3061M 系列

技术参考指南

文档版本　02

发布日期　2024-05-27

前 言

概述

本文档主要描述使用 3061M 系列产品进行开发的具体相关信息：包含各个功能模块的内部结构、功能概述、各种工作模式的使用、寄存器配置等详细信息。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
3061M 系列	-

读者对象

本文档（本指南）主要适用于以下工程师：

- 硬件测试工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明

符号	说明
⚠ 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
⚠ 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
⚠ 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其他不可预知的结果。 “须知”不涉及人身伤害。
🕮 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

寄存器访问类型约定

本文中可能出现下列寄存器访问类型，它们所代表的含义如下。

类型	说明
RO	只读，不可写。
WO	只写。
RW	可读可写。
RC	读清零。
RWSC	可读可写，能够被逻辑事件清零。
RW1C	可读，写 1 清零，写 0 保持不变。
W1_PULSE	可读，写 1 后片内自清零，即产生一个脉冲。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2022-09-28	00B01	第 1 次临时版本发布。
2024-01-26	00B02	第 2 次临时版本发布。 勘误：删除 CRG 中时钟系统框图，删除 SPI，UART，I2C，TIMER 和 WWDG 的 2/4/8 分频，以及相关联外设章节的 2/4/8 分频。
2024-02-02	00B03	第 3 次临时版本发布。 补充 ADC 的输入通道 DAC_OUT 和 AVDD/3，刷新 ADC GPIO 触发源描述为 IO 触发。
2024-02-26	00B04	第 4 次临时版本发布。 修改 IWDG 章节：删除 RIS 和 MIS 寄存器，reserve int_en 寄存器。删除 IWDG 的中断。 删除 PMC wakeup1 相关功能描述。
2024-03-07	00B05	第 5 次临时版本发布。 修订 6.3.4 章节里“清除频率错误中断”有关的描述
2024-03-15	00B06	第 6 次临时版本发布。 修订 24.15 章节，ADC_SOC0_CFG 寄存器 cfg_soc0_ch_sel 通道选择描述。 新增 24.15 章节，ADC_AVDD_EN 寄存器。 补充 24.2.5 章节 ADC 采样 AVDD/3 通道的使能配置约束。 补充 24.5.4 章节 ADC 通道选择描述。
2024-03-19	00B07	第 7 次临时版本发布。 修订 5.3.4 章节，hclk 的衍生时钟增加之前多删除的 SPI, UART, I2C, TIMER 和 WWDG。

修订日期	版本	修订说明
		修订 5.5 章节，删掉 PERI_CRG128 中的多余部分，删掉 PERI144~PERI147 中的多余部分。
2024-04-01	01	第 1 次正式版本发布。 优化各章节描述。
2024-05-27	02	第 2 次正式版本发布。 优化各章节描述。

目 录

前 言 i

1 系统 1

1.1 系统架构 1

1.2 地址空间映射 2

1.2.1 概述 2

1.2.2 空间映射 3

1.3 嵌入式 SYSRAM 6

1.4 系统启动与固件升级 6

1.4.1 概述 6

1.4.2 启动方式 7

1.5 中断映射 7

2 处理器 10

2.1 特性 10

2.1.1 功能描述 10

2.2 编程模型 11

2.2.1 通用寄存器组 11

2.2.2 控制与状态寄存器 13

2.2.3 标准 CSR 寄存器概览 13

2.2.4 标准 CSR 寄存器描述 15

2.2.5 自定义 CSR 寄存器概览 32

2.2.6 自定义 CSR 寄存器描述 33

2.2.7 特权模式 48

2.2.8 休眠模式 49

2.2.9 物理内存保护.....50
2.3 指令介绍.....51
2.3.1 标准扩展指令.....51
2.3.2 自定义指令.....56
2.3.2.1 LDMIA.....56
2.3.2.2 STMIA.....58
2.3.2.3 L.LI.....59
2.3.2.4 C.LBU.....59
2.3.2.5 C.SB.....60
2.3.2.6 C.PUSH/C.POP/C.POPRET.....60
2.3.2.7 BEQI/BGEI/BGEUI/BLTI/BLTUI/BNEI.....67
2.3.2.8 ADDSHF/ SUBSHF/ORSHF/XORSHF/ANDSHF.....68
2.3.2.9 C.UXTB/C.UXTH.....69
2.3.2.10 C.SH/C.LHU.....70
2.3.2.11 MULIADD.....71
2.3.2.12 J16M/JAL16M.....72
2.3.3 指令执行周期.....73
2.4 异常与中断.....73
2.4.1 异常.....73
2.4.1.1 异常类型.....73
2.4.1.2 异常响应流程.....75
2.4.2 中断.....76
2.4.2.1 中断类型.....76
2.4.2.2 中断响应流程.....77
2.4.2.3 中断编程模型.....77
2.4.2.4 硬件中断嵌套.....82
2.4.3 硬件错误与死锁.....83
2.5 MTimer.....84
2.5.1 概述.....84
2.5.2 特性.....84
2.5.3 功能描述.....84
2.5.4 工作方式.....84
2.5.4.1 初始化.....84
2.5.4.2 64bit 计数值读取.....85
2.5.4.3 中断处理和 64bit 比较值更新.....85

2.5.5 MTimer 寄存器概览 86
2.5.6 MTimer 寄存器描述 87
2.6 调试功能 89
3 嵌入式闪存(Flash) 90
3.1 概述 90
3.1.1 Flash 存储器介绍 90
3.1.2 Flash 控制器介绍 91
3.2 特性 91
3.3 功能描述 92
3.3.1 Flash 存储空间管理 92
3.3.2 Flash 控制器初始化 93
3.3.2.1 Flash 时钟控制 93
3.3.2.2 Flash 错误码纠错控制 94
3.3.2.3 Flash Main region 空片标志查询 94
3.3.2.4 Flash 低功耗管理 94
3.3.2.5 Flash 直接读取访问加速 95
3.3.2.5.1 Cache 机制 95
3.3.2.5.2 预取机制 96
3.3.3 Flash 擦除操作 96
3.3.4 Flash 编程操作 98
3.3.5 Flash 读操作 100
3.4 info_rgn0 描述 100
3.5 info_rgn1 描述 101
3.6 Flash 分区保护机制 103
3.7 Flash 保护 104
3.8 Flash 中断 106
3.9 Flash 寄存器 108
3.9.1 Flash 控制器普通寄存器概览 108
3.9.2 Flash 控制器普通寄存器描述 110
3.9.3 Flash 控制器特殊寄存器概览 128
3.9.4 Flash 控制器特殊寄存器描述 129

3.9.5 FLASH_RGN0_SHADOWS 寄存器概览 135
3.9.6 FLASH_RGN0_SHADOWS 寄存器描述 136
3.9.6.1 DIE_ID_0 136
3.9.6.2 DIE_ID_1 136
3.9.6.3 DIE_ID_2 137
3.9.6.4 DIE_ID_ 3 137
3.9.6.5 CHIP_ID 137
3.9.7 FLASH_RGN1_SHADOWS 寄存器概览 137
3.9.8 FLASH_RGN1_SHADOWS 寄存器描述 138
3.9.8.1 FUNC_JTAG_ENABLE 140
3.9.8.2 ISP_BOOT_ENABLE 140
3.9.8.3 MAIN_RGN0_SIZE 141
3.9.9 Protection level 回退操作寄存器概览 142
3.9.10 Protection Level 回退操作寄存器描述 143
4 电源控制(PMC) 145
4.1 概述 145
4.2 特性 145
4.3 电源 145
4.3.1 概述 145
4.3.2 线性调压器 147
4.3.3 电源监控 147
4.3.3.1 上电复位（POR）/掉电复位（PDR） 147
4.3.3.2 可编程电压检测器(PVD) 147
4.4 低功耗模式 147
4.4.1 概述 147
4.4.2 工作模式 148
4.4.3 睡眠模式 148
4.4.4 深睡眠模式 149
4.5 PMC 寄存器概览 150
4.6 PMC 寄存器描述 151
5 时钟复位生成(CRG) 156
5.1 概述 156
5.2 复位 156

5.2.1 复位源描述 156
5.2.2 复位域描述 157
5.3 时钟 158
5.3.1 源时钟 159
5.3.2 PLL 159
5.3.3 衍生时钟 160
5.3.4 输出时钟 161
5.3.5 寄存器写保护 162
5.4 CRG_PERIAPBIF 寄存器概览 162
5.5 CRG_PERIAPBIF 寄存器描述 166
6 时钟频率监控(CMM) 197
6.1 概述 197
6.2 特性 197
6.3 功能描述 198
6.3.1 参考时钟和目标时钟 198
6.3.1.1 参考时钟选择和分频 198
6.3.1.2 目标时钟选择和分频 199
6.3.2 时钟频率检测和时钟失效检测 199
6.3.3 中断描述 199
6.4 典型配置 201
6.4.1 初始化的流程 201
6.4.2 更新配置的流程 202
6.4.3 检测到时钟失效的恢复流程 202
6.4.4 典型示例和注意事项 203
6.5 CMM 寄存器概览 204
6.6 CMM 寄存器描述 205
7 管脚复用控制(IOCMG) 213
7.1 概述 213
7.2 特性 213
7.3 I/O 模拟模式 213
7.4 I/O 晶振模式 214

7.5 IOCMG 寄存器 214
7.5.1 IOCMG_AON 寄存器概览 214
7.5.2 IOCMG_AON 寄存器描述 214
7.5.3 IOCMG_CORE 寄存器概览 218
7.5.4 IOCMG_CORE 寄存器描述 221
8 通用 I/O(GPIO) 262
8.1 概述 262
8.2 特性 262
8.3 功能描述 263
8.4 典型配置 264
8.5 GPIO 控制器寄存器概览 265
8.6 GPIO 控制器寄存器描述 266
9 系统配置控制器(SYSCTRL) 271
9.1 概述 271
9.2 功能描述 271
9.2.1 系统关键寄存器的写保护控制 271
9.2.2 系统复位事件次数统计 272
9.2.3 系统软件复位 272
9.2.4 系统软件中断 272
9.2.5 系统状态查询 272
9.2.6 可读可写的通用寄存器 272
9.2.7 APT 同步启动 273
9.2.8 APT 管脚事件数字滤波 273
9.2.9 PVD 阈值和使能控制 273
9.2.10 XTAL 管脚时钟模式控制 273
9.2.11 SYSRAM 分区配置 274
9.3 SYSCTRL0 寄存器概览 274
9.4 SYSCTRL0 寄存器描述 275
9.5 SYSCTRL1 寄存器概览 280
9.6 SYSCTRL1 寄存器描述 281

10 直接存储器访问控制器(DMA).....289
10.1 概述.....289
10.2 特性.....289
10.3 功能描述.....290
10.3.1 功能框图.....290
10.3.2 DMA 通道的传输.....291
10.3.2.1 DMA 传输类型和流控.....291
10.3.2.2 外设请求线（SREQ、BREQ 和外设流控请求线）.....293
10.3.2.3 Single 传输和 Burst 传输.....296
10.3.2.4 DMA 通道的传输位宽.....297
10.3.2.5 DMA 通道的传输个数.....297
10.3.3 DMA 通道的优先级.....298
10.3.4 DMA 链表.....298
10.3.5 DMA 工作流程.....300
10.4 典型配置.....302
10.4.1 时钟门控.....302
10.4.2 初始化.....302
10.4.3 启动通道.....303
10.4.4 中断处理.....303
10.4.5 DMA 链表实现连续传输.....304
10.4.6 DMA_Cn_CONTROL 使用说明.....304
10.4.7 DMA_Cn_CONFIG 使用说明.....306
10.5 DMA 寄存器概览.....307
10.6 DMA 寄存器描述.....308
11 循环冗余校验计算单元(CRC).....324
11.1 概述.....324
11.2 特性.....324
11.3 功能描述.....325
11.4 典型配置.....326
11.4.1 CRC 校验码产生.....326
11.5 CRC_CFG 寄存器概览.....327

11.6 CRC_CFG 寄存器描述 328

12 捕获模块(CAPM) 332

12.1 概述 332
12.2 特性 332
12.3 功能描述 333
12.3.1 输入滤波和预分频 334
12.3.2 边沿个数的统计和比较 335
12.3.3 TSR 计数器 335
12.3.4 TSR 计数器同步功能 336
12.3.5 捕获序列和捕获模式 337
12.3.6 复位工作状态 337
12.3.7 触发 DMA 请求 337
12.3.8 仿真器调试 338
12.4 典型配置与部分场景使用建议 338
12.4.1 缺省配置 338
12.4.2 通用配置流程（以 CAPM0 捕获通道为例） 338
12.4.3 去初始化流程（以 CAPM0 捕获通道为例） 340
12.4.4 TSR 计数器同步功能 341
12.4.4.1 硬件同步 CAPM（以 CAPM0 为例） 341
12.4.4.2 软件同步 CAPM（以 CAPM0 为例） 341
12.4.5 霍尔传感 341
12.4.5.1 配置滤波 342
12.4.5.2 平均转速和变化趋势 343
12.4.5.3 转子位置区域 343
12.4.6 PWM 通信场景 345
12.4.6.1 通过调频传递信息 345
12.4.6.2 通过调占空比传递信息（约定 PWM 波频率/周期不变） 346
12.4.7 边沿个数的统计和比较 348
12.5 CAPMx_CCR 寄存器概览（x=0,1,2） 349
12.6 CAPMx_CCR 寄存器描述（x=0,1,2） 350
12.7 CAPM_COMM 寄存器概览 362

12.8 CAPM_COMM 寄存器描述.....363

13 正交解码模块(QDM).....369

13.1 概述.....369

13.2 特性.....369

13.3 功能描述.....370

13.3.1 滤波，极性选择和互换.....371

13.3.2 解码模式.....371

13.3.3 倍频解码.....373

13.3.4 PPU 位置计数/复位/初始化/锁存模式.....376

13.3.4.1 3 种位置计数模式.....376

13.3.4.2 4 种位置计数复位模式.....377

13.3.4.3 软件初始化位置计数器.....377

13.3.4.4 3 种 Z 相初始化位置计数器模式.....377

13.3.4.5 位置计数锁存.....378

13.3.5 PPU 位置比较和同步输出.....378

13.3.6 TSU 计时和时间戳.....379

13.3.7 PTU 周期模式/计时/周期触发.....379

13.3.8 PTU 看门狗模式.....380

13.4 典型配置和部分场景的使用建议.....381

13.4.1 编码器的解码/位置计数的校准和初始化.....381

13.4.1.1 PPU 位置计数和硬件全自动校准的典型配置流程.....381

13.4.1.2 PPU 位置计数和硬件单次校准的典型配置流程.....381

13.4.1.3 PPU 位置计数和软件校准的典型配置流程.....382

13.4.1.4 PTU 看门狗模式的典型配置流程.....382

13.4.2 速度计算.....383

13.4.2.1 M 法计算速度.....383

13.4.2.2 MT 法计算速度.....384

13.4.3 QDM 用作独立 TIMER.....386

13.5 QDM 寄存器概览.....386

13.6 QDM 寄存器描述.....388

14 高级 PWM 定时器(APT).....407

14.1 概述 407
14.1.1 APT 模块概述 407
14.1.2 关键指标参数 412
14.1.2.1 PWM 频率 412
14.1.2.2 PWM 周期 412
14.1.2.3 PWM 分辨率 413
14.1.2.4 死区的宽度 413
14.1.2.5 死区分辨率 413
14.1.2.6 占空比调节范围 413
14.1.3 子模块概述 413
14.2 缓存加载功能介绍 416
14.2.1 功能概述 416
14.2.2 全局加载信号 glb_ld 生成 417
14.2.3 支持缓存加载的寄存器列表 418
14.3 计数控制子模块（Timer-Control） 421
14.3.1 计数控制子模块的功能 422
14.3.2 计数控制子模块的关键信号 423
14.3.3 计算 PWM 的频率 426
14.3.4 计数参考值和屏蔽窗口参考值的配置 427
14.3.5 屏蔽窗口产生逻辑 430
14.3.6 同步功能 434
14.3.6.1 同步输入 435
14.3.6.2 同步输出 437
14.4 波形生成子模块（PWM-Generation） 438
14.4.1 波形生成子模块的功能 439
14.4.2 波形生成子模块的关键信号 439
14.4.3 波形生成子模块输入信号优先级 442
14.4.4 波形生成子模块控制波形动作 443
14.4.5 常用波形图 445
14.5 死区生成子模块（DeadBand-Generation） 454
14.5.1 死区生成子模块的功能 454

14.5.2 死区生成子模块的典型配置.....457
14.6 输出保护子模块（Output-Protect）.....458
14.6.1 输出保护子模块的功能.....459
14.6.2 输出保护子模块的典型操作.....460
14.7 事件管理子模块（Event-Management）.....463
14.7.1 事件管理子模块的功能.....464
14.7.2 极性选择.....464
14.7.3 TRIP_OR 信号的产生.....465
14.7.4 输入选择.....465
14.7.5 事件组合.....466
14.7.6 事件屏蔽.....466
14.7.7 输出选择.....467
14.8 中断生成子模块（Interrupt-Generation）.....467
14.8.1 中断生成子模块的功能.....468
14.9 触发信号生成子模块（Trigger signal Generation）.....469
14.9.1 触发信号生成子模块的功能.....470
14.10 APTx 寄存器概览.....473
14.11 APTx 寄存器描述.....477
15 通用 PWM 定时器(GPT).....554
15.1 概述.....554
15.2 特性.....554
15.2.1 特性列表.....554
15.2.2 关键指标参数.....554
15.3 功能描述.....555
15.3.1 PWM 周期.....555
15.3.2 PWM 占空比.....555
15.3.3 PWM 个数.....556
15.3.4 GPT 的使能.....556
15.3.5 中断功能.....556
15.3.6 DMA 请求.....557
15.3.7 触发 ADC 采样.....557

15.3.8 寄存器的缓存加载.....557
15.4 工作方式.....558
15.5 GPTx 寄存器概览.....558
15.6 GPTx 寄存器描述.....559
16 基本定时器(TIMER).....569
16.1 概述.....569
16.2 特性.....569
16.3 功能描述.....570
16.3.1 TIMER 计数时钟的分频.....570
16.3.2 TIMER 的计数模式.....570
16.3.3 产生 DMA 请求和触发 ADC 启动采样的信号.....571
16.3.4 中断.....571
16.4 工作方式.....572
16.5 TIMERx 寄存器概览.....572
16.6 TIMERx 寄存器描述.....573
17 窗口看门狗(WWDG).....579
17.1 概述.....579
17.2 特性.....579
17.3 功能描述.....579
17.4 典型配置.....581
17.4.1 计数初值配置.....581
17.4.2 模块初始化配置.....582
17.4.3 中断处理过程.....582
17.5 WWDG_CFG 寄存器概览.....582
17.6 WWDG_CFG 寄存器描述.....583
18 独立看门狗(IWDG).....587
18.1 概述.....587
18.2 特性.....587
18.3 功能描述.....587
18.4 典型配置.....589

18.4.1 计数初值配置.....589
18.4.2 模块初始化配置.....589
18.5 IWDG_CFG 寄存器概览.....590
18.6 IWDG_CFG 寄存器描述.....591
19 集成电路接口(I2C).....594
19.1 概述.....594
19.2 特性.....594
19.3 功能框图.....595
19.4 功能描述.....596
19.4.1 SCL 和 SDA 相关配置.....596
19.4.1.1 SCL 配置.....596
19.4.1.2 SDA 配置.....597
19.4.1.3 数字滤波.....598
19.4.1.4 总线空闲.....598
19.4.2 FIFO.....599
19.4.3 SCL 延长.....599
19.4.4 ACK 和 NACK.....601
19.4.5 数据传输位序.....601
19.4.6 Master 和 Slave.....601
19.4.7 Slave 自身地址和特殊地址.....602
19.4.8 I2C 总线时序合成.....603
19.4.8.1 I2C 7bit 寻址写操作标准时序.....605
19.4.8.1.1 Master-Transmitter.....605
19.4.8.1.2 Slave-Receiver.....606
19.4.8.2 I2C 7bit 寻址读操作标准时序.....607
19.4.8.2.1 Master-Receiver.....607
19.4.8.2.2 Slave-Transmitter.....608
19.4.8.3 I2C 7bit 寻址写读操作组合标准时序.....609
19.4.8.3.1 Master-Transmitter 和 Master-Receiver.....609
19.4.8.3.2 Slave-Receiver 和 Slave-Transmitter.....611
19.4.8.4 I2C 10bit 寻址写操作标准时序.....612
19.4.8.4.1 Master-Transmitter.....612
19.4.8.4.2 Slave-Receiver.....613

19.4.8.5 I2C 10bit 寻址读操作标准时序 614
19.4.8.5.1 Master-Receiver 614
19.4.8.5.2 Slave-Transmitter 615
19.4.8.6 I2C 非标准时序（例子 1） 616
19.4.8.6.1 Master-Transmitter 616
19.4.8.6.2 Slave-Receiver 617
19.4.8.6.3 Master-Receiver 618
19.4.8.6.4 Slave-Transmitter 618
19.4.9 Bus Clear 和 GPIO 模式 619
19.4.10 DMA 操作 620
19.4.11 I2C 中断 620
19.5 工作方式 623
19.5.1 初始化配置 623
19.5.1.1 I2C 控制器的时钟和复位 623
19.5.1.2 I2C 总线配置 624
19.5.1.2.1 I2C 总线 SCL 速率配置 624
19.5.1.2.2 I2C 总线 SDA 建立保持时间配置 624
19.5.1.2.3 I2C 总线滤波/时钟超时和空闲配置 624
19.5.1.2.4 I2C 总线数据传输位序配置 625
19.5.1.3 Master 和 Slave 的配置 625
19.5.1.3.1 SCL Stretching 配置 625
19.5.1.3.2 接收 ACK 和 NACK 模式配置 625
19.5.1.3.3 Slave 自身地址和特殊地址配置 625
19.5.1.3.4 FIFO 配置 625
19.5.1.3.5 I2C 控制器 Master 和 Slave 使能 625
19.5.2 中断方式下的写操作数据传输 625
19.5.2.1 Master-Transmitter 625
19.5.2.2 Slave-Receiver 627
19.5.3 中断方式下的读操作数据传输 628
19.5.3.1 Master-Receiver 628
19.5.3.2 Slave-Transmitter 629
19.5.4 查询方式下的写操作数据传输 631
19.5.4.1 Master-Transmitter 631
19.5.4.2 Slave-Receiver 632

19.5.5 查询方式下的读操作数据传输 633
19.5.5.1 Master-Receiver 633
19.5.5.2 Slave-Transmitter 634
19.5.6 DMA 方式下的写操作数据传输 636
19.5.6.1 Master-Transmitter 636
19.5.6.2 Slave-Receiver 637
19.5.7 DMA 方式下的读操作数据传输 638
19.5.7.1 Master-Receiver 638
19.5.7.2 Slave-Transmitter 639
19.5.8 异常处理流程 639
19.6 I2C 寄存器概览 640
19.7 I2C 寄存器描述 642
20 通用异步收发传输器(UART) 669
20.1 概述 669
20.2 特性 669
20.3 功能描述 670
20.3.1 FIFO 671
20.3.2 波特率生成 671
20.3.3 中断功能 672
20.3.4 发送完成状态 673
20.3.5 数据位序 674
20.3.6 过采样倍数 674
20.3.7 接收超时 674
20.3.8 自动波特率检测 674
20.3.9 字符检测 675
20.3.10 DMA 功能 675
20.3.11 应用框图 675
20.3.12 协议介绍 676
20.4 典型配置 677
20.4.1 中断或查询方式下的数据传输 677
20.4.2 DMA 方式下的数据传输 678
20.4.3 自动波特率检测 679
20.5 UART 寄存器概览 681

20.6 UART 寄存器描述.....683

21 同步串行外设接口(SPI) 705

21.1 概述 705
21.2 特性 705
21.3 功能描述 706
21.4 接口时序 708
21.4.1 Motorola SPI 接口 708
21.4.1.1 接口时序概览.....709
21.4.1.2 SPO=0、SPH=0.....709
21.4.1.3 SPO=0、SPH=1.....710
21.4.1.4 SPO=1、SPH=0.....711
21.4.1.5 SPO=1、SPH=1.....712
21.4.2 TI 同步串行接口 713
21.4.3 Microwire 接口.....714
21.5 工作方式 716
21.5.1 工作模式.....716
21.5.2 时钟与复位.....717
21.5.3 中断处理.....717
21.5.4 初始化.....718
21.5.5 查询方式下的数据传输.....718
21.5.6 中断方式下的数据传输.....719
21.5.7 DMA 方式下的数据传输.....720
21.6 SPI 寄存器概览 720
21.7 SPI 寄存器描述 721

22 CAN 控制器 731

22.1 概述 731
22.2 特性 731
22.3 功能描述 732
22.3.1 编码方式.....732
22.3.2 波特率.....732
22.3.3 总线速率与传输距离的关系.....733

22.3.4 帧类型 734
22.3.5 典型应用 734
22.3.6 测试模式 735
22.3.7 Basic 模式 736
22.3.8 报文对象和 IF 报文对象接口寄存器组 737
22.3.9 报文接收过滤功能 739
22.4 典型配置 740
22.4.1 初始化所有报文对象 740
22.4.2 传输速率配置 741
22.4.3 配置 Tx 报文对象 742
22.4.4 配置 Rx 报文对象 743
22.4.5 中断查询与处理 745
22.4.6 报文接收过滤 746
22.4.7 连续报文接收 747
22.4.8 错误处理 747
22.5 CAN 寄存器概览 748
22.6 CAN 寄存器描述 750
23 ADC 电压参考(ADCVREF) 780
23.1 概述 780
23.2 特性 780
23.3 ADCVREF 使能 780
23.4 工作模式 781
23.5 VREF_REG 寄存器概览 781
23.6 VREF_REG 寄存器描述 782
24 模数转换器（ADC） 784
24.1 概述 784
24.2 特性 784
24.2.1 框架 786
24.2.2 时钟 786
24.2.3 分辨率 787
24.2.4 参考电压 788

24.2.5 输入信号.....788
24.2.6 ADC 时序.....789
24.2.7 采样结果.....792
24.3 启动.....793
24.4 转换模式.....793
24.4.1 单次转换模式.....793
24.4.2 连续转换模式.....794
24.5 转换开始模块.....795
24.5.1 SOC 框架.....795
24.5.2 触发源配置.....796
24.5.3 SOC 触发溢出标志.....796
24.5.4 通道选择.....797
24.5.5 转换完成标志.....797
24.5.6 转换结果.....797
24.6 转换优先级.....798
24.6.1 轮询优先级组.....798
24.6.2 高优先级组.....800
24.7 数据处理.....802
24.8 过采样.....802
24.9 后处理模块.....804
24.9.1 PPB 框架.....804
24.9.2 采样延迟记录.....805
24.9.3 数据偏移.....805
24.9.4 误差计算.....805
24.9.5 阈值检测.....806
24.9.6 过零点检测.....806
24.10 中断.....806
24.10.1 数据中断.....806
24.10.2 事件中断.....807
24.10.3 异常中断.....808
24.11 DMA 请求.....809

24.12 温度传感器 810
24.13 典型场景配置 811
24.13.1 APT 触发单次转换 811
24.13.2 软件触发连续转换 811
24.13.3 APT 触发过采样 812
24.13.4 TIMER 触发多个 SOC 采样 813
24.14 ADC_SYS_REG 寄存器概览 814
24.15 ADC_SYS_REG 寄存器描述 822
24.16 TSENSOR_REG 寄存器概览 872
24.17 TSENSOR_REG 寄存器描述 873
25 可编程增益放大器(PGA) 874
25.1 概述 874
25.2 特性 874
25.3 PGA 使能 875
25.4 工作模式 875
25.4.1 内置电阻模式 876
25.4.2 外置电阻模式 877
25.5 PGA_REG 寄存器概览 878
25.6 PGA_REG 寄存器描述 878
26 数模转换器(DAC) 881
26.1 概述 881
26.2 特性 881
26.3 DAC 使能 882
26.4 工作方式 882
26.5 DAC_REG 寄存器概览 882
26.6 DAC_REG 寄存器描述 883
27 模拟比较器(ACMP) 884
27.1 概述 884
27.2 特性 884
27.3 比较器使能 885

27.4 迟滞电压 885
27.5 端口选择 886
27.6 比较极性选择....887
27.7 滤波功能 887
27.8 屏蔽功能 888
27.9 结果输出 889
27.10 中断....889
27.11 ACMP_REG 寄存器概览....890
27.12 ACMP_REG 寄存器描述....891
28 缩略语 898

插图目录

图 1-1 系统架构图 2
图 2-1 PUSH 指令操作 1 到 5 个寄存器的示例 64
图 2-2 POP 指令操作 1 到 5 个寄存器的示例 65
图 2-3 调试通路架构示意图 89
图 3-1 Flash 内部存储结构示意图 91
图 3-2 Flash 中断原理示意图 106
图 4-1 MCU 电源域示意图 146
图 4-2 低功耗模式状态图 148
图 5-1 时钟系统框图 158
图 6-1 CMM 功能模块图 198
图 8-1 模块框图 263
图 8-2 比特屏蔽对应关系 267
图 9-1 PVD toggle 触发示意图 273
图 10-1 DMA 功能框图 290
图 10-2 DMA 链表单个节点数据结构和多个节点链接示例 299
图 10-3 LLI 更新通道寄存器示意图 299
图 10-4 DMA 链表实现连续传输 304
图 11-1 模块框图 325
图 11-2 输入输出数据反序示例图 325
图 12-1 三通道 CAPM 功能模块框图 333

图 12-2 单通道功能模块详细框图（以 CAPM0 捕获通道为例）.....334
图 12-3 输入预分频图例（以 CAPM0 捕获通道 0 为例）.....335
图 12-4 CAPM 波形示例图（霍尔传感器场景）.....342
图 12-5 CAPM 波形示例图（PWM 调频通信）.....346
图 12-6 CAPM 波形示例图（PWM 调占空比通信）.....348
图 13-1 QDM 特性模块简要框图.....369
图 13-2 QDM 功能模块详细框图.....371
图 13-3 正交型编码器的 A 相、B 相、Z 相波形图示例（顺时针旋转）.....372
图 13-4 正交型编码器的 A 相、B 相、Z 相波形图示例（逆时针旋转）.....372
图 13-5 脉冲方向型编码器 A 相、B 相、Z 相波形图示例（先顺时针旋转，后逆时针旋转）.....372
图 13-6 非标准 TYPE1 型编码器波形图示例（先顺时针旋转，后逆时针旋转）.....372
图 13-7 非标准 TYPE2 型编码器波形图示例（先顺时针旋转，后逆时针旋转）.....373
图 13-8 正交输入型 1X/2X/4X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）.....374
图 13-9 脉冲方向型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）.....375
图 13-10 非标 TYPE1 型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）.....375
图 13-11 非标 TYPE2 型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）.....376
图 13-12 TSU 计时和时间戳示例.....379
图 13-13 PTU 计数和周期触发示例.....380
图 14-1 多个 APT 模块在系统中的连接.....408
图 14-2 单个 APT 模块的内部子模块和对外连接信号.....409
图 14-3 APT 模块关键内部信号连接.....411
图 14-4 支持缓存加载的寄存器的加载机制.....416
图 14-5 全局加载信号 glb_ld 的产生机制.....417
图 14-6 计数控制子模块.....422

图 14-7 计数控制子模块信号和寄存器.....424
图 14-8 增计数模式下 PWM 频率.....426
图 14-9 减计数模式下 PWM 频率.....427
图 14-10 先增后减计数模式下 PWM 频率.....427
图 14-11 增计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图.....428
图 14-12 减计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图.....429
图 14-13 先增后减计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图.....429
图 14-14 屏蔽窗口产生逻辑功能框图.....430
图 14-15 增计数模式下 APTxMSKWD 的产生波形.....432
图 14-16 减计数模式下 APTxMSKWD 的产生波形.....433
图 14-17 先增后减计数模式下 APTxMSKWD 的产生波形.....434
图 14-18 同步输入方案.....435
图 14-19 增计数模式同步波形.....436
图 14-20 减计数模式同步波形.....436
图 14-21 先增后减计数模式同步波形，TC_PHS[rg_cnt_dir]=0.....436
图 14-22 先增后减计数模式同步波形，TC_PHS[rg_cnt_dir]=1.....437
图 14-23 同步输出方案.....437
图 14-24 波形生成子模块.....438
图 14-25 波形生成子模块的输入输出.....440
图 14-26 输出 PG_PWMA 或 PG_PWMB 上可能的波形动作.....444
图 14-27 先增后减计数模式对称 PWM 波形.....447
图 14-28 增计数模式，单沿非对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，高电平有效.....448
图 14-29 增计数模式，单沿非对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，低电平有效.....449
图 14-30 增计数模式，PG_PWMA 输出非对称脉冲，脉冲占空比可独立控制.....450

图 14-31 先增后减计数模式，双沿对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，低电平有效.....451
图 14-32 先增后减计数模式，双沿对称 PWM 互补波形，PG_PWMA 和 PG_PWMB 独立控制占空比.....452
图 14-33 先增后减计数模式，双沿非对称 PWM 波形，PG_PWMA 独立控制占空比，低电平有效..453
图 14-34 利用 AEVTC1 和 BEVTC2 生成 PWM 波形.....453
图 14-35 死区生成子模块.....454
图 14-36 死区生成子模块的配置项.....455
图 14-37 插入上升沿延时前后的波形图.....455
图 14-38 插入下降沿延时前后的波形图.....456
图 14-39 典型死区配置的波形图.....458
图 14-40 输出保护子模块.....459
图 14-41 不同保护模式下，APTx_PWMA 和 APTx_PWMB 的保护动作.....462
图 14-42 事件管理子模块.....463
图 14-43 事件管理子模块功能框图.....464
图 14-44 TRIP1~TRIP12 信号的产生.....465
图 14-45 事件屏蔽功能.....466
图 14-46 中断生成子模块.....468
图 14-47 定时中断产生机制.....468
图 14-48 触发信号生成子模块.....470
图 14-49 APTxSOCA 信号产生机制.....470
图 14-50 APTxSOCB 信号产生机制.....472
图 17-1 模块框图.....580
图 18-1 模块框图.....588
图 19-1 I2C 控制器的功能框图.....595
图 19-2 I2C 协议约束的 I2C 总线器件 SCL 和 SDA 线的时序定义.....597
图 19-3 I2C 7bit 寻址 Master-Transmitter 标准时序图.....606

图 19-4 I2C 7bit 寻址 Slave-Receiver 标准时序图 607
图 19-5 I2C 7bit 寻址 Master-Receiver 标准时序图 608
图 19-6 I2C 7bit 寻址 Slave-Transmitter 标准时序图.....609
图 19-7 I2C 7bit 寻址 Master-Transmitter 和 Master-Receiver 标准时序图.....610
图 19-8 I2C 7bit 寻址 Slave-Receiver 和 Slave-Transmitter 标准时序图.....611
图 19-9 I2C 10bit 寻址 Master-Transmitter 标准时序图.....612
图 19-10 I2C 10bit 寻址 Slave-Receiver 标准时序图 613
图 19-11 I2C 10bit 寻址 Master-Receiver 标准时序图.....614
图 19-12 I2C 10bit 寻址 Slave-Transmitter 标准时序图.....615
图 19-13 I2C Master-Transmitter 非标准时序图（例子 1） 616
图 19-14 I2C Slave-Receiver 非标准时序图（例子 1） 617
图 19-15 I2C Master-Receiver 非标准时序图（例子 1）.....618
图 19-16 I2C Slave- Transmitter 非标准时序图（例子 1） 619
图 20-1 模块框图.....670
图 20-2 UART2 线典型应用框图 676
图 20-3 UART4 线典型应用框图 676
图 21-1 SPI 控制器在 MCU 中的位置.....706
图 21-2 SPI 接 Slave 时的连接关系.....707
图 21-3 SPI 接口时序概览图.....709
图 21-4 SPI 单帧帧格式（SPO=0、SPH=0） 709
图 21-5 SPI 连续帧帧格式（SPO=0、SPH=0） 709
图 21-6 SPI 单帧帧格式（SPO=0、SPH=1） 710
图 21-7 SPI 连续帧帧格式（SPO=0、SPH=1） 711
图 21-8 SPI 单帧帧格式（SPO=1、SPH=0） 711
图 21-9 SPI 连续帧帧格式（SPO=1、SPH=0） 712
图 21-10 SPI 单帧帧格式（SPO=1、SPH=1） 712
图 21-11 SPI 连续帧帧格式（SPO=1、SPH=1） 713

图 21-12 TI 同步串行单帧帧格式.....713
图 21-13 TI 同步串行连续帧帧格式.....714
图 21-14 Microwire 单帧帧格式（不插入等待周期的 Master 模式，或 Slave 模式）.....714
图 21-15 Microwire 单帧帧格式（插入等待周期的 Master 模式）.....715
图 21-16 Microwire 连续帧帧格式（不插入等待周期的 Master 模式，或 Slave 模式）.....715
图 21-17 Microwire 连续帧帧格式（插入等待周期的 Master 模式）.....715
图 21-18 图 SPI 中断处理.....718
图 22-1 位填充示意图.....732
图 22-2 1 个 CAN bit 示意图.....733
图 22-3 CAN 总线控制器对接外置收发器时的应用.....734
图 22-4 CAN 控制器在 Silent 模式下的示意图.....735
图 22-5 CAN 控制器在 Loop-back 模式下的示意图.....736
图 22-6 CAN 控制器在 Silent 模式和 Loop-back 模式同时开启时的示意图.....736
图 22-7 报文对象数据结构和 IF 报文对象接口寄存器组示意图.....738
图 22-8 报文接收过滤功能图示.....739
图 23-1 ADC_VREF 工作时序图.....781
图 24-1 ADC 模块框架.....786
图 24-2 ADC 时钟框架.....787
图 24-3 ADC 工作时序.....789
图 24-4 ADC 输入模型.....791
图 24-5 单次转换过程.....794
图 24-6 两次触发转换过程.....794
图 24-7 连续转换过程.....795
图 24-8 SOC 框架.....795
图 24-9 轮询优先级组转换.....799
图 24-10 高优先级组转换.....801
图 24-11 数据处理.....802

图 24-12 后处理模块框架.....804
图 24-13 采样延迟时间.....805
图 24-14 数据中断结构.....807
图 24-15 事件中断结构.....808
图 24-16 异常中断结构.....809
图 25-1 PGA 结构框图.....875
图 25-2 内置电阻模式结构图.....876
图 25-3 外置电阻模式结构图.....877
图 26-1 DAC 结构框图.....881
图 26-2 DAC 时序.....882
图 27-1 比较器结构框图.....884
图 27-2 ACMP 和 DAC 配合时序.....885
图 27-3 比较器滤波功能.....887
图 27-4 比较器屏蔽功能.....888
图 27-5 比较器屏蔽逻辑.....889
图 27-6 比较器中断.....890

表格目录

表 1-1 地址空间映射表..... 3
表 1-2 中断映射表..... 7
表 2-1 整型通用寄存器.....11
表 2-2 浮点通用寄存器.....12
表 2-3 标准 CSR 寄存器概览 13
表 2-4 自定义 CSR 寄存器概览.....32
表 2-5 Mret/wfi/ecall 指令在不同特权模式下的行为差异 48
表 2-6 RISC-V 32bit 指令的 opcode 分类映射表.....51
表 2-7 RISC-V 16bit 指令的 opcode 分类映射表.....51
表 2-8 支持的 RV32I 基本指令集指令.....52
表 2-9 支持的 RV32M 扩展指令集指令.....54
表 2-10 支持的 RV32C 扩展指令集指令 54
表 2-11 支持的 RV32F 单精度浮点扩展指令集指令.....55
表 2-12 支持的 RV32B 扩展指令集指令（仅 CLZ） 56
表 2-13 LDMIA 指令编码 56
表 2-14 LDMIA 和 STMIA 指令的 gpr_mask 与 callee/caller 寄存器编号对应表.....57
表 2-15 STMIA 指令编码 58
表 2-16 L.LI 指令编码 59
表 2-17 C.LBU 指令编码.....59
表 2-18 C.LBU 和 C.SB 指令 rs1′ rs2′ 与 rd′ 位域指定的寄存器编号.....60

表 2-19 自定义 C.SB 指令编码.....60
表 2-20 C.PUSH/C.POP/C.POPRET 16-bit 指令编码.....60
表 2-21 C.PUSH/C.POP/C.POPRET 指令的 opc 位域说明.....61
表 2-22 C.PUSH/C.POP/C.POPRET 的 rcount 位域说明.....61
表 2-23 立即数比较分支指令编码.....67
表 2-24 立即数比较分支指令中 opc 位域说明.....67
表 2-25 移位运算指令的编码.....68
表 2-26 移位运算指令的 opc 位域说明.....68
表 2-27 移位运算指令的 shtype 位域编码说明.....69
表 2-28 C.UXTB, C.UXTH 的编码.....69
表 2-29 C.UXTB, C.UXTH 的 opc 位域说明.....70
表 2-30 C.SH 和 C.LHU 的指令编码.....70
表 2-31 C.SH 和 C.LHU 的指令语义.....71
表 2-32 MULIADD 指令编码(X=uimm[0]).....71
表 2-33 MULIADD 指令的 opc 位域.....71
表 2-34 JAL16M/J16M 指令的编码.....72
表 2-35 JAL16M/J16M 指令的 opc 位域说明.....72
表 2-36 mcause 中不同异常的编码表.....74
表 2-37 Ccause 中不同异常的编码表.....75
表 2-38 mcause 中的中断编码表.....76
表 2-39 非标准本地中断相关的自定义 CSR.....78
表 2-40 各非标准本地中断的寄存器地址和位域.....79
表 2-41 硬件错误和死锁场景.....83
表 2-42 MTimer 寄存器概览（基址是 0x14380000）.....86
表 3-1 Flash 的存储空间按下表进行管理（基地址 0x0300_0000）.....92
表 3-2 Flash 读分频系数与系统时钟频率适配表.....93
表 3-3 Flash info_rgn0 描述列表（基地址 0x0080_0000）.....100

表 3-4 Flash info_rgn1 描述列表（基地址 0x0080_1000）.....101
表 3-5 Flash 控制器正常中断列表.....106
表 3-6 Flash 控制器异常中断列表.....107
表 3-7 Flash 控制器普通寄存器概览(基址是 0x1471_0000）.....108
表 3-8 Flash 控制器特殊寄存器概览(基址是 0x1471_0800).....128
表 3-9 FLASH_RGN0_SHADOWS 寄存器概览（基址是 0x1472_0000）.....135
表 3-10 FLASH_RGN1_SHADOWS 寄存器概览（基址是 0x1472_0000）.....137
表 3-11 Protection level 回退操作寄存器概览（基址是 0x1472_0000）.....142
表 4-1 PMC 寄存器概览（基址是 0x147E_0000）.....150
表 5-1 复位控制域.....157
表 5-2 CRG_PERIAPBIF 寄存器概览（基址是 0x1000_0000）.....162
表 6-1 CMM 中断.....200
表 6-2 CMM 寄存器概览（基址是 0x1001_0000）.....204
表 7-1 IOCMG_AON 寄存器概览（基址是 0x147F_0000）.....214
表 7-2 IOCMG_CORE 寄存器概览（基址是 0x147F_0000）.....219
表 8-1 MCU GPIO 控制器寄存器对应的基地址.....265
表 8-2 GPIO 控制器寄存器概览.....266
表 9-1 SYSCTRL0 寄存器概览（基址是 0x1010_0000）.....274
表 9-2 SYSCTRL1 寄存器概览（基址是 0x1010_0000）.....280
表 10-1 传输类型和流控模式.....292
表 10-2 DMA 的外设请求线分配.....294
表 10-3 dbsize 及 sbsize 的值与其对应的 burst 长度.....305
表 10-4 dwidth 和 swidth 的值与其对应传输位宽.....305
表 10-5 流控制器及传输类型位定义.....306
表 10-6 DMA 寄存器概览（基址是 0x1C00_0000）.....307
表 10-7 DMA 寄存器偏移地址变量表.....308
表 11-1 CRC_CFG 寄存器概览（基址是 0x00000）.....327

表 12-1 CAPM 模块地址空间分配 334
表 12-2 霍尔传感器的真值表示例 342
表 12-3 软件中断方式下，CAPM 的捕获序列示例（霍尔传感器场景） 344
表 12-4 软件轮询方式下，CAPM 的捕获序列示例（霍尔传感器场景） 345
表 12-5 CAPMx_CCR 寄存器概览（CAPM0~2 捕获通道的基地址分别是 0x14B0_0000，0x14B0_1000，0x14B0_2000） 349
表 12-6 CAPM_COMM 寄存器概览（基址是 0x14B0_3000） 362
表 13-1 QDM 寄存器概览 386
表 14-1 子模块配置参数 413
表 14-2 全局加载信号的触发源 418
表 14-3 支持缓存加载的寄存器 418
表 14-4 计数控制子模块关键信号 424
表 14-5 波形生成子模块可能的输入信号 440
表 14-6 先增后减计数模式下，对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级 442
表 14-7 增计数模式下对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级 442
表 14-8 减计数模式下对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级 443
表 14-9 典型的死区配置 457
表 14-10 所有可能的保护动作 461
表 14-11 APTx 寄存器概览（APTx 的基址是 0x14A0_x000） 473
表 15-1 GPTx 寄存器概览（GPTx 的基址是 0x1470_x000） 558
表 16-1 TIMERx 寄存器概览 573
表 17-1 WWDG_CFG 寄存器概览（基址是 0x1440_0000） 583
表 18-1 IWDG_CFG 寄存器概览（基址是 0x1440_1000） 590
表 19-1 协议约束的 I2C 总线器件 SCL 和 SDA 线的部分特性 597
表 19-2 I2C 协议 Slave 特殊地址（保留地址段） 603
表 19-3 时序命令（mst_slv_cmd 位域）的主机和从机译码结果 604
表 19-4 I2C 7bit 寻址 Master-Transmitter 标准时序配置序列 606

表 19-5 I2C 7bit 寻址 Slave-Receiver 标准时序配置序列.....607
表 19-6 I2C 7bit 寻址 Master-Receiver 标准时序配置序列.....608
表 19-7 I2C 7bit 寻址 Slave-Transmitter 标准时序配置序列.....609
表 19-8 I2C 7bit 寻址 Master-Transmitter 和 Master-Receiver 标准时序配置序列.....610
表 19-9 I2C 7bit 寻址 Slave-Receiver 和 Slave-Transmitter 标准时序配置序列.....611
表 19-10 I2C 10bit 寻址 Master-Transmitter 标准时序配置序列.....612
表 19-11 I2C 10bit 寻址 Slave-Receiver 标准时序配置序列.....613
表 19-12 I2C 10bit 寻址 Master-Receiver 标准时序配置序列.....614
表 19-13 I2C 10bit 寻址 Slave-Transmitter 标准时序配置序列.....615
表 19-14 I2C Master-Transmitter 非标准时序配置序列（例子 1）.....616
表 19-15 I2C Slave-Receiver 非标准时序配置序列（例子 1）.....617
表 19-16 I2C Master-Receiver 非标准时序配置序列（例子 1）.....618
表 19-17 I2C Slave-Receiver 非标准时序配置序列（例子 1）.....619
表 19-18 I2C 模块中断.....621
表 19-19 I2C 寄存器概览.....640
表 20-1 UART 寄存器概览.....681
表 21-1 SPI 寄存器概览.....721
表 22-1 CAN 波特率常用配置.....733
表 22-2 CAN 总线位速率与传输距离对应表.....734
表 22-3 CAN 寄存器概览.....748
表 23-1 VREF_REG 寄存器概览（基址是 0x1810_0000）.....782
表 24-1 ADC 输入信号连接关系.....788
表 24-2 ADC 工作时间描述.....789
表 24-3 ADC 典型采样率.....790
表 24-4 Rs 电阻取值.....790
表 24-5 采样时间选择.....791
表 24-6 电压转换为 12bit 结果.....792

表 24-7 12bit 结果转换为电压 793
表 24-8 触发源选择 796
表 24-9 模拟输入通道选择 797
表 24-10 过采样精度 803
表 24-11 温度传感器参数 810
表 24-12 ADC_SYS_REG 寄存器概览（基址是 0x1800_0000） 814
表 24-13 TSENSOR_REG 寄存器概览（基址是 0x1850_0000） 872
表 25-1 PGA 增益选择 876
表 25-2 外部电阻增益与前馈电容补偿配置关系表 877
表 25-3 PGA_REG 寄存器概览（基址是 PGA0：0x18200000；PGA1：0x18201000） 878
表 26-1 DAC_REG 寄存器概览（基址是 0x1840_0000） 882
表 27-1 ACMP 迟滞电压选择 885
表 27-2 ACMP P 端输入选择 886
表 27-3 ACMP N 端输入选择 886
表 27-4 输出极性选择 887
表 27-5 比较器结果选择 889
表 27-6 ACMP_REG 寄存器概览（基址是 0x1830_0000） 890

1 系统

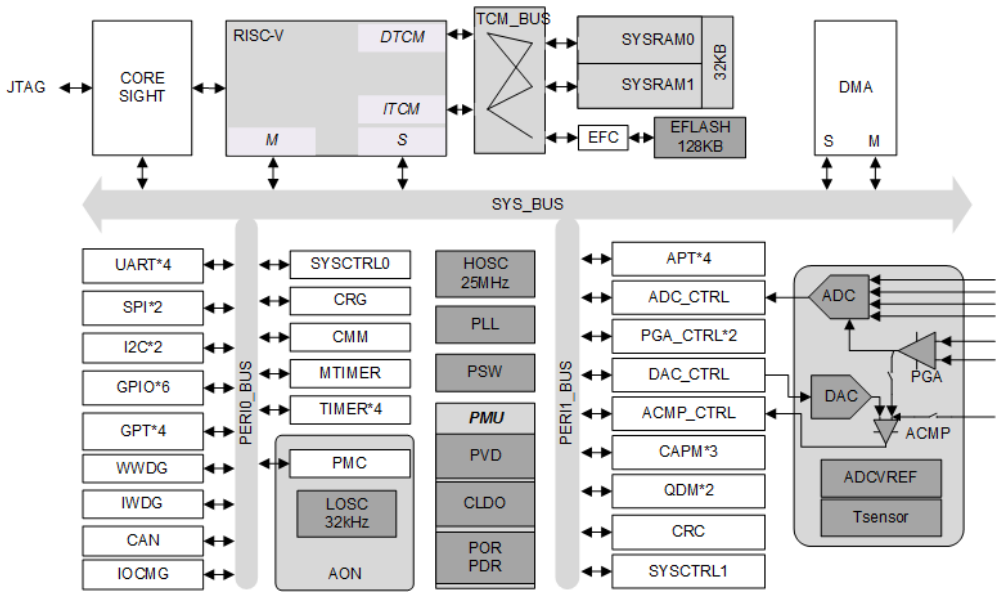
1.1 系统架构

主系统包含以下部件：

- masters：
  - RISC-V CPU
  - DMA
  - CORESIGHT
- slaves：
  - 嵌入式 SYSRAM
  - 嵌入式 Flash Memory
  - 系统外设

这些模块通过 AMBA 总线架构互连，如图 1-1 所示。

图1-1 系统架构图



TCM_BUS

此总线将 RISC-V CPU 的 TCM 接口与存储器互连。当 ITCM 和 DTCM 同时访问相同存储器时，DTCM 优先级高于 ITCM。

SYS_BUS

此系统总线实现 RISC-V CPU、DMA 和 CORESIGHT 的主接口访问系统外设与 DMA 主接口经 RISC-V CPU TCM 接口访问存储器的通路。

PERI0_BUS/ PERI1_BUS

此外设总线将各系统外设连接到 SYS_BUS，时钟频率与 SYS_BUS 时钟频率相同。

1.2 地址空间映射

1.2.1 概述

程序存储器、数据存储器、寄存器和 I/O 端口排列在同一个线性（即地址连续）的 4GB 地址空间内。

各字节按小端格式对齐（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）。

1.2.2 空间映射

所有未分配给片上存储器和外设的地址空间均视为“保留区”。有关可用存储器和寄存器区域的详细映射，如表 1-1 所示。

表1-1 地址空间映射表

起始地址	结束地址	功能	大小	说明
0x0000_0000	0x01FF_FFFF	Reserved	-	-
0x0200_0000	0x0200_7FFF	SYSRAM_ITCM	32KB	-
0x0200_8000	0x02FF_FFFF	Reserved	-	-
0x0300_0000	0x03FF_FFFF	EFLASH	16MB	main_rgn,128KB： 0x0300_0000~0x0301_FFFF info_rgn0,4KB： 0x0380_0000~0x0380_0FFF info_rgn1,2KB： 0x0380_1000~0x0380_17FF info_rgn2,10KB： 0x0380_1800~0x0380_3FFF
0x0400_0000	0x0400_7FFF	SYSRAM_DTCM	32KB	-
0x0400_8000	0x0FFF_FFFF	Reserved	-	-
0x1000_0000	0x1000_0FFF	CRG	4KB	-
0x1000_1000	0x1000_FFFF	Reserved	-	-
0x1001_0000	0x1001_0FFF	CMM	4KB	-
0x1001_1000	0x100F_FFFF	Reserved	-	-
0x1010_0000	0x1010_FFFF	SYSCTRL	64KB	-
0x1011_0000	0x13FF_FFFF	Reserved	-	-
0x1400_0000	0x1400_0FFF	UART0	4KB	-
0x1400_1000	0x1400_1FFF	UART1	4KB	-
0x1400_2000	0x1400_2FFF	UART2	4KB	-
0x1400_3000	0x1400_3FFF	UART3	4KB	-
0x1400_4000	0x140F_FFFF	Reserved	-	-
0x1410_0000	0x1410_0FFF	I2C0	4KB	-
0x1410_1000	0x1410_1FFF	I2C1	4KB	-

起始地址	结束地址	功能	大小	说明
0x1410_2000	0x141F_FFFF	Reserved	-	-
0x1420_0000	0x1420_0FFF	SPI0	4KB	-
0x1420_1000	0x1420_1FFF	SPI1	4KB	-
0x1420_2000	0x142F_FFFF	Reserved	-	-
0x1430_0000	0x1430_0FFF	TIMER0	4KB	-
0x1430_1000	0x1430_1FFF	TIMER1	4KB	-
0x1430_2000	0x1430_2FFF	TIMER2	4KB	-
0x1430_3000	0x1430_3FFF	TIMER3	4KB	-
0x1430_4000	0x1437_FFFF	Reserved	-	-
0x1438_0000	0x1438_0FFF	MTIMER	4KB	-
0x1438_1000	0x143F_FFFF	Reserved	-	-
0x1440_0000	0x1440_0FFF	WWDG	4KB	-
0x1440_1000	0x1440_1FFF	IWDG	4KB	-
0x1440_2000	0x144F_FFFF	Reserved	-	-
0x1450_0000	0x1450_0FFF	GPIO0	4KB	-
0x1450_1000	0x1450_1FFF	GPIO1	4KB	-
0x1450_2000	0x1450_2FFF	GPIO2	4KB	-
0x1450_3000	0x1450_3FFF	GPIO3	4KB	-
0x1450_4000	0x1450_4FFF	GPIO4	4KB	-
0x1450_5000	0x1450_5FFF	GPIO5	4KB	-
0x1450_6000	0x145F_FFFF	Reserved	-	-
0x1460_0000	0x1460_0FFF	CAN	4KB	
0x1460_1000	0x146F_FFFF	Reserved	-	-
0x1470_0000	0x1470_0FFF	GPT0	4KB	-
0x1470_1000	0x1470_1FFF	GPT1	4KB	-
0x1470_2000	0x1470_2FFF	GPT2	4KB	-
0x1470_3000	0x1470_3FFF	GPT3	4KB	-
0x1470_4000	0x1470_FFFF	Reserved	-	-
0x1471_0000	0x1472_7FFF	EFC	96KB	-
0x1472_8000	0x147D_FFFF	Reserved	-	-

起始地址	结束地址	功能	大小	说明
0x147E_0000	0x147E_0FFF	PMC	4KB	-
0x147E_1000	0x147E_FFFF	Reserved	-	-
0x147F_0000	0x147F_0FFF	IOCMG	4KB	-
0x147F_1000	0x147F_FFFF	Reserved	-	-
0x1480_0000	0x1480_0FFF	CRC	4KB	-
0x1480_1000	0x149F_FFFF	Reserved	-	-
0x14A0_0000	0x14A0_0FFF	APT0	4KB	-
0x14A0_1000	0x14A0_1FFF	APT1	4KB	-
0x14A0_2000	0x14A0_2FFF	APT2	4KB	-
0x14A0_3000	0x14A0_3FFF	APT3	4KB	-
0x14A0_4000	0x14AF_FFFF	Reserved	-	-
0x14B0_0000	0x14B0_3FFF	CAPM	16KB	CAPM0：0x0000~0x0FFF CAPM1：0x1000~0x1FFF CAPM2：0x2000~0x2FFF CAPM_COMM：0x3000~0x3FFF
0x14B0_4000	0x14BF_FFFF	Reserved	-	-
0x14C0_0000	0x14C0_0FFF	QDM0	4KB	-
0x14C0_1000	0x14C0_1FFF	QDM1	4KB	-
0x14C0_2000	0x17FF_FFFF	Reserved	-	-
0x1800_0000	0x1800_0FFF	ADC	4KB	-
0x1800_1000	0x181F_FFFF	Reserved	-	-
0x1820_0000	0x1820_0FFF	PGA0	4KB	-
0x1820_1000	0x1820_1FFF	PGA1	4KB	-
0x1820_2000	0x182F_FFFF	Reserved	-	-
0x1830_0000	0x1830_0FFF	ACMP	4KB	-
0x1830_1000	0x183F_FFFF	Reserved	-	-
0x1840_0000	0x1840_0FFF	DAC	4KB	-
0x1840_1000	0x184F_FFFF	Reserved	-	-
0x1850_0000	0x1850_0FFF	TSENSOR	4KB	-
0x1850_1000	0x185F_FFFF	Reserved	-	-

起始地址	结束地址	功能	大小	说明
0x1860_0000	0x1860_0FFF	ANA_CTRL_TOP	4KB	-
0x1860_1000	0x1BFF_FFFF	Reserved	-	-
0x1C00_0000	0x1C00_0FFF	DMA	4KB	-
0x1C00_1000	0x1FFF_FFFF	Reserved	-	-
0x2000_0000	0xFFFF_FFFF	Reserved	-	-

1.3 嵌入式 SYSRAM

嵌入式 SYSRAM 能够以最大系统时钟频率按字节、半字（16 位）或全字（32 位）访问，无需等待周期。

SYSRAM 总容量为 32KB，可根据实际应用按 4KB 粒度划分为两个 bank：SYSRAM0 和 SYSRAM1。当 CPU 的 ITCM 和 DTCM 分别访问 SYSRAM0 和 SYSRAM1 时，能获得最大性能。可通过系统控制器 SYSCTRL 的 SYSRAM_MAP 寄存器配置 SYSRAM 空间分配。

SYSRAM 支持奇偶校验功能，数据总线宽度为 36 位，其中有 4 位用于奇偶校验（每字节 1 位）。奇偶校验位在数据写入 SYSRAM 时进行计算和保存，然后在读取时自动进行校验。如果某一字节校验失败，则生成校验错误中断。该错误中断也连接到 APT0~3 的系统事件 sys_mem_fail，用于刹车控制。可通过系统控制器 SYSCTRL 的 SYSRAM_ERR 寄存器查询或清除 SYSRAM 奇偶校验错误标志。奇偶校验功能默认关闭，可通过将 Flash info_rgn1 的 sysram_parity_disable 标志位清零开启。

1.4 系统启动与固件升级

1.4.1 概述

MCU 内置启动 ROM（BOOTROM），当 MCU 复位撤销后 CPU 固定从 BOOTROM 开始执行启动引导程序。MCU 支持通过 UART、SPI、I2C 或 JTAG 对 Flash 内容进行固件升级（烧写）。

1.4.2 启动方式

MCU 支持两种启动模式：升级模式与正常启动模式，由 UPDATE_MODE 管脚上电锁存值决定。可通过系统控制器 SYSCTRL 的 SC_SYS_STAT 寄存器查询当前启动模式，或清除升级模式。

- 正常启动模式：BOOTROM 程序将直接跳转至 Flash 主存储区启动用户程序。但存在以下两种状态的任意一种时，CPU 将暂停启动用户程序，此时可以通过 JTAG/SWD 接口调试（或烧写）用户程序。
  - 状态 1：系统控制器 SYSCTRL 的 SC_HRST_REG0 寄存器值等于 0xA5A5A5A5。此寄存器不受系统软复位控制。
  - 状态 2：eFlash 主存储区首个字的值不等于 0xA37E95BD。
- 升级模式：BOOTROM 程序将初始化 UART、SPI、I2C 通信接口，与上位机通信完成固件升级、版本查询等功能。

1.5 中断映射

MCU 共支持 96 个外设中断，对应 CPU 中断号 26~121。外设中断映射如表 1-2 所示。中断处理详细信息，请参见 2 处理器章节。

表1-2 中断映射表

CPU 中断号	中断源
26	SOFTWARE
27	UART3
28	UART0
29	UART1
30	UART2
31	MTIMER
32	TIMER0
33	TIMER1
34	TIMER2
35	TIMER3
36	GPT0_INT

CPU 中断号	中断源
37	GPT0_PRD_INT
38	GPT1_INT
39	GPT1_PRD_INT
40	WWDG
41	保留
42	I2C0
43	I2C1
44	SPI0
45	SPI1
46	CAN
47	保留
48	APT0_EVT
49	APT0_TMR
50	APT1_EVT
51	APT1_TMR
52	APT2_EVT
53	APT2_TMR
54	APT3_EVT
55	APT3_TMR
56~67	保留
68	CMM
69	保留
70	CAPM0
71	CAPM1
72	CAPM2
73	QDM0
74	QDM1
75~76	保留
77	DMA_TC

CPU 中断号	中断源
78	DMA_ERR
79	SYSRAM_PARITY_ERR
80	保留
81	EFC
82	EFC_ERR
83	保留
84	ACMP
85	PVD
86	GPT2_INT
87	GPT2_PRD_INT
88	GPT3_INT
89	GPT3_PRD_INT
90~91	保留
92	ADC_EVENT
93	ADC_INT1
94	ADC_INT2
95	ADC_INT3
96	ADC_INT4
97~108	保留
109	GPIO0
110	GPIO1
111	GPIO2
112	GPIO3
113	GPIO4
114	GPIO5
115~121	保留

2 处理器

2.1 特性

2.1.1 功能描述

3061M 系列产品的处理器是一个基于 RISC-V ISA 设计的 32bit MCU 核，具有以下特点：

- 支持 RV32IMCF 指令集（I：32bit 整型基础指令集，M：整型乘除扩展，C：16bit 压缩指令扩展，F：单精度浮点扩展）+自定义指令集。
- 支持 Machine 和 User 特权模式。
- 支持物理内存保护 PMP（Physical Memory Protection）。
- 单发射 3 级顺序流水线微架构。
- 支持静态分支预测。
- 支持最大 4G Byte 地址空间，具体可访问范围由 MCU 总线架构决定。
- 支持通过指令紧耦合存储 ITCM（Instruction Tightly-Coupled Memory）接口访问指令和数据。
- 支持通过数据紧耦合存储 DTCM（Data Tightly-Coupled Memory）接口访问数据。
- 支持系统通过外部 AHB（Advanced High-performance Bus）总线接口访问 ITCM 和 DTCM。
- 支持 AHB 外设总线访问系统外设寄存器。
- 支持小端数据排布。
- 支持直接和向量中断模式。
- 支持 WFI（Wait For Interrupt）低功耗模式。
- 支持 RISC-V 标准调试机制。

- 支持通过 AHB 外设总线访问的定时器 MTimer。

RISC-V 相关文档：

- RISC-V 指令集非特权标准，https://github.com/riscv/riscv-isa-manual/releases/download/archive/riscv-spec-v2.2.pdf。
- RISC-V 指令集特权标准，https://github.com/riscv/riscv-isa-manual/releases/download/archive/riscv-privileged-v1.10.pdf。
- RISC-V 汇编编程手册，https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md。
- RISC-V 调试标准，https://github.com/riscv/riscv-debug-spec/releases/download/task_group_vote/riscv-debug-draft.pdf。

2.2 编程模型

2.2.1 通用寄存器组

RISC-V RV32I 基础指令集定义了 32 个 32bit 整型通用寄存器加上 1 个 32bit 的 PC 寄存器，对于支持单精度浮点扩展（RISC-V 标准 F 扩展）的处理器，额外定义了 32 个 32bit 的浮点通用寄存器。

PC 寄存器指示当前执行指令的地址，除了整型通用寄存器 X0 为恒 0 值，对 X0 的写操作会被忽略，其他通用寄存器则并无特殊的硬件规格差异，理论上可以互相替换使用。但 RISC-V 组织为了软件编程的统一性，制定了软件使用通用寄存器的一些约定，并定义了每个通用寄存器的 ABI（Application Binary Interface）名字，如 X1 作为函数调用的返回地址寄存器 ra（Return Address），X2 作为栈指针 sp（Stack Pointer），X3 作为全局指针 gp（Global Pointer）等。同时还约定了软件在函数调用时，哪些寄存器作为传入参数或返回值，哪些寄存器属于调用者（Caller）进行入栈保护，哪些寄存器属于被调用者（Callee）进行入栈保护。

表2-1 整型通用寄存器

整型通用寄存器	ABI 名字	描述	约定的保护者
x0	zero	Hard-wired zero	-
x1	ra	Return address	调用者（Caller）
x2	sp	Stack pointer	被调用者（Callee）

整型通用寄存器	ABI 名字	描述	约定的保护者
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5	t0	Temporary/alternate link register	调用者（Caller）
x6-7	t1–2	Temporaries	调用者（Caller）
x8	s0/fp	Saved register/frame pointer	被调用者（Callee）
x9	s1	Saved register	被调用者（Callee）
x10-11	a0–1	Function arguments/return values	调用者（Caller）
x12-17	a2–7	Function arguments	调用者（Caller）
x18-27	s2–11	Saved registers	被调用者（Callee）
x28-31	t3–6	Temporaries	调用者（Caller）

浮点通用寄存器是独立于整型通用寄存器的，RISC-V 规定每一条指令访问的资源是整型通用寄存器还是浮点通用寄存器，对于浮点整型转换等指令，可能需要同时用到整型和浮点通用寄存器。

表2-2 浮点通用寄存器

浮点通用寄存器	ABI 名字	描述	约定的保护者
f0-7	ft0-7	FP temporaries	调用者（Caller）
f8-9	fs0-1	FP saved registers	被调用者（Callee）
f10-11	fa0-1	FP arguments/return values	调用者（Caller）
f12-17	fa2-7	FP arguments	调用者（Caller）
f18-27	fs2-11	FP saved registers	被调用者（Callee）
f28-31	ft8-11	FP temporaries	调用者（Caller）

2.2.2 控制与状态寄存器

控制与状态寄存器 CSR（Control and Status Register）是用于控制处理器运行、提供处理器运行状态信息的寄存器。CSR 寄存器具有独立于系统地址空间的地址空间，所以访问系统地址空间的 load/store 指令并不能用于访问 CSR 寄存器，RISC-V 提供专用的 CSR 指令用于读写 CSR 寄存器，包括 CSRRW、CSRRS、CSRRC、CSRRWI、CSRRSI、CSRRCI 以及一些汇编伪指令，具体指令使用方式可参考《RISC-V 指令集非特权标准》、《RISC-V 指令集特权标准》和《RISC-V 汇编编程手册》。

RISC-V 标准根据 CSR 寄存器的可访问特权模式和是否支持写操作，将 CSR 寄存器分成几种类别，本处理器支持的 CSR 涉及到 URW、URO、MRW、MRO 四种类别属性。其中 URW 表示该 CSR 是用户模式读写寄存器，支持用户模式和机器模式进行读写访问。URO 表示该 CSR 是用户模式只读寄存器，支持用户模式和机器模式进行读访问，而对 URO 类别的 CSR 发起写操作会触发异常。MRW 表示该 CSR 是机器模式读写寄存器，支持机器模式下进行读写访问，而处于用户模式下进行读写访问会触发异常。MRO 表示该 CSR 是机器模式只读寄存器，只支持机器模式下进行读操作，而任何模式下执行写操作或者在用户模式下执行读操作都会触发异常。

说明

控制与状态寄存器的命名沿用 RISC-V 标准的风格，寄存器名为小写，位域名为大写。

2.2.3 标准 CSR 寄存器概览

标准 CSR 寄存器概览如表 2-3 所示。

表2-3 标准 CSR 寄存器概览

偏移地址	权限	名称	描述	页码
0xC00	URO	cycle	时钟拍数计数器的低 32bit 寄存器。	cycle
0xC02	URO	instret	指令退休计数器的低 32bit 寄存器。	instret
0xC80	URO	cycleh	时钟拍数计数器的高 32bit 寄存器。	cycleh
0xC82	URO	instreth	指令退休计数器的高 32bit 寄存器。	instreth
0x300	MRW	mstatus	机器模式状态寄存器。	mstatus
0x301	MRW	misa	指令集与扩展寄存器。	misa
0x304	MRW	mie	机器模式中断使能寄存器。	mie

偏移地址	权限	名称	描述	页码
0x305	MRW	mtvec	机器模式异常处理服务函数基地址寄存器。	mtvec
0x306	MRW	mcounteren	机器模式计数器使能寄存器。	mcounteren
0x340	MRW	mscratch	机器模式异常处理服务函数 scratch 寄存器。	mscratch
0x341	MRW	mepc	机器模式异常 PC 寄存器。	mepc
0x342	MRW	mcause	机器模式异常原因寄存器。	mcause
0x343	MRW	mtval	机器模式异常信息寄存器。	mtval
0x344	MRW	mip	机器模式中断挂起状态寄存器。	mip
0x3A0~0x3A1	MRW	pmpcfg0~ pmpcfg1	物理内存保护配置寄存器 0~1。	pmpcfg0~pmpcfg1
0x3B0~0x3B7	MRW	pmpaddr0~ pmpaddr7	物理内存保护地址寄存器 0~7。	pmpaddr0~ pmpaddr7
0xB00	MRW	mcycle	机器模式时钟拍数计数器的低 32bit 寄存器。	mcycle
0xB02	MRW	minstret	机器模式指令退休计数器的低 32bit 寄存器。	minstret
0xB80	MRW	mcycleh	机器模式时钟拍数计数器的高 32bit 寄存器。	mcycleh
0xB82	MRW	minstreth	机器模式指令退休计数器的高 32bit 寄存器。	minstreth
0x001	URW	fflags	浮点累积异常标识寄存器。	fflags
0x002	URW	frm	浮点动态舍入模式寄存器。	frm
0x003	URW	fcsr	浮点控制与状态寄存器。	fcsr

2.2.4 标准 CSR 寄存器描述

cycle

cycle 为时钟拍数计数器的低 32bit 寄存器。

Offset Address: 0xC00 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	CYCLE	时钟拍数计数器的低 32bit，处理器每拍时钟加 1；是 mcycle 的别名寄存器，与 mcycle 共用同一个物理寄存器；仅当 mcounteren[CY]为 1 时，用户模式下可访问此寄存器。	0x00000000

instret

instret 为指令退休计数器的低 32bit 寄存器。

Offset Address: 0xC02 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	INSTRET	指令退休计数器的低 32bit，处理器每完成一条指令加 1；是 minstret 的别名寄存器，与 minstret 共用同一个物理寄存器；仅当 mcounteren[IR]为 1 时，用户模式下可访问此寄存器。	0x00000000

cycleh

cycleh 为时钟拍数计数器的高 32bit 寄存器。

Offset Address: 0xC80 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	CYCLEH	时钟拍数计数器的高 32bit，当 cycle 计数溢出时加 1；是 mcycleh 的别名寄存器，与 mcycleh 共用同一个物理寄存器；仅当 mcounteren[CY]为 1 时，用户模式下可访	0x00000000

Bits	Access	Name	Description	Reset
			问此寄存器。	

instreth

instreth 为指令退休计数器的高 32bit 寄存器。

Offset Address: 0xC82 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	INSTRETH	指令退休计数器的高 32bit，当 instret 计数溢出时加 1；是 minstreth 的别名寄存器，与 minstreth 共用同一个物理寄存器；仅当 mcounteren[IR]为 1 时，用户模式下可访问此寄存器。	0x00000000

mstatus

mstatus 为机器模式状态寄存器。

Offset Address: 0x300 Total Reset Value: 0x0000_1800

Bits	Access	Name	Description	Reset
[31]	RO	SD	脏（dirty）状态汇总，当 FS 位域的值为 11 时，此位域为 1，否则此位域为 0。	0x0
[30:18]	-	RESERVED	保留。	0x0000
[17]	RW	MPRV	修改加载（load）和存储（store）查询 PMP 时使用的特权模式，具体功能请参考《RISC-V 指令集特权标准》。 0：load 和 store 操作使用当前处理器的特权模式进行 PMP 查询； 1：load 和 store 操作使用 MPP 位域的值作为特权模式进行 PMP 查询。	0x0
[16:15]	-	RESERVED	保留。	0x0
[14:13]	RW	FS	浮点单元状态，具体功能请参考《RISC-V	0x0

Bits	Access	Name	Description	Reset
			指令集特权标准》。 00：关闭状态（off），此时所有访问浮点通用寄存器和浮点控制与状态寄存器的指令都会上报异常，软件使用浮点功能前必须设置成其他状态值； 01：初始状态（initial），此时浮点单元可用，执行任何写浮点通用寄存器或者浮点控制与状态寄存器的指令会使此位域变成11； 10：干净状态（clean），此时浮点单元可用，执行任何写浮点通用寄存器或者浮点控制与状态寄存器的指令会使此位域变成11； 11：脏状态（dirty），此时浮点单元可用，执行任何写浮点通用寄存器或者浮点控制与状态寄存器的指令，此位域不变。	
[12:11]	RW	MPP	处理器响应异常前所处的特权模式，具体功能请参考《RISC-V 指令集特权标准》。 00：用户模式； 11：机器模式； 其他：保留。	0x3
[10:8]	-	RESERVED	保留。	0x0
[7]	RW	MPIE	处理器响应异常前的中断使能情况，具体功能请参考《RISC-V 指令集特权标准》。 0：中断关闭； 1：中断打开。	0x0
[6:4]	-	RESERVED	保留。	0x0
[3]	RW	MIE	机器模式中断使能，具体功能请参考《RISC-V 指令集特权标准》。 0：中断关闭；	0x0

Bits	Access	Name	Description	Reset
			1：中断打开。	
[2:0]	-	RESERVED	保留。	0x0

misa

misa 为指令集与扩展寄存器。

Offset Address: 0x301 Total Reset Value: 0x4090_1124

Bits	Access	Name	Description	Reset
[31:30]	RO	MXL	机器模式整型通用寄存器位宽。 01：支持 32bit 整型通用寄存器； 其他：保留。	0x1
[29:24]	-	RESERVED	保留。	0x00
[23]	RO	X	非标准扩展（自定义指令和中断机制）。 0：保留； 1：支持。	0x1
[22:21]	-	RESERVED	保留。	0x0
[20]	RO	U	用户模式。 0：保留； 1：支持。	0x1
[19:13]	-	RESERVED	保留。	0x00
[12]	RO	M	整型乘除法扩展。 0：保留； 1：支持。	0x1
[11:9]	-	RESERVED	保留。	0x0
[8]	RO	I	RV32I/64I/128I 整型基础指令集。 0：保留； 1：支持。	0x1

Bits	Access	Name	Description	Reset
[7:6]	-	RESERVED	保留。	0x0
[5]	RW	F	单精度浮点扩展，不支持浮点单元的处理器，此位域保留。 0：不支持； 1：支持。	0x1
[4:3]	-	RESERVED	保留。	0x0
[2]	RO	C	压缩指令集扩展。 0：保留； 1：支持。	0x1
[1:0]	-	RESERVED	保留。	0x0

mie

mie 为机器模式中断使能寄存器。

Offset Address: 0x304 Total Reset Value: 0x0000_1000

Bits	Access	Name	Description	Reset
[31]	RW	LOCIE31	本地中断 31 使能。 0：不使能； 1：使能。	0x0
[30]	RW	LOCIE30	本地中断 30 使能。 0：不使能； 1：使能。	0x0
[29]	RW	LOCIE29	本地中断 29 使能。 0：不使能； 1：使能。	0x0
[28]	RW	LOCIE28	本地中断 28 使能。 0：不使能； 1：使能。	0x0

Bits	Access	Name	Description	Reset
[27]	RW	LOCIE27	本地中断 27 使能。 0：不使能; 1：使能。	0x0
[26]	RW	LOCIE26	本地中断 26 使能。 0：不使能; 1：使能。	0x0
[25:13]	-	RESERVED	保留。	0x0000
[12]	-	RESERVED	保留。	0x1
[11:0]	-	RESERVED	保留。	0x000

mtvec

mtvec 为机器模式异常处理服务函数基地址寄存器。

Offset Address: 0x305 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RW	BASE	中断异常向量表基地址的 31bit~2bit。	0x00000000
[1]	-	RESERVED	保留。	0x0
[0]	RW	MODE	中断异常处理函数的地址模式。 0：直接模式，所有异常和中断统一跳转到基地址; 1：向量模式，异常跳转到基地址，中断跳转到基地址偏移 4*中断编号的地址。	0x0

mcounteren

mcounteren 为机器模式计数器使能寄存器。

Offset Address: 0x306 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:3]	-	RESERVED	保留。	0x00000000
[2]	RW	IR	用户模式下 instret CSR 的访问使能。 0：用户模式下访问 instret 或者 instreth CSR 会触发异常； 1：用户模式下运行访问 instret 和 instreth CSR。	0x0
[1]	-	RESERVED	保留。	0x0
[0]	RW	CY	用户模式下 cycle CSR 的访问使能。 0：用户模式下访问 cycle 或者 cycleh CSR 会触发异常； 1：用户模式下运行访问 cycle 和 cycleh CSR。	0x0

mscratch

mscratch 为机器模式异常处理服务函数 scratch 寄存器。

Offset Address: 0x340 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MSCRATCH	没有特殊硬件功能的临时 CSR，软件一般用于换栈操作。	0x00000000

mepc

mepc 为机器模式异常 PC 寄存器。

Offset Address: 0x341 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RW	MEPC	在处理器响应中断或者异常时，此寄存器记录触发异常或者被中断打断的指令的 PC 值； 在执行 mret 指令时，处理器会将 pc 值设置	0x00000000

Bits	Access	Name	Description	Reset
			成 mepc 值； 详细说明请参考《RISC-V 指令集特权标准》。	
[0]	-	RESERVED	保留。	0x0

mcause

mcause 为机器模式异常原因寄存器。

Offset Address: 0x342 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RW	INTERRUPT	指示处理器响应的是中断还是异常。 0：异常； 1：中断。	0x0
[30:0]	RW	EXCEPTION_CODE	当 INTERRUPT 位域是 1 时： 此位域记录的是处理器响应的中断的中断编号，具体中断映射请参见表 1-2。 当 INTERRUPT 位域是 0 时： 此位域记录的是处理器响应的异常原因代号。 0x0000_0000：指令地址非对齐； 0x0000_0001：指令访问错误； 0x0000_0002：非法指令； 0x0000_0003：断点 breakpoint； 0x0000_0004：读数据地址非对齐； 0x0000_0005：读数据访问错误； 0x0000_0006：写数据地址非对齐； 0x0000_0007：写数据访问错误； 0x0000_0008：用户模式下的环境调用 ECALL； 0x0000_000B：机器模式下的环境调用	0x00000000

Bits	Access	Name	Description	Reset
			ECALL； 其他：保留。	

mtval

mtval 为机器模式异常信息寄存器。

Offset Address: 0x343 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MTVAL	异常信息，根据异常类型记录的可能是异常地址或者是异常指令机器码。 详细说明请参考《RISC-V 指令集特权标准》。	0x00000000

mip

mip 为机器模式中断挂起状态寄存器。

Offset Address: 0x344 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RO	LOCIP31	本地中断 31 挂起状态。 0：没挂起； 1：挂起。	0x0
[30]	RO	LOCIP30	本地中断 30 挂起状态。 0：没挂起； 1：挂起。	0x0
[29]	RO	LOCIP29	本地中断 29 挂起状态。 0：没挂起； 1：挂起。	0x0
[28]	RO	LOCIP28	本地中断 28 挂起状态。 0：没挂起；	0x0

Bits	Access	Name	Description	Reset
			1：挂起。	
[27]	RO	LOCIP27	本地中断 27 挂起状态。 0：没挂起； 1：挂起。	0x0
[26]	RO	LOCIP26	本地中断 26 挂起状态。 0：没挂起； 1：挂起。	0x0
[25:0]	-	RESERVED	保留。	0x0000000

pmpcfg0~pmpcfg1

pmpcfg0~1 为物理内存保护配置寄存器 0~1。

Offset Address: 0x3A0~0x3A1 Total Reset Value: -

每个 config 寄存器包括 4 个 PMP 区间的配置，下表仅为 pmpcfg0 各位域的说明，pmpcfg0 控制了 PMP 区间 0~3 的配置，而 pmpcfg1 以相同的位域排布方式控制 PMP 区间 4~7 的配置。

Bits	Access	Name	Description	Reset
[31]	RW	PMP3CFG_L	锁定状态。 0：区间的配置寄存器未锁定，权限控制仅对用户模式的访问生效； 1：区间的配置寄存器锁定，写无效，且权限控制对用户模式和机器模式的访问生效。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[30:29]	-	RESERVED	保留。	0x0
[28:27]	RW	PMP3CFG_A	地址模式。 00：OFF，区间无效； 01：TOR；	0x0

Bits	Access	Name	Description	Reset
			其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	
[26]	RW	PMP3CFG_X	区间执行权限控制。 0：区间不可执行指令； 1：区间可执行指令。	-
[25]	RW	PMP3CFG_W	区间写权限控制。 0：区间不可执行写操作； 1：区间可执行写操作。	-
[24]	RW	PMP3CFG_R	区间读权限控制。 0：区间不可执行读操作； 1：区间可执行读操作。	-
[23]	RW	PMP2CFG_L	锁定状态。 0：区间的配置寄存器未锁定，权限控制仅对用户模式的访问生效； 1：区间的配置寄存器锁定，写无效，且权限控制对用户模式和机器模式的访问生效。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[22:21]	-	RESERVED	保留。	0x0
[20:19]	RW	PMP2CFG_A	地址模式。 00：OFF，区间无效； 01：TOR； 其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[18]	RW	PMP2CFG_X	区间执行权限控制。 0：区间不可执行指令；	-

Bits	Access	Name	Description	Reset
			1：区间可执行指令。	
[17]	RW	PMP2CFG_W	区间写权限控制。 0：区间不可执行写操作； 1：区间可执行写操作。	-
[16]	RW	PMP2CFG_R	区间读权限控制。 0：区间不可执行读操作； 1：区间可执行读操作。	-
[15]	RW	PMP1CFG_L	锁定状态。 0：区间的配置寄存器未锁定，权限控制仅对用户模式的访问生效； 1：区间的配置寄存器锁定，写无效，且权限控制对用户模式和机器模式的访问生效。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[14:13]	-	RESERVED	保留。	0x0
[12:11]	RW	PMP1CFG_A	地址模式。 00：OFF，区间无效； 01：TOR； 其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[10]	RW	PMP1CFG_X	区间执行权限控制。 0：区间不可执行指令； 1：区间可执行指令。	-
[9]	RW	PMP1CFG_W	区间写权限控制。 0：区间不可执行写操作； 1：区间可执行写操作。	-

Bits	Access	Name	Description	Reset
[8]	RW	PMP1CFG_R	区间读权限控制。 0：区间不可执行读操作； 1：区间可执行读操作。	-
[7]	RW	PMP0CFG_L	锁定状态。 0：区间的配置寄存器未锁定，权限控制仅对用户模式的访问生效； 1：区间的配置寄存器锁定，写无效，且权限控制对用户模式和机器模式的访问生效。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[6:5]	-	RESERVED	保留。	0x0
[4:3]	RW	PMP0CFG_A	地址模式。 00：OFF，区间无效； 01：TOR； 其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[2]	RW	PMP0CFG_X	区间执行权限控制。 0：区间不可执行指令； 1：区间可执行指令。	-
[1]	RW	PMP0CFG_W	区间写权限控制。 0：区间不可执行写操作； 1：区间可执行写操作。	-
[0]	RW	PMP0CFG_R	区间读权限控制。 0：区间不可执行读操作； 1：区间可执行读操作。	-

pmpaddr0~ pmpaddr7

pmpaddr0~7 为物理内存保护地址寄存器 0~7。

Offset Address: 0x3B0~0x3B7 Total Reset Value: -

Bits	Access	Name	Description	Reset
[31:0]	RW	PMPADDR0~ PMPADDR7	物理内存保护地址寄存器 0~7， 详细说明请参考《RISC-V 指令集特权标准》。	-

mcycle

mcycle 为机器模式时钟拍数计数器的低 32bit 寄存器。

Offset Address: 0xB00 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MCYCLE	机器模式的时钟拍数计数寄存器的低 32bit，处理器的每拍时钟加 1。	0x00000000

minstret

minstret 为机器模式指令退休计数器的低 32bit 寄存器。

Offset Address: 0xB02 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MINSTRET	机器模式的指令退休计数寄存器的低 32bit，处理器每完成一条指令加 1。	0x00000000

mcycleh

mcycleh 为机器模式时钟拍数计数器高 32bit 寄存器。

Offset Address: 0xB80 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MCYCLEH	时钟拍数计数器的高 32bit，当 mcycle 计数溢出时加 1。	0x00000000

minstreth

minstreth 为机器模式指令退休计数器高 32bit 寄存器。

Offset Address: 0xB82 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MINSTRETH	指令退休计数器的高 32bit，当 minstret 计数溢出时加 1。	0x00000000

fflags

fflags 为浮点累积异常标识寄存器。

Offset Address: 0x001 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	RESERVED	保留。	0x0000000
[4]	RW	NV	浮点运算非法操作异常标识； 为 fcsr 寄存器 NV 位域的别名寄存器； 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[3]	RW	DZ	浮点运算除 0 异常标识； 为 fcsr 寄存器 DZ 位域的别名寄存器； 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[2]	RW	OF	浮点运算上溢异常标识； 为 fcsr 寄存器 OF 位域的别名寄存器； 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[1]	RW	UF	浮点运算下溢异常标识； 为 fcsr 寄存器 UF 位域的别名寄存器； 详细说明请参考《RISC-V 指令集特权标准》。	0x0

Bits	Access	Name	Description	Reset
[0]	RW	NX	浮点运算非精确异常标识； 为 fcsr 寄存器 NX 位域的别名寄存器； 详细说明请参考《RISC-V 指令集特权标准》。	0x0

frm

frm 为浮点动态舍入模式寄存器。

Offset Address: 0x002 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	RESERVED	保留。	0x00000000
[2:0]	RW	FRM	浮点运算舍入模式，为 fcsr 寄存器 frm 位域的别名寄存器。 000：RNE，舍入到最接近的值，当舍入距离一样时，舍入为偶数； 001：RTZ，向 0 进行舍入； 010：RDN，向负无穷进行舍入； 011：RUP，向正无穷进行舍入； 100：RMM，舍入到最接近的值，当舍入距离一样时，舍入为量级更大的数； 其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	0x0

fcsr

fcsr 为浮点控制与状态寄存器。

Offset Address: 0x003 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:8]	-	RESERVED	保留。	0x000000
[7:5]	RW	FRM	浮点运算舍入模式。 000：RNE，舍入到最接近的值，当舍入距离一样时，舍入为偶数； 001：RTZ，向 0 进行舍入； 010：RDN，向负无穷进行舍入； 011：RUP，向正无穷进行舍入； 100：RMM，舍入到最接近的值，当舍入距离一样时，舍入为量级更大的数； 其他：保留。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[4]	RW	NV	浮点运算非法操作异常标识。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[3]	RW	DZ	浮点运算除 0 异常标识。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[2]	RW	OF	浮点运算上溢异常标识。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[1]	RW	UF	浮点运算下溢异常标识。 详细说明请参考《RISC-V 指令集特权标准》。	0x0
[0]	RW	NX	浮点运算非精确异常标识。 详细说明请参考《RISC-V 指令集特权标准》。	0x0

2.2.5 自定义 CSR 寄存器概览

自定义 CSR 寄存器概览如表 2-4 所示。

表2-4 自定义 CSR 寄存器概览

偏移地址	权限	名称	描述	页码
0x7C8	MRW	mibnk	硬件中断嵌套控制寄存器。	mibnk
0x7D1	MRW	tcmarb	ITCM 和 DTCM 访问仲裁控制寄存器。	tcmarb
0x7D8	MRW	memattr	PMP 区间 0~7 的存储器属性寄存器。	memattr
0x7ED	MRO	cipri	当前中断优先级寄存器。	cipri
0x7DA	MRW	mtflag	机器模式中断标识寄存器。	mtflag
0x7EF	MRW	mtveclock	机器模式异常处理服务函数基地址锁定寄存器。	mtveclock
0xBC0~0xBCB	MRW	locipri0~ locipri11	非标准本地中断优先级寄存器 0~11。	locipri0~ locipri11
0xBE0~0xBE2	MRW	locien0~ locien2	非标准本地中断使能寄存器 0~2。	locien0~ locien2
0xBE8~0xBEA	MRW	locipd0~ locipd2	非标准本地中断挂起状态寄存器 0~2。	locipd0~ locipd2
0xBF0	MRW	locipclr	非标准本地中断挂起状态清除寄存器。	locipclr
0xBFE	MRW	prithd	中断优先级阈值寄存器。	prithd
0xFC2	MRO	cxcptsc	自定义异常状态和原因信息寄存器。	cxcptsc

2.2.6 自定义 CSR 寄存器描述

mibnk

mibnk 为硬件中断嵌套控制寄存器。

Offset Address: 0x7C8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	RESERVED	保留。	0x00000000
[1]	RW	HSS	硬件自动换栈使能。 0：不使能； 1：使能（仅在硬件中断嵌套使能的情况下支持）。	0x0
[0]	RW	MB	硬件中断嵌套使能。 0：不使能； 1：使能。	0x0

tcmarb

tcmarb 为 ITCM 和 DTCM 访问仲裁控制寄存器。

Offset Address: 0x7D1 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:9]	-	RESERVED	保留。	0x000000
[8:7]	RW	EPRIV	外部访问总线的优先级。 仅在 RR 位域为 0 时有效。	0x0
[6]	-	RESERVED	保留。	0x0
[5:4]	RW	DPRIV	数据读写访问的优先级。 仅在 RR 位域为 0 时有效。	0x0
[3]	-	RESERVED	保留。	0x0
[2:1]	RW	IPRIV	读取指令访问的优先级。	0x0

Bits	Access	Name	Description	Reset
			仅在 RR 位域为 0 时有效。	
[0]	RW	RR	ITCM 和 DTCM 访问仲裁模式。 0：固定优先级； 1：轮询优先级 RR（Round Robin）。	0x1

mematttr

mematttr 为 PMP 区间 0~7 的存储器属性寄存器。

Offset Address: 0x7D8 Total Reset Value: -

Bits	Access	Name	Description	Reset
[31:28]	RW	ATTR7	PMP 区间 7 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xA：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-
[27:24]	RW	ATTR6	PMP 区间 6 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-

Bits	Access	Name	Description	Reset
[23:20]	RW	ATTR5	PMP 区间 5 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-
[19:16]	RW	ATTR4	PMP 区间 4 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-
[15:12]	RW	ATTR3	PMP 区间 3 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-

Bits	Access	Name	Description	Reset
[11:8]	RW	ATTR2	PMP 区间 2 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-
[7:4]	RW	ATTR1	PMP 区间 1 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-
[3:0]	RW	ATTR0	PMP 区间 0 的存储器属性。 0x0：device non-bufferable； 0x3：normal non-cacheable bufferable； 0x7：效果等同 0x3 normal non-cacheable bufferable； 0xB：效果等同 0x3 normal non-cacheable bufferable； 0xF：效果等同 0x3 normal non-cacheable bufferable； 其他：保留。	-

cipri

cipri 为当前中断优先级寄存器。

Offset Address: 0x7ED Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	RESERVED	保留。	0x00000000
[3:0]	RO	CIPRI	当前处理器响应的中断的优先级。	0x0

mtflag

mtflag 为机器模式中断标识寄存器。

Offset Address: 0x7DA Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	MTFLAG	没有特殊硬件用途的临时寄存器，类似于 mscratch 提供给软件使用。	0x00000000

mtveclock

mtveclock 为机器模式异常处理服务函数基地址锁定寄存器。

Offset Address: 0x7EF Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	RESERVED	保留。	0x00000000
[0]	RW	LOCK	mtvec 寄存器的锁定状态。 0：mtvec 寄存器可写； 1：mtvec 寄存器和 mtveclock 寄存器都被锁定，只能读不可写。	0x0

locipri0~ locipri11

locipri0~ locipri11 为非标准本地中断优先级寄存器 0~11。

Offset Address: 0xBC0~0xBCB Total Reset Value: 0x0000_0000

每个 locipri 寄存器包括 8 个非标准本地中断优先级的配置，下表仅为 locipri0 各位域的说明，locipri0 控制了本地中断 26~33 的优先级，而 locipri1~ locipri11 以相同的位域排布方式控制本地中断 34~121 的优先级。每个中断的中断优先级可设置为 0~15，数值越大优先级越高，即 15 为最高优先级，而中断优先级设置为 0 的中断，不会被处理器响应，相当于没有打开该中断的使能。如果多个中断设置的中断优先级相同，则中断编号越小的中断优先级越高。

Bits	Access	Name	Description	Reset
[31:28]	RW	PRI33	本地中断 33 的优先级。	0x0
[27:24]	RW	PRI32	本地中断 32 的优先级。	0x0
[23:20]	RW	PRI31	本地中断 31 的优先级。	0x0
[19:16]	RW	PRI30	本地中断 30 的优先级。	0x0
[15:12]	RW	PRI29	本地中断 29 的优先级。	0x0
[11:8]	RW	PRI28	本地中断 28 的优先级。	0x0
[7:4]	RW	PRI27	本地中断 27 的优先级。	0x0
[3:0]	RW	PRI26	本地中断 26 的优先级。	0x0

locien0~ locien2

locien0~2 为非标准本地中断使能寄存器 0~2。

Offset Address: 0xBE0~0xBE2 Total Reset Value: 0x0000_0000

每个 locien 寄存器包括 32 个非标准本地中断使能的配置，下表仅为 locien0 各位域的说明，locien0 控制了本地中断 32~63 的使能，而 locien1~ locien2 以相同的位域排布方式控制本地中断 64~121 的使能。

Bits	Access	Name	Description	Reset
[31]	RW	EN63	本地中断 63 使能。 0：不使能； 1：使能。	0x0
[30]	RW	EN62	本地中断 62 使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[29]	RW	EN61	本地中断 61 使能。 0：不使能； 1：使能。	0x0
[28]	RW	EN60	本地中断 60 使能。 0：不使能； 1：使能。	0x0
[27]	RW	EN59	本地中断 59 使能。 0：不使能； 1：使能。	0x0
[26]	RW	EN58	本地中断 58 使能。 0：不使能； 1：使能。	0x0
[25]	RW	EN57	本地中断 57 使能。 0：不使能； 1：使能。	0x0
[24]	RW	EN56	本地中断 56 使能。 0：不使能； 1：使能。	0x0
[23]	RW	EN55	本地中断 55 使能。 0：不使能； 1：使能。	0x0
[22]	RW	EN54	本地中断 54 使能。 0：不使能； 1：使能。	0x0
[21]	RW	EN53	本地中断 53 使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[20]	RW	EN52	本地中断 52 使能。 0：不使能； 1：使能。	0x0
[19]	RW	EN51	本地中断 51 使能。 0：不使能； 1：使能。	0x0
[18]	RW	EN50	本地中断 50 使能。 0：不使能； 1：使能。	0x0
[17]	RW	EN49	本地中断 49 使能。 0：不使能； 1：使能。	0x0
[16]	RW	EN48	本地中断 48 使能。 0：不使能； 1：使能。	0x0
[15]	RW	EN47	本地中断 47 使能。 0：不使能； 1：使能。	0x0
[14]	RW	EN46	本地中断 46 使能。 0：不使能； 1：使能。	0x0
[13]	RW	EN45	本地中断 45 使能。 0：不使能； 1：使能。	0x0
[12]	RW	EN44	本地中断 44 使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[11]	RW	EN43	本地中断 43 使能。 0：不使能； 1：使能。	0x0
[10]	RW	EN42	本地中断 42 使能。 0：不使能； 1：使能。	0x0
[9]	RW	EN41	本地中断 41 使能。 0：不使能； 1：使能。	0x0
[8]	RW	EN40	本地中断 40 使能。 0：不使能； 1：使能。	0x0
[7]	RW	EN39	本地中断 39 使能。 0：不使能； 1：使能。	0x0
[6]	RW	EN38	本地中断 38 使能。 0：不使能； 1：使能。	0x0
[5]	RW	EN37	本地中断 37 使能。 0：不使能； 1：使能。	0x0
[4]	RW	EN36	本地中断 36 使能。 0：不使能； 1：使能。	0x0
[3]	RW	EN35	本地中断 35 使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[2]	RW	EN34	本地中断 34 使能。 0：不使能； 1：使能。	0x0
[1]	RW	EN33	本地中断 33 使能。 0：不使能； 1：使能。	0x0
[0]	RW	EN32	本地中断 32 使能。 0：不使能； 1：使能。	0x0

locipd0~ locipd2

locipd0~2 为非标准本地中断挂起状态寄存器 0~2。

Offset Address: 0xBE8~0xBEA Total Reset Value: 0x0000_0000

每个 locipd 寄存器包括 32 个非标准本地中断的挂起状态，下表仅为 locipd0 各位域的说明，locipd0 包含本地中断 32~63 的挂起状态，而 locipd1~ locipd2 以相同的位域排布方式表示本地中断 64~121 的挂起状态。

Bits	Access	Name	Description	Reset
[31]	RO	PD63	本地中断 63 挂起状态。 0：没挂起； 1：挂起。	0x0
[30]	RO	PD62	本地中断 62 挂起状态。 0：没挂起； 1：挂起。	0x0
[29]	RO	PD61	本地中断 61 挂起状态。 0：没挂起；	0x0

Bits	Access	Name	Description	Reset
			1：挂起。	
[28]	RO	PD60	本地中断 60 挂起状态。 0：没挂起； 1：挂起。	0x0
[27]	RO	PD59	本地中断 59 挂起状态。 0：没挂起； 1：挂起。	0x0
[26]	RO	PD58	本地中断 58 挂起状态。 0：没挂起； 1：挂起。	0x0
[25]	RO	PD57	本地中断 57 挂起状态。 0：没挂起； 1：挂起。	0x0
[24]	RO	PD56	本地中断 56 挂起状态。 0：没挂起； 1：挂起。	0x0
[23]	RO	PD55	本地中断 55 挂起状态。 0：没挂起； 1：挂起。	0x0
[22]	RO	PD54	本地中断 54 挂起状态。 0：没挂起； 1：挂起。	0x0
[21]	RO	PD53	本地中断 53 挂起状态。 0：没挂起； 1：挂起。	0x0
[20]	RO	PD52	本地中断 52 挂起状态。 0：没挂起；	0x0

Bits	Access	Name	Description	Reset
			1：挂起。	
[19]	RO	PD51	本地中断 51 挂起状态。 0：没挂起； 1：挂起。	0x0
[18]	RO	PD50	本地中断 50 挂起状态。 0：没挂起； 1：挂起。	0x0
[17]	RO	PD49	本地中断 49 挂起状态。 0：没挂起； 1：挂起。	0x0
[16]	RO	PD48	本地中断 48 挂起状态。 0：没挂起； 1：挂起。	0x0
[15]	RO	PD47	本地中断 47 挂起状态。 0：没挂起； 1：挂起。	0x0
[14]	RO	PD46	本地中断 46 挂起状态。 0：没挂起； 1：挂起。	0x0
[13]	RO	PD45	本地中断 45 挂起状态。 0：没挂起； 1：挂起。	0x0
[12]	RO	PD44	本地中断 44 挂起状态。 0：没挂起； 1：挂起。	0x0
[11]	RO	PD43	本地中断 43 挂起状态。 0：没挂起；	0x0

Bits	Access	Name	Description	Reset
			1：挂起。	
[10]	RO	PD42	本地中断 42 挂起状态。 0：没挂起； 1：挂起。	0x0
[9]	RO	PD41	本地中断 41 挂起状态。 0：没挂起； 1：挂起。	0x0
[8]	RO	PD40	本地中断 40 挂起状态。 0：没挂起； 1：挂起。	0x0
[7]	RO	PD39	本地中断 39 挂起状态。 0：没挂起； 1：挂起。	0x0
[6]	RO	PD38	本地中断 38 挂起状态。 0：没挂起； 1：挂起。	0x0
[5]	RO	PD37	本地中断 37 挂起状态。 0：没挂起； 1：挂起。	0x0
[4]	RO	PD36	本地中断 36 挂起状态。 0：没挂起； 1：挂起。	0x0
[3]	RO	PD35	本地中断 35 挂起状态。 0：没挂起； 1：挂起。	0x0
[2]	RO	PD34	本地中断 34 挂起状态。 0：没挂起；	0x0

Bits	Access	Name	Description	Reset
			1：挂起。	
[1]	RO	PD33	本地中断 33 挂起状态。 0：没挂起； 1：挂起。	0x0
[0]	RO	PD32	本地中断 32 挂起状态。 0：没挂起； 1：挂起。	0x0

locipclr

locipclr 为非标准本地中断挂起状态清除寄存器。

Offset Address: 0xBF0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	-	RESERVED	保留。	0x000000
[7:0]	RW	ID	写入本地中断的中断编号可清除对应本地中断的挂起状态； 仅支持非标准本地中断，注意非标准本地中断的中断编号从 26 开始。	0x00

prithd

prithd 为中断优先级阈值寄存器。

Offset Address: 0xBFE Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	RESERVED	保留。	0x0000000
[3:0]	RW	THD	本地中断优先级阈值； 只有使能且优先级大于优先级阈值的处于挂起状态的中断会被处理器响应或者将处理器从睡眠模式中唤醒。	0x0

Bits	Access	Name	Description	Reset
			注意：由于中断优先级阈值最小为 0，所以优先级为 0 的中断永远都不会被响应。	

cxcptsc

cxcptsc 为自定义异常状态和原因信息寄存器。

Offset Address: 0xFC2 Total Reset Value: -

Bits	Access	Name	Description	Reset
[31]	RO	LKUP	处理器核死锁状态。 0：核未处于死锁状态； 1：核处于死锁状态。	0x0
[30]	RO	HDFT	处理器核硬件错误状态。 0：核未处于硬件错误状态； 1：核处于硬件错误状态。	0x0
[29:4]	-	RESERVED	保留。	0x0000000
[3:0]	RO	CCAUSE	自定义异常原因信息。 0x0：无效的异常原因编码； 0x1：请求地址不属于处理器可访问的地址区间； 0x2：保留； 0x3：AHB 外设总线返回总线错误； 0x4：请求跨 PMP 区间； 0x5：处理器内部的系统寄存器访问错误； 0x6：没有匹配到 PMP 区间； 0x7：PMP 查询错误，没有 PMP 权限； 0x8：保留； 0x9：CSR 访问错误； 0xA：LDMIA/STMIA/PUSH/POP/POPRET 指令错误；	-

Bits	Access	Name	Description	Reset
			其他：保留。	

2.2.7 特权模式

处理器支持机器模式（Machine Mode）和用户模式（User Mode）两种 RISC-V 定义的特权模式。复位后处理器运行在机器模式，用户可通过指令切换到用户模式。无论正在运行程序的是什么模式，当出现异常和中断的时候，处理器都会切换到机器模式进行中断和异常处理。

机器模式和用户模式的主要区别：

- CSR 访问：处理器处于机器模式下可以访问所有的 CSR 寄存器，而处于用户模式下只能访问用户模式 CSR 寄存器（权限为 URW、URO），如果访问机器模式 CSR 寄存器（权限为 MRW、MRO）会上报异常，具体用户模式 CSR 寄存器和机器模式 CSR 寄存器的区分请参考《RISC-V 指令集特权标准》的定义。
- PMP 权限查询：处理器处于机器模式下默认拥有整个地址空间的访问权限，包括读写和执行权限，除非某一个 PMP 区间处于锁定状态，机器模式下访问锁定的 PMP 区间才会受对应的 PMP 区间权限约束。而用户模式下则是默认没有任何地址的访问权限，只有访问一个有效 PMP 区间，同时权限满足要求的情况下，才能正常访问该区间的地址，否则上报异常。具体的 PMP 功能请参考《RISC-V 指令集特权标准》的相关章节。
- 中断屏蔽：机器模式下，mstatus[MIE]为 0 处理器不会响应除 NMI 外的其他中断。而在用户模式下，处理器总是会响应所有中断，不受 mstatus[MIE]的影响。
- 特权指令的执行：mret/wfi/ecall 这些特殊指令在机器模式和在用户模式下执行的行为是有差异的。

表2-5 Mret/wfi/ecall 指令在不同特权模式下的行为差异

指令	机器模式	用户模式
mret	将处理器的特权模式设置为 mstatus[MPP]中设置的模式，同时将 mstatus[MPIE]的值复制到 mstatus[MIE]控制中断开关，然后将 PC 设置为 mepc 寄存器中的值。	上报非法指令异常。
wfi	进入 sleep mode，直到被中断或者 halt req 唤醒。	当作 NOP 指令处理。
ecall	触发异常，mcause 为 11，表示是从机器模式状态执行的 ecall	触发异常，mcause 为 8，

指令	机器模式	用户模式
	指令。	表示是从用户模式状态执行的 ecall 指令。

从机器模式切换到用户模式：

步骤 1 将 mstatus[MIE]设置为 0 屏蔽所有中断。

步骤 2 将 mstatus[MPP]设置为 0（用户模式）。

步骤 3 将 mepc 设置为进入用户模式后将要执行的第一条指令的 PC。

步骤 4 执行 mret 指令，处理器会进入用户模式并自动将 PC 设置成 mepc 的值。

----结束

从用户模式切换到机器模式：

步骤 1 在用户模式执行 ecall 指令，处理器进入异常服务函数。

步骤 2 在异常服务函数中根据 mcause 值判断是否为用户模式下执行 ecall，如果 mcause[INTERRUPT]为 0 且 mcause[EXCEPTION_CODE]为 8，则继续执行步骤 3，否则执行其他异常或者中断处理操作。

步骤 3 在异常服务函数中将 mstatus[MPP]设置为 3（机器模式）。

步骤 4 如果进入机器模式后需要打开中断响应，将 mstatus[MPIE]配置为 1，否则配置为 0。

步骤 5 配置 mepc 为进入机器模式后将要执行的第一条指令的 PC。

执行 mret 指令，处理器会进入机器模式并自动将 PC 设置成 mepc 的值。

----结束

2.2.8 休眠模式

处理器支持 RISC-V 标准的 WFI 指令，当处理器提交一条 WFI 指令后，会等待未完成的在 WFI 指令前提交的 store 操作执行完，以及数据和外设访问总线没有未完成的请求，然后处理器会进入休眠模式，停止继续执行后续的指令，关闭大部分模块的时钟，直到有中断出现或者调试器发起 halt 请求才会从休眠模式中唤醒，重新打开时钟开始工作。

在被中断唤醒的时候，处理器执行的指令由 mstatus[MIE]的值决定，如果 mstatus[MIE]为 1，则响应中断，进入中断服务函数，如果 mstatus[MIE]为 0，则执行的是 WFI 指令后的指令。

在被调试器 halt 请求唤醒的时候，处理器打开时钟进入调试模式被调试器接管控制，调试器执行 resume 操作让处理器恢复指令执行时，处理器不会再次进入休眠模式，而是执行 WFI 指令后的指令。

处理器处于休眠模式，ITCM 和 DTCM 仍然可以被外部总线访问，访问行为不会唤醒处理器。

2.2.9 物理内存保护

处理器支持 RISC-V 标准的物理内存保护 PMP 功能，物理内存保护功能可以将地址空间划分为多个区域，并设置这些区域的执行、读、写权限，控制非信任软件对敏感内存空间的访问，从而实现内存保护功能。

处理器支持的 PMP 具有以下规格：

- 有 8 个配置项，支持最多 8 个地址区间的划分和权限配置。
- 区间地址划分的粒度为 32Byte，即每个区间起始地址必须是 32Byte 对齐，大小必须是 32Byte 对齐。
- 支持 TOR（Top Of Range）地址模式。

PMP 的具体使用方法请参考《RISC-V 指令集特权标准》。

此外本处理器在标准 PMP 基础上扩展了 memory 属性配置功能，可设置地址区间的 memory 访问属性为 0（device non-bufferable），3（normal non-cacheable bufferable），7（write-back read-allocate），11（write-back write-allocate）和 15（write-back read- and write-allocate）。

ITCM 和 DTCM 区间的默认属性为 3（normal non-cacheable bufferable），如果设置的 PMP 区间为 ITCM 或 DTCM 区间，需要将 PMP 区间的属性设置为 3（normal non-cacheable bufferable）。

AHB 外设总线区间的默认属性为 0（device non-bufferable），如果设置的 PMP 区间为 AHB 外设总线区间，需要将 PMP 区间的属性设置为 0（device non-bufferable）。

2.3 指令介绍

2.3.1 标准扩展指令

本处理器实现了 RV32IMCF 扩展指令集，即 32bit 整型基础指令集、乘除法指令集、压缩指令集和单精度浮点指令集，以及一条比特操作 B 扩展指令集的 CLZ 指令。本章节列举了 RISC-V 的指令 opcode 分类以及本处理器支持的标准扩展指令，具体指令介绍及机器码请参考《RISC-V 指令集非特权标准》相关章节。

表2-6 RISC-V 32bit 指令的 opcode 分类映射表

inst[6:5]	instr[4:2]							
	000	001	010	011	100	101	110	111 (>32b)
00	LOAD	LOAD-FP	custom-0	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	custom-1	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	reserved	*custom-2/RV128*	48b
11	BRANCH	JALR	reserved	JAL	SYSTEM	reserved	*custom-3/RV128*	>=80b

表2-7 RISC-V 16bit 指令的 opcode 分类映射表

inst[1:0]	inst[15:13]								ISA
	000	001	010	011	100	101	110	111	
00	ADDI4SPN	FLD	LW	FLW	reserved	FSD	SW	FSW	RV32
		FLD		LD		FSD		SD	RV64
		LQ		LD		SQ		SD	RV128
01	ADDI4SPN	JAL	LI	LUI / ADDI16SP	MISC-ALU	J	BEQZ	BNEZ	RV32
		ADDIW							RV64
		ADDIW							RV128
10	SLLI	FLDSP	LWSP	FLWSP	J[AL]R /MV	FSDSP	SWSP	FSWSP	RV32

inst[1:0]	inst[15:13]								ISA
	000	001	010	011	100	101	110	111	
		FLDSP		LDSP	/ADD	FSDSP		SDSP	RV64
		LQ		LDSP		SQ		SDSP	RV128
11	>16 bit								

表2-8 支持的 RV32I 基本指令集指令

指令	Opcode 类别	指令类别
LUI	LUI	跳转和条件分支指令
AUIPC	AUIPC	
JAL	JAL	
JALR	JALR	
BEQ	BRANCH	
BNE	BRANCH	
BLT	BRANCH	
BGE	BRANCH	
BLTU	BRANCH	
BGEU	BRANCH	
LB	LOAD	加载和存储指令
LH	LOAD	
LW	LOAD	
LBU	LOAD	
LHU	LOAD	
SB	STORE	
SH	STORE	
SW	STORE	
ADDI	OP-IMM	整型运算指令
SLTI	OP-IMM	
SLTIU	OP-IMM	
XORI	OP-IMM	

指令	Opcode 类别	指令类别
ORI	OP-IMM	
ANDI	OP-IMM	
SLLI	OP-IMM	
SRLI	OP-IMM	
SRAI	OP-IMM	
ADD	OP	
SUB	OP	
SLL	OP	
SLT	OP	
SLTU	OP	
XOR	OP	
SRL	OP	
SRA	OP	
OR	OP	
AND	OP	
CSRRW	SYSTEM	CSR 读写指令
CSRRS	SYSTEM	
CSRRC	SYSTEM	
CSRRWI	SYSTEM	
CSRRSI	SYSTEM	
CSRRCI	SYSTEM	
ECALL	SYSTEM	环境调用指令
EBREAK	SYSTEM	环境断点指令
MRET	SYSTEM	机器模式 trap 返回指令
WFI	SYSTEM	等待中断指令
FENCE	MISC-MEM	memory 访问保序指令
FENCE.I	MISC-MEM	指令读取保序指令

表2-9 支持的 RV32M 扩展指令集指令

指令	Opcode 类别	指令类别
MUL	OP	乘法
MULH	OP	
MULHSU	OP	
MULHU	OP	
DIV	OP	除法和求余
DIVU	OP	
REM	OP	
REMU	OP	

表2-10 支持的 RV32C 扩展指令集指令

指令	Opcode 类别	指令类别
C.LW	C0	加载和存储指令
C.LWSP	C2	
C.SW	C0	
C.SWSP	C2	
C.J	C1	跳转和条件分支指令
C.JAL	C1	
C.JR	C2	
C.JALR	C2	
C.BEQZ	C1	
C.BNEZ	C1	
C.LI	C1	整型计算指令
C.LUI	C1	
C.ADDI16SP	C1	
C.ADDI	C1	
C.ADDI4SPN	C0	
C.SLLI	C2	
C.SRLI	C1	

指令	Opcode 类别	指令类别
C.SRAI	C1	
C.ANDI	C1	
C.AND	C1	
C.MV	C2	
C.SUB	C1	
C.XOR	C1	
C.OR	C1	
C.ADD	C2	
C.NOP	C1	
C.BREAK	C2	

表2-11 支持的 RV32F 单精度浮点扩展指令集指令

指令	Opcode 类别	指令类别
FLW	LOAD-FP	加载和存储指令
C.FLW	C0	
FSW	STORE-FP	
C.FSW	C0	
C.FLWSP	C2	
C.FSWSP	C2	
FMADD.S	FMADD	运算指令
FMSUB.S	FMSUB	
FNMADD.S	FNMADD	
FNMSUB.S	FNMSUB	
FADD.S	OP-FP	
FSUB.S	OP-FP	
FMUL.S	OP-FP	
FDIV.S	OP-FP	
FSQRT.S	OP-FP	
FMIN.S	OP-FP	

指令	Opcode 类别	指令类别
FMAX.S	OP-FP	
FSGNJ.S	OP-FP	转换和转移指令
FSGNJN.S	OP-FP	
FSGNJX.S	OP-FP	
FCVT.W.S	OP-FP	
FCVT.WU.S	OP-FP	
FCVT.S.W	OP-FP	
FCVT.S.WU	OP-FP	
FMV.X.W	OP-FP	
FMV.W.X	OP-FP	
FEQ.S	OP-FP	比较指令
FLT.S	OP-FP	
FLE.S	OP-FP	
FCLASS.S	OP-FP	分类指令

表2-12 支持的 RV32B 扩展指令集指令（仅 CLZ）

指令	Opcode 类别	指令类别
CLZ	I-type	基本 bit 操作指令

2.3.2 自定义指令

2.3.2.1 LDMIA

表2-13 LDMIA 指令编码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
e	gpr_mask[15:5]											rs1				000			gpr_mask[4:0]			0001011									

LDMIA 是一个用于实现加载连续递增地址数据到多个整型通用寄存器的指令。指令编码中的 rs1 位域指定的通用寄存器中的值为加载起始地址。指令编码中的 e 位域和 gpr_mask 位域共同决定了加载数据回写的通用寄存器组，具体映射关系参照表 2-14，其中 e 位域决定操作的寄存器组，gpr_mask 则决定该寄存器组内每一个通用寄存器是否有效。

LDMIA 指令从 gpr_mask 最高 bit 开始找到第一个为 1 的位置对应的寄存器，加载基地址的数据到该寄存器，然后访问地址加 4，再从 gpr_mask 中找到下一个为 1 的位置对应的寄存器，加载数据到该寄存器，重复这个过程直到 gpr_mask 中指示的所有寄存器都加载完毕，指令执行过程中以及完成后 rs1 中的值不变。

LDMIA 加载数据到寄存器时，会加载 memory 中最小地址的数据到最大编号的寄存器中，而 memory 中最大地址的数据则会被加载到最小编号的寄存器中。

LDMIA 指令只支持 word 对齐的基地址，如果 rs1 中记录的基地址不是 4Byte 对齐，LDMIA 指令会上报读数据地址非对齐异常。

如果 rs1 寄存器为其中一个加载的目的寄存器，处理器会上报非法指令异常。

表2-14 LDMIA 和 STMIA 指令的 gpr_mask 与 callee/caller 寄存器编号对应表

e = 0 = callee																
gpr_mask 各位域	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
整型通用寄存器	X1	X2	X8	X9	X10	X11	X18	X19	X20	X21	X22	X23	X24	X25	X26	X27
e = 1 = caller																
gpr_mask 各位域	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
整型通用寄存器	X1	X5	X6	X7	X10	X11	X12	X13	X14	X15	X16	X17	X28	X29	X30	X31

示例：

```
rs1 = 0x1000
e = 1’b1 = caller
gpr_mask = 16xA143
```

此 LDMIA 指令会分别从 0x1000，0x1004，0x1008，0x100C，0x1010，0x1014 的 memory 地址加载数据回写到 caller 通用寄存器 X31，X29，X14，X12，X5，X1 中。

汇编语法：

```
LDMIA {register_list}, (<rs1>)
```

汇编示例：

```
LDMIA {ra, t0-t2, a0-a7, t3-t6}, (sp)
LDMIA {sp, s0, s1-s11}, (t0)
LDMIA {X1, X5-X7, X10-X17, X28-X31}, (X2)
LDMIA {X2, X8-X9, X18-X27}, (X5)
```

2.3.2.2 STMIA

表2-15 STMIA 指令编码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
e	gpr_mask[15:5]											rs1					001			gpr_mask[4:0]					0001011						

STMIA 与 LDMIA 指令各位域的使用方式一致，具体 e 位域和 gpr_mask 位域与实际操作寄存器之间的映射关系参考表 2-14。与 LDMIA 不同，STMIA 指令将指定寄存器里面的值存储到 memory 中。STMIA 指令会将最大编号的有效寄存器写到基地址的 memory，而将最小编号的有效寄存器写到最大的 memory 地址。

STMIA 只支持 Word 对齐的基地址，如果 rs1 中记录的基地址不是 4Byte 对齐，LDMIA 指令会上报写数据地址非对齐异常。

示例：

```
rs1 = 0x1000
e = 1’b0 = callee
gpr_mask = 0x13D4
```

此 STMIA 指令会将 callee 通用寄存器 X24，X21，X20，X19，X18，X10，X8 的 32bit 数据分别写到 memory 的 0x1000，0x1004，0x1008，0x100C，0x1010，0x1014，0x1018 地址中。

汇编语法：

```
STMIA {register_list}, (<rs1>)
```

汇编示例：

```
STMIA {ra, t0-t2, a0-a7, t3-t6}, (sp)
STMIA {sp, s0, s1-s11}, (t0)
STMIA {X1, X5-X7, X10-X17, X28-X31}, (X2)
STMIA {X2, X8-X9, X18-X17}, (X5)
```

2.3.2.3 L.LI

表2-16 L.LI 指令编码

47	46	45	44	43	44	43	..	...	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[31:0]																	0000				rd					0011111						

L.LI 是一条 48bit 长度的指令，功能是将指令编码中的 32bit imm 位域写入到 rd 位域指定的寄存器中。

汇编语法：

```
li <rd>, <imm>
```

汇编示例：

```
li t1, 0x80000024
li s1, 0x70000036
li x6, 0x80000024
li x9, 0x70000036
```

2.3.2.4 C.LBU

表2-17 C.LBU 指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001			uimm[0|4:3]			rs1′			uimm[2:1]		rd′			00	

C.LBU 指令从 memory 指定地址加载 8bit 数据，然后进行 0 扩展到 32bit，再写入到 rd′ 位域指定的寄存器中。加载的 memory 地址以 rs1′ 位域指定的寄存器值作为基地址，再加上 5bit uimm 位域 0 扩展后的偏移值。与部分标准 C 扩展压缩指令一样，C.LBU 指令使用 3bit 的 rs1′ 和 rd′ 位域来指定 x8 ~ x15 这 8 个寄存器，具体映射关系如表 2-18 所示。

表2-18 C.LBU 和 C.SB 指令 rs1′ rs2′ 与 rd′ 位域指定的寄存器编号

C.LBU/C.SB rs1′ rs2′ rd′ 编码	3b000	3b001	3b010	3b011	3b100	3b101	3b110	3b111
整型通用寄存器	x8	x9	x10	x11	x12	x13	x14	x15

汇编语法：

```
lbu <rd'>, <uimm>(<rs1'>)
```

汇编示例：

```
lbu a1, 6(a2)
lbu x11, 6(x12)
```

2.3.2.5 C.SB

表2-19 自定义 C.SB 指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101			uimm[0|4:3]			rs1′			uimm[2:1]		rs2′			00	

C.SB 与 C.LBU 指令对应，是将 rs2′ 寄存器中的低 8bit 写入到指定 memory 地址中。写入的 memory 地址以 rs1′ 位域指定的寄存器值作为基地址，再加上 5bit uimm 位域 0 扩展后的偏移值。与部分标准 C 扩展压缩指令一样，C.SB 指令使用 3bit 的 rs1′ 和 rs2′ 域来指定 x8 ~ x15 这 8 个寄存器。

汇编语法：

```
sb <rs2'>, <uimm>(<rs1'>)
```

汇编示例：

```
sb a1, 6(a2)
sb x11, 6(x12)
```

2.3.2.6 C.PUSH/C.POP/C.POPRET

表2-20 C.PUSH/C.POP/C.POPRET 16-bit 指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	sp16imm					rcount				opc		0	0

表2-21 C.PUSH/C.POP/C.POPRET 指令的 opc 位域说明

Opc 位域	标识	描述
00	C.POP	出栈寄存器值，调整栈指针释放栈空间。
01	C.POPRET	出栈寄存器值，调整栈指针释放栈空间并执行函数返回。
10	C.PUSH	调整栈指针开辟栈空间，入栈寄存器值。
11	reserved	触发非法指令异常。

表2-22 C.PUSH/C.POP/C.POPRET 的 rcount 位域说明

Rcount 位域	出入栈的寄存器的 ABI 名字	出入栈的寄存器的数字型名字	需要的 16Byte blocks	SP 栈指针调整值*	汇编语法中支持的 SP 栈指针调整值*
0	reserved	reserved	reserved	reserved	reserved
1	ra	x1	1	± 16(blocks+sp16imm)	±(16 to 512) in steps of 16
2	ra, s0	x1, x8			
3	ra, s0-s1	x1, x8-x9			
4	ra, s0-s2	x1, x8-x9, x18			
5	ra, s0-s3	x1, x8-x9, x18-x19	2	±16(blocks +sp16imm)	±(32 to 528) in steps of 16
6	ra, s0-s4	x1, x8-x9, x18-x20			
7	ra, s0-s5	x1, x8-x9, x18-x21			
8	ra, s0-s6	x1, x8-x9, x18-x22			
9	ra, s0-s7	x1, x8-x9, x18-x23	3	±16(blocks +sp16imm)	±(48 to 544) in steps of 16
10	ra, s0-s8	x1, x8-x9, x18-x24			
11	ra, s0-s9	x1, x8-x9, x18-x25			
12	ra, s0-s10	x1, x8-x9, x18-x26			
13	ra, s0-s11	x1, x8-x9, x18-x27	4	±16(blocks +sp16imm)	±(64 to 560) in steps of 16
14	ra, s0-s11, a0	x1, x8-x9, x18-x27, x10			
15	ra, s0-s11, a0-a1	x1, x8-x9, x18-x27, x10-x11			

*对于 SP 栈指针调整值，PUSH 指令为正，POP/POPRET 指令为负。

C.PUSH/C.POP/C.POPRET 指令的编码里面没有显式指定使用的寄存器，指令操作的栈空间地址隐含地在 sp（x2）寄存器指定，而出入栈操作的具体寄存器，则是通过 rcount 域进行编码确定，具体 rcount 值和出入栈寄存器的映射关系如表 2-22 所示。

指令中的 sp16imm 位域是一个无符号 5bit 立即数，用来定义指令执行时栈指针需要额外调整的 16Byte 地址块数量。这是为了函数能使用指令一次性开辟足够的栈空间用于临时变量存储，而不需要另外执行栈空间调整操作。

所以最终栈指针调整的大小跟入栈寄存器需要占用的 16Byte 空间块数量以及 sp16imm 域的值有关系，如表 2-22 倒数第二列所示。

1. C.PUSH 指令

   PUSH 指令把 rcount 选择的寄存器写入到 memory 中，完成入栈操作，写入的地址为紧贴当前栈指针往更小地址延伸的空间。寄存器在 memory 中的排序与表 2-22 第 2、3 列描述的一致，ra 寄存器写入到最大的地址位置，也就是 sp-4，后续的寄存器在 memory 中顺序紧凑排列，每个寄存器占 4Byte 空间，且往小地址方向发展，最后一个寄存器写入到 sp-4*N 的地址，其中 N 是入栈的寄存器个数，也就是 rcount 值。

   在所有寄存器写 memory 的操作完成后，栈指针寄存器 sp 会减去 16*(((N+3)/4)+sp16imm)。其中 sp16imm 可以使指令支持最多多开辟 496Byte 的栈空间用来存储其他变量，而表达式(N+3)/4 表示入栈 N 个寄存器需要多少个 16Byte 空间块。

2. C.POP 指令

   POP 指令是将 memory 的数据加载到 rcount 指定的寄存器中，完成出栈操作。寄存器在 memory 中的排序与表 2-22 第 2、3 列描述的一致，ra 寄存器从最大的地址位置读取，也就是 sp+16*(((N+3)/4)+sp16imm)-4，而最后一个寄存器读取的 memory 地址为 sp+4*(4*sp16imm+3-((N+3) mod 4))，其中 N 是入栈的寄存器个数，也就是 rcount 值。

   在所有数据加载完成后，栈指针寄存器 sp 会加上 16*(((N+3)/4)+sp16imm)。

3. C.POPRET 指令

   C.POPRET 指令组合了 C.POP 指令和 RET 伪指令的功能，在执行完 C.POP 操作后，以 C.POP 操作中加载的新 ra 寄存器值作为跳转目标执行 RET 指令，实现函数返回跳转。

4. 异常情况

如果 sp(x2)寄存器的值不是 16Byte 对齐，执行 PUSH 指令会触发写数据地址非对齐异常。

如果 sp(x2)寄存器的值不是 16Byte 对齐，执行 POP/POPRET 指令会触发读数据地址非对齐异常。

如果 rcount 为 0 或者 opc 为 3，执行 PUSH/POP/POPRET 指令会触发非法指令异常。

5. 汇编语法

```
PUSH {register_list}, -size
POP {register_list}, size
POPRET {register_list}, size
```

PUSH/POP/POPRET 指令的汇编语法由指令名，寄存器列表，以及栈调整大小的值组成，其中栈调整大小是以 Byte 为单位，请参考表 2-22 的最后一列。对于 PUSH 指令，栈调整大小为负数，而 POP、POPRET 指令的栈调整大小为正数。寄存器列表只能包含指令支持的序列组合，请参考表 2-22 第 2、3 列，寄存器列表的描写可以任意顺序，且支持寄存器的 ABI 名字或者 X 加数字的命名方式。

合法的栈调整大小值需要满足以下要求:

- PUSH 指令为负值，POP/POPRET 指令为正值。
- 必须是 16 的倍数。
- 必须大于等于出入栈寄存器数量的 4 倍。
- 绝对值必须符合公式 16*(((N+3)/4)+M)，其中 N 是寄存器数量，而 M 则是 sp16imm 的值，且 0<=M<32。

6. Push/Pop 寄存器操作示例

RISC-V 定义的栈是 16Byte 对齐且往小地址生长，所以栈指针总是以 16Byte 为单位进行调整，所以如果出入栈的寄存器数量不是 4 的倍数，计算占用的空间时会使用寄存器个数往上对齐到 4 的倍数。下列示例中约定了 sp16imm 只为 0。

图2-1 PUSH 指令操作 1 到 5 个寄存器的示例

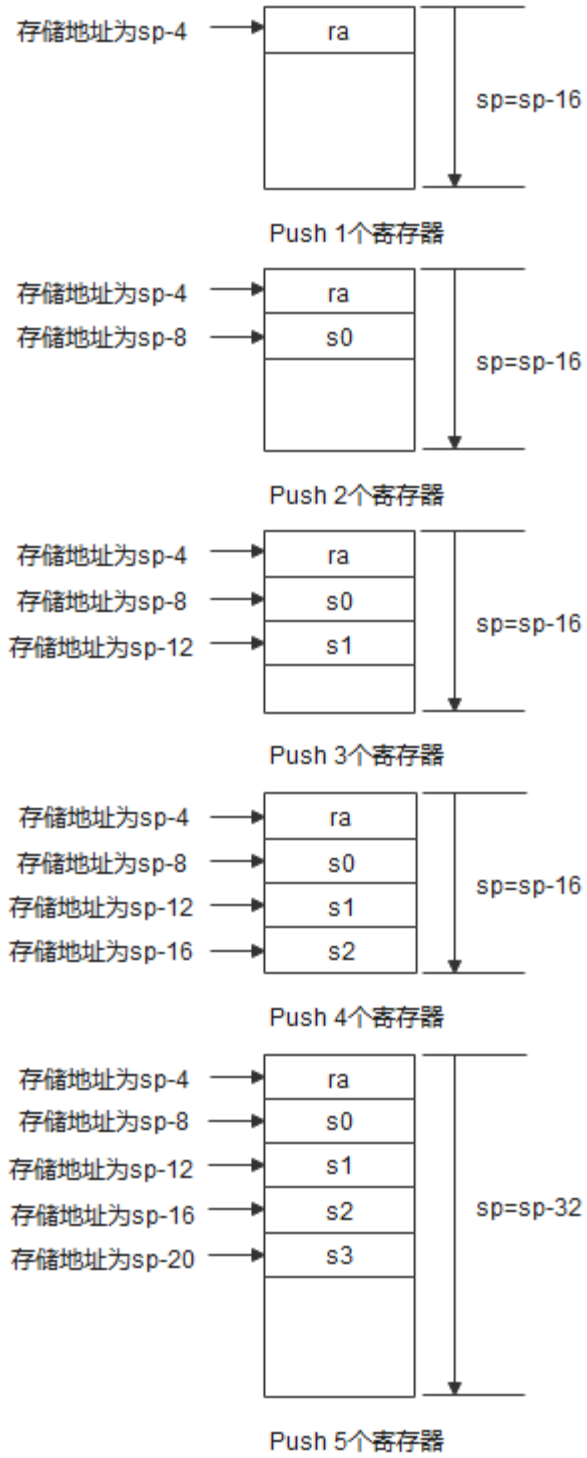
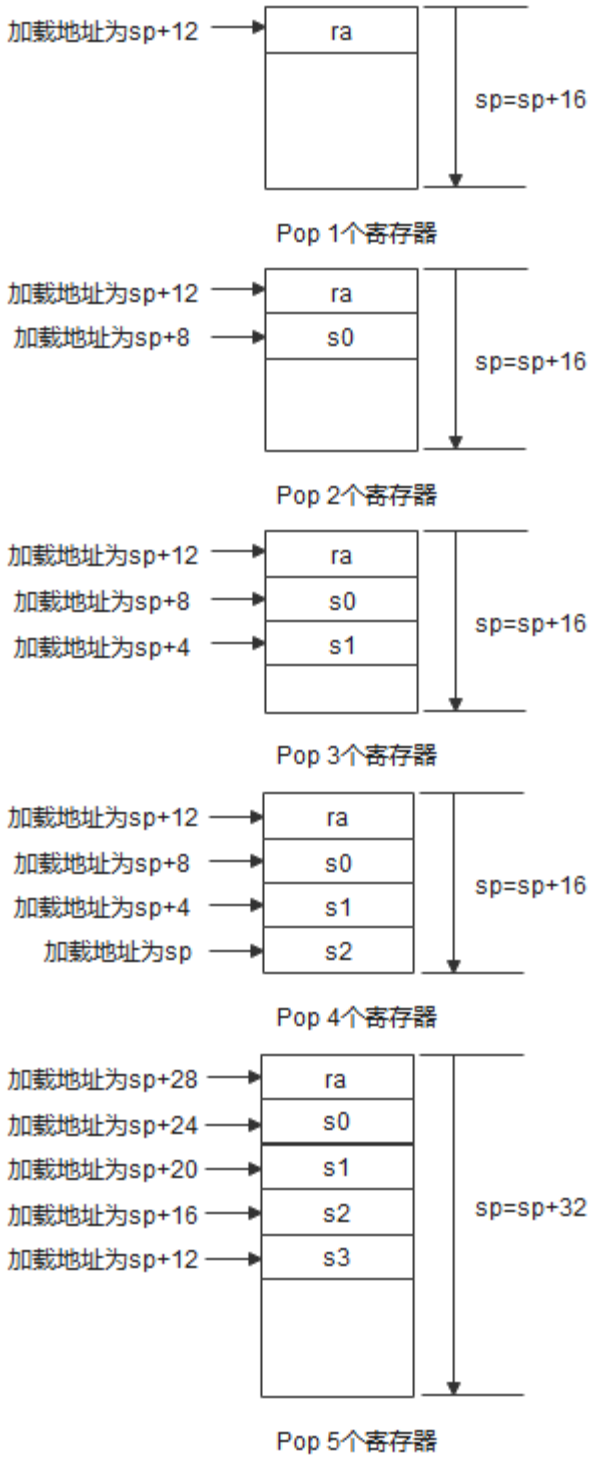


图2-2 POP 指令操作 1 到 5 个寄存器的示例



7. 汇编示例

- C.PUSH 指令：

```
push  {ra, s0-s4}, -64
```

各编码的位域值：

```
rcount=6,  sp16imm=2,  opc=2
```

等价的指令序列：

```
sw  s4, -24(sp)
sw  s3, -20(sp)
sw  s2, -16(sp)
sw  s1, -12(sp)
sw  s0, -8 (sp)
sw  ra, -4 (sp)
```

addi sp, sp, -64 ; //-64 = -(32+16*2) 请参考表 2-22。

- C.POP 指令：

```
pop   {x1, x8-x9, x18-x25}, 256
```

各编码的位域值：

```
rcount=11,  sp16imm=13,  opc=0
```

等价的指令序列：

```
lw  x25, 212(sp)
lw  x24, 216(sp)
lw  x23, 220(sp)
lw  x22, 224(sp)
lw  x21, 228(sp)
lw  x20, 232(sp)
lw  x19, 236(sp)
lw  x18, 240(sp)
lw  x9,  244(sp)
lw  x8,  248(sp)
lw  x1,  252(sp)
```

addi sp, sp, 256; //256 = (48+16*13) 请参考表 2-22。

- C.POPRET 指令：

```
popret   {x1, x8-x9, x18-x19}, 32
```

各编码的位域值：

```
rcount=5,  sp16imm=0,  opc=1
```

等价的指令序列：

```
lw  x19, 12(sp)
lw  x18, 16(sp)
lw  x9,  20(sp)
lw  x8,  24(sp)
lw  x1,  28(sp)
```

addi sp, sp, 32; //32 = (32+16*0) 请参考表 2-22。

ret; //expands to: jalr x0, x1, 0

2.3.2.7 BEQI/BGEI/BGEUI/BLTI/BLTUI/BNEI

表2-23 立即数比较分支指令编码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cmpimm[7:0]								offset[9:6]				rs1					opc			offset[5:1]				0111011							

表2-24 立即数比较分支指令中 opc 位域说明

Opc 位域	标识	语义 [a]
000	BEQI	如果 (rs1 == sext(cmpimm)) 则跳转到 PC + sext(offset)
101	BGEI	如果 (rs1>=(signed) sext(cmpimm)) 则跳转到 PC + sext(offset)
111	BGEUI	如果 (rs1>=(unsigned) zext(cmpimm)) 则跳转到 PC + sext(offset)
100	BLTI	如果 (rs1<(signed) sext(cmpimm)) 则跳转到 PC + sext(offset)
110	BLTUI	如果 (rs1<(unsigned) zext(cmpimm)) 则跳转到 PC + sext(offset)
001	BNEI	如果 (rs1 != sext(cmpimm)) 则跳转到 PC + sext(offset)
010	reserved	触发非法指令异常
011	reserved	触发非法指令异常

a：sext 为符号位扩展，zext 为 0 扩展，signed 表示为有符号比较，unsigned 表示为无符号比较。

这类指令是将下面的指令序列：

```
c.li a2,#2
beq a4, a2, <offset>
```

组合到一条指令里面：

```
beqi a4, #2, <offset>
```

从而提升代码密度和性能。主要用于固定循环次数的循环结束判断。

此类指令最大的 PC 跳转范围是 -512 ~ +510，且要求跳转偏移是 2 对齐。

对于有符号比较的指令，比较的立即数值取值范围为-128 ~ +127。

对于无符号比较的指令，比较的立即数值取值范围为 0 ~ 255。

汇编语法：

```
beqi <rs1>, <cmpimm>, <offset>
bnei <rs1>, <cmpimm>, <offset>
bgei <rs1>, <cmpimm>, <offset>
bgeui <rs1>, <cmpimm>, <offset>
blti <rs1>, <cmpimm>, <offset>
```

bltui<rs1>, <cmpimm>, <offset>汇编示例：

```
beqi s1, #1, <label>
bnei s1, #2, <label>
blti x1, #1, <label>
bgei x2, #2, <label>
```

2.3.2.8 ADDSHF/ SUBSHF/ORSHF/XORSHF/ANDSHF

表2-25 移位运算指令的编码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
shtype		shamt					rs2					rs1					opc			rd					0011011						

表2-26 移位运算指令的 opc 位域说明

Opc 位域	标识	语义
000	ADDSHF	rd = rs1 + preshift(rs2, shtype)
001	SUBSHF	rd = rs1 - preshift(rs2, shtype)
010	ORSHF	rd = rs1 | preshift(rs2, shtype)
011	XORSHF	rd = rs1 ^ preshift(rs2, shtype)
100	ANDSHF	rd = rs1 & preshift(rs2, shtype)
101	reserved	触发非法指令异常
110	reserved	触发非法指令异常
111	reserved	触发非法指令异常

表2-27 移位运算指令的 shtype 位域编码说明

Shtype 位域	标识	语义	描述
00	SLL	rs2 << shamt	左移。
01	SRL	rs2 >> shamt	逻辑右移。
10	SRA	sra(rs2, shamt)	算术右移。
11	ROR	ror(rs2, shamt)	循环右移。

这类指令将一个移位指令和一个运算指令的序列功能组合到一个指令中实现。

须知

shamt=0 会触发非法指令异常。

汇编语法：

```
add       <rd>, <rs1>, <rs2>, <preshift> #shamt
sub       <rd>, <rs1>, <rs2>, <preshift> #shamt
or        <rd>, <rs1>, <rs2>, <preshift> #shamt
xor       <rd>, <rs1>, <rs2>, <preshift> #shamt
and       <rd>, <rs1>, <rs2>, <preshift> #shamt
```

汇编示例：

```
add       t1, s2, s3, sll #1 // t1 = s2 + (s3 << 1)
sub       t1, s2, s3, ror #30// t1 = s2 - ror(s3, 30)
or        t1, s2, s3, srl #30// t1 = s2 | (s3 >> 30)
xor       t1, s2, s3, srl #30// t1 = s2 ^ sra(s3, 30)
and       t1, s2, s3, sll #30// t1 = s2 & (s3 << 30)
add       x1, x2, x3, sll #1 // x1 = x2 + (x3 << 1)
sub       x1, x2, x3, ror #30// x1 = x2 - ror(x3, 30)
or        x1, x2, x3, srl #30// x1 = x2 | (x3 >> 30)
xor       x1, x2, x3, sra #30// x1 = x2 ^ sra(x3, 30)
and       x1, x2, x3, sll #30// x1 = x2 & (x3 << 30)
```

2.3.2.9 C.UXTB/C.UXTH

表2-28 C.UXTB, C.UXTH 的编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	rs1′ /rd′			opc		000			0	1

表2-29 C.UXTB, C.UXTH 的 opc 位域说明

Opc 位域	标识	语义
00	C.UXTB	rd′ = zero_ext(rs1′ [7:0])
01	C.UXTH	rd′ = zero_ext(rs1′ [15:0])
10	reserved	触发非法指令异常
11	reserved	触发非法指令异常

- C.UXTB 和 C.UXTH 分别用于提取寄存器中的低 8bit 和低 16bit 数据，其中 rs1′ 和 rd′ 位域只能使用 x8 ~ x15 寄存器。
- C.UXTB 只提取 rs1′ 寄存器中的低 8bit，然后 0 扩展到 32bit，再写入到 rd′ 寄存器中。如 c.uxtb xn 指令可以替换 32bit 指令 andi xn,xn,255。
- C.UXTH 只提取 rs1′ 寄存器中的低 16bit，然后 0 扩展到 32bit，再写入到 rd′ 寄存器中。如 c.uxth xn 指令可以替换指令序列 c.slli xn, #16;c.srli x8,#16。

汇编语法：

```
uxth <rd′/rs1′> // rd′ = zero_ext(rs1′[15:0])
uxtb <rd′/rs1′> // rd′ = zero_ext(rs1′[7:0])
```

汇编示例：

```
uxth s0 // s0 = zero_ext(t1[15:0])
uxtb s1 // s1 = zero_ext(t2[7:0])
uxth x8 // x8 = zero_ext(x8[15:0])
uxtb x9 // x9 = zero_ext(x9[7:0])
```

2.3.2.10 C.SH/C.LHU

表2-30 C.SH 和 C.LHU 的指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-
1	0	1	uimm[5:3]			rs1′			uimm[2:1]		rs2′			1	0	c.sh
0	0	1	uimm[5:3]			rs1′			uimm[2:1]		rd′			1	0	c.lhu

- C.LHU，C.SH 是 Half-word 数据加载存储的 16bit 压缩指令版本。

- 指令的 uimm 位域的 bit0 是隐含为 0 的，即指令的偏移值只能是 2 对齐，且偏移范围是 0 ~ 62 Byte。
- 指令使用 rs1’ rs2’ 和 rd’，所以只能使用 x8 ~ x15 寄存器。

表2-31 C.SH 和 C.LHU 的指令语义

指令	语义	描述
c.lhu	x[8+rd’] = zero_extend(M[x[8+rs1’] + uimm[5:0]])	从 memory 加载无符号半字数据到寄存器。
c.sh	M[x[8+rs1’] + uimm[5:0]] = x[8+rs2’]	将寄存器里面的半字数据存储到 memory 中。

汇编语法：

```
lhu  <rd’>, #<uimm>(<rs1’>)
sh   <rs2’>, #<uimm>(<rs1’>)
```

汇编示例：

```
lhu  s0, #26(s1)
sh   s0, #26(s1)
lhu  x8, #26(x9)
sh   x8, #26(x9)
```

2.3.2.11 MULIADD

表2-32 MULIADD 指令编码(X=uimm[0])

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uimm [7:1]							rs2					rs1					X	opc		rd					1011011						

表2-33 MULIADD 指令的 opc 位域

Opc 位域	标识	语义
00	reserved	触发非法指令异常
01	MULIADD	rd = rs1 + (rs2 * zero_ext(uimm))
10	reserved	触发非法指令异常

Opc 位域	标识	语义
11	reserved	触发非法指令异常

MULIADD 指令完成两个 32bit 寄存器的相乘，乘积再加上一个立即数的运算。其中立即数 uimm 的取指范围为 0 ~ 255。

汇编语法：

```
muliadd   <rd>, <rs1>, <rs2>, #<uimm>
```

汇编示例：

```
muliadd   a0, a2, a3, #7; // a0 = a2 + (a3 * 7)
muliadd   x4, x2, x3, #7; // x4 = x2 + (x3 * 7)
```

2.3.2.12 J16M/JAL16M

表2-34 JAL16M/J16M 指令的编码

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[20|10:1|11|19:12]																				imm [24:21]				opc	1111011						

表2-35 JAL16M/J16M 指令的 opc 位域说明

Opc 位域	标识	描述
0	JAL16M	跳转到当前 PC ±16MB 范围的地址，并记录下一条指令的 PC 到 ra(x1)作为返回地址。
1	J16M	跳转到当前 PC ±16MB 范围的地址。

RV32I 中的标准 JAL(R)指令只能支持 ±1MB 范围的跳转，JAL16M 和 J16M 则能支持±16MB 范围的跳转。JAL16M 指令隐含地使用 ra(x1)寄存器作为目的寄存器，而 J16M 指令隐含地使用 zero(x0)寄存器作为目的寄存器。

汇编语法：

汇编语法与标准的 JAL/J 指令一致，工具链在识别到跳转范围超过 ±1MB 且链接寄存器为 ra 时会自动使用 JAL16M 和 J16M 指令。

```
jal  ra,  #<imm>;   // jump to PC + sext(imm), link to ra
j         #<imm>;   // jump to PC + sext(imm)
```

汇编示例：

```
jal ra, #0x80 0000; // jump to PC+0x800000 (out of +1MB range), link to ra
j       #0xF0 0000; // jump to PC+0xF00000 (out of +1MB range)
jal x1, #0x80 0000; // jump to PC+0x800000 (out of +1MB range), link to x1
j       #0xF0 0000; // jump to PC+0xF00000 (out of +1MB range)
```

2.3.3 指令执行周期

- 整型除法和求余指令为不定长，根据操作数不同执行周期在 4~19 拍之间。
- 浮点除法和开方根指令为不定长，根据操作数不同执行周期在 4~16 拍之间。
- 其他运算类指令均为 1 拍完成。
- 加载和存储指令执行周期受系统总线访问延时影响，LDMIA、STMIA、C.PUSH、C.POP 和 C.POPRET 指令的执行周期还跟操作的寄存器数量相关。
- CSR 和系统指令，指令本身是 1 拍完成，但由于涉及到处理器状态变化，处理器会清空流水线，导致后续指令无法流水，造成性能损失。

2.4 异常与中断

异常与中断是指一些需要处理器暂停当前主线任务跳转到特定服务函数进行专项处理的事件，一些文档中将中断归纳为异常中的一种，本文档异常指处理器执行指令过程中出现的一些错误或是执行了一些特定的系统指令，中断则是指从外设连接到处理器的用于表征特定硬件事件出现的信号。

RISC-V 定义的跟异常中断有关的标准 CSR 有 mstatus、mcause、mepc、mtvec、mtval、mscratch、mie 和 mip，可以参考《RISC-V 指令集特权标准》查看详细的 CSR 功能定义。处理器支持通过 mtvec 设置直接模式和向量模式两种中断响应模式。

2.4.1 异常

2.4.1.1 异常类型

RISC-V 标准定义了如表 2-36 所示的一些异常类型，当出现这些异常事件的时候，处理器暂停原来的程序流执行并响应异常，跳转到特定的服务函数入口，软件通过读取 mcause CSR 即可获取具体的 exception code 进行异常分析。

需要注意的是大部分异常事件为同步异常，只有数据存储（store）操作遇到总线错误的异常事件是异步异常。对于同步异常，处理器保证响应异常的时候，异常指令前的指令都已经完成提交，而异常指令后的指令不会造成任何影响，所以能提供精确的现场信息用于问题定位。对于异步异常，处理器不保证异常响应的时间，有可能处理器响应异常的时候，导致问题的指令在很早之前已完成提交，包括其后面的指令也可能已经完成提交，所以无法提供精确的现场信息用于问题定位。

表2-36 mcause 中不同异常的编码表

INTERRUPT 位域	Exception Code 位域编码值	描述
0	0	指令地址非对齐。
0	1	指令访问错误。
0	2	非法指令。
0	3	断点 breakpoint。
0	4	读数据地址非对齐。
0	5	读数据访问错误。
0	6	写数据地址非对齐。
0	7	写数据访问错误。
0	8	用户模式下的环境调用 ECALL。
0	9	保留。
0	10	保留。
0	11	机器模式下的环境调用 ECALL。
0	≥ 12	保留。

由于 RISC-V 定义的异常粒度比较粗，为了更好地定位异常原因，本处理器定义了一个 CSR 寄存器用于记录更加细致的异常原因，其编码如表 2-37 所示，在处理器响应中断时会将对应的 ccause 编码值记录到自定义 CSR cxcptsc 中，根据 mcause 和 ccause 能快速明确异常上报的原因。

表2-37 Ccause 中不同异常的编码表

CCAUSE 编码值	描述
0	无效的异常原因编码。
1	请求地址不属于处理器可访问的地址区间。
2	保留。
3	AHB 外设总线返回总线错误。
4	请求跨 PMP 区间。
5	处理器内部的系统寄存器访问错误。
6	没有匹配到 PMP 区间。
7	PMP 查询错误，没有 PMP 权限。
8	保留。
9	CSR 访问错误。
10	LDMIA/STMIA/PUSH/POP/POPRET 指令错误。
11	保留。

2.4.1.2 异常响应流程

当异常事件出现，处理器响应这些异常事件的时候硬件会自动执行以下操作：

步骤 1 将响应异常前的特权模式记录到 mstatus[MPP]中。

步骤 2 将处理器特权模式设置为机器模式。

步骤 3 将 mstatus[MIE]值赋值到 mstatus[MPIE]中。

步骤 4 将 mstatus[MIE]值设置为 0 屏蔽中断响应。

步骤 5 将出现异常的指令的 PC 值记录在 mepc 中。

步骤 6 将异常编号记录到 mcause 中。

步骤 7 将自定义异常编号记录到 cxcptsc[CCAUSE]中。

步骤 8 将指定的异常信息记录到 mtval 中。

步骤 9 将 PC 值设置为 mtvec 记录的异常处理服务函数基地址，进行取指和指令执行。

----结束

在完成异常处理函数后，软件需要执行 mret 指令返回主线任务中，在执行 mret 指令时，处理器硬件会自动执行以下操作：

步骤 1 将处理器特权模式设置成 mstatus[MPP]中记录的值。

步骤 2 将 mstatus[MPP]设置为 0。

步骤 3 将 mstatus[MIE]的值设置成 mstatus[MPIE]的值。

步骤 4 将 mstatus[MPIE]设置为 1。

步骤 5 将处理器 PC 值设置为 mepc 中记录的值，进行取指和指令执行。

----结束

2.4.2 中断

2.4.2.1 中断类型

为了简化中断机制，加快中断响应，处理器定义了非标准自定义本地中断，本地中断使用 26 号及以后的中断编号，完整的中断编号如表 2-38 所示，处理器响应本地中断时，对应的编号会直接记录在 mcause 中。

处理器支持直接和向量两种中断响应模式，模式选择由 mtvec[MODE]决定。mtvec[MODE]为 0 表示直接模式（direct mode），所有中断都跳转到 mtvec 记录的异常处理向量基地址。mtvec[MODE]为 1 表示向量模式（vector mode），处理器响应中断时会跳转到 mtvec[BASE]记录的异常处理向量基地址偏移中断编号乘 4 的地址（mtvec[BASE]+4*int_id）。

表2-38 mcause 中的中断编码表

INTERRUPT 位域	Exception Code 位域编码	描述
1	0~25	保留，未使用。
1	26~121	非标准本地中断，具体中断映射请参考表 1-2。
1	≥122	保留，未使用。

2.4.2.2 中断响应流程

当中断出现，处理器响应这些中断事件的时候硬件会自动执行以下操作：

步骤 1 将响应异常或中断前的特权模式记录到 mstatus[MPP]中。

步骤 2 将处理器特权模式设置为机器模式。

步骤 3 将 mstatus[MIE]值赋值到 mstatus[MPIE]中。

步骤 4 将 mstatus[MIE]值设置为 0 屏蔽中断响应。

步骤 5 将被中断的指令的 PC 值记录在 mepc 中。

步骤 6 将中断编号记录到 mcause 中。

步骤 7 将该被响应中断的优先级记录到 cipri 中。

步骤 8 清除被响应中断的挂起信号。

步骤 9 根据 mtvec[MODE]中设置的直接或者向量模式进行 PC 跳转。

----结束

在完成中断处理函数后，软件需要执行 mret 指令返回主线任务中，在执行 mret 指令时，处理器硬件会自动执行以下操作：

步骤 1 将处理器特权模式设置成 mstatus[MPP]中记录的值。

步骤 2 将 mstatus[MPP]设置为 0。

步骤 3 将 mstatus[MIE]的值设置成 mstatus[MPIE]的值。

步骤 4 将 mstatus[MPIE]设置为 1。

步骤 5 将处理器 PC 设置为 mepc 中记录的值。

----结束

2.4.2.3 中断编程模型

处理器定义的非标准自定义本地中断支持使能配置，优先级配置和优先级阈值配置，这些配置通过使用标准的 mie 和 mip 寄存器加上自定义 CSR 实现，相关的自定义 CSR 如表 2-39 所示。

本地中断的使能受 mie 和 locien0~ locien2 寄存器控制，具体的中断编号和使能寄存器的映射关系请参考表 2-40，归纳后的设置方法为：

- 对于中断编号 N 且 N 小于 32 的中断，通过 mie 寄存器的第 Nbit 控制中断使能。
- 对于中断编号 N 且 N 大于等于 32 的中断，通过 locien((N-32)/32)寄存器的第(N%32）bit 控制中断使能。

本地中断的挂起状态可通过 mip 和 locipd0~ locipd2 寄存器读取，具体的中断编号和挂起状态寄存器的映射关系请参考表 2-40，归纳后的设置方法为：

- 对于中断编号 N 且 N 小于 32 的中断，通过 mip 寄存器的第 Nbit 读取中断挂起状态。
- 对于中断编号 N 且 N 大于等于 32 的中断，通过 locipd((N-32）/32)寄存器的第(N%32）bit 读取中断挂起状态。

本地中断的优先级通过 locipri0~ locipri11 寄存器进行配置，具体的中断编号和优先级寄存器的映射关系请参考表 2-40，对于中断编号 N 的中断，中断优先级由 locipri((N-26)/8）寄存器的(((N-26)%8)*4)到(((N-26)%8)*4+3)共 4bit 控制。

每个本地中断支持的可设置优先级为 0~15，数值越大优先级越高，即 15 为最高优先级，而中断优先级设置为 0 的中断不会被处理器响应。如果多个中断设置的中断优先级相同，则中断编号越小的中断优先级越高。

处理器定义了 prithd CSR 控制中断响应优先级的全局阈值，当一个使能的中断处于挂起状态且优先级大于中断优先级阈值才会被处理器响应并处理。

处理器还实现了 locipclr CSR 用于清除中断挂起信号，软件往该 CSR 写入需要清除挂起状态的中断的编号，对应中断的挂起信号会置 0。注意处理器响应中断时会自动清除挂起状态，此软件主动清除功能主要用于一次响应处理多个中断的场景。

另外处理器实现了 cipri CSR 记录当前被响应中断的优先级。

表2-39 非标准本地中断相关的自定义 CSR

偏移地址	权限	名称	描述	页码
0x7ED	MRO	cipri	当前中断优先级寄存器。	cipri
0xBC0~0xBCB	MRW	locipri0~ locipri11	非标准本地中断优先级寄存器 0~11。	locipri0~ locipri11
0xBE0~0xBE2	MRW	locien0~ locien2	非标准本地中断使能寄存器 0~2。	locien0~ locien2
0xBE8~0xBEA	MRW	locipd0~ locipd2	非标准本地中断挂起状态寄存器	locipd0~

偏移地址	权限	名称	描述	页码
			0~2。	locipd2
0xBF0	MRW	locipclr	非标准本地中断挂起状态清除寄存器。	locipclr
0xBFE	MRW	prithd	中断优先级阈值寄存器。	prithd

表2-40 各非标准本地中断的寄存器地址和位域

本地中断编号	使能寄存器	优先级寄存器	挂起状态寄存器
26	mie.bit[26]	locipri0.bit[3:0]	locipd0.bit[0]
27	mie.bit[27]	locipri0.bit[7:4]	locipd0.bit[1]
28	mie.bit[28]	locipri0.bit[11:8]	locipd0.bit[2]
29	mie.bit[29]	locipri0.bit[15:12]	locipd0.bit[3]
30	mie.bit[30]	locipri0.bit[19:16]	locipd0.bit[4]
31	mie.bit[31]	locipri0.bit[23:20]	locipd0.bit[5]
32	locien0.bit[0]	locipri0.bit[27:24]	locipd0.bit[6]
33	locien0.bit[1]	locipri0.bit[31:28]	locipd0.bit[7]
34	locien0.bit[2]	locipri1.bit[3:0]	locipd0.bit[8]
35	locien0.bit[3]	locipri1.bit[7:4]	locipd0.bit[9]
36	locien0.bit[4]	locipri1.bit[11:8]	locipd0.bit[10]
37	locien0.bit[5]	locipri1.bit[15:12]	locipd0.bit[11]
38	locien0.bit[6]	locipri1.bit[19:16]	locipd0.bit[12]
39	locien0.bit[7]	locipri1.bit[23:20]	locipd0.bit[13]
40	locien0.bit[8]	locipri1.bit[27:24]	locipd0.bit[14]
41	locien0.bit[9]	locipri1.bit[31:28]	locipd0.bit[15]
42	locien0.bit[10]	locipri2.bit[3:0]	locipd0.bit[16]
43	locien0.bit[11]	locipri2.bit[7:4]	locipd0.bit[17]
44	locien0.bit[12]	locipri2.bit[11:8]	locipd0.bit[18]
45	locien0.bit[13]	locipri2.bit[15:12]	locipd0.bit[19]
46	locien0.bit[14]	locipri2.bit[19:16]	locipd0.bit[20]
47	locien0.bit[15]	locipri2.bit[23:20]	locipd0.bit[21]

本地中断编号	使能寄存器	优先级寄存器	挂起状态寄存器
48	locien0.bit[16]	locipri2.bit[27:24]	locipd0.bit[22]
49	locien0.bit[17]	locipri2.bit[31:28]	locipd0.bit[23]
50	locien0.bit[18]	locipri3.bit[3:0]	locipd0.bit[24]
51	locien0.bit[19]	locipri3.bit[7:4]	locipd0.bit[25]
52	locien0.bit[20]	locipri3.bit[11:8]	locipd0.bit[26]
53	locien0.bit[21]	locipri3.bit[15:12]	locipd0.bit[27]
54	locien0.bit[22]	locipri3.bit[19:16]	locipd0.bit[28]
55	locien0.bit[23]	locipri3.bit[23:20]	locipd0.bit[29]
56	locien0.bit[24]	locipri3.bit[27:24]	locipd0.bit[30]
57	locien0.bit[25]	locipri3.bit[31:28]	locipd0.bit[31]
58	locien0.bit[26]	locipri4.bit[3:0]	locipd1.bit[0]
59	locien0.bit[27]	locipri4.bit[7:4]	locipd1.bit[1]
60	locien0.bit[28]	locipri4.bit[11:8]	locipd1.bit[2]
61	locien0.bit[29]	locipri4.bit[15:12]	locipd1.bit[3]
62	locien0.bit[30]	locipri4.bit[19:16]	locipd1.bit[4]
63	locien0.bit[31]	locipri4.bit[23:20]	locipd1.bit[5]
64	locien1.bit[0]	locipri4.bit[27:24]	locipd1.bit[6]
65	locien1.bit[1]	locipri4.bit[31:28]	locipd1.bit[7]
66	locien1.bit[2]	locipri5.bit[3:0]	locipd1.bit[8]
67	locien1.bit[3]	locipri5.bit[7:4]	locipd1.bit[9]
68	locien1.bit[4]	locipri5.bit[11:8]	locipd1.bit[10]
69	locien1.bit[5]	locipri5.bit[15:12]	locipd1.bit[11]
70	locien1.bit[6]	locipri5.bit[19:16]	locipd1.bit[12]
71	locien1.bit[7]	locipri5.bit[23:20]	locipd1.bit[13]
72	locien1.bit[8]	locipri5.bit[27:24]	locipd1.bit[14]
73	locien1.bit[9]	locipri5.bit[31:28]	locipd1.bit[15]
74	locien1.bit[10]	locipri6.bit[3:0]	locipd1.bit[16]
75	locien1.bit[11]	locipri6.bit[7:4]	locipd1.bit[17]
76	locien1.bit[12]	locipri6.bit[11:8]	locipd1.bit[18]
77	locien1.bit[13]	locipri6.bit[15:12]	locipd1.bit[19]

本地中断编号	使能寄存器	优先级寄存器	挂起状态寄存器
78	locien1.bit[14]	locipri6.bit[19:16]	locipd1.bit[20]
79	locien1.bit[15]	locipri6.bit[23:20]	locipd1.bit[21]
80	locien1.bit[16]	locipri6.bit[27:24]	locipd1.bit[22]
81	locien1.bit[17]	locipri6.bit[31:28]	locipd1.bit[23]
82	locien1.bit[18]	locipri7.bit[3:0]	locipd1.bit[24]
83	locien1.bit[19]	locipri7.bit[7:4]	locipd1.bit[25]
84	locien1.bit[20]	locipri7.bit[11:8]	locipd1.bit[26]
85	locien1.bit[21]	locipri7.bit[15:12]	locipd1.bit[27]
86	locien1.bit[22]	locipri7.bit[19:16]	locipd1.bit[28]
87	locien1.bit[23]	locipri7.bit[23:20]	locipd1.bit[29]
88	locien1.bit[24]	locipri7.bit[27:24]	locipd1.bit[30]
89	locien1.bit[25]	locipri7.bit[31:28]	locipd1.bit[31]
90	locien1.bit[26]	locipri8.bit[3:0]	locipd2.bit[0]
91	locien1.bit[27]	locipri8.bit[7:4]	locipd2.bit[1]
92	locien1.bit[28]	locipri8.bit[11:8]	locipd2.bit[2]
93	locien1.bit[29]	locipri8.bit[15:12]	locipd2.bit[3]
94	locien1.bit[30]	locipri8.bit[19:16]	locipd2.bit[4]
95	locien1.bit[31]	locipri8.bit[23:20]	locipd2.bit[5]
96	locien2.bit[0]	locipri8.bit[27:24]	locipd2.bit[6]
97	locien2.bit[1]	locipri8.bit[31:28]	locipd2.bit[7]
98	locien2.bit[2]	locipri9.bit[3:0]	locipd2.bit[8]
99	locien2.bit[3]	locipri9.bit[7:4]	locipd2.bit[9]
100	locien2.bit[4]	locipri9.bit[11:8]	locipd2.bit[10]
101	locien2.bit[5]	locipri9.bit[15:12]	locipd2.bit[11]
102	locien2.bit[6]	locipri9.bit[19:16]	locipd2.bit[12]
103	locien2.bit[7]	locipri9.bit[23:20]	locipd2.bit[13]
104	locien2.bit[8]	locipri9.bit[27:24]	locipd2.bit[14]
105	locien2.bit[9]	locipri9.bit[31:28]	locipd2.bit[15]
106	locien2.bit[10]	locipri10.bit[3:0]	locipd2.bit[16]
107	locien2.bit[11]	locipri10.bit[7:4]	locipd2.bit[17]

本地中断编号	使能寄存器	优先级寄存器	挂起状态寄存器
108	locien2.bit[12]	locipri10.bit[11:8]	locipd2.bit[18]
109	locien2.bit[13]	locipri10.bit[15:12]	locipd2.bit[19]
110	locien2.bit[14]	locipri10.bit[19:16]	locipd2.bit[20]
111	locien2.bit[15]	locipri10.bit[23:20]	locipd2.bit[21]
112	locien2.bit[16]	locipri10.bit[27:24]	locipd2.bit[22]
113	locien2.bit[17]	locipri10.bit[31:28]	locipd2.bit[23]
114	locien2.bit[18]	locipri11.bit[3:0]	locipd2.bit[24]
115	locien2.bit[19]	locipri11.bit[7:4]	locipd2.bit[25]
116	locien2.bit[20]	locipri11.bit[11:8]	locipd2.bit[26]
117	locien2.bit[21]	locipri11.bit[15:12]	locipd2.bit[27]
118	locien2.bit[22]	locipri11.bit[19:16]	locipd2.bit[28]
119	locien2.bit[23]	locipri11.bit[23:20]	locipd2.bit[29]
120	locien2.bit[24]	locipri11.bit[27:24]	locipd2.bit[30]
121	locien2.bit[25]	locipri11.bit[31:28]	locipd2.bit[31]

2.4.2.4 硬件中断嵌套

处理器支持硬件中断嵌套，无需任何软件干预即可支持高优先级中断抢占低优先级中断的能力，处理器同时支持硬件自动换栈功能。

通过 CSR 指令将 mibnk[MB]设置成 1 打开处理器的硬件中断嵌套功能。在硬件中断嵌套功能打开的情况下，在处理器响应一个中断的下一拍即能继续响应任意更高优先级中断从而实现中断抢占嵌套，但整型和浮点通用寄存器仍需要中断服务函数软件进行出入栈保护和恢复。

在使能硬件中断嵌套功能的情况下，通过 CSR 指令将 mibnk[HSS]设置写成 1 即可打开处理器的硬件自动换栈功能。在使能硬件自动换栈功能的情况下，处理器从主程序流即非中断服务函数中响应中断，会自动交换 X2 通用寄存器和 mscratch CSR 中的值，出现中断嵌套时则不会触发自动交换行为，嵌套状态下从中断服务函数返回也不会触发自动交换行为，而从最外一层中断的服务函数返回主程序流会再次交换 X2 通用寄存器和 mscratch CSR 中的值，从而自动实现业务栈和中断栈的切换。

须知

整型和浮点通用寄存器的上下文保护由中断服务函数软件实现。

在没有使能硬件中断嵌套功能的情况下，不可以使能硬件自动换栈功能。

2.4.3 硬件错误与死锁

在处理器运行主程序流时被中断或者异常打断进入中断或异常服务函数，在中断或异常服务函数中处理器能提供足够的信息返回主程序流继续执行指令。而当处理器在响应中断或者异常进入服务函数后，又响应了新的 NMI 中断或者同步异常，处理器会再次进入服务函数造成异常嵌套，此时处理器无法提供足够的信息退出两层嵌套返回主程序流中，这种两层异常嵌套的状态被称为硬件错误状态（Hard Fault）。

当处理器已经处于硬件错误状态，又再一次出现 NMI 中断或者同步异常，处理器会进入死锁状态（Lock Up），进入死锁状态后处理器会停止工作，不会继续执行指令也不会响应中断，但此时仿真器还是可以接管处理器执行调试操作。

另外一种会让处理器进入死锁状态的场景是在已经响应 NMI 中断执行中断处理函数的时候，出现了同步异常事件。

表 2-41 描述了各种状态下出现不同事件时，状态的更新情况。

表2-41 硬件错误和死锁场景

-	出现 NMI	出现同步异常	出现异步异常	出现中断
NMI 中断处理函数	保持当前状态	进入死锁状态	保持当前状态	保持当前状态
异常处理函数	进入 NMI 处理函数	进入硬件错误状态	保持当前状态	保持当前状态
中断处理函数	进入 NMI 处理函数	进入硬件错误状态	保持当前状态	保持当前状态
硬件错误状态	进入死锁状态	进入死锁状态	保持当前状态	保持当前状态

2.5 MTimer

2.5.1 概述

MTimer 模块可用于完成定时和计数功能，MTimer 模块挂载在 MCU 外设总线上，处理器通过加载或存储指令访问 MTimer 的寄存器，不支持通过 CSR 指令直接访问 MTimer 模块的寄存器。

2.5.2 特性

- 具有 1 个 64bit 的计数寄存器，每次访问只能读写高 32bit 或者低 32bit。
- 具有 1 个 64bit 的比较寄存器，每次访问只能读写高 32bit 或者低 32bit。
- 当 MTimer 使能且计数寄存器值大于比较寄存器值时会上报中断，中断方式为电平中断，往比较寄存器的高 32bit 或者低 32bit 写入任意值会清除中断状态。
- 计数时钟与处理器工作时钟一致。
- 支持时钟分频计数，可以设置分频值 N（1~1024），每隔 N 个计数时钟周期计数寄存器加 1。
- 支持寄存器控制调试模式下是否停止计数。

2.5.3 功能描述

MTimer 的核心是一个 64bit 的加法计数器，当 MTimer 处于使能状态时，根据选择的计数时钟和配置的分频值 N（1~1024），每隔 N 个计数时钟周期计数值加 1。

当 MTimer 使能且计数值大于等于设置的比较寄存器时，会上报一个电平中断，计数值继续加 1 递增。

当计数值为 0xFFFFFFFFFFFFFFFF 时，继续加 1 会导致溢出，此时计数值会变成 0，然后继续加 1 计数。

MTimer 的中断状态可以通过往比较寄存器的高 32bit 或者低 32bit 写入任意值进行清除。

2.5.4 工作方式

2.5.4.1 初始化

系统初始化时应对 MTimer 进行初始化，初始化时应按以下步骤进行配置：

步骤 1 配置 TIMER_DIV 寄存器设置分频值，该寄存器为减 1 配置，复位后默认值为 0，即 1 分频。

步骤 2 根据定时时长需要，配置 MTIMECMP 和 MTIMECMP_H 设置比较值。

步骤 3 复位后计数值默认值为 0，如果有需要，可以配置 MTIME 和 MTIME_H 寄存器设置计数初始值。

步骤 4 配置 TIMER_CTRL 寄存器，控制调试模式下是否继续计数，选择计数时钟源，同时启动 MTimer 计数。

----结束

2.5.4.2 64bit 计数值读取

由于每次只能读取 32bit 的计数值，读取完整的 64bit 计数值需要两次读操作，而两次读取之间还在继续计数，所以正确读取 64bit 计数值需要遵循以下步骤：

