的中断，可以在任何转换后配置中断请求

19.4 功能描述

19.4.1 SOC 工作原理

ADC 模块工作原理基于 SOC 机制。每个 SOC 对应于单个通道的单次转换，主要有三种配置：(1)启动转换的触发事件；(2)等待转换的通道；(3)采样窗口大小。每个 SOC 都是独立配置的，可以任意配置触发事件、转换通道和采样窗口大小。可以根据需要为多个 SOC 配置相同的触发事件、转换通道和采样窗口。SOC 机制提供了一种非常灵活的方式来配置通道转换，包括：(1)使用不同触发事件对不同通道进行转换；(2)使用相同触发事件对相同通道进行多次转换；(3)使用相同触发事件对不同通道进行转换等等。

SOCx 的触发事件源由 ADC_SOCxCTL1 寄存器中的 TRIGSEL 和 ADC_INTSOCSEL1 或 ADC_INTSOCSEL2 寄存器中的相应位的组合配置。软件也可以用 ADC_SOCFRC1 寄存器强制执行 SOC 事件。SOCx 的通道和采样窗口大小是由 ADC_SOCxCTL 寄存器的 CHSEL 和 ACQPS 字段配置的。

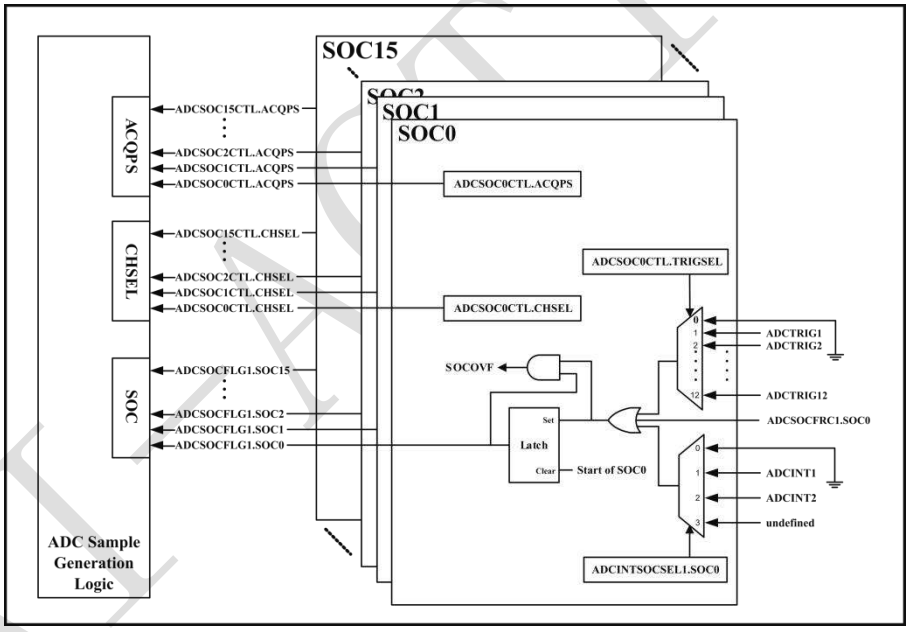


图 19-2　SOC 结构框图

例如，要配置通道 ADCINA1 上的单个转换在 EPWM3 定时器达到其周期匹配时发生，则必须首先设置 EPWM3 以在周期匹配时输出 SOCA 或 SOCB 信号。请参阅增强型脉宽调制器模块(EPWM)了解如何做到这一点。在本例中，我们将使用 SOCA。然后，使用其中一个 SOC 寄存器的 ADC_SOCxCTL 位来设置。选择哪个 SOC 没有区别，因此我们选择使用 SOC0。ADC 允许的最快采样窗口为 6 个周期。选择采样窗口的最快时间，通道 ADCINA1 用于通道转换，EPWM3 为 SOC0 触发事件，我们分别将 ACQPS 为设置为 0，将 CHSEL 位设置为 1，将 TRIGSEL 位设置为 9。写入寄存器的结果值为：

```
ADC_SOC0CTL = 4840h;                    //(ACQPS = 0,CHSEL = 1,TRIGSEL = 9)
```

这样配置时，将在 EPWM3 SOCA 事件中启动 ADCINA1 的单次转换，并将结果值存储在 ADC_RESULT0 寄存器中。

相反，如果需要对 ADCINA1 进行三次采样，则可以为 SOC1，SOC2 和 SOC3 提供与 SOC0 相同的配置。

```
ADC_SOC1CTL = 4840h;        //(ACQPS = 0，CHSEL = 1，TRIGSEL = 9)
ADC_SOC2CTL = 4840h;        //(ACQPS = 0，CHSEL = 1，TRIGSEL = 9)
ADC_SOC3CTL = 4840h;        //(ACQPS = 0，CHSEL = 1，TRIGSEL = 9)
```

这样配置后，将在 EPWM3 SOCA 事件中串行使能 ADCINA1 的四次转换，并将结果值存储在 ADC_RESULT0 – ADC_RESULT3 寄存器中。

另一个应用程序可能需要从同一触发事件中采样 3 个不同的信号。这可以通过简单地更改 SOC0-SOC2 的 CHSEL 位，使 TRIGSEL 位保持不变来完成。

```
ADC_SOC0CTL = 4840h;        //(ACQPS = 0，CHSEL = 1，TRIGSEL = 9)
ADC_SOC1CTL = 4880h;        //(ACQPS = 0，CHSEL = 2，TRIGSEL = 9)
ADC_SOC2CTL = 48C0h;        //(ACQPS = 0，CHSEL = 3，TRIGSEL = 9)
```

通过这种方式配置后，将在 EPWM3 SOCA 事件中串行启动三个转换。通道 ADCINA1 上的转换结果将显示在 ADC_RESULT0 中。通道 ADCINA2 上的转换结果将显示在 ADC_RESULT1 中。通道 ADCINA3 上的转换结果将显示在 ADC_RESULT2 中。转换的通道和触发事件与转换结果的显示位置无关。 RESULT 寄存器与 SOC 相关联。

19.4.1.1 ADC 采样和保持窗口

外部驱动器快速有效地驱动模拟信号的能力各不相同。有些电路需要较长的时间来正确地将电荷转移到 ADC 的采样电容中。为了解决这个问题，ADC 支持对每个单个 SOC 配置的采样窗口长度进行控制。每个 ADC_SOCxCTL 寄存器都有一个 3 位字段 ACQPS，它决定了采样和保持(S/H)窗口大小。

表 19-1　不同 ACQPS 配置下的采样时间

ADC Clock	ACQPS	Sample Window	Conversion Time (30 cycles)	Total Time to Process Analog Voltage
160MHz	0	37.50ns	187.50ns	225.00ns
160MHz	2	262.50ns	187.50ns	450.00ns
160MHz	4	1162.50ns	187.50ns	1350.00ns
160MHz	6	4762.50ns	187.50ns	4950.00ns

注意：总时间是对于一次转换，不包括随时间推移增加平均速度的效果。

如图 19-3 所示，ADCIN 引脚可以建模为 RC 电路。VREFLO 接地后，ADCIN 上从 0 到 3V 的电压摆幅会产生 2ns 的典型 RC 时间常数。

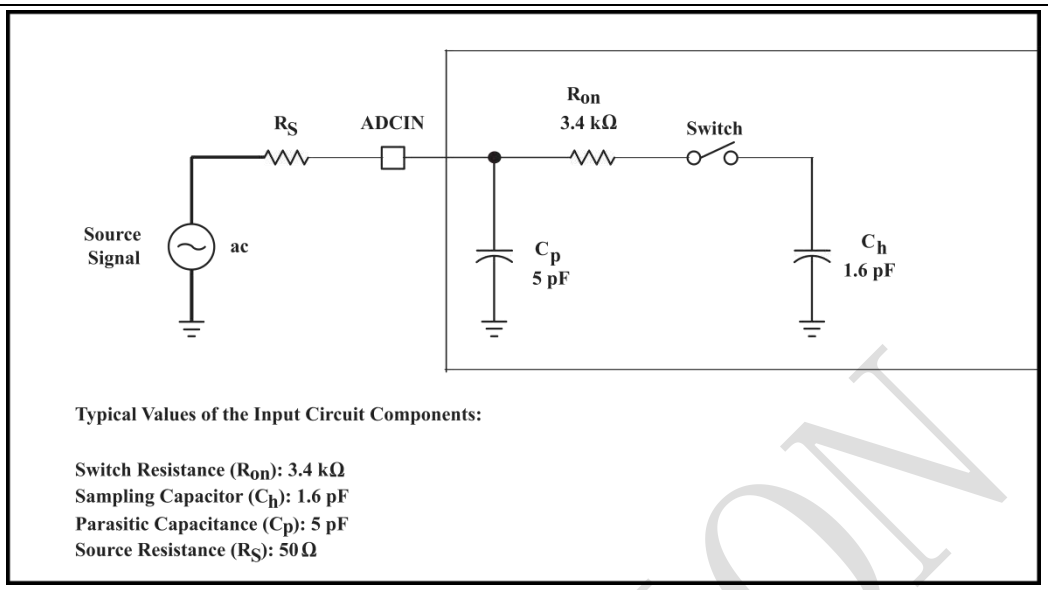


图 19-3　ADCINx 输入模型图

注意：ADC 在转换过程中不会偏置 Ch 电容器的电压，因此存在以下行为：

1. *当 ADCIN 引脚未连接到源信号时，没有预定的 ADC 转换值*
2. *ADC 转换之间的剩余电荷将保留在 Ch 上*
3. *如果 ACQPS 窗口太短而使 Ch 无法稳定，则顺序转换可能会受到串扰*

19.4.1.2　触发事件

可以将每个 SOC 配置为在任一输入触发事件上触发。如果需要可以为同一通道配置多个 SOC。以下是可用输入触发事件的列表：

- 软件
- EPWM0-6 SOCA 和 SOCB

有关这些触发事件的配置详细信息，请参见 ADC_SOCxCTL 寄存器位定义。

另外，ADCINT1 和 ADCINT2 可以反馈触发另一次转换。该配置由 ADC_INTSOCSEL1 / 2 寄存器控制。如果需要连续的转换流，则此模式很有用。

19.4.1.3　通道选择

每个 SOC 可以配置为转换任何可用的 ADCIN 输入通道。将 SOC 配置为连续采样模式时，ADC_SOCxCTL 寄存器的 CHSEL 位定义要转换的通道。将 SOC 配置为同步采样模式时，CHSEL 位的最高有效位将被丢弃，较低的三个位决定哪一对通道被转换。

ADCINA0 与 VREFHI 共享，因此在使用外部参考电压模式时不能作为可变输入源。

19.4.2 单次转换支持

该模式允许在轮转仲裁方案中对下一个触发的 SOC 执行单次转换。单次模式仅对轮转仲裁中存在的通道有效。在轮转仲裁方案中未配置为触发 SOC 的通道将根据 ADC_SOCPRICTL.SOCPRIORITY 的内容获得优先级。

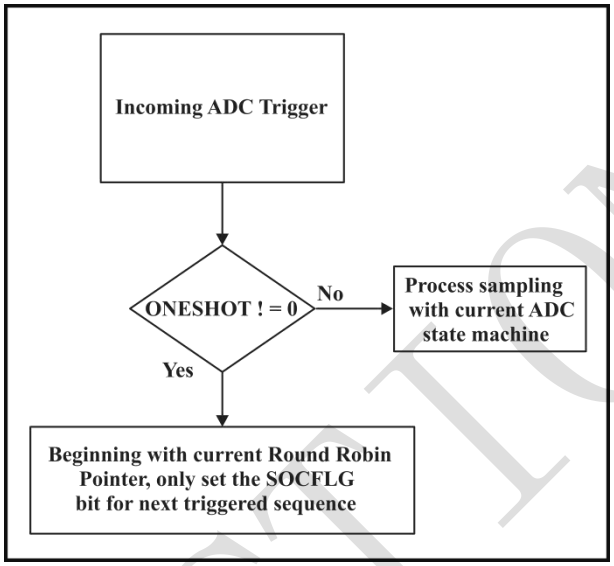


图 19-4 单次转换

下文为单次模式对顺序模式和同步模式的影响。

顺序模式：仅允许在轮转仲裁模式下的下一个有源 SOC（从当前轮转指针向上）来生成 SOC；其他 SOC 的所有其他触发事件将被忽略。

同步模式：如果当前轮转指针具有同时启用的 SOC，有效 SOC 将从当前轮转指针增加 2。这是因为同步模式将为 SOCx 和 SOCx + 1 创建结果，而 SOCx + 1 永远不会被用户触发。

19.4.3 ADC 转换优先级

当同时设置多个 SOC 标志时，两种优先级形式之一确定它们的转换顺序。默认优先级方法是轮转仲裁。在此方案中，没有 SOC 具有比其他任何 SOC 更高的固有优先级。优先级取决于轮转指针（RRPOINTER）。反映在 ADC_SOCPRICTL.RRPOINTER 指向最后转换的 SOC。最高优先级 SOC 被赋予下一个大于 RRPOINTER 值的值，并在 SOC15 之后回绕到 SOC0。复位时该值为 16，0 则表示已经发生了转换。当 RRPOINTER 等于 16 时，SOC0 优先级最高。当写 SOCPRICTL 寄存器时，RRPOINTER 将通过模块复位来复位。

图 19-5 给出了轮转优先级方法的示例。

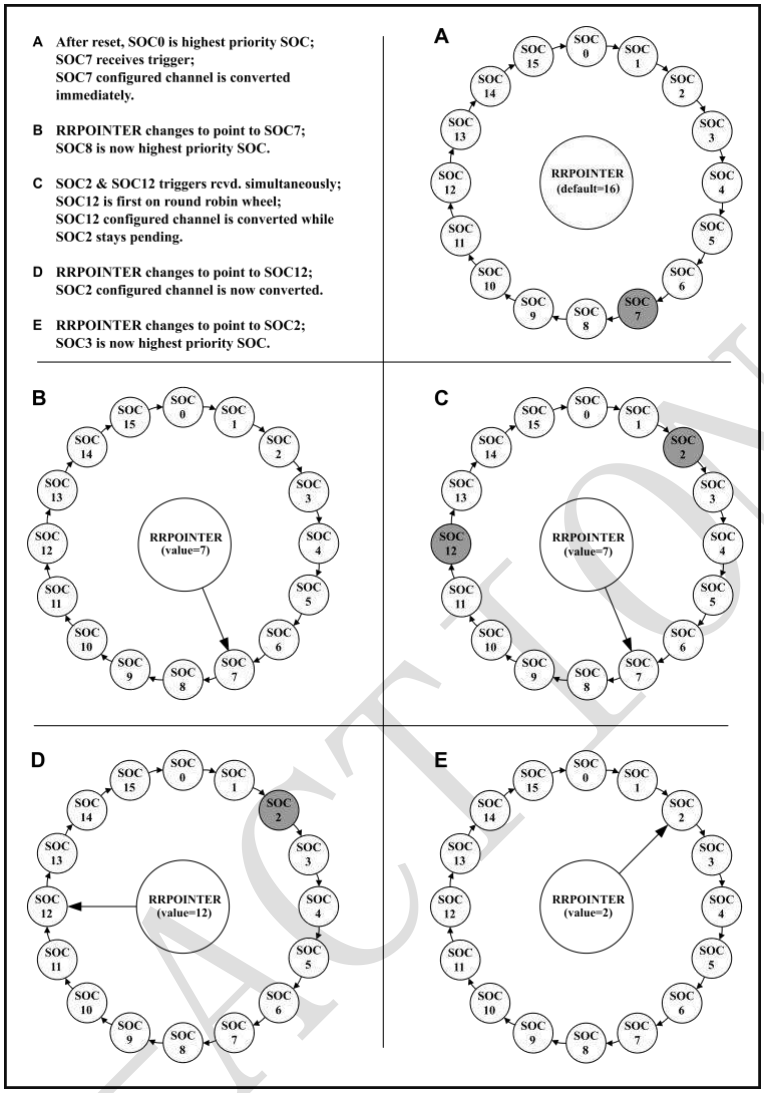


图 19-5　轮转优先级示例

ADC_SOCPRICTL 寄存器中的 SOCPRIORITY 位可用于从单个到所有 SOC 分配高优先级。当配置为高优先级时，SOC 将在任何当前转换完成后中断轮转，并将其插入作为下一次转换。转换完成后，轮转将在中断处继续。如果同时触发两个高优先级 SOC，则编号较低的 SOC 优先。

高优先级模式首先分配给 SOC0，然后按递增的数字顺序分配。写入 SOCPRIORITY 位的值定义了非高优先级的第一个 SOC，即如果将值 4 写入 SOCPRIORITY，则将 SOC0，SOC1，SOC2 和 SOC3 定义为高优先级，其中 SOC0 最高优先级。

图 19-6 给出了一个使用高优先级 SOC 的示例。

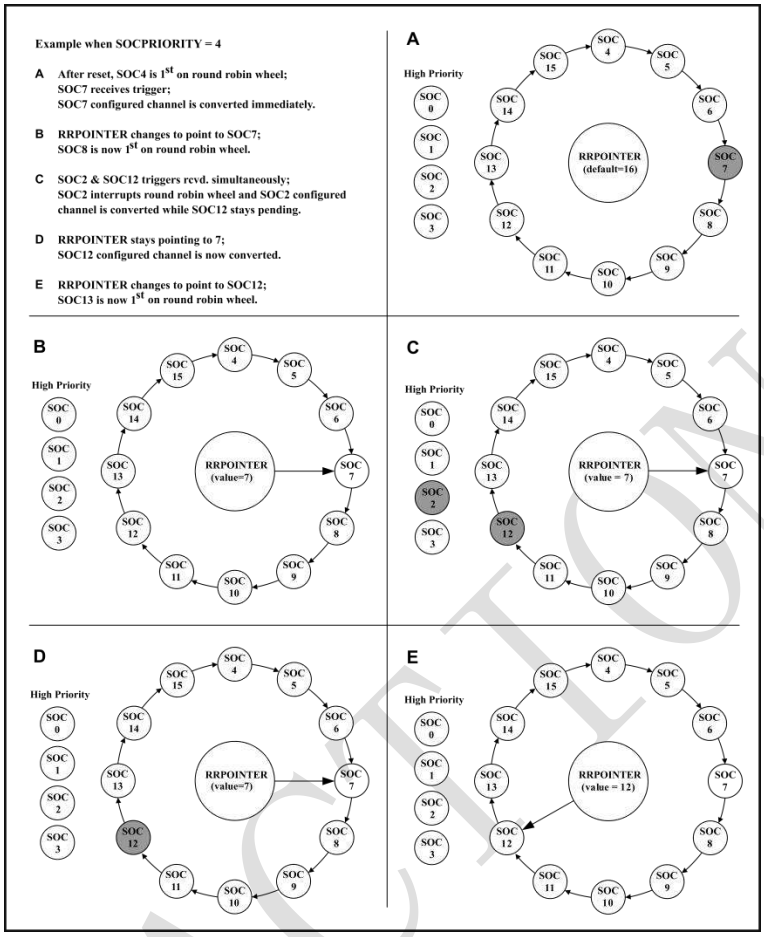


图 19-6　高优先级示例

19.4.4　同步采样模式

在一些应用中，保持两个信号采样之间的延迟最小是很重要的。ADC 包含双采样和保持电路，以允许同时对两个不同的通道进行采样。同时，通过 ADC_SAMPLEMODE 寄存器为一对 SOCx 配置了同步采样模式。

偶数 SOCx 和随后的奇数 SOCx（SOC0 和 SOC1）与一个使能位（在这种情况下为 SIMULEN0）配对在一起。配对动作如下：

- SOCx 的任何一个触发事件都将启动一对采样转换电路。
- 转换的通道对由 A 通道和 B 通道组成，对应触发的 SOCx 的 CHSEL 位的值。取值范围为 0 -7。
- 两个通道将同时采样。
- A 通道将始终首先转换。
- 偶数 EOCx 脉冲将基于 A 通道转换生成，奇数 EOCx 脉冲将基于 B 通道转换生成。
- A 通道转换的结果放在偶数 ADC_RESULTx 寄存器中，而 B 通道转换的结果写到奇数 ADC_RESULTx 寄存器中。

例如，如果 ADC_SAMPLEMODE.SIMULEN0 位置 1，并且 SOC0 的配置如下：
CHSEL = 2(ADCINA2 / ADCINB2 pair)

TRIGSEL = 5(ADCTRIG5 = EPWM1.ADCSOCA)

当 EPWM1 发出 ADCSOCA 触发信号时，将同时对 ADCINA2 和 ADCINB2 进行采样（假定优先级）。紧接着，ADCINA2 通道将被转换，它的值将被存储在 ADC_RESULT0 寄存器中。取决于 ADC_CTL1.INTPULSEPOS 的设置，当 ADCINA2 的转换开始或完成时，将发生 EOC0 脉冲。然后，将转换 ADCINB2 通道并将其值存储在 ADC_RESULT1 寄存器中。取决于 ADC_CTL1.INTPULSEPOS 的设置，当 ADCINB2 的转换开始或完成时，将发生 EOC1 脉冲。

通常在应用程序中，期望仅使用该对中的偶数 SOCx，但是也可以使用奇数 SOCx 或者两者都使用。在后一种情况下，两个 SOCx 触发事件都将开始转换。由于两个 SOCx 会将其结果存储到相同的 ADC_RESULTx 寄存器中，可能会相互覆盖。

SOCx 的优先级规则与顺序采样模式相同。

19.4.5 EOC 和中断

正如有 16 个独立的 SOCx 配置集一样，也有 16 个 EOCx 脉冲。在连续采样模式下，EOCx 直接与 SOCx 相关联。在同步采样模式下，偶数和随后的奇数 EOCx 对与偶数和随后的奇数 SOCx 对相关联。根据 ADC_CTL1.INTPULSEPOS 设置，EOCx 脉冲将在转换开始或结束时发生。

ADC 包含 9 个可标记或传递到 NVIC 的中断。每个中断都可以将任何可用的 EOCx 信号作为信号源。在 INTSELxNy 寄存器中完成了以 EOCx 为源的配置。此外，可以将 ADCINT1 和 ADCINT2 信号配置为生成 SOCx 的触发信号。这有利于创建连续的转化流。

图 19-7 给出了 ADC 中断结构的框图。

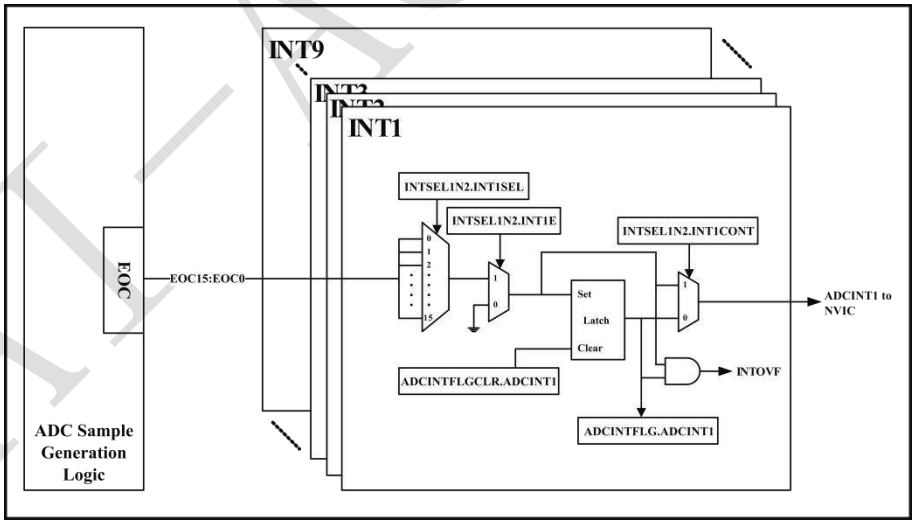


图 19-7　中断结构

19.4.6 上电顺序

ADC 重置为 ADC 关闭状态。在写入任何 ADC 寄存器之前，必须将 PCLKCR0 寄存器中的 ADCENCLK 位置 1。给 ADC 上电时，请按照以下顺序进行：

1. 如果需要外部基准电压源，请使用 ADC_CTL1 寄存器的位 3（ADCREFSEL）使能该模式。
2. 通过将 ADC_CTL1 寄存器中的第 7-5 位（ADCPWDN，ADCBGPWD，ADCREFPWD）置 1，将参考电路，带隙电路和模拟电路一起上电。
3. 通过将 ADC_CTL1 寄存器的第 14 位（ADCENABLE）置 1 来使能 ADC。
4. 在执行第一次转换之前，在步骤 2 之后需要 1 毫秒的延迟。或者，可以同时执行步骤 1 到 3。

关断 ADC 电源时，可以同时清除步骤 2 中的寄存器位（ADCPWDN，ADCBGPWD，ADCREFPWD）。

19.4.7 ADC 校准

任何转换器内都固有零偏置误差和全量程增益误差。ADC 出厂时已进行校准，同时允许用户针对任何应用环境影响（例如环境温度）修改偏置校正。除非在某些模拟条件下或者需要修改出厂设置，否则不需要用户执行任何特定操作。

19.4.7.1 ADC 零点校正

零偏置误差定义为在 VREFLO 电压转换时产生的数字值。这个基本误差会影响 ADC 的所有转换，并与全量程增益和线性规格一起决定转换器的 DC 精度。零偏置误差可以为正或负，即 VREFLO 可以输出正值，也可以为负值，高于 VREFLO 一级以上的电压仍会读为零值。为了纠正这个错误，将错误的二进制补码写入 ADC_SOFFTRIM 寄存器。在 ADC 结果寄存器中得到结果之前，该寄存器中包含的值将被应用。该操作完全包含在 ADC 内核中，因此结果时序不会受到影响，并且对于任何调整值，ADC 的完全动态范围都将保持不变。用户可以修改 ADC_SOFFTRIM 寄存器以补偿由应用环境引起的额外偏置误差。

通过使用 ADC_CTRL1 寄存器中的 VREFLOCONV 位，可以在不占用 ADC 通道的情况下完成此操作。

使用以下步骤重新校准 ADC 偏置误差：

1. 将 ADC_SOFFTRIM 设置为 80（50h）。这会增加一个人为偏移，以解决可能存在于 ADC 内核中的负偏移。
2. 将 ADC_CTL1.VREFLOCONV 设置为 1。这将 VREFLO 内部连接到输入通道 B5。
3. 在 B5 上执行多次转换（采样 VREFLO），并考虑板噪声的平均值。
4. 将 ADC_SOFFTRIM 设置为 80（50h）减去在步骤 3 中获得的平均值。这将从步骤 1 中删除了人为偏移，并创建了偏移误差的二进制补码。
5. 将 ADC_CTL1.VREFLOCONV 设置为 0。这会将 B5 重新连接到外部 ADCINB5 输入引脚。

19.4.7.2 ADC 全量程增益校准

随着电压输入的增加，增益误差会作为增量误差发生。最大输入电压下会发生全量程增益误差。与偏置误差一样，增益误差可以为正或负。正的全量程增益误差表示在输入最大电压之前已达到满量程数字结果。负的全量程误差表示将永远无法获得完整的数字结果。将出厂调整值写入 ADC_REFTRIM 寄存器，以校正 ADC 全量程增益误差。

19.4.8 内部/外部参考电压选择

19.4.8.1 内部参考电压

ADC 可以在两种不同的参考模式下运行，这些模式由 ADC_CTL1.ADCREFSEL 位选择。默认情况下，选择内部带隙来产生 ADC 的参考电压。这将在 0 到 3V 范围内转换显示的电压。在此模式下，控制转换的方程式为：

Digital Value = 0 when Input ≤0V

Digital Value = 4096 [(Input – VREFLO)/3V] when 0v < Input < 3V

Digital Value = 4095 when Input ≥ 3V

注意：

1. 所有分数均被截断
2. 在这种模式下，VREFLO 必须接地。这是在某些设备上内部完成的。

19.4.8.2 外部参考电压

为了转换成比例信号形式的电压，应选择外部 VREFHI / VREFLO 引脚来产生参考电压。与内部带隙模式的固定 0 至 3V 输入范围相比，比率模式的输入范围为 VREFLO 至 VREFHI。转换后的值将缩放到该范围。例如 VREFLO 设为 0.5v，而 VREFHI 设为 3.0v，则将 1.75v 的电压转换为 2048 的数字结果。在某些封装上，VREFLO 在内部接地，因此限制为 0v。在此模式下控制转换的方程式为：

Digital Value = 0 when Input ≤ VREFLO

Digital Value = 4096 [(Input – VREFLO)/(VREFHI – VREFLO)] when VREFLO < Input < VREFHI

Digital Value = 4095, when Input ≥ VREFHI

注意：所有分数均被截断

19.4.9 内部温度传感器

内部温度传感器测量芯片的结温。可以使用 ADC_CTL1.TEMPCONV 位控制的开关，通过通道 A5 上的 ADC 对传感器输出进行采样。该开关允许 A5 同时用作外部 ADC 输入引脚又用作温度传感器接入点。对温度传感器进行采样时，ADCINA5 上的外部电路对采样没有影响。

19.4.9.1 传递函数

温度传感器的输出和最终的 ADC 值随结温的升高而增加。此信息可用于将 ADC 传感器采样转换为温度单位。

确定温度的传递函数定义为：

Temperature = (sensor - Offset) * Slope

19.5 寄存器描述

19.5.1 寄存器列表

ADC 基地址：**0x4003 8000**

偏移	实例地址	名称	默认值	描述
0x00	0x40038000	ADC_CTL1	0x00000000	ADC 控制寄存器 1
0x04	0x40038004	ADC_CTL2	0x00000000	ADC 控制寄存器 2
0x08	0x40038008	ADC_INTFLG	0x00000000	ADC 中断标志寄存器
0x0C	0x4003800C	ADC_INTFLGCLR	0x00000000	ADC 中断标志清除寄存器
0x10	0x40038010	ADC_INTOVF	0x00000000	ADC 中断溢出寄存器
0x14	0x40038014	ADC_INTOVFCLR	0x00000000	ADC 中断溢出清除寄存器
0x18	0x40038018	ADC_INTSEL1N2	0x00000000	ADC 中断选择 1/2 寄存器
0x1C	0x4003801C	ADC_INTSEL3N4	0x00000000	ADC 中断选择 3/4 寄存器
0x20	0x40038020	ADC_INTSEL5N6	0x00000000	ADC 中断选择 5/6 寄存器
0x24	0x40038024	ADC_INTSEL7N8	0x00000000	ADC 中断选择 7/8 寄存器
0x28	0x40038028	ADC_INTSEL9	0x00000000	ADC 中断选择 9 寄存器
0x2C	0x4003802C	ADC_REFTRIM	0x00000000	ADC 参考校准寄存器
0x30	0x40038030	ADC_SOFFTRIM0	0x00000000	ADC 单端偏置校准寄存器 0
0x34	0x40038034	ADC_SOFFTRIM1	0x00000000	ADC 单端偏置校准寄存器 1
0x38	0x40038038	ADC_DOFFTRIM0	0x00000000	ADC 差分偏置校准寄存器 0
0x3C	0x4003803C	ADC_DOFFTRIM1	0x00000000	ADC 差分偏置校准寄存器 1
0x40	0x40038040	ADC_SGAINTRIM0	0x00002000	ADC 单端增益校准寄存器 0
0x44	0x40038044	ADC_SGAINTRIM1	0x00002000	ADC 单端增益校准寄存器 1
0x48	0x40038048	ADC_DGAINTRIM0	0x00002000	ADC 差分增益校准寄存器 0
0x4C	0x4003804C	ADC_DGAINTRIM1	0x00002000	ADC 差分增益校准寄存器 1
0x50	0x40038050	ADC_SOCPRICTL	0x00000000	ADC SOC 优先级寄存器
0x54	0x40038054	ADC_SAMPLEMODE	0x00000000	ADC 采样模式寄存器
0x60	0x40038060	ADC_INTSOCSEL1	0x00000000	ADC 中断触发 SOC 选择寄存器 1
0x64	0x40038064	ADC_INTSOCSEL2	0x00000000	ADC 中断触发 SOC 选择寄存器 2
0x70	0x40038070	ADC_SOCFLG1	0x00000000	ADC SOC 标志寄存器
0x74	0x40038074	ADC_SOCFRC1	0x00000000	ADC SOC 强制寄存器
0x78	0x40038078	ADC_SOCOVF1	0x00000000	ADC SOC 溢出寄存器
0x7C	0x4003807C	ADC_SOCOVFCLR1	0x00000000	ADC SOC 溢出清除寄存器
0x80	0x40038080	ADC_SOC0CTL	0x00000000	ADC SOC0 控制寄存器
0x84	0x40038084	ADC_SOC1CTL	0x00000000	ADC SOC1 控制寄存器
0x88	0x40038088	ADC_SOC2CTL	0x00000000	ADC SOC2 控制寄存器
0x8C	0x4003808C	ADC_SOC3CTL	0x00000000	ADC SOC3 控制寄存器
0x90	0x40038090	ADC_SOC4CTL	0x00000000	ADC SOC4 控制寄存器
0x94	0x40038094	ADC_SOC5CTL	0x00000000	ADC SOC5 控制寄存器
0x98	0x40038098	ADC_SOC6CTL	0x00000000	ADC SOC6 控制寄存器

0x9C	0x4003809C	ADC_SOC7CTL	0x00000000	ADC SOC7 控制寄存器
0xA0	0x400380A0	ADC_SOC8CTL	0x00000000	ADC SOC8 控制寄存器
0xA4	0x400380A4	ADC_SOC9CTL	0x00000000	ADC SOC9 控制寄存器
0xA8	0x400380A8	ADC_SOC10CTL	0x00000000	ADC SOC10 控制寄存器
0xAC	0x400380AC	ADC_SOC11CTL	0x00000000	ADC SOC11 控制寄存器
0xB0	0x400380B0	ADC_SOC12CTL	0x00000000	ADC SOC12 控制寄存器
0xB4	0x400380B4	ADC_SOC13CTL	0x00000000	ADC SOC13 控制寄存器
0xB8	0x400380B8	ADC_SOC14CTL	0x00000000	ADC SOC14 控制寄存器
0xBC	0x400380BC	ADC_SOC15CTL	0x00000000	ADC SOC15 控制寄存器
0xC0	0x400380C0	ADC_RESULT0	0x00000000	ADC 结果寄存器 0
0xC4	0x400380C4	ADC_RESULT1	0x00000000	ADC 结果寄存器 1
0xC8	0x400380C8	ADC_RESULT2	0x00000000	ADC 结果寄存器 2
0xCC	0x400380CC	ADC_RESULT3	0x00000000	ADC 结果寄存器 3
0xD0	0x400380D0	ADC_RESULT4	0x00000000	ADC 结果寄存器 4
0xD4	0x400380D4	ADC_RESULT5	0x00000000	ADC 结果寄存器 5
0xD8	0x400380D8	ADC_RESULT6	0x00000000	ADC 结果寄存器 6
0xDC	0x400380DC	ADC_RESULT7	0x00000000	ADC 结果寄存器 7
0xE0	0x400380E0	ADC_RESULT8	0x00000000	ADC 结果寄存器 8
0xE4	0x400380E4	ADC_RESULT9	0x00000000	ADC 结果寄存器 9
0xE8	0x400380E8	ADC_RESULT10	0x00000000	ADC 结果寄存器 10
0xEC	0x400380EC	ADC_RESULT11	0x00000000	ADC 结果寄存器 11
0xF0	0x400380F0	ADC_RESULT12	0x00000000	ADC 结果寄存器 12
0xF4	0x400380F4	ADC_RESULT13	0x00000000	ADC 结果寄存器 13
0xF8	0x400380F8	ADC_RESULT14	0x00000000	ADC 结果寄存器 14
0xFC	0x400380FC	ADC_RESULT15	0x00000000	ADC 结果寄存器 15

19.5.2 寄存器详细描述

19.5.2.1 ADC 控制寄存器 1（ADC_CTL1）

- **名称：ADC Control Register 1**
- **偏移地址：0x00**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:15	14	13	12:8	7	6
名称		ADCENABLE	ADCBSY	ADCBSYCHN	ADCPWDN	ADCCHPWD
访问		R/W	R	R	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0
位	5	4	3	2	1	0
名称	ADCREFPWD		ADCREFSEL	INTPULSEPOS	VREFLOCONV	TEMPCONV
访问	R/W		R/W	R/W	R/W	R/W
复位值	0x0		0x0	0x0	0x0	0x0

字段	说明
[14] ADCENABLE	**ADC 使能信号（ADC Enable）** 0：ADC 模块不使能 1：ADC 模块使能，在开启 ADC 转换前必须配置此位
[13] ADCBSY	**ADC Busy 信号（ADC Busy）** 送入 ADC 状态机，决定 ADC 是否可用于采样 0：ADC 可用于采样下一个通道 1：ADC 正忙，无法采样其他通道
[12:8] ADCBSYCHN	**当前 ADC SOC 信号生成时设置此位（ADC Busy Channel）** ADCBSY=0：保持上一个被转换的 SOC 值 ADCBSY=1：反映当前正在转换的 SOC 值 0x0：SOC0 正在被转换/上一次被转换　0x1：SOC1 正在被转换/上一次被转换 0x2：SOC2 正在被转换/上一次被转换　0x3：SOC3 正在被转换/上一次被转换 0x4：SOC4 正在被转换/上一次被转换　0x5：SOC5 正在被转换/上一次被转换 0x6：SOC6 正在被转换/上一次被转换　0x7：SOC7 正在被转换/上一次被转换 0x8：SOC8 正在被转换/上一次被转换　0x9：SOC9 正在被转换/上一次被转换 0xa：SOC10 正在被转换/上一次被转换　0xb：SOC11 正在被转换/上一次被转换 0xc：SOC12 正在被转换/上一次被转换　0xd：SOC13 正在被转换/上一次被转换 0xe：SOC14 正在被转换/上一次被转换　0xf：SOC15 正在被转换/上一次被转换 0x1x：Reserved
[7] ADCPWDN	**ADC Bias 电路掉电（低有效）（ADC Power Down）** 0：内核中 ADC Bias 电路掉电 1：内核中 ADC Bias 电路上电

[6] ADCCHPWD	**ADC Channel 电路掉电（低有效）（ADC Channel Power Down）** 0：内核中 ADC Channel 电路掉电 1：内核中 ADC Channel 电路上电
[5] ADCREFPWD	**ADC Reference 电路掉电（低有效）（ADC Reference Power Down）** 0：内核中 ADC Reference 电路掉电 1：内核中 ADC Reference 电路上电
[3] ADCREFSEL	**内部或外部 Reference 参考电压选择（ADC Reference Select）** 0：使用内部 Reference 1：使用外部 Reference
[2] INTPULSEPOS	**中断信号生成控制（ADC Interrupt Pulse Position）** 0：中断信号在 ADC 采样结束开始转换时生成 1：中断信号在 ADC 结果锁存到结果寄存器时生成
[1] VREFLOCONV	**VREFLO 转换（ADC VREFLO Conversion）** 当此位置 1 时 ADC 通道 B5（ADC1 通道 5）连接 VREFLO，不连接 ADCINB5 PIN 的输入 0：ADCINB5 PIN 连接到 ADC 模块，不连接 VREFLO 1：VREFLO 连接到 ADC 模块
[0] TEMPCONV	**温度传感器转换（ADC TEMP Conversion）** 当此位置 1 时 ADC 通道 A5（ADC0 通道 5）连接温度传感器输入，不连接 ADCINA5 PIN 的输入 0：ADCINA5 PIN 连接到 ADC 模块，不连接内部温度传感器 1：内部温度传感器连接到 ADC 模块

19.5.2.2 ADC 控制寄存器 2（ADC_CTL2）

- **名称：ADC Control Register 2**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:12	11	10:8	7:6
名称		ADCSATENABLE	ADCOVSSHIFT	ADCOVSRATIO
访问		R/W	R/W	R/W
复位值		0x0	0x0	0x0
位	5	4	3:2	1:0
名称		ADCOVSENABLE	ADCISEL	
访问		R/W	R/W	
复位值		0x0	0x0	

字段	说明
[11] ADCSATENABLE	**ADC 饱和计算控制位（ADC Saturation Enable）** 0：不支持饱和计算，结果可以为负数 1：支持饱和计算，结果保持为非负数，在 0x0000 和 0x7fff 处饱和

[10:8] ADCOVSSHIFT	**过采样数据移位位数（ADC Oversample Shift）** 0：不右移　1：右移 1 位　2：右移 2 位 3：右移 3 位　4：右移 4 位　7-5：Reserved
[7:6] ADCOVSRATIO	**过采样数据叠加比率（ADC Oversample Ratio）** 0：2x 过采样　1：4x 过采样 2：8x 过采样　3：16x 过采样
[4] ADCOVSENABLE	**过采样使能控制位（ADC Oversample Enable）** 0：不支持过采样 1：支持过采样
[3:2] ADCISEL	**偏置电流档位（ADC Bias Current Select）** 0：10 uA　1：12 uA 2：14 uA　3：8 uA

19.5.2.3 ADC 中断标志寄存器（ADC_INTFLG）

- **名称：ADC Interrupt Flag Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:9	8	7	6	5
名称		ADCINT9	ADCINT8	ADCINT7	ADCINT6
访问		R	R	R	R
复位值		0x0	0x0	0x0	0x0
位	4	3	2	1	0
名称	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
访问	R	R	R	R	R
复位值	0x0	0x0	0x0	0x0	0x0

字段	说明
[8] ADCINT9	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

[7] ADCINT8	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[6] ADCINT7	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[5] ADCINT6	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[4] ADCINT5	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

[3] ADCINT4	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[2] ADCINT3	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[1] ADCINT2	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[0] ADCINT1	**ADC 中断标志位（ADC Interrupt Flag）** 此位指示 ADCINT 脉冲是否生成 0：无 ADC 中断脉冲生成 1：ADC 中断脉冲生成 如果 ADC 中断在连续模式（INTSELxNy 寄存器）下，只要选定的 EOC 事件产生则更多的中断脉冲就会产生，无论此标志位是否置 1。 如果连续模式未使能，则不会有更多的中断脉冲生成，直到用户使用 ADCINTFLGCLR 寄存器将此位清除为止。如果 ADCINTFLG 标志已置位时 EOC 事件产生，则 ADCINTOVF 标志将置位。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

19.5.2.4 ADC 中断标志清除寄存器（ADC_INTFLGCLR）

- **名称：ADC Interrupt Flag Clear Register**
- **偏移地址：0x0C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:9	8	7	6	5
名称		ADCINT9	ADCINT8	ADCINT7	ADCINT6
访问		W	W	W	W
复位值		0x0	0x0	0x0	0x0
位	4	3	2	1	0
名称	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
访问	W	W	W	W	W
复位值	0x0	0x0	0x0	0x0	0x0

字段	说明
[8] ADCINT9	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[7] ADCINT8	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[6] ADCINT7	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成

[5] ADCINT6	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[4] ADCINT5	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[3] ADCINT4	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[2] ADCINT3	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成
[1] ADCINT2	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成

[0] ADCINT1	**ADC 中断标志清除位（ADC Interrupt Flag Clear）** 0：无动作 1：清除 ADCINTFLG 寄存器中的相应标志 清除或设置标志位的边界条件：如果在同一个时钟周期内，硬件尝试将标志位置位，同时软件尝试将标志位清除，将发生以下情况： 1）软件优先级更高，将会清除此标志 2）硬件置位将无效，没有信号会被锁存并传递到 NVIC 3）溢出标志或情形将会生成

19.5.2.5 ADC 中断溢出寄存器（ADC_INTOVF）

- **名称：ADC Interrupt Overflow Register**
- **偏移地址：0x10**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:9	8	7	6	5
名称		ADCINT9	ADCINT8	ADCINT7	ADCINT6
访问		R	R	R	R
复位值		0x0	0x0	0x0	0x0
位	4	3	2	1	0
名称	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
访问	R	R	R	R	R
复位值	0x0	0x0	0x0	0x0	0x0

字段	说明
[8] ADCINT9	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[7] ADCINT8	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

[6] ADCINT7	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[5] ADCINT6	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[4] ADCINT5	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[3] ADCINT4	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[2] ADCINT3	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。
[1] ADCINT2	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

[0] ADCINT1	**ADC 中断溢出位（ADC Interrupt Overflow）** 此位指示当生成 ADCINT 脉冲时是否发生溢出情形。如果相应 ADCINTFLG 位置 1，且另外一个选定的 EOC 触发源到来，则溢出情形发生。 0：未检测到 ADC 中断溢出 1：检测到 ADC 中断溢出 溢出位并不关注连续模式位的状态。无论是否为连续模式，溢出条件都会生成。在非连续模式下，在正常中断动作可以恢复之前，必须清除 ADCINTFLG 与 ADCINTOVF 标志。

19.5.2.6 ADC 中断溢出清除寄存器（ADC_INTOVFCLR）

- **名称：ADC Interrupt Flag Clear Register**
- **偏移地址：0x14**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:9	8	7	6	5
名称		ADCINT9	ADCINT8	ADCINT7	ADCINT6
访问		W	W	W	W
复位值		0x0	0x0	0x0	0x0
位	4	3	2	1	0
名称	ADCINT5	ADCINT4	ADCINT3	ADCINT2	ADCINT1
访问	W	W	W	W	W
复位值	0x0	0x0	0x0	0x0	0x0

字段	说明
[8] ADCINT9	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[7] ADCINT8	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[6] ADCINT7	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1

[5] ADCINT6	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[4] ADCINT5	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[3] ADCINT4	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[2] ADCINT3	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[1] ADCINT2	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1
[0] ADCINT1	**ADC 中断溢出清除标志（ADC Interrupt Overflow Clear）** 0：无动作 1：将 ADCINTOVF 寄存器的相应溢出标志清除。如果软件尝试置位此位的动作与硬件尝试置位 ADCINTOVF 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCINTOVF 位会置 1

19.5.2.7 ADC 中断选择 1/2 寄存器（ADC_INTSEL1N2）

- 名称：ADC Interrupt Select 1 And 2 Register
- 偏移地址：0x18
- 默认值：0x00000000
- 返回寄存器列表

位	31:14	13	12:8	7:6	5	4:0
名称		INT2E	INT2SEL		INT1E	INT1SEL
访问		R/W	R/W		R/W	R/W
复位值		0x0	0x0		0x0	0x0

字段	说明
[13]	**ADCINTx 中断使能位（ADC Interrupt Enable）**
INT2E	0：ADCINTx 不使能
	1：ADCINTx 使能
[12:8]	**ADCINTx 触发源选择（ADC Interrupt Source Select）**
INT2SEL	0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源
	…...
	0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved
[5]	**ADCINTx 中断使能位（ADC Interrupt Enable）**
INT1E	0：ADCINTx 不使能
	1：ADCINTx 使能
[4:0]	**ADCINTx 触发源选择（ADC Interrupt Source Select）**
INT1SEL	0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源
	…...
	0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved

19.5.2.8 ADC 中断选择 3/4 寄存器（ADC_INTSEL3N4）

- 名称：**ADC Interrupt Select 3 And 4 Register**
- 偏移地址：**0x1C**
- 默认值：**0x00000000**
- 返回寄存器列表

位	31:14	13	12:8	7:6	5	4:0
名称		INT4E	INT4SEL		INT3E	INT3SEL
访问		R/W	R/W		R/W	R/W
复位值		0x0	0x0		0x0	0x0

字段	说明
[13] INT4E	**ADCINTx 中断使能位（ADC Interrupt Enable）** 0：ADCINTx 不使能 1：ADCINTx 使能
[12:8] INT4SEL	**ADCINTx 触发源选择（ADC Interrupt Source Select）** 0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源 …… 0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved
[5] INT3E	**ADCINTx 中断使能位（ADC Interrupt Enable）** 0：ADCINTx 不使能 1：ADCINTx 使能
[4:0] INT3SEL	**ADCINTx 触发源选择（ADC Interrupt Source Select）** 0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源 …… 0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved

19.5.2.9 ADC 中断选择 5/6 寄存器（ADC_INTSEL5N6）

- **名称：ADC Interrupt Select 5 And 6 Register**
- **偏移地址：0x20**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:14	13	12:8	7:6	5	4:0
名称		INT6E	INT6SEL		INT5E	INT5SEL
访问		R/W	R/W		R/W	R/W
复位值		0x0	0x0		0x0	0x0

字段	说明
[13]	**ADCINTx 中断使能位（ADC Interrupt Enable）**
INT6E	0：ADCINTx 不使能
	1：ADCINTx 使能
[12:8]	**ADCINTx 触发源选择（ADC Interrupt Source Select）**
INT6SEL	0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源
	…...
	0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved
[5]	**ADCINTx 中断使能位（ADC Interrupt Enable）**
INT5E	0：ADCINTx 不使能
	1：ADCINTx 使能
[4:0]	**ADCINTx 触发源选择（ADC Interrupt Source Select）**
INT5SEL	0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源
	…...
	0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved

19.5.2.10 ADC 中断选择 7/8 寄存器（ADC_INTSEL7N8）

- 名称：ADC Interrupt Select 7 And 8 Register
- 偏移地址：0x24
- 默认值：0x00000000
- 返回寄存器列表

位	31:14	13	12:8	7:6	5	4:0
名称		INT8E	INT8SEL		INT7E	INT7SEL
访问		R/W	R/W		R/W	R/W
复位值		0x0	0x0		0x0	0x0

字段	说明
[13] INT8E	**ADCINTx 中断使能位（ADC Interrupt Enable）** 0：ADCINTx 不使能 1：ADCINTx 使能
[12:8] INT8SEL	**ADCINTx 触发源选择（ADC Interrupt Source Select）** 0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源 …… 0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved
[5] INT7E	**ADCINTx 中断使能位（ADC Interrupt Enable）** 0：ADCINTx 不使能 1：ADCINTx 使能
[4:0] INT7SEL	**ADCINTx 触发源选择（ADC Interrupt Source Select）** 0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源 …… 0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved

19.5.2.11 ADC 中断选择 9 寄存器（ADC_INTSEL9）

- 名称：ADC Interrupt Select 9 Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:6	5	4:0
名称		INT9E	INT9SEL
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[5] INT9E	**ADCINTx 中断使能位（ADC Interrupt Enable）** 0：ADCINTx 不使能 1：ADCINTx 使能
[4:0] INT9SEL	**ADCINTx 触发源选择（ADC Interrupt Source Select）** 0x0：EOC0 为 ADCINTx 的触发源　0x1：EOC1 为 ADCINTx 的触发源 …… 0xf：EOC15 为 ADCINTx 的触发源　Others：Reserved

19.5.2.12 ADC 参考校准寄存器（ADC_REFTRIM）

- 名称：ADC Reference Trim Register
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7:4	3:0
名称		EXTTRIM	INTTRIM
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[7:4] EXTTRIM	**ADC 外部 Reference 单端转差分校准位（ADC External Trim）**
[3:0] INTTRIM	**ADC 内部 Reference 单端转差分校准位（ADC Internal Trim）**

19.5.2.13 ADC 单端偏置校准寄存器 0（ADC_SOFFTRIM0）

- **名称：ADC Single-End Offset Trim Register 0**
- **偏移地址：0x30**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		OFFTRIM
访问		R/W
复位值		0x0

字段	说明
[15:0] OFFTRIM	**ADC 偏置补偿（ADC Offset Trim）** ADC 偏置的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的偏置误差。此寄存器对应于 ADC Core0 的单端通道，适用于通过 ADC Core0 的单端通道转换的数据。

19.5.2.14 ADC 单端偏置校准寄存器 1（ADC_SOFFTRIM1）

- **名称：ADC Single-End Offset Trim Register 1**
- **偏移地址：0x34**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		OFFTRIM
访问		R/W
复位值		0x0

字段	说明
[15:0] OFFTRIM	**ADC 偏置补偿（ADC Offset Trim）** ADC 偏置的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的偏置误差。此寄存器对应于 ADC Core1 的单端通道，适用于通过 ADC Core1 的单端通道转换的数据。

19.5.2.15 ADC 差分偏置校准寄存器 0（ADC_DOFFTRIM0）

- 名称：ADC Differential Offset Trim Register 0
- 偏移地址：0x38
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		OFFTRIM
访问		R/W
复位值		0x0

字段	说明
[15:0] OFFTRIM	**ADC 偏置补偿（ADC Offset Trim）** ADC 偏置的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的偏置误差。此寄存器对应于 ADC Core0 的差分通道，适用于通过 ADC Core0 的差分通道转换的数据。

19.5.2.16 ADC 差分偏置校准寄存器 1（ADC_DOFFTRIM1）

- 名称：ADC Differential Offset Trim Register 1
- 偏移地址：0x3C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		OFFTRIM
访问		R/W
复位值		0x0

字段	说明
[15:0] OFFTRIM	**ADC 偏置补偿（ADC Offset Trim）** ADC 偏置的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的偏置误差。此寄存器对应于 ADC Core1 的差分通道，适用于通过 ADC Core1 的差分通道转换的数据。

19.5.2.17 ADC 单端增益校准寄存器 0（ADC_SGAINTRIM0）

- 名称：ADC Single-End Gain Trim Register 0
- 偏移地址：0x40
- 默认值：0x00002000
- 返回寄存器列表

位	31:16	15:0
名称		GAINTRIM
访问		R/W
复位值		0x2000

字段	说明
[15:0] GAINTRIM	**ADC 增益补偿（ADC Gain Trim）** ADC 增益的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的增益误差。此寄存器对应于 ADC Core0 的单端通道，适用于通过 ADC Core0 的单端通道转换的数据。

19.5.2.18 ADC 单端增益校准寄存器 1（ADC_SGAINTRIM1）

- 名称：ADC Single-End Gain Trim Register 1
- 偏移地址：0x44
- 默认值：0x00002000
- 返回寄存器列表

位	31:16	15:0
名称		GAINTRIM
访问		R/W
复位值		0x2000

字段	说明
[15:0] GAINTRIM	**ADC 增益补偿（ADC Gain Trim）** ADC 增益的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的增益误差。此寄存器对应于 ADC Core1 的单端通道，适用于通过 ADC Core1 的单端通道转换的数据。

19.5.2.19 ADC 差分增益校准寄存器 0（ADC_DGAINTRIM0）

- 名称：ADC Differential Gain Trim Register 0
- 偏移地址：0x48
- 默认值：0x00002000
- 返回寄存器列表

位	31:16	15:0
名称		GAINTRIM
访问		R/W
复位值		0x2000

字段	说明
[15:0] GAINTRIM	**ADC 增益补偿（ADC Gain Trim）** ADC 增益的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的增益误差。此寄存器对应于 ADC Core0 的差分通道，适用于通过 ADC Core0 的差分通道转换的数据。

19.5.2.20 ADC 差分增益校准寄存器 1（ADC_DGAINTRIM1）

- 名称：ADC Differential Gain Trim Register 1
- 偏移地址：0x4C
- 默认值：0x00002000
- 返回寄存器列表

位	31:16	15:0
名称		GAINTRIM
访问		R/W
复位值		0x2000

字段	说明
[15:0] GAINTRIM	**ADC 增益补偿（ADC Gain Trim）** ADC 增益的形式为 16 位补码，范围为-32768 到+32767。这些位在出厂测试的时候通过启动程序装载到寄存器中。修改这些位的数值可以补偿板级引入的增益误差。此寄存器对应于 ADC Core1 的差分通道，适用于通过 ADC Core1 的差分通道转换的数据。

19.5.2.21 ADC SOC 优先级寄存器（ADC_SOCPRICTL）

- **名称：ADC Start of Conversion Priority Control Register**
- **偏移地址：0x50**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15	14:11	10:5	4:0
名称		ONESHOT		RRPOINTER	SOCPRIORITY
访问		R/W		R	R/W
复位值		0x0		0x0	0x0

字段	说明
[15] ONESHOT	**单次模式控制位（ADC One-Shot Mode）** 0：单次模式不使能 1：单次模式使能
[10:5] RRPOINTER	**轮转优先级指针（ADC Round-Robin Pointer）** 保存上一个转换过的轮转 SOCx 数值，此值将被轮转机制用于决定转换顺序。 0x0：SOC0 是上一个轮转转换的 SOC，SOC1 具有最高的轮转优先级 0x1：SOC1 是上一个轮转转换的 SOC，SOC2 具有最高的轮转优先级 ….. 0xf：SOC15 是上一个轮转转换的 SOC，SOC0 具有最高的轮转优先级 0x20：复位值，用于指示没有 SOC 被转换过。SOC0 具有最高轮转优先级。当外设被复位，或当 ADC_SOCPRICTL 寄存器被写入时，此位被置为此值。在后一种情况下，如果一个转换正在进行当中，此转换会完成并且完成之后新优先级生效。 Others：Reserved
[4:0] SOCPRIORITY	**SOC 优先级（ADC SOC Priority）** 定义优先级模式的截止点以及 SOCx 的轮转仲裁。 0x0：SOC 优先级对所有通道都是以轮转模式处理 0x1：SOC0 为高优先级，SOC1-SOC15 以轮转模式处理 0x2：SOC0-SOC1 为高优先级，SOC2-SOC15 以轮转模式处理 …… 0xf：SOC0-SOC14 为高优先级，SOC15 以轮转模式处理 0x10：所有 SOCx 均为高优先级，由 SOCx 数值仲裁 Others：Reserved

19.5.2.22 ADC 采样模式寄存器（ADC_SAMPLEMODE）

- **名称：ADC Sample Mode Register**
- **偏移地址：0x54**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:8	7	6	5	4
名称		SIMULEN14	SIMULEN12	SIMULEN10	SIMULEN8
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	3	2	1	0	
名称	SIMULEN6	SIMULEN4	SIMULEN2	SIMULEN0	
访问	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	

字段	说明
[7] SIMULEN14	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[6] SIMULEN12	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[5] SIMULEN10	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。

[4] SIMULEN8	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[3] SIMULEN6	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[2] SIMULEN4	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[1] SIMULEN2	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。
[0] SIMULEN0	**SOCx/SOCx+1 的同步采样使能（ADC SOCx/SOCx+1 Simultaneous Sample Mode）** 将 SOCx 与 SOCx+1 在同步采样模式中配对。当 ADC 正在转换 SOCx/SOCx+1 时，此位不可以被置 1。 0：SOCx 与 SOCx+1 单路采样模式设置。CHSEL 域的所有位定义要转换的通道。EOC0 与 SOC0 有关，EOC1 与 SOC1 有关。SOC0 的结果存放在 ADCRESULT0 寄存器，SOC1 的结果存放在 ADCRESULT1 寄存器。 1：SOCx 与 SOCx+1 同步采样模式设置。CHSEL 域的最低 3 位定义要转换的通道对。EOC0 和 EOC1 与 SOC0 和 SOC1 对有关。SOC1 和 SOC1 的结果分别存放在 ADCRESULT0 与 ADCRESULT1 寄存器。

19.5.2.23 ADC 中断触发 SOC 选择寄存器 1（ADC_INTSOCSEL1）

- 名称：ADC Interrupt Trigger SOC Select 1 Register
- 偏移地址：0x60
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:14	13:12	11:10	9:8
名称		SOC7	SOC6	SOC5	SOC4
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	7:6	5:4	3:2	1:0	
名称	SOC3	SOC2	SOC1	SOC0	
访问	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	

字段	说明
[15:14] SOC7	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[13:12] SOC6	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[11:10] SOC5	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[9:8] SOC4	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[7:6] SOC3	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved

[5:4] SOC2	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[3:2] SOC1	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[1:0] SOC0	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved

19.5.2.24 ADC 中断触发 SOC 选择寄存器 2（ADC_INTSOCSEL2）

- **名称：ADC Interrupt Trigger SOC Select 2 Register**
- **偏移地址：0x64**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:14	13:12	11:10	9:8
名称		SOC15	SOC14	SOC13	SOC12
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	7:6	5:4	3:2	1:0	
名称	SOC11	SOC10	SOC9	SOC8	
访问	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	

字段	说明
[15:14] SOC15	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved

[13:12] SOC14	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[11:10] SOC13	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[9:8] SOC12	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[7:6] SOC11	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[5:4] SOC10	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[3:2] SOC9	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved
[1:0] SOC8	**SOCx ADC 中断触发源选择（ADC SOCx Interrupt Trigger Select）** 选择 ADCINT 触发 SOCx。ADCINT 触发源，与 ADCSOCxCTL 寄存器中 TRIGSEL 位配置的触发源，以及 ADCSOCFRC1 寄存器中的软件强制触发源，三者是相或的关系。 0：没有 ADCINT 会触发 SOCx　1：ADCINT1 会触发 SOCx 2：ADCINT2 会触发 SOCx　3：Reserved

19.5.2.25 ADC SOC 标志寄存器（ADC_SOCFLG1）

- 名称：ADC SOC Flag 1 Register
- 偏移地址：0x70
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15	14	13	12	11	10	9	8
名称		SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
访问		R	R	R	R	R	R	R	R
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	7	6	5	4	3	2	1	0	
名称	SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0	
访问	R	R	R	R	R	R	R	R	
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[15] SOC15	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位。
[14] SOC14	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[13] SOC13	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位

[12] SOC12	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[11] SOC11	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[10] SOC10	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[9] SOC9	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[8] SOC8	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[7] SOC7	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位

[6] SOC6	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[5] SOC5	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[4] SOC4	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[3] SOC3	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[2] SOC2	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位
[1] SOC1	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位

[0] SOC0	**SOCx 转换开始标志（ADC SOCx Start Of Conversion Flag）** 此位指示了各个 SOCx 转换的状态。 0：没有 SOCx 采样请求 1：接收到触发源并产生 SOCx 采样请求 当相应 SOCx 转换开始之后，此位会自动清零。如果在同一个时钟周期内同时收到置位与清除此位的请求，则这种仲裁发生时，无论置位与复位源是什么，此位将会置位而清除动作无效。在这种情况下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论此位之前是否置位

19.5.2.26 ADC SOC 强制寄存器（ADC_SOCFRC1）

- **名称：ADC SOC Force 1 Register**
- **偏移地址：0x74**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15	14	13	12	11	10	9	8
名称		SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
访问		W	W	W	W	W	W	W	W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	7	6	5	4	3	2	1	0	
名称	SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0	
访问	W	W	W	W	W	W	W	W	
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[15] SOC15	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[14] SOC14	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。

[13] SOC13	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[12] SOC12	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[11] SOC11	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[10] SOC10	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[9] SOC9	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。

[8] SOC8	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[7] SOC7	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[6] SOC6	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[5] SOC5	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[4] SOC4	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。

[3] SOC3	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[2] SOC2	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[1] SOC1	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。
[0] SOC0	**SOCx 强制转换开始标志（ADC SOCx Force Start Of Conversion Flag）** 此位写 1 会将 ADCSOCFLG1 寄存器中的 SOCx 标志强制置为 1。 可用于初始化一个软件生成的转换。此位写 0 没有作用。 0：无动作 1：强制 SOCx 标志为 1。一旦优先级给到 SOCx 时将会产生一次转换。 如果软件尝试置位此位的动作与硬件尝试清除 ADCSOCFLG1 寄存器的 SOCx 标志的动作发生在同一个时钟周期，则软件的优先级更高，ADCSOCFLG1 位会置 1。在这种模式下 ADCSOCOVF1 寄存器的溢出标志不会受影响，无论 ADCSOCFLG1 位之前是否置位。

19.5.2.27 ADC SOC 溢出寄存器（ADC_SOCOVF1）

- 名称：ADC SOC Overflow 1 Register
- 偏移地址：0x78
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15	14	13	12	11	10	9	8
名称		SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
访问		R	R	R	R	R	R	R	R
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	7	6	5	4	3	2	1	0	
名称	SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0	
访问	R	R	R	R	R	R	R	R	
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[15] SOC15	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[14] SOC14	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[13] SOC13	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[12] SOC12	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[11] SOC11	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。

[10] SOC10	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[9] SOC9	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[8] SOC8	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[7] SOC7	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[6] SOC6	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[5] SOC5	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[4] SOC4	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[3] SOC3	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）** 此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。

[2]	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）**
SOC2	此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[1]	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）**
SOC1	此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。
[0]	**SOCx 转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Flag）**
SOC0	此位指示当前已有 SOCx 事件挂起时，又产生一个 SOCx 事件。 0：无 SOCx 事件溢出 1：SOCx 事件溢出 溢出条件不会阻止 SOCx 事件被正常处理。它仅指示有一个触发事件可能丢失。

19.5.2.28 ADC SOC 溢出清除寄存器（ADC_SOCOVFCLR1）

- 名称：ADC SOC Overflow Clear 1 Register
- 偏移地址：0x7C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15	14	13	12	11	10	9	8
名称		SOC15	SOC14	SOC13	SOC12	SOC11	SOC10	SOC9	SOC8
访问		W	W	W	W	W	W	W	W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	7	6	5	4	3	2	1	0	
名称	SOC7	SOC6	SOC5	SOC4	SOC3	SOC2	SOC1	SOC0	
访问	W	W	W	W	W	W	W	W	
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[15] SOC15	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[14] SOC14	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[13] SOC13	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[12] SOC12	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。

[11] SOC11	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[10] SOC10	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[9] SOC9	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[8] SOC8	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[7] SOC7	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[6] SOC6	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[5] SOC5	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。

[4] SOC4	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[3] SOC3	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[2] SOC2	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[1] SOC1	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。
[0] SOC0	**SOCx 清除转换开始溢出标志（ADC SOCx Start Of Conversion Overflow Clear）** 此位写 1 清除 ADCSOCOVF1 寄存器中的 SOCx 溢出标志，此位写 0 没有作用。 0：无动作 1：清除 SOCx 溢出标志 如果软件尝试置位此位的动作与硬件尝试置位 ADCSOCOVF1 寄存器的溢出标志的动作发生在同一个时钟周期，则硬件的优先级更高，ADCSOCOVF1 位会置 1。

19.5.2.29 ADC SOC0 控制寄存器（ADC_SOC0CTL）

- **名称：ADC SOC0 Control Register**
- **偏移地址：0x80**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.30 ADC SOC1 控制寄存器（ADC_SOC1CTL）

- **名称：ADC SOC1 Control Register**
- **偏移地址：0x84**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11]	**SOCx 触发源选择（ADC SOCx Trigger Source Select）**
TRIGSEL	该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。
	0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA
	0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA
	0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA
	0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA
	0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA
	0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA
	0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA
	0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。
	序列采样模式下（SIMULENx=0）：
	0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7
	0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7
	同步采样模式下（SIMULENx=1）：
	0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1
	.....　0x7：ADCINA7/ADCINB7　0x8：Reserved
	0x9：Reserved　.....　0xf：Reserved

[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0] ACQPS	**SOCx 采样时间配置（ADC SOCx Sample Time）** 该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.31 ADC SOC2 控制寄存器（ADC_SOC2CTL）

- **名称：ADC SOC2 Control Register**
- **偏移地址：0x88**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　Others：Reserved

[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3]	**SOCx 通道模式配置（ADC SOCx Channel Mode）**
CHMOD	该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.32 ADC SOC3 控制寄存器（ADC_SOC3CTL）

- **名称：ADC SOC3 Control Register**
- **偏移地址：0x8C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.33 ADC SOC4 控制寄存器（ADC_SOC4CTL）

- **名称：ADC SOC4 Control Register**
- **偏移地址：0x90**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11]	**SOCx 触发源选择（ADC SOCx Trigger Source Select）**
TRIGSEL	该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。
	0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA
	0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA
	0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA
	0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA
	0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA
	0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA
	0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA
	0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。
	序列采样模式下（SIMULENx=0）：
	0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7
	0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7
	同步采样模式下（SIMULENx=1）：
	0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1
	.....　0x7：ADCINA7/ADCINB7　0x8：Reserved
	0x9：Reserved　.....　0xf：Reserved

[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0] ACQPS	**SOCx 采样时间配置（ADC SOCx Sample Time）** 该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.34 ADC SOC5 控制寄存器（ADC_SOC5CTL）

- **名称：ADC SOC5 Control Register**
- **偏移地址：0x94**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　Others：Reserved

[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3]	**SOCx 通道模式配置（ADC SOCx Channel Mode）**
CHMOD	该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.35 ADC SOC6 控制寄存器（ADC_SOC6CTL）

- **名称：ADC SOC6 Control Register**
- **偏移地址：0x98**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.36 ADC SOC7 控制寄存器（ADC_SOC7CTL）

- **名称：ADC SOC7 Control Register**
- **偏移地址：0x9C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11]	**SOCx 触发源选择（ADC SOCx Trigger Source Select）**
TRIGSEL	该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。
	0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA
	0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA
	0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA
	0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA
	0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA
	0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA
	0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA
	0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。
	序列采样模式下（SIMULENx=0）：
	0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7
	0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7
	同步采样模式下（SIMULENx=1）：
	0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1
	.....　0x7：ADCINA7/ADCINB7　0x8：Reserved
	0x9：Reserved　.....　0xf：Reserved

[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0] ACQPS	**SOCx 采样时间配置（ADC SOCx Sample Time）** 该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.37 ADC SOC8 控制寄存器（ADC_SOC8CTL）

- **名称：ADC SOC8 Control Register**
- **偏移地址：0xA0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　Others：Reserved

[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3]	**SOCx 通道模式配置（ADC SOCx Channel Mode）**
CHMOD	该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.38 ADC SOC9 控制寄存器（ADC_SOC9CTL）

- 名称 ADC SOC9 Control Register
- 偏移地址：0xA4
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.39 ADC SOC10 控制寄存器（ADC_SOC10CTL）

- **名称：AADC SOC10 Control Register**
- **偏移地址：0xA8**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11]	**SOCx 触发源选择（ADC SOCx Trigger Source Select）**
TRIGSEL	该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。
	0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA
	0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA
	0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA
	0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA
	0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA
	0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA
	0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA
	0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。
	序列采样模式下（SIMULENx=0）：
	0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7
	0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7
	同步采样模式下（SIMULENx=1）：
	0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1
	.....　0x7：ADCINA7/ADCINB7　0x8：Reserved
	0x9：Reserved　.....　0xf：Reserved

[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0] ACQPS	**SOCx 采样时间配置（ADC SOCx Sample Time）** 该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.40 ADC SOC11 控制寄存器（ADC_SOC11CTL）

- **名称：ADC SOC11 Control Register**
- **偏移地址：0xAC**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　Others：Reserved

[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3]	**SOCx 通道模式配置（ADC SOCx Channel Mode）**
CHMOD	该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.41 ADC SOC12 控制寄存器（ADC_SOC12CTL）

- **名称：ADC SOC12 Control Register**
- **偏移地址：0xB0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.42 ADC SOC13 控制寄存器（ADC_SOC13CTL）

- **名称：ADC SOC13 Control Register**
- **偏移地址：0xB4**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11]	**SOCx 触发源选择（ADC SOCx Trigger Source Select）**
TRIGSEL	该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。
	0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA
	0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA
	0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA
	0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA
	0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA
	0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA
	0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA
	0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。
	序列采样模式下（SIMULENx=0）：
	0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7
	0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7
	同步采样模式下（SIMULENx=1）：
	0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1
	.....　0x7：ADCINA7/ADCINB7　0x8：Reserved
	0x9：Reserved　.....　0xf：Reserved

[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0] ACQPS	**SOCx 采样时间配置（ADC SOCx Sample Time）** 该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.43 ADC SOC14 控制寄存器（ADC_SOC14CTL）

- **名称：ADC SOC14 Control Register**
- **偏移地址：0xB8**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　Others：Reserved

[9:6]	**SOCx 通道选择（ADC SOCx Channel Select）**
CHSEL	该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3]	**SOCx 通道模式配置（ADC SOCx Channel Mode）**
CHMOD	该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式
[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。 0：采样时间为 6 个 ADCclk 周期　1：采样时间为 18 个 ADCclk 周期 2：采样时间为 42 个 ADCclk 周期　3：采样时间为 90 个 ADCclk 周期 4：采样时间为 186 个 ADCclk 周期　5：采样时间为 378 个 ADCclk 周期 6：采样时间为 762 个 ADCclk 周期　7：采样时间为 1530 个 ADCclk 周期

19.5.2.44 ADC SOC15 控制寄存器（ADC_SOC15CTL）

- **名称：ADC SOC15 Control Register**
- **偏移地址：0xBC**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:11	10	9:6	5:4	3	2:0
名称		TRIGSEL		CHSEL		CHMOD	ACQPS
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[15:11] TRIGSEL	**SOCx 触发源选择（ADC SOCx Trigger Source Select）** 该位配置当优先级给到 SOCx 时，选择哪一个触发源来置位 ADCSOCFLG1 寄存器中相应的 SOCx 标志，以启动一次转换。这一配置可以被 ADCINTSOCSEL1 或 ADCINTSOCSEL2 寄存器的相应 SOCx 域所覆盖。 0x0：ADCTRG0 - 仅软件触发　　0x5：ADCTRG5 - EPWM0，SOCA 0x6：ADCTRG6 - EPWM0，SOCB　　0x7：ADCTRG7 - EPWM1，SOCA 0x8：ADCTRG8 - EPWM1，SOCB　　0x9：ADCTRG9 - EPWM2，SOCA 0xa：ADCTRG10 - EPWM2，SOCB　　0xb：ADCTRG11 - EPWM3，SOCA 0xc：ADCTRG12 - EPWM3，SOCB　　0xd：ADCTRG13 - EPWM4，SOCA 0xe：ADCTRG14 - EPWM4，SOCB　　0xf ：ADCTRG15 - EPWM5，SOCA 0x10：ADCTRG16 - EPWM5，SOCB　　0x11：ADCTRG17 – EPWM6，SOCA 0x12：ADCTRG18 – EPWM6，SOCB　　Others：Reserved
[9:6] CHSEL	**SOCx 通道选择（ADC SOCx Channel Select）** 该位选择当 ADC 收到 SOCx 时转换哪一个通道。 序列采样模式下（SIMULENx=0）： 0x0：ADCINA0　0x1：ADCINA1　.....　0x7：ADCINA7 0x8：ADCINB0　0x9：ADCINB1　.....　0xf：ADCINB7 同步采样模式下（SIMULENx=1）： 0x0：ADCINA0/ADCINB0　0x1：ADCINA1/ADCINA1 .....　0x7：ADCINA7/ADCINB7　0x8：Reserved 0x9：Reserved　.....　0xf：Reserved
[3] CHMOD	**SOCx 通道模式配置（ADC SOCx Channel Mode）** 该位控制 SOCx 的通道模式。 0：转换通道为单端模式 1：转换通道为差分模式

[2:0]	**SOCx 采样时间配置（ADC SOCx Sample Time）**
ACQPS	该位控制 SOCx 的采样时间。
	0：采样时间为 6 个 ADCclk 周期　　1：采样时间为 18 个 ADCclk 周期
	2：采样时间为 42 个 ADCclk 周期　　3：采样时间为 90 个 ADCclk 周期
	4：采样时间为 186 个 ADCclk 周期　　5：采样时间为 378 个 ADCclk 周期
	6：采样时间为 762 个 ADCclk 周期　　7：采样时间为 1530 个 ADCclk 周期

19.5.2.45 ADC 结果寄存器 0（ADC_RESULT0）

- **名称：ADC Result Register 0**
- **偏移地址：0xC0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0]	**SOCx 转换结果（ADC SOCx Result）**
RESULT	序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.46 ADC 结果寄存器 1（ADC_RESULT1）

- **名称：ADC Result Register 1**
- **偏移地址：0xC4**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.47 ADC 结果寄存器 2（ADC_RESULT2）

- 名称：ADC Result Register 2
- 偏移地址：0xC8
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.48 ADC 结果寄存器 3（ADC_RESULT3）

- 名称：ADC Result Register 3
- 偏移地址：0xCC
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.49 ADC 结果寄存器 4（ADC_RESULT4）

- **名称：ADC Result Register 4**
- **偏移地址：0xD0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.50 ADC 结果寄存器 5（ADC_RESULT5）

- 名称：ADC Result Register 5
- 偏移地址：0xD4
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.51 ADC 结果寄存器 6（ADC_RESULT6）

- 名称：ADC Result Register 6
- 偏移地址：0xD8
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.52 ADC 结果寄存器 7（ADC_RESULT7）

- **名称：ADC Result Register 7**
- **偏移地址：0xDC**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.53 ADC 结果寄存器 8（ADC_RESULT8）

- **名称：ADC Result Register 8**
- **偏移地址：0xE0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.54 ADC 结果寄存器 9（ADC_RESULT9）

- 名称：ADC Result Register 9
- 偏移地址：0xE4
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.55 ADC 结果寄存器 10（ADC_RESULT10）

- 名称：ADC Result Register 10
- 偏移地址：0xE8
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.56 ADC 结果寄存器 11（ADC_RESULT11）

- 名称：ADC Result Register 11
- 偏移地址：0xEC
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.57 ADC 结果寄存器 12（ADC_RESULT12）

- **名称：AADC Result Register 12**
- **偏移地址：0xF0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.58 ADC 结果寄存器 13（ADC_RESULT13）

- 名称：ADC Result Register 13
- 偏移地址：0xF4
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.59 ADC 结果寄存器 14（ADC_RESULT14）

- **名称：ADC Result Register 14**
- **偏移地址：0xF8**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

19.5.2.60 ADC 结果寄存器 15（ADC_RESULT15）

- **名称：ADC Result Register 15**
- **偏移地址：0xFC**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RESULT
访问		R
复位值		0x0

字段	说明
[15:0] RESULT	**SOCx 转换结果（ADC SOCx Result）** 序列采样模式下（SIMULENx=0）： 在 ADC 完成一次 SOC 转换之后，转换结果存放在相应的 ADCRESULTx 寄存器中。例如，如果 SOC4 用于采样 ADCINA1，则完整的转换结果将存放在 ADCRESULT4 中。 同步采样模式下（SIMULENx=1）： 在 ADC 完成一个采样对的转换之后，转换结果存放在相应的相应的 ADCRESULTx 和 ADCRESULTx + 1 寄存器中（假定 x 是偶数）。例如，对于 SOC4，这些完整的转换结果将存放在 ADCRESULT4 和 ADCRESULT5 中。

20 比较器（DACCMP）

20.1 简介

比较器模块是 VDDA 域中的模拟电压比较器。核心模拟电路包括比较器，其输入和输出以及内部 DAC 参考。模块支持的数字电路包括 DAC 控制，与其他片上逻辑的接口，输出滤波模块和可编程控制信号。

20.2 结构框图

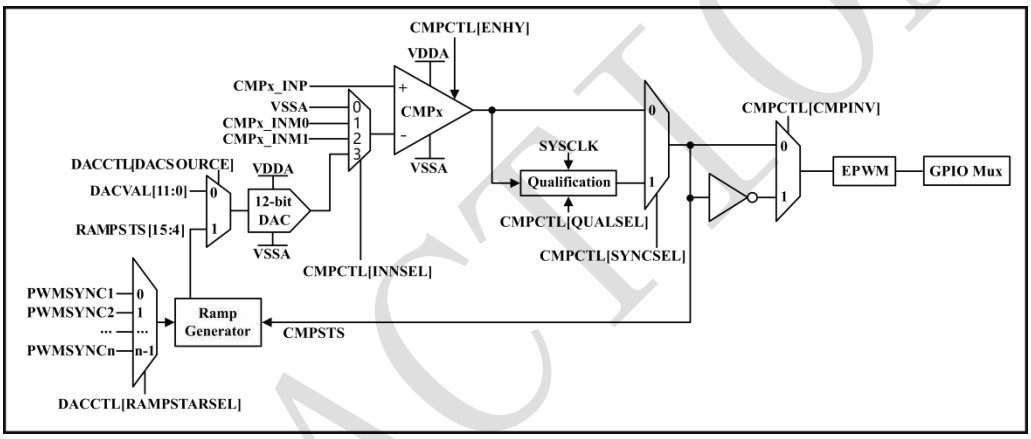


图 20-1 比较器结构框图

注意：比较器迟滞反馈默认情况下处于启用状态，可能会干扰高阻抗输入信号。使用 DACCMP 模块中的 DACCMP_CMPCTL 寄存器配置比较器迟滞。

20.3 主要特性

比较器模块可以监视两个外部模拟输入，或者使用内部 DAC 参考作为另一个输入来监视一个外部模拟输入。比较器的输出可以异步传出，也可以滤波并与系统时钟周期同步。比较器输出被路由到 EPWM 关断区域模块以及 GPIO 输出多路复用器。

