

ESP32-C3

技术参考手册

PRELIMINARY



预发布 v0.4
乐鑫信息科技
版权 © 2021

目录

1	ESP-RISC-V CPU	18
1.1	概述	18
1.2	特性	18
1.3	地址分布	19
1.4	配置与状态寄存器 (CSR)	19
1.4.1	寄存器列表	19
1.4.2	寄存器	20
1.5	中断控制器	27
1.5.1	特性	27
1.5.2	功能描述	27
1.5.3	建议操作	29
1.5.3.1	延迟	29
1.5.3.2	配置流程	29
1.5.4	寄存器列表	30
1.5.5	寄存器	31
1.6	调试	34
1.6.1	概述	34
1.6.2	特性	34
1.6.3	功能描述	35
1.6.4	寄存器列表	35
1.6.5	寄存器	35
1.7	硬件触发器	38
1.7.1	特性	38
1.7.2	功能描述	38
1.7.3	触发执行流程	39
1.7.4	寄存器列表	39
1.7.5	寄存器	39
1.8	存储器保护	42
1.8.1	概述	42
1.8.2	特性	42
1.8.3	功能描述	42
1.8.4	寄存器列表	42
1.8.5	寄存器	43
2	通用 DMA 控制器 (GDMA)	44
2.1	概述	44
2.2	特性	44
2.3	架构	44
2.4	功能描述	45
2.4.1	链表	45
2.4.2	外设到存储及存储到外设的数据传输	46
2.4.3	存储到存储数据传输	47

2.4.4	启动 DMA	47
2.4.5	读链表	48
2.4.6	数据传输结束标志	48
2.4.7	访问片内 RAM	48
2.4.8	仲裁	49
2.4.9	带宽	49
2.5	GDMA 中断	49
2.6	编程流程	50
2.6.1	GDMA TX 通道配置流程	50
2.6.2	GDMA RX 通道配置流程	50
2.6.3	GDMA 存储器到存储器配置流程	51
2.7	寄存器列表	52
2.8	寄存器	55
3	系统和存储器	72
3.1	概述	72
3.2	主要特性	72
3.3	功能描述	73
3.3.1	地址映射	73
3.3.2	内部存储器	74
3.3.3	外部存储器	76
3.3.3.1	外部存储器地址映射	76
3.3.3.2	高速缓存	76
3.3.3.3	Cache 操作	77
3.3.4	GDMA 地址空间	77
3.3.5	模块/外设	78
3.3.5.1	模块/外设地址空间映射	78
4	eFuse 控制器 (EFUSE)	81
4.1	概述	81
4.2	主要特性	81
4.3	功能描述	81
4.3.1	结构	81
4.3.1.1	EFUSE_WR_DIS	85
4.3.1.2	EFUSE_RD_DIS	85
4.3.1.3	数据存储方式	85
4.3.2	软件烧写参数	86
4.3.3	软件读取参数	88
4.3.4	eFuse VDDQ 时序	89
4.3.5	硬件模块使用参数	89
4.3.6	中断	89
4.4	寄存器列表	90
4.5	寄存器	94
5	IO MUX 和 GPIO 交换矩阵 (GPIO, IO MUX)	134
5.1	概述	134

5.2	主要特性	134
5.3	结构概览	134
5.4	通过 GPIO 交换矩阵的外设输入	136
5.4.1	概述	136
5.4.2	信号同步	136
5.4.3	功能描述	137
5.4.4	简单 GPIO 输入	138
5.5	通过 GPIO 交换矩阵的外设输出	138
5.5.1	概述	138
5.5.2	功能描述	138
5.5.3	简单 GPIO 输出	139
5.5.4	Sigma Delta 调制输出 (SDM)	140
5.5.4.1	功能描述	140
5.5.4.2	配置方法	140
5.6	IO MUX 的直接输入输出功能	140
5.6.1	概述	140
5.6.2	功能描述	141
5.7	GPIO 管脚的模拟功能	141
5.8	管脚 Hold 特性	141
5.9	GPIO 管脚供电和电源管理	141
5.9.1	GPIO 管脚供电	142
5.9.2	电源管理	142
5.10	外设信号列表	142
5.11	IO MUX 管脚功能列表	148
5.12	IO MUX 管脚模拟功能列表	149
5.13	寄存器列表	149
5.13.1	GPIO 交换矩阵寄存器列表	149
5.13.2	IO MUX 寄存器列表	151
5.13.3	SDM 寄存器列表	152
5.14	寄存器	152
5.14.1	GPIO 交换矩阵寄存器	152
5.14.2	IO MUX 寄存器	160
5.14.3	SDM 寄存器	162
6	复位和时钟	164
6.1	复位	164
6.1.1	概述	164
6.1.2	结构图	164
6.1.3	特性	164
6.1.4	功能描述	165
6.2	时钟	165
6.2.1	概述	166
6.2.2	结构图	166
6.2.3	特性	166
6.2.4	功能描述	167
6.2.4.1	CPU 时钟	167

6.2.4.2	外设时钟	167
6.2.4.3	Wi-Fi 和 Bluetooth® LE 时钟	169
6.2.4.4	RTC 时钟	169
7	芯片 Boot 控制	170
7.1	概述	170
7.2	Boot 模式控制	170
7.3	ROM 代码日志打印控制	171
8	中断矩阵 (INTMATRIX)	173
8.1	概述	173
8.2	特性	173
8.3	功能描述	173
8.3.1	外部中断源	173
8.3.2	CPU 中断	177
8.3.3	分配外部中断源至 CPU 外部中断	177
8.3.3.1	分配一个外部中断源 Source_X 至 CPU 外部中断	177
8.3.3.2	分配多个外部中断源 Source_X _n 至 CPU 外部中断	177
8.3.3.3	关闭 CPU 外部中断源 Source_X	177
8.3.4	查询外部中断源当前的中断状态	177
8.4	寄存器列表	178
8.5	寄存器	182
9	系统定时器 (SYSTIMER)	188
9.1	概述	188
9.2	主要特性	188
9.3	时钟源选择	189
9.4	功能描述	189
9.4.1	计数器	189
9.4.2	比较器和报警	190
9.4.3	同步操作	191
9.4.4	中断	191
9.5	编程示例	191
9.5.1	读取当前计数器的值	191
9.5.2	在单次报警模式下配置一次性报警	192
9.5.3	在周期报警模式下配置周期性报警	192
9.5.4	唤醒后时间补偿	192
9.6	寄存器列表	193
9.7	寄存器	194
10	定时器组 (TIMG)	205
10.1	概述	205
10.2	功能描述	206
10.2.1	16 位预分频器与时钟选择器	206
10.2.2	54 位时基计数器	206
10.2.3	报警产生	206

10.2.4	定时器重新加载	207
10.2.5	低功耗时钟 (SLOW_CLK) 频率计算	207
10.2.6	中断	208
10.3	配置与使用	208
10.3.1	定时器用作简单时钟	208
10.3.2	定时器用于单次报警	209
10.3.3	定时器用于周期性报警	209
10.3.4	SLOW_CLK 频率计算	209
10.4	寄存器列表	211
10.5	寄存器	212
11	看门狗定时器 (WDT)	222
11.1	概述	222
11.2	数字看门狗定时器	222
11.2.1	主要特性	222
11.2.2	功能描述	223
11.2.2.1	时钟源与 32 位计数器	223
11.2.2.2	阶段与超时动作	224
11.2.2.3	写保护	224
11.2.2.4	Flash 引导保护	225
11.3	模拟看门狗定时器	225
11.3.1	主要特性	225
11.3.2	SWD 控制器	225
11.3.2.1	结构	226
11.3.2.2	工作流程	226
11.4	中断	226
11.5	寄存器	226
12	XTAL32K 看门狗定时器 (XTWDT)	228
12.1	概述	228
12.2	主要特性	228
12.2.1	XTAL32K 看门狗定时器的中断及唤醒	228
12.2.2	BACKUP32K_CLK	228
12.3	功能描述	228
12.3.1	工作流程	228
12.3.2	BACKUP32K_CLK 实现原理	229
12.3.3	BACKUP32K_CLK 分频因子配置方法	229
13	系统寄存器 (SYSREG)	230
13.1	概述	230
13.2	主要特性	230
13.3	功能描述	230
13.3.1	系统和存储器寄存器	230
13.3.1.1	内部存储器	230
13.3.1.2	片外存储器	231
13.3.1.3	RSA 存储器	231

13.3.2	时钟配置寄存器	231
13.3.3	中断信号寄存器	231
13.3.4	低功耗管理寄存器	232
13.3.5	外设时钟门控和复位寄存器	232
13.4	寄存器列表	234
13.5	寄存器	235
14	辅助调试 (Debug Assist)	247
14.1	概述	247
14.2	主要特性	247
14.3	功能描述	247
14.3.1	区域读写监测	247
14.3.2	栈指针监测	247
14.3.3	PC 记录	247
14.3.4	CPU/DMA 总线访问记录	247
14.4	工作流程	247
14.4.1	区域监测和栈监测配置	247
14.4.2	PC 记录配置	249
14.4.3	CPU/DMA 总线访问记录配置	249
14.5	寄存器列表	252
14.6	寄存器	254
15	SHA 加速器 (SHA)	270
15.1	概述	270
15.2	主要特性	270
15.3	工作模式简介	270
15.4	功能描述	271
15.4.1	信息预处理	271
15.4.1.1	附加填充比特	271
15.4.1.2	信息解析	271
15.4.1.3	哈希初始值 (Initial Hash Value)	272
15.4.2	哈希运算流程	272
15.4.2.1	Typical SHA 模式下的运算流程	272
15.4.2.2	DMA-SHA 模式下的运算流程	273
15.4.3	信息摘要存储	274
15.4.4	中断	274
15.5	寄存器列表	275
15.6	寄存器	276
16	AES 加速器 (AES)	279
16.1	概述	279
16.2	主要特性	279
16.3	工作模式简介	279
16.4	Typical AES 工作模式	280
16.4.1	密钥、明文、密文	280
16.4.2	字节序	281

16.4.3	Typical AES 工作模式的流程	283
16.5	DMA-AES 工作模式	284
16.5.1	密钥、明文、密文	284
16.5.2	字节序	285
16.5.3	标准增量函数	285
16.5.4	块个数	285
16.5.5	初始向量	286
16.5.6	DMA-AES 工作模式的流程	286
16.6	存储器列表	287
16.7	寄存器列表	287
16.8	寄存器	288
17	RSA 加速器 (RSA)	292
17.1	概述	292
17.2	主要特性	292
17.3	功能描述	292
17.3.1	大数模幂运算	292
17.3.2	大数模乘运算	294
17.3.3	大数乘法运算	294
17.3.4	控制加速	295
17.4	存储器列表	296
17.5	寄存器列表	297
17.6	寄存器	298
18	HMAC 加速器 (HMAC)	302
18.1	主要特性	302
18.2	功能描述	302
18.2.1	上行模式	302
18.2.2	下行 JTAG 启动模式	302
18.2.3	下行数字签名模式	303
18.2.4	HMAC eFuse 配置	303
18.2.5	调用 HMAC 流程 (详细说明)	304
18.3	HMAC 算法细节	306
18.3.1	附加填充比特	306
18.3.2	HMAC 算法结构	306
18.4	寄存器列表	308
18.5	寄存器	310
19	数字签名 (DS)	316
19.1	概述	316
19.2	主要特性	316
19.3	功能描述	316
19.3.1	概述	316
19.3.2	私钥运算子	316
19.3.3	软件需要做的准备工作	317
19.3.4	硬件工作流程	318

19.3.5 软件工作流程	318
19.4 存储器列表	320
19.5 寄存器列表	321
19.6 寄存器	322
20 时钟毛刺检测	324
20.1 概述	324
20.2 功能描述	324
20.2.1 时钟毛刺检测	324
20.2.2 复位	324
21 随机数发生器 (RNG)	325
21.1 概述	325
21.2 主要特性	325
21.3 功能描述	325
21.4 编程指南	325
21.5 寄存器列表	326
21.6 寄存器	326
22 UART 控制器 (UART)	327
22.1 概述	327
22.2 主要特性	327
22.3 UART 架构	328
22.4 功能描述	329
22.4.1 时钟与复位	329
22.4.2 UART RAM	329
22.4.3 波特率产生与检测	330
22.4.3.1 波特率产生	330
22.4.3.2 波特率检测	331
22.4.4 UART 数据帧	332
22.4.5 RS485	333
22.4.5.1 驱动控制	333
22.4.5.2 转换延时	333
22.4.5.3 总线侦听	333
22.4.6 IrDA	334
22.4.7 唤醒	334
22.4.8 流控	335
22.4.8.1 硬件流控	335
22.4.8.2 软件流控	336
22.4.9 GDMA 模式	337
22.4.10 UART 中断	337
22.4.11 UCHI 中断	338
22.5 编程流程	338
22.5.1 寄存器类型	338
22.5.1.1 同步寄存器	338
22.5.1.2 静态寄存器	339

22.5.1.3	立即寄存器	340
22.5.2	具体步骤	340
22.5.2.1	URAT n 模块初始化	341
22.5.2.2	URAT n 通信配置	342
22.5.2.3	启动 URAT n	342
22.6	寄存器列表	343
22.7	寄存器	345
23	I2C 控制器 (I2C)	379
23.1	概述	379
23.2	主要特性	379
23.3	I2C 架构	380
23.4	功能描述	382
23.4.1	时钟配置	382
23.4.2	滤除 SCL 和 SDA 噪声	382
23.4.3	SCL 时钟拉伸	382
23.4.4	SCL 空闲时产生 SCL 脉冲	383
23.4.5	同步	383
23.4.6	漏级开路输出	384
23.4.7	时序参数配置	384
23.4.8	超时控制	385
23.4.9	指令配置	386
23.4.10	TX/RX RAM 数据存储	387
23.4.11	数据转换	388
23.4.12	寻址模式	388
23.4.13	10 位寻址的读写标志位检查	388
23.4.14	启动控制器	388
23.5	编程示例	388
23.5.1	I2C 主机写入从机, 7 位寻址, 单次命令序列	389
23.5.1.1	场景介绍	389
23.5.1.2	配置示例	389
23.5.2	I2C 主机写入从机, 10 位寻址, 单次命令序列	391
23.5.2.1	场景介绍	391
23.5.2.2	配置示例	391
23.5.3	I2C 主机写入从机, 7 位双地址寻址, 单次命令序列	392
23.5.3.1	场景介绍	392
23.5.3.2	配置示例	393
23.5.4	I2C 主机写入从机, 7 位寻址, 多次命令序列	394
23.5.4.1	场景介绍	394
23.5.4.2	配置示例	395
23.5.5	I2C 主机读取从机, 7 位寻址, 单次命令序列	396
23.5.5.1	场景介绍	396
23.5.5.2	配置示例	396
23.5.6	I2C 主机读取从机, 10 位寻址, 单次命令序列	398
23.5.6.1	场景介绍	398
23.5.6.2	配置示例	398

23.5.7 I2C 主机读取从机，7 位双寻址，单次命令序列	400
23.5.7.1 场景介绍	400
23.5.7.2 配置示例	400
23.5.8 I2C 主机读取从机，7 位寻址，多次命令序列	402
23.5.8.1 场景介绍	402
23.5.8.2 配置示例	403
23.6 中断	404
23.7 寄存器列表	406
23.8 寄存器	408
24 USB 串口/JTAG 控制器 (USB_SERIAL_JTAG)	426
24.1 概述	426
24.2 特性	426
24.3 功能描述	427
24.3.1 CDC-ACM USB 接口描述	427
24.3.2 CDC-ACM 固件接口描述	428
24.3.3 USB-JTAG 接口：JTAG 命令处理器	429
24.3.4 USB-JTAG 接口：CMD_REP 使用示例	430
24.3.5 USB-JTAG 接口：响应捕捉单元	430
24.3.6 USB-JTAG 接口：控制传输请求	430
24.4 操作建议	431
24.5 寄存器列表	433
24.6 寄存器	434
25 双线汽车接口 (TWAI)	448
25.1 主要特性	448
25.2 功能性协议	448
25.2.1 TWAI 性能	448
25.2.2 TWAI 报文	449
25.2.2.1 数据帧和远程帧	449
25.2.2.2 错误帧和过载帧	451
25.2.2.3 帧间距	453
25.2.3 TWAI 错误	453
25.2.3.1 错误类型	453
25.2.3.2 错误状态	454
25.2.3.3 错误计数	454
25.2.4 TWAI 位时序	455
25.2.4.1 标称位	455
25.2.4.2 硬同步与再同步	455
25.3 结构概述	456
25.3.1 寄存器模块	457
25.3.2 位流处理器	457
25.3.3 错误管理逻辑	457
25.3.4 位时序逻辑	457
25.3.5 接收滤波器	458
25.3.6 接收 FIFO	458

25.4	功能描述	458
25.4.1	模式	458
25.4.1.1	复位模式	458
25.4.1.2	操作模式	458
25.4.2	位时序	458
25.4.3	中断管理	459
25.4.3.1	接收中断 (RXI)	460
25.4.3.2	发送中断 (TXI)	460
25.4.3.3	错误报警中断 (EWI)	460
25.4.3.4	数据溢出中断 (DOI)	460
25.4.3.5	被动错误中断 (TXI)	460
25.4.3.6	仲裁丢失中断 (ALI)	461
25.4.3.7	总线错误中断 (BEI)	461
25.4.3.8	总线状态中断 (BSI)	461
25.4.4	发送缓冲器与接收缓冲器	461
25.4.4.1	缓冲器概述	461
25.4.4.2	帧信息	462
25.4.4.3	帧标识符	462
25.4.4.4	帧数据	463
25.4.5	接收 FIFO 和数据溢出	463
25.4.6	接收滤波器	464
25.4.6.1	单滤波模式	464
25.4.6.2	双滤波模式	465
25.4.7	错误管理	466
25.4.7.1	错误报警限制	466
25.4.7.2	被动错误	467
25.4.7.3	离线状态与离线恢复	467
25.4.8	错误捕捉	467
25.4.9	仲裁丢失捕捉	469
25.5	寄存器列表	470
25.6	寄存器	471
26	LED PWM 控制器 (LEDC)	483
26.1	概述	483
26.2	特性	483
26.3	功能描述	483
26.3.1	架构	483
26.3.2	定时器	484
26.3.2.1	时钟源	484
26.3.2.2	时钟分频器配置	484
26.3.2.3	14 位计数器	485
26.3.3	PWM 生成器	486
26.3.4	占空比渐变	487
26.3.5	中断	487
26.4	寄存器列表	488
26.5	寄存器	490

27 红外遥控 (RMT)	497
27.1 概述	497
27.2 主要特性	497
27.3 功能描述	497
27.3.1 RMT 架构	498
27.3.2 RMT RAM	498
27.3.3 时钟	499
27.3.4 发射器	500
27.3.4.1 普通发送模式	500
27.3.4.2 乒乓发送模式	500
27.3.4.3 发送加载波	500
27.3.4.4 持续发送模式	500
27.3.4.5 多通道同时发送	501
27.3.5 接收器	501
27.3.5.1 普通接收模式	501
27.3.5.2 乒乓接收模式	501
27.3.5.3 接收滤波	501
27.3.5.4 接收去载波	502
27.3.6 配置参数更新	502
27.3.7 中断	503
27.4 寄存器列表	504
27.5 寄存器	505
28 片上传感器与模拟信号处理	517
28.1 概述	517
28.2 SAR ADC	517
28.2.1 概述	517
28.2.2 特性	517
28.2.3 功能描述	517
28.2.3.1 输入信号	519
28.2.3.2 ADC 转换和衰减	519
28.2.3.3 DIG ADC 控制器	519
28.2.3.4 DIG ADC 时钟	520
28.2.3.5 DMA 支持	520
28.2.3.6 DIG ADC FSM	520
28.2.3.7 ADC 滤波器	523
28.2.3.8 阈值监控	524
28.2.3.9 SAR ADC2 仲裁器	524
28.3 温度传感器	525
28.3.1 概述	525
28.3.2 特性	525
28.3.3 功能描述	525
28.4 中断	525
28.5 寄存器列表	526
28.6 寄存器	526

29 相关文档和资源	539
词汇列表	540
外设相关词汇	540
寄存器相关词汇	540
修订历史	541

PRELIMINARY

表格

1-1	CPU 地址分布	19
1-3	中断 ID 与异常向量地址	28
1-6	NAPOT 编码的 maddress	38
2-1	配置寄存器与外设选择关系表	46
2-2	链表描述符参数对齐要求	48
2-3	GDMA 支持的总带宽	49
3-1	地址映射	74
3-2	内部存储器地址映射	75
3-3	外部存储器地址映射	76
3-4	模块/外设地址空间映射表	78
4-1	BLOCK0 参数	82
4-2	密钥用途数值对应的含义	84
4-3	BLOCK1-10 参数	84
4-4	软件读取寄存器信息	88
4-5	VDDQ 默认时序参数配置	89
5-1	GPIO 交换矩阵外设信号	143
5-2	IO MUX 管脚功能	148
5-3	芯片上电过程中的管脚毛刺	149
5-4	IO MUX 管脚的模拟功能	149
6-1	复位源	165
6-2	CPU_CLK 时钟源选择	167
6-3	CPU_CLK 时钟频率	167
6-4	外设时钟	168
6-5	APB_CLK 时钟	169
6-6	CRYPTO_CLK 时钟	169
7-1	管脚默认上拉/下拉	170
7-2	系统启动模式	170
7-3	ROM 代码日志打印控制	171
8-1	CPU 外部中断配置寄存器、外部中断状态寄存器、外部中断源	175
9-1	UNIT _n 配置控制位	190
9-2	报警触发条件	191
9-3	同步操作	191
10-1	可逆计数器向上计数时的报警触发场景	207
10-2	可逆计数器向下计数时的报警触发场景	207
13-1	内存功耗控制位	231
13-2	外设时钟门控与复位控制位	232
14-1	CPU 包格式	250
14-2	DMA 包格式	250
14-3	DMA 访问来源	250
14-4	写内存数据格式	251
15-1	工作模式选择	270
15-2	运算标准选择	271
15-3	不同运算标准信息摘要的寄存器占用情况	274

16-1	工作模式	280
16-2	密钥长度和加解密方向	280
16-3	状态返回值	280
16-4	Typical AES 文本字节序	281
16-5	AES-128 密钥字节序	281
16-6	AES-256 密钥字节序	282
16-7	块模式选择	284
16-8	状态返回值	284
16-9	TEXT-PADDING	285
16-10	DMA AES 存储字节序	285
17-1	加速效果	296
18-1	HMAC 功能及配置数值	304
22-1	UART _n 同步寄存器	339
22-2	UART _n 静态寄存器	340
23-1	I2C 同步寄存器	383
24-1	标准 CDC-ACM 控制请求	428
24-2	CDC-ACM 中 RTS 和 DTR 的设置	428
24-3	半字节中的命令	429
24-4	USB-JTAG 控制请求	431
24-5	JTAG 功能描述符	431
24-6	复位芯片进入下载模式	432
24-7	复位 SoC 进行启动	432
25-1	不同帧类型、帧格式下的域及子域信息	451
25-2	错误帧中的位域信息	452
25-3	过载帧中的位域信息	452
25-4	帧间距中的域信息	453
25-5	名义位时序中包含的段	455
25-6	TWAI_BUS_TIMING_0_REG 的 bit 信息 (0x18)	459
25-7	TWAI_BUS_TIMING_1_REG 的 bit (0x1c)	459
25-8	SFF 与 EFF 的缓冲器布局	461
25-9	TX/RX 帧信息 (SFF/EFF); TWAI 地址 0x40	462
25-10	TX/RX 标识符 1 (SFF); TWAI 地址 0x44	462
25-11	TX/RX 标识符 2 (SFF); TWAI 地址 0x48	462
25-12	TX/RX 标识符 1 (EFF); TWAI 地址 0x44	463
25-13	TX/RX 标识符 2 (EFF); TWAI 地址 0x48	463
25-14	TX/RX 标识符 3 (EFF); TWAI 地址 0x4c	463
25-15	TX/RX 标识符 4 (EFF); TWAI 地址 0x50	463
25-16	TWAI_ERR_CODE_CAP_REG 中的位信息 (0x30)	468
25-17	SEG.4 - SEG.0 的位信息	468
25-18	TWAI_ARB_LOST_CAP_REG 中的位信息 (0x2c)	469
27-1	更新配置参数	502
28-1	SAR ADC 的信号输入	519
28-2	温度传感器的温度偏移	525

插图

1-1	CPU 框图	18
1-2	调试系统架构	34
2-1	具有 GDMA 功能的模块和 GDMA 通道	44
2-2	GDMA 引擎的架构	45
2-3	链表结构图	45
2-4	链表关系图	47
3-1	系统结构与地址映射结构	73
3-2	Cache 系统结构	77
3-3	具有 GDMA 功能的外设/模块	78
4-1	移位寄存器电路图（前 32 字节）	86
4-2	移位寄存器电路图（后 12 字节）	86
5-1	IO MUX 和 GPIO 交换矩阵框图（简图）	135
5-2	IO MUX 和 GPIO 交换矩阵框图（详图）	135
5-3	焊盘内部结构	136
5-4	GPIO 输入经 APB 时钟上升沿或下降沿同步	137
5-5	GPIO 输入信号滤波时序图	137
6-1	四种复位等级	164
6-2	系统时钟	166
8-1	中断矩阵结构图	173
9-1	系统定时器结构图	188
9-2	系统定时器生成报警	189
10-1	定时器组	205
10-2	定时器组架构	206
11-1	看门狗定时器概览	222
11-2	ESP32-C3 的看门狗定时器	223
11-3	SWD 控制器结构	226
12-1	XTAL32K 看门狗定时器	228
18-1	HMAC 附加填充比特示意图	306
18-2	HMAC 结构示意图	307
19-1	软件准备工作与硬件工作流程	317
20-1	XTAL_CLK 脉宽	324
21-1	噪声源	325
22-1	UART 基本架构图	328
22-2	UART 共享 RAM 图	329
22-3	UART 控制器分频	331
22-4	UART 信号下降沿较差时序图	331
22-5	UART 数据帧结构	332
22-6	AT_CMD 字符格式	332
22-7	RS485 模式驱动控制结构图	333
22-8	SIR 模式编解码时序图	334
22-9	IrDA 编解码结构图	334
22-10	硬件流控图	335
22-11	硬件流控信号连接图	336

22-12 GDMA 模式数据传输	337
22-13 UART 编程流程	341
23-1 I2C 主机基本架构	380
23-2 I2C 从机基本架构	380
23-3 I2C 协议时序 (引自 The I2C-bus specification Version 2.1 Fig.31)	381
23-4 I2C 时序参数 (引自 The I2C-bus specification Version 2.1 Table5)	381
23-5 I2C 时序图	384
23-6 I2C 命令寄存器结构	386
23-7 I2C 主机写 7 位寻址的从机	389
23-8 I2C 主机写 10 位寻址的从机	391
23-9 I2C 主机写 7 位双地址寻址从机	392
23-10 I2C 主机分段写 7 位寻址的从机	394
23-11 I2C 主机读 7 位寻址的从机	396
23-12 I2C 主机读 10 位寻址的从机	398
23-13 I2C 主机从 7 位寻址从机的 M 地址读取 N 个数据	400
23-14 I2C 主机分段读 7 位寻址的从机	402
24-1 USB Serial/JTAG 高层框图	426
24-2 USB Serial/JTAG 框图	427
25-1 数据帧和远程帧中的位域	450
25-2 错误帧中的位域	452
25-3 过载帧中的位域	452
25-4 帧间距中的域	453
25-5 标称位构成	455
25-6 TWAI 概略图	456
25-7 接收滤波器	464
25-8 单滤波模式	465
25-9 双滤波模式	466
25-10 错误状态变化	467
25-11 丢失仲裁的 bit 位置	469
26-1 LED PWM 控制器架构	483
26-2 定时器和 PWM 生成器功能块	484
26-3 LEDC_CLK_DIV_TIMER ^x 非整数时的分频	485
26-4 LED PWM 输出信号图	486
26-5 输出信号占空比渐变图	487
27-1 RMT 结构框图	498
27-2 RAM 中脉冲编码结构	498
28-1 SAR ADC 的功能概况	518
28-2 DIG ADC FSM 概况	521
28-3 APB_SARADC_SAR_PATT_TAB1_REG 与样式 0 - 3	522
28-4 APB_SARADC_SAR_PATT_TAB2_REG 与样式 4 - 7	522
28-5 样式表中的样式结构	522
28-6 cmd0 配置示例	522
28-7 cmd1 配置示例	523
28-8 DMA 数据格式	523

1 ESP-RISC-V CPU

1.1 概述

ESP-RISC-V CPU 是基于 RISC-V ISA 的 32 位内核，包括基本整数 (I)，乘法/除法 (M) 和压缩 (C) 标准扩展。ESP-RISC-V CPU 内核具有 4 级有序标量流水线，针对面积、功耗、性能等进行了优化。CPU 内核架构包含中断控制器 (INTC)、调试模块 (DM) 和用于访问存储器和外设的系统总线 (SYS BUS) 接口。

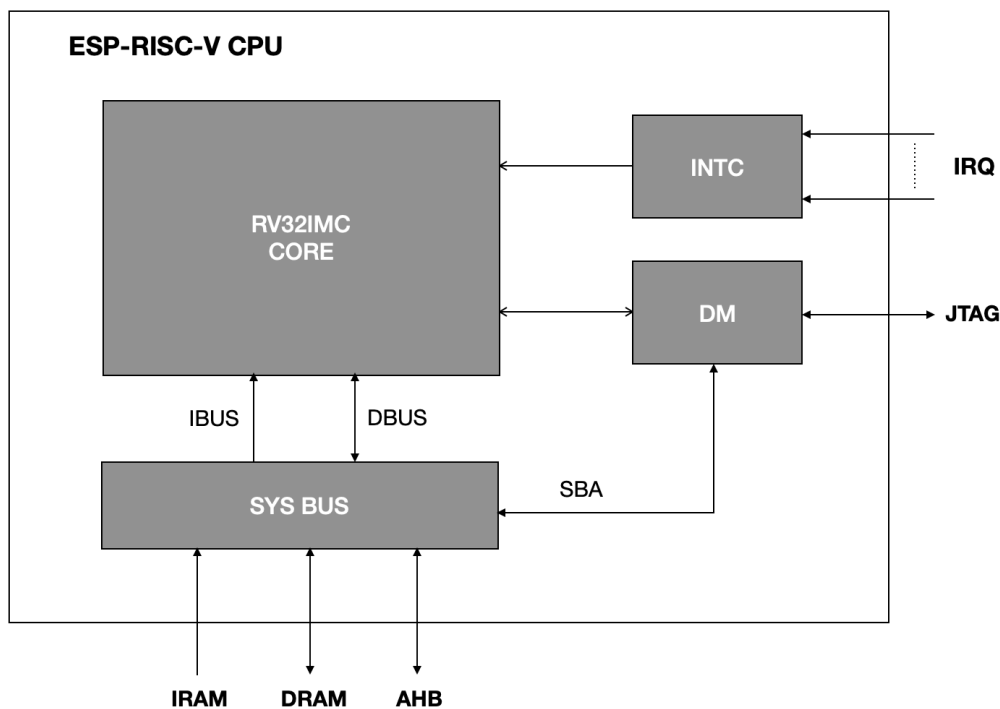


图 1-1. CPU 框图

1.2 特性

- 时钟工作频率高达 160 MHz
- 通过 IRAM/DRAM 接口零等待周期访问片上 SRAM 和缓存中的程序和数据
- 中断控制器 (INTC) 具有多达 31 个向量中断，可配置优先级和阈值级别
- 调试模块 (DM) 符合 RISC-V 调试规范 v0.13，支持通过行业标准的 JTAG/USB 端口连接外部调试器
- 调试器通过系统总线 (SBA) 直接访问存储器和外设
- 硬件触发器符合 RISC-V 调试规范 v0.13，具有多达 8 个断点/观察点
- 物理存储器保护 (PMP)，最多可配置 16 个区域
- 32 位 AHB 系统总线，用于访问外设
- 可配置的核心性能指标事件

1.3 地址分布

下表列出了 CPU 可访问的指令地址空间，数据地址空间，调试地址空间和通过系统总线访问的外设地址空间。

表 1-1. CPU 地址分布

名称	描述	起始地址	结束地址	访问
IRAM	指令地址空间	0x4000_0000	0x47FF_FFFF	读/写
DRAM	数据地址空间	0x3800_0000	0x3FFF_FFFF	读/写
DM	调试地址空间	0x2000_0000	0x27FF_FFFF	读/写
AHB	AHB 地址空间	* 默认	* 默认	读/写

* 默认：IRAM、DRAM、DM 地址范围以外的地址空间通过 AHB 总线访问。

1.4 配置与状态寄存器 (CSR)

1.4.1 寄存器列表

下表为 CPU 可访问的 CSR 列表。除了自定义的性能计数器 CSR 外，所有已实现的 CSR 都遵循 RISC-V 指令集手册 V1.10 第二卷“特权架构” (RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10) 中所述的位域标准映射。必须注意的是，受 CPU 中实现的功能子集的限制，即使在标准 CSR 中也并非实现了所有位域。有关详细的 CSR 寄存器描述，请参阅下一小节。

名称	描述	地址	访问
机器模式信息 CSR			
<code>mvendorid</code>	机器模式供应商编号寄存器	0xF11	只读
<code>marchid</code>	机器模式架构编号寄存器	0xF12	只读
<code>mimpid</code>	机器模式硬件实现编号寄存器	0xF13	只读
<code>mhartid</code>	机器模式硬件线程编号寄存器	0xF14	只读
机器模式异常设置 CSR			
<code>mstatus</code>	机器模式状态寄存器	0x300	读/写
<code>misa</code> ¹	机器模式 ISA 寄存器	0x301	读/写
<code>mtvec</code> ²	机器模式异常向量寄存器	0x305	读/写
机器模式异常处理 CSR			
<code>mscratch</code>	机器模式暂存寄存器	0x340	读/写
<code>mepc</code>	机器模式异常程序计数器	0x341	读/写
<code>mcause</code> ³	机器模式异常原因寄存器	0x342	读/写
<code>mtval</code>	机器模式异常值寄存器	0x343	读/写
物理存储器保护 (PMP) CSR			
<code>pmpcfg0</code>	物理存储器保护配置寄存器	0x3A0	读/写
<code>pmpcfg1</code>	物理存储器保护配置寄存器	0x3A1	读/写
<code>pmpcfg2</code>	物理存储器保护配置寄存器	0x3A2	读/写
<code>pmpcfg3</code>	物理存储器保护地址寄存器	0x3A3	读/写

¹尽管 `misa` 具有读/写属性，但由于它的域是硬连线的，所以写操作无效。在 RISC-V 术语中称为 WARL（写入任意数值读取合法数值）。

²`mtvec` 仅支持在向量模式下对异常处理进行配置，基地址为 256 字节对齐。

³`mcause` 中反映的外部中断 ID 也包括 RISC-V 标准为核心内部中断源预留的 ID。

名称	描述	地址	访问
pmpaddr0	物理存储器保护地址寄存器	0x3B0	读/写
pmpaddr1	物理存储器保护地址寄存器	0x3B1	读/写
....			
pmpaddr15	物理存储器保护地址寄存器	0x3BF	读/写
触发器模块 CSR（与调试模块共用）			
tselect	触发器选择寄存器	0x7A0	读/写
tdata1	触发器抽象数据寄存器 1	0x7A1	读/写
tdata2	触发器抽象数据寄存器 2	0x7A2	读/写
tcontrol	全局触发器控制寄存器	0x7A5	读/写
调试模式 CSR			
dcsr	调试模式控制与状态寄存器	0x7B0	读/写
dpc	调试模式 PC 寄存器	0x7B1	读/写
dscratch0	调试模式暂存寄存器 0	0x7B2	读/写
dscratch1	调试模式暂存寄存器 1	0x7B3	读/写
程序计数器 CSR（自定义）⁴			
mpcer	程序计数器事件寄存器	0x7E0	读/写
mpcmr	程序计数器模式寄存器	0x7E1	读/写
mpccr	程序计数器计数寄存器	0x7E2	读/写

请注意，如果对上表中只读属性的任何 CSR 尝试执行写入/置位/清除操作，CPU 将生成非法指令异常。

1.4.2 寄存器

Register 1.1. mvendorid (0xF11)

31	0
0x00000612	
Reset	

MVENDORID 供应商编号。（只读）

Register 1.2. marchid (0xF12)

31	0
0x80000001	
Reset	

MARCHID 架构编号。（只读）

⁴这些自定义机器模式 CSR 已经在 RISC-V 标准为用户保留的地址空间中实现。

Register 1.3. mimpid (0xF13)

MMPID	
31	0
0x00000001	
Reset	

MIMPID 架构编号。(只读)

Register 1.4. mhartid (0xF14)

MHARTID 硬件线程编号。(只读)

Register 1.5. mstatus (0x300)

(reserved)										TW		(reserved)										MPP		(reserved)		MPIE		(reserved)		MIE		(reserved)	
31										22	21	13										12	11	10	8	7	6	4	3	2	0		
0x000										0	0x00										0x0	0x0	0x0	0	0x0	0	0x0	Reset					

MIE 全局机器模式中断使能。(读/写)

MPIE 之前的 **MIE**。(读/写)

MPP 机器之前的特权模式。(读/写)

可能的值:

- 0x0: 用户模式
- 0x3: 机器模式

说明：仅低位可写。由于高位直接绑定低位，写入高位将被忽略。

TW 超时等待。(读/写)

如果该位置 1，用户模式下的 WFI（等待中断）指令将导致非法指令异常。

Register 1.6. misa (0x301)

MXL		(reserved)		Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Reset
31	30	29	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	
0x1		0x0		0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	Reset

MXL 机器 XLEN = 1 (32 位)。(只读)

Z 保留 = 0。(只读)

Y 保留 = 0。(只读)

X 非标准扩展 = 0。(只读)

W 保留 = 0。(只读)

V 保留 = 0。(只读)

U 实现用户模式 = 1。(只读)

T 保留 = 0。(只读)

S 实现监督模式 = 0。(只读)

R 保留 = 0。(只读)

Q 四精度浮点扩展 = 0。(只读)

P 保留 = 0。(只读)

O 保留 = 0。(只读)

N 支持用户级别中断 = 0。(只读)

M 整数乘除法标准扩展 = 1。(只读)

L 保留 = 0。(只读)

K 保留 = 0。(只读)

J 保留 = 0。(只读)

I RV32I 基本 ISA = 1。(只读)

H 虚拟机管理程序扩展 = 0。(只读)

G 其他标准扩展 = 0。(只读)

F 单精度浮点扩展 = 0。(只读)

E RV32E 基本 ISA = 0。(只读)

D 双精度浮点扩展 = 0。(只读)

C 压缩标准扩展 = 1。(只读)

B 保留 = 0。(只读)

A 原子标准扩展 = 0。(只读)

Register 1.7. mtvec (0x305)

BASE								(reserved)				MODE	
31						8	7			2	1	0	
0x000000								0x00				0x1	
Reset													

MODE 仅支持向量模式 **0x1**。(只读)

BASE 异常向量基址的高 24 位为 256 字节对齐。(读/写)

Register 1.8. mscratch (0x340)

MSCRATCH																															
31																															0
0x00000000																															
Reset																															

MSCRATCH 用户自定义的机器暂存寄存器。(读/写)

Register 1.9. mepc (0x341)

MEPC																													
31																													0
0x00000000																													
Reset																													

MEPC 机器陷阱/异常程序计数器。(读/写)

当 CPU 遇到异常时，此域将自动更新为 CPU 将要执行的指令的地址。

Register 1.10. mcause (0x342)

Interrupt Flag										(reserved)										Exception Code																
31	30																													5	4	0				
0	0x00000000																													0x00					Reset	

Exception Code CPU 进入异常时，此域将自动更新为最近的异常或中断的唯一 ID。(读/写)

可能的异常 ID:

- 0x1: PMP 指令访问错误
- 0x2: 非法指令
- 0x3: 硬件断点/观察点或 EBREAK
- 0x5: PMP 读存储器访问错误
- 0x7: PMP 写存储器访问错误
- 0x8: 用户模式环境调用 (ECALL)
- 0xb: 机器模式环境调用

说明: 异常 ID 0x0 (指令地址非对齐) 不存在, 因为 CPU 在取指时始终屏蔽地址的最低位。

Interrupt Flag CPU 进入异常时, 此标志位将自动更新。(读/写)

如果被置位, 则表示最近的陷阱是由中断引起。在异常情况下保持为 0。

说明: 中断控制器将中断编号 1-31 全部用于外部中断源, 而 RISC-V 标准则为内核的内部中断源预留了编号 0-15。

Register 1.11. mtval (0x343)

MTVAL																															
31																															0
0x00000000																															Reset

MTVAL 机器模式异常值。(读/写)

将自动更新为与异常有关的数据, 该数据可能有助于处理该异常。

根据异常编号有以下解读:

- 0x1: 指令虚拟地址错误
- 0x2: 指令 opcode 错误
- 0x5: 存储器读操作的数据地址错误
- 0x7: 存储器写操作的数据地址错误

说明: 该寄存器不支持其他异常 ID 和中断。

Register 1.12. mpcer (0x7E0)

(reserved)											INST_COMP (BRANCH_TAKEN BRANCH JMP_UNCOND STORE LOAD IDLE JMP_HAZARD LD_HAZARD INST CYCLE												
31											11	10	9	8	7	6	5	4	3	2	1	0	
0x000												0	0	0	0	0	0	0	0	0	0	0	Reset

INST_COMP 计数压缩指令。(读/写)

BRANCH_TAKEN 采取分支跳转。(读/写)

BRANCH 计数分支。(读/写)

JMP_UNCOND 计数无条件跳转。(读/写)

STORE 计数存储器写操作。(读/写)

LOAD 计数存储器读操作。(读/写)

IDLE 计数 IDLE 周期。(读/写)

JMP_HAZARD 计数跳转冲突。(读/写)

LD_HAZARD 计数存储器读操作冲突。(读/写)

INST 计数指令。(读/写)

CYCLE 计数时钟周期。(读/写)

注意：每个位选择一个特定事件由计数器递增计数。如果多个事件被选择且同时发生，则计数器只递增 1。

Register 1.13. mpcmr (0x7E1)

(reserved)										COUNT_SAT		COUNT_EN	
31									2	1	0		
0								1	1	Reset			

COUNT_SAT 计数器饱和控制。(读/写)

可能的值：

- 0: 超出最大值上溢
- 1: 超出最大值暂停

COUNT_EN 计数器使能控制。(读/写)

可能的值：

- 0: 禁能
- 1: 使能

Register 1.14. mpccr (0x7E2)

MPCCR	
31	0
0x00000000	
Reset	

MPCCR 程序计数器计数的值。(读/写)

PRELIMINARY

1.5 中断控制器

1.5.1 特性

中断控制器能够捕获、屏蔽来自 RISC-V CPU 外部的中断源，并对中断源的优先级进行动态仲裁。中断控制器具有以下特性：

- 多达 31 个具有唯一 ID (1-31) 的异步中断
- 支持通过读写存储器匹配寄存器进行配置
- 15 个优先级级别，可以分配给不同的中断
- 支持电平触发或边沿触发的中断源
- 可配置的全局阈值，用于屏蔽优先级较低的中断
- 与异常向量地址偏移量匹配的中断 ID

1.5.2 功能描述

每个中断 ID 都有 5 个属性：

1. 使能状态 (0-1):

- 决定是否允许由 CPU 捕获和处理中断。
- 通过写入 `INT_ENABLE_REG` 相应的域进行配置。

2. 类型 (0-1):

- 在中断信号的上升沿使能门锁状态。
- 通过写入 `INT_TYPE_REG` 相应的域进行配置。
- 类型保持为 0 的中断称为“电平”类型中断。
- 类型保持为 1 的中断称为“边沿”类型中断。

3. 优先级 (1-15):

- 当有多个中断在等待时，决定 CPU 先处理哪一个中断。
- 通过写入中断 ID n (1-31) 的 `INT_PRIORITY_n_REG` 进行配置。
- 优先级为零或小于 `INT_THRESH_REG` 指定阈值的中断将被屏蔽。
- 优先级最低为 1，最高为 15。
- 具有相同优先级的中断通过其 ID 静态确定优先级，ID 越小，优先级越高。

4. 等待状态 (0-1):

- 反映已使能且未被屏蔽的中断信号被捕获时的状态。
- 通过读取 `INT_EIP_REG` 中的相应位获得每个中断 ID 的等待状态。
- 如果没有更高优先级的中断在等待，则当前在等待的中断将导致 CPU 进入异常。
- 如果在等待的中断抢占 CPU 并导致其跳转到相应的异常向量地址，则称该中断为“已声明”。
- 所有在等待的中断都为“未声明”。

5. 清除状态 (0-1):

- 切换此属性将仅清除已声明的边沿类型中断的等待状态。
- 通过先置位然后清零 `INT_CLEAR_REG` 中的相应位进行切换。
- 电平类型的中断的等待状态不受切换操作的影响，而必须从中断源中清除。
- 未声明的边沿类型中断的等待状态可以被清空，方法是先清零 `INT_ENABLE_REG` 中的相应位再置位 `INT_CLEAR_REG` 中的相同位。

当 CPU 处理在等待的中断时，会进行以下操作：

- 将当前未执行指令的地址保存在 `mepc` 中，以便之后恢复执行。
- 将 `mcause` 的值更新为正在处理的中断 ID。
- 将 `MIE` 的状态复制到 `MPIE`，然后清零 `MIE`，从而全局禁用中断。
- 通过跳转到 `mtvec` 中存储的地址的字对齐偏移量进入异常。

表 1-3 列出了每个中断 ID 及其对应的异常向量地址。简而言之，中断 $ID = i$ 的字对齐的异常地址 = $(mtvec + 4i)$ 。

说明： $ID = 0$ 不可用，不能用于捕获中断，这是因为相应的异常向量地址 $(mtvec + 0x00)$ 已经预留给异常。

表 1-3. 中断 ID 与异常向量地址

ID	地址	ID	地址	ID	地址	ID	地址
0	NA	8	$mtvec + 0x20$	16	$mtvec + 0x40$	24	$mtvec + 0x60$
1	$mtvec + 0x04$	9	$mtvec + 0x24$	17	$mtvec + 0x44$	25	$mtvec + 0x64$
2	$mtvec + 0x08$	10	$mtvec + 0x28$	18	$mtvec + 0x48$	26	$mtvec + 0x68$
3	$mtvec + 0x0c$	11	$mtvec + 0x2c$	19	$mtvec + 0x4c$	27	$mtvec + 0x6c$
4	$mtvec + 0x10$	12	$mtvec + 0x30$	20	$mtvec + 0x50$	28	$mtvec + 0x70$
5	$mtvec + 0x14$	13	$mtvec + 0x34$	21	$mtvec + 0x54$	29	$mtvec + 0x74$
6	$mtvec + 0x18$	14	$mtvec + 0x38$	22	$mtvec + 0x58$	30	$mtvec + 0x78$
7	$mtvec + 0x1c$	15	$mtvec + 0x3c$	23	$mtvec + 0x5c$	31	$mtvec + 0x7c$

在跳转到异常向量之后，执行流程取决于软件实现，但一般来说该中断将在某个中断服务程序 (ISR) 中被处理（并清除），然后在 CPU 遇到 `MRET` 指令后恢复正常程序流。

执行 `MRET` 指令后，CPU 将进行以下操作：

- 将 `MPIE` 的状态复制回 `MIE`，然后清零 `MPIE`。这意味着，如果之前置位了 `MPIE`，则执行 `MRET` 后 `MIE` 将被置位，进而全局使能中断。
- 跳转到 `mepc` 中存储的地址，然后恢复执行。

软件可以在 ISR 内部实现中断嵌套，具体请参考章节 1.5.3。

中断控制器具有以下行为特点：

- 仅当中断具有非零优先级、大于或等于阈值寄存器中的值时，它才会反映在 `INT_EIP_REG` 中。
- 如果一个中断反映在 `INT_EIP_REG` 中但是还未被处理，则可以通过降低它的优先级或提高全局阈值将其屏蔽（进而防止 CPU 对其进行处理）。

- 如果一个中断反映在 `INT_EIP_REG` 中，要清除它（防止被处理），则必须将其禁用（如果是边沿属性的中断则需要清除）。

1.5.3 建议操作

1.5.3.1 延迟

配置中断控制器时应考虑延迟问题。

在稳态操作中，中断控制器的等待时间固定为 4 个周期。稳态操作的意思是最近没有对中断控制器寄存器作任何更改。这意味着一个中断从被中断控制器断言到被 CPU 开始处理刚好消耗 4 个周期。这也意味着，在抢占发生之前，CPU 最多可以执行 5 条指令。

当寄存器被修改时，中断控制器会进入临时状态，然后需要最多 4 个周期才能再次进入稳态。在临时状态期间，中断的顺序可能无法预测，因此，需要软件采取一些安全措施以避免任何同步问题。

还须注意的是，中断控制器的配置寄存器位于 APB 地址范围内，因此对这些寄存器的读写访问可能需要消耗几个周期。

考虑到上述特征，建议用户在修改中断控制器寄存器时遵循以下操作顺序：

1. 保存 `MIE` 的状态，然后将其清零
2. 通过“读-修改-写”的方式写中断控制器寄存器
3. 执行 `FENCE` 指令以等待所有未完成的写操作完成
4. 最后，恢复 `MIE` 的状态

如上述步骤显示，建议用户在配置中断控制器寄存器之前先全局禁用中断 (`MIE=0`)，然后立即恢复 `MIE`。

执行完上述操作后，中断控制器将恢复稳态操作。

1.5.3.2 配置流程

默认情况下，`mstatus` 里的 `MIE` 为 0，即全局禁用中断。在中断堆栈初始化（包括将 `mtvec` 设置为中断向量地址）完成之后，软件必须将 `MIE` 置为 1。

在正常情况下，如果要使能某个中断 n ，可以遵循以下步骤：

1. 保存 `MIE` 的状态，然后将其清零
2. 根据中断的类型（边沿/电平），置位或取消置位 `INT_TYPE_REG` 中的第 n 个位
3. 通过写入 `INT_PRIORITY_n_REG` 指定优先级（最低为 1，最高为 15）
4. 置位 `INT_ENABLE_REG` 中的第 n 个位
5. 执行 `FENCE` 指令
6. 恢复 `MIE` 的状态

当一个或多个中断在等待时，CPU 将确认（声明）最高优先级的中断，然后跳转到与该中断 ID 相对应的异常向量地址。软件可以通过读取 `mcause` 来推断异常类型（`mcause(31)` 为 1 代表中断，为 0 代表异常）和中断 ID（`mcause(4-0)` 提供中断或异常的 ID）。如果异常向量中的每个表项都是指向不同异常处理程序的跳转指令，则软件无需做此推断。最后，异常处理程序会将程序指引到该中断相应的 ISR。

进入 ISR 后，如果中断为边沿类型，则软件必须切换 `INT_CLEAR_REG` 中的第 n 个位，如果是电平类型中断，则必须清除相应的中断源。

软件还可以更新 `INT_THRESH_REG` 的值并置位 `MIE` 来让更高优先级的中断抢占当前 ISR（即嵌套），但是，在此之前，必须先保存所有状态 CSR（`mepc`、`mstatus`、`mcause` 等），这是由于发生嵌套时状态 CSR 的值会被覆盖。之后，在退出 ISR 时，再恢复这些 CSR 的值。

最后，程序从 ISR 返回到异常处理程序之后，可以执行 `MRET` 指令以恢复正常程序流。

如果不再需要中断 `n` 并且需要将其禁用，则可以遵循以下操作步骤：

1. 保存 `MIE` 的状态，然后将其清零
2. 读取 `INT_EIP_REG` 检查中断是否在等待
3. 置位/取消置位 `INT_ENABLE_REG` 中的第 `n` 个位
4. 如果中断属于边沿类型并且在等待，则必须切换 `INT_CLEAR_REG` 中的第 `n` 个位以清空它的等待状态
5. 执行 `FENCE` 指令
6. 恢复 `MIE` 的状态

以上只是建议的操作方案，实际操作由软件实现决定。

1.5.4 寄存器列表

本小节的所有地址均为相对于中断控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
<code>INT_ENABLE_REG</code>	使能 CPU 处理中断	0x0104	读/写
<code>INT_TYPE_REG</code>	指定中断类型为电平/边沿类型	0x0108	读/写
<code>INT_CLEAR_REG</code>	写入清除“脉冲”类型中断	0x010C	读/写
<code>INT_EIP_REG</code>	外部中断的等待状态	0x0110	只读
<code>INT_PRIORITY_1_REG</code>	设置中断 ID=1 的优先级	0x0118	读/写
<code>INT_PRIORITY_2_REG</code>	设置中断 ID=2 的优先级	0x011C	读/写
<code>INT_PRIORITY_3_REG</code>	设置中断 ID=3 的优先级	0x0120	读/写
<code>INT_PRIORITY_4_REG</code>	设置中断 ID=4 的优先级	0x0124	读/写
<code>INT_PRIORITY_5_REG</code>	设置中断 ID=5 的优先级	0x0128	读/写
<code>INT_PRIORITY_6_REG</code>	设置中断 ID=6 的优先级	0x012C	读/写
<code>INT_PRIORITY_7_REG</code>	设置中断 ID=7 的优先级	0x0130	读/写
<code>INT_PRIORITY_8_REG</code>	设置中断 ID=8 的优先级	0x0134	读/写
<code>INT_PRIORITY_9_REG</code>	设置中断 ID=9 的优先级	0x0138	读/写
<code>INT_PRIORITY_10_REG</code>	设置中断 ID=10 的优先级	0x013C	读/写
<code>INT_PRIORITY_11_REG</code>	设置中断 ID=11 的优先级	0x0140	读/写
<code>INT_PRIORITY_12_REG</code>	设置中断 ID=12 的优先级	0x0144	读/写
<code>INT_PRIORITY_13_REG</code>	设置中断 ID=13 的优先级	0x0148	读/写
<code>INT_PRIORITY_14_REG</code>	设置中断 ID=14 的优先级	0x014C	读/写
<code>INT_PRIORITY_15_REG</code>	设置中断 ID=15 的优先级	0x0150	读/写
<code>INT_PRIORITY_16_REG</code>	设置中断 ID=16 的优先级	0x0154	读/写
<code>INT_PRIORITY_17_REG</code>	设置中断 ID=17 的优先级	0x0158	读/写
<code>INT_PRIORITY_18_REG</code>	设置中断 ID=18 的优先级	0x015C	读/写
<code>INT_PRIORITY_19_REG</code>	设置中断 ID=19 的优先级	0x0160	读/写

名称	描述	地址	访问
INT_PRIORITY_20_REG	设置中断 ID=20 的优先级	0x0164	读/写
INT_PRIORITY_21_REG	设置中断 ID=21 的优先级	0x0168	读/写
INT_PRIORITY_22_REG	设置中断 ID=22 的优先级	0x016C	读/写
INT_PRIORITY_23_REG	设置中断 ID=23 的优先级	0x0170	读/写
INT_PRIORITY_24_REG	设置中断 ID=24 的优先级	0x0174	读/写
INT_PRIORITY_25_REG	设置中断 ID=25 的优先级	0x0178	读/写
INT_PRIORITY_26_REG	设置中断 ID=26 的优先级	0x017C	读/写
INT_PRIORITY_27_REG	设置中断 ID=27 的优先级	0x0180	读/写
INT_PRIORITY_28_REG	设置中断 ID=28 的优先级	0x0184	读/写
INT_PRIORITY_29_REG	设置中断 ID=29 的优先级	0x0188	读/写
INT_PRIORITY_30_REG	设置中断 ID=30 的优先级	0x018C	读/写
INT_PRIORITY_31_REG	设置中断 ID=31 的优先级	0x0190	读/写
INT_THRESH_REG	设置中断优先级阈值	0x0194	读/写

1.5.5 寄存器

本小节的所有地址均为相对于中断控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 1.15. INT_ENABLE_REG (0x0104)

INT_ENABLE		(reserved)
31	1	0
0x00000000		0
		Reset

INT_ENABLE[n] 置位第 n 个位使能中断 n 。（读/写）

- 0：禁能
- 1：使能

Register 1.16. INT_TYPE_REG (0x0108)

INT_TYPE		(reserved)
31	1	0
0x00000000		0
		Reset

INT_TYPE[n] 置位第 n 个位捕获中断 n 的上升沿。（读/写）

- 0：电平类型（检测信号电平）
- 1：脉冲类型（检测上升沿）

Register 1.17. INT_CLEAR_REG (0x010C)

INT_CLEAR																(reserved)	
31															1	0	Reset
0x00000000																0	

INT_CLEAR[n] 置位第 n 个位清除中断 n 的等待状态。(读/写)

仅对“脉冲”类型的中断有效，“电平”类型中断应在中断源进行清除。

注意置位后需要手动切换至 0。

- 0: 无关项
- 1: 清除等待状态

Register 1.18. INT_EIP_REG (0x0110)

INT_EIP																(reserved)	
31															1	0	Reset
0x00000000																0	

INT_EIP[n] 读取第 n 个位获得中断 n 的等待状态。(只读)

只有被使能的且阈值以上的中断才会反映在该域中。

- 0: 不在等待
- 1: 在等待

Register 1.19. INT_PRIORITY_ n _REG (n : 1-31) (0x0114+4* n)

(reserved)												INT_PRIORITY _n				Reset
31											4	3			0	
0x00000000												0x0				

INT_PRIORITY_ n 写入 4 位数值到第 n 个寄存器配置中断 n 的优先级。(读/写)

说明：不管阈值是多少，优先级为 0 的中断都会被屏蔽。

Register 1.20. INT_THRESH_REG (0x0194)

(reserved)																INT_THRESH			
31													4	3			0		
0x00000000																0x0		Reset	

INT_THRESH 写入 4 位的数值配置所有中断的全局优先级阈值。(读/写)

所有优先级低于阈值的中断都会被屏蔽。

说明：不管阈值是多少，优先级为 0 的中断都会被屏蔽。

1.6 调试

1.6.1 概述

本节介绍如何调试和测试在 CPU 内核上运行的软件。调试功能由标准 JTAG 管脚提供，并符合 RISC-V 外部调试支持规范版本 0.13 (RISC-V External Debug Support Specification version 0.13)。

图 1-2 为外部调试系统架构图。

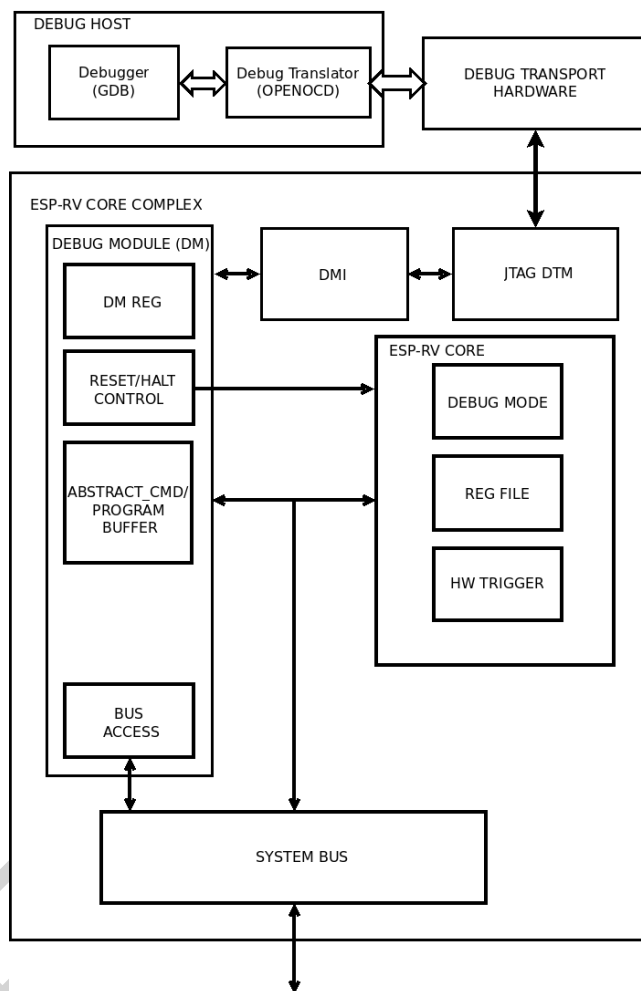


图 1-2. 调试系统架构

用户与运行调试器 (Debugger, 例如 GDB) 的调试主机 (DEBUG HOST, 例如笔记本电脑) 进行交互。调试器通过调试转换器 (Debug Translator, 可能包含硬件驱动, 例如 OPENOCD) 与调试传输硬件 (DEBUG TRANSPORT, 例如 Olimex USB-JTAG 适配器) 进行通信。调试传输硬件通过标准 JTAG 接口将调试主机连接到 ESP-RV 内核的调试传输模块 (JTAG DTM)。JTAG DTM 使用调试模块接口 (DMI) 提供对调试模块 (DM) 的访问。

DM 允许调试器暂停内核。抽象命令提供对 GPR (通用寄存器) 的访问。程序缓冲区允许调试器在内核上执行任意代码, 从而读取 CPU 内核的其他运行状态。CPU 内核的其他运行状态也可以由其他抽象命令读取。ESP-RV 内核带有一个支持 8 个触发器的触发器模块。当满足触发条件时, 内核将自发暂停并通知调试模块。

系统总线访问的 block 无需使用 RISC-V 内核即可访问存储器和外设寄存器。

1.6.2 特性

基础调试功能具有以下特性:

- 向调试器提供有关实现的必要信息
- 支持暂停和恢复 CPU 内核
- CPU 内核寄存器（包括 CSR）可以由调试器读取/写入
- CPU 可以从复位后执行的第一条指令开始就被调试
- 可以通过调试器复位 CPU 内核
- 可以在软件断点（植入的断点指令）上暂停 CPU
- 硬件单步调试
- 通过程序缓冲区在暂停的 CPU 中执行任意指令。支持 16 字的程序缓冲区。
- 支持系统总线的字对齐地址访问
- 支持八个硬件触发器（可用作断点/观察点），具体见章节 1.7

1.6.3 功能描述

调试机制遵守 RISC-V 外部调试支持规范版本 0.13。有关调试功能的详细介绍，请参考 RISC-V 外部调试支持规范。

1.6.4 寄存器列表

下表列出了 ESP-RV 内核支持的调试 CSR。

名称	描述	地址	访问
dcsr	调试控制和状态寄存器	0x7B0	读/写
dpc	调试 PC 寄存器	0x7B1	读/写
dscratch0	调试暂存寄存器 0	0x7B2	读/写
dscratch1	调试暂存寄存器 1	0x7B3	读/写

所有调试模块寄存器的实现均符合 RISC-V 外部调试支持规范版本 0.13。请参考 RISC-V 外部调试支持规范获取详细信息。

1.6.5 寄存器

以下是 ESP-RV 内核支持的调试 CSR 的详细描述。

Register 1.21. dcsr (0x7B0)

xdebugver				reserved												ebreakm		reserved		ebreaku		reserved		stopcount		stoptime		cause		reserved		step		prv	
31	28	27					16	15	14	13	12	11	10	9	8		6	5		3	2	1	0												
4			0					0	0	0	0	0	0	0	0	0	0		0		0	0	0	Reset											

xdebugver 调试版本。(只读)

- 4: 存在外部调试支持

ebreakm 置位后，机器模式中的 ebreak 指令进入调试模式。(读/写)

ebreaku 置位后，用户/程序模式中的 ebreak 指令进入调试模式。(读/写)

stopcount 此域没有实现。调试器会始终读出 0。(只读)

stoptime 此性能没有实现。调试器会始终读出 0。(只读)

cause 说明进入调试模式的原因。当单个周期中有多个原因导致进入调试模式，会反映出具有最高优先级数值的那个原因。(只读)

1. 执行了一条 ebreak 指令（优先级 3）
2. 触发模块引起暂停（优先级 4）
3. haltreq 被置位（优先级 2）
4. step 被置位导致 CPU 单步执行（优先级 1）

其他值保留供以后使用。

step 当被置位且不处于调试模式时，内核将仅执行单个指令，然后进入调试模式。当该位置 1 时，中断被使能*。如果指令由于异常而未能完成，则内核将在执行异常处理程序之前立即进入调试模式，并置位相应的异常寄存器。(读/写)

prv 保存 CPU 进入调试模式时候的特权级别。退出调试模式时，调试器可以更改此值以改变内核的特权级别。仅支持 0x3（机器模式）和 0x0（用户模式）。

* 注意：与 RISC-V 调试规格版本 0.13 不同。

Register 1.22. dpc (0x7B1)

dpc																													
31																													0
0																													Reset

dpc 进入调试模式后，dpc 将写入遇到异常的指令的虚拟地址。恢复执行时，CPU 内核的 PC 将更新为 dpc 保存的虚拟地址。调试器可以写入 dpc 配置 CPU 恢复执行的位置。(读/写)

Register 1.23. dscratch0 (0x7B2)

dscratch0	
31	0
0	
Reset	

dscratch0 供调试模块内部使用。(读/写)

Register 1.24. dscratch1 (0x7B3)

dscratch1	
31	0
0	
Reset	

dscratch1 供调试模块内部使用。(读/写)

1.7 硬件触发器

1.7.1 特性

硬件触发器模块提供了断点和观察点功能，供调试使用。硬件触发器具有以下特性：

- 8 个独立触发单元
- 每个单元都可以配置为匹配程序计数器的地址或存储器访问地址
- 可以通过引起断点异常来抢占执行
- 可以暂停执行并将控制权转移给调试器
- 支持 NAPOT（2 的幂次方对齐）地址编码

1.7.2 功能描述

硬件触发器模块提供了 4 个 CSR，见[寄存器列表](#)。其中，[tdata1](#) 和 [tdata2](#) 是抽象 CSR，也就是说它们是用于访问某个触发单元中的内部寄存器的影子寄存器，一次访问一个触发单元。

要选择特定的触发单元，需要将相应的编号 (0-7) 写入 [tselect](#) CSR。当写入有效数值时，抽象 CSR [tdata1](#) 和 [tdata2](#) 将自动匹配该触发单元的内部寄存器。每个触发单元都有两个内部寄存器，即 [mcontrol](#) 和 [maddress](#)，它们分别与 [tdata1](#) 和 [tdata2](#) 匹配。

向 [tselect](#) 写入超过最大编号的数值时会导致该数值被裁剪为最大的编号，此编号可以被读回。这个特性可用于枚举初始化期间或使用调试器时可用的触发器。

由于软件或调试器可能需要知道所选触发器的类型以便正确解读 [tdata1](#) 和 [tdata2](#)，因此 [tdata1](#) 的 4 个位 (31-28) 对所选触发器的类型进行了编码。此域为只读访问属性，并且值始终为 0x2，代表匹配类型触发器，因此，可以推断 [tdata1](#) 和 [tdata2](#) 会通过 [mcontrol](#) 和 [maddress](#) 被解读。RISC-V 调试规范 v0.13 提供了其他可能值的信息，但是该触发模块仅支持 0x2 类型。

一旦选定了触发单元，就可以通过置位 [mcontrol](#) CSR ([tdata1](#)) 中相应的域并将目标地址写入 [maddress](#) CSR ([tdata2](#)) 来对该触发单元进行配置。

通过写入 [mcontrol](#) 的 [action](#) 域，可以将每个触发单元配置为引起断点异常或进入调试模式。该域只能从调试器写入，因此默认情况下，触发器（如果启用）将引起断点异常。

每个触发单元的 [mcontrol](#) 都有一个 [hit](#) 域。在 CPU 暂停或进入异常后，通过读取该域可以查明是否是触发单元触发了。触发器触发后该域会立即被置位，但在恢复操作之前必须被手动清零，虽然不清零不会影响正常执行。

每个触发单元仅支持地址匹配，该地址可以是存储器访问地址，也可以是指令的虚拟地址。通过写入所选触发单元的 [maddress](#) ([tdata2](#)) CSR，可以指定区域的地址和大小。大于 1 个字节的区域大小通过 NAPOT 编码（见[表 1-6](#)）指定，并通过置位 [mcontrol](#) 中 [match](#) 域来使能。注意，根据定义，NAPOT 编码地址的起始地址与区域大小对齐（即，是区域大小的整数倍）。

表 1-6. NAPOT 编码的 [maddress](#)

maddress (31-0)	起始地址	大小（字节）
aaa...aaaaaaaa0	aaa...aaaaaaaa0	2
aaa...aaaaaaaa01	aaa...aaaaaaaa00	4
aaa...aaaaaaaa011	aaa...aaaaaaaa000	8
aaa...aaaaaaa0111	aaa...aaaaaaa0000	16

....		
a01...1111111111	a00...0000000000	2 ³¹

tcontrol CSR 对所有触发单元都是通用的。在机器模式下，当程序在异常处理程序中执行时，该寄存器可用于阻止触发器重复引起异常。默认情况下 ISR 内部的断点异常也被禁用，但是，出于调试目的，可以在进入 ISR 之前手动使能断点异常。如果将触发器配置为进入调试模式，则此 CSR 不相关。

1.7.3 触发执行流程

当触发器触发引起硬件线程暂停并进入调试模式时 ($action = 1$):

- **dpc** 被设置为当前 PC（在解码阶段）
- **dcsr** 的 **cause** 域被设置为 2，表示暂停是由于触发器触发引起
- 与触发的触发器对应的 **hit** 域被置位

当触发器触发引起硬件线程进入异常时 ($action = 0$):

- **mepc** 被设置为当前 PC（在解码阶段）
- **mcause** 被设置为 3，即断点异常
- **mpte** 被设置为异常发生之前的 **mte** 的值
- **mte** 被设置为 0
- 与触发的触发器对应的 **hit** 域被置位

说明：如果两个触发器同时触发，一个 $action = 0$ ， $action = 1$ ，则硬件线程会暂停并进入调试模式。

1.7.4 寄存器列表

下表列出了 CPU 可访问的触发模块 CSR，只有在机器模式下才可以对它们进行读写。

名称	描述	地址	访问
tselect	触发器选择寄存器	0x7A0	读/写
tdata1	触发器抽象数据寄存器 1	0x7A1	读/写
tdata2	触发器抽象数据寄存器 2	0x7A2	读/写
tcontrol	全局触发器控制寄存器	0x7A5	读/写

1.7.5 寄存器

Register 1.25. **tselect** (0x7A0)

(reserved)			tselect	
31	3	2	0	
0x00000000			0x0	Reset

tselect 触发器单元编号 (0-7)。(读/写)

Register 1.26. tdata1 (0x7A1)

type		dmode		data	
31	28	27	26	0	
0x2		0		0x3e00000	

Reset

type 触发器类型。(只读)

仅支持匹配类型 (0x2)，此域保留。

dmode 如果某触发器正在被调试器使用，则此域置为 1。（读/写*）

- 0: 在调试模式和机器模式下都能写入 tdata1 和 tdata2
- 1: 只有在调试模式下才能写入 tdata1 和 tdata2。其他模式下的写操作将被忽略。

* 说明：仅支持调试模式下的写操作。

data 保存抽象 tdata1 的内容。(读/写)

由于仅支持匹配类型 (0x2) 触发器，此域将始终被解读为 `mcontrol` 的域。

Register 1.27. tdata2 (0x7A2)

31	0
0x00000000	

Reset

tdata2 保存抽象 tdata2 的内容。(读/写)

由于仅支持匹配类型 (0x2) 触发器，此域将始终被解读为 `maddress`。

Register 1.28. tcontrol (0x7A5)

(reserved)								mpte		(reserved)								mte					
31								8		7		6		1								0	
0x000000								0				0x00								0		Reset	

mpte 机器模式下前一个触发器使能域。(读/写)

- 当 CPU 在机器模式下进入异常，`mte` 的值会自动写入此域。
- 当 CPU 执行 MRET，此域的值会返回 `mte`，此域变为 0。

mte 机器模式下触发器使能域。(读/写)

- 当 CPU 在机器模式下进入异常, 此域的值会自动写入 `mpte`, 然后此域变为 0, 并且 `action=0` 的触发器被全局禁用。
- 当 CPU 执行 MRET, `mpte` 的值会自动返回此域。

Register 1.29. mcontrol (0x7A1)

(reserved)				dmode		(reserved)				hit		(reserved)				action		(reserved)		match		m		(reserved)		u		execute		store		load			
31				28	27	26				21	20	19				16	15				12	11	10				7	6	5	4	3	2	1	0	
0x2				0		0x1f				0		0				0		0		0		0		0		0		0		0		0		Reset	

dmode 与 tdata1 的 dmode 一致。

hit 如果选定的触发器之前触发过，则此域为 1。（读/写）
此域必须手动清零。

action 配置选定的触发器在触发时进行以下操作。（读/写）
有效选项为：

- 0x0：引起断点异常
- 0x1：进入调试模式（仅当 dmode = 1 时有效）

说明：写入无效数值会导致此域变为默认值 0x0。

match 配置触发器进行数据/指令地址的下匹配操作。（读/写）
有效选项为：

- 0x0：严格字节匹配，即与访问中某个字节对应的地址必须严格匹配 maddress 的值。
- 0x1：NAPOT 匹配，即访问中至少有一个字节处于 maddress 中规定的 NAPOT 区域。

说明：写入超过最大值的数值会被裁剪为最大值 0x1。

m 置位使选定的触发器在机器模式下操作。（读/写）

u 置位使选定的触发器在用户模式下操作。（读/写）

execute 置位使选定的触发器在 CPU 执行具有匹配的虚拟地址的指令之前触发。（读/写）

store 置位使选定的触发器在 CPU 执行具有匹配的数据地址的存储器写操作之前触发。（读/写）

load 置位使选定的触发器在 CPU 执行具有匹配的数据地址的存储器读操作之前触发。（读/写）

Register 1.30. maddress (0x7A2)

maddress																															
31																															0
0x00000000																															
Reset																															

maddress 选定的触发器执行匹配操作时使用的地址。（读/写）
当 mcontrol 中的 match=1 时由 NAPOT 解码。

1.8 存储器保护

1.8.1 概述

CPU 内核包含一个物理存储器保护单元，可以供软件设置存储器访问特权（读、写、执行权限）。该物理存储器保护单元不完全等同于 RISC-V 指令集手册 V1.10 第二卷“特权架构”中描述的 PMP，下文会详细描述不同之处。

如需了解 RISC-V PMP 的更多信息，请参考 RISC-V 指令集手册 V1.10 第二卷“特权架构”。

1.8.2 特性

PMP 单元用于控制对物理存储器的访问。它支持 16 个区域，可以对最小 4 个字节大小的区域进行权限设定。ESP32-C3 芯片的物理存储器保护单元与 RISC-V 规范中定义的 PMP 的不同之处在于：

- 不支持静态优先级，即不支持重叠区域
- 支持的最大 NAPOT 范围是 1 GB

根据 RISC-V 特权架构规范，PMP 表项是静态优先级排序的，并且，与存储器访问地址的任意字节匹配的最小编号的 PMP 表项将决定该访问是否成功。这意味着，当任意地址匹配多个 PMP 表项，即多个 PMP 表项的重叠区域时，编号最小的 PMP 表项将决定该访问是否成功。

但是，ESP32-C3 的 RISC-V CPU PMP 单元并未实现静态优先级。因此，软件应确保所有使能的 PMP 表项都有唯一的区域，即它们之间没有重叠区域。如果软件仍试图对具有重叠区域且权限相互矛盾的多个 PMP 表项进行配置，只要访问与其中一个 PMP 表项匹配，则访问成功。如果访问与所有使能的 PMP 表项都不匹配，则会产生一个异常。

1.8.3 功能描述

软件可以设置 PMP 单元的配置和地址寄存器，以保存错误并确保安全执行。PMP CSR 只能在机器模式下进行配置。写入、读取和执行权限检测一旦被使能，则将根据 16 个 `pmpcfgX` 和 `pmpaddrX` 寄存器（见[寄存器列表](#)）中的配置值作用于用户模式下的所有存储器访问。

默认情况下，PMP 允许机器模式下的所有存储器访问，而撤销用户模式下的所有访问。这意味着必须通过 `pmpcfg` 和 `pmpaddr` 寄存器（见[寄存器列表](#)）设置用户模式可以访问的地址范围和有效权限，以确保访问成功。但是在机器模式下没有此要求，因为机器模式下默认 PMP 允许所有访问。如果在机器模式下也需要 PMP 检测，则软件可以将所需 PMP 表项的锁定位置位来使能权限检测。锁定位一旦置位，就只能通过 CPU 复位被清零。

如果从存储器区域提取指令而没有执行权限，则会在处理器级别生成异常，并且在 `mcause` CSR 中异常原因被设置为指令访问错误。同样，任何没有有效读/写权限的读写访问都将生成异常，并且 `mcause` 会更新为读取存储器访问错误或写入存储器访问错误。如果发生存储器读写异常，则存储器访问地址会更新到 `mtval` CSR 中。

1.8.4 寄存器列表

下表列出了 CPU 可访问的 PMP CSR，只有在机器模式下才可以对它们进行读写。

名称	描述	地址	访问
<code>pmpcfg0</code>	物理存储器保护配置寄存器	0x3A0	读/写
<code>pmpcfg1</code>	物理存储器保护配置寄存器	0x3A1	读/写
<code>pmpcfg2</code>	物理存储器保护配置寄存器	0x3A2	读/写
<code>pmpcfg3</code>	物理存储器保护配置寄存器	0x3A3	读/写

名称	描述	地址	访问
pmpaddr0	物理存储器保护地址寄存器	0x3B0	读/写
pmpaddr1	物理存储器保护地址寄存器	0x3B1	读/写
pmpaddr2	物理存储器保护地址寄存器	0x3B2	读/写
pmpaddr3	物理存储器保护地址寄存器	0x3B3	读/写
pmpaddr4	物理存储器保护地址寄存器	0x3B4	读/写
pmpaddr5	物理存储器保护地址寄存器	0x3B5	读/写
pmpaddr6	物理存储器保护地址寄存器	0x3B6	读/写
pmpaddr7	物理存储器保护地址寄存器	0x3B7	读/写
pmpaddr8	物理存储器保护地址寄存器	0x3B8	读/写
pmpaddr9	物理存储器保护地址寄存器	0x3B9	读/写
pmpaddr10	物理存储器保护地址寄存器	0x3BA	读/写
pmpaddr11	物理存储器保护地址寄存器	0x3BB	读/写
pmpaddr12	物理存储器保护地址寄存器	0x3BC	读/写
pmpaddr13	物理存储器保护地址寄存器	0x3BD	读/写
pmpaddr14	物理存储器保护地址寄存器	0x3BE	读/写
pmpaddr15	物理存储器保护地址寄存器	0x3BF	读/写

1.8.5 寄存器

PMP 单元实现了 RISC-V 指令集手册 V1.10 第二卷“特权架构”中定义的所有 pmpcfg0-3 和 pmpaddr0-15 CSR。

2 通用 DMA 控制器 (GDMA)

2.1 概述

通用直接存储访问 (General Direct Memory Access, GDMA) 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输。软件可以在无需 CPU 干预的情况下通过 GDMA 快速搬移数据，从而降低了 CPU 的工作负载，提高了效率。

ESP32-C3 GDMA 共有 6 个独立的通道，其中包括 3 个发送通道和 3 个接收通道。这 6 个通道被支持 GDMA 功能的外设所共享，用户可以将通道分配给任何支持 DMA 功能的外设。这些外设包括：SPI2，UHCIO (UART0/UART1)，I2S，AES，SHA 和 ADC。UART0 与 UART1 共用一个 UHCIO 接口。

GDMA 支持通道间固定优先级及轮询仲裁以管理外设不同的带宽需求。

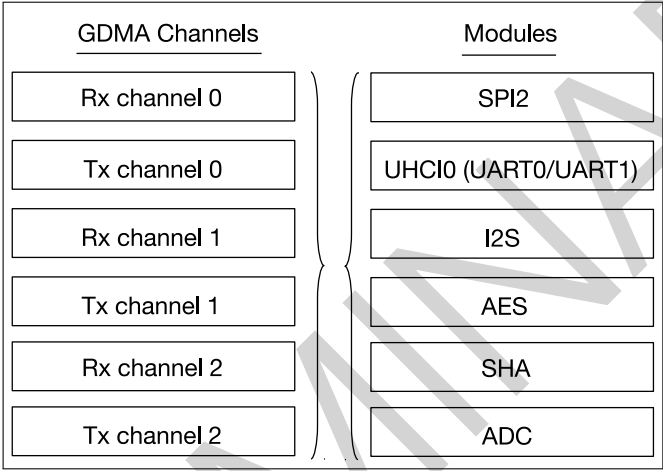


图 2-1. 具有 GDMA 功能的模块和 GDMA 通道

2.2 特性

GDMA 控制器具有以下几个特点：

- AHB 总线架构
- 数据传输以字节为单位，传输数据量可软件编程
- 支持链表
- 访问内部 RAM 时，支持 INCR burst 传输
- GDMA 能够访问的内部 RAM 最大地址空间为 384 KB
- 包含 3 个 TX、3 个 RX 通道
- 任一通道支持可配置的外设选择
- 通道间固定优先级及轮询仲裁

2.3 架构

ESP32-C3 中所有需要进行高速数据传输的模块都具有 GDMA 功能。GDMA 控制器与 CPU 的数据总线使用相同的地址空间访问内部 RAM。图 2-2 为 GDMA 引擎基本架构图。