步骤 1 读取 MTIME_H 寄存器，获取高 32bit 的计数值。

步骤 2 读取 MTIME 寄存器，获取低 32bit 的计数值。

步骤 3 再次读取 MTIME_H 寄存器，获取高 32bit 的计数值。

步骤 4 比较步骤 1 和步骤 3 读取的两次高 32bit 计数值，如果两次结果不一致，回到步骤 1 重新执行，直到两次结果一致后可组合高低 32bit 计数值得到 64bit 的计数值。

----结束

2.5.4.3 中断处理和 64bit 比较值更新

对于固定周期定时触发中断的应用，初始化完成后，响应 MTimer 中断后根据定时周期更新比较值，即可让 MTimer 按固定周期产生中断，参考步骤如下：

步骤 1 根据定时周期要求计算新的比较值。

步骤 2 配置 MTIMECMP_H 寄存器，将比较值的高 32bit 设置为 0xFFFFFFFF。

步骤 3 配置 MTIMECMP 寄存器，设置比较值的低 32bit 为更新值。

步骤 4 配置 MTIMECMP_H 寄存器，设置比较值的高 32bit 为更新值。

须知

往 MTIMECMP 或者 MTIMECMP_H 写入任意值即可清除中断状态，所以不需要额外的中断清除步骤。

重新设置比较值时，需要两次 32bit 写操作，先将比较值的高 32bit 设置为 0xFFFFFFFF 可以防止写入更新值过程中出现中间值导致误报中断。

当计算新的 64bit 比较值时出现溢出，建议重置计数值和比较值，否则会出现中断误报。

----结束

2.5.5 MTimer 寄存器概览

MTimer 寄存器概览如表 2-42 所示。

表2-42 MTimer 寄存器概览（基址是 0x14380000）

偏移地址	名称	描述	页码
0x0000	TIMER_CTRL	MTimer 控制寄存器	TIMER_CTRL
0x0004	TIMER_DIV	MTimer 分频寄存器	2.5.5 MTimer 寄存器概览
0x0008	MTIME	MTimer 计数值低 32bit 寄存器	2.5.5 MTimer 寄存器概览
0x000C	MTIME_H	MTimer 计数值高 32bit 寄存器	2.5.5 MTimer 寄存器概览
0x0010	MTIMECMP	MTimer 比较值低 32bit 寄存器	MTIMECMP
0x0014	MTIMECMP_H	MTimer 比较值高 32bit 寄存器	MTIMECMP_H

2.5.6 MTimer 寄存器描述

TIMER_CTRL

TIMER_CTRL 为 MTimer 控制寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2]	RW	stop_tmr_en	调试模式停止计数控制。 0：调试模式下定时间继续计数； 1：调试模式下定时器停止计数。	0x0
[1]	RW	clksrc	MTimer 时钟源选择。 0：使用主时钟（处理器时钟）； 1：保留。	0x0
[0]	RW	enable	MTimer 使能。 0：MTimer 不使能； 1：MTimer 使能，当计数值大于等于比较值时会上报中断。	0x0

TIMER_DIV

TIMER_DIV 为 MTimer 分频寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9:0]	RW	div	定时器分频控制，每 div+1 个计时时钟周期会让计数寄存器加 1。	0x000

MTIME

MTIME 为 MTimer 计数值低 32bit 寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	mtime	MTimer 计数寄存器低 32bit。	0x00000000

MTIME_H

MTIME_H 为 MTimer 计数值高 32bit 寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	mtime_h	MTimer 计数寄存器高 32bit，当计数器低 32bit 溢出时加 1。	0x00000000

MTIMECMP

MTIMECMP 为 MTimer 比较值低 32bit 寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	mtimecmp	MTimer 比较寄存器低 32bit，写入任意值会清除中断状态。	0x00000000

MTIMECMP_H

MTIMECMP_H 为 MTimer 比较值高 32bit 寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	mtimecmp_h	MTimer 比较寄存器高 32bit，写入任意值会清除中断状态。	0x00000000

2.6 调试功能

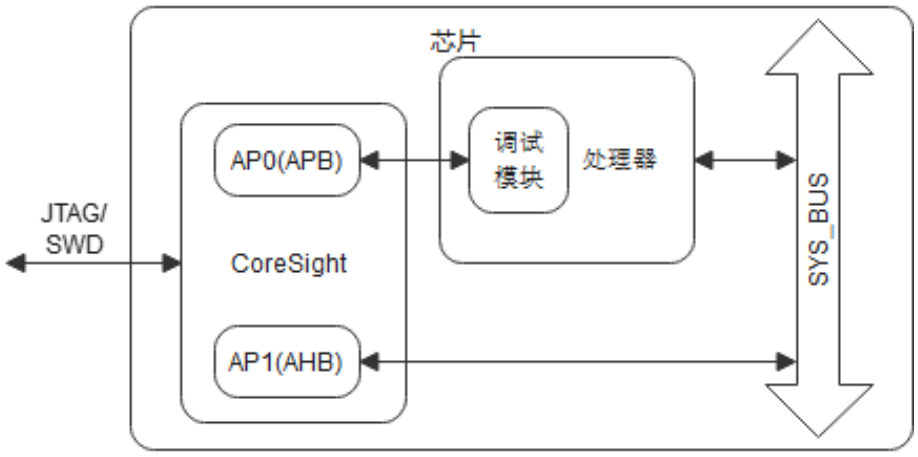
本处理器实现了 RISC-V debug spec 0.13 的调试协议，支持访问整型通用寄存器的 abstract command，支持 3 个 program buffer，支持 ndmreset、halt、resume、step 控制功能，支持 4 个硬件 trigger，硬件 tigger 支持指令 PC 比较、load store 地址比较、store 数据比较，从而实现断点和 watchpoint。

通过与调试工具配合可以完成处理器运行控制、单步调试、处理器资源访问（包括通用寄存器和控制与状态寄存器）、硬件断点和 watchpoint、通过处理器访问系统资源（包括存储和外设）等调试功能。

具体调试功能请参考《RISC-V 调试标准》，推荐使用已支持 RISC-V 调试标准的常用调试工具套件。

MCU 的调试通路如图 2-3 所示，外部调试器可通过 JTAG（Joint Test Action Group）接口或者 SWD（Serial Wire Debug）接口访问 MCU 的 CoreSight 模块，CoreSight 模块有两个 AP（Access Port）口，AP0 为 APB（Advanced Peripheral Bus）总线接口，可访问处理器的调试模块从而实现处理器的调试功能，AP1 为 AHB 总线接口且连接到系统总线 SYS_BUS 上，可通过系统总线访问 SYSRAM、FLASH 以及外设模块。

图2-3 调试通路架构示意图



3 嵌入式闪存(Flash)

3.1 概述

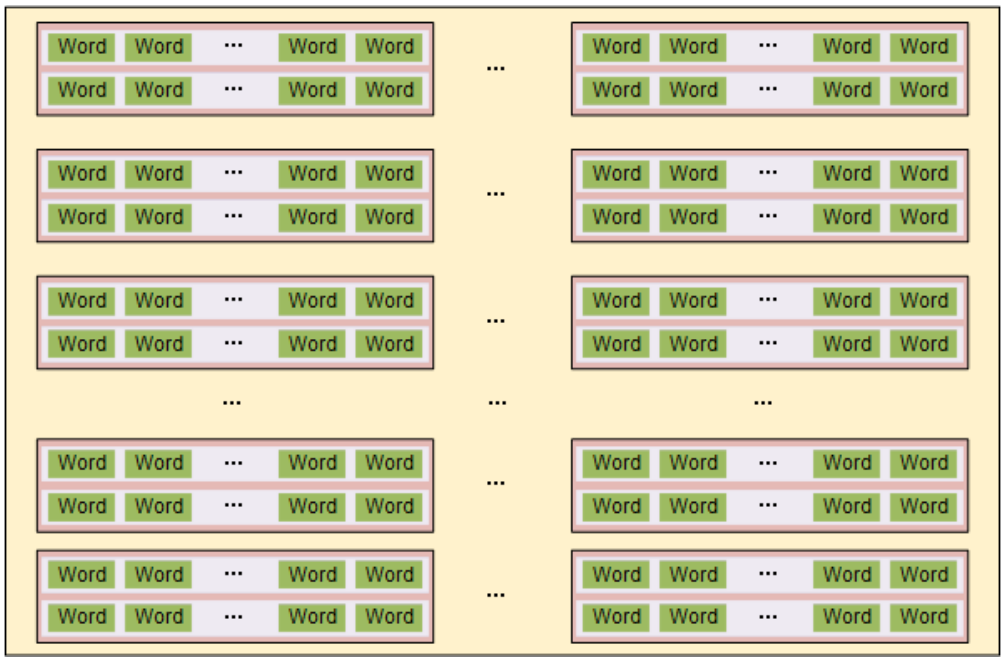
3.1.1 Flash 存储器介绍

MCU 内置 Flash 存储器具备可擦除、可编程、快速读取数据、非易失以及可重复等特性，可用于存储用户程序、数据等。

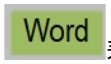
Flash 存储器的最小存储单元是 bit，64 个 bit 组合成一个 Flash-Word，64 个 Flash-Word 组合成一个 Flash-Row（64bit × 64 = 512Byte），2 个 Flash-Row 组合成一个 Flash-Page（512Byte × 2 = 1KB）。

Flash 存储器内部管理如图 3-1 所示。

图3-1 Flash 内部存储结构示意图



说明

- 
表示一个 Flash-Page；
- 
表示一个 Flash-Row；
- 
表示一个 Flash-Word。

3.1.2 Flash 控制器介绍

Flash 控制器处理对 Flash 存储器的擦除（Erase）、编程（Program）和读（Read）操作以及读写保护和安全机制。

Flash 控制器通过预取和缓存系统加速代码执行。

3.2 特性

- 高达 128KB Flash 存储空间（主存储区）
- 存储空间管理：
  - 主存储区（Main Region）

main_rgn0：程序与用户参数存储区，大小用户可配。

main_rgn1：程序与用户参数存储区，大小用户可配。

- 信息存储区（Information Region）

  info_rgn0：固定参数存储区。

  info_rgn1：受保护参数存储区。

  info_rgn2：启动引导程序区。

- 页大小（Page size）：1KB。
- 行大小（Row size）：512Byte。

- 64bit 读数据宽度
- 64bit 编程数据宽度

Flash 闪存接口特性：

- XIP（eXecute In Place）特性
- Flash 命令序列方式编程/擦除操作
- 主存储区 JTAG 读保护
- 信息存储区 JTAG 操作保护
- 预取缓冲区
- 直接读取操作缓存：256 Cache Lines（2KB）
- 错误码纠错：64bit 纠一检二

3.3 功能描述

3.3.1 Flash 存储空间管理

Flash 存储器以 64bit 为最小操作单元进行数据存储管理。

表3-1 Flash 的存储空间按下表进行管理（基地址 0x0300_0000）

区域		起始地址	结束地址	区域容量	建议用途
Main Region		0x00_0000	0x01_FFFF	128KB	程序+参数
Information Region	info_rgn0	0x80_0000	0x80_0FFF	4KB	固定参数
	info_rgn1	0x80_1000	0x80_17FF	2KB	保护参数

区域		起始地址	结束地址	区域容量	建议用途
	info_rgn2	0x80_1800	0x80_3FFF	10KB	引导程序

3.3.2 Flash 控制器初始化

3.3.2.1 Flash 时钟控制

Flash 读时钟由系统时钟分频而来，Flash 的读时钟与系统时钟频率关系如表 3-2 所示。

表3-2 Flash 读分频系数与系统时钟频率适配表

系统时钟频率（MHz）	nread_div 值
150 ≥ clk > 112.5	3
112.5 ≥ clk > 75	2
75 ≥ clk > 37.5	1
37.5 ≥ clk	0

当用户修改系统时钟频率时，必须按以下流程进行修改。

系统时钟升频步骤

步骤 1 参考表 3-2 要求配置 Flash 读分频系数 EFLASH_CLK_CFG[nread_div]。

步骤 2 配置 EFLASH_CLK_CFG[busclk_sw_req]为 1，等待 EFLASH_CLK_CFG[busclk_sw_req]自动清零，表示 nread_div 修改已经生效。

步骤 3 修改系统时钟频率为目标频率。

----结束

系统时钟降频步骤

步骤 1 修改系统时钟频率为目标频率。

步骤 2 参考表 3-2 要求配置 Flash 读分频系数 EFLASH_CLK_CFG[nread_div]。

步骤 3 配置 EFLASH_CLK_CFG[busclk_sw_req]为 1，等待 EFLASH_CLK_CFG[busclk_sw_req]自动清零 0，表示 nread_div 修改已经生效。

----结束

3.3.2.2 Flash 错误码纠错控制

数据存储在 Flash 内部时，对数据以 64bit 为单位进行 ECC（Error Correcting Code）保护。

- 编程时，对 64bit 数据生成一个 ECC 校验码，存储在 Flash 内部。
- 读数据时，对存储器读出数据进行检错纠错。
- 支持 64bit 数据实现纠 1 检 2 功能：
  - 1bit 数据发生错误，可以检测错误，可以纠正；
  - 2bit 数据发生错误，可以检测错误，不能纠正；
  - 2bit 以上数据发生错误，不能正确检测，不能纠正。

  如果 1bit 错误被检测和纠正，INT_RAW_STATUS[int_raw_err_ecc_chk]标志位将会被置位，如果 INT_ENABLE[int_en_err_ecc_chk]使能，将产生相关中断；

  如果 2bit 错误被检测，INT_RAW_STATUS [int_raw_err_ecc_corr]标志位将会被置位，如果 INT_ENABLE[int_en_err_ecc_corr]使能，将产生相关中断。
- 对已擦除未编程的数据，提供 ECC 过滤检查功能：
  - FLASH_ECC_CTRL[flash_ecc_blank_filter_enable]为“0”时，读数据根据 FLASH_ECC_CTRL 寄存器控制进行 Main Region/Information Region 检查或纠正；
  - FLASH_ECC_CTRL[flash_ecc_blank_filter_enable]为“1”时，过滤读数据内容，未编程的数据不进行 ECC 检查与纠正。

3.3.2.3 Flash Main region 空片标志查询

Flash 控制器在初始化过程中会自动读取 Main Region 首地址数据，并判断首地址是否经过编程，用户可以访问 Flash 控制器寄存器 FLASH_INIT_STATUS[main_rgn_empty]获取此标志。

3.3.2.4 Flash 低功耗管理

在无 Flash 操作期间，提供自动管理 Flash 进入低功耗的功能；Flash 控制器的编程/擦除逻辑用户可以软件动态配置管理；Flash 控制器内的 SRAM（Static Random

Accecss Memory）动态时钟门控功能可以软件配置开启，此三类低功耗措施可通过以下步骤实现。

Flash 自动低功耗

步骤 1 配置 LP_CNT_CFG[sleep_cnt]，设定进入 Flash 休眠状态的空闲时间阈值。

步骤 2 配置 LOW_POWER_CFG[auto_sleep_en]为 1，使能自动低功耗管理功能，当无 Flash 操作时间达到 LP_CNT_CFG[sleep_cnt]设定值时，Flash 控制器控制 Flash 进入休眠状态，此状态下当收到新的 Flash 操作请求时，自动退出休眠状态。

须知

从休眠状态唤醒需要 10us。

----结束

Flash 控制器擦除/编程低功耗

配置 LOW_POWER_CFG[apb_cmd_clk_en]为 0，关闭擦除/编程操作逻辑时钟。

须知

在此状态下，禁止发起擦除/编程请求。

Flash 控制器 Cache 低功耗

配置 LOW_POWER_CFG[sram_ck_gt_en]为 1，使能内部 SRAM 时钟动态门控功能，此功能使能后，SRAM 只有在需要读写的时候打开时钟，实现 cycle 级别的低功耗控制。

3.3.2.5 Flash 直接读取访问加速

3.3.2.5.1 Cache 机制

Flash 控制器直接访问读接口提供一个具有 256 Cache Line（64bit）的 Cache 供使用，Cache 替换策略为 PLRU（Pseudo Least Recently Used）策略。

Cache 功能默认不开启，如果用户需要开启 Cache 功能，通过以下步骤操作：

步骤 1 选择 Cache 查询策略。

配置 CACHE_CTRL[cache_policy_sel]。

0：Normal Cache，所有的数据请求都会查询是否在 Cache 中；

1：Branch Cache，只有跳转分支的数据会查询 Cache，顺序读取数据不查询 Cache，只有当 prefetch_enable 生效时，此配置有效，否则都是 Normal Cache。

步骤 2 使能 Cache 功能。

配置 CACHE_CTRL[cache_enable]为 1，使能 Cache 功能；

配置 CACHE_CTRL[cache_enable]为 0，关闭 Cache 功能。

步骤 3 Cache 数据无效请求。

当系统启动后，用户擦除/编程 Flash 后，Cache 中的数据可能会与 Flash 存储数据不一致，用户应该发送 Cache 数据无效请求，防止访问到 Cache 中的不一致数据。

配置 CACHE_CTRL[cache_invalid_req]为 1，发送 Cache 数据无效请求给 Flash 控制器。

步骤 4 Cache 数据无效请求结果查询。

读 CACHE_CTRL[cache_invalid_req]，检查是否为 0，如果自动清零，表示 Flash 控制器已经完成 Cache 数据无效操作，否则请继续等待此 bit 清零。

----结束

3.3.2.5.2 预取机制

Flash 控制器直接读接口提供读顺序地址访问向前预取功能，此功能默认关闭，可以通过配置 PFEFETCH_CTRL[prefetch_enable]位域，动态配置开启与关闭。

此功能有助于减小 CPU 访问 Flash 的 Latency，建议开启使用。

须知

使能 PFEFETCH_CTRL[prefetch_enable]后，如果进行擦除/编程操作后，预取 buffer 中的数据与 Flash 中数据会不一致，需要软件通过配置 PFEFETCH_CTRL[prefetch_invalid_req]的操作丢弃预取 Buffer 中的残留数据。

3.3.3 Flash 擦除操作

Flash 存储器在编程之前必须先进行擦除操作。

Flash 控制器提供两种擦除方式，Mass Erase 与 Page Erase。Mass Erase 会将整个 Main Region 擦除掉，Page Erase 则只擦除指定擦除地址所处的 Page 区域。两种擦除的步骤如下。

须知

擦除操作未完成时下电，可能导致 Flash 存储器发生致命损伤，擦除未完成时，禁止对 Flash 下电。

Mass Erase 操作步骤

步骤 1 配置 LOW_POWER_CFG[apb_cmd_clk_en]为 1，使能相关逻辑。

步骤 2 查询 EFLASH_CMD.bit[0]为 0，确认无 Flash CMD（Command，即擦除或编程操作）在执行。

步骤 3 配置 MAGIC_LOCK 寄存器为 0xFEDC_BA98，解锁 Erase。

步骤 4 配置 EFLASH_ADDR 寄存器为 0。

步骤 5 配置 EFLASH_CMD.bit[10:8]为 110，选择 Mass Erase。

步骤 6 配置 EFLASH_CMD.bit[0]为 1，启动 Erase 操作。

步骤 7 如是中断模式，跳过此步骤。

如果是轮询模式，等待 EFLASH_CMD.bit[0]为 0，表示 Erase 操作已结束。

步骤 8 查询 INT_RAW_STATUS.bit[17:16]位域，检查是否存在错误：

- bit[16]=1：表示之前操作未执行，请通过步骤 9 确认原因；
- bit[17]=1：表示操作已经执行，但 Erase 未成功，可能是存储介质异常。

步骤 9 查询 FLASH_STATUS 寄存器确认错误原因。

步骤 10 配置 INT_CLEAR 寄存器，清除相应的中断。

步骤 11 配置 MAGIC_LOCK 寄存器为 0x0，锁定 Erase/Program CMD。

----结束

Page Erase 操作步骤

Page Erase 可以擦除 Main Region/Information Region 内的 Page，一次操作擦除一个 Page。

步骤 1 配置 LOW_POWER_CFG[apb_cmd_clk_en]为 1，使能相关逻辑。

步骤 2 查询 EFLASH_CMD.bit[0]为 0，确认无 Flash CMD 在执行。

步骤 3 配置 MAGIC_LOCK 寄存器为 0xFEDC_BA98，解锁 Erase 操作。

步骤 4 配置 EFLASH_ADDR.bit[23]位域，选择 Region；bit[23]=0 选择 Main Region，bit[23]=1 选择 Information Region；配置 EFLASH_ADDR.bit[22:0]位域，选择具体 Region 内 Page Address（地址按 Page Size 对齐）。

步骤 5 配置 EFLASH_CMD.bit[10:8]为 100 选择 Main Region，EFLASH_CMD.bit[10:8]为 101 选择 Information Region，要求与步骤 4 的地址选择 Region 匹配。

步骤 6 配置 EFLASH_CMD.bit[0]为 1，启动 Erase 操作。

步骤 7 如果是中断模式，跳过此步骤。

如果是轮询模式，等待查询 EFLASH_CMD.bit[0]为 0，表示 Erase 操作已结束。

步骤 8 查询 INT_RAW_STATUS.bit[17:16]位域，检查是否存在错误：

- bit[16]=1：表示之前操作未执行，请通过步骤 9 确认原因；
- bit[17]=1：表示操作已经执行，但 Erase 未成功，可能是存储介质异常。

步骤 9 查询 FLASH_STATUS 寄存器，确认错误原因。

步骤 10 配置 INT_CLEAR 寄存器，清除相应的中断。

步骤 11 配置 MAGIC_LOCK 寄存器为 0x0，锁定 Erase/Program CMD。

----结束

3.3.4 Flash 编程操作

同一个 Flash 存储单元（4bit）在擦除之后只允许编程 1 次，如果需要重复编程，需要先对相应地址区域执行擦除操作，要求用户软件做此管理。

一次编程命令支持编程的数据大小为：8Byte、16Byte × n，n 为 1~16 的整数；同时要求一次编程命令，“起始地址 + 编程数据大小”的寻址范围不能跨越 Flash-Row（512Byte）边界。

须知

编程操作未完成时下电，可能导致 Flash 存储器发生致命损伤，编程未完成时，禁止对 Flash 下电。

步骤 1 配置 LOW_POWER_CFG[apb_cmd_clk_en]为 1，使能相关逻辑。

步骤 2 查询 EFLASH_CMD.bit[0]为 0，确认无 Flash CMD 在执行。

步骤 3 配置 MAGIC_LOCK 寄存器为 0xFEDC_BA98，解锁 Program 操作。

步骤 4 查询 BUF_CLEAR.bit[15:8]，确认 Program Buffer 无历史残留数据，如果有残留数据配置 BUF_CLEAR.bit[0]为 1，进行历史数据清理。

步骤 5 配置 PGM_WDATA 寄存器，写入编程数据到 Program Buffer，数据从低位地址开始写入。

步骤 6 配置 EFLASH_ADDR.bit[23]位域，选择 Region；bit[23]=0 选择 Main Region，bit[23]=1 选择 Information Region；配置 bit[22:2]选择具体 Region 内编程起始地址，要求一次 Program 寻址范围地址不能跨越 Flash-Row（512Byte）边界。

步骤 7 配置 EFLASH_CMD.bit[10:8]为 100 选择 Main Region，EFLASH_CMD.bit[10:8]为 101 选择 Information Region，要求与步骤 6 的地址选择 Region 匹配。

步骤 8 配置 EFLASH_CMD.bit[0]为 1，启动 Program 操作。

步骤 9 如是中断模式，跳过此步骤。

- 查询 EFLASH_CMD.bit[0]为，表示 Program 操作已结束。

步骤 10 查询 INT_RAW_STATUS.bit[16]是否存在错误：

bit[16]=1：表示之前操作未执行，请跟进步骤 11 确认原因。

步骤 11 查询 FLASH_STATUS 寄存器，确认错误原因。

步骤 12 配置 INT_CLEAR 寄存器，清除相应的中断。

步骤 13 配置 MAGIC_LOCK 寄存器为 0x0，锁定 Erase/Program CMD。

----结束

3.3.5 Flash 读操作

直接地址读操作

所有区域都可以通过直接地址访问方式实现数据读取。

须知

直接访问时需要避免访问超过有效地址范围外的操作，此行为会导致系统错误；

直接访问数据时，若发生 ECC 错误，会生成
INT_RAW_STATUS[int_raw_err_ecc_chk]与
INT_RAW_STATUS[int_raw_err_ecc_corr]错误中断。

3.4 info_rgn0 描述

info_rgn0 指起始地址 0x0080_0000 到结束地址 0x0080_0FFF 之间的 FLASH 存储空间，容量 4KB，称为“固定参数区”。其中数据在 MCU 出厂时已烧写，用户不可烧写，仅支持读取部分值。

表3-3 Flash info_rgn0 描述列表（基地址 0x0080_0000）

偏移地址（128bit 为一组）	组内 bit 偏移	名称	描述
0x0000	31:0	DIE_ID_0	MCU 出厂 DIE_ID_0，DIE_ID 寄存器。
	63:32	DIE_ID_1	MCU 出厂 DIE_ID_1，DIE_ID 寄存器。
	95:64	DIE_ID_2	MCU 出厂 DIE_ID_2，DIE_ID 寄存器。
	127:96	DIE_ID_3	MCU 出厂 DIE_ID_3，DIE_ID 寄存器。
0x0010	127:0	reserved	保留，用户不可烧写。
0x0020	31:0	CHIP_ID	MCU 出厂 CHIP_ID 寄存器。
	127:32	reserved	保留，用户不可烧写。
0x0020~0x0FFF	-	reserved	保留，用户不可烧写。

3.5 info_rgn1 描述

info_rgn1 指起始地址 0x0080_1000 到结束地址 0x0080_17FF 之间的 FLASH 存储空间，容量 2KB，称为“保护参数区”。其中数据可由用户烧写。

表3-4 Flash info_rgn1 描述列表（基地址 0x0080_1000）

偏移地址（128bit 为一组）	组内 bit 偏移	名称	描述
0x0000	7:0	protection_level	Protection_level 权限管理标志。 0xFF：JTAG/SWD 接口权限开放； 其他：JTAG/SWD 接口权限受限，只能访问 Flash 寄存器空间，读写擦 eflash main rgn1 区域，以及读取 info_rgn0/1 区域。 当 info_rgn1_unlock 处于未锁定（0xFF）时，支持保护等级回退机制：通过 JTAG/SWD 接口或 CPU 向 Flash 发起 protection_level 回退请求，会依次擦除 Flash main_rgn0/1 和 info_rgn1。
	127:8	reserved	保留。
0x0010	0:0	uart0_enable	UART0 使能标志。 0：禁用； 1：使能。
	1:1	uart1_enable	UART1 使能标志。 0：禁用； 1：使能。
	2:2	uart2_enable	UART2 使能标志。 0：禁用； 1：使能。
	3:3	uart3_enable	UART3 使能标志。

偏移地址（128bit 为一组）	组内 bit 偏移	名称	描述
			0：禁用； 1：使能。
	31:4	reserved	保留。
	39:32	func_jtag_enable	JTAG/SWD 使能标志。 0xFF：允许； 其他：禁止。 禁止：无法通过 JTAG/SWD 接口访问 MCU； 允许：可以通过 JTAG/SWD 接口访问 MCU。
	40:40	sysram_parity_disable	SYSRAM 奇偶校验使能标志。 0：使能； 1：禁用。
	63:41	reserved	保留。
	64:64	uart_boot_enable	UART BOOT 升级使能标志。 0：禁用，bootrom 无法通过 UART 进行升级； 1：使能，bootrom 可以通过 UART 进行升级。
	65:65	spi_boot_enable	SPI BOOT 升级使能标志。 0：禁用，bootrom 无法通过 SPI 进行升级； 1：使能，bootrom 可以通过 SPI 进行升级。
	66:66	i2c_boot_enable	I2C BOOT 升级使能标志。 0：禁用，bootrom 无法通过 I2C 进行串口升级； 1：使能，bootrom 可以通过 I2C 进行串口升级。

偏移地址（128bit 为一组）	组内 bit 偏移	名称	描述
	67:67	can_boot_enable	CAN BOOT 升级使能标志。 0：禁用，bootrom 无法通过 CAN 进行串口升级； 1：使能，bootrom 可以通过 CAN 进行串口升级。
	95:68	reserved	保留。
	105:96	main_rgn0_size	main rgn0 容量大小配置，单位：Page（1KB/Page）。 main rgn 中小于等于 main_rgn0_size 的 Page 属于 main rgn0，大于 main_rgn0_size 的 Page 属于 main rgn1。
	127:106	reserved	保留。
0x0020	7:0	info_rgn1_unlock	info_rgn1 区域锁定标志。 0xFF：未锁定； 其他：锁定。 锁定状态下，无法对 info_rgn1 区域进行擦除、编程操作，不支持保护等级回退功能。
	127:8	reserved	保留。
0x0030~0x07FF	-	reserved	保留，用户自定义。

3.6 Flash 分区保护机制

Flash 提供对四个存储空间进行不同的数据访问权限保护控制。

- Main Region：
  - 用户通过 info_rgn1 中的 main_rgn0_size 选项字节自定义 Main Region 的分区权限控制。

- main_rgn0：

  FLASH_REGION1_CTRL.bit[31]为 1 时，标识存在 main_rgn0；为 0 时，标识不存在 main_rgn0；

  FLASH_REGION0_START_ADDR 与 FLASH_REGION0_END_ADDR 标识 main_rgn0 的地址范围；

  FLASH_REGION0_CTRL.bit[23:8]标识不同 mid 的 Master 是否有权限对 main_rgn0 擦除和编程。

- main_rgn1：

  FLASH_REGION1_CTRL.bit[31]为 1 时，标识存在 main_rgn1；为 0 时，标识不存在 main_rgn1；

  FLASH_REGION1_START_ADDR 与 FLASH_REGION1_END_ADDR 标识 main_rgn1 的地址范围；

  FLASH_REGION1_CTRL.bit[23:8]标识不同 mid 的 master 是否有权限对 main_rgn1 有擦除和编程。

- info_rgn0：

  此区域存储 MCU 规格等信息，不可被修改。

- info_rgn1：

  此区域支持用户自定义修改权限，请参见表 3-4 关于 protection_level 和 info_rgn1_unlock 选项 bit 的描述。

- info_rgn2：

  此区域存储 MCU 启动引导程序，不可被修改。

3.7 Flash 保护

3061M 系列产品使用以下 Flash 参数来控制 MCU 的调试能力：

- info_rgn1 参数 func_jtag_enable：
  - 为 0xFF 时，JTAG/SWD 接口可用，可访问 MCU 所有寄存器空间，可对 Flash 进行读取、编程和擦除操作，可访问 SRAM。
  - 为 0xFF 以外的值时，JTAG/SWD 接口禁用。
  - 未烧写该参数时，默认值为 0xFF。
- info_rgn1 参数 protection level：

  - 为 0xFF 以外的值时，JTAG/SWD 接口受限，只能访问 Flash 寄存器空间，读取、编程、擦除 Flash main rgn1 区域，以及读取 info_rgn0/1 区域。若想取消该限制，可通过将 protection level 参数从非 0xFF 值擦除至 0xFF 来实现 MCU 保护等级的回退。
  - 由于为 0xFF 以外的值时，JTAG/SWD 接口不能擦除 info_rgn1 区域，因此只能通过发起 protection level 回退操作来擦除该参数。protection level 回退功能在本节描述。
  - 为 0xFF 时，JTAG/SWD 不受上述限制。
  - 未烧写该参数时，默认值为 0xFF。
- info_rgn1 参数 info_rgn1_unlock：
  - 为 0xFF 以外的值时，无法对 info_rgn1 区域进行擦除、编程操作，此时可以限制 protection level 回退功能的发起，即保持对 JTAG/SWD 接口权限的限制。
  - 为 0xFF 时，无上述限制。
  - 未烧写该参数时，默认值为 0xFF。

用户可根据实际需求，通过编程 Flash 中的上述参数来调整 MCU 保护级别。

须知

通过编程调整时，需要通过系统复位或者上下电来触发，使调整生效，而不是编程完 Flash 参数后立即触发。

Protection level 回退功能

在 info_rgn1_unlock 为 0xFF 时，可以通过 JTAG/SWD 接口或 CPU 向 Flash 发起 Protection level 回退操作，该操作会擦除 main_rgn0、main_rgn1 区和 info_rgn1 区，在擦除 info_rgn1 区的时候会将 protection level 和 func_jtag_enable 这两个参数擦为 0xFF。

在 info_rgn1_unlock 为 0xFF 时发起回退操作，flash 控制器将通过寄存器 CLR_PRT_LEVEL_RESULT 返回错误标志，这也被视为一种防止保护级别从“更安全”级别回退到“欠安全”级别的一种保护手段。

Protection level 回退具体步骤如下：

步骤 1 确保 info_rgn1_unlock 的值为 0xFF，若为其他值，无法发起 protection level 回退操作。可以通过读取 Flash 的 shadow 寄存器 INFO_RGN1_UNLOCK 获取其值。

步骤 2 配置 CLR_PRT_LEVEL_OP 值为 0xABEF_CD00（配置其他值无法发起命令）。

步骤 3 查询 CLR_PRT_LEVEL_RESULT。当查询到 CLR_PRT_LEVEL_RESULT[execute_finish]为 0x1 时，代表 protection level 回退命令已经成功执行完成。当查询到 CLR_PRT_LEVEL_RESULT[executing_state]为 0x1 时，代表回退命令执行中。当查询到 CLR_PRT_LEVEL_RESULT[info_rgn1_unlock_err]为 0x1 时，代表当前的 info_rgn1_unlock 值不为 0xFF，不支持发起 protection level 回退。

须知

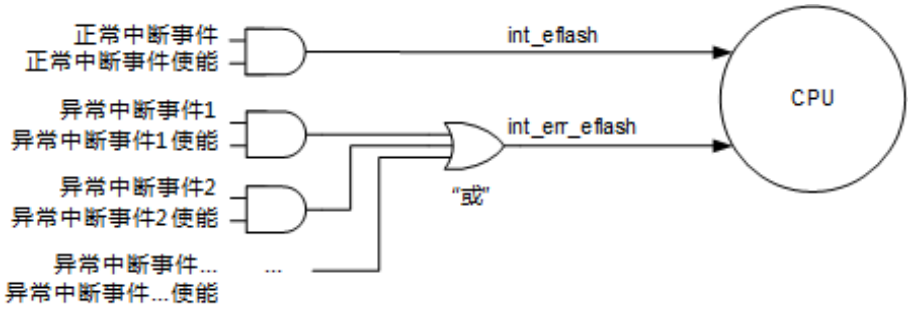
回退操作未完成时禁止对 Flash 下电，否则可能导致 Flash 存储器发生致命损伤。

----结束

3.8 Flash 中断

Flash 控制器提供两个硬中断到 CPU，区分正常中断与异常中断，其原理如图 3-2 所示。

图3-2 Flash 中断原理示意图



原始中断事件如表 3-5 和表 3-6 所示描述。

表3-5 Flash 控制器正常中断列表

中断事件	中断事件说明	中断使能控制	中断清除
int_raw_finish	Flash CMD 操作完成原始中断寄存	int_en_finish	int_clr_finish

中断事件	中断事件说明	中断使能控制	中断清除
	器。 0：无操作或者操作未完成； 1：操作已完成。		寄存器写 1

表3-6 Flash 控制器异常中断列表

中断事件	中断事件说明	中断使能控制	中断清除
int_raw_err_ecc_chk	main_rgn0/main_rgn1 读数据 ECC 检验错误原始中断寄存器。 0：无错误，或者读数据 ECC 检错功能未使能； 1：发生 ECC 检查错误。	int_en_err_ecc_chk	int_clr_err_ecc_chk 寄存器写 1
int_raw_err_ecc_corr	main_rgn0/main_rgn1 读数据 ECC 纠正错误原始中断寄存器。 0：无错误，或者读数据 ECC 纠错功能未使能； 1：发生无法纠正的 ECC 错误。	int_en_err_ecc_corr	int_clr_err_ecc_corr 寄存器写 1
int_raw_err_ahb	AHB 请求错误原始中断寄存器。 0：无错误； 1：AHB 读请求地址超过 main_rgn0/main_rgn1 范围或发生了 AHB 写请求。	int_en_err_ahb	int_clr_err_ahb 寄存器写 1
int_err_erase	擦除错误原始中断状态寄存器。 0：擦除操作 Verify 成功； 1：擦除操作 Verify 失败。	int_en_err_erase	int_clr_err_erase 寄存器写 1
int_raw_err_illegal	非法 Flash CMD 操作错误原始中断寄存器。 0：无错误；	int_en_err_illegal	int_clr_err_illegal 寄存器写 1

中断事件	中断事件说明	中断使能控制	中断清除
	1：存在 Flash CMD 操作错误，具体错误，需要查询 FLASH_STATUS 寄存器。		

3.9 Flash 寄存器

3.9.1 Flash 控制器普通寄存器概览

表3-7 Flash 控制器普通寄存器概览(基址是 0x1471_0000)

偏移地址	名称	描述	页码
0x000	EFLASH_CMD	命令寄存器。	EFLASH_CMD
0x004	EFLASH_ADDR	地址寄存器。	EFLASH_ADDR
0x080	CMD_CFG_COMMON	Flash CMD 配置寄存器。	CMD_CFG_COMMON
0x100	INT_RAW_STATUS	中断原始状态寄存器。	INT_RAW_STATUS
0x104	INT_STATUS	中断状态寄存器。	INT_STATUS
0x108	INT_ENABLE	中断使能配置寄存器。	INT_ENABLE
0x10C	INT_CLEAR	中断清除寄存器。	INT_CLEAR
0x120	PFEFETCH_CTRL	预取控制寄存器。	PFEFETCH_CTRL
0x124	CACHE_CTRL	Cache 控制寄存器。	CACHE_CT

偏移地址	名称	描述	页码
			RL
0x12C	FLASH_ECC_CTRL	Flash ECC 检错纠错使能控制寄存器。	FLASH_ECC_CTRL
0x130	FLASH_STATUS	Flash CMD 操作 FLASH 状态寄存器。	FLASH_STATUS
0x138	AHB_ERR_ADDR	AHB 错误请求地址记录寄存器。	AHB_ERR_ADDR
0x144	FLASH_REGION0_START_ADDR	main_rgn0 起始地址寄存器。	FLASH_REGION0_START_ADDR
0x148	FLASH_REGION0_END_ADDR	main_rgn0 结束地址寄存器。	FLASH_REGION0_END_ADDR
0x14C	FLASH_REGION0_CTRL	main_rgn0 控制寄存器。	FLASH_REGION0_CTRL
0x150	FLASH_REGION1_START_ADDR	main_rgn1 起始地址寄存器。	FLASH_REGION1_START_ADDR
0x154	FLASH_REGION1_END_ADDR	main_rgn1 结束地址寄存器。	FLASH_REGION1_END_ADDR
0x158	FLASH_REGION1_CTRL	main_rgn1 控制寄存器。	FLASH_REGION1_CTRL

偏移地址	名称	描述	页码
0x200	MAGIC_LOCK	Flash CMD 操作魔术字保护寄存器。	MAGIC_LOCK
0x3F0	EFLASH_CAPACITY_0	模块信息寄存器 0。	EFLASH_CAPACITY_0
0x3F4	EFLASH_CAPACITY_1	模块信息寄存器 1。	EFLASH_CAPACITY_1
0x3F8	EFLASH_CAPACITY_2	模块信息寄存器 2。	EFLASH_CAPACITY_2
0x400	PGM_WDATA	编程数据寄存器。	PGM_WDATA
0x604	BUF_CLEAR	编程数据 Buffer 清理寄存器。	PGM_WDATA

3.9.2 Flash 控制器普通寄存器描述

须知

访问 EFLASH_CMD、CMD_CFG_COMMON 和 PGM_WDATA 寄存器前，需要配置解锁 MAGIC_LOCK 寄存器，否则写入或者读取无效。

EFLASH_CMD

EFLASH_CMD 为命令寄存器。

Offset Address: 0x000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	-	reserved	保留。	0x00
[25:20]	RW	cmd_pgm_size	Program Size 寄存器，Unit:Word（32bit）。	0x00

Bits	Access	Name	Description	Reset
			0x0：2； 0x1：4； 0x2：8； ... 0x0F：60； 0x10：64； 其他：非法值。 cmd_addr + cmd_pgm_size 寻址范围不能跨越 Flash-Row（64Word），cmd_addr 需要按 64bit 对齐。	
[19:11]	-	reserved	保留。	0x000
[10:8]	RW	cmd_code	操作字寄存器。 010：MAIN Program； 011：INFO Program； 100：MAIN Erase； 101：INFO Erase； 110：Mass Erase（Only Main 区，不含 INFO 区）； 其他：无操作。	0x0
[7:6]	RO	exec_state	Flash CMD 执行状态 debug 寄存器。 00（IDLE）：无操作或操作已完成； 01（RUNNING）：有操作在执行； 10（FINISH）：操作完成，但完成标示未清除（中断模式才生效）； 11：保留。	0x0
[5:1]	-	reserved	保留。	0x00
[0]	RWSC	cmd_start	Flash CMD 操作启动寄存器。 写 1 启动 Flash CMD 操作； 写 0 无影响；	0x0

Bits	Access	Name	Description	Reset
			读 1 Flash CMD 操作未完成; 读 0 Flash CMD 操作完成或无操作。	

EFLASH_ADDR

EFLASH_ADDR 为地址寄存器。

Offset Address: 0x004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:24]	-	reserved	保留。	0x00
[23:0]	RW	cmd_addr	Program 或者 Erase 的起始地址寄存器，Unit:Byte（8bit）； Main Region 起始地址：0x00_0000; Information Region 起始地址：0x80_0000。 **注意：控制器自动忽略低 2bit 地址。**	0x000000

CMD_CFG_COMMON

CMD_CFG_COMMON 为 Flash CMD 配置寄存器。

Offset Address: 0x080 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RW	int_mode	Flash CMD 操作方式控制寄存器。 0：轮询方式; 1：中断方式，此方式 Flash CMD 操作完成后需要配置 int_clr_finish 清除完成标识，方可进行下一次操作。	0x0
[0]	-	reserved	保留。	0x0

INT_RAW_STATUS

INT_RAW_STATUS 为中断原始状态寄存器。

Offset Address: 0x100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	RO	int_raw_err_ecc_chk	Main Region 或 Information Region 读数据 ECC 检验错误原始中断寄存器。 0：无错误，或者读数据 ECC 检错功能未使能； 1：发生 ECC 检查错误。	0x0
[19]	RO	int_raw_err_ecc_corr	Main Region 或 Information Region 读数据 ECC 纠正错误原始中断寄存器。 0：无错误，或者读数据 ECC 纠错功能未使能； 1：发生无法纠正的 ECC 错误。	0x0
[18]	RO	int_raw_err_ahb	AHB 请求错误原始中断寄存器。 0：无错误； 1：AHB 读请求地址超过 Main Region 或 Information Region 范围或发生了 AHB 写请求。	0x0
[17]	RO	int_raw_err_erase	擦除错误原始中断状态寄存器。 0：擦除操作 Verify 成功； 1：擦除操作 Verify 失败。	0x0
[16]	RO	int_raw_err_illegal	非法 Flash CMD 操作错误原始中断寄存器。 0：无错误； 1：存在 Flash CMD 操作错误，具体错误，需要查询 FLASH_STATUS 寄存器。	0x0
[15:5]	-	reserved	保留。	0x000
[4]	RO	int_raw_finish	Flash CMD 操作完成原始中断寄存器。	0x0

Bits	Access	Name	Description	Reset
			0：无操作或者操作未完成； 1：操作已完成。	
[3:0]	-	reserved	保留。	0x0

INT_STATUS

INT_STATUS 为中断状态寄存器。

Offset Address: 0x104 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	RO	int_err_ecc_chk	Main Region 或 Information Region 读数据 ECC 检验错误中断状态寄存器。 0：无错误，或者读数据 ECC 检错功能未使能； 1：发生 ECC 检查错误。	0x0
[19]	RO	int_err_ecc_corr	Main Region 或 Information Region 读数据 ECC 纠正错误中断状态寄存器。 0：无错误，或者读数据 ECC 纠错功能未使能； 1：发生无法纠正的 ECC 错误。	0x0
[18]	RO	int_err_ahb	AHB 请求错误中断状态寄存器。 0：无错误； 1：AHB 读请求地址超过 Main Region 或 Information Region 范围或发生了 AHB 写请求。	0x0
[17]	RO	int_err_erase	擦除错误中断状态寄存器。 0：擦除操作 Verify 成功； 1：擦除操作 Verify 失败。	0x0
[16]	RO	int_err_illegal	非法 Flash CMD 操作错误中断状态寄存	0x0

Bits	Access	Name	Description	Reset
			器。 0：无错误； 1：存在 Flash CMD 操作错误，具体错误，需要查询 FLASH_STATUS 寄存器。	
[15:5]	-	reserved	保留。	0x000
[4]	RO	int_finish	Flash CMD 操作完成中断状态寄存器。 0：无操作或者操作未完成； 1：操作已完成。	0x0
[3:0]	RO	reserved	保留。	0x0

INT_ENABLE

INT_ENABLE 为中断使能配置寄存器。

Offset Address: 0x108 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	RW	int_en_err_ecc_chk	Main Region 或 Information Region 读数据 ECC 检验错误中断使能寄存器。 0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	0x0
[19]	RW	int_en_err_ecc_corr	Main Region 或 Information Region 读数据 ECC 纠正错误中断使能寄存器。 0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	0x0
[18]	RW	int_en_err_ahb	AHB 请求错误中断使能寄存器。	0x0

Bits	Access	Name	Description	Reset
			0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	
[17]	RW	int_en_err_erase	擦除错误中断使能寄存器。 0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	0x0
[16]	RW	int_en_err_illegal	非法 Flash CMD 操作错误中断使能寄存器。 0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	0x0
[15:5]	-	reserved	保留。	0x000
[4]	RW	int_en_finish	Flash CMD 操作完成中断使能寄存器。 0：不使能，上报原始中断，不上报中断状态，不上报 CPU 中断； 1：使能，上报原始中断，上报中断状态，上报 CPU 中断。	0x0
[3:0]	-	reserved	保留。	0x0

INT_CLEAR

INT_CLEAR 为中断清除寄存器。

Offset Address: 0x10C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	W1_PUL	int_clr_err_ecc_ch	Main Region 或 Information Region 读数	0x0

Bits	Access	Name	Description	Reset
	SE	k	据 ECC 检验错误中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	
[19]	W1_PULSE	int_clr_err_ecc_corr	Main Region 或 Information Region 读数据 ECC 纠正错误中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	0x0
[18]	W1_PULSE	int_clr_err_ahb	AHB 请求错误中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	0x0
[17]	W1_PULSE	int_clr_err_erase	擦除错误中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	0x0
[16]	W1_PULSE	int_clr_err_illegal	非法 Flash CMD 操作错误中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	0x0
[15:5]	-	reserved	保留。	0x000
[4]	W1_PULSE	int_clr_finish	Flash CMD 操作完成中断清除寄存器。 0：不清除； 1：清除原始中断与中断状态。	0x0
[3:0]	-	reserved	保留。	0x0

PFEFETCH_CTRL

PFEFETCH_CTRL 为预取控制寄存器。

Offset Address: 0x120 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RWSC	prefetch_invalid_req	Cache 数据无效请求控制寄存器。 写 0：无效； 写 1：请求 prefetch_invalid，逻辑完成 invalid 后自动清除该 bit； 读 0：最近一次 invalid 请求已完成； 读 1：最近一次 invalid 请求未完成。	0x0
[7:1]	-	reserved	保留。	0x00
[0]	RW	prefetch_enable	预取使能控制寄存器。 0：预取使能关闭； 1：每次 AHB 传输完成尾地址向前预取 16Byte 数据（以 8Byte 对齐）。	0x0

CACHE_CTRL

CACHE_CTRL 为 Cache 控制寄存器。

Offset Address: 0x124 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12]	RW	cache_policy_sel	Cache 策略选择控制寄存器。 0：Normal Cache； 1：Branch Cache。	0x0
[11:9]	-	reserved	保留。	0x0
[8]	RWSC	cache_invalid_req	Cache 数据无效请求控制寄存器。 写 0：无效； 写 1：请求 Cache invalid，逻辑完成 invalid 后自动清除该 bit； 读 0：最近一次 invalid 请求已完成；	0x0

Bits	Access	Name	Description	Reset
			读 1：最近一次 invalid 请求未完成。	
[7:1]	-	reserved	保留。	0x00
[0]	RW	cache_enable	Cache 使能控制寄存器。 0：Cache 使能关闭； 1：Cache 使能打开，AHB 传输数据将会查询 Cache，Cache Hit 则数据从 Cache 获取，Cache miss 则数据从 eFlash 获取，并将此地址数据缓存进入 Cache。	0x0

FLASH_ECC_CTRL

FLASH_ECC_CTRL 为 Flash ECC 检错纠错使能控制寄存器。

Offset Address: 0x12C Total Reset Value: 0x0000_001C

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RW	flash_ecc_blank_filter_enable	Flash 未编程区域 ECC 屏蔽过滤使能控制（72bit 全 1 判断为未编程区域）。 0：不使能此功能，所有 Read 访问数据都根据 flash_ecc_correct_enable、flash_ecc_check_enable 配置进行相应的 ECC 检查或者纠正； 1：使能此功能，检查 Read 访问数据内容，只有非全 1（72bit）数据根据 flash_ecc_correct_enable、flash_ecc_check_enable 配置进行相应的 ECC 检查或者纠正。	0x1
[3]	RW	flash_info_ecc_correct_enable	Flash Information Region ECC 纠错功能控制。 0：回读数据发生错误时，不进行 ECC 纠错； 1：回读数据发生 1bit 错误时，进行 ECC	0x1

Bits	Access	Name	Description	Reset
			纠错。	
[2]	RW	flash_info_ecc_check_enable	Flash Information Region 检错使能控制（Program 强制 ECC 打开）。 0：回读数据不进行 ECC 检查； 1：回读数据进行 ECC 检查。	0x1
[1]	RW	flash_main_ecc_correct_enable	Flash Main Region ECC 纠错功能控制。 0：回读数据发生错误时，不进行 ECC 纠错； 1：回读数据发生 1bit 错误时，进行 ECC 纠错。	0x0
[0]	RW	flash_main_ecc_check_enable	Flash Main Region 检错使能控制（Program 强制 ECC 打开）。 0：回读数据不进行 ECC 检查； 1：回读数据进行 ECC 检查。	0x0

FLASH_STATUS

FLASH_STATUS 为 Flash CMD 操作 FLASH 状态寄存器。

Offset Address: 0x130 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:18]	-	reserved	保留。	0x0000
[17]	RO	address_unmap	Flash CMD 操作地址是否越界指示寄存器。 0：无错误； 1：地址越界错误。	0x0
[16]	RO	parameter_illegal	Flash CMD 操作参数是否有效指示寄存器。 0：无错误； 1：操作参数错误。	0x0

Bits	Access	Name	Description	Reset
[15:14]	-	reserved	保留。	0x0
[13]	RO	main_rgn1_illegal	Flash CMD 操作 main_rgn1 是否发生非法访问指示寄存器。 0：无错误； 1：main_rgn1 发生非法访问。	0x0
[12]	RO	main_rgn0_illegal	Flash CMD 操作 main_rgn0 是否发生非法访问指示寄存器。 0：无错误； 1：main_rgn0 发送非法访问。	0x0
[11]	-	reserved	保留。	0x0
[10]	RO	info_rgn2_illegal	Flash CMD 操作 info_rgn2 是否发生非法访问指示寄存器。 0：无错误； 1：info_rgn2 发生非法访问。	0x0
[9]	RO	info_rgn1_illegal	Flash CMD 操作 info_rgn1 是否发生非法访问指示寄存器。 0：无错误； 1：info_rgn1 发生非法访问。	0x0
[8]	RO	info_rgn0_illegal	Flash CMD 操作 info_rgn0 是否发生非法访问指示寄存器。 0：无错误； 1：info_rgn0 发生非法访问。	0x0
[7]	-	reserved	保留。	0x0
[6:4]	RO	mid_illegal	发生非法访问时，Flash CMD 寄存器非法 mid value。	0x0
[3]	-	reserved	保留。	0x0
[2:0]	RO	opcode_illegal	发生非法访问时，Flash CMD 寄存器非法 opcode value。	0x0

AHB_ERR_ADDR

AHB_ERR_ADDR 为 AHB 错误请求地址记录寄存器。

Offset Address: 0x138 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	ahb_err_address	int_raw_err_ahb 发生时，记录第一次发生 AHB 错误访问的地址信息，包括越界与 AHB 写操作。 配置中断清除寄存器 int_clr_err_ahb 时，该寄存器清 0，并开始记录下一次错误请求地址信息。	0x00000000

FLASH_REGION0_START_ADDR

FLASH_REGION0_START_ADDR 为 MAIN REGION0 起始地址寄存器。

Offset Address: 0x144 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16:2]	RO	main_rgn0_start_addr	main_rgn0 起始地址寄存器，Unit:Word（32bit）。	0x0000
[1:0]	-	reserved	保留。	0x0

FLASH_REGION0_END_ADDR

FLASH_REGION0_END_ADDR 为 MAIN REGION0 结束地址寄存器。

Offset Address: 0x148 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16:2]	RO	main_rgn0_end_addr	main_rgn0 结束地址寄存器，Unit:Word（32bit）。	0x0000

Bits	Access	Name	Description	Reset
[1:0]	-	reserved	保留。	0x0

FLASH_REGION0_CTRL

FLASH_REGION0_CTRL 为 MAIN REGION0 控制寄存器。

Offset Address: 0x14C Total Reset Value: 0x00FF_FFFF

Bits	Access	Name	Description	Reset
[31]	RO	main_rgn0_active	是否激活区域访问控制寄存器。 0：不激活; 1：激活。	0x0
[30:24]	-	reserved	保留。	0x00
[23:16]	RO	main_rgn0_mid_e	允许对 main_rgn0 擦除操作的 mid。 bit[7:0]分别标识 mid 为 7~0 的 master 是否允许擦除操作。 0：禁止; 1：允许。	0xFF
[15:8]	RO	main_rgn0_mid_p	允许对 main_rgn0 编程操作的 mid。 bit[7:0]分别标识 mid 为 7~0 的 master 是否允许编程操作。 0：禁止; 1：允许。	0xFF
[7:0]	-	reserved	保留。	0x00

FLASH_REGION1_START_ADDR

FLASH_REGION1_START_ADDR 为 MAIN REGION1 起始地址寄存器。

Offset Address: 0x150 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16:2]	RO	main_rgn1_start_addr	main_rgn1 起始地址寄存器，Unit:Word（32bit）。	0x0000
[1:0]	-	reserved	保留。	0x0

FLASH_REGION1_END_ADDR

FLASH_REGION1_END_ADDR 为 MAIN REGION1 结束地址寄存器。

Offset Address: 0x154 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16:2]	RO	main_rgn1_end_addr	main_rgn1 结束地址寄存器，Unit:Word（32bit）。	0x0000
[1:0]	-	reserved	保留。	0x0

FLASH_REGION1_CTRL

FLASH_REGION1_CTRL 为 MAIN REGION1 控制寄存器。

Offset Address: 0x158 Total Reset Value: 0x00FF_FFFF

Bits	Access	Name	Description	Reset
[31]	RO	main_rgn1_active	是否激活区域访问控制寄存器。 0：不激活； 1：激活。	0x0
[30:24]	-	reserved	保留。	0x00
[23:16]	RO	main_rgn1_mid_e	允许对 main_rgn1 擦除操作的 mid。 bit[7:0]分别标识 mid 为 7~0 的 master 是否允许擦除操作。 0：禁止；	0xFF

Bits	Access	Name	Description	Reset
			1：允许。	
[15:8]	RO	main_rgn1_mid_p	允许对 main_rgn1 编程操作的 mid。 bit[7:0]分别标识 mid 为 7 ~ 0 的 master 是否允许编程操作。 0：禁止； 1：允许。	0xFF
[7:0]	-	reserved	保留。	0x00

MAGIC_LOCK

MAGIC_LOCK 为 Flash CMD 操作魔术字保护寄存器。

Offset Address: 0x200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	magic_lock	写入 0xFEDC_BA98，魔术字解锁，Flash CMD 相关操作寄存器写入权限打开； 写入其他数值，魔术字锁定，Flash CMD 相关操作寄存器写入权限关闭； 读出 0xFEDC_BA98 表示解锁，读出 0 表示锁定； 默认为锁定状态，Flash CMD 相关操作寄存器包括：EFLASH_CMD、EFLASH_ADDR、PGM_WDATA。	0x00000000

EFLASH_CAPACITY_0

EFLASH_CAPACITY_0 为模块信息寄存器 0。

Offset Address: 0x3F0 Total Reset Value: 0x0002_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	main_capacity	Main Region 容量指示寄存器， Unit:Byte。	0x00020000

EFLASH_CAPACITY_1

EFLASH_CAPACITY_1 为模块信息寄存器 1。

Offset Address: 0x3F4 Total Reset Value: 0x4000_0400

Bits	Access	Name	Description	Reset
[31:16]	RO	information_capacity	Information Region 容量指示寄存器， Unit:Byte。	0x4000
[15:0]	RO	page_size	eFlash Page 容量指示寄存器， Unit:Byte。	0x0400

EFLASH_CAPACITY_2

EFLASH_CAPACITY_2 为模块信息寄存器 2。

Offset Address: 0x3F8 Total Reset Value: 0x0A83_3332

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27:24]	RO	min_erase_size	最小擦除 Size（2**n）。 Unit:Byte。 1KB。	0xA
[23:20]	RO	max_pgm_size	连续编程最大 Size（2**n）。 Unit:Byte。 256Byte。	0x8
[19:16]	RO	min_pgm_size_main	Main Region 最小编程 Size（2**n）。 Unit:Byte。 8Byte。	0x3
[15:12]	RO	min_pgm_size_information	Information Region 最小编程 Size（2**n）。 Unit:Byte。 8Byte。	0x3

Bits	Access	Name	Description	Reset
[11:8]	RO	io_write_length_main	写 Main Region IO Size（2**n）。 Unit:Byte（不算 ECC bit）。 8Byte。	0x3
[7:4]	RO	io_write_length_information	写 Information Region IO Size（2**n）。 Unit:Byte（不算 ECC bit）。 8Byte。	0x3
[3:0]	RO	io_read_length	读 IO Size (2**n)。 Unit:Byte（不算 ECC bit）。 4Byte。	0x2

PGM_WDATA

PGM_WDATA 为编程数据寄存器。

Offset Address: 0x400 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	WO	pgm_wdata	编程数据 Buffer 寄存器。 0x400~0x5FC，都映射到同一写数据寄存器； 写 0x400~0x5FC 任意一个寄存器一次，写数据 Buffer 数据增加一个 Word（32bit），此 Buffer 数据先进先出。	0x00000000

BUF_CLEAR

BUF_CLEAR 为编程数据 Buffer 清理寄存器。

Offset Address: 0x604 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:8]	RO	pgm_wbuf_cnt	编程数据 Buffer 当前存量数据 size 指示寄	0x00

Bits	Access	Name	Description	Reset
			存器，Unit:Word（32bit）。	
[7:1]	-	reserved	保留。	0x00
[0]	W1_PULSE	pgm_wdata_clr	编程数据 Buffer 数据清除控制寄存器。 写 0：无效果； 写 1：清除当前 Buffer 中残留数据。	0x0

3.9.3 Flash 控制器特殊寄存器概览

表3-8 Flash 控制器特殊寄存器概览(基址是 0x1471_0800)

偏移地址	名称	描述	页码
0x140	EFLASH_CLK_CFG	时钟分频寄存器。	EFLASH_CLK_CFG
0x188	READ_QOS	读 QOS 配置寄存器。	READ_QOS
0x200	LOW_POWER_CFG	低功耗控制寄存器。	LOW_POWER_CFG
0x204	LP_CNT_CFG	低功耗计数控制寄存器。	LP_CNT_CFG
0x208	LOW_POWER_STATUS	低功耗状态寄存器。	LOW_POWER_STATUS
0x308	Flash_init_status	初始化状态寄存器。	FLASH_INIT_STATUS
0x310	ECC_CNT	ECC 错误计数寄存器。	ECC_CNT
0x314	ECC_ADDR	ECC 错误地址寄存器。	ECC_ADDR
0x318	ECC_SYNDORME	ECC 错误码寄存器。	ECC_SYNDORME

偏移地址	名称	描述	页码
0x324	CACHE_HIT_CLR	Cache 命中率清除寄存器。	CACHE_HIT_CLR
0x328	CACHE_LOOKUP_CNT	Cache 查询次数寄存器。	CACHE_LOOKUP_CNT
0x32C	CACHE_HIT_CNT	Cache 命中次数寄存器。	CACHE_HIT_CNT

3.9.4 Flash 控制器特殊寄存器描述

EFLASH_CLK_CFG

EFLASH_CLK_CFG 为时钟分频寄存器。

Offset Address: 0x140 Total Reset Value: 0x0000_0010

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	RWSC	busclk_sw_req	nread_div 是否切换完成配置查询寄存器。 写 1 后触发电路进行切换，回读判断是否切换完成。 读 0：切换完成; 读 1：切换未完成。	0x0
[19:8]	-	reserved	保留。	0x000
[7:4]	RW	nread_div	Flash Read，系统总线主时钟与 Flash 时钟比值（n+1）。 0x0：1 分频; 0x1：2 分频; ..... 0xF：16 分频。	0x1
[3:0]	-	reserved	保留。	0x0

READ_QOS

READ_QOS 为读 QOS 配置寄存器。

Offset Address: 0x188 Total Reset Value: 0x0100_2001

Bits	Access	Name	Description	Reset
[31:16]	RW	pe_wait_timeout_cylces	pe_wait 使能后，PE（Program and Erase）请求被阻塞 cycle 超过 pe_wait_timeout_cylces+pe_wait_cycles，提高 PE 请求优先级，优先响应 PE 请求。	0x0100
[15:8]	RW	pe_wait_cycle	为避免稀疏的 Read 操作被 PE 操作阻塞，在完成 Read 操作后维持 pe_wait_cycle 个 cycle（总线时钟），在此期间阻塞 PE 请求。 0 表示不阻塞。	0x20
[7:1]	-	reserved	保留。	0x00
[0]	RW	pe_wait_en	pe_wait 功能使能。 0：不使能； 1：使能，Read 结束后，在 pe_wait_cycle 时间内，阻塞 PE 请求，在此 cycle 内如果有新的 Read 请求，会优先响应 Read。	0x1

LOW_POWER_CFG

LOW_POWER_CFG 为低功耗控制寄存器。

Offset Address: 0x200 Total Reset Value: 0x0000_0100

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16]	RW	auto_sleep_en	自动休眠低功耗功能控制。 0：关闭； 1：开启，当 Main 状态机 IDLE 时间超过	0x0

Bits	Access	Name	Description	Reset
			sleep_cnt 阈值时，逻辑自动控制 Macro 进入 DPD（Deep Power Down）模式。	
[15:10]	-	reserved	保留。	0x00
[9]	RW	sram_ck_gt_en	内部 SRAM 时钟动态门控使能控制。 0：SRAM 时钟常开； 1：SRAM 时钟动态门控，需要读写时逻辑自动打开，读写结束自动关闭。	0x0
[8]	RW	apb_cmd_clk_en	Flash CMD 相关逻辑时钟使能控制。 0：关闭 Erase/Program 相关数字逻辑时钟； 1：打开 Erase/Program 相关数字逻辑时钟。	0x1
[7:2]	-	reserved	保留。	0x00
[1:0]	RWSC	switch_lp_mode	软件直接控制低功耗切换。 00：操作完成； 01：唤醒； 10：睡眠； 11：保留。 软件写入非 0 值，触发状态跳转，跳转完成寄存器自动回到 0。	0x0

LP_CNT_CFG

LP_CNT_CFG 为低功耗计数控制寄存器。

Offset Address: 0x204 Total Reset Value: 0x0000_0010

Bits	Access	Name	Description	Reset
[31:12]	-	reserved	保留。	0x00000
[11:0]	RW	sleep_cnt	自动低功耗 IDLE 等待时间（us 时钟计数）。	0x010

Bits	Access	Name	Description	Reset
			不可为 0，最大 2.6ms。 默认：16us。	

LOW_POWER_STATUS

LOW_POWER_STATUS 为低功耗状态寄存器。

Offset Address: 0x208 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1:0]	RO	lp_status	控制器低功耗状态指示寄存器。 00：正常工作； 01：初始化； 10：睡眠； 11：下电。	0x0

FLASH_INIT_STATUS

FLASH_INIT_STATUS 为初始化状态寄存器。

Offset Address: 0x308 Total Reset Value: 0x0000_0100

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RO	main_rgn_empty	Main Region 第一个地址是否被编程执行寄存器。 0：第一个地址被编程过； 1：第一个地址未被编程过。	0x1
[7:1]	-	reserved	保留。	0x00
[0]	RO	eflash_init_fail	Flash 初始化失败标志位。	0x0

ECC_CNT

ECC_CNT 为 ECC 错误计数寄存器。

Offset Address: 0x310 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	dfx_flash_ecc_correct_error_cnt	Main Region 或 Information Region 读数据发生 ECC 纠错失败次数，每次对应一个 64bit。 软件清除 int_raw_err_ecc_corr 原始中断后，重新统计。	0x0000
[15:0]	RO	dfx_flash_ecc_check_error_cnt	Main Region 或 Information Region 读数据发生 ECC 检查错误次数，每次对应一个 64bit。 软件清除 int_raw_err_ecc_chk 原始中断后，重新统计。	0x0000

ECC_ADDR

ECC_ADDR 为 ECC 错误地址寄存器。

Offset Address: 0x314 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	dfx_flash_ecc_first_correct_error_addr	读 Flash 第一次发生 ECC 检查错误的物理地址，Unit:64bit。 软件清除 int_raw_err_ecc_corr 原始中断后，重新统计。 最高 2 位表示区域： 00：表示 Main Region； 01：表示 Information Region。	0x0000
[15:0]	RO	dfx_flash_ecc_first_check_error_addr	读 Flash 第一次发生 ECC 检查错误的物理地址，Unit:64bit。 软件清除 int_raw_err_ecc_chk 原始中断后，重新统计。	0x0000

Bits	Access	Name	Description	Reset
			最高 2 位表示区域： 00：表示 Main Region； 01：表示 Information Region。	

ECC_SYNDORME

ECC_SYNDORME 为 ECC 错误码寄存器。

Offset Address: 0x318 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:24]	-	reserved	保留。	0x00
[23:16]	RO	dfx_flash_ecc_first_correct_error_syndrome	读 Flash 第一次发生 ECC 纠错错误的错误字符。 软件清除 int_raw_err_ecc_corr 原始中断后，重新统计。	0x00
[15:8]	-	reserved	保留。	0x00
[7:0]	RO	dfx_flash_ecc_first_check_error_syndrome	读 Flash 第一次发生 ECC 检查错误的错误字符。 软件清除 int_raw_err_ecc_chk 原始中断后，重新统计。	0x00

CACHE_HIT_CLR

CACHE_HIT_CLR 为 Cache 命中率清除寄存器。

Offset Address: 0x324 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	W1_PULSE	dfx_cache_hit_clr	dfx_cache_lookup_cnt 与 dfx_cache_hit_cnt 计数器清零控制寄存器。	0x0

Bits	Access	Name	Description	Reset
			写 0 无效果; 写 1 dfx_cache_lookup_cnt 与 dfx_cache_hit_cnt 清零。	

CACHE_LOOKUP_CNT

CACHE_LOOKUP_CNT 为 Cache 查询次数寄存器。

Offset Address: 0x328 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	dfx_cache_lookup_cnt	Cache 查询次数统计计数器。 Unit：次。	0x00000000

CACHE_HIT_CNT

CACHE_HIT_CNT 为 Cache 命中次数寄存器。

Offset Address: 0x32C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	dfx_cache_hit_cnt	cache 查询命中次数统计计数器。 Unit：次。	0x00000000

3.9.5 FLASH_RGN0_SHADOWS 寄存器概览

表3-9 FLASH_RGN0_SHADOWS 寄存器概览（基址是 0x1472_0000）

偏移地址	名称	描述	页码
0x0000	DIE_ID_0	MCU 出厂 DIE_ID_0，DIE_ID 寄存器。	3.9.6.1 DIE_ID_0
0x0004	DIE_ID_1	MCU 出厂 DIE_ID_1，DIE_ID 寄存器。	3.9.6.2 DIE_ID_1

偏移地址	名称	描述	页码
0x0010	DIE_ID_2	MCU 出厂 DIE_ID_2，DIE_ID 寄存器。	3.9.6.3 DIE_ID_2
0x0014	DIE_ID_3	MCU 出厂 DIE_ID_3，DIE_ID 寄存器。	3.9.6.4 DIE_ID_3
0x0020	CHIP_ID	MCU 出厂 CHIP_ID 寄存器。	3.9.6.5 CHIP_ID

3.9.6 FLASH_RGN0_SHADOWS 寄存器描述

本节内容描述 info_rgn0 的 shadow 寄存器。(基址是 0x1472_0000)

在 MCU 上电后，info_rgn0 的 shadow 寄存器会将部分 info_rgn0 的 FLASH 数据载入，用户可直接对指定寄存器地址进行读取，得到对应 info_rgn0 区域的值。

由于 info_rgn0 区域的数据不可被用户烧写，因此每次上电载入的值为固定值。

3.9.6.1 DIE_ID_0

DIE_ID_0 为 MCU 出厂 DIE_ID_0，DIE_ID 寄存器。

Offset Address: 0x0000 Total Reset Value: 0xXXXX_XXXX（X 代表出厂前编程）

Bits	Access	Name	Description	Reset
[31:0]	RO	die_id_0	MCU 出厂 DIE_ID_0，每颗 MCU 具有特异值。	每块 MCU 具有特异值，每次复位都为固定值。

3.9.6.2 DIE_ID_1

DIE_ID_1 为 MCU 出厂 DIE_ID_1，DIE_ID 寄存器。

Offset Address: 0x0004 Total Reset Value: 0xXXXX_XXXX（X 代表出厂前编程）

Bits	Access	Name	Description	Reset
[31:0]	RO	die_id_1	MCU 出厂 DIE_ID_1，每颗 MCU 具有特异值。	每块 MCU 具有特异值，每次复位都为固定值。

3.9.6.3 DIE_ID_2

DIE_ID_2 为 MCU 出厂 DIE_ID_2，DIE_ID 寄存器。

Offset Address: 0x0010 Total Reset Value: 0xXXXX_XXXX（X 代表出厂前编程）

Bits	Access	Name	Description	Reset
[31:0]	RO	die_id_2	MCU 出厂 DIE_ID_2，每颗 MCU 具有特异值。	每块 MCU 具有特异值，每次复位都为固定值。

3.9.6.4 DIE_ID_ 3

DIE_ID_3 为 MCU 出厂 DIE_ID_3，DIE_ID 寄存器。

Offset Address: 0x0014 Total Reset Value: 0xXXXX_XXXX（X 代表出厂前编程）

Bits	Access	Name	Description	Reset
[31:0]	RO	die_id_3	MCU 出厂 DIE_ID_3，每颗 MCU 具有特异值。	每块 MCU 具有特异值，每次复位都为固定值。

3.9.6.5 CHIP_ID

CHIP_ID 为 MCU 出厂 CHIP_ID，DIE_ID 寄存器。

Offset Address: 0x0020 Total Reset Value: 0xXXXX_XXXX（X 代表出厂前编程）

Bits	Access	Name	Description	Reset
[31:0]	RO	chip_id	MCU 出厂 CHIP_ID，每颗 MCU 具有特异值。	每块 MCU 具有特异值，每次复位都为固定值。

3.9.7 FLASH_RGN1_SHADOWS 寄存器概览

表3-10 FLASH_RGN1_SHADOWS 寄存器概览（基址是 0x1472_0000）

偏移地址	名称	描述	页码
0x1000	PRT_LEVEL_STATUS	protection level 管理标志寄存器。	PRT_LEVEL_STATUS

偏移地址	名称	描述	页码
0x1010	UART_ENABLE	UART 使能标志寄存器。	UART_ENABLE
0x1014	FUNC_JTAG_ENABLE	FUNC JTAG 使能标志寄存器。	3.9.8.1 FUNC_JTAG_ENABLE
0x1018	ISP_BOOT_ENABLE	BOOTROM 串口升级使能标志寄存器。	3.9.8.2 ISP_BOOT_ENABLE
0x101c	MAIN_RGN0_SIZE	main rgn0 容量大小配置寄存器。	3.9.8.3 MAIN_RGN0_SIZE
0x1020	INFO_RGN1_UNLOCK	info_rgn1 锁定标志寄存器。	INFO_RGN1_UNLOCK

3.9.8 FLASH_RGN1_SHADOWS 寄存器描述

本节内容描述 info_rgn1 的 shadow 寄存器。(基址是 0x1472_0000)

在 MCU 上电后，info_rgn1 的 shadow 寄存器会将部分 info_rgn1 的 Flash 数据载入，用户可直接对指定寄存器地址进行读取，得到对应 info_rgn1 区域的值。

info_rgn1 区域可以由用户进行烧写。如果需要改动 info_rgn1 shadow 寄存器的值，则需要烧写对应 Flash 区域，并对 MCU 进行复位。解复位后，shadow 寄存器的值将被刷新为烧写值。

PRT_LEVEL_STATUS

PRT_LEVEL_STATUS 为 protection level 管理标志。

Offset Address: 0x1000

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000

Bits	Access	Name	Description	Reset
[7:0]	RO	prt_level_status	protection level 管理标志。 0xFF：JTAG/SWD 接口权限开放； 其他：JTAG/SWD 接口权限受限，只能访问 FLASH 寄存器空间，读写擦 eflash main rgn1，以及读 info_rgn0/1 区域。 当 info_rgn1 处于未锁定时（info_rgn1_unlock=0xFF），protection_level 支持回退机制：通过向 FLASH 发起回退请求，依次擦除 eflash main0/1_rgn 和 info_rgn1，PRT_LEVEL_STATUS 的值将变为 0xFF。	Flash 地址 0x80_1000 烧写值，若未曾烧写，默认值为 0xFF。

UART_ENABLE

UART_ENABLE 为 UART 使能标志。

Offset Address: 0x1010

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x000_0000
[3]	RO	uart3_enable	UART3 标志。 0：禁用； 1：使能。	FLASH 地址 0x80_1010 烧写值，若未曾烧写，默认值为 0x1。
[2]	RO	uart2_enable	UART2 能标志。 0：禁用； 1：使能。	FLASH 地址 0x80_1010 烧写值，若未曾烧写，默认值为 0x1。
[1]	RO	uart1_enable	UART1 使能标志。 0：禁用； 1：使能。	FLASH 地址 0x80_1010 烧写值，若未曾烧写，默认值为 0x1。
[0]	RO	uart0_enable	UART0 能标志。	FLASH 地址 0x80_1010 烧写值，若未曾烧写，默认值

Bits	Access	Name	Description	Reset
			0：禁用； 1：使能。	为 0x1。

3.9.8.1 FUNC_JTAG_ENABLE

FUNC_JTAG_ENABLE 为 FUNC JTAG 使能标志。

Offset Address: 0x1014

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x00
[8]	RO	sysram_parity_disable	SYSRAM 奇偶校验使能标志。 0：使能； 1：禁用。	FLASH 地址 0x80_1014 烧写值，若未曾烧写，默认值为 0x1。
[7:0]	RO	func_jtag_enable	FUNC JTAG 使能标志。 0xFF：允许； 其他：禁止。 禁止：无法通过 JTAG/SWD 接口访问 MCU； 允许：可以通过 JTAG/SWD 接口访问 MCU。	FLASH 地址 0x80_1014 烧写值，若未曾烧写，默认值为 0xFF。

3.9.8.2 ISP_BOOT_ENABLE

ISP_BOOT_ENABLE 为 BOOTROM 串口升级使能标志寄存器。

Offset Address: 0x1018

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x000_0000

Bits	Access	Name	Description	Reset
[3]	RO	can_boot_enable	CAN BOOT 升级使能标志。 0：禁用，bootrom 无法通过 CAN 进行串口升级； 1：使能，bootrom 可以通过 CAN 进行串口升级。	Flash 地址 0x80_1018 烧写值，若未曾烧写，默认值为 0x1。
[2]	RO	i2c_boot_enable	I2C BOOT 升级使能标志。 0：禁用，bootrom 无法通过 I2C 进行串口升级； 1：使能，bootrom 可以通过 I2C 进行串口升级。	Flash 地址 0x80_1018 烧写值，若未曾烧写，默认值为 0x1。
[1]	RO	spi_boot_enable	SPI BOOT 升级使能标志。 0：禁用，bootrom 无法通过 SPI 进行升级； 1：使能，bootrom 可以通过 SPI 进行升级。	Flash 地址 0x80_1018 烧写值，若未曾烧写，默认值为 0x1。
[0]	RO	uart_boot_enable	UART BOOT 升级使能标志。 0：禁用，bootrom 无法通过 UART 进行升级； 1：使能，bootrom 可以通过 UART 进行升级。	Flash 地址 0x80_1018 烧写值，若未曾烧写，默认值为 0x1。

3.9.8.3 MAIN_RGN0_SIZE

MAIN_RGN0_SIZE 为 main rgn0 容量大小配置寄存器。

Offset Address: 0x101c

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x00_0000
[9:0]	RO	main_rgn0_size	main rgn0 容量大小配置，单位：Page（1KB/Page）。 main rgn 中小于等于 main_rgn0_size 的	FLASH 地址 0x80_101c 烧写值，若未曾烧写，

Bits	Access	Name	Description	Reset
			Page 属于 main rgn0，大于 main_rgn0_size 的 Page 属于 main rgn1。	默认值为 0x3FF。

INFO_RGN1_UNLOCK

INFO_RGN1_UNLOCK 为 info_rgn1 锁定标志。

Offset Address: 0x1020

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x00_0000
[7:0]	RO	info_rgn1_unlock	info_rgn1 区域锁定标志。 0xFF：未锁定; 其他：锁定。 锁定状态下，无法对 info_rgn1 区域进行擦除、编程操作，不支持 protection level 回退功能。	Flash 地址 0x80_1020 烧写值，若未曾烧写，默认值为 0xFF。

3.9.9 Protection level 回退操作寄存器概览

表3-11 Protection level 回退操作寄存器概览（基址是 0x1472_0000）

偏移地址	名称	描述	页码
0x4000	CLR_PRT_LEVEL_OP	protection level 回退启动寄存器。	CLR_PRT_LEVEL_OP
0x4004	CLR_PRT_LEVEL_RESULT	protection level 回退结果寄存器。	CLR_PRT_LEVEL_RESULT

3.9.10 Protection Level 回退操作寄存器描述

本节内容描述 protection level 回退操作的相关寄存器。(基址是 0x1472_0000)

CLR_PRT_LEVEL_OP

CLR_PRT_LEVEL_OP 为 protection level 回退启动寄存器。

Offset Address: 0x4000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	clr_prt_level_op	0xABEF_CD00：配置成该值，如果 info_rgn1_unlock 为未锁定，则发起回退 protection_level 操作。操作执行中时，寄存器将锁定为该值，不会被改写。若执行过程中发现 error，则逻辑自动将本寄存器改写为 32'h0。 0xABEF_CD10：正确启动了回退操作，且执行结果无 error，则由逻辑控制自动从 0xABEF_CD00 转为该值，并锁定，不会被改写。 其他：不可启动回退操作。	0x00000000

CLR_PRT_LEVEL_RESULT

CLR_PRT_LEVEL_RESULT 为 protection level 回退操作的结果寄存器。

Offset Address: 0x4004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12]	RO	int_raw_err_erase	擦除错误原始中断状态寄存器。 0：擦除操作 Verify 成功； 1：擦除操作 Verify 失败。	0x0
[11]	RO	int_raw_err_illegal	非法 Flash CMD 操作错误原始中断寄存器。 0：无错误；	0x0

Bits	Access	Name	Description	Reset
			1：存在 Flash CMD 操作错误，具体错误，需要查询 FLASH_STATUS 寄存器。	
[10:6]	RO	info_rgn1_err_page	清 info_rgn1 区域时，出现 ERROR 时，试图擦除的 Page 的页数。	0x00
[5]	RO	main_array_err	在清 Main Region 区域过程中发生错误。	0x0
[4]	RO	info_rgn1_err	在清 info_rgn1 区域过程中发生错误。	0x0
[3]	RO	reserved	保留。	0x0
[2]	RO	info_rgn1_unlock_err	info_rgn1 区域被 lock，无法执行回退。	0x0
[1]	RO	executing_state	正在执行 protection level 回退。	0x0
[0]	RO	execute_finish	回退 protection level 操作结束，无论是否有 error。	0x0

4 电源控制(PMC)

4.1 概述

PMC（Power Manage Controller）通过对 MCU 内部电源的管理，实现 MCU 的上/下电时序、开/关机功能和低功耗模式切换。

4.2 特性

PMC 具备以下特性：

- 上/下电时序控制
- 低功耗模式切换
- 支持 POR/PDR
- 支持定时唤醒
- WAKEUP 管脚唤醒源选择
- WAKEUP 管脚唤醒源极性可配

4.3 电源

4.3.1 概述

MCU 支持的工作电压范围为 2.4V ~ 3.63V，MCU 电源域示意如图 4-1 所示。针对特定外设，MCU 提供了几种不同的电源：

- V_{DD} = 2.4V ~ 3.63V

V_{DD} 是数字域电源，为内部调压器、Flash、I/O 管脚、LOSC 等供电，通过 VDD 管脚从外部提供。

- V_{DDA} = 2.4V~3.63V

  V_{DDA} 是模拟域电源，为模拟资源，如 ADC、DAC、ACMP、PGA、温度传感器（Tsensor）、PLL、HOSC 等供电，通过 VDDA 管脚从外部提供。

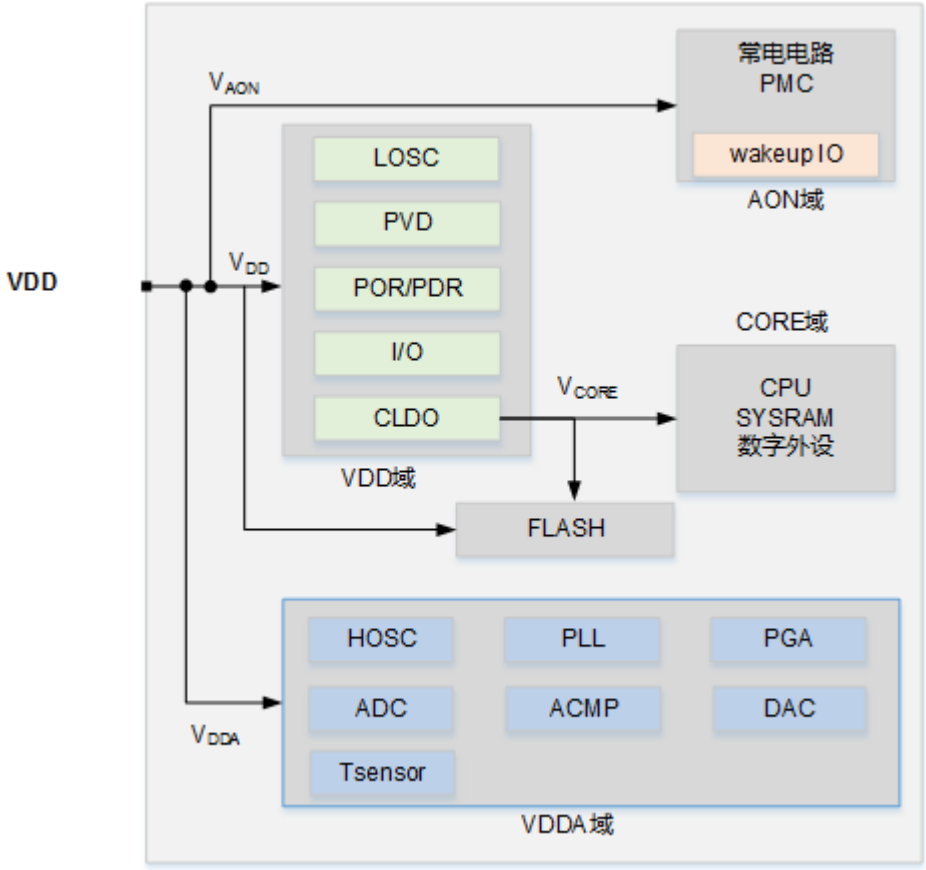
- V_{AON} = V_{DD}

  V_{AON} 是常电域电源，为常电电路、PMC、唤醒 IO 等供电，通过 VDD 管脚从外部提供。

- V_{CORE} ≈ 1.1V

  V_{CORE} 是 MCU 内部 CORE 域的数字电源，为数字外设、SYSRAM 和 Flash 供电。V_{CORE} 由嵌入式线性调压器 CLDO 提供，可配置关闭。

图4-1 MCU 电源域示意图



4.3.2 线性调压器

MCU 嵌入式集成线性调压器 CLDO，为 MCU 内部电路供电，其输出电压通常设置为 1.1V。在 MCU 上电后，会自动启动 CLDO，进入工作模式。根据用户应用模式，可以通过关闭 CLDO 进入深睡眠模式，以降低整 MCU 功耗。

4.3.3 电源监控

4.3.3.1 上电复位（POR）/掉电复位（PDR）

MCU 具有一个集成的上电复位（POR）/掉电复位（PDR），在所有功耗模式下都有效。

上电期间，POR 将使 MCU 保持复位状态，直到 V_{DD} 电源电压达到 POR 阈值。此时，将释放 MCU 复位信号并使能 CLDO，启动用户程序。在掉电期间，V_{DD} 电源电压降至 PDR 阈值时，MCU 再次被置于复位状态。

4.3.3.2 可编程电压检测器(PVD)

可以使用 PVD 监视 V_{DD} 电源电压，方法是将 V_{DD} 与预设阈值电压比较，判断电压是否超出预期。PVD 的具体使用方法请参见 9.2.9 PVD 阈值和使能控制章节。

4.4 低功耗模式

4.4.1 概述

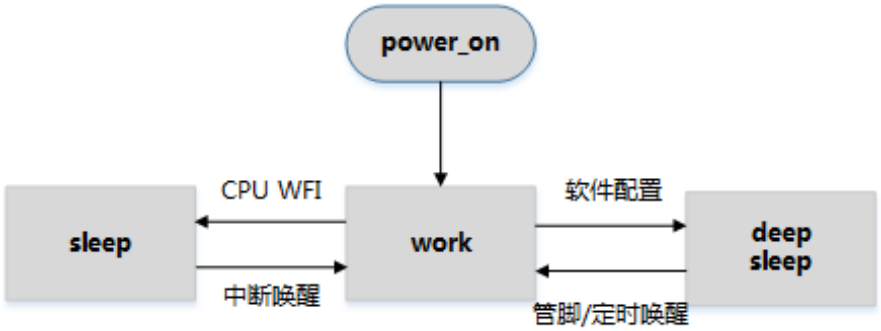
默认情况下，系统复位或上电复位后，MCU 进入工作模式。系统提供了多个低功耗模式，可在 CPU 不需要运行时节省功耗。由用户根据应用选择具体的低功耗模式，以在低功耗、短启动时间和可用唤醒源之间寻求最佳平衡。MCU 有两种低功耗模式：

- 睡眠模式：CPU 时钟关闭，但所有外设都可以运行，并在发生中断时唤醒 CPU。
- 深睡眠模式：CLDO 关闭，V_{CORE} 域断电。此时，V_{CORE} 域中的时钟都会停止，PLL、HOSC 和 XTAL 振荡器被禁止。仅 AON 域正常工作，LOSC 为 AON 域提供工作时钟。

此外，还可以通过以下方法降低工作模式下的功耗：

- 降低系统时钟频率。
- 不使用某外设时，关闭其时钟。

图4-2 低功耗模式状态图



4.4.2 工作模式

默认情况下，系统复位或上电复位后，MCU 进入工作模式，系统主时钟由 HOSC 提供。

降低系统时钟频率

在工作模式下，可以通过对 CRG 分频器配置（降低系统和外设的时钟频率）的方式减少系统功耗。在进入睡眠模式之前，也可以使用这些分频器降低外设频率。

有关详细信息，请参见 5 时钟复位生成(CRG)章节。

外设时钟门控

在工作模式下，可以关闭各外设的时钟来降低功耗。如需进一步降低睡眠模式的功耗，可在执行 WFI 指令之前关闭外设时钟。

外设时钟门控由 CRG 寄存器控制。有关详细信息，请参见 5 时钟复位生成(CRG)章节。

4.4.3 睡眠模式

睡眠模式下，CPU 停止运行，处于低功耗状态（仅保留中断唤醒功能），除此之外与工作模式相同。如需进一步降低睡眠模式的功耗，可在进入睡眠模式之前通过配置 CRG 相关寄存器降低系统时钟频率和关闭外设时钟。

睡眠模式下的 I/O 状态

在睡眠模式下，所有 I/O 管脚的状态与运行模式下相同。

进入睡眠模式

在工作模式下，当系统没有中断挂起且 CPU 执行 WFI 指令后，MCU 将进入睡眠模式。

退出睡眠模式

任意未屏蔽的外设中断都能唤醒系统。

4.4.4 深睡眠模式

深睡眠模式下，CLDO 被关闭，PLL、HOSC、XTAL 振荡器也会关闭。所有寄存器（AON 域中的寄存器除外）内容和 SYSRAM 内容都将丢失。

深睡眠模式下的 I/O 状态

深睡眠模式下，RESETN 和 WAKEUP 功能相关的管脚状态与运行模式相同。其他 I/O 管脚状态为高阻态。

进入深睡眠模式

在工作模式下，将 PMC 的 LOWPOWER_MODE[deepsleep_req]置 1，MCU 将进入深睡眠模式。

退出深睡眠模式

MCU 可以通过以下方式退出深睡眠模式：

- RESETN 管脚触发：RESETN 管脚持续低电平时间大于 50us。
- WAKEUP 事件触发：通过 WAKEUP 功能管脚触发，支持上升沿、下降沿、高电平、低电平触发方式。MCU 最大支持 3 个 WAKEUP 功能管脚，可独立配置触发方式和是否使能。详细配置方式请参见 PMC 的 WAKEUP_CTRL 寄存器描述。还可通过 PMC 的 LOWPOWER_STATUS 寄存器查询 WAKEUP 管脚状态和当前系统启动的唤醒源。
- 定时事件触发：定时事件采用 LOSC 时钟驱动的减计数器至 0 值时触发。可通过 PMC 的 WAKEUP_CTRL[cnt32k_wakeup_en]寄存器配置是否使能此唤醒功能，CNT32K_WAKE_CYC 寄存器配置计数器的周期数。

须知

禁止在对 eFlash 进行擦除和写操作时，配置系统进入深睡眠模式。否则将有可能对 eFlash 造成损坏。

4.5 PMC 寄存器概览

PMC 寄存器概览如表 4-1 所示。

表4-1 PMC 寄存器概览（基址是 0x147E_0000）

偏移地址	名称	描述	页码
0x200	LOWPOWER_MODE	低功耗模式控制寄存器。	LOWPOWER_MODE
0x204	CNT32K_WAKE_CYC	定时唤醒周期设置寄存器。	CNT32K_WAKE_CYC
0x208	WAKEUP_CTRL	deepsleep 模式唤醒控制寄存器。	WAKEUP_CTRL
0x20C	LOWPOWER_STATUS	低功耗状态查询寄存器。	LOWPOWER_STATUS
0xF00	AON_USER_REG0	AON 域用户寄存器 0。	AON_USER_REG0
0xF04	AON_USER_REG1	AON 域用户寄存器 1。	AON_USER_REG1
0xF08	AON_USER_REG2	AON 域用户寄存器 2。	AON_USER_REG2
0xF0C	AON_USER_REG3	AON 域用户寄存器 3。	AON_USER_REG3

4.6 PMC 寄存器描述

LOWPOWER_MODE

LOWPOWER_MODE 为低功耗模式控制寄存器。

Offset Address: 0x200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	WO	deepsleep_req	系统进入 deepsleep 模式请求。 对此寄存器位域写 1 操作将触发 MCU 进入 deepsleep 模式流程。	0x0
[3:0]	-	reserved	保留。	0x0

CNT32K_WAKE_CYC

CNT32K_WAKE_CYC 为定时唤醒周期设置寄存器。

Offset Address: 0x204 Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:0]	RW	cnt32k_wakeup_cycle	定时唤醒周期数（计数时钟 clk_losc）。	0x0000FFFF

WAKEUP_CTRL

WAKEUP_CTRL 为 deepsleep 模式唤醒控制寄存器。

Offset Address: 0x208 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	-	reserved	保留。	0x0000
[16]	RW	cnt32k_wakeup_en	定时唤醒使能。 0：不使能; 1：使能。	0x0
[15:12]	-	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[11]	RW	wakeup3_en	WAKEUP3 使能。 0：不使能; 1：使能。	0x0
[10]	RW	wakeup2_en	WAKEUP2 使能。 0：不使能; 1：使能。	0x0
[9]	-	reserved	保留。	0x0
[8]	RW	wakeup0_en	WAKEUP0 使能。 0：不使能; 1：使能。	0x0
[7:6]	RW	wakeup3_act_mode	WAKEUP3 有效模式选择。 00：上升沿有效; 01：下降沿有效; 10：高电平有效; 11：低电平有效。	0x0
[5:4]	RW	wakeup2_act_mode	WAKEUP2 有效模式选择。 00：上升沿有效; 01：下降沿有效; 10：高电平有效; 11：低电平有效。	0x0
[3:2]	-	reserved	保留。	0x0
[1:0]	RW	wakeup0_act_mode	WAKEUP0 有效模式选择。 00：上升沿有效; 01：下降沿有效; 10：高电平有效; 11：低电平有效。	0x0

LOWPOWER_STATUS

LOWPOWER_STATUS 为低功耗状态查询寄存器。

Offset Address: 0x20C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15]	RO	wakeup3_status	wakeup3 唤醒源状态。 0：唤醒源低电平； 1：唤醒源高电平。	0x0
[14]	RO	wakeup2_status	wakeup2 唤醒源状态。 0：唤醒源低电平； 1：唤醒源高电平。	0x0
[13]	-	reserved	保留。	0x0
[12]	RO	wakeup0_status	wakeup0 唤醒源状态。 0：唤醒源低电平； 1：唤醒源高电平。	0x0
[11:10]	-	reserved	保留。	0x0
[9]	RO	starup_from_deepsleep	从 deepsleep 状态启动标识。 0：未从 deepsleep 模式启动； 1：从 deepsleep 模式启动。	0x0
[8]	RO	starup_from_shutdown	MCU 初次上电启动标识。 0：不是初次上电启动； 1：是初次上电启动。	0x0
[7:6]	-	reserved	保留。	0x0
[5:0]	RO	wakeup_src_lock	启动唤醒源查询。 bit[5]：RESETN 管脚唤醒； bit[4]：WAKEUP3 唤醒； bit[3]：WAKEUP2 唤醒； bit[2]：保留；	0x00

Bits	Access	Name	Description	Reset
			bit[1]：WAKEUP0 唤醒； bit[0]：cnt32k 定时唤醒。	

AON_USER_REG0

AON_USER_REG0 为 AON 域用户寄存器 0。

Offset Address: 0xF00 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	aon_user_reg0	AON 电源域通用寄存器 0。	0x00000000

AON_USER_REG1

AON_USER_REG1 为 AON 域用户寄存器 1。

Offset Address: 0xF04 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	aon_user_reg1	AON 电源域通用寄存器 1。	0x00000000

AON_USER_REG2

AON_USER_REG2 为 AON 域用户寄存器 2。

Offset Address: 0xF08 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	aon_user_reg2	AON 电源域通用寄存器 2。	0x00000000

AON_USER_REG3

AON_USER_REG3 为 AON 域用户寄存器 3。

Offset Address: 0xF0C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:0]	RW	aon_user_reg3	AON 电源域通用寄存器 3。	0x00000000

5 时钟复位生成(CRG)

5.1 概述

CRG（clock & reset generator）时钟复位生成器为 MCU 各模块提供时钟和复位控制。将不同的源时钟分发到系统总线和各个外设，也对时钟进行门控管理用于低功耗控制。

5.2 复位

MCU 共有 5 种类型的复位源：电源复位、管脚复位、电压监控复位、看门狗复位、软件复位。

5.2.1 复位源描述

- 电源复位

  电源复位由以下事件触发：

  - power-on reset (POR)；
  - 从 deepsleep 模式退出。

- 管脚复位

  当处于 WORK 模式下，复位管脚 RESETN 被外部驱动为低电平且持续 50us 以上时，触发管脚复位。

- 电压监控复位

  当 PVD 模块检测到 VDD 供电电压超出设置阈值时，触发电压监控复位。该复位是否有效，可由 PERI_CRG65 寄存器配置。

- 看门狗复位

  当看门狗模块（WWDG/IWDG）未被正常刷新时，触发看门狗复位。

- 软件复位

  当对系统控制器（SYSCTRL0）的 SC_SYS_RES 寄存器进行写操作时，触发软件复位。SC_SYS_RES 寄存器有写保护机制。仅当系统控制器（SYSCTRL0）的 SC_LOCKEN[sc_cfg_lock]位域为 0 时，对 SC_SYS_RES 寄存器的写操作才有效，才能触发软件复位。

5.2.2 复位域描述

复位触发后可对 MCU 不同属性或工作域的寄存器进行复位，如表 5-1 所示。

表5-1 复位控制域

电源域	寄存器	电源复位		管脚复位	电压监控复位	看门狗复位	软件复位
		POR	deepsleep 模式退出				
AON	所有寄存器（PMC/IOCMG_AON）	●	-	-	-	-	-
CORE	SYSCTRL0.USER_POR_REG0~1	●	●	-	-	-	-
	SYSCTRL0.SC_HRST_REG0 SYSCTRL0.USER_HRST_REG0~1	●	●	●	●	●	-
	SYSCTRL0.SC_SYS_STAT[update_mode]	●	●	●	●	●	-
	其他寄存器	●	●	●	●	●	●

- 电源复位

  POR 时，会将 MCU 内部所有寄存器复位；

  从 deepsleep 模式退出时，CORE 电源域所有寄存器被复位，但 AON 电源域寄存器不受影响。

- 管脚复位、电压监控复位、看门狗复位
  - MCU 内部 CORE 电源域寄存器被复位，但不包含：系统控制器（SYSCTRL0）中 SYSCTRL0.USER_POR_REG0~1 属性的寄存器。
  - AON 电源域寄存器不受管脚复位影响。

- 软件复位

  当对系统控制器（SYSCTRL0）的 SC_SYS_RES 寄存器进行写操作时，触发软件复位。SC_SYS_RES 寄存器有写保护机制。仅当系统控制器（SYSCTRL0）的 SC_LOCKEN[sc_cfg_lock]位域为 0 时，对 SC_SYS_RES 寄存器的写操作才有效，才能触发软件复位。

  - 软件复位会将 CORE 电源域内寄存器复位，但不包含：

    系统控制器（SYSCTRL0）的 SC_SYS_STAT[update_mode]位域；

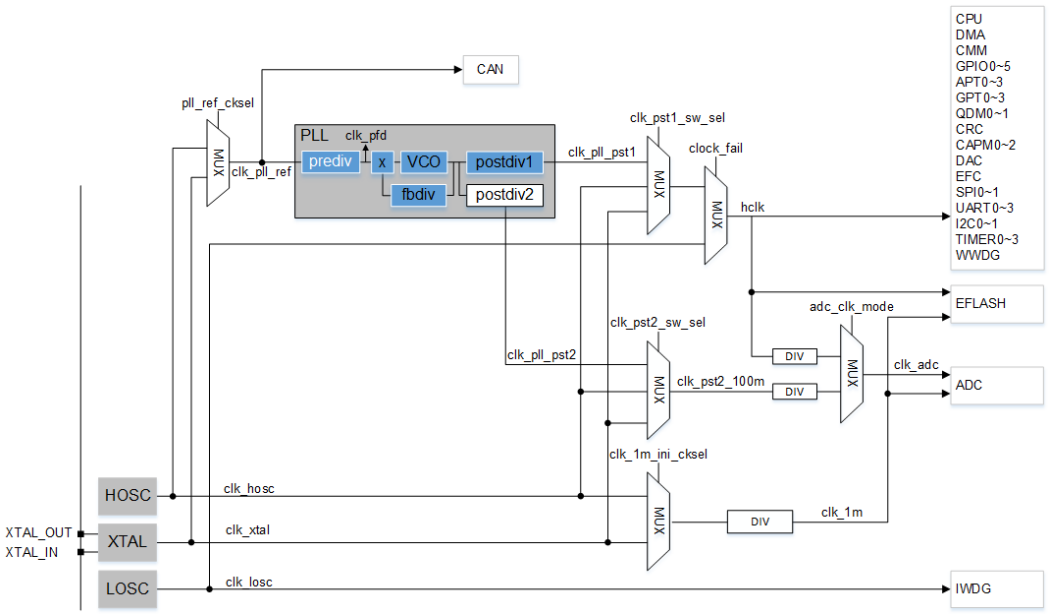
    系统控制器（SYSCTRL0）中 USER_POR_REG0~1、SC_HRST_REG0~1 和 SC_HRST_REG0 属性的寄存器。

  - AON 电源域寄存器（包含 PMC 和 IOCMG_AON）不受软件复位影响。

5.3 时钟

本章节介绍 MCU 的时钟源和时钟域，以及如何在应用中配置时钟。MCU 的时钟系统如图 5-1 所示。

图5-1 时钟系统框图



5.3.1 源时钟

源时钟用于衍生 MCU 各模块所需的时钟。MCU 共有三个源时钟：HOSC、LOSC 和 XTAL。

HOSC

HOSC 是 MCU 集成的高精度高速振荡器，提供 25MHz 时钟 clk_hosc。clk_hosc 是 MCU CORE 电源域的默认时钟。

LOSC

LOSC 是 MCU 集成的低速振荡器，提供 32kHz 时钟 clk_losc。clk_losc 是 MCU AON 电源域的默认时钟，也为 IWDG 模块提供计数时钟。

XTAL

XTAL 支持高速外部晶体或外部时钟源，提供 4MHz~30MHz 时钟 clk_xtal。

内部和外部源时钟的详细电气特性，请参见《3061M 系列数据手册》。

5.3.2 PLL

PLL 分频器

- 支持前置分频器 prediv，1/2/3/4/5/6/7/8 分频系数可配。
- 支持反馈分频器 fbdiv，6~127 范围可配。
- 支持后置分频器 postdiv1，1/2/3/4/5/6/7/8/分频系数可配。
- 支持后置分频器 postdiv2，1/2/3/4/5/6/7/8/分频系数可配。

PLL 频率范围

- PLL 输入参考时钟 clk_pll_ref，允许频率范围 $F_{clk_pll_ref}$：4MHz ~30MHz。
- PLL 内部参考时钟 clk_pfd，其频率 $F_{clk_pfd} = F_{clk_pll_ref}$ / prediv，允许频率范围：4MHz ~ 7.5MHz。
- PLL VCO 输出时钟 clk_vco，其频率 $F_{clk_vco} = F_{clk_pfd}$ * fbdiv，允许频率范围：100MHz ~ 300MHz。
- PLL 输出时钟 clk_pll_pst1，其频率 $F_{clk_pll_pst1} = F_{clk_vco}$ / postdiv1 = $F_{clk_pll_ref}$ / prediv * fbdiv / postdiv1，允许频率最大：150MHz。

- PLL 输出时钟 clk_pll_pst2，其频率 $F_{clk_pll_pst2} = F_{clk_vco} / postdiv2 = F_{clk_pll_ref} / prediv * fbdiv / postdiv2$，允许频率最大：100MHz。

PLL 配置流程

步骤 1 解锁 CRG 寄存器写保护，当系统控制器（SYSCTRL0）的 SC_LOCKEN[crg_cfg_lock]位域为 0 时，对 PLL 配置寄存器的写操作才有效。

步骤 2 关闭 PLL 和 PLL 锁定检测，通过 CRG 的 PERI_CRG4[pll_pd]位域置位为值 1 实现。

步骤 3 PLL 参考时钟 clk_pll_ref 源选择，通过配置 CRG 的 PERI_CRG0 寄存器实现。

步骤 4 PLL 内部分频器 prediv、fbdiv、postdiv 配置，通过配置 CRG 的 PERI_CRG1~PERI_CRG3 寄存器实现。各分频器频率范围请参考 PLL 频率范围描述。

步骤 5 使能 PLL 和 PLL 锁定检测，通过 CRG 的 PERI_CRG4[pll_pd]位域为 0 实现。

步骤 6 等待 PLL 锁定，通过查询 CRG 的 PERI_CRG7 或者 PERI_CRG8 寄存器实现。

----结束

5.3.3 衍生时钟

hclk

hclk 为系统主工作时钟，系统复位后默认选择 clk_hosc 为系统时钟。其时钟源有：

- clk_hosc（默认值）
- clk_xtal
- clk_pll_pst1（PLL 输出，最大 150MHz）
- clk_losc（clock_fail 模式触发时，切换到该时钟；功能详细描述请参见 6 时钟频率监控(CMM)的内容）

hclk 为 CPU、SYSRAM、高速外设以及总线 SYS_BUS 提供时钟。高速外设有：

- DMA
- CMM
- GPIO0~5
- APT0~3
- GPT0~3
- QDM0~1
- CRC
- CAPM0~2

- EFC
- SPI0~1
- UART0~3
- I2C0~1
- Timer0~3
- WWDG
- ADC 控制器
- PGA 控制器 0~1
- ACMP 控制器
- DAC 控制器
- ADCVREF 控制器

clk_pst2_100m

clk_pst2_100m 主要为 ADC 提供工作时钟，最大支持 100MHz，其时钟源有：

- clk_hosc（默认值）
- clk_xtal
- clk_pll_pst2（PLL 输出时钟，最大 100MHz）

clk_adc

clk_adc 为 ADC 部分工作时钟，最大支持 100MHz。其时钟源有：

- clk_pst2_100m（默认值）
- hclk

clk_1m

clk_1m 为 ADC 和 eFlash 做计数时钟使用，最大支持 1MHz，其时钟源有：

- clk_hosc（默认值）
- clk_xtal

5.3.4 输出时钟

可以通过 TEST_CLK 信号将 MCU 内部时钟提供给外部使用。TEST_CLK 时钟源可通过 PERI_CRG639[test_clk_sel]位域选择，支持的时钟有：

- clk_pfd
- clk_hosc
- clk_losc
- clk_xtal

- clk_adc/4
- hclk/6

5.3.5 寄存器写保护

为避免非法配置导致系统异常，CRG 部分寄存器支持写保护功能。该功能由系统控制器（SYSCTRL0）的 SC_LOCKEN[crg_cfg_lock]位域控制。当 crg_cfg_lock 位域值为 1 时，对 CRG 写保护寄存器进行写操作无效。CRG 中支持写保护的寄存器为 PERI_CRG0 ~ 4、PERI_CRG320 和 HOSC_PD，详见寄存器描述。

5.4 CRG_PERIAPBIF 寄存器概览

CRG_PERIAPBIF 寄存器概览如表 5-2 所示。

表5-2 CRG_PERIAPBIF 寄存器概览（基址是 0x1000_0000）

偏移地址	名称	描述	页码
0x0000	PERI_CRG0	PLL 配置寄存器 0	PERI_CRG0
0x0004	PERI_CRG1	PLL 配置寄存器 1	PERI_CRG1
0x0008	PERI_CRG2	PLL 配置寄存器 2	PERI_CRG2
0x000C	PERI_CRG3	PLL 配置寄存器 3	PERI_CRG3
0x0010	PERI_CRG4	PLL 配置寄存器 4	PERI_CRG4
0x001C	PERI_CRG7	PLL 输出时钟频率稳定标志寄存器	PERI_CRG7
0x0020	PERI_CRG8	PLL 输出 lock 去毛刺信号寄存器	PERI_CRG8
0x0100	PERI_CRG64	系统时钟控制寄存器	PERI_CRG64
0x0104	PERI_CRG65	PVD 复位控制寄存器	PERI_CRG65
0x0108	PERI_CRG66	CLK_1M 时钟源头选择控制寄存器	PERI_CRG66
0x010C	PERI_CRG67	CLK_1M 时钟分频控制寄存器	PERI_CRG67

偏移地址	名称	描述	页码
0x0140	PERI_CRG80	UART0 时钟及软复位控制寄存器	PERI_CRG80
0x0144	PERI_CRG81	UART1 时钟及软复位控制寄存器	PERI_CRG81
0x0148	PERI_CRG82	UART2 时钟及软复位控制寄存器	PERI_CRG82
0x014C	PERI_CRG83	UART3 时钟及软复位控制寄存器	PERI_CRG83
0x0180	PERI_CRG96	SPI0 时钟及软复位控制寄存器	PERI_CRG96
0x0184	PERI_CRG97	SPI1 时钟及软复位控制寄存器	PERI_CRG97
0x01C0	PERI_CRG112	I2C0 时钟及软复位控制寄存器	PERI_CRG112
0x01C4	PERI_CRG113	I2C1 时钟及软复位控制寄存器	PERI_CRG113
0x0200	PERI_CRG128	WWDG 时钟及软复位控制寄存器	PERI_CRG128
0x0240	PERI_CRG144	Timer0 时钟及软复位控制寄存器	PERI_CRG144
0x0244	PERI_CRG145	Timer1 时钟及软复位控制寄存器	PERI_CRG145
0x0248	PERI_CRG146	Timer2 时钟及软复位控制寄存器	PERI_CRG146
0x024C	PERI_CRG147	Timer3 时钟及软复位控制寄存器	PERI_CRG147
0x0280	PERI_CRG160	CAPM0 时钟及软复位控制寄存器	PERI_CRG160
0x0284	PERI_CRG161	CAPM1 时钟及软复位控制寄存器	PERI_CRG161
0x0288	PERI_CRG162	CAPM2 时钟及软复位控制寄存器	PERI_CRG16

偏移地址	名称	描述	页码
			2
0x02C0	PERI_CRG176	CAN 时钟及软复位控制寄存器	PERI_CRG176
0x0300	PERI_CRG192	DMA 时钟及软复位控制寄存器	PERI_CRG192
0x0340	PERI_CRG208	CMM 时钟及软复位控制寄存器	PERI_CRG208
0x0380	PERI_CRG224	CRC 时钟及软复位控制寄存器	PERI_CRG224
0x03C0	PERI_CRG240	IWDG 时钟及软复位控制寄存器	PERI_CRG240
0x0400	PERI_CRG256	APT0 时钟及软复位控制寄存器	PERI_CRG256
0x0404	PERI_CRG257	APT1 时钟及软复位控制寄存器	PERI_CRG257
0x0408	PERI_CRG258	APT2 时钟及软复位控制寄存器	PERI_CRG258
0x040C	PERI_CRG259	APT3 时钟及软复位控制寄存器	PERI_CRG259
0x0440	PERI_CRG272	GPT0 时钟及软复位控制寄存器	PERI_CRG272
0x0444	PERI_CRG273	GPT1 时钟及软复位控制寄存器	PERI_CRG273
0x0448	PERI_CRG274	GPT2 时钟及软复位控制寄存器	PERI_CRG274
0x044C	PERI_CRG275	GPT3 时钟及软复位控制寄存器	PERI_CRG275
0x0480	PERI_CRG288	GPIO0 时钟及软复位控制寄存器	PERI_CRG28

偏移地址	名称	描述	页码
			8
0x0484	PERI_CRG289	GPIO1 时钟及软复位控制寄存器	PERI_CRG289
0x0488	PERI_CRG290	GPIO2 时钟及软复位控制寄存器	PERI_CRG290
0x048C	PERI_CRG291	GPIO3 时钟及软复位控制寄存器	PERI_CRG291
0x0490	PERI_CRG292	GPIO4 时钟及软复位控制寄存器	PERI_CRG292
0x0494	PERI_CRG293	GPIO5 时钟及软复位控制寄存器	PERI_CRG293
0x04C0	PERI_CRG304	QDM0 时钟及软复位控制寄存器	PERI_CRG304
0x04C4	PERI_CRG305	QDM1 时钟及软复位控制寄存器	PERI_CRG305
0x0500	PERI_CRG320	Flash 时钟控制寄存器	PERI_CRG320
0x09FC	PERI_CRG639	TEST 时钟选择控制寄存器	PERI_CRG639
0x0A00	PERI_CRG640	ADC 分频时钟配置寄存器	PERI_CRG640
0x0A04	PERI_CRG641	ADC 时钟及软复位控制寄存器	PERI_CRG641
0x0A50	PERI_CRG660	ANA 时钟控制寄存器	PERI_CRG660
0x0A60	PERI_CRG664	VREF 软复位请求寄存器	PERI_CRG664
0x0A70	PERI_CRG668	ACMP 软复位请求寄存器	PERI_CRG66

偏移地址	名称	描述	页码
			8
0x0A80	PERI_CRG672	DAC 软复位请求寄存器	PERI_CRG672
0x0A90	PERI_CRG676	PGA0 软复位请求寄存器	PERI_CRG676
0x0A94	PERI_CRG677	PGA1 软复位请求寄存器	PERI_CRG677
0x0F0C	HOSC_PD	HOSC 配置寄存器	HOSC_PD
0x0F10	HOSC_LOCK	HOSC 输出时钟频率稳定标志寄存器	HOSC_LOCK

5.5 CRG_PERIAPBIF 寄存器描述

PERI_CRG0

PERI_CRG0 为 PLL 配置寄存器 0。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	pll_ref_cksel	PLL 输入参考时钟 clk_pll_ref 时钟源选择信号。 0：选择 clk_hosc； 1：选择 clk_xtal。	0x0

PERI_CRG1

PERI_CRG1 为 PLL 配置寄存器 1。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0004 Total Reset Value: 0x0000_0003

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3:0]	RW	pll_prediv	PLL 内部前置分频器分频比。 0x0：1 分频； 0x1：2 分频； 0x2：3 分频； 0x3：4 分频； 0x4：5 分频； 0x5：6 分频； 0x6：7 分频； 0x7：8 分频； 其他：未使用。	0x3

PERI_CRG2

PERI_CRG2 为 PLL 配置寄存器 2。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0008 Total Reset Value: 0x0000_0030

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:0]	RW	pll_fbdiv	PLL 内部反馈分频器分频比。 0x00 ~ 0x06：6 分频； 0x07 ~ 0x7F：7~127 分频； 其他：未使用。	0x30

PERI_CRG3

PERI_CRG3 为 PLL 配置寄存器 3。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x000C Total Reset Value: 0x0000_0021

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:4]	RW	pll_postdiv2	PLL 内部后置分频器分频比。 0x0：1 分频； 0x1：2 分频； 0x2：3 分频； 0x3：4 分频； 0x4：5 分频； 0x5：6 分频； 0x6：7 分频； 0x7：8 分频； 其他：未使用。	0x2
[3:0]	RW	pll_postdiv1	PLL 内部后置分频器分频比。 0x0：1 分频；	0x1

Bits	Access	Name	Description	Reset
			0x1：2 分频； 0x2：3 分频； 0x3：4 分频； 0x4：5 分频； 0x5：6 分频； 0x6：7 分频； 0x7：8 分频； 其他：未使用。	

PERI_CRG4

PERI_CRG4 为 PLL 配置寄存器 4。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0010 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	pll_pd	PLL 使能控制。 0：PLL 上电使能； 1：PLL 下电关闭。	0x1

PERI_CRG7

PERI_CRG7 为 PLL 输出时钟频率稳定标志寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RO	pll_lock	PLL 时钟稳定标志位信号。 0：PLL 输出频率未稳定； 1：PLL 输出频率稳定。	0x0

PERI_CRG8

PERI_CRG8 为 pll 输出 lock 去毛刺信号寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RO	pll_lock_deglitch	pll lock 硬件滤毛刺之后信号。	0x0

PERI_CRG64

PERI_CRG64 为系统时钟控制寄存器。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5:4]	RW	clk_pst2_sw_sel	clk_pst2_100m 时钟选择信号。 00：选择 clk_hosc； 01：选择 clk_xtal； 10：选择 clk_pll_pst2； 11：保留。	0x0

Bits	Access	Name	Description	Reset
[3:2]	RO	reserved	保留。	0x0
[1:0]	RW	clk_pst1_sw_sel	hclk 时钟选择信号。 00：选择 clk_hosc； 01：选择 clk_xtal； 10：选择 clk_pll_pst1； 11：保留。	0x0

PERI_CRG65

PERI_CRG65 为 PVD 复位控制寄存器。

Offset Address: 0x0104 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	pvd_rst_enable	pvd 复位使能。 0：不使能； 1：使能。	0x0

PERI_CRG66

PERI_CRG66 为 CLK_1M 时钟源头选择控制寄存器。

Offset Address: 0x0108 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	clk_1m_ini_cksel	1MHz 时钟和 clk_bg_chopper 时钟的源头时钟选择信号。 0：选择 clk_hosc，默认； 1：选择 clk_xtal。	0x0

PERI_CRG67

PERI_CRG67 为 CLK_1M 时钟分频控制寄存器。

Offset Address: 0x010C Total Reset Value: 0x0000_0029

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5:0]	RW	clk_1m_div	clk_1m 时钟分频参数，配置范围为[0~63] 输出时钟频率为输入时钟频率/(clk_1m_div + 1)。 默认 42 分频，配'd29。	0x29

PERI_CRG80

PERI_CRG80 为 UART0 时钟及软复位控制寄存器。

Offset Address: 0x0140 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	uart0_srst_req	UART0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	uart0_cken	UART0 时钟使能。 0：关闭; 1：打开。	0x0

PERI_CRG81

PERI_CRG81 为 UART1 时钟及软复位控制寄存器。

Offset Address: 0x0144 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	uart1_srst_req	UART1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	uart1_cken	UART1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG82

PERI_CRG82 为 UART2 时钟及软复位控制寄存器。

Offset Address: 0x0148 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	uart2_srst_req	UART2 软复位请求: 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	uart2_cken	UART2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG83

PERI_CRG83 为 UART3 时钟及软复位控制寄存器。

Offset Address: 0x014C Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	uart3_srst_req	UART3 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	uart3_cken	UART3 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG96

PERI_CRG96 为 SPI0 时钟及软复位控制寄存器。

Offset Address: 0x0180 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	spi0_srst_req	SPI0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	spi0_cken	SPI0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG97

PERI_CRG97 为 SPI1 时钟及软复位控制寄存器。

Offset Address: 0x0184 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	spi1_srst_req	SPI1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	spi1_cken	SPI1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG112

PERI_CRG112 为 I2C0 时钟及软复位控制寄存器。

Offset Address: 0x01C0 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	i2c0_srst_req	I2C0 软复位请求: 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	i2c0_cken	I2C0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG113

PERI_CRG113 为 I2C1 时钟及软复位控制寄存器。

Offset Address: 0x01C4 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	i2c1_srst_req	I2C1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	i2c1_cken	I2C1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG128

PERI_CRG128 为 WWDG 时钟及软复位控制寄存器。

Offset Address: 0x0200 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x00
[16]	RW	wwdg_srst_req	WWDG 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	wwdg_cken	WWDG 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG144

PERI_CRG144 为 Timer0 时钟及软复位控制寄存器。

Offset Address: 0x0240 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x00
[16]	RW	timer0_srst_req	Timer0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	timer0_cken	Timer0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG145

PERI_CRG145 为 Timer1 时钟及软复位控制寄存器。

Offset Address: 0x0244 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x00
[16]	RW	timer1_srst_req	Timer1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	timer1_cken	Timer1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG146

PERI_CRG146 为 Timer2 时钟及软复位控制寄存器。

Offset Address: 0x0248 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x00
[16]	RW	timer2_srst_req	Timer2 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	timer2_cken	Timer2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG147

PERI_CRG147 为 Timer3 时钟及软复位控制寄存器。

Offset Address: 0x024C Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x00
[16]	RW	timer3_srst_req	Timer3 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	timer3_cken	Timer3 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG160

PERI_CRG160 为 CAPM0 时钟及软复位控制寄存器。

Offset Address: 0x0280 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	capm0_srst_req	CAPM0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	capm0_cken	CAPM0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG161

PERI_CRG161 为 CAPM1 时钟及软复位控制寄存器。

Offset Address: 0x0284 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	capm1_srst_req	CAPM1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	capm1_cken	CAPM1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG162

PERI_CRG162 为 CAPM2 时钟及软复位控制寄存器。

Offset Address: 0x0288 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	capm2_srst_req	CAPM2 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	capm2_cken	CAPM2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG176

PERI_CRG176 为 CAN 时钟及软复位控制寄存器。

Offset Address: 0x02C0 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	can_srst_req	CAN 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	can_cken	CAN 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG192

PERI_CRG192 为 DMA 时钟及软复位控制寄存器。

Offset Address: 0x0300 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	dma_srst_req	DMA 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	dma_cken	DMA 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG208

PERI_CRG208 为 CMM 时钟及软复位控制寄存器。

Offset Address: 0x0340 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	cmm_srst_req	CMM 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	cmm_cken	CMM 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG224

PERI_CRG224 为 CRC 时钟及软复位控制寄存器。

Offset Address: 0x0380 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	crc_srst_req	CRC 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	crc_cken	CRC 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG240

PERI_CRG240 为 IWDG 时钟及软复位控制寄存器。

Offset Address: 0x03C0 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	iwdg_srst_req	IWDG 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	iwdg_cken	IWDG 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG256

PERI_CRG256 为 APT0 时钟及软复位控制寄存器。

Offset Address: 0x0400 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	apt0_srst_req	APT0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	apt0_cken	APT0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG257

PERI_CRG257 为 APT1 时钟及软复位控制寄存器。

Offset Address: 0x0404 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	apt1_srst_req	APT1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	apt1_cken	APT1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG258

PERI_CRG258 为 APT2 时钟及软复位控制寄存器。

Offset Address: 0x0408 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	apt2_srst_req	APT2 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	apt2_cken	APT2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG259

PERI_CRG259 为 APT3 时钟及软复位控制寄存器。

Offset Address: 0x040C Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	apt3_srst_req	APT3 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	apt3_cken	APT3 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG272

PERI_CRG272 为 GPT0 时钟及软复位控制寄存器。

Offset Address: 0x0440 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpt0_srst_req	GPT0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpt0_cken	GPT0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG273

PERI_CRG273 为 GPT1 时钟及软复位控制寄存器。

Offset Address: 0x0444 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpt1_srst_req	GPT1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpt1_cken	GPT1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG274

PERI_CRG274 为 GPT2 时钟及软复位控制寄存器。

Offset Address: 0x0448 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpt2_srst_req	GPT2 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpt2_cken	GPT2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG275

PERI_CRG275 为 GPT3 时钟及软复位控制寄存器。

Offset Address: 0x044C Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpt3_srst_req	GPT3 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpt3_cken	GPT3 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG288

PERI_CRG288 为 gpio0 时钟及软复位控制寄存器。

Offset Address: 0x0480 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio0_srst_req	GPIO0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio0_cken	GPIO0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG289

PERI_CRG289 为 GPIO1 时钟及软复位控制寄存器。

Offset Address: 0x0484 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio1_srst_req	GPIO1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio1_cken	GPIO1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG290

PERI_CRG290 为 GPIO2 时钟及软复位控制寄存器。

Offset Address: 0x0488 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio2_srst_req	GPIO2 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio2_cken	GPIO2 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG291

PERI_CRG291 为 GPIO3 时钟及软复位控制寄存器。

Offset Address: 0x048C Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio3_srst_req	GPIO3 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio3_cken	GPIO3 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG292

PERI_CRG292 为 GPIO4 时钟及软复位控制寄存器。

Offset Address: 0x0490 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio4_srst_req	GPIO4 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio4_cken	GPIO4 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG293

PERI_CRG293 为 GPIO5 时钟及软复位控制寄存器。

Offset Address: 0x0494 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	gpio5_srst_req	GPIO5 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	gpio5_cken	GPIO5 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG304

PERI_CRG304 为 QDM0 时钟及软复位控制寄存器。

Offset Address: 0x04C0 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	qdm0_srst_req	QDM0 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	qdm0_cken	QDM0 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG305

PERI_CRG305 为 QDM1 时钟及软复位控制寄存器。

Offset Address: 0x04C4 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	qdm1_srst_req	QDM1 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	qdm1_cken	QDM1 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG320

PERI_CRG320 为 Flash 时钟控制寄存器。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0500 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	eflash_cken	EFlash 时钟使能。 0：关断； 1：打开。	0x1

PERI_CRG639

PERI_CRG639 为 TEST 时钟选择控制寄存器。

Offset Address: 0x09FC Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:19]	RO	reserved	保留。	0x0000
[18:16]	RW	test_clk_sel	TEST_CLK 时钟选择。 000：pll_pfd； 001：clk_hosc； 010：clk_losc； 011：clk_xtal； 101：clk_adc/4； 110：hclk/6； 其他：保留	0x0
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	test_clk_en	test_clk_en 时钟观测使能信号。 0：不使能； 1：使能。	0x1

PERI_CRG640

PERI_CRG640 为 ADC 分频时钟配置寄存器。

Offset Address: 0x0A00 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	RW	reserved	保留。	0x000000
[9:8]	RW	clk_adc_div1	clk_adc 选择 clk_pst2_100m 通路的时钟分频参数，配置范围[0~3]。 输出时钟频率为 clk_pst2_100m/(clk_adc_div1 + 1)。 默认不分频，配 0。	0x0
[7:2]	RO	reserved	保留。	0x00
[1:0]	RW	clk_adc_div0	clk_adc 选择 hclk 通路的时钟分频参数，配置范围[0~3]。 输出时钟频率为 hclk/(clk_adc_div0 + 1)。 默认不分频，配 0。	0x0

PERI_CRG641

PERI_CRG641 为 ADC 时钟及软复位控制寄存器。

Offset Address: 0x0A04 Total Reset Value: 0x0101_0000

Bits	Access	Name	Description	Reset
[31:25]	RO	reserved	保留。	0x00
[24]	RW	adc_clk_mode	ADC 时钟源选择信号。 0：clk_pst2_100m 经过分频后的时钟； 1：hclk 时钟经过分频后的时钟。	0x1
[23:17]	RO	reserved	保留。	0x00
[16]	RW	adc_srst_req	ADC 软复位请求。 0：不复位； 1：复位。	0x1

Bits	Access	Name	Description	Reset
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	adc_cken	ADC 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG660

PERI_CRG660 为 ANA 时钟控制寄存器。

Offset Address: 0x0A50 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	ana_srst_req	ANA 软复位请求。 0：不复位; 1：复位。	0x1
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	clk_ana_cken	ANA 时钟使能。 0：关断; 1：打开。	0x0

PERI_CRG664

PERI_CRG664 为 VREF 软复位请求。

Offset Address: 0x0A60 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	vref_srst_req	VREF 软复位请求。 0：不复位; 1：复位。	0x1

Bits	Access	Name	Description	Reset
[15:0]	RO	reserved	保留。	0x0000

PERI_CRG668

PERI_CRG668 为 ACMP 软复位请求。

Offset Address: 0x0A70 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	acmp_srst_req	ACMP 软复位请求。 0：不复位; 1：复位。	0x1
[15:0]	RO	reserved	保留。	0x0000

PERI_CRG672

PERI_CRG672 为 DAC 软复位请求。

Offset Address: 0x0A80 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	dac_srst_req	DAC 软复位请求。 0：不复位; 1：复位。	0x1
[15:0]	RO	reserved	保留。	0x0000

PERI_CRG676

PERI_CRG676 为 PGA0 软复位请求。

Offset Address: 0x0A90 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	pga0_srst_req	PGA0 软复位请求。 0：不复位; 1：复位。	0x1
[15:0]	RO	reserved	保留。	0x0000

PERI_CRG677

PERI_CRG677 为 PGA1 软复位请求。

Offset Address: 0x0A94 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	pga1_srst_req	PGA1 软复位请求。 0：不复位; 1：复位。	0x1
[15:0]	RO	reserved	保留。	0x0000

HOSC_PD

HOSC_PD 为 HOSC 配置寄存器。

须知

该寄存器可以被寄存器 SC_LOCKEN[crg_cfg_lock]写保护控制，只有禁用写保护时，对这个寄存器的写操作才有效。

Offset Address: 0x0F0C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000

Bits	Access	Name	Description	Reset
[0]	RW	hosc_pd	HOSC 使能信号。 0：HOSC 上电使能； 1：HOSC 下电关闭。	0x0

HOSC_LOCK

HOSC_LOCK 为 HOSC LOCK 信号观测。

Offset Address: 0x0F10 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RO	hosc_lock	HOSC 时钟稳定标志位信号。 0：HOSC 输出频率未稳定； 1：HOSC 输出频率稳定。	0x0

6 时钟频率监控(CMM)

6.1 概述

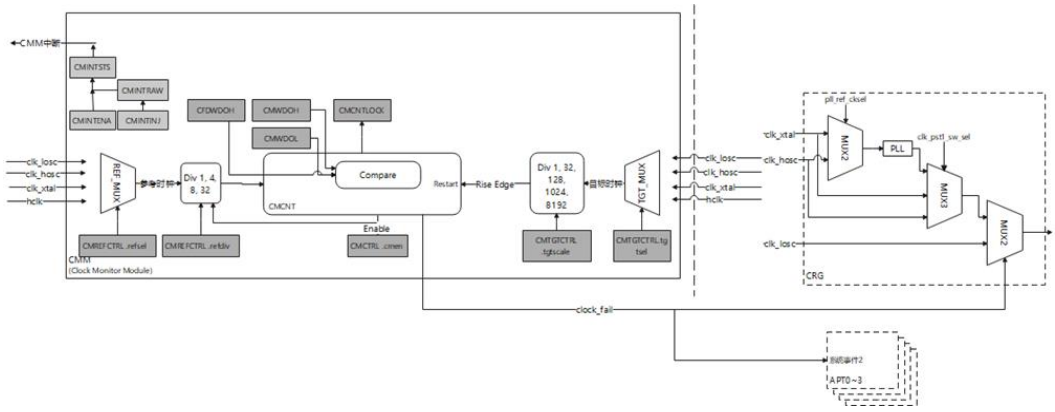
时钟频率监控模块 CMM（Clock Monitor Module）使用参考时钟来检测目标时钟的频率和目标时钟是否失效。

6.2 特性

- 支持参考时钟 4 选 1。
- 支持参考时钟最大 32 分频。
- 支持目标时钟 4 选 1。
- 支持目标时钟最大 8192 分频。
- 支持检测目标时钟的频率。
- 支持检测目标时钟是否失效（CFD 功能）。
- 支持目标时钟失效自动保护，产生系统事件输出到 APT 模块。

6.3 功能描述

图6-1 CMM 功能模块图



CMM 的功能是检测时钟频率是否偏移，检测时钟是否失效（CFD 功能）。

CMM 使用参考时钟（分频后）对目标时钟（分频后）计数。计数器 CMCNT，根据参考时钟（分频后）进行增计数。在每个目标时钟（分频后）的时钟边沿，保存此时计数值到 CMCNTLOCK 中，并清零计数器 CMCNT，表示本次检测结束。如此循环往复，直到检测到时钟失效，计数冻结。

须知

参考时钟（分频后）频率必须 ≥ 2×目标时钟（分频后）频率。

6.3.1 参考时钟和目标时钟

6.3.1.1 参考时钟选择和分频

参考时钟是指频率相对稳定的时钟。CMM 模块的时钟频率检测和时钟失效检测功能，依赖于参考时钟的稳定。

软件配置 CMREFCTRL[refsel]，可以从 clk_losc，clk_hosc，clk_xtal 和 hclk 中选 1 个时钟，作为 CMM 模块的参考时钟源。

软件配置 CMREFCTRL[refdiv]，可以对参考时钟源进行分频。

CMCNT 计数器按照参考时钟（分频后）的频率递增。

6.3.1.2 目标时钟选择和分频

目标时钟是被测时钟，在 CMM 模块的时钟频率检测和时钟失效检测功能中，被测试的时钟就是目标时钟。

软件配置 CMTGTCTRL[tgtsel]，可以从 clk_losc，clk_hosc，clk_xtal 和 hclk 中选 1 个时钟，作为 CMM 模块的目标时钟源。

软件配置 CMTGTCTRL[tgtscale]，可以将目标时钟源进行分频。

CMCNT 计数器，在（分频后的）目标时钟的上升沿处，锁存计数值到 CMCNTLOCK 寄存器中。锁存后，清零 CMCNT 计数器。

6.3.2 时钟频率检测和时钟失效检测

软件配置 CMCTRL[cmen]为 1，可以使能 CMM 模块。CMM 模块使能后，CMCNT 计数器开始计数，按照参考时钟（分频后）的频率递增。在目标时钟（分频后）的上升沿处，锁存计数值到 CMCNTLOCK 寄存器中。

- 如果锁存值 CMCNTLOCK 小于 CMWDOL[cmwdol]，或者大于 CMWDOH[cmwdoh]，那么认为目标时钟频率错误，产生 freq_err_raw 原始中断。
- 如果锁存值 CMCNTLOCK 大于 CMWDOH[cfdwdoh]，那么认为目标时钟失效，产生 clk_fail_raw 原始中断，输出 clock_fail 信号给 CRG 做目标时钟失效的自动保护，同时产生系统事件输出到 APT 模块。

须知

配置 CMWDOH 寄存器时，建议 CMWDOH[cfdwdoh]大于 CMWDOH[cmwdoh]。

注入频率失效中断，只上报频率失效中断，不会输出 clock_fail 信号，CRG 不会对目标时钟做自动保护，APT 模块不会接收到系统事件。

CMM 模块的时钟频率检测和时钟失效检测功能不支持的场景：参考时钟和目标时钟选择了同一个时钟源。即使配置不同的分频，CMM 模块也不支持这种情况下的检测。故参考时钟和目标时钟应选择不同的时钟源。

6.3.3 中断描述

CMM 一共有 4 种中断状态，分别是检测结束中断、时钟失效中断、频率错误中断、计数溢出中断。

表6-1 CMM 中断

CMM 中断状态	原始中断寄存器	中断使能寄存器	中断注入寄存器	中断上报条件
检测结束中断 chk_end_int	CMINTRAW[chk_end_raw]	CMINTENA[chk_end_en]	CMINTINJ[chk_end_inj]	每次检测结束时。
时钟失效中断 clk_fail_int	CMINTRAW[clk_fail_raw]	CMINTENA[clk_fail_en]	CMINTINJ[clk_fail_inj]	每次检测结束时，CMCNTLOCK > CMWDOH[cfdwdoh]。
频率错误中断 freq_err_int	CMINTRAW[freq_err_raw]	CMINTENA[freq_err_en]	CMINTINJ[freq_err_inj]	每次检测结束时，CMCNTLOCK < CMWDOL[cmwdol]或者 CMCNTLOCK > CMWDOH[cmwdoh]。
计数溢出中断 cnt_ovf_int	CMINTRAW[cnt_ovf_raw]	CMINTENA[cnt_ovf_int]	CMINTINJ[cnt_ovf_inj]	CMCNT 计数溢出时。

- 每次检测结束时，CMM 上报检测结束中断状态 chk_end_int。CMM 继续下一次检测。中断状态 chk_end_int = chk_end_raw & chk_end_en，即只有使能了 chk_end_en，才能上报检测结束中断。软件配置 chk_end_inj 为 1，可以向 chk_end_raw 写入 bit1。软件对 chk_end_raw 写 1，清除 chk_end_raw 原始中断。
- 每次检测结束时，如果 CMCNTLOCK > CMWDOH[cfdwdoh]，CMM 上报时钟失效中断状态 clk_fail_int。CMCNT 被冻结。中断状态 clk_fail_int = clk_fail_raw & clk_fail_en，即只有使能了 clk_fail_en，才能上报时钟失效中断。软件配置 clk_fail_inj 为 1，可以向 clk_fail_raw 写入 bit1。软件对 clk_fail_raw 写 1，清除 clk_fail_raw 原始中断。清除后，CMCNT 清零并重新开始计数。
- 每次检测结束时，如果 CMCNTLOCK < CMWDOL[cmwdol]或者 CMCNTLOCK > CMWDOH[cmwdoh]，CMM 上报频率错误中断状态 freq_err_int。CMM 继续下一次检测。中断状态 freq_err_int = freq_err_raw & freq_err_en，即只有使能了 freq_err_en，才能上报频率错误中断。软件配置 freq_err_inj 为 1，可以向 freq_err_raw 写入 bit1。软件对 freq_err_raw 写 1，清除 freq_err_raw 原始中断。清除频率错误中断后，需要配置 CMCTRL[cmen]为 0 来关闭 CMM，再重新配置 CMCTRL[cmen]为 1 来使能 CMM，才能恢复 CMM 的工作。
- 当计数器溢出时，CMM 上报计数溢出中断状态 cnt_ovf_int。CMM 计数器复位至 0，并且继续计数。中断状态 cnt_ovf _int = cnt_ovf_raw & cnt_ovf_en，即只有使能了 cnt_ovf_en，才能上报计数溢出中断。软件配置 cnt_ovf_inj 为 1，可以向 cnt_ovf_raw 写入 bit1。软件对 cnt_ovf_raw 写 1，清除 cnt_ovf_raw 原始中断。

须知

用户通过配置 CMCTRL[cmen]为 1，可以使能 CMM。在此之后，检查结束，频率错误或者计数溢出中断，都不能使 CMM 停止工作（即 CMCNT 停止计数）。而当目标时钟失效，会使 CMCNT 冻结计数，CMM 停止工作，软件清除时钟失效中断后，CMM 尝试重新开始工作。用户通过配置 CMCTRL[cmen]为 0，可以关闭 CMM。

请参见 6.6 CMM 寄存器描述获得每个位域的详细信息。

6.4 典型配置

6.4.1 初始化的流程

步骤 1 关闭 CMM 模块使能（CMCTRL[cmen]）。

步骤 2 等待 200us。

步骤 3 配置目标时钟源选择（CMTGTCTRL[tgtsel]）。

步骤 4 配置目标时钟分频系数（CMTGTCTRL[tgtscale]）。

步骤 5 配置参考时钟源选择（CMREFCTRL[refsel]），不能与目标时钟源选择一致。

步骤 6 配置参考时钟分频系数（CMREFCTRL[refdiv]）。

步骤 7 配置时钟失效检查窗口上限（CMWDOH[cfdwdoh]）。建议 cfdwdoh 大于 cmwdoh。

步骤 8 配置频率错误检查窗口上下限（CMWDOL[cmwdol]，CMWDOH[cmwdoh]）。

步骤 9 若软件使用中断方式：配置中断使能（CMINTENA）。

步骤 10 配置时钟失效 clock_fail 信号使能（CMCTRL[cfen]）。

步骤 11 打开 CMM 模块使能（CMCTRL[cmen]）。

----结束

须知

参考时钟（分频后）频率必须 ≥ 2×目标时钟（分频后）频率。

6.4.2 更新配置的流程

步骤 1 关闭 CMM 模块使能（CMCTRL[cmen]）。

步骤 2 等待 200us。

步骤 3 配置目标时钟源选择（CMTGTCTRL[tgtsel]）。

步骤 4 配置目标时钟分频系数（CMTGTCTRL[tgtscale]）。

步骤 5 配置参考时钟源选择（CMREFCTRL[refsel]），不能与目标时钟源选择一致。

步骤 6 配置参考时钟分频系数（CMREFCTRL[refdiv]）。

步骤 7 配置时钟失效检查窗口上限（CMWDOH[cfdwdoh]）。建议 cfdwdoh 大于 cmwdoh。

步骤 8 配置检查窗口上下限（CMWDOL，CMWDOH）。

步骤 9 若软件使用中断方式：配置中断使能（CMINTENA）。

步骤 10 配置时钟失效 clock_fail 信号使能（CMCTRL[cfen]）。

步骤 11 打开 CMM 模块使能（CMCTRL[cmen]）。

----结束

须知

步骤 2 中的等待时间可根据参考时钟频率进行计算，等待时间 > 6×参考时钟周期。举个例子：参考时钟为 CLK_LOSC 时，等待时间 > 6×CLK_LOSC 周期，可以配置为 200us。在任意配置下，200us 及其以上均可以确保 CMM 工作正常。

步骤 1、步骤 2、步骤 11 也可以略过，即可以动态更新配置，但是步骤 10 必须替换为，先配置 CMINTRAW[chk_end_raw]为 1（清除旧的检查结束原始中断），然后等待 CMINTRAW[chk_end_raw]再次为 1 时清除此原始中断，即动态更新配置须等待首次检测结束。否则动态更新配置完成后，会导致 CMM 第一次检查结束时保存的 CMCNTLOCK 不可预测。

6.4.3 检测到时钟失效的恢复流程

步骤 1 配置 SC_LOCKEN 为 0xEA510000，解除 CRG 相关寄存器的锁定。

步骤 2 配置 PERI_CRG4[pll_pd]为 1，下电 PLL。

步骤 3 配置 PERI_CRG4[pll_pd]为 0，上电 PLL。

步骤 4 检查 CRG.PERI_CRG8[pll_lock]是否等于 1，即 PLL 是否重新锁定。若检查到 1，表示 PLL 重新锁定，主时钟已恢复，则进行下一步。否则重复步骤 4。

步骤 5 配置 CMINTRAW[clk_fail_raw]为 1，清除时钟失效中断。此时，CMCNT 清零重新开始计数，CMM 重新开始工作，同时 clock_fail 硬件信号拉低，CRG 自动切换回主时钟。

----结束

6.4.4 典型示例和注意事项

一旦目标时钟源选择（CMTGTCTRL[tgtsel]），目标时钟分频系数（CMTGTCTRL[tgtscale]）确定后，（分频后的）目标时钟周期就已经可以计算得出。例如：目标时钟源为 200MHz，目标时钟分频系数为 1024 分频，则被测量的理论周期为 5ns×1024 = 5120ns = 5.12us。

一旦参考时钟源选择（CMREFCTRL[refsel]），参考时钟分频系数（CMTGTCTRL[refdiv]）确定后，测量结束时的理论计数值就确定了。例如：参考时钟源为 100MHz，参考时钟分频系数为不分频，且被测量的理论周期为 5.12us，则理想计数值为 5.12us × 100MHz = 512，那么检查窗口上下限可以根据该值和需要的精度来配置。

须知

在上述示例中，被测量的理论周期为 5.12us，也就是每 5.12us 测量一次。如果此时使能了检测结束中断（CMINTENA[chk_end_en]配置为 1），那么 CMM 的模块中断会在每次测量结束时上报，即每 5.12us 上报一次，CPU 将会接收到高频中断。

目标时钟频率的容许偏差：

以下计算的假设参考时钟频率为理想频率。

参考时钟（分频后）理想周期：$T_{ref_ideal}=\frac{\text{参考时钟分频值}}{F_{ref}}$；

目标时钟（分频后）理想周期：$T_{tgt_ideal}=\frac{\text{目标时钟分频值}}{F_{tgt}}$；

计数值：$CMCNTLOCK=\frac{T_{tgt_ideal}}{T_{ref_ideal}}=\frac{\text{目标时钟分频值}\times F_{ref}}{\text{参考时钟分频值}\times F_{tgt}}$；

允许的范围：$CMWDOL \le CMCNTLOCK \le CMWDOH$；

即：$F_{tgt} \ge \frac{\text{目标时钟分频值} \times F_{ref}}{\text{参考时钟分频值} \times CMWDOH}$ 且 $F_{tgt} \le \frac{\text{目标时钟分频值} \times F_{ref}}{\text{参考时钟分频值} \times CMWDOL}$。

6.5 CMM 寄存器概览

CMM 寄存器概览如表 6-2 所示。

表6-2 CMM 寄存器概览（基址是 0x1001_0000）

偏移地址	名称	描述	页码
0x0000	CMVER	CMM Version，CMM 版本信息寄存器	CMVER
0x0004	CMCTRL	CMM Control，CMM 控制寄存器	CMCTRL
0x0008	CMTGTCTRL	CMM Target Clock Control，CMM 目标时钟控制寄存器	CMTGTCTRL
0x000C	CMREFCTRL	CMM Reference Clock Control，CMM 参考时钟控制寄存器	CMREFCTRL
0x0010	CMWDOH	CMM Check Window High，CMM 检查窗口上限寄存器	CMWDOH
0x0014	CMWDOL	CMM Check Window Low，CMM 检查窗口下限寄存器	CMWDOL
0x0018	CMCNTLOCK	CMM Count Locked Value，CMM 计数锁存值寄存器	CMCNTLOCK
0x001C	CMINTENA	CMM Interrupt Enable，CMM 中断使能寄存器	CMINTENA
0x0020	CMINTSTS	CMM Interrupt Status，CMM 中断状态寄存器	CMINTSTS
0x0024	CMINTRAW	CMM Raw Interrupt，CMM 原始中断寄存器	CMINTRAW
0x0028	CMINTINJ	CMM Interrupt Injection，CMM 中断注入寄存器	CMINTINJ

偏移地址	名称	描述	页码
0x002C	CMSTS	CMM Status，CMM 状态寄存器	CMSTS

6.6 CMM 寄存器描述

CMVER

CMVER 为 CMM Version，CMM 版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0222_0507

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27:24]	RO	release_ver	版本信息。	0x2
[23:16]	RO	year	年。	0x22
[15:0]	RO	month_day	月日。	0x0507

CMCTRL

CMCTRL 为 CMM Control，CMM 控制寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RW	cfen	CFD 时钟失效检测使能。若使能该功能，当 CMCNT 大于 cfwdoh 时，clock_fail 信号有效，自动将系统主频切换到 CLK_LOSC，同时输出到 APT 模块。若不使能该功能，cfwdoh 无效，clock_fail 信号无效。该功能受 cmen 控制。 0：不使能； 1：使能。	0x0

Bits	Access	Name	Description	Reset
[0]	RW	cmen	CMM 模块使能。 0：不使能； 1：使能。	0x0

CMTGTCTRL

CMTGTCTRL 为 CMM Target Clock Control，CMM 目标时钟控制寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:7]	-	reserved	保留。	0x0000000
[6:4]	RW	tgtscale	CMM 目标时钟分频系数，偶数倍占空比 50%分频，CMM 使用分频后的时钟作为被监控对象。 000：不分频； 001：32 分频； 010：128 分频； 011：1024 分频； 100：8192 分频； 其他：8192 分频。	0x0
[3]	-	reserved	保留。	0x0
[2:0]	RW	tgtsel	CMM 工作目标时钟源选择。 000：目标时钟源 0，clk_losc； 001：目标时钟源 1，clk_hosc； 010：目标时钟源 2，clk_xtal； 011：目标时钟源 3，hclk； 其他：保留。	0x0

CMREFCTRL

CMREFCTRL 为 CMM Reference Clock Control，CMM 参考时钟控制寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5:4]	RW	refdiv	CMM 参考时钟分频系数，CMM 使用分频后的时钟作为监控参考。 00：不分频； 01：4 分频； 10：8 分频； 11：32 分频。	0x0
[3:2]	-	reserved	保留。	0x0
[1:0]	RW	refsel	CMM 工作参考时钟源选择。 00：参考时钟源 0，clk_losc； 01：参考时钟源 1，clk_hosc； 10：参考时钟源 2，clk_xtal； 11：参考时钟源 3，hclk。	0x0

CMWDOH

CMWDOH 为 CMM Check Window High，CMM 检查窗口上限寄存器。

Offset Address: 0x0010 Total Reset Value: 0xFFFF_FFFF

Bits	Access	Name	Description	Reset
[31:16]	RW	cfdwdoh	CMM 时钟失效检查窗口上界限，CMCNT 大于该值说明目标时钟失效，产生 clk_fail_raw 原始中断。	0xFFFF
[15:0]	RW	cmwdoh	CMM 时钟频率错误检查窗口上界限，CMCNT 大于该值说明频率错误，产生 freq_err_raw 原始中断。	0xFFFF

CMWDOL

CMWDOL 为 CMM Check Window Low，CMM 检查窗口下限寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	cmwdol	CMM 时钟频率错误检查窗口下界限，CMCNT 小于该值说明频率错误，产生 freq_err_raw 原始中断。	0x0000

CMCNTLOCK

CMCNTLOCK 为 CMM Count Locked Value，CMM 计数锁存值寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RO	cmcnt_lock	CMM 内部计数器计数锁存值。	0x0000

CMINTENA

CMINTENA 为 CMM Interrupt Enable，CMM 中断使能寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW	clk_fail_en	CMM 时钟失效中断使能。不使能时，即屏蔽相应的中断状态，不影响相应的原始中断。若全部不使能，CFD 不会上报中断给 CPU。 0：不使能； 1：使能。	0x0
[2]	RW	freq_err_en	CMM 频率错误中断使能。不使能时，即屏	0x0

Bits	Access	Name	Description	Reset
			蔽相应的中断状态，不影响相应的原始中断。若全部不使能，CMM 不会上报中断给 CPU。 0：不使能； 1：使能。	
[1]	RW	chk_end_en	CMM 检测结束中断使能。不使能时，即屏蔽相应的中断状态，不影响相应的原始中断。若全部不使能，CMM 不会上报中断给 CPU。 0：不使能； 1：使能。	0x0
[0]	RW	cnt_ovf_en	CMM 计数溢出中断使能。不使能时，即屏蔽相应的中断状态，不影响相应的原始中断。若全部不使能，CMM 不会上报中断给 CPU。 0：不使能； 1：使能。	0x0

CMINTSTS

CMINTSTS 为 CMM Interrupt Status，CMM 中断状态寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RO	clk_fail_int	CMM 时钟失效中断状态。 0：无中断； 1：有中断。	0x0
[2]	RO	freq_err_int	CMM 频率错误中断状态。 0：无中断； 1：有中断。	0x0

Bits	Access	Name	Description	Reset
[1]	RO	chk_end_int	CMM 检测结束中断状态。 0：无中断； 1：有中断。	0x0
[0]	RO	cnt_ovf_int	CMM 计数溢出中断状态。 0：无中断； 1：有中断。	0x0

CMINTRAW

CMINTRAW 为 CMM Raw Interrupt，CMM 原始中断寄存器。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW1C	clk_fail_raw	CMM 时钟失效原始中断。当 cmen 为 1 时，才可能产生此中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[2]	RW1C	freq_err_raw	CMM 频率错误原始中断。当 cmen 为 1 时，才可能产生此中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[1]	RW1C	chk_end_raw	CMM 检测结束中断原始中断。当 cmen 为 1 时，才可能产生此中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[0]	RW1C	cnt_ovf_raw	CMM 计数溢出中断原始中断。当 cmen 为 1 时，才可能产生此中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0

CMINTINJ

CMINTINJ 为 CMM Interrupt Injection，CMM 中断注入寄存器。

Offset Address: 0x0028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	W1_PULSE	clk_fail_inj	CMM 时钟失效中断注入。 0：不注入； 1：注入。写 1 后该位域自动清零。	0x0
[2]	W1_PULSE	freq_err_inj	CMM 频率错误中断注入。 0：不注入； 1：注入。写 1 后该位域自动清零。	0x0
[1]	W1_PULSE	chk_end_inj	CMM 检测结束中断注入。 0：不注入； 1：注入。写 1 后该位域自动清零。	0x0
[0]	W1_PULSE	cnt_ovf_inj	CMM 计数溢出中断注入。 0：不注入； 1：注入。写 1 后该位域自动清零。	0x0

CMSTS

CMSTS 为 CMM Status，CMM 状态寄存器。

Offset Address: 0x002C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RO	cnt_frozen	CMCNT 计数器是否被冻结。当 CMCNT 计数大于 cfdwdoh 时，CMCNT 被冻结。清除 clk_fail 中断后，CMCNT 被解冻。 0：CMCNT 未被冻结；	0x0

Bits	Access	Name	Description	Reset
			1：CMCNT 已被冻结。	

7 管脚复用控制(IOCMG)

7.1 概述

IOCMG（I/O Control & Multiplex Generater）实现对 MCU I/O 管脚的状态控制和功能复用管理。除特殊说明外，所有 I/O 管脚默认为 GPIO 输入功能。

7.2 特性

IOCMG 能够为 I/O 管脚提供以下可配置特性：

- 上/下拉
- 施密特输入开/关
- 输出驱动能力
- 输出信号边沿快/慢
- 数字/模拟模式切换
- 复用功能选择

7.3 I/O 模拟模式

MCU 部分 I/O 支持模拟模式。当 I/O 管脚复用功能编号大于等于 12 时，I/O 处于模拟模式。模拟模式下，应禁用 I/O 上拉、下拉配置，此时通过 GPIO 读取 I/O 状态值为低电平。I/O 复用功能选择为 ADC、ACMP、PGA 模拟输入或 DAC、PGA 模拟输出等模拟功能时，还需要在各模块配置中使能相应功能。

7.4 I/O 晶振模式

MCU 提供两个管脚（PA3 和 PA4）支持外部晶振模式，可为 MCU 提供高精度时钟。当两个管脚功能分别选择为 XTAL_OUT、XTAL_IN，且系统配置控制器（SYSCTRL）的 XTAL_CFG[osc_ie]和 XTAL_CFG[osc_e]均配置为高电平时，I/O 处于晶振模式。晶振模式下，应禁用 XTAL_OUT、XTAL_IN 功能管脚的上拉、下拉配置。

MCU 还支持外部时钟模式。当管脚功能选择为 XTAL_OUT，且系统配置控制器（SYSCTRL）的 XTAL_CFG[osc_ie]配置为高电平，XTAL_CFG[osc_e]配置为低电平时，可通过 XTAL_OUT 功能管脚输入外部时钟。外部时钟模式下，应禁用 XTAL_OUT 功能管脚的上拉、下拉配置。XTAL_IN 功能管脚不受影响，可以独立复用为其他功能，如 GPIO。

7.5 IOCMG 寄存器

7.5.1 IOCMG_AON 寄存器概览

IOCMG_AON 寄存器概览如表 7-1 所示。

表7-1 IOCMG_AON 寄存器概览（基址是 0x147F_0000）

偏移地址	名称	描述	页码
0x0000	IOCFG_PA2	PA2 管脚控制寄存器	IOCFG_PA2
0x0010	IOCFG_PF0	PF0 管脚控制寄存器	IOCFG_PF0
0x0018	IOCFG_PD3	PD3 管脚控制寄存器	IOCFG_PD3
0x001C	IOCFG_PC4	PC4 管脚控制寄存器	IOCFG_PC4

7.5.2 IOCMG_AON 寄存器描述

IOCFG_PA2

IOCFG_PA2 为 PA2 管脚控制寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0731

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x1
[9]	RW	pa2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x1
[7]	RW	pa2_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pa2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

IOCFG_PF0

IOCFG_PF0 为 PF0 管脚控制寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pf0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pf0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pf0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pf0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pf0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pf0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD3

IOCFG_PD3 为 PD3 管脚控制寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd3_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd3_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd3_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd3_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC4

IOCFG_PC4 为 PC4 管脚控制寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc4_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc4_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc4_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc4_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc4_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc4_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

7.5.3 IOCMG_CORE 寄存器概览

IOCMG_CORE 寄存器概览如表 7-2 所示。

表7-2 IOCMG_CORE 寄存器概览（基址是 0x147F_0000）

偏移地址	名称	描述	页码
0x0100	IOCFG_PA7	PA7 管脚控制寄存器	IOCFG_PA7
0x0104	IOCFG_PF1	PF1 管脚控制寄存器	IOCFG_PF1
0x0108	IOCFG_PC7	PC7 管脚控制寄存器	IOCFG_PC7
0x010C	IOCFG_PC6	PC6 管脚控制寄存器	IOCFG_PC6
0x0110	IOCFG_PC5	PC5 管脚控制寄存器	IOCFG_PC5
0x0114	IOCFG_PF2	PF2 管脚控制寄存器	IOCFG_PF2
0x0118	IOCFG_PD7	PD7 管脚控制寄存器	IOCFG_PD7
0x011C	IOCFG_PD6	PD6 管脚控制寄存器	IOCFG_PD6
0x0120	IOCFG_PD5	PD5 管脚控制寄存器	IOCFG_PD5
0x0124	IOCFG_PF3	PF3 管脚控制寄存器	IOCFG_PF3
0x0128	IOCFG_PB5	PB5 管脚控制寄存器	IOCFG_PB5
0x012C	IOCFG_PB6	PB6 管脚控制寄存器	IOCFG_PB6
0x0130	IOCFG_PB7	PB7 管脚控制寄存器	IOCFG_PB7
0x0134	IOCFG_PE7	PE7 管脚控制寄存器	IOCFG_PE7
0x0138	IOCFG_PE5	PE5 管脚控制寄存器	IOCFG_PE5
0x013C	IOCFG_PE6	PE6 管脚控制寄存器	IOCFG_PE6
0x0140	IOCFG_PB3	PB3 管脚控制寄存器	IOCFG_PB3
0x0144	IOCFG_PB4	PB4 管脚控制寄存器	IOCFG_PB4
0x0148	IOCFG_PD0	PD0 管脚控制寄存器	IOCFG

偏移地址	名称	描述	页码
			_PD0
0x014C	IOCFG_PD1	PD1 管脚控制寄存器	IOCFG_PD1
0x0150	IOCFG_PD2	PD2 管脚控制寄存器	IOCFG_PD2
0x0154	IOCFG_PE0	PE0 管脚控制寄存器	IOCFG_PE0
0x015C	IOCFG_PE1	PE1 管脚控制寄存器	IOCFG_PE1
0x0160	IOCFG_PE2	PE2 管脚控制寄存器	IOCFG_PE2
0x0164	IOCFG_PE3	PE3 管脚控制寄存器	IOCFG_PE3
0x0168	IOCFG_PB0	PB0 管脚控制寄存器	IOCFG_PB0
0x016C	IOCFG_PB1	PB1 管脚控制寄存器	IOCFG_PB1
0x0170	IOCFG_PD4	PD4 管脚控制寄存器	IOCFG_PD4
0x0174	IOCFG_PE4	PE4 管脚控制寄存器	IOCFG_PE4
0x0178	IOCFG_PC0	PC0 管脚控制寄存器	IOCFG_PC0
0x017C	IOCFG_PC1	PC1 管脚控制寄存器	IOCFG_PC1
0x0180	IOCFG_PC2	PC2 管脚控制寄存器	IOCFG_PC2
0x0184	IOCFG_PC3	PC3 管脚控制寄存器	IOCFG_PC3
0x0188	IOCFG_PA0	PA0 管脚控制寄存器	IOCFG_PA0
0x018C	IOCFG_PA1	PA1 管脚控制寄存器	IOCFG_PA1
0x0190	IOCFG_PA3	PA3 管脚控制寄存器	IOCFG_PA3
0x0194	IOCFG_PA4	PA4 管脚控制寄存器	IOCFG_PA4
0x0198	IOCFG_PB2	PB2 管脚控制寄存器	IOCFG_PB2

偏移地址	名称	描述	页码
0x01A0	IOCFG_PA5	PA5 管脚控制寄存器	IOCFG_PA5
0x01A4	IOCFG_PA6	PA6 管脚控制寄存器	IOCFG_PA6

7.5.4 IOCMG_CORE 寄存器描述

IOCFG_PA7

IOCFG_PA7 为 PA7 管脚控制寄存器。

Offset Address: 0x0100 Total Reset Value: 0x0000_02B1

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa7_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa7_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa7_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pa7_pd	管脚下拉控制。 0：关闭； 1：打开。	0x1
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa7_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4；	0x3

Bits	Access	Name	Description	Reset
			01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	
[3:0]	RW	pa7_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

IOCFG_PF1

IOCFG_PF1 为 PF1 管脚控制寄存器。

Offset Address: 0x0104 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pf1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pf1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pf1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pf1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pf1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。	0x3

Bits	Access	Name	Description	Reset
			00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	
[3:0]	RW	pf1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC7

IOCFG_PC7 为 PC7 管脚控制寄存器。

Offset Address: 0x0108 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc7_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc7_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc7_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc7_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc7_ds	管脚驱动能力选择（档位越高，驱动能力	0x3

Bits	Access	Name	Description	Reset
			越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	
[3:0]	RW	pc7_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC6

IOCFG_PC6 为 PC6 管脚控制寄存器。

Offset Address: 0x010C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc6_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc6_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc6_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc6_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[5:4]	RW	pc6_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc6_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC5

IOCFG_PC5 为 PC5 管脚控制寄存器。

Offset Address: 0x0110 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc5_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc5_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc5_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc5_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0

Bits	Access	Name	Description	Reset
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc5_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc5_func	管脚复用功能选择: 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PF2

IOCFG_PF2 为 PF2 管脚控制寄存器。

Offset Address: 0x0114 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pf2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pf2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pf2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pf2_pd	管脚下拉控制。 0：关闭；	0x0

Bits	Access	Name	Description	Reset
			1：打开。	
[6]	-	reserved	保留。	0x0
[5:4]	RW	pf2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pf2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD7

IOCFG_PD7 为 PD7 管脚控制寄存器。

Offset Address: 0x0118 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd7_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd7_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd7_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd7_pd	管脚下拉控制。	0x0

Bits	Access	Name	Description	Reset
			0：关闭； 1：打开。	
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd7_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd7_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD6

IOCFG_PD6 为 PD6 管脚控制寄存器。

Offset Address: 0x011C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd6_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd6_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd6_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0

Bits	Access	Name	Description	Reset
[7]	RW	pd6_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd6_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd6_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD5

IOCFG_PD5 为 PD5 管脚控制寄存器。

Offset Address: 0x0120 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd5_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd5_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd5_pu	管脚上拉控制。 0：关闭；	0x0

Bits	Access	Name	Description	Reset
			1：打开。	
[7]	RW	pd5_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd5_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd5_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PF3

IOCFG_PF3 为 PF3 管脚控制寄存器。

Offset Address: 0x0124 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pf3_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pf3_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pf3_pu	管脚上拉控制。	0x0

Bits	Access	Name	Description	Reset
			0：关闭； 1：打开。	
[7]	RW	pf3_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pf3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pf3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB5

IOCFG_PB5 为 PB5 管脚控制寄存器。

Offset Address: 0x0128 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb5_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb5_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1

Bits	Access	Name	Description	Reset
[8]	RW	pb5_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb5_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb5_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb5_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB6

IOCFG_PB6 为 PB6 管脚控制寄存器。

Offset Address: 0x012C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb6_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb6_sr	管脚电平转换速率控制。 0：快沿输出；	0x1

Bits	Access	Name	Description	Reset
			1：慢沿输出。	
[8]	RW	pb6_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb6_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb6_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb6_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB7

IOCFG_PB7 为 PB7 管脚控制寄存器。

Offset Address: 0x0130 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb7_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb7_sr	管脚电平转换速率控制。	0x1

Bits	Access	Name	Description	Reset
			0：快沿输出； 1：慢沿输出。	
[8]	RW	pb7_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb7_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb7_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb7_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE7

IOCFG_PE7 为 PE7 管脚控制寄存器。

Offset Address: 0x0134 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe7_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0

Bits	Access	Name	Description	Reset
[9]	RW	pe7_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe7_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe7_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe7_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe7_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE5

IOCFG_PE5 为 PE5 管脚控制寄存器。

Offset Address: 0x0138 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe5_se	管脚施密特输入控制。 0：关闭；	0x0

Bits	Access	Name	Description	Reset
			1：打开。	
[9]	RW	pe5_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe5_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe5_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe5_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe5_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE6

IOCFG_PE6 为 PE6 管脚控制寄存器。

Offset Address: 0x013C Total Reset Value: 0x0000_0220

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe6_se	管脚施密特输入控制。	0x0

Bits	Access	Name	Description	Reset
			0：关闭； 1：打开。	
[9]	RW	pe6_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe6_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe6_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe6_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x2
[3:0]	RW	pe6_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB3

IOCFG_PB3 为 PB3 管脚控制寄存器。

Offset Address: 0x0140 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000

Bits	Access	Name	Description	Reset
[10]	RW	pb3_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb3_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pb3_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb3_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB4

IOCFG_PB4 为 PB4 管脚控制寄存器。

Offset Address: 0x0144 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb4_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb4_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pb4_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb4_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb4_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb4_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD0

IOCFG_PD0 为 PD0 管脚控制寄存器。

Offset Address: 0x0148 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD1

IOCFG_PD1 为 PD1 管脚控制寄存器。

Offset Address: 0x014C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD2

IOCFG_PD2 为 PD2 管脚控制寄存器。

Offset Address: 0x0150 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd2_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE0

IOCFG_PE0 为 PE0 管脚控制寄存器。

Offset Address: 0x0154 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pe0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列 数据手册》。	0x0

IOCFG_PE1

IOCFG_PE1 为 PE1 管脚控制寄存器。

Offset Address: 0x015C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pe1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE2

IOCFG_PE2 为 PE2 管脚控制寄存器。

Offset Address: 0x0160 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pe2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe2_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE3

IOCFG_PE3 为 PE3 管脚控制寄存器。

Offset Address: 0x0164 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe3_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pe3_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe3_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe3_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB0

IOCFG_PB0 为 PB0 管脚控制寄存器。

Offset Address: 0x0168 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pb0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB1

IOCFG_PB1 为 PB1 管脚控制寄存器。

Offset Address: 0x016C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pb1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pb1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PD4

IOCFG_PD4 为 PD4 管脚控制寄存器。

Offset Address: 0x0170 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pd4_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pd4_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pd4_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pd4_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pd4_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pd4_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PE4

IOCFG_PE4 为 PE4 管脚控制寄存器。

Offset Address: 0x0174 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pe4_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pe4_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pe4_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pe4_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pe4_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pe4_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC0

IOCFG_PC0 为 PC0 管脚控制寄存器。

Offset Address: 0x0178 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC1

IOCFG_PC1 为 PC1 管脚控制寄存器。

Offset Address: 0x017C Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC2

IOCFG_PC2 为 PC2 管脚控制寄存器。

Offset Address: 0x0180 Total Reset Value: 0x0000_0210

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc2_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x1
[3:0]	RW	pc2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PC3

IOCFG_PC3 为 PC3 管脚控制寄存器。

Offset Address: 0x0184 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pc3_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pc3_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pc3_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pc3_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pc3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pc3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PA0

IOCFG_PA0 为 PA0 管脚控制寄存器。

Offset Address: 0x0188 Total Reset Value: 0x0000_02B1

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa0_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa0_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa0_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pa0_pd	管脚下拉控制。 0：关闭； 1：打开。	0x1
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa0_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pa0_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

IOCFG_PA1

IOCFG_PA1 为 PA1 管脚控制寄存器。

Offset Address: 0x018C Total Reset Value: 0x0000_0311

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa1_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa1_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa1_pu	管脚上拉控制。 0：关闭； 1：打开。	0x1
[7]	RW	pa1_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa1_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x1
[3:0]	RW	pa1_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

IOCFG_PA3

IOCFG_PA3 为 PA3 管脚控制寄存器。

Offset Address: 0x0190 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa3_se1	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa3_sr1	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa3_pu1	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pa3_pd1	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa3_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pa3_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PA4

IOCFG_PA4 为 PA4 管脚控制寄存器。

Offset Address: 0x0194 Total Reset Value: 0x0000_0230

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa4_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa4_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa4_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pa4_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa4_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pa4_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PB2

IOCFG_PB2 为 PB2 管脚控制寄存器。

Offset Address: 0x0198 Total Reset Value: 0x0000_06B0

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pb2_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x1
[9]	RW	pb2_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pb2_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pb2_pd	管脚下拉控制。 0：关闭； 1：打开。	0x1
[6]	-	reserved	保留。	0x0
[5:4]	RW	pb2_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pb2_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x0

IOCFG_PA5

IOCFG_PA5 为 PA5 管脚控制寄存器。

Offset Address: 0x01A0 Total Reset Value: 0x0000_0221

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa5_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa5_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa5_pu	管脚上拉控制。 0：关闭； 1：打开。	0x0
[7]	RW	pa5_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa5_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x2
[3:0]	RW	pa5_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

IOCFG_PA6

IOCFG_PA6 为 PA6 管脚控制寄存器。

Offset Address: 0x01A4 Total Reset Value: 0x0000_0331

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	pa6_se	管脚施密特输入控制。 0：关闭； 1：打开。	0x0
[9]	RW	pa6_sr	管脚电平转换速率控制。 0：快沿输出； 1：慢沿输出。	0x1
[8]	RW	pa6_pu	管脚上拉控制。 0：关闭； 1：打开。	0x1
[7]	RW	pa6_pd	管脚下拉控制。 0：关闭； 1：打开。	0x0
[6]	-	reserved	保留。	0x0
[5:4]	RW	pa6_ds	管脚驱动能力选择（档位越高，驱动能力越大）。 00：IO2 档位 4； 01：IO2 档位 3； 10：IO2 档位 2； 11：IO2 档位 1。	0x3
[3:0]	RW	pa6_func	管脚复用功能选择。 具体复用功能请参考《3061M 系列数据手册》。	0x1

8 通用 I/O(GPIO)

8.1 概述

MCU 支持 6 组 GPIO（General Purpose Input/Output）控制器，即 GPIO0～GPIO5。每组 GPIO 控制器最多提供 8 个可编程的 GPIO。

GPIO 控制器可以根据电平或跳变值产生可屏蔽的中断。gpio_int（General Purpose Input Output Interrupt）信号给中断控制器一个指示，表示有中断发生。

须知

- MCU 相关管脚上下拉、施密特输入、模拟\数字、锁定、复用等功能请参见 7 管脚复用控制(IOCMG)章节。
- 对于默认是输出信号的管脚，如果复用 GPIO 控制器功能，请注意对接 MCU 和器件的管脚必须是输入。否则若对接管脚之间的电平不一致，可能会损坏 MCU。