20.4 功能描述

20.4.1 比较器功能

比较器是一个模拟比较器模块，因此其输出与系统时钟异步。比较器的真值表如表 20-1 所示。

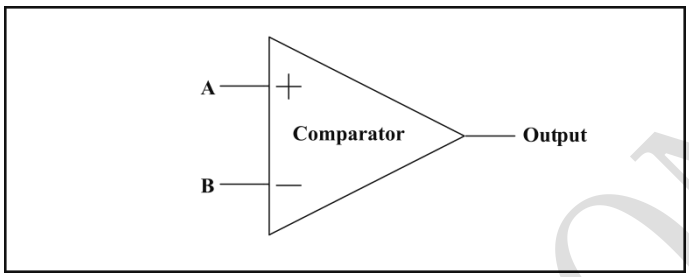


图 20-2 比较器

表 20-1 比较器真值表

Voltages	Output
Voltage A > Voltage B	1
Voltage B > Voltage A	0

由于比较器输出的响应中存在迟滞现象，所以没有定义电压 A=电压 B 的条件。这也限制了比较器输出对输入电压噪声的灵敏度。

比较器输出支持 4 档迟滞大小，分别是 0mV/10mV/20mV/30mV，迟滞大小通过 DACCMP_CMPCTL 寄存器的 ENHY 位配置。

经滤波后，比较器的输出状态由 DACCMP_CMPSTS.CMPSTS 位反映。如果未启用模块时钟，则 DACCMP_CMPSTS 寄存器将不会更新。

20.4.2 DAC 基准电压

每个比较器都包含一个内部 12 位电压 DAC 基准电压，可用于提供比较器的反相输入（B 侧输入）。DAC 的电压输出由 DACCMP_DACVAL.DACVAL 位字段控制。DAC 的输出由以下公式给出：

$$V = \frac{DACVAL * (VDDA - VSSA)}{4095}$$

由于 DAC 也在模拟域中，它不需要时钟来维持其电压输出。但是，需要时钟来修改控制 DAC 的数字输入。

20.4.3 斜坡发生器输入

选中后，斜坡发生器（见图 20-3）可以产生一个斜坡下降的 DAC 输出信号。在这种模式下，DAC 使用 16 位 DACCMP_RAMPSTS 向下计数寄存器的最高 12 位作为其输入。

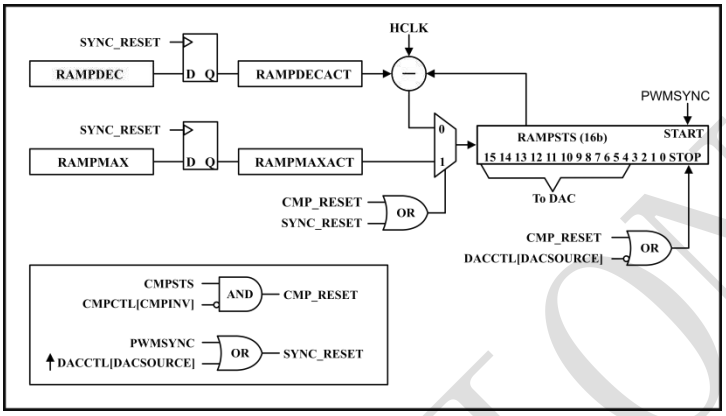


图 20-3 斜坡发生器框图

注意：斜坡发生器的 PWMSYNC 信号来自 EPWM 寄存器 EPWMx_TBCTL[PWMSYNCSEL] 位。PWMSYNC 信号与 EPWMSYNCI 和 EPWMSYNCO 信号不同。

当接收到选定的 PWMSYNC 信号时，DACCMP_RAMPSTS 寄存器的值设置为 DACCMP_RAMPMAX 的值，此后每个 HCLK 周期从 DACCMP_RAMPSTS 中减去 DACCMP_RAMPDECACT 的值。通过设置 DACSOURCE = 1 首次启用 Ramp 发生器时，将从 DACCMP_RAMPMAX 加载 RAMPSTS 的值，并且该寄存器保持静态，直到接收到第一个 PWMSYNC 信号为止。

如果在斜坡发生器有效时比较器将 CMPSTS 位置 1，则 RAMPSTS 寄存器将复位为 DACCMP_RAMPMAXACT 的值，并保持静态，直到接收到下一个 PWMSYNC 信号为止。如果 DACCMP_RAMPSTS 的值达到零，则 DACCMP_RAMPSTS 寄存器将保持静态为零，直到接收到下一个 PWMSYNC 信号为止。

为了减少在更新斜坡生成器 DACCMP_RAMPMAX 和 DACCMP_RAMPDEC 值时出现竞争情况的可能性，只有影子寄存器 DACCMP_RAMPMAX 和 DACCMP_RAMPDEC 具有写权限。影子寄存器的值在下一个 PWMSYNC 信号上复制到有效寄存器。用户软件应采取进一步措施，以避免在与 PWMSYNC 信号相同的周期内写入影子寄存器，否则可能会丢失先前的影子寄存器值。

图 20-4 进一步说明了斜坡发生器的行为。

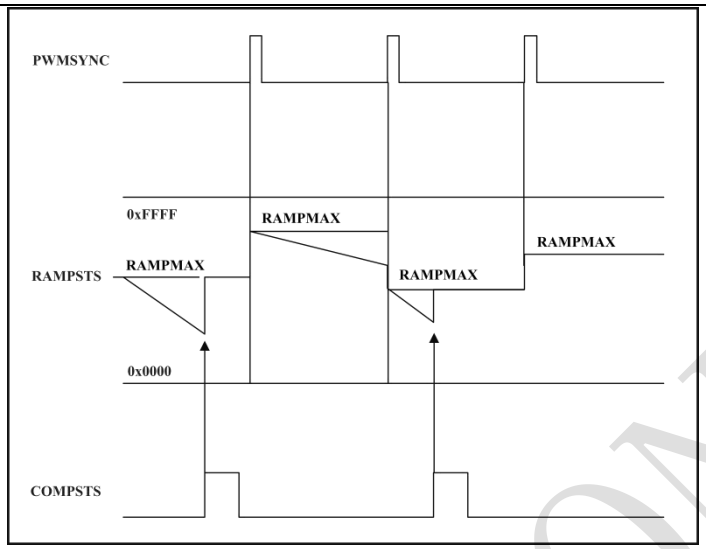


图 20-4　斜坡发生器行为

20.4.4　初始化

在使用比较器模块之前，有两个步骤必须执行：

1. 通过向 DACCMP_DACCTL 寄存器的 ENDAC 位写入 1 来启用 DAC 参考。
2. 通过向 DACCMP_CMPCTL 寄存器的 ENCMP 位写入 1 来启用比较器模块。

20.4.5　数字域操作

在比较器的输出端，还有两个功能模块可用于影响比较器输出的行为。他们是：

1. 反相器电路：由 DACCMP_CMPCTL 寄存器中的 CMPINV 位控制； 将逻辑“非”应用于比较器的输出。该函数是异步的，而其控制需要时钟来更改其值。
2. 滤波模块：由 DACCMP_CMPCTL 寄存器中的 QUALSEL 位字段控制，并由 DACCMP_CMPCTL 寄存器中的 SYNCSEL 位进行选通。该模块可用作简单的滤波器，仅在比较器与系统时钟同步后才使比较器的输出通过，并由 QUALSEL 位字段中定义的系统时钟数滤波。

20.5 寄存器描述

20.5.1 寄存器列表

DACCMP0 基地址：**0x4003 9000**
DACCMP1 基地址：**0x4003 9100**
DACCMP2 基地址：**0x4003 9200**

偏移	实例地址	名称	默认值	描述
0x00	0x40039000	DACCMP_CMPCTL	0x000000F2	CMP 控制寄存器
0x04	0x40039004	DACCMP_CMPSTS	0x00000000	CMP 状态寄存器
0x10	0x40039010	DACCMP_DACCTL	0x00000000	DAC 控制寄存器
0x14	0x40039014	DACCMP_DACSTS	0x00000000	DAC 状态寄存器
0x1C	0x4003901C	DACCMP_DACVAL	0x00000000	DAC 输出值寄存器
0x20	0x40039020	DACCMP_RAMPMAXACT	0x00000000	DAC 斜坡最大值有效寄存器
0x24	0x40039024	DACCMP_RAMPMAX	0x00000000	DAC 斜坡最大值寄存器
0x28	0x40039028	DACCMP_RAMPDECACT	0x00000000	DAC 斜坡递减值有效寄存器
0x2C	0x4003902C	DACCMP_RAMPDEC	0x00000000	DAC 斜坡递减值寄存器
0x30	0x40039030	DACCMP_RAMPSTS	0x00000000	DAC 斜坡发生器状态寄存器

【说明】上表中，x=0、1、2。

20.5.2 寄存器详细描述

20.5.2.1 CMP 控制寄存器（DACCMP_CMPCTL）

- 名称：CMP Control Register
- 偏移地址：0x00
- 默认值：0x000000F2
- 返回寄存器列表

位	31:16	15	14	13:8	7:6
名称		CMPINV	SYNCSEL	QUALSEL	INNSEL
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x3
位	5:4	3	2	1	0
名称	ENHY	FALIE	RISIE	ENPOL	ENCMP
访问	R/W	R/W	R/W	R/W	R/W
复位值	0x3	0x0	0x0	0x1	0x0

字段	说明
[15] CMPINV	**CMP 输出极性（CMP Output Invert）** 0：输出不翻转 1：输出翻转
[14] SYNCSEL	**CMP 输出源选择（CMP Output Sync Select）** 0：模拟输出 1：去抖动输出
[13:8] QUALSEL	**CMP 输出去抖动选择（CMP Output Qualify Select）** 0：关闭去抖动 1：CMP 输出变化的持续时间需要大于 2 个系 统时钟,才会影响到输出 X：CMP 输出变化的持续时间需要大于 X+1 个 系统时钟,才会影响到输出 … 63：CMP 输出变化的持续时间需要大于 64 个系统时钟,才会影响到输出
[7:6] INNSEL	**负端源选择（CMP INN Source Select）** 00：VSSA 01：CMPx_INM0 10：CMPx_INM1 11：DACx_OUT
[5:4] ENHY	**CMP 迟滞（CMP Hysteresis Select）** 00：0 mV 01：10 mV 10：20 mV 11：30 mV

[3] FALIE	**下降沿中断使能（CMP Fall Interrupt Enable）** 0：不使能 1：使能
[2] RISIE	**上升沿中断使能（CMP Rise Interrupt Enable）** 0：不使能 1：使能
[1] ENPOL	**CMP 极性（CMP Polarity Select）** 0：正端大于负端为 0 1：正端大于负端为 1
[0] ENCMP	**CMP 使能（CMP Enable）** 0：不使能 1：使能

20.5.2.2 CMP 状态寄存器（DACCMP_CMPSTS）

- **名称：CMP Status Register**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:3	2	1	0
名称		CMPSTS	FALL	RISE
访问		R	R/W1C	R/W1C
复位值		0x0	0x0	0x0

字段	说明
[2] CMPSTS	**CMP 输出状态（CMP Output Status）** 0：输出为低（输出极性变化之前） 1：输出为高（输出极性变化之前）
[1] FALL	**CMP 下降沿中断标志位（CMP Fall Interrupt Flag）** 0：无下降沿 1：有下降沿 写 1 清 0
[0] RISE	**CMP 上升沿中断标志位（CMP Rise Interrupt Flag）** 0：无上升沿 1：有上升沿 写 1 清 0

20.5.2.3 DAC 控制寄存器（DACCMP_DACCTL）

- 名称：DAC Control Register
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:9	9	8	7	6:4
名称		DACSOURCE	RAMPDIRECT	RAMPSTOPSEL	RAMPSTARSEL
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	3	2	1	0	
名称	DONBIE	DONIE	ENDACOUT	ENDAC	
访问	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	

字段	说明
[9] DACSOURCE	**DAC 输出源选择位（DAC Output Source Select）** 0：DACVAL（寄存器配置） 1：RAMP（斜坡发生器）
[8] RAMPDIRECT	**RAMP 产生方向选择（DAC Ramp Direction Select）** 0：幅度变小 1：幅度变大
[7] RAMPSTOPSEL	**RAMP 结束脉冲触发源选择位（DAC Ramp Stop Select）** 0：CMP Output 1：Soft Trigger B
[6:4] RAMPSTARSEL	**RAMP 起始脉冲触发源选择位（DAC Ramp Start Select）** 000：Pwm Sync 0 001：Pwm Sync 1 ... 110：Pwm Sync 6 111：Soft Trigger
[3] DONBIE	**DAC DONB 中断使能位（DAC Done B Interrupt Enable）** 0：不使能 1：使能
[2] DONIE	**DAC DON 中断使能位（DAC Done Interrupt Enable）** 0：不使能 1：使能
[1] ENDACOUT	**DAC 输出使能位（DAC Output Enable）** 0：不使能 1：使能

[0] ENDAC	**DAC 使能位（DAC Enable）** 0：不使能 1：使能

20.5.2.4 DAC 状态寄存器（DACCMP_DACSTS）

- **名称：DAC Status Register**
- **偏移地址：0x14**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:4	3	2	1	0
名称		SWTB	SWT	DONB	DON
访问		W	W	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0

字段	说明
[3] SWTB	**DAC 斜坡补偿（RAMP）结束触发位（DAC Ramp Stop Trigger）** 0：不触发 1：触发
[2] SWT	**DAC 软件触发位 / 斜坡补偿（RAMP）起始触发位（DAC Ramp Start Trigger）** 0：不触发 1：触发
[1] DONB	**DAC 数据输出完成标志位（DAC Done B Interrupt Flag）** 对应 Soft Trigger B 0：未完成 1：完成 写 1 清 0
[0] DON	**DAC 数据输出完成标志位（DAC Done Interrupt Flag）** 对应 Soft Trigger 0：未完成 1：完成 写 1 清 0

20.5.2.5 DAC 输出值寄存器（DACCMP_DACVAL）

- 名称：DAC Value Register
- 偏移地址：0x1C
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:0
名称		DACVAL
访问		R/W
复位值		0x0

字段	说明
[11:0] DACVAL	DAC 原始输出数据（DAC Output Value）

20.5.2.6 DAC 斜坡最大值有效寄存器（DACCMP_RAMPMAXACT）

- 名称：DAC Ramp Maximum Value Active Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RAMPMAXVAL
访问		R
复位值		0x0

字段	说明
[15:0] RAMPMAXVAL	DAC 斜坡补偿波形最大数据（DAC Ramp Maximum Active Value） 整数部分&小数部分 实际寄存器

20.5.2.7 DAC 斜坡最大值寄存器（DACCMP_RAMPMAX）

- 名称：DAC Ramp Maximum Value Register
- 偏移地址：0x24
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RAMPMAXVAL
访问		R/W
复位值		0x0

字段	说明
[15:0]	**DAC 斜坡补偿波形最大数据（DAC Ramp Maximum Value）**
RAMPMAX	整数部分&小数部分
	影子寄存器

20.5.2.8 DAC 斜坡递减值有效寄存器（DACCMP_RAMPDECACT）

- 名称：DAC Ramp Decrement Value Active Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RAMPDECVAL
访问		R
复位值		0x0

字段	说明
[15:0]	**DAC 斜坡补偿波形递减数据（DAC Ramp Decrement Active Value）**
RAMPDECVAL	整数部分&小数部分
	实际寄存器

20.5.2.9 DAC 斜坡递减值寄存器（DACCMP_RAMPDEC）

- 名称：DAC Ramp Decrement Value Register
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RAMPDECVAL
访问		R/W
复位值		0x0

字段	说明
[15:0]	**DAC 斜坡补偿波形递减数据（DAC Ramp Decrement Value）**
RAMPDECVAL	整数部分&小数部分
	影子寄存器

20.5.2.10 DAC 斜坡发生器状态寄存器（DACCMP_RAMPSTS）

- 名称：DAC Ramp Generator Status Register
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		RAMPVALUE
访问		R
复位值		0x0

字段	说明
[15:0]	**DAC 斜坡补偿后输出数据（DAC Ramp Output Value）**
RAMPVALUE	整数部分&小数部分

21 增强型脉宽调制器（EPWM）

21.1 简介

芯片内置一个 EPWM 模块，可以以最小的 CPU 占用或干预来生成复杂的脉冲宽度调制波形。EPWM 模块高度可编程且非常灵活，各个 EPWM 子模块均具有生成脉冲宽度调制波形所需的定时和控制资源。EPWM 模块由 7 个 EPWM 子模块构建，这些模块可以单独运行，也可以根据需要共同运行以形成系统，避免相互耦合或相互干扰。

在本文档中，信号或模块名中的字母 x 用于表示芯片上的 EPWM 子模块。例如，输出信号 EPWMxA 和 EPWMxB 指来自 EPWMx 通道的输出信号。因此，EPWM0A 和 EPWM0B 属于 EPWM0，同样 EPWM3A 和 EPWM3B 属于 EPWM3。

21.2 结构框图

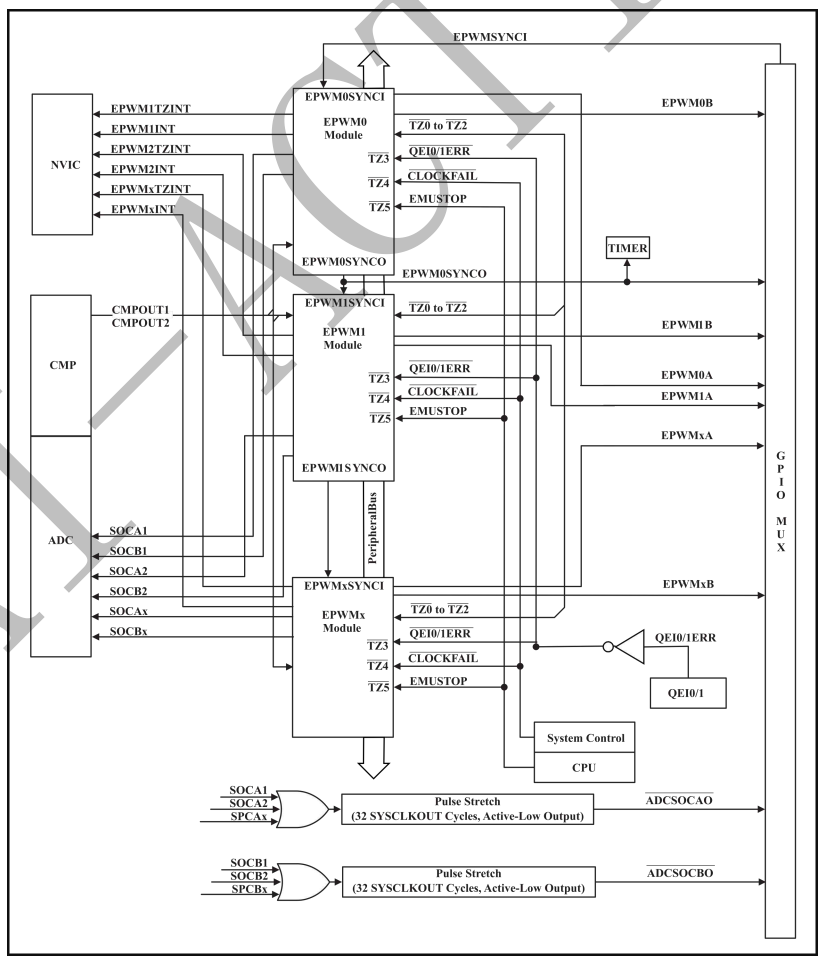


图 21-1 多个 EPWM 子模块

每个 EPWM 子模块由 8 个子单元组成，并通过图 21-2 中所示的信号在系统内连接。

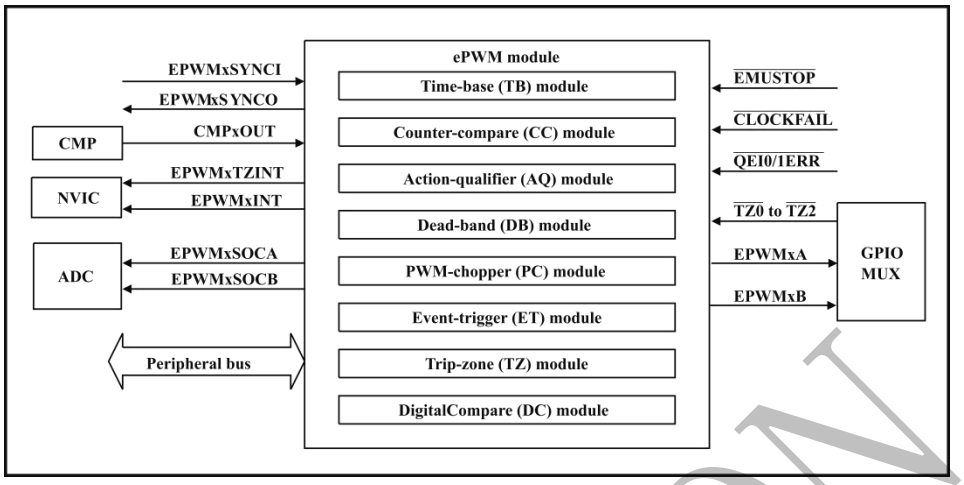


图 21-2　EPWM 子模块及信号连接图

图 21-3 显示了单个 EPWM 模块的更多内部细节。EPWM 模块使用的主要信号是：

- PWM 输出信号（EPWMxA 和 EPWMxB）。
- 关断区域信号（$\overline{TZ0}$ 至 $\overline{TZ5}$）。

  这些输入信号会向 EPWM 模块发出 EPWM 模块外部的故障情况警报。可以将芯片上的每个模块配置为使用或忽略任何关断区域信号。$\overline{TZ0}$ 至 $\overline{TZ2}$ 关断区域信号可以配置为通过 GPIO 模块的异步输入。$\overline{TZ3}$ 连接到来自 QEI 模块的反向 QEI 错误信号（QEI0/1ERR）。$\overline{TZ4}$ 连接到系统时钟故障逻辑，而 $\overline{TZ5}$ 连接到 CPU 的 EMUSTOP 输出。这样，可以在时钟故障或 CPU 停止时配置关断操作。
- 时基同步输入（EPWMxSYNCI）和输出（EPWMxSYNCO）信号。
  同步信号以雏菊链的方式将 EPWM 模块链接在一起。可以将每个模块配置为使用或忽略其同步输入。时钟同步输入和输出信号仅针对 EPWM0（EPWM 模块＃0）引出到引脚。EPWM0 的同步输出（EPWM0 SYNCO）也连接到第一个增强型捕获模块（ECAP）的 SYNCI。
- ADC 转换开始信号（EPWMxSOCA 和 EPWMxSOCB）。
  每个 EPWM 模块都有两个 ADC 转换开始信号。任何 EPWM 模块都可以触发转换开始。在 EPWM 的事件触发子模块中配置了触发转换开始的事件。
- 比较器输出信号（CMPxOUT）。
  比较器模块的输出信号与关断区域信号一起可以生成数字比较事件。
- 外围总线
  外设总线为 32 位宽，允许对 EPWM 寄存器进行 32 位写入。

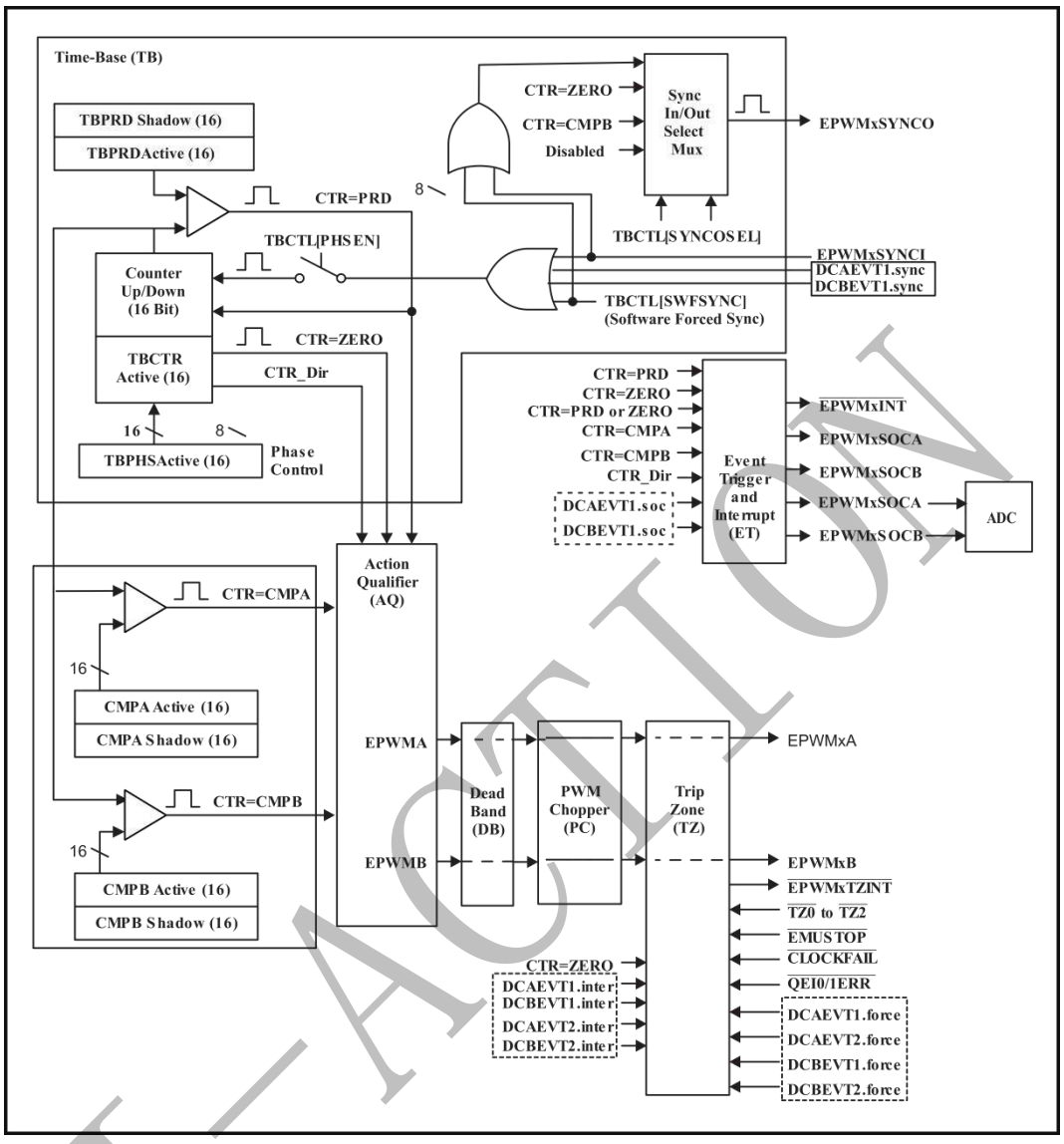


图 21-3　EPWM 子模块及关键信号互连图

图 21-3 还显示了关键的内部子模块互连信号。每个子模块在其各自的部分中进行了详细描述。

21.3 主要特性

每个 EPWM 模块都支持以下功能：

- 具有周期和频率控制的专用 16 位时基计数器
- 两个 PWM 输出（EPWMxA 和 EPWMxB），可以在以下配置中使用：
  - 两个独立的 PWM 输出，具有单沿操作
  - 两个独立的 PWM 输出，具有双沿对称操作
  - 一个独立的 PWM 输出，具有双沿非对称操作
- 通过软件对 PWM 信号进行异步重载控制。
- 相对于其他 EPWM 模块，可编程的相位控制支持滞后或超前操作。
- 逐周期地进行硬件锁定（同步）的相位关系。
- 具有独立的上升沿和下降沿延迟控制的死区。在故障情况下，可以逐周期触发和单次触发的可编程关断区域分配。
- 关断条件可能会在 PWM 输出上强制高，低或高阻抗状态逻辑电平。
- 比较器模块的输出和关断区域的输入可以生成事件，已过滤的事件或关断条件。
- 所有事件均可触发 CPU 中断和 ADC 转换开始（SOC）。
- 可编程的事件预分频可最大程度地减少中断时的 CPU 开销。
- 通过高频载波信号进行 PWM 斩波，对脉冲变压器栅极驱动器很有用。

每个 EPWM 模块都连接到图 21-1 所示的输入/输出信号。这些信号将在后续章节中详细介绍。

21.4 功能描述

21.4.1 EPWM 子模块

每个 EPWM 外设都包含八个子模块。 这些子模块中的每个子模块都可以执行可由软件配置的特定任务。

21.4.1.1 概述

表 21-1 列出了八个关键子模块以及其主要配置参数的列表。例如，如果需要调整或控制 PWM 波形的占空比，则应在相应章节中查看计数比较子模块以获取相关详细信息。

表 21-1 子模块的配置参数

Submodule	Configuration Parameter or Option
时基(TB)	● 配置 PWM 时基计数器（EPWM_TBCTR）的频率或周期。 ● 设置时基计数器的模式： ➢ 向上计数模式：用于非对称 PWM ➢ 向下计数模式：用于非对称 PWM ➢ 上下计数模式：用于对称 PWM ● 配置相对于另一个 EPWM 模块的时基相位。 ● 通过硬件或软件同步模块之间的时基计数器。 ● 在同步事件之后配置时基计数器的方向（向上或向下）。 ● 配置当模拟器停止设备时时基计数器的行为。 ● 指定 EPWM 模块同步输出的来源： ➢ 同步输入信号 ➢ 时基计数等于零 ➢ 时基计数等于计数比较器 B（CMPB） ➢ 没有产生输出同步信号
计数比较 (CC)	● 指定输出 EPWMxA 或输出 EPWMxB 的 PWM 占空比 ● 指定 EPWMxA 或 EPWMxB 输出上发生开关事件的时间
Action-qualifier (AQ)	● 指定发生时基或计数比较子模块事件时采取的操作类型： ➢ 不执行任何操作 ➢ 输出 EPWMxA 或 EPWMxB 切换为高电平 ➢ 输出 EPWMxA 或 EPWMxB 切换为低电平 ➢ 输出 EPWMxA 或 EPWMxB 切换 ● 通过软件控制强制 PWM 输出状态 ● 通过软件配置和控制 PWM 死区
死区(DB)	● 控制上下开关之间的传统互补死区关系 ● 指定输出上升沿延迟值 ● 指定输出下降沿延迟值 ● 完全绕过死区模块。在这种情况下，PWM 波形不加修改地通过。 ● 启用半周期时钟以实现双分辨率的选项

PWM 斩波（PC）	● 创建一个斩波（载波）频率。 ● 斩波脉冲序列中第一个脉冲的脉冲宽度。 ● 第二个及后续脉冲的占空比。 ● 完全忽略 PWM 斩波器模块。在这种情况下，PWM 波形不加修改地通过。
触发（TZ）	● 配置 EPWM 模块以对关断区域信号或数字比较事件中的一个，全部或不做出反应。 ● 指定发生故障时采取的触发操作： ● 强制 EPWMxA 或 EPWMxB 为高 ● 强制 EPWMxA 或 EPWMxB 为低 ● 强制 EPWMxA 或 EPWMxB 进入高阻抗状态 ● 配置 EPWMxA 或 EPWMxB 以忽略任何关断条件。 ● 配置 EPWM 对每个关断区域信号做出反应的频率： ➢ 单次触发 ➢ 逐周期 ● 使能关断区域以启动中断。 ● 完全绕过关断区域模块
事件触发(ET)	● 启用将触发中断的 EPWM 事件。 ● 启用将触发 ADC 转换开始事件的 EPWM 事件。 ● 指定事件触发的速率（每次发生或第二或第三次发生） ● 循环，设置或清除事件标志
数字比较(DC)	● 启用比较器（CMP）模块输出和关断区域信号来创建事件和已过滤事件 ● 指定事件过滤选项以捕获 TBCTR 计数器或生成空白窗口

21.4.1.2 时基(TB)子模块

每个 EPWM 模块都有自己的时基子模块，该子模块确定 EPWM 模块的所有事件时序。内置的同步逻辑使多个 EPWM 模块的时基可以作为一个系统一起工作。图 21-4 说明了时基模块在 EPWM 中的作用。

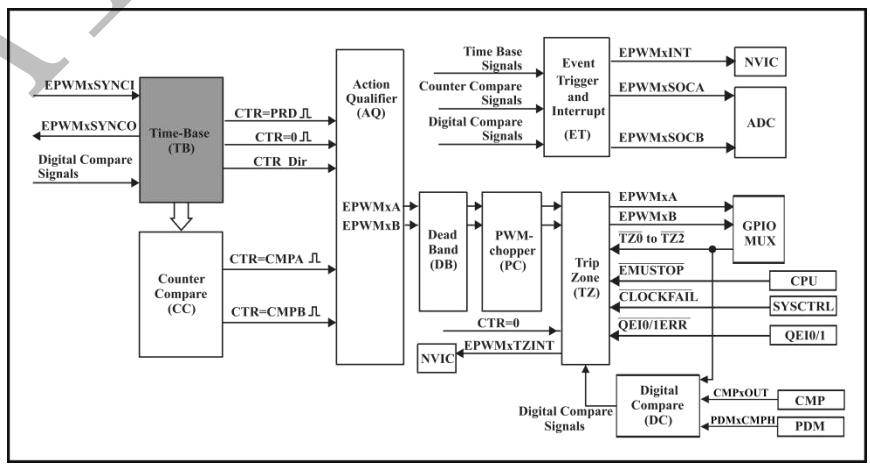


图 21-4　时基子模块结构框图

时基子模块的应用

可以为以下内容配置时基子模块：

- 指定 EPWM 时基计数器（EPWM_TBCTR）的频率或周期以控制事件发生的频率。
- 管理与其他 EPWM 模块的时基同步。
- 与其他 EPWM 模块保持相位关系。
- 将时基计数器设置为向上计数，向下计数或上下计数模式。
- 实现以下事件：
  - CTR = PRD：时基计数器等于指定的周期（TBCTR = TBPRD）。
  - CTR =0：时基计数器等于零（TBCTR = 0x0000）。

控制时基子模块

表 21-2 显示了用来控制时基子模块的寄存器。

表 21-2　时基子模块寄存器

Register	Address offset	Shadowed	Description
EPWM_TBPRD	0x0000	YES	Time-Base Phase Register
EPWM_TBPHS	0x0004	NO	Time-Base Phase Register
EPWM_TBCTR	0x0008	NO	Time-Base Counter Register
EPWM_TBCTL	0x000C	NO	Time-Base Control Register
EPWM_TBSTS	0x0010	NO	Time-Base Status Register

图 21-5 中的框图显示了时基子模块的关键信号和寄存器。表 21-3 提供了与时基子模块关联的关键信号的说明。

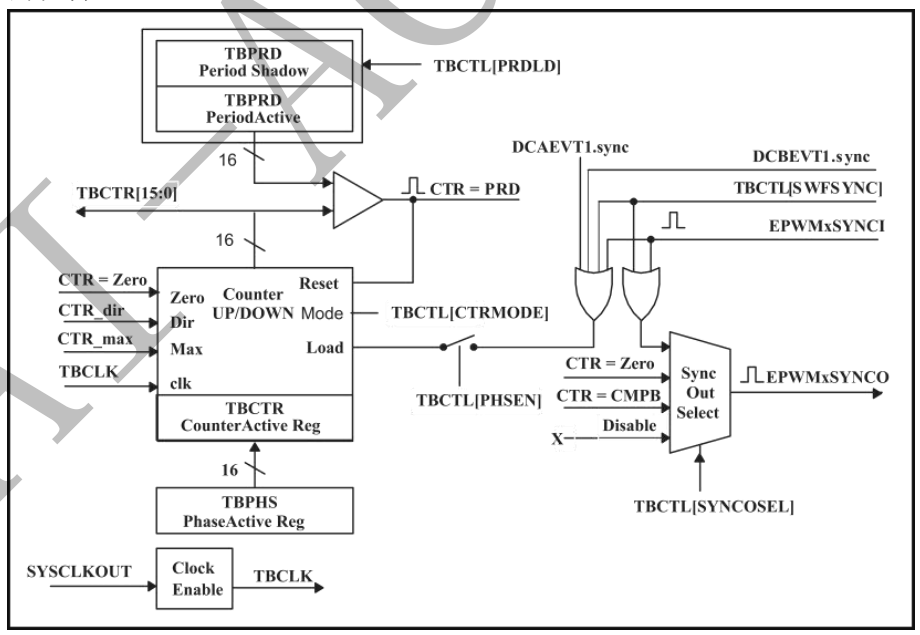


图 21-5　时基子模块信号与寄存器

表 21-3 时基子模块关键信号

Signal	Description
EPWMxSYNCI	Time-base synchronization input. Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module (EPWM0) this signal comes from a device pin. For subsequent ePWM modules this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth.
EPWMxSYNCO	Time-base synchronization output. This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain.The ePWM module generates this signal from one of three event sources: 1. EPWMxSYNCI (Synchronization input pulse) 2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x0000). 3. CTR = CMPB: The time-base counter equal to the counter-Compare B (TBCTR = CMPB) register.
CTR = PRD	Time-base counter equal to the specified period. This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.
CTR = Zero	Time-base counter equal to zero This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x0000.
CTR = CMPB	Time-base counter equal to active counter-Compare B register (TBCTR = CMPB). This event is generated by the counter-Compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction. Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF) Generated event when the TBCTR value reaches its maximum value. This signal is only used only as a status bit
TBCLK	Time-base clock. This is a prescaled version of the system clock (EPWMCLK) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

计算 PWM 周期和频率

PWM 事件的频率由 TBPRD 寄存器和时基计数器的模式控制。图 21-6 显示了当周期设置为 4（TBPRD = 4）时，向上计数，向下计数和上下计数时基计数器模式的周期（T_{pwm}）和频率（F_{pwm}）关系。每一步的时间增量由时基时钟（TBCLK）定义，时基时钟等于 EPWM 功能时钟。

时基计数器有三种由时基控制寄存器(TBCTL)选择的操作模式:

- 上下计数模式：
  在此模式下，时基计数器从零开始递增，直到达到周期(TBPRD)值为止。当达到周期值时，时基计数器将递减直到其达到零。此时，计数器重复该模式并开始递增。
- 向上计数模式：
  在此模式下，时基计数器从零开始递增，直到达到周期寄存器（TBPRD）中的值为止。当达到周期值时，时基计数器将重置为零，并再次开始递增。
- 向下计数模式：
  在此模式下，时基计数器从周期（TBPRD）值开始递减，直到达到零为止。当它达到零时，时基计数器将重置为周期值，并再次开始递减。

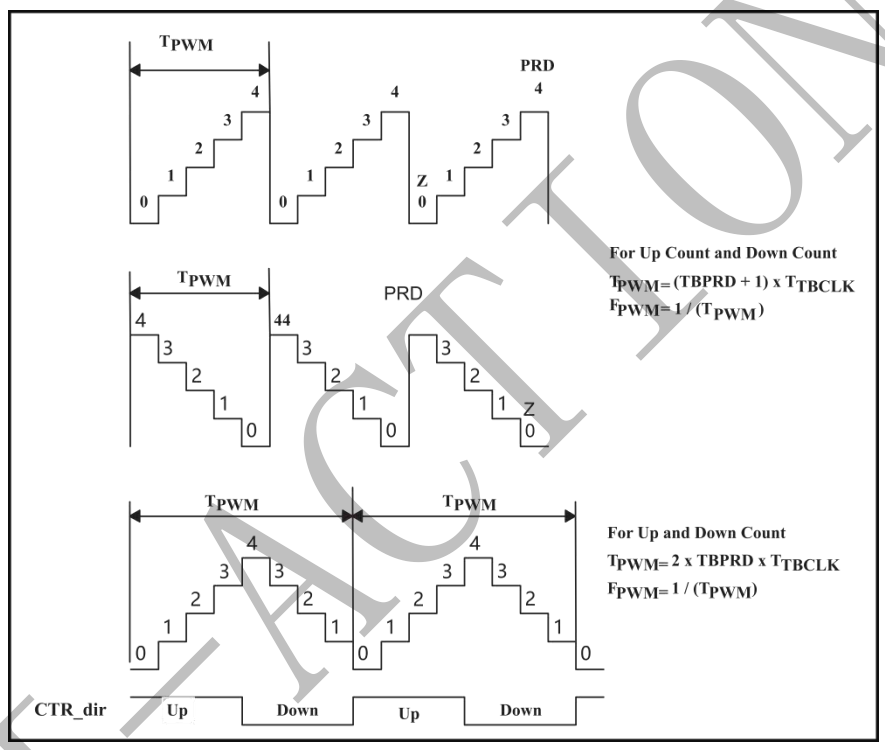


图 21-6　时基频率与周期

时基周期影子寄存器

时基周期寄存器（TBPRD）具有影子寄存器。影子允许寄存器更新与硬件同步。以下定义用于描述 EPWM 模块中的所有影子寄存器：

- 有效寄存器
  有效寄存器控制硬件，并负责硬件引起或调用的动作。
- 影子寄存器
  影子寄存器为有效寄存器缓冲区或提供一个临时的保持位置。它对任何控制硬件都没有直接影响。在关键时刻，影子寄存器的内容将传输至有效寄存器。这样可以防止由于寄存器由软件异步修改而导致的损坏或伪造操作。

  影子周期寄存器的存储器地址与有效寄存器相同。写入或读取哪个寄存器由 TBCTL [PRDLD]位决定。该位启用和禁用 TBPRD 影子寄存器，如下所示：
- 时基周期影子模式:
  当 TBCTL[PRDLD] = 0 时，使能 TBPRD 影子寄存器。对 TBPRD 内存地址的读写操作都

将转到影子寄存器。当时基计数器等于 0 (TBCTR = 0x0000)时，影子寄存器内容被转移到有效寄存器(TBPRD (active)←TBPRD (shadow))。缺省情况下，TBPRD 影子寄存器处于开启状态。

- 时基立即加载模式：
  如果选择了立即加载模式(TBCTL[PRDLD] = 1)，那么对 TBPRD 内存地址的读或写操作将直接进入有效寄存器。

时基计数同步

时基同步方案连接设备上的所有 EPWM 模块。每个 EPWM 模块都有一个同步输入（EPWMxSYNCI）和一个同步输出（EPWMxSYNCO）。第一个实例（EPWM1）的输入同步来自外部引脚。其余 EPWM 模块的可能同步连接如图 21-7 所示。

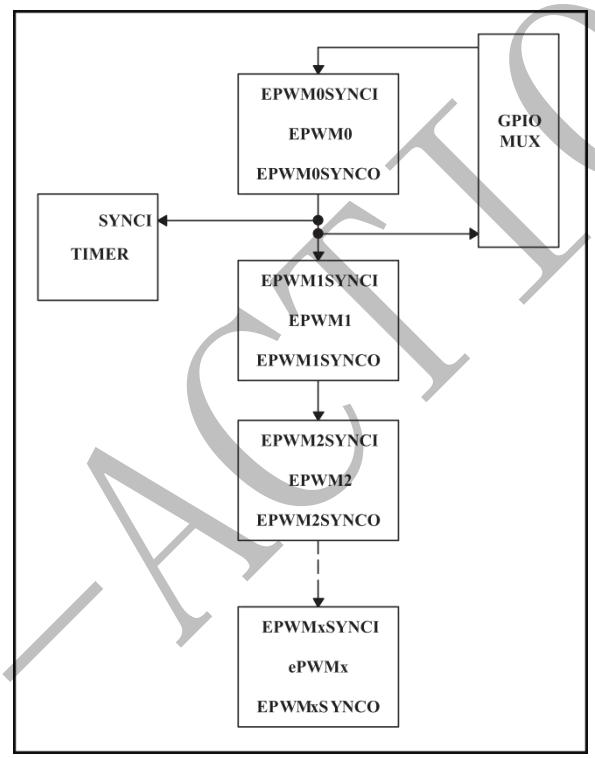


图 21-7　时基计数同步方案

可将每个 EPWM 模块配置为使用或忽略同步输入。如果将 TBCTL [PHSEN]位置 1，则当出现以下情况之一时，EPWM 模块的时基计数器（TBCTR）将自动加载相位寄存器（TBPHS）的内容：

- EPWMxSYNCI：同步输入脉冲：
  当检测到输入同步脉冲时，将相位寄存器的值加载到计数器寄存器中（TBPHS→TBCTR）。该操作在下一个有效的时基时钟（TBCLK）沿发生。
  内部主模块到从模块的延迟为 1 TBCLK
- 软件强制同步脉冲：
  写 1 到 TBCTL[SWFSYNC]控制位调用软件强制同步。这个脉冲与同步输入信号是一致的，因此具有与 EPWMxSYNCI 上的脉冲相同的效果。
- 数字比较事件同步脉冲：
  可以将 DCAEVT1 和 DCBEVT1 数字比较事件配置为生成与 EPWM x SYNCI 具有相同影响的同步脉冲。

此功能使 EPWM 模块可以自动同步到另一个 EPWM 模块的时基。超前或滞后相位控制可以添加到不同的 EPWM 模块产生的波形来同步它们。在向上计数模式下，TBCTL[PSHDIR]位在同步事件发生后立即配置时基计数器的方向。新的方向独立于同步事件之前的方向。在递增或递减模式下，PHSDIR 位被忽略。

清零 TBCTL [PHSEN]位会将 EPWM 配置为忽略同步输入脉冲。同步脉冲仍可以同步到 EPWMxSYNCO，并用于同步其他 EPWM 模块。这样您可以设置主时基（例如，EPWM0），而下游模块（EPWM1~EPWMx）可以选择与主控时基同步运行。

时基计数器模式和时序波形

时基计数器有四种工作模式:

- 向上计数模式，不对称
- 向下计数模式，不对称
- 上下计数模式，对称
- 冻结时基计数器保持不变的当前值

为了说明前三种模式的操作，下面的计时图显示了何时生成事件以及时基如何响应 EPWMxSYNCI 信号。

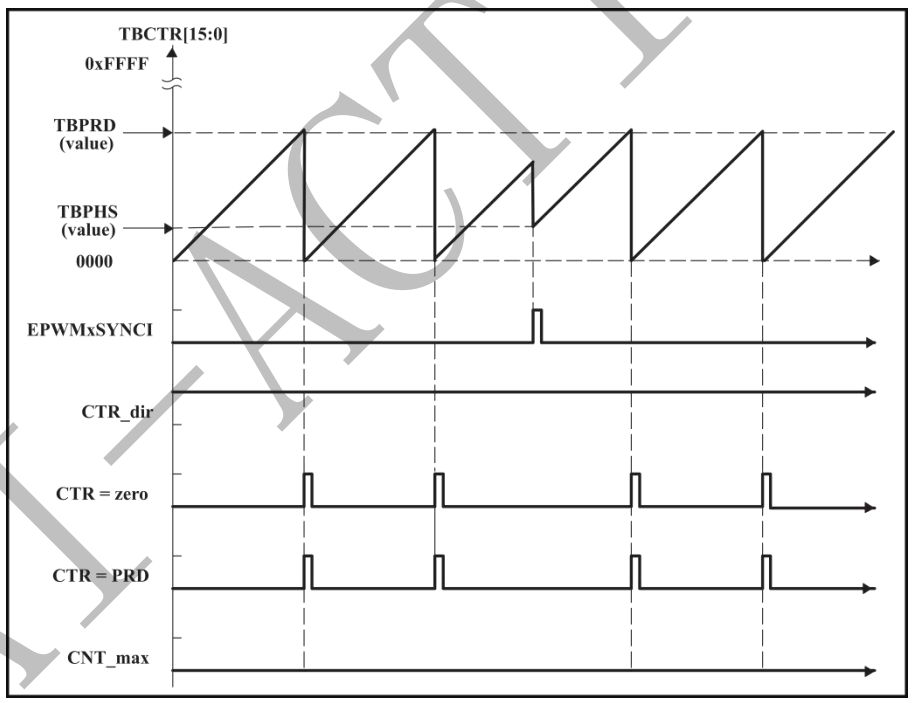


图 21-8　时基向上计数波形

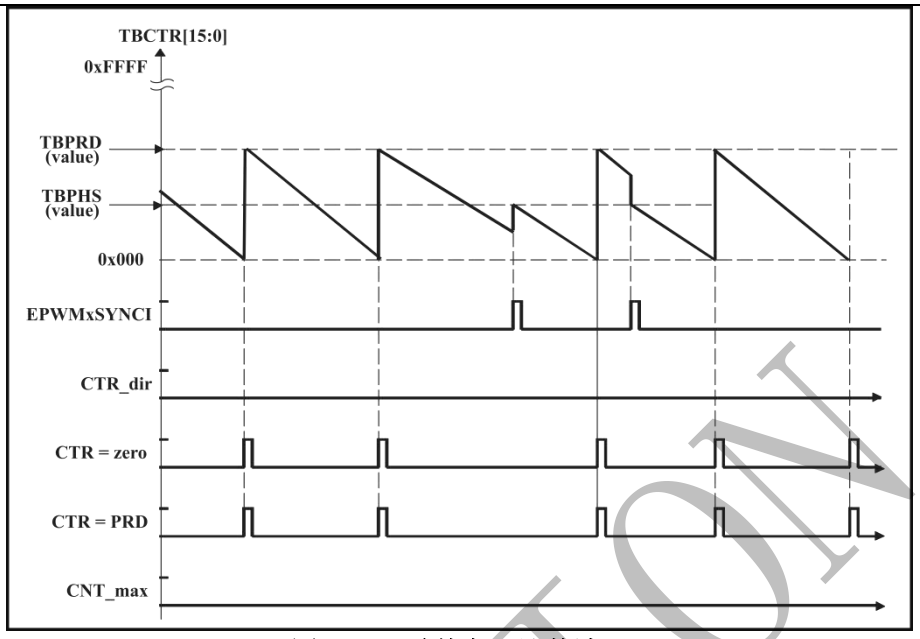


图 21-9　时基向下计数波形

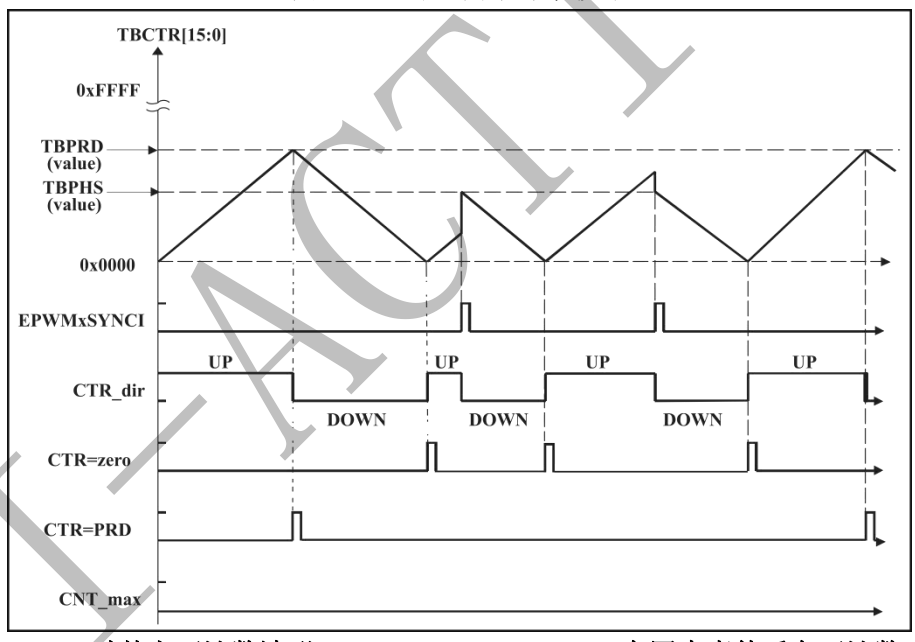


图 21-10　时基上下计数波形，TBCTL[PHSDIR = 0] 在同步事件后向下计数

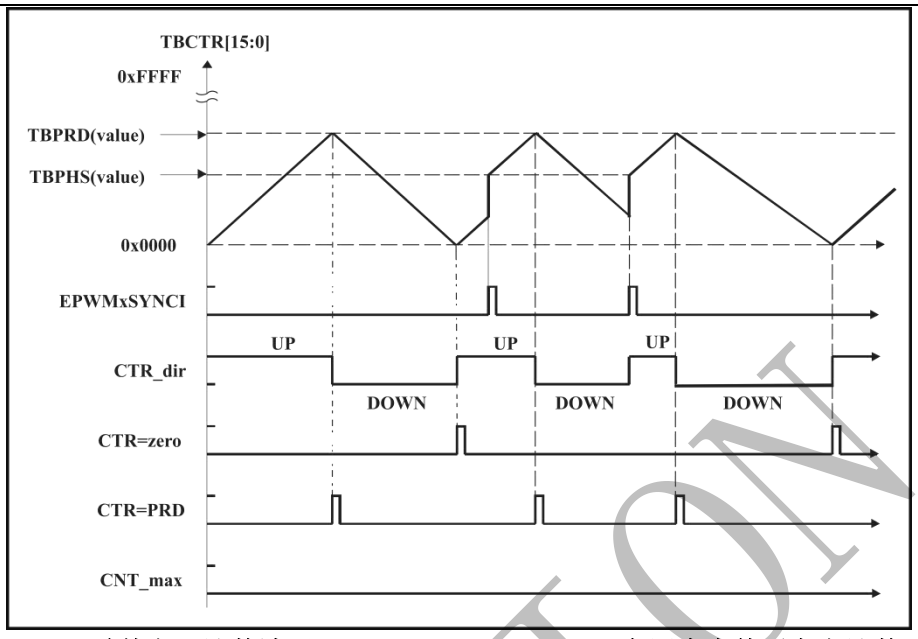


图 21-11　时基上下计数波形，TBCTL[PHSDIR = 1] 在同步事件后向上计数

21.4.1.3 计数比较（CC）子模块

图 21-12 说明了 ePWM 中的计数比较子模块。

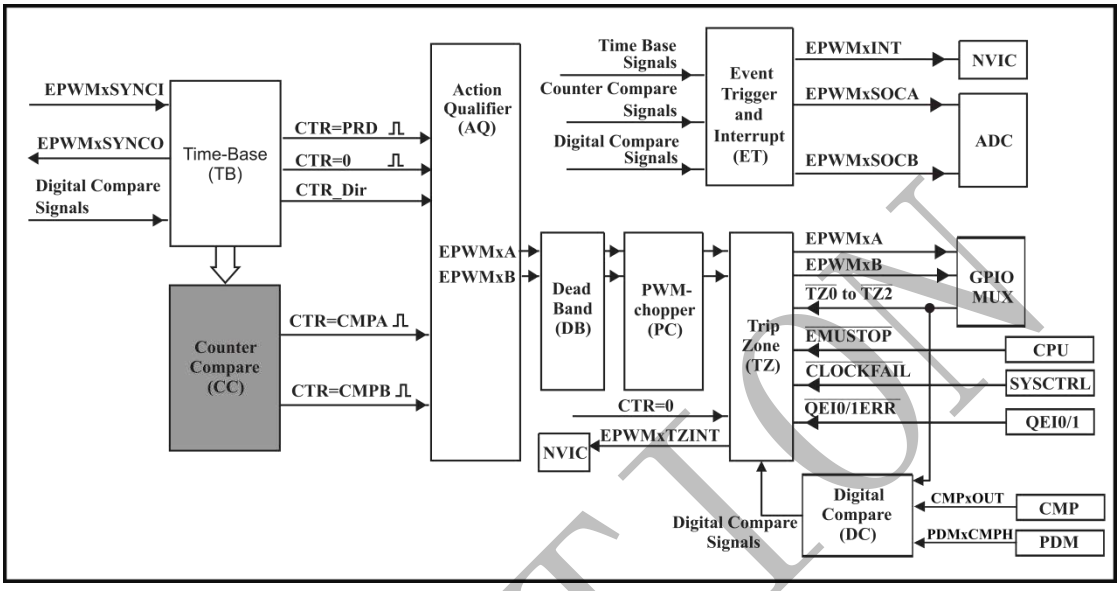


图 21-12　计数比较子模块

计数比较子模块的应用

计数比较子模块将时基计数器值作为输入,将该值与计数比较 A（CMPA）和计数比较 B（CMPB）寄存器进行比较,当该值等于比较寄存器之一时，计数比较单元会生成一个适当的事件。

计数比较：

- 使用 CMPA 和 CMPB 寄存器基于可编程时间生成事件
  - CTR = CMPA：时基计数器等于计数比较 A 寄存器（TBCTR = CMPA）
  - CTR = CMPB：时基计数器等于计数比较 B 寄存器（TBCTR = CMPB）
- 如果正确配置了动作限定符子模块，则控制 PWM 占空比
- 遮盖新的比较值，以防止在有效的 PWM 周期内出现损坏或毛刺

控制计数比较子模块

计数比较子模块的操作由表 21-4 中所示的寄存器控制。

表 21-4　计数比较寄存器

Register	Address offset	Shadowed	Description
CMPA	0x0020	YES	Counter-Compare A Register
CMPB	0x0024	YES	Counter-Compare B Register
CMPC	0x0028	YES	Counter-Compare C Register
CMPCTL	0x002C	NO	Counter-Compare Control Register

图 21-13 显示了计数比较子模块的基本结构。

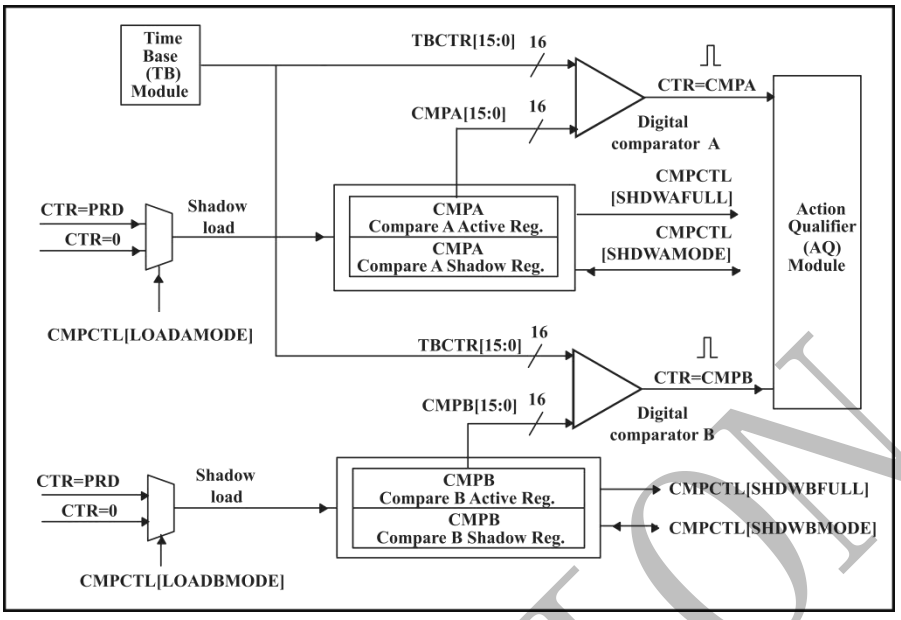


图 21-13　计数比较子模块的基本结构

表 21-5 中描述了与计数比较子模块关联的关键信号。

表 21-5　计数比较子模块关键信号

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-Compare A value	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the active counter-Compare B value	TBCTR = CMPB
CTR = PRD	Time-base counter equal to the active period. Used to load active counter-Compare A and B registers from the shadow register	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero. Used to load active counter-Compare A and B registers from the shadow register	TBCTR = 0x0000

计数比较子模块的详细功能

计数比较子模块负责基于两个比较寄存器生成两个独立的比较事件：

1. CTR = CMPA：时基计数器等于计数比较 A 寄存器（TBCTR = CMPA）
2. CTR = CMPB：时基计数器等于计数比较 B 寄存器（TBCTR = CMPB）

对于向上计数或向下计数模式，每个事件每个周期仅发生一次。对于上下计数模式，如果比较值介于 0x0000-TBPRD 之间，则每个事件每个周期发生两次；如果比较值等于 0x0000 或等于 TBPRD，则每个事件发生一次。这些事件被输入动作限定子模块，在该模块中，它们将由计数器方向限定，并转换为动作（如果启用）。

计数比较寄存器 CMPA 和 CMPB 都有一个关联的影子寄存器。影子提供了一种使寄存器更新与硬件同步的方法。使用影子时，仅在关键时刻才对动作限定进行更新。这样可以防止由于寄存器由软件异步修改而导致的损坏或伪造操作。动作限定和影子寄存器的存储地址相同。写入或读取哪个寄存器由 CMPCTL [SHDWAMODE]和 CMPCTL [SHDWBMODE]位确定。这些位分别启用和禁用 CMPA 影子寄存器和 CMPB 影子寄存器。两种负载模式的行为描述如下：

影子模式：

CMPA 的影子模式通过清除 CMPCTL [SHDWAMODE]位来启用，而 CMPB 的影子寄存器通过清除 CMPCTL [SHDWBMODE]位来启用。默认情况下，CMPA 和 CMPB 都启用了影子模式。

如果使能了影子寄存器，则影子寄存器的内容将在以下由 CMPCTL [LOADAMODE]和 CMPCTL [LOADBMODE]寄存器位指定的事件之一转移到有效寄存器：

- CTR = PRD：时基计数器等于周期（TBCTR = TBPRD）
- CTR = 0：时基计数器等于 0（TBCTR = 0x0000）
- CTR = PRD 和 CTR = 0

立即加载模式：

如果选择立即加载模式（TBCTL [SHADWAMODE] = 1 或 TBCTL [SHADWBMODE] = 1），则对寄存器的读取或写入将直接进入有效寄存器。

计数模式定时波形

计数比较模块可以在所有三种计数模式下生成比较事件：

- 向上计数模式：用于生成非对称 PWM 波形
- 向下计数模式：用于生成非对称 PWM 波形
- 上下计数模式：用于生成对称的 PWM 波形

为了最好地说明三种模式的操作，图 21-14 至图 21-17 中的时序图显示了何时产生事件以及 EPWMxSYNCI 信号如何相互作用。

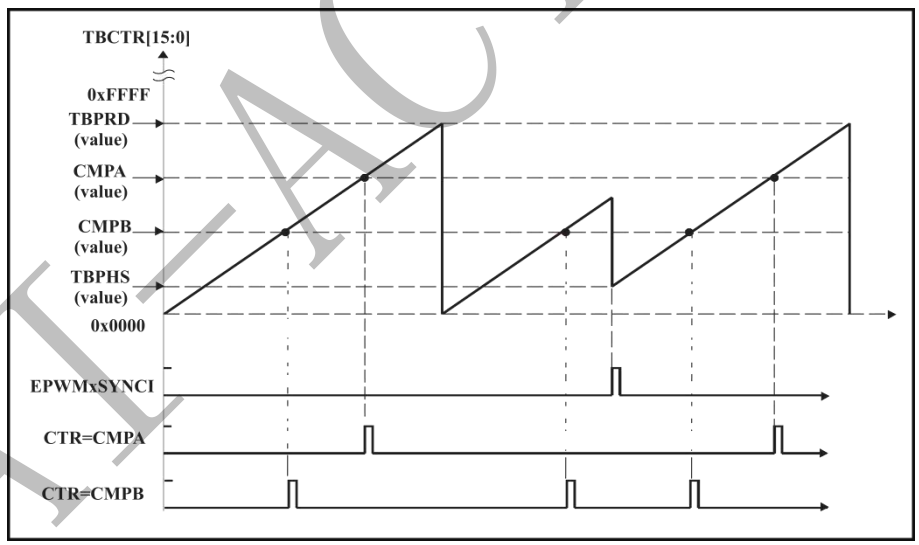


图 21-14　向上计数模式的计数比较事件

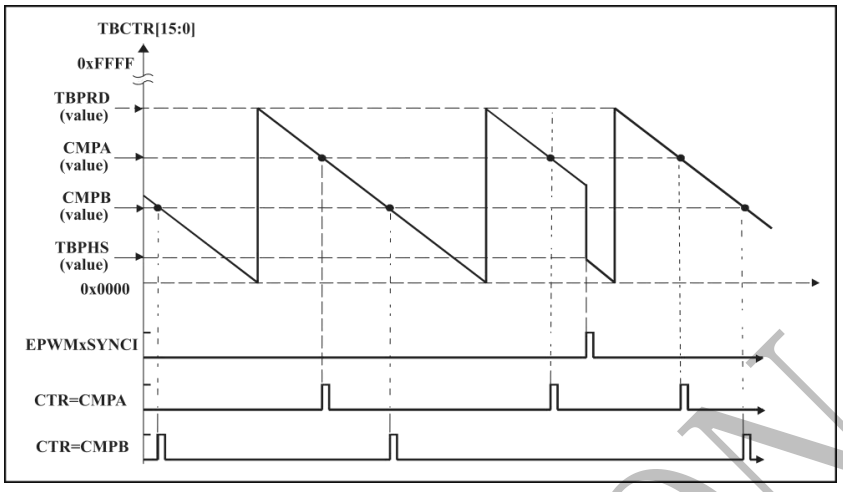


图 21-15　向下计数模式的计数比较事件

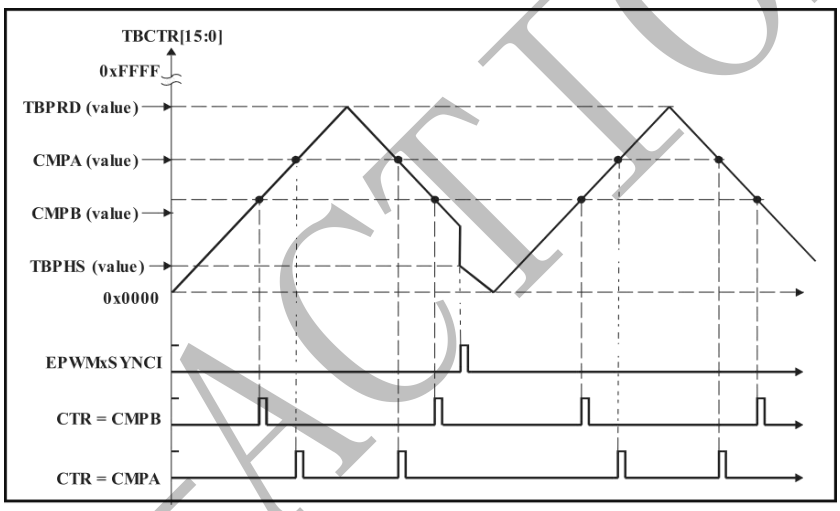


图 21-16　上下计数模式的计数比较事件，TBCTL[PHSDIR = 0] 在同步事件后向下计数

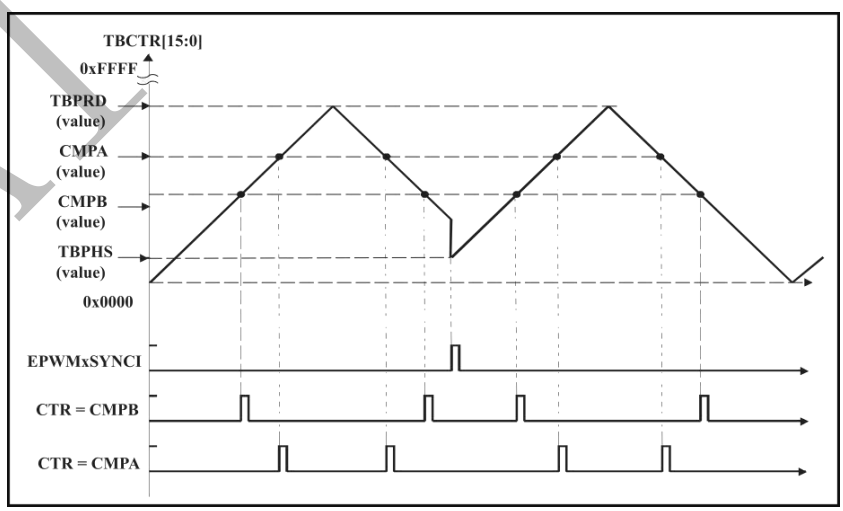


图 21-17　上下计数模式的计数比较事件，TBCTL[PHSDIR = 1] 在同步事件后向上计数

21.4.1.4 动作限定（AQ）子模块

图 21-18 显示了 EPWM 系统中的动作限定（AQ）子模块（请参见阴影块）。

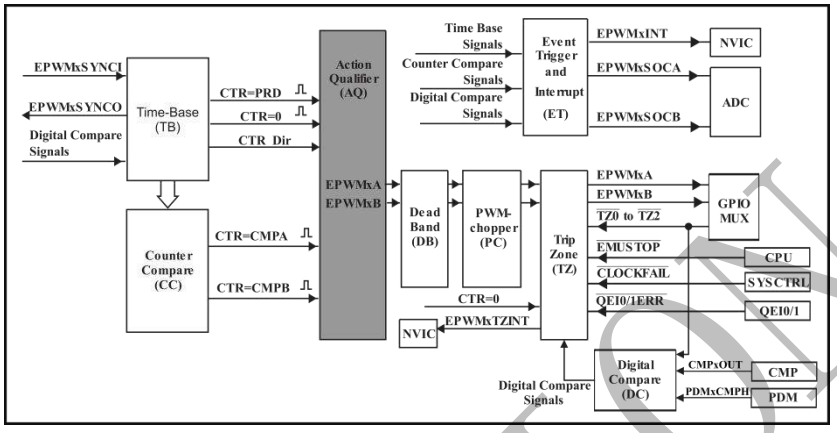


图 21-18　动作限定子模块

动作限定子模块在波形构建和 PWM 生成中具有最重要的作用。它决定将哪些事件转换为各种有效类型，从而在 EPWMxA 和 EPWMxB 输出上产生所需的开关波形。

动作限定子模块的目的

动作限定子模块负责以下各项：

- 根据以下事件来确定和生成动作（设置，清除，切换）：
  - CTR = PRD：与周期(TBCTR = TBPRD)相等的时基计数器。
  - CTR =0：时基计数器等于 0（TBCTR = 0x0000）
  - CTR = CMPA：时基计数器等于计数比较 A 寄存器（TBCTR = CMPA）
  - CTR = CMPB：时基计数器等于计数比较 B 寄存器（TBCTR = CMPB）
- 管理这些事件并发发生时的优先级
- 当时基计数器增加和减少时，提供对事件的独立控制

动作限定子模块控制和状态寄存器定义

动作限定子模块的操作通过表 21-6 中的寄存器进行控制。

表 21-6　动作限定子模块寄存器

Register	Address offset	Shadowed	Description
AQCTLA	0x0030	NO	Action-Qualifier Control Register For Output A (EPWMxA)
AQCTLB	0x0034	NO	Action-Qualifier Control Register For Output B (EPWMxB)
AQSFRC	0x0038	NO	Action-Qualifier Software Force Register
AQCSFRC	0x003C	YES	Action-Qualifier Continuous Software Force

动作限定子模块基于事件驱动的逻辑。可以将其视为可编程的十字开关，其输入事件和输出动作均由图 21-19 所示的一组寄存器进行软件控制。

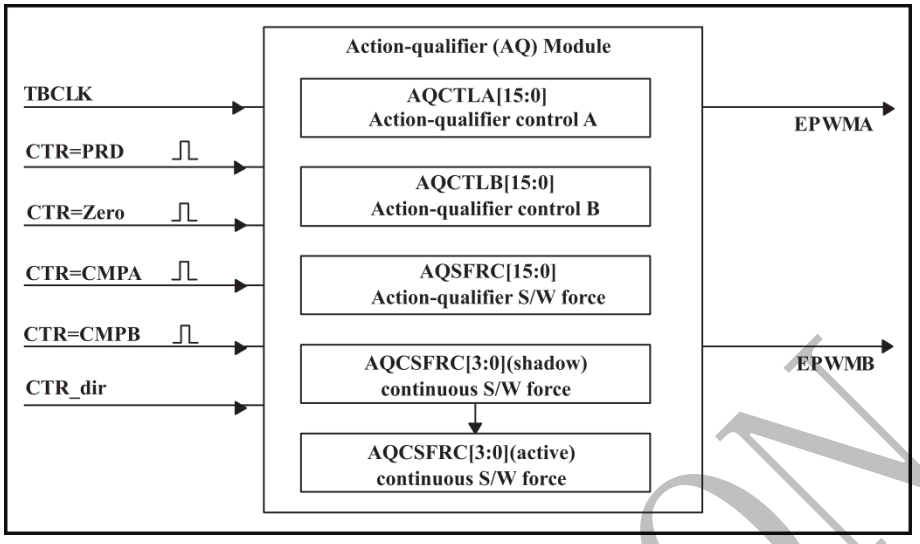


图 21-19　动作限定子模块输入输出

为方便起见，表 21-7 中再次汇总了可能的输入事件。

表 21-7　动作限定子模块可能的输入事件

Signal	Description of Event	Registers Compared
CTR = PRD	Time-base counter equal to the period value.	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero.	TBCTR = 0x0000
CTR = CMPA	Time-base counter equal to the active counter-Compare A value	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the active counter-Compare B value	TBCTR = CMPB
Software forced event	Asynchronous event initiated by software	

软件强制操作是一个有用的异步事件。这个控制由寄存器 AQSFRC 和 AQCSFRC 处理。

动作限定子模块控制发生特定事件时两个输出 EPWMxA 和 EPWMxB 的行为。输入到动作限定子模块的事件通过计数器方向（向上或向下）进一步限定。这允许在递增和递减阶段对输出进行独立操作。

对输出 EPWMxA 和 EPWMxB 可能施加的操作是:

- 设置为高
  将“输出 EPWMxA”或“EPWMxB”设置为高电平。
- 设置为低:
  将“输出 EPWMxA”或“EPWMxB”设置为低电平。
- 切换:
  如果 EPWMxA 或 EPWMxB 被拉高，则将输出拉低。
  如果 EPWMxA 或 EPWMxB 被拉低，则将输出拉高。
- 不执行任何操作:
  保持输出 EPWMxA 和 EPWMxB 与当前设置的级别相同。尽管“不执行任何操作”选项阻止事件导致对 EPWMxA 和 EPWMxB 输出的操作，但该事件仍然可以触发中断和 ADC 开始转换。

分别为任意一种输出(EPWMxA 或 EPWMxB)指定操作。所有事件都可以配置为在给定输出上生成操作。例如，CTR = CMPA 和 CTR = CMPB 都可以在输出 EPWMxA 上运行。所有限定操作都是通过在本节末尾找到的控制寄存器配置的。

为了清晰起见，本文档中的图纸使用了一组符号动作。这些符号汇总在图 21-20 中。每个符号代表一个动作作为时间上的标记。一些动作在时间上是固定的(零和周期)，而 CMPA 和 CMPB 动作是可移动的，它们的时间位置分别通过计数比较 A 和 B 寄存器编程。要关闭或禁用某个动作，请使用“不执行任何操作”选项;这是重置时的默认设置。

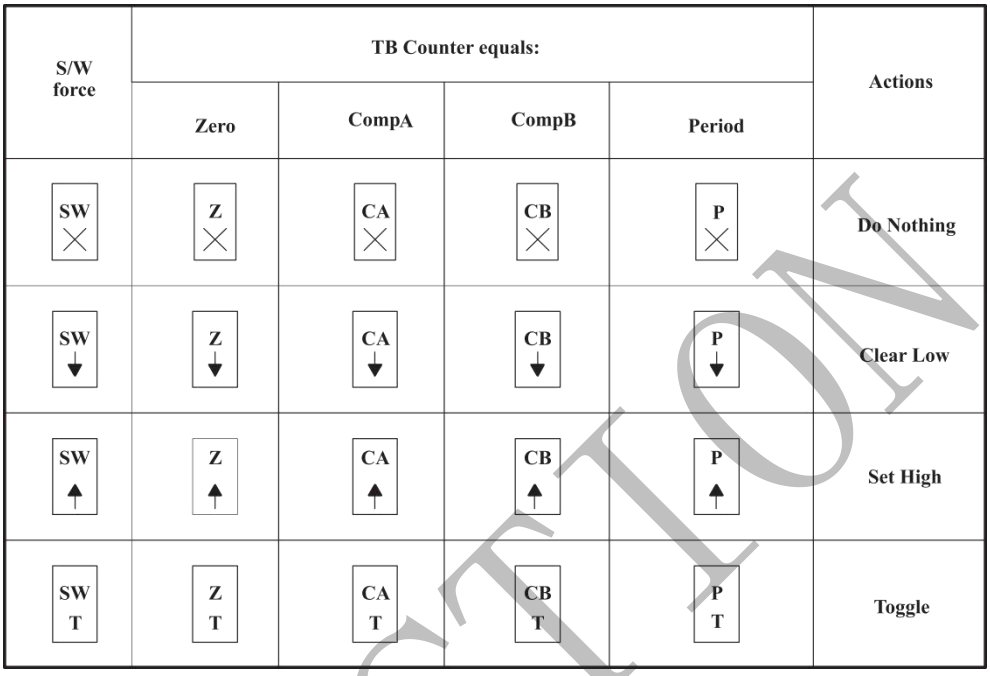


图 21-20　EPWMxA 和 EPWMxB 输出可能的动作限定动作

动作限定事件优先级

EPWM 动作限定可以同时接收多个事件。在这种情况下，事件由硬件分配优先级。一般的规则是时间较晚发生的事件具有较高的优先级，而软件强制事件总是具有最高的优先级。上下计数模式的事件优先级如表 21-8 所示。优先级为 1 的优先级最高，级别为 7 的优先级最低。根据 TBCTR 的方向，优先级略有变化。

表 21-8　上下计数模式下的动作限定事件优先级

Priority Level	Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD	Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR= 1
1(Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals Zero	Counter equals period (TBPRD)
5	Counter equals CMPB on down-count (CBD)	Counter equals CMPB on up-count (CBU)
6(Lowest)	Counter equals CMPA on down-count (CAD)	Counter equals CMPA on up-count (CAU)

表 21-9 显示了向上计数模式的动作限定优先级。在这种情况下，计数器方向始终定义为向上，因此永远不会发生向下计数事件。

表 21-9　向上计数模式的动作限定事件优先级

Priority Level	Event
1(Highest)	Software forced event
2	Counter equals period (TBPRD)

3	Counter equals CMPB on up-count (CBU)
4	Counter equals CMPA on up-count (CAU)
5(Lowest)	Counter equals Zero

表 21-10 显示了向下计数模式的动作限定优先级。在这种情况下，计数器方向始终定义为向下，因此永远不会发生向上计数事件。

表 21-10　向下计数模式的动作限定事件优先级

Priority Level	Event
1(Highest)	Software forced event
2	Counter equals Zero
3	Counter equals CMPB on down-count (CBD)
4	Counter equals CMPA on down-count (CAD)
5(Lowest)	Counter equals period (TBPRD)

可以将比较值设置为大于周期。在这种情况下，将按照表 21-11 所示执行操作。

表 21-11　CMPA / CMPB 大于周期时行为

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAD/CBD
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a Compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB > TBPRD, then the event will not occur.	Never occurs
Down-Count Mode	Never occurs	If CMPA/CMPB < TBPRD, the event will occur on a Compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCTR=TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a Compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCTR = TBPRD).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a Compare match (TBCTR=CMPA or CMPB). If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD)

常用配置波形

注意：本文档中的波形显示了静态比较寄存器值的 EPWM 行为。在运行的系统中，通常每个周期从其各自的影子寄存器更新一次有效比较寄存器（CMPA 和 CMPB）。用户指定更新的时间；当时基计数器达到零或时基计数器达到周期时。在某些情况下，基于新值的操作可能会延迟一个周期，或者基于旧值的操作可能会额外生效。一些 PWM 配置可以避免这种情况。其中包括但不限于以下内容：

使用上下计数模式生成对称 PWM：

- 如果将 CMPA / CMPB 加载为零，则使用大于或等于 1 的 CMPA / CMPB 值。
- 如果定期加载 CMPA / CMPB，则使用小于或等于 TBPRD-1 的 CMPA / CMPB 值。