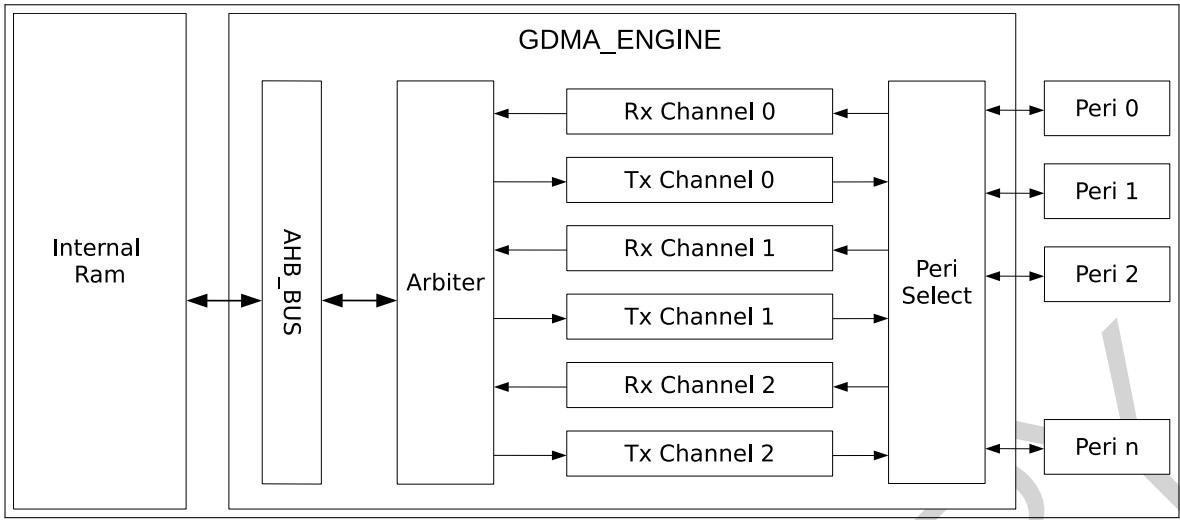


图 2-2. GDMA 引擎的架构

GDMA 引擎共有 6 个独立的通道，其中包括 3 个发送通道和 3 个接收通道。每个通道可选择与不同的外设相连，从而实现通道资源被外设共享。GDMA 引擎通过 AHB_BUS 将数据存入内部 RAM 或者将数据从内部 RAM 取出。内部 RAM 的具体使用范围详见章节 3 系统和存储器。软件可以通过挂载链表的方式来使用 GDMA 引擎。链表本身须存储在片内 RAM 中，包括 outlink n 与 inlink n ，本文以 n 来表示通道号， n 为 0 ~ 2。GDMA 从片内 RAM 中取得链表，然后根据 outlink n 中的内容将相应 RAM 中的数据发送出去，也可根据 inlink n 中的内容将接收的数据存入指定 RAM 地址空间。

2.4 功能描述

2.4.1 链表

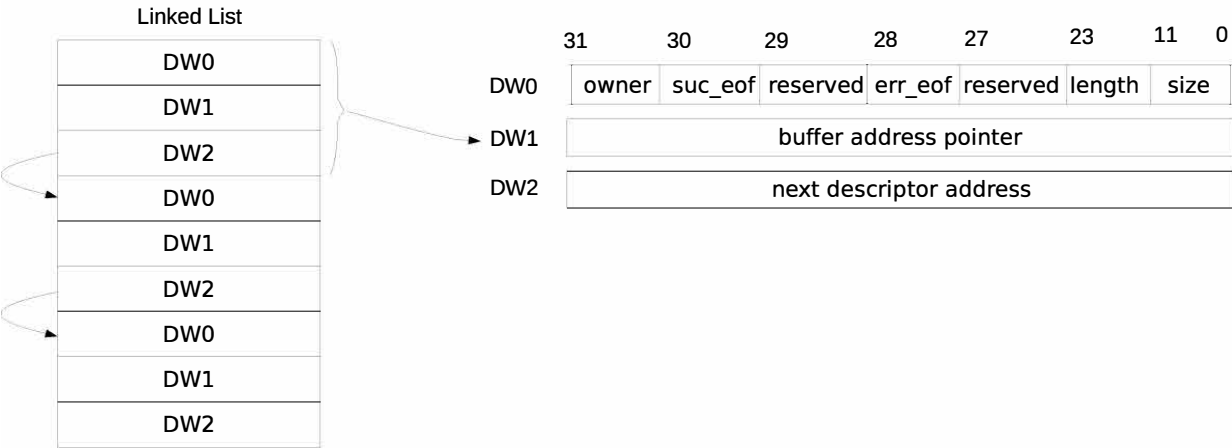


图 2-3. 链表结构图

图 2-3 所示为链表的结构图。发送链表与接收链表结构相同。每个链表由一个或者若干个描述符构成，一个描述符由 3 个字组成。链表应存放在内部 RAM 中，供 GDMA 引擎使用。描述符每一字段的意义如下：

- owner (DW0) [31]: 表示当前描述符对应的 buffer 允许的操作者。
1'b0: 允许的操作者为 CPU;

1'b1: 允许的操作者为 GDMA 控制器。

在 GDMA 使用完该描述符对应的 buffer 后，对于接收描述符，硬件默认会自动将该 bit 清零；对于发送描述符，需要将 GDMA_OUT_AUTO_WRBK_CH_n 置 1，硬件才会自动将该 bit 清零。软件也可通过置位 GDMA_OUT_LOOP_TEST_CH_n 或 GDMA_IN_LOOP_TEST_CH_n 来关闭硬件自动清零的功能。软件在挂载链表时需要将该 bit 置 1。

注意：本文以 GDMA_OUT 开头的寄存器对应 TX 通道寄存器，以 GDMA_IN 开头的寄存器对应 RX 通道寄存器。

- suc_eof (DW0) [30]: 表示结束标志。
 - 1'b0: 当前描述符不是链表中的最后一个描述符；
 - 1'b1: 当前描述符为数据包的最后一个描述符。对于接收描述符，需要软件将该 bit 写 0，硬件会在收完 1 个包后将该 bit 置 1。对于发送描述符，需要软件在包的最后一个描述符中的该 bit 置 1。
- Reserved (DW0) [29]: 保留。此位为无关项。
- err_eof (DW0) [28]: 表示接收结束错误标志。
该 bit 只用于 UHCIO 利用 GDMA 接收数据。对于接收描述符，硬件在收完 1 个包并检测到接收数据错误会将该 bit 置 1。
- Reserved (DW0) [27:24]: 保留。
- length (DW0) [23:12]: 表示当前描述符对应的 buffer 中的有效字节数。对于发送描述符，该段由软件填写，表示从 buffer 中读取数据时需要读取的字节数；对于接收描述符，该段由硬件使用完该 buffer 后或者接收到最后一个数据时自动填写，表示 buffer 中存储的有效字节数。
- size (DW0) [11:0]: 表示当前描述符对应的 buffer 容量的字节数。
- buffer address pointer (DW1): buffer 的地址。
- next descriptor address (DW2): 下一个描述符的地址。如果当前描述符为链表中最后一个描述符时 (即 suc_eof = 1)，该值可以为 0。该地址必须指向片内 RAM 的地址空间。

用 GDMA 接收数据时，如果接收数据的长度小于当前描述符指定的 buffer 长度，那么下一个描述符对应的接收数据不会占用该 buffer 的剩余空间。

2.4.2 外设到存储及存储到外设的数据传输

GDMA 支持存储到外设及外设到存储的数据传输，分别对应 TX 及 RX 功能。TX 通道通过 outlink_n 实现将指定存储区域中的数据搬运到外设的发送端；RX 通道通过 inlink_n 实现将外设接收到的数据搬运到指定的存储区域。

每个 RX/TX 通道均可以被配置连接到任意一个支持 GDMA 功能的外设，表2-1 所示为配置寄存器与其对应外设的关系。当其中一个通道已经与某一个外设连接时，其他通道将不能配置为与该外设连接。

表 2-1. 配置寄存器与外设选择关系表

GDMA_IN_PERI_SEL_CH _n GDMA_OUT_PERI_SEL_CH _n	外设
0	SPI2
1	Reserved
2	UHCIO
3	I2S

4	Reserved
5	Reserved
6	AES
7	SHA
8	ADC

2.4.3 存储到存储数据传输

GDMA 支持存储到存储的数据传输。置位 `GDMA_MEM_TRANS_EN_CHn`，TX 通道_n的输出将与 RX 通道_n的输入相连，从而使能存储到存储的数据传输功能。需要注意的是，一个 TX 通道只与其编号对应的 RX 通道相连而实现存储到存储的数据传输。

2.4.4 启动 DMA

软件通过挂载链表的方式来使用 GDMA。对于接收数据，软件挂载好接收链表并准备好接收数据，配置 `GDMA_INLINK_ADDR_CHn` 字段指向第一个接收链表描述符。置位 `GDMA_INLINK_START_CHn` 位启动 GDMA。对于发送数据，软件挂载好发送链表并准备好发送数据，配置 `GDMA_OUTLINK_ADDR_CHn` 字段指向第一个发送链表描述符。置位 `GDMA_OUTLINK_START_CHn` 位启动 GDMA。`GDMA_INLINK_START_CHn` 与 `GDMA_OUTLINK_START_CHn` 位由硬件自动清零。

有时您可能想要在 DMA 数据传输已经开始后追加更多描述符。要挂载更多描述符，原本看似只需清空已挂载链表最后一个描述符的 EOF 位，并将该描述符的 next descriptor address (DW2) 字段配置为新链表第一个描述符。但如果 DMA 数据传输已经或马上就要结束，这个方法便行不通了。GDMA 引擎有专门的逻辑来确保数据传输继续或重启：如果数据传输仍在进行，GDMA 引擎会确保顾及到新追加的描述符；如果数据传输已经结束，GDMA 引擎会重启数据传输，传输新追加的描述符。这个逻辑由 Restart 功能实现。

软件使用 Restart 功能时，需要重写已挂载链表的最后一个描述符，使其第三个 word 中的内容（即 DW2）指向新链表的首地址；然后置位 `GDMA_INLINK_RESTART_CHn` 或者 `GDMA_OUTLINK_RESTART_CHn`（这两个位由硬件自动清零），如图 2-4 所示，硬件会在读取已挂载链表的最后一个描述符时，获取新挂载链表的地址，从而继续处理新挂载的链表。

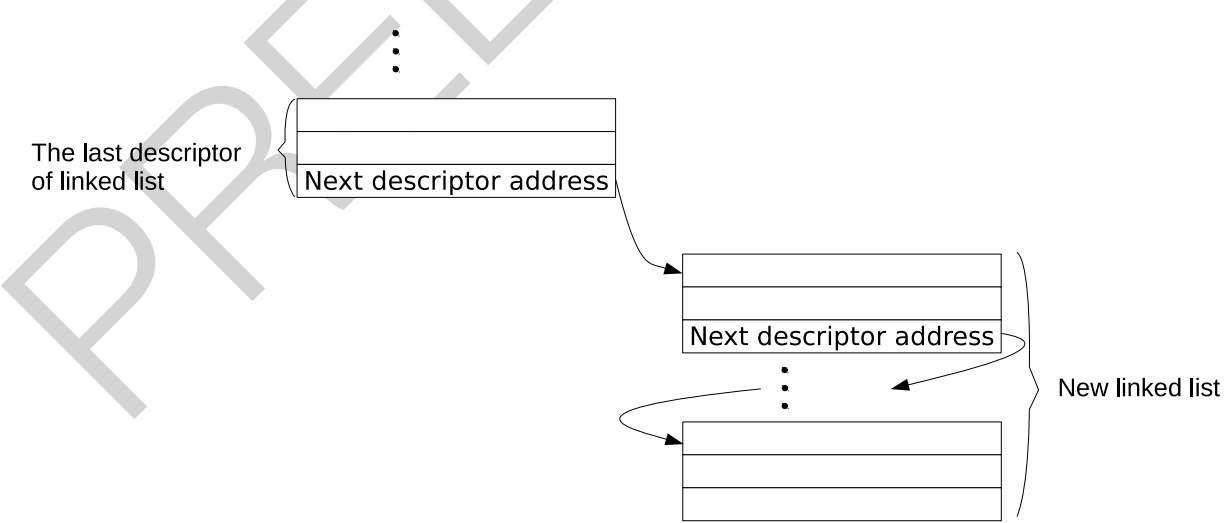


图 2-4. 链表关系图

2.4.5 读链表

软件在配置并启动 GDMA 后，GDMA 会从内部 RAM 读取链表。GDMA 会检查读入的链表描述符是否正确。只有当链表描述符通过检查时，GDMA 对应的通道才会开始搬运数据。当链表描述符没有通过检查，硬件将触发描述符错误中断 (GDMA_IN_DSCR_ERR_CH n _INT 或者 GDMA_OUT_DSCR_ERR_CH n _INT)，同时该通道将会处于阻塞状态，停止工作。

描述符检查项包括：

- GDMA_IN_CHECK_OWNER_CH n 或者 GDMA_OUT_CHECK_OWNER_CH n 置 1 时，检查描述符的 owner 比特。如果该比特为 0，表示当前操作者为 CPU，则不通过检查。GDMA_IN_CHECK_OWNER_CH n 或者 GDMA_OUT_CHECK_OWNER_CH n 置 0 时，不检查描述符的 owner 比特；
- 检查描述符中第二个 word 指示的地址是否在 0x3FC80000 ~ 0x3FCDFFFF 时（请参见本节 2.4.7）。如果在该范围内，则通过检查。否则，不通过检查。

软件在检查到通道描述符错误中断后，需要复位对应的通道，并置位 GDMA_OUTLINK_START_CH n 或者 GDMA_INLINK_START_CH n 位启动 GDMA。

注意：描述符的第三个 word 指示的地址只能在片内，指向下一个可用描述符；所有描述符都需存在内存中。

2.4.6 数据传输结束标志

GDMA 通过 EOF 来指示数据传输结束。

发送数据时，置位 GDMA_OUT_TOTAL_EOF_CH n _INT_ENA 位使能 GDMA_OUT_TOTAL_EOF_CH n _INT 中断，当带有 EOF 标志的描述符对应 buffer 的数据传输完成后，GDMA 会产生该中断。

接收数据时，置位 GDMA_IN_SUC_EOF_CH n _INT_ENA 位使能 GDMA_IN_SUC_EOF_CH n _INT 中断，表示数据接收完成。GDMA 还支持中断 GDMA_IN_ERR_EOF_CH n _INT，置位 GDMA_IN_ERR_EOF_CH n _INT_ENA 使能该中断，表示接收完成但接收数据有错误。需要注意的是，只有当通道连接的外设为 UHCI0 时，才支持该中断。

软件在检测到 GDMA_OUT_TOTAL_EOF_CH n _INT 或 GDMA_IN_SUC_EOF_CH n _INT 中断时，可以记录 GDMA_OUT_EOF_DES_ADDR_CH n 或 GDMA_IN_SUC_EOF_DES_ADDR_CH n 字段的值，即最后一个描述符的地址。这样，软件可以知道哪些描述符已经被使用并根据需要回收描述符。

注意：本章中提到发送链表描述符的 EOF 为 suc_eof，接收链表描述符的 EOF 可以为 suc_eof 和 err_eof。

2.4.7 访问片内 RAM

GDMA 任意 RX/TX 通道均可以访问片内 RAM，其可访问的片内地址空间为 0x3FC80000 ~ 0x3FCDFFFF。为加速数据传输速率，支持突发传输模式。置位 GDMA_IN_DATA_BURST_EN_CH n 使能 RX 通道突发传输模式；置位 GDMA_OUT_DATA_BURST_EN_CH n 使能 TX 通道突发传输模式。默认情况下，突发传输没有使能。

表 2-2. 链表描述符参数对齐要求

inlink/outlink	burst mode	size	length	buffer address pointer
inlink	0	无对齐要求	无对齐要求	无对齐要求
	1	字对齐	无对齐要求	字对齐

outlink	0	无对齐要求	无对齐要求	无对齐要求
	1	无对齐要求	无对齐要求	无对齐要求

如表2-2所示为访问片内时，链表描述符参数配置对齐要求。

当突发模式没有被使能时，无论是发送链表描述符还是接收链表描述符，其参数 size、length 及 buffer address pointer 均没有字对齐的要求。也就是说，对于一个描述符，在可访问的片内地址空间，GDMA 可以从任意起始地址，读出配置长度的数据，长度取值范围为 1 ~ 4095；或者，将接收到的数据长度（1 ~ 4095）写入任意起始地址开始的连续地址。

当突发模式使能时，对于发送链表描述符，参数 size、length 及 buffer address pointer 均没有字对齐的要求。而对于接收链表描述符，除了参数 length，参数 size 和 buffer address pointer 均需要保持字对齐。

2.4.8 仲裁

为了确保及时响应高速低延迟的外设请求，比如 SPI 等，GDMA 在通道仲裁机制中引入固定优先级，即每个通道的优先级可配置。GDMA 支持 10（0 ~ 9）个等级的优先级。其数值越大，对应的优先级越高，请求响应越及时。当若干个通道配置为相同的优先级时，这几个通道间对请求的响应将采用轮询仲裁机制。

需要注意的是，所有外设总的吞吐率之和不能超过 GDMA 能支持的最大有效带宽，从而保证低优先级的外设请求也能获得响应。

2.4.9 带宽

GDMA 作为 AHB 主机，通过发起 AHB 传输来访问存储空间。在不考虑其他 AHB 主机（WIFI）与 GDMA 竞争的情况下，GDMA 能支持的总带宽计算公式如下：

每个通道均使能突发传输模式： $8/5 * fhclk$ MB/s；

每个通道均不使能突发传输模式： $4/3 * fhclk$ MB/s；

其中 fhclk 为 AHB 时钟频率，固定为 80 MHz。根据上述计算公式，GDMA 的总带宽如表2-3所示：

表 2-3. GDMA 支持的总带宽

fpclk 突发传输模式	各通道均不使能	各通道均使能
80 MHz	106.6 MB/s	128 MB/s

需要注意的是，由于 GDMA 通过描述符来控制数据的传输，总的数据传输量包含读描述符自身的字节数。一个描述符对应的传输效率为： $length/(length + 12)$

其中 length 为描述符中参数，12 对应描述符的 3 个字。因此，在多链表描述符的应用中，应尽可能的提高单个描述符的 length 来提高传输效率，其最大值为 99.7%。

软件在分配带宽给外设时，可以通过公式： $T * (length + 12)/length$ 来预估该外设占用的 GDMA 带宽。其中 T 为外设的吞吐率。

2.5 GDMA 中断

- GDMA_OUT_TOTAL_EOF_CH n _INT：对于发送通道 n ，当一个链表（可包含多个链表描述符）对应的所有数据都已发送完成时触发此中断。

- GDMA_IN_DSCR_EMPTY_CH n _INT: 对于接收通道 n , 当接收链表描述符指向的 buffer 大小小于待接收数据长度时触发此中断。
- GDMA_OUT_DSCR_ERR_CH n _INT: 对于发送通道 n , 当发送链表描述符里有错误时触发此中断。
- GDMA_IN_DSCR_ERR_CH n _INT: 对于接收通道 n , 当接收链表描述符里有错误时触发此中断。
- GDMA_OUT_EOF_CH n _INT: 对于发送通道 n , 当发送描述符的 EOF 位为 1, 并且该描述符对应的数据发送完成时触发此中断。当GDMA_OUT_EOF_MODE_CH n 为 0 时, 该描述符对应的最后一个数据进入到 GDMA TX 通道时, 该中断触发; 当GDMA_OUT_EOF_MODE_CH n 为 1 时, 该描述符对应的最后一个数据从 GDMA TX 通道取出时, 该中断触发。
- GDMA_OUT_DONE_CH n _INT: 对于发送通道 n , 当一个发送链表描述符对应的数据发送完成时触发此中断。
- GDMA_IN_ERR_EOF_CH n _INT: 对于接收通道 n , 当接收的一个数据包中有错误发生时触发此中断。(该中断只用于外设选择 UHCIO (UART0/UART1) 时)。
- GDMA_IN_SUC_EOF_CH n _INT: 对于接收通道 n , 当一个数据包接收完成时触发此中断。
- GDMA_IN_DONE_CH n _INT: 对于接收通道 n , 当一个接收链表描述符对应的数据接收完成时触发此中断。

2.6 编程流程

2.6.1 GDMA TX 通道配置流程

利用 GDMA 发送数据时, GDMA TX 通道的软件配置流程如下:

1. 对寄存器 GDMA_OUT_RST_CH n 置 1 然后置 0, 复位 GDMA TX 通道状态机和 FIFO 指针;
2. 挂载好发送链表, 配置寄存器 GDMA_OUTLINK_ADDR_CH n 指向第一个发送链表描述符;
3. 配置 GDMA_PERI_OUT_SEL_CH n 为对应的外设号, 见表2-1;
4. 置位 GDMA_OUTLINK_START_CH n 启动 GDMA TX 通道发送数据;
5. 配置对应的外设 (SPI2、UHCIO (UART0/UART1)、I2S、AES、SHA、ADC), 并启动该外设, 具体配置请参考对应的外设章节;
6. 等待 GDMA_OUT_EOF_CH n _INT 中断, 即数据传输完成。

2.6.2 GDMA RX 通道配置流程

利用 GDMA 接收数据时, GDMA RX 通道的软件配置流程如下:

1. 对寄存器 GDMA_IN_RST_CH n 置 1 然后置 0, 复位 GDMA RX 通道状态机和 FIFO 指针;
2. 挂载好接收链表, 配置寄存器 GDMA_INLINK_ADDR_CH n 指向第一个接收链表描述符;
3. 配置 GDMA_PERI_IN_SEL_CH n 为对应的外设号, 见表2-1;
4. 置位 GDMA_INLINK_START_CH n 启动 GDMA RX 通道发送数据;
5. 配置对应的外设 (SPI2、UHCIO (UART0/UART1)、I2S、AES、ADC), 并启动该外设, 具体配置请参考对应的外设章节;
6. 等待 GDMA_IN_SUC_EOF_CH n _INT 中断, 即一个数据包接收完成。

2.6.3 GDMA 存储器到存储器配置流程

利用 GDMA 从存储到存储搬运数据时配置流程如下：

1. 对寄存器 `GDMA_OUT_RST_CHn` 置 1 然后置 0，复位 GDMA TX 通道状态机和 FIFO 指针；
2. 对寄存器 `GDMA_IN_RST_CHn` 置 1 然后置 0，复位 GDMA RX 通道状态机和 FIFO 指针；
3. 挂载好发送链表，配置寄存器 `GDMA_OUTLINK_ADDR_CHn` 指向第一个发送链表描述符；
4. 挂载好接收链表，配置寄存器 `GDMA_INLINK_ADDR_CHn` 指向第一个接收链表描述符；
5. 置位 `GDMA_MEM_TRANS_EN_CHn` 使能 memory-to-memory 传输功能；
6. 置位 `GDMA_OUTLINK_START_CHn` 启动 GDMA TX 通道发送数据；
7. 置位 `GDMA_INLINK_START_CHn` 启动 GDMA RX 通道发送数据；
8. 等待 `GDMA_IN_SUC_EOF_CHn_INT` 中断，即一次数据搬运完成。

2.7 寄存器列表

本小节的所有地址均为相对于通用 DMA 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
中段寄存器			
GDMA_INT_RAW_CH0_REG	接收通道 0 的原始中断状态	0x0000	R/WTC/SS
GDMA_INT_ST_CH0_REG	接收通道 0 的屏蔽中断	0x0004	RO
GDMA_INT_ENA_CH0_REG	接收通道 0 的中断使能位	0x0008	R/W
GDMA_INT_CLR_CH0_REG	接收通道 0 的中断清除位	0x000C	WT
GDMA_INT_RAW_CH1_REG	接收通道 1 的原始中断状态	0x0010	R/WTC/SS
GDMA_INT_ST_CH1_REG	接收通道 1 的屏蔽中断	0x0014	RO
GDMA_INT_ENA_CH1_REG	接收通道 1 的中断使能位	0x0018	R/W
GDMA_INT_CLR_CH1_REG	接收通道 1 的中断清除位	0x001C	WT
GDMA_INT_RAW_CH2_REG	接收通道 2 的原始中断状态	0x0020	R/WTC/SS
GDMA_INT_ST_CH2_REG	接收通道 2 的屏蔽中断	0x0024	RO
GDMA_INT_ENA_CH2_REG	接收通道 2 的中断使能位	0x0028	R/W
GDMA_INT_CLR_CH2_REG	接收通道 2 的中断清除位	0x002C	WT
配置寄存器			
GDMA_MISC_CONF_REG	MISC 寄存器	0x0044	R/W
版本寄存器			
GDMA_DATE_REG	版本控制寄存器	0x0048	R/W
配置寄存器			
GDMA_IN_CONF0_CH0_REG	接收通道 0 的配置寄存器 0	0x0070	R/W
GDMA_IN_CONF1_CH0_REG	接收通道 0 的配置寄存器 1	0x0074	R/W
GDMA_IN_POP_CH0_REG	接收通道 0 的数据弹出控制寄存器	0x007C	varies
GDMA_IN_LINK_CH0_REG	接收通道 0 的链表配置和控制寄存器	0x0080	varies
GDMA_OUT_CONF0_CH0_REG	发送通道 0 的配置寄存器 0	0x00D0	R/W
GDMA_OUT_CONF1_CH0_REG	发送通道 0 的配置寄存器 1	0x00D4	R/W
GDMA_OUT_PUSH_CH0_REG	发送通道 0 的数据推送控制寄存器	0x00DC	varies
GDMA_OUT_LINK_CH0_REG	发送通道 0 的链表配置和控制寄存器	0x00E0	varies
GDMA_IN_CONF0_CH1_REG	接收通道 1 的配置寄存器 0	0x0130	R/W
GDMA_IN_CONF1_CH1_REG	接收通道 1 的配置寄存器 1	0x0134	R/W
GDMA_IN_POP_CH1_REG	接收通道 1 的数据弹出控制寄存器	0x013C	varies
GDMA_IN_LINK_CH1_REG	接收通道 1 的链表配置和控制寄存器	0x0140	varies
GDMA_OUT_CONF0_CH1_REG	发送通道 1 的配置寄存器 0	0x0190	R/W
GDMA_OUT_CONF1_CH1_REG	发送通道 1 的配置寄存器 1	0x0194	R/W
GDMA_OUT_PUSH_CH1_REG	发送通道 1 的数据推送控制寄存器	0x019C	varies
GDMA_OUT_LINK_CH1_REG	发送通道 1 的链表配置和控制寄存器	0x01A0	varies
GDMA_IN_CONF0_CH2_REG	接收通道 2 的配置寄存器 0	0x01F0	R/W
GDMA_IN_CONF1_CH2_REG	接收通道 2 的配置寄存器 1	0x01F4	R/W

名称	描述	地址	访问
GDMA_IN_POP_CH2_REG	接收通道 2 的数据弹出控制寄存器	0x01FC	varies
GDMA_IN_LINK_CH2_REG	接收通道 2 的链表配置和控制寄存器	0x0200	varies
GDMA_OUT_CONF0_CH2_REG	发送通道 2 的配置寄存器 0	0x0250	R/W
GDMA_OUT_CONF1_CH2_REG	发送通道 2 的配置寄存器 1	0x0254	R/W
GDMA_OUT_PUSH_CH2_REG	发送通道 2 的数据推送控制寄存器	0x025C	varies
GDMA_OUT_LINK_CH2_REG	发送通道 2 的链表配置和控制寄存器	0x0260	varies
状态寄存器			
GDMA_INFIFO_STATUS_CH0_REG	接收通道 0 的 RX FIFO 状态	0x0078	RO
GDMA_IN_STATE_CH0_REG	接收通道 0 的接收状态	0x0084	RO
GDMA_IN_SUC_EOF_DES_ADDR_CH0_REG	接收通道 0 传输完成时的接收链表描述符地址	0x0088	RO
GDMA_IN_ERR_EOF_DES_ADDR_CH0_REG	接收通道 0 发生错误时的接收链表描述符地址	0x008C	RO
GDMA_IN_DSCR_CH0_REG	接收通道 0 当前的接收链表描述符地址	0x0090	RO
GDMA_IN_DSCR_BF0_CH0_REG	接收通道 0 最后一个接收链表描述符地址	0x0094	RO
GDMA_IN_DSCR_BF1_CH0_REG	接收通道 0 倒数第二个接收链表描述符地址	0x0098	RO
GDMA_OUTFIFO_STATUS_CH0_REG	发送通道 0 的 TX FIFO 状态	0x00D8	RO
GDMA_OUT_STATE_CH0_REG	发送通道 0 的发送状态	0x00E4	RO
GDMA_OUT_EOF_DES_ADDR_CH0_REG	发送通道 0 传输完成时的发送链表描述符地址	0x00E8	RO
GDMA_OUT_EOF_BFR_DES_ADDR_CH0_REG	发送通道 0 传输完成时的最后一个发送链表描述符地址	0x00EC	RO
GDMA_OUT_DSCR_CH0_REG	发送通道 0 当前的发送链表描述符地址	0x00F0	RO
GDMA_OUT_DSCR_BF0_CH0_REG	发送通道 0 最后一个发送链表描述符地址	0x00F4	RO
GDMA_OUT_DSCR_BF1_CH0_REG	发送通道 0 倒数第二个发送链表描述符地址	0x00F8	RO
GDMA_INFIFO_STATUS_CH1_REG	接收通道 1 的 RX FIFO 状态	0x0138	RO
GDMA_IN_STATE_CH1_REG	接收通道 1 的接收状态	0x0144	RO
GDMA_IN_SUC_EOF_DES_ADDR_CH1_REG	接收通道 1 传输完成时的接收链表描述符地址	0x0148	RO
GDMA_IN_ERR_EOF_DES_ADDR_CH1_REG	接收通道 1 发生错误时的接收链表描述符地址	0x014C	RO
GDMA_IN_DSCR_CH1_REG	接收通道 1 当前的接收链表描述符地址	0x0150	RO
GDMA_IN_DSCR_BF0_CH1_REG	接收通道 1 最后一个接收链表描述符地址	0x0154	RO
GDMA_IN_DSCR_BF1_CH1_REG	接收通道 1 倒数第二个接收链表描述符地址	0x0158	RO
GDMA_OUTFIFO_STATUS_CH1_REG	发送通道 1 的 TX FIFO 状态	0x0198	RO
GDMA_OUT_STATE_CH1_REG	发送通道 1 的发送状态	0x01A4	RO
GDMA_OUT_EOF_DES_ADDR_CH1_REG	发送通道 1 传输完成时的发送链表描述符地址	0x01A8	RO
GDMA_OUT_EOF_BFR_DES_ADDR_CH1_REG	发送通道 1 传输完成时的最后一个发送链表描述符地址	0x01AC	RO
GDMA_OUT_DSCR_CH1_REG	发送通道 1 当前的发送链表描述符地址	0x01B0	RO

名称	描述	地址	访问
GDMA_OUT_DSCR_BF0_CH1_REG	发送通道 1 最后一个发送链表描述符地址	0x01B4	RO
GDMA_OUT_DSCR_BF1_CH1_REG	发送通道 1 倒数第二个发送链表描述符地址	0x01B8	RO
GDMA_INFIFO_STATUS_CH2_REG	接收通道 2 的 RX FIFO 状态	0x01F8	RO
GDMA_IN_STATE_CH2_REG	接收通道 2 的接收状态	0x0204	RO
GDMA_IN_SUC_EOF_DES_ADDR_CH2_REG	接收通道 2 传输完成时的接收链表描述符地址	0x0208	RO
GDMA_IN_ERR_EOF_DES_ADDR_CH2_REG	接收通道 2 发生错误时的接收链表描述符地址	0x020C	RO
GDMA_IN_DSCR_CH2_REG	接收通道 2 当前的接收链表描述符地址	0x0210	RO
GDMA_IN_DSCR_BF0_CH2_REG	接收通道 2 最后一个接收链表描述符地址	0x0214	RO
GDMA_IN_DSCR_BF1_CH2_REG	接收通道 2 倒数第二个接收链表描述符地址	0x0218	RO
GDMA_OUTFIFO_STATUS_CH2_REG	发送通道 2 的 TX FIFO 状态	0x0258	RO
GDMA_OUT_STATE_CH2_REG	发送通道 2 的发送状态	0x0264	RO
GDMA_OUT_EOF_DES_ADDR_CH2_REG	发送通道 2 传输完成时的发送链表描述符地址	0x0268	RO
GDMA_OUT_EOF_BFR_DES_ADDR_CH2_REG	发送通道 2 传输完成时的最后一个发送链表描述符地址	0x026C	RO
GDMA_OUT_DSCR_CH2_REG	发送通道 2 当前的发送链表描述符地址	0x0270	RO
GDMA_OUT_DSCR_BF0_CH2_REG	发送通道 2 最后一个发送链表描述符地址	0x0274	RO
GDMA_OUT_DSCR_BF1_CH2_REG	发送通道 2 倒数第二个发送链表描述符地址	0x0278	RO
优先级寄存器			
GDMA_IN_PRI_CH0_REG	接收通道 0 的优先级寄存器	0x009C	R/W
GDMA_OUT_PRI_CH0_REG	发送通道 0 的优先级寄存器	0x00FC	R/W
GDMA_IN_PRI_CH1_REG	接收通道 1 的优先级寄存器	0x015C	R/W
GDMA_OUT_PRI_CH1_REG	发送通道 1 的优先级寄存器	0x01BC	R/W
GDMA_IN_PRI_CH2_REG	接收通道 2 的优先级寄存器	0x021C	R/W
GDMA_OUT_PRI_CH2_REG	发送通道 2 的优先级寄存器	0x027C	R/W
外设选择寄存器			
GDMA_IN_PERI_SEL_CH0_REG	接收通道 0 的外设选择	0x00A0	R/W
GDMA_OUT_PERI_SEL_CH0_REG	发送通道 0 的外设选择	0x0100	R/W
GDMA_IN_PERI_SEL_CH1_REG	接收通道 1 的外设选择	0x0160	R/W
GDMA_OUT_PERI_SEL_CH1_REG	发送通道 1 的外设选择	0x01C0	R/W
GDMA_IN_PERI_SEL_CH2_REG	接收通道 2 的外设选择	0x0220	R/W
GDMA_OUT_PERI_SEL_CH2_REG	发送通道 2 的外设选择	0x0280	R/W

2.8 寄存器

本小节的所有地址均为相对于通用 DMA 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 2.1. GDMA_INT_RAW_CH n _REG (n : 0-2) (0x0000+16* n)

(reserved)																				GDMA_OUT_FIFO_UDF_CH0_INT_RAW GDMA_OUT_FIFO_OVF_CH0_INT_RAW GDMA_IN_FIFO_UDF_CH0_INT_RAW GDMA_IN_FIFO_OVF_CH0_INT_RAW GDMA_OUT_TOTAL_EOF_CH0_INT_RAW GDMA_IN_DSCR_EMPTY_CH0_INT_RAW GDMA_OUT_DSCR_ERR_CH0_INT_RAW GDMA_IN_DSCR_ERR_CH0_INT_RAW GDMA_IN_DONE_CH0_INT_RAW GDMA_IN_SUC_EOF_CH0_INT_RAW GDMA_IN_DONE_CH0_INT_RAW											
31													13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

GDMA_IN_DONE_CH n _INT_RAW 接收通道 0 接收到接收链表描述符指向的最后一个字节数据时，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_IN_SUC_EOF_CH n _INT_RAW 接收通道 0 接收到接收链表描述符指向的最后一个字节数据时，该原始中断位翻转至高电平。对于 UHCIO，接收通道 0 接收到接收链表描述符指向的最后一个字节数据且没有检测到数据错误时，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_IN_ERR_EOF_CH n _INT_RAW 仅在接收通道 0 连接 UHCIO，且检测到数据错误时，该原始中断位翻转至高电平。如通道连接其他外设，该原始中断保留。(R/WTC/SS)

GDMA_OUT_DONE_CH n _INT_RAW 发送通道 0 将发送链表描述符指向的最后一个字节数据发送给外设时，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_OUT_EOF_CH n _INT_RAW 发送通道 0 从存储器读取了发送链表描述符指向的最后一个字节数据时，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_IN_DSCR_ERR_CH n _INT_RAW 接收通道 0 检测到接收链表描述符错误时，包括 owner 位错误、接收链表描述符的第二个字和第三个字错误，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_OUT_DSCR_ERR_CH n _INT_RAW 发送通道 0 检测到发送链表描述符错误时，包括 owner 位错误、发送链表描述符的第二个字和第三个字错误，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_IN_DSCR_EMPTY_CH n _INT_RAW 接收通道 0 接收链表指向的 RX FIFO 已满，数据接收未完成，但没有更多接收链表时，该原始中断位翻转至高电平。(R/WTC/SS)

GDMA_OUT_TOTAL_EOF_CH n _INT_RAW 发送通道 0 发送了发送链表（包括一个或几个发送链表描述符）对应的数据时，该原始中断位翻转至高电平。(R/WTC/SS)

见下页...

Register 2.1. GDMA_INT_RAW_CH n _REG (n : 0-2) (0x0000+16* n)

接上页...

GDMA_INFIFO_OVF_CH n _INT_RAW 接收通道 0 的 L1 FIFO 上溢时，该原始中断位翻转至高电平。
(R/WTC/SS)

GDMA_INFIFO_UDF_CH n _INT_RAW 接收通道 0 的 L1 FIFO 下溢时，该原始中断位翻转至高电平。
(R/WTC/SS)

GDMA_OUTFIFO_OVF_CH n _INT_RAW 发送通道 0 的 L1 FIFO 上溢时，该原始中断位翻转至高电平。
(R/WTC/SS)

GDMA_OUTFIFO_UDF_CH n _INT_RAW 发送通道 0 的 L1 FIFO 下溢时，该原始中断位翻转至高电平。
(R/WTC/SS)

Register 2.2. GDMA_INT_ST_CH n _REG (n : 0-2) (0x0004+16* n)

(reserved)																								GDMA_OUTFIFO_UDF_CH0_INT_ST GDMA_OUTFIFO_OVF_CH0_INT_ST GDMA_INFIFO_UDF_CH0_INT_ST GDMA_INFIFO_OVF_CH0_INT_ST GDMA_OUT_TOTAL_EOF_CH0_INT_ST GDMA_IN_DSCR_EMPTY_CH0_INT_ST GDMA_OUT_DSCR_ERR_CH0_INT_ST GDMA_OUT_EOF_CH0_INT_ST GDMA_IN_DSCR_ERR_CH0_INT_ST GDMA_OUT_DONE_CH0_INT_ST GDMA_IN_SUC_EOF_CH0_INT_ST GDMA_IN_ERR_EOF_CH0_INT_ST GDMA_IN_DONE_CH0_INT_ST													
31													13											12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									

GDMA_IN_DONE_CH n _INT_ST GDMA_IN_DONE_CH_INT 中断的原始状态位。(RO)

GDMA_IN_SUC_EOF_CH n _INT_ST GDMA_IN_SUC_EOF_CH_INT 中断的原始状态位。(RO)

GDMA_IN_ERR_EOF_CH n _INT_ST GDMA_IN_ERR_EOF_CH_INT 中断的原始状态位。(RO)

GDMA_OUT_DONE_CH n _INT_ST GDMA_OUT_DONE_CH_INT 中断的原始状态位。(RO)

GDMA_OUT_EOF_CH n _INT_ST GDMA_OUT_EOF_CH_INT 中断的原始状态位。(RO)

GDMA_IN_DSCR_ERR_CH n _INT_ST GDMA_IN_DSCR_ERR_CH_INT 中断的原始状态位。(RO)

GDMA_OUT_DSCR_ERR_CH n _INT_ST GDMA_OUT_DSCR_ERR_CH_INT 中断的原始状态位。
(RO)

GDMA_IN_DSCR_EMPTY_CH n _INT_ST GDMA_IN_DSCR_EMPTY_CH_INT 中断的原始状态位。
(RO)

GDMA_OUT_TOTAL_EOF_CH n _INT_ST GDMA_OUT_TOTAL_EOF_CH_INT 中断的原始状态位。
(RO)

GDMA_INFIFO_OVF_CH n _INT_ST GDMA_INFIFO_OVF_L1_CH_INT 中断的原始状态位。(RO)

GDMA_INFIFO_UDF_CH n _INT_ST GDMA_INFIFO_UDF_L1_CH_INT 中断的原始状态位。(RO)

GDMA_OUTFIFO_OVF_CH n _INT_ST GDMA_OUTFIFO_OVF_L1_CH_INT 中断的原始状态位。(RO)

GDMA_OUTFIFO_UDF_CH n _INT_ST GDMA_OUTFIFO_UDF_L1_CH_INT 中断的原始状态位。(RO)

Register 2.3. GDMA_INT_ENA_CH_n_REG (*n*: 0-2) (0x0008+16**n*)

(reserved)																								GDMA_OUTFIFO_UDF_CH0_INT_ENA GDMA_OUTFIFO_OVF_CH0_INT_ENA GDMA_INFIFO_UDF_CH0_INT_ENA GDMA_INFIFO_OVF_CH0_INT_ENA GDMA_OUT_TOTAL_EOF_CH0_INT_ENA GDMA_IN_DSCR_EMPTY_CH0_INT_ENA GDMA_OUT_DSCR_ERR_CH0_INT_ENA GDMA_IN_DSCR_ERR_CH0_INT_ENA GDMA_OUT_DONE_CH0_INT_ENA GDMA_IN_ERR_EOF_CH0_INT_ENA GDMA_IN_SUC_EOF_CH0_INT_ENA GDMA_IN_DONE_CH0_INT_ENA													
31													13											12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										

- GDMA_IN_DONE_CH_n_INT_ENA GDMA_IN_DONE_CH_INT 中断的使能位。(R/W)
- GDMA_IN_SUC_EOF_CH_n_INT_ENA GDMA_IN_SUC_EOF_CH_INT 中断的使能位。(R/W)
- GDMA_IN_ERR_EOF_CH_n_INT_ENA GDMA_IN_ERR_EOF_CH_INT 中断的使能位。(R/W)
- GDMA_OUT_DONE_CH_n_INT_ENA GDMA_OUT_DONE_CH_INT 中断的使能位。(R/W)
- GDMA_OUT_EOF_CH_n_INT_ENA GDMA_OUT_DONE_CH_INT 中断的使能位。(R/W)
- GDMA_IN_DSCR_ERR_CH_n_INT_ENA GDMA_IN_DSCR_ERR_CH_INT 中断的使能位。(R/W)
- GDMA_OUT_DSCR_ERR_CH_n_INT_ENA GDMA_OUT_DSCR_ERR_CH_INT 中断的使能位。(R/W)
- GDMA_IN_DSCR_EMPTY_CH_n_INT_ENA GDMA_IN_DSCR_EMPTY_CH_INT 中断的使能位。(R/W)
- GDMA_OUT_TOTAL_EOF_CH_n_INT_ENA GDMA_OUT_TOTAL_EOF_CH_INT 中断的使能位。(R/W)
- GDMA_INFIFO_OVF_CH_n_INT_ENA GDMA_INFIFO_OVF_L1_CH_INT 中断的使能位。(R/W)
- GDMA_INFIFO_UDF_CH_n_INT_ENA GDMA_INFIFO_UDF_L1_CH_INT 中断的使能位。(R/W)
- GDMA_OUTFIFO_OVF_CH_n_INT_ENA GDMA_OUTFIFO_OVF_L1_CH_INT 中断的使能位。(R/W)
- GDMA_OUTFIFO_UDF_CH_n_INT_ENA GDMA_OUTFIFO_UDF_L1_CH_INT 中断的使能位。(R/W)

Register 2.4. GDMA_INT_CLR_CH_n_REG (*n*: 0-2) (0x000C+16**n*)

(reserved)																												GDMA_OUTFIFO_UDF_CH0_INT_CLR GDMA_OUTFIFO_OVF_CH0_INT_CLR GDMA_INFIFO_UDF_CH0_INT_CLR GDMA_INFIFO_OVF_CH0_INT_CLR GDMA_OUT_TOTAL_EOF_CH0_INT_CLR GDMA_IN_DSCR_ERR_CH0_INT_CLR GDMA_OUT_DSCR_EMPTY_CH0_INT_CLR GDMA_IN_DSCR_ERR_CH0_INT_CLR GDMA_OUT_DONE_CH0_INT_CLR GDMA_IN_ERR_EOF_CH0_INT_CLR GDMA_IN_SUC_EOF_CH0_INT_CLR GDMA_IN_DONE_CH0_INT_CLR													
31																13												12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0												0	0	0	0	0	0	0	0	0	0	0	0		

GDMA_IN_DONE_CH_n_INT_CLR 置位此位, 清除 GDMA_IN_DONE_CH_INT 中断。(WT)

GDMA_IN_SUC_EOF_CH_n_INT_CLR 置位此位, 清除 GDMA_IN_SUC_EOF_CH_INT 中断。(WT)

GDMA_IN_ERR_EOF_CH_n_INT_CLR 置位此位, 清除 GDMA_IN_ERR_EOF_CH_INT 中断。(WT)

GDMA_OUT_DONE_CH_n_INT_CLR 置位此位, 清除 GDMA_OUT_DONE_CH_INT 中断。(WT)

GDMA_OUT_EOF_CH_n_INT_CLR 置位此位, 清除 GDMA_OUT_EOF_CH_INT 中断。(WT)

GDMA_IN_DSCR_ERR_CH_n_INT_CLR 置位此位, 清除 GDMA_IN_DSCR_ERR_CH_INT 中断。(WT)

GDMA_OUT_DSCR_ERR_CH_n_INT_CLR 置位此位, 清除 GDMA_OUT_DSCR_ERR_CH_INT 中断。
(WT)

GDMA_IN_DSCR_EMPTY_CH_n_INT_CLR 置位此位, 清除 GDMA_IN_DSCR_EMPTY_CH_INT 中
断。(WT)

GDMA_OUT_TOTAL_EOF_CH_n_INT_CLR 置位此位, 清除 GDMA_OUT_TOTAL_EOF_CH_INT 中
断。(WT)

GDMA_INFIFO_OVF_CH_n_INT_CLR 置位此位, 清除 GDMA_INFIFO_OVF_L1_CH_INT 中断。(WT)

GDMA_INFIFO_UDF_CH_n_INT_CLR 置位此位, 清除 GDMA_INFIFO_UDF_L1_CH_INT 中断。(WT)

GDMA_OUTFIFO_OVF_CH_n_INT_CLR 置位此位, 清除 GDMA_OUTFIFO_OVF_L1_CH_INT 中断。
(WT)

GDMA_OUTFIFO_UDF_CH_n_INT_CLR 置位此位, 清除 GDMA_OUTFIFO_UDF_L1_CH_INT 中断。
(WT)

60

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

反馈文档意见

60

反馈文档意见

反馈文档意见

60

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

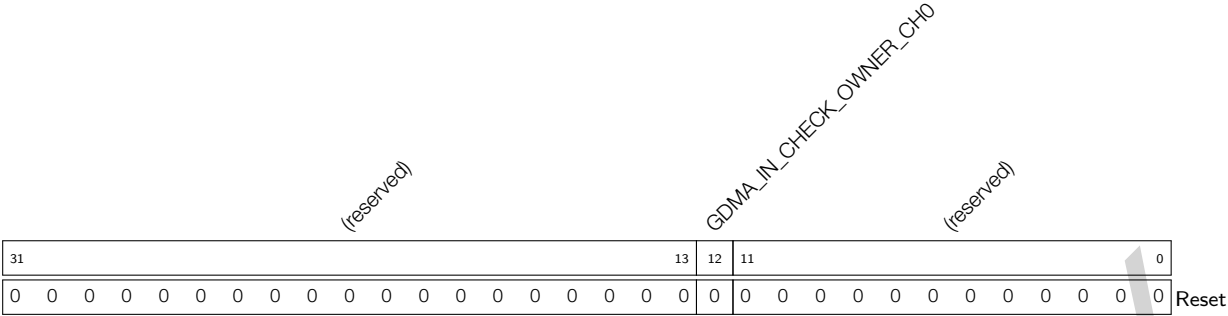
反馈文档意见

反馈文档意见

反馈文档意见

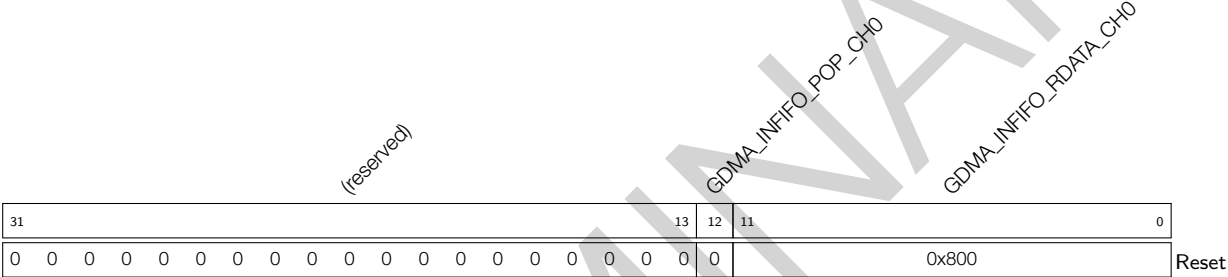
反馈文档意见

Register 2.8. GDMA_IN_CONF1_CH_n_REG (*n*: 0-2) (0x0074+192**n*)



GDMA_IN_CHECK_OWNER_CH_n 置位此位，使能链表描述符 owner 位检查。(R/W)

Register 2.9. GDMA_IN_POP_CH_n_REG (*n*: 0-2) (0x007C+192**n*)



GDMA_INFIFO_RDATA_CH_n 存储从 GDMA FIFO 中弹出的数据。(RO)

GDMA_INFIFO_POP_CH_n 置位此位，从 GDMA FIFO 中弹出数据。(R/W/SC)

Register 2.10. GDMA_IN_LINK_CH_n_REG (*n*: 0-2) (0x0080+192**n*)

(reserved)								GDMA_INLINK_PARK_CH0 GDMA_INLINK_RESTART_CH0 GDMA_INLINK_START_CH0 GDMA_INLINK_STOP_CH0 GDMA_INLINK_AUTO_RET_CH0																GDMA_INLINK_ADDR_CH0																																																																															
31								25								24	23	22	21	20	19								0																																																																										
0								0								0								0								0								0								1								0								0								0								1								0x000								Reset							

- GDMA_INLINK_ADDR_CH_n** 存储第一个接收链表描述符地址的低 20 位。(R/W)
- GDMA_INLINK_AUTO_RET_CH_n** 当前接收数据有错误时，置位此位返回当前接收链表描述符的地址。(R/W)
- GDMA_INLINK_STOP_CH_n** 置位此位，停止处理接收链表描述符。(R/W/SC)
- GDMA_INLINK_START_CH_n** 置位此位，开始处理接收链表描述符。(R/W/SC)
- GDMA_INLINK_RESTART_CH_n** 置位此位，挂载新的接收链表描述符。(R/W/SC)
- GDMA_INLINK_PARK_CH_n** 1：接收链表描述符的状态机空闲；0：接收链表描述符的状态机工作中。(RO)

63

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

63
反馈文档意见

63
反馈文档意见

63
反馈文档意见

63
反馈文档意见

63
反馈文档意见

63

63
反馈文档意见

Register 2.13. GDMA_OUT_PUSH_CH_{*n*}_REG (*n*: 0-2) (0x00DC+192n*)**

Diagram illustrating the structure of the **GDMA_OUTFIFO_PUSH_CH0** register. The register is 32 bits wide, divided into three fields:

- (reserved)**: 24 bits (bits 31 to 8).
- GDMA_OUTFIFO_PUSH_CH0**: 2 bits (bits 7 to 6).
- GDMA_OUTFIFO_WDATA_CH0**: 6 bits (bits 5 to 0).

The register value is shown as **0x0** for the lower 6 bits and **0** for the upper 26 bits. A **Reset** label points to the bottom of the register.

GDMA_OUTFIFO_WDATA_CH_n 存储需推送至 GDMA FIFO 的数据。(R/W)

GDMA_OUTFIFO_PUSH_CH_n 置位此位，将数据推送至 GDMA FIFO 中。(R/W/SC)

Register 2.14. GDMA_OUT_LINK_CH_{*n*}_REG (*n*: 0-2) (0x00E0+192n*)**

(reserved)																GDMA_OUTLINK_PARK_CH0																GDMA_OUTLINK_RESTART_CH0																GDMA_OUTLINK_START_CH0																GDMA_OUTLINK_STOP_CH0																GDMA_OUTLINK_ADDR_CH0															
31								24								23				22				21				20				19				0																																																											
0								0								0				0				0				0				1				0				0				0				0x000																Reset																															

GDMA_OUTLINK_ADDR_CH_n 存储第一个发送链表描述符地址的低 20 位。(R/W)

GDMA_OUTLINK_STOP_CH_n 置位此位，停止处理发送链表描述符。(R/W/SC)

GDMA_OUTLINK_START_CH_n 置位此位，开始处理发送链表描述符。(R/W/SC)

GDMA_OUTLINK_RESTART_CH_n 置位此位，在最后一个地址挂载新的发送链表。(R/W/SC)

GDMA_OUTLINK_PARK_CH_n 1: 发送链表描述符的状态机空闲; 0: 发送链表描述符的状态机工作中。(RO)

Register 2.15. GDMA_INFIFO_STATUS_CH_n_REG (n: 0-2) (0x0078+192*n)

(reserved)				GDMA_IN_BUF_HUNGRY_CH0				GDMA_IN_REMAIN_UNDER_4B_CH0				GDMA_IN_REMAIN_UNDER_3B_CH0				GDMA_IN_REMAIN_UNDER_2B_CH0				GDMA_IN_REMAIN_UNDER_1B_CH0				(reserved)								GDMA_INFIFO_CNT_CH0				GDMA_INFIFO_EMPTY_CH0				GDMA_INFIFO_FULL_CH0			
31					28	27	26	25	24	23	22									8	7					2	1	0					Reset										
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0						

GDMA_INFIFO_FULL_CH_n 接收通道 0 的 L1 RX FIFO 已满。(RO)

GDMA_INFIFO_EMPTY_CH_n 接收通道 0 的 L1 RX FIFO 为空。(RO)

GDMA_INFIFO_CNT_CH_n 接收通道 0 的 L1 RX FIFO 存储的字节数。(RO)

GDMA_IN_REMAIN_UNDER_1B_CH_n 保留。(RO)

GDMA_IN_REMAIN_UNDER_2B_CH_n 保留。(RO)

GDMA_IN_REMAIN_UNDER_3B_CH_n 保留。(RO)

GDMA_IN_REMAIN_UNDER_4B_CH_n 保留。(RO)

GDMA_IN_BUF_HUNGRY_CH_n 保留。(RO)

Register 2.16. GDMA_IN_STATE_CH_n_REG (n: 0-2) (0x0084+192*n)

(reserved)										GDMA_IN_STATE_CH0				GDMA_IN_DSCR_STATE_CH0																GDMA_INLINK_DSCR_ADDR_CH0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31									23	22	20	19	18	17																																			0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

GDMA_INLINK_DSCR_ADDR_CH_n 当前接收链表描述符的地址。(RO)

GDMA_IN_DSCR_STATE_CH_n 保留。(RO)

GDMA_IN_STATE_CH_n 保留。(RO)

Register 2.17. GDMA_IN_SUC_EOF_DES_ADDR_CH_{*n*}_REG (*n*: 0-2) (0x0088+192**n*)

GDMA_IN_SUC_EOF_DES_ADDR_CH0	
31	0
0x000000	
Reset	

GDMA_IN_SUC_EOF_DES_ADDR_CH_{*n*} 接收链表描述符的 EOF 为 1 时，该描述符的地址。(RO)

Register 2.18. GDMA_IN_ERR_EOF_DES_ADDR_CH_{*n*}_REG (*n*: 0-2) (0x008C+192**n*)

GDMA_IN_ERR_EOF_DES_ADDR_CH0	
31	0
0x000000	
Reset	

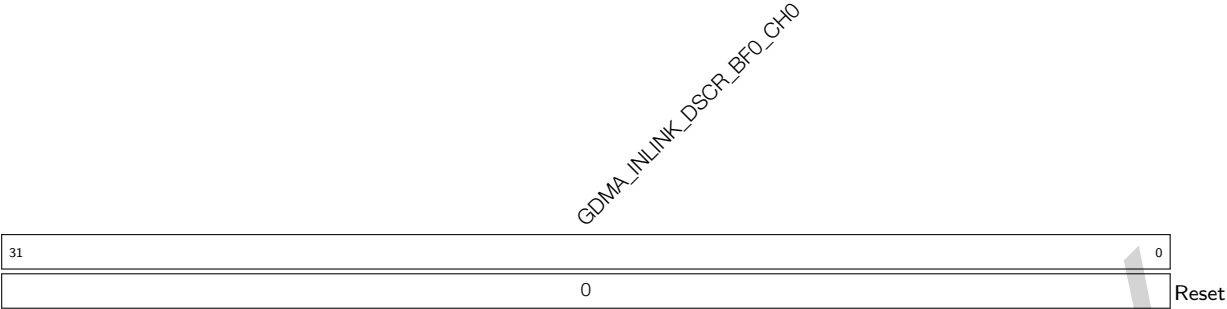
GDMA_IN_ERR_EOF_DES_ADDR_CH_{*n*} 当前接收数据有错误时，接收链表描述符的地址。仅在连接外设为 UHCI 0 时使用。(RO)

Register 2.19. GDMA_IN_DSCR_CH_{*n*}_REG (*n*: 0-2) (0x0090+192**n*)

GDMA_INLINK_DSCR_CH0	
31	0
0	
Reset	

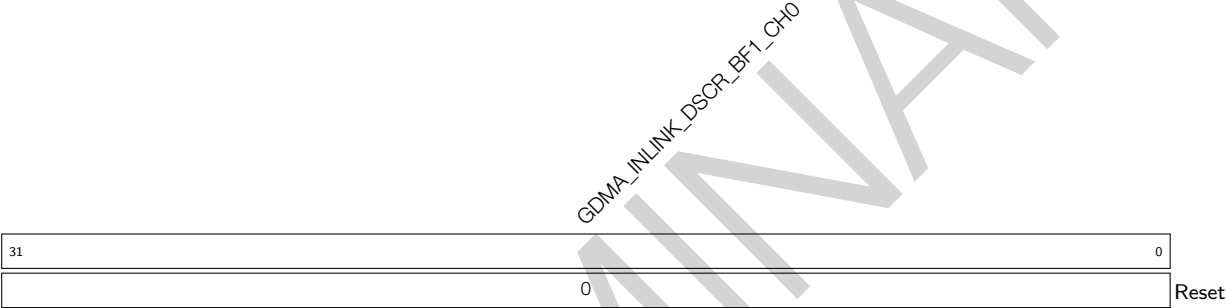
GDMA_INLINK_DSCR_CH_{*n*} 当前接收链表描述符 *x* 的地址。(RO)

Register 2.20. GDMA_IN_DSCR_BF0_CH_{*n*}_REG (*n*: 0-2) (0x0094+192**n*)



GDMA_INLINK_DSCR_BF0_CH_{*n*} 最后一个接收链表描述符 x-1 的地址。(RO)

Register 2.21. GDMA_IN_DSCR_BF1_CH_{*n*}_REG (*n*: 0-2) (0x0098+192**n*)



GDMA_INLINK_DSCR_BF1_CH_{*n*} 倒数第二个接收链表描述符 x-2 的地址。(RO)

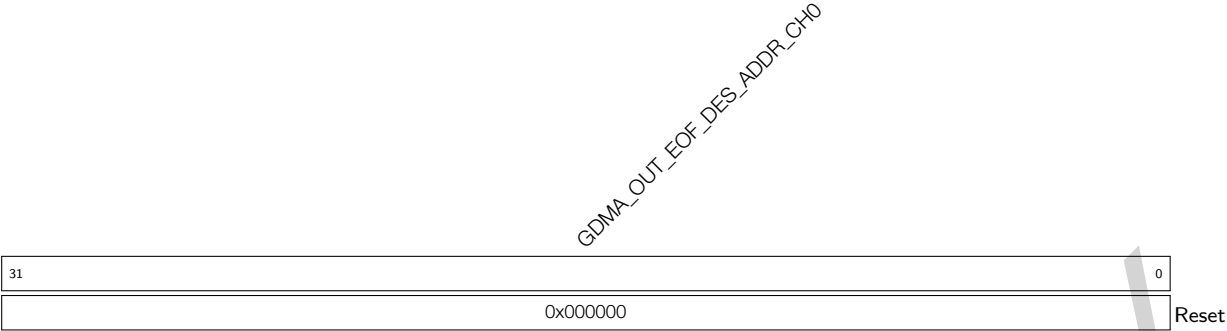
68

反馈文档意见

GDMA_OUT_STATE_CH0
GDMA_OUT_DSCR_STATE_CH0
GDMA_OUTLINK_DSCR_ADDR_CH0

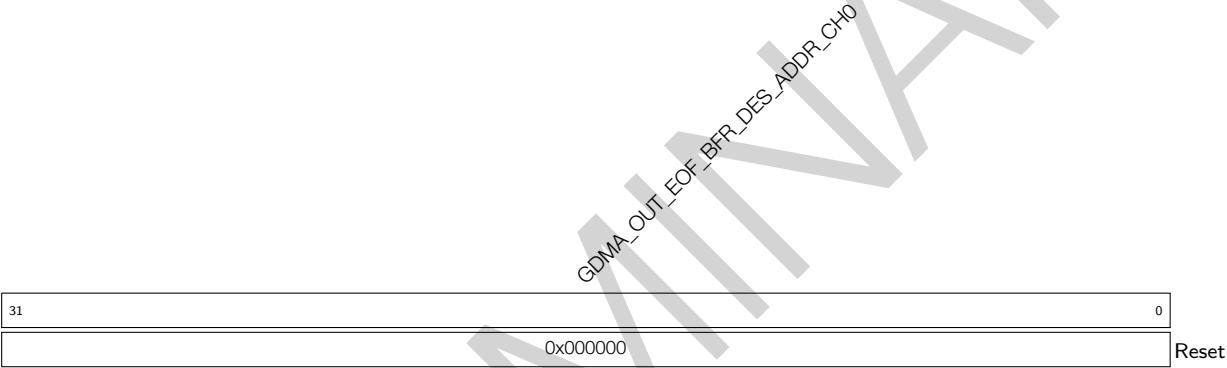
ESP32-C3 TRM (预发布 v0.4)

Register 2.24. GDMA_OUT_EOF_DES_ADDR_CH_{*n*}_REG (*n*: 0-2) (0x00E8+192**n*)



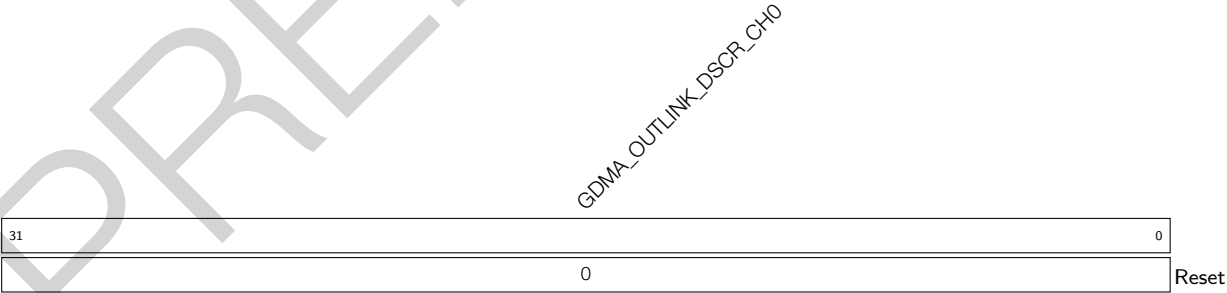
GDMA_OUT_EOF_DES_ADDR_CH_{*n*} 发送链表描述符的 EOF 为 1 时，该描述符的地址。(RO)

Register 2.25. GDMA_OUT_EOF_BFR_DES_ADDR_CH_{*n*}_REG (*n*: 0-2) (0x00EC+192**n*)



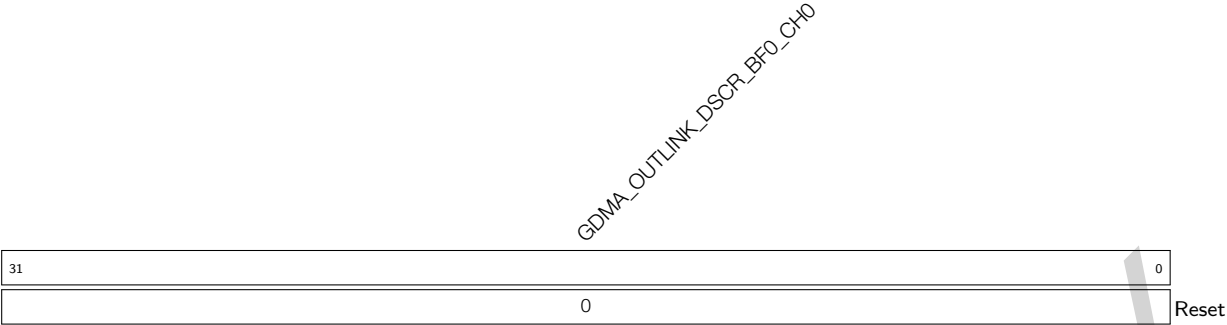
GDMA_OUT_EOF_BFR_DES_ADDR_CH_{*n*} 倒数第二个发送链表描述符的地址。(RO)

Register 2.26. GDMA_OUT_DSCR_CH_{*n*}_REG (*n*: 0-2) (0x00F0+192**n*)



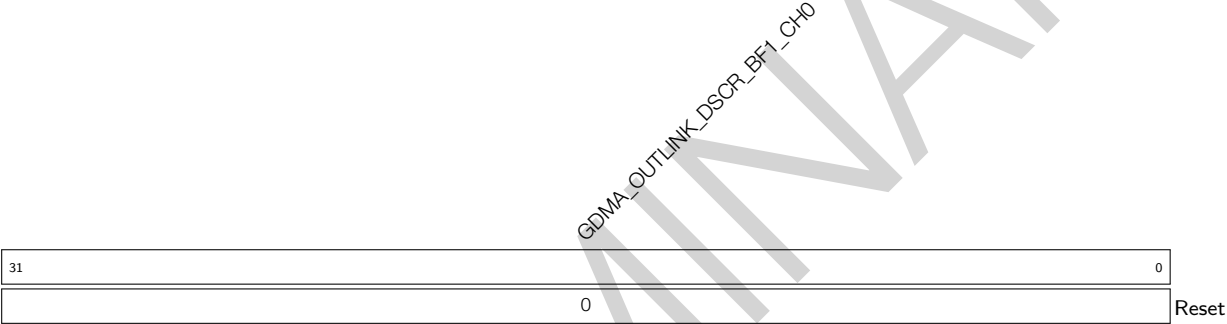
GDMA_OUTLINK_DSCR_CH_{*n*} 当前链表描述符 y 的地址。(RO)

Register 2.27. GDMA_OUT_DSCR_BF0_CH_{*n*}_REG (*n*: 0-2) (0x00F4+192**n*)



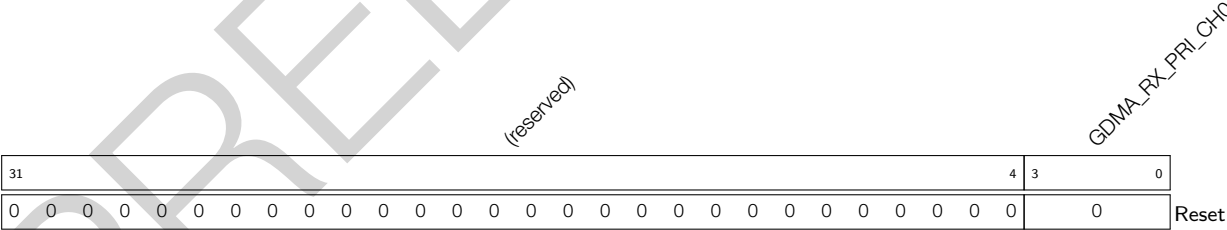
GDMA_OUTLINK_DSCR_BF0_CH_{*n*} 最后一个发送链表描述符 y-1 的地址。(RO)

Register 2.28. GDMA_OUT_DSCR_BF1_CH_{*n*}_REG (*n*: 0-2) (0x00F8+192**n*)



GDMA_OUTLINK_DSCR_BF1_CH_{*n*} 倒数第二个接收链表描述符 x-2 的地址。(RO)

Register 2.29. GDMA_IN_PRI_CH_{*n*}_REG (*n*: 0-2) (0x009C+192**n*)



GDMA_RX_PRI_CH_{*n*} 接收通道 0 的优先级。该值越大，优先级越高。(R/W)

Register 2.30. GDMA_OUT_PRI_CH_{*n*}_REG (*n*: 0-2) (0x00FC+192**n*)

(reserved)																								GDMA_TX_PRI_CH0			
31																								4	3	0	
0 0																								0		Reset	

GDMA_TX_PRI_CH_{*n*} 发送通道 0 的优先级。该值越大，优先级越高。(R/W)

Register 2.31. GDMA_IN_PERI_SEL_CH_{*n*}_REG (*n*: 0-2) (0x00A0+192**n*)

(reserved)																								GDMA_PERI_IN_SEL_CH0					
31																								6		5		0	
0 0																								0x3f		Reset			

GDMA_PERI_IN_SEL_CH_{*n*} 用于选择接收通道 0 连接的外设。0: SPI2; 1: 保留; 2: UHCI0; 3: I2S; 4: 保留; 5: 保留; 6: AES; 7: SHA; 8: ADC。(R/W)

Register 2.32. GDMA_OUT_PERI_SEL_CH_{*n*}_REG (*n*: 0-2) (0x0100+192**n*)

(reserved)																								GDMA_PERI_OUT_SEL_CH0					
31																								6		5		0	
0 0																								0x3f		Reset			

GDMA_PERI_OUT_SEL_CH_{*n*} 用于选择发送通道 0 连接的外设。0: SPI2; 1: 保留; 2: UHCI0; 3: I2S; 4: 保留; 5: 保留; 6: AES; 7: SHA; 8: ADC。(R/W)

3 系统和存储器

3.1 概述

ESP32-C3 是一个超低功耗和高度集成的系统，它集成了一颗 RISC-V 32 位单核处理器，四级流水线架构，主频高达 160 MHz。所有的内部存储器、外部存储器以及外设都分布在 CPU 的总线上。

3.2 主要特性

- **地址空间**
 - 792 KB 内部存储器指令地址空间
 - 552 KB 内部存储器数据地址空间
 - 836 KB 外设地址空间
 - 8 MB 外部存储器指令虚地址空间
 - 8 MB 外部存储器数据虚地址空间
 - 384 KB 内部 DMA 地址空间
- **内部存储器**
 - 384 KB 内部 ROM
 - 400 KB 内部 SRAM
 - 8 KB RTC 存储器
- **外部存储器**
 - 最大支持 16 MB 片外 flash
- **外设空间**
 - 总计 35 个模块/外设
- **GDMA**
 - 7 个具有 GDMA 功能的模块/外设

图 3-1 描述了系统结构与地址映射结构。

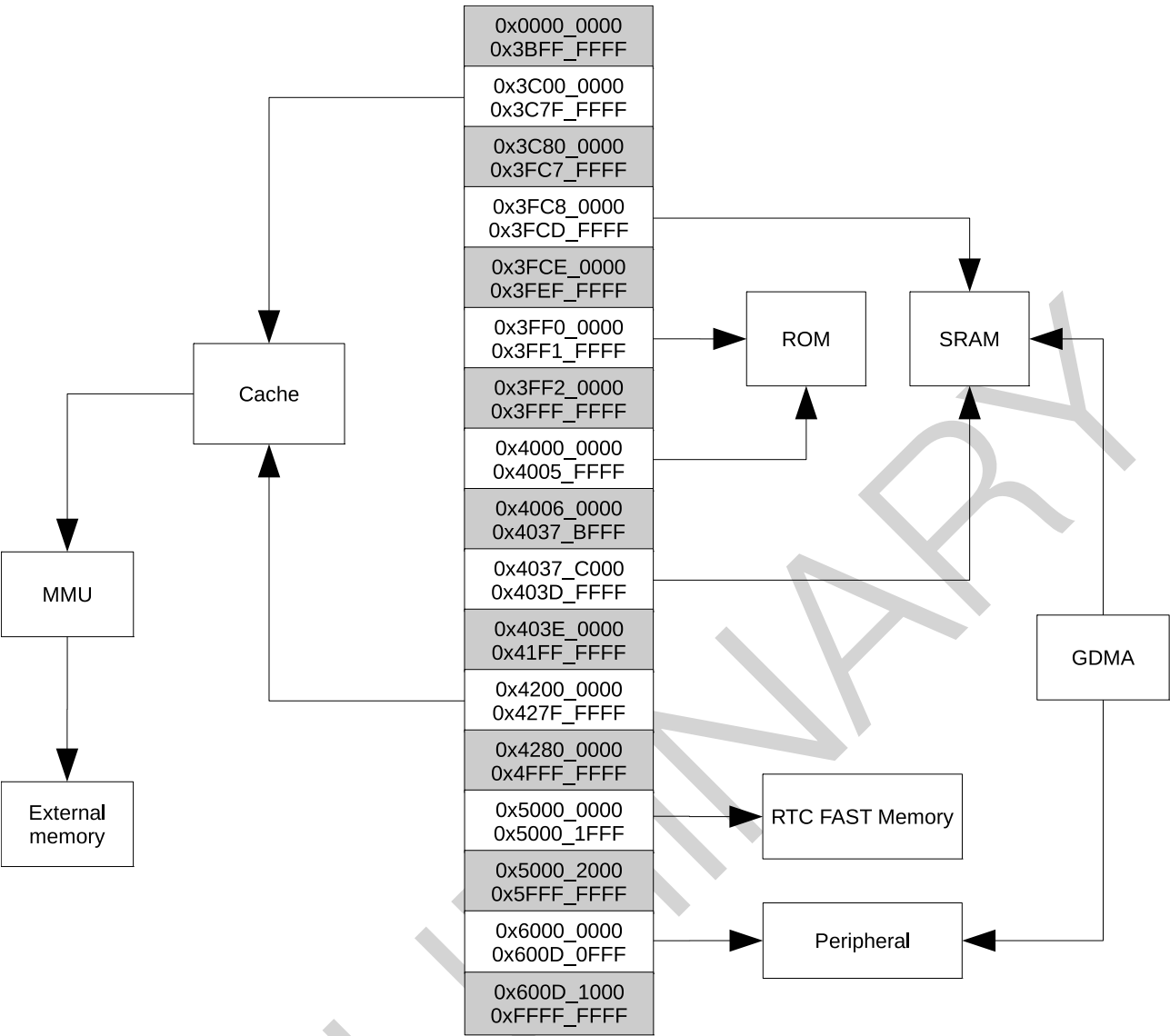


图 3-1. 系统结构与地址映射结构

说明:

- 图中灰色背景标注的地址空间不可用。
- 地址空间中可用的地址范围可能大于实际可用的内存。

3.3 功能描述

3.3.1 地址映射

地址 0x4000_0000 以下的部分属于数据总线的地址范围，地址 0x4000_0000 ~ 0x4FFF_FFFF 部分为指令总线的地址范围，地址 0x5000_0000 及以上的部分是数据总线与指令总线共用的地址范围。

CPU 的数据总线与指令总线都为小端序。CPU 可以通过数据总线进行单字节、双字节、4 字节的数据访问。CPU 也可以通过指令总线进行数据访问，但只能是 4 字节对齐的访问。

CPU 能够：

- 通过数据总线与指令总线直接访问内部存储器；
- 通过 Cache 访问映射到虚地址空间的外部存储器；
- 通过数据总线直接访问模块/外设。

表 3-1 描述了数据总线与指令总线中的各段地址所能访问的目标。

系统中部分内部存储器与部分外部存储器既可以被数据总线访问也可以被指令总线访问，这种情况下，CPU 可以通过多个地址访问到同一目标。

表 3-1. 地址映射

总线类型	边界地址		容量	目标
	低位地址	高位地址		
	0x0000_0000	0x3BFF_FFFF		保留
数据	0x3C00_0000	0x3C7F_FFFF	8 MB	外部存储器
	0x3C80_0000	0x3FC7_FFFF		保留
数据	0x3FC8_0000	0x3FCD_FFFF	384 KB	内部存储器
	0x3FCE_0000	0x3FEF_FFFF		保留
数据	0x3FF0_0000	0x3FF1_FFFF	128 KB	内部存储器
	0x3FF2_0000	0x3FFF_FFFF		保留
指令	0x4000_0000	0x4005_FFFF	384 KB	内部存储器
	0x4006_0000	0x4037_BFFF		保留
指令	0x4037_C000	0x403D_FFFF	400 KB	内部存储器
	0x403E_0000	0x41FF_FFFF		保留
指令	0x4200_0000	0x427F_FFFF	8 MB	外部存储器
	0x4280_0000	0x4FFF_FFFF		保留
数据/指令	0x5000_0000	0x5000_1FFF	8 KB	内部存储器
	0x5000_2000	0x5FFF_FFFF		保留
数据/指令	0x6000_0000	0x600D_0FFF	836 KB	外设
	0x600D_1000	0xFFFF_FFFF		保留

3.3.2 内部存储器

ESP32-C3 的内部存储器包含如下三种类型：

- 内部 ROM (384 KB)：内部 ROM 是只读存储器，不可编程。其中存放有一些系统底层软件的 ROM 代码（软件指令和一些只读数据）。
- 内部 SRAM (400 KB)：内部静态存储器（SRAM）是易失性存储器，可以快速响应 CPU 的访问请求（通常一个 CPU 时钟周期）。
 - SRAM 中的一部分可以被配置成外部存储器访问的缓存。
 - SRAM 中的某些部分只可以被 CPU 的指令总线访问。
 - SRAM 中的某些部分既可以被 CPU 的指令总线访问，又可以被 CPU 的数据总线访问。
- RTC 存储器 (8 KB)：RTC 存储器以静态 RAM (SRAM) 方式实现，因此也是易失性存储器。但是，在 deep sleep 模式下，存放在 RTC 存储器中的数据不会丢失。

- RTC 快速存储器 (8 KB): RTC 快速存储器只可以被 CPU 访问, 通常用来存放一些在休眠模式下仍需保持的程序指令和数据。

基于上述对三种类型的内部存储器的描述, ESP32-C3 的内部存储器可以被分为三个部分: 内部 ROM (384 KB)、内部 SRAM (400 KB)、RTC 快速存储器 (8 KB)。

CPU 通过不同的总线访问这几部分内部存储器时会有些许限制 (如某些部分只允许 CPU 通过数据总线访问), 据此内部存储器可以被区分的更加细致。表 3-2 列出了所有内部存储器以及可以访问内部存储器的数据总线与指令总线地址段。

表 3-2. 内部存储器地址映射

总线类型	边界地址		容量 (KB)	目标
	低位地址	高位地址		
数据	0x3FF0_0000	0x3FF1_FFFF	128	内部 ROM 1
	0x3FC8_0000	0x3FCD_FFFF	384	内部 SRAM 1
指令	0x4000_0000	0x4003_FFFF	256	内部 ROM 0
	0x4004_0000	0x4005_FFFF	128	内部 ROM 1
	0x4037_C000	0x4037_FFFF	16	内部 SRAM 0
	0x4038_0000	0x403D_FFFF	384	内部 SRAM 1
数据/指令	0x5000_0000	0x5000_1FFF	8	RTC 快速存储器

说明:

所有的内部存储器都接受权限管理。只有获取到访问内部存储器的访问权限, 才可以执行正常的访问操作, CPU 访问内部存储器时才可以被响应。关于权限管理的更多信息, 请参考章节 2 [权限控制 \(PMS\)](#) [to be added later]。

1. 内部 ROM 0

内部 ROM 0 的容量为 256 KB, 只读。如表 3-2 所示, CPU 只可以通过指令总线地址段 0x4000_0000~0x4003_FFFF 访问这部分存储器。

2. 内部 ROM 1

内部 ROM 1 的容量为 128 KB, 只读。如表 3-2 所示, CPU 可以通过指令总线地址段 0x4004_0000~0x4005_FFFF 或数据总线地址段 0x3FF0_0000~0x3FF1_FFFF 同序访问这部分存储器。

这两段地址同序访问内部 ROM 1 是指: 地址 0x4004_0000 与 0x3FF0_0000 访问到相同的字, 0x4004_0004 与 0x3FF0_0004 访问到相同的字, 0x4004_0008 与 0x3FF0_0008 访问到相同的字, 以此类推 (下文的“同序访问”也参照此描述)。

3. 内部 SRAM 0

内部 SRAM 0 的容量为 16 KB, 可读可写。如表 3-2 所示, CPU 只可以通过指令总线访问这部分存储器。

通过权限管理, 这部分存储器可以被配置为指令缓存, 用来缓存外部存储器的指令或只读数据。此时, 已被配置为指令缓存的部分不可以被 CPU 访问。关于权限管理的更多信息, 请参考章节 2 [权限控制 \(PMS\)](#) [to be added later]。

4. 内部 SRAM 1

内部 SRAM 1 容量为 384 KB, 可读可写。如表 3-2 所示, CPU 可以通过数据或指令总线同序访问。

5. RTC 快速存储器

RTC 快速存储器容量为 8 KB，为可读可写的 SRAM。如表 3-2 所示，CPU 可以通过数据/指令总线的共用地址段 0x5000_0000 ~ 0x5000_1FFF 访问这部分存储器。

3.3.3 外部存储器

ESP32-C3 支持以 SPI、Dual SPI、Quad SPI、QPI 等接口形式连接片外 flash。ESP32-C3 还支持基于 XTS-AES 算法的硬件手动加密和自动解密功能，从而保护开发者片外 flash 中的程序和数据。

3.3.3.1 外部存储器地址映射

CPU 借助缓存（Cache）来访问外部存储器。Cache 将根据内存管理单元（Memory Management Unit, MMU）中的信息把 CPU 的地址映射为访问片外存储的实地址。经过地址映射，ESP32-C3 最大支持 16 MB 的片外 flash。

通过高速缓存，ESP32-C3 可支持以下地址空间映射。请注意，指令总线地址空间（8 MB）和数据总线地址空间（8 MB）是共用的。

- 8 MB 的指令总线地址空间以 64 KB 为单位映射到片外 flash。
- 8 MB 的数据总线（只读）地址空间以 64 KB 为单位映射到片外 flash。

表 3-3 列出了在访问外部存储器时 CPU 的数据总线与指令总线与 Cache 的对应关系。

表 3-3. 外部存储器地址映射

总线类型	边界地址		容量 (MB)	目标
	低位地址	高位地址		
数据（只读）	0x3C00_0000	0x3C7F_FFFF	8	Uniform Cache
指令	0x4200_0000	0x427F_FFFF	8	Uniform Cache

说明：

只有获取到外部存储器的访问权限，CPU 访问外部存储器时才可以被响应。关于权限管理的更多信息，请参考章节 2 权限控制 (PMS) [to be added later]。

3.3.3.2 高速缓存

如图 3-2 所示，ESP32-C3 采用一个只读的统一 cache，为 8 路组相联，容量为 16 KB，块大小为 32 字节。当 cache 处于工作状态时，将占用部分内部存储空间（参见第 3.3.2 节关于内部 SRAM 0 的描述）。