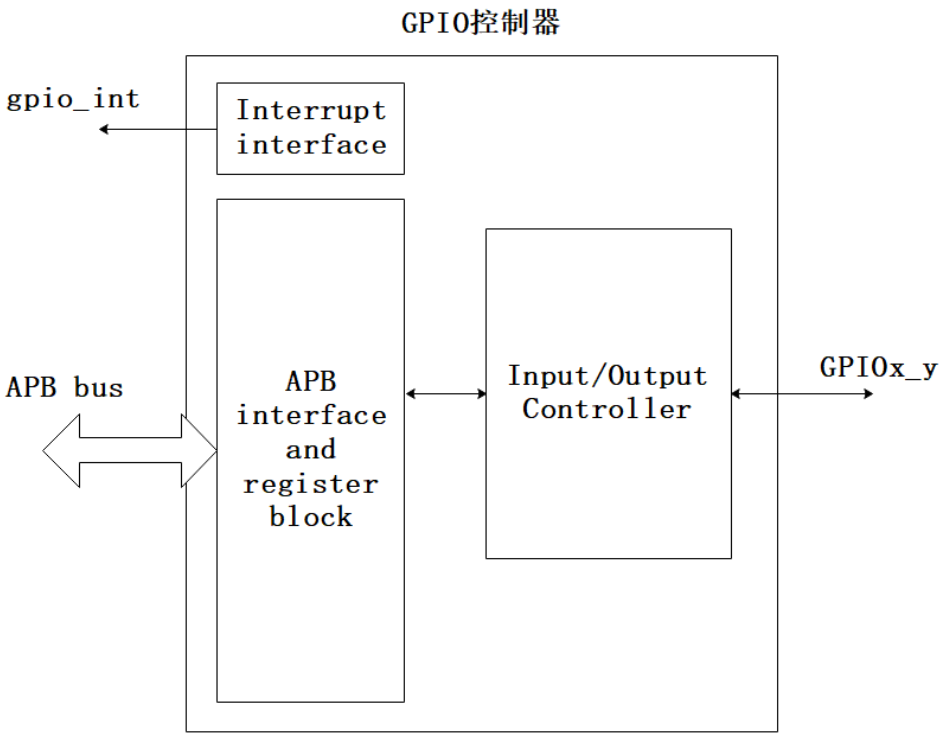
8.2 特性

每个 GPIO 均可独立配置为输入、输出。

- 作为输入 GPIO 时，可作为中断源。中断触发类型可以灵活选择。
- 作为输出 GPIO 时，每个 GPIO 都可以独立清 0 或置 1。
- 支持中断屏蔽。

8.3 功能描述

图8-1 模块框图



GPIO 控制器通过总线配置寄存器的值，以控制 GPIO 为输入/输出，当作为输入时可以产生输入中断信号。

GPIOx_y 是 GPIO 控制器的输入/输出信号，x 取 0~5，对应 MCU 支持的 6 组 GPIO 控制器。y 取 0~7，代表每组 GPIO 控制器提供的 8 个可编程 GPIO。

可以通过配置 GPIO_DIR 对每个 GPIO 单独配置为输入/输出：

- 当 GPIO 作为输出时，可以通过 GPIO_DATA 对每个 GPIO 单独清 0 或置 1。
- 当 GPIO 作为输入时，可以通过 GPIO_DATA 读取相应 GPIO 的输入值。

中断功能

GPIO 控制器的中断由 GPIO_IS、GPIO_IBE、GPIO_IEV、GPIO_IE、GPIO_RIS、GPIO_MIS 和 GPIO_IC 寄存器控制，可以配置 GPIO 控制器的中断触发类型以及清除中断等。

GPIO_IS、GPIO_IBE 和 GPIO_IEV 三个寄存器共同决定了中断源的特性和中断触发类别。GPIO_IS 决定中断是边沿触发还是电平触发。当配置边沿触发时，GPIO_IBE 决定中断是单边沿还是双边沿触发。GPIO_IEV 决定中断是下降沿/低电平触发还是上升沿/高电平触发。

可以配置 GPIO_IE 以屏蔽相应中断。可以通过读 GPIO_RIS 和 GPIO_MIS 来查询中断的原始状态和屏蔽后状态。中断状态还支持通过 GPIO_IC 进行清除。

GPIO 控制器触发后产生可屏蔽的中断 gpio_int，中断信号送至中断控制器，表示 GPIO 控制器有中断产生，系统可以进入中断函数执行相应操作。

8.4 典型配置

接口复位

在 MCU 上电复位或系统复位时，GPIO 控制器模块同时被复位，GPIO 在复位之后默认处于输入状态。

通用输入输出

每个 GPIO 可以单独配置为输入或者输出，具体步骤如下：

步骤 1 配置寄存器 GPIO_DIR，选择 GPIO 是作为输入还是输出。

步骤 2 当 GPIO 配置成输入时，读取 GPIO_DATA 寄存器可查看输入信号值；当 GPIO 配置成输出时，通过向 GPIO_DATA 寄存器写入输出值可以控制 GPIO 的输出。

----结束

须知

当 GPIO 用作输出时，建议禁止 GPIO 控制器的中断功能，否则当输出信号符合中断产生条件时，会产生 GPIO 中断。

中断操作

GPIO 控制器的中断通过 GPIO_IS 等 7 个寄存器进行控制。通过这些寄存器可以选择中断源、极性以及边沿特性。GPIO 控制器对应的中断号及中断映射关系请参见 1.5 中断映射。

GPIO_IS、GPIO_IBE 和 GPIO_IEV 三个寄存器共同决定了中断源的特性和中断触发类别。

通过 GPIO_RIS 和 GPIO_MIS 分别读取中断的原始状态和屏蔽后的状态。通过 GPIO_IE 可以屏蔽中断，控制中断的最终上报情况。当多个 GPIO 中断同时产生时，会统一汇集成一个中断信号 gpio_int 进行上报。此外还提供了单独的 GPIO_IC 用于对中断状态进行清除控制。

每个 GPIO 都可以配置成中断模式，配置步骤如下：

步骤 1 配置 GPIO_IS，选择边沿触发或电平触发。

步骤 2 配置 GPIO_IEV，选择下降沿/上升沿触发和高电平/低电平触发。

步骤 3 如果选择边沿触发，需配置 GPIO_IBE，选择单沿或双沿触发方式。

步骤 4 向寄存器 GPIO_IC 写 0xFF，清除中断。

步骤 5 配置 GPIO_IE 相应 bit 为 1，使能中断。

----结束

8.5 GPIO 控制器寄存器概览

MCU 的 GPIO 控制器基地址如表 8-1 所示。

表8-1 MCU GPIO 控制器寄存器对应的基地址

GPIO 控制器	基地址
GPIO0	0x1450_0000
GPIO1	0x1450_1000
GPIO2	0x1450_2000
GPIO3	0x1450_3000
GPIO4	0x1450_4000
GPIO5	0x1450_5000

说明

GPIOn 对应的寄存器地址为：GPIOn 基地址+该寄存器偏移地址。

表 8-2 是单组 GPIO 控制器内部寄存器的偏移地址以及定义，GPIO0 ~ GPIOn 具有相同的寄存器组。

表8-2 GPIO 控制器寄存器概览

偏移地址	名称	描述	页码
0x000 ~ 0x3FC	GPIO_DATA	GPIO 数据寄存器	GPIO_DATA
0x400	GPIO_DIR	GPIO 方向控制寄存器	GPIO_DIR
0x404	GPIO_IS	GPIO 中断触发寄存器	GPIO_IS
0x408	GPIO_IBE	GPIO 双沿触发中断寄存器	GPIO_IBE
0x40C	GPIO_IEV	GPIO 触发中断条件寄存器	GPIO_IEV
0x410	GPIO_IE	GPIO 中断屏蔽寄存器	GPIO_IE
0x414	GPIO_RIS	GPIO 原始中断状态寄存器	GPIO_RIS
0x418	GPIO_MIS	GPIO 屏蔽中断状态寄存器	GPIO_MIS
0x41C	GPIO_IC	GPIO 中断清除寄存器	GPIO_IC

8.6 GPIO 控制器寄存器描述

GPIO_DATA

GPIO_DATA 为 GPIO 数据寄存器，用来对输入或输出数据进行缓存。

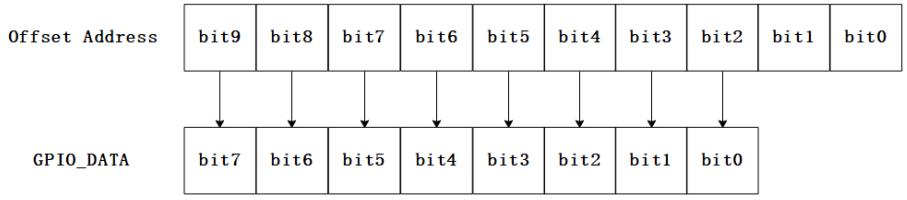
当配置 GPIO_DIR 中对应位为输出时，写入 GPIO_DATA 寄存器的值将会输出到相应的 GPIO；如果配置为输入时，将会读取相应输入 GPIO 的值。

须知

当 GPIO_DIR 相应的比特配置为输入时，有效读取的结果将返回 GPIO 的值；当配置为输出的时候，有效读取的结果将返回写入的值。

GPIO_DATA 寄存器利用偏移地址 Offset Address 实现了读写寄存器比特的屏蔽操作。GPIO_DATA 是一组寄存器的统称，对应 256 个地址空间，0x000 ~ 0x3FC，每个寄存器都有 8bit。Offset Address bit9 ~ bit2 分别对应 GPIO_DATA bit[7:0]，当相应的 bit 为高时，则可以对相应的位进行读写操作；反之，若对应 bit 为低则不能进行读写操作。如图 8-2 所示。

图8-2 比特屏蔽对应关系



例如：

- 若地址为 0x3FC（0b11_1111_1100），则对 GPIO_DATA 这 8bit 读写操作全部有效。
- 若地址为 0x328（0b11_0010_1000），则对 GPIO_DATA bit[7]、GPIO_DATA bit[6]、GPIO_DATA bit[3]、GPIO_DATA bit[1]的读写操作有效。
- 若地址为 0x000（0b00_0000_0000），则任意读写操作均无效。
- 若地址为 0x200（0b10_0000_0000），则仅对 GPIO_DATA bit[7]的读写操作有效。

Offset Address: 0x000 ~ 0x3FC Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_data	当 GPIO 配置为输入模式时，为 GPIO 输入数据；当 GPIO 配置为输出模式时，为输出数据。各比特均可独立控制。与 GPIO_DIR 配合使用。	0x00

GPIO_DIR

GPIO_DIR 为 GPIO 方向控制寄存器。用来配置 GPIO 方向。

Offset Address: 0x400 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_dir	GPIO 方向控制寄存器。bit[7:0]分别对应 GPIO_DATA bit[7:0]，各比特可独立控制。 0：输入； 1：输出。	0x00

GPIO_IS

GPIO_IS 为 GPIO 中断触发寄存器。用来配置 GPIO 触发电平方式。

Offset Address: 0x404 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_is	GPIO 中断触发控制寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，各比特独立控制。 0：边沿触发中断； 1：电平触发中断。	0x00

GPIO_IBE

GPIO_IBE 为 GPIO 双沿触发中断寄存器。用来配置 GPIO 沿触发方式。

Offset Address: 0x408 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_ibe	GPIO 中断沿触发控制寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，各比特独立控制。 0：单边沿触发中断，具体是上升沿还是下降沿触发由 GPIO_IEV 控制； 1：双边触发中断。	0x00

GPIO_IEV

GPIO_IEV 为 GPIO 触发中断条件寄存器。用来配置 GPIO 触发中断条件。

Offset Address: 0x40C Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_iev	GPIO 触发中断条件寄存器，bit[7:0]分别对应 GPIO_DATA[7:0]，各比特独立控制。 0：下降沿或低电平触发中断，具体由 GPIO_IS 控制； 1：上升沿或高电平触发中断。	0x00

GPIO_IE

GPIO_IE 为 GPIO 中断屏蔽寄存器。用来屏蔽 GPIO 中断。

Offset Address: 0x410 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW	gpio_ie	GPIO 中断屏蔽寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，各比特独立控制。 0：屏蔽中断； 1：不屏蔽中断。	0x00

GPIO_RIS

GPIO_RIS 为 GPIO 原始中断状态寄存器。用来查询 GPIO 原始中断状态。

Offset Address: 0x414 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RO	gpio_ris	GPIO 原始中断寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，指示未屏蔽的中断状态。该状态不受 GPIO_IE 寄存器屏蔽控制。 0：未发生中断； 1：已发生中断。	0x00

GPIO_MIS

GPIO_MIS 为 GPIO 屏蔽状态中断寄存器。用来查询 GPIO 屏蔽后的中断状态。对所有位进行或操作得到中断输出 gpio_int。

Offset Address: 0x418 Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RO	gpio_mis	GPIO 屏蔽后中断寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，指示经屏蔽后的中断状态。该状态受 GPIO_IE 寄存器屏蔽控制。 0：中断无效； 1：中断有效。	0x00

GPIO_IC

GPIO_IC 为 GPIO 中断清除寄存器。用来清除 GPIO 产生的中断，同时清除 GPIO_RIS 寄存器和 GPIO_MIS 寄存器。

Offset Address: 0x41C Total Reset Value: 0x00

Bits	Access	Name	Description	Reset
[7:0]	RW1C	gpio_ic	GPIO 中断清除寄存器，bit[7:0]分别对应 GPIO_DATA bit[7:0]，各比特可独立控制。 0：无影响； 1：清除中断。	0x00

9 系统配置控制器(SYSCTRL)

9.1 概述

系统控制器 SYSCTRL（System Controller）包含一组配置寄存器，其主要功能如下：

- 为系统关键寄存器提供写保护控制。
- 系统复位事件次数统计。
- 触发系统软件复位。
- 触发软件中断。
- 系统状态查询。
- 提供可读可写的通用寄存器。
- APT 同步启动配置。
- APT 管脚事件数字滤波配置。
- PVD 阈值和使能控制。
- XTAL 管脚时钟模式控制。
- SYSRAM 分区配置。

9.2 功能描述

9.2.1 系统关键寄存器的写保护控制

MCU 内部部分关键寄存器具有写保护功能，仅在写使能的情况下才能进行写操作。此类寄存器有 SYSCTRL 和 CRG 的部分寄存器，其写使能分别由 SC_LOCKEN[sc_cfg_lock]和 SC_LOCKEN[crg_cfg_lock]控制。

SC_LOCKEN 寄存器本身也具有防误写功能，此寄存器仅支持 word 写。当且仅当写操作数的高 16bit（SC_LOCKEN[31:16]）为 0xEA51 时，对此寄存器的低 16bit（SC_LOCKEN[15:0]）写操作才有效。例如，要打开 CRG 关键寄存器的写使能，需要向 SC_LOCKEN 写入 0xEA51_0002。

9.2.2 系统复位事件次数统计

MCU 支持对系统复位事件次数统计，支持的复位事件有软件复位、管脚复位、wwdg 复位、iwdg 复位。可以通过 SC_RST_CNT0 和 SC_RST_CNT1 进行查询复位次数。MCU 上电复位或进入 deepsleep 低功耗模式会清零次数统计。

9.2.3 系统软件复位

对寄存器 SC_SYS_RES[softresreq]执行写操作将触发系统软件复位。此寄存器具有写保护功能，具体请参见 9.2.1 系统关键寄存器的写保护控制章节。

9.2.4 系统软件中断

将寄存器 SC_SOFT_INT[software_int]置位，可触发系统软件中断；将寄存器 SC_SOFT_INT[software_int]清零，可清除系统软件中断。可以通过向寄存器 SC_SOFT_EVT_ID 写入不同值，用以区分软件中断事件。

9.2.5 系统状态查询

可通过寄存器 SC_SYS_STAT[update_mode]查询系统是否处于升级模式，此标识由 MCU 复位撤离时自动锁存管脚 UPDATE_MODE 状态获取。向 SC_SYS_STAT[update_mode_clear]写 1，可清除升级模式。当 MCU 处于 MCU 升级模式时，JTAG 相关的管脚强制为 JTAG 功能，软件无法更改复用关系。

可通过寄存器 CPU_STATUS 查询 CPU 状态。

可通过寄存器 SYSRAM_ERR 查询 SYSRAM 是否存在奇偶校验错误，也可对 SYSRAM_ERR[sysram_parity_err_clr]执行写操作清除校验错误。

9.2.6 可读可写的通用寄存器

系统控制器提供 6 个 32bit 通用寄存器，分别为 USER_POR_REG0/1、USER_HRST_REG0/1、USER_REG0/1。它们有不同的复位清零控制，其中：

- USER_POR_REG0/1：仅被 POR 复位或从低功耗模式唤醒时复位。
- USER_HRST_REG0/1：不会被系统软件复位所复位。

- USER_REG0/1：所有复位发生时都可复位此寄存器。

9.2.7 APT 同步启动

可通过寄存器 APT_RUN 控制 APT 模块是否使能。可以同时使能多个 APT 模块，以实现 APT 模块同步启动。

9.2.8 APT 管脚事件数字滤波

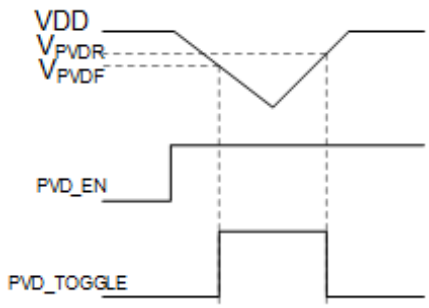
可通过寄存器 APT_POE_FILTER、APT_EVTIO_FILTER、APT_EVTMP_FILTER 对 APT 的管脚事件进行数字滤波，用以滤除噪声。滤波最大时长为 255 个系统总线时钟周期。

9.2.9 PVD 阈值和使能控制

支持使用 PVD 模块监视 V_{DD} 电源电压，方法是将 V_{DD} 与预设阈值电压比较，判断电压是否超出预期。可通过寄存器 PVD_CFG 配置 PVD 的上升沿阈值、下降沿阈值和使能控制。为保证 PVD 正常工作，上升沿阈值应高于下降沿阈值。

PVD 模块可以产生 pvd_toggle 标志，用于指示 MCU 供电电压 VDD 是高于还是低于 PVD 阈值。当 VDD 降至 PVD 下降沿阈值以下时，pvd_toggle 为高电平；当 VDD 上升至 PVD 上升沿阈值以上时，pvd_toggle 为低电平。pvd_toggle 标志为高电平时，能够触发中断事件，也可以触发 MCU 系统复位（请参见 PERI_CRG65 寄存器）。该功能的用处之一是可以在中断服务程序中执行紧急关闭系统的任务。

图9-1 PVD toggle 触发示意图



9.2.10 XTAL 管脚时钟模式控制

可通过寄存器 XTAL_CFG 对晶体管脚的工作模式进行控制。晶体管脚默认为 GPIO 模式，可通过 XTAL_CFG[osc_e]和 XTAL_CFG[osc_ie]置位使晶体管脚处于晶振模式。

晶体管脚处于晶振模式时，可通过 XTAL_CFG[osc_ds]调节晶体管脚对外部晶体的驱动能力。

9.2.11 SYSRAM 分区配置

可通过寄存器 SYSRAM_MAP[sysram_split]对整个 SYSRAM 空间进行分区配置。最小分区粒度为 4KB。

9.3 SYSCTRL0 寄存器概览

SYSCTRL0 寄存器概览如表 9-1 所示。

表9-1 SYSCTRL0 寄存器概览（基址是 0x1010_0000）

偏移地址	名称	描述	页码
0x0004	SC_SYS_RES	系统软件复位寄存器	SC_SYS_RES
0x0008	SC_RST_CNT0	复位次数统计寄存器 0	SC_RST_CNT0
0x000C	SC_RST_CNT1	复位次数统计寄存器 1	SC_RST_CNT1
0x0018	SC_SYS_STAT	系统启动模式寄存器	SC_SYS_STAT
0x0020	SC_SOFT_INT	软件中断寄存器	SC_SOFT_INT
0x0024	SC_SOFT_EVT_ID	软中断事件 ID 寄存器	SC_SOFT_EVT_ID
0x0044	SC_LOCKEN	关键寄存器的写保护控制寄存器	SC_LOCKEN
0x0200	SC_HRST_REG0	SC 专用硬复位寄存器 0	SC_HRST_REG0
0x0E00	USER_POR_REG0	USER 专用 POR 复位寄存器 0	USER_POR_REG0
0x0E04	USER_POR_REG1	USER 专用 POR 复位寄存器 1	USER_POR_REG1
0x0E40	USER_HRST_REG0	USER 专用硬复位寄存器 0	USER_HRST_RE

偏移地址	名称	描述	页码
			G0
0x0E44	USER_HRST_REG1	USER 专用硬复位寄存器 1	USER_HRST_REG1
0x0E80	USER_REG0	USER 专用寄存器 0	USER_REG0
0x0E84	USER_REG1	USER 专用寄存器 1	USER_REG1

9.4 SYSCTRL0 寄存器描述

SC_SYS_RES

SC_SYS_RES 为系统软件复位寄存器。

须知

该寄存器受写保护控制，由寄存器 SC_LOCKEN[sc_cfg_lock]决定。只有写使能时，对此寄存器的写操作才有效。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	softresreq	对此寄存器位域的任意写操作都会导致系统软复位。	0x0

SC_RST_CNT0

SC_RST_CNT0 为复位次数统计寄存器 0。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	ext_rst_cnt	RESETN 管脚复位次数统计。	0x0000

Bits	Access	Name	Description	Reset
[15:0]	RO	soft_rst_cnt	软复位次数统计。	0x0000

SC_RST_CNT1

SC_RST_CNT1 为复位次数统计寄存器 1。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	iwdg_rst_cnt	iwdg 复位次数统计。	0x0000
[15:0]	RO	wdg_rst_cnt	wwdg 复位次数统计。	0x0000

SC_SYS_STAT

SC_SYS_STAT 为系统启动模式寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RO	update_mode	系统升级标识，MCU 上电时锁存。 0：不升级，直接启动模式； 1：升级模式。 **注：当处于升级模式时，MCU JTAG 相关管脚强制为 JTAG 功能，无法软件切换为其他复用功能。**	0x0
[3:1]	-	reserved	保留。	0x0
[0]	RW	update_mode_clear	系统升级标识清除寄存器，MCU 上电完成后生效。 0：不清除； 1：清除。	0x0

SC_SOFT_INT

SC_SOFT_INT 为软件中断寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	software_int	软件中断寄存器。 0：清除中断； 1：产生中断。	0x0

SC_SOFT_EVT_ID

SC_SOFT_EVT_ID 为软中断事件 ID 寄存器。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	swint_evt_id	软中断事件 ID，可用于软中断事件类型定义。	0x00000000

SC_LOCKEN

SC_LOCKEN 为关键寄存器的写保护控制寄存器。

须知

仅当 word 写操作数的[31:16]位域为 0xEA51 时，才能对此寄存器的[15:0]位域进行写操作。

Offset Address: 0x0044 Total Reset Value: 0x0000_0003

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RW	sc_cfg_lock	SYSCTRL 相关寄存器写保护。 0：写使能；	0x1

Bits	Access	Name	Description	Reset
			1：写禁止。	
[0]	RW	crg_cfg_lock	CRG 相关寄存器写保护。 0：写使能； 1：写禁止。	0x1

SC_HRST_REG0

SC_HRST_REG0 为 SC 专用硬复位寄存器 0。

须知

该寄存器不会被系统软件复位所复位。

Offset Address: 0x0200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	sc_hrst_reg0	系统正常启动流程中，如果检测到此寄存器值为 0xA5A5A5A5，则 CPU 将停止启动系统，此时可以通过 JTAG/SWD 接口调试（或烧写）。	0x00000000

USER_POR_REG0

USER_POR_REG0 为 USER 专用 POR 复位寄存器 0。

须知

该寄存器仅被 POR 复位或从低功耗模式唤醒时复位。

Offset Address: 0x0E00 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_por_reg0	USER 专用 REG0。	0x00000000

USER_POR_REG1

USER_POR_REG1 为 USER 专用 POR 复位寄存器 1。

须知

该寄存器仅被 POR 复位或从低功耗模式唤醒时复位。

Offset Address: 0x0E04 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_por_reg1	USER 专用 REG1。	0x00000000

USER_HRST_REG0

USER_HRST_REG0 为 USER 专用硬复位寄存器 0。

须知

该寄存器不会被系统软复位所复位。

Offset Address: 0x0E40 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_hrst_reg0	USER 专用 REG2。	0x00000000

USER_HRST_REG1

USER_HRST_REG1 为 USER 专用硬复位寄存器 1。

须知

该寄存器不会被系统软复位所复位。

Offset Address: 0x0E44 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_hrst_reg1	USER 专用 REG3。	0x00000000

USER_REG0

USER_REG0 为 USER 专用寄存器 0。

Offset Address: 0x0E80 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_reg0	USER 专用 REG4。	0x00000000

USER_REG1

USER_REG1 为 USER 专用寄存器 1。

Offset Address: 0x0E84 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	user_reg1	USER 专用 REG5。	0x00000000

9.5 SYSCTRL1 寄存器概览

SYSCTRL1 寄存器概览如表 9-2 所示。

表9-2 SYSCTRL1 寄存器概览（基址是 0x1010_0000）

偏移地址	名称	描述	页码
0x8000	APT_RUN	APT 使能控制寄存器	APT_RUN
0x8010	APT_POE_FILTER	APT POE 滤波控制寄存器	APT_POE_FILTER
0x8014	APT_EVTIO_FILTER	APT EVTIO 滤波控制寄存器	APT_EVTIO_FILTER
0x8018	APT_EVTMP_FILTER	APT EVTMP 滤波控制寄存器	APT_EVTMP_FILTER
0x8100	XTAL_CFG	晶体 IO 控制寄存器	XTAL_CFG

偏移地址	名称	描述	页码
0x8300	SYSRAM_ERR	SYSRAM 奇偶校验状态寄存器	SYSRAM_ERR
0x8304	SYSRAM_MAP	SYSRAM 空间分配寄存器	SYSRAM_MAP
0x8400	PVD_CFG	PVD 配置寄存器	PVD_CFG
0x8404	PVD_STATUS	PVD 状态寄存器	PVD_STATUS
0x9000	CPU_STATUS	CPU 状态寄存器	CPU_STATUS

9.6 SYSCTRL1 寄存器描述

APT_RUN

APT_RUN 为 APT 使能控制寄存器。

Offset Address: 0x8000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW	apt3_run	APT3 使能。 0：不使能; 1：使能。	0x0
[2]	RW	apt2_run	APT2 使能。 0：不使能; 1：使能。	0x0
[1]	RW	apt1_run	APT1 使能。 0：不使能; 1：使能。	0x0
[0]	RW	apt0_run	APT0 使能。 0：不使能;	0x0

Bits	Access	Name	Description	Reset
			1：使能。	

APT_POE_FILTER

APT_POE_FILTER 为 APT POE 滤波控制寄存器。

Offset Address: 0x8010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:27]	-	reserved	保留。	0x00
[26]	RW	poe2_filter_en	POE2 FILTER 使能。 0：不使能; 1：使能。	0x0
[25]	RW	poe1_filter_en	POE1 FILTER 使能。 0：不使能; 1：使能。	0x0
[24]	RW	poe0_filter_en	POE0 FILTER 使能。 0：不使能; 1：使能。	0x0
[23:16]	RW	poe2_filter_level	POE2 FILTER 周期数（系统总线时钟）。	0x00
[15:8]	RW	poe1_filter_level	POE1 FILTER 周期数（系统总线时钟）。	0x00
[7:0]	RW	poe0_filter_level	POE0 FILTER 周期数（系统总线时钟）。	0x00

APT_EVTIO_FILTER

APT_EVTIO_FILTER 为 APT EVTIO 滤波控制寄存器。

Offset Address: 0x8014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	-	reserved	保留。	0x00
[25]	RW	apt_evtio5_filter_e	APT EVTIO5 FILTER 使能。	0x0

Bits	Access	Name	Description	Reset
		n	0：不使能; 1：使能。	
[24]	RW	apt_evtio4_filter_en	APT EVTIO4 FILTER 使能。 0：不使能; 1：使能。	0x0
[23:16]	-	reserved	保留。	0x00
[15:8]	RW	apt_evtio5_filter_level	APT EVTIO5 FILTER 周期数（系统总线时钟）。	0x00
[7:0]	RW	apt_evtio4_filter_level	APT EVTIO4 FILTER 周期数（系统总线时钟）。	0x00

APT_EVTMP_FILTER

APT_EVTMP_FILTER 为 APT EVTMP 滤波控制寄存器。

Offset Address: 0x8018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:27]	-	reserved	保留。	0x00
[26]	RW	apt_evtmp6_filter_en	APT EVTMP6 FILTER 使能。 0：不使能; 1：使能。	0x0
[25]	RW	apt_evtmp5_filter_en	APT EVTMP5 FILTER 使能。 0：不使能; 1：使能。	0x0
[24]	RW	apt_evtmp4_filter_en	APT EVTMP4 FILTER 使能。 0：不使能; 1：使能。	0x0
[23:16]	RW	apt_evtmp6_filter_level	APT EVTMP6 FILTER 周期数（系统总线时钟）。	0x00

Bits	Access	Name	Description	Reset
[15:8]	RW	apt_evtmp5_filter_level	APT EVTMP5 FILTER 周期数（系统总线时钟）。	0x00
[7:0]	RW	apt_evtmp4_filter_level	APT EVTMP4 FILTER 周期数（系统总线时钟）。	0x00

XTAL_CFG

XTAL_CFG 为晶体 IO 控制寄存器。

Offset Address: 0x8100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5]	RW	osc_ie	晶体 IO 时钟输入使能。 0：不使能; 1：使能。	0x0
[4]	RW	osc_e	晶体 IO 谐振 buf 使能。 0：不使能; 1：使能。	0x0
[3:0]	RW	osc_ds	晶体 IO 驱动能力配置。 配置值越大，驱动能力越大。	0x0

SYSRAM_ERR

SYSRAM_ERR 为 SYSRAM 奇偶校验状态寄存器。

Offset Address: 0x8300 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2]	RO	sysram1_parity_err	SYSRAM 奇偶校验错误状态。 0：无错误;	0x0

Bits	Access	Name	Description	Reset
			1：有错误。	
[1]	RO	sysram0_parity_err	SYSRAM 奇偶校验错误状态。 0：无错误； 1：有错误。	0x0
[0]	WO	sysram_parity_err_clr	SYSRAM 奇偶校验错误状态清除。 向该位域执行写动作，会清除 SYSRAM 奇偶校验错误状态。	0x0

SYSRAM_MAP

SYSRAM_MAP 为 SYSRAM 空间分配寄存器。

须知

该寄存器受写保护控制，由寄存器 SC_LOCKEN[sc_cfg_lock]决定。只有写使能时，对此寄存器的写操作才有效。

Offset Address: 0x8304 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2:0]	RW	sysram_split	SYSRAM 空间分配。 000：SYSRAM0 16KB，SYSRAM1 16KB； 001：SYSRAM0 4KB，SYSRAM1 28KB； 010：SYSRAM0 8KB，SYSRAM1 24KB； 011：SYSRAM0 12KB，SYSRAM1 20KB； 100：SYSRAM0 16KB，SYSRAM1 16KB；	0x0

Bits	Access	Name	Description	Reset
			101：SYSRAM0 20KB，SYSRAM1 12KB； 110：SYSRAM0 24KB，SYSRAM1 8KB； 111：SYSRAM0 28KB，SYSRAM1 4KB。	

PVD_CFG

PVD_CFG 为 PVD 配置寄存器。

Offset Address: 0x8400 Total Reset Value: 0x0000_0048

Bits	Access	Name	Description	Reset
[31:7]	-	reserved	保留。	0x0000000
[6:4]	RW	pvd_fall_thd	PVD 下降沿阈值。 000：2.08V； 001：2.18V； 010：2.28V； 011：2.38V； 100：2.48V； 101：2.58V； 110：2.68V； 111：2.78V。	0x4
[3:1]	RW	pvd_rise_thd	PVD 上升沿阈值。 000：2.18V； 001：2.28V； 010：2.38V； 011：2.48V； 100：2.58V； 101：2.68V；	0x4

Bits	Access	Name	Description	Reset
			110：2.78V； 111：2.88V。	
[0]	RW	pvd_en	PVD 使能标识。 0：PVD 关闭； 1：PVD 使能。	0x0

PVD_STATUS

PVD_STATUS 为 PVD 状态寄存器。

Offset Address: 0x8404 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RO	pvd_toggle	PVD 触发标识。	0x0

CPU_STATUS

CPU_STATUS 为 CPU 状态寄存器。

Offset Address: 0x9000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RO	cpu_pc_valid	CPU PC 值有效状态。 0：PC 值无效； 1：PC 值有效。	0x0
[30:6]	-	reserved	保留。	0x0000000
[5]	RO	cpu_sleep_mode	CPU SLEEP 状态。 0：非 sleep 状态； 1：sleep 状态。	0x0
[4]	RO	cpu_debug_mode	CPU DEBUG 状态。 0：非 debug 状态；	0x0

Bits	Access	Name	Description	Reset
			1：debug 状态。	
[3]	RO	cpu_hard_fault_mode	CPU HARD FAULT 状态。 0：非 hard_fault 状态; 1：hard_fault 状态。	0x0
[2]	RO	cpu_lockup_mode	CPU LOCKUP 状态。 0：非 lockup 状态; 1：lockup 状态。	0x0
[1:0]	RO	reserved	保留。	0x0

10 直接存储器访问控制器(DMA)

10.1 概述

DMA（Direct Memory Access）是一种高速的数据传输操作，不通过 CPU，直接在存储器和外设、外设和外设、存储器和存储器之间进行数据传输，减轻 CPU 的负担并减少了 CPU 中断处理开销。

10.2 特性

MCU 中 DMA 有如下特性：

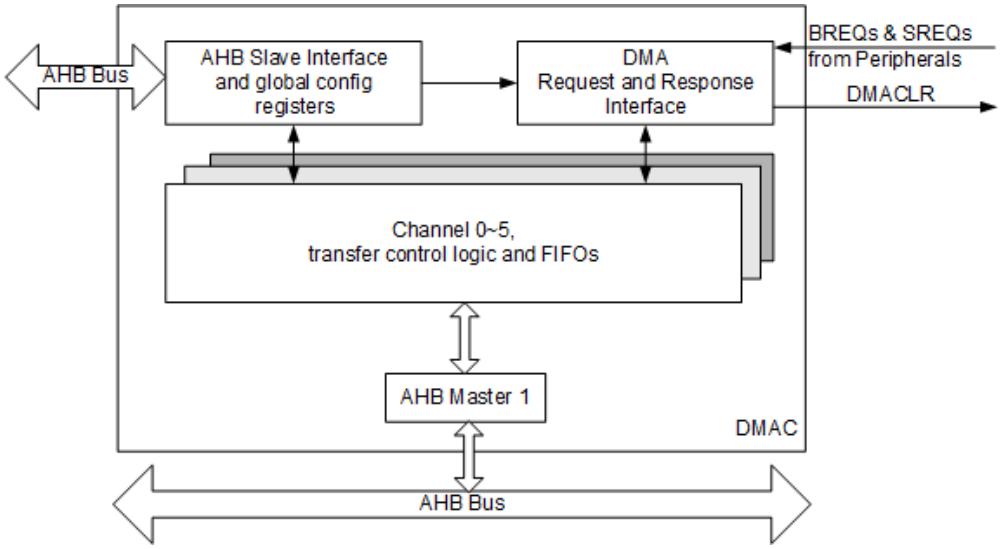
- 支持 8bit、16bit、32bit 数据位宽传输。
- 支持 1 个位宽为 32bit 的 Master 总线接口用于数据传输。
- 支持 6 个 DMA 通道，每个通道均支持存储器到存储器、存储器到外设、外设到存储器、外设到外设的传输。
- 支持 4 级优先级，每个通道可配置不同的优先级。
- 支持 Burst 传输，每个通道可配置 Burst 传输的个数。
- 支持源地址和目的地址自动递增或不递增。
- 支持 DMA 流控和外设流控。
- 支持链表。

10.3 功能描述

10.3.1 功能框图

DMA 的功能框图如图 10-1 所示。

图10-1 DMA 功能框图



说明

- DMA 通道优先级可配置，相同优先级配置的通道固定优先从高到低对应的通道号依次为 0～5。当来自 2 个外设的 DMA 请求同时有效时，优先级高的通道先开始传输。软件须在使能通道前配置通道的优先级，一旦通道被使能且未完成传输，禁止修改通道的优先级，否则可能导致数据丢失。
- 每个 DMA 通道的 FIFO 大小为 1 x 32bit。

DMA 的每一个通道都内含一组传输控制逻辑和一个 FIFO，传输控制逻辑自动完成以下过程：

步骤 1 根据通道的流控配置，判断能否开始传输。若满足开始传输的条件，则进行不同通道之间的优先级仲裁。优先级高的通道先开始传输。

步骤 2 等待源设备的 DMA 请求。

步骤 3 从软件配置的源地址读取数据。

步骤 4 缓存到通道内部的 FIFO 中。

步骤 5 等待目的设备的 DMA 请求。

步骤 6 从通道 FIFO 取出数据。

步骤 7 写入到软件配置的目的地址。

步骤 8 循环以上步骤，直到 DMA 已完成软件配置的传输总长度的搬移。

----结束

说明

源地址可以不在源设备的地址空间内，同理，目的地址也可以不在目的设备的地址空间内。

10.3.2 DMA 通道的传输

10.3.2.1 DMA 传输类型和流控

DMA 的每个通道，根据源设备和目的设备的不同，均支持四种传输类型，分别是存储器到存储器、存储器到外设、外设到存储器和外设到外设。

须知

存储器指的是地址空间映射的 RAM，或者地址空间映射的外设寄存器。

存储器到存储器的传输，建议使用通道 4、5 来执行。以避免总线上其他 AHB master 较长时间无法获得总线。

外设的请求线序号见表 10-1。需要注意的是部分外设没有 DMA 请求线，但是外设相关寄存器已经映射到了地址空间（即 DMA 可以寻址并访问），则这些外设相关寄存器可以视为是存储器，需要配置为存储器到存储器的传输类型。

DMA 每个通道均支持两种流控模式，分别是 DMA 流控和外设流控。外设是否支持外设流控，请参见各外设章节。

四种传输类型和两种流控模式一共形成 8 种组合，配置 DMA_Cn_CONFIG[flow_ctrl] 来选择 8 种组合中的某一种，详见表 10-1，8 种组合详细描述如下：

- 存储器到存储器，DMA 流控：源设备和目的设备均为存储器，存储器的 DMA 请求被视为始终有效。DMA 控制了传输流程，当已传输数据的大小 = 总传输数据的大小（源设备传输位宽 × trans_size），DMA 完成通道传输，通道自动关闭。
- 存储器到外设，DMA 流控：源设备是存储器，其 DMA 请求被视为始终有效。目的设备是外设，其 DMA 请求由外设请求线发出。DMA 控制了传输流程，当已传输数据的大小=总传输数据的大小（源设备传输位宽 × trans_size），DMA 完成通道传输，通道自动关闭。

- 外设到存储器，DMA 流控：源设备是外设，其 DMA 请求由外设请求线发出。目的设备是存储器，存储器的 DMA 请求被视为始终有效。DMA 控制了传输流程，当已传输数据的大小 = 总传输数据的大小（源设备传输位宽 × trans_size），DMA 完成通道传输，通道自动关闭。
- 外设到外设，DMA 流控：源设备是外设，目的设备也是外设，两者的 DMA 请求由外设请求线发出。DMA 控制了传输流程，当已传输数据的大小 = 总传输数据的大小（源设备传输位宽 × trans_size），DMA 完成通道传输，通道自动关闭。
- 存储器到外设，外设流控：源设备是存储器，其 DMA 请求被视为始终有效。目的设备是外设，其 DMA 请求需要具体的外设经过其请求线发出。外设控制了传输流程，通道传输的数据大小由外设决定，与 trans_size 无关，外设流控请求线有效时，对应最后 1 次 DMA Single 传输，之后关闭通道。
- 外设到存储器，外设流控：源设备是外设，其 DMA 请求具体的外设经过其请求线发出。目的设备是存储器，存储器的 DMA 请求被视为始终有效。外设控制了传输流程，通道传输的数据大小由外设决定，与 trans_size 无关，外设流控请求线有效时，对应最后 1 次 DMA Single 传输，之后关闭通道。
- 外设到外设，目的外设流控：源设备是外设，目的设备也是外设，两者的 DMA 请求由外设请求线发出。目的外设控制了传输流程，通道传输的数据大小由目的外设决定，与 trans_size 无关，目的外设流控请求线有效时，对应最后 1 次 DMA Single 传输，之后关闭通道。
- 外设到外设，源外设流控：源设备是外设，目的设备也是外设，两者的 DMA 请求均需要具体的外设经过其请求线发出。源外设控制了传输流程，通道传输的数据大小由源外设决定，与 trans_size 无关，源外设流控请求线有效时，对应最后 1 次 DMA Single 传输，之后关闭通道。

表10-1 传输类型和流控模式

传输类型	流控模式	源设备的 DMA 请求	目的设备的 DMA 请求	传输数据的总大小	DMA 通道完成传输，通道关闭
存储器到存储器	DMA 流控	无请求线，始终有效。	无请求线，始终有效。	源设备传输位宽 × trans_size	已传输数据的大小=传输数据的总大小
存储器到外设	DMA 流控	无请求线，始终有效。	来自目的外设请求线。		
外设到存储器	DMA 流控	来自源外设请求线。	无请求线，始终有效。		

传输类型	流控模式	源设备的 DMA 请求	目的设备的 DMA 请求	传输数据的总大小	DMA 通道完成传输，通道关闭
外设到外设	DMA 流控	来自外设请求线。	来自目的外设请求线。		
存储器到外设	外设流控	无请求线，始终有效。	来自目的外设请求线。	由目的外设决定	来自目的外设的流控请求线，对应最后 1 次 DMA Single 传输。
外设到存储器	外设流控	来自源外设请求线。	无请求线，始终有效。	由源外设决定	来自源外设的流控请求线，对应最后 1 次 DMA Single 传输。
外设到外设	源外设流控	来自源外设请求线。	来自目的外设请求线。	由源外设决定	来自源外设的流控请求线，对应最后 1 次 DMA Single 传输。
外设到外设	目的外设流控	来自源外设请求线。	来自目的外设请求线。	由目的外设决定	来自目的外设的流控请求线，对应最后 1 次 DMA Single 传输。

10.3.2.2 外设请求线（SREQ、BREQ 和外设流控请求线）

外设利用 DMA 硬件请求线向 DMA 发起数据传输的请求。软件配置 DMA_Cn_CONFIG[flow_ctrl]调整传输类型，请参见 10.3.2.1 DMA 传输类型和流控章节。

- 存储器没有请求线，其 DMA 请求（SREQ 和 BREQ 请求线）在任意时刻均有效。
- 外设根据自身的配置发起 DMA 请求（SREQ 和 BREQ 请求线）。外设硬件请求线的具体信息，请参见表 10-2。软件可配置 DMA_Cn_CONFIG[src_periph]选择源外设，可配置 DMA_Cn_CONFIG[dest_periph]选择目的外设。

外设请求线分类。根据 DMA 通道的传输类型和传输方向分为：

- 源外设 SREQ 请求线：源外设请求 DMA 从源地址读出 1 个数据，数量固定为 1 个，数据的宽度由 DMA_Cn_CONTROL[swidth]所决定。
- 源外设 BREQ 请求线：源外设请求 DMA 从源地址读出多个数据，数量由 DMA_Cn_CONTROL[sbsize]所决定，数据的宽度由 DMA_Cn_CONTROL[swidth]所决定。

- 目的外设 SREQ 请求线：目的外设请求 DMA 向目的地址写入 1 个数据，数量固定为 1 个，数据的宽度由 DMA_Cn_CONTROL[dwidth]所决定。
- 目的外设 BREQ 请求线：目的外设请求 DMA 向目的地址写入多个数据，数量由 DMA_Cn_CONTROL[dbsize]所决定，数据的宽度由 DMA_Cn_CONTROL[dwidth]所决定。
- 源外设流控请求线：源外设请求 DMA 从源地址读出最后 1 个数据，数量固定为 1 个，数据的宽度由 DMA_Cn_CONTROL[swidth]所决定。该最后 1 个数据写入目的地址后，DMA 通道自动关闭。
- 目的外设流控请求线：目的外设请求 DMA 向目的地址写入最后 1 个数据，数量固定为 1 个，数据的宽度由 DMA_Cn_CONTROL[dwidth]所决定。该最后 1 个数据写入目的地址后，DMA 通道自动关闭。

  当 DMA_Cn_CONFIG[flow_ctrl]配置的传输类型包含外设的时候，软件须通过配置 DMA_Cn_CONFIG[src_periph]和[dest_periph]改变源外设和目的外设。外设请求线和对应外设的关系如表 10-2 所示。
- 例如，外设请求线编号 6 为 UART1 的 RX 通道请求线，外设请求线编号 7 为 UART1 的 TX 通道请求线。若预期使用 DMA 通道 3，将 UART1 接收的数据搬移到 UART1 再发送出去，则应配置外设请求号 6 作为通道 3 的源设备，即配置 DMA_Cn_CONFIG (n=3) [src_periph]为 0x6，配置外设请求号 7 作为通道 3 的目的设备，即 DMA_Cn_CONFIG (n=3) [dest_periph]为 0x7。

表10-2 DMA 的外设请求线分配

外设硬件请求线编号	对应外设
0	I2C0 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
1	I2C0 TX FIFO 的 DMA（SREQ，BREQ）请求线
2	I2C1 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
3	I2C1 TX FIFO 的 DMA（SREQ，BREQ）请求线
4	UART0 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
5	UART0 TX FIFO 的 DMA（SREQ，BREQ）请求线
6	UART1 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
7	UART1 TX FIFO 的 DMA（SREQ，BREQ）请求线
8	UART2 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线

外设硬件请求线编号	对应外设
9	UART2 TX FIFO 的 DMA（SREQ，BREQ）请求线
10	CAPM0 的 DMA（BREQ）请求线
11	CAPM1 的 DMA（BREQ）请求线
12	CAPM2 的 DMA（BREQ）请求线
13	ADC0 的 DMA（SREQ，BREQ）请求线
14	TIMER0 的 DMA（SREQ，BREQ）请求线
15	TIMER1 的 DMA（SREQ，BREQ）请求线
16	TIMER2 的 DMA（SREQ，BREQ）请求线
17	TIMER3 的 DMA（SREQ，BREQ）请求线
18	SPI0 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
19	SPI0 TX FIFO 的 DMA（SREQ，BREQ）请求线
20	SPI1 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
21	SPI1 TX FIFO 的 DMA（SREQ，BREQ）请求线
22	APT0 的 DMA（SREQ，BREQ）请求线
23	APT1 的 DMA（SREQ，BREQ）请求线
24	APT2 的 DMA（SREQ，BREQ）请求线
25	APT3 的 DMA（SREQ，BREQ）请求线
26	GPT0 的 DMA（SREQ，BREQ）请求线
27	GPT1 的 DMA（SREQ，BREQ）请求线
28	GPT2 的 DMA（SREQ，BREQ）请求线
29	GPT3 的 DMA（SREQ，BREQ）请求线
30	UART3 RX FIFO 的 DMA（SREQ，BREQ，外设流控）请求线
31	UART3 TX FIFO 的 DMA（SREQ，BREQ）请求线

说明

存储器没有请求线，其 DMA 请求（SREQ 和 BREQ）是始终有效的。当 DMA 通道的源设备为存储器时，软件无需配置 DMA_Cn_CONFIG[src_periph]。当 DMA 通道的目的设备为存储器时，软件无需配置 DMA_Cn_CONFIG[dest_periph]。

10.3.2.3 Single 传输和 Burst 传输

DMA 通道支持 2 种传输，分别为：

- Single 传输（对应外设 SREQ 请求线）
  - SREQ 请求根据方向，分为源外设 SREQ 请求，目的外设 SREQ 请求。分别表示源外设请求 DMA 从源地址读出 1 个数据，目的外设请求 DMA 向目的地址写入 1 个数据。
  - Single 传输（SREQ 请求线）的长度固定为 1，即 DMA 通道从源地址读取 1 个数据或向目的地址写 1 个数据。例如，当 DMA_Cn_CONTROL[swidth] = DMA_Cn_CONTROL[dwidth]均为 Byte 时，意思是每当源外设发起 1 次 SREQ 请求，DMA 通道从源地址读 1 个 Byte 数据，每当目的外设发起 1 次 SREQ 请求，DMA 通道向目的地址写 1 个 Byte 数据。
- Burst 传输（对应外设 BREQ 请求线）
  - BREQ 请求根据方向，分为源外设 BREQ 请求，目的外设 BREQ 请求。分别表示源外设请求 DMA 从源地址读出多个数据，目的外设请求 DMA 向目的地址写入多个数据。
  - Burst 传输的长度为软件预先设定的值。软件可通过配置 DMA_Cn_CONTROL[sbsize]设定通道从源外设读数据的 burst 长度，DMA_Cn_CONTROL[dbsize]设定通道向目的外设写数据的 burst 长度。例如，当 sbsize=dbsize=4，swidth=dwidth 为 Half-word 时，意思是每当源外设发起 1 次 BREQ 请求，DMA 从源地址读 4 个 Half-word；每当目的外设发起 1 次 BREQ 请求，DMA 从向目的地址写 4 个 Half-word。
  - 当 sbsize=1 时，Burst 传输退化为 Single 传输。

当外设发起了 DMA 请求（SREQ 和 BREQ 请求线），DMA 自动管理此时进行 Burst 传输还是 Single 传输。规则如下：

说明

目的设备 dbsize 的配置有限制。目的设备 dbsize × dwidth 的整数倍必须等于 trans_size × swidth。否则可能导致数据丢失。

- 若 DMA 流控，当剩余的传输个数 trans_size × swidth 大于等于 Burst 传输的长度 sbsize × swidth

- 若源外设的 BREQ 请求有效，进行 Burst 传输，从源地址读多个数据。
- 源外设的 SREQ 被忽略。
- 若源外设只支持 BREQ，不支持 SREQ。源设备 sbsize 的配置有限制。源设备 sbsize × swidth 的整数倍必须等于 trans_size × swidth。
- 若目的外设的 BREQ 请求有效，进行 Burst 传输，向目的地址写多个数据。
- 目的外设的 SREQ 被忽略。

- 若 DMA 流控，当剩余的传输个数 trans_size × swidth 小于 Burst 传输的长度 sbsize × swidth
  - 若源外设的 SREQ 请求有效，进行 Single 传输，从源地址读 1 个数据。
  - 源外设的 BREQ 被忽略。
  - 若目的外设的 BREQ 请求有效，进行 Burst 传输，向目的地址写入剩余数据。
  - 目的外设的 SREQ 被忽略。
- 若外设流控，当外设的流控请求线有效时，DMA 完成最后 1 次 DMA Single 传输，然后 DMA 的通道完成，通道自动关闭。

每当源外设发起 1 次 DMA 请求（通过外设请求线），DMA 按照规则进行传输后，DMA 向源设备发出 DMA 请求清除信号 DMACLR，应答源设备的 DMA 请求。

每当目的外设发起 1 次 DMA 请求（通过外设请求线），DMA 按照规则进行传输后，DMA 向目的设备发出 DMA 请求清除信号 DMACLR，应答目的设备的 DMA 请求。

10.3.2.4 DMA 通道的传输位宽

DMA 通道的传输位宽：分为源地址的传输位宽，目的地址的传输位宽。分别表示从源地址读数据时的数据宽度，向目的地址写数据的数据宽度。软件可通过配置 DMA_Cn_CONTROL[swidth]和[dwidth]来选择传输位宽是 Byte，Half-word 或者 Word。

10.3.2.5 DMA 通道的传输个数

DMA 通道的传输个数，可通过配置 DMA_Cn_CONTROL[trans_size]设置。例如：

- 当 trans_size = 8，swidth=dwidth 为 Byte 时，意思是 DMA 传输个数为 8 个 Byte。
- 当 trans_size = 8，swidth=dwidth 为 Half-word 时，意思是 DMA 传输个数为 8 个 Half-word。
- 当 trans_size = 8，swidth=dwidth 为 Word 时，意思是 DMA 传输个数为 8 个 Word。

- 当 trans_size = 8，swidth 为 Byte，dwidth 为 Word 时，意思是 DMA 传输个数为 8 个 Byte（即 2 个 Word）。

须知

如果源地址的传输位宽小于目的地址的传输位宽，那么 DMA_Cn_CONTROL[trans_size]的值乘以源地址的传输位宽必须是目的地址的传输位宽的整数倍。否则，可能导致数据的丢失。

10.3.3 DMA 通道的优先级

DMA 支持 6 个通道，6 个通道的配置相互独立，共享 1 个 AHB Master 进行数据传输。当多个通道的源设备或者目的设备的 DMA 请求同时有效时，通道之间会根据优先级进行仲裁。

- 通道的配置相互独立，相关的寄存器：DMA_Cn_SRC_ADDR，DMA_Cn_DEST_ADDR，DMA_Cn_LLI，DMA_Cn_CONTROL，DMA_Cn_CONFIG（n=0，1，2，3，4，5）。
- 通道的优先级可配置，软件通过配置 DMA_Cn_CONFIG[ch_priority]来改变通道的优先级，支持 4 级优先级。相同优先级的通道，序号越低的通道优先级越高。

须知

若通道的任务涉及存储器到存储器的大量数据搬移，该通道配置为高优先级时，可能会导致 AHB Master 被该高优先级的通道一直占用，从而导致低优先级的通道无法及时搬移数据。所以，针对存储器到存储器的大量数据搬移的任务，建议使用序号较高的通道（如通道 4、5），且配置为较低的优先级。

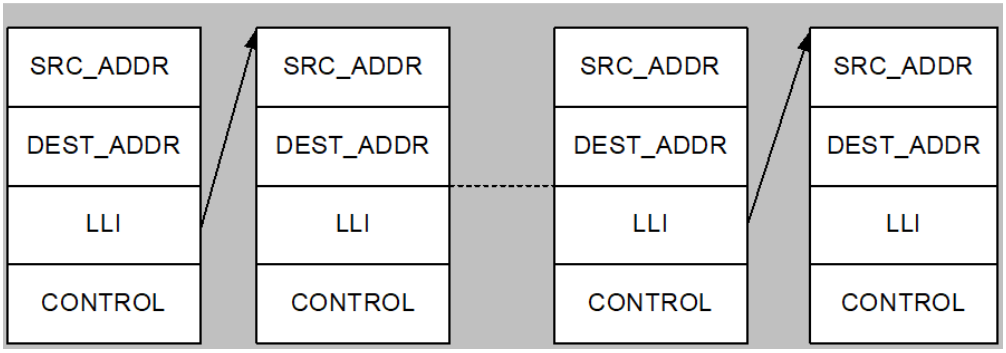
10.3.4 DMA 链表

DMA 的链表节点的数据结构为：

- DMA 通道 n（n=0，1，2，3，4，5）源地址寄存器 DMA_Cn_SRC_ADDR，设置当前链表节点的源地址。
- DMA 通道 n（n=0，1，2，3，4，5）目的地址寄存器 DMA_Cn_DEST_ADDR，设置当前链表节点的目的地址。

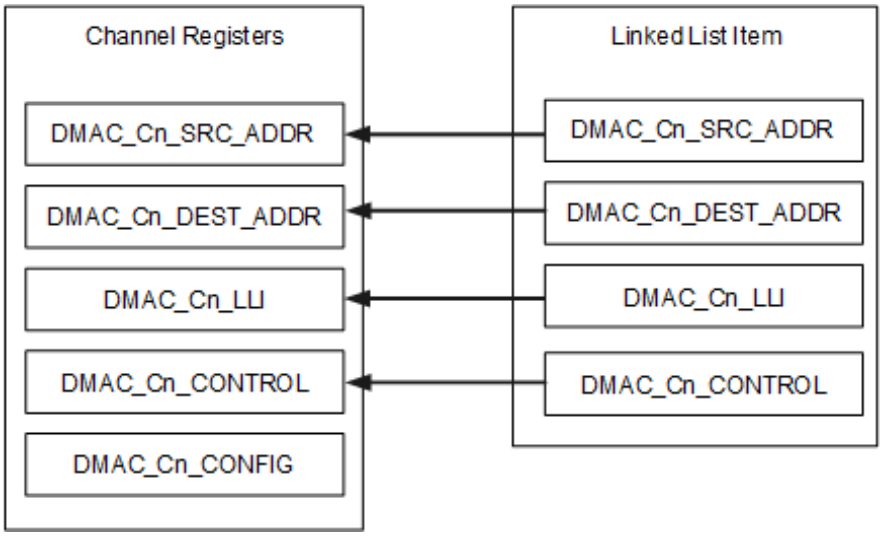
- DMA 通道 n（n=0，1，2，3，4，5）链表信息寄存器 DMA_Cn_LLI，设置下一个链表节点数据结构的存储地址。
- DMA 通道 n（n=0，1，2，3，4，5）控制寄存器 DMA_Cn_CONTROL，设置源/目的设备的位宽、sbsize、dbsize、地址递增以及 trans_size 等参数。

图10-2 DMA 链表单个节点数据结构和多个节点链接示例



链表节点（LLI，Linked List Item）更新通道寄存器示意如图 10-3 所示。

图10-3 LLI 更新通道寄存器示意图



须知

链表单个节点的数据结构由 4 个 Word 组成，存储的地址须连续。为保证链表节点载入的效率，建议链表单个节点保存的地址按 4 Word 对齐。链表的 LLI 字段不能配置大于 0xFFFF_FFF0 的数。

如果 LLI 的值为 0，表示当前节点是链表的链尾，则当本节点对应的数据块全部传完后，该通道就会被关闭。

10.3.5 DMA 工作流程

DMA 基本工作流程如下：

步骤 1 软件选择 DMA 的一个通道用于 DMA 传输，配置该通道的源地址、目的地址、链表的头指针、传输数据个数、源设备和目的设备，流控模式和传输类型，并启动该通道。一旦通道被启动，DMA 通道开始检测源设备和目的设备的 DMA 请求是否有效。

须知

源设备/目的设备指的是某个外设或者存储器，外设的请求根据外设的配置是间断有效的，存储器的请求是始终有效的。

步骤 2 源设备向 DMA 发起 DMA 请求（SREQ 和 BREQ 请求）。

- DMA 通道的源设备可以是存储器或者是外设。
  - 若源设备为存储器，DMA 默认其 DMA 请求（SREQ 和 BREQ 请求）始终有效。
  - 若源设备为外设，需要外设发出 DMA 请求（SREQ 和 BREQ 请求），请求 DMA 通道从源地址读数据。
- DMA 通道的源设备可以更改：配置 DMA_Cn_CONTROL[src_periph]更改具体的外设，请参见 10.3.2.2 外设请求线（SREQ、BREQ 和外设流控请求线）章节。

步骤 3 DMA 通道应答源设备的 DMA 请求，从源地址读取数据并存入通道内部的 FIFO 中。

- 源地址可更改，通过配置 DMA_Cn_SRC_ADDR 更改。源地址可以不在源设备的地址空间内。
- 读数据的宽度（Byte，Half-word，Word）由源设备的传输位宽配置所决定，通过配置 DMA_Cn_CONTROL[swidth]更改，请参见 10.3.2.4 DMA 通道的传输位宽章节。
- 若源设备的 SREQ 请求有效，DMA 通道从源地址读出 1 个数据，应答该 SREQ 请求。
  - 读数据的数量固定为 1，不可配置更改。
- 若源设备的 BREQ 请求有效，DMA 从源设备读出多个数据，应答该 BREQ 请求。

- 读数据的数量可配置更改，通过配置 DMA_Cn_CONTROL[sbsize]更改，请参见 10.3.2.1 DMA 传输类型和流控章节。

步骤 4 目的设备向 DMA 发起 DMA 请求（SREQ 和 BREQ 请求）。

- DMA 通道的目的设备可以是存储器或者是外设。
  - 若目的设备为存储器，DMA 默认其 DMA 请求（SREQ 和 BREQ 请求）始终有效。
  - 若目的设备为外设，需要外设发出 DMA 请求（SREQ 和 BREQ 请求），请求 DMA 通道向目的地址写数据。
- DMA 通道的目的设备可以更改：通过配置 DMA_Cn_CONFIG[dest_periph]更改，请参见 10.3.2.2 外设请求线（SREQ、BREQ 和外设流控请求线）章节。

步骤 5 DMA 通道应答目的设备的 DMA 请求，从通道内部的 FIFO 中取出数据并写入目的地址。

- 目的地址可更改，通过配置 DMA_Cn_DEST_ADDR 更改。目的地址可以不在目的设备的地址空间内。
- 写数据的宽度（Byte，Half-word，Word）由目的设备的传输位宽配置所决定，通过配置 DMA_Cn_CONTROL[dwidth]更改，请参见 10.3.2.4 DMA 通道的传输位宽章节。
- 若目的设备的 SREQ 请求有效，DMA 通道向目的地址写入 1 个数据，应答该 SREQ 请求。
  - 写数据的数量固定为 1，不可配置更改。
- 若目的设备的 BREQ 请求有效，DMA 通道向目的地址写入多个数据，应答该 BREQ 请求。
  - 写数据的数量可配置更改，通过配置 DMA_Cn_CONTROL[dbsize]更改，请参见 10.3.2.1 DMA 传输类型和流控章节。

步骤 6 步骤 2、步骤 3 和步骤 4、步骤 5 可能是并发执行的，因为源设备和目的设备有可能同时发起 DMA 请求。当出现 DMA 通道 FIFO 被写满而目的设备的 DMA 请求仍未产生、或者 DMA 通道 FIFO 被读空而源设备的 DMA 请求仍未产生，DMA 自动阻塞源设备或目的设备的 DMA 请求，直到相应的 FIFO 空满状态解除为止。在 DMA 与源设备、目的设备的多次交互过程中，DMA 重复执行步骤 2～步骤 5，直到软件配置的传输总长度全部完成传输时，发出传输完成中断（该中断可被屏蔽）。此时，如果寄存器 DMA_Cn_LLI 不为 0，则以该寄存器的值为地址读取链表节点，并将读取值依次载入 DMA_Cn_SRC_ADDR、DMA_Cn_DEST_ADDR、DMA_Cn_LLI 以及

DMA_Cn_CONTROL 这四个寄存器（如图 10-2 所示），然后回到步骤 2。若寄存器 DMA_Cn_LLI 的值为 0 则停止当前的 DMA 传输，通道自动关闭，传输过程结束。

----结束

10.4 典型配置

10.4.1 时钟门控

当出现以下情况时，可以通过软件关闭 DMA，并关断 DMA 时钟以降低功耗：

- DMA 的各个通道均处于空闲状态且没有 DMA 传输请求。
- DMA_Cn_CONFIG[ch_en]等于 0，DMA 所有通道被关闭。

关断 DMA 时钟的步骤如下：

步骤 1 向 DMA_Cn_CONFIG[ch_en]写 0，关闭 DMA 通道。

步骤 2 向 DMA_CONFIG[dma_enable]写 0，不使能 DMA。

步骤 3 向 CRG 寄存器 PERI_CRG192[dma_cken]写 0，关闭 DMA 模块时钟门控，最终关闭 DMA 时钟。

步骤 4 当需要使用 DMA 进行数据传输时，重新打开时钟后启动 DMA。

----结束

10.4.2 初始化

初始化 DMA 步骤如下：

步骤 1 写寄存器 DMA_CONFIG，向 DMA_CONFIG[dma_enable]写 1，启动 DMA。

步骤 2 写寄存器 DMA_INT_ERR_CLR 和 DMA_INT_TC_CLR 的所有位为 1，清除所有中断状态。

步骤 3 写寄存器 DMA_SYNC 相应位为 0，设置需要进行同步的外设硬件请求。

步骤 4 依次配置并关闭各个通道。向每个通道的 DMA_Cn_CONFIG[ch_en]写 0，关闭通道。

----结束

10.4.3 启动通道

DMA 初始化完成之后，需要配置并启动 DMA 通道，才可以使用 DMA 来进行数据传输。通道的配置启动步骤如下：

步骤 1 读 DMA_ENABLED_CHNS 寄存器，找出处于非传输状态的通道，并从中选择一个通道用于配置。

步骤 2 向寄存器 DMA_INT_ERR_CLR 和 DMA_INT_TC_CLR 的相应位写 1，清除选定通道的中断状态。

步骤 3 写通道寄存器 DMA_Cn_SRC_ADDR，设置源设备访问首地址。

步骤 4 写通道寄存器 DMA_Cn_DEST_ADDR，设置目的设备访问首地址。

步骤 5 写通道寄存器 DMA_Cn_LLI，设置链表信息，如果配置通道用于单块数据传输，则将通道寄存器 DMA_Cn_LLI 配置为 0；如果配置通道用于链表数据传输，则将通道寄存器 DMA_Cn_LLI 配置为链表的头指针。

步骤 6 写通道寄存器 DMA_Cn_CONTROL，设置源/目的设备的位宽、BURST 长度、地址是否递增以及传输长度等参数。

步骤 7 写通道寄存器 DMA_Cn_CONFIG，设置本通道的 DMA 请求信号、通道的流控方式及中断屏蔽。

步骤 8 写通道寄存器 DMA_Cn_CONFIG[ch_en]为 1，启动该通道。

----结束

10.4.4 中断处理

DMA 通道传输完成之后或传输过程中出现错误，都会上报中断给 CPU。中断程序的处理步骤如下：

步骤 1 读中断状态寄存器 DMA_INT_STAT，找出发出中断请求的通道。当多个通道同时发出中断请求时，先服务优先级最高的中断。

步骤 2 读寄存器 DMA_INT_TC_STAT，比较选定的位是否为 1，以确定对应通道发出的中断是否是传输完成中断。若是，则转到步骤 4 执行；否则转到步骤 3 继续执行。

步骤 3 读寄存器 DMA_INT_ERR_STAT，比较选定的位是否为 1，以确定对应通道发出的中断是否是错误中断。若是，则转到步骤 5 执行；否则退出中断处理。

步骤 4 处理传输完成中断，可分为以下几个子步骤：

1. 写寄存器 DMA_INT_TC_CLR，对选定的位写入 1，清除对应通道的中断状态。
2. 处理目的地址中的数据。
3. 退出中断处理。

步骤 5 处理错误中断，可分为以下几个子步骤：

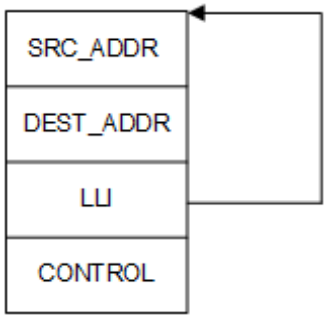
1. 写寄存器 DMA_INT_ERR_CLR，对选定的位写入 1，清除对应通道的中断状态。
2. 记录错误信息，有必要的话重新配置并启动该通道。
3. 退出中断处理。

----结束

10.4.5 DMA 链表实现连续传输

软件可利用 DMA 链表功能实现连续传输，即传输个数是无限个。软件配置 DMA_Cn_LLI，指向链表节点自身地址。DMA 控制器先传输链表节点内配置好的传输个数，此后重新加载同样的链表节点，实现无限个传输个数。如图 10-4 所示。

图10-4 DMA 链表实现连续传输



10.4.6 DMA_Cn_CONTROL 使用说明

通道控制寄存器 DMA_Cn_CONTROL 包含了 DMA 通道控制信息，如传输个数（trans_size）、burst 长度（dbsize，sbsize）、传输位宽（dwidth，swidth）等。

每个寄存器在对应的通道被启动前，都要由软件对其直接编程。当通道被启动后，该寄存器的值在传完 1 个完整的数据块后，从链表节点载入时更新。

当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。在通道停止传输时，可进行该寄存器的读操作。

DMA_Cn_CONTROL 寄存器的 dbsize 及 sbsize 的值与其对应的 BURST 长度如表 10-3 所示。

表10-3 dbsize 及 sbsize 的值与其对应的 burst 长度

dbsize 或 sbsize 的值	BURST 长度
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

DMA_Cn_CONTROL 寄存器的 dwidth 和 swidth 的值与其对应传输位宽如表 10-4 所示。

表10-4 dwidth 和 swidth 的值与其对应传输位宽

swidth 或 dwidth 的值	传输位宽
000	Byte（8bit）
001	Half-word（16bit）
010	Word（32bit）
011	reserved
100	reserved
101	reserved
110	reserved
111	reserved

配置寄存器 DMA_Cn_CONTROL 时需注意：

- 当源设备的传输位宽小于目的设备传输位宽时，源设备的传输位宽与 trans_size 的乘积必须等于目的设备传输位宽的整数倍，否则 FIFO 中的数据将会滞留并丢失。
- swidth 和 dwidth 字段不能设置为未定义的位宽。
- 当该通道是 DMA 流控，传输数据的总大小 = 源设备的传输位宽 × trans_size，trans_size 被写 0 会导致 DMA 不发生任何传输动作，软件应负责关闭此 DMA 通道并对此通道重新编程。当该通道的流控是外设流控，trans_size 字段不被使用，传输数据的总大小由外设决定，请参见 10.3.2.1 DMA 传输类型和流控。
- 不能对 DMA_Cn_CONTROL 寄存器进行普通的写/读测试。因为 trans_size 字段不是一个普通的可写入并读回相同值的寄存器字段。当写入时，该字段相当于一个控制寄存器，决定了 DMA 应传输多少个数据；当读回时，该字段则相当于一个状态寄存器，返回剩下的待传输数据个数（以源设备传输位宽为单位）。
- 当 trans_size 字段的设置值大于源设备或目的设备中的 FIFO 的深度（是外设内部的 FIFO，不是 DMA 内部的 FIFO），则 DMA 的源地址或目的地址必须被设为不递增模式，否则有可能导致外设内部的 FIFO 溢出。

10.4.7 DMA_Cn_CONFIG 使用说明

表 10-5 描述了 DMA_Cn_CONFIG 寄存器的 flow_ctrl 字段对应的流控和传输类型。

表10-5 流控制器及传输类型位定义

bit 值	传输类型	流控
000	存储器至存储器	DMA
001	存储器至外设	DMA
010	外设至存储器	DMA
011	源外设至目的外设	DMA
100	源外设至目的外设	目的外设
101	存储器至外设	目的外设
110	外设至存储器	源外设
111	源外设至目的外设	源外设

10.5 DMA 寄存器概览

DMA 寄存器概览如表 10-6 所示。

表10-6 DMA 寄存器概览（基址是 0x1C00_0000）

偏移地址	名称	描述	页码
0x0000	DMA_INT_STAT	DMA 中断状态寄存器	DMA_INT_STAT
0x0004	DMA_INT_TC_STAT	DMA 传输完成中断状态寄存器	DMA_INT_TC_STAT
0x0008	DMA_INT_TC_CLR	DMA 传输完成中断清除寄存器	DMA_INT_TC_CLR
0x000C	DMA_INT_ERR_STAT	DMA 错误中断状态寄存器	DMA_INT_ERR_STAT
0x0010	DMA_INT_ERR_CLR	DMA 错误中断清除寄存器	DMA_INT_ERR_CLR
0x0014	DMA_RAW_INT_TC_STAT	DMA 原始传输完成中断寄存器	DMA_RAW_INT_TC_STAT
0x0018	DMA_RAW_INT_ERR_STAT	DMA 原始错误中断寄存器	DMA_RAW_INT_ERR_STAT
0x001C	DMA_ENABLED_CHNS	DMA 通道使能状态寄存器	DMA_ENABLED_CHNS
0x0030	DMA_CONFIG	DMA 配置寄存器	DMA_CONFIG
0x0034	DMA_SYNC	DMA 请求线同步使能寄存器	DMA_SYNC
0x0100 + n*0x20	DMA_Cn_SRC_ADDR	DMA 通道 n(n=0~5)源地址寄存器	DMA_Cn_SRC_ADDR
0x0104 +	DMA_Cn_DEST_ADDR	DMA 通道 n(n=0~5)目的地址寄存器	DMA_Cn_DES

偏移地址	名称	描述	页码
n*0x20			T_ADDR
0x0108 + n*0x20	DMA_Cn_LLI	DMA 通道 n(n=0~5)链表信息寄存器	DMA_Cn_LLI
0x010C + n*0x20	DMA_Cn_CONTROL	DMA 通道 n(n=0~5)控制寄存器	DMA_Cn_CONTROL
0x110 + n*0x20	DMA_Cn_CONFIG	DMA 通道 n(n=0~5)配置寄存器	DMA_Cn_CONFIG

DMA 寄存器偏移地址中变量的取值范围和含义如表 10-7 所示。

表10-7 DMA 寄存器偏移地址变量表

变量名称	取值范围	描述
n	0～5	对应 DMA 通道 0，1，2，3，4，5

10.6 DMA 寄存器描述

DMA_INT_STAT

DMA_INT_STAT 为 DMA 中断状态寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_int_stat	通道 5 经屏蔽后的中断状态。 0：未产生中断； 1：已产生中断（传输错误或传输完成）。	0x0
[4]	RO	ch4_int_stat	通道 4 经屏蔽后的中断状态。 0：未产生中断；	0x0

Bits	Access	Name	Description	Reset
			1：已产生中断（传输错误或传输完成）。	
[3]	RO	ch3_int_stat	通道 3 经屏蔽后的中断状态。 0：未产生中断； 1：已产生中断（传输错误或传输完成）。	0x0
[2]	RO	ch2_int_stat	通道 2 经屏蔽后的中断状态。 0：未产生中断； 1：已产生中断（传输错误或传输完成）。	0x0
[1]	RO	ch1_int_stat	通道 1 经屏蔽后的中断状态。 0：未产生中断； 1：已产生中断（传输错误或传输完成）。	0x0
[0]	RO	ch0_int_stat	通道 0 经屏蔽后的中断状态。 0：未产生中断； 1：已产生中断（传输错误或传输完成）。	0x0

DMA_INT_TC_STAT

DMA_INT_TC_STAT 为 DMA 传输完成中断状态寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_int_tc_stat	通道 5 经屏蔽后的传输完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[4]	RO	ch4_int_tc_stat	通道 4 经屏蔽后的传输完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[3]	RO	ch3_int_tc_stat	通道 3 经屏蔽后的传输完成中断状态。 0：未产生中断；	0x0

Bits	Access	Name	Description	Reset
			1：已产生中断。	
[2]	RO	ch2_int_tc_stat	通道 2 经屏蔽后的传输完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[1]	RO	ch1_int_tc_stat	通道 1 经屏蔽后的传输完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[0]	RO	ch0_int_tc_stat	通道 0 经屏蔽后的传输完成中断状态。 0：未产生中断； 1：已产生中断。	0x0

DMA_INT_TC_CLR

DMA_INT_TC_CLR 为 DMA 传输完成中断清除寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	WO	ch5_int_tc_clr	清除通道 5 传输完成中断。 0：不清除； 1：清除。	0x0
[4]	WO	ch4_int_tc_clr	清除通道 4 传输完成中断。 0：不清除； 1：清除。	0x0
[3]	WO	ch3_int_tc_clr	清除通道 3 传输完成中断。 0：不清除； 1：清除。	0x0
[2]	WO	ch2_int_tc_clr	清除通道 2 传输完成中断。	0x0

Bits	Access	Name	Description	Reset
			0：不清除； 1：清除。	
[1]	WO	ch1_int_tc_clr	清除通道 1 传输完成中断。 0：不清除； 1：清除。	0x0
[0]	WO	ch0_int_tc_clr	清除通道 0 传输完成中断。 0：不清除； 1：清除。	0x0

DMA_INT_ERR_STAT

DMA_INT_ERR_STAT 为 DMA 错误中断状态寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_int_err_stat	通道 5 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[4]	RO	ch4_int_err_stat	通道 4 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[3]	RO	ch3_int_err_stat	通道 3 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[2]	RO	ch2_int_err_stat	通道 2 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0

Bits	Access	Name	Description	Reset
[1]	RO	ch1_int_err_stat	通道 1 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[0]	RO	ch0_int_err_stat	通道 0 经屏蔽后的错误中断状态。 0：未产生中断； 1：已产生中断。	0x0

DMA_INT_ERR_CLR

DMA_INT_ERR_CLR 为 DMA 错误中断清除寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	WO	ch5_int_err_clr	清除通道 5 错误中断。 0：不清除； 1：清除。	0x0
[4]	WO	ch4_int_err_clr	清除通道 4 错误中断。 0：不清除； 1：清除。	0x0
[3]	WO	ch3_int_err_clr	清除通道 3 错误中断。 0：不清除； 1：清除。	0x0
[2]	WO	ch2_int_err_clr	清除通道 2 错误中断。 0：不清除； 1：清除。	0x0
[1]	WO	ch1_int_err_clr	清除通道 1 错误中断。 0：不清除；	0x0

Bits	Access	Name	Description	Reset
			1：清除。	
[0]	WO	ch0_int_err_clr	清除通道 0 错误中断。 0：不清除； 1：清除。	0x0

DMA_RAW_INT_TC_STAT

DMA_RAW_INT_TC_STAT 为 DMA 原始传输完成中断寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_raw_int_tc	通道 5 原始传输完成中断状态。 0：未产生传输完成中断； 1：产生传输完成中断。	0x0
[4]	RO	ch4_raw_int_tc	通道 4 原始传输完成中断状态。 0：未产生传输完成中断； 1：产生传输完成中断。	0x0
[3]	RO	ch3_raw_int_tc	通道 3 原始传输完成中断状态。 0：未产生传输完成中断； 1：产生传输完成中断。	0x0
[2]	RO	ch2_raw_int_tc	通道 2 原始传输完成中断状态。 0：未产生传输完成中断； 1：产生传输完成中断。	0x0
[1]	RO	ch1_raw_int_tc	通道 1 原始传输完成中断状态。 0：未产生传输完成中断； 1：产生传输完成中断。	0x0
[0]	RO	ch0_raw_int_tc	通道 0 原始传输完成中断状态。	0x0

Bits	Access	Name	Description	Reset
			0：未产生传输完成中断； 1：产生传输完成中断。	

DMA_RAW_INT_ERR_STAT

DMA_RAW_INT_ERR_STAT 为 DMA 原始错误中断寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_raw_int_err	通道 5 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0
[4]	RO	ch4_raw_int_err	通道 4 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0
[3]	RO	ch3_raw_int_err	通道 3 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0
[2]	RO	ch2_raw_int_err	通道 2 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0
[1]	RO	ch1_raw_int_err	通道 1 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0
[0]	RO	ch0_raw_int_err	通道 0 原始错误中断状态。 0：未产生错误中断； 1：产生错误中断。	0x0

DMA_ENABLED_CHNS

DMA_ENABLED_CHNS 为 DMA 通道使能状态寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RO	ch5_enabled	通道 5 使能状态。 0：禁止； 1：使能。	0x0
[4]	RO	ch4_enabled	通道 4 使能状态。 0：禁止； 1：使能。	0x0
[3]	RO	ch3_enabled	通道 3 使能状态。 0：禁止； 1：使能。	0x0
[2]	RO	ch2_enabled	通道 2 使能状态。 0：禁止； 1：使能。	0x0
[1]	RO	ch1_enabled	通道 1 使能状态。 0：禁止； 1：使能。	0x0
[0]	RO	ch0_enabled	通道 0 使能状态。 0：禁止； 1：使能。	0x0

DMA_CONFIG

DMA_CONFIG 为 DMA 配置寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	dma_enable	DMA 控制器使能。 0：禁止 DMA 控制器； 1：使能 DMA 控制器。	0x0

DMA_SYNC

DMA_SYNC 为 DMA 请求线同步使能寄存器。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	dma_sync_disable	控制是否需要对请求线进行同步，每 bit 对应请求请参见 10.3.2.2 外设请求线（SREQ、BREQ 和外设流控请求线）章节。 0：使能对应外设的 DMA 请求信号同步逻辑； 1：禁止对应外设的 DMA 请求信号同步逻辑。	0x00000000

DMA_Cn_SRC_ADDR

DMA_Cn_SRC_ADDR 为 DMA 通道 n（n=0，1，2，3，4，5）源地址寄存器。

寄存器的偏移地址为 0x100+n*0x20。其中 n 的取值为 0～5，分别对应 DMA 通道 0～5。

每个寄存器在对应的通道被启动前都要由软件对其直接编程。当通道被启动后，该寄存器在下列情况下更新：

- 当源地址递增时。
- 当传完一个完整的数据块后，从链表节点中载入时。
- 当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。对该寄存器的读操作一般是用在通道停止传输的时候，此时读取值显示的是 DMA 读最后一项时的源地址。

Offset Address: 0x0100 + n*0x20(n = 0～5) Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	src_addr	DMA 源地址。	0x00000000

DMA_Cn_DEST_ADDR

DMA_Cn_DEST_ADDR 为 DMA 通道 n（n=0，1，2，3，4，5）目的地址寄存器。

偏移地址为：0x104+n*0x20。其中，n 的取值为 0～5，分别对应 DMA 通道 0～5。

通道目的地址寄存器 DMA_Cn_DEST_ADDR 包含了当前待传数据的目的地址。每个寄存器在对应的通道被启动前，都要由软件对其直接编程。当通道被启动后，该寄存器在下列情况下更新：

- 目的地址递增。
- 传完一个完整的数据块后，从链表节点中载入。
- 当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。在通道停止传输时，读该寄存器，此时读取值显示的是 DMA 写最后一项时的目的地址。

Offset Address: 0x0104 + n*0x20(n = 0～5)　Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	dest_addr	DMA 目的地址。	0x00000000

DMA_Cn_LLI

DMA_Cn_LLI 为 DMA 通道 n（n=0，1，2，3，4，5）链表信息寄存器。

偏移地址为：0x108+n*0x20。其中，n 的取值为 0～5，分别对应 DMA 通道 0～DMA 通道 5。详细内容请参见 10.3.4 DMA 链表章节。

Offset Address: 0x0108 + n*0x20(n = 0～5)　Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RW	ll_item	下一个链表节点地址的[31:2]位，地址位[1:0]为 0。要求链表地址 4 字节对齐。	0x00000000

Bits	Access	Name	Description	Reset
[1:0]	RO	reserved	保留，写入时必须写 0，读出时应屏蔽该位。	0x0

DMA_Cn_CONTROL

DMA_Cn_CONTROL 为 DMA 通道 n（n=0，1，2，3，4，5）控制寄存器。

偏移地址为：0x10C+n*0x20。其中，n 的取值为 0～5，分别对应 DMA 通道 0～DMA 通道 5。

Offset Address: 0x010C+n*0x20(n = 0～5)　Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RW	int_tc_enable	传输完成中断使能位。该位用于决定当前链表节点是否触发传输完成中断。 0：当前链表节点不触发传输完成中断； 1：当前链表节点触发传输完成中断。	0x0
[30:28]	RO	reserved	保留。	0x0
[27]	RW	dest_incr	目的地址递增。 0：目的地址不递增； 1：目的地址每传一个数就递增一次。 目的设备为外设时目的地址不递增；目的设备为存储器时目的地址递增。	0x0
[26]	RW	src_incr	源地址递增。 0：源地址不递增； 1：源地址每传一个数就递增一次。 源设备为外设时源地址不递增；源设备为存储器时源地址递增。	0x0
[25:24]	RO	reserved	保留。	0x0
[23:21]	RW	dwidth	目的设备传输位宽。 宽于 Master 位宽的传输位宽是非法的。	0x0

Bits	Access	Name	Description	Reset
			目的设备和源设备的位宽可以不一样，硬件自动对数据进行 pack 和 unpack。 dwidth 的值和具体的位宽对应关系如表 10-4 所示。	
[20:18]	RW	swidth	源设备传输位宽。 宽于 Master 位宽的传输位宽是非法的。 目的设备和源设备的位宽可以不一样，硬件自动对数据进行 pack 和 unpack。 swidth 的值和具体的位宽对应关系如表 10-4 所示。	0x0
[17:15]	RW	dbsize	目的设备 burst 长度。 表示目的设备请求 1 次 burst 传输（BREQ），DMA 向目的地址（固定或递增地址）写数据的个数。 若目的设备是外设，内部有 FIFO，burst 长度不能超过其 FIFO 深度，建议配合 FIFO 水线使用。 若目的设备是外设，内部无 FIFO，若配置了目的地址递增，则 burst 长度不能超过外设的地址空间。否则，递增的目的地址会超过该设备的地址空间，若该地址无映射，那么可能导致无法预料的结果。 若目的设备是存储器，若配置了目的地址递增，则 burst 长度的配置需要考虑存储器地址边界。目的地址+burst 长度不能超过其存储器地址边界。否则，目的地址递增会访问到超过该设备的地址空间，若该地址无映射，那么可能导致无法预料的结果。 若目的设备只支持 BREQ，不支持 SREQ。目前设备 dbsize 的配置有限制。目的设备 dbsize * dwidth 的整数倍须等于 trans_size * swidth。 dbsize 的值和具体的传输长度的对应关系如表 10-3 所示。	0x0
[14:12]	RW	sbsize	源设备 burst 长度。 表示源设备请求 1 次 burst 传输（BREQ），DMA 从源地址（固定或递增地址）读数据的个数。 若源设备是外设，内部有 FIFO，burst 长度不能超过	0x0

Bits	Access	Name	Description	Reset
			其 FIFO 深度，建议配合 FIFO 水线使用。 若源设备是外设，内部无 FIFO，若配置了源地址递增，则 burst 长度不能超过外设的地址空间。否则，递增的源地址会超过该设备的地址空间，若该地址无映射，那么可能导致无法预料的结果。 若源设备是存储器，若配置了源地址递增，则 burst 长度的配置需要考虑存储器地址边界。源地址+burst 长度不能超过其存储器地址边界。否则，源地址递增会访问到超过该设备的地址空间，若该地址无映射，那么可能导致无法预料的结果。 若源设备只支持 BREQ，不支持 SREQ。源设备 sbsize 的配置有限制。源设备 sbsize * swidth 的整数倍须等于 trans_size * swidth。 sbsize 的值和具体的传输长度的对应关系如表 10-3 所示。	
[11:0]	RW	trans_size	通过写该寄存器可设定 DMA 传输个数，前提是 DMA 是流控制器。这里 transfer size 表示的源设备待传数据的个数。 读该寄存器可得到在与目的设备相连的总线上已传出的数据个数。 当该通道处于活动状态时，读该寄存器得不到有效信息。这是因为当软件得到读出的寄存器值，该寄存器的值已经随着通道传输改变了。对该寄存器的读操作一般是用在通道被启动后然后又停止传输时。	0x000

DMA_Cn_CONFIG

DMA_Cn_CONFIG 为 DMA 通道 n（n=0，1，2，3，4，5）配置寄存器。

偏移地址为：0x110 + n*0x20。其中，n 的取值为 0～5，分别对应 DMA 通道 0～DMA 通道 5。

该寄存器在新的链表节点被载入时不会被更新。

Offset Address: 0x110 + n*0x20(n = 0 ~ 5) Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	-	reserved	保留。	0x00
[25:24]	RW	ch_priority	通道优先级。相同优先级的通道，序号低的优先。须在使能通道前配置，一旦通道被使能且未完成传输，禁止修改通道优先级，否则可能导致数据丢失。 00：低优先； 01：中优先； 10：高优先； 11：最高优先。	0x0
[23:19]	-	reserved	保留。	0x0
[18]	RW	ch_halt	Halt 位。 0：允许 DMA 请求； 1：忽略后来的 DMA 请求，通道 FIFO 中的内容都被传完。 该位可以和 Active 位以及通道使能位一起用于无数据丢失地关闭一个 DMA 通道。	0x0
[17]	RO	ch_active	Active 位。 0：通道 FIFO 中没有数据； 1：通道 FIFO 中有数据。 该位可以和 Halt 位以及通道使能位一起用于无数据丢失地关闭一个 DMA 通道。	0x0
[16]	RW	ch_lock	Lock 位。 0：禁止总线上 lock 传输； 1：使能总线上 lock 传输。	0x0
[15]	RW	tc_int_msk	传输完成中断屏蔽位。 0：屏蔽本通道的传输完成中断。 1：不屏蔽本通道的传输完成中断。	0x0
[14]	RW	err_int_msk	错误中断屏蔽位。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽本通道的错误中断。 1：不屏蔽本通道的错误中断。	
[13:11]	RW	flow_ctrl	流控及传输类型字段。 该字段用于指定流控制器和传输类型。流控制器可以是 DMA、源设备和目的设备。 传输类型可以是存储器到外设、外设到存储器、外设到外设、存储器到存储器。详细描述如表 10-5 所示。	0x0
[10:6]	RW	dest_periph	目的设备。该字段用于选择一个外设请求信号作为本通道的 DMA 目的设备的请求信号。 如果 DMA 传输的目的设备是存储器则该字段被忽略。	0x0
[5:1]	RW	src_periph	源设备。该字段用于选择一个外设请求信号作为本通道的 DMA 源设备的请求信号。 如果 DMA 传输的源设备是存储器则该字段被忽略。	0x0
[0]	RW	ch_en	通道使能位。读该寄存器可得知本通道当前是处于什么状态（也可通过读寄存器 DMA_ENABLED_CHNS 得到）。 0：关闭通道； 1：启动通道。 通过清零可关闭通道。将该位被清零时，当前的总线传输会继续执行直到完成。然后通道关闭，FIFO 中剩余的数据全部丢失；当最后一个 LLI 完成或传输中出现错误时，通道也会被关闭，同时该位被清零；如果要关闭通道，而又不使通道 FIFO 中的数据丢失，则 Halt 位也必须同时被置位，使通道忽略后来的 DMA 请求。然后必须轮询 active 位，直到其值变为 0，表明通道 FIFO 中不再留有数据。此时才能够清除 enable 位。 通过置位启动通道必须先重新初始化通道，然后才能再次启动通道；若通过简单的置位启动通道，会引发不可预测性的后果。 当刚通过写 DMA_Cn_CONFIG bit[0]位关闭一个通道时，必须要等到轮询到寄存器 DMA_ENABLED_CHNS 中的相应 bit 为 0 后，才能将 DMA_Cn_CONFIG bit[0]重新置位。这	0x0

Bits	Access	Name	Description	Reset
			是因为通道实际的关闭并没有在将 DMA_Cn_CONFIG bit[0] 清零后立即生效。总线 burst 的运行时延时也必须要考虑到。	

11 循环冗余校验计算单元(CRC)

11.1 概述

CRC（Cyclic Redundancy Check）是循环冗余校验计算单元的简称。

CRC 校验通常用来校验数据传输或者存储过程的完整性，是数据通信领域最常用的一种差错校验码。

本 MCU 包含一个 CRC 循环冗余校验计算单元，支持 4 种算法多项式可配，可以被使用以产生 8/16/32bit 数据的 CRC 码值。

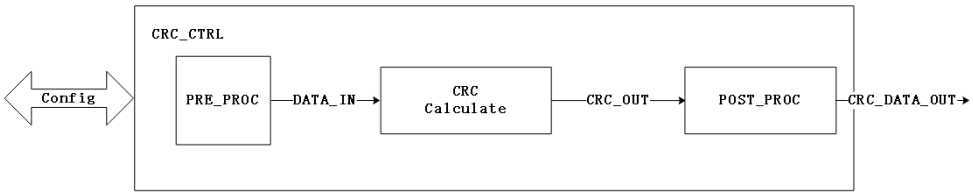
11.2 特性

CRC 模块有以下特点:

- 支持 4 种 CRC 多项式可配（CRC8-07 / CRC16-1021 / CRC16-8005 / CRC32-04C11DB7）。
- 支持 CRC 初值配置。
- 支持 CRC 码值计算。
- 支持 CRC 输入数据按 Byte 反序。
- 支持 CRC 输出数据按有效位宽反序。
- 支持 CRC 输出数据与指定值异或。
- 支持 CRC 输入数据大小端选择。
- 每个时钟周期，CRC 计算 8bit 数据。

11.3 功能描述

图11-1 模块框图



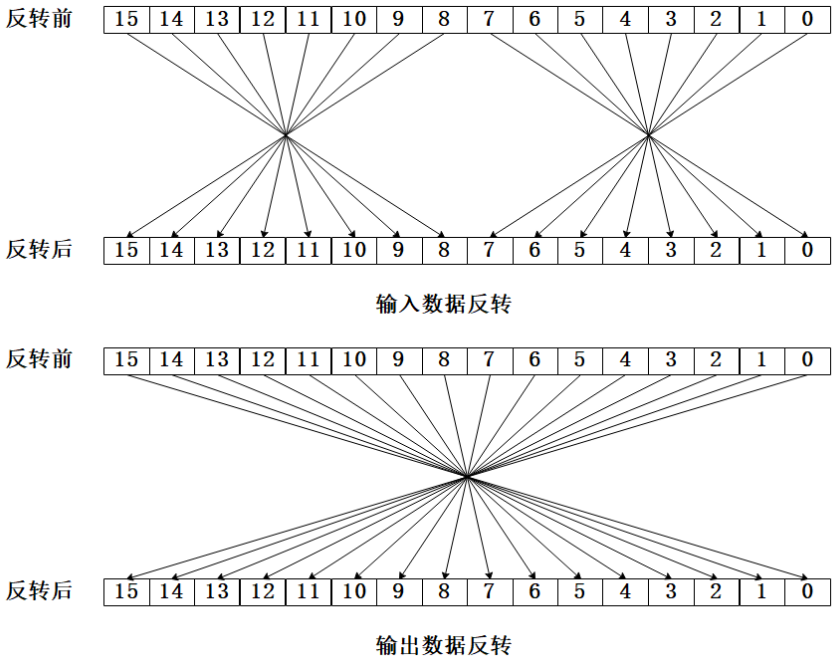
CRC 模块通过总线配置算法多项式、初值等参数，并通过总线读出 CRC 的计算码值。

算法规格

通过多种配置（多项式、异或值、输入/输出数据反序、初值等）的组合，本模块可以实现多种 CRC 标准算法。

输入数据的反序在每个字节内部以比特位为单位进行按位反序，输出数据的反序在整个数据内以比特位为单位进行按位反序，以 16bit 输入输出数据为例，具体如图 11-2 所示。

图11-2 输入输出数据反序示例图



本 CRC 模块可以用于产生 CRC 校验码，具体步骤请参见 11.4.1 CRC 校验码产生章节。

11.4 典型配置

11.4.1 CRC 校验码产生

操作步骤如下：

步骤 1 配置 CRC_CALC_RESET 来软复位 CRC 模块，此软复位只会复位 CRC 内部计算模块，不会复位其他寄存器接口模块。

步骤 2 根据需要的 CRC 算法，配置 CRC_CALC_CFG、CRC_PRE_CFG[crc_pre_endian_mode]、CRC_PRE_CFG[crc_pre_byte_reverse]、CRC_POST_CFG[crc_post_out_reverse] 、 CRC_POST_CFG[crc_post_xor_enable] ，若 crc_post_xor_enable 使能，配置 CRC_POST_XOR_VALUE。

步骤 3 配置 CRC_CALC_INIT_VALUE，向 CRC_CALC_INIT 写 1 加载计算初值。

步骤 4 向 CRC_DATA_IN 写入需要计算的任意位宽数据。

步骤 5 不需要等待任何标志位，直接读出 CRC_OUT 中的数据。读出的数据即为 CRC 产生的校验码。

----结束

须知

- 对于写入任意位宽的数据，建议使用对应变量类型的指针对 CRC_DATA_IN 寄存器的地址进行写操作。
- 为了防止编译器进行优化，导致读出操作发生在写入数据之前，进而导致读出错误数据，建议使用 volatile 关键字声明。
- 大小端模式：默认为小端模式，因为 MCU 内所有数据默认按照小端存储。若开启大端模式时，当输入为 32bit 有效数据，实际 CRC 模块接收到的数据为 {data[7:0],dara[15:8],data[23:16],data[31:24]}。当输入为 16bit 有效数据，实际为 {data[7:0],dara[15:8]}；当输入为 8bit 有效数据，不进行转换。

11.5 CRC_CFG 寄存器概览

CRC_CFG 寄存器概览如表 11-1 所示。

表11-1 CRC_CFG 寄存器概览（基址是 0x00000）

偏移地址	名称	描述	页码
0x000	CRC_CALC_CFG	CRC 计算模块，配置寄存器	CRC_CALC_CFG
0x004	CRC_CALC_RESET	CRC 计算模块软复位寄存器	CRC_CALC_RESET
0x008	CRC_CALC_INIT	CRC 计算模块，初始化寄存器	CRC_CALC_INIT
0x00C	CRC_POST_XOR_VALUE	CRC 后级处理，异或值寄存器	CRC_POST_XOR_VALUE
0x010	CRC_CALC_INIT_VALUE	CRC 计算模块，初始值寄存器	CRC_CALC_INIT_VALUE
0x014	CRC_DATA_IN	CRC 输入数据寄存器	CRC_DATA_IN
0x018	CRC_OUT	CRC 结果寄存器	CRC_OUT

偏移地址	名称	描述	页码
0x01C	CRC_PRE_CFG	CRC 前级处理，配置寄存器	CRC_PRE_CFG
0x020	CRC_POST_CFG	CRC 后级处理，配置寄存器	CRC_POST_CFG

11.6 CRC_CFG 寄存器描述

CRC_CALC_CFG

CRC_CALC_CFG 为 CRC 计算模块，配置寄存器。

Offset Address: 0x000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2:0]	RW	crc_calc_poly	CRC 计算模块，多项式配置： 000：CRC-8，多项式 0x07； 001：保留； 010：CRC-16，多项式 0x8005； 011：CRC-16，多项式 0x1021； 100：CRC-32，多项式 0x04c11d87； 其他：保留。	0x0

CRC_CALC_RESET

CRC_CALC_RESET 为 CRC 计算模块软复位寄存器，复位计算，不复位配置。

Offset Address: 0x004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	W1_PUL	crc_calc_reset	CRC 计算模块软复位，该位域写 1 自动清	0x0

Bits	Access	Name	Description	Reset
	SE		零。复位 CRC 计算，不复位配置。	

CRC_CALC_INIT

CRC_CALC_INIT 为 CRC 计算模块，初始化寄存器。

Offset Address: 0x008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	W1_PULSE	crc_calc_init	CRC 初始值加载信号，该位域写 1 自动清零，CRC 计算初始化。 软件写入该位域，CRC 计算初始化完成后，CRC 从 CRC 初始值（CRC_CALC_INIT_VALUE[crc_calc_init_value]位域）开始计算。 **注意：软件配置该位域后，再写入 CRC_DATA_IN[crc_data_in]位域开始计算。**	0x0

CRC_POST_XOR_VALUE

CRC_POST_XOR_VALUE 为 CRC 后级处理，异或值寄存器。

Offset Address: 0x00C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	crc_post_xor_value	CRC 后级处理，异或值（XOR）。	0x00000000

CRC_CALC_INIT_VALUE

CRC_CALC_INIT_VALUE 为 CRC 计算模块，初始值寄存器。

Offset Address: 0x010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:0]	RW	crc_calc_init_value	CRC 计算模块，初始值。	0x00000000

CRC_DATA_IN

CRC_DATA_IN 为 CRC 输入数据寄存器。

Offset Address: 0x014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	crc_data_in	CRC 输入数据。软件写入需要计算的数据。	0x00000000

CRC_OUT

CRC_OUT 为 CRC 结果寄存器。

Offset Address: 0x018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	crc_out	CRC 结果，超过 CRC 多项式宽度的比特位是 0。	0x00000000

CRC_PRE_CFG

CRC_PRE_CFG 为 CRC 前级处理，配置寄存器。

Offset Address: 0x01C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RW	crc_pre_endian_mode	crc_data_in 数据大小端。 0：小端； 1：大端。	0x0
[0]	RW	crc_pre_byte_reverse	crc_data_in 数据在字节内，按位反序（RefIn）。	0x0

Bits	Access	Name	Description	Reset
			0：不反序; 1：反序。	

CRC_POST_CFG

CRC_POST_CFG 为 CRC 后级处理，配置寄存器。

Offset Address: 0x020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RW	crc_post_out_reverse	CRC 结果在多项式宽度内，按位反序（RefOut）。 0：不反序; 1：反序。	0x0
[0]	RW	crc_post_xor_enable	CRC 结果异或（XOR）。 0：不异或; 1：异或。	0x0

12 捕获模块(CAPM)

12.1 概述

CAPM（Capture Module）捕获模块用于捕获 MCU 管脚输入信号的边沿，记录边沿到来时对应的时间戳，可用于通用数字信号捕获场景，也可实现电机专用场景。

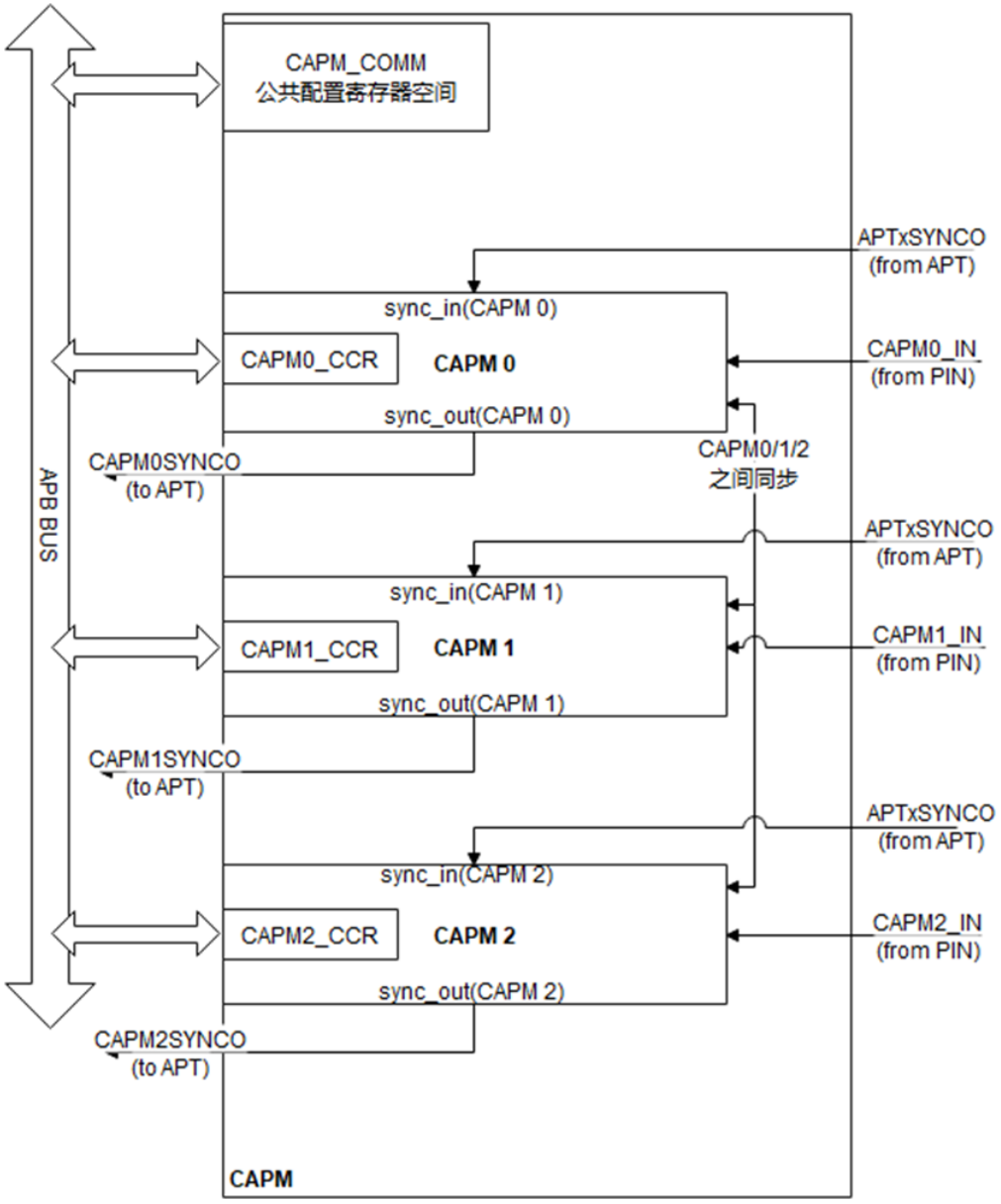
如：霍尔传感器，PWM 波通信等应用。

12.2 特性

- 支持 3 路捕获通道（分别是 CAPM0、CAPM1 和 CAPM2）。
- 支持输入滤波（支持过滤毛刺宽度 1~8192 个时钟周期），输入预分频（2~510 偶数分频）。
- 支持输入电平实时检测。
- 支持对边沿个数的统计。
- 支持对统计的边沿个数进行比较。
- 支持 32bit TSR（Time-Stamp Register）计数器，支持使用分频时钟进行计数。
- 支持每路通道捕获最多 4 个事件，保存最多 4 个事件时间戳，支持捕获溢出检测。
- 支持循环捕获，单轮次捕获。
- 支持时间计数相位同步功能，同步输入（APT 硬件同步、软件同步），同步输出。
- 支持复位工作状态，不复位配置寄存器。
- 支持产生 DMA 请求。
- 支持仿真器调试，暂停捕获通道中的 TSR 计数。

三通道 CAPM 功能框图如图 12-1、图 12-2 所示，CAPM 的 3 路捕获通道功能完全一致。

图12-1 三通道 CAPM 功能模块框图



12.3 功能描述

CAPM 模块支持 3 路捕获通道（分别是 CAPM0、CAPM1 和 CAPM2），每路捕获通道的功能一致，单路捕获通道功能模块图 12-2。3 路捕获通道的寄存器组 CAPM_CCR

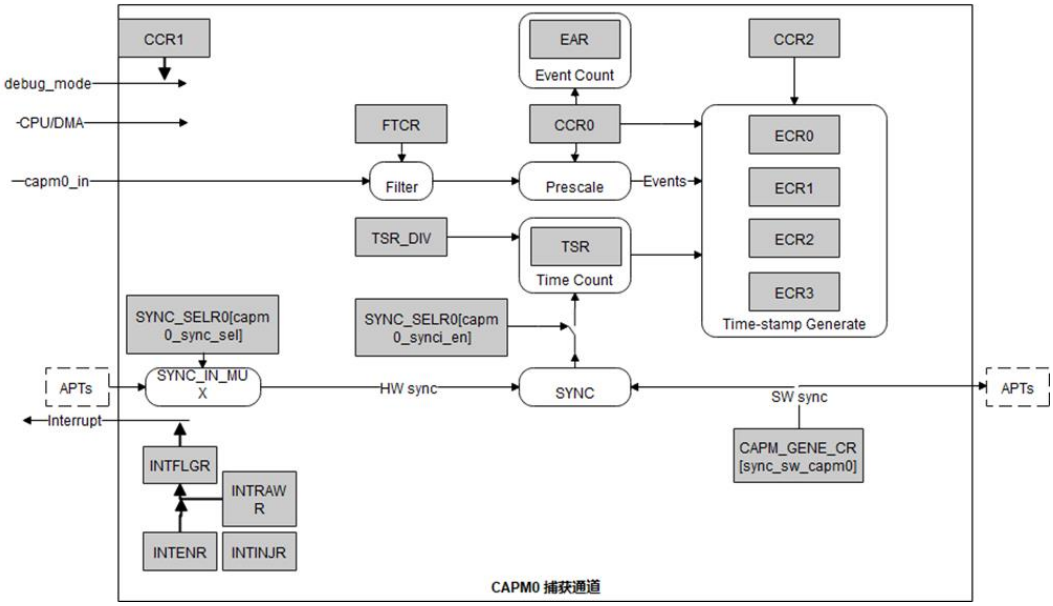
（CAPM Control & Configure Registers）独立，内部偏移地址一致，但是寄存器组的基地址不一致，详见表 12-1。CAPM 公共配置寄存器组 CAPM_COMM（CAPM Common Registers）对所有的 3 路捕获通道生效。

CAPM 模块内部地址分布如表 12-1 所示。

表12-1 CAPM 模块地址空间分配

基地址	CAPM 模块寄存器组
0x14B00000 + 0x1000*x	CAPMx_CCR（CAPMx 捕获通道寄存器组，x=0，1，2）
0x14B03000	CAPM_COMM（CAPM 模块公共配置寄存器组）

图12-2 单通道功能模块详细框图（以 CAPM0 捕获通道为例）



12.3.1 输入滤波和预分频

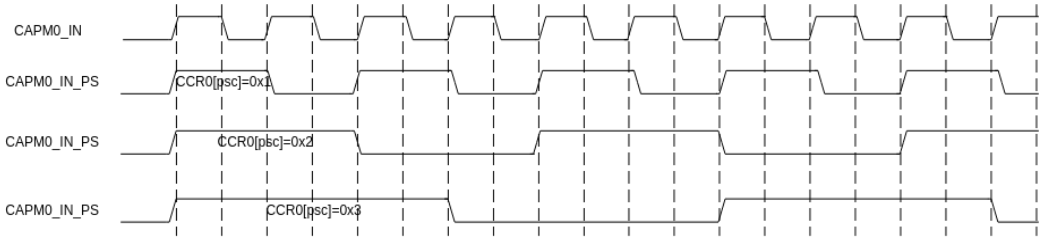
- 支持输入滤波（支持过滤毛刺宽度 1~8192 个时钟周期）：可通过配置 FTCR 使能滤波和调整滤波宽度。

须知

滤波会导致信号延时，延迟的时间与滤波宽度的配置相同。

- 支持输入预分频（2~510 偶数分频）：可通过 CCR0[psc]调整分频。

图12-3 输入预分频图例（以 CAPM0 捕获通道 0 为例）



12.3.2 边沿个数的统计和比较

- 支持边沿个数的统计。
  - 每路捕获通道支持对预分频后的输入信号进行边沿个数的统计。
  - 可配置 CCR0[cnt_edge_sel]选择统计的类型（上升沿、下降沿、双沿），计数结果保存在 EAR 中。
- 支持对统计的边沿个数进行比较，并上报中断。
  - 每路捕获通道支持对统计的边沿个数进行比较，可配置 EAR_CMP 调整比较值。
  - 当 EAR 等于 EAR_CMP 时，比较匹配，产生边沿个数比较匹配原始中断 earcmp_match_raw，并上报边沿个数比较匹配中断 earcmp_match_int。软件须写 EAR 清零后，再对 earcmp_match_raw 写 1 清除中断。
  - 当 EAR 计数溢出时，产生边沿数量计数溢出原始中断 ear_ovf_raw，并上报边沿数量计数溢出中断 ear_ovf_int。软件须写 EAR 清零后，再对 ear_ovf_raw 写 1 清除中断。

12.3.3 TSR 计数器

支持 32bit 位宽 TSR 计数，计数分频，软件读 TSR 时间戳计数器可获取计数值。

- TSR 计数：每路捕获通道支持 TSR 计数，位宽 32bit，可回读 TSR 查询当前计数值。若计数溢出，产生 TSR 溢出中断 tsr_ovf_int。
- TSR 计数的分频时钟：可配置 TSR_DIV 调整用于 TSR 计数的分频时钟。
- TSR 计数的使能：可配置 CAPM_GENE_CR[tsr_stop_capm0]使能 TSR 计数。

支持每路捕获通道最多可以保存 4 个事件时间戳，支持捕获溢出检测。

- 事件：每路捕获通道支持的 4 个事件分别是 EVT0（Event0）、EVT1、EVT2 和 EVT3。
- 捕获边沿：可通过配置 CCR0[evt0pol]、CCR0[evt1pol]、CCR0[evt2pol]和 CCR0[evt3pol]，分别改变 4 个事件的捕获边沿，可以选择捕获输入信号的上升沿还是下降沿。例如：若软件配置 CCR0[evt0pol]为 0，那么 EVT0 捕获输入信号的下降沿，当前 TSR 计数值保存到 ECR0 中。
- 捕获后复位：分别可通过配置 CCR0[evt0rst]、CCR0[evt1rst]、CCR0[evt2rst]、CCR0[evt3rst]分别使能事件 0~3 是否复位 TSR（先捕获后复位）。例如：若软件配置 CCR0[evt0rst]为 1，那么当 EVT0 捕获输入信号时，当前 TSR 计数值保存到 ECR0 中，之后 TSR 被复位清零。
- 捕获使能：可配置 CCR0[ecrlden]使能捕获功能。每个事件捕获成功时将 TSR 的值保存于对应事件 EVT0~3 的时间戳寄存器 ECR0、ECR1、ECR2 和 ECR3 中，并且上报事件中断 evt0_int、evt1_int、evt2_int 和 evt3_int。
- 捕获溢出：若在新的时间戳保存前（也就是 ECR0/1/2/3 更新前），CPU 或 DMA 未能回读 ECR0/1/2/3 的值，则新的时间戳会覆盖老的时间戳，并且产生捕获溢出中断 ecr_ovf_int。

12.3.4 TSR 计数器同步功能

3 路捕获通道的 TSR 计数器支持 APT 硬件同步，支持软件同步。

每路捕获通道（以 CAPM0 为例），可以通过配置 SYNC_SELR0[capm0_synci_en]使能同步功能。

若使能同步功能，当 APT 硬件同步信号有效时，或者配置软件同步时，TSR 的值被同步为 SYNC_PHS 的值，同时重启捕获序列。

若不使能同步功能，那么 APT 硬件同步无效，软件同步也无效。

- APT 硬件同步：每路捕获通道支持 APT 硬件同步（以 CAPM0 为例），可独立选择来自 APT 的硬件同步信号，可通过配置 SYNC_SELR0[capm0_sync_sel]调整。
- 软件同步：每路捕获通道支持软件同步（以 CAPM0 为例），可通过配置 CAPM_GENE_CR[sync_sw_capm0]触发软件同步。

须知

APT 硬件同步和软件同步不冲突，不互斥。

- 同步输出：每路捕获通道选择的 APT 硬件同步信号和软件同步做逻辑或，产生 CAPM 模块的同步输出。

12.3.5 捕获序列和捕获模式

支持事件序号递增的捕获序列，循环捕获，单轮次捕获。

- 捕获序列：按事件 EVT0~3 序号递增的顺序，进行捕获。
- 捕获模式：支持循环捕获和单轮次捕获，可通过配置 CCR2[cap_mode]调整。
  - 循环捕获：配置 CCR2[cap_mode]为 0 设置循环捕获，配置 CCR2[seq_stop]调整循环边界。例如：CCR2 配置为 0x4 时（也就是 EVT2 为捕获循环边界，捕获模式为循环捕获），捕获序列为 ECR0→ECR1→ECR2→ECR0→ECR1→ECR2...如此循环。时间戳保存的序列与事件一一对应：ECR0→ECR1→ECR2→ECR0→ECR1→ECR2...如此循环。
  - 单轮次捕获：配置 CCR2[cap_mode]为 1 设置单轮次捕获，配置 CCR2[seq_stop]调整序列终点。例如：CCR2 配置为 0x5 时（也就是 EVT2 为捕获终点，捕获模式为单轮次捕获），捕获序列为 ECR0→ECR1→ECR2→结束。时间戳保存的序列与事件一一对应：ECR0→ECR1→ECR2。

12.3.6 复位工作状态

支持复位工作状态，不复位配置寄存器。

- 复位工作状态：每路捕获通道支持复位工作状态（以 CAPM0 为例），可通过配置 CAPM_GENE_CR[stat_rst_capm0]复位工作状态。
  - 重启捕获功能，捕获序列从 EVT0 开始。
  - 重启输入预分频功能和时间计数分频功能。
  - 复位 TSR，ECR0~3，EAR，复位 INTFLGR 中断状态和 INTRAWR 原始中断。
  - 不复位其他寄存器。如：CCR0，CCR1，CCR2 和 TSR_DIV 等配置寄存器，INTENR 中断使能寄存器。

12.3.7 触发 DMA 请求

支持触发 DMA 请求，支持 DMA 溢出检测。

- DMA 请求：每路捕获通道支持触发 DMA 请求，可配置 CCR0[dmaevt_sel]调整具体某一个事件用于产生 DMA 请求。

- DMA 溢出：当旧的 DMA 请求未被处理，新的请求又产生时，产生 DMA 请求溢出中断，可回读 INTFLGR[dmareq_ovf_int]查询中断状态。

12.3.8 仿真器调试

支持仿真器接入调试 CAPM 模块。

- 仿真器调试：每路捕获通道均支持仿真器接入并调试。
- 仿真器停止 TSR 计数：暂停 CPU 时同时停止 TSR 计数，可配置相应捕获通道 CAPM_CCR 寄存器组中的 CCR1[emu_stop_en]使能此功能。

12.4 典型配置与部分场景使用建议

说明

- CAPM 模块所有配置都是动态生效的。
- 该章节中，假设 CAPM 的工作时钟 pclk_capm0/1/2 为 200MHz，即采样周期为 5ns，工作在不同时钟时请带入新的周期计算。
- 该章节中，推荐的使用方法并不是测量/计算所需信息的唯一使用方法，请理解并灵活运用。
- 如无特殊说明，示例场景应用所需的每路捕获通道的配置都相同。

12.4.1 缺省配置

缺省配置下，模块配置如下：

- 滤波功能默认使能，过滤 0 个时钟 cycle 宽度的毛刺。
- 输入信号不进行预分频。
- TSR 时间计数缺省为 0 且不计数，计数不分频。
- 捕获终点/循环边界为 EVT3，捕获模式为循环捕获。
- 不使能捕获，不产生事件，不会触发 TSR 复位。
- 同步功能不使能。
- 不会上报任何中断。

12.4.2 通用配置流程（以 CAPM0 捕获通道为例）

前置：在计算周期信息时，软件应当知道当前模块的工作时钟周期。

步骤 1 配置 PERI_CRG160[capm0_srst_req]为 0，解除 CAPM0 通道的复位。

步骤 2 配置 PERI_CRG160[capm0_cken]为 1，打开 CAPM0 通道的时钟门控。

步骤 3 配置 CCR0 和 CCR2（持续生效），配置不分先后顺序。

- 配置时间计数分频 TSR_DIV[tscnt_div]；
- 配置捕获模式 CCR2[cap_mode]；
- 配置捕获序列终点/循环边界 CCR2[seq_stop]；
- 配置输入信号滤波 FTCR[ft_en]和 FTCR[ft_lev]；
- 配置输入信号预分频系数 CCR0[psc]；
- 配置触发 DMA 请求的事件 CCR0[dmaevt_sel]；
- 配置每个事件边沿极性 CCR0[evtxpol] (x=0/1/2/3)；
- 配置每个事件是否触发计数器复位 CCR0[evtxrst] (x=0/1/2/3)。

步骤 4 配置中断使能。

- 中断使能：INTENR[evt0_en]，INTENR[evt1_en]，INTENR[evt2_en]，INTENR[evt3_en]，INTENR[tsr_ovf_en]，INTENR[ecr_ovf_en]，INTENR[earcmp_match_en]，INTENR[ear_ovf_en]，INTENR[dmareq_ovf_en]开启或关闭相应的中断上报；
- 中断查询：读 INTFLGR[evt0_int]，INTFLGR[evt1_int]，INTFLGR[evt2_int]，INTFLGR[evt3_int]，INTFLGR[tsr_ovf_int]，INTFLGR[ecr_ovf_int]，INTFLGR[earcmp_match_int]，INTFLGR[ear_ovf_int]，INTFLGR[dmareq_ovf_int]获悉相应的中断标志；
- 中断清除：对 INTRAWR[evt0_raw]，INTRAWR[evt1_raw]，INTRAWR[evt2_raw]，INTRAWR[evt3_raw]，INTRAWR[tsr_ovf_raw]，INTRAWR[ecr_ovf_raw]，INTRAWR[earcmp_match_raw]，INTRAWR[ear_ovf_raw]，INTRAWR[dmareq_ovf_raw]写清除相应的中断。

说明

如果需要使用软件注入中断，则对 INTINJR[evt0_inj]，INTINJR[evt1_inj]，INTINJR[evt2_inj]，INTINJR[evt3_inj]，INTINJR[tsr_ovf_inj]，INTINJR[ecr_ovf_inj]，INTINJR[earcmp_match_inj]，INTINJR[ear_ovf_inj]，INTINJR[dmareq_ovf_inj]写注入相应的中断。

边沿计数比较匹配中断的清除：软件须写 EAR 清零后，再对 INTRAWR[earcmp_match_raw]写 1，以清除边沿计数比较匹配中断。

边沿计数溢出中断的清除：软件须写 EAR 清零后，再对 INTRAWR[ear_ovf_raw]写 1，以清除边沿计数溢出中断。

步骤 5 配置 CAPM_GENE_CR[tsr_stop_capm0]使能 CAPM0 捕获通道的 TSR 计数，配置 CCR0[ecrlden]开启捕获序列。

- 如果是单轮次模式，配置 CAPM_GENE_CR[restart_capm0]为 1，进行新的一轮捕获。
- 如果是循环模式，配置 CAPM_GENE_CR[stat_rst_capm0]为 1，恢复初始工作状态。

步骤 6 软件有如下 3 种方式得知事件 0/1/2/3 是否有效：

- 查询方式：查询 ECSEQR[nxtldecr]可知捕获序列中的下一个事件（也就是下一个将被更新的事件的时间戳寄存器 ECR）。当 nxtldecr = x 时，表示下一个事件是 EVTx（x=0，1，2，3）。因为捕获序列是顺序递增的，也就是说事件 x ~ 0 的时间戳寄存器是有效可读的。
- 中断方式：配置事件的中断使能 INTENR[evtx_en]（x=0，1，2，3），在相应的事件有效（也就是事件的时间戳寄存器 ECR 被更新）时，上报中断。因为捕获序列是顺序递增的，也就是在中断状态 INTFLGR[evtx_int]（x=0，1，2，3）有效时，事件 x ~ 0 的时间戳寄存器是有效可读的。
- DMA 方式：配置 DMA 请求选择 CCR0[dmaevt_sel]，选择相应的事件有效（也就是事件的时间戳寄存器 ECR 被更新）时，产生 DMA 请求。

须知

如果事件的时间戳寄存器的旧值未被读取，而该事件有效，即新的值被更新，此时产生捕获溢出中断（若使能了捕获溢出中断）。

步骤 7 事件有效后，读取事件的时间戳寄存器 ECR0~3，获取时间戳。

步骤 8 CPU 计算出需要的信息。

----结束

12.4.3 去初始化流程（以 CAPM0 捕获通道为例）

步骤 1 配置 CAPM_GENE_CR[tsr_stop_capm0]为 1，停止 CAPM0 通道 TSR 计数。

步骤 2 配置 PERI_CRG160[capm0_cken]为 0，关闭 CAPM0 通道的时钟门控。

步骤 3 配置 PERI_CRG160[capm0_srst_req]为 1，复位 CAPM0 通道，包括 CAPM0_CCR 寄存器的配置信息和 CAPM0 通道的工作状态。

说明

配置 PERI_CRG160[capm0_srst_req]为 1，并不会复位 CAPM_COMM 寄存器的配置。

----结束

12.4.4 TSR 计数器同步功能

12.4.4.1 硬件同步 CAPM（以 CAPM0 为例）

步骤 1 配置 SYNC_PHS 寄存器设置同步相位。

步骤 2 配置 SYNC_SELR0[capm0_sync_sel]选择硬件同步源。

步骤 3 配置 SYNC_SELR0[capm0_synci_en]为 1 使能同步功能。当硬件同步源发出同步信号时，CAPM0 通道将 TSR 置位为 SYNC_PHS，同时产生同步输出。

----结束

12.4.4.2 软件同步 CAPM（以 CAPM0 为例）

步骤 1 配置 SYNC_PHS 寄存器设置同步相位。

步骤 2 配置 SYNC_SELR0[capm0_synci_en]为 1 使能同步功能。

步骤 3 配置寄存器 CAPM_GENE_CR[sync_sw_capm0]为 1（多 bit 同时写 1 可以实现多通道同步）。此时，CAPM0 通道将 TSR 置位为 SYNC_PHS，同时产生同步输出。

----结束

12.4.5 霍尔传感

霍尔传感器场景下，CAPM 波形示例图如图 12-4 所示。该霍尔传感器的输出线和输出真值示例如表 12-2 所示（注意实际电机可能跟示例不同）。

图12-4 CAPM 波形示例图（霍尔传感器场景）

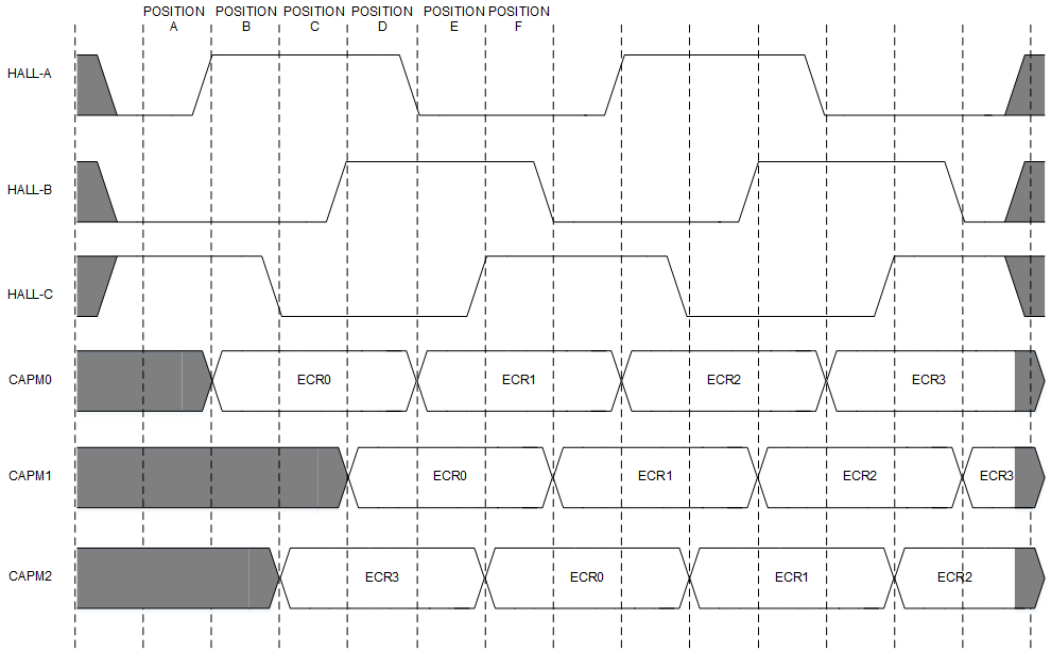


表12-2 霍尔传感器的真值表示例

波形图中的标记	霍尔真值顺序			绕组相线输出		
-	HALL-A	HALL-B	HALL-C	U	V	W
POSITION A	0	0	1	X	1	0
POSITION B	1	0	1	0	1	X
POSITION C	1	0	0	0	X	1
POSITION D	1	1	0	X	0	1
POSITION E	0	1	0	1	0	X
POSITION F	0	1	1	1	X	0

电机出轴方向：逆时针。

12.4.5.1 配置滤波

步骤 1 配置 FTCR[ft_lev]。滤波时间计算为 $FTCR[ft_lev] \times T_{pclk_capm}$ （T_{pclk_capm} 为 CAPM 模块的时钟周期）。

步骤 2 配置 FTCR[ft_en]为 1，打开滤波功能。如果不需要使用滤波，配置 FTCR[ft_en]为 0。

步骤 3 通道 1、2 配置和通道 0 相同。

----结束

12.4.5.2 平均转速和变化趋势

步骤 1 配置 INTENR，开启所有中断使能。

步骤 2 配置 TSR 计数分频 TSR_DIV[tscnt_div]。TSR 计数增加 1 对应的时间为 $\frac{TSR_DIV[tscnt_div]+1}{f_{clk_capm}}$。

步骤 3 配置 CCR0[evt$_x$pol]为 0，所有事件为上升沿事件。

步骤 4 配置 CCR0[evt$_x$rst]为 1，所有事件触发 TSR 计数器清零。

步骤 5 配置 CCR0[psc]，改变输入预分频。

步骤 6 开启 CCR0[ecrlden]，进入工作模式。

步骤 7 通道 1、2 配置和通道 0 相同。

----结束

说明

- 如果产生 INTFLGR[tsr_ovf]中断，说明计数分频的配置过小，直接改大 TSR_DIV[tscnt_div]，再对 CAPM_GENE_CR[stat_rst_capm0| stat_rst_capm1| stat_rst_capm2]写复位工作状态。
- 如果产生 INTFLGR[evtx]事件中断，直接读取相应 ECRx；或者等待事件发生，由 DMA 将 ECR0~3 搬到内存，从内存中读取。
- 假设电机旋转一圈，霍尔单相信号周期个数为 n。则预分频 psc 对应的电机旋转圈数 $N=\frac{预分频系数}{n}$。则在 N 圈内，$平均速度=\frac{N}{ECR\times 5ns\times 时间步长}RPM$。注意实际电机型号的不同，旋转一圈的霍尔传感器输出波形的周期个数也不同。
- 比较 ECR0~3 的值，可知短时间内转速的变化趋势。

须知

当计数时间步长小（计数分频小）而输入分频大的时候，计算比较准确。

低转速可能会使 TSR 计数器溢出，仅供参考。

12.4.5.3 转子位置区域

步骤 1 配置 INTENR 寄存器的所有 bit 位域为 1，使能所有中断。

步骤 2 配置事件 0 和 2 为上升沿事件 CCR0[evt0pol]和 CCR0[evt2pol]为 1，而事件 1 和 3 为下降沿事件 CCR0[evt1pol]和 CCR0[evt3pol]为 0。

步骤 3 配置 SYNC_SELR0[capm0_synci_en]为 1，使能同步功能。

步骤 4 配置 CCR0[ecrlden]为 1，进入工作模式。

步骤 5 通道 1、2 配置和通道 0 相同

步骤 6 配置 CAPM_GENE_CR[sync_sw_capm0|sync_sw_capm1|sync_sw_capm2]为 1，触发通道 0、1、2 软件同步。

----结束

说明

- 软件中断方式：当产生 CAPMx[INTFLGR.evtx]中断时，读取 CAPM0，1，2 的 ECSEQR[nxtldecr]，得到下个被加载的 ECR 序号，与图 12-4 对应的位置判断如表 12-3 所示。上述步骤 1 配置了事件 0 和 2 为上升沿，事件 1 和 3 为下降沿，即 nxtldecr=00 或 10 表明下一个被加载的事件是 1 或 3，表明当前是低电平，对应第一列的真值表的 0。
- 软件轮询方式：任意时刻，读取 CAPMx_CCR 寄存器组中的 ECSEQR[crt_edge]，得到当前输入信号电平，与图 12-4 对应的位置判断如表 12-4 所示。

在上述配置下，CAPMx_CCR 寄存器组中的 ECSEQR[nxtldecr]可以直接对应转子所在区域。

任意状态，包括电机停止状态下，读取 CAPMx_CCR 寄存器组中的 ECSEQR[crt_edge]可以直接对应到转子所在区域。

表12-3 软件中断方式下，CAPM 的捕获序列示例（霍尔传感器场景）

波形图中的标记	HALL-A/B/C	CAPM0_CCR 寄存器组中的 ECSEQR[nxtldecr]	CAPM1_CCR 寄存器组中的 ECSEQR[nxtldecr]	CAPM2_CCR 寄存器组中的 ECSEQR[nxtldecr]
POSITION A	001	00 或 10	00 或 10	11 或 01
POSITION B	101	01 或 11	00 或 10	11 或 01
POSITION C	100	01 或 11	00 或 10	00 或 10
POSITION D	110	01 或 11	01 或 11	00 或 10
POSITION E	010	10 或 00	01 或 11	00 或 10

波形图中的标记	HALL-A/B/C	CAPM0_CCR 寄存器组中的 ECSEQR[nxtldecr]	CAPM1_CCR 寄存器组中的 ECSEQR[nxtldecr]	CAPM2_CCR 寄存器组中的 ECSEQR[nxtldecr]
POSITION F	011	10 或 00	01 或 11	01 或 11

表12-4 软件轮询方式下，CAPM 的捕获序列示例（霍尔传感器场景）

波形图中的标记	HALL-A/B/C	CAPM0_CCR 寄存器组中的 ECSEQR[crt_edge]	CAPM1_CCR 寄存器组中的 ECSEQR[crt_edge]	CAPM2_CCR 寄存器组中的 ECSEQR[crt_edge]
POSITION A	001	00	00	11
POSITION B	101	11	00	11
POSITION C	100	11	00	00
POSITION D	110	11	11	00
POSITION E	010	00	11	00
POSITION F	011	00	11	11

*（二进制 11 = 十进制 3）

12.4.6 PWM 通信场景

PWM 通信（单向）依靠事先约定好的规则，仅靠一根连线，由主 MCU 向从 MCU 传递信息。例如：

- 调频：约定由 PWM 频率传递信息，将频率直接按公式转换为某种数值信息。
- 调占空比：约定由 PWM 波的占空比传递信息，PWM 波频率不变。例如 50%占空比为 0，90%占空比为正向最大，10%占空比为负向最大。

12.4.6.1 通过调频传递信息

步骤 1 配置 TSR 计数器分频 TSR_DIV[tscnt_div]。

步骤 2 配置捕获模式 CCR2[cap_mode]为单轮次捕获模式。

步骤 3 配置捕获序列终点 CCR2[seq_stop]为 0x0，事件 0。

步骤 4 配置输入信号预分频系数 CCR0[psc]。

步骤 5 配置产生 DMA 请求的边沿事件 CCR0[dmaevt_sel]。

步骤 6 配置每个边沿采集极性 CCR0[evt0pol]为上升沿。

步骤 7 配置每个边沿是否触发计数器复位 CCR0[evt0rst]。

步骤 8 配置 INTENR，使能 evt0，ecr_ovf，tsr_ovf 中断。

步骤 9 配置 CAPM_GENE_CR[tsr_stop_capm0]，使能时间计数；配置 CCR0[ecrlden]，进入工作模式。

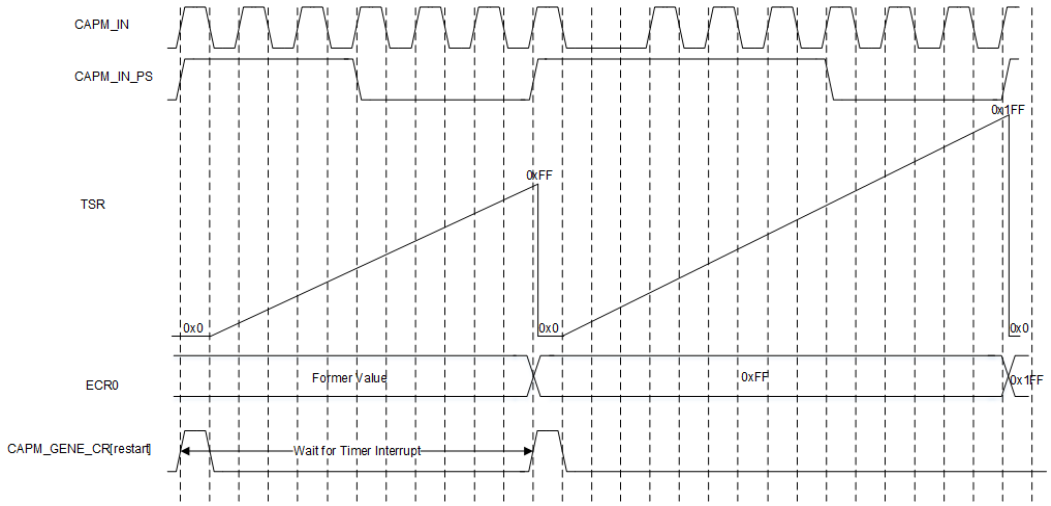
步骤 10 在定时任务中，配置 CAPM_COMM 寄存器组中的 CAPM_GENE_CR[restart]为 1 开启测量，当 evt1 中断发生时，再开启新的定时任务，如此往复。

步骤 11 读取 ECR0 的值。

步骤 12 平均频率 = 输入信号预分频系数/(ECR0 读数×TSR 分频×时钟 clk_capmx 周期）。

----结束

图12-5 CAPM 波形示例图（PWM 调频通信）



12.4.6.2 通过调占空比传递信息（约定 PWM 波频率/周期不变）

步骤 1 配置计数器分频 TSR_DIV[tscnt_div]，设置计数器 TSR 的步长。

步骤 2 配置 CCR2[cap_mode]为 1，设置捕获模式为单轮次捕获模式。

步骤 3 配置 CCR2[seq_stop]为 0x1，设置捕获序列终点为事件 1。

步骤 4 配置 CCR0[psc]为 0x0，设置输入信号预分频系数为不分频。

步骤 5 配置 CCR0[dmaevt_sel]为 0x1，设置事件 1 产生 DMA 请求。若使用中断，则此步骤忽略。

步骤 6 配置 CCR0[evt0pol]为 1，CCR0[evt1pol]为 0，设置事件 0 的捕获边沿极性为上升沿，事件 1 的捕获边沿极性为下降沿。

步骤 7 配置 CCR0[evt0rst]为 1，CCR0[evt1rst]为 0，设置事件 0 触发计数器复位，事件 1 不触发计数器复位。

步骤 8 配置 INTENR[evt1_en]，ecr_ovf_en，tsr_ovf_en 为 1，使能 evt1_int，ecr_ovf_int，tsr_ovf_int 中断。

步骤 9 配置 CAPM_COMM 寄存器组中的 CAPM_GENE_CR[tsr_stop_capm0] 为 0，使能时间计数。配置 CCR0[ecrlden]为 1，进入工作模式。

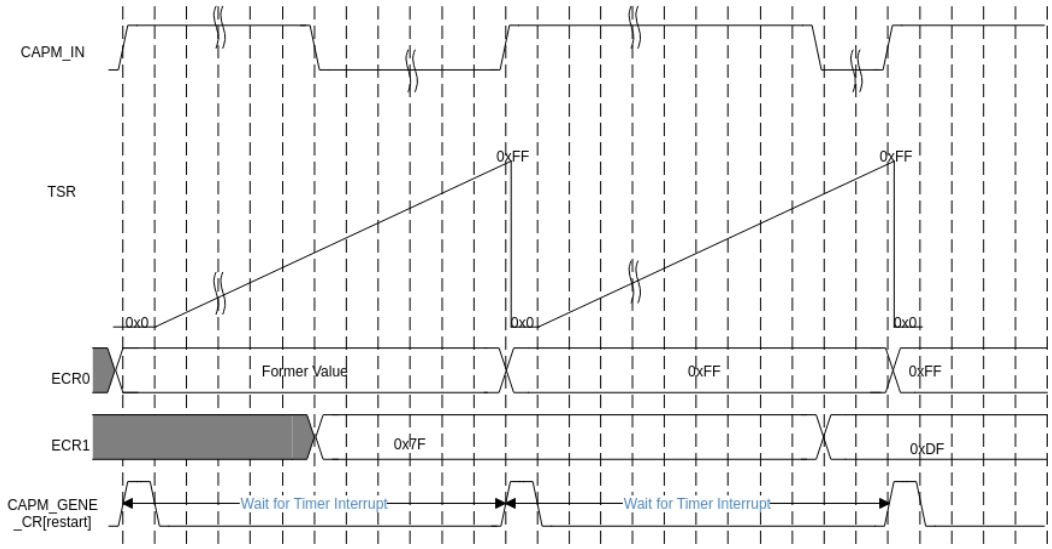
步骤 10 在定时任务中，对 CAPM_COMM 寄存器组中的 CAPM_GENE_CR[restart_capm0]写 1 开启测量，当 evt1 中断发生时，再开启新的定时任务，如此往复。

步骤 11 读取 ECR0，ECR1 的值。因为事件 0 触发了计数器复位，而事件 1 未触发复位，故高电平持续时间即为 ECR1，低电平时间即为 ECR0-ECR1。

步骤 12 占空比 = ECR1 × TSR 分频 × 时钟 clk_capmx 周期/约定的 PWM 波周期。

----结束

图12-6 CAPM 波形示例图（PWM 调占空比通信）



12.4.7 边沿个数的统计和比较

步骤 1 配置 EAR_CMP[earcmp]可以设置边沿数量比较值。

步骤 2 配置 INTENR[ear_ovf_en]，可以使能 ear_ovf_int 边沿数量计数溢出中断。

步骤 3 配置 INTENR[earcmp_match_en]为 1，可以使能 earcmp_match_int 边沿计数比较匹配中断。

步骤 4 配置 CCR0[cnt_edge_sel]，可以设置计数的边沿类型（上升，下降，双沿）。

步骤 5 读 EAR，可以获得边沿个数。

步骤 6 边沿计数等于比较值 EAR_CMP[earcmp]时，产生 earcmp_match_int 中断。若没有使能 earcmp_match_int 中断，则无中断上报。

步骤 7 边沿计数溢出时，产生 ear_ovf_int 中断。若没有使能 ear_ovf_int 中断，则无中断上报。

步骤 8 配置 CAPM_GENE_CR[stat_rst_capm0]为 1，可以复位工作状态。

步骤 9 重新配置 CCR0[cnt_edge_sel]为 0x0（不计数任何类型边沿），关闭计数功能，注意此时 EAR 会暂停而不会清零。

----结束

12.5 CAPMx_CCR 寄存器概览（x=0,1,2）

CAPMx_CCR 寄存器概览如表 12-5 所示。

表12-5 CAPMx_CCR 寄存器概览（CAPM0~2 捕获通道的基地址分别是 0x14B0_0000，0x14B0_1000，0x14B0_2000）

偏移地址	名称	描述	页码
0x0000	REV_INFO	Revision Information，版本信息寄存器	REV_INFO
0x0004	TSR	Time-Stamp Register，时间戳计数器	TSR
0x0008	TSR_DIV	Time-Stamp Divider Register，时间戳计数器分频配置寄存器	TSR_DIV
0x000C	EAR	Edge Amount Register，边沿数量寄存器	EAR
0x0010	EAR_CMP	EAR Compare Value Reister，边沿数量比较寄存器	EAR_CMP
0x0014	SYNC_PHS	Sync Phase，同步相位寄存器	SYNC_PHS
0x0018	ECR0	Event 0 Capture Register，事件 0 时间戳寄存器	ECR0
0x001C	ECR1	Event 1 Capture Register，事件 1 时间戳寄存器	ECR1
0x0020	ECR2	Event 2 Capture Register，事件 2 时间戳寄存器	ECR2
0x0024	ECR3	Event 3 Capture Register，事件 3 时间戳寄存器	ECR3
0x0028	ECSEQR	Event Capture Sequence Register，事件捕获序列寄存器	ECSEQR
0x002C	FTCR	Filter Control Register，滤波控制寄存器	FTCR
0x0030	CCR0	Control & Configure Register 0，输入信号处理，事件产生和行为模式寄存器	CCR0
0x0034	CCR1	Control & Configure Register 1，仿真器模式寄存器	CCR1
0x0038	CCR2	Control & Configure Register 2，工作模式寄存器	CCR2
0x0040	INTENR	Interrupt Enable Register，中断使能寄存器	INTENR
0x0044	INTRAWR	Raw Interrupt Register，原始中断寄存器	INTRAWR

偏移地址	名称	描述	页码
0x0048	INTINJR	Interrupt Injection Register，中断注入寄存器	INTINJR
0x004C	INTFLGR	Interrupt Flag Register，中断状态寄存器	INTFLGR

12.6 CAPMx_CCR 寄存器描述（x=0,1,2）

REV_INFO

REV_INFO 为 Revision Information，版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0121_0915

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27:24]	RO	revision	IP 版本号。	0x1
[23:0]	-	reserved	保留。	0x210915

TSR

TSR 为 Time-Stamp Register，时间戳计数器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	tsr	时间戳计数器，可用于监视计数器状态。	0x00000000

TSR_DIV

TSR_DIV 为 Time-Stamp Divider Register，时间戳计数器分频配置寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[15:0]	RW	tscnt_div	计数器分频。 0x0000：每 1clk 计数器+1； 0x0001：每 2clk 计数器+1； 0x0002：每 3clk 计数器+1； ... 0xFFFF：每 65536clk 计数器+1。	0x0000

EAR

EAR 为 Edge Amount Register，边沿数量寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RO	ear	边沿数量计数值，写清零。	0x0000

EAR_CMP

EAR_CMP 为 EAR Compare Value Register，边沿数量比较寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	earcmp	边沿数量计数比较值。	0xFFFF

SYNC_PHS

SYNC_PHS 为 Sync Phase，同步相位寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	sync_phs	计数器同步相位值寄存器，硬件同步源或	0x0000000

Bits	Access	Name	Description	Reset
			软件同步（sync_sw）有效时，将 TSR 寄存器置位为该值。	0

ECR0

ECR0 为 Event 0 Capture Register，事件 0 时间戳寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	ecr0	捕获寄存器 0（事件 0）。	0x00000000

ECR1

ECR1 为 Event 1 Capture Register，事件 1 时间戳寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	ecr1	捕获寄存器 1（事件 1）。	0x00000000

ECR2

ECR2 为 Event 2 Capture Register，事件 2 时间戳寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	ecr2	捕获寄存器 2（事件 2）。	0x00000000

ECR3

ECR3 为 Event 3 Capture Register，事件 3 时间戳寄存器。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:0]	RO	ecr3	捕获寄存器 3（事件 3）。	0x00000000

ECSEQR

ECSEQR 为 Event Capture Sequence Register，事件捕获序列寄存器。

Offset Address: 0x0028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:2]	RO	crt_edge	当前输入信号电平（反映原输入信号的电平，不受预分频影响）。 00：当前为低电平。 01：上升沿； 10：下降沿； 11：高电平。	0x0
[1:0]	RO	nxtldecr	读取下个被加载的 ECR。 00：ECR0； 01：ECR1； 10：ECR2； 11：ECR3。	0x0

FTCR

FTCR 为 Filter Control Register，滤波控制寄存器。

Offset Address: 0x002C Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:14]	-	reserved	保留。	0x00000
[13:1]	RW	ft_lev	滤波等级配置。 0x0000：不过滤毛刺； 0x0001：过滤最宽 1 时钟周期宽度毛刺；	0x0000

Bits	Access	Name	Description	Reset
			... 0x1FFF：过滤最宽 8191 时钟周期宽度毛刺。	
[0]	RW	ft_en	滤波功能使能开关。 0：关闭； 1：开启。	0x1

CCR0

CCR0 为 Control & Configure Register 0，输入信号处理，事件产生和行为模式寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0600

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20:19]	RW	cnt_edge_sel	边沿数量计数的边沿类型选择。 00：不记录边沿数量； 01：对上升沿数量计数； 10：对下降沿数量计数； 11：对双沿数量计数。	0x0
[18:11]	RW	psc	输入信号预分频系数。 0x00：不分频； 0x01：输入信号 2 倍分频； 0x02：输入信号 4 倍分频； ... 0xFF：输入信号 510 倍分频。	0x00
[10:9]	RW	dmaevt_sel	在捕获序列中，决定由哪个事件产生 DMA 请求。 00：事件 0（EVT0）； 01：事件 1（EVT1）；	0x3

Bits	Access	Name	Description	Reset
			10：事件 2（EVT2）； 11：事件 3（EVT3）。	
[8]	RW	ecrlden	捕获使能。 0：关闭捕获； 1：开启捕获。	0x0
[7]	RW	evt3rst	事件 3（EVT3）是否复位 TSR。 0：EVT3 不复位 TSR； 1：EVT3 复位 TSR。	0x0
[6]	RW	evt3pol	事件 3（EVT3）捕获边沿选择。 0：EVT3 捕获下降沿； 1：EVT3 捕获上升沿。	0x0
[5]	RW	evt2rst	事件 2（EVT2）是否复位 TSR。 0：EVT2 不复位 TSR； 1：EVT2 复位 TSR。	0x0
[4]	RW	evt2pol	事件 2（EVT2）捕获边沿选择。 0：EVT2 捕获下降沿； 1：EVT2 捕获上升沿。	0x0
[3]	RW	evt1rst	事件 1（EVT1）是否复位 TSR。 0：EVT1 不复位 TSR； 1：EVT1 复位 TSR。	0x0
[2]	RW	evt1pol	事件 1（EVT1）捕获边沿选择。 0：EVT1 捕获下降沿； 1：EVT1 捕获上升沿。	0x0
[1]	RW	evt0rst	事件 0（EVT0）是否复位 TSR。 0：EVT0 不复位 TSR； 1：EVT0 复位 TSR。	0x0
[0]	RW	evt0pol	事件 0（EVT0）捕获边沿选择。	0x0

Bits	Access	Name	Description	Reset
			0：EVT0 捕获下降沿； 1：EVT0 捕获上升沿。	

CCR1

CCR1 为 Control & Configure Register 1，仿真器模式寄存器。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW	emu_stop_en	仿真器停止 TSR 使能。 0：禁止仿真器停止计数器； 1：允许仿真器停止计数器。	0x0
[2:0]	-	reserved	保留。	0x0

CCR2

CCR2 为 Control & Configure Register 2，工作模式寄存器。

Offset Address: 0x0038 Total Reset Value: 0x0000_0006

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2:1]	RW	seq_stop	捕获序列终点/循环边界。 00：事件 0（EVT0）为捕获终点/循环边界； 01：事件 1（EVT1）为捕获终点/循环边界； 10：事件 2（EVT2）为捕获终点/循环边界； 11：事件 3（EVT3）为捕获终点/循环边界。	0x3

Bits	Access	Name	Description	Reset
[0]	RW	cap_mode	捕获模式选择。 0：循环捕获模式；（到循环边界后循环）； 1：单轮次捕获模式。（到捕获终点后结束）。	0x0

INTENR

INTENR 为 Interrupt Enable Register，中断使能寄存器。

Offset Address: 0x0040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RW	dmareq_ovf_en	DMA 请求溢出中断使能。不使能时，即屏蔽相应的中断状态，不影响相应的原始中断。若全部不使能，模块不会上报中断给 CPU。 0：不使能； 1：使能。	0x0
[7]	RW	ear_ovf_en	边沿数量计数溢出中断使能。 0：不使能； 1：使能。	0x0
[6]	RW	earcmp_match_en	边沿数量计数比较匹配中断使能。 0：不使能； 1：使能。	0x0
[5]	RW	ecr_ovf_en	捕获溢出中断使能。 0：不使能； 1：使能。	0x0
[4]	RW	tsr_ovf_en	TSR 溢出中断使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[3]	RW	evt3_en	事件 3 中断使能。 0：不使能； 1：使能。	0x0
[2]	RW	ev2_en	事件 2 中断使能。 0：不使能； 1：使能。	0x0
[1]	RW	evt1_en	事件 1 中断使能。 0：不使能； 1：使能。	0x0
[0]	RW	evt0_en	事件 0 中断使能。 0：不使能； 1：使能。	0x0

INTRAWR

INTRAWR 为 Raw Interrupt Register，原始中断寄存器。

Offset Address: 0x0044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RW1C	dmareq_ovf_raw	DMA 请求溢出原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[7]	RW1C	ear_ovf_raw	边沿数量计数溢出原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[6]	RW1C	earcmp_match_raw	边沿数量比较匹配原始中断，写 1 清除。 0：无原始中断；	0x0

Bits	Access	Name	Description	Reset
			1：有原始中断。	
[5]	RW1C	ecr_ovf_raw	捕获溢出原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[4]	RW1C	tsr_ovf_raw	TSR 溢出原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[3]	RW1C	evt3_raw	事件 3 原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[2]	RW1C	evt2_raw	事件 2 原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[1]	RW1C	evt1_raw	事件 1 原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0
[0]	RW1C	evt0_raw	事件 0 原始中断，写 1 清除。 0：无原始中断； 1：有原始中断。	0x0

INTINJR

INTINJR 为 Interrupt Injection Register，中断注入寄存器。

Offset Address: 0x0048 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	W1_PULSE	dmareq_ovf_inj	注入 DMA 请求溢出原始中断。写 1 自动清零。	0x0

Bits	Access	Name	Description	Reset
			0：不注入； 1：注入。	
[7]	W1_PULSE	ear_ovf_inj	注入边沿数量计数溢出原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[6]	W1_PULSE	earcmp_match_inj	注入边沿数量比较匹配原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[5]	W1_PULSE	ecr_ovf_inj	注入捕获溢出原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[4]	W1_PULSE	tsr_ovf_inj	注入 TSR 溢出原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[3]	W1_PULSE	evt3_inj	注入事件 3 原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[2]	W1_PULSE	evt2_inj	注入事件 2 原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[1]	W1_PULSE	evt1_inj	注入事件 1 原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0
[0]	W1_PULSE	evt0_inj	注入事件 0 原始中断。写 1 自动清零。 0：不注入； 1：注入。	0x0

INTFLGR

INTFLGR 为 Interrupt Flag Register，中断标志/状态寄存器。

Offset Address: 0x004C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RO	dmareq_ovf_int	DMA 请求溢出中断标志。 0：无中断； 1：有中断。	0x0
[7]	RO	ear_ovf_int	边沿数量计数溢出中断标志。 0：无中断； 1：有中断。	0x0
[6]	RO	earcmp_match_int	边沿数量计数比较匹配中断标志。 0：无中断； 1：有中断。	0x0
[5]	RO	ecr_ovf_int	捕获溢出中断标志。 0：无中断； 1：有中断。	0x0
[4]	RO	tsr_ovf_int	TSR 溢出中断标志。 0：无中断； 1：有中断。	0x0
[3]	RO	evt3_int	事件 3 中断标志。 0：无中断； 1：有中断。	0x0
[2]	RO	evt2_int	事件 2 中断标志。 0：无中断； 1：有中断。	0x0
[1]	RO	evt1_int	事件 1 中断标志。	0x0

Bits	Access	Name	Description	Reset
			0：无中断； 1：有中断。	
[0]	RO	evt0_int	事件 0 中断标志。 0：无中断； 1：有中断。	0x0

12.7 CAPM_COMM 寄存器概览

CAPM_COMM 寄存器概览如表 12-6 所示。

表12-6 CAPM_COMM 寄存器概览（基址是 0x14B0_3000）

偏移地址	名称	描述	页码
0x0000	REV_INFO	Revision Information，版本信息寄存器	REV_INFO
0x0004	CAPM_GENE_CR	CAPM general control Reg，通用配置寄存器	CAPM_GENE_CR
0x0008	SYNC_SELR0	Sync Selection Register(for CAPM0)，CAPM0 同步设置寄存器	SYNC_SELR0
0x000C	SYNC_SELR1	Sync Selection Register(for CAPM1)，CAPM1 同步设置寄存器	SYNC_SELR1
0x0010	SYNC_SELR2	Sync Selection Register(for CAPM2)，CAPM2 同步设置寄存器	SYNC_SELR2
0x0028	INPUT_SELR0	Input Source Selection Register(for CAPM0)，CAPM0 输入源选择寄存器	INPUT_SELR0
0x002C	INPUT_SELR1	Input Source Selection Register(for CAPM1)，CAPM1 输入源选择寄存器	INPUT_SELR1
0x0030	INPUT_SELR2	Input Source Selection Register(for CAPM2)，CAPM2 输	INPUT_SELR

偏移地址	名称	描述	页码
		入源选择寄存器	2

12.8 CAPM_COMM 寄存器描述

REV_INFO

REV_INFO 为 Revision Information，版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0121_0528

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27:24]	RO	revision	IP 版本号。	0x1
[23:0]	-	reserved	保留。	0x210528

CAPM_GENE_CR

CAPM_GENE_CR 为 Capture Module General Control Register，通用配置寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0700

Bits	Access	Name	Description	Reset
[31:27]	-	reserved	保留。	0x00
[26]	W1_PULSE	sync_sw_capm2	触发 CAPM2 同步，TSR 的值被同步为 SYNC_PHS 的值，同时重启捕获序列。软件写 1 后自动清零。 0：不同步； 1：同步。写 1 后自动清零。	0x0
[25]	W1_PULSE	sync_sw_capm1	触发 CAPM1 同步，TSR 的值被同步为 SYNC_PHS 的值，同时重启捕获序列。软件写 1 后自动清零。	0x0

Bits	Access	Name	Description	Reset
			0：不同步； 1：同步。写 1 后自动清零。	
[24]	W1_PULSE	sync_sw_capm0	触发 CAPM0 同步，TSR 的值被同步为 SYNC_PHS 的值，同时重启捕获序列。软件写 1 后自动清零。 0：不同步； 1：同步。写 1 后自动清零。	0x0
[23:19]	-	reserved	保留。	0x00
[18]	W1_PULSE	stat_rst_capm2	CAPM2 工作状态复位，复位范围包括：TSR 计数器；EAR 计数器；捕获序列；TSR 分频计数器；原始中断。 0：不复位； 1：复位。写 1 后自动清零。	0x0
[17]	W1_PULSE	stat_rst_capm1	CAPM1 工作状态复位，复位范围包括：TSR 计数器、EAR 计数器、捕获序列、TSR 分频计数器、原始中断。 0：不复位； 1：复位。写 1 后自动清零。	0x0
[16]	W1_PULSE	stat_rst_capm0	CAPM0 工作状态复位，复位范围包括：TSR 计数器、EAR 计数器、捕获序列、TSR 分频计数器、原始中断。 0：不复位； 1：复位。写 1 后自动清零。	0x0
[15:11]	-	reserved	保留。	0x00
[10]	RW	tsr_stop_capm2	CAPM2 TSR 暂停计数使能。 0：CAPM2 TSR 继续计数； 1：CAPM2 TSR 暂停计数。	0x1
[9]	RW	tsr_stop_capm1	CAPM1 TSR 暂停计数使能。 0：CAPM1 TSR 继续计数；	0x1

Bits	Access	Name	Description	Reset
			1：CAPM1 TSR 暂停计数。	
[8]	RW	tsr_stop_capm0	CAPM0 TSR 暂停计数使能。 0：CAPM0 TSR 继续计数； 1：CAPM0 TSR 暂停计数。	0x1
[7:3]	-	reserved	保留。	0x00
[2]	W1_PULSE	restart_capm2	CAPM2 开始一次新的单轮次捕获。 0：不开始； 1：开始。写 1 后自动清零。	0x0
[1]	W1_PULSE	restart_capm1	CAPM1 开始一次新的单轮次捕获。 0：不开始； 1：开始。写 1 后自动清零。	0x0
[0]	W1_PULSE	restart_capm0	CAPM0 开始一次新的单轮次捕获。 0：不开始； 1：开始。写 1 后自动清零。	0x0

SYNC_SELR0

SYNC_SELR0 为 Sync Selection Register(for CAPM0)，CAPM0 同步设置选择寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RW	capm0_synci_en	CAPM0 硬件，软件同步使能。 0：不使能； 1：使能。	0x0
[3:0]	RW	capm0_sync_sel	CAPM0 的硬件同步源选择。 0x0：无同步源； 0x1：APT0 同步输出信号 APT0SYNCO；	0x0

Bits	Access	Name	Description	Reset
			0x2：APT1 同步输出信号 APT1SYNCO； 0x3：APT2 同步输出信号 APT2SYNCO； 0x4：APT3 同步输出信号 APT3SYNCO； 其他：无同步源。	

SYNC_SELR1

SYNC_SELR1 为 Sync Selection Register(for CAPM1)，CAPM1 同步设置选择寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RW	capm1_synci_en	CAPM1 硬件，软件同步使能。 0：不使能； 1：使能。	0x0
[3:0]	RW	capm1_sync_sel	CAPM1 的硬件同步源选择： 同 capm0_sync_sel。	0x0

SYNC_SELR2

SYNC_SELR2 为 Sync Selection Register(for CAPM2)，CAPM2 同步设置选择寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RW	capm2_synci_en	CAPM2 硬件，软件同步使能。 0：不使能； 1：使能。	0x0
[3:0]	RW	capm2_sync_sel	CAPM2 的硬件同步源选择：	0x0

Bits	Access	Name	Description	Reset
			同 capm0_sync_sel。	

INPUT_SELR0

INPUT_SELR0 为 Input Source Selection Register(for CAPM0)，CAPM0 输入源选择寄存器。

Offset Address: 0x0028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	capm0_in_sel	CAPM0 输入源选择。 0：CAPM0_IN； 1：无。	0x0

INPUT_SELR1

INPUT_SELR1 为 Input Source Selection Register(for CAPM1)，CAPM1 输入源选择寄存器。

Offset Address: 0x002C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	capm1_in_sel	CAPM1 输入源选择。 0：CAPM1_IN； 1：无。	0x0

INPUT_SELR2

INPUT_SELR2 为 Input Source Selection Register(for CAPM2)，CAPM2 输入源选择寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	capm2_in_sel	CAPM2 输入源选择。 0：CAPM2_IN； 1：无。	0x0

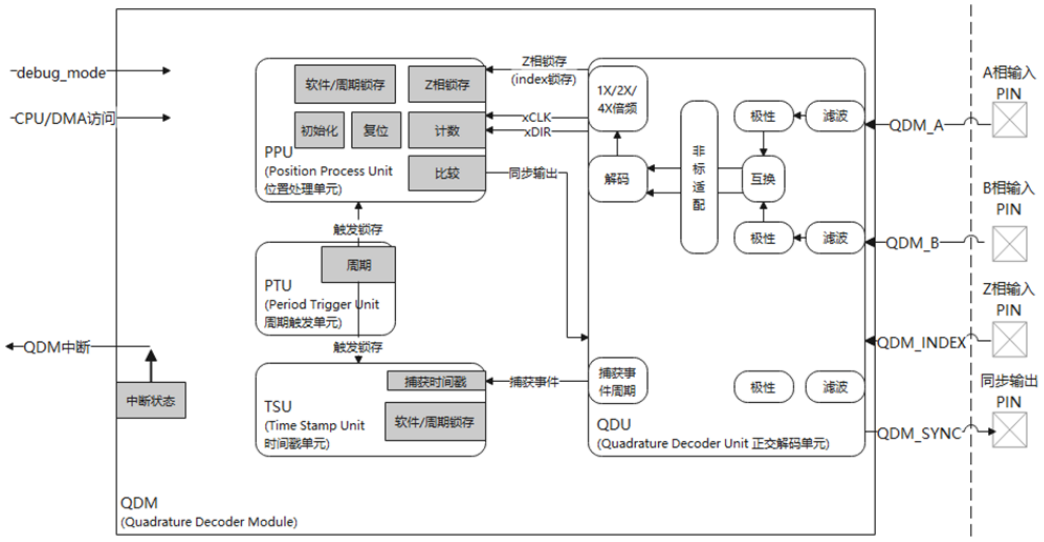
13 正交解码模块(QDM)

13.1 概述

QDM（Quadrature Decoder Module）是正交解码模块用于解码增量编码器，针对增量编码器输出的 A/B/Z 三相信号进行解码，记录位置信息，方向信息和时间信息。QDM 内部集成了正交解码单元 QDU，位置处理单元 PPU，时间戳单元 TSU，周期触发单元 PTU，实现了多种类型的增量编码器解码，支持 1X/2X/4X 倍频解码，支持位置计数的多种校准模式、初始化模式、锁存模式，可编程的位置区间内计时，可编程的时间区间内计数位置，A/B 相信号看门狗等功能。用于绝对位置记录，相对位置记录，旋转速度计算等应用场景。

13.2 特性

图13-1 QDM 特性模块简要框图

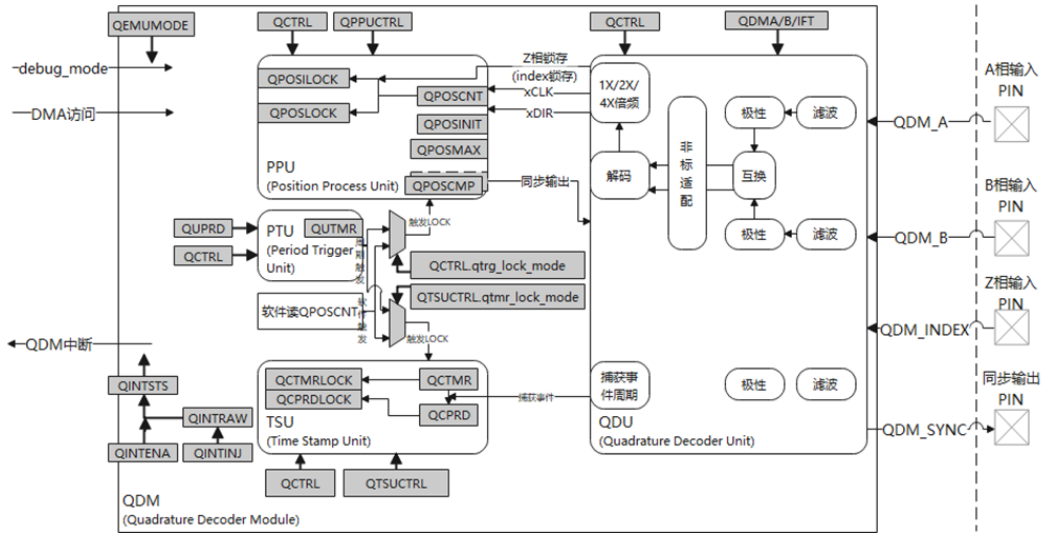


- 支持解码 4 种类型的增量编码器：正交型编码器、脉冲方向型编码器、非标准 TYPE1 型编码器、非标准 TYPE2 型编码器。
- 支持 A/B/Z 相信号滤波（支持过滤毛刺宽度 1~8192 个时钟周期），支持 A/B/Z 相信号极性选择，支持 A/B 相信号互换。
- 支持倍频解码：1X、2X、4X 倍频解码。
- 支持 PPU 位置计数，位宽 32bit，支持独立使能，支持 3 种计数模式。
- 支持 PPU 位置计数复位，4 种复位模式。
- 支持 PPU 位置计数初始化，3 种初始化模式。
- 支持 PPU 位置计数锁存，3 种 Z 相锁存模式，2 种锁存模式（支持软件触发锁存，PTU 触发锁存）。
- 支持 PPU 位置计数比较功能，支持位置比较缓存模式，支持比较输出同步信号。
- 支持 TSU 时间戳记录，位宽 32bit，可配置的位置区间内，TSU 产生时间戳（记录时间），用于速度计算。
- 支持 PTU 周期触发，位宽 32bit，可配置的时间区间内，触发 PPU 产生位置记录（记录位置）。
- 支持 PTU 看门狗模式，位宽 32bit，检测 A/B 信号输入。
- 支持 DMA 访问。
- 支持仿真器接入，3 种接入模式。

13.3 功能描述

QDM 各项功能关系与相应寄存器如图 13-2 所示。

图13-2 QDM 功能模块详细框图



下面详细介绍各种功能，包括：13.3.1 滤波，极性选择和互换；13.3.2 解码模式；13.3.3 倍频解码；13.3.4 PPU 位置计数/复位/初始化/锁存模式；13.3.5 PPU 位置比较和同步输出；13.3.6 TSU 计时和时间戳；13.3.7 PTU 周期模式/计时/周期触发；13.3.8 PTU 看门狗模式。

13.3.1 滤波，极性选择和互换

- 支持输入的 A/B/Z 相滤波，可配置 QDMAFT，QDMBFT，QDMIFT 调整各相的滤波等级。
- 支持输入的 A/B/Z 相极性选择，可配置 QCTRL[qdma_polarity]，QCTRL[qdmb_polarity]，QCTRL[qdmi_polarity]改变极性选择。
- 支持 A/B 相互换，可配置 QCTRL[qdm_ab_swap]使能。

须知

滤波会导致信号延时，延迟的时间与滤波宽度的配置相同。

13.3.2 解码模式

支持解码 4 种类型的增量编码器：正交型编码器、脉冲方向型编码器、非标准 TYPE1 型编码器、非标准 TYPE2 型编码器。QDU 解码模式可通过配置 QCTRL[qdu_mode]选择。

不同制造商生产的编码器，A 相/B 相含义，相位差和对应的旋转方向，Z 相（index）长度均有区别。不同类型编码器的 A 相、B 相、Z 相（又称为 Index）波形图示例如图 13-3、图 13-4、图 13-6 和图 13-7 所示。

图13-3 正交型编码器的 A 相、B 相、Z 相波形图示例（顺时针旋转）

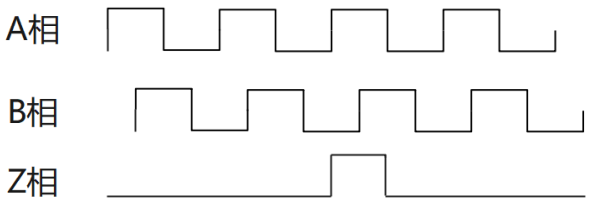


图13-4 正交型编码器的 A 相、B 相、Z 相波形图示例（逆时针旋转）

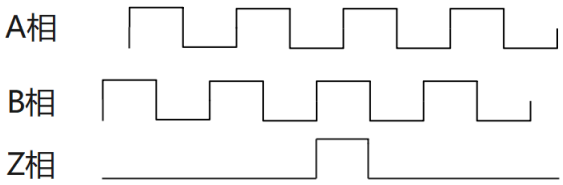


图13-5 脉冲方向型编码器 A 相、B 相、Z 相波形图示例（先顺时针旋转，后逆时针旋转）

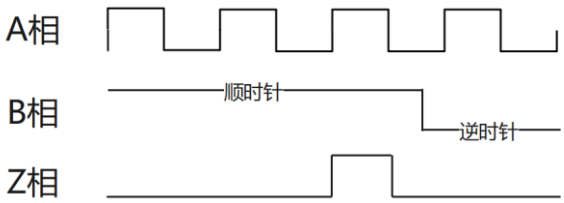


图13-6 非标准 TYPE1 型编码器波形图示例（先顺时针旋转，后逆时针旋转）

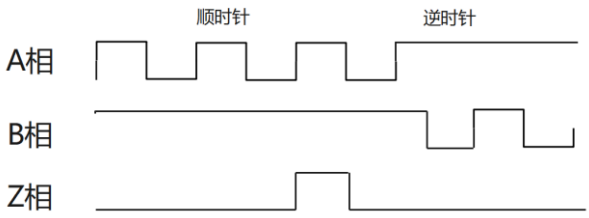
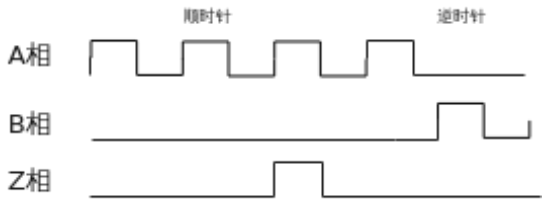


图13-7 非标准 TYPE2 型编码器波形图示例（先顺时针旋转，后逆时针旋转）



13.3.3 倍频解码

支持 1X/2X/4X 倍频解码，配置 QCTRL[qdu_xclk]选择不同倍频。

- 1X 倍频解码：A 相上升沿产生位置脉冲 xCLK，位置计数单元 PPU 根据方向 xDIR，位置脉冲 xCLK 和计数模式 QPPUCTRL[pcnt_mode]，记录位置信息。
- 2X 倍频解码：A 相双沿产生位置脉冲 xCLK，位置计数单元 PPU 根据方向 xDIR，位置脉冲 xCLK 和计数模式 QPPUCTRL[pcnt_mode]，记录位置信息。
- 4X 倍频解码：A 相和 B 相双沿产生位置脉冲 xCLK，位置计数单元 PPU 根据方向 xDIR，位置脉冲 xCLK 和计数模式 QPPUCTRL[pcnt_mode]，记录位置信息。