这意味着在 PWM 周期中将始终存在至少一个 TBCLK 周期的脉冲，系统往往会忽略它。

使用上下计数模式生成不对称 PWM：

- 要实现 50%-0%的非对称 PWM，请使用以下配置：在周期上加载 CMPA / CMPB，使用周期动作清除 PWM，使用比较动作设置 PWM。将比较值从 0 调制到 TBPRD，以实现 50%-0%的 PWM 占空比。

使用向上计数模式生成不对称 PWM：

- 要实现 0-100%的非对称 PWM，请使用以下配置：在 TBPRD 上加载 CMPA / CMPB。使用零动作设置 PWM，并使用上比较动作清除 PWM。将比较值从 0 调制到 TBPRD + 1，以实现 0-100%PWM 占空比。

图 21-21 显示了如何使用 TBCTR 的上下计数模式生成对称 PWM 波形。在此模式下，通过对波形的向上计数和向下计数部分使用相等的比较匹配，可以实现 0%-100%的直流调制。在示例中，使用 CMPA 进行比较。当计数器递增时，CMPA 匹配会将 PWM 输出拉高。同样，当计数器递减时，比较匹配会将 PWM 信号拉低。当 CMPA = 0 时，PWM 信号在整个周期内为低电平，给出了 0%的占空比波形。当 CMPA = TBPRD 时，PWM 信号为高电平，从而实现 100%的占空比。

在实践中使用此配置时，如果将 CMPA / CMPB 加载为零，则使用大于或等于 1 的 CMPA / CMPB 值。如果按周期加载 CMPA / CMPB，则使用小于或等于 TBPRD -1 的 CMPA / CMPB 值。这意味着在 PWM 周期中将始终存在至少一个 TBCLK 周期的脉冲，系统往往会忽略它。

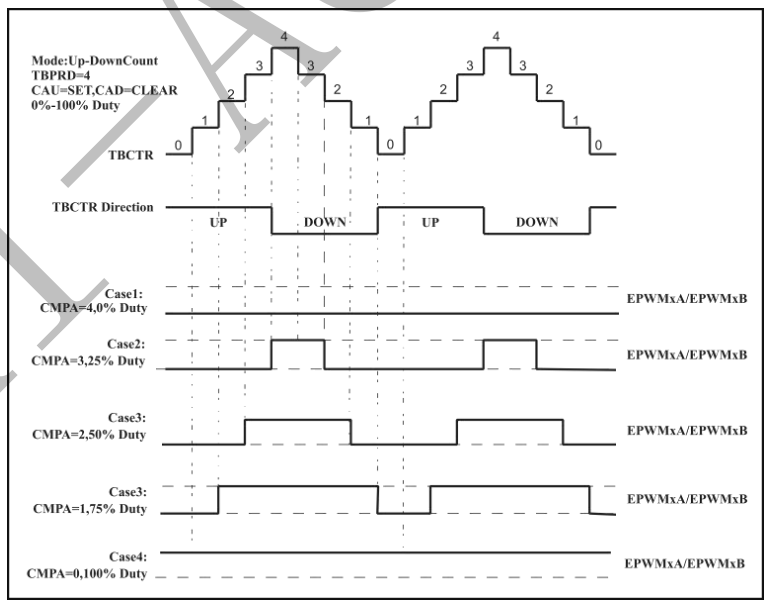


图 21-21　上下计数模式对称波形

图 21-22 到图 21-27 中的 PWM 波形显示了一些常见的动作限定配置。下图和示例中使用的一些约定如下：

- TBPRD、CMPA 和 CMPB 指写入各自寄存器的值。硬件使用的是它们各自的寄存器，而不是影子寄存器。
- CMPx，指 CMPA 或 CMPB。

- EPWMxA 和 EPWMxB 是 EPWMx 输出的信号
- Up-Down 表示向上和向下计数模式，Up 表示向上计数模式，Down 表示向下计数模式
- Sym =对称，Asym =非对称

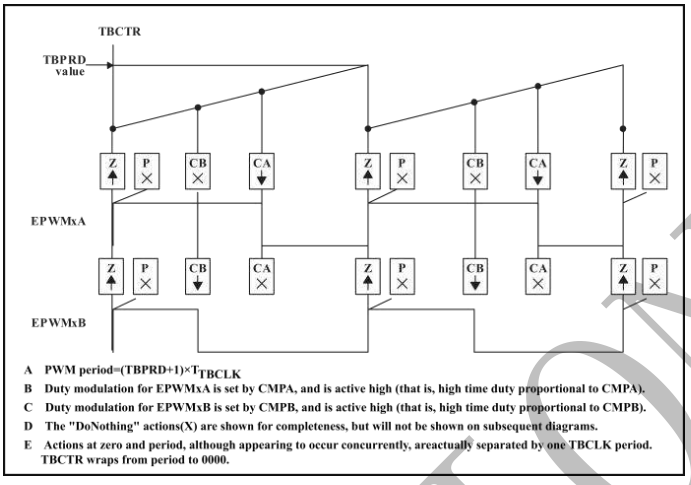


图 21-22 向上计数单沿非对称波形，在 EPWMxA 和 EPWMxB 上调制-高有效

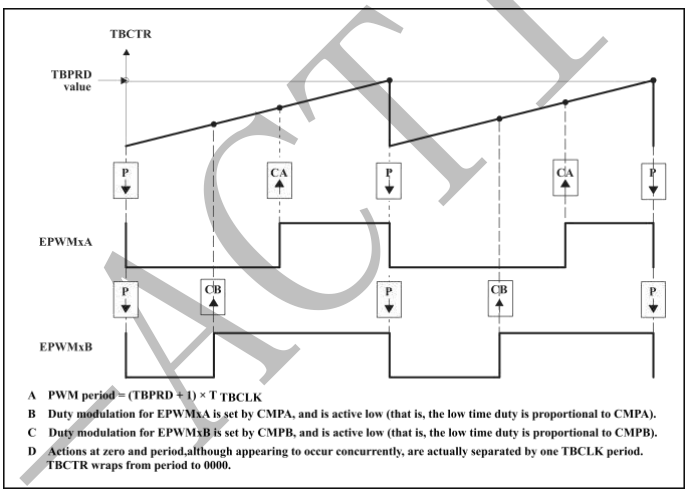


图 21-23 向上计数单沿非对称波形，在 EPWMxA 和 EPWMxB 上调制-低有效

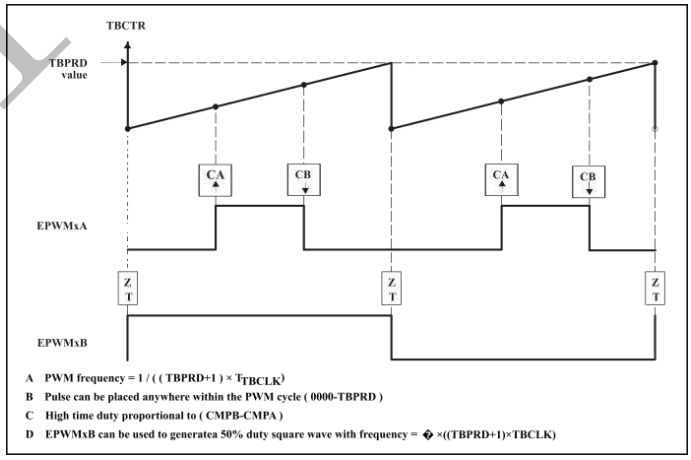


图 21-24 向上计数脉冲排布对称波形，在 EPWMxA 上单独调制

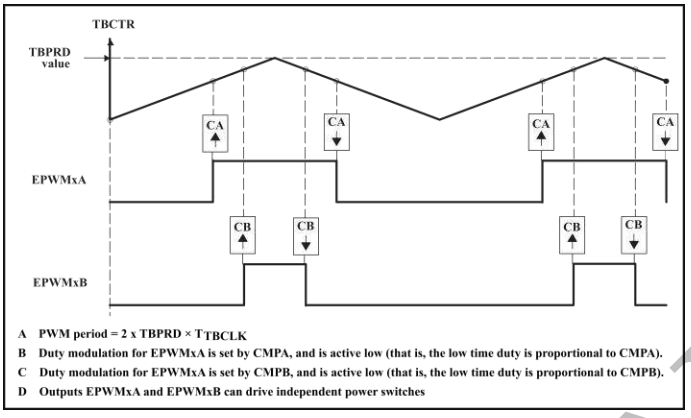


图 21-25 上下计数双沿对称波形，在 EPWMxA 和 EPWMxB 上调制-低有效

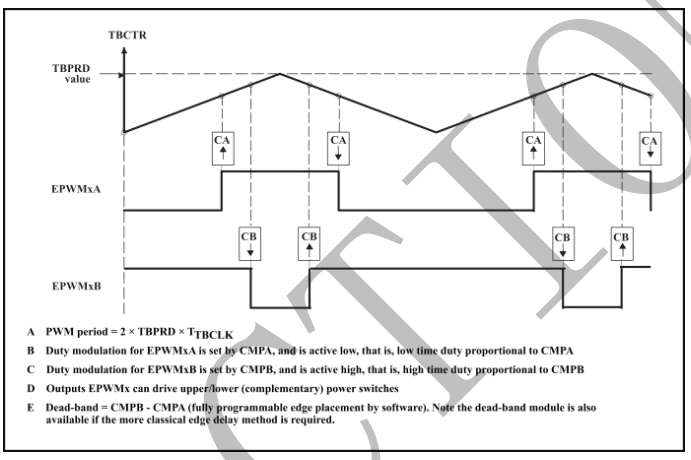


图 21-26 上下计数双沿对称波形，在 EPWMxA 和 EPWMxB 上调制-互补

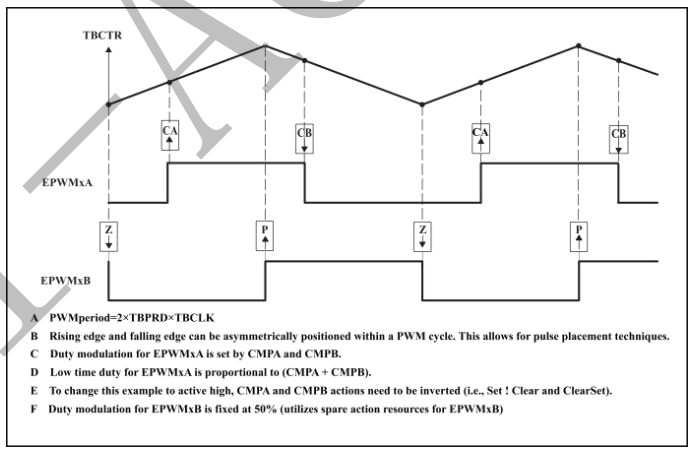


图 21-27 上下计数双沿非对称波形，在 EPWMxA 上单独调制-低有效

21.4.1.5 死区（DB）子模块

图 21-28 说明了 EPWM 模块中的死区子模块。

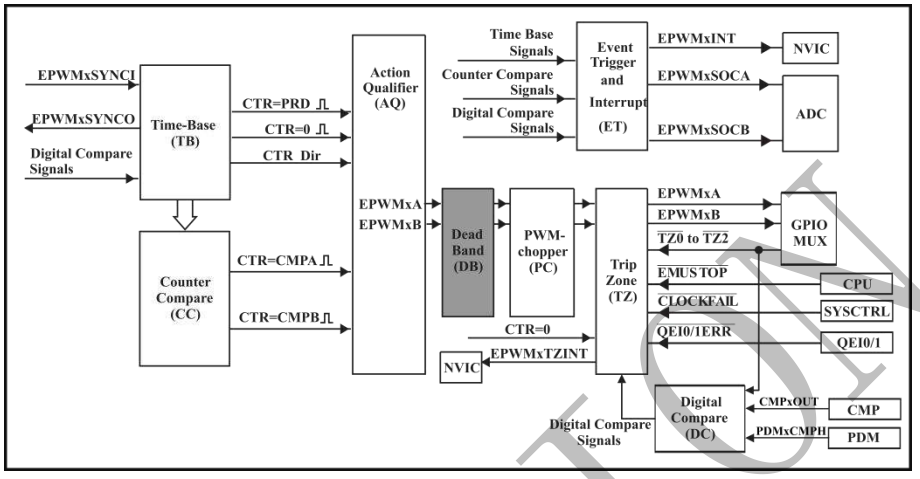


图 21-28 死区子模块

死区子模块的目的

“动作限定（AQ）模块”部分讨论了如何通过使用 EPWM 模块的 CMPA 和 CMPB 资源完全控制边缘位置来生成所需的死区。但是，如果需要带有极性控制的更为经典的基于边缘延迟的死区，则应使用此处描述的死区子模块。

死区模块的主要功能是：

- 从单个 EPWMxA 输入生成具有死区关系的恰当信号对（EPWMxA 和 EPWMxB）
- 可编程信号对包括：
  - 高电平有效（AH）
  - 低电平有效（AL）
  - 高互补有效（AHC）
  - 低互补有效（ALC）
- 在上升沿添加可编程延迟（RED）
- 在下降沿添加可编程延迟（FED）
- 可以完全绕开信号路径

死区子模块的控制

死区子模块的操作通过以下寄存器进行控制：

表 21-12 死区子模块寄存器

Register	Address offset	Shadowed	Description
DBCTL	0x0040	NO	Dead-Band Control Register
DBRED	0x0044	NO	Dead-Band Rising Edge Delay Count Register
DBFED	0x0048	NO	Dead-Band Falling Edge Delay Count Register

死区子模块的详细功能

以下部分提供了操作重点。

死区子模块具有两组独立的选择选项，如图 21-29 所示。

- 输入源选择：
  死区模块的输入信号是来自动作限定的 EPWMxA 和 EPWMxB 输出信号。在本节中，它们将被称为 EPWMxA In 和 EPWMxB In。使用 DBCTL [IN_MODE）控制位，可以选择每个延迟（下降沿或上升沿）的信号源：
  - EPWMxA In 是下降沿和上升沿延迟的来源。这是默认模式。
  - EPWMxA In 是下降沿延迟的来源，EPWMxB In 是上升沿延迟的来源。
  - EPWMxA In 是上升沿延迟的来源，EPWMxB In 是下降沿延迟的来源。
  - EPWMxB In 是下降沿和上升沿延迟的来源。
- 半周期时钟：
  死区子模块可以使用半周期时钟来加倍分辨率(即计数器时钟为 2×TBCLK)。
- 输出模式控制：
  通过 DBCTL [OUT_MODE]位配置输出模式。这些位确定是否将下降沿延迟，上升沿延迟，或两者都应用于输入信号。
- 极性控制：
  极性控制（DBCTL [POLSEL]）允许指定在将上升沿延迟信号和/或下降沿延迟信号发送出死区子模块之前是否要对其进行反转。

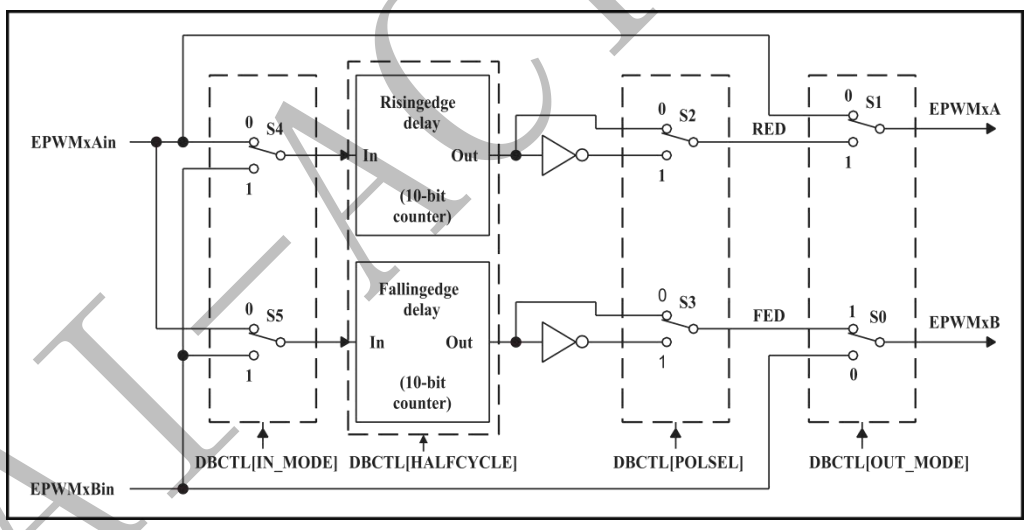


图 21-29　死区子模块可选配置

尽管支持所有组合，但并非所有都是典型的使用模式。表 21-13 列出了一些经典的死区配置。这些模式假定已配置 DBCTL [IN_MODE]，使得 EPWMxA In 是下降沿和上升沿延迟的源。可以通过更改输入信号源来实现增强或非传统模式。如表 21-13 所示，有以下几种模式：

- 模式 1：绕过下降沿延迟（FED）和上升沿延迟（RED）
  允许完全禁用 PWM 信号路径中的死区子模块。
- 模式 2-5：经典死区极性设置：
  这些代表典型的极性配置，应解决可用的工业电源开关栅极驱动器要求的所有高/低互补有效模式。这些典型情况的波形如图 21-30 所示。请注意，为了生成与图 21-30 等效的波形，请配置动作限定子模块以生成信号，如 EPWMxA 所示。
- 模式 6：绕过上升沿延迟，模式 7：绕过下降沿延迟

最后，表 21-13 中的最后两个条目显示了组合，其中绕过了下降沿延迟（FED）或上升沿延迟（RED）块。

表 21-13　经典死区操作模式

Mode	Mode Description	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	X	X	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxA In with No Delay	0 or 1	0 or 1	1	0

图 21-30 给出了占空比为 0%-100%的典型情况的波形。

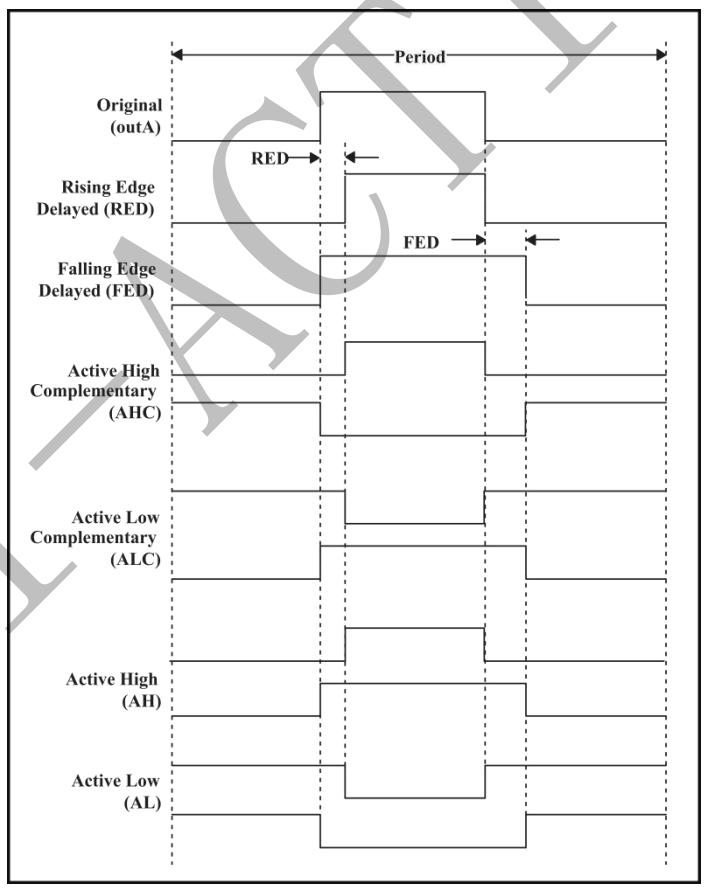


图 21-30　典型情况（0%<占空比<100%）的死区配置

死区子模块是支持上升沿（RED）和下降沿（FED）延迟的值。延迟量是使用 DBRED 和 DBFED 寄存器编程的。它们是 10 位寄存器，其值表示信号沿被延迟的时基时钟 TBCLK 的周期。例如，计算下降沿延迟和上升沿延迟的公式为：

$FED = DBFED \times T_{TBCLK}$

RED = DBRED × T_{TBCLK}

其中 T_{TBCLK} 是 TBCLK 的周期，即 EPWMCLK 的预分频版本。

启用半周期时钟后，用于计算下降沿延迟和上升沿延迟的公式变为：

FED = DBFED × T_{TBCLK}/2

RED = DBRED × T_{TBCLK}/2

21.4.1.6 PWM 斩波（PC）子模块

图 21-31 说明了 EPWM 模块中的 PWM 斩波（PC）子模块。

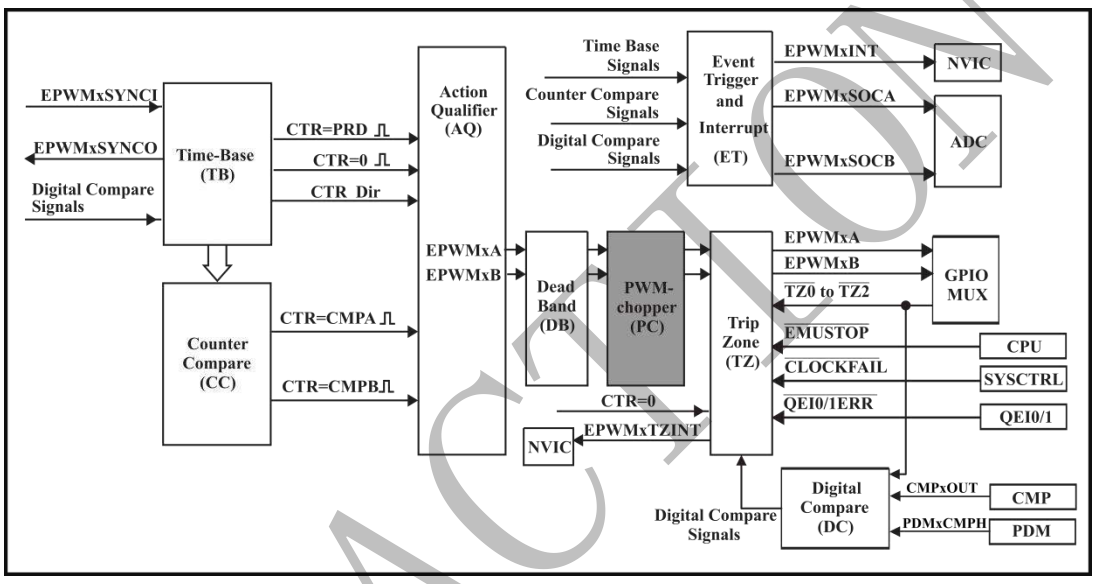


图 21-31 PWM 斩波子模块

PWM 斩波子模块允许高频载波信号调制由动作限定和死区子模块生成的 PWM 波形。如果需要基于脉冲变压器的栅极驱动器来控制电源开关元件，则此功能非常重要。

PWM 斩波子模块的目的

PWM 斩波器子模块的关键功能是：

- 可编程斩波（载波）频率
- 第一个脉冲的可编程脉冲宽度
- 第二个后续脉冲的可编程占空比
- 如果不需要，可以完全忽略

控制 PWM 斩波子模块

PWM 斩波子模块的操作由表 21-14 中的寄存器控制。

表 21-14 PWM 斩波子模块寄存器

mnemonic	Address offset	Shadowed	Description
PCCTL	0x0050	NO	PWM-chopper Control Register

PWM 斩波子模块的详细功能

图 21-32 显示了 PWM 斩波器子模块的详细功能。载波时钟来自 EPWMCLK。它的频率和占空比是通过 PCCTL 寄存器中的 CHPFREQ 和 CHPDUTY 位控制的。单发模块是一种功能，可提供能量较高的首个脉冲以确保硬件开关和快速电源开关打开，而随后的脉冲维持脉冲，以确保电源开关保持导通。单发脉冲宽度通过 OSHTWTH 位编程。可以通过 CHPEN 位完全禁用（忽略）PWM 斩波器子模块。

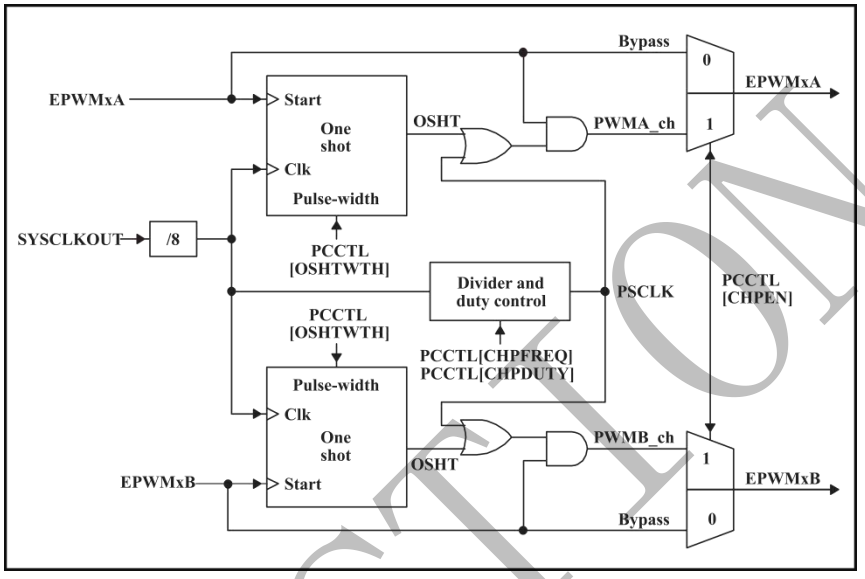


图 21-32　PWM 斩波子模块细节选项

波形

图 21-33 仅显示了斩波动作的简化波形。单发脉冲和占空比控制未显示。在以下各节中将讨论单发脉冲和占空比控制的详细信息

单发脉冲

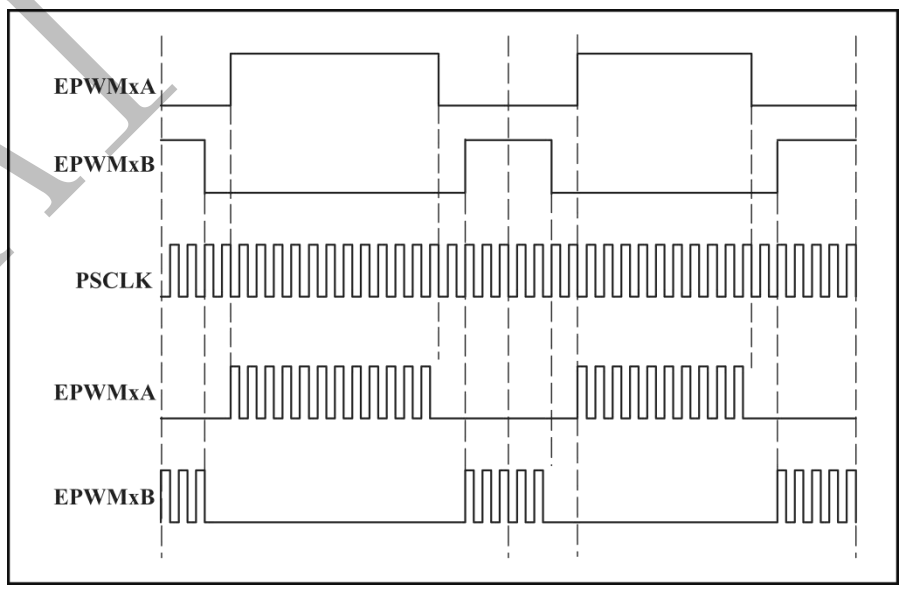


图 21-33　仅显示斩波动作的简单 PWM 斩波子模块波形

可以将第一个脉冲的宽度编程为 16 种可能的脉冲宽度值中的任何一个。第一个脉冲的宽度或周期由下式给出：

$T_{1stpulse} = T_{EPWMCLK} \times 16 \times OSHTWTH$

其中 TEPWMCLK 是系统时钟的周期（EPWMCLK），OSHTWTH 是四个控制位（值从 1 到 16）

图 21-34 显示了第一个和后续的持续脉冲，表 21-15 给出了 EPWMCLK =160MHz 时可能的脉冲宽度值。

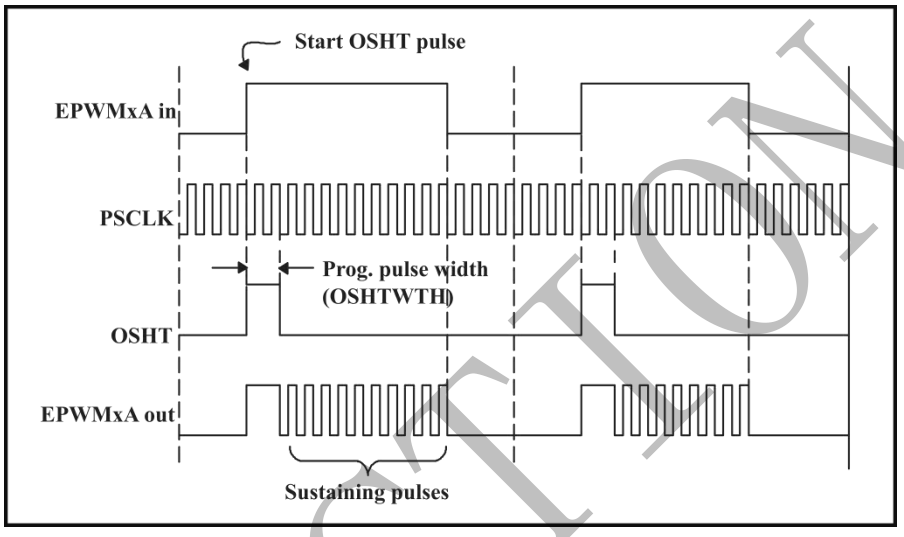


图 21-34　显示首个脉冲和后续保持脉冲的 PWM 斩波器子模块波形

表 21-15　EPWMCLK = 160 MHz 时可能的脉冲宽度值

OSHTWTHz(hex)	Pulse Width(nS)
0	100
1	200
2	300
3	400
4	500
5	600
6	700
7	800
8	9000
9	1000
A	1100
B	1200
C	1300
D	1400
E	1500
F	1600

占空比控制

基于脉冲变压器的栅极驱动器设计需要理解变压器和相关电路的磁性或特性。饱和度就是这样的考虑因素之一。为了帮助栅极驱动器设计人员，第二个及后续脉冲的占空比已被编程。这些持续脉冲可确保导通期间在电源开关栅极上保持正确的驱动强度和极性，因此可编程占空比允许通过软件控制来调整或优化设计。

图 21-35 显示了可以通过对 CHPDUTY 位进行编程来实现的占空比控制。可以选择七个可能的占空比之一，范围为 12.5%至 87.5%。

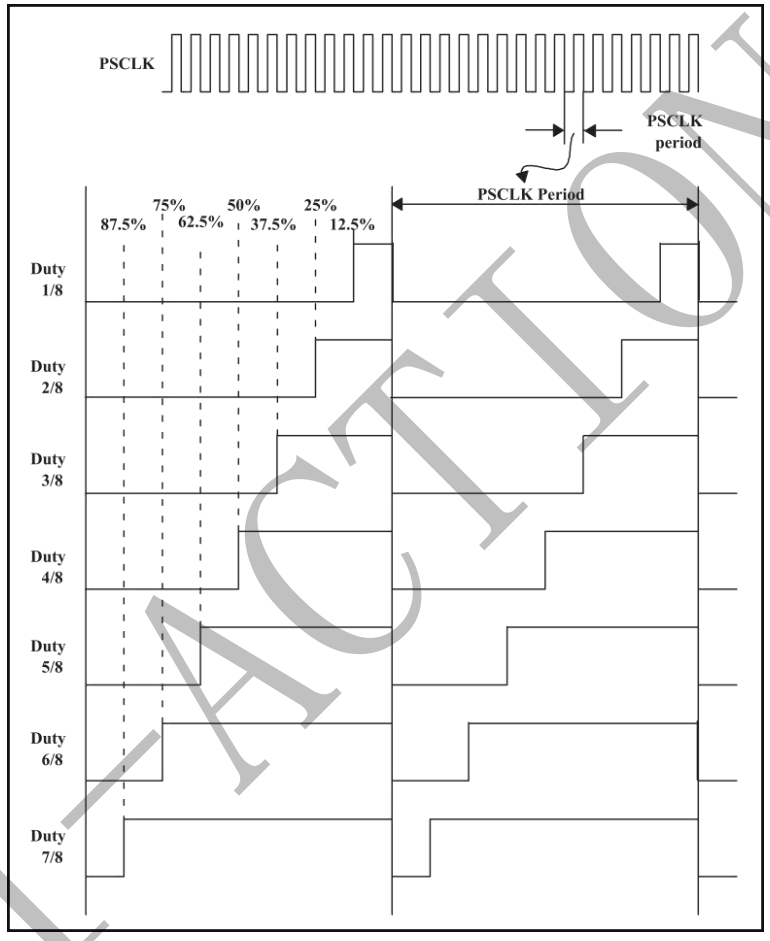


图 21-35　显示保持脉冲的脉冲宽度（占空比）控制的 PWM 斩波子模块波形

21.4.1.7 关断区域（TZ）子模块

图 21-36 显示了关断区域（TZ）子模块如何放置在 EPWM 模块中。

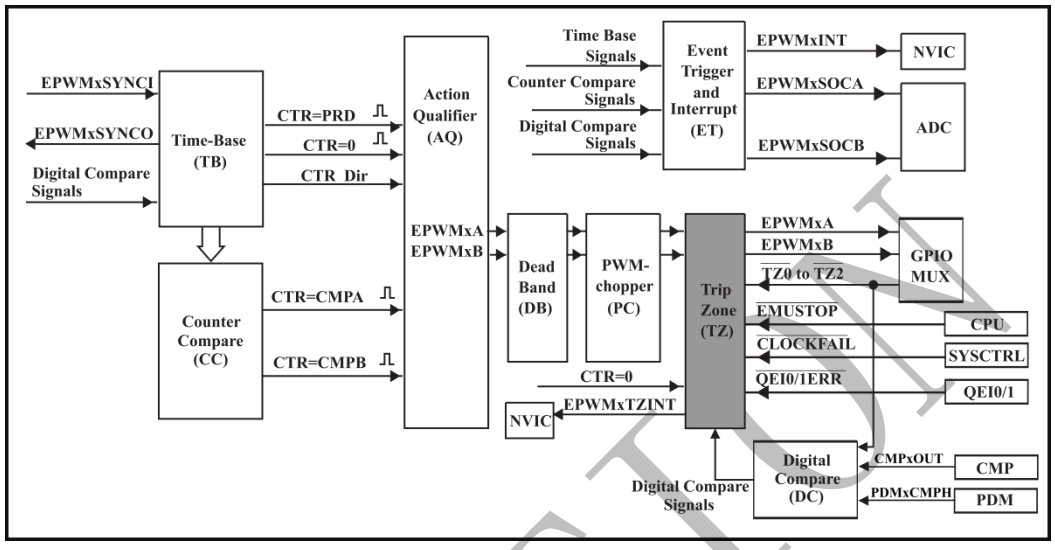


图 21-36 关断区域子模块

每个 EPWM 模块都连接到六个 $\overline{TZn}$ 信号（$\overline{TZ0}$ 至 $\overline{TZ5}$）。$\overline{TZ0}$ 至 $\overline{TZ2}$ 来自 GPIO 多路复用器。$\overline{TZ3}$ 来源于 QEI 模块的 QEI0/1ERR 反相信号。$\overline{TZ4}$ 连接到系统时钟故障逻辑，并且 $\overline{TZ5}$ 来自 CPU 的 EMUSTOP 输出。这些信号指示外部故障或关断条件，并且可以对 EPWM 输出进行编程以在发生故障时做出相应的响应。

关断区域子模块的目的

关断区域子模块的主要功能有:

- 触发输入 $\overline{TZ0}$ 至 $\overline{TZ5}$ 可以灵活地映射到任何 EPWM 模块。
- 在发生故障时，可以将输出 EPWMxA 和 EPWMxB 强制为以下状态之一：
  - 高
  - 低
  - 高阻抗
  - 不执行任何操作
- 在主要的小型电路或过流环境下支持单出错操作。
- 支持电流限制下的周期出错操作（CBC）。
- 支持基于片上模拟比较器模块输出和/或 $\overline{TZ0}$ 至 $\overline{TZ2}$ 信号的状态的数字比较跳闸(DC)。
- 每个触发区输入和数字比较(DC)子模块 DCAEVT1/2 或 DCBEVT1/2 事件可以分配给单次或循环操作。
- 在任何触发区管脚都可能产生中断。
- 还支持软件强制出错。
- 如果不需要，触发区子模块可以完全忽略。

关断区域子模块的详细功能

关断区域子模块的操作通过以下寄存器进行控制。

表 21-16 关断区域子模块寄存器

Register	Address offset	Shadowed	Description
TZSEL	0x0060	NO	Trip-Zone Select Register
TZCTL	0x0064	NO	Trip-Zone Control Register
TZEINT	0x0068	NO	Trip-Zone Enable Interrupt Register
TZFLG	0x006C	NO	Trip-Zone Flag Register
TZCLR	0x0070	NO	Trip-Zone Clear Register
TZFRC	0x0074	NO	Trip-Zone Clear Register
TZDCSEL	0x0078	NO	Trip-Zone Digital Compare Select Register

关断区域子模块的操作概述

以下各节介绍了关断区域子模块的操作重点和配置选项。

关断区域信号 $\overline{TZ0}$ 至 $\overline{TZ5}$（也统称为 $\overline{TZn}$）是低电平有效输入信号。当其中一个信号变低时，或者根据 TZDCSEL 寄存器事件选择发生 DCAEVT1/2 或 DCBEVT1/2 时，表明发生了触发事件。每个 EPWM 模块可以单独配置为忽略或使用每个关断区域信号或 DC 事件。特定 EPWM 模块使用哪个关断区域信号或 DC 事件由该特定 EPWM 模块的 TZSEL 寄存器确定。关断区域信号可能与系统时钟（EPWMCLK）同步，也可能不同步，并在 GPIO MUX 块内进行数字滤波。$\overline{TZn}$ 输入的最小 3 * TBCLK 低脉冲宽度足以触发 EPWM 模块上的故障条件。如果脉冲宽度小于此宽度，则关断条件可能不会被 CBC 或 OST 锁存器锁存。异步触发可确保如果由于任何原因缺少时钟，则仍可通过 $\overline{TZn}$ 输入上存在的有效事件来使输出触发。GPIO 或外设必须正确配置。

每个 $\overline{TZn}$ 输入可单独配置为 EPWM 模块提供逐周期或单次触发事件。可以将 DCAEVT1 和 DCBEVT1 事件配置为直接使 EPWM 模块触发或向该模块提供一次触发事件。同样，DCAVET2 和 DCBEVT2 事件也可以配置为直接使 EPWM 模块触发或向模块提供逐周期触发事件。此配置分别由 TZSEL [DCAEVT1/2]，TZSEL [DCBEVT1/2]，TZSEL [CBCn]和 TZSEL [OSHTn]控制位（其中 n 对应于关断输入）确定。

- 逐周期

当发生逐周期触发事件时，将立即在 EPWMxA 和/或 EPWMxB 输出上执行 TZCTL [TZA]和 TZCTL [TZB]位中指定的操作。表 21-17 列出了可能的操作。另外，逐周期关断事件标志（TZFLG[CBC]）被置位，如果在 TZEINT 寄存器和 PIE 外设中使能了 TZINT 中断，则将产生该中断。如果通过 TZEINT 寄存器启用了 CBC 中断，并且通过 TZSEL 寄存器选择了 DCAEVT2 或 DCBEVT2 作为 CBC 中断源，则无需在 TZEINT 寄存器中也启用 DCAEVT2 或 DCBEVT2 中断，因为 DC 事件通过 CBC 机制触发中断。如果不再存在触发事件，则当 EPWM 时基计数器达到零（TBCTR=0x0000）时，将自动清除输入上的指定条件。因此，在该模式下，每个 PWM 周期都会清除或重置触发事件。TZFLG [CBC]标志位将保持置位状态，直到通过写入 TZCLR [CBC]位将其手动清除为止。如果将 TZFLG [CBC]位清零时，循环触发事件仍然存在，则它将立即被重新设置。

- 单次触发

当单次触发事件发生时，在 EPWMxA 和/或 EPWMxB 输出上立即执行 TZCTL[TZA]和 TZCTL[TZB]位中指定的操作。可能的操作如表 21-17 所示。

此外，如果在 TZEINT 寄存器和 PIE 外设中使能了单次触发事件标志（TZFLG [OST]），则将会产生 TZINT 中断。必须通过写入 TZCLR [OST]位来手动清除单关断条件。

如果通过 TZEINT 寄存器启用一次触发中断，并且通过 TZSEL 寄存器选择 DCAEVT1 或 DCBEVT1 作为 OSHT 中断源，则不必在 TZEINT 寄存器中也启用 DCAEVT1 或 DCBEVT1 中断。因为 DC 事件通过 OSHT 机制触发中断。

- 数字比较事件（DCAEVT1/2 和 DCBEVT1/2）：

根据 TZDCSEL 寄存器选择的 DCAH/DCAL 和 DCBH/DCBL 信号的组合，将生成数字比较 DCAEVT1/2 或 DCBEVT1/2 事件。通过 DCTRIPSEL 寄存器选择产生 DCAH / DCAL 和 DCBH / DCBL 信号的信号，这些信号可以是关断区域输入引脚，也可以是模拟比较器 CMPxOUT 信号。

当发生数字比较事件时，在 EPWMxA 和/或 EPWMxB 输出上执行 TZCTL [DCAEVT1/2]和 TZCTL [DCBEVT1/2]位中指定的操作。表 21-17 列出了可能的操作。此外，如果在 TZEINT 寄存器和 PIE 外设中启用了相关的 DC 关断事件标志（TZFLG [DCAEVT1/2] / TZFLG [DCBEVT1/2]），则将产生 TZINT 中断。

当不再存在 DC 触发事件时，引脚上的指定条件将自动清除。TZFLG [DCAEVT1/2]或 TZFLG [DCBEVT1/2]标志位将保持置位状态，直到通过写入 TZCLR [DCAEVT1/2]或 TZCLR [DCBEVT1/2]位将其手动清除为止。如果清除 TZFLG [DCAEVT1/2]或 TZFLG [DCBEVT1/2]标志时，直流关断事件仍然存在，那么它将立即被重新设置。

可以通过 TZCTL 寄存器位字段为每个 EPWM 输出引脚分别配置发生关断事件时采取的措施。如表 21-17 所示，可以对关断事件采取四种可能的措施之一。

表 21-17　对关断事件可能的操作

TZCTL Register bit-field Settings	EPWMxA and/or EPWMxB	Comment
0,0	High-Impedance	Tripped
0,1	Force to High State	Tripped
1,0	Force to Low State	Tripped
1,1	No Change	Do Nothing. No change is made to the output

产生触发事件中断

图 21-37 和图 21-38 分别说明了关断区域子模块的控制和中断逻辑。

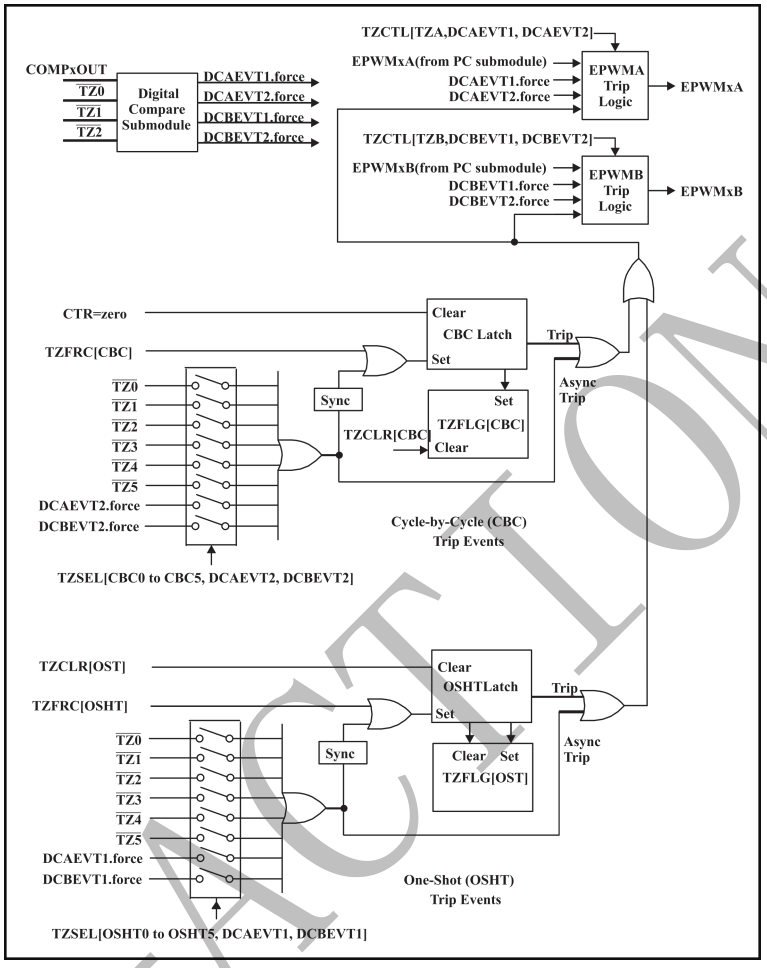


图 21-37　关断区域子模块模式控制逻辑

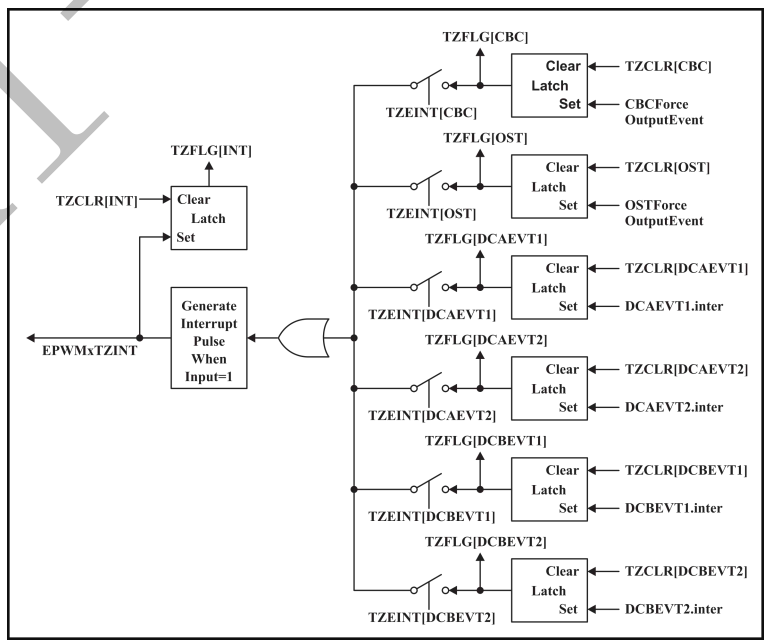


图 21-38　关断区域子模块中断逻辑

21.4.1.8 事件触发（ET）子模块

事件触发子模块的主要功能：

- 接收由时基，计数比较和数字比较子模块生成的事件输入
- 使用时基方向信息进行上/下事件过滤
- 使用预分频逻辑在以下位置发出中断请求和 ADC 转换开始：
  - 每一个事件
  - 每两个事件
  - 每三个事件
- 通过事件计数器和标志提供事件生成的完整可见性
- 允许软件强制中断和 ADC 开始转换

事件触发子模块由时基子模块，计数比较和数字比较子模块生成的事件控制，在发生选定事件时生成 CPU 的中断或 ADC 的转换脉冲。图 21-39 说明了事件触发子模块在 EPWM 系统中的适合位置。

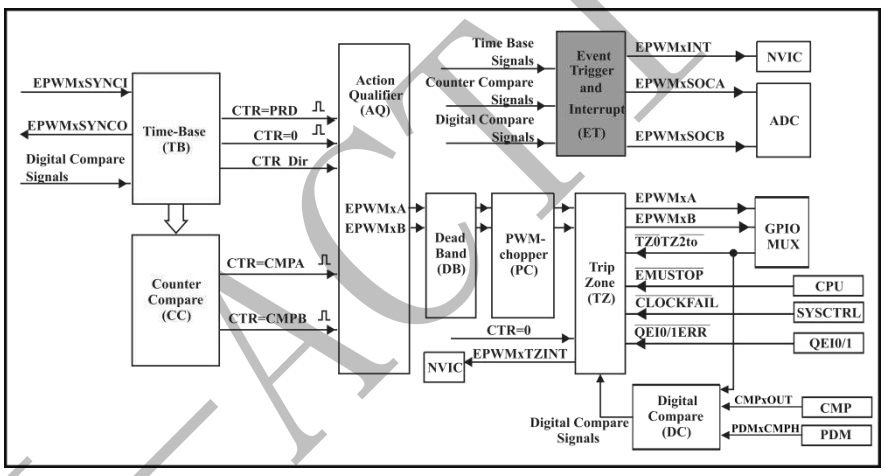


图 21-39　事件触发子模块

事件触发子模块的操作概述

以下各节描述了事件触发子模块的操作重点。每个 EPWM 模块都有一条连接到 NVIC 的中断请求线和两个连接到 ADC 模块的转换开始信号。如图 21-40 所示，所有 EPWM 模块的 ADC 转换开始都连接到 ADC 的各个 ADC 触发输入，因此多个模块可以通过 ADC 触发输入启动 ADC 转换开始。

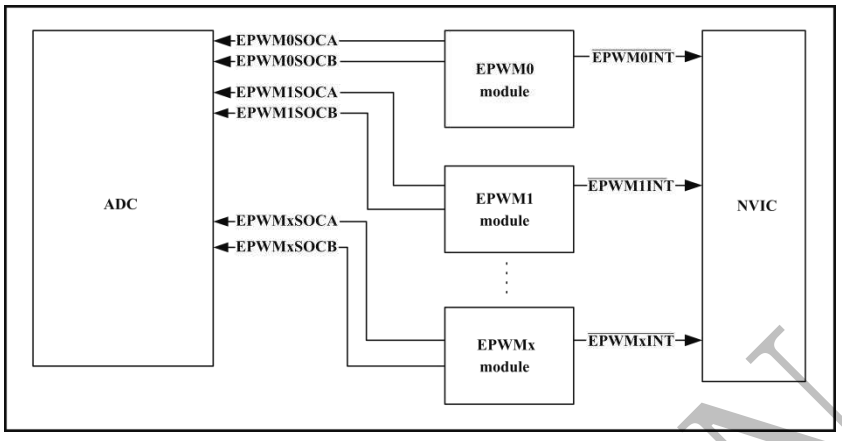


图 21-40　事件触发子模块与 ADC SOC 连接关系

事件触发子模块监视各种事件条件（图 21-41 所示的事件触发子模块的左侧输入），并且可以配置为在发出中断请求或 ADC 转换开始之前对这些事件进行预分频。事件触发预分频逻辑可以在以下位置发出中断请求和 ADC 转换开始：

- 每一个事件
- 每两个事件
- 每三个事件

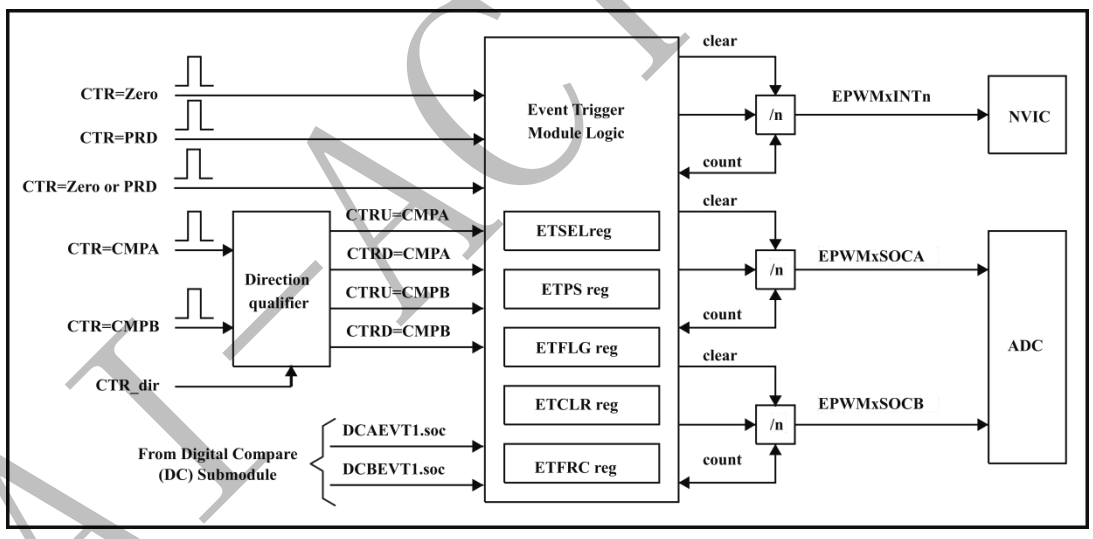


图 21-41　事件触发子模块，显示了输入事件及预分频输出

表 21-18 中显示了用于配置事件触发子模块的关键寄存器：

表 21-18 事件触发子模块寄存器

Register	Address offset	Shadowed	Description
ETSEL	0x00B0	NO	Event-trigger Selection Register
ETPS	0x00B4	NO	Event-trigger Prescale Register
ETFLG	0x00B8	NO	Event-trigger Flag Register
ETCLR	0x00BC	NO	Event-trigger Clear Register
ETFRC	0x00C0	NO	Event-trigger Force Register

- ETSEL：选择哪些可能的事件将触发中断或开始 ADC 转换
- ETPS：用于对上述事件预缩放选项进行编程。
- ETFLG：这些标志位指示选定事件和预缩放事件的状态。
- ETCLR：这些位允许您通过软件清除 ETFLG 寄存器中的标志位。
- ETFRC：这些位允许软件强制事件。对于调试或软件干预很有用。

图 21-42，图 21-43 和图 21-44 显示了各种寄存器如何与中断和 ADC 转换开始逻辑交互的详细信息。

图 21-42 显示了事件触发的中断产生逻辑。中断周期（ETPS[INTPRD]）位指定导致产生中断脉冲所需的事件数。可供选择的有:

- 不产生中断。
- 在每一个事件上生成一个中断
- 在每两个事件上生成一个中断
- 在每三个事件上生成一个中断

哪个事件可以引起中断是由中断选择(ETSEL[INTSEL])位配置的。事件可以是以下之一:

- 时基计数器等于零(TBCTR = 0x0000)。
- 时间基计数器等于周期(TBCTR = TBPRD)。
- 时基计数器等于零或周期(TBCTR = 0x0000 || TBCTR = TBPRD)
- 定时器递增时，时基计数器等于比较 A 寄存器（CMPA）。
- 定时器递减时，时基计数器等于比较 A 寄存器（CMPA）。
- 定时器递增时，时基计数器等于比较 B 寄存器（CMPB）。
- 定时器递减时，时基计数器等于比较 B 寄存器（CMPB）。
- 定时器递增时，时基计数器等于比较 C 寄存器（CMPC）。
- 定时器递减时，时基计数器等于比较 C 寄存器（CMPC）。

可以从中断事件计数器（ETPS [INTCNT]）寄存器中读取已发生的事件数。也就是说，当发生指定事件时，ETPS [INTCNT]位将递增，直到达到 ETPS [INTPRD]指定的值。当 ETPS [INTCNT] = ETPS [INTPRD]时，计数器停止计数，并设置其输出。仅在将中断发送到 PIE 时清除计数器。

当 ETPS [INTCNT]达到 ETPS [INTPRD]时，将发生以下现象：

- 如果开启中断，即 ETSEL [INTEN] = 1，并且中断标志清零，即 ETFLG [INT] = 0，则产生一个中断脉冲并置位中断标志，即 ETFLG [INT] = 1，并且事件计数器清除 ETPS [INTCNT] =0，计数器将再次开始对事件进行计数。
- 如果禁止中断，即 ETSEL [INTEN] = 0，计数器在达到周期值 ETPS [INTCNT] = ETPS [INTPRD]时，将停止对事件进行计数。

- 如果开启中断，但是中断标志已被设置，则计数器将保持其输出高电平，直到 ENTFLG [INT]标志被清除。这允许在服务一个中断时挂起一个中断。

写入 INTPRD 值时，会发生以下情况：

- 写入 INTPRD 位将自动清除计数器 INTCNT = 0，并且计数器输出将复位（因此不会产生中断）。
- 将 1 写入 ETFRC[INT]位将使事件计数器 INTCNT 递增。当 INTCNT=INTPRD 时，计数器的行为如上所述。
- 当 INTPRD = 0 时，计数器被禁用，因此将不会检测到任何事件，并且 ETFRC [INT]位也将被忽略。

上面的定义意味着您可以在每个事件，第二个事件或第三个事件上生成一个中断。不能每四个或四个以上事件生成一个中断。

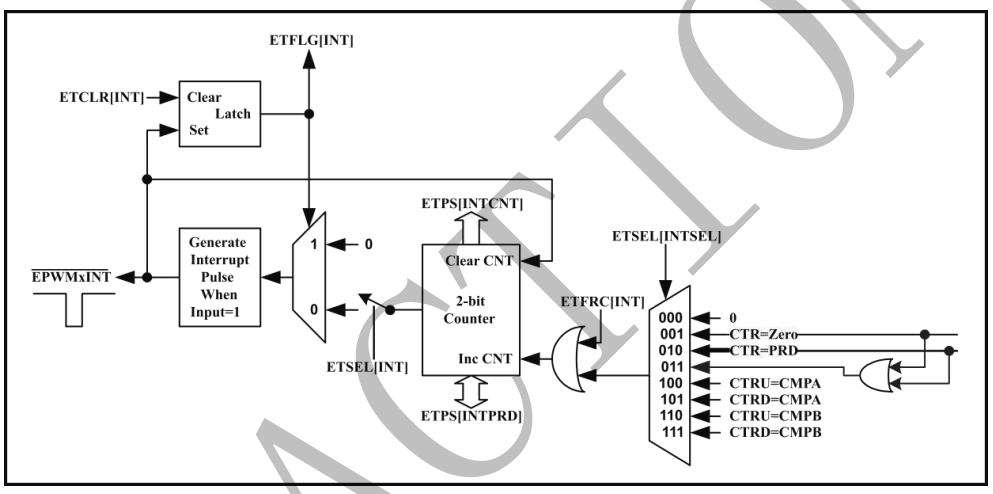


图 21-42　事件触发中断生成

图 21-43 显示了事件触发的转换开始 A（SOCA）脉冲发生器的操作。ETPS[SOCACNT]计数器值和 ETPS [SOCAPRD]周期值的行为与中断发生器类似，不同的是脉冲是连续产生的。即当产生脉冲时，脉冲标记 ETFLG[SOCA]被锁存，但是它不停止进一步的脉冲产生。使能/禁用位 ETSEL [SOCAEN]停止脉冲产生，但是仍然可以对输入事件进行计数，直到达到周期值为止（与中断产生逻辑一样）。可以在 ETSEL [SOCASEL]和 ETSEL [SOCBSEL]位中分别配置将触发 SOCA 和 SOCB 脉冲的事件。可能的事件与可以通过添加来自数字比较（DC）子模块的 DCAEVT1.soc 和 DCBEVT1.soc 事件信号为中断生成逻辑指定的事件相同。

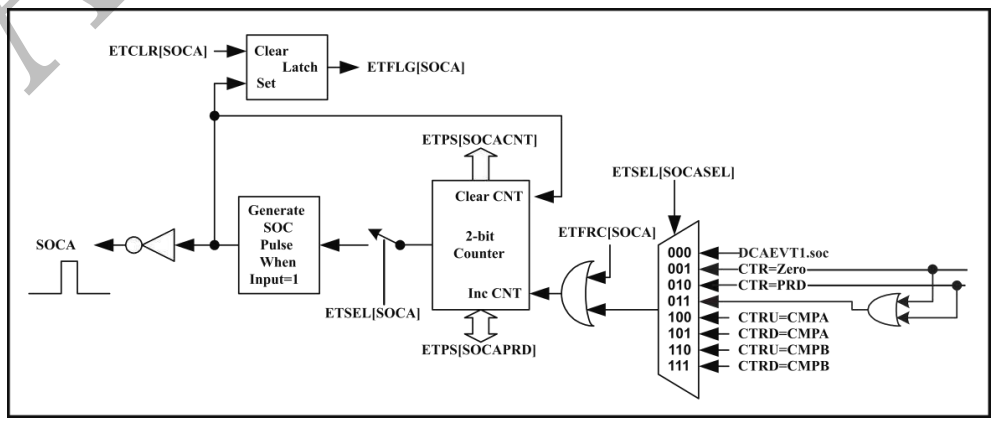


图 21-43　事件触发 SOCA 脉冲生成

图 21-44 显示了事件触发的转换开始-B（SOCB）脉冲发生器的操作。事件触发事件的 SOCB 脉冲发生器的运行方式与 SOCA 相同。

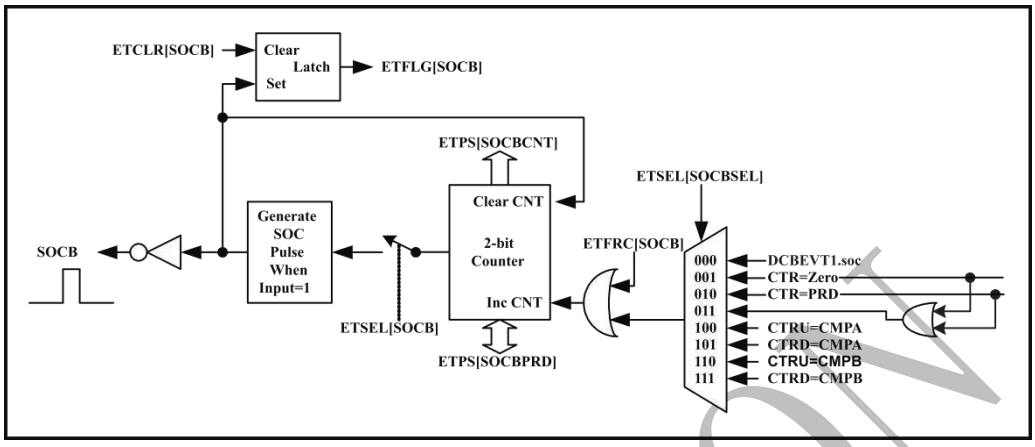


图 21-44　事件触发 SOCB 脉冲生成

21.4.1.9　数字比较（DC）子模块

图 21-45 显示了 EPWM 系统中的数字比较（DC）子模块（请参见阴影块）。

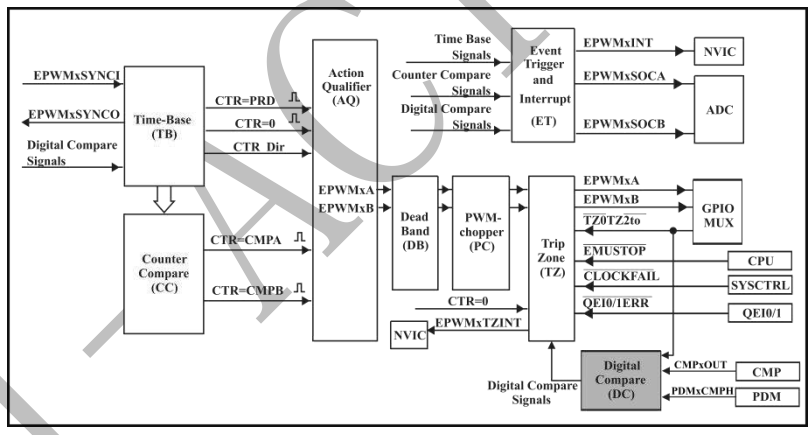


图 21-18　数字比较子模块

图 21-45 说明了数字比较（DC）子模块信号与 EPWM 系统中其他子模块的接口。

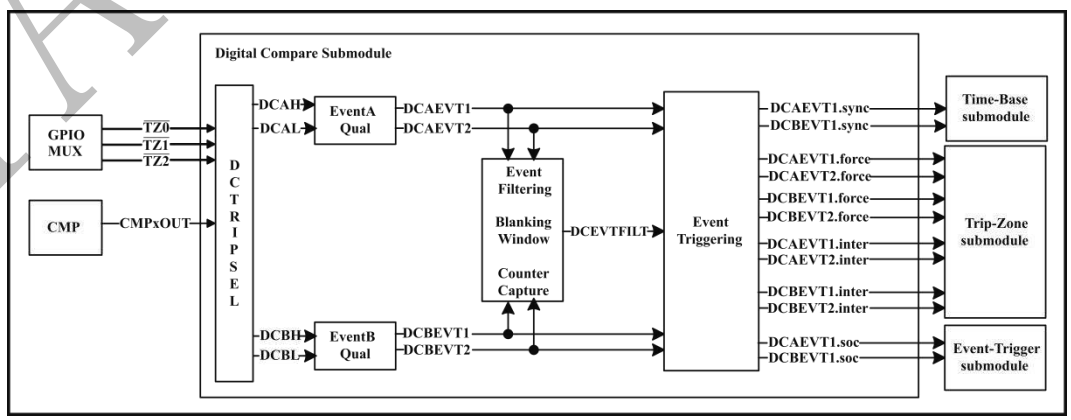


图 21-45　数字比较子模块框图

数字比较（DC）子模块比较 EPWM 模块外部的信号（例如，来自模拟比较器的 CMPxOUT

信号）以直接生成 PWM 事件/动作，然后将其馈送到事件触发，关断区域和时基子模块。此外，还支持消隐窗口功能，以从 DC 事件信号中滤除噪声或不需要的脉冲。

数字比较子模块的目的

数字比较子模块的主要功能是：

- 模拟比较器（CMP）模块的输出以及 $\overline{TZ0}$，$\overline{TZ1}$ 和 $\overline{TZ2}$ 输入生成数字比较 A 高/低（DCAH，DCAL）和数字比较 B 高/低（DCBH，DCBL）信号。
- DCAH / L 和 DCBH / L 信号触发事件，然后可以对其进行过滤或将其直接馈送到关断区域，事件触发和时基子模块：
  - 产生关断区域中断
  - 生成 ADC 转换开始
  - 强制事件
  - 生成用于同步 EPWM 模块 TBCTR 的同步事件。

  可以选择事件过滤（消隐窗口逻辑）使输入信号消隐以消除噪声。

数字比较子模块的应用

数字比较子模块的操作通过以下寄存器进行控制：

表 21-19 数字比较子模块寄存器

Register	Address offset	Shadowed	Description
TZDCSEL	0x0078	NO	Trip Zone Digital Compare Select Register
DCTRIPSEL	0x0080	NO	Digital Compare Trip Select Register
DCACTL	0x0084	NO	Digital Compare A Control Register
DCBCTL	0x0088	NO	Digital Compare B Control Register
DCFCTL	0x008C	NO	Digital Compare Filter Control Register
DCCAPCTL	0x0090	NO	Digital Compare Capture Control Register
DCCAP	0x0094	Yes	Digital Compare Counter Capture Register
DCFOFFSET	0x0098	Writes	Digital Compare Filter Offset Register
DCFOFFSETCNT	0x009C	NO	Digital Compare Filter Offset Counter Register
DCFWINDOW	0x00A0	NO	Digital Compare Filter Window Register
DCFWINDOWCNT	0x00A4	NO	Digital Compare Filter Window Counter Register

数字比较子模块的详细功能

以下各节描述了数字比较子模块的详细功能和配置选项。

数字比较事件

可以通过 DCTRIPSEL 位选择来自模拟比较器（CMP）模块的关断区域输入（$\overline{TZ0}$，$\overline{TZ1}$ 和 $\overline{TZ2}$）和 CMPxOUT 信号，以生成数字比较 A 高电平和低电平（DCAH/L）和数字比较 B 高电平和低电平（DCBH/L）信号。然后，TZDCSEL 寄存器的配置将限定对所选 DCAH/L 和 DCBH/L 信号的操作，这些信号会生成 DCAEVT1/2 和 DCBEVT1/2 事件（事件鉴定 A 和 B）。

注意：$\overline{TZn}$ 信号在用作 DCEVT 触发功能时，被视为正常输入信号，可以定义为高电平有效或低电平有效输入。当 $\overline{TZn}$，DCAEVTx.force 或 DCBEVTx.force 信号处于活动状态时，EPWM 输出异步触发。为了保持锁存状态，至少需要 3 * TBCLK 同步脉冲宽度。如果脉冲宽度小于 3*TBCLK 同步脉冲宽度，则关断条件可能会或可能不会被 CBC 或 OST 锁存器锁存。

然后可以对 DCAEVT1/2 和 DCBEVT1/2 事件进行过滤，以提供事件信号的过滤版本（DCEVTFILT），或者可以绕过过滤。过滤将在后续章节中进一步讨论。DCAEVT1/2 和 DCBEVT1/2 事件信号或经过滤波的 DCEVTFILT 事件信号都可以对关断区域子模块，TZ 中断，ADC SOC 或 PWM 同步信号产生强制信号。

- force 信号：
  DCAEVT1 / 2.force 信号强制关断区域条件，该条件直接影响 EPWMxA 引脚上的输出（通过 TZCTL [DCAEVT1 或 DCAEVT2]配置），或者如果 DCAEVT1/2 信号被选择为单次触发或逐周期触发源（通过 TZSEL 寄存器），DCAEVT1/2.force 信号可以通过 TZCTL[TZA]配置影响关断动作。DCBEVT1/2.force 信号的行为类似，但会影响 EPWMxB 输出引脚而不是 EPWMxA 输出引脚。TZCTL 寄存器上发生冲突的操作的优先级如下（最高优先级覆盖较低优先级）：
  Output EPWMxA: TZA (highest) -> DCAEVT1 -> DCAEVT2 (lowest)
  Output EPWMxB: TZB (highest) -> DCBEVT1 -> DCBEVT2 (lowest)
- 中断信号：
  DCAEVT1/2.interrupt 信号生成对 NIC 的关断区域中断。要使能中断，用户必须将 TZEINT 寄存器中的 DCAEVT1，DCAEVT2，DCBEVT1 或 DCBEVT2 位置 1。一旦发生这些事件之一，就会触发 EPWMxTZINT 中断，并且必须将 TZCLR 寄存器中的相应位置 1，以清除该中断。
- soc 信号：
  DCAEVT1.soc 信号与事件触发子模块连接，可以选择该事件为通过 ETSEL [SOCASEL]位生成 ADC 转换开始 A（SOCA）脉冲的事件。同样，可以选择 DCBEVT1.soc 信号作为事件，该事件通过 ETSEL [SOCBSEL]位生成 ADC 转换开始 B（SOCB）脉冲。
- sync 信号：
  DCAEVT1.sync 和 DCBEVT1.sync 事件与 EPWMxSYNCI 输入信号和 TBCTL [SWFSYNC]信号进行“或”运算以生成到时基计数器的同步脉冲。

  下图显示了如何处理 DCAEVT1，DCAEVT2 或 DCEVTFLT 信号以生成数字比较 A 事件强制，中断，SOC 和同步信号。

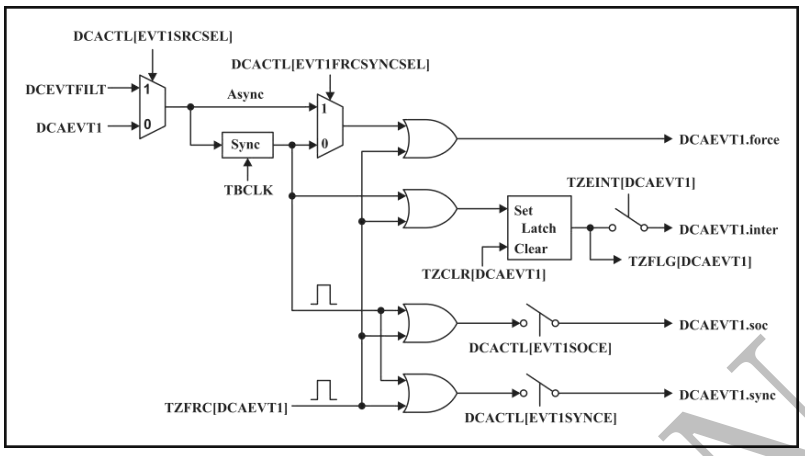


图 21-46　DCAEVT1 事件产生

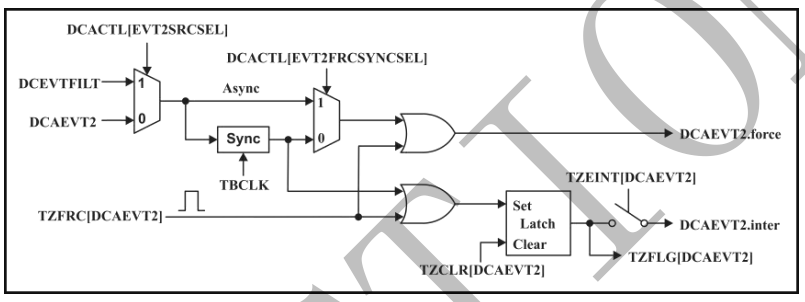


图 21-47　DCAEVT2 事件产生

下图显示了如何处理 DCBEVT1，DCBEVT2 或 DCEVTFLT 信号以生成数字比较 B 事件强制，中断，SOC 和同步信号 。

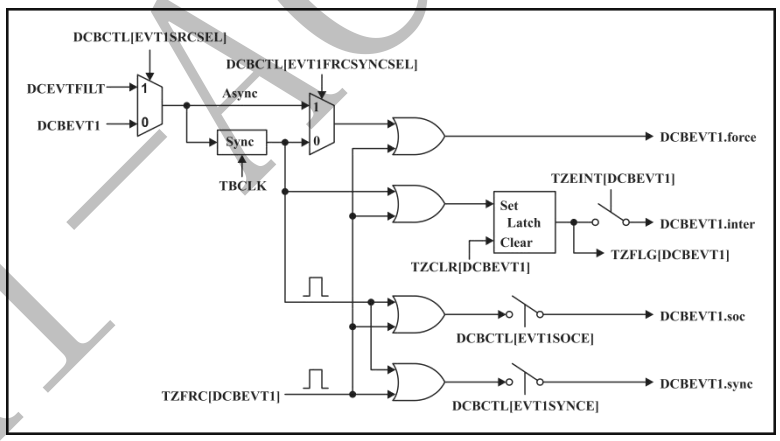


图 21-48　DCBEVT1 事件产生

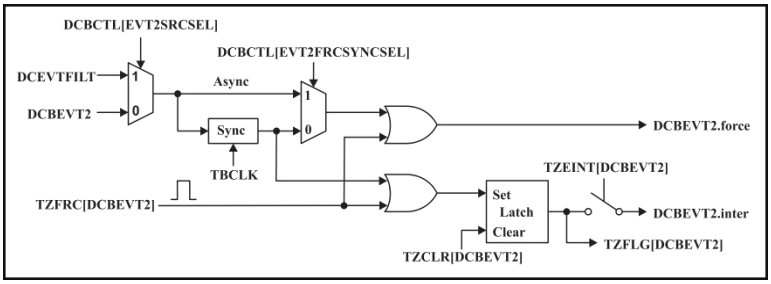


图 21-49　DCBEVT2 事件产生

事件过滤

可以通过事件过滤逻辑对 DCAEVT1/2 和 DCBEVT1/2 事件进行过滤，以通过在特定时间段内消隐事件来消除噪声。这在以下情况下非常有用：可以选择模拟比较器输出来触发 DCAEVT1/2 和 DCBEVT1/2 事件，并且消隐逻辑用于触发 PWM 输出或生成中断或 ADC 转换开始之前滤除信号的潜在噪声。事件过滤还可以捕获触发事件的 TBCTR 值。下图显示了事件过滤逻辑的详细信息。

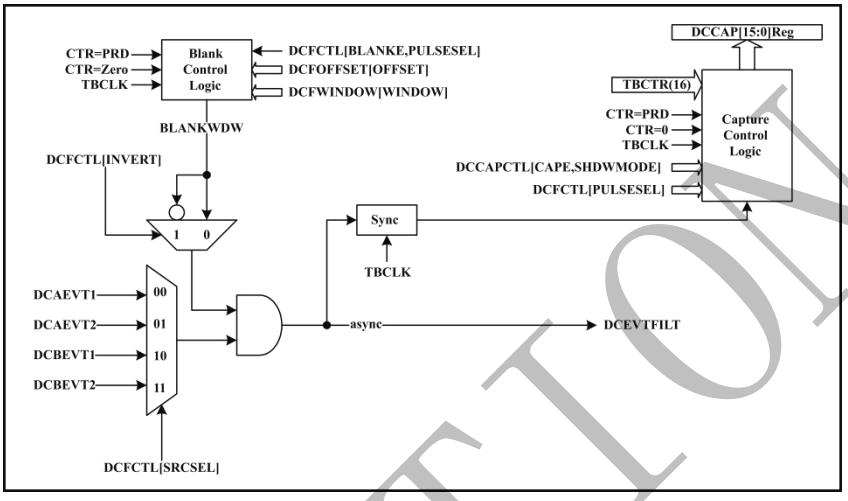


图 21-50　事件过滤

如果启用了消隐逻辑，则选择数字比较事件之一 DCAEVT1，DCAEVT2，DCBEVT1，DCBEVT2 进行滤波。消隐窗口将滤除信号处于活动状态时发生的所有事件，它将与 CTR=PRD 脉冲或 CTR=0 脉冲（由 DCFCTL[PULSESEL]位配置）对齐。TBCLK 计数中的偏移值被编程到 DCFOFFSET 寄存器中，它确定在 CTR=PRD 或 CTR=0 脉冲之后的哪一点消隐窗口开始。应用程序将空白窗口的持续时间（以偏移计数器到期后的 TBCLK 计数为单位）写入 DCFWINDOW 寄存器。在消隐窗口期间，所有事件都将被忽略。在消隐窗口结束之前和之后，事件可以像以前一样生成 soc，sync，interrupt 和 force 信号。

图 21-51 给出了 EPWM 周期内失调和消隐窗口的几种时序条件。请注意，如果消隐窗口越过 CTR= 0 或 CT=PRD 边界，下一个窗口在 CTR=0 或 CTR=PRD 脉冲后仍然以相同的偏移值开始。

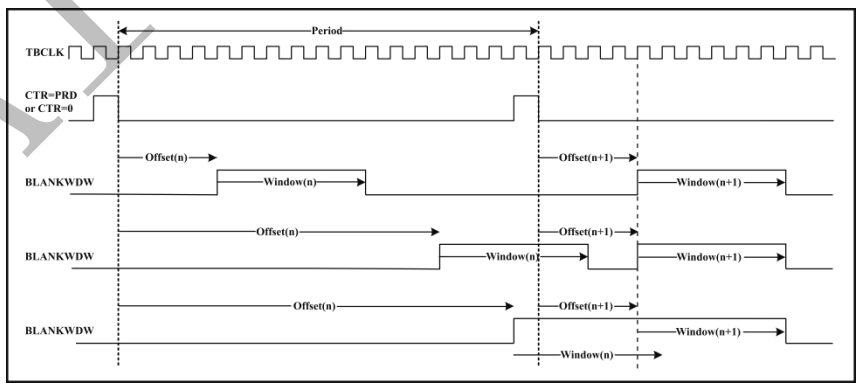


图 21-51　消隐窗口时序图

21.4.2 电源拓扑的应用

单个 EPWM 模块具有必需的定时和控制资源，可以作为独立模块运行，也可以与其他相同的 EPWM 模块同步运行。

21.4.2.1 多个模块概述

这个简化的 EPWM 模块图显示了如何通过多个 EPWM 模块协同工作来控制多开关电源拓扑所需的关键资源。

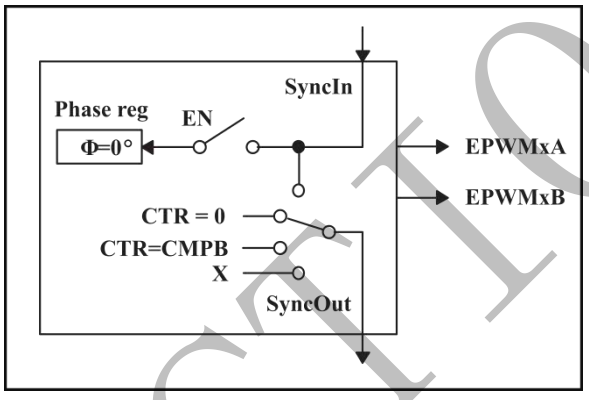


图 21-52 简化的 EPWM 模块图

21.4.2.2 关键配置功能

各模块的关键配置选择如下:

- SyncIn 的选项
- 在进入的同步选通上将自己的计数器与相位寄存器一起加载：启用（EN）开关已关闭
- 不执行任何操作或忽略传入的同步选通—启用开关
- 同步通过：SyncOut 连接到 SyncIn
- 主模式，在 PWM 边界提供同步：SyncOut 连接到 CTR = PRD
- 主模式，可在任何可编程时间点提供同步：SyncOut 连接到 CTR = CMPB
- 模块处于独立模式，不提供与其他模块的同步-SyncOut 连接到 X（已禁用）
- SyncOut 的选项
  - 同步通过：SyncOut 连接到 SyncIn
  - 主模式，在 PWM 边界提供同步：SyncOut 连接到 CTR = PRD
  - 主模式，可在任何可编程时间点提供同步：SyncOut 连接到 CTR = CMPB
  - 模块处于独立模式，不提供与其他模块的同步：SyncOut 连接到 X（禁用）

对于 SyncOut 的每种选择，模块还可以选择在 SyncIn 选通输入上向其自己的计数器加载新的相位值，或者选择忽略它，即通过使能开关。尽管可以进行各种组合，但最常用的两种模式（主模块和从模块模式）如图 21-53 所示。

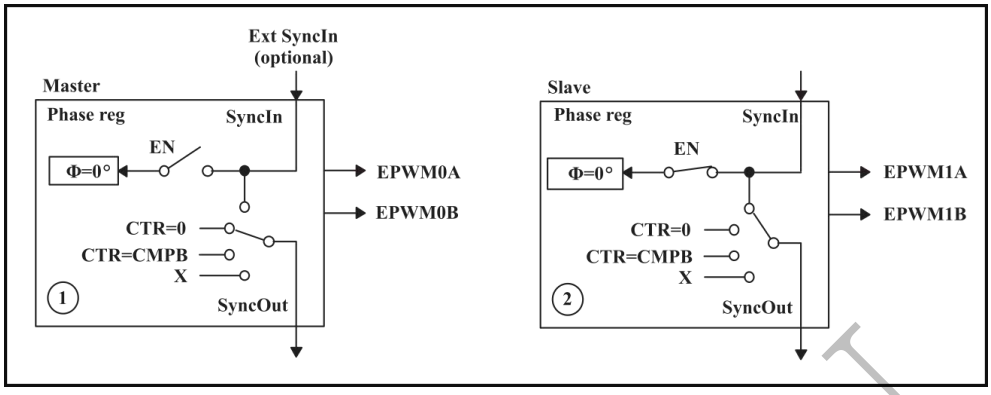


图 21-53　EPWM0 配置为主机，EPWM1 配置为从机

21.4.2.3　用独立的频率控制多个 Buck 转换器

Buck 是最简单的电源转换拓扑之一。配置为主机的单个 EPWM 模块可以以相同的 PWM 频率控制两个 Buck 级。如果每个 Buck 转换器需要独立的频率控制，则必须为每个转换器级分配一个 EPWM 模块。图 21-54 显示了四个 Buck 级，每个 Buck 级在独立的频率下运行。在这种情况下，所有四个 EPWM 模块都被配置为主模块，并且不使用同步。图 21-55 显示了由图 21-54 所示的设置生成的波形。请注意，尽管有四个阶段，但仅显示了三个波形。

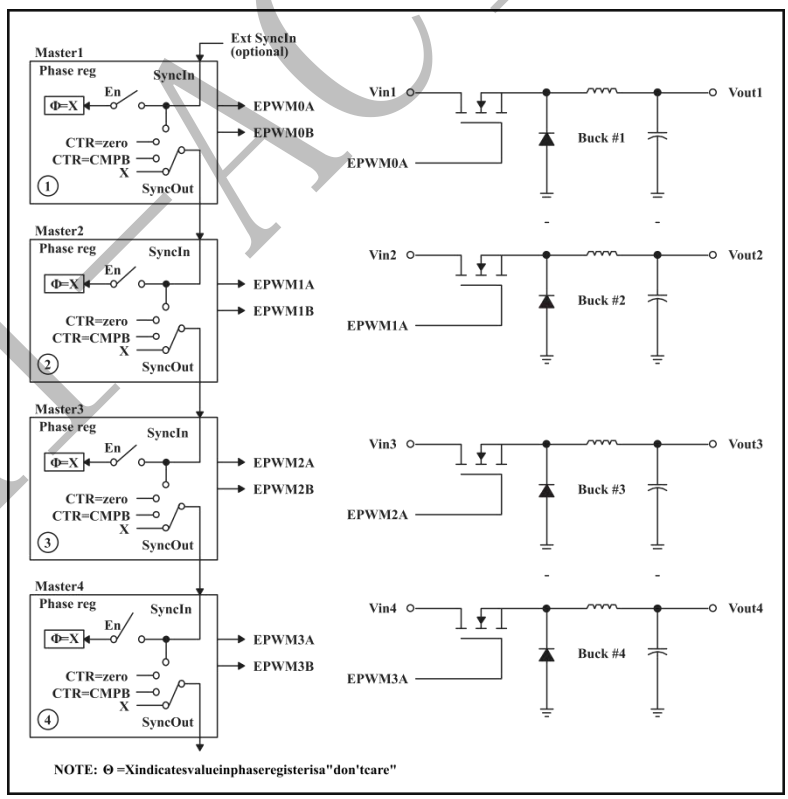


图 21-54　四个 Buck 级控制图（注：此处 Fpwm1 != Fpwm2 != Fpwm3 != Fpwm4）

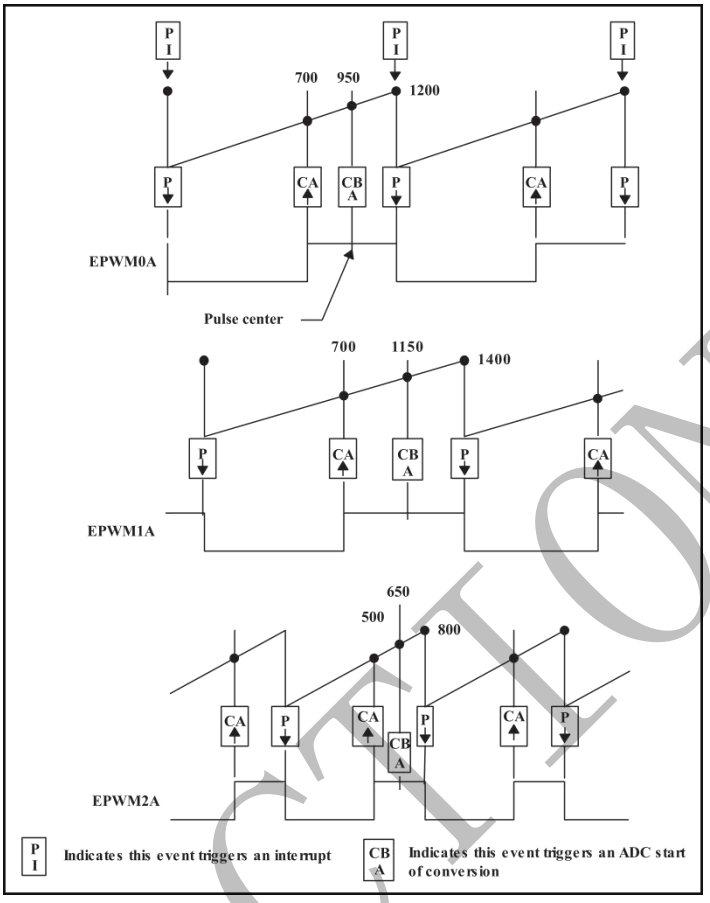


图 21-55　Buck 波形图（注：此处只显示三个 Buck）

21.4.2.4　使用相同频率控制多个 Buck 转换器

如果需要同步，则 EPWM 模块 2 可以配置为从机，并且能以模块 1 的整数倍（N）频率运行。从主机到从机的同步信号可确保这些模块保持锁定状态。图 21-56 显示了这种配置。图 21-57 显示了该配置生成的波形。

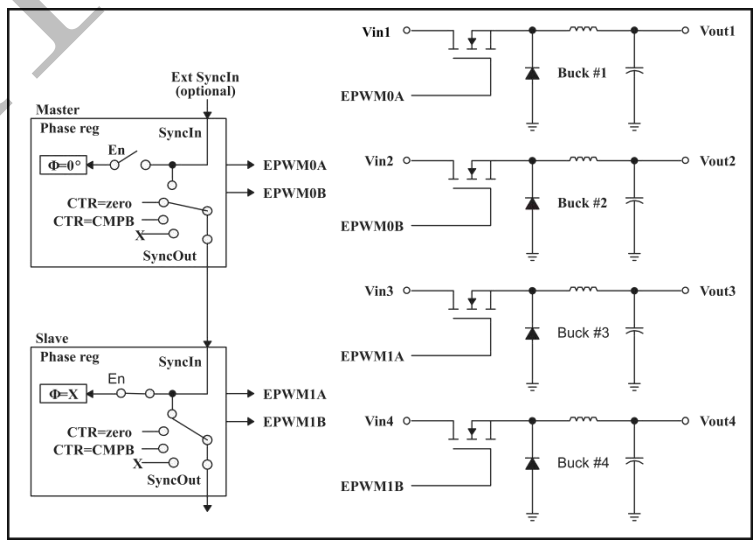


图 21-56　四个 Buck 级控制图

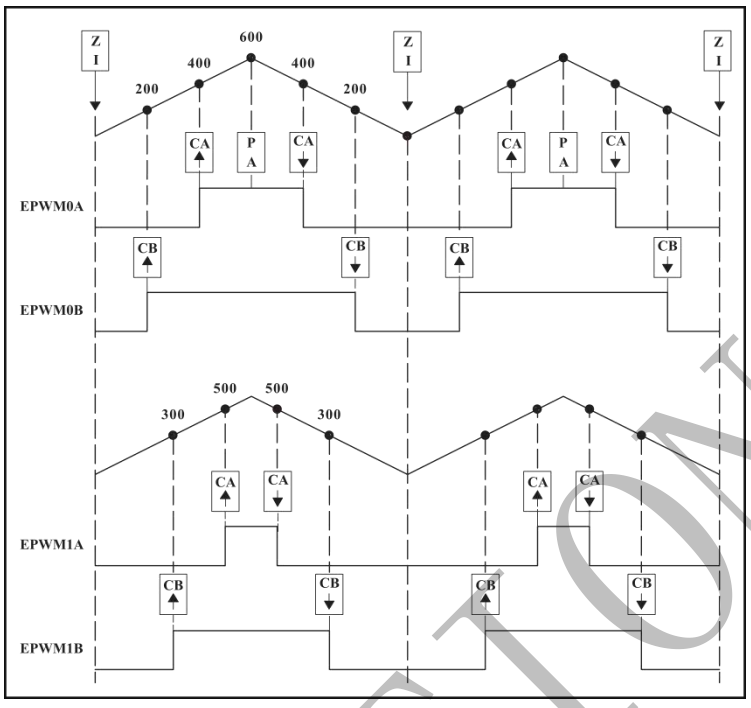


图 21-57　Buck 波形图（注：此处 Fpwm2 = Fpwm1）

21.4.2.5　控制多个 Half-H 桥(HHB)转换器

这些相同的 EPWM 模块也可以解决需要控制多个开关元件的拓扑。可以通过单个 EPWM 模块控制 Half-H 桥级。该控制可以扩展到多级。图 21-58 显示了对两个同步的 Half-H 桥级的控制，其中，级 2 可以在级 1 的整数倍（N）频率下工作。图 21-59 显示了由图 21-58 所示的配置生成的波形。

将从机 2 配置为同步直通；如果需要，此配置允许第三个 Half-H 桥由 PWM 模块 3 控制，并且最重要的是，保持与主机 1 同步。

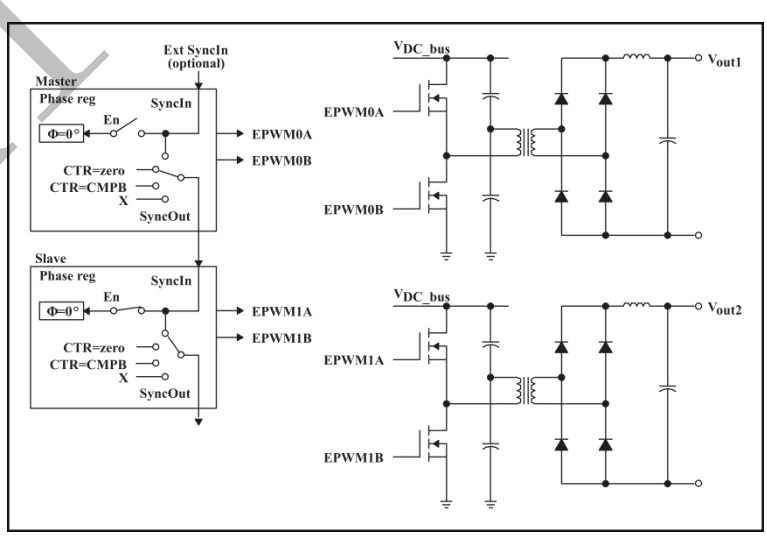


图 21-58　两个 Half-H 桥控制图

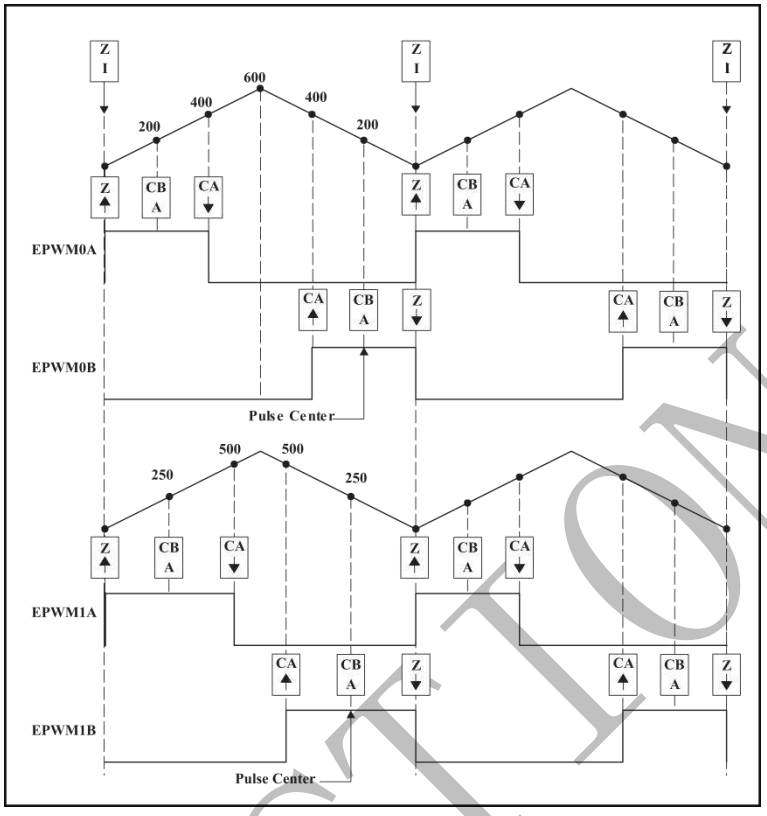


图 21-59　Half-H 桥波形图（注：此处 Fpwm2 = Fpwm1）

21.4.2.6　控制电机(ACI 和 PMSM)的双三相逆变器

多个模块控制单个功率级的思路可以扩展到三相逆变器应用中。在这种情况下，可以使用三个 PWM 模块控制六个开关元件，每个逆变器支路一个 PWM 模块控制。每个支路必须以相同的频率切换，并且所有支路必须同步。一个主机+两个从机配置可以轻松满足此要求。图 21-60 显示了六个 PWM 模块如何控制两个独立的三相逆变器。每个都运行一个电机。

如前几节所示，我们可以选择以不同的频率运行每个逆变器，（模块 1 和模块 4 为主机，如图 21-61 所示），也可以使用一个主机（模块 1）与五个从机进行同步。在这种情况下，模块 4、5 和 6（均相等）的频率可以是模块 1、2、3（均相等）的频率的整数倍。

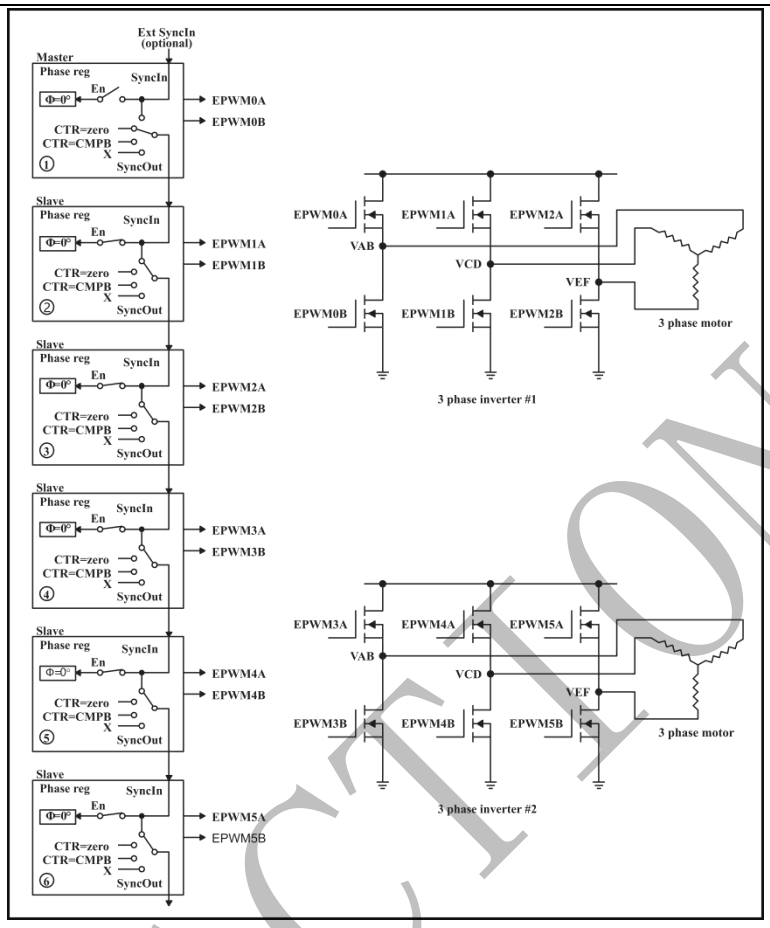


图 21-60　两个三相逆变器级控制图，通常用于电机控制

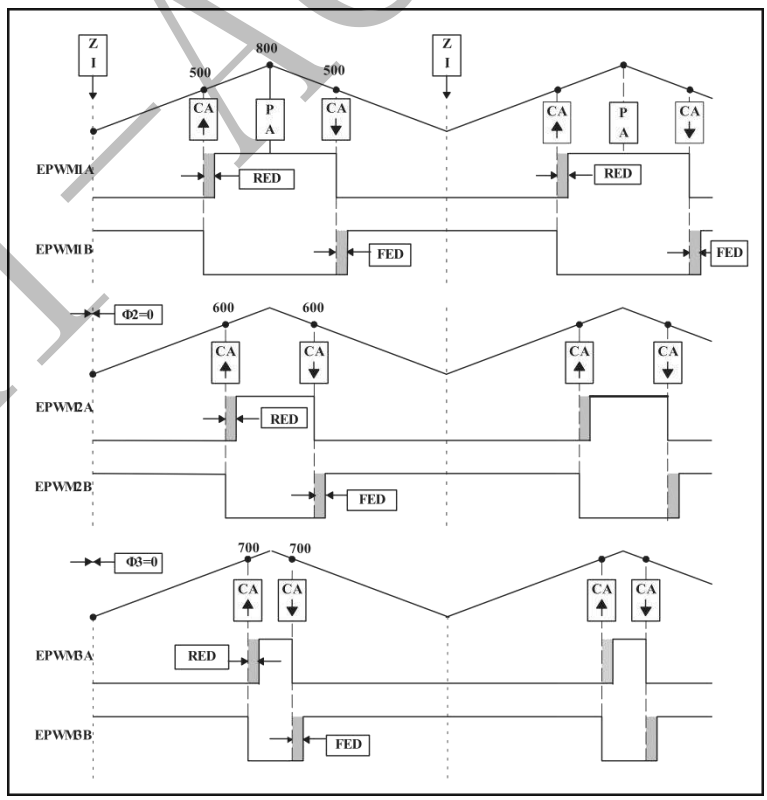


图 21-61 三相逆变器波形图，只显示一个逆变器

21.4.2.7 在 PWM 模块之间使用相位控制的实际应用

到目前为止，没有一个示例使用相位寄存器（TBPHS）。它要么被设置为零，要么其值无关紧要。但是，通过将适当的值编程到 TBPHS 中，多个 PWM 模块可以解决另一类电源拓扑，这些拓扑依赖于支路（或级）之间的相位关系来进行正确操作。如 TB 模块部分所述，PWM 模块可以配置为允许同步脉冲使 TBPHS 寄存器加载到 TBCTR 寄存器中。为了说明这个概念，图 21-62 显示了一个主模块和从模块，它们具有 120°的相位关系，即从模块先于主模块。

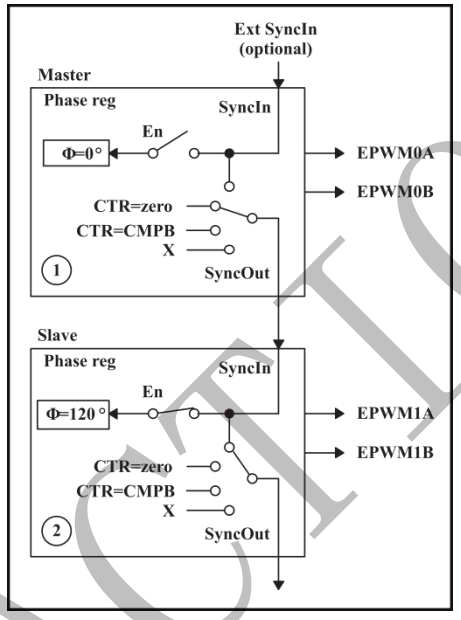


图 21-62　PWM 模块间相位控制配置图

图 21-63 显示了该配置的相关时序波形。在此，主机和从机的 TBPRD = 600。对于从机，TBPHS = 200（200/600 X 360°= 120°）。当主机产生一个同步脉冲（CTR = PRD）时，TBBPS = 200 的值就会被加载到从机 TBCTR 寄存器中，因此从机时基始终领先于主机的时基 120°。

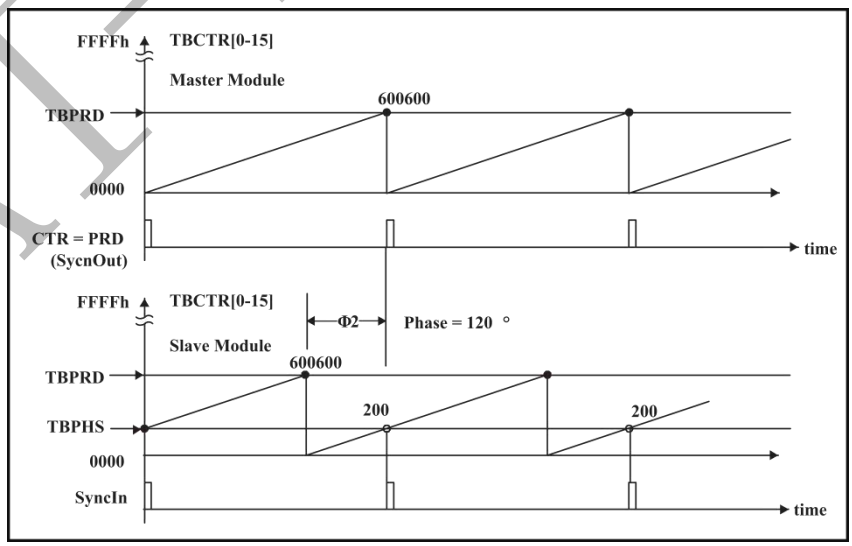


图 21-63　模块间相位控制时序波形图

21.4.2.8 控制三相交错式 DC / DC 转换器

图 21-64 显示了一种流行的电源拓扑，它利用了模块之间的相位偏移。该系统使用三个 PWM 模块，其中模块 1 配置为主机。为了工作，相邻模块之间的相位关系必须为 F = 120°。这是通过将从 TBPHS 寄存器 2 和 3 分别设置为周期值的 1/3 和 2/3 来实现的。例如，如果将 600 个计数的值加载到周期寄存器中，则 TBPHS（从机 2）= 200，而 TBPHS（从机 3）=400。两个从机都与主机 1 同步。

通过适当设置 TBPHS 值，可以将此概念扩展到四个或更多相。下式给出了 N 相的 TBPHS 值：

TBPHS（N,M）=（TBPRD / N）×（-1）

其中：

N =相数

M = PWM 模块编号

例如，对于三相情况（N = 3），TBPRD = 600，

TBPHS（3,2）=（600/3）x（2-1）= 200（即从机模块 2 的相位值）

TBPHS（3,3）= 400（从机模块 3 的相位值）

图 21-64 配置波形如图 21-65 所示。

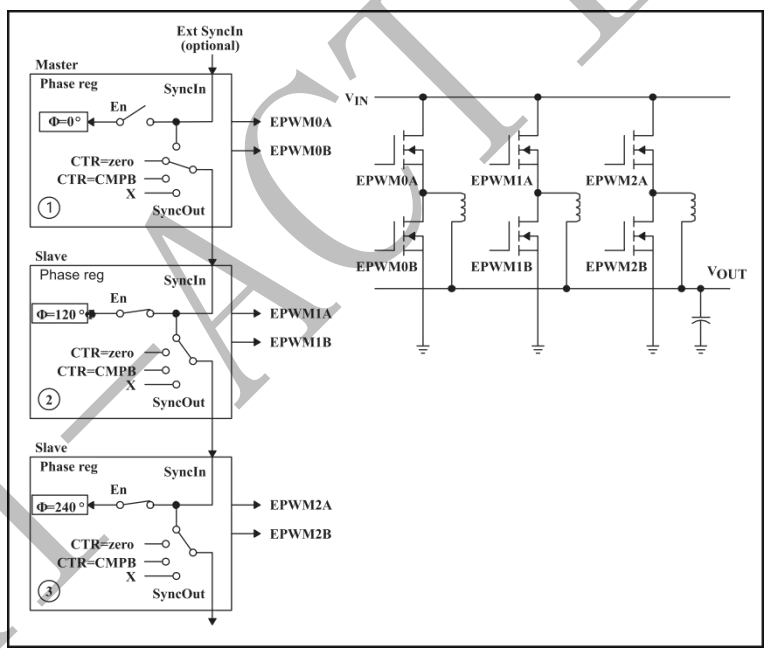


图 21-64　三相交错 DC/DC 转换器控制图

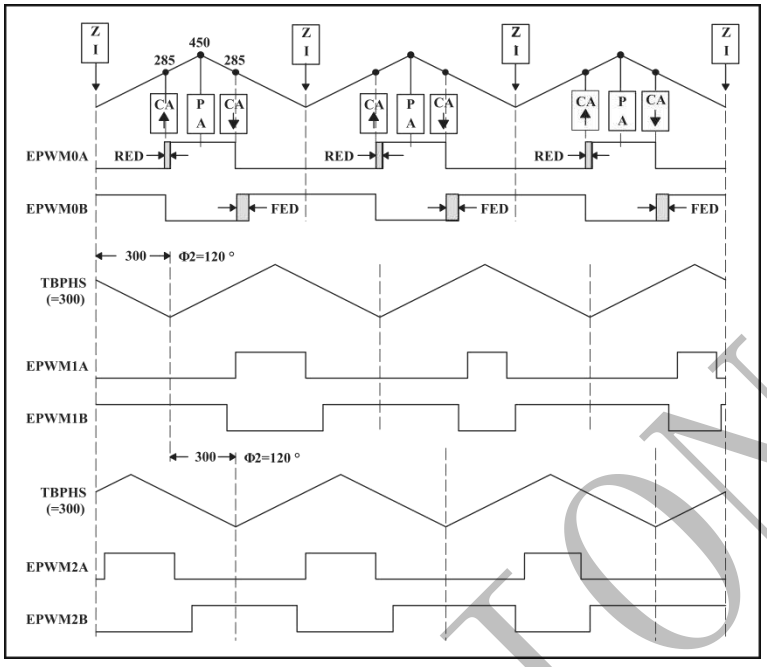


图 21-65　三相交错 DC/DC 转换器波形图

21.4.2.9　控制零电压开关全桥（ZVSFB）转换器

如图 21-66 所示，假设各支路(模块)之间存在静态或恒定的相位关系。在这种情况下，通过调制占空比来实现控制。也可以逐周期地动态改变相位值。这有助于控制一类被称为移相全桥或零电压开关全桥的电力拓扑结构。这里的控制参数不是占空比（占空比保持在大约 50% 不变）；相反，它是支路之间的相位关系。这样的系统可以通过分配两个 PWM 模块的资源来控制单个功率级来实现，而这个功率级又需要控制四个开关元件。主从模块同步控制 Full-H 桥如图 21-67 所示。在这种情况下，主从模块都需要以相同的 PWM 频率进行切换。相位是由使用从相位寄存器(TBPHS)控制的。主寄存器不使用相位寄存器，因此可以初始化为零。

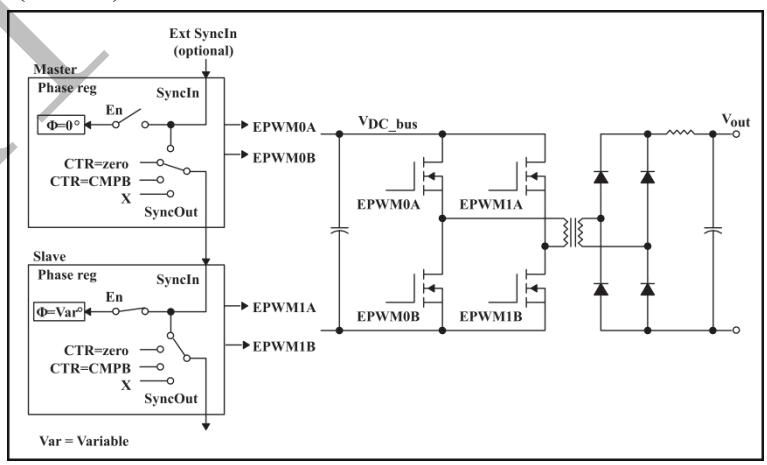


图 21-66　Full-H 桥控制图

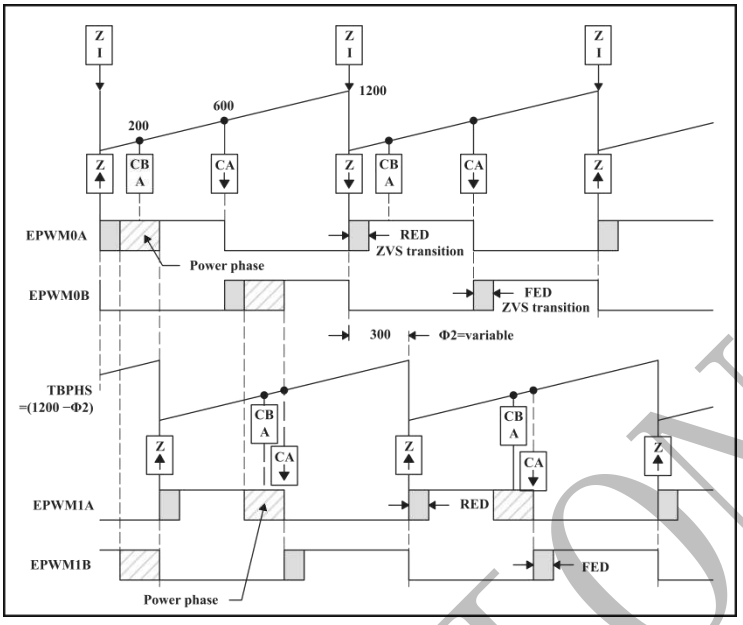


图 21-67　ZVS Full-H 桥波形图

21.4.2.10　控制峰值电流模式控制 Buck 模块

峰值电流控制技术具有许多优点，例如自动过流限制，快速校正输入电压变化以及降低磁饱和度。图 21-68 显示了 EPWM1A 以及片上模拟比较器在降压转换器拓扑结构中的使用。输出电流通过一个电流检测电阻器检测，并馈送到片上比较器的正端。内部可编程 10 位 DAC 可用于在比较器的负端提供参考峰值电流。或者，可以在此输入端连接一个外部基准。比较器输出是数字比较子模块的输入。EPWM 模块的配置方式是，一旦感测到的电流达到峰值参考值，就会使 EPWM1A 输出触发。使用了逐周期触发机制。图 21-69 显示了该配置生成的波形。

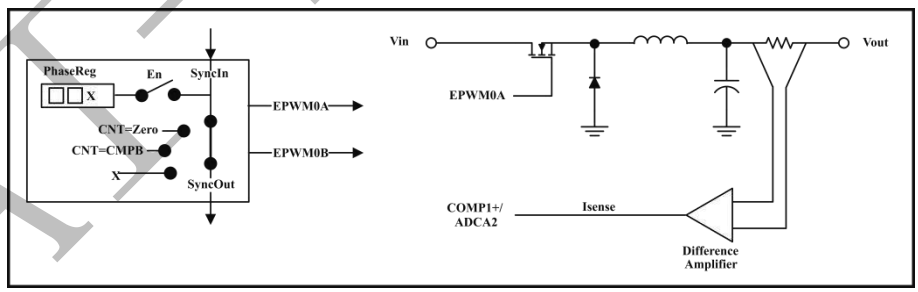


图 21-68　Buck 转换器峰值电流模式控制图

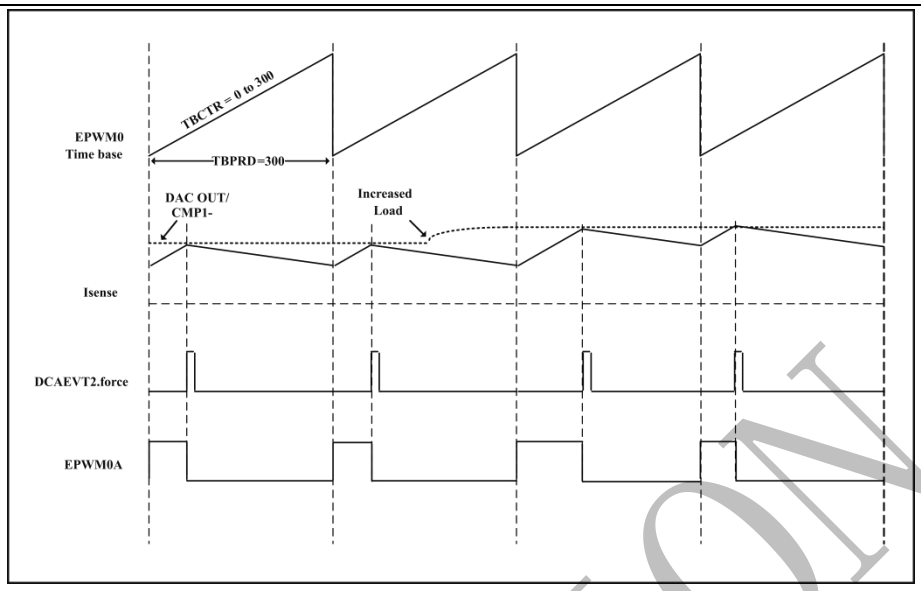


图 21-69　峰值电流模式控制波形图

21.4.2.11　控制 H 桥 LLC 谐振转换器

多年来，在电力电子领域中，谐振变换器的各种拓扑结构已经广为人知。除了这些以外，H 桥 LLC 谐振转换器拓扑最近在许多需要高效率和功率密度的消费电子应用中得到普及。在此示例中，详细介绍了 EPWM1 的单通道配置，但该配置可以轻松扩展到多通道。这里，受控参数不是占空比（保持恒定在大约 50%）；它是频率。尽管死区没有被控制并将保持恒定为 300ns（即 30 @ 100MHz TBCLK），但用户可以实时更新死区，以通过调整足够的软切换时间延迟来提高效率。

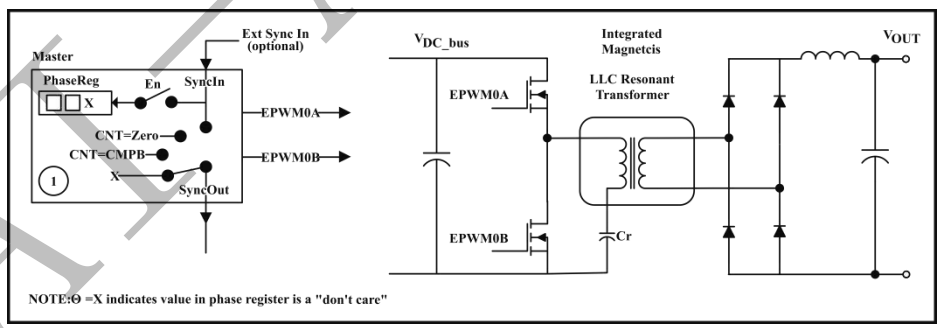


图 21-70　两个谐振转换器级控制图

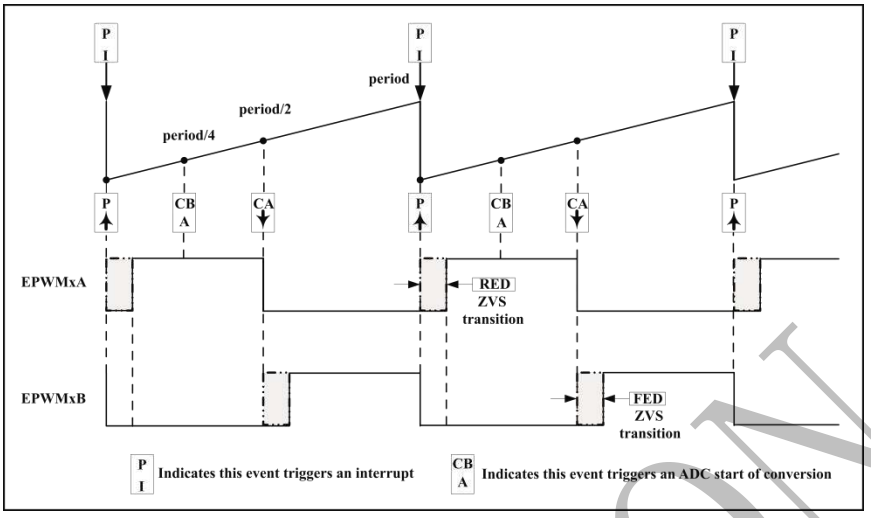


图 21-71　H 桥 LLC 谐振转换器 PWM 波形图

21.5 寄存器描述

21.5.1 寄存器列表

EPWM0 基地址：**0x4003 A000**
EPWM1 基地址：**0x4003 A100**
EPWM2 基地址：**0x4003 A200**
EPWM3 基地址：**0x4003 A300**
EPWM4 基地址：**0x4003 A400**
EPWM5 基地址：**0x4003 A500**
EPWM6 基地址：**0x4003 A600**

偏移	实例地址	名称	默认值	描述
0x00	0x4003A000	EPWM_TBPRD	0x00000000	EPWM 时基周期寄存器
0x04	0x4003A004	EPWM_TBPHS	0x00000000	EPWM 时基相位寄存器
0x08	0x4003A008	EPWM_TBCTR	0x00000000	EPWM 时基计数寄存器
0x0C	0x4003A00C	EPWM_TBCTL	0x00000003	EPWM 时基控制寄存器
0x10	0x4003A010	EPWM_TBSTS	0x00000001	EPWM 时基状态寄存器
0x20	0x4003A020	EPWM_CMPA	0x00000000	EPWM 计数比较 A 寄存器
0x24	0x4003A024	EPWM_CMPB	0x00000000	EPWM 计数比较 B 寄存器
0x28	0x4003A028	EPWM_CMPC	0x00000000	EPWM 计数比较 C 寄存器
0x2C	0x4003A02C	EPWM_CMPCTL	0x00000000	EPWM 计数比较控制寄存器
0x30	0x4003A030	EPWM_AQCTLA	0x00000000	EPWM 动作限定输出 A 控制寄存器
0x34	0x4003A034	EPWM_AQCTLB	0x00000000	EPWM 动作限定输出 B 控制寄存器
0x38	0x4003A038	EPWM_AQSFRC	0x00000000	EPWM 动作限定软件强制寄存器
0x3C	0x4003A03C	EPWM_AQCSFRC	0x00000000	EPWM 动作限定连续软件强制寄存器
0x40	0x4003A040	EPWM_DBCTL	0x00000000	EPWM 死区控制寄存器
0x44	0x4003A044	EPWM_DBRED	0x00000000	EPWM 死区上升沿延迟寄存器
0x48	0x4003A048	EPWM_DBFED	0x00000000	EPWM 死区下降沿延迟寄存器
0x50	0x4003A050	EPWM_PCCTL	0x00000000	EPWM PWM 斩波寄存器
0x60	0x4003A060	EPWM_TZSEL	0x00000000	EPWM 关断区域选择寄存器
0x64	0x4003A064	EPWM_TZCTL	0x00000000	EPWM 关断区域控制寄存器
0x68	0x4003A068	EPWM_TZEINT	0x00000000	EPWM 关断区域使能中断寄存器
0x6C	0x4003A06C	EPWM_TZFLAG	0x00000000	EPWM 关断区域标志寄存器
0x70	0x4003A070	EPWM_TZCLR	0x00000000	EPWM 关断区域清除寄存器
0x74	0x4003A074	EPWM_TZFRC	0x00000000	EPWM 关断区域强制寄存器
0x78	0x4003A078	EPWM_TZDCSEL	0x00000000	EPWM 关断区域数字比较事件选择寄存器
0x80	0x4003A080	EPWM_DCTRIPSEL	0x00000000	EPWM 数字比较关断选择寄存器
0x84	0x4003A084	EPWM_DCACTL	0x00000000	EPWM 数字比较 A 控制寄存器
0x88	0x4003A088	EPWM_DCBCTL	0x00000000	EPWM 数字比较 B 控制寄存器
0x8C	0x4003A08C	EPWM_DCFCTL	0x00000000	EPWM 数字比较滤波控制寄存器
0x90	0x4003A090	EPWM_DCCAPCTL	0x00000000	EPWM 数字比较捕获控制寄存器
0x94	0x4003A094	EPWM_DCCAP	0x00000000	EPWM 数字比较捕获寄存器
0x98	0x4003A098	EPWM_DCFOFFSET	0x00000000	EPWM 数字比较滤波偏移寄存器
0x9C	0x4003A09C	EPWM_DCFOFFSETCNT	0x00000000	EPWM 数字比较滤波偏移计数寄存器

0xA0	0x4003A0A0	EPWM_DCFWINDOW	0x00000000	EPWM 数字比较滤波窗口寄存器
0xA4	0x4003A0A4	EPWM_DCFWINDOWCNT	0x00000000	EPWM 数字比较滤波窗口计数寄存器
0xB0	0x4003A0B0	EPWM_ETSEL	0x00000000	EPWM 事件触发寄存器
0xB4	0x4003A0B4	EPWM_ETPS	0x00000000	EPWM 事件触发预分频寄存器
0xB8	0x4003A0B8	EPWM_ETFLAG	0x00000000	EPWM 事件触发标志寄存器
0xBC	0x4003A0BC	EPWM_ETCLR	0x00000000	EPWM 事件触发清除寄存器
0xC0	0x4003A0C0	EPWM_ETFRC	0x00000000	EPWM 事件触发强制寄存器

21.5.2 寄存器详细描述

21.5.2.1 EPWM 时基周期寄存器（EPWM_TBPRD）

- 名称：EPWM Time Base Period Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		TBPRD
访问		R/W
复位值		0x0

字段	说明
[15:0] TBPRD	**时基计数器周期控制位（EPWM Time Base Period）** 该位可控制 PWM 的频率大小。默认使用 TBPRD 影子寄存器。 当 PRDLD（TBCTL[0]）为 0 时，使能影子寄存器，对 TBPRD 位的任何读写操作都会作用在影子寄存器上 当 PRDLD（TBCTL[0]）为 1 时，任何读写操作都是直接作用在 TBPRD 本体（激活寄存器）上 注意：影子寄存器和激活寄存器共享相同的内存地址，也就是说对影子寄存器的操作就是对激活寄存器的操作，但两者对于读写操作后的结果生效方式不同。即当使用者在已产生 PWM 波的期间改写了 TBPRD，若此时影子寄存器使能，则当时基计数器等于 0 时输出的 PWM 波才会发生变化；若此时不使能影子寄存器，则 PWM 波的输出立即发生变化。

21.5.2.2 EPWM 时基相位寄存器（EPWM_TBPHS）

- 名称：EPWM Time Base Phase Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		TBPHS
访问		R/W
复位值		0x0

字段	说明
[15:0] TBPHS	**相对于提供同步输入信号的时基，所选的 EPWM 的时基计数器的相位选择位（EPWM Time Base Phase）** 当 PHSEN（TBCTL[2]）= 0 时，同步事件会被忽略，时基计数器不加载相位 当 PHSEN（TBCTL[2]）= 1 时，当同步事件发生时时基计数器加载相位。同步事件由输入同步信号（SYNCI）激励产生，也可以由软件强制产生

21.5.2.3 EPWM 时基计数寄存器（EPWM_TBCTR）

- 名称：EPWM Time Base Counter Register
- 偏移地址：0x08
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		TBCTR
访问		R/W
复位值		0x0

字段	说明
[15:0] TBCTR	**时基计数器当前值（EPWM Time Base Counter）** 必须给 READ_TBCTR 写 1 并等待 READ_TBCTR 变为 0 才有效。

21.5.2.4 EPWM 时基控制寄存器（EPWM_TBCTL）

- 名称：EPWM Time Base Control Register
- 偏移地址：0x0C
- 默认值：0x00000003
- 返回寄存器列表

位	31:14	13	12:10	9	8	7
名称		TBDIR		READTBCTR	PWMSYNCSEL	SYNCOSELH
访问		R/W		R/WAC	R/W	R/W
复位值		0x0		0x0	0x0	0x0
位	6	5:4	3	2	1:0	
名称	SWFSYNC	SYNCOSELL	PRDLD	PHSEN	CTRMODE	
访问	R/WAC	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	0x3	

字段	说明
[13] TBDIR	**相对于提供同步输入信号的时基，所选的 EPWM 的时基计数器的方向选择位（EPWM Time Base Direction）** 当 PHSEN（TBCTL[2]）＝0 时，如果忽略同步事件，时基计数器不加载方向 当 PHSEN（TBCTL[2]）＝1 时，时基计数器加载该方向。同步事件由输入同步信号（SYNCI）激励产生，也可以由软件强制产生 注意：该位仅当增-减模式下有效。
[9] READTBCT	**TBCTR 读操作启动控制位（EPWM Time Base Read Counter）** 0：不启动 TBCTR 的读操作 1：启动 TBCTR 的读操作，当 TBCTR 正在进行读操作，READTBCTR 保持为 1；当 TBCTR 完成读操作，此位自动清 0
[8] PWMSYNCSEL	**PWMSYNC 源选择位（EPWM PWMSYNC Source Select）** PWMSYNC 信号应用于将其他外设（如 DAC+CMP）与指定 PWM 模块时间同步。 0：PWMSYNC 信号由 TBCTR=PRD 脉冲产生 1：PWMSYNC 信号由 TBCTR=ZRO 脉冲产生
[7] SYNCOSELH	**SYNCOSEL 寄存器最高位（EPWM Sync Out Select High）** SYNCOSEL={SYNCOSELH，SYNCOSELL}，由此位域与最低位组合而成
[6] SWFSYNC	**软件同步控制位（EPWM Soft Force Sync）** 0：无任何操作 1：强制发生软件同步，当软件同步正在进行，此位保持为 1；当软件同步完成，此位自动清 0

[5:4] SYNCOSELL	**SyncOut 信号选择控制位最低位（EPWM Sync Out Select Low）** 000：SyncIn 001：CTR = 0，时基计数器的值为 0 010：CTR = CMPB，时基计数器等于 CMPB 的值 100：CTR = CMPA，时基计数器等于 CMPA 的值 101：CTR = CMPC，时基计数器等于 CMPC 的值 Other：SyncOut 输出 0
[3] PRDLD	**TBPRD（TBPRD_SD[15:0]）从影子寄存器（Shadow Register）处加载数据的控制位（EPWM Time Base Period Load）** 0：当时基计数器为 0 时 TBPRD（激活寄存器）从影子寄存器处加载数据 1： TBPRD（激活寄存器）可直接更新，不经过影子寄存器，对 TBPRD 的读写操作直接影响激活寄存器，立即发生改变
[2] PHSEN	**时基计数器与相位寄存器的相位同步使能位（EPWM Time Base Phase Enable）** 0：同步脉冲发生时，时基计数器（[TBCTR]）不加载相位寄存器的值 1：同步脉冲发生时，时基计数器（[TBCTR]）加载相位寄存器的值
[1:0] CTRMODE	**时基计数器模式选择位（EPWM Time Base Counter Mode）** 时基计数器的模式一旦选定就不能再更改，如果更改了计数器的模式，更改将在下一个时基计数器时钟周期来临时生效，并且当前计数器的值会在模式更改前增加或减少。 00：递增模式，时基计数器的值将不断增加至周期指（TBPRD），然后复位为 0 01：递减模式，时基计数器将从周期值开始递减，直到 0 再回周期值 10：增-减模式，时基计数器将从 0 开始递增至周期值，再递减至 0 11：时基计数器不发生变化

21.5.2.5 EPWM 时基状态寄存器（EPWM_TBSTS）

- 名称：EPWM Time Base Status Register
- 偏移地址：0x10
- 默认值：0x00000001
- 返回寄存器列表

位	31:3	2	1	0
名称		CTRMAX	SYNCI	TBCTR_DIR
访问		R/W1C	R/W1C	R
复位值		0x0	0x0	0x1

字段	说明
[2] CTRMAX	**时基计数器最大值锁存状态位（EPWM Time Base Maximum Latched Flag）** 0：时基计数器没有到达过最大值。写 0 没有作用。 1：时基计数器到达过最大值 0xFFFF。写 1 会清除这个锁存事件。
[1] SYNCI	**输入同步锁存状态位（EPWM Time Base Sync Latched Flag）** 0：无任何效果，当该位为 0 说明没有外部同步事件发生 1：该位为 1 说明有外部同步事件发生（EPWMxSYNCI）。向该位写 1 将清除锁存事件。
[0] TBCTR_DIR	**时基计数器方向状态控制位（EPWM Time Base Counter Direction Status）** 复位时计数器被冻结，因此该位无意义。为了让该位有意义，需要先将 CTRMODE（TBCTL[1:0]）设置为适当的模式。 0：当前时基计数器向下计数 1：当前时基计数器向上计数

21.5.2.6 EPWM 计数比较 A 寄存器（EPWM_CMPA）

- 名称：EPWM Counter Compare A Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CMPA
访问		R/W
复位值		0x0

字段	说明
[15:0] CMPA	**时基计数比较器 A 的值的读取位（EPWM Counter Compare A）** 默认写入此寄存器的数据保存在影子寄存器中。由 CMPCTL[4]决定影子寄存器的打开和关闭。 当 SHDWAMODE（CMPCTL[4]）为 0 时，使能影子寄存器。对本寄存器的数据写入将保存在影子寄存器中。LOADAMODE（CMPCTL[1:0]）位决定影子寄存器中哪一个事件会被加载进激活寄存器中。 当 SHDWAMODE（CMPCTL[4]）为 1 时，关闭影子寄存器。写入的数据将直接进入激活寄存器，即寄存器主动控制硬件 注意：在执行写操作之前，会读取 SHDWAFULL（CMPCTL[6]）位的值判断影子 FIFO 是否已满。

21.5.2.7 EPWM 计数比较 B 寄存器（EPWM_CMPB）

- 名称：EPWM Counter Compare B Register
- 偏移地址：0x24
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CMPB
访问		R/W
复位值		0x0

字段	说明
[15:0] CMPB	**时基计数比较器 B 的值的读取位（EPWM Counter Compare B）** 默认写入的数据保存在影子寄存器中。由 SHDWBMODE（CMPCTL[5]）决定影子寄存器的打开与关闭。当为 0 时，使能影子寄存器。写入的数据会自动保存在影子寄存器中。LOADBMODE（CMPCTL[3:2]）决定了哪一个事件从影子寄存器加载进激活寄存器中；当 SHDWBMODE（CMPCTL[5]）为 1 时，关闭影子寄存器。写入的数据直接保存在激活寄存器中，由该寄存器主动控制硬件。 注意：执行写操作之前，先读取 SHDWBFULL（CMPCTL[7]）的值判断影子 FIFO 是否已满。

21.5.2.8 EPWM 计数比较 C 寄存器（EPWM_CMPC）

- 名称：EPWM Counter Compare C Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CMPC
访问		R/W
复位值		0x0

字段	说明
[15:0] CMPC	**时基计数比较器 C 的值的读取位（EPWM Counter Compare C）** 此特殊功能寄存器（SFR）只用于启动 ADC 转换。 默认写入的数据保存在影子寄存器中，由 SHDWCMODE（CMPCTL[10]）决定影子寄存器的打开与关闭。 当 SHDWCMODE（CMPCTL[10]）为 0 时，使能影子寄存器。写入的数据将自动保存在影子寄存器中。LOADCMODE（CMPCTL[9:8]）决定哪一个事件从影子寄存器中加载进激活寄存器；当 SHDWCMODE（CMPCTL[10]）为 1 时，关闭影子寄存器。写入的数据直接保存在激活寄存器中，由该寄存器主动控制硬件。 注意：执行写操作之前，可以先读取 SHDWCFULL（CMPCTL[11]）的值判断当前影子 FIFO 是否已满。

21.5.2.9 EPWM 计数比较控制寄存器（EPWM_CMPCTL）

- **名称：EPWM Counter Compare Control Register**
- **偏移地址：0x2C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16		15	14	13:12	11:10	9
名称			SHDWCFULL	SHDWCMODE	LOADCMODE		SHDWBFULL
访问			R	R/W	R/W		R
复位值			0x0	0x0	0x0		0x0
位	8	7	6	5	4	3:2	1:0
名称	SHDWAFULL		SHDWBMODE		SHDWAMODE	LOADBMODE	LOADAMODE
访问	R		R/W		R/W	R/W	R/W
复位值	0x0		0x0		0x0	0x0	0x0

字段	说明
[15] SHDWCFULL	**时基计数比较器 C 影子寄存器已满的状态标志位（EPWM Counter Compare C Shadow Full）** 每发生一次 load-strobe，该位就自动清 0。 0：比较器 C 的影子 FIFO 还未满 1：比较器 C 的影子 FIFO 已满，执行 CPU 写操作时会覆盖当前影子 FIFO 的值
[14] SHDWCMODE	**时基计数比较器 C 寄存器工作模式选择位（EPWM Counter Compare C Shadow Mode）** 0：影子模式（shadow mode），运行双 buffer。通所有的写操作都会通过 CPU 访问影子寄存器 1：立即模式（immediate mode），只使用了比较器 C 激活寄存器。所有的读写操作都可以直接访问比较器 C 激活寄存器本身执行比较操作
[13:12] LOADCMODE	**时基计数比较器 C 的激活比较寄存器从时基计数比较器 C 的影子寄存器加载数据的模式选择位（EPWM Counter Compare C Load Mode）** 当在立即模式（CMPCTL[14] = 1）下，本位无效。 00：当时基计数器等于 0 时加载数据（TBCTR = 0x0000） 01：当时基计数器等于周期值时加载数据（TBCTR = TBPRD） 10：当时基计数器等于 0 或等于周期值时加载数据（TBCTR = 0x0000 或 TBCTR = TBPRD） 11：时基计数器被冻结，不可能进行数据加载
[9] SHDWBFULL	**时基计数比较器 B 影子寄存器已满的状态标志位（EPWM Counter Compare B Shadow Full）** 当 load-strobe 发生时本位自动清 0 0：比较器 B 影子 FIFO 未满 1：比较器 B 影子 FIFO 已满；执行 CPU 写操作将覆盖当前影子 FIFO 的值
[8] SHDWAFULL	**时基计数比较器 A 影子寄存器已满的状态标志位（EPWM Counter Compare A Shadow Full）** 每发生一次 load-strobe，该标志位会自动清 0 0：CMPA 影子 FIFO 未满 1：CMPA 影子 FIFO 已满，执行 CPU 写操作将覆盖当前影子 FIFO 的值

[6] SHDWBMODE	**时基计数比较器 B 寄存器工作模式选择位（EPWM Counter Compare B Shadow Mode）** 0：影子模式（shadow mode），运行双 buffer。通所有的写操作都会通过 CPU 访问影子寄存器 1：立即模式（immediate mode），只使用了比较器 B 激活寄存器。所有的读写操作都可以直接访问比较器 B 激活寄存器本身执行比较操作
[4] SHDWAMODE	**时基计数比较器 A 寄存器工作模式选择位（EPWM Counter Compare A Shadow Mode）** 0：影子模式（shadow mode），运行双 buffer。所有的写操作都会通过 CPU 访问影子寄存器 1：立即模式（immediate mode），仅使用比较 A 激活寄存器。所有的读写操作都可以直接访问比较器 A 激活寄存器本身立即进行比较
[3:2] LOADBMODE	**时基计数比较器 B 的比较器从时基计数比较器 B 的影子寄存器加载数据的选择位（EPWM Counter Compare B Load Mode）** 当 CMPB 工作在立即模式下（CMPCTL[6] = 1）时，本位无效。 00：时基计数器等于 0 时加载数据（TBCTR = 0x0000） 01：当时基计数器等于周期值时加载数据（TBCTR = TBPRD） 10：当时基计数器等于 0 或等于周期值时加载数据（TBCTR = 0x0000 或 TBCTR = TBPRD），即在一个周期内加载两次数据 11 时基计数器被冻结，无法加载数据
[1:0] LOADAMODE	**时基计数比较器 A 的比较器从时基计数比较器 A 的影子寄存器加载数据的选择位（EPWM Counter Compare A Load Mode）** 当 CMPA 工作在立即模式（CMPCTL[4] = 1）下时，本位无效。 00：当时基计数器等于 0 时加载数据（TBCTR = 0x0000） 01：当时基计数器等于周期值时加载数据（TBCTR = TBPRD） 10：当时基计数器等于 0 或等于周期值时加载数据（TBCTR = 0x0000 或 TBCTR = TBPRD），即一个周期内加载两次数据 11：时基计数器被冻结，无法加载数据

21.5.2.10 EPWM 动作限定输出 A 控制寄存器（EPWM_AQCTLA）

- 名称：EPWM Action Qualifier Output A Control Register
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:10	9:8	7:6	5:4	3:2	1:0
名称		CBDA	CBUA	CADA	CAUA	PRDA	ZROA
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[11:10] CBDA	**当时基计数器向下计数且等于 CMPB 激活寄存器的值时，产生动作（EPWM Action Qualifier Output A When Compare B Down）** 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转， EPWMxA 输出低电平强制翻转为高电平；输出高电平时强制翻转为低电平
[9:8] CBUA	**当时基计数器向上计数且等于 CMPB 激活寄存器的值时，产生动作（EPWM Action Qualifier Output A When Compare B Up）** 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转，EPWMxA 输出低电平时强制翻转为高电平；输出高电平时强制翻转为低电平
[7:6] CADA	**当时基计数器向下计数且等于 CMPA 激活寄存器的值时，产生动作（EPWM Action Qualifier Output A When Compare A Down）** 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转，EPWMxA 输出低电平时强制翻转为高电平；输出高电平时强制翻转为低电平
[5:4] CAUA	**当时基计数器向上计数且等于 CMPA 激活寄存器的值时，产生动作（EPWM Action Qualifier Output A When Compare A Up）** 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转，EPWMxA 输出低电平时强制翻转为高电平；输出高电平时强制翻转为低电平
[3:2] PRDA	**当时基计数器等于周期值时，产生动作（EPWM Action Qualifier Output A When Period）** 注意：在默认情况下，当计数值等于周期值且计数器处于增-减计数模式下，方向定义为 0 或向下计数 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转，EPWMxA 输出低电平时强制翻转为高电平；输出高电平时强制翻转为低电平

[1:0] ZROA	**当时基计数器等于 0 时，产生动作（EPWM Action Qualifier Output A When Zero）** 注意：在默认情况下，当计数值为 0 且处于增-减计数模式下时，方向定义为 1 或向上计数。 00：不动作 01：清 0，强制 EPWMxA 输出低电平 10：置 1，强制 EPWMxA 输出高电平 11：翻转，EPWMxA 输出低电平时强制翻转为高电平；输出高电平时强制翻转为低电平

21.5.2.11 EPWM 动作限定输出 B 控制寄存器（EPWM_AQCTLB）

- **名称：EPWM Action Qualifier Output B Control Register**
- **偏移地址：0x34**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:12	11:10	9:8	7:6	5:4	3:2	1:0
名称		CBDB	CBUB	CADB	CAUB	PRDB	ZROB
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[11:10] CBDB	**当时基计数器向下计数并且等于 CMPB 的激活寄存器时，产生动作（EPWM Action Qualifier Output B When Compare B Down）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时会被强制翻转为高电平；输出高电平时会被强制翻转为低电平
[9:8] CBUB	**当时基计数器向上计数并且等于 CMPB 的激活寄存器时，产生动作（EPWM Action Qualifier Output B When Compare B Up）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时会被强制翻转为高电平；输出高电平时会被强制翻转为低电平
[7:6] CADB	**当时基计数器递减计数且等于 CMPA 的激活寄存器时，产生动作（EPWM Action Qualifier Output B When Compare A Down）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时会被强制翻转为高电平；输出高电平时会被强制翻转为低电平

[5:4] CAUB	**当时基计数器向上计数且等于 CMPA 的激活寄存器时，产生动作（EPWM Action Qualifier Output B When Compare A Up）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时会被强制翻转为高电平；输出高电平时会被强制翻转为低电平
[3:2] PRDB	**当时基计数器等于周期值时，产生动作（EPWM Action Qualifier Output B When Period）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时会被强制翻转为高电平；输出高电平时会被强制翻转为低电平 注意：在默认情况下，当计数器的值等于周期值且在计数器增-减模式下时，计数器方向定义为 0 或向下计数。
[1:0] ZROB	**当时基计数器等于 0 时，产生动作（EPWM Action Qualifier Output B Zero）** 00：不动作 01：清 0，强制 EPWMxB 输出低电平 10：置 1，强制 EPWMxB 输出高电平 11：翻转，EPWMxB 输出低电平时强制翻转为高电平；EPWMxB 输出高电平时强制翻转为低电平 注意：在默认情况下，当计数值等于 0 且计数器处于增-减计数模式下，方向定义为 1 或向上计数。

21.5.2.12 EPWM 动作限定软件强制寄存器（EPWM_AQSFRC）

- **名称：EPWM Action Qualifier Soft Force Register**
- **偏移地址：0x38**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:8	7:6	5	4:3	2	1:0
名称		RLDCSF	OTSFB	ACTSFB	OTSFA	ACTSFA
访问		R/W	W	R/W	W	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[7:6] RLDCSF	**AQSFRC 和 AQCSFRC 的激活寄存器从对应的影子寄存器处再次加载数据的时机选择位（EPWM Actions Qualifier AQSFRC/AQCSFRC Reload Mode）** 00：当时基计数器等于 0 时加载数据（TBCTR = 0x0000） 01：时基计数器等于周期值时加载数据（TBCTR = TBPRD） 10：当时基计数器等于 0 或等于周期值时加载数据（TBCTR = 0x0000 或 TBPRD，即一个周期内加载两次数据） 11：立即加载，执行读写操作时，由 CPU 直接访问 AQSFRC 和 AQCSFRC 激活寄存器无需经过其影子寄存器

[5] OTSFB	**输出 B 上的单次软件强制事件（EPWM One-Shot Soft Force Output B）** 0：没有任何效果，读返回 0。 1：激发一个单次软件强制事件 每激发一个单次强制事件，该位就会自动清 0。
[4:3] ACTSFB	**当单次软件强制事件 B 被调用时，产生动作（EPWM Action One-Shot Soft Force Output B）** 00：不动作 01：清 0，强制输出低电平 10：置 1，强制输出高电平 11：翻转，将输出的低电平翻转为高电平；输出的高电平翻转为低电平 注意：此动作不受计数器的方向限制
[2] OTSFA	**输出 A 上的单次软件强制事件（EPWM One-Shot Soft Force Output A）** 0：没有任何效果，读返回 0。 1：激发一个单次软件强制事件 每激发一个单次强制事件，该位就会自动清 0。
[1:0] ACTSFA	**当单次软件强制事件 A 被调用时，产生动作（EPWM Action One-Shot Soft Force Output A）** 00：不动作 01：清除，输出低电平 10：置位，输出高电平 11：翻转，高电平翻转为低电平，低电平翻转为高电平 注意：此动作不受计数器方向限制。

21.5.2.13 EPWM 动作限定连续软件强制寄存器（EPWM_AQCSFRC）

- **名称：EPWM Action Qualifier Continuous Soft Force Register**
- **偏移地址：0x3C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:4	3:2	1:0
名称		CSFB	CSFA
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[3:2] CSFB	**输出 B 上的连续软件强制事件选择位（EPWM Continuous Soft Force Output B）** 在立即模式下，一个连续强制事件作用于下一个 TBCLK 边沿。 在影子模式下，通过设置 RLDCSF（AQSFRC[7:6]）位配置影子模式，加载最新的配置到激活寄存器后，一个连续强制事件作用于下一个 TBCLK 边沿。 00：不使能，无任何效果 01：在输出 B 上强制输出持续低电平 10：在输出 B 上强制输出持续高电平 11：不使能软件强制，没有任何效果

[1:0] CSFA	**输出 A 上的连续软件强制事件选择位（EPWM Continuous Soft Force Output A）** 在立即模式下，一个连续强制事件作用于下一个 TBCLK 边沿。 在影子模式下，通过设置 RLDCSF（AQSFRC[7:6]）位配置影子模式，加载最新的配置到激活寄存器后，一个连续强制事件作用于下一个 TBCLK 边沿。 00：不使能强制，无任何效果 01：在输出 A 上强制输出持续低电平 10：在输出 A 上强制输出持续高电平 11：不使能软件强制，没有任何效果

21.5.2.14 EPWM 死区控制寄存器（EPWM_DBCTL）

- **名称：EPWM Dead Band Control Register**
- **偏移地址：0x40**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15	14:6	5:4	3:2	1:0
名称		HALFCYCLE		IN_MODE	POLSEL	OUT_MODE
访问		R/W		R/W	R/W	R/W
复位值		0x0		0x0	0x0	0x0

字段	说明
[15] HALFCYCLE	**半/全周期时钟使能位（EPWM Dead Band Half Cycle）** 0：使能全周期时钟。每得到一个 TBCLK 的正边沿，死区计数器就会加 1 1：使能半周期时钟。每得到一个 TBCLK 正边沿，死区计数器就会加 2
[5:4] IN_MODE	**死区输入模式控制位（EPWM Dead Band Input Mode）** IN_MODE 允许用户为下降沿和上升沿延时选择输入源。为了产生典型的死区波形，默认由 EPWMxA In 作为上升沿和下降沿延时的来源。 00：EPWMxA In（来自动作产生子模块）是上升沿和下降沿延时的来源 01：EPWMxB In（来自动作产生子模块）是上升沿延时的信号源；EPWMxA In（来自动作产生子模块）是下降沿延时的信号源 10：EPWMxA In（来自动作产生子模块）是上升沿延时的信号源；EPWMxB In（来自动作产生子模块）是下降沿延时的信号源 11：EPWMxB In（来自动作产生子模块）是上升沿延时和下降沿的信号源
[3:2] POLSEL	**极性选择控制位（EPWM Dead Band Polarity Select）** POLSEL 允许用户在延时信号输出到死区子模块之前可以有选择地转换其中一个延时信号。以下描述相对于数字电机控制逆变器一条支路的典型上/下开关控制。当 OUT_MODE（DBCTL[1:0]）=“11”、IN_MODE（DBCTL[5:4]）= “00”时，有可能可以组成其他增强模式，但一般不使用这些模式。 00：高电平有效（AH）模式，EPWMxA 和 EPWMxB 都不翻转（默认情况） 01：激活低电平互补（ALC）模式，EPWMxA 翻转后输出 10：激活高电平互补（AHC）模式，EPWMxB 翻转后输出 11：低电平有效（AL）模式，EPWMxA 和 EPWMxB 翻转后输出

[1:0] OUT_MODE	**死区输出模式控制位（EPWM Dead Band Output Mode）** OUT_MODE 允许用户选择使能或忽略上升沿或下降沿延时的死区生成。 00：两个输出信号都忽略死区产生。在此模式下，来自动作产生子模块的 EPWMxA 和 EPWMxB 的输出可以直接传输到事件触发子模块。在此模式下，POLSEL（DBCTL[3:2]）和 IN_MODE（DBCTL[5:4]）位都没有任何效果。 01：不使能上升沿延迟。动作产生子模块的 EPWMxA 信号可以直接传输到 Event-trigger 子模块的 EPWMxA 输入。下降沿延时信号在 EPWMxB 的输出处可见。延时信号的输出则由 IN_MODE（DBCTL[5:4]）所决定。 10：上升沿延时信号在 EPWMxA 输出处可见。由 IN_MODE（DBCTL[5:4]）位决定输入信号的延时；不使能下降沿延时，动作产生子模块的 EPWMxB 信号可以直接传输到事件触发子模块的 EPWMxB 的输入。 11：EPWMxA 输出的上升沿延时和 EPWMxBD 输出的下降沿延时的死区全使能。由 IN_MODE（DBCTL[5:4]）决定输入信号的延时。

21.5.2.15 EPWM 死区上升沿延迟寄存器（EPWM_DBRED）

- **名称：EPWM Dead Band Rising Edge Delay Register**
- **偏移地址：0x44**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:10	9:0
名称		RED
访问		R/W
复位值		0x0

字段	说明
[9:0] RED	**上升沿延时计数器（EPWM Dead Band Rising Edge Delay）** 10 位计数器。

21.5.2.16 EPWM 死区下降沿延迟寄存器（EPWM_DBFED）

- 名称：EPWM Dead Band Falling Edge Delay Register
- 偏移地址：0x48
- 默认值：0x00000000
- 返回寄存器列表

位	31:10	9:0
名称		FED
访问		R/W
复位值		0x0

字段	说明
[9:0]	**下降沿延时计数器（EPWM Dead Band Falling Edge Delay）**
FED	10 位计数器。

21.5.2.17 EPWMPWM 斩波寄存器（EPWM_PCCTL）

- 名称：EPWM PWM-Chopper Control Register
- 偏移地址：0x50
- 默认值：0x00000000
- 返回寄存器列表

位	31:11	10:8	7:5	4:1	0
名称		CHPDUTY	CHPFREQ	OSHTWTH	CHPEN
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0

字段	说明
[10:8]	**斩波占空比值（EPWM PWM Chopper Duty）**
CHPDUTY	此位定义斩波信号的占空比 111：7/8 …… 000：0/8（仅存在第一个脉冲）
[7:5]	**斩波载波频率值（EPWM PWM Chopper Frequency）**
CHPFREQ	此位定义载波频率为 FCHPFRQ＝f_{EPWM} / （16*（CARFRQ＋1）） 111：1.25 MHz（f_{EPWM} / 128） …… 000：10 MHz（f_{EPWM} / 16）

[4:1]	**斩波启动脉冲宽度（EPWM PWM Chopper One-Shot Width）**
OSHTWTH	此位定义在输出信号的上升沿之后的初始脉冲宽度
	t_{1STPW} ＝ （OSHTWTH＋1）*16*t_{EPWM}
	1111：1.6 us（16/10MHz）
	……
	0000：100 ns（1/10MHz）
	注意：此处假定 t_{EPWM} = 160 MHz
[0]	**斩波输出使能（EPWM PWM Chopper Enable）**
CHPEN	此位开启输出斩波
	1：输出斩波开启
	0：输出斩波关闭

21.5.2.18 EPWM 关断区域选择寄存器（EPWM_TZSEL）

- **名称：EPWM Trip Zone Select Register**
- **偏移地址：0x60**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15	14	13	12	11
名称		DCBEVT1_OSTEN	DCAEVT1_OSTEN	OSHT5	OSHT4	OSHT3
访问		R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0
位	10	9	8	7	6	5
名称	OSHT2	OSHT1	OSHT0	DCBEVT2_CBCEN	DCAEVT2_CBCEN	CBC5
访问	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0	0x0	0x0
位	4	3	2	1	0	
名称	CBC4	CBC3	CBC2	CBC1	CBC0	
访问	R/W	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	0x0	

字段	说明
[15]	**数字比较输出事件 B1 的使能位（EPWM Trip Zone DCBEVT1 One-Shot Enable）**
DCBEVT1_OSTEN	0：不使能 DCBEVT1 作为 EPWM 模块的单次触发源
	1：使能 DCBEVT1 作为 EPWM 模块的单次触发源
[14]	**数字比较输出事件 A1 使能位（EPWM Trip Zone DCAEVT1 One-Shot Enable）**
DCAEVT1_OSTEN	0：不使能 DCAEVT1 作为 EPWM 模块的单次触发源
	1：使能 DCAEVT1 作为 EPWM 模块的单次触发源

[13] OSHT5	**故障区 5（TZ5）的使能选择位（EPWM Trip Zone TZ5 One-Shot Enable）** 0：不使能 TZ5 作为 EPWM 模块的单次触发源 1：使能 TZ5 作为 EPWM 模块的单次触发源
[12] OSHT4	**故障区 4（TZ4）的使能选择位（EPWM Trip Zone TZ4 One-Shot Enable）** 0：不使能 TZ5 作为 EPWM 模块的单次触发源 1：使能 TZ5 作为 EPWM 模块的单次触发源
[11] OSHT3	**故障区 3（TZ3）的使能选择位（EPWM Trip Zone TZ3 One-Shot Enable）** 0：不使能 TZ3 作为 EPWM 模块的单次触发源 1：使能 TZ3 作为 EPWM 模块的单次触发源
[10] OSHT2	**故障区 2（TZ2）的使能选择位（EPWM Trip Zone TZ2 One-Shot Enable）** 0：不使能 TZ2 作为 EPWM 模块的单次触发源 1：使能 TZ2 作为 EPWM 模块的单次触发源
[9] OSHT1	**故障区 1（TZ1）的使能选择位（EPWM Trip Zone TZ1 One-Shot Enable）** 0：不使能 TZ1 作为 EPWM 模块的单次触发源 1：使能 TZ1 作为 EPWM 模块的单次触发源
[8] OSHT0	**故障区 0（TZ0）的使能选择位（EPWM Trip Zone TZ0 One-Shot Enable）** 0：不使能 TZ0 作为 EPWM 的模块单次触发源 1：使能 TZ0 作为 EPWM 的模块单次触发源
[7] DCBEVT2_CBCEN	**数字比较输出事件 B2 选择位（EPWM Trip Zone DCBEVT2 Cycle-By-Cycle Enable）** 0：不使能 DCBEVT2 作为 EPWM 模块的周期循环触发源 1：使能 DCBEVT2 作为 EPWM 模块的周期循环触发源
[6] DCAEVT2_CBCEN	**数字比较输出事件 A2 选择位（EPWM Trip Zone DCAEVT2 Cycle-By-Cycle Enable）** 0：不使能 DCAEVT2 作为 EPWM 模块的周期循环触发源 1：使能 DCAEVT2 作为 EPWM 模块的周期循环触发源
[5] CBC5	**故障区 5（TZ5）的使能选择位（EPWM Trip Zone TZ5 Cycle-By-Cycle Enable）** 0：不使能 TZ5 作为 EPWM 模块的周期循环触发源 1：使能 TZ4 作为 EPWM 模块的周期循环触发源
[4] CBC4	**故障区 4（TZ4）的使能选择位（EPWM Trip Zone TZ4 Cycle-By-Cycle Enable）** 0：不使能 TZ4 作为 EPWM 模块的周期循环触发源 1：使能 TZ4 作为 EPWM 模块的周期循环触发源
[3] CBC3	**故障区 3（TZ3）的使能选择位（EPWM Trip Zone TZ3 Cycle-By-Cycle Enable）** 0：不使能 TZ3 作为 EPWM 模块的周期循环触发源 1：使能 TZ3 作为 EPWM 模块的周期循环触发源
[2] CBC2	**故障区 2（TZ2）的使能选择位（EPWM Trip Zone TZ2 Cycle-By-Cycle Enable）** 0：不使能 TZ2 作为 EPWM 模块的周期循环触发源 1：使能 TZ2 作为 EPWM 模块的周期循环触发源
[1] CBC1	**故障区 1（TZ1）的使能选择位（EPWM Trip Zone TZ1 Cycle-By-Cycle Enable）** 0：不使能 TZ1 作为 EPWM 模块的周期循环触发源 1：使能 TZ1 作为 EPWM 模块的周期循环触发源
[0] CBC0	**故障区 0（TZ0）的使能选择位（EPWM Trip Zone TZ0 Cycle-By-Cycle Enable）** 0：不使能 TZ0 作为 EPWM 模块的周期循环触发源 1：使能 TZ0 作为 EPWM 模块的周期循环触发源

21.5.2.19 EPWM 关断区域控制寄存器（EPWM_TZCTL）

- 名称：EPWM Trip Zone Control Register
- 偏移地址：0x64
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:10	9:8	7:6	5:4	3:2	1:0
名称		DCBEVT2_FRCEN	DCBEVT1_FRCEN	DCAEVT2_FRCEN	DCAEVT1_FRCEN	TZB	TZA
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[11:10] DCBEVT2_FRCEN	**数字比较输出事件 B2 在 EPWMxB 的动作选择位（EPWM Trip Zone DCBEVT2 Action Output B）** 00：高阻抗（EPWMxB =高阻抗状态） 01：强制 EPWMxB 输出为高 10：强制 EPWMxB 输出为低 11：无任何效果，不使能 trip
[9:8] DCBEVT1_FRCEN	**数字比较输出事件 B1 在 EPWMxB 的动作选择位（EPWM Trip Zone DCBEVT1 Action Output B）** 00：高阻抗（EPWMxB =高阻抗状态） 01：强制 EPWMxB 输出为高 10：强制 EPWMxB 输出为低 11：无任何效果，不使能 trip
[7:6] DCAEVT2_FRCEN	**数字比较输出事件 A2 在 EPWMxA 的动作选择位（EPWM Trip Zone DCAEVT2 Action Output A）** 00：高阻抗（EPWMxA =高阻抗状态） 01：强制 EPWMxA 输出为高 10：强制 EPWMxA 输出为低 11：无任何效果，不使能 trip
[5:4] DCAEVT1_FRCEN	**数字比较输出事件 A1 在 EPWMxA 的动作选择位（EPWM Trip Zone DCAEVT1 Action Output A）** 00：高阻抗（EPWMxA =高阻抗状态） 01：强制 EPWMxA 输出为高 10：强制 EPWMxA 输出为低 11：无任何效果，不使能 trip
[3:2] TZB	**EPWMxB 事件发生选择位（EPWM Trip Zone Trip Action Output B）** 当发生 trip 事件时，EPWMxB 会发生下列动作。Tripo_zone 引脚可以使得在 TZSEL 寄存器中定义事件。 00：高阻抗（EPWMxB = 高阻抗状态） 01：强制 EPWMxB 输出为高 10：强制 EPWMxB 输出为低 11：无任何效果，EPWMxB 无任何动作发生

[1:0] TZA	**EPWMxA 事件发生选择位（EPWM Trip Zone Trip Action Output A）** 当发生 trip 事件时，EPWMxA 会发生下列动作。Tripo_zone 引脚可以使得在 TZSEL 寄存器中定义事件。 00：高阻抗（EPWMxA = 高阻抗状态） 01：强制 EPWMxA 输出为高 10：强制 EPWMxA 输出为低 11：无任何效果，EPWMxA 无任何动作发生