指令总线和数据总线可以同时访问该 cache，但此时 cache 只能对其中一个作出相应。当 cache 缺失时，cache 控制器会向外部存储器发起请求。

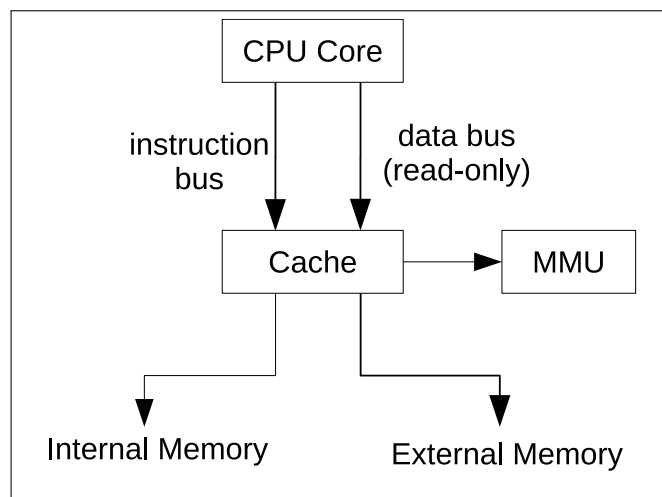


图 3-2. Cache 系统结构

3.3.3.3 Cache 操作

ESP32-C3 cache 支持如下几种操作：

1. **失效 (Invalidate)**：该操作用于删除 cache 中的有效数据。该操作完成后，删除的数据将仅存于外部存储器中。如果 CPU 接着去访问该数据，那么需要访问外部存储器。该操作包括两种类型：自动失效 (Auto-Invalidate) 和手动失效 (Manual-Invalidate)。手动失效仅对 cache 中落入指定区域的地址对应的数据做失效处理，而自动失效会对 cache 中的所有数据做失效处理。
2. **预取 (Preload)**：功能用于将指令和数据提前加载到 cache 中。预取操作的最小单位为 1 个块。预取分为手动预取 (Manual-Preload) 和自动预取 (Auto-Preload)，手动预取是指硬件按软件指定的虚地址预取一段连续的数据；自动预取是指硬件根据当前命中/缺失（取决于配置）的地址，自动地预取一段连续的数据。
3. **锁定/解锁 (Lock/Unlock)**：该操作用于保护 cache 中的数据不被替换掉。锁定分为预锁定和手动锁定。预锁定开启时，cache 在填充缺失数据到 cache 时，如果该数据落在指定区域，则将该数据锁定，未落入指定区域的数据不会被锁定。手动锁定开启时，cache 检查 cache 中的数据，并将落在指定区域的数据锁定，未落入指定区域的数据不会被锁定。当缺失发生时，cache 会优先替换掉未被锁定的那一路的数据，因此锁定区域的数据会一直保存在 cache 中。但当所有路都被锁定时，cache 将进行正常替换，就像所有路都没有被锁定一样。解锁是锁定的逆操作，但解锁只有手动解锁。

请注意，手动失效操作只对未被锁定的数据起作用。如果想对已锁定的数据执行手动失效操作，请先解锁这些数据。

3.3.4 GDMA 地址空间

ESP32-C3 中的 GDMA (General Direct Memory Access) 外设可提供直接内存访问 (Direct Memory Access, DMA) 服务，包括：

- 内部存储器中不同位置的数据搬运；
- 模块/外设和内部存储器之间的数据搬运。

GDMA 可以通过与数据总线完全相同的地址读写内部 SRAM 1，即 GDMA 通过地址访问内部 SRAM 1。请注意，GDMA 无法访问被 cache 占用的内部存储器。

ESP32-C3 中共有 7 个外设/模块可以和 GDMA 联合工作。如图 3-3 所示，其中的 7 根竖线依次对应这 7 个具有 GDMA 功能的外设/模块，横线表示 GDMA 的某一通道（可为任意通道），竖线与横线的交点表示对应外设/模块

可以访问 GDMA 的某一通道。同一行上有多个交点则表示这几个外设/模块不可以同时开启 GDMA 功能。

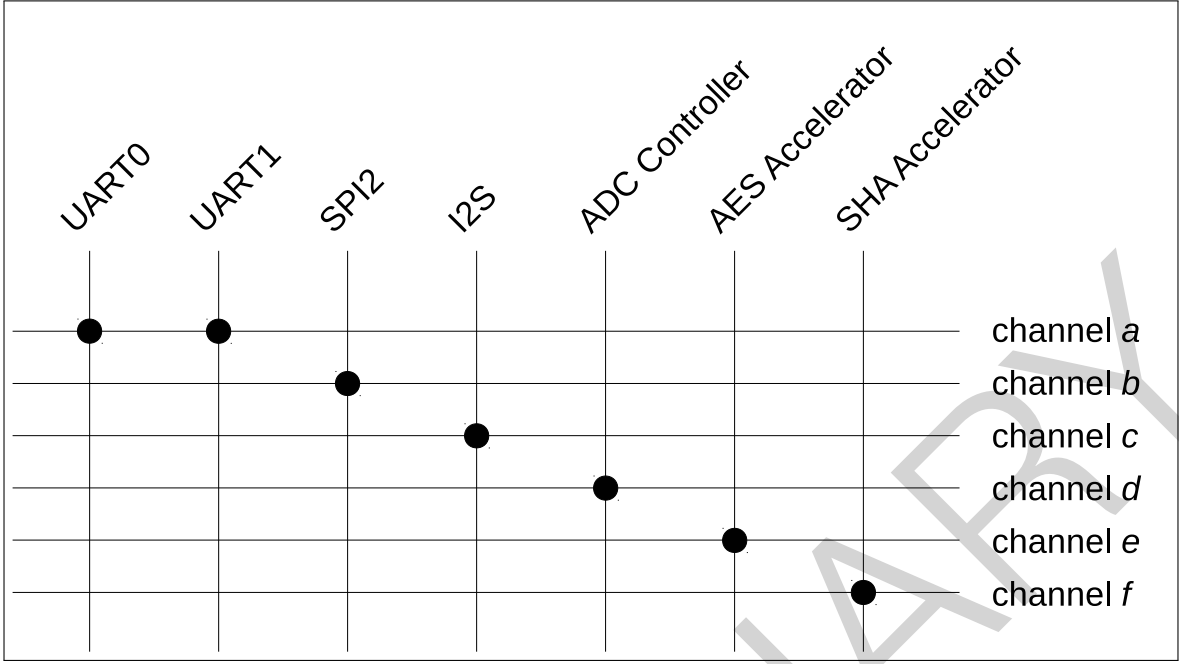


图 3-3. 具有 GDMA 功能的外设/模块

具有 GDMA 功能的模块/外设通过 GDMA 可以访问任何 GDMA 可以访问到的存储器。更多关于 GDMA 的信息，请参考章节 2 通用 DMA 控制器 (GDMA)。

说明：
当使用 GDMA 访问任何存储器时，都需要获取对应的访问权限，否则访问将会失败。关于权限管理的更多信息，请参考章节 2 权限控制 (PMS) [to be added later]。

3.3.5 模块/外设

CPU 可通过数据/指令总线的共用地址段 0x6000_0000 ~ 0x600D_0FFF 访问模块/外设。

3.3.5.1 模块/外设地址空间映射

表 3-4 详细列出了模块/外设地址空间的各段地址与其能访问到的模块/外设的映射关系。其中，“边界地址”（包括低位地址和高位地址）栏中的两列数值共同决定了对应模块/外设的地址空间。

表 3-4. 模块/外设地址空间映射表

目标	边界地址		容量 (KB)	说明
	低位地址	高位地址		
UART 控制器 0	0x6000_0000	0x6000_0FFF	4	
保留	0x6000_1000	0x6000_1FFF		
SPI 控制器 1	0x6000_2000	0x6000_2FFF	4	
SPI 控制器 0	0x6000_3000	0x6000_3FFF	4	
GPIO	0x6000_4000	0x6000_4FFF	4	

见下页

表 3-4 – 接上页

目标	边界地址		容量 (KB)	说明
	低位地址	高位地址		
保留	0x6000_5000	0x6000_6FFF		
TIMER	0x6000_7000	0x6000_7FFF	4	
低功耗管理	0x6000_8000	0x6000_8FFF	4	
IO MUX	0x6000_9000	0x6000_9FFF	4	
保留	0x6000_A000	0x6000_FFFF		
UART 控制器 1	0x6001_0000	0x6001_0FFF	4	
保留	0x6001_1000	0x6001_2FFF		
I2C 控制器	0x6001_3000	0x6001_3FFF	4	
UHCIO	0x6001_4000	0x6001_4FFF	4	
保留	0x6001_5000	0x6001_5FFF		
红外遥控	0x6001_6000	0x6001_6FFF	4	
保留	0x6001_7000	0x6001_8FFF		
LED PWM 控制器	0x6001_9000	0x6001_9FFF	4	
eFuse 控制器	0x6001_A000	0x6001_AFFF	4	
保留	0x6001_B000	0x6001_EFFF		
定时器组 0	0x6001_F000	0x6001_FFFF	4	
定时器组 1	0x6002_0000	0x6002_0FFF	4	
保留	0x6002_1000	0x6002_2FFF		
系统定时器	0x6002_3000	0x6002_3FFF	4	
SPI 控制器 2	0x6002_4000	0x6002_4FFF	4	
保留	0x6002_5000	0x6002_5FFF		
APB 控制器	0x6002_6000	0x6002_6FFF	4	
保留	0x6002_7000	0x6002_AFFF		
双线汽车接口	0x6002_B000	0x6002_BFFF	4	
保留	0x6002_C000	0x6002_CFFF		
I2S 控制器	0x6002_D000	0x6002_DFFF	4	
保留	0x6002_E000	0x6003_9FFF		
AES 加速器	0x6003_A000	0x6003_AFFF	4	
SHA 加速器	0x6003_B000	0x6003_BFFF	4	
RSA 加速器	0x6003_C000	0x6003_CFFF	4	
数字签名	0x6003_D000	0x6003_DFFF	4	
HMAC 加速器	0x6003_E000	0x6003_EFFF	4	
通用 DMA 控制器	0x6003_F000	0x6003_FFFF	4	
ADC 控制器	0x6004_0000	0x6004_0FFF	4	
保留	0x6004_1000	0x6002_FFFF		
USB Serial/JTAG 控制器	0x6004_3000	0x6004_3FFF	4	
保留	0x6004_4000	0x600B_FFFF		
系统寄存器	0x600C_0000	0x600C_0FFF	4	
Sensitive Register	0x600C_1000	0x600C_1FFF	4	
中断矩阵	0x600C_2000	0x600C_2FFF	4	
保留	0x600C_3000	0x600C_3FFF		

见下页

表 3-4 – 接上页

目标	边界地址		容量 (KB)	说明
	低位地址	高位地址		
Configure Cache	0x600C_4000	0x600C_BFFF	32	
片外存储器加密与解密	0x600C_C000	0x600C_CFFF	4	
保留	0x600C_D000	0x600C_DFFF		
辅助调试	0x600C_E000	0x600C_EFFF	4	
保留	0x600C_F000	0x600C_FFFF		
World 控制器	0x600D_0000	0x600D_0FFF	4	

4 eFuse 控制器 (EFUSE)

4.1 概述

ESP32-C3 系统中有一块 4096 位的 eFuse，其中存储着参数内容。eFuse 的各个位一旦被烧写为 1，则不能再恢复为 0。eFuse 控制器按照软件配置完成对 eFuse 中各参数中的各个位的烧写。这些参数有些可以通过 eFuse 控制器被软件读取，有些直接由硬件模块使用。

4.2 主要特性

- 一次性可编程存储
- 烧写保护可配置
- 软件读取保护可配置
- 使用多种硬件编码方式保护参数内容

4.3 功能描述

4.3.1 结构

eFuse 从结构上分成 11 个块 (BLOCK0 ~ BLOCK10)。

BLOCK0 存储大部分参数，其中 9 位供硬件使用，软件不可见；还有 60 位处于保留状态，留作未来使用。

表 4-1 列出了 BLOCK0 中的参数名称、偏移地址、位宽、是否可供硬件使用、烧写保护，以及描述信息。

在这些参数中，**EFUSE_WR_DIS** 用于控制其他参数的烧写，**EFUSE_RD_DIS** 用于控制软件读取 BLOCK4 ~ BLOCK10。更多关于这两个参数的信息请见章节 4.3.1.1、4.3.1.2。

表 4-1. BLOCK0 参数

参数	偏移	位宽	硬件使用	EFUSE_WR_DIS 烧写保护位	描述
EFUSE_WR_DIS	0	32	Y	N/A	禁止 eFuse 烧写
EFUSE_RD_DIS	32	7	Y	0	禁止软件读取 eFuse BLOCK4 ~ 10 的内容
EFUSE_DIS_ICACHE	40	1	Y	2	关闭 ICache
EFUSE_DIS_USB_JTAG	41	1	Y	2	关闭 usb 转 jtag 功能
EFUSE_DIS_DOWNLOAD_ICACHE	42	1	Y	2	在 Download 模式下关闭 ICache
EFUSE_DIS_USB_SERIAL_JTAG	43	1	Y	2	禁用 usb_serial_jtag 模块
EFUSE_DIS_FORCE_DOWNLOAD	44	1	Y	2	禁止强制芯片进入 Download 模式
EFUSE_DIS_TWAI	46	1	Y	2	关闭 TWAI 控制器功能
EFUSE_JTAG_SEL_ENABLE	47	1	Y	2	置 1 表明直接使用 jtag 功能。
EFUSE_SOFT_DIS_JTAG	48	3	Y	31	烧写奇数个 1 表示禁用 JTAG，可通过 HMAC 重新启动
EFUSE_DIS_PAD_JTAG	51	1	Y	2	硬件永远禁用 JTAG
EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT	52	1	Y	2	在 download boot 模式下禁用 flash 加密功能
EFUSE_USB_EXCHG_PINS	57	1	Y	30	交换 USB D+/D- 管脚
EFUSE_VDD_SPI_AS_GPIO	58	1	N	30	VDD SPI 引脚和普通 gpio 选择信号，置 1 表明作为普通 gpio 引脚。
EFUSE_WDT_DELAY_SEL	80	2	Y	3	选择 RTC WDT 超时阈值
EFUSE_SPI_BOOT_CRYPT_CNT	82	3	Y	4	使能 SPI boot 加解密，奇数个 1：使能；偶数个 1：关闭
EFUSE_SECURE_BOOT_KEY_REVOKE0	85	1	N	5	使能撤销第一个 secure boot（安全启动）密钥
EFUSE_SECURE_BOOT_KEY_REVOKE1	86	1	N	6	使能撤销第二个 secure boot 密钥
EFUSE_SECURE_BOOT_KEY_REVOKE2	87	1	N	7	使能撤销第三个 secure boot 密钥
EFUSE_KEY_PURPOSE_0	88	4	Y	8	Key0 用途 (purpose)，见表 4-2
EFUSE_KEY_PURPOSE_1	92	4	Y	9	Key1 用途，见表 4-2
EFUSE_KEY_PURPOSE_2	96	4	Y	10	Key2 用途，见表 4-2
EFUSE_KEY_PURPOSE_3	100	4	Y	11	Key3 用途，见表 4-2
EFUSE_KEY_PURPOSE_4	104	4	Y	12	Key4 用途，见表 4-2
EFUSE_KEY_PURPOSE_5	108	4	Y	13	Key5 用途，见表 4-2

参数	偏移	位宽	硬件使用	EFUSE_WR_DIS 烧写保护位	描述
EFUSE_SECURE_BOOT_EN	116	1	N	15	使能 secure boot
EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE	117	1	N	16	Secure boot 的撤销采用激进策略
EFUSE_FLASH_TPUW	124	4	N	18	上电后 flash 等待时间，单位为 ms/2，值为 15 时，等待时间为 7.5 ms
EFUSE_DIS_DOWNLOAD_MODE	128	1	N	18	关闭所有 download boot 模式
EFUSE_UART_PRINT_CHANNEL	130	1	N	18	置 1 禁止 usb 打印功能
EFUSE_DIS_USB_DOWNLOAD_MODE	132	1	N	18	在 UART download boot 模式下关闭 USB OTG 下载功能
EFUSE_ENABLE_SECURITY_DOWNLOAD	133	1	N	18	使能 UART 安全下载模式（仅支持读写 flash）
EFUSE_UART_PRINT_CONTROL	134	2	N	18	控制 UART boot 信息输出模式。2'b00：强制打印；2'b01：由 GPIO8 控制，低电平打印；2'b10：由 GPIO 8 控制，高电平打印；2'b11：强制关闭打印
EFUSE_FORCE_SEND_RESUME	141	1	N	18	强制 ROM 代码在 SPI 启动过程中发送 SPI flash 继续指令
EFUSE_SECURE_VERSION	142	16	N	18	安全版本（用于 ESP-IDF 的防回滚功能）
EFUSE_ERR_RST_ENABLE	159	1	N	19	1：启用 BLOCK0 错误寄存器检查；0：禁用上述检查

表 4-2 为密钥用途各个数值对应的含义。通过配置参数 EFUSE_KEY_PURPOSE_*n* 来声明 KEY_{*n*} 用途 (*n*: 0 ~ 5)。

表 4-2. 密钥用途数值对应的含义

密钥用途数值	含义
0	指定为用户使用（仅为软件使用）
1	保留
2	指定为 XTS_AES_256_KEY_1 使用（用于 flash/SRAM 加解密）
3	指定为 XTS_AES_256_KEY_2 使用（用于 flash/SRAM 加解密）
4	指定为 XTS_AES_128_KEY 使用（用于 flash/SRAM 加解密）
5	指定为 HMAC Downstream（下行）模式（JTAG 和数字签名）使用
6	指定为 HMAC Downstream 模式下的 JTAG 使用
7	指定为 HMAC Downstream 模式下的数字签名使用
8	指定为 HMAC Upstream（上行）模式使用
9	指定为 SECURE_BOOT_DIGEST0 使用（secure boot 密钥摘要）
10	指定为 SECURE_BOOT_DIGEST1 使用（secure boot 密钥摘要）
11	指定为 SECURE_BOOT_DIGEST2 使用（secure boot 密钥摘要）

表 4-3 列出了 BLOCK1 ~ BLOCK10 中存储的参数的信息。

表 4-3. BLOCK1-10 参数

块	参数	位宽	硬件使用	EFUSE_WR_DIS 烧写保护位	EFUSE_RD_DIS 软件读取保护位	描述
BLOCK1	EFUSE_MAC	48	N	20	N/A	MAC 地址
	EFUSE_SPI_PAD_CONFIGURE	[0:5]	N	20	N/A	CLK
		[6:11]	N	20	N/A	Q (D1)
		[12:17]	N	20	N/A	D (D0)
		[18:23]	N	20	N/A	CS
		[24:29]	N	20	N/A	HD (D3)
		[30:35]	N	20	N/A	WP (D2)
		[36:41]	N	20	N/A	DQS
		[42:47]	N	20	N/A	D4
		[48:53]	N	20	N/A	D5
		[54:59]	N	20	N/A	D6
		[60:65]	N	20	N/A	D7
	EFUSE_SYS_DATA_PART0	78	N	20	N/A	系统数据
BLOCK2	EFUSE_SYS_DATA_PART1	256	N	21	N/A	系统数据
BLOCK3	EFUSE_USR_DATA	256	N	22	N/A	用户数据
BLOCK4	EFUSE_KEY0_DATA	256	Y	23	0	KEY0 或用户数据
BLOCK5	EFUSE_KEY1_DATA	256	Y	24	1	KEY1 或用户数据
BLOCK6	EFUSE_KEY2_DATA	256	Y	25	2	KEY2 或用户数据
BLOCK7	EFUSE_KEY3_DATA	256	Y	26	3	KEY3 或用户数据
BLOCK8	EFUSE_KEY4_DATA	256	Y	27	4	KEY4 或用户数据
BLOCK9	EFUSE_KEY5_DATA	256	Y	28	5	KEY5 或用户数据

块	参数	位宽	硬件使用	EFUSE_WR_DIS 烧写保护位	EFUSE_RD_DIS 软件读取保护位	描述
BLOCK10	EFUSE_SYS_DATA_PART2	256	N	29	6	系统数据

其中, BLOCK4 ~ 9 分别存储 KEY0 ~ 5, 表示 eFuse 中至多可以烧写 6 个 256 位的密钥。每烧写一个密钥, 还需要烧写该密钥用途的数值 (见表 4-2)。例如, 软件将用于 HMAC Downstream 模式下的 JTAG 功能的密钥烧写到 KEY3 (即 BLOCK7), 还需要将密钥用途的数值 6 烧写到 EFUSE_KEY_PURPOSE_3。

BLOCK1 ~ BLOCK10 均采用 RS 编码方式, 因此参数烧写受到一定的限制, 具体请参考章节 4.3.1.3 和章节 4.3.2。

4.3.1.1 EFUSE_WR_DIS

参数 EFUSE_WR_DIS 决定了 eFuse 中所有的参数是否处于烧写保护状态。烧写完 EFUSE_WR_DIS 参数后, 需要更新 eFuse 读寄存器才能生效。

表 4-1 以及表 4-3 中的“EFUSE_WR_DIS 烧写保护位”列描述了各参数的烧写保护状态具体由 EFUSE_WR_DIS 的哪个位决定。

当某个参数对应的烧写保护位为 0 时, 表示此参数未处于烧写保护状态, 可以烧写该参数, 但已经被烧写的参数不能被重复烧写。

当某个参数对应的烧写保护位为 1 时, 表示此参数处于烧写保护状态, 此参数的每一个位都无法被更改, 未被烧写的位永远为 0, 已经被烧写的位永远为 1。所以如果某个参数已经处于烧写保护状态了, 则会一直处在该状态, 无法再更改。

4.3.1.2 EFUSE_RD_DIS

所有参数中, 只有 BLOCK4 ~ BLOCK10 的参数受软件读取保护状态的约束, 即表 4-3 中“EFUSE_RD_DIS 软件读取保护”列非“N/A”的参数。烧写完 EFUSE_RD_DIS 参数后, 需要更新 eFuse 读寄存器才能生效。

参数 EFUSE_RD_DIS 中的某个位为 0, 表示此位管理的参数未处于软件读取保护状态; 某个位为 1, 表示此位管理的参数处于软件读取保护状态。

除 BLOCK4 ~ BLOCK10 之外, 其他参数不受软件读取保护状态的约束, 均可被软件读取。

BLOCK4 ~ BLOCK10 即使被配置处于读取保护状态, 仍然可以通过设置 EFUSE_KEY_PURPOSE_0 被硬件使用。

4.3.1.3 数据存储方式

eFuse 使用硬件编码机制保护数据, 对用户不可见。

BLOCK0 使用 4 备份方式存储参数, 即 BLOCK0 中的所有参数 (除了 EFUSE_WR_DIS) 均在 eFuse 中存储了 4 份。4 备份机制对软件不可见。

BLOCK1 ~ BLOCK10 使用 RS (44, 32) 编码方式, 最多支持自动校正 6 个字节。本文 RS (44, 32) 使用的本源多项式为 $p(x) = x^8 + x^4 + x^3 + x^2 + 1$ 。

如图 4-1 和 4-2 所示, 软件对 32 字节参数进行 RS (44, 32) 编码处理, 将 32 字节数据处理为 44 字节, 其中:

- 字节 [0:31] 为数据本身

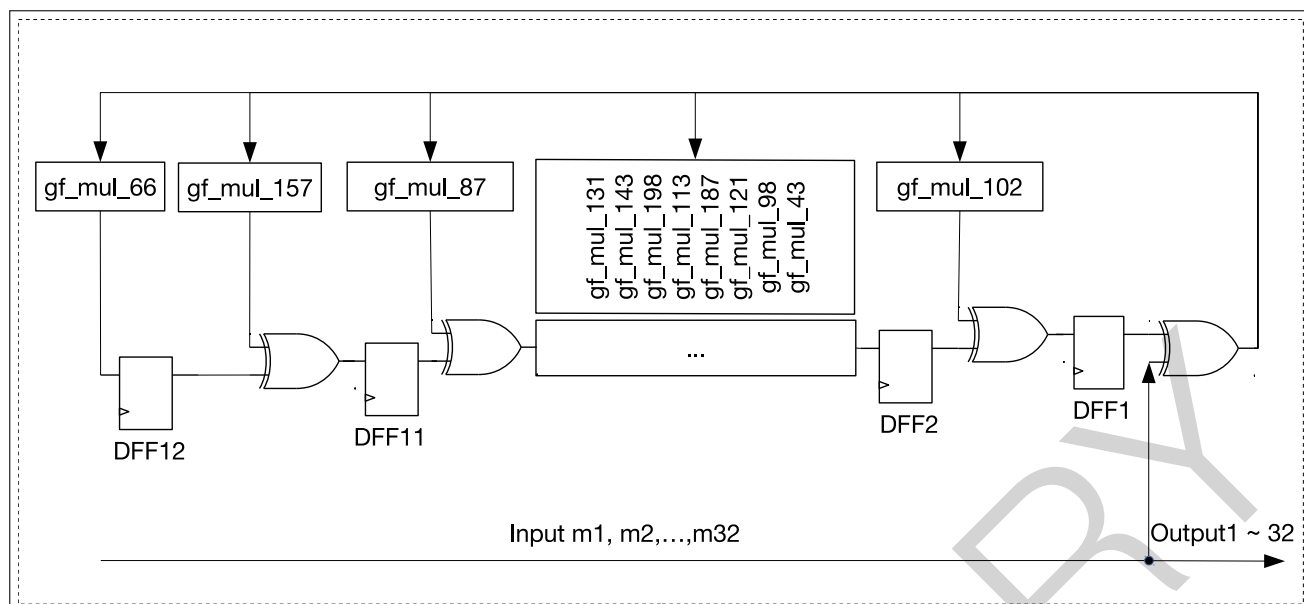


图 4-1. 移位寄存器电路图 (前 32 字节)

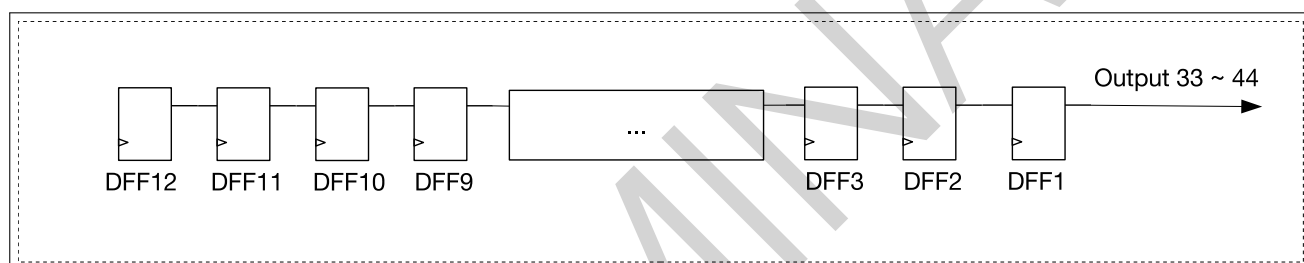


图 4-2. 移位寄存器电路图 (后 12 字节)

- 字节 [32:43] 为储存在 8 位触发器 DFF1, DFF2, ..., DFF12 中的奇偶校验字节 (gf_mul_n 为 $GF(2^8)$ 域中某一字节数据与元素 α^n 相乘的结果, n 为整数)

然后, 硬件将这 44 字节数据一起烧入 eFuse。eFuse 控制器会在读 eFuse 的过程中自动完成解码和自动校正。

由于 RS 校验码是在整个 256 位的 eFuse block 上生成的, 因此每个 block 只能写入一次。

4.3.2 软件烧写参数

烧写 eFuse 参数时, 需要按块烧写。BLOCK0 ~ BLOCK10 共用同一段地址来存储即将烧写的参数。通过配置 [EFUSE_BLK_NUM](#) 参数表明当前需要烧写的是哪一个块。

烧写 BLOCK0

当 [EFUSE_BLK_NUM](#) = 0 时, 烧写 BLOCK0。[EFUSE_PGM_DATA0_REG](#) 寄存器存储着 [EFUSE_WR_DIS](#)。[EFUSE_PGM_DATA1_REG](#) ~ [EFUSE_PGM_DATA5_REG](#) 用来存储即将烧写的参数的有效信息, 其中 9 位为硬件可用、软件不可见的有效信息, 必须写入 0, 对应位置为:

- [EFUSE_PGM_DATA1_REG](#)[24:21]
- [EFUSE_PGM_DATA1_REG](#)[31:27]

[EFUSE_PGM_DATA6_REG](#) ~ [EFUSE_PGM_DATA7_REG](#) 以及

[EFUSE_PGM_CHECK_VALUE0_REG](#) ~ [EFUSE_PGM_CHECK_VALUE2_REG](#) 中的数据不影响 BLOCK0 的烧写。

烧写 BLOCK1

当 `EFUSE_BLK_NUM = 1` 时, `EFUSE_PGM_DATA0_REG ~ EFUSE_PGM_DATA5_REG` 存储着 BLOCK1 即将烧写的参数, `EFUSE_PGM_CHECK_VALUE0_REG ~ EFUSE_PGM_CHECK_VALUE2_REG` 中存储着对应的 RS 校验码。`EFUSE_PGM_DATA6_REG ~ EFUSE_PGM_DATA7_REG` 中的数据不影响 BLOCK1 的烧写。软件计算 BLOCK1 的 RS 校验码时, 应该视这 8 个字节的数据为 0。

烧写 BLOCK2 ~ 10

当 `EFUSE_BLK_NUM = 2 ~ 10` 时, `EFUSE_PGM_DATA0_REG ~ EFUSE_PGM_DATA7_REG` 存储着即将烧写的参数, `EFUSE_PGM_CHECK_VALUE0_REG ~ EFUSE_PGM_CHECK_VALUE2_REG` 中存储着对应的 RS 校验码。

烧写流程

烧写参数的流程如下:

1. 配置 `EFUSE_BLK_NUM` 参数, 决定烧写哪一个块。
2. 将需要烧写的参数填写到寄存器 `EFUSE_PGM_DATA0_REG ~ EFUSE_PGM_DATA7_REG` 和 `EFUSE_PGM_CHECK_VALUE0_REG ~ EFUSE_PGM_CHECK_VALUE2_REG` 中。
3. 确保 eFuse 烧写电压 VDDQ 的配置正确, 具体请参考章节 4.3.4。
4. 配置寄存器 `EFUSE_CONF_REG` 的 `EFUSE_OP_CODE` 位域为 0x5A5A。
5. 配置寄存器 `EFUSE_CMD_REG` 的 `EFUSE_PGM_CMD` 位域为 1。
6. 轮询寄存器 `EFUSE_CMD_REG` 直到其为 0x0, 或者等待烧写完成中断产生。识别烧写/读取完成中断产生的方法详见章节 4.3.3 最后的说明。
7. 将 `EFUSE_PGM_DATA0_REG ~ EFUSE_PGM_DATA7_REG` 和 `EFUSE_PGM_CHECK_VALUE0_REG ~ EFUSE_PGM_CHECK_VALUE2_REG` 中写入的参数清零。
8. 执行更新 eFuse 读寄存器操作使写入的新值生效, 具体请参考章节 4.3.3。
9. 检查错误寄存器内容。若读取错误寄存器内数值不为 0, 需要再次执行上述步骤 1 ~ 7 重新烧写一次。解决烧写不充分导致错误寄存器内数值不为 0 的问题, 对于不同的 eFuse 块, 需要检查的错误寄存器如下。
 - BLOCK0: `EFUSE_RD_REPEAT_ERR0_REG ~ EFUSE_RD_REPEAT_ERR4_REG`
 - BLOCK1: `EFUSE_RD_RS_ERR0_REG[2:0]`, `EFUSE_RD_RS_ERR0_REG[7]`
 - BLOCK2: `EFUSE_RD_RS_ERR0_REG[6:4]`, `EFUSE_RD_RS_ERR0_REG[11]`
 - BLOCK3: `EFUSE_RD_RS_ERR0_REG[10:8]`, `EFUSE_RD_RS_ERR0_REG[15]`
 - BLOCK4: `EFUSE_RD_RS_ERR0_REG[14:12]`, `EFUSE_RD_RS_ERR0_REG[19]`
 - BLOCK5: `EFUSE_RD_RS_ERR0_REG[18:16]`, `EFUSE_RD_RS_ERR0_REG[23]`
 - BLOCK6: `EFUSE_RD_RS_ERR0_REG[22:20]`, `EFUSE_RD_RS_ERR0_REG[27]`
 - BLOCK7: `EFUSE_RD_RS_ERR0_REG[26:24]`, `EFUSE_RD_RS_ERR0_REG[31]`
 - BLOCK8: `EFUSE_RD_RS_ERR0_REG[30:28]`, `EFUSE_RD_RS_ERR1_REG[3]`
 - BLOCK9: `EFUSE_RD_RS_ERR1_REG[2:0]`, `EFUSE_RD_RS_ERR1_REG[2:0][7]`
 - BLOCK10: `EFUSE_RD_RS_ERR1_REG[2:0][6:4]`

限制

BLOCK0 中不同的参数，甚至对于同一个参数中的不同位可以在多次烧写中分别完成。但是并不推荐这样做，而是建议尽量减少烧写次数。我们建议对于某个参数中的所有需要烧写的位都在一次烧写中完成。并且当 EFUSE_WR_DIS 的某个位管理的所有参数都烧写之后，就立即烧写 EFUSE_WR_DIS 的这个位。甚至可以在同一次烧写中既烧写 EFUSE_WR_DIS 的某个位管理的所有参数，同时也烧写 EFUSE_WR_DIS 的这个位。另外严禁对已经烧写了的位重复烧写，否则将发生烧写错误。

BLOCK1 中数据信息在出厂时已经烧写完毕，不允许再次烧写。

BLOCK2 ~ 10 中每一个 BLOCK 都只能烧写一次，不允许重复烧写。

4.3.3 软件读取参数

软件不能直接读取 eFuse 中烧写的信息内容。eFuse 控制器能够将烧写的信息读取到对应的地址段的寄存器内，软件再通过读取以 EFUSE_RD_ 开始的寄存器来获取 eFuse 信息。下表 4-4 列出了读取数据的寄存器名称以及对应烧写时的烧写寄存器名称。

表 4-4. 软件读取寄存器信息

BLOCK	读寄存器	烧写寄存器
0	EFUSE_RD_WR_DIS_REG	EFUSE_PGM_DATA0_REG
0	EFUSE_RD_REPEAT_DATA0 ~ 4_REG	EFUSE_PGM_DATA1 ~ 5_REG
1	EFUSE_RD_MAC_SPI_SYS_0 ~ 5_REG	EFUSE_PGM_DATA0 ~ 5_REG
2	EFUSE_RD_SYS_PART1_0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG
3	EFUSE_RD_USR_DATA0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG
4-9	EFUSE_RD_KEY _n _DATA0 ~ 7_REG (<i>n</i> : 0 ~ 5)	EFUSE_PGM_DATA0 ~ 7_REG
10	EFUSE_RD_SYS_PART2_0 ~ 7_REG	EFUSE_PGM_DATA0 ~ 7_REG

更新 eFuse 读寄存器

eFuse 控制器读取内部 eFuse 来更新相应寄存器的数据。读取操作在系统复位时进行，也可以根据需要由软件手动触发（例如在需要读取新烧写 eFuse 中的数据内容时）。软件触发 eFuse 读取操作的流程如下：

1. 配置寄存器 EFUSE_CONF_REG 的 EFUSE_OP_CODE 位域为 0x5AA5。
2. 配置寄存器 EFUSE_CMD_REG 的 EFUSE_READ_CMD 位域为 1。
3. 轮询寄存器 EFUSE_CMD_REG 直到其为 0x0，或者等待 read_done interrupt（读取完成中断）产生，识别烧写/读取完成中断产生的方法详见下方说明。
4. 软件从 eFuse 存储器中读取参数的值。

eFuse 读寄存器中的数值将一直保持到下一次执行更新 eFuse 读操作。

烧写错误检测

烧写错误记录寄存器允许软件检测 eFuse 参数的备份是否有不一致的错误。

EFUSE_RD_REPEAT_ERR0 ~ 3_REG 寄存器用于指示 BLOCK0 中除了 EFUSE_WR_DIS 外的其他参数的烧写是否出错（对应位为 1 代表烧写出错，此位作废；为 0 代表烧写正确）。

EFUSE_RD_RS_ERR0 ~ 1_REG 寄存器记录 eFuse 读 BLOCK1 ~ BLOCK10 过程中，纠错的字节数目以及 RS 解码是否失败的信息。

每次更新 eFuse 读寄存器操作完成之后，上述寄存器内的数值都会被更新。

识别烧写/读取操作完成

识别烧写/读取操作完成的方法如下。位 1 对应烧写操作，位 0 对应读取操作。

- 方法 1:
 1. 轮询寄存器 `EFUSE_INT_RAW_REG` 的位 1/0，直到位 1/0 为 1，表示烧写/读取操作完成。
- 方法 2:
 1. 将寄存器 `EFUSE_INT_ENA_REG` 的位 1/0 置 1，使 eFuse 控制器能够产生烧写/读取完成中断。
 2. 配置中断矩阵使 CPU 能够响应 eFuse 的中断信号，可参见 8 中断矩阵 (*INTMATRIX*)。
 3. 等待烧写/读取完成中断产生。
 4. 对寄存器 `EFUSE_INT_CLR_REG` 的位 1/0 置 1 以清除烧写/读取完成中断。

注意事项

在 eFuse 控制器执行寄存器更新操作过程中，会复用 `EFUSE_PGM_DATAn_REG` ($n=0, 1, \dots, 7$) 寄存器的存储空间，所以在启动 eFuse 控制器更新寄存器之前，不要将有意义的写入上述寄存器中。

芯片启动过程中，eFuse 控制器会自动更新 eFuse 数据到软件可访问的寄存器。用户可以通过读取相应的寄存器获取 eFuse 内烧写的的数据。因此，用户无需再驱动 eFuse 控制器执行读更新操作。

4.3.4 eFuse VDDQ 时序

eFuse 控制器工作在 20 MHz 时钟频率下，其烧写电压 VDDQ 的配置参数需要满足以下条件：

- `EFUSE_DAC_NUM` (烧写电压上升周期数)，默认烧写电压为 2.5 V，每个上升周期增加 0.01 V，该参数对应的默认值为 255；
- `EFUSE_DAC_CLK_DIV` (烧写电压时钟分频系数)，要求烧写电压时钟周期大于 1 μ s；
- `EFUSE_PWR_ON_NUM` (eFuse 烧写电压上电等待时间)，要求该等待时间结束后烧写电压已稳定，即要求配置数值大于 `EFUSE_DAC_CLK_DIV * EFUSE_DAC_NUM`；
- `EFUSE_PWR_OFF_NUM` (烧写电压掉电等待时间)，要求该时间大于 10 μ s；

表 4-5. VDDQ 默认时序参数配置

<code>EFUSE_DAC_NUM</code>	<code>EFUSE_DAC_CLK_DIV</code>	<code>EFUSE_PWR_ON_NUM</code>	<code>EFUSE_PWR_OFF_NUM</code>
0xFF	0x28	0x3000	0x190

4.3.5 硬件模块使用参数

硬件模块使用参数是通过电路连接实现的，软件无法干预这个过程。硬件使用的参数为表 4-1 和 4-3 “硬件使用”一栏中标记为“Y”的参数。

4.3.6 中断

- 烧写完成中断：当 eFuse 烧写完成后，此中断被触发。如果要启动该中断信号，需将寄存器 `EFUSE_INT_ENA_REG` 的 `EFUSE_PGM_DONE_INT_ENA` 域置 1。
- 读取完成中断：当 eFuse 读取完成后，此中断被触发。如果要启动该中断信号，需将寄存器 `EFUSE_INT_ENA_REG` 的 `EFUSE_READ_DONE_INT_ENA` 域置 1。

4.4 寄存器列表

本小节的所有地址均为相对于 eFuse 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

07

名称	描述	地址	访问
烧写数据寄存器			
EFUSE_PGM_DATA0_REG	存放待烧写数据的第 0 个寄存器内容	0x0000	R/W
EFUSE_PGM_DATA1_REG	存放待烧写数据的第 1 个寄存器内容	0x0004	R/W
EFUSE_PGM_DATA2_REG	存放待烧写数据的第 2 个寄存器内容	0x0008	R/W
EFUSE_PGM_DATA3_REG	存放待烧写数据的第 3 个寄存器内容	0x000C	R/W
EFUSE_PGM_DATA4_REG	存放待烧写数据的第 4 个寄存器内容	0x0010	R/W
EFUSE_PGM_DATA5_REG	存放待烧写数据的第 5 个寄存器内容	0x0014	R/W
EFUSE_PGM_DATA6_REG	存放待烧写数据的第 6 个寄存器内容	0x0018	R/W
EFUSE_PGM_DATA7_REG	存放待烧写数据的第 7 个寄存器内容	0x001C	R/W
EFUSE_PGM_CHECK_VALUE0_REG	存放待烧写 RS 代码的第 0 个寄存器数据	0x0020	R/W
EFUSE_PGM_CHECK_VALUE1_REG	存放待烧写 RS 代码的第 1 个寄存器数据	0x0024	R/W
EFUSE_PGM_CHECK_VALUE2_REG	存放待烧写 RS 代码的第 2 个寄存器数据	0x0028	R/W
读取数据寄存器			
EFUSE_RD_WR_DIS_REG	BLOCK0 的第 0 个寄存器内容	0x002C	RO
EFUSE_RD_REPEAT_DATA0_REG	BLOCK0 的第 1 个寄存器内容	0x0030	RO
EFUSE_RD_REPEAT_DATA1_REG	BLOCK0 的第 2 个寄存器内容	0x0034	RO
EFUSE_RD_REPEAT_DATA2_REG	BLOCK0 的第 3 个寄存器内容	0x0038	RO
EFUSE_RD_REPEAT_DATA3_REG	BLOCK0 的第 4 个寄存器内容	0x003C	RO
EFUSE_RD_REPEAT_DATA4_REG	BLOCK0 的第 5 个寄存器内容	0x0040	RO
EFUSE_RD_MAC_SPI_SYS_0_REG	BLOCK1 的第 0 个寄存器内容	0x0044	RO
EFUSE_RD_MAC_SPI_SYS_1_REG	BLOCK1 的第 1 个寄存器内容	0x0048	RO
EFUSE_RD_MAC_SPI_SYS_2_REG	BLOCK1 的第 2 个寄存器内容	0x004C	RO
EFUSE_RD_MAC_SPI_SYS_3_REG	BLOCK1 的第 3 个寄存器内容	0x0050	RO
EFUSE_RD_MAC_SPI_SYS_4_REG	BLOCK1 的第 4 个寄存器内容	0x0054	RO
EFUSE_RD_MAC_SPI_SYS_5_REG	BLOCK1 的第 5 个寄存器内容	0x0058	RO
EFUSE_RD_SYS_PART1_DATA0_REG	BLOCK2 (system) 的第 0 个寄存器内容	0x005C	RO
EFUSE_RD_SYS_PART1_DATA1_REG	BLOCK2 (system) 的第 1 个寄存器内容	0x0060	RO
EFUSE_RD_SYS_PART1_DATA2_REG	BLOCK2 (system) 的第 2 个寄存器内容	0x0064	RO
EFUSE_RD_SYS_PART1_DATA3_REG	BLOCK2 (system) 的第 3 个寄存器内容	0x0068	RO
EFUSE_RD_SYS_PART1_DATA4_REG	BLOCK2 (system) 的第 4 个寄存器内容	0x006C	RO
EFUSE_RD_SYS_PART1_DATA5_REG	BLOCK2 (system) 的第 5 个寄存器内容	0x0070	RO
EFUSE_RD_SYS_PART1_DATA6_REG	BLOCK2 (system) 的第 6 个寄存器内容	0x0074	RO
EFUSE_RD_SYS_PART1_DATA7_REG	BLOCK2 (system) 的第 7 个寄存器内容	0x0078	RO
EFUSE_RD_USR_DATA0_REG	BLOCK3 (user) 的第 0 个寄存器内容	0x007C	RO
EFUSE_RD_USR_DATA1_REG	BLOCK3 (user) 的第 1 个寄存器内容	0x0080	RO
EFUSE_RD_USR_DATA2_REG	BLOCK3 (user) 的第 2 个寄存器内容	0x0084	RO
EFUSE_RD_USR_DATA3_REG	BLOCK3 (user) 的第 3 个寄存器内容	0x0088	RO
EFUSE_RD_USR_DATA4_REG	BLOCK3 (user) 的第 4 个寄存器内容	0x008C	RO

名称	描述	地址	访问
EFUSE_RD_USR_DATA5_REG	BLOCK3 (user) 的第 5 个寄存器内容	0x0090	RO
EFUSE_RD_USR_DATA6_REG	BLOCK3 (user) 的第 6 个寄存器内容	0x0094	RO
EFUSE_RD_USR_DATA7_REG	BLOCK3 (user) 的第 7 个寄存器内容	0x0098	RO
EFUSE_RD_KEY0_DATA0_REG	BLOCK4 (KEY0) 的第 0 个寄存器内容	0x009C	RO
EFUSE_RD_KEY0_DATA1_REG	BLOCK4 (KEY0) 的第 1 个寄存器内容	0x00A0	RO
EFUSE_RD_KEY0_DATA2_REG	BLOCK4 (KEY0) 的第 2 个寄存器内容	0x00A4	RO
EFUSE_RD_KEY0_DATA3_REG	BLOCK4 (KEY0) 的第 3 个寄存器内容	0x00A8	RO
EFUSE_RD_KEY0_DATA4_REG	BLOCK4 (KEY0) 的第 4 个寄存器内容	0x00AC	RO
EFUSE_RD_KEY0_DATA5_REG	BLOCK4 (KEY0) 的第 5 个寄存器内容	0x00B0	RO
EFUSE_RD_KEY0_DATA6_REG	BLOCK4 (KEY0) 的第 6 个寄存器内容	0x00B4	RO
EFUSE_RD_KEY0_DATA7_REG	BLOCK4 (KEY0) 的第 7 个寄存器内容	0x00B8	RO
EFUSE_RD_KEY1_DATA0_REG	BLOCK5 (KEY1) 的第 0 个寄存器内容	0x00BC	RO
EFUSE_RD_KEY1_DATA1_REG	BLOCK5 (KEY1) 的第 1 个寄存器内容	0x00C0	RO
EFUSE_RD_KEY1_DATA2_REG	BLOCK5 (KEY1) 的第 2 个寄存器内容	0x00C4	RO
EFUSE_RD_KEY1_DATA3_REG	BLOCK5 (KEY1) 的第 3 个寄存器内容	0x00C8	RO
EFUSE_RD_KEY1_DATA4_REG	BLOCK5 (KEY1) 的第 4 个寄存器内容	0x00CC	RO
EFUSE_RD_KEY1_DATA5_REG	BLOCK5 (KEY1) 的第 5 个寄存器内容	0x00D0	RO
EFUSE_RD_KEY1_DATA6_REG	BLOCK5 (KEY1) 的第 6 个寄存器内容	0x00D4	RO
EFUSE_RD_KEY1_DATA7_REG	BLOCK5 (KEY1) 的第 7 个寄存器内容	0x00D8	RO
EFUSE_RD_KEY2_DATA0_REG	BLOCK6 (KEY2) 的第 0 个寄存器内容	0x00DC	RO
EFUSE_RD_KEY2_DATA1_REG	BLOCK6 (KEY2) 的第 1 个寄存器内容	0x00E0	RO
EFUSE_RD_KEY2_DATA2_REG	BLOCK6 (KEY2) 的第 2 个寄存器内容	0x00E4	RO
EFUSE_RD_KEY2_DATA3_REG	BLOCK6 (KEY2) 的第 3 个寄存器内容	0x00E8	RO
EFUSE_RD_KEY2_DATA4_REG	BLOCK6 (KEY2) 的第 4 个寄存器内容	0x00EC	RO
EFUSE_RD_KEY2_DATA5_REG	BLOCK6 (KEY2) 的第 5 个寄存器内容	0x00F0	RO
EFUSE_RD_KEY2_DATA6_REG	BLOCK6 (KEY2) 的第 6 个寄存器内容	0x00F4	RO
EFUSE_RD_KEY2_DATA7_REG	BLOCK6 (KEY2) 的第 7 个寄存器内容	0x00F8	RO
EFUSE_RD_KEY3_DATA0_REG	BLOCK7 (KEY3) 的第 0 个寄存器内容	0x00FC	RO
EFUSE_RD_KEY3_DATA1_REG	BLOCK7 (KEY3) 的第 1 个寄存器内容	0x0100	RO
EFUSE_RD_KEY3_DATA2_REG	BLOCK7 (KEY3) 的第 2 个寄存器内容	0x0104	RO
EFUSE_RD_KEY3_DATA3_REG	BLOCK7 (KEY3) 的第 3 个寄存器内容	0x0108	RO
EFUSE_RD_KEY3_DATA4_REG	BLOCK7 (KEY3) 的第 4 个寄存器内容	0x010C	RO
EFUSE_RD_KEY3_DATA5_REG	BLOCK7 (KEY3) 的第 5 个寄存器内容	0x0110	RO
EFUSE_RD_KEY3_DATA6_REG	BLOCK7 (KEY3) 的第 6 个寄存器内容	0x0114	RO
EFUSE_RD_KEY3_DATA7_REG	BLOCK7 (KEY3) 的第 7 个寄存器内容	0x0118	RO
EFUSE_RD_KEY4_DATA0_REG	BLOCK8 (KEY4) 的第 0 个寄存器内容	0x011C	RO
EFUSE_RD_KEY4_DATA1_REG	BLOCK8 (KEY4) 的第 1 个寄存器内容	0x0120	RO
EFUSE_RD_KEY4_DATA2_REG	BLOCK8 (KEY4) 的第 2 个寄存器内容	0x0124	RO
EFUSE_RD_KEY4_DATA3_REG	BLOCK8 (KEY4) 的第 3 个寄存器内容	0x0128	RO
EFUSE_RD_KEY4_DATA4_REG	BLOCK8 (KEY4) 的第 4 个寄存器内容	0x012C	RO
EFUSE_RD_KEY4_DATA5_REG	BLOCK8 (KEY4) 的第 5 个寄存器内容	0x0130	RO
EFUSE_RD_KEY4_DATA6_REG	BLOCK8 (KEY4) 的第 6 个寄存器内容	0x0134	RO
EFUSE_RD_KEY4_DATA7_REG	BLOCK8 (KEY4) 的第 7 个寄存器内容	0x0138	RO

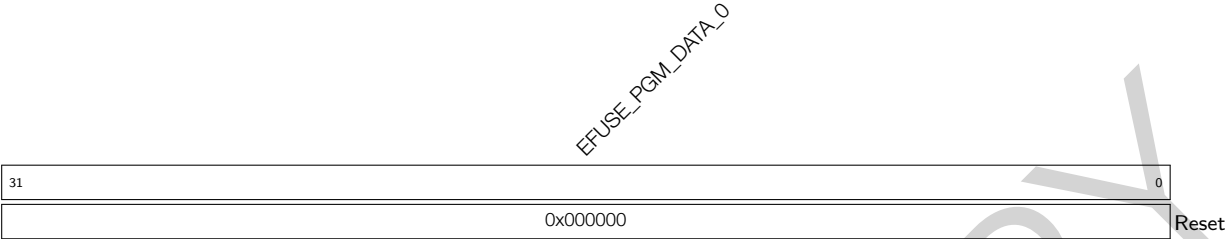
名称	描述	地址	访问
EFUSE_RD_KEY5_DATA0_REG	BLOCK9 (KEY5) 的第 0 个寄存器内容	0x013C	RO
EFUSE_RD_KEY5_DATA1_REG	BLOCK9 (KEY5) 的第 1 个寄存器内容	0x0140	RO
EFUSE_RD_KEY5_DATA2_REG	BLOCK9 (KEY5) 的第 2 个寄存器内容	0x0144	RO
EFUSE_RD_KEY5_DATA3_REG	BLOCK9 (KEY5) 的第 3 个寄存器内容	0x0148	RO
EFUSE_RD_KEY5_DATA4_REG	BLOCK9 (KEY5) 的第 4 个寄存器内容	0x014C	RO
EFUSE_RD_KEY5_DATA5_REG	BLOCK9 (KEY5) 的第 5 个寄存器内容	0x0150	RO
EFUSE_RD_KEY5_DATA6_REG	BLOCK9 (KEY5) 的第 6 个寄存器内容	0x0154	RO
EFUSE_RD_KEY5_DATA7_REG	BLOCK9 (KEY5) 的第 7 个寄存器内容	0x0158	RO
EFUSE_RD_SYS_PART2_DATA0_REG	BLOCK10 (system) 的第 0 个寄存器内容	0x015C	RO
EFUSE_RD_SYS_PART2_DATA1_REG	BLOCK10 (system) 的第 1 个寄存器内容	0x0160	RO
EFUSE_RD_SYS_PART2_DATA2_REG	BLOCK10 (system) 的第 2 个寄存器内容	0x0164	RO
EFUSE_RD_SYS_PART2_DATA3_REG	BLOCK10 (system) 的第 3 个寄存器内容	0x0168	RO
EFUSE_RD_SYS_PART2_DATA4_REG	BLOCK10 (system) 的第 4 个寄存器内容	0x016C	RO
EFUSE_RD_SYS_PART2_DATA5_REG	BLOCK10 (system) 的第 5 个寄存器内容	0x0170	RO
EFUSE_RD_SYS_PART2_DATA6_REG	BLOCK10 (system) 的第 6 个寄存器内容	0x0174	RO
EFUSE_RD_SYS_PART2_DATA7_REG	BLOCK10 (system) 的第 7 个寄存器内容	0x0178	RO
报告寄存器			
EFUSE_RD_REPEAT_ERR0_REG	BLOCK0 参数烧写错误记录第 0 个寄存器	0x017C	RO
EFUSE_RD_REPEAT_ERR1_REG	BLOCK0 参数烧写错误记录第 1 个寄存器	0x0180	RO
EFUSE_RD_REPEAT_ERR2_REG	BLOCK0 参数烧写错误记录第 2 个寄存器	0x0184	RO
EFUSE_RD_REPEAT_ERR3_REG	BLOCK0 参数烧写错误记录第 3 个寄存器	0x0188	RO
EFUSE_RD_REPEAT_ERR4_REG	BLOCK0 参数烧写错误记录第 4 个寄存器	0x0190	RO
EFUSE_RD_RS_ERR0_REG	记录 BLOCK1 ~ 10 参数烧写错误信息的第 0 个寄存器	0x01C0	RO
EFUSE_RD_RS_ERR1_REG	记录 BLOCK1 ~ 10 参数烧写错误信息的第 1 个寄存器	0x01C4	RO
配置寄存器			
EFUSE_CLK_REG	eFuse 时钟配置寄存器	0x01C8	R/W
EFUSE_CONF_REG	eFuse 运行模式配置寄存器	0x01CC	R/W
EFUSE_CMD_REG	eFuse 指令寄存器	0x01D4	varies
EFUSE_DAC_CONF_REG	eFuse 烧写电压控制寄存器	0x01E8	R/W
EFUSE_RD_TIM_CONF_REG	eFuse 读取时序参数配置寄存器	0x01EC	R/W
EFUSE_WR_TIM_CONF1_REG	eFuse 烧写时序参数第 1 个配置寄存器	0x01F4	R/W
EFUSE_WR_TIM_CONF2_REG	eFuse 烧写时序参数第 2 个配置寄存器	0x01F8	R/W
状态寄存器			
EFUSE_STATUS_REG	eFuse 状态寄存器	0x01D0	RO
中断寄存器			
EFUSE_INT_RAW_REG	eFuse 原始中断寄存器	0x01D8	R/ WC/ SS
EFUSE_INT_ST_REG	eFuse 中断状态寄存器	0x01DC	RO
EFUSE_INT_ENA_REG	eFuse 中断使能寄存器	0x01E0	R/W
EFUSE_INT_CLR_REG	eFuse 中断清除寄存器	0x01E4	WO

名称	描述	地址	访问
版本寄存器			
EFUSE_DATE_REG	版本控制寄存器	0x01FC	R/W

4.5 寄存器

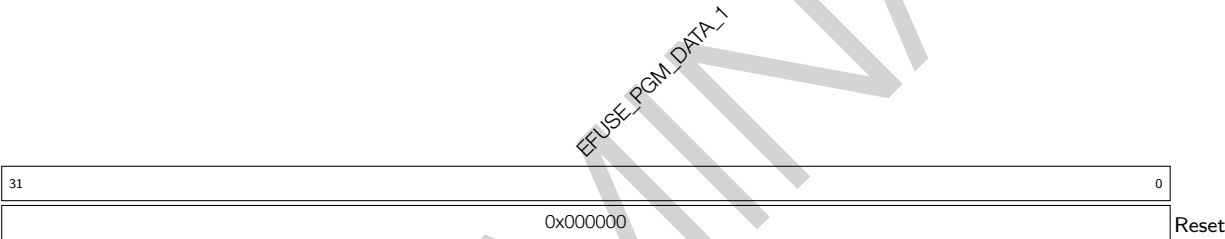
本小节的所有地址均为相对于 eFuse 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 4.1. EFUSE_PGM_DATA0_REG (0x0000)



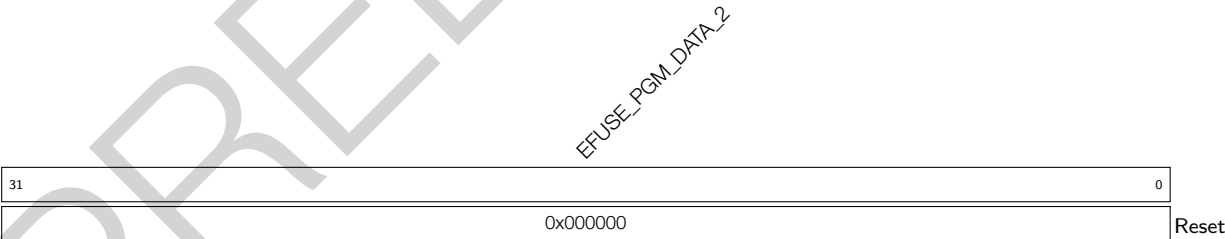
EFUSE_PGM_DATA_0 存放待烧写数据的第 0 个 32 位数据内容。(R/W)

Register 4.2. EFUSE_PGM_DATA1_REG (0x0004)



EFUSE_PGM_DATA_1 存放待烧写数据的第 1 个 32 位数据内容。(R/W)

Register 4.3. EFUSE_PGM_DATA2_REG (0x0008)



EFUSE_PGM_DATA_2 存放待烧写数据的第 2 个 32 位数据内容。(R/W)

Register 4.4. EFUSE_PGM_DATA3_REG (0x000C)

EFUSE_PGM_DATA_3	
31	0
0x000000	
Reset	

EFUSE_PGM_DATA_3 存放待烧写数据的第 3 个 32 位数据内容。(R/W)

Register 4.5. EFUSE_PGM_DATA4_REG (0x0010)

EFUSE_PGM_DATA_4	
31	0
0x000000	
Reset	

EFUSE_PGM_DATA_4 存放待烧写数据的第 4 个 32 位数据内容。(R/W)

Register 4.6. EFUSE_PGM_DATA5_REG (0x0014)

EFUSE_PGM_DATA_5	
31	0
0x000000	
Reset	

EFUSE_PGM_DATA_5 存放待烧写数据的第 5 个 32 位数据内容。(R/W)

Register 4.7. EFUSE_PGM_DATA6_REG (0x0018)

EFUSE_PGM_DATA_6	
31	0
0x000000	
Reset	

EFUSE_PGM_DATA_6 存放待烧写数据的第 6 个 32 位数据内容。(R/W)

Register 4.8. EFUSE_PGM_DATA7_REG (0x001C)

EFUSE_PGM_DATA_7	
31	0
0x000000	
Reset	

EFUSE_PGM_DATA_7 存放待烧写数据的第 7 个 32 位数据内容。(R/W)

Register 4.9. EFUSE_PGM_CHECK_VALUE0_REG (0x0020)

EFUSE_PGM_RS_DATA_0	
31	0
0x000000	
Reset	

EFUSE_PGM_RS_DATA_0 存放待烧写 RS 代码的第 0 个 32 位数据内容。(R/W)

Register 4.10. EFUSE_PGM_CHECK_VALUE1_REG (0x0024)

EFUSE_PGM_RS_DATA_1	
31	0
0x000000	
Reset	

EFUSE_PGM_RS_DATA_1 存放待烧写 RS 代码的第 1 个 32 位数据内容。(R/W)

Register 4.11. EFUSE_PGM_CHECK_VALUE2_REG (0x0028)

EFUSE_PGM_RS_DATA_2	
31	0
0x000000	
Reset	

EFUSE_PGM_RS_DATA_2 存放待烧写 RS 代码的第 2 个 32 位数据内容。(R/W)

Register 4.12. EFUSE_RD_WR_DIS_REG (0x002C)

EFUSE_WR_DIS	
31	0
0x000000	
Reset	

EFUSE_WR_DIS 置位禁用 eFuse 烧写。(RO)

Register 4.13. EFUSE_RD_REPEAT_DATA0_REG (0x0030)

(reserved)					EFUSE_VDD_SPI_AS_GPIO EFUSE_USB_EXCHG_PINS					(reserved)					EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT EFUSE_DIS_PAD_JTAG					EFUSE_SOFT_DIS_JTAG EFUSE_JTAG_SEL_ENABLE EFUSE_DIS_TWAI EFUSE_RPT4_RESERVED6 EFUSE_DIS_FORCE_DOWNLOAD EFUSE_DIS_USB_SERIAL_JTAG EFUSE_DIS_DOWNLOAD_ICACHE EFUSE_DIS_USB_JTAG EFUSE_DIS_ICACHE EFUSE_DIS_RTC_RAM_BOOT					EFUSE_RD_DIS				
31			27	26	25	24		21	20	19	18		16	15	14	13	12	11	10	9	8	7	6					0	
0	0	0	0	0	0	0	0	0	0	0	0		0x0	0	0	0	0	0	0	0	0	0	0			0x0		Reset	

EFUSE_RD_DIS 置位禁止软件读取 eFuse Block4 ~ 10 的内容。(RO)

EFUSE_DIS_RTC_RAM_BOOT 保留 (采用 4 备份编码)。(RO)

EFUSE_DIS_ICACHE 置位禁用 lcache。(RO)

EFUSE_DIS_USB_JTAG 置位禁用 usb_serial_jtag 模块 USB 转 JTAG 的功能。(RO)

EFUSE_DIS_DOWNLOAD_ICACHE 置位在下载模式下关闭 lcache (boot_mode[3:0] 为 0, 1, 2, 3, 6, 7)。(RO)

EFUSE_DIS_USB_SERIAL_JTAG 置位禁用 usb_serial_jtag 模块。(RO)

EFUSE_DIS_FORCE_DOWNLOAD 置位禁止强制芯片进入下载模式。(RO)

EFUSE_RPT4_RESERVED6 保留 (采用 4 备份编码)。(RO)

EFUSE_DIS_TWAI 置位禁用 TWAI 功能。(RO)

EFUSE_JTAG_SEL_ENABLE 置位直接使用 JTAG 功能。(RO)

EFUSE_SOFT_DIS_JTAG 软关断 JTAG 功能 (奇数个比特值为 1 表示关断), 用户还可以通过 HAMC 模块再次打开 JTAG。(RO)

EFUSE_DIS_PAD_JTAG 硬关断 JTAG 功能, 永久关断。(RO)

EFUSE_DIS_DOWNLOAD_MANUAL_ENCRYPT 置位在 download boot 模式下关闭 flash 加密功能 (SPI 启动时除外)。(RO)

EFUSE_USB_EXCHG_PINS 置位交换 USB D+ 和 D- 管脚。(RO)

EFUSE_VDD_SPI_AS_GPIO 置位表明 VDD SPI 作为普通引脚。(RO)

Register 4.14. EFUSE_RD_REPEAT_DATA1_REG (0x0034)

EFUSE_KEY_PURPOSE_1		EFUSE_KEY_PURPOSE_0		EFUSE_SECURE_BOOT_KEY_REVOKE2		EFUSE_SECURE_BOOT_KEY_REVOKE1		EFUSE_SECURE_BOOT_KEY_REVOKE0		EFUSE_SPI_BOOT_CRYPT_CNT		EFUSE_WDT_DELAY_SEL		EFUSE_RPT4_RESERVED2		
31	28	27	24	23	22	21	20	18	17	16	15				0	
0x0		0x0		0	0	0	0x0	0x0	0x00							Reset

EFUSE_RPT4_RESERVED2 保留（采用 4 备份编码）。(RO)

EFUSE_WDT_DELAY_SEL 选择 RTC 看门狗超时阈值，单位为慢速时钟周期。00: 40,000 个慢速时钟周期；01: 80,000 个慢速时钟周期；10: 160,000 个慢速时钟周期；11: 320,000 个慢速时钟周期。(RO)

EFUSE_SPI_BOOT_CRYPT_CNT 置位使能 SPI boot 加解密。奇数个 1：使能；偶数个 1：禁用。(RO)

EFUSE_SECURE_BOOT_KEY_REVOKE0 置位使能撤销第一个安全启动密钥。(RO)

EFUSE_SECURE_BOOT_KEY_REVOKE1 置位使能撤销第二个安全启动密钥。(RO)

EFUSE_SECURE_BOOT_KEY_REVOKE2 置位使能撤销第三个安全启动密钥。(RO)

EFUSE_KEY_PURPOSE_0 Key0 用途。(RO)

EFUSE_KEY_PURPOSE_1 Key1 用途。(RO)

Register 4.15. EFUSE_RD_REPEAT_DATA2_REG (0x0038)

EFUSE_FLASH_TPUW				EFUSE_RPT4_RESERVED0				EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE				EFUSE_SECURE_BOOT_EN				EFUSE_RPT4_RESERVED3				EFUSE_KEY_PURPOSE_5				EFUSE_KEY_PURPOSE_4				EFUSE_KEY_PURPOSE_3				EFUSE_KEY_PURPOSE_2			
31	28	27	22	21	20	19	16	15	12	11	8	7	4	3	0	Reset																			
0x0				0x0				0	0	0x0				0x0				0x0				0x0													

EFUSE_KEY_PURPOSE_2 Key2 用途。(RO)

EFUSE_KEY_PURPOSE_3 Key3 用途。(RO)

EFUSE_KEY_PURPOSE_4 Key4 用途。(RO)

EFUSE_KEY_PURPOSE_5 Key5 用途。(RO)

EFUSE_RPT4_RESERVED3 保留（采用 4 备份编码）。(RO)

EFUSE_SECURE_BOOT_EN 置位使能安全启动。(RO)

EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE 置位使能密钥失效的激进策略。(RO)

EFUSE_RPT4_RESERVED0 保留（采用 4 备份编码）。(RO)

EFUSE_FLASH_TPUW 配置上电后 flash 等待时间，单位为 ms。该值小于 15 时，等待时间为配置的值；该值大于等于 15 时，等待时间为配置的时间的 2 倍。(RO)

Register 4.16. EFUSE_RD_REPEAT_DATA3_REG (0x003C)

EFUSE_ERR_RST_ENABLE EFUSE_RPT4_RESERVED1			EFUSE_SECURE_VERSION										EFUSE_FORCE_SEND_RESUME EFUSE_RPT4_RESERVED5					EFUSE_UART_PRINT_CONTROL EFUSE_ENABLE_SECURITY_DOWNLOAD EFUSE_DIS_USB_DOWNLOAD_MODE EFUSE_RPT4_RESERVED7 EFUSE_USB_PRINT_CHANNEL EFUSE_RPT4_RESERVED8 EFUSE_DIS_DOWNLOAD_MODE				
31	30	29											8	7	6	5	4	3	2	1	0	Reset
0	0		0x00										0		0	0x0	0	0	0	0	0	

EFUSE_DIS_DOWNLOAD_MODE 置位关闭下载模式 (boot_mode[3:0] = 0, 1, 2, 3, 6, 7)。 (RO)

EFUSE_RPT4_RESERVED8 保留 (采用 4 备份编码)。 (RO)

EFUSE_USB_PRINT_CHANNEL 置位关闭 USB 打印。 (RO)

EFUSE_RPT4_RESERVED7 保留 (采用 4 备份编码)。 (RO)

EFUSE_DIS_USB_DOWNLOAD_MODE 置位关闭 USB 下载。 (RO)

EFUSE_ENABLE_SECURITY_DOWNLOAD 置位使能安全下载模式。 (RO)

EFUSE_UART_PRINT_CONTROL 设置 UART 打印的类型。00: 强制使能打印; 01: GPIO8 低电平复位时, 使能打印; 10GPIO8 高电平复位时, 使能打印; 11: 强制关闭打印。 (RO)

EFUSE_RPT4_RESERVED5 保留 (采用 4 备份编码)。 (RO)

EFUSE_FORCE_SEND_RESUME 置位强制 ROM 代码在 SPI 启动过程中发送恢复指令。 (RO)

EFUSE_SECURE_VERSION 表明 IDF 安全版本 (用于 ESP-IDF 的防回滚功能)。 (RO)

EFUSE_RPT4_RESERVED1 保留 (采用 4 备份编码)。 (RO)

EFUSE_ERR_RST_ENABLE 置位表明使能 BLOCK0 的错误寄存器检查。 (RO)

Register 4.17. EFUSE_RD_REPEAT_DATA4_REG (0x0040)

(reserved)								EFUSE_RPT4_RESERVED4														
31								24	23													0
0	0	0	0	0	0	0	0			0x0000												Reset

EFUSE_RPT4_RESERVED4 保留 (采用 4 备份编码)。 (RO)

Register 4.18. EFUSE_RD_MAC_SPI_SYS_0_REG (0x0044)

EFUSE_MAC_0	
31	0
0x000000	
Reset	

EFUSE_MAC_0 存储 MAC 地址低 32 位的内容。(RO)

Register 4.19. EFUSE_RD_MAC_SPI_SYS_1_REG (0x0048)

EFUSE_SPI_PAD_CONF_0		EFUSE_MAC_1	
31	16	15	0
0x00		0x00	
Reset		Reset	

EFUSE_MAC_1 存储 MAC 地址高 16 位的内容。(RO)

EFUSE_SPI_PAD_CONF_0 存储 SPI_PAD_CONF 第 0 部分的内容。(RO)

Register 4.20. EFUSE_RD_MAC_SPI_SYS_2_REG (0x004C)

EFUSE_SPI_PAD_CONF_1	
31	0
0x000000	
Reset	

EFUSE_SPI_PAD_CONF_1 存储 SPI_PAD_CONF 第 1 部分的内容。(RO)

Register 4.21. EFUSE_RD_MAC_SPI_SYS_3_REG (0x0050)

EFUSE_SYS_DATA_PART0_0																		EFUSE_SPI_PAD_CONF_2														
31																		18	17											0		
0x00																		0x000														Reset

EFUSE_SPI_PAD_CONF_2 存储 SPI_PAD_CONF 第 2 部分的内容。(RO)

EFUSE_SYS_DATA_PART0_0 存储系统数据第 0 部分的第 1 个 14 位内容。(RO)

Register 4.22. EFUSE_RD_MAC_SPI_SYS_4_REG (0x0054)

EFUSE_SYS_DATA_PART0_1																															
31																															0
0x000000																															
Reset																															

EFUSE_SYS_DATA_PART0_1 存储系统数据第 0 部分的第 1 个 32 位内容。(RO)

Register 4.23. EFUSE_RD_MAC_SPI_SYS_5_REG (0x0058)

EFUSE_SYS_DATA_PART0_2																															
31																															0
0x000000																															
Reset																															

EFUSE_SYS_DATA_PART0_2 存储系统数据第 0 部分的第 2 个 32 位内容。(RO)

Register 4.24. EFUSE_RD_SYS_PART1_DATA0_REG (0x005C)

EFUSE_SYS_DATA_PART1_0	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_0 存储系统数据第 1 部分的第 0 个 32 位的内容。(RO)

Register 4.25. EFUSE_RD_SYS_PART1_DATA1_REG (0x0060)

EFUSE_SYS_DATA_PART1_1	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_1 存储系统数据第 1 部分的第 1 个 32 位内容。(RO)

Register 4.26. EFUSE_RD_SYS_PART1_DATA2_REG (0x0064)

EFUSE_SYS_DATA_PART1_2	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_2 存储系统数据第 1 部分的第 2 个 32 位内容。(RO)

Register 4.27. EFUSE_RD_SYS_PART1_DATA3_REG (0x0068)

EFUSE_SYS_DATA_PART1_3	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_3 存储系统数据第 1 部分的第 3 个 32 位内容。(RO)

Register 4.28. EFUSE_RD_SYS_PART1_DATA4_REG (0x006C)

EFUSE_SYS_DATA_PART1_4	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_4 存储系统数据第 1 部分的第 4 个 32 位内容。(RO)

Register 4.29. EFUSE_RD_SYS_PART1_DATA5_REG (0x0070)

EFUSE_SYS_DATA_PART1_5	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_5 存储系统数据第 1 部分的第 5 个 32 位内容。(RO)

Register 4.30. EFUSE_RD_SYS_PART1_DATA6_REG (0x0074)

EFUSE_SYS_DATA_PART1_6	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_6 存储系统数据第 1 部分的第 6 个 32 位内容。(RO)

Register 4.31. EFUSE_RD_SYS_PART1_DATA7_REG (0x0078)

EFUSE_SYS_DATA_PART1_7	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART1_7 存储系统数据第 1 部分的第 7 个 32 位内容。(RO)

Register 4.32. EFUSE_RD_USR_DATA0_REG (0x007C)

EFUSE_USR_DATA0	
31	0
0x000000	
Reset	

EFUSE_USR_DATA0 存储 BLOCK3 (user) 第 0 个 32 位内容。(RO)

Register 4.33. EFUSE_RD_USR_DATA1_REG (0x0080)

EFUSE_USR_DATA1	
31	0
0x000000	
Reset	

EFUSE_USR_DATA1 存储 BLOCK3 (user) 第 1 个 32 位内容。(RO)

Register 4.34. EFUSE_RD_USR_DATA2_REG (0x0084)

EFUSE_USR_DATA2	
31	0
0x000000	
Reset	

EFUSE_USR_DATA2 存储 BLOCK3 (user) 第 2 个 32 位内容。(RO)

Register 4.35. EFUSE_RD_USR_DATA3_REG (0x0088)

EFUSE_USR_DATA3	
31	0
0x000000	
Reset	

EFUSE_USR_DATA3 存储 BLOCK3 (user) 第 3 个 32 位内容。(RO)

Register 4.36. EFUSE_RD_USR_DATA4_REG (0x008C)

EFUSE_USR_DATA4	
31	0
0x000000	
Reset	

EFUSE_USR_DATA4 存储 BLOCK3 (user) 第 4 个 32 位内容。(RO)

Register 4.37. EFUSE_RD_USR_DATA5_REG (0x0090)

EFUSE_USR_DATA5	
31	0
0x000000	
Reset	

EFUSE_USR_DATA5 存储 BLOCK3 (user) 第 5 个 32 位内容。(RO)

Register 4.38. EFUSE_RD_USR_DATA6_REG (0x0094)

EFUSE_USR_DATA6	
31	0
0x000000	
Reset	

EFUSE_USR_DATA6 存储 BLOCK3 (user) 第 6 个 32 位内容。(RO)

Register 4.39. EFUSE_RD_USR_DATA7_REG (0x0098)

EFUSE_USR_DATA7	
31	0
0x000000	
Reset	

EFUSE_USR_DATA7 存储 BLOCK3 (user) 第 7 个 32 位内容。(RO)

Register 4.40. EFUSE_RD_KEY0_DATA0_REG (0x009C)

EFUSE_KEY0_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA0 存储 KEY0 第 0 个 32 位内容。(RO)

Register 4.41. EFUSE_RD_KEY0_DATA1_REG (0x00A0)

EFUSE_KEY0_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA1 存储 KEY0 第 1 个 32 位内容。(RO)

Register 4.42. EFUSE_RD_KEY0_DATA2_REG (0x00A4)

EFUSE_KEY0_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA2 存储 KEY0 第 2 个 32 位内容。(RO)

Register 4.43. EFUSE_RD_KEY0_DATA3_REG (0x00A8)

EFUSE_KEY0_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA3 存储 KEY0 第 3 个 32 位内容。(RO)

Register 4.44. EFUSE_RD_KEY0_DATA4_REG (0x00AC)

EFUSE_KEY0_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA4 存储 KEY0 第 4 个 32 位内容。(RO)

Register 4.45. EFUSE_RD_KEY0_DATA5_REG (0x00B0)

EFUSE_KEY0_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA5 存储 KEY0 第 5 个 32 位内容。(RO)

Register 4.46. EFUSE_RD_KEY0_DATA6_REG (0x00B4)

EFUSE_KEY0_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA6 存储 KEY0 第 6 个 32 位内容。(RO)

Register 4.47. EFUSE_RD_KEY0_DATA7_REG (0x00B8)

EFUSE_KEY0_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY0_DATA7 存储 KEY0 第 7 个 32 位内容。(RO)

Register 4.48. EFUSE_RD_KEY1_DATA0_REG (0x00BC)

EFUSE_KEY1_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA0 存储 KEY1 第 0 个 32 位内容。(RO)

Register 4.49. EFUSE_RD_KEY1_DATA1_REG (0x00C0)

EFUSE_KEY1_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA1 存储 KEY1 第 1 个 32 位内容。(RO)

Register 4.50. EFUSE_RD_KEY1_DATA2_REG (0x00C4)

EFUSE_KEY1_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA2 存储 KEY1 第 2 个 32 位内容。(RO)

Register 4.51. EFUSE_RD_KEY1_DATA3_REG (0x00C8)

EFUSE_KEY1_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA3 存储 KEY1 第 3 个 32 位内容。(RO)

Register 4.52. EFUSE_RD_KEY1_DATA4_REG (0x00CC)

EFUSE_KEY1_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA4 存储 KEY1 第 4 个 32 位内容。(RO)

Register 4.53. EFUSE_RD_KEY1_DATA5_REG (0x00D0)

EFUSE_KEY1_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA5 存储 KEY1 第 5 个 32 位内容。(RO)

Register 4.54. EFUSE_RD_KEY1_DATA6_REG (0x00D4)

EFUSE_KEY1_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA6 存储 KEY1 第 6 个 32 位内容。(RO)

Register 4.55. EFUSE_RD_KEY1_DATA7_REG (0x00D8)

EFUSE_KEY1_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY1_DATA7 存储 KEY1 第 7 个 32 位内容。(RO)

Register 4.56. EFUSE_RD_KEY2_DATA0_REG (0x00DC)

EFUSE_KEY2_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA0 存储 KEY2 第 0 个 32 位内容。(RO)

Register 4.57. EFUSE_RD_KEY2_DATA1_REG (0x00E0)

EFUSE_KEY2_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA1 存储 KEY2 第 1 个 32 位内容。(RO)

Register 4.58. EFUSE_RD_KEY2_DATA2_REG (0x00E4)

EFUSE_KEY2_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA2 存储 KEY2 第 2 个 32 位内容。(RO)

Register 4.59. EFUSE_RD_KEY2_DATA3_REG (0x00E8)

EFUSE_KEY2_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA3 存储 KEY2 第 3 个 32 位内容。(RO)

Register 4.60. EFUSE_RD_KEY2_DATA4_REG (0x00EC)

EFUSE_KEY2_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA4 存储 KEY2 第 4 个 32 位内容。(RO)

Register 4.61. EFUSE_RD_KEY2_DATA5_REG (0x00F0)

EFUSE_KEY2_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA5 存储 KEY2 第 5 个 32 位内容。(RO)

Register 4.62. EFUSE_RD_KEY2_DATA6_REG (0x00F4)

EFUSE_KEY2_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA6 存储 KEY2 第 6 个 32 位内容。(RO)

Register 4.63. EFUSE_RD_KEY2_DATA7_REG (0x00F8)

EFUSE_KEY2_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY2_DATA7 存储 KEY2 第 7 个 32 位内容。(RO)

Register 4.64. EFUSE_RD_KEY3_DATA0_REG (0x00FC)

EFUSE_KEY3_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA0 存储 KEY3 第 0 个 32 位内容。(RO)

Register 4.65. EFUSE_RD_KEY3_DATA1_REG (0x0100)

EFUSE_KEY3_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA1 存储 KEY3 第 1 个 32 位内容。(RO)

Register 4.66. EFUSE_RD_KEY3_DATA2_REG (0x0104)

EFUSE_KEY3_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA2 存储 KEY3 第 2 个 32 位内容。(RO)

Register 4.67. EFUSE_RD_KEY3_DATA3_REG (0x0108)

EFUSE_KEY3_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA3 存储 KEY3 第 3 个 32 位内容。(RO)

Register 4.68. EFUSE_RD_KEY3_DATA4_REG (0x010C)

EFUSE_KEY3_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA4 存储 KEY3 第 4 个 32 位内容。(RO)

Register 4.69. EFUSE_RD_KEY3_DATA5_REG (0x0110)

EFUSE_KEY3_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA5 存储 KEY3 第 5 个 32 位内容。(RO)

Register 4.70. EFUSE_RD_KEY3_DATA6_REG (0x0114)

EFUSE_KEY3_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA6 存储 KEY3 第 6 个 32 位内容。(RO)

Register 4.71. EFUSE_RD_KEY3_DATA7_REG (0x0118)

EFUSE_KEY3_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY3_DATA7 存储 KEY3 第 7 个 32 位内容。(RO)

Register 4.72. EFUSE_RD_KEY4_DATA0_REG (0x011C)

EFUSE_KEY4_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA0 存储 KEY4 第 0 个 32 位内容。(RO)

Register 4.73. EFUSE_RD_KEY4_DATA1_REG (0x0120)

EFUSE_KEY4_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA1 存储 KEY4 第 1 个 32 位内容。(RO)

Register 4.74. EFUSE_RD_KEY4_DATA2_REG (0x0124)

EFUSE_KEY4_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA2 存储 KEY4 第 2 个 32 位内容。(RO)

Register 4.75. EFUSE_RD_KEY4_DATA3_REG (0x0128)

EFUSE_KEY4_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA3 存储 KEY4 第 3 个 32 位内容。(RO)

Register 4.76. EFUSE_RD_KEY4_DATA4_REG (0x012C)

EFUSE_KEY4_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA4 存储 KEY4 第 4 个 32 位内容。(RO)

Register 4.77. EFUSE_RD_KEY4_DATA5_REG (0x0130)

EFUSE_KEY4_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA5 存储 KEY4 第 5 个 32 位内容。(RO)

Register 4.78. EFUSE_RD_KEY4_DATA6_REG (0x0134)

EFUSE_KEY4_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA6 存储 KEY4 第 6 个 32 位内容。(RO)

Register 4.79. EFUSE_RD_KEY4_DATA7_REG (0x0138)

EFUSE_KEY4_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY4_DATA7 存储 KEY4 第 7 个 32 位内容。(RO)