图13-8 正交输入型 1X/2X/4X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）

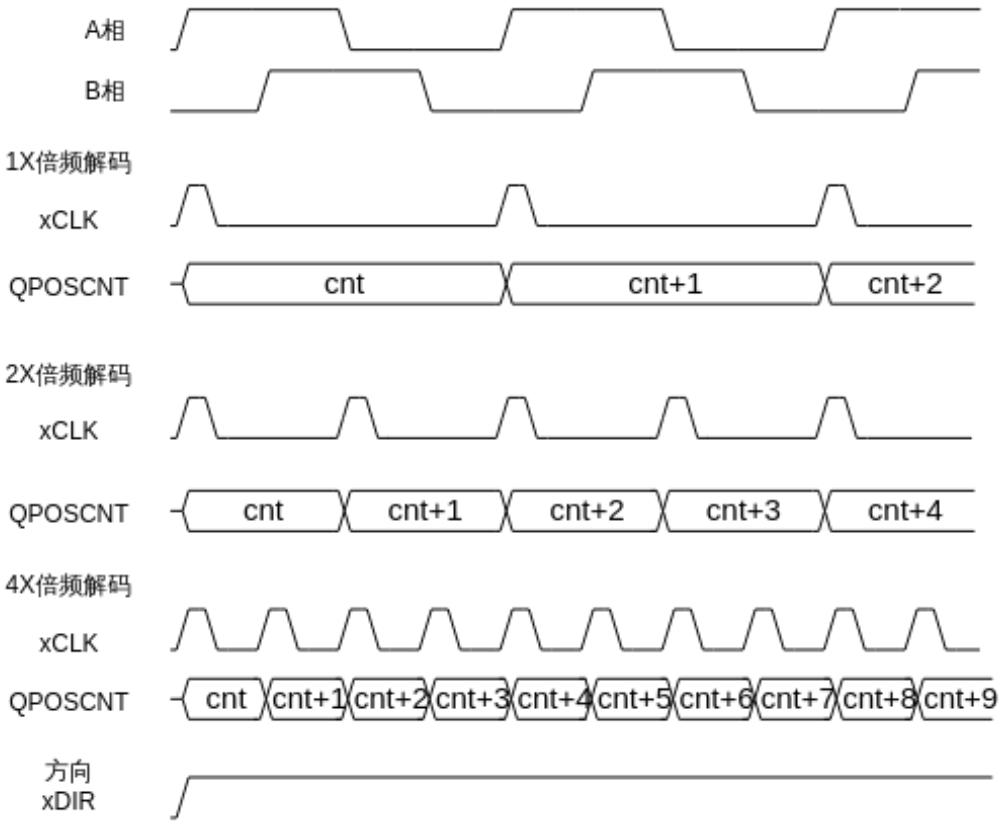


图13-9 脉冲方向型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）

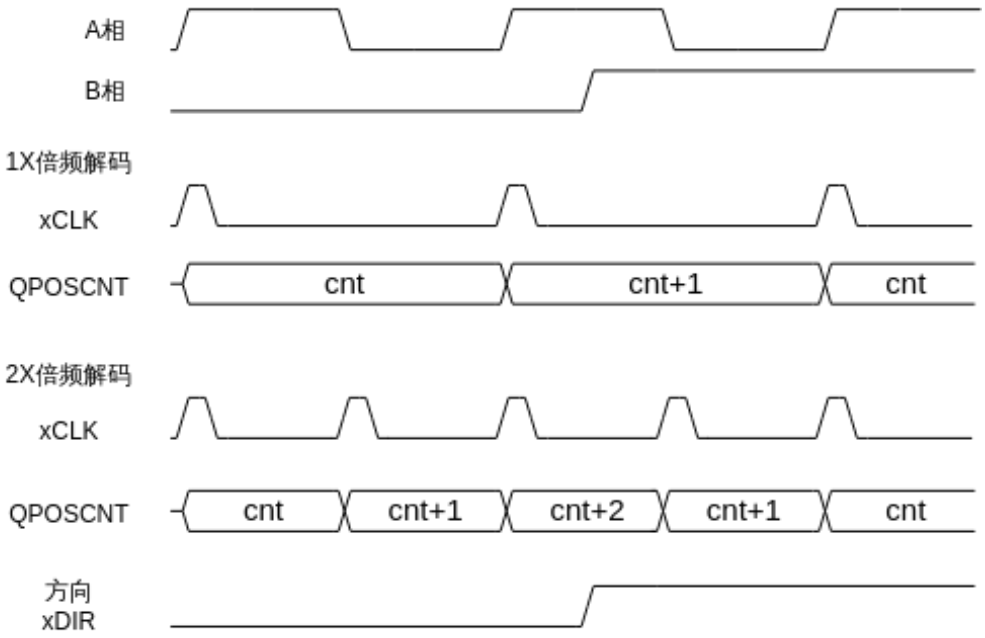


图13-10 非标 TYPE1 型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）

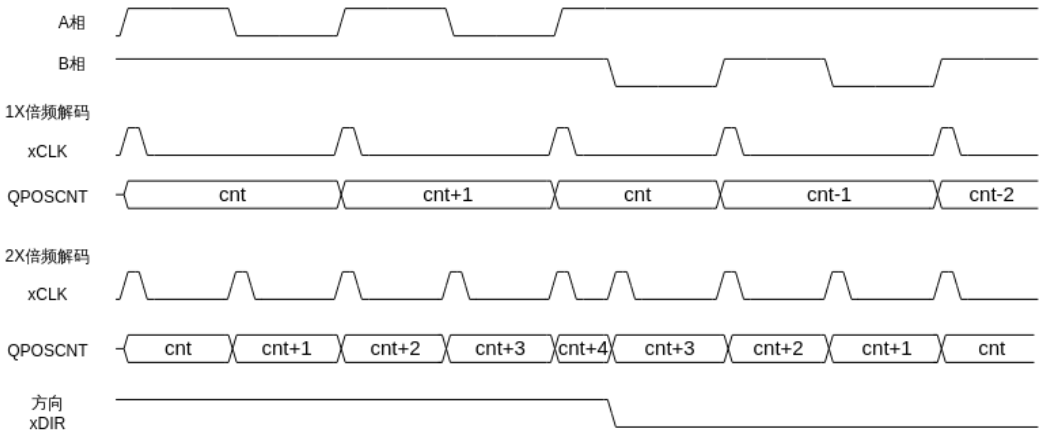
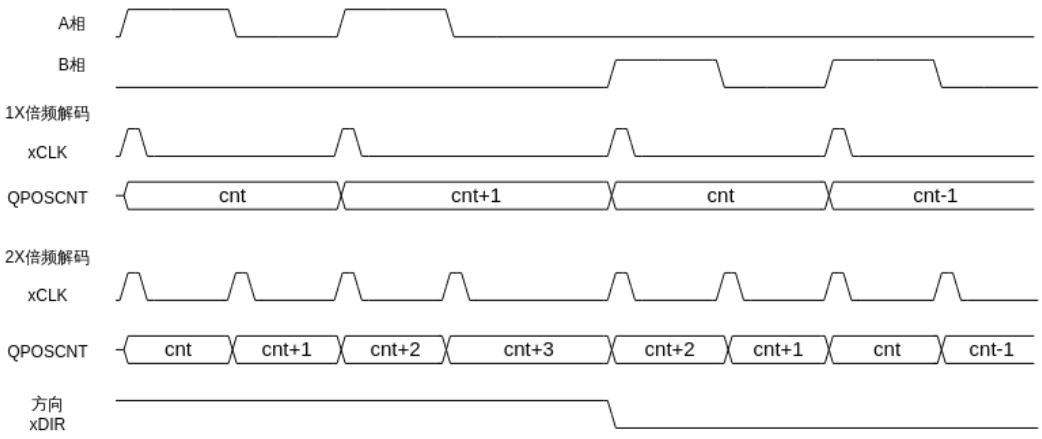


图13-11 非标 TYPE2 型 1X/2X 倍频解码图示（xCLK-xDIR 计数模式，即 QPPUCTRL[pcnt_mode]=0x0）



须知

只有正交型解码模式支持 1X/2X/4X 倍频解码，脉冲方向型和非标准型解码只支持 1X/2X 倍频解码。

13.3.4 PPU 位置计数/复位/初始化/锁存模式

13.3.4.1 3 种位置计数模式

配置 QPPUCTRL[pcnt_mode]选择不同的位置计数模式。

- xCLK-xDIR 计数模式：PPU 对倍频解码后产生的脉冲 xCLK 计数，根据方向 xDIR 进行增计数或者减计数。计数上溢出（即 QPOSCNT 等于 QPOSMAX 且继续增计数）自动清零，并且产生位置计数器上溢出中断 pcnt_ovf_int。计数下溢出（即 QPOSCNT 等于 0 且继续减计数）自动置为最大值 QPOSMAX，并且产生位置计数器下溢出中断 pcnt_udf_int。
- 增计数模式：PPU 对倍频解码后产生的脉冲 xCLK 进行固定增计数。计数上溢出（即 QPOSCNT 等于 QPOSMAX 且继续增计数）自动清零，并且产生位置计数器上溢出中断 pcnt_ovf_int。
- 减计数模式：PPU 对倍频解码后产生的脉冲 xCLK 进行固定减计数。计数下溢出（即 QPOSCNT 等于 0 且继续减计数）自动置为最大值 QPOSMAX，并且产生位置计数器下溢出中断 pcnt_udf_int。

13.3.4.2 4 种位置计数复位模式

配置 QPPUCTRL[pcnt_rst_mode]选择不同的位置计数复位模式。

- 硬件全自动校准（Z 相标记点复位）：第一次旋转经过码盘 Z 相光栅，在光栅从关闭到开启切换时，记录此时的物理位置（即为 Z 相标记点），根据此时旋转方向复位位置计数（顺时针（xDIR=1）则清零位置计数，逆时针（xDIR=0）则置位位置计数为 QPOSMAX），完成第一次位置计数（QPOSCNT）的校准。之后每次码盘旋转到 Z 相标记点时，都会进行校准。编码器顺时针旋转（xDIR=1）时，在 Z 相标记点处，QPOSCNT 置为 0，随着编码器顺时针旋转（xDIR=1），QPOSCNT 从 0 开始增计数；编码器逆时针旋转（xDIR=0）时，在 Z 相标记点处，QPOSCNT 置为 QPOSMAX，随着编码器逆时针旋转（xDIR=0），QPOSCNT 从 QPOSMAX 开始减计数。
- 溢出复位：上溢出后清零，下溢出后置位到 QPOSMAX。
- 硬件单次校准：第一次旋转经过码盘 Z 相光栅，在光栅从关闭到开启切换时，记录此时的物理位置（即为 Z 相标记点）。根据此时旋转方向复位位置计数（顺时针（xDIR=1）则清零位置计数，逆时针（xDIR=0）则置位位置计数为 QPOSMAX），完成单次校准。之后位置计数为溢出复位。
- 周期触发复位（PTU 触发复位）：每当 PTU 计数（QUTMR）到周期（QUPRD）时，根据此时旋转方向复位位置计数（顺时针（xDIR=1）则清零位置计数，逆时针（xDIR=0）则置位位置计数为 QPOSMAX）。

13.3.4.3 软件初始化位置计数器

- 软件配置 QPPUCTRL[pcnt_sw_init]为 1，开启软件初始化，QPOSCNT 初始化为 QPOSINIT 且保持。软件配置 QPPUCTRL[pcnt_sw_init]为 0，关闭软件初始化。

13.3.4.4 3 种 Z 相初始化位置计数器模式

配置 QPPUCTRL[pcnt_idx_init_mode]选择 Z 相初始化位置计数器模式。

- 硬件全自动初始化：第一次旋转经过码盘 Z 相光栅，在光栅从关闭到开启切换时，记录此时的物理位置（即为 Z 相标记点），将位置计数（QPOSCNT）初始化为软件配置的初始化值（QPOSINIT），完成第一次位置计数（QPOSCNT）的初始化。之后每次码盘旋转到 Z 相标记点时，都会进行初始化。
- Z 相上升沿初始化：Z 相上升沿时，将 QPOSCNT 初始化为 QPOSINIT。
- Z 相下降沿初始化：Z 相下降沿时，将 QPOSCNT 初始化为 QPOSINIT。

请参见 13.6 QDM 寄存器描述章节获得每个位域的详细信息。

13.3.4.5 位置计数锁存

PPU 支持根据 Z 相信号将当前位置计数值 QPOSCNT 锁存到 QPOSILOCK 寄存器，配置 QPPUCTRL[pcnt_idx_lock_mode]选择 Z 相锁存模式。

- Z 相标记点锁存（硬件全自动锁存）：Z 相标记点时，将 QPOSCNT 锁存到 QPOSILOCK。
- Z 相上升沿锁存：Z 相上升沿时，将 QPOSCNT 锁存到 QPOSILOCK。
- Z 相下降沿锁存：Z 相下降沿时，将 QPOSCNT 锁存到 QPOSILOCK。

PPU 支持根据 PTU 的周期触发信号或者 QPOSCNT 的读操作将 QPOSCNT 锁存到 QPOSLOCK 寄存器。配置 QCTRL[qtrg_lock_mode]选择触发源。

须知

Z 相标记点：用于硬件全自动校准（Z 相标记点复位），也可以用于硬件全自动初始化，还可以用于硬件全自动锁存（Z 相标记点锁存）。复位的优先级高于初始化，也就是说当使能了硬件全自动校准，也使能了硬件全自动初始化时，只有硬件全自动校准会生效。当校准和锁存均选择 Z 相标记点时，每次经过 Z 相标记点，均会先锁存后校准。

13.3.5 PPU 位置比较和同步输出

- 支持 PPU 位置计数比较功能。

  PPU 支持位置比较，可通过配置 QPOSCMP 调整比较值，可通过回读 QPOSCMPA 得到实际生效的值，可通过配置 QPPUCTRL[ppu_poscmp_en]使能位置比较功能。当位置计数 QPOSCNT 等于 QPOSCMPA 时，位置比较匹配，产生位置比较匹配中断 pcnt_cpm_int。

- 支持位置比较缓存模式，支持 2 种缓存加载模式（load 模式）。

  PPU 位置比较值 QPOSCMP 支持缓存模式，可通过配置 QPPUCTRL[ppu_cmpshd_en]使能缓存模式。不使能缓存模式时，QPOSCMPA 实时等于 QPOSCMP。使能缓存模式后，QPOSCMP 支持 2 种缓存加载模式（即 QPOSCMP 加载到 QPOSCMPA），可通过配置 QPPUCTRL[ppu_cmpshd_ld]调整缓存寄存器的加载模式。

- 支持同步输出。

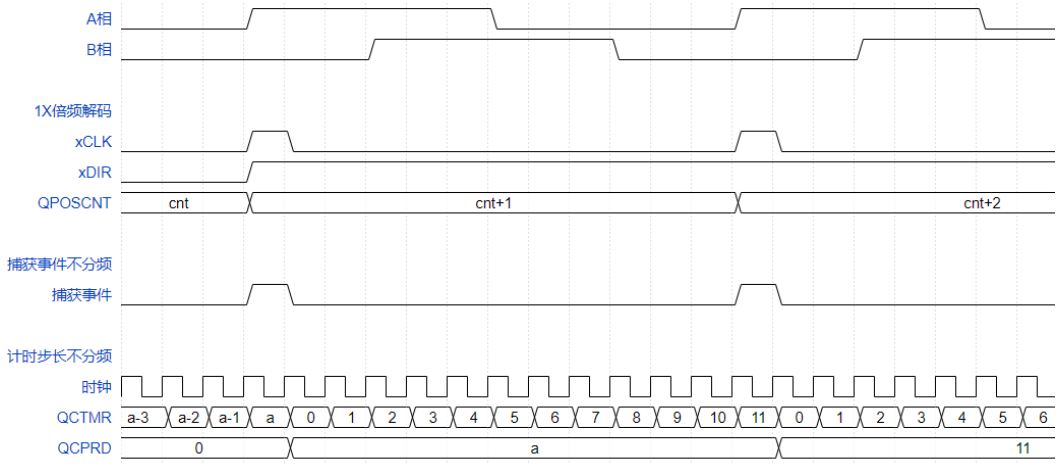
  PPU 支持位置比较匹配时，输出同步信号，可通过配置 QPPUCTRL[syncout_en]使能同步输出功能，可通过配置 QPPUCTRL[ppu_syncout_pl]和[ppu_syncout_pw]调整同步信号的极性和脉宽。

13.3.6 TSU 计时和时间戳

- 软件可通过配置 QCTRL[ptu_en]使能 TSU。
- TSU 计时步长（即 QCTMR 每加 1 所经过的时间）可通过配置 QTSUCTRL[tsu_prescaler]改变。
- TSU 时间戳保存在 QCPRD 中，记录的是捕获事件（即相邻的位置脉冲 xCLK 或位置脉冲 xCLK 的分频）经过的时间，可通过配置 QTSUCTRL[cevt_prescaler]改变捕获事件分频。

如图 13-12 所示，在 1X 倍频解码下，A 相上升沿解码得到捕获事件。当捕获事件有效时，将 QCTMR 的值锁存到 QCPRD 中。

图13-12 TSU 计时和时间戳示例



TSU 支持在触发源触发时将 QCTMR 和 QCPRD 锁存到 QCTMRLOCK 和 QCPRDLOCK。触发源有 PTU 的周期触发信号和 QPOSCNT 的读操作，配置 QTSUCTRL[qtmr_lock_mode]选择触发源。

请参见 13.6 QDM 寄存器描述章节获得每个位域的详细信息。

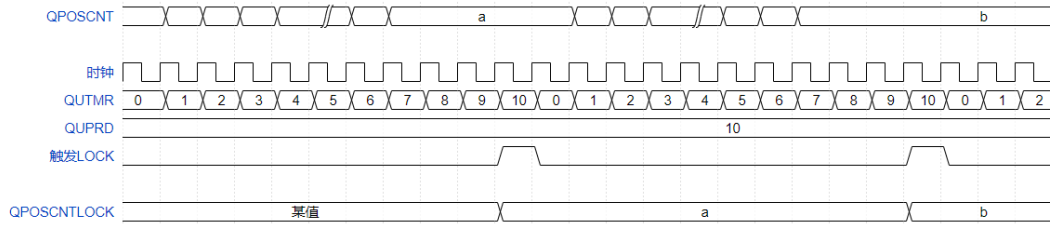
13.3.7 PTU 周期模式/计时/周期触发

- 软件可通过配置 QCTRL[ptu_en]使能 PTU。
- PTU 计时步长为 QDM 的时钟周期 pclk_qdm。
- PTU 计时周期可配置，真实周期等于 QUPRD+1，通过配置 QUPRD 改变周期。

- PTU 可触发 QDM 中断，可通过配置 QINTENA[utmr_prd_en]使能中断。当 PTU 计数达到配置的周期后，上报中断，中断状态可通过查询 QINTSTS[utmr_prd_int]得到。
- PTU 计时达到周期后，产生周期触发信号。该信号可触发 PPU 位置计数锁存，将 QPOSCNT 的值锁存到 QPOSLOCK，可通过配置 QCTRL[qtrg_lock_mode]改变锁存模式。该信号也可触发 TSU 计时锁存，QCTMR 的值锁存到 QCTMRLOCK，QCPRD 的值锁存到 QCPRDLOCK，可通过配置 QTSUCTRL[qtmr_lock_mode]改变锁存模式。

如图 13-13 所示，当 QUTMR 计数到预先配置的 QUPRD 时，周期触发锁存，将 QPOSCNT 的值锁存到 QPOSCNTLOCK 中。

图13-13 PTU 计数和周期触发示例



请参见 13.6 QDM 寄存器描述章节获得每个位域的详细信息。

13.3.8 PTU 看门狗模式

PTU 支持看门狗模式，位宽 32bit，检测 A/B 信号输入；

软件可通过配置 QCTRL[ptu_mode]改变 PTU 的工作模式。PTU 看门狗周期可通过配置 QUPRD 调整。PTU 从 0 开始计时，利用倍频解码后的 xCLK 自动喂狗（清零）。若 PTU 从 0 计时到周期过程中，喂狗失败（即 xCLK 失效），则产生失速中断 sped_lst_int，且 PTU 计时自动清零。

须知

PTU 周期模式和看门狗模式为二选一，无法同时使用。

请参见 13.6 QDM 寄存器描述章节获得每个位域的详细信息。

13.4 典型配置和部分场景的使用建议

13.4.1 编码器的解码/位置计数的校准和初始化

13.4.1.1 PPU 位置计数和硬件全自动校准的典型配置流程

步骤 1 根据具体应用选定的编码器，配置 QDM 解码模式（QCTRL[qdu_mode]），A 相 B 相（经过极性选择后）信号是否交换（QCTRL[qdm_ab_swap]），A 相 B 相 Z 相输入极性选择（QCTRL[qdma_polarity]、QCTRL[qdmb_polarity]、QCTRL[qdmi_polarity]）以及滤波（QDMAFT、QDMBFT、QDMIFT）。

步骤 2 根据应用需要的位置精确度，配置位置脉冲倍频（QCTRL[qdu_xclk]）。

步骤 3 配置位置计数模式（QPPUCTRL[pcnt_mode]）为 xCLK-xDIR 计数模式（0x0）。如果此时顺时针旋转（即 xDIR=1），位置计数根据位置脉冲（xCLK）递增；反之递减。

步骤 4 配置位置计数复位（QPPUCTRL[pcnt_rst_mode]）为硬件全自动校准，即 Z 相标记位复位（0x0）。或者配置位置计数初始化（QPPUCTRL[pcnt_idx_init_mode]）为硬件全自动初始化，即 Z 相标记点初始化（0x1）。

步骤 5 根据编码器线数和位置脉冲倍频，配置位置计数最大值（QPOSMAX）= 线数 × 位置脉冲倍数-1。

步骤 6 使能 PPU，配置 QCTRL[ppu_en]为 0x1。

----结束

13.4.1.2 PPU 位置计数和硬件单次校准的典型配置流程

步骤 1 根据具体应用选定的编码器，配置 QDM 解码模式（QCTRL[qdu_mode]），A 相 B 相（经过极性选择后）信号是否交换（QCTRL[qdm_ab_swap]），A 相 B 相 Z 相输入极性选择（QCTRL[qdma_polarity]、QCTRL[qdmb_polarity]、QCTRL[qdmi_polarity]）。

步骤 2 根据应用需要的位置精确度，配置位置脉冲倍频（QCTRL[qdu_xclk]）。

步骤 3 配置位置计数模式（QPPUCTRL[pcnt_mode]）为 xCLK-xDIR 计数模式（0x0）。如果此时顺时针旋转（xDIR=1），位置计数根据位置脉冲（xCLK）递增；反之递减。

步骤 4 配置位置计数复位（QPPUCTRL[pcnt_rst_mode]）为硬件单次校准（0x2）。

步骤 5 根据编码器线数和位置脉冲倍频，配置位置计数最大值（QPOSMAX）= 线数 × 位置脉冲倍数-1。

步骤 6 使能 PPU，配置 QCTRL[ppu_en]为 0x1。

----结束

13.4.1.3 PPU 位置计数和软件校准的典型配置流程

步骤 1 根据具体应用选定的编码器，配置 QDM 解码模式（QCTRL[qdu_mode]），A 相 B 相（经过极性选择后）信号是否交换（QCTRL[qdm_ab_swap]），A 相 B 相 Z 相输入极性选择（QCTRL[qdma_polarity]、QCTRL[qdmb_polarity]、QCTRL[qdmi_polarity]）。

步骤 2 根据应用需要的位置精确度，配置位置脉冲倍频（QCTRL[qdu_xclk]）。

步骤 3 配置位置计数模式（QPPUCTRL[pcnt_mode]）为 xCLK-xDIR 计数模式（0x0）。此时顺时针旋转（xDIR=1），位置计数根据位置脉冲（xCLK）递增；反之递减。

步骤 4 配置位置计数复位（QPPUCTRL[pcnt_rst_mode]）为溢出复位（0x1）。

步骤 5 根据编码器线数和位置脉冲倍频，配置位置计数最大值（QPOSMAX）= 线数 × 位置脉冲倍数 -1。

步骤 6 使能 PPU，配置 QCTRL[ppu_en]为 0x1。

步骤 7 软件需要校准时，直接将校准值写入 QPOSCNT。或者将校准值写入 QPOSINIT，再配置软件初始化（QPPUCTRL[pcnt_sw_init]）为 0x1，此时 QPOSCNT 等于 QPOSINIT 且不会计数，直到软件配置 QPPUCTRL[pcnt_sw_init]为 0x0。

----结束

13.4.1.4 PTU 看门狗模式的典型配置流程

步骤 1 参考 13.4.1 编码器的解码/位置计数的校准和初始化章节配置编码器的解码、校准等。

步骤 2 配置 PTU 模式（QCTRL[ptu_mode]）为看门狗模式（0x1）。此时解码倍频后的 xCLK 信号会自动喂狗。

步骤 3 配置 PTU 计数终值，即喂狗超时值（QUPRD）。

步骤 4 打开中断使能，配置失速中断使能（QINTENA[sped_lst_en]）为 0x1。

步骤 5 使能 PTU，配置 QCTRL[ptu_en]为 0x1。

步骤 6 若编码器失速，xCLK 静默，未在 QUPRD 内完成喂狗，则上报失速中断。

----结束

须知

看门狗和 M 法速度计算无法同时使用，分别对应 PTU 的 2 种工作模式，即看门狗模式和周期触发模式。

13.4.2 速度计算

13.4.2.1 M 法计算速度

M 法测量给定时间区间（测量周期 T）内的转动角度Δx = x(t+T) - x(t)，通过平均转速 V=Δx/T 计算出测量周期内的平均转速。当测量周期 T 趋近于 0 时，平均转速 V 取极限为瞬时转速。该算法需要用到 QDM 解码，PPU 位置计数和 PTU 周期计时。

典型配置如下：

步骤 1 配置触发锁存模式（QCTRL[qtrg_lock_mode]）为周期触发（0x1）。

步骤 2 参考 13.4.1 编码器的解码、位置计数的校准和初始化章节配置编码器的解码、校准等。

步骤 3 配置 PTU 工作模式（QCTRL[ptu_mode]）为周期工作模式（0x0）。

步骤 4 配置 PTU 周期 QUPRD 为 $\frac{测量周期T}{f_{pclk_qdm}}$。

步骤 5 使能单位时间周期中断（QINTENA[utmr_prd_en]）。

步骤 6 使能 PTU，配置 QCTRL[ptu_en]为 0x1。

步骤 7 进入中断，读出位置锁存值 QPOSLOCK。

步骤 8 若位置计数复位模式（QPPUCTRL[pcnt_rst_mode]）为单位时间触发复位，转动角度Δx 对应的计数值 = QPOSLOCK 的值。

步骤 9 若位置计数复位模式为溢出复位，旋转方向未发生变化且计数未发生溢出，转动角度Δx 对应的计数值 = 本次 QPOSLOCK - 上次 QPOSLOCK。

步骤 10 时间 $T=QUPRD/f_{pclk_qdm}$，将周期 QUPRD 转换为时间物理量 T。

步骤 11 弧度 $\Delta x=\Delta x对应的计数值/(编码器线数*位置脉冲倍频)$，将位置计数值转换为弧度物理量Δx。

步骤 12 根据公式 $V(RPM)=\frac{\Delta x}{T}=\frac{\Delta x对应的计数值/(编码器线数\times 2^{qdu_xclk})}{QUPRD/f_{pclk_qdm}}\times 60$ 得到每分钟转速 RPM。

----结束

须知

当位置计数初始化模式（QPPUCTRL[pcnt_idx_init_mode]）配置为 0x1，0x2，0x3 时，Z 相会改变位置计数（QPOSCNT）。在 PTU 的单个周期内，请确保位置计数没有被初始化。

M 法速度计算精度分析：（此处为纯理论分析，不考虑软件变量类型带来的误差）

设 $m=\Delta x$对应的计数值，$T=\frac{QUPRD}{f_{pclk_qdm}}$，$L=$编码器线数

$$V(RPM)=\frac{\Delta x}{T}=\frac{\Delta x\text{对应的计数值}/(\text{编码器线数}\times 2^{qdu_xclk})}{QUPRD/f_{pclk_qdm}}\times 60=\frac{m}{T\times L\times 2^{qdu_xclk}}\times 60$$

则速度计算的粒度：

$$\Delta V(RPM)=\frac{\Delta x_2-\Delta x_1}{T}=\frac{(m+1)}{T\times L\times 2^{qdu_xclk}}\times 60-\frac{m}{T\times L\times 2^{qdu_xclk}}\times 60=\frac{1}{T\times L\times 2^{qdu_xclk}}\times 60$$

相对误差（编码器线数有限引入量化误差）：

$$RelativeError=\frac{1}{m}$$

其中常量：QUPRD，qdu_xclk 均为软件配置的值，pclk_qdm 时钟频率，编码器线数。

其中变量：m。

故 M 法更适合高转速场景下测量速度。

13.4.2.2 MT 法计算速度

MT 法测量在给定码盘的旋转区间 mX，转动消耗的时间Δt，通过平均转速 V=mX/Δt 计算出码盘在 mX 区间内的平均转速。当 mX 趋近于 0，平均转速 V 取极限为瞬时转速。

典型配置如下：

步骤 1 配置触发 TSU 锁存模式（QTSUCTRL[qtmr_lock_mode]）软件读 QPOSCNT 触发（0x0）。

步骤 2 参考 13.4.1 编码器的解码、位置计数的校准和初始化章节配置编码器的解码、校准等。

步骤 3 配置 TSU 计时步长（QTSUCTRL[tsu_prescaler]）。

步骤 4 配置捕获事件分频（QTSUCTRL[cevt_prescaler]）。计算可得，码盘旋转区间 mX 内的位置计数 = 位置脉冲倍频*捕获事件分频。

步骤 5 使能 TSU，配置 QCTRL[tsu_en]为 0x1。

步骤 6 软件可以先读 QPOSCNT，此时 QUPRD 锁存在 QCPRDLOCK 中（即Δt 时间对应的计数值）。

步骤 7 软件也可以读 QCPRD，得到Δt 时间对应的计数值。

步骤 8 时间 $\Delta t = QCPRD \times 2^{tsu_prescaler} / f_{pclk_qdm}$，将时间计数值转换为时间物理量。

步骤 9 弧度 $mx = 2^{cevt_prescaler} /(\text{编码器线数} \times \text{位置脉冲倍频})$，转换为弧度物理量。

步骤 10 根据公式

$$V(RPM) = \frac{mX}{\Delta t} = \frac{2^{cevt_prescaler} /（\text{编码器线数} \times 2^{qdu_xclk}）}{QCPRD \times 2^{tsu_prescaler} / f_{pclk_qdm}} \times 60$$

计算得到每分钟转速 RPM。

----结束

当捕获事件分频 cevt_prescaler 等于 0x0 时（即捕获事件不分频），此时 MT 法退化为 T 法。

MT 法速度计算精度分析：（此处为纯理论分析，不考虑软件变量类型带来的误差）

设：$M = 2^{cevt_prescaler}$，$T = \frac{QCPRD}{f_{pclk_qdm}}$，$L = \text{编码器线数}$

$$V(RPM) = \frac{mX}{\Delta t} = \frac{2^{cevt_prescaler} /（\text{编码器线数} \times 2^{qdu_xclk}）}{QCPRD \times 2^{tsu_prescaler} / f_{pclk_qdm}} \times 60 = \frac{M}{T \times L \times 2^{tsu_prescaler+qdu_xclk}} \times 60$$

则速度计算的粒度：$\Delta V(RPM) = \frac{mX}{\Delta t_2 - \Delta t_1} = （\frac{M}{T\text{-}1} - \frac{M}{T}） \times \frac{60}{L \times 2^{tsu_prescaler+qdu_xclk}} = \frac{M}{T \times (T\text{-}1)} \times \frac{60}{L \times 2^{tsu_prescaler+qdu_xclk}}$

相对误差（时间计数分频引入量化误差）：$RelativeError = \frac{\Delta V}{V} = \frac{1}{T\text{-}1}$

故，若希望计算粒度小，相对误差也小，须使 T 尽可能大，即 tsu_presclaer 须配置小的值，但须注意 TSU 计数是否溢出。软件通过调整 cevt_prescaler，tsu_prescaler，可以应对高低转速场景下的速度测量。

13.4.3 QDM 用作独立 TIMER

步骤 1 配置 PTU 工作模式（QCTRL[ptu_mode]）为周期工作模式（0x0）。

步骤 2 配置 PTU 周期 QUPRD。

步骤 3 打开单位时间周期中断使能（QINTENA[utmr_prd_en]）。

步骤 4 使能 PTU，配置 QCTRL[ptu_en]为 0x1。

步骤 5 PTU 会周期性上报定时中断，$周期=\frac{QUPRD}{f_{pclk_qdm}}$ 。

----结束

13.5 QDM 寄存器概览

地址空间内：

- QDM0 寄存器基地址为：0x14C0_0000。
- QDM1 寄存器基地址为：0x14C0_1000。

QDM 寄存器概览如表 13-1 所示。

表13-1 QDM 寄存器概览

偏移地址	名称	描述	页码
0x0000	QDMVER	QDM Version，QDM 版本信息寄存器	QDMVER
0x0004	QEMUMODE	QDM Emulation Mode，QDM 仿真器模式配置寄存器	QEMUMODE
0x0008	QCTRL	QDM Control，QDM 控制寄存器	QCTRL
0x000C	QPPUCTRL	QDM Position Process Unit Control，PPU 控制寄存器	QPPUCTRL
0x0010	QTSUCTRL	QDM Time Stamp Unit Control，TSU 控制寄存器	QTSUCTRL

偏移地址	名称	描述	页码
0x0014	QINTENA	QDM Interrupt Enable，QDM 中断使能寄存器	QINTENA
0x0018	QINTSTS	QDM Interrupt Status，QDM 中断状态寄存器	QINTSTS
0x001C	QINTRAW	QDM Raw Interrupt，QDM 原始中断寄存器	QINTRAW
0x0020	QINTINJ	QDM Interrupt Injection，QDM 中断注入寄存器	QINTINJ
0x0024	QDMSTS	QDM Status，QDM 状态寄存器	QDMSTS
0x0028	QPOSCNT	QDM PPU Position Counter Value，QDM PPU 位置计数值寄存器	QPOSCNT
0x002C	QPOSINIT	QDM PPU Position Counter Initialization Value，QDM PPU 位置初始化值寄存器	QPOSINIT
0x0030	QPOSMAX	QDM PPU Position Counter Maximum Value，QDM PPU 位置计数最大值寄存器	QPOSMAX
0x0034	QPOSCMP	QDM PPU Position Counter (Shadowed) Compare Value，QDM PPU 位置比较值缓存寄存器	QPOSCMP
0x0038	QPOSILOCK	QDM PPU QPOSCNT Index Locked Value，QDM Z 相信号锁存值寄存器	QPOSILOCK
0x003C	QPOSLOCK	QDM PPU QPOSCNT Locked Value，位置单位时间触发/软件锁存值寄存器	QPOSLOCK
0x0040	QUTMR	QDM PTU Counter Value，PTU 的计数值寄存器	QUTMR
0x0044	QUPRD	QDM PTU Period Value，PTU 的计数周期寄存器	QUPRD
0x0048	QCTMR	QDM TSU Counter Value，TSU 时间计数值寄存器	QCTMR
0x004C	QCMAX	QDM TSU Counter Maximum Value，TSU 时间计数最大值寄存器	QCMAX
0x0050	QCPRD	QDM TSU-captured CEVT's Period，邻近的 2 个的捕获事件 CEVT 之前的时间计数值寄存器	QCPRD
0x0054	QCTMRLOCK	QCTMR Locked Value，QCTMR 锁存值寄存器	QCPR

偏移地址	名称	描述	页码
			D
0x0058	QCPRDLOCK	QCPRD Locked Value，QCPRD 锁存值寄存器	QCTMRLOCK
0x005C	QDMAFT	QDM PHASE A Filter，A 相信号滤波寄存器	QDMAFT
0x0060	QDMBFT	QDM PHASE B Filter，B 相信号滤波寄存器	QDMBFT
0x0064	QDMIFT	QDM PHASE Z (Index) Filter，Z 相信号滤波寄存器	QDMIFT
0x0068	QPOSCMPA	QDM Position Counter Active Compare Value，位置比较值寄存器，实际用于比较的值	QPOSCMPA

13.6 QDM 寄存器描述

QDMVER

QDMVER 为 QDM Version。QDM 版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0122_0831

Bits	Access	Name	Description	Reset
[31:27]	-	reserved	保留。	0x00
[26]	RO	release_ver	版本信息。	0x0
[25]	RO	release_step	版本信息。	0x0
[24]	RO	release_substep	版本信息。	0x1
[23:16]	RO	year	年。	0x22
[15:0]	RO	month_day	月日。	0x0831

QEMUMODE

QEMUMODE 为 QDM Emulation Mode。仿真器模式配置寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1:0]	RW	emu_mode	QDM 仿真器接入模式。 00：内部计数器（QPOSCNT、QUTMR、QCTMR）当仿真器暂停 CPU 时，立即停止； 01：QPOSCNT 达到最大值复位后，停止；QUTMR 当前周期计数完成后，停止；QCTMR 在下一个 UTTRG 后，停止； 10：内部计数器不受仿真器暂停 CPU 的影响； 11：与 10 一致。	0x0

QCTRL

QCTRL 为 QDM Control。QDM 控制寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:14]	RW	qdu_mode	QDM 解码模式。 00：正交输入解码模式。输入为正交编码器 A，B，Z 相信号； 01：脉冲方向输入解码模式。输入为脉冲，方向信号； 10：非标准 TYPE1 输入解码模式； 11：非标准 TYPE2 输入解码模式。	0x0
[13:12]	RW	qdu_xclk	QDM 位置脉冲倍频解码。 00：1X 倍频。A 相上升沿； 01：2X 倍频。A 相双沿； 10：4X 倍频。A 相和 B 相双沿。仅当 qdu_mode 为正交输入解码模式时才有	0x0

Bits	Access	Name	Description	Reset
			效，其他模式时是 2X 倍频； 11：保留。	
[11]	RW	qdm_ab_swap	QDM A 相 B 相（经过极性选择后）信号交换。 0：不交换； 1：交换。	0x0
[10]	RW	qdma_polarity	QDM A 相输入极性选择。 0：直通； 1：取反。	0x0
[9]	RW	qdmb_polarity	QDM B 相输入极性选择。 0：直通； 1：取反。	0x0
[8]	RW	qdmi_polarity	QDM Z 相（Index）输入极性选择。 0：直通； 1：取反。	0x0
[7:5]	-	reserved	保留。	0x0
[4]	RW	qtrg_lock_mode	QDM 触发锁存模式选择。 0：当总线（CPU/DMA）读 QPOSCNT 时：锁存位置计数 QPOSCNT 到 QPOSLOCK； 1：当周期触发单元 PTU 使能且周期触发信号有效时：锁存位置计数 QPOSCNT 到 QPOSLOCK。	0x0
[3]	RW	ptu_mode	QDM PTU 工作模式。 0：周期工作模式。产生周期 trigger 信号； 1：看门狗工作模式。监控输入信号，用于失速检测。	0x0
[2]	RW	tsu_en	QDM TSU 时间戳单元使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能。TSU 停止计数； 1：使能。TSU 开始计数。	
[1]	RW	ppu_en	QDM 位置处理单元 PPU 使能。 0：不使能 QDM PPU。PPU 停止计数； 1：使能 QDM PPU。PPU 开始计数。	0x0
[0]	RW	ptu_en	QDM 周期触发单元 PTU 使能。 0：不使能 QDM PTU。PTU 停止计数； 1：使能 QDM PTU。PTU 开始计数。	0x0

QPPUCTRL

QPPUCTRL 为 QDM Position Process Unit Control，PPU 控制寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:29]	-	reserved	保留。	0x0
[28]	RW	pcnt_sw_init	软件初始化位置计数器。 0：软件不初始化位置计数器； 1：软件初始化位置计数器，QPOSCNT=QPOSINIT。软件写 1 后不会自动清零。	0x0
[27:26]	RW	pcnt_mode	位置计数器计数模式。 00：xCLK-xDIR 计数模式。对解码后的 xCLK，xDIR 计数。xDIR 为 0 时，表示逆时针旋转，计数器减计数。反之，增计数； 01：增计数模式。对 QDMA 增计数； 10：减计数模式。对 QDMA 减计数； 11：同 00。	0x0
[25:24]	RW	pcnt_rst_mode	QDM 位置计数器复位选择。 00：硬件全自动校准（Z 相标记点复位）；	0x0

Bits	Access	Name	Description	Reset
			01：溢出复位（上溢出后清零，下溢出后置位到 qposmax）； 10：硬件单次校准（硬件校准一次后，溢出复位）； 11：单位时间触发（PTU 触发）。	
[23:22]	RW	pcnt_idx_init_mode	QDM Z 相（Index）初始化位置计数器模式。 00：不使能； 01：硬件全自动初始化（Z 相标记点初始化），将 QPOSCNT 初始化为 QPOSINIT（优先级低于复位）； 10：Z 相（Index）上升沿时，将 QPOSCNT 初始化为 QPOSINIT； 11：Z 相（Index）下降沿时，将 QPOSCNT 初始化为 QPOSINIT。	0x0
[21:20]	RW	pcnt_idx_lock_mode	QDM Z 相（Index）锁存模式选择。（锁存 PPU 位置计数器 QPOSCNT 到 QPOSILOCK 中）。 00：同 11； 01：在 Z 相(Index)上升沿处锁存； 10：在 Z 相(Index)下降沿处锁存； 11：在 Z 相标记点锁存。	0x0
[19]	-	reserved	保留。	0x0
[18]	RW	ppu_cmpshd_en	QDM 位置比较缓存寄存器的使能。 0：不使能缓存寄存器； 1：使能缓存寄存器。	0x0
[17]	RW	ppu_cmpshd_ld	QDM 位置比较缓存寄存器 load 模式。 0：当 QPOSCNT 等于 0 时，把位置比较缓存寄存器 load 进位置比较 active 寄存器； 1：当比较成功时，把位置比较缓存寄存器	0x0

Bits	Access	Name	Description	Reset
			load 进位置比较 active 寄存器。	
[16]	RW	ppu_poscmp_en	QDM 位置比较功能使能。 0：不使能位置比较功能； 1：使能位置比较功能。	0x0
[15:14]	-	reserved	保留。	0x0
[13]	RW	syncout_en	QDM 位置比较同步输出（syncout）使能。 0：不使能同步输出功能； 1：使能同步输出功能。	0x0
[12]	RW	ppu_syncout_pl	QDM 位置比较同步输出（syncout）极性。 0：高有效脉冲输出； 1：低有效脉冲输出。	0x0
[11:0]	RW	ppu_syncout_pw	QDM 位置比较同步输出（syncout）脉宽选择。 0x000：1×4 个时钟周期； 0x001：2×4 个时钟周期； 0x002：3×4 个时钟周期； 0x003：4×4 个时钟周期； ... 0xFFF：4096×4 个时钟周期。	0x000

QTSUCTRL

QTSUCTRL 为 QDM Time Stamp Unit Control，TSU 控制寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	-	reserved	保留。	0x000000
[8]	RW	qtmr_lock_mode	QDM TSU 锁存模式。 0：软件读 QPOSCNT 时，锁存 QCTMR	0x0

Bits	Access	Name	Description	Reset
			到 QCTMRLOCK，锁存 QCPRD 到 QCPRDLOCK； 1：单位时间触发信号有效时，锁存 QCTMR 到 QCTMRLOCK，锁存 QCPRD 到 QCPRDLOCK。	
[7:4]	RW	tsu_prescaler	QDM TSU 计时步长选择。 0x0：等于时钟周期； 0x1：2 倍时钟周期； 0x2：4 倍时钟周期； 0x3：8 倍时钟周期； 0x4：16 倍时钟周期； 0x5：32 倍时钟周期； 0x6：64 倍时钟周期； 0x7：128 倍时钟周期； 0x8：256 倍时钟周期； 其他：256 倍时钟周期。	0x0
[3:0]	RW	cevt_prescaler	QDM 捕获事件 CEVT 分频选择。 0x0：捕获事件频率 = xCLK 不分频； 0x1：捕获事件频率 = xCLK 的 2 分频； 0x2：捕获事件频率 = xCLK 的 4 分频； 0x3：捕获事件频率 = xCLK 的 8 分频； 0x4：捕获事件频率 = xCLK 的 16 分频； 0x5：捕获事件频率 = xCLK 的 32 分频； 0x6：捕获事件频率 = xCLK 的 64 分频； 0x7：捕获事件频率 = xCLK 的 128 分频； 0x8：捕获事件频率 = xCLK 的 256 分频； 0x9：捕获事件频率 = xCLK 的 512 分频； 0xA：捕获事件频率 = xCLK 的 1024 分频；	0x0

Bits	Access	Name	Description	Reset
			0xB：捕获事件频率 = xCLK 的 2048 分频； 其他：保留。	

QINTENA

QINTENA 为 QDM Interrupt Enable，QDM 中断使能寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9]	RW	utmr_prd_en	QDM PTU 周期中断使能。不使能时，即屏蔽相应的中断状态，不影响相应的原始中断。若全部不使能，模块不会上报中断给 CPU。 0：不使能； 1：使能。	0x0
[8]	RW	indx_lck_en	QDM Z 相（Index）锁存功能中断使能。 0：不使能； 1：使能。	0x0
[7]	RW	pcnt_cpm_en	QDM 位置比较匹配中断使能。 0：不使能； 1：使能。	0x0
[6]	RW	pcnt_cpr_en	QDM 位置比较就绪中断使能。 0：不使能； 1：使能。	0x0
[5]	RW	pcnt_ovf_en	QDM 位置计数器上溢出中断使能。 0：不使能； 1：使能。	0x0
[4]	RW	pcnt_udf_en	QDM 位置计数器下溢出中断使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[3]	RW	qdir_chg_en	QDM 正交方向变化中断使能。 0：不使能； 1：使能。	0x0
[2]	RW	sped_lst_en	QDM 失速中断使能。 0：不使能； 1：使能。	0x0
[1]	RW	qphs_err_en	QDM 正交相位错误中断使能。 0：不使能； 1：使能。	0x0
[0]	RW	pcnt_err_en	QDM 位置计数错误中断使能。 0：不使能； 1：使能。	0x0

QINTSTS

QINTSTS 为 QDM Interrupt Status，QDM 中断状态寄存器。受中断使能控制。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9]	RO	utmr_prd_int	QDM PTU 周期中断。 0：无中断； 1：有中断。	0x0
[8]	RO	indx_lck_int	QDM Z 相（Index）锁存功能中断。 0：无中断； 1：有中断。	0x0
[7]	RO	pcnt_cpm_int	QDM 位置比较匹配中断。	0x0

Bits	Access	Name	Description	Reset
			0：无中断； 1：有中断。	
[6]	RO	pcnt_cpr_int	QDM 位置比较就绪中断。 0：无中断； 1：有中断。	0x0
[5]	RO	pcnt_ovf_int	QDM 位置计数器上溢出中断。 0：无中断； 1：有中断。	0x0
[4]	RO	pcnt_udf_int	QDM 位置计数器下溢出中断。 0：无中断； 1：有中断。	0x0
[3]	RO	qdir_chg_int	QDM 正交方向变化中断。 0：无中断； 1：有中断。	0x0
[2]	RO	sped_lst_int	QDM 失速中断。 0：无中断； 1：有中断。	0x0
[1]	RO	qphs_err_int	QDM 正交相位错误中断。 0：无中断； 1：有中断。	0x0
[0]	RO	pcnt_err_int	QDM 位置计数错误中断。 0：无中断； 1：有中断。	0x0

QINTRAW

QINTRAW 为 QDM Raw Interrupt，QDM 原始中断寄存器。不受中断使能控制。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9]	RW1C	utmr_prd_raw	QDM PTU 周期原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[8]	RW1C	indx_lck_raw	QDM Z 相（Index）锁存功能原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[7]	RW1C	pcnt_cpm_raw	QDM 位置比较匹配原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[6]	RW1C	pcnt_cpr_raw	QDM 位置比较就绪原始中断，当 QPOSCMP 缓存寄存器载入 active 寄存器时此 bit 置高。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[5]	RW1C	pcnt_ovf_raw	QDM 位置计数器上溢出原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[4]	RW1C	pcnt_udf_raw	QDM 位置计数器下溢出原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[3]	RW1C	qdir_chg_raw	QDM 正交方向变化原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[2]	RW1C	sped_lst_raw	QDM 失速原始中断。当 ptu_mode 为 1 时，才可能产生此中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0

Bits	Access	Name	Description	Reset
[1]	RW1C	qphs_err_raw	QDM 正交相位错误原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0
[0]	RW1C	pcnt_err_raw	QDM 位置计数错误原始中断。 0：无原始中断； 1：有原始中断。写 1 清除中断。	0x0

QINTINJ

QINTINJ 为 QDM Interrupt Injection，QDM 中断注入寄存器。注入原始中断。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9]	W1_PULSE	utmr_prd_inj	QDM PTU 周期中断注入。 0：不注入； 1：注入。	0x0
[8]	W1_PULSE	indx_lck_inj	QDM Z 相（Index）锁存功能中断注入。 0：不注入； 1：注入。	0x0
[7]	W1_PULSE	pcnt_cpm_inj	QDM 位置比较匹配中断注入。 0：不注入； 1：注入。	0x0
[6]	W1_PULSE	pcnt_cpr_inj	QDM 位置比较就绪中断注入。 0：不注入； 1：注入。	0x0
[5]	W1_PULSE	pcnt_ovf_inj	QDM 位置计数器上溢出中断注入。 0：不注入； 1：注入。	0x0

Bits	Access	Name	Description	Reset
[4]	W1_PULSE	pcnt_udf_inj	QDM 位置计数器下溢出中断注入。 0：不注入； 1：注入。	0x0
[3]	W1_PULSE	qdir_chg_inj	QDM 正交方向变化中断注入。 0：不注入； 1：注入。	0x0
[2]	W1_PULSE	sped_lst_inj	QDM 失速中断注入。 0：不注入； 1：注入。	0x0
[1]	W1_PULSE	qphs_err_inj	QDM 正交相位错误中断注入。 0：不注入； 1：注入。	0x0
[0]	W1_PULSE	pcnt_err_inj	QDM 位置计数错误中断注入。 0：不注入； 1：注入。	0x0

QDMSTS

QDMSTS 为 QDM Status，QDM 状态寄存器。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7]	RW1C	cevt_sts	QDM 捕获事件的状态。 0：没有检测到捕获事件； 1：检测到捕获事件。写 1 清零。	0x0
[6]	RO	qdir_sts	QDM 正交方向状态（实时值）。 0：逆时针（xDIR=0）； 1：顺时针（xDIR=1）。	0x0

Bits	Access	Name	Description	Reset
[5]	RW1C	sepd_lst_sts	QDM PTU watchdog 模式计数超出最大值，即失速状态。在 QUTMR 计数到 QUPRD 都没有 XCLK，触发失速中断。 0：未发生； 1：已发生。写 1 清零。	0x0
[4]	RW1C	qctmr_ovf_sts	QDM TSU 时间计数超出最大值，发生溢出。 0：未发生； 1：已发送。写 1 清零。	0x0
[3]	RW1C	qcdr_err_sts	QDM 捕获方向错误的状态。 0：未发生； 1：邻近捕获事件 CEVT 之间旋转方向发生了变化。写 1 清零。	0x0
[2]	RO	fidx_dir_sts	QDM 第一次经过 Z 相标记点时的方向。 0：逆时针（xDIR=0）； 1：顺时针（xDIR=1）。	0x0
[1]	RW1C	fidx_is_sts	QDM 是否第一次经过 Z 相标记点。 0：未第一次经过 Z 相标记点； 1：已第一次经过 Z 相标记点。写 1 清零。	0x0
[0]	RO	pcnt_err_sts	QDM 位置计数错误状态。当且仅当 pcnt_rst_mode 为硬件全自动校准模式且 pcnt_idx_lock_mode 为 Z 相标记点锁存时，计数错误检查功能有效。每次经过 Z 相标记点时都会更新 QPOSILOCK 的值，若该值不等于 0 且不等于 QPOSMAX，则认为位置计数错误。 0：上一次经过 Z 相标记点时，位置计数没有错误； 1：上一次经过 Z 相标记点时，位置计数有错误。	0x0

QPOSCNT

QPOSCNT 为 QDM PPU Position Counter Value，QDM PPU 位置计数值寄存器。

Offset Address: 0x0028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qposcnt	QDM 位置计数值。可读可写，仿真器接入暂停生效后，不可写。	0x00000000

QPOSINIT

QPOSINIT 为 QDM PPU Position Counter Initialization Value，QDM PPU 位置初始化值寄存器。

Offset Address: 0x002C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qposinit	QDM 位置初始化值。	0x00000000

QPOSMAX

QPOSMAX 为 QDM PPU Position Counter Maximum Value，QDM PPU 位置计数最大值寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qposmax	QDM 位置计数最大值。	0x00000000

QPOSCMP

QPOSCMP 为 QDM PPU Position Counter (Shadowed) Compare Value，QDM PPU 位置比较值缓存寄存器。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:0]	RW	qposcmp	QDM 位置比较值，缓存寄存器。	0x00000000

QPOSILOCK

QPOSILOCK 为 QDM PPU QPOSCNT Index Locked Value，QDM Z 相信号锁存值寄存器。配置 pcnt_idx_lock_mode 可改变锁存模式。

Offset Address: 0x0038 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qposilock	QDM 位置 Z 相信号（index）锁存值。 配置 pcnt_idx_lock_mode，可改变 Z 相信号（index）锁存模式。软件可写该寄存器，优先级低于锁存。	0x00000000

QPOSLOCK

QPOSLOCK 为 QDM PPU QPOSCNT Locked Value，位置单位时间触发/软件锁存值寄存器。配置 qtrg_lock_mode，可改变锁存模式。

Offset Address: 0x003C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qposlock	QDM 位置单位时间触发/软件锁存值。 配置 qtrg_lock_mode，可改变锁存模式。软件可写该寄存器，优先级低于锁存。	0x00000000

QUTMR

QUTMR 为 QDM PTU Counter Value，PTU 的计数值寄存器。配置 ptu_mode 可改变 PTU 工作模式。

Offset Address: 0x0040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qutmr	QDM 周期计数器（PTU）的计数值。	0x00000000

Bits	Access	Name	Description	Reset
			配置 ptu_mode 可改变 PTU 工作模式。可读可写，仿真器接入暂停生效后，不可写。	

QUPRD

QUPRD 为 QDM PTU Period Value，PTU 的计数周期寄存器。

Offset Address: 0x0044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	quprd	QDM 周期计数器（PTU）的计数终值。	0x00000000

QCTMR

QCTMR 为 QDM TSU Counter Value，TSU 时间计数值寄存器。

Offset Address: 0x0048 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qctmr	QDM TSU 时间计数值。可读可写，仿真器接入暂停生效后，不可写。	0x00000000

QCMAX

QCMAX 为 QDM TSU Counter Maximum Value，TSU 时间计数最大值寄存器。

Offset Address: 0x004C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qcmax	QDM TSU 时间计数最大值。	0x00000000

QCPRD

QCPRD 为 QDM TSU-captured CEVT's Period，邻近的 2 个的捕获事件 CEVT 之间的时间计数值寄存器。

Offset Address: 0x0050 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	qcprd	QDM 邻近 2 个的捕获事件 CEVT 之间的时间计数值。	0x00000000

QCTMRLOCK

QCTMRLOCK 为 QCTMR Locked Value，QCTMR 锁存值寄存器。

Offset Address: 0x0054 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qctmrlock	QDM QCTMR 锁存值。 配置 qtmr_lock_mode，可改变锁存模式。 软件可写该寄存器，优先级低于锁存。	0x00000000

QCPRDLOCK

QCPRDLOCK 为 QCPRD Locked Value，QCPRD 锁存值寄存器。

Offset Address: 0x0058 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	qcprdlock	QDM QCPRD 锁存值。 配置 qtmr_lock_mode，可改变锁存模式。 软件可写该寄存器，优先级低于锁存。	0x00000000

QDMAFT

QDMAFT 为 QDM PHASE A Filter，A 相信号滤波寄存器。

Offset Address: 0x005C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12:0]	RW	qdma_ft_level	A 相信号的滤波等级，滤波宽度 qdma_ft_level 个时钟周期。	0x0000

QDMBFT

QDMBFT 为 QDM PHASE B Filter，B 相信号滤波寄存器。

Offset Address: 0x0060 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12:0]	RW	qdmb_ft_level	B 相信号的滤波等级，滤波宽度 qdmb_ft_level 个时钟周期。	0x0000

QDMIFT

QDMIFT 为 QDM PHASE Z (Index) Filter，Z 相信号滤波寄存器。

Offset Address: 0x0064 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12:0]	RW	qdmi_ft_level	Z 相（Index）信号的滤波等级，滤波宽度 qdmi_ft_level 个时钟周期。	0x0000

QPOSCMPA

QPOSCMPA 为 QDM Position Counter Active Compare Value，位置比较值寄存器，实际用于比较的值。

Offset Address: 0x0068 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RO	qposcmpa	QDM 位置比较值，实际用于比较的值。	0x00000000

14 高级 PWM 定时器(APT)

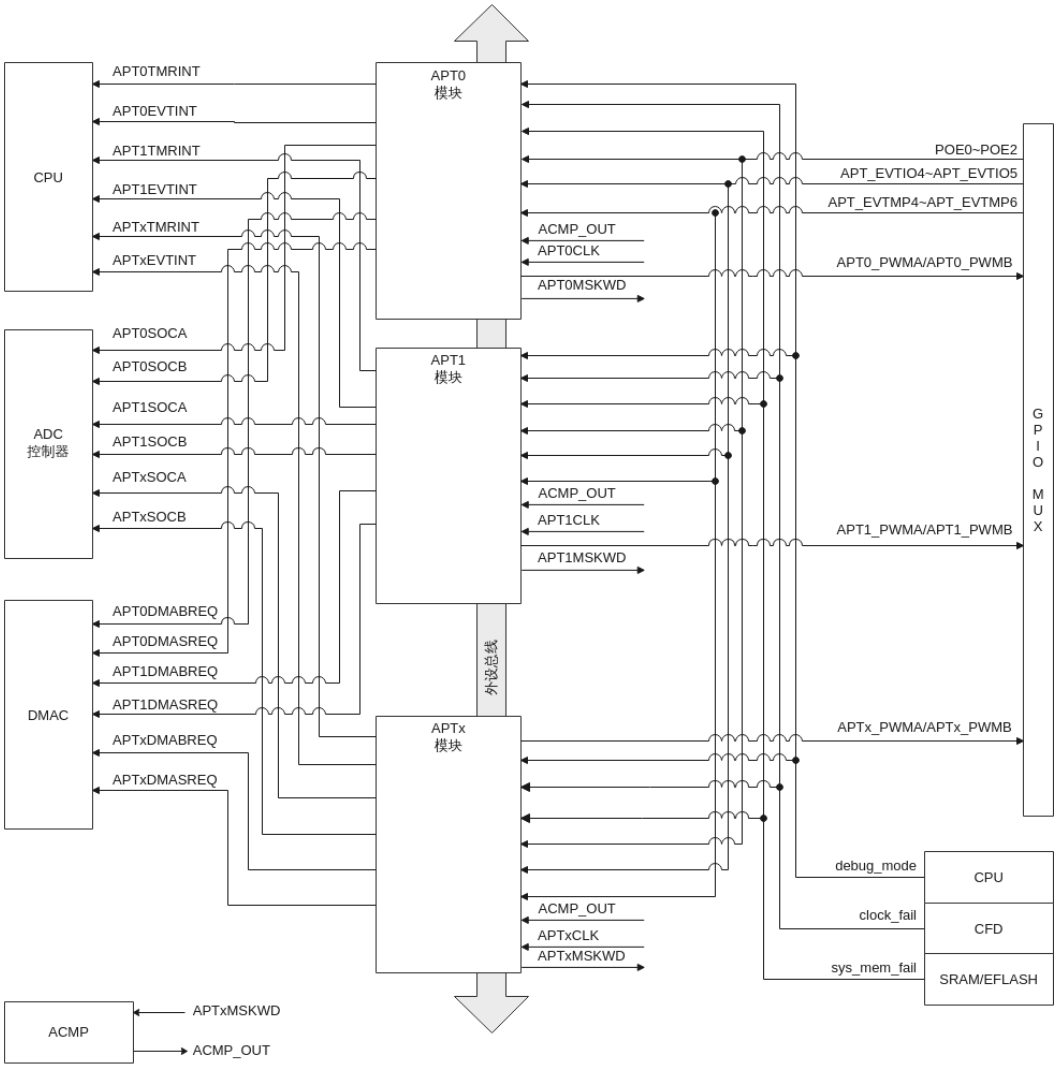
14.1 概述

APT（Advanced PWM Timer）：高级 PWM（Pulse Width Modulation）定时器，可以产生 PWM 波控制外部开关器件的导通和关断，广泛应用于数字电机控制、开关电源控制等领域。

14.1.1 APT 模块概述

一个 MCU 中包含 4 个 APT 模块，多个 APT 模块在系统中的连接如图 14-1 所示。

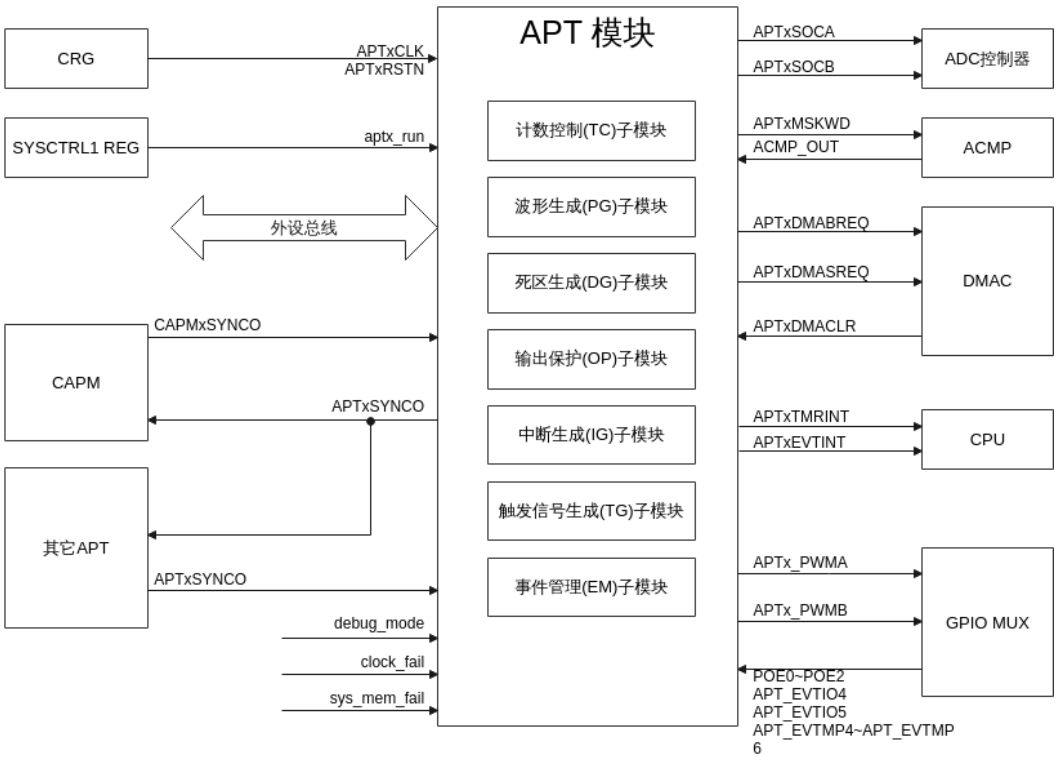
图14-1 多个 APT 模块在系统中的连接



每个 APT 模块有两个 PWM 输出：APTx_PWMA 和 APTx_PWMB，包括以下子模块：

- 计数控制子模块（Timer-Control）
- 波形生成子模块（PWM-Generation）
- 死区生成子模块（DeadBand-Generation）
- 输出保护子模块（Output-Protect）
- 中断生成子模块（Interrupt-Generation）
- 触发信号生成子模块（Trigger signal Generation）
- 事件管理子模块（Event-Management）

图14-2 单个 APT 模块的内部子模块和对外连接信号



每个 APT 模块支持以下特性：

- 用于周期和频率控制的专用 12 位分频计数器 DIVCNT 和 16 位时基计数器 TCCNT。
- 两个 PWM 输出（APTx_PWMA 和 APTx_PWMB），可配置成以下波形：
  - 两个独立 PWM 输出，单向计数非对称模式；
  - 两个独立 PWM 输出，双向计数对称模式；
  - 两个独立 PWM 输出，双向计数非对称模式。
- 通过软件对 PWM 信号进行异步控制。
- 支持与其他 APT 同步相位操作。
- 支持死区生成，具有独立的上升沿和下降沿延迟控制。
- 故障条件下支持周期性保护和持续性保护。
- 故障条件下可以强制 PWM 输出为高电平、低电平、翻转电平、锁定电平、保持动作或高阻态。
- 故障信号可以触发事件中断。

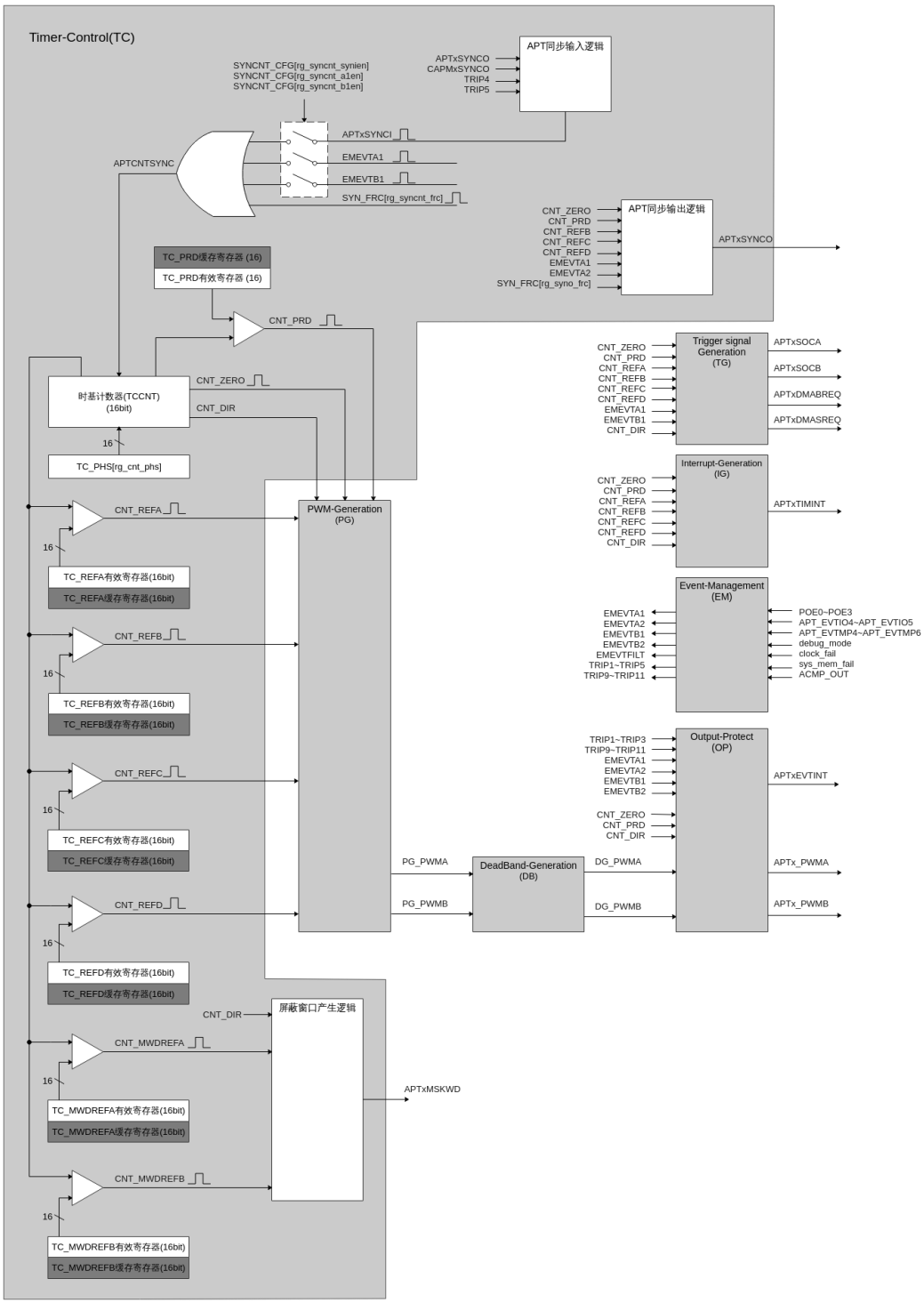
- 计数器计数到指定时刻可以触发定时中断，可以按比例缩减定时中断以减少 CPU 开销。
- 计数器计数到指定时刻可以触发 ADC 启动转换（Start Of Conversion, SOC）、产生 DMA 传输请求。

多个 APT 模块可以通过同步信号链接在一起，作为单个系统运行，也可以独立运行。

图 14-3 展示了更多 APT 模块的内部细节，主要信号有：

- PWM 输出信号（APTx_PWMA 和 APTx_PWMB）
  - PWM 输出信号输出到 MCU 管脚。
- 故障信号（POE0~POE2 信号、debug_mode 信号、clock_fail 信号、sys_mem_fail 信号）
  - 这些输入信号表示 APT 模块外部的故障，设备上的每个 APT 模块都可以配置为使用或忽略任何故障信号；
  - POE0~POE2 信号来自 MCU 引脚的输入；
  - debug_mode 信号连接 CPU 的 debug_mode 输出，支持在 CPU 进入 debug 模式时进行故障保护；
  - clock_fail 信号连接 CFD 模块的 clock_fail 输出，支持在时钟丢失时进行故障保护；
  - sys_mem_fail 信号连接系统配置控制器的 sys_mem_fail 输出，支持在 SRAM 发生奇偶校验错误或 EFALSH 发生 ECC 错误时进行故障保护。
- 同步输入信号（APTxSYNCI）、同步输出信号（APTxSYNCO）
  - 每个 APT 模块都可以使用 SYNI_CFG 寄存器配置与其他 APT 或 CAPM 同步。每个 APT 模块还可以产生同步输出信号，APTxSYNCO 的源可通过 SYNO_CFG 寄存器选择。有关更多信息，请参见 14.3.6 同步功能章节。
- 触发 ADC 启动转换信号（APTxSOCA 和 APTxSOCB）
  - 每个 APT 模块都有两个触发 ADC 启动转换信号，任何 APT 模块都可以触发转换启动，在 APT 的触发信号生成子模块中选择 APTxSOCA 和 APTxSOCB 的源。
- 事件管理子模块的输出信号
  - 事件管理子模块可以将 POE0~POE2、ACMP_OUT、APT_EVTMP4~APT_EVTMP6 信号组合产生合并事件（EMEVTA1、EMEVTA2、EMEVTB1、EMEVTB2）和滤波事件（EMEVTFILT）。

图14-3 APT 模块关键内部信号连接



14.1.2 关键指标参数

14.1.2.1 PWM 频率

- 递增/递减计数模式

$$PWM频率=\frac{工时时钟频率}{(分频系数+1)*(计数周期值+1)}$$

$$F_{pwm}=\frac{F_{APTCLK}}{(TC_MODE[rg_div_fac]+1)*（TC_PRD[rg_cnt_prd]+1）}$$

举例：

工作时钟频率 F_{APTCLK}=150MHz，分频系数 TC_MODE[rg_div_fac]配置为 9，计数周期值 TC_PRD[rg_cnt_prd]配置为 999，得到的 PWM 频率为

$$\frac{150MHz}{10*1000}=15kHz$$

。

- 先增后减计数模式

$$PWM频率=\frac{工作时钟频率}{(分频系数+1)*计数周期值*2}$$

$$F_{pwm}=\frac{F_{APTCLK}}{(TC_MODE[rg_div_fac]+1)*TC_PRD[rg_cnt_prd]*2}$$

举例：

工作时钟频率 F_{APTCLK}=150MHz，分频系数 TC_MODE[rg_div_fac]配置为 9，计数周期值 TC_PRD[rg_cnt_prd]配置为 1000，得到的 PWM 频率为

$$\frac{150MHz}{10*1000*2}=7.5kHz$$

。

分频系数 TC_MODE[rg_div_fac]的范围为 0~4095，计数周期值 TC_PRD[rg_cnt_prd]的范围为 0~65535。

14.1.2.2 PWM 周期

$$PWM周期=\frac{1}{PWM波的频率}$$

$$T_{pwm}=\frac{1}{F_{pwm}}$$

14.1.2.3 PWM 分辨率

PWM 分辨率与配置的分频系数相关：

$$PWM分辨率=工作时钟周期*(分频系数+1)=T_{APTCLK}*(TC_MODE[rg_div_fac]+1)$$

举例：当工作时钟频率为 150MHz 时，如果不使用分频（TC_MODE[rg_div_fac]=0），则产生的 PWM 分辨率为 6.67ns。如果 10 倍分频（TC_MODE[rg_div_fac]=9），则产生的 PWM 分辨率为 66.7ns。

14.1.2.4 死区的宽度

插入死区的宽度 = 工作时钟周期 T_{APTCLK}*（分频系数 TC_MODE[rg_div_fac]+1）*死区延时配置值（DG_RED[rg_dg_red]或 DG_FED[rg_dg_fed]）。

例如：当工作时钟频率为 150MHz，10 倍分频（TC_MODE[rg_div_fac]=9），死区延时配置值为 100，则$插入死区的宽度 = \frac{10*100}{150MHz} = 6.67us$。

死区延时配置值范围 0~65535。

14.1.2.5 死区分辨率

死区分辨率与配置的分频系数相关：

$$死区分辨率=工作时钟周期*(分频系数+1)=T_{APTCLK}*(TC_MODE[rg_div_fac]+1)$$

例如：当工作时钟频率为 150MHz 时，如果不使用分频（TC_MODE[rg_div_fac]=0），则死区分辨率为 6.67ns。如果 10 倍分频（TC_MODE[rg_div_fac]=9），则死区分辨率为 66.7ns。

14.1.2.6 占空比调节范围

占空比调节范围为 0~100%。

14.1.3 子模块概述

每个 APT 中包括 7 个子模块，可由软件配置每个子模块执行特定的功能，如表 14-1 所示。

表14-1 子模块配置参数

子模块	配置参数和配置选项

子模块	配置参数和配置选项
计数控制（TC）	• 对 APT 时钟（APTCLK）进行分频。 • 配置 PWM 时基计数器（TCCNT）计数周期。 • 设置时基计数器的计数模式。 – 增计数模式：用于非对称 PWM； – 减计数模式：用于非对称 PWM； – 先增后减计数模式：用于对称 PWM。 • 配置相对于其他 APT 的相位。 • 配置同步事件发生后，时基计数器的计数方向（增计数或减计数）。 • 当 MCU 处于 debug 模式时，时基计数器的行为可配置为暂停或不暂停。 • 指定 APT 模块同步输出（APTxSYNCO）的触发源。 • 指定输出 APTx_PWMA 和 APTx_PWMB 的占空比。 • 指定定时中断和触发 ADC 启动转换信号（SOC）产生的时刻。 • 产生屏蔽窗口 APTxMSKWD。 • 配置此模块中相关寄存器的缓存寄存器是否加载。
波形生成（PG）	• 指定当分频计数器和时基计数器等于计数参考值（TC_REFA/TC_REFB/TC_REFC/TC_REFD）或事件管理模块输出合并事件有效时的波形动作。 – APTx_PWMA/APTx_PWMB 变为高电平； – APTx_PWMA/APTx_PWMB 变为低电平； – APTx_PWMA/APTx_PWMB 翻转电平； – APTx_PWMA/APTx_PWMB 保持动作。 • 通过软件强制控制 PWM 输出状态。 • 配置此模块中相关寄存器的缓存寄存器是否加载。
死区生成（DG）	• 控制 APTx_PWMA 和 APTx_PWMB 之间的死区关系。 • 指定输出波形的上升沿延时值。 • 指定输出波形的下降沿延时值。 • 可屏蔽死区生成子模块，此时 PWM 波形不经过任何修改。 • 允许 APTx_PWMB 相对于 APTx_PWMA 输出存在相移。

子模块	配置参数和配置选项
	• 配置此模块中相关寄存器的缓存寄存器是否加载。
输出保护（OP）	• 配置 APT 对故障信号作出保护动作。 • 指定故障信号发生时采取的保护动作。 – 强制 APTx_PWMA/APTx_PWMB 为高电平； – 强制 APTx_PWMA/APTx_PWMB 为低电平； – 强制 APTx_PWMA/APTx_PWMB 翻转电平； – 强制 APTx_PWMA/APTx_PWMB 锁定电平； – 强制 APTx_PWMA/APTx_PWMB 保持动作； – 强制 APTx_PWMA/APTx_PWMB 为高阻态。 • 配置 APTx_PWMA/APTx_PWMB 忽略任何故障信号。 • 配置 APT 对每个故障信号的保护频率。 – 持续性保护； – 周期性保护。 • 可屏蔽输出保护子模块。 • 软件可配置，实现周期性清除保护动作。
中断生成（IG）	• 选择触发 APT 定时中断的条件。 • 选择触发 APT 事件中断的条件。 • 指定定时中断产生的频率。 • 支持中断查询和清除。
触发信号生成（TG）	• 选择触发 ADC 启动转换信号产生的条件。 • 指定触发 ADC 启动转换信号产生的频率。 • 选择 DMA Burst 请求产生的条件。 • 选择 DMA Single 请求产生的条件。
事件管理（EM）	对 POE0~POE2、ACMP_OUT、APT_EVTMP4~APT_EVTMP6 信号进行组合，产生合并事件和滤波事件。

14.2 缓存加载功能介绍

14.2.1 功能概述

APT 模块的部分寄存器支持缓存加载功能，每个支持缓存功能的寄存器都有一个有效寄存器和缓存寄存器，支持有效寄存器的更新与硬件同步：

- 有效寄存器

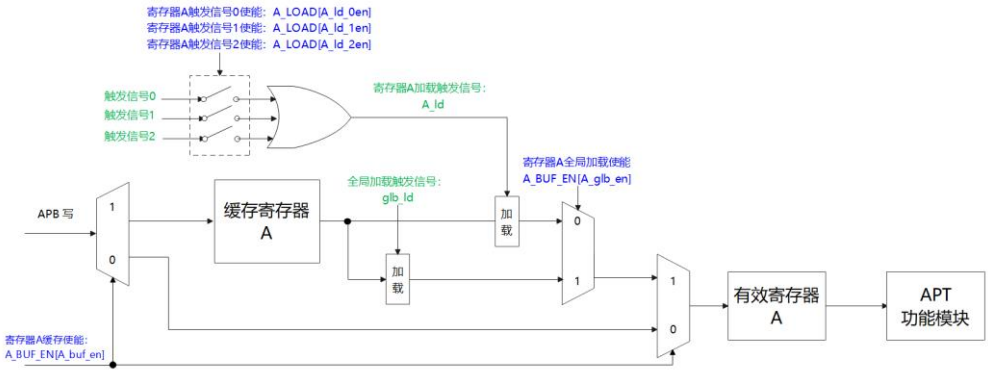
  有效寄存器直接控制硬件操作。

- 缓存寄存器

  缓存寄存器为有效寄存器提供一个临时缓存的位置，它对任何硬件操作没有直接控制作用，缓存寄存器中的值会在特定时间点加载到有效寄存器中，这可以防止由于软件异步修改寄存器而带来的错误操作。

每个支持缓存加载的寄存器都支持两种加载模式，如图 14-4 所示。

图14-4 支持缓存加载的寄存器的加载机制



- 寄存器独立加载模式

  当寄存器 A 的全局加载使能配为 0 且寄存器 A 的缓存使能配为 1 时，处于寄存器独立加载模式。当寄存器 A 的加载触发信号 A_ld 有效时，缓存寄存器加载到有效寄存器。寄存器 A 的加载触发信号 A_ld 有多个触发源，且每个触发源都支持单独使能。

- 寄存器全局加载模式

  当寄存器 A 的全局加载使能和寄存器 A 的缓存使能都配置为 1 时，处于寄存器全局加载模式。当全局加载信号 glb_ld 有效时，缓存寄存器加载到有效寄存器。全

局加载信号 glb_ld 有多个触发源，且每个触发源都支持单独使能。所有支持缓存加载功能的寄存器共享同一个全局加载信号 glb_ld。

须知

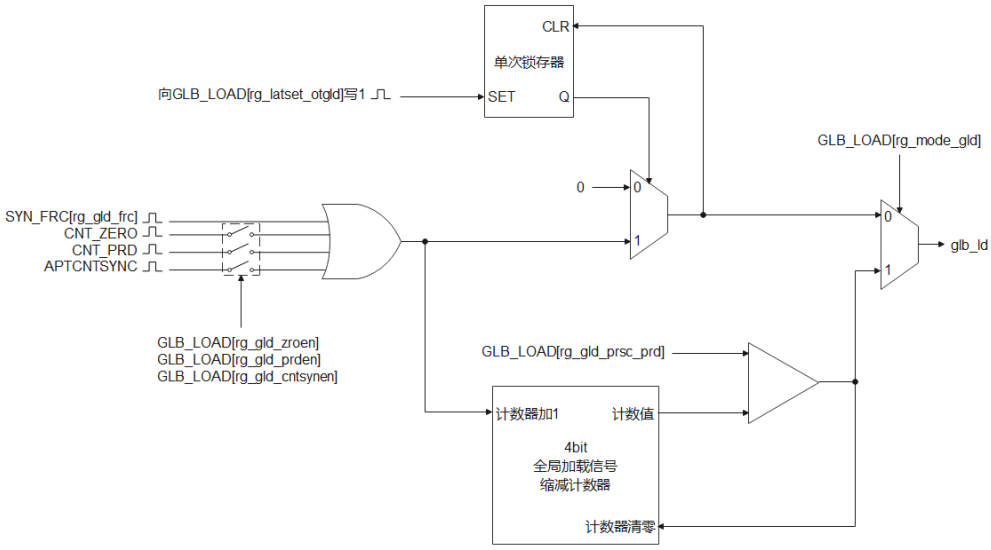
缓存的加载行为需要一个 APTCLK 周期生效。即有效寄存器的更新需要一个 APTCLK 周期生效。

DG 子模块支持缓存加载的寄存器（DG_RED/DG_FED/DG_CFG）不能和 PG/TC 子模块使用同一个触发源来触发缓存加载。

14.2.2 全局加载信号 glb_ld 生成

全局加载信号 glb_ld 的产生机制如图 14-5 所示。

图14-5 全局加载信号 glb_ld 的产生机制



glb_ld 的生成有两种模式：

- 单次加载模式：GLB_LOAD[rg_mode_gld]配置为 0 时：

  单次加载模式，是软件向 GLB_LOAD[rg_latset_otgld]写 1 后，等待触发源的第一个脉冲，作为全局加载信号 glb_ld，并且同时清除 GLB_LOAD[rg_latset_otgld]，保证只有一次触发加载生效。

- 不限次数加载模式：GLB_LOAD[rg_mode_gld]配置为 1 时：

不限次数加载模式支持对触发源进行缩减，例如：可以配置缩减系数 GLB_LOAD[rg_gld_prsc_prd]为 2，表示触发信号每触发 2 次才会生成一次全局加载信号 glb_ld。

两种模式，都需要配置触发信号的来源，触发信号的来源如表 14-2 所示。

表14-2 全局加载信号的触发源

触发信号名称	触发信号使能	触发信号描述
CNT_ZERO	GLB_LOAD[rg_gld_zroen]	分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于 0（TCCNT=0）。
CNT_PRD	GLB_LOAD[rg_gld_prden]	分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于计数周期值（TCCNT=TC_PRD[rg_cnt_prd]）。
APTCNTSYNC	GLB_LOAD[rg_gld_cntsynen]	计数器同步信号。
SYN_FRC[rg_gld_frc]	-	软件配置立即触发生效。

14.2.3 支持缓存加载的寄存器列表

表14-3 支持缓存加载的寄存器

支持缓存加载寄存器	全局加载使能	缓存使能	触发信号使能	
TC_PRD	TC_BUF_EN[rg_prd_gld_en]	TC_BUF_EN[rg_prd_buf_en]	CNT_ZERO	TC_PRD_LOAD[rg_prd_ld_zroen]
			EMEVTA1	TC_PRD_LOAD[rg_prd_ld_a1en]
			EMEVTB1	TC_PRD_LOAD[rg_prd_ld_b1en]
			APTxSYNCl	TC_PRD_LOAD[rg_prd_ld_synen]
TC_REFA	TC_BUF_EN[rg_refa_gld_en]	TC_BUF_EN[rg_refa_buf_en]	CNT_ZERO	TC_REF_LOAD[rg_refa_ld_zroen]

支持缓存加载寄存器	全局加载使能	缓存使能	触发信号使能	
			CNT_PRD	TC_REF_LOAD [rg_refa_ld_prden]
			EMEVTA1	TC_REF_LOAD [rg_refa_ld_a1en]
			EMEVTB1	TC_REF_LOAD [rg_refa_ld_b1en]
			APTxSYNCI	TC_REF_LOAD [rg_refa_ld_synen]
TC_REFB	TC_BUF_EN[rg_refb_gld_en]	TC_BUF_EN[rg_refb_buf_en]	CNT_ZERO	TC_REF_LOAD [rg_refb_ld_zroen]
			CNT_PRD	TC_REF_LOAD [rg_refb_ld_prden]
			EMEVTA1	TC_REF_LOAD [rg_refb_ld_a1en]
			EMEVTB1	TC_REF_LOAD [rg_refb_ld_b1en]
			APTxSYNCI	TC_REF_LOAD [rg_refb_ld_synen]
TC_REFC	TC_BUF_EN[rg_refc_gld_en]	TC_BUF_EN[rg_refc_buf_en]	CNT_ZERO	TC_REF_LOAD [rg_refc_ld_zroen]
			CNT_PRD	TC_REF_LOAD [rg_refc_ld_prden]
			EMEVTA1	TC_REF_LOAD [rg_refc_ld_a1en]
			EMEVTB1	TC_REF_LOAD [rg_refc_ld_b1en]
			APTxSYNCI	TC_REF_LOAD [rg_refc_ld_synen]
TC_REFD	TC_BUF_EN[rg_refd_gld_en]	TC_BUF_EN[rg_refd_buf_en]	CNT_ZERO	TC_REF_LOAD [rg_refd_ld_zroen]
			CNT_PRD	TC_REF_LOAD [rg_refd_ld_prden]
			EMEVTA1	TC_REF_LOAD

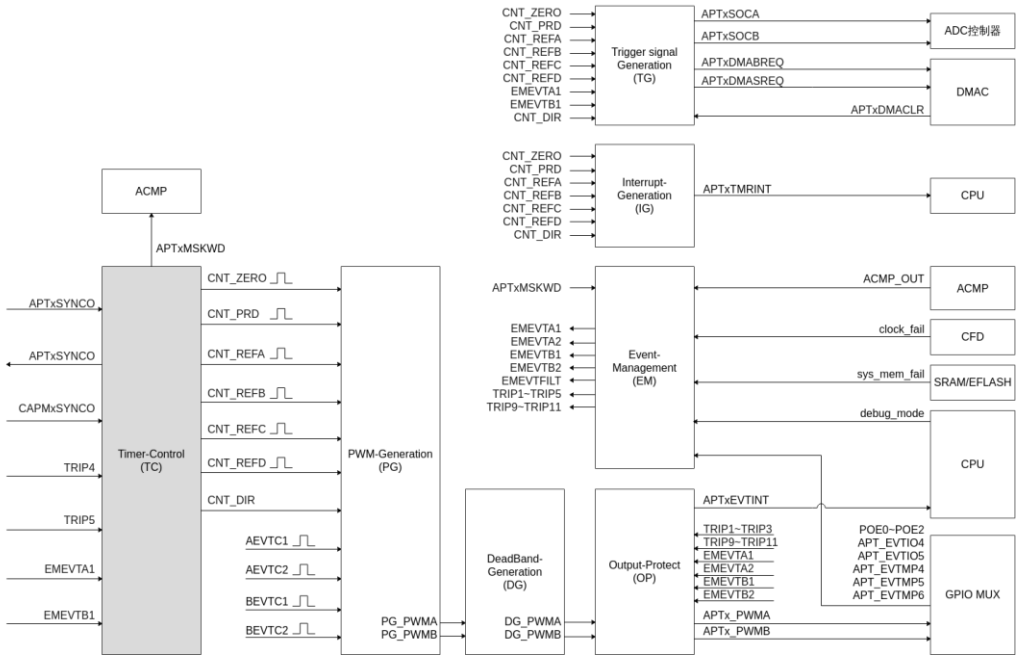
支持缓存加载寄存器	全局加载使能	缓存使能	触发信号使能	
				[rg_refd_ld_a1en]
			EMEVTB1	TC_REF_LOAD [rg_refd_ld_b1en]
			APTxSYNCI	TC_REF_LOAD [rg_refd_ld_synen]
TC_MWDREFA	TC_MWD_BUF_EN [rg_mwdrefa_gld_en]	TC_MWD_BUF_EN[rg_mwdrefa_buf_en]	CNT_ZERO	TC_MWD_LOAD [rg_mwdrefa_ld_zroen]
			CNT_PRD	TC_MWD_LOAD [rg_mwdrefa_ld_prden]
TC_MWDREFB	TC_MWD_BUF_EN [rg_mwdrefb_gld_en]	TC_MWD_BUF_EN[rg_mwdrefb_buf_en]	CNT_ZERO	TC_MWD_LOAD [rg_mwdrefb_ld_zroen]
			CNT_PRD	TC_MWD_LOAD [rg_mwdrefb_ld_prden]
TC_MWD_ACT	TC_MWD_BUF_EN [rg_mwd_act_gld_en]	TC_MWD_BUF_EN[rg_mwd_act_buf_en]	CNT_ZERO	TC_MWD_LOAD [rg_mwd_act_ld_zroen]
			CNT_PRD	TC_MWD_LOAD [rg_mwd_act_ld_prden]
PG_ACT_A	PG_BUF_EN[rg_acta_gld_en]	PG_BUF_EN[rg_acta_buf_en]	CNT_ZERO	PG_ACT_LD [rg_pga_actld_zroen]
			CNT_PRD	PG_ACT_LD [rg_pga_actld_prden]
			EMEVTA1	PG_ACT_LD [rg_pga_actld_a1en]
			EMEVTB1	PG_ACT_LD [rg_pga_actld_b1en]
			APTxSYNCI	PG_ACT_LD [rg_pga_actld_synen]
PG_ACT_B	PG_BUF_EN[rg_actb_gld_en]	PG_BUF_EN[rg_actb_buf_en]	CNT_ZERO	PG_ACT_LD [rg_pgb_actld_zroen]
			CNT_PRD	PG_ACT_LD [rg_pgb_actld_prden]
			EMEVTA1	PG_ACT_LD [rg_pgb_actld_a1en]

支持缓存加载寄存器	全局加载使能	缓存使能	触发信号使能	
			EMEVTB1	PG_ACT_LD [rg_pgb_actld_b1en]
			APTxSYNCI	PG_ACT_LD [rg_pgb_actld_synen]
PG_OUT_FRC	PG_BUF_EN[rg_frc_gld_en]	PG_BUF_EN[rg_frc_buf_en]	CNT_ZERO	PG_ACT_LD [rg_pg_frcld_zroen]
			CNT_PRD	PG_ACT_LD [rg_pg_frcld_prden]
			APTxSYNCI	PG_ACT_LD [rg_pg_frcld_synen]
DG_RED	DG_BUF_EN[rg_red_gld_en]	DG_BUF_EN[rg_red_buf_en]	CNT_ZERO	DG_BUF_LOAD [rg_red_ld_zroen]
			CNT_PRD	DG_BUF_LOAD [rg_red_ld_prden]
DG_FED	DG_BUF_EN[rg_fed_gld_en]	DG_BUF_EN[rg_fed_buf_en]	CNT_ZERO	DG_BUF_LOAD [rg_fed_ld_zroen]
			CNT_PRD	DG_BUF_LOAD [rg_fed_ld_prden]
DG_CFG	DG_BUF_EN[rg_cfg_gld_en]	DG_BUF_EN[rg_cfg_buf_en]	CNT_ZERO	DG_BUF_LOAD [rg_cfg_ld_zroen]
			CNT_PRD	DG_BUF_LOAD [rg_cfg_ld_prden]

14.3 计数控制子模块（Timer-Control）

每个 APT 模块都有独立的计数控制子模块，用作 APT 模块所有的时序基础。计数控制子模块可以控制多个 APT 模块处于同步工作模式。图 14-6 显示了计数控制子模块在 APT 中的位置。

图14-6 计数控制子模块



14.3.1 计数控制子模块的功能

计数控制子模块具有以下功能:

- 配置计数控制子模块的计数时钟频率，通过分频计数器（DIVCNT）对 APTCLK 进行分频，使得时基计数器（TCCNT）以较低的频率进行计数。
- 指定时基计数器（TCCNT）的计数周期。
- 管理与其他 APT 模块的时基计数器（TCCNT）的同步。
- 管理与其他 APT 模块保持固定相位关系。
- 将时基计数器（TCCNT）配置为增计数、减计数或先增后减计数模式。
- 产生以下比较匹配信号:
  - CNT_PRD

    分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于指定计数周期 TCCNT=TC_PRD[rg_cnt_prd])。
  - CNT_ZERO

    分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于 0（TCCNT=0)。
  - CNT_REFA

分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于计数参考值 A 寄存器 TC_REFA[rg_cnt_refa]。

- CNT_REFB

  分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于计数参考值 B 寄存器 TC_REFB[rg_cnt_refb]。

- CNT_REFC

  分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于计数参考值 C 寄存器 TC_REFC[rg_cnt_refc]。

- CNT_REFD

  分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于计数参考值 D 寄存器 TC_REFD[rg_cnt_refd]。

- 产生以下屏蔽窗口匹配信号：
  - CNT_MWDREFA

    分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于屏蔽窗口参考值 A 寄存 TC_MWDREFA[rg_mwd_refa]。

  - CNT_MWDREFB

    分频计数器（DIVCNT）等于 0（DIVCNT=0），且时基计数器（TCCNT）等于屏蔽窗口参考值 B 寄存 TC_MWDREFB[rg_mwd_refb]。

- 比较匹配信号可以用于控制 PWM 占空比。
- 屏蔽窗口匹配信号结合屏蔽窗口动作配置寄存器 TC_MWD_ACT，控制屏蔽窗口 APTxMSKWD 的产生。
- 计数周期值寄存器 TC_PRD、计数参考值寄存器 TC_REFx、屏蔽窗口参考值寄存器 TC_MWDREFx 和屏蔽窗口动作配置寄存器 TC_MWD_ACT 支持缓存加载。

14.3.2 计数控制子模块的关键信号

图 14-7 显示了计数控制子模块的关键信号和寄存器，表 14-4 列出了与计数控制子模块相关的关键信号的描述。

图14-7 计数控制子模块信号和寄存器

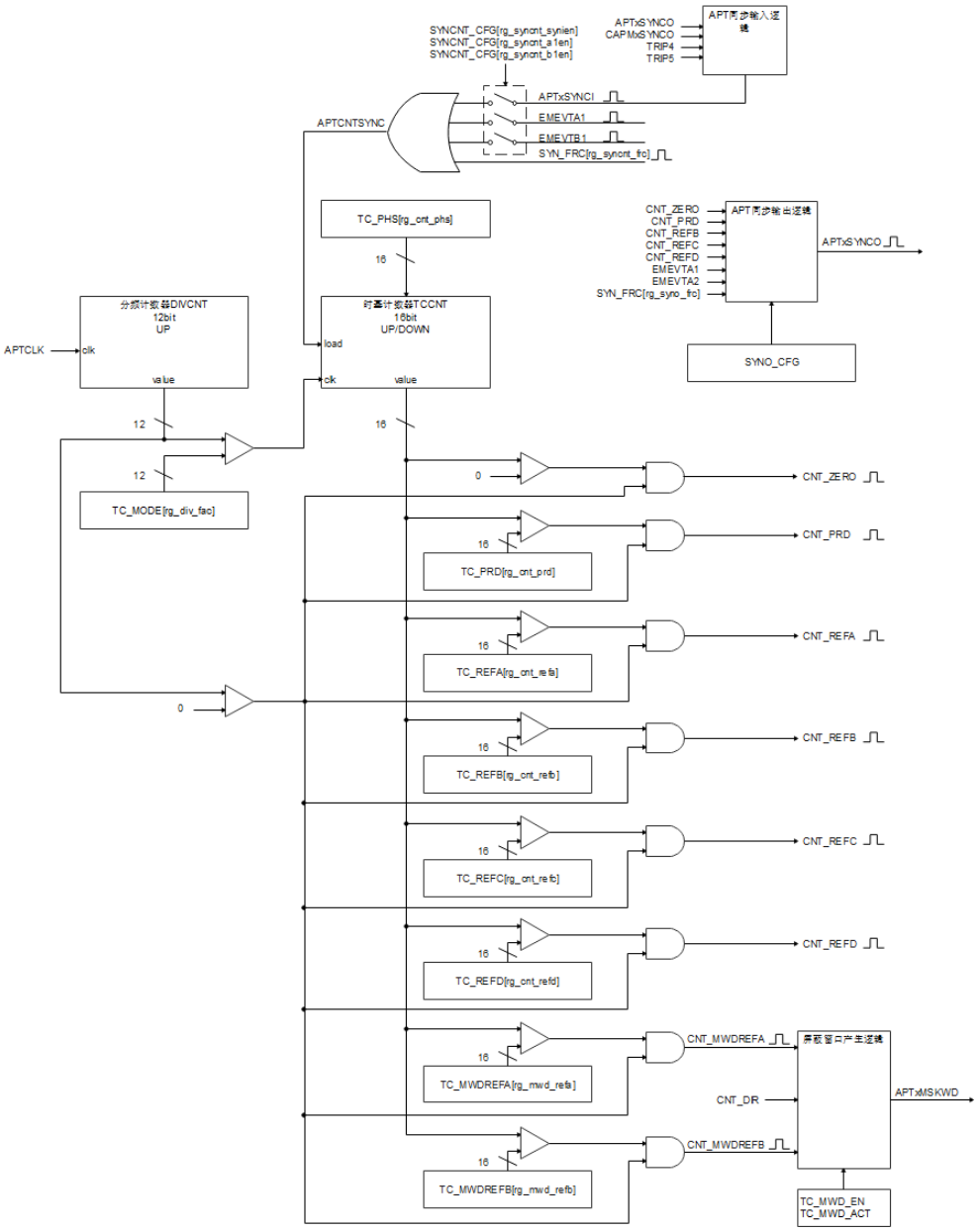


表14-4 计数控制子模块关键信号

信号	描述
APTxSYNCI	同步输入信号。 有关所有可选的同步输入脉冲源，请参见 SYNI_CFG 寄存器。有关同步功能的详细信息，请参见 14.3.6 同步功能章节。
APTxSYNCO	同步输出信号。

信号	描述
	该输出脉冲用于同步其他 APT 或 CAPM 的计数器，使用 SYNO_CFG 寄存器选择同步输出脉冲的来源。
APTCNTSYNC	计数器同步信号。 用于将时基计数器 TCCNT 同步到指定值 TC_PHS[rg_cnt_phs]。
CNT_PRD	DIVCNT 等于 0，且 TCCNT 等于计数周期值。 每当 DIVCNT=0，且 TCCNT=TC_PRD[rg_cnt_prd]，就会产生该信号。
CNT_ZERO	DIVCNT 等于 0，且 TCCNT 等于 0。 每当 DIVCNT=0，且 TCCNT=0 时，就会产生该信号。
CNT_REFA	DIVCNT 等于 0，且 TCCNT 等于计数参考值 A 寄存器 TC_REFA[rg_cnt_refa]。
CNT_REFB	DIVCNT 等于 0，且 TCCNT 等于计数参考值 B 寄存器 TC_REFB[rg_cnt_refb]。
CNT_REFC	DIVCNT 等于 0，且 TCCNT 等于计数参考值 C 寄存器 TC_REFC[rg_cnt_refc]。
CNT_REFD	DIVCNT 等于 0，且 TCCNT 等于计数参考值 D 寄存器 TC_REFD[rg_cnt_refd]。
CNTT_MWDREFA	DIVCNT 等于 0，且 TCCNT 等于屏蔽窗口参考值 A 寄存器 TC_MWDREFA[rg_mwd_refa]。
CNTT_MWDREFB	DIVCNT 等于 0，且 TCCNT 等于屏蔽窗口参考值 B 寄存器 TC_MWDREFB[rg_mwd_refb]。
CNT_DIR	TCCNT 的计数方向。 表示 APT 时基计数器的当前计数方向。当增计数时，此信号为高电平，当减计数时，此信号为低电平。
APTxMSKWD	屏蔽窗口。 用于屏蔽 EMEVTA1_TEMP、EMEVTA2_TEMP、EMEVTB1_TEMP 或 EMEVTB2_TEMP 的噪声。还可以输出到 ACMP 模块，用于屏蔽 ACMP 的输出结果 ACMP_OUT。

14.3.3 计算 PWM 的频率

PWM 的频率由分频计数器（DIVCNT）和时基计数器（TCCNT）控制，分频计数器和时基计数器的计数频率又由 TC_MODE[rg_div_fac]和 TC_PRD[rg_cnt_prd]决定，图 14-8 ~ 图 14-10 分别显示了增计数、减计数和先增后减计数模式下 PWM 频率（F_{PWM}）与 TC_MODE[rg_div_fac]和 TC_PRD[rg_cnt_prd]的关系。

时基计数器有三种工作模式，由计数模式寄存器（TC_MODE）选择：

- 增计数模式

  在增计数模式下，时基计数器从 0 开始递增，直到达到计数周期值（TC_PRD）。当达到计数周期值时，时基计数器重置为零，并再次开始递增。

- 减计数模式

  在减计数模式下，时基计数器从计数周期值（TC_PRD）开始递减，直到达到 0。当达到 0 时，时基计数器将重置为计数周期值，并再次开始递减。

- 先增后减计数模式

  在先增后减计数模式下，时基计数器从 0 开始递增，直到达到计数周期值（TC_PRD）。当达到计数周期值时，时基计数器将递减，直到达到 0，然后时基计数器重新开始递增。

图14-8 增计数模式下 PWM 频率

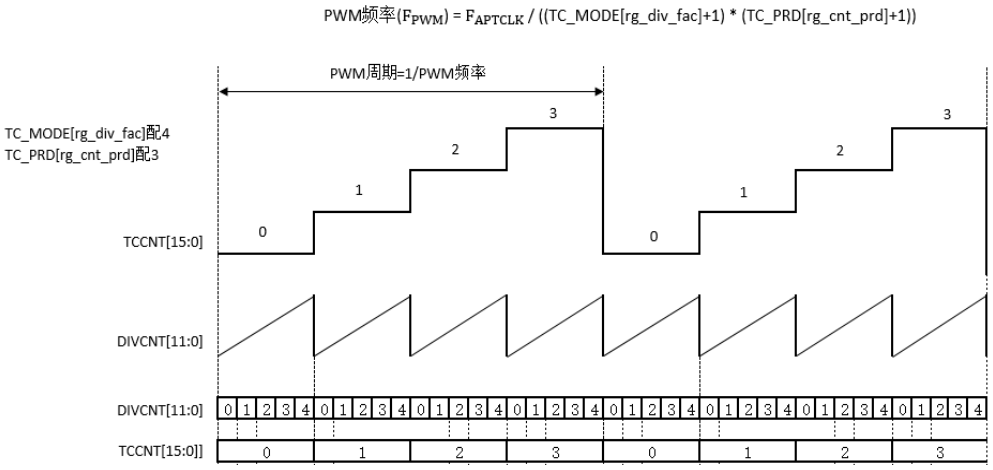


图14-9 减计数模式下 PWM 频率

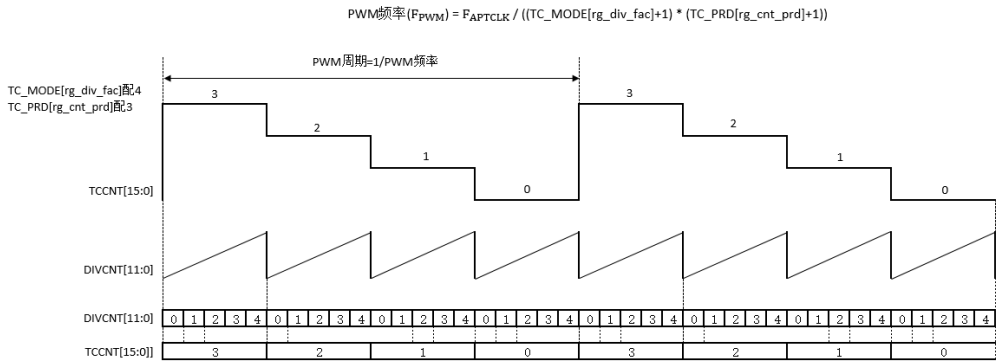
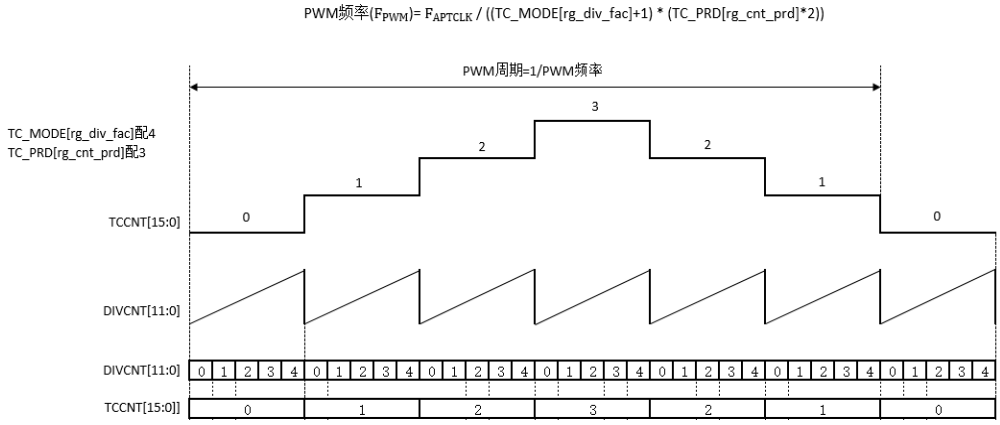


图14-10 先增后减计数模式下 PWM 频率



14.3.4 计数参考值和屏蔽窗口参考值的配置

计数控制子模块可以产生 4 个比较匹配信号和 2 个屏蔽窗口匹配信号。4 个比较匹配信号可用于波形生成子模块、中断生成子模块和触发信号生成子模块；2 个屏蔽窗口匹配信号可以用来控制屏蔽窗口 APTxMSKWD 的产生。

- CNT_REFA

  DIVCNT 等于 0，且 TCCNT 等于计数参考值 A 寄存器 TC_REFA[rg_cnt_refa]。
- CNT_REFB

  DIVCNT 等于 0，且 TCCNT 等于计数参考值 B 寄存器 TC_REFB[rg_cnt_refb]。
- CNT_REFC

  DIVCNT 等于 0，且 TCCNT 等于计数参考值 C 寄存器 TC_REFC[rg_cnt_refc]。
- CNT_REFD

  DIVCNT 等于 0，且 TCCNT 等于计数参考值 D 寄存器 TC_REFD[rg_cnt_refd]。

- CNT_MWDREFA

  DIVCNT 等于 0，且 TCCNT 等于屏蔽窗口参考值 A 寄存 TC_MWDREFA[rg_mwd_refa]。

- CNT_MWDREFB

  DIVCNT 等于 0，且 TCCNT 等于屏蔽窗口参考值 B 寄存 TC_MWDREFB[rg_mwd_refb]。

图 14-11 ~ 图 14-13 显示了增计数、减计数、先增后减计数模式下，CNT_ZERO、CNT_PRD、CNT_REFA、CNT_REFB、CNT_REFC、CNT_REFD、CNT_MWDREFA、CNT_MWDREFB 信号产生的波形图。

图14-11 增计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图

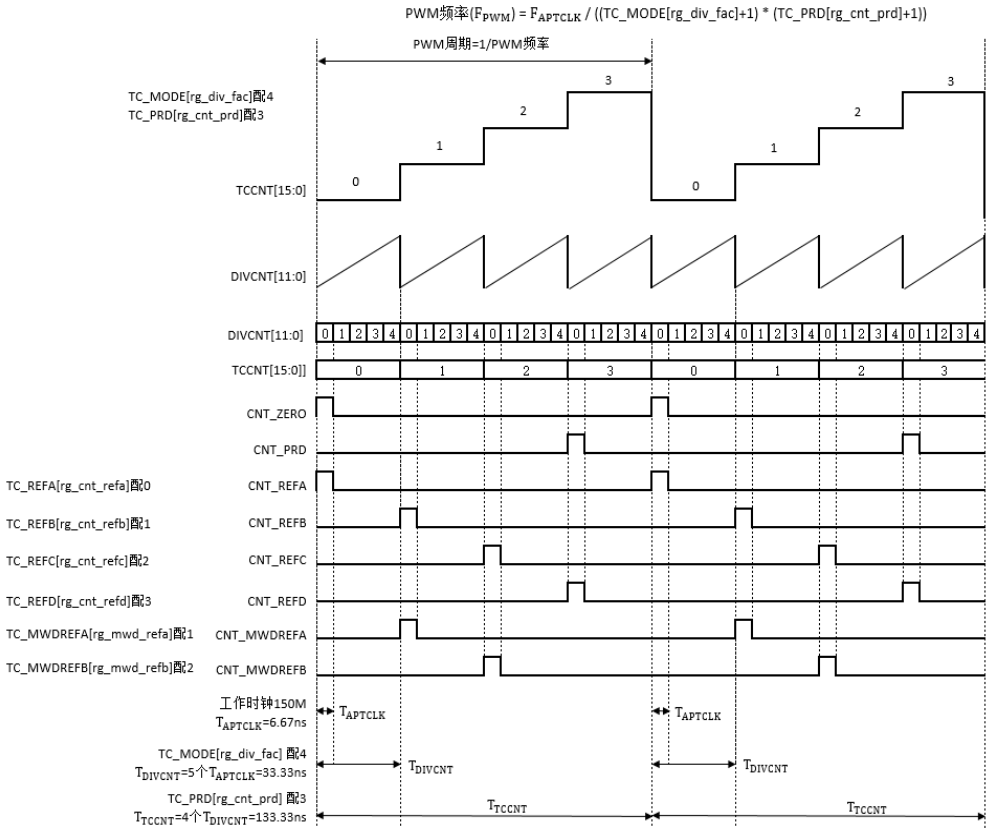


图14-12 减计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图

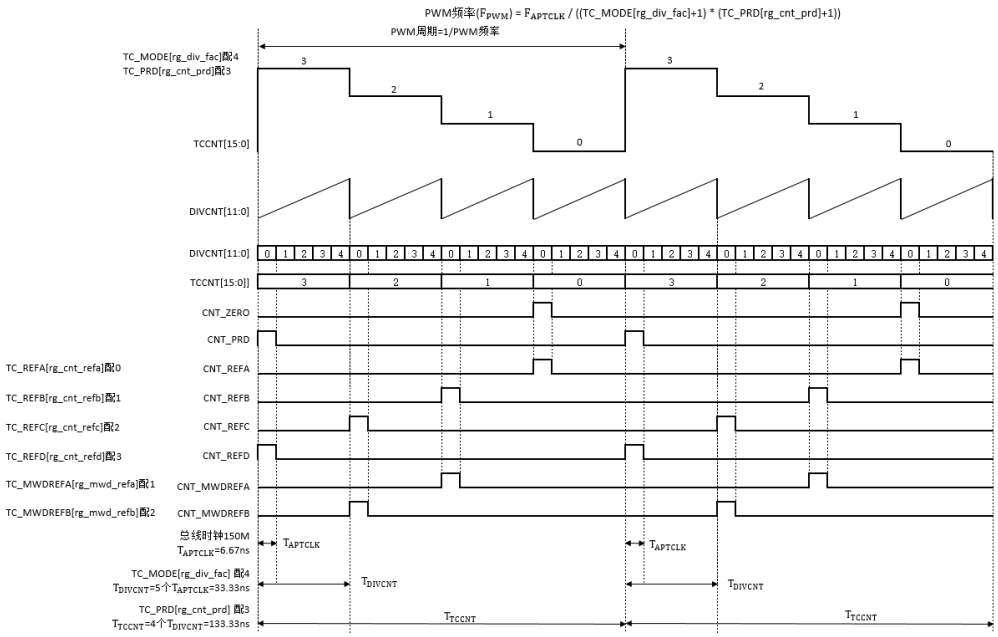
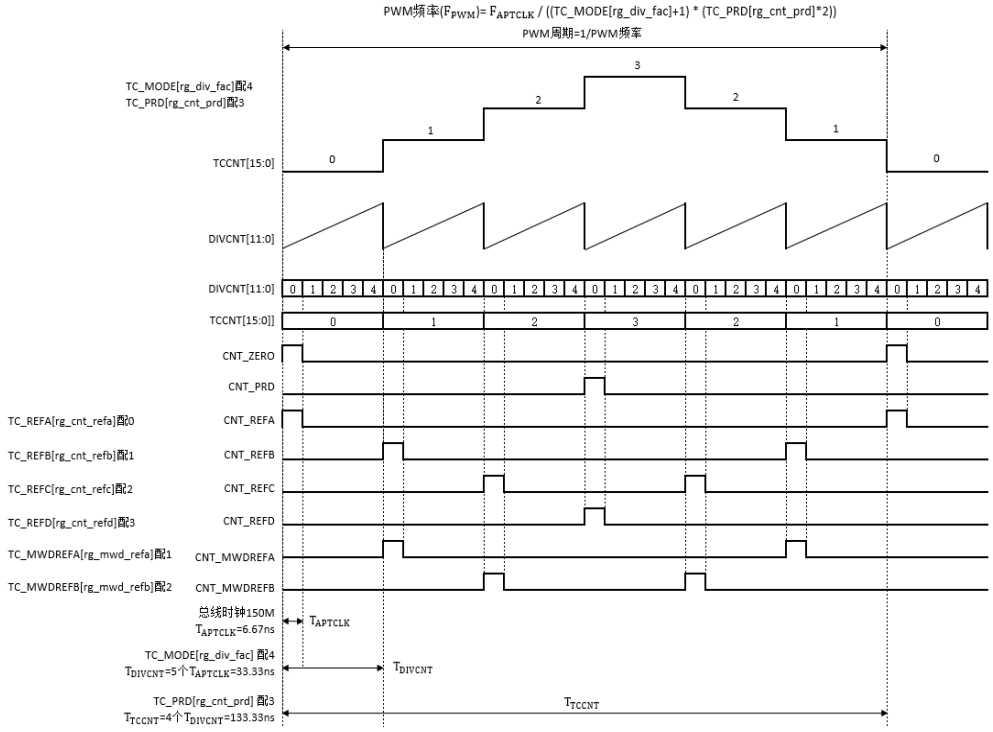


图14-13 先增后减计数模式下，比较匹配信号和屏蔽窗口匹配信号产生波形图



对于增计数和减计数模式，每个匹配信号在每个周期只发生一次。对于先增后减计数模式，如果计数参考值在 0~TC_PRD 之间，则每个匹配信号在每个周期发生两次，如果参考值等于 0 或 TC_PRD，则每个匹配信号在每个周期只发生一次。

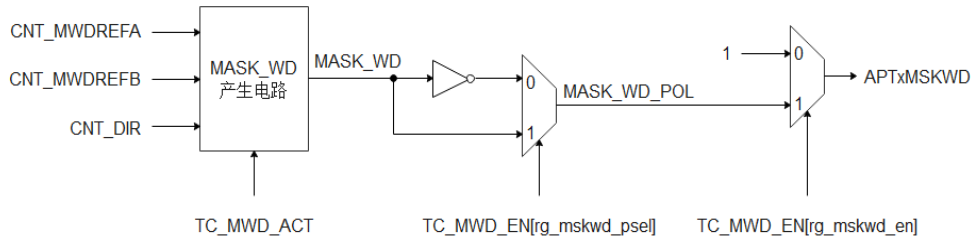
比较匹配信号作为波形动作生成子模块的输入，与计数方向一起控制 PWM 波的动作。

屏蔽窗口匹配信号与计数方向一起控制屏蔽窗口 APTxMSKWD 的产生。

14.3.5 屏蔽窗口产生逻辑

屏蔽窗口产生逻辑可以产生屏蔽窗口 APTxMSKWD，逻辑的功能框图如图 14-14 所示。

图14-14 屏蔽窗口产生逻辑功能框图



- MASK_WD 的产生

  当屏蔽窗口匹配信号 CNT_MWDREFA、CNT_MWDREFB 有效时，结合计数方向（增计数或减计数）CNT_DIR，控制 MASK_WD 的动作。

  MASK_WD 的动作由 TC_MWD_ACT 寄存器配置，可能的动作有：

  - 置为高电平

    将 MASK_WD 设置为高电平。

  - 置为低电平

    将 MASK_WD 设置为低电平。

  - 翻转电平

    如果 MASK_WD 当前为高电平，则将其设置为低电平。如果 MASK_WD 当前为低电平，则将其设置为高电平。

  - 无影响

    MASK_WD 保持为当前的电平。

- MASK_WD_POL 的产生

  MASK_WD_POL 的产生受 TC_MWD_EN[rg_mskwd_psel]控制。

当 TC_MWD_EN[rg_mskwd_psel]=0 时，MASK_WD_POL 的源是 MASK_WD 取反后的波形。

当 TC_MWD_EN[rg_mskwd_psel]=1 时，MASK_WD_POL 的源是 MASK_WD。

TC_MWD_EN[rg_mskwd_psel]默认值为 0，即 MASK_WD_POL 默认是 MASK_WD 取反后的波形。

- APTxMSKWD 的产生

  APTxMSKWD（低电平有效）的产生受屏蔽窗口使能位 TC_MWD_EN[rg_mskwd_en]控制。

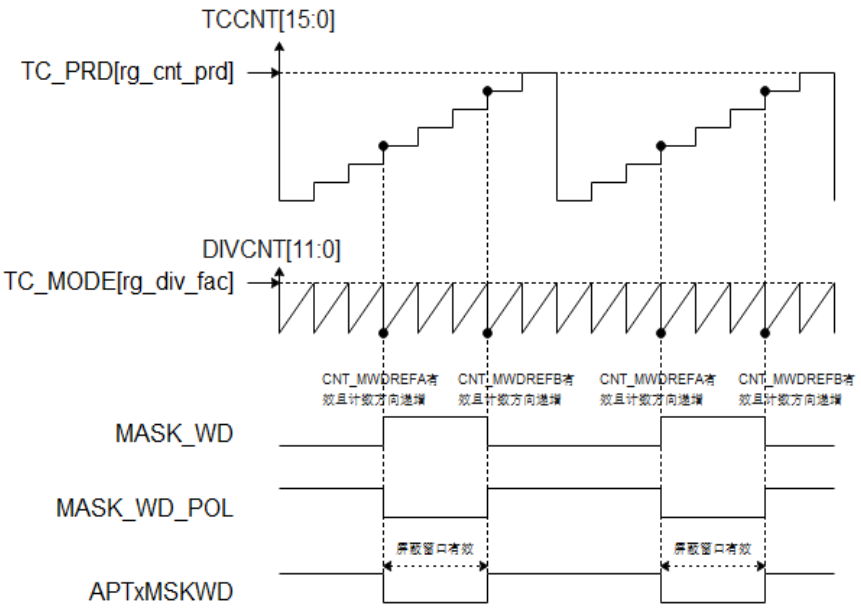
  当 TC_MWD_EN[rg_mskwd_en]=0 时，禁能屏蔽窗口，APTxMSKWD 固定为高电平。

  当 TC_MWD_EN[rg_mskwd_en]=1 时，使能屏蔽窗口，APTxMSKWD 的源是 MASK_WD_POL。

  TC_MWD_EN[rg_mskwd_en]默认值为 0，禁能屏蔽窗口，因此默认状态下 APTxMSKWD 固定为高电平。

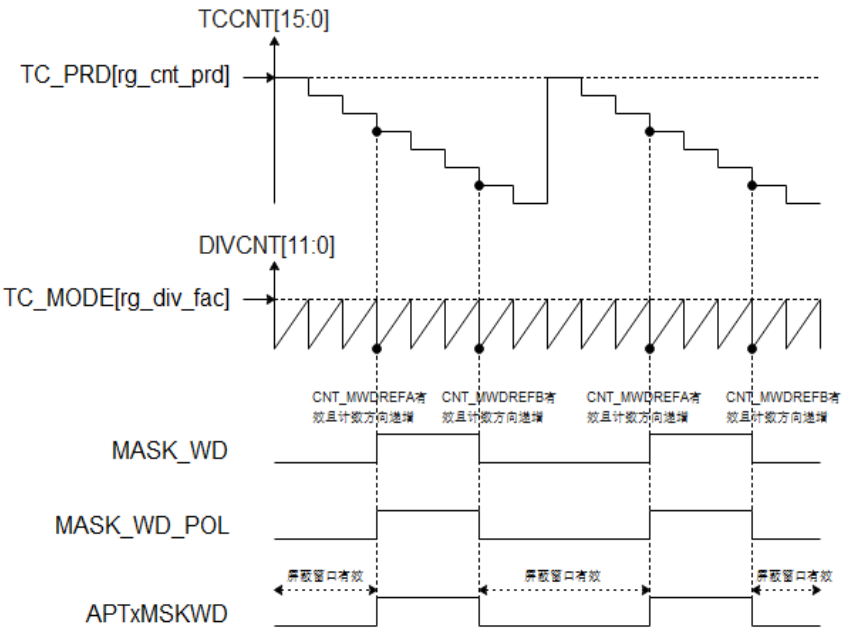
图 14-15 ~ 图 14-17 显示了增计数、减计数和先增后减计数模式下 APTxMSKWD 的产生波形。

图14-15 增计数模式下 APTxMSKWD 的产生波形



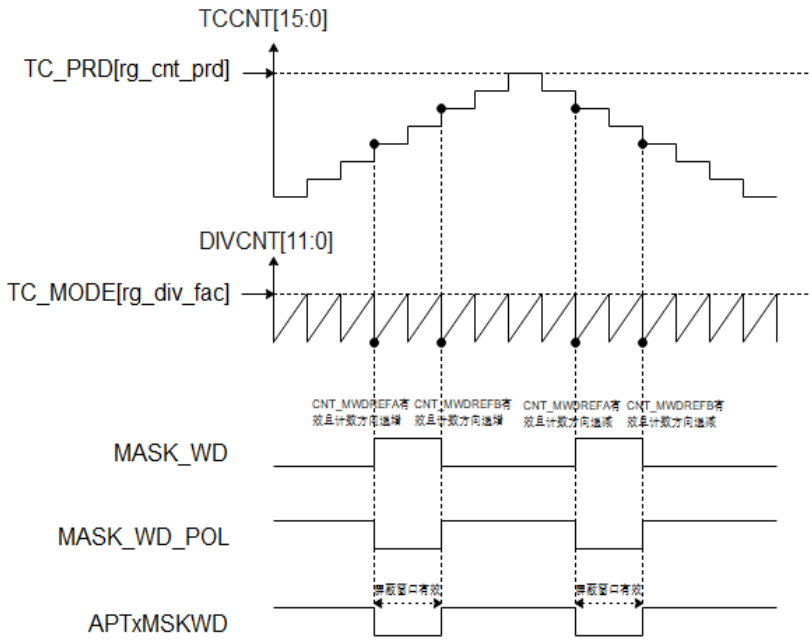
TC_MWD_ACT[rg_mwdrefa_act_inc]=10，代表当CNT_MWDREFA有效且计数方向递增时，MASK_WD置为高电平。
TC_MWD_ACT[rg_mwdrefb_act_inc]=01，代表当CNT_MWDREFB有效且计数方向递增时，MASK_WD置为低电平。
TC_MWD_EN[rg_mskwd_psel]=0，代表MASK_WD_POL的源是MASK_WD取反后的波形。

图14-16 减计数模式下 APTxMSKWD 的产生波形



TC_MWD_ACT[rg_mwdrefa_act_dec]=10，代表当CNT_MWDREFA有效且计数方向递减时，MASK_WD置为高电平。
TC_MWD_ACT[rg_mwdrefb_act_dec]=01，代表当CNT_MWDREFB有效且计数方向递减时，MASK_WD置为低电平。
TC_MWD_EN[rg_mskwd_psel]=1，代表MASK_WD_POL的源是MASK_WD。

图14-17 先增后减计数模式下 APTxMSKWD 的产生波形



TC_MWD_ACT[rg_mwdrefa_act_inc]=10，代表当CNT_MWDREFA有效且计数方向递增时，MASK_WD置为高电平。
TC_MWD_ACT[rg_mwdrefb_act_inc]=01，代表当CNT_MWDREFB有效且计数方向递增时，MASK_WD置为低电平。
TC_MWD_ACT[rg_mwdrefa_act_dec]=10，代表当CNT_MWDREFA有效且计数方向递减时，MASK_WD置为高电平。
TC_MWD_ACT[rg_mwdrefb_act_dec]=01，代表当CNT_MWDREFB有效且计数方向递减时，MASK_WD置为低电平。
TC_MWD_EN[rg_mskwd_psel]=0，代表MASK_WD_POL的源是MASK_WD取反后的波形。

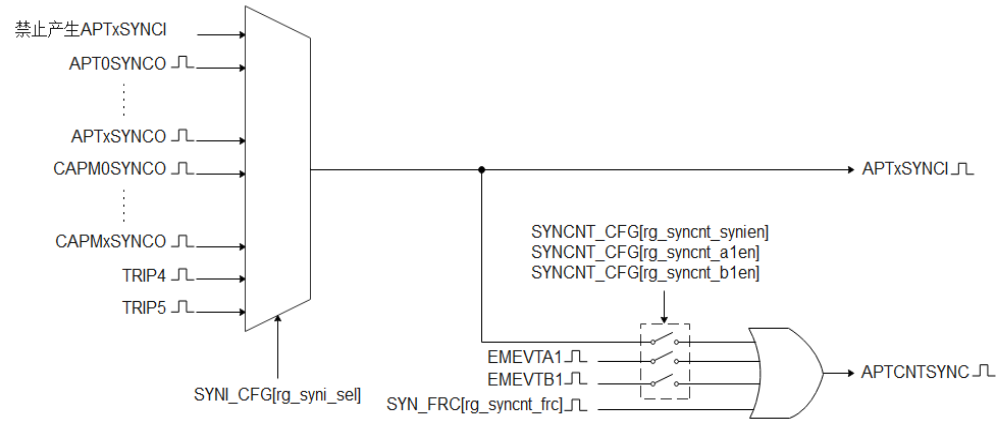
14.3.6 同步功能

同步功能可以使 APT 模块时基计数器与其他 APT 模块的时基计数器保持同步，进一步使不同 APT 模块生成的 PWM 波形保持一定的相位关系。

每个 APT 模块都有一个同步输入信号（APTxSYNCI）和一个同步输出信号（APTxSYNCO）。

14.3.6.1 同步输入

图14-18 同步输入方案



在图 14-18 中，CAPMxSYNCO 来自 CAPMx 的同步输出信号，TRIP4、TRIP5 来自事件管理子模块。

如果计数器同步信号（APTCNTSYNC）有效，则相位控制寄存器中的值加载到时基计数器中（TC_PHS[rg_cnt_phs]→TCCNT）。APTCNTSYNC 的触发源有：

- APTxSYNCI

  当 SYNCNT_CFG[rg_syncnt_synien]=1，且检测到 APTxSYNCI 信号的上升沿。

- 合并事件 EMEVTA1

  当 SYNCNT_CFG[rg_syncnt_a1en]=1，且检测到 EMEVTA1 上升沿。

- 合并事件 EMEVTB1

  当 SYNCNT_CFG[rg_syncnt_b1en]=1，且检测到 EMEVTB1 上升沿。

- 软件强制同步

  向 SYN_FRC[rg_syncnt_frc]写 1 可产生软件强制同步。

在先增后减计数模式下，APTCNTSYNC 有效后时基计数器的计数方向根据 TC_PHS[rg_cnt_dir]位的配置改变，新的计数方向与 APTCNTSYNC 有效之前的计数方向无关。在增计数或减计数模式下，TC_PHS[rg_cnt_dir]位被忽略。示例如图 14-19 ~ 图 14-22 所示。

图14-19 增计数模式同步波形

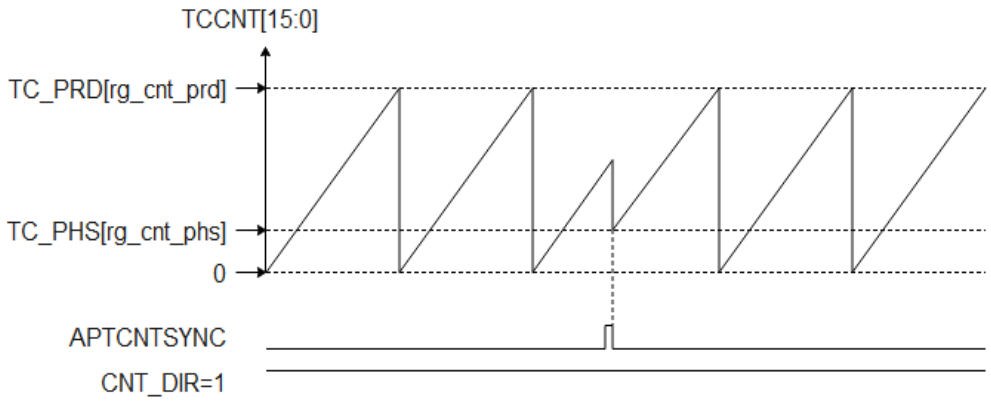


图14-20 减计数模式同步波形

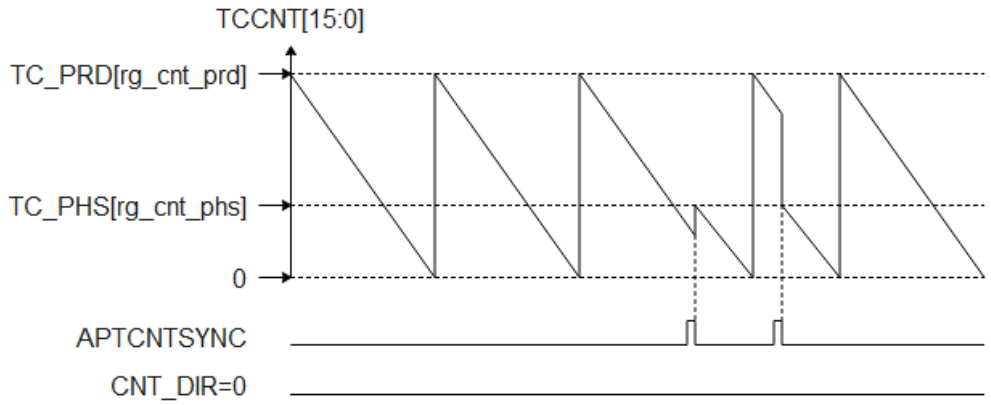


图14-21 先增后减计数模式同步波形，TC_PHS[rg_cnt_dir]=0

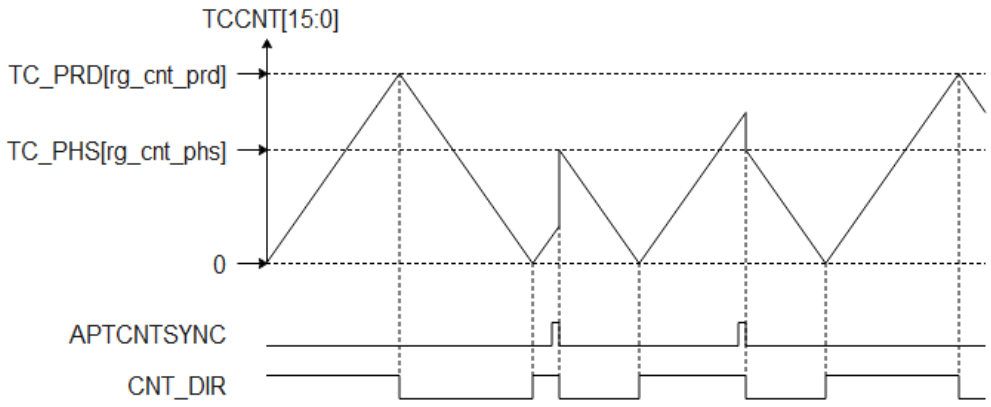
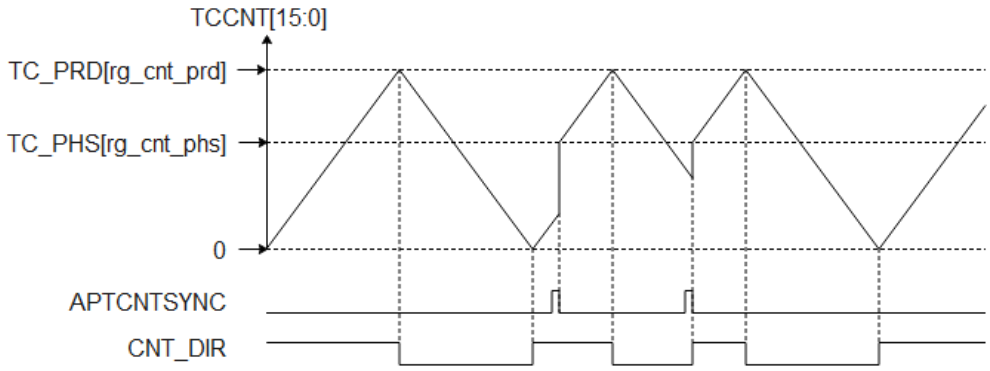


图14-22 先增后减计数模式同步波形，TC_PHS[rg_cnt_dir]=1



14.3.6.2 同步输出

图14-23 同步输出方案

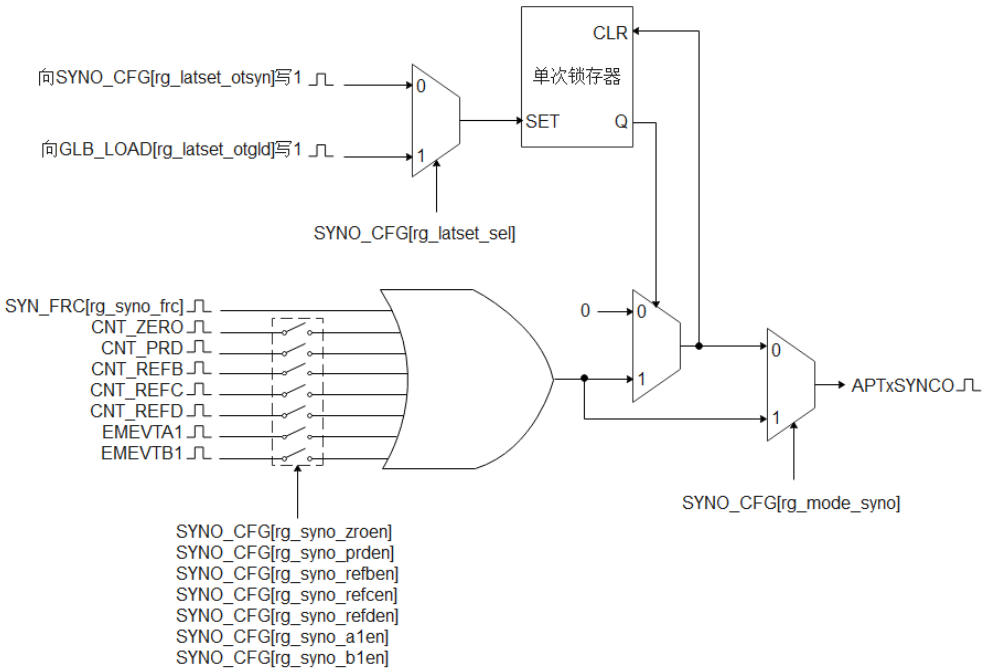


图 14-23 是 APT 模块的同步输出方案。APTxSYNCO 有以下触发源，通过 SYNO_CFG 寄存器选择：

- CNT_ZERO
- CNT_PRD
- CNT_REFB
- CNT_REFC
- CNT_REFD

- EMEVTA1
- EMEVTB1

软件可以通过向 SYN_FRC[rg_syno_frc]写 1，强制产生 APTxSYNCO。

APTxSYNCO 产生模式有两种：单次同步输出模式和不限次数同步输出模式，通过 SYNO_CFG[rg_mode_syno]选择。

- 单次同步输出模式

  软件向 SYNO_CFG[rg_latset_otsyn]或 GLB_LOAD[rg_latset_otgld]写 1 后，等待触发源的第一个脉冲，作为同步输出信号 APTxSYNCO，并且同时清除单次锁存器，保证只有一次同步输出信号有效。

- 不限次数同步输出模式

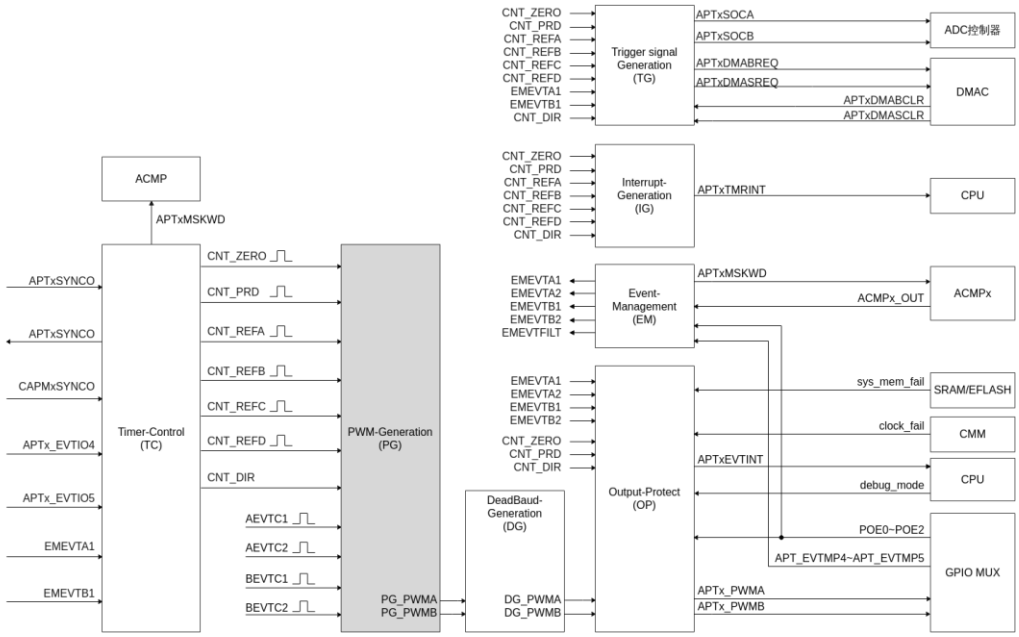
  每当触发源有效时，就会产生同步输出信号 APTxSYNCO。

14.4 波形生成子模块（PWM-Generation）

APT 系统中波形生成（PG）子模块如图 14-24 所示。

波形生成子模块在波形构造和 PWM 生成中起着最重要的作用，它决定哪些信号被转换为各种动作类型，从而在 APTx_PWMA 和 APTx_PWMB 输出上产生所需的波形。

图14-24 波形生成子模块



14.4.1 波形生成子模块的功能

波形生成子模块基于以下信号结合时基计数器的计数方向控制波形动作，并在以下信号同时有效时，管理优先级：

- CNT_ZERO：DIVCNT 等于 0，且 TCCNT 等于 0。
- CNT_PRD： DIVCNT 等于 0，且 TCCNT 等于计数周期值。
- CNT_REFA：DIVCNT 等于 0，且 TCCNT 等于计数参考值 A 寄存器 TC_REFA[rg_cnt_refa]。
- CNT_REFB：DIVCNT 等于 0，且 TCCNT 等于计数参考值 B 寄存器 TC_REFB[rg_cnt_refb]。
- CNT_REFC：DIVCNT 等于 0，且 TCCNT 等于计数参考值 C 寄存器 TC_REFC[rg_cnt_refc]。
- CNT_REFD：DIVCNT 等于 0，且 TCCNT 等于计数参考值 D 寄存器 TC_REFD[rg_cnt_refd]。
- AEVTC1、AEVTC2、BEVTC1、BEVTC2 信号：基于合并事件、POE0 ~ POE2 信号、APTxSYNCI 信号产生，由 PG_EVTC_SEL 寄存器的相关位域选择有效条件。

波形动作配置寄存器 PG_ACT_A、PG_ACT_B 和强制输出配置寄存器 PG_OUT_FRC 支持缓存加载。

14.4.2 波形生成子模块的关键信号

波形生成子模块框图如图 14-25 所示。

图14-25 波形生成子模块的输入输出

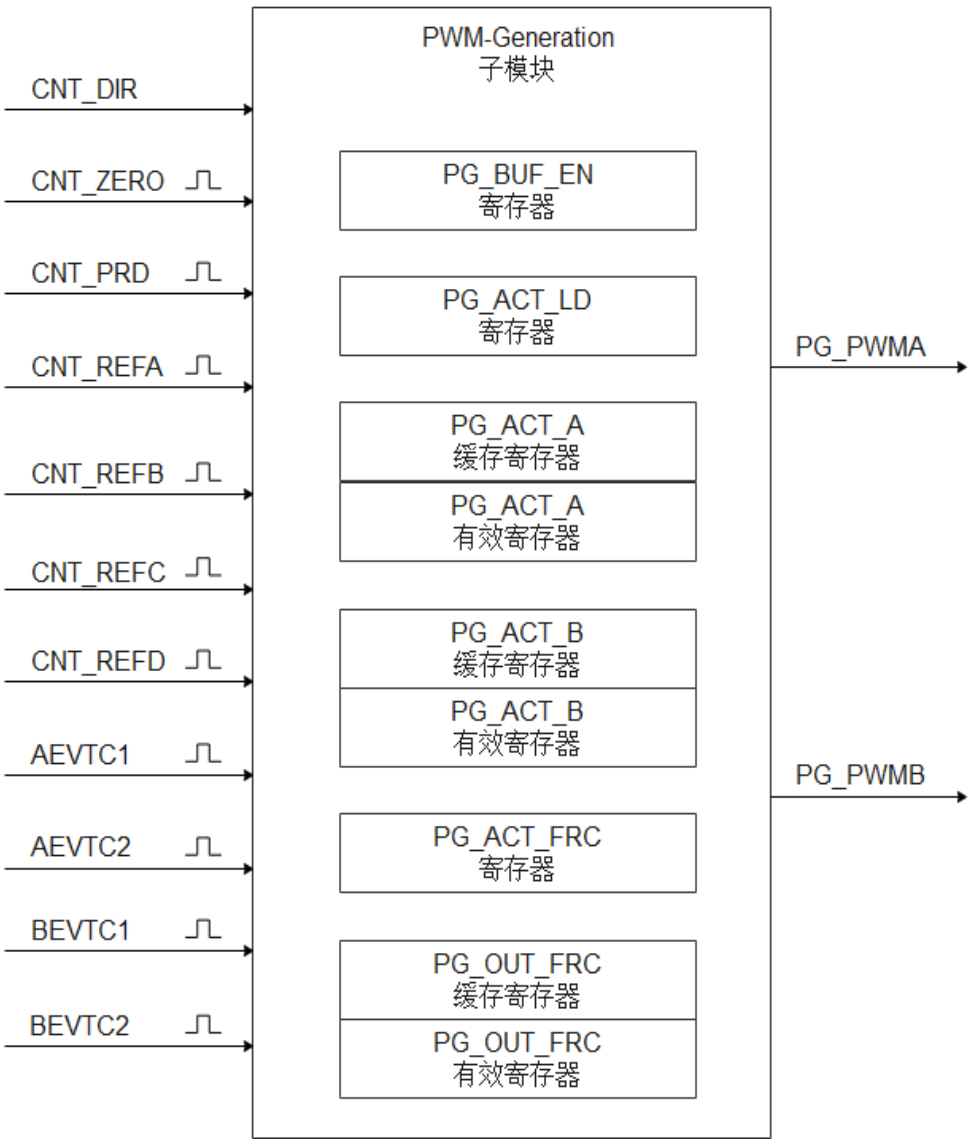


表 14-5 总结了波形生成子模块的输入信号。

表14-5 波形生成子模块可能的输入信号

信号	描述	计数器匹配条件
CNT_ZERO	分频计数器等于 0，且时基计数器等于 0。	DIVCNT=0 & TCCNT=0
CNT_PRD	分频计数器等于 0，且时基计数器等于计数周期值。	DIVCNT=0 & TCCNT=TC_PRD

信号	描述	计数器匹配条件
CNT_REFA	分频计数器等于 0，且时基计数器等于计数参考值 A 寄存器 TC_REFA[rg_cnt_refa]。	DIVCNT=0 & TCCNT=TC_REFA[rg_cnt_refa]
CNT_REFB	分频计数器等于 0，且时基计数器等于计数参考值 B 寄存器 TC_REFB[rg_cnt_refb]。	DIVCNT=0 & TCCNT=TC_REFB[rg_cnt_refb]
CNT_REFC	分频计数器等于 0，且时基计数器等于计数参考值 C 寄存器 TC_REFC[rg_cnt_refc]。	DIVCNT=0 & TCCNT=TC_REFC[rg_cnt_refc]
CNT_REFD	分频计数器等于 0，且时基计数器等于计数参考值 D 寄存器 TC_REFD[rg_cnt_refd]。	DIVCNT=0 & TCCNT=TC_REFD[rg_cnt_refd]
AEVTC1	基于合并事件、POE0~POE2 信号、APTxSYNCI 信号产生。	-
AEVTC2	基于合并事件、POE0~POE2 信号、APTxSYNCI 信号产生。	-
BEVTC1	基于合并事件、POE0~POE2 信号、APTxSYNCI 信号产生。	-
BEVTC2	基于合并事件、POE0~POE2 信号、APTxSYNCI 信号产生。	-
软件强制动作	软件发起的异步信号。	-
软件强制输出	软件发起的异步信号。	-

软件强制动作由 PG_ACT_FRC 寄存器控制，软件通过配置 PG_ACT_FRC 寄存器会在 PG_PWMA 或 PG_PWMB 上产生一次波形动作。

软件强制输出由 PG_OUT_FRC 寄存器控制，软件通过配置 PG_OUT_FRC 寄存器会对 PG_PWMA 或 PG_PWMB 产生持续影响。

14.4.3 波形生成子模块输入信号优先级

APT 波形生成子模块可能同时接收多个输入信号，在这种情况下，硬件将为输入信号分配优先级。先增后减计数模式的输入信号优先级如表 14-6 所示。优先级等级 1 是最高优先级，优先级等级 9 是最低优先级。当信号同时有效时，按照优先级高的信号配置的波形动作产生 PWM 波。

表14-6 先增后减计数模式下，对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级

优先级等级	PG_PWMA	PG_PWMB
1（最高）	软件强制输出	软件强制输出
2	软件强制动作	软件强制动作
3	AEVTC1	BEVTC1
4	AEVTC2	BEVTC2
5	CNT_REFD	CNT_REFD
6	CNT_REFC	CNT_REFC
7	CNT_REFB	CNT_REFB
8	CNT_REFA	CNT_REFA
9（最低）	CNT_ZERO(增计数阶段)/CNT_PRD(减计数阶段)	CNT_ZERO(增计数阶段)/CNT_PRD(减计数阶段)

表 14-7 显示了增计数模式下输入信号的优先级。

表14-7 增计数模式下对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级

优先级等级	PG_PWMA	PG_PWMB
1（最高）	软件强制输出	软件强制输出
2	软件强制动作	软件强制动作
3	CNT_PRD	CNT_PRD
4	AEVTC1	BEVTC1
5	AEVTC2	BEVTC2
6	CNT_REFD	CNT_REFD
7	CNT_REFC	CNT_REFC

优先级等级	PG_PWMA	PG_PWMB
8	CNT_REFB	CNT_REFB
9	CNT_REFA	CNT_REFA
10（最低）	CNT_ZERO	CNT_ZERO

表 14-8 显示了减计数模式下输入信号的优先级。

表14-8 减计数模式下对 PG_PWMA 和 PG_PWMB 产生影响的输入信号优先级

优先级等级	PG_PWMA	PG_PWMB
1（最高）	软件强制输出	软件强制输出
2	软件强制动作	软件强制动作
3	CNT_ZERO	CNT_ZERO
4	AEVTC1	BEVTC1
5	AEVTC2	BEVTC2
6	CNT_REFD	CNT_REFD
7	CNT_REFC	CNT_REFC
8	CNT_REFB	CNT_REFB
9	CNT_REFA	CNT_REFA
10（最低）	CNT_PRD	CNT_PRD

14.4.4 波形生成子模块控制波形动作

波形生成子模块的输入信号与计数方向（增计数或减计数）结合，控制两个输出 PG_PWMA 和 PG_PWMB 在特定信号有效时的动作，这允许在增计数和减计数阶段对输出执行独立的动作。

输出 PG_PWMA 和 PG_PWMB 可能的动作有：

- 置为高电平

  将输出 PG_PWMA 或 PG_PWMB 设置为高电平。
- 置为低电平

  将输出 PG_PWMA 或 PG_PWMB 设置为低电平。

- 翻转电平

  如果 PG_PWMA 或 PG_PWMB 当前为高电平，则将其设置为低电平。如果 PG_PWMA 或 PG_PWMB 当前为低电平，则将其设置为高电平。

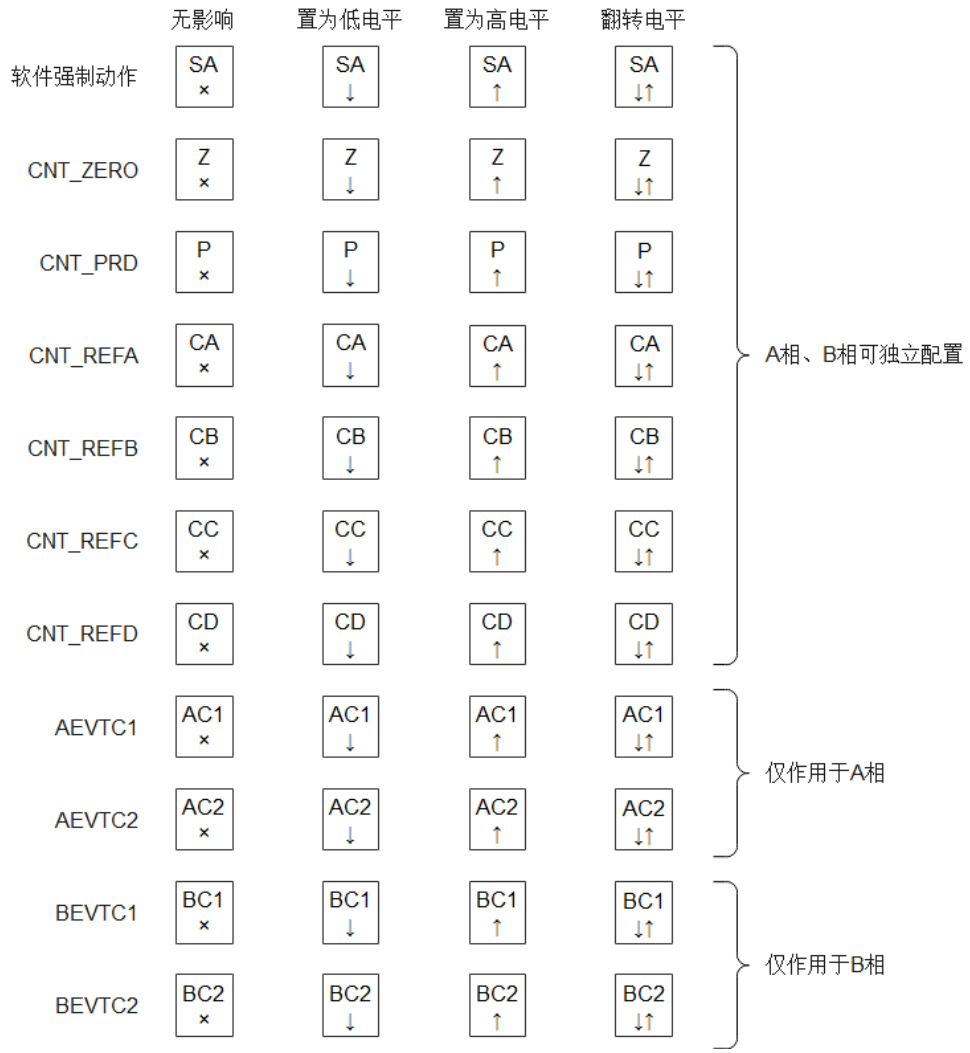
- 无影响

  输出 PG_PWMA 或 PG_PWMB 保持为当前的电平。

波形动作可以对 PG_PWMA 和 PG_PWMB 独立设置。

为清楚起见，本章的图中使用了一组符号，图 14-26 总结了这些符号。每个符号表示在一个时间点，PG_PWMA 或 PG_PWMB 的波形动作。

图14-26 输出 PG_PWMA 或 PG_PWMB 上可能的波形动作



PG_EVTC_SEL 寄存器用于选择 AEVTC1、AEVTC2、BEVTC1、BEVTC2 信号的触发源，AEVTC1 和 AEVTC2 信号只会对 PG_PWMA 产生影响，BEVTC1 和 BEVTC2 信号只会对 PG_PWMB 产生影响。其余信号对 PG_PWMA 和 PG_PWMB 均可以产生影响。

14.4.5 常用波形图

本文档中的波形显示了 APT 计数参考值寄存器值静态配置的行为。

在运行的系统中，计数参考值寄存器（TC_REFA/TC_REFB/TC_REFC/TC_REFD）和波形动作寄存器（PG_ACT_A/PG_ACT_B/PG_ACT_FRC/PG_OUT_FRC）的有效寄存器通常每周期从其各自的缓存寄存器更新一次，用户指定更新的时间，并对缓存寄存器配置新值，控制下一周期的 PWM 波形。关于缓存功能请参见 14.2 缓存加载功能介绍。

须知

基于缓存加载的机制，缓存寄存器加载到有效寄存器需要一个 APTCLK 周期，参见 14.2 缓存加载功能介绍。

在有些情况下，基于新值的动作会延期一个周期后才生效，旧值的动作会额外生效一个周期。可以使用 PWM 波形配置来规避，包括但是不限于：

- 先增后减计数模式对称 PWM 波形：
- 如果在 CNT_ZERO 加载新值，则配置增计数动作相关的计数参考值寄存器大于等于 1。
- 如果在 CNT_PRD 加载新值，则配置减计数动作相关的计数参考值寄存器小于等于 TC_PRD-1。
- 先增后减计数模式非对称 PWM 波形：
- 为了实现 50%~0%占空比的非对称 PWM 波可使用如下配置：在 CNT_PRD 加载新值，使用 CNT_PRD 拉低 PWM 波，使用一个计数参考值（如 TC_REFA）在计数方向递增时拉高 PWM 波。配置 TC_REFA 为 0~TC_PRD 可实现 50%~0%占空比。
- 增计数模式单沿非对称 PWM 波形：
- 为了实现 0%~100%非对称 PWM 波可实现如下配置：在 CNT_PRD 加载新值，使用 CNT_ZERO 拉高 PWM 波，使用一个计数参考值（如 TC_REFA）在计数方向递增时拉低 PWM 波。配置 TC_REFA 为 0~TC_PRD+1 可实现 0%~100%占空比。

图 14-27 显示了如何使用先增后减计数模式生成对称 PWM 波形。在所示的示例中，TC_REFA 用于进行比较。当计数器递增时，CNT_REFA 将把 PWM 输出拉高。同样，当计数器递减时，CNT_REFA 将把 PWM 信号拉低。

场景 1：TC_REFA=4，PWM 信号在整个周期内都为低电平，得到 0%占空比波形。

场景 2：TC_REFA=3，得到 25%占空比波形。

场景 3：TC_REFA=2，得到 50%占空比波形。

场景 4：TC_REFA=1，得到 75%占空比波形。

场景 5：TC_REFA=0，PWM 信号在整个周期内都为高电平，得到 100%占空比。

图14-27 先增后减计数模式对称 PWM 波形

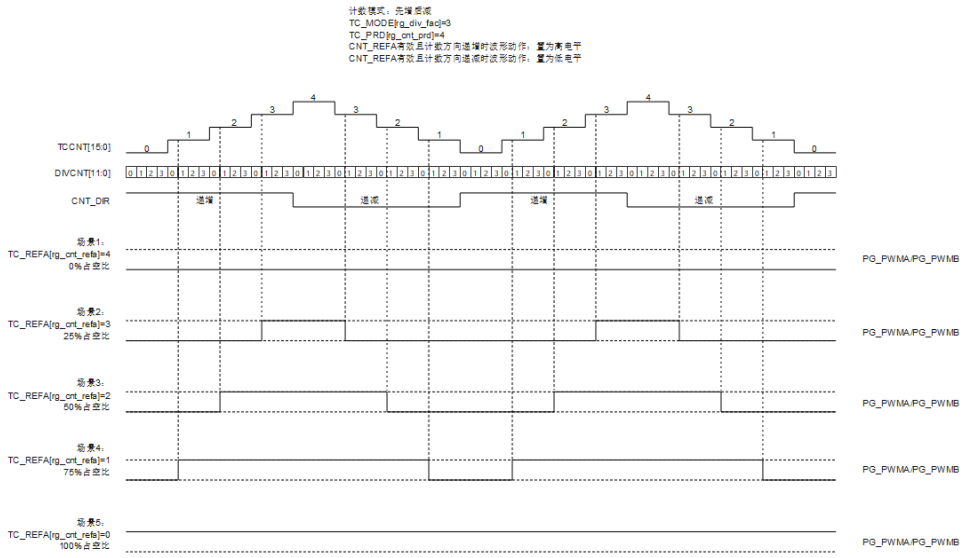


图 14-28 ~ 图 14-34 显示了一些常用的波形动作配置下的 PWM 波形。

图14-28 增计数模式，单沿非对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，高电平有效

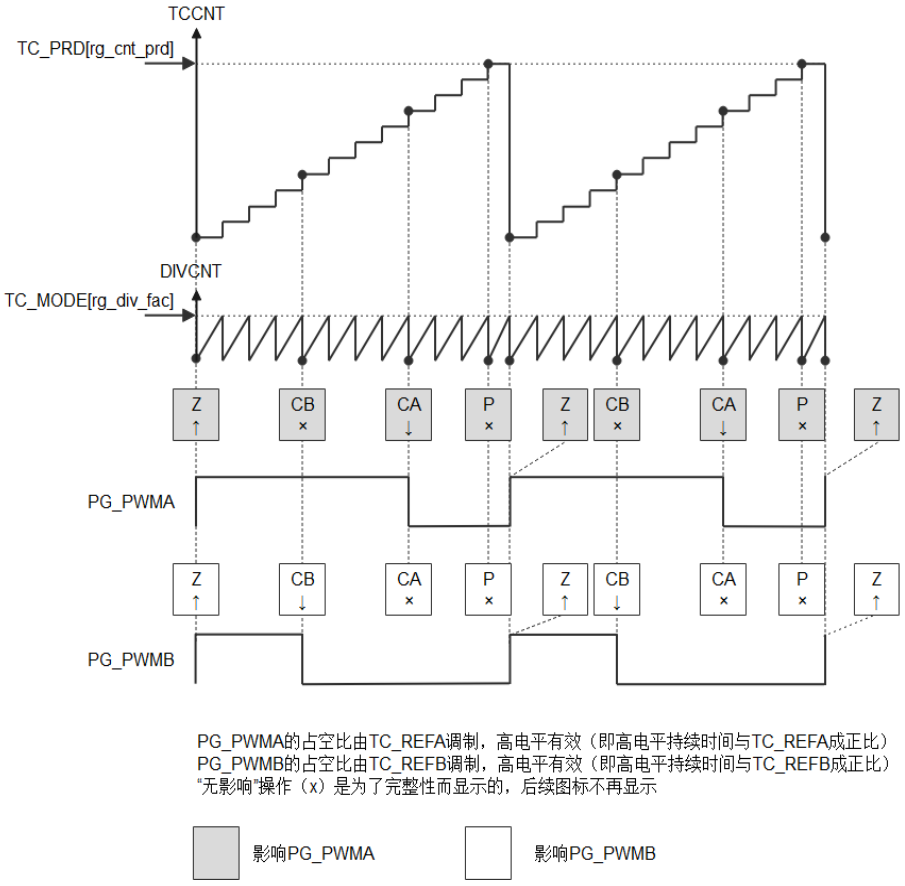


图14-29 增计数模式，单沿非对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，低电平有效

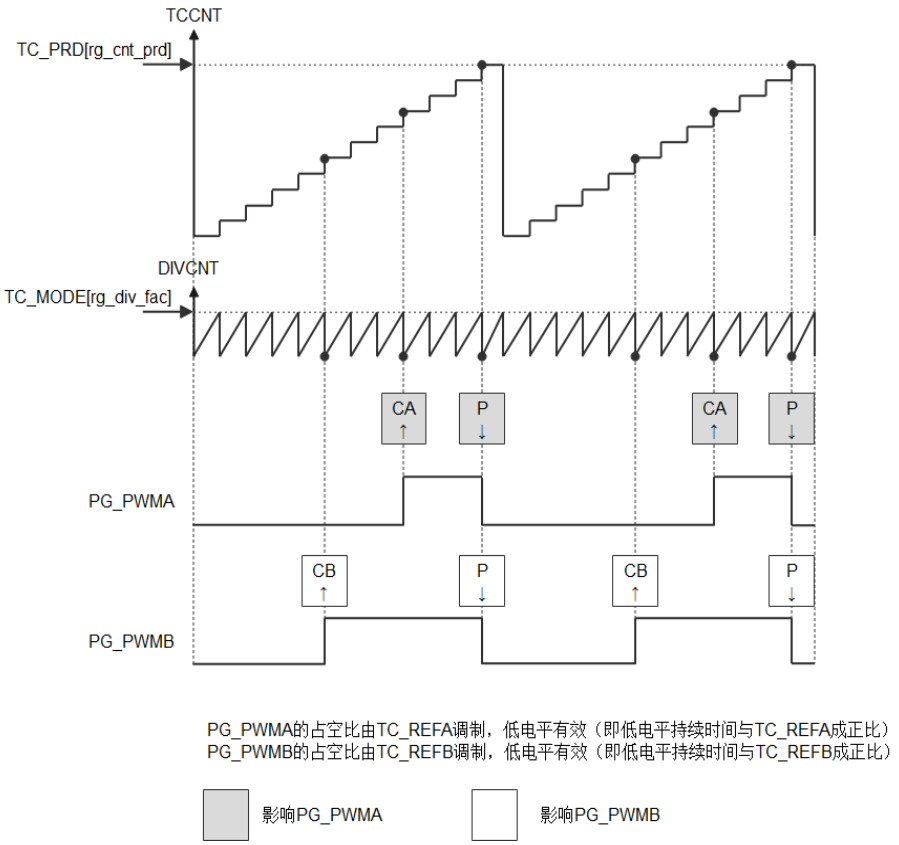


图14-30 增计数模式，PG_PWMA 输出非对称脉冲，脉冲占空比可独立控制

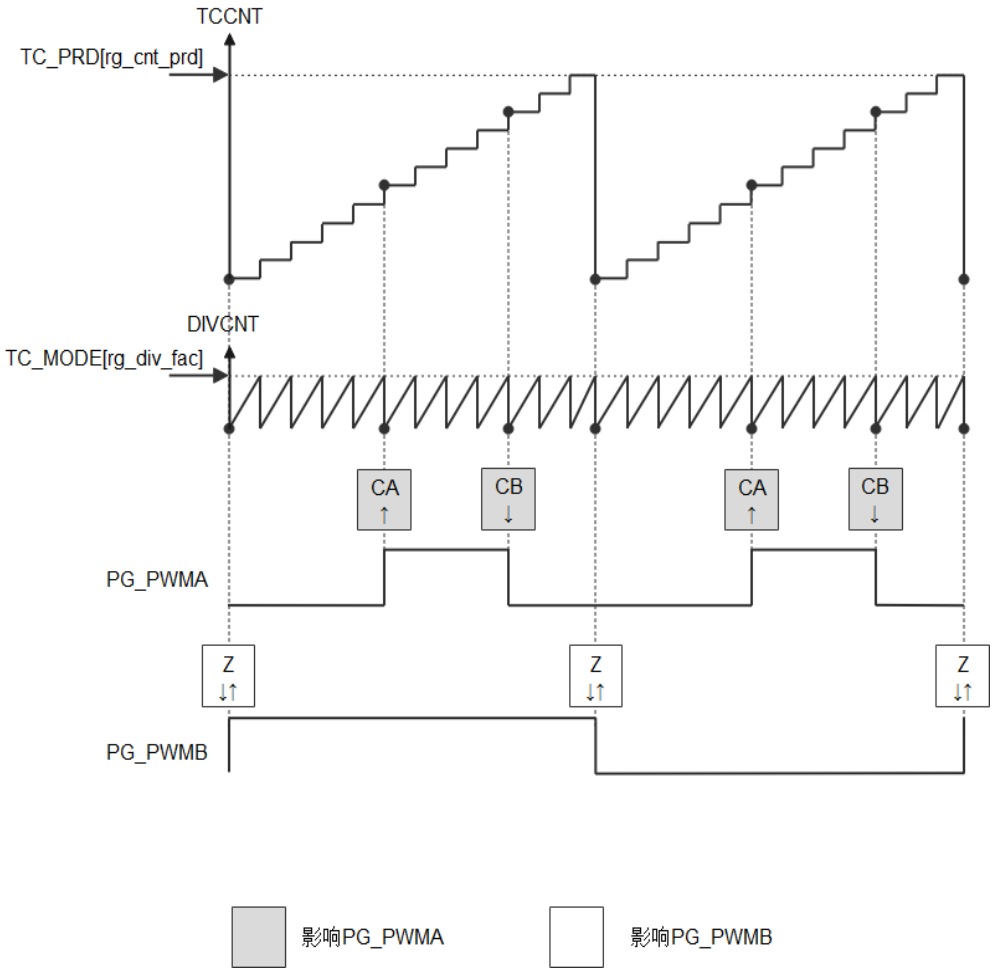


图14-31 先增后减计数模式，双沿对称 PWM 波形，PG_PWMA 和 PG_PWMB 独立控制占空比，低电平有效

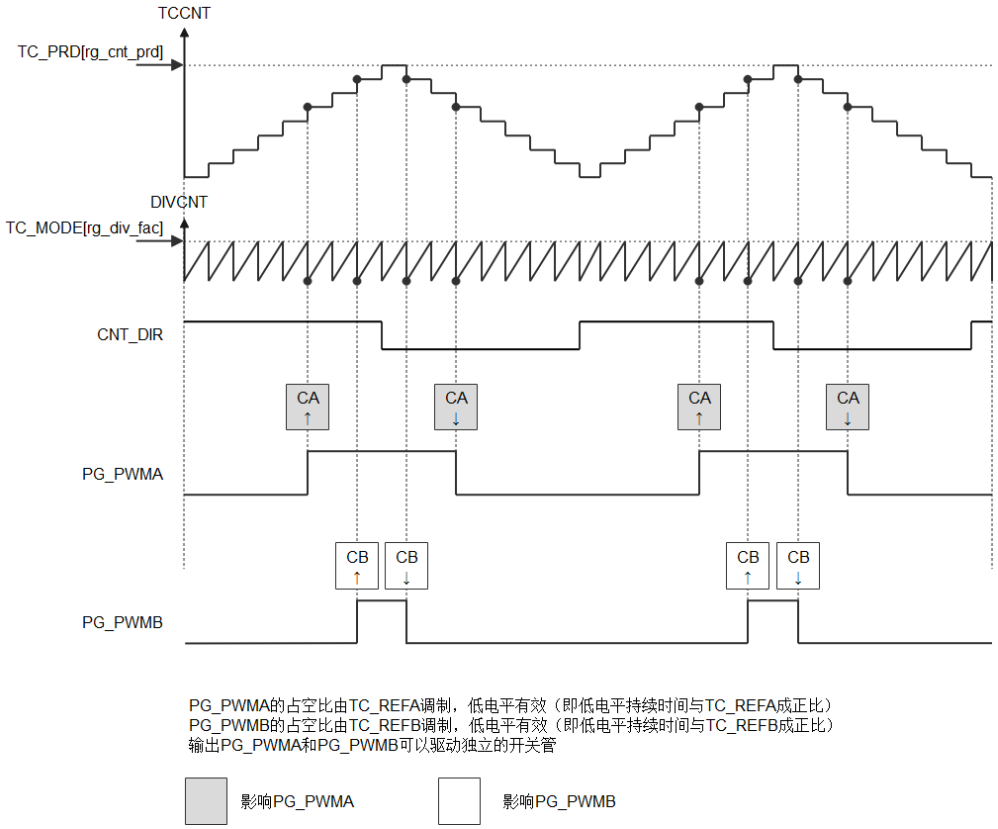


图14-32 先增后减计数模式，双沿对称 PWM 互补波形，PG_PWMA 和 PG_PWMB 独立控制占空比

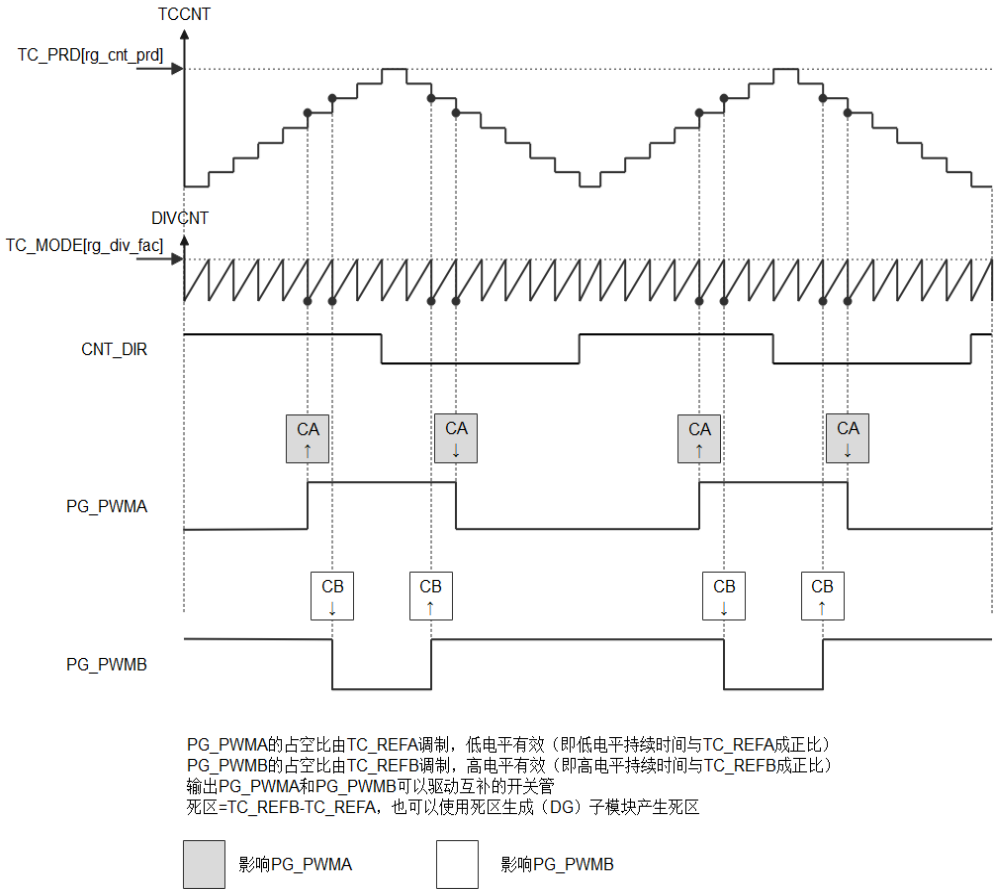


图14-33 先增后减计数模式，双沿非对称 PWM 波形，PG_PWMA 独立控制占空比，低电平有效

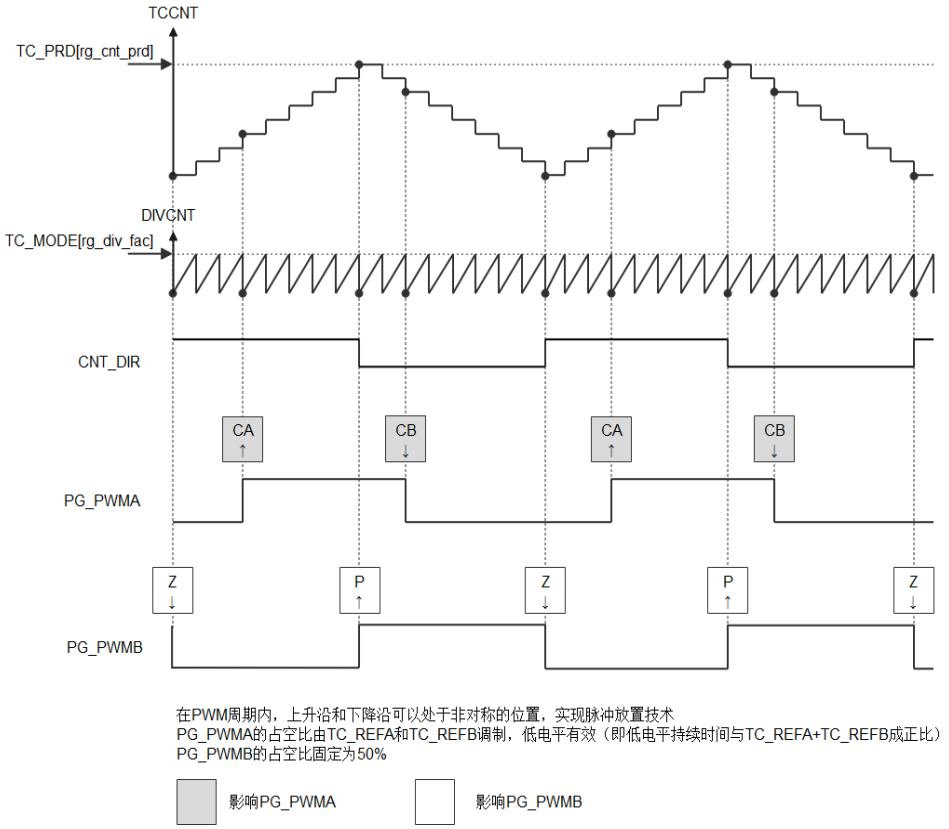
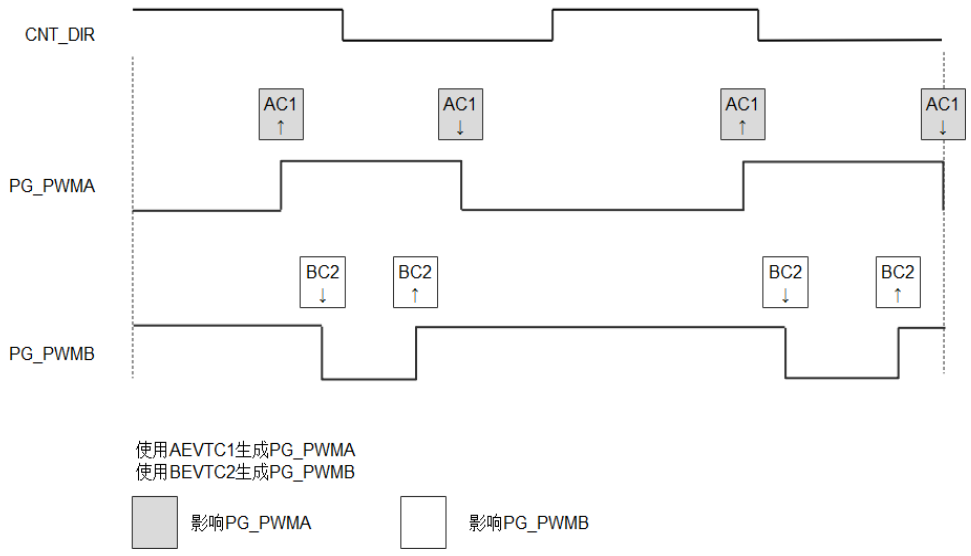