21.5.2.20 EPWM 关断区域使能中断寄存器（EPWM_TZEINT）

- **名称：EPWM Trip Zone Enable Interrupt Register**
- **偏移地址：0x68**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		DCBEVT2INTE	DCBEVT1INTE	DCAEVT2INTE	DCAEVT1INTE	OSTINTE	CBCINTE	
访问		R/W	R/W	R/W	R/W	R/W	R/W	
复位值		0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[6] DCBEVT2INTE	**数字比较器输出事件 B2 的中断使能位（EPWM Trip Zone DCBEVT2 Interrupt Enable）** 0：不使能 1：使能
[5] DCBEVT1INTE	**数字比较器输出事件 B1 的中断使能位（EPWM Trip Zone DCBEVT1 Interrupt Enable）** 0：不使能 1：使能
[4] DCAEVT2INTE	**数字比较器输出事件 A2 的中断使能位（EPWM Trip Zone DCAEVT2 Interrupt Enable）** 0：不使能 1：使能
[3] DCAEVT1INTE	**数字比较器输出事件 A1 的中断使能位（EPWM Trip Zone DCAEVT1 Interrupt Enable）** 0：不使能 1：使能
[2] OSTINTE	**故障区单次中断使能位（EPWM Trip Zone One-Shot Interrupt Enable）** 0：不使能单次中断 1：使能单次中断，一个单次触发事件可以触发一个 TZINT 的中断
[1] CBCINTE	**故障区周期循环（CBC）中断使能位（EPWM Trip Zone Cycle-By-Cycle Interrupt Enable）** 0：不使能周期循环（CBC）中断 1：使能周期循环中断，一个周期循环事件可以触发一个 TZINT 中断

21.5.2.21 EPWM 关断区域标志寄存器（EPWM_TZFLAG）

- 名称：EPWM Trip Zone Flag Register
- 偏移地址：0x6C
- 默认值：0x00000000
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		DCBEVT2_FLAG	DCBEVT1_FLAG	DCAEVT2_FLAG	DCAEVT1_FLAG	OST	CBC	INT
访问		R	R	R	R	R	R	R
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[6] DCBEVT2_FLAG	**数字比较输出事件 B2 的状态标志位（EPWM Trip Zone DCBEVT2 Flag）** 0：DCBEVT2 没有发生 trip 事件 1：DCBEVT2 发生了一个 trip 事件
[5] DCBEVT1_FLAG	**数字比较输出事件 B1 的状态标志位（EPWM Trip Zone DCBEVT1 Flag）** 0：DCBEVT1 没有发生 trip 事件 1：DCBEVT1 发生了一个 trip 事件
[4] DCAEVT2_FLAG	**数字比较输出事件 A2 的状态标志位（EPWM Trip Zone DCAEVT2 Flag）** 0：DCAEVT2 没有发生 trip 事件 1：DCAEVT2 发生了一个 trip 事件
[3] DCAEVT1_FLAG	**数字比较输出事件 A1 的状态标志位（EPWM Trip Zone DCAEVT1 Flag）** 0：DCAEVT1 没有发生 trip 事件 1：DCAEVT1 发生了一个 trip 事件
[2] OST	**单次 trip 事件的状态标志位（EPWM Trip Zone One-Shot Flag）** 0：没有发生单次 trip 事件 1：单次 trip 信号源引脚上发生了一个单次 trip 事件 注意：当在 CLROST（EWPMx_TZCLR[2]）上写入适当的值时该位清 0。
[1] CBC	**周期循环 trip 事件的状态标志位（EPWM Trip Zone Cycle-By-Cycle Flag）** 0：不发生周期循环（CBC）trip 事件 1：在周期循环（CBC）trip 信号源的信号上发生一个 trip 事件。该位需由用户手动清 0。当该位清 0 后，如果周期循环 trip 事件仍然存在，那么该位将立即再次置 1。当 EPWM 的时基计数器等于 0 时（TBCTR = 0x0000），如果周期循环 trip 事件不再存在，那么将自动清除信号上的 trip 事件。无论在循环的什么时候 CBC 清 0，只有当 TBCTR = 0x0000 时，信号上的 trip 事件才会被清 0。当向 CLRCBC（EWPMx_TZCLR[1]）中写入适当的值，该位会被清 0。
[0] INT	**Trip 锁存中断状态标志位（EPWM Trip Zone Interrupt Flag）** 0：没有中断发生 1：TZINT PIE 中断发生，因为发生了 Trip 事件 注意：只有当该位清 0 之后才会产生下一个 TZINT PIE 中断。当 CBC 或 OST 置 1 时，如果中断标志位清 0，那么就会产生其他中断脉冲。清除所有的中断位可以防止产生更多中断。向 CLRINT（EWPMx_TZCLR[0]）写入适当的值可以清除该位。

21.5.2.22 EPWM 关断区域清除寄存器（EPWM_TZCLR）

- 名称：EPWM Trip Zone Clear Register
- 偏移地址：0x70
- 默认值：0x00000000
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		CLRDCBEVT2	CLRDCBEVT1	CLRDCAEVT2	CLRDCAEVT1	CLROST	CLRCBC	CLRINT
访问		W	W	W	W	W	W	W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[6] CLRDCBEVT2	**数字比较输出事件 B2 的状态清除位（EPWM Trip Zone DCBEVT2 Clear）** 写 1 清 0
[5] CLRDCBEVT1	**数字比较输出事件 B1 的状态清除位（EPWM Trip Zone DCBEVT1 Clear）** 写 1 清 0
[4] CLRDCAEVT2	**数字比较输出事件 A2 的状态清除位（EPWM Trip Zone DCAEVT2 Clear）** 写 1 清 0
[3] CLRDCAEVT1	**数字比较输出事件 A1 的状态清除位（EPWM Trip Zone DCAEVT1 Clear）** 写 1 清 0
[2] CLROST	**单次 trip 事件的状态清除位（EPWM Trip Zone One-Shot Clear）** 写 1 清 0
[1] CLRCBC	**周期循环 trip 事件的状态清除位（EPWM Trip Zone Cycle-By-Cycle Clear）** 写 1 清 0
[0] CLRINT	**Trip 中断状态清除位（EPWM Trip Zone Interrupt Clear）** 写 1 清 0

21.5.2.23 EPWM 关断区域强制寄存器（EPWM_TZFRC）

- 名称：EPWM Trip Zone Force Register
- 偏移地址：0x74
- 默认值：0x00000000
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		FRCDCBEVT2	FRCDCBEVT1	FRCDCAEVT2	FRCDCAEVT1	FRCOST	FRCCBC	
访问		W	W	W	W	W	W	
复位值		0x0	0x0	0x0	0x0	0x0	0x0	

字段	说明
[6] FRCDCBEVT2	**数字比较输出事件 B2 的强制标志位（EPWM Trip Zone DCBEVT2 Force）** 0：无任何效果，读返回 0 1：强制 DCBEVT2 事件 trip，并设置 DCBEVT2_FLAG（TZFLAG[6]）为 1
[5] FRCDCBEVT1	**数字比较输出事件 B1 的强制标志位（EPWM Trip Zone DCBEVT1 Force）** 0：无任何效果，读返回 0 1：强制 DCBEVT1 事件 trip，并设置 DCBEVT1_FLAG（TZFLAG[5]）为 1
[4] FRCDCAEVT2	**数字比较输出事件 A2 的强制标志位（EPWM Trip Zone DCAEVT2 Force）** 0：无任何效果，读返回 0 1：强制 DCAEVT2 事件 trip，并设置 DCAEVT2_FLAG（TZFLAG[4]）为 1
[3] FRCDCAEVT1	**数字比较输出事件 A1 的强制标志位（EPWM Trip Zone DCAEVT1 Force）** 0：无任何效果，读返回 0 1：强制 DCAEVT1 事件 trip，并设置 DCAEVT1_FLAG（TZFLAG[3]）为 1
[2] FRCOST	**单次 trip 事件的软件强制标志位（EPWM Trip Zone One-Shot Force）** 0：无任何效果，读返回 0 1：强制发生单次 trip 事件并设置 OST（TZFLAF[2]）为 1
[1] FRCCBC	**周期循环（CBC）trip 事件的软件强制标志位（EPWM Trip Zone Cycle-By-Cycle Force）** 0：无任何效果，读返回 0 1：强制发生一个周期循环事件并设置 CBC 位（TZFLAG[1]）为 1

21.5.2.24 EPWM 关断区域数字比较事件选择寄存器（EPWM_TZDCSEL）

- 名称：EPWM Trip Zone Digital Compare Event Select Register
- 偏移地址：0x78
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:9	8:6	5:3	2:0
名称		DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0

字段	说明	
[11:9]	**数字比较器输出 B 事件 2 的选择位（EPWM Trip Zone DCBEVT2 Select）**	
DCBEVT2	000：没有事件发生	001：DCBH = 低电平, DCBL = 无效
	010：DCBH = 高电平, DCBL = 无效	011：DCBL = 低电平, DCBH = 无效
	100：DCBL = 高电平, DCBH = 无效	101：DCBL = 高电平, DCBH = 低电平
	110：DCBL = 低电平, DCBH = 高电平	111：保留
[8:6]	**数字比较器输出 B 事件 1 的选择位（EPWM Trip Zone DCBEVT1 Select）**	
DCBEVT1	000：没有事件发生	001：DCBH = 低电平, DCBL = 无效
	010：DCBH = 高电平, DCBL = 无效	011：DCBL = 低电平, DCBH = 无效
	100：DCBL = 高电平, DCBH = 无效	101：DCBL = 高电平, DCBH = 低电平
	110：DCBL = 低电平, DCBH = 高电平	111：保留
[5:3]	**数字比较器输出 A 事件 2 的选择位（EPWM Trip Zone DCAEVT2 Select）**	
DCAEVT2	000：没有事件发生	001：DCAH = 低电平, DCAL = 无效
	010：DCAH = 高电平, DCAL = 无效	011：DCAL = 低电平, DCAH = 无效
	100：DCAL = 高电平, DCAH = 无效	101：DCAL = 高电平, DCAH = 低电平
	110：DCAL = 低电平, DCAH = 高电平	111：保留
[2:0]	**数字比较器输出 A 事件 1 的选择位（EPWM Trip Zone DCAEVT1 Select）**	
DCAEVT1	000：没有事件发生	001：DCAH = 低电平, DCAL = 无效
	010：DCAH = 高电平, DCAL = 无效	011：DCAL = 低电平, DCAH = 无效
	100：DCAL = 高电平, DCAH = 无效	101：DCAL = 高电平, DCAH = 低电平
	110：DCAL = 低电平, DCAH = 高电平	111：保留

21.5.2.25 EPWM 数字比较关断选择寄存器（EPWM_DCTRIPSEL）

- 名称：EPWM Digital Compare Trip Select Register
- 偏移地址：0x80
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:12	11:8	7:4	3:0
名称		DCBLCMPSEL	DCBHCMPSEL	DCALCMPSEL	DCAHCMPSEL
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0

字段	说明
[15:12] DCBLCMPSEL	**数字比较器 B 低电平(DCBL)输入信号选择位，定义了 DCBL 的源信号(EPWM Digital Compare DCBL Select)** 当 TZ 信号用作故障信号时，作为正常输入来处理并且定义为高电平有效或低电平有效。 0000：TZ0 输入信号　0001：TZ1 输入信号　0010：TZ2 输入信号 0011：TZ3 输入信号　1000：CMP0OUT 输入信号　1001：CMP1OUT 输入信号 1010：CMP2OUT 输入信号　1100：PDM0CMPH 输入信号 1101：PDM0CMPL 输入信号　1110：PDM1CMPH 输入信号 1111：PDM1CMPL 输入信号 注意：没有显示的值作为保留。
[11:8] DCBHCMPSEL	**数字比较器 B 高电平（DCBH）输入信号选择位，定义 DCBH 输入的信号源（EPWM Digital Compare DCBH Select）** 当 TZ 信号用作故障信号时，作为正常输入来处理并且定义为高电平有效或低电平有效。 0000：TZ0 输入信号　0001：TZ1 输入信号　0010：TZ2 输入信号 0011：TZ3 输入信号　1000：CMP0OUT 输入信号　1001：CMP1OUT 输入信号 1010：CMP2OUT 输入信号　1100：PDM0CMPH 输入信号 1101：PDM0CMPL 输入信号　1110：PDM1CMPH 输入信号 1111：PDM1CMPL 输入信号 注意：没有显示的值作为保留。
[7:4] DCALCMPSEL	**数字比较器 A 低电平(DCAL)输入信号选择位，定义了 DCAL 的源信号(EPWM Digital Compare DCAL Select)** 当 TZ 信号用作故障信号时，作为正常输入来处理并且定义为高电平有效或低电平有效。 0000：TZ0 输入信号　0001：TZ1 输入信号　0010：TZ2 输入信号 0011：TZ3 输入信号　1000：CMP0OUT 输入信号　1001：CMP1OUT 输入信号 1010：CMP2OUT 输入信号　1100：PDM0CMPH 输入信号 1101：PDM0CMPL 输入信号　1110：PDM1CMPH 输入信号 1111：PDM1CMPL 输入信号 注意：没有显示的值作为保留。

[3:0] DCAHCMPSEL	**数字比较器 A 高电平（DCAH）输入信号选择位，定义 DCAH 输入的信号源（EPWM Digital Compare DCAH Select）** 当 TZ 信号用作故障信号时，作为正常输入来处理并且定义为高电平有效或低电平有效。 0000：TZ0 输入信号　0001：TZ1 输入信号　0010：TZ2 输入信号 0011：TZ3 输入信号　1000：CMP0OUT 输入信号　1001：CMP1OUT 输入信号 1010：CMP2OUT 输入信号　1100：PDM0CMPH 输入信号 1101：PDM0CMPL 输入信号　1110：PDM1CMPH 输入信号 1111：PDM1CMPL 输入信号 注意：没有显示的值作为保留。

21.5.2.26 EPWM 数字比较 A 控制寄存器（EPWM_DCACTL）

- **名称：EPWM Digital Compare A Control Register**
- **偏移地址：0x84**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:10	9	8	7:4
名称		AEVT2FRCSYNCSEL	AEVT2SRCSEL	
访问		R/W	R/W	
复位值		0x0	0x0	
位	3	2	1	0
名称	AEVT1SYNCE	AEVT1SOCE	AEVT1FRCSYNCSEL	AEVT1SRCSEL
访问	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0

字段	说明
[9] AEVT2FRCSYNCSEL	**DCAEVT2 强制同步信号选择控制位（EPWM Digital Compare DCAEVT2 Force Sync Select）** 0：选择同步信号源 1：选择非同步信号源
[8] AEVT2SRCSEL	**写 DCAEVT2 源信号选择控制位（EPWM Digital Compare DCAEVT2 Source Select）** 0：信号源为 DCAEVT2 1：信号源为 DCEVTFILT
[3] AEVT1SYNCE	**DCAEVT1 同步信号使能位（EPWM Digital Compare DCAEVT1 Sync Enable）** 0：不产生同步信号 1：产生同步信号
[2] AEVT1SOCE	**DCAEVT1 的 SOC 信号使能位（EPWM Digital Compare DCAEVT1 Soc Enable）** 0：不产生 SOC 信号 1:：产生 SOC 信号

[1] AEVT1FRCSYNCSEL	**DCAEVT1 强制同步信号选择控制位（EPWM Digital Compare DCAEVT1 Force Sync Select）** 0：选择同步信号源 1：选择非同步信号源
[0] AEVT1SRCSEL	**DCAEVT1 信号源选择位（EPWM Digital Compare DCAEVT1 Source Select）** 0：信号源为 DCAEVT1 1：信号源为 DCEVTFILT

21.5.2.27 EPWM 数字比较 B 控制寄存器（EPWM_DCBCTL）

- **名称：EPWM Digital Compare B Control Register**
- **偏移地址：0x88**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:10	9	8	7:4
名称		BEVT2FRCSYNCSEL	BEVT2SRCSEL	
访问		R/W	R/W	
复位值		0x0	0x0	
位	3	2	1	0
名称	BEVT1SYNCE	BEVT1SOCE	BEVT1FRCSYNCSEL	BEVT1SRCSEL
访问	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0

字段	说明
[9] BEVT2FRCSYNCSEL	**DCBEVT2 强制同步信号选择控制位（EPWM Digital Compare DCBEVT2 Force Sync Select）** 0：选择同步信号源 1：选择非同步信号源
[8] BEVT2SRCSEL	**写 DCBEVT2 源信号选择控制位（EPWM Digital Compare DCBEVT2 Source Select）** 0：信号源为 DCBEVT2 1：信号源为 DCEVTFILT
[3] BEVT1SYNCE	**DCBEVT1 同步信号使能位（EPWM Digital Compare DCBEVT1 Sync Enable）** 0：不产生同步信号 1：产生同步信号
[2] BEVT1SOCE	**DCBEVT1 的 SOC 信号使能位（EPWM Digital Compare DCBEVT1 Soc Enable）** 0：不产生 SOC 信号 1:：产生 SOC 信号
[1] BEVT1FRCSYNCSEL	**DCBEVT1 强制同步信号选择控制位（EPWM Digital Compare DCBEVT1 Force Sync Select）** 0：选择同步信号源 1：选择非同步信号源

[0] BEVT1SRCSEL	**DCBEVT1 信号源选择位（EPWM Digital Compare DCBEVT1 Source Select）** 0：源信号为 DCBEVT1 1：源信号为 DCEVTFILT

21.5.2.28 EPWM 数字比较滤波控制寄存器（EPWM_DCFCTL）

- **名称：EPWM Digital Compare Filter Control Register**
- **偏移地址：0x8C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:10	9	8	7:6
名称		READWINDOWCNT	READOFFSETCNT	
访问		R/WAC	R/WAC	
复位值		0x0	0x0	
位	5:4	3	2	1:0
名称	PULSESEL	BLANKINV	BLANKE	FILTSRCSEL
访问	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0

字段	说明
[9] READWINDOWCNT	**WINDOWCNT 读操作启动控制位（EPWM Digital Compare Read Window Counter）** 写 0：不启动 WINDOWCNT 的读操作 写 1：启动 WINDOWCNT 的读操作，当 WINDOWCNT 正在进行读操作，READWINDOWCNT 保持为 1；当 OFFSETCNT 完成读操作，此位自动清 0
[8] READOFFSETCNT	**OFFSETCNT 读操作启动控制位（EPWM Digital Compare Read Offset Counter）** 写 0：不启动 OFFSETCNT 的读操作 写 1：启动 OFFSETCNT 的读操作，当 OFFSETCNT 正在进行读操作，READOFFSETCNT 保持为 1；当 OFFSETCNT 完成读操作，此位自动清 0
[5:4] PULSESEL	**Blanking 和 Capture Alignment 的脉冲选择控制位（EPWM Digital Compare Pulse Select）** 00：时基计数器等于周期值（TBCTR = TBPRD） 01：时基计数器等于 0（TBCTR = 0x0000） 10：保留 11：保留
[3] BLANKINV	**Blanking Window 反转（EPWM Digital Compare Blank Invert）** 0：Blanking window 不反转 1：Blanking window 反转
[2] BLANKE	**Blanking Window 使能位（EPWM Digital Compare Blank Enable）** 0：不使能 Blanking window 1：使能 Blanking window

[1:0] FILTSRCSEL	**滤波块（Filter Block Signal）信号源选择位（EPWM Digital Compare Filter Source Select）** 00：信号源为 DCAEVT1　01：信号源为 DCAEVT2 10：信号源为 DCBEVT1　11：信号源为 DCBEVT2

21.5.2.29 EPWM 数字比较捕获控制寄存器（EPWM_DCCAPCTL）

- **名称：EPWM Digital Compare Capture Control Register**
- **偏移地址：0x90**
- **默认值：0x00000000**
- 返回寄存器列表
-

位	31:4	1	0
名称		SHDWMODE	CAPE
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[1] SHDWMODE	**时基计数器 capture 影子模式选择位（EPWM Digital Compare Capture Shadow Mode）** 0：使能影子模式。当 TBCTR = TBPRD 或 TBCTR =0 时（由 PULSELSEL（DCFCTL[5:4]）决定），DCCAP 激活寄存器从 DCCAP 影子寄存器处复制数据。CPU 读取 DCCAP 时返回的是 DCCAP 影子寄存器的内容。 1：使能激活模式，在此模式下，影子寄存器模式被关闭。CPU 读取 DCCAP 时返回的是 DCCAP 激活寄存器中的值。
[0] CAPE	**时基计数器 capture 功能使能位（EPWM Digital Compare Capture Enable）** 0：不使能时基计数器的 capture 功能 1：使能时基计数器的 capture 功能

21.5.2.30 EPWM 数字比较捕获寄存器（EPWM_DCCAP）

- **名称：EPWM Digital Compare Capture Register**
- **偏移地址：0x94**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		DCCAP
访问		R
复位值		0x0

字段	说明
[15:0] DCCAP	**数字比较时基计数器 capture 位（EPWM Digital Compare Capture）** 设置 CAPE（DCCAPCTL[0]）为 1，使能时基计数器 capture 功能。 如果使能时基计数器的 capture，在 dcevt filter 的输出出现上升沿时记录时基计数器的值。直到时基计数器值等于 0 或周期值时（在 PULSESEL（DCFCTL[5:4]）设置）才会产生下一个 capture。 在 SHDWMODE（DCCAPCTL[1]）位设置 DCCAP 的影子是否使能，默认 DCCAP 为影子模式。 如果 SHDWMODE（DCCAPCTL[1]）= 0，使能影子模式。在此模式下，当 TBCTR = TBPRD 或 TBCTR = 0 时（由 PULSESEL（DCFCTL[7:6]）设置），DCCAP 激活寄存器会将数据复制到 DCCAP 影子寄存器。CPU 读取 DCCAP 时将返回影子寄存器中的值。 如果 SHDWMODE（DCCAPCTL[1]）= 1，关闭影子模式。CPU 读取 DCCAP 时返回 DCCAP 激活寄存器中的值。 注意：DCCAP 激活寄存器和影子寄存器共享同一个内存映射地址。

21.5.2.31 EPWM 数字比较滤波偏移寄存器（EPWM_DCFOFFSET）

- 名称：EPWM Digital Compare Filter Offset Register
- 偏移地址：0x98
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		OFFSET
访问		R/W
复位值		0x0

字段	说明
[15:0] OFFSET	**Blanking Window 偏移量设置（EPWM Digital Compare Filter Offset）** 此位指定从 blanking window 引用到应用 blanking window 的时基计数器时钟的周期个数。blanking window 引用是由 PULSESEL（DCFCTL[5:4]）位定义的时基计数器等于 0 或周期值。 该偏移寄存器是影子寄存器并且激活寄存器被加载在由 PULSESEL（DCFCTL[5:4]）定义的参考点处。当激活寄存器被加载时，偏移计数器也被初始化并且开始向下计数。当计数器满时，应用 blanking window。如果当前 blanking window 是激活模式，则重启 blanking window 计数器。

21.5.2.32 EPWM 数字比较滤波偏移计数寄存器（EPWM_DCFOFFSETCNT）

- 名称：EPWM Digital Compare Filter Offset Counter Register
- 偏移地址：0x9C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		OFFSETCNT
访问		R
复位值		0x0

字段	说明
[15:0] OFFSETCNT	**Blanking Window 偏移量设置（EPWM Digital Compare Filter Offset Counter）** 此位指定从 blanking window 引用到应用 blanking window 的时基计数器时钟的周期个数。blanking window 引用是由 PULSESEL（DCFCTL[5:4]）位定义的时基计数器等于 0 或周期值。 该偏移寄存器是影子寄存器并且激活寄存器被加载在由 PULSESEL（DCFCTL[5:4]）定义的参考点处。当激活寄存器被加载时，偏移计数器也被初始化并且开始向下计数。当计数器满时，应用 blanking window。如果当前 blanking window 是激活模式，则重启 blanking window 计数器。

21.5.2.33 EPWM 数字比较滤波窗口寄存器（EPWM_DCFWINDOW）

- **名称：EPWM Digital Compare Filter Window Register**
- **偏移地址：0xA0**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:12	11:0
名称		WINDOW
访问		R/W
复位值		0x0

字段	说明
[11:0] WINDOW	**Blanking Window 的宽度设置（EPWM Digital Compare Filter Window）** 0x000：不产生 blanking window 0x001-0xFFF：指定在时基计数器周期内的 blanking window 的宽度。当偏移计数器满时启动 blanking window。接着，window 计数器加载完成并开始向下计数。如果当前的 blanking window 是 active 的并且偏移计数器满，重启 blanking window 计数器。 Blanking window 可以通过一个 PWM 周期边界，也可以大于周期值，当 offset 值为 0，window 值需要配为 period+1，才可以完全 blanking，若 window 值等于 period，会有 1cycle 的脉冲没有被过滤。如果当前 blanking window 是激活的，重启 blanking window。

21.5.2.34 EPWM 数字比较滤波窗口计数寄存器（EPWM_DCFWINDOWCNT）

- 名称：EPWM Digital Compare Filter Window Counter Register
- 偏移地址：0xA4
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:0
名称		WINDOWCNT
访问		R
复位值		0x0

字段	说明
[11:0] WINDOWCNT	**Blanking Window 的宽度设置（EPWM Digital Compare Filter Window Counter）** 0x000：不产生 blanking window 0x001-0xFFF：指定在时基计数器周期内的 blanking window 的宽度。当偏移计数器满时启动 blanking window。接着，window 计数器加载完成并开始向下计数。如果当前的 blanking window 是 active 的并且偏移计数器满，重启 blanking window 计数器。 Blanking window 可以通过一个 PWM 周期边界，也可以大于周期值，当 offset 值为 0，window 值需要配为 period+1，才可以完全 blanking，若 window 值等于 period，会有 1cycle 的脉冲没有被过滤。如果当前 blanking window 是激活的，重启 blanking window。

21.5.2.35 EPWM 事件触发寄存器（EPWM_ETSEL）

- 名称：EPWM Event Trigger Select Register
- 偏移地址：0xB0
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15	14:12	11	10:8	7
名称		SOCBEN	SOCBSELL	SOCAEN	SOCASELL	SOCBSELH
访问		R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0
位	6	5	4	3	2:0	
名称	SOCASELH		INTSELH	INTEN	INTSELL	
访问	R/W		R/W	R/W	R/W	
复位值	0x0		0x0	0x0	0x0	

字段	说明
[15] SOCBEN	**ADC 启动脉冲 B（EPWMxSOCB）的使能位（EPWM Event Trigger Soc B Enable）** 0：不使能 EPWMxSOCB 1：使能 EPWMxSOCB 脉冲
[14:12] SOCBSELL	**EPWMxSOCB 选择控制位最低位（EPWM Event Trigger Soc B Select Low）** SOCBSEL 决定 EPWMxSOCB 脉冲信号产生的时间。 0000：使能 DCBEVT1 启动的事件（DCBEVT1 SOC） 0001：使能时基计数器等于 0 的事件（TBCTR = 0x0000） 0010：使能时基计数器等于周期值的事件（TBCTR = TBPRD） 0011：使能时基计数器等于 0 或等于周期值的事件（TBCTR = 0x0000 和 TBCTR = TBPRD） 0100：当定时器增加时，使能时基计数器等于 CMPA 的事件 0101：当定时器递减时，使能时基计数器等于 CMPA 的事件 0110：当定时器递增时，使能时基计数器等于 CMPB 的事件 0111：当定时器递减时，使能时基计数器等于 CMPB 的事件 1000：当定时器递增时，使能时基计数器等于 CMPC 的事件 1001：当定时器递减时，使能时基计数器等于 CMPC 的事件
[11] SOCAEN	**ADC 启动脉冲 A（EPWMxSOCA）的使能位（EPWM Event Trigger Soc A Enable）** 0：不使能 EPWMxSOCA 1：使能 EPWMxSOCA

[10:8] SOCASELL	**EPWMxSOCA 选择控制位最低位（EPWM Event Trigger Soc A Select Low）** SOCASEL 决定 EPWMxSOCA 脉冲产生的时间。 0000：使能 DCAEVT1 启动的事件（DCAEVT1 SOC） 0001：使能时基计数器等于 0 的事件（TBCTR = 0x0000） 0010：使能时基计数器等于周期值的事件（TBCTR = TBPRD） 0011：使能时基计数器等于 0 或等于周期值的事件（TBCTR = 0x0000 和 TBCTR = TBPRD） 0100：当定时器递增时，使能时基计数器等于 CMPA 的事件 0101：当定时器递减时，使能时基计数器等于 CMPA 的事件 0110：当定时器递增时，使能时基计数器等于 CMPB 的事件 0111：当定时器递减时，使能时基计数器等于 CMPB 的事件 1000：当定时器递增时，使能时基计数器等于 CMPC 的事件 1001：当定时器递减时，使能时基计数器等于 CMPC 的事件
[7] SOCBSELH	**SOCBSEL 寄存器最高位（EPWM Event Trigger Soc B Select High）** SOCBSEL={SOCBSELH，SOCBSELL}，由此位域与最低位组合而成
[6] SOCASELH	**SOCASEL 寄存器最高位（EPWM Event Trigger Soc A Select High）** SOCASEL={SOCASELH，SOCASELL}，由此位域与最低位组合而成
[4] INTSELH	**INTSEL 寄存器最高位（EPWM Event Trigger Interrupt Select High）** INTSEL={INTSELH，INTSELL}，由此位域与最低位组合而成
[3] INTEN	**EPWM 中断（INT）使能位（EPWM Event Trigger Interrupt Enable）** 0：不使能 INT 1：使能 INT
[2:0] INTSELL	**EPWM 中断(INT) 选择控制位最低位（EPWM Event Trigger Interrupt Select Low）** 0000：保留 0001：使能时基计数器等于 0 的事件（TBCTR = 0x0000） 0010：使能时基计数器等于周期值的事件（TBCTR = TBPRD） 0011：使能时基计数器等于 0 或等于周期值的事件（TBCTR = 0x0000 和 TBCTR = TBPRD） 0100：当定时器递增时，使能时基计数器等于 CMPA 的事件 0101：当定时器递减时，使能时基计数器等于 CMPA 的事件 0110：当定时器递增时，使能时基计数器等于 CMPB 的事件 0111：当定时器递减时，使能时基计数器等于 CMPB 的事件 1000：当定时器递增时，使能时基计数器等于 CMPC 的事件 1001：当定时器递减时，使能时基计数器等于 CMPC 的事件

21.5.2.36 EPWM 事件触发预分频寄存器（EPWM_ETPS）

- 名称：EPWM Event Trigger Prescale Register
- 偏移地址：0xB4
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:14	13:12	11:10
名称		SOCBCNT	SOCBPRD	SOCACNT
访问		R	R/W	R
复位值		0x0	0x0	0x0
位	9:8	7:4	3:2	1:0
名称	SOCAPRD		INTCNT	INTPRD
访问	R/W		R	R/W
复位值	0x0		0x0	0x0

字段	说明
[15:14] SOCBCNT	**EPWM 的 ADC 启动事件 B（EPWMxSOCB）的计数寄存器（EPWM Event Trigger Soc B Counter）** SOCBCNT 位表示发生了多少个由 SOCBSEL 选择的事件。 00：没有事件发生　　01：发生 1 个事件 10：发生 2 个事件　　11：发生 3 个事件
[13:12] SOCBPRD	**EPWM 的 ADC 启动事件 B（EPWMxSOCB）的周期选择位（EPWM Event Trigger Soc B Period）** SOCBPRD 决定了在产生一个 EPWMxSOCB 脉冲之前需要发生了多少个在 SOCBSEL 选择的事件。为了产生脉冲，SOCBEN 要设为 1。甚至当 SOCB 等于 1 时，SOCB 脉冲也会产生。一旦产生 SOCB 脉冲，SOCBCNT 就会自动清 0。 00：不使能 SOCB 事件计数器，没有 EPWMxSOCB 脉冲产生 01：在第一个事件内产生 EPWMxSOCB 脉冲：SOCBCNT = 01 10：在第二个事件内产生 EPWMxSOCB 脉冲：SOCBCNT = 10 11：在第三个事件内产生 EPWMxSOCB 脉冲：SOCBCNT = 11
[11:10] SOCACNT	**EPWM 的 ADC 启动事件 A（EPWMxSOCA）计数器寄存器（EPWM Event Trigger Soc A Counter）** 本位表示发生了多少个在 SOCASEL 中选择的事件。 00：没有事件发生　　01：发生 1 个事件 10：发生 2 个事件　　11：发生 3 个事件
[9:8] SOCAPRD	**EPWM 的 ADC 启动事件 A（EPWMxSOCA）的周期选择位（EPWM Event Trigger Soc A Period）** SOCAPRD 决定了在产生一个 EPWMxSOCA 脉冲之前需要发生了多少个在 SOCASEL 选择的事件。为了产生脉冲，SOCAEN 要设为 1。甚至当 SOCA 等于 1 时，SOCB 脉冲也会产生。一旦产生 SOCA 脉冲，SOCACNT 就会自动清 0。 00：不使能 SOCA 事件计数器。没有 EPWMxSOCA 脉冲信号产生 01：在第一个事件中产生 EPWMxSOCA 脉冲：SOCACNT = 01 10：在第二个事件中产生 EPWMxSOCA 脉冲：SOCACNT = 10 11：在第三个事件中产生 EPWMxSOCA 脉冲：SOCACNT = 11

[3:2] INTCNT	**EPWM 中断事件（INT）计数器寄存器（EPWM Event Trigger Interrupt Counter）** 本位指示发生了多少 个 INTSEL 中选择的事件,当产生一个中断时，该位会自动清 0。如果没有中断发生或者当 ETFLAG[0] = 1 时，如果达到周期值计数器会停止计数。 00：没有事件发生　　01：发生 1 个事件 10：发生 2 个事件　　11：发生 3 个事件
[1:0] INTPRD	**EPWM 中断周期选择位（EPWM Event Trigger Interrupt Period）** 该位决定了在产生中断之前，需要发生多少个 INTSEL 选择的事件。为了成功打开中断，必须打开中断使能位 INT。如果当前的中断产生了中断标志 INT = 1，那么等到清除中断标志位 CLRINT 置 1 后才会再次产生中断。当一个中断正在执行时，允许挂起另一个中断。一旦产生了中断，INTCNT 会自动清 0。 如果 INTPRD 不为 0 并清除中断标志位 INT，则向 INTPRD 写入一个与当前计数器相同的值可以触发中断。 向 INTPRD 写入一个小于当前计数器的值将进入一个未定义的状态。 如果同时发生计数器事件和向 INTPRD 写入一个 0 或非 0 的值时，该计数器向上计数。 00：不打开中断事件计数器。没有中断发生并且忽略 ETFRC 01：在第一个事件中产生一个中断，INTCNT = 01 10：在第二个事件中产生一个中断，INTCNT = 10 11：在第三个事件中产生一个中断，INTCNT = 11

21.5.2.37 EPWM 事件触发标志寄存器（EPWM_ETFLAG）

- 名称：EPWM Event Trigger Flag Register
- 偏移地址：0xB8
- 默认值：0x00000000
- 返回寄存器列表

位	31:4	3	2	1	0
名称		SOCB	SOCA		INT
访问		R	R		R
复位值		0x0	0x0		0x0

字段	说明
[3] SOCB	**EPWM 的 ADC 启动 B（EPWMxSOCB）的状态标志位（EPWM Event Trigger Soc B Flag）** 0：没有 EPWMxSOCB 发生 1：指示在 EPWMxSOCB 发生一个启动转换的脉冲。即使标志位 SOCB 被设为 1，EPWMxSOCB 也会持续产生输出
[2] SOCA	**EPWM 的 ADC 启动 A（EPWMxSOCA）的状态标志位（EPWM Event Trigger Soc A Flag）** 与 INT 不同，即使标志位 SOCA 被设为 1，EPWMxSOCA 也会持续产生输出 0：没有 EPWMxSOCA 发生 1：指示在 EPWMxSOCA 发生一个启动转换的脉冲。即使标志位 SOCA 被设为 1，EPWMxSOCA 也会持续产生输出
[0] INT	**EPWM 的中断状态标志位（INT）（EPWM Event Trigger Interrupt Flag）** 0：没有中断发生 1：有一个 EPWM 的中断发生 注意：只有当此标志位清 0 后才会产生下一个标志位。

21.5.2.38 EPWM 事件触发清除寄存器（EPWM_ETCLR）

- 名称：EPWM Event Trigger Clear Register
- 偏移地址：0xBC
- 默认值：0x00000000
- 返回寄存器列表

位	31:4	3	2	1	0
名称		SOCB	SOCA		INT
访问		W	W		W
复位值		0x0	0x0		0x0

字段	说明
[3] SOCB	**EPWM 的 ADC 启动 B（EPWMxSOCB）的标志清除控制位（EPWM Event Trigger Soc B Clear）** 0：无任何效果，读返回 0 1：SOCB 标志位清 0
[2] SOCA	**EPWM 的 ADC 启动 A（EPWMxSOCA）的标志清除控制位（EPWM Event Trigger Soc A Clear）** 0：无任何效果，读返回 0 1：清除 SOCA 标志位
[0] INT	**EPWM 中断标志位（INT）清除控制位（EPWM Event Trigger Interrupt Clear）** 0：无任何效果，读返回 0 1：清除中断标志位 INT

21.5.2.39 EPWM 事件触发强制寄存器（EPWM_ETFRC）

- 名称：EPWM Event Trigger Force Register
- 偏移地址：0xC0
- 默认值：0x00000000
- 返回寄存器列表

位	31:4	3	2	1	0
名称		SOCB	SOCA		INT
访问		W	W		W
复位值		0x0	0x0		0x0

字段	说明
[3] SOCB	**SOCB 强制控制位（EPWM Event Trigger Soc B Force）** 如果 SOCB 使能，就会产生一个 SOCB 脉冲，则必须设置 SOCB 为 1 0：没有任何效果，读返回 0 1：在 EPWMxSOCB 处产生脉冲并设置 SOCB，本位通常用于测试

[2] SOCA	**SOCA 强制控制位（EPWM Event Trigger Soc A Force）** 如果 SOCA 使能，就会产生一个 SOCA 脉冲，则必须设置 SOCA 为 1。 0：没有任何效果，读返回 0 1：在 EPWMxSOCA 处产生脉冲并设置 SOCA，本位通常用于测试
[0] INT	**INT 强制控制位（EPWM Event Trigger Interrupt Force）** 如果中断使能，将产生中断。那么必须设置中断标志 INT 为 1。 0：无任何效果，读返回 0 1：在 EPWMxINT 产生一个中断并设置中断标志位 INT，该位通常用于测试

22 脉冲密度调制（PDM）

22.1 简介

本章介绍 sigma delta 滤波器模块（PDM）。PDM 是一个双通道数字滤波器，专门用于电机控制应用中的电流测量和旋转变压器位置解码。每个输入通道可以接收一个独立 delta-sigma（ΔΣ）调制器比特流。比特流由两个可单独编程的数字抽取滤波器处理。该滤波器组包括一个快速比较器（辅助滤波器），用于立即数字阈值比较，以进行过流和欠流监视以及过零检测。

22.2 结构框图

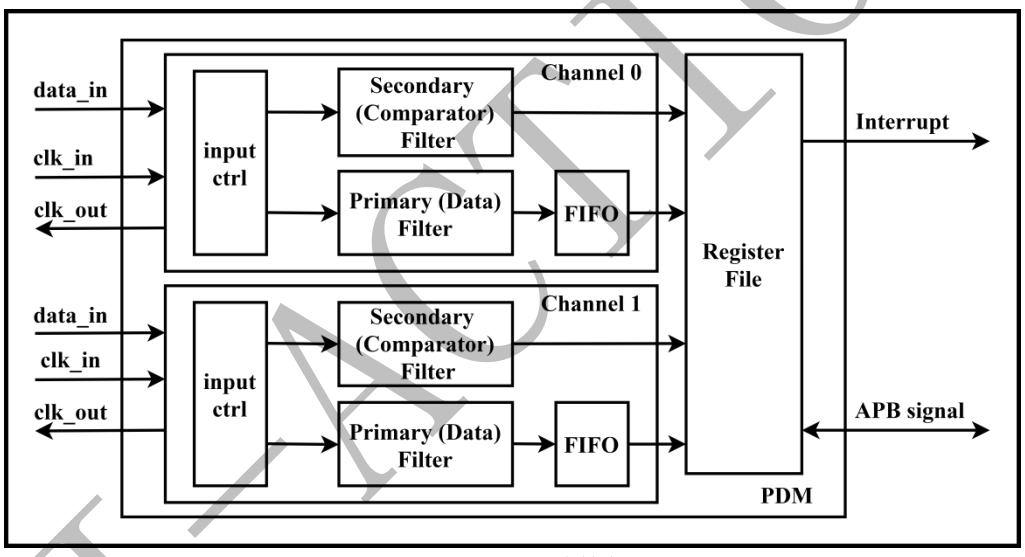


图 22-1 PDM 结构框图

22.3 主要特性

PDM 的主要功能如下：

- 支持 SPI 主从模式操作
- 支持 DMA 模式
- 支持 32k 种不同的波特率配置
- 支持双通道 PDM
- 每个 PDM 通道都有一个可配置的辅助滤波器（比较器）单元：
  - 提供四种不同的滤波器类型选择（$Sinc^1$ / $Sinc^2$ / Sincfast / $Sinc^3$）选项
  - 能够检测超值情况和欠值情况情况
  - 比较器滤波器单元（COSR）的 OSR 值，可在 1 到 32 之间编程
- 每个 PDM 通道都有一个可配置主滤波器（数据滤波器）单元：
  - 提供四种不同的滤波器类型选择（$Sinc^1$ / $Sinc^2$ / Sincfast / $Sinc^3$）选项
  - 数据过滤器单元（DOSR）的 OSR 值，可在 1 到 256 之间进行编程
  - 能够启用或禁用（或同时禁用）单个过滤器模块
- 主滤波器可以有 24 位和 16 位输出选择

可以将 PWMx.SOCA / SOCB 配置为按每个数据过滤器通道用作 SDSYNC 源。

22.4 功能描述

22.4.1 SPI 控制单元

PDM 和外界ΣΔ调制器的数据交互是通过 SPI 协议来实现的。端口信号如下表所示：

表 22-1 PDM 数据交互端口信息

名称	信号类型	描述
CLKOUT[1:0]	SPI 时钟输出	SPI 时钟输出，用于为外部ΣΔ调制器提供时钟信号
CLKIN[1:0]	SPI 时钟输入	SPI 时钟输入，从外部ΣΔ调制器获取的时钟信号
DATAIN[1:0]	SPI 数据输入	SPI 数据输入，从外部ΣΔ调制器获取的数据信号

PDM 可作为主机，也可作为从机，取决于 PDM_CTRL[16]的 MASTER_EN 寄存器位，向该位写 1 代表 PDM 作为主机，否则作为从机。

作为主机时，PDM 可以通过配置 PDM_CTRL[15:0]的 BAUD 寄存器来配置输入时钟的波特率，波特率公式如下所示：

$$CLKOUT = \frac{F_{SYSCLK}}{BAUD \times 2}$$

由公式可知，时钟输出的 CLKOUT 与系统总线时钟的频率有关。同时，可以配置 PDM_CTRL[17]的 SCPOL 设置输出 CLKOUT 的极性。

作为从机时，通过配置 PDM_CTRL[18]的 SAMPMODE 来决定当前 PDM 是通过上升沿采样还是下降沿采样。
因为输入时钟的不确定性，若是丢失沿的话很可能导致后续数据的失效，因此可以通过配置 PDM_CLKTO 寄存器去限制输入时钟 CLKIN 的最大翻转时长，以系统时钟为计数，如果翻转不及时，则会产生中断。

22.4.2 SINC 滤波器

PDM 包含 $Sinc^x$ 类型数字滤波器实现。此 $Sinc^x$ 滤波器执行输入数字数据流滤波，这会导致减小输出数据速率（抽取）以及增大输出数据分辨率。可对 $Sinc^x$ 数字滤波器进行配置，以达到所需输出数据速率和所需输出数据分辨率。可配置参数有：

- 滤波器阶数/类型：
  - FastSinc
  - $Sinc^1$
  - $Sinc^2$
  - $Sinc^3$
- 滤波器过采样/抽取率：最大取值 256

滤波器支持以下传递函数（H 域中的冲激响应）：

- $Sinc^x$ 滤波器类型：$H(z) = (\frac{1-z^{-OSR}}{1-z^{-1}})^x$

- FastSinc 滤波器类型： $H(z) = (\frac{1-z^{-OSR}}{1-z^{-1}})^2 \times (1 + z^{-2\times OSR})$

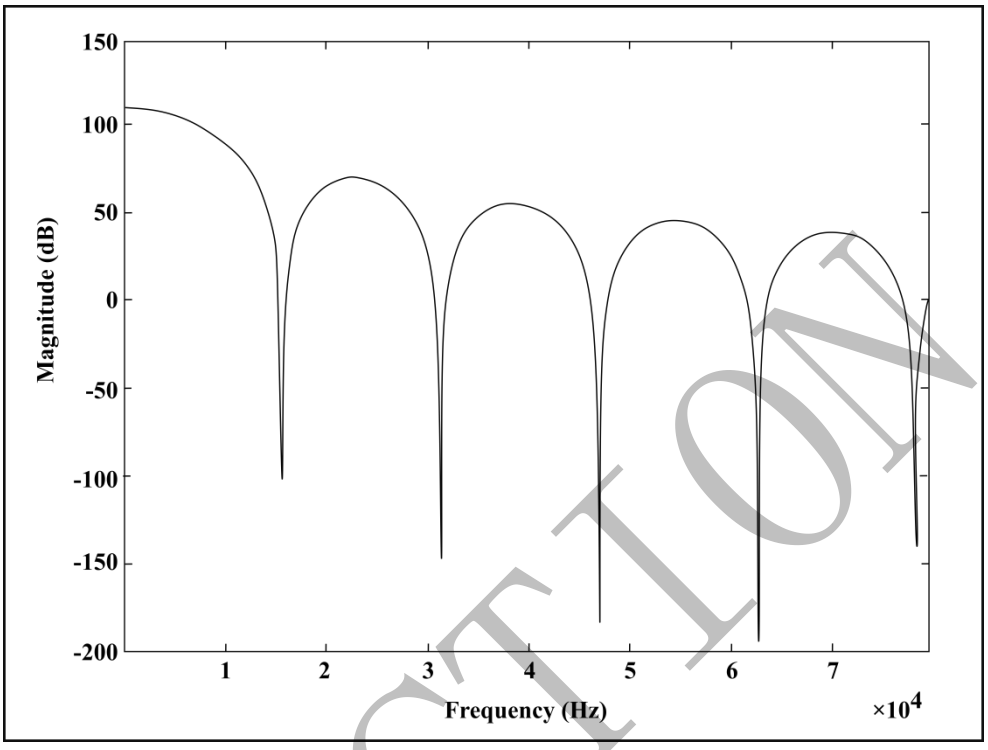


图 22-2　Sinc³ 的滤波器响应

表 22-2　滤波器最大输出分辨率（来自滤波器输出的峰值数据值），基于某些 OSR 值

OSR	Sinc¹	Sinc²	Sinc³	FastSinc
x	+/- x	+/- x^2	+/- x^3	+/- $2x^2$
4	+/- 4	+/- 16	+/- 64	+/- 32
8	+/- 8	+/- 64	+/- 512	+/- 128
32	+/- 32	+/- 1024	+/- 32768	+/- 2048
64	+/- 64	+/- 4096	+/- 262144	+/- 8192
128	+/- 128	+/- 16384	+/- 2097152	+/- 32768
256	+/- 256	+/- 65536	+/- 16777216	+/- 131072

22.4.3　主滤波器

数据过滤器是可配置的 Sinc 过滤器，它支持以下过滤器类型：Sinc¹，Sinc²，Sinc³ 和 SincFast。数据过滤器 OSR（DOSR）设置可以配置为 1 到 256，并且与比较器过滤器无关。数据滤波器的有效分辨率取决于数据滤波器类型，DOSR 和 sigma-delta 调制器频率。默认情况下，数据过滤器是不启动的。数据过滤器以 24 位或者 16 位有符号整数格式输出。该滤波器单元将低输入信号转换为“-1”，将高输入信号转换为“1”。结果计算为数据滤波器的输出给出了正值和负值。

22.4.3.1 24bit 或者 16bit 输出

数据过滤器输出可以 24 位或 16 位格式表示。

24 位数据过滤器表示形式：

- PDM_DFCR[13]的 DFDR=1 时，数据过滤器输出以 24 位格式表示。在这种配置下，写入 PDM_DFCR[19:16]的 SHIFT 位对数据滤波器的输出没有任何影响。

16 位数据过滤器表示形式：

- 默认情况下，数据过滤器输出以 16 位格式表示
- PDM_DFCR[13]的 DFDR=0 时，数据过滤器输出以 16 位格式表示。但是，用户需要配置相应的 PDM_DFCR[19:16]的 SHIFT 位，以控制将 24 位字的哪个 16 位部分作为输出。

例如，对于以下数据过滤器配置：

- 过滤器类型= $Sinc^3$
- OSR = 128
- PDM_DFCR[13]的 DFDR=0

上述配置对于带符号输出值的数据过滤器的范围可以在-2097152 到 2097152 之间。但是，16 位带符号输出仅支持从-32768 到 32767 的值。因此，需要将移位控制位（PDM_DFCR[19:16]的 SHIFT）配置为 7，以正确表示 16 位格式的数据过滤器输出。下表显示了不同 OSR 和滤波器类型的移位控制位的最小推荐配置设置。

表 22-3 SHIFT 位的最小推荐配置

OSR	$Sinc^1$	$Sinc^2$	FastSinc	$Sinc^3$
1~31	0	0	0	0
32~40	0	0	0	1
41~50	0	0	0	2
51~63	0	0	0	3
64~80	0	0	0	4
81~101	0	0	0	5
102~127	0	0	0	6
128~161	0	0	1	7
162~181	0	0	1	8
182~203	0	1	2	8
204~255	0	1	2	9
256	0	2	3	10

若 shift 配置得比表格中的数值小，可能会导致 16 位输出数据有误。

22.4.3.2 FIFO

每个主（数据）过滤器通道都有一个深度为 8 的 24 位 FIFO。

中断

- FIOINTR 中断：当 FIFO 已满且当前还有一个写入操作时就会触发 FIOINTR 中断，该中断对 PDM_ISR[4]写 1 清 0
- FIFUINTR 中断：当 FIFO 里面数据的数量大于或者等于预设值（PDM_FCSR[2:0]的 PDMFLT）时就会触发 FIFINTR，该中断当 FIFO 中数据数量小于预设值时自动清 0

等待同步功能

FIFO 等待同步功能可用于忽略数据过滤器中的数据写入事件，直到触发 SDSYNC（来自 PWM）事件为止。

当 PDM_SDSYNC[6]的 WTSYNCEN=0 时，数据滤波器中的数据将会一直写入 FIFO 中，直到 FIFO 被填满。

当 PDM_SDSYNC[6]的 WTSYNCEN=1 时，那么只有在接收到 SDSYNC 事件后，FIFO 才会接收数据滤波器的数据写入事件，直到 FIFO 被填满或者 FIFINTR 中断起来。

当 PDM_SDSYNC[9]的 FFSYNCCLREN=1 时，接收到 SDSYNC 事件后会把 FIFO 中的数据清除。

22.4.3.3 SDSYNC 事件

主（数据）滤波器可以相对于 PWM 事件（称为 SDSYNC 事件）进行同步。来自 PWM 模块的 SDSYNC 信号会复位 DOSR 计数器。默认情况下，此功能是禁用的，可以通过设置 PDM_DFCR [14]的 SDSYNCEN=1 来启用。每个主滤波器都可以与任何可用的 PWMx 同步。PDM_SDSYNC[5:0]的 SDSYNCSEL 位允许用户配置哪个 PWM 信号向主滤波器提供 SDSYNC 脉冲，如下表所示：

表 22-4 SDSYNCSEL 配置对应的 PWM 的信号表

	SDSYNCSEL[1:0]		
SDSYNCSEL[5:2]	0	1	2~3
0	PWM0.SOCA	PWM0.SOCB	Reserved
1	PWM1.SOCA	PWM1.SOCB	Reserved
2	PWM2.SOCA	PWM2.SOCB	Reserved
3	PWM3.SOCA	PWM3.SOCB	Reserved
4	PWM4.SOCA	PWM4.SOCB	Reserved
5	PWM5.SOCA	PWM5.SOCB	Reserved
6	PWM6.SOCA	PWM6.SOCB	Reserved
7~15	Reserved	Reserved	Reserved

由于 Sinc 滤波器的固有体系结构（Sinc¹，Sinc²，Sinc³，SincFast），根据滤波器类型的不同，前几个样本将是错误的。表 22-5 显示在以下情况下不正确的样本数：

- 首次启用/配置 Sinc 过滤器时。
- 在操作过程中禁用/重新启用或重新配置了 Sinc 过滤器时。
- 数据过滤器从 PWM 接收到 SDSYNC 事件时。

表 22-5　启动并配置滤波器后不正确的样本数

滤波器类型	启用并配置过滤器后不正确的样本数
Sinc¹	没有错误的样本
Sinc²	第一个样本是错误的
Sinc³	前两个样本是错误的
FastSinc	前两个样本是错误的

22.4.4 比较滤波器

大多数控制系统要求在电流或电压超出范围时通过使 PWM 跳闸来保护系统。 次级（比较器）滤波器的主要目的是允许用户以快速的建立时间监视输入条件。这使用户可以使 PWM 跳闸，以保护系统免受潜在损害。

比较器滤波器是可配置的 Sinc 滤波器，它支持以下滤波器类型：Sinc¹，Sinc²，Sinc³ 和 SincFast。比较器 OSR（COSR）设置可以配置为 1 到 32，并且与数据过滤器无关。比较器滤波器的有效分辨率取决于比较器滤波器的类型，COSR 和 sigma-delta 调制器频率。默认情况下，比较器滤波器被禁用。比较器滤波器的输出以 16 位无符号格式表示。该滤波器单元将低输入信号转换为“0”，将高输入信号转换为“1”。计算结果仅给出比较器滤波器输出的正值。

表 22-6. 比较器滤波器可以使用不同的 OSR 存储的不同的满量程值

OSR	Sinc¹	Sinc²	Sinc³	FastSinc
x	0~x	0~x^2	0~x^3	0~$2x^2$
4	0~4	0~16	0~64	0~32
8	0~8	0~64	0~512	0~128
32	0~32	0~1024	0~32768	0~2048

22.4.4.1 高阈值比较器

- 高阈值比较器可用于检测超值情况。
- 当比较器数据>=高阈值寄存器（PDM_CMPH[15:0]的 HLT）时，将产生高阈值事件。将会产生 CPU 中断

22.4.4.2 低阈值比较器

- 低阈值比较器可用于检测欠值情况。
- 当比较器数据<=下阈值寄存器（PDM_CMPL[15:0]的 LLT）时，将产生一个下阈值事件。将会产生 CPU 中断

22.5 寄存器描述

22.5.1 寄存器列表

PDM0 基地址：**0x4003 2000**
PDM1 基地址：**0x4003 2100**

偏移	实例地址	名称	默认值	描述
0x00	0x40032000	PDM_ENABLE	0x00000000	PDM 使能寄存器
0x04	0x40032004	PDM_CTRL	0x00000000	PDM 控制寄存器
0x08	0x40032008	PDM_DFCR	0x00000000	PDM 主滤波器控制寄存器
0x0C	0x4003200C	PDM_CFCR	0x00000000	PDM 比较滤波器控制寄存器
0x10	0x40032010	PDM_FCSR	0x00000000	PDM FIFO 控制寄存器
0x14	0x40032014	PDM_IER	0x00000000	PDM 中断使能寄存器
0x18	0x40032018	PDM_ISR	0x00000000	PDM 中断状态寄存器
0x1C	0x4003201C	PDM_DDAT	0x00000000	PDM 主滤波器数据寄存器
0x20	0x40032020	PDM_CDAT	0x00000000	PDM 比较滤波器数据寄存器
0x24	0x40032024	PDM_FDAT	0x00000000	PDM FIFO 数据寄存器
0x28	0x40032028	PDM_CMPH	0x00007FFF	PDM 高阈值比较寄存器
0x2C	0x4003202C	PDM_CMPL	0x00000000	PDM 低阈值比较寄存器
0x30	0x40032030	PDM_CLKTO	0x00003FFF	PDM 时钟翻转超时寄存器
0x34	0x40032034	PDM_SDSYNC	0x00000400	PDM 同步控制寄存器

【说明】上表中，x=0，1。

22.5.2 寄存器描述

22.5.2.1 PDM 使能寄存器（PDM_ENABLE）

- 名称：PDM Enable Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:1	0
名称		ENABLE
访问		R/W
复位值		0x0

字段	说明
[0]	**PDM 使能位（PDM Enable）**
ENABLE	0：PDM 不使能
	1：PDM 使能

22.5.2.2 PDM 控制寄存器（PDM_CTRL）

- 名称：PDM Control Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31	30:28	27:20	19	18	17	16	15:0
名称		RXDLY		DMAEN	SAMPMODE	SCPOL	MSTEN	BAUD
访问		R/W		R/W	R/W	R/W	R/W	R/W
复位值		0x0		0x0	0x0	0x0	0x0	0x0

字段	说明
[30:28]	**SPI 接收采样延迟（SPI RX Delay）**
RXDLY	PDM 作为主机 RX 时设置的 delay chain，用于延时输入数据的采样时钟，延时单位为 1 个系统时钟周期。
[19]	**DMA 使能位（DMA Enable）**
DMAEN	0：不使能
	1：使能
[18]	**从机时采样模式选择（Slave Sample Mode）**
SAMPMODE	0：时钟上升沿采样
	1：时钟下降沿采样

[17] SCPOL	**串行时钟极性（Serial Clock Polarity）** 0：串行时钟的无效状态为低 1：串行时钟的无效状态为高
[16] MSTEN	**PDM 主机使能位（PDM Master Enable）** 0：PDM 作为从机 1：PDM 作为主机
[15:0] BAUD	**波特率配置（Baud Rate Configuration）** 为 0 时不向外发送时钟；输出时钟的频率如下 Foutclk = Fpclk / (baudr*2)

22.5.2.3 PDM 主滤波器控制寄存器（PDM_DFCR）

- **名称：PDM Data Filter Control Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:20	19:16	15	14	13	12	11:9	8	7:0
名称		SHIFT		SDSYNCEN	DFDR	DFBYPASS	DFSST	DFEN	DOSR
访问		R/W		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[19:16] SHIFT	**移位控制选择位（Shift Control Select）** 最大值为 13，超过 13 的值均以 13 为准。只在 DFDR=0 时有效。 0：取计算结果的[15:0]作为输出。　　1：取计算结果的[16:1]作为输出。 2：取计算结果的[17:2]作为输出。 …… 13：取计算结果的[28:13]作为输出。　14~15：Reserved
[14] SDSYNCEN	**SDSYNC 信号使能位（SDSYNC Enable）** PDM 是否接收 PWM 的 SDSYNC 作为同步信号。 0：不接收 1：接收
[13] DFDR	**主滤波器输出数据位宽选择（Data Filter Output Data Length）** 0：16bit 输出 1：24bit 输出
[12] DFBYPASS	**主滤波器 bypass 使能（Data Filter Bypass Enable）** 使能时，滤波器输出会直接等于滤波器输入。 0：bypass 不使能 1：bypass 使能

[11:9] DFSST	**主滤波器类型（Data Filter Structure）** 000：FastSinc 001：Sinc^1 010：Sinc^2 011：Sinc^3
[8] DFEN	**主滤波器使能位（Data Filter Enable）** 0：不使能 1：使能
[7:0] DOSR	**主滤波器过采样值（Data filter Oversampling ratio）** 配置的真实值会比写入的值+1。例如需要跑过采样 256，那么需要向该寄存器写入 255。

22.5.2.4 PDM 比较滤波器控制寄存器（PDM_CFCR）

- **名称：PDM Comparator Filter Control Register**
- **偏移地址：0x0C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:11	10	9	8:6	5	4:0
名称		CMPSEL	CFBYPASS	CFSST	CFEN	COSR
访问		R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[10] CMPSEL	**比较滤波器输出选择（Comparator Filter Output Select）** 比较滤波器 CMPL 和 CMPH 输出选择，CMPL 指的是比较滤波器输出低于预设值(PDM_CMPL)，CMPH 指的是比较滤波器输出高于预设值(PDM_CMPH) 0：CMPL 和 CMPH 分别输出 1：CMPL 和 CMPH 或起来输出
[9] CFBYPASS	**比较滤波器 bypass 使能位（Comparator Filter Bypass Enable）** 使能时，滤波器输出会直接等于滤波器输入。 0：bypass 不使能 1：bypass 使能
[8:6] CFSST	**比较滤波器类型（Comparator Filter Structure）** 000：FastSinc 001：Sinc^1 010：Sinc^2 011：Sinc^3
[5] CFEN	**比较滤波器使能位（Comparator Filter Enable）** 0：不使能 1：使能

[4:0] COSR	**比较滤波器过采样值（Comparator filter Oversampling ratio）** 配置的真实值会比写入的值+1。例如需要跑过采样 32，那么需要向该寄存器写入 31。

22.5.2.5 PDM FIFO 控制寄存器（PDM_FCSR）

- **名称：PDM FIFO Control Status Register**
- **偏移地址：0x10**
- **默认值：0x00010000**
- 返回寄存器列表

位	31:19	18	17	16	15:12	11:8	7:3	2:0
名称		FIFORST	FIFOFULL	FIFOEMPTY		PDMFST		PDMFLT
访问		R/W	R	R		R		R/W
复位值		0x0	0x0	0x1		0x0		0x0

字段	说明
[18] FIFORST	**FIFO 复位（FIFO Reset）** 写 1 复位，自动清 0。 0：无作用 1：FIFO 复位
[17] FIFOFULL	**FIFO 满标志（FIFO Full Flag）** 0：FIFO 未满 1：FIFO 满
[16] FIFOEMPTY	**FIFO 空标志（FIFO Empty Flag）** 0：FIFO 非空 1：FIFO 空
[11:8] PDMFST	**FIFO 状态（FIFO Status）** 0000：FIFO 中没有数据 0001：FIFO 中有 1 个数据 0010：FIFO 中有 2 个数据 0011：FIFO 中有 3 个数据 …… 1000：FIFO 中有 8 个数据
[2:0] PDMFLT	**FIFO 满中断阈值选择。** 000：FIFO 中有 1 个或 1 个以上数据时触发 FIFO 满中断 001：FIFO 中有 2 个或 2 个以上数据时触发 FIFO 满中断 010：FIFO 中有 3 个或 3 个以上数据时触发 FIFO 满中断 011：FIFO 中有 4 个或 4 个以上数据时触发 FIFO 满中断 …… 111：FIFO 中有 8 个

22.5.2.6 PDM 中断使能寄存器（PDM_IER）

- 名称：PDM Interrupt Enable Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		CTIE	FIFIE	FIOIE	DOFIE	DFIE	LLIE	HLIE
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[6] CTIE	**CTOINTR 中断使能（Clock Timeout Interrupt Enable）** 0：不使能 1：使能
[5] FIFIE	**FIFINTR 中断使能（FIFO Full Interrupt Enable）** 0：不使能 1：使能
[4] FIOIE	**FIOINTR 中断使能（FIFO Overflow Interrupt Enable）** 0：不使能 1：使能
[3] DOFIE	**DOINTR 中断使能（Data Overflow Interrupt Enable）** 0：不使能 1：使能
[2] DFIE	**DFINTR 中断使能（Data Finish Interrupt Enable）** 0：不使能 1：使能
[1] LLIE	**LLINTR 中断使能（Low Level Interrupt Enable）** 0：不使能 1：使能
[0] HLIE	**HLINTR 中断使能（High Level Interrupt Enable）** 0：不使能 1：使能

22.5.2.7 PDM 中断状态寄存器（PDM_ISR）

- 名称：PDM Interrupt Status Register
- 偏移地址：0x18
- 默认值：0x00000000
- 返回寄存器列表

位	31:7	6	5	4	3	2	1	0
名称		CTOINTR	FIFINTR	FIOINTR	DOINTR	DFINTR	LLINTR	HLINTR
访问		R/W1C	R	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[6] CTOINTR	**时钟超时中断（Clock Timeout Interrupt）** PDM 作为从机时，输入使能翻转的时长超过了 CLKTO 寄存器的值时会触发中断。 0：无效果 1：起中断 对该位写 1 会清除 CTOINTR 中断
[5] FIFINTR	**FIFO 满中断（FIFO Full Interrupt）** FIFO 中的数据大于预设值（PDMFLT）时会触发中断。 0：无效果 1：起中断 FIFO 中的数据小于等于预设值（PDMFLT）时会自动清除 FIFINTR 中断
[4] FIOINTR	**FIFO 溢出中断（FIFO Overflow Interrupt）** FIFO 已满，且当前 FIFO 中还有一个数据写入操作，造成 FIFO 溢出并触发中断。 0：无效果 1：起中断 对该位写 1 会清除 FIOINTR 中断
[3] DOINTR	**数据溢出中断（Data Overflow Interrupt）** DFINTR 中断已经触发，软件没读取 PDM_CDAT 也没有对 DFINTR 位写 1 去清除 DFINTR，此时比较滤波器完成了一个新的计算结果并覆盖 PDM_CDAT 寄存器时，会触发 DOINTR 中断。 0：无效果 1：起中断 对该位写 1 会清除 DOINTR 中断
[2] DFINTR	**数据完成中断（Data Finish Interrupt）** 比较滤波器完成计算并将数据写入到 PDM_CDAT 寄存器中，此时触发中断代表 PDM_CDAT 有效可读。 0：无效果 1：起中断 读取 PDM_CDAT 寄存器或者对该位写 1 都可以清除 DFINTR 中断。

[1] LLINTR	**低阈值中断（Low Level Interrupt）** 比较滤波器的计算结果小于等于预设值（LLT）时触发中断。 0：无效果 1：起中断 对该位写 1 会清除 LLINTR 中断
[0] HLINTR	**高阈值中断（High Level Interrupt）** 比较滤波器的计算结果大于等于预设值（HLT）时触发中断。 0：无效果 1：起中断 对该位写 1 会清除 HLINTR 中断

22.5.2.8 PDM 主滤波器数据寄存器（PDM_DDAT）

- **名称：PDM Data Filter Data Register**
- **偏移地址：0x1C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:24	23:0
名称		DDAT
访问		R
复位值		0x0

字段	说明
[23:0] DDAT	**主滤波器数据寄存器（Data Filter Data Register）** 主滤波器计算结果

22.5.2.9 PDM 比较滤波器数据寄存器（PDM_CDAT）

- 名称：PDM Comparator Filter Data Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		CDAT
访问		R
复位值		0x0

字段	说明
[15:0]	**比较滤波器数据寄存器（Comparator Filter Data Register）**
CDAT	比较滤波器计算结果

22.5.2.10 PDM FIFO 数据寄存器（PDM_FDAT）

- 名称：PDM FIFO Data Register
- 偏移地址：0x24
- 默认值：0x00000000
- 返回寄存器列表

位	31:24	23:0
名称		FDAT
访问		R
复位值		0x0

字段	说明
[23:0]	**FIFO 数据寄存器（FIFO Data Register）**
FDAT	读取 FIFO 中接收到的主滤波器计算结果

22.5.2.11 PDM 高阈值比较寄存器（PDM_CMPH）

- 名称：PDM High-Level Compare Register
- 偏移地址：0x28
- 默认值：0x00007FFF
- 返回寄存器列表

位	31:16	15:0
名称		HLT
访问		R/W
复位值		0x7FFF

字段	说明
[15:0] HLT	比较滤波器的高阈值（Comparator High-Level Data）

22.5.2.12 PDM 低阈值比较寄存器（PDM_CMPL）

- 名称：PDM Low-Level Compare Register
- 偏移地址：0x2C
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		LLT
访问		R/W
复位值		0x0

字段	说明
[15:0] LLT	比较滤波器的低阈值（Comparator Low-Level Data）

22.5.2.13 PDM 时钟翻转超时寄存器（PDM_CLKTO）

- 名称：PDM Clock Rollover Timeout Register
- 偏移地址：0x30
- 默认值：0x00003FFF
- 返回寄存器列表

位	31:24	23:0
名称		CLKTO
访问		R/W
复位值		0x3FFF

字段	说明
[23:0]	**PDM 时钟超时寄存器（Clock Timeout Register）**
CLKTO	PDM 输入时钟翻转超时阈值，以系统时钟为单位

22.5.2.14 PDM 同步控制寄存器（PDM_SDSYNC）

- 名称：PDM Synchronization Control Register
- 偏移地址：0x34
- 默认值：0x00000400
- 返回寄存器列表

位	31:11	10	9	8	7	6	5:0
名称		WTSCLREN	FFSYNCCLREN	WTSYNCLR	WTSYNFLG	WTSYNCEN	SYNCSEL
访问		R/W	R/W	R/W	R	R/W	R/W
复位值		0x1	0x0	0x0	0x0	0x0	0x0

字段	说明
[10]	**WTSYNFLG 清除方式使能位（WTSYNFLG Clear Enable）**
WTSCLREN	0：当对 WTSYNCLR 写 1 时 WTSYNFLG 会清 0
	1：当 FIFINTR 中断触发或者对 WTSYNCLR 写 1 时 WTSYNFLG 会清 0
[9]	**SDSYNC 信号复位 FIFO 使能位（FIFO Reset By SDSYNC）**
FFSYNCCLREN	0：SDSYNC 信号不会触发 FIFO 复位
	1：SDSYNC 信号会触发 FIFO 复位
[8]	**WTSYNFLG 清除位（WTSYNFLG Clear）**
WTSYNCLR	0：无作用
	1：清除 WTSYNFLG
[7]	**收到 SDSYNC 标志（Receive SDSYNC Flag）**
WTSYNFLG	PDM 收到 SDSYNC 信号时会使 WTSYNFLG 置 1。

[6]	**FIFO 在 WTSYNFLG 标志下的表现（The performance of FIFO under the WTSYNFLG）**
WTSYNCEN	0：FIFO 在任何时候都可以写入
	1：FIFO 只在 WTSYNFLG 为 1 时才能写入，在 WTSYNFLG 为 0 时不能写入
[5:0]	**SDSYNC 选择（SDSYNC Select）**
SYNCSEL	PWM 有 14 个信号会输出给 PDM，PDM 只选择其中一个信号作为 SDSYNC。

23 内部集成接口（I2C）

23.1 简介

I2C 总线是一个两线串行接口，其中两线位串行数据（SDA）和串行时钟（SCL）线在连接到总线器件间传递信息。每个器件都有一个唯一的地址识别，而且都可以作为一个发送或接收器。除了发送器和接收器外，器件在执行数据传输时也可以被看做是主机或者从机。主机是初始化总线的数据传输并产生允许传输的时钟信号的器件。此时，任何被寻址的器件都被认为是从机。

I2C 可以工作在标准模式（数据传输速率为 0~100 Kbps），快速模式（数据传输速率最大为 400 Kbps）。

23.2 结构框图

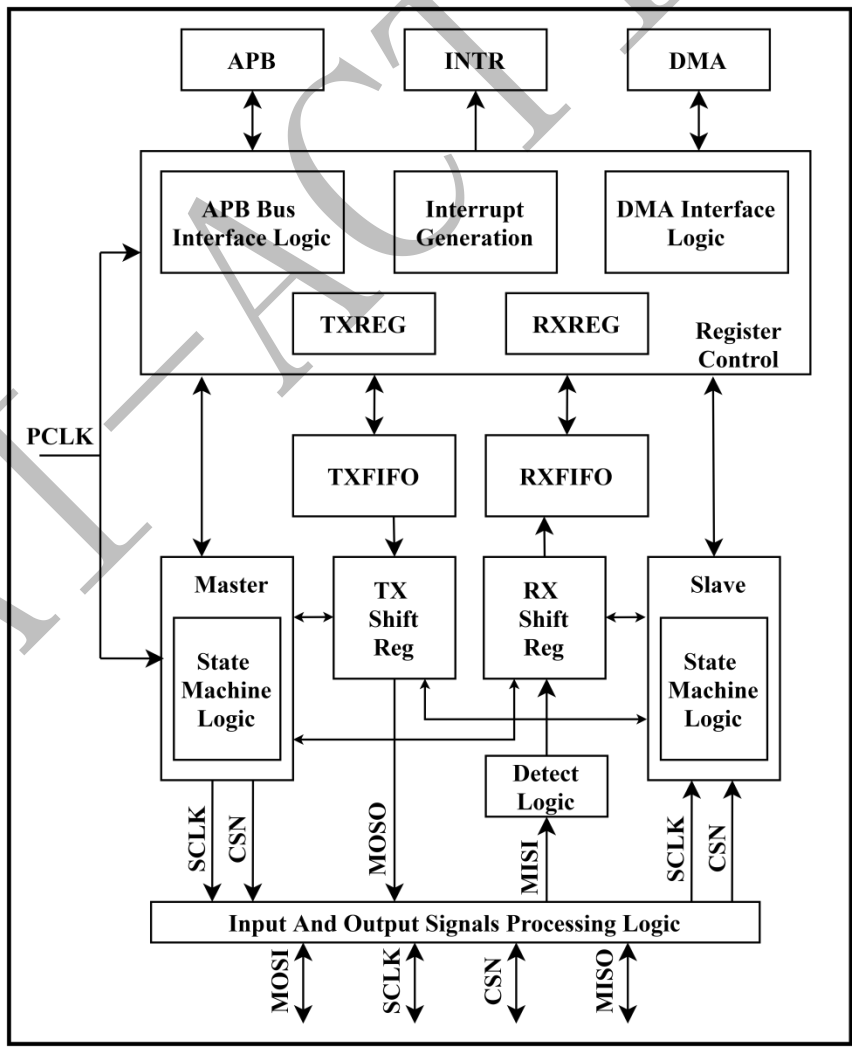


图 23-1 I2C 顶层结构框图

23.3 主要特性

- 半双工操作模式
- 支持主从模式
- 支持 7 位地址和 10 位地址
- 支持标准模式 100Kbps 和快速模式 400Kbps
- 可产生 START、STOP、RESTART、ACK 信号检测
- 在主模式下只支持一个主机
- 支持 DMA 操作
- 分别有深度为 16 的 RXFIFO 和 TXFIFO
- 支持中断和查询操作

23.4 功能描述

23.4.1 I2C 协议

23.4.1.1 I2C 电气结构

SDA 和 SCL 都是双向线路，通过连接到电流源或上拉电阻，产生正电源电压电阻（见图 23-2）。当 I2C 总线处于空闲状态时，SCL 和 SDA 两条线均为高。连接到总线的设备的输出级必须具有漏极开路才能执行线与功能，同时还需要配置 GPIO 上拉。

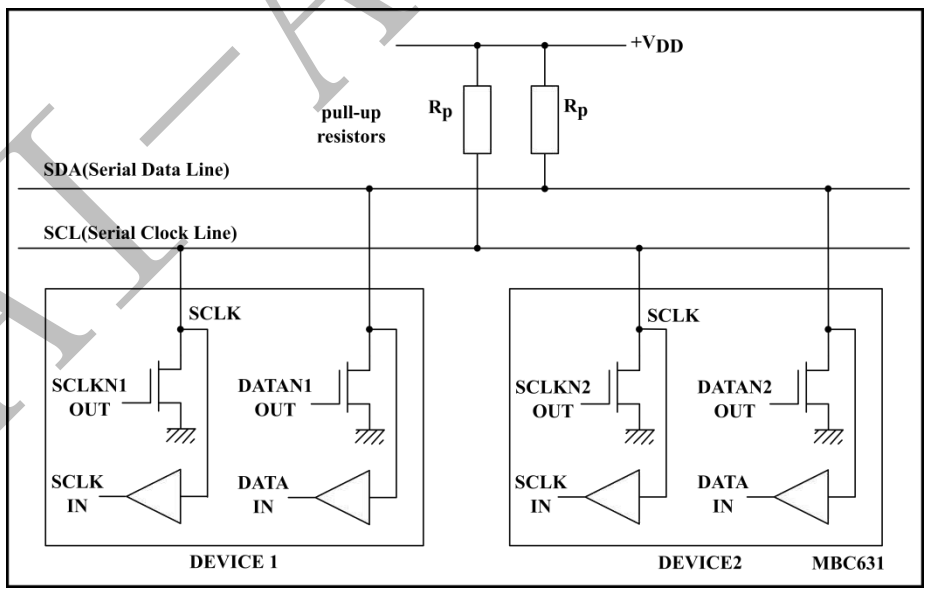


图 23-2　I2C 总线电气结构图

23.4.1.2 I2C 基本结构

I2C 基本结构由下：

- Transmitter：将数据发送到总线的设备。Transmitter 可以是发送数据到总线的主机设备（Master-Transmitter），也可以是响应主机的要求将数据发送到总线的从机设备（Slave-Transmitter）。
- Receiver：从总线接收数据的设备。Receiver 可以是根据自己的请求接收数据的设备（Master-Receiver），也可以是响应于来自主设备的请求的设备（slave-Receiver）。
- Master：初始化传输（START 命令），生成时钟（SCL）信号并终止传输（STOP 命令）的组件。主机可以是 Transmitter 或 Receiver。
- Slave：主机寻址的设备。从机可以是 Transmitter 或 Receiver。
- SDA：数据信号线（串行数据）。
- SCL：时钟信号线（串行时钟）。
- START（RESTART）：数据传输以 START 或 RESTART 条件开始。 SDA 数据线的电平从高电平变为低电平，而 SCL 时钟线保持高电平。当发生这种情况时，总线变得繁忙。
- STOP：数据传输因 STOP 条件而终止。 当 SDA 数据线上的电平从低电平变为高电平，而 SCL 时钟线保持高电平时，就会发生这种情况。数据传输终止后，总线将再次空闲。

上述结构关系如图 23-3 所示。

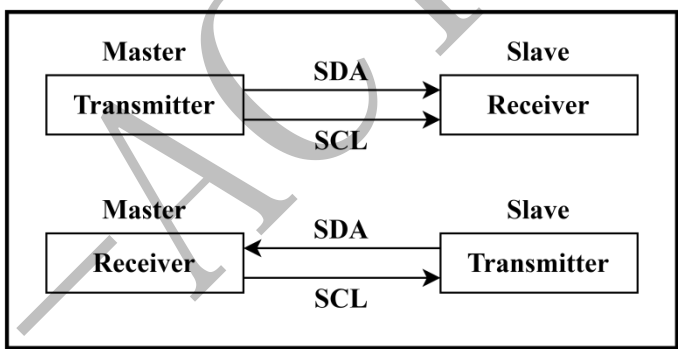


图 23-3 主/从和发送器/接收器的关系

Master 负责产生时钟并控制数据传输。Slave 负责向 Master 发送数据或从 Master 接收数据。数据确认由接收数据的设备发送，该设备可以是 Master，也可以是 Slave。

每个 Slave 都有一个由系统设计人员确定的唯一地址。当 Master 要与从机通信时，Master 会先发送一个 START / RESTART 命令，接着发送 Slave 的地址和一个控制位（R / W）。Slave 在接收到地址之后会发送一个确认（ACK）脉冲。

如果 Master（Transmitter）正在写入 Slave（Receiver），则 Receiver 将获得一个字节的数据。此事务将继续进行，直到 Master 以 STOP 条件终止传输为止。如果 Master 正在从 Slave 读取数据，则 Slave（Transmitter）向 Master（Receiver）发送一个字节的数据，然后 Master 用 ACK 脉冲确认该事务。该事务持续进行，直到 Master 在接收到最后一个字节后不确认（NACK）事务来终止传输，然后 Master 发出 STOP 条件或在发出 RESTART 条件后寻址另一个 Slave。上述行为如图 23-4 所示。