Register 4.80. EFUSE_RD_KEY5_DATA0_REG (0x013C)

EFUSE_KEY5_DATA0	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA0 存储 KEY5 第 0 个 32 位内容。(RO)

Register 4.81. EFUSE_RD_KEY5_DATA1_REG (0x0140)

EFUSE_KEY5_DATA1	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA1 存储 KEY5 第 1 个 32 位内容。(RO)

Register 4.82. EFUSE_RD_KEY5_DATA2_REG (0x0144)

EFUSE_KEY5_DATA2	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA2 存储 KEY5 第 2 个 32 位内容。(RO)

Register 4.83. EFUSE_RD_KEY5_DATA3_REG (0x0148)

EFUSE_KEY5_DATA3	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA3 存储 KEY5 第 3 个 32 位内容。(RO)

Register 4.84. EFUSE_RD_KEY5_DATA4_REG (0x014C)

EFUSE_KEY5_DATA4	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA4 存储 KEY5 第 4 个 32 位内容。(RO)

Register 4.85. EFUSE_RD_KEY5_DATA5_REG (0x0150)

EFUSE_KEY5_DATA5	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA5 存储 KEY5 第 5 个 32 位内容。(RO)

Register 4.86. EFUSE_RD_KEY5_DATA6_REG (0x0154)

EFUSE_KEY5_DATA6	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA6 存储 KEY5 第 6 个 32 位内容。(RO)

Register 4.87. EFUSE_RD_KEY5_DATA7_REG (0x0158)

EFUSE_KEY5_DATA7	
31	0
0x000000	
Reset	

EFUSE_KEY5_DATA7 存储 KEY5 第 7 个 32 位内容。(RO)

Register 4.88. EFUSE_RD_SYS_PART2_DATA0_REG (0x015C)

EFUSE_SYS_DATA_PART2_0	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_0 存储系统数据第 2 部分的第 0 个 32 位内容。(RO)

Register 4.89. EFUSE_RD_SYS_PART2_DATA1_REG (0x0160)

EFUSE_SYS_DATA_PART2_1	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_1 存储系统数据第 2 部分的第 1 个 32 位内容。(RO)

Register 4.90. EFUSE_RD_SYS_PART2_DATA2_REG (0x0164)

EFUSE_SYS_DATA_PART2_2	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_2 存储系统数据第 2 部分的第 2 个 32 位内容。(RO)

Register 4.91. EFUSE_RD_SYS_PART2_DATA3_REG (0x0168)

EFUSE_SYS_DATA_PART2_3	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_3 存储系统数据第 2 部分的第 3 个 32 位内容。(RO)

Register 4.92. EFUSE_RD_SYS_PART2_DATA4_REG (0x016C)

EFUSE_SYS_DATA_PART2_4	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_4 存储系统数据第 2 部分的第 4 个 32 位内容。(RO)

Register 4.93. EFUSE_RD_SYS_PART2_DATA5_REG (0x0170)

EFUSE_SYS_DATA_PART2_5	
31	0
0x000000	
Reset	

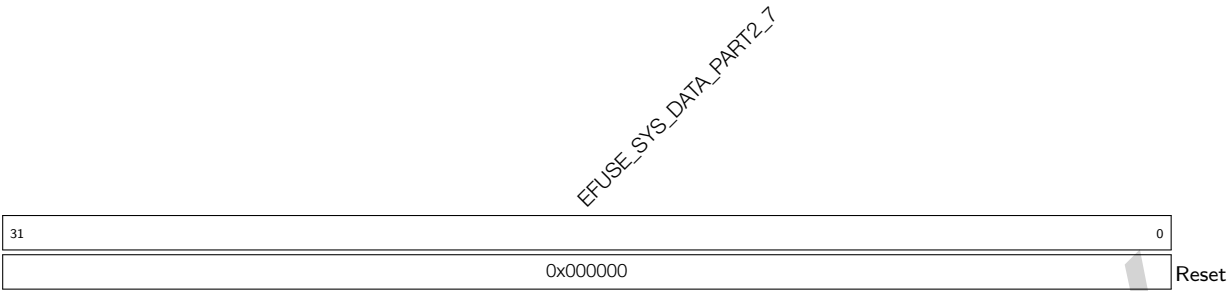
EFUSE_SYS_DATA_PART2_5 存储系统数据第 2 部分的第 5 个 32 位内容。(RO)

Register 4.94. EFUSE_RD_SYS_PART2_DATA6_REG (0x0174)

EFUSE_SYS_DATA_PART2_6	
31	0
0x000000	
Reset	

EFUSE_SYS_DATA_PART2_6 存储系统数据第 2 部分的第 6 个 32 位内容。(RO)

Register 4.95. EFUSE_RD_SYS_PART2_DATA7_REG (0x0178)



EFUSE_SYS_DATA_PART2_7 存储系统数据第 2 部分的第 7 个 32 位内容。(RO)

125

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

EFUSE_KEY_PURPOSE_1_ERR 若该参数中任意比特为 1，表明出现烧写错误。(RO)

Register 4.98. EFUSE_RD_REPEAT_ERR2_REG (0x0184)

EFUSE_FLASH_TPUW_ERR				EFUSE_RPT4_RESERVED0_ERR				EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE_ERR				EFUSE_SECURE_BOOT_EN_ERR				EFUSE_RPT4_RESERVED3_ERR				EFUSE_KEY_PURPOSE_5_ERR				EFUSE_KEY_PURPOSE_4_ERR				EFUSE_KEY_PURPOSE_3_ERR				EFUSE_KEY_PURPOSE_2_ERR					
31	28	27		22	21	20	19	16	15	12	11		8	7		4	3		0	Reset																	
0x0				0x0				0	0	0x0				0x0				0x0				0x0				0x0											

- EFUSE_KEY_PURPOSE_2_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_KEY_PURPOSE_3_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_KEY_PURPOSE_4_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_KEY_PURPOSE_5_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED3_ERR** 保留。(RO)
- EFUSE_SECURE_BOOT_EN_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_SECURE_BOOT_AGGRESSIVE_REVOKE_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED0_ERR** 保留。(RO)
- EFUSE_FLASH_TPUW_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)

Register 4.99. EFUSE_RD_REPEAT_ERR3_REG (0x0188)

EFUSE_ERR_RST_ENABLE_ERR			EFUSE_RPT4_RESERVED1_ERR			EFUSE_SECURE_VERSION_ERR			EFUSE_FORCE_SEND_RESUME_ERR			EFUSE_RPT4_RESERVED5_ERR			EFUSE_UART_PRINT_CONTROL_ERR			EFUSE_ENABLE_SECURITY_DOWNLOAD_ERR			EFUSE_DIS_USB_DOWNLOAD_MODE_ERR			EFUSE_RPT4_RESERVED7_ERR			EFUSE_USB_PRINT_CHANNEL_ERR			EFUSE_RPT4_RESERVED8_ERR			EFUSE_DIS_DOWNLOAD_MODE_ERR									
31	30	29												14	13	12												8	7	6	5	4	3	2	1	0						
0	0	0x00											0	0			0x0			0	0	0	0	0	0	0	0	0	0	0	0	0										
																												Reset														

- EFUSE_DIS_DOWNLOAD_MODE_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED8_ERR** 保留。(RO)
- EFUSE_USB_PRINT_CHANNEL_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED7_ERR** 保留。(RO)
- EFUSE_DIS_USB_DOWNLOAD_MODE_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_ENABLE_SECURITY_DOWNLOAD_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_UART_PRINT_CONTROL_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED5_ERR** 保留。(RO)
- EFUSE_FORCE_SEND_RESUME_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_SECURE_VERSION_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)
- EFUSE_RPT4_RESERVED1_ERR** 保留。(RO)
- EFUSE_ERR_RST_ENABLE_ERR** 若该参数中任意比特为 1，表明出现烧写错误。(RO)

Register 4.100. EFUSE_RD_REPEAT_ERR4_REG (0x0190)

(reserved)																EFUSE_RPT4_RESERVED4_ERR																		
31									24									23																0
0 0 0 0 0 0 0 0									0x0000																Reset									

- EFUSE_RPT4_RESERVED4_ERR** 保留。(RO)

Register 4.101. EFUSE_RD_RS_ERR0_REG (0x01C0)

EFUSE_KEY3_FAIL		EFUSE_KEY4_ERR_NUM		EFUSE_KEY2_FAIL		EFUSE_KEY3_ERR_NUM		EFUSE_KEY1_FAIL		EFUSE_KEY2_ERR_NUM		EFUSE_KEY0_FAIL		EFUSE_KEY1_ERR_NUM		EFUSE_USR_DATA_FAIL		EFUSE_KEY0_ERR_NUM		EFUSE_SYS_PART1_FAIL		EFUSE_USR_DATA_ERR_NUM		EFUSE_MAC_SPI_8M_FAIL		EFUSE_SYS_PART1_NUM		(reserved)		EFUSE_MAC_SPI_8M_ERR_NUM	
31	30	28	27	26	24	23	22	20	19	18	16	15	14	12	11	10	8	7	6	4	3	2	0	Reset							
0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0	0	0x0								

EFUSE_MAC_SPI_8M_ERR_NUM 指示用户数据的错误字节个数。(只读)

EFUSE_SYS_PART1_NUM 指示第 1 部分系统数据的错误字节个数。(只读)

EFUSE_MAC_SPI_8M_FAIL 0: 无烧写错误, MAC_SPI_8M 的数据是可靠的; 1: 烧写数据失败, 错误字节数超过 6。(只读)

EFUSE_USR_DATA_ERR_NUM 指示用户数据的错误字节个数。(只读)

EFUSE_SYS_PART1_FAIL 0: 无烧写错误, 第 1 部分的系统数据是可靠的; 1: 烧写数据失败, 错误字节数超过 6。(只读)

EFUSE_KEY0_ERR_NUM 指示 KEY0 的错误字节个数。(只读)

EFUSE_USR_DATA_FAIL 0: 无烧写错误, 用户数据是可靠的; 1: 烧写数据失败, 错误字节数超过 6。(只读)

EFUSE_KEY1_ERR_NUM 指示 KEY1 的错误字节个数。(只读)

EFUSE_KEY0_FAIL 0: 无烧写错误, KEY0 数据是可靠的; 1: KEY0 烧写失败, 错误字节数超过 6。(只读)

EFUSE_KEY2_ERR_NUM 指示 KEY2 的错误字节个数。(只读)

EFUSE_KEY1_FAIL 0: 无烧写错误, KEY1 数据是可靠的; 1: KEY1 烧写失败, 错误字节数超过 6。(只读)

EFUSE_KEY3_ERR_NUM 指示 KEY3 的错误字节个数。(只读)

EFUSE_KEY2_FAIL 0: 无烧写错误, KEY2 数据是可靠的; 1: KEY2 烧写失败, 错误字节数超过 6。(只读)

EFUSE_KEY4_ERR_NUM 指示 KEY4 的错误字节个数。(只读)

EFUSE_KEY3_FAIL 0: 无烧写错误, KEY3 数据是可靠的; 1: KEY3 烧写失败, 错误字节数超过 6。(只读)

Register 4.102. EFUSE_RD_RS_ERR1_REG (0x01C4)

(reserved)																								EFUSE_KEY5_FAIL		EFUSE_SYS_PART2_ERR_NUM		EFUSE_KEY4_FAIL		EFUSE_KEY5_ERR_NUM		
31																								8	7	6	4		3	2	0	
0 0																								0	0x0		0	0x0		Reset		

Reset

EFUSE_KEY5_ERR_NUM 指示 KEY5 的错误字节个数。(只读)

EFUSE_KEY5_FAIL 0: 无烧写错误, KEY5 数据是可靠的; 1: KEY4 数据烧写失败, 错误字节数超过 6。(只读)

EFUSE_SYS_PART2_ERR_NUM 指示第 2 部分系统数据的错误字节个数。(只读)

EFUSE_KEY5_FAIL 0: 无烧写错误, KEY5 数据是可靠的; 1: 数据烧写失败, 错误字节数超过 6。(只读)

Register 4.103. EFUSE_CLK_REG (0x01C8)

(reserved)																EFUSE_CLK_EN		(reserved)														EFUSE_EFUSE_MEM_FORCE_PU		EFUSE_MEM_CLK_FORCE_ON		EFUSE_EFUSE_MEM_FORCE_PD	
31																17	16	15															3	2	1	0	Reset
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0														0	1	0				

Reset

EFUSE_EFUSE_MEM_FORCE_PD 置位强制 eFuse SRAM 进入省电模式。(R/W)

EFUSE_MEM_CLK_FORCE_ON 置位强制激活 eFuse SRAM 的时钟信号。(R/W)

EFUSE_EFUSE_MEM_FORCE_PU 置位强制 eFuse SRAM 进入工作模式。(R/W)

EFUSE_CLK_EN 置位强制使能 eFuse 存储器的时钟信号。(R/W)

Register 4.104. EFUSE_CONF_REG (0x01CC)

(reserved)																EFUSE_OP_CODE																	
31																16	15																0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x00																	Reset

EFUSE_OP_CODE 0x5A5A: 运行烧写指令; 0x5AA5: 运行读取指令。(R/W)

Register 4.105. EFUSE_CMD_REG (0x01D4)

[illegible]

EFUSE_READ_CMD 置位发送读取指令。(R/WS/SC)

EFUSE_PGM_CMD 置位发送烧写指令。(R/WS/SC)

EFUSE_BLK_NUM 表明烧写哪个块，值 0~10 分别对应 BLOCK0~10。(R/W)

Register 4.106. EFUSE_DAC_CONF_REG (0x01E8)

(reserved)																EFUSE_OE_CLR		EFUSE_DAC_NUM								EFUSE_DAC_CLK_PAD_SEL								EFUSE_DAC_CLK_DIV							
31																18		17	16								9	8	7	0											
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0		255								0		28								Reset					

EFUSE_DAC_CLK_DIV 控制烧写电压的爬升时钟分频系数。(R/W)

EFUSE_DAC_CLK_PAD_SEL 无关项。(R/W)

EFUSE_DAC_NUM 烧写供电的上升周期。(R/W)

EFUSE_OE_CLR 降低烧写电压的供电能力。(R/W)

Register 4.107. EFUSE_RD_TIM_CONF_REG (0x01EC)

EFUSE_READ_INIT_NUM																(reserved)																	
31																	24	23															0
0x12																0 0																Reset	

EFUSE_READ_INIT_NUM 配置 eFuse 的首次读取时间。(R/W)

Register 4.108. EFUSE_WR_TIM_CONF1_REG (0x01F0)

(reserved)																EFUSE_PWR_ON_NUM																(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31								24								23								8								7								0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0								0															

EFUSE_PWR_ON_NUM 配置 VDDQ 的上电时间。(R/W)

Register 4.109. EFUSE_WR_TIM_CONF2_REG (0x01F4)

(reserved)																EFUSE_PWR_OFF_NUM																															
31																15																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x190																Reset															

EFUSE_PWR_OFF_NUM 配置 VDDQ 的掉电时间。(R/W)

Register 4.110. EFUSE_STATUS_REG (0x01D0)

(reserved)																EFUSE_REPEAT_ERR_CNT										(reserved)								EFUSE_STATE																																																																																													
31																18																17																10																9																4																3																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x0																0 0 0 0 0 0 0 0																0x0																Reset																																																															

EFUSE_STATE 表明 eFuse 控制器所处的状态。(RO)

EFUSE_REPEAT_ERR_CNT 表明烧写 BLOCK0 时的错误位的个数。(RO)

Register 4.111. EFUSE_INT_RAW_REG (0x01D8)

(reserved)																												EFUSE_PGM_DONE_INT_RAW EFUSE_READ_DONE_INT_RAW																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31																										2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EFUSE_READ_DONE_INT_RAW 读取完成中断的原始中断状态位。(R/WC/SS)

EFUSE_PGM_DONE_INT_RAW 烧写完成中断的原始中断状态位。(R/WC/SS)

Register 4.112. EFUSE_INT_ST_REG (0x01DC)

(reserved)																															EFUSE_PGM_DONE_INT_ST EFUSE_READ_DONE_INT_ST																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

EFUSE_READ_DONE_INT_ST 读取完成中断的状态位。(RO)

EFUSE_PGM_DONE_INT_ST 烧写完成中断的状态位。(RO)

Register 4.113. EFUSE_INT_ENA_REG (0x01E0)

(reserved)																															EFUSE_PG EFUSE_REPEAT																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EFUSE_READ_DONE_INT_ENA 读取完成中断的使能位。(R/W)

EFUSE_PGM_DONE_INT_ENA 烧写完成中断的使能位。(R/W)

Register 4.114. EFUSE_INT_CLR_REG (0x01E4)

(reserved)																															EFUSE_PGM EFUSE_REF																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

EFUSE_READ_DONE_INT_CLR 读取完成中断的清除位。(WO)

EFUSE_PGM_DONE_INT_CLR 烧写完成中断的清除位。(WO)

Register 4.115. EFUSE_DATE_REG (0x01FC)

(reserved)				EFUSE_DATE																							31	28	27	0
0	0	0	0	0x2006300																							Reset			

EFUSE_DATE 版本控制寄存器。(R/W)

5 IO MUX 和 GPIO 交换矩阵 (GPIO, IO MUX)

5.1 概述

ESP32-C3 芯片有 22 个物理通用输入输出管脚 (GPIO Pin)。每个管脚都可用作一个通用 IO，或连接一个内部的外设信号。利用 GPIO 交换矩阵和 IO MUX，可配置外设模块的输入信号来源于任何的 IO 管脚，并且外设模块的输出信号也可连接到任意 IO 管脚。这些模块共同组成了芯片的 IO 控制。

注意：这 22 个物理 GPIO 管脚的编号为：0 ~ 21。

5.2 主要特性

GPIO 交换矩阵主要特性

- GPIO 交换矩阵是外设输入输出信号和 GPIO 管脚之间的全交换矩阵；
- 42 个外设输入信号可以选择任意一个 GPIO 管脚的输入信号；
- 每个 GPIO 管脚的输出信号可以来自 78 个外设输出信号的任意一个；
- 支持输入信号经 GPIO SYNC 模块同步至 APB 时钟总线；
- 支持输入信号滤波；
- 支持 Sigma Delta 调制输出 (SDM)；
- 支持 GPIO 简单输入输出。

IO MUX 主要特性

- 为每个 GPIO 管脚提供一个寄存器 `IO_MUX_GPIOn_REG`，每个管脚可配置成：
 - GPIO 功能，连接 GPIO 交换矩阵；
 - 直连功能，旁路 GPIO 交换矩阵。
- 支持快速信号如 SPI、JTAG、UART 等可以旁路 GPIO 交换矩阵以实现更好的高频数字特性。所以高速信号会直接通过 IO MUX 输入和输出。

5.3 结构概览

本小节主要介绍 IO MUX 以及 GPIO 交换矩阵的架构，其中：

- 图 5-1 简要展示了 IO MUX 和 GPIO 交换矩阵的工作流程；
- 图 5-2 详细展示了 IO MUX 和 GPIO 交换矩阵将信号引入外设和引出至管脚的具体过程；
- 图 5-3 展示了 GPIO 管脚的接口逻辑。

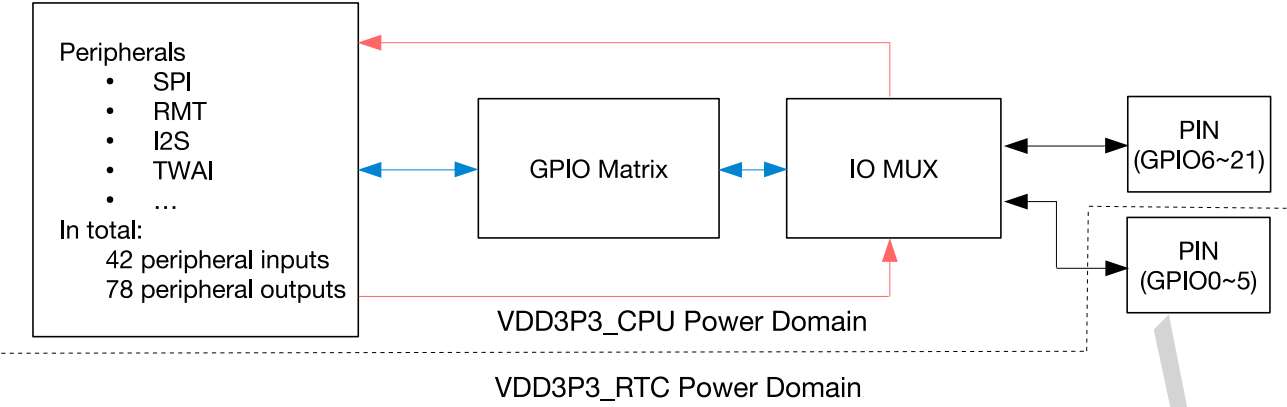


图 5-1. IO MUX 和 GPIO 交换矩阵框图（简图）

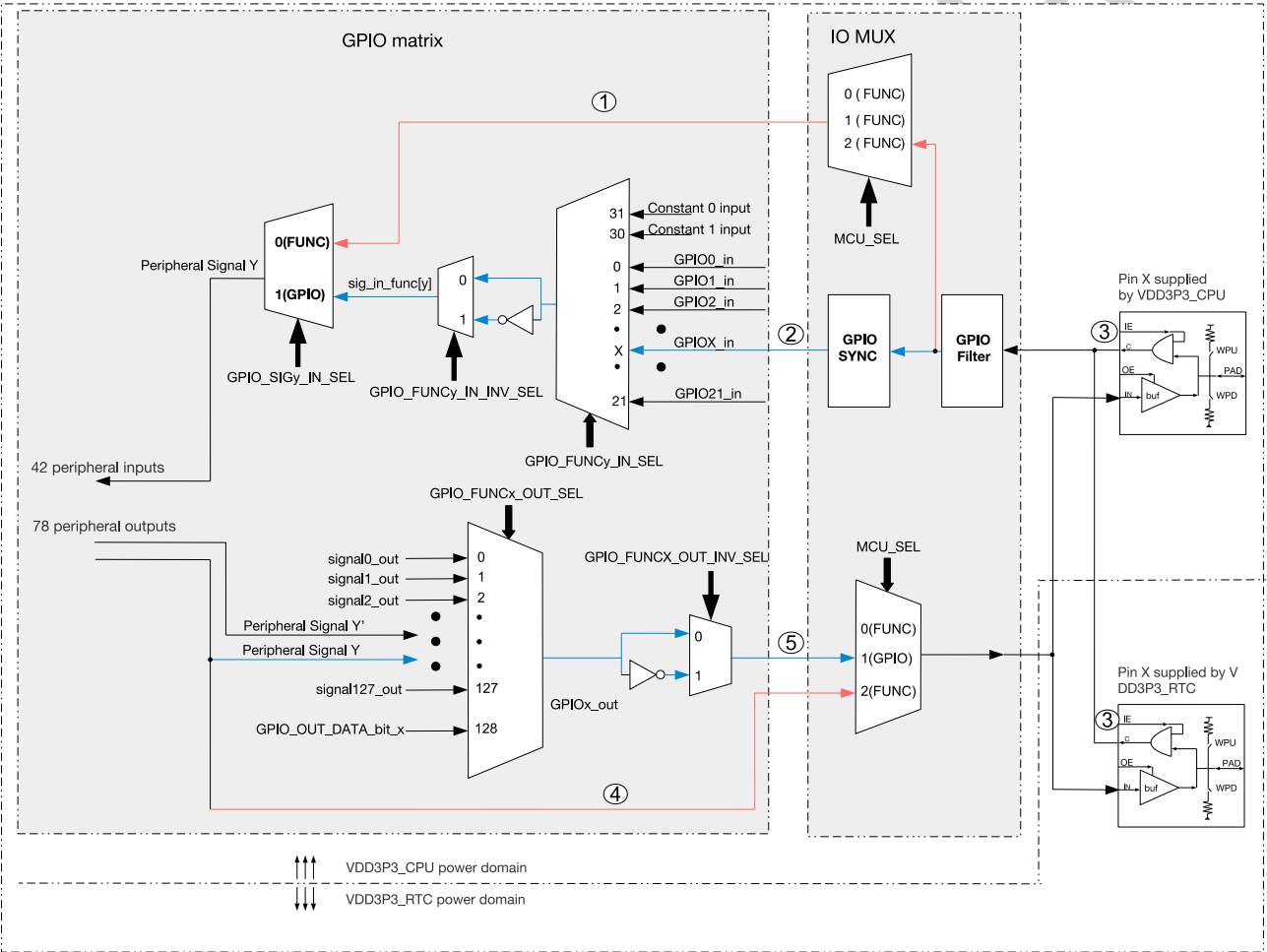


图 5-2. IO MUX 和 GPIO 交换矩阵框图（详图）

1. 仅有部分输入信号可以直接通过 IO MUX 直连外设，这些输入信号在表 5-1 “信号可由 IO MUX 直接输入” 一栏中被标为 “yes”。剩余其它信号只能通过 GPIO 交换矩阵连接至外设；
2. ESP32-C3 共有 22 个 GPIO 管脚，因此从 GPIO SYNC 进入到 GPIO 交换矩阵的输入共有 22 个；
3. 位于 VDD3P3_CPU 电源域和 VDD3P3_RTC 电源域的管脚由 IE、OE、WPU 和 WPD 信号控制；
4. 仅有部分输出信号可通过 IO MUX 直连管脚，这些输出信号在表 5-1 “信号可由 IO MUX 直接输出” 一栏中被标为 “yes”。剩余其它信号只能通过 GPIO 交换矩阵连接至外设；

5. 从 GPIO 交换矩阵到 IO MUX 的输出共有 22 个，对应 GPIO X: 0 ~ 21

图 5-3 展示了芯片焊盘的内部结构，即芯片逻辑与 GPIO 管脚之间的电气接口。22 个 GPIO 管脚均采用这一结构，且由 IE、OE、WPU 和 WPD 信号控制。

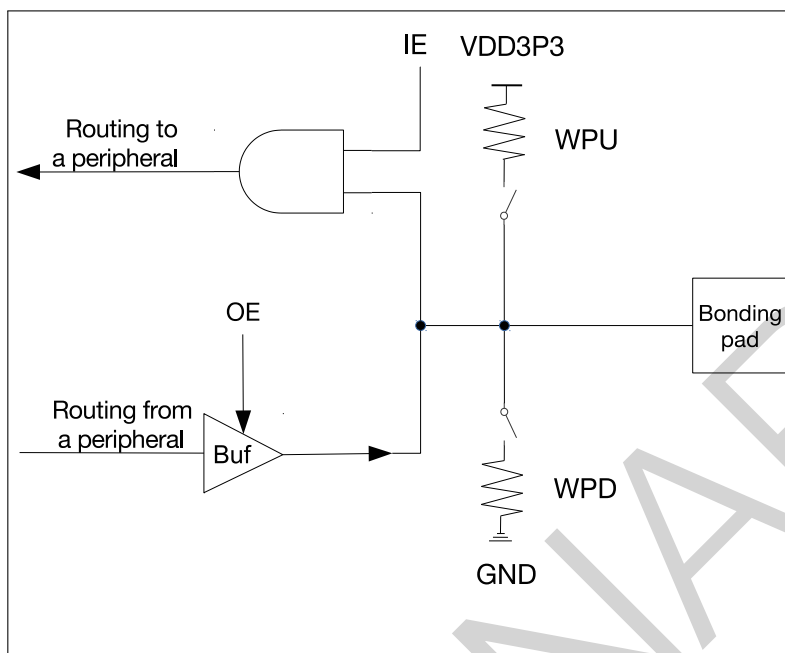


图 5-3. 焊盘内部结构

说明：

- IE: 输入使能
- OE: 输出使能
- WPU: 内部弱上拉
- WPD: 内部弱下拉
- Bonding pad: 接合焊盘，芯片逻辑的结点，实现芯片封装内晶片与 GPIO 管脚之间的物理连接。

5.4 通过 GPIO 交换矩阵的外设输入

5.4.1 概述

为实现通过 GPIO 交换矩阵接收外部输入信号，需要配置 GPIO 交换矩阵从 22 个 GPIO (0 ~ 21) 中获取外部输入信号，见交换矩阵表格 5-1。并需要配置外设输入选择通过 GPIO 交换矩阵接收输入信号。

5.4.2 信号同步

如图 5-2 所示，对于信号输入，外部输入信号从 GPIO 管脚输入，经 GPIO SYNC 模块同步至 APB 总线时钟后进入 GPIO 交换矩阵。外部输入信号也可以通过 IO MUX 直接进入外设，但信号无法经由 GPIO SYNC 模块同步。

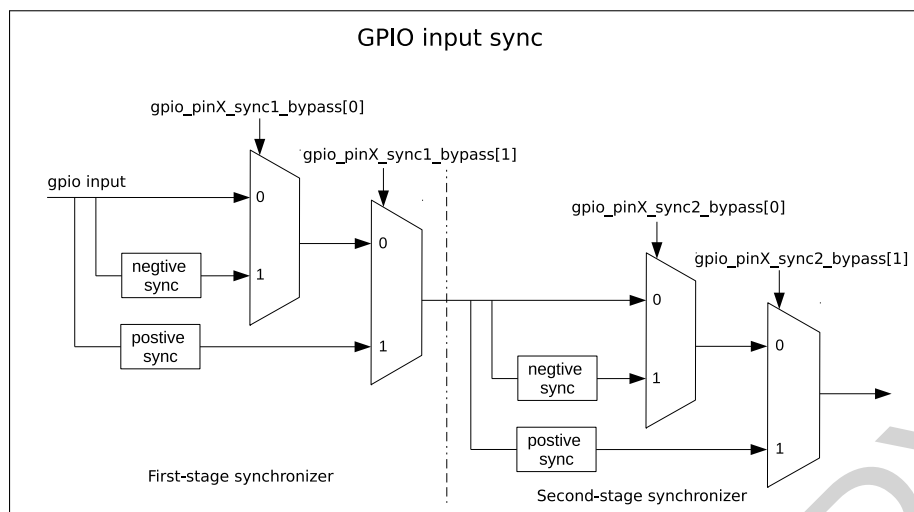


图 5-4. GPIO 输入经 APB 时钟上升沿或下降沿同步

GPIO SYNC 模块的功能如图 5-4 所示。其中，negative sync 为 GPIO 输入经过 APB 时钟的下降沿同步，positive sync 为 GPIO 输入经过 APB 时钟上升沿同步。

5.4.3 功能描述

把某个外设输入信号 Y 绑定到某个 GPIO 管脚 X^1 的配置过程如下：

1. 在 GPIO 交换矩阵中配置外设信号 Y 的 `GPIO_FUNC y _IN_SEL_CFG_REG` 寄存器：

- 置位 `GPIO_SIG y _IN_SEL` 选择通过 GPIO 交换矩阵接收外部输入信号。
- 设置 `GPIO_FUNC y _IN_SEL` 为需要的 GPIO 管脚编号，此处应为 X 。

注意：并不是所有外设信号都有有效的 `GPIO_SIG y _IN_SEL` 位，即有些外设信号只能通过 GPIO 交换矩阵接收外部输入信号。

2. 可选：置位 `IO_MUX_GPIO n _FILTER_EN` 使能 GPIO 管脚的输入信号滤波功能，如图 5-5 所示。只有当输入信号的有效宽度大于两个时钟周期时，输入信号才会被采样。否则，输入信号将会被滤掉。

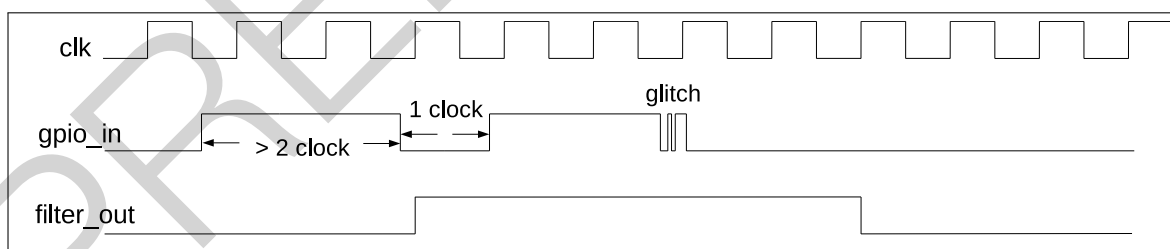


图 5-5. GPIO 输入信号滤波时序图

3. 同步 GPIO 输入信号。配置 GPIO 管脚 X 的 `GPIO_PIN x _REG` 来同步 GPIO 输入信号，过程如下：

- 如图 5-4 所示，配置 `GPIO_PIN x _SYNC1_BYPASS` 使能输入信号第一拍为上升沿或下降沿同步。
- 如图 5-4 所示，配置 `GPIO_PIN x _SYNC2_BYPASS` 使能输入信号第二拍为上升沿或下降沿同步。

4. 配置 IO MUX 寄存器使能 GPIO 管脚的输入功能。配置 GPIO 管脚 X 的 `IO_MUX_GPIO x _REG`，过程如下：

- 置位 `IO_MUX_GPIO x _FUN_IE` 使能输入²。

- 置位或清零 `IO_MUX_GPIOx_FUN_WPU` 和 `IO_MUX_GPIOx_FUN_WPD`, 使能或关闭内部上拉/下拉电阻。

例如, 要把 I2S MCLK 输入信号³ (`I2S_MCLK_in`, 信号索引号 12) 绑定到 GPIO7, 请按照以下步骤操作。注意, GPIO7 也叫做 MTDO 管脚。

1. 置位 `GPIO_FUNC12_IN_SEL_CFG_REG` 寄存器的 `GPIO_SIG12_IN_SEL` 位, 使能通过 GPIO 交换矩阵接收外部输入信号;
2. 配置 `GPIO_FUNC12_IN_SEL_CFG_REG` 寄存器中的 `GPIO_FUNC12_IN_SEL` 为 7, 即选择管脚 GPIO7;
3. 置位 `IO_MUX_GPIO7_REG` 寄存器中 `IO_MUX_GPIO7_FUN_IE` 位使能管脚输入。

说明:

1. 同一个输入管脚可以同时绑定多个输入信号;
2. 置位 `GPIO_FUNCy_IN_INV_SEL` 可以把输入信号取反;
3. 无需将输入信号绑定到一个 GPIO 管脚也可以使外设读取恒低或恒高电平的输入值。实现方式为选择特定的 `GPIO_FUNCy_IN_SEL` 输入值而不是一个 GPIO 序号:
 - 设置 `GPIO_FUNCy_IN_SEL` 为 0x1F, 则输入信号恒为 0;
 - 设置 `GPIO_FUNCy_IN_SEL` 为 0x1E, 则输入信号恒为 1。

5.4.4 简单 GPIO 输入

`GPIO_IN_REG` 寄存器存储着每个 GPIO 管脚的输入值。任意 GPIO 管脚的输入值都可以随时读取而无需为某一个外设信号配置 GPIO 交换矩阵。但需要配置 GPIO 管脚 x 对应的 `IO_MUX_GPIOx_REG` 中 `IO_MUX_GPIOx_FUN_IE` 位以使能输入, 如章节 5.4.2 所述。

5.5 通过 GPIO 交换矩阵的外设输出

5.5.1 概述

为实现通过 GPIO 交换矩阵输出外设信号, 需要配置 GPIO 交换矩阵将外设信号 (即在表 5-1 中“输出信号”一栏所列出的信号) 输出到 22 个 GPIO (0~21) 管脚。

输出信号从外设输出到 GPIO 交换矩阵, 然后到达 IO MUX。IO MUX 必须设置相应管脚为 GPIO 功能, 这样输出 GPIO 信号就能连接到相应管脚。

说明:

表 5-1 中输出索引号为 97~100 的外设信号, 没有连接至外设, 可配置为从一个 GPIO 管脚输出后, 直接由另一个 GPIO 管脚输入 (索引号: 97~100)。

5.5.2 功能描述

如图 5-2 所示, 对于信号输出, 78 个输出信号 (即在表 5-1 中“输出信号”列的所有信号) 中的某一个信号通过 GPIO 交换矩阵到达 IO MUX, 然后连接到某个 GPIO 管脚。

输出外设信号 Y 到某一 GPIO 管脚 X^1 的步骤如下:

1. 在 GPIO 交换矩阵中配置 GPIO 管脚 x 的 `GPIO_FUNCx_OUT_SEL_CFG_REG` 寄存器和 `GPIO_ENABLE_REG[x]` 字段。推荐使用相应 `W1TS` (写 1 置位) 和 `W1TC` (写 1 清零) 寄存器来更新 `GPIO_ENABLE_REG`

寄存器中的值：

- 设置 `GPIO_FUNCx_OUT_SEL_CFG_REG` 寄存器的 `GPIO_FUNCx_OUT_SEL` 字段为外设输出信号 Y 的索引号 (Y)。
 - 要将信号强制使能为输出模式，需要将 GPIO 管脚 X 对应的 `GPIO_FUNCx_OUT_SEL_CFG_REG` 寄存器中 `GPIO_FUNCx_OEN_SEL` 字段置位；同时需要将 `GPIO_ENABLE_W1TS_REG` 中的相应位置位。或者，将 `GPIO_FUNCx_OEN_SEL` 清零，即选择采用外设的输出使能信号，此时输出使能信号由内部逻辑功能决定。比如，表 5-1 中“`GPIO_FUNCn_OEN_SEL = 0` 时输出信号的输出使能信号”一栏的 `SPIQ_oe` 信号。
 - 置位 `GPIO_ENABLE_W1TC_REG` 中相应位可以关闭 GPIO 管脚的输出。
2. 要选择以开漏方式输出，可以设置 GPIO 管脚 X 的 `GPIO_PINx_REG` 寄存器中 `GPIO_PINx_PAD_DRIVER` 位。
 3. 配置 IO MUX 寄存器来选择经由 GPIO 交换矩阵输出信号。配置 GPIO 管脚 X 的 `IO_MUX_GPIOx_REG` 的过程如下：
 - 配置 GPIO 管脚 X 的 `IO_MUX_GPIOx_MCU_SEL` 为所需的管脚功能。此处选择数值 1，即 Function 1 (GPIO 功能)，适用于所有管脚。
 - 设置 `IO_MUX_GPIOx_FUN_DRV` 字段为特定的输出强度值 (0 ~ 3)，值越大，输出驱动能力越强：
 - 0: ~5 mA
 - 1: ~10 mA
 - 2: ~20 mA (默认值)
 - 3: ~40 mA
 - 在开漏模式下，通过置位/清零 `IO_MUX_GPIOx_FUN_WPU` 和 `IO_MUX_GPIOx_FUN_WPD` 使能或关闭上拉/下拉电阻。

说明：

1. 某一个外设的输出信号可以同时从多个管脚输出；
2. 置位 `GPIO_FUNCx_OUT_INV_SEL` 可以把输出的信号取反。

5.5.3 简单 GPIO 输出

GPIO 交换矩阵也可用于简单 GPIO 输出，具体配置如下：

- 设置 GPIO 交换矩阵 `GPIO_FUNCn_OUT_SEL` 寄存器为特定的外设索引值 128 (0x80)；
- 设置 `GPIO_OUT_REG` 寄存器中相应位的值为期望 GPIO 输出的值。

说明：

- `GPIO_OUT_REG[0] ~ GPIO_OUT_REG[21]` 对应 GPIO0 ~ GPIO21，`GPIO_OUT_REG[25:22]` 无效。
- 推荐使用相应的 W1TS 和 W1TC 寄存器，例如 `GPIO_OUT_W1TS/GPIO_OUT_W1TC` 来置位/清零 `GPIO_OUT_REG`。

5.5.4 Sigma Delta 调制输出 (SDM)

5.5.4.1 功能描述

125 个外设输出信号中有四个信号（在表 5-1 中索引为：55 ~ 58）支持 1-bit 二阶 SDM 调制输出。上述四个信号通道默认输出使能。Sigma Delta 调制器可实现输出可配占空比的 PDM（脉冲密度调制）信号。二阶 SDM 调制的转换公式如下：

$$H(z) = X(z)z^{-1} + E(z)(1-z^{-1})^2$$

$E(z)$ 为量化误差， $X(z)$ 为输入。

Sigma Delta 调制器内部支持对 APB_CLK 的 1 ~ 256 倍分频：

- 置位 GPIOSD_FUNCTION_CLK_EN 使能调制器时钟；
- 配置 GPIOSD_SD n _PRESCALE 实现分频。 n 取值范围为 0 ~ 3，对应四个信号通道。

分频后的时钟周期为调制器输出单位脉冲的周期。

GPIOSD_SD n _IN 为有符号数，范围为 [-128, 127]，配置此寄存器控制输出 PDM 信号的占空比¹。

- GPIOSD_SD n _IN = -128，调制器输出信号占空比为 0%；
- GPIOSD_SD n _IN = 0，调制器输出信号占空比接近 50%；
- GPIOSD_SD n _IN = 127，调制器输出信号占空比接近 100%。

PDM 信号占空比计算公式为：

$$Duty_Cycle = \frac{GPIOSD_SDn_IN + 128}{256}$$

说明：

对 PDM 信号来说，占空比是指在若干脉冲周期内（比如 256 个脉冲周期），高电平占整个统计周期的比值。

5.5.4.2 配置方法

SDM 的配置方法如下：

- 将 SDM 输出经 GPIO 交换矩阵连接至相应管脚，见 5.5.2 章节；
- 置位 GPIOSD_FUNCTION_CLK_EN，使能 SDM 时钟；
- 配置 GPIOSD_SD n _PRESCALE 寄存器设置时钟分频系数；
- 配置 GPIOSD_SD n _IN 寄存器设置 SDM 输出信号的占空比。

5.6 IO MUX 的直接输入输出功能

5.6.1 概述

快速信号如 SPI、JTAG 等会旁路 GPIO 交换矩阵以实现更好的高频数字特性。所以高速信号会直接通过 IO MUX 输入和输出。

这样比使用 GPIO 交换矩阵的灵活度要低，即每个 GPIO 管脚的 IO MUX 寄存器只有较少的功能选择，但可以实现更好的高频数字特性。

5.6.2 功能描述

对于外设输入信号，旁路 GPIO 交换矩阵必须配置两个寄存器：

1. GPIO 管脚的 `IO_MUX_GPIO n _MCU_SEL` 必须设置为相应的管脚功能，章节 5.11 列出了管脚功能。
2. 清零 `GPIO_SIG n _IN_SEL`，直接将输入信号连接到外设。

对于外设输出信号，旁路 GPIO 交换矩阵只需将 GPIO 管脚的 `IO_MUX_GPIO n _MCU_SEL` 配置为相应的管脚功能即可。

说明：

并非所有外设输入/输出信号均可直接通过 IO MUX 连接到外设，某些输入/输出信号只能通过 GPIO 交换矩阵连接到外设。

5.7 GPIO 管脚的模拟功能

ESP32-C3 部分 GPIO 管脚具有模拟功能。用于模拟功能时，请确保已按照下述方法关闭了上拉电阻和下拉电阻：

- 设置 `IO_MUX_GPIO n _MCU_SEL` 为 1，同时清零 `IO_MUX_GPIO n _FUN_IE`、`IO_MUX_GPIO n _FUN_WPU`、`IO_MUX_GPIO n _FUN_WPD`；
- 置位 `GPIO_ENABLE_W1TC $[n]$` ，清除输出使能。

表 5-4 列出了 ESP32-C3 管脚的模拟功能。

5.8 管脚 Hold 特性

每个 GPIO 管脚（包括 RTC 管脚 GPIO0 ~ GPIO5）都有单独的 Hold 功能，由 RTC 寄存器控制。管脚的 Hold 功能被置上后，管脚在置上 Hold 那一刻的状态被强制保持，无论内部信号如何变化，修改 IO MUX 配置或者 GPIO 配置，都不会改变管脚的状态。应用如果希望在看门狗超时触发内核复位和系统复位时或者 Deep-sleep 时管脚的状态不被改变，就需要提前把 Hold 置上。

说明：

- 对于数字管脚 (GPIO6 ~ 21)，若要在 Deep-sleep 中保持管脚输入输出的状态值，需要在掉电之前将寄存器 `RTC_CNTL_DIG_PAD_HOLD_REG` 中的 `RTC_CNTL_DIG_PAD_HOLD $[n]$` 位置 1。在芯片被唤醒后，若要关闭 Hold 功能，可将寄存器 `RTC_CNTL_DIG_PAD_HOLD $[n]$` 设置为 0。
- 对于 RTC 管脚 (GPIO0 ~ 5)，管脚的输入输出值由寄存器 `RTC_CNTL_RTC_PAD_HOLD_REG` 中的相应位控制。用户可置位或清除相应位来实现 Hold 或 Unhold 管脚输入输出值。

5.9 GPIO 管脚供电和电源管理

5.9.1 GPIO 管脚供电

GPIO 管脚供电请参考《ESP32-C3 规格书》中管脚定义章节。所有管脚均可用于将芯片从 Light-sleep 中唤醒，但仅有 VDD3P3_RTC 域中的管脚 (GPIO0 ~ GPIO5) 可用于将芯片从 Deep-sleep 唤醒。

5.9.2 电源管理

ESP32-C3 的管脚可分为如下两种不同的电源域。

- VDD3P3_RTC: RTC 和 CPU 的输入电源
- VDD3P3_CPU: CPU 的输入电源

5.10 外设信号列表

表 5-1 列出了所有经由 GPIO 交换矩阵的外设输入输出信号。

请注意 GPIO_FUNC*n*_OEN_SEL 位的配置：

- GPIO_FUNC*n*_OEN_SEL = 1，则寄存器 GPIO_ENABLE_REG 中的相应位 *n* 将用于控制信号输出使能。
 - GPIO_ENABLE_REG = 0：输出关闭；
 - GPIO_ENABLE_REG = 1：输出使能；
- GPIO_FUNC*n*_OEN_SEL = 0，则输出信号的使能由外设控制，例如表 5-1 中“GPIO_FUNC*n*_OEN_SEL = 0 时输出信号的输出使能信号”一栏的 SPIQ_oe。注意，使能信号 SPIQ_oe 可设置为 1 (1'd1) 或 0 (1'd0)，具体由外设的配置决定。如果“GPIO_FUNC*n*_OEN_SEL = 0 时输出信号的输出使能信号”一栏中为 1'd1，则表示寄存器 GPIO_FUNC*n*_OEN_SEL 已清零，输出信号默认始终使能。

说明：

信号连续编号，但并非所有信号均有效。

- 表 5-1 “输入信号”一栏中有名字的信号均为有效输入信号；
- 表 5-1 “输出信号”一栏中有名字的信号均为有效输出信号。

表 5-1. GPIO 交换矩阵外设信号

信号索引	输入信号	默认值	信号可经由 IO MUX 直接输入	输出信号	GPIO_FUNC _n _OEN_SEL = 0 时 输出信号的输出使能信号	信号可经由 IO MUX 直接输出
0	SPIQ_in	0	yes	SPIQ_out	SPIQ_oe	yes
1	SPID_in	0	yes	SPID_out	SPID_oe	yes
2	SPIHD_in	0	yes	SPIHD_out	SPIHD_oe	yes
3	SPIWP_in	0	yes	SPIWP_out	SPIWP_oe	yes
4	-	-	-	SPICLK_out_mux	SPICLK_oe	yes
5	-	-	-	SPICS0_out	SPICS0_oe	yes
6	U0RXD_in	0	yes	U0TXD_out	1'd1	yes
7	U0CTS_in	0	yes	U0RTS_out	1'd1	no
8	U0DSR_in	0	no	U0DTR_out	1'd1	no
9	U1RXD_in	0	yes	U1TXD_out	1'd1	no
10	U1CTS_in	0	yes	U1RTS_out	1'd1	no
11	U1DSR_in	0	no	U1DTR_out	1'd1	no
12	I2S_MCLK_in	0	no	I2S_MCLK_out	1'd1	no
13	I2SO_BCK_in	0	no	I2SO_BCK_out	1'd1	no
14	I2SO_WS_in	0	no	I2SO_WS_out	1'd1	no
15	I2SI_SD_in	0	no	I2SO_SD_out	1'd1	no
16	I2SI_BCK_in	0	no	I2SI_BCK_out	1'd1	no
17	I2SI_WS_in	0	no	I2SI_WS_out	1'd1	no
18	gpio_bt_priority	0	no	gpio_wlan_prio	1'd1	no
19	gpio_bt_active	0	no	gpio_wlan_active	1'd1	no
20	-	-	-	-	1'd1	no
21	-	-	-	-	1'd1	no
22	-	-	-	-	1'd1	no
23	-	-	-	-	1'd1	no
24	-	-	-	-	1'd1	no

信号索引	输入信号	默认值	信号可经由 IO MUX 直接输入	输出信号	GPIO_FUNC _n _OEN_SEL = 0 时 输出信号的输出使能信号	信号可经由 IO MUX 直接输出
25	-	-	-	-	1'd1	no
26	-	-	-	-	1'd1	no
27	-	-	-	-	1'd1	no
28	cpu_gpio_in0	0	no	cpu_gpio_out0	cpu_gpio_out_oen0	no
29	cpu_gpio_in1	0	no	cpu_gpio_out1	cpu_gpio_out_oen1	no
30	cpu_gpio_in2	0	no	cpu_gpio_out2	cpu_gpio_out_oen2	no
31	cpu_gpio_in3	0	no	cpu_gpio_out3	cpu_gpio_out_oen3	no
32	cpu_gpio_in4	0	no	cpu_gpio_out4	cpu_gpio_out_oen4	no
33	cpu_gpio_in5	0	no	cpu_gpio_out5	cpu_gpio_out_oen5	no
34	cpu_gpio_in6	0	no	cpu_gpio_out6	cpu_gpio_out_oen6	no
35	cpu_gpio_in7	0	no	cpu_gpio_out7	cpu_gpio_out_oen7	no
36	-	-	-	usb_jtag_tck	1'd1	no
37	-	-	-	usb_jtag_tms	1'd1	no
38	-	-	-	usb_jtag_tdi	1'd1	no
39	-	-	-	usb_jtag_tdo	1'd1	no
40	-	-	-	-	1'd1	no
41	-	-	-	-	1'd1	no
42	-	-	-	-	1'd1	no
43	-	-	-	-	1'd1	no
44	-	-	-	-	1'd1	no
45	ext_adc_start	0	no	ledc_ls_sig_out0	1'd1	no
46	-	-	-	ledc_ls_sig_out1	1'd1	no
47	-	-	-	ledc_ls_sig_out2	1'd1	no
48	-	-	-	ledc_ls_sig_out3	1'd1	no
49	-	-	-	ledc_ls_sig_out4	1'd1	no
50	-	-	-	ledc_ls_sig_out5	1'd1	no
51	rmt_sig_in0	0	no	rmt_sig_out0	1'd1	no

信号索引	输入信号	默认值	信号可经由 IO MUX 直接输入	输出信号	GPIO_FUNCn_OEN_SEL = 0 时 输出信号的输出使能信号	信号可经由 IO MUX 直接输出
52	rmt_sig_in1	0	no	rmt_sig_out1	1'd1	no
53	I2CEXT0_SCL_in	1	no	I2CEXT0_SCL_out	I2CEXT0_SCL_oe	no
54	I2CEXT0_SDA_in	1	no	I2CEXT0_SDA_out	I2CEXT0_SDA_oe	no
55	-	-	-	gpio_sd0_out	1'd1	no
56	-	-	-	gpio_sd1_out	1'd1	no
57	-	-	-	gpio_sd2_out	1'd1	no
58	-	-	-	gpio_sd3_out	1'd1	no
59	-	-	-	I2SO_SD1_out	1'd1	no
60	-	-	-	-	1'd1	no
61	-	-	-	-	1'd1	no
62	-	-	-	-	1'd1	no
63	FSPICLK_in	0	yes	FSPICLK_out_mux	FSPICLK_oe	yes
64	FSPIQ_in	0	yes	FSPIQ_out	FSPIQ_oe	yes
65	FSPID_in	0	yes	FSPID_out	FSPID_oe	yes
66	FSPIHD_in	0	yes	FSPIHD_out	FSPIHD_oe	yes
67	FSPIWP_in	0	yes	FSPIWP_out	FSPIWP_oe	yes
68	FSPICS0_in	0	yes	FSPICS0_out	FSPICS0_oe	yes
69	-	-	-	FSPICS1_out	FSPICS1_oe	no
70	-	-	-	FSPICS2_out	FSPICS2_oe	no
71	-	-	-	FSPICS3_out	FSPICS3_oe	no
72	-	-	-	FSPICS4_out	FSPICS4_oe	no
73	-	-	-	FSPICS5_out	FSPICS5_oe	no
74	twai_rx	1	no	twai_tx	1'd1	no
75	-	-	-	twai_bus_off_on	1'd1	no
76	-	-	-	twai_clkout	1'd1	no
77	-	-	-	-	1'd1	no
78	-	-	-	-	1'd1	no

信号索引	输入信号	默认值	信号可由 IO MUX 直接输入	输出信号	GPIO_FUNC <i>n</i> _OEN_SEL = 0 时 输出信号的输出使能信号	信号可由 IO MUX 直接输出
79	-	-	-	-	1'd1	no
80	-	-	-	-	1'd1	no
81	-	-	-	-	1'd1	no
82	-	-	-	-	1'd1	no
83	-	-	-	-	1'd1	no
84	-	-	-	-	1'd1	no
85	-	-	-	-	1'd1	no
86	-	-	-	-	1'd1	no
87	-	-	-	-	1'd1	no
88	-	-	-	-	1'd1	no
89	-	-	-	ant_sel0	1'd1	no
90	-	-	-	ant_sel1	1'd1	no
91	-	-	-	ant_sel2	1'd1	no
92	-	-	-	ant_sel3	1'd1	no
93	-	-	-	ant_sel4	1'd1	no
94	-	-	-	ant_sel5	1'd1	no
95	-	-	-	ant_sel6	1'd1	no
96	-	-	-	ant_sel7	1'd1	no
97	sig_in_func_97	0	no	sig_in_func97	1'd1	no
98	sig_in_func_98	0	no	sig_in_func98	1'd1	no
99	sig_in_func_99	0	no	sig_in_func99	1'd1	no
100	sig_in_func_100	0	no	sig_in_func100	1'd1	no
101	-	-	-	-	1'd1	no
102	-	-	-	-	1'd1	no
103	-	-	-	-	1'd1	no
104	-	-	-	-	1'd1	no
105	-	-	-	-	1'd1	no

信号索引	输入信号	默认值	信号可经由 IO MUX 直接输入	输出信号	GPIO_FUNC <i>n</i> _OEN_SEL = 0 时 输出信号的输出使能信号	信号可经由 IO MUX 直接输出
106	-	-	-	-	1'd1	no
107	-	-	-	-	1'd1	no
108	-	-	-	-	1'd1	no
109	-	-	-	-	1'd1	no
110	-	-	-	-	1'd1	no
111	-	-	-	-	1'd1	no
112	-	-	-	-	1'd1	no
113	-	-	-	-	1'd1	no
114	-	-	-	-	1'd1	no
115	-	-	-	-	1'd1	no
116	-	-	-	-	1'd1	no
117	-	-	-	-	1'd1	no
118	-	-	-	-	1'd1	no
119	-	-	-	-	1'd1	no
120	-	-	-	-	1'd1	no
121	-	-	-	-	1'd1	no
122	-	-	-	-	1'd1	no
123	-	-	-	CLK_OUT_out1	1'd1	no
124	-	-	-	CLK_OUT_out2	1'd1	no
125	-	-	-	CLK_OUT_out3	1'd1	no
126	-	-	-	SPICS1_out	1'd1	no
127	-	-	-	usb_jtag_trst	1'd1	no

5.11 IO MUX 管脚功能列表

表 5-2 列出了所有 GPIO 管脚的 IO MUX 功能。

表 5-2. IO MUX 管脚功能

GPIO	管脚名称	功能 0	功能 1	功能 2	功能 3	驱动强度	复位	说明
0	XTAL_32K_P	GPIO0	GPIO0	-	-	2	0	R
1	XTAL_32K_N	GPIO1	GPIO1	-	-	2	0	R
2	GPIO2	GPIO2	GPIO2	FSPIQ	-	2	1	R
3	GPIO3	GPIO3	GPIO3	-	-	2	1	R
4	MTMS	MTMS	GPIO4	FSPIHD	-	2	1	R
5	MTDI	MTDI	GPIO5	FSPIWP	-	2	1	R
6	MTCK	MTCK	GPIO6	FSPICLK	-	2	1*	G
7	MTDO	MTDO	GPIO7	FSPID	-	2	1	G
8	GPIO8	GPIO8	GPIO8	-	-	2	1	-
9	GPIO9	GPIO9	GPIO9	-	-	2	3	-
10	GPIO10	GPIO10	GPIO10	FSPICS0	-	2	1	G
11	VDD_SPI	GPIO11	GPIO11	-	-	2	0	-
12	SPIHD	SPIHD	GPIO12	-	-	2	3	-
13	SPIWP	SPIWP	GPIO13	-	-	2	3	-
14	SPICS0	SPICS0	GPIO14	-	-	2	3	-
15	SPICLK	SPICLK	GPIO15	-	-	2	3	-
16	SPID	SPID	GPIO16	-	-	2	3	-
17	SPIQ	SPIQ	GPIO17	-	-	2	3	-
18	GPIO18	GPIO18	GPIO18	-	-	3	0	USB, G
19	GPIO19	GPIO19	GPIO19	-	-	3	0*	USB
20	U0RXD	U0RXD	GPIO20	-	-	2	1	G
21	U0TXD	U0TXD	GPIO21	-	-	2	1	-

驱动强度

“驱动强度”一栏所示为每个管脚复位后的默认驱动强度。

- 0 - 驱动电流 = ~5 mA
- 1 - 驱动电流 = ~10 mA
- 2 - 驱动电流 = ~20 mA
- 3 - 驱动电流 = ~40 mA

复位

“复位”一栏所示为每个管脚复位后的默认配置。

- 0 - IE = 0 (输入关闭)
- 1 - IE = 1 (输入使能)
- 2 - IE = 1, WPD = 1 (输入使能, 下拉电阻使能)

- **3** - IE = 1, WPU = 1 (输入使能, 上拉电阻使能)
- **0*** - IE = 0, WPU = 0, GPIO19 的 USB 上拉默认值为 1, 因此, 其上拉电阻使能, 具体见说明。
- **1*** - 如果 EFUSE_DIS_PAD_JTAG = 1, 则 MTCK 管脚复位后浮空, 即 IE = 1。如果 EFUSE_DIS_PAD_JTAG = 0, 则 MTCK 管脚连接内部上拉电阻, 即 IE = 1, WPU = 1。

说明

- **R** - 代表位于 VDD3P3_RTC 电源域的管脚, 部分具有模拟功能, 见表 5-4。
- **USB** - GPIO18、GPIO19 为 USB 管脚。USB 管脚的上拉控制由管脚上拉和 USB 上拉共同控制。当其中任意一个为 1 时, 对应管脚上拉电阻使能。USB 上拉值对应寄存器 USB_SERIAL_JTAG_DP_PULLUP。
- **G** - 管脚在芯片上电过程中有毛刺, 具体见表 5-3。

表 5-3. 芯片上电过程中的管脚毛刺

管脚	毛刺类型	典型持续时间 (ns)
MTCK	低电平毛刺	5
MTDO	低电平毛刺	5
GPIO10	低电平毛刺	5
U0RXD	低电平毛刺	5
GPIO18	上拉	50000

5.12 IO MUX 管脚模拟功能列表

表 5-4 列出了具有模拟功能的 IO MUX 管脚。

表 5-4. IO MUX 管脚的模拟功能

GPIO 编号	管脚名称	模拟功能 0	模拟功能 1
0	XTAL_32K_P	XTAL_32K_P	ADC1_CH0
1	XTAL_32K_N	XTAL_32K_N	ADC1_CH1
2	GPIO2	-	ADC1_CH2
3	GPIO3	-	ADC1_CH3
4	MTMS	-	ADC1_CH4

5.13 寄存器列表

5.13.1 GPIO 交换矩阵寄存器列表

本小节的所有地址均为相对于 GPIO 基地址的地址偏移量 (相对地址), 具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
GPIO_BT_SELECT_REG	GPIO 位选择寄存器	0x0000	R/W
GPIO_OUT_REG	GPIO 输出寄存器	0x0004	R/W/SS
GPIO_OUT_W1TS_REG	GPIO 输出置位寄存器	0x0008	WT

名称	描述	地址	访问
GPIO_OUT_W1TC_REG	GPIO 输出清除寄存器	0x000C	WT
GPIO_ENABLE_REG	GPIO 输出使能寄存器	0x0020	R/W/SS
GPIO_ENABLE_W1TS_REG	GPIO 输出使能置位寄存器	0x0024	WT
GPIO_ENABLE_W1TC_REG	GPIO 输出使能清除寄存器	0x0028	WT
GPIO_STRAP_REG	Strapping 管脚寄存器	0x0038	RO
GPIO_IN_REG	GPIO 输入寄存器	0x003C	RO
GPIO_STATUS_REG	GPIO 中断状态寄存器	0x0044	R/W/SS
GPIO_STATUS_W1TS_REG	GPIO 中断状态置位寄存器	0x0048	WT
GPIO_STATUS_W1TC_REG	GPIO 中断状态清除寄存器	0x004C	WT
GPIO_PCPU_INT_REG	GPIO PRO_CPU 中断状态寄存器	0x005C	RO
GPIO_PCPU_NMI_INT_REG	GPIO PRO_CPU 非屏蔽中断状态寄存器	0x0060	RO
GPIO_STATUS_NEXT_REG	GPIO 中断源寄存器	0x014C	RO
管脚配置寄存器			
GPIO_PIN0_REG	配置 GPIO0 管脚	0x0074	R/W
GPIO_PIN1_REG	配置 GPIO1 管脚	0x0078	R/W
GPIO_PIN2_REG	配置 GPIO2 管脚	0x007C	R/W
GPIO_PIN3_REG	配置 GPIO3 管脚	0x0080	R/W
GPIO_PIN4_REG	配置 GPIO4 管脚	0x0084	R/W
GPIO_PIN5_REG	配置 GPIO5 管脚	0x0088	R/W
GPIO_PIN6_REG	配置 GPIO6 管脚	0x008C	R/W
GPIO_PIN7_REG	配置 GPIO7 管脚	0x0090	R/W
GPIO_PIN8_REG	配置 GPIO8 管脚	0x0094	R/W
GPIO_PIN9_REG	配置 GPIO9 管脚	0x0098	R/W
GPIO_PIN10_REG	配置 GPIO10 管脚	0x009C	R/W
GPIO_PIN11_REG	配置 GPIO11 管脚	0x00A0	R/W
GPIO_PIN12_REG	配置 GPIO12 管脚	0x00A4	R/W
GPIO_PIN13_REG	配置 GPIO13 管脚	0x00A8	R/W
GPIO_PIN14_REG	配置 GPIO14 管脚	0x00AC	R/W
GPIO_PIN15_REG	配置 GPIO15 管脚	0x00B0	R/W
GPIO_PIN16_REG	配置 GPIO16 管脚	0x00B4	R/W
GPIO_PIN17_REG	配置 GPIO17 管脚	0x00B8	R/W
GPIO_PIN18_REG	配置 GPIO18 管脚	0x00BC	R/W
GPIO_PIN19_REG	配置 GPIO19 管脚	0x00C0	R/W
GPIO_PIN20_REG	配置 GPIO20 管脚	0x00C4	R/W
GPIO_PIN21_REG	配置 GPIO21 管脚	0x00C8	R/W
输入配置寄存器			
GPIO_FUNC0_IN_SEL_CFG_REG	外设输入信号 0 配置寄存器	0x0154	R/W
GPIO_FUNC1_IN_SEL_CFG_REG	外设输入信号 1 配置寄存器	0x0158	R/W
GPIO_FUNC2_IN_SEL_CFG_REG	外设输入信号 2 配置寄存器	0x015C	R/W
...
GPIO_FUNC125_IN_SEL_CFG_REG	外设输入信号 125 配置寄存器	0x0348	R/W
GPIO_FUNC126_IN_SEL_CFG_REG	外设输入信号 126 配置寄存器	0x034C	R/W
GPIO_FUNC127_IN_SEL_CFG_REG	外设输入信号 127 配置寄存器	0x0350	R/W

名称	描述	地址	访问
输出配置寄存器			
GPIO_FUNC0_OUT_SEL_CFG_REG	GPIO0 管脚的输出配置寄存器	0x0554	R/W
GPIO_FUNC1_OUT_SEL_CFG_REG	GPIO1 管脚的输出配置寄存器	0x0558	R/W
GPIO_FUNC2_OUT_SEL_CFG_REG	GPIO2 管脚的输出配置寄存器	0x055C	R/W
GPIO_FUNC3_OUT_SEL_CFG_REG	GPIO3 管脚的输出配置寄存器	0x0560	R/W
GPIO_FUNC4_OUT_SEL_CFG_REG	GPIO4 管脚的输出配置寄存器	0x0564	R/W
GPIO_FUNC5_OUT_SEL_CFG_REG	GPIO5 管脚的输出配置寄存器	0x0568	R/W
GPIO_FUNC6_OUT_SEL_CFG_REG	GPIO6 管脚的输出配置寄存器	0x056C	R/W
GPIO_FUNC7_OUT_SEL_CFG_REG	GPIO7 管脚的输出配置寄存器	0x0570	R/W
GPIO_FUNC8_OUT_SEL_CFG_REG	GPIO8 管脚的输出配置寄存器	0x0574	R/W
GPIO_FUNC9_OUT_SEL_CFG_REG	GPIO9 管脚的输出配置寄存器	0x0578	R/W
GPIO_FUNC10_OUT_SEL_CFG_REG	GPIO10 管脚的输出配置寄存器	0x057C	R/W
GPIO_FUNC11_OUT_SEL_CFG_REG	GPIO11 管脚的输出配置寄存器	0x0580	R/W
GPIO_FUNC12_OUT_SEL_CFG_REG	GPIO12 管脚的输出配置寄存器	0x0584	R/W
GPIO_FUNC13_OUT_SEL_CFG_REG	GPIO13 管脚的输出配置寄存器	0x0588	R/W
GPIO_FUNC14_OUT_SEL_CFG_REG	GPIO14 管脚的输出配置寄存器	0x058C	R/W
GPIO_FUNC15_OUT_SEL_CFG_REG	GPIO15 管脚的输出配置寄存器	0x0590	R/W
GPIO_FUNC16_OUT_SEL_CFG_REG	GPIO16 管脚的输出配置寄存器	0x0594	R/W
GPIO_FUNC17_OUT_SEL_CFG_REG	GPIO17 管脚的输出配置寄存器	0x0598	R/W
GPIO_FUNC18_OUT_SEL_CFG_REG	GPIO18 管脚的输出配置寄存器	0x059C	R/W
GPIO_FUNC19_OUT_SEL_CFG_REG	GPIO19 管脚的输出配置寄存器	0x05A0	R/W
GPIO_FUNC20_OUT_SEL_CFG_REG	GPIO20 管脚的输出配置寄存器	0x05A4	R/W
GPIO_FUNC21_OUT_SEL_CFG_REG	GPIO21 管脚的输出配置寄存器	0x05A8	R/W
版本寄存器			
GPIO_DATE_REG	版本控制寄存器	0x06FC	R/W
时钟门控寄存器			
GPIO_CLOCK_GATE_REG	GPIO 时钟门控寄存器	0x062C	R/W

5.13.2 IO MUX 寄存器列表

本小节的所有地址均为相对于 IO MUX 基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

名称	描述	地址	访问
配置寄存器			
IO_MUX_PIN_CTRL_REG	时钟输出配置寄存器	0x0000	R/W
IO_MUX_GPIO0_REG	XTAL_32K_P 的 IO MUX 管脚配置寄存器	0x0004	R/W
IO_MUX_GPIO1_REG	XTAL_32K_N 的 IO MUX 管脚配置寄存器	0x0008	R/W
IO_MUX_GPIO2_REG	GPIO2 的 IO MUX 管脚配置寄存器	0x000C	R/W
IO_MUX_GPIO3_REG	GPIO3 的 IO MUX 管脚配置寄存器	0x0010	R/W
IO_MUX_GPIO4_REG	MTMS 的 IO MUX 管脚配置寄存器	0x0014	R/W
IO_MUX_GPIO5_REG	MTDI 的 IO MUX 管脚配置寄存器	0x0018	R/W
IO_MUX_GPIO6_REG	MTCK 的 IO MUX 管脚配置寄存器	0x001C	R/W
IO_MUX_GPIO7_REG	MTDO 的 IO MUX 管脚配置寄存器	0x0020	R/W

名称	描述	地址	访问
IO_MUX_GPIO8_REG	GPIO8 的 IO MUX 管脚配置寄存器	0x0024	R/W
IO_MUX_GPIO9_REG	GPIO9 的 IO MUX 管脚配置寄存器	0x0028	R/W
IO_MUX_GPIO10_REG	GPIO10 的 IO MUX 管脚配置寄存器	0x002C	R/W
IO_MUX_GPIO11_REG	VDD_SPI 的 IO MUX 管脚配置寄存器	0x0030	R/W
IO_MUX_GPIO12_REG	SPIHD 的 IO MUX 管脚配置寄存器	0x0034	R/W
IO_MUX_GPIO13_REG	SPIWP 的 IO MUX 管脚配置寄存器	0x0038	R/W
IO_MUX_GPIO14_REG	SPICS0 的 IO MUX 管脚配置寄存器	0x003C	R/W
IO_MUX_GPIO15_REG	SPICLK 的 IO MUX 管脚配置寄存器	0x0040	R/W
IO_MUX_GPIO16_REG	SPID 的 IO MUX 管脚配置寄存器	0x0044	R/W
IO_MUX_GPIO17_REG	SPIQ 的 IO MUX 管脚配置寄存器	0x0048	R/W
IO_MUX_GPIO18_REG	GPIO18 的 IO MUX 管脚配置寄存器	0x004C	R/W
IO_MUX_GPIO19_REG	GPIO19 的 IO MUX 管脚配置寄存器	0x0050	R/W
IO_MUX_GPIO20_REG	U0RXD 的 IO MUX 管脚配置寄存器	0x0054	R/W
IO_MUX_GPIO21_REG	U0TXD 的 IO MUX 管脚配置寄存器	0x0058	R/W
版本寄存器			
IO_MUX_DATE_REG	版本控制寄存器	0x00FC	R/W

5.13.3 SDM 寄存器列表

本小节的所有地址均为相对于 GPIO 基地址 + 0x0F00 的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

名称	描述	地址	访问
配置寄存器			
GPIOSD_SIGMADELTA0_REG	SDM0 占空比配置寄存器	0x0000	R/W
GPIOSD_SIGMADELTA1_REG	SDM1 占空比配置寄存器	0x0004	R/W
GPIOSD_SIGMADELTA2_REG	SDM2 占空比配置寄存器	0x0008	R/W
GPIOSD_SIGMADELTA3_REG	SDM3 占空比配置寄存器	0x000C	R/W
GPIOSD_SIGMADELTA_CG_REG	时钟门控配置寄存器	0x0020	R/W
GPIOSD_SIGMADELTA_MISC_REG	MISC 寄存器	0x0024	R/W
版本寄存器			
GPIOSD_SIGMADELTA_VERSION_REG	版本控制寄存器	0x0028	R/W

5.14 寄存器

5.14.1 GPIO 交换矩阵寄存器

本小节的所有地址均为相对于 GPIO 基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 5.1. GPIO_BT_SELECT_REG (0x0000)

GPIO_BT_SEL																															
31																															0
0x000000																															
Reset																															

GPIO_BT_SEL 保留 (R/W)

Register 5.2. GPIO_OUT_REG (0x0004)

(reserved)																															GPIO_OUT_DATA_ORIG																																			
31						26						25																									0																													
0						0						0						0						0						0						0x00000																									Reset					

GPIO_OUT_DATA_ORIG 简单 GPIO 输出模式下 GPIO0 ~ 21 的输出值。bit0 ~ bit21 的值分别对应 GPIO0 ~ GPIO21 的值。bit22 ~ bit25 无效。(R/W/SS)

Register 5.3. GPIO_OUT_W1TS_REG (0x0008)

(reserved)																															GPIO_OUT_W1TS														
31						26						25																									0								
0						0						0						0						0						0x00000						Reset									

GPIO_OUT_W1TS GPIO0 ~ 21 输出置位寄存器, bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位置 1, 则 GPIO_OUT_REG 中相应位也将置 1。注: 推荐使用此寄存器来置位 GPIO_OUT_REG。(WT)

Register 5.4. GPIO_OUT_W1TC_REG (0x000C)

Diagram illustrating the structure of the `GPIO_OUT_W1TC` register. The register is 32 bits wide, divided into two main sections:

- Reserved:** Bits 31 to 26 (6 bits) are reserved.
- GPIO_OUT_W1TC:** Bits 25 to 0 (26 bits) are used for the write-to-clear operation. This section is further divided into:
 - Bits 25 to 20 (6 bits): Labeled `0`.
 - Bits 19 to 0 (20 bits): Labeled `0x00000` and `Reset`.

GPIO_OUT_W1TC GPIO0~21 输出清零寄存器, bit0~bit21 对应 GPIO0~21, bit22~bit25 无效。每一位置 1, 则 **GPIO_OUT_REG** 中相应位会清零。注: 推荐使用此寄存器来清零 **GPIO_OUT_REG**。
(WT)

Register 5.5. GPIO_ENABLE_REG (0x0020)

Diagram illustrating the structure of the `GPIO_ENABLE_DATA` register. The register is 32 bits wide, divided into two 16-bit halves. The top half (bits 31-16) is labeled `(reserved)`. The bottom half (bits 15-0) is labeled `GPIO_ENABLE_DATA` and contains the value `0x00000`. A `Reset` label is at the bottom right.

GPIO_ENABLE_DATA GPIO0 ~ 21 输出使能寄存器, bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。(R/W/SS)

Register 5.6. GPIO_ENABLE_W1TS_REG (0x0024)

Diagram illustrating the structure of the `GPIO_ENABLE_W1TS` register. The register is 32 bits wide. Bits 31 down to 26 are reserved. Bits 25 down to 0 are used for `GPIO_ENABLE_W1TS`. The value `0x000000` is shown in the lower 26 bits.

GPIO_ENABLE_W1TS GPIO0 ~ 21 输出使能置位寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位置 1, 则 [GPIO_ENABLE_REG](#) 中相应位也将置 1。注: 推荐使用此寄存器来置位 [GPIO_ENABLE_REG](#)。(WT)

Register 5.7. GPIO_ENABLE_W1TC_REG (0x0028)

(reserved)						GPIO_ENABLE_W1TC																							
31	26					25																							0
0	0	0	0	0	0	0x00000																							Reset

GPIO_ENABLE_W1TC GPIO0 ~ 21 输出使能清零寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位置 1, 则 [GPIO_ENABLE_REG](#) 中相应位会清零。注: 推荐使用此寄存器清零 [GPIO_ENABLE_REG](#)。(WT)

Register 5.8. GPIO_STRAP_REG (0x0038)

(reserved)																GPIO_STRAPPING													
31	16															15													0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00												Reset

GPIO_STRAPPING GPIO Strapping 值。(RO)


- bit 0: 对应 GPIO2
- bit 2: 对应 GPIO8
- bit 3: 对应 GPIO9

Register 5.9. GPIO_IN_REG (0x003C)

(reserved)						GPIO_IN_DATA_NEXT																							
31	26					25																							0
0	0	0	0	0	0	0x00000																							Reset

GPIO_IN_DATA_NEXT GPIO0 ~ 21 输入值。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位代表一个管脚的片外输入值, 0 表示低电平, 1 表示高电平。(RO)

Register 5.10. GPIO_STATUS_REG (0x0044)

(reserved)						GPIO_STATUS_INTERRUPT																													
31						26	25																											0	
0	0	0	0	0	0	0	0x00000																												 Reset

GPIO_STATUS_INTERRUPT GPIO0 ~ 21 中断状态寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。(R/W/SS)

Register 5.11. GPIO_STATUS_W1TS_REG (0x0048)

(reserved)						GPIO_STATUS_W1TS																											
31						26	25																										0
0	0	0	0	0	0	0x00000																										Reset	

GPIO_STATUS_W1TS GPIO0 ~ 21 中断状态置位寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位置 1, 则 [GPIO_STATUS_INTERRUPT](#) 中相应位也将置 1。注: 推荐使用此寄存器来置位 [GPIO_STATUS_INTERRUPT](#)。(WT)

Register 5.12. GPIO_STATUS_W1TC_REG (0x004C)

(reserved)						GPIO_STATUS_W1TC																											
31						26	25																										0
0	0	0	0	0	0	0x00000																										Reset	

GPIO_STATUS_W1TC GPIO0 ~ 21 中断状态清除寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。每一位置 1, 则 [GPIO_STATUS_INTERRUPT](#) 中相应位会清零。注: 推荐使用此寄存器来清零 [GPIO_STATUS_INTERRUPT](#)。(WT)

Register 5.13. GPIO_PCPU_INT_REG (0x005C)

(reserved)						GPIO_PROCPU_INT																																
31						26																25																0
0	0	0	0	0	0	0	0x00000																	Reset														

GPIO_PROCPU_INT GPIO0 ~ 21 PRO_CPU 中断状态。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。如果 [GPIO_PIN_n_REG](#) 中 bit13 有效, 即使能 CPU 中断, 则此寄存器所示的中断状态应与 [GPIO_STATUS_REG](#) 中相应位的中断状态一致。(RO)

Register 5.14. GPIO_PCPU_NMI_INT_REG (0x0060)

(reserved)						GPIO_PROCPU_NMI_INT																	
31						26	25																0
0	0	0	0	0	0	0x00000																	Reset

GPIO_PROCPU_NMI_INT GPIO0 ~ 21 PRO_CPU 非屏蔽中断状态寄存器。bit0 ~ bit21 对应 GPIO0 ~ 21, bit22 ~ bit25 无效。如果 [GPIO_PIN_n_REG](#) 中 bit14 有效, 即使能 CPU 非屏蔽中断, 则该寄存器所示的中断状态应与 [GPIO_STATUS_REG](#) 中相应位的中断状态一致。(RO)

Register 5.15. GPIO_PIN n _REG (n : 0-21) (0x0074+4* n)

(reserved)														GPIO_PIN _n _INT_ENA				GPIO_PIN _n _CONFIG				GPIO_PIN _n _WAKEUP_ENABLE				GPIO_PIN _n _INT_TYPE				(reserved)				GPIO_PIN _n _SYNC1_BYPASS				GPIO_PIN _n _PAD_DRIVER				GPIO_PIN _n _SYNC2_BYPASS							
31														18				17				13				12		11		10		9		7		6		5		4		3		2		1		0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0														0x0				0x0		0		0x0		0		0		0x0		0		0x0		0		0x0		Reset											

GPIO_PIN n _SYNC2_BYPASS 使能 GPIO 输入信号第二拍为上升沿或下降沿同步。0: 关闭同步; 1: 下降沿同步; 2 或 3: 上升沿同步。(R/W)

GPIO_PIN_n_PAD_DRIVER 管脚驱动选择。0: 正常输出; 1: 开漏输出。(R/W)

GPIO_PIN_n_SYNC1_BYPASS 使能 GPIO 输入信号第一拍为上升沿或下降沿同步。0: 关闭同步; 1: 下降沿同步; 2 或 3: 上升沿同步。(R/W)

GPIO_PIN_n_INT_TYPE 中断类型选择。(R/W)

- 0: 禁用 GPIO 中断
- 1: 上升沿触发
- 2: 下降沿触发
- 3: 任一沿触发
- 4: 低电平触发
- 5: 高电平触发

GPIO_PIN_n_WAKEUP_ENABLE 使能 GPIO 唤醒，仅能将 CPU 从 Light-sleep 模式唤醒。(R/W)

GPIO_PIN_n_CONFIG 保留。(R/W)

GPIO_PIN_n_INT_ENA 中断使能位。bit13: 使能 CPU 中断; bit14: 使能 CPU 非屏蔽中断。(R/W)

Register 5.16. GPIO STATUS NEXT REG (0x014C)

Diagram illustrating the structure of the `GPIO_STATUS_INTERRUPT` register. The register is 32 bits wide, divided into two 16-bit halves. The top half is labeled `(reserved)` and the bottom half is labeled `GPIO_STATUS_INTERRUPT_NEXT`. The bottom half contains a `Reset` field and a `0x00000` value.