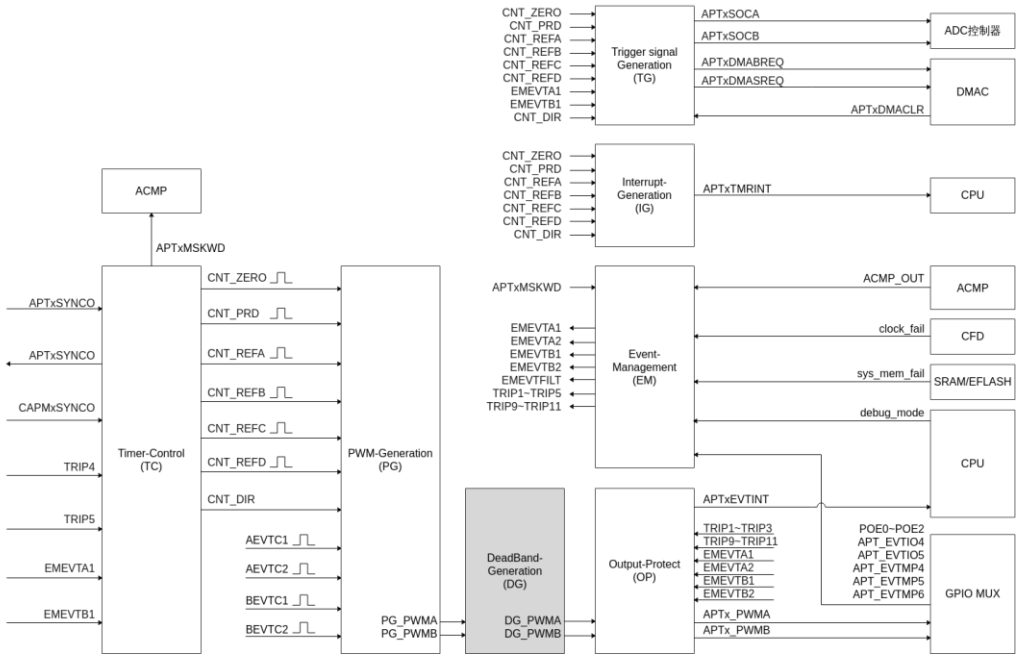
图14-34 利用 AEVTC1 和 BEVTC2 生成 PWM 波形



14.5 死区生成子模块（DeadBand-Generation）

APT 系统中死区生成子模块如图 14-35 所示。

图14-35 死区生成子模块



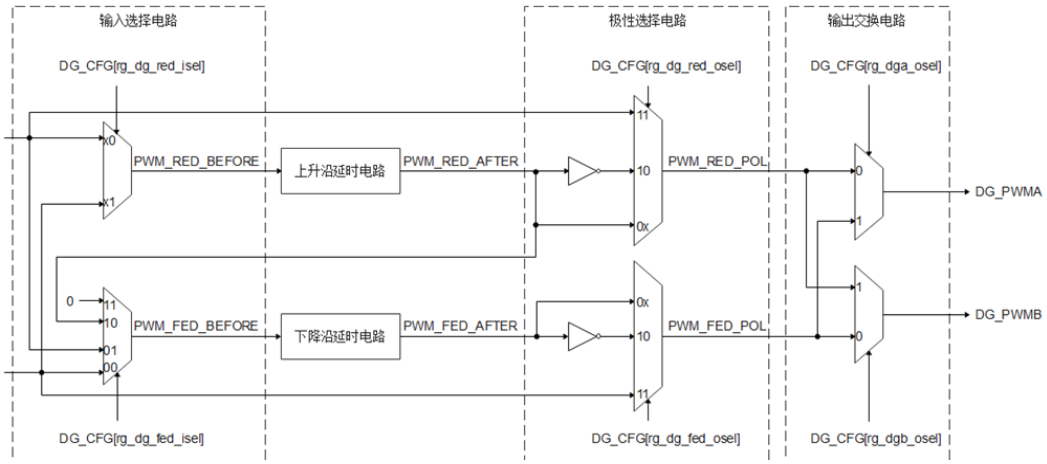
14.5.1 死区生成子模块的功能

死区生成子模块的关键功能如下：

- 基于 PG 模块的 PWM 输出（PG_PWMA 或 PG_PWMB），生成两个具有死区关系的 PWM 波（DG_PWMA 或 DG_PWMB）。
- 对上升沿插入可配置的延时（RED）。
- 对下降沿插入可配置的延时（FED）。
- 屏蔽死区生成功能。
- 死区上升沿延时寄存器 DG_RED、死区下降沿延时寄存器 DG_FED 和死区配置寄存器 DG_CFG 支持缓存加载。

  死区生成子模块的配置项如图 14-36 所示。

图14-36 死区生成子模块的配置项



- 输入选择电路

  输入选择电路可以通过 DG_CFG[rg_dg_red_isel]选择需要插入上升沿延时的波形 PWM_RED_BEFORE，也可以通过 DG_CFG[rg_dg_fed_isel]选择需要插入下降沿延时的波形 PWM_FED_BEFORE。

  PWM_RED_BEFORE 的源有 PG_PWMA 和 PG_PWMB，PWM_FED_BEFORE 的源有 PG_PWMA、PG_PWMB 和插入上升沿延时后的波形 PWM_RED_AFTER。

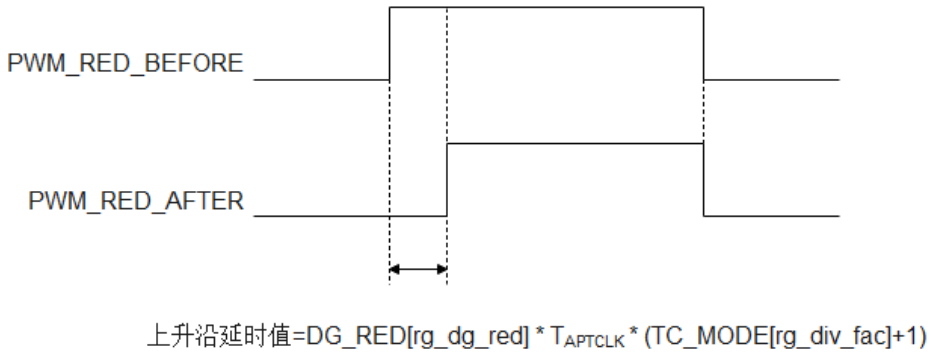
- 上升沿延时电路

  上升沿延时电路会对 PWM_RED_BEFORE 的上升沿插入延时，得到 PWM_RED_AFTER。

  上升沿延时值 = DG_RED[rg_dg_red] * T_{APTCLK} * (TC_MODE[rg_div_fac]+1)。

  图 14-37 显示了插入上升沿延时前后的波形图。

图14-37 插入上升沿延时前后的波形图



须知

如果 PWM_RED_BEFORE 的高电平时长小于上升沿延时值，则 PWM_RED_AFTER 将一直为低电平。

如果在上升沿延时过程中，DG_RED[rg_dg_red]发生了变化，不会影响本周期 PWM 的上升沿延时，会在下一个 PWM 周期生效。

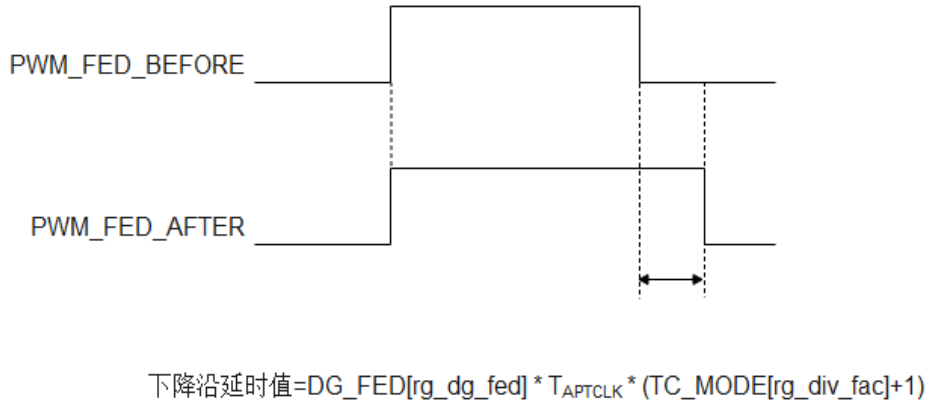
- 下降沿延时电路

  下降沿延时电路会对 PWM_FED_BEFORE 的下降沿插入延时，得到 PWM_FED_AFTER。

  下降沿延时值=DG_FED[rg_dg_fed] * T_{APTCLK} * (TC_MODE[rg_div_fac]+1)。

  图 14-38 显示了插入下降沿延时前后的波形图。

图14-38 插入下降沿延时前后的波形图



须知

如果 PWM_FED_BEFORE 的低电平时长小于下降沿延时值，则 PWM_FED_AFTER 将一直为高电平。

如果在下降沿延时过程中，DG_FED[rg_dg_fed]发生了变化，不会影响本周期 PWM 的下降沿延时，会在下一个 PWM 周期生效。

- 极性选择电路

极性选择电路可以对 PWM_RED_AFTER 和 PWM_FED_AFTER 选择极性，分别由 DG_CFG[rg_dg_red_osel]和 DG_CFG[rg_dg_fed_osel]控制，经过极性选择之后得到 PWM_RED_POL 和 PWM_FED_POL。

- 输出交换电路

  输出交换电路可以将 DG_PWMA 映射到 PWM_RED_POL 和 PWM_FED_POL，也可以将 DG_PWMB 映射到 PWM_RED_POL 和 PWM_FED_POL，通过 DG_CFG[rg_dga_osel]和 DG_CFG[rg_dgb_osel]选择。

14.5.2 死区生成子模块的典型配置

表 14-9 列举了一些典型的死区配置，这些配置假设 PG_PWMA 是上升沿延时电路和下降沿延时电路的源（DG_CFG[rg_dg_red_isel]=x0，DG_CFG[rg_dg_fed_isel]=01），输出不进行交换（DG_CFG[rg_dga_osel]=0，DG_CFG[rg_dgb_osel]=0）。

表14-9 典型的死区配置

模式	模式描述	DG_CFG[rg_dg_red_osel]	DG_CFG[rg_dg_fed_osel]
1	PG_PWMA 和 PG_PWMB 直通（没有任何延时）	11	11
2	高电平有效互补	0x	10
3	低电平有效互补	10	0x
4	高电平有效	0x	0x
5	低电平有效	10	10

表 14-9 列举的各种配置可以分为以下几类：

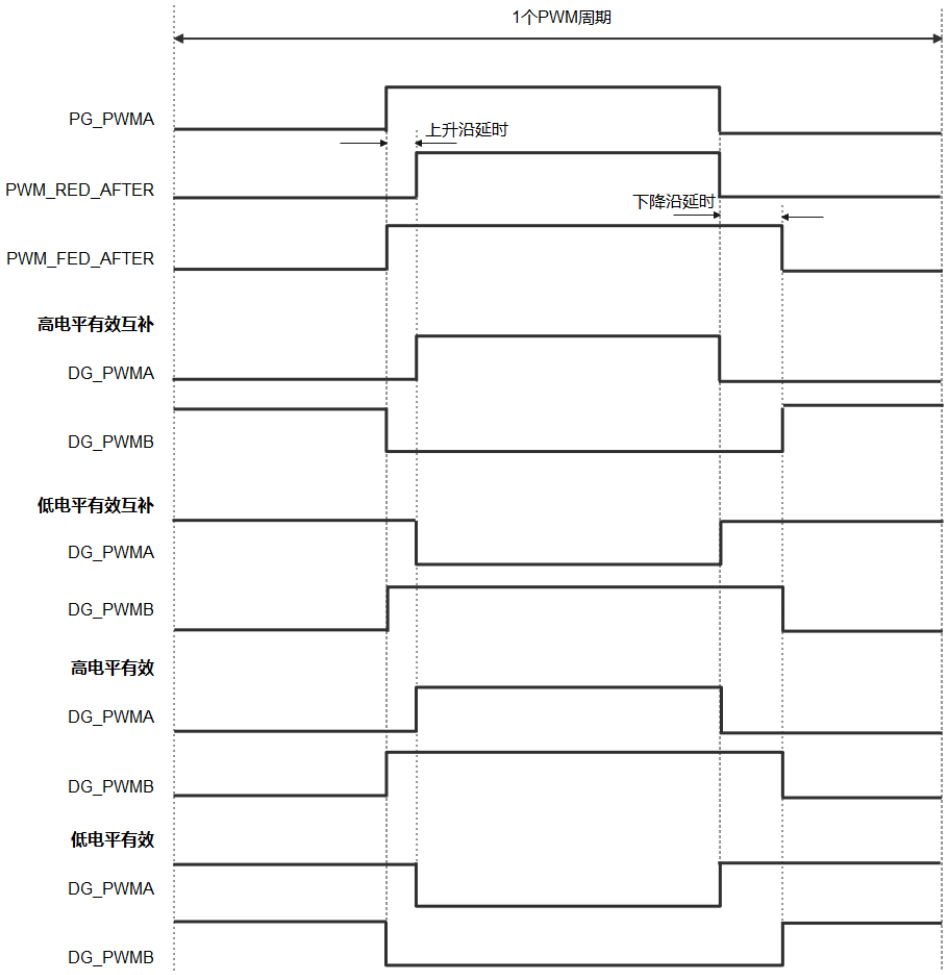
- 模式 1：屏蔽上升沿延时电路和下降沿延时电路。

  该配置可以不对 PG_PWMA 和 PG_PWMB 插入任何延时。

- 模式 2 ~ 5：典型死区配置。

  这些典型的死区配置，可以满足工业开关管所需的所有死区模式（高电平导通或低电平导通）。这些典型死区配置的波形如图 14-39 所示。

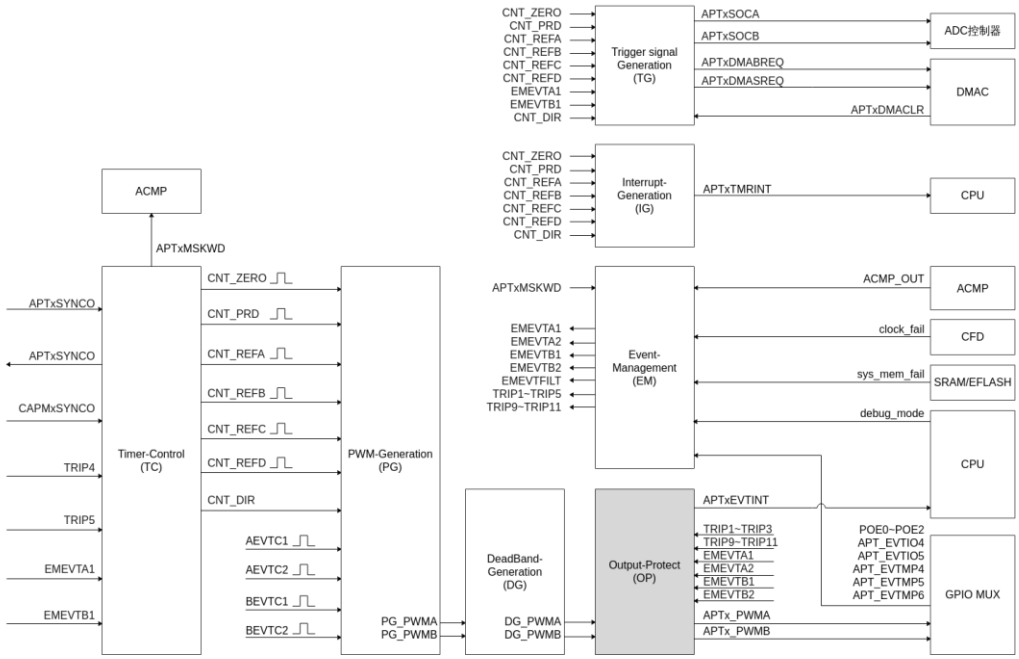
图14-39 典型死区配置的波形图



14.6 输出保护子模块（Output-Protect）

APT 系统中输出保护子模块如图 14-40 所示。

图14-40 输出保护子模块



每个 APT 模块有 10 个故障信号，来自事件管理子模块的输出：TRIP1 ~ TRIP3、TRIP9~TRIP11、4 个合并事件 EMEVTA1、EMEVTA2、EMEVTB1、EMEVTB2。这些信号反映了外部的故障，高电平有效，APT 可配置为在故障信号有效时做出相应的保护动作。

14.6.1 输出保护子模块的功能

输出保护子模块的主要功能是：

- 在故障信号有效时，将输出 APTx_PWMA 和 APTx_PWMB 强制为以下动作：
  - 不做任何动作
  - 低电平
  - 高电平
  - 锁定电平
  - 翻转电平
  - 高阻态
- 支持故障信号有效时做持续性保护。
- 支持故障信号有效时做周期性保护。
- 每个故障信号都可以触发事件中断 APTxEVTINT。

- 支持软件强制输出保护。
- 如果不需要，可以完全屏蔽输出保护子模块。

14.6.2 输出保护子模块的典型操作

当故障信号中的一个有效时，表示发生了故障，APT 模块可以配置为响应或忽略故障信号，由 OC_MODE 寄存器的相关位域选择。时钟丢失时，故障信号也可以对输出进行保护。

每个故障信号进行单独配置为锁存或不锁存，由 OC_LAT_EN 寄存器配置。每个故障信号还可以单独配置为周期性保护或持续性保护，由 OC_MODE 寄存器的相关位域选择。

故障信号对 APT 模块产生的影响有 3 个：产生故障信号标志、产生事件中断标志并上报事件中断、对 APTx_PWMA 和 APTx_PWMB 进行保护，以下内容将分别介绍故障信号锁存、非锁存、周期性保护、持续性保护配置下对 APT 模块的 3 个影响。

- 产生故障信号标志

  故障信号有效时，OC_EVT_FLAG 寄存器中对应故障信号的标志会置位。软件向寄存器的相关位域写 1 可以清除对应故障信号的标志。

- 产生事件中断标志并上报事件中断

  故障信号有效时，如果通过 INT_EVT_EN 相关位域使能了对应故障信号的中断时，则会将事件中断标志（OC_EVT_FLAG[ro_int_flag_evt]）置位，并上报事件中断（APTxEVTINT）。清除中断时，需要先清除故障信号的标志，再向 OC_EVT_FLAG[rg_int_clr_evt]写 1 清除事件中断。

- 对 APTx_PWMA 和 APTx_PWMB 进行保护
  - 当选择不锁存时

    故障信号有效时，OC_ACT_A 和 OC_ACT_B 寄存器中指定的保护动作将立即在 APTx_PWMA 和 APTx_PWMB 上生效，表 14-10 列举了所有可能的保护动作。故障信号不再有效时，APTx_PWMA 和 APTx_PWMB 立即恢复正常输出。

  - 当选择锁存时

    故障信号有效时，OC_ACT_A 和 OC_ACT_B 寄存器中指定的保护动作将立即在 APTx_PWMA 和 APTx_PWMB 上生效，表 14-10 列举了所有可能的保护动作。故障信号不再有效时，APTx_PWMA 和 APTx_PWMB 不会立即恢复正常输出，而是根据选择的周期性保护或持续性保护，来选择恢复正常输出的时刻。

周期性保护：如果故障信号不再有效，则在 CNT_ZERO 或 CNT_PRD 信号有效时，APTx_PWMA 和 APTx_PWMB 恢复正常输出，由 OC_PRD_CLR 寄存器配置。因此，在此模式下每个 PWM 周期都会自动清除对 APTx_PWMA 和 APTx_PWMB 的保护。

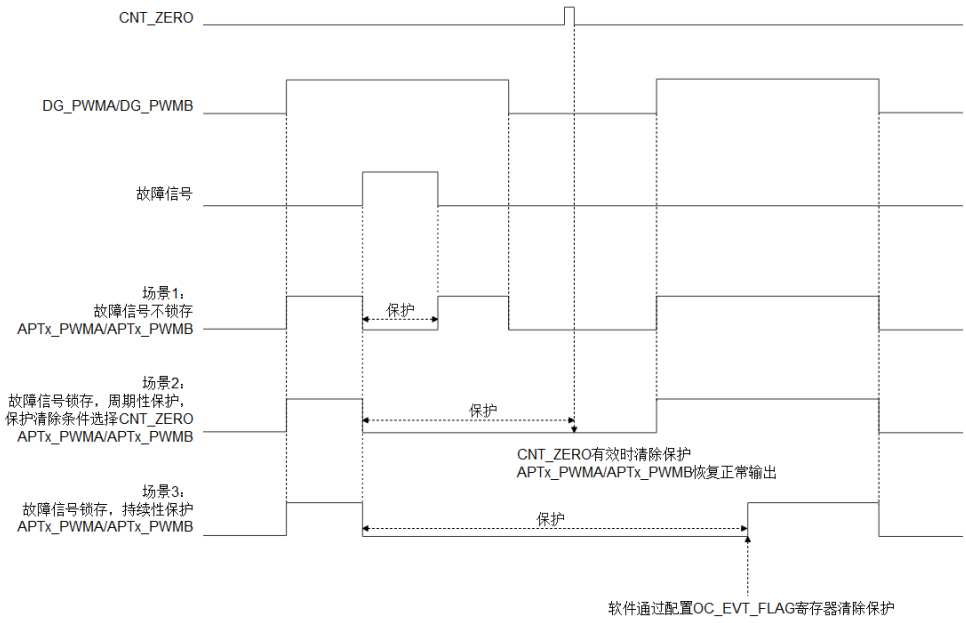
持续性保护：必须通过软件向 OC_EVT_FLAG 寄存器的相关位域写 1 才能清除对 APTx_PWMA 和 APTx_PWMB 的保护。

表14-10 所有可能的保护动作

OC_ACT_A 和 OC_ACT_B 位域配置	APTx_PWMA 和 APTx_PWMB 保护动作
000	不做任何动作
001	低电平
010	高电平
011	锁定电平
100	翻转电平
其他	高阻态

图 14-41 显示了故障信号不锁存、故障信号锁存且周期性保护、故障信号锁存且持续性保护时，APTx_PWMA 和 APTx_PWMB 的保护动作，保护动作选择“低电平”。

图14-41 不同保护模式下，APTx_PWMA 和 APTx_PWMB 的保护动作



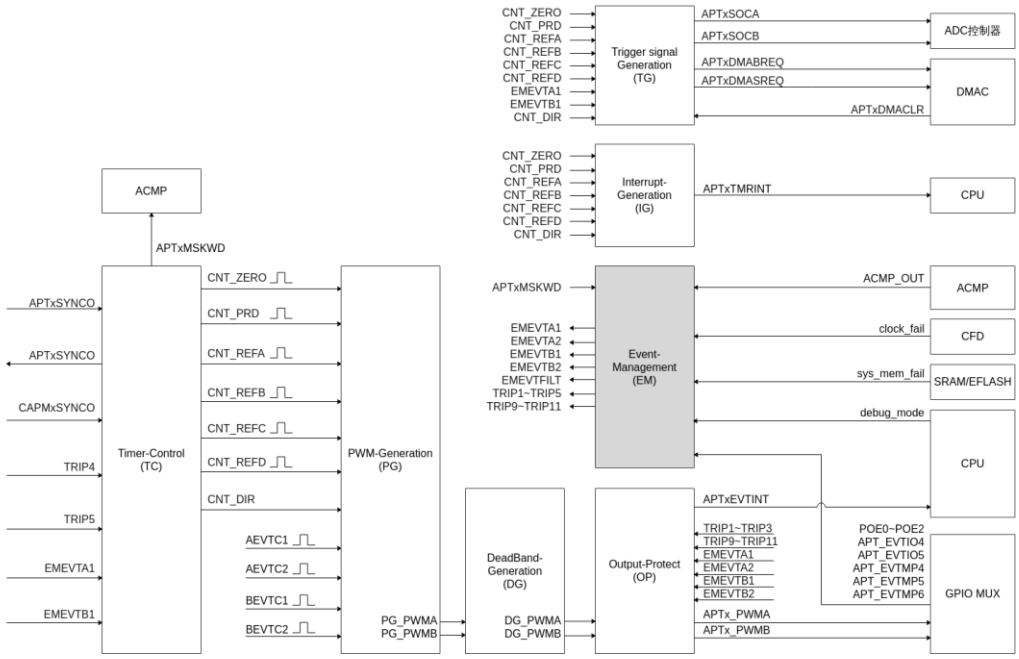
配置举例：TRIP1 有效时，对 APTx_PWMA 进行周期性保护，保护动作为“高电平”，在 CNT_ZERO 时恢复 APTx_PWMA 正常输出。EMEVTA1 有效时，对 APTx_PWMB 进行持续性保护，保护动作为“高阻态”。

- 配置 OC_MODE[rg_oc_en_evt1]=1：使能 TRIP1 信号的保护。
- 配置 OC_MODE[rg_oc_en_evta1]=1：使能 EMEVTA1 信号的保护。
- 配置 OC_LAT_EN[rg_oc_laten_evt1]=1：使能 TRIP1 信号的锁存。
- 配置 OC_MODE[rg_oc_mode_evt1]=1：使能 TRIP1 信号的周期性保护。
- 配置 OC_PRD_CLR[rg_oc_clr_zroen_evt1]=1：配置 TRIP1 信号为在 CNT_ZERO 信号有效时清除保护。
- 配置 OC_ACT_A[rg_oca_evtio_inc]和 OC_ACT_A[rg_oca_evtio_dec]=010，选择保护动作为“高电平”。
- 配置 OC_LAT_EN[rg_oc_laten_evta1]=1：使能 EMEVTA1 信号的锁存。
- 配置 OC_MODE[rg_oc_mode_evta1]=0：使能 EMEVTA1 信号的持续性保护。
- 配置 OC_ACT_B[rg_ocb_evta1_inc]和 OC_ACT_B[rg_ocb_evta1_dec]=111，选择保护动作为“高阻态”。

14.7 事件管理子模块（Event-Management）

图 14-42 显示了事件管理子模块与 APT 其他子模块的互连。

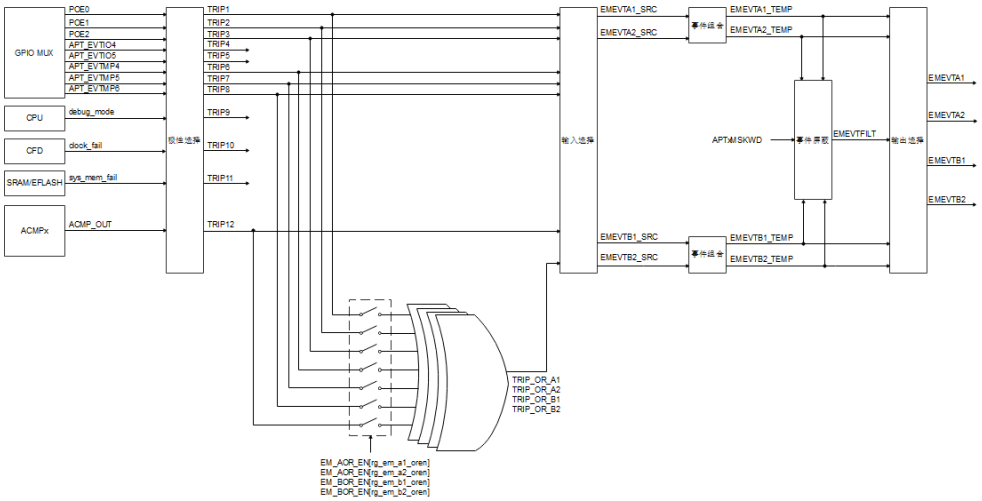
图14-42 事件管理子模块



事件管理子模块对 APT 模块的外部信号（POE0~POE2、ACMP_OUT、APT_EVTMP4 ~ APT_EVTMP6）进行组合，产生合并事件和滤波事件，输出到计数控制子模块、波形生成子模块、输出保护子模块以及触发信号生成子模块。此外，事件管理子模块可以使用计数控制子模块产生的屏蔽窗口 APTxMSKWD，用来屏蔽 POE0 ~ POE2、ACMP_OUT、APT_EVTMP4~APT_EVTMP6 信号中的噪声。

14.7.1 事件管理子模块的功能

图14-43 事件管理子模块功能框图



事件管理子模块的主要功能包括：

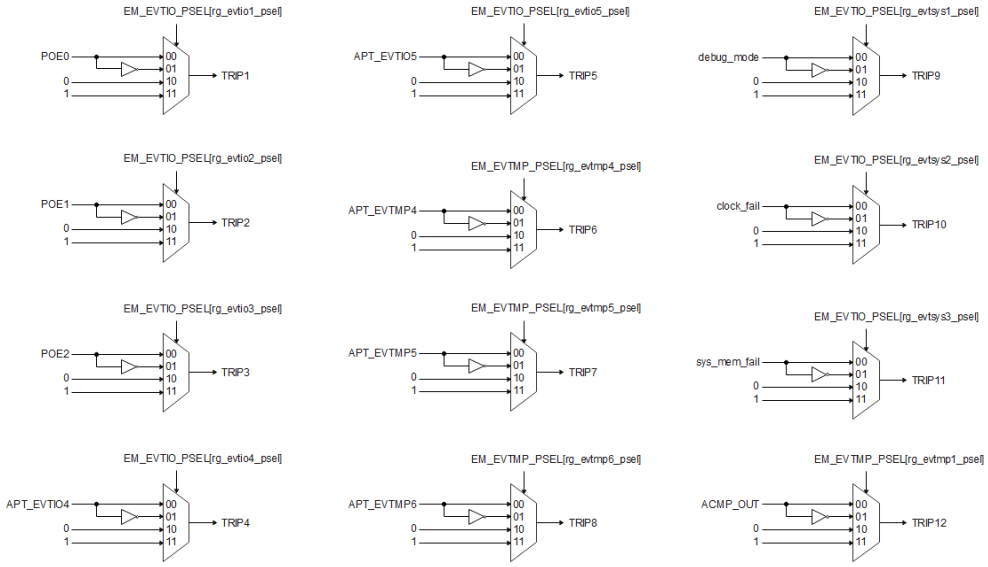
- 对 POE0 ~ POE2、ACMP_OUT、APT_EVTMP4 ~ APT_EVTMP6 信号进行组合，得到合并事件 EMEVTA1、EMEVTA2、EMEVTB1、EMEVTB2，以及滤波事件 EMEVTFILT。
- 合并事件和滤波事件可以输出到计数控制子模块、波形生成子模块、输出保护子模块以及触发信号生成子模块：
  - 输出到计数控制子模块，用来生成同步信号 APTxSYNCI 和 APTCNTSYNC；
  - 输出到波形生成子模块，用来生成 AEVTC1、AEVTC2、BEVTC1、BEVTC2 信号，进一步控制 PG_PWMA 和 PG_PWMB 的波形生成；
  - 输出到输出保护子模块，用作故障信号，触发 APTx_PWMA 和 APTx_PWMB 的保护动作，产生事件中断；
  - 输出到触发信号生成子模块，用来控制 APTxSOCA、APTxSOCB、APTxDMABREQ、APTxDMASREQ 信号的产生。
- 事件屏蔽功能可以用来屏蔽 POE0 ~ POE2、ACMP_OUT、APT_EVTMP4 ~ APT_EVTMP6 信号中的噪声。

14.7.2 极性选择

极性选择电路可以通过 EM_EVTIO_PSEL 和 EM_EVTMP_PSEL 寄存器对 POE0 ~ POE2、APT_EVTIO4 ~ APT_EVTIO5、APT_EVTMP4 ~ APT_EVTMP6、

debug_mode、clock_fail、sys_mem_fail、ACMP_OUT 选择极性，得到 TRIP1~TRIP12 信号，TRIP1~TRIP12 高电平有效，如图 14-44 所示。

图14-44 TRIP1~TRIP12 信号的产生



14.7.3 TRIP_OR 信号的产生

为了允许一次有多个信号可以影响合并事件（EMEVTA1、EMEVTA2、EMEVTB1、EMEVTB2）的产生，有 4 个或逻辑将 TRIP1 ~ TRIP3、TRIP6 ~ TRIP8、TRIP12 汇聚在一起作为 EMEVTA1_SRC、EMEVTA2_SRC、EMEVTB1_SRC、EMEVTB2_SRC 的“组合输入”（TRIP_OR_A1、TRIP_OR_A2、TRIP_OR_B1、TRIP_OR_B2），通过 EM_AOR_EN 和 EM_BOR_EN 寄存器选择。其中：

- TRIP_OR_A1 是 EMEVTA1_SRC 的“组合输入”。
- TRIP_OR_A2 是 EMEVTA2_SRC 的“组合输入”。
- TRIP_OR_B1 是 EMEVTB1_SRC 的“组合输入”。
- TRIP_OR_B2 是 EMEVTB2_SRC 的“组合输入”。

14.7.4 输入选择

输入选择电路产生 EMEVTA1_SRC、EMEVTA2_SRC、EMEVTB1_SRC、EMEVTB2_SRC，功能如下：

- 通过 EM_MRG_SEL[rg_em_a1_sel]寄存器从 TRIP1 ~ TRIP3、TRIP6 ~ TRIP8、TRIP12、TRIP_OR_A1 中选择一个作为 EMEVTA1_SRC。

- 通过 EM_MRG_SEL[rg_em_a2_sel]寄存器从 TRIP1 ~ TRIP3、TRIP6 ~ TRIP8、TRIP12、TRIP_OR_A2 中选择一个作为 EMEVTA2_SRC。
- 通过 EM_MRG_SEL[rg_em_b1_sel]寄存器从 TRIP1 ~ TRIP3、TRIP6 ~ TRIP8、TRIP12、TRIP_OR_B1 中选择一个作为 EMEVTB1_SRC。
- 通过 EM_MRG_SEL[rg_em_b2_sel]寄存器从 TRIP1 ~ TRIP3、TRIP6 ~ TRIP8、TRIP12、TRIP_OR_B2 中选择一个作为 EMEVTB2_SRC。

14.7.5 事件组合

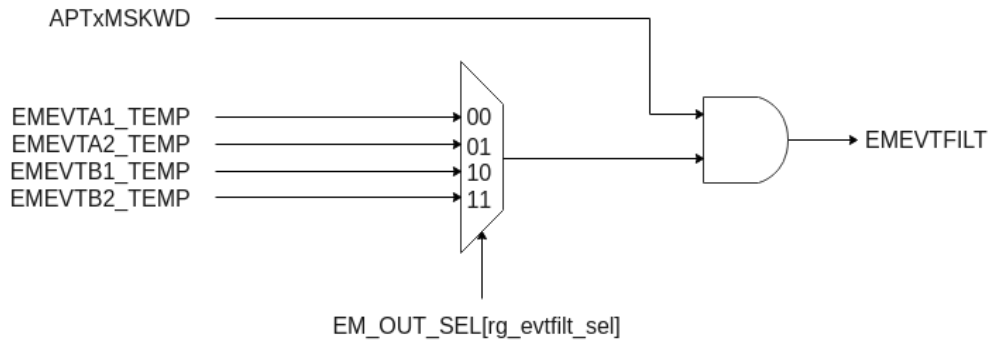
事件组合电路产生 EMEVTA1_TEMP、EMEVTA2_TEMP、EMEVTB1_TEMP、EMEVTB2_TEMP，用户通过：

- EM_MRG_SEL[rg_evta1t_sel]选择 EMEVTA1_TEMP 有效的条件。
- EM_MRG_SEL[rg_evta2t_sel]选择 EMEVTA2_TEMP 有效的条件。
- EM_MRG_SEL[rg_evtb1t_sel]选择 EMEVTB1_TEMP 有效的条件。
- EM_MRG_SEL[rg_evtb2t_sel]选择 EMEVTB2_TEMP 有效的条件。

14.7.6 事件屏蔽

用户可以通过事件屏蔽功能，屏蔽 EMEVTA1_TEMP、EMEVTA2_TEMP、EMEVTB1_TEMP 或 EMEVTB2_TEMP 的噪声，防止误触发事件中断、输出保护以及 SOC 信号。事件屏蔽功能的详细信息如图 14-45 所示。

图14-45 事件屏蔽功能



软件通过 EM_OUT_SEL[rg_evtfilt_sel]寄存器从 EMEVTA1_TEMP、EMEVTA2_TEMP、EMEVTB1_TEMP、EMEVTB2_TEMP 中选择一个进行事件屏蔽，得到滤波事件 EMEVTFILT。

屏蔽窗口 APTxMSKWD 低电平有效，不使能屏蔽窗口功能时（TC_MWD_EN[rg_mskwd_en]=0）默认为高电平。

14.7.7 输出选择

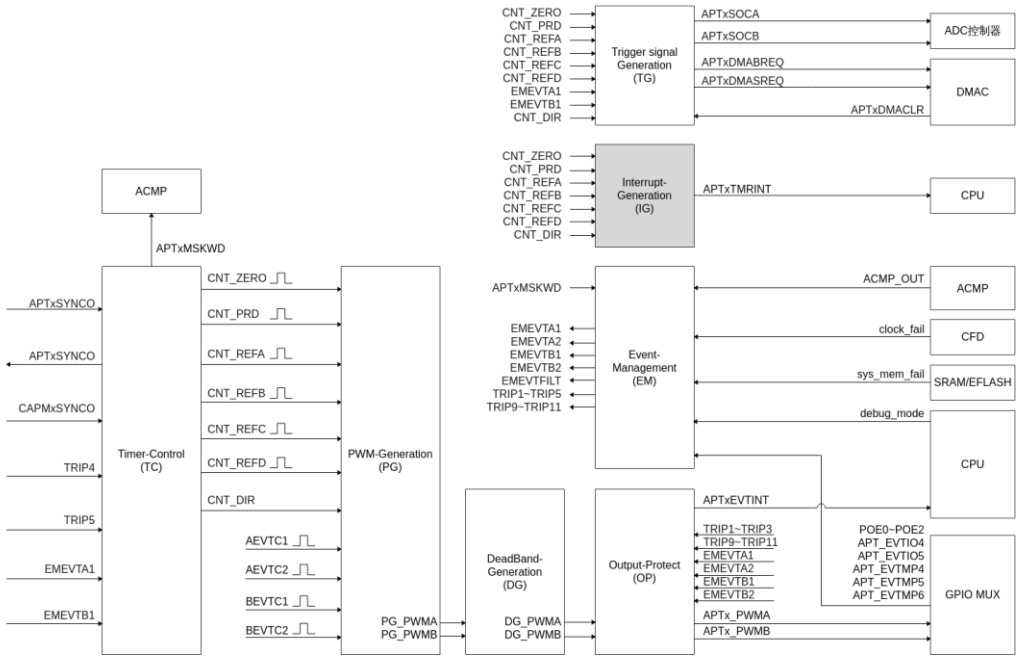
输出选择逻辑可以通过 EM_OUT_SEL 寄存器的相关位域选择 EMEVTA1、EMEVTA2、EMEVTB1、EMEVTB2 的源：

- EMEVTA1
  - EM_OUT_SEL[rg_evta1_sel]=0，EMEVTA1_TEMP 作为 EMEVTA1 的源；
  - EM_OUT_SEL[rg_evta1_sel]=1，EMEVTFILT 作为 EMEVTA1 的源。
- EMEVTA2
  - EM_OUT_SEL[rg_evta2_sel]=0，EMEVTA2_TEMP 作为 EMEVTA2 的源；
  - EM_OUT_SEL[rg_evta2_sel]=1，EMEVTFILT 作为 EMEVTA2 的源。
- EMEVTB1
  - EM_OUT_SEL[rg_evtb1_sel]=0，EMEVTB1_TEMP 作为 EMEVTB1 的源；
  - EM_OUT_SEL[rg_evtb1_sel]=1，EMEVTFILT 作为 EMEVTB1 的源。
- EMEVTB2
  - EM_OUT_SEL[rg_evtb2_sel]=0，EMEVTB2_TEMP 作为 EMEVTB2 的源；
  - EM_OUT_SEL[rg_evtb2_sel]=1，EMEVTFILT 作为 EMEVTB2 的源。

14.8 中断生成子模块（Interrupt-Generation）

APT 系统中中断生成子模块如图 14-46 所示。

图14-46 中断生成子模块

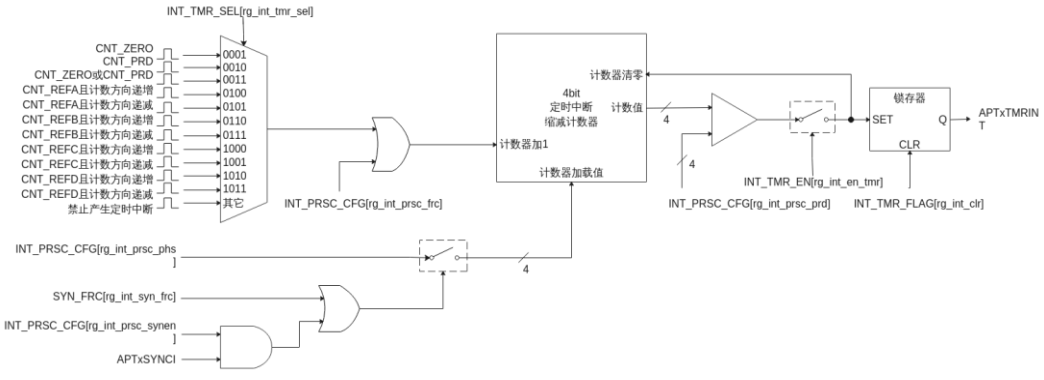


14.8.1 中断生成子模块的功能

中断生成子模块的功能是接收计数控制子模块的输出信号，在特定的时刻上报定时中断（APTxTMRINT）。

图 14-47 显示了定时中断产生的机制。

图14-47 定时中断产生机制



- 中断置位

  中断生成子模块有一个独立的 4bit 定时中断缩减计数器对以下计数控制子模块的输出信号之一（通过 INT_TMR_SEL[rg_int_tmr_sel]选择）进行增计数：

  - CNT_ZERO

- CNT_PRD
- CNT_ZERO 或 CNT_PRD
- CNT_REFA 且计数方向递增
- CNT_REFA 且计数方向递减
- CNT_REFB 且计数方向递增
- CNT_REFB 且计数方向递减
- CNT_REFC 且计数方向递增
- CNT_REFC 且计数方向递减
- CNT_REFD 且计数方向递增
- CNT_REFD 且计数方向递减

当指定的信号有效时，定时中断缩减计数器将递增，直到计数到 INT_PRSC_CFG[rg_int_prsc_prd]配置值后清零，如果定时中断使能（INT_TMR_EN[rg_int_en_tmr]=1），则上报定时中断。

定时中断支持软件注入中断，软件通过向 INT_PRSC_CFG[rg_int_prsc_frc]写 1，会将定时中断缩减计数器加 1。

定时中断缩减计数器支持同步，通过配置 INT_PRSC_CFG[rg_int_prsc_synen]=1 使能，使能同步功能后，当 APTxSYNCI 信号有效时，定时中断缩减计数器的计数值会被更新成 INT_PRSC_CFG[rg_int_prsc_phs]中配置的值。当软件向 SYN_FRC[rg_int_syn_frc]写 1 时，也会触发一次定时中断缩减计数器的同步。

软件可以通过读取 INT_PRSC_CFG[ro_int_prsc_cnt]获取定时中断缩减计数器的计数值。

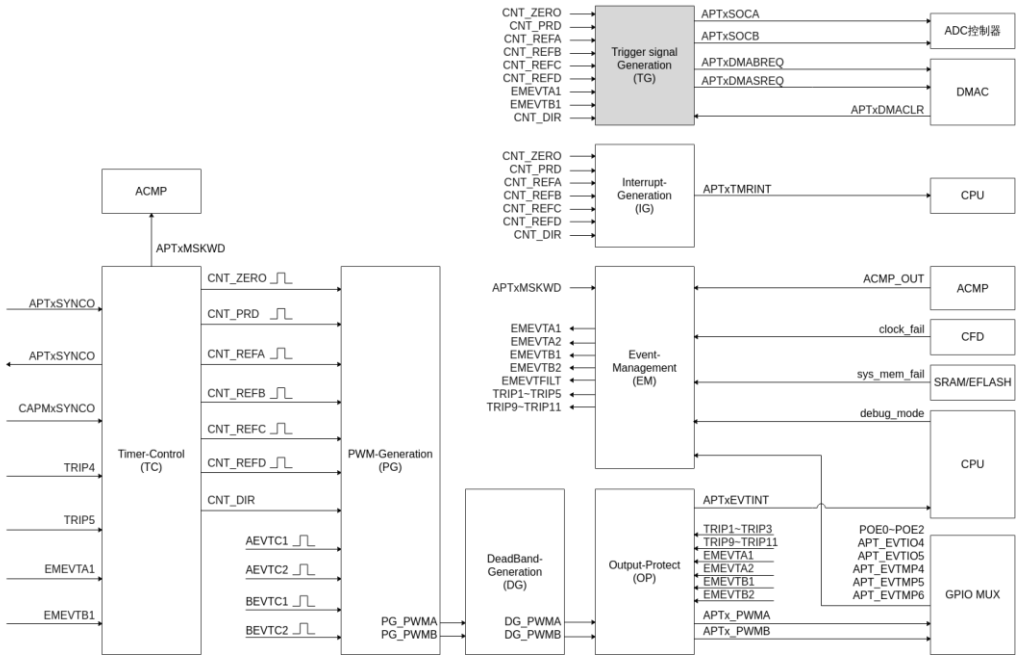
- 中断查询和清除

  软件通过读 INT_TMR_FLAG[ro_int_flag_tmr]获取定时中断标志，向 INT_TMR_FLAG[rg_int_clr_tmr]写 1 清除定时中断标志。

14.9 触发信号生成子模块（Trigger signal Generation）

APT 系统中触发信号生成子模块如图 14-48 所示。

图14-48 触发信号生成子模块



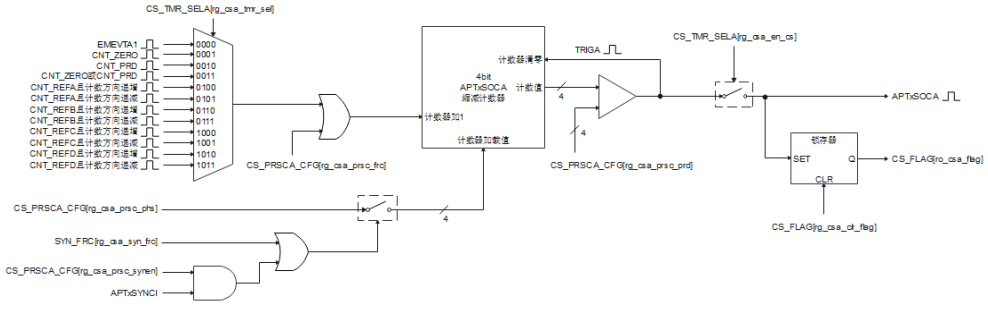
14.9.1 触发信号生成子模块的功能

触发信号生成子模块的功能是接收计数控制子模块的输出信号和事件管理子模块输出的合并事件，在特定的时刻产生两个触发 ADC 启动转换（APTxSOCA 和 APTxSOCB）的信号，以及两个 DMA 请求（APTxDMABREQ 和 APTxDMASREQ）。

- APTxSOCA 信号

  APTxSOCA 信号产生机制如图 14-49 所示。

图14-49 APTxSOCA 信号产生机制



触发信号生成子模块有一个独立的 4bit APTxSOCA 缩减计数器对以下信号之一（通过 CS_TMR_SELA[rg_csa_tmr_sel]选择）进行增计数：

- 合并事件 EMEVTA1 有效（脉冲有效）
- CNT_ZERO
- CNT_PRD
- CNT_ZERO 或 CNT_PRD
- CNT_REFA 且计数方向递增
- CNT_REFA 且计数方向递减
- CNT_REFB 且计数方向递增
- CNT_REFB 且计数方向递减
- CNT_REFC 且计数方向递减
- CNT_REFC 且计数方向递减
- CNT_REFD 且计数方向递减
- CNT_REFD 且计数方向递减

当指定的信号有效时，APTxSOCA 缩减计数器将递增，直到计数到 CS_PRSCA_CFG[rg_csa_prsc_prd]配置值后产生 TRIGA 信号并清零计数器，如果 APTxSOCA 使能（CS_TMR_SELA[rg_csa_en_cs]=1），则会产生 APTxSOCA。

APTxSOCA 支持软件注入，软件通过向 CS_PRSCA_CFG[rg_csa_prsc_frc]写 1，会将 APTxSOCA 缩减计数器加 1。

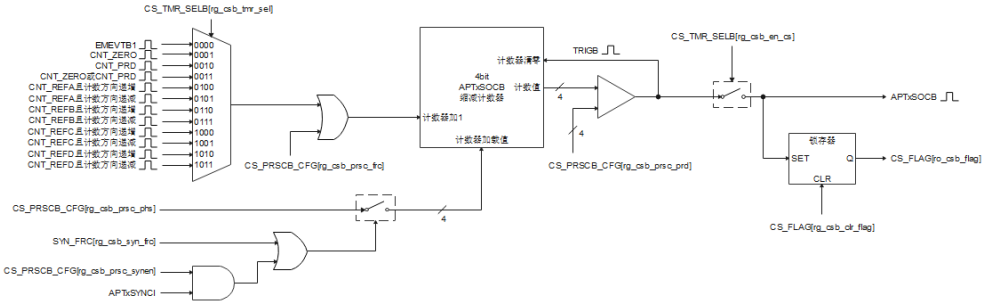
APTxSOCA 缩减计数器支持同步，通过配置 CS_PRSCA_CFG[rg_csa_prsc_synen]=1 使能，使能同步功能后，当 APTxSYNCI 信号有效时，APTxSOCA 缩减计数器的计数值会被更新成 CS_PRSCA_CFG[rg_csa_prsc_phs]中配置的值。当软件向 SYN_FRC[rg_csa_syn_frc]写 1 时，也会触发一次 APTxSOCA 缩减计数器的同步。

如果产生了 APTxSOCA，CS_FLAG[ro_csa_flag]标志会置位，软件通过向 CS_FLAG[rg_csa_clr_flag]写 1 清除标志。

- APTxSOCB 信号

  APTxSOCB 信号产生机制如图 14-50 所示。

图14-50 APTxSOCB 信号产生机制



触发信号生成子模块有一个独立的 4bit APTxSOCB 缩减计数器对以下信号之一（通过 CS_TMR_SELB[rg_csb_tmr_sel]选择）进行增计数：

- 合并事件 EMEVTB1 有效（脉冲有效）
- CNT_ZERO
- CNT_PRD
- CNT_ZERO 或 CNT_PRD
- CNT_REFA 且计数方向递增
- CNT_REFA 且计数方向递减
- CNT_REFB 且计数方向递增
- CNT_REFB 且计数方向递减
- CNT_REFC 且计数方向递减
- CNT_REFC 且计数方向递减
- CNT_REFD 且计数方向递减
- CNT_REFD 且计数方向递减

当指定的信号有效时，APTxSOCB 缩减计数器将递增，直到计数到 CS_PRSCB_CFG[rg_csb_prsc_prd]配置值后产生 TRIGB 信号并清零计数器，如果 APTxSOCB 使能（CS_TMR_SELB[rg_csb_en_cs]=1），则会产生 APTxSOCB。

APTxSOCB 支持软件注入，软件通过向 CS_PRSCB_CFG[rg_csb_prsc_frc]写 1，会将 APTxSOCB 缩减计数器加 1。

APTxSOCB 缩减计数器支持同步，通过配置 CS_PRSCB_CFG[rg_csb_prsc_synen]=1 使能，使能同步功能后，当 APTxSYNCI 信号有效时，APTxSOCB 缩减计数器的计数值会被更新成 CS_PRSCB_CFG[rg_csb_prsc_phs]中配置的值。当软件向 SYN_FRC[rg_csb_syn_frc]写 1 时，也会触发一次 APTxSOCB 缩减计数器的同步。

如果产生了 APTxSOCB，CS_FLAG[ro_csb_flag]标志会置位，软件通过向 CS_FLAG[rg_csb_clr_flag]写 1 清除标志。

- APTxDMABREQ 信号

  当以下条件之一满足时，产生 APTxDMABREQ 信号（通过 CS_DMA[rg_dma_breq_sel]选择）：

  - TRIGA 信号有效
  - TRIGB 信号有效

- APTxDMASREQ 信号

  当以下条件之一满足时，产生 APTxDMASREQ 信号（通过 CS_DMA[rg_dma_sreq_sel]选择）：

  - TRIGA 信号有效
  - TRIGB 信号有效

14.10 APTx 寄存器概览

APTx（其中 x=0~4）寄存器概览如表 14-11 所示。

表14-11 APTx 寄存器概览（APTx 的基址是 0x14A0_x000）

偏移地址	名称	描述	页码
0x0000	VER_INFO	版本信息寄存器	VER_INFO
0x0010	TC_MODE	计数模式寄存器	TC_MODE
0x0014	TC_PHS	计数相位控制寄存器	TC_PHS
0x0018	TC_OVRID	计数器初始化寄存器	TC_OVRID
0x0020	TC_PRD	计数周期寄存器	TC_PRD
0x0030	TC_REFA	计数参考值 A 寄存器	TC_REFA
0x0034	TC_REFB	计数参考值 B 寄存器	TC_REFB
0x0038	TC_REFC	计数参考值 C 寄存器	TC_REFC

偏移地址	名称	描述	页码
0x003C	TC_REFD	计数参考值 D 寄存器	TC_REFD
0x0050	TC_BUF_EN	计数器缓存使能寄存器	TC_BUF_EN
0x0054	TC_PRD_LOAD	计数周期寄存器独立加载触发信号选择寄存器	TC_PRD_LOAD
0x0058	TC_REF_LOAD	计数参考值寄存器独立加载触发信号选择寄存器	TC_REF_LOAD
0x005C	TC_MWD_EN	屏蔽窗口使能寄存器	TC_MWD_EN
0x0060	TC_MWDREFA	屏蔽窗口参考值 A 寄存器	TC_MWDREFA
0x0064	TC_MWDREFB	屏蔽窗口参考值 B 寄存器	TC_MWDREFB
0x0068	TC_MWD_ACT	MASK_WD 动作寄存器	TC_MWD_ACT
0x006C	TC_MWD_BUF_EN	屏蔽窗口缓存使能寄存器	TC_MWD_BUF_EN
0x0070	TC_MWD_LOAD	屏蔽窗口寄存器独立加载触发信号选择寄存器	TC_MWD_LOAD
0x0074	TC_STS	计数状态寄存器	TC_STS
0x0100	PG_ACT_A	PG_PWMA 动作配置寄存器	PG_ACT_A
0x0104	PG_ACT_B	PG_PWMB 动作配置寄存器	PG_ACT_B
0x0110	PG_ACT_FRC	强制动作配置寄存器	PG_ACT_FRC
0x0114	PG_OUT_FRC	强制输出配置寄存器	PG_OUT_FRC
0x0120	PG_BUF_EN	波形生成缓存使能寄存器	PG_BUF_EN

偏移地址	名称	描述	页码
0x0130	PG_ACT_LD	波形生成动作独立加载触发信号选择寄存器	PG_ACT_LD
0x0140	PG_EVTC_SEL	波形生成事件 C 触发源选择寄存器	PG_EVTC_SEL
0x0200	DG_RED	死区上升沿延时配置寄存器	DG_RED
0x0204	DG_FED	死区下降沿延时配置寄存器	DG_FED
0x0208	DG_CFG	死区配置寄存器	DG_CFG
0x0210	DG_BUF_EN	死区生成缓存使能寄存器	DG_BUF_EN
0x0214	DG_BUF_LOAD	死区生成独立加载触发信号选择寄存器	DG_BUF_LOAD
0x0300	OC_MODE	输出保护模式寄存器	OC_MODE
0x0304	OC_LAT_EN	故障信号锁存使能寄存器	OC_LAT_EN
0x0310	OC_ACT_A	APTx_PWMA 输出保护寄存器	OC_ACT_A
0x0314	OC_ACT_B	APTx_PWMB 输出保护寄存器	OC_ACT_B
0x0320	OC_EVT_FLAG	故障信号和事件中断标志寄存器	OC_EVT_FLAG
0x0324	OC_PRD_CLR	周期性保护自动清除寄存器	OC_PRD_CLR
0x0330	OC_FRC_EVT	强制产生故障信号寄存器	OC_FRC_EVT
0x0410	INT_EVT_EN	事件中断使能寄存器	INT_EVT_EN
0x0414	INT_TMR_EN	定时中断使能寄存器	INT_TMR_EN
0x0420	INT_TMR_FLAG	定时中断标志寄存器	INT_TMR_FLAG
0x0424	INT_TMR_SEL	定时中断的触发条件寄存器	INT_TMR_SE

偏移地址	名称	描述	页码
			L
0x0428	INT_PRSC_CFG	定时中断缩减配置寄存器	INT_PRSC_CFG
0x0500	CS_TMR_SELA	APTxSOCA 的触发源选择寄存器	CS_TMR_SELA
0x0504	CS_TMR_SELB	APTxSOCB 的触发源选择寄存器	CS_TMR_SELB
0x0508	CS_PRSCA_CFG	APTxSOCA 缩减配置寄存器	CS_PRSCA_CFG
0x050C	CS_PRSCB_CFG	APTxSOCB 缩减配置寄存器	CS_PRSCB_CFG
0x0510	CS_FLAG	APTxSOCA/APTxSOCB 标志寄存器	CS_FLAG
0x0520	CS_DMA	DMA 请求配置寄存器	CS_DMA
0x0600	EM_EVTIO_PSEL	事件管理极性选择寄存器 1	EM_EVTIO_PSEL
0x0604	EM_EVTMP_PSEL	事件管理极性选择寄存器 2	EM_EVTMP_PSEL
0x0608	EM_AOR_EN	事件管理组合输入选择寄存器 A	EM_AOR_EN
0x060C	EM_BOR_EN	事件管理组合输入选择寄存器 B	EM_BOR_EN
0x0610	EM_MRG_SEL	事件管理输入选择寄存器	EM_MRG_SEL
0x0614	EM_OUT_SEL	事件管理输出选择寄存器	EM_OUT_SEL
0x0700	SYNI_CFG	同步输入配置寄存器	SYNI_CFG
0x0704	SYNCNT_CFG	计数器同步信号配置寄存器	SYNCNT_CFG
0x0708	SYNO_CFG	同步输出配置寄存器	SYNO_CFG

偏移地址	名称	描述	页码
0x0710	GLB_LOAD	全局加载配置寄存器	GLB_LOAD
0x0720	LOAD_STS	缓存状态寄存器	LOAD_STS
0x0730	SYN_FRC	软件强制寄存器	SYN_FRC

14.11 APTx 寄存器描述

VER_INFO

VER_INFO 为版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0010

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:4]	RO	main_version	主版本号。	0x1
[3:0]	RO	sub_version	子版本号。	0x0

TC_MODE

TC_MODE 为计数模式寄存器。

Offset Address: 0x0010 Total Reset Value: 0x2000_0000

Bits	Access	Name	Description	Reset
[31:30]	RO	reserved	保留。	0x0
[29:28]	RW	rg_emu_stop	CPU debug 模式，APT 停止的方式。注意，如果停止 APT，那么在 CPU debug 模式下，整个 APT 都会停止工作，包括输出控制功能。 0X：不停止； 10：停止分频计数器（DIVCNT）和时基计	0x2

Bits	Access	Name	Description	Reset
			数器（TCCNT）； 11：停止 APT。	
[27:16]	RW	rg_div_fac	分频系数。 分频倍数 = 分频系数 +1。	0x000
[15:2]	RO	reserved	保留。	0x0000
[1:0]	RW	rg_cnt_mode	时基计数器（TCCNT）的工作模式。 00：递增计数； 01：递减计数； 10：先增后减计数； 11：冻结。	0x0

TC_PHS

TC_PHS 为计数相位控制寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RW	rg_cnt_dir	当 APTCNTSYNC 有效时，更新时基计数器（TCCNT）的计数方向，仅在先增后减计数模式有效。 0：递减； 1：递增。	0x0
[30:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_phs	当 APTCNTSYNC 有效时，更新时基计数器（TCCNT）的计数值。	0x0000

TC_OVRID

TC_OVRID 为计数器初始化寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	W1_PULSE	rg_cnt_ovrid_en	时基计数器（TCCNT）和分频计数器（DIVCNT）的初始化使能。 0：无效; 1：初始化时基计数器（TCCNT）和分频计数器（DIVCNT）的计数值。	0x0
[30:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_ovrid	初始化时基计数器（TCCNT）的计数值。	0x0000

TC_PRD

TC_PRD 为计数周期寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_2710

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	rg_cnt_prd	时基计数器（TCCNT）的计数周期。	0x2710

TC_REFA

TC_REFA 为计数参考值 A 寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_refa	时基计数器（TCCNT）的参考值 A，应当小于或等于计数周期。	0x0000

TC_REFB

TC_REFB 为计数参考值 B 寄存器。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_refb	时基计数器（TCCNT）的参考值 B，应当小于或等于计数周期。	0x0000

TC_REFC

TC_REFC 为计数参考值 C 寄存器。

Offset Address: 0x0038 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_refc	时基计数器（TCCNT）的参考值 C，应当小于或等于计数周期。	0x0000

TC_REFD

TC_REFD 为计数参考值 D 寄存器。

Offset Address: 0x003C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0
[15:0]	RW	rg_cnt_refd	时基计数器（TCCNT）的参考值 D，应当小于或等于计数周期。	0x0000

TC_BUF_EN

TC_BUF_EN 为计数器缓存使能寄存器。

Offset Address: 0x0050 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000

Bits	Access	Name	Description	Reset
[11]	RW	rg_refd_gld_en	TC_REFD 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[10]	RW	rg_refd_buf_en	TC_REFD 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[9]	RW	rg_refc_gld_en	TC_REFC 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[8]	RW	rg_refc_buf_en	TC_REFC 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[7]	RW	rg_refb_gld_en	TC_REFB 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[6]	RW	rg_refb_buf_en	TC_REFB 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[5]	RW	rg_refa_gld_en	TC_REFA 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[4]	RW	rg_refa_buf_en	TC_REFA 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[3:2]	RO	reserved	保留。	0x0
[1]	RW	rg_prd_gld_en	TC_PRD 寄存器的全局加载使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[0]	RW	rg_prd_buf_en	TC_PRD 寄存器的缓存使能。 0：不使能； 1：使能。	0x0

TC_PRD_LOAD

TC_PRD_LOAD 为计数周期寄存器独立加载触发信号选择寄存器。

Offset Address: 0x0054 Total Reset Value: 0x0000_0011

Bits	Access	Name	Description	Reset
[31:5]	RO	reserved	保留。	0x0000000
[4]	RW	rg_prd_ld_synen	选择 APTxSYNCI 有效时，是否使能 TC_PRD 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[3]	RW	rg_prd_ld_b1en	选择 EMEVTB1 有效时，是否使能 TC_PRD 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[2]	RW	rg_prd_ld_a1en	选择 EMEVTA1 有效时，是否使能 TC_PRD 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[1]	RO	reserved	保留。	0x0
[0]	RW	rg_prd_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_PRD 寄存器的独立加载。 0：不使能； 1：使能。	0x1

TC_REF_LOAD

TC_REF_LOAD 为计数参考值寄存器独立加载触发信号选择寄存器。

Offset Address: 0x0058 Total Reset Value: 0x1111_1111

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28]	RW	rg_refd_ld_synen	选择 APTxSYNCI 有效时，是否使能 TC_REFD 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[27]	RW	rg_refd_ld_b1en	选择 EMEVTB1 有效时，是否使能 TC_REFD 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[26]	RW	rg_refd_ld_a1en	选择 EMEVTA1 有效时，是否使能 TC_REFD 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[25]	RW	rg_refd_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_REFD 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[24]	RW	rg_refd_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_REFD 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[23:21]	RO	reserved	保留。	0x0
[20]	RW	rg_refc_ld_synen	选择 APTxSYNCI 有效时，是否使能 TC_REFC 寄存器的独立加载。 0：不使能；	0x1

Bits	Access	Name	Description	Reset
			1：使能。	
[19]	RW	rg_refc_ld_b1en	选择 EMEVTB1 有效时，是否使能 TC_REFC 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[18]	RW	rg_refc_ld_a1en	选择 EMEVTA1 有效时，是否使能 TC_REFC 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[17]	RW	rg_refc_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_REFC 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[16]	RW	rg_refc_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_REFC 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[15:13]	RO	reserved	保留。	0x0
[12]	RW	rg_refb_ld_synen	选择 APTxSYNCI 有效时，是否使能 TC_REFB 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[11]	RW	rg_refb_ld_b1en	选择 EMEVTB1 有效时，是否使能 TC_REFB 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[10]	RW	rg_refb_ld_a1en	选择 EMEVTA1 有效时，是否使能 TC_REFB 寄存器的独立加载。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[9]	RW	rg_refb_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_REFB 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[8]	RW	rg_refb_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_REFB 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[7:5]	RO	reserved	保留。	0x0
[4]	RW	rg_refa_ld_synen	选择 APTxSYNCI 有效时，是否使能 TC_REFA 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[3]	RW	rg_refa_ld_b1en	选择 EMEVTB1 有效时，是否使能 TC_REFA 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[2]	RW	rg_refa_ld_a1en	选择 EMEVTA1 有效时，是否使能 TC_REFA 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[1]	RW	rg_refa_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_REFA 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[0]	RW	rg_refa_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_REFA 寄存器的独立加载。 0：不使能；	0x1

Bits	Access	Name	Description	Reset
			1：使能。	

TC_MWD_EN

TC_MWD_EN 为屏蔽窗口使能寄存器。

Offset Address: 0x005C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RW	rg_mskwd_en	屏蔽窗口使能位。 0：不使能； 1：使能。	0x0
[30:1]	RO	reserved	保留。	0x00000000
[0]	RW	rg_mskwd_psel	MASK_WD_POL 的源选择。 0：MASK_WD_POL 的源是 MASK_WD 取反后的波形； 1：MASK_WD_POL 的源是 MASK_WD。	0x0

TC_MWDREFA

TC_MWDREFA 为屏蔽窗口参考值 A 寄存器。

Offset Address: 0x0060 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	rg_mwd_refa	屏蔽窗口参考值 A，应当小于或等于计数周期。	0x0000

TC_MWDREFB

TC_MWDREFB 为屏蔽窗口参考值 B 寄存器。

Offset Address: 0x0064 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	rg_mwd_refb	屏蔽窗口参考值 B，应当小于或等于计数周期。	0x0000

TC_MWD_ACT

TC_MWD_ACT 为 MASK_WD 动作寄存器。

Offset Address: 0x0068 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:6]	RW	rg_mwdrefb_act_dec	CNT_MWDREFB 有效且 TCCNT 计数方向递减时，MASK_WD 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[5:4]	RW	rg_mwdrefb_act_inc	CNT_MWDREFB 有效且 TCCNT 计数方向递增时，MASK_WD 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[3:2]	RW	rg_mwdrefa_act_dec	CNT_MWDREFA 有效且 TCCNT 计数方向递减时，MASK_WD 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[1:0]	RW	rg_mwdrefa_act_in	CNT_MWDREFA 有效且 TCCNT 计数方	0x0

Bits	Access	Name	Description	Reset
		c	向递增时，MASK_WD 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	

TC_MWD_BUF_EN

TC_MWD_BUF_EN 为屏蔽窗口缓存使能寄存器。

Offset Address: 0x006C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RW	rg_mwd_act_gld_en	TC_MWD_ACT 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[4]	RW	rg_mwd_act_buf_en	TC_MWD_ACT 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[3]	RW	rg_mwdrefb_gld_en	TC_MWDREFB 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[2]	RW	rg_mwdrefb_buf_en	TC_MWDREFB 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[1]	RW	rg_mwdrefa_gld_en	TC_MWDREFA 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[0]	RW	rg_mwdrefa_buf_en	TC_MWDREFA 寄存器的缓存使能。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	

TC_MWD_LOAD

TC_MWD_LOAD 为屏蔽窗口寄存器独立加载触发信号选择寄存器。

Offset Address: 0x0070 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7]	RW	rg_mwd_act_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_MWD_ACT 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[6]	RW	rg_mwd_act_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_MWD_ACT 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[5]	RO	reserved	保留。	0x0
[4]	RW	rg_mwdrefb_ld_prden	选择 CNT_PRD 有效时，是否使能 TC_MWDREFB 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[3]	RW	rg_mwdrefb_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_MWDREFB 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[2]	RO	reserved	保留。	0x0
[1]	RW	rg_mwdrefa_ld_prden	选择 CNT_PRD 有效时，是否使能	0x0

Bits	Access	Name	Description	Reset
			TC_MWDREFA 寄存器的独立加载。 0：不使能； 1：使能。	
[0]	RW	rg_mwdrefa_ld_zroen	选择 CNT_ZERO 有效时，是否使能 TC_MWDREFA 寄存器的独立加载。 0：不使能； 1：使能。	0x0

TC_STS

TC_STS 为计数状态寄存器。

Offset Address: 0x0074 Total Reset Value: 0x8000_0000

Bits	Access	Name	Description	Reset
[31]	RO	ro_cnt_dir	时基计数器（TCCNT）的当前计数方向。 0：递减计数； 1：递增计数。	0x1
[30:28]	RO	reserved	保留。	0x0
[27:16]	RO	ro_div_cnt	分频计数器（DIVCNT）的当前计数值。	0x000
[15:0]	RO	ro_cnt_val	时基计数器（TCCNT）的当前计数值。	0x0000

PG_ACT_A

PG_ACT_A 为 PG_PWMA 动作配置寄存器。

Offset Address: 0x0100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:28]	RO	reserved	保留。	0x0
[27:26]	RW	rg_pga_act_evtc2_dec	AEVTC2 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。	0x0

Bits	Access	Name	Description	Reset
			00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	
[25:24]	RW	rg_pga_act_evtc2_inc	AEVTC2 有效且 TCCNT 计数方向递增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[23:22]	RW	rg_pga_act_evtc1_dec	AEVTC1 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[21:20]	RW	rg_pga_act_evtc1_inc	AEVTC1 有效且 TCCNT 计数方向递增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[19:18]	RW	rg_pga_act_refd_dec	CNT_REFD 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[17:16]	RW	rg_pga_act_refd_inc	CNT_REFD 有效且 TCCNT 计数方向递	0x0

Bits	Access	Name	Description	Reset
			增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	
[15:14]	RW	rg_pga_act_refc_dec	CNT_REFC 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[13:12]	RW	rg_pga_act_refc_inc	CNT_REFC 有效且 TCCNT 计数方向递增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[11:10]	RW	rg_pga_act_refb_dec	CNT_REFB 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[9:8]	RW	rg_pga_act_refb_inc	CNT_REFB 有效且 TCCNT 计数方向递增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0

Bits	Access	Name	Description	Reset
[7:6]	RW	rg_pga_act_refa_dec	CNT_REFA 有效且 TCCNT 计数方向递减时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[5:4]	RW	rg_pga_act_refa_inc	CNT_REFA 有效且 TCCNT 计数方向递增时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[3:2]	RW	rg_pga_act_prd	CNT_PRD 有效时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[1:0]	RW	rg_pga_act_zro	CNT_ZERO 有效时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0

PG_ACT_B

PG_ACT_B 为 PG_PWMB 动作配置寄存器。

Offset Address: 0x0104 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:28]	RO	reserved	保留。	0x0
[27:26]	RW	rg_pgb_act_evtc2_dec	BEVTC2 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[25:24]	RW	rg_pgb_act_evtc2_inc	BEVTC2 有效且 TCCNT 计数方向递增时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[23:22]	RW	rg_pgb_act_evtc1_dec	BEVTC1 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[21:20]	RW	rg_pgb_act_evtc1_inc	BEVTC1 有效且 TCCNT 计数方向递增时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[19:18]	RW	rg_pgb_act_refd_dec	CNT_REFD 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响；	0x0

Bits	Access	Name	Description	Reset
			01：置为低电平； 10：置为高电平； 11：翻转电平。	
[17:16]	RW	rg_pgb_act_refd_inc	CNT_REFD 有效且 TCCNT 计数方向递增时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[15:14]	RW	rg_pgb_act_refc_dec	CNT_REFC 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[13:12]	RW	rg_pgb_act_refc_inc	CNT_REFC 有效且 TCCNT 计数方向递增时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[11:10]	RW	rg_pgb_act_refb_dec	CNT_REFB 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[9:8]	RW	rg_pgb_act_refb_inc	CNT_REFB 有效且 TCCNT 计数方向递增	0x0

Bits	Access	Name	Description	Reset
			时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	
[7:6]	RW	rg_pgb_act_refa_dec	CNT_REFA 有效且 TCCNT 计数方向递减时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[5:4]	RW	rg_pgb_act_refa_inc	CNT_REFA 有效且 TCCNT 计数方向递增时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[3:2]	RW	rg_pgb_act_prd	CNT_PRD 有效时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[1:0]	RW	rg_pgb_act_zro	CNT_ZERO 有效时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0

PG_ACT_FRC

PG_ACT_FRC 为强制动作配置寄存器。

Offset Address: 0x0110 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:7]	RO	reserved	保留。	0x0000000
[6]	W1_PULSE	rg_pgb_evt_frc	软件强制 PG_PWMB 动作，每次写 1 产生一次 PG_PWMB 动作，然后自动清零。 0：写 0 无效； 1：写 1 有效。	0x0
[5:4]	RW	rg_pgb_act_evt_frc	软件强制 PG_PWMB 动作有效时，PG_PWMB 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0
[3]	RO	reserved	保留。	0x0
[2]	W1_PULSE	rg_pga_evt_frc	软件强制 PG_PWMA 动作，每次写 1 产生一次 PG_PWMA 动作，然后自动清零。 0：写 0 无效； 1：写 1 有效。	0x0
[1:0]	RW	rg_pga_act_evt_frc	软件强制 PG_PWMA 动作有效时，PG_PWMA 的动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0

PG_OUT_FRC

PG_OUT_FRC 为强制输出配置寄存器。

Offset Address: 0x0114 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:7]	RO	reserved	保留。	0x0000000
[6]	RW	rg_pgb_frc_en	PG_PWMB 软件强制输出使能。 0：不使能； 1：使能。	0x0
[5:4]	RW	rg_pgb_frc_act	PG_PWMB 软件强制输出使能时，PG_PWMB 的输出状态。 00：锁定电平； 01：置为低电平； 10：置为高电平； 11：保留。	0x0
[3]	RO	reserved	保留。	0x0
[2]	RW	rg_pga_frc_en	PG_PWMA 软件强制输出使能。 0：不使能； 1：使能。	0x0
[1:0]	RW	rg_pga_frc_act	PG_PWMA 软件强制输出使能时，PG_PWMA 的输出状态。 00：锁定电平； 01：置为低电平； 10：置为高电平； 11：保留。	0x0

PG_BUF_EN

PG_BUF_EN 为波形生成缓存使能寄存器。

Offset Address: 0x0120 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RW	rg_frc_gld_en	PG_OUT_FRC 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[4]	RW	rg_frc_buf_en	PG_OUT_FRC 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[3]	RW	rg_actb_gld_en	PG_ACT_B 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[2]	RW	rg_actb_buf_en	PG_ACT_B 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[1]	RW	rg_acta_gld_en	PG_ACT_A 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[0]	RW	rg_acta_buf_en	PG_ACT_A 寄存器的缓存使能。 0：不使能； 1：使能。	0x0

PG_ACT_LD

PG_ACT_LD 为波形生成动作独立加载触发信号选择寄存器。

Offset Address: 0x0130 Total Reset Value: 0x0011_1111

Bits	Access	Name	Description	Reset
[31:21]	RO	reserved	保留。	0x000
[20]	RW	rg_pg_frcld_synen	选择 APTxSYNCI 有效时，是否使能 PG_OUT_FRC 寄存器的独立加载。	0x1

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[19:18]	RO	reserved	保留。	0x0
[17]	RW	rg_pg_frcld_prden	选择 CNT_PRD 有效时，是否使能 PG_OUT_FRC 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[16]	RW	rg_pg_frcld_zroen	选择 CNT_ZERO 有效时，是否使能 PG_OUT_FRC 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[15:13]	RO	reserved	保留。	0x0
[12]	RW	rg_pgb_actld_synen	选择 APTxSYNCI 有效时，是否使能 PG_ACT_B 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[11]	RW	rg_pgb_actld_b1en	选择 EMEVTB1 有效时，是否使能 PG_ACT_B 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[10]	RW	rg_pgb_actld_a1en	选择 EMEVTA1 有效时，是否使能 PG_ACT_B 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[9]	RW	rg_pgb_actld_prden	选择 CNT_PRD 有效时，是否使能 PG_ACT_B 寄存器的独立加载。 0：不使能； 1：使能。	0x0

Bits	Access	Name	Description	Reset
[8]	RW	rg_pgb_actld_zroen	选择 CNT_ZERO 有效时，是否使能 PG_ACT_B 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[7:5]	RO	reserved	保留。	0x0
[4]	RW	rg_pga_actld_synen	选择 APTxSYNCI 有效时，是否使能 PG_ACT_A 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[3]	RW	rg_pga_actld_b1en	选择 EMEVTB1 有效时，是否使能 PG_ACT_A 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[2]	RW	rg_pga_actld_a1en	选择 EMEVTA1 有效时，是否使能 PG_ACT_A 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[1]	RW	rg_pga_actld_prden	选择 CNT_PRD 有效时，是否使能 PG_ACT_A 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[0]	RW	rg_pga_actld_zroen	选择 CNT_ZERO 有效时，是否使能 PG_ACT_A 寄存器的独立加载。 0：不使能； 1：使能。	0x1

PG_EVTC_SEL

PG_EVTC_SEL 为波形生成事件 C 触发源选择寄存器。

Offset Address: 0x0140 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:12]	RW	rg_pgb_evtc2_sel	BEVTC2 触发源选择。 0x1：EMEVTA1_TEMP； 0x2：EMEVTA2_TEMP； 0x3：EMEVTB1_TEMP； 0x4：EMEVTB2_TEMP； 0x5：EMEVTFILT； 0x6：TRIP1； 0x7：TRIP2； 0x8：TRIP3； 0x9：APTxSYNCI； 其他：禁止产生 BEVTC2。	0x0
[11:8]	RW	rg_pgb_evtc1_sel	BEVTC1 触发源选择。 0x1：EMEVTA1_TEMP； 0x2：EMEVTA2_TEMP； 0x3：EMEVTB1_TEMP； 0x4：EMEVTB2_TEMP； 0x5：EMEVTFILT； 0x6：TRIP1； 0x7：TRIP2； 0x8：TRIP3； 0x9：APTxSYNCI； 其他：禁止产生 BEVTC1。	0x0
[7:4]	RW	rg_pga_evtc2_sel	AEVTC2 触发源选择。 0x1：EMEVTA1_TEMP； 0x2：EMEVTA2_TEMP； 0x3：EMEVTB1_TEMP；	0x0

Bits	Access	Name	Description	Reset
			0x4：EMEVTB2_TEMP； 0x5：EMEVTFILT； 0x6：TRIP1； 0x7：TRIP2； 0x8：TRIP3； 0x9：APTxSYNCI； 其他：禁止产生 AEVTC2。	
[3:0]	RW	rg_pga_evtc1_sel	AEVTC1 触发源选择。 0x1：EMEVTA1_TEMP； 0x2：EMEVTA2_TEMP； 0x3：EMEVTB1_TEMP； 0x4：EMEVTB2_TEMP； 0x5：EMEVTFILT； 0x6：TRIP1； 0x7：TRIP2； 0x8：TRIP3； 0x9：APTxSYNCI； 其他：禁止产生 AEVTC1。	0x0

DG_RED

DG_RED 为死区上升沿延时配置寄存器。

Offset Address: 0x0200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	rg_dg_red	死区上升沿延时配置。	0x0000

DG_FED

DG_FED 为死区下降沿延时配置寄存器。

Offset Address: 0x0204 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	rg_dg_fed	死区下降沿延时配置。	0x0000

DG_CFG

DG_CFG 为死区配置寄存器。

Offset Address: 0x0208 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	RO	reserved	保留。	0x000000
[9]	RW	rg_dgb_osel	选择 DG_PWMB 的源。 0：选择 PWM_FED_POL 作为 DG_PWMB； 1：选择 PWM_RED_POL 作为 DG_PWMB。	0x0
[8]	RW	rg_dga_osel	选择 DG_PWMA 的源。 0：选择 PWM_RED_POL 作为 DG_PWMA； 1：选择 PWM_FED_POL 作为 DG_PWMA。	0x0
[7:6]	RW	rg_dg_fed_osel	选择 PWM_FED_POL 的源。 0X：选择 PWM_FED_AFTER 作为 PWM_FED_POL； 10：选择 PWM_FED_AFTER 取反后作为 PWM_FED_POL； 11：选择 PG_PWMB 作为 PWM_FED_POL。	0x0

Bits	Access	Name	Description	Reset
[5:4]	RW	rg_dg_red_osel	选择 PWM_RED_POL 的源。 0X：选择 PWM_RED_AFTER 作为 PWM_RED_POL； 10：选择 PWM_RED_AFTER 取反后作为 PWM_RED_POL； 11：选择 PG_PWMA 作为 PWM_RED_POL。	0x0
[3:2]	RW	rg_dg_fed_isel	选择 PWM_FED_BEFORE 的源。 00：选择 PG_PWMB 作为 PWM_FED_BEFORE； 01：选择 PG_PWMA 作为 PWM_FED_BEFORE； 10：选择 PWM_RED_BEFORE 作为 PWM_FED_BEFORE； 11：将 PWM_FED_BEFORE 固定为低电平。	0x0
[1:0]	RW	rg_dg_red_isel	选择 PWM_RED_BEFORE 的源。 X0：选择 PG_PWMA 作为 PWM_RED_BEFORE； X1：选择 PG_PWMB 作为 PWM_RED_BEFORE。	0x0

DG_BUF_EN

DG_BUF_EN 为死区生成缓存使能寄存器。

Offset Address: 0x0210 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RW	rg_cfg_gld_en	DG_CFG 寄存器的全局加载使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[4]	RW	rg_cfg_buf_en	DG_CFG 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[3]	RW	rg_fed_gld_en	DG_CFG 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[2]	RW	rg_fed_buf_en	DG_FED 寄存器的缓存使能。 0：不使能； 1：使能。	0x0
[1]	RW	rg_red_gld_en	DG_RED 寄存器的全局加载使能。 0：不使能； 1：使能。	0x0
[0]	RW	rg_red_buf_en	DG_RED 寄存器的缓存使能。 0：不使能； 1：使能。	0x0

DG_BUF_LOAD

DG_BUF_LOAD 为死区生成独立加载触发信号选择寄存器。

Offset Address: 0x0214 Total Reset Value: 0x0001_0101

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17]	RW	rg_cfg_ld_prden	选择 CNT_PRD 有效时，是否使能 DG_CFG 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[16]	RW	rg_cfg_ld_zroen	选择 CNT_ZERO 有效时，是否使能	0x1

Bits	Access	Name	Description	Reset
			DG_CFG 寄存器的独立加载。 0：不使能； 1：使能。	
[15:10]	RO	reserved	保留。	0x00
[9]	RW	rg_fed_ld_prden	选择 CNT_PRD 有效时，是否使能 DG_FED 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[8]	RW	rg_fed_ld_zroen	选择 CNT_ZERO 有效时，是否使能 DG_FED 寄存器的独立加载。 0：不使能； 1：使能。	0x1
[7:2]	RO	reserved	保留。	0x00
[1]	RW	rg_red_ld_prden	选择 CNT_PRD 有效时，是否使能 DG_RED 寄存器的独立加载。 0：不使能； 1：使能。	0x0
[0]	RW	rg_red_ld_zroen	选择 CNT_ZERO 有效时，是否使能 DG_RED 寄存器的独立加载。 0：不使能； 1：使能。	0x1

OC_MODE

OC_MODE 为输出保护模式寄存器。

Offset Address: 0x0300 Total Reset Value: 0x0070_0070

Bits	Access	Name	Description	Reset
[31:28]	RO	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[27]	RW	rg_oc_mode_evtb2	EMEVTB2 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[26]	RW	rg_oc_mode_evtb1	EMEVTB1 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[25]	RW	rg_oc_mode_evta2	EMEVTA2 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[24]	RW	rg_oc_mode_evta1	EMEVTA1 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[23]	RO	reserved	保留。	0x0
[22]	RW	rg_oc_mode_evts3	TRIP11 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x1
[21]	RW	rg_oc_mode_evts2	TRIP10 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x1
[20]	RW	rg_oc_mode_evts1	TRIP9 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x1
[19]	RO	reserved	保留。	0x0
[18]	RW	rg_oc_mode_evt3	TRIP3 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[17]	RW	rg_oc_mode_evt2	TRIP2 的输出保护模式选择。	0x0

Bits	Access	Name	Description	Reset
			0：持续性保护； 1：周期性保护。	
[16]	RW	rg_oc_mode_evt1	TRIP1 的输出保护模式选择。 0：持续性保护； 1：周期性保护。	0x0
[15:12]	RO	reserved	保留。	0x0
[11]	RW	rg_oc_en_evtb2	EMEVTB2 的输出保护使能。 0：不使能； 1：使能。	0x0
[10]	RW	rg_oc_en_evtb1	EMEVTB1 的输出保护使能。 0：不使能； 1：使能。	0x0
[9]	RW	rg_oc_en_evta2	EMEVTA2 的输出保护使能。 0：不使能； 1：使能。	0x0
[8]	RW	rg_oc_en_evta1	EMEVTA1 的输出保护使能。 0：不使能； 1：使能。	0x0
[7]	RO	reserved	保留。	0x0
[6]	RW	rg_oc_en_evts3	TRIP11 的输出保护使能。 0：不使能； 1：使能。	0x1
[5]	RW	rg_oc_en_evts2	TRIP10 的输出保护使能。 0：不使能； 1：使能。	0x1
[4]	RW	rg_oc_en_evts1	TRIP9 的输出保护使能。 0：不使能；	0x1

Bits	Access	Name	Description	Reset
			1：使能。	
[3]	RO	reserved	保留。	0x0
[2]	RW	rg_oc_en_evt3	TRIP3 的输出保护使能。 0：不使能； 1：使能。	0x0
[1]	RW	rg_oc_en_evt2	TRIP2 的输出保护使能。 0：不使能； 1：使能。	0x0
[0]	RW	rg_oc_en_evt1	TRIP1 的输出保护使能。 0：不使能； 1：使能。	0x0

OC_LAT_EN

OC_LAT_EN 为故障信号锁存使能寄存器。

Offset Address: 0x0304 Total Reset Value: 0x0000_0F77

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11]	RW	rg_oc_laten_evtb2	EMEVTB2 的锁存使能。 0：不使能； 1：使能。	0x1
[10]	RW	rg_oc_laten_evtb1	EMEVTB1 的锁存使能。 0：不使能； 1：使能。	0x1
[9]	RW	rg_oc_laten_evta2	EMEVTA2 的锁存使能。 0：不使能； 1：使能。	0x1

Bits	Access	Name	Description	Reset
[8]	RW	rg_oc_laten_evta1	EMEVTA1 的锁存使能。 0：不使能； 1：使能。	0x1
[7]	RO	reserved	保留。	0x0
[6]	RW	rg_oc_laten_evts3	TRIP11 的锁存使能。 0：不使能； 1：使能。	0x1
[5]	RW	rg_oc_laten_evts2	TRIP10 的锁存使能。 0：不使能； 1：使能。	0x1
[4]	RW	rg_oc_laten_evts1	TRIP9 的锁存使能。 0：不使能； 1：使能。	0x1
[3]	RO	reserved	保留。	0x0
[2]	RW	rg_oc_laten_evt3	TRIP3 的锁存使能。 0：不使能； 1：使能。	0x1
[1]	RW	rg_oc_laten_evt2	TRIP2 的锁存使能。 0：不使能； 1：使能。	0x1
[0]	RW	rg_oc_laten_evt1	TRIP1 的锁存使能。 0：不使能； 1：使能。	0x1

OC_ACT_A

OC_ACT_A 为 APTx_PWMA 输出保护寄存器。

Offset Address: 0x0310 Total Reset Value: 0x0007_0007

Bits	Access	Name	Description	Reset
[31]	RO	reserved	保留。	0x0
[30:28]	RW	rg_oca_evtb2_dec	EMEVTB2 有效且 TCCNT 计数方向递减时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[27:25]	RW	rg_oca_evtb1_dec	EMEVTB1 有效且 TCCNT 计数方向递减时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[24:22]	RW	rg_oca_evta2_dec	EMEVTA2 有效且 TCCNT 计数方向递减时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[21:19]	RW	rg_oca_evta1_dec	EMEVTA1 有效且 TCCNT 计数方向递减时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平；	0x0

Bits	Access	Name	Description	Reset
			010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	
[18:16]	RW	rg_oca_evtio_dec	TRIP1 ~ TRIP3 或 TRIP9 ~ TRIP11 有效且 TCCNT 计数方向递减时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x7
[15]	RO	reserved	保留。	0x0
[14:12]	RW	rg_oca_evtb2_inc	EMEVTB2 有效且 TCCNT 计数方向递增时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[11:9]	RW	rg_oca_evtb1_inc	EMEVTB1 有效且 TCCNT 计数方向递增时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平；	0x0

Bits	Access	Name	Description	Reset
			其他：高阻态。	
[8:6]	RW	rg_oca_evta2_inc	EMEVTA2 有效且 TCCNT 计数方向递增时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[5:3]	RW	rg_oca_evta1_inc	EMEVTA1 有效且 TCCNT 计数方向递增时，APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[2:0]	RW	rg_oca_evtio_inc	TRIP1 ~ TRIP3 或 TRIP9 ~ TRIP11 有效且 TCCNT 计数方向递增时， APTx_PWMA 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x7

OC_ACT_B

OC_ACT_B 为 APTx_PWMB 输出保护寄存器。

Offset Address: 0x0314 Total Reset Value: 0x0007_0007

Bits	Access	Name	Description	Reset
[31]	RO	reserved	保留。	0x0
[30:28]	RW	rg_ocb_evtb2_dec	EMEVTB2 有效且 TCCNT 计数方向递减时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[27:25]	RW	rg_ocb_evtb1_dec	EMEVTB1 有效且 TCCNT 计数方向递减时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[24:22]	RW	rg_ocb_evta2_dec	EMEVTA2 有效且 TCCNT 计数方向递减时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[21:19]	RW	rg_ocb_evta1_dec	EMEVTA1 有效且 TCCNT 计数方向递减时，APTx_PWMB 的输出状态。 000：不做任何动作；	0x0

Bits	Access	Name	Description	Reset
			001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	
[18:16]	RW	rg_ocb_evtio_dec	TRIP1 ~ TRIP3 或 TRIP9 ~ TRIP11 有效且 TCCNT 计数方向递减时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x7
[15]	RO	reserved	保留。	0x0
[14:12]	RW	rg_ocb_evtb2_inc	EMEVTB2 有效且 TCCNT 计数方向递增时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[11:9]	RW	rg_ocb_evtb1_inc	EMEVTB1 有效且 TCCNT 计数方向递增时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平；	0x0

Bits	Access	Name	Description	Reset
			100：翻转电平； 其他：高阻态。	
[8:6]	RW	rg_ocb_evta2_inc	EMEVTA2 有效且 TCCNT 计数方向递增时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[5:3]	RW	rg_ocb_evta1_inc	EMEVTA1 有效且 TCCNT 计数方向递增时，APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x0
[2:0]	RW	rg_ocb_evtio_inc	TRIP1 ~ TRIP3 或 TRIP9 ~ TRIP11 有效且 TCCNT 计数方向递增时， APTx_PWMB 的输出状态。 000：不做任何动作； 001：低电平； 010：高电平； 011：锁定电平； 100：翻转电平； 其他：高阻态。	0x7

OC_EVT_FLAG

OC_EVT_FLAG 为故障信号和事件中断标志寄存器。

Offset Address: 0x0320 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	W1_PULSE	rg_int_clr_evt	事件中断 APTxEVTINT 的中断清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[30:28]	RO	reserved	保留。	0x0
[27]	W1_PULSE	rg_oc_clr_evtb2	EMEVTB2 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[26]	W1_PULSE	rg_oc_clr_evtb1	EMEVTB1 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[25]	W1_PULSE	rg_oc_clr_evta2	EMEVTA2 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[24]	W1_PULSE	rg_oc_clr_evta1	EMEVTA1 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[23]	RO	reserved	保留。	0x0
[22]	W1_PULSE	rg_oc_clr_evts3	TRIP11 故障信号标志的清除位，写 1 清 0。 0：写 0 无效；	0x0

Bits	Access	Name	Description	Reset
			1：写 1 有效。	
[21]	W1_PULSE	rg_oc_clr_evts2	TRIP10 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[20]	W1_PULSE	rg_oc_clr_evts1	TRIP9 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[19]	RO	reserved	保留。	0x0
[18]	W1_PULSE	rg_oc_clr_evt3	TRIP3 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[17]	W1_PULSE	rg_oc_clr_evt2	TRIP2 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[16]	W1_PULSE	rg_oc_clr_evt1	TRIP1 故障信号标志的清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[15]	RO	ro_int_flag_evt	事件中断 APTxEVTINT 的中断标志位。 0：APTxEVTINT 无效； 1：APTxEVTINT 有效。	0x0
[14:12]	RO	reserved	保留。	0x0
[11]	RO	ro_oc_flag_evtb2	EMEVTB2 故障信号标志位。 0：EMEVTB2 故障信号无效；	0x0

Bits	Access	Name	Description	Reset
			1：EMEVTB2 故障信号有效。	
[10]	RO	ro_oc_flag_evtb1	EMEVTB1 故障信号标志位。 0：EMEVTB1 故障信号无效； 1：EMEVTB1 故障信号有效。	0x0
[9]	RO	ro_oc_flag_evta2	EMEVTA2 故障信号标志位。 0：EMEVTA2 故障信号无效； 1：EMEVTA2 故障信号有效。	0x0
[8]	RO	ro_oc_flag_evta1	EMEVTA1 故障信号标志位。 0：EMEVTA1 故障信号无效； 1：EMEVTA1 故障信号有效。	0x0
[7]	RO	reserved	保留。	0x0
[6]	RO	ro_oc_flag_evts3	TRIP11 故障信号标志位。 0：TRIP11 故障信号无效； 1：TRIP11 故障信号有效。	0x0
[5]	RO	ro_oc_flag_evts2	TRIP10 故障信号标志位。 0：TRIP10 故障信号无效； 1：TRIP10 故障信号有效。	0x0
[4]	RO	ro_oc_flag_evts1	TRIP9 故障信号标志位。 0：TRIP9 故障信号无效； 1：TRIP9 故障信号有效。	0x0
[3]	RO	reserved	保留。	0x0
[2]	RO	ro_oc_flag_evt3	TRIP3 故障信号标志位。 0：TRIP3 故障信号无效； 1：TRIP3 故障信号有效。	0x0
[1]	RO	ro_oc_flag_evt2	TRIP2 故障信号标志位。 0：TRIP2 故障信号无效； 1：TRIP2 故障信号有效。	0x0

Bits	Access	Name	Description	Reset
[0]	RO	ro_oc_flag_evt1	TRIP1 故障信号标志位。 0：TRIP1 故障信号无效； 1：TRIP1 故障信号有效。	0x0

OC_PRD_CLR

OC_PRD_CLR 为周期性保护自动清除寄存器。

Offset Address: 0x0324 Total Reset Value: 0x0000_0F77

Bits	Access	Name	Description	Reset
[31:28]	RO	reserved	保留。	0x0
[27]	RW	rg_oc_clr_prden_evtb2	当 EMEVTB2 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[26]	RW	rg_oc_clr_prden_evtb1	当 EMEVTB1 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[25]	RW	rg_oc_clr_prden_evta2	当 EMEVTA2 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[24]	RW	rg_oc_clr_prden_evta1	当 EMEVTA1 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[23]	RO	reserved	保留。	0x0
[22]	RW	rg_oc_clr_prden_evts3	当 TRIP11 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[21]	RW	rg_oc_clr_prden_evts2	当 TRIP10 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[20]	RW	rg_oc_clr_prden_evts1	当 TRIP9 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[19]	RO	reserved	保留。	0x0
[18]	RW	rg_oc_clr_prden_evt3	当 TRIP3 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[17]	RW	rg_oc_clr_prden_evt2	当 TRIP2 设置成周期性保护时，选择是否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x0
[16]	RW	rg_oc_clr_prden_evt1	当 TRIP1 设置成周期性保护时，选择是	0x0

Bits	Access	Name	Description	Reset
			否在 CNT_PRD 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	
[15:12]	RO	reserved	保留。	0x0
[11]	RW	rg_oc_clr_zroen_evtb2	当 EMEVTB2 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[10]	RW	rg_oc_clr_zroen_evtb1	当 EMEVTB1 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[9]	RW	rg_oc_clr_zroen_evta2	当 EMEVTA2 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[8]	RW	rg_oc_clr_zroen_evta1	当 EMEVTA1 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[7]	RO	reserved	保留。	0x0
[6]	RW	rg_oc_clr_zroen_evts3	当 TRIP11 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。	0x1

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[5]	RW	rg_oc_clr_zroen_evts2	当 TRIP10 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[4]	RW	rg_oc_clr_zroen_evts1	当 TRIP9 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[3]	-	reserved	保留。	0x0
[2]	RW	rg_oc_clr_zroen_evt3	当 TRIP3 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[1]	RW	rg_oc_clr_zroen_evt2	当 TRIP2 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1
[0]	RW	rg_oc_clr_zroen_evt1	当 TRIP1 设置成周期性保护时，选择是否在 CNT_ZERO 有效时清除对 APTx_PWMA/APTx_PWMB 的保护。 0：不使能； 1：使能。	0x1

OC_FRC_EVT

OC_FRC_EVT 为强制产生故障信号寄存器。

Offset Address: 0x0330 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11]	RW	rg_oc_frc_evtb2	强制产生 EMEVTB2 故障信号。 0：无效； 1：有效。	0x0
[10]	RW	rg_oc_frc_evtb1	强制产生 EMEVTB1 故障信号。 0：无效； 1：有效。	0x0
[9]	RW	rg_oc_frc_evta2	强制产生 EMEVTA2 故障信号。 0：无效； 1：有效。	0x0
[8]	RW	rg_oc_frc_evta1	强制产生 EMEVTA1 故障信号。 0：无效； 1：有效。	0x0
[7]	RO	reserved	保留。	0x0
[6]	RW	rg_oc_frc_evts3	强制产生 TRIP11 故障信号。 0：无效； 1：有效。	0x0
[5]	RW	rg_oc_frc_evts2	强制产生 TRIP10 故障信号。 0：无效； 1：有效。	0x0
[4]	RW	rg_oc_frc_evts1	强制产生 TRIP9 故障信号。 0：无效； 1：有效。	0x0
[3]	RO	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[2]	RW	rg_oc_frc_evt3	强制产生 TRIP3 故障信号。 0：无效； 1：有效。	0x0
[1]	RW	rg_oc_frc_evt2	强制产生 TRIP2 故障信号。 0：无效； 1：有效。	0x0
[0]	RW	rg_oc_frc_evt1	强制产生 TRIP1 故障信号。 0：无效； 1：有效。	0x0

INT_EVT_EN

INT_EVT_EN 为事件中断使能寄存器。

Offset Address: 0x0410 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11]	RW	rg_int_en_evtb2	EMEVTB2 触发事件中断的使能位，选择故障信号 EMEVTB2 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[10]	RW	rg_int_en_evtb1	EMEVTB1 触发事件中断的使能位，选择故障信号 EMEVTB1 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[9]	RW	rg_int_en_evta2	EMEVTA2 触发事件中断的使能位，选择故障信号 EMEVTA2 有效时，是否产生事件中断 APTxEVTINT。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[8]	RW	rg_int_en_evta1	EMEVTA1 触发事件中断的使能位，选择故障信号 EMEVTA1 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[7]	RO	reserved	保留。	0x0
[6]	RW	rg_int_en_evts3	TRIP11 触发事件中断的使能位，选择故障信号 TRIP11 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[5]	RW	rg_int_en_evts2	TRIP10 触发事件中断的使能位，选择故障信号 TRIP10 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[4]	RW	rg_int_en_evts1	TRIP9 触发事件中断的使能位，选择故障信号 TRIP9 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[3]	RO	reserved	保留。	0x0
[2]	RW	rg_int_en_evt3	TRIP3 触发事件中断的使能位，选择故障信号 TRIP3 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0

Bits	Access	Name	Description	Reset
[1]	RW	rg_int_en_evt2	TRIP2 触发事件中断的使能位，选择故障信号 TRIP2 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0
[0]	RW	rg_int_en_evt1	TRIP1 触发事件中断的使能位，选择故障信号 TRIP1 有效时，是否产生事件中断 APTxEVTINT。 0：不使能； 1：使能。	0x0

INT_TMR_EN

INT_TMR_EN 为定时中断使能寄存器。

Offset Address: 0x0414 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	rg_int_en_tmr	定时中断 APTxTMRINT 的使能位。 0：不使能； 1：使能。	0x0

INT_TMR_FLAG

INT_TMR_FLAG 为定时中断标志寄存器。

Offset Address: 0x0420 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	W1_PULSE	rg_int_clr_tmr	定时中断 APTxTMRINT 的中断清除位，写 1 清 0。	0x0

Bits	Access	Name	Description	Reset
			0：写 0 无效； 1：写 1 有效。	
[15:1]	RO	reserved	保留。	0x0000
[0]	RO	ro_int_flag_tmr	定时中断 APTxTMRINT 的中断标志位。 0：APTxTMRINT 无效； 1：APTxTMRINT 有效。	0x0

INT_TMR_SEL

INT_TMR_SEL 为定时中断的触发源选择寄存器。

Offset Address: 0x0424 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3:0]	RW	rg_int_tmr_sel	定时中断 APTxTMRINT 的触发源选择。 0x1：CNT_ZERO 有效； 0x2：CNT_PRD 有效； 0x3：CNT_ZRO 或 CNT_PRD 有效； 0x4：CNT_REFA 有效且 TCCNT 计数方向递增； 0x5：CNT_REFA 有效且 TCCNT 计数方向递减； 0x6：CNT_REFB 有效且 TCCNT 计数方向递增； 0x7：CNT_REFB 有效且 TCCNT 计数方向递减； 0x8：CNT_REFC 有效且 TCCNT 计数方向递增； 0x9：CNT_REFC 有效且 TCCNT 计数方向递减； 0xA：CNT_REFD 有效且 TCCNT 计数方	0x0

Bits	Access	Name	Description	Reset
			向递增; 0xB：CNT_REFD 有效且 TCCNT 计数方向递减; 其他：保留。	

INT_PRSC_CFG

INT_PRSC_CFG 为定时中断缩减配置寄存器。

Offset Address: 0x0428 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25]	W1_PULSE	rg_int_prsc_frc	定时中断缩减计数器强制加 1。 0：写 0 无效; 1：写 1 有效。	0x0
[24]	RW	rg_int_prsc_syne n	定时中断缩减计数器的同步使能。 0：不使能; 1：使能。	0x0
[23:20]	RO	reserved	保留。	0x0
[19:16]	RW	rg_int_prsc_phs	定时中断缩减计数器的同步值。	0x0
[15:12]	RO	reserved	保留。	0x0
[11:8]	RO	ro_int_prsc_cnt	定时中断缩减计数器的计数值。	0x0
[7:4]	RO	reserved	保留。	0x0
[3:0]	RW	rg_int_prsc_prd	定时中断的缩减比例。 0x1：每发生 1 次有效的定时中断触发源，就产生定时中断; 0x2：每发生 2 次有效的定时中断触发源，就产生定时中断; 0x3：每发生 3 次有效的定时中断触发	0x1

Bits	Access	Name	Description	Reset
			源，就产生定时中断； ... 0xF：每发生 15 次有效的定时中断触发源，就产生定时中断。 其他：保留。	

CS_TMR_SELA

CS_TMR_SELA 为 APTxSOCA 的触发源选择寄存器。

Offset Address: 0x0500 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	rg_csa_en_cs	APTxSOCA 的使能位。 0：不使能； 1：使能。	0x0
[15:4]	RO	reserved	保留。	0x000
[3:0]	RW	rg_csa_tmr_sel	APTxSOCA 的触发源选择。 0x0：EMEVTA1 有效； 0x1：CNT_ZERO 有效； 0x2：CNT_PRD 有效； 0x3：CNT_ZERO 或 CNT_PRD 有效； 0x4：CNT_REFA 有效且 TCCNT 计数方向递增； 0x5：CNT_REFA 有效且 TCCNT 计数方向递减； 0x6：CNT_REFB 有效且 TCCNT 计数方向递增； 0x7：CNT_REFB 有效且 TCCNT 计数方向递减； 0x8：CNT_REFC 有效且 TCCNT 计数方	0x1

Bits	Access	Name	Description	Reset
			向递增； 0x9：CNT_REFC 有效且 TCCNT 计数方向递减； 0xA：CNT_REFD 有效且 TCCNT 计数方向递增； 0xB：CNT_REFD 有效且 TCCNT 计数方向递减； 其他：保留。	

CS_TMR_SELB

CS_TMR_SELB 为 APTxSOCB 的触发源选择寄存器。

Offset Address: 0x0504 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	rg_csb_en_cs	APTxSOCB 的使能位。 0：不使能； 1：使能。	0x0
[15:4]	RO	reserved	保留。	0x000
[3:0]	RW	rg_csb_tmr_sel	APTxSOCB 的触发源选择。 0x0：EMEVTB1 有效； 0x1：CNT_ZERO 有效； 0x2：CNT_PRD 有效； 0x3：CNT_ZERO 或 CNT_PRD 有效； 0x4：CNT_REFA 有效且 TCCNT 计数方向递增； 0x5：CNT_REFA 有效且 TCCNT 计数方向递减； 0x6：CNT_REFB 有效且 TCCNT 计数方向递增；	0x1

Bits	Access	Name	Description	Reset
			0x7：CNT_REFB 有效且 TCCNT 计数方向递减; 0x8：CNT_REFC 有效且 TCCNT 计数方向递增; 0x9：CNT_REFC 有效且 TCCNT 计数方向递减; 0xA：CNT_REFD 有效且 TCCNT 计数方向递增; 0xB：CNT_REFD 有效且 TCCNT 计数方向递减; 其他：保留。	

CS_PRSCA_CFG

CS_PRSCA_CFG 为 APTxSOCA 缩减配置寄存器。

Offset Address: 0x0508 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25]	W1_PULSE	rg_csa_prsc_frc	APTxSOCA 缩减计数器强制加 1。 0：写 0 无效; 1：写 1 有效。	0x0
[24]	RW	rg_csa_prsc_syne n	APTxSOCA 缩减计数器的同步使能。 0：不使能; 1：使能。	0x0
[23:20]	RO	reserved	保留。	0x0
[19:16]	RW	rg_csa_prsc_phs	APTxSOCA 缩减计数器的同步值。	0x0
[15:4]	RO	reserved	保留。	0x000
[3:0]	RW	rg_csa_prsc_prd	APTxSOCA 的缩减比例。 0x1：每发生 1 次有效的 APTxSOCA 的	0x1

Bits	Access	Name	Description	Reset
			触发源，就产生 APTxSOCA； 0x2：每发生 2 次有效的 APTxSOCA 的触发源，就产生 APTxSOCA； 0x3：每发生 3 次有效的 APTxSOCA 的触发源，就产生 APTxSOCA； ... 0xF：每发生 15 次有效的 APTxSOCA 的触发源，就产生 APTxSOCA。 其他：保留。	

CS_PRSCB_CFG

CS_PRSCB_CFG 为 APTxSOCB 缩减配置寄存器。

Offset Address: 0x050C Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25]	W1_PULSE	rg_csb_prsc_frc	APTxSOCB 缩减计数器强制加 1。 0：写 0 无效； 1：写 1 有效。	0x0
[24]	RW	rg_csb_prsc_syne n	APTxSOCB 缩减计数器的同步使能。 0：不使能； 1：使能。	0x0
[23:20]	RO	reserved	保留。	0x0
[19:16]	RW	rg_csb_prsc_phs	APTxSOCB 缩减计数器的同步值。	0x0
[15:4]	RO	reserved	保留。	0x000
[3:0]	RW	rg_csb_prsc_prd	APTxSOCB 的缩减比例。 0x1：每发生 1 次有效的 APTxSOCB 的触发源，就产生 APTxSOCB；	0x1

Bits	Access	Name	Description	Reset
			0x2：每发生 2 次有效的 APTxSOCB 的触发源，就产生 APTxSOCB； 0x3：每发生 3 次有效的 APTxSOCB 的触发源，就产生 APTxSOCB； ... 0xF：每发生 15 次有效的 APTxSOCB 的触发源，就产生 APTxSOCB。 其他：保留。	

CS_FLAG

CS_FLAG 为 APTxSOCA/APTxSOCB 标志寄存器。

Offset Address: 0x0510 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17]	W1_PULSE	rg_csb_clr_flag	APTxSOCB 标志清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[16]	W1_PULSE	rg_csa_clr_flag	APTxSOCA 标志清除位，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[15:2]	RO	reserved	保留。	0x0000
[1]	RO	ro_csb_flag	APTxSOCB 的标志位。 0：没有产生 APTxSOCB； 1：已经产生 APTxSOCB。	0x0
[0]	RO	ro_csa_flag	APTxSOCA 的标志位。 0：没有产生 APTxSOCA； 1：已经产生 APTxSOCA。	0x0

CS_DMA

CS_DMA 为 DMA 请求配置寄存器。

Offset Address: 0x0520 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3:2]	RW	rg_dma_sreq_sel	DMA Single 请求的触发源选择。 01：TRIGA 信号有效; 10：TRIGB 信号有效; 其他：禁止产生 DMA Single 请求。	0x0
[1:0]	RW	rg_dma_breq_sel	DMA Burst 请求的源头选择。 01：TRIGA 信号有效; 10：TRIGB 信号有效; 其他：禁止产生 DMA Burst 请求。	0x0

EM_EVTIO_PSEL

EM_EVTIO_PSEL 为事件管理极性选择寄存器 1。

Offset Address: 0x0600 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	RO	reserved	保留。	0x000
[21:20]	RW	rg_evtsys3_psel	TRIP11 的极性选择。 00：TRIP11 = sys_mem_fail; 01：TRIP11 = ~sys_mem_fail; 10：TRIP11 = 0; 11：TRIP11 = 1。	0x0
[19:18]	RW	rg_evtsys2_psel	TRIP10 的极性选择。 00：TRIP10 = clock_fail; 01：TRIP10 = ~clock_fail; 10：TRIP10 = 0;	0x0

Bits	Access	Name	Description	Reset
			11：TRIP10 = 1。	
[17:16]	RW	rg_evtsys1_psel	TRIP9 的极性选择。 00：TRIP9 = debug_mode； 01：TRIP9 = ~debug_mode； 10：TRIP9 = 0； 11：TRIP9 = 1。	0x0
[15:10]	RO	reserved	保留。	0x00
[9:8]	RW	rg_evtio5_psel	TRIP5 的极性选择。 00：TRIP5 = APT_EVTIO5； 01：TRIP5 = ~APT_EVTIO5； 10：TRIP5 = 0； 11：TRIP5 = 1。	0x0
[7:6]	RW	rg_evtio4_psel	TRIP4 的极性选择。 00：TRIP4 = APT_EVTIO4； 01：TRIP4 = ~APT_EVTIO4； 10：TRIP4 = 0； 11：TRIP4 = 1。	0x0
[5:4]	RW	rg_evtio3_psel	TRIP3 的极性选择。 00：TRIP3 = POE2； 01：TRIP3 = ~POE2； 10：TRIP3 = 0； 11：TRIP3 = 1。	0x0
[3:2]	RW	rg_evtio2_psel	TRIP2 的极性选择。 00：TRIP2 = POE1； 01：TRIP2 = ~POE1； 10：TRIP2 = 0； 11：TRIP2 = 1。	0x0
[1:0]	RW	rg_evtio1_psel	TRIP1 的极性选择。	0x0

Bits	Access	Name	Description	Reset
			00：TRIP1 = POE0； 01：TRIP1 = ~POE0； 10：TRIP1 = 0； 11：TRIP1 = 1。	

EM_EVTMP_PSEL

EM_EVTMP_PSEL 为事件管理极性选择寄存器 2。

Offset Address: 0x0604 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:10]	RW	rg_evtmp6_psel	TRIP8 的极性选择。 00：TRIP8 = APT_EVTMP6； 01：TRIP8 = ~APT_EVTMP6； 10：TRIP8 = 0； 11：TRIP8 = 1。	0x0
[9:8]	RW	rg_evtmp5_psel	TRIP7 的极性选择。 00：TRIP7 = APT_EVTMP5； 01：TRIP7 = ~APT_EVTMP5； 10：TRIP7 = 0； 11：TRIP7 = 1。	0x0
[7:6]	RW	rg_evtmp4_psel	TRIP6 的极性选择。 00：TRIP6 = APT_EVTMP4； 01：TRIP6 = ~APT_EVTMP4； 10：TRIP6 = 0； 11：TRIP6 = 1。	0x0
[5:2]	RO	reserved	保留。	0x0
[1:0]	RW	rg_evtmp1_psel	TRIP12 的极性选择。	0x0

Bits	Access	Name	Description	Reset
			00：TRIP12 = ACMP_OUT； 01：TRIP12 = ~ACMP_OUT； 10：TRIP12 = 0； 11：TRIP12 = 1。	

EM_AOR_EN

EM_AOR_EN 为事件管理组合输入选择寄存器 A。

Offset Address: 0x0608 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RO	reserved	保留。	0x0
[30:16]	RW	rg_em_a2_oren	TRIP_OR_A2 的触发源选择，每个比特对应一个信号。 bit[24]：代表 TRIP8 的使能位； bit[23]：代表 TRIP7 的使能位； bit[22]：代表 TRIP6 的使能位； bit[19]：代表 TRIP12 的使能位； bit[18]：代表 TRIP3 的使能位； bit[17]：代表 TRIP2 的使能位； bit[16]：代表 TRIP1 的使能位； 其他比特无效，应配为 0。 当相应的比特为 1 时，代表对应的信号参与逻辑或； 当相应的比特为 0 时，代表对应的信号不参与逻辑或。	0x0000
[15]	RO	reserved	保留。	0x0
[14:0]	RW	rg_em_a1_oren	TRIP_OR_A1 的触发源选择，每个比特对应一个信号。 bit[8]：代表 TRIP8 的使能位；	0x0000

Bits	Access	Name	Description	Reset
			bit[7]：代表 TRIP7 的使能位； bit[6]：代表 TRIP6 的使能位； bit[3]：代表 TRIP12 的使能位； bit[2]：代表 TRIP3 的使能位； bit[1]：代表 TRIP2 的使能位； bit[0]：代表 TRIP1 的使能位； 其他比特无效，应配为 0。 当相应的比特为 1 时，代表对应的信号参与逻辑或； 当相应的比特为 0 时，代表对应的信号不参与逻辑或。	

EM_BOR_EN

EM_BOR_EN 为事件管理组合输入选择寄存器 B。

Offset Address: 0x060C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RO	reserved	保留。	0x0
[30:16]	RW	rg_em_b2_oren	TRIP_OR_B2 的触发源选择，每个比特对应一个信号。 bit[24]：代表 TRIP8 的使能位； bit[23]：代表 TRIP7 的使能位； bit[22]：代表 TRIP6 的使能位； bit[19]：代表 TRIP12 的使能位； bit[18]：代表 TRIP3 的使能位； bit[17]：代表 TRIP2 的使能位； bit[16]：代表 TRIP1 的使能位； 其他比特无效，应配为 0。 当相应的比特为 1 时，代表对应的信号参与逻辑或；	0x0000

Bits	Access	Name	Description	Reset
			当相应的比特为 0 时，代表对应的信号不参与逻辑或。	
[15]	RO	reserved	保留。	0x0
[14:0]	RW	rg_em_b1_oren	TRIP_OR_B1 的触发源选择，每个比特对应一个信号。 bit[8]：代表 TRIP8 的使能位； bit[7]：代表 TRIP7 的使能位； bit[6]：代表 TRIP6 的使能位； bit[3]：代表 TRIP12 的使能位； bit[2]：代表 TRIP3 的使能位； bit[1]：代表 TRIP2 的使能位； bit[0]：代表 TRIP1 的使能位； 其他比特无效，应配为 0。 当相应的比特为 1 时，代表对应的信号参与逻辑或； 当相应的比特为 0 时，代表对应的信号不参与逻辑或。	0x0000

EM_MRG_SEL

EM_MRG_SEL 为事件管理输入选择寄存器。

Offset Address: 0x0610 Total Reset Value: 0x5151_0000

Bits	Access	Name	Description	Reset
[31]	RO	reserved	保留。	0x0
[30:28]	RW	rg_evtb2t_sel	EMEVTB2_TEMP 有效（高电平有效）的条件选择。 0x1：EMEVTB1_SRC 为高电平； 0x2：EMEVTB1_SRC 为高电平，且 EMEVTB2_SRC 为低电平； 0x3：EMEVTB1_SRC 为高电平，且	0x5

Bits	Access	Name	Description	Reset
			EMEVTB2_SRC 为高电平； 0x4：EMEVTB1_SRC 为低电平，且 EMEVTB2_SRC 为高电平； 0x5：EMEVTB2_SRC 为高电平； 其他：EMEVTB2_TEMP 输出低电平。	
[27]	RO	reserved	保留。	0x0
[26:24]	RW	rg_evtb1t_sel	EMEVTB1_TEMP 有效（高电平有效）的条件选择。 0x1：EMEVTB1_SRC 为高电平； 0x2：EMEVTB1_SRC 为高电平，且 EMEVTB2_SRC 为低电平； 0x3：EMEVTB1_SRC 为高电平，且 EMEVTB2_SRC 为高电平； 0x4：EMEVTB1_SRC 为低电平，且 EMEVTB2_SRC 为高电平； 0x5：EMEVTB2_SRC 为高电平； 其他：EMEVTB1_TEMP 输出低电平。	0x1
[23]	RO	reserved	保留。	0x0
[22:20]	RW	rg_evta2t_sel	EMEVTA2_TEMP 有效（高电平有效）的条件选择。 0x1：EMEVTA1_SRC 为高电平； 0x2：EMEVTA1_SRC 为高电平，且 EMEVTA2_SRC 为低电平； 0x3：EMEVTA1_SRC 为高电平，且 EMEVTA2_SRC 为高电平； 0x4：EMEVTA1_SRC 为低电平，且 EMEVTA2_SRC 为高电平； 0x5：EMEVTA2_SRC 为高电平； 其他：EMEVTA2_TEMP 输出低电平。	0x5
[19]	RO	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[18:16]	RW	rg_evta1t_sel	EMEVTA1_TEMP 有效（高电平有效）的条件选择。 0x1：EMEVTA1_SRC 为高电平； 0x2：EMEVTA1_SRC 为高电平，且 EMEVTA2_SRC 为低电平； 0x3：EMEVTA1_SRC 为高电平，且 EMEVTA2_SRC 为高电平； 0x4：EMEVTA1_SRC 为低电平，且 EMEVTA2_SRC 为高电平； 0x5：EMEVTA2_SRC 为高电平； 其他：EMEVTA1_TEMP 输出低电平。	0x1
[15:12]	RW	rg_em_b2_sel	EMEVTB2_SRC 触发源选择。 0x0：TRIP1； 0x1：TRIP2； 0x2：TRIP3； 0x3：TRIP12； 0x6：TRIP6； 0x7：TRIP7； 0x8：TRIP8； 0xF：TRIP_OR_B2； 其他：EMEVTB2_SRC 输出低电平。	0x0
[11:8]	RW	rg_em_b1_sel	EMEVTB1_SRC 触发源选择。 0x0：TRIP1； 0x1：TRIP2； 0x2：TRIP3； 0x3：TRIP12； 0x6：TRIP6； 0x7：TRIP7； 0x8：TRIP8；	0x0

Bits	Access	Name	Description	Reset
			0xF：TRIP_OR_B1； 其他：EMEVTB1_SRC 输出低电平。	
[7:4]	RW	rg_em_a2_sel	EMEVTA2_SRC 触发源选择。 0x0：TRIP1； 0x1：TRIP2； 0x2：TRIP3； 0x3：TRIP12； 0x6：TRIP6； 0x7：TRIP7； 0x8：TRIP8； 0xF：TRIP_OR_A2； 其他：EMEVTA2_SRC 输出低电平。	0x0
[3:0]	RW	rg_em_a1_sel	EMEVTA1_SRC 触发源选择。 0x0：TRIP1； 0x1：TRIP2； 0x2：TRIP3； 0x3：TRIP12； 0x6：TRIP6； 0x7：TRIP7； 0x8：TRIP8； 0xF：TRIP_OR_A1； 其他：EMEVTA1_SRC 输出低电平。	0x0

EM_OUT_SEL

EM_OUT_SEL 为事件管理输出选择寄存器。

Offset Address: 0x0614 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000

Bits	Access	Name	Description	Reset
[5:4]	RW	rg_evtfilt_sel	EMEVTFILT 的触发源选择。 00：EMEVTA1_TEMP； 01：EMEVTA2_TEMP； 10：EMEVTB1_TEMP； 11：EMEVTB2_TEMP。	0x0
[3]	RW	rg_evtb2_sel	EMEVTB2 的触发源选择。 0：EMEVTB2_TEMP； 1：EMEVTFILT。	0x0
[2]	RW	rg_evtb1_sel	EMEVTB1 的触发源选择。 0：EMEVTB1_TEMP； 1：EMEVTFILT。	0x0
[1]	RW	rg_evta2_sel	EMEVTA2 的触发源选择。 0：EMEVTA2_TEMP； 1：EMEVTFILT。	0x0
[0]	RW	rg_evta1_sel	EMEVTA1 的触发源选择。 0：EMEVTA1_TEMP； 1：EMEVTFILT。	0x0

SYNI_CFG

SYNI_CFG 为同步输入配置寄存器。

Offset Address: 0x0700 Total Reset Value: 0x0000_000F

Bits	Access	Name	Description	Reset
[31:21]	RO	reserved	保留。	0x000
[20]	W1_PULSE	rg_syni_clr	APTxSYNCI 的标志清除位，写 1 清 0。 0：写 0 无效； 1：写 1 清除 APTxSYNCI 的标志位 ro_syni_flag。	0x0

Bits	Access	Name	Description	Reset
[19:17]	RO	reserved	保留。	0x0
[16]	RO	ro_syni_flag	APTxSYNCI 的标志位。 0：APTxSYNCI 无效； 1：APTxSYNCI 有效。	0x0
[15:4]	RO	reserved	保留。	0x000
[3:0]	RW	rg_syni_sel	APTxSYNCI 的触发源选择。 0x0：APT0SYNCO； 0x1：APT1SYNCO； 0x2：APT2SYNCO； 0x3：APT3SYNCO； 0x9：CAPM0SYNCO； 0xA：CAPM1SYNCO； 0xB：CAPM2SYNCO； 0xC：TRIP4； 0xD：TRIP5； 其他：禁止产生 APTxSYNCI。	0xF

SYNCNT_CFG

SYNCNT_CFG 为计数器同步信号配置寄存器。

Offset Address: 0x0704 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RW	rg_syncnt_synien	当 APTxSYNCI 有效时，选择是否产生 APTCNTSYNC。 0：不使能； 1：使能。	0x0
[1]	RW	rg_syncnt_b1en	当 EMEVTB1 有效时，选择是否产生	0x0

Bits	Access	Name	Description	Reset
			APTCNTSYNC。 0：不使能； 1：使能。	
[0]	RW	rg_syncnt_a1en	当 EMEVTA1 有效时，选择是否产生 APTCNTSYNC。 0：不使能； 1：使能。	0x0

SYNO_CFG

SYNO_CFG 为同步输出配置寄存器。

Offset Address: 0x0708 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:11]	RO	reserved	保留。	0x000000
[10]	W1_PULSE	rg_latset_otsyn	控制同步输出单次锁存器置位。 0：写 0 无效； 1：写 1 有效。	0x0
[9]	RW	rg_latset_sel	当选择单次同步输出模式，选择同步输出单次锁存器置位的条件。 0：向 SYNO_CFG[rg_latset_otsyn]写 1 时置位； 1：向 GLB_LOAD[rg_latset_otgld]写 1 时置位。	0x0
[8]	RW	rg_mode_syno	同步输出的模式选择。 0：单次同步输出； 1：不限次数同步输出。	0x0
[7]	RW	rg_syno_refden	当 CNT_REFD 有效时，选择是否产生 APTxSYNCO。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[6]	RW	rg_syno_refcen	当 CNT_REFC 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x0
[5]	RW	rg_syno_refben	当 CNT_REFB 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x0
[4]	RO	reserved	保留。	0x0
[3]	RW	rg_syno_b1en	当 EMEVTB1 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x0
[2]	RW	rg_syno_a1en	当 EMEVTA1 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x0
[1]	RW	rg_syno_prden	当 CNT_PRD 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x0
[0]	RW	rg_syno_zroen	当 CNT_ZERO 有效时，选择是否产生 APTxSYNCO。 0：不使能； 1：使能。	0x1

GLB_LOAD

GLB_LOAD 为全局加载配置寄存器。

Offset Address: 0x0710 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	W1_PULSE	rg_latset_otgld	控制全局加载单次锁存器置位。 0：写 0 无效； 1：写 1 有效。	0x0
[15:13]	RO	reserved	保留。	0x0
[12]	RW	rg_mode_gld	全局加载的模式选择。 0：单次加载； 1：不限次数加载。	0x0
[11:8]	RW	rg_gld_prsc_prd	全局加载信号缩减计数器的缩减比例。 0x1：每发生 1 次有效的全局加载触发信号，就产生 glb_ld； 0x2：每发生 2 次有效的全局加载触发信号，就产生 glb_ld； 0x3：每发生 3 次有效的全局加载触发信号，就产生 glb_ld； ... 0xF：每发生 15 次有效的全局加载触发信号，就产生 glb_ld。 其他：保留。	0x0
[7:3]	RO	reserved	保留。	0x00
[2]	RW	rg_gld_cntsynen	使能 APTCNTSYNC 作为全局加载触发信号。 0：不使能； 1：使能。	0x0
[1]	RW	rg_gld_prden	使能 CNT_PRD 作为全局加载触发信号。	0x0

Bits	Access	Name	Description	Reset
			0：不使能； 1：使能。	
[0]	RW	rg_gld_zroen	使能 CNT_ZERO 作为全局加载触发信号。 0：不使能； 1：使能。	0x1

LOAD_STS

LOAD_STS 为缓存状态寄存器。

Offset Address: 0x0720 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:19]	RO	reserved	保留。	0x0000
[18]	RO	tc_mwd_act_ld_sts	当使能了 TC_MWD_ACT 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[17]	RO	tc_mwdrefb_ld_sts	当使能了 TC_MWDREFB 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[16]	RO	tc_mwdrefa_ld_sts	当使能了 TC_MWDREFA 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[15]	RO	reserved	保留。	0x0
[14]	RO	dg_cfg_ld_sts	当使能了 DG_CFG 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空；	0x0

Bits	Access	Name	Description	Reset
			1：缓存寄存器满。	
[13]	RO	dg_fed_ld_sts	当使能了 DG_FEG 寄存器的缓存，指示 DG_FEG 寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[12]	RO	dg_red_ld_sts	当使能了 DG_REG 寄存器的缓存，指示 DG_REG 寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[11]	RO	reserved	保留。	0x0
[10]	RO	pg_out_frc_ld_sts	当使能了 PG_OUT_FRC 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[9]	RO	pg_act_b_ld_sts	当使能了 PG_ACT_B 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[8]	RO	pg_act_a_ld_sts	当使能了 PG_ACT_A 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[7:5]	RO	reserved	保留。	0x0
[4]	RO	tc_refd_ld_sts	当使能了 TC_REFD 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[3]	RO	tc_refc_ld_sts	当使能了 TC_REFC 寄存器的缓存，指示	0x0

Bits	Access	Name	Description	Reset
			寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	
[2]	RO	tc_refb_ld_sts	当使能了 TC_REFB 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[1]	RO	tc_refa_ld_sts	当使能了 TC_REFA 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[0]	RO	tc_prd_ld_sts	当使能了 TC_PRD 寄存器的缓存，指示寄存器的缓存状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0

SYN_FRC

SYN_FRC 为软件强制寄存器。

Offset Address: 0x0730 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28]	W1_PULSE	rg_synpg_frc	强制产生一个 PG_ACT_A/PG_ACT_B 寄存器的独立加载的触发信号。 0：写 0 无效； 1：写 1 有效。	0x0
[27:25]	RO	reserved	保留。	0x0
[24]	W1_PULSE	rg_int_syn_frc	强制定时中断缩减计数器更新成 INT_PRSC_CFGrg_int_prsc_phs]中的	0x0

Bits	Access	Name	Description	Reset
			值。 0：写 0 无效; 1：写 1 有效。	
[23:22]	RO	reserved	保留。	0x0
[21]	W1_PULSE	rg_csb_syn_frc	强制产生一次 APTxSOCB 的触发源。 0：写 0 无效; 1：写 1 有效。	0x0
[20]	W1_PULSE	rg_csa_syn_frc	强制产生一次 APTxSOCA 的触发源。 0：写 0 无效; 1：写 1 有效。	0x0
[19:13]	RO	reserved	保留。	0x00
[12]	W1_PULSE	rg_gld_frc	强制产生一次全局加载信号的触发信号。 0：写 0 无效; 1：写 1 有效。	0x0
[11:5]	RO	reserved	保留。	0x00
[4]	W1_PULSE	rg_syno_frc	强制产生一次 APTxSYNCO 的触发源。 0：写 0 无效; 1：写 1 有效。	0x0
[3:1]	RO	reserved	保留。	0x0
[0]	W1_PULSE	rg_syncnt_frc	强制产生一个 APTCNTSYNC 信号。 0：写 0 无效; 1：写 1 有效。	0x0

15 通用 PWM 定时器(GPT)

15.1 概述

GPT（General PWM Timer）通用 PWM（Pulse Width Modulation）定时器可以产生 PWM 波。

15.2 特性

15.2.1 特性列表

MCU 提供 4 个 GPT 模块，每个 GPT 模块可以输出一路 PWM 波，对于每个 GPT：

- 支持 12 位分频计数器 DIVCNT、16 位时基计数器 TCCNT。
- 占空比 0~100%可配。
- 支持输出无限个数或有限个数 PWM 波。
- 支持在 TCCNT 计数到计数周期值时产生周期中断、DMA 请求和触发 ADC 采样。
- 支持在有限个数 PWM 输出结束时产生通道输出完成中断、DMA 请求和触发 ADC 采样。

15.2.2 关键指标参数

- PWM 频率

$$PWM\text{频率}=\frac{\text{工作时钟频率}}{(\text{分频系数}+1)*(\text{计数周期值}+1)}$$

$$F_{pwm}=\frac{F_{GPTCLK}}{(GPT_TC_DIV[rg_div_fac]+1)*(GPT_TC_PRD[rg_cnt_prd]+1)}$$

举例：

工作时钟频率 F_{GPTCLK}=150MHz，分频系数 GPT_TC_DIV[rg_div_fac]配置为 9，计数周期值 GPT_TC_PRD[rg_cnt_prd]配置为 999，得到的 PWM 频率为 $\frac{150MHz}{10*1000}=15kHz$。

- PWM 周期

  PWM 周期 = 1 / PWM 频率

- PWM 分辨率

  PWM 分辨率 = 分频时钟周期。

  $PWM分辨率=工作时钟周期*（分频系数+1）=T_{GPTCLK}*(GPT_TC_DIV[rg_div_fac]+1)$

  举例：当工作时钟频率为 150MHz 时，如果不使用分频（GPT_TC_DIV[rg_div_fac]=0），则产生的 PWM 分辨率为 6.67ns。如果 10 倍分频（GPT_TC_DIV[rg_div_fac]=9），则产生的 PWM 分辨率为 66.7ns。

- 占空比调节范围

  占空比调节范围为 0~100%。

15.3 功能描述

15.3.1 PWM 周期

GPT 内部有一个 12bit 分频计数器 DIVCNT 和 16bit 时基计数器 TCCNT 用来控制 PWM 周期。

DIVCNT 可以对工作时钟（GPTCLK）1 ~ 4096 任意分频，通过 GPT_TC_DIV[rg_div_fac]配置，配置范围 0 ~ 4095。

TCCNT 对分频后的时钟进行增计数，计数到 GPT_TC_PRD[rg_cnt_prd]配置值后清零，配置范围 0 ~ 65535。

15.3.2 PWM 占空比

GPT 可以通过 GPT_TC_REFA0 和 GPT_TC_REFB0 设置两个计数参考值，当 TCCNT 等于配置的计数参考值时，输出波形发生改变，波形动作由 GPT_PG_ACT0 控制，波形动作有：无影响、置为低电平、置为高电平和翻转电平 4 个动作。

通过 GPT_TC_REFA0、GPT_TC_REFB0、GPT_PG_ACT0 可以得到 0 ~ 100%可调的占空比。

须知

当 GPT_TC_REFA0[rg_cnt_refa0]和 GPT_TC_REFB0[rg_cnt_refb0]相同时，如果计数器等于参考点且 GPT_PG_ACT0[rg_pg_act0_refa0]配置动作不为“无影响”，则波形动作由 GPT_PG_ACT0[rg_pg_act0_refa0]的配置决定。如果计数器等于参考点且 GPT_PG_ACT0[rg_pg_act0_refa0]配置动作为“无影响”，则波形动作由 GPT_PG_ACT0[rg_pg_act0_refb0]的配置决定。

15.3.3 PWM 个数

GPT 支持输出无限个数 PWM 波和有限个数 PWM 波，由 GPT_PWM0_CFG[rg_pwm0_keep]控制。当 GPT_PWM0_CFG[rg_pwm0_keep]为 0 时，输出有限个数 PWM 波，个数由 GPT_PWM0_CFG[rg_pwm0_num]配置，配置范围为 0~1023 个。当 GPT_PWM0_CFG[rg_pwm0_keep]为 1 时，输出无限个数 PWM 波。

支持在输出 PWM 期间实时更改 PWM 输出个数配置，更改后会在当前计数周期结束后生效，然后按照新的配置输出 PWM。例如：旧的配置为输出 10 个 PWM 波，在输出第 3 个 PWM 波期间修改新的配置为输出 5 个 PWM 波，则会在第 3 个 PWM 完整输出后，再输出 5 个 PWM 波，一共输出 8 个 PWM 波。

软件可通过读取 GPT_PWM0_STA[ro_pwm0_run_sta]确定是否正在输出 PWM 波，通过读取 GPT_PWM0_STA[ro_pwm0_num_sta]确定正在输出第几个 PWM 波。

15.3.4 GPT 的使能

通过 GPT_EN[rg_gpt_en]配置 GPT 使能，使能之后计数器开始计数，GPT 开始输出 PWM 波。当通道输出配置个数的 PWM 波后，[rg_gpt_en]会自动禁能。

15.3.5 中断功能

GPT 有两个中断源：周期中断和通道输出完成中断。

周期中断：通过 GPT_INT_EN[rg_prd_int_en]使能，当计数器等于 GPT_TC_PRD[rg_cnt_prd]配置的值后置位，软件可通过读 GPT_INT_FLAG[ro_prd_int_flag]获取周期中断状态，通过向

GPT_INT_FLAG[rg_prd_int_clr]写 1 清除周期中断。软件可通过向 GPT_INT_INJ[rg_prd_int_inj]写 1 注入周期中断。

通道输出完成中断：通过 GPT_INT_EN[rg_pwm0_int_en]使能，当通道输出配置个数的 PWM 波后置位，软件可通过读 GPT_INT_FLAG[ro_pwm0_int_flag]获取通道输出完成中断状态，通过向 GPT_INT_FLAG[rg_pwm0_int_clr]写 1 清除通道输出完成中断。软件可通过向 GPT_INT_INJ[rg_pwm0_int_inj]写 1 注入通道输出完成中断。

15.3.6 DMA 请求

GPT 可以产生 DMA Burst 请求，两个触发条件：计数周期结束和通道输出完成配置个数的 PWM 波，分别由 GPT_SOCDR_EN[rg_dbr_prd_en]和 GPT_SOCDR_EN[rg_dbr_pwm0_en]使能。

GPT 可以产生 DMA Single 请求，两个触发条件：计数周期结束和通道输出完成配置个数的 PWM 波，分别由 GPT_SOCDR_EN[rg_dsr_prd_en]和 GPT_SOCDR_EN[rg_dsr_pwm0_en]使能。

15.3.7 触发 ADC 采样

GPT 可以产生触发 ADC 采样的信号（SOC 信号），两个触发条件：计数周期结束和通道输出完成配置个数的 PWM 波，分别由 GPT_SOCDR_EN[rg_soc_prd_en]和 GPT_SOCDR_EN[rg_soc_pwm0_en]使能。

15.3.8 寄存器的缓存加载

GPT_TC_PRD、GPT_TC_REFA0、GPT_TC_REFB0 和 GPT_PG_ACT0 寄存器支持立即加载和缓存加载两种模式，由 GPT_BUF_LOAD_EN[rg_buf_load_en]使能缓存加载。在立即加载模式下，对以上四个寄存器的更新会立即生效。在缓存加载模式下，对以上四个寄存器的更新会在当前计数周期结束后生效。

GPT_PWM0_CFG 寄存器只有缓存加载模式，GPT 运行期间对该寄存器的更新只会在当前计数周期结束后生效。

在缓存加载模式下，通过读 GPT_LOAD_STS 相关位域可以获取相关寄存器的缓存加载状态，例如：当 GPT_LOAD_STS[tc_prd_ld_sts]为 1 时，表示上次 GPT_TC_PRD 的更新值还未生效；当 GPT_LOAD_STS[tc_prd_ld_sts]为 0 时，表示上次 GPT_TC_PRD 的更新值已经生效。在立即加载模式下，GPT_LOAD_STS 的 pg_act0_ld_sts、tc_refb0_ld_sts、tc_refa0_ld_sts、tc_prd_ld_sts 位域固定为 0。

须知

立即加载模式下，如果计数周期寄存器的更新值大于当前计数器的计数值，计数器会继续计数至更新值后清零，如果计数周期寄存器的更新值小于等于当前计数器的计数值，计数器会立即清零。

当对 GPT_TC_PRD、GPT_TC_REFA0、GPT_TC_REFB0、GPT_PG_ACT0 寄存器使用缓存加载模式时，在初始化 GPT 配置时，要先使能缓存加载，再配置 GPT_TC_PRD、GPT_TC_REFA0、GPT_TC_REFB0、GPT_PG_ACT0 寄存器。

15.4 工作方式

GPT 输出配置流程如下：

步骤 1 通过计算得到需要的周期和占空比。

步骤 2 如果使用缓存加载模式，则需要先配置 GPT_BUF_LOAD_EN[rg_buf_load_en]=1。

步骤 3 根据希望输出的 PWM 的频率、占空比配置 GPT_TC_DIV、GPT_TC_PRD、GPT_TC_REFA0、GPT_TC_REFB0、GPT_PG_ACT0 寄存器。

步骤 4 根据希望输出的 PWM 个数配置 GPT_PWM0_CFG。

步骤 5 如果要使能中断，则配置 GPT_INT_EN 寄存器的相关位域。

步骤 6 如果要使能 DMA 请求和触发 ADC 采样，则配置 GPT_SOCDR_EN 寄存器的相关位域。

步骤 7 配置 GPT_EN[rg_gpt_en] = 1，使能 GPT。

----结束

15.5 GPTx 寄存器概览

GPTx（x=0，1，2，3）寄存器概览如表 15-1 所示。

表15-1 GPTx 寄存器概览（GPTx 的基址是 0x1470_x000）

偏移地址	名称	描述	页码
0x0000	GPT_VER_INFO	GPTx 的版本信息寄存器	GPT_VER_INFO

偏移地址	名称	描述	页码
0x0010	GPT_TC_DIV	GPTx 的分频系数寄存器	GPT_TC_DIV
0x0014	GPT_TC_PRD	GPTx 的计数周期寄存器	GPT_TC_PRD
0x0018	GPT_TC_REFA0	GPTx 的计数参考值 A0 寄存器	GPT_TC_REFA0
0x001C	GPT_TC_REFB0	GPTx 的计数参考值 B0 寄存器	GPT_TC_REFB0
0x0030	GPT_TC_STS	GPTx 的计数状态寄存器	GPT_TC_STS
0x0100	GPT_PG_ACT0	GPTx 的通道 0 动作配置寄存器	GPT_PG_ACT0
0x0200	GPT_INT_EN	GPTx 的中断使能寄存器	GPT_INT_EN
0x0204	GPT_INT_FLAG	GPTx 的中断标志寄存器	GPT_INT_FLAG
0x0208	GPT_INT_INJ	GPTx 的中断注入寄存器	GPT_INT_INJ
0x0300	GPT_SOCDR_EN	GPTx 的 SOC/DMA 请求使能寄存器	GPT_SOCDR_EN
0x0400	GPT_PWM0_CFG	GPTx 的通道 0 配置寄存器	GPT_PWM0_CFG
0x040C	GPT_EN	GPTx 的使能寄存器	GPT_EN
0x0410	GPT_PWM0_STA	GPTx 的通道 0 状态寄存器	GPT_PWM0_STA
0x0500	GPT_BUF_LOAD_EN	GPTx 的缓存加载使能寄存器	GPT_BUF_LOAD_EN
0x0504	GPT_LOAD_STS	GPTx 的缓存加载状态寄存器	GPT_LOAD_STS

15.6 GPTx 寄存器描述

GPT_VER_INFO

GPT_VER_INFO 为版本信息寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0010

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7:4]	RO	main_version	主版本号	0x1
[3:0]	RO	sub_version	子版本号	0x0

GPT_TC_DIV

GPT_TC_DIV 为分频系数寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	-	reserved	保留。	0x00000
[11:0]	RW	rg_div_fac	分频系数。 分频倍数 = 分频系数 +1。	0x000

GPT_TC_PRD

GPT_TC_PRD 为计数周期寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	rg_cnt_prd	时基计数器（TCCNT）的计数周期。	0x0000

GPT_TC_REFA0

GPT_TC_REFA0 为计数参考值 A0 寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[15:0]	RW	rg_cnt_refa0	时基计数器（TCCNT）的参考值 A0，应当小于或等于计数周期。	0x0000

GPT_TC_REFB0

GPT_TC_REFB0 为计数参考值 B0 寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	rg_cnt_refb0	时基计数器（TCCNT）的参考值 B0，应当小于或等于计数周期。	0x0000

GPT_TC_STS

GPT_TC_STS 为计数状态寄存器。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27:16]	RO	ro_div_cnt	分频计数器（DIVCNT）的当前计数值。	0x000
[15:0]	RO	ro_cnt_val	时基计数器（TCCNT）的当前计数值。	0x0000

GPT_PG_ACT0

GPT_PG_ACT0 为通道 0 动作配置寄存器。

Offset Address: 0x0100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5:4]	RW	rg_pg_act0_refb0	时基计数器（TCCNT）等于参考值 B0 时，通道 0 的 PWM 输出动作。	0x0

Bits	Access	Name	Description	Reset
			00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	
[3:2]	-	reserved	保留。	0x0
[1:0]	RW	rg_pg_act0_refa0	时基计数器（TCCNT）等于参考值 A0 时，通道 0 的 PWM 输出动作。 00：无影响； 01：置为低电平； 10：置为高电平； 11：翻转电平。	0x0

GPT_INT_EN

GPT_INT_EN 为中断使能寄存器。

Offset Address: 0x0200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	RW	rg_pwm0_int_en	通道 0 输出完成中断的使能。 0：禁能； 1：使能。	0x0
[3:1]	-	reserved	保留。	0x0
[0]	RW	rg_prd_int_en	周期中断的使能。 0：禁能； 1：使能。	0x0

GPT_INT_FLAG

GPT_INT_FLAG 为中断标志寄存器。

Offset Address: 0x0204 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:21]	-	reserved	保留。	0x000
[20]	W1_PULSE	rg_pwm0_int_clr	通道 0 输出完成中断的中断清除，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[19:17]	-	reserved	保留。	0x0
[16]	W1_PULSE	rg_prd_int_clr	周期中断的中断清除，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0
[15:5]	-	reserved	保留。	0x000
[4]	RO	ro_pwm0_int_flag	通道 0 输出完成中断的中断标志。 0：通道 0 输出完成中断标志无效； 1：通道 0 输出完成中断标志有效。	0x0
[3:1]	-	reserved	保留。	0x0
[0]	RO	ro_prd_int_flag	周期中断的中断标志。 0：周期中断标志无效； 1：周期中断标志有效。	0x0

GPT_INT_INJ

GPT_INT_INJ 为中断注入寄存器。

Offset Address: 0x0208 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:5]	-	reserved	保留。	0x0000000
[4]	W1_PULSE	rg_pwm0_int_inj	软件注入通道 0 输出完成中断，写 1 清 0。	0x0

Bits	Access	Name	Description	Reset
			0：写 0 无效； 1：写 1 有效。	
[3:1]	-	reserved	保留。	0x0
[0]	W1_PULSE	rg_prd_int_inj	软件注入周期中断，写 1 清 0。 0：写 0 无效； 1：写 1 有效。	0x0

GPT_SOCDR_EN

GPT_SOCDR_EN 为 SOC/DMA 请求使能寄存器。

Offset Address: 0x0300 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	-	reserved	保留。	0x000000
[9]	RW	rg_dbr_pwm0_en	通道 0 输出完成产生 DMA Burst request 信号的使能。 0：通道 0 输出完成时禁止产生 DMA Burst request； 1：通道 0 输出完成时产生 DMA Burst request 信号。	0x0
[8]	RW	rg_dbr_prd_en	计数周期结束产生 DMA Burst request 信号的使能。 0：计数周期结束时禁止产生 DMA Burst request； 1：计数周期结束时产生 DMA Burst request 信号。	0x0
[7:6]	-	reserved	保留。	0x0
[5]	RW	rg_dsr_pwm0_en	通道 0 输出完成产生 DMA Single request 信号的使能。 0：通道 0 输出完成时禁止产生 DMA	0x0

Bits	Access	Name	Description	Reset
			Single request; 1：通道 0 输出完成时产生 DMA Single request 信号。	
[4]	RW	rg_dsr_prd_en	计数周期结束产生 DMA Single request 信号的使能。 0：计数周期结束时禁止产生 DMA Single request; 1：计数周期结束时产生 DMA Single request 信号。	0x0
[3:2]	-	reserved	保留。	0x0
[1]	RW	rg_soc_pwm0_en	通道 0 输出完成产生 SOC 信号的使能。 0：通道 0 输出完成时禁止产生 SOC 信号; 1：通道 0 输出完成时产生 SOC 信号。	0x0
[0]	RW	rg_soc_prd_en	计数周期结束产生 SOC 信号的使能。 0：计数周期结束时禁止产生 SOC 信号; 1：计数周期结束时产生 SOC 信号。	0x0

GPT_PWM0_CFG

GPT_PWM0_CFG 为通道 0 配置寄存器。

Offset Address: 0x0400 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31]	RW	rg_pwm0_keep	通道 0 输出 PWM 的模式。 0：通道 0 输出有限个数 PWM，输出个数由 GPT_PWM0_CFG[rg_pwm0_num]配置; 1：通道 0 输出无限个数 PWM。	0x1
[30:10]	-	reserved	保留。	0x000000

Bits	Access	Name	Description	Reset
[9:0]	RW	rg_pwm0_num	通道 0 输出 PWM 的个数。 仅当 GPT_PWM0_CFG[rg_pwm0_keep]为 0 时有效。	0x000

GPT_EN

GPT_EN 为 GPT 使能寄存器。

Offset Address: 0x040C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RWSC	rg_gpt_en	GPT 使能控制。 0：GPT 通道关闭; 1：GPT 通道使能。	0x0

GPT_PWM0_STA

GPT_PWM0_STA 为通道 0 状态寄存器。

Offset Address: 0x0410 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RO	ro_pwm0_run_sta	通道 0 的输出状态。 0：表示目前通道 0 不在输出 PWM 波; 1：表示目前通道 0 正在输出 PWM 波。	0x0
[30:10]	-	reserved	保留。	0x000000
[9:0]	RO	ro_pwm0_num_sta	通道 0 输出 PWM 的个数。 当 GPT_PWM0_CFG[rg_pwm0_keep]为 1 时，一直为 0; 当 GPT_PWM0_CFG[rg_pwm0_keep]为 0 时， GPT_PWM0_STA[ro_pwm0_num_sta]表	0x000

Bits	Access	Name	Description	Reset
			示通道 0 正在输出第几个 PWM 波。	

GPT_BUF_LOAD_EN

GPT_BUF_LOAD_EN 为缓存加载使能寄存器。

Offset Address: 0x0500 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	rg_buf_load_en	支持缓存功能的寄存器的缓存加载使能。 0：禁能； 1：使能。	0x0

GPT_LOAD_STS

GPT_LOAD_STS 为缓存加载状态寄存器。

Offset Address: 0x0504 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12]	RO	pwm0_cfg_ld_sts	当使能了缓存加载功能，指示 GPT_PWM0_CFG 寄存器缓存的状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[11:9]	-	reserved	保留。	0x0
[8]	RO	pg_act0_ld_sts	当使能了缓存加载功能，指示 GPT_PG_ACT0 寄存器缓存的状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[7:3]	-	reserved	保留。	0x00

Bits	Access	Name	Description	Reset
[2]	RO	tc_refb0_ld_sts	当使能了缓存加载功能，指示 GPT_TC_REFB0 寄存器缓存的状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[1]	RO	tc_refa0_ld_sts	当使能了缓存加载功能，指示 GPT_TC_REFA0 寄存器缓存的状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0
[0]	RO	tc_prd_ld_sts	当使能了缓存加载功能，指示 GPT_TC_PRD 寄存器缓存的状态。 0：缓存寄存器空； 1：缓存寄存器满。	0x0

16 基本定时器(TIMER)

16.1 概述

TIMER 模块主要实现定时、计数功能，可以供程序用作定时和计数。MCU 提供 4 个 TIMER。

16.2 特性

TIMER 具有以下特点：

- 支持将 clk_timer 在 TIMER 内部进行 3 档预分频：1 分频、16 分频或 256 分频。
- 支持 32bit/16bit 减计数。
- 支持 3 种计数模式：自由运行模式、周期模式和单次计数模式。
- 支持随时读取当前的计数值。
- 支持产生 DMA 请求、产生触发 ADC 启动采样的信号（SOC 信号，Start Of Convertion）。
- 支持当计数值减到 0 时会产生 TIMER 定时中断。
- 支持当 DMA 请求溢出时产生 DMA 请求溢出中断。

16.3 功能描述

16.3.1 TIMER 计数时钟的分频

TIMER 支持可编程 3 档预分频，通过配置 TIMERx_CONTROL[timerpre]位域，将计数时钟 clk_timer 在 TIMER 内部再次进行 1 分频、16 分频或 256 分频。

16.3.2 TIMER 的计数模式

TIMER 支持 32bit/16bit 减计数，通过配置 TIMERx_CONTROL[timersize]位域进行选择。

TIMER 有以下 3 种计数模式，通过 TIMERx_CONTROL[oneshot]和 TIMERx_CONTROL[timermode]寄存器配置：

- 自由运行模式

  TIMER 持续计数。当 32bit 计数时，TIMER 首先从 TIMERx_LOAD 寄存器载入计数初始值，减计数到 0 后会回转到 0xFFFF_FFFF 重新开始减计数。当 16bit 计数时，TIMER 首先从 TIMERx_LOAD 寄存器载入计数初始值，减计数到 0 后会回转到 0xFFFF 重新开始减计数。

  当选择 32bit 计数时，TIMERx_LOAD 寄存器的 32bit 全部有效。当选择 16bit 计数时，TIMERx_LOAD 寄存器只有低 16bit 有效。

- 周期模式

  TIMER 持续计数，当配置 TIMERx_LOAD 寄存器后，TIMERx_BGLOAD 寄存器的值会被同步更新成 TIMERx_LOAD 寄存器中的值，如果再次配置 TIMERx_BGLOAD 寄存器，TIMERx_BGLOAD 寄存器中的值会发生更新。TIMER 首先从 TIMERx_LOAD 寄存器载入计数初始值，减计数到 0 后，则会回转到 TIMERx_BGLOAD 寄存器中的值重新开始减计数。

  当选择 32bit 计数时，TIMERx_BGLOAD 寄存器的 32bit 全部有效。当选择 16bit 计数时，TIMERx_BGLOAD 寄存器只有低 16bit 有效。

- 单次计数模式

  TIMER 首先从 TIMERx_LOAD 寄存器载入计数初始值，减计数到 0 后就停止计数。

  如果想要重新开始一次单次计数，需要保证 TIMER 处于使能状态，重新配置一次 TIMERx_LOAD 寄存器。

须知

如果在 TIMER 计数期间配置 TIMERx_LOAD 寄存器，TIMER 会立即从新的配置值开始减计数。

TIMER 支持随时读取当前的计数值，软件可以通过读取 TIMERx_VALUE 寄存器以获取当前的计数值。

16.3.3 产生 DMA 请求和触发 ADC 启动采样的信号

TIMER 支持产生 DMA 请求和触发 ADC 启动采样的信号：

- 配置 TIMERx_CONTROLB[dmabreqen] =1，当 TIMER 减到 0 时，产生 DMA Burst 请求。
- 配置 TIMERx_CONTROLB[dmasreqen] =1，当 TIMER 减到 0 时，产生 DMA Single 请求。
- 配置 TIMERx_CONTROLB[socen] =1，当 TIMER 减到 0 时，产生触发 ADC 启动采样的信号。

16.3.4 中断

TIMER 支持产生定时中断和 DMA 请求溢出中断：

- 定时中断

  如果使能定时中断（TIMERx_CONTROL[timerintenable]=1），则每当 TIMER 计数到 0 时，都会产生定时中断。

  软件可以通过读 TIMERx_RIS[timerris]获取定时中断的原始中断状态，通过读 TIMERx_MIS[timermis]获取屏蔽后的定时中断状态。

  软件可以通过写 TIMERx_INTCLR 寄存器清除定时中断。

- DMA 请求溢出中断

  TIMER 发起 DMA 请求后：

  如果接收到 DMA 返回的清除信号，则 DMA 请求会被清除。

  如果还未接收到 DMA 返回的清除信号，或者在 DMA 返回的清除信号有效期间，TIMER 再次计数到 0，则不会产生新的 DMA 请求。如果使能了 DMA 请求溢出中断（TIMERx_CONTROL[dmaovintenable] =1），则会产生 DMA 请求溢出中断。

  软件可以通过读 TIMERx_RIS[dmaovris]获取 DMA 请求溢出中断的原始中断状态，通过读 TIMERx_MIS[dmaovmis]获取屏蔽后的 DMA 请求溢出中断状态。

软件可以通过向 TIMERx_DMAOVINTCLR[dmaovintclr]位写 1 清除 DMA 请求溢出中断。

16.4 工作方式

初始化

系统初始化时应对 TIMER 进行初始化。初始化 TIMERx（TIMERx 中的“x”取值为 0~3）时应按以下步骤进行配置：

步骤 1 配置 CRG 相关寄存器，打开 TIMERx 时钟门控。

步骤 2 配置 TIMERx_CONTROL[timersize]，选择 32bit 计数或 16bit 计数。

步骤 3 配置 TIMERx_CONTROL[oneshot]和 TIMERx_CONTROL[timermode]，选择 TIMERx 的计数模式。

步骤 4 配置 TIMERx_CONTROL[timerpre]，设置 TIMERx 的预分频倍数。

步骤 5 配置 TIMERx_LOAD 寄存器，设置 TIMERx 的计数初始值。

步骤 6 如果计数模式为周期计数模式，且计数周期与计数初始值不同，则需要配置 TIMERx_BGLOAD 寄存器，设置 TIMERx 的计数周期值。如果计数模式为周期计数模式，且计数周期与计数初始值相同，或计数模式为自由运行模式或单次模式，则忽略步骤 5。

步骤 7 配置 TIMERx_CONTROL[timerintenable]和 TIMERx_CONTROL[dmaovintenable]，设置定时中断和 DMA 请求溢出中断的屏蔽位。

步骤 8 配置 TIMERx_CONTROLB 寄存器，设置 DMA burst 请求、DMA single 请求和触发 ADC 启动采样的功能是否使能。

步骤 9 配置 TIMERx_CONTROL[timeren] =1，启动 TIMERx 计数。

----结束

16.5 TIMERx 寄存器概览

TIMERx（x=0~3）寄存器概览如表 16-1 所示。

TIMER0 的基址是 0x1430_0000；

TIMER1 的基址是 0x1430_1000；

TIMER2 的基址是 0x1430_2000；

TIMER3 的基址是 0x1430_3000。

表16-1 TIMERx 寄存器概览

偏移地址	名称	描述	页码
0x000	TIMERx_LOAD	计数初值寄存器	TIMERx_LOAD
0x004	TIMERx_VALUE	当前计数值寄存器	TIMERx_VALUE
0x008	TIMERx_CONTROL	TIMER 控制寄存器	TIMERx_CONTROL
0x00C	TIMERx_INTCLR	定时中断清除寄存器	TIMERx_INTCLR
0x010	TIMERx_RIS	原始中断寄存器	TIMERx_RIS
0x014	TIMERx_MIS	屏蔽后中断寄存器	TIMERx_MIS
0x018	TIMERx_BGLOAD	计数周期值寄存器	TIMERx_BGLOAD
0x01C	TIMERx_CONTROLB	TIMERx 控制寄存器 B	TIMERx_CONTROLB
0x020	TIMERx_DMAOVINTCLR	DMA 请求溢出中断清除寄存器	TIMERx_DMAOVINTCLR

16.6 TIMERx 寄存器描述

TIMERx_LOAD

TIMERx_LOAD 为计数初值寄存器。用来配置定时器的计数初值。

说明

- 向 TIMERx_LOAD 寄存器写入的最小有效值为 1。
- 当向 TIMERx_LOAD 写 0 时，TIMER 将会立刻产生 1 个中断。

Offset Address: 0x000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	timerload	TIMER 的计数初值。	0x00000000

TIMERx_VALUE

TIMERx_VALUE 为当前计数值寄存器。用于给出正在递减的计数器的当前值。

当向 TIMERx_LOAD 寄存器的写操作发生后，TIMERx_VALUE 立刻反映出计数器的新载入值。

Offset Address: 0x004 Total Reset Value: 0xFFFF_FFFF

Bits	Access	Name	Description	Reset
[31:0]	RO	timervalue	TIMERx 当前计数值。	0xFFFFFFFF

TIMERx_CONTROL

TIMERx_CONTROL 为 TIMER 控制寄存器。

说明

当选择用周期模式进行计数时，需要将 TIMERx_CONTROL.timermode 置 1、TIMERx_CONTROL.oneshot 置 0。

Offset Address: 0x008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7]	RW	timeren	定时器使能。 0：TIMER 禁止； 1：TIMER 使能。	0x0
[6]	RW	timermode	定时器的计数模式。 0：自由运行模式；	0x0

Bits	Access	Name	Description	Reset
			1：周期模式。	
[5]	RW	timerintenable	定时中断屏蔽。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[4]	RW	dmaovintenable	DMA 请求溢出中断屏蔽。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[3:2]	RW	timerpre	该字段用于设置 TIMER 的预分频系数。 00：对 clk_timer 进行 1 倍预分频； 01：对 clk_timer 进行 16 倍预分频； 10：对 clk_timer 进行 256 倍预分频； 11：对 clk_timer 进行 256 倍预分频。	0x0
[1]	RW	timersize	选择 16bit/32bit 计数器操作模式。 0：16bit 计数器； 1：32bit 计数器。	0x0
[0]	RW	oneshot	选择计数模式为单次计数模式还是周期计数模式。 0：周期计数模式或自由运行模式； 1：单次计数模式。	0x0

TIMERx_INTCLR

TIMERx_INTCLR 为中断清除寄存器。对该寄存器的任何写操作都会清除相应计数器的中断状态。

须知

本寄存器是只写寄存器，写进去任意值，都会引起 TIMER 清中断。

Offset Address: 0x00C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	WO	timerintclr	写该寄存器，清除 TIMERx 的中断输出。	0x00000000

TIMERx_RIS

TIMERx_RIS 为原始中断寄存器。

Offset Address: 0x010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RO	dmaovris	DMA 请求溢出中断的原始中断状态。 0：未产生中断; 1：已产生中断。	0x0
[0]	RO	timerris	定时中断的原始中断状态。 0：未产生中断; 1：已产生中断。	0x0

TIMERx_MIS

TIMERx_MIS 为屏蔽后中断寄存器。

Offset Address: 0x014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	-	reserved	保留。	0x00000000
[1]	RO	dmaovmis	屏蔽后的 DMA 请求溢出中断状态。 0：中断无效; 1：中断有效。	0x0
[0]	RO	timermis	屏蔽后的定时中断状态。 0：中断无效; 1：中断有效。	0x0

TIMERx_BGLOAD

TIMERx_BGLOAD 为计数周期值寄存器。

Offset Address: 0x018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:0]	RW	timerbgload	TIMERx 的计数周期值。	0x00000000

TIMERx_CONTROLB

TIMERx_CONTROLB 为 TIMER 的 DMA/ADC 控制寄存器。

Offset Address: 0x01C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2]	RW	socen	触发 ADC 采样信号（SOC 信号）使能位。 0：计数器计到 0 时禁止产生 SOC 信号； 1：计数器计到 0 时可以产生 SOC 信号。	0x0
[1]	RW	dmasreqen	DMA Single 请求使能位。 0：计数器计到 0 时禁止产生 DMA Single 请求； 1：计数器计到 0 时可以产生 DMA Single 请求。	0x0
[0]	RW	dmabreqen	DMA Burst 请求使能位。 0：计数器计到 0 时禁止产生 DMA Burst 请求； 1：计数器计到 0 时可以产生 DMA Burst 请求。	0x0

TIMERx_DMAOVINTCLR

TIMERx_DMAOVINTCLR 为 TIMER 的 DMA 请求溢出中断清除寄存器。

Offset Address: 0x020 Total Reset Value: 0x0000_0000

须知

本寄存器是只写寄存器，写进去 1 会清除 DMA 请求溢出中断，且写 1 自动清零，写入 0 没有作用。

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	WO	dmaov_intclr	DMA 请求溢出中断清除位。 0：写 0 无效; 1：写 1 清零。	0x0

17 窗口看门狗(WWDG)

17.1 概述

看门狗 WatchDog 用于系统异常情况下，一定时间内发出中断/复位信号，以复位整个系统或使系统进入中断函数执行相应操作。

WWDG（Window WatchDog）计数时钟为 clk_wwdg，可以配置此时钟的分频系数以控制 WWDG 计数频率。分频系数由本模块预分频寄存器 WWDG_PRE_DIV 控制。