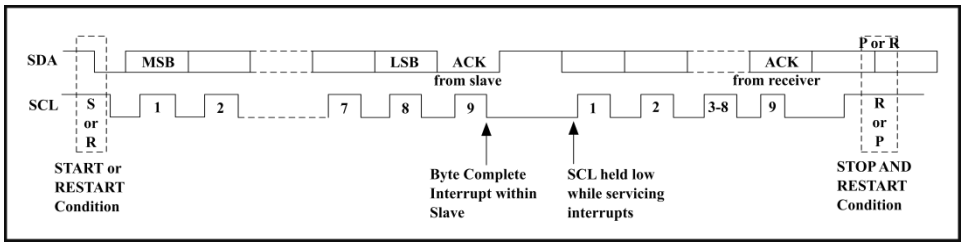


图 23-4　I2C 总线的数据传输

23.4.1.3　STOP 和 START

总线空闲时，SCL 和 SDA 信号均通过总线上的外部上拉电阻上拉。 当 Master 希望在总线上开始传输时，Master 发出 START（SCL 为高时 SDA 信号由高到低的跳变）；当 Master 希望终止发送时，主机发出 STOP（SCL 为高时 SDA 线从低到高的跳变）。当数据在总线上传输时，当 SCL 为高时，SDA 线必须稳定。图 23-5 显示了 START 和 STOP 条件的时序。

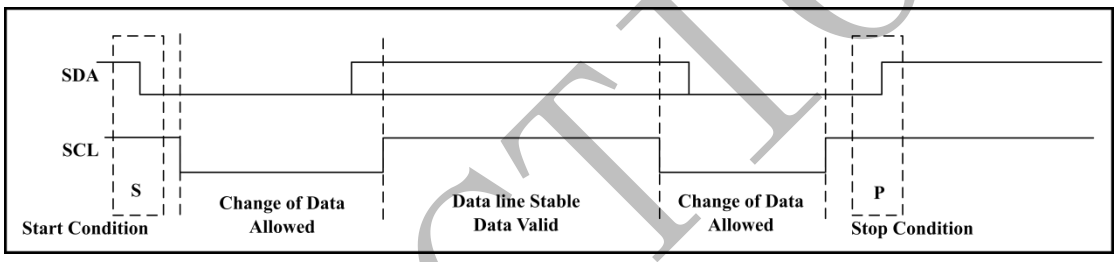


图 23-5　START 和 STOP 的时序

23.4.1.4　寻址从协议

地址格式有两种：7 位地址格式和 10 位地址格式。

7 位地址格式

图 23-6 中显示在起始条件（S）后发送的一个字节的前 7 位（bit 7:1）为从机地址，最低位（bit 0）是数据方向位，当 bit 0 为 0，表示主机写数据到从机，1 表示主机从从机读数据。

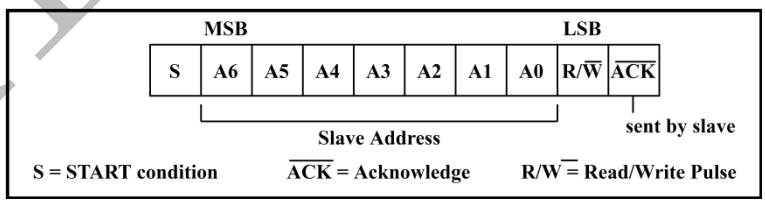


图 23-6　7 位地址格式

10 位地址格式

在 10 位的地址格式中，发送 2 个字节来传输 10 位地址。发送的第一个字节的位的描述如下：第一个 5 位（bit 7:3）用于告示从机接下来是 10 位的传输。第一个字节的后两个字节（bit 2:1）位从机地址的 bit 9:8，最低位（bit 0）是数据方向位（R/W）。传输的第二个字节为 10 位地址的低八位。具体如图 23-7 所示。

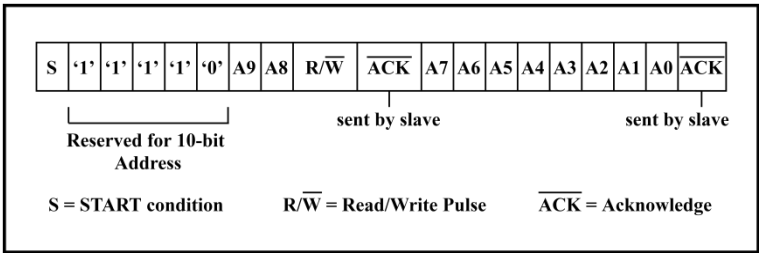


图 23-7　10 位地址格式

表 23-1 定义了特殊用途和保留的第一个字节地址。

表 23-1　I2C / SMBus 第一个字节中的位定义

Slave Address	R/W Bit	Description
0000 000	0	通用呼叫地址。i2c 将数据放置在接收缓冲区中，并发出常规调用中断。
0000 000	1	START 字节。
0000 001	X	CBUS 地址。 i2c 忽略这些访问。
0000 010	X	Reserved.
0000 011	X	Reserved
0000 1XX	X	高速主模式
1111 1XX	X	Reserved
1111 0XX	X	10 位从机寻址。
0001 000	X	SMbus Host
0001 100	X	SMBus 警报响应地址
1100 001	X	SMBus 设备默认地址

I2C 不会限制使用这些保留的地址。但是，如果使用这些保留的地址，则可能会与其他 I2C 组件不兼容。

23.4.1.5　收发协议

Master 可以作为 Master-Transmitter 或 Master-Receiver。 Slave 响应来自 Master 的请求，以分别向总线发送数据或从总线接收数据，分别充当 Slave-Transmitter 或 Slave-Receiver。

Master-Transmitter 和 Slave-Receiver

所有数据均以字节格式传输，每次数据传输的字节数没有限制。在 Master 发送地址和 R/W 位或 Master 向 Slave 发送数据字节之后，Slave-Receiver 必须以确认信号（ACK）进行响应。当 Slave-Receiver 未响应 ACK 脉冲时，Master 通过发出 STOP 条件中止传输。如图 23-8 所示，Slave 必须将 SDA 线保持高电平，以便 Master 可以中止传输。如果 Master-Transmitter 正在发送数据，Slave-Receiver 在接收到每个字节的数据后用 ACK 响应 Master-Transmitter。

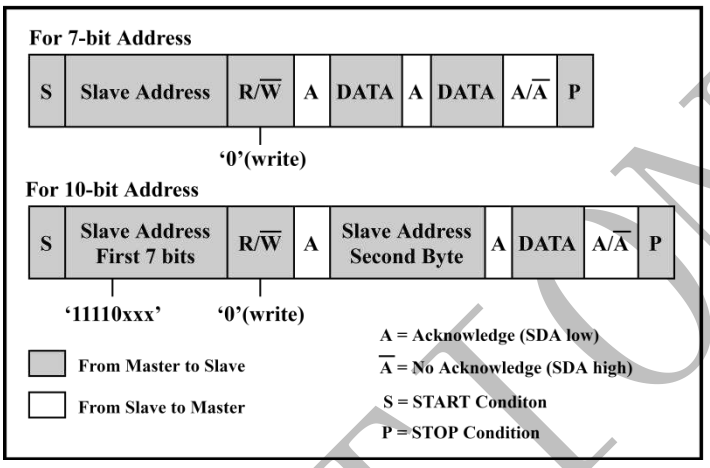


图 23-8　Master-Transmitter 协议

Slave-Transmitter 和 Master-Receiver

如图 23-9 所示，如果 Master 正在接收数据，则在接收到一个字节的数据（最后一个字节除外）之后，Master 以一个 ACK 脉冲响应 Slave-Transmitter，通知 Slave-Transmitter 这是最后一个字节。Slave-Transmitter 检测到无应答（NACK）后放弃 SDA 线，Master 可以发出 STOP。

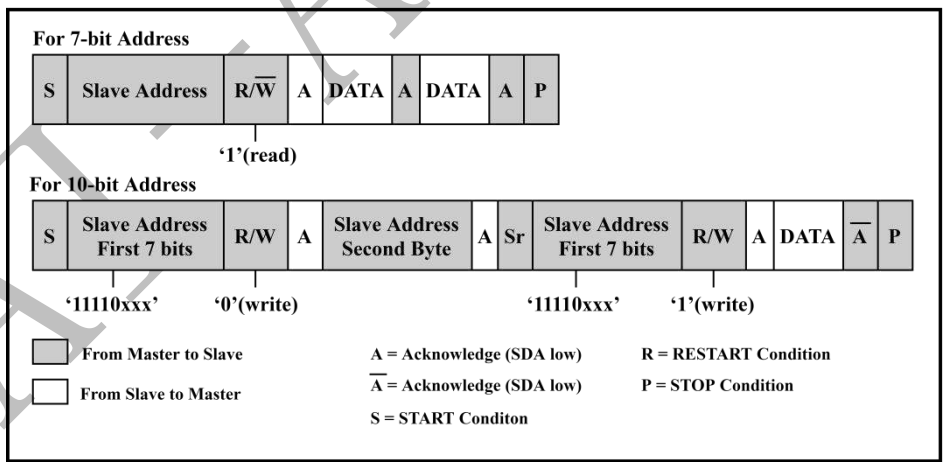


图 23-9　Master-Receiver 协议

当 Master 不希望以 STOP 放弃总线时，Master 可以发出 RESTART。RESTART 波形与 START 相同。

23.4.1.6 Tx FIFO 管理以及 START, STOP 和 RESTART 生成

当 Tx FIFO 变空时，I2C 不会产生 STOP。此时，I2C 将 SCL 线保持为低电平，从而使总线停顿，直到 Tx FIFO 中有新的数据可用为止。仅当用户通过设置写入 I2C_CTRL 寄存器的命令的第 9 位（STOP）明确要求时，才产生 STOP。

图 23-10 说明了当 Tx FIFO 作为 Master-Transmitter 工作时 Tx FIFO 变空时 I2C 的行为，并显示了 STOP 的产生。

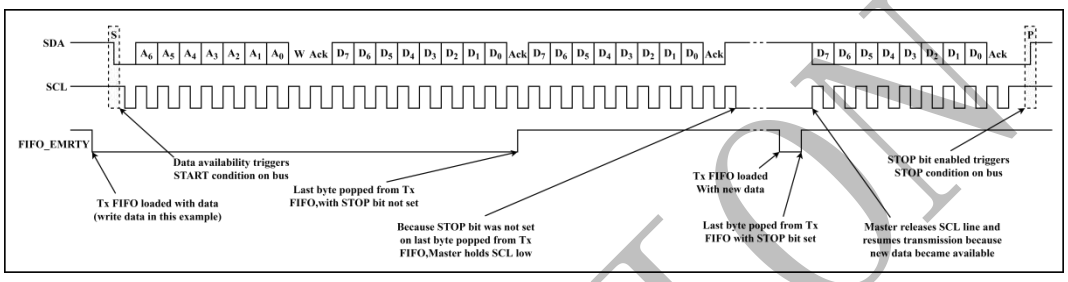


图 23-10　Master-Transmitter—Tx FIFO 变空

图 23-11 说明了当 Tx FIFO 作为 Master-Receiver 工作时 Tx FIFO 变空时 I2C 的行为，并显示了 STOP 的产生。

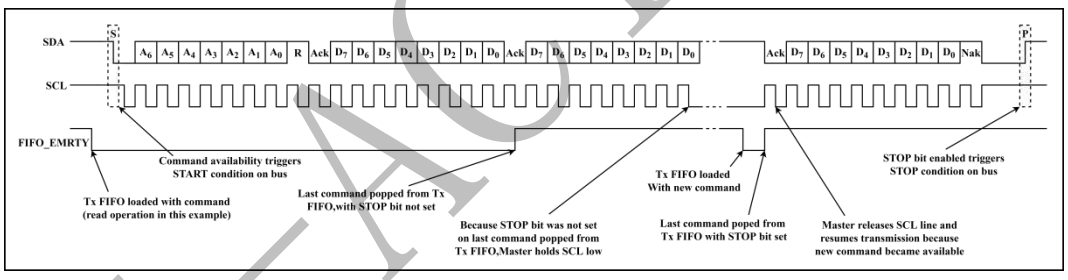


图 23-11　Master-Receiver—Tx FIFO 变空

用户可以控制 I2C 总线上 RESTART 条件的产生。如果设置了 I2C_CTRL 寄存器的第 10 位（RESTART），则在将数据字节写入从设备或从设备读取数据字节之前，将产生 RESTART。

图 23-12 说明了作为 Master-Transmitter 工作期间的这种情况。

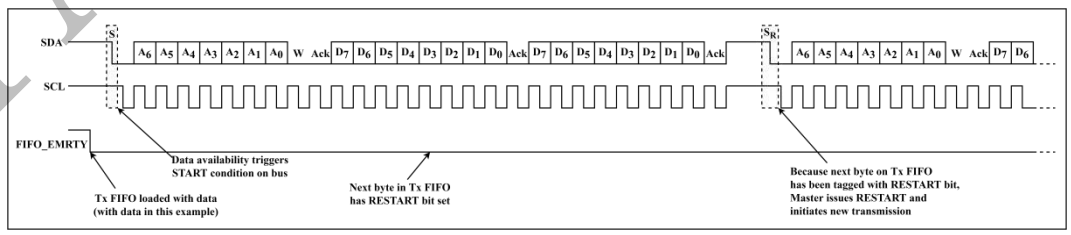


图 23-12　Master-Transmitter 工作

图 23-13 说明了作为 Master-Receiver 工作期间的这种情况。

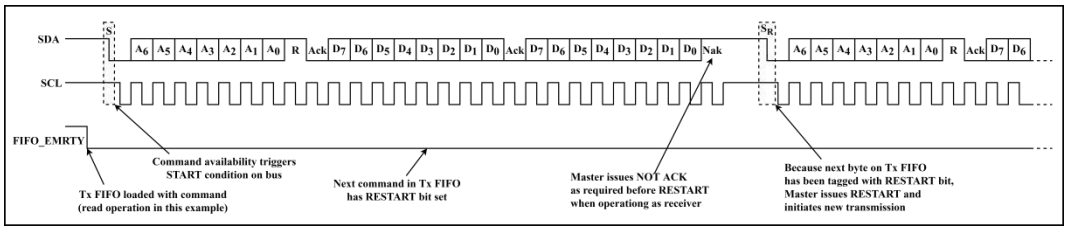


图 23-13　Master-Receiver 工作

23.4.2　操作模式

I2C 在同一时间里只能设置为 I2C Master 模式或者 I2C Slave 模式，不能同时进行。通过配置 I2C_ENABLE 的 MSTEN 位来配置 I2C 作为 Master 还是 Slave。

23.4.2.1　Slave Mode

初始化配置

要将 I2C 用作 Slave，需要执行以下步骤：

1. 对 I2C_ENABLE 寄存器的 BIT0 写入 0 来关闭 I2C。
2. 写入 I2C_SAR 寄存器（位 9:0）以设置 Slave 地址。这是 I2C 响应的地址。
3. 写入 I2C_ENABLE 寄存器以指定支持哪种寻址类型（通过将 BIT9 设置为 7 位或 10 位），在 BIT8（MSTEN）中写入“0”，以仅从机模式启用 I2C。
4. 通过在 I2C_ENABLE 寄存器的 BIT0 中写入“1”来启动 I2C。

单字节的 Slave-Transmitter

当总线上的另一个 I2C 主设备寻址 I2C 并请求数据时，此时 I2C 充当从 Transmitter，并发生以下步骤：

1. 外部 I2C 主设备使用与 I2C 的 I2C_SAR 寄存器中的从设备地址启动 I2C 传输。
2. I2C 确认发送的地址并识别传输方向，以指示它正在充当从 Transmitter。
3. I2C 产生 RDREQI 中断（I2C_INTR 寄存器的 BIT5）并将 SCL 线保持为低电平。它处于等待状态，直到软件响应为止。
   如果由于 I2C_INTREN 寄存器的 BIT5（RDREQIE）被设置为 0 而屏蔽了 RDREQ 中断，则建议使用硬件或软件定时例程来指示 CPU 进行对 CPU 的定期读取 I2C_INTR 寄存器。读取 RDREQI 中断后，软件必须采取行动来满足 I2C 传输要求：使用的时序间隔应为 I2C 可以处理的最快 SCL 时钟周期的 10 倍。例如，对于 400 kb / s，定时间隔为 25us。
4. 如果在接收到读取请求之前，Tx FIFO 中剩余数据，则 I2C 会产生 TXABRTI 中断（I2C_INTR 寄存器的 BIT6）以从 TX FIFO 中清除旧数据。

注意：由于每当 TX_ABRT 中断事件发生时，I2C 的 Tx FIFO 都会被强制进入刷新/复位状态，因此在尝试写入 Tx FIFO 之前，软件必须通过对 I2C_IN 寄存器的 BIT6 写 1 从而释放 I2C。

5. 软件将要写入的数据写入 I2C_DATA 寄存器。

6. 在继续之前，软件必须清除 I2C_INT 寄存器的 RDREQI 和 TXABRTI 中断（分别为 BIT5 和 BIT6 写 1）。
7. I2C 释放 SCL 并发送字节。
8. 主机可以通过发出 RESTART 条件来保持 I2C 总线，也可以通过发出 STOP 条件来释放 I2C 总线。

单字节的 Slave-Receiver

当总线上的另一个 I2C 主设备寻址 I2C 并发送数据时，此时 I2C 充当从 Receiver，并发生以下步骤：

1. 外部 I2C 主设备启动 I2C 传输，其地址与 I2C_SAR 寄存器中 I2C 的从地址相匹配。
2. I2C 识别地址以及传输方向，并用 ACK 响应。
3. I2C 接收发送的字节，并将其放入接收缓冲区。

注意：如果在压入一个字节时 Rx FIFO 完全充满了数据，I2C 会将 SCL 线保持为低电平，直到 Rx FIFO 有一定空间，然后继续下一个读取请求。

4. I2C 产生 RXFIE 中断（I2C_INT [2]寄存器）。
5. 软件可以从 I2C_DATA 寄存器中读取字节（位 7：0）。
6. 另一个主设备可以通过发出 RESTART 条件来保持 I2C 总线，或者通过发出 STOP 条件来释放总线。

23.4.2.2 Master Mode

初始化配置

1. 通过将 0 写入 I2C_ENABLE 寄存器的位 0 来禁用 I2C。
2. 写入 I2C_ENABLE 寄存器，指定当设备是主机（BIT8）时 I2C 是否以 7/10 位（BIT9）寻址模式开始传输。
3. 将要寻址的 I2C 器件的地址写入 I2C_TAR 寄存器。
4. CMD、STOP 和 RESTART 还有需要传输的数据长度写入 I2C_CTRL 寄存器（写之前需要保证 I2C_ENABLE[4]的 ICUP 已经写 1）。
5. 通过将 1 写入 I2C_ENABLE 寄存器的位 0 来启用 I2C。
6. 配置 I2C_DATA 寄存器的数据。

程序流程图

下面的流程图为 I2C 接口作为主机的程序示例：

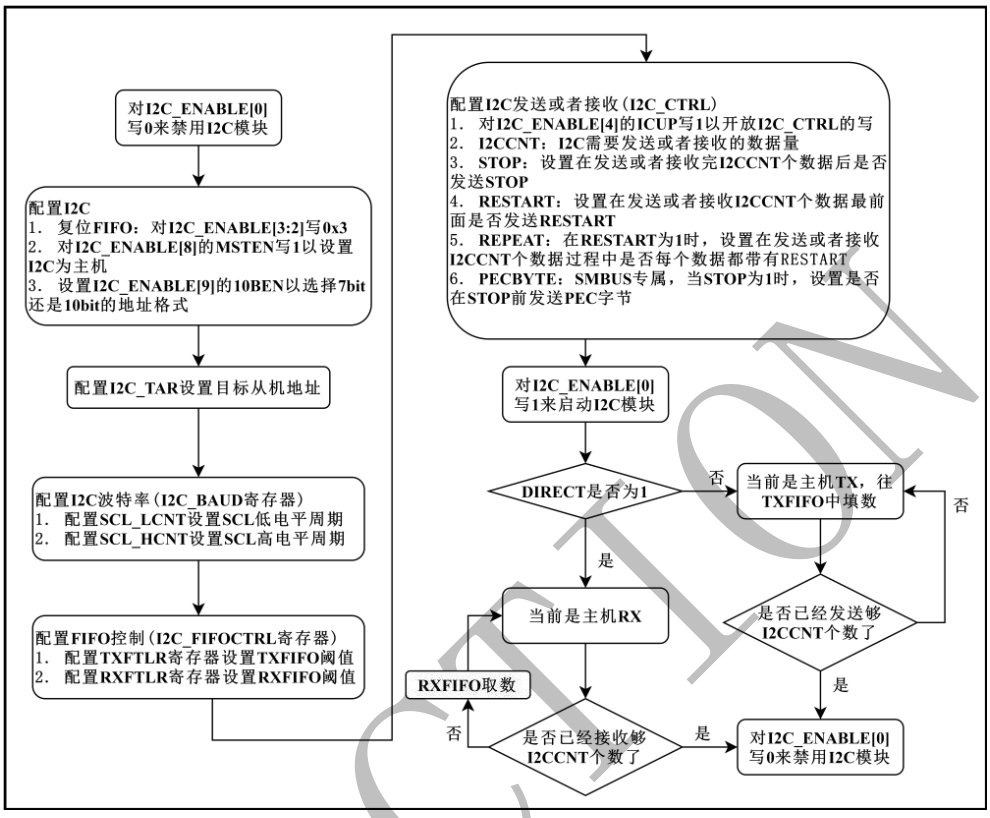


图 23-14　I2C 接口主机流程图

23.4.2.3　I2C_CTRL 寄存器

1. 要写 I2C_CTRL[7:0]的 I2C_CNT 时，需要保证 I2C_ENABLE[4]的 I2C_CTRL_UP 已经写 1，否则将无法更新 I2C_CNT 的值。
2. 传输方向写入 I2C_CTRL[8]的 DIRECT，写 1 时代表主机作为接收器，写 0 代表主机作为发送器。
3. I2C_CTRL[9]的 STOP 位指的是 I2C_CNT 的最后一个数据是否要发送 STOP，如果设置为 1，I2C_CNT 的最后一个数据完成后会发送 STOP，否则总线会进入挂起状态
4. I2C_CTRL[10]的 RESTART 和 I2C_CTRL[11]的 REPEAT 位，REPEAT 位只在 RESTART 位为 1 的时候才会有效。RESTART 位为 1 时，当 REPEAT 为 0，RESTART 只会在 I2C_CNT 的第一个数据时候生成，其余数据都不会生成；当 REPEAT 为 1，RESTART 在 I2C_CNT 的每一个数据都会生成 RESTART。

23.4.3　I2C_CLK 频率配置

当 I2C 配置为主设备时，必须先设置 I2C_BAUD 寄存器，然后才能进行任何 I2C 总线事务以确保正确的 I / O 时序。I2C_BAUD 寄存器是：

- SCLHCNT
- SCLLCNT

注意：START，STOP 和 RESTART 寄存器的 tBUF 时序和建立/保持时间将 HCNT / * LCNT 寄存器设置用于相应的速度模式。*

下表列出了来自 I2C_BAUD 编程寄存器的 I2C 时序参数。

表 23-2　从* CNT 寄存器推导 I2C 时序参数

时序参数	符号	时序
SCL 时钟的低电平周期	t_{LOW}	SCL_LCNT
SCL 时钟的高电平周期	t_{HIGH}	SCL_HCNT
重复 START 的建立时间	$t_{SU;STA}$	SCL_LCNT
重复 START 的保持时间	$t_{HD;STA}$	SCL_HCNT
STOP 的建立时间	$t_{SU;STO}$	SCL_HCNT
STOP 和 START 条件之间的总线空闲时间	t_{BUF}	SCL_LCNT
数据保持时间	$t_{HD;DAT}$	TIME_HDDAT
数据建立时间	$t_{SU;DAT}$	TIME_SUDAT

23.4.4　SDA 保持时间和建立时间

I2C 协议规范要求 SDA 信号（$t_{HD;DAT}$）的保持时间为 300ns，以此来渡过 SCL 下降沿的未定义区。SCL 和 SDA 信号的电路板延迟可能意味着 I2C 主设备满足了保持时间要求，但 I2C 从设备却没有达到（或反之）。当每个应用遇到不同的板延迟时，I2C 提供了一个软件可编程寄存器（I2C_TIMING[7:0]：HDDAT），用于动态调整 SDA 保持时间。I2C_TIMING[7:0]位用于控制从机和主机模式下发送期间（SCL 从高电平变为低电平之后）SDA 的保持时间。

I2C 协议规定要求 SDA 信号（$t_{SU;DAT}$）的建立时间最小为 250ns，保证接收器在上升沿后能够准确的采样到准确的值，但是在不同电路板的延迟中可能会导致采样点的不准确，因此在应对不同的板级延迟时，I2C 提供了一个软件可编程寄存器（I2C_TIMING[15:8]：SUDAT），用于动态调整 SDA 建立时间。

下图为标准 I2C 协议中建立时间和保持时间的时序图

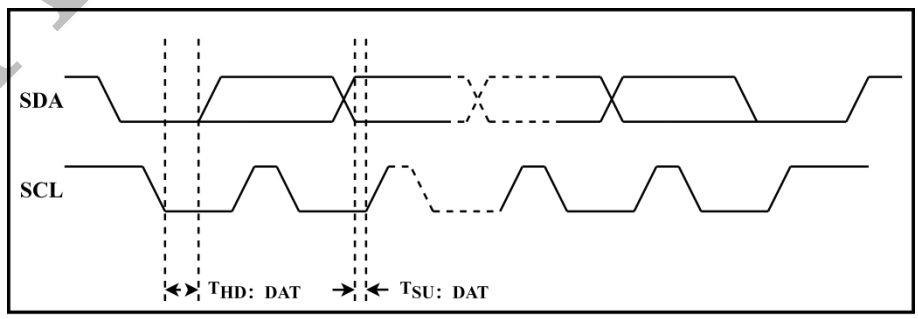


图 23-15　I2C 协议中的建立时间和保持时间

23.4.4.1 从机接收器中的 SDA 保持时序

当 I2C 充当接收器时，根据 I2C 协议，设备应在内部保持 SDA 线以桥接 SCL 的逻辑 1 和逻辑 0 之间的未定义间隙。I2C_TIMING[23:16]的 RXSUDAT 可用于更改内部保持时间，I2C 应用于输入 SDA 线。I2C_TIMING[23:16]的 RXSUDAT 寄存器中的每个值代表一个 ic_clk 周期的单位。I2C_TIMING[23:16]的 RXSUDAT 的最小值为 0。该保持时间仅在 SCL 为 HIGH 时适用。SCL 内部变为低电平后，接收器不会扩展 SDA。

图 23-16 显示了 i2c 作为接收器，其 I2C_TIMING[23:16]的 RXSUDAT 设置为大于或等于 3。

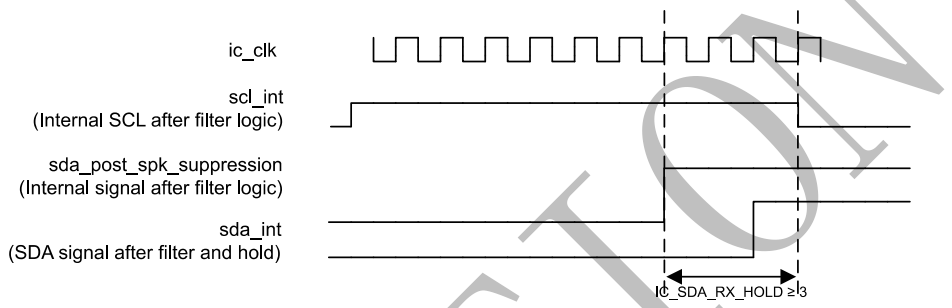


图 23-16 RXSUDAT 大于或等于 3

如果 I2C_TIMING[23:16]的 RXSUDAT 大于 3，则 I2C 不会在 3 个系统时钟周期之后保持 SDA，因为 SCL 在内部变为低电平。

图 23-17 显示了 I2C 作为接收器，并将 I2C_TIMING[23:16]的 RXSUDAT 设置为 2。

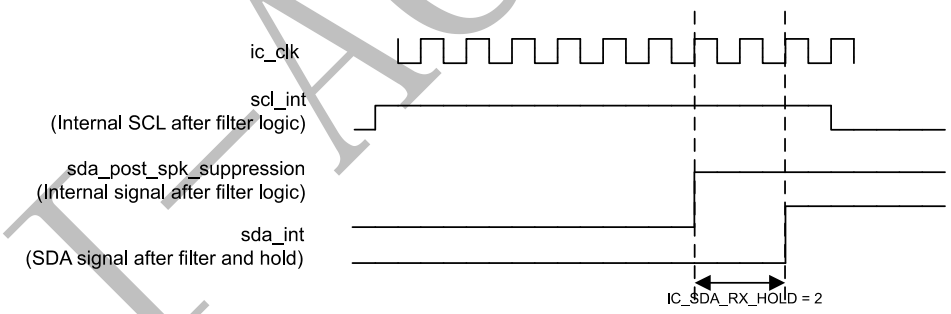


图 23-17 RXSUDAT 等于 2

23.4.5 I2C 中断

根据寻址协议的不同，可以分为 7bit 地址寻址和 10bit 地址寻址，图 23-18 展示的是在 7bit 地址下主发从收、主收从发和寻址失败的时序图，图 23-19 展示的是在 10bit 地址下主发从收、主收从发和寻址失败的时序图。

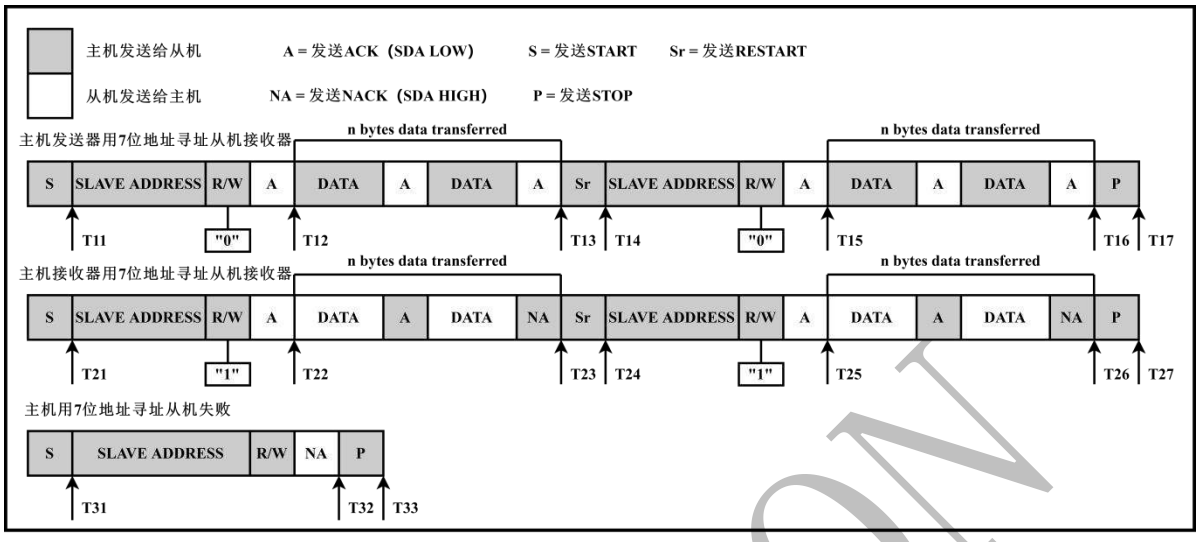


图 23-18　7bit 地址模式下发送接收

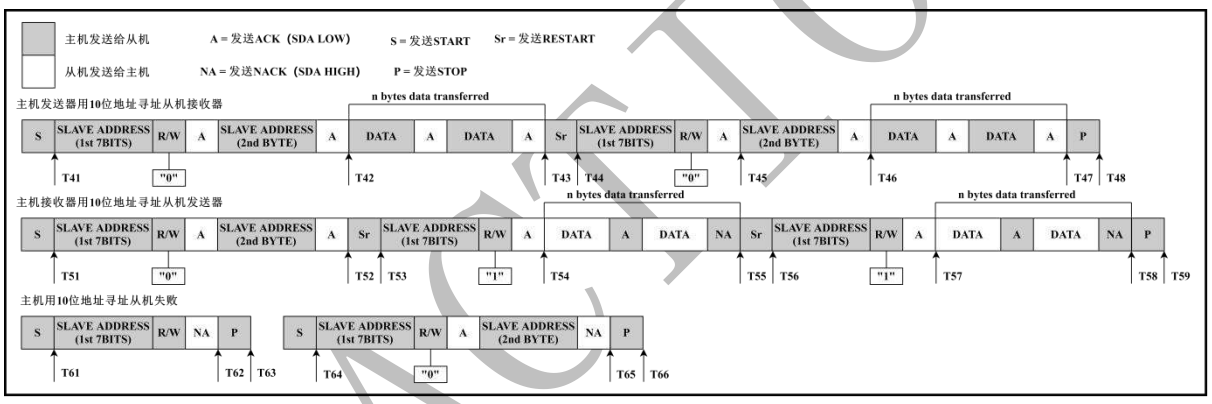


图 23-19　10bit 地址模式下发送接收

下表中列出了 I2C 的中断位含义以及它们对应图 22-22 和图 22-23 的置位时间点。

表 23-3　中断位的描述

中断位	描述	置位时间点
MTXAI	作为主机完成了发送地址（无论收到 ACK 还是 NACK 都会起来）	T12，T15，T22，T25，T32，T42，T46，T52，T57
RXGCI	作为从机收到了 General Call 地址	-
PECRXI	收到 PEC 数据	-
SEXTOI	作为从机，START->ACK、ACK->ACK 和 ACK->STOP 的累计时间 Tlow:sext 超时了	-
MEXTOI	作为主机，START->STOP 的累计时间 Tlow:mext 超时了	-
ALEDETI	作为主机检测是否收到了从机发出的 ALERT 信号	-
NACKI	作为主机检测收到了 NACK	T32，T62，T65
	作为从机检测收到了 NACK	T23，T26，T55，T58
MOHI	I2C 作为主机且当前没有数据可以发送和接收，也没有 STOP 位的产生也就是 SCL 为低，主机处于挂起状态	-
SPDETI	I2C 检测到了 STOP 位	T17，T27，T33，T48，T59，T63，T66

STDETI	I2C 检测到了 START 和 RESTART 位	T11，T14，T21，T24，T31，T41，T44，T51，T53，T56，T61，T64
RXADI	I2C 作为从机时接收到了完整的 10bit 地址或者 7bit 地址并且此时的命令是从机接收	T22，T25，T54，T57
TXADI	I2C 作为从机时接收到了完整的 10bit 地址或者 7bit 地址并且此时的命令是从机发送	T12，T15，T42，T45
MDEI	主机发送或接受完 I2C_CNT 个数据时起中断（发送会完全发完才起中断）	T13，T16，T23，T26，T43，T47，T55，T58
RXOFI	当 RXFIFO 已满并且新数据被接收写入 FIFO 中时，会起溢出错误中断。	-
TXEI	当 TXFIFO 中的数据量小于或者等于预设值（TXFTLR）时置位。	-
RXFI	当 RXFIFO 中的数据量大于或等于预设值（RXFTLR）时置位。	-

23.4.6 DMA 控制接口

I2C 接口支持使用 DMA 来发送和接收数据。通过设置 I2C_ENABLE[6:5]位可以单独开启 DMA 发送或者接收。发送时 RXFIFO 变满（RXFLTR 有关）或者 TXFIFO 变空（TXFLTR 有关）时，会产生 DMA 请求。DMA 控制器会将数据从 I2C_DATA 寄存器中传送到预置的存储区或者将数据从预置的存储区装载到 I2C_DATA 寄存器中去。

23.5 寄存器描述

23.5.1 寄存器列表

I2C 基地址：**0x4000 0000**

偏移	实例地址	名称	默认值	描述
0x00	0x40000000	I2C_ENABLE	0x00000000	I2C 使能寄存器
0x04	0x40000004	I2C_CTRL	0x00000000	I2C 控制寄存器
0x08	0x40000008	I2C_BAUD	0x00000000	I2C 波特率寄存器
0x0c	0x4000000C	I2C_FIFOCTL	0x00000500	I2C FIFO 控制寄存器
0x10	0x40000010	I2C_DATA	0x00000000	I2C 数据读写寄存器
0x14	0x40000014	I2C_INTREN	0x00000000	I2C 中断使能寄存器
0x18	0x40000018	I2C_INTR	0x00000002	I2C 中断寄存器
0x1c	0x4000001C	I2C_TAR	0x00000000	I2C 发送地址寄存器
0x20	0x40000020	I2C_SAR	0x00000000	I2C 接收地址寄存器
0x24	0x40000024	I2C_PEC	0x00000000	I2C PEC 接收寄存器
0x28	0x40000028	I2C_TIMING	0x00001800	I2C 时序寄存器
0x2c	0x4000002C	I2C_TIMEOUT	0x07A2004F	I2C 超时寄存器
0x30	0x40000030	I2C_STATUS	0x00000000	I2C 状态寄存器

23.5.2 寄存器详细描述

23.5.2.1 I2C 使能寄存器（I2C_ENABLE）

- **名称：I2C Enable Register**
- **偏移地址：0x00**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:15	14	13	12	11	10	9	8
名称		SMHEN	SMARPEN	SMALEN	NAEN	GCEN	10BEN	MSTEN
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x1	0x0	0x0	0x0	0x0
位	7	6	5	4	3	2	1	0
名称		DMATXEN	DMARXEN	ICUP	TFRST	RFRST	ABORT	I2CEN
访问		R/W	R/W	R/W	W	W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[14] SMHEN	**SMBUS Host 使能位（SMBUS Host Enable）** 使能后主机发送的地址将固定为 0x8，从机收到的地址为 0x8 才会回复 ACK 0：HOST 不使能 1：HOST 使能
[13] SMARPEN	**SMBUS ARP 使能位（SMBUS APR Enable）** SMBUS ARP 从机默认地址使能，使能主机后发送的地址将固定为 0x61，从机收到的地址为 0x61 才会回复 ACK 0：ARP 不使能 1：ARP 使能
[12] SMALEN	**SMBUS Alert 使能位（SMBUS ALERT ENABLE）** 0：Alert 不使能 1：Alert 使能
[11] NAEN	**NACK 使能位（NACK Enable）** I2C 从机收到数据后是否回复 ACK 0：回复 ACK 1：不回复 ACK
[10] GCEN	**广播使能位（General Call Enable）** 0：General Call 地址不使能 1：General Call 地址使能 注意：主机要发送 General Call 时，I2C_10BIT_EN 必须为 0，DIRECT 也必须为 0
[9] 10BEN	**I2C 10bit 地址使能位（I2C 10BIT Address Enable）** 0：7bit 模式 1：10bit 模式

[8] MSTEN	**I2C 主机使能位（I2C Master Enable）** 0：I2C 作为从机 1：I2C 作为主机
[6] DMATXEN	**I2C 发送 DMA 使能位（TX DMA Enable）** 该位启用/禁用发送 FIFO DMA 通道 0：禁用发送 DMA 通道 1：使能发送 DMA 通道
[5] DMARXEN	**I2C 接收 DMA 使能位（RX DMA Enable）** 该位启用/禁用接收 FIFO DMA 通道 0：禁用接收 DMA 通道 1：使能接收 DMA 通道
[4] ICUP	**I2C_CTRL 寄存器更新使能位（I2C_CTRL Update）** 写 I2C_CTRL 之前需将该位写 1 才能改 I2C_CTRL 的配置，配置完 I2C_CTRL 寄存器后自动清 0 0：I2C_CTRL 不可写 1：更新 I2C_CTRL 内容
[3] TFRST	**TXFIFO 复位（TXFIFO RESET）** TXFIFO 复位信号，自清 0 0：TXFIFO 正常 1：TXFIFO 复位
[2] RFRST	**RXFIFO 复位（RXFIFO RESET）** RXFIFO 复位信号，自清 0 0：RXFIFO 正常 1：RXFIFO 复位
[1] ABORT	**I2C 中止控制位（I2C ABORT）** 0：无 1：中止 I2C 传输 自动清 0
[0] I2CEN	**I2C 使能位（I2C ENABLE）** 使能时，将不能对控制寄存器进行配置。 0：I2C 不使能 1：I2C 使能

23.5.2.2 I2C 控制寄存器（I2C_CTRL）

- **名称：I2C Control Register**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:13	12	11	10	9	8	7:0
名称		PECBYTE	REPEAT	RESTART	STOP	DIRECT	I2CCNT
访问		R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[12] PECBYTE	**SMBUS PEC 模式（SMBUS PECBYTE Mode）** 主机：发送 PEC 字节，STOP 位为 0 时该位无效 0：STOP 之前不发送 PEC 字节 1：STOP 之前发送 PEC 字节 从机：在 I2CCNT 个数据之后进入到 PEC 模式，发送或者接收 PEC 0：从机不发送或者不接收 PEC 1：从机发送或者接收 PEC
[11] REPEAT	**I2C 重复模式（I2C Repeat Mode）** 只在 RESTART 为 1 的时候有效，RESTART 模式 0：只有在 I2CCNT 的第一个数发 RESTART 1：在 I2CCNT 的每一个数都发 RESTART
[10] RESTART	**I2C RESTART 配置（I2C RESTART Configuration）** 该位控制在发送或接收数据之前是否发出 RESTART。 0：则在发送/接收数据之前（根据 DIRECT 的值）将不发出 RESTART。 1：则在发送/接收数据之前（根据 DIRECT 的值）将发出 RESTART。
[9] STOP	**I2C STOP 配置（I2C STOP Configuration）** 该位控制在发送或接收了数据之后是否发出 STOP。 0：发送/接收完第 I2CCNT 个数据之后不发出 STOP。主机将会 SCL 线保持为低电平并停止总线，直到 I2CCNT 的值刷新。 1：发送/接收完第 I2CCNT 个数据之后发出 STOP。
[8] DIRECT	**I2C 传输方向配置（I2C Direct Configuration）** 该位控制是执行读操作还是写操作。作为从机时，该位不控制方向。作为主机时，它仅控制方向。 0：主机 TX 1：主机 RX
[7:0] I2CCNT	**I2C 发送/接收数据量（I2C TX/RX Data Count）** I2C 数据计数寄存器，写入当前要发送或者接收的数据量

23.5.2.3 I2C 波特率寄存器（I2C_BAUD）

- 名称：I2C Baud Rate Register
- 偏移地址：0x08
- 默认值：0x00000F0F
- 返回寄存器列表

位	31:16	15:8	7:0
名称		SCLHCNT	SCLLCNT
访问		R/W	R/W
复位值		0xf	0xf

字段	说明
[15:8] SCLHCNT	**SCL 高电平计数（SCL High Count）** SCL 高电平持续的时长（系统时钟周期为单位），实际值是配置值+1，最小配置 0xf
[7:0] SCLLCNT	**SCL 低电平计数（SCL Low Count）** SCL 低电平持续的时长（系统时钟周期为单位），实际值是配置值+1，最小配置 0xf

23.5.2.4 I2C FIFO 控制寄存器（I2C_FIFOCTL）

- 名称：I2C FIFO Control Register
- 偏移地址：0x0C
- 默认值：0x00000500
- 返回寄存器列表

位	31:29	28:24	23:21	20:16	15:12	11
名称		RXFLR		TXFLR		RFF
访问		R		R		R
复位值		0x0		0x0		0x0
位	10	9	8	7:4	3:0	
名称	RFE	TFF	TFE	RXFTLR	TXFTLR	
访问	R	R	R	R/W	R/W	
复位值	0x1	0x0	0x1	0x0	0x0	

字段	说明
[28:24] RXFLR	**RXFIFO 实时剩余数据量（RXFIFO Level Register）**
[20:16] TXFLR	**TXFIFO 实时剩余数据量（TXFIFO Level Register）**

[11] RFF	**接收 FIFO 满标志（RXFIFO Full Flag）** 0：接收 FIFO 未满 1：接收 FIFO 已满 当 RX FIFO 不再满时，该位被清除。
[10] RFE	**接收 FIFO 空标志（RXFIFO Empty Flag）** 0：接收 FIFO 不为空 1：接收 FIFO 为空 当 RX FIFO 不为空时，该位被清除。
[9] TFF	**发送 FIFO 满标志（TXFIFO Full Flag）** 0：发送 FIFO 未满 1：发送 FIFO 为满 当 TX FIFO 不再满时，该位被清除。
[8] TFE	**发送 FIFO 空标志（TXFIFO Empty Flag）** 0：发送 FIFO 不为空 1：发送 FIFO 为空 当 TX FIFO 不为空时，该位被清除。
[7:4] RXFTLR	**RXFIFO 满中断触发的预设值。** 4'b0000：FIFO 中有 1 个或 1 个以上数据就会触发 RXFIFO 满中断 4'b0001：FIFO 中有 2 个或 2 个以上数据就会触发 RXFIFO 满中断 4'b0010：FIFO 中有 3 个或 3 个以上数据就会触发 RXFIFO 满中断 4'b0011：FIFO 中有 4 个或 4 个以上数据就会触发 RXFIFO 满中断 4'b0100：FIFO 中有 5 个或 5 个以上数据就会触发 RXFIFO 满中断 4'b0101：FIFO 中有 6 个或 6 个以上数据就会触发 RXFIFO 满中断 4'b0110：FIFO 中有 7 个或 7 个以上数据就会触发 RXFIFO 满中断 4'b0111：FIFO 中有 8 个或 8 个以上数据就会触发 RXFIFO 满中断 4'b1000：FIFO 中有 9 个或 9 个以上数据就会触发 RXFIFO 满中断 4'b1001：FIFO 中有 10 个或 10 个以上数据就会触发 RXFIFO 满中断 4'b1010：FIFO 中有 11 个或 11 个以上数据就会触发 RXFIFO 满中断 4'b1011：FIFO 中有 12 个或 12 个以上数据就会触发 RXFIFO 满中断 4'b1100：FIFO 中有 13 个或 13 个以上数据就会触发 RXFIFO 满中断 4'b1101：FIFO 中有 14 个或 14 个以上数据就会触发 RXFIFO 满中断 4'b1110：FIFO 中有 15 个或 15 个以上数据就会触发 RXFIFO 满中断 4'b1111：FIFO 中有 16 个数据就会触发 RXFIFO 满中断

[3:0]	**TXFIFO 空中断触发的预设值。**
TXFTLR	4'b0000：FIFO 中有 0 个就会触发 TXFIFO 空中断
	4'b0001：FIFO 中有 1 个或 1 个以下就会触发 TXFIFO 空中断
	4'b0010：FIFO 中有 2 个或 2 个以下就会触发 TXFIFO 空中断
	4'b0011：FIFO 中有 3 个或 3 个以下就会触发 TXFIFO 空中断
	4'b0100：FIFO 中有 4 个或 4 个以下就会触发 TXFIFO 空中断
	4'b0101：FIFO 中有 5 个或 5 个以下就会触发 TXFIFO 空中断
	4'b0110：FIFO 中有 6 个或 6 个以下就会触发 TXFIFO 空中断
	4'b0111：FIFO 中有 7 个或 7 个以下就会触发 TXFIFO 空中断
	4'b1000：FIFO 中有 8 个或 8 个以下就会触发 TXFIFO 空中断
	4'b1001：FIFO 中有 9 个或 9 个以下就会触发 TXFIFO 空中断
	4'b1010：FIFO 中有 10 个或 10 个以下就会触发 TXFIFO 空中断
	4'b1011：FIFO 中有 11 个或 11 个以下就会触发 TXFIFO 空中断
	4'b1100：FIFO 中有 12 个或 12 个以下就会触发 TXFIFO 空中断
	4'b1101：FIFO 中有 13 个或 13 个以下就会触发 TXFIFO 空中断
	4'b1110：FIFO 中有 14 个或 14 个以下就会触发 TXFIFO 空中断
	4'b1111：FIFO 中有 15 个或 15 个以下就会触发 TXFIFO 空中断

23.5.2.5 I2C 数据读写寄存器（I2C_DATA）

- **名称：I2C Data Write/Read Register**
- **偏移地址：0x10**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:8	7:0
名称		DATA
访问		R/W
复位值		0x00

字段	说明
[7:0]	**I2C 数据读写寄存器（I2C Data Write/Read Register）**
DATA	W：往 FIFO 写要发送的数据
	R：读取从外界收到的并已经存到 FIFO 中的数据）

23.5.2.6 I2C 中断使能寄存器（I2C_INTREN）

- 名称：I2C Interrupt Enable Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:16		15	14	13	12	11	10	9
名称			MTXAIE	RXGCIE	PECRXIE	SEXTOIE	MEXTOIE	ALDETIE	NACKIE
访问			R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值			0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	8	7	6	5	4	3	2	1	0
名称	MOHIE	SPDETIE	STDETIE	RXADIE	TXADIE	MDEIE	RXOFIE	TXEIE	RXFIE
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[15] MTXAIE	**MTXAI 中断使能（Master TX Adderss Done Interrupt Enable）** 0：不使能 1：使能
[14] RXGCIE	**RXGCI 中断使能（Slave RX General Call Interrupt Enable）** 0：不使能 1：使能
[13] PECRXIE	**PECRXI 中断使能（Master or Slave RX PEC Data Interrupt Enable）** 0：不使能 1：使能
[12] SEXTOIE	**SEXTOI 中断使能（Slave Tsext Timeout Interrupt Enable）** 0：不使能 1：使能
[11] MEXTOIE	**MEXTOI 中断使能（Master Tmext Timeout Interrupt Enable）** 0：不使能 1：使能
[10] ALDETIE	**ALDETI 中断使能（Master Detect Alert Signal Interrupt Enable）** 0：不使能 1：使能
[9] NACKIE	**NACKI 中断使能（Master or Slave RX NACK Interrupt Enable）** 0：不使能 1：使能
[8] MOHIE	**MOHI 中断使能（Master On Hold Interrupt Enable）** 0：不使能 1：使能

[7] SPDETIE	**SPDETI 中断使能（Master or Slave Detect STOP Interrupt Enable）** 0：不使能 1：使能
[6] STDETIE	**STDETI 中断使能（Master or Slave Detect START Interrupt Enable）** 0：不使能 1：使能
[5] RXADIE	**RXADI 中断使能（Slave RX Address And Command is SLAVE RX Interrupt Enable）** 0：不使能 1：使能
[4] TXADIE	**TXADI 中断使能（Slave RX Address And Command is SLAVE TX Interrupt Enable）** 0：不使能 1：使能
[3] MDEIE	**MDEI 中断使能（Master TX/RX Done Interrupt Enable）** 0：不使能 1：使能
[2] RXOFIE	**RXOFI 中断使能（RXFIFO Overflow Interrupt Enable）** 0：不使能 1：使能
[1] TXEIE	**TXEI 中断使能（TXFIFO Empty Interrupt Enable）** 0：不使能 1：使能
[0] RXFIE	**RXFI 中断使能（RXFIFO Full Interrupt Enable）** 0：不使能 1：使能

23.5.2.7 I2C 中断寄存器（I2C_INTR）

- **名称：I2C Interrupt Register**
- **偏移地址：0x18**
- **默认值：0x00000002**
- 返回寄存器列表

位	31:16		15	14	13	12	11	10	9
名称			MTXAI	RXGCI	PECRXI	SEXTOI	MEXTOI	ALDETI	NACKI
访问			R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值			0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	8	7	6	5	4	3	2	1	0
名称	MOHI	SPDETI	STDETI	RXADI	TXADI	MDEI	RXOFI	TXEI	RXFI
访问	R	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R	R
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x0

字段	说明
[15] MTXAI	**主机发送地址完成中断（Master TX Adderss Done Interrupt）** 作为主机完成了发送地址（无论收到 ACK 还是 NACK 都会起来） 0：无 1：发送地址完毕 对该位写 1 将清除 MTXADDRI 中断
[14] RXGCI	**从机收到广播中断（Slave RX General Call Interrupt）** 作为从机收到了 General Call 地址 0：无 1：收到了 General Call 地址 对该位写 1 将清除 RXGCI 中断
[13] PECRXI	**主从收到 PEC 中断（Master or Slave RX PEC Data Interrupt）** 收到 PEC 数据 0：未收到 PEC 数据 1：收到 PEC 数据 对该位写 1 将清除 PECRXI 中断
[12] SEXTOI	**从机 Tsext 超时中断（Slave Tsext Timeout Interrupt）** 作为从机，START->ACK、ACK->ACK 和 ACK->STOP 的累计时间 Tlow:sext 超时了 0：未超时 1：超时 对该位写 1 将清除 SEXTTOI
[11] MEXTOI	**主机 Tmext 超时中断（Master Tmext Timeout Interrupt）** 作为主机，START->STOP 的累计时间 Tlow:mext 超时了 0：未超时 1：超时 对该位写 1 将清除 MEXTTOI 中断
[10] ALDETI	**主机检测 Alert 中断（Master Detect Alert Signal Interrupt）** 作为主机检测是否收到了从机发出的 ALERT 信号 0：未收到 ALERT 信号 1：收到 ALERT 信号 对该位写 1 将清除 ALERTDETI 中断
[9] NACKI	**主从收到 NACK 中断（Master or Slave RX NACK Interrupt）** I2C 检测收到了 NACK 0：未收到 NACK 1：收到 NACK 对该位写 1 将清除 NACKI 中断
[8] MOHI	**主机挂起中断（Master On Hold Interrupt）** I2C 作为主机且当前没有数据可以发送和接收，也没有 STOP 位的产生也就是 SCL 为低，主机处于挂起状态 0：主机处于其他状态 1：主机处于挂起状态 I2C_EN 关闭、主机发送或者接收数据、检测到 STOP 位都会清除 MSTONHOLDI 中断

[7] SPDETI	**主从检测 STOP 中断（Master or Slave Detect STOP Interrupt）** I2C 检测到了 STOP 位 0：未检测到 STOP 位 1：检测到了 STOP 位 对该位写 1 将清除 STOPDETI 中断
[6] STDETI	**主从检测 START 中断（Master or Slave Detect START Interrupt）** I2C 检测到了 START 位 0：未检测到 START 位 1：检测到了 START 位 对该位写 1 将清除 STARTDETI 中断
[5] RXADI	**从机收到地址且命令是从机 RX（Slave RX Address And Command is SLAVE RX Interrupt）** I2C 作为从机时接收到了完整的 10bit 地址或者 7bit 地址并且此时的命令是从机接收 0：地址未接收完毕且命令是从机接收 1：地址接收完毕且命令是从机接收 对该位写 1 将清除 RXADDRMI 中断
[4] TXADI	**从机收到地址且命令是从机 TX（Slave RX Address And Command is SLAVE TX Interrupt）** I2C 作为从机时接收到了完整的 10bit 地址或者 7bit 地址并且此时的命令是从机发送 0：地址未接收完毕且命令是从机发送 1：地址接收完毕且命令是从机发送 对该位写 1 将清除 TXADDRMI 中断
[3] MDEI	**主机发送接收完成中断（Master TX/RX Done Interrupt）** 主机发送或接受完 I2C_CNT 个数据时起中断（发送会完全发完才起中断） 0：未发送或者接收完毕 1：发送或者接收完毕 对该位写 1 将清除 MSTDONEI 中断
[2] RXOFI	**RXFIFO 溢出中断（RXFIFO Overflow Interrupt）** RXFIFO 溢出错误位。当 RXFIFO 已满并且新数据被接收写入 FIFO 中时，会起溢出错误中断。此时 FIFO 中的数据将保留，而接收的新数据将丢失。 0：无溢出错误 1：溢出错误 对该位写 1 将清除 RXOFI 中断。
[1] TXEI	**TXFIFO 空中断（TXFIFO Empty Interrupt）** 与 TXFTLR 寄存器有关，当 TXFIFO 中的数据量小于或者等于预设值（TXFTLR）时置位。 0：TXFIFO 数据量大于预设值 1：TXFIFO 数据量小于等于预设值 当 TXFIFO 数据量大于预设值时 TXFIFOEI 中断自动清 0
[0] RXFI	**RXFIFO 满中断（RXFIFO Full Interrupt）** 与 RXFTLR 寄存器有关，当 RXFIFO 中的数据量大于或等于预设值（RXFTLR）时置位。 0：RXFIFO 数据量小于预设值 1：RXFIFO 数据量大于等于预设值 当 RXFIFO 数据量小于预设值时 RXFIFOFI 中断自动清 0

23.5.2.8 I2C 发送地址寄存器（I2C_TAR）

- 名称：I2C Target Address Register
- 偏移地址：0x1C
- 默认值：0x00000000
- 返回寄存器列表

位	31:10	9:0
名称		TAR
访问		R/W
复位值		0x0

字段	说明
[9:0]	**主机目标地址寄存器（Master Target Address Register）**
TAR	作为主机时要发送的地址

23.5.2.9 I2C 接收地址寄存器（I2C_SAR）

- 名称：I2C Receive Address Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:10	9:0
名称		SAR
访问		R/W
复位值		0x0

字段	说明
[9:0]	**从机接收地址寄存器（Slave Receive Address Register ）**
SAR	作为从机时和主机通信的设备地址

23.5.2.10 I2C 接收地址寄存器 (I2C_PEC)

- **名称：I2C RX PEC Register**
- **偏移地址：0x24**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:8	7:0
名称		RXPEC
访问		R
复位值		0x0

字段	说明
[7:0] RXPEC	**SMBUS 收到 PEC 数据寄存器（SMBUS RX PEC Data Register ）** 收到的 PEC 值存放在该寄存器，当 PECRXI 中断起来时数据才有效 关闭 I2C 使能将会清除数据

23.5.2.11 I2C 时序寄存器（I2C_TIMING）

- **名称：I2C Timing Register**
- **偏移地址：0x28**
- **默认值：0x00001800**
- 返回寄存器列表

位	31:24	23:16	15:8	7:0
名称		RXSUDAT	SUDAT	HDDAT
访问		R/W	R/W	R/W
复位值		0x0	0x18	0x0

字段	说明
[23:16] RXSUDAT	**主从数据建立时间配置寄存器（Data Setup Time Configuration Register）** 数据建立时间配置，也就是主从作为 RX 时数据采样的延迟，单位是系统时钟周期，用于生成 TSU:DAT 时序，实际值是配置值
[15:8] SUDAT	**从机数据建立时间配置寄存器（Slave Data Setup Time Configuration Register）** 从机作为 TX 时，数据建立时间配置，单位是系统时钟周期，用于生成 TSU:DAT 时序，实际值是配置值
[7:0] HDDAT	**主从数据保持时间配置寄存器（Data Hold Time Configuration Register）** 数据保持时间配置，也就是主从作为 TX 时数据准备的延迟，单位是系统时钟周期，用于生成 THD:DAT 时序，实际值就是配置值

23.5.2.12 I2C 超时寄存器（I2C_TIMEOUT）

- 名称：I2C Timeout Register
- 偏移地址：0x2c
- 默认值：0x07A2004F
- 返回寄存器列表

位	31:28	27:16	15:12	11:0
名称		SEXTTO		MEXTTO
访问		R/W		R/W
复位值		0x7a2		0x4f

字段	说明
[27:16] SEXTTO	**从机 Tsext 时序配置寄存器（Slave Tsext Timing Configuration Register）** 作为从机时，用于生成 TLOW:SEXT 时序，实际值=（配置值+1）*256
[11:0] MEXTTO	**主机 Tmext 时序配置寄存器（Master Tmext Timing Configuration Register）** 作为主机时，用于生成 TLOW:MEXT 时序，实际值=（配置值+1）*256

23.5.2.13 I2C 状态寄存器（I2C_STATUS）

- 名称：I2C Status Register
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:11	10	9	8	7	6
名称		MGCERR	M7BNACK	M10B1NACK	M10B2NACK	MDNACK
访问		R	R	R	R	R
复位值		0x0	0x0	0x0	0x0	0x0
位	5	4	3	2	1	0
名称	MPECNACK	S7BNMTC	S10B1NMTC	S10B2NMTC	SDNACK	SPECNACK
访问	R	R	R	R	R	R
复位值	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[10] MGCERR	**主机发送广播错误（Master TX General Call Error Status）** I2C 作为主机时，正要发送 GENERAL CALL 时，DIRECT 设置的为 1，和协议不符 当 I2C 使能关闭时清 0

[9] M7BNACK	**主机发送 7BIT 地址但收到 NACK（Master TX 7BIT Adderess But RX NACK Status）** I2C 作为主机时，7bit 模式下，发完地址之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[8] M10B1NACK	**主机发送第一个 10BIT 地址但收到 NACK（Master TX First 10BIT Address But RX NACK Status）** I2C 作为主机时，10bit 模式下，发完 10bit 的第一个地址之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[7] M10B2NACK	**主机发送第二个 10BIT 地址但收到 NACK（Master TX Second 10BIT Address But RX NACK Status）** I2C 作为主机时，10bit 模式下，发完 10bit 的第二个地址之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[6] MDNACK	**主机发送数据但收到 NACK（Master TX Data But RX NACK Status）** I2C 作为主机时，发完数据之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[5] MPECNACK	**主机发送 PEC 数据但收到 NACK（Master TX PEC Data But RX NACK Status）** I2C 作为主机时，发完 PEC 之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[4] S7BNMTC	**从机收到 7BIT 地址但不匹配（Slave RX 7BIT Address But No Match Status）** I2C 作为从机时，7bit 模式下，收到的 7bit 地址和 I2C_SAR 不匹配 当 I2CC 使能关闭时清 0
[3] S10B1NMTC	**从机收到第一个 10BIT 地址但不匹配（Slave RX First 10BIT Address But No Match Status）** I2C 作为从机时，10bit 模式下，收到的 10bit 的第一个地址和 I2C_SAR 不匹配 当 I2CC 使能关闭时清 0
[2] S10B2NMTC	**从机收到第二个 10BIT 地址但不匹配（Slave RX Second 10BIT Address But No Match Status）** I2C 作为从机时，10bit 模式下，收到的 10bit 的第二个地址和 I2C_SAR 不匹配 当 I2CC 使能关闭时清 0
[1] SDNACK	**从机发送数据但收到 NACK（Slave TX Data But RX NACK Status）** I2C 作为从机时，发完数据之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除
[0] SPECNACK	**从机发送 PEC 数据但收到 NACK（Slave TX PEC Data But RX NACK Status）** I2C 作为从机时，发完 PEC 之后收到了是 NACK 而不是 ACK 当 NACKI 中断被清除时，该位也会被清除

24 串行外设接口（SPI）

24.1 简介

串行外围设备接口（SPI），它是一个高速同步串行输入和输出（I / O）端口，SPI 允许 MCU 与外部设备以全双工、同步、串行方式通信，软件也可以通过查询状态或者中断来通信。

24.2 结构框图

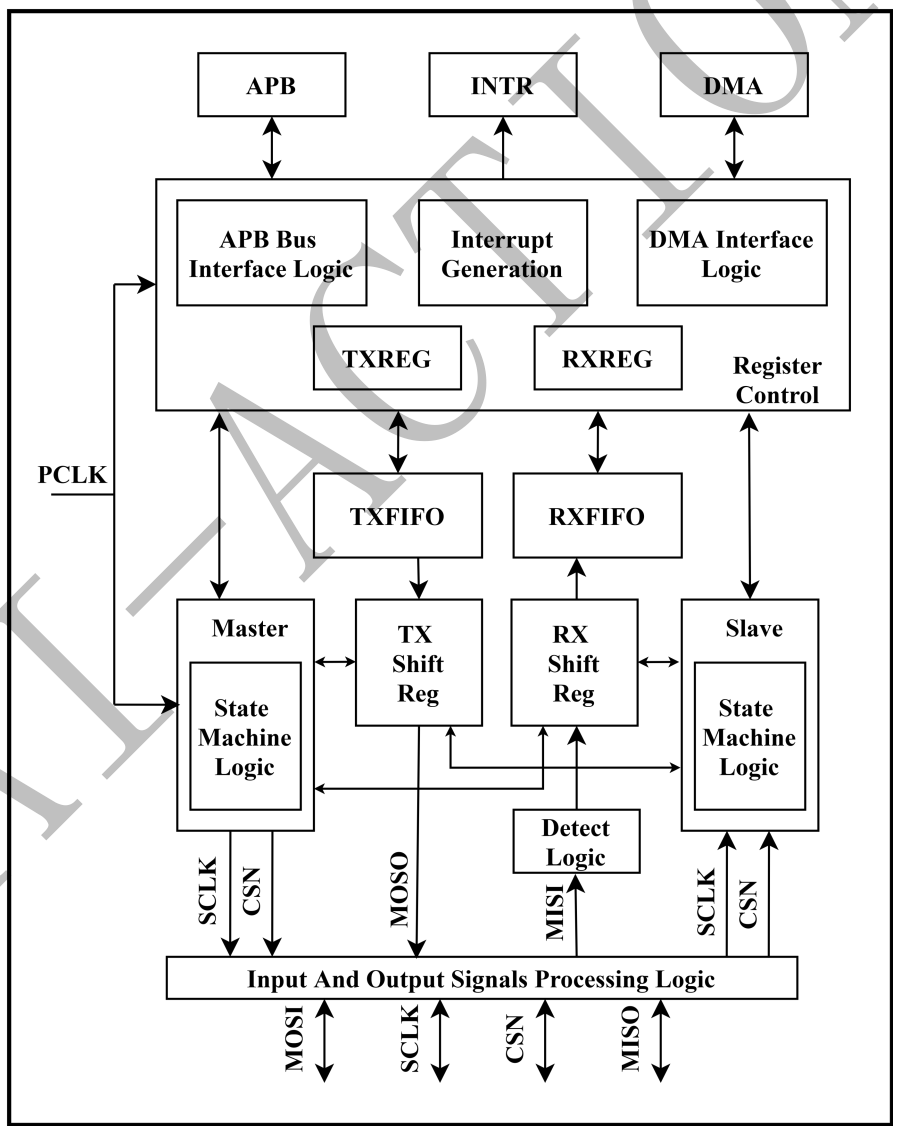


图 24-1 SPI 结构框图

24.3 主要特性

SPI 的主要功能如下：

- 完全兼容 Motorola 的 SPI 规格
- 在 4 根线下支持全双工同步传输
- 在 3 根线下支持半双工同步传输
- 支持主机模式和从机模式
- 分别有深度为 16 的 RXFIFO 和 TXFIFO
- 主从最高通信速度为 20MHz
- 可编程的时钟极性和相位
- 支持 2~16 位的数据位长度同时发送和接收
- 支持 DMA 模式
- 支持中断或者轮询方式
- 支持主机 RX 的 delay chain 配置

24.4 功能描述

24.4.1 SPI 协议

24.4.1.1 SPI 电气结构

SPI 的通信原理很简单，它以主从方式工作，这种模式通常有一个主设备和一个或多个从设备，需要至少 4 根线，事实上 3 根也可以(单向传输时)。也是所有基于 SPI 的设备共有的，它们是 MOSI(数据线)、MISO(数据线)、SCLK(时钟)、CS(片选)。

1. MOSI：主设备数据输出，从设备数据输入
2. MISO：主设备数据输入，从设备数据输出
3. SCLK：时钟信号，由主设备产生
4. CS：从设备使能信号，由主设备控制。当有多个从设备的时候，因为每个从设备上都有一个片选引脚接入到主设备机中，当我们的主设备和某个从设备通信时将需要将从设备对应的片选引脚电平拉低或者是拉高。

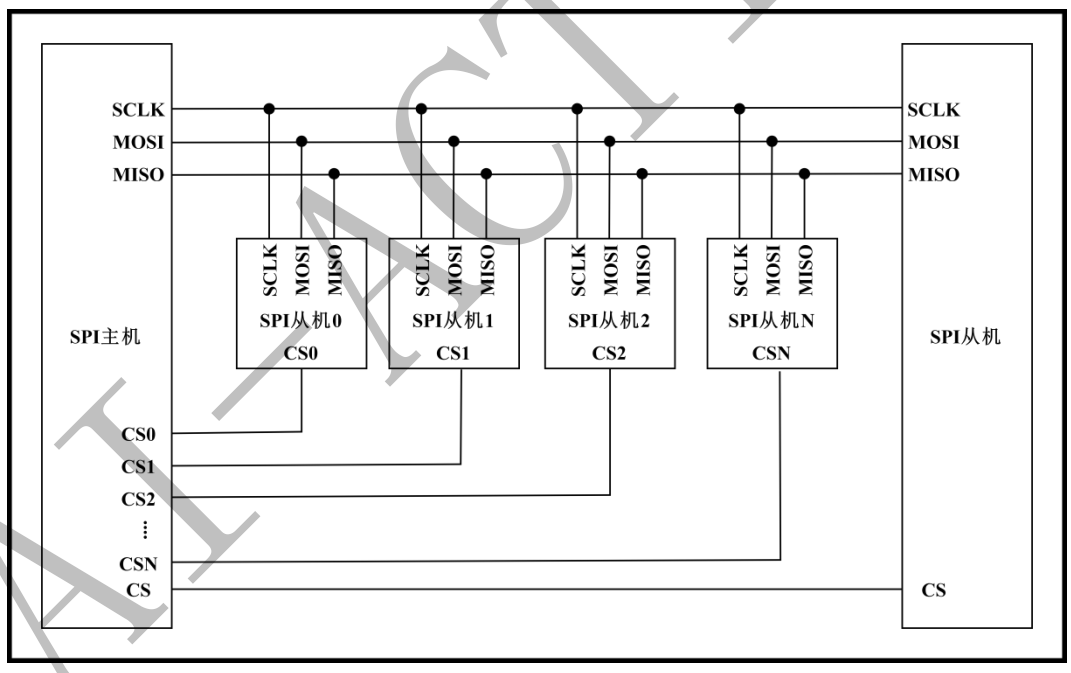


图 24-2　SPI 主从拓扑图

24.4.1.2 SPI 的 4 种标准模式

SPI 通信有 4 种不同的模式，不同的从设备可能在出厂是就是配置为某种模式，这是不能改变的；但我们的通信双方必须是工作在同一模式下，所以我们可以对我们的主设备的 SPI 模式进行配置，通过 CPOL（时钟极性）和 CPHA（时钟相位）来控制我们主设备的通信模式，具体如下：

- Mode0：CPOL=0，CPHA=0
- Mode1：CPOL=0，CPHA=1

- Mode2：CPOL=1，CPHA=0
- Mode3：CPOL=1，CPHA=1

时钟极性 CPOL 是用来配置 SCLK 的电平出于哪种状态时是空闲态或者有效态，时钟相位 CPHA 是用来配置数据采样是在第几个边沿：

- CPOL=0，表示当 SCLK=0 时处于空闲态，所以有效状态就是 SCLK 处于高电平时
- CPOL=1，表示当 SCLK=1 时处于空闲态，所以有效状态就是 SCLK 处于低电平时
- CPHA=0，表示数据采样是在第 1 个边沿，数据发送在第 2 个边沿
- CPHA=1，表示数据采样是在第 2 个边沿，数据发送在第 1 个边沿

例如：

- CPOL=0，CPHA=0：此时空闲态时，SCLK 处于低电平，数据采样是在第 1 个边沿，也就是 SCLK 由低电平到高电平的跳变，所以数据采样是在上升沿，数据发送是在下降沿。
- CPOL=0，CPHA=1：此时空闲态时，SCLK 处于低电平，数据发送是在第 1 个边沿，也就是 SCLK 由低电平到高电平的跳变，所以数据采样是在下降沿，数据发送是在上升沿。
- CPOL=1，CPHA=0：此时空闲态时，SCLK 处于高电平，数据采集是在第 1 个边沿，也就是 SCLK 由高电平到低电平的跳变，所以数据采集是在下降沿，数据发送是在上升沿。
- CPOL=1，CPHA=1：此时空闲态时，SCLK 处于高电平，数据发送是在第 1 个边沿，也就是 SCLK 由高电平到低电平的跳变，所以数据采集是在上升沿，数据发送是在下降沿。

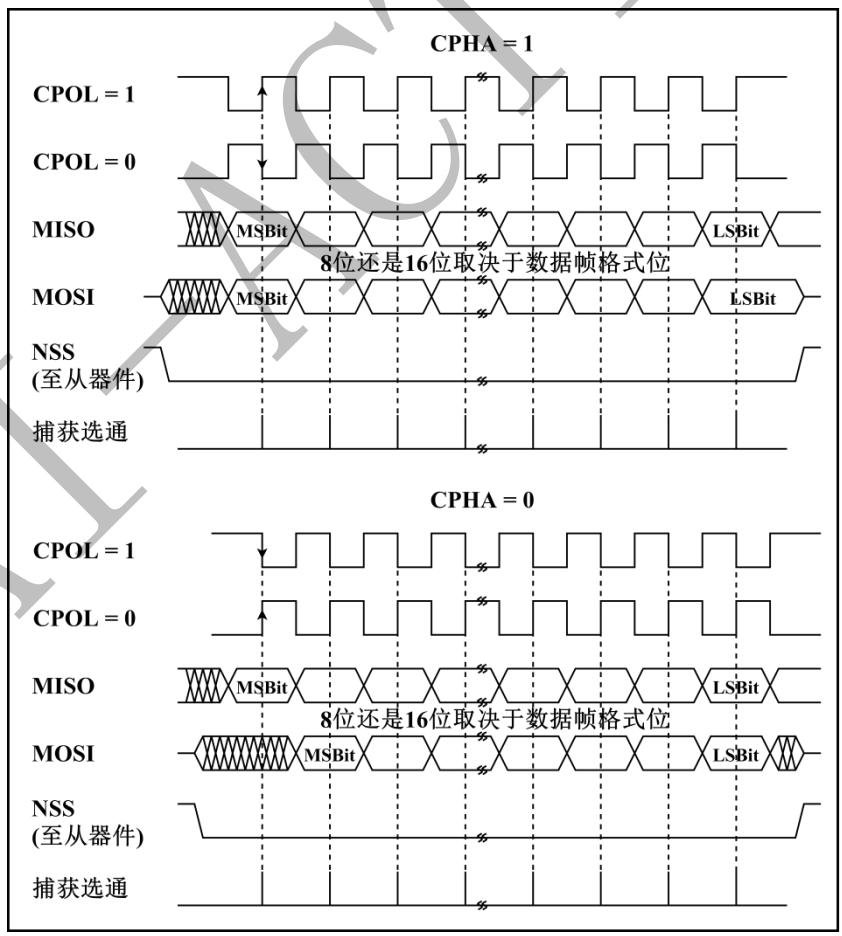


图 24-3　SPI 四种模式

24.4.2 SPI 主设备

将 SPI_CTRL[0]设置为 1 时，SPI 作为主设备。作为主设备时，SPI 将控制所有串行传输。

24.4.2.1 波特率

SPI 中只有一个系统时钟，通过系统时钟来做分频时钟，寄存器是 SPI_BAUD[15:0]，配置波特率可以配置成 2~65534 中间的偶数，奇数配置时硬件会自动把第 0 位置为 0，也就是配置为 3 时，实际上配置的是 2，波特率的公式如下

$$Fsclk = \frac{Fpclk}{BAUD}$$

需要注意的是，如果 BAUD 设置为 0，那么 SPI 将会进入不可使用的状态。

24.4.2.2 传输模式

SPI 总共有三种传输模式，配置的寄存器为 SPI_CTRL[9:8]的 SPI_MODE，可以配置为发送和接收模式（SPI_MODE=2’b11）、仅发送模式（SPI_MODE=2’b01）和仅接收模式（SPI_MODE=2’b10）。

以下为寄存器操作模式

1. 将 SPI_ENABLE 的 BIT0 写 0 关闭 SPI，并且配置 BIT1 的 TXFIFO 复位和 BIT2 的 RXFIFO 复位，均自清 0。
2. 根据当前需求配置 SPI_ENABLE 中的 BIT4，配置是否需要采用四线模式。
3. 根据以下列表配置 SPI，可以以任何顺序配置
   a) SPI_CTRL [0]的 MSTEN 设置为 1；
   b) SPI_CTRL [2:1]配置串行时钟极性和相位，对于 master 来说，必须将串行时钟极性和相位设置为和 slave 一致
   c) 如果是仅接收模式（SPI_MODE==2’b10），需要设置 SPI_RXCNT [15:0]的 SPI_RXCNT 来告知硬件当前 SPI 启动后需要向从机接收的总数据量。
   d) 设置波特率
   e) 根据需求配置 SPI_INTREN 中的中断使能，开启或者关闭哪些中断
   f) 设置 FIFO 阈值，SPI_FIFOCTRL[3:0]的 TXFTLR 和 SPI_FIFOCTRL[7:4]的 RXFTLR
   g) 如果需要 DMA 传输，则配置 SPI_ENABLE[8]的 DMARXEN 和 SPI_ENABLE[9]的 DMATXEN
4. 对 SPI_ENABLE 的 BIT0 写 1 启动 SPI 传输
5. 若为仅接受模式，则等待中断或者状态位去读取 FIFO 即可。
6. 若为仅发送模式，将需要发送的数据写入 SPI_DATA 寄存器中去
7. 若为发送和接收模式，将需要发送的数据写入 SPI_DATA 寄存器 ，同时根据中断或者状态位去读取 FIFO。
8. 完成之后对 SPI_ENABLE 的 BIT0 写 0 以关闭 SPI。

24.4.2.3 RX Delay Chain

作为主机，在跑高速 SPI RX 时，主从之间可能存在一些不可避免的延时逻辑，标准的 SPI 采样可能也会采样不到数据，这时需要引入 RX Delay Chain，通过将采样时钟往后延时的方式进行采样，提高采样的准确性。

主机发送的 sclk_out 信号和从从机接收的 rxd 数据信号的往返路由延迟可能意味着 rxd 信号的时序（如主机所见）已偏离正常采样时间。图 24-4 说明了这种情况。

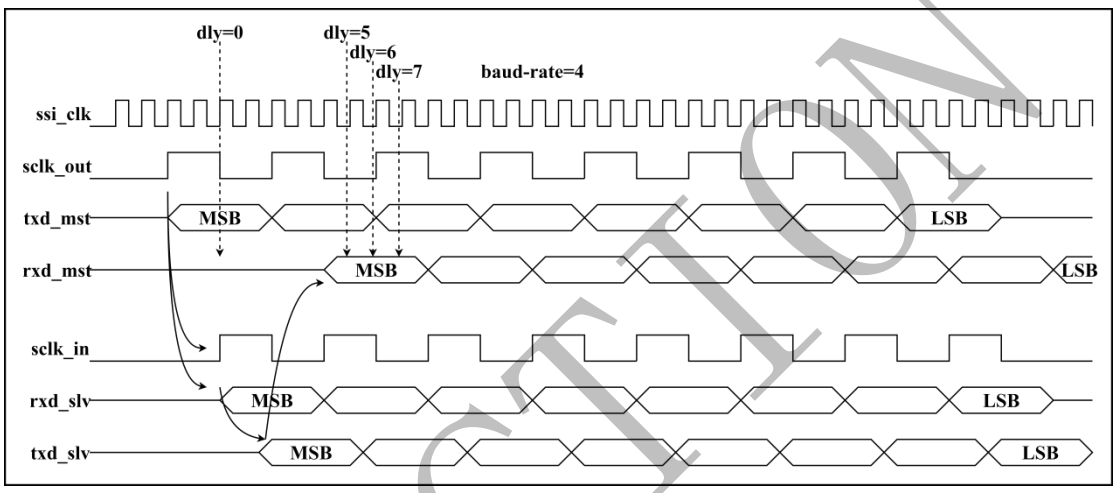


图 24-4　主机 RX 时序

从设备对 sclk_out 信号进行采样并将数据驱动到串行总线上，实际上这已经可能意味着主机对 rxd 信号进行采样之前，rxd 尚未稳定到正确的值。图 24-4 显示了一个示例，该示例显示了当主机对端口进行采样时，高速 rxd 信号上的延迟会导致主机得到一个错误的 rxd 值。因此如果没有采样延迟逻辑，用户必须提高传输的波特率，SPI 将无法跑高速，这样子去保证 rxd 信号的建立时间在范围内，降低了串行接口的频率。

对此，在 SPI 中可以配置寄存器 SPI_CTRL[18:16]的 RX_DEDLY 寄存器，配置为 0 代表零延时，配置为 1 代表采样时钟较 sclk_out 延时一个系统时钟，最大支持延时 7 个系统时钟。