GPIO_STATUS_INTERRUPT_NEXT GPIO0 ~ 21 中断源信号，可以设置为上升沿中断、下降沿中断、电平敏感中断或任一沿中断。bit0 ~ bit21 对应 GPIO0 ~ 21，bit22 ~ bit25 无效。(RO)

Register 5.17. GPIO_FUNC n _IN_SEL_CFG_REG (n : 0-127) (0x0154+4* n)

(reserved)																												GPIO_FUNC _n _IN_SEL				GPIO_FUNC _n _IN_INV_SEL				GPIO_FUNC _n _IN_SEL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																												7	6	5	4	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

GPIO_FUNC n _IN_SEL 外设输入信号 n 的选择控制位。此位选择 1 个 GPIO 交换矩阵输入管脚与信号连接，或者选择 0x1E 与恒高电平输入信号连接，或者选择 0x1F 与恒低电平输入信号连接。(R/W)

GPIO_FUNC n _IN_INV_SEL 反转输入值。1: 反转; 0: 不反转。(R/W)

GPIO_SIG n _IN_SEL 旁路 GPIO 交换矩阵。1: 通过 GPIO 交换矩阵; 0: 直接通过 IO MUX 连接信号与外设。(R/W)

Register 5.18. GPIO_FUNC n _OUT_SEL_CFG_REG (n : 0-21) (0x0554+4* n)

(reserved)																GPIO_FUNC _n _OEN_INV_SEL				GPIO_FUNC _n _OEN_SEL				GPIO_FUNC _n _OUT_INV_SEL				GPIO_FUNC _n _OUT_SEL			
31																11				10	9	8	7	0							
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0 0 0 0				0x80				Reset							

Reset

GPIO_FUNC n _OUT_SEL GPIO 管脚输出 n 的选择控制位。如果该字段设置为 Y ($0 \leq Y < 128$), 则外设输出信号 Y 将连接至 GPIO n 输出。如果该字段设置为 128, 则寄存器 **GPIO_OUT_REG** 和 **GPIO_ENABLE_REG** 中的 bit n 将用作输出值和输出使能。(R/W)

GPIO_FUNC n _OUT_INV_SEL 0: 不反转输出值; 1: 反转输出值。(R/W)

GPIO_FUNC n _OEN_SEL 0: 采用外设的输出使能信号; 1: 强制使用 **GPIO_ENABLE_REG** 的 bit n 用作输出使能信号。(R/W)

GPIO_FUNC n _OEN_INV_SEL 0: 不反转输出使能信号; 1: 反转输出使能信号。(R/W)

Register 5.19. GPIO_CLOCK_GATE_REG (0x062C)

GPIO_CLK_EN 时钟门控使能。此位置 1，则时钟自由运转。(R/W)

Register 5.20. GPIO_DATE_REG (0x06FC)

(reserved)				GPIO_DATE_REG																
31	28	27																	0	
0	0	0	0	0x2006130																Reset

GPIO_DATA_REG 版本控制寄存器。(R/W)

5.14.2 IO MUX 寄存器

本小节的所有地址均为相对于 IO MUX 基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 5.21. IO_MUX_PIN_CTRL_REG (0x0000)

(reserved)																								IO_MUX_CLK_OUT3								IO_MUX_CLK_OUT2								IO_MUX_CLK_OUT1								
31												12												11	8				7	4				3	0													
0 0																								0x7								0xf								0xf								Reset

IO_MUX_CLK_OUTx 配置 I2S 外设时钟输出到 CLK_OUT_outx，需要设置 IO_MUX_CLK_OUTx 为 0x0。有关 CLK_OUT_outx 的信息，见表 5-1。(R/W)

5.14.3 SDM 寄存器

本小节的所有地址均为相对于 GPIO 基地址 + 0x0F00 的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 5.24. GPIOSD_SIGMADELTA_{*n*}_REG (*n*: 0-3) (0x0000+4**n*)

(reserved)																GPIOSD_SD _{<i>n</i>} _PRESCALE								GPIOSD_SD _{<i>n</i>} _IN																															
31																16								15								8								7								0							
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0xff								0x0								Reset																							

GPIOSD_SD_{*n*}_IN 配置 SDM 输出信号的占空比。(R/W)

GPIOSD_SD_{*n*}_PRESCALE 配置 APB_CLK 分频系数。(R/W)

Register 5.25. GPIOSD_SIGMADELTA.CG_REG (0x0020)

GPIOSD_CLK_EN																														(reserved)									
31	30																														0								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset				

GPIOSD_CLK_EN 使能 SDM 配置寄存器的时钟。(R/W)

Register 5.26. GPIOSD_SIGMADELTA_MISC_REG (0x0024)

(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
31	30	29	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOSD_FUNCTION_CLK_EN 使能 SDM 的时钟。(R/W)

GPIOSD_SPI_SWAP 保留。(R/W)

Register 5.27. GPIOSD_SIGMADELTA_VERSION_REG (0x0028)

(reserved)				GPIOSD_DATE																										
31	28	27																											0	
0	0	0	0	0x2006230																										Reset

GPIOSD_DATE 版本控制寄存器。(R/W)

6 复位和时钟

6.1 复位

6.1.1 概述

ESP32-C3 提供四种级别的复位方式，分别是 CPU 复位、内核复位、系统复位和芯片复位。除芯片复位外其它复位方式不影响片上内存存储的数据。图 6-1 展示了整个芯片系统的结构以及四种复位等级。

6.1.2 结构图

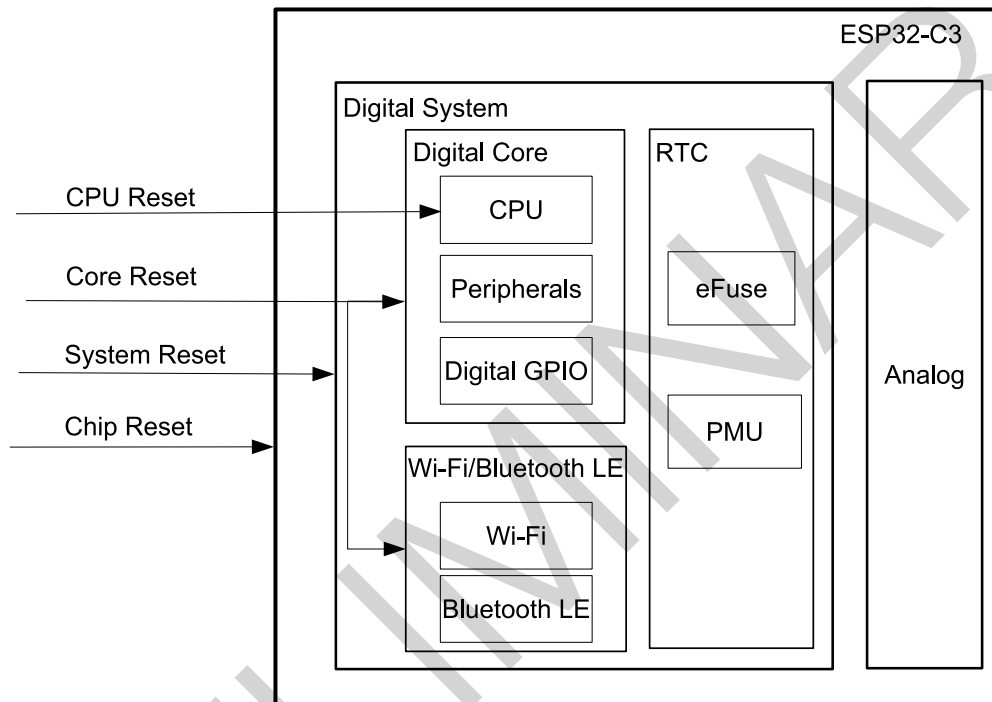


图 6-1. 四种复位等级

6.1.3 特性

- 支持四种复位等级：
 - CPU 复位：复位 CPU 核。复位释放后，程序将从 CPU Reset Vector 开始执行；
 - 内核复位：复位除 RTC 以外的其它数字系统，包括 CPU、外设、Wi-Fi、Bluetooth® LE 及数字 GPIO；
 - 系统复位：复位包括 RTC 在内的整个数字系统；
 - 芯片复位：复位整个芯片。
- 支持软件复位和硬件复位：
 - 软件复位：CPU 配置相关寄存器可触发软件复位，见章节 4 低功耗管理 (RTC_CNTL) [to be added later]；
 - 硬件复位：硬件复位直接由硬件电路触发。

说明:

如果 CPU 发生复位，则 [SENSITIVE 寄存器](#) 也将复位。

6.1.4 功能描述

上述任一复位发生时，CPU 将立刻复位。复位释放后，CPU 可通过读取寄存器 RTC_CNTL_RESET_CAUSE_PROCPU 获取复位源。

表 6-1 列出了从上述寄存器中可能读出的复位源以及触发的复位方式。

表 6-1. 复位源

编码	复位源	复位等级	说明
0x01	芯片复位	芯片复位	见表下方说明 ¹
0x0F	欠压系统复位	芯片复位或系统复位	欠压检测器触发的系统复位，见表下方说明 ²
0x10	RWDT 系统复位	系统复位	详见章节 11 看门狗定时器 (WDT)
0x12	超级看门狗复位	系统复位	详见章节 11 看门狗定时器 (WDT)
0x13	时钟毛刺复位	系统复位	详见章节 20 时钟毛刺检测
0x03	软件系统复位	内核复位	配置 RTC_CNTL_SW_SYS_RST 寄存器触发
0x05	Deep-sleep 复位	内核复位	详见章节 4 低功耗管理 (RTC_CNTL) [to be added later]
0x07	MWDT0 内核复位	内核复位	详见章节 11 看门狗定时器 (WDT)
0x08	MWDT1 内核复位	内核复位	详见章节 11 看门狗定时器 (WDT)
0x09	RWDT 内核复位	内核复位	详见章节 11 看门狗定时器 (WDT)
0x14	eFuse 复位	内核复位	eFuse CRC 校验错误触发复位
0x15	USB (UART) 复位	内核复位	外部 USB 主机发送特定命令给 USB-Serial-JTAG 的 Serial 接口将触发此复位，详见章节 24 USB 串口/JTAG 控制器 (USB_SERIAL_JTAG)
0x16	USB (JTAG) 复位	内核复位	外部 USB 主机发送特定命令给 USB-Serial-JTAG 的 JTAG 接口将触发此复位，详见章节 24 USB 串口/JTAG 控制器 (USB_SERIAL_JTAG)
0x17	电源毛刺复位	内核复位	电源毛刺触发复位
0x0B	MWDT0 CPU 复位	CPU 复位	详见章节 11 看门狗定时器 (WDT)
0x0C	软件 CPU 复位	CPU 复位	配置 RTC_CNTL_SW_PROCPU_RST 寄存器触发
0x0D	RWDT CPU 复位	CPU 复位	详见章节 11 看门狗定时器 (WDT)
0x11	MWDT1 CPU 复位	CPU 复位	详见章节 11 看门狗定时器 (WDT)

¹ 芯片复位的触发源包括以下两项：

- 芯片上电触发芯片复位
- 欠压检测器触发芯片复位

² 欠压检测器在检测到欠压状态时，将根据寄存器配置，选择触发系统复位或者芯片复位。详见章节 [4 低功耗管理 \(RTC_CNTL\)](#) [to be added later]。

6.2 时钟

6.2.1 概述

ESP32-C3 的时钟主要来源于振荡器 (oscillator, OSC)、RC 振荡电路和 PLL 时钟生成电路。上述时钟源产生的时钟经时钟分频器或时钟选择器等时钟模块的处理，使得大部分功能模块可以根据不同功耗和性能需求来获取及选择对应频率的工作时钟。图 6-2 为系统时钟结构。

6.2.2 结构图

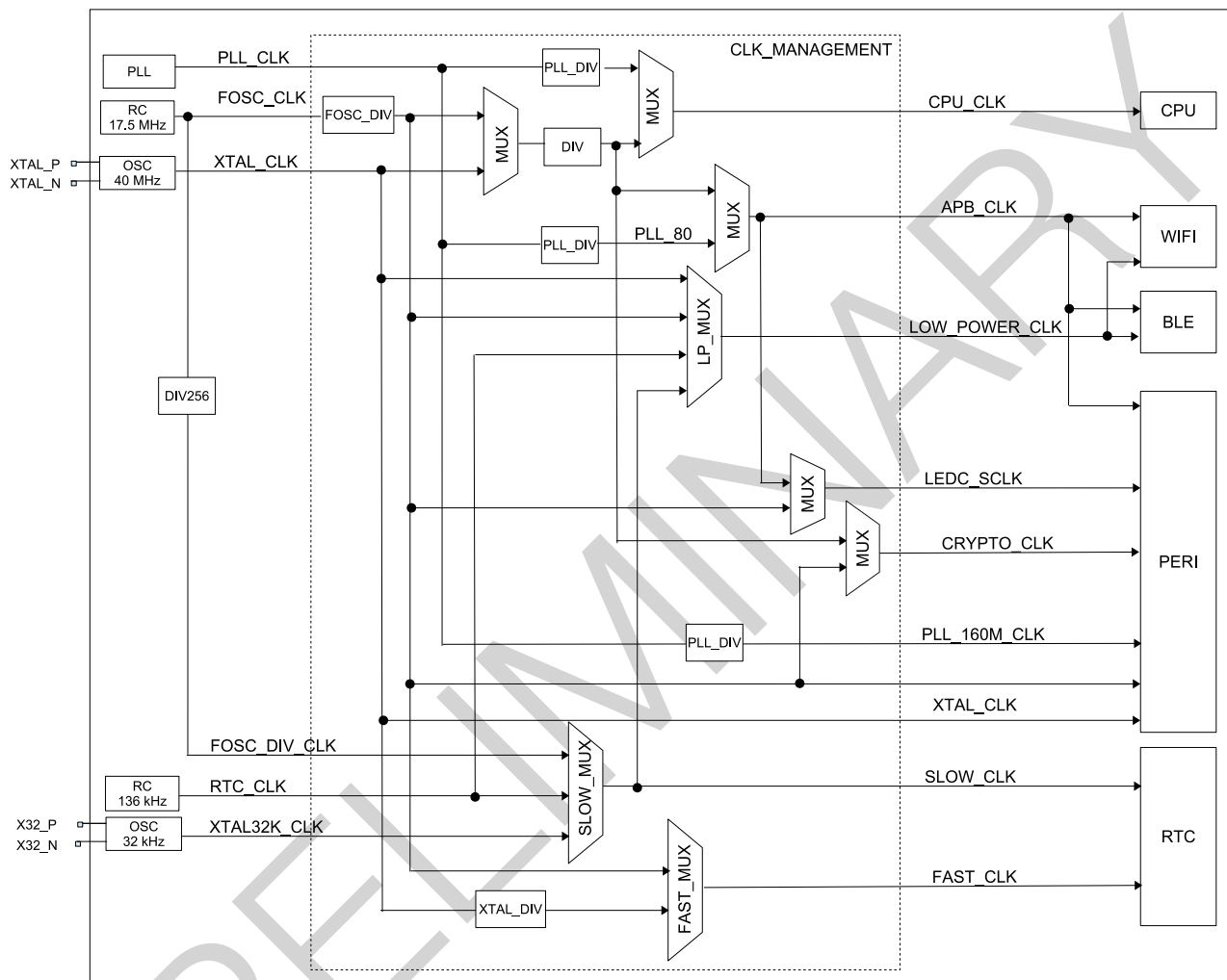


图 6-2. 系统时钟

6.2.3 特性

ESP32-C3 的时钟根据频率不同，可分为：

- 高性能时钟，主要为 CPU 和数字外设提供工作时钟
 - PLL_CLK：320 MHz 或 480 MHz 内部 PLL 时钟
 - XTAL_CLK：40 MHz 外部晶振时钟
- 低功耗时钟，主要为 RTC 模块以及部分处于低功耗模式的外设提供工作时钟
 - XTAL32K_CLK：32 kHz 外部晶振时钟
 - FOSC_CLK：内置快速 RC 振荡器时钟，频率可调节（通常为 17.5 MHz）

- FOSC_DIV_CLK: 内置快速 RC 振荡器分频时钟，由内置快速 RC 振荡器时钟经 256 分频生成
- RTC_CLK: 内置慢速 RC 振荡器，频率可调节（通常为 136 kHz）

6.2.4 功能描述

6.2.4.1 CPU 时钟

如图 6-2 所示，CPU_CLK 为 CPU 主时钟。CPU 在最高效工作模式下，主频可以达到 160 MHz。同时，CPU 能够在超低频下工作（通常为 2 MHz），以减少功耗。CPU_CLK 由 SYSTEM_SOC_CLK_SEL 来选择时钟源，允许选择 PLL_CLK、FOSC_CLK 或 XTAL_CLK 作为 CPU_CLK 的时钟源。具体请参考表 6-2 和表 6-3。默认状态下，CPU 的时钟为 XTAL_CLK，且分频系数为 2 分频，即 20 MHz。

表 6-2. CPU_CLK 时钟源选择

SYSTEM_SOC_CLK_SEL 值	时钟源
0	XTAL_CLK
1	PLL_CLK
2	FOSC_CLK

表 6-3. CPU_CLK 时钟频率

时钟源	SEL_0*	SEL_1*	SEL_2*	CPU 时钟频率
XTAL_CLK	0	-	-	CPU_CLK = XTAL_CLK/(SYSTEM_PRE_DIV_CNT + 1) SYSTEM_PRE_DIV_CNT 默认值为 1，范围 0 ~ 1023。
PLL_CLK (480 MHz)	1	1	0	CPU_CLK = PLL_CLK/6 CPU_CLK 频率为 80 MHz。
PLL_CLK (480 MHz)	1	1	1	CPU_CLK = PLL_CLK/3 CPU_CLK 频率为 160 MHz。
PLL_CLK (320 MHz)	1	0	0	CPU_CLK = PLL_CLK/4 CPU_CLK 频率为 80 MHz。
PLL_CLK (320 MHz)	1	0	1	CPU_CLK = PLL_CLK/2 CPU_CLK 频率为 160 MHz。
FOSC_CLK	2	-	-	CPU_CLK = FOSC_CLK/(SYSTEM_PRE_DIV_CNT + 1) SYSTEM_PRE_DIV_CNT 默认值为 1，范围 0 ~ 1023。

* 寄存器 SYSTEM_SOC_CLK_SEL 的值；
* 寄存器 SYSTEM_PLL_FREQ_SEL 的值；
* 寄存器 SYSTEM_CPUPERIOD_SEL 的值。

6.2.4.2 外设时钟

外设所需要的时钟包括 APB_CLK、CRYPTO_CLK、PLL_160M_CLK、LEDC_SCLK、XTAL_CLK 和 FOSC_CLK。表 6-4 列出了接入各个外设的时钟。

表 6-4. 外设时钟

Peripheral	XTAL_CLK	APB_CLK	PLL_160M_CLK	(RTC) FAST_CLK	FOSC_CLK	CRYPTO_CLK	LEDC_SCLK
TIMG	Y	Y					
I2S	Y		Y				
UHCI		Y					
UART	Y	Y			Y		
RMT	Y	Y			Y		
I2C	Y				Y		
SPI	Y	Y					
eFuse Controller				Y			
SARADC		Y					
Temperature Sensor	Y				Y		
USB		Y					
CRYPTO						Y	
TWAI Controller		Y					
LEDC	Y	Y	Y		Y		Y
SYS_TIMER	Y	Y					

APB_CLK 时钟

如表 6-5 所示，APB_CLK 的频率由 CPU_CLK 的时钟源决定。

表 6-5. APB_CLK 时钟

CPU_CLK 时钟源	APB_CLK 频率
PLL_CLK	80 MHz
XTAL_CLK	CPU_CLK
FOSC_CLK	CPU_CLK

CRYPTO_CLK 时钟

如表 6-6 所示，CRYPTO_CLK 的频率由 CPU_CLK 的时钟源决定。

表 6-6. CRYPTO_CLK 时钟

CPU_CLK 时钟源	CRYPTO_CLK 频率
PLL_CLK	160 MHz
XTAL_CLK	CPU_CLK
FOSC_CLK	CPU_CLK

PLL_160M_CLK 时钟

PLL_160M_CLK 是 PLL_CLK 根据当前 PLL 的频率分频所得。

LEDC_SCLK 时钟

LEDC 模块能将 FOSC_CLK 作为时钟源使用，即在 APB_CLK 关闭的时候，LEDC 也可工作。换言之，当系统处于低功耗模式时，其它外设都将停止工作（APB_CLK 关闭），但是 LEDC 仍然可以通过 FOSC_CLK 来正常工作。

6.2.4.3 Wi-Fi 和 Bluetooth® LE 时钟

Wi-Fi 和 Bluetooth LE 必须在 CPU_CLK 时钟源选择 PLL_CLK 下才能工作。只有当 Wi-Fi 和 Bluetooth LE 进入低功耗模式时，才能暂时关闭 PLL_CLK。

LOW_POWER_CLK 允许选择 XTAL32K_CLK、XTAL_CLK、FOSC_CLK、SLOW_CLK（RTC 当前所选的慢速时钟）用于 Wi-Fi 和 Bluetooth LE 的低功耗模式。

6.2.4.4 RTC 时钟

SLOW_CLK 和 FAST_CLK 的时钟源为低频时钟。RTC 模块能够在大多数时钟源关闭的状态下工作。SLOW_CLK 允许选择 RTC_CLK、XTAL32K_CLK 或 FOSC_DIV_CLK，用于驱动功耗管理模块。FAST_CLK 允许选择 XTAL_CLK 的分频时钟或 FOSC_CLK 的分频时钟，用于驱动片上传感器模块。

7 芯片 Boot 控制

7.1 概述

ESP32-C3 共有三个 Strapping 管脚：

- GPIO2
- GPIO8
- GPIO9

Strapping 管脚可用于控制 ESP32-C3 芯片上电或硬件复位时的一些功能：

- 控制 Boot 模式
- 控制 ROM 代码日志打印到 UART

在系统复位过程中，包括上电复位、欠压复位和模拟超级看门狗复位，（请参考 6 复位和时钟章节），硬件将采样 Strapping 管脚电平存储到锁存器中，并一直保持到芯片掉电或关闭。GPIO2、GPIO8 和 GPIO9 锁存的状态可以通过软件从寄存器 `GPIO_STRAPPING` 中读取。

GPIO9 默认连接内部上拉电阻。如果这一管脚没有外部连接或者连接的外部线路处于高阻抗状态，内部弱上拉将决定这一管脚输入电平的默认值，如表 7-1 所示。

表 7-1. 管脚默认上拉/下拉

管脚	默认值
GPIO2	N/A
GPIO8	N/A
GPIO9	上拉

如需改变 Strapping 管脚的默认值，用户可以应用外部下拉/上拉电阻，或者应用主机 MCU 的 GPIO 来控制 ESP32-C3 上电复位时的 Strapping 管脚电平。复位释放后，Strapping 管脚和普通管脚功能相同。

说明：

以下小节介绍了芯片复位时的功能以及控制该功能使用到的 Strapping 组合模式。请使用本章节所介绍的组合，其它组合可能会导致不可控结果。

7.2 Boot 模式控制

复位释放后，GPIO2、GPIO8 和 GPIO9 共同控制 Boot 模式。

表 7-2. 系统启动模式

启动模式	GPIO2	GPIO8	GPIO9
SPI Boot 模式	1	x	1
Download Boot 模式	1	1	0

表 7-2 列出了 GPIO2、GPIO8 和 GPIO9 的 Strapping 值及其对应的系统启动模式。此处“x”表示该项为无关项。

在 SPI Boot 模式下，CPU 通过从 SPI flash 中读取程序来启动系统。SPI Boot 模式可进一步细分为以下两种启动方式：

- 常规 flash 启动方式：支持安全启动，程序运行在 RAM 中；
- 直接启动方式：不支持安全启动，程序直接运行在 flash 中。如需使能这一启动方式，请确保下载至 flash 的 bin 文件其前两个字（地址：0x42000000）为 0xaebd041d。

在 Download Boot 模式下，用户可通过 UART0 或 USB 接口将代码下载至 flash 中，或将程序加载到 SRAM 并在 SRAM 中运行程序。

下面几个 eFuse 可用于控制启动模式的具体行为：

- [EFUSE_DIS_FORCE_DOWNLOAD](#)

如果此 eFuse 设置为 0（默认），软件可通过设置 RTC_CNTL_FORCE_DOWNLOAD_BOOT，触发 CPU 复位，将芯片启动模式强制从 SPI Boot 模式切换至 Download Boot 模式；如果此 eFuse 设置为 1，则禁用 RTC_CNTL_FORCE_DOWNLOAD_BOOT。

- [EFUSE_DIS_DOWNLOAD_MODE](#)

如果此 eFuse 设置为 1，则禁用 Download Boot 模式。

- [EFUSE_ENABLE_SECURITY_DOWNLOAD](#)

如果此 eFuse 设置为 1，则在 Download Boot 模式下，只允许读取、写入和擦除明文 flash，不支持 SRAM 或寄存器操作。如已禁用 Download Boot 模式，请忽略此 eFuse。

USB Serial/JTAG 控制器可将芯片从 SPI Boot 模式强制切换到 Download Boot 模式，或从 Download Boot 模式强制切换到 SPI Boot 模式。更多信息，请参考章节 [24 USB 串口/JTAG 控制器 \(USB_SERIAL_JTAG\)](#)。

7.3 ROM 代码日志打印控制

在系统启动早期阶段，GPIO8 与 eFuse [UART_PRINT_CONTROL](#) 一起控制 ROM 代码日志打印。

表 7-3. ROM 代码日志打印控制

eFuse ¹	GPIO8	ROM 代码日志打印
0	x	启动过程中，ROM 代码日志始终打印至 UART，此时 GPIO8 的值被忽略
1	0	启动过程中使能打印
	1	启动过程中关闭打印
2	0	启动过程中关闭打印
	1	启动过程中使能打印
3	x	启动过程中始终关闭打印，此时 GPIO8 的值被忽略

¹ eFuse: EFUSE_UART_PRINT_CONTROL

ROM 代码日志上电默认打印至 U0TXD，也可配置成打印到 USB Serial/JTAG 控制器，具体由 [EFUSE_USB_PRINT_CHANNEL](#) 控制：

- 0：打印至 USB
- 1：打印至 UART

注意：如果 [EFUSE_USB_PRINT_CHANNEL](#) 设置为 0，即选择打印至 USB，但如果 USB Serial/JTAG 控制器已被禁用的话，则 ROM 代码将无法打印。

PRELIMINARY

8 中断矩阵 (INTMTRX)

8.1 概述

ESP32-C3 中断矩阵将任一外部中断源单独映射到 ESP-RISC-V CPU 的任一外部中断上，以便在外设中断信号产生后，及时通知 CPU 进行处理。

ESP32-C3 有 62 个外部中断源，但 CPU 只支持 31 个中断。因此，将这些外部中断源映射至 CPU 中断必须使用中断矩阵。

说明：

本章节只涉及将外部中断源映射到 CPU 中断，关于中断配置、中断向量表、中断服务程序推荐处理机制请参考章节 1 [ESP-RISC-V CPU](#)。

8.2 特性

- 接收 62 个外部中断源作为输入
- 生成 31 个 CPU 的外部中断作为输出
- 支持查询外部中断源当前的中断状态
- 支持配置 CPU 的中断优先级、中断类型、中断阈值以及中断使能

中断矩阵的结构如图 8-1 所示。

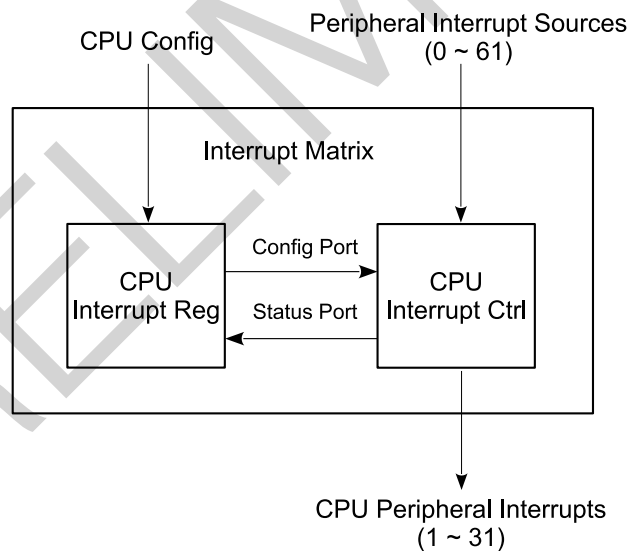


图 8-1. 中断矩阵结构图

8.3 功能描述

8.3.1 外部中断源

ESP32-C3 共有 62 个外部中断源。表 8-1 列出了所有外部中断源，以及对应的中断配置寄存器与中断状态寄存器。

- “No.”：表示外部中断源序号，范围：0 ~ 61
- “中断源”：外部中断源名称
- “配置寄存器”：用于将外部中断源分配至 CPU 外部中断
- “状态寄存器”：用于读取中断源的中断状态
 - “状态寄存器 - 位”：表示在状态寄存器中的比特位置，用于记录相应中断源的状态
 - “状态寄存器 - 名称”：表示状态寄存器的名称

表 8-1. CPU 外部中断配置寄存器、外部中断状态寄存器、外部中断源

No.	中断源	配置寄存器	位	状态寄存器名称
0	保留	保留	0	INTERRUPT_CORE0_INTR_STATUS_0_REG
1	保留	保留	1	
2	PWR_INTR	INTERRUPT_CORE0_PWR_INTR_MAP_REG	2	
3	保留	保留	3	
4	保留	保留	4	
5	保留	保留	5	
6	保留	保留	6	
7	保留	保留	7	
8	保留	保留	8	
9	保留	保留	9	
10	保留	保留	10	
11	I2C_MST_INT	INTERRUPT_CORE0_I2C_MST_INT_MAP_REG	11	
12	SLC0_INTR	INTERRUPT_CORE0_SLC0_INTR_MAP_REG	12	
13	SLC1_INTR	INTERRUPT_CORE0_SLC1_INTR_MAP_REG	13	
14	APB_CTRL_INTR	INTERRUPT_CORE0_APB_CTRL_INTR_MAP_REG	14	
15	UHCIO_INTR	INTERRUPT_CORE0_UHCIO_INTR_MAP_REG	15	
16	GPIO_INTERRUPT_PRO	INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_MAP_REG	16	
17	GPIO_INTERRUPT_PRO_NMI	INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_NMI_MAP_REG	17	
18	SPI_INTR_1	INTERRUPT_CORE0_SPI_INTR_1_MAP_REG	18	
19	SPI_INTR_2	INTERRUPT_CORE0_SPI_INTR_2_MAP_REG	19	
20	I2S1_INT	INTERRUPT_CORE0_I2S1_INT_MAP_REG	20	
21	UART_INTR	INTERRUPT_CORE0_UART_INTR_MAP_REG	21	
22	UART1_INTR	INTERRUPT_CORE0_UART1_INTR_MAP_REG	22	
23	LEDC_INT	INTERRUPT_CORE0_LEDC_INT_MAP_REG	23	
24	EFUSE_INT	INTERRUPT_CORE0_EFUSE_INT_MAP_REG	24	
25	CAN_INT	INTERRUPT_CORE0_CAN_INT_MAP_REG	25	
26	USB_INTR	INTERRUPT_CORE0_USB_INTR_MAP_REG	26	
27	RTC_CORE_INTR	INTERRUPT_CORE0_RTC_CORE_INTR_MAP_REG	27	
28	RMT_INTR	INTERRUPT_CORE0_RMT_INTR_MAP_REG	28	
29	I2C_EXT0_INTR	INTERRUPT_CORE0_I2C_EXT0_INTR_MAP_REG	29	
30	TIMER_INT1	INTERRUPT_CORE0_TIMER_INT1_MAP_REG	30	
31	TIMER_INT2	INTERRUPT_CORE0_TIMER_INT2_MAP_REG	31	

No.	中断源	配置寄存器	位	状态寄存器名称
32	TG_T0_INT	INTERRUPT_CORE0_TG_T0_INT_MAP_REG	0	INTERRUPT_CORE0_INTR_STATUS_1_REG
33	TG_WDT_INT	INTERRUPT_CORE0_TG_WDT_INT_MAP_REG	1	
34	TG1_T0_INT	INTERRUPT_CORE0_TG1_T0_INT_MAP_REG	2	
35	TG1_WDT_INT	INTERRUPT_CORE0_TG1_WDT_INT_MAP_REG	3	
36	CACHE_IA_INT	INTERRUPT_CORE0_CACHE_IA_INT_MAP_REG	4	
37	SYSTIMER_TARGET0_INT	INTERRUPT_CORE0_SYSTIMER_TARGET0_INT_MAP_REG	5	
38	SYSTIMER_TARGET1_INT	INTERRUPT_CORE0_SYSTIMER_TARGET1_INT_MAP_REG	6	
39	SYSTIMER_TARGET2_INT	INTERRUPT_CORE0_SYSTIMER_TARGET2_INT_MAP_REG	7	
40	SPI_MEM_REJECT_INTR	INTERRUPT_CORE0_SPI_MEM_REJECT_INTR_MAP_REG	8	
41	ICACHE_PRELOAD_INT	INTERRUPT_CORE0_ICACHE_PRELOAD_INT_MAP_REG	9	
42	ICACHE_SYNC_INT	INTERRUPT_CORE0_ICACHE_SYNC_INT_MAP_REG	10	
43	APB_ADC_INT	INTERRUPT_CORE0_APB_ADC_INT_MAP_REG	11	
44	DMA_CH0_INT	INTERRUPT_CORE0_DMA_CH0_INT_MAP_REG	12	
45	DMA_CH1_INT	INTERRUPT_CORE0_DMA_CH1_INT_MAP_REG	13	
46	DMA_CH2_INT	INTERRUPT_CORE0_DMA_CH2_INT_MAP_REG	14	
47	RSA_INTR	INTERRUPT_CORE0_RSA_INTR_MAP_REG	15	
48	AES_INTR	INTERRUPT_CORE0_AES_INTR_MAP_REG	16	
49	SHA_INTR	INTERRUPT_CORE0_SHA_INTR_MAP_REG	17	
50	CPU_INTR_FROM_CPU_0	INTERRUPT_CORE0_CPU_INTR_FROM_CPU_0_MAP_REG	18	
51	CPU_INTR_FROM_CPU_1	INTERRUPT_CORE0_CPU_INTR_FROM_CPU_1_MAP_REG	19	
52	CPU_INTR_FROM_CPU_2	INTERRUPT_CORE0_CPU_INTR_FROM_CPU_2_MAP_REG	20	
53	CPU_INTR_FROM_CPU_3	INTERRUPT_CORE0_CPU_INTR_FROM_CPU_3_MAP_REG	21	
54	ASSIST_DEBUG_INTR	INTERRUPT_CORE0_ASSIST_DEBUG_INTR_MAP_REG	22	
55	DMA_APB_PMS_MONITOR_VIOLATE_INTR	INTERRUPT_CORE0_DMA_APBPERI_PMS_MONITOR_VIOLATE_INTR_MAP_REG	23	
56	CORE_0_IRAM0_PMS_MONITOR_VIOLATE_INTR	INTERRUPT_CORE0_CORE_0_IRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG	24	
57	CORE_0_DRAM0_PMS_MONITOR_VIOLATE_INTR	INTERRUPT_CORE0_CORE_0_DRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG	25	
58	CORE_0_PIF_PMS_MONITOR_VIOLATE_INTR	INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_INTR_MAP_REG	26	
59	CORE_0_PIF_PMS_MONITOR_VIOLATE_SIZE_INTR	INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_SIZE_INTR_MAP_REG	27	
60	BACKUP_PMS_VIOLATE_INT	INTERRUPT_CORE0_CACHE_CORE0_ACS_INT_MAP_REG	28	
61	CACHE_CORE0_ACS_INT	INTERRUPT_CORE0_CACHE_CORE0_ACS_INT_MAP_REG	29	

8.3.2 CPU 中断

ESP32-C3 采用非 RISC-V 标准规范中断机制，CPU 共有 31 个中断号 (1 ~ 31)，每个中断：

- 优先级可设置为 1 ~ 15（数字越大优先级越高）；
- 可配置为电平触发或者边沿触发；
- 可通过设置中断阈值屏蔽低优先级的中断。

说明：

CPU 中断的具体配置见章节 1 *ESP-RISC-V CPU*。

8.3.3 分配外部中断源至 CPU 外部中断

在本小节中，我们将使用以下术语描述中断矩阵相关操作：

- Source_X：代表某个外部中断源，其中 X 为中断源编号，详见表 8-1。
- INTERRUPT_CORE0_SOURCE_X_MAP_REG：CPU 外部中断源 (Source_X) 的中断配置寄存器。
- NUM_P：表示 CPU 中断编号，范围：1 ~ 31。
- Interrupt_P：表示中断号为 Num_P 的 CPU 中断。

8.3.3.1 分配一个外部中断源 Source_X 至 CPU 外部中断

将外部中断源 Source_X 对应的寄存器 INTERRUPT_CORE0_SOURCE_X_MAP_REG 配成 Num_P，即可将该中断源分配至序号为 Num_P 的 CPU 中断 (Interrupt_P)。

8.3.3.2 分配多个外部中断源 Source_Xn 至 CPU 外部中断

将各个中断源对应的寄存器 INTERRUPT_CORE0_SOURCE_Xn_MAP_REG 均配置成相同的 Num_P，即可将多个中断源 Source_Xn 分配至同一 CPU 外部中断 Interrupt_P。上述任一外设中断均会触发 CPU 外部中断 Interrupt_P。待中断触发后，CPU 需查询中断状态寄存器，判断产生中断的外设。更多信息，见章节 1 *ESP-RISC-V CPU*。

8.3.3.3 关闭 CPU 外部中断源 Source_X

将中断源对应的寄存器 INTERRUPT_CORE0_SOURCE_X_MAP_REG 配置成 0，即可关闭外部中断源。

8.3.4 查询外部中断源当前的中断状态

读取寄存器 INTERRUPT_CORE0_INTR_STATUS_n_REG（只读）中特定位的值可以获取 CPU 外部中断源当前的中断状态。寄存器 INTERRUPT_CORE0_INTR_STATUS_n_REG 与外部中断源的对应关系如表 8-1 所示。

8.4 寄存器列表

本小节的所有地址均为相对于中断矩阵基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
中断源映射寄存器			
INTERRUPT_CORE0_PWR_INTR_MAP_REG	PWR_INTR 中断源映射寄存器	0x0008	R/W
INTERRUPT_CORE0_I2C_MST_INT_MAP_REG	I2C_MST_INT 中断源映射寄存器	0x002C	R/W
INTERRUPT_CORE0_SLC0_INTR_MAP_REG	SLC0_INTR 中断源映射寄存器	0x0030	R/W
INTERRUPT_CORE0_SLC1_INTR_MAP_REG	SLC1_INTR 中断源映射寄存器	0x0034	R/W
INTERRUPT_CORE0_APB_CTRL_INTR_MAP_REG	APB_CTRL_INTR 中断源映射寄存器	0x0038	R/W
INTERRUPT_CORE0_UHCI0_INTR_MAP_REG	UHCI0_INTR 中断源映射寄存器	0x003C	R/W
INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_MAP_REG	GPIO_INTERRUPT_PRO 中断源映射寄存器	0x0040	R/W
INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_NMI_MAP_REG	GPIO_INTERRUPT_PRO_NMI 中断源映射寄存器	0x0044	R/W
INTERRUPT_CORE0_SPI_INTR_1_MAP_REG	SPI_INTR_1 中断源映射寄存器	0x0048	R/W
INTERRUPT_CORE0_SPI_INTR_2_MAP_REG	SPI_INTR_2 中断源映射寄存器	0x004C	R/W
INTERRUPT_CORE0_I2S1_INT_MAP_REG	I2S1_INT 中断源映射寄存器	0x0050	R/W
INTERRUPT_CORE0_UART_INTR_MAP_REG	UART_INTR 中断源映射寄存器	0x0054	R/W
INTERRUPT_CORE0_UART1_INTR_MAP_REG	UART1_INTR 中断源映射寄存器	0x0058	R/W
INTERRUPT_CORE0_LEDC_INT_MAP_REG	LEDC_INT 中断源映射寄存器	0x005C	R/W
INTERRUPT_CORE0_EFUSE_INT_MAP_REG	EFUSE_INT 中断源映射寄存器	0x0060	R/W
INTERRUPT_CORE0_CAN_INT_MAP_REG	CAN_INT 中断源映射寄存器	0x0064	R/W
INTERRUPT_CORE0_USB_INTR_MAP_REG	USB_INTR 中断源映射寄存器	0x0068	R/W
INTERRUPT_CORE0_RTC_CORE_INTR_MAP_REG	RTC_CORE_INTR 中断源映射寄存器	0x006C	R/W
INTERRUPT_CORE0_RMT_INTR_MAP_REG	RMT_INTR 中断源映射寄存器	0x0070	R/W
INTERRUPT_CORE0_I2C_EXT0_INTR_MAP_REG	I2C_EXT0 intr 中断源映射寄存器	0x0074	R/W
INTERRUPT_CORE0_TIMER_INT1_MAP_REG	TIMER_INT1 中断源映射寄存器	0x0078	R/W
INTERRUPT_CORE0_TIMER_INT2_MAP_REG	TIMER_INT2 中断源映射寄存器	0x007C	R/W
INTERRUPT_CORE0_TG_T0_INT_MAP_REG	TG_T0_INT 中断源映射寄存器	0x0080	R/W
INTERRUPT_CORE0_TG_WDT_INT_MAP_REG	TG_WDT_INT 中断源映射寄存器	0x0084	R/W
INTERRUPT_CORE0_TG1_T0_INT_MAP_REG	TG1_T0_INT 中断源映射寄存器	0x0088	R/W

名称	描述	地址	访问
INTERRUPT_CORE0_TG1_WDT_INT_MAP_REG	TG1_WDT_INT 中断源映射寄存器	0x008C	R/W
INTERRUPT_CORE0_CACHE_IA_INT_MAP_REG	CACHE_IA_INT 中断源映射寄存器	0x0090	R/W
INTERRUPT_CORE0_SYSTIMER_TARGET0_INT_MAP_REG	SYSTIMER_TARGET0_INT 中断源映射寄存器	0x0094	R/W
INTERRUPT_CORE0_SYSTIMER_TARGET1_INT_MAP_REG	SYSTIMER_TARGET1_INT 中断源映射寄存器	0x0098	R/W
INTERRUPT_CORE0_SYSTIMER_TARGET2_INT_MAP_REG	SYSTIMER_TARGET2_INT 中断源映射寄存器	0x009C	R/W
INTERRUPT_CORE0_SPI_MEM_REJECT_INTR_MAP_REG	SPI_MEM_REJECT_INTR 中断源映射寄存器	0x00A0	R/W
INTERRUPT_CORE0_ICACHE_PRELOAD_INT_MAP_REG	ICACHE_PRELOAD_INT 中断源映射寄存器	0x00A4	R/W
INTERRUPT_CORE0_ICACHE_SYNC_INT_MAP_REG	ICACHE_SYNC_INT 中断源映射寄存器	0x00A8	R/W
INTERRUPT_CORE0_APB_ADC_INT_MAP_REG	APB_ADC_INT 中断源映射寄存器	0x00AC	R/W
INTERRUPT_CORE0_DMA_CH0_INT_MAP_REG	DMA_CH0_INT 中断源映射寄存器	0x00B0	R/W
INTERRUPT_CORE0_DMA_CH1_INT_MAP_REG	DMA_CH1_INT 中断源映射寄存器	0x00B4	R/W
INTERRUPT_CORE0_DMA_CH2_INT_MAP_REG	DMA_CH2_INT 中断源映射寄存器	0x00B8	R/W
INTERRUPT_CORE0_RSA_INT_MAP_REG	RSA_INT 中断源映射寄存器	0x00BC	R/W
INTERRUPT_CORE0_AES_INT_MAP_REG	AES_INT 中断源映射寄存器	0x00C0	R/W
INTERRUPT_CORE0_SHA_INT_MAP_REG	SHA_INT 中断源映射寄存器	0x00C4	R/W
INTERRUPT_CORE0_CPU_INTR_FROM_CPU_0_MAP_REG	CPU_INTR_FROM_CPU_0 中断源映射寄存器	0x00C8	R/W
INTERRUPT_CORE0_CPU_INTR_FROM_CPU_1_MAP_REG	CPU_INTR_FROM_CPU_1 中断源映射寄存器	0x00CC	R/W
INTERRUPT_CORE0_CPU_INTR_FROM_CPU_2_MAP_REG	CPU_INTR_FROM_CPU_2 中断源映射寄存器	0x00D0	R/W
INTERRUPT_CORE0_CPU_INTR_FROM_CPU_3_MAP_REG	CPU_INTR_FROM_CPU_3 intr 中断源映射寄存器	0x00D4	R/W
INTERRUPT_CORE0_ASSIST_DEBUG_INTR_MAP_REG	ASSIST_DEBUG_INTR 中断源映射寄存器	0x00D8	R/W
INTERRUPT_CORE0_DMA_APBPERI_PMS_MONITOR_VIOLATE_INTR_MAP_REG	DMA_APBPERI_PMS_MONITOR_VIOLATE 中断源映射寄存器	0x00DC	R/W
INTERRUPT_CORE0_CORE_0_IRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG	IRAM0_PMS_MONITOR_VIOLATE 中断源映射寄存器	0x00E0	R/W
INTERRUPT_CORE0_CORE_0_DRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG	DRAM0_PMS_MONITOR_VIOLATE 中断源映射寄存器	0x00E4	R/W
INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_INTR_MAP_REG	PIF_PMS_MONITOR_VIOLATE 中断源映射寄存器	0x00E8	R/W

名称	描述	地址	访问
INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_SIZE_INTR_MAP_REG	PIF_PMS_MONITOR_VIOLATE_SIZE 中断源映射寄存器	0x00EC	R/W
INTERRUPT_CORE0_BACKUP_PMS_VIOLATE_INTR_MAP_REG	BACKUP_PMS_VIOLATE 中断源映射寄存器	0x00F0	R/W
INTERRUPT_CORE0_CACHE_CORE0_ACS_INT_MAP_REG	CACHE_CORE0_ACS 中断源映射寄存器	0x00F4	R/W
中断源状态寄存器			
INTERRUPT_CORE0_INTR_STATUS_0_REG	中断源状态寄存器 0	0x00F8	RO
INTERRUPT_CORE0_INTR_STATUS_1_REG	中断源状态寄存器 1	0x00FC	RO
时钟寄存器			
INTERRUPT_CORE0_CLOCK_GATE_REG	时钟寄存器	0x0100	R/W
CPU 中断寄存器			
INTERRUPT_CORE0_CPU_INT_ENABLE_REG	CPU 中断使能配置寄存器	0x0104	R/W
INTERRUPT_CORE0_CPU_INT_TYPE_REG	CPU 中断类型配置寄存器	0x0108	R/W
INTERRUPT_CORE0_CPU_INT_CLEAR_REG	CPU 中断清零寄存器	0x010C	R/W
INTERRUPT_CORE0_CPU_INT_EIP_STATUS_REG	CPU 中断阻塞状态寄存器	0x0110	RO
INTERRUPT_CORE0_CPU_INT_PRI_1_REG	CPU 中断优先级配置寄存器 1	0x0118	R/W
INTERRUPT_CORE0_CPU_INT_PRI_2_REG	CPU 中断优先级配置寄存器 2	0x011C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_3_REG	CPU 中断优先级配置寄存器 3	0x0120	R/W
INTERRUPT_CORE0_CPU_INT_PRI_4_REG	CPU 中断优先级配置寄存器 4	0x0124	R/W
INTERRUPT_CORE0_CPU_INT_PRI_5_REG	CPU 中断优先级配置寄存器 5	0x0128	R/W
INTERRUPT_CORE0_CPU_INT_PRI_6_REG	CPU 中断优先级配置寄存器 6	0x012C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_7_REG	CPU 中断优先级配置寄存器 7	0x0130	R/W
INTERRUPT_CORE0_CPU_INT_PRI_8_REG	CPU 中断优先级配置寄存器 8	0x0134	R/W
INTERRUPT_CORE0_CPU_INT_PRI_9_REG	CPU 中断优先级配置寄存器 9	0x0138	R/W
INTERRUPT_CORE0_CPU_INT_PRI_10_REG	CPU 中断优先级配置寄存器 10	0x013C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_11_REG	CPU 中断优先级配置寄存器 11	0x0140	R/W
INTERRUPT_CORE0_CPU_INT_PRI_12_REG	CPU 中断优先级配置寄存器 12	0x0144	R/W
INTERRUPT_CORE0_CPU_INT_PRI_13_REG	CPU 中断优先级配置寄存器 13	0x0148	R/W
INTERRUPT_CORE0_CPU_INT_PRI_14_REG	CPU 中断优先级配置寄存器 14	0x014C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_15_REG	CPU 中断优先级配置寄存器 15	0x0150	R/W

名称	描述	地址	访问
INTERRUPT_CORE0_CPU_INT_PRI_16_REG	CPU 中断优先级配置寄存器 16	0x0154	R/W
INTERRUPT_CORE0_CPU_INT_PRI_17_REG	CPU 中断优先级配置寄存器 17	0x0158	R/W
INTERRUPT_CORE0_CPU_INT_PRI_18_REG	CPU 中断优先级配置寄存器 18	0x015C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_19_REG	CPU 中断优先级配置寄存器 19	0x0160	R/W
INTERRUPT_CORE0_CPU_INT_PRI_20_REG	CPU 中断优先级配置寄存器 20	0x0164	R/W
INTERRUPT_CORE0_CPU_INT_PRI_21_REG	CPU 中断优先级配置寄存器 21	0x0168	R/W
INTERRUPT_CORE0_CPU_INT_PRI_22_REG	CPU 中断优先级配置寄存器 22	0x016C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_23_REG	CPU 中断优先级配置寄存器 23	0x0170	R/W
INTERRUPT_CORE0_CPU_INT_PRI_24_REG	CPU 中断优先级配置寄存器 24	0x0174	R/W
INTERRUPT_CORE0_CPU_INT_PRI_25_REG	CPU 中断优先级配置寄存器 25	0x0178	R/W
INTERRUPT_CORE0_CPU_INT_PRI_26_REG	CPU 中断优先级配置寄存器 26	0x017C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_27_REG	CPU 中断优先级配置寄存器 27	0x0180	R/W
INTERRUPT_CORE0_CPU_INT_PRI_28_REG	CPU 中断优先级配置寄存器 28	0x0184	R/W
INTERRUPT_CORE0_CPU_INT_PRI_29_REG	CPU 中断优先级配置寄存器 29	0x0188	R/W
INTERRUPT_CORE0_CPU_INT_PRI_30_REG	CPU 中断优先级配置寄存器 30	0x018C	R/W
INTERRUPT_CORE0_CPU_INT_PRI_31_REG	CPU 中断优先级配置寄存器 31	0x0190	R/W
INTERRUPT_CORE0_CPU_INT_THRESH_REG	CPU 中断阈值配置寄存器	0x0194	R/W
版本寄存器			
INTERRUPT_CORE0_INTERRUPT_DATE_REG	版本控制寄存器	0x07FC	R/W

8.5 寄存器

本小节的所有地址均为相对于中断矩阵基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 8.1. INTERRUPT_CORE0_PWR_INTR_MAP_REG (0x0008)

Register 8.2. INTERRUPT_CORE0_I2C_MST_INT_MAP_REG (0x002C)

Register 8.3. INTERRUPT_CORE0_SLC0_INTR_MAP_REG (0x0030)

Register 8.4. INTERRUPT_CORE0_SLC1_INTR_MAP_REG (0x0034)

Register 8.5. INTERRUPT_CORE0_APB_CTRL_INTR_MAP_REG (0x0038)

Register 8.6. INTERRUPT_CORE0_UHCIO_INTR_MAP_REG (0x003C)

Register 8.7. INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_MAP_REG (0x0040)

Register 8.8. INTERRUPT_CORE0_GPIO_INTERRUPT_PRO_NMI_MAP_REG (0x0044)

Register 8.9. INTERRUPT_CORE0_SPI_INTR_1_MAP_REG (0x0048)

Register 8.10. INTERRUPT_CORE0_SPI_INTR_2_MAP_REG (0x004C)

Register 8.11. INTERRUPT_CORE0_I2S1_INT_MAP_REG (0x0050)

Register 8.12. INTERRUPT_CORE0_UART_INTR_MAP_REG (0x0054)

Register 8.13. INTERRUPT_CORE0_UART1_INTR_MAP_REG (0x0058)

Register 8.14. INTERRUPT_CORE0_LEDC_INT_MAP_REG (0x005C)

Register 8.15. INTERRUPT_CORE0_EFUSE_INT_MAP_REG (0x0060)

Register 8.16. INTERRUPT_CORE0_CAN_INT_MAP_REG (0x0064)

Register 8.17. INTERRUPT_CORE0_USB_INTR_MAP_REG (0x0068)

Register 8.18. INTERRUPT_CORE0_RTC_CORE_INTR_MAP_REG (0x006C)

Register 8.19. INTERRUPT_CORE0_RMT_INTR_MAP_REG (0x0070)

Register 8.20. INTERRUPT_CORE0_I2C_EXT0_INTR_MAP_REG (0x0074)

Register 8.21. INTERRUPT_CORE0_TIMER_INT1_MAP_REG (0x0078)

Register 8.22. INTERRUPT_CORE0_TIMER_INT2_MAP_REG (0x007C)

Register 8.23. INTERRUPT_CORE0_TG_T0_INT_MAP_REG (0x0080)

Register 8.24. INTERRUPT_CORE0_TG_WDT_INT_MAP_REG (0x0084)

Register 8.25. INTERRUPT_CORE0_TG1_T0_INT_MAP_REG (0x0088)

Register 8.26. INTERRUPT_CORE0_TG1_WDT_INT_MAP_REG (0x008C)

Register 8.27. INTERRUPT_CORE0_CACHE_IA_INT_MAP_REG (0x0090)

Register 8.28. INTERRUPT_CORE0_SYSTIMER_TARGET0_INT_MAP_REG (0x0094)

Register 8.29. INTERRUPT_CORE0_SYSTIMER_TARGET1_INT_MAP_REG (0x0098)

Register 8.30. INTERRUPT_CORE0_SYSTIMER_TARGET2_INT_MAP_REG (0x009C)

Register 8.31. INTERRUPT_CORE0_SPI_MEM_REJECT_INTR_MAP_REG (0x00A0)

Register 8.32. INTERRUPT_CORE0_ICACHE_PRELOAD_INT_MAP_REG (0x00A4)

Register 8.33. INTERRUPT_CORE0_ICACHE_SYNC_INT_MAP_REG (0x00A8)

Register 8.34. INTERRUPT_CORE0_APB_ADC_INT_MAP_REG (0x00AC)

Register 8.35. INTERRUPT_CORE0_DMA_CH0_INT_MAP_REG (0x00B0)

Register 8.36. INTERRUPT_CORE0_DMA_CH1_INT_MAP_REG (0x00B4)

Register 8.37. INTERRUPT_CORE0_DMA_CH2_INT_MAP_REG (0x00B8)

Register 8.38. INTERRUPT_CORE0_RSA_INT_MAP_REG (0x00BC)

Register 8.39. INTERRUPT_CORE0_AES_INT_MAP_REG (0x00C0)

Register 8.40. INTERRUPT_CORE0_SHA_INT_MAP_REG (0x00C4)

Register 8.41. INTERRUPT_CORE0_CPU_INTR_FROM_CPU_0_MAP_REG (0x00C8)

Register 8.42. INTERRUPT_CORE0_CPU_INTR_FROM_CPU_1_MAP_REG (0x00CC)

Register 8.43. INTERRUPT_CORE0_CPU_INTR_FROM_CPU_2_MAP_REG (0x00D0)

Register 8.44. INTERRUPT_CORE0_CPU_INTR_FROM_CPU_3_MAP_REG (0x00D4)

Register 8.45. INTERRUPT_CORE0_ASSIST_DEBUG_INTR_MAP_REG (0x00D8)

Register 8.46. INTERRUPT_CORE0_DMA_APBPERI_PMS_MONITOR_VIOLATE_INTR_MAP_REG (0x00DC)

Register 8.47. INTERRUPT_CORE0_CORE_0_IRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG (0x00E0)

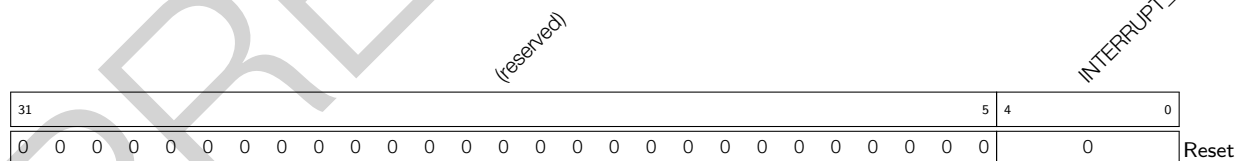
Register 8.48. INTERRUPT_CORE0_CORE_0_DRAM0_PMS_MONITOR_VIOLATE_INTR_MAP_REG (0x00E4)

Register 8.49. INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_INTR_MAP_REG (0x00E8)

Register 8.50. INTERRUPT_CORE0_CORE_0_PIF_PMS_MONITOR_VIOLATE_SIZE_INTR_MAP_REG (0x00EC)

Register 8.51. INTERRUPT_CORE0_BACKUP_PMS_VIOLATE_INTR_MAP_REG (0x00F0)

Register 8.52. INTERRUPT_CORE0_CACHE_CORE0_ACS_INT_MAP_REG (0x00F4)



INTERRUPT_CORE0_SOURCE_X_MAP 将中断源 SOURCE_X 映射至 CPU 外部中断。中断源 SOURCE_X 见表 8-1。(R/W)

Register 8.53. INTERRUPT_CORE0_INTR_STATUS_0_REG (0x00F8)

INTERRUPT_CORE0_INTR_STATUS_0																															
31																															0
0x000000																															
Reset																															

INTERRUPT_CORE0_INTR_STATUS_0 用于存储外部中断源的状态，每一位均代表一个外部中断源的状态，对应中断编号源：0 ~ 31。如果对应的位为 1，则表示该中断源触发了中断。(RO)

Register 8.54. INTERRUPT_CORE0_INTR_STATUS_1_REG (0x00FC)

INTERRUPT_CORE0_INTR_STATUS_1																															
31																															0
0x000000																															
Reset																															

INTERRUPT_CORE0_INTR_STATUS_1 用于存储外部中断源的状态，每一位均代表一个外部中断源的状态，对应中断编号源：32 ~ 61。如果对应的位为 1，则表示该中断源触发了中断。(RO)

Register 8.55. INTERRUPT_CORE0_CLOCK_GATE_REG (0x0100)

(reserved)																															INTERRUPT		
31																															1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

INTERRUPT_CORE0_CLK_EN 置 1 强制使能中断寄存器的时钟门控。(R/W)

Register 8.56. INTERRUPT_CORE0_CPU_INT_ENABLE_REG (0x0104)

31																																0
0																																Reset

INTERRUPT_CORE0_CPU_INT_ENABLE 在相应位写 1，即可使能对应 CPU 中断。(R/W)

Register 8.57. INTERRUPT_CORE0_CPU_INT_TYPE_REG (0x0108)

31																																0																																															
0																																																																Reset															

INTERRUPT_CORE0_CPU_INT_TYPE 配置 CPU 中断类型。0：电平触发；1：边沿触发。(R/W)

Register 8.58. INTERRUPT_CORE0_CPU_INT_CLEAR_REG (0x010C)

31																																0																															
0																																																															
Reset																																																															

INTERRUPT_CORE0_CPU_INT_CLEAR 在相应位写 1，即可清除对应的 CPU 中断。(R/W)

Register 8.59. INTERRUPT_CORE0_CPU_INT_EIP_STATUS_REG (0x0110)

INTERRUPT_CORE0_CPU_INT_EIP_STATUS																															
31																															0
0																															
Reset																															

INTERRUPT_CORE0_CPU_INT_EIP_STATUS 用于存储 CPU 中断的阻塞状态。更多信息，请参考章节 1 *ESP-RISC-V CPU*。(RO)

Register 8.60. INTERRUPT_CORE0_CPU_INT_PRI_n_REG (n: 1 - 31)(0x0118 + 0x4*n)

INTERRUPT_CORE0_CPU_INT_PRI_n_MAP																													
31	(reserved)																												0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

INTERRUPT_CORE0_CPU_PRI_n_MAP 用于设置 CPU 中断 *n* 的优先级，可配置为 1 ~ 15。数字越大，优先级越高。更多配置信息，见章节 1 *ESP-RISC-V CPU*。(R/W)

Register 8.61. INTERRUPT_CORE0_CPU_INT_THRESH_REG (0x0194)

INTERRUPT_CORE0_CPU_INT_THRESH																													
31	(reserved)																												0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

INTERRUPT_CORE0_CPU_INT_THRESH 用于设置 CPU 中断阈值。仅当中断的优先级等于或高于该阈值，CPU 才会响应该中断。(R/W)

Register 8.62. INTERRUPT_CORE0_INTERRUPT_DATE_REG (0x07FC)

(reserved)				INTERRUPT_CORE0_INTERRUPT_DATE																									
31	28	27																											0
0	0	0	0																										Reset

INTERRUPT_CORE0_INTERRUPT_DATE 版本寄存器。(R/W)

9 系统定时器 (SYSTIMER)

9.1 概述

ESP32-C3 芯片内置一组 52 位系统定时器。该定时器可用于生成操作系统所需的滴答定时中断，也可以用作普通的定时器生成周期或单次延时中断。在 RTC 定时器的协助下，系统定时器可在芯片从 Deep-sleep 或 Light-sleep 唤醒后补偿睡眠时间。

系统定时器内置两个计数器（UNIT0 和 UNIT1）以及三个比较器（COMP0、COMP1 和 COMP2）。比较器用于监控计数器的计数值是否达到报警值。定时器的功能块图见图 9-1。

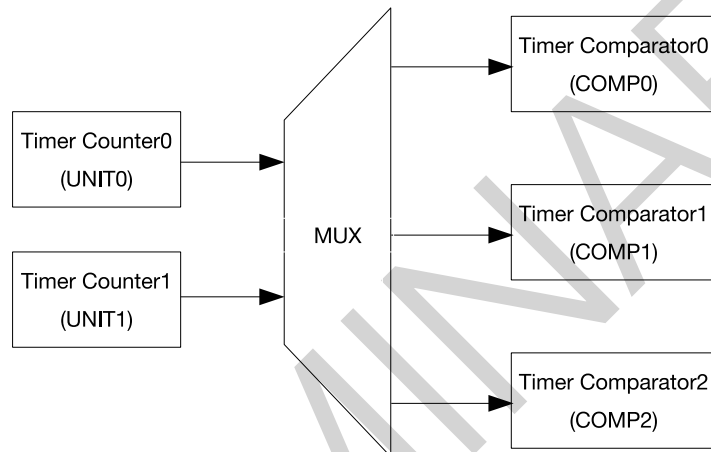


图 9-1. 系统定时器结构图

9.2 主要特性

- 由两个 52 位计数器和三个 52 位比较器组成
- 软件通过 APB_CLK 访问寄存器
- 计数采用 CNT_CLK 时钟，两次计数周期的平均频率为 16 MHz
- CNT_CLK 的时钟源为 XTAL_CLK (40 MHz)
- 支持 52 位报警值 (t) 和 26 位报警周期 (δt)
- 支持两种报警模式：
 - 单次报警模式：根据设定的目标报警值 (t)，生成一次性报警
 - 周期报警模式：根据设定的报警周期 (δt)，生成周期性报警
- 三个比较器可根据设置的报警值 (t) 或报警周期 (δt) 生成三个独立中断
- 芯片从 Deep-sleep 或 Light-sleep 唤醒之后，系统定时器可以通过软件加载 RTC 定时器记录的睡眠时间，然后进行补偿。
- CPU 处于停止状态或处于在线调试状态时，系统定时器可选择停止运行或继续运行。

9.3 时钟源选择

计数器和比较器使用 XTAL_CLK 用作时钟源。XTAL_CLK 经分数分频后, 在一个计数周期生成频率为 $f_{XTAL_CLK}/3$ 的时钟信号, 然后在另一个计数周期生成频率为 $f_{XTAL_CLK}/2$ 的时钟信号。因此, 计数器使用的时钟 CNT_CLK, 其实际平均频率为 $f_{XTAL_CLK}/2.5$, 即 16 MHz, 见图 9-2。每个 CNT_CLK 时钟周期, 计数递增 $1/16 \mu s$, 即 16 个周期递增 $1 \mu s$ 。

配置寄存器等软件操作则是由 APB_CLK 提供时钟信号。更多有关 APB_CLK 的信息, 见章节 6 复位和时钟。

用户可使用以下系统寄存器的相关位来控制系统定时器:

- 置位寄存器 SYSTEM_PERIP_CLK_EN0_REG 中 SYSTEM_SYSTIMER_CLK_EN 位使能系统定时器的 APB_CLK 信号;
- 置位寄存器 SYSTEM_PERIP_CLK_EN0_REG 中 SYSTEM_SYSTIMER_RST 位, 复位系统定时器。

注意, 复位后, 系统定时器的寄存器将恢复到默认值。更多信息可参考章节 13 系统寄存器 (SYSREG) 中表: 外设时钟门控与复位控制位。

9.4 功能描述

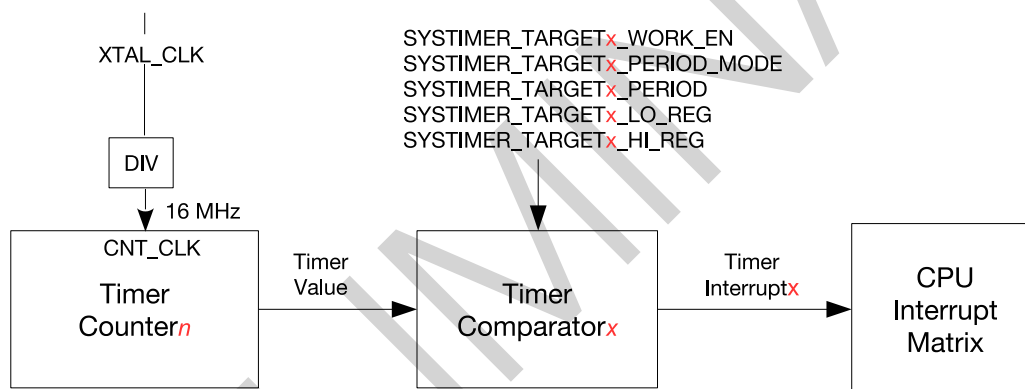


图 9-2. 系统定时器生成报警

图 9-2 展示了系统定时器生成报警的过程。在上述过程中用到一个计数器和一个比较器, 比较器将根据比较结果, 生成报警中断。

9.4.1 计数器

系统定时器提供两个 52 位计数器, 下文用 UNIT n 表示, n 可以取 0 或 1。计数器使用 16 MHz CNT_CLK 作为计数时钟。用户可通过配置寄存器 SYSTIMER_CONF_REG 中下面两个位来控制计数器 UNIT n :

- SYSTIMER_TIMER_UNIT n _WORK_EN: 置位此位, 使能计数器 UNIT n ;
- SYSTIMER_TIMER_UNIT n _CORE0_STALL_EN: 置位此位, CPU 停止运行后, 计数器 UNIT n 也将停止工作。CPU 恢复运行后, 计数器重新开始工作。

UNIT n 的具体配置见下表，其中假设 CPU 当前状态为停止工作。

表 9-1. UNIT n 配置控制位

SYSTIMER_TIMER_UNIT n _WORK_EN	SYSTIMER_TIMER_UNIT n _CORE0_STALL_EN	计数器 UNIT n
0	x [*]	未处于工作状态。
1	1	暂停计数，但 CPU 苏醒后，会继续计数。
1	0	不受影响，照常计数。

^{*} x: 无关项

计数器 UNIT n 处于工作状态时，计数值按计数周期递增。UNIT n 停止工作，则计数值将保持不变，不再递增。

计数起始值的低 32 位和高 20 位分别从 SYSTIMER_TIMER_UNIT n _LOAD_LO 和 SYSTIMER_TIMER_UNIT n _LOAD_HI 装载。置位 SYSTIMER_TIMER_UNIT n _LOAD 将触发重装载事件，当前计数起始值立即更新。如果计数器 UNIT n 处于工作状态，则将从新装载的计数值开始计数。

置位 SYSTIMER_TIMER_UNIT n _UPDATE 将触发更新事件，当前计数值的低 32 位和高 20 位被锁存至寄存器 SYSTIMER_TIMER_UNIT n _VALUE_LO 和 SYSTIMER_TIMER_UNIT n _VALUE_HI 后，SYSTIMER_TIMER_UNIT n _VALUE_VALID 会被置起。SYSTIMER_TIMER_UNIT n _VALUE_LO 和 SYSTIMER_TIMER_UNIT n _VALUE_HI 寄存器中的值保持不变，直至下次更新事件发生。

9.4.2 比较器和报警

系统定时器有三个 52 位比较器，用 COMP x 表示，其中 x 可以取 0、1、2。比较器可根据设置的不同报警值 (t) 或报警周期 (δt)，触发不同的中断。

用户可配置寄存器 SYSTIMER_TARGET x _PERIOD_MODE 选择比较器 COMP x 生成报警的模式：

- 1：选择周期报警模式
- 0：选择单次报警模式

选择周期报警模式时，寄存器 SYSTIMER_TARGET x _PERIOD 中的值为报警周期 (δt)。假设当前计数值为 t_1 ，经过一段时间，当计数值达到 $t_1 + \delta t$ 时，将触发一次报警中断。再经过一段时间，当计数值达到 $t_1 + 2 * \delta t$ 时，将再次触发一次报警中断，以此类推。通过上述方式即可实现周期性报警。

选择单次报警模式时，SYSTIMER_TIMER_TARGET x _LO 和 SYSTIMER_TIMER_TARGET x _HI 分别提供报警值 (t) 的低 32 位和高 20 位。假设当前计数值为 t_2 ($t_2 \leq t$)，经过一段时间，当计数到报警值 (t) 时，则触发一次报警。与周期报警模式不同，单次报警模式仅生成一次报警中断。

用户可配置寄存器 SYSTIMER_TARGET x _TIMER_UNIT_SEL 选择用于与 COMP x 进行比较的计数器值，然后生成报警：

- 1：选择与计数器 UNIT 1 的计数值进行比较
- 0：选择与计数器 UNIT 0 的计数值进行比较

置位 SYSTIMER_TARGET x _WORK_EN，COMP x 开始进行比较：

- 在单次报警模式下，COMP x 将比较计数器中的实际计数值与寄存器中设置的报警值 (t)；
- 在周期报警模式下，COMP x 将比较计数器中的实际计数值与 $t_1 + n * \delta t$ ($n = 1, 2, 3, \dots$)。

实际计数值等于报警值 (t)，或等于 $t_1 + n \cdot \delta t$ ($n = 1, 2, 3, \dots$)，则触发一次报警中断。但如果设定的报警值 (t) 小于当前计数值，即报警值 (t) 已成为过去，或当前计数值超过设定的报警值 (t) 一定范围 ($0 \sim 2^{51} - 1$)，则也将立即触发中断。当前计数值 t_c 、报警值 t_t 和触发报警的关系如下表所示：

表 9-2. 报警触发条件

t_c 与 t_t 的关系	触发条件
$t_c - t_t \leq 0$	当 $t_c = t_t$ 时，触发报警
$0 \leq t_c - t_t < 2^{51} - 1$	立即触发报警
$t_c - t_t \geq 2^{51} - 1$	t_c 达到最大值 $52'h\text{ffffffffffff}$ 后溢出，然后从 0 开始计数，计数达到 t_t 时触发报警

9.4.3 同步操作

软件操作与计数器和比较器工作在不同时钟频率下，因此需要对部分配置寄存器进行同步。完整的同步过程包括下面两个步骤：

1. 通过软件向配置寄存器写入合适的值，见表 9-3 第一列；
2. 通过软件置位相应的同步使能位，开始同步操作，见表 9-3 第二列。

表 9-3. 同步操作

需要同步的字段	同步使能位
SYSTIMER_TIMER_UNIT n _LOAD_LO SYSTIMER_TIMER_UNIT n _LOAD_HI	SYSTIMER_TIMER_UNIT n _LOAD
SYSTIMER_TARGET x _PERIOD SYSTIMER_TIMER_TARGET x _HI SYSTIMER_TIMER_TARGET x _LO	SYSTIMER_TIMER_COMP x _LOAD

9.4.4 中断

上述三个比较器均有一个对应的报警中断，即 SYSTIMER_TARGET x _INT 中断，该中断为电平类型中断。比较器开始触发报警，即拉高中断信号。中断信号将一直保持高电平，直至软件清除中断。用户可置位 SYSTIMER_TARGET x _INT_ENA 使能中断。

9.5 编程示例

9.5.1 读取当前计数器的值

1. 置位 SYSTIMER_TIMER_UNIT n _UPDATE，将计数器 UNIT n 的值更新至寄存器 SYSTIMER_TIMER_UNIT n _VALUE_HI 和 SYSTIMER_TIMER_UNIT n _VALUE_LO；
2. 轮询 SYSTIMER_TIMER_UNIT n _VALUE_VALID，直至其值为 1。之后，用户可从寄存器 SYSTIMER_TIMER_UNIT n _VALUE_HI 和 SYSTIMER_TIMER_UNIT n _VALUE_LO 中读取计数器的值；
3. 读取寄存器 SYSTIMER_TIMER_UNIT n _VALUE_LO (低 32 位) 和 SYSTIMER_TIMER_UNIT n _VALUE_HI (高 20 位)。

9.5.2 在单次报警模式下配置一次性报警

1. 设置 `SYSTIMER_TARGETx_TIMER_UNIT_SEL` 选择与 `COMPx` 进行比较的计数器；
2. 读取当前计数器的值，步骤见章节 9.5.1。读取的当前值可用于计算步骤 4 中的报警值 (t)；
3. 清除 `SYSTIMER_TARGETx_PERIOD_MODE`，使能单次报警模式；
4. 设置报警值 (t)，并将报警值 (t) 的低 32 位和高 20 位分别写入 `SYSTIMER_TIMER_TARGETx_LO` 和 `SYSTIMER_TIMER_TARGETx_HI`；
5. 置位 `SYSTIMER_TIMER_COMPx_LOAD`，同步报警值 (t)，即将报警值 (t) 装载至比较器 `COMPx`；
6. 置位 `SYSTIMER_TARGETx_WORK_EN` 使能选择的比较器 `COMPx`；比较器 `COMP` 开始比较计数值与报警值 (t)；
7. 置位 `SYSTIMER_TARGETx_INT_ENA`，使能中断。Unit_n 达到报警值 (t) 则触发一次报警中断 `SYSTIMER_TARGETx_INT`。

9.5.3 在周期报警模式下配置周期性报警

1. 设置 `SYSTIMER_TARGETx_TIMER_UNIT_SEL` 选择与 `COMPx` 进行比较的计数器；
2. 将报警周期 (δt) 写入 `SYSTIMER_TARGETx_PERIOD`；
3. 置位 `SYSTIMER_TIMER_COMPx_LOAD` 同步报警周期值，即将 (δt) 装载至比较器 `COMPx`；
4. 置位 `SYSTIMER_TARGETx_PERIOD_MODE` 将 `COMPx` 配置为周期报警模式；
5. 置位 `SYSTIMER_TARGETx_WORK_EN` 使能选择的比较器 `COMPx`；比较器 `COMPx` 开始将计数值与计数初始值 + $n \cdot \delta t$ ($n = 1, 2, 3 \dots$) 进行比较；
6. 置位 `SYSTIMER_TARGETx_INT_ENA`，使能中断。Unit_n 计数达到计数初始值 + $n \cdot \delta t$ ($n = 1, 2, 3 \dots$)，则触发一次 `SYSTIMER_TARGETx_INT` 中断。

9.5.4 唤醒后时间补偿

1. 在芯片进入 Deep-sleep 或 Light-sleep 之前，用户需配置 RTC 定时器用于精确记录睡眠时间，见低功耗管理章节；
2. 系统从睡眠模式唤醒后，读取 RTC 定时器记录的睡眠时间；
3. 读取当前系统定时器的计数值，见章节 9.5.1；
4. 将 RTC 记录的睡眠时间，单位：RTC_SLOW_CLK 周期，转换成以 CNT_CLK (16 MHz) 周期为单位的睡眠时间。例如，如果 RTC_SLOW_CLK 频率为 32 kHz，则 RTC 定时器记录的时间乘以 500 即可。
5. 将 RTC 定时器转换后的值加到系统定时器当前计数值：
 - 将计算所得值，低 32 位写入 `SYSTIMER_TIMER_UNITn_LOAD_LO`，高 20 位写入 `SYSTIMER_TIMER_UNITn_LOAD_HI`；
 - 置位 `SYSTIMER_TIMER_UNITn_LOAD`，将新的定时器值装载到系统定时器。这样即可完成系统定时器更新。

9.6 寄存器列表

本小节的所有地址均为相对于系统定时器基地址的地址偏移量（相对地址），具体基地址请见章节 3 [系统和存储器](#) 中的表 3-4。

名称	描述	地址	访问
时钟控制寄存器			
SYSTIMER_CONF_REG	配置系统定时器的时钟	0x0000	R/W
UNIT0 控制和配置寄存器			
SYSTIMER_UNIT0_OP_REG	读取 UNIT0 的值到相应寄存器	0x0004	varies
SYSTIMER_UNIT0_LOAD_HI_REG	待装载至 UNIT0 的值，高 20 位	0x000C	R/W
SYSTIMER_UNIT0_LOAD_LO_REG	待装载至 UNIT0 的值，低 32 位	0x0010	R/W
SYSTIMER_UNIT0_VALUE_HI_REG	UNIT0 的读值，高 20 位	0x0040	RO
SYSTIMER_UNIT0_VALUE_LO_REG	UNIT0 的读值，低 32 位	0x0044	RO
SYSTIMER_UNIT0_LOAD_REG	计数器 UNIT0 的装载同步寄存器	0x005C	WT
UNIT1 控制和配置寄存器			
SYSTIMER_UNIT1_OP_REG	读取计数器 UNIT1 的值	0x0008	varies
SYSTIMER_UNIT1_LOAD_HI_REG	待装载至计数器 UNIT1 的值，高 20 位	0x0014	R/W
SYSTIMER_UNIT1_LOAD_LO_REG	待装载至计数器 UNIT1 的值，低 32 位	0x0018	R/W
SYSTIMER_UNIT1_VALUE_HI_REG	计数器 UNIT1 的读值，高 20 位	0x0048	RO
SYSTIMER_UNIT1_VALUE_LO_REG	计数器 UNIT1 的读值，低 32 位	0x004C	RO
SYSTIMER_UNIT1_LOAD_REG	计数器 UNIT1 的装载同步寄存器	0x0060	WT
比较器 COMP0 的控制和配置寄存器			
SYSTIMER_TARGET0_HI_REG	待装载至比较器 COMP0 的报警值，高 20 位	0x001C	R/W
SYSTIMER_TARGET0_LO_REG	待装载至比较器 COMP0 的报警值，低 32 位	0x0020	R/W
SYSTIMER_TARGET0_CONF_REG	配置比较器 COMP0 的报警模式	0x0034	R/W
SYSTIMER_COMP0_LOAD_REG	比较器 COMP0 的装载同步寄存器	0x0050	WT
比较器 COMP1 的控制和配置寄存器			
SYSTIMER_TARGET1_HI_REG	待装载至比较器 COMP1 的报警值，高 20 位	0x0024	R/W
SYSTIMER_TARGET1_LO_REG	待装载至比较器 COMP1 的报警值，低 32 位	0x0028	R/W
SYSTIMER_TARGET1_CONF_REG	配置比较器 COMP1 的报警模式	0x0038	R/W
SYSTIMER_COMP1_LOAD_REG	比较器 COMP1 的装载同步寄存器	0x0054	WT
比较器 COMP2 的控制和配置寄存器			
SYSTIMER_TARGET2_HI_REG	待装载至比较器 COMP2 的报警值，高 20 位	0x002C	R/W
SYSTIMER_TARGET2_LO_REG	待装载至比较器 COMP2 的报警值，低 32 位	0x0030	R/W
SYSTIMER_TARGET2_CONF_REG	配置比较器 COMP2 的报警模式	0x003C	R/W
SYSTIMER_COMP2_LOAD_REG	比较器 COMP2 的装载同步寄存器	0x0058	WT
中断寄存器			
SYSTIMER_INT_ENA_REG	系统定时器的中断使能寄存器	0x0064	R/W
SYSTIMER_INT_RAW_REG	系统定时器的原始中断寄存器	0x0068	R/WTC/SS
SYSTIMER_INT_CLR_REG	系统定时器的中断清除寄存器	0x006C	WT
SYSTIMER_INT_ST_REG	系统定时器的中断状态寄存器	0x0070	RO
版本寄存器			
SYSTIMER_DATE_REG	版本控制寄存器	0x00FC	R/W

195

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

195

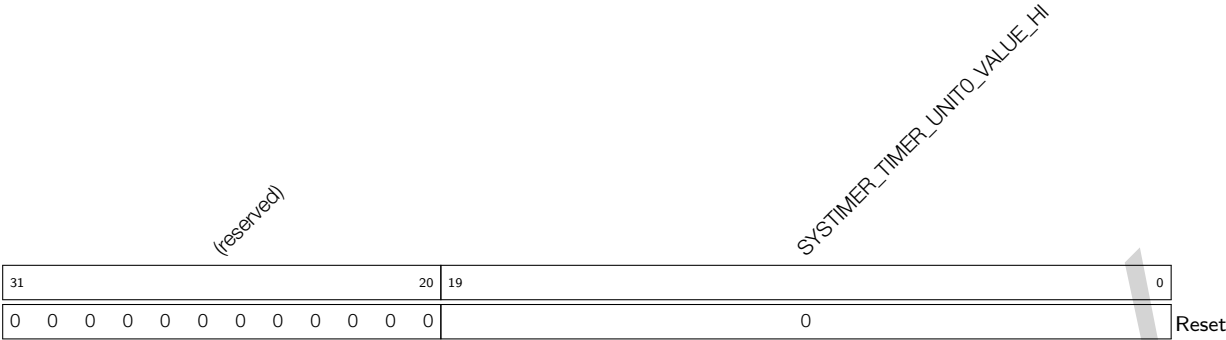
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

195

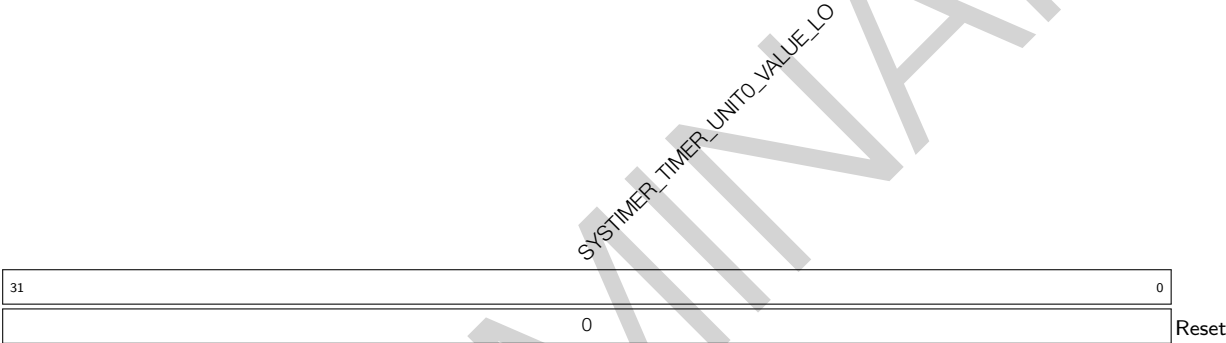
反馈文档意见

Register 9.5. SYSTIMER_UNIT0_VALUE_HI_REG (0x0040)



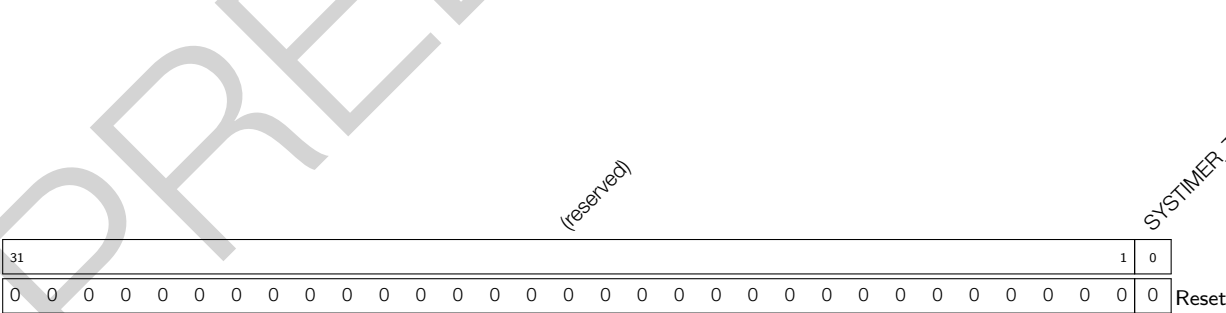
SYSTIMER_TIMER_UNIT0_VALUE_HI 计数器 UNIT0 的读数，高 20 位。(RO)