17.2 特性

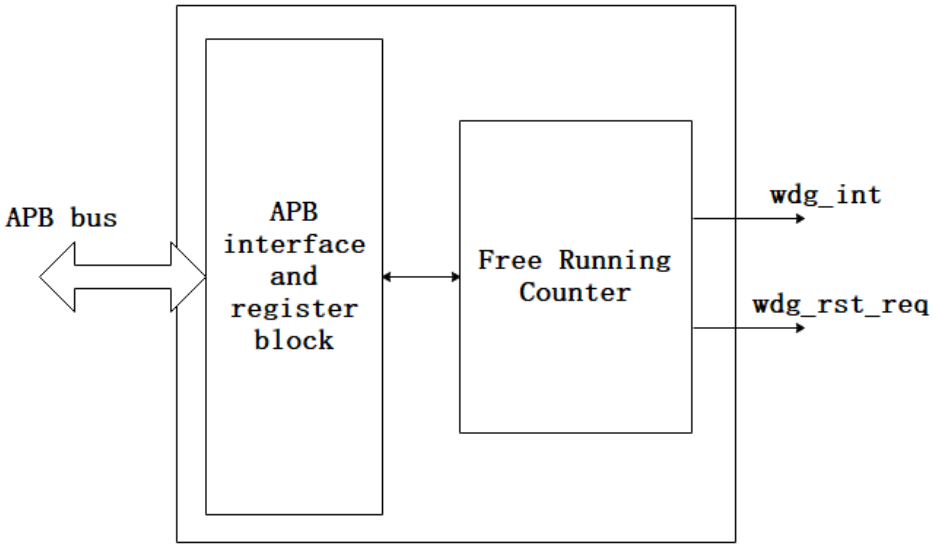
WWDG 具备以下特点:

- 内部具有一个减法计数器（16bit）。
- 支持超时时间间隔（即计数初值）可配置。
- 支持寄存器锁定，防止寄存器被误改。
- 支持超时中断产生。
- 支持复位信号产生。
- 支持调试模式，调试模式下自动停止 WWDG 计数。
- 支持窗口模式，在窗口外喂狗会触发复位。
- 支持计数器当前值回读。

17.3 功能描述

WatchDog 模块框图如图 17-1 所示。

图17-1 模块框图



系统通过总线配置 WatchDog 计数初值、复位/中断信号使能等参数。WatchDog 内部计数器开始计数，定时发出 wdg_int 中断请求给系统，并在系统没有响应中断的情况下（如：死机），发出 wdg_rst_req 复位信号，使系统复位，达到监控系统运行的目的。如果用户需要在精确的时间内保护系统，可以使能窗口模式，任何在窗口外的喂狗操作都会复位系统。

功能原理

WatchDog 的运行基于 1 个 16bit 减法计数器，计数初值由寄存器 WWDG_LOAD 载入。计数初值的计算见 17.4.1 计数初值配置。在 WatchDog 时钟使能情况下，计数值在每个计数时钟的上升沿减 1。

窗口模式未使能时，当计数值递减到 WWDG_LOAD/2，WatchDog 将产生一个中断信号 wdg_int。然后在下一个计数时钟上升沿，继续递减计数。WatchDog 计数器当前的计数值可以从 WWDG_VALUE 查询。

如果计数器的计数值递减到 0 时，CPU 还没有清除 WatchDog 中断，则 WatchDog 将发出复位信号 wdg_rst_req，以复位整个系统，同时计数器停止计数。

窗口模式使能时，需要事先配置 WWDG_WINDOW 窗口值寄存器。要求 WWDG_WINDOW 一定小于 WWDG_LOAD。

当计数值递减到窗口值时，WatchDog 将产生一个中断信号 wdg_int。然后在下一个计数时钟上升沿，继续递减计数。如果计数器的计数值递减到 0 时，CPU 还没有清除

WatchDog 中断，则 WatchDog 将发出复位信号 wdg_rst_req，以复位整个系统，同时计数器停止计数。

在窗口之外（即大于窗口值小于加载值）的喂狗操作，也会复位整个系统。

中断和复位

根据实际应用需要，可通过配置 WWDG_CONTROL 使能或者禁止 WatchDog 产生中断和复位信号。用户可以自行选择是否开启中断。若不开启中断，则用户需要自行选择合适的喂狗时机进行喂狗。

WatchDog 的原始中断状态和屏蔽后中断状态可通过 WWDG_RIS 和 WWDG_MIS 查询。通过向 WWDG_KEY 写入 0xAA 进行喂狗操作清除中断。

指令字寄存器

通过配置 WWDG_KEY 寄存器，可以对 WatchDog 进行写保护、开启计数、关闭计数、喂狗等操作:

- 向 WWDG_KEY 写入 0x55，可以打开所有 WatchDog 寄存器的写权限。
- 向 WWDG_KEY 寄存器写入其他任何值，可以关闭所有 WatchDog 寄存器（WWDG_KEY 寄存器除外）的写权限。

  该特性保护 WatchDog 的寄存器不被软件错误修改，从而使得在异常情况下，WatchDog 不至于被软件错误地中止操作。
- 向 WWDG_KEY 寄存器写入 0xCC，可以开启 WatchDog。
- 向 WWDG_KEY 寄存器写入 0xDD，可以关闭 WatchDog。
- 向 WWDG_KEY 寄存器写入 0xAA，即进行喂狗操作，可以使 WatchDog 从 WWDG_LOAD 重新载入计数初值进行计数，并清除中断。

调试模式

在调试模式下，WatchDog 使能自动关闭，以防止干扰正常的调试操作。

17.4 典型配置

17.4.1 计数初值配置

在向寄存器 WWDG_LOAD 载入计数初值之前需要进行计算。

$Value_{WWDG_LOAD} = T_{WWDG} \times f_{clk}$

说明

其中上式中，各参数代表的意思分别是：

- T_{WWDG} 表示 WatchDog 计数时间；
- $Value_{WWDG_LOAD}$ 表示 WatchDog 计数初值；
- f_{clk} 表示 WatchDog 计数时钟频率。计数时钟频率由 WWDG_PRE_DIV 控制。

17.4.2 模块初始化配置

系统上电复位后 WatchDog 计数器处于停止计数状态，在系统初始化过程中需要将 WatchDog 初始化并启动其运行。WatchDog 的初始化过程如下：

步骤 1 配置 CRG，使能 WatchDog 时钟和解除复位。

步骤 2 对 WWDG_KEY 寄存器写 0x55，开启写权限。根据想要设置的超时时长和 WatchDog 的时钟频率，配置 WWDG_PRE_DIV，计算计数初值。写寄存器 WWDG_LOAD，设定计数初值。

步骤 3 写寄存器 WWDG_CONTROL，解除中断/复位屏蔽/使能窗口模式。若使能窗口模式，则配置 WWDG_WINDOW。

向 WWDG_KEY 写入 0xCC，启动 WatchDog 计数并写保护其他寄存器。

----结束

17.4.3 中断处理过程

收到 WatchDog 发出的中断后，系统应及时清除其中断状态，并使其载入计数初值重新开始计数。若不及时清除中断状态，则在 WatchDog 计数值递减到 0 后，会产生复位信号复位整个系统，WatchDog 停止工作。WatchDog 中断处理的过程如下：

步骤 1 向 WWDG_KEY 写 0xAA，清除 WatchDog 的中断状态，同时也使 WatchDog 自动载入计数初值重新开始计数。此时自然也开启了其他寄存器的写保护。

----结束

17.5 WWDG_CFG 寄存器概览

WWDG_CFG 寄存器概览如表 17-1 所示。

表17-1 WWDG_CFG 寄存器概览（基址是 0x1440_0000）

偏移地址	名称	描述	页码
0x0000	WWDG_LOAD	计数初值寄存器	WWDG_LOAD
0x0004	WWDG_VALUE	计数器当前值寄存器	WWDG_VALUE
0x0008	WWDG_WINDOW	窗口值寄存器	WWDG_WINDOW
0x000C	WWDG_KEY	指令字寄存器	WWDG_KEY
0x0010	WWDG_PRE_DIV	预分频寄存器	WWDG_PRE_DIV
0x0014	WWDG_CONTROL	控制寄存器	WWDG_CONTROL
0x0018	WWDG_RIS	原始中断标志寄存器	WWDG_RIS
0x001C	WWDG_MIS	屏蔽后中断标志寄存器	WWDG_MIS

17.6 WWDG_CFG 寄存器描述

WWDG_LOAD

WWDG_LOAD 为计数初值寄存器。

Offset Address: 0x0000 Total Reset Value: 0xFFFF

Bits	Access	Name	Description	Reset
[15:0]	RW	wwdg_load	计数初值。	0xFFFF

WWDG_VALUE

WWDG_VALUE 为计数器当前值寄存器。

Offset Address: 0x0004 Total Reset Value: 0xFFFF

Bits	Access	Name	Description	Reset
[15:0]	RO	wwdg_value	WatchDog 计数器当前值。	0xFFFF

WWDG_WINDOW

WWDG_WINDOW 为窗口值寄存器。

Offset Address: 0x0008 Total Reset Value: 0xFFFF

Bits	Access	Name	Description	Reset
[15:0]	RW	wwdg_window	窗口值，禁止大于计数初值寄存器，否则会产生预期之外的动作。	0xFFFF

WWDG_KEY

WWDG_KEY 为指令字寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_00DD

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7:0]	RW	wwdg_key	对该寄存器写入 0xAA 使 WatchDog 从寄存器 WWDG_LOAD 中重新载入初值计数（喂狗），并可清除 WatchDog 的中断。 向该寄存器写入 0x55，可打开所有可写寄存器的写权限； 向该寄存器写入其他值，关闭所有可写寄存器写权限；本寄存器始终写有效； 向该寄存器写入 0xCC，可开启 WWDG； 向该寄存器写入 0xDD，可关闭 WWDG。	0xDD

WWDG_PRE_DIV

WWDG_PRE_DIV 为预分频寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	wwdg_pre_div	工作时钟的 2 的幂次分频。 0x0：不分频; 0x1：2 分频; 0x2：4 分频: ... 0xD：8192 分频; 其他：保留。 最多支持 8192 分频。 配置大于最大分频值的值，按最大分频值进行分频。	0x0

WWDG_CONTROL

WWDG_CONTROL 为控制寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2]	RW	window_mode_en	WatchDog 窗口模式使能。 0：禁止; 1：使能。 当禁止窗口模式时，无需设置 wwdg_window，中断固定在 wwdg_load/2 处产生，此时在任意时刻喂狗均不会触发复位信号，复位信号只有在计数值为 0 且未清除中断时才会产生。 当使能窗口模式时，wwdg_window 值生效，中断在计数器值等于窗口值时产生，在窗口值之外喂狗会产生复位信号，在计数为 0 且未清除中断时也会产生复位信	0x0

Bits	Access	Name	Description	Reset
			号。	
[1]	RW	resen	WatchDog 复位信号输出使能。 0：禁止； 1：使能。	0x0
[0]	RW	inten	WatchDog 中断信号输出使能。 0：禁止 wwdgmis 产生； 1：使能 wwdgmis。	0x0

WWDG_RIS

WWDG_RIS 为原始中断标志寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RO	wwdgris	WatchDog 原始中断状态。 0：未产生中断； 1：已产生中断。	0x0

WWDG_MIS

WWDG_MIS 为屏蔽后中断标志寄存器。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RO	wwdgmis	WatchDog 屏蔽后的中断状态。 0：未产生中断或者中断被屏蔽； 1：已产生中断。	0x0

18 独立看门狗(IWDG)

18.1 概述

独立看门狗 IWDG 用于系统异常情况下，一定时间内发出复位信号，以复位整个系统。

IWDG（Independent WatchDog）计数时钟为 clk_iwdg，不分频时，频率与 MCU 内置低速振荡器（LOSC）频率一致。可以通过 IWDG_PRE_DIV 控制时钟的分频系数。

18.2 特性

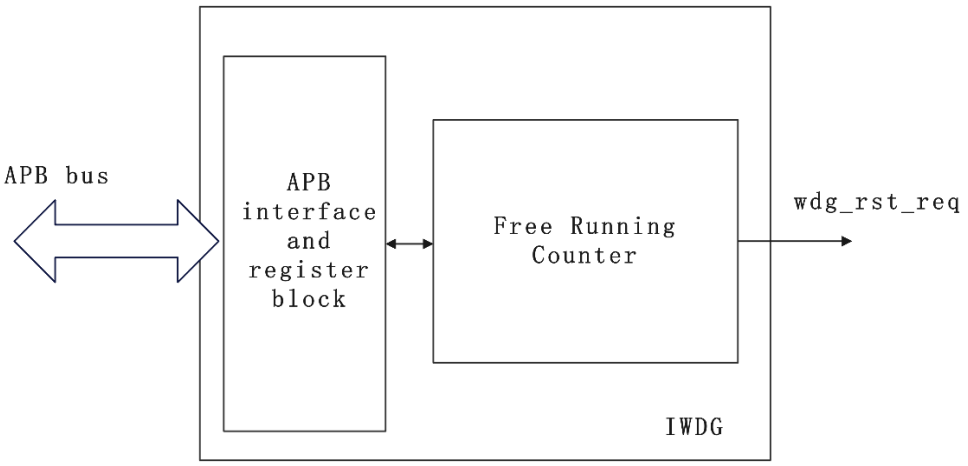
IWDG 具备以下特点：

- 内部具有一个减法计数器（8bit）。
- 支持超时时间间隔（即计数初值）可配置。
- 支持寄存器锁定，防止寄存器被误改。
- 支持复位信号产生。
- 支持调试模式，调试模式下自动停止 IWDG 计数。
- 支持窗口模式，在窗口外喂狗会触发复位。
- 支持计数器当前值回读。

18.3 功能描述

IWDG 模块框图如图 18-1 所示。

图18-1 模块框图



系统通过总线配置 IWDG 计数初值、复位信号使能等参数。IWDG 内部计数器开始计数，在系统没有定时进行喂狗操作的情况下（如：死机），发出 wdg_rst_req 复位信号，使系统复位，达到监控系统运行的目的。如果用户需要在精确的时间内保护系统，可以使能窗口模式，任何在窗口外的喂狗操作都会复位系统。

功能原理

IWDG 的运行基于 1 个 8bit 减法计数器，计数初值由寄存器 IWDG_LOAD 载入。计数初值的计算请参见 18.4.1 计数初值配置章节。在 IWDG 时钟使能情况下，计数值在每个计数时钟的上升沿减 1。IWDG 计数器当前的计数值可以从 IWDG_VALUE 查询。

如果计数器的计数值递减到 0 时，CPU 还没有进行喂狗操作，则 IWDG 将发出复位信号 wdg_rst_req，以复位整个系统，同时计数器停止计数。

窗口模式使能时，需要事先配置 IWDG_WINDOW 窗口值寄存器。要求 IWDG_WINDOW 一定小于 IWDG_LOAD。同时禁止配置窗口值小于等于 4，否则命令不能及时生效，会复位整个系统。

在窗口之外（即大于窗口值小于加载值）的喂狗操作，也会复位整个系统。

复位

根据实际应用需要，可通过配置 IWDG_CONTROL 使能或者禁止 IWDG 产生复位信号。用户需要自行选择合适的喂狗时机进行喂狗。

通过向 IWDG_KEY 写入 0xAA 进行喂狗操作。

指令字寄存器

通过配置 IWDG_KEY 寄存器，可以对 IWDG 进行写保护、开启计数、关闭计数、喂狗等操作：

- 向 IWDG_KEY 写入 0x55，可以打开所有 IWDG 寄存器的写权限。
- 向 IWDG_KEY 寄存器写入其他任何值，可以关闭所有 IWDG 寄存器（IWDG_KEY 寄存器除外）的写权限。

  该特性保护 IWDG 的寄存器不被软件错误修改，从而使得在异常情况下，IWDG 不至于被软件错误地中止操作。
- 向 IWDG_KEY 寄存器写入 0xCC，可以开启 IWDG。
- 向 IWDG_KEY 寄存器写入 0xDD，可以关闭 IWDG。
- 向 IWDG_KEY 寄存器写入 0xAA，即进行喂狗操作，可以使 IWDG 从 IWDG_LOAD 重新载入计数初值进行计数。

调试模式

在调试模式下，IWDG 使能自动关闭，以防止干扰正常的调试操作。

18.4 典型配置

18.4.1 计数初值配置

在向寄存器 IWDG_LOAD 载入计数初值之前需要进行计算。

$$Value_{IWDG_LOAD} = \mathbf{T}_{IWDG} \times \mathbf{f}_{clk}$$

说明

其中上式中，各参数代表的意思分别是：

- $\mathbf{T}_{IWDG}$ 表示 IWDG 计数时间；
- $Value_{IWDG_LOAD}$ 表示 IWDG 计数初值；
- $\mathbf{f}_{clk}$ 表示 IWDG 计数时钟频率。计数时钟频率由 IWDG_PRE_DIV 控制。

18.4.2 模块初始化配置

系统上电复位后 IWDG 计数器处于停止计数状态，在系统初始化过程中需要将 IWDG 初始化并启动其运行。IWDG 的初始化过程如下：

步骤 1 配置 CRG，使能 IWDG 时钟和解除复位。等待 200us。

步骤 2 对 IWDG_KEY 寄存器写 0x55，开启写权限。根据想要设置的超时时长和 IWDG 的时钟频率，配置 IWDG_PRE_DIV，计算计数初值。写寄存器 IWDG_LOAD，设定计数初值。

步骤 3 写寄存器 IWDG_CONTROL，解除复位屏蔽/使能窗口模式。若使能窗口模式，则配置 IWDG_WINDOW。

步骤 4 向 IWDG_KEY 写入 0xCC，启动 IWDG 计数并写保护其他寄存器。

----结束

18.5 IWDG_CFG 寄存器概览

IWDG_CFG 寄存器概览如表 18-1 所示。

表18-1 IWDG_CFG 寄存器概览（基址是 0x1440_1000）

偏移地址	名称	描述	页码
0x0000	IWDG_LOAD	计数初值寄存器	IWDG_LOAD
0x0004	IWDG_VALUE	计数器当前值寄存器	IWDG_VALUE
0x0008	IWDG_WINDOW	窗口值寄存器	IWDG_WINDOW
0x000C	IWDG_KEY	指令字寄存器	IWDG_KEY
0x0010	IWDG_PRE_DIV	预分频寄存器	IWDG_PRE_DIV
0x0014	IWDG_CONTROL	控制寄存器	IWDG_CONTROL

18.6 IWDG_CFG 寄存器描述

IWDG_LOAD

IWDG_LOAD 为计数初值寄存器。

Offset Address: 0x0000 Total Reset Value: 0xFF

Bits	Access	Name	Description	Reset
[7:0]	RW	iwdg_load	计数初值。	0xFF

IWDG_VALUE

IWDG_VALUE 为计数器当前值寄存器。

Offset Address: 0x0004 Total Reset Value: 0xFF

Bits	Access	Name	Description	Reset
[7:0]	RO	iwdg_value	IWDG 计数器当前值，读出值延后于实际计数器的当前值约 50us。	0xFF

IWDG_WINDOW

IWDG_WINDOW 为窗口值寄存器。

Offset Address: 0x0008 Total Reset Value: 0xFF

Bits	Access	Name	Description	Reset
[7:0]	RW	iwdg_window	窗口值，禁止大于计数初值寄存器，否则会产生预期之外的动作。 禁止配置窗口值小于等于 4，否则喂狗指令不会及时生效，会导致 IWDG 复位。	0xFF

IWDG_KEY

IWDG_KEY 为指令字寄存器。

Offset Address: 0x000C Total Reset Value: 0x0000_00DD

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7:0]	RW	iwdg_key	对该寄存器写入 0xAA 使 IWDG 从寄存器 IWDG_LOAD 中重新载入初值计数（喂狗）。 向该寄存器写入 0x55，可打开所有可写寄存器的写权限； 向该寄存器写入其他值，关闭所有可写寄存器写权限；本寄存器始终写有效； 向该寄存器写入 0xCC，可开启 IWDG； 向该寄存器写入 0xDD，可关闭 IWDG。	0xDD

IWDG_PRE_DIV

IWDG_PRE_DIV 为预分频寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	iwdg_pre_div	工作时钟的 2 的幂次分频。 0x0：不分频； 0x1：2 分频； 0x2：4 分频： ... 0x8：256 分频； 其他：保留。 最多支持 256 分频。 配置大于最大分频值的值，按最大分频值进行分频。	0x0

IWDG_CONTROL

IWDG_CONTROL 为控制寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	-	reserved	保留。	0x00000000
[2]	RW	window_mode_en	IWDG 窗口模式使能。 0：禁止； 1：使能。 当禁止窗口模式时，无需设置 iwdg_window，此时在任意时刻喂狗均不会触发复位信号，复位信号只有在计数值为 0 时才会产生。 当使能窗口模式时，iwdg_window 值生效，在窗口值之外喂狗会产生复位信号，在计数为 0 时也会产生复位信号。	0x0
[1]	RW	resen	IWDG 复位信号输出使能。 0：禁止； 1：使能。	0x0
[0]	-	reserved	保留。	0x0

19 集成电路接口(I2C)

19.1 概述

I2C（Inter-Integrated Circuit）控制器支持 Master（主机）和 Slave（从机）接口，完成 CPU 对 I2C 总线上连接的从机的读写访问，或者响应 I2C 总线上连接的主机的读写访问。I2C 控制器的主机和从机兼容 NXP I2C-bus specification and user manual version 6.0 协议。

MCU 共有 2 个 I2C 控制器。

19.2 特性

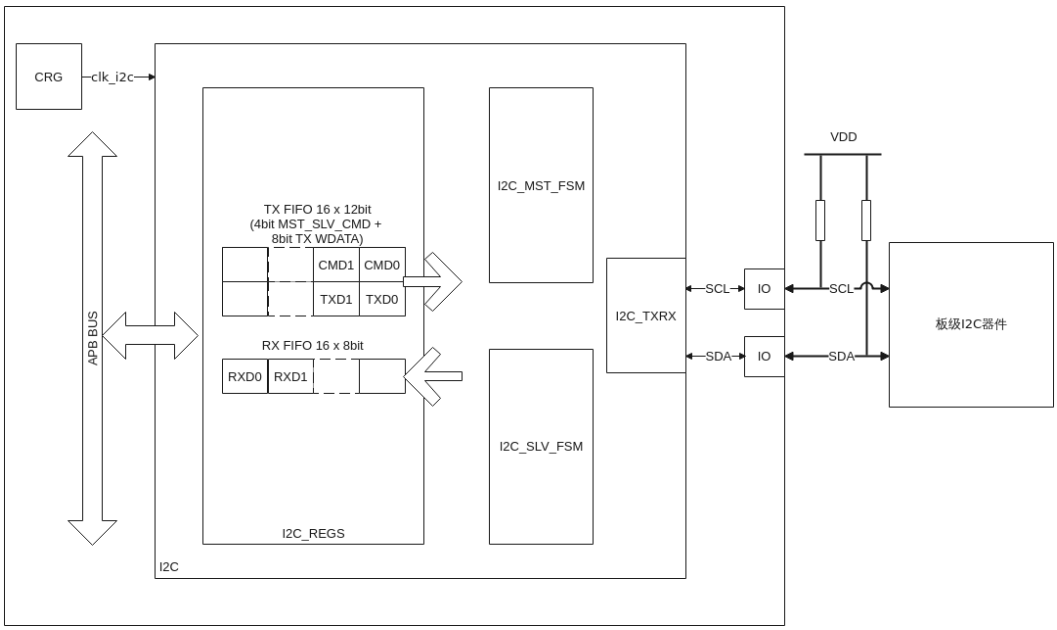
I2C 控制器具有以下功能特点：

- 支持 Master（主机）作为 Master-Transmitter 或者 Master-Receiver 工作。
- 支持 Slave（从机）作为 Slave-Transmitter 或者 Slave-Receiver 工作。
- 支持 I2C 标准时序和各种非标准时序。
- 支持 16 x 12bit 的 TX FIFO 和 16 x 8bit 的 RX FIFO。
- 支持标准地址（7bit）和扩展地址（10bit）。
- 支持标准模式（最高 100 kHz）和快速模式（最高 400 kHz）。
- 支持 DMA 操作。
- 支持多主机总线仲裁。
- 支持 SCL 时钟线和 SDA 数据线 Spike Suppression（数字滤波）功能。
- 支持 Clock synchronization（时钟同步），SCL Stretching（SCL 延长）。
- 支持 General Call，Software Reset 和 Start Byte 功能。

- 支持灵活配置的 ACK/NACK。
- 支持 2 个 Slave 地址和地址掩码。
- 支持 I2C 总线 SCL 时钟低电平超时检测。
- 不支持 CBUS 器件。

19.3 功能框图

图19-1 I2C 控制器的功能框图



I2C 控制器的核心功能模块包括 I2C_REGS（I2C 寄存器）模块、I2C_MST_FSM（主机状态机）模块、I2C_SLV_FSM（从机状态机）模块，I2C_TXRX（I2C 发送接收模块)。如图 19-1 所示，软件写 TX FIFO，I2C_MST_FSM / I2C_SLV_FSM 状态机模块执行时序命令，I2C_TXRX 模块发送和接收数据。

CPU 通过 MCU 内部总线读写 I2C 寄存器，配置 I2C 控制器。I2C 控制器支持 Master-only 模式（主机模式)、Slave-only 模式（从机模式）和 Master-Slave 模式（主从模式)。在 Master-Only 模式下，I2C 控制器开启主机状态机，关闭从机状态机，主机状态机译码并执行时序命令，完成 I2C 总线的写操作或者读操作。在 Slave-Only 模式下，I2C 控制器开启从机状态机，关闭主机状态机，（在检测到地址匹配之后）从机状态机译码并执行时序命令，完成 I2C 总线的写操作或者读操作。在 Master-Slave 模式下，I2C 控制器同时开启主机状态机和从机状态机，主机状态机和从机状态机译码并执行

时序命令，完成 I2C 总线的写操作或者读操作。I2C 控制器执行时序命令，通过板级布线与 I2C 器件进行通信。

I2C 控制器支持标准时序和非标准时序，标准地址（7bit）和扩展地址（10bit），详细描述请参见 19.4.8 I2C 总线时序合成章节。

I2C 控制器支持标准模式（最高 100 kHz）和快速模式（最高 400 kHz），传输速率取决于 I2C 控制器的工作时钟频率以及时钟线（SCL）的配置。I2C 控制器工作时钟频率的配置，请参见 5 时钟复位生成(CRG)。时钟线（SCL）的配置，请参见 19.4.1.1 SCL 配置章节。

I2C 控制器支持 16 x 12bit 的 TX FIFO 和 16 x 8bit 的 RX FIFO。TX FIFO 的 12bit 由 4bit 时序命令，8bit 待发送的数据构成。RX FIFO 的 8bit 是已接收的数据。软件需要把期望执行的命令和待发送的数据以{时序命令，数据}的格式写入 I2C_TX_FIFO 寄存器的{mst_slv_cmd，tx_fifo_wdata}位域。I2C 控制器执行{时序命令，数据}发送数据，或者接收数据。软件读 I2C_RX_FIFO[rx_fifo_rdata]以获取板级 I2C 器件发送给 I2C 控制器的数据。即，软件需要构造{时序命令，数据}数据类型，合成 I2C 的各种时序，详细指导请参见 19.4.8 I2C 总线时序合成章节。

TX FIFO 和 RX FIFO 的水线影响 TX FIFO 水线中断和 I2C TX FIFO 的 DMA Burst 请求。可以通过配置 I2C_TX_WATERMARK[tx_watermark]和 I2C_RX_WATERMARK[rx_watermark]，设置 TX FIFO 和 RX FIFO 的水线。

I2C FIFO 支持 DMA 操作。TX FIFO 支持 DMA 存储器到外设（TX FIFO），外设到外设（TX FIFO）传输类型。RX FIFO 支持 DMA 外设（RX FIFO）到存储器，外设（RX FIFO）到外设传输类型，支持外设流控。详细指导请参见 19.5.6 DMA 方式下的写操作数据传输、19.5.7 DMA 方式下的读操作数据传输章节。

19.4 功能描述

19.4.1 SCL 和 SDA 相关配置

19.4.1.1 SCL 配置

I2C 控制器支持标准模式（最高 100 kHz）和快速模式（最高 400 kHz）。软件需要配置 I2C_SCL_CFG[scl_high_time]和[scl_low_time]，来调整 SCL 高电平时长和低电平时长。配置 SCL 时可以参考表 19-1。

I2C 控制器支持 SCL 时钟线低电平超时检测，在 I2C 总线挂死时，上报中断（scl_low_timeout 中断），便于软件做通信恢复。软件配置 I2C_SCL_TIMEOUT[scl_low_timeout]，调整 SCL 低电平超时的阈值时间。

19.4.1.2 SDA 配置

I2C 控制器支持配置 SDA 线的保持时间和建立时间。软件需要配置 I2C_SDA_CFG[sda_hold_time]来直接调整 SDA 保持时间。建立时间由 SCL 时钟低电平时间减去 SDA 保持时间间接得出。配置 SDA 时可以参考表 19-1。

I2C 控制器支持 SDA 线的延迟采样。当 SDA 到达 MCU 管脚时，若数据保持时间不足，可以使用该特性，软件配置 I2C_SCL_CFG[sda_delay_time]，延迟采样 SDA 数据，增加保持时间，以满足 I2C 控制器采样所需。

图19-2 I2C 协议约束的 I2C 总线器件 SCL 和 SDA 线的时序定义

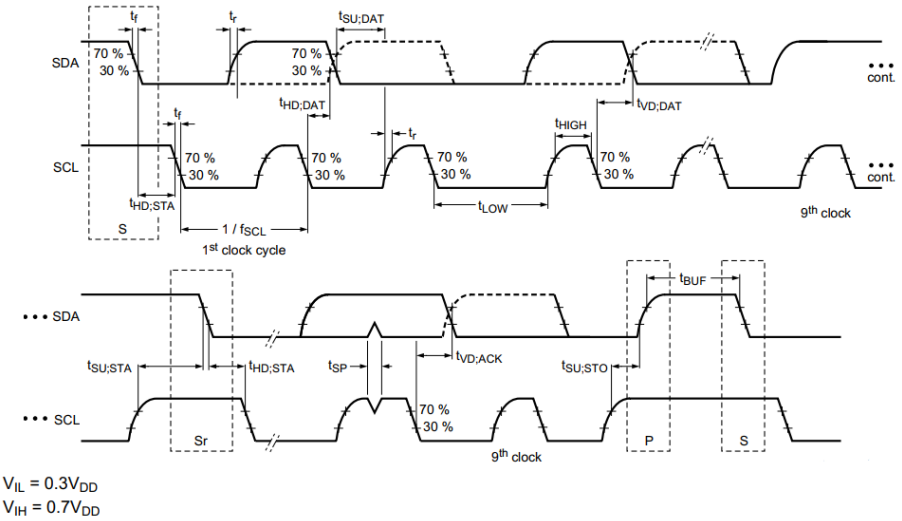


表19-1 协议约束的 I2C 总线器件 SCL 和 SDA 线的部分特性

符号	参数	标准模式		快速模式		单位
		Min	Max	Min	Max	
f_{SCL}	SCL 时钟频率	0	100	0	400	kHz
t_{LOW}	SCL 时钟低电平时间	4.7	-	0.6	-	μs
t_{HIGH}	SCL 时钟高电平时间	4.0	-	0.6	-	μs
$t_{SU;STA}$	ReSTART 条件的建立时间	4.7	-	0.6	-	μs

符号	参数	标准模式		快速模式		单位
		Min	Max	Min	Max	
$t_{HD;STA}$	(Re)START 条件的保持时间	4.0	-	0.6	-	μs
$t_{SU;DAT}$	数据建立时间	250	-	100	-	ns
$t_{HD;DAT}$	数据保持时间	0	-	0	-	μs
$t_{SU;STO}$	STOP 条件的建立时间	4.0	-	0.6	-	μs
t_{BUF}	总线空闲时间（在 STOP 条件和 START 条件之间）	4.7	-	1.3	-	μs
t_r	SDA 和 SCL 的上升时间（0.3VDD 上升到 0.7VDD）	-	1000	20	300	ns
t_f	SDA 和 SCL 的下降时间（0.7VDD 下降到 0.3VDD）	-	300	20 × (VDD / 5.5 V)	300	ns

19.4.1.3 数字滤波

I2C 控制器支持 SCL 和 SDA 的 spike suppression（数字滤波），最大可过滤 I2C 工作时钟 15 个周期的毛刺。软件配置 I2C_FILTER[spike_filter_time]，调整滤波功能。

说明

数字滤波开启后，SCL 和 SDA 的输入信号会延迟相同个数的周期时间。

19.4.1.4 总线空闲

I2C 控制器支持 Bus Free（总线空闲）检测，最大可检测 I2C 工作时钟 65535 个周期时间的空闲。当总线空闲时，Master 才能发出 START 条件以启动帧的传输。当总线忙时，若软件请求 Master 启动传输，Master 会等到总线空闲后，才能发出 START 条件以启动帧的传输。软件配置 I2C_BUS_FREE[bus_free_time]，调整总线空闲检测的阈值时间。

说明

I2C Bus Free（总线空闲）：控制器检测到 STOP 条件后，SCL 和 SDA 同时为高电平并持续到达阈值时间，控制器认为总线空闲。

19.4.2 FIFO

I2C 控制器支持 16 x 12bit 的 TX FIFO 和 16 x 8bit 的 RX FIFO，深度均为 16。TX FIFO 和 RX FIFO 的水线均可独立配置。水线的配置影响水线中断和 DMA Burst 请求。

- 软件可以配置 I2C_TX_WATERMARK[tx_watermark]，单独配置 TX FIFO 水线。TX FIFO 有效数目小于等于 TX FIFO 水线时，产生 TX FIFO 水线原始中断，产生 TX FIFO 的 DMA Burst 请求。
- 软件可以配置 I2C_RX_WATERMARK[rx_watermark]，单独配置 RX FIFO 水线。RX FIFO 有效数目大于等于 RX FIFO 水线时，产生 RX FIFO 水线原始中断，产生 RX FIFO 的 DMA Burst 请求。

I2C 控制器支持软件配置复位 TX FIFO 和 RX FIFO，写 I2C_CTRL1[rst_tx_fifo]或 I2C_CTRL1[rst_rx_fifo]分别复位 TX FIFO 和 RX FIFO。

说明

I2C 控制器的 TX FIFO 宽度是 12bit，由 4bit mst_slv_cmd（时序命令）和 8bit tx_fifo_wdata（数据）构成。详细功能请参见 19.4.8 I2C 总线时序合成章节。

19.4.3 SCL 延长

I2C 控制器 Master（主机）和 Slave（从机）均支持 SCL Stretching（SCL 延长），该功能缺省使能，软件可配置 I2C_MODE[scl_stretch_disable]同时关闭 Master 和 Slave 的 SCL Stretching。

使能时，若条件满足，Master 拉低 SCL，暂停 I2C 通信，并保持直到条件不满足。触发 Master SCL Stretching 的条件：

- 当软件启动 Master 后，执行 MST_P 时序命令之前，如果 TX FIFO 空（即 Master 无法读到下一个时序命令）：
  - 若使能 SCL Stretching，此时触发 Master SCL Stretching，拉低 SCL，暂停 I2C 通信，保持到 TX FIFO 非空（即 Master 能读到下一个时序命令）。TX FIFO 非空后，Master 读到下一个时序命令，恢复 SCL；
  - 若不使能 SCL Stretching，此时不触发 Master SCL Stretching，Master 自动发出 STOP 条件，结束当前帧的传输。
- 当 Master 时序命令为 RD 系列时序命令（即期望接收 Data），但是 RX FIFO 满：
  - 若使能 SCL Stretching，此时触发 Master SCL Stretching，拉低 SCL，暂停 I2C 通信，Master 暂停执行时序命令，保持到 RX FIFO 非满。RX FIFO 非满后，继续执行该时序命令，恢复 SCL；

  - 若不使能 SCL Stretching，此时不触发 Master SCL Stretching，Master 不暂停执行时序命令，但是新的数据被丢弃，不会被保存在 RX FIFO 中。

- 当 Master 时序命令为 INT 系列命令：
  - 若使能 SCL Stretching，此时触发 Master SCL Stretching，Master 执行 INT 系列命令，上报中断后，Master 暂停执行新的时序命令，拉低 SCL，暂停 I2C 通信，保持到中断清除。中断清除后，Master 继续执行下一个时序命令，恢复 SCL，恢复 I2C 通信；
  - 若不使能 SCL Stretching，此时不触发 Master SCL Stretching，Master 执行 INT 系列命令，上报中断后，继续执行下一个时序命令。

使能时，若条件满足，Slave 拉低 SCL 并保持到条件不满足。触发 Slave SCL Stretching 的条件：

- 当 Slave 被寻址到，但是 TX FIFO 空（即 Slave 无法读到新的时序命令）：
  - 若使能 SCL Stretching，此时触发 Slave SCL Stretching，拉低 SCL，暂停 I2C 通信，保持到 TX FIFO 非空（即 Slave 能读到新的时序命令）。TX FIFO 非空后，Slave 读到下一个时序命令，释放 SCL，恢复 I2C 通信；
  - 若不使能 SCL Stretching，此时不触发 Master SCL Stretching，SCL 正常翻转，Slave 等待 TX FIFO 非空（读到下一个时序命令）或者下一个的（Re）START 条件。
- 当 Slave 时序命令为 RD 系列命令（即期望接收 Data），但是 RX FIFO 满：
  - 若使能 SCL Stretching，此时触发 Slave SCL Stretching，拉低 SCL，暂停 I2C 通信，保持到 RX FIFO 非满。RX FIFO 非满后，释放 SCL，恢复 I2C 通信；
  - 若不使能 SCL Stretching，此时不触发 Master SCL Stretching，SCL 正常翻转，Slave 不暂停执行时序命令，但是新的数据被丢弃，不会被保存在 RX FIFO 中。
- 当 Slave 时序命令为 INT 系列命令：
  - 若使能 SCL Stretching，此时触发 Slave SCL Stretching，Slave 执行 INT 系列命令，上报中断后，Slave 暂停执行新的时序命令，拉低 SCL，暂停 I2C 通信，保持到中断清除。中断清除后，Slave 继续执行下一个时序命令，恢复 SCL，恢复 I2C 通信；
  - 若不使能 SCL Stretching，此时不触发 Slave SCL Stretching，Slave 执行 INT 系列命令，上报中断后，继续执行下一个时序命令。

19.4.4 ACK 和 NACK

I2C 控制器支持灵活配置发送 ACK（Acknowledge）或者 NACK（Not Acknowledge），可配置每一个 Byte 接收后期望发送的应答 bit 是 ACK（低电平）还是 NACK（高电平）。软件需要在做 I2C 时序合成时，使用以 TACK 或者 TNACK 为后缀的时序命令来实现 ACK 或者 NACK 的发送，请参见 19.4.8 I2C 总线时序合成章节。

I2C 控制器支持灵活配置接收 ACK 或者 NACK，可配置每一个 byte 发送后期望接收的应答 bit 是 ACK（低电平）还是 NACK（高电平）。软件需要在做 I2C 时序合成时，使用以 RACK 或者 RNACK 为后缀的时序命令来实现 ACK 或者 NACK 的接收，请参见 19.4.8 I2C 总线时序合成章节。

软件可配置 I2C_MODE[rack_mode]，调整接收 ACK/NACK 的模式：

- 严格 ACK/NACK 模式：如果 Master-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，那么 Master 自动发出 STOP condition，结束 I2C 通信后，自动退出 Master。如果 Slave-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，那么 Slave 退出当前 I2C 帧的通信，SDA 输出高阻。
- 忽略 ACK/NACK 模式：如果 Master-Transmitter，或 Slave-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，忽略此不匹配，继续执行下一个时序命令。

19.4.5 数据传输位序

I2C 控制器支持数据 Byte 的 bit 传输位序调整，同时对 Master 和 Slave 生效，软件可配置 I2C_MODE[lit_end]，调整数据传输位序配置：

- MSbit-First 模式：最高位 bit 先传输。
- LSbit-First 模式：最低位 bit 先传输。

19.4.6 Master 和 Slave

I2C 控制器支持 Master-Only 模式、Slave-Only 模式、Master-Slave 模式，缺省为 I2C 控制器不使能，软件可配置 I2C_MODE[mst_slv_function]来配置具体的模式。

- Master-Only 模式：I2C 控制器的 Master 使能，Slave 禁能。软件需要配置 I2C_CTRL1[mst_start]启动 Master 执行时序命令。
- Slave-Only 模式：I2C 控制器的 Slave 使能，Master 禁能。Slave 自动检测 START 条件，之后检测地址是否匹配，若匹配，上报中断（slv_addr_match 地址

匹配中断)，执行 TX FIFO 中的时序命令，若不匹配，Slave 退出当前帧的通信，检测下一个 START 条件和地址。

- Master-Slave 模式：I2C 控制器的 Master 和 Slave 同时使能。软件需要配置 I2C_CTRL1[mst_start]启动 Master 执行时序命令。Slave 自动检测 START 条件，之后检测地址是否匹配，若匹配，上报中断（slv_addr_match 地址匹配中断），执行 TX FIFO 中的时序命令，若不匹配，Slave 退出当前帧的通信，检测下一个 START 条件和地址。

说明

Master-Slave 模式，在大多数使用场景下，I2C 控制器 Master 寻址的地址不会跟 I2C 控制器 Slave 自身的地址匹配。即，I2C 控制器不是 Master 在工作，就是 Slave 在工作。但是也存在特殊情况，当 Master 寻址的地址刚好跟 Slave 自身的地址匹配时（可认为是一种在线自测试），TX FIFO 中的一个时序命令二进制同时被 Master 和 Slave 共享，该二进制会被 Master 和 Slave 分别解析出不同的含义，请参见表 19-3。

19.4.7 Slave 自身地址和特殊地址

I2C 控制器支持 2 个 Slave 地址配置，相应地支持 2 个 Slave 地址掩码，可支持特定的 Slave 单一地址或者特定的 Slave 地址区间。

- 软件配置 I2C_OWN_ADDR[own_address]和[own_address_mask]，调整 Slave 自身地址范围（第 1 个 Slave 自身地址）。
- 软件配置 XMB_DEV_ADDR[xmb_address_en]使能 I2C Device 专用地址（第 2 个 Slave 自身地址），配置 XMB_DEV_ADDR[xmb_address]和 XMB_DEV_ADDR[xmb_address_mask]，调整 I2C Device 专用地址范围。

说明

地址 bit 位与掩码 bit 位一一对应：当掩码 bit 为 1，控制器 Slave 地址相应的 bit 位参与地址匹配对比。当掩码 bit 位为 0，控制器 Slave 地址相应的 bit 位不参与地址匹配对比。即，地址匹配的条件：控制器 Slave 的地址按位与掩码 = 接收到的地址按位与掩码。

2 个 Slave 地址其中任意 1 个匹配，即认为是 Slave 被寻址到。

I2C 控制器 Slave 支持 7bit 地址和 10bit 地址，软件配置 I2C_OWN_ADDR[I2C_10bit_slave_en]位域来使能 7bit 和 10bit 混合地址。

- 不使能该位域，Slave 处于 7bit 寻址模式，Slave 将 10bit Slave Addressing (0b1111 0xx) 视为普通地址，即使接收到也不进入 10bit 寻址模式。在该配置下，I2C_OWN_ADDR[own_address]低 7bit 和 I2C_OWN_ADDR[own_address_mask]低 7bit 被用于地址检查。在该配置下，建议 Slave 的自身地址避开 10bit Slave Addressing (0b1111 0xx) 特殊地址段。然

而，若 I2C 总线上无 10bit 器件，则 Slave 的自身地址可以配置在 10bit Slave Addressing (0b1111 0xx) 地址范围内。

- 使能该位域，Slave 处于 7bit 和 10bit 混合寻址模式，接收 10bit Slave Addressing (0b1111 0xx) 特殊地址，视为 10bit 寻址的第 1 个地址 Byte，Slave 进入 10bit 寻址模式，响应（返回 ACK）以此地址开始的帧。在该配置下，Slave 的自身地址配置须避开 10bit Slave Addressing (0b1111 0xx) 特殊地址段。

I2C 控制器支持 I2C 协议定义的 Slave 特殊地址（又称为保留地址段），软件需要使能特定的位域来使能 Slave 特殊地址，如表 19-2 所示。

说明

I2C Slave 自身地址的配置建议避开 Slave 特殊地址（保留地址段）。若配置在 Slave 特殊地址范围内，那么检测到 Slave 自身地址匹配时，Slave 也会接收并应答。

表19-2 I2C 协议 Slave 特殊地址（保留地址段）

Slave 地址	R/W bit	描述	使能配置
0b0000 000	0	General Call Address	I2C_OWN_ADDR[i2c_general_call_en]。
0b0000 000	1	START Byte Address	I2C_OWN_ADDR[i2c_start_byte_en]。
0b0000 001	X	CBUS Address	缺省不使能，不接收该地址，不应答。
0b0000 010	X	reserved for different bus format	缺省不使能，不接收该地址，不应答。
0b0000 011	X	reserved for future purpose	缺省不使能，不接收该地址，不应答。
0b0000 1XX	X	Hs-mode master code	缺省不使能，不接收该地址，不应答。
0b1111 1XX	1	Device ID Address	I2C_OWN_ADDR[i2c_device_id_en]。
0b1111 0XX	X	10-bit Slave Addressing	I2C_OWN_ADDR[i2c_10bit_slave_en]。

19.4.8 I2C 总线时序合成

I2C 控制器主机状态机和从机状态机通过按顺序执行的硬件 FIFO，及其中的{时序命令，数据}这种特殊的数据结构，合成标准时序和非标准时序。软件把期望执行的命令和待发送的数据以{时序命令，数据}的数据结构格式，写入 I2C_TX_FIFO 寄存器的

[mst_slv_cmd]和[tx_fifo_wdata]位域。I2C 控制器以先进先出的顺序读 TX FIFO，按顺序合成 I2C 时序。

表19-3 时序命令（mst_slv_cmd 位域）的主机和从机译码结果

时序命令十六进制	主机译码结果	从机译码结果
0x0	MST_INT1（Master 上报时序命令中断 1，根据 scl_stretch_disable 位域的配置，启动 SCL Stretch 功能，等待软件清除该中断后，才执行下一个时序命令）。	SLV_INT1（Slave 上报时序命令中断 1，根据 scl_stretch_disable 位域的配置，启动 SCL Stretch 功能，等待软件清除该中断后，才执行下一个时序命令）。
0x1	MST_S（Master 发送 START condition）。	SLV_S（Slave 不发送也不接收数据，进入空闲状态，等待下一个（re）START condition 和地址）。
0x2	MST_TD_RACK（Master 发送 1 个字节数据，每个字节发送后接收低电平 ACK）。	SLV_RD_TACK（Slave 接收 1 个字节数据，每个字节接收后发送低电平 ACK）。
0x3	MST_TD_RNACK（Master 发送 1 个字节数据，字节发送后接收高电平 NACK）。	SLV_RD_TNACK（Slave 接收 1 个字节数据，字节接收后发送高电平 NACK）。
0x4	MST_TD（Master 发送 1 个字节数据，无应答位）。	SLV_RD（Slave 接收 1 个字节数据，无应答位）。
0x5	MST_RD_TACK（Master 接收 1 个字节数据，每个字节接收后发送低电平 ACK）。	SLV_TD_RACK（Slave 发送 1 个字节数据，每个字节发送后接收低电平 ACK）。
0x6	MST_RD_TNACK（Master 接收 1 个字节数据，字节接收后发送高电平 NACK）。	SLV_TD_RNACK（Slave 发送 1 个字节数据，字节发送后接收高电平 NACK）。
0x7	MST_RD（Master 接收 1 个字节数据，无应答位）。	SLV_TD（Slave 发送 1 个字节数据，无应答位）。
0x8	保留。	保留。

时序命令十六进制	主机译码结果	从机译码结果
0x9	保留。	保留。
0xA	MST_TPEC_RACK（Master 发送 PEC 字节，接收低电平 ACK）。	SLV_RPEC_TACK（Slave 接收 PEC 字节，发送低电平 ACK）。
0xB	MST_TPEC_RNACK（Master 发送 PEC 字节，接收高电平 NACK）。	SLV_RPEC_TNACK（Slave 接收 PEC 字节，发送高电平 NACK）。
0xC	保留。	保留。
0xD	MST_RPEC_TACK（Master 接收 PEC 字节，发送低电平 ACK）。	SLV_TPEC_RACK（Slave 发送 PEC 字节，接收低电平 ACK）。
0xE	MST_RPEC_TNACK（Master 接收 PEC 字节，发送高电平 NACK）。	SLV_TPEC_RNACK（Slave 发送 PEC 字节，接收高电平 NACK）。
0xF	MST_P（Master 发送 STOP condition）。	SLV_P（Slave 不发送也不接收，进入空闲状态，等待下一个（re）START condition 和地址）。

19.4.8.1 I2C 7bit 寻址写操作标准时序

19.4.8.1.1 Master-Transmitter

Master 执行时序命令，发出 START condition（开始帧），寻址 7bit Slave 地址 + Write Bit（写操作），发送数据，接收 ACK/NACK，发出 STOP condition（结束帧）。如图 19-3 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-4 所示。

图19-3 I2C 7bit 寻址 Master-Transmitter 标准时序图

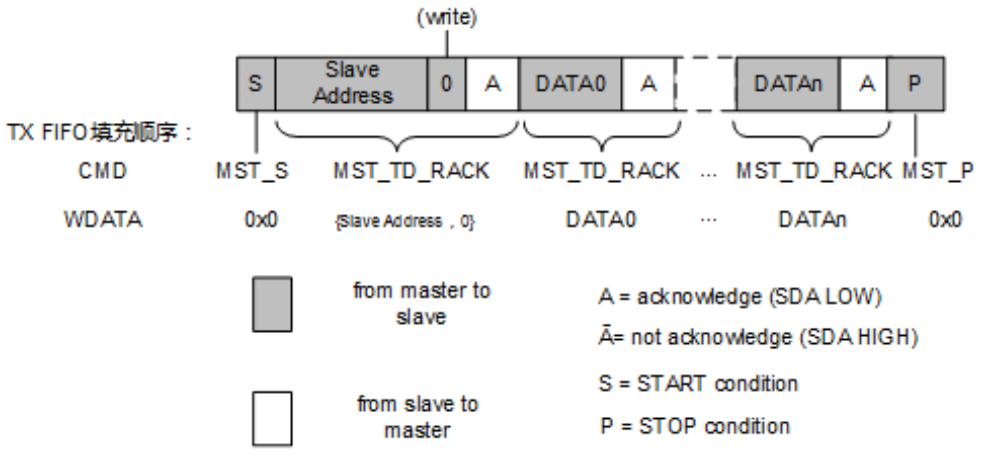


表19-4 I2C 7bit 寻址 Master-Transmitter 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，Slave Address，0 }
第 3 次	{ MST_TD_RACK，DATA0 }
第 4 次	{ MST_TD_RACK，DATA1 }
...	...
第 n+3 次	{ MST_TD_RACK，DATAn }
最后	{ MST_P，0x0 }

19.4.8.1.2 Slave-Receiver

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，接收数据，返回 ACK/NACK。Slave 执行 SLV_P 时序命令，可以提前停止数据接收，Slave 也可以自动检测 STOP condition，停止数据接收，如图 19-4 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-5 所示。

图19-4 I2C 7bit 寻址 Slave-Receiver 标准时序图

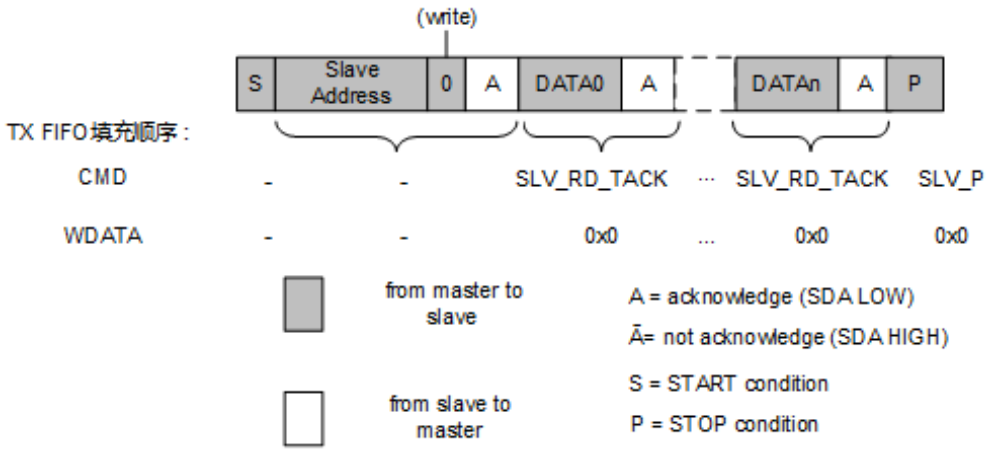


表19-5 I2C 7bit 寻址 Slave-Receiver 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_RD_TACK，0x0 }
第 2 次	{ SLV_RD_TACK，0x0 }
…	…
第 n+1 次	{ SLV_RD_TACK，0x0 }
最后	{ SLV_P，0x0 }

说明

Slave 执行 SLV_RD 系列时序命令，读数据 DATA0、DATA1、… DATAn 依次存入 I2C_RX_FIFO。

19.4.8.2 I2C 7bit 寻址读操作标准时序

19.4.8.2.1 Master-Receiver

Master 执行时序命令，发出 START condition（开始帧），寻址 7bit Slave 地址 + Read Bit（读操作），接收数据，发送 ACK/NACK，发出 STOP condition（结束帧）。Master 接收的数据被依次存入 RX FIFO 中，故软件需要读 RX FIFO 获取接收的数据，如图 19-5 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-6 所示。

图19-5 I2C 7bit 寻址 Master-Receiver 标准时序图

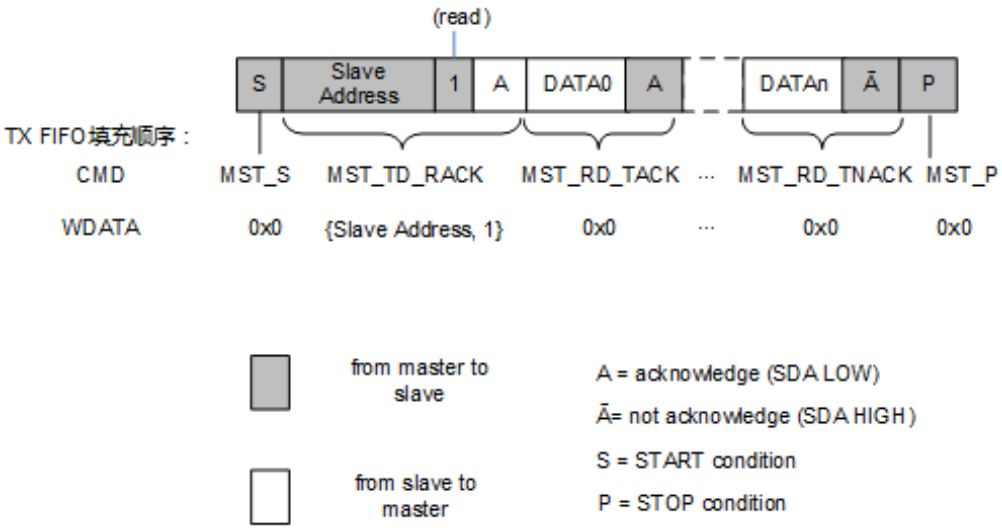


表19-6 I2C 7bit 寻址 Master-Receiver 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，Slave Address，1 }
第 3 次	{ MST_RD_TACK，0x0 }
第 4 次	{ MST_RD_TACK，0x0 }
...	...
第 n+3 次	{ MST_RD_TNACK，0x0 }
最后	{ MST_P，0x0 }

说明

Master 执行 MST_RD 系列时序命令，读数据 DATA0、DATA1、...DATAn 依次存入 I2C_RX_FIFO。

19.4.8.2.2 Slave-Transmitter

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，发送数据，接收 ACK/NACK。Slave 执行 SLV_P 时序命令，可以提前停止数据发

送，Slave 也可以自动检测 STOP condition，停止数据发送，如图 19-6 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-7 所示。

图19-6 I2C 7bit 寻址 Slave-Transmitter 标准时序图

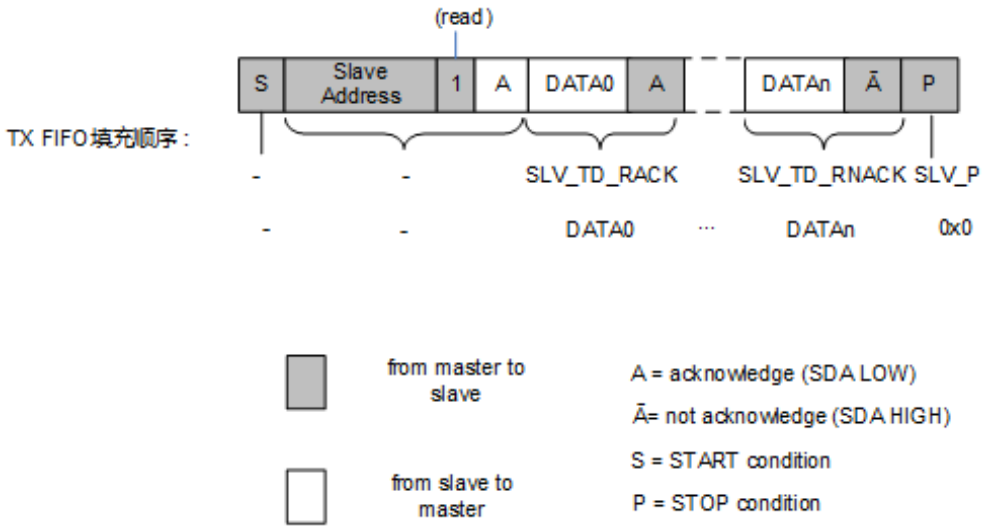


表19-7 I2C 7bit 寻址 Slave-Transmitter 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_TD_RACK，DATA0 }
第 2 次	{ SLV_TD_RACK，DATA1 }
...	...
第 n+1 次	{ SLV_TD_RNACK，DATAn }
最后	{ SLV_P，0x0 }

19.4.8.3 I2C 7bit 寻址写读操作组合标准时序

19.4.8.3.1 Master-Transmitter 和 Master-Receiver

Master 执行时序命令，发出 START condition（开始帧），寻址 7bit Slave 地址 + Write Bit（写操作），发送数据，接收 ACK/NACK，发出 ReSTART condition（开始新的操作），寻址 7bit Slave 地址 + Read Bit（读操作），接收数据，发送 ACK/NACK，

发出 STOP condition（结束帧）。如图 19-7 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-8 所示。

图19-7 I2C 7bit 寻址 Master-Transmitter 和 Master-Receiver 标准时序图

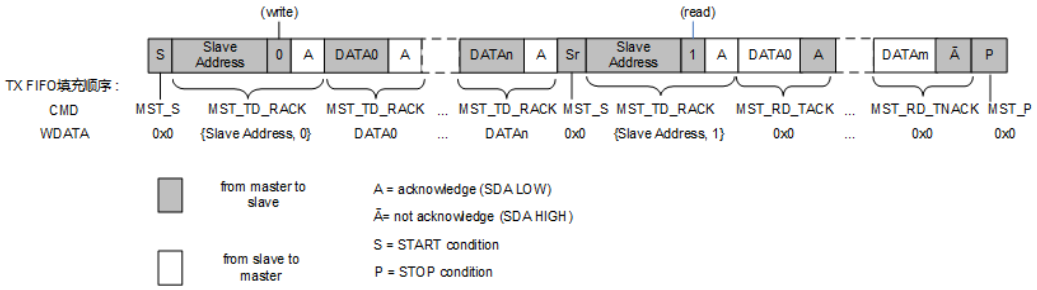


表19-8 I2C 7bit 寻址 Master-Transmitter 和 Master-Receiver 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，Slave Address，0 }
第 3 次	{ MST_TD_RACK，DATA0 }
第 4 次	{ MST_TD_RACK，DATA1 }
...	...
第 n+3 次	{ MST_TD_RACK，DATAn }
第 n+4 次	{ MST_S，0x0 }
第 n+5 次	{ MST_TD_RACK，Slave Address，1 }
第 n+6 次	{ MST_RD_TACK，0x0 }
第 n+7 次	{ MST_RD_TACK，0x0 }
...	...
第 n+m+6 次	{ MST_RD_TNACK，0x0 }
最后	{ MST_P，0x0 }

说明

Master 执行 MST_RD 系列时序命令，读数据 DATA0、DATA1、... DATAn 依次存入 I2C_RX_FIFO。

19.4.8.3.2 Slave-Receiver 和 Slave-Transmitter

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，接收数据，返回 ACK/NACK。Slave 自动接收 ReSTART condition，自动检测地址。当地址匹配后，Slave 执行命令序列，发送数据，接收 ACK/NACK。Slave 执行 SLV_P 时序命令，可以提前停止数据接收/发送，Slave 也可以自动检测 STOP condition，停止数据接收/发送。如图 19-8 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-9 所示。

图19-8 I2C 7bit 寻址 Slave-Receiver 和 Slave-Transmitter 标准时序图

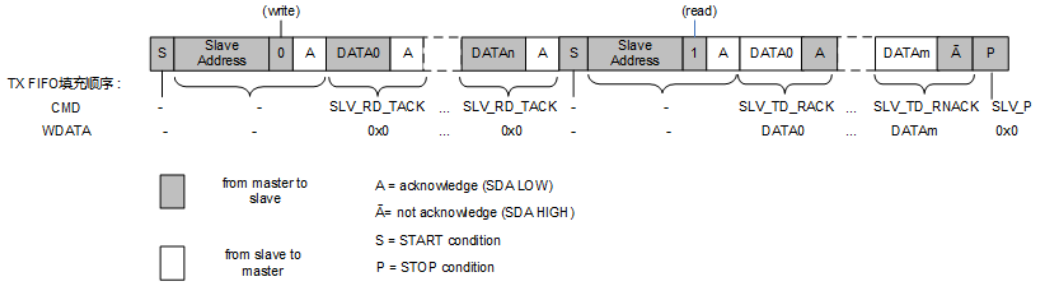


表19-9 I2C 7bit 寻址 Slave-Receiver 和 Slave-Transmitter 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_RD_TACK，0x0}
第 2 次	{ SLV_RD_TACK，0x0 }
...	...
第 n+1 次	{ SLV_RD_TACK，0x0 }
第 n+2 次	{ SLV_TD_RACK，DATA0 }
第 n+3 次	{ SLV_TD_RACK，DATA1 }
...	...
第 n+m+2 次	{ SLV_TD_RNACK，DATAm }

I2C_TX_FIFO 寄存器的写入次序	写入的值
最后	{ SLV _P，0x0 }

说明

Slave 执行 SLV_RD 系列时序命令，读数据 DATA0、DATA1、... DATAn 依次存入 I2C_RX_FIFO。

19.4.8.4 I2C 10bit 寻址写操作标准时序

19.4.8.4.1 Master-Transmitter

Master 执行时序命令，发出 START condition，寻址 10bit Slave 地址，发送数据，接收 ACK/NACK，发出 STOP condition，结束读操作。如图 19-9 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-10 所示。

图19-9 I2C 10bit 寻址 Master-Transmitter 标准时序图

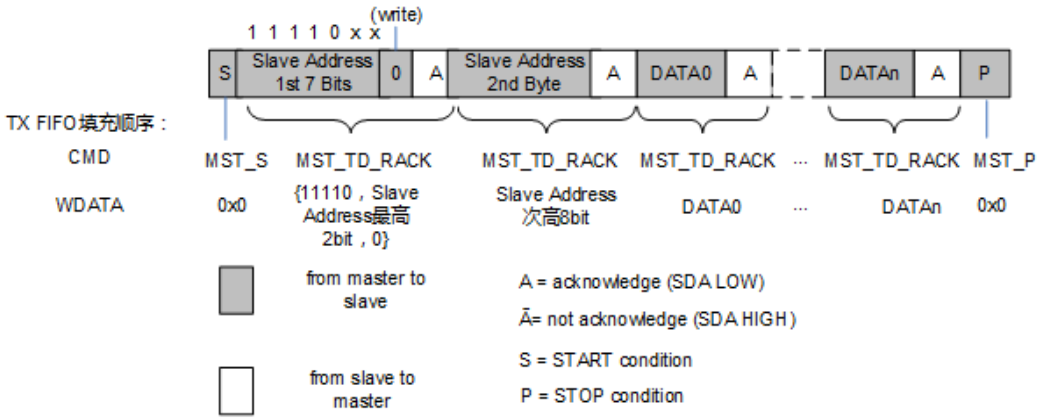


表19-10 I2C 10bit 寻址 Master-Transmitter 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，11110，Slave Address 最高 2bit，0 }
第 3 次	{ MST_TD_RACK，Slave Address 次高 8bit }
第 4 次	{ MST_TD_RACK，DATA0 }
第 5 次	{ MST_TD_RACK，DATA1 }

I2C_TX_FIFO 寄存器的写入次序	写入的值
...	...
第 n+4 次	{ MST_TD_RACK，DATAn }
最后	{ MST_P，0x0 }

19.4.8.4.2 Slave-Receiver

Slave 自动接收 START condition，自动检测 10bit 地址（软件需要在初始化 I2C Slave 时使能 I2C_OWN_ADDR[i2c_10bit_slave_en]位域）。当地址匹配后，Slave 执行命令序列，发送数据，接收 ACK/NACK。Slave 执行 SLV_P 时序命令，可以提前停止数据发送，Slave 也可以自动检测 STOP condition，停止数据发送。如图 19-10 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-11 所示。

图19-10 I2C 10bit 寻址 Slave-Receiver 标准时序图

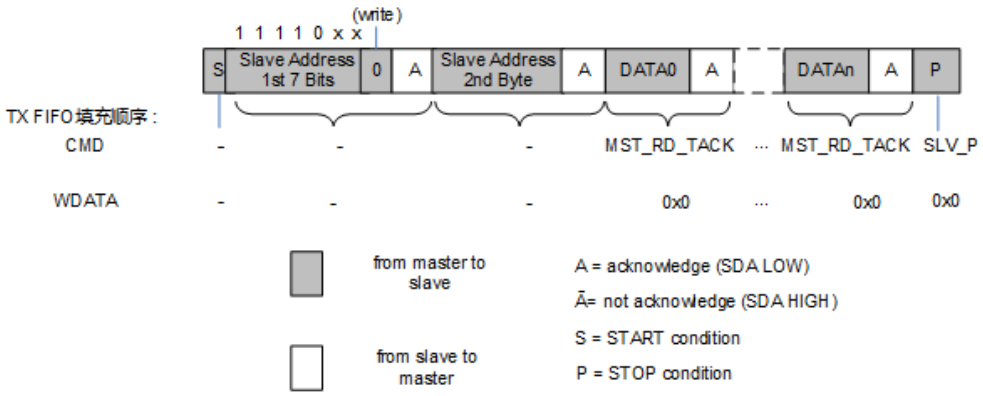


表19-11 I2C 10bit 寻址 Slave-Receiver 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_RD_TACK，0x0 }
第 2 次	{ SLV_RD_TACK，0x0 }
...	...
第 n+1 次	{ SLV_RD_TACK，0x0 }
最后	{ SLV_P，0x0 }

说明

Slave 执行 SLV_RD 系列时序命令，读数据 DATA0、DATA1、... DATAn 依次存入 I2C_RX_FIFO。

19.4.8.5 I2C 10bit 寻址读操作标准时序

19.4.8.5.1 Master-Receiver

Master 执行时序命令，发出 START condition（开始帧），寻址 10bit Slave 地址，接收数据，发送 ACK/NACK，发出 STOP condition（结束帧）。Master 接收的数据被依次存入 RX FIFO 中，故软件需要读 RX FIFO 获取接收的数据。如图 19-11 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-12 所示。

图19-11 I2C 10bit 寻址 Master-Receiver 标准时序图

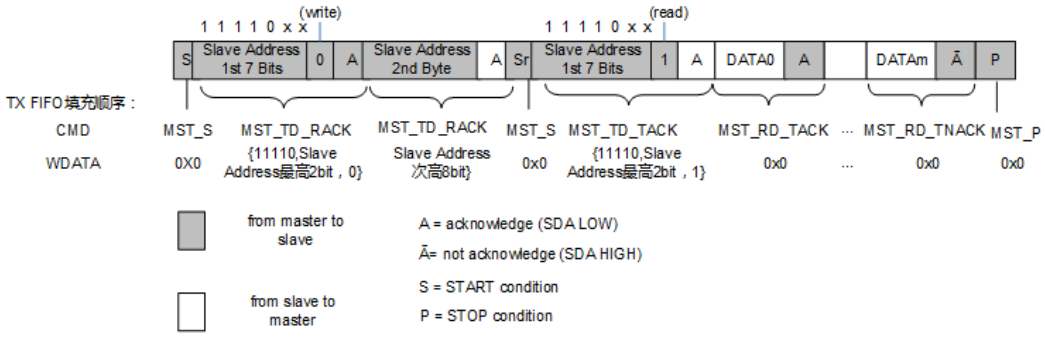


表19-12 I2C 10bit 寻址 Master-Receiver 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，11110，Slave Address 最高 2bit，0 }
第 3 次	{ MST_TD_RACK，Slave Address 次高 8bit }
第 4 次	{ MST_S，0x0 }
第 5 次	{ MST_TD_RACK，11110，Slave Address 最高 2bit，1 }
第 6 次	{ MST_RD_TACK，0x0 }

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 7 次	{ MST_RD_TACK，0x0 }
...	...
第 n+6 次	{ MST_RD_TNACK，0x0 }
最后	{ MST_P，0x0 }

说明

Master 执行 MST_RD 系列时序命令，读数据 DATA0、DATA1、... DATAn 依次存入 I2C_RX_FIFO。

19.4.8.5.2 Slave-Transmitter

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，发送数据，接收 ACK/NACK。Slave 执行 SLV_P 时序命令，可以提前停止数据发送，Slave 也可以自动检测 STOP condition，停止数据发送。如图 19-12 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-13 所示。

图19-12 I2C 10bit 寻址 Slave-Transmitter 标准时序图

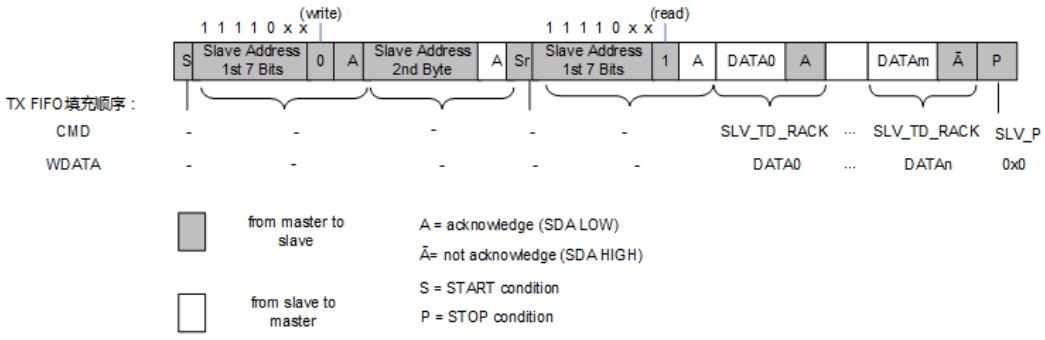


表19-13 I2C 10bit 寻址 Slave-Transmitter 标准时序配置序列

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_TD_RACK，DATA0 }
第 2 次	{ SLV_TD_RACK，DATA1 }
...	...

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 n+1 次	{ SLV_TD_RNACK，DATAn }
最后	{ SLV_P，0x0 }

19.4.8.6 I2C 非标准时序（例子 1）

19.4.8.6.1 Master-Transmitter

I2C 非标准协议，图 19-13 为一个非标准时序，此时序约束 Master 连续向 Slave 发送数据不接受 ACK 应答，发送最后一个数据才接收一个 ACK 应答。如图 19-13 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-14 所示。

图19-13 I2C Master-Transmitter 非标准时序图（例子 1）

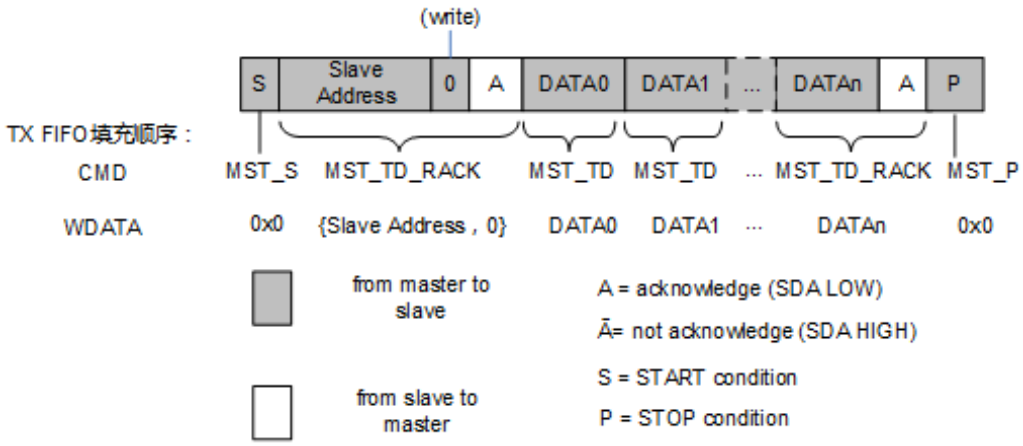


表19-14 I2C Master-Transmitter 非标准时序配置序列（例子 1）

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，Slave Address，0 }
第 3 次	{ MST_TD，DATA0 }
第 4 次	{ MST_TD，DATA1 }
...	...

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 n+3 次	{ MST_TD_RACK，DATAn }
最后	{ MST_P，0x0 }

19.4.8.6.2 Slave-Receiver

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，连续接收数据不发送 ACK 应答，接收最后一个数据才发送一个 ACK 应答。Slave 执行 SLV_P 时序命令，可以提前停止数据接收，Slave 也可以自动检测 STOP condition，停止数据接收。如图 19-14 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-15 所示。

图19-14 I2C Slave-Receiver 非标准时序图（例子 1）

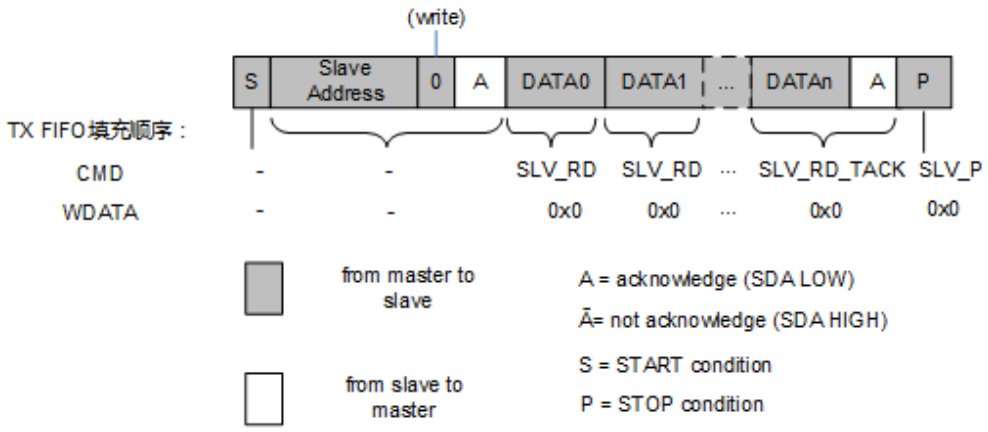


表19-15 I2C Slave-Receiver 非标准时序配置序列（例子 1）

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_RD，0x0 }
第 2 次	{ SLV_RD，0x0 }
...	...
第 n+1 次	{ SLV_RD_TACK，0x0 }
最后	{ SLV_P，0x0 }

19.4.8.6.3 Master-Receiver

I2C 非标准协议，图 19-15 为一个非标准时序，此时序约束 Master 连续从 Slave 接收数据不发送 ACK 应答，读到最后一个数据才发送一个 ACK 应答。如图 19-15 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-16 所示。

图19-15 I2C Master-Receiver 非标准时序图（例子 1）

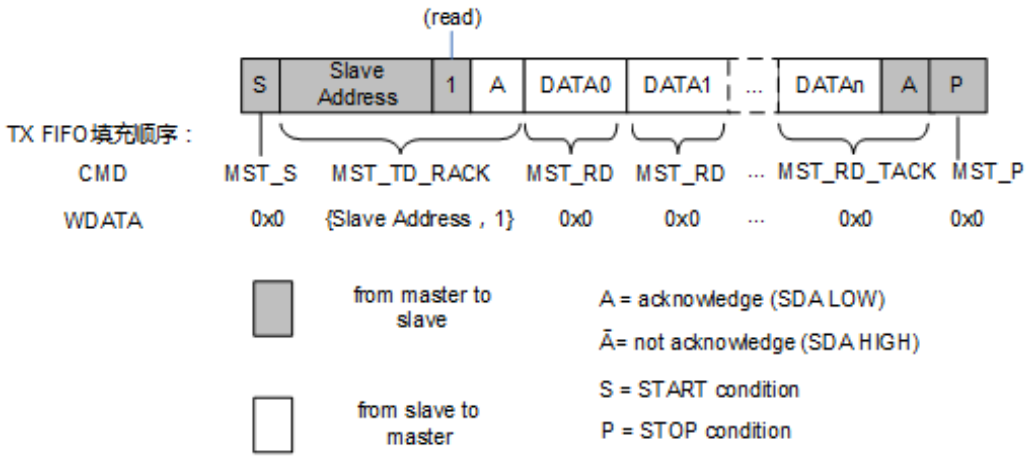


表19-16 I2C Master-Receiver 非标准时序配置序列（例子 1）

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ MST_S，0x0 }
第 2 次	{ MST_TD_RACK，Slave Address，1 }
第 3 次	{ MST_RD，0x0 }
第 4 次	{ MST_RD，0x0 }
...	...
第 n+3 次	{ MST_RD_TACK，0x0 }
最后	{ MST_P，0x0 }

19.4.8.6.4 Slave-Transmitter

Slave 自动接收 START condition，自动检测地址。当地址匹配后，Slave 执行命令序列，连续发送数据不接收 ACK 应答，发送最后一个数据才接收一个 ACK 应答。Slave 执行 SLV_P 时序命令，可以提前停止数据发送，Slave 也可以自动检测 STOP

condition，停止数据发送。如图 19-16 所示。软件需要将{时序命令，数据}依次写入 TX FIFO，如表 19-17 所示。

图19-16 I2C Slave- Transmitter 非标准时序图（例子 1）

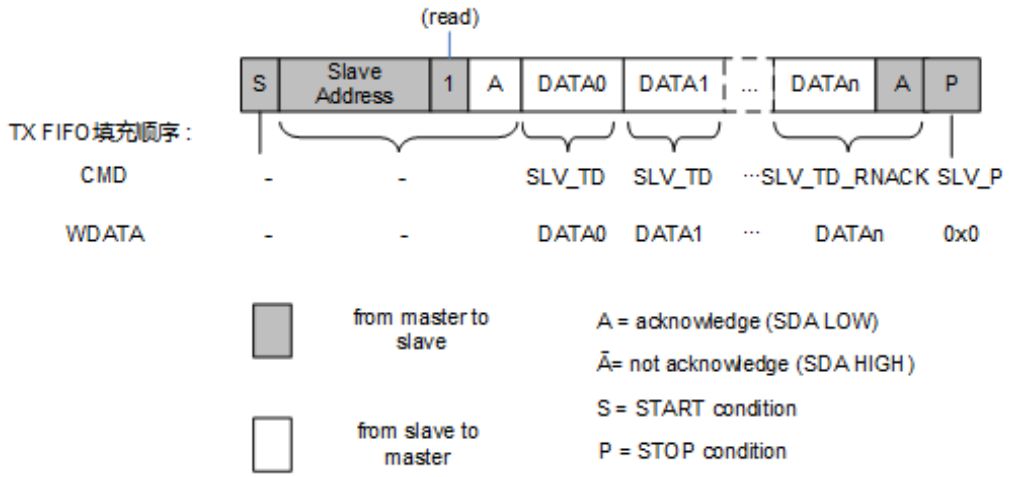


表19-17 I2C Slave-Receiver 非标准时序配置序列（例子 1）

I2C_TX_FIFO 寄存器的写入次序	写入的值
第 1 次	{ SLV_TD，DATA0 }
第 2 次	{ SLV_TD，DATA1 }
...	...
第 n+1 次	{ SLV_TD_RNACK，DATAn }
最后	{ SLV_P，0x0 }

19.4.9 Bus Clear 和 GPIO 模式

软件配置 I2C_CTRL2[gpio_mode]，进入 GPIO 模式。软件可配置 I2C_CTRL2[force_scl]和 I2C_CTRL2[force_sda]，控制 SCL 和 SDA 管脚。

I2C 通信中，SCL 时钟线不太可能有 stuck-at-0（常 0）的情况，SDA 数据线有可能有常 0 的情况。

- 若 SCL 时钟线常 0，优先采取的措施是配置 I2C_MODE[mst_slv_function]，不使能 I2C 控制器然后重新使能，以重启 I2C 控制器。若 SCL 时钟线仍为常 0，则是

I2C 总线上的其他设备挂死，须使用硬件复位信号复位 I2C 总线设备，或者重新下电上电挂死的设备。

- 若 SDA 数据线常 0，可能被 I2C 总线上的其他设备错误地输出常 0。软件需要先配置 I2C_MODE[mst_slv_function]，关闭 I2C 控制器，然后执行 I2C 协议规定的 Bus clear 流程。具体操作是软件配置 I2C_CTRL2[gpio_mode]，在 GPIO 模式下，控制 I2C_CTRL2[force_scl]，在 SCL 时钟线上发出 9 个时钟脉冲。SDA 输出常 0 的 I2C 设备应该在这 9 个时钟的某位置，释放总线。如果没有释放，那么需要使用硬件复位或者下电上电异常设备。

19.4.10 DMA 操作

I2C 的 FIFO 支持 DMA 操作。TX FIFO 支持 DMA 存储器到外设（TX FIFO），外设到外设（TX FIFO）传输类型。RX FIFO 支持 DMA 外设（RX FIFO）到存储器，外设（RX FIFO）到外设传输类型，支持外设流控。软件配置 I2C_CTRL1[dma_operation]，调整 I2C 外设请求的使能和禁能。软件配置 I2C_CTRL1[dma_rx_lsreq_en]，使能 I2C RX 外设流控。

说明

I2C RX 外设流控下，当 I2C 检测到 I2C Bus Free（总线空闲）后，通过 I2C RX 外设流控请求线，通知 DMA 搬移 RX FIFO 中的最后 1 个数据，DMA 搬移完成后自动关闭通道。

19.4.11 I2C 中断

I2C 模块的中断状态分别是：

- Slave 检测到地址匹配中断状态（slv_addr_match_int）。
- Slave 接收应答位非预期中断状态（slv_rx_ack_unmatch_int）。
- Slave 接收 PEC 错误中断状态（slv_pec_check_fail_int）。
- Master 接收 PEC 错误中断状态（mst_pec_check_fail_int）。
- Slave 时序命令中断状态（slv_cmd_int1）。
- Master 时序命令中断状态（mst_cmd_int1）。
- SMBus SMBSUS#信号下降沿中断状态（smb_suspend_int）。
- SMBus SMBALERT#信号下降沿中断状态（smb_alert_int）。
- SCL 低电平超时中断状态（scl_low_timeout_int）。
- Master 时序命令正常完成中断状态（mst_cmd_done_int）。
- 仲裁丢失中断状态（arb_lost_int）。

- START 检测中断状态（start_det_int）。
- STOP 检测中断状态（stop_det_int）。
- TX FIFO 请求中断状态（tx_data_request_int）。
- RX FIFO 请求中断状态（rx_data_ready_int）。
- TX FIFO 非满中断状态（tx_fifo_not_full_int）。
- TX FIFO 空中断状态（tx_fifo_empty_int）。
- TX FIFO 水线中断（tx_le_watermark_int）。
- RX FIFO 满中断（rx_fifo_full_int）。
- RX FIFO 水线中断（rx_ge_watermark_int）。
- RX FIFO 非空中断（rx_fifo_not_empty_int）。
- Master 接收应答位非预期中断状态（mst_rx_ack_unmatch_int）。

表19-18 I2C 模块中断

I2C 中断状态	原始中断寄存器	中断使能寄存器	中断清除方法	中断上报条件
slv_addr_match_int	slv_addr_match_raw	slv_addr_match_en	软件置位 slv_addr_match_raw	I2C Slave，根据 I2C_OWN_ADDR 寄存器的配置，接收到匹配的 Slave 地址。
slv_rx_ack_unmatch_int	slv_rx_ack_unmatch_raw	slv_rx_ack_unmatch_en	软件置位 slv_rx_ack_unmatch_raw	I2C Slave，接收到的 ACK/NACK 不匹配，非预期。
slv_pec_check_fail_int	slv_pec_check_fail_raw	slv_pec_check_fail_en	软件置位 slv_pec_check_fail_raw	I2C Slave，接收到的 PEC 校验错误。
mst_pec_check_fail_int	mst_pec_check_fail_raw	mst_pec_check_fail_en	软件置位 mst_pec_check_fail_raw	I2C Master，接收到的 PEC 校验错误。
slv_cmd_int1	slv_cmd_int1_raw	slv_cmd_int1_en	软件置位 slv_cmd_int1_raw	I2C Slave，执行 SLV_INT1 命令。
mst_cmd_int1	mst_cmd_raw	mst_cmd_int1_en	软件置位 mst_cmd_int1_raw	I2C Master，执行 MST_INT1 命令。
smb_suspend_int	smb_suspend_raw	smb_suspend_en	软件置位	SMBSUS#信号下降

I2C 中断状态	原始中断寄存器	中断使能寄存器	中断清除方法	中断上报条件
			smb_suspend_raw	沿。
smb_alert_int	smb_alert_raw	smb_alert_en	软件置位 smb_alert_raw	SMBALERT#信号下降沿。
scl_low_timeout_int	scl_low_timeout_raw	scl_low_timeout_en	软件置位 scl_low_timeout_raw	SCL 处于低电平的时间超过 I2C_SCL_TIMEOUT 配置的时间。
mst_cmd_done_int	mst_cmd_done_raw	mst_cmd_done_en	软件置位 mst_cmd_done_raw	I2C Master，完成 STOP Conditon 的发送。
arb_lost_int	arb_lost_raw	arb_lost_en	软件置位 arb_lost_raw	I2C 仲裁丢失。
start_det_int	start_det_raw	start_det_en	软件置位 start_det_raw	I2C 模块检测到 I2C 总线上有 START Condition。
stop_det_int	stop_det_raw	stop_det_en	软件置位 stop_det_raw	I2C 模块检测到 I2C 总线上有 STOP Condition。
tx_data_request_int	tx_data_request_raw	tx_data_request_en	软件置位 tx_data_request_raw	I2C Master，在通信期间，TX FIFO 空了，TX FIFO 请求新的{时序命令，数据}数据结构。
rx_data_ready_int	rx_data_ready_raw	rx_data_ready_en	软件置位 rx_data_ready_raw	I2C 的 RX FIFO 接收到了新的数据。
tx_fifo_not_full_int	tx_fifo_not_full_raw	tx_fifo_not_full_en	软件置位 tx_fifo_not_full_raw	I2C 的 TX FIFO 非满。
tx_fifo_empty_int	tx_fifo_empty_raw	tx_fifo_empty_en	软件置位 tx_fifo_empty_raw	I2C 的 TX FIFO 空。
tx_le_watermark_int	tx_le_watermark_raw	tx_le_watermark_en	软件置位 tx_le_watermark_raw	I2C 的 TX FIFO 填充水平小于等于 I2C_TX_WATERMARK

I2C 中断状态	原始中断寄存器	中断使能寄存器	中断清除方法	中断上报条件
				寄存器配置的值
rx_fifo_full_int	rx_fifo_full_raw	rx_fifo_full_en	软件置位 rx_fifo_full_raw	I2C 的 RX FIFO 满、
rx_ge_watermark_int	rx_ge_watermark_raw	rx_ge_watermark_en	软件置位 rx_ge_watermark_raw	I2C 的 RX FIFO 填充水平大于等于 I2C_RX_WATERMARK 寄存器配置的值。
rx_fifo_not_empty_int	rx_fifo_not_empty_raw	rx_fifo_not_empty_en	软件置位 rx_fifo_not_empty_raw	I2C 的 RX FIFO 非空。
mst_rx_ack_unmatch_int	mst_rx_ack_unmatch_raw	mst_rx_ack_unmatch_en	软件置位 mst_rx_ack_unmatch_raw	I2C Master，接收的 ACK/NACK 不匹配，非预期。

19.5 工作方式

19.5.1 初始化配置

软件需要先完成 19.5.1.1 I2C 控制器的时钟和复位章节，然后无先后顺序地完成本章节的其余的初始化配置，最后完成 19.5.1.3.5 I2C 控制器 Master 和 Slave 使能。

19.5.1.1 I2C 控制器的时钟和复位

I2C 工作时钟为 clk_i2c。在初始化 I2C 之前，需要复位 I2C 模块，打开 I2C 时钟，解除 I2C 模块的复位。

- 复位：软件须配置 CRG 模块的 PERI_CRG112[i2c0_srst_req]，复位 I2C0，或 PERI_CRG113[i2c1_srst_req]，复位 I2C1。
- 时钟的使能：软件须配置 CRG 模块的 PERI_CRG112[i2c0_cken]，使能 I2C0 的时钟，或 PERI_CRG113[i2c1_cken]，使能 I2C1 的时钟。
- 解复位：软件须配置 CRG 模块的 PERI_CRG112[i2c0_srst_req]，解除 I2C0 的复位，或 PERI_CRG113[i2c1_srst_req]，解除 I2C1 的复位。

19.5.1.2 I2C 总线配置

19.5.1.2.1 I2C 总线 SCL 速率配置

SCL 周期时间对 I2C 控制器的速率起决定性影响，也就是 SCL 的低电平时间和高电平时间。软件需要配置 I2C_SCL_CFG[scl_high_time]和[scl_low_time]，来调整 SCL 高电平时长和低电平时长。IO 驱动能力，板级上拉电阻和板级负载会影响t_r、t_f。

- SCL 高电平时间计算公式是
  $T_{clk_i2c} \times (\text{scl_high_time} + 8 + \text{spike_filter_time}) + \frac{t_r + t_f}{2}$
- SCL 低电平时间计算公式是$T_{clk_i2c} \times (\text{scl_low_time} + 9 + \text{spike_filter_time}) + \frac{t_r + t_f}{2}$
- SCL 周期计算公式是
  $T_{clk_i2c} \times (\text{scl_high_time} + \text{scl_low_time} + 17 + \text{spike_filter_time} \times 2) + t_r + t_f$

建议 SCL 高电平时间，在标准模式下为 SCL 周期的 1/2，快速模式下为 SCL 周期的 0.36。

- 标准模式下，建议$\text{scl_high_time} = (f_{clk\ i2c}/f_{SCL}) \text{ x } 0.5 - 8 - \text{spike_filter_time}$
- 快速模式下，建议$\text{scl_high_time} = (f_{clk\ i2c}/f_{SCL}) \text{ x } 0.36 - 8 - \text{spike_filter_time}$

建议 SCL 低电平时间，在标准模式下为 SCL 周期的 1/2，快速模式下为 SCL 周期的 0.64。

- 标准模式下，建议$\text{scl_low_time} = (f_{clk\ i2c}/f_{SCL}) \text{ x } 0.5 - 9 - \text{spike_filter_time}$
- 快速模式下，建议$\text{scl_low_time} = (f_{clk\ i2c}/f_{SCL}) \text{ x } 0.64 - 9 - \text{spike_filter_time}$

19.5.1.2.2 I2C 总线 SDA 建立保持时间配置

软件需要配置 I2C_SDA_CFG[sda_hold_time]来直接调整 SDA 保持时间。建立时间由 SCL 时钟低电平时间减去 SDA 保持时间间接得出，请参见 19.4.1.2 SDA 配置章节。

19.5.1.2.3 I2C 总线滤波/时钟超时和空闲配置

- 软件配置 I2C_FILTER[spike_filter_time]，调整滤波功能。对 SCL 和 SDA 滤波，最大可过滤 I2C 工作时钟 15 个周期的毛刺。
- 软件配置 I2C_SCL_TIMEOUT[scl_low_timeout]，调整 SCL 低电平超时的阈值时间。在 I2C 总线挂死时，上报中断（scl_low_timeout 中断），便于软件做通信恢复。
- 软件配置 I2C_BUS_FREE[bus_free_time]，调整总线空闲检测的阈值时间。

19.5.1.2.4 I2C 总线数据传输位序配置

软件可配置 I2C_MODE[lit_end]，调整数据传输位序。该功能请参见 19.4.5 数据传输位序章节。

19.5.1.3 Master 和 Slave 的配置

19.5.1.3.1 SCL Stretching 配置

I2C 控制器 Master（主机）和 Slave（从机）均支持 SCL Stretching，该功能缺省使能。软件可配置 I2C_MODE[scl_stretch_disable]同时关闭 Master 和 Slave 的 SCL Stretching。该功能请参见 19.4.3 SCL 延长章节。

19.5.1.3.2 接收 ACK 和 NACK 模式配置

软件可配置 I2C_MODE[rack_mode]，调整接收 ACK/NACK 的模式。该功能请参见 19.4.4 ACK 和 NACK 章节。

19.5.1.3.3 Slave 自身地址和特殊地址配置

软件配置 I2C_OWN_ADDR 寄存器，配置 XMB_DEV_ADDR 寄存器，完成 Slave 自身地址和特殊地址的配置，请参见 19.4.7 Slave 自身地址和特殊地址章节。

19.5.1.3.4 FIFO 配置

软件配置 I2C_TX_WATERMARK[tx_watermark]，配置 I2C_RX_WATERMARK[rx_watermark]，完成 FIFO 水线的配置，请参见 19.4.2 FIFO 章节。

软件配置 I2C_CTRL1[rst_tx_fifo]和 I2C_CTRL1[rst_rx_fifo]位域复位 TX FIFO 和 RX FIFO。

19.5.1.3.5 I2C 控制器 Master 和 Slave 使能

软件配置 I2C_MODE[mst_slv_function]来配置 I2C 控制器处于 Master-Only 模式，或者 Slave-Only 模式，或者 Master-Slave 模式，或者关闭 I2C 控制器，请参见 19.4.6 Master 和 Slave 章节。

19.5.2 中断方式下的写操作数据传输

19.5.2.1 Master-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x1811，打开 mst_cmd_done 中断，arb_lost 中断，tx_le_watermark 中断，mst_rx_ack_unmatch 中断。

步骤 3 在中断服务程序里，查询 I2C 中断状态寄存器 I2C_INTR_STAT。

- 若 mst_cmd_done_int 中断状态为 1，表示所有时序命令已完成，所有数据已经被发送，则跳到步骤 5；
- 若 arb_lost_int 中断状态为 1，在多 Master 总线拓扑结构中，表示总线上其他 Master 器件获得了总线仲裁，软件需要重发，具体操作是跳转到步骤 5，之后清空 TX FIFO 和 RX FIFO 并重新执行写操作。在单 Master 总线拓扑结构中，表示软件配置的时序命令未与 Slave 支持的时序匹配，请参见 19.4.8 I2C 总线时序合成章节，修改软件以适配 Slave 器件的时序格式；

说明

使用控制器 Master 时，需要适配 Slave 设备所支持的时序格式。否则，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 控制器 Master 需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 器件正在输出数据 0，会拉低 SDA。此时 Master 无法结束 I2C 帧的通信（即无法完成 STOP 条件，超过 8192 个 I2C 工作时钟后，Master 仲裁丢失，上报 arb_lost_int 中断）。所以，在对接 Slave 器件时，软件需要配置时序命令，使得 I2C 控制器 Master 适配此时 Slave 器件支持的时序。

- 若 mst_rx_ack_unmatch_int 中断状态为 1，则认为是异常；
- 若 tx_le_watermark_int 中断状态为 1，软件需要根据对接 Slave 器件的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Master 自动读出时序命令，开始合成时序）。

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。在中断服务程序里，软件写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

步骤 4 清除或者禁能 I2C 中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除触发本次中断处理的 tx_le_watermark 中断，然后等待下一次中断服务程序被触发；
- 如果已经没有待配置的{时序命令，数据}，则配置 I2C_INTR_EN 为，禁能 tx_le_watermark 中断。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.2.2 Slave-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x300200，打开 slv_addr_match_int 中断，slv_rx_ack_unmatch_int 中断，stop_det_int 中断。

步骤 3 在中断服务程序里，查询 I2C 中断状态寄存器 I2C_INTR_STAT。

- 若 slv_addr_match_int 中断状态为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit。配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x300214，打开 tx_le_watermark_int 中断，rx_ge_watermark_int 中断；
- 若 slv_rx_ack_unmatch_int 中断状态为 1，则认为是异常；
- 若 stop_det_int 中断状态为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望读到的数据个数，软件需要读 RX FIFO 获取 Slave 接收到的数据直到 RX FIFO 空。如果 RX FIFO 读空后，仍未达到期望读到的数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果 RX FIFO 读空后，达到了期望读到的数据个数，表示传输完成，跳转到步骤 5 执行；

说明

Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

- 若 tx_le_watermark_int 中断状态为 1，软件需要根据期望 Slave 支持的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Slave 自动读出时序命令，开始合成时序）；

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。在中断服务程序里，软件写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

- 若 rx_ge_watermark_int 中断状态为 1，软件需要读 RX FIFO 获取 Slave 接收到的数据直到 RX FIFO 空。

说明

软件可以读 I2C_FSM_STAT[rx_fifo_vld_num]获取 RX FIFO 状态。

步骤 4 清除或者禁能 I2C 中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除触发本次中断处理的 tx_le_watermark 中断，然后等待下一次中断服务程序被触发；
- 如果已经没有待配置的{时序命令，数据}，则配置 I2C_INTR_EN 寄存器的 tx_le_watermark_en 位域为 0，禁能 tx_le_watermark 中断；
- RX FIFO 空后，如果还未达到期望读到的数据个数，则配置原始中断寄存器 I2C_INTR_RAW 为 0x4，清除触发本次中断处理的 rx_ge_watermark 中断，然后等待下一次中断服务程序被触发；
- 如果已经达到期望读到的数据个数，则配置配置 I2C_INTR_EN 寄存器的 rx_ge_watermark_en 位域为 0，禁能 rx_ge_watermark 中断。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.3 中断方式下的读操作数据传输

19.5.3.1 Master-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x1845，打开 mst_cmd_done 中断，arb_lost 中断，tx_fifo_not_full 中断，rx_ge_watermark 中断，mst_rx_ack_unmatch 中断。

步骤 3 在中断服务程序里，查询 I2C 中断状态寄存器 I2C_INTR_STAT。

- 若 mst_cmd_done_int 中断状态为 1，表示所有时序命令已完成，所有数据已经被发送，则跳到步骤 5；
- 若 arb_lost_int 中断状态为 1，表示总线上其他 Master 器件获得了总线仲裁，Master 仲裁丢失；
- 若 mst_rx_ack_unmatch_int 中断状态为 1，则认为是异常；
- 若 tx_fifo_not_full_int 中断状态为 1，软件需要根据对接 Slave 器件的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Master 自动读出时序命令，开始合成时序）；

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。在中断服务程序里，软件写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT 寄存器的 tx_fifo_vld_num 位域获取 TX FIFO 状态。

- 若 rx_ge_watermark_int 中断状态为 1，软件需要读 RX FIFO 获取 Master 接收到的数据直到 RX FIFO 空。

说明

软件可以读 I2C_FSM_STAT[rx_fifo_vld_num]获取 RX FIFO 状态。

步骤 4 清除或者禁能 I2C 中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除触发本次中断处理的 tx_le_watermark 中断，然后等待下一次中断服务程序被触发；
- 如果已经没有待配置的{时序命令，数据}，则配置 I2C_INTR_EN 寄存器的 tx_fifo_not_full_en 位域为，禁能 tx_fifo_not_full_int 中断。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.3.2 Slave-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x200210，打开 slv_addr_match_int 中断，stop_det_int 中断，tx_le_watermark_int 中断。

步骤 3 在中断服务程序里，查询 I2C 中断状态寄存器 I2C_INTR_STAT。

- 若 slv_addr_match_int 中断状态为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit。配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x300214，打开 tx_le_watermark_int 中断，rx_ge_watermark_int 中断；
- 若 stop_det_int 中断状态为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望发送数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果达到了期望发送数据的个数，表示总线通信正常结束。跳转到步骤 5；

  说明

  Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

- 若 tx_le_watermark_int 中断状态为 1，软件需要根据期望 Slave 支持的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Slave 自动读出时序命令，开始合成时序）。

  说明

  时序命令的配置总次数可能超过 TX FIFO 的深度。在中断服务程序里，软件写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

步骤 4 清除或者禁能 I2C 中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除触发本次中断处理的 tx_le_watermark 中断，然后等待下一次中断服务程序被触发；
- 如果已经没有待配置的{时序命令，数据}，则配置 I2C_INTR_EN 寄存器的 tx_le_watermark_en 位域为 0，禁能 tx_le_watermark 中断。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.4 查询方式下的写操作数据传输

19.5.4.1 Master-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，关闭所有中断。

步骤 3 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 mst_cmd_done_raw 原始中断为 1，表示所有时序命令已完成，所有数据已经被发送，则跳到步骤 5，否则不断执行步骤 3；
- 若 arb_lost_raw 原始中断为 1，在多 Master 总线拓扑结构中，表示总线上其他 Master 器件获得了总线仲裁，软件需要重发，具体操作是跳转到步骤 5，之后清空 TX FIFO 和 RX FIFO 并重新执行写操作。在单 Master 总线拓扑结构中，表示软件配置的时序命令未与 Slave 支持的时序匹配，请参见 19.4.8 I2C 总线时序合成章节，修改软件以适配 Slave 器件的时序格式；

说明

使用 I2C 控制器 Master 时，需要适配 Slave 设备所支持的时序格式。否则，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 控制器 Master 需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 器件正在输出数据 0，会拉低 SDA。此时 Master 无法结束 I2C 帧的通信（即无法完成 STOP 条件，超过 8192 个 I2C 工作时钟后，Master 仲裁丢失，上报 arb_lost_int 中断）。所以，在对接 Slave 器件时，软件需要配置时序命令，使得 I2C 控制器 Master 适配此时 Slave 器件支持的时序。

- 若 mst_rx_ack_unmatch_raw 原始中断为 1，则认为是异常；
- 若 tx_le_watermark_raw 原始中断为 1，软件需要根据对接 Slave 器件的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Master 自动读出时序命令，开始合成时序）。

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。软件轮询 TX FIFO 状态，写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

步骤 4 清除原始中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除 tx_le_watermark 原始中断，然后回到步骤 3；
- 如果已经没有待配置的{时序命令，数据}，然后回到步骤 3。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.4.2 Slave-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，关闭所有中断。

步骤 3 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 slv_addr_match_raw 原始中断为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit，继续查询其余的原始中断。否则，循环执行步骤 3 查询 slv_addr_match_raw 原始中断，直到超时；
- 若 slv_rx_ack_unmatch_raw 原始中断为 1，则认为是异常；
- 若 stop_det_int 中断状态为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望读到的数据个数，软件需要读 RX FIFO 获取 Slave 接收到的数据直到 RX FIFO 空。如果 RX FIFO 读空后，仍未达到期望读到的数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果 RX FIFO 读空后，达到了期望读到的数据个数，表示传输完成，跳转到步骤 5 执行；

说明

Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

- 若 tx_le_watermark_raw 原始中断为 1，软件需要根据期望 Slave 支持的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Slave 自动读出时序命令，开始合成时序）；

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。软件查询 TX FIFO 状态，写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT 寄存器的 tx_fifo_vld_num 位域获取 TX FIFO 状态。

- 若 rx_ge_watermark_raw 原始状态为 1，软件需要读 RX FIFO 获取 Slave 接收到的数据直到 RX FIFO 空。

软件可以读 I2C_FSM_STAT[rx_fifo_vld_num]获取 RX FIFO 状态。

步骤 4 清除原始中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除 tx_le_watermark 原始中断，然后回到步骤 3。
- 如果已经没有待配置的{时序命令，数据}，然后回到步骤 3。
- RX FIFO 空后，如果还未达到期望读到的数据个数，则配置原始中断寄存器 I2C_INTR_RAW 为 0x4，清除 rx_ge_watermark 原始中断，然后回到步骤 3。
- 如果已经达到期望读到的数据个数，则配置原始中断寄存器 I2C_INTR_RAW 为 0x4，清除 rx_ge_watermark 原始中断，然后执行步骤 5。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.5 查询方式下的读操作数据传输

19.5.5.1 Master-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，关闭所有中断。

步骤 3 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 mst_cmd_done_raw 原始中断为 1，表示所有时序命令已完成，所有数据已经被发送，软件读 RX FIFO 获取 Master 接收到的数据直到 RX FIFO 空，跳到步骤 4；
- 若 arb_lost_raw 原始中断为 1，表示总线上其他 Master 器件获得了总线仲裁，Master 仲裁丢失，软件根据应用决定是否重新启动传输；

- 若 mst_rx _ack_unmatch_raw 原始中断为 1，则认为是异常；
- 若 tx_fifo_not_full_raw 原始中断为 1，软件需要根据对接 Slave 器件的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Master 自动读出时序命令，开始合成时序）。

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。软件轮询 TX FIFO 状态，写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

- 若 rx_ge_watermark_raw 原始中断为 1，软件需要读 RX FIFO 获取 Master 接收到的数据直到 RX FIFO 空。

说明

软件可以读 I2C_FSM_STAT 寄存器的 rx_fifo_vld_num 位域获取 RX FIFO 状态。

步骤 4 清除原始中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除 tx_le_watermark 原始中断，然后回到步骤 3；
- 如果已经没有待配置的{时序命令，数据}，然后回到步骤 3；
- RX FIFO 空后，如果还未达到期望读到的数据个数，则配置原始中断寄存器 I2C_INTR_RAW 为 0x4，清除 rx_ge_watermark 原始中断，然后回到步骤 3；
- 如果已经达到期望读到的数据个数，则配置原始中断寄存器 I2C_INTR_RAW 为 0x4，清除 rx_ge_watermark 原始中断，然后执行步骤 5。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.5.2 Slave-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，关闭所有中断。

步骤 3 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 slv_addr_match_raw 原始中断为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit，继续查询其余的原始中断。否则，循环执行步骤 3 查询 slv_addr_match_raw 原始中断，直到超时；
- 若 stop_det_raw 原始中断为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望发送数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果达到了期望发送数据的个数，表示总线通信正常结束。跳转到步骤 5；

说明

Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

- 若 tx_le_watermark_raw 原始中断为 1，软件需要根据期望 Slave 支持的时序，写 TX FIFO，完成时序命令的配置，请参见 19.4.8 I2C 总线时序合成章节（I2C 控制器 Slave 自动读出时序命令，开始合成时序）。

说明

时序命令的配置总次数可能超过 TX FIFO 的深度。软件轮询 TX FIFO 状态，写数据入 TX FIFO 直到 TX FIFO 满，或者直到没有待写的{时序命令，数据}。软件可以读 I2C_FSM_STAT[tx_fifo_vld_num]获取 TX FIFO 状态。

步骤 4 清除原始中断。

- TX FIFO 满后，如果还有待配置的{时序命令，数据}，则配置原始中断寄存器 I2C_INTR_RAW 为 0x10，清除 tx_le_watermark 原始中断，然后回到步骤 3；
- 如果已经没有待配置的{时序命令，数据}，然后回到步骤 3。

步骤 5 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.6 DMA 方式下的写操作数据传输

19.5.6.1 Master-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 软件需要根据对接 Slave 器件的时序，声明{时序命令，数据}的数据结构类型的数组，并做初始化赋值，请参见 19.4.8 I2C 总线时序合成章节。

步骤 3 获取一个 DMA 通道，用于 I2C TX 方向，类型为存储器到外设（TX FIFO）。

步骤 4 完成 DMA 通道的设置并启动，请参见 10 直接存储器访问控制器(DMA)章节。

步骤 5 等待 DMA 完成中断。DMA 完成中断表示 DMA 从源地址搬移了已配置的数据量到 I2C TX FIFO 中。不表示 I2C 控制器将 TX FIFO 中的{时序命令，数据}都执行完毕。

步骤 6 不断查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 mst_cmd_done_raw 原始中断为 1，表示所有时序命令已完成，所有数据已经被发送，则跳到步骤 7，否则不断执行步骤 6；
- 若 arb_lost_raw 原始中断为 1，在多 Master 总线拓扑结构中，表示总线上其他 Master 器件获得了总线仲裁，软件需要重发，具体操作是跳转到步骤 5，之后清空 TX FIFO 和 RX FIFO 并重新执行写操作。在单 Master 总线拓扑结构中，表示软件配置的时序命令未与 Slave 支持的时序匹配，请参见 19.4.8 I2C 总线时序合成章节，修改软件以适配 Slave 器件的时序格式；

说明

使用控制器 Master 时，需要适配 Slave 设备所支持的时序格式。否则，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 控制器 Master 需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 器件正在输出数据 0，会拉低 SDA。此时 Master 无法结束 I2C 帧的通信（即无法完成 STOP 条件，超过 8192 个 I2C 工作时钟后，Master 仲裁丢失，上报 arb_lost_int 中断）。所以，在对接 Slave 器件时，软件需要配置时序命令，使得 I2C 控制器 Master 适配此时 Slave 器件支持的时序。

- 若 mst_rx_ack_unmatch_raw 原始中断为 1，则认为是异常。

步骤 7 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.6.2 Slave-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 软件需要根据对接 Slave 器件的时序，声明{时序命令，数据}的数据结构类型的数组，并做初始化赋值，请参见 19.4.8 I2C 总线时序合成章节。

步骤 3 获取两个 DMA 通道，一个通道用于 I2C TX 方向，类型为存储器到外设（TX FIFO），一个通道用于 I2C RX 方向，类型为外设（RX FIFO）到存储器。

步骤 4 完成两个 DMA 通道的设置并启动，请参见 10 直接存储器访问控制器(DMA)章节。

步骤 5 等待 RX 方向 DMA 通道的完成中断。RX 方向 DMA 通道完成后，表示 DMA 从 I2C RX FIFO 中搬走了 DMA 已配置的数据量，搬到了目的地址。

步骤 6 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 slv_addr_match_raw 原始中断为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit，继续查询其余的原始中断。否则，循环执行步骤 6 查询 slv_addr_match_raw 原始中断，直到超时；
- 若 slv_rx_ack_unmatch_raw 原始中断为 1，则认为是异常；
- 若 stop_det_int 中断状态为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望读到的数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果达到了期望读到的数据个数，表示传输完成，跳转到步骤 7 执行。

说明

Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

步骤 7 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.7 DMA 方式下的读操作数据传输

19.5.7.1 Master-Receiver

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 软件需要根据对接 Slave 器件的时序，声明{时序命令，数据}的数据结构类型的数组，并做初始化赋值，请参见 19.4.8 I2C 总线时序合成章节。

步骤 3 获取两个 DMA 通道，一个通道用于 I2C TX 方向，类型为存储器到外设（TX FIFO），一个通道用于 I2C RX 方向，类型为外设（RX FIFO）到存储器。

步骤 4 完成两个 DMA 通道的设置并启动，请参见 10 直接存储器访问控制器(DMA)章节。

步骤 5 等待 RX 方向 DMA 通道的完成中断。RX 方向 DMA 通道完成后，表示 DMA 从 I2C RX FIFO 中搬走了 DMA 已配置的数据量，搬到了目的地址。

步骤 6 不断查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 mst_cmd_done_raw 原始中断为 1，表示所有时序命令已完成，所有数据已经被发送，则跳到步骤 7，否则不断执行步骤 6；
- 若 arb_lost_raw 原始中断为 1，在多 Master 总线拓扑结构中，表示总线上其他 Master 器件获得了总线仲裁，软件需要重发，具体操作是跳转到步骤 5，之后清空 TX FIFO 和 RX FIFO 并重新执行写操作。在单 Master 总线拓扑结构中，表示软件配置的时序命令未与 Slave 支持的时序匹配，请参见 19.4.8 I2C 总线时序合成章节，修改软件以适配 Slave 器件的时序格式；

说明

使用控制器 Master 时，需要适配 Slave 设备所支持的时序格式。否则，有可能提前结束 I2C 帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 控制器 Master 需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 器件正在输出数据 0，会拉低 SDA。此时 Master 无法结束 I2C 帧的通信（即无法完成 STOP 条件，超过 8192 个 I2C 工作时钟后，Master 仲裁丢失，上报 arb_lost_int 中断）。所以，在对接 Slave 器件时，软件需要配置时序命令，使得 I2C 控制器 Master 适配此时 Slave 器件支持的时序。

- 若 mst_rx_ack_unmatch_raw 原始中断为 1，则认为是异常。

步骤 7 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.7.2 Slave-Transmitter

请遵循以下步骤：

步骤 1 完成 I2C 控制器的初始化，请参见 19.5.1 初始化配置章节。

步骤 2 软件需要根据对接 Slave 器件的时序，声明{时序命令，数据}的数据结构类型的数组，并做初始化赋值，请参见 19.4.8 I2C 总线时序合成章节。

步骤 3 获取一个 DMA 通道，用于 I2C TX 方向，类型为存储器到外设（TX FIFO）。

步骤 4 完成 DMA 通道的设置并启动，请参见 10 直接存储器访问控制器(DMA)章节。

步骤 5 等待 DMA 完成中断。DMA 完成中断表示 DMA 从源地址搬移了已配置的数据量到 I2C TX FIFO 中。不表示 I2C 控制器将 TX FIFO 中的{时序命令，数据}都执行完毕。

步骤 6 查询 I2C 原始中断寄存器 I2C_INTR_RAW。

- 若 slv_addr_match_raw 原始中断为 1，表示 Slave 检测到地址匹配，Slave 被寻址到，软件读 I2C_RX_ADDR 寄存器可得到 Slave 接收到的地址和 R/Wbit，继续查询其余的原始中断。否则，循环执行步骤 6 查询 slv_addr_match_raw 原始中断，直到超时；
- 若 stop_det_raw 原始中断为 1，表示 I2C 总线上的 Master 设备已结束 I2C 当前帧的通信，此时如果未达到期望发送数据个数，表示总线上的 Master 设备提前结束了 I2C 帧的通信。如果达到了期望发送数据的个数，表示总线通信正常结束。跳转到步骤 7。

说明

Master 设备在未适配 Slave 设备所支持的时序格式之前，有可能提前结束帧的通信（也就是帧的长度短于 Slave 支持的时序）。在此情形下，I2C 总线上的 Master 设备需要在 SCL 高电平期间借助板级上拉电阻拉高 SDA（完成 STOP 条件），正常情况下，STOP 条件能完成，但是可能发生的特殊情况是 Slave 正在输出数据 0，会拉低 SDA。此时 Master 设备无法结束 I2C 帧的通信（即无法完成 STOP 条件）。所以，在对接 I2C 控制器的 Slave 时，Master 设备那一侧需要适配此时 Slave 支持的时序（支持的时序由软件定义）。

步骤 7 配置 I2C 中断使能寄存器 I2C_INTR_EN 为 0x0，I2C 原始中断寄存器 I2C_INTR_RAW 为 0x3FFFFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

----结束

19.5.8 异常处理流程

请遵循以下步骤：

步骤 1 配置 I2C 全局配置寄存器 I2C_MODE 的 mst_slv_function 为 0，不使能控制器。

步骤 2 配置中断使能寄存器 I2C_INTR_EN 为 0x0，I2C_INTR_RAW 为 0x1FFF，清除中断并且禁止中断上报，为不同工作方式之间的切换做好清理工作。

步骤 3 配置 I2C_CTRL2 寄存器的 gpio_mode 为 1，进入 GPIO 模式。

步骤 4 配置 I2C_CTRL2 寄存器的 force_scl 和 force_sda，执行 Bus Clear 流程，请参见 19.4.9 Bus Clear 和 GPIO 模式章节。

----结束

19.6 I2C 寄存器概览

地址空间内：

- I2C0 寄存器基地址为：0x1410_0000。
- I2C1 寄存器基地址为：0x1410_1000。

I2C 寄存器概览如表 19-19 所示。

表19-19 I2C 寄存器概览

偏移地址	名称	描述	页码
0x0000	I2C_MODE	I2C 模式配置寄存器	I2C_MODE
0x0004	I2C_SCL_CFG	I2C SCL 高低电平时间寄存器	I2C_SCL_CFG
0x0008	I2C_SDA_CFG	I2C SDA 时序配置寄存器	I2C_SDA_CFG
0x000C	I2C_OWN_ADDR	I2C Slave 地址配置寄存器	I2C_OWN_ADDR
0x0010	XMB_DEV_ADDR	I2C SMBus PMBus Device 专用地址配置寄存器	XMB_DEV_ADDR
0x0018	I2C_RX_ADDR	I2C Slave 已接收的地址，R/Wbit 寄存器	I2C_RX_ADDR
0x0020	I2C_TX_FIFO	I2C TX FIFO 数据寄存器	I2C_TX_FIFO
0x0024	I2C_RX_FIFO	I2C RX FIFO 数据寄存器	I2C_RX_FIFO

偏移地址	名称	描述	页码
0x00C8	I2C_TX_WATERMARK	I2C 发送水线寄存器	I2C_TX_WATERMARK
0x00CC	I2C_RX_WATERMARK	I2C 接收水线寄存器	I2C_RX_WATERMARK
0x00D0	I2C_CTRL1	I2C 控制寄存器 1	I2C_CTRL1
0x00D4	I2C_CTRL2	I2C 控制寄存器 2	I2C_CTRL2
0x00D8	I2C_FIFO_STAT	I2C FIFO 状态寄存器	I2C_FIFO_STAT
0x00DC	I2C_FSM_STAT	I2C 状态机状态寄存器	I2C_FSM_STAT
0x00E0	I2C_INTR_RAW	I2C 原始中断寄存器	I2C_INTR_RAW
0x00E4	I2C_INTR_EN	I2C 中断使能寄存器	I2C_INTR_EN
0x00E8	I2C_INTR_STAT	I2C 中断状态寄存器	I2C_INTR_STAT
0x0100	I2C_VERSION	I2C 版本号寄存器	I2C_VERSION
0x0104	I2C_SCL_TIMEOUT	I2C SCL 超时阈值寄存器	I2C_SCL_TIMEOUT
0x0108	I2C_BUS_FREE	I2C 总线空闲阈值寄存器	I2C_BUS_FREE
0x010C	I2C_FILTER	I2C SDA 和 SCL 滤波配置寄存器	I2C_FILTER

19.7 I2C 寄存器描述

I2C_MODE

I2C_MODE 为 I2C 模式配置寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5]	RW	scl_stretch_disable	SCL Stretching 使能位。使能后若条件满足，拉低 SCL 并保持到条件不满足。不使能时，即使条件满足，SCL 也不会被拉低。该使能位同时对 Master 和 Slave 生效。 0：使能； 1：不使能。	0x0
[4]	RW	rack_mode	接收 ACK/NACK 的模式。 0：严格 ACK/NACK。如果 Master-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，那么发出 STOP condition，自动退出 Master。如果 Slave-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，那么 Slave 退出当前 I2C 帧的通信，SDA 输出高阻。 1：忽略 ACK/NACK。如果 Master-Transmitter，或 Slave-Transmitter 接收到的 ACK/NACK 与配置的时序命令所期望的 ACK/NACK 不匹配，忽略此不匹配，继续下一个时序命令。	0x0
[3]	RW	xmb_pec_en	SMBus 和 PMBus 的 PEC（CRC-8，多项式 0x07）计算使能。使能此位后，I2C 控制器会自动计算 PEC。软件需要通过时序命令控制是否发送 PEC 和接收 PEC。	0x0

Bits	Access	Name	Description	Reset
			0：SMBus PEC 不使能计算； 1：SMBus PEC 使能计算。	
[2]	RW	lit_end	数据传输顺序。对 Master 和 Slave 功能有效。 0：MSbit-First 模式； 1：LSbit-First 模式。	0x0
[1:0]	RW	mst_slv_function	Master 功能和 Slave 功能选择。 00：控制器不使能； 01：控制器处于 Master-only 模式； 10：控制器处于 Slave-only 模式； 11：控制器处于 Master-Slave 模式。	0x0

I2C_SCL_CFG

I2C_SCL_CFG 为 I2C SCL 高低电平时间寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RW	scl_high_time	SCL 高电平时长。Master 有效。 建议在标准模式下为 SCL 周期的 1/2，快速模式下为 SCL 周期的 0.36。 标准模式下（f_{SCL} 为 SCL 时钟线的频率）： scl_high_time = (f_{clk_i2c}/f_{SCL}) x 0.5 - 8 - spike_filter_time 快速模式下（FSCL 为 SCL 时钟线的频率）： scl_high_time =(f_{clk_i2c}/f_{SCL}) x 0.36 - 8 - spike_filter_time 以参考时钟 f_{clk_i2c}=100MHz 且工作在快速模式下（f_{SCL} = 400kHz）为例，若不滤波，scl_high_time =(100MHz / 400kHz) x	0x0000

Bits	Access	Name	Description	Reset
			0.36 - 8 = 82。	
[15:0]	RW	scl_low_time	SCL 低电平时长。Master 有效。 建议在标准模式下为 SCL 周期的 1/2，快速模式下为 SCL 周期的 0.64。 标准模式下（f_{SCL} 为 SCL 时钟线的频率）： scl_low_time = (f_{clk_i2c}/f_{SCL})x 0.5 - 9 - spike_filter_time 快速模式下（f_{SCL} 为 SCL 时钟线的频率）： scl_low_time =(f_{clk_i2c}/f_{SCL})x 0.64 - 9 - spike_filter_time 以参考时钟 f_{clk_i2c}=100MHz 且工作在快速模式下（f_{SCL} = 400 kHz）为例，若不滤波，scl_low_time =(100MHz / 400kHz) x 0.64 - 9 = 151。	0x0000

I2C_SDA_CFG

I2C_SDA_CFG 为 I2C SDA 时序配置寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RW	sda_hold_time	SDA 保持时间配置。 建议配置为 sda_hold_duration = SCL 周期 × 0.3。 **注意：只能控制 SDA 输出的 hold time 时长。**	0x0000
[15:4]	-	reserved	保留。	0x000
[3:0]	RW	sda_delay_time	SDA 延迟采样配置。延迟 sda_delay_time 个 clk_i2c 工作时钟周期。 **注意：对 SDA 进行 delay 是指从 SDA 管脚进来的信号经过寄存后再送给内部控制**	0x0

Bits	Access	Name	Description	Reset
			器。此域决定 SDA 延迟多少个工作时钟周期后才进行采样。	

I2C_OWN_ADDR

I2C_OWN_ADDR 为 I2C Slave 地址配置寄存器。

Offset Address: 0x000C Total Reset Value: 0x003F_F000

Bits	Access	Name	Description	Reset
[31:28]	-	reserved	保留。	0x0
[27]	RW	i2c_10bit_slave_en	I2C 控制器 Slave 接收 10bit Slave Addressing (0b1111 0xx)，也正常接收 7bit 地址。 0：不使能，即 7bit 寻址模式；Slave 将 10bit Slave Addressing (0b1111 0xx)视为普通地址，I2C_OWN_ADDR 寄存器的 own_address 位域低 7bit 和 own_address_mask 位域低 7bit 被用于地址检查； 1：使能，即 7bit 和 10bit 混合寻址模式。Slave 接收 10bit Slave Addressing (0b1111 0xx)，视为 10bit 寻址的第 1 个地址 Byte，响应（返回 ACK）以此地址开始的帧。	0x0
[26]	RW	i2c_start_byte_en	I2C 控制器 Slave 使能接收 START Byte Address。 0：不使能，Slave 不接收 START Byte Address (0b00000001)，不响应（返回 NACK）以此地址开始的帧； 1：使能，Slave 接收 START Byte Address (0b00000001)，根据协议，不响应（返回 NACK）以此地址开始的帧。	0x0
[25]	RW	i2c_device_id_en	I2C 控制器 Slave 使能接收 Device ID	0x0

Bits	Access	Name	Description	Reset
			Address。 0：不使能，Slave 不接收 Device ID Address (0b11111xx1)，不响应（返回 NACK）以此地址开始的帧； 1：使能，Slave 接收 Device ID Address (0b11111xx1)，响应（返回 ACK）以此地址开始的帧。	
[24]	RW	i2c_general_call_en	I2C 控制器 Slave 使能接收 General Call Address。 0：不使能，Slave 不接收 General Call Address (0b00000000)，不响应（返回 NACK）以此地址开始的帧； 1：使能，Slave 接收 General Call Address (0b00000000)，响应（返回 ACK）以此地址开始的帧。	0x0
[23:22]	-	reserved	保留。	0x0
[21:12]	RW	own_address_mask	I2C 控制器 Slave 的地址掩码。当掩码 bit 为 1，控制器 Slave 地址相应的 bit 位参与地址匹配对比。当掩码 bit 位 0，控制器 Slave 地址相应的 bit 位不参与地址匹配对比。即，地址匹配的条件：控制器 Slave 的地址按位与掩码=接收到的地址按位与掩码。	0x3FF
[11:10]	-	reserved	保留。	0x0
[9:0]	RW	own_address	I2C 控制器 Slave 的地址。7bit 寻址时，低 7bit 被用于检查地址匹配。10bit 寻址时，整个 10bit 全部被用于检查地址匹配。地址匹配时，响应（返回 ACK）Master 发出的地址；地址不匹配时，不响应（返回 NACK）Master 发出的地址。	0x000

XMB_DEV_ADDR

XMB_DEV_ADDR 为 I2C SMBus PMBus Device 专用地址配置寄存器。

Offset Address: 0x0010 Total Reset Value: 0x003F_F000

Bits	Access	Name	Description	Reset
[31:30]	-	reserved	保留。	0x0
[29]	RW	pmb_zone_write_en	I2C 控制器 Slave 使能接收 PMBus ZONE WRITE Address。 0：不使能，Slave 不接收 PMBus ZONE WRITE Address (0b0110111)，不响应（返回 NACK）以此地址开始的帧; 1：使能，Slave 接收 PMBus ZONE WRITE Address (0b0110111)，响应（返回 ACK）以此地址开始的帧。	0x0
[28]	RW	pmb_zone_read_en	I2C 控制器 Slave 使能接收 PMBus ZONE READ Address。 0：不使能，Slave 不接收 PMBus ZONE READ Address (0b0101000)，不响应（返回 NACK）以此地址开始的帧; 1：使能，Slave 接收 PMBus ZONE READ Address (0b0101000)，响应（返回 ACK）以此地址开始的帧。	0x0
[27]	-	reserved	保留。	0x0
[26]	RW	smb_dev_default_en	I2C 控制器 Slave 使能接收 SMBus Device Default Address。 0：不使能，Slave 不接收 SMBus Device Default Address (0b1100001)，不响应（返回 NACK）以此地址开始的帧; 1：使能，Slave 接收 SMBus Device Default Address (0b1100001)，响应（返回 ACK）以此地址开始的帧。	0x0
[25]	RW	smb_alert_response_en	I2C 控制器 Slave 使能接收 SMBus Alert Response Address。	0x0

Bits	Access	Name	Description	Reset
			0：不使能，Slave 不接收 SMBus Alert Response Address (0b0001100)，不响应（返回 NACK）以此地址开始的帧; 1：使能，Slave 接收 SMBus Alert Response Address (0b0001100)，响应（返回 ACK）以此地址开始的帧。	
[24]	RW	smb_host_notify_en	I2C 控制器 Slave 使能接收 SMBus Host Address。 0：不使能，Slave 不接收 SMBus Host Address (0b0001000)，不响应（返回 NACK）以此地址开始的帧; 1：使能，Slave 接收 SMBus Host Address (0b0001000)，响应（返回 ACK）以此地址开始的帧。	0x0
[23]	-	reserved	保留。	0x0
[22]	RW	xmb_address_en	I2C SMBus PMBus Device 专用地址使能。 0：不使能，Slave 不接收 xmb_address，不响应（返回 NACK）； 1：使能，Slave 接收 xmb_address，掩码后匹配时，响应（返回 ACK）。	0x0
[21:12]	RW	xmb_address_mask	I2C SMBus PMBus Device 专用地址掩码。当掩码 bit 为 1，控制器 Slave 专用地址相应的 bit 位参与地址匹配对比。当掩码 bit 位 0，控制器 Slave 专用地址相应的 bit 位不参与地址匹配对比。即，地址匹配的条件：控制器 Slave 的地址按位与掩码 = 接收到的地址按位与掩码。	0x3FF
[11:10]	-	reserved	保留。	0x0
[9:0]	RW	xmb_address	I2C SMBus PMBus Device 专用地址。7bit 寻址时，低 7bit 被用于检查地址匹配。10bit 寻址时，整个 10bit 全部被用于	0x000

Bits	Access	Name	Description	Reset
			检查地址匹配。地址匹配时，响应（返回 ACK）Master 发出的地址；地址不匹配时，不响应（返回 NACK）Master 发出的地址。	

I2C_RX_ADDR

I2C_RX_ADDR 为 I2C Slave 已接收的地址，R/Wbit 寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:11]	RO	reserved	保留。	0x000000
[10:1]	RO	rx_addr	Slave 已接收到的地址。	0x000
[0]	RO	rx_rw	Slave 已接收到的 R/Wbit。	0x0

I2C_TX_FIFO

I2C_TX_FIFO 为 I2C TX FIFO 数据寄存器。是命令 FIFO 和 TX 数据 FIFO 的入口，FIFO 深度 16。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:8]	WO	mst_slv_cmd	该位域是 I2C 控制器 Master 时序命令 FIFO 入口。软件配置 mst_start 后，Master 开始工作，执行 Master CMD FIFO 中的命令，直到 P 时序命令执行完成，退出 Master 功能。同时该位域也是，I2C 控制器 Slave 时序命令 FIFO 入口。当接收到 START condition 或者 repeated START condition，Master 器件发出的地址匹配且 R/W 位为 1，开始执行 FIFO 中的命令。当 Slave 检测到 Master 器件发出	0x0

Bits	Access	Name	Description	Reset
			STOP condition 时，自动结束时序命令的执行。 0x0：MST_INT1 或者 SLV_INT1； 0x1：MST_S 或者 SLV_S； 0x2：MST_TD_RACK 或者 SLV_RD_TACK； 0x3：MST_TD_RNACK 或者 SLV_RD_TNACK； 0x4：MST_TD 或者 SLV_RD； 0x5：MST_RD_TACK 或者 SLV_TD_RACK； 0x6：MST_RD_TNACK 或者 SLV_TD_RNACK； 0x7：MST_RD 或者 SLV_TD； 0x8：保留； 0x9：保留； 0xA：MST_TPEC_RACK 或者 SLV_RPEC_TACK； 0xB：MST_TPEC_RNACK 或者 SLV_RPEC_TNACK； 0xC：保留； 0xD：MST_RPEC_TACK 或者 SLV_TPEC_RACK； 0xE：MST_RPEC_TNACK 或者 SLV_TPEC_RNACK； 0xF：MST_P 或者 SLV_P。 **注意：** **1. 执行发送数据命令后，若 TX FIFO 为空，控制器自动开启 Clock Stretch 功能（SCL 拉低并保持），直到 TX FIFO 非空；**	

Bits	Access	Name	Description	Reset
			2. 执行接收数据命令后，若 RX FIFO 满了，控制器自动开启 Clock Stretch 功能（SCL 拉低并保持），直到 RX FIFO 非满。	
[7:0]	WO	tx_fifo_wdata	TX FIFO 入口。只写。 软件写入待发送出去的数据。	0x00

I2C_RX_FIFO

I2C_RX_FIFO 为 I2C RX FIFO 数据寄存器。是 RX 数据 FIFO 的出口，FIFO 深度 16。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7:0]	RO	rx_fifo_rdata	RX FIFO 出口。只读。 软件读出 I2C 总线上收到的数据。	0x00

I2C_TX_WATERMARK

I2C_TX_WATERMARK 为 I2C 发送水线寄存器。

Offset Address: 0x00C8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	tx_watermark	TX FIFO 水线。影响 TX FIFO 水线中断，和 I2C TX FIFO 的 DMA Burst 请求。TX FIFO 有效数目小于等于 TX FIFO 水线时，产生 TX FIFO 水线原始中断，产生 TX FIFO 的 DMA Burst 请求。 0x0：等于 0； 0x1：小于或等于 1； 0x2：小于或等于 2；	0x0

Bits	Access	Name	Description	Reset
			0x3：小于或等于 3； ... 0xF：小于或等于 15。	

I2C_RX_WATERMARK

I2C_RX_WATERMARK 为 I2C 接收水线寄存器。

Offset Address: 0x00CC Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	rx_watermark	RX FIFO 水线。影响 RX FIFO 水线中断，和 I2C RX FIFO 的 DMA Burst 请求。RX FIFO 有效数目大于等于 RX FIFO 水线时，产生 RX FIFO 水线原始中断，产生 RX FIFO 的 DMA Burst 请求。 0x0：大于或等于 0； 0x1：大于或等于 1； 0x2：大于或等于 2； 0x3：大于或等于 3； ... 0xF：大于或等于 15。 **注意：该位域配置 0 时，当 RX FIFO 为空时也会产生 RX FIFO 的 DMA Burst 请求，故 DMA 传输方式，禁止软件配置为 0，否则 DMA 可能从 RX FIFO 中读到错误数据。**	0x0

I2C_CTRL1

I2C_CTRL1 为 I2C 控制寄存器 1。

Offset Address: 0x00D0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:11]	-	reserved	保留。	0x000000
[10]	RW	dma_rx_lsreq_en	DMA I2C 模块 RX 外设流控使能，是否使能 I2C 模块外设流控请求线。该位域受 dma_operation 位域控制。I2C RX 外设流控下，当 I2C 检测到 I2C Bus Free（总线空闲）后，通过 I2C RX 外设流控请求线，通知 DMA 搬移最后 1 个数据，DMA 搬移完成后自动关闭通道。 0：不使能 I2C RX 外设流控，外设流控请求线无效； 1：使能 I2C RX 外设流控，I2C 模块通过外设流控请求线通知 DMA 发起最后 1 次 RX 的 Single 传输。	0x0
[9:8]	RW	dma_operation	DMA 操作控制。TX FIFO 的 DMA Single 请求产生条件：TX FIFO 非满。TX FIFO 的 DMA Burst 请求产生条件：TX FIFO 有效数目小于等于配置的 TX FIFO 水线。 RX FIFO 的 DMA Single 请求产生条件：RX FIFO 非空。RX FIFO 的 DMA Burst 请求产生条件：RX FIFO 有效数目大于等于配置的 RX FIFO 水线。 00：非 DMA 方式； 01：禁能 TX FIFO 的 DMA Single 请求和 Burst 请求，使能 RX FIFO 的 DMA Single 请求和 Burst 请求； 10：使能 TX FIFO 的 DMA Single 请求和 Burst 请求，禁能 RX FIFO 的 DMA Single 请求和 Burst 请求； 11：同时使能 RX FIFO 和 TX FIFO 的 DMA Single 请求和 Burst 请求。	0x0
[7:4]	-	reserved	保留。	0x0
[3]	W1_PULSE	rst_tx_fifo	复位 TX FIFO。写 1 生效，清空 TX	0x0

Bits	Access	Name	Description	Reset
			FIFO。 0：清空完成； 1：清空进行中。	
[2]	W1_PULSE	rst_rx_fifo	复位 RX FIFO。写 1 生效，清空 RX FIFO。 0：清空完成； 1：清空进行中。	0x0
[1]	-	reserved	保留。	0x0
[0]	RWSC	mst_start	Master 启动控制。此位仅在控制器处于 Master-only 和 Master-slave 模式下生效。写 1 开始 Master 功能，等待 TX FIFO 非空时执行 Master 命令，当 P 命令执行完毕或者异常后会自动把此位清 0。 0：不启动 Master，当 P 命令执行完毕，或者总裁丢失，或者 ACK 不匹配，自动清 0； 1：启动 Master，等待 TX FIFO 非空时执行 Master 命令。	0x0

I2C_CTRL2

I2C_CTRL2 为 I2C 控制寄存器 2。

Offset Address: 0x00D4 Total Reset Value: 0x0007_3011

Bits	Access	Name	Description	Reset
[31:19]	-	reserved	保留。	0x0000
[18]	RO	smb_suspend_n_in	SMBus SMBSUS#输入值。 0：低电平输入； 1：高电平输入。	0x1
[17]	RW	smb_suspend_n_out	SMBus SMBSUS#输出值。	0x1

Bits	Access	Name	Description	Reset
			0：低电平输出； 1：高电平输出。	
[16]	RW	smb_suspend_n_oe_n	SMBus SMBSUS#是否输出。 0：Push-Pull 输出。 1：不输出，高阻；	0x1
[15:14]	-	reserved	保留。	0x0
[13]	RO	smb_alert_n_in	SMBus SMBALERT#输入值。 0：低电平输入； 1：高电平输入。	0x1
[12]	RW	smb_alert_n_oe_n	SMBus SMBALERT#输出值。 0：低电平 Open-Drain 输出； 1：Open-Drain 输出高阻。	0x1
[11:9]	-	reserved	保留。	0x0
[8]	RW	gpio_mode	I2C 管脚当 GPIO 使用。 0：I2C 管脚不受 force_scl 和 force_sda 控制； 1：I2C 管脚由 force_scl 和 force_sda 控制。	0x0
[7:5]	-	reserved	保留。	0x0
[4]	RW	force_scl	强制设置 I2C SCL 管脚的值。 0：设成 0，总线状态一定为 0； 1：设成 1，总线状态由线上的其他设备决定。	0x1
[3:1]	-	reserved	保留。	0x0
[0]	RW	force_sda	强制设置 I2C SDA 管脚的值。 0：设成 0，总线状态一定为 0； 1：设成 1，总线状态由线上的其他设备决定。	0x1

I2C_FIFO_STAT

I2C_FIFO_STAT 为 I2C FIFO 状态寄存器。

Offset Address: 0x00D8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:13]	-	reserved	保留。	0x00000
[12:8]	RO	tx_fifo_vld_num	TX FIFO 里的有效数目。	0x00
[7:5]	-	reserved	保留。	0x0
[4:0]	RO	rx_fifo_vld_num	RX FIFO 里的有效数目。	0x00

I2C_FSM_STAT

I2C_FSM_STAT 为 I2C 状态机状态寄存器。

Offset Address: 0x00DC Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	-	reserved	保留。	0x0
[30]	RO	i2c_bus_free	SCL 和 SDA 处于高电平并持续到达阈值时间，控制器认为总线空闲。软件可配置 I2C_BUS_FREE 调整阈值时间。 0：总线忙； 1：总线空闲。	0x0
[29:26]	-	reserved	保留。	0x0
[25]	RO	slv_busy	Slave 忙。Slave 检测到总线上有 START 条件且地址匹配后，Slave 进入忙状态。Slave 检测到总线上有 STOP 条件，或者 Slave 执行 S 命令，或者 Slave 执行 P 命令，Slave 退出忙状态。 0：Slave 空闲； 1：Slave 忙。	0x0

Bits	Access	Name	Description	Reset
[24:21]	RO	slv_cmd_exe	Slave 正在执行的时序命令。时序命令的具体含义，请参见 I2C_TX_FIFO[mst_slv_cmd]位域。	0x0
[20:15]	-	reserved	保留。	0x00
[14]	RO	mst_busy	Master 忙。Master 执行 S 命令后，Master 进入忙状态。Master 仲裁丢失或者 Master 执行完成 P 命令后，Master 退出忙状态。 0：Master 空闲； 1：Master 忙。	0x0
[13:10]	RO	mst_cmd_exe	Master 正在执行的时序命令。时序命令的具体含义，请参见 I2C_TX_FIFO[mst_slv_cmd]位域。	0x0
[9:0]	-	reserved	保留。	0x000

I2C_INTR_RAW

I2C_INTR_RAW 为 I2C 原始中断寄存器。

Offset Address: 0x00E0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RW1C	slv_addr_match_raw	当 Slave 检测到接收到的地址匹配时，产生该原始中断。原始中断&中断使能=中断状态，软件须对原始中断写 1 清除该中断。 0：未接受到地址，或者地址不匹配； 1：地址匹配。	0x0
[20]	RW1C	slv_rx_ack_unmatch_raw	Slave 接收应答位是否符合时序命令预期。写 1 清除该原始中断。 0：符合预期；	0x0

Bits	Access	Name	Description	Reset
			1：不符合预期。	
[19]	RW1C	slv_pec_check_fail_raw	Slave 检查接收到 PEC 错误。 0：PEC 正确； 1：PEC 错误。	0x0
[18]	RW1C	mst_pec_check_fail_raw	Master 检查接收到 PEC 错误。 0：PEC 正确； 1：PEC 错误。	0x0
[17]	RW1C	slv_cmd_int1_raw	Slave 时序命令中断 1，Slave 执行 INT1 命令时产生此原始中断。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[16]	RW1C	mst_cmd_int1_raw	Master 时序命令中断 1，Master 执行 INT1 命令时产生此原始中断。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[15]	RW1C	smb_suspend_raw	检测到 SMBus SMBSUS# 输入信号下降沿。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[14]	RW1C	smb_alert_raw	检测到 SMBus SMBALERT# 输入信号下降沿。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[13]	RW1C	scl_low_timeout_raw	检测到 SCL 低电平超时。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0

Bits	Access	Name	Description	Reset
[12]	RW1C	mst_cmd_done_raw	Master 时序命令正常完成，Master 执行完成 P 时序命令时产生此原始中断。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[11]	RW1C	arb_lost_raw	仲裁丢失。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[10]	RW1C	start_det_raw	检查到 START。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[9]	RW1C	stop_det_raw	检查到 STOP。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[8]	RW1C	tx_data_request_raw	TX FIFO 请求新的命令和数据。此时发生 Clock Stretching，暂停总线。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[7]	RW1C	rx_data_ready_raw	RX FIFO 接收到了新的数据。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[6]	RW1C	tx_fifo_not_full_raw	TX FIFO 非满。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[5]	RW1C	tx_fifo_empty_raw	TX FIFO 空。写 1 清除该原始中断。 0：无中断；	0x0

Bits	Access	Name	Description	Reset
			1：有中断。	
[4]	RW1C	tx_le_watermark_raw	TX FIFO 里的数据个数小于等于水线。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[3]	RW1C	rx_fifo_full_raw	RX FIFO 满。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[2]	RW1C	rx_ge_watermark_raw	RX FIFO 里的数据个数大于等于水线。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[1]	RW1C	rx_fifo_not_empty_raw	RX FIFO 非空。写 1 清除该原始中断。 0：无中断； 1：有中断。	0x0
[0]	RW1C	mst_rx_ack_unmatch_raw	Master 接收应答位是否符合时序命令预期。写 1 清除该原始中断。 0：符合预期； 1：不符合预期。	0x0

I2C_INTR_EN

I2C_INTR_EN 为 I2C 中断使能寄存器。

Offset Address: 0x00E4 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RW	slv_addr_match_en	slv_addr_match 中断使能。中断状态 = 原始中断&中断使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[20]	RW	slv_rx_ack_unmatch_en	slv_rx_ack_unmatch 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[19]	RW	slv_pec_check_fail_en	slv_pec_check_fail 中断使能。中断状态 =原始中断&中断使能。 0：不使能; 1：使能。	0x0
[18]	RW	mst_pec_check_fail_en	mst_pec_check_fail 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[17]	RW	slv_cmd_int1_en	Slave 时序命令中断 1 使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[16]	RW	mst_cmd_int1_en	Master 时序命令中断 1 使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[15]	RW	smb_suspend_en	smb_suspend 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[14]	RW	smb_alert_en	smb_alert 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0

Bits	Access	Name	Description	Reset
[13]	RW	scl_low_timeout_en	SCL 低电平超时中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[12]	RW	mst_cmd_done_en	mst_cmd_done 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[11]	RW	arb_lost_en	arb_lost 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[10]	RW	start_det_en	start_det 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[9]	RW	stop_det_en	stop 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[8]	RW	tx_data_request_en	tx_data_request 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[7]	RW	rx_data_ready_en	rx_data_ready 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[6]	RW	tx_fifo_not_full_en	tx_fifo_not_full 中断使能。中断状态 = 原	0x0

Bits	Access	Name	Description	Reset
			始中断&中断使能。 0：不使能; 1：使能。	
[5]	RW	tx_fifo_empty_en	tx_fifo_empty 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[4]	RW	tx_le_watermark_en	tx_le_watermark 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[3]	RW	rx_fifo_full_en	rx_fifo_full 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[2]	RW	rx_ge_watermark_en	rx_ge_watermark 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[1]	RW	rx_fifo_not_empty_en	rx_fifo_not_empty 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0
[0]	RW	mst_rx_ack_unmatch_en	mst_ack_unmatch 中断使能。中断状态 = 原始中断&中断使能。 0：不使能; 1：使能。	0x0

I2C_INTR_STAT

I2C_INTR_STAT 为 I2C 中断状态寄存器。

Offset Address: 0x00E8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RO	slv_addr_match_int	Slave 检测到接收到的地址匹配。 0：无中断; 1：有中断。	0x0
[20]	RO	slv_rx_ack_unmatch_int	Slave 接收应答位是否符合时序命令预期。 0：无中断; 1：有中断。	0x0
[19]	RO	slv_pec_check_fail_int	Slave 检查接收到 PEC 错误。 0：无中断; 1：有中断。	0x0
[18]	RO	mst_pec_check_fail_int	Master 检查接收到 PEC 错误。 0：无中断; 1：有中断。	0x0
[17]	RO	slv_cmd_int1	Slave 时序命令中断 1。 0：无中断; 1：有中断。	0x0
[16]	RO	mst_cmd_int1	Master 时序命令中断 1。 0：无中断; 1：有中断。	0x0
[15]	RO	smb_suspend_int	检测到 SMBus SMBSUS# 输入信号下降沿。 0：无中断; 1：有中断。	0x0
[14]	RO	smb_alert_int	检测到 SMBus SMBALERT# 输入信号下	0x0

Bits	Access	Name	Description	Reset
			降沿。 0：无中断； 1：有中断。	
[13]	RO	scl_low_timeout_int	SCL 低电平超时中断。 0：无中断； 1：有中断。	0x0
[12]	RO	mst_cmd_done_int	Master 时序命令正常完成。 0：无中断； 1：有中断。	0x0
[11]	RO	arb_lost_int	仲裁丢失。 0：无中断； 1：有中断。	0x0
[10]	RO	start_det_int	检查到 START。 0：无中断； 1：有中断。	0x0
[9]	RO	stop_det_int	检查到 STOP。 0：无中断； 1：有中断。	0x0
[8]	RO	tx_data_request_int	TX FIFO 请求新的数据。当 Master-Transmitter 或者 Slave-Transmitter 发送数据，但是 TX FIFO 空，此时发生 Clock Stretching，暂停总线，并产生该中断。 0：无中断； 1：有中断。	0x0
[7]	RO	rx_data_ready_int	RX FIFO 接收了新的数据。当 Master-Receiver 或者 Slave-Receiver 接收到了新的数据，产生该中断。 0：无中断；	0x0

Bits	Access	Name	Description	Reset
			1：有中断。	
[6]	RO	tx_fifo_not_full_int	TX FIFO 非满。 0：无中断； 1：有中断。	0x0
[5]	RO	tx_fifo_empty_int	TX FIFO 空。 0：无中断； 1：有中断。	0x0
[4]	RO	tx_le_watermark_int	TX FIFO 里的数据个数小于等于水线。 0：无中断； 1：有中断。	0x0
[3]	RO	rx_fifo_full_int	RX FIFO 满。 0：无中断； 1：有中断。	0x0
[2]	RO	rx_ge_watermark_int	RX FIFO 里的数据个数大于等于水线。 0：无中断； 1：有中断。	0x0
[1]	RO	rx_fifo_not_empty_int	RX FIFO 非空。 0：无中断； 1：有中断。	0x0
[0]	RO	mst_rx_ack_unmatch_int	应答位是否符合预期。 0：符合预期； 1：不符合预期。	0x0

I2C_VERSION

I2C_VERSION 为 I2C 版本号寄存器。

Offset Address: 0x0100 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:0]	RO	version	I2C 控制器版本号。	0x00000001

I2C_SCL_TIMEOUT

I2C_SCL_TIMEOUT 为 I2C SCL 超时阈值寄存器。

Offset Address: 0x0104 Total Reset Value: 0x007F_FFFF

Bits	Access	Name	Description	Reset
[31:23]	-	reserved	保留。	0x000
[22:0]	RW	scl_low_timeout	SCL 低电平超时配置。单位为 I2C 工作时钟周期。超时后，产生 SCL 低电平超时原始中断 scl_low_timeout_raw。 0x000000：0 个 I2C 工作时钟周期； 0x000001：1 个 I2C 工作时钟周期； ... 0x7FFFFF：8388607 个 I2C 工作时钟周期。	0x7FFFFF

I2C_BUS_FREE

I2C_BUS_FREE 为 I2C 总线空闲阈值寄存器。

Offset Address: 0x0108 Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	bus_free_time	总线空闲阈值。单位为 I2C 工作时钟周期。控制器检测到 STOP 条件后，SCL 和 SDA 同时为高电平并持续到达阈值时间，控制器认为总线空闲。 0x0000：0 个 I2C 工作时钟周期； 0x0001：1 个 I2C 工作时钟周期； ...	0xFFFF

Bits	Access	Name	Description	Reset
			0xFFFF：65535 个 I2C 工作时钟周期。 **注意：控制器 Master 功能只有在总线空闲时才能发送命令代号为 MST_S 的时序命令。**	

I2C_FILTER

I2C_FILTER 为 I2C SDA 和 SCL 滤波配置寄存器。

Offset Address: 0x010C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	spike_filter_time	SDA 和 SCL 毛刺过滤配置。 0x0：不过滤毛刺； 0x1：过滤 1 个 I2C 工作时钟周期的毛刺； 0x2：过滤 2 个 I2C 工作时钟周期的毛刺； ... 0xF：过滤 15 个 I2C 工作时钟周期的毛刺。	0x0

20 通用异步收发传输器(UART)

20.1 概述

通用异步收发器 UART（Universal Asynchronous Receiver Transmitter）是一个全双工异步串行的通信接口，主要功能是将来自外围设备的数据进行串并转换之后传入内部总线，以及将数据进行并串转换之后输出到外围设备，从而实现设备间的通信。

MCU 提供 4 个 UART 控制器，均支持 4 线模式，支持硬件流控。

须知

当使用 UART 的接收功能时，需要在 UART 初始化前对 RXD 管脚配置为上拉。

20.2 特性

UART 模块有以下特点：

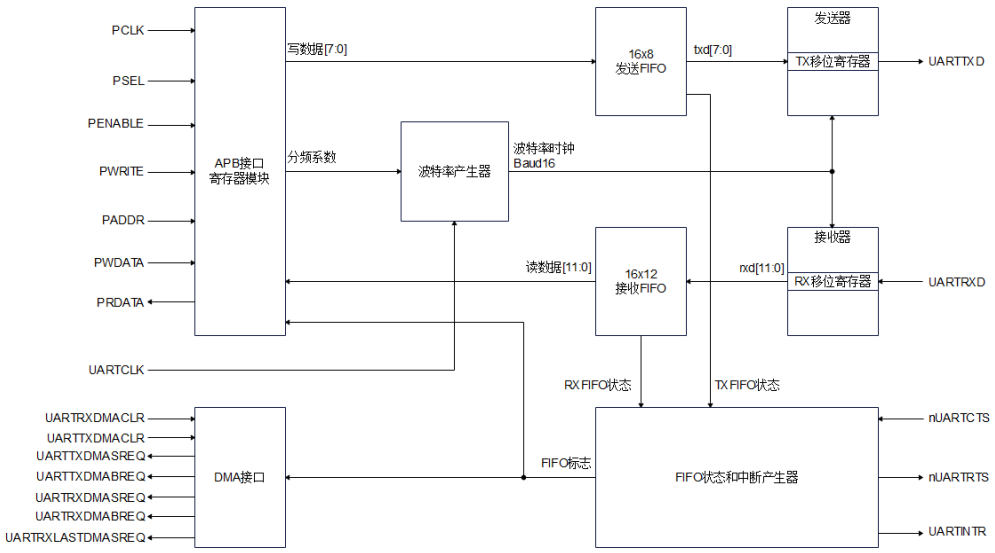
- 支持 16 x 8bit 的发送 FIFO 和 16 x 12bit 的接收 FIFO。
- 支持数据位和停止位的位宽可编程。数据位可通过编程设定为 5/6/7/8bit ； 停止位可通过编程设定为 1bit 或 2bit。
- 支持奇校验、偶校验、0/1 校验或无校验。
- 支持传输速率（波特率）可编程。
- 支持字符匹配中断、自动波特率检测错误中断、自动波特率检测完成中断、接收 FIFO 满中断、接收 FIFO 非空中断、接收 FIFO 空中断、发送 FIFO 非空中断、发送 FIFO 空中断、溢出错误中断、break 错误中断、校验中断、帧错误中断、接收 FIFO 水线中断、发送 FIFO 水线中断、接收超时中断、CTS 调制状态中断。

- 支持原始中断状态查询和屏蔽后中断状态查询。
- 支持发送完成状态查询和清除。
- 支持通过编程禁止 UART 模块或者单独禁能 UART 发送/接收功能。
- 支持关断 UART 时钟（通过 CRG 模块配置）。
- 支持 DMA 操作（与 DMA 模块配合使用）。
- 支持发送数据和接收数据位序可配置（MSB/LSB first）。
- 支持 12 ~ 16 倍过采样可配置。
- 支持接收超时时长可配置，配置范围为 1 ~ 65536 个 bit 时长。
- 支持波特率自动检测。
- 支持字符检测，字符可配置。

20.3 功能描述

UART 模块框图如图 20-1 所示。通过总线控制 UART 模块进行数据的发送和接收操作，并生成相应的 DMA 请求和中断信号。

图20-1 模块框图



20.3.1 FIFO

RX_FIFO 为 16 × 12bit，TX_FIFO 为 16 × 8bit。接收数据比发送数据多 4bit 的接收错误标志位。发送/接收的数据以及具体的接收错误标志位见 UART_DR 或 UART_RSR。

可以通过 UART_LCR_H 使能 FIFO。当不使能 FIFO 时，FIFO 相当于一个深度为 1 的保持寄存器，可以存放 1 帧数据。接收保持寄存器为 1 × 12bit，发送保持寄存器为 1 × 8bit。此时仍然可以通过 UART_FR 查询保持寄存器的空满状态。

FIFO 的空满、忙闲状态可以通过 UART_FR 查询。

可以通过 UART_IFLS 设置 FIFO 水线。接收 FIFO 水线中断在接收 FIFO 中数据量大于等于接收 FIFO 水线时触发，发送 FIFO 水线中断在发送 FIFO 中数据量小于等于发送 FIFO 水线时触发。

20.3.2 波特率生成

通过配置寄存器 UART_IBRD 和 UART_FBRD 可以设置 UART 工作的波特率，波特率计算公式为：

波特率=uart_clk /（过采样倍数 x 分频系数）

uart_clk 频率由 CRG 配置。过采样倍数由 UART_SPCFG[spcfg]配置，配置范围为 12~16。分频系数由整数和小数两部分组成，分别对应寄存器 UART_IBRD 和 UART_FBRD。

分频系数值的计算以及分频系数寄存器的配置举例如下：

如果要求波特率为 230,400bit/s，并且 UART 参考时钟频率为 100MHz，过采样倍数为 16，那么分频系数为$(100 \times 10^6)/(16 \times 230400) = 27.1267$，因此 IBRD（整数部分）为 27，FBRD（小数部分）为 0.1267。

计算 6bit UART_FBRD 寄存器中的数值：根据 $m=integer(FBRD \times 2^n+0.5)$ (n=UART_FBRD 寄存器的宽度)，计算出 $m = integer(0.1267 \times 2^6 + 0.5) = 9$，因此在 UART_IBRD 寄存器中配置 0x1B，UART_FBRD 寄存器中配置 0x9。

当分频系数小数部分配置成 9 时，波特率除数的实际数值为 $27+9/2^6 = 27.1406$，产生的波特率为$(100 \times 10^6)/(16 \times 27.1406) = 230282.3077$，误差率为(实际波特率 - 理想波特率)/理想波特率 =(230282.3077−230400)/230400 = -0.051082%。

使用 6bit UART_FBRD 寄存器最大的误差率为 $1/2^6$ = 1.56%，此时连续传输超过 64 个比特就会产生数据错位。实际 UART 的一帧传输小于 64 个比特，因此不会产生数据错位。

20.3.3 中断功能

UART 模块支持字符匹配中断、自动波特率检测错误中断、自动波特率检测完成中断、接收 FIFO 满中断、接收 FIFO 非空中断、接收 FIFO 空中断、发送 FIFO 非空中断、发送 FIFO 空中断、溢出错误中断、break 错误中断、校验中断、帧错误中断、接收 FIFO 水线中断、发送 FIFO 水线中断、接收超时中断、CTS 调制状态中断。可以通过 UART_IMSC 设置相应中断的屏蔽状态。屏蔽之前的原始中断状态可以通过 UART_RIS 查询。屏蔽之后的中断状态通过 UART_MIS 查询。通过 UART_ICR 清除相应的中断状态。

- 通过 UART_IMSC[cmim]解除屏蔽字符匹配中断，字符匹配中断在接收到软件配置的字符后触发。软件通过向 UART_ICR[cmic]写 1 清除该中断。
- 通过 UART_IMSC[abdeim]解除屏蔽自动波特率检测错误中断，自动波特率检测错误中断置位条件是自动波特率检测期间，如果检测到的波特率过慢（对应的分频系数大于 65535）或过快（对应的分频系数小于 1）。软件可以通过向 UART_ICR[abdeic]写 1 清除该中断。
- 通过 UART_IMSC[abdcim]解除屏蔽自动波特率检测完成中断，自动波特率检测完成中断置位条件是自动波特率检测到的波特率在合法范围内（1 到 65535 之间），且对应的分频系数成功配置到寄存器中。软件通过向 UART_ICR[abdcic]写 1 清除该中断。
- 通过 UART_IMSC[rxffim]解除屏蔽接收 FIFO 满中断，接收 FIFO 满中断在接收 FIFO 满时会一直触发。软件通过读出接收 FIFO 的数据直到接收 FIFO 非满，或者向 UART_ICR[rxffic]写 1，清除该中断。
- 通过 UART_IMSC[rxfneim]解除屏蔽接收 FIFO 非空中断，接收 FIFO 非空中断在接收 FIFO 非空时会一直触发。软件通过读空接收 FIFO 的数据，或者向 UART_ICR[rxfneic]写 1，清除该中断。
- 通过 UART_IMSC[rxfeim]解除屏蔽接收 FIFO 空中断，接收 FIFO 空中断在接收 FIFO 为空时会一直触发。UART 接收数据直到接收 FIFO 非空，或者向 UART_ICR[rxfeic]写 1，清除该中断。
- 通过 UART_IMSC[txfneim]解除屏蔽发送 FIFO 非空中断，发送 FIFO 非空中断在发送 FIFO 非空时会一直触发。UART 发送数据直到发送 FIFO 空，或者向 UART_ICR[txfneic]写 1，清除该中断。

- 通过 UART_IMSC[txfeim]解除屏蔽发送 FIFO 空中断，发送 FIFO 空中断在发送 FIFO 为空时会一直触发。软件通过写入发送 FIFO 直到发送 FIFO 非空，或者向 UART_ICR[txfeic]写 1，清除该中断。
- 通过 UART_IMSC[oeim]解除屏蔽溢出错误中断，溢出错误中断在接收 FIFO 满，并再次接收到新的数据时触发。软件向 UART_ICR[oeic]写 1，清除该中断。
- 通过 UART_IMSC[beim]解除屏蔽 break 错误中断，break 错误中断在接收到 break 帧（接收数据持续低电平的时间比一个全字传输（包括起始位、数据位、校验位、停止位）还要长）时触发。软件向 UART_ICR[beic]写 1，清除该中断。
- 通过 UART_IMSC[peim]解除屏蔽校验中断，校验中断在检测到接收数据校验位错误时触发。软件向 UART_ICR[peic]写 1，清除该中断。
- 通过 UART_IMSC[feim]解除屏蔽帧错误中断，帧错误中断在检测到接收数据停止位错误时触发。软件向 UART_ICR[feic]写 1，清除该中断。
- 通过 UART_IMSC[rtim]解除屏蔽接收超时中断，接收超时中断在接收 FIFO 非空，且超过配置的 bit 周期没有接收到新数据时触发。软件通过读空接收 FIFO，或者向 UART_ICR[rtic]写 1，清除该中断。
- 通过 UART_IMSC[txim]解除屏蔽发送 FIFO 水线中断，发送 FIFO 水线中断在发送 FIFO 内数据个数小于或等于配置的发送 FIFO 水线时触发。软件通过写入发送 FIFO 直到发送 FIFO 中数据个数大于发送 FIFO 水线，并且向 UART_ICR[txic]写 1，清除该中断。
- 通过 UART_IMSC[rxim]解除屏蔽接收 FIFO 水线中断，接收 FIFO 水线中断在接收 FIFO 内数据大于或等于配置的接收 FIFO 水线时触发。软件通过读出接收 FIFO 的数据直到接收 FIFO 中数据个数小于接收 FIFO 水线，并且向 UART_ICR[rxic]写 1，清除该中断。
- 通过 UART_IMSC[ctsmim]解除屏蔽 CTS 调制状态中断，CTS 调制状态中断在输入信号 CTS 发生变化时触发。软件通过向 UART_ICR[ctsmic]写 1，清除该中断。

须知

除接收超时中断外，其余中断的原始中断状态在满足中断条件时就会置 1，无论对应的中断屏蔽位是否为 1。而接收超时中断的原始中断状态（UART_RIS[rtris]）只有在屏蔽位（UART_IMSC[rtim]）为 1，且满足接收超时中断条件时才会置 1。

20.3.4 发送完成状态

发送完成状态标志位（UART_RIS[txtcris]）会在以下条件置 1：

- UART 使能，发送 FIFO 为空，且发送数据线 TXD 上已经没有数据在传输。
- 软件可以通过向 UART_ICR[txtcic]写 1 清除发送完成状态标志位。

20.3.5 数据位序

当 UART_DS[msbfirst]配置为 0 时，UART 按照 LSB first 接收和发送数据。

当 UART_DS[msbfirst]配置为 1 时，UART 按照 MSB first 接收和发送数据。

20.3.6 过采样倍数

可通过 UART_SPCFG[spcfg]配置过采样倍数，配置范围为 12 ~ 16。假设过采样倍数为 n，则会对 RXD 的每个 bit 采样 n 次，取中间 3 次的采样值，按照多数原则确定最终采样值。

20.3.7 接收超时

可通过 UART_RTCFG[rtcfg]配置需要检测的接收超时时长，配置范围为 1 ~ 65536 个 bit 时长，当配置为 0 时对应超时时长为 1 个 bit，当配置为 65535 时对应超时时长为 65536 个 bit。

如果使能接收超时中断，则当接收 FIFO 非空，且超过配置的时间没有再接收到新的数据时会触发接收超时中断。

20.3.8 自动波特率检测

UART 通过接收一个 8bit 数据位宽的大写字符 Z 来自动检测波特率，检测成功后会自动设置 UART_IBRD 和 UART_FBRD 寄存器的值。

自动波特率检测的使能与四个使能位相关：UART_ABDEN[abden]、UART_CR[rxe]、UART_CR[txe]和 UART_CR[uarten]。只有当 UART_ABDEN[abden]和 UART_CR[uarten]使能，且 UART_CR[rxe]和 UART_CR[txe]禁能时，才能进行自动波特率检测。使能 UART_CR[rxe]和 UART_CR[txe]会自动禁能[abden]。UART_ABDEN[abdenvld]等于 0 表示不可以进行自动波特率检测，等于 1 表示可以进行自动波特率检测，软件可以通过读取该标志决定是否开始进行自动波特率检测。

UART_ABDEN[abdbusy]等于 0 表示未在进行自动波特率检测，等于 1 表示正在进行自动波特率检测，软件可以通过读取该标志决定是否禁能自动波特率检测。

如果使能了自动波特率检测完成中断，则在正确计算出分频系数（计算得到的分频系数在 1 和 65535 之间），并配置到 UART_IBRD 和 UART_FBRD 寄存器中后，产生自动波特率检测完成中断。

如果使能了自动波特率检测错误中断，则在计算得到的分频系数小于 1 或大于 65535 时，产生自动波特率检测错误中断。

20.3.9 字符检测

UART 支持字符检测，软件通过 UART_CHARMATCH[charmaten]使能字符检测功能，通过 UART_CHARMATCH[charmat]配置希望检测的字符，如果使能字符匹配中断，则会在接收到配置字符后产生字符匹配中断。

20.3.10 DMA 功能

UART 支持与 DMA 模块配合实现 DMA 数据搬运功能。可以通过 UART_DMACR[txdmae]和 UART_DMACR[rxdmae]进行控制，配置发送 FIFO 和接收 FIFO 的 DMA 使能。

如果使能发送 FIFO 的 DMA 功能，则在发送 FIFO 中数据量小于等于发送 FIFO 水线 UART_IFLS[txiflsel]，产生发送通道的 DMA BREQ 请求。在发送 FIFO 非满时，产生发送通道的 DMA SREQ 请求。

如果使能接收 FIFO 的 DMA 功能，则在接收 FIFO 中数据量大于等于接收 FIFO 水线 UART_IFLS[rxiflsel]，产生接收通道的 DMA BREQ 请求。在接收 FIFO 非空时，产生接收通道的 DMA SREQ 请求。

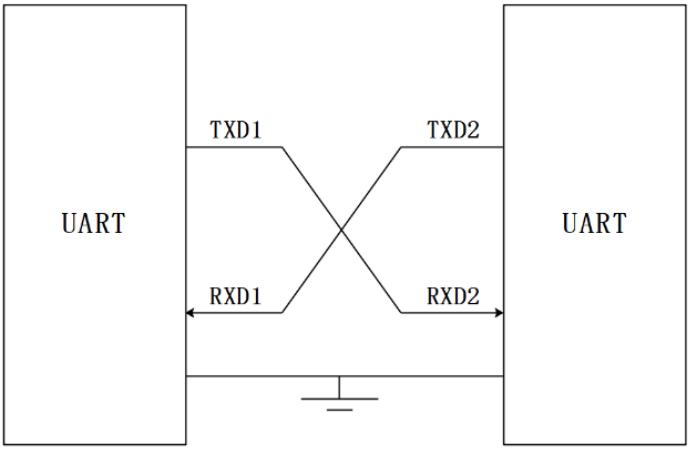
具体步骤请查看 20.4.2 DMA 方式下的数据传输章节。

当发生溢出错误中断、break 错误中断、校验中断或帧错误中断时，可以通过 UART_DMACR[dmaonerr]设置接收通道的 DMA 请求无效，避免对错误数据进行 DMA 搬移。另外，UART 接收通道还支持 DMA LASTREQ。当使能 DMA LASTREQ 时，如果接收 FIFO 中有 1 个数据，且满足接收超时的条件，则 UART 会产生 DMA LASTREQ，告诉 DMA 控制器数据搬移结束。

20.3.11 应用框图

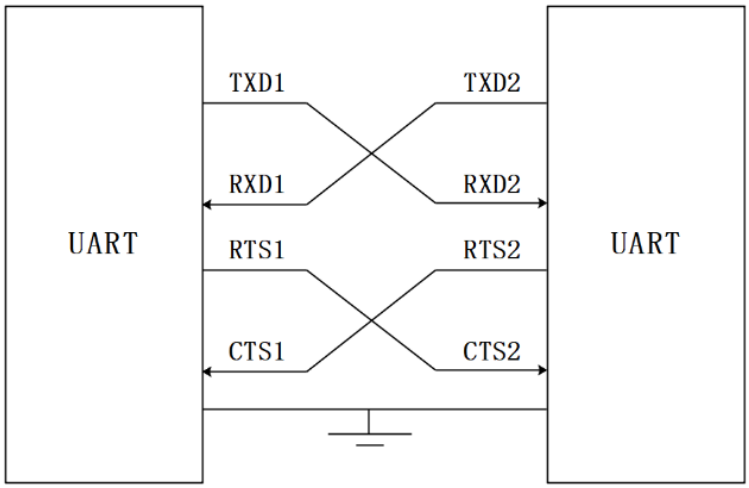
UART 的 2 线典型应用框图如图 20-2 所示。

图20-2 UART2 线典型应用框图



UART 的 4 线典型应用框图如图 20-3 所示，支持硬件流控。

图20-3 UART4 线典型应用框图



20.3.12 协议介绍

UART 的一次帧传输主要包括起始信号、数据位、校验位和结束信号。数据帧从某一 UART 的 TXD 端输出，从另一个 UART 的 RXD 端输入。各信号类型和位宽可以通过 UART_LCR_H 进行配置。

起始信号、数据位、校验位和结束信号的含义如下：

- 起始信号（start bit）

一个数据帧开始的标志，UART 协议规定 TXD/RXD 信号出现一个低电平就表示一个数据帧的开始。在 UART 不传输数据时，应该保持高电平。

- 数据位（data bit）

  数据位宽可以通过 UART_LCR_H 配置成 5bit/6bit/7bit/8bit 数据位宽。

- 校验位（parity bit）

  校验位是 1 比特纠错信号，UART 的校验位有奇校验、偶校验和固定校验位，同时支持校验位的使能和禁止，详细描述请见 UART_LCR_H 寄存器。

- 结束信号（stop bit）

  结束信号即数据帧的停止位，支持通过 UART_LCR_H 配置为 1bit 或 2bit。数据帧的结束信号就是 TXD/RXD 变成高电平。

20.4 典型配置

20.4.1 中断或查询方式下的数据传输

初始化

初始化步骤如下：

步骤 1 向 UART_CR[uarten]写 0，使 UART 处于禁止状态。

步骤 2 配置 UART_SPCFG 设定相应的过采样倍数。

步骤 3 写相应的配置值到 UART_IBRD、UART_FBRD 寄存器，配置波特率。

步骤 4 配置 UART_CR、UART_LCR_H，设定 UART 工作模式。

步骤 5 配置 UART_IFLS 设定发送及接收 FIFO 水线。

步骤 6 如果驱动程序采用中断方式，则需设定 UART_IMSC，使能相应中断信号；如果采用查询方式，则需禁止产生相应中断信号。

步骤 7 配置 UART_DS 设定相应的传输数据位序。

步骤 8 按需要配置 UART_RTCFG 和 UART_CHARMATCH。

步骤 9 向 UART_CR[uarten]写 1，使能 UART，完成初始化配置。

----结束

数据发送

数据发送步骤如下：

步骤 1 将发送数据写入 UART_DR，启动数据发送。

步骤 2 查询方式下，如果进行连续数据发送，需要通过读取 UART_FR[txff]检测 TX_FIFO 状态。如果 UART_FR[txff]为 0，即 TX_FIFO 未满，则可以向 TX_FIFO 中发送数据；中断方式下，则检测 UART_MIS[txmis]发送中断状态位，决定是否向 TX_FIFO 中发送数据。当 UART_MIS[txmis]置 1，此时 TX_FIFO 内数据量小于等于发送数据水线，可以向 TX_FIFO 中发送数据。