24.4.3 SPI 从设备

将 SPI_CTRL[0]设置为 0 时，SPI 作为从设备。作为从设备时，SPI 所有串行传输均由主机启动和控制。

作为从机，SPI 操作模式：

1. 将 SPI_ENABLE 的 BIT0 写 0 关闭 SPI，并且配置 BIT1 的 TXFIFO 复位和 BIT2 的 RXFIFO 复位，均自清 0。
2. 根据当前需求配置 SPI_ENABLE 中的 BIT4，配置是否需要采用四线模式。
3. 根据以下列表配置 SPI，可以以任何顺序配置
   a) SPI_CTRL [0]的 MSTEN 设置为 0；

b) SPI_CTRL [1:0]配置串行时钟极性和相位，必须将串行时钟极性和相位设置为和 master 一致
c) 根据需求配置 SPI_INTREN 中的中断使能，开启或者关闭哪些中断
d) 设置 FIFO 阈值，SPI_FIFOCTRL[3:0]的 TXFTLR 和 SPI_FIFOCTRL[7:4]的 RXFTLR
e) 如果需要 DMA 传输，则配置 SPI_ENABLE[8]的 DMARXEN 和 SPI_ENABLE[9]的 DMATXEN

4. 对 SPI_ENABLE 的 BIT0 写 1 启动 SPI 传输
5. 若为接收模式，则等待中断或者状态位去读取 FIFO 即可。
6. 若为发送模式，将需要发送的数据写入 DATA 寄存器中去
7. 完成之后对 SPI_ENABLE 的 BIT0 写 0 以关闭 SPI。

24.4.4 DMA 模式

UART 接口支持使用 DMA 来发送和接收数据。通过设置 SPI_ENABLE[9:8]位可以单独开启 DMA 发送或者接收。发送时 RXFIFO 变满（RXFLTR 有关）或者 TXFIFO 变空（TXFLTR 有关）时，会产生 DMA 请求。DMA 控制器会将数据从 SPI_DATA 寄存器中传送到预置的存储区或者将数据从预置的存储区装载到 SPI_DATA 寄存器中去。

24.5 SPI 寄存器描述

24.5.1 寄存器列表

SPI0 基地址：**0x4001 0000**
SPI1 基地址：**0x4001 1000**

偏移	实例地址	名称	默认值	描述
0x00	0x40010000	SPI_ENABLE	0x00002000	SPI 使能寄存器
0x04	0x40010004	SPI_CTRL	0x00000000	SPI 控制寄存器
0x08	0x40010008	SPI_BAUD	0x00000000	SPI 波特率寄存器
0x0c	0x4001000C	SPI_FIFOCTL	0x00000500	SPI FIFO 控制寄存器
0x10	0x40010010	SPI_DATA	0x00000000	SPI 数据读写寄存器
0x14	0x40010014	SPI_INTREN	0x00000000	SPI 中断使能寄存器
0x18	0x40010018	SPI_INTR	0x00000000	SPI 中断寄存器
0x1c	0x4001001C	SPI_RXCNT	0x00000000	SPI 接收数据数量寄存器

【说明】上表中，x=0，1。

24.5.2 寄存器详细描述

24.5.2.1 SPI 使能寄存器（SPI_ENABLE）

- 名称：SPI Enable Register
- 偏移地址：0x00
- 默认值：0x00002000
- 返回寄存器列表

位	31:14	13	12	11:10	9	8
名称		SWCS	SWCSEN		DMATXEN	DMARXEN
访问		R/W	R/W		R/W	R/W
复位值		0x1	0x0		0x0	0x0
位	7:5	4	3	2	1	0
名称		CSSEL	SPI3WEN	TFRST	RFRST	SPIEN
访问		R/W	R/W	W	W	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[13] SWCS	**SPI 软件输出（SPI Software CS Output）** 0：CS 输出 0 1：CS 输出 1
[12] SWCSEN	**SPI CS 软件输出使能（SPI Software CS Output Enable）** 0：使用硬件的 CS 输出 1：使用软件的 CS 输出
[9] DMATXEN	**发送 DMA 使能（SPI TX DMA Enable）** 该位启用/禁用发送 FIFO DMA 通道 0：禁用发送 DMA 通道 1：使能发送 DMA 通道
[8] DMARXEN	**接收 DMA 使能（SPI RX DMA Enable）** 该位启用/禁用接收 FIFO DMA 通道 0：禁用接收 DMA 通道 1：使能接收 DMA 通道
[4] CSSEL	**CS 模式选择（SPI CS Mode Select）** 0：只要 FIFO 数据不断 CS 就不会拉高 1：无论 FIFO 中还有没有数据发送或接受的每一个数据发送完毕 CS 都会拉高
[3] SPI3WEN	**SPI 3 线使能（SPI 3Wire Enable）** 0：GPIO 采用 4 线接法和 SPI 交互 1：GPIO 采用 3 线接法和 SPI 交互

[2]	**TXFIFO 复位（TXFIFO RESET）**
TFRST	TXFIFO 复位信号，自清 0
	0：TXFIFO 正常
	1：TXFIFO 复位
[1]	**RXFIFO 复位（RXFIFO RESET）**
RFRST	RXFIFO 复位信号，自清 0
	0：RXFIFO 正常
	1：RXFIFO 复位
[0]	**SPI 使能位（SPI Enable）**
SPIEN	使能时，将不能对控制寄存器进行配置。
	0：SPI 不使能
	1：SPI 使能

24.5.2.2 SPI 控制寄存器（SPI_CTRL）

- **名称：SPI Control Register**
- **偏移地址：0x04**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:19	18:16	15:12	11:8	7:6
名称		RXDLY		LENSEL	
访问		R/W		R/W	
复位值		0x0		0x0	
位	5:4	3	2	1	0
名称	SPIMODE	SLVHZOE	SCPOL	SCPHA	MSTEN
访问	R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0	0x0

字段	说明
[18:16]	**接收数据采样延迟（Master Rx Data Delay Chain ）**
RXDLY	该寄存器用于延迟 rxd 输入信号的采样。系统时钟下的延时
[11:8]	**串行 SPI 数据长度（Serial Data Length Select）**
LENSEL	0：Reserved
	1：2 位串行数据传输
	2：3 位串行数据传输
	3：4 位串行数据传输
	4：5 位串行数据传输
	5：6 位串行数据传输
	……
	f：16 位串行数据传输

[5:4]	**SPI 传输模式（SPI Transmission mode）**
SPIMODE	00：Reserved
	01：仅发送模式
	10：仅接收模式
	11：发送和接收模式
[3]	**从机输出高阻态使能位（Slave High-Z Output Enable ）**
SLVHZOE	仅从机有效，当 SLV_OE 为 1，从机的 txd 将会始终输出高阻态。
	0：从机输出正常
	1：从机输出高阻态
[2]	**串行时钟极性（SPI SCLK Polarity）**
SCPOL	0：串行时钟无效状态为低
	1：串行时钟无效状态为高
[1]	**串行时钟相位（SPI SCLK Phase）**
SCPHA	表示 SPI 在 SCLK 第几个边沿开始采样
	0：SPI 在 SCLK 第 0 个边沿开始采样
	1：SPI 在 SCLK 第 1 个边沿开始采样
[0]	**SPI 的主机使能位（SPI Master Enable）**
MSTEN	0：SPI 作为从机
	1：SPI 作为主机

24.5.2.3 SPI 波特率寄存器（SPI_BAUD）

- **名称：SPI Baud Rate Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		BAUD
访问		R/W
复位值		0x0

字段	说明
[15:0]	**SPI 时钟分频器（SPI Baud Rate）**
BAUD	只能配置偶数。 sclk_out 的频率从以下公式得出：
	Fsclk_out = F 系统时钟频率/ BAUD
	取值范围为：2~65534 之间的偶数

24.5.2.4 SPI FIFO 控制寄存器（SPI_FIFOCTL）

- 名称：SPI FIFO Control Register
- 偏移地址：0x0C
- 默认值：0x00000500
- 返回寄存器列表

位	31:29	28:24	23:21	20:16	15:12	11
名称		RXFLR		TXFLR		RFF
访问		R		R		R
复位值		0x0		0x0		0x0
位	10	9	8	7:4	3:0	
名称	RFE	TFF	TFE	RXFTLR	TXFTLR	
访问	R	R	R	R/W	R/W	
复位值	0x1	0x0	0x1	0x0	0x0	

字段	说明
[28:24] RXFLR	**RXFIFO 实时剩余数据量（RXFIFO Level Register）**
[20:16] TXFLR	**TXFIFO 实时剩余数据量（TXFIFO Level Register）**
[11] RFF	**接收 FIFO 满标志（RXFIFO Full Flag）** 0：接收 FIFO 未满 1：接收 FIFO 已满 当 RX FIFO 不再满时，该位被清除。
[10] RFE	**接收 FIFO 空标志（RXFIFO Empty Flag）** 0：接收 FIFO 不为空 1：接收 FIFO 为空 当 RX FIFO 不为空时，该位被清除。
[9] TFF	**发送 FIFO 满标志（TXFIFO Full Flag）** 0：发送 FIFO 未满 1：发送 FIFO 为满 当 TX FIFO 不再满时，该位被清除。
[8] TFE	**发送 FIFO 空标志（TXFIFO Empty Flag）** 0：发送 FIFO 不为空 1：发送 FIFO 为空 当 TX FIFO 不为空时，该位被清除。

[7:4]	**RXFIFO 满中断触发的预设值。**
RXFTLR	4'b0000：FIFO 中有 1 个或 1 个以上数据就会触发 RXFIFO 满中断
	4'b0001：FIFO 中有 2 个或 2 个以上数据就会触发 RXFIFO 满中断
	4'b0010：FIFO 中有 3 个或 3 个以上数据就会触发 RXFIFO 满中断
	4'b0011：FIFO 中有 4 个或 4 个以上数据就会触发 RXFIFO 满中断
	4'b0100：FIFO 中有 5 个或 5 个以上数据就会触发 RXFIFO 满中断
	4'b0101：FIFO 中有 6 个或 6 个以上数据就会触发 RXFIFO 满中断
	4'b0110：FIFO 中有 7 个或 7 个以上数据就会触发 RXFIFO 满中断
	4'b0111：FIFO 中有 8 个或 8 个以上数据就会触发 RXFIFO 满中断
	4'b1000：FIFO 中有 9 个或 9 个以上数据就会触发 RXFIFO 满中断
	4'b1001：FIFO 中有 10 个或 10 个以上数据就会触发 RXFIFO 满中断
	4'b1010：FIFO 中有 11 个或 11 个以上数据就会触发 RXFIFO 满中断
	4'b1011：FIFO 中有 12 个或 12 个以上数据就会触发 RXFIFO 满中断
	4'b1100：FIFO 中有 13 个或 13 个以上数据就会触发 RXFIFO 满中断
	4'b1101：FIFO 中有 14 个或 14 个以上数据就会触发 RXFIFO 满中断
	4'b1110：FIFO 中有 15 个或 15 个以上数据就会触发 RXFIFO 满中断
	4'b1111：FIFO 中有 16 个数据就会触发 RXFIFO 满中断
[3:0]	**TXFIFO 空中断触发的预设值。**
TXFTLR	4'b0000：FIFO 中有 0 个就会触发 TXFIFO 空中断
	4'b0001：FIFO 中有 1 个或 1 个以下就会触发 TXFIFO 空中断
	4'b0010：FIFO 中有 2 个或 2 个以下就会触发 TXFIFO 空中断
	4'b0011：FIFO 中有 3 个或 3 个以下就会触发 TXFIFO 空中断
	4'b0100：FIFO 中有 4 个或 4 个以下就会触发 TXFIFO 空中断
	4'b0101：FIFO 中有 5 个或 5 个以下就会触发 TXFIFO 空中断
	4'b0110：FIFO 中有 6 个或 6 个以下就会触发 TXFIFO 空中断
	4'b0111：FIFO 中有 7 个或 7 个以下就会触发 TXFIFO 空中断
	4'b1000：FIFO 中有 8 个或 8 个以下就会触发 TXFIFO 空中断
	4'b1001：FIFO 中有 9 个或 9 个以下就会触发 TXFIFO 空中断
	4'b1010：FIFO 中有 10 个或 10 个以下就会触发 TXFIFO 空中断
	4'b1011：FIFO 中有 11 个或 11 个以下就会触发 TXFIFO 空中断
	4'b1100：FIFO 中有 12 个或 12 个以下就会触发 TXFIFO 空中断
	4'b1101：FIFO 中有 13 个或 13 个以下就会触发 TXFIFO 空中断
	4'b1110：FIFO 中有 14 个或 14 个以下就会触发 TXFIFO 空中断
	4'b1111：FIFO 中有 15 个或 15 个以下就会触发 TXFIFO 空中断

24.5.2.5 SPI 数据读写寄存器（SPI_DATA）

- 名称：SPI Data Write/Read Register
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称		DATA
访问		R/W
复位值		0x00

字段	说明
[15:0] DATA	**SPI 数据读写寄存器（SPI Data Write/Read Register）** W：往 FIFO 写要发送的数据 R：读取从外界收到的并已经存到 FIFO 中的数据）

24.5.2.6 SPI 中断使能寄存器（SPI_INTREN）

- 名称：SPI Interrupt Enable Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:5	4	3	2	1	0
名称		RXDEIE	TXDEIE	RXOFIE	TXEIE	RXFIE
访问		R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[4] RXDEIE	**RXDEI 中断使能（SPI RX Done Interrupt Enable）** 0：不使能 1：使能
[3] TXDEIE	**TXDEI 中断使能（SPI TX Done Interrupt Enable）** 0：不使能 1：使能
[2] RXOFIE	**RXOFI 中断使能（RXFIFO Overflow Interrupt Enable）** 0：不使能 1：使能

[1] TXEIE	**TXEI 中断使能（TXFIFO Empty Interrupt Enable）** 0：不使能 1：使能
[0] RXFIE	**RXFI 中断使能（RXFIFO Full Interrupt Enable）** 0：不使能 1：使能

24.5.2.7 SPI 中断寄存器（SPI_INTR）

- **名称：SPI Interrupt Register**
- **偏移地址：0x18**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:5	4	3	2	1	0
名称		RXDEI	TXDEI	RXOFI	TXEI	RXFI
访问		R/W1C	R/W1C	R/W1C	R	R
复位值		0x0	0x0	0x0	0x1	0x0

字段	说明
[4] RXDEI	**SPI 接收完成中断（SPI RX Done Interrupt ）** SPI 作为主机且模式为仅 RX 模式时，当 SPI 接收到 SPI_RXCNT 个数据时会 RXDONEI 中断 0：未接收完毕 1：接收完毕 对该位写 1 将清除 RXDONEI 中断
[3] TXDEI	**SPI 发送完成中断（SPI TX Done Interrupt ）** SPI 在做 TX 时，当完全发送完一个 SPI 帧且 TXFIFO 为空时起中断 0：未发送完毕 1：发送完毕 对该位写 1 将清除 TXDONEI 中断
[2] RXOFI	**RXFIFO 溢出错误位（RXFIFO Overflow Interrupt）** 当 RXFIFO 已满并且新数据被接收写入 FIFO 中时，会起溢出错误中断。此时 FIFO 中的数据将保留，而接收的新数据将丢失。 0：无溢出错误 1：溢出错误 对该位写 1 将清除 RXOFI 中断。
[1] TXEI	**TXFIFO 空中断（TXFIFO Empty Interrupt ）** 与 TXFTLR 寄存器有关，当 TXFIFO 中的数据量小于或者等于预设值（TXFTLR）时置位。 0：TXFIFO 数据量大于预设值 1：TXFIFO 数据量小于等于预设值 当 TXFIFO 数据量大于预设值时 TXFIFOEI 中断自动清 0

[0]	**RXFIFO 满中断（RXFIFO Full Interrupt ）**
RXFI	与 RXFTLR 寄存器有关，当 RXFIFO 中的数据量大于或等于预设值（RXFTLR）时置位。 0：RXFIFO 数据量小于预设值 1：RXFIFO 数据量大于等于预设值 当 RXFIFO 数据量小于预设值时 RXFIFOFI 中断自动清 0

24.5.2.8 SPI 接收数据数量寄存器（SPI_RXCNT）

- **名称：SPI RX Count Register**
- **偏移地址：0x1C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:16	15:0
名称		RXCNT
访问		R/W
复位值		0x0

字段	说明
[15:0]	**SPI 主机接收数据量（SPI RX Count）**
RXCNT	设置 SPI 连续接收的数据帧数。仅当 MSTEN=1 时可用。仅 RX 模式有用

25 通用异步收发器（UART）

25.1 简介

通用异步收发器(UART) 提供了一种灵活的方法与使用工业标准 NRZ 异步串行数据格式的外部设备之间进行全双工数据交换。UART 利用小数波特率发生器提供宽范围的波特率选择。它支持同步单向通信和半双工单线通信。

25.2 结构框图

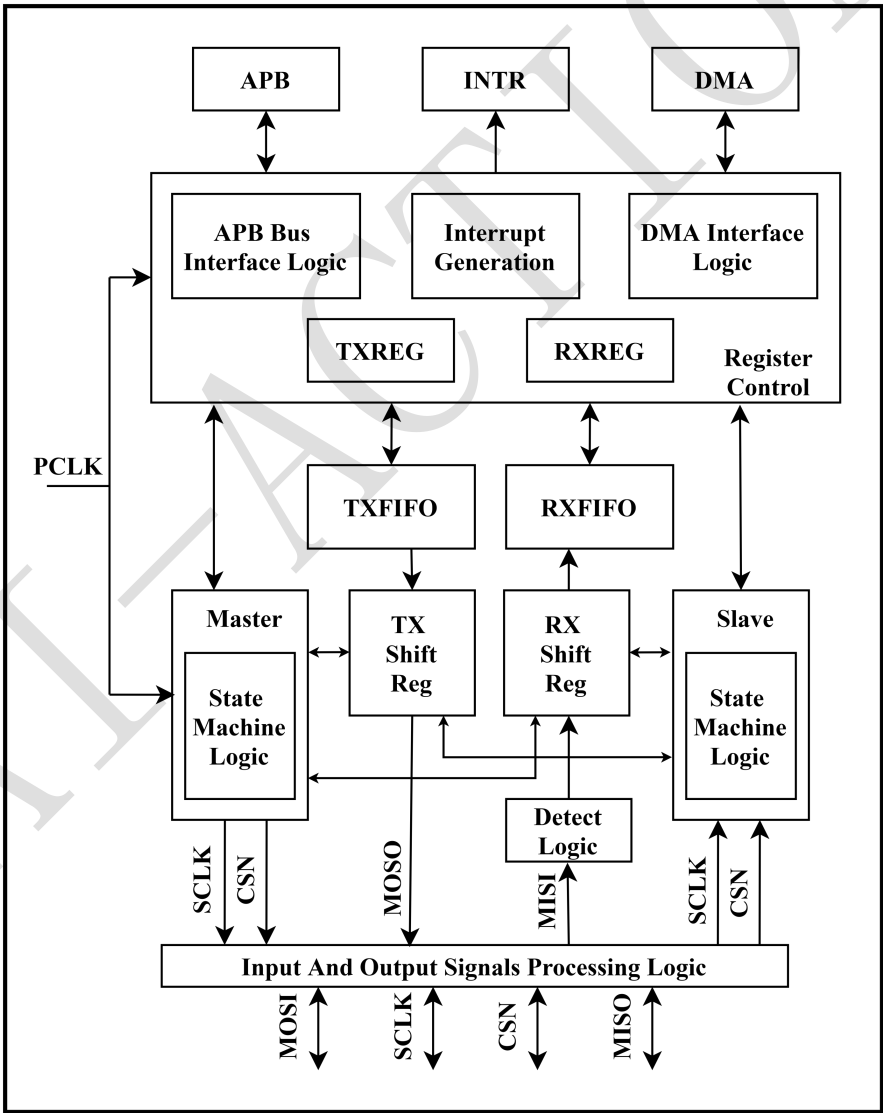


图 25-1 UART 结构框图

25.3 主要特性

UART 的主要功能如下：

- 支持异步方式下 RS-232S 协议，符合工业标准 16550
- 全双工异步操作
- 支持小数波特率
- 支持 5~9 位串行数据传输
- 错误的 stop、start 和 parity 位检测
- 支持 RS485 接口
- 可选的奇偶检验位和停止位数
- 16 位的 FIFO 深度
- 支持 DMA 操作
- 支持中断或轮询操作

25.4 功能描述

25.4.1 UART 协议

由于 UART 与设备间的通信是异步的，在串行数据中要加入两个数据位（start 和 stop）来标志通信的开始和结束。通过这些数据位可以让两个设备间进行同步。

串行数据的格式如图 25-2 所示：

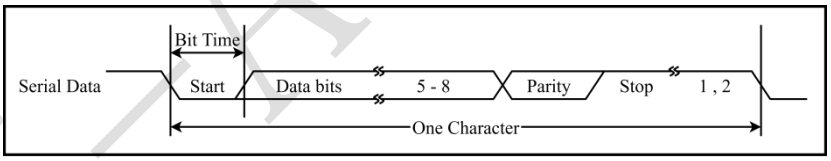


图 25-2　串行数据格式

UART 传输的字符中可以加入校验位，在最后一位数据位之后，位于 STOP 位之前。主要用来对 UART 接收的数据进行简单错误检查。UART 的寄存器（LCR）用于控制串行字符特征。数据位在 START 位之后发出，并以 LSB 位开始。紧接着是可选的校验位，STOP 位（可以为 1/2 位）。UART 传输中，每一位的传输时间都是一样的。每一位的传输时间称为位周期，一个位周期等于 16 个波特时钟周期。

为了保证线上传输的稳定性，接收端检测到 start 位后，在位周期的中点处开始采集输入的数据。因为每个数据位传输时间的波特时钟 cycle 数量是已知的，不难计算数据采集的中点。即 START 位中点后的每 16 个波特时钟 cycle。除了对输入进行去抖动外，这种采样方式也可以避免检测到错误的 START 位。短暂的不稳定信号可以通过去抖动滤掉，线上不会产生传输。如果一个不稳定信号时间长到可以躲过被去抖动滤掉，那么 START 位只有在经过半个位周期再次采样为低时才算检测有效。

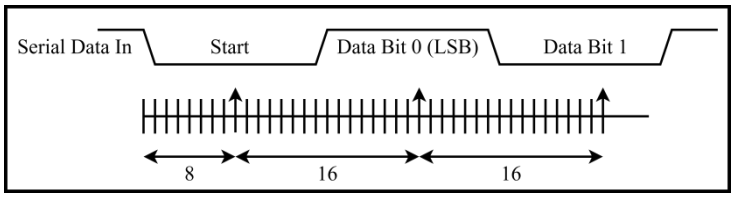


图 25-3　接收数据采样点

UART 的波特率是由寄存器（UART_BAUD[19:4]的 BAUD_INTE 和 UART_BAUD[3:0]的 BAUD_FRAC）控制的。

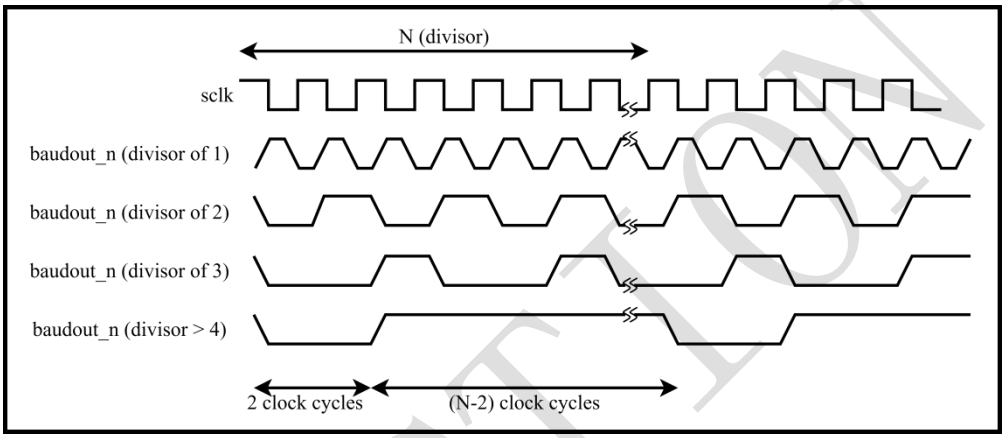


图 25-4　波特率参考时序图

25.4.2　RS485 协议

RS485 协议可以使用 RS485 接口进行数据传输。在使能 RS485 接口支持后，会相应加入驱动使能信号（DE）和接收使能信号（RE）。DE 和 RE 信号是由硬件生成的。

RS485 的配置步骤：

1. 通过接收控制寄存器（UART_CTRL）的 BIT8 来使能或不使能 RS485 模式；
2. UART_CTRL 的 BIT10 用于选择 DE 信号的极性；

25.4.3　9bit 数据传输

在发送和接收模式中，UART 都可以配置成 9bit 数据传输。字符中第 9 个数据位，位于第 8 个数据位和校验位之间。图中，D8 表示第 9 个 bit。

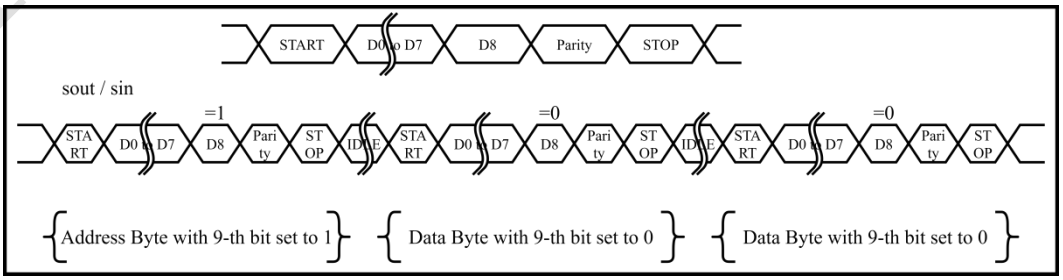


图 25-5　9bit 参数

通过使能 9bit 数据传输模式，UART 可以用于一个 master 和多个 slave 组成的多点系统。当 master 和多个 slave 中的一个进行通信，需要向 slave 发送一堆数据时，master 首先需要发送一个地址字节表明目标 slave。地址字节和数据字节的不同在于：地址字节的第 9 个 bit 为 1。地址字节发出后，所有 slave 都会对比地址字节和自己的地址。只有一个 slave（和地址字节匹配）会被使能来接收 master 发送的数据，之后 master 开始发送数据。没有匹配上的 slaves 会无视发送过来的数据，直到接收到一个新的地址字节。

配置 UART 9bit 数据传输的步骤如下：

- UART_CTRL[16]是使能或不使能 9bit 数据传输；
- UART_CTRL[17]是用于选择在接收情况下的地址匹配模式；
- UART_CTRL[19]是用于使能发送情况下的地址发送；
- UART_CTRL[18]选择地址发送模式；
- BIT9_TAR 和 BIT9_RAR 寄存器分别用于发送地址和匹配接收到的地址；
- 9 位的 SPI_DATA 寄存器是用于执行 9bit 模式下的数据传输；

UART 支持两种发送模式：

- 硬件地址匹配接收模式（9BIT_RXMODE（UART_CTRL[17]）为 1）
- 软件地址匹配接收模式（9BIT_RXMODE（UART_CTRL[17]）为 0）

硬件地址匹配接收模式

在硬件地址匹配接收模式中，如果接收到的字符的第 9 个数据位为 1，UART 通过接收地址寄存器（BIT9_RAR）中配置的地址来匹配接收到字符。如果接收到的地址与 BIT9_RAR 寄存器中配置的地址匹配上，那接下来的数据字节将会放进 RX FIFO 中。如果地址不匹配，UART 会丢掉后续的数据字节，直到接收到一个匹配的地址。在地址匹配下，UART 的数据传输流程图 25-6 所示。

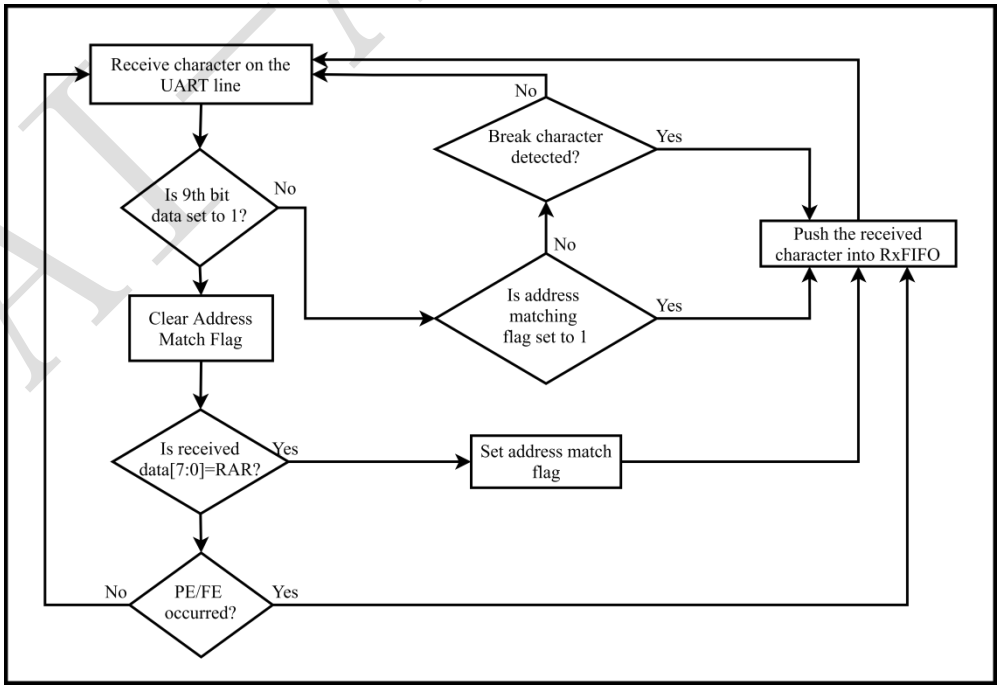


图 25-6 硬件地址匹配接收模式

软件地址匹配接收模式

在这种模式下，UART 不会将接收到的地址和 BIT9_RAR 寄存器进行匹配。UART 一直接收 9bit 的数据，并将数据放进 RX FIFO。每当接收到地址字节时，用户必须比较地址。当地址不匹配时，用户可以通过控制寄存器（UART_CTRL）的 RFRST 位来刷新或重置 RX FIFO。

25.4.4 小数波特率支持

UART 支持小数波特率：可以让用户配置波特率分频值的小数部分由此产生波特率的小数部分，减少频率误差。UART 需要通过软件配置处理让波特率在 2%的频率误差内。UART 的波特率是由系统时钟以及分频锁存寄存器（BAUD_INTE 和 BAUD_FRAC）共同控制的。波特率由以下因素决定：

- 串行时钟（系统时钟）频率
- 预期的波特率
- 波特率分频值（BAUD_INTE 和 BAUD_FRAC）
- 可接受的波特率误差，%ERROR

波特率的计算公式如下：

$$波特率=\frac{串行时钟频率}{(16\times 分频值)}$$

其中，分频值是 BAUD_INTE 和 BAUD_FRAC 配置的值，串行时钟频率是 UART 中系统时钟的频率。

配置 UART 小数波特率：通过分频锁存小数寄存器（BAUD_FRAC）配置分频小数部分的值。

可配置的小数波特率分频值让波特时钟较传统的整数分频有更好的分辨率。小数波特率分频中的小数和整数部分都是可配置的。由此波特率分频值的更准确的公式为：

$$波特率=\frac{串行时钟频率}{(16\times 分频值)}=BRD_I + BRD_F$$

其中，BRD_I 表示分频值的整数部分，BRD_F 表示分频值的小数部分。

25.4.5 DMA

UART 接口支持使用 DMA 来发送和接收数据。通过设置 UART_ENABLE[5:4]位可以单独开启 DMA 发送或者接收。发送时 RXFIFO 变满（RXFLTR 有关）或者 TXFIFO 变空（TXFLTR 有关）时，会产生 DMA 请求。DMA 控制器会将数据从 UART_DATA 寄存器中传送到预置的存储区或者将数据从预置的存储区装载到 UART_DATA 寄存器中去。

25.5 寄存器描述

25.5.1 寄存器列表

URAT0 基地址：**0x4000 2000**
URAT1 基地址：**0x4000 3000**

偏移	实例地址	名称	默认值	描述
0x00	0x40002000	UART_ENABLE	0x00000000	UART 使能寄存器
0x04	0x40002004	UART_CTRL	0x00000000	UART 控制寄存器
0x08	0x40002008	UART_BAUD	0x00000000	UART 波特率寄存器
0x0c	0x4000200C	UART_FIFOCTL	0x00000500	UART FIFO 控制寄存器
0x10	0x40002010	UART_DATA	0x00000000	UART 数据读写寄存器
0x14	0x40002014	UART_INTREN	0x00000000	UART 中断使能寄存器
0x18	0x40002018	UART_INTR	0x00000000	UART 中断寄存器
0x1c	0x4000201C	UART_DET	0x00000000	RS485 DE 时序寄存器
0x20	0x40002020	UART_TAT	0x00000000	RS485 周转时间寄存器
0x24	0x40002024	UART_RAR	0x00000000	9BIT 接收地址寄存器
0x28	0x40002028	UART_TAR	0x00000000	9BIT 发送地址寄存器

25.5.2 寄存器详细描述

25.5.2.1 UART 使能寄存器（UART_ENABLE）

- **名称：UART Enable Register**
- **偏移地址：0x00**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:6	5	4	3	2	1:0
名称		DMATXEN	DMARXEN	TFRST	RFRST	UARTMODE
访问		R/W	R/W	W	W	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[5] DMATXEN	**发送 DMA 使能（UART TX DMA Enable）** 该位启用/禁用发送 FIFO DMA 通道 0：禁用发送 DMA 通道 1：使能发送 DMA 通道
[4] DMARXEN	**接收 DMA 使能（UART RX DMA Enable）** 该位启用/禁用接收 FIFO DMA 通道 0：禁用接收 DMA 通道 1：使能接收 DMA 通道
[3] TFRST	**TXFIFO 复位（TXFIFO RESET）** TXFIFO 复位信号，自清 0 0：TXFIFO 正常 1：TXFIFO 复位
[2] RFRST	**RXFIFO 复位（RXFIFO RESET）** RXFIFO 复位信号，自清 0 0：RXFIFO 正常 1：RXFIFO 复位
[1:0] UARTMODE	**UART 模式选择（UART Mode Select）** 00：Resered 01：仅 TX 模式 10：仅 RX 模式 11：同时 TX 和 RX

25.5.2.2 UART 控制寄存器（UART_CTRL）

- 名称：UART Control Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:20	19	18	17	16
名称		BIT9TXAD	BIT9TXMD	BIT9RXMD	BIT9EN
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	15:14	13	12	11	10
名称		DEOUT	REOUT	RS485TXMD	DEPOL
访问		R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0
位	9	8	7	6	5
名称	REPOL	RS485EN		BKEN	PRTST
访问	R/W	R/W		R/W	R/W
复位值	0x0	0x0		0x0	0x0
位	4	3	2	1:0	
名称	PRTSEL	PRTEN	SPSEL	LENSEL	
访问	R/W	R/W	R/W	R/W	
复位值	0x0	0x0	0x0	0x0	

字段	说明
[19] BIT9TXAD	**9BIT 模式下发送地址控制位（UART 9BIT TX Address Flag ）** 该位控制何时发送地址，自动清 0。 0：uart 正常 1：uart 接下来将发送 9 位串行地址，第 9 位为 1，其余 8 位与 TAR 寄存器配置的一致。
[18] BIT9TXMD	**发送地址匹配模式（UART 9BIT TX Mode）** 作为主机用于发送期间是否启动地址匹配功能。 0：普通模式。UART 不区分地址，全都视为数据发送出去，此时 UART_DATA 的第 8 位可写。 1：地址匹配模式。对 UART_CTRL[19]位写 1 会将地址发送出去
[17] BIT9RXMD	**接收地址匹配模式（UART 9BIT RX Mode）** 作为从机用于接收期间是否启动地址匹配功能。 0：普通模式。uart 不进行匹配地址，直接将收到的 9 位字符包括地址全部写入 RXFIFO 中去，由软件自行处理。 1：地址匹配模式。检测到第 9 位为 1 时，取出后面的 8 位数据检查是否和 RAR 寄存器中配置的地址完全匹配。如果匹配，随后的数据将视为有效数据，uart 开始接收数据；否则随后的数据将不接收。

[16] BIT9EN	**UART 9BIT 使能位（UART 9BIT Enable）** 用于是否启用 9 位数据进行发送和接收传输。 0：使用 LENSEL 选择的数据位数 1：9bit 模式
[13] DEOUT	**DE 信号输出（UART RS485 DE Output）** 在 RS485_TXMODE 为 0 时有效 0：DE 输出 0 1：DE 输出 1
[12] REOUT	**RE 信号输出（UART RS485 RE Output）** 在 RS485_TXMODE 为 0 时有效 0：RE 输出 0 1：RE 输出 1
[11] RS485TXMD	**RS485 传输模式（UART RS485 TX Mode）** 0：软件模式，这种情况下 DE 和 RE 完全由软件控制，DE 的值将由 DE_OUT 寄存器控制，RE 的值将由 RE_OUT 寄存器控制 1：在这种模式下，DE 和 RE 是互斥的。默认状态为“RE 状态”，即 UART 保持接收状态接收。如果软件写 TXFIFO，则 UART 会在确保没有进行接收的前提下，将“RE 状态”变为“ DE 状态”。一旦 TX FIFO 为空，UART 会重新进入“RE 状态”。在这种操作模式下，当从 RE 切换到 DE 或从 DE 切换到 RE 时，硬件会考虑 TAT 寄存器中编程的周转时序。在这种模式下，“DE 状态”和“RE 状态”严格互补。
[10] DEPOL	**DE 极性（UART RS485 DE Polarity）** 0：DE 低有效 1：DE 高有效
[9] REPOL	**RE 极性（UART RS485 RE Polarity）** 0：RE 低有效 1：RE 高有效
[8] RS485EN	**RS485 使能（UART RS485 Enable）** 0：Disable 1：Enable
[6] BKEN	**BREAK 控制使能位（UART Break Enable）** 用于将 break 信号发送给接收设备。 0：正常模式 1：串行输出强制为 0 状态输出。
[5] PRTST	**强制校验使能位（UART Stick Parity Enable）** 用于在奇偶校验位开启（Parity_En=1）的情况下强制奇偶校验位。 0：不使用强制校验位 1：当 Parity_Sel 设置为 1 时，奇偶检验位强制发送或者检测逻辑 0；当 Parity_Sel 设置为 0 时，奇偶检验位强制发送或者检测逻辑 1.
[4] PRTSEL	**奇偶检验位选择（UART Parity Select）** 0：发送或检查奇数个逻辑 1 1：发送或检查偶数个逻辑 1
[3] PRTEN	**奇偶检验位使能（UART Parity Enable）** 0：Disable 1：Enable

[2]	**STOP 选择（UART STOP Select）**
SPSEL	0：1 个 STOP 位
	1：2 个 STOP 位
[1:0]	**串行数据长度选择（UART Length Select）**
LENSEL	00：5bit
	01：6bit
	10：7bit
	11：8bit

25.5.2.3 UART 波特率寄存器（UART_BAUD）

- **名称：UART Baud Rate Register**
- **偏移地址：0x08**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:20	19:4	3:0
名称		BAUD_INTE	BAUD_FRAC
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[19:4]	**波特率整数部分（UART Baud Rate Integer Part）**
BAUD_INTE	除数锁存器寄存器，包含 UART 的波特率除数的整数部分。
[3:0]	**波特率小数部分（UART Baud Rate Fractional Part）**
BAUD_FRAC	除数锁存器寄存器，包含 UART 的波特率除数的小数部分。

25.5.2.4 UART FIFO 控制寄存器（UART_FIFOCTL）

- 名称：UART FIFO Control Register
- 偏移地址：0x0C
- 默认值：0x00000500
- 返回寄存器列表

位	31:29	28:24	23:21	20:16	15:12	11
名称		RXFLR		TXFLR		RFF
访问		R		R		R
复位值		0x0		0x0		0x0
位	10	9	8	7:4	3:0	
名称	RFE	TFF	TFE	RXFTLR	TXFTLR	
访问	R	R	R	R/W	R/W	
复位值	0x1	0x0	0x1	0x0	0x0	

字段	说明
[28:24] RXFLR	**RXFIFO 实时剩余数据量（RXFIFO Level Register）**
[20:16] TXFLR	**TXFIFO 实时剩余数据量（TXFIFO Level Register）**
[11] RFF	**接收 FIFO 满标志（RXFIFO Full Flag）** 0：接收 FIFO 未满 1：接收 FIFO 已满 当 RX FIFO 不再满时，该位被清除。
[10] RFE	**接收 FIFO 空标志（RXFIFO Empty Flag）** 0：接收 FIFO 不为空 1：接收 FIFO 为空 当 RX FIFO 不为空时，该位被清除。
[9] TFF	**发送 FIFO 满标志（TXFIFO Full Flag）** 0：发送 FIFO 未满 1：发送 FIFO 为满 当 TX FIFO 不再满时，该位被清除。
[8] TFE	**发送 FIFO 空标志（TXFIFO Empty Flag）** 0：发送 FIFO 不为空 1：发送 FIFO 为空 当 TX FIFO 不为空时，该位被清除。

[7:4]	**RXFIFO 满中断触发的预设值。**
RXFTLR	4'b0000：FIFO 中有 1 个或 1 个以上数据就会触发 RXFIFO 满中断
	4'b0001：FIFO 中有 2 个或 2 个以上数据就会触发 RXFIFO 满中断
	4'b0010：FIFO 中有 3 个或 3 个以上数据就会触发 RXFIFO 满中断
	4'b0011：FIFO 中有 4 个或 4 个以上数据就会触发 RXFIFO 满中断
	4'b0100：FIFO 中有 5 个或 5 个以上数据就会触发 RXFIFO 满中断
	4'b0101：FIFO 中有 6 个或 6 个以上数据就会触发 RXFIFO 满中断
	4'b0110：FIFO 中有 7 个或 7 个以上数据就会触发 RXFIFO 满中断
	4'b0111：FIFO 中有 8 个或 8 个以上数据就会触发 RXFIFO 满中断
	4'b1000：FIFO 中有 9 个或 9 个以上数据就会触发 RXFIFO 满中断
	4'b1001：FIFO 中有 10 个或 10 个以上数据就会触发 RXFIFO 满中断
	4'b1010：FIFO 中有 11 个或 11 个以上数据就会触发 RXFIFO 满中断
	4'b1011：FIFO 中有 12 个或 12 个以上数据就会触发 RXFIFO 满中断
	4'b1100：FIFO 中有 13 个或 13 个以上数据就会触发 RXFIFO 满中断
	4'b1101：FIFO 中有 14 个或 14 个以上数据就会触发 RXFIFO 满中断
	4'b1110：FIFO 中有 15 个或 15 个以上数据就会触发 RXFIFO 满中断
	4'b1111：FIFO 中有 16 个数据就会触发 RXFIFO 满中断
[3:0]	**TXFIFO 空中断触发的预设值。**
TXFTLR	4'b0000：FIFO 中有 0 个就会触发 TXFIFO 空中断
	4'b0001：FIFO 中有 1 个或 1 个以下就会触发 TXFIFO 空中断
	4'b0010：FIFO 中有 2 个或 2 个以下就会触发 TXFIFO 空中断
	4'b0011：FIFO 中有 3 个或 3 个以下就会触发 TXFIFO 空中断
	4'b0100：FIFO 中有 4 个或 4 个以下就会触发 TXFIFO 空中断
	4'b0101：FIFO 中有 5 个或 5 个以下就会触发 TXFIFO 空中断
	4'b0110：FIFO 中有 6 个或 6 个以下就会触发 TXFIFO 空中断
	4'b0111：FIFO 中有 7 个或 7 个以下就会触发 TXFIFO 空中断
	4'b1000：FIFO 中有 8 个或 8 个以下就会触发 TXFIFO 空中断
	4'b1001：FIFO 中有 9 个或 9 个以下就会触发 TXFIFO 空中断
	4'b1010：FIFO 中有 10 个或 10 个以下就会触发 TXFIFO 空中断
	4'b1011：FIFO 中有 11 个或 11 个以下就会触发 TXFIFO 空中断
	4'b1100：FIFO 中有 12 个或 12 个以下就会触发 TXFIFO 空中断
	4'b1101：FIFO 中有 13 个或 13 个以下就会触发 TXFIFO 空中断
	4'b1110：FIFO 中有 14 个或 14 个以下就会触发 TXFIFO 空中断
	4'b1111：FIFO 中有 15 个或 15 个以下就会触发 TXFIFO 空中断

25.5.2.5 UART 数据读写寄存器（UART_DATA）

- **名称：UART DATA Register**
- **偏移地址：0x10**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:12	11	10	9	8	7:0
名称		BKST	FMST	PRST	BIT9	DATA
访问		R	R	R	R	R/W
复位值		0x0	0x0	0x0	0x0	0x0

字段	说明
[11] BKST	**BREAK 状态（Break Status）** 当前读回来的数据是否为 break 状态 0：数据正常 1：数据位 break 状态
[10] FMST	**帧状态（Frame Status）** 当前读回来的数据的 stop 位是否正确 0：stop 位正常 1：stop 位错误
[9] PRST	**奇偶校验位状态（Parity Status）** 当前读回来的数据的奇偶校验位是否正确 0：奇偶校验位正常 1：奇偶校验位错误
[8] BIT9	**第 9 位状态（9th BIT Status）** 串行输入端口的 MSB 第 9 位接收到的数据字节
[7:0] DATA	**UART 数据读写寄存器（UART Data Write/Read Register）** W：往 FIFO 写要发送的数据 R：读取从外界收到的并已经存到 FIFO 中的数据）

25.5.2.6 UART 中断使能寄存器（UART_INTREN）

- 名称：UART Interrupt Enable Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7	6	5	4	3	2	1	0
名称		TXDEIE	RXDATTOIE	BKIE	FEIE	PEIE	RXOFIE	TXEIE	RXFIE
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] TXDEIE	**TXDEI 中断使能（UART TX Done Interrupt Enable）** 0：不使能 1：使能
[6] RXDATTOIE	**RXDATTOI 中断使能（UART RX Data Timeout Interrupt Enable）** 0：不使能 1：使能
[5] BKIE	**BKI 中断使能（UART Break Interrupt Enable）** 0：不使能 1：使能
[4] FEIE	**FEI 中断使能（UART Frame Error Interrupt Enable）** 0：不使能 1：使能
[3] PEIE	**PEI 中断使能（UART Parity Error Interrupt Enable）** 0：不使能 1：使能
[2] RXOFIE	**RXOFI 中断使能（RXFIFO Overflow Interrupt Enable）** 0：不使能 1：使能
[1] TXEIE	**TXEI 中断使能（TXFIFO Empty Interrupt Enable）** 0：不使能 1：使能
[0] RXFIE	**RXFI 中断使能（RXFIFO Full Interrupt Enable）** 0：不使能 1：使能

25.5.2.7 UART 中断寄存器（UART_INTR）

- **名称：UART Interrupt Register**
- **偏移地址：0x18**
- **默认值：0x00000002**
- 返回寄存器列表

位	31:8	7	6	5	4	3	2	1	0
名称		TXDEI	RXDATTOI	BKI	FEI	PEI	RXOFI	TXEI	RXFI
访问		R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R	R
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x1	0x0

字段	说明
[7] TXDEI	**UART 发送完成中断（UART TX Done Interrupt ）** UART 发送完成且 TXFIFO 为空起中断。 0：未发送完成 1：发送完成 对该位写 1 将清除 TXDONEI 中断
[6] RXDATTOI	**RX 数据超时中断位（UART RX Data Timeout Interrupt ）** 指的是在最近的 4 个帧长度时间内，没有字符进出 RXFIFO，并且 RXFIFO 不为空 0：无超时 1：超时 对该位写 1 将清除 RXDATTOI 中断
[5] BKI	**BREAK 中断位（UART Break Interrupt ）** 用于指示检测到串行数据上的 break 序列。串行输入保持逻辑 0 状态的时间长于 START 位+LENSEL 个串行数据位+奇偶检验位+STOP 位的总和，并且读取了 FIFO 就会置位。 0：非 break 状态 1：break 状态 对该位写 1 将清除 BI 中断
[4] FEI	**帧错误中断位（UART Frame Error Interrupt ）** UART 检测到接收数据中没有有效的 stop 位时就会发生帧错误。 0：没有帧错误 1：帧错误 对该位写 1 将清除 FEI 中断
[3] PEI	**奇偶校验位中断位（UART Parity Error Interrupt ）** uart 检测到接收数据中的奇偶校验位错误时就会置位。 0：奇偶校验位正确 1：奇偶校验位错误 对该位写 1 将清除 PEI 中断

[2] RXOFI	**RXFIFO 溢出错误位（RXFIFO Overflow Interrupt）** 当 RXFIFO 已满并且新数据被接收写入 FIFO 中时，会起溢出错误中断。此时 FIFO 中的数据将保留，而接收的新数据将丢失。 0：无溢出错误 1：溢出错误 对该位写 1 将清除 RXOFI 中断。
[1] TXEI	**TXFIFO 空中断（TXFIFO Empty Interrupt ）** 与 TXFTLR 寄存器有关，当 TXFIFO 中的数据量小于或者等于预设值（TXFTLR）时置位。 0：TXFIFO 数据量大于预设值 1：TXFIFO 数据量小于等于预设值 当 TXFIFO 数据量大于预设值时 TXFIFOEI 中断自动清 0
[0] RXFI	**RXFIFO 满中断（RXFIFO Full Interrupt ）** 与 RXFTLR 寄存器有关，当 RXFIFO 中的数据量大于或等于预设值（RXFTLR）时置位。 0：RXFIFO 数据量小于预设值 1：RXFIFO 数据量大于等于预设值 当 RXFIFO 数据量小于预设值时 RXFIFOFI 中断自动清 0

25.5.2.8 RS485 DE 时序寄存器（UART_DET）

- **名称：UART DE Timing Register**
- **偏移地址：0x1C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:24	23:16	15:8	7:0
名称		DEAST		AST
访问		R/W		R/W
复位值		0x0		0x0

字段	说明
[23:16] DEAST	**DE 取消断言时间（DE DE-Assert Timing）** 从发送完 STOP 位后一直到 DE 的下降沿之间的时间，配置的值是系统时钟的周期数。
[7:0] AST	**DE 断言时间（DE Assert Timing）** 从 DE 上升沿到发送 START 位之间的时间，配置的值是系统时钟的周期数。

25.5.2.9 RS485 周转时间寄存器（UART_TAT）

- 名称：UART TurnAround Timing Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:16	15:0
名称	DERET	REDET
访问	R/W	R/W
复位值	0x0	0x0

字段	说明
[31:16]	**DE 到 RE 的 TurnAround 时间（DE-RE TurnAround Timing）**
DERET	DE 无效到 RE 断言的周转时间（以串行时钟为单位）。 注意：实际时间是配置的值+1 个时钟周期。
[15:0]	**RE 到 DE 的 TurnAround 时间（RE-DE TurnAround Timing）**
REDET	RE 无效到 DE 断言的周转时间（以串行时钟为单位）。 注意：实际时间是配置的值+1 个时钟周期。

25.5.2.10 9BIT 接收地址寄存器（UART_RAR）

- 名称：UART Receive Address Register
- 偏移地址：0x24
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7:0
名称		RAR
访问		R/W
复位值		0x0

字段	说明
[7:0]	**接收模式下的地址寄存器（Receive Address Register）**
RAR	若串行数据第 9 位为 1，则对照该寄存器检查其余的 8 位。 注：仅当 ADDR_MATCH 和 DLS_E 设置为 1 时，该寄存器才有用

25.5.2.11 9BIT 发送地址寄存器（UART_TAR）

- 名称：UART Transmit Address Register
- 偏移地址：0x28
- 默认值：0x00000000
- 返回寄存器列表

位	31:8	7:0
名称		TAR
访问		R/W
复位值		0x0

字段	说明
[7:0]	**发送模式下的地址匹配寄存器（Transmit Address Register）**
TAR	如果启用了 BIT9EN 位，同时将 BIT9TXAD 设置为 1，则 UART 会发送第 9 位为 1，其余 8 位为该寄存器的串行数据。 注：该寄存器仅用于发送地址，数据通过配置 UART_DATA 发送。

26 控制器局域网络（CAN）

26.1 简介

CAN 通信是按框架组织的。存在两种类型的框架：标准框架和扩展框架。对于 CAN 2.0，最大数据有效载荷最多为 8 个字节。

在总线访问方面，所有 CAN 节点均相等。没有超级节点，因为 CAN 是多主控总线。数据寻址是使用消息标识符完成的。在 CAN 网络中，只有一个节点应发送带有特定标识符的消息。所有节点都接收所有消息，并且节点主机控制器必须决定是否通过适当的消息标识符对其进行寻址。为了减少主机控制器的负载，CAN 内核可以使用验收滤波器。这些过滤器将所有接收到的消息标识符与用户可选的位模式进行比较。仅当消息通过接受过滤器时，才会通过信号发送到主机控制器。

CAN 2.0B 定义了高达 1Mbit/s 的数据比特率。

26.2 结构框图

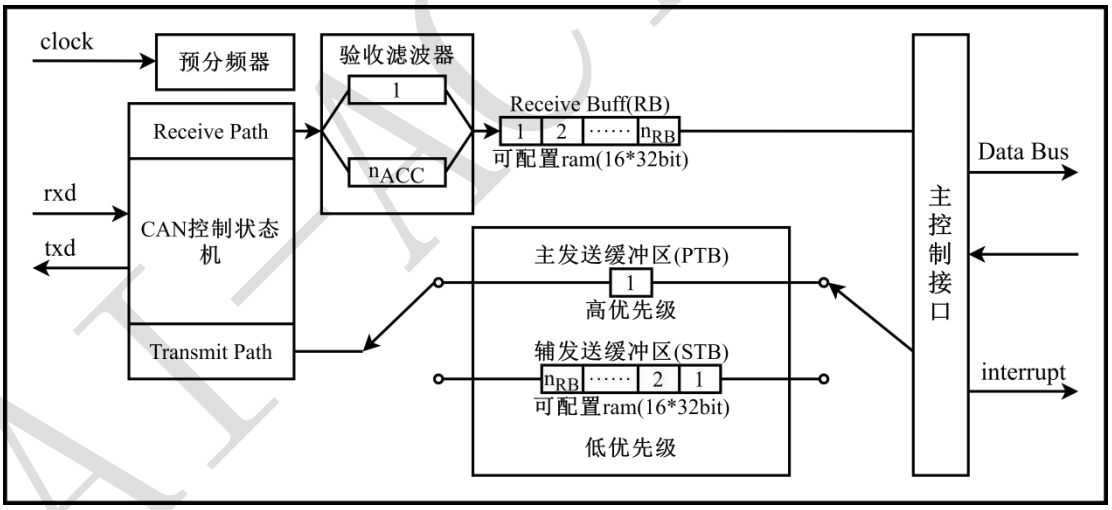


图 26-1　CAN 结构框图

26.3 主要特性

- 支持 CAN 规范
  - CAN 2.0B（最高 8 字节有效负载）
- 自由可编程数据速率
  - CAN 2.0B 定义了高达 1Mbit / s 的数据速率
- 可编程波特率预分频器（1/2 至 1/256）
- 主机接口和 CAN 协议机器的单独时钟域
- 可配置的接收缓冲区（RB）大小
  - 通用参数选择缓冲区插槽的数量
  - 类似于 FIFO 的行为（基于双端口内存）
  - 收到的“不接受”或“不正确”的消息不会覆盖已存储的消息
- 两个发送缓冲区
  - 一个主发送缓冲器（PTB）
  - 可选的可配置二级发送缓冲器（STB）
- 可以包含或不包含 STB。如果包含，则通用参数选择 1 到 16 个缓冲区插槽。
- 类似 FIFO 的行为
  - 一个用于 PTB 和 STB 的双端口存储模块
- 独立的可编程内部 29 位接收滤波器
  - 通用参数可选择的接收滤波器数量，范围为 1 到 16
- 扩展功能
  - 单发传输模式（用于 PTB 和/或 STB）
  - 仅收听模式
  - 回送模式（内部和外部）
  - 收发器待机模式
- 扩展状态和错误报告
  - 捕获上次发生的错误
  - 捕获仲裁丢失的位置
  - 可编程错误警告极限
- 不同的主机控制器接口
  - 32 位同步主机控制器接口；8 位主机的包装器
  - 根据要求，可选的应用程序特定于主机控制器的接口
- 可配置的中断源

26.4 功能描述

表 26-1 英文缩写列表

Abbreviation	Description
ACF	Acceptance Filter
PTB	Primary Transmit Buffer (high priority)
RB	Receive Buffer
SP	Sample Point
STB	Secondary Transmit Buffer (low priority)
TQ	Time Quanta (CAN specification)

26.4.1 时钟

CAN 使用两个时钟域：一个用于主机接口，一个用于 CAN IP 核。这样，就可以选择以最适合所需 CAN 总线数据速率的时钟来操作 CAN，而主机则可以自由地以不同（更高）的时钟速度进行操作。

时钟域交叉（CDC）通过双缓冲同步器和双向握手机制完成。以下提供有关 CDC 的一些详细信息：

- 控制和状态寄存器分别移至相应的时钟域。例如。中断标志使用双向握手，而状态寄存器使用单向同步器。
- 在启动期间，当 RESET 位变为无效时，验收过滤器将复制到 CAN 时钟域。当 CAN 节点在总线空闲时间之后参与通信时，复制接受过滤器将完成。
- 接收的帧存储在较小的临时接收缓冲区中。该临时缓冲区仅保存部分帧，并在必要时将这些部分复制到 RBUF。
- 将要传输的帧逐步从 TBUF 复制到较小的临时传输缓冲区。

使用同步器和握手机制会导致一些延迟。双向握手在每个域中最多需要 3 个时钟。

26.4.2 Bus-Off 状态

使用寄存器 CFG_STAT 中的状态位 BUSOFF 指示“Bus-Off”状态。如果 CAN 节点的传输错误计数器大于 255，则会自动进入“Bus-Off”状态。然后，它将不再参与进一步的通信，直到它再次返回到错误激活状态。如果启用了 EIE，则将 BUSOFF 设置为 1 也会设置 EIF 中断。如果 CAN 节点通过加电复位进行复位或接收到 128 个 11 位隐性位序列（恢复序列），则它将返回到错误激活状态。

在“Bus-Off”状态下，RECNT 用于计数恢复序列，而 TECNT 保持不变。请注意，进入总线关闭状态时 TECNT 会翻转，因此可能保持较小的值。因此，建议在节点进入总线关闭状态之前使用 TECNT，然后再进入状态位 BUSOFF。

如果节点从“Bus-Off”状态恢复，则 RECNT 和 TECNT 将自动重置为 0。

26.4.3 Acceptance Filters（ACF）

为了减轻主机控制器接收到的帧的负载，内核使用了接受过滤器。CANCTRL 内核在接受过滤期间检查消息标识符。因此，每个验收滤波器的长度为 29 位。

如果报文通过了其中一个过滤器，则它将被接受。如果被接受，则报文将被存储到 RB 中，如果启用了 RIE，则最终将设置 RIF。如果不接受该报文，则不设置 RIF，并且不增加 RB FIFO 指针。不接受的消息将被丢弃，并被下一条报文覆盖。任何未接受的报文都不会覆盖已存储的有效消息。与接受过滤的结果无关，CAN-CTRL 内核检查总线上的每条报文并向总线发送确认或错误帧。

接受掩码定义应比较的位，而接受代码定义适当的值。将屏蔽位设置为 0 可使所选接受代码位与相应的报文标识符位进行比较。设置为 1 的掩码位被禁用以进行接受检查，这将导致接受报文。

标识符位将与相应的接受码位 ACODE 进行如下比较：

- 标准：ID（10：0）和 ACODE（10：0）
- 扩展：ID（28：0）和 ACODE（28：0）

示例：如果 AMASK_x（0）= 0，并且所有其他 AMASK_x 位均为 1，则对于接受的消息，最后一个 ID 位的值必须等于 ACODE（0），滤波器将忽略所有其他 ID 位。

26.4.4 接收报文

接收到的数据将被存储在 RB 中， RB 的行为类似于 FIFO。如果启用了 RIE，则每个接收到的有效且已接受的报文都将设置 RIF = 1。根据填充状态设置 RSTAT。当已填充的缓冲区数等于可编程值 AFWL 时，如果启用了 RAFIE，则会设置 RAFIF。如果所有缓冲区都已满，则如果启用了 RFIE，则会设置 RFIF。

RB 始终将包含最早报文的报文插槽映射到 RBUF 寄存器。

如果 RB 已满，则下一条传入报文将被临时存储，直到通过为止（第 6 个 EOF 位）。然后，如果启用了 ROIE，则最早的报文将被最新报文覆盖，并设置 ROIF。如果主机控制器读取最旧的报文并在新的传入消息变为有效之前设置 RREL，则不会覆盖任何报文。

处理消息接收

如果没有接受过滤，CAN-CTRL 内核将发出信号通知每个帧的接收，并且将要求主机决定是否对其进行寻址。这将导致主机控制器上的负担很大。

可以禁用中断并使用接受过滤器来减轻主机控制器的负载。对于基本操作，如果启用了 RIE 并且 CAN-CTRL 内核已收到有效消息，则 RIF 设置为 1。为了减少接收中断的数量，可以使用 RAIE / RAIF（RB 几乎完全中断）或 RFIE / RFIF（RB 完全中断）代替 RIE / RIF（接收中断）。可使用 AFWL 编程“几乎满极限”。

RB 包含许多 RB 空间，其可以在合成之前使用通用参数来选择。阅读 RB 的步骤如下：

1. 使用 RBUF 寄存器从 RB FIFO 中读取最早的报文。
2. 释放 RREL=1 的 RB 插槽。这将选择下一条报文（下一个 FIFO 插槽），RBUF 将自动更新。
3. 重复这些操作，直到 RSTAT 发出空 RB 为止。

如果 RB FIFO 已满，则如果新接收到的报文被识别为有效（第 6 个 EOF 位），则最早的报文将被覆盖。在此事件之前，不会覆盖任何报文。这将为主机控制器留出足够的时间，以便在 RB FIFO 已满且发生所选中断后，从 RB 读取至少一帧。为了实现此行为，RB 包含比综合参数 RBUF_SLOTS 指定的更多（隐藏）的插槽。该隐藏的插槽用于接收消息，对其进行验证并在覆盖较旧的报文之前检查它是否与接受过滤器匹配。

26.4.5 发送报文

在开始任何传输之前，至少一个传输缓冲区（PTB 或 STB）必须装有一条消息。 TPE 发信号通知 PTB 是否锁定，TSSTAT 发信号通知 STB 的填充状态。

TBUF 寄存器提供对 PTB 和 STB 的访问。以下是推荐的编程流程：

1. 将 TBSEL 设置为所需值以选择 PTB 或 STB。
2. 将帧写入 TBUF 寄存器。
3. 对于 STB，将 TSNEXT = 1 设置为完成此 STB 插槽的加载。

CAN 2.0 消息的最大有效载荷长度为 8 字节。每个消息的单独长度由 DLC 定义。对于远程帧（RTR 位），DLC 变得毫无意义，因为远程帧的数据长度始终为 0 字节。由于 DLC 和 RTR，要求主机控制器将 TSNEXT 设置为跳到下一个 STB 插槽。所有 TBUF 字节均可按任何顺序写入。

如果 TBSEL = 0 选择了 PTB，则将 TSNEXT = 1 设置是没有意义的。在这种情况下，TSNEXT 会自动清除，不会造成错误。

使用 PTB 时，应将 TPE 位置 1 以开始发送。要使用 STB，必须将 TSONE 设置为开始传输单个消息（最旧的消息），或者将 TSALL 设置为传输所有消息。

PTB 始终具有比 STB 高的优先级。如果两个发送缓冲区都具有发送顺序，则无论帧标识符如何，始终将首先发送 PTB 消息。如果来自 STB 的传输已经处于活动状态，则它将在下一个可能的传输位置（下一个帧间时隙）发送来自 PTB 的消息之前完成。 PTB 传输完成或中止之后，CANCTRL 内核返回以处理来自 STB 的其他挂起消息。

传输完成后，将设置以下传输中断：

- 对于 PTB，如果启用 TPIE，则 TPIF 会置 1。
- 对于使用 TSONE 的 STB，如果完成了一条消息并启用了 TSIE，则 TSIF 会置 1。
- 对于使用 TSALL 的 STB，如果所有消息均已完成并且启用了 TSIE，则 TSIF 会置 1。换句话说：如果 STB 为空，则 TSIE 会置 1。因此，如果在开始 TSALL 传输之后，主机控制器将附加消息写入 STB，则 TSIF 置 1 之前，还将发送附加消息。

STB 为空时设置 TSONE 或 TSALL 是没有意义的。在这种情况下，TSONE 或 TSALL 将自动重置。不会设置中断标志，也不会发送任何帧。

消息传输中止

如果出现这种情况，即由于其低优先级而无法发送发送缓冲区中的消息，这将长时间阻塞缓冲区。为了避免这种情况，如果尚未开始传输，则主机控制器可以通过分别设置 TPA 或 TSA 撤回传输请求。

TPA 和 TSA 都提供一个中断标志：AIF。CAN 协议机器仅在不向 CAN 总线传输任何内容时才执行中止操作。因此，以下内容是有效的：

- 总线仲裁期间不中止。
  - 如果节点丢失仲裁，则中止将在之后执行。
  - 如果节点赢得仲裁，则将发送该帧。
- 传输帧时不中止。
  - 如果成功发送了帧，则将成功发送信号通知主机控制器。在这种情况下，不会发出中止的信号。这是通过相应的中断和状态位完成的。
  - 在 CAN 节点未收到确认的传输失败后，错误计数器将递增，并执行中止操作。
  - 如果主机已命令发送所有帧（TSALL = 1），则 STB 中至少剩余一帧，则已完成的帧和中止都将信号发送给主机。

由于这些事实，中止传输可能需要一些时间，具体取决于 CAN 通信速度和帧长度。如果执行中止，则将导致以下操作：

- TPA 释放 PTB，导致 TPE = 0。释放 PTB 之后，帧数据仍存储在 PTB 中。
- TSA 释放一个（最旧的）消息插槽或 STB FIFO 的所有消息插槽。这取决于是使用 TSONE 还是 TSALL 来开始传输。TSSTAT 将相应更新。

在 STB 中释放帧会导致丢弃帧数据，因为主机由于类似 FIFO 的行为而无法访问它。

不建议同时设置 TPA 和 TSA。如果主机控制器决定仍然执行此操作，则将设置 AIF，并且如果可能的话，来自 PTB 和 STB 的传输都将中止。如已经说明的，如果在中止可以执行之前将完成一次传输，这将导致发出成功传输的信号。因此，如果启用，可以设置以下中断标志：

- AIF（一次用于 PTB 和 STB 传输中止）
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF（很少，仅当主机不立即处理 TPIF 时才会发生）
- TPIF + TSIF + AIF（很少，只有在主机不立即处理 TPIF 和 TSIF 的情况下才会发生）

要清除整个 STB，需要同时设置 TSALL 和 TSA。为了检测是否由于丢失仲裁而无法长时间发送消息，主机可以使用 ALIF / ALIE。

26.4.6 扩展状态和错误报告

在 CAN 总线通信期间，可能会发生传输错误。以下功能支持对其进行检测和分析。

可编程错误警告极限

RECNT 和 TECNT 对接收/发送期间的错误进行计数。主机控制器可使用位于寄存器 LIMIT 中的可编程错误警告限制 EWL，以对这些事件做出灵活的反应。可以从 8 到 128 的 8 个误差中选择极限值：

错误计数限制=（EWL + 1）* 8

如果在以下情况下由 EIE 启用，则将设置中断 EIF：

- 错误警告限制的边界已被 RECNT 或 TECNT 沿任一方向交叉
- BUSOFF 位已在任一方向上更改。

仲裁丢失捕获（ALC）

内核能够检测仲裁丢失的仲裁字段中的确切位位置。如果已启用此事件，则可以通过 ALIF 中断来发出信号。如果节点能够赢得仲裁，则 ALC 的值保持不变。然后，ALC 保留最后一次仲裁失败的旧值。

ALC 的值定义如下：帧从 SOF 位开始，然后发送 ID 的第一位。该第一 ID 位的 ALC 值为 0，第二 ID 位的 ALC 值为 1，依此类推。

仅在仲裁字段中允许仲裁。因此，ALC 的最大值为 31，这是扩展帧中的 RTR 位。

注意：如果标准远程帧与扩展帧进行仲裁，则扩展帧将在 IDE 位丢失仲裁，ALC 将为 12。

错误种类（KOER）

内核识别 CAN 总线上的错误，并将最后的错误事件存储在 KOER 位中。如果使能了 BEIF 中断，则可以指示 CAN 总线错误。每个新的错误事件都会覆盖 KOER 的先前存储的值。因此，主机控制器必须对错误事件做出快速反应。成功发送或接收帧后，将重置 KOER。

26.4.7 扩展功能

单发传输

有时不希望自动重传。因此，可以通过位 TPSS 为发送缓冲区 PTB 和通过位 TSSS 为发送缓冲区 STB 分别设置仅发送一次消息的顺序。在这种情况下，如果选定的传输处于活动状态，则在发生错误或仲裁丢失的情况下，将不会执行任何重新传输。

在立即成功传输的情况下，与正常传输没有区别。但是，如果在传输过程中发生错误，则将 TPIF 置位（如果启用），则会清除相应的传输缓冲区，并更新 KOER 和错误计数器。因此，对于单发传输，仅 TPIF 不能指示帧是否已成功传输或发生了错误。

如果将单发传输与 TSALL 一起使用，并且 STB 中有一个以上的帧，则对于每一帧都将进行单发传输。无论是否成功传输了任何帧（例如由于 ACK 错误），CAN-CTRL 都会前进到下一个帧，如果 STB 为空则停止。

因此，在这种情况下只有错误计数器指示发生了什么。这可能非常复杂，因为如果两个帧之一出现错误，主机将无法检测到哪一个是成功的。

仅收听模式（LOM）

LOM 提供了监视 CAN 总线的能力，而对总线没有任何影响。

另一个应用是自动比特率检测，其中主机控制器尝试不同的时序设置，直到没有错误发生为止。

在 LOM 中，内核无法将显性位写入总线（没有活动的错误标志或过载标志，没有确认），但是这些显性位在内部环回模式下在内部重新路由，以便内核接收自己的位。将在 LOM 中监视错误（KOER，BEIF）。

对 LOM 重要：

- 无论任何错误情况，错误计数器均保持不变。
- 如果一个节点发送帧，则仅当至少一个附加节点连接到不在 LOM 中的总线时，才会生成 ACK。这样就不会有错误，并且所有节点（包括 LOM 中的那些节点）都将接收该帧。

如果除发送节点之外的所有节点都在 LOM 中，则总线上将没有 ACK，并且发送器将生成 ACK 错误，并且可能会重新发送帧。因为仅当总线是隐性的，直到 ACK 之后帧比特的第 6 个末尾才接收到帧，因此错误帧将违反此规则，因此 LOM 中的节点不会接收到该帧。

传输处于活动状态时，不应启用 LOM。主机控制器必须注意这一点。没有来自 CAN-CTRL 的附加保护。如果启用了 LOM，则无法开始传输。

总线连接测试

要检查节点是否连接到总线，必须执行以下步骤：

- 传送框架。如果节点连接到总线，则其 TX 位在其 RX 输入上可见。
- 如果还有其他节点连接到 CAN 总线，则期望传输成功（包括来自其他节点的确认），没有错误信号。
- 如果该节点是连接到 CAN 总线的唯一节点（但是总线，收发器和 CAN-CTRL 内核之间的连接很好），则第一个常规错误情况发生在 ACK 插槽中，因为没有来自用户的确认其他节点。如果启用且 KOER =“100”（ACK 错误），则将产生 BEIF 错误中断。
- 如果与收发器或总线的连接断开，则在 SOF 位之后立即设置 BEIF 错误中断，并且 KOER =“001”（位错误）。

环回模式（LBMI 和 LBME）

CAN 支持两种环回模式：内部（LBMI）和外部（LBME）。两种模式都导致接收到自己发送的帧，这对于自检很有用。

在 LBMI 中，CAN-CTRL 与 CAN 总线断开连接，并且 txd 输出设置为隐性。输出数据流将反馈到输入。

在 LBME 中，CAN-CTRL 保持与收发器的连接，并且已传输的帧在总线上可见。在这种模式下，CAN-CTRL 接收自己的帧。是否存在来自另一个节点的 ACK 无关紧要。

在环回模式下，内核接收自己的消息，将其存储在 RBUF 中，并设置相应的接收中断标志（如

果启用）。在回送模式传输期间，将禁用 ACK 错误生成，并且如果启用，将设置适当的传输中断。

LBMI 对于芯片内部和软件测试很有用，而 LBME 可以测试收发器及其连接。

传输处于活动状态时，不应同时激活两种环回模式。主机控制器必须注意这一点。没有来自 CAN-CTRL 的附加保护。

如果该节点已连接到 CAN 总线，则不能仅通过将 LBMI 设置为 0 来完成从 LBMI 切换回正常操作的操作，因为这可能是在另一个 CAN 节点正在传输时发生的。在这种情况下，应通过将 RESET 位设置为 1 来切换回正常工作。这将自动将 LBMI 清零。最后，可以禁止 RESET，并且内核将返回正常工作状态。与此相反，可以每次禁用 LBME。

收发器待机模式

使用寄存器位 STBY 来驱动输出信号 stby。它可用于激活收发器的待机模式。

激活待机后，将无法进行任何传输，因此无法设置 TPE，TSONE 和 TSALL。另一方面，如果传输已经激活，则 CAN-CTRL 不允许设置 STBY（设置了 TPE，TSONE 或 TSALL）。

如果设置了 STBY，则收发器进入低功耗模式。在这种模式下，它无法全速接收帧，但会监视 CAN 总线是否处于显性状态。如果主导状态在收发器数据手册中定义的时间内处于活动状态，则收发器会将 rxd 信号拉低。如果 rxd 为低，则 CAN-CTRL 自动将 STBY 清除为 0，这将禁用收发器的待机模式。这是在不中断主机控制器的情况下以静默方式完成的。

从待机模式切换到活动模式需要花费一些时间，因此收发器无法成功接收初始唤醒帧。因此，最近处于待机状态的节点将不会以 ACK 响应。如果总线上没有 CAN 节点通过 ACK 响应唤醒帧，则将导致唤醒帧发送器的 ACK 错误。然后，发送器将自动重复该帧。在重复过程中，收发器将返回活动模式，并且 CAN-CTRL 将接收该帧并以 ACK 响应

总结：一个节点发送一帧用于唤醒。如果所有其他节点都处于待机模式，则发送器会收到 ACK 错误，并将自动重复帧。在重复期间，节点返回到活动模式，并将以 ACK 响应。

STBY 不受复位位的影响。

26.4.8 软件复位

如果 CFG_STAT 中的 RESET 位设置为 1，则软件复位有效。 如果 RESET = 1，则多个组件将被强制进入复位状态，但是 RESET 不会触摸所有组件。 某些组件仅对硬件重置敏感。 对于软件和硬件复位，所有位的复位值始终相同。

表 26-2 复位对寄存器的影响

Register	RESET	Comment
RBUF	Yes	所有 RB 插槽均标记为空。 RBUF 包含未知数据。
TBUF	Yes	所有机顶盒插槽均标记为空。 由于 TBSEL，TBUF 指向 PTB。
LBME	Yes	-
LBMI	Yes	-

TPSS	Yes	-
TSSS	Yes	-
RACTIVE	Yes	即使接收处于活动状态，接收也会立即被取消。 不会产生 ACK。
TACTIVE	Yes	所有传输均立即通过 RESET 终止。 如果传输处于活动状态，则将导致错误的帧。其他节点将生成错误帧。
BUSOFF	No	-
TBSEL	Yes	TBUF 已固定为指向 PTB。
LOM	Yes	-
STBY	No	-
TPE	Yes	-
TPA	Yes	-
TSONE	Yes	-
TSALL	Yes	-
TSA	Yes	-
TSNEXT	Yes	-
TSSTAT	Yes	所有 STB 插槽均标记为空。
ROV	Yes	所有 RB 插槽均标记为空。
RREL	Yes	-
RSTAT	Yes	所有 RB 插槽均标记为空。
RIE	No	-
ROIE	No	-
RFIE	No	-
RAFIE	No	-
TPIE	No	-
TSIE	No	-
EIE	No	-
TSFF	Yes	所有 STB 插槽均标记为空。
RIF	Yes	-
ROIF	Yes	-
RFIF	Yes	-
RAFIF	Yes	-
TPIF	Yes	-
TSIF	Yes	-
EIF	No	-
AIF	Yes	-
EWARN	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
ALIE	No	-
ALIF	Yes	-
BEIE	No	-
BEIF	Yes	-

BITTIME_x	No	如果 RESET = 1，则该寄存器是可写的；如果是 0，则该寄存器是写锁定的。
S_PRESC, F_PRESC	No	如果 RESET = 1，则该寄存器是可写的；如果是 0，则该寄存器是写锁定的。
AFWL	No	-
EWL	Yes	-
KOER	Yes	-
ALC	Yes	-
RECNT	No	-
TECNT	No	-
SELMASK	No	-
ACFADR	No	-
AE_x	No	-
ACODE_x	No	如果 RESET = 1，则该寄存器是可写的；如果是 0，则该寄存器是写锁定的。
AMASK_x	No	如果 RESET = 1，则该寄存器是可写的；如果是 0，则该寄存器是写锁定的。

26.5 寄存器定义

26.5.1 寄存器列表

CAN 基地址：0x4003 3000

偏移	实例地址	名称	默认值	描述
0x00	0x40033000	CAN_RBUF	0x00	CAN RXFIFO 寄存器
0x48	0x40033048	CAN_TBUF	0x00	CAN TXFIFO 寄存器
0x90	0x40033090	CAN_CFG_STAT	0x80	CAN 控制和状态寄存器
0x91	0x40033091	CAN_TCMD	0x00	CAN 命令寄存器
0x92	0x40033092	CAN_TCTRL	0x00	CAN 发送控制寄存器
0x93	0x40033093	CAN_RCTRL	0x20	CAN 接收控制寄存器
0x94	0x40033094	CAN_RTIE	0xFE	CAN 发送接收中断使能寄存器
0x95	0x40033095	CAN_RTIF	0x00	CAN 发送接收中断寄存器
0x96	0x40033096	CAN_ERRINT	0x00	CAN 错误中断使能和中断寄存器
0x97	0x40033097	CAN_LIMIT	0x1B	CAN 警告极限寄存器
0x98	0x40033098	CAN_BITTIME0	0x83	CAN 位时序寄存器 0
0x99	0x40033099	CAN_BITTIME1	0x42	CAN 位时序寄存器 1
0x9A	0x4003309A	CAN_BITTIME2	0x32	CAN 位时序寄存器 2
0x9C	0x4003309C	CAN_S_PRESC	0x01	CAN 慢速预分频器寄存器
0x9D	0x4003309D	CAN_F_PRESC	0x01	CAN 快速预分频器寄存器
0xA0	0x400330A0	CAN_EALCAP	0x00	CAN 错误和仲裁丢失捕获寄存器
0xA2	0x400330A2	CAN_RECNT	0x00	CAN 接收错误计数寄存器
0xA3	0x400330A3	CAN_TECNT	0x00	CAN 发送错误计数寄存器
0xA4	0x400330A4	CAN_ACFCTRL	0x00	CAN 验收滤波器控制寄存器
0xA6	0x400330A6	CAN_ACF_EN_0	0x01	CAN 验收过滤器使能寄存器 0
0xA7	0x400330A7	CAN_ACF_EN_1	0x00	CAN 验收过滤器使能寄存器 1
0xA8	0x400330A8	CAN_ACF	0x00	CAN 验收过滤器寄存器

26.5.2 寄存器描述

26.5.2.1 CAN 控制和状态寄存器（CAN_CFG_STAT）

- **名称：CAN Configuration and Status Register**
- **偏移地址：0x90**
- **默认值：0x80**
- 返回寄存器列表

位	7	6	5	4	3	2	1	0
名称	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x1	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] RESET	**RESET 位** 0：无复位 1：CPU 执行 CAN-CTRL 的本地复位。 仅当 RESET = 1 时才能修改节点配置（BITTIME_x，x_PRESC，接收代码和掩码）。CAN-CTRL 内核不断监听总线。验收过滤器可能会导致不可预测的不良行为。因此，仅当 CAN 模块复位并因此与总线断开连接时，才能更改这些参数。 RESET 位将使多个组件进入复位状态。如果节点进入“总线关闭”状态，则会自动设置 RESET。 复位后，经过 11 个连续的隐型位，CAN 将恢复到协议规定的 ERROR-ACTIVE 状态，并参与 CAN 通信。CAN 标准要求此延迟。如果将 RESET 设置为 1 并立即设置为 0，则需要一些时间才能将 RESET 读取为 0 并变为无效状态。原因是时钟域从主机跨越到 CAN 时钟域。根据主机和 CAN 时钟之间的关系，只要需要，RESET 就会保持激活状态。
[6] LBME	**外部环回模式** 0：禁用 1：已启用 传输处于活动状态时，不应启用 LBME。
[5] LBMI	**内部环回模式** 0：禁用 1：已启用 传输处于活动状态时，不应启用 LBMI。
[4] TPSS	**PTB 的传输主要单发模式** 0：禁用 1：已启用
[3] TSSS	**STB 的传输辅助单发模式** 0：禁用 1：已启用

[2] RACTIVE	**接收激活（接收状态位）** 0：没有接收活动。 1：控制器当前正在接收数据或远程帧。
[1] TACTIVE	**传输激活（传输状态位）** 0：没有发送活动。 1：控制器当前正在传输数据或远程帧。
[0] BUSOFF	**总线关闭（总线状态位）** 0：控制器状态为“ bus on”。 1：控制器状态为“总线关闭”。

26.5.2.2 CAN 命令寄存器（CAN_TCMD）

- **名称：CAN Command Register**
- **偏移地址：0x91**
- **默认值：0x00**
- 返回寄存器列表

位	7	6	5	4	3	2	1	0
名称	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] TBSEL	**发送缓冲器选择** 选择要加载消息的发送缓冲区。使用 TBUF 寄存器进行访问。始终在写 TBUF 寄存器以及设置 TSNEXT 时 TBSEL 必须保持稳定。 0：PTB（高优先级缓冲区） 1：STB（类似于 FIFO）
[6] LOM	**仅听模式** 0：禁用 1：已启用 传输处于活动状态时，不应启用 LOM。如果启用了 LOM，则无法开始传输。
[5] STBY	**收发器待机模式** 0：禁用 1：已启用 该寄存器位连接到输出信号 stby，可用于控制收发器的待机模式。 如果 TPE = 1，TSONE = 1 或 TSALL = 1，则 STBY 不能设置为 1。 如果主机将 STBY 设置为 0，则主机需要等待收发器启动所需的时间，然后主机才能请求新的传输

[4] TPE	**传输 PTB 启用** 0：PTB 无传输 1：高优先级 PTB 消息的传输使能 如果设置了 TPE，则来自 PTB 的消息将在下一个可能的发送位置发送。从机开始的传输将在此之前完成，但是待发送的新消息会延迟到 PTB 消息完成之后。 TPE 保持设置状态，直到成功发送了消息或使用 TPA 中止消息为止。仅主机控制器可以设置 TPE。 主机控制器可以将 TPE 设置为 1，但不能将其重置为 0。这只能通过使用 TPA 并中止消息来实现。 在短时间内，CAN-CTRL 内核会重置该位，主机无法设置该位。 如果 RESET = 1，BUSOFF = 1，STBY = 1 或 LOM = 1，则该位将重置为硬件复位值。
[3] TPA	**传送 PTB 中止** 0：不中止 1：中止 TPE = 1 请求但尚未开始的来自 TPB 的传输。（消息的数据字节保留在 PTB 中。） 该位必须由主机控制器设置，并将通过 CAN-CTRL 复位。设置 TPA 会自动使 TPE 无效。 主机控制器可以将 TPA 设置为 1，但不能将其重置为 0。 在短时间内，CAN-CTRL 内核会重置该位，主机无法设置该位。 如果 RESET = 1 或 BUSOFF = 1，则该位将重置为硬件重置值。TPA 不应与 TPE 同时设置。
[2] TSONE	**传输下一帧** 0：STB 没有传输。 1：启用 STB 中最早的消息的传输。消息格式存储在该消息所属的 IDE 缓冲区中。一旦总线空闲并且没有待处理的 PTB（位 TPE）请求，控制器就开始传输。 TSONE 保持设置状态，直到成功发送了消息或使用 TSA 中止消息为止。只有主机控制器可以设置 TSONE。 主机控制器可以将 TSONE 设置为 1，但不能将其重置为 0。这只能通过使用 TSA 并中止消息来实现。 在短时间内，CAN-CTRL 内核会重置该位，主机无法设置该位。 如果 RESET = 1，BUSOFF = 1，STBY = 1 或 LOM = 1，则该位将重置为硬件复位值。
[1] TSALL	**传输所有辅助帧** 0：STB 没有传输。 1：启用 STB 中所有消息的传输。消息格式存储在适当的 IDE 缓冲区中。一旦总线空闲并且没有待处理的 PTB（位 TPE）请求，控制器就开始传输。 TSALL 保持设置状态，直到所有消息都已成功发送或使用 TSA 中止为止。仅 CPU 可以设置 TSALL。 主机控制器可以将 TSALL 设置为 1，但不能将其重置为 0。这只能使用 TSA 并中止消息来实现。 在短时间内，CAN-CTRL 内核会重置该位，CPU 无法设置该位。 如果 RESET = 1 或 BUSOFF = 1 或 STBY = 1 或 LOM = 1，则该位将重置为硬件复位值。 如果在传输期间向机顶盒加载了新帧，则新帧也将被传输。换句话说：当 STB 变空时，由 TSALL 发起的传输完成。 如果 STBY = 1，则 TSALL 不能设置为 1。

[0] TSA	**传输二次中止** 0：不中止 1：中止来自 STB 的已请求但尚未开始的传输。对于 TSONE 传输，仅中止一帧，而对于 TSALL 传输，所有帧均被中止。将释放一个或所有消息槽，这些消息槽将更新 TSSTAT。由于 STB 的类似于 FIFO 的行为，所有中止的消息都将丢失，因为它们不再可访问。 该位必须由主机控制器设置，并将通过 CAN-CTRL 复位。设置 TSA，分别自动取消激活 TSONE 或 TSALL。 主机控制器可以将 TSA 设置为 1，但不能将其重置为 0。 如果 RESET = 1 或 BUSOFF = 1，则该位将重置为硬件重置值。 TSA 不应与 TSONE 或 TSALL 同时设置。

26.5.2.3 CAN 发送控制寄存器（CAN_TCTRL）

- **名称：CAN Transmit Control Register**
- **偏移地址：0x92**
- **默认值：0x00**
- 返回寄存器列表

位	7	6	5	4:0
名称		TSNEXT		TSSTAT
访问		R/W		R
复位值		0x0		0x0

字段	说明
[6] TSNEXT	**发送缓冲区下一个使能** 0：无动作 1：STB 插槽已满，选择下一个 FIFO 插槽。 将所有帧字节写入 TBUF 寄存器后，主机控制器必须将 TSNEXT 设置为 1，表明该插槽已满。然后，CAN-CTRL 内核将 TBUF 寄存器连接到下一个 FIFO 插槽。一旦将插槽标记为已填充，就可以使用 TSONE 或 TSALL 开始传输。 可以在一次写访问中一起设置 TSNEXT 和 TSONE 或 TSALL。 TSNEXT 必须由主机控制器设置，并在设置后立即由 CAN-CTRL 内核自动重置。 仅当 TBSEL = 1 选择了 STB 时，设置 TSNEXT 才有意义。如果 TBSEL = 0，则 TSNEXT 被忽略并自动清除。它没有任何危害。 如果所有插槽均已填满，则 TSNEXT 保持置位状态，直到一个插槽可用为止。
[4:0] TSSTAT	传输辅助状态位已填充消息缓冲区的数量（0…16）。

26.5.2.4 CAN 接收控制寄存器（CAN_RCTRL）

- 名称：CAN **Receive Control Register**
- 偏移地址：**0x93**
- 默认值：**0x20**
- 返回寄存器列表

位	7:6	5	4	3:2	1:0
名称		ROV	RREL		RSTAT
访问		R/W	R/W		R
复位值		0x1	0x0		0x0

字段	说明
[5] ROV	**接收缓冲区 OVerflow** 0：无溢出。 1：溢出。至少丢失一条消息。 通过设置 RREL = 1 清除 ROV。
[4] RREL	**接收缓冲区释放** 主机控制器确认已清空实际的 RB 插槽。然后，CAN-CTRL 核心指向下一个 RB 插槽。RSTAT 更新。 0：no release 1：release：主机已清空 RB。
[1:0] RSTAT	**接收缓冲区状态** 00：空 01：非空非满（AFWL） 10：大于阈值且无溢出（AFWL 可编程阈值） 11：满（如果发生溢出，则设置停留时间–有关溢出信号，请参阅 ROV）

26.5.2.5 CAN 发送接收中断使能寄存器（CAN_RTIE）

- 名称：CAN Receive and Transmit Interrupt Enable Register
- 偏移地址：0x94
- 默认值：0xFE
- 返回寄存器列表

位	7	6	5	4	3	2	1	0
名称	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x0

字段	说明
[7] RIE	**接收中断使能** 0：Disabled 1：Enabled
[6] ROIE	**RB 溢出中断使能** 0：Disabled 1：Enabled
[5] RFIE	**RB 全中断使能** 0：Disabled 1：Enabled
[4] RAFIE	**RB 几乎全中断使能** 0：Disabled 1：Enabled
[3] TPIE	**传输主中断使能** 0：Disabled 1：Enabled
[2] TSIE	**传输辅助中断使能** 0：Disabled 1：Enabled
[1] EIE	**错误中断使能** 0：Disabled 1：Enabled
[0] TSFF	**发送辅助缓冲区满标志** 0：The STB is not filled with the maximal count of messages 1：The STB is filled with the maximal count of messages.