Register 9.6. SYSTIMER_UNIT0_VALUE_LO_REG (0x0044)



SYSTIMER_TIMER_UNIT0_VALUE_LO 计数器 UNIT0 的读数，低 32 位。(RO)

Register 9.7. SYSTIMER_UNIT0_LOAD_REG (0x005C)



SYSTIMER_TIMER_UNIT0_LOAD 计数器 UNIT0 的同步使能信号。置位此位，将重新装载寄存器 [SYSTIMER_TIMER_UNIT0_LOAD_HI](#) 和 [SYSTIMER_TIMER_UNIT0_LOAD_LO](#) 的值到计数器 UNIT0。(WT)

197

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

197

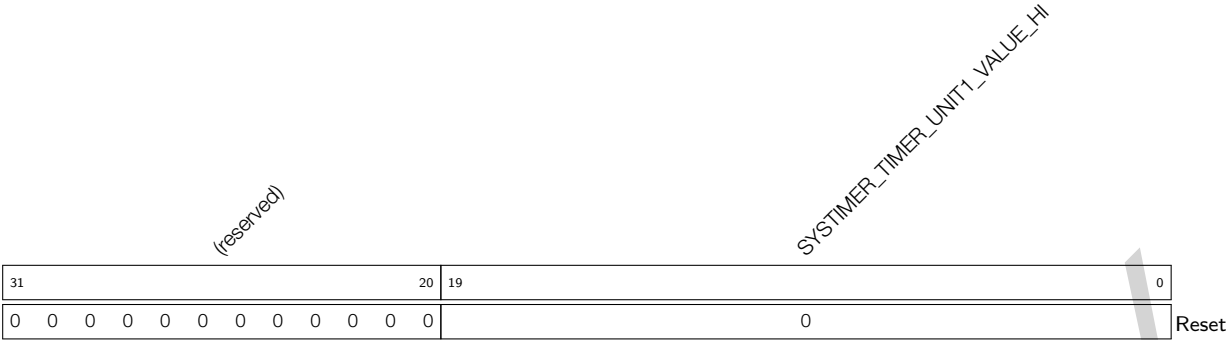
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

197

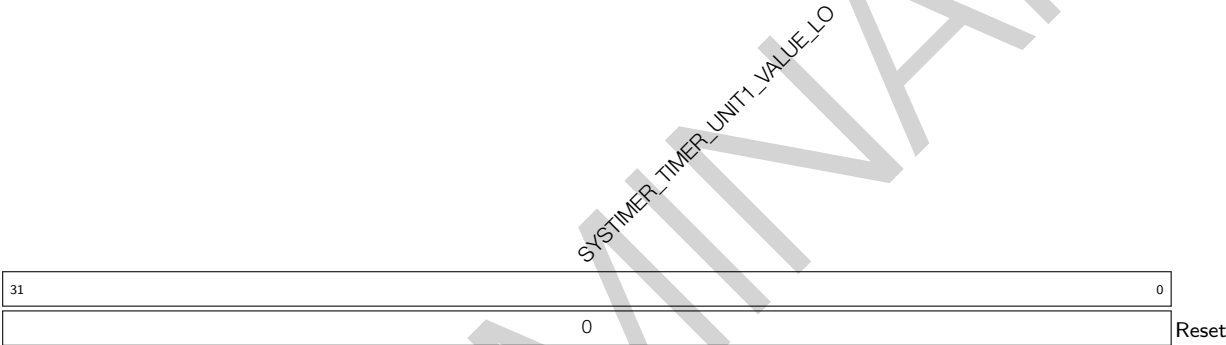
反馈文档意见

Register 9.11. SYSTIMER_UNIT1_VALUE_HI_REG (0x0048)



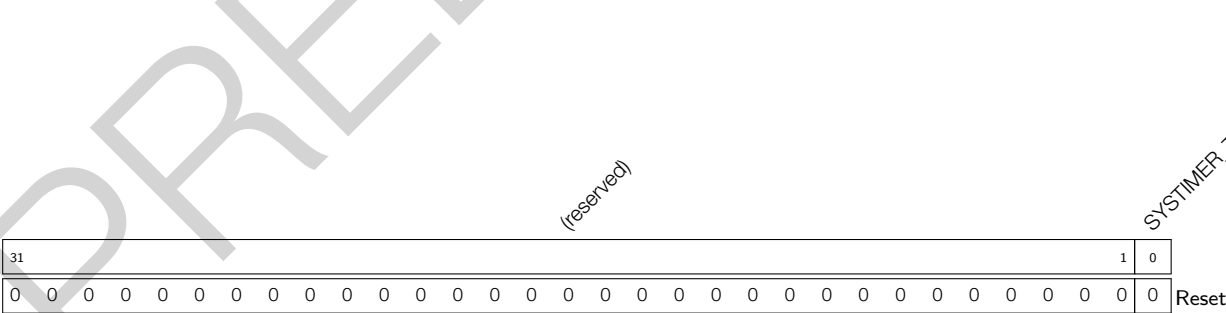
SYSTIMER_TIMER_UNIT1_VALUE_HI 计数器 UNIT1 的读数，高 20 位。(RO)

Register 9.12. SYSTIMER_UNIT1_VALUE_LO_REG (0x004C)



SYSTIMER_TIMER_UNIT1_VALUE_LO 计数器 UNIT1 的读数，低 32 位。(RO)

Register 9.13. SYSTIMER_UNIT1_LOAD_REG (0x0060)



SYSTIMER_TIMER_UNIT1_LOAD 计数器 UNIT1 的同步使能信号。置位此位，将重新装载寄存器 [SYSTIMER_TIMER_UNIT1_LOAD_HI](#) 和 [SYSTIMER_TIMER_UNIT1_LOAD_LO](#) 的值到计数器 UNIT1。(WT)

Register 9.14. SYSTIMER_TARGET0_HI_REG (0x001C)

(reserved)												SYSTIMER_TIMER_TARGET0_HI																																						
31												20																			19																			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0																		0																			
																																Reset																		

SYSTIMER_TIMER_TARGET0_HI 待装载至 COMP0 的报警值，高 20 位。(R/W)

Register 9.15. SYSTIMER_TARGET0_LO_REG (0x0020)

31																																0																																
0																																																																Reset

SYSTIMER_TIMER_TARGET0_LO 待装载至 COMP0 的报警值，低 32 位。(R/W)

Register 9.16. SYSTIMER_TARGET0_CONF_REG (0x0034)

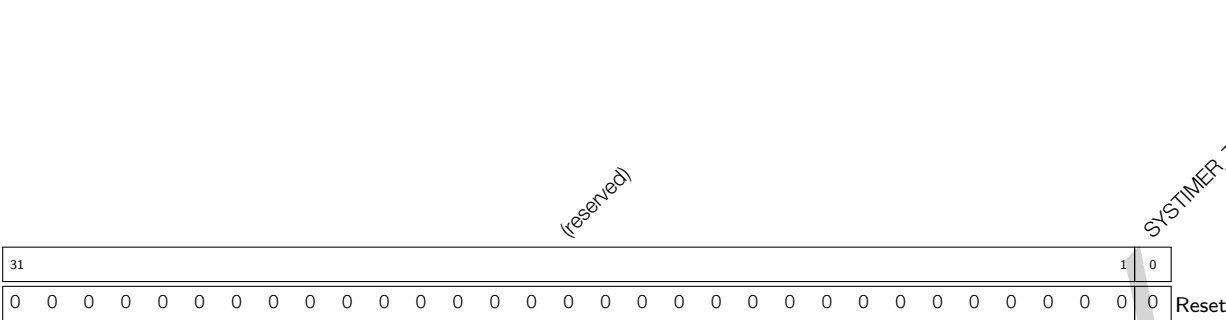
SYSTIMER_TARGET0_TIMER_UNIT_SEL										SYSTIMER_TARGET0_PERIOD																	
SYSTIMER_TARGET0_PERIOD_MODE																											
(reserved)																											
31	30	29														26	25	0									
0	0	0	0	0	0	0	0x00000																				
Reset																											

SYSTIMER_TARGET0_PERIOD 待装载至 COMP0 的报警周期。(R/W)

SYSTIMER_TARGET0_PERIOD_MODE 设置 COMP0 为周期报警模式。(R/W)

SYSTIMER_TARGET0_TIMER_UNIT_SEL 选择要与 COMP0 比较的计数器。(R/W)

Register 9.17. SYSTIMER_COMP0_LOAD_REG (0x0050)



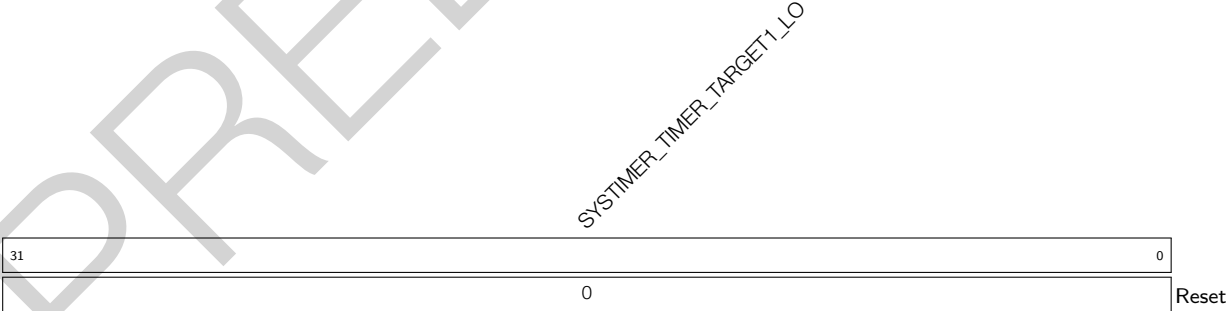
SYSTIMER_TIMER_COMP0_LOAD 比较器 COMP0 的同步使能信号。置位此位，将重新装载报警值或报警周期到 COMP0。(WT)

Register 9.18. SYSTIMER_TARGET1_HI_REG (0x0024)



SYSTIMER_TIMER_TARGET1_HI 待装载至 COMP1 的报警值，高 20 位。(R/W)

Register 9.19. SYSTIMER_TARGET1_LO_REG (0x0028)



SYSTIMER_TIMER_TARGET1_LO 待装载至 COMP1 的报警值，低 32 位。(R/W)

Register 9.20. SYSTIMER_TARGET1_CONF_REG (0x0038)

31		30		29		26		25																0
0		0		0		0		0		0x00000														

SYSTIMER_TARGET1_TIMER_UNIT_SEL

SYSTIMER_TARGET1_PERIOD_MODE

(reserved)

SYSTIMER_TARGET1_PERIOD

Reset

- SYSTIMER_TARGET1_PERIOD 待装载至 COMP1 的报警周期。(R/W)
- SYSTIMER_TARGET1_PERIOD_MODE 设置 COMP1 为周期报警模式。(R/W)
- SYSTIMER_TARGET1_TIMER_UNIT_SEL 选择要与 COMP1 比较的计数器。(R/W)

Register 9.21. SYSTIMER_COMP1_LOAD_REG (0x0054)

																												1	0
0																												0	0

(reserved)

SYSTIMER_TIMER_COMP1_LOAD

Reset

- SYSTIMER_TIMER_COMP1_LOAD 比较器 COMP1 的同步使能信号。置位此位，将重新装载报警值或报警周期到 COMP1。(WT)

Register 9.22. SYSTIMER_TARGET2_HI_REG (0x002C)

																																0
0																				0												0

(reserved)

SYSTIMER_TIMER_TARGET2_HI

Reset

- SYSTIMER_TIMER_TARGET2_HI 待装载至比较器 COMP2 的报警值，高 20 位。(R/W)

Register 9.23. SYSTIMER_TARGET2_LO_REG (0x0030)

SYSTIMER_TIMER_TARGET2_LO																															
31																															0
0																															
Reset																															

SYSTIMER_TIMER_TARGET2_LO 待装载至比较器 COMP2 的报警值，低 32 位。(R/W)

Register 9.24. SYSTIMER_TARGET2_CONF_REG (0x003C)

SYSTIMER_TARGET2_CONF_REG																													
SYSTIMER_TARGET2_TIMER_UNIT_SEL SYSTIMER_TARGET2_PERIOD_MODE (reserved) SYSTIMER_TARGET2_PERIOD																													
31	30	29			26	25																							0
0	0	0	0	0	0	0	0x00000																						Reset

- SYSTIMER_TARGET2_PERIOD** 待装载至 COMP2 的报警周期。(R/W)
- SYSTIMER_TARGET2_PERIOD_MODE** 设置 COMP2 为周期报警模式。(R/W)
- SYSTIMER_TARGET2_TIMER_UNIT_SEL** 选择要与 COMP2 比较的计数器。(R/W)

Register 9.25. SYSTIMER_COMP2_LOAD_REG (0x0058)

(reserved) SYSTIMER_TIMER_COMP2_LOAD																													
31																													1 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																													Reset

SYSTIMER_TIMER_COMP2_LOAD 比较器 COMP2 的同步使能信号。置位此位，将重新装载报警值或报警周期至 COMP2。(WT)

Register 9.26. SYSTIMER_INT_ENA_REG (0x0064)

(reserved)																										SYSTIMER_TARGET2_INT_ENA			
																										SYSTIMER_TARGET1_INT_ENA			
																										SYSTIMER_TARGET0_INT_ENA			
31																										3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- SYSTIMER_TARGET0_INT_ENA
- SYSTIMER_TARGET0_INT 中断使能位。(R/W)
- SYSTIMER_TARGET1_INT_ENA
- SYSTIMER_TARGET1_INT 中断使能位。(R/W)
- SYSTIMER_TARGET2_INT_ENA
- SYSTIMER_TARGET2_INT 中断使能位。(R/W)

Register 9.27. SYSTIMER_INT_RAW_REG (0x0068)

(reserved)																										SYSTIMER_TARGET2_INT_RAW			
																										SYSTIMER_TARGET1_INT_RAW			
																										SYSTIMER_TARGET0_INT_RAW			
31																										3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- SYSTIMER_TARGET0_INT_RAW
- SYSTIMER_TARGET0_INT 中断原始位。(R/WTC/SS)
- SYSTIMER_TARGET1_INT_RAW
- SYSTIMER_TARGET1_INT 中断原始位。(R/WTC/SS)
- SYSTIMER_TARGET2_INT_RAW
- SYSTIMER_TARGET2_INT 中断原始位。(R/WTC/SS)

Register 9.28. SYSTIMER_INT_CLR_REG (0x006C)

(reserved)																																SYSTIMER_TARGET2_INT_CLR SYSTIMER_TARGET1_INT_CLR SYSTIMER_TARGET0_INT_CLR					
31																															3	2	1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- SYSTIMER_TARGET0_INT_CLR SYSTIMER_TARGET0_INT 中断清除位。(WT)
- SYSTIMER_TARGET1_INT_CLR SYSTIMER_TARGET1_INT 中断清除位。(WT)
- SYSTIMER_TARGET2_INT_CLR SYSTIMER_TARGET2_INT 中断清除位。(WT)

Register 9.29. SYSTIMER_INT_ST_REG (0x0070)

(reserved)																												SYSTIMER_TARGET2_INT_ST SYSTIMER_TARGET1_INT_ST SYSTIMER_TARGET0_INT_ST			
31																												3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- SYSTIMER_TARGET0_INT_ST SYSTIMER_TARGET0_INT 中断状态位。(RO)
- SYSTIMER_TARGET1_INT_ST SYSTIMER_TARGET1_INT 中断状态位。(RO)
- SYSTIMER_TARGET2_INT_ST SYSTIMER_TARGET2_INT 中断状态位。(RO)

Register 9.30. SYSTIMER_DATE_REG (0x00FC)

SYSTIMER_DATE																														31	0
0x2006171																														Reset	

- SYSTIMER_DATE 版本控制寄存器。(R/W)

10 定时器组 (TIMG)

10.1 概述

通用定时器可用于准确设定时间间隔、在一定间隔后触发（周期或非周期的）中断或充当硬件时钟。如图 10-1 所示，ESP32-C3 包含两个定时器组，即定时器组 0 和定时器组 1。每个定时器组有一个通用定时器（下文用 T0 表示）和一个主系统看门狗定时器。所有通用定时器均基于 16 位预分频器和 54 位可自动重新加载的可逆计数器。

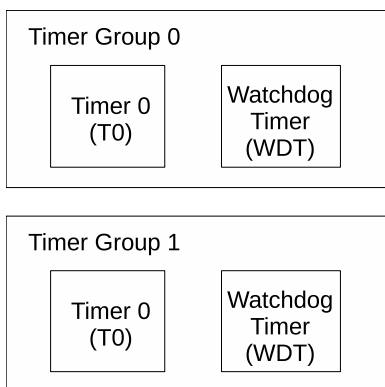


图 10-1. 定时器组

本章包含主系统看门狗定时器的寄存器描述，其功能描述请参阅章节 11 看门狗定时器 (WDT)。本章中“定时器”指代通用定时器。

定时器具有如下功能：

- 16 位时钟预分频器，分频系数为 2 到 65536
- 54 位时基计数器可配置成递增或递减
- 可读取时基计数器的实时值
- 暂停和恢复时基计数器
- 可配置的报警产生机制
- 计数器值重新加载（报警时自动重新加载或软件控制的即时重新加载）
- 电平触发中断

10.2 功能描述

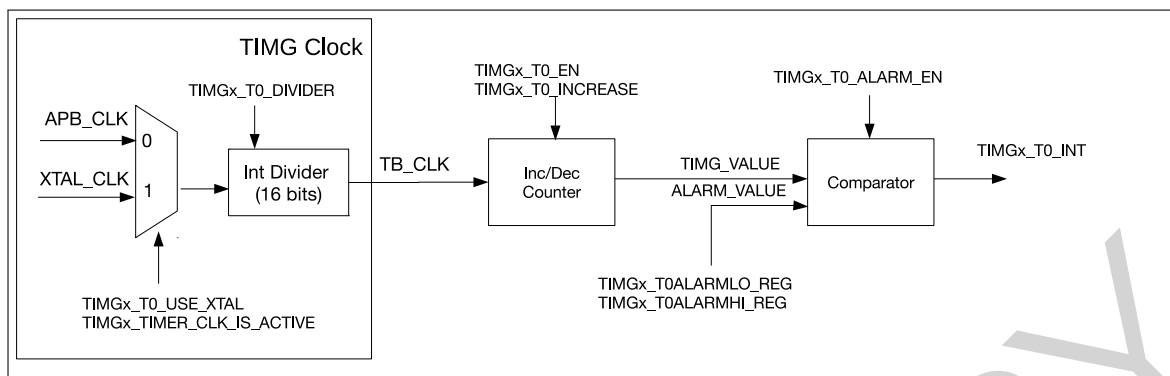


图 10-2. 定时器组架构

图 10-2 为定时器组的 T0。T0 包含一个时钟选择器、一个 16 位整数预分频器、一个时基计数器和一个用于产生警报的比较器。

10.2.1 16 位预分频器与时钟选择器

每个定时器可通过配置寄存器 `TIMG_T0CONFIG_REG` 的 `TIMG_T0_USE_XTAL` 字段，选择 APB 时钟 (APB_CLK) 或外部时钟 (XTAL_CLK) 作为时钟源。要开启所选时钟，需置位 `TIMG_REGCLK_REG` 寄存器的 `TIMG_TIMER_CLK_IS_ACTIVE` 字段，要关闭时钟需清零该字段。所选时钟经 16 位预分频器分频，产生时基计数器使用的时基计数器时钟 (TB_CLK)。16 位预分频器的分频系数可通过 `TIMG_T0_DIVIDER` 字段配置，选取从 2 到 65536 之间的任意值。注意，将 `TIMG_T0_DIVIDER` 置 0 后，分频系数会变为 65536。`TIMG_T0_DIVIDER` 置 1 时，实际分频系数为 2，计数器的值为实际时间的一半。

要更改 16 位预分频器，需先重新配置 `TIMG_T0_DIVIDER` 字段，再将 `TIMG_T0_DIVIDER_RST` 置 1，同时需要关闭定时器（即 `TIMG_T0_EN` 必须清零），否则会造成不可预知的结果。

10.2.2 54 位时基计数器

54 位时基计数器基于 TB_CLK，可通过 `TIMG_T0_INCREASE` 字段配置为递增或递减。时基计数器可通过置位或清零 `TIMG_T0_EN` 字段使能或关闭。使能时，时基计数器的值会在每个 TB_CLK 周期递增或递减。关闭时，时基计数器暂停计数。注意，`TIMG_T0_EN` 置位后，`TIMG_T0_INCREASE` 字段还可以更改，时基计数器可立即改变计数方向。

时基计数器 54 位定时器的当前值必须被锁入两个寄存器，才能被 CPU 读取（因为 CPU 为 32 位）。在 `TIMG_T0UPDATE_REG` 上写任意值，54 位定时器的值可立即锁入寄存器 `TIMG_T0LO_REG` 和 `TIMG_T0HI_REG`，两个寄存器分别锁存低 32 位和高 22 位。在 `TIMG_T0UPDATE_REG` 被写入新值之前，寄存器 `TIMG_T0LO_REG` 和 `TIMG_T0HI_REG` 的值保持不变，以便 CPU 读值。

10.2.3 报警产生

定时器可配置为在当前值与报警值相同时触发报警。报警会产生中断，（可选择）让定时器的当前值自动重新加载（详见第 10.2.4 节）。

54 位报警值可在 `TIMG_T0ALARMLO_REG` 和 `TIMG_T0ALARMHI_REG` 配置，两者分别代表报警值的低 32 位和高 22 位。但是，只有置位 `TIMG_T0_ALARM_EN` 字段使能报警功能后，配置的报警值才会生效。为解决报警使能“过晚”（即报警使能时，定时器的值已过报警值），可逆计数器向上计数时，若定时器的当前值高于报警值（在一定范围内），或可逆计数器向下计数时，定时器的当前值低于报警值（在一定范围内），硬件都会立即

触发报警。表 10-1 和表 10-2 说明了定时器当前值、报警值与报警触发的关系。假设定时器当前值和报警值如下：

- $TIMG_VALUE = \{TIMG_TOHI_REG, TIMG_TOLO_REG\}$
- $ALARM_VALUE = \{TIMG_TOALARMHI_REG, TIMG_TOALARMLO_REG\}$

表 10-1. 可逆计数器向上计数时的报警触发场景

场景	范围	报警
1	$ALARM_VALUE - TIMG_VALUE > 2^{53}$	触发
2	$0 < ALARM_VALUE - TIMG_VALUE \leq 2^{53}$	可逆计数器向上计数， $TIMG_VALUE$ 达到 $ALARM_VALUE$ 时报警
3	$0 \leq TIMG_VALUE - ALARM_VALUE < 2^{53}$	触发
4	$TIMG_VALUE - ALARM_VALUE \geq 2^{53}$	可逆计数器向上计数达到最大值时，重新开始从 0 向上计数， $TIMG_VALUE$ 达到 $ALARM_VALUE$ 时触发报警

表 10-2. 可逆计数器向下计数时的报警触发场景

场景	范围	报警
5	$TIMG_VALUE - ALARM_VALUE > 2^{53}$	触发
6	$0 < TIMG_VALUE - ALARM_VALUE \leq 2^{53}$	可逆计数器向下计数， $TIMG_VALUE$ 达到 $ALARM_VALUE$ 时报警
7	$0 \leq ALARM_VALUE - TIMG_VALUE < 2^{53}$	触发
8	$ALARM_VALUE - TIMG_VALUE \geq 2^{53}$	可逆计数器向下计数达到最小值时，重新开始从最大值向下计数， $TIMG_VALUE$ 达到 $ALARM_VALUE$ 时触发报警

报警时，`TIMG_TO_ALARM_EN` 字段自动清零，在下次置位 `TIMG_TO_ALARM_EN` 前不会再次报警。

10.2.4 定时器重新加载

定时器重新加载指将定时器的低 32 位和高 22 位分别更新为寄存器 `TIMG_TO_LOAD_LO` 和 `TIMG_TO_LOAD_HI` 存储的重新加载值。但是，把重新加载值写入 `TIMG_TO_LOAD_LO` 和 `TIMG_TO_LOAD_HI` 寄存器不会改变定时器的当前值。写入的重新加载值会被定时器忽视，直到重新加载事件被触发。重新加载事件可由软件即时重新加载或报警时自动重新加载触发。

CPU 在寄存器 `TIMG_TOLOAD_REG` 写任意值会触发软件即时重新加载，定时器的当前值会立即改变。如置位 `TIMG_TO_EN`，定时器会继续从新数值开始递增或递减计数。如清零 `TIMG_TO_EN`，定时器将保持当前值，直至计数重新使能。

报警时自动重新加载功能可让定时器在报警时重新加载，从重新加载值开始继续递增或递减计数。该功能通常用于周期性报警时重置定时器的值。要使能报警时自动重新加载，需将 `TIMG_TO_AUTORELOAD` 字段置 1。如未使能该功能，报警后定时器的值会在过报警值后继续递增或递减。

10.2.5 低功耗时钟 (SLOW_CLK) 频率计算

定时器组可以通过使用 `XTAL_CLK` 计算低功耗时钟的三个慢速时钟源 `RTC_CLK`、`RTC20M_D256_CLK` 和 `XTAL32K_CLK` 的实际频率。计算方式如下：

1. 通过周期性或单次计算的方式启动频率计算模块；
2. 在接收到计算开始的信号后，两个分别工作在 XTAL_CLK 以及 SLOW_CLK 的计数器同时开始计数，当 SLOW_CLK 的计数器达到设定的计算周期 C0 时，同时停止两个计数器；
3. 通过 XTAL_CLK 的计数器值 C1 即可计算 SLOW_CLK 的时钟频率： $f_{rtc} = \frac{C0 \times f_{XTAL_CLK}}{C1}$

10.2.6 中断

每个定时器都有一根连接至 CPU 的中断线。因此，每个定时器组有两根中断线。定时器每次产生的电平中断必须由 CPU 清除。

电平中断在报警后（或看门狗定时器阶段超时）触发。报警（或阶段超时）后，电平中断会一直被拉高，直至手动清除中断。要能使定时器的中断，TIMG_T0_INT_ENA 需置 1。

每个定时器组的中断受一组寄存器控制。每个定时器在下列寄存器中都有对应的位：

- TIMG_T0_INT_RAW：报警时置 1。该位在写值到对应的 TIMG_T0_INT_CLR 位后才会被清零。
- TIMG_WDT_INT_RAW：阶段超时置 1。该位在写值到对应的 TIMG_WDT_INT_CLR 位后才会被清零。
- TIMG_T0_INT_ST：反映每个定时器中断的状态，通过用 TIMG_T0_INT_ENA 屏蔽 TIMG_T0_INT_RAW 位来生成。
- TIMG_WDT_INT_ST：反映每个看门狗定时器中断的状态，通过用 TIMG_WDT_INT_ENA 屏蔽 TIMG_WDT_INT_RAW 位来生成。
- TIMG_T0_INT_ENA：用于使能或屏蔽组内定时器的中断状态位。
- TIMG_WDT_INT_ENA：用于使能或屏蔽组内看门狗定时器的中断状态位。
- TIMG_T0_INT_CLR：置 1 此位清除定时器中断，定时器在 TIMG_T0_INT_RAW 和 TIMG_T0_INT_ST 的对应位会清零。注意，下一个中断产生前，必须清除定时器中断。
- TIMG_WDT_INT_CLR：置 1 此位清除定时器中断，看门狗定时器在 TIMG_WDT_INT_RAW 和 TIMG_WDT_INT_ST 的对应位会清零。注意，下一个中断产生前，必须清除看门狗定时器中断。

10.3 配置与使用

10.3.1 定时器用作简单时钟

1. 配置时基计数器。
 - 置位或清除 TIMG_T0_USE_XTAL 字段选择时钟源。
 - 置位 TIMG_T0_DIVIDER 配置 16 位预分频器。
 - 置位或清除 TIMG_T0_INCREASE 配置定时器方向。
 - 在 TIMG_T0_LOAD_LO 和 TIMG_T0_LOAD_HI 上写初始值设置定时器的初始值，然后在 TIMG_T0LOAD_REG 上写任意值将初始值重新加载进定时器。
2. 置位 TIMG_T0_EN 开启定时器。
3. 获得定时器的当前值。
 - 在 TIMG_T0UPDATE_REG 上写任意值锁存定时器的当前值。
 - 从 TIMG_T0LO_REG 和 TIMG_T0HI_REG 读取锁存的定时器值。

10.3.2 定时器用于单次报警

1. 按照第 10.3.1 节的第 1 步配置时基计数器。
2. 配置报警。
 - 置位 `TIMG_T0_ALARMLO_REG` 和 `TIMG_T0_ALARMHI_REG` 配置报警值。
 - 置位 `TIMG_T0_INT_ENA` 使能中断。
3. 清零 `TIMG_T0_AUTORELOAD` 关闭自动重新加载。
4. 置位 `TIMG_T0_ALARM_EN` 开启报警。
5. 处理报警中断。
 - 置位定时器在 `TIMG_T0_INT_CLR` 的对应位清除中断。
 - 清零 `TIMG_T0_EN` 关闭定时器。

10.3.3 定时器用于周期性报警

1. 按照第 10.3.1 节的第 1 步配置时基计数器。
2. 按照第 10.3.2 节的第 2 步配置报警。
3. 置位 `TIMG_T0_AUTORELOAD` 使能自动重新加载,将重新加载值写入 `TIMG_T0_LOAD_LO` 和 `TIMG_T0_LOAD_HI`。
4. 置位 `TIMG_T0_ALARM_EN` 开启报警。
5. 处理报警中断 (每次报警时重复)。
 - 置位定时器在 `TIMG_T0_INT_CLR` 的对应位清除中断。
 - 如下一次报警需要新的报警值和重新加载值 (即每次都有不同的报警间隔), 则应根据需要重新配置 `TIMG_T0_ALARMLO_REG`、`TIMG_T0_ALARMHI_REG`、`TIMG_T0_LOAD_LO` 和 `TIMG_T0_LOAD_HI`。否则, 上述寄存器应保持不变。
 - 置位 `TIMG_T0_ALARM_EN` 重新使能报警。
6. (最后一次报警时) 关闭定时器。
 - 置位定时器在 `TIMG_T0_INT_CLR` 的对应位清除中断。
 - 清零 `TIMG_T0_EN` 关闭定时器。

10.3.4 SLOW_CLK 频率计算

1. 单次计算
 - 设置 `TIMG_RTC_CALI_CLK_SEL` 选择需要计算频率的时钟 (SLOW_CLK 的时钟源), 设置 `TIMG_RTC_CALI_MAX` 配置频率计算时间。
 - 清空 `TIMG_RTC_CALI_START_CYCLING` 选择单次校准模式, 然后配置 `TIMG_RTC_CALI_START` 开启两个计数器。
 - 等待 `TIMG_RTC_CALI_RDY` 的值变为 1, 读取 `TIMG_RTC_CALI_VALUE` 获取 XTAL_CLK 计数器值, 计算 SLOW_CLK 频率。
2. 周期性计算

- 设置 `TIMG_RTC_CALI_CLK_SEL` 选择需要计算频率的时钟(`SLOW_CLK`的时钟源),设置 `TIMG_RTC_CALI_MAX` 配置频率计算时间。
- 使能 `TIMG_RTC_CALI_START_CYCLING`, 硬件将不间断进行频率计算过程。
- 只要`TIMG_RTC_CALI_CYCLING_DATA_VLD` 为 1, 即表示`TIMG_RTC_CALI_VALUE` 有效。

3. 超时

如果 `SLOW_CLK` 的计数器没有在`TIMG_RTC_CALI_TIMEOUT_RST_CNT` 的 `XTAL_CLK` 计数器内完成计数, 将置位 `TIMG_RTC_CALI_TIMEOUT` 标记计算超时。

10.4 寄存器列表

本小节的所有地址均为相对于 **定时器组** 基地址的地址偏移量（相对地址），具体基地址请见章节 3 **系统和存储器** 中的表 3-4。

名称	描述	地址	访问
定时器 0 控制和配置寄存器			
TIMG_T0CONFIG_REG	定时器 0 配置寄存器	0x0000	varies
TIMG_T0LO_REG	定时器 0 的当前值，低 32 位	0x0004	RO
TIMG_T0HI_REG	定时器 0 的当前值，高 22 位	0x0008	RO
TIMG_T0UPDATE_REG	写值将当前定时器的值复制到 TIMG_T0LO_REG 或 TIMG_T0HI_REG	0x000C	R/ W/ SC
TIMG_T0ALARMLO_REG	定时器 0 的报警值，低 32 位	0x0010	R/W
TIMG_T0ALARMHI_REG	定时器 0 的报警值，高位	0x0014	R/W
TIMG_T0LOADLO_REG	定时器 0 的重新加载值，低 32 位	0x0018	R/W
TIMG_T0LOADHI_REG	定时器 0 的重新加载值，高 22 位	0x001C	R/W
TIMG_T0LOAD_REG	写值从 TIMG_T0LOADLO_REG 或 TIMG_T0LOADHI_REG 上加载定时器	0x0020	WT
看门狗定时器控制和配置寄存器			
TIMG_WDTCONFIG0_REG	看门狗定时器配置寄存器	0x0048	varies
TIMG_WDTCONFIG1_REG	看门狗定时器预分频器寄存器	0x004C	varies
TIMG_WDTCONFIG2_REG	看门狗定时器阶段 0 超时值	0x0050	R/W
TIMG_WDTCONFIG3_REG	看门狗定时器阶段 1 超时值	0x0054	R/W
TIMG_WDTCONFIG4_REG	看门狗定时器阶段 2 超时值	0x0058	R/W
TIMG_WDTCONFIG5_REG	看门狗定时器阶段 3 超时值	0x005C	R/W
TIMG_WDTFEED_REG	写值喂看门狗定时器	0x0060	WT
TIMG_WDTWPROTECT_REG	看门狗写保护寄存器	0x0064	R/W
RTC 频率计算控制和配置寄存器			
TIMG_RTCCALICFG_REG	RTC 频率计算配置寄存器 0	0x0068	varies
TIMG_RTCCALICFG1_REG	RTC 频率计算配置寄存器 1	0x006C	RO
TIMG_RTCCALICFG2_REG	RTC 频率计算配置寄存器 2	0x0080	varies
中断寄存器			
TIMG_INT_ENA_TIMERS_REG	中断使能位	0x0070	R/W
TIMG_INT_RAW_TIMERS_REG	原始中断状态	0x0074	R/ SS/ WTC
TIMG_INT_ST_TIMERS_REG	屏蔽中断状态	0x0078	RO
TIMG_INT_CLR_TIMERS_REG	中断清除位	0x007C	WT
版本寄存器			
TIMG_NTIMERS_DATE_REG	版本控制寄存器	0x00F8	R/W
时钟配置寄存器			
TIMG_REGCLK_REG	定时器组时钟门控寄存器	0x00FC	R/W

10.5 寄存器

本小节的所有地址均为相对于 **定时器组** 基地址的地址偏移量（定时器组 0 和定时器组 1 各自的相对地址），具体基地址请见章节 3 **系统和存储器** 中的表 3-4。

Register 10.1. TIMG_T0CONFIG_REG (0x0000)

TIMG_T0_EN TIMG_T0_INCREASE TIMG_T0_AUTORELOAD				TIMG_T0_DIVIDER										TIMG_T0_DIVIDER_RST (reserved) TIMG_T0_ALARM_EN TIMG_T0_USE_XTAL (reserved)																0
31	30	29	28	13										12	11	10	9	8									0			
0	1	1	0x01										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

TIMG_T0_USE_XTAL 0: 使用 APB_CLK 作为定时器组的源时钟；1: 使用 XTAL_CLK 作为定时器组的源时钟。(R/W)

TIMG_T0_ALARM_EN 置 1 后，报警使能。报警时，此位自动清零。(R/W/SC)

TIMG_T0_DIVIDER_RST 置 1 后，复位定时器 x 时钟分频器的计数器。(WT)

TIMG_T0_DIVIDER 定时器 x 时钟 (TO_clk) 的预分频器值。(R/W)

TIMG_T0_AUTORELOAD 置 1 后，定时器 x 报警时自动重新加载使能。(R/W)

TIMG_T0_INCREASE 置 1 后，定时器 x 的时基计数器会在每个时钟周期后递增。清零后，定时器 x 的时基计数器会递减。(R/W)

TIMG_T0_EN 置 1 后，定时器 x 时基计数器使能。(R/W)

Register 10.2. TIMG_T0LO_REG (0x0004)

TIMG_T0_LO																																
31																															0	
0x000000																																Reset

TIMG_T0_LO 在 TIMG_T0UPDATE_REG 上写值后，可读取定时器 x 时基计数器的低 32 位。(RO)

Register 10.3. TIMG_T0HI_REG (0x0008)

(reserved)											TIMG_T0_HI																																																																																							
31											22											21											0																																																																	
0											0											0											0											0											0											0											0x0000											Reset										

TIMG_T0_HI 在 TIMG_T0UPDATE_REG 上写值后，可读取定时器 x 时基计数器的高 22 位。(RO)

Register 10.4. TIMG_T0UPDATE_REG (0x000C)

TIMG_T0_UPDATE		(reserved)																											
31	30																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset																													

TIMG_T0_UPDATE 在 TIMG_T0UPDATE_REG 上写 0 或 1，计数器的值被锁住。(R/W/SC)

Register 10.5. TIMG_T0ALARMLO_REG (0x0010)

TIMG_TO_ALARM_LO																															
31																															0
0x000000																															
Reset																															

TIMG_T0_ALARM_LO 定时器 x 时基计数器触发警报值的低 32 位。(R/W)

Register 10.6. TIMG_T0ALARMHI_REG (0x0014)

(reserved)										TIMG_TO_ALARM_HI																				
31																				22	21									0
0	0	0	0	0	0	0	0	0	0	0	0x0000																			
Reset																														

TIMG_T0_ALARM_HI 定时器 x 时基计数器触发警报值的高 22 位。(R/W)

Register 10.7. TIMG_T0LOADLO_REG (0x0018)

TIMG_T0_LOAD_LO																																																									
31																													0																												
0x000000																																																									
Reset																																																									

TIMG_T0_LOAD_LO 定时器 x 时基计数器重新加载的低 32 位值。(R/W)

Register 10.8. TIMG_T0LOADHI_REG (0x001C)

(reserved)										TIMG_T0_LOAD_HI											
31										22	21									0	
0	0	0	0	0	0	0	0	0	0	0x0000											Reset

TIMG_T0_LOAD_HI 定时器 \times 时基计数器重新加载的高 22 位值。(R/W)

Register 10.9. TIMG_T0LOAD_REG (0x0020)

																TIMG_T0_LOAD															
31																														0	
		0x000000																												Reset	

TIMG_T0_LOAD 写任意值触发定时器 \times 时基计数器重新加载。(WT)

Register 10.10. TIMG_WDTCONFIG0_REG (0x0048)

TIMG_WDT_EN		TIMG_WDT_STG0		TIMG_WDT_STG1		TIMG_WDT_STG2		TIMG_WDT_STG3		TIMG_WDT_CONF_UPDATE_EN		TIMG_WDT_USE_XTAL		TIMG_WDT_CPU_RESET_LENGTH		TIMG_WDT_SYS_RESET_LENGTH		TIMG_WDT_FLASHBOOT_MOD_EN		TIMG_WDT_PROCPU_RESET_EN		TIMG_WDT_APPCPU_RESET_EN		(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31	30	29	28	27	26	25	24	23	22	21	20	18	17	15	14	13	12	11											0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0x1		0x1		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMG_WDT_APPCPU_RESET_EN WDT 复位 CPU 使能。(R/W)

TIMG_WDT_PROCPU_RESET_EN WDT 复位 CPU 使能。(R/W)

TIMG_WDT_FLASHBOOT_MOD_EN 置 1 后, flash 启动保护使能。(R/W)

TIMG_WDT_SYS_RESET_LENGTH 系统复位信号长度选择。0: 100 ns; 1: 200 ns; 2: 300 ns; 3: 400 ns; 4: 500 ns; 5: 800 ns; 6: 1.6 μ s; 7: 3.2 μ s。(R/W)

TIMG_WDT_CPU_RESET_LENGTH CPU 复位信号长度选择。0: 100 ns; 1: 200 ns; 2: 300 ns; 3: 400 ns; 4: 500 ns; 5: 800 ns; 6: 1.6 μ s; 7: 3.2 μ s。(R/W)

TIMG_WDT_USE_XTAL 选择看门狗定时器的时钟。0: APB_CLK; 1: XTAL_CLK。(R/W)

TIMG_WDT_CONF_UPDATE_EN 更新看门狗定时器配置寄存器。(WT)

TIMG_WDT_STG3 阶段 3 配置。0: 关闭; 1: 中断; 2: 复位 CPU; 3: 复位系统。(R/W)

TIMG_WDT_STG2 阶段 2 配置。0: 关闭; 1: 中断; 2: 复位 CPU; 3: 复位系统。(R/W)

TIMG_WDT_STG1 阶段 1 配置。0: 关闭; 1: 中断; 2: 复位 CPU; 3: 复位系统。(R/W)

TIMG_WDT_STG0 阶段 0 配置。0: 关闭; 1: 中断; 2: 复位 CPU; 3: 复位系统。(R/W)

TIMG_WDT_EN 置 1 后, MWDT 使能。(R/W)

Register 10.11. TIMG_WDTCONFIG1_REG (0x004C)

TIMG_WDT_CLK_PRESCALE																(reserved)										TIMG_WDT_DIVCNT_RST		0
31																16	15										1	0
0x01																0 0 0 0 0 0 0 0 0 0 0 0 0 0 0										0	0	0

TIMG_WDT_DIVCNT_RST 置 1 后, 复位看门狗定时器时钟分频器的计数器。(WT)

TIMG_WDT_CLK_PRESCALE MWDT 时钟预分频器值。MWDT 时钟长度 = 12.5 ns * TIMG_WDT_CLK_PRESCALE。(R/W)

Register 10.12. TIMG_WDTCONFIG2_REG (0x0050)

TIMG_WDT_STG0_HOLD	
31	0
26000000	
Reset	

TIMG_WDT_STG0_HOLD 阶段 0 超时时间，单位是 MWDAT 时钟周期。(R/W)

Register 10.13. TIMG_WDTCONFIG3_REG (0x0054)

TIMG_WDT_STG1_HOLD	
31	0
0x7fffff	
Reset	

TIMG_WDT_STG1_HOLD 阶段 1 超时时间，单位是 MWDAT 时钟周期。(R/W)

Register 10.14. TIMG_WDTCONFIG4_REG (0x0058)

TIMG_WDT_STG2_HOLD	
31	0
0x0ffff	
Reset	

TIMG_WDT_STG2_HOLD 阶段 2 超时时间，单位是 MWDAT 时钟周期。(R/W)

Register 10.15. TIMG_WDTCONFIG5_REG (0x005C)

TIMG_WDT_STG3_HOLD	
31	0
0x0ffff	
Reset	

TIMG_WDT_STG3_HOLD 阶段 3 超时时间，单位是 MWDAT 时钟周期。(R/W)

Register 10.16. TIMG_WDTFEED_REG (0x0060)

TIMG_WDT_FEED	
31	0
0x000000	
Reset	

TIMG_WDT_FEED 写任意值喂 MWDAT。(WT)

Register 10.17. TIMG_WDTWPROTECT_REG (0x0064)

TIMG_WDT_WKEY	
31	0
0x50d83aa1	
Reset	

TIMG_WDT_WKEY 如果寄存器的值与复位值不同，写保护使能。(R/W)

Register 10.20. TIMG_RTCCALICFG2_REG (0x0080)

TIMG_RTC_CAL_TIMEOUT_THRES										TIMG_RTC_CAL_TIMEOUT_RST_CNT					(reserved)			TIMG_RTC_CAL_TIMEOUT		
31							7	6	3			2	1	0						
0x1fffff							3			0			0	0	Reset					

TIMG_RTC_CALI_TIMEOUT 提示时钟频率计算超时。(RO)

TIMG_RTC_CALI_TIMEOUT_RST_CNT 频率计算超时复位周期。(R/W)

TIMG_RTC_CALI_TIMEOUT_THRES RTC 频率计算定时器的阈值。频率计算定时器的值超过此值时触发超时。(R/W)

Register 10.21. TIMG_INT_ENA_TIMERS_REG (0x0070)

(reserved)																												TIMG_WDT_INT_ENA		TIMG_TO_INT_ENA	
31																												2	1	0	
0 0																												0 0		Reset	

TIMG_TO_INT_ENA TIMG_TO_INT 中断的使能位。(R/W)

TIMG_WDT_INT_ENA TIMG_WDT_INT 中断的使能位。(R/W)

Register 10.22. TIMG_INT_RAW_TIMERS_REG (0x0074)

(reserved)																				TIMG_WDT_INT_RAW TIMG_TO_INT_RAW																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
31																												2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIMG_TO_INT_RAW TIMG_TO_INT 中断的原始中断状态位。(R/SS/WTC)

TIMG_WDT_INT_RAW TIMG_WDT_INT 中断的原始中断状态位。(R/SS/WTC)

Register 10.23. TIMG_INT_ST_TIMERS_REG (0x0078)

(reserved)																												TIMG_WDT_INT_ST TIMG_TO_INT_ST	
31																											2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																													Reset

TIMG_TO_INT_ST TIMG_TO_INT 中断的屏蔽中断状态位。(RO)

TIMG_WDT_INT_ST TIMG_WDT_INT 中断的屏蔽中断状态位。(RO)

Register 10.24. TIMG_INT_CLR_TIMERS_REG (0x007C)

(reserved)																															TIMG_WDT_INT_CLR TIMG_TO_INT_CLR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
31																															2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

TIMG_TO_INT_CLR 置位此位，清除 TIMG_TO_INT 中断。(WT)

TIMG_WDT_INT_CLR 置位此位，清除 TIMG_WDT_INT 中断。(WT)

Register 10.25. TIMG_NTIMERS_DATE_REG (0x00F8)

(reserved)				TIMG_NTIMGS_DATE																									
31	28	27	0																										
0	0	0	0	0x2006191																									Reset

TIMG_NTIMGS_DATE 版本控制寄存器。(R/W)

221
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

11 看门狗定时器 (WDT)

11.1 概述

看门狗定时器是一种硬件定时器，用于检测和修复故障。软件必须定期喂狗（复位），以防超时。系统或软件若出现不可预知的问题（比如软件卡在某个循环或逾期事件中）将无法按时喂狗，造成看门狗超时。因此，看门狗定时器有助于检测、处理系统或软件的错误行为。

如图 11-1 所示，ESP32-C3 中有三个数字看门狗定时器：章节 10 定时器组 (TIMG) 描述的两个定时器组中各有一个（称作主系统看门狗定时器，缩写为 MWDT），RTC 模块中有一个（称作 RTC 看门狗定时器，缩写为 RWDT）。数字看门狗在运行期间会经历四个阶段（除非看门狗按时喂狗或者处于关闭状态），每个阶段均可配置单独的超时时间和超时动作，其中 MWDT 支持中断、CPU 复位和内核复位三种超时动作，RWDT 支持中断、CPU 复位、内核复位和系统复位四种超时动作（详见章节 11.2.2.2 阶段与超时动作）。每个阶段的超时时间都可单独设置。

在 flash 引导模式下，RWDT 和定时器组 0 的 MWDT 会默认使能，以检测引导过程中发生的错误，并恢复运行。

ESP32-C3 中还有一个模拟看门狗定时器——超级看门狗 (SWD)。超级看门狗是模拟域的超低功耗电路，可以防止系统在数字电路异常状态下运行，并在必要时复位系统。

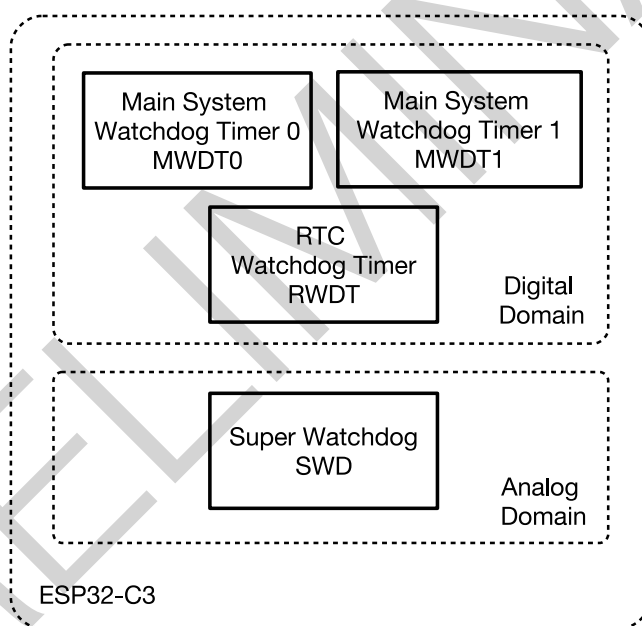


图 11-1. 看门狗定时器概览

请注意，本章节仅包含看门狗定时器的功能描述，其寄存器部分详见章节 10 定时器组 (TIMG) 和章节 4 低功耗管理 (RTC_CNTL) [to be added later]。

11.2 数字看门狗定时器

11.2.1 主要特性

看门狗定时器具有如下特性：

- 四个阶段，每个阶段都可配置超时时间。每阶段都可单独配置、使能和关闭

- 如在某个阶段发生超时，MWD0 会采取中断、CPU 复位和内核复位中的一种超时动作，RWDT 则会采取中断、CPU 复位、内核复位和系统复位中的一种超时动作
 - 32 位超时计数器
 - 写保护，防止 RWDT 和 MWD0 配置误改动
 - Flash 启动保护
- 如果在预定时间内 SPI flash 的引导过程没有完成，看门狗会重启整个主系统

11.2.2 功能描述

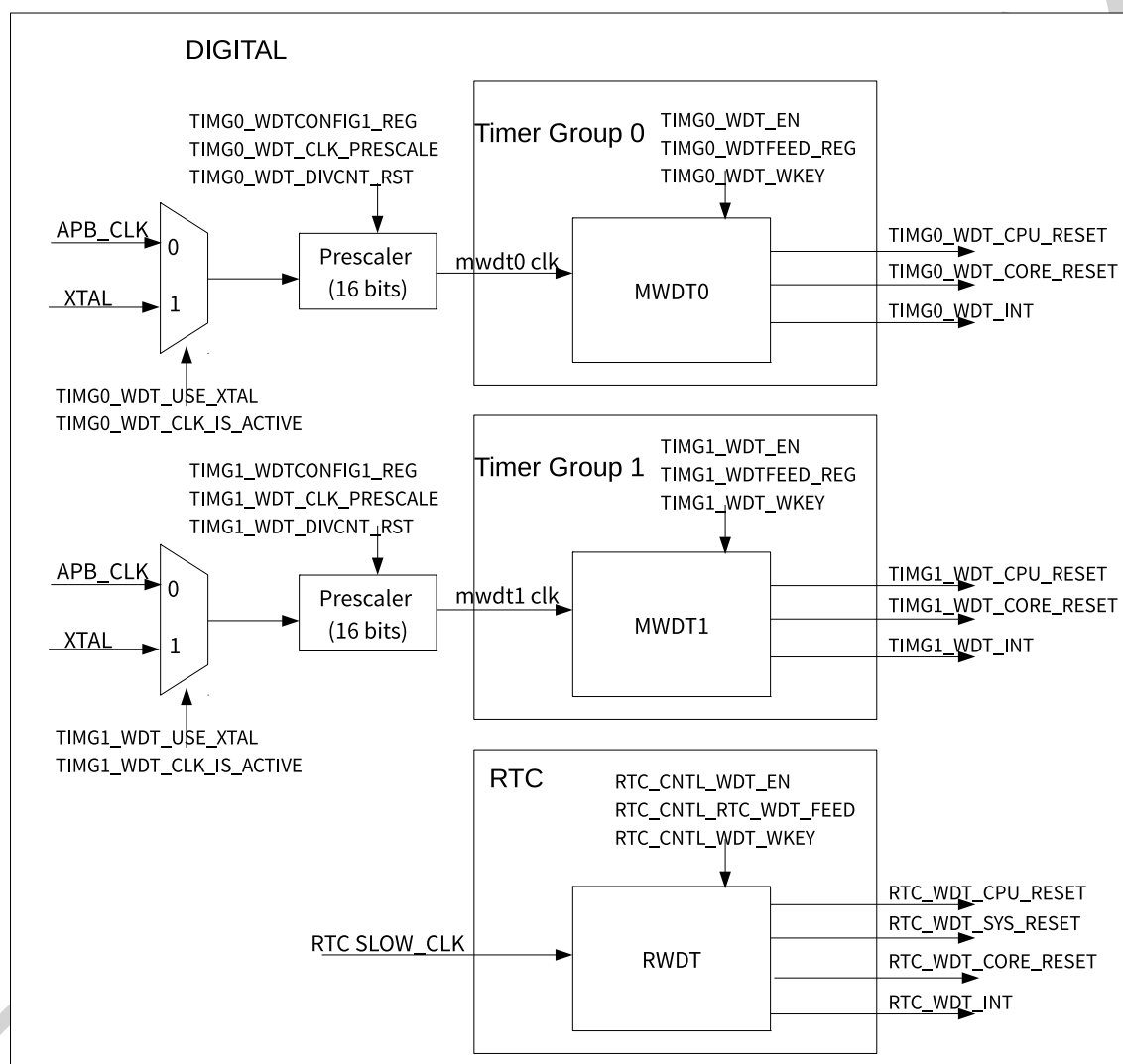


图 11-2. ESP32-C3 的看门狗定时器

图 11-2 为 ESP32-C3 数字系统中的三个看门狗定时器。

11.2.2.1 时钟源与 32 位计数器

每个看门狗定时器的核心是一个 32 位计数器。

MWD0 可通过设置 `TIMG_WDTCONFIG0_REG` 寄存器的 `TIMG_WDT_USE_XTAL` 字段选择 APB 时钟 (APB_CLK) 或外部时钟 (XTAL_CLK) 作为时钟源。将 `TIMG_REGCLK_REG` 寄存器的 `TIMG_WDT_CLK_IS_ACTIVE` 字段置 1

开启时钟，清零关闭时钟。时钟经由可配置的 16 位预分频器分频。MWD 的 16 位预分频器可通过 `TIMG_WDTCONFIG1_REG` 寄存器的 `TIMG_WDT_CLK_PRESCALE` 字段配置。`TIMG_WDT_DIVCNT_RST` 字段置位时，预分频器复位，可立即重新配置。

RWDT 直接将 RTC 慢速时钟（RTC 慢速时钟源详见章节 6 复位和时钟）用作时钟源。

MWD 和 RWDT 看门狗可分别通过设置 `TIMG_WDT_EN` 和 `RTC_CNTL_WDT_EN` 字段使能。看门狗使能后，其内部 32 位计数器的值会在每个时钟源周期内累加 1，直到达到该阶段的超时时间（即在该阶段发生超时）。如发生超时，计数器的值会重置为 0，同时看门狗进入下一阶段。如果软件成功喂狗，看门狗定时器会回到阶段 0，并将计数器的值重置为 0。软件向 `TIMG_WDTFEED_REG` 和 `RTC_CNTL_RTC_WDT_FEED` 寄存器内写入任意值，便可分别为 MDWT 和 RWDT 喂狗。

11.2.2.2 阶段与超时动作

定时器在各阶段可以配置不同的超时时间和对应的超时动作。某一阶段超时会触发对应的超时动作，同时计数器的值被重置为 0，看门狗进入下一阶段。MWD 和 RWDT 有四个阶段（称为阶段 0 至阶段 3）。看门狗定时器会循环工作（即从阶段 0 至阶段 3，再回到阶段 0）。

MWD 每个阶段的超时时间可用 `TIMG_WDTCONFIGi_REG`（*i* 的范围是 2 到 5）寄存器配置，RWDT 的超时时间可用 `RTC_CNTL_WDT_STGj_HOLD`（*j* 的范围是 0 到 3）字段配置。

值得注意的是，RWDT 在阶段 0 的超时时间 (T_{hold0}) 受 eFuse 寄存器 `EFUSE_RD_REPEAT_DATA1_REG` 的 `EFUSE_WDT_DELAY_SEL` 字段和 `RTC_CNTL_WDT_STG0_HOLD` 字段共同影响，关系如下：

$$T_{hold0} = RTC_CNTL_WDT_STG0_HOLD \ll (EFUSE_WDT_DELAY_SEL + 1)$$

其中， \ll 为左移运算符。

如某个阶段超时，下列超时动作之一将会执行：

- 触发中断
如阶段超时，中断被触发。
- CPU 复位 – 复位 CPU 核心
如阶段超时，复位 CPU 核心。
- 内核复位 – 复位主系统
如阶段超时，主系统（包括 MWD、CPU 和所有外设）复位。功耗管理单元和 RTC 外设不会复位。
- 系统复位 – 复位主系统、功耗管理单元和 RTC 外设
如阶段超时，主系统、功耗管理单元和 RTC 外设（详见章节 4 低功耗管理 (`RTC_CNTL`) [to be added later]）同时复位。此动作仅可在 RWDT 中实现。
- 关闭
该阶段对系统不产生影响。

MWD 所有阶段的超时动作均在 `TIMG_WDTCONFIG0_REG` 寄存器中配置。RWDT 的超时动作可在 `RTC_CNTL_WDTCONFIG0_REG` 寄存器配置。

11.2.2.3 写保护

看门狗定时器对于检测和处理系统或软件错误而言至关重要，不应轻易关闭（例如，因写寄存器位置错误而误将看门狗关闭）。因此，MWD 和 RWDT 引入写保护机制，防止看门狗因无意的写操作而被关闭或篡改。

写保护机制通过每个看门狗定时器的写密钥字段运行 (MWDT 看门狗使用 `TIMG_WDT_WKEY`, RWDT 看门狗使用 `RTC_CNTL_WDT_WKEY`)。必须向看门狗定时器的写密钥字段写入 `0x50D83AA1`, 才能修改其它看门狗寄存器。如果写密钥字段的值不是 `0x50D83AA1`, 任何试图向看门狗定时器寄存器 (除了向写密钥字段本身) 写值的操作都会被忽略。推荐按以下步骤访问看门狗定时器:

1. 将 `0x50D83AA1` 写入看门狗定时器的写密钥字段, 关闭写保护。
2. 根据需要修改看门狗, 如喂狗或改变配置。
3. 向看门狗定时器的写密钥字段上写入除 `0x50D83AA1` 以外的任意值, 重新使能写保护。

11.2.2.4 Flash 引导保护

在 flash 引导模式下, 定时器组 0 (见图 10-1 定时器组) 的 MWDT 和 RWDT 会默认使能。MWDT 的阶段 0 的默认超时动作为系统复位。RWDT 的阶段 0 超时动作也是系统复位 (复位主系统和 RTC)。引导后, 应将 `TIMG_WDT_FLASHBOOT_MOD_EN` 和 `RTC_CNTL_WDT_FLASHBOOT_MOD_EN` 位清零, 分别关闭 MWDT 和 RWDT 的 flash 引导保护。然后, 软件可以配置 MWDT 和 RWDT。

11.3 模拟看门狗定时器

超级看门狗 (SWD) 是模拟域的超低功耗电路, 可以防止系统在数字电路异常状态下运行, 并在必要时复位系统。SWD 包含一个看门狗电路, 需在每个超时阶段 (约不足一秒) 至少喂狗一次。该电路会在看门狗超时时间约 100 ms 之前发送 `WD_INTR` 信号提醒系统喂狗。

如果系统不回应 SWD 的喂狗请求, 看门狗超时, SWD 会产生系统电平信号 `SWD_RSTB`, 复位芯片上的整个数字电路。

11.3.1 主要特性

SWD 具有如下特性:

- 超低功耗
- 用中断提醒 SWD 即将超时
- 软件有多种专用的方法喂 SWD, 让 SWD 监控整个操作系统的工作状态

11.3.2 SWD 控制器

11.3.2.1 结构

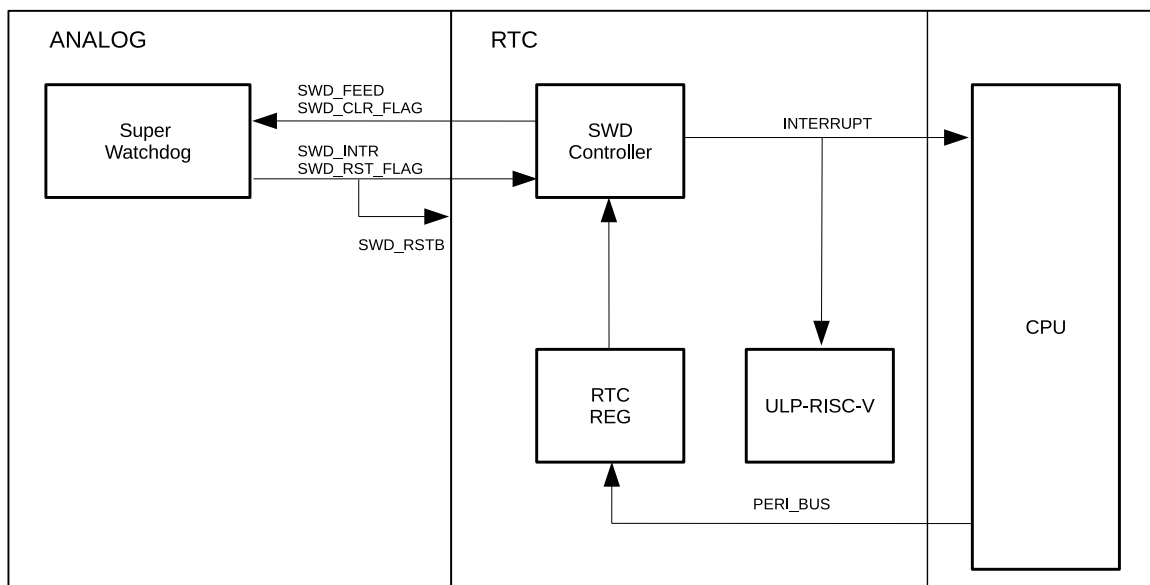


图 11-3. SWD 控制器结构

11.3.2.2 工作流程

正常状态下：

- SWD 控制器收到 SWD 的喂狗请求。
- SWD 控制器可以向主 CPU 或 ULP-RISC-V 发送中断。
- 主 CPU 可以决定是通过置位 `RTC_CNTL_SWD_FEED` 直接喂狗，还是发送中断让 ULP-RISC-V 置位该字段喂狗。
- CPU 或 ULP-RISC-V 喂狗时，需要先向 `RTC_CNTL_SWD_WKEY` 写 0x8F1D312A 关闭 SWD 控制器的写保护。这样做可以防止系统在数字电路异常状态下运行时误喂 SWD。
- 如将 `RTC_CNTL_SWD_AUTO_FEED_EN` 置 1，SWD 控制器也可配置为在不需要 CPU 或 ULP-RISC-V 干预的情况下喂 SWD。

复位后：

- 可查看 `RTC_CNTL_RESET_CAUSE_PROCPU[5:0]` 获知 CPU 复位原因。
如 `RTC_CNTL_RESET_CAUSE_PROCPU[5:0] == 0x12`，则表示上一次复位的原因是 SWD 复位。
- 置位 `RTC_CNTL_SWD_RST_FLAG_CLR` 清除 SWD 复位标志。

11.4 中断

看门狗定时器中断，请前往章节 10 定时器组 (TIMG) 的第 10.2.6 节 中断 查看。

11.5 寄存器

MWDT 寄存器是定时器组模块的一部分，在章节 10 定时器组 (TIMG) 的第 10.4 节 寄存器列表 中有详细描述。RWDT 和 SWD 寄存器是 RTC 模块的一部分，在章节 4 低功耗管理 (RTC_CNTL) [to be added later] 的第 6 节 寄

[寄存器列表](#) 中有详细描述。

PRELIMINARY

12 XTAL32K 看门狗定时器 (XTWDT)

12.1 概述

ESP32-C3 的 XTAL32K 看门狗定时器是用于检测 XTAL32K_CLK 时钟的工作状态，有 XTAL32K_CLK 停振监测，切换 RTC 时钟源等功能。当外部晶振 XTAL32K_CLK 作为 RTC 的 SLOW_CLK 源（时钟描述详见章节 6 [复位和时钟](#)），若 XTAL32K_CLK 时钟停振，XTAL32K 看门狗定时器会将 XTAL32K_CLK 替换为 RTC_CLK 的分频时钟 BACKUP32K_CLK 并发送中断（若芯片处于 Light-sleep 和 Deep-sleep 状态则唤醒 CPU），由软件重启 XTAL32K_CLK，并切回。

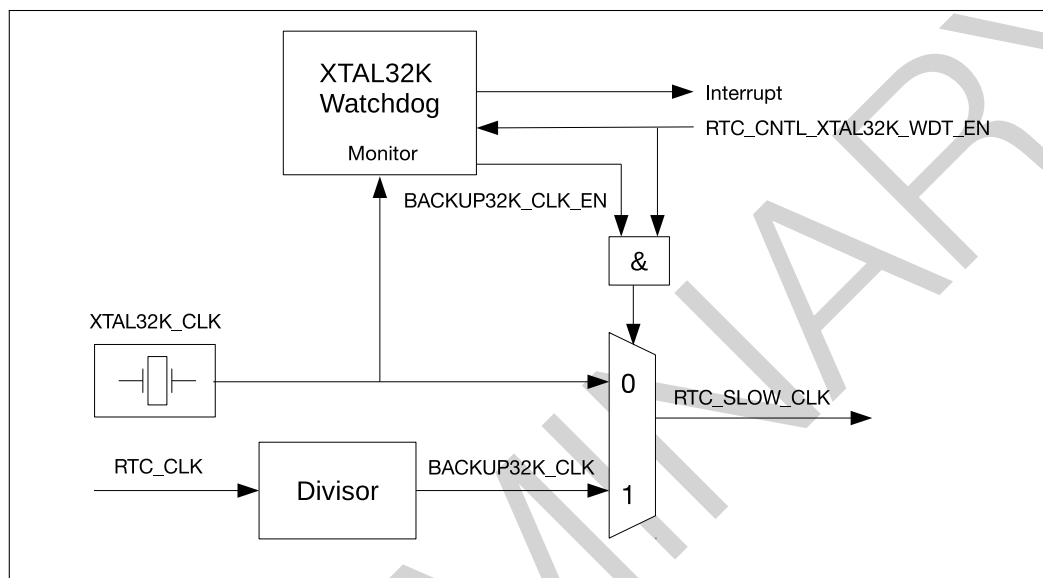


图 12-1. XTAL32K 看门狗定时器

12.2 主要特性

12.2.1 XTAL32K 看门狗定时器的中断及唤醒

XTAL32K 看门狗定时器监控到 XTAL32K_CLK 停振时，将发起停振中断 RTC_XTAL32K_DEAD_INT（中断描述详见章节 4 [低功耗管理 \(RTC_CNTL\) \[to be added later\]](#)），如果 CPU 处于 Light-sleep 和 Deep-sleep 状态，将唤醒 CPU。

12.2.2 BACKUP32K_CLK

XTAL32K 看门狗定时器监控到 XTAL32K_CLK 停振后，将使用 RTC_CLK 的分频时钟 BACKUP32K_CLK（频率约为 32 kHz）替代 XTAL32K_CLK 作为 RTC 的 SLOW_CLK 维持系统继续正常工作。

12.3 功能描述

12.3.1 工作流程

1. 使能 [RTC_CNTL_XTAL32K_WDT_EN](#)，XTAL32K 看门狗定时器将由空闲状态转入计数状态，看门狗的计数器（工作时钟为 RTC_CLK）在检测到 XTAL32K 的时钟上升沿时将被清零，否则将持续计数；当计数器的值达到 [RTC_CNTL_XTAL32K_WDT_TIMEOUT](#) 时，发出中断/唤醒信号，随后计数器复位。

- 如果 `RTC_CNTL_XTAL32K_AUTO_BACKUP` 已置 1 且步骤 1 已完成, XTAL32K 看门狗定时器会自动开启 `BACKUP32K_CLK`, 替换 RTC 的慢速时钟源 `RTC SLOW_CLK`, 保证系统能够正常运行, 以及工作在 RTC 慢速时钟 `RTC SLOW_CLK` 的定时器 (如 `RTC_TIMER` 等) 能够保持计时准确性。时钟频率的配置参见 12.3.2。
- 软件通过 `RTC_CNTL_XPD_XTAL_32K` 位开关 `XTAL32K_CLK` 的 XPD (no power-down 的缩写, 意为不要关闭电源) 信号来重启 `XTAL32K_CLK`, 然后通过将 `RTC_CNTL_XTAL32K_WDT_EN` 位设置为 0 (`BACKUP32K_CLK_EN` 会随之自动清零), RTC 的 `SLOW_CLK` 时钟源将从 `BACKUP32K_CLK` 切回到 `XTAL32K_CLK`。若是芯片处于 Light-sleep 和 Deep-sleep 状态, 则 XTAL32K 看门狗定时器将唤醒 CPU, 完成上述操作。

12.3.2 BACKUP32K_CLK 实现原理

由于 `RTC_CLK` 的时钟频率存在芯片差异, 所以为保证 `BACKUP32K_CLK` 生效期间, `RTC_TIMER` 等使用 RTC 的 `SLOW_CLK` 工作的定时器依然能够准确计时, 需要根据 `RTC_CLK` (详见章节 4 低功耗管理 (*RTC_CNTL*) [to be added later]) 的实际频率, 可通过配置 `RTC_CNTL_XTAL32K_CLK_FACTOR_REG` 调整 `BACKUP32K_CLK` 的分频系数。该寄存器的每个字节对应一个分频因子 ($x_0 \sim x_7$)。 `BACKUP32K_CLK` 的分频器是一个分母恒为 4 的小数分频器, 算法如下:

$$f_{back_clk}/4 = f_{rtc_clk}/S$$

$$S = x_0 + x_1 + \dots + x_7$$

其中 f_{back_clk} 为分频后的 `BACKUP32K_CLK` 目标频率为 32.768 kHz; f_{rtc_clk} 为当前 `RTC_CLK` 的实际频率; $x_0 \sim x_7$ 分别对应四个 `BACKUP32K` 时钟信号的高低电平的脉宽, 单位为 `RTC_CLK` 的周期。

12.3.3 BACKUP32K_CLK 分频因子配置方法

根据 12.3.2 小节的分频原理描述, 可以通过以下步骤计算并完成分频因子的配置:

- 根据 `RTC_CLK` 的频率以及 `BACKUP32K` 的目标分频频率计算出分频因子的总和 S ;
- 计算出分频器的整数部分, $N = f_{rtc_clk}/f_{back_clk}$;
- 因为 `BACKUP32K` 的分频因子是单个脉宽 (高或低电平), 所以需要将分频系数的整数部分分成两份, 计算分频因子的整数部分, $M = N/2$;
- 根据 M 和 S 确定 $x_n = M$ 以及 $x_n = M + 1$ 的个数, $M + 1$ 即为分频因子的小数部分。

例如, `RTC_CLK` 的时钟频率为 163 kHz, 则 $f_{rtc_clk} = 163000$, $f_{back_clk} = 32768$, $S = 20$, $M = 2$, 所以满足条件的 $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\} = \{2, 3, 2, 3, 2, 3, 2, 3\}$, `BACKUP32K_CLK` 的时钟频率为 32.6 kHz。

13 系统寄存器 (SYSREG)

13.1 概述

ESP32-C3 集成了丰富的外设，且允许对不同外设模块进行独立控制，从而在保持最佳性能的同时将功耗降至最低。具体来说，ESP32-C3 设计了一系列系统配置寄存器，用于芯片的时钟管理（时钟门控）、功耗管理、外设模块及核心模块配置。本章将简要例举这些系统配置寄存器及其功能。

13.2 主要特性

ESP32-C3 的系统寄存器可用于控制以下外设和模块：

- 系统和存储器
- 时钟
- 软件中断
- 低功耗管理寄存器
- 外设时钟门控和复位

13.3 功能描述

13.3.1 系统和存储器寄存器

13.3.1.1 内部存储器

以下寄存器用以控制 ESP32-C3 内存的功耗，具体来说：

- 在寄存器 `APB_CTRL_CLKGATE_FORCE_ON_REG` 中：
 - 设置 `APB_CTRL_ROM_CLKGATE_FORCE_ON` 的相应位可分别控制 Internal ROM 0 和 Internal ROM 1 的时钟门控；
 - 设置 `APB_CTRL_SRAM_CLKGATE_FORCE_ON` 的相应位可分别控制 Internal SRAM 的时钟门控。
 - 配置为 1 时，ROM 或 SRAM 内存的时钟门控始终开启；配置为 0 时，则 ROM 或 SRAM 内存的时钟门控在被访问时自动打开，没有访问时自动关闭。因此，建议将本寄存器配置为 0，以降低功耗。
- 在寄存器 `APB_CTRL_MEM_POWER_DOWN_REG` 中：
 - 设置 `APB_CTRL_ROM_POWER_DOWN` 的相应位可分别控制 Internal ROM 0 和 Internal ROM 1 进入 Retention 状态；
 - 设置 `APB_CTRL_SRAM_POWER_DOWN` 的相应位可分别控制 Internal SRAM 进入 Retention 状态。
 - Retention 状态是存储器的一种低功耗模式。在此状态下，存储器中的数据不会丢失，但是不允许访问，因此可降低功耗。所以，如果用户在一段时间内不会访问某些存储器，也可以配置 PD 寄存器让这些存储器进入 Retention 状态，以降低功耗。
- 在寄存器 `APB_CTRL_MEM_POWER_UP_REG` 中：
 - 默认情况下，芯片进入 Light-sleep 时会让所有的存储器进入 Retention 状态。

- 设置 [APB_CTRL_ROM_POWER_UP](#) 的相应位可分别控制 Internal ROM 0 和 Internal ROM 1 在芯片进入 Light-sleep 时不会进入 Retention 状态；
- 设置 [APB_CTRL_SRAM_POWER_UP](#) 的相应位可分别控制 Internal SRAM 在芯片进入 Light-sleep 时不会进入 Retention 状态。

更多有关内存功耗控制位的对应关系，请见下方表 13-1。

表 13-1. 内存功耗控制位

内存	低地址 1	高地址 1	低地址 2	高地址 2	控制域
ROM 0	0x4000_0000	0x4003_FFFF	-	-	Bit0
ROM 1	0x4004_0000	0x4005_FFFF	0x3FF0_0000	0x3FF1_FFFF	Bit1
SRAM Block 0	0x4037_C000	0x4037_FFFF	-	-	Bit0
SRAM Block 1	0x4038_0000	0x4039_FFFF	0x3FC8_0000	0x3FC9_FFFF	Bit1
SRAM Block 2	0x403A_0000	0x403B_FFFF	0x3FCA_0000	0x3FCB_FFFF	Bit2
SRAM Block 3	0x403C_0000	0x403D_FFFF	0x3FCC_0000	0x3FCD_FFFF	Bit3

更多信息，请见章节 3 系统和存储器。

13.3.1.2 片外存储器

[SYSTEM_EXTERNAL_DEVICE_ENCRYPT_DECRYPT_CONTROL_REG](#) 可用于控制外部存储的加解密配置，详情请见章节 5 片外存储器加密与解密 (*XTS_AES*) [to be added later]。

13.3.1.3 RSA 存储器

[SYSTEM_RSA_PD_CTRL_REG](#) 可控制 RSA 加速器中的 SRAM 存储器。

- [SYSTEM_RSA_MEM_PD](#) 置 1 控制 RSA 存储器进入 Retention 状态。此位的优先级最低，其设置可被 [SYSTEM_RSA_MEM_FORCE_PU](#) 覆盖。当 [数字签名 \(DS\)](#) 使用 RSA 加速器时，此位无效。
- [SYSTEM_RSA_MEM_FORCE_PU](#) 置 1 控制 RSA 存储器再芯片进入 Light sleep 时不会进入 Retention 状态。此位的优先级第二高，可覆盖 [SYSTEM_RSA_MEM_PD](#) 的设置。
- [SYSTEM_RSA_MEM_FORCE_PD](#) 置 1 控制 RSA 存储器进入 Retention 状态。此位的优先级最高，可以覆盖 [SYSTEM_RSA_MEM_FORCE_PU](#) 的设置。

13.3.2 时钟配置寄存器

以下系统寄存器用于系统和外设时钟源和时钟频率的配置。更多信息，请见章节 6 复位和时钟。

- [SYSTEM_CPU_PER_CONF_REG](#)
- [SYSTEM_SYSCLK_CONF_REG](#)
- [SYSTEM_BT_LPCK_DIV_FRAC_REG](#)

13.3.3 中断信号寄存器

以下系统寄存器用于产生中断信号，经过配置可通过中断矩阵，产生不同的 CPU 外设中断。当以下寄存器写为 1 时，会产生中断信号，可用于软件自己控制中断的产生。更多信息，请见章节 8 中断矩阵 (*INTMTRX*)。

- [SYSTEM_CPU_INTR_FROM_CPU_0_REG](#)
- [SYSTEM_CPU_INTR_FROM_CPU_1_REG](#)
- [SYSTEM_CPU_INTR_FROM_CPU_2_REG](#)
- [SYSTEM_CPU_INTR_FROM_CPU_3_REG](#)

13.3.4 低功耗管理寄存器

以下系统寄存器用于低功耗管理。更多信息，请见章节 4 低功耗管理 ([RTC_CNTL](#)) [to be added later]。

- [SYSTEM_RTC_FASTMEM_CONFIG_REG](#): 用于配置 RTC 快速内存的 CRC;
- [SYSTEM_RTC_FASTMEM_CRC_REG](#): 配置 CRC 校验值。

13.3.5 外设时钟门控和复位寄存器

以下系统寄存器用于控制外设时钟门控和复位，相应位分别为不同外设的门控使能和复位使能，详见下方表 13-2。

- [SYSTEM_CACHE_CONTROL_REG](#)
- [SYSTEM_PERIP_CLK_EN0_REG](#)
- [SYSTEM_PERIP_RST_EN0_REG](#)
- [SYSTEM_PERIP_CLK_EN1_REG](#)
- [SYSTEM_PERIP_RST_EN1_REG](#)

表 13-2. 外设时钟门控与复位控制位

组件	时钟使能位 ¹	复位使能位 ²³
Cache 控制	SYSTEM_CACHE_CONTROL_REG	
DCACHE	SYSTEM_DCACHE_CLK_ON	SYSTEM_DCACHE_RESET
ICACHE	SYSTEM_ICACHE_CLK_ON	SYSTEM_ICACHE_RESET
CPU	SYSTEM_CPU_PERI_CLK_EN_REG	SYSTEM_CPU_PERI_RST_EN_REG
DEBUG_ASSIST	SYSTEM_CLK_EN_ASSIST_DEBUG	SYSTEM_RST_EN_ASSIST_DEBUG
外设	SYSTEM_PERIP_CLK_EN0_REG	SYSTEM_PERIP_RST_EN0_REG
TIMER	SYSTEM_TIMERS_CLK_EN	SYSTEM_TIMERS_RST
SPI0 / SPI1	SYSTEM_SPI01_CLK_EN	SYSTEM_SPI01_RST
UART0	SYSTEM_UART_CLK_EN	SYSTEM_UART_RST
UART1	SYSTEM_UART1_CLK_EN	SYSTEM_UART1_RST
SPI2	SYSTEM_SPI2_CLK_EN	SYSTEM_SPI2_RST
I2C0	SYSTEM_EXT0_CLK_EN	SYSTEM_EXT0_RST
UHCI0	SYSTEM_UHCI0_CLK_EN	SYSTEM_UHCI0_RST
RMT	SYSTEM_RMT_CLK_EN	SYSTEM_RMT_RST
LED PWM 控制器	SYSTEM_LEDC_CLK_EN	SYSTEM_LEDC_RST
Timer Group0	SYSTEM_TIMERGROUP_CLK_EN	SYSTEM_TIMERGROUP_RST
Timer Group1	SYSTEM_TIMERGROUP1_CLK_EN	SYSTEM_TIMERGROUP1_RST
TWAI 控制器	SYSTEM_CAN_CLK_EN	SYSTEM_CAN_RST

接下页

表 13-2 – 接下页

组件	时钟使能位 ¹	复位使能位 ²³
USB_DEVICE	SYSTEM_USB_DEVICE_CLK_EN	SYSTEM_USB_DEVICE_RST
UART MEM	SYSTEM_UART_MEM_CLK_EN ⁴	SYSTEM_UART_MEM_RST
APB SARADC	SYSTEM_APB_SARADC_CLK_EN	SYSTEM_APB_SARADC_RST
ADC 控制器	SYSTEM_ADC2_ARB_CLK_EN	SYSTEM_ADC2_ARB_RST
System 定时器	SYSTEM_SYSTIMER_CLK_EN	SYSTEM_SYSTIMER_RST
加速器	SYSTEM_PERIP_CLK_EN1_REG	SYSTEM_PERIP_RST_EN1_REG
TSENS	SYSTEM_TSENS_CLK_EN	SYSTEM_TSENS_RST
DMA	SYSTEM_DMA_CLK_EN	SYSTEM_DMA_RST ⁵
HMAC	SYSTEM_CRYPTO_HMAC_CLK_EN	SYSTEM_CRYPTO_HMAC_RST ⁶
数字签名	SYSTEM_CRYPTO_DS_CLK_EN	SYSTEM_CRYPTO_DS_RST ⁷
RSA 加速器	SYSTEM_CRYPTO_RSA_CLK_EN	SYSTEM_CRYPTO_RSA_RST
SHA 加速器	SYSTEM_CRYPTO_SHA_CLK_EN	SYSTEM_CRYPTO_SHA_RST
AES 加速器	SYSTEM_CRYPTO_AES_CLK_EN	SYSTEM_CRYPTO_AES_RST