步骤 3 通过检测 UART_FR[txfe]是否为 1，判断 UART 是否完成全部数据发送。

----结束

数据接收

数据接收的处理方式如下：

- 查询方式下，进行数据接收时通过读取 UART_FR[rxfe]检测 RX_FIFO 状态，如果 UART_FR[rxfe]为，则 RX_FIFO 非空，可以读取 RX_FIFO 中的数据。
- 中断方式下，则检测 UART_MIS[rxmis]接收中断状态位，决定是否读取 RX_FIFO 中的数据。当 UART_MIS[rxmis]置 1，此时 RX_FIFO 内数据量大于等于接收 FIFO 水线，可以读取 RX_FIFO 中数据。

20.4.2 DMA 方式下的数据传输

初始化

初始化步骤如下：

步骤 1 向 UART_CR[uarten]写 0，使 UART 处于禁止状态。

步骤 2 配置 UART_SPCFG 设定相应的过采样倍数。

步骤 3 写相应的配置值到 UART_IBRD、UART_FBRD 寄存器，配置波特率。

步骤 4 配置 UART_CR、UART_LCR_H，设定相应的 UART 工作模式。

步骤 5 配置 UART_IFLS 设定相应的发送及接收 FIFO 水线。

步骤 6 如果驱动程序采用中断方式，则需设定 UART_IMSC，使能相应中断信号；如果采用查询方式，则应禁止产生相应中断信号。

步骤 7 配置 UART_DS 设定相应的传输数据位序。

步骤 8 按需要配置 UART_RTCFG 和 UART_CHARMATCH。

步骤 9 向 UART_CR[uarten]写 1，使能 UART，完成初始化配置。

----结束

数据发送

数据发送步骤如下：

步骤 1 配置 DMA 数据通道，包括数据传输源地址、目的地址、数据传输个数、传输类型等参数。具体配置请参见 10 直接存储器访问控制器(DMA)章节的相关描述。

步骤 2 配置 UART_DMACR[txdmae]为 1，使能 UART 发送通道的 DMA 功能。

步骤 3 通过 DMA 中断上报或状态查询，判断数据是否发送完成，如果完成则关闭 UART 的 DMA 发送功能，具体请参见 10 直接存储器访问控制器(DMA)章节。

----结束

数据接收

数据接收步骤如下：

步骤 1 配置 DMA 数据通道，包括数据传输源地址、目的地址、数据传输个数、传输类型等参数。具体配置请参见 10 直接存储器访问控制器(DMA)章节的相关描述。

步骤 2 配置 UART_DMACR[rxdmae]为 1，使能 UART 接收通道的 DMA 功能。

步骤 3 通过 DMA 中断上报或状态查询，判断数据是否接收完成，如果完成则关闭 UART 的 DMA 接收功能，具体请参见 10 直接存储器访问控制器(DMA)章节。

----结束

20.4.3 自动波特率检测

步骤 1 向 UART_CR[uarten]写 0，使 UART 处于禁止状态。

步骤 2 配置 UART_SPCFG 设定相应的过采样倍数。

步骤 3 配置 UART_CR 和 UART_LCR_H，设定相应的 UART 工作模式，需要注意的是需要将 UART_CR[rxe]和 UART_CR[txe]禁能。

步骤 4 配置 UART_IFLS 设定相应的发送及接收 FIFO 水线。

步骤 5 如果驱动程序采用中断方式，则还需使能相应中断信号；如果采用查询方式，则需禁止产生相应中断信号。

步骤 6 配置 UART_DS 设定相应的传输数据位序。

步骤 7 按需要配置 UART_RTCFG 和 UART_CHARMATCH。

步骤 8 向 UART_CR[uarten]写 1，使能 UART，完成初始化配置。

步骤 9 配置 UART_IMSC[abdcim]和 UART_IMSC[abdeim]为 1，使能自动波特率检测完成中断和自动波特率检测错误中断。

步骤 10 配置 UART_ABDEN[abden] = 1，使能自动波特率检测。

步骤 11 读 UART_ABDEN[abdenvld]，确认等于 1。

步骤 12 按照 8bit 数据位宽向 UART 发送大写字符 Z，等待自动波特率检测完成中断或自动波特率检测错误中断。如果自动波特率检测完成中断拉起，则跳到步骤 14。如果自动波特率检测错误中断拉起，则跳到步骤 13。

步骤 13 等待 UART_ABDEN[abdbusy]拉低，清除自动波特率检测错误中断，如果需要重新检测则跳到步骤 12。

步骤 14 等待 UART_ABDEN[abdbusy]拉低，清除自动波特率检测完成中断，如果需要重新检测则跳到步骤 12。如果不需要继续检测，则使能 UART_CR[rxe]和 UART_CR[txe]，准备按照自动检测得到的波特率进行数据传输。

----结束

数据传输期间进行检测

步骤 1 向 UART_CR[uarten]写 0，使 UART 处于禁止状态。

步骤 2 配置 UART_SPCFG 设定相应的过采样倍数。

步骤 3 写相应的配置值到 UART_IBRD、UART_FBRD 寄存器，配置传输速率。

步骤 4 配置 UART_CR、UART_LCR_H，设定相应的 UART 工作模式。

步骤 5 配置 UART_IFLS 设定相应的发送及接收 FIFO 水线。

步骤 6 如果驱动程序采用中断方式，则需设定 UART_IMSC，使能相应中断信号；如果采用查询方式，则需禁止产生相应中断信号。

步骤 7 配置 UART_DS 设定相应的传输数据 bit 顺序。

步骤 8 按需要配置 UART_RTCFG 和 UART_CHARMATCH。

步骤 9 向 UART_CR[uarten]写 1，使能 UART，完成初始化配置。

步骤 10 在 UART 进行数据传输期间，禁能 UART_CR[rxe]和 UART_CR[txe]，配置 UART_ABDEN[abden] = 1，但会等到当前字符传输结束后才会真正生效。

步骤 11 读 UART_ABDEN[abdenvld]，确认等于 1。

步骤 12 按照 8bit 数据位宽向 UART 发送大写字符 Z，等待自动波特率检测完成中断或自动波特率检测错误中断。如果自动波特率检测完成中断拉起，则跳到 步骤 14。如果自动波特率检测错误中断拉起，则跳到 步骤 13。

步骤 13 等待 UART_ABDEN[abdbusy]拉低，清除自动波特率检测错误中断，如果需要重新检测则跳到 步骤 12。

步骤 14 等待 UART_ABDEN[abdbusy]拉低，清除自动波特率检测完成中断，如果需要重新检测则跳到 步骤 12。如果不需要继续检测，则使能 UART_CR[rxe]和 UART_CR[txe]，准备按照自动检测得到的波特率进行数据传输。

----结束

20.5 UART 寄存器概览

地址空间内：

- UART0 寄存器基地址为：0x1400_0000。
- UART1 寄存器基地址为：0x1400_1000。
- UART2 寄存器基地址为：0x1400_2000。
- UART3 寄存器基地址为：0x1400_3000。

UART 寄存器概览如表 20-1 所示。

表20-1 UART 寄存器概览

偏移地址	名称	描述	页码
0x000	UART_DR	数据寄存器	UART_DR
0x004	UART_RSR	接收状态寄存器/错误清除寄存器	UART_RSR

偏移地址	名称	描述	页码
0x018	UART_FR	标志寄存器	UART_FR
0x024	UART_IBRD	整数波特率寄存器	UART_IBRD
0x028	UART_FBRD	小数波特率寄存器	UART_FBRD
0x02C	UART_LCR_H	线控寄存器	UART_LCR_H
0x030	UART_CR	控制寄存器	UART_CR
0x034	UART_IFLS	中断 FIFO 水线选择寄存器	UART_IFLS
0x038	UART_IMSC	中断屏蔽寄存器	UART_IMSC
0x03C	UART_RIS	原始中断状态寄存器	UART_RIS
0x040	UART_MIS	屏蔽后中断状态寄存器	UART_MIS
0x044	UART_ICR	中断清除寄存器	UART_ICR
0x048	UART_DMACR	DMA 控制寄存器	UART_DMACR
0x050	UART_DS	数据传输顺序配置寄存器	UART_DS
0x054	UART_RTCFG	接收超时时长配置寄存器	UART_RTCFG
0x058	UART_SPCFG	过采样配置寄存器	UART_SPCFG
0x05C	UART_ABDEN	自动波特率检测使能寄存器	UART_ABDEN
0x060	UART_CHARMATCH	字符匹配配置寄存器	UART_CHARMATCH

20.6 UART 寄存器描述

UART_DR

UART_DR 为 UART 数据寄存器，存放接收数据和发送数据，同时可以从该寄存器中读出接收状态。对于要发送的数据，占据 8bit 寄存器空间；对于接收到的数据，占据 12bit 寄存器空间。

Offset Address: 0x000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	-	reserved	保留。	0x00000
[11]	RO	oe	溢出错误。 0：无溢出错误； 1：有溢出错误，即接收 FIFO 满后接收到了新的数据。	0x0
[10]	RO	be	Break 错误。 0：无 break 错误； 1：有 break 错误，即接收数据持续低电平的时间比一个全字传输（包括起始位、数据位、校验位、停止位）还要长。	0x0
[9]	RO	pe	校验错误。 0：无校验错误； 1：有校验错误。	0x0
[8]	RO	fe	帧错误。 0：无帧错误； 1：有帧错误（错误的停止位）。	0x0
[7:0]	RW	data	接收数据和发送数据。	0x00

UART_RSR

UART_RSR 为接收状态寄存器/错误清除寄存器。

- 寄存器读时作为接收状态寄存器。

- 寄存器写时作为错误清除寄存器。

接收状态也可以从 UART_DR 中读出。从 UART_DR 中读出的 break、frame、parity 的状态信息要比从 UART_RSR 读出的信息优先级高（即 UART_RSR[be]、UART_RSR[pe]、UART_RSR[fe]只有对 UART_DR 进行读操作才会刷新，UART_RSR[oe]不会刷新）。

对 UART_RSR 寄存器的任何写操作都会对 UART_RSR 寄存器进行复位，清除错误。

Offset Address: 0x004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW	oe	溢出错误。 0：无溢出错误； 1：溢出错误。 当接收 FIFO 满时，接收 FIFO 中的内容保持有效，因为不会有下一个数据写到接收 FIFO 中，只是移位寄存器会溢出。CPU 必须立刻读数据以腾空 FIFO。	0x0
[2]	RW	be	break 错误。 0：无 break 错误； 1：break 错误。 break 错误的条件：接收数据持续低电平的时间比一个全字传输（包括起始位、数据位、校验位、停止位）还要长。	0x0
[1]	RW	pe	校验错误。 0：无校验错误； 1：接收数据的校验错误。 FIFO 模式下，该错误与 FIFO 顶部的数据相关联。	0x0
[0]	RW	fe	帧错误。 0：无帧错误； 1：接收到的数据的停止位错误（有效的停	0x0

Bits	Access	Name	Description	Reset
			止位为 1）。	

UART_FR

UART_FR 为 UART 标志寄存器。

Offset Address: 0x018 Total Reset Value: 0x0000_0197

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000001
[7]	RO	txfe	发送 FIFO 空标志，该位的含义由 UART_LCR_H[fen]的状态决定。 如果 UART_LCR_H[fen]为 0，则当发送保持寄存器空时该位置 1； 如果 UART_LCR_H[fen]为 1，则当发送 FIFO 为空时该位置 1。	0x1
[6]	RO	rxff	接收 FIFO 满标志，该位的含义由 UART_LCR_H[fen]的状态决定。 如果 UART_LCR_H[fen]为 0，则当接收保持寄存器满时该位置 1； 如果 UART_LCR_H[fen]为 1，则当接收 FIFO 为满时该位置 1。	0x0
[5]	RO	txff	发送 FIFO 满标志，该位的含义由 UART_LCR_H[fen]的状态决定。 如果 UART_LCR_H[fen]为 0，则当发送保持寄存器满时该位置 1； 如果 UART_LCR_H[fen]为 1，当发送 FIFO 为满时该位置 1。	0x0
[4]	RO	rxfe	接收 FIFO 空标志，该位的含义由 UART_LCR_H[fen]的状态决定。 如果 UART_LCR_H[fen]为 0，则当接收保持寄存器空时该 bit 置 1；	0x1

Bits	Access	Name	Description	Reset
			如果 UART_LCR_H[fen]为 1，则当接收 FIFO 为空时该位就置 1。	
[3]	RO	busy	UART 忙闲状态位。 0：UART 空闲或者完成发送数据； 1：UART 正忙于发送数据。 该位一旦置位，该状态一直保持到整个字节（包括所有的停止位）完全从移位寄存器中发送出去。 一旦发送 FIFO 非空该位就置位，不管 UART 使能与否。	0x0
[2:1]	-	reserved	保留。	0x3
[0]	RO	cts	CTS 状态位。只对支持硬件流控的 UART 有效，为输入 CTS 管脚的状态取反。	0x1

UART_IBRD

UART_IBRD 为整数波特率寄存器。

Offset Address: 0x024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	baud divint	整数波特率分频值。复位时全部清 0。	0x0000

UART_FBRD

UART_FBRD 为小数波特率寄存器。

Offset Address: 0x028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5:0]	RW	baud divfrac	小数波特率分频值。复位时全部清 0。	0x00

须知

整数波特率寄存器和小数波特率寄存器的值必须等到当前数据发送和接收完毕才能更新。

最小的分频值为 1，最大的分频值为 65535(2^{16} - 1)。即 UART_IBRD = 0 是无效的，此时 UART_FBRD 也将被忽略。同样，如果 UART_IBRD = 65535（0xFFFF），UART_FBRD 就只能是 0，如果比 0 大，则会导致发送和接收的失败。

UART_LCR_H

UART_LCR_H 为线控寄存器，UART_LCR_H、UART_IBRD、UART_FBRD 组成一个 30bit 宽的寄存器 UART_LCR。UART_LCR 只有在对 UART_LCR_H 写入后才会刷新。因此，如果更新 UART_IBRD 和 UART_FBRD 的内容，必须同时更新 UART_LCR_H。

Offset Address: 0x02C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	-	reserved	保留。	0x000000
[7]	RW	sps	stick parity 选择。 0：固定校验位禁止。具体校验位由奇偶校验算法决定； 1：固定校验位使能。当配置为偶校验时，校验位固定为 0，当配置为奇校验时，校验位固定为 1。	0x0
[6:5]	RW	wlen	指示发送和接收一个帧内数据比特的数目。 00：5bit； 01：6bit； 10：7bit； 11：8bit。	0x0
[4]	RW	fen	发送和接收 FIFO 使能控制。 0：不使能发送和接收 FIFO；	0x0

Bits	Access	Name	Description	Reset
			1：使能发送和接收 FIFO。	
[3]	RW	stp2	发送帧尾停止位比特选择。 0：发送的帧尾接 1bit 停止位； 1：发送的帧尾接 2bit 停止位。 接收逻辑在接收时不检查 2bit 的停止位。	0x0
[2]	RW	eps	发送和接收过程中的奇偶校验选择。 0：在发送和接收过程中生成奇校验或检查奇校验； 1：在发送和接收过程中生成偶校验或检查偶校验。 当 pen 为 0 时，该位不起作用。	0x0
[1]	RW	pen	校验选择位。 0：不作校验； 1：发送方向产生校验，接收方向作校验检查。	0x0
[0]	RW	brk	发送 break。 0：无效； 1：在完成当前数据的发送后，TXD 连续输出低电平。 **注意：** **要正确的执行 break 命令，软件将该位置 1 的时间必须超过 2 个完整帧；在正常使用中，该位必须清 0。**	0x0

须知

当 UART_LCR_H[brk]置 1，或软件在数据传输过程中禁用 UART 后再重新使能，此时 FIFO 的数据完整性不能保证。

UART_CR

UART_CR 为 UART 控制寄存器。

配置 UART_CR 遵循以下步骤:

步骤 1 向 UART_CR[uarten]写 0，禁止 UART。

步骤 2 等待当前数据发送或接收结束。

步骤 3 将 UART_LCR_H[fen]清 0 以刷新 FIFO。

步骤 4 重新配置 UART_CR。

步骤 5 向 UART_CR[uarten]写 1，使能 UART。

----结束

Offset Address: 0x030 Total Reset Value: 0x0000_0300

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15]	RW	ctsen	CTS 硬件流控使能。 0：不使能 CTS 硬件流控; 1：使能 CTS 硬件流控，只有当 nUARTCTS 信号有效时才发送数据。	0x0
[14]	RW	rtsen	RTS 硬件流控使能。 0：不使能 RTS 硬件流控; 1：使能 RTS 硬件流控，只有当接收 FIFO 有空间时才请求接收数据。	0x0
[13:12]	-	reserved	保留。	0x0
[11]	RW	rts	请求发送。 该 bit 为 UART modem 状态输出信号 nUARTRTS 的取反。 0：输出信号不变; 1：即该 bit 配置为 1，则输出信号为 0。	0x0
[10]	-	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[9]	RW	rxe	UART 接收使能。 0：禁止； 1：使能。 在接收的过程中如果 UART 被禁止，则当前数据的接收就会在正常停止之前结束。	0x1
[8]	RW	txe	UART 发送使能。 0：禁止； 1：使能。 在发送的过程中如果 UART 被禁止，则当前数据的发送就会在正常停止之前结束。	0x1
[7]	RW	lbe	环回使能。 0：禁止； 1：UARTTXD 输出内部环回到 UARTRXD。	0x0
[6:1]	-	reserved	保留。	0x00
[0]	RW	uarten	UART 使能。 0：禁止； 1：使能。 如果在发送和接收过程中将 UART 禁止，则会在正常停止之前结束当前数据的传送。	0x0

UART_IFLS

UART_IFLS 为中断 FIFO 水线选择寄存器，用于设置 FIFO 的中断（UART_TXINTR 或 UART_RXINTR）触发水线。FIFO 的中断只有在 FIFO 内数据量通过设定水线时才触发。接收 FIFO 水线中断在接收 FIFO 内有效数据个数增加且通过设定水线时触发，触发一次。发送 FIFO 水线中断在发送 FIFO 内有效数据个数减少且通过设定水线时触发一次。

Offset Address: 0x034 Total Reset Value: 0x0000_0101

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:12]	-	reserved	保留。	0x00000
[11:8]	RW	rxiflsel	配置接收 FIFO 的水线。当接收 FIFO 中的数据个数满足条件时，产生接收 FIFO 中断（RXRIS）。如果使能了接收 FIFO DMA 功能，产生接收 FIFO DMA burst 传输请求。 0x0：大于或等于 0； 0x1：大于或等于 1； 0x2：大于或等于 2； 0x3：大于或等于 3； ... 0xF：大于或等于 15。 如果 DMA 控制器配置为搬移 FIFO 的数据，那么接收 FIFO 的水线需要大于 DMA 控制器的 Burst 长度	0x1
[7:4]	-	reserved	保留。	0x0
[3:0]	RW	txiflsel	配置发送 FIFO 的水线。当发送 FIFO 中的数据个数满足条件时，产生发送 FIFO 中断（TXRIS）。如果使能了发送 FIFO DMA 功能，同时产生发送 FIFO DMA burst 传输请求。 0x0：等于 0； 0x1：小于或等于 1； 0x2：小于或等于 2； 0x3：小于或等于 3； ... 0xF：小于或等于 15。 如果 DMA 控制器配置为搬移 FIFO 的数据，那么发送 FIFO 的水线需要小于 DMA 控制器的 Burst 长度。	0x1

UART_IMSC

UART_IMSC 为中断屏蔽寄存器，用于屏蔽中断。当读寄存器时，读出的是相关中断屏蔽的当前值。当对寄存器写 1 时，不屏蔽相应中断。写 0 屏蔽相应中断。

Offset Address: 0x038 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RW	cmim	字符匹配中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断	0x0
[20]	RW	abdeim	自动波特率检测错误中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[19]	RW	abdcim	自动波特率检测完成中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[18]	RW	rxffim	接收 FIFO 满中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[17]	RW	rxfneim	接收 FIFO 非空中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[16]	RW	rxfeim	接收 FIFO 空中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[15:14]	-	reserved	保留。	0x0
[13]	RW	txfneim	发送 FIFO 非空中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0

Bits	Access	Name	Description	Reset
[12]	RW	txfeim	发送 FIFO 空中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[11]	-	reserved	保留。	0x0
[10]	RW	oeim	溢出错误中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[9]	RW	beim	break 错误中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[8]	RW	peim	校验中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[7]	RW	feim	帧错误中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[6]	RW	rtim	接收超时中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[5]	RW	txim	发送中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[4]	RW	rxim	接收中断的屏蔽状态。 0：屏蔽该中断； 1：不屏蔽该中断。	0x0
[3:2]	-	reserved	保留。	0x0
[1]	RW	ctsmim	CTS 中断的屏蔽状态。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽该中断； 1：不屏蔽该中断。	
[0]	-	reserved	保留。	0x0

UART_RIS

UART_RIS 为原始中断状态寄存器。

Offset Address: 0x03C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RO	cmris	原始的字符匹配中断状态。 0：未产生中断； 1：已产生中断。	0x0
[20]	RO	abderis	原始的自动波特率检测错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[19]	RO	abdcris	原始的自动波特率检测完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[18]	RO	rxffris	原始的接收 FIFO 满中断状态。 0：未产生中断； 1：已产生中断。	0x0
[17]	RO	rxfneris	原始的接收 FIFO 非空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[16]	RO	rxferis	原始的接收 FIFO 空中断状态。 0：未产生中断； 1：已产生中断。	0x0

Bits	Access	Name	Description	Reset
[15]	-	reserved	保留。	0x0
[14]	RO	txtcris	发送完成状态位。 0：发送完成条件未触发； 1：发送完成条件触发。	0x0
[13]	RO	txfneris	原始的发送 FIFO 非空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[12]	RO	txferis	原始的发送 FIFO 空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[11]	-	reserved	保留。	0x0
[10]	RO	oeris	原始的溢出错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[9]	RO	beris	原始的 break 错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[8]	RO	peris	原始的校验中断状态。 0：未产生中断； 1：已产生中断。	0x0
[7]	RO	feris	原始的帧错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[6]	RO	rtris	原始的接收超时中断状态。 0：未产生中断； 1：已产生中断。 **注意：只有当 UART_IMSC[rtim] = 1 时，该**	0x0

Bits	Access	Name	Description	Reset
			位才能被置位，这样可以通过屏蔽位降低 UART 功耗。因此通过读该位和 UART_MIS[rtmis]可得到同样的接收超时中断状态。	
[5]	RO	txris	原始的发送水线中断状态。 0：未产生中断； 1：已产生中断。	0x0
[4]	RO	rxris	原始的接收水线中断状态。 0：未产生中断； 1：已产生中断。	0x0
[3:2]	-	reserved	保留。	0x0
[1]	RO	ctsmis	原始的 CTS 中断状态。 0：未产生中断； 1：已产生中断。	0x0
[0]	-	reserved	保留。	0x0

UART_MIS

UART_MIS 为屏蔽后中断状态寄存器。

Offset Address: 0x040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	RO	cmmis	屏蔽后的字符匹配中断状态。 0：未产生中断； 1：已产生中断。	0x0
[20]	RO	abdemis	屏蔽后的自动波特率检测错误中断状态。 0：未产生中断； 1：已产生中断。	0x0

Bits	Access	Name	Description	Reset
[19]	RO	abdcmis	屏蔽后的自动波特率检测完成中断状态。 0：未产生中断； 1：已产生中断。	0x0
[18]	RO	rxffmis	屏蔽后的接收 FIFO 满中断状态。 0：未产生中断； 1：已产生中断。	0x0
[17]	RO	rxfnemis	屏蔽后的接收 FIFO 非空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[16]	RO	rxfemis	屏蔽后的接收 FIFO 空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[15:14]	-	reserved	保留。	0x0
[13]	-	reserved	保留。	0x0
[12]	RO	txfemis	屏蔽后的发送 FIFO 空中断状态。 0：未产生中断； 1：已产生中断。	0x0
[11]	-	reserved	保留。	0x0
[10]	RO	oemis	屏蔽后的溢出错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[9]	RO	bemis	屏蔽后的 break 错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[8]	RO	pemis	屏蔽后的校验中断状态。 0：未产生中断； 1：已产生中断。	0x0

Bits	Access	Name	Description	Reset
[7]	RO	femis	屏蔽后的帧错误中断状态。 0：未产生中断； 1：已产生中断。	0x0
[6]	RO	rtmis	屏蔽后的接收超时中断状态。 0：未产生中断； 1：已产生中断。	0x0
[5]	RO	txmis	屏蔽后的发送中断状态。 0：未产生中断； 1：已产生中断。	0x0
[4]	RO	rxmis	屏蔽后的接收中断状态。 0：未产生中断； 1：已产生中断。	0x0
[3:2]	-	reserved	保留。	0x0
[1]	RO	ctsmmis	屏蔽后的 CTS 中断状态。 0：未产生中断； 1：已产生中断。	0x0
[0]	-	reserved	保留。	0x0

UART_ICR

UART_ICR 为中断清除寄存器，写 1 时相应的中断被清除，写 0 则不起作用。对于溢出错误中断，除了对此寄存器 bit10 写 1 以清除原始溢出错误中断外，还需要写 UART_RSR[oe]以清除溢出错误状态。

Offset Address: 0x044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:22]	-	reserved	保留。	0x000
[21]	WO	cmic	清除字符匹配中断状态。 0：无效；	0x0

Bits	Access	Name	Description	Reset
			1：清除中断。	
[20]	WO	abdeic	清除自动波特率检测错误中断状态。 0：无效； 1：清除中断。	0x0
[19]	WO	abdcic	清除自动波特率检测完成中断状态。 0：无效； 1：清除中断。	0x0
[18]	WO	rxffic	清除接收 FIFO 满中断状态。 0：无效； 1：清除中断。	0x0
[17]	WO	rxfneic	清除接收 FIFO 非空中断状态。 0：无效； 1：清除中断。	0x0
[16]	WO	rxfeic	清除接收 FIFO 空中断状态。 0：无效； 1：清除中断。	0x0
[15]	-	reserved	保留。	0x0
[14]	WO	txtcic	清除发送完成中断状态。 0：无效； 1：清除中断。	0x0
[13]	WO	txfneic	清除发送 FIFO 非空中断状态。 0：无效； 1：清除中断。	0x0
[12]	WO	txfeic	清除发送 FIFO 空中断状态。 0：无效； 1：清除中断。	0x0
[11]	-	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[10]	WO	oeic	清除溢出错误中断。 0：无效； 1：清除中断。	0x0
[9]	WO	beic	清除 break 错误中断。 0：无效； 1：清除中断。	0x0
[8]	WO	peic	清除校验中断。 0：无效； 1：清除中断。	0x0
[7]	WO	feic	清除帧错误中断。 0：无效； 1：清除中断。	0x0
[6]	WO	rtic	清除接收超时中断。 0：无效； 1：清除中断。	0x0
[5]	WO	txic	清除发送中断。 0：无效； 1：清除中断。	0x0
[4]	WO	rxic	清除接收中断。 0：无效； 1：清除中断。	0x0
[3:2]	-	reserved	保留。	0x0
[1]	WO	ctsmic	清除 CTS 中断。 0：无效； 1：清除中断。	0x0
[0]	-	reserved	保留。	0x0

UART_DMACR

UART_DMACR 为 DMA 控制寄存器，用于配置发送通道和接收通道的 DMA 使能。

Offset Address: 0x048 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3]	RW	rxlastsreq_en	UART RX DMA 支持最后一笔数据的 REQ 使能。使能打开后，传输最后一笔数据时会发出 LASTREQ 给 DMA 控制器。可以自动对 DMA 进行硬件流控。 0：禁止； 1：使能。	0x0
[2]	RW	dmaonerr	UART 错误中断（UARTEINTR）出现时的接收通道 DMA 使能控制。 0：当 UART 错误中断（UARTEINTR）有效时，接收通道 DMA 的请求输出（UARTRXDMASREQ 或 UARRTXDMABREQ）有效； 1：当 UART 错误中断（UARTEINTR）有效时，接收通道 DMA 的请求输出（UARTRXDMASREQ 或 UARRTXDMABREQ）无效。	0x0
[1]	RW	txdmae	发送通道的 DMA 使能控制。 0：禁止； 1：使能。	0x0
[0]	RW	rxdmae	接收通道的 DMA 使能控制。 0：禁止； 1：使能。	0x0

UART_DS

UART_DS 为数据位序配置寄存器，用于配置数据位序。

Offset Address: 0x050 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	-	reserved	保留。	0x00000000
[0]	RW	msbfirst	最高有效位在前使能。 0：在起始位之后，首先使用数据位的 LSB 发送/接收数据； 1：在起始位之后，首先使用数据位的 MSB 发送/接收数据。	0x0

UART_RTCFG

UART_RTCFG 为接收超时时长配置寄存器，用于配置判断接收超时的条件。

Offset Address: 0x054 Total Reset Value: 0x0000_001F

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:0]	RW	rtcfg	配置接收超时时长，单位是 bit 时长。 接收超时时长=配置值+1。	0x001F

UART_SPCFG

UART_SPCFG 为过采样配置寄存器，用于配置过采样倍数。

Offset Address: 0x058 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	-	reserved	保留。	0x0000000
[3:0]	RW	spcfg	配置过采样倍数。 0x0：16 倍过采样； 0x1：15 倍过采样； 0x2：14 倍过采样； 0x3：13 倍过采样； 0x4：12 倍过采样；	0x0

Bits	Access	Name	Description	Reset
			其他：禁止配置。	

UART_ABDEN

UART_ABDEN 为自动波特率检测使能寄存器，用于配置自动波特率检测使能。

Offset Address: 0x05C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	-	reserved	保留。	0x0000000
[5]	RO	abdenvld	abden 已经有效的标志。 0：abden 还未生效； 1：abden 已经生效，UART 可以接收大写字符 Z 进行自动波特率检测。	0x0
[4]	RO	abdbusy	自动波特率检测忙标志。 0：不在进行自动波特率检测； 1：正在进行自动波特率检测。	0x0
[3:1]	-	reserved	保留	0x0
[0]	RW	abden	自动波特率检测使能。 0：禁止自动波特率检测； 1：使能自动波特率检测。 如果 UART_CR[uarten]使能，且使能 UART_CR[rxe]或 UART_CR[txe]，则 abden 会被自动清零。	0x0

UART_CHARMATCH

UART_CHARMATCH 为字符匹配配置寄存器，用于配置需要匹配的字符。

Offset Address: 0x060 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31]	RW	charmaten	字符匹配检测使能。	0x0

Bits	Access	Name	Description	Reset
			0：禁止字符匹配检测； 1：使能字符匹配检测。	
[30:8]	-	reserved	保留。	0x000000
[7:0]	RW	charmat	需要匹配的字符二进制。	0x00

21 同步串行外设接口(SPI)

21.1 概述

SPI 控制器实现数据的串并、并串转换，可以作为 Master（主机）与外部设备进行同步串行通信，也可作为 Slave（从机）与外设对接。支持 Motorola SPI 接口、TI 串行同步接口和 MicroWire 接口三种外设接口协议。

21.2 特性

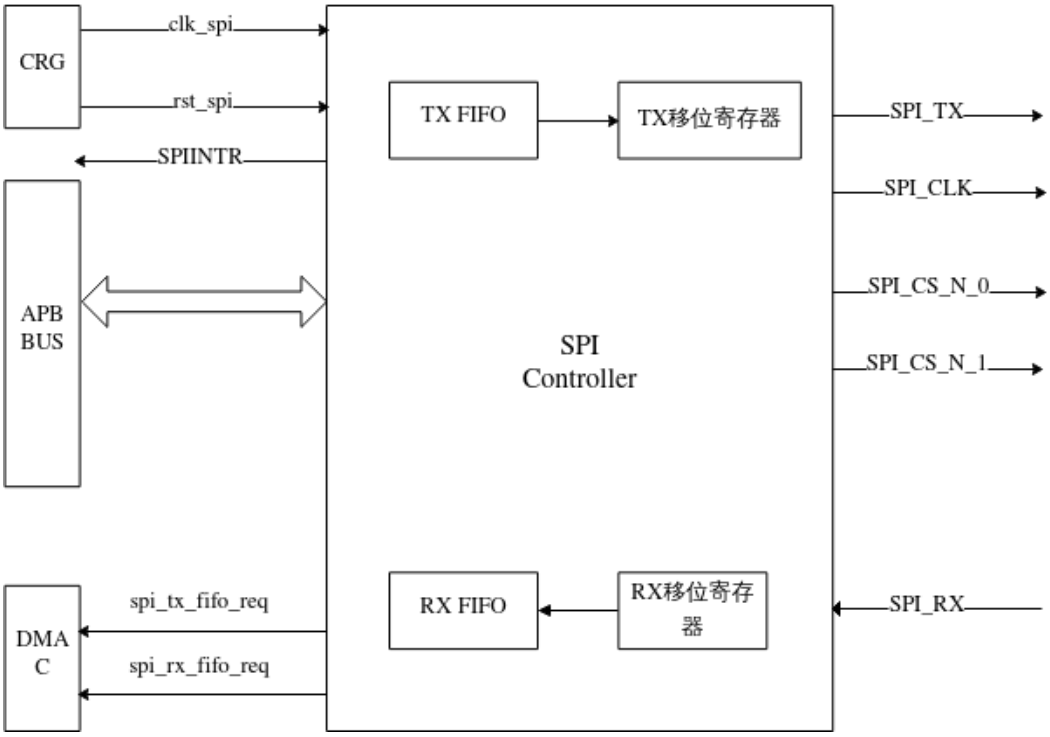
SPI 的功能特点有：

- 支持接口时钟频率可编程。
- 支持主模式和从模式。
- 支持双片选。片选支持逻辑自动控制，或者软件控制。
- 支持 16 x 16 bit 的 TX（发送） FIFO 和 16 x 16 bit 的 RX（接收）FIFO。
- 支持 4bit ~ 16bit 可编程的串行数据帧长度。
- 支持单帧和连续帧格式。
- 支持 Motorola SPI 全双工工作模式，时钟极性、相位可配置。
- 支持 MicroWire 半双工工作模式。
- 支持 TI 同步串行接口全双工工作模式。
- 支持 DMA 操作。

21.3 功能描述

SPI 控制器，如图 21-1 所示。CPU 通过总线配置 SPI 控制器。SPI 控制器与外围 SPI 设备对接通信，软件将需要写入的数据写入 TX FIFO，然后 SPI 控制器转换成 SPI 总线协议发送给设备，软件也可以从 RX FIFO 中读回外围设备发送给 SPI 控制器的数据。SPI 控制器支持中断模式，支持向 CPU 上报中断。SPI 控制器也支持 DMA 模式，不通过 CPU，在 SPI 控制器和存储器之间直接进行数据传输。

图21-1 SPI 控制器在 MCU 中的位置

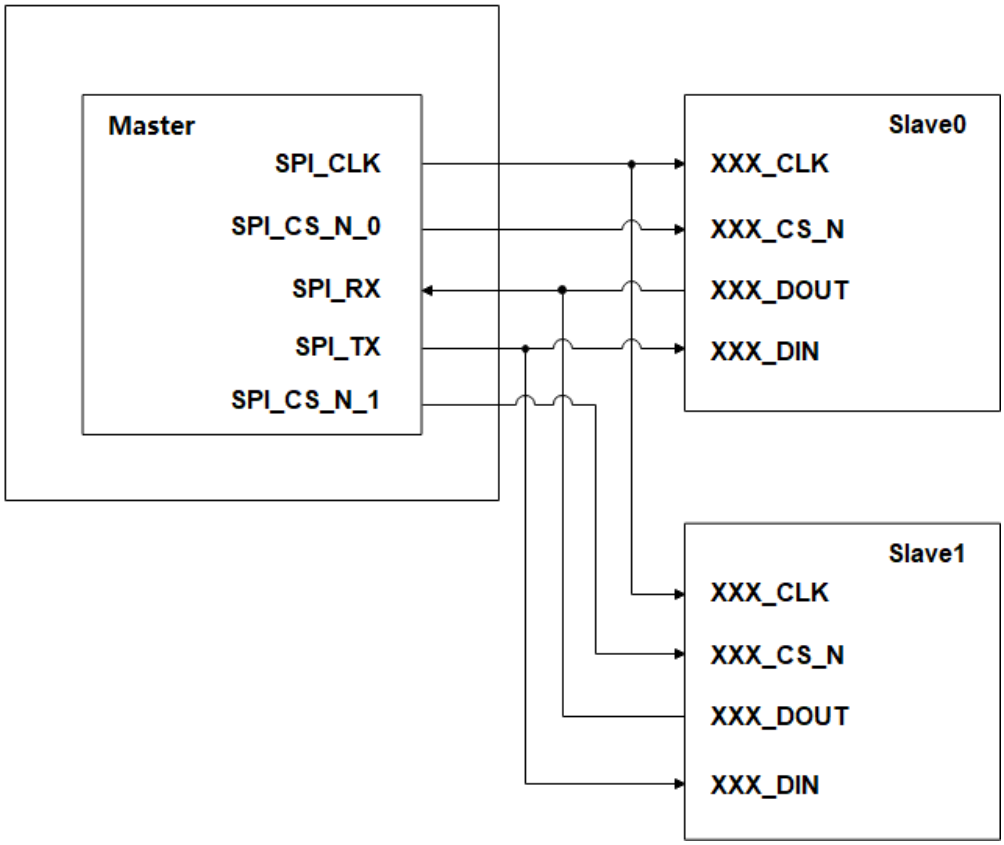


SPI 控制器支持接口时钟频率可编程，接口时钟频率与 clk_spi、串行时钟速率以及时钟分频因子有关，通过配置 SPICR0[scr]和 SPICPSR[cpsdvsr]来调整 SPI 通信速率，具体请参见 21.5.2 时钟与复位章节。

SPI 控制器支持主模式和从模式，通过配置 SPICR1[ms]以选择主/从模式。SPI 总线规定了 2 个 SPI 设备之间通信必须由主设备（Master）来控制从设备（Slave）。一个 Master 设备可以通过提供时钟 SPI_CLK 以及对 Slave 设备进行片选（SPI_CS_N）来控制多个 Slave 设备。SPI 总线还规定 Slave 设备的时钟由 Master 设备通过时钟 SPI_CLK 管脚提供给 Slave 设备，Slave 设备本身不能产生或控制时钟。如果没有时钟，那么 Slave 设备不能正常工作。

SPI 控制器支持两片选。图 21-2 描述了 SPI 接 2 个 Slave 时的连接关系。Master 的 SPI_CS_N_0 和 Slave0 的 XXX_CS_N 相连，SPI_CS_N_1 和 Slave1 的 XXX_CS_N 相连。通过配置 SPICSNCR[spi_csn_sel]来设置片选选择，配置为 00 时，选通 Slave0；配置为 01 时，则选通 Slave1。

图21-2 SPI 接 Slave 时的连接关系



SPI 控制器支持 16 x 16 bit 的 TX FIFO 和 16 x 16 bit 的 RX FIFO。写操作时，写入的数据通过 SPIDR[data]进入 TX FIFO；读操作时，通过 SPIDR[data]从 RX FIFO 中读出数据。此外，通过配置 SPITXFIFOCR[txintsize]和 SPIRXFIFOCR[rxintsize]，设置 TX FIFO 和 RX FIFO 的水线。用户通过读取 SPISR 寄存器，可以获取到 TX FIFO 和 RX FIFO 的当前状态。

SPI 控制器支持 4bit～16bit 可编程的串行数据帧长度，通过配置 SPICR0[dss]改变。

SPI 控制器支持单帧和连续帧格式，可以发送单帧数据，也可以连续发送连续帧数据，具体内容请参见 21.4 接口时序章节。

SPI 控制器支持 Motorola SPI 全双工工作模式、MicroWire 半双工工作模式和 TI 同步串行接口全双工工作模式，通过配置 SPICR0[frf]选择工作模式。在 Motorola SPI 全双

工工作模式下，通过配置 SPICR0[sph]和 SPICR0[spo]以分别设置时钟相位和时钟极性。

SPI 控制器的接口时序请参见 21.4 接口时序章节。

SPI 控制器支持 DMA 操作，支持从内存搬运数据至 TX FIFO 再发送至从机，也支持将读回的数据存入 RX FIFO 再搬运至内存。SPI 控制器支持 single 和 burst DMA 操作，single DMA 操作无需配置，在 TX FIFO 有空位时，产生 TX FIFO single DMA 请求，在 RX FIFO 有数据时，产生 RX FIFO single DMA 请求。burst DMA 操作的 TX FIFO 和 RX FIFO 的水线分别通过配置 SPITXFIFOCR[txintsize]和 SPIRXFIFOCR[rxintsize]设置。具体工作方式请参见 21.5 工作方式章节的 DMA 方式下的数据传输部分。

21.4 接口时序

图 21-4～图 21-11 中的缩略语含义为：

- MSB：Most Significant Bit，最高位置的 bit。
- LSB：Least Significant Bit，最低位置的 bit。
- Q：Q is an undefined signal，未定义。

须知

时序图中的 SPI_CS_N 是 SPI 片选信号，软件通过配置 SPICSNCR[spi_csn_sel]选择。

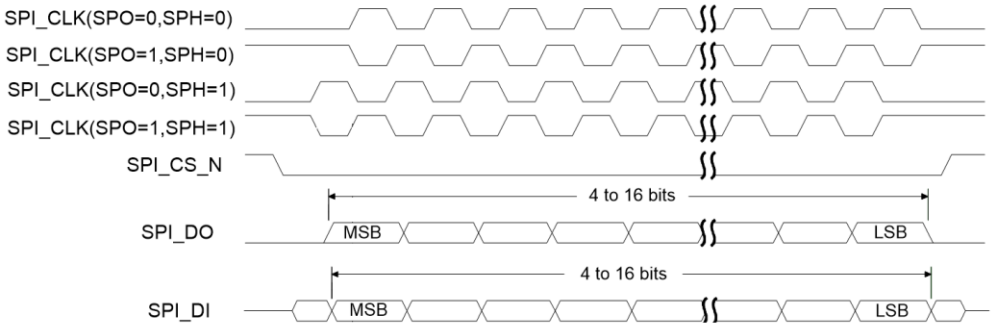
21.4.1 Motorola SPI 接口

说明

SPO 表示 SPICLKOUT 极性，SPH 表示 SPICLKOUT 相位。它们是寄存器 SPICR0 bit[7:6]。

21.4.1.1 接口时序概览

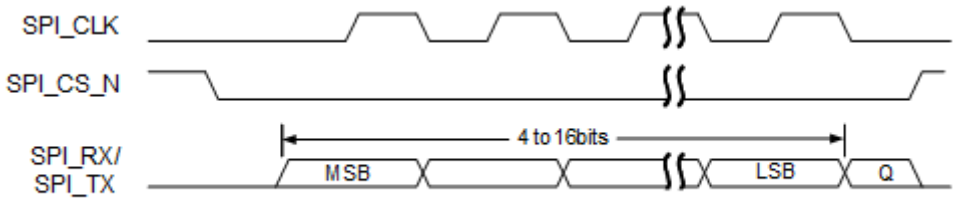
图21-3 SPI 接口时序概览图



21.4.1.2 SPO=0、SPH=0

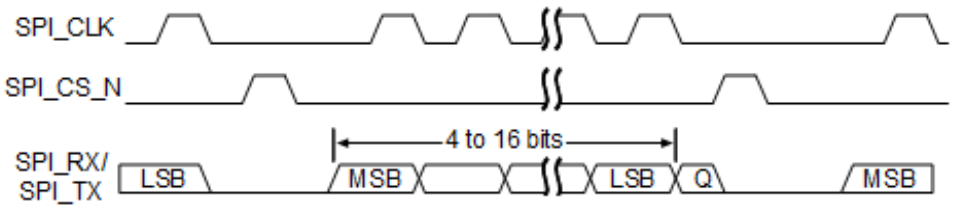
SPI 单帧帧格式如图 21-4 所示。

图21-4 SPI 单帧帧格式（SPO=0、SPH=0）



SPI 连续帧帧格式如图 21-5 所示。

图21-5 SPI 连续帧帧格式（SPO=0、SPH=0）



在该模式下，当 SPI 处于空闲状态时：

- SPI_CLK 信号为低。
- SPI_CS_N 信号为高。
- 发送数据线 SPI_TX 强制为低。

当 SPI 处于使能状态，而且发送 FIFO 有数据时，SPI 总线的 Master 输出 SPI_CS_N 信号为低，表示开始传输数据。来自对端的数据立刻发送到 SPI 控制器的接收数据线 SPI_RX。半个 SPI_CLK 时钟周期之后，SPI 控制器将数据传输到 SPI_TX。此时 Master 和 Slave 数据都已经有效，SPI_CLK 管脚在接下来的半个 SPI_CLK 时钟周期之后变为高电平。数据在 SPI_CLK 时钟的上升沿被采样，在时钟的下降沿被发送。

如果传输单个 word（单帧），当捕捉到最后 1bit 数据时，SPI_CS_N 在接下来的 1 个 SPI_CLK 时钟周期之后恢复为高电平。

如果是连续的传输（连续帧），SPI_CS_N 信号在每个 word 传输之间必须拉高一个时钟周期。这是因为 SPH 为 0 时，SPI_CS_N 会固定 Slave 内部串行设备寄存器的数据，使它不会变化。因此在连续传输时，主设备必须在每个 word 传输之间将 SPI_CS_N 信号拉高。连续传输结束时，SPI_CS_N 在捕获到最后 1bit 之后的 1 个 SPI_CLK 时钟周期之后恢复为高电平。

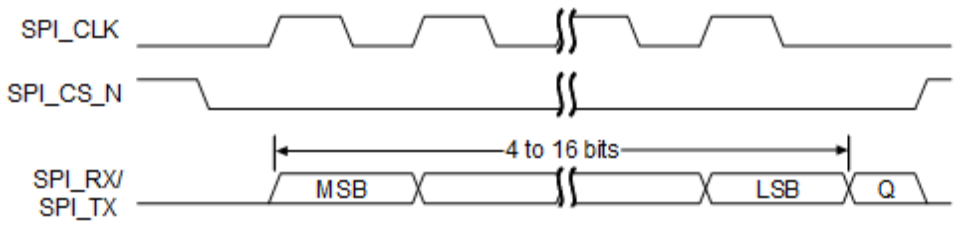
说明

如果在连续帧传输过程中，预期的 SPI_CS_N 信号与上述的行为不匹配，那么软件需要通过配置 SPICSNCR[spi_csn_mode]直接控制 SPI_CS_N 信号的高低。

21.4.1.3 SPO=0、SPH=1

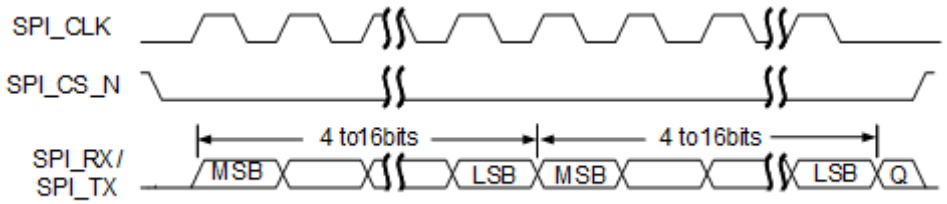
SPI 单帧帧格式如图 21-6 所示。

图21-6 SPI 单帧帧格式（SPO=0、SPH=1）



SPI 连续帧帧格式如图 21-7 所示。

图21-7 SPI 连续帧帧格式（SPO=0、SPH=1）



在该模式下，当 SPI 处于空闲状态时：

- SPI_CLK 信号为低。
- SPI_CS_N 为高。
- 发送数据线 SPI_TX 强制为低。

当 SPI 为使能状态，而且发送 FIFO 有数据时，SPI 总线的 Master 输出 SPI_CS_N 信号为低表示开始传输数据。半个 SPI_CLK 时钟周期之后，Master 和 Slave 的有效数据分别在各自的传输线上有效。同时，SPI_CLK 从第一个上升沿开始有效。数据在 SPI_CLK 时钟的下降沿被采样，在时钟的上升沿被发送。

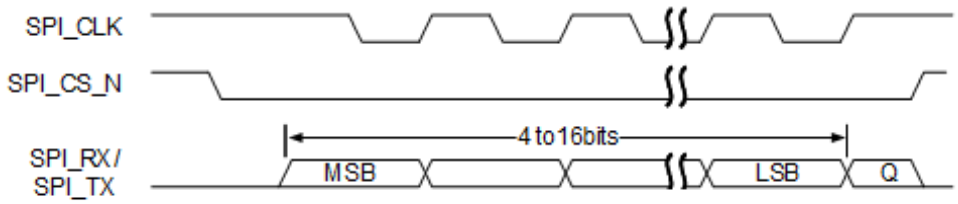
如果传输单个 word（单帧），当采样到最后 1bit 数据时，SPI_CS_N 在接下来的 1 个 SPI_CLK 时钟之后恢复为高电平。

当连续传输时（连续帧），在传输数据 word 之间 SPI_CS_N 保持为低。连续传输结束时，SPI_CS_N 在最后 1bit 采样之后的 1 个 SPI_CLK 时钟之后恢复为高电平。

21.4.1.4 SPO=1、SPH=0

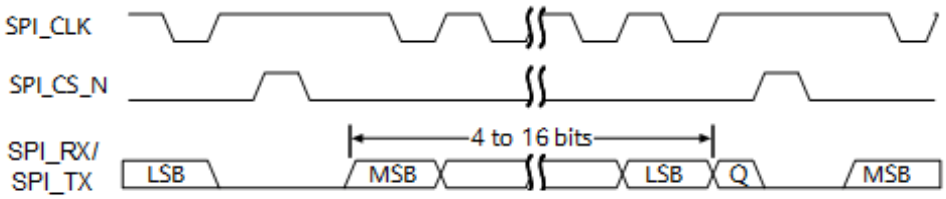
SPI 单帧帧格式如图 21-8 所示。

图21-8 SPI 单帧帧格式（SPO=1、SPH=0）



SPI 连续帧帧格式如图 21-9 所示。

图21-9 SPI 连续帧帧格式（SPO=1、SPH=0）



在该配置下，当 SPI 处于空闲状态时：

- SPI_CLK 信号为高。
- SPI_CS_N 信号为高。
- 发送数据线 SPI_TX 强制为低。

当 SPI 为使能状态，而且发送 FIFO 有数据时，SPI 总线的 Master 输出 SPI_CS_N 信号为低表示开始传输数据。此时 Slave 的数据立刻发送到 Master 的接收数据线 SPI_RX。半个 SPI_CLK 周期之后，Master 的有效数据传送到 SPI_TX。再过半个 SPI_CLK 时钟周期之后，Master 输出 SPI_CLK 为低。这表示数据在 SPI_CLK 时钟的下降沿被采样，在 SPI_CLK 时钟的上升沿被发送。

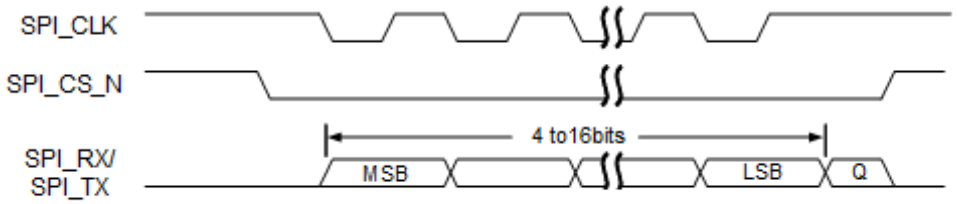
如果传输单个 Word（单帧），当采样到最后 1bit 数据时，SPI_CS_N 在接下来的 1 个 SPI_CLK 时钟之后恢复为高电平。

如果是连续的传输（连续帧），SPI_CS_N 信号在每个 Word 传输之间必须拉高。这是因为当 SPH 为 0 时，SPI_CS_N 固定其内部串行设备寄存器的数据，使它不会变化。SPI_CS_N 在采样到最后 1bit 数据之后的 1 个 SPI_CLK 时钟周期之后恢复为高电平。

21.4.1.5 SPO=1、SPH=1

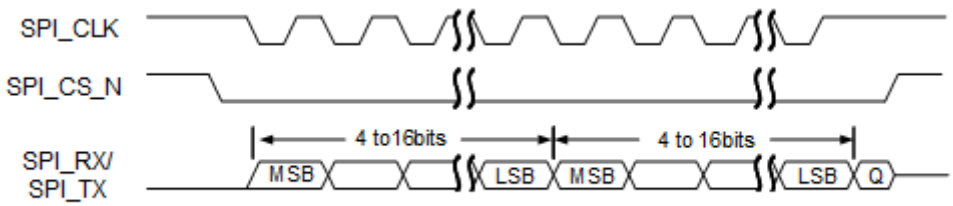
SPI 单帧帧格式如图 21-10 所示。

图21-10 SPI 单帧帧格式（SPO=1、SPH=1）



SPI 连续帧帧格式如图 21-11 所示。

图21-11 SPI 连续帧帧格式（SPO=1、SPH=1）



在该模式下，当 SPI 处于空闲状态时：

- SPI_CLK 信号为高。
- SPI_CS_N 信号为高。
- 发送数据线 SPI_TX 强制为低。

当 SPI 为使能状态，而且发送 FIFO 有数据时，SPI 总线的 Master 输出 SPI_CS_N 信号为低表示开始传输数据。半个 SPI_CLK 时钟周期后，Master 和 Slave 数据在各自的传输线上有效。同时，时钟 SPI_CLK 从第 1 个下降沿开始有效。数据在 SPI_CLK 时钟的上升沿被采样，在时钟的下降沿被发送。

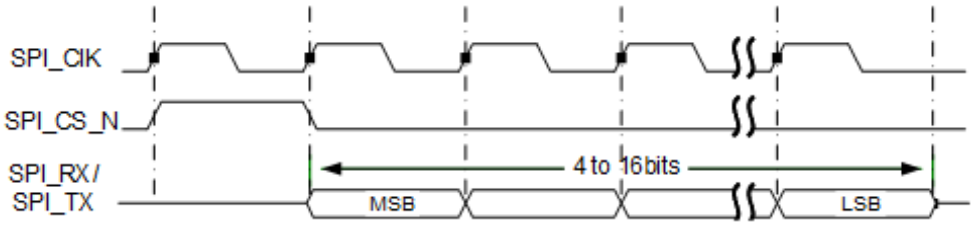
如果是传输单个 Word（单帧），SPI_CS_N 在最后 1bit 被采样之后的 1 个 SPI_CLK 时钟周期之后恢复为高电平。

如果是连续传输（连续帧），SPI_CS_N 信号始终保持为低。SPI_CS_N 在采样到最后 1bit 之后的 1 个 SPI_CLK 时钟周期之后恢复到高电平。对于连续传输（连续帧），SPI_CS_N 在传输过程中一直保持为低，结束方式与单帧相同。

21.4.2 TI 同步串行接口

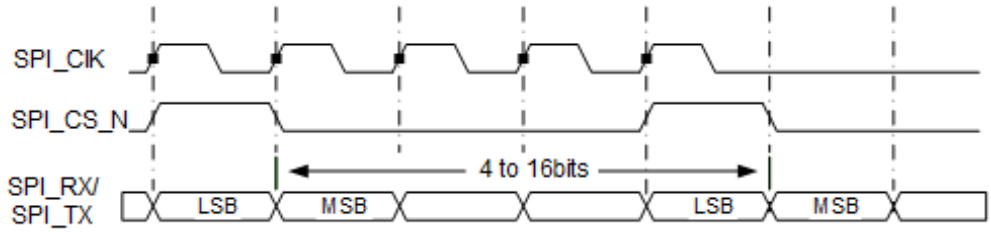
TI 同步串行单帧帧格式如图 21-12 所示。

图21-12 TI 同步串行单帧帧格式



TI 同步串行连续帧帧格式如图 21-13 所示。

图21-13 TI 同步串行连续帧帧格式



在该模式下，当 SPI 处于空闲状态时：

- SPI_CLK 为低电平。
- SPI_CS_N 为低电平。
- 传输数据线 SPI_TX 保持为高阻。

一旦发送 FIFO 有数据，SPI_CS_N 就会产生一个 SPI_CLK 时钟周期的高电平脉冲，被发送的数据就会从发送 FIFO 传送到发送串行移位寄存器。在 SPI_CLK 时钟的下一个上升沿，4bit～16bit 数据帧的 MSB 就会从 SPI_TX 移位输出。同样，从外部设备串行接收数据的 MSB 会从 SPI_RX 管脚移位输入。

SPI 控制器和片外串行设备在 SPI_CLK 时钟的下降沿将数据采样，存入串行移位寄存器。接收串行寄存器在接收到 LSB 之后的第一个 SPI_CLK 时钟上升沿将数据存入接收 FIFO。

21.4.3 Microwire 接口

Microwire 单帧帧格式如图 21-14 和图 21-15 所示。

图21-14 Microwire 单帧帧格式（不插入等待周期的 Master 模式，或 Slave 模式）

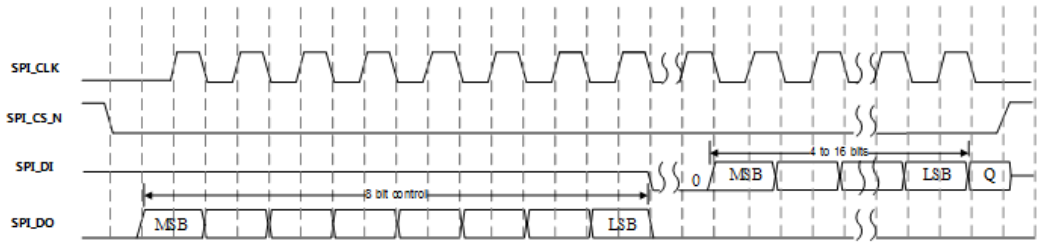
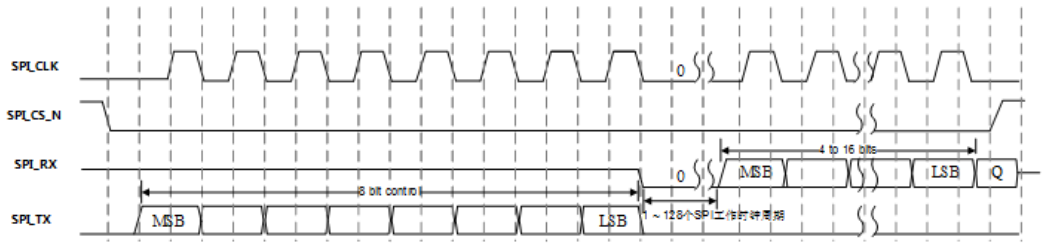


图21-15 Microwire 单帧帧格式（插入等待周期的 Master 模式)



Master 模式下，Microwire 帧在 SPI_TX LSB 结束和 SPI_RX MSB 开始之间可以延迟 1～128 个时钟周期（等待期间 SPI_CLK 保持为 0，RX 不接收数据)。

Microwire 连续帧帧格式如图 21-16 和图 21-17 所示。

图21-16 Microwire 连续帧帧格式（不插入等待周期的 Master 模式，或 Slave 模式)

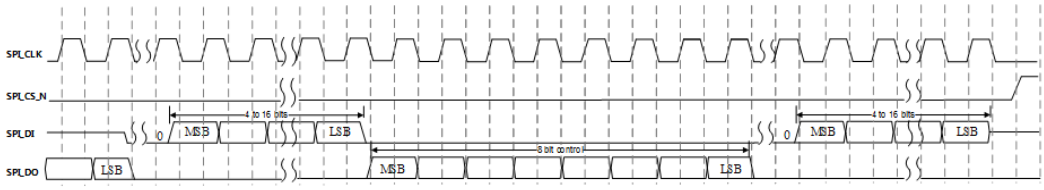
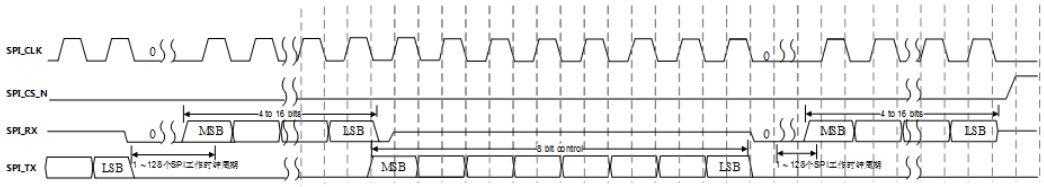


图21-17 Microwire 连续帧帧格式（插入等待周期的 Master 模式)



Master 模式下，Microwire 帧在 SPI_TX LSB 结束和 SPI_RX MSB 开始之间可以延迟 1～128 个时钟周期（等待期间 SPI_CLK 保持为 0，RX 不接收数据)。

说明

Slave 模式不支持插入等待周期。插入等待周期的功能，只对 Master 生效。

Microwire 的格式与 Motorola SPI 的格式非常相近，使用 master-slave 信息的传输技术，只不过 Motorola SPI 是全双工通信，而 Microwire 半双工通信。在 SPI 控制器向外部 MCU 发送串行数据的时候，都要先加 8bit 控制字。在这个过程中，SPI 控制器没有接收到任何数据。传输完毕之后，SPI 总线上的其他器件对接收到的数据进行解码。

用户可以通过配置 SPICR1[waitval]为 n（等待 n+1 个 clk_spi 工作时钟周期），且配置 SPICR1[waiten]为 1，在 SPI_TX LSB 结束和 SPI_RX MSB 开始之间插入等待周期，等待期间 SPI_CLK 保持为 0，RX 不接收数据。那么在与 8bit 控制信息间隔 n+1 个时钟周期之后，SPI_CLK 正常输出，slave 开始响应所需求的数据。返回的数据为 4bit～16bit 数据。使得整个帧的长度为 12bit～24bit。插入等待周期的单帧格式和连续帧格式如图 21-15 和图 21-17 所示。

用户也可以通过不配置 SPICR1[waitval]，且配置 SPICR1[waiten]为 0，在 SPI_TX LSB 结束和 SPI_RX MSB 开始之间不会插入等待周期。那么在 8bit 控制信息间隔 1 个时钟周期之后，slave 开始响应所需求的数据。返回的数据为固定的 1bit 0 加上 4bit～16bit 数据，使得整个帧的长度为 13bit～25bit。不插入等待周期的单帧格式和连续帧格式的时序如图 21-14 和图 21-16 所示。

在该模式下，当 SPI 处于空闲状态时：

- SPI_CLK 信号为低电平。
- SPI_CS_N 为高电平。
- 发送数据线 SPI_TX 强制为低电平。

向发送 FIFO 内部写进一个控制字节开始一次传输。SPI_CS_N 下降沿后，数据开始传输，发送 FIFO 的数据被发送到串行移位寄存器，8bit 控制帧的 MSB 被发送到发送管脚 SPI_TX。在帧的传输过程中，SPI_CS_N 保持为低。SPI_RX 在这个传输过程中保持为高阻。

片外的串行从设备在 SPI_CLK 时钟的每一个上升沿将数据锁存到串行移位寄存器中。当从设备锁存完最后 1 个 bit 的数据之后，在接下来的 1 个时钟周期的等待时间里，对接收到的数据开始解码，然后从设备在 SPI_CLK 时钟的下降沿发送数据到 SPI_RX。对单个数据（单帧）传输来说，在帧的结尾，SPI_CS_N 在最后 1 个 bit 写到接收串行寄存器之后的 1 个时钟周期后拉高，这样就使接收到的数据写入接收 FIFO。

对于连续的传输（连续帧），数据传输的开始和结束都和单个数据（单帧）的传输方式相同。在这个传送过程中，SPI_CS_N 一直保持为低，传输的数据也是连续的。下一帧的控制字直接和上一帧的 LSB 相邻。

21.5 工作方式

21.5.1 工作模式

SPI 的工作模式分为中断、查询、和 DMA 方式下的数据传输。

21.5.2 时钟与复位

输出 SPI 接口时钟频率计算方式如下：

Fspiclkout = Fclk_spi/(cpsdvsr ×(1+scr))

- Fclk_spi：SPI 的工作参考时钟 150MHz。
- cpsdvsr、scr 请查询状态寄存器 SPICPSR、控制寄存器 SPICR0。

支持独立软复位和时钟门控，具体请参见 5.5 CRG_PERIAPBIF 寄存器描述章节。

21.5.3 中断处理

SPI 有 5 个中断，其中前 4 个是独立中断源、可屏蔽、高电平有效。最后 1 个组合中断。

- SPIRXINTR

  接收 FIFO 中断请求。当接收 FIFO 的有效数据个数大于等于接收 FIFO 中断水线时，该中断置位。软件可配置 SPIRXFIFOCR[rxintsize]，配置接收 FIFO 中断水线。
- SPITXINTR

  发送 FIFO 中断请求。当发送 FIFO 的有效数据个数小于等于发送 FIFO 中断水线时，该中断置位。软件可配置 SPITXFIFOCR[txintsize]，配置发送 FIFO 中断水线。
- SPIRORINTR

  接收溢出中断请求。当 FIFO 已满，且又有新的数据需要写入 FIFO 时，会引起 FIFO 溢出，该中断置位。此时数据被写入接收移位寄存器，而不是 FIFO。通过写寄存器 SPIICR[roric]清除该中断。
- SPIRTINTR

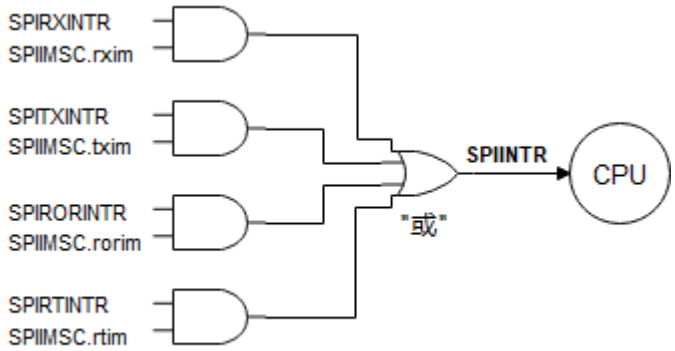
  接收超时中断请求。当接收 FIFO 非空且 SPI_CLK 未发生翻转，超过 64 个 SPI 工作时钟周期，该中断置位。

  此时表明接收 FIFO 中仍有数据需要读取。如果接收 FIFO 被读空或者当有新的数据被接收到 SPIRXD 中，该中断解除置位。也可以通过写寄存器 SPIICR[rtic]清除该中断。
- SPIINTR

  组合中断，为以上 4 个中断经过“或”运算后的结果。如果上述 4 个独立中断中任意一个置位且使能（未被屏蔽），该中断置位。

SPI 的中断 SPIINTR 如图 21-18 所示，4 个原始中断可以通过 SPIRIS 寄存器查询；也可以通过配置 SPIIMSC 相应位域进行中断屏蔽，屏蔽后的中断可以通过 SPIMIS 寄存器查询。

图21-18 图 SPI 中断处理



21.5.4 初始化

初始化步骤如下：

步骤 1 向寄存器 SPICR1[sse]写“0”，禁止 SPI。

步骤 2 配置 SPICPSR[spi_csn_sel]，选择片选信号。

步骤 3 写寄存器 SPICR0，配置帧格式及传输数据位宽等参数。

步骤 4 配置寄存器 SPICPSR，设定时钟分频因子。

步骤 5 中断方式下，设置寄存器 SPIIMSC，使能相应中断信号；查询、DMA 方式下，应禁止产生相应中断信号。

步骤 6 中断或 DMA 方式，设置 SPITXFIFOCR 和 SPIRXFIFOCR。

步骤 7 DMA 方式下，配置寄存器 SPIDMAR，使能 SPI 的 DMA 功能。

----结束

21.5.5 查询方式下的数据传输

具体步骤如下：

步骤 1 向寄存器 SPICR1[sse]写“1”，使能 SPI。

步骤 2 将需发送的数据连续写到寄存器 SPIDR。

步骤 3 轮询寄存器 SPIDR，直到 SPISR[bsy]为 0（表示总线不忙）、SPISR[tfe]为 1（表示发送 FIFO 已空）、SPISR[rne]为 1（表示接收 FIFO 非空）。

步骤 4 读出数据，需保证读空接收 FIFO（可通过查询 SPISR[rne]得到）。

须知

Motorola SPI/TI 同步串行接口的全双工特性，每发一个数据就接收一个数据，即使只需发送数据，也需要清空接收 FIFO。

若期望发送单帧，在步骤 2 中，软件须保证仅写入 1 个有效 Byte/Half-word。

若期望发送连续帧，在步骤 2 中，软件须保证有效的 Byte/Half-word 被连续写入发送 FIFO，一旦发送 FIFO 不为空，SPI 开始发送，直到发送 FIFO 为空，结束连续帧。若非连续写入发送 FIFO，在中途，发送 FIFO 可能空，SPI 提前结束连续帧，导致连续帧被截断。

步骤 5 向寄存器 SPICR1[sse]写“0”，禁止 SPI。

----结束

21.5.6 中断方式下的数据传输

具体步骤如下：

步骤 1 向寄存器 SPICR1[sse]写“1”，使能 SPI。

步骤 2 将需发送的数据连续写到寄存器 SPIDR。

步骤 3 等待中断 SPIRXINTR，读出数据。循环直到读出所有数据。

须知

Motorola SPI/TI 同步串行接口的全双工特性，每发一个数据就接收一个数据，即使只需发送数据，也需要清空接收 FIFO。

若期望发送单帧，在步骤 2 中，软件须保证仅写入 1 个有效 Byte/Half-word。

若期望发送连续帧，在步骤 2 中，软件须保证有效的 Byte/Half-word 被连续写入发送 FIFO，一旦发送 FIFO 不为空，SPI 开始发送，直到发送 FIFO 为空，结束连续帧。若非连续写入发送 FIFO，在中途，发送 FIFO 可能空，SPI 提前结束连续帧，导致连续帧被截断。

步骤 4 向寄存器 SPICR1[sse]写“0”，禁止 SPI。

----结束

21.5.7 DMA 方式下的数据传输

具体步骤如下：

步骤 1 获取一个 DMA 通道。

步骤 2 向寄存器 SPICR1[sse]写“1”，使能 SPI。

步骤 3 发送数据。

1. 配置该 DMA 通道的配置寄存器和控制寄存器中的相关参数。
2. 启动 DMA，响应 SPI 发送 FIFO 的 DMA 请求进行数据传输。
3. 通过 DMA 中断上报，判断数据是否发送完成，如果完成则关闭 SPI 的 DMA 功能。

步骤 4 接收数据。

1. 配置该 DMA 通道的配置寄存器和控制寄存器中的相关参数。
2. 启动 DMA，响应 SPI 接收 FIFO 的 DMA 请求进行数据传输。
3. 通过 DMA 中断上报，判断数据是否接收完成，如果完成则关闭 SPI 的 DMA 功能。

步骤 5 向寄存器 SPICR1[sse]写“0”，禁止 SPI。

----结束

21.6 SPI 寄存器概览

SPI 寄存器概览如表 21-1 所示。

SPI0 寄存器基地址为：0x1420_0000。

SPI1 寄存器基地址为：0x1420_1000。

表21-1 SPI 寄存器概览

偏移地址	名称	描述	页码
0x000	SPICR0	控制寄存器 0	SPICR0
0x004	SPICR1	控制寄存器 1	SPICR1
0x008	SPIDR	数据寄存器	SPIDR
0x00C	SPISR	状态寄存器	SPISR
0x010	SPICPSR	时钟分频寄存器	SPICPSR
0x014	SPIIMSC	中断屏蔽寄存器	SPIIMSC
0x018	SPIRIS	原始中断状态寄存器	SPIRIS
0x01C	SPIMIS	屏蔽后中断状态寄存器	SPIMIS
0x020	SPIICR	中断清除寄存器	SPIICR
0x024	SPIDMAR	DMA 控制寄存器	SPIDMAR
0x028	SPITXFIFOCR	发送 FIFO 控制寄存器	SPITXFIFOCR
0x02C	SPIRXFIFOCR	接收 FIFO 控制寄存器	SPIRXFIFOCR
0x030	SPICSNCR	片选选择寄存器	SPICSNCR

21.7 SPI 寄存器描述

SPICR0

SPICR0 为控制寄存器 0。

Offset Address: 0x000 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:8]	RW	scr	串行时钟率，取值范围 0～255。scr 的值用来产生 SPI 发送和接收的速率，公式为 F_{clk_spi}/(cpsdvsr x (1+scr))。	0x00

Bits	Access	Name	Description	Reset
			cpsdvsr 是一个 2～254 之间的偶数，由寄存器 SPICPSR 配置。	
[7]	RW	sph	SPICLKOUT 相位，具体含义请参见“21.4 接口时序”的 SPI 帧格式。	0x0
[6]	RW	spo	SPICLKOUT 极性，具体含义请参见“21.4 接口时序”的 SPI 帧格式。	0x0
[5:4]	RW	frf	帧格式选择。 00：Motorola SPI 帧格式； 01：TI 同步串行帧格式； 10：National Microwire 帧格式； 11：保留。	0x0
[3:0]	RW	dss	设置数据位宽。 0x3：4bit； 0x4：5bit； 0x5：6bit； 0x6：7bit； 0x7：8bit； 0x8：9bit； 0x9：10bit； 0xA：11bit； 0xB：12bit； 0xC：13bit； 0xD：14bit； 0xE：15bit； 0xF：16bit； 其他：保留。	0x0

SPICR1

SPICR1 为控制寄存器 1。

Offset Address: 0x004 Total Reset Value: 0x7F00

Bits	Access	Name	Description	Reset
[15]	RW	waiten	等待使能，当 SPICR1 寄存器的 sm 配置为 Master 模式且 SPICR0 寄存器的 frf 配置为 Microwire 帧格式时有效。 0：不使能； 1：使能。	0x0
[14:8]	RW	waitval	Microwire 帧格式时，写和读之间的等待拍数。当 SPICR1 寄存器的 sm 配置为 Master 模式且 waiten 为 1 并且帧格式为 National Microwire 时有效。 0x00：等待周期为 1 个工作时钟 clk_spi； 0x01：等待周期为 2 个工作时钟 clk_spi； ... 0x7F：等待周期为 128 个工作时钟 clk_spi。	0x7F
[7:5]	-	reserved	保留。	0x0
[4]	RW	bitend	数据正反序控制位。 0：数据顺序为 MSB 到 LSB； 1：数据顺序为 LSB 到 MSB。	0x0
[3]	-	reserved	保留。	0x0
[2]	RW	ms	设置 Master 或者 Slave 模式，此位只能在 SPI 不使能时改变。 0：Master 模式； 1：Slave 模式。	0x0
[1]	RW	sse	设置 SPI 使能。 0：不使能； 1：使能。	0x0

Bits	Access	Name	Description	Reset
[0]	RW	lbm	设置环回模式。 0：正常的串行接口操作使能； 1：发送串行移位寄存器的输出在内部连接到接收串行移位寄存器的输入上。	0x0

SPIDR

SPIDR 为数据寄存器。

Offset Address: 0x008 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:0]	RW	data	发送/接收 FIFO。 读：从接收 FIFO 读出数据； 写：向发送 FIFO 写入数据。 如果数据 bit 数少于 16 则必须右对齐。SPI 控制器发送时将忽略高位未使用的 bit 位，SPI 控制器接收时则自动将数据右对齐。	0x0000

SPISR

SPISR 为状态寄存器。

Offset Address: 0x00C Total Reset Value: 0x0003

Bits	Access	Name	Description	Reset
[15:5]	-	reserved	保留。	0x000
[4]	RO	bsy	SPI 忙标记。 0：空闲； 1：忙。	0x0
[3]	RO	rff	接收 FIFO 是否已满。 0：未满； 1：已满。	0x0

Bits	Access	Name	Description	Reset
[2]	RO	rne	接收 FIFO 是否未空。 0：已空; 1：未空。	0x0
[1]	RO	tnf	发送 FIFO 是否未满。 0：已满; 1：未满。	0x1
[0]	RO	tfe	发送 FIFO 是否已空。 0：未空; 1：已空。	0x1

SPICPSR

SPICPSR 为时钟分频寄存器。

Offset Address: 0x010 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:8]	-	reserved	保留。	0x00
[7:0]	RW	cpsdvsr	时钟分频因子。此值必须是 2 ~ 254 之间的偶数，取决于输入时钟 SPI_CLK 的频率。最低位读作“0”。	0x00

SPIIMSC

SPIIMSC 为中断屏蔽寄存器。

Offset Address: 0x014 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:4]	-	reserved	保留。	0x000
[3]	RW	txim	发送 FIFO 中断屏蔽。 0：发送 FIFO 中断被屏蔽，不使能;	0x0

Bits	Access	Name	Description	Reset
			1：发送 FIFO 中断未被屏蔽，使能。	
[2]	RW	rxim	接收 FIFO 中断屏蔽。 0：接收 FIFO 中断被屏蔽，不使能； 1：接收 FIFO 中断未被屏蔽，使能。	0x0
[1]	RW	rtim	接收超时中断。 0：接收超时中断屏蔽，不使能； 1：接收超时中断不屏蔽，使能。	0x0
[0]	RW	rorim	接收溢出中断屏蔽。 0：接收 FIFO 溢出中断屏蔽，不使能； 1：接收 FIFO 溢出中断不屏蔽，使能。	0x0

SPIRIS

SPIRIS 为原始中断状态寄存器。值“0”表示无中断，值“1”表示有中断。

Offset Address: 0x018 Total Reset Value: 0x0008

Bits	Access	Name	Description	Reset
[15:4]	-	reserved	保留。	0x000
[3]	RO	txris	发送 FIFO 中断的原始中断状态。 0：大于 SPITXFIFOCR[txintsize]中配置的中断水线； 1：等于或小于 SPITXFIFOCR[txintsize]中配置的中断水线。	0x1
[2]	RO	rxris	接收 FIFO 中断的原始中断状态。 0：小于 SPIRXFIFOCR[rxintsize]中配置的中断水线； 1：等于或大于 SPIRXFIFOCR[rxintsize]中配置的中断水线。	0x0
[1]	RO	rtris	接收超时中断的原始中断状态。当接收 FIFO 非空且 SPI_CLK 未发生翻转，超过	0x0

Bits	Access	Name	Description	Reset
			64 个 SPI 工作时钟周期时，此位域置 1。对 SPIICR[rtic]写 1，可清除此位域。	
[0]	RO	rorris	接收溢出中断的原始中断状态。当接收 FIFO 已满且额外的数据被接收时，发生溢出中断，此位域置 1。对 SPIICR[roric]写 1，可清除此位域。	0x0

SPIMIS

SPIMIS 为屏蔽后中断状态寄存器。值“0”表示无中断，值“1”表示有中断。

Offset Address: 0x01C Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:4]	-	reserved	保留。	0x000
[3]	RO	txmis	发送 FIFO 中断屏蔽后的状态。	0x0
[2]	RO	rxmis	接收 FIFO 中断屏蔽后的状态。	0x0
[1]	RO	rtmis	接收超时中断屏蔽后的状态。	0x0
[0]	RO	rormis	接收溢出中断屏蔽后的状态。	0x0

SPIICR

SPIICR 为中断清除寄存器。写“1”清除中断，写“0”无影响。

Offset Address: 0x020 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:2]	-	reserved	保留。	0x0000
[1]	RW1C	rtic	清除接收超时中断。	0x0
[0]	RW1C	roric	清除接收溢出中断。	0x0

SPIDMAR

SPIDMAR 为 DMA 控制寄存器。

Offset Address: 0x024 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:3]	-	reserved	保留位。	0x0000
[2]	RW	rxdmalsreqe	接收 FIFO DMA 外设流控（SPI 流控）使能寄存器。当 RX FIFO 非空且 SPI 超时，LSREQ 硬件请求线置 1。 0：禁止； 1：使能。	0x0
[1]	RW	txdmae	发送 FIFO DMA 请求使能寄存器。 0：禁止； 1：使能。	0x0
[0]	RW	rxdmae	接收 FIFO DMA 请求使能寄存器。 0：禁止； 1：使能。	0x0

SPITXFIFOCR

SPITXFIFOCR 为发送 FIFO 控制寄存器。

Offset Address: 0x028 Total Reset Value: 0x0001

Bits	Access	Name	Description	Reset
[15:4]	-	reserved	保留。	0x000
[3:0]	RW	txintsize	配置发送 FIFO 的水线。当发送 FIFO 中的数据个数满足条件时，产生发送 FIFO 中断（TXRIS）。如果使能了发送 FIFO DMA 功能，同时产生发送 FIFO DMA burst 传输请求。 0x0：等于 0； 0x1：小于或等于 1；	0x1

Bits	Access	Name	Description	Reset
			0x2：小于或等于 2； 0x3：小于或等于 3； ... 0xF：小于或等于 15。 如果 DMA 控制器配置为搬移 FIFO 的数据，那么发送 FIFO 的水线需要小于 DMA 控制器的 Burst 长度。	

SPIRXFIFOCR

SPIRXFIFOCR 为接收 FIFO 控制寄存器。

Offset Address: 0x02C Total Reset Value: 0x0001

Bits	Access	Name	Description	Reset
[15:4]	-	reserved	保留。	0x000
[3:0]	RW	rxintsize	配置接收 FIFO 的水线。当接收 FIFO 中的数据个数满足条件时，产生接收 FIFO 中断（RXRIS）。如果使能了接收 FIFO DMA 功能，产生接收 FIFO DMA burst 传输请求。 0x0：大于或等于 0； 0x1：大于或等于 1； 0x2：大于或等于 2； 0x3：大于或等于 3； ... 0xF：大于或等于 15。 如果 DMA 控制器配置为搬移 FIFO 的数据，那么接收 FIFO 的水线需要大于 DMA 控制器的 Burst 长度。	0x1

SPICSNCR

SPICSNCR 为 SPI 片选选择寄存器。

Offset Address: 0x030 Total Reset Value: 0x0000

Bits	Access	Name	Description	Reset
[15:6]	-	reserved	保留。	0x000
[5:4]	RW	spi_csn_mode	配置 SPI 片选产生的方式。 00：自动。片选信号由 MCU 逻辑根据所选时序自动产生； 01：altasens 模式。当采用 Motorola SPI 帧格式时，片选信号由 SPI 使能信号控制，使能后，片选拉低，否则片选拉高； 10：受软件控制，置 0； 11：受软件控制，置 1。	0x0
[3:2]	-	reserved	保留。	0x0
[1:0]	RW	spi_csn_sel	配置选择 SPI 片选。 00：片选 0； 01：片选 1； 其他：保留。	0x0

22 CAN 控制器

22.1 概述

控制器域网 CAN（Controller Area Network）是国际上应用最广泛的现场总线之一，具有高性能、高可靠性、高实时性的特点。MCU 中 CAN 控制器支持 CAN Protocol Version 2.0 part A 和 B。

22.2 特性

CAN 总线控制器模块有以下特点：

- CAN 总线控制器的工作时钟可选 clk_xtal 或者 clk_hosc。
- 支持标准技术规范 CAN 2.0A 和 CAN 2.0B。
- 支持多设备时的总线仲裁。
- 支持传输速率可编程，最高可达 1Mbps。
- 支持 32 个报文对象，每个报文对象均可编程。
- 支持主动错误和被动错误的自我判定以及故障节点的隔离。
- 支持错误的自我修复。
- 支持自动重传模式。
- 支持报文接收过滤功能。
- 支持中断屏蔽。
- 支持连续报文接收。
- 支持测试模式，包括 Loop-back、Silent 模式以及两者同时开启。
- 支持 CAN 总线物理层测试，软件强制 Tx 输出，和回读 Rx 输入。

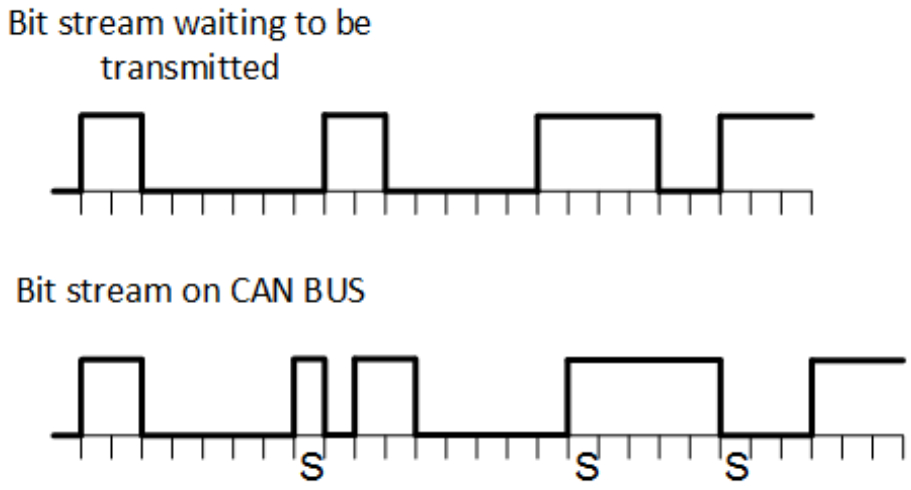
- 支持 Basic 模式。

22.3 功能描述

22.3.1 编码方式

CAN 总线采用单极性不归零编码（NRZ，None-Return-to-Zero），相比曼彻斯特码具有更好的 EMC 性能。当 CAN 控制器发送连续 5 个相同电平的位就会插入一个相反电平的填充位，CAN 控制器接收时会删除这个填充位，然而在一些固定的位场中不使用位填充，如数据帧或远程帧的 CRC 界定符、应答场和帧结束，错误帧和过载帧均不通过位填充的方法进行编码。位填充示意图如图 22-1 所示。

图22-1 位填充示意图

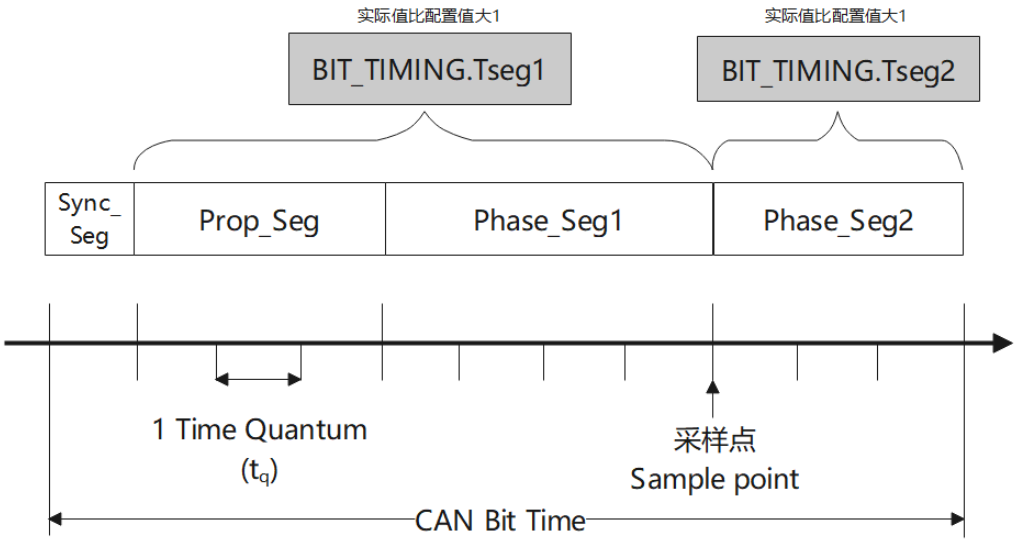


22.3.2 波特率

协议规定了 1 个 CAN bit 由 Sync_Seg、Prop_Seg、Phase_Seg1 和 Phase_Seg2 四部分组成，每部分的最小时间单位为 1 个 time quantum (t_q)。其中协议规定了 Sync_Seg 固定为 1 个 t_q，无需配置。寄存器 BIT_TIMING[Tseg1]配置了 Prog_Seg + Phase_Seg1，[Tseg2]配置了 Phase_Seg2。注意 Tseg1 和 Tseg2 的实际生效值比配置值大 1。

$$波特率=\frac{1}{1个CAN\,bit的时间}bps$$

图22-2 1 个 CAN bit 示意图



CAN 波特率常用配置如表 22-1 所示。

表22-1 CAN 波特率常用配置

时钟频率	波特率	采样点百分比	BIT_TIMING 配置值
25MHz	1000kbps	60%	0x1104
25MHz	500kbps	60%	0x3404
25MHz	250kbps	60%	0x7a04
25MHz	250kbps	60%	0x3409
25MHz	125kbps	60%	0x7a09
25MHz	125kbps	60%	0x3413
25MHz	125kbps	62.5%	0x2318
25MHz	100kbps	60%	0x3418
25MHz	50kbps	60%	0x7a18
25MHz	50kbps	60%	0x3431

22.3.3 总线速率与传输距离的关系

CAN 总线上任意两个节点之间的最大传输距离与其位速率关系如表 22-2 所示。

表22-2 CAN 总线位速率与传输距离对应表

传输数率（Kbps）	1000	500	250	125	100	50	20	10	5
估算最大距离（m）	40	130	270	530	620	1300	3300	6700	10000

22.3.4 帧类型

CAN 总线控制器支持 4 种类型的帧。

- 数据帧用于发送节点向其他节点发送数据。
- 远程帧用于接收节点向发送节点请求数据，当发送节点成功接收远程帧后，将向接收节点返回一个具有相同仲裁场的数据帧。
- 错误帧用于控制器检测到错误时向总线通知错误。
- 过载帧用于接收节点通知发送节点其尚未准备好接收数据，需要在先行和后续的数据帧（或远程帧）之间提供一个附加的延时。

说明

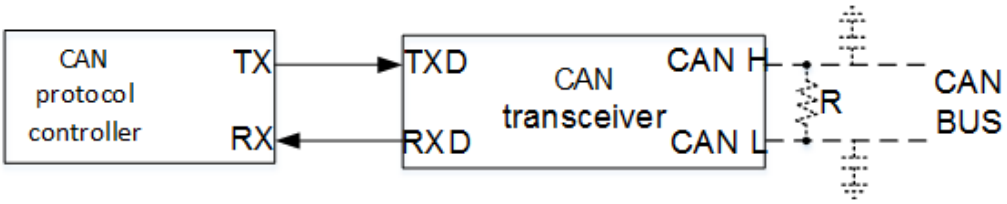
数据帧和和远程帧都分为标准格式（报文 ID 宽度 11bit）和扩展格式（报文 ID 宽度 29bit）。

另外，无论前面的帧是什么类型（数据帧、远程帧、错误帧、过载帧），数据帧或远程帧与其前面帧的隔离是通过帧间间隙实现的，而过载帧与错误帧之前没有帧间间隙，多个过载帧之间也不是由帧间间隙隔离的。

22.3.5 典型应用

CAN 总线控制器外接收发器时的典型应用框图如图 22-3 所示，具体请参考对接收发器的用户手册。

图22-3 CAN 总线控制器对接外置收发器时的应用

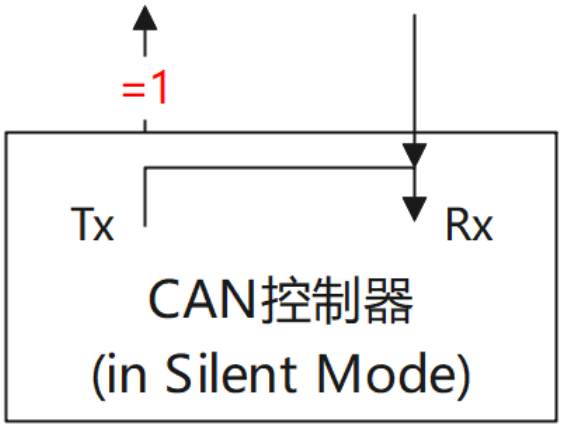


22.3.6 测试模式

CAN 控制器支持测试模式，分为 Silent 模式，Loop-back 模式以及这两个模式同时开启。软件需要先配置 CAN_CONTROL[test]，使能测试模式，才能配置 CAN_TEST 寄存器，从而开启 Silent 模式，Loop-back 模式或者同时开启这两个模式。

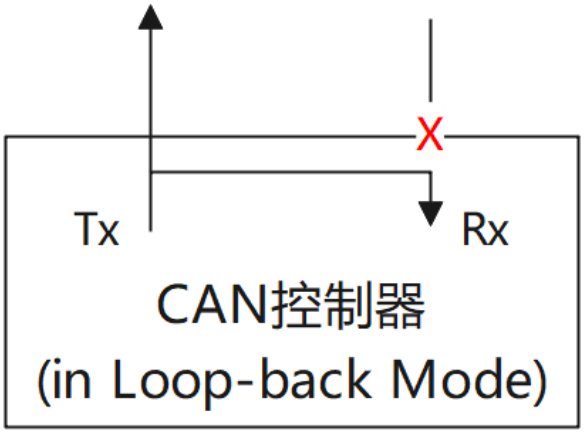
Silent 模式用于分析 CAN 总线流量，不影响当前 CAN 网络，CAN 控制器向外发送静默，也就是保持隐形电平（输出 1），不能发起传输。CAN 控制器将协议规定的显性电平（帧 ACK，错误帧，过载帧）内部环回到 Rx 侧。软件可通过配置 CAN_TEST[Silent]，使能该模式。图 22-4 是 Silent 模式下的示意图。

图22-4 CAN 控制器在 Silent 模式下的示意图



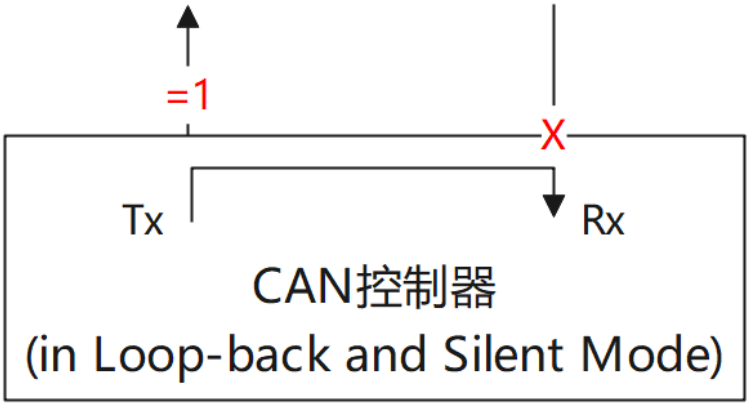
Loop-back 模式用于功能自测试，CAN 控制器可以向外发送帧，同时 Tx 内部环回到 Rx 接收自己发出的帧，忽略帧 ACK，忽略总线上的帧。软件可通过配置 CAN_TEST[lback]，使能该模式。图 22-5 是 Loop-back 模式下的示意图。

图22-5 CAN 控制器在 Loop-back 模式下的示意图



Silent 模式和 Loop-back 模式同时开启用于在线自测试，CAN 节点从总线脱离，既不向 CAN 总线发送帧，也不接收 CAN 总线上的帧。控制器处于自发自收的状态。软件可通过配置 CAN_TEST[lback]和[silent]使能。

图22-6 CAN 控制器在 Silent 模式和 Loop-back 模式同时开启时的示意图



CAN 控制器支持软件强制 Tx 输出和实时回读 Rx 输入。软件可通过配置 CAN_TEST[Tx]置位或者清零来强制 Tx 为 1 或者 0。这可用于检查 CAN 总线的物理层是否正常工作。当 Tx 输出等于 Rx 回读时，表示 CAN 总线的物理层正常工作。

22.3.7 Basic 模式

CAN 控制器支持 Basic 模式。此时 32 个报文对象均被关闭，所有跟报文对象相关的寄存器均无效。IF1 报文对象接口寄存器组变为 Tx 报文缓存，IF2 报文对象接口寄存器组变为 Rx 报文缓存。IF1 和 IF2 报文对象接口寄存器组均是地址空间直接映射的，

即 CPU 能直接通过地址访问 Tx 报文和 Rx 报文。软件可通过配置 CAN_TEST[basic]使能 Basic 模式。

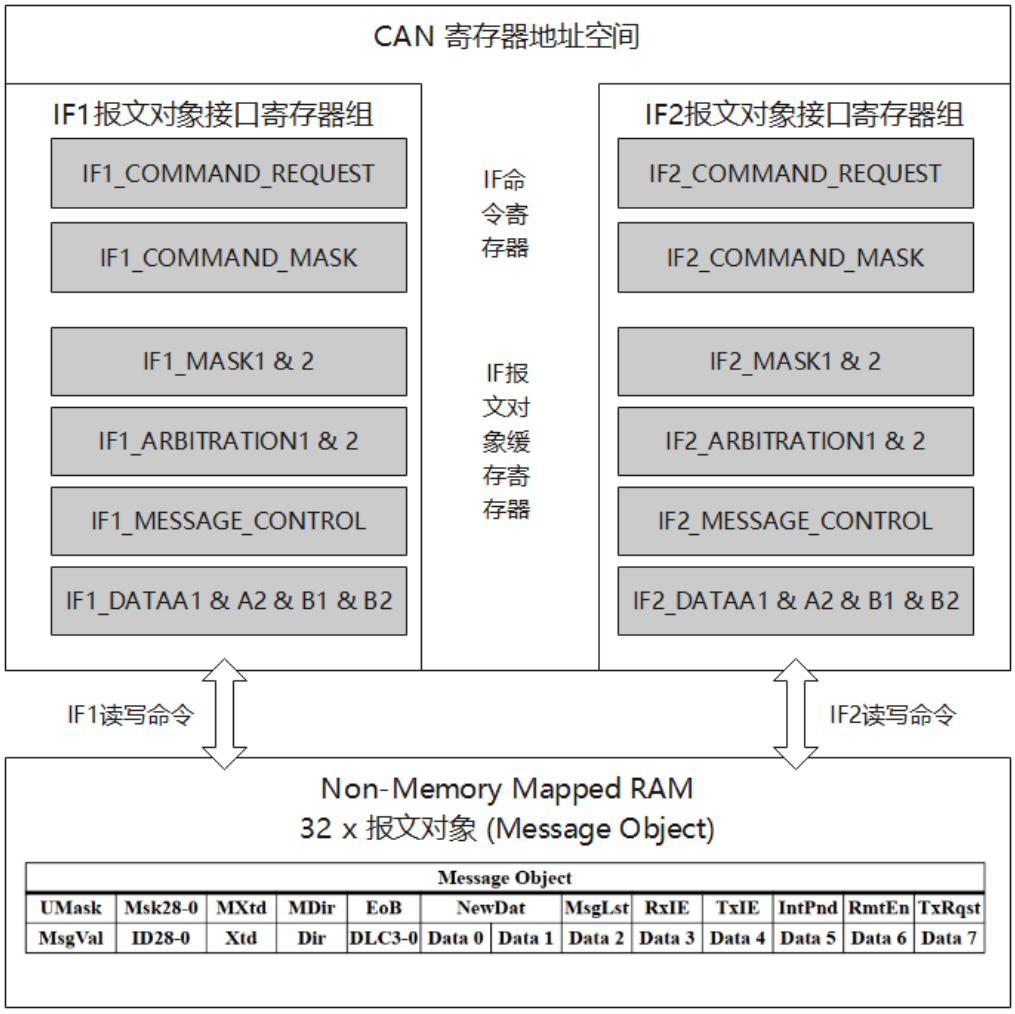
在 Basic 模式下，报文接收过滤功能被关闭，所有报文均被接收保存在 Rx 报文缓存中。

在 Basic 模式下，Tx 报文需要软件配置 IF1_COMMAND_REQUEST[busy]置位来请求发送。

22.3.8 报文对象和 IF 报文对象接口寄存器组

CAN 控制器最多支持 32 个报文对象。报文对象分为 Tx 报文对象和 Rx 报文对象，报文对象的 Dir 位配置了该报文对象是 Tx 方向还是 Rx 方向。Tx 报文对象可以发送数据帧，也可以接收远程帧并发送数据帧。Rx 报文对象可以接收数据帧，也可以发送远程帧。

图22-7 报文对象数据结构和 IF 报文对象接口寄存器组示意图



报文对象存储在非地址空间映射的 RAM 中，CPU/DMA 无法直接读写报文对象，必须通过编程配置 IF1 或者 IF2 报文对象接口寄存器组发出读写命令来读写报文对象。

IF 报文对象接口寄存器组总共有 2 组，分别为 IF1 和 IF2。下面以 IF1 为例介绍 IF 报文对象接口寄存器组。IF1 分为 2 部分。一部分是 IF1 命令寄存器：IF1_COMMAND_REQUEST、IF1_COMMAND_MASK。另一部分是 IF1 报文对象缓存寄存器：IF1_MASK1&2、IF1_ARBITRATION1&2、IF1_DATAA1&A2&B1&B2。

IF 写命令：将 IF 报文对象缓存写入报文对象中。

IF 读命令：将报文对象回读到 IF 报文对象缓存中。

IF 命令掩码可指定当前命令操作的对象，即指定相应的 IF 报文对象缓存寄存器的位域。

IF1 和 IF2 功能等价，下面以 IF1 为例，用户须配置 IF1_COMMAND_MASK，IF1_MASK1&2，IF1_ARBITRATION1&2，IF1_MESSAGE_CONTROL，IF1_DATA，最后将预期访问的报文对象编号写入 IF1_COMMAND_REQUEST。此时 CAN 控制器结合 IF1 命令（IF1_COMMAND_REQUEST）和 IF1 命令掩码（IF1_COMMAND_MASK），执行读写命令，读写相应的报文对象。用户可回读 IF1_COMMAND_REQUEST[busy]查询命令是否已完成。

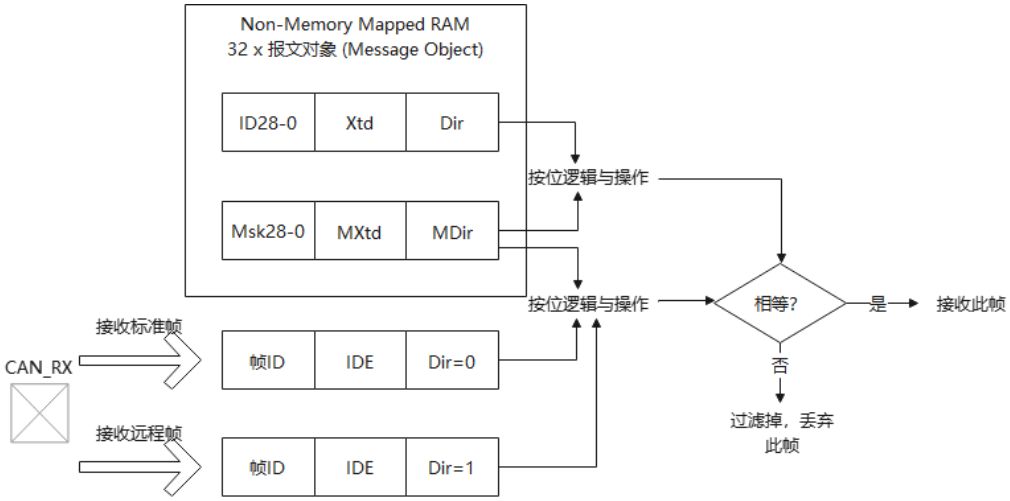
须知

IF1 和 IF2 的功能等价。

22.3.9 报文接收过滤功能

在配置 Tx 报文对象和 Rx 报文对象时，可以使用 CAN 控制器的接收过滤功能。接收过滤功能受使能控制（UMask）。报文对象的 mxtd、MDir、Msk 为报文掩码。Msk 掩码指明了报文对象 ID 的哪些 bit 起作用。具体的过滤原则是，Msk 掩码和总线上帧 ID 做按位与操作，Msk 掩码和报文对象 ID 做按位与操作，两个结果一致，那么总线上的帧被接收。如图 22-8 所示，CAN 控制器自动完成接收过滤，通过过滤后的帧保存在匹配的报文对象中，未通过过滤的帧被丢弃。

图22-8 报文接收过滤功能图示



对于 Tx 报文对象（Dir=1），使能远程帧（RmtEN=1），无论使不使能接收过滤功能（UMask=0 或 1）：Tx 报文对象在成功接收到匹配的远程帧后，Txrqst 自动置 1，并发送数据帧。

对于 Tx 报文对象（Dir=1），不使能远程帧（RmtEN=0），不使能接收过滤功能（UMask=0）：Tx 报文对象在成功接收到匹配的远程帧后，该远程帧被丢弃。

对于 Tx 报文对象（Dir=1），不使能远程帧（RmtEN=0），使能接收过滤功能（UMask=1）：Tx 报文对象在成功接收到匹配的远程帧后，该远程帧的仲裁段和控制段被保存在 Tx 报文对象中，报文对象的 NewDat 位置 1，DATA 数据段不更新。远程帧被当做数据帧接收。

对于 Rx 报文对象（Dir=0），使能接收过滤功能（UMask=1）：Rx 报文对象根据接收掩码（mxtd，MDir，Msk28-0）接收符合的报文。掩码 mxtd=1，表示报文对象检查数据帧 IDE 字段，根据报文对象的 Xtd，只接收标准帧或者拓展帧。

22.4 典型配置

22.4.1 初始化所有报文对象

初始化可以将 32 个报文对象配置为默认值，步骤如下：

步骤 1 向 CAN_CONTROL[init]写 0x1，开始初始化配置。

步骤 2 确认 IF1 报文对象缓存寄存器（IF2_MASK1&2，IF2_ARBITRATION1&2，IF1_MESSAGE_CONTROL，IF1_DATA）均为默认值。

步骤 3 向 IF1_COMMAND_MASK 寄存器写 0xF3，准备配置报文对象。

步骤 4 向 IF1_COMMAND_REQUEST 寄存器写 0x1，执行命令，将 IF1 报文对象缓存寄存器写入报文对象 1。

步骤 5 读 IF1_COMMAND_REQUEST 寄存器，busy 位为 1 表示命令仍在执行，只有当 Busy 位为 0 时才能初始化下一个报文对象。重复步骤 3 到步骤 5 直至全部 32 个报文对象完成初始化。

步骤 6 向 CAN_CONTROL 寄存器写 0x41，允许修改 BIT_TIMING 寄存器。

步骤 7 根据网络传输需求，配置 BIT_TIMING 寄存器以满足总线的传输速率。

步骤 8 根据中断需求，配置 CAN_CONTROL[eie]，[sie]和[ie]。

步骤 9 向 CAN_CONTROL[init]写 0，完成初始化配置。

----结束

须知

- 在步骤 3 和步骤 4 中，初始化报文对象时应当写 0x2，初始化报文对象 3 时应当写 0x3，以此类推。
- 对于低传输速率的配置，需要在步骤 6 之后配置 BRP_EXTENSION[brpe]。波特率预分频拓展到最大，其中高位由 BRP_EXTENSION[brpe]组成，低位由 BIT_TIMING[brp]组成。
- 完成初始化配置后，至少要等 11 个位时间后才能开始帧传输。

22.4.2 传输速率配置

通过配置 BIT_TIMING 寄存器可以设置 CAN 总线控制器工作的传输速率，详细描述请参见 22.3.2 波特率章节。CAN 控制器的工作时钟可选 clk_xtal 或者 clk_hosc，下面以 24MHz 为例，示例如下：

步骤 1 CAN 总线控制器的工作频率 f_{clk_can} 为 24 MHz，t_{clk_can}=41.67ns，假设 CAN 总线网络的传输速率为 500Kbps，那么位时间 $t_{bit\text{-}time}$ 为 2000ns。

步骤 2 配置 BIT_TIMING[brp]为 2，即 t_q 为工作时钟 t_{clk_can} 的 3 倍，$t_q = 3 \times t_{clk_can}$ =125ns，故位时间 $t_{bit\text{-}time}$=16 × t_q。

步骤 3 传输段时间 t_{prop} 由系统的延迟决定，其长度是双绞线物理延迟 $t_{delay\text{-}of\text{-}bus\text{-}line}$ 与总线收发器延迟 $t_{delay\text{-}of\text{-}bus\text{-}driver}$ 和 $t_{delay\text{-}of\text{-}receiver\text{-}circuit}$ 总和的两倍，即：

$t_{prop} = (t_{delay\text{-}of\text{-}bus\text{-}driver} + t_{delay\text{-}of\text{-}receiver\text{-}circuit} + t_{delay\text{-}of\text{-}bus\text{-}line}) \times 2 \approx$ multiple of t_q

假设 $t_{delay\text{-}of\text{-}bus\text{-}driver}$ 约为 50ns，$t_{delay\text{-}of\text{-}receiver\text{-}circuit}$ 约为 30ns，$t_{delay\text{-}of\text{-}bus\text{-}line}$ 约为 170ns，那么 $t_{prop} \approx$ 500ns = 4× t_q。

说明

- 若不是 t_q 的整数倍关系，则整数位进 1。比如 t_{prop}≈5.1 × tq，则取 t_{prop}=6 × t_q。

步骤 4 可以使用如下方式推导 t_{seg1} 和 t_{seg2}。除去 t_{sync}（固定为 1 个 t_q）和 t_{prop}，位时间还剩 $t_{seg2}+t_{seg1}$=11 × t_q。由于是奇数倍，那么 $t_{seg2}=t_{seg1}+1$，即 t_{seg1} 为 5 × t_q，t_{seg2} 为 6 × t_q，此时采样点百分比为 68.75%。如果剩余量为 t_q 的偶数倍，则 $t_{seg2}=t_{seg1}$。

也可以由实际应用所需采样点百分比（$采样点百分比=1-\frac{t_{seg1}}{t_{bit\text{-}time}}\times 100\%$）反向推导出 t_{seg1} 和 t_{seg2}。举个例子，以采样点百分比 80%为例子，t_{seg1}=(1-80%) × $t_{bit\text{-}time}$=0.2×16× t_q=3.2×t_q≈3× t_q, t_{seg2}=11 × t_q-3× t_q=8× t_q，此时实际采样点百分比为 81.25%。

步骤 5 同步跳转宽度 t_{SJW}=min(4，t_{seg1}) = 4 × t_q。

步骤 6 计算 TSEG1= t_{seg1}+ t_{prop}-1=5+4-1=8；计算 TSEG2= t_{seg2}-1=6-1=5；计算 SJW= t_{SJW}-1=4-1=3，计算 BRP=2，因此 BIT_TIMING 配置 0x58C3。

步骤 7 若步骤 6 的计算结果超过寄存器有效值，返回步骤 2 更改 brp 值后重新开始。

步骤 8 若步骤 7 的计算结果超过寄存器有效值，请在步骤 2 的基础上配置 BRP_EXTENSION[brpe]后重新开始。

----结束

须知

- 建议位时间是 t_q 的整数倍。
- CAN 总线网络的传输速率必须是固定且唯一的。

22.4.3 配置 Tx 报文对象

Tx 报文对象可以发送数据帧，可以接收远程帧并发送相应的数据帧。发送步骤如下：

步骤 1 根据发送帧 ID 的不同，向 IF1_ARBITRATION1 和 IF1_ARBITRATION2 寄存器写入相应的值。具体如下：

如果发送标准数据帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x5，bit[12:2]写报文 ID，bit[1:0]写 0x0，并向 IF1_ARBITRATION1 寄存器写 0x0。

如果发送扩展数据帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x7，bit[12:0]写报文 ID28~16 位，并 IF1_ARBITRATION1 寄存器写报文 ID15~0 位。

如果接收标准远程帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x5，bit[12:2]写报文 ID，bit[1:0]写 0x0，并向 IF1_ARBITRATION1 寄存器写 0x0。

如果接收扩展远程帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x7，bit[12:0]写报文 ID28~16 位，并向 IF1_ARBITRATION1 寄存器写报文 ID15~0 位。

步骤 2 向 IF1_MESSAGE_CONTROL[NewDat]、[txrqst]、[eob]置 1，[dlc]置发送数据帧的长度，[txie]和[rxie]根据中断需求设置，其余位置 0。

如果接收远程帧，则向 IF1_MESSAGE_CONTROL[newdat]置 1，[rmten]置 1。可以向 IF1_MESSAGE_CONTROL[umask]置 1 开启接收过滤功能，根据期望接收到的远程帧的仲裁段，向 IF2_MASK2 和 IF2_MASK1 寄存器写入掩码。

如果不接受远程帧，则向 IF1_MESSAGE_CONTROL[rmten]置 0，[umask]置 0。

步骤 3 依次向 IF1_DATAA1、IF1_DATAA2、IF1_DATAB1、IF1_DATAB2 写入待传输数据。

步骤 4 向 IF2_COMMAND_MASK 寄存器写 0xF3。

步骤 5 向 IF2_COMMAND_REQUEST[message number]写入一个空闲的报文对象的序号，之后 CAN 控制器执行写命令，写入指定的报文对象。

----结束

须知

- 如果发送标准数据帧或扩展数据帧，步骤 2 中 rxie 位的配置可以忽略。
- 当 Tx 报文对象 rmten 位为 1 时，成功接收远程帧后，控制器将 Tx 报文对象的 txrqst 自动置 1，自动发出一个数据帧。
- 连续数据帧的发送应当在当前帧发送完成后再配置下一帧。

22.4.4 配置 Rx 报文对象

Rx 报文对象可以发送远程帧，可以接收数据帧。

接收单一 ID 报文，即开启报文接收过滤功能，只接收单一 ID 的报文，过滤掉其他 ID 的报文，步骤如下：

步骤 1 根据接收帧 ID 的不同，在接收该帧之前应向 IF1_ARBITRATION1 和 IF1_ARBITRATION2 寄存器写入相应的值。注意报文 ID 应与接收的报文相同。具体如下：

如果接收标准数据帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x4，bit[12:2]写报文 ID，bit[1:0]写 0x0，并向 IF1_ARBITRATION1 寄存器写 0x0。

如果接收扩展数据帧，则向 IF1_ARBITRATION2bit[15:13]写 0x6，bit[12:0]写报文 ID28~16 位，并向 IF1_ARBITRATION1 寄存器写报文 ID15~0 位。

如果发送标准远程帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x4，bit[12:2]写报文 ID，bit[1:0]写 0x0，并向 IF1_ARBITRATION1 寄存器写 0x0。

如果发送扩展远程帧，则向 IF1_ARBITRATION2 bit[15:13]写 0x6，bit[12:0]写报文 ID28~16 位，并向 IF1_ARBITRATION1 寄存器写报文 ID15~0 位。

步骤 2 向 IF2_MASK2 寄存器写 0xDFFF。

步骤 3 向 IF2_MASK1 寄存器写 0xFFFF。掩码全有效，表示报文接收过滤功能只接收单一 ID 报文，即 Rx 报文对象相同 ID 的报文，其他 ID 的报文均被过滤掉。

如果需要接收某些 ID 的报文（非单一报文）。举两个例子。

例子 1：只接收报文 ID=0x7F2 的标准数据帧和报文 ID=0x7F3 的标准数据帧，那么 IF1_ARBITRATION2 bit[12:2]写 0x7F2，IF2_MASK2[mxtd]写 1，[mdir]写 1，[msk]的 28~16 写 0x7FE（忽略 ID 的最后一个 bit）。

例子 2：只接收报文 ID=0x7FFFFF2 的扩展数据帧和报文 ID=0x7FFFFF3 拓展数据帧，那么 IF1_ARBITRATION2 bit[12:0]写 0x7FF，IF1_ARBITRATION1 写 0xFFF2，IF2_MASK2[mxtd]写 1，[mdir]写 1，[Msk]的 28~16 写 0x1FFF，IF1_MASK1 写 0xFFFE（忽略 ID 的最后一个 bit）

步骤 4 向 IF1_MESSAGE_CONTROL[eob]置 1，[TxIE]和[RxIE]根据中断需求设置，其余位置 0。

如果发送远程帧，则向[txrqst]置 1。

步骤 5 向 IF2_COMMAND_MASK 寄存器写 0xF3。

步骤 6 向 IF2_COMMAND_REQUEST[message number]写一个空闲的报文对象的序号，之后 CAN 控制执行写命令，写入指定的报文对象。

----结束

当报文对象成功接收数据帧时，需要配置 IF 报文对象接口寄存器组，执行读命令，命令完成后通过 IF1_ARBITRATION1、IF1_ARBITRATION2、IF1_DATAA1、IF1_DATAA2、IF1_DATAB1、IF1_DATAB2 寄存器查看接收的数据帧。

步骤 1 向 IF2_COMMAND_MASK 寄存器写 0x7F。

步骤 2 向 IF2_COMMAND_REQUEST[message number]写上述接收数据帧的报文对象序号，执行读命令，回读指定的报文对象到 IF2 报文对象接口寄存器组。

步骤 3 只有当 IF2_COMMAND_REQUEST[busy]为 0 时才能读取 IF1_DATAA1、IF1_DATAA2、IF1_DATAB1、IF1_DATAB2 寄存器内部的数据。

----结束

须知

- IF1 和 IF2 报文对象接口寄存器组等价，上述步骤中 IF1 和 IF2 均可，但是在同一个命令中不可混合使用。
- 如果只接收标准数据帧或扩展数据帧，步骤 4 中 txie 位的配置可以忽略。
- 当 Rx 报文对象发出了远程帧，接收节点返回了数据帧且被 Rx 报文对象接收完成。返回的数据帧被保存在 Rx 报文对象中，软件可以通过 IF 报文对象接口寄存器组发出读命令，回读 Rx 报文对象到 IF 报文对象接口寄存器组中，通过 IF1_ARBITRATION1、IF1_ARBITRATION2、IF1_DATAA1、IF1_DATAA2、IF1_DATAB1、IF1_DATAB2 寄存器查看返回的数据帧。

22.4.5 中断查询与处理

CAN 总线控制器的中断信号可以分为两类：

- 状态中断，中断号为 0x8000，表示 CAN 总线控制器发送或接收成功一个报文或检测到一个总线错误或发出错误警告或处于总线关闭状态。
- 报文对象中断，中断号为 0x0001~0x0020，表示对应报文对象为中断源。

其中，状态中断的优先级高于报文对象中断。

- 当 CAN_CONTROL bit[1]使能后，CAN 总线控制器能够产生中断信号。
- 当 CAN_CONTROL bit[2]使能后，CAN_STATUS[rxok]、[txok]以及[lec]将触发状态中断。
- 当 CAN_CONTROL bit[3]使能后，CAN_STATUS[boff]和[ewarn]将引发状态中断。

可以通过查询 CAN_STATUS 寄存器了解详细状态。

在传输过程中，在配置报文对象时，若软件将 IF1_MESSAGE_CONTROL[txie]或[rxie]被置 1，那么在发送或者接收成功后，CAN 控制器会将 intpnd 置为 1，这将导致报文对象中断的产生。

须知

CAN_STATUS[epass]的改变以及写入[rxok]、[txok]以及[lec]不会引起状态中断的产生。

如果中断信号有效且中断类型为状态中断。软件执行如下步骤清除状态中断。

22.4.6 报文接收过滤

报文接收过滤功能详细描述请参见 22.3.9 报文接收过滤功能章节。

配置步骤如下：

步骤 1 在数据帧或远程帧的接收步骤 2，根据实际屏蔽需求配置 IF1_MASK2[mxtd]、[mdir]和[msk]，置 1 表示掩码对应的 Xtd、Dir、Msk 位用于报文过滤，置 0 表示不用于报文过滤。

步骤 2 在数据帧或远程帧的接收步骤 3，根据实际屏蔽需求配置 IF1_MASK1[msk]，置 1 表示该位对应的 Arbitration（仲裁）位域用于报文过滤，置 0 表示不用于报文过滤。标准帧格式可忽略此步骤。

步骤 3 在数据帧或远程帧的接收步骤 4，置 CAN_CONTROL[umask]为 1，使能报文接收过滤功能。

----结束

如果需要接收固定某一个 ID 的报文（单一报文）。如下面的例子 1。

例子 1：向 IF1_MASK2 寄存器写 0xDFFF，向 IF1_MASK1 寄存器写 0xFFFF。掩码全有效，表示报文接收过滤功能只接收单一 ID 报文，即 Rx 报文对象相同 ID 的报文，其他 ID 的报文均被过滤掉。

如果需要接收某些 ID 的报文（非单一报文）。如下面的例子 2 和例子 3。

例子 2：只接收报文 ID=0x7F2 的标准数据帧和报文 ID=0x7F3 的标准数据帧，那么 IF1_ARBITRATION2 bit[12:2]写 0x7F2，IF1_MASK2[mxtd]写 1，[mdir]写 1，[msk]的 28~16 写 0x7FE（忽略 ID 的最后一个 bit）。

例子 3：只接收报文 ID=0x7FFFFF2 的扩展数据帧和报文 ID=0x7FFFFF3 拓展数据帧，那么 IF1_ARBITRATION2 bit[12:0]写 0x7FF，IF1_ARBITRATION1 写 0xFFF2，IF2_MASK2[mxtd]写 1，[mdir]写 1，[msk]的 28~16 写 0x1FFF，IF2_MASK1 写 0xFFFE（忽略 ID 的最后一个 bit）。

须知

若同一节点接收的报文 ID 同时满足报文对象 1 和报文对象 2 且 eob 位均为 1，则该报文实际保存在报文对象 1 中，即报文对象的序号越低，优先级越高。当存在多个这样的报文需要接收时，报文对象 1 的内容会被新的报文覆盖。

22.4.7 连续报文接收

CAN 总线控制器可以通过配置 IF2_MESSAGE_CONTROL[eob]来支持接收连续的数据帧或者远程帧，即在配置 Rx 报文对象时，将空闲的报文对象除序号最大的外，其余的 eob 位均置 0，序号最大的 eob 位置 1。这样，这些空闲的报文对象就组成一片连续的缓存，以此接收发送节点发送的连续信息。

例如，假设 CAN 网络上的一个节点每次发送的信息为 64 字节，即为 8 个数据帧，那么控制器可以将空闲报文对象 1~7 的 eob 位置 0，报文对象 8 的 eob 位置 1，ID 均置为匹配 ID，其余配置相同。此时，该节点的信息将被存储在报文对象 1~8 内，CPU 可以通过访问 IF 报文对象接口寄存器读取报文对象 1~8 来获取数据。

须知

若同一节点接收的报文 ID 同时满足报文对象 1 和报文对象 2，报文对象 1 的 eob 位为 0，报文对象 2 的 eob 为 1 时（即报文对象 2 是连续报文的最后 1 个对象），则该报文实际保存在报文对象 1 中，即报文对象的序号越低，优先级越高。当存在 2 个这样的报文需要接收时，报文对象 1 接收第 1 个报文，报文对象 2 接收第 2 个报文。当报文数量超过了连续报文对象的数量时，报文对象的内容会依次被覆盖。

22.4.8 错误处理

CAN 总线控制器主动错误和被动错误的处理流程符合标准协议 CAN 2.0A 和 CAN 2.0B。软件可以通过寄存器 CAN_STATUS 和寄存器 CAN_ERROR_COUNTER 了解错误信息。

一旦 CAN 总线控制器进入 Bus-off 状态，该控制器将被 CAN 总线排除。只有重新初始化后，该控制器才能再次参与总线通信。并且在初始化完成后，该节点必须在总线上检测到 128 次连续的 11 个隐性位才能重新回到主动错误状态，此时错误计数器清零。

Bus-off 状态的修复步骤如下：

步骤 1 查询 CAN_STATUS[boff]，如果该位为 1，表明该节点进入 Bus-off（总线关闭）状态。

步骤 2 向 CAN_CONTROL[init]置 1，重新初始化该节点。

步骤 3 向 CAN_CONTROL[init]置 0，初始化完成，该节点开始监测总线。

步骤 4 查询 CAN_STATUS[boff]，直至该位为 0 为止，即表明该节点退出 Bus-off 状态。

----结束

须知

- 只有当检测到次连续的 11 个隐性位后，该节点才能退出 Bus-off 状态，并错误计数器清零，否则将重新监测。
- 当 CAN 控制器退出 Bus-off 状态并且 CAN_CONTROL 寄存器 bit[5]使能时，控制器将自动重新发送上次未发送成功的数据帧或远程帧，直至成功或再次进入 Bus-off 状态为止。

22.5 CAN 寄存器概览

CAN 寄存器基地址为 0x1460_0000。

CAN 寄存器概览如表 22-3 所示。

表22-3 CAN 寄存器概览

偏移地址	名称	描述	页码
0x0000	CAN_CONTROL	CAN 控制寄存器	CAN_CONTROL
0x0004	CAN_STATUS	CAN 状态寄存器	CAN_STATUS
0x0008	CAN_ERROR_COUNTER	CAN 错误计数寄存器	CAN_ERROR_COUNTER
0x000c	BIT_TIMING	位时间寄存器	BIT_TIMING
0x0010	CAN_INTERRUPT	CAN 中断寄存器	CAN_INTERRUPT
0x0014	CAN_TEST	CAN 调试寄存器	CAN_TEST
0x0018	BRP_EXTENSION	BRP 扩展寄存器	BRP_EXTENSION
0x0020	IF1_COMMAND_REQUEST	IF1 命令请求寄存器	IF1_COMMAND_REQUEST
0x0024	IF1_COMMAND_MASK	IF1 命令掩码寄存器	IF1_COMMAND_MASK

偏移地址	名称	描述	页码
0x0028	IF1_MASK1	IF1 掩码寄存器 1	IF1_MASK1
0x002c	IF1_MASK2	IF1 掩码寄存器 2	IF1_MASK2
0x0030	IF1_ARBITRATION1	IF1 仲裁寄存器 1	IF1_ARBITRATION1
0x0034	IF1_ARBITRATION2	IF1 仲裁寄存器 2	IF1_ARBITRATION2
0x0038	IF1_MESSAGE_CONTROL	IF1 报文控制寄存器	IF1_MESSAGE_CONTROL
0x003c	IF1_DATAA1	IF1 数据 A1 寄存器	IF1_DATAA1
0x0040	IF1_DATAA2	IF1 数据 A2 寄存器	IF1_DATAA2
0x0044	IF1_DATAB1	IF1 数据 B1 寄存器	IF1_DATAB1
0x0048	IF1_DATAB2	IF1 数据 B2 寄存器	IF1_DATAB2
0x0080	IF2_COMMAND_REQUEST	IF2 命令请求寄存器	IF2_COMMAND_REQUEST
0x0084	IF2_COMMAND_MASK	IF2 命令掩码寄存器	IF2_COMMAND_MASK
0x0088	IF2_MASK1	IF2 掩码寄存器 1	IF2_MASK1
0x008c	IF2_MASK2	IF2 掩码寄存器 2	IF2_MASK2
0x0090	IF2_ARBITRATION1	IF2 仲裁寄存器 1	IF2_ARBITRATION1
0x0094	IF2_ARBITRATION2	IF2 仲裁寄存器 2	IF2_ARBITRATION2
0x0098	IF2_MESSAGE_CONTROL	IF2 报文控制寄存器	IF2_MESSAGE_CONTROL
0x009c	IF2_DATAA1	IF2 数据 A1 寄存器	IF2_DATAA1
0x00a0	IF2_DATAA2	IF2 数据 A2 寄存器	IF2_DATAA2

偏移地址	名称	描述	页码
0x00a4	IF2_DATAB1	IF2 数据 B1 寄存器	IF2_DATAB1
0x00a8	IF2_DATAB2	IF2 数据 B2 寄存器	IF2_DATAB2
0x0100	TRANSMISSION_REQUEST1	传输请求状态寄存器 1	TRANSMISSION_REQUEST1
0x0104	TRANSMISSION_REQUEST2	传输请求状态寄存器 2	TRANSMISSION_REQUEST2
0x0120	NEW_DATA1	新数据状态寄存器 1	NEW_DATA1
0x0124	NEW_DATA2	新数据状态寄存器 2	NEW_DATA2
0x0140	INTERRUPT_PENDING1	中断待处理状态寄存器 1	INTERRUPT_PENDING1
0x0144	INTERRUPT_PENDING2	中断待处理状态寄存器 2	INTERRUPT_PENDING2
0x0160	MESSAGE_VALID1	报文有效状态寄存器 1	MESSAGE_VALID1
0x0164	MESSAGE_VALID2	报文有效状态寄存器 2	MESSAGE_VALID2

22.6 CAN 寄存器描述

CAN_CONTROL

CAN_CONTROL 为 CAN 控制寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7]	RW	test	测试模式使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[6]	RW	cce	更改配置使能。 0：禁止，CPU 不能写 Bit_Timing 寄存器； 1：使能，当 init 为 1 时，CPU 能写 Bit_Timing 寄存器。	0x0
[5]	RW	dar	关闭自动重传。 0：开启自动重传，自动重传受干扰的报文； 1：关闭自动重传。	0x0
[4]	RO	reserved	保留。	0x0
[3]	RW	eie	错误中断使能。 0：禁止； 1：使能错误中断，CAN_STATUS 寄存器中的 BOff 或 EWarn 变化时会产生中断。	0x0
[2]	RW	sie	状态改变中断使能。 0：禁止； 1：使能状态改变中断，报文传输成功或检测到 CAN 总线错误时会产生中断。	0x0
[1]	RW	ie	模块中断使能。 0：禁止； 1：使能，CAN 模块中断总开关。	0x0
[0]	RW	init	初始化使能。 0：禁止，此时 CAN 控制器正常工作； 1：使能，此时 CAN 控制器初始化。	0x1

CAN_STATUS

CAN_STATUS 为 CAN 状态寄存器。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7]	RO	boff	Bus-off 状态。 0：CAN 模块不处于 Bus-off 状态； 1：CAN 模块处于 Bus-off 状态。	0x0
[6]	RO	ewarn	警告状态。 0：所有错误计数器均未达到 96； 1：至少有一个错误计数器达到 96。	0x0
[5]	RO	epass	错误状态。 0：处于 CAN 协议所定义的主动错误； 1：处于 CAN 协议所定义的被动错误。	0x0
[4]	RW	rxok	报文接收状态。 0：接收报文失败，rxok 不会被 CAN 控制器复位； 1：接收报文成功（独立于报文接收过滤），rxok 只有 CPU 可以清零。	0x0
[3]	RW	txok	报文发送状态。 0：发送报文失败，txok 不会被 CAN 控制器复位； 1：发送报文成功（没有错误且被至少 1 个 CAN 节点应答），txok 只有 CPU 可以清零。	0x0
[2:0]	RW	lec	错误状态码，用来表示 CAN 总线上最后一次错误状态。 000：无错误； 001：填充错误； 010：形式错误； 011：应答错误； 100：bit1 错误；	0x0

Bits	Access	Name	Description	Reset
			101：bit0 错误； 110：CRC 错误； 111：未使用，没有检测到 CAN 总线事件，CPU 最后将此值写入 lec。	

CAN_ERROR_COUNTER

CAN_ERROR_COUNTER 为 CAN 错误计数寄存器。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RO	rp	被动错误接收状态。 0：接收错误计数器未达到 CAN 协议所定义的被动错误水线； 1：接收错误计数器达到 CAN 协议所定义的被动错误水线。	0x0
[14:8]	RO	rec	接收错误计数器。 有效值在 0~127。	0x00
[7:0]	RO	tec	发送错误计数器。 有效值在 0~255。	0x00

BIT_TIMING

BIT_TIMING 为位时间寄存器。

Offset Address: 0x000c Total Reset Value: 0x0000_2301

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14:12]	RW	tseg2	相位缓冲段 2。 配置值为 0~7，实际值比配置值多 1。	0x2

Bits	Access	Name	Description	Reset
[11:8]	RW	tseg1	相位缓冲段 1。 配置值为 1~15，实际值比配置值多 1。	0x3
[7:6]	RW	sjw	再同步跳转宽度。 配置值为 0~3，实际值比配置值多 1。	0x0
[5:0]	RW	brp	波特率系数。 配置值为 0~63，实际值比配置值多 1。	0x01

CAN_INTERRUPT

CAN_INTERRUPT 为 CAN 中断寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	intId	中断 ID。 0x0000：无中断； 0x0001：中断源是第 1 个报文对象； 0x0002：中断源是第 2 个报文对象； 0x0003：中断源是第 3 个报文对象； 0x0004：中断源是第 4 个报文对象； 0x0005：中断源是第 5 个报文对象； 0x0006：中断源是第 6 个报文对象； 0x0007：中断源是第 7 个报文对象； 0x0008：中断源是第 8 个报文对象； 0x0009：中断源是第 9 个报文对象； 0x000A：中断源是第 10 个报文对象； 0x000B：中断源是第 11 个报文对象； 0x000C：中断源是第 12 个报文对象； 0x000D：中断源是第 13 个报文对象；	0x0000

Bits	Access	Name	Description	Reset
			0x000E：中断源是第 14 个报文对象； 0x000F：中断源是第 15 个报文对象； 0x0010：中断源是第 16 个报文对象； 0x0011：中断源是第 17 个报文对象； 0x0012：中断源是第 18 个报文对象； 0x0013：中断源是第 19 个报文对象； 0x0014：中断源是第 20 个报文对象； 0x0015：中断源是第 21 个报文对象； 0x0016：中断源是第 22 个报文对象； 0x0017：中断源是第 23 个报文对象； 0x0018：中断源是第 24 个报文对象； 0x0019：中断源是第 25 个报文对象； 0x001A：中断源是第 26 个报文对象； 0x001B：中断源是第 27 个报文对象； 0x001C：中断源是第 28 个报文对象； 0x001D：中断源是第 29 个报文对象； 0x001E：中断源是第 30 个报文对象； 0x001F：中断源是第 31 个报文对象； 0x0020：中断源是第 32 个报文对象； 0x0021 ~ 0x7FFF：无效； 0x8000：状态中断； 0x8001~0xFFFF：无效。	

CAN_TEST

CAN_TEST 为 CAN 调试寄存器。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000

Bits	Access	Name	Description	Reset
[7]	RO	rx	CAN_RX 管脚监控。 0：CAN 总线为显性（CAN_RX=’0’）； 1：CAN 总线为隐形（CAN_RX=’1’）。	0x0
[6:5]	RW	tx	CAN_TX 管脚控制。 00：不控制 CAN_TX 管脚； 01：采样点能被 CAN_TX 管脚监视； 10：CAN_TX 管脚驱动显性值（'0’）； 11：CAN_TX 管脚驱动隐形值（'1’）。	0x0
[4]	RW	lback	Loop back 模式使能。 0：禁止； 1：使能。	0x0
[3]	RW	silent	Silent 模式使能。 0：禁止； 1：使能。	0x0
[2]	RW	basic	Basic 模式使能。 0：禁止； 1：使能，此时 IF1 寄存器用作 Tx Buffer，IF2 寄存器用作 Rx Buffer。	0x0
[1:0]	RO	reserved	保留。	0x0

BRP_EXTENSION

BRP_EXTENSION 为 BRP 扩展寄存器。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000

Bits	Access	Name	Description	Reset
[3:0]	RW	brpe	波特率系数扩展。 扩展后的波特率系数有 brpe 和 brp 共同组成，其中高位是 brpe，低位是 brp。 此时，波特率系数的配置值最大可达到 1023，实际值比配置值多 1。	0x0

IF1_COMMAND_REQUEST

IF1_COMMAND_REQUEST 为 IF1 命令请求寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RO	busy	Busy 标志。在 Basic 模式下，该位域可读可写。 0：IF1 命令已经完成； 1：IF1 命令执行中。	0x0
[14:6]	RO	reserved	保留。	0x000
[5:0]	RW	message number	报文对象序号。CPU 写该位域后，IF1 命令开始执行。 0x00：无效值，将指向 0x20； 0x01 ~ 0x20：报文对象序号； 0x21 ~ 0x3F：无效值，将指向 0x01 ~ 0x1F。	0x01

IF1_COMMAND_MASK

IF1_COMMAND_MASK 为 IF1 命令掩码寄存器。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7]	RW	wr_rd	读写命令，控制 IF1 报文缓冲寄存器和 Message RAM 传输方向。 0：读命令，将 Message RAM 中存储的报文对象回读到 IF1 报文缓冲寄存器； 1：写命令，将 IF1 报文缓冲寄存器写入到 Message RAM 中存储的报文对象。	0x0
[6]	RW	mask	命令掩码，控制 IF1_MASK 的传输。 当 wr_rd 为 1，写命令时： 0：不写入，报文对象的 msk、mdir、mxtd 不改变； 1：将 IF1_MASK1 和 IF1_MASK2 寄存器中的 msk、mdir、mxtd 写入报文对象。 当 wr_rd 为 0，读命令时： 0：不回读，IF1_MASK 不改变； 1：将报文对象的 Msk、Mdir、mxtd 回读到 IF1_MASK1 和 IF1_MASK2 寄存器中。	0x0
[5]	RW	Arb	命令掩码，控制 IF1_ARBITRATION 的传输。 当 wr_rd 寄存器为 1，写命令时： 0：不写入，报文对象的 ID、Dir、Xtd、MsgVal 不改变； 1：将 IF1_ARBITRATION1 和 IF1_ARBITRATION2 中的 ID、Dir、Xtd、MsgVal 位写入报文对象。 当 wr_rd 寄存器为 0，读命令时： 0：不回读，IF1_ ARBITRATION 不改变； 1：将报文对象的 ID、Dir、Xtd、MsgVal 回读到 IF1_ARBITRATION1 和 IF1_ARBITRATION2 寄存器中。	0x0

Bits	Access	Name	Description	Reset
[4]	RW	Control	命令掩码，控制 IF1_MESSAGE_CONTROL 的传输。 当 wr_rd 寄存器为 1，写命令时： 0：不写入，报文对象中的 IF1_MESSAGE_CONTROL 相关 bit 位不改变； 1：将 IF1_MESSAGE_CONTROL 的 bit 位写入报文对象。 当 wr_rd 寄存器为 0，读命令时： 0：不回读，IF1_MESSAGE_CONTROL 不改变； 1：将报文对象中的相应 bit 位回读到 IF1_MESSAGE_CONTROL 寄存器中。	0x0
[3]	RW	ClrIntPnd	命令掩码，清除报文对象的中断待处理。 当 wr_rd 寄存器为 1，写命令时： 该位域忽略。 当 wr_rd 寄存器为 0，读命令时： 0：报文对象的 IntPnd 位保持不变； 1：清除报文对象的 IntPnd 位。	0x0
[2]	RW	txrqst_NewDat	命令掩码，控制报文对象的 txrqst 位或者 NewDat 位。 当 wr_rd 寄存器为 1 时，写命令时： 访问 txrqst 位。 0：将 IF1_MESSAGE_CONTROL 寄存器的 txrqst 位写入报文对象； 1：忽略 IF1_MESSAGE_CONTROL 寄存器的 txrqst 位。直接置位报文对象的 txrqst 位，请求报文发送。 当 wr_rd 为 0 时，读命令时： 访问 newdat 位，IF1_MESSAGE_CONTROL 寄存器的	0x0

Bits	Access	Name	Description	Reset
			newdata 位不会被更新。 0：报文对象的 newdat 位均保持不变； 1：清除报文对象的 newdat 位。	
[1]	RW	dataa	命令掩码，控制 IF1_DATAA 的传输。 当 wr_rd 为 1 时，写命令时： 0：报文对象中的数据 Byte 0~3 保持不变； 1：将 IF1_DATAA 寄存器写入报文对象，更新报文对象中的数据 Byte0~3。 当 wr_rd 寄存器为 0 时，读命令时： 0：IF1_DATAA 寄存器保持不变； 1：将报文对象中的数据 Byte0~3 回读到 IF1_DATAA 寄存器中。	0x0
[0]	RW	DataB	命令掩码，控制 IF1_DATAB 的传输。 当 wr_rd 寄存器为 1 时，写命令时： 0：报文对象中的数据 Byte 4~7 保持不变； 1：将 IF1_DATAB 寄存器写入报文对象，更新报文对象中的数据 Byte4~7。 当 wr_rd 寄存器为 0 时，读命令时： 0：IF1_DATAB 寄存器保持不变； 1：将报文对象中的数据 Byte4~7 回读到 IF1_DATAB 寄存器中。	0x0

IF1_MASK1

IF1_MASK1 为 IF1 掩码寄存器 1。

Offset Address: 0x0028 Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[15:0]	RW	Msk	报文对象 ID 第 15 位至第 0 位掩码，用于报文接收过滤。 0：ID 第 15 位至第 0 位不用于报文接收过滤； 1：ID 第 15 位至第 0 位用于报文接收过滤。	0xFFFF

IF1_MASK2

IF1_MASK2 为 IF1 掩码寄存器 2。

Offset Address: 0x002c Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	MXtd	报文对象的拓展 ID 位（Xtd）掩码，用于接收报文过滤。 0：Xtd 不用于报文接收过滤； 1：Xtd 用于报文接收过滤。	0x1
[14]	RW	MDir	报文对象的方向位（Dir）掩码，用于接收报文过滤。 0：Dir 不用于报文接收过滤； 1：Dir 用于报文接收过滤。	0x1
[13]	RO	reserved	保留。	0x1
[12:0]	RW	Msk	报文对象 ID 第 28 位至第 16 位掩码，用于接收报文过滤。 0：ID 第 28 位至第 16 位不用于报文接收过滤； 1：ID 第 28 位至第 16 位用于报文接收过滤。	0x1FFF

IF1_ARBITRATION1

IF1_ARBITRATION1 为 IF1 仲裁寄存器 1。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	ID	报文对象的报文 ID 第 15 位至第 0 位。 **注意：** **当报文对象用于收发扩展帧时，应使用全部 ID；** **当报文对象用于收发标准帧时，应使用 IF1_ARBITRATION2[ID]的高 11 位。**	0x0000

IF1_ARBITRATION2

IF1_ARBITRATION2 为 IF1 仲裁寄存器 2。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	MsgVal	报文对象有效使能。 0：禁止，该报文对象无效； 1：使能，该报文对象有效。	0x0
[14]	RW	Xtd	报文对象收发帧格式。 0：该报文对象收发标准格式帧； 1：该报文对象收发扩展格式帧。	0x0
[13]	RW	Dir	报文对象的方向。 0：报文对象为接收方向，接收报文； 1：报文对象为发送方向，发送报文。	0x0
[12:0]	RW	ID	报文对象的报文 ID 第 28 位至第 16 位。 **注意：**	0x0000

Bits	Access	Name	Description	Reset
			当报文对象用于收发扩展帧时，应使用全部 ID； **当报文对象用于收发标准帧时，应使用 IF2_ARBITRATION2[ID]的高 11 位。**	

IF1_MESSAGE_CONTROL

IF1_MESSAGE_CONTROL 为 IF1 报文控制寄存器。

Offset Address: 0x0038 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	NewDat	报文对象的新数据写入状态。 报文对象为发送方向时： 0：无新的帧数据写入报文对象，报文对象成功发送后会清除报文对象中的 NewDat； 1：有新的帧数据写入报文对象。 报文对象为接收方向时： 0：报文对象没有接收到新的帧； 1：报文对象接收到了新的帧。	0x0
[14]	RW	MsgLst	报文对象的报文丢失标识（仅在报文对象为接收方向时有效）。 0：报文对象无报文丢失； 1：报文对象有报文丢失，当报文对象的 NewDat 位未被清除，新的报文已接收时，报文对象的 MsgLst 置 1。	0x0
[13]	RW	IntPnd	报文对象的中断待处理标识。 0：该报文对象不是中断源； 1：该报文对象是中断源。	0x0
[12]	RW	Umask	报文对象是否使用报文掩码，用于报文接	0x0

Bits	Access	Name	Description	Reset
			收过滤。 0：不使用； 1：使用相关掩码位（Msk28~0，mxtd 和 Mdir 用于报文接收过滤。	
[11]	RW	TxIE	发送中断使能。 0：不使能，帧发送成功后 IntPnd 保持不变； 1：使能，帧发送成功后 IntPnd 将被至 1。	0x0
[10]	RW	RxIE	接收中断使能。 0：不使能，帧接收成功后 IntPnd 保持不变； 1：使能，帧接收成功后 IntPnd 将被至 1。	0x0
[9]	RW	RmtEn	远程帧使能。 0：不使能，接收远程帧后 txrqst 保持不变； 1：使能，接收远程帧后 txrqst 置 1。	0x0
[8]	RW	txrqst	传输请求。 0：该报文对象没在等待传输； 1：该报文对象请求传输，并且还没传输完成。	0x0
[7]	RW	eob	多报文接收模式。 0：该报文对象不是多报文接收模式中最后一个报文对象； 1：该报文对象是单一报文或是多报文接收模式中最后一个报文对象。	0x0
[6:4]	RO	reserved	保留。	0x0
[3:0]	RW	DLC	数据长度编码。 0x0~0x8：数据帧有 0~8 个数据字节； 0x9~0xF：数据帧有 8 个数据字节。	0x0

IF1_DATAA1

IF1_DATAA1 为 IF1 数据 A1 寄存器。

Offset Address: 0x003c Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	DATA1	CAN 帧数据 Byte 1。	0x00
[7:0]	RW	DATA0	CAN 帧数据 Byte 0。	0x00

IF1_DATAA2

IF1_DATAA2 为 IF1 数据 A2 寄存器。

Offset Address: 0x0040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	DATA3	CAN 帧数据 Byte 3。	0x00
[7:0]	RW	DATA2	CAN 帧数据 Byte 2。	0x00

IF1_DATAB1

IF1_DATAB1 为 IF1 数据 B1 寄存器。

Offset Address: 0x0044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	DATA5	CAN 帧数据 Byte 5。	0x00
[7:0]	RW	DATA4	CAN 帧数据 Byte 4。	0x00

IF1_DATAB2

IF1_DATAB2 为 IF1 数据 B2 寄存器。

Offset Address: 0x0048 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	DATA7	CAN 帧数据 Byte 7。	0x00
[7:0]	RW	DATA6	CAN 帧数据 Byte 6。	0x00

IF2_COMMAND_REQUEST

IF2_COMMAND_REQUEST 为 IF2 命令请求寄存器。

Offset Address: 0x0080 Total Reset Value: 0x0000_0001

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RO	BUSY	Busy 标志。 0：IF2 命令已经完成; 1：IF2 命令执行中。	0x0
[14:6]	RO	reserved	保留。	0x000
[5:0]	RW	Message Number	报文对象序号。CPU 写该位域后，IF2 命令开始执行。 0x00：无效值，将指向 0x20; 0x01~0x20：报文对象序号; 0x21~0x3F：无效值，将指向 0x01~0x1F。	0x01

IF2_COMMAND_MASK

IF2_COMMAND_MASK 为 IF2 命令掩码寄存器。

Offset Address: 0x0084 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7]	RW	wr_rd	读写命令，控制 IF2 报文缓冲寄存器和 Message RAM 传输方向。 0：读命令，将 Message RAM 中存储的报文对象回读到 IF2 报文缓冲寄存器； 1：写命令，将 IF2 报文缓冲寄存器写入到 Message RAM 中存储的报文对象。	0x0
[6]	RW	mask	命令掩码，控制 IF2_MASK 的传输。 当 wr_rd 寄存器为 1，写命令时： 0：不写入，报文对象的 ID Mask、mdir、mxtd 不改变； 1：将 IF2_MASK1 和 IF2_MASK2 寄存器中的 ID Mask、mdir、mxtd 写入报文对象。 当 wr_rd 寄存器为 0，读命令时： 0：不回读，IF2_MASK 不改变； 1：将报文对象的 ID Mask、mdir、mxtd 回读到 IF2_MASK1 和 IF2_MASK2 寄存器中。	0x0
[5]	RW	arb	命令掩码，控制 IF2_ARBITRATION 的传输。 当 wr_rd 寄存器为 1，写命令时： 0：不写入，报文对象的 ID、dir、xtd、msgval 不改变； 1：将 IF2_ARBITRATION1 和 IF2_ARBITRATION2 中的 ID、dir、xtd、msgval 位写入报文对象。 当 wr_rd 寄存器为 0，读命令时： 0：不回读，IF2_ARBITRATION 不改变； 1：将报文对象的 ID、dir、xtd、msgval 回读到 IF2_ARBITRATION1 和	0x0

Bits	Access	Name	Description	Reset
			IF2_ARBITRATION2 寄存器中。	
[4]	RW	control	命令掩码，控制 IF2_MESSAGE_CONTROL 的传输。 当 wr_rd 寄存器为 1，写命令时： 0：不写入，报文对象中的 IF2_MESSAGE_CONTROL 相关 bit 位不改变； 1：将 IF2_MESSAGE_CONTROL 的 bit 位写入报文对象。 当 wr_rd 寄存器为 0，读命令时： 0：不回读，IF2_MESSAGE_CONTROL 不改变； 1：将报文对象中的相应 bit 位回读到 IF2_MESSAGE_CONTROL 寄存器中。	0x0
[3]	RW	clrintpnd	命令掩码，清除报文对象的中断待处理。 当 wr_rd 寄存器为 1，写命令时： 该位域忽略。 当 wr_rd 寄存器为 0，读命令时： 0：报文对象的 intpnd 位保持不变； 1：清除报文对象的 intpnd 位。	0x0
[2]	RW	txrqst_newdat	命令掩码，控制报文对象的 txrqst 位或者 NewDat 位。 当 wr_rd 寄存器为 1 时，写命令时： 访问 txrqst 位。 0：报文对象 txrqst 位保持不变； 1：置位报文对象的 txrqst 位，请求报文发送。 当 wr_rd 寄存器为 0 时，读命令时： 访问 newdat 位。 0：报文对象的 newdat 位保持不变；	0x0

Bits	Access	Name	Description	Reset
			1：清除报文对象的 newdat 位。	
[1]	RW	DataA	命令掩码，控制 IF2_DATAA 的传输。 当 wr_rd 寄存器为 1 时，写命令时： 0：报文对象中的数据 Byte 0~3 保持不变； 1：将 IF2_DATAA 寄存器写入报文对象，更新报文对象中的数据 Byte0~3。 当 wr_rd 寄存器为 0 时，读命令时： 0：IF2_DATAA 寄存器保持不变； 1：将报文对象中的数据 Byte0~3 回读到 IF2_DATAA 寄存器中。	0x0
[0]	RW	DataB	命令掩码，控制 IF2_DATAB 的传输。 当 wr_rd 寄存器为 1 时，写命令时： 0：报文对象中的数据 Byte 4~7 保持不变； 1：将 IF2_DATAB 寄存器写入报文对象，更新报文对象中的数据 Byte4~7。 当 wr_rd 寄存器为 0 时，读命令时： 0：IF2_DATAB 寄存器保持不变； 1：将报文对象中的数据 Byte4~7 回读到 IF2_DATAB 寄存器中。	0x0

IF2_MASK1

IF2_MASK1 为 IF2 掩码寄存器 1。

Offset Address: 0x0088 Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	msk	报文对象 ID 第 15 位至第 0 位掩码，用于报文接收过滤。	0xFFFF

Bits	Access	Name	Description	Reset
			0：ID 第 15 位至第 0 位不用于报文接收过滤； 1：ID 第 15 位至第 0 位用于报文接收过滤。	

IF2_MASK2

IF2_MASK2 为 IF2 掩码寄存器 2。

Offset Address: 0x008c Total Reset Value: 0x0000_FFFF

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	mxtd	报文对象的拓展 ID 位（xtd）掩码，用于报文接收过滤。 0：xtd 不用于报文接收过滤； 1：xtd 用于报文接收过滤。	0x1
[14]	RW	mdir	报文对象的方向位（dir）掩码，用于报文接收过滤。 0：dir 不用于报文接收过滤； 1：dir 用于报文接收过滤。	0x1
[13]	RO	reserved	保留。	0x1
[12:0]	RW	msk	报文对象 ID 第 28 位至第 16 位掩码，用于报文接收过滤。 0：ID 第 28 位至第 16 位不用于报文接收过滤； 1：ID 第 28 位至第 16 位用于报文接收过滤。	0x1FFF

IF2_ARBITRATION1

IF2_ARBITRATION1 为 IF2 仲裁寄存器 1。

Offset Address: 0x0090 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	id	报文对象的报文 ID 第 15 位至第 0 位。 **注意：** **当报文对象用于收发扩展帧时，应使用全部 ID；** **当报文对象用于收发标准帧时，应使用** IF2_ARBITRATION2**[id]的高 11 位。**	0x0000

IF2_ARBITRATION2

IF2_ARBITRATION2 为 IF2 仲裁寄存器 2。

Offset Address: 0x0094 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	msgval	报文对象有效使能。 0：禁止，该报文对象无效； 1：使能，该报文对象有效。	0x0
[14]	RW	xtd	报文对象收发帧格式。 0：该报文对象收发标准格式帧； 1：该报文对象收发扩展格式帧。	0x0
[13]	RW	dir	报文对象的方向。 0：报文对象为接收方向，接收报文； 1：报文对象为发送方向，发送报文。	0x0
[12:0]	RW	id	报文对象的报文 id 第 28 位至第 16 位。 **注意：** **当报文对象用于收发扩展帧时，应使用全部 id；** **当报文对象用于收发标准帧时，应使用**	0x0000

Bits	Access	Name	Description	Reset
			IF2_ARBITRATION2**[id]的高 11 位。**	

IF2_MESSAGE_CONTROL

IF2_MESSAGE_CONTROL 为 IF2 报文控制寄存器。

Offset Address: 0x0098 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW	newdat	报文对象的帧数据标识。 报文对象为发送方向时： 0：无新的帧数据写入报文对象，报文对象成功发送后会清除报文对象中的 newdat； 1：有新的帧数据写入报文对象。 报文对象为接收方向时： 0：报文对象没有接收到新的帧； 1：报文对象接收到了新的帧。	0x0
[14]	RW	msglst	报文对象的报文丢失标识（仅在报文对象为接收方向时有效）。 0：报文对象无报文丢失； 1：报文对象有报文丢失，当报文对象的 newdat 位未被清除，新的报文已接收时，报文对象的 msglst 置 1。	0x0
[13]	RW	intpnd	报文对象的中断待处理标识。 0：该报文对象不是中断源； 1：该报文对象是中断源。	0x0
[12]	RW	umask	报文对象是否使用报文掩码，用于报文接收过滤。 0：不使用； 1：使用相关掩码位（Msk28-0，mxtd 和	0x0

Bits	Access	Name	Description	Reset
			mdir）用于报文接收过滤。	
[11]	RW	txie	发送中断使能。 0：不使能，帧发送成功后 intpnd 保持不变； 1：使能，帧发送成功后 intpnd 将被至 1。	0x0
[10]	RW	rxie	接收中断使能。 0：不使能，帧接收成功后 intpnd 保持不变； 1：使能，帧接收成功后 intpnd 将被至 1。	0x0
[9]	RW	rmten	远程帧使能。 0：不使能，接收远程帧后 txrqst 保持不变； 1：使能，接收远程帧后 txrqst 置 1。	0x0
[8]	RW	txrqst	传输请求。 0：该报文对象没在等待传输； 1：该报文对象请求传输，并且还没传输完成。	0x0
[7]	RW	eob	多报文接收模式。 0：该报文对象不是多报文接收模式中最后一个报文对象； 1：该报文对象是单一报文或是多报文接收模式中最后一个报文对象。	0x0
[6:4]	RO	reserved	保留。	0x0
[3:0]	RW	dlc	数据长度编码。 0x0~0x8：数据帧有 0-8 个数据字节； 0x9~0xF：数据帧有 8 个数据字节。	0x0

IF2_DATAA1

IF2_DATAA1 为 IF2 数据 A1 寄存器。

Offset Address: 0x009c Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	data1	CAN 帧数据 Byte 1。	0x00
[7:0]	RW	data0	CAN 帧数据 Byte 0。	0x00

IF2_DATAA2

IF2_DATAA2 为 IF2 数据 A2 寄存器。

Offset Address: 0x00a0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	data3	CAN 帧数据 Byte 3。	0x00
[7:0]	RW	data2	CAN 帧数据 Byte 2。	0x00

IF2_DATAB1

IF2_DATAB1 为 IF2 数据 B1 寄存器。

Offset Address: 0x00a4 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	data5	CAN 帧数据 Byte 5。	0x00
[7:0]	RW	data4	CAN 帧数据 Byte 4。	0x00

IF2_DATAB2

IF2_DATAB2 为 IF2 数据 B2 寄存器。

Offset Address: 0x00a8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:8]	RW	data7	CAN 帧数据 Byte 7。	0x00
[7:0]	RW	data6	CAN 帧数据 Byte 6。	0x00

TRANSMISSION_REQUEST1

TRANSMISSION_REQUEST1 为传输请求状态寄存器 1。

Offset Address: 0x0100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	txrqst16-1	传输请求状态（txrqst16-1 每 bit 分别对应报文对象 16-1）。CPU 可以回读检查传输请求中的报文对象。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象发出传输请求。 0：该报文对象没有等待传输； 1：该报文对象请求发送，并且还没传输完成。	0x0000

TRANSMISSION_REQUEST2

TRANSMISSION_REQUEST2 为传输请求状态寄存器 2。

Offset Address: 0x0104 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	txrqst32-17	传输请求状态（txrqst32-17 每 bit 分别对应	0x0000

Bits	Access	Name	Description	Reset
			报文对象 32-17）。CPU 可以回读检查传输请求中的报文对象。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象发出传输请求。 0：该报文对象没有等待传输； 1：该报文对象请求发送，并且还没传输完成。	

NEW_DATA1

NEW_DATA1 为新数据状态寄存器 1。

Offset Address: 0x0120 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	newdat16-1	新数据写入状态（newdat16-1 每 bit 分别对应报文对象 16-1）。CPU 可以回读检查报文对象是否接收到新的数据帧。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 newdat 位域置位或清零。CAN 控制器的报文对象在接收报文成功后，相应的 newdat 自动置位。 0：无新数据写入该报文对象； 1：有新数据写入该报文对象。	0x0000

NEW_DATA2

NEW_DATA2 为新数据状态寄存器 2。

Offset Address: 0x0124 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	newdat32-17	新数据写入状态（newdat32-17 每 bit 分别	0x0000

Bits	Access	Name	Description	Reset
			对应报文对象 32-17）。CPU 可以回读检查报文对象是否接收到新的数据帧。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 newdat 位域置位或清零。CAN 控制器的报文对象在接收报文成功后，相应的 newdat 自动置位。 0：无新数据写入该报文对象； 1：有新数据写入该报文对象。	

INTERRUPT_PENDING1

INTERRUPT_PENDING1 为中断待处理状态寄存器 1。

Offset Address: 0x0140 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	intpnd16-1	中断待处理状态（intpnd16-1 每 bit 分别对应报文对象 16-1）。CPU 可以回读检查报文对象是否是中断源。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 intpnd 位域置位或清零。CAN 控制器的报文对象在接收报文成功后，或者发送报文成功后，相应的 intpnd 自动置位。 0：该报文对象不是中断源； 1：该报文对象是中断源。	0x0000

INTERRUPT_PENDING2

INTERRUPT_PENDING2 为中断待处理状态寄存器 2。

Offset Address: 0x0144 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[15:0]	RO	intpnd32-17	中断待处理状态（intpnd32-17 每 bit 分别对应报文对象 32-17）。CPU 可以回读检查报文对象是否是中断源。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 IntPnd 位域置位或清零。CAN 控制器的报文对象在接收报文成功后，或者发送报文成功后，相应的 intpnd 自动置位。 0：该报文对象不是中断源； 1：该报文对象是中断源。	0x0000

MESSAGE_VALID1

MESSAGE_VALID1 为报文有效状态寄存器 1。

Offset Address: 0x0160 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	msgval16-1	报文对象有效状态（msgval16-1 每 bit 分别对应报文对象 16-1）。CPU 可以回读检查报文对象是否有效。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 msgval 位域置位或清零。 0：该报文对象无效； 1：该报文对象有效。	0x0000

MESSAGE_VALID2

MESSAGE_VALID2 为报文有效状态寄存器 2。

Offset Address: 0x0164 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	msgval32-17	报文对象有效状态（msgval32-17 每 bit 分	0x0000

Bits	Access	Name	Description	Reset
			别对应报文对象 32-17）。CPU 可以回读检查报文对象是否有效。CPU 可以通过 IF 报文对象接口寄存器组，配置报文对象 msgval 位域置位或清零。 0：该报文对象无效； 1：该报文对象有效。	

23 ADC 电压参考(ADCVREF)

23.1 概述

MCU 中集成了高精度的 ADC 电压参考模块，该模块主要包括电压基准源和输出缓冲器。电压基准源为缓冲器提供参考电压和参考电流，输出缓冲器为 ADC 中 DAC 电容阵列及采样保持电路提供带有瞬态响应能力较强的电压基准。

23.2 特性

- 支持斩波功能
- 输出电压 1.1V

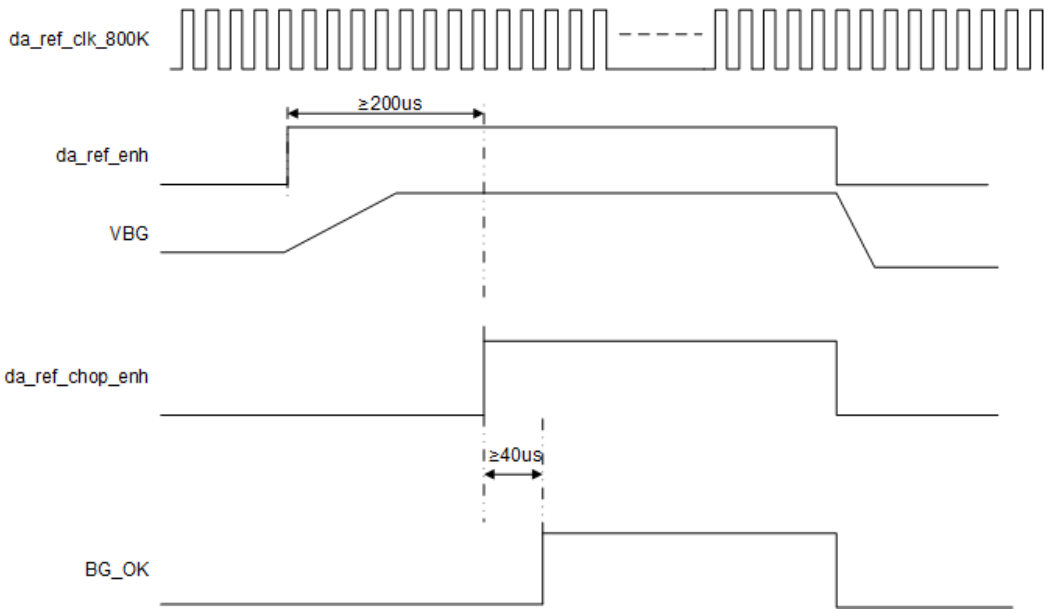
23.3 ADCVREF 使能

ADCVREF 使能前，打开时钟门控 PERI_CRG660[clk_ana_cken]，解除复位状态 PERI_CRG660[ana_srst_req]，解除 ADCVREF 复位状态 PERI_CRG664[vref_srst_req]。

ADCVREF 使能通过寄存器 VREF_CTRL_REG0[da_ref_enh]进行控制，斩波功能通过寄存器 VREF_CTRL_REG1[da_ref_chop_enh]控制使能。

23.4 工作模式

图23-1 ADC_VREF 工作时序图



ADCVREF 使能流程如下：

步骤 1 ADCVREF 使能前，打开时钟门控 PERI_CRG660[clk_ana_cken]，解除 ADCVREF 复位状态 PERI_CRG664[vref_srst_req]。

步骤 2 可选择配置 VREF_CTRL_REG1[da_ref_temp_trim_enh]为 1，使能高精度模式，提升 ADCVREF 的温漂性能。

步骤 3 配置 VREF_CTRL_REG0[da_ref_enh]为 1，使能 ADCVREF。

步骤 4 等待 200us 后，配置 VREF_CTRL_REG1[da_ref_chop_enh]为 1，使能 ADCVREF 斩波功能。

步骤 5 等待 40us 后，ADCVREF 完成启动。

----结束

23.5 VREF_REG 寄存器概览

VREF_REG 寄存器概览如表 23-1 所示。

表23-1 VREF_REG 寄存器概览（基址是 0x1810_0000）

偏移地址	名称	描述	页码
0x0000	VREF_CTRL_REG0	VREF 控制寄存器 0	VREF_CTRL_REG0
0x0020	VREF_CTRL_REG1	VREF 控制寄存器 1	VREF_CTRL_REG1
0x0040	VREF_TRIM_REG0	VREF TRIM 寄存器 0	VREF_TRIM_REG0
0x0044	VREF_TRIM_REG1	VREF TRIM 寄存器 1	VREF_TRIM_REG1

23.6 VREF_REG 寄存器描述

VREF_CTRL_REG0

VREF_CTRL_REG0 为 VREF 控制寄存器 0。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	da_ref_enh	VREF 模块使能。 0：关闭； 1：使能。	0x0

VREF_CTRL_REG1

VREF_CTRL_REG1 为 VREF 控制寄存器 1。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[16]	RW	da_ref_temp_trim_enh	VREF 高精度模式使能。 0：关闭; 1：使能。	0x0
[15:1]	RO	reserved	保留。	0x0000
[0]	RW	da_ref_chop_enh	VREF 斩波功能使能。 0：关闭; 1：使能。	0x0

VREF_TRIM_REG0

VREF_TRIM_REG0 为 VREF TRIM 寄存器 0。

Offset Address: 0x0040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:24]	RW	da_ref_buf_trim	Vref_buffer trim 信息。	0x00
[23:16]	RW	da_ref_vbg_trim	VREF 电压的 trim 信息。	0x00
[15:8]	RW	da_ref_vref_trim	IBIAS 电压的 trim 信息。	0x00
[7:0]	RW	da_iref_trim	Tsensor trim 信息。	0x00

VREF_TRIM_REG1

VREF_TRIM_REG1 为 VREF TRIM 寄存器 1。

Offset Address: 0x0044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:24]	RW	da_ref_temp_trim0	基准温漂 trim 信息。	0x00
[23:16]	RW	da_ref_temp_trim1	基准温漂 trim 信息。	0x00
[15:8]	RW	da_ref_temp_trim2	基准温漂 trim 信息。	0x00
[7:0]	RW	da_ref_temp_trim3	基准温漂 trim 信息。	0x00

24 模数转换器（ADC）

24.1 概述

模数转换器模块采用逐次逼近型 ADC，也称为 SAR ADC（Successive Approximation ADC）。ADC 分为模拟和数字两个部分，其中模拟部分包含通道选择电路、逐次逼近电路，数字部分包含 ADC 采样控制、转换逻辑、总线接口、过采样、后处理功能以及与其他模块的连接。

24.2 特性

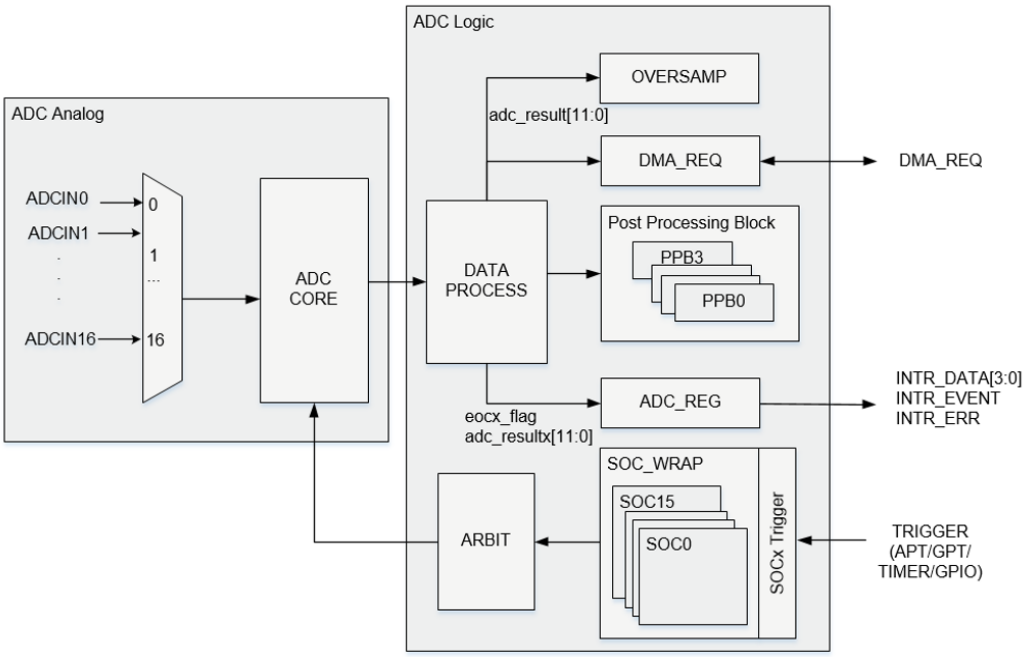
- 高性能
  - 12bit 采样精度
  - 采样率最高 3MSPS
  - 可编程控制采样时间
  - 支持误差校准功能
  - 支持 DMA 请求
- 模拟输入
  - 19 个模拟信号输入
- 工作模式
  - 支持单次转换模式
  - 支持连续转换模式
- 转换开始模块 SOC（Start of Conversion）
  - 16 个独立配置的 SOC

- 2 种可配置优先级组
- 20 个硬件触发源（APT/GPT/TIMER/GPIO）和 1 个软件触发源可选择
- 16 个独立寻址的转换结果寄存器和转换完成标志 EOC（End of Conversion）

- 中断
  - 4 个可灵活配置的数据中断
  - 1 个事件中断
  - 1 个异常中断
- 过采样
  - 12bit ~ 16bit 过采样精度
  - 8 ~ 256 倍硬件过采样
  - 独立的过采样结果寄存器
- 后处理功能
  - 4 个独立的后处理模块
  - 数据偏移
  - 误差计算
  - 阈值检测
  - 过零点检测
  - 采样延迟记录