26.5.2.6 CAN 发送接收中断寄存器 （CAN_RTIF）

- **名称：CAN Receive and Transmit Interrupt Flag Register**
- **偏移地址：0x95**
- **默认值：0x00**
- 返回寄存器列表

位	7	6	5	4	3	2	1	0
名称	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
访问	R	R	R	R	R	R	R	R
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] RIF	**接收中断标志** 0：没有接收到帧。 1：数据或远程帧已被接收，并且在接收缓冲区中可用。
[6] ROIF	**RB 溢出中断标志** 0：没有 RB 被覆盖。 1：RB 中至少覆盖了一条接收到的消息。 如果发生超限，将同时设置 ROIF 和 RFIF。
[5] RFIF	**RB 全中断标志** 0：RB FIFO 未满。 1：所有 RB 已满。如果在接收到下一条有效消息之前没有释放任何 RB，则最早的消息将丢失。
[4] RAFIF	**RB 几乎全中断标志** 0：已填充的 RB 插槽数＜AFWL_i 1：已填充的 RB 插槽数量≥AFWL_i
[3] TPIF	**传输主中断标志** 0：PTB 的传输尚未完成。 1：请求的 PTB 传输已成功完成。
[2] TSIF	**传输辅助中断标志** 0：未成功完成 STB 的传输。 1：请求的 STB 传输已成功完成。
[1] EIF	**错误中断标志** 0：不变。 1：错误警告限制的边界已沿任一方向越过，或者 BUSOFF 位已沿任一方向被改变。
[0] AIF	**中止中断标志** 0：没有执行中止。 1：设置 TPA 或 TSA 后，相应的消息已中止。建议不要同时设置 TPA 和 TSA，因为这两个源都是 AIF。 AIF 没有关联的使能寄存器。

26.5.2.7 CAN 错误中断使能和中断寄存器（CAN_ERRINT）

- 名称：CAN Error Interrupt Enable and Flag Register
- 偏移地址：0x96
- 默认值：0x00
- 返回寄存器列表

位	7	6	5	4	3	2	1	0
名称	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF
访问	R	R	R/W	R	R/W	R	R/W	R
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[7] EWARN	**达到错误警告限制** 0：两个计数器中的值均小于 EWL 1：错误计数器 RECNT 或 TECNT 之一等于或大于 EWL
[6] EPASS	**错误被动模式有效** 0：不活动（节点错误活动） 1：主动（节点是错误被动）
[5] EPIE	**错误被动中断使能**
[4] EPIF	**错误被动中断标志** 如果错误状态从主动错误变为被动错误，反之亦然，并且允许了该中断，则将激活 EPIF。
[3] ALIE	**仲裁丢失中断使能**
[2] ALIF	**仲裁丢失中断标志**
[1] BEIE	**总线错误中断使能**
[0] BEIF	**总线错误中断标志**

26.5.2.8 CAN 警告极限寄存器（CAN_LIMIT）

- **名称：CAN Warning Limits Register**
- **偏移地址：0x97**
- **默认值：0x1B**
- 返回寄存器列表

位	7:4	3:0
名称	AFWL	EWL
访问	R/W	R/W
复位值	0x1	0xB

字段	说明
[7:4] AFWL	**接收缓冲区几乎满警告限制** AFWL 定义内部警告限制 AFWL_i，其中 n_{RB} 为可用 RB 插槽的数量。 $AFWL\_i = \begin{cases} AFWL \mid n_{RB} < 16 \\ 2 \cdot AFWL \mid 16 \le n_{RB} < 32 \\ 4 \cdot AFWL \mid 32 \le n_{RB} < 64 \end{cases}$ 将 AFWL_i 与已填充的 RB 时隙数进行比较，如果相等，则触发 RAFIF。AFWL_i 的有效范围。 AFWL_i 是没有意义的，会自动视为 0x1。 （请注意，AFWL 在此规则中是指的，而不是 AFWL_i。） AFWL_i > n_{RB} 是没有意义的，会自动视为 nRB。 AFWL_i = n_{RB} 是有效值，但请注意，RFIF 也存在。
[3:0] EWL	可编程错误警告极限=（EWL + 1）* 8。可能的极限值：8，16，…128。EWL 的值控制 EIF。 需要使用 CDC 将 EWL 从主机传输到 CAN 时钟域。在传输过程中，EWL 寄存器的位将为主机写锁定几个时钟，直到 CDC 完成。

26.5.2.9 CAN 位时序寄存器 0（CAN_BITTIME0）

- **名称：CAN Bit Timing Register0**
- **偏移地址：0x98**
- **默认值：0x83**
- 返回寄存器列表

位	7:4	3:0
名称	F_SJW	S_Seg_1
访问	R/W	R/W
复位值	0x8	0x3

字段	说明
[7:4]	**同步跳转宽度（快速）**
F_SJW	同步跳跃宽度 $t_{SJW}=(SJW+1)\cdot TQ$ 是用于缩短或延长重新同步的位时间的最大时间，其中 TQ 是时间量。
[3:0]	**位定时段 1（慢速）**
S_Seg_1	比特时间开始后，采样点将设置为 $t_{Seg\_1}=(Seg\_1+2)\cdot TQ$。Seg_1 = 0 是没有意义的，会自动视为 1。

26.5.2.10 CAN 位时序寄存器 1（CAN_BITTIME1）

- **名称：CAN Bit Timing Register1**
- **偏移地址：0x99**
- **默认值：0x42**
- 返回寄存器列表

位	7:4	3:0
名称	F_Seg_2	S_Seg_2
访问	R/W	R/W
复位值	0x4	0x2

字段	说明
[7:4]	**位定时段 2（快速）**
F_Seg_2	采样点之后的时间 $t_{Seg\_2}=(Seg\_2+1)\cdot TQ$。 Seg_2 = 0 毫无意义，并自动视为 1。
[3:0]	**位时序段 2（低速）**
S_Seg_2	采样点之后的时间 $t_{Seg\_2}=(Seg\_2+1)\cdot TQ$。 Seg_2 = 0 毫无意义，并自动视为 1。

26.5.2.11 CAN 位时序寄存器 2（CAN_BITTIME2）

- 名称：CAN Bit Timing Register2
- 偏移地址：0x9A
- 默认值：0x32
- 返回寄存器列表

位	7:4	3:0
名称	F_Seg_1	S_SJW
访问	R/W	R/W
复位值	0x3	0x2

字段	说明
[7:4]	**位定时段 1（快速）**
F_Seg_1	比特时间开始后，采样点将设置为 $t_{Seg\_1} = (Seg\_1 + 2)\cdot TQ$。
	Seg_1 = 0 是没有意义的，会自动视为 1。
[3:0]	**同步跳跃宽度（慢速）**
S_SJW	同步跳转宽度 $t_{SJW} = (SJW + 1)\cdot TQ$ 是用于缩短或延长重新同步的位时间的最大时间，其中 TQ 是时间量

26.5.2.12 CAN 慢速预分频器寄存器（CAN_S_PRESC）

- 名称：CAN Standard Prescaler Registers
- 偏移地址：0x9C
- 默认值：0x01
- 返回寄存器列表

位	7:0
名称	S_PRESC
访问	R/W
复位值	0x1

字段	说明
[7:0]	**预分频器（慢速和快速）**
S_PRESC	预分频器将系统时钟分频以获得时间量子时钟 tq_clk。
	有效范围 PRESC = [0x01，0xff]导致分频器值为 2 到 256。
	禁止 PRESC = 0 并自动将其视为 1。（这是必需的，因为此内核的位时序逻辑需要它。）

26.5.2.13 CAN 快速预分频器寄存器（CAN_F_PRESC）

- **名称：CAN Fast Prescaler Registers**
- **偏移地址：0x9D**
- **默认值：0x01**
- 返回寄存器列表

位	7:0
名称	F_PRESC
访问	R/W
复位值	0x1

字段	说明
[7:0] F_PRESC	**预分频器（慢速和快速）** 预分频器将系统时钟分频以获得时间量子时钟 tq_clk。 有效范围 PRESC = [0x01，0xff]导致分频器值为 2 到 256。 禁止 PRESC = 0 并自动将其视为 1。（这是必需的，因为此内核的位时序逻辑需要它。）

26.5.2.14 CAN 错误和仲裁丢失捕获寄存器（CAN_EALCAP）

- **名称：CAN Error and Arbitration Lost Capture Register**
- **偏移地址：0xA0**
- **默认值：0x00**
- 返回寄存器列表

位	7:5	4:0
名称	KOER	ALC
访问	R	R
复位值	0x0	0x0

字段	说明
[7:5] KOER	**错误代码** 000：没有错误　001：位错误　010：格式错误 011：资料错误　100：确认错误　101：CRC 错误 110：其他错误 （自己的错误标志后的主要位，接收到的错误标志的时间太长，ACK 错误后在 Passive-Error-Flag 期间的主要位） 111：未使用 成功发送或接收帧后，将重置 KOER。
[4:0] ALC	仲裁丢失捕获（丢失仲裁的帧中的位位置）

26.5.2.15 CAN 接收错误计数寄存器（CAN_RECNT）

- **名称：CAN Receive Error Count Register**
- **偏移地址：0xA2**
- **默认值：0x00**
- 返回寄存器列表

位	7:0
名称	RECNT
访问	R
复位值	0x0

字段	说明
[7:0]	**接收错误计数（接收期间的错误数）**
RECNT	RECNT 按照 CAN 规范中的定义递增和递减。
	RECNT 不会溢出。RECNT 信号 0xff＝255 作为最大值。

26.5.2.16 CAN 发送错误计数寄存器（CAN_TECNT）

- **名称：CAN Transmit Error Count Register**
- **偏移地址：0xA3**
- **默认值：0x00**
- 返回寄存器列表

位	7:0
名称	TECNT
访问	R
复位值	0x0

字段	说明
[7:0]	**传输错误计数（传输期间的错误数）**
TECNT	TECNT 按照 CAN 规范中的定义递增和递减。
	TECNT 不溢出。TECNT 信号 0xff＝255 作为最大值。

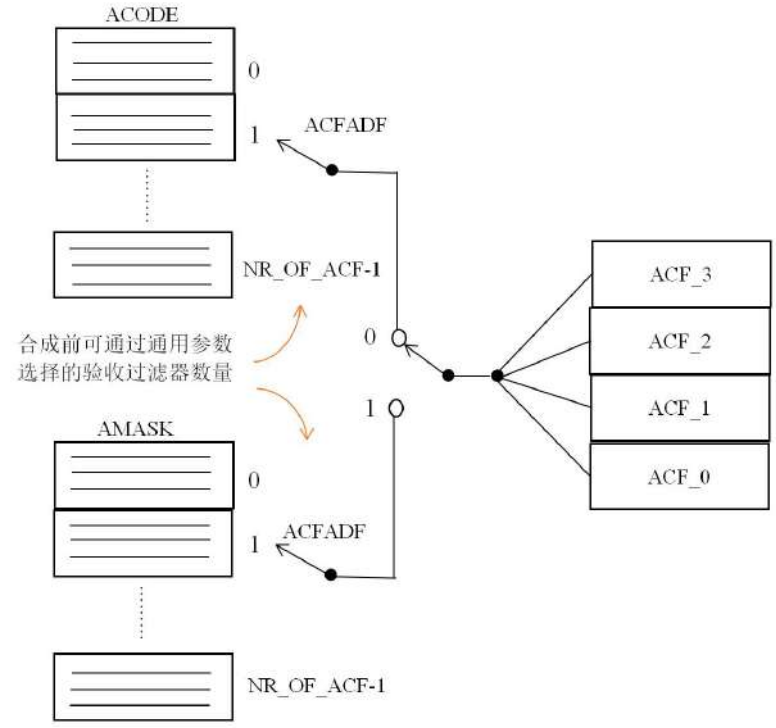
26.5.2.17 CAN 验收滤波器控制寄存器（CAN_ACFCTRL）

- **名称：CAN Acceptance Filter Control Register**
- **偏移地址：0xA4**
- **默认值：0x00**
- 返回寄存器列表

位	7:6	5	4	3:0
名称		SELMASK		ACFADR
访问		R/W		R/W
复位值		0x0		0x0

字段	说明
[5] SELMASK	**选择接收 MASK** 0：将 ACF_x 寄存器注册为接收代码 1：将 ACF_x 点注册到接收掩码。 ACFADR 选择一种特定的验收过滤器。
[3:0] ACFADR	**验收过滤器地址** ACFADR 指向特定的验收滤波器。所选过滤器可通过寄存器 ACF_x 访问。SELMASK 位在接收代码和所选验收滤波器的掩码之间进行选择。 ACFADR> NR_OF_ACF-1 的值是没有意义的，并自动视为值 NR_OF_ACF-1。

验收滤波器寄存器 ACF_x 提供对验收滤波器代码 ACODE_x 和验收滤波器掩码 AMASK_x 的访问，具体取决于 SELMASK 的设置。仅当 RESET = 1 时，才可以对 ACF_x 进行读/写访问。如果 RESET = 0，则从 ACF_x 读取结果为 0。



26.5.2.18 CAN 验收滤波器使能寄存器 0（CAN_ACF_EN_0）

- **名称：CAN Acceptance Filter Enable0 Register**
- **偏移地址：0xA6**
- **默认值：0x01**
- 返回寄存器列表

位	7:0
名称	AE_0
访问	R/W
复位值	0x1

字段	说明
[7:0]	**验收过滤器启用**
AE_0	0：禁用过滤器 1：启用验收滤波器 每个验收过滤器（AMASK / ACODE）可以单独启用或禁用。硬件重置后，默认情况下仅启用过滤器编号 0。 禁用的过滤器拒绝消息。如果适当的 AMASK / ACODE 配置匹配，则只有启用的过滤器才能接收消息。要接收所有消息，必须通过设置 AE_x = 1，AMASK_x = 0xff 和 ACODE_x = 0x00 启用一个过滤器 x。这是过滤器 x = 0 进行硬件重置后的默认配置，同时禁用了所有其他过滤器。

26.5.2.19 CAN 验收滤波器使能寄存器 1 （CAN_ACF_EN_1）

- **名称：CAN Acceptance Filter Enable 1 Register**
- **偏移地址：0xA7**
- **默认值：0x00**
- 返回寄存器列表

位	7:0
名称	AE_1
访问	R/W
复位值	0x0

字段	说明
[7:0]	**验收过滤器启用**
AE_1	0：禁用过滤器 1：启用验收滤波器 每个验收过滤器（AMASK / ACODE）可以单独启用或禁用。禁用的过滤器拒绝消息。如果适当的 AMASK / ACODE 配置匹配，则只有启用的过滤器才能接收消息。

26.5.2.20 CAN 验收过滤器寄存器（CAN_ACF）

- 名称：CAN Acceptance Filter Register
- 偏移地址：0xA8
- 默认值：0x00
- 返回寄存器列表

位	31	30	29	28:0
名称		AIDEE	AIDE	ACF
访问		R/W	R/W	R/W
复位值		0x0	0x0	0x0

字段	说明
[30] AIDEE	**接收掩码 IDE 位检查使能** 0：验收滤波器接收标准或扩展帧 1：验收滤波器接收 AIDE 定义的标准或扩展 上电复位仅影响滤波器 0。所有其他过滤器保持未初始化状态。
[29] AIDE	**接收掩码 IDE 位值** 如果 AIDEE = 1，则： 0：验收滤波器仅接收标准帧 1：验收滤波器仅接收扩展帧 上电复位仅影响滤波器 0。所有其他过滤器保持未初始化状态。
[28:0] ACF	ACODE_x 或者 AMASK_x

27 正交编码器接口（QEI）

27.1 简介

QEI 模块通常配合编码器来获取位置、方向及转速信息，主要包含如下几个单元：正交解码单元（QDU）；位置计数器及控制单元（PCCU）；边沿捕获单元（QCAP）；定时器基准单元（UTIME）。

27.2 结构框图

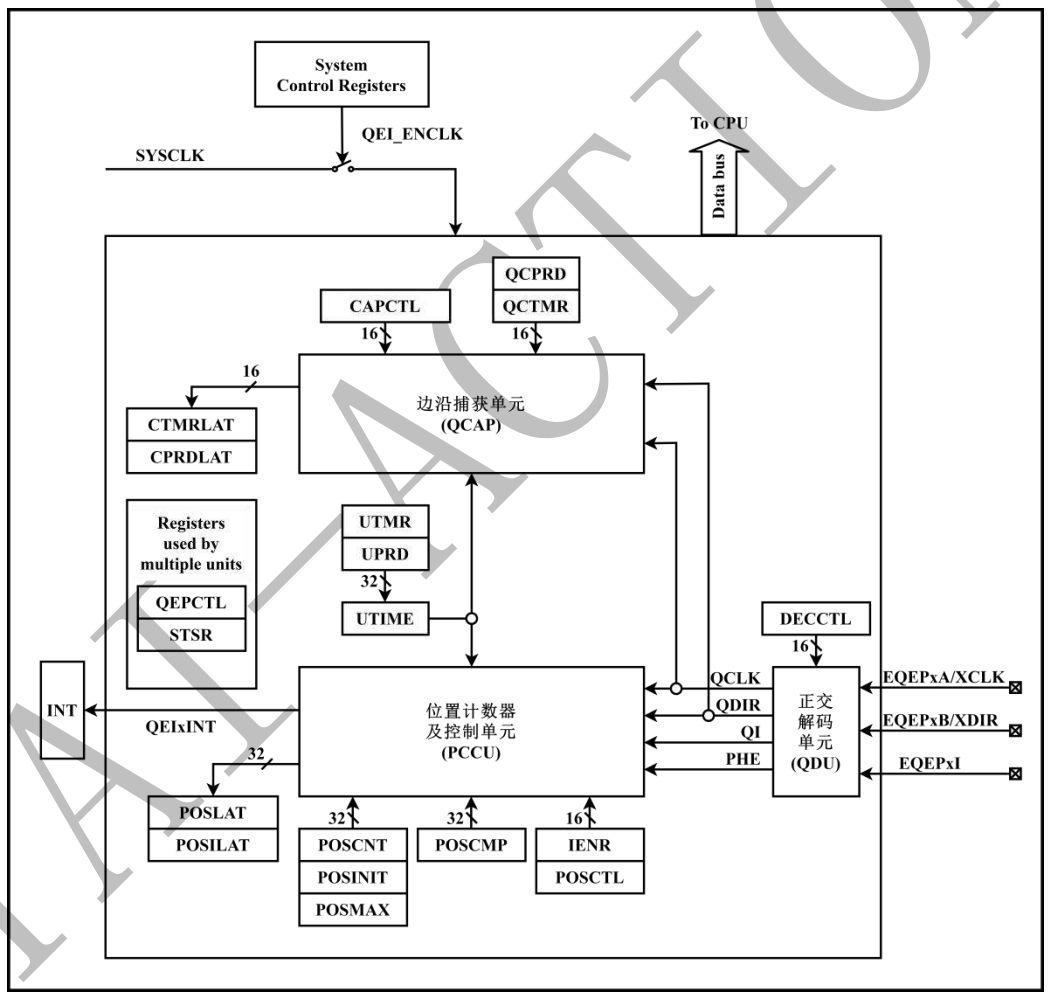


图 27-1 QEI 结构框图

27.3 主要特性

QEI 模块的输入信号主要有如下 3 路：

- QEPA/XCLK 与 QEPB/XDIR
  - 正交时钟模式
    在正交时钟模式下，QEI 提供两路互差 90°的脉冲信号 QEPA 及 QEPB，用两者之间的相位关系可判断旋转方向，脉冲信号的频率可判断转速。
  - 方向计数模式
    在方向计数模式下，方向以及脉冲信号分别由 XDIR 及 XCLK 单独提供。
- QEPI
  编码器通过索引脉冲信号 QEPI 来表明绝对起始地址，这路信号在每个旋转周期内用来复位芯片内部 QEI 模块的计数器。

27.4 功能描述

27.4.1.1 位置计数器的输入模式

位置计数器的输入模式由 QEI_DECCTL.QSRC 来决定，包括以下 4 种模式：正交计数模式、方向计数模式、增计数模式及减计数模式，以下将对这几种技术模式做详细的介绍。

正交计数模式

在正交计数模式下，方向判断逻辑电路通过判断 QEPA 及 QEPB 之间的相位关系来获得方向，并储存在 QEI_STSR.QDS 中。表 27-1 和图 27-2 分别以真值表和状态机方式来说明方向判断逻辑电路的工作过程。

表 27-1 方向判断逻辑电路真值表

上次边沿	当前边沿	QDIR	QPOSCNT
QA↑	QB↑	UP	增加
	QB↓	DOWN	减小
	QA↓	TOGGLE	增加或减小
QA↓	QB↓	UP	增加
	QB↑	DOWN	减小
	QA↑	TOGGLE	增加或减小
QB↑	QA↑	DOWN	减小
	QA↓	UP	增加
	QB↓	TOGGLE	增加或减小
QB↓	QA↓	DOWN	减小
	QA↑	UP	增加
	QB↑	TOGGLE	增加或减小

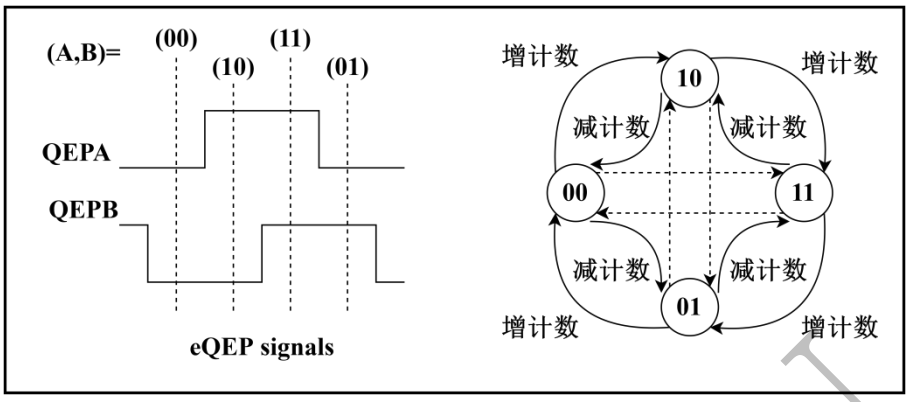


图 27-2　正交解码状态机

QEPA 及 QEPB 的上升沿和下降沿都将作为位置计数器的触发事件，因此 QEI 逻辑产生的计数脉冲的频率是每个输入脉冲的 4 倍，图 27-2 给出了 QEI 模块计数时钟及方向的解码逻辑。

正常情况下 QEPA 与 QEPB 之间的相位将相差 90°，当系统检测到两者之间的相位同步时，会将相位错误标志位 QEI_STSR.PHE 置位，同时会产生中断事件。

正常情况下 QEPA 连接到解码器的输入端 QA，QEPB 连接到解码器的输入端 QB，可通过设定 QEI_DECCTL.SWAP 位将两者交换，即 QEPA 接到 QB，而 QEPB 接到 QA。

方向计数模式

有些编码器直接提供方向信号以及计数时钟，在这种情况下可使用方向计数模式，QEPA 直接为位置计数器提供计数脉冲，QEPB 为位置计数器提供方向信息。当 QEPB 为高时，位置计数器在每个计数时钟的上升沿增加。当 QEPB 为低时，位置计数器在每个计数时钟的上升沿减小。

增计数模式

计数器的方向直接被硬件设定为增计数模式，位置计数器用来测量 QEPA 输入信号的频率。将 QEI_DECCTL.XCR 置位将使能 QEPA 输入的 2 个边沿都产生计数脉冲，从而将检测精度提高一倍。

减计数模式

计数器的方向直接被硬件设定为减计数模式，位置计数器用来测量 QEPA 输入信号的频率。将 QEI_DECCTL.XCR 置位将使能 QEPA 输入的 2 个边沿都产生计数脉冲，从而将检测精度提高一倍。

27.4.1.2　QEI 输入极性选择

QEI 模块的每路输入信号都可通过 QEI_DECCTL[3:0]位来控制输入极性，可以配置对应的位来对输入信号进行取反，例如，将 QEI_DECCTL.QIP 置位会将索引脉冲取反。

27.4.1.3 位置比较同步输出功能

QEI 模块包括一个位置比较单元，当位置计数寄存器 QEI_POSCNT 的值与位置比较寄存器 QEI_POSCMP 的值相等时会产生一个同步信号。

27.4.2 位置计数器及控制单元

位置计数器及控制单元（PCCU）通过两个寄存器 QEI_QEPCTL 与 QEI_POSCTL 来设定位置计数器的运行模式、初始化/锁存模式以及位置比较同步信号的产生。

27.4.2.1 位置计数器的运行模式

在一些系统中，位置计数器在多个旋转周期内连续累加，提供了相对初始位置的位移量。例如，将正交编码器安装在打印机的电机上，每次将打印机机头移动到初始位置时，位置计数器复位，位置计数器中的值随机头的移动而增加，从而记录了打印机机头相对初始位置移动的绝对距离。在其他系统中，位置计数器的值在每个旋转周期内由索引脉冲复位，位置计数器的值提供了相对索引脉冲位置的角度。

位置计数器可配置成如下 4 种运行模式：

1. 位置计数器在索引脉冲到来时发生复位
2. 位置计数器在达到最大计数值时复位，
3. 位置计数器仅在第一个索引脉冲到来时复位
4. 位置计数器在单位时间输出事件时复位（频率测量）

以上所有运行模式中，计数器在增加到 QEI_POSMAX 时，如果下个计数脉冲到来，将复位到 0；计数器在减计数到 0 时，如果下个计数脉冲到来，将复位到 QEI_POSMAX。

位置计数器在索引脉冲到来时发生复位（QEI_QEPCTL.PCRM=00）

正向运行时，如果出现索引脉冲信号，位置计数器在下一个 QEI 时钟到来时被复位到 0；反向运行时，如果出现索引脉冲信号，位置计数器在下一个 QEI 时钟到来时被复位为 QEI_POSMAX 寄存器中的值。

将第一个索引脉冲边沿到来后的正交信号的边沿定义为索引标志时刻，QEI 模块记录第一个索引标志的发生（QEI_STSR.QDF）以及第一个索引事件发生时的方向（QEI_STSR.QDI），还记录第一个索引标志对应的正交信号边沿，从而使用这个相同的正交边沿完成复位操作。例如，如果第一次复位操作发生在正向运行过程中的 QEPB 的下降沿，那么以后所有正向运行的复位操作都将发生在 QEPB 的下降沿，而反向运行的复位操作发生在 QEPB 的上升沿。

每个索引事件发生时，位置计数器的值被锁存到 QEI_POSILAT 寄存器中，运行方向也被记录到 QEI_STSR.QDS 位中。如果 QEI_POSILAT 中的值不等于 0 或 QEI_POSMAX，那么位置计数器错位标志位（QEI_STSR.PCE）将置位。位置计数器错位标志位在每次索引脉冲事件发生时进行更新，而中断标志位则必须由软件清除。

位置计数器在达到最大数值时复位（QEI_QEPCTL.PCRM=01）

正向运行时，如果位置计数器的值到达 QPOSMAX 寄存器中的值，在下一个 QEI 时钟信号到来时将位置计数器复位为 0，并且将位置计数器上溢标志位置位；反向运行时，如果位置计数器的值到达 0，在下一个 QEI 时钟信号到来时将位置计数器复位到 QPOSMAX，并将位置计数器下溢标志位置位。

位置计数器仅在第一个索引脉冲到来时复位（QEI_QEPCTL.PCRM=10）

正向运行时，如果出现索引脉冲信号，位置计数器在下一个 QEI 时钟到来时被复位到 0，反向运行时，如果出现索引脉冲信号，位置计数器在下一个 QEI 时间到来时被复位为 QEI_POSMAX 寄存器中的值。但需要注意的是，以上复位操作只发生在第一次索引事件到来时，接下来的索引事件不能将位置计数器复位，位置计数器以后的复位操作与第二种情况描述的相同。

位置计数器在 Timer 事件时复位（QEI_QEPCTL.PCRM=11）

在该模式下，当一次 Timer 事件发生时，QEI_POSCNT 的值被锁存到 QEI_POSLAT 寄存器中，并且 QEI_POSCNT 被复位到 0 或 QEI_POSMAX（这由方向控制位决定），该模式可用于频率的测量。

27.4.2.2 位置计数器的锁存

QEI 模块的索引输入（Index input）可以将位置计数器的锁存到 QEI_POSILAT 寄存器中。

索引事件锁存

许多应用中，必须在每个索引事件发生时将位置计数器的值复位且要将位置计数器运行在 32 位模式下（QEI_QEPCTL.PCRM=01 或 10）。在这种情况下，可在每个索引事件发生时将位置计数器的值以及方向进行锁存，具有如下 3 种选择。

1. 上升沿锁存（QEI_QEPCTL.IEL=01）
   位置计数器的当前值（QEI_POSCNT）在每次索引信号的上升沿被锁存到 QEI_POSILAT 寄存器中。
2. 下降沿锁存（QEI_QEPCTL.IEL=10）
   位置计数器的当前值（QEI_POSCNT）在每次索引信号的下降沿被锁存到 QEI_POSILAT 寄存器中。
3. 索引事件标志时刻锁存（QEI_QEPCTL.IEL=11）
   索引事件的标志时刻定义为索引脉冲第一个边沿后的正交信号的边沿，在这个边沿将位置计数器的当前值（QEI_POSCNT）锁存到 QEI_POSILAT 寄存器中。

位置计数器的值被锁存后，锁存事件中断标志位 QEI_STSR.IEL 被置位。

27.4.2.3 位置计数器的初始化

位置计数器可采用如下 2 种初始化方法。

1. 使用索引事件初始化
   在索引脉冲的上升沿或下降沿可对位置计数器进行初始化。如果 QEI_QEPCTL.IEI=2，将在索引脉冲的上升沿将 QEI_POSINIT 寄存器中的值装载到位置计数器 QEI_POSCNT 中；如果 QEI_QEPCTL.IEI=3，初始化过程将发生在下降沿。
2. 软件初始化
   通过向 QEI_QEPCTL.SWI 中写 1 将对位置计数器发起一次软件初始化过程。QEI_QEPCTL.SWI 位并不会自动清零，但当再次向其写 1 时会发起另一次初始化过程。

27.4.2.4 QEI 位置比较单元

QEI 模块包含一个位置比较单元，当匹配时用来产生同步输出信号和/或中断信号。

当 QEI_POSCNT=QEI_POSCMP 时即产生一次比较匹配事件，将 QEI_STSR.PCM 置位，并输出脉冲宽度可调的同步脉冲与触发外部器件。例如，如果 QEI_POSCMP=2，增计数时匹配事件将发生从 1 到 2 的跳变过程，减计数匹配事件将发生从 3 到 2 的跳变过程。

位置比较单元的脉冲扩展功能可在匹配事件发生时产生脉冲宽度可调的同步脉冲信号，如果先前输出的同步脉冲仍有效，新的匹配事件又到来，那么脉宽扩展功能将允许根据新的匹配事件产生同步脉冲信号。

27.4.3 边沿捕获单元

模块内部集成了一个边沿捕获单元，用来测量单位位移量之间的时间，利用这个模块可用下式测量低速段的转速：

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{x}{\Delta T} \qquad (3)$$

式中，单位位移量×定义为 N 个正交脉冲数。

捕获定时器 QCTMR 的计数脉冲由 QEI_CAPCTL.CCPS 位对系统总线时钟分频而来，每次 EVENT 触发脉冲都会将捕获定时器 QCTMR 中的值锁存到捕获周期寄存器 QEI_QCPRD 中，然后捕获定时器复位，QEI_STSR.CDS 置位用来表明 QEI_QCPRD 中以锁存一个新值，在软件读取捕获周期寄存器 QEI_QCPRD 的值之前可首先检查此位，向此位写 1 将对其清零。

如果满足以下两个条件，则单位位移量所经历的时间△T 测量正确。

1. 捕获定时器的值没有超过 65535
2. 而在两次事件间隔内，无转动方向的改变

如果捕获定时器的值出现上溢，上溢错误标志位 QEI_STSR.COE 将置位，如果在 2 次 EVENT 事件间隔内出现方向的改变，则错误标志位将置位。

捕获定时器 QCTMR 及捕获周期寄存器 QEI_QCPRD 的值可在如下 2 个事件发生时被锁存：

1. CPU 读 QEI_POSCNT 寄存器
2. 定时器基准单元超时事件。

定时器基准单元由一个 32 位的定时器及一个周期寄存器组成，定时器的计数时钟为系统总线时钟，当定时器的值等于周期寄存器的值时，将会产生一次超时事件，将 QEI_STSR.UTO 置位。

如果 QEI_CAPCTL.CMD=0，在 CPU 读取 QEI_POSCNT 值时，捕获定时器及捕获周期寄存器的值将会分别锁存到 QEI_CTMRLAT 及 QEI_CPRDLAT 寄存器中，如果 QEI_CAPCTL.CMD=1，在定时器基准单元超时事件发生时将把位置计数器、捕获定时器及捕获周期寄存器的值分别锁存到 QEI_POSLAT、QEI_CTMRLAT 及 QEI_CPRDLAT 寄存器中。利用此锁存功能，可用下式测量高速段的转速：

$$v(k) \approx \frac{x(k)-x(k-1)}{T} = \frac{\Delta x}{T} \qquad (4)$$

式 3，式 4 中变量所对应的硬件寄存器如表 27-2 所示。

表 27-2 硬件寄存器变量

变量	对应的硬件寄存器
T	单元周期寄存器 QUPRD
△x	增加的位移量=QPOSLAT(k)-QPOSLAT(k-1)
x	由 CAPCTL[UPPS]位定义的固定位移量
△T	捕获周期寄存器 QCPRDLAT

27.4.4 中断结构

QEI 模块可产生 11 路中断信号：PCE、PHE、QDC、PCU、PCO、PCM、IEL 及 UTO。中断控制寄存器 QEI_IENR 用来使能/禁止相应的中断事件，中断标志寄存器 QEI_STSR 用来表明各中断事件发生的情况，并且包括全局中断标志位。

27.5 寄存器描述

27.5.1 寄存器列表

QEI0 基地址：**0x4003 0000**
QEI1 基地址：**0x4003 1000**

偏移	实例地址	名称	默认值	描述
0x00	0x40030000	QEI_POSCNT	0x00000000	QEI 位置计数寄存器
0x04	0x40030004	QEI_POSINIT	0x00000000	QEI 位置初始化寄存器
0x08	0x40030008	QEI_POSMAX	0xFFFFFFFF	QEI 位置最大值寄存器
0x0C	0x4003000C	QEI_POSCMP	0x00000000	QEI 位置比较值寄存器
0x10	0x40030010	QEI_POSILAT	0x00000000	QEI 位置索引锁存寄存器
0x14	0x40030014	QEI_POSLAT	0x00000000	QEI 位置定时锁存寄存器
0x20	0x40030020	QEI_UTMR	0x00000000	QEI 定时计数寄存器
0x24	0x40030024	QEI_UPRD	0xFFFFFFFF	QEI 定时周期寄存器
0x30	0x40030030	QEI_DECCTL	0x00000000	QEI 解码控制寄存器
0x34	0x40030034	QEI_QEPCTL	0x00000000	QEI 位置控制寄存器
0x38	0x40030038	QEI_POSCTL	0x00000000	QEI 位置比较控制寄存器
0x3C	0x4003003C	QEI_CAPCTL	0x00000000	QEI 捕获控制寄存器
0x40	0x40030040	QEI_QCTMR	0x00000000	QEI 捕获计数寄存器
0x44	0x40030044	QEI_QCPRD	0x00000000	QEI 捕获周期寄存器
0x48	0x40030048	QEI_CTMRLAT	0x00000000	QEI 捕获计数锁存寄存器
0x4C	0x4003004C	QEI_CPRDLAT	0x00000000	QEI 捕获周期锁存寄存器
0x50	0x40030050	QEI_IENR	0x00000000	QEI 中断使能寄存器
0x54	0x40030054	QEI_STSR	0x00000000	QEI 中断状态寄存器

【说明】上表中，x=0，1。

27.5.2 寄存器详细描述

27.5.2.1 QEI 位置计数寄存器（QEI_POSCNT）

- 名称：QEI Position Counter Register
- 偏移地址：0x00
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	POSCNT
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 位置计数寄存器(QEI Position Counter Register)**
POSCNT	QEI 位置计数器根据旋转方向进行递增或者递减计数，QEI 位置计数器根据不同模式进行计数(详细参考模式介绍寄存器)，会根据不同触发事件进行初始化操作； Note：QEI 位置计数器在使能前完成初始化操作，开始计数后不能修改；

27.5.2.2 QEI 位置初始化寄存器（QEI_POSINIT）

- 名称：QEI Position Counter Initial Register
- 偏移地址：0x04
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	POSINIT
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 位置初始化寄存器(QEI Position Counter Initial Register)**
POSINIT	位置初始化寄存器用于在外边 Index 或者 Software 事件触发下，对 QEI 位置计数器完成初始化操作；

27.5.2.3 QEI 位置最大值寄存器（QEI_POSMAX）

- **名称：QEI Position Counter Max Register**
- **偏移地址：0x08**
- **默认值：0xFFFFFFFF**
- 返回寄存器列表

位	31:0
名称	POSMAX
访问	R/W
复位值	FFFF FFFF

字段	说明
[31:0]	**QEI 位置最大值寄存器(QEI Position Counter Max Register)**
POSMAX	位置计数器计数上限设置，当位置计数器向上计数到 Max 上限会复位到 0；或者递减计数到 0 会复位到 Max 值；

27.5.2.4 QEI 位置比较值寄存器（QEI_POSCMP）

- **名称：QEI Position Counter Compare Register**
- **偏移地址：0x0C**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:0
名称	POSCMP
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 位置比较值寄存器 (QEI Position Counter Compare Register)**
POSCMP	位置计数比较值，当位置计数器计数到 Compare 值会产生一个比较中断；

27.5.2.5 QEI 位置索引锁存寄存器（QEI_POSILAT）

- 名称：QEI Position Counter Index Latch Register
- 偏移地址：0x10
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	POSILAT
访问	R
复位值	0x0

字段	说明
[31:0]	**QEI 位置索引锁存寄存器（QEI Position Counter Index Latch Register）**
POSILAT	位置计数器 Index 事件触发锁存寄存器；

27.5.2.6 QEI 位置定时锁存寄存器（QEI_POSLAT）

- 名称：QEI Position Counter Timer Latch Register
- 偏移地址：0x14
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	POSLAT
访问	R
复位值	0x0

字段	说明
[31:0]	**QEI 位置定时锁存寄存器（QEI Position Counter Timer Latch Register）**
POSLAT	位置计数器 Timer 事件触发锁存寄存器；

27.5.2.7 QEI 定时计数寄存器（QEI_UTMR）

- 名称：QEI Timer Counter Register
- 偏移地址：0x20
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	UTMR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 定时计数寄存器（QEI Timer Counter Register）**
UTMR	Timer 计数器计数值等于 Compare 值时会产生中断与事件； Note：启动 Timer 前可以设置初始值，开始计数后不能修改；

27.5.2.8 QEI 定时周期寄存器（QEI_UPRD）

- 名称：QEI Timer Counter Period Register
- 偏移地址：0x24
- 默认值：0xFFFFFFFF
- 返回寄存器列表

位	31:0
名称	UPRD
访问	R/W
复位值	0xFFFF FFFF

字段	说明
[31:0]	**QEI 定时周期寄存器（QEI Timer Counter Period Register）**
UPRD	Timer 计数比较值，当计数值等于 Period 值时会产生中断与事件；

27.5.2.9 QEI 解码控制寄存器（QEI_DECCTL）

- 名称：QEI Decoder Control Register
- 偏移地址：0x30
- 默认值：0x00000000
- 返回寄存器列表

位	31:15		14	13:12	11:7	6
名称			DCM	QSRC		XCR
访问			R/W	R/W		R/W
复位值			0x0	0x0		0x0
位	5	4	3	2	1	0
名称	IGATE	SWAP	QBP	QAP		QIP
访问	R/W	R/W	R/W	R/W		R/W
复位值	0x0	0x0	0x0	0x0		0x0

字段	说明
[14] DCM	**QEI 方向计数模式（QEI Direction Counter Mode）** 0：模式 0 - Counter 在 QEA 边沿递增，在 QEB 边沿递减(X1 和 X2 模式)； 1：模式 1 - Counter 在 QEA 边沿增减，方向依赖于 QEB 信号，高增低减； Note： 1）仅在 QEI WorkMode 选择方向计数模式下才有效； 2）在模式 0 下，X1 模式在 QEA 上升沿递增，在 QEB 上升沿递减；X2 模式在 QEA 双沿递增，在 QEB 双沿递减； 3）在模式 1 下，X1 模式在 QEA 上升沿并且 QEB 为高递增，为低递减；X2 在 QEA 双沿并且 QEB 为高递增，为低递减；
[13:12] QSRC	**QEI 位置计数器源选择（QEI Position-counter Source Selection）** 00：正交计数模式-X1 模式在 QA 上下边沿计数，X2 模式在 QA/QB 上下边沿计数 01：方向计数模式-参考[14]位描述 10：向上计数模式- Counter 在 QEA 边沿递增，和 QEB 无关 11：向下计数模式- Counter 在 QEA 边沿递减，和 QEB 无关 Note：采用单边沿还是双边沿依据选择 X1 还是 X2 模式；"
[6] XCR	**QEI 时钟速率控制（QEI Clock Rate Control）** 0：X1 模式 1：X2 模式 Note： 1）正交计数模式，X1 模式在 QA 上下边沿计数，X2 模式在 QA/QB 上下边沿计数； 2）方向计数模式，X1 模式在 QA 上升沿计数，X2 模式在 QA/QB 上下边沿计数；
[5] IGATE	**QEI 索引信号使能（QEI Index Gate Enable）** 0：索引信号 Index 可以复位位置计数器 1：索引信号 Index 不可复位位置计数器（Gate 生效）

[4] SWAP	**QEI 正交时钟交换控制（QEI QEA/B SWAP Enable）** 0：内部不做交换处理 1：内部实现 A/B 交换
[3] QBP	**QEI QEB 信号输入极性选择（QEI QEB Input Polarity）** 0：Default 为低电平输入，高电平有效； 1：取反输入；
[2] QAP	**QEI QEA 信号输入极性选择（QEI QEA Input Polarity）** 0：Default 为低电平输入，高电平有效； 1：取反输入；
[0] QIP	**QEI Index 信号输入极性选择（QEI Index Input Polarity）** 0：Default 为低电平输入，高电平有效； 1：取反输入；

27.5.2.10 QEI 位置控制寄存器（QEI_QEPCTL）

- **名称：QEI Control Register**
- **偏移地址：0x34**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:14	13:12	11:9	8	7:6	5:4	3:2	1	0
名称		PCRM		SWI	IEI	IEL		UTE	QPE
访问		R/W		R/W1C	R/W	R/W		R/W	R/W
复位值		0x0		0x0	0x0	0x0		0x0	0x0

字段	说明
[13:12] PCRM	**QEI 位置计数器复位选择（QEI Position Counter Reset Selection）** 00：位置计数器在 Index 脉冲后的第一个正交信号边沿复位(区分正反方向)； 01：位置计数器计数到最大值复位； 10：位置计数器仅在第一个 Index 脉冲后的第一个正交信号边沿复位； 11：位置计数器在 Timer 事件时复位； Note：每种模式下均包含位置计数器出现上溢与下溢时触发复位；
[8] SWI	**QEI 位置计数器初始化使能（QEI Position Counter Software Initial Enable）** 0：Disable 1：写 1 发起初始化操作，将位置计数器初始化(初始化为 INIT 的值-0x4 寄存器)； Note：该 Bit 为写 1 自清，写 1 发起软件 Initial 操作，写 0 无效；
[7:6] IEI	**QEI 位置计数器索引初始选择（QEI Position Counter Index Initial Selection）** 0x：Disable 10：在 Index 信号的上升沿将位置计数器初始化为 Initial 值； 11：在 Index 信号的下降沿将位置计数器初始化为 Initial 值；

[5:4] IEL	**QEI 位置计数器索引锁存选择（QEI Position Counter Index Latch Selection）** 00：Disable 01：在 Index 信号的上升沿将位置计数器的值锁存； 10：在 Index 信号的下降沿将位置计数器的值锁存； 11：在 Index 信号有效后的第一个正交信号边沿锁存(区分正反方向)；
[1] UTE	**QEI Timer 使能（QEI Timer Enable）** 0：不使能 1：使能
[0] QPE	**QEI 使能（QEI Enable）** 0：不使能 1：使能

27.5.2.11 QEI 位置比较控制寄存器（QEI_POSCTL）

- **名称：QEI Position Contorl Register**
- **偏移地址：0x38**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:2	1	0
名称		PSE	CCE
访问		R/W	R/W
复位值		0x0	0x0

字段	说明
[1] PSE	**QEI 位置计数比较影子寄存器使能（QEI Position Counter Compare Shadow Enable）** 0：关闭 Compare 影子寄存器功能 1：使能 Compare 影子寄存器功能
[0] CCE	**QEI 位置计数比较使能（QEI Position Counter Compare Enable）** 0：关闭位置计数比较功能 1：使能位置计数比较功能

27.5.2.12 QEI 捕获控制寄存器（QEI_CAPCTL）

- 名称：QEI Capture Control Register
- 偏移地址：0x3C
- 默认值：0x00000000
- 返回寄存器列表

位	31:12	11:8	7	6:4	3:2	1	0
名称		UPPS		CCPS		CMD	CEN
访问		R/W		R/W		R/W	R/W
复位值		0x0		0x0		0x0	0x0

字段	说明
[11:8]	**QEI 捕获事件分频（QEI Capture Event Division）**
UPPS	0x0: QCLK/1
	0x1: QCLK/2
	0x2: QCLK/4
	……
	0xB: QCLK/2048
[6:4]	**QEI 捕获定时器分频（QEI Capture Timer Clock Division）**
CCPS	0x0: HCLK/1
	0x1: HCLK/2
	0x2: HCLK/4
	……
	0x7: HCLK/128
[1]	**QEI 捕获锁存模式（QEI Capture Latch Mode）**
CMD	0：在 CPU 读取位置计数器(0x00 寄存器)时锁存
	1：在定时器超时事件时锁存（QEI TIMER 定时器）
[0]	**QEI 捕获使能（QEI Capture Enable）**
CEN	0：关闭捕获功能
	1：使能捕获功能

27.5.2.13 QEI 捕获计数寄存器（QEI_QCTMR）

- 名称：QEI Capture Counter Register
- 偏移地址：0x40
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CTMR
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 捕获计数寄存器（QEI Capture Counter Register）**
CTMR	捕获定时计数器，基于 Capture Time Clock 进行计时，即基于分频后的时钟进行计时； Note：每次捕获更新事件会对 Counter 清 0，重新下一次计时；

27.5.2.14 QEI 捕获周期寄存器（QEI_QCPRD）

- 名称：QEI Capture Period Register
- 偏移地址：0x44
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CPRD
访问	R/W
复位值	0x0

字段	说明
[31:0]	**QEI 捕获周期寄存器（QEI Capture Period Register）**
CPRD	每次捕获更新事件会将 Capture Counter 计数值锁存，并产生中断；

27.5.2.15 QEI 捕获计数锁存寄存器（QEI_CTMRLAT）

- 名称：QEI Capture Counter Latch Register
- 偏移地址：0x48
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CTMRLAT
访问	R
复位值	0x0

字段	说明
[31:0] CTMRLAT	**QEI 捕获计数锁存寄存器（QEI Compare Counter Latch Register）** 捕获定时器的值在两种事件下可以锁存到寄存器中： 1）CPU 读取位置计数器的值； 2）TMR 定时器超时事件；

27.5.2.16 QEI 捕获周期锁存寄存器（QEI_CPRDLAT）

- 名称：QEI Capture Period Latch Register
- 偏移地址：0x4C
- 默认值：0x00000000
- 返回寄存器列表

位	31:0
名称	CPRDLAT
访问	R
复位值	0x0

字段	说明
[31:0] CPRDLAT	**QEI 捕获周期锁存寄存器（QEI Capture Period Latch Register）** 捕获定时器的值在两种事件下可以锁存到寄存器中： 1）CPU 读取位置计数器的值； 2）TMR 定时器超时事件；

27.5.2.17 QEI 中断使能寄存器（QEI_IENR）

- 名称：QEI Interrupt Enable Register
- 偏移地址：0x50
- 默认值：0x00000000
- 返回寄存器列表

位	31:11	10	9	8	7	6
名称		CDE	PRE	PIE	UTO	IEL
访问		R/W	R/W	R/W	R/W	R/W
复位值		0x0	0x0	0x0	0x0	0x0
位	5	4	3	2	1	0
名称	PCM	PCO	PCU	QDC	QPE	PCE
访问	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0x0	0x0	0x0	0x0	0x0	0x0

字段	说明
[10] CDE	**QEI Capture 捕获成功中断使能（QEI Capture Done Interrupt Enable）** 0：关闭 1：使能
[9] PRE	**QEI 位置计数器复位中断使能（QEI Position Counter Reset Interrupt Enable）** 0：关闭 1：使能
[8] PIE	**QEI 位置计数器初始化中断使能（QEI Position Counter Initial Interrupt Enable）** 0：关闭 1：使能
[7] UTO	**QEI 定时器溢出中断使能（QEI Timer Overflow Interrupt Enable）** 0：关闭 1：使能 Position Counter Unit Timer 比较匹配中断；
[6] IEL	**QEI 位置计数器锁存中断使能（QEI Position Counter Latch Interrupt Enable）** 0：关闭 1：使能
[5] PCM	**QEI 位置计数器比较中断使能（QEI Position Counter Compare Interrupt Enable）** 0：关闭 1：使能
[4] PCO	**QEI 位置计数器上溢中断使能（QEI Position Counter Overflow Interrupt Enable）** 0：关闭 1：使能
[3] PCU	**QEI 位置计数器下溢中断使能（QEI Position Counter Underflow Interrupt Enable）** 0：关闭 1：使能

[2]	**QEI 正交方向变化中断使能（QEI Direction Change Interrupt Enable）**
QDC	0：关闭
	1：使能
[1]	**QEI 相位错误中断使能（QEI Phase Error Interrupt Enable）**
QPE	0：关闭
	1：使能
[0]	**QEI 位置计数器错误中断使能（QEI Position Counter Error Interrupt Enable）**
PCE	0：关闭
	1：使能

27.5.2.18 QEI 中断状态寄存器（QEI_STSR）

- **名称：QEI Interrupt Status Register**
- **偏移地址：0x54**
- **默认值：0x00000000**
- 返回寄存器列表

位	31:17	16	15	14	13	12	11	10	9
名称		QDF	QDI	QDS	COE	CDE	FIS	CDS	PRS
访问		R	R	R	R/W1C	R/W1C	R	R/W1C	R/W1C
复位值		0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
位	8	7	6	5	4	3	2	1	0
名称	PIS	UTO	IEL	PCM	PCO	PCU	QDC	PHE	PCE
访问	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
复位值	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0

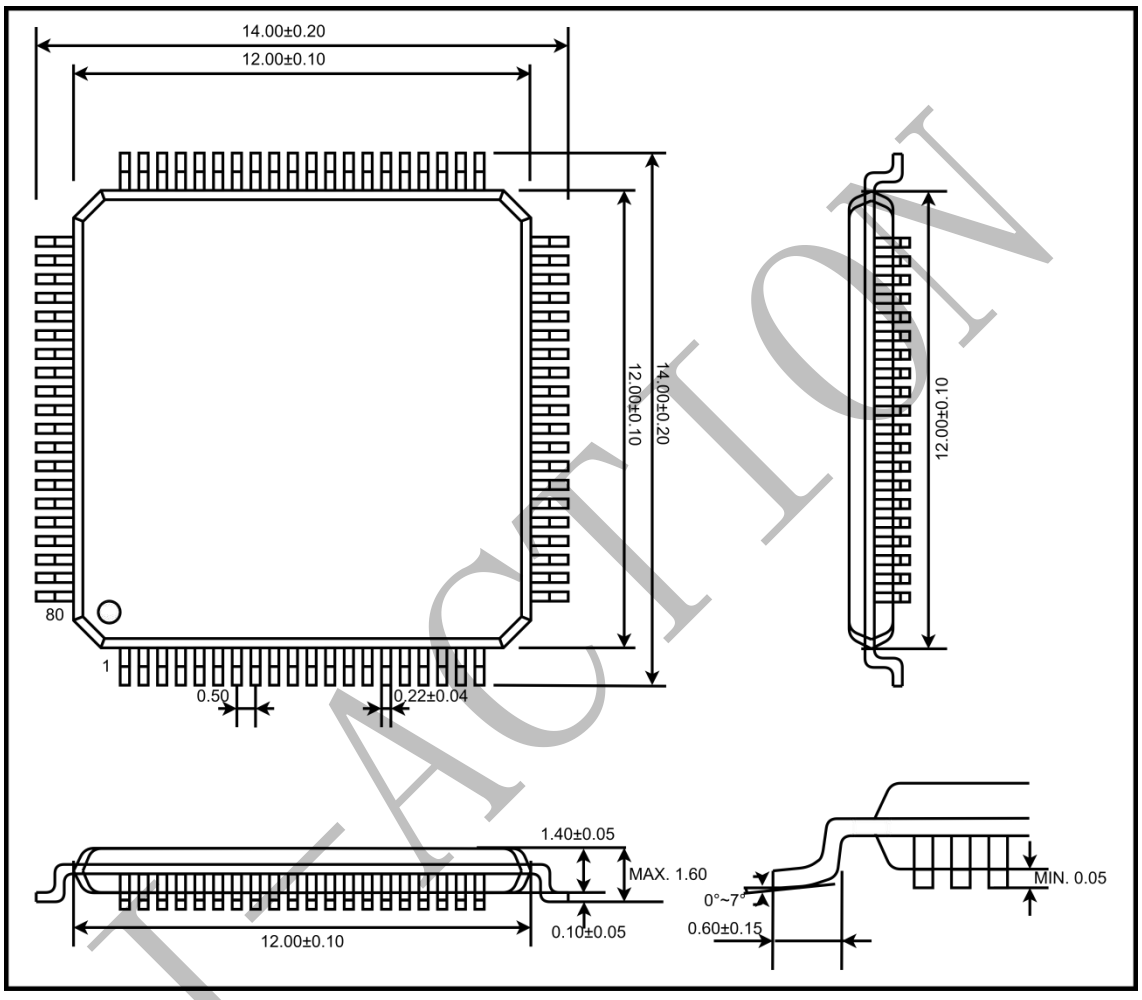
字段	说明
[16]	**QEI 首次电机转向标志位（QEI Direction First Index Status）**
QDF	0：首次索引正交脉冲有效时电机方向为反转
	1：首次索引正交脉冲有效时电机方向正转
	Note: 该位只读，写无效；
[15]	**QEI 每次电机转向标志位（QEI Direction Index Latch Status）**
QDI	0：每次索引正交脉冲有效时电机方向为反转
	1：每次索引正交脉冲有效时电机方向为正转
	Note: 该位只读，写无效；
[14]	**QEI 实时电机转向标志位（QEI Direction Status）**
QDS	0：反转
	1：正转
	Note: 该位只读，写无效；

[13] COE	**QEI 捕获上溢错误标志位（QEI Capture Overflow Error Status）** 0：无错误 1：捕获计数器出现上溢错误 Note：该位写 1 清 0；
[12] CDE	**QEI 捕获方向错误标志位（QEI Capture Direction Error Status）** 0：无错误 1：在两次捕获事件间隔内方向发生了变化 Note：该位写 1 清 0；
[11] FIS	**QEI 出现首次索引标志位（QEI The First Index Status）** 0：没出现首次索引正交脉冲 1：有出现首次索引正交脉冲
[10] CDS	**QEI Capture 捕获成功状态标志位（QEI Capture Done Interrupt Status）** 0：Capture 捕获未成功 1：Capture 捕获成功标志 Note：该位写 1 清 0；
[9] PRS	**QEI 位置计数器复位标志位（QEI Position Counter Reset Interrupt Status）** 0：位置计数器复位未触发 1：位置计数器复位触发标志 Note：该位写 1 清 0；
[8] PIS	**QEI 位置计数器初始化标志位（QEI Position Counter Initial Interrupt Status）** 0：位置计数器初始化未触发 1：位置计数器初始化触发标志 Note：该位写 1 清 0；
[7] UTO	**QEI 定时器溢出标志位（QEI Timer Overflow Interrupt Status）** 0：Timer 比较计数未完成 1：Timer 比较计数完成标志 Note：该位写 1 清 0；
[6] IEL	**QEI 位置计数器锁存标志位（QEI Position Counter Latch Interrupt Status）** 0：位置计数器锁存未触发 1：位置计数器锁存触发标志 Note：该位写 1 清 0；
[5] PCM	**QEI 位置计数器比较成功标志位（QEI Position Counter Compare Interrupt Status）** 0：位置计数比较匹配未成功 1：位置计数比较匹配成功标志 Note：该位写 1 清 0；
[4] PCO	**QEI 位置计数器上溢标志位（QEI Position Counter Overflow Interrupt Status）** 0：位置计数器未上溢 1：位置计数器上溢标志 Note：该位写 1 清 0；
[3] PCU	**QEI 位置计数器下溢标志位（QEI Position Counter Underflow Interrupt Status）** 0：位置计数器未下溢 1：位置计数器下溢标志 Note：该位写 1 清 0；

[2]	**QEI 正交方向变化标志位（QEI Direction Change Interrupt Status）**
QDC	0：正交方向未变化
	1：正交方向变化标志
	Note: 该位写 1 清 0；
[1]	**QEI 相位错误标志位（QEI Phase Error Interrupt Status）**
PHE	0：相位未错误
	1：相位错误标志
	Note：1）当 QEA/QEB 信号相位同时发生变化为相位错误；
	2）该位写 1 清 0；
[0]	**QEI 位置计数器错误标志位（QEI Position Counter Error Interrupt Status）**
PCE	0：位置计数器未错误
	1：位置计数器错误标志
	Note：1）位置计数值大于 POSMAX 值；
	2）位置计数值从 0 计数到全 F(并且 POSMAX 不为全 F)；
	3）该位写 1 清 0；

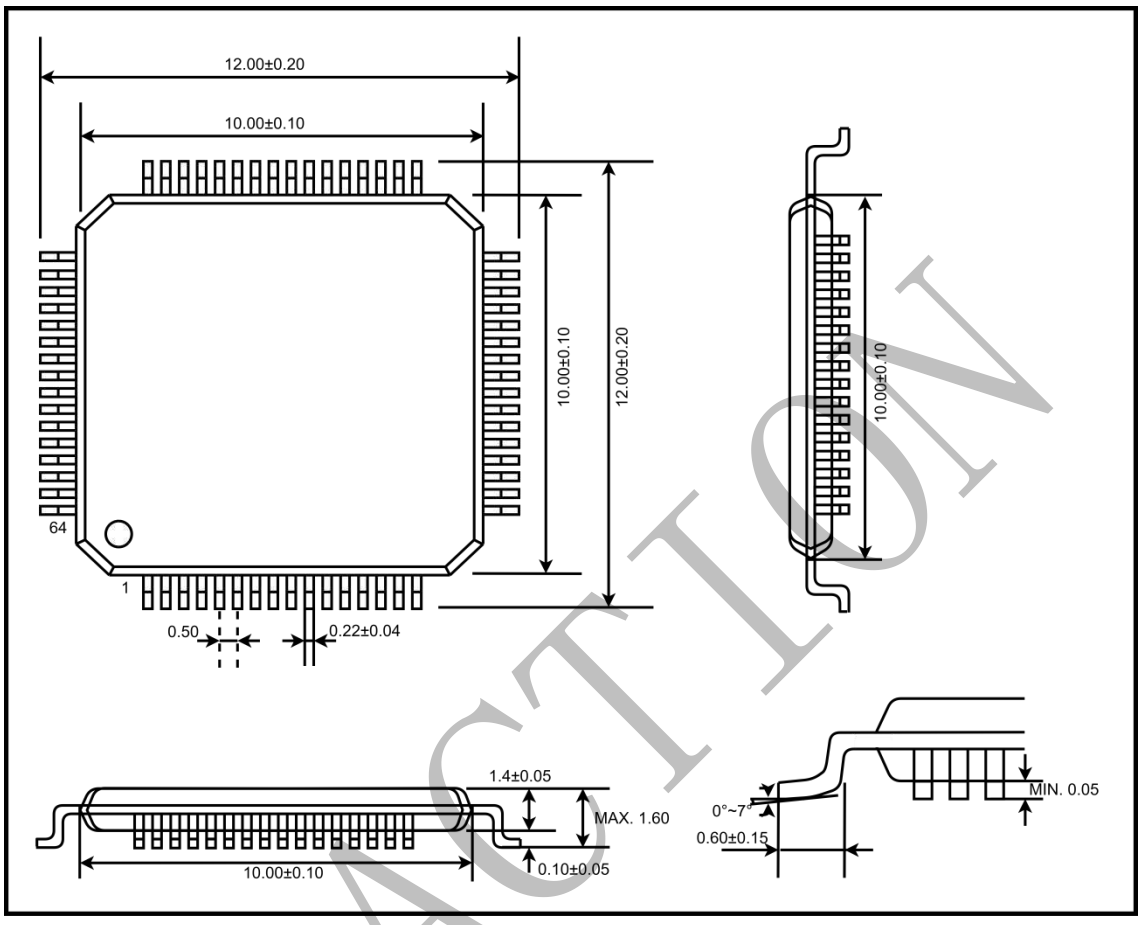
28 封装信息

LQFP 80L：



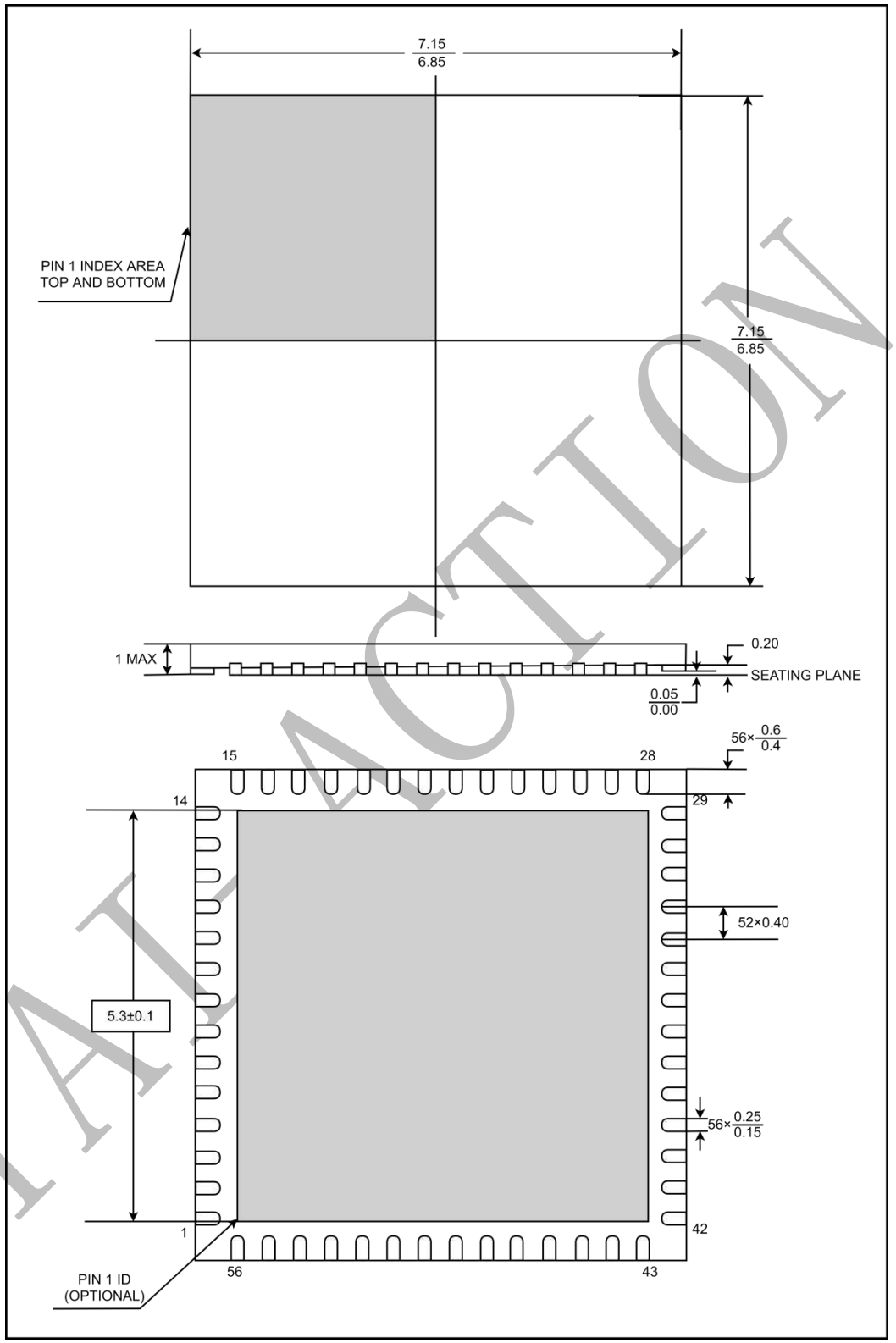
NOTE：UNITS OF MEASURE = MILLIMETER

LQFP 64L：



NOTE：UNITS OF MEASURE = MILLIMETER

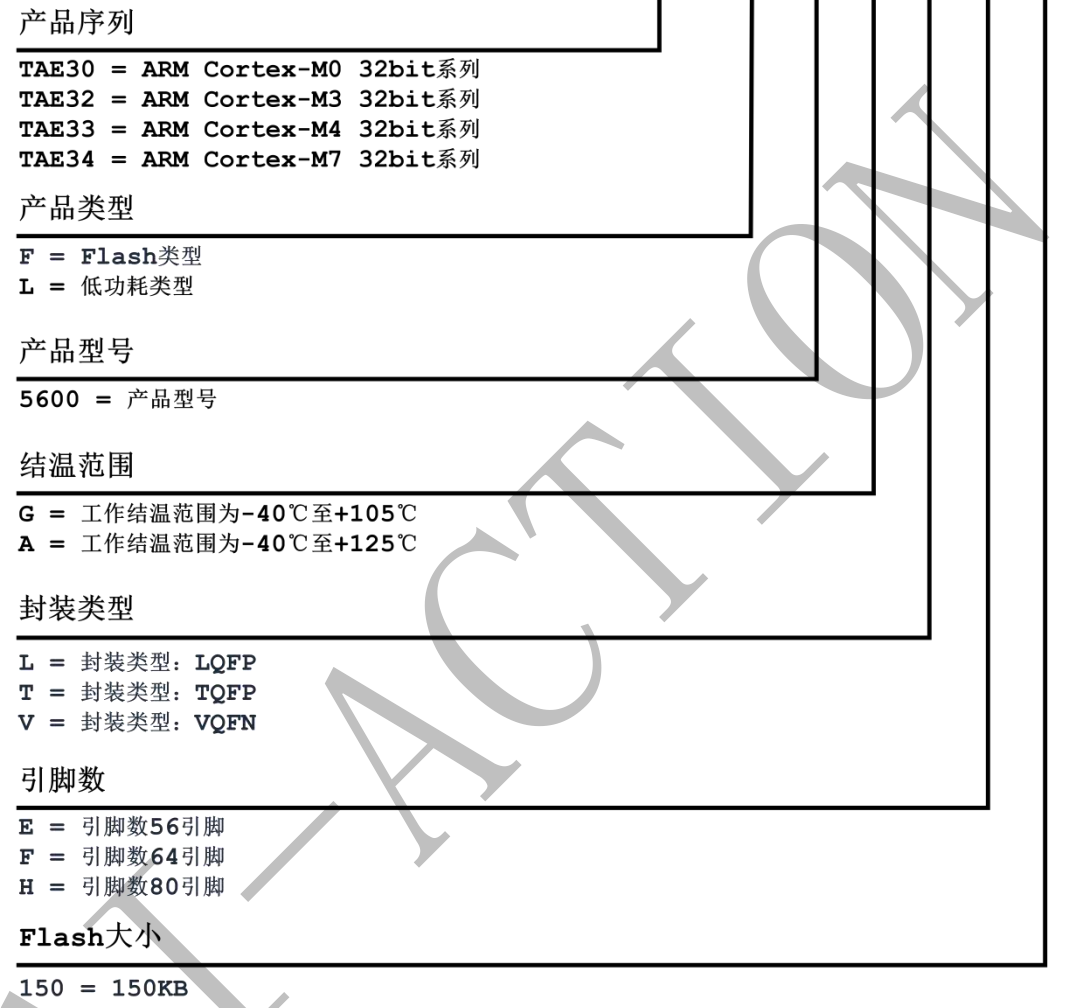
VQFN 56L：



NOTE：UNITS OF MEASURE = MILLIMETER

29 订购信息

Example: TAE32 F 5600 A L H 150



版本历史

日期	版本	版本记录
2021/12/20	v1.0	初始版本
2022/1/7	v1.1	修改图 8-1 时钟电路简图 修改 5.5.2.12FLASH 读保护寄存器（FLASH_RDPR）位域
2022/1/19	v2.0	统一所有寄存器描述格式 修改 15.5.2.4DMA 控制寄存器（DMA_CTR）位 31/30； 统一所有图片的格式
2022/1/27	V2.1	寄存器列表与寄存器详细描述添加超链接跳转
2022/2/9	V2.2	更新 I2C 章节 23.4.2.4~23.4.2.5 内容
2022/5/10	V2.3	6.5.2.4DFLASH 状态寄存器 位[0]英文名称修改