¹ 时钟控制寄存器相应比特置 1 表示打开对应时钟，置 0 表示关闭对应时钟。

² 复位寄存器响应比特置 1 表示使能复位状态，对应外设进行复位，置 0 表示关闭复位状态，对应外设正常工作。

³ 复位寄存器无法通过硬件清除，因此软件将外设复位后需要清除复位寄存器。

⁴ UART 存储器为所有 UART 外设所共用，因此只要有一个 UART 在工作，UART 存储器就不能处于门控状态。

⁵ 当外设需要通过 DMA 进行数据传输时，比如 UCHIO、SPI、I2S、LCD_CAM、AES、SHA、ADC 等，需要同时将 DMA 的时钟打开。

⁶ 该位复位后，SHA 加速器也会同时被复位。

⁷ 该位复位后，AES 加速器、SHA 加速器和 RSA 加速器也会同时被复位。

13.4 寄存器列表

本小节的所有地址均为相对于系统寄存器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问权限
外设时钟控制寄存器			
SYSTEM_CPU_PERI_CLK_EN_REG	CPU 外设时钟使能寄存器	0x0000	R/W
SYSTEM_CPU_PERI_RST_EN_REG	CPU 外设时钟复位寄存器	0x0004	读写
SYSTEM_PERIP_CLK_EN0_REG	系统外设时钟使能寄存器 0	0x0010	读写
SYSTEM_PERIP_CLK_EN1_REG	系统外设时钟使能寄存器 1	0x0014	读写
SYSTEM_PERIP_RST_EN0_REG	系统外设时钟复位寄存器 0	0x0018	读写
SYSTEM_PERIP_RST_EN1_REG	系统外设时钟复位寄存器 1	0x001C	读写
SYSTEM_CACHE_CONTROL_REG	Cache 时钟控制寄存器	0x0040	读写
时钟配置寄存器			
SYSTEM_CPU_PER_CONF_REG	CPU 时钟配置寄存器	0x0008	读写
SYSTEM_SYSCLK_CONF_REG	系统时钟配置寄存器	0x0058	可变
低功耗控制寄存器			
SYSTEM_BT_LPCK_DIV_FRAC_REG	低功耗时钟配置寄存器 1	0x0024	读写
SYSTEM_RTC_FASTMEM_CONFIG_REG	快速内存 CRC 配置寄存器	0x0048	varies
SYSTEM_RTC_FASTMEM_CRC_REG	快速内存 CRC 结果寄存器	0x004C	只读
CPU 中断控制寄存器			
SYSTEM_CPU_INTR_FROM_CPU_0_REG	CPU 中断控制寄存器 0	0x0028	读写
SYSTEM_CPU_INTR_FROM_CPU_1_REG	CPU 中断控制寄存器 1	0x002C	读写
SYSTEM_CPU_INTR_FROM_CPU_2_REG	CPU 中断控制寄存器 2	0x0030	读写
SYSTEM_CPU_INTR_FROM_CPU_3_REG	CPU 中断控制寄存器 3	0x0034	读写
系统和内存控制寄存器			
SYSTEM_RSA_PD_CTRL_REG	RSA 内存掉电寄存器	0x0038	读写
SYSTEM_EXTERNAL_DEVICE_ENCRYPT_DECRYPT_CONTROL_REG	外部内存加解密控制寄存器	0x0044	读写
时钟门控控制寄存器			
SYSTEM_CLOCK_GATE_REG	时钟门控寄存器	0x0054	读写
日期寄存器			
SYSTEM_DATE_REG	版本寄存器	0x0FFC	读写

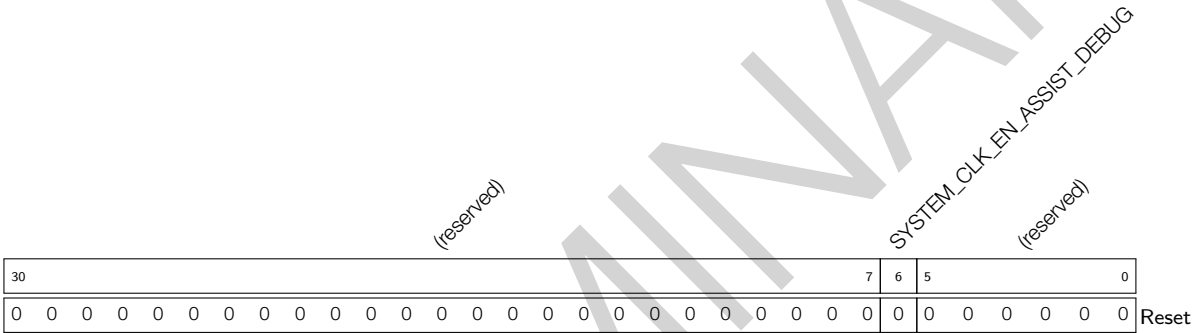
本小节的所有地址均为相对于 apb 控制寄存器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问权限
配置寄存器			
APB_CTRL_CLKGATE_FORCE_ON_REG	内存时钟门控使能寄存器	0x00A4	R/W
APB_CTRL_MEM_POWER_DOWN_REG	内存控制寄存器	0x00A8	R/W
APB_CTRL_MEM_POWER_UP_REG	内存控制寄存器	0x00AC	R/W

13.5 寄存器

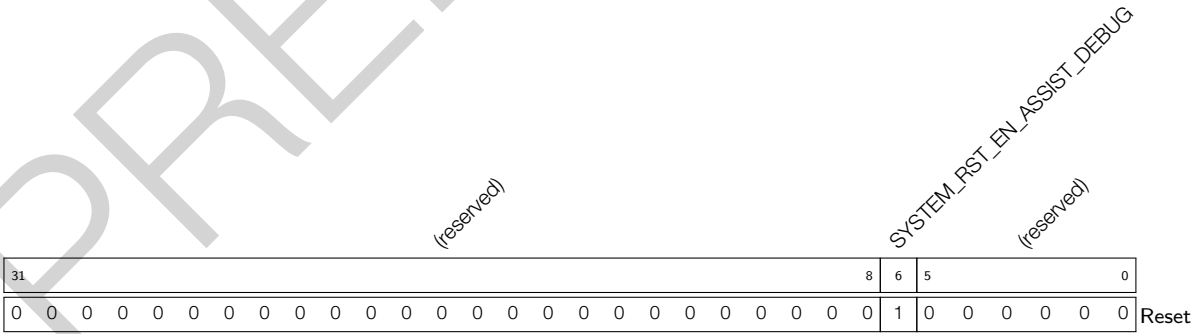
本小节的所有地址均为相对于系统寄存器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 13.1. SYSTEM_CPU_PERI_CLK_EN_REG (0x0000)



SYSTEM_CLK_EN_ASSIST_DEBUG 置 1 使能 ASSIST_DEBUG 时钟。更多信息，请见章节 14 辅助调试 (Debug Assist)。 (R/W)

Register 13.2. SYSTEM_CPU_PERI_RST_EN_REG (0x0004)



SYSTEM_RST_EN_ASSIST_DEBUG 置 1 复位 ASSIST_DEBUG 时钟。更多信息，请见章节 14 辅助调试 (Debug Assist)。 (R/W)

Register 13.3. SYSTEM_PERIP_CLK_EN0_REG (0x0010)

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

SYSTEM_TIMERS_CLK_EN 置 1 使能 TIMERS 时钟。(R/W)

SYSTEM_SPI01_CLK_EN 置 1 使能 SPI0 / SPI1 时钟。(R/W)

SYSTEM_UART_CLK_EN 置 1 使能 UART 时钟。(R/W)

SYSTEM_UART1_CLK_EN 置 1 使能 UART1 时钟。(R/W)

SYSTEM_SPI2_CLK_EN 置 1 使能 SPI2 时钟。(R/W)

SYSTEM_EXT0_CLK_EN 置 1 使能 I2C_EXT0 时钟。(R/W)

SYSTEM_UHCI0_CLK_EN 置 1 使能 UHCI0 时钟。(R/W)

SYSTEM_RMT_CLK_EN 置 1 使能 RMT 时钟。(R/W)

SYSTEM_LEDC_CLK_EN 置 1 使能 LEDC 时钟。(R/W)

SYSTEM_TIMERGROUP_CLK_EN 置 1 使能 TIMER GROUP 时钟。(R/W)

SYSTEM_TIMERGROUP1_CLK_EN 置 1 使能 TIMERGROUP1 时钟。(R/W)

SYSTEM_CAN_CLK_EN 置 1 使能 TWAI 时钟。(R/W)

SYSTEM_I2S1_CLK_EN 置 1 使能 I2S1 时钟。(R/W)

SYSTEM_USB_DEVICE_CLK_EN 置 1 使能 USB DEVICE 时钟。(R/W)

SYSTEM_UART_MEM_CLK_EN 置 1 使能 UART_MEM 时钟。(R/W)

SYSTEM_SPI3_DMA_CLK_EN 置 1 使能 SPI3 DMA 时钟。(R/W)

SYSTEM_APB_SARADC_CLK_EN 置 1 使能 APB_SARADC 时钟。(R/W)

SYSTEM_SYSTIMER_CLK_EN 置 1 使能 SYSTEMTIMER 时钟。(R/W)

SYSTEM_ADC2_ARB_CLK_EN 置 1 使能 ADC2_ARB 时钟。(R/W)

Register 13.4. SYSTEM_PERIP_CLK_EN1_REG (0x0014)

(reserved)																								SYSTEM_TSENS_CLK_EN				(reserved)				SYSTEM_DMA_CLK_EN		SYSTEM_CRYPT_HMAC_CLK_EN		SYSTEM_CRYPT_DS_CLK_EN		SYSTEM_CRYPT_RSA_CLK_EN		SYSTEM_CRYPT_SHA_CLK_EN		SYSTEM_CRYPT_AES_CLK_EN	
31																								11				10	9	7		6	5	4	3	2	1	0					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset															

SYSTEM_CRYPT_AES_CLK_EN 置 1 使能 AES 时钟。(R/W)

SYSTEM_CRYPT_SHA_CLK_EN 置 1 使能 SHA 时钟。(R/W)

SYSTEM_CRYPT_RSA_CLK_EN 置 1 使能 RSA 时钟。(R/W)

SYSTEM_CRYPT_DS_CLK_EN 置 1 使能 DS 时钟。(R/W)

SYSTEM_CRYPT_HMAC_CLK_EN 置 1 使能 HMAC 时钟。(R/W)

SYSTEM_DMA_CLK_EN 置 1 使能 DMA 时钟。(R/W)

SYSTEM_TSENS_CLK_EN 置 1 使能 TSENS 时钟。(R/W)

Register 13.5. SYSTEM_PERIP_RST_EN0_REG (0x0018)

<div>(reserved)</div> <div>SYSTEM_ADC2_ARB_RST</div> <div>SYSTEM_SYSTIMER_RST</div> <div>SYSTEM_APB_SARADC_RST</div> <div>SYSTEM_SPI3_DMA_RST</div> <div>(reserved)</div> <div>SYSTEM_UART_MEM_RST</div> <div>SYSTEM_USB_DEVICE_RST</div> <div>SYSTEM_I2S1_RST</div> <div>(reserved)</div> <div>SYSTEM_CAN_RST</div> <div>(reserved)</div> <div>SYSTEM_TIMERGROUP1_RST</div> <div>(reserved)</div> <div>SYSTEM_TIMERGROUP_RST</div> <div>(reserved)</div> <div>SYSTEM_LEDC_RST</div> <div>(reserved)</div> <div>SYSTEM_RMT_RST</div> <div>SYSTEM_UHCI0_RST</div> <div>SYSTEM_EXT0_RST</div> <div>SYSTEM_SPI2_RST</div> <div>SYSTEM_UART1_RST</div> <div>(reserved)</div> <div>SYSTEM_UART_RST</div> <div>SYSTEM_SPI01_RST</div> <div>SYSTEM_TIMERS_RST</div>																																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18		16	15	14	13	12	11	10	9	8	7	6	5	4		3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

Reset

SYSTEM_TIMERS_RST 置 1 复位 TIMERS。(R/W)**SYSTEM_SPI01_RST** 置 1 复位 SPI01。(R/W)**SYSTEM_UART_RST** 置 1 复位 UART。(R/W)**SYSTEM_UART1_RST** 置 1 复位 UART1。(R/W)**SYSTEM_SPI2_RST** 置 1 复位 SPI2。(R/W)**SYSTEM_EXT0_RST** 置 1 复位 I2C_EXT0。(R/W)**SYSTEM_UHCI0_RST** 置 1 复位 UHCI0。(R/W)**SYSTEM_RMT_RST** 置 1 复位 RMT。(R/W)**SYSTEM_LEDC_RST** 置 1 复位 LEDC。(R/W)**SYSTEM_TIMERGROUP_RST** 置 1 复位 TIMERGROUP。(R/W)**SYSTEM_TIMERGROUP1_RST** 置 1 复位 TIMERGROUP1。(R/W)**SYSTEM_CAN_RST** 置 1 复位 TAWI。(R/W)**SYSTEM_I2S1_RST** 置 1 复位 I2S1。(R/W)**SYSTEM_USB_DEVICE_RST** 置 1 复位 USB DEVICE。(R/W)**SYSTEM_UART_MEM_RST** 置 1 复位 UART_MEM。(R/W)**SYSTEM_SPI3_DMA_RST** 置 1 复位 SPI3。(R/W)**SYSTEM_APB_SARADC_RST** 置 1 复位 APB_SARADC。(R/W)**SYSTEM_SYSTIMER_RST** 置 1 复位 SYSTIMER。(R/W)**SYSTEM_ADC2_ARB_RST** 置 1 复位 ADC2_ARB。(R/W)

Register 13.6. SYSTEM_PERIP_RST_EN1_REG (0x001C)

(reserved)																				SYSTEM_TSENS_RST		(reserved)		SYSTEM_DMA_RST		SYSTEM_CRYPTO_HMAC_RST		SYSTEM_CRYPTO_DS_RST		SYSTEM_CRYPTO_RSA_RST		SYSTEM_CRYPTO_SHA_RST		(reserved)				
31											11	10	9	7	6	5	4	3	2	1	0																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	Reset

SYSTEM_CRYPTO_AES_RST 置 1 复位 CRYPTO_AES。(R/W)

SYSTEM_CRYPTO_SHA_RST 置 1 复位 CRYPTO_SHA。(R/W)

SYSTEM_CRYPTO_RSA_RST 置 1 复位 CRYPTO_RSA。(R/W)

SYSTEM_CRYPTO_DS_RST 置 1 复位 CRYPTO_DS。(R/W)

SYSTEM_CRYPTO_HMAC_RST 置 1 复位 CRYPTO_HMAC。(R/W)

SYSTEM_DMA_RST 置 1 复位 DMA。(R/W)

SYSTEM_TSENS_RST 置 1 复位 TSENS。(R/W)

Register 13.7. SYSTEM_CACHE_CONTROL_REG (0x0040)

(reserved)																												SYSTEM_DCACHE_RESET					SYSTEM_DCACHE_CLK_ON					SYSTEM_ICACHE_RESET					SYSTEM_ICACHE_CLK_ON				
31																												4					3	2	1	0											
0																												0					0	1	0	1	Reset										

SYSTEM_ICACHE_CLK_ON 置 1 使能 i-cache 时钟。(R/W)

SYSTEM_ICACHE_RESET 置 1 复位 i-cache。(R/W)

SYSTEM_DCACHE_CLK_ON 置 1 使能 d-cache 时钟。(R/W)

SYSTEM_DCACHE_RESET 置 1 复位 d-cache。(R/W)

Register 13.8. SYSTEM_CPU_PER_CONF_REG (0x0008)

(reserved)																								SYSTEM_CPU_WAITI_DELAY_NUM				SYSTEM_CPU_WAIT_MODE_FORCE_ON				SYSTEM_PLL_FREQ_SEL				SYSTEM_CPUPERIOD_SEL			
31																								8	7	4				3	2	1	0						
0 0																								0x0				1	1	0	Reset								

SYSTEM_CPUPERIOD_SEL 选择 CPU 时钟频率。具体配置，请见章节 6 复位和时钟 中的表 6-4。
(R/W)

SYSTEM_PLL_FREQ_SEL 选择 PLL 时钟频率。具体配置，请见章节 6 复位和时钟 中的表 6-4。(R/W)

SYSTEM_CPU_WAIT_MODE_FORCE_ON 置 1 强制打开 CPU 等待中断模式下的门控时钟。通常情况下，CPU 执行 WFI 指令后会进入等待中断模式。在此模式下 CPU 的时钟门控一直处于关闭状态，直到中断产生，因此可降低功耗。若此位置 1，CPU 的门控时钟会被强制打开，不受 WFI 指令的影响。(R/W)

SYSTEM_CPU_WAITI_DELAY_NUM 设置 CPU 在收到 WFI 指令后进入 CPU 等待中断模式后，关闭 CPU 的门控时钟需要的等待周期。(R/W)

Register 13.9. SYSTEM_BT_LPCLK_DIV_FRAC_REG (0x0024)

(reserved)								SYSTEM_LPCLK_SEL_RTC_SLOW																(reserved)							
31	29	28	27	26	25	24	23																	0							
0	0	0	0	0	0	1	0	1																Reset							

SYSTEM_LPCLK_SEL_RTC_SLOW 选择 RTC 慢速时钟为低功耗时钟。(R/W)

SYSTEM_LPCLK_SEL_8M 选择 8 MHz 时钟为低功耗时钟。(R/W)

SYSTEM_LPCLK_SEL_XTAL 选择 XTAL 时钟为低功耗时钟。(R/W)

SYSTEM_LPCLK_SEL_XTAL32K 选择 XTAL32K 时钟为低功耗时钟。(R/W)

SYSTEM_LPCLK_RTC_EN 置 1 使能 RTC 低功耗时钟。(R/W)

Register 13.10. SYSTEM_SYSCLK_CONF_REG (0x0058)

(reserved)																SYSTEM_CLK_XTAL_FREQ							SYSTEM_SOC_CLK_SEL							SYSTEM_PRE_DIV_CNT																																		
31																19							18							12							11							10							9							0						
0																0							0							0x1							Reset																											

SYSTEM_PRE_DIV_CNT 设置预分频器计数器。具体配置，请见章节 6 复位和时钟 中的表 6-4。(R/W)

SYSTEM_SOC_CLK_SEL 选择 SoC 时钟。具体配置，请见章节 6 复位和时钟 中的表 6-2。(R/W)

SYSTEM_CLK_XTAL_FREQ 读取晶振频率（单位：MHz）。(只读)

Register 13.11. SYSTEM_RTC_FASTMEM_CONFIG_REG (0x0048)

SYSTEM_RTC_MEM_CRC_FINISH																SYSTEM_RTC_MEM_CRC_LEN																SYSTEM_RTC_MEM_CRC_ADDR																SYSTEM_RTC_MEM_CRC_START																(reserved)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31	30																														20	19	9																														8	7	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
0	0x7ff																														0x0																														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

SYSTEM_RTC_MEM_CRC_START 置 1 启动 RTC 内存的 CRC 校验。(R/W)

SYSTEM_RTC_MEM_CRC_ADDR 设置 CRC 校验的 RTC 存储地址。(R/W)

SYSTEM_RTC_MEM_CRC_LEN 设置用于 CRC 校验的 RTC 存储长度（基于起始地址）。(R/W)

SYSTEM_RTC_MEM_CRC_FINISH 储存 RTC 存储 CRC 校验状态。高电平表示校验完成，低电平表示校验未完成。(RO)

Register 13.12. SYSTEM_RTC_FASTMEM_CRC_REG (0x004C)

SYSTEM_RTC_MEM_CRC_RES																															
31																															0
0																															
Reset																															

SYSTEM_RTC_MEM_CRC_RES 储存 RTC 存储的 CRC 校验结果。(RO)

Register 13.13. SYSTEM_CPU_INTR_FROM_CPU_0_REG (0x0028)

(reserved)																															SYSTEM_CPU_INTR_FROM_CPU_0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SYSTEM_CPU_INTR_FROM_CPU_0 置 1 生成 CPU 中断 0。该位需在 ISR 过程中由软件清 0。(R/W)

Register 13.14. SYSTEM_CPU_INTR_FROM_CPU_1_REG (0x002C)

(reserved)																															SYSTEM_CPU_INTR_FROM_CPU_1		
31																															1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

SYSTEM_CPU_INTR_FROM_CPU_1 置 1 生成 CPU 中断 1。该位需在 ISR 过程中由软件清 0。(R/W)

243
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

243
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

243
反馈文档意见

243
反馈文档意见

Register 13.17. SYSTEM_RSA_PD_CTRL_REG (0x0038)

(reserved)																																SYSTEM_RSA_MEM_FORCE_PD SYSTEM_RSA_MEM_FORCE_PU SYSTEM_RSA_MEM_PD			
31																															3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset	

- SYSTEM_RSA_MEM_PD** 置 1 控制 RSA 存储器进入 Retention 状态。此位的优先级最低，其设置可被 **SYSTEM_RSA_MEM_FORCE_PU** 覆盖。当数字签名占用 RSA 加速器时，该位无效。(R/W)
- SYSTEM_RSA_MEM_FORCE_PU** 置 1 控制 RSA 存储器再芯片进入 Light sleep 时不会进入 Retention 状态。此位的优先级第二高，可覆盖 **SYSTEM_RSA_MEM_PD** 的设置。(R/W)
- SYSTEM_RSA_MEM_FORCE_PD** 置 1 控制 RSA 存储器进入 Retention 状态。此位的优先级最高，可以覆盖 **SYSTEM_RSA_MEM_FORCE_PU** 的设置。(R/W)

Register 13.18. SYSTEM_EXTERNAL_DEVICE_ENCRYPT_DECRYPT_CONTROL_REG (0x0044)

(reserved)																												SYSTEM_ENABLE_DOWNLOAD_MANUAL_ENCRYPT SYSTEM_ENABLE_DOWNLOAD_G0CB_DECRYPT SYSTEM_ENABLE_DOWNLOAD_DB_ENCRYPT SYSTEM_ENABLE_SPI_MANUAL_ENCRYPT				
31																												4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset	

- SYSTEM_ENABLE_SPI_MANUAL_ENCRYPT** 置 1 在 SPI Boot 模式下使能手动加密 (Manual Encryption)。(R/W)
- SYSTEM_ENABLE_DOWNLOAD_DB_ENCRYPT** 置 1 在 Download Boot 模式下使能自动加密 (Auto Encryption)。(R/W)
- SYSTEM_ENABLE_DOWNLOAD_G0CB_DECRYPT** 置 1 在 Download Boot 模式下使能自动解密 (Auto Decryption)。(R/W)
- SYSTEM_ENABLE_DOWNLOAD_MANUAL_ENCRYPT** 置 1 在 Download Boot 模式下使能手动加密 (Manual Encryption)。(R/W)

Register 13.19. SYSTEM_CLOCK_GATE_REG (0x0054)

(reserved)																														SYSTEM_CLK_EN		
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

SYSTEM_CLK_EN 置 1 使能系统时钟。(R/W)

Register 13.20. SYSTEM_DATE_REG (0x0FFC)

(reserved)				SYSTEM_DATE																												
31	28			27																											0	
0	0	0	0	0x2007150																												Reset

SYSTEM_DATE 版本控制寄存器。(R/W)

本小节的所有地址均为相对于 apb 控制寄存器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 13.21. APB_CTRL_CLKGATE_FORCE_ON_REG (0x00A4)

(reserved)																								APB_CTRL_SRAM_CLKGATE_FORCE_ON				APB_CTRL_ROM_CLKGATE_FORCE_ON							
31																								6		5		2		1		0			
0 0																								0xf				3				Reset			

APB_CTRL_ROM_CLKGATE_FORCE_ON 置 1 配置 ROM 内存的时钟门控始终打开；置 0 则配置 ROM 内存的时钟门控在被访问时自动打开，没有访问时自动关闭。(R/W)

APB_CTRL_SRAM_CLKGATE_FORCE_ON 置 1 配置 SRAM 内存的时钟门控始终打开；置 0 则配置 SRAM 内存的时钟门控在被访问时自动打开，没有访问时自动关闭。(R/W)

Register 13.22. APB_CTRL_MEM_POWER_DOWN_REG (0x00A8)

(reserved)																								APB_CTRL_SRAM_POWER_DOWN		APB_CTRL_ROM_POWER_DOWN	
31																			6	5			2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

APB_CTRL_ROM_POWER_DOWN 控制 Internal ROM 进入 Retention 状态。(R/W)

APB_CTRL_SRAM_POWER_DOWN 控制 Internal SRAM 进入 Retention 状态。(R/W)

Register 13.23. APB_CTRL_MEM_POWER_UP_REG (0x00AC)

(reserved)																								APB_CTRL_SRAM_POWER_UP		APB_CTRL_ROM_POWER_UP	
31																			6	5			2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0xf	3	Reset

APB_CTRL_ROM_POWER_UP 控制 Internal ROM 在芯片进入 Light-sleep 时不会进入 Retention 状态。(R/W)

APB_CTRL_SRAM_POWER_UP 控制 Internal SRAM 在芯片进入 Light-sleep 时不会进入 Retention 状态。(R/W)

14 辅助调试 (Debug Assist)

14.1 概述

辅助调试模块提供一套调试功能，可用于软件开发时进行调试定位问题所在。

14.2 主要特性

- **读写监测**：监测 CPU 总线是否在限定的地址范围内进行读写操作，若发生读写操作则触发中断。
- **栈指针 (SP) 监测**：监测栈指针是否超出限定的范围，若超出范围则产生中断。
- **程序计数器 (PC) 记录**：记录 PC，可以获得上一次 CPU 复位时的 PC 值。
- **总线访问记录**：记录总线访问信息，当 CPU 或者 DMA 写了某个特殊值时，会记录此次写行为的地址和 PC 值，并将这些信息记录到 SRAM 中。

14.3 功能描述

14.3.1 区域读写监测

为确认 CPU 是否在某段地址范围（即区域）进行过读写行为，辅助调试模块能够监测 CPU 的数据总线和外设总线，当总线在监测地址范围进行读写操作时会触发中断。数据总线可同时监测两段地址范围，假设命名为数据总线区域 0 和数据总线区域 1，外设总线也可同时监测两段地址范围，假设命名为外设总线区域 0 和外设总线区域 1。区域 0 和区域 1 根据需要进行配置。

14.3.2 栈指针监测

为防止栈溢出或者错误的压栈弹栈，辅助调试模块能够监测栈指针，当栈指针超过限定的上下边界时会记录 PC 指针并产生中断，边界由软件进行配置。

14.3.3 PC 记录

在某些时候软件开发者希望知道上次 CPU 复位时的 PC 指针。比如，在程序卡死只能复位时，开发者可能希望读取复位时的 PC 指针以便知道程序在哪里卡死，然后进行调试。辅助调试模块可以记录 CPU 复位时的 PC，方便开发者进行调试。

14.3.4 CPU/DMA 总线访问记录

辅助调试模块能够实时记录 CPU 数据总线和 DMA 总线的写行为，当总线在某个范围内发生写操作或者写某个特定的值时，此次操作的总线类型、PC 和地址等信息会被记录下来，并按照一定的格式存储到 SRAM 中。

14.4 工作流程

14.4.1 区域监测和栈监测配置

区域监测可监测数据总线和外设总线的读写行为，每个总线可同时监测两个区域。详细监测模式如下。

- 数据总线读写监测
 - 监测数据总线在区域 0 是否进行读操作

- 监测数据总线在区域 0 是否进行写操作
- 监测数据总线在区域 1 是否进行读操作
- 监测数据总线在区域 1 是否进行写操作
- 外设总线读写监测
 - 监测外设总线在区域 0 是否进行读操作
 - 监测外设总线在区域 0 是否进行写操作
 - 监测外设总线在区域 1 是否进行读操作
 - 监测外设总线在区域 1 是否进行写操作
- 栈指针监测
 - 监测栈指针是否越过上限
 - 监测栈指针是否越过下限

区域监测和栈监测的配置流程如下：

1. 配置监测区域和栈指针

- 数据总线区域 0 由 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MIN_REG` 和 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MAX_REG` 进行配置。
- 数据总线区域 1 由 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MIN_REG` 和 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MAX_REG` 进行配置。
- 外设总线区域 0 由 `ASSIST_DEBUG_CORE_0_AREA_PIF_0_MIN_REG` 和 `ASSIST_DEBUG_CORE_0_AREA_PIF_0_MAX_REG` 进行配置。
- 外设总线区域 1 由 `ASSIST_DEBUG_CORE_0_AREA_PIF_1_MIN_REG` 和 `ASSIST_DEBUG_CORE_0_AREA_PIF_1_MAX_REG` 进行配置。
- 栈指针范围由 `ASSIST_DEBUG_CORE_0_SP_MIN_REG` 和 `ASSIST_DEBUG_CORE_0_SP_MAX_REG` 进行配置。

2. 配置中断

- 配置 `ASSIST_DEBUG_CORE_0_INTR_ENA_REG` 用于使能不同模式的中断。
- 配置 `ASSIST_DEBUG_CORE_0_INTR_RAW_REG` 用于查询不同模式的中断状态。
- 配置 `ASSIST_DEBUG_CORE_0_INTR_CLR_REG` 用于清除不同模式的中断。

3. 配置 `ASSIST_DEBUG_CORE_0_MONTR_ENA_REG` 使能不同的监测模式，可同时使能。

比如，若想监测数据总线地址 A 到地址 B 范围内是否进行过写操作，可通过数据总线区域 0 或者区域 1 进行监测，以区域 0 为例：

1. 配置 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MIN_REG` 为 A。
2. 配置 `ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MAX_REG` 为 B。
3. 配置 `ASSIST_DEBUG_CORE_0_INTR_ENA_REG` bit[1] 使能数据总线区域 0 写监测中断。
4. 配置 `ASSIST_DEBUG_CORE_0_MONTR_ENA_REG` bit[1] 使能数据总线区域 0 写监测。
5. 配置中断矩阵将 `ASSIST_DEBUG_INT` 映射到 CPU 中断上（参考章节 8 中断矩阵 (*INTMATRIX*)）。

6. 发生中断后

- 读取 `ASSIST_DEBUG_CORE_0_INTR_RAW_REG` 确认是什么操作触发的中断。
- 如果是区域监测引发的中断，读取 `ASSIST_DEBUG_CORE_0_AREA_PC` 可获取触发中断时刻的 PC 值，读取 `ASSIST_DEBUG_CORE_0_AREA_SP` 可获取触发中断时刻的 SP 值。
- 如果是栈监测引发的中断，读取 `ASSIST_DEBUG_CORE_0_SP_PC` 可获取触发中断时刻的 PC 值。
- 配置 `ASSIST_DEBUG_CORE_0_INTR_CLR_REG` 不同位可清除不同的中断。

14.4.2 PC 记录配置

CPU 输出一个 PC 信号给辅助调试模块，当 `ASSIST_DEBUG_CORE_0_RCD_PDEBUGEN` 配置为 1 时，该信号才有效，否则一直为 0。同时当 `ASSIST_DEBUG_CORE_0_RCD_RECORDEN` 配置为 1 时，`ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG` 会去采样 CPU PC 信号，否则保持原值。

寄存器 `ASSIST_DEBUG_CORE_0_RCD_EN_REG`、`ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG` 描述见 14.18、14.19。

当 CPU 发生复位时，`ASSIST_DEBUG_CORE_0_RCD_EN_REG` 会被复位，但是 `ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG` 不会被复位，因此后者会一直保持复位时刻的 PC 值。

14.4.3 CPU/DMA 总线访问记录配置

总线访问记录的配置流程如下：

1. 配置监测地址范围

- 配置 `ASSIST_DEBUG_LOG_MIN_REG` 和 `ASSIST_DEBUG_LOG_MAX_REG` 确定地址范围。

2. 配置监测模式 (`ASSIST_DEBUG_LOG_MODE`)

- 写监测（监测总线是否发生写操作）
- 字 (word) 监测（监测总线是否写某个特定 word）
- 半字 (halfword) 监测（监测总线是否写某个特定 halfword）
- 字节 (byte) 监测（监测总线是否写某个特定 byte）

3. 配置需要监测的特殊值

- 监测模式为 word 监测时，`ASSIST_DEBUG_LOG_DATA_0_REG` 即为监测的特定 word。
- 监测模式为 halfword 监测时，`ASSIST_DEBUG_LOG_DATA_0_REG` 的 [15:0] 即为监测的特定 halfword。
- 监测模式为 byte 监测时，`ASSIST_DEBUG_LOG_DATA_0_REG` 的 [7:0] 即为监测的特定 byte。
- `ASSIST_DEBUG_LOG_DATA_MASK_REG` 用于屏蔽 `ASSIST_DEBUG_LOG_DATA_0_REG` 的相应 byte。当某一 byte 被屏蔽，表示该 byte 可为任意的值，比如 word 监测，`ASSIST_DEBUG_LOG_DATA_0_REG` 配置为 0x01020304，`ASSIST_DEBUG_LOG_DATA_MASK_REG` 配置为 0x1，因此只要总线写 0x010203XX 都会被记录。

4. 配置记录信息的存储地址范围

- `ASSIST_DEBUG_LOG_MEM_START_REG` 和 `ASSIST_DEBUG_LOG_MEM_END_REG` 为存储地址范围配置寄存器。记录信息的存储地址范围为 0x3FCC_0000 ~ 0x3FCD_FFFF。

- 配置辅助调试模块对 internal SRAM 的使用权限，只有开启了辅助调试模块对 SRAM 的使用权限才能访问 SRAM。关于如何配置，详情见章节 2 权限控制 (PMS) [to be added later]。

5. 配置写内存的存储模式：loop 模式和非 loop 模式

- loop 模式下，会在配置地址范围内进行循环写，当写到结束地址时回到起始地址开始写，覆盖之前记录的信息。
例如，地址范围为 0~4，有 1~10 十次写操作，那么第 5 次写操作写到地址 4 后，第 6 次写操作会回到地址 0 开始写，第 6~10 写操作会覆盖之前的 0~4 写操作的数据。
- 非 loop 模式下，当写到结束地址后，会一直停留在结束地址进行写，不会覆盖最开始的记录信息。
例如，地址范围为 0~4，有 1~10 十次写操作，那么第 5 次写操作写到地址 4 后，第 6~10 写操作会一直在地址 4 上写，并且地址 4 最后会保留第 10 次写操作的数据。

6. 配置总线使能

- 配置 ASSIST_DEBUG_LOG_ENA 使能 CPU 或 DMA 总线访问记录，可同时使能。

当总线访问记录结束后需要从内存中读取记录数据并进行解码。记录数据里只有两种包格式，CPU 数据包和 DMA 数据包，对应 CPU 数据总线记录信息和 DMA 总线访问记录信息，其包格式如表 14-1、14-2 所示：

表 14-1. CPU 包格式

Bit[49:29]	Bit[28:2]	Bit[1:0]
addr_offset	pc_offset	format

表 14-2. DMA 包格式

Bit[24:6]	Bit[5:2]	Bit[1:0]
addr_offset	dma_source	format

从包格式可以看出，CPU 数据包共 50 位，DMA 数据包共 25 位。下面介绍各个域的含义：

- format 表示此次数据包的类型，1 表示 CPU 数据包，3 表示 DMA 数据包，其余值保留。
- pc_offset 记录了 CPU 的 PC 指针的偏移量，实际 PC = pc_offset + 0x4000_0000。
- addr_offset 记录了此次写操作的地址偏移，实际地址 = addr_offset + ASSIST_DEBUG_LOG_MIN_REG。
- dma_source 记录了哪一个外设发起的 DMA 访问，具体见 14-3。

表 14-3. DMA 访问来源

值	来源
1	SPI2
2	reserved
3	reserved
4	AES
5	SHA
6	ADC_DAC
7	I2S0
8	reserved
9	LCD_CAM

值	来源
10	reserved
11	UHCI0
12	reserved
13	LC
14	reserved
15	reserved

数据包缓存到内部 buffer 中，当 buffer 中数据满 125 位后扩展为 128 位数据写到 internal SRAM 中，写入内存的数据格式如表 14-4 所示。

表 14-4. 写内存数据格式

Bit[127:3]	Bit[2:0]
有效数据包	START_FLAG

由于 CPU 数据包共 50 位，DMA 数据包共 25 位，因此每次产生的记录信息最少 25 位，最多 75 位。当内部 buffer 满 125 位后，将数据写进内存，因此存在有的数据包被分成两部分，前部分被写进内存，后部分留在 buffer 中等待下次写入的情况，留在 buffer 中的数据称为残留数据。START_FLAG 表示本次 125 位数据中残留数据的位数（位数 = START_FLAG * 25），也可以理解为本次第一个完整数据包的起始位 (bit)。比如当前已经发生了 4 次 DMA 写操作，此时 buffer 里有 4 个 DMA 数据包共 100 位数据，此时再发生一次 CPU 写操作，会产生 50 位记录数据，此时 buffer 会将前面 100 位数据和 CPU 数据包里的前 25 位写入 SRAM，后 25 位留在 buffer 中等待下次写入，那么下次写入时数据格式中的 START_FLAG 会指明上次残留有 25 位在本次数据中。

当采用 loop 模式时，若在配置的存储地址范围内循环了若干次，残留数据会干扰解析，因此确定第一个有效数据包的位置时必须过滤残留数据。使用 START_FLAG 和 [ASSIST_DEBUG_LOG_MEM_CURRENT_ADDR_REG](#) 确定了数据包的起始位置之后，数据都是连续的，可忽略 START_FLAG 标志信息。

注意，由于内部存在 buffer，当 buffer 不满时不会写入内存，在解析时，应确保所有的数据都已写入内存，通过关闭总线访问记录能确保 buffer 中的数据全部写入内存中。当总线使能 [ASSIST_DEBUG_LOG_ENA](#) 全部配置为 0 时，若 buffer 有数据未写入内存中会以高位补零的形式以 128 位写到内存中。

数据包解析流程如下：

- [ASSIST_DEBUG_LOG_MEM_FULL_FLAG](#) 确定数据是否溢出配置范围，如果未溢出，则以 [ASSIST_DEBUG_LOG_MEM_START_REG](#) 作为第一个数据包的起始地址，若溢出并且开启了 loop 模式，则以 [ASSIST_DEBUG_LOG_MEM_CURRENT_ADDR_REG](#) 作为第一个数据包的起始地址。
- 从起始地址处开始读取数据并解析，每次读取 128 位。
- 通过 START_FLAG 来确认第一个数据包的起始位 (bit)，起始位为 START_FLAG * 25 + 3。

注意 START_FLAG 只用来找到起始位，找到第一个数据包之后后续数据中需过滤掉该信息。

数据解析完成之后需要通过配置 [ASSIST_DEBUG_CLR_LOG_MEM_FULL_FLAG](#) 清除 [ASSIST_DEBUG_LOG_MEM_FULL_FLAG](#) 标志位。

14.5 寄存器列表

本小节的所有地址均为相对于辅助调试基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

名称	描述	地址	访问
监测配置寄存器			
ASSIST_DEBUG_CORE_0_MONTR_ENA_REG	监测使能配置寄存器	0x0000	读/写
ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MIN_REG	数据总线区域 0 监测地址配置寄存器	0x0010	读/写
ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MAX_REG	数据总线区域 0 监测地址配置寄存器	0x0014	读/写
ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MIN_REG	数据总线区域 1 监测地址配置寄存器	0x0018	读/写
ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MAX_REG	数据总线区域 1 监测地址配置寄存器	0x001C	读/写
ASSIST_DEBUG_CORE_0_AREA_PIF_0_MIN_REG	外设总线区域 0 监测地址配置寄存器	0x0020	读/写
ASSIST_DEBUG_CORE_0_AREA_PIF_0_MAX_REG	外设总线区域 0 监测地址配置寄存器	0x0024	读/写
ASSIST_DEBUG_CORE_0_AREA_PIF_1_MIN_REG	外设总线区域 1 监测地址配置寄存器	0x0028	读/写
ASSIST_DEBUG_CORE_0_AREA_PIF_1_MAX_REG	外设总线区域 1 监测地址配置寄存器	0x002C	读/写
ASSIST_DEBUG_CORE_0_AREA_PC_REG	区域监测 CPU PC 状态寄存器	0x0030	只读
ASSIST_DEBUG_CORE_0_AREA_SP_REG	区域监测 CPU SP 状态寄存器	0x0034	只读
ASSIST_DEBUG_CORE_0_SP_MIN_REG	栈监测边界配置寄存器	0x0038	读/写
ASSIST_DEBUG_CORE_0_SP_MAX_REG	栈监测边界配置寄存器	0x003C	读/写
ASSIST_DEBUG_CORE_0_SP_PC_REG	栈监测 CPU PC 状态寄存器	0x0040	只读
中断配置寄存器			
ASSIST_DEBUG_CORE_0_INTR_RAW_REG	中断状态寄存器	0x0004	只读
ASSIST_DEBUG_CORE_0_INTR_ENA_REG	中断使能配置寄存	0x0008	读/写
ASSIST_DEBUG_CORE_0_INTR_CLR_REG	清除中断配置寄存器	0x000C	读/写
PC 记录配置寄存器			
ASSIST_DEBUG_CORE_0_RCD_EN_REG	PC 记录使能配置寄存器	0x0044	读/写
PC 记录状态寄存器			
ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG	记录信息寄存器	0x0048	只读
ASSIST_DEBUG_CORE_0_RCD_PDEBUGSP_REG	记录信息寄存器	0x004C	只读
总线记录配置寄存器			

名称	描述	地址	访问
ASSIST_DEBUG_LOG_SETTING_REG	总线访问记录配置寄存器	0x0070	读/写
ASSIST_DEBUG_LOG_DATA_0_REG	总线访问监测数据配置寄存器	0x0074	读/写
ASSIST_DEBUG_LOG_DATA_MASK_REG	总线访问监测数据屏蔽配置寄存器	0x0078	读/写
ASSIST_DEBUG_LOG_MIN_REG	总线访问监测范围配置寄存器	0x007C	读/写
ASSIST_DEBUG_LOG_MAX_REG	总线访问监测范围配置寄存器	0x0080	读/写
ASSIST_DEBUG_LOG_MEM_START_REG	记录数据写入内存的起始地址	0x0084	读/写
ASSIST_DEBUG_LOG_MEM_END_REG	记录数据写入内存的结束地址	0x0088	读/写
ASSIST_DEBUG_LOG_MEM_CURRENT_ADDR_REG	指示下一次写内存的写地址	0x008C	只读
ASSIST_DEBUG_LOG_MEM_FULL_FLAG_REG	记录溢出状态寄存器	0x0090	不定
CPU 状态寄存器			
ASSIST_DEBUG_CORE_0_LASTPC_BEFORE_EXCEPTION_REG	CPU 异常前的最后一条指令的 PC	0x0094	只读
ASSIST_DEBUG_CORE_0_DEBUG_MODE_REG	CPU 处于 Debug Mode 的标志寄存器	0x0098	只读
版本寄存器			
ASSIST_DEBUG_DATE_REG	版本寄存器	0x01FC	读/写

14.6 寄存器

本小节的所有地址均为相对于辅助调试基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 14.1. ASSIST_DEBUG_CORE_0_MONTR_ENA_REG (0x0000)

(reserved)																								ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_ENA ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_ENA ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_ENA ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_ENA ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_ENA ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_ENA ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_ENA ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_ENA ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_ENA ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_ENA										
31																							10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_ENA 数据总线在区域 0 内读操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_ENA 数据总线在区域 0 内写操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_ENA 数据总线在区域 1 内读操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_ENA 数据总线在区域 1 内写操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_ENA 外设总线在区域 0 内读操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_ENA 外设总线在区域 0 内写操作的监测使能。(读/写)

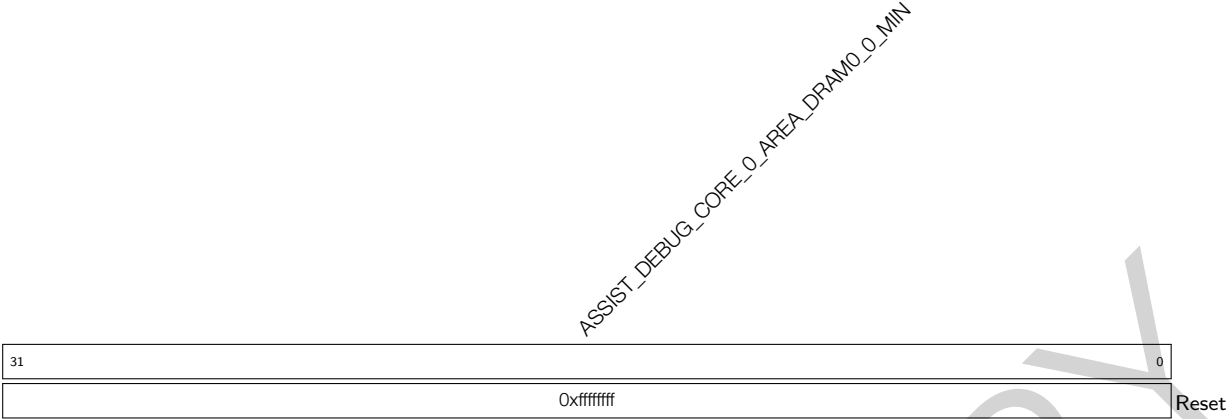
ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_ENA 外设总线在区域 1 内读操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_ENA 外设总线在区域 1 内写操作的监测使能。(读/写)

ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_ENA 栈指针小于栈监测区域的下边界的监测使能。(读/写)

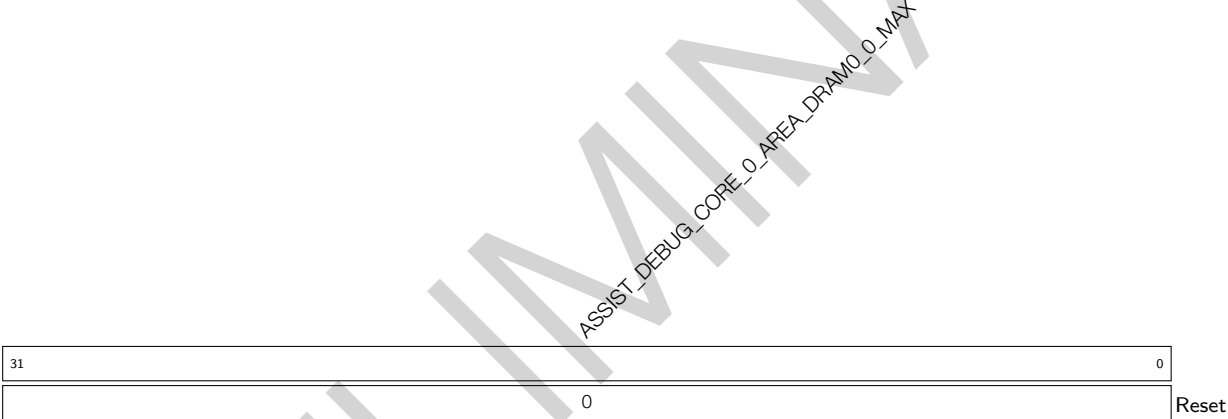
ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_ENA 栈指针大于栈监测区域的上边界的监测使能。(读/写)

Register 14.2. ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MIN_REG (0x0010)



ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MIN 数据总线区域 0 的下边界。(读/写)

Register 14.3. ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MAX_REG (0x0014)



ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_MAX 数据总线区域 0 的上边界。(读/写)

Register 14.4. ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MIN_REG (0x0018)

31	0
0xffffffff	
Reset	

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MIN 数据总线区域 1 的下边界。(读/写)

Register 14.5. ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MAX_REG (0x001C)

31	0
0	
Reset	

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_MAX 数据总线区域 1 的上边界。(读/写)

Register 14.6. ASSIST_DEBUG_CORE_0_AREA_PIF_0_MIN_REG (0x0020)

31	0
0xffffffff	
Reset	

ASSIST_DEBUG_CORE_0_AREA_PIF_0_MIN 外设总线区域 0 的下边界。(读/写)

Register 14.7. ASSIST_DEBUG_CORE_0_AREA_PIF_0_MAX_REG (0x0024)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_MAX	
31	0
0	
Reset	

ASSIST_DEBUG_CORE_0_AREA_PIF_0_MAX 外设总线区域 0 的上边界。(读/写)

Register 14.8. ASSIST_DEBUG_CORE_0_AREA_PIF_1_MIN_REG (0x0028)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_MIN	
31	0
0xffffffff	
Reset	

ASSIST_DEBUG_CORE_0_AREA_PIF_1_MIN 外设总线区域 1 的下边界。(读/写)

Register 14.9. ASSIST_DEBUG_CORE_0_AREA_PIF_1_MAX_REG (0x002C)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_MAX	
31	0
0	
Reset	

ASSIST_DEBUG_CORE_0_AREA_PIF_1_MAX 外设总线区域 1 的上边界。(读/写)

Register 14.10. ASSIST_DEBUG_CORE_0_AREA_PC_REG (0x0030)

ASSIST_DEBUG_CORE_0_AREA_PC																															
0																															
Reset																															

ASSIST_DEBUG_CORE_0_AREA_PC 区域监测下，记录触发中断时刻的 PC 值。(只读)

Register 14.11. ASSIST_DEBUG_CORE_0_AREA_SP_REG (0x0034)

ASSIST_DEBUG_CORE_0_AREA_SP															
31															0
0															
Reset															

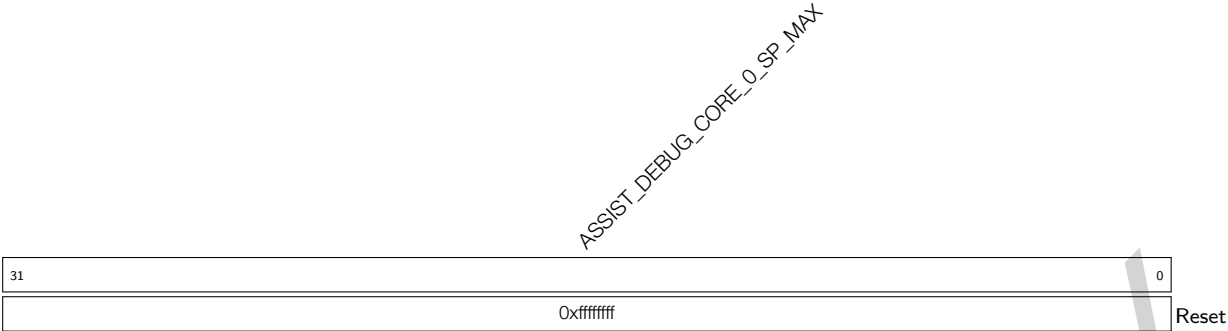
ASSIST_DEBUG_CORE_0_AREA_SP 区域监测下，记录触发中断时刻的 SP 值。(只读)

Register 14.12. ASSIST_DEBUG_CORE_0_SP_MIN_REG (0x0038)

ASSIST_DEBUG_CORE_0_SP_MIN																															
31																															0
0																															
Reset																															

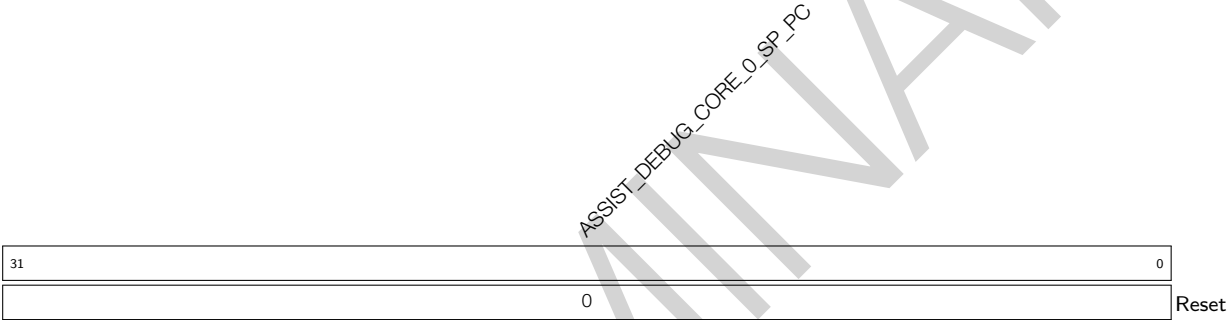
ASSIST_DEBUG_CORE_0_SP_MIN 栈指针的下边界。(读/写)

Register 14.13. ASSIST_DEBUG_CORE_0_SP_MAX_REG (0x003C)



ASSIST_DEBUG_CORE_0_SP_MAX 栈指针的上边界。(读/写)

Register 14.14. ASSIST_DEBUG_CORE_0_SP_PC_REG (0x0040)



ASSIST_DEBUG_CORE_0_SP_PC 记录栈指针监测的 PC 值。(只读)

Register 14.15. ASSIST_DEBUG_CORE_0_INTR_RAW_REG (0x0004)

(reserved)										ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_RAW ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_RAW ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_RAW ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_RAW ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_RAW ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_RAW ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_RAW ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_RAW ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_RAW ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_RAW										
31										10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset																				

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_RAW 数据总线在区域 0 内读操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_RAW 数据总线在区域 0 内写操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_RAW 数据总线在区域 1 内读操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_RAW 数据总线在区域 1 内写操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_RAW 外设总线在区域 0 内读操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_RAW 外设总线在区域 0 内写操作的中断状态。(只读)

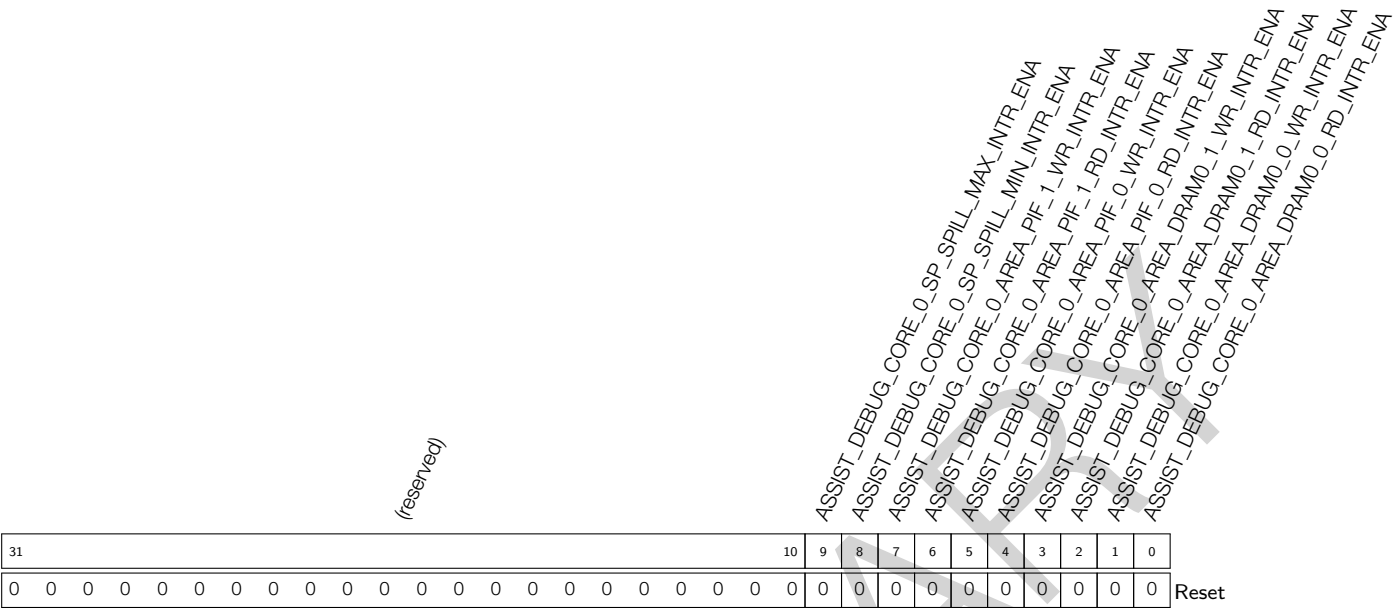
ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_RAW 外设总线在区域 1 内读操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_RAW 外设总线在区域 1 内写操作的中断状态。(只读)

ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_RAW 栈指针小于栈监测区域的下边界的中断状态。(只读)

ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_RAW 栈指针大于栈监测区域的上边界的中断状态。(只读)

Register 14.16. ASSIST_DEBUG_CORE_0_INTR_ENA_REG (0x0008)



ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_INTR_ENA 数据总线在区域 0 内读操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_INTR_ENA 数据总线在区域 0 内写操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_INTR_ENA 数据总线在区域 1 内读操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_INTR_ENA 数据总线在区域 1 内写操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_INTR_ENA 外设总线在区域 0 内读操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_INTR_ENA 外设总线在区域 0 内写操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_INTR_ENA 外设总线在区域 1 内读操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_INTR_ENA 外设总线在区域 1 内写操作的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_INTR_ENA 栈指针小于栈监测区域的下边界的中断使能。(读/写)

ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_INTR_ENA 栈指针大于栈监测区域的上边界的中断使能。(读/写)

Register 14.17. ASSIST_DEBUG_CORE_0_INTR_CLR_REG (0x000C)

(reserved)																						ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_CLR (ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_CLR (ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_CLR (ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_CLR (ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_CLR (ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_CLR (ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_CLR (ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_CLR (ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_CLR (ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_CLR										
31																				10	9	8	7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset										

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_RD_CLR 清除数据总线在区域 0 内读操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_0_WR_CLR 清除数据总线在区域 0 内写操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_RD_CLR 清除数据总线在区域 1 内读操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_DRAM0_1_WR_CLR 清除数据总线在区域 1 内写操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_RD_CLR 清除外设总线在区域 0 内读操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_0_WR_CLR 清除外设总线在区域 0 内写操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_RD_CLR 清除外设总线在区域 1 内读操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_AREA_PIF_1_WR_CLR 清除外设总线在区域 1 内写操作的中断。(读/写)

ASSIST_DEBUG_CORE_0_SP_SPILL_MIN_CLR 清除栈指针小于栈监测区域的下边界的中断。(读/写)

ASSIST_DEBUG_CORE_0_SP_SPILL_MAX_CLR 清除栈指针大于栈监测区域的上边界的中断。(读/写)

Register 14.18. ASSIST_DEBUG_CORE_0_RCD_EN_REG (0x0044)

(reserved)																												2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																												Reset		

ASSIST_DEBUG_CORE_0_RCD_RECORDEN PC 记录使能，配置为 1 时，ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG 开始实时记录 PC。(读/写)

ASSIST_DEBUG_CORE_0_RCD_PDEBUGEN CPU 调试使能，配置为 1 时，CPU 才会输出 PC。(读/写)

Register 14.19. ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC_REG (0x0048)

ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC																															
31																															0
0x000000																															
Reset																															

ASSIST_DEBUG_CORE_0_RCD_PDEBUGPC 记录复位时刻的 PC 值。(只读)

Register 14.20. ASSIST_DEBUG_CORE_0_RCD_PDEBUGSP_REG (0x004C)

ASSIST_DEBUG_CORE_0_RCD_PDEBUGSP																															
31																															0
0x000000																															
																															Reset

ASSIST_DEBUG_CORE_0_RCD_PDEBUGSP 记录 SP。(只读)

Register 14.21. ASSIST_DEBUG_LOG_SETTING_REG (0x0070)

(reserved)																								ASSIST_DEBUG_LOG_MEM_LOOP_ENABLE				ASSIST_DEBUG_LOG_MODE				ASSIST_DEBUG_LOG_ENA			
31																								8	7	6	3			2	0			Reset	
0 0																								1	0			0							

ASSIST_DEBUG_LOG_ENA 使能 CPU 或 DMA 总线访问记录。bit[0]: CPU 总线访问记录; bit[1]: 保留; bit[2]: DMA 总线访问记录。(读/写)

ASSIST_DEBUG_LOG_MODE 配置监测模式。bit[0]: 写监测; bit[1]: word 监测; bit[2]: halfword 监测; bit[3]: byte 监测。(读/写)

ASSIST_DEBUG_LOG_MEM_LOOP_ENABLE 配置写内存的存储模式。1: loop 模式; 0: 非 loop 模式。(读/写)

Register 14.22. ASSIST_DEBUG_LOG_DATA_0_REG (0x0074)

ASSIST_DEBUG_LOG_DATA_0																															
31																															0
0																															
Reset																															

ASSIST_DEBUG_LOG_DATA_0 配置总线访问监测的特殊值。(读/写)

Register 14.23. ASSIST_DEBUG_LOG_DATA_MASK_REG (0x0078)

(reserved)																ASSIST_DEBUG_LOG_DATA_SIZE													
31																16	15												0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
																													Reset

ASSIST_DEBUG_LOG_DATA_SIZE 用于屏蔽 [ASSIST_DEBUG_LOG_DATA_0_REG](#) 的相应 byte。(读/写)

Register 14.24. ASSIST_DEBUG_LOG_MIN_REG (0x007C)

ASSIST_DEBUG_LOG_MIN																															
31																															0
0																															
Reset																															

ASSIST_DEBUG_LOG_MIN 配置监测地址的下边界。(读/写)

Register 14.25. ASSIST_DEBUG_LOG_MAX_REG (0x0080)

ASSIST_DEBUG_LOG_MAX	
31	0
0	
Reset	

ASSIST_DEBUG_LOG_MAX 配置监测地址的上边界。(读/写)

Register 14.26. ASSIST_DEBUG_LOG_MEM_START_REG (0x0084)

ASSIST_DEBUG_LOG_MEM_START	
31	0
0	
Reset	

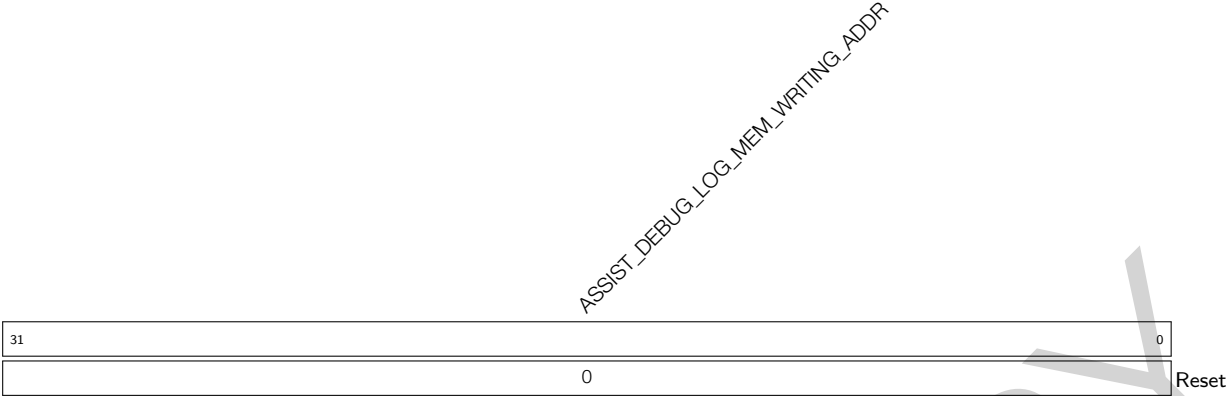
ASSIST_DEBUG_LOG_MEM_START 配置记录数据写入内存的起始地址。(读/写)

Register 14.27. ASSIST_DEBUG_LOG_MEM_END_REG (0x0088)

ASSIST_DEBUG_LOG_MEM_END	
31	0
0	
Reset	

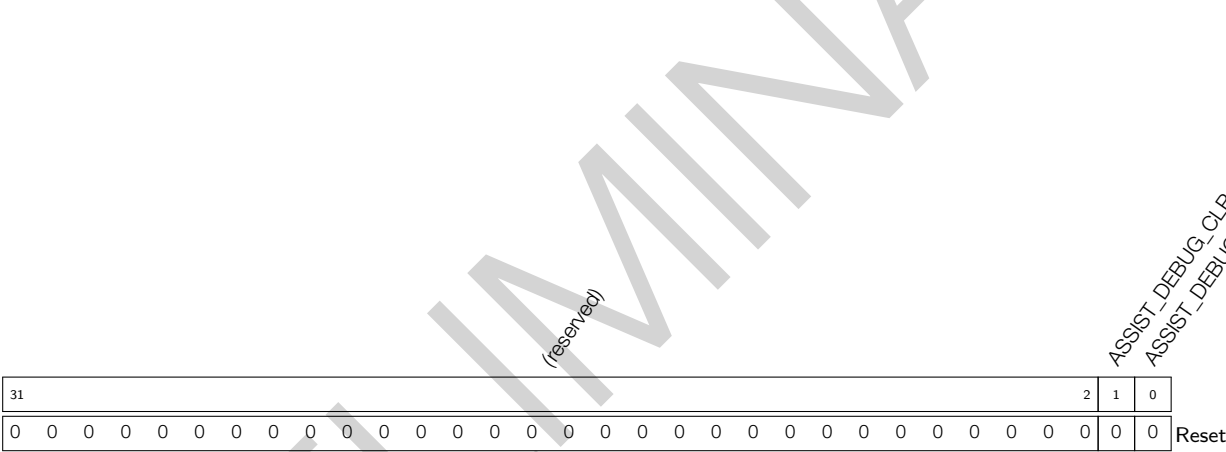
ASSIST_DEBUG_LOG_MEM_END 配置记录数据写入内存的结束地址。(读/写)

Register 14.28. ASSIST_DEBUG_LOG_MEM_CURRENT_ADDR_REG (0x008C)



ASSIST_DEBUG_LOG_MEM_CURRENT_ADDR 指示下一次写内存的写地址。(只读)

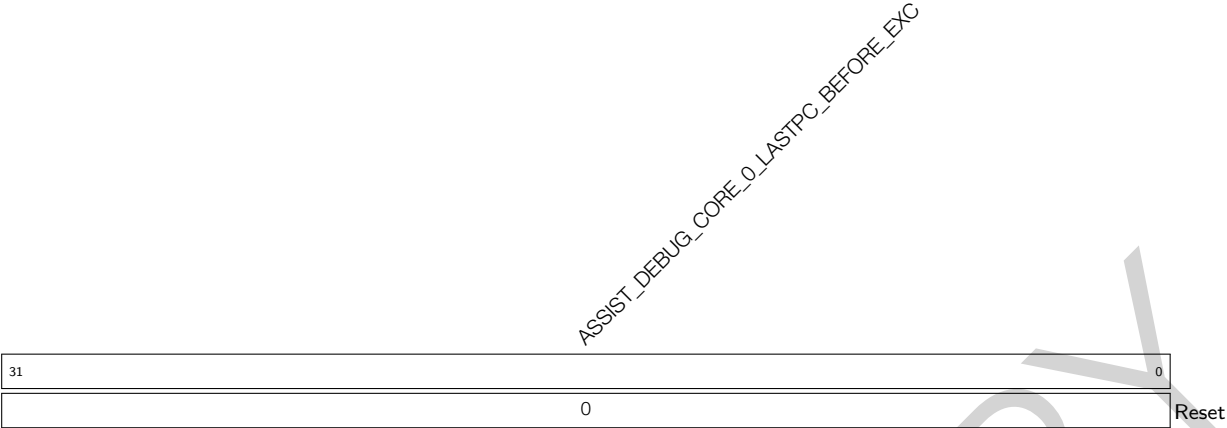
Register 14.29. ASSIST_DEBUG_LOG_MEM_FULL_FLAG_REG (0x0090)



ASSIST_DEBUG_LOG_MEM_FULL_FLAG 为 1 则表示数据溢出存储地址范围。(只读)

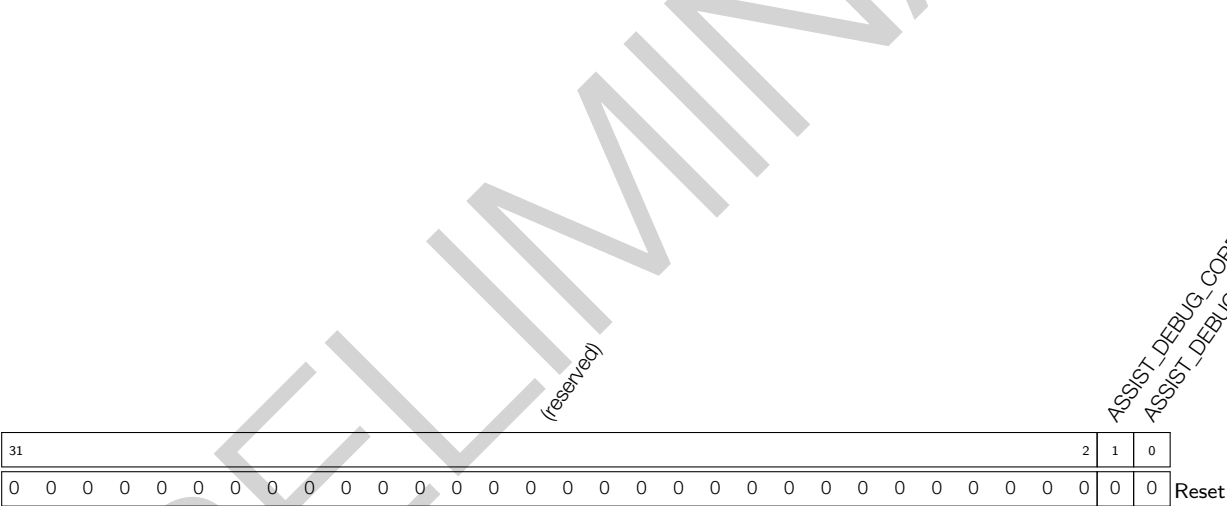
ASSIST_DEBUG_CLR_LOG_MEM_FULL_FLAG 置 1 清除 ASSIST_DEBUG_LOG_MEM_FULL_FLAG 标志位。默认值为 0。(读/写)

Register 14.30. ASSIST_DEBUG_CORE_0_LASTPC_BEFORE_EXCEPTION_REG (0x0094)



ASSIST_DEBUG_CORE_0_LASTPC_BEFORE_EXC 记录 CPU 异常前的最后一条指令的 PC。(只读)

Register 14.31. ASSIST_DEBUG_CORE_0_DEBUG_MODE_REG (0x0098)



ASSIST_DEBUG_CORE_0_DEBUG_MODE 指示 RISC-V CPU 是否处于调试模式。1：处于调试模式；0：未处于调试模式。(只读)

ASSIST_DEBUG_CORE_0_DEBUG_MODULE_ACTIVE 指示 RISC-V CPU 调试模块的工作状态。1：处于工作状态；其它：未处于工作状态（只读）

Register 14.32. ASSIST_DEBUG_DATE_REG (0x01FC)

(reserved)				ASSIST_DEBUG_DATE																							
31	28	27																									0
0	0	0	0																								Reset

ASSIST_DEBUG_DATE 版本控制寄存器。(读/写)

15 SHA 加速器 (SHA)

15.1 概述

ESP32-C3 内置 SHA（安全哈希算法）硬件加速器可完成 SHA 运算，具有 [Typical SHA](#) 和 [DMA-SHA](#) 两种工作模式。整体而言，相比基于纯软件的 SHA 运算，SHA 硬件加速器能够极大地提高运算速度。

15.2 主要特性

ESP32-C3 的 SHA 硬件加速器：

- 支持 [FIPS PUB 180-4 规范](#) 中的以下运算标准
 - SHA-1 运算
 - SHA-224 运算
 - SHA-256 运算
- 提供两种工作模式
 - Typical SHA 工作模式
 - DMA-SHA 工作模式
- 允许插入 (interleaved) 功能（仅限 Typical SHA 工作模式）
- 允许中断功能（仅限 DMA-SHA 工作模式）

15.3 工作模式简介

ESP32-C3 内置的 SHA 加速器支持两种工作模式。

- [Typical SHA 工作模式](#)：所有数据读写统一通过 CPU 访问完成。
- [DMA-SHA 工作模式](#)：所有读数据通过硬件上的 DMA 完成。具体来说，用户可配置 DMA 控制器，由 DMA 控制器提供 SHA 运算过程中所需的数据信息。因此，可以释放 CPU 执行其他任务。

用户可通过配置 [SHA_START_REG](#) 或 [SHA_DMA_START_REG](#) 选择 SHA 加速器的工作模式，先配置的工作模式生效，具体请见表 15-1。

表 15-1. 工作模式选择

工作模式	选择方式
Typical SHA	SHA_START_REG 置 1
DMA-SHA	SHA_DMA_START_REG 置 1

用户可通过配置 [SHA_MODE_REG](#) 寄存器选择 SHA 加速器的运算标准，具体请见表 15-2。

表 15-2. 运算标准选择

哈希运算标准	SHA_MODE_REG 的配置
SHA-1	0
SHA-224	1
SHA-256	2

注意：

ESP32-C3 的 [数字签名 \(DS\)](#) 和 HMAC 模块也会调用 SHA 加速器。此时，用户无法正常访问 SHA 加速器。

15.4 功能描述

SHA 加速器可以提取信息摘要 (message digest)，其主要工作流程分为两步：[信息预处理](#)和[哈希运算](#)。

15.4.1 信息预处理

信息预处理分为三个主要步骤：[附加填充比特](#)、[信息解析](#)和[设置初始哈希值](#)。

15.4.1.1 附加填充比特

SHA 加速器仅能处理长度为 512 位及其整倍数的信息。因此，在将信息送至 SHA 加速器进行运算前，应先通过软件操作将信息填充为符合要求的长度。

假设待处理信息 M 的长度为 m 位，则填充步骤见下：

1. 首先，在待处理信息后填充 1 个“1”；
2. 随后，再填充 k 个“0”。其中， k 为满足 $m + 1 + k \equiv 448 \bmod 512$ 的最小非负数解；
3. 最后，在末尾填充一个 64 位的信息块。该信息块的内容为用二进制表示的待处理信息的长度，即 m 的值。

更多详情，请参考 [FIPS PUB 180-4 规范](#) 中的“5.1 Padding the Message”章节。

15.4.1.2 信息解析

在完成信息填充后，我们还需将待处理信息（及其填充）解析为 N 个 512 位的信息块，即 $M^{(1)}$ 、 $M^{(2)}$ 、...、 $M^{(N)}$ 。一个 512 位信息块包括 16 个 32 位的字 (word)，则第 i 个信息块的第一个 32 位字表示为 $M_0^{(i)}$ ，第二个 32 位字表示为 $M_1^{(i)}$ ，...，第 16 个 32 位字表示为 $M_{15}^{(i)}$ 。

SHA 加速器在工作时，每次处理的信息块数据均将按照如下规则写入相应的寄存器中：将 $M_0^{(i)}$ 存放在 [SHA_M_0_REG](#) 中， $M_1^{(i)}$ 存放在 [SHA_M_1_REG](#)，...， $M_{15}^{(i)}$ 存放在 [SHA_M_15_REG](#) 中。

说明：

有关“信息块”及相关概念的描述，请参考 [FIPS PUB 180-4 规范](#) 中“2.1 Glossary of Terms and Acronyms”章节。

15.4.1.3 哈希初始值 (Initial Hash Value)

在进行哈希运算前，首先必须设置哈希初始值 $H^{(0)}$ ，其中 SHA-1、SHA-224 和 SHA-256 运算的哈希初始值为常量 C，且已经固定在硬件中，无需额外配置。

15.4.2 哈希运算流程

在完成信息预处理后，ESP32-C3 SHA 加速器将正式开始哈希运算，最终根据不同运算标准得到不同长度的信息摘要。正如上文所述，ESP32-C3 SHA 加速器支持 [Typical SHA](#) 和 [DMA-SHA](#) 两种工作模式，下面将对这两种工作模式的具体流程进行介绍。

15.4.2.1 Typical SHA 模式下的运算流程

通常情况下，ESP32-C3 的 SHA 会处理完当前信息的所有信息块并生成该信息的信息摘要，之后再开始计算新的信息摘要。

不过，ESP32-C3 SHA 加速器还支持“interleaved”运算（Typical SHA 和 DMA-SHA 工作模式均支持），即在完成当前信息的所有运算前，允许插入其他运算任务。

- 在 [Typical SHA](#) 工作模式下，用户每计算完一个信息块后均可插入新的运算；
- 而在 [DMA-SHA](#) 工作模式下，用户必须等待本次 DMA 运算全部完成才可以插入新的运算。

具体来说，用户可以将存储在 [SHA_H_n_REG](#) 寄存器中的信息摘要暂时保存到其他地方，然后让 SHA 加速器来完成其他优先级更高的运算任务。当插入的运算结束后，用户再将之前暂存的信息摘要重新写入 [SHA_H_n_REG](#) 中，并继续完成之前中断的计算。

Typical SHA 的具体运算流程

1. 选择运算标准。
 - 配置 [SHA_MODE_REG](#) 寄存器，设置运算标准。具体配置，请参考表 15-2。
2. 处理当前信息块。
 - 将当前信息块写入 [SHA_M_n_REG](#) 寄存器。
3. 启动 SHA 加速器¹。
 - 如果为首次运算，则对 [SHA_START_REG](#) 寄存器置 1，启动 SHA 加速器的运算。此时，SHA 加速器按照步骤 1 中选定的运算标准，使用硬件中固定的哈希初始值进行运算；
 - 如果非首次运算²，则对 [SHA_CONTINUE_REG](#) 寄存器置 1，启动 SHA 加速器的运算。此时，SHA 加速器使用 [SHA_H_n_REG](#) 寄存器中的值作为哈希初始值进行运算。
4. 查询当前信息块的处理进度。
 - 轮询寄存器 [SHA_BUSY_REG](#) 一直到读回的值为 0，代表 SHA 硬件加速器已完成对当前信息块的计算，进入“空闲”状态³。
5. 选择是否有后续的待处理信息块。
 - 如果存在后续待处理信息块，则跳回执行步骤 2。
 - 否则，继续执行。
6. 获取信息摘要：

- 从寄存器堆 [SHA_H_n_REG](#) 取出信息摘要。

说明:

1. 这里，在 SHA 加速器进行硬件运算时，如果存在后续待处理信息块，软件还可以同时将后续信息块写入 [SHA_M_n_REG](#) 寄存器，以节省时间。
2. 比如重新启动 SHA 加速器完成之前暂停任务的情况。
3. 这里，你可以选择是否需要插入其他任务。如需插入，请前往 [插入任务工作流程](#) 具体查看。

如上文所述，ESP32-C3 SHA 加速器支持在 **Typical SHA 模式** 下“插入”任务。

具体工作流程如下。

1. 保存插入前任务的以下数据，准备将 SHA 加速器的使用权移交给插入的任务。
 - 读取并保存寄存器 [SHA_MODE_REG](#) 中的运算标准类型。
 - 读取并保存寄存器堆 [SHA_H_n_REG](#) 中的信息摘要。
2. 执行插入的任务。具体按照插入运行类型的不同，请见 [Typical SHA](#) 或 [DMA-SHA 工作流程](#)。
3. 恢复插入前任务的以下数据，准备将 SHA 加速器的使用权交还给插入前的任务。
 - 将获得使用权前保存的运算标准类型重新写入寄存器 [SHA_MODE_REG](#);
 - 将获得使用权前保存的信息摘要写入寄存器堆 [SHA_H_n_REG](#)。
4. 将之前任务的下一个待处理信息块写入 [SHA_M_n_REG](#) 寄存器，并对 [SHA_CONTINUE_REG](#) 寄存器置 1，重新启动 SHA 加速器，完成之前暂停的任务。

15.4.2.2 DMA-SHA 模式下的运算流程

ESP32-C3 SHA 加速器在 DMA-SHA 工作模式下不支持在完成每个“信息块”运算后插入新的运算，即用户必须在每次 DMA 运算（可能包括 1 个或多个信息块）全部结束后才能插入新的运算。这种情况下，用户如有插入运算需求，可将较大信息块进行拆分，并进行多次 DMA 运算。每次 DMA 运算之间允许插入其他运算标准的计算任务。

单次 DMA 运算最多可以处理 63 个数据块。

与 Typical SHA 不同，SHA 在 DMA-SHA 工作模式下，运算过程中的数据搬运过程均由硬件完成。具体配置可见章节 [2 通用 DMA 控制器 \(GDMA\)](#)。

DMA-SHA 的具体工作流程

1. 选择运算标准。
 - 配置 [SHA_MODE_REG](#) 寄存器，设置运算标准。具体配置，请参考表 [15-2](#)。
2. 选择是否启用中断。请将 [SHA_INT_ENA_REG](#) 寄存器配置为 1 以启动中断。
3. 配置块个数。
 - 将待加密数据的总块数 M 写入 [SHA_DMA_BLOCK_NUM_REG](#) 寄存器。
4. 开始 DMA-SHA 运算。
 - 如果当前 DMA-SHA 运算为接着另一次 DMA-SHA 的运算，需要提前将另一次计算得到的信息摘要写入寄存器堆 [SHA_H_n_REG](#) 中，随后将 1 写入寄存器 [SHA_DMA_CONTINUE_REG](#);

- 否则，只需要将 1 写入寄存器 [SHA_DMA_START_REG](#)。
5. 等待 DMA-SHA 运算结束。判断 DMA-SHA 运算结束有以下两种方法：
- 轮询寄存器 [SHA_BUSY_REG](#) 结果为 0。
 - 等待中断信号产生。此时，应及时通过软件将 [SHA_INT_CLEAR_REG](#) 寄存器置为 1 以清除中断。
6. 获取信息摘要
- 从寄存器堆 [SHA_H_n_REG](#) 取出信息摘要。

15.4.3 信息摘要存储

哈希运算完成之后，计算得到的信息摘要被 SHA 加速器更新至对应的 [SHA_H_n_REG](#) (n : 0 ~ 7) 寄存器中。不同运算标准得到的信息摘要长度也不同，详情见表 15-3：

表 15-3. 不同运算标准信息摘要的寄存器占用情况

哈希运算标准	信息摘要长度（位）	寄存器占用情况 ¹
SHA-1	160	SHA_H_0_REG ~ SHA_H_4_REG
SHA-224	224	SHA_H_0_REG ~ SHA_H_6_REG
SHA-256	256	SHA_H_0_REG ~ SHA_H_7_REG

¹ 信息摘要从左至右存放，第一个 word 存放在寄存器 [SHA_H_0_REG](#) 中，第二个 word 存放在寄存器 [SHA_H_1_REG](#) 中，以此类推。

15.4.4 中断

SHA 加速器在 DMA-SHA 工作模式下允许中断发生。用户可通过将 [SHA_INT_ENA_REG](#) 寄存器配置为 1 开启中断。如开启中断功能，SHA 加速器在完成运算时，中断发生。注意，该中断必须由软件将 [SHA_INT_CLEAR_REG](#) 寄存器置为 1 进行清除。由于 SHA 加速器在 Typical SHA 工作模式下的时间开销较小，因此不支持中断功能。