24.2.1 框架

图24-1 ADC 模块框架



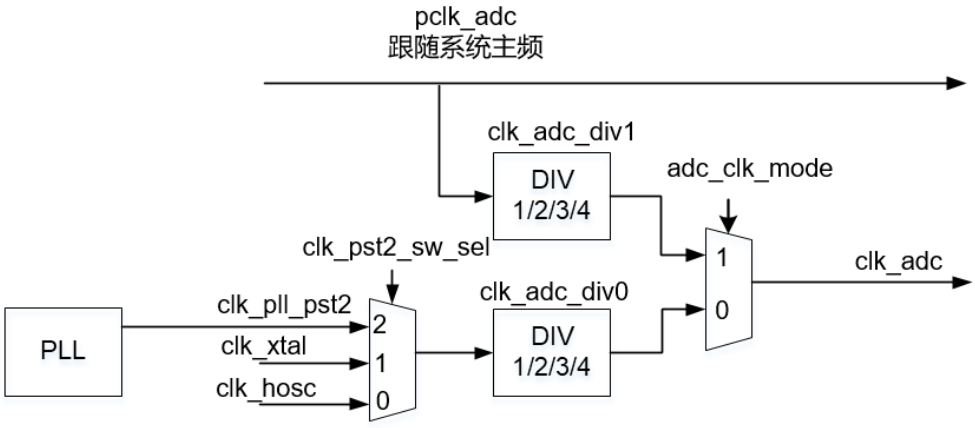
24.2.2 时钟

ADC 具有两个时钟，分别为系统时钟（pclk_adc）和 ADC 工作时钟（clk_adc），由系统 CRG 模块产生。

系统时钟（pclk_adc）主要完成 APB 总线操作、SOC 控制、过采样、数据后处理、DMA 请求产生等功能，频率跟随系统主频。

ADC 工作时钟（clk_adc）主要完成模拟电路控制、电压转换控制、数据处理等功能，最高频率为 100MHz。

图24-2 ADC 时钟框架



clk_adc 可通过寄存器 PERI_CRG641[adc_clk_mode]配置选择与 pclk_adc 为同步或异步关系：

- 若选择为同步关系，则 clk_adc 由 pclk_adc 经过寄存器 PERI_CRG640[clk_adc_div1]可配 1/2/3/4 分频得到，避免由异步处理造成触发信号和采样位置之间存在不确定的偏差。
- 若选择为异步关系，则 clk_adc 由所选时钟源经过寄存器 PERI_CRG640[clk_adc_div0]可配 1/2/3/4 分频得到，与系统时钟 pclk_adc 相互独立。clk_adc 异步时钟源有三种输入源，通过 CRG 时钟配置寄存器 PERI_CRG64[clk_pst2_sw_sel]进行选择：
  - PLL 输出时钟，经过寄存器配置分频后时钟。
  - HOSC 时钟（clk_hosc）。
  - 晶振时钟（clk_xtal）。

须知

clk_adc 最高频率为 100MHz，当所选时钟源频率大于 100MHz 时，需要配置 PERI_CRG640[clk_adc_div0]或 PERI_CRG640[clk_adc_div1]，调整分频系数，使得 clk_adc 频率不超过 100MHz。

24.2.3 分辨率

分辨率表示了 ADC 能够转换量化模拟电压的精度。

ADC 的分辨率为 12bit。

在过采样功能下，分辨率可以配置选择 13bit ~ 16bit，过采样结果存储在过采样结果寄存器中，且会按过采样倍数降低 ADC 的采样率。

24.2.4 参考电压

ADC 内置参考电压缓冲器，ADC 输入电压量程为 0V ~ 3.3V，不支持 MCU 外部参考电压。

24.2.5 输入信号

ADC 仅支持单端信号输入，输入信号通过输入管脚 ADCINx 进行采样。

表24-1 ADC 输入信号连接关系

输入通道	连接信号	输入通道	连接信号
ADCIN0	PGA0_OUT	ADCIN10	来自管脚（PD6）
ADCIN1	PGA1_OUT	ADCIN11	来自管脚（PD5）
ADCIN2	来自管脚（PA5）	ADCIN12	来自管脚（PF3）
ADCIN3	来自管脚（PA6）	ADCIN13	来自管脚（PB5）
ADCIN4	来自管脚（PA7）	ADCIN14	来自管脚（PB6）
ADCIN5	来自管脚（PF1）	ADCIN15	来自管脚（PE7）
ADCIN6	来自管脚（PC6）	ADCIN16	TSENSOR_OUT
ADCIN7	来自管脚（PC5）	ADCIN17	DAC_OUT
ADCIN8	来自管脚（PF2）	ADCIN18	AVDD/3
ADCIN9	来自管脚（PD7）	-	-

当采样 TSENSOR_OUT、DAC_OUT 和 AVDD/3 时，需要使用 ADC 最大采样时间。

当采样 AVDD/3 时，需要配置寄存器 ADC_AVDD_EN[cfg_avdd_en]为 1 进行使能。

24.2.6 ADC 时序

SARADC 的工作过程从收到触发采样请求开始，经过触发优先级仲裁、异步处理、采样通道提前建立等采样延迟时间，进入采样窗口，采样完成后进行数据转换和数据处理，最终置位转换结束标志，得到转换结果，表示了完成一次采样转换的过程。

图24-3 ADC 工作时序

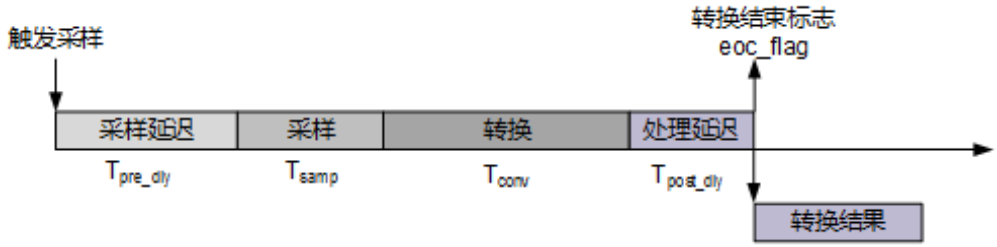


表 24-2 对 ADC 工作过程中各部分时间进行说明。

表24-2 ADC 工作时间描述

工作时间	描述	周期数
T_{pre_dly}	采样延迟时间，表示从触发信号产生，到 ADC 进行采样之间的延迟，包括触发延迟、优先级判决延迟、异步处理延迟、通道切换延迟。	3 个 pclk_adc 周期+11 个 clk_adc 周期 + 约 4 ~ 5 个 clk_adc 周期的异步处理时间（若 pclk_adc 和 clk_adc 为同频同步关系，则异步处理时间为确定的 5 个 clk_adc 周期）。
T_{samp}	采样时间，表示 ADC 的实际采样窗口时间。	ADC_SOCx_CFG[cfg_socx_samptime_sel]选择的 clk_adc 周期。
T_{conv}	转换时间，表示 ADC 进行数据转换的时间。	28 个 clk_adc 周期。
T_{post_dly}	后处理时间，表示 ADC 的数据处理时间。	3 个 clk_adc 周期 + 2 个 pclk_adc 周期 + 约 4 ~ 5 个 clk_adc 周期的异步处理时间（若 pclk_adc 和 clk_adc 为同频同步关系，则异步处理时间为确定的 5 个 clk_adc 周期）。

ADC 采用流水线设计，采样延迟时间 T_{pre_dly}和后处理时间 T_{post_dly}仅影响触发信号到当次采样结果的延迟，并不会降低 ADC 的实际采样率，即可在当次采样处于转换阶段时，产生下一次单次采样触发，ADC 会流水的进行采样过程，使得实际采样率不变。

ADC 的采样率由采样时间和转换时间决定。

典型场景下各采样过程时间如表 24-3 所示。

表24-3 ADC 典型采样率

clk_adc 时钟周期 Tclk_adc	采样时间 T_{samp}	转换时间 T_{conv}	单次采样转换总时间	采样率
10ns	50ns	280ns	330ns	3MSPS
10ns	220ns	280ns	500ns	2MSPS
20ns	440ns	560ns	1000ns	1MSPS

ADC 的外部信号源的驱动能力不同，所需要的采样窗口时间也不同，因此 SOC 可以通过配置寄存器 ADC_SOCx_CFG[cfg_socx_samptime_sel]控制该 SOC 的采样时间。

采样窗口时间分析如下：

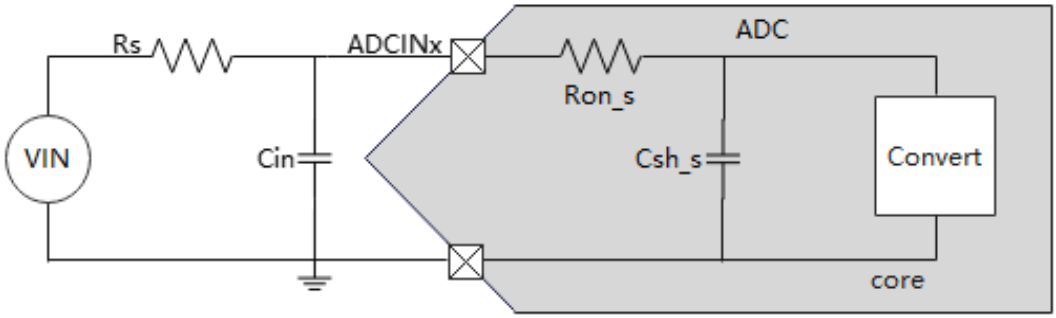
如图 24-4 所示，待采样输入信号 VIN 经过 Rs&Cin 滤波网络进入到 ADC，Cin 为 ADC 输入通道 PAD 处的近端滤波电容（Cin 推荐容值为 100pF）；Ron_s 和 Csh_s 为 ADC 等效采样网络；为了避免 Rs&Cin 滤波网络影响采样，Rs 电阻取值满足如下条件：Rs(Ω)×Cin(nF)<samp_times(ns)/12 如表 24-4 所示。

表24-4 Rs 电阻取值

samp_cycles（clk_adc cycles）	samp_times（clk_adc=100MHz）	Max.Rs（Ω） Rs×Cin<samp_times/12
5	50ns	40Ω
7	70ns	50Ω
10	100ns	80Ω
12	120ns	100Ω
15	150ns	120Ω
22	220ns	180Ω

samp_cycles（clk_adc cycles）	samp_times（clk_adc=100MHz）	Max.Rs（Ω） Rs×Cin<samp_times/12
30	300ns	250Ω
50	500ns	400Ω
75	750ns	600Ω
100	1us	800Ω
125	1.25us	1000Ω
150	1.5us	1200Ω
200	2us	1600Ω
300	3us	2500Ω
400	4us	3300Ω
500	5us	4100Ω

图24-4 ADC 输入模型



ADC_SOCx_CFG[cfg_socx_samptime_sel]控制 ADC 的采样时间，位宽为 4bit，配置值与采样周期选择对应关系如表 24-5 所示。

表24-5 采样时间选择

cfg_socx_samptime_sel	采样时间（clk_adc 周期）
0x0	5
0x1	7

cfg_socx_samptime_sel	采样时间（clk_adc 周期）
0x2	10
0x3	12
0x4	15
0x5	22
0x6	30
0x7	50
0x8	75
0x9	100
0xA	125
0xB	150
0xC	200
0xD	300
0xE	400
0xF	500

24.2.7 采样结果

根据给定的模拟输入电压，对应的理想数字转换结果，如表 24-6 所示。

表24-6 电压转换为 12bit 结果

模拟输入电压（V）	ADC 转换结果（LSB）
ADCINx ≤ 0	adc_resultx = 0
0 < ADCINx < 3.3	adc_resultx = 4096 * ADCINx / 3.3
ADCINx ≥ 3.3	adc_resultx = 4095

例如：输入电压为 2V，则 ADC 理论采样结果为 4096 * 2 / 3.3 = 2482。

根据采样得到的数据结果，对应模拟电压，如表 24-7 所示。

表24-7 12bit 结果转换为电压

ADC 转换结果（LSB）	模拟输入电压（V）
adc_resultx = 0	ADCINx ≤ 0
0 < adc_resultx < 4095	ADCINx = 3.3 * adc_resultx / 4096
adc_resultx = 4095	ADCINx ≥ 3.3

例如：ADC 采样结果为 1500，则对应采样电压为 3.3 * 1500 / 4096 = 1.208V

24.3 启动

MCU 上电 ADC 处于默认关断状态。ADC 的启动流程如下：

步骤 1 使能 ADC 工作前，打开 ADC 时钟门控 PERI_CRG6411[adc_cken]，解除 ADC 复位状态 PERI_CRG641[adc_srst_req]。

步骤 2 配置 ADC 模拟初始化参数 ADC_ANA_CTRL0[cfg_sar_samp_cap_sel]为 0x4。

步骤 3 使能 ADC，配置寄存器 ADC_EN[cfg_adc_en]为 1，等待 100us，ADC 状态稳定后再进行采样触发和结果获取。等待过程中可进行 ADC 寄存器配置。

----结束

24.4 转换模式

本章节描述的转换模式仅针对单个 SOC 行为进行说明，每个 SOC 可独立配置转换模式。当触发多个 SOC 时，请参见 24.6 转换优先级章节进行优先级仲裁后，再进行采样转换。

24.4.1 单次转换模式

单次转换模式下，SOC 由触发源上升沿触发采样，对所选配置通道进行单次采样转换，转换完成后置位 EOC 到寄存器 ADC_EOC_FLAG[eocx_flag]中，并将转换结果置于该 SOC 的结果寄存器 ADC_RESULTx 中。

寄存器 ADC_SOCx_CFG[cfg_socx_cont_en]配置为 0，使能 SOC 工作在单次转换模式。

图24-5 单次转换过程

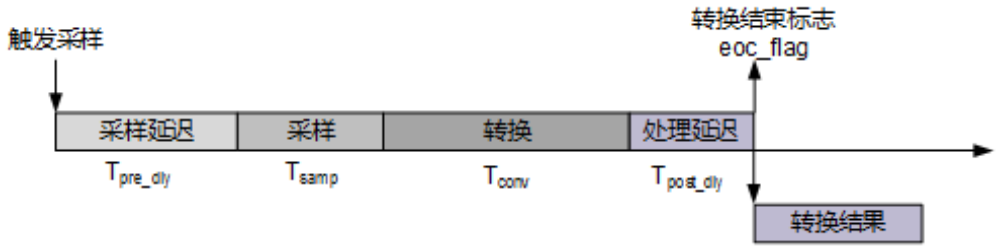
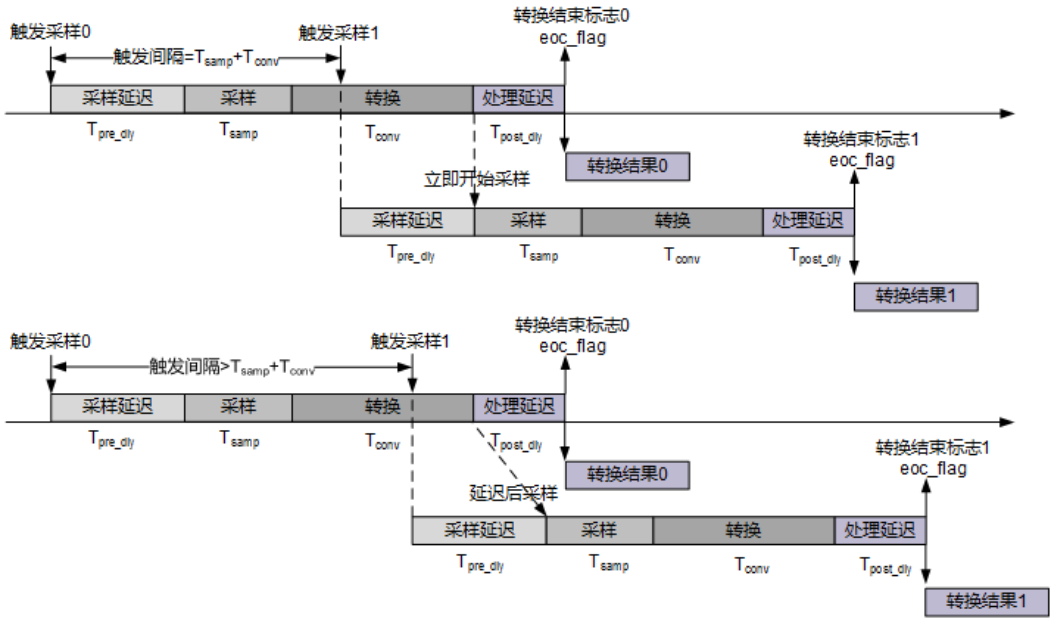


图 24-6 表示了对同一个 SOC 进行两次触发单次转换的过程。两次触发的时间间隔若等于 $T_{samp}+T_{conv}$，则 ADC 会在第一次转换完成时立即进入第二次的采样窗口，实际采样率为 $1/(T_{samp}+T_{conv})$。两次触发的时间间隔若大于 $T_{samp}+T_{conv}$，则 ADC 会在第一次转换完成后，延迟一段时间再进入第二次的采样窗口。两次触发的时间间隔若小于 $T_{samp}+T_{conv}$，则会产生触发溢出，第二次触发请求会被忽略，请参见 24.5.3 SOC 触发溢出标志章节描述。

图24-6 两次触发转换过程



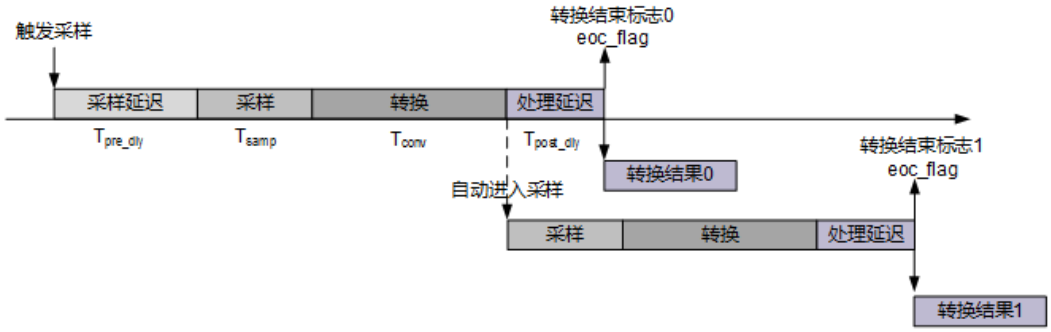
24.4.2 连续转换模式

连续转换模式下，单次触发后，SOC 进行一次采样后，逻辑自动产生下一次采样请求，进行连续采样，并不断刷新转换完成标志和采样转换结果。

寄存器 ADC_SOCx_CFG[cfg_socx_cont_en]配置为 1，使能 SOC 工作在连续转换模式。将寄存器 ADC_SOCx_CFG[cfg_socx_cont_en]配置为 0，则 SOC 切换为单次转换，关闭连续转换模式，完成当前转换后，若没有新的触发，则停止采样。

当 SOC 收到一次触发源触发后，ADC 在采样转换完成后，自动触发当前 SOC 进行下一次采样，不需要额外的等待时间和触发。每次转换处理完成后，会产生转换结束标志 EOC 和转换结果。

图24-7 连续转换过程



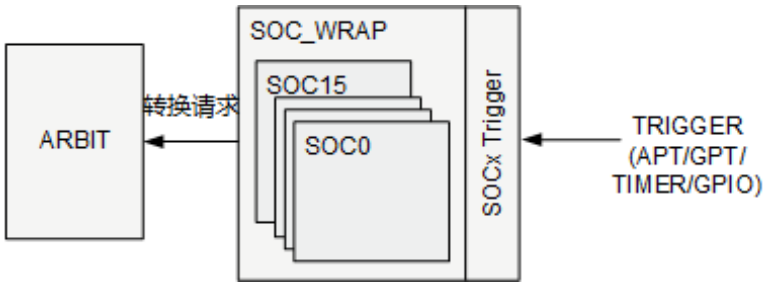
24.5 转换开始模块

ADC 的触发和采样功能是通过可配置的 SOC 进行控制的，每个 SOC 的配置控制一个通道的采样行为。每个 SOC 有独立寄存器进行配置，SOC 的配置包括触发源选择、转换模式选择、转换通道选择以及采样窗口时间配置。

24.5.1 SOC 框架

ADC 共包含有 16 个 SOC，其框架如图 24-8 所示。

图24-8 SOC 框架



24.5.2 触发源配置

每个 SOC 可以选择一个触发源进行触发转换，通过 ADC_SOCx_CFG[cfg_socx_trig_sel]进行配置选择，可以选择的触发源如表 24-8 所示。

表24-8 触发源选择

cfg_socx_trig_sel	触发源	cfg_socx_trig_sel	触发源
0x0	软件触发	0xD	TIMER0
0x1	APT0SOCA	0xE	TIMER1
0x2	APT0SOCB	0xF	TIMER2
0x3	APT1SOCA	0x10	TIMER3
0x4	APT1SOCB	0x11	IO 触发（PD5）
0x5	APT2SOCA	0x12	IO 触发（PF3）
0x6	APT2SOCB	0x13	IO 触发（PF2）
0x7	APT3SOCA	0x14	IO 触发（PF1）
0x8	APT3SOCB	else	软件触发
0x9	GPT0	-	-
0xA	GPT1	-	-
0xB	GPT2	-	-
0xC	GPT3	-	-

软件触发通过写寄存器 ADC_SOFT_TRIG 触发 SOC。

IO 触发通过外部输入信号的上升沿触发 ADC 进行采样。

24.5.3 SOC 触发溢出标志

每个 SOC 有独立的触发溢出标志，在当前 SOC 已有未被响应的触发时，若触发源再次触发该 SOC，则会产生 SOC 触发溢出标志，上报溢出中断到 ADC_ERR_INT 异常中断状态寄存器中。

24.5.4 通道选择

每个 SOC 可以配置采样 19 个输入通道的任意一个通道，通过寄存器 ADC_SOCx_CFG[cfg_socx_ch_sel]进行选择，如表 24-9 所示。

表24-9 模拟输入通道选择

cfg_socx_ch_sel	INPUT	cfg_socx_ch_sel	INPUT
0x0	ADCIN0	0xB	ADCIN11
0x1	ADCIN1	0xC	ADCIN12
0x2	ADCIN2	0xD	ADCIN13
0x3	ADCIN3	0xE	ADCIN14
0x4	ADCIN4	0xF	ADCIN15
0x5	ADCIN5	0x10	ADCIN16
0x6	ADCIN6	0x11	ADCIN17
0x7	ADCIN7	0x12	ADCIN18
0x8	ADCIN8	-	-
0x9	ADCIN9	-	-
0xA	ADCIN10	-	-

24.5.5 转换完成标志

每个 SOC 都有一个对应的转换完成标志 EOC，表示该 SOC 的采样转换完成，可以从对应的结果寄存器中取出数据，EOC 标志可以用于指示读取结果和触发中断。

EOC 标志在转换处理完成时置位，从寄存器 ADC_EOC_FLAG[eocx_flag]中读取，软件读取到该标志有效后，可以从对应的 SOC 结果寄存器中读取转换结果。

EOC 标志寄存器为写 1 清，向对应 bit 位写入 1，可清除 EOC 标志。

EOC 标志可以触发中断，表示转换完成，请参见 24.10 中断章节描述。

24.5.6 转换结果

每个 SOC 有独立的结果寄存器，转换结果存放在各个 SOC 的结果寄存器 ADC_RESULTx[adc_resultx]中。

24.6 转换优先级

ADC 控制器共 16 个 SOC，当多个 SOC 同时请求发起采样时，通过转换优先级来判断响应顺序。ADC 将 SOC 分为两种优先级组：高优先级组和轮询优先级组。默认情况下，所有 SOC 均处于轮询优先级组中。16 个 SOC 根据寄存器 ADC_ARBT0[cfg_soc_priority]划分优先级组，cfg_soc_priority 各 bit 决定了每个 SOC 的优先级。

24.6.1 轮询优先级组

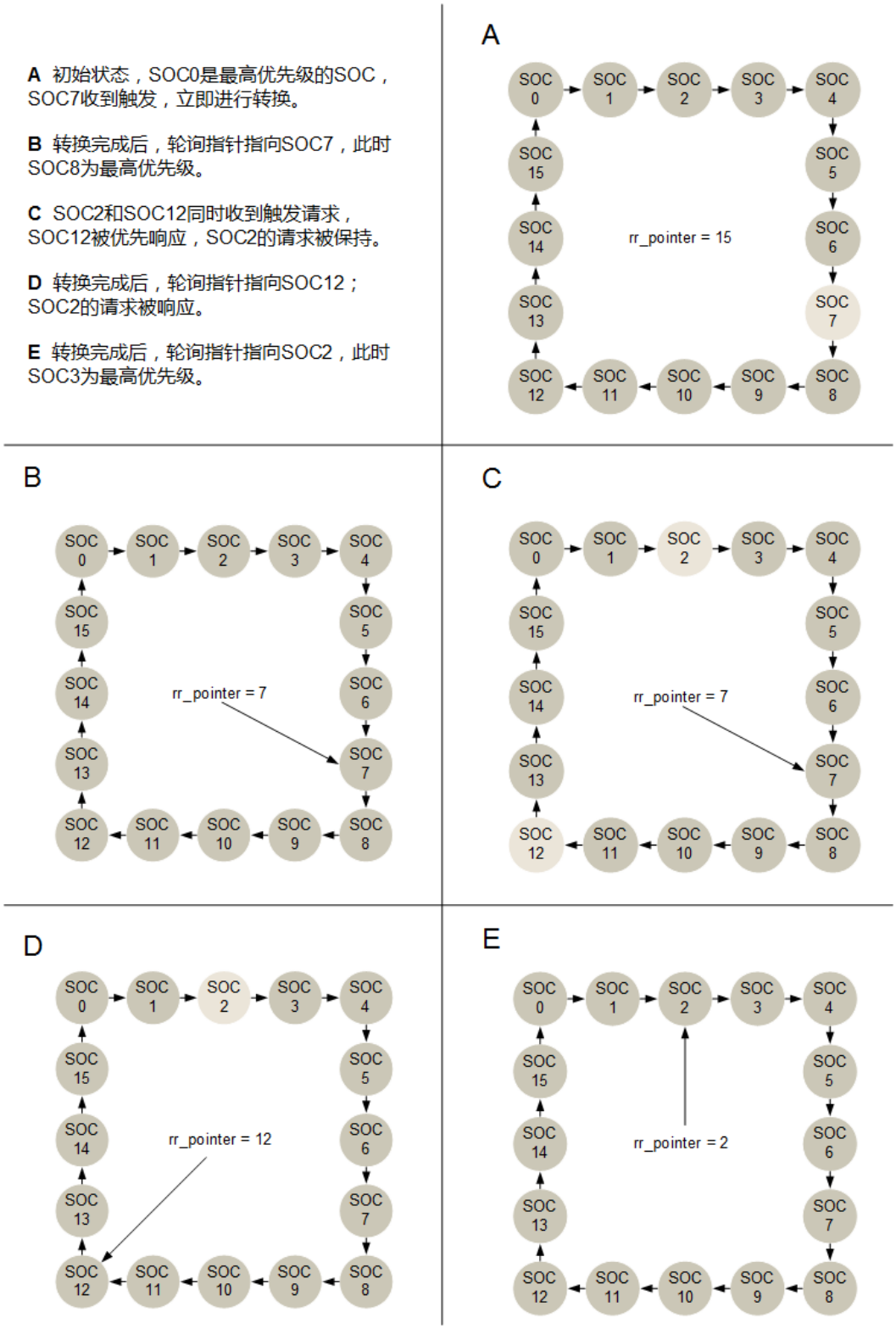
轮询优先级组内所有 SOC 平级，由优先级轮询指针 rr_pointer 反应当前组内的优先级顺序，ADC 在轮询优先级组内，优先响应标号为 rr_pointer+1 的 SOC 采样请求。当轮询指针为 15 时，表示优先响应 SOC0 的采样请求。

轮询指针初始值为 15，每完成一次轮询优先级组内 SOC 转换后，轮询指针指向组内上一个完成转换的 SOC。高优先级组内 SOC 的转换不会影响轮询指针。

轮询指针可以由寄存器 ADC_ARBT1[cfg_rr_pointer_reset]重新置位回 15。当前轮询指针位置可以由寄存器 ADC_ARBT2[rr_pointer]读取。

图 24-9 表示一个轮询优先级组转换例子。

图24-9 轮询优先级组转换



24.6.2 高优先级组

高优先级组的整体优先级高于轮询优先级组，即高优先级组的 SOC 能够插入到轮询优先级组的 SOC 队列中，但不能去打断当前正在采样转换的轮询优先级组 SOC。

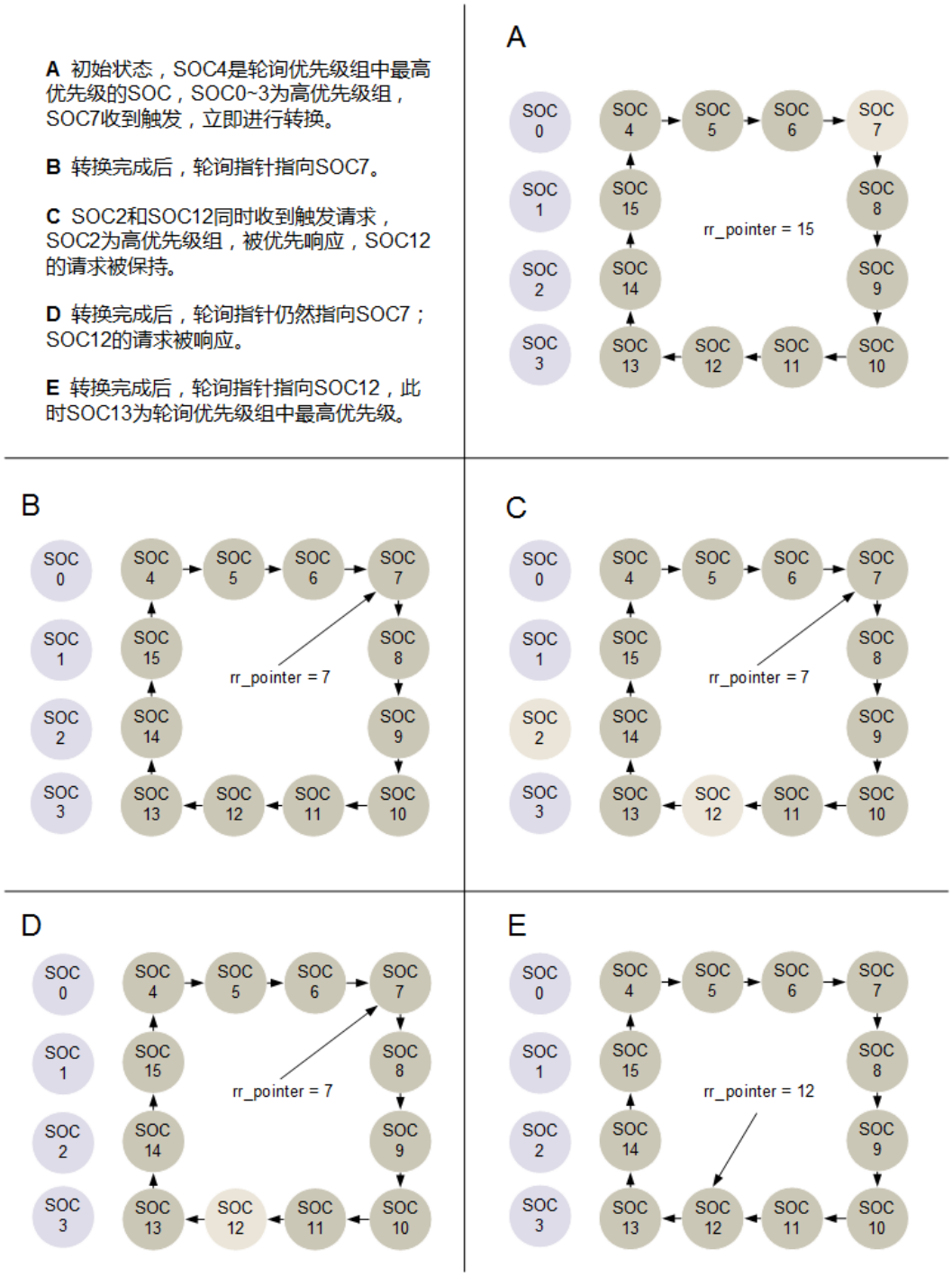
高优先级组内，标号越低的 SOC 优先级越高，即当 SOC0 和 SOC1 均在高优先级组时，SOC0 优先级高于 SOC1。

须知

若高优先级组内的 SOC 配置为连续转换模式，则 ADC 仅会响应高优先级组内标号最小的连续转换模式 SOC 和标号低于该 SOC 的请求，其余 SOC 的采样请求会被阻塞。例如：SOC0，SOC1，SOC4，SOC5 为高优先级组，且 SOC4 和 SOC5 均为连续转换模式，其余 SOC 为轮询优先级组，则 ADC 仅响应 SOC0，SOC1 和 SOC4 的采样请求，其余 SOC 请求均会被阻塞。

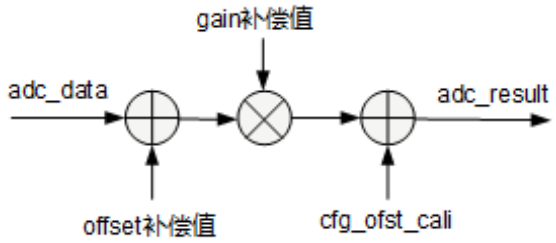
图 24-10 表示一个高优先级组转换例子。

图24-10 高优先级组转换



24.7 数据处理

图24-11 数据处理



MCU 上电后，自动加载 ADC 自身 offset 和 gain 的误差校准值，作用在所有 ADC 通道上。

此外，ADC 提供额外的系统级 offset 补偿，可由用户自定义补偿配置。ADC_PROCESS0[cfg_ofst_cali]配置补偿值，为 12bit 有符号小数，补码表示，bit[0]是小数位。

若用户需要自定义校准值，ADC 提供了三组可配置 offset 和 gain 补偿值参数。每组参数可独立配置作用在一个模拟通道上，替换该通道的默认校准值。

寄存器 ADC_OEGE_CTRL0/1/2 配置三组参数，寄存器 ADC_OEGE_CH_SEL 配置每组参数作用通道。

- ADC_OEGE_CH_SEL[cfg_oege_ch_selx]配置选择当前参数作用的模拟通道。
- ADC_OEGE_CTRLx[cfg_ofstx]为配置 offset 校准值，为 12bit 的有符号小数，补码表示，bit[0]是小数位。例如，offset 校准值配置为 0xFF1，表示 offset 为 -7.5LSB。
- ADC_OEGE_CTRLx[cfg_gainx]为配置 gain 校准值，为 13bit 的无符号小数，bit[11:0]是小数位。

24.8 过采样

ADC 支持对单个 SOC 的采样结果进行过采样处理，将 SOC 的多次采样结果平均，提升 ADC 的采样精度，最大可将采样精度提升至 16bit。

过采样功能根据配置的过采样倍数 N 和平均次数 M，对采样结果按公式处理，得到过采样结果：$\text{oversamp data} = \frac{1}{2^M}\sum_0^{2^N-1} soc\ result$

过采样功能通过寄存器 ADC_OVERSAMP[cfg_oversamp_en]配置使能。

寄存器 ADC_OVERSAMP[cfg_oversamp_soc_sel]配置选择单个 SOC 进行过采样。

过采样倍数 N 通过寄存器 ADC_OVERSAMP[cfg_oversamp_n]配置选择，表示过采样倍数为 2^N 倍，N 可配置范围为 3 ~ 8。

平均次数 M 通过寄存器 ADC_OVERSAMP[cfg_oversamp_m]配置选择，表示平均次数为 2^M，M 可配置范围为 0~8。

不同的过采样倍数 N 和平均次数 M 配置选择，对应最终的采样精度关系如表 24-10 所示。

表24-10 过采样精度

N	M=0	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8
3	15	14	13	12	11	10	9	8	7
4	16	15	14	13	12	11	10	9	8
5	-	16	15	14	13	12	11	10	9
6	-	-	16	15	14	13	12	11	10
7	-	-	-	16	15	14	13	12	11
8	-	-	-	-	16	15	14	13	12

过采样结果存储在过采样结果寄存器 ADC_OVERSAMP_RESULT[oversamp_data]中，过采样结果有效标志通过寄存器 ADC_EVENT_INT[intr_oversamp_data_vld]表示。

对于不同的过采样倍数 N，过采样倍数越大，过采样结果越准确，但累积数据会越多，过采样结果的等效采样率会越低。

过采样模式和选择的 SOC 工作模式联合使用，若选择的 SOC 为单次转换模式，则单次触发自动完成一次过采样（采样 2^N 次），过采样完成之前，该 SOC 不再接收新的触发；若选择的 SOC 为连续转换模式，则单次触发后连续完成过采样，过采样结果会每 2^N 次采样自动刷新一次。

须知

过采样请求也会按照 24.6 转换优先级章节描述的优先级进行仲裁，若过采样 SOC 不为最高优先级，可能会被其他 SOC 打断，导致非等间隔采样。

24.9 后处理模块

ADC 集成了 4 个后处理模块 PPB（Post Processing Block），为采样转换后的数据进行后处理。

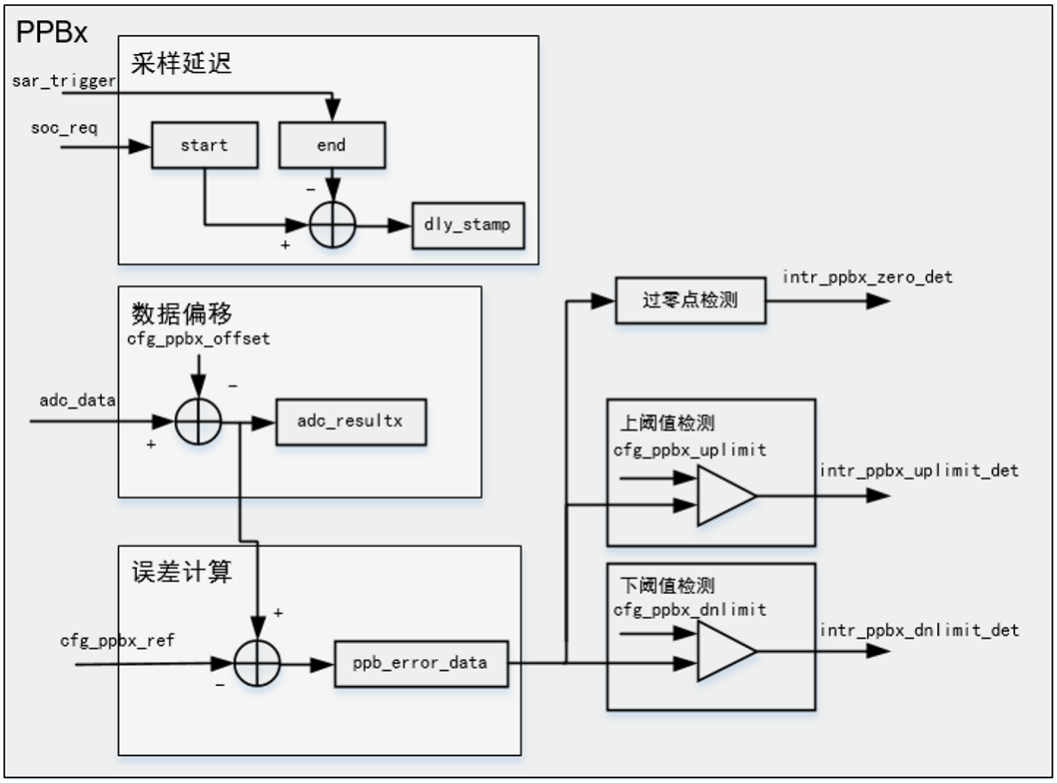
PPB 提供采样延迟记录、数据偏移、误差计算、阈值检测、过零点检测功能。
ADC_PPBx_CTRL0[cfg_ppbx_dly_en]配置采样延时记录使能，
ADC_PPBx_CTRL0[cfg_ppbx_offset_en]配置数据偏移使能，
ADC_PPBx_CTRL0[cfg_ppbx_detect_en]配置误差计算、阈值检测、过零点检测使能。

每个后处理模块可以通过寄存器 ADC_PPBx_CTRL0[cfg_ppbx_soc_sel]单独配置选择 16 个 SOC 中的任意一个进行处理。

24.9.1 PPB 框架

后处理模块框架如图 24-12 所示。

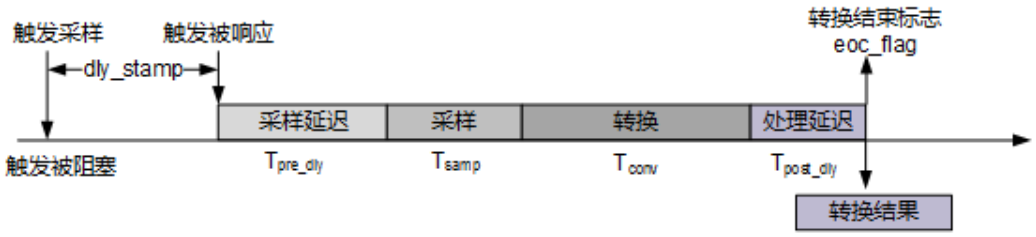
图24-12 后处理模块框架



24.9.2 采样延迟记录

后处理模块记录的采样延迟可以从寄存器 ADC_PPBx_RESULT[ppbx_dly_stamp]获取，记录了所选择的 SOC 从接收触发到触发被响应的延迟。

图24-13 采样延迟时间



24.9.3 数据偏移

若配置使能数据偏移功能，则将 PPB 模块选择的 SOC 的采样结果与配置偏移量（ADC_PPBx_CTRL0[cfg_ppbx_offset]）求和，并将计算偏移后的结果存入该 SOC 的结果寄存器 ADC_PPBx_RESULT[adc_resultx]中。

数据偏移功能可通过寄存器 ADC_PPBx_CTRL0[cfg_ppbx_offset_en]独立配置使能。

可配置偏移量 ADC_PPBx_CTRL0[cfg_ppbx_offset]为 12bit 的有符号整数，补码表示，表示的偏移范围为 -2048 LSB ~ +2047LSB，精度为 1LSB。

24.9.4 误差计算

若配置使能检测功能，则计算 PPB 模块数据偏移后的结果与配置参考值（ADC_PPBx_CTRL2[cfg_ppbx_ref]）之间的误差，利用配置参考值减去数据偏移结果，得到误差结果存储在独立的误差结果寄存器 ADC_PPBx_RESULT[ppbx_error_data]中。

可配置参考值 ADC_PPBx_CTRL2[cfg_ppbx_ref]为 12bit 无符号整数，可配置范围为 0~4095LSB，精度为 1LSB。

误差计算结果 ADC_PPBx_RESULT[ppbx_error_data]为 13bit 的有符号整数，补码表示。

误差计算结果有效标志记录在 ADC_EVENT_INT[intr_ppbx_error_data_vld]寄存器中，可在中断控制中选择是否触发事件中断。

24.9.5 阈值检测

若配置使能检测功能，则将误差计算结果与配置的上下限阈值进行比较，若误差计算结果大于配置上阈值（ADC_PPBx_CTRL1[cfg_ppbx_uplimit]），则置位上阈值超限标志，若误差计算结果小于配置下阈值（ADC_PPBx_CTRL1[cfg_ppbx_dnlimit]），则置位下阈值超限标志。

可配置的上下限阈值均为 13bit 有符号整数，补码表示，可配置范围为 -4096LSB ~ +4095LSB，精度为 1LSB。配置上阈值应大于等于下阈值。

上阈值超限标志记录在 ADC_EVENT_INT[intr_ppbx_uplimit_det]寄存器中，可在中断控制中选择是否触发事件中断。

下阈值超限标志记录在 ADC_EVENT_INT[intr_ppbx_dnlimit_det]寄存器中，可在中断控制中选择是否触发事件中断。

24.9.6 过零点检测

若配置使能检测功能，则将误差计算结果进行过零点检测，若误差计算结果发生正负翻转，则置位过零点检测标志。

过零点检测标志记录在 ADC_EVENT_INT[intr_ppbx_zero_det]寄存器中，可在中断控制中选择是否触发事件中断。

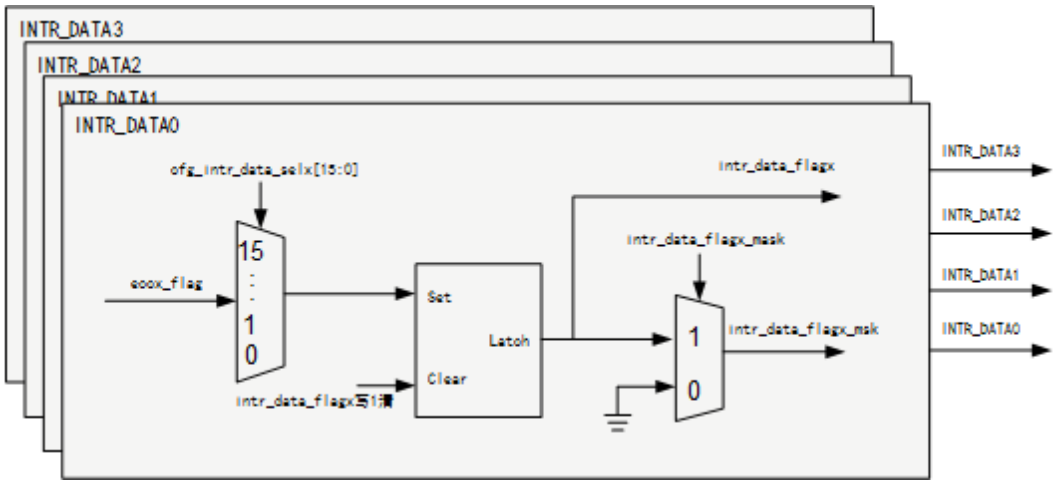
24.10 中断

ADC 模块共有 4 个独立的数据中断、1 个事件中断和 1 个异常中断。

24.10.1 数据中断

数据中断结构如图 24-14 所示。

图24-14 数据中断结构



数据中断状态在中断状态寄存器 ADC_INT_DATA_MSK 中可查询。

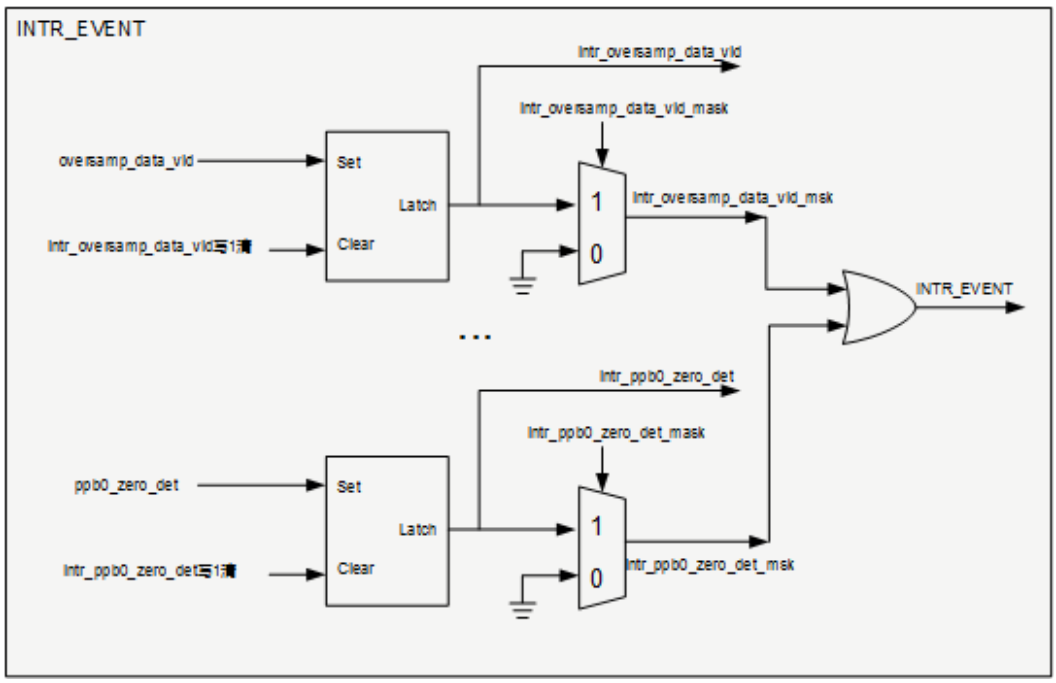
4 个数据中断可以通过中断屏蔽寄存器 ADC_DATA_FLAG_MASK[intr_data_flagx_mask]独立配置屏蔽。

数据中断可以配置 ADC_INT_DATA_0[cfg_intr_data_selx]选择 16 个 SOC 中的任意多个作为触发源。

24.10.2 事件中断

事件中断可指示 ADC 控制器的内部检测事件，事件中断如图 24-15 所示。

图24-15 事件中断结构



事件中断状态通过中断状态寄存器 ADC_EVENT_INT_MSK 中查询。

事件中断中的触发源可通过中断屏蔽寄存器 ADC_EVENT_INT_MASK 中的对应位进行屏蔽。

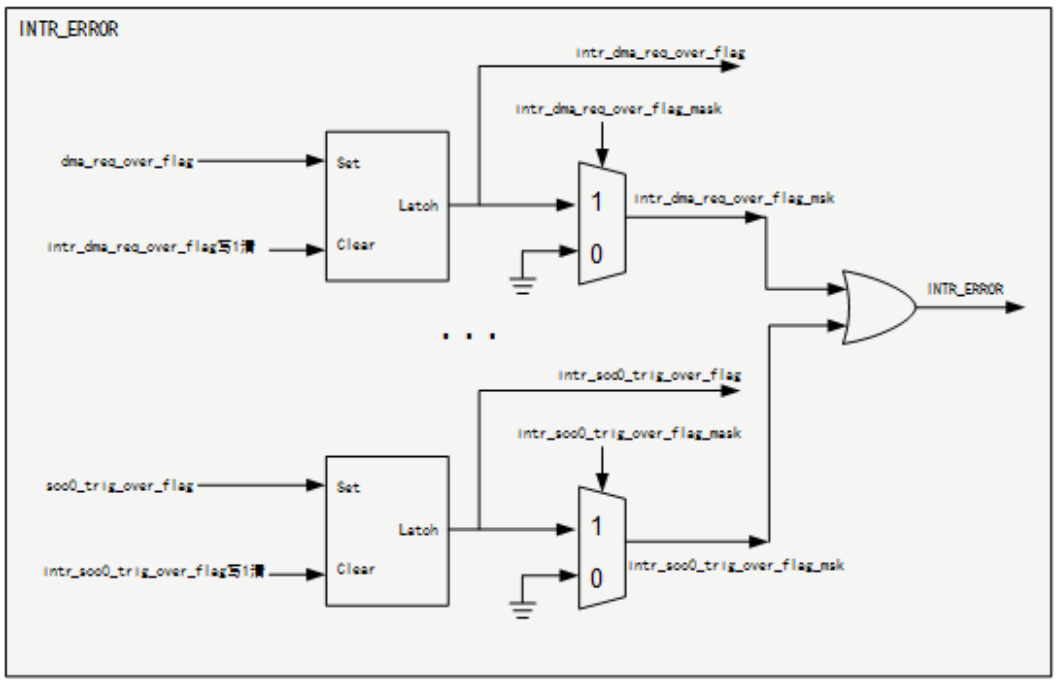
事件中断的触发源包括：

- 后处理模块中的误差计算结果有效标志 ADC_EVENT_INT_MSK[intr_ppbx_error_data_vld_msk]。
- 后处理模块中的上下阈值超限检测 ADC_EVENT_INT_MSK[intr_ppbx_uplimit_det_msk]和 ADC_EVENT_INT_MSK[intr_ppbx_dnlimit_det_msk]。
- 后处理模块中的过零点检测 ADC_EVENT_INT_MSK[intr_ppbx_zero_det_msk]。
- 过采样结果有效标志 ADC_EVENT_INT_MSK[intr_oversamp_data_vld_msk]。
- 校准模式完成标志 ADC_EVENT_INT_MSK[intr_cali_done_msk]。

24.10.3 异常中断

异常中断标志了 ADC 内部工作异常状态，异常中断结构如图 24-16 所示。

图24-16 异常中断结构



异常中断可通过中断状态寄存器 ADC_ERR_INT_MSK 进行查询。

异常中断中的触发源可通过中断屏蔽寄存器 ADC_ERR_INT_MASK 中的对应位进行屏蔽。

异常中断的触发源包括：

- SOC 触发溢出 ADC_ERR_INT_MSK[intr_socx_trig_over_flag_msk]。
- DMA 请求溢出 ADC_ERR_INT_MSK[intr_dma_req_over_flag_msk]。

24.11 DMA 请求

ADC 集成了 1 个 DMA 请求模块，用于上报 DMA 请求，采样结果可以由 DMA 进行数据搬移，DMA 支持 burst 模式和 single 模式。

DMA 请求模块可由 16 个 SOC 中的任意一个在转换完成后触发，通过寄存器 ADC_DMA[cfg_dma_soc_sel]进行配置选择。

DMA 上报的请求类型支持 BURST 和 SINGLE 两种请求模式，BURST 模式由 ADC_DMA[cfg_dma_brst_req_sel]寄存器配置使能，SINGLE 模式由 ADC_DMA[cfg_dma_sing_req_sel]寄存器配置使能。

当现有的 DMA 请求未被响应清除时，发起新的 DMA 请求，则会产生 DMA 请求溢出标志 ADC_ERR_INT[intr_dma_req_over_flag]。

24.12 温度传感器

ADC 的输入通道 ADCIN16 固定连接温度传感器（Tsensor），温度传感器将温度转化成电压，两者成线性关系，通过测量传感器的输出电压就可以知道当前器件的结温。

表24-11 温度传感器参数

测量温度范围	温度系数 k	温度精度（校准后）
-40℃ ~ +125℃	4.23mV/℃	±8℃

通过 ADC 获取温度结果步骤如下：

步骤 1 将 TSENSOR_CTRL_REG[cfg_tsensor_pd]设置为 0，使能温度传感器。

步骤 2 等待至少 40us 温度传感器从掉电模式唤醒的启动时间。

步骤 3 将 SOC 配置到 ADCIN16 通道后启动 A/D 转换。

步骤 4 当 A/D 转换完成时。读取 ADC 采样结果。

步骤 5 推荐采样 16 次，取平均值做为最终结果 $V_{tsen_vptat_n}$。

步骤 6 使用以下公式计算温度

温度系数 k=0.00423

$$T_n = \frac{V_{tsen_vptat_n} - 1.32}{k} + 25$$

待测温度

其中： $V_{tsen_vptat_n}$ 是在温度为 T_n 时 Tsensor 输出电压。

----结束

24.13 典型场景配置

24.13.1 APT 触发单次转换

单次转换以 SOC1 由 APT0SOCA 触发源触发采样 PGA1_OUT 通道为例，采样周期选择为 5 个 clk，配置流程如下：

步骤 1 参考 24.3 启动章节进行 ADC 启动配置，配置 ADC 使能后的等待时间中可进行后续步骤初始化配置和采样配置。

步骤 2 配置 SOC1 的控制寄存器 ADC_SOC1_CFG。

- cfg_soc1_cont_en 配置为，选择单次转换模式。
- cfg_soc1_trig_sel 配置为，选择 APT0SOCA 为触发源。
- cfg_soc1_samptime_sel 配置为，选择采样周期为 5 个周期。
- cfg_soc1_ch_sel 配置为，选择采样 PAG1_OUT 通道。

步骤 3 可选择是否将 EOC 标志绑定到数据中断，配置中断寄存器 ADC_INT_DATA_0 或 ADC_INT_DATA_1，选择将 eoc1_flag 绑定到对应数据中断。

步骤 4 通过 APT0 产生触发信号。

步骤 5 读取转换完成标志寄存器 ADC_EOC_FLAG[eoc1_flag]，结果为 1 时读取结果寄存器 ADC_RESULT1[adc_result1]，得到采样结果。

----结束

24.13.2 软件触发连续转换

连续转换以 SOC1 由软件触发源触发采样 PGA1_OUT 通道为例，采样周期选择为 5 个 clk，配置流程如下：

步骤 1 参考 24.3 启动章节进行 ADC 启动配置，配置 ADC 使能后的等待时间中可进行后续步骤初始化配置和采样配置。

步骤 2 配置 SOC1 的控制寄存器 ADC_SOC1_CFG。

- cfg_soc1_cont_en 配置为，选择连续转换模式。
- cfg_soc1_trig_sel 配置为，选择软件触发源。
- cfg_soc1_samptime_sel 配置为，选择采样周期为 5 个周期。
- cfg_soc1_ch_sel 配置为，选择采样 PAG1_OUT 通道。

步骤 3 可选择是否将 EOC 标志绑定到数据中断，配置中断寄存器 ADC_INT_DATA_0 或 ADC_INT_DATA_1，选择将 eoc1_flag 绑定到对应数据中断。

步骤 4 配置软件触发寄存器 ADC_SOFT_TRIG[cfg_soc1_soft_trig]为 1，产生触发信号。

步骤 5 读取转换完成标志寄存器 ADC_EOC_FLAG[eoc1_flag]，结果为 1 时读取结果寄存器 ADC_RESULT1[adc_result1]，得到采样结果。

----结束

24.13.3 APT 触发过采样

以 SOC3 由 APT1SOCB 触发源触发过采样 ADCIN6 通道为例，采样周期选择为 5 个 clk，过采样倍数选择为 16 倍，过采样精度截取为 15bit。配置流程如下：

步骤 1 参考 24.3 启动章节进行 ADC 启动配置，配置 ADC 使能后的等待时间中可进行后续步骤初始化配置和采样配置。

步骤 2 配置 SOC3 的控制寄存器 ADC_SOC3_CFG。

- cfg_soc3_cont_en 配置为，选择单次转换模式。
- cfg_soc3_trig_sel 配置为，选择 APT1SOCB 为触发源。
- cfg_soc3_samptime_sel 配置为，选择采样周期为 5 个周期。
- cfg_soc3_ch_sel 配置为，选择采样 ADCIN6 通道。

步骤 3 配置过采样寄存器 ADC_OVERSAMP。

- cfg_oversamp_m 配置为 0x1，截取精度为 15bit。
- cfg_oversamp_n 配置为 0x4，过采样倍数为 16 倍。
- cfg_oversamp_soc_sel 配置为 0x3，选择 SOC3。
- cfg_oversamp_en 配置为 1，使能过采样功能。

步骤 4 通过 APT1 产生触发信号。

步骤 5 读取转换完成标志寄存器 ADC_EVENT_INT[intr_oversamp_data_vld]，结果为 1 时读取结果寄存器 ADC_OVERSAMP_RESULT[oversamp_data]，得到采样结果。

----结束

24.13.4 TIMER 触发多个 SOC 采样

单个触发源可触发多个 SOC 对不同通道进行采样，且不同通道的采样窗口可根据不同输入信号的驱动能力进行配置。

以 TIMER0 触发 SOC2/SOC5/SOC7，分别采样 PGA0_OUT/ADCIN7/Tsensor 通道为例，采样周期分别选择为 5/10/500 个 clk，配置流程如下：

步骤 1 参考 24.3 启动章节进行 ADC 启动配置，配置 ADC 使能后的等待时间中可进行后续步骤初始化配置和采样配置。

步骤 2 配置 SOC2 的控制寄存器 ADC_SOC2_CFG。

- cfg_soc2_cont_en 配置为，选择单次转换模式。
- cfg_soc2_trig_sel 配置为，选择 TIMER0 为触发源。
- cfg_soc2_samptime_sel 配置为，选择采样周期为 5 个周期。
- cfg_soc2_ch_sel 配置为，选择采样 PAG0_OUT 通道。

步骤 3 配置 SOC5 的控制寄存器 ADC_SOC5_CFG。

- cfg_soc5_cont_en 配置为，选择单次转换模式。
- cfg_soc5_trig_sel 配置为，选择 TIMER0 为触发源。
- cfg_soc5_samptime_sel 配置为，选择采样周期为 10 个周期。
- cfg_soc5_ch_sel 配置为，选择采样 ADCIN7 通道。

步骤 4 配置 SOC7 的控制寄存器 ADC_SOC7_CFG。

- cfg_soc7_cont_en 配置为，选择单次转换模式。
- cfg_soc7_trig_sel 配置为，选择 TIMER0 为触发源。
- cfg_soc7_samptime_sel 配置为，选择采样周期为 500 个周期。
- cfg_soc7_ch_sel 配置为，选择采样 Tsensor 通道。

步骤 5 可选择是否将 EOC 标志绑定到数据中断，配置中断寄存器 ADC_INT_DATA_0 或 ADC_INT_DATA_1，选择将 eoc2_flag/eoc5_flag/eoc7_flag 绑定到对应数据中断。

步骤 6 通过 TIMER0 产生触发信号。

步骤 7 读取转换完成标志寄存器 ADC_EOC_FLAG[eocx_flag]，结果为 1 时读取结果寄存器 ADC_RESULTx[adc_resultx]，得到采样结果。

----结束

24.14 ADC_SYS_REG 寄存器概览

ADC_SYS_REG 寄存器概览如表 24-12 所示。

表24-12 ADC_SYS_REG 寄存器概览（基址是 0x1800_0000）

偏移地址	名称	描述	页码
0x0000	ADC_RESULT0	ADC 结果寄存器 0	ADC_RESULT0
0x0004	ADC_RESULT1	ADC 结果寄存器 1	ADC_RESULT1
0x0008	ADC_RESULT2	ADC 结果寄存器 2	ADC_RESULT2
0x000C	ADC_RESULT3	ADC 结果寄存器 3	ADC_RESULT3
0x0010	ADC_RESULT4	ADC 结果寄存器 4	ADC_RESULT4
0x0014	ADC_RESULT5	ADC 结果寄存器 5	ADC_RESULT5
0x0018	ADC_RESULT6	ADC 结果寄存器 6	ADC_RESULT6
0x001C	ADC_RESULT7	ADC 结果寄存器 7	ADC_RESULT7
0x0020	ADC_RESULT8	ADC 结果寄存器 8	ADC_RESULT8
0x0024	ADC_RESULT9	ADC 结果寄存器 9	ADC_RESULT9
0x0028	ADC_RESULT10	ADC 结果寄存器 10	ADC_RESULT10
0x002C	ADC_RESULT11	ADC 结果寄存器 11	ADC_RESULT11

偏移地址	名称	描述	页码
0x0030	ADC_RESULT12	ADC 结果寄存器 12	ADC_RESULT12
0x0034	ADC_RESULT13	ADC 结果寄存器 13	ADC_RESULT13
0x0038	ADC_RESULT14	ADC 结果寄存器 14	ADC_RESULT14
0x003C	ADC_RESULT15	ADC 结果寄存器 15	ADC_RESULT15
0x0070	ADC_EOC_FLAG	ADC eoc 状态寄存器	ADC_EOC_FLAG
0x0100	ADC_SOC0_CFG	ADC SOC0 配置寄存器	ADC_SOC0_CFG
0x0104	ADC_SOC1_CFG	ADC SOC1 配置寄存器	ADC_SOC1_CFG
0x0108	ADC_SOC2_CFG	ADC SOC2 配置寄存器	ADC_SOC2_CFG
0x010C	ADC_SOC3_CFG	ADC SOC3 配置寄存器	ADC_SOC3_CFG
0x0110	ADC_SOC4_CFG	ADC SOC4 配置寄存器	ADC_SOC4_CFG
0x0114	ADC_SOC5_CFG	ADC SOC5 配置寄存器	ADC_SOC5_CFG
0x0118	ADC_SOC6_CFG	ADC SOC6 配置寄存器	ADC_SOC6_CFG
0x011C	ADC_SOC7_CFG	ADC SOC7 配置寄存器	ADC_SOC7_CFG
0x0120	ADC_SOC8_CFG	ADC SOC8 配置寄存器	ADC_SOC8_CFG

偏移地址	名称	描述	页码
0x0124	ADC_SOC9_CFG	ADC SOC9 配置寄存器	ADC_SOC9_CFG
0x0128	ADC_SOC10_CFG	ADC SOC10 配置寄存器	ADC_SOC10_CFG
0x012C	ADC_SOC11_CFG	ADC SOC11 配置寄存器	ADC_SOC11_CFG
0x0130	ADC_SOC12_CFG	ADC SOC12 配置寄存器	ADC_SOC12_CFG
0x0134	ADC_SOC13_CFG	ADC SOC13 配置寄存器	ADC_SOC13_CFG
0x0138	ADC_SOC14_CFG	ADC SOC14 配置寄存器	ADC_SOC14_CFG
0x013C	ADC_SOC15_CFG	ADC SOC15 配置寄存器	ADC_SOC15_CFG
0x0160	ADC_SOFT_TRIG	ADC 软件触发寄存器	ADC_SOFT_TRIG
0x0200	ADC_ARBT0	ADC 优先级寄存器 0	ADC_ARBT0
0x0204	ADC_ARBT1	ADC 优先级寄存器 1	ADC_ARBT1
0x0208	ADC_ARBT2	ADC 优先级寄存器 2	ADC_ARBT2
0x0220	ADC_OVERSAMP	ADC 过采样配置寄存器	ADC_OVERSAMP
0x0224	ADC_OVERSAMP_RESULT	ADC 过采样结果寄存器	ADC_OVERSAMP_RESULT
0x0250	ADC_PPB0_CTRL0	ADC 后处理模块 0 配置寄存器 0	ADC_PPB0_CTRL0
0x0254	ADC_PPB0_CTRL1	ADC 后处理模块 0 配置寄存器 1	ADC_PPB0_C

偏移地址	名称	描述	页码
			TRL1
0x0258	ADC_PPB0_CTRL2	ADC 后处理模块 0 配置寄存器 2	ADC_PPB0_CTRL2
0x025C	ADC_PPB0_RESULT	ADC 后处理模块 0 结果寄存器	ADC_PPB0_RESULT
0x0260	ADC_PPB1_CTRL0	ADC 后处理模块 1 配置寄存器 0	ADC_PPB1_CTRL0
0x0264	ADC_PPB1_CTRL1	ADC 后处理模块 1 配置寄存器 1	ADC_PPB1_CTRL1
0x0268	ADC_PPB1_CTRL2	ADC 后处理模块 1 配置寄存器 2	ADC_PPB1_CTRL2
0x026C	ADC_PPB1_RESULT	ADC 后处理模块 1 结果寄存器	ADC_PPB1_RESULT
0x0270	ADC_PPB2_CTRL0	ADC 后处理模块 2 配置寄存器 0	ADC_PPB2_CTRL0
0x0274	ADC_PPB2_CTRL1	ADC 后处理模块 2 配置寄存器 1	ADC_PPB2_CTRL1
0x0278	ADC_PPB2_CTRL2	ADC 后处理模块 2 配置寄存器 2	ADC_PPB2_CTRL2
0x027C	ADC_PPB2_RESULT	ADC 后处理模块 2 结果寄存器	ADC_PPB2_RESULT
0x0280	ADC_PPB3_CTRL0	ADC 后处理模块 3 配置寄存器 0	ADC_PPB3_CTRL0
0x0284	ADC_PPB3_CTRL1	ADC 后处理模块 3 配置寄存器 1	ADC_PPB3_CTRL1
0x0288	ADC_PPB3_CTRL2	ADC 后处理模块 3 配置寄存器 2	ADC_PPB3_CTRL2
0x028C	ADC_PPB3_RESULT	ADC 后处理模块 3 结果寄存器	ADC_PPB3_R

偏移地址	名称	描述	页码
			ESULT
0x02B0	ADC_INT_DATA_0	ADC 数据中断配置寄存器 0	ADC_INT_DATA_0
0x02B4	ADC_INT_DATA_1	ADC 数据中断配置寄存器 1	ADC_INT_DATA_1
0x02B8	ADC_INT_DATA_FLAG	ADC 数据中断原始状态寄存器	ADC_INT_DATA_FLAG
0x02BC	ADC_INT_DATA_MSK	ADC 数据中断屏蔽后状态寄存器	ADC_INT_DATA_MSK
0x02C0	ADC_DATA_FLAG_MASK	ADC 数据中断屏蔽寄存器	ADC_DATA_FLAG_MASK
0x02C4	ADC_EVENT_INT	ADC 事件中断原始状态寄存器	ADC_EVENT_INT
0x02C8	ADC_EVENT_INT_MSK	ADC 事件中断屏蔽后状态寄存器	ADC_EVENT_INT_MSK
0x02CC	ADC_EVENT_INT_MASK	ADC 事件中断屏蔽寄存器	ADC_EVENT_INT_MASK
0x02D0	ADC_ERR_INT	ADC 异常中断原始状态寄存器	ADC_ERR_INT
0x02D4	ADC_ERR_INT_MSK	ADC 异常中断屏蔽后状态寄存器	ADC_ERR_INT_MSK
0x02D8	ADC_ERR_INT_MASK	ADC 异常中断屏蔽寄存器	ADC_ERR_INT_MASK
0x02F0	ADC_DMA	ADC DMA 配置寄存器	ADC_DMA
0x0300	ADC_EN	ADC 使能寄存器	ADC_E

偏移地址	名称	描述	页码
			N
0x0310	ADC_EN_DLY	ADC 使能延迟寄存器	ADC_EN_DLY
0x0400	ADC_MODE	ADC 工作模式配置寄存器	ADC_MODE
0x0420	ADC_OEGE_CH_SEL	ADC OE GE 通道配置寄存器	ADC_OEGE_CH_SEL
0x0440	ADC_OEGE_CTRL0	ADC OE GE 配置寄存器 0	ADC_OEGE_CTRL0
0x0444	ADC_OEGE_CTRL1	ADC OE GE 配置寄存器 1	ADC_OEGE_CTRL1
0x0448	ADC_OEGE_CTRL2	ADC OE GE 配置寄存器 2	ADC_OEGE_CTRL2
0x0460	ADC_PROCESS0	ADC 数据处理寄存器 0	ADC_PROCESS0
0x0464	ADC_PROCESS1	ADC 数据处理寄存器 1	ADC_PROCESS1
0x0480	ADC_STATUS	ADC 状态寄存器	ADC_STATUS
0x0500	ADC_WEIGHT_CFG15	ADC 电容权重配置寄存器	ADC_WEIGHT_CFG15
0x0504	ADC_WEIGHT_CFG14	ADC 电容权重配置寄存器	ADC_WEIGHT_CFG14
0x0508	ADC_WEIGHT_CFG13	ADC 电容权重配置寄存器	ADC_WEIGHT_CFG13
0x050C	ADC_WEIGHT_CFG12	ADC 电容权重配置寄存器	ADC_WEIGHT_CFG12
0x0510	ADC_WEIGHT_CFG11	ADC 电容权重配置寄存器	ADC_WEIGH

偏移地址	名称	描述	页码
			T_CFG 11
0x0514	ADC_WEIGHT_CFG10	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 10
0x0518	ADC_WEIGHT_CFG9	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 9
0x051C	ADC_WEIGHT_CFG8	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 8
0x0520	ADC_WEIGHT_CFG7	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 7
0x0524	ADC_WEIGHT_CFG6	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 6
0x0528	ADC_WEIGHT_CFG5	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 5
0x052C	ADC_WEIGHT_CFG4	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 4
0x0530	ADC_WEIGHT_CFG3	ADC 电容权重配置寄存器	ADC_ WEIGH T_CFG 3
0x0540	ADC_CAP_TRG	ADC 电容校准触发寄存器	ADC_C AP_TR G
0x0550	ADC_CAP_M1	ADC 电容校准模式 1 寄存器	ADC_C AP_M1
0x0654	ADC_ANA_CTRL0	ADC 模拟寄存器 0	ADC_A NA_CT RL0
0x0658	ADC_AVDD_EN	AVDD 通道使能寄存器	ADC_A VDD_E

偏移地址	名称	描述	页码
			N
0x0800	ADC_TSENSOR_TRIM	Tsensor trim 寄存器	ADC_TSENSOR_TRIM
0x0804	ADC_OEGE_TRIM	ADC OE GE TRIM 寄存器	ADC_OEGE_TRIM
0x0810	ADC_PGA0_OEGE_TRIM0	ADC PGA0 OE GE TRIM 寄存器 0	ADC_PGA0_OEGE_TRIM0
0x0814	ADC_PGA0_OEGE_TRIM1	ADC PGA0 OE GE TRIM 寄存器 1	ADC_PGA0_OEGE_TRIM1
0x0818	ADC_PGA0_OEGE_TRIM2	ADC PGA0 OE GE TRIM 寄存器 2	ADC_PGA0_OEGE_TRIM2
0x081C	ADC_PGA0_OEGE_TRIM3	ADC PGA0 OE GE TRIM 寄存器 3	ADC_PGA0_OEGE_TRIM3
0x0820	ADC_PGA1_OEGE_TRIM0	ADC PGA1 OE GE TRIM 寄存器 0	ADC_PGA1_OEGE_TRIM0
0x0824	ADC_PGA1_OEGE_TRIM1	ADC PGA1 OE GE TRIM 寄存器 1	ADC_PGA1_OEGE_TRIM1
0x0828	ADC_PGA1_OEGE_TRIM2	ADC PGA1 OE GE TRIM 寄存器 2	ADC_PGA1_OEGE_TRIM2
0x082C	ADC_PGA1_OEGE_TRIM3	ADC PGA1 OE GE TRIM 寄存器 3	ADC_PGA1_OEGE_TRIM3
0x0840	ADC_ANA_TRIM	ADC 模拟 TRIM 寄存器	ADC_ANA_TRIM

24.15 ADC_SYS_REG 寄存器描述

ADC_RESULT0

ADC_RESULT0 为 ADC 结果寄存器 0。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result0	SOC0 结果寄存器。	0x000

ADC_RESULT1

ADC_RESULT1 为 ADC 结果寄存器 1。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result1	SOC1 结果寄存器。	0x000

ADC_RESULT2

ADC_RESULT2 为 ADC 结果寄存器 2。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result2	SOC2 结果寄存器。	0x000

ADC_RESULT3

ADC_RESULT3 为 ADC 结果寄存器 3。

Offset Address: 0x000C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result3	SOC3 结果寄存器。	0x000

ADC_RESULT4

ADC_RESULT4 为 ADC 结果寄存器 4。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result4	SOC4 结果寄存器。	0x000

ADC_RESULT5

ADC_RESULT5 为 ADC 结果寄存器 5。

Offset Address: 0x0014 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result5	SOC5 结果寄存器。	0x000

ADC_RESULT6

ADC_RESULT6 为 ADC 结果寄存器 6。

Offset Address: 0x0018 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result6	SOC6 结果寄存器。	0x000

ADC_RESULT7

ADC_RESULT7 为 ADC 结果寄存器 7。

Offset Address: 0x001C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result7	SOC7 结果寄存器。	0x000

ADC_RESULT8

ADC_RESULT8 为 ADC 结果寄存器 8。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result8	SOC8 结果寄存器。	0x000

ADC_RESULT9

ADC_RESULT9 为 ADC 结果寄存器 9。

Offset Address: 0x0024 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result9	SOC9 结果寄存器。	0x000

ADC_RESULT10

ADC_RESULT10 为 ADC 结果寄存器 10。

Offset Address: 0x0028 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000

Bits	Access	Name	Description	Reset
[11:0]	RO	adc_result10	SOC10 结果寄存器。	0x000

ADC_RESULT11

ADC_RESULT11 为 ADC 结果寄存器 11。

Offset Address: 0x002C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result11	SOC11 结果寄存器。	0x000

ADC_RESULT12

ADC_RESULT12 为 ADC 结果寄存器 12。

Offset Address: 0x0030 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result12	SOC12 结果寄存器。	0x000

ADC_RESULT13

ADC_RESULT13 为 ADC 结果寄存器 13。

Offset Address: 0x0034 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result13	SOC13 结果寄存器。	0x000

ADC_RESULT14

ADC_RESULT14 为 ADC 结果寄存器 14。

Offset Address: 0x0038 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result14	SOC14 结果寄存器。	0x000

ADC_RESULT15

ADC_RESULT15 为 ADC 结果寄存器 15。

Offset Address: 0x003C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RO	adc_result15	SOC15 结果寄存器。	0x000

ADC_EOC_FLAG

ADC_EOC_FLAG 为 ADC eoc 状态寄存器。

Offset Address: 0x0070 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	RW1C	eoc15_flag	SOC15 的 EOC 标志寄存器。	0x0
[14]	RW1C	eoc14_flag	SOC14 的 EOC 标志寄存器。	0x0
[13]	RW1C	eoc13_flag	SOC13 的 EOC 标志寄存器。	0x0
[12]	RW1C	eoc12_flag	SOC12 的 EOC 标志寄存器。	0x0
[11]	RW1C	eoc11_flag	SOC11 的 EOC 标志寄存器。	0x0
[10]	RW1C	eoc10_flag	SOC10 的 EOC 标志寄存器。	0x0
[9]	RW1C	eoc9_flag	SOC9 的 EOC 标志寄存器。	0x0
[8]	RW1C	eoc8_flag	SOC8 的 EOC 标志寄存器。	0x0
[7]	RW1C	eoc7_flag	SOC7 的 EOC 标志寄存器。	0x0

Bits	Access	Name	Description	Reset
[6]	RW1C	eoc6_flag	SOC6 的 EOC 标志寄存器。	0x0
[5]	RW1C	eoc5_flag	SOC5 的 EOC 标志寄存器。	0x0
[4]	RW1C	eoc4_flag	SOC4 的 EOC 标志寄存器。	0x0
[3]	RW1C	eoc3_flag	SOC3 的 EOC 标志寄存器。	0x0
[2]	RW1C	eoc2_flag	SOC2 的 EOC 标志寄存器。	0x0
[1]	RW1C	eoc1_flag	SOC1 的 EOC 标志寄存器。	0x0
[0]	RW1C	eoc0_flag	SOC0 的 EOC 标志寄存器。	0x0

ADC_SOC0_CFG

ADC_SOC0_CFG 为 ADC SOC0 配置寄存器。

Offset Address: 0x0100 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc0_cont_en	SOC0 连续转换模式使能。 0：不使能； 1：使能。	0x0
[13:9]	RW	cfg_soc0_trig_sel	SOC0 触发源选择。 0x00：软件触发； 0x01：APT0SOCA； 0x02：APT0SOCB； 0x03：APT1SOCA； 0x04：APT1SOCB； 0x05：APT2SOCA； 0x06：APT2SOCB； 0x07：APT3SOCA； 0x08：APT3SOCB；	0x00

Bits	Access	Name	Description	Reset
			0x09：GPT0； 0x0A：GPT1； 0x0B：GPT2； 0x0C：GPT3； 0x0D：TIMER0； 0x0E：TIMER1； 0x0F：TIMER2； 0x10：TIMER3； 0x11：GPIO； 0x12：GPIO； 0x13：GPIO； 0x14：GPIO； 其他：软件触发。	
[8:5]	RW	cfg_soc0_samptime_sel	SOC0 采样周期选择。 0x0：5clk； 0x1：7clk； 0x2：10clk； 0x3：12clk； 0x4：15clk； 0x5：22clk； 0x6：30clk； 0x7：50clk； 0x8：75clk； 0x9：100clk； 0xA：125clk； 0xB：150clk； 0xC：200clk； 0xD：300clk； 0xE：400clk；	0x0

Bits	Access	Name	Description	Reset
			0xF：500clk。	
[4:0]	RW	cfg_soc0_ch_sel	SOC0 通道选择。 0x00：PGA0_OUT； 0x01：PGA1_OUT； 0x02：管脚输入； 0x03：管脚输入； 0x04：管脚输入； 0x05：管脚输入； 0x06：管脚输入； 0x07：管脚输入； 0x08：管脚输入； 0x09：管脚输入； 0x0A：管脚输入； 0x0B：管脚输入； 0x0C：管脚输入； 0x0D：管脚输入； 0x0E：管脚输入； 0x0F：管脚输入； 0x10：TSENSOR_OUT； 0x11：DAC_OUT； 0x12：AVDD/3； 其他：未使用。	0x00

ADC_SOC1_CFG

ADC_SOC1_CFG 为 ADC SOC1 配置寄存器。

Offset Address: 0x0104 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000

Bits	Access	Name	Description	Reset
[14]	RW	cfg_soc1_cont_en	SOC1 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc1_trig_sel	SOC1 触发源选择。	0x00
[8:5]	RW	cfg_soc1_samptime_sel	SOC1 采样周期选择。	0x0
[4:0]	RW	cfg_soc1_ch_sel	SOC1 通道选择。	0x00

ADC_SOC2_CFG

ADC_SOC2_CFG 为 ADC SOC2 配置寄存器。

Offset Address: 0x0108 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc2_cont_en	SOC2 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc2_trig_sel	SOC2 触发源选择。	0x00
[8:5]	RW	cfg_soc2_samptime_sel	SOC2 采样周期选择。	0x0
[4:0]	RW	cfg_soc2_ch_sel	SOC2 通道选择。	0x00

ADC_SOC3_CFG

ADC_SOC3_CFG 为 ADC SOC3 配置寄存器。

Offset Address: 0x010C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc3_cont_en	SOC3 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc3_trig_sel	SOC3 触发源选择。	0x00
[8:5]	RW	cfg_soc3_samptime_sel	SOC3 采样周期选择。	0x0
[4:0]	RW	cfg_soc3_ch_sel	SOC3 通道选择。	0x00

ADC_SOC4_CFG

ADC_SOC4_CFG 为 ADC SOC4 配置寄存器。

Offset Address: 0x0110 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc4_cont_en	SOC4 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc4_trig_sel	SOC4 触发源选择。	0x00
[8:5]	RW	cfg_soc4_samptime_sel	SOC4 采样周期选择。	0x0
[4:0]	RW	cfg_soc4_ch_sel	SOC4 通道选择。	0x00

ADC_SOC5_CFG

ADC_SOC5_CFG 为 ADC SOC5 配置寄存器。

Offset Address: 0x0114 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc5_cont_en	SOC5 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc5_trig_sel	SOC5 触发源选择。	0x00
[8:5]	RW	cfg_soc5_samptime_sel	SOC5 采样周期选择。	0x0
[4:0]	RW	cfg_soc5_ch_sel	SOC5 通道选择。	0x00

ADC_SOC6_CFG

ADC_SOC6_CFG 为 ADC SOC6 配置寄存器。

Offset Address: 0x0118 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc6_cont_en	SOC6 连续转换模式使能。	0x0

Bits	Access	Name	Description	Reset
[13:9]	RW	cfg_soc6_trig_sel	SOC6 触发源选择。	0x00
[8:5]	RW	cfg_soc6_samptime_sel	SOC6 采样周期选择。	0x0
[4:0]	RW	cfg_soc6_ch_sel	SOC6 通道选择。	0x00

ADC_SOC7_CFG

ADC_SOC7_CFG 为 ADC SOC7 配置寄存器。

Offset Address: 0x011C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc7_cont_en	SOC7 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc7_trig_sel	SOC7 触发源选择。	0x00
[8:5]	RW	cfg_soc7_samptime_sel	SOC7 采样周期选择。	0x0
[4:0]	RW	cfg_soc7_ch_sel	SOC7 通道选择。	0x00

ADC_SOC8_CFG

ADC_SOC8_CFG 为 ADC SOC8 配置寄存器。

Offset Address: 0x0120 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc8_cont_en	SOC8 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc8_trig_sel	SOC8 触发源选择。	0x00
[8:5]	RW	cfg_soc8_samptime_sel	SOC8 采样周期选择。	0x0
[4:0]	RW	cfg_soc8_ch_sel	SOC8 通道选择。	0x00

ADC_SOC9_CFG

ADC_SOC9_CFG 为 ADC SOC9 配置寄存器。

Offset Address: 0x0124 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc9_cont_en	SOC9 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc9_trig_sel	SOC9 触发源选择。	0x00
[8:5]	RW	cfg_soc9_samptime_sel	SOC9 采样周期选择。	0x0
[4:0]	RW	cfg_soc9_ch_sel	SOC9 通道选择。	0x00

ADC_SOC10_CFG

ADC_SOC10_CFG 为 ADC SOC10 配置寄存器。

Offset Address: 0x0128 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc10_cont_en	SOC10 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc10_trig_sel	SOC10 触发源选择。	0x00
[8:5]	RW	cfg_soc10_samptime_sel	SOC10 采样周期选择。	0x0
[4:0]	RW	cfg_soc10_ch_sel	SOC10 通道选择。	0x00

ADC_SOC11_CFG

ADC_SOC11_CFG 为 ADC SOC11 配置寄存器。

Offset Address: 0x012C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc11_cont_en	SOC11 连续转换模式使能。	0x0

Bits	Access	Name	Description	Reset
[13:9]	RW	cfg_soc11_trig_sel	SOC11 触发源选择。	0x00
[8:5]	RW	cfg_soc11_samptime_sel	SOC11 采样周期选择。	0x0
[4:0]	RW	cfg_soc11_ch_sel	SOC11 通道选择。	0x00

ADC_SOC12_CFG

ADC_SOC12_CFG 为 ADC SOC12 配置寄存器。

Offset Address: 0x0130 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc12_cont_en	SOC12 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc12_trig_sel	SOC12 触发源选择。	0x00
[8:5]	RW	cfg_soc12_samptime_sel	SOC12 采样周期选择。	0x0
[4:0]	RW	cfg_soc12_ch_sel	SOC12 通道选择。	0x00

ADC_SOC13_CFG

ADC_SOC13_CFG 为 ADC SOC13 配置寄存器。

Offset Address: 0x0134 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc13_cont_en	SOC13 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc13_trig_sel	SOC13 触发源选择。	0x00
[8:5]	RW	cfg_soc13_samptime_sel	SOC13 采样周期选择。	0x0
[4:0]	RW	cfg_soc13_ch_sel	SOC13 通道选择。	0x00

ADC_SOC14_CFG

ADC_SOC14_CFG 为 ADC SOC14 配置寄存器。

Offset Address: 0x0138 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc14_cont_en	SOC14 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc14_trig_sel	SOC14 触发源选择。	0x00
[8:5]	RW	cfg_soc14_samptime_sel	SOC14 采样周期选择。	0x0
[4:0]	RW	cfg_soc14_ch_sel	SOC14 通道选择。	0x00

ADC_SOC15_CFG

ADC_SOC15_CFG 为 ADC SOC15 配置寄存器。

Offset Address: 0x013C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14]	RW	cfg_soc15_cont_en	SOC15 连续转换模式使能。	0x0
[13:9]	RW	cfg_soc15_trig_sel	SOC15 触发源选择。	0x00
[8:5]	RW	cfg_soc15_samptime_sel	SOC15 采样周期选择。	0x0
[4:0]	RW	cfg_soc15_ch_sel	SOC15 通道选择。	0x00

ADC_SOFT_TRIG

ADC_SOFT_TRIG 为 ADC 软件触发寄存器。

Offset Address: 0x0160 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15]	W1_PULSE	cfg_soc15_soft_trig	SOC15 软件触发。	0x0

Bits	Access	Name	Description	Reset
			0：不触发; 1：触发脉冲。	
[14]	W1_PULSE	cfg_soc14_soft_trig	SOC14 软件触发。	0x0
[13]	W1_PULSE	cfg_soc13_soft_trig	SOC13 软件触发。	0x0
[12]	W1_PULSE	cfg_soc12_soft_trig	SOC12 软件触发。	0x0
[11]	W1_PULSE	cfg_soc11_soft_trig	SOC11 软件触发。	0x0
[10]	W1_PULSE	cfg_soc10_soft_trig	SOC10 软件触发。	0x0
[9]	W1_PULSE	cfg_soc9_soft_trig	SOC9 软件触发。	0x0
[8]	W1_PULSE	cfg_soc8_soft_trig	SOC8 软件触发。	0x0
[7]	W1_PULSE	cfg_soc7_soft_trig	SOC7 软件触发。	0x0
[6]	W1_PULSE	cfg_soc6_soft_trig	SOC6 软件触发。	0x0
[5]	W1_PULSE	cfg_soc5_soft_trig	SOC5 软件触发。	0x0
[4]	W1_PULSE	cfg_soc4_soft_trig	SOC4 软件触发。	0x0
[3]	W1_PULSE	cfg_soc3_soft_trig	SOC3 软件触发。	0x0
[2]	W1_PULSE	cfg_soc2_soft_trig	SOC2 软件触发。	0x0
[1]	W1_PULSE	cfg_soc1_soft_trig	SOC1 软件触发。	0x0
[0]	W1_PULSE	cfg_soc0_soft_trig	SOC0 软件触发。	0x0

ADC_ARBT0

ADC_ARBT0 为 ADC 优先级寄存器 0。

Offset Address: 0x0200 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	cfg_soc_priority	SOC 优先级组配置（配置对应 bit 位的 SOC 为高优先级组）。	0x0000

Bits	Access	Name	Description	Reset
			0：轮询优先级组； 1：高优先级组。	

ADC_ARBT1

ADC_ARBT1 为 ADC 优先级寄存器 1。

Offset Address: 0x0204 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	W1_PULSE	cfg_rr_pointer_reset	轮询指针重置。 0：不重置； 1：重置指针到 15。	0x0

ADC_ARBT2

ADC_ARBT2 为 ADC 优先级寄存器 2。

Offset Address: 0x0208 Total Reset Value: 0x0000_000F

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3:0]	RO	rr_pointer	轮询指针位置。	0xF

ADC_OVERSAMP

ADC_OVERSAMP 为 ADC 过采样配置寄存器。

Offset Address: 0x0220 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:12]	RW	cfg_oversamp_m	过采样截取精度选择，与 cfg_oversamp_n 联合配置，配置方式如表 24-10 所示。	0x0

Bits	Access	Name	Description	Reset
[11:8]	RW	cfg_oversamp_n	过采样累积倍数选择。 0x3：8 倍； 0x4：16 倍； 0x5：32 倍； 0x6：64 倍； 0x7：128 倍； 0x8：256 倍； 其他：保留。	0x0
[7:4]	RW	cfg_oversamp_soc_sel	过采样 SOC 选择。 0x0：SOC0； 0x1：SOC1； ... 0xF：SOC15。	0x0
[3:1]	RO	reserved	保留。	0x0
[0]	RW	cfg_oversamp_en	过采样使能。 0：不使能； 1：使能。	0x0

ADC_OVERSAMP_RESULT

ADC_OVERSAMP_RESULT 为 ADC 过采样结果寄存器。

Offset Address: 0x0224 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RO	oversamp_data	过采样结果，低位（12bit ~ 16bit）为有效精度。	0x0000

ADC_PPB0_CTRL0

ADC_PPB0_CTRL0 为 ADC 后处理模块 0 配置寄存器 0。

Offset Address: 0x0250 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:20]	RW	cfg_ppb0_offset	后处理补偿 offset 值（有符号整数，补码表示）。	0x000
[19:8]	RO	reserved	保留。	0x000
[7:4]	RW	cfg_ppb0_soc_sel	后处理模块 SOC 选择。 0x0：SOC0； 0x1：SOC1； ... 0xF：SOC15。	0x0
[3]	RW	cfg_ppb0_detect_en	后处理阈值检测使能。 0：不使能； 1：使能。	0x0
[2]	RW	cfg_ppb0_offset_en	后处理 offset 补偿使能。 0：不使能； 1：使能。	0x0
[1]	RW	cfg_ppb0_dly_en	后处理采样延迟计数使能。 0：不使能； 1：使能。	0x0
[0]	RO	reserved	保留。	0x0

ADC_PPB0_CTRL1

ADC_PPB0_CTRL1 为 ADC 后处理模块 0 配置寄存器 1。

Offset Address: 0x0254 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00

Bits	Access	Name	Description	Reset
[25:13]	RW	cfg_ppb0_uplimit	后处理误差检测上限阈值（有符号整数，补码表示）。	0x0000
[12:0]	RW	cfg_ppb0_dnlimit	后处理误差检测下限阈值（有符号整数，补码表示）。	0x0000

ADC_PPB0_CTRL2

ADC_PPB0_CTRL2 为 ADC 后处理模块 0 配置寄存器 2。

Offset Address: 0x0258 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_ppb0_ref	后处理误差参考值（无符号整数）。	0x800

ADC_PPB0_RESULT

ADC_PPB0_RESULT 为 ADC 后处理模块 0 结果寄存器。

Offset Address: 0x025C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	ppb0_dly_stamp	后处理采样延迟计数值。	0x0000
[15:13]	RO	reserved	保留。	0x0
[12:0]	RO	ppb0_error_data	后处理误差计算结果（有符号整数，补码表示）。	0x0000

ADC_PPB1_CTRL0

ADC_PPB1_CTRL0 为 ADC 后处理模块 1 配置寄存器 0。

Offset Address: 0x0260 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:20]	RW	cfg_ppb1_offset	后处理补偿 offset 值（有符号整数，补	0x000

Bits	Access	Name	Description	Reset
			码表示）。	
[19:8]	RO	reserved	保留。	0x000
[7:4]	RW	cfg_ppb1_soc_sel	后处理模块 SOC 选择。 0x0：SOC0； 0x1：SOC1； ... 0xF：SOC15。	0x0
[3]	RW	cfg_ppb1_detect_en	后处理阈值检测使能。 0：不使能； 1：使能。	0x0
[2]	RW	cfg_ppb1_offset_en	后处理 offset 补偿使能。 0：不使能； 1：使能。	0x0
[1]	RW	cfg_ppb1_dly_en	后处理采样延迟计数使能。 0：不使能； 1：使能。	0x0
[0]	RO	reserved	保留。	0x0

ADC_PPB1_CTRL1

ADC_PPB1_CTRL1 为 ADC 后处理模块 1 配置寄存器 1。

Offset Address: 0x0264 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25:13]	RW	cfg_ppb1_uplimit	后处理误差检测上限阈值（有符号整数，补码表示）。	0x0000
[12:0]	RW	cfg_ppb1_dnlimit	后处理误差检测下限阈值（有符号整数，补码表示）。	0x0000

ADC_PPB1_CTRL2

ADC_PPB1_CTRL2 为 ADC 后处理模块 1 配置寄存器 2。

Offset Address: 0x0268 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_ppb1_ref	后处理误差参考值（无符号整数）。	0x800

ADC_PPB1_RESULT

ADC_PPB1_RESULT 为 ADC 后处理模块 1 结果寄存器。

Offset Address: 0x026C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	ppb1_dly_stamp	后处理采样延迟计数值。	0x0000
[15:13]	RO	reserved	保留。	0x0
[12:0]	RO	ppb1_error_data	后处理误差计算结果（有符号整数，补码表示）。	0x0000

ADC_PPB2_CTRL0

ADC_PPB2_CTRL0 为 ADC 后处理模块 2 配置寄存器 0。

Offset Address: 0x0270 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:20]	RW	cfg_ppb2_offset	后处理补偿 offset 值（有符号整数，补码表示）。	0x000
[19:8]	RO	reserved	保留。	0x000
[7:4]	RW	cfg_ppb2_soc_sel	后处理模块 SOC 选择。 0x0：SOC0； 0x1：SOC1；	0x0

Bits	Access	Name	Description	Reset
			... 0xF：SOC15。	
[3]	RW	cfg_ppb2_detect_en	后处理阈值检测使能。 0：不使能； 1：使能。	0x0
[2]	RW	cfg_ppb2_offset_en	后处理 offset 补偿使能。 0：不使能； 1：使能。	0x0
[1]	RW	cfg_ppb2_dly_en	后处理采样延迟计数使能。 0：不使能； 1：使能。	0x0
[0]	RO	reserved	保留。	0x0

ADC_PPB2_CTRL1

ADC_PPB2_CTRL1 为 ADC 后处理模块 2 配置寄存器 1。

Offset Address: 0x0274 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25:13]	RW	cfg_ppb2_uplimit	后处理误差检测上限阈值（有符号整数，补码表示）。	0x0000
[12:0]	RW	cfg_ppb2_dnlimit	后处理误差检测下限阈值（有符号整数，补码表示）。	0x0000

ADC_PPB2_CTRL2

ADC_PPB2_CTRL2 为 ADC 后处理模块 2 配置寄存器 2。

Offset Address: 0x0278 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_ppb2_ref	后处理误差参考值（无符号整数）。	0x800

ADC_PPB2_RESULT

ADC_PPB2_RESULT 为 ADC 后处理模块 2 结果寄存器。

Offset Address: 0x027C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	ppb2_dly_stamp	后处理采样延迟计数值。	0x0000
[15:13]	RO	reserved	保留。	0x0
[12:0]	RO	ppb2_error_data	后处理误差计算结果（有符号整数，补码表示）。	0x0000

ADC_PPB3_CTRL0

ADC_PPB3_CTRL0 为 ADC 后处理模块 3 配置寄存器 0。

Offset Address: 0x0280 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:20]	RW	cfg_ppb3_offset	后处理补偿 offset 值（有符号整数，补码表示）。	0x000
[19:8]	RO	reserved	保留。	0x000
[7:4]	RW	cfg_ppb3_soc_sel	后处理模块 SOC 选择。 0x0：SOC0； 0x1：SOC1； ... 0xF：SOC15。	0x0
[3]	RW	cfg_ppb3_detect_en	后处理阈值检测使能。 0：不使能；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	
[2]	RW	cfg_ppb3_offset_en	后处理 offset 补偿使能。 0：不使能； 1：使能。	0x0
[1]	RW	cfg_ppb3_dly_en	后处理采样延迟计数使能。 0：不使能； 1：使能。	0x0
[0]	RO	reserved	保留。	0x0

ADC_PPB3_CTRL1

ADC_PPB3_CTRL1 为 ADC 后处理模块 3 配置寄存器 2。

Offset Address: 0x0284 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25:13]	RW	cfg_ppb3_uplimit	后处理误差检测上限阈值（有符号整数，补码表示）。	0x0000
[12:0]	RW	cfg_ppb3_dnlimit	后处理误差检测下限阈值（有符号整数，补码表示）。	0x0000

ADC_PPB3_CTRL2

ADC_PPB3_CTRL2 为 ADC 后处理模块 3 配置寄存器 2。

Offset Address: 0x0288 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_ppb3_ref	后处理误差参考值（无符号整数）。	0x800

ADC_PPB3_RESULT

ADC_PPB3_RESULT 为 ADC 后处理模块 3 结果寄存器。

Offset Address: 0x028C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RO	ppb3_dly_stamp	后处理采样延迟计数值。	0x0000
[15:13]	RO	reserved	保留。	0x0
[12:0]	RO	ppb3_error_data	后处理误差计算结果（有符号整数，补码表示）。	0x0000

ADC_INT_DATA_0

ADC_INT_DATA_0 为 ADC 数据中断配置寄存器 0。

Offset Address: 0x02B0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RW	cfg_intr_data_sel1	数据中断 1 的 SOC 选择。	0x0000
[15:0]	RW	cfg_intr_data_sel0	数据中断 0 的 SOC 选择。	0x0000

ADC_INT_DATA_1

ADC_INT_DATA_1 为 ADC 数据中断配置寄存器 1。

Offset Address: 0x02B4 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	RW	cfg_intr_data_sel3	数据中断 3 的 SOC 选择。	0x0000
[15:0]	RW	cfg_intr_data_sel2	数据中断 2 的 SOC 选择。	0x0000

ADC_INT_DATA_FLAG

ADC_INT_DATA_FLAG 为 ADC 数据中断原始状态寄存器。

Offset Address: 0x02B8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3]	RW1C	intr_data_flag3	数据中断原始状态寄存器。	0x0
[2]	RW1C	intr_data_flag2	数据中断原始状态寄存器。	0x0
[1]	RW1C	intr_data_flag1	数据中断原始状态寄存器。	0x0
[0]	RW1C	intr_data_flag0	数据中断原始状态寄存器。	0x0

ADC_INT_DATA_MSK

ADC_INT_DATA_MSK 为 ADC 数据中断屏蔽后状态寄存器。

Offset Address: 0x02BC Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3]	RO	intr_data_flag3_msk	数据中断屏蔽后状态寄存器。	0x0
[2]	RO	intr_data_flag2_msk	数据中断屏蔽后状态寄存器。	0x0
[1]	RO	intr_data_flag1_msk	数据中断屏蔽后状态寄存器。	0x0
[0]	RO	intr_data_flag0_msk	数据中断屏蔽后状态寄存器。	0x0

ADC_DATA_FLAG_MASK

ADC_DATA_FLAG_MASK 为 ADC 数据中断屏蔽寄存器。

Offset Address: 0x02C0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:4]	RO	reserved	保留。	0x0000000
[3]	RW	intr_data_flag3_mask	数据中断屏蔽寄存器。 0：屏蔽中断; 1：不屏蔽。	0x0
[2]	RW	intr_data_flag2_mask	数据中断屏蔽寄存器。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽中断; 1：不屏蔽。	
[1]	RW	intr_data_flag1_mask	数据中断屏蔽寄存器。 0：屏蔽中断; 1：不屏蔽。	0x0
[0]	RW	intr_data_flag0_mask	数据中断屏蔽寄存器。 0：屏蔽中断; 1：不屏蔽。	0x0

ADC_EVENT_INT

ADC_EVENT_INT 为 ADC 事件中断原始状态寄存器。

Offset Address: 0x02C4 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17]	RW1C	intr_cali_done	模式 1 校准完成中断原始状态寄存器。	0x0
[16]	RW1C	intr_oversamp_data_vld	过采样数据有效中断原始状态寄存器。	0x0
[15]	RW1C	intr_ppb3_error_data_vld	PPB3 误差计算数据有效中断原始状态寄存器。	0x0
[14]	RW1C	intr_ppb3_dnlimit_det	PPB3 下阈值超限中断原始状态寄存器。	0x0
[13]	RW1C	intr_ppb3_uplimit_det	PPB3 上阈值超限中断原始状态寄存器。	0x0
[12]	RW1C	intr_ppb3_zero_det	PPB3 过零点中断原始状态寄存器。	0x0
[11]	RW1C	intr_ppb2_error_data_vld	PPB2 误差计算数据有效中断原始状态寄存器。	0x0
[10]	RW1C	intr_ppb2_dnlimit_det	PPB2 下阈值超限中断原始状态寄存器。	0x0
[9]	RW1C	intr_ppb2_uplimit_	PPB2 上阈值超限中断原始状态寄存器。	0x0

Bits	Access	Name	Description	Reset
		det		
[8]	RW1C	intr_ppb2_zero_det	PPB2 过零点中断原始状态寄存器。	0x0
[7]	RW1C	intr_ppb1_error_data_vld	PPB1 误差计算数据有效中断原始状态寄存器。	0x0
[6]	RW1C	intr_ppb1_dnlimit_det	PPB1 下阈值超限中断原始状态寄存器。	0x0
[5]	RW1C	intr_ppb1_uplimit_det	PPB1 上阈值超限中断原始状态寄存器。	0x0
[4]	RW1C	intr_ppb1_zero_det	PPB1 过零点中断原始状态寄存器。	0x0
[3]	RW1C	intr_ppb0_error_data_vld	PPB0 误差计算数据有效中断原始状态寄存器。	0x0
[2]	RW1C	intr_ppb0_dnlimit_det	PPB0 下阈值超限中断原始状态寄存器。	0x0
[1]	RW1C	intr_ppb0_uplimit_det	PPB0 上阈值超限中断原始状态寄存器。	0x0
[0]	RW1C	intr_ppb0_zero_det	PPB0 过零点中断原始状态寄存器。	0x0

ADC_EVENT_INT_MSK

ADC_EVENT_INT_MSK 为 ADC 事件中断屏蔽后状态寄存器。

Offset Address: 0x02C8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17]	RO	intr_cali_done_msk	模式 1 校准完成中断屏蔽后状态寄存器。	0x0
[16]	RO	intr_oversamp_data_vld_msk	过采样数据有效中断屏蔽后状态寄存器。	0x0
[15]	RO	intr_ppb3_error_data_vld_msk	PPB3 误差计算数据有效中断屏蔽后状态寄存器。	0x0
[14]	RO	intr_ppb3_dnlimit_det_msk	PPB3 下阈值超限中断屏蔽后状态寄存器。	0x0

Bits	Access	Name	Description	Reset
[13]	RO	intr_ppb3_uplimit_det_msk	PPB3 上阈值超限中断屏蔽后状态寄存器。	0x0
[12]	RO	intr_ppb3_zero_det_msk	PPB3 过零点中断屏蔽后状态寄存器。	0x0
[11]	RO	intr_ppb2_error_data_vld_msk	PPB2 误差计算数据有效中断屏蔽后状态寄存器。	0x0
[10]	RO	intr_ppb2_dnlimit_det_msk	PPB2 下阈值超限中断屏蔽后状态寄存器。	0x0
[9]	RO	intr_ppb2_uplimit_det_msk	PPB2 上阈值超限中断屏蔽后状态寄存器。	0x0
[8]	RO	intr_ppb2_zero_det_msk	PPB2 过零点中断屏蔽后状态寄存器。	0x0
[7]	RO	intr_ppb1_error_data_vld_msk	PPB1 误差计算数据有效中断屏蔽后状态寄存器。	0x0
[6]	RO	intr_ppb1_dnlimit_det_msk	PPB1 下阈值超限中断屏蔽后状态寄存器。	0x0
[5]	RO	intr_ppb1_uplimit_det_msk	PPB1 上阈值超限中断屏蔽后状态寄存器。	0x0
[4]	RO	intr_ppb1_zero_det_msk	PPB1 过零点中断屏蔽后状态寄存器。	0x0
[3]	RO	intr_ppb0_error_data_vld_msk	PPB0 误差计算数据有效中断屏蔽后状态寄存器。	0x0
[2]	RO	intr_ppb0_dnlimit_det_msk	PPB0 下阈值超限中断屏蔽后状态寄存器。	0x0
[1]	RO	intr_ppb0_uplimit_det_msk	PPB0 上阈值超限中断屏蔽后状态寄存器。	0x0
[0]	RO	intr_ppb0_zero_det_msk	PPB0 过零点中断屏蔽后状态寄存器。	0x0

ADC_EVENT_INT_MASK

ADC_EVENT_INT_MASK 为 ADC 事件中断屏蔽寄存器。

Offset Address: 0x02CC Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17]	RW	intr_cali_done_mask	模式 1 校准完成中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[16]	RW	intr_oversamp_data_vld_mask	过采样数据有效中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[15]	RW	intr_ppb3_error_data_vld_mask	PPB3 误差计算数据有效中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[14]	RW	intr_ppb3_dnlimit_det_mask	PPB3 下阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[13]	RW	intr_ppb3_uplimit_det_mask	PPB3 上阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[12]	RW	intr_ppb3_zero_det_mask	PPB3 过零点中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[11]	RW	intr_ppb2_error_data_vld_mask	PPB2 误差计算数据有效中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0

Bits	Access	Name	Description	Reset
[10]	RW	intr_ppb2_dnlimit_det_mask	PPB2 下阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[9]	RW	intr_ppb2_uplimit_det_mask	PPB2 上阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[8]	RW	intr_ppb2_zero_det_mask	PPB2 过零点中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[7]	RW	intr_ppb1_error_data_vld_mask	PPB1 误差计算数据有效中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[6]	RW	intr_ppb1_dnlimit_det_mask	PPB1 下阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[5]	RW	intr_ppb1_uplimit_det_mask	PPB1 上阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[4]	RW	intr_ppb1_zero_det_mask	PPB1 过零点中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[3]	RW	intr_ppb0_error_data_vld_mask	PPB0 误差计算数据有效中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[2]	RW	intr_ppb0_dnlimit_det_mask	PPB0 下阈值超限中断屏蔽寄存器。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽中断； 1：不屏蔽。	
[1]	RW	intr_ppb0_uplimit_det_mask	PPB0 上阈值超限中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[0]	RW	intr_ppb0_zero_det_mask	PPB0 过零点中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0

ADC_ERR_INT

ADC_ERR_INT 为 ADC 异常中断原始状态寄存器。

Offset Address: 0x02D0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW1C	intr_dma_req_over_flag	DMA 请求溢出中断原始状态寄存器。	0x0
[15]	RW1C	intr_soc15_trig_over_flag	SOC15 触发溢出中断原始状态寄存器。	0x0
[14]	RW1C	intr_soc14_trig_over_flag	SOC14 触发溢出中断原始状态寄存器。	0x0
[13]	RW1C	intr_soc13_trig_over_flag	SOC13 触发溢出中断原始状态寄存器。	0x0
[12]	RW1C	intr_soc12_trig_over_flag	SOC12 触发溢出中断原始状态寄存器。	0x0
[11]	RW1C	intr_soc11_trig_over_flag	SOC11 触发溢出中断原始状态寄存器。	0x0
[10]	RW1C	intr_soc10_trig_over_flag	SOC10 触发溢出中断原始状态寄存器。	0x0
[9]	RW1C	intr_soc9_trig_over_flag	SOC9 触发溢出中断原始状态寄存器。	0x0
[8]	RW1C	intr_soc8_trig_over_flag	SOC8 触发溢出中断原始状态寄存器。	0x0

Bits	Access	Name	Description	Reset
[7]	RW1C	intr_soc7_trig_over_flag	SOC7 触发溢出中断原始状态寄存器。	0x0
[6]	RW1C	intr_soc6_trig_over_flag	SOC6 触发溢出中断原始状态寄存器。	0x0
[5]	RW1C	intr_soc5_trig_over_flag	SOC5 触发溢出中断原始状态寄存器。	0x0
[4]	RW1C	intr_soc4_trig_over_flag	SOC4 触发溢出中断原始状态寄存器。	0x0
[3]	RW1C	intr_soc3_trig_over_flag	SOC3 触发溢出中断原始状态寄存器。	0x0
[2]	RW1C	intr_soc2_trig_over_flag	SOC2 触发溢出中断原始状态寄存器。	0x0
[1]	RW1C	intr_soc1_trig_over_flag	SOC1 触发溢出中断原始状态寄存器。	0x0
[0]	RW1C	intr_soc0_trig_over_flag	SOC0 触发溢出中断原始状态寄存器。	0x0

ADC_ERR_INT_MSK

ADC_ERR_INT_MSK 为 ADC 异常中断屏蔽后状态寄存器。

Offset Address: 0x02D4 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RO	intr_dma_req_over_flag_msk	DMA 请求溢出中断屏蔽后状态寄存器。	0x0
[15]	RO	intr_soc15_trig_over_flag_msk	SOC15 触发溢出中断屏蔽后状态寄存器。	0x0
[14]	RO	intr_soc14_trig_over_flag_msk	SOC14 触发溢出中断屏蔽后状态寄存器。	0x0
[13]	RO	intr_soc13_trig_over_flag_msk	SOC13 触发溢出中断屏蔽后状态寄存器。	0x0
[12]	RO	intr_soc12_trig_over_flag_msk	SOC12 触发溢出中断屏蔽后状态寄存器。	0x0
[11]	RO	intr_soc11_trig_over_flag_msk	SOC11 触发溢出中断屏蔽后状态寄存器。	0x0
[10]	RO	intr_soc10_trig_over_flag_msk	SOC10 触发溢出中断屏蔽后状态寄存器。	0x0

Bits	Access	Name	Description	Reset
[9]	RO	intr_soc9_trig_over_flag_msk	SOC9 触发溢出中断屏蔽后状态寄存器。	0x0
[8]	RO	intr_soc8_trig_over_flag_msk	SOC8 触发溢出中断屏蔽后状态寄存器。	0x0
[7]	RO	intr_soc7_trig_over_flag_msk	SOC7 触发溢出中断屏蔽后状态寄存器。	0x0
[6]	RO	intr_soc6_trig_over_flag_msk	SOC6 触发溢出中断屏蔽后状态寄存器。	0x0
[5]	RO	intr_soc5_trig_over_flag_msk	SOC5 触发溢出中断屏蔽后状态寄存器。	0x0
[4]	RO	intr_soc4_trig_over_flag_msk	SOC4 触发溢出中断屏蔽后状态寄存器。	0x0
[3]	RO	intr_soc3_trig_over_flag_msk	SOC3 触发溢出中断屏蔽后状态寄存器。	0x0
[2]	RO	intr_soc2_trig_over_flag_msk	SOC2 触发溢出中断屏蔽后状态寄存器。	0x0
[1]	RO	intr_soc1_trig_over_flag_msk	SOC1 触发溢出中断屏蔽后状态寄存器。	0x0
[0]	RO	intr_soc0_trig_over_flag_msk	SOC0 触发溢出中断屏蔽后状态寄存器。	0x0

ADC_ERR_INT_MASK

ADC_ERR_INT_MASK 为 ADC 异常中断屏蔽寄存器。

Offset Address: 0x02D8 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	intr_dma_req_over_flag_mask	DMA 请求溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[15]	RW	intr_soc15_trig_over_flag_mask	SOC15 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[14]	RW	intr_soc14_trig_ov	SOC14 触发溢出中断屏蔽寄存器。	0x0

Bits	Access	Name	Description	Reset
		er_flag_mask	0：屏蔽中断； 1：不屏蔽。	
[13]	RW	intr_soc13_trig_over_flag_mask	SOC13 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[12]	RW	intr_soc12_trig_over_flag_mask	SOC12 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[11]	RW	intr_soc11_trig_over_flag_mask	SOC11 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[10]	RW	intr_soc10_trig_over_flag_mask	SOC10 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[9]	RW	intr_soc9_trig_over_flag_mask	SOC9 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[8]	RW	intr_soc8_trig_over_flag_mask	SOC8 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[7]	RW	intr_soc7_trig_over_flag_mask	SOC7 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[6]	RW	intr_soc6_trig_over_flag_mask	SOC6 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[5]	RW	intr_soc5_trig_over_flag_mask	SOC5 触发溢出中断屏蔽寄存器。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽中断； 1：不屏蔽。	
[4]	RW	intr_soc4_trig_over_flag_mask	SOC4 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[3]	RW	intr_soc3_trig_over_flag_mask	SOC3 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[2]	RW	intr_soc2_trig_over_flag_mask	SOC2 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[1]	RW	intr_soc1_trig_over_flag_mask	SOC1 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0
[0]	RW	intr_soc0_trig_over_flag_mask	SOC0 触发溢出中断屏蔽寄存器。 0：屏蔽中断； 1：不屏蔽。	0x0

ADC_DMA

ADC_DMA 为 ADC DMA 配置寄存器。

Offset Address: 0x02F0 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5]	RW	cfg_dma_brst_req_sel	DMA BURST 请求使能。 0：不使能； 1：使能。	0x0
[4]	RW	cfg_dma_sing_req	DMA SINGLE 请求使能。	0x0

Bits	Access	Name	Description	Reset
		_sel	0：不使能； 1：使能。	
[3:0]	RW	cfg_dma_soc_sel	DMA SOC 选择。 0x0：SOC0； 0x1：SOC1； ... 0xF：SOC15。	0x0

ADC_EN

ADC_EN 为 ADC 使能寄存器。

Offset Address: 0x0300 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	cfg_adc_en	ADC 使能寄存器。 0：关闭； 1：使能。	0x0

ADC_EN_DLY

ADC_EN_DLY 为 ADC 使能延迟寄存器。

Offset Address: 0x0310 Total Reset Value: 0x0000_0064

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:0]	RW	cfg_en_dly	ADC 使能后延迟时间（us）。	0x64

ADC_MODE

ADC_MODE 为 ADC 工作模式配置寄存器。

Offset Address: 0x0400 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RO	reserved	保留。	0x00000000
[1:0]	RW	cfg_adc_mode	ADC 工作模式。 00：正常工作模式； 01：电容校准模式 1； 其他：保留。	0x0

ADC_OEGE_CH_SEL

ADC_OEGE_CH_SEL 为 ADC OE GE 通道配置寄存器。

Offset Address: 0x0420 Total Reset Value: 0x001F_1F1F

Bits	Access	Name	Description	Reset
[31:21]	RO	reserved	保留。	0x000
[20:16]	RW	cfg_oege_ch_sel2	连接通道选择（该配置选择的通道，由 cfg_gain2/cfg_ofst2 进行补偿）。	0x1F
[15:13]	RO	reserved	保留。	0x0
[12:8]	RW	cfg_oege_ch_sel1	连接通道选择（该配置选择的通道，由 cfg_gain1/cfg_ofst1 进行补偿）。	0x1F
[7:5]	RO	reserved	保留。	0x0
[4:0]	RW	cfg_oege_ch_sel0	连接通道选择（该配置选择的通道，由 cfg_gain0/cfg_ofst0 进行补偿）。	0x1F

ADC_OEGE_CTRL0

ADC_OEGE_CTRL0 为 ADC OE GE 配置寄存器 0。

Offset Address: 0x0440 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0

Bits	Access	Name	Description	Reset
[28:16]	RW	cfg_gain0	软件配置 GE 值（无符号小数，bit[11:0]是小数位）。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_ofst0	软件配置 OE 值（有符号小数，补码表示，bit[0]是小数位）。	0x000

ADC_OEGE_CTRL1

ADC_OEGE_CTRL1 为 ADC OE GE 配置寄存器 1。

Offset Address: 0x0444 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_gain1	软件配置 GE 值（有符号小数，补码表示，bit[0]是小数位）。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_ofst1	软件配置 OE 值（有符号小数，补码表示，bit[0]是小数位）。	0x000

ADC_OEGE_CTRL2

ADC_OEGE_CTRL2 为 ADC OE GE 配置寄存器 2。

Offset Address: 0x0448 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_gain2	软件配置 GE 值（无符号小数，bit[11:0]是小数位）。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_ofst2	软件配置 OE 值（有符号小数，补码表	0x000

Bits	Access	Name	Description	Reset
			示，bit[0]是小数位）。	

ADC_PROCESS0

ADC_PROCESS0 为 ADC 数据处理寄存器 0。

Offset Address: 0x0460 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_ofst_cali	第二级 OFFSET 补偿值（有符号小数，补码表示，bit[0]是小数位）。	0x000

ADC_PROCESS1

ADC_PROCESS1 为 ADC 数据处理寄存器 1。

Offset Address: 0x0464 Total Reset Value: 0x0000_001B

Bits	Access	Name	Description	Reset
[31:5]	RO	reserved	保留。	0x0000000
[4:2]	RW	cfg_noise_add_bits	数字加噪大小。	0x6
[1]	RW	cfg_w_norm_sel	权重归一化使能。 0：不使能； 1：使能。	0x1
[0]	RW	cfg_noise_add_en	数字加噪使能。 0：不使能； 1：使能。	0x1

ADC_STATUS

ADC_STATUS 为 ADC 状态寄存器。

Offset Address: 0x0480 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RO	adc_busy	ADC 工作状态。 0：空闲； 1：工作中。	0x0

ADC_WEIGHT_CFG15

ADC_WEIGHT_CFG15 为 ADC 电容权重配置寄存器。

Offset Address: 0x0500 Total Reset Value: 0x0003_B800

Bits	Access	Name	Description	Reset
[31:19]	RO	reserved	保留。	0x0000
[18:0]	RW	cfg_weight15	电容 15 权重配置值（12bit 整数位，7bit 小数位）。	0x3B800

ADC_WEIGHT_CFG14

ADC_WEIGHT_CFG14 为 ADC 电容权重配置寄存器。

Offset Address: 0x0504 Total Reset Value: 0x0002_0000

Bits	Access	Name	Description	Reset
[31:18]	RO	reserved	保留。	0x0000
[17:0]	RW	cfg_weight14	电容 14 权重配置值（11bit 整数位，7bit 小数位）。	0x20000

ADC_WEIGHT_CFG13

ADC_WEIGHT_CFG13 为 ADC 电容权重配置寄存器。

Offset Address: 0x0508 Total Reset Value: 0x0001_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000

Bits	Access	Name	Description	Reset
[16:0]	RW	cfg_weight13	电容 13 权重配置值（10bit 整数位，7bit 小数位）。	0x10000

ADC_WEIGHT_CFG12

ADC_WEIGHT_CFG12 为 ADC 电容权重配置寄存器。

Offset Address: 0x050C Total Reset Value: 0x0000_8000

Bits	Access	Name	Description	Reset
[31:16]	RO	reserved	保留。	0x0000
[15:0]	RW	cfg_weight12	电容 12 权重配置值（9bit 整数位，7bit 小数位）。	0x8000

ADC_WEIGHT_CFG11

ADC_WEIGHT_CFG11 为 ADC 电容权重配置寄存器。

Offset Address: 0x0510 Total Reset Value: 0x0000_4000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14:0]	RW	cfg_weight11	电容 11 权重配置值（8bit 整数位，7bit 小数位）。	0x4000

ADC_WEIGHT_CFG10

ADC_WEIGHT_CFG10 为 ADC 电容权重配置寄存器。

Offset Address: 0x0514 Total Reset Value: 0x0000_4000

Bits	Access	Name	Description	Reset
[31:15]	RO	reserved	保留。	0x00000
[14:0]	RW	cfg_weight10	电容 10 权重配置值（8bit 整数位，7bit 小数位）。	0x4000

ADC_WEIGHT_CFG9

ADC_WEIGHT_CFG9 为 ADC 电容权重配置寄存器。

Offset Address: 0x0518 Total Reset Value: 0x0000_2000

Bits	Access	Name	Description	Reset
[31:14]	RO	reserved	保留。	0x00000
[13:0]	RW	cfg_weight9	电容 9 权重配置值（7bit 整数位，7bit 小数位）。	0x2000

ADC_WEIGHT_CFG8

ADC_WEIGHT_CFG8 为 ADC 电容权重配置寄存器。

Offset Address: 0x051C Total Reset Value: 0x0000_1000

Bits	Access	Name	Description	Reset
[31:13]	RO	reserved	保留。	0x00000
[12:0]	RW	cfg_weight8	电容 8 权重配置值（6bit 整数位，7bit 小数位）。	0x1000

ADC_WEIGHT_CFG7

ADC_WEIGHT_CFG7 为 ADC 电容权重配置寄存器。

Offset Address: 0x0520 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_weight7	电容 7 权重配置值（5bit 整数位，7bit 小数位）。	0x800

ADC_WEIGHT_CFG6

ADC_WEIGHT_CFG6 为 ADC 电容权重配置寄存器。

Offset Address: 0x0524 Total Reset Value: 0x0000_0800

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11:0]	RW	cfg_weight6	电容 6 权重配置值（5bit 整数位，7bit 小数位）。	0x800

ADC_WEIGHT_CFG5

ADC_WEIGHT_CFG5 为 ADC 电容权重配置寄存器。

Offset Address: 0x0528 Total Reset Value: 0x0000_0400

Bits	Access	Name	Description	Reset
[31:11]	RO	reserved	保留。	0x000000
[10:0]	RW	cfg_weight5	电容 5 权重配置值（4bit 整数位，7bit 小数位）。	0x400

ADC_WEIGHT_CFG4

ADC_WEIGHT_CFG4 为 ADC 电容权重配置寄存器。

Offset Address: 0x052C Total Reset Value: 0x0000_0200

Bits	Access	Name	Description	Reset
[31:10]	RO	reserved	保留。	0x000000
[9:0]	RW	cfg_weight4	电容 4 权重配置值（3bit 整数位，7bit 小数位）。	0x200

ADC_WEIGHT_CFG3

ADC_WEIGHT_CFG3 为 ADC 电容权重配置寄存器。

Offset Address: 0x0530 Total Reset Value: 0x0000_0100

Bits	Access	Name	Description	Reset
[31:9]	RO	reserved	保留。	0x000000

Bits	Access	Name	Description	Reset
[8:0]	RW	cfg_weight3	电容 3 权重配置值（2bit 整数位，7bit 小数位）。	0x100

ADC_CAP_TRG

ADC_CAP_TRG 为 ADC 电容校准触发寄存器。

Offset Address: 0x0540 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RO	reserved	保留。	0x00000000
[1]	W1_PULSE	cfg_weight_ini	权重初始化，将软件配置权重值加载到 ADC 中。	0x0
[0]	W1_PULSE	cfg_mode1_start	模式 1 校准触发，触发逻辑自动校准权重值。	0x0

ADC_CAP_M1

ADC_CAP_M1 为 ADC 电容校准模式 1 寄存器。

Offset Address: 0x0550 Total Reset Value: 0x0000_0D3B

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000
[11]	RW	cfg_weight_limit_bypass	权重上下限 bypass 使能。 0：不使能； 1：使能。	0x1
[10]	RW	cfg_mode1_caplsb_sel	模式 1 低位电容使能。 0：不使能； 1：使能。	0x1
[9:7]	RW	cfg_weight_limit_sel	权重上下限档位选择。 000：±1%；	0x2

Bits	Access	Name	Description	Reset
			001：±2%； 010：±4%； 011：±8%； 其他：保留。	
[6:3]	RW	cfg_cap_index	模式 1 起始电容配置。	0x7
[2:0]	RW	cfg_mode1_acc_sel	模式 1 累积次数选择。 000：2； 001：256； 010：512； 011：1024； 100：2048； 其他：保留。	0x3

ADC_ANA_CTRL0

ADC_ANA_CTRL0 为 ADC 模拟寄存器 0。

Offset Address: 0x0654 Total Reset Value: 0x0000_0009

Bits	Access	Name	Description	Reset
[31:28]	RW	cfg_sar_samp_cap_sel	ADC 采样电容数量配置。	0x0
[27:24]	RW	cfg_sar_vref	ADC VREF 预留寄存器。	0x0
[23:20]	RW	cfg_sar_vcm	ADC VCM 预留寄存器。	0x0
[19:16]	RW	cfg_sar_comp	ADC COMP 预留寄存器。	0x0
[15:5]	RO	reserved	保留。	0x000
[4:3]	RW	cfg_muxtime_sel	通道提前切换周期选择。 00：5clk； 01：10clk； 10：15clk；	0x1

Bits	Access	Name	Description	Reset
			11：20clk。	
[2]	RW	cfg_latch_dly_sel	SAMP 下降沿是否延迟 1 个周期产生 latch 选择。 0：不延迟； 1：延迟。	0x0
[1]	RW	cfg_comp_chop_en	ADC 比较器的 Chopper 使能信号。 0：不使能； 1：使能。	0x0
[0]	RW	cfg_inmux_en	SARADC 通道选择控制使能信号。 0：不使能； 1：使能。	0x1

ADC_AVDD_EN

ADC_AVDD_EN 为 AVDD 通道使能寄存器。

Offset Address: 0x0658 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x0000000
[0]	RW	cfg_avdd_en	AVDD/3 通道使能： 0：不使能； 1：使能。	0x0

ADC_TSENSOR_TRIM

ADC_TSENSOR_TRIM 为 Tsensor trim 寄存器。

Offset Address: 0x0800 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:12]	RO	reserved	保留。	0x00000

Bits	Access	Name	Description	Reset
[11:0]	RW	cfg_tsensor_ofst_trim	Tsensor offset 补偿 trim 值。	0x000

ADC_OEGE_TRIM

ADC_OEGE_TRIM 为 ADC OE GE TRIM 寄存器。

Offset Address: 0x0804 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_gain_cali_trim	ADC 通用 GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_ofst_cali_trim	ADC 通用 OFFSET 校准 trim 值。	0x000

ADC_PGA0_OEGE_TRIM0

ADC_PGA0_OEGE_TRIM0 为 ADC PGA0 OE GE TRIM 寄存器 0。

Offset Address: 0x0810 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga0_gain_trim2	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga0_ofst_trim2	OFFSET 校准 trim 值。	0x000

ADC_PGA0_OEGE_TRIM1

ADC_PGA0_OEGE_TRIM1 为 ADC PGA0 OE GE TRIM 寄存器 1。

Offset Address: 0x0814 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga0_gain_trim4	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga0_ofst_trim4	OFFSET 校准 trim 值。	0x000

ADC_PGA0_OEGE_TRIM2

ADC_PGA0_OEGE_TRIM2 为 ADC PGA0 OE GE TRIM 寄存器 2。

Offset Address: 0x0818 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga0_gain_trim8	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga0_ofst_trim8	OFFSET 校准 trim 值。	0x000

ADC_PGA0_OEGE_TRIM3

ADC_PGA0_OEGE_TRIM3 为 ADC PGA0 OE GE TRIM 寄存器 3。

Offset Address: 0x081C Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga0_gain_trim16	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga0_ofst_trim16	OFFSET 校准 trim 值。	0x000

ADC_PGA1_OEGE_TRIM0

ADC_PGA1_OEGE_TRIM0 为 ADC PGA1 OE GE TRIM 寄存器 0。

Offset Address: 0x0820 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga1_gain_trim2	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga1_ofst_trim2	OFFSET 校准 trim 值。	0x000

ADC_PGA1_OEGE_TRIM1

ADC_PGA1_OEGE_TRIM1 为 ADC PGA1 OE GE TRIM 寄存器 1。

Offset Address: 0x0824 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga1_gain_trim4	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga1_ofst_trim4	OFFSET 校准 trim 值。	0x000

ADC_PGA1_OEGE_TRIM2

ADC_PGA1_OEGE_TRIM2 为 ADC PGA1 OE GE TRIM 寄存器 2。

Offset Address: 0x0828 Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga1_gain_trim8	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga1_ofst_trim8	OFFSET 校准 trim 值。	0x000

ADC_PGA1_OEGE_TRIM3

ADC_PGA1_OEGE_TRIM3 为 ADC PGA1 OE GE TRIM 寄存器 3。

Offset Address: 0x082C Total Reset Value: 0x1000_0000

Bits	Access	Name	Description	Reset
[31:29]	RO	reserved	保留。	0x0
[28:16]	RW	cfg_pga1_gain_trim16	GAIN 校准 trim 值。	0x1000
[15:12]	RO	reserved	保留。	0x0
[11:0]	RW	cfg_pga1_ofst_trim16	OFFSET 校准 trim 值。	0x000

ADC_ANA_TRIM

ADC_ANA_TRIM 为 ADC 模拟 TRIM 寄存器。

Offset Address: 0x0840 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:16]	-	reserved	保留。	0x0000
[15:8]	RW	cfg_sar_trim1	ADC 模拟 trim 寄存器 1。	0x00
[7:0]	RW	cfg_sar_trim0	ADC 模拟 trim 寄存器 0。	0x00

24.16 TSENSOR_REG 寄存器概览

TSENSOR_REG 寄存器概览如表 24-13 所示。

表24-13 TSENSOR_REG 寄存器概览（基址是 0x1850_0000）

偏移地址	名称	描述	页码
0x0000	TSENSOR_CTRL_REG	TSENSOR 控制寄存器	TSENSOR_CTRL_REG
0x0010	TSENSOR_TRIM_REG	TSENSOR TRIM 寄存器	TSENSOR_TR

偏移地址	名称	描述	页码
			IM_REG

24.17 TSENSOR_REG 寄存器描述

TSENSOR_CTRL_REG

TSENSOR_CTRL_REG 为 TSENSOR 控制寄存器。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	cfg_tsensor_pd	Tsensor 的关断信号： 0：使能 tsensor； 1：关断 tsensor。	0x0

TSENSOR_TRIM_REG

TSENSOR_TRIM_REG 为 TSENSOR TRIM 寄存器。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:0]	RW	cfg_tsensor_trim	Tsensor trim 信息。	0x00

25 可编程增益放大器(PGA)

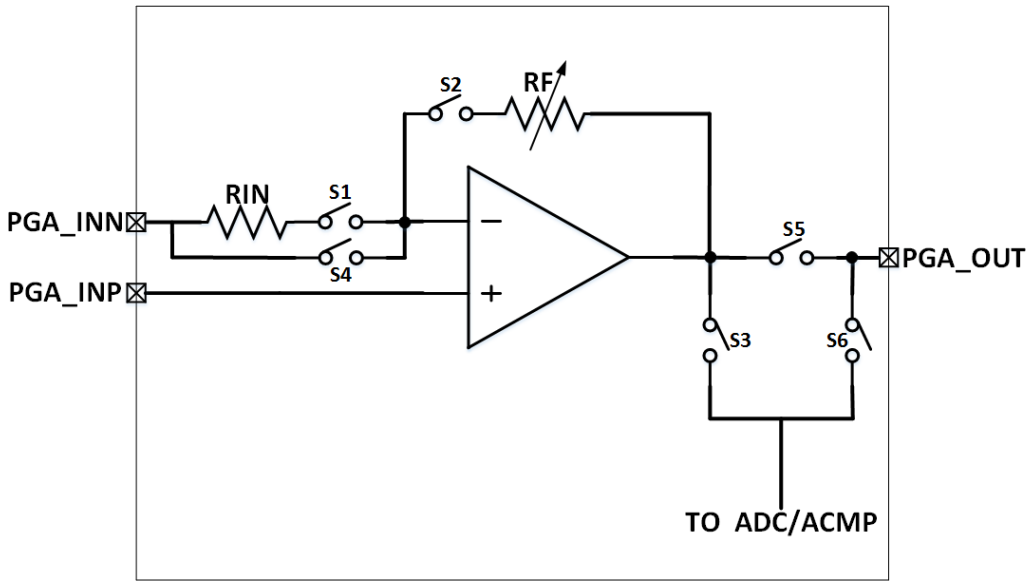
25.1 概述

PGA 为可编程增益运算放大器。PGA 同时是 A/D 转换器的高性能模拟前端。PGA 集成多个内部增益选项，用于调整宽范围的输入电压信号，使 A/D 转换器满量程信号达到均一化，提高测量精度。

25.2 特性

- 支持全差分输入。
- 内部电阻模式，增益可编程为 2、4、8 或 16。
- 外部电阻模式，灵活可调增益。

图25-1 PGA 结构框图



25.3 PGA 使能

PGA 使能前，打开时钟门控 PERI_CRG660[clk_ana_cken]，解除复位状态 PERI_CRG660[ana_srst_req]，解除 PGA0 复位状态 PERI_CRG676[pga0_srst_req] 或者 PGA1 复位状态 PERI_CRG677[pga1_srst_req]。

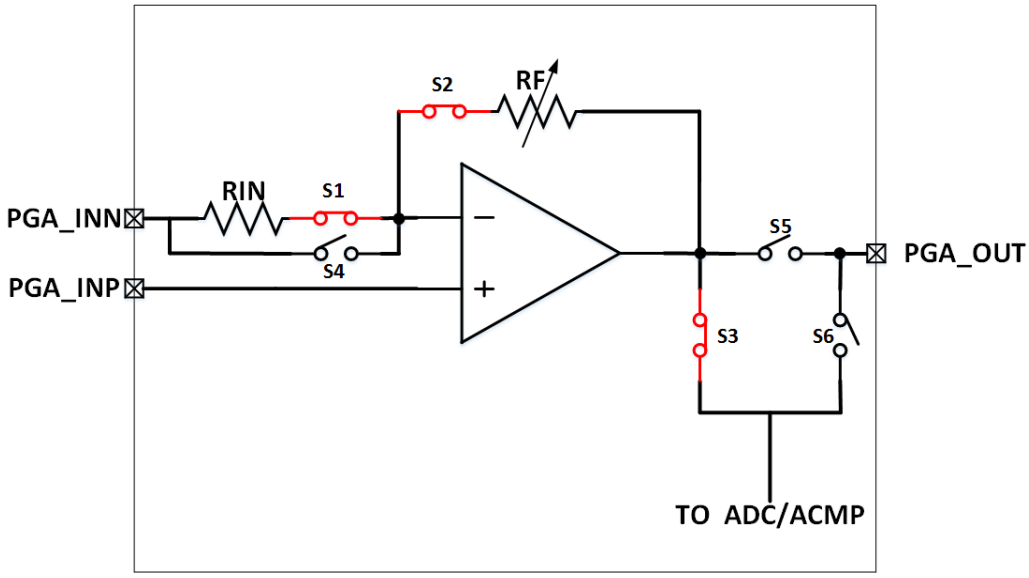
PGA 使能通过寄存器 PGA_CTRL_REG0[da_pga_enh]进行控制。

25.4 工作模式

PGA 可以工作在内置电阻模式和外置电阻模式，通过寄存器 PGA_CTRL_REG1[da_pga_mode_ctrl]进行选择。

25.4.1 内置电阻模式

图25-2 内置电阻模式结构图



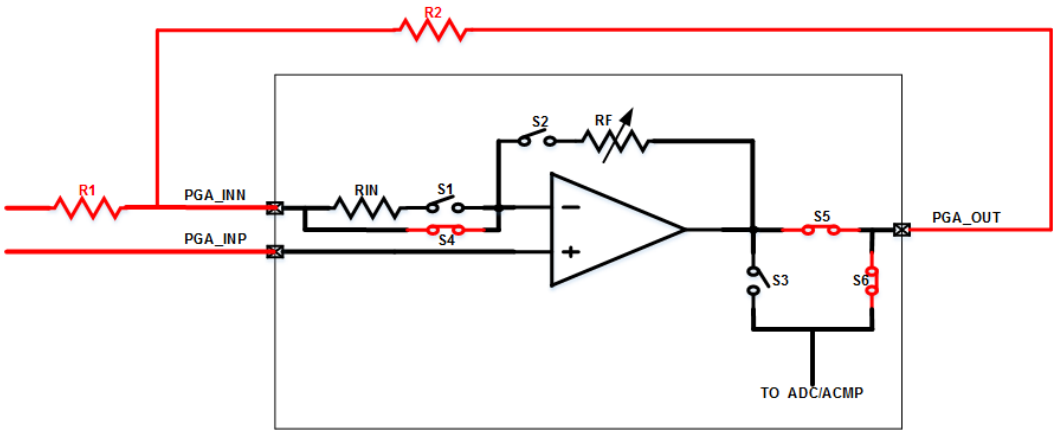
内置电阻模式下，PGA 的输出连接到 ADC/ACMP 输入，不会输出到 PGA_OUT 管脚。PGA 的增益档位通过寄存器 PGA_CTRL_REG1[da_pga_gain_ctrl]进行配置选择。

表25-1 PGA 增益选择

da_pga_gain_ctrl	增益
00	x2
01	x4
10	x8
11	x16

25.4.2 外置电阻模式

图25-3 外置电阻模式结构图



外置电阻模式下，PGA 的输出连接到 ADC/ACMP 输入和 PGA_OUT 管脚。PGA 的增益由电阻 R2 和 R1 确定，Gain=1+R2/R1。

用户根据设定增益（1+R2/R1）进行前馈电容补偿配置（PCB 设计时，PGA_INN 需约束寄生电容小于 5pF），前馈电容补偿可以改善 PGA 稳定性，提升 PGA 建立速度，因此必须进行配置；配置通过寄存器 PGA_CTRL_REG1[da_pga_cf_ctrl]和 PGA_RSV_REG[da_pga_cf_ctrl1]进行前馈电容补偿选择。

表25-2 外部电阻增益与前馈电容补偿配置关系表

增益（1+R2/R1）	PGA_CTRL_REG1[da_pga_cf_ctrl]	PGA_RSV_REG[da_pga_cf_ctrl1]
x2	000	00
x3	001	00
x4	010	00
x5	011	00
x6	100	00
x7	101	00
x8	110	00
x9	111	00
x10~x12	111	01
x13~x15	111	10

增益 (1+R2/R1)	PGA_CTRL_REG1[da_pga_cf_ctrl]	PGA_RSV_REG[da_pga_cf_ctrl1]
x16~x20	111	11

- 用户设定增益（1+R2/R1）对应的前馈电容补偿向下配置，例如设定增益为 2.4，建议配置为增益 X3 的前馈电容补偿；设定增益为 9.6，建议配置为增益 X10~X12 的前馈电容补偿；
- 用户设定增益（1+R2/R1）建议不超过 20 进行使用；

25.5 PGA_REG 寄存器概览

PGA_REG 寄存器概览如表 25-3 所示。

表25-3 PGA_REG 寄存器概览（基址是 PGA0：0x18200000；PGA1：0x18201000）

偏移地址	名称	描述	页码
0x0000	PGA_CTRL_REG0	PGA 控制寄存器 0	PGA_CTRL_REG0
0x0004	PGA_CTRL_REG1	PGA 控制寄存器 1	PGA_CTRL_REG1
0x0020	PGA_TRIM_REG	PGA TRIM 寄存器	PGA_TRIM_REG
0x0060	PGA_RSV_REG	PGA 预留寄存器	PGA_RSV_REG

25.6 PGA_REG 寄存器描述

PGA_CTRL_REG0

PGA_CTRL_REG0 为 PGA 控制寄存器 0。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	da_pga_enh	PGA 使能信号。 0：关闭； 1：使能。	0x0

PGA_CTRL_REG1

PGA_CTRL_REG1 为 PGA 控制寄存器 1。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16]	RW	da_pga_mode_ctrl	PGA 模式配置。 0：内部电阻增益模式； 1：外部电阻增益模式。	0x0
[15:10]	RO	reserved	保留。	0x00
[9:8]	RW	da_pga_gain_ctrl	PGA 内部电阻增益配置。 00：x2； 01：x4； 10：x8； 11：x16。	0x0
[7:3]	RO	reserved	保留。	0x00
[2:0]	RW	da_pga_cf_ctrl	PGA 外部增益模式下的前馈电容补偿配置。 000：x2； 001：x3； 010：x4； 011：x5； 100：x6；	0x0

Bits	Access	Name	Description	Reset
			101：x7； 110：x8； 111：x9。	

PGA_TRIM_REG

PGA_TRIM_REG 为 PGA TRIM 寄存器。

Offset Address: 0x0020 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:9]	RO	reserved	保留。	0x000000
[8:0]	RW	da_pga_vos_trim	PGA 的失调 trim 信息。	0x000

PGA_RSV_REG

PGA_RSV_REG 为 PGA 预留寄存器。

Offset Address: 0x0060 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RO	reserved	保留。	0x00000000
[1:0]	RW	da_pga_cf_ctrl1	外部增益模式下的前馈电容补偿配置。 00：保留； 01：x10~x12； 10：x13~x15； 11：x16~x20。 **注意：配置时需要将 PGA_CTRL_REG1[da_pga_cf_ctrl]配置为 111。**	0x0

26 数模转换器(DAC)

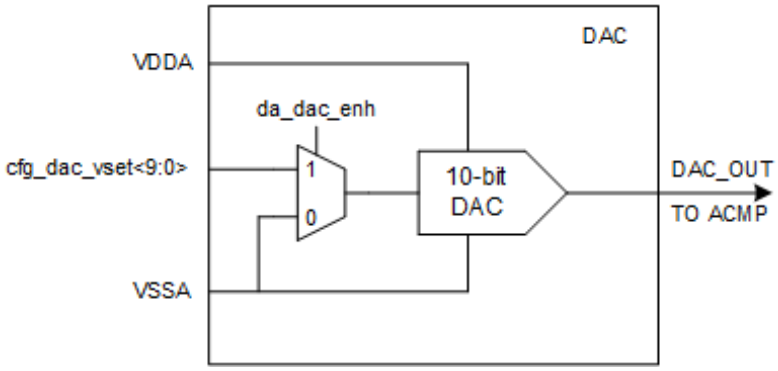
26.1 概述

DAC 将数字信号转换成对应的模拟电压，实现了数字到模拟的转换。

26.2 特性

- 10bit 分辨率
- 并行数据输入
- 300kHz 转换速率

图26-1 DAC 结构框图



26.3 DAC 使能

DAC 使能前，打开时钟门控 PERI_CRG660[clk_ana_cken]，解除复位状态 PERI_CRG660[ana_srst_req]，解除 DAC 复位状态 PERI_CRG672[dac_srst_req]。

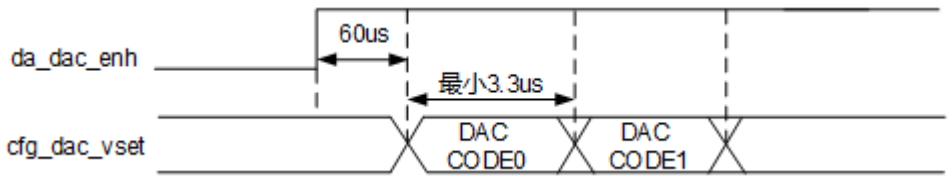
DAC 使能通过寄存器 DAC_CTRL_REG0[da_dac_enh]进行控制，配置 DAC 使能后，等待 60us 后 DAC 开始稳定工作，可以进行电压配置。

26.4 工作方式

DAC 工作为静态配置，最大码值切换速率为 300kHz，两次配置切换码值的时间间隔最小为 3.3us，配置寄存器 DAC_CTRL_REG1[cfg_dac_vset]设置 DAC 电压。

DAC 输出电压期望结果为 V = (cfg_dac_vset * (VDDA − VSSA)) / 1024。

图26-2 DAC 时序



26.5 DAC_REG 寄存器概览

DAC_REG 寄存器概览如表 26-1 所示。

表26-1 DAC_REG 寄存器概览（基址是 0x1840_0000）

偏移地址	名称	描述	页码
0x0000	DAC_CTRL_REG0	DAC 控制寄存器 0	DAC_CTRL_REG0
0x0010	DAC_CTRL_REG1	DAC 控制寄存器 1	DAC_CTRL_REG1

26.6 DAC_REG 寄存器描述

DAC_CTRL_REG0

DAC_CTRL_REG0 为 DAC 控制寄存器 0。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	da_dac_enh	DAC 使能信号。 0：关闭; 1：使能。	0x0

DAC_CTRL_REG1

DAC_CTRL_REG1 为 DAC 控制寄存器 1。

Offset Address: 0x0010 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:10]	RO	reserved	保留。	0x000000
[9:0]	RW	cfg_dac_vset	DAC 电压档位。	0x000

27 模拟比较器(ACMP)

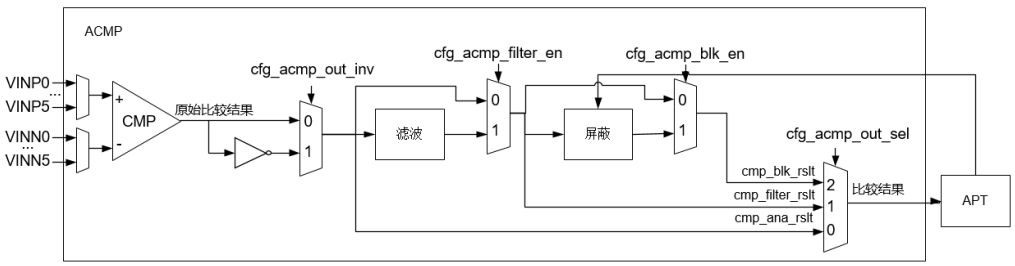
27.1 概述

ACMP 为模拟电压比较器，选择两个输入源进行电压比较，比较结果可以经过数字滤波和屏蔽处理。ACMP 的比较信号有三个来源，即管脚输入、DAC 输出或 PGA 输出。

27.2 特性

- 包括 1 个 2 输入比较器。
- 6 组比较源可配置选择。
- 最小有效差分输入电压为 20mV。
- 共模输入电压范围支持 0V ~ 输入电源大小。
- 最大 65536 个系统时钟周期的滤波档位。
- 可配置比较结果屏蔽。

图27-1 比较器结构框图



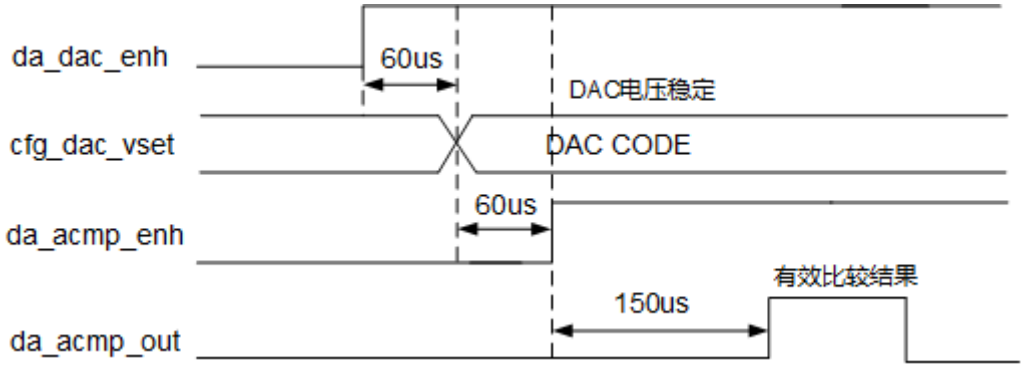
27.3 比较器使能

比较器使能前，打开时钟门控 PERI_CRG660[clk_ana_cken]，解除复位状态 PERI_CRG660[ana_srst_req]，解除 ACMP 复位状态 PERI_CRG668[acmp_srst_req]。

通过寄存器 ACMP_CTRL_REG0[da_acmp_enh]进行控制，配置比较器使能后，等待 150us，比较器输出结果生效。

当与 DAC 配合使用时，需要先使能 DAC，等待 60us 后配置 DAC 电压，再等待 60us 后使能 ACMP，确保在比较器使能时 DAC 输出电压稳定，比较器结果不会出现误翻转。

图27-2 ACMP 和 DAC 配合时序



27.4 迟滞电压

比较器的迟滞电压通过寄存器 ACMP_RSV_REG[da_acmp_hyvol_sel]进行选择。

表27-1 ACMP 迟滞电压选择

da_acmp_hyvol_sel	迟滞电压（mV）
00	0
01	10
10	20
11	30

27.5 端口选择

通过寄存器 ACMP_CTRL_REG1[da_acmp_input_psel]]和[da_acmp_input_nsel]可配置分别选择比较器的 P 端和 N 端的输入端口。

表27-2 ACMP P 端输入选择

da_acmp_input_psel	选择端口	信号源
000	VINP0	PGA0_OUT
001	VINP1	PGA1_OUT
010	VINP2	来自管脚（PA5）
011	VINP3	来自管脚（PC5）
100	VINP4	来自管脚（PD5）
101	VINP5	DAC_OUT
其他	保留	保留

表27-3 ACMP N 端输入选择

da_acmp_input_nsel	选择端口	信号源
000	VINN0	DAC_OUT
001	VINN1	无
010	VINN2	来自管脚（PA6）
011	VINN3	来自管脚（PC6）
100	VINN4	来自管脚（PD6）
101	VINN5	DAC_OUT
其他	保留	保留

27.6 比较极性选择

比较器可通过寄存器 ACMP_CTRL_REG2[cfg_acmp_out_inv]选择电压比较的输出极性。

表27-4 输出极性选择

cfg_acmp_out_inv	输出极性
0	当 VINP > VINN 时，比较器结果为 1； 当 VINP < VINN 时，比较器结果为 0。
1	当 VINP > VINN 时，比较器结果为 0； 当 VINP < VINN 时，比较器结果为 1。

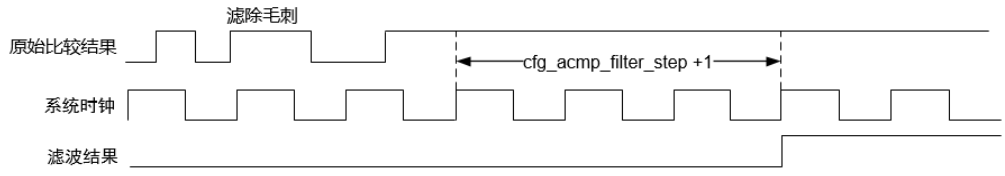
27.7 滤波功能

对于模拟比较结果，ACMP 的数字部分可以对比较结果进行滤波处理，提供最大 65536 个系统时钟周期的去抖范围可以配置选择。

数字滤波功能由寄存器 ACMP_CTRL_REG3[cfg_acmp_filter_en]使能。

滤波长度由寄存器 ACMP_CTRL_REG3[cfg_acmp_filter_step]进行配置，单位长度为系统主频时钟周期，即脉冲宽度小于(cfg_acmp_filter_step+1)*T_{pclk} 的比较结果会被滤除。滤波长度最大 16bit 可配，配置为 0 则将比较结果在系统时钟下延迟一拍输出。

图27-3 比较器滤波功能



27.8 屏蔽功能

ACMP 可以配置 ACMP_CTRL_REG4[cfg_acmp_blk_en]选择是否对比较结果进行屏蔽处理。

屏蔽窗口的来源可以为 APT 模块输出的屏蔽窗口，也可以为软件配置屏蔽窗口。寄存器 ACMP_CTRL_REG4[cfg_acmp_blk_sel]选择屏蔽信号来自 APT 屏蔽窗口或软件配置。ACMP_CTRL_REG4[cfg_acmp_blk_win]寄存器为软件配置屏蔽窗口。

寄存器 ACMP_CTRL_REG4[cfg_acmp_blk_pol_sel]可选择屏蔽窗口的有效电平，若配置为 0，则屏蔽窗口为高电平进行屏蔽，输出屏蔽结果，为低电平时输出比较结果。若配置为 1，则屏蔽窗口为低电平进行屏蔽，输出屏蔽结果，为高电平时输出比较结果。

寄存器 ACMP_CTRL_REG4[cfg_acmp_blk_rslt_pol]可选择屏蔽结果的极性。

须知

与 APT 输出的屏蔽窗口配合工作时，需要将屏蔽窗口的有效电平 ACMP_CTRL_REG4[cfg_acmp_blk_pol_sel]配置为 1，屏蔽结果的极性 ACMP_CTRL_REG4[cfg_acmp_blk_rslt_pol]配置为 0，保证极性与 APT 匹配。

图27-4 比较器屏蔽功能

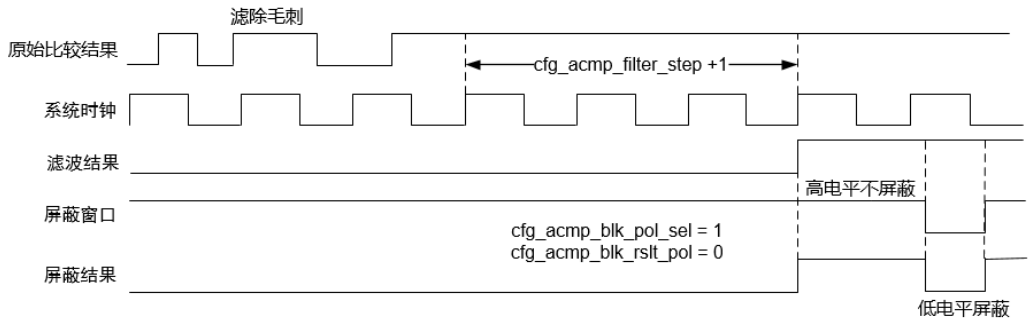
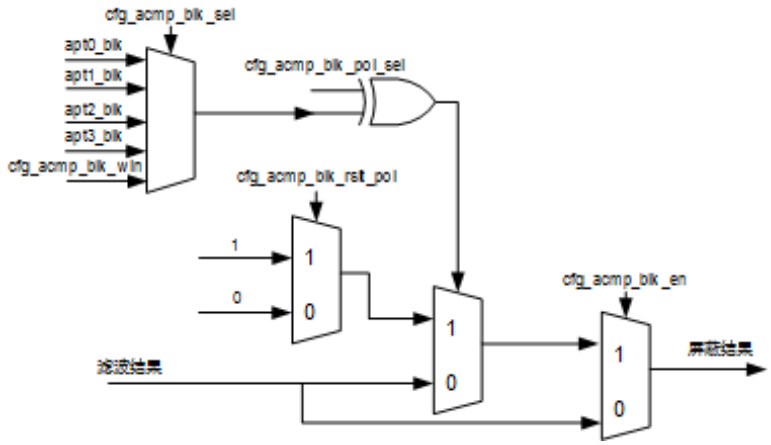


图27-5 比较器屏蔽逻辑



27.9 结果输出

比较器的输出结果可以由寄存器 ACMP_CTRL_REG2[cfg_acmp_out_sel]进行配置选择。

表27-5 比较器结果选择

cfg_acmp_out_sel	比较结果
0	模拟原始比较结果
1	数字滤波比较结果
2	数字滤波和屏蔽后比较结果
3	0

此外，比较结果可以通过寄存器 ACMP_RSLT_REG 读取，进行观测。

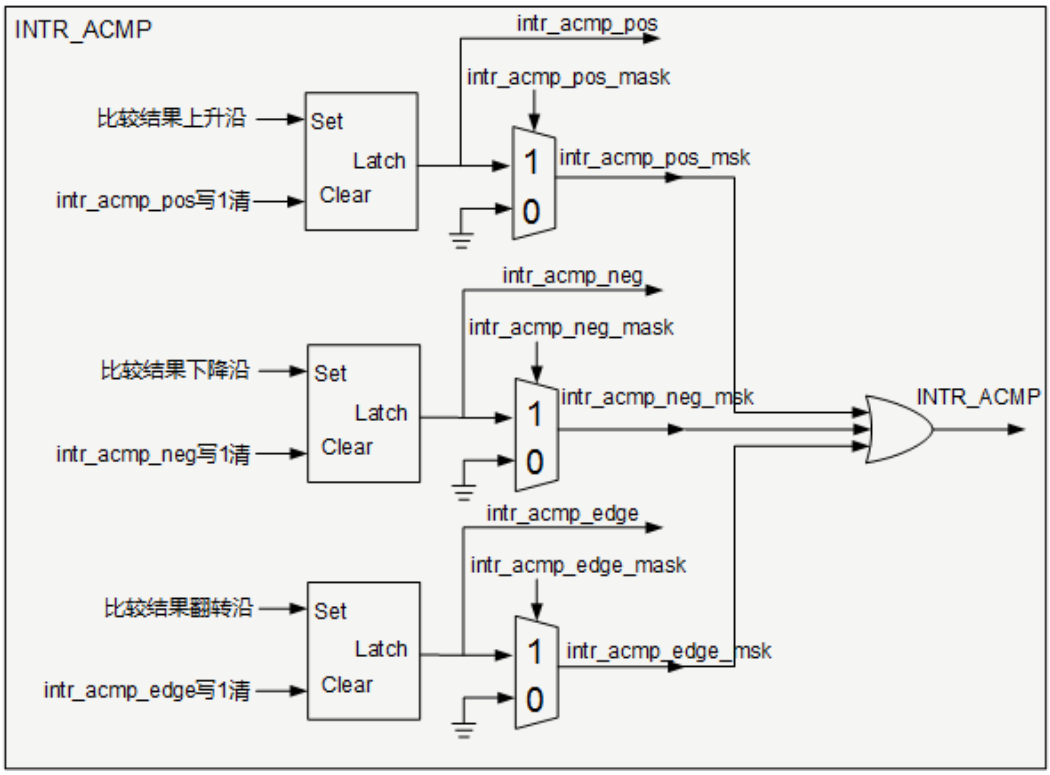
27.10 中断

比较器对比较结果进行边沿检测，并上报检测中断。

比较器中断在中断状态寄存器 ACMP_INTR_MSK 进行查询。

比较器中断的触发源包括：比较结果的上升沿、下降沿和翻转沿。触发源可通过中断屏蔽寄存器 ACMP_INTR_MASK 进行单独屏蔽。

图27-6 比较器中断



27.11 ACMP_REG 寄存器概览

ACMP_REG 寄存器概览如表 27-6 所示。

表27-6 ACMP_REG 寄存器概览（基址是 0x1830_0000）

偏移地址	名称	描述	页码
0x0000	ACMP_CTRL_REG0	ACMP 控制寄存器 0	ACMP_CTRL_REG0
0x0004	ACMP_CTRL_REG1	ACMP 控制寄存器 1	ACMP_CTRL_REG1
0x0008	ACMP_CTRL_REG2	ACMP 控制寄存器 2	ACMP_CTRL_

偏移地址	名称	描述	页码
			REG2
0x0040	ACMP_CTRL_REG3	ACMP 滤波控制寄存器	ACMP_CTRL_REG3
0x0044	ACMP_CTRL_REG4	ACMP 屏蔽控制寄存器	ACMP_CTRL_REG4
0x0050	ACMP_INTR	ACMP 中断原始状态寄存器	ACMP_INTR
0x0054	ACMP_INTR_MSK	ACMP 中断屏蔽后状态寄存器	ACMP_INTR_MSK
0x0058	ACMP_INTR_MASK	ACMP 中断屏蔽寄存器	ACMP_INTR_MASK
0x0070	ACMP_RSLT_REG	ACMP 结果寄存器	ACMP_RSLT_REG
0x0088	ACMP_TRIM_REG	ACMP TRIM 寄存器	ACMP_TRIM_REG
0x008C	ACMP_RSV_REG	ACMP 预留寄存器	ACMP_RSV_REG

27.12 ACMP_REG 寄存器描述

ACMP_CTRL_REG0

ACMP_CTRL_REG0 为 ACMP 控制寄存器 0。

Offset Address: 0x0000 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:1]	RO	reserved	保留。	0x00000000
[0]	RW	da_acmp_enh	比较器使能信号。 0：关闭；	0x0

Bits	Access	Name	Description	Reset
			1：使能。	

ACMP_CTRL_REG1

ACMP_CTRL_REG1 为 ACMP 控制寄存器 1。

Offset Address: 0x0004 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:6]	RO	reserved	保留。	0x0000000
[5:3]	RW	da_acmp_input_nsel	比较器输入通道 N 端选通。 000：VINN0； 001：VINN1； 010：VINN2； 011：VINN3； 100：VINN4； 101：VINN5； 其他：未使用。	0x0
[2:0]	RW	da_acmp_input_psel	比较器输入通道 P 端选通。 000：VINP0； 001：VINP1； 010：VINP2； 011：VINP3； 100：VINP4； 101：VINP5； 其他：未使用。	0x0

ACMP_CTRL_REG2

ACMP_CTRL_REG2 为 ACMP 控制寄存器 2。

Offset Address: 0x0008 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RW	cfg_acmp_out_inv	比较器结果极性选择。 0：结果不取反； 1：结果取反。	0x0
[1:0]	RW	cfg_acmp_out_sel	比较器输出结果选择。 00：原始比较结果； 01：滤波后结果； 10：屏蔽后结果； 11：0。	0x0

ACMP_CTRL_REG3

ACMP_CTRL_REG3 为 ACMP 滤波控制寄存器。

Offset Address: 0x0040 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:17]	RO	reserved	保留。	0x0000
[16:1]	RW	cfg_acmp_filter_step	比较器滤波步长。	0x0000
[0]	RW	cfg_acmp_filter_en	比较器滤波使能。 0：关闭； 1：使能。	0x0

ACMP_CTRL_REG4

ACMP_CTRL_REG4 为 ACMP 屏蔽控制寄存器。

Offset Address: 0x0044 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:26]	RO	reserved	保留。	0x00
[25]	RW	cfg_acmp_blk_rslt_pol	屏蔽结果极性选择。	0x0

Bits	Access	Name	Description	Reset
			0：屏蔽结果为低电平； 1：屏蔽结果为高电平。	
[24]	RW	cfg_acmp_blk_pol_sel	屏蔽窗口极性选择。 0：高电平屏蔽窗有效； 1：低电平屏蔽窗有效。	0x0
[23:17]	RO	reserved	保留。	0x00
[16]	RW	cfg_acmp_blk_win	软件配置屏蔽窗口。	0x0
[15:11]	RO	reserved	保留。	0x00
[10:8]	RW	cfg_acmp_blk_sel	比较器屏蔽窗口选择。 000：软件配置屏蔽窗口； 001：APT0 输出屏蔽窗口； 010：APT1 输出屏蔽窗口； 011：APT2 输出屏蔽窗口； 100：APT3 输出屏蔽窗口； 其他：0。	0x0
[7:1]	RO	reserved	保留。	0x00
[0]	RW	cfg_acmp_blk_en	比较器屏蔽使能。 0：关闭； 1：使能。	0x0

ACMP_INTR

ACMP_INTR 为 ACMP 中断原始状态寄存器。

Offset Address: 0x0050 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RW1C	intr_acmp_pos	比较结果上升沿中断状态。	0x0

Bits	Access	Name	Description	Reset
[1]	RW1C	intr_acmp_neg	比较结果下降沿中断状态。	0x0
[0]	RW1C	intr_acmp_edge	比较结果翻转沿中断状态。	0x0

ACMP_INTR_MSK

ACMP_INTR_MSK 为 ACMP 中断屏蔽后状态寄存器。

Offset Address: 0x0054 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RO	intr_acmp_pos_msk	比较结果上升沿屏蔽后中断状态。	0x0
[1]	RO	intr_acmp_neg_msk	比较结果下降沿屏蔽后中断状态。	0x0
[0]	RO	intr_acmp_edge_msk	比较结果翻转沿屏蔽后中断状态。	0x0

ACMP_INTR_MASK

ACMP_INTR_MASK 为 ACMP 中断屏蔽寄存器。

Offset Address: 0x0058 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RW	intr_acmp_pos_mask	比较结果上升沿中断屏蔽寄存器。 0：屏蔽中断; 1：不屏蔽。	0x0
[1]	RW	intr_acmp_neg_mask	比较结果下降沿中断屏蔽寄存器。 0：屏蔽中断; 1：不屏蔽。	0x0
[0]	RW	intr_acmp_edge_mask	比较结果翻转沿中断屏蔽寄存器。 0：屏蔽中断;	0x0

Bits	Access	Name	Description	Reset
			1：不屏蔽。	

ACMP_RSLT_REG

ACMP_RSLT_REG 为 ACMP 结果寄存器。

Offset Address: 0x0070 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:3]	RO	reserved	保留。	0x00000000
[2]	RO	cmp_blk_rslt	比较器屏蔽后结果。	0x0
[1]	RO	cmp_filter_rslt	比较器滤波后结果。	0x0
[0]	RO	cmp_ana_rslt	原始比较结果。	0x0

ACMP_TRIM_REG

ACMP_TRIM_REG 为 ACMP TRIM 寄存器。

Offset Address: 0x0088 Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:8]	RO	reserved	保留。	0x000000
[7:0]	RW	da_acmp_trim	ACMP TIRM 寄存器。	0x00

ACMP_RSV_REG

ACMP_RSV_REG 为 ACMP 预留寄存器。

Offset Address: 0x008C Total Reset Value: 0x0000_0000

Bits	Access	Name	Description	Reset
[31:2]	RO	reserved	保留。	0x00000000
[1:0]	RW	da_acmp_hyvol_sel	迟滞电压选择。 00：0mV；	0x0

Bits	Access	Name	Description	Reset
			01：10mV； 10：20mV； 11：30mV。	

28 缩略语

缩略语	英文全称	中文解释
ACK	Acknowledge	应答
ACMP	Analog Comparator	模拟比较器
ADC	Analog Digital Converter	模数转换器
AHB	Advanced High-performance Bus	先进高性能总线
AP	Access Port	访问端口
APB	Advanced Peripheral Bus	高级外设总线
APT	Advanced PWM Timer	高级 PWM 定时器
BREQ	Burst Request	突发请求
BRP	Baud Rate Prescaler	波特率系数
CAN	Controller Area Network	控制器域网
CAPM	Capture Module	捕获模块
CFD	Clock Failure Detection	时钟失效检测
CMM	Clock Monitor Module	时钟频率监控
CRG	clock & reset generator	时钟和复位生成器
CSR	Control and Status Register	控制与状态寄存器
CTS	Clear To Send	允许发送

缩略语	英文全称	中文解释
DAC	Digital to Analog Conversion	数模转换器
DG	DeadBand Generation	死区生成
DMA	Direct Memory Access	直接内存访问
DMAC	Direct Memory Access Controller	直接存储器访问控制器
DPD	Deep Power Down	深度功耗下降
DTCM	Data Tightly-Coupled Memory	数据紧耦合内存
ECC	Error Correcting Code	内存纠错码
ECR	Event Capture Register	捕获事件
ECSEQR	Event Capture Sequence Register	事件捕获序列
EM	Event Management	事件管理
EOC	End-of-Conversion	结束转换标志
EVT	Extreme Value Theory	极值理论
FED	Falling Edge Delay	下降沿延时
FIFO	First In First Out	先进先出
Flash CMD	Flash Command	Flash 命令，即擦除或编程操作
FTCR	Filter Control Register	滤波控制
GE	Gain Error	增益误差
GPIO	General-Purpose Input/Output	通用输入输出
GPT	General PWM Timer	通用 PWM 定时器
I2C	Inter-Integrated Circuit	集成电路接口
IG	Interrupt Generation	中断生成
ISP	In-System Programming	在系统编程
ITCM	Instruction Tightly Coupled Memory	指令紧耦合内存
IWDG	Independent Watch Dog	独立看门狗

缩略语	英文全称	中文解释
JTAG	Joint Test Action Group	联合测试行动小组调试接口
LEC	Last Error Code	最近一次的错误代码
LLI	Linked List Item	链表项
LSB	Least Significant Bit	最低有效位
MCU	Microcontroller Unit	微控制器单元
MSB	Most Significant Bit	最高有效位
NACK	Not Acknowledge	非应答
NRZ	None-Return-to-Zero	不归零编码
OE	Offset Error	偏移误差
OP	Output Protect	输出保护
PDR	Power Down Reset	掉电复位
PE	Program and Erase	编程和擦除
PEC	Packet Error Code	数据包错误码
PG	PWM Generation	波形生成
PGA	Programmable Gain Amplifier	可编程增益放大器
PLL	Phase Locked Loop	锁相环
PLRU	Pseudo Least Recently Used	伪最近最少被使用替换
PMC	Power Manage Controller	电源控制
PMP	Physical Memory Protection	物理内存保护
POR	Power On Reset	上电复位
PPB	Post-Processing Block	后处理模块
PPU	Position Process Unit	位置处理单元
PTU	Period Trigger Unit	周期触发单元
PVD	Programmable Voltage Detector	可编程电压检测器

缩略语	英文全称	中文解释
PWM	Pulse Width Modulation	脉冲宽度调制
QDM	Quadrature Decoder Module	正交解码模块
QDU	Quadrature Decoder Unit	正交解码模单元
RD	Read	读
RED	Rising Edge Delay	上升沿延时
RR	Round Robin	轮询优先级
RTS	Request To Send	请求发送
RXD	Receive Data	接收数据
SCL	Serial Clock Line	串行时钟线
SDA	Serial Data Line	串行数据线
SOC	Start Of Conversion	启动转换
SPI	Shared Peripheral Interrupt	共享外设中断
SRAM	Static Random Accecss Memory	静态随机访问存储器
SREQ	Single Request	单个请求
SWD	Serial Wire Debug	串行线调试接口
TC	Timer Control	计数控制
TOR	Top Of Range	区间顶端地址模式
TSU	Time Stamp Unit	时间戳单元
TXD	Transmit Data	发送数据
UART	Universal Asynchronous Receiver Transmitter	通用异步收发传输器
WFI	Wait For Interrupt	等待中断指令
WWDG	Windowed Watch Dog	窗口看门狗
XIP	eXecute In Place	就地执行