15.5 寄存器列表

本小节的所有地址均为相对于 SHA 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	权限
控制与状态寄存器			
SHA_CONTINUE_REG	继续 SHA 运算（仅用于 Typical SHA 模式）	0x0014	WO
SHA_BUSY_REG	指示 SHA 加速器是否处于“忙碌”状态	0x0018	RO
SHA_DMA_START_REG	启动 SHA 加速器的 DMA-SHA 模式	0x001C	WO
SHA_START_REG	启动 SHA 加速器的 Typical SHA 模式	0x0010	WO
SHA_DMA_CONTINUE_REG	继续 SHA 运算（仅用于 DMA-SHA 模式）	0x0020	WO
SHA_INT_CLEAR_REG	DMA-SHA 中断清除寄存器	0x0024	WO
SHA_INT_ENA_REG	DMA-SHA 中断使能寄存器	0x0028	R/W
版本寄存器			
SHA_DATE_REG	版本控制寄存器	0x002C	R/W
配置寄存器			
SHA_MODE_REG	配置 SHA 加速器的运算标准	0x0000	R/W
数据寄存器			
SHA_DMA_BLOCK_NUM_REG	信息块个数寄存器（仅用于 DMA-SHA 工作模式）	0x000C	R/W
SHA_H_0_REG	哈希值	0x0040	R/W
SHA_H_1_REG	哈希值	0x0044	R/W
SHA_H_2_REG	哈希值	0x0048	R/W
SHA_H_3_REG	哈希值	0x004C	R/W
SHA_H_4_REG	哈希值	0x0050	R/W
SHA_H_5_REG	哈希值	0x0054	R/W
SHA_H_6_REG	哈希值	0x0058	R/W
SHA_H_7_REG	哈希值	0x005C	R/W
SHA_M_1_REG	输入信息	0x0084	R/W
SHA_M_2_REG	输入信息	0x0088	R/W
SHA_M_3_REG	输入信息	0x008C	R/W
SHA_M_4_REG	输入信息	0x0090	R/W
SHA_M_5_REG	输入信息	0x0094	R/W
SHA_M_6_REG	输入信息	0x0098	R/W
SHA_M_7_REG	输入信息	0x009C	R/W
SHA_M_8_REG	输入信息	0x00A0	R/W
SHA_M_9_REG	输入信息	0x00A4	R/W
SHA_M_10_REG	输入信息	0x00A8	R/W
SHA_M_11_REG	输入信息	0x00AC	R/W
SHA_M_12_REG	输入信息	0x00B0	R/W
SHA_M_13_REG	输入信息	0x00B4	R/W
SHA_M_14_REG	输入信息	0x00B8	R/W
SHA_M_15_REG	输入信息	0x00BC	R/W

15.6 寄存器

本小节的所有地址均为相对于 SHA 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 15.1. SHA_START_REG (0x0010)

(reserved)																															SHA_START																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SHA_START 置 1 启动 SHA 加速器的 Typical SHA 模式。(RO)

Register 15.2. SHA_CONTINUE_REG (0x0014)

(reserved)																															SHA_CONTINUE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

SHA_CONTINUE 置 1 继续 SHA 加速器的 Typical SHA 运算。(WO)

Register 15.3. SHA_BUSY_REG (0x0018)

(reserved)																															SHA_BUSY_STATE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SHA_BUSY_STATE 指示 SHA 是否处于“忙碌”状态。(RO) 1'h0: 空闲 1'h1: 忙碌

Register 15.4. SHA_DMA_START_REG (0x001C)

(reserved)																															SHA_DMA_START				
31																															1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

SHA_DMA_START 置 1 启动 SHA 加速器的 DMA-SHA 模式。(WO)

Register 15.5. SHA_DMA_CONTINUE_REG (0x0020)

(reserved)																														SHA_DMA_CONTINUE	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

SHA_DMA_CONTINUE 置 1 继续 SHA 加速器的 DMA-SHA 运算。(WO)

Register 15.6. SHA_INT_CLEAR_REG (0x0024)

(reserved)																														SHA_CLEAR_INTERRUPT	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

SHA_CLEAR_INTERRUPT 清除 DMA-SHA 中断。(WO)

Register 15.7. SHA_INT_ENA_REG (0x0028)

(reserved)																														SHA_INTERRUPT_ENA	
31																														1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

SHA_INTERRUPT_ENA 使能 DMA-SHA 中断。(R/W)

Register 15.8. SHA_DATE_REG (0x002C)

(reserved)		SHA_DATE																													
31	30	29																												0	
0	0		0x20190402																												

Reset

SHA_DATE 版本控制寄存器。(R/W)

278

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

278

ESP32-C3 TRM (预发布 v0.4)

278

278

ESP32-C3 TRM (预发布 v0.4)

278

278

ESP32-C3 TRM (预发布 v0.4)

278

16 AES 加速器 (AES)

16.1 概述

ESP32-C3 内置 AES（高级加密标准）硬件加速器可使用 AES 算法，完成数据的加解密运算，具有 [Typical AES](#) 和 [DMA-AES](#) 两种工作模式。整体而言，相比基于纯软件的 AES 运算，AES 硬件加速器能够极大地提高运算速度。

16.2 主要特性

ESP32-C3 支持以下特性：

- Typical AES 工作模式
 - AES-128/AES-256 加解密运算
- DMA-AES 工作模式
 - AES-128/AES-256 加解密运算
 - 块（加密）模式
 - * ECB (Electronic Codebook)
 - * CBC (Cipher Block Chaining)
 - * OFB (Output Feedback)
 - * CTR (Counter)
 - * CFB8 (8-bit Cipher Feedback)
 - * CFB128 (128-bit Cipher Feedback)
 - 中断发生

16.3 工作模式简介

ESP32-C3 内置的 AES 加速器支持 Typical AES 和 DMA-AES 两种工作模式。

- Typical AES 工作模式：
 - 支持使用 128 位或 256 位密钥进行加密与解密运算，即 [NIST FIPS 197](#) 标准中的 AES-128 和 AES-256 加解密运算。

这种情况下，明文/密文的读/写操作统一通过 CPU 访问完成。

- DMA-AES 工作模式：
 - 支持使用 128 位或 256 位密钥进行加密与解密运算，即 [NIST FIPS 197](#) 标准中的 AES-128 和 AES-256 加解密运算；
 - 还支持 [NIST SP 800-38A](#) 标准中的 ECB/CBC/OFB/CTR/CFB8/CFB128 等块加密模式运算。

在这种情况下，明文/密文的传输通过硬件上的 DMA 完成，计算完成时会有中断发生。

用户可通过配置 [AES_DMA_ENABLE_REG](#) 选择 AES 加速器的工作模式，具体参考表 16-1。

表 16-1. 工作模式

AES_DMA_ENABLE_REG	工作模式
0	Typical AES
1	DMA-AES

用户可通过配置 [AES_MODE_REG](#) 寄存器选择密钥长度和解密方向，具体可参考表 16-2。

表 16-2. 密钥长度和解密方向

AES_MODE_REG[2:0]	密钥长度和解密方向
0	AES-128 加密
1	保留
2	AES-256 加密
3	保留
4	AES-128 解密
5	保留
6	AES-256 解密
7	保留

有关 Typical AES 和 DMA-AES 两种工作模式的具体介绍，请见下方 16.4 章节和 16.5 章节。

注意：

ESP32-C3 的[数字签名 \(DS\)](#)模块也会调用 AES 加速器。此时，用户无法正常访问 AES 加速器。

16.4 Typical AES 工作模式

在 Typical AES 工作模式下，AES 加速器的状态值可查看寄存器 [AES_STATE_REG](#)，具体见表 16-3 所示：

表 16-3. 状态返回值

返回值	描述	状态说明
0	IDLE	加速器空闲或计算完成
1	WORK	加速器忙于计算

16.4.1 密钥、明文、密文

寄存器 [AES_KEY_n_REG](#) 用于存放密钥，由 8 个 32 位寄存器组成。

- 如果为 AES-128 加解密运算，则 128 位密钥在寄存器 [AES_KEY_0_REG](#) ~ [AES_KEY_3_REG](#) 中。
- 如果为 AES-256 加解密运算，则 256 位密钥在寄存器 [AES_KEY_0_REG](#) ~ [AES_KEY_7_REG](#) 中。

寄存器 [AES_TEXT_IN_m_REG](#) 和 [AES_TEXT_OUT_m_REG](#) 用于存放明文和密文，各由 4 个 32 位寄存器组成。

- 如果为 AES-128/256 加密运算，则运算开始之前用明文初始化寄存器 [AES_TEXT_IN_m_REG](#)。运算完成之后，AES 加速器将把密文更新入寄存器 [AES_TEXT_OUT_m_REG](#)。

- 如果为 AES-128/256 解密运算，则运算开始之前用密文初始化寄存器 [AES_TEXT_IN_m_REG](#)。运算完成之后，AES 加速器将把明文更新入寄存器 [AES_TEXT_OUT_m_REG](#)。

16.4.2 字节序

文本字节序

在 Typical AES 工作模式下，AES 加速器可以使用密钥对 128 位的 block 进行加解密。在操作寄存器 [AES_TEXT_IN_m_REG](#) 和 [AES_TEXT_OUT_m_REG](#) 中的数据时，用户应遵循表 16-4 中定义的文本字节序。

表 16-4. Typical AES 文本字节序

明文/密文				
State ¹	c ²			
	0	1	2	3
r	0	AES_TEXT_x_0_REG[7:0]	AES_TEXT_x_1_REG[7:0]	AES_TEXT_x_2_REG[7:0]
	1	AES_TEXT_x_0_REG[15:8]	AES_TEXT_x_1_REG[15:8]	AES_TEXT_x_2_REG[15:8]
	2	AES_TEXT_x_0_REG[23:16]	AES_TEXT_x_1_REG[23:16]	AES_TEXT_x_2_REG[23:16]
	3	AES_TEXT_x_0_REG[31:24]	AES_TEXT_x_1_REG[31:24]	AES_TEXT_x_2_REG[31:24]

¹ 有关“State（以及 c 和 r）”的详细定义，请参考 [NIST FIPS 197](#) 中“3.4 The State”章节。

² 其中，x = IN 或 OUT。

密钥字节序

在 Typical AES 工作模式下，在向寄存器 [AES_KEY_n_REG](#) 中填入数据时，用户应遵循表 16-5 和表 16-6 中定义的文本字节序。

表 16-5. AES-128 密钥字节序

Bit ¹	w[0]	w[1]	w[2]	w[3] ²
[31:24]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]
[23:16]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]
[15:8]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]
[7:0]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]

¹ Bit 列代表 w[0] ~ w[3] 每个 word 中的各个字节。

² w[0] ~ w[3] 符合标准 [NIST FIPS 197](#) 中“5.2 Key Expansion”章节中对“the first Nk words of the expanded key”的描述。

表 16-6. AES-256 密钥字节序

Bit ¹	w[0]	w[1]	w[2]	w[3]	w[4]	w[5]	w[6]	w[7] ²
[31:24]	AES_KEY_0_REG[7:0]	AES_KEY_1_REG[7:0]	AES_KEY_2_REG[7:0]	AES_KEY_3_REG[7:0]	AES_KEY_4_REG[7:0]	AES_KEY_5_REG[7:0]	AES_KEY_6_REG[7:0]	AES_KEY_7_REG[7:0]
[23:16]	AES_KEY_0_REG[15:8]	AES_KEY_1_REG[15:8]	AES_KEY_2_REG[15:8]	AES_KEY_3_REG[15:8]	AES_KEY_4_REG[15:8]	AES_KEY_5_REG[15:8]	AES_KEY_6_REG[15:8]	AES_KEY_7_REG[15:8]
[15:8]	AES_KEY_0_REG[23:16]	AES_KEY_1_REG[23:16]	AES_KEY_2_REG[23:16]	AES_KEY_3_REG[23:16]	AES_KEY_4_REG[23:16]	AES_KEY_5_REG[23:16]	AES_KEY_6_REG[23:16]	AES_KEY_7_REG[23:16]
[7:0]	AES_KEY_0_REG[31:24]	AES_KEY_1_REG[31:24]	AES_KEY_2_REG[31:24]	AES_KEY_3_REG[31:24]	AES_KEY_4_REG[31:24]	AES_KEY_5_REG[31:24]	AES_KEY_6_REG[31:24]	AES_KEY_7_REG[31:24]

¹ Bit 列代表 w[0] ~ w[7] 每个 word 中的各个字节。
² w[0] ~ w[7] 符合标准 [NIST FIPS 197](#) 中 “5.2 Key Expansion” 章节中对 “the first Nk words of the expanded key” 的描述。

16.4.3 Typical AES 工作模式的流程

单次运算

1. 对寄存器 `AES_DMA_ENABLE_REG` 写入 0。
2. 初始化寄存器 `AES_MODE_REG`、`AES_KEY_n_REG`、`AES_TEXT_IN_m_REG`。
3. 启动运算。对寄存器 `AES_TRIGGER_REG` 写入 1。
4. 等待运算完成。轮询寄存器 `AES_STATE_REG`，直到读到 0。
5. 从寄存器 `AES_TEXT_OUT_m_REG` 读取结果。

连续运算

在连续运算过程中，每次运算完成之后，只有寄存器 `AES_TEXT_IN_m_REG` 和 `AES_TEXT_OUT_m_REG` (m : 0-3) 会被 AES 加速器更新，而 `AES_DMA_ENABLE_REG`、`AES_MODE_REG`、`AES_KEY_n_REG` 等寄存器中的内容不会变化。所以进行连续运算时可以简化初始化操作。

1. 第一次运算之前对寄存器 `AES_DMA_ENABLE_REG` 写入 0。
2. 第一次运算之前初始化寄存器 `AES_MODE_REG` 和 `AES_KEY_n_REG`。
3. 更新寄存器 `AES_TEXT_IN_m_REG`。
4. 启动运算。对寄存器 `AES_TRIGGER_REG` 写入 1。
5. 等待运算完成。轮询寄存器 `AES_STATE_REG`，直到读到 0。
6. 从寄存器 `AES_TEXT_OUT_m_REG` 读取结果。返回步骤 3，进行下一轮运算。

16.5 DMA-AES 工作模式

在 DMA-AES 工作模式下，AES 加速器可支持 ECB/CBC/OFB/CTR/CFB8/CFB128 等 6 种块模式运算。用户可以通过配置 [AES_BLOCK_MODE_REG](#) 寄存器选择具体运算类型，具体可参考表 16-7。

表 16-7. 块模式选择

AES_BLOCK_MODE_REG [2:0]	块模式
0	ECB (Electronic Code Book)
1	CBC (Cipher Block Chaining)
2	OFB (Output FeedBack)
3	CTR (Counter)
4	CFB8 (8-bit Cipher FeedBack)
5	CFB128 (128-bit Cipher FeedBack)
6	保留
7	保留

AES 加速器的状态值可查看寄存器 [AES_STATE_REG](#)，具体见表 16-8 所示：

表 16-8. 状态返回值

返回值	描述	状态说明
0	IDLE	加速器空闲
1	WORK	加速器忙于计算
2	DONE	加速器计算完成

AES 加速器在 DMA-AES 工作模式下允许中断发生，软件清零。中断功能默认关闭，用户可通过将 [AES_INT_ENA_REG](#) 寄存器配置为 1 开启中断。如开启中断功能，AES 加速器在完成计算时，中断发生。

16.5.1 密钥、明文、密文

块运算模式

在块运算模式下，AES 加速器的源数据来自 DMA，结果数据也将被写入 DMA。

- 如果为加密运算，则 DMA 从 memory 中读取明文数据流并将其传给 AES。AES 计算出密文后将密文写入 DMA。DMA 再将密文写入 memory。
- 如果为解密运算，则 DMA 从 memory 中读取密文数据流并将其传给 AES。AES 计算出明文后将明文写入 DMA。DMA 再将明文写入 memory。

AES 加速器在进行块运算时，结果数据与源数据的大小保持一致。此时，DMA 的数据搬运过程和 AES 的计算过程有所交叠，因此总工作时间有所减少。

值得注意的是，AES 加速器在 DMA-AES 工作模式下要求源数据的大小必须是 128 位的整数倍，否则需要将原始明文封装为 128 位的整数倍，即在原比特串 (bit string) 尾部尽可能少的补 “0”，具体过程见表 16-9 所示。

表 16-9. TEXT-PADDING

Function : TEXT-PADDING()	
Input	: X , bit string.
Output	: $Y = \text{TEXT-PADDING}(X)$, whose length is the nearest integral multiples of 128 bits.
Steps Let us assume that X is a data-stream that can be split into n parts as following: $X = X_1 X_2 \cdots X_{n-1} X_n$ Here, the lengths of $X_1, X_2, \cdots, X_{n-1}$ all equal to 128 bits, and the length of X_n is t ($0 < t \leq 127$). If $t = 0$, then $\text{TEXT-PADDING}(X) = X;$ If $0 < t \leq 127$, define a 128-bit block, X_n^* , and let $X_n^* = X_n 0^{128-t}$, then $\text{TEXT-PADDING}(X) = X_1 X_2 \cdots X_{n-1} X_n^* = X 0^{128-t}$	

16.5.2 字节序

在 DMA-AES 工作模式下，源数据和结果数据的传输完全由 DMA 完成，因此不支持字节序的控制调节，但要求它们在 memory 中以一定的方式来存放，且要求数据量必须是 block 的整数倍。

举例说明，假设 DMA 需要搬运 2 个 block 大小的源数据：

- 十六进制：0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20

假设起始地址为 0x0280，则源数据在 memory 中的存放位置如表 16-10 所示。结果数据也遵从相同的存放规则，在此不多做介绍。

表 16-10. DMA AES 存储字节序

地址	字节	地址	字节	地址	字节	地址	字节
0x0280	0x01	0x0281	0x02	0x0282	0x03	0x0283	0x04
0x0284	0x05	0x0285	0x06	0x0286	0x07	0x0287	0x08
0x0288	0x09	0x0289	0x0A	0x028A	0x0B	0x028B	0x0C
0x028C	0x0D	0x028D	0x0E	0x028E	0x0F	0x028F	0x10
0x0290	0x11	0x0291	0x12	0x0292	0x13	0x0293	0x14
0x0294	0x15	0x0295	0x16	0x0296	0x17	0x0297	0x18
0x0298	0x19	0x0299	0x1A	0x029A	0x1B	0x029B	0x1C
0x029C	0x1D	0x029D	0x1E	0x029E	0x1F	0x029F	0x20

16.5.3 标准增量函数

AES 加速器在进行 CTR 块运算时，还可提供两种标准增量函数供用户选择：INC₃₂ 和 INC₁₂₈。用户可通过将寄存器 AES_INC_SEL_REG 置为 0 或 1 选择 INC₃₂ 或 INC₁₂₈ 标准增量函数。更多有关标准增量函数的内容，请见 [NIST SP 800-38A](#) 标准中的“B.1 The Standard Incrementing Function”章节。

16.5.4 块个数

寄存器 AES_BLOCK_NUM_REG 存放明文或密文的块个数 (Block Number)，其值等于 $\text{length}(\text{TEXT-PADDING}(P))/128$ ，也等于 $\text{length}(\text{TEXT-PADDING}(C))/128$ 。这里的 P 指明文 (plaintext)， C 指密文 (ciphertext)。该寄存器

仅在 DMA-AES 工作模式下有意义。

16.5.5 初始向量

存储器 `AES_IV_MEM` 的空间大小为 16 字节，仅在块运算模式下有效。对于 CBC/OFB/CFB8/CFB128 等操作，`AES_IV_MEM` 用于存放初始向量 (Initialization Vector, IV) 的值。对于 CTR 操作，`AES_IV_MEM` 存放初始计数器 (Initial Counter Block, ICB) 的值。

IV 和 ICB 都是 128-bit 长的比特串，从左向右被分割成 16 个字节 (Byte0, Byte1, Byte2, ..., Byte15)，构成一个字节序列，在 `AES_IV_MEM` 中存放时需要遵循表 16-10 中的字节序规则，即 Byte0 存放在 `AES_IV_MEM` 中的最低地址中，Byte15 存放在 `AES_IV_MEM` 中的最高地址中。

更多有关 IV 和 ICB 的信息，请参考 [NIST SP 800-38A](#) 标准。

16.5.6 DMA-AES 工作模式的流程

1. 选择一条 DMA 通道与 AES 加速器连接，配置 DMA 链表，而后启动 DMA。详情请见章节 2 通用 DMA 控制器 (GDMA)。
2. 配置 AES：
 - 对寄存器 `AES_DMA_ENABLE_REG` 写入 1。
 - 选择是否开启中断。根据需要设置寄存器 `AES_INT_ENA_REG` 的值。
 - 初始化 `AES_MODE_REG` 和 `AES_KEY_n_REG` 寄存器。
 - 配置 `AES_BLOCK_MODE_REG` 寄存器，选择具体块加密模式。详见表 16-7。
 - 初始化寄存器 `AES_BLOCK_NUM_REG`，请参照章节 16.5.4。
 - 初始化寄存器 `AES_INC_SEL_REG`（仅在 CTR 块模式下使用）。
 - 初始化存储器 `AES_IV_MEM`（在 ECB 块模式下不使用）。
3. 启动运算。对寄存器 `AES_TRIGGER_REG` 写入 1。
4. 等待运算完成。轮询寄存器 `AES_STATE_REG`，直到读到 2。如果开启了中断功能，也可以等待 `AES_INT` 中断产生。
5. 确认 DMA 完成从 AES 到内存的数据传输。此时，结果数据已经被 DMA 写入 memory，可以直接从中读取。详情请参考章节 2 通用 DMA 控制器 (GDMA)。
6. 如果开启了中断，当处理中断程序完成后，请及时对寄存器 `AES_INT_CLR_REG` 写 1 以清除中断。
7. 对寄存器 `AES_DMA_EXIT_REG` 写入 1 释放 AES 加速器。之后如果再读取寄存器 `AES_STATE_REG` 将读到 0。该步操作可以提前完成，但必须在步骤 4 之后。

16.6 存储器列表

本小节的所有地址均为相对于 AES 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	大小（比特）	起始地址	结束地址	访问权限
AES_IV_MEM	存储器 IV	16 字节	0x0050	0x005F	(R/W)

16.7 寄存器列表

本小节的所有地址均为相对于 AES 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
密钥寄存器			
AES_KEY_0_REG	AES 密钥资料寄存器 0	0x0000	R/W
AES_KEY_1_REG	AES 密钥资料寄存器 1	0x0004	R/W
AES_KEY_2_REG	AES 密钥资料寄存器 2	0x0008	R/W
AES_KEY_3_REG	AES 密钥资料寄存器 3	0x000C	R/W
AES_KEY_4_REG	AES 密钥资料寄存器 4	0x0010	R/W
AES_KEY_5_REG	AES 密钥资料寄存器 5	0x0014	R/W
AES_KEY_6_REG	AES 密钥资料寄存器 6	0x0018	R/W
AES_KEY_7_REG	AES 密钥资料寄存器 7	0x001C	R/W
TEXT_IN 寄存器			
AES_TEXT_IN_0_REG	源数据资料寄存器 0	0x0020	R/W
AES_TEXT_IN_1_REG	源数据资料寄存器 1	0x0024	R/W
AES_TEXT_IN_2_REG	源数据资料寄存器 2	0x0028	R/W
AES_TEXT_IN_3_REG	源数据资料寄存器 3	0x002C	R/W
TEXT_OUT 寄存器			
AES_TEXT_OUT_0_REG	结果数据资料寄存器 0	0x0030	RO
AES_TEXT_OUT_1_REG	结果数据资料寄存器 1	0x0034	RO
AES_TEXT_OUT_2_REG	结果数据资料寄存器 2	0x0038	RO
AES_TEXT_OUT_3_REG	结果数据资料寄存器 3	0x003C	RO
配置寄存器			
AES_MODE_REG	选择密钥长度和加解密方向	0x0040	R/W
AES_DMA_ENABLE_REG	选择 AES 加速器工作模式	0x0090	R/W
AES_BLOCK_MODE_REG	选择 DMA-AES 下的块运算模式	0x0094	R/W
AES_BLOCK_NUM_REG	块数量配置寄存器	0x0098	R/W
AES_INC_SEL_REG	标准增量函数选择寄存器	0x009C	R/W
控制 / 状态寄存器			
AES_TRIGGER_REG	开始运算寄存器	0x0048	WO
AES_STATE_REG	运算状态寄存器	0x004C	RO
AES_DMA_EXIT_REG	退出运算寄存器	0x00B8	WO
中断寄存器			
AES_INT_CLR_REG	DMA-AES 中断清除	0x00AC	WO
AES_INT_ENA_REG	DMA-AES 中断使能寄存器	0x00B0	R/W

16.8 寄存器

本小节的所有地址均为相对于 AES 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 16.1. AES_KEY_ *n* _REG (*n*: 0-7) (0x0000+4**n*)

31	0
0x00000000	
Reset	

AES_KEY_ *n* _REG (*n*: 0-7) AES 密钥资料寄存器。(R/W)

Register 16.2. AES_TEXT_IN_ *m* _REG (*m*: 0-3) (0x0020+4**m*)

31	0
0x00000000	
Reset	

AES_TEXT_IN_ *m* _REG (*m*: 0-3) Typical AES 文本输入寄存器。(R/W)

Register 16.3. AES_TEXT_OUT_ *m* _REG (*m*: 0-3) (0x0030+4**m*)

31	0
0x00000000	
Reset	

AES_TEXT_OUT_ *m* _REG (*m*: 0-3) Typical AES 文本输出寄存器。(RO)

Register 16.4. AES_MODE_REG (0x0040)

(reserved)		AES_MODE	
31	3	2	0
0x00000000		0	
		Reset	

AES_MODE 选择 AES 加速器的密钥长度和解密方向，详情请见表 16-2。(R/W)

Register 16.5. AES_DMA_ENABLE_REG (0x0090)

(reserved)															AES_DMA_ENABLE	
31														1	0	Reset
0x00000000															0	

AES_DMA_ENABLE 选择 AES 加速器的工作模式。0: Typical AES, 1: DMA-AES。详情请见表 16-1。
(R/W)

Register 16.6. AES_BLOCK_MODE_REG (0x0094)

(reserved)															AES_BLOCK_MODE		
31														3	2	0	Reset
0x00000000															0		

AES_BLOCK_MODE 选择 AES 加速器在 DMA-AES 工作模式下的块模式，详情请见表 16-7。(R/W)

Register 16.7. AES_BLOCK_NUM_REG (0x0098)

31																0	Reset
0x00000000																	

AES_BLOCK_NUM 在 DMA-AES 运算中待加解密的文本块数。详情请见章节 16.5.4。(R/W)

Register 16.8. AES_INC_SEL_REG (0x009C)

(reserved)															AES_INC_SEL	
31														1	0	Reset
0x00000000															0	

AES_INC_SEL 选择 CTR 块模式使用的标准增量函数。置 0 选择 INC₃₂ 标准增量函数，置 1 选择 INC₁₂₈ 标准增量函数。(R/W)

Register 16.9. AES_TRIGGER_REG (0x0048)

(reserved)															AES_TRIGGER	
31														1	0	Reset
0x00000000															x	

AES_TRIGGER 写入 1 使能 AES 运算。(WO)

Register 16.10. AES_STATE_REG (0x004C)

(reserved)															AES_STATE		
31														2	1	0	Reset
0x00000000															0x0		

AES_STATE AES 状态寄存器。详见表 16-3 (Typical AES 工作模式) 和表 16-8 (DMA-AES 工作模式)。(RO)

Register 16.11. AES_DMA_EXIT_REG (0x00B8)

(reserved)															AES_DMA_EXIT	
31														1	0	Reset
0x00000000															x	

AES_DMA_EXIT 在 DMA-AES 运算完成后, 在下一次配置 AES 任何寄存器之前, 写入 1 使 AES 回到空闲状态。(WO)

Register 16.12. AES_INT_CLR_REG (0x00AC)

(reserved)															AES_INT_CLR	
31														1	0	Reset
0x00000000															x	

AES_INT_CLR 写入 1 清除 AES 中断。(WO)

Register 16.13. AES_INT_ENA_REG (0x00B0)

(reserved)		AES_INT_ENA	
31	1	0	
0x00000000		0	Reset

AES_INT_ENA 写入 1 使能 AES 中断功能，写入 0 关闭 AES 中断功能。(R/W)

17 RSA 加速器 (RSA)

17.1 概述

RSA 加速器可为多种运用于“RSA 非对称式加密演算法”的高精度计算提供硬件支持，能够极大地降低此类运算的软件复杂度，且支持多种“运算子长度”，具有很高的运算效率。

17.2 主要特性

RSA 加速器支持以下功能：

- 大数模幂运算（支持两个加速选项）
- 大数模乘运算
- 大数乘法运算
- 多种运算子长度
- 中断功能

17.3 功能描述

RSA 加速器的激活仅需使能 `SYSTEM_PERIP_CLK_EN1_REG` 外围时钟的 `SYSTEM_CRYPT_RSA_CLK_EN` 位，并同时清零 `SYSTEM_RSA_PD_CTRL_REG` 寄存器中的 `SYSTEM_RSA_MEM_PD` 位。

不过，RSA 加速器激活后还须等待 [RSA 相关存储器](#) 初始化完成后才能开始工作。具体来说，寄存器 `RSA_CLEAN_REG` 读 0 时初始化开始，读 1 时初始化完成。因此，在复位后首次使用 RSA 加速器时，软件需要先查询寄存器 `RSA_CLEAN_REG` 的值是否为 1，以确保 RSA 加速器可正常工作。

此外，RSA 加速器支持中断功能，可对寄存器 `RSA_INTERRUPT_ENA_REG` 写 1 / 0 以开启 / 关闭中断。RSA 加速器的中断功能默认开启。

注意：

ESP32-C3 的 [数字签名 \(DS\)](#) 模块也会调用 RSA 加速器。此时，用户无法正常访问 RSA 加速器。

17.3.1 大数模幂运算

大数模幂运算的算法是 $Z = X^Y \bmod M$ ，它是基于 Montgomery Multiplication（蒙哥马利乘法）实现的。因此，对于大数模幂运算，除了需要运算子 X 、 Y 、 M 外，还需要额外两个运算子，即参数 \bar{r} 和 M' 。这两个参数需要通过软件提前运算得到。

RSA 加速器支持运算子长度为 $N = 32 \times x$ ($x \in \{1, 2, 3, \dots, 96\}$) 的大数模幂运算。 Z 、 X 、 Y 、 M 和 \bar{r} 的位宽为这 96 种中的任意一种，要求它们的位宽必须相同，而 M' 的位宽始终是 32。

设进制数

$$b = 2^{32}$$

则运算子可以由若干个 b 进制数来表示：

$$n = \frac{N}{32}$$

$$Z = (Z_{n-1}Z_{n-2} \cdots Z_0)_b$$

$$X = (X_{n-1}X_{n-2} \cdots X_0)_b$$

$$Y = (Y_{n-1}Y_{n-2} \cdots Y_0)_b$$

$$M = (M_{n-1}M_{n-2} \cdots M_0)_b$$

$$\bar{r} = (\bar{r}_{n-1}\bar{r}_{n-2} \cdots \bar{r}_0)_b$$

其中 $Z_{n-1} \cdots Z_0$ 、 $X_{n-1} \cdots X_0$ 、 $Y_{n-1} \cdots Y_0$ 、 $M_{n-1} \cdots M_0$ 、 $\bar{r}_{n-1} \cdots \bar{r}_0$ 分别表示一个 b 进制数，位宽皆为 32。且 Z_{n-1} 、 X_{n-1} 、 Y_{n-1} 、 M_{n-1} 、 \bar{r}_{n-1} 分别为 Z 、 X 、 Y 、 M 、 \bar{r} 最高位的 b 进制数，而 Z_0 、 X_0 、 Y_0 、 M_0 、 \bar{r}_0 分别为 Z 、 X 、 Y 、 M 、 \bar{r} 最低位的 b 进制数。

另设 $R = b^n$ ，则计算得参数 $\bar{r} = R^2 \bmod M$ 。

M' 可使用下方公式计算：

$$M^{-1} \times M + 1 = R \times R^{-1}$$

$$M' = M^{-1} \bmod b$$

注意，上方公式适用于使用扩展二进制 GCD 算法的运算。

大数模幂运算的软件流程为：

1. 对寄存器 [RSA_INTERRUPT_ENA_REG](#) 写 1 / 0 以开启 / 关闭中断。
2. 配置相关寄存器。
 - (a) 对寄存器 [RSA_MODE_REG](#) 写入 $(\frac{N}{32} - 1)$ 。
 - (b) 对寄存器 [RSA_M_PRIME_REG](#) 写入 M' 。
 - (c) 根据需要配置加速选项相关寄存器。请参照章节 17.3.4 获取详细信息。
3. 将 X_i 、 Y_i 、 M_i 、 $\bar{r}_i (i \in \{0, 1, \dots, n-1\})$ 分别写入存储器 [RSA_X_MEM](#)、[RSA_Y_MEM](#)、[RSA_M_MEM](#)、[RSA_Z_MEM](#)。每块存储器的容量都是 96 字 (word)。每块存储器的每一个字刚好存放一个 b 进制数。这些存储器都是低地址存放运算子的低位进制数，高地址存放运算子的高位进制数。
只需要根据运算子长度，将各个运算子中有效的数据写入存储器，没有使用到的存储器可以是任意值。
4. 对寄存器 [RSA_MODEXP_START_REG](#) 写入 1 启动计算。
5. 等待运算结束。轮询寄存器 [RSA_IDLE_REG](#) 直到读到 1，或者等待 RSA 中断产生。
6. 从存储器 [RSA_Z_MEM](#) 读出运算结果 $Z_i (i \in \{0, 1, \dots, n-1\})$ 。
7. 若中断功能已开启，对寄存器 [RSA_CLEAR_INTERRUPT_REG](#) 写入 1 以清除中断。

运算结束后，寄存器 [RSA_MODE_REG](#) 中存储的运算子长度信息以及存储器 [RSA_Y_MEM](#) 中的 Y_i 、存储器 [RSA_M_MEM](#) 中的 M_i 、寄存器 [RSA_M_PRIME_REG](#) 中的 M' 都不会变化。但是，存储器 [RSA_X_MEM](#) 中的 X_i 与存储器 [RSA_Z_MEM](#) 中的 \bar{r}_i 都已经被覆盖。所以当需要连续运算时，只需要更新被覆盖的存储器即可。

17.3.2 大数模乘运算

大数模乘运算 $Z = X \times Y \bmod M$ 也是基于 Montgomery Multiplication 实现的。因此，与大数模幂运算类似，也需要预先通过软件计算额外的两个运算子 \bar{r} 和 M' 。

RSA 加速器也支持 96 种运算子长度的大数模乘运算。

大数模乘运算的软件流程为：

1. 对寄存器 `RSA_INTERRUPT_ENA_REG` 写 1 / 0 以开启 / 关闭中断。
2. 配置相关寄存器。
 - (a) 对寄存器 `RSA_MODE_REG` 写入 $(\frac{N}{32} - 1)$ 。
 - (b) 对寄存器 `RSA_M_PRIME_REG` 写入 M' 。
3. 将 X_i 、 Y_i 、 M_i 、 \bar{r}_i ($i \in \{0, 1, \dots, n-1\}$) 分别写入存储器 `RSA_X_MEM`、`RSA_Y_MEM`、`RSA_M_MEM`、`RSA_Z_MEM`。每块存储器的容量都是 96 字 (word)。
 每块存储器的每一个字刚好存放一个 b 进制数。这些存储器都是低地址存放运算子的低位进制数，高地址存放运算子的高位进制数。
 只需要根据运算子长度，将各个运算子中有效的数据写入存储器，没有使用到的存储器可以是任意值。
4. 对寄存器 `RSA_MODMULT_START_REG` 写入 1。
5. 等待运算结束。轮询寄存器 `RSA_IDLE_REG` 直到读到 1，或者等待 RSA 中断产生。
6. 从存储器 `RSA_Z_MEM` 读出运算结果 Z_i ($i \in \{0, 1, \dots, n-1\}$)。
7. 若中断功能已开启，对寄存器 `RSA_CLEAR_INTERRUPT_REG` 写入 1 以清除中断。

运算结束后，寄存器 `RSA_MODE_REG` 中存储的运算子长度信息以及存储器 `RSA_X_MEM` 中的 X_i 、存储器 `RSA_Y_MEM` 中的 Y_i 、存储器 `RSA_M_MEM` 中的 M_i 、寄存器 `RSA_M_PRIME_REG` 中的 M' 都不会变化。但是，存储器 `RSA_Z_MEM` 中的 \bar{r}_i 已经被覆盖。所以当需要连续运算时，只需要更新被覆盖的存储器即可。

17.3.3 大数乘法运算

大数乘法运算实现了 $Z = X \times Y$ 。其中 Z 的长度是运算子 X 、 Y 长度的两倍。所以 RSA 加速器只支持运算子 X 、 Y 长度为 $N = 32 \times x$ ($x \in \{1, 2, 3, \dots, 48\}$) 的大数乘法运算。运算子 Z 的长度 \hat{N} 为 $2 \times N$ 。

大数乘法运算的软件流程为：

1. 对寄存器 `RSA_INTERRUPT_ENA_REG` 写 1 / 0 以开启 / 关闭中断。
2. 配置相关寄存器。对寄存器 `RSA_MODE_REG` 写入 $(\frac{\hat{N}}{32} - 1)$ ，即 $(\frac{N}{16} - 1)$ 。
3. 将 X_i 、 Y_i ($i \in \{0, 1, \dots, n-1\}$) 分别写入存储器 `RSA_X_MEM`、`RSA_Z_MEM`。每块存储器的每一个字刚好存放一个 b 进制数。这些存储器都是低地址存放运算子的低位进制数，高地址存放运算子的高位进制数。
 n 为 $\frac{N}{32}$ 。
 X_i ($i \in \{0, 1, \dots, n-1\}$) 要填充到存储器 `RSA_X_MEM` 中的第 i 个字对应的地址中，但需要注意的是， Y_i ($i \in \{0, 1, \dots, n-1\}$) 并不是要填充到存储器 `RSA_Z_MEM` 中的第 i 个字对应的地址中，而是需要填充到存储器 `RSA_Z_MEM` 中的第 $n+i$ 个字对应的地址中，即存储器 `RSA_Z_MEM` 的基地址加上偏移量 $4 \times (n+i)$ 。
 只需要根据运算子长度，将这两个运算子中有效的数据写入存储器，没有使用到的存储器可以是任意值。
4. 对寄存器 `RSA_MULT_START_REG` 写入 1。

5. 等待运算结束。轮询寄存器 [RSA_IDLE_REG](#) 直到读到 1，或者等待 RSA 中断产生。
6. 从存储器 [RSA_Z_MEM](#) 读出运算结果 Z_i ($i \in \{0, 1, \dots, \hat{n} - 1\}$)。 \hat{n} 为 $2 \times n$ 。
7. 若中断功能已开启，对寄存器 [RSA_CLEAR_INTERRUPT_REG](#) 写入 1 以清除中断。

运算结束后，寄存器 [RSA_MODE_REG](#) 中存储的运算子长度信息以及存储器 [RSA_X_MEM](#) 中的 X_i 都不会变化。但是，存储器 [RSA_Z_MEM](#) 中的 Y_i 已经被覆盖。所以当需要连续运算时，只需要更新必需的寄存器与存储器即可。

17.3.4 控制加速

对于大数模幂运算，ESP32-C3 的 RSA 加速器还特别提供 [SEARCH](#) 和 [CONSTANT_TIME](#) 两个选项，可提高运算速度。默认情况下，这两个选项均处于不加速状态，可以单独使用，也可以同时使用。

具体来说，当这两个选项均处于不加速状态时，求解 $Z = X^Y \bmod M$ 的时间开销完全由运算子长度决定。否则，只要有某个选项携带有加速效果，那么运算的时间开销还与 Y 的 0/1 分布有关。

为了更清楚地说明问题，首先假设 Y 的二进制表示为：

$$Y = (\tilde{Y}_{N-1}\tilde{Y}_{N-2}\cdots\tilde{Y}_{t+1}\tilde{Y}_t\tilde{Y}_{t-1}\cdots\tilde{Y}_0)_2$$

其中，

- N 代表 Y 的长度，
- \tilde{Y}_t 的值为 1，
- $\tilde{Y}_{N-1}, \tilde{Y}_{N-2}, \dots, \tilde{Y}_{t+1}$ 的值均为 0，
- 且 $\tilde{Y}_{t-1}, \tilde{Y}_{t-2}, \dots, \tilde{Y}_0$ 中包括 m 个 0，其余 $t-m$ 全部为 1，即 $\tilde{Y}_{t-1}\tilde{Y}_{t-2}, \dots, \tilde{Y}_0$ 的汉明重量 (Hamming weight) 为 $t-m$ 。

此时，当启动任一选项时：

- [SEARCH](#) 选项 ([RSA_SEARCH_ENABLE](#) 置 1 开启加速)
 - RSA 加速器将忽略所有 \tilde{Y}_i ($i > \alpha$) 位。其中，加速位置 α 可通过 [RSA_SEARCH_POS_REG](#) 寄存器配置。 α 的最大值不能超过 $N-1$ ，否则相当于没有加速；且不建议小于 t ，否则无法正确求解 $Z = X^Y \bmod M$ 。当设置 α 为 t 时，加速效果最佳。此时， $\tilde{Y}_{N-1}, \tilde{Y}_{N-2}, \dots, \tilde{Y}_{t+1}$ 中的 0 位将在运算中全部被忽略。
- [CONSTANT_TIME](#) 选项 ([RSA_CONSTANT_TIME_REG](#) 置 0 开启加速)
 - RSA 加速器在运算过程中将简化对 Y 中 0 位的处理。因此不难想象， Y 中的 0 越多，加速效果越明显。

为了直观地展示这两个选项带来的加速效果，下面通过一个典型实例加以说明。在 $Z = X^Y \bmod M$ 中， N 等于 3072， Y 等于 65537。表 17-1 展示了 4 种选项组合对应的时间开销。注意，这里 [SEARCH](#) 选项开启时设定 α 为 16。

表 17-1. 加速效果

SEARCH 选项	CONSTANT_TIME 选项	时间开销
不加速	不加速	376.405 ms
加速	不加速	2.260 ms
不加速	加速	1.203 ms
加速	加速	1.165 ms

可以看到：

- 当两个选项均处于不加速状态时，时间开销最大。
- 当两个选项均处于加速状态时，时间开销最小。
- 相比于不加速状态，任一选项处于加速状态时的时间开销明显大幅度降低。

17.4 存储器列表

请注意，这里的地址都是相对于 RSA 加速器基地址的地址偏移量（相对地址），详见章节 3 [系统和存储器](#) 中的表 3-4。

名称	描述	大小（字节）	起始地址	结束地址	访问
RSA_M_MEM	存储器 M	384	0x0000	0x017F	R/W
RSA_Z_MEM	存储器 Z	384	0x0200	0x037F	R/W
RSA_Y_MEM	存储器 Y	384	0x0400	0x057F	R/W
RSA_X_MEM	存储器 X	384	0x0600	0x077F	R/W

17.5 寄存器列表

本小节的所有地址均为相对于 RSA 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
RSA_M_PRIME_REG	M' 存储器	0x0800	R/W
RSA_MODE_REG	RSA 长度模式	0x0804	R/W
RSA_CONSTANT_TIME_REG	固定时间选项	0x0820	R/W
RSA_SEARCH_ENABLE_REG	使能 search 加速选项	0x0824	R/W
RSA_SEARCH_POS_REG	search 起始位置	0x0828	R/W
状态/控制寄存器			
RSA_CLEAN_REG	RSA 清除寄存器	0x0808	RO
RSA_MODEXP_START_REG	模幂运算起始位	0x080C	WO
RSA_MODMULT_START_REG	模乘运算起始位	0x0810	WO
RSA_MULT_START_REG	乘法运算起始位	0x0814	WO
RSA_IDLE_REG	RSA 闲置寄存器	0x0818	RO
中断寄存器			
RSA_CLEAR_INTERRUPT_REG	RSA 中断清除寄存器	0x081C	WO
RSA_INTERRUPT_ENA_REG	RSA 中断使能寄存器	0x082C	R/W
版本寄存器			
RSA_DATE_REG	RSA 日期与版本寄存器	0x0830	R/W

17.6 寄存器

本小节的所有地址均为相对于 RSA 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 17.1. RSA_M_PRIME_REG (0x0800)

31

0

0x00000000

Reset

RSA M PRIME REG 此寄存器存储 M' 。(R/W)

Register 17.2. RSA_MODE_REG (0x0804)

[illegible]

RSA_MODE 此寄存器存储模幂运算的模式。(R/W)

Register 17.3. RSA_CLEAN_REG (0x0808)

[illegible]

RSA_CLEAN 一旦存储器初始化结束，此位为 1。(RO)

Register 17.4. RSA_MODEXP_START_REG (0x080C)

[illegible]

RSA_MODEXP_START 写入 1 以开始模幂运算。(WR)

Register 17.5. RSA_MODMULT_START_REG (0x0810)

(reserved)																														1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSA_MODMULT_START 写入 1 以开始模乘运算。(WR)

Register 17.6. RSA_MULT_START_REG (0x0814)

(reserved)																														1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSA_MULT_START 写入 1 以开始乘法运算。(WR)

Register 17.7. RSA_IDLE_REG (0x0818)

(reserved)																														1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSA_IDLE 当 RSA 空闲时，此位为 1。(RO)

Register 17.8. RSA_CLEAR_INTERRUPT_REG (0x081C)

(reserved)																														1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSA_CLEAR_INTERRUPT RSA 中断清除寄存器。写入 1 清除中断。(WR)

Register 17.9. RSA_CONSTANT_TIME_REG (0x0820)

(reserved)																															RSA_CONSTANT_TIME		
31																															1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

RSA_CONSTANT_TIME_REG 控制模幂运算中的 constant_time 选项。0: 加速; 1: 不加速 (默认)。(R/W)

Register 17.10. RSA_SEARCH_ENABLE_REG (0x0824)

(reserved)																															RSA_SEARCH_ENABLE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RSA_SEARCH_ENABLE 控制模幂运算中的 search 选项。1: 加速; 0: 不加速 (默认)。(R/W)

Register 17.11. RSA_SEARCH_POS_REG (0x0828)

(reserved)												RSA_SEARCH_POS																							
31												12	11															0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000														Reset

RSA_SEARCH_POS 模幂运算中的 search 加速选项。用于配置 search 的起始位置。(R/W)

Register 17.12. RSA_INTERRUPT_ENA_REG (0x082C)

																															RSA_INTERRUPT_ENA	
(reserved)																																
31																														1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset

RSA_INTERRUPT_ENA RSA 中断使能寄存器。写入 1 开启中断，默认开启。(R/W)

301
[反馈文档意见](#)

ESP32-C3 TRM (预发布 v0.4)

RSA_DATE 版本控制寄存器 (R/W)。

18 HMAC 加速器 (HMAC)

如 RFC 2104 中所述，HMAC 模块通过 hash 算法和密钥计算得到数据信息的信息认证码 (MAC)。其中的 hash 算法是 SHA-256，长度为 256-bit 的 HMAC 密钥存储在 eFuse 的密钥块中，可配置成不能被软件读取。

18.1 主要特性

- 标准 HMAC-SHA-256 算法
- HMAC 计算的 hash 结果仅支持可配的硬件外设访问（下行模式）
- 兼容挑战-应答身份验证算法
- 生成数字签名外设所需的密钥（下行模式）
- 重启软禁用的 JTAG（下行模式）

18.2 功能描述

HMAC 模块可工作于两种模式，即上行模式和下行模式。上行模式中，由用户提供 HMAC 信息，且用户回读其计算结果；下行模式中，HMAC 模块作为其他内部硬件的密钥导出函数 (KDF)。例如，烧写奇数个 1 至 EFUSE_SOFT_DIS_JTAG，可暂时关闭 JTAG。此时，用户可使用 HMAC 模块在下行模式下暂时重启 JTAG。

芯片上电后，HMAC 模块将检查 eFuse 中是否烧写有 DS 模块所需要的功能密钥，如果存在该密钥，将自动进入下行数字签名模式，完成相应密钥计算。

18.2.1 上行模式

任何支持 HMAC-SHA-256 算法的挑战-应答协议都可以使用该方式。假设挑战-应答协议中的通讯双方称为 A，B，想要交换的完整的数据信息为 M，协议的一般验证流程为：

- A 计算出一个特殊的随机数信息 M
- A 将 M 发送给 B
- B 计算 HMAC 结果（通过 M 和密钥）并将其发送给 A
- A 内部计算 HMAC 结果（通过 M 和密钥）
- A 比较两次计算结果。如比较结果相同，则验证通过了 B 的身份

用户应执行下述步骤计算 HMAC 值：

1. 初始化 HMAC 模块，进入上行模式。
2. 将正确填充的信息写入外设中，一次写入一个块中。
3. 从外设寄存器中回读 HMAC 值。

有关此过程的详细步骤，可参见章节 18.2.5。

18.2.2 下行 JTAG 启动模式

JTAG 可被暂时关闭，此时，用户可使用 HMAC 模块重启 JTAG 功能。该模式下，用户应提供某 eFuse 密钥的 HMAC 计算结果。HMAC 模块将检查用户提供的数值与计算的数值是否匹配。如果二者相同，JTAG 将被启动，直到用户再次调用 HMAC 模块清除计算结果并关闭 JTAG。

eFuse 存储器中有两个参数可以关闭 JTAG 调试：EFUSE_DIS_PAD_JTAG 和 EFUSE_SOFT_DIS_JTAG。前者烧写为 1，JTAG 将被永久关闭；后者烧写奇数个 1，JTAG 将被暂时关闭。详细信息可参见章节 4 [eFuse 控制器 \(EFUSE\)](#)。置位 EFUSE_SOFT_DIS_JTAG 后，可在 HMAC 外设的下行模式下计算得到重启 JTAG 所需的密钥，当用户配置的密钥与计算结果一致时，完成 JTAG 重启。

重启 JTAG：

1. 用户通过初始化时钟和重置 HMAC 信号启动 HMAC 模块，并通过配置 HMAC_SET_PARA_PURPOSE_REG 位进入下行 JTAG 启动模式，等待完成计算任务。详细步骤可参见章节 18.2.5。
2. 用户将 1 写入 HMAC_SOFT_JTAG_CTRL_REG 寄存器进入 JTAG 重启比较模式。
3. 用户将预先在本地使用 SHA-256 和已知的随机密钥对 32 字节的 0x00 进行 HMAC 计算得到的 256-bit 数值结果按照 word 的大端序依次写入 HMAC_WR_JTAG_REG。
4. 如果 HMAC 计算的结果与用户本地计算的数值结果匹配，则 JTAG 重启。否则，JTAG 仍保持关闭状态。
5. 在用户将 1 烧写入寄存器 [HMAC_SET_INVALIDATE_JTAG_REG](#) 或上电重启之前，JTAG 将保持关闭状态。如需再次重启 JTAG，用户需要重复上述步骤。

18.2.3 下行数字签名模式

数字签名 (DS) 模块使用 AES-CBC 加密其参数。HMAC 模块作为密钥导出函数 (KDF) 导出解密上述参数的 AES 密钥。这些密钥存储于 eFuse 密钥块中。

在启用 DS 模块之前，用户必须先通过 HMAC 模块计算得到 DS 模块工作时所需的密钥。详细信息可参见章节 19 [数字签名 \(DS\)](#)。芯片上电后，HMAC 模块将检查 eFuse 中是否烧写有 DS 模块所需要的功能密钥，如果存在该密钥，将自动进入下行数字签名模式，完成相应密钥计算。

18.2.4 HMAC eFuse 配置

每个烧写入 eFuse 密钥块的 HMAC 密钥都有其功能，指定该密钥用于实现哪一功能。HMAC 中，未配置匹配的功能数值的密钥则无法执行计算。当前 HMAC 模块共支持 3 种功能：下行模式下的 JTAG 重启功能和 DS 密钥导出功能以及上行模式下的 HMAC 计算功能。表 18-1 列出了各功能对应的配置寄存器的数值，其中还有一种功能可指定密钥既用于 JTAG 重启，也可用于 DS 密钥导出。

启动 HMAC 模块进行计算之前，用户应先读取 4 [eFuse 控制器 \(EFUSE\)](#) 中 EFUSE_KEY_PURPOSE_x 的值 (x 为 0 ~ 5, eFuse 中共有 6 个密钥)，检查 eFuse 中是否有密钥。例如，在上行模式下，如果 EFUSE_KEY_PURPOSE_0~5 中没有 EFUSE_KEY_PURPOSE_HMAC_UP，则说明 eFuse 中没有上行模式可用的密钥。此时，可按照下述步骤将密钥烧写入 eFuse 中：

1. 准备一个 256-bit HMAC 密钥，将其烧写到空的 eFuse 密钥块 y 中 (eFuse 中共有 6 个密钥块，分别为 4 ~ 9，因此 y 的值为 4 ~ 9。因此，若密钥为 key0，则对应的密钥块为 block4)，并声明该密钥块的功能为 EFUSE_KEY_PURPOSE_ $(y-4)$ 。例如在上行模式下，烧写密钥后，应将 EFUSE_KEY_PURPOSE_HMAC_UP (对应值为 6) 写入 EFUSE_KEY_PURPOSE_ $(y-4)$ 。更多烧写 eFuse 密钥的详细信息，可参见章节 4 [eFuse 控制器 \(EFUSE\)](#)。
2. 配置 eFuse 密钥块读保护功能，使软件无法读取密钥值。用户将步骤中生成的随机密钥安全地存储在其他位置。

请注意，功能为 EFUSE_KEY_PURPOSE_HMAC_DOWN_ALL 的密钥既可用于 JTAG 重启，也可用于 DS 密钥导出。

表 18-1. HMAC 功能及配置数值

功能	模式	数值	描述
JTAG 重启	下行模式	6	EFUSE_KEY_PURPOSE_HMAC_DOWN_JTAG
DS 密钥导出	下行模式	7	EFUSE_KEY_PURPOSE_HMAC_DOWN_DIGITAL_SIGNATURE
HMAC 计算	上行模式	8	EFUSE_KEY_PURPOSE_HMAC_UP
JTAG 重启和 DS 密钥导出	下行模式	5	EFUSE_KEY_PURPOSE_HMAC_DOWN_ALL

配置 HMAC 的功能

用户应将所使用功能对应的数值写入寄存器 `HMAC_SET_PARA_PURPOSE_REG` 完成配置(可参见章节 18.2.5)。否则, HMAC 将终止计算。

选取 eFuse 的密钥块

eFuse 控制器共提供 6 个密钥块, KEY0 ~ 5。用户将编号 `n` 写入寄存器 `HMAC_SET_PARA_KEY_REG`, 表示选择 KEY`n` 作为本次 HMAC 模块运行时使用的密钥。

需要注意的是, eFuse memory 中的密钥在烧写时都定义了功能用途, 只有当 HMAC 的配置功能与 KEY`n` 定义的功能用途相匹配时, HMAC 模块才会执行配置好的计算任务。否则, 返回匹配错误结果并结束当前计算任务。

比如, 如果用户选择了 KEY3 作为本次计算的密钥, 且烧写入 KEY_PURPOSE_3 中的数值为 6 (EFUSE_KEY_PURPOSE_HMAC_DOWN_JTAG), 则参照表 18-1 可知, KEY3 是用于 JTAG 重启的密钥。如果此时寄存器 `HMAC_SET_PARA_PURPOSE_REG` 中配置的数值也是 6, HMAC 外设才会启动 JTAG 重启功能的计算。

18.2.5 调用 HMAC 流程 (详细说明)

用户调用 ESP32-C3 中 HMAC 流程如下:

1. 启动 HMAC 模块
 - (a) 置位寄存器 `SYSTEM_PERIP_CLK_EN1_REG` 中 HMAC 和 SHA 的外设时钟位, 清除寄存器 `SYSTEM_PERIP_RST_EN1_REG` 中相应的外设重启位。关于如何配置时钟和重启信号, 请参见章节 3 系统和存储器。
 - (b) 将数值 1 写入寄存器 `HMAC_SET_START_REG`。
2. 配置 HMAC 密钥和密钥功能
 - (a) 将表示密钥功能 `m` 写入寄存器 `HMAC_SET_PARA_PURPOSE_REG`。表 18-1 描述了数值 `m` 对应的密钥功能, 可参见章节 18.2.4。
 - (b) 通过将数值 `n` 写入寄存器 `HMAC_SET_PARA_KEY_REG`, 选择 eFuse memory 中的 KEY`n` 作为本次计算的密钥 (`n` 的取值范围为 0 ~ 5), 可参见章节 18.2.4。
 - (c) 将数值 1 写入寄存器 `HMAC_SET_PARA_FINISH_REG`, 完成配置工作。
 - (d) 读取寄存器 `HMAC_QUERY_ERROR_REG`。如果返回值为 1, 表明选取的密钥块与配置的密钥功能不匹配, 结束本次计算任务; 如果返回值为 0, 表明选取的密钥块与配置的密钥功能匹配, 可以执行计算流程。

- (e) 如果设置 `HMAC_SET_PARA_PURPOSE_REG` 的数值不为 8，表明 HMAC 模块将工作在下行模式下，跳转到步骤 3；如果设置其数值为 8，表明 HMAC 模块将工作在上行模式下，跳转到步骤 4。

3. 下行模式

- (a) 轮询状态寄存器 `HMAC_QUERY_BUSY_REG`，当读取到该寄存器的值为 0 时，继续下一步骤。
- (b) 为了清除硬件内部(JTAG 或 DS)的计算结果以节省空间,用户可将数值 1 写入寄存器 `HMAC_SET_INVALIDATE_JTAG` 清除 JTAG 密钥生成的结果；也可将数值 1 写入寄存器 `HMAC_SET_INVALIDATE_DS_REG` 清除数字签名密钥生成的结果。此后，如需重启 JTAG 或 DS 模块，需重新进行 HMAC 调用流程。

4. 上行模式下传输数据块 Block_n ($n \geq 1$)

- (a) 轮询状态寄存器 `HMAC_QUERY_BUSY_REG`，当读取到该寄存器的值为 0 时，继续下一步骤。
- (b) 将 512 位的数据块 Block_n 写入寄存器 `HMAC_WDATA0~15_REG` 中，随后将数值 1 写入寄存器 `HMAC_SET_MESSAGE_ONE_REG`，HMAC 模块将计算该数据块。
- (c) 轮询状态寄存器 `HMAC_QUERY_BUSY_REG`，当读取到该寄存器的值为 0 时，继续下一步骤。
- (d) 根据待处理数据总比特数是否为 512 的整数倍，将产生不同数据块。
- 如果待处理数据总比特数是 512 的整数倍，有以下 3 种选项：
 - i. 如果 Block_{n+1} 存在，将数值 1 写入寄存器 `HMAC_SET_MESSAGE_ING_REG`，令 $n = n + 1$ ，随后跳转到步骤 4.(b)。
 - ii. 如果 Block_n 是最后一个待处理数据块，用户希望由硬件进行 SHA 附加填充，将数值 1 写入寄存器 `HMAC_SET_MESSAGE_END_REG`，随后跳转到步骤 6。
 - iii. 如果 Block_n 是最后一个填充的数据块，且用户已在软件中进行 SHA 附加填充时，将数值 1 写入寄存器 `HMAC_SET_MESSAGE_PAD_REG`，随后跳转到步骤 5。
 - 如果待处理数据总比特数不是 512 的倍数，有以下 3 种选项。注意，这种情况下用户应对数据进行 SHA 附加填充，且填充后待输入数据总比特数应为 512 的整数倍。
 - i. 如果 Block_n 是唯一一个数据块，且 $n = 1$ ，同时 Block₁ 已经包含了所有的填充位，则将数值 1 写入寄存器 `HMAC_ONE_BLOCK_REG`，随后跳转到步骤 6。
 - ii. 如果 Block_n 是倒数第二个填充数据块，将数值 1 写入寄存器 `HMAC_SET_MESSAGE_PAD_REG`，执行附加填充比特操作，随后跳转到步骤 5。
 - iii. 如果 Block_n 既不是最后一个也不是倒数第二个数据块，将数值 1 写入寄存器 `HMAC_SET_MESSAGE_ING_REG`，令 $n = n + 1$ ，随后跳转到步骤 4.(b)。

5. 进行 SHA 附加填充

- (a) 用户根据 18.3.1 章节描述对最后一个数据块进行 SHA 附加填充，并将该数据块写入寄存器 `HMAC_WDATA0~15_REG`，随后将数值 1 写入寄存器 `HMAC_SET_MESSAGE_ONE_REG`，HMAC 模块开始计算该数据块。
- (b) 跳转到步骤 6。

6. 读取上行模式下的结果 hash 值

- (a) 轮询状态寄存器 `HMAC_QUERY_BUSY_REG`，当读取到该寄存器的值为 0 时，继续下一步。
- (b) 从寄存器 `HMAC_RDATA0~7_REG` 中读取 hash 结果数值。
- (c) 将数值 1 写入寄存器 `HMAC_SET_RESULT_FINISH_REG`，结束当次计算。

(d) 上行模式下的操作完成。

说明:

DS 模块和 HMAC 模块可直接调用或在内部使用 SHA 加速器，但不能同时与其共享硬件资源。因此在 HMAC 模块运行过程中，SHA 模块无法被 CPU 和 DS 模块调用。

18.3 HMAC 算法细节

18.3.1 附加填充比特

HMAC 模块中采用 SHA-256 作为加密 HASH 算法。该算法中，若待输入数据的总比特数不是 512 的倍数，用户须在软件中应用 SHA-256 附加填充算法。SHA-256 附加填充算法与 [FIPS PUB 180-4](#) 中章节“Padding the Message”所描述相同。下行模式下，用户无需输入数据或进行数据填充，HMAC 模块默认使用 32-byte 0x00 方式重启 JTAG，以及 32-byte 0xff 方式生成 DS 模块的 AES 密钥。

如图 18-1 所示，假设待处理数据长度为 m 个比特，填充步骤如下：

1. 在待处理数据末尾附加 1 个比特长度的数值“1”。
2. 附加 k 个比特的数值“0”。其中， k 为满足 $m + 1 + k \equiv 448(\text{mod}512)$ 的最小非负数。
3. 附加一个 64 位的整数值作为二进制块。该二进制块的内容为待填充数据作为一个大端二进制整数值 m 的长度。

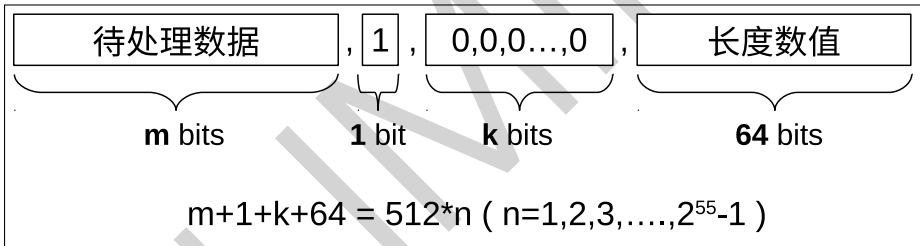


图 18-1. HMAC 附加填充比特示意图

下行模式下，用户无需输入数据或进行附加填充。上行模式下，若待填充数据总比特数是 512 的整数倍，则用户可通过将数值 1 写入 `HMAC_SET_MESSGAE_END_REG` 配置由硬件完成 SHA 附加填充操作，也可通过将数值 1 写入 `HMAC_SET_MESSAGE_PAD_REG` 自行完成填充操作；若待填充数据总比特数不是 512 的整数倍，则用户只能自行完成 SHA 附加填充操作。数据填充操作完成后，用户应参照章节 18.2.5 完成后续配置。

18.3.2 HMAC 算法结构

HMAC 模块中应用的算法结构示意图如 18-2 所示。这是 RFC 2104 中描述的标准 HMAC 算法。

图 18-2 中，

1. ipad 是由 64 个 0x36 字节组成的 512-bit 数据块。
2. opad 是由 64 个 0x5c 字节组成的 512-bit 数据块。

首先，HMAC 模块在 256-bit 的密钥 K 的比特序列后附加上 256-bit 的 0 序列，得到 512-bit 的 K_0 。再对 K_0 和 ipad 进行异或运算，得到 512-bit 的 S_1 。将总比特数为 512 倍数的待输入数据附加到 512-bit 的 S_1 数值后，使用 SHA-256 加密算法计算得到 256-bit 的 H_1 。

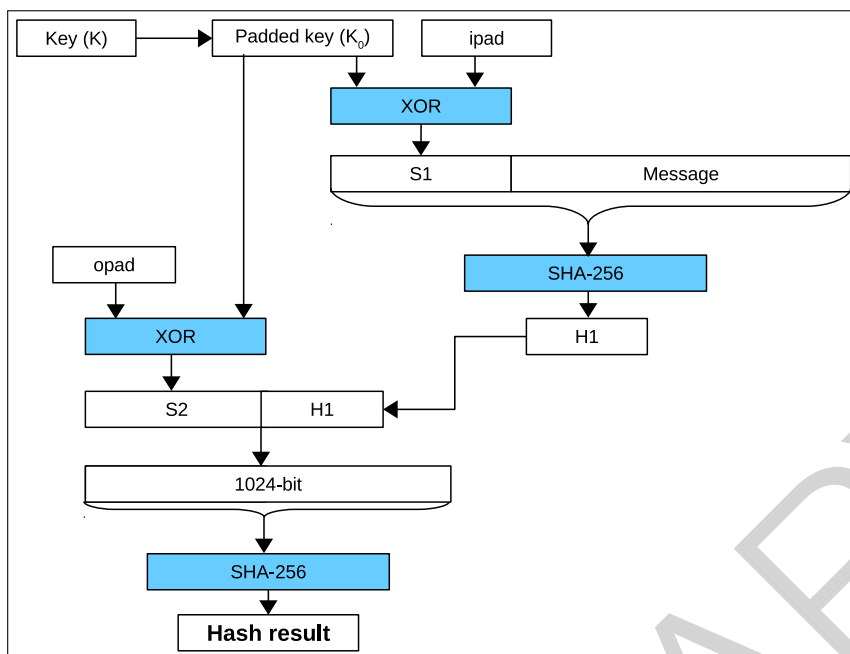


图 18-2. HMAC 结构示意图

HMAC 模块通过对 K_0 和 opad 进行异或运算得到 S2，将 256-bit 的 hash 计算结果附加到 512-bit 的 S2 数值后，得到 768-bit 长度的序列，使用 18.3.1 章节中描述的 SHA 附加填充算法将该序列填充成 1024-bit 的序列，最后使用 SHA-256 加密算法计算得到的最终 hash 结果 (256-bit)。

18.4 寄存器列表

本小节的所有地址均为相对于 HMAC 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

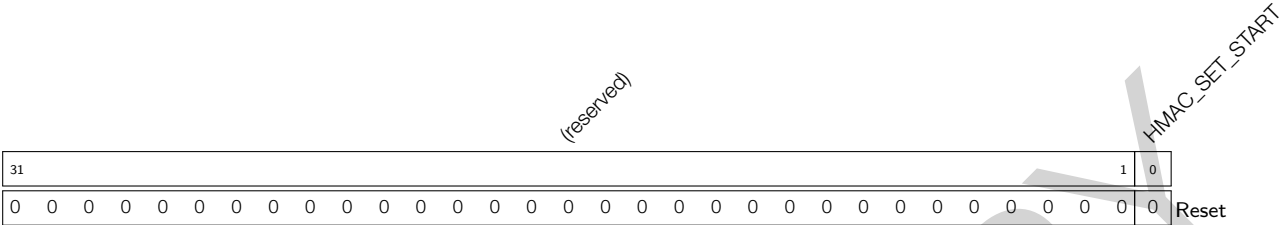
名称	描述	地址	访问
Control/Status Registers			
HMAC_SET_START_REG	HMAC 开始控制寄存器	0x0040	WO
HMAC_SET_PARA_FINISH_REG	HMAC 配置完成寄存器	0x004C	WO
HMAC_SET_MESSAGE_ONE_REG	HMAC 信息控制寄存器	0x0050	WO
HMAC_SET_MESSAGE_ING_REG	HMAC 信息继续寄存器	0x0054	WO
HMAC_SET_MESSAGE_END_REG	HMAC 信息终止寄存器	0x0058	WO
HMAC_SET_RESULT_FINISH_REG	HMAC 读取结果完成寄存器	0x005C	WO
HMAC_SET_INVALIDATE_JTAG_REG	注销 JTAG 结果寄存器	0x0060	WO
HMAC_SET_INVALIDATE_DS_REG	注销数字签名结果寄存器	0x0064	WO
HMAC_QUERY_ERROR_REG	存储用户配置的密钥和功能的配对结果	0x0068	RO
HMAC_QUERY_BUSY_REG	存储 HMAC 模块的忙碌状态	0x006C	RO
configuration Registers			
HMAC_SET_PARA_PURPOSE_REG	HMAC 参数配置寄存器	0x0044	WO
HMAC_SET_PARA_KEY_REG	HMAC 密钥配置寄存器	0x0048	WO
HMAC_SOFT_JTAG_CTRL_REG	重启 JTAG 寄存器 0	0x00F8	WO
HMAC_WR_JTAG_REG	重启 JTAG 寄存器 1	0x00FC	WO
HMAC Message Block			
HMAC_WR_MESSAGE_0_REG	信息寄存器 0	0x0080	WO
HMAC_WR_MESSAGE_1_REG	信息寄存器 1	0x0084	WO
HMAC_WR_MESSAGE_2_REG	信息寄存器 2	0x0088	WO
HMAC_WR_MESSAGE_3_REG	信息寄存器 3	0x008C	WO
HMAC_WR_MESSAGE_4_REG	信息寄存器 4	0x0090	WO
HMAC_WR_MESSAGE_5_REG	信息寄存器 5	0x0094	WO
HMAC_WR_MESSAGE_6_REG	信息寄存器 6	0x0098	WO
HMAC_WR_MESSAGE_7_REG	信息寄存器 7	0x009C	WO
HMAC_WR_MESSAGE_8_REG	信息寄存器 8	0x00A0	WO
HMAC_WR_MESSAGE_9_REG	信息寄存器 9	0x00A4	WO
HMAC_WR_MESSAGE_10_REG	信息寄存器 10	0x00A8	WO
HMAC_WR_MESSAGE_11_REG	信息寄存器 11	0x00AC	WO
HMAC_WR_MESSAGE_12_REG	信息寄存器 12	0x00B0	WO
HMAC_WR_MESSAGE_13_REG	信息寄存器 13	0x00B4	WO
HMAC_WR_MESSAGE_14_REG	信息寄存器 14	0x00B8	WO
HMAC_WR_MESSAGE_15_REG	信息寄存器 15	0x00BC	WO
HMAC Upstream Result			
HMAC_RD_RESULT_0_REG	Hash 结果寄存器 0	0x00C0	RO
HMAC_RD_RESULT_1_REG	Hash 结果寄存器 1	0x00C4	RO
HMAC_RD_RESULT_2_REG	Hash 结果寄存器 2	0x00C8	RO
HMAC_RD_RESULT_3_REG	Hash 结果寄存器 3	0x00CC	RO
HMAC_RD_RESULT_4_REG	Hash 结果寄存器 4	0x00D0	RO

名称	描述	地址	访问
HMAC_RD_RESULT_5_REG	Hash 结果寄存器 5	0x00D4	RO
HMAC_RD_RESULT_6_REG	Hash 结果寄存器 6	0x00D8	RO
HMAC_RD_RESULT_7_REG	Hash 结果寄存器 7	0x00DC	RO
Control/Status Registers			
HMAC_SET_MESSAGE_PAD_REG	软件填充寄存器	0x00F0	WO
HMAC_ONE_BLOCK_REG	One block 信息寄存器	0x00F4	WO
Version Register			
HMAC_DATE_REG	版本控制寄存器	0x01FC	R/W

18.5 寄存器

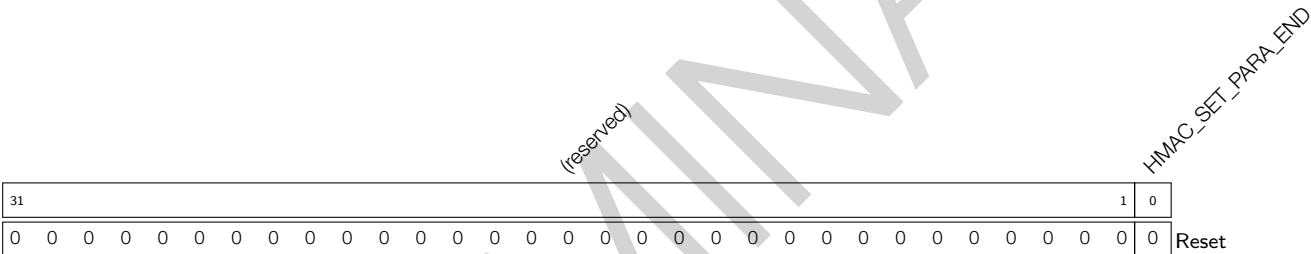
本小节的所有地址均为相对于 HMAC 加速器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 18.1. HMAC_SET_START_REG (0x0040)



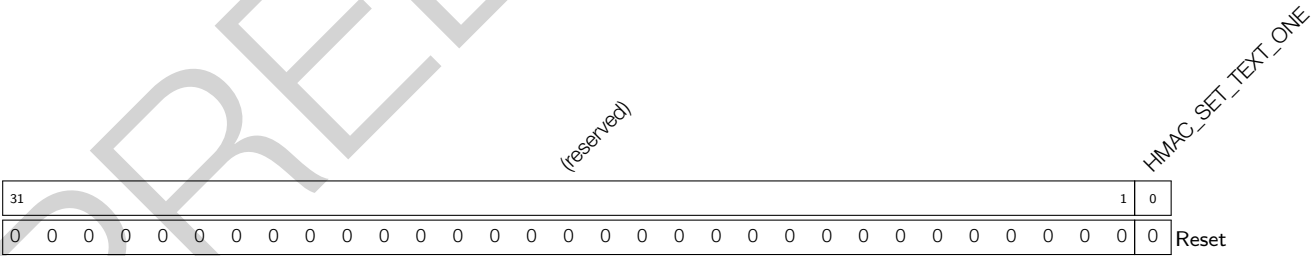
HMAC_SET_START 置 1 启动 HMAC。(WO)

Register 18.2. HMAC_SET_PARA_FINISH_REG (0x004C)



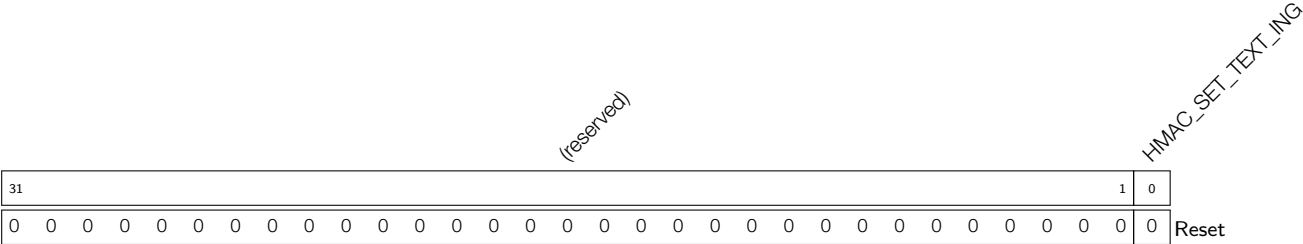
HMAC_SET_PARA_END 置 1 完成 HMAC 配置。(WO)

Register 18.3. HMAC_SET_MESSAGE_ONE_REG (0x0050)



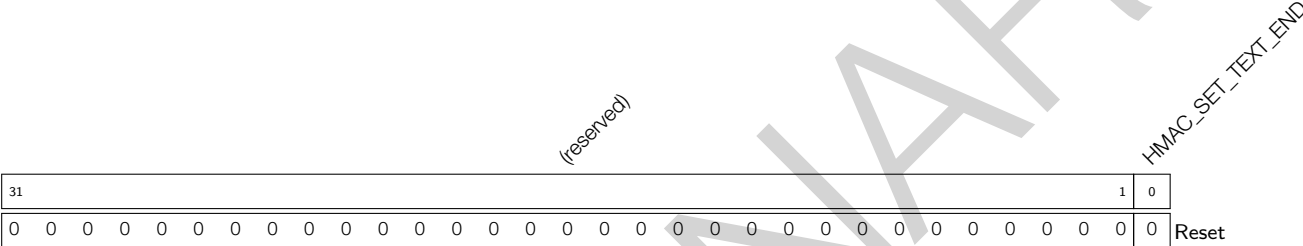
HMAC_SET_TEXT_ONE 调用 SHA 计算一个数据块。(WO)

Register 18.4. HMAC_SET_MESSAGE_ING_REG (0x0054)



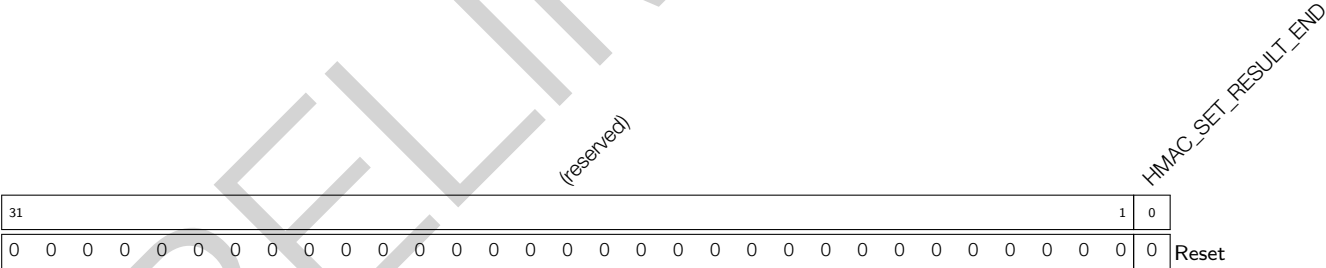
HMAC_SET_TEXT_ING 置 1 表明仍存在未处理的数据块。(WO)

Register 18.5. HMAC_SET_MESSAGE_END_REG (0x0058)



HMAC_SET_TEXT_END 置 1 开始硬件填充。(WO)

Register 18.6. HMAC_SET_RESULT_FINISH_REG (0x005C)



HMAC_SET_RESULT_END 置 1 退出上行模式，并清空计算结果。(WO)

Register 18.7. HMAC_SET_INVALIDATE_JTAG_REG (0x0060)

(reserved)																														HMAC_SE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
31																														1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HMAC_SET_INVALIDATE_JTAG 置 1 清空下行模式下 JTAG 重启功能的计算结果。(WO)

Register 18.8. HMAC_SET_INVALIDATE_DS_REG (0x0064)

(reserved)																															HMAC_Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

HMAC_SET_INVALIDATE_DS 置 1 清空下行模式下 DS 功能的计算结果。(WO)

Register 18.9. HMAC_QUERY_ERROR_REG (0x0068)

(reserved)																															HMAC_QU																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HMAC_QUIREY_CHECK 指示 HMAC 密钥与功能是否匹配。(RO)

- 0: HMAC 密钥和功能匹配。
- 1: 错误。

Register 18.10. HMAC_QUERY_BUSY_REG (0x006C)

(reserved)																																HMAC_BUSY_STATE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HMAC_BUSY_STATE 指示 HMAC 是否处于忙碌状态。执行计算任务之前，请确保 HMAC 已空闲。
(RO)

- 0: 空闲。
- 1: HMAC 仍处于工作状态。

Register 18.11. HMAC_SET_PARA_PURPOSE_REG (0x0044)

(reserved)																										HMAC_PURPOSE_SET																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																										4				3				0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
0																										0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0				0			

HMAC_PURPOSE_SET 设置 HMAC 功能，请参阅表 18-1。(WO)

Register 18.12. HMAC_SET_PARA_KEY_REG (0x0048)

(reserved)																															HMAC_KEY_SET					
31																															3		2		0	
0 0																															0		Reset			

HMAC_KEY_SET 选择 HMAC 密钥。共有 6 个密钥，编号 0 至 5，将选择的密钥编号写入该字段即可。(WO)

Register 18.13. HMAC_WR_MESSAGE_ n _REG (n : 0-15) (0x0080+4* n)

HMAC_WDATA_0																															
31																															0
0																															
																															Reset

HMAC_WDATA_ n 存储信息的第 n 个 32 位数据信息。(WO)

Register 18.14. HMAC_RD_RESULT_ n _REG (n : 0-7) (0x00C0+4* n)

HMAC_RDATA_0																															
31																															0
0																															
																															Reset

HMAC_RDATA_ n 读取 hash 结果的第 n 个 32 位。(RO)

Register 18.15. HMAC_SET_MESSAGE_PAD_REG (0x00F0)

(reserved)																										HMAC_SET_TEXT_PAD																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
31																										1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HMAC_SET_TEXT_PAD 置 1 表明由软件执行填充操作。(WO)

Register 18.16. HMAC_ONE_BLOCK_REG (0x00F4)

(reserved)																															HMAC_SET_ONE_BLOCK																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

HMAC_SET_ONE_BLOCK 置 1 表明无需填充。(WO)

Register 18.17. HMAC_SOFT_JTAG_CTRL_REG (0x00F8)

(reserved)																																HMAC_SOFT_JTAG_CTRL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																																1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HMAC_SOFT_JTAG_CTRL 置 1 开启 JTAG 验证模式。(WO)

Register 18.18. HMAC_WR_JTAG_REG (0x00FC)

HMAC_WR_JTAG																															
31																															0
x																															
Reset																															

HMAC_WR_JTAG 重启 JTAG 比较数值输入寄存器。(WO)

Register 18.19. HMAC_DATE_REG (0x00F8)

(reserved)			HMAC_DATE																												
31	30	29																													0
0	0	0x20190402																												Reset	

HMAC_DATE 版本控制寄存器。(R/W)

19 数字签名 (DS)

19.1 概述

数字签名技术在密码学算法层面上用于验证消息的真实性和完整性。这可用于向服务器验证设备自身身份，或验证消息是否未被篡改。

ESP32-C3 包含一个数字签名 (Digital Signature, DS) 模块，可基于 RSA 高效生成数字签名。数字签名模块使用预先加密的参数计算出签名，HMAC 作为密钥导出函数加密这些参数。反过来，HMAC 则使用 eFuse 作为输入密钥。上述整个过程都发生在硬件层面，因此在计算过程中，不论是解密 RSA 参数的密钥还是用于 HMAC 密钥导出函数的输入密钥都对软件不可见。

19.2 主要特性

- RSA 数字签名支持密钥长度最大为 3072 位
- 私钥数据已加密，并且只能由 DS 读取
- SHA-256 摘要用于保护私钥数据免遭攻击者篡改

19.3 功能描述

19.3.1 概述

DS 模块计算 RSA 签名操作 $Z = X^Y \bmod M$ ，其中 Z 是签名， X 是输入消息， Y 和 M 是 RSA 私钥参数。

私钥参数以密文形式存储在 flash 中。对其解密而使用的密钥 (DS_KEY) 只能由 DS 模块通过 HMAC 模块获取，并且 HMAC 模块求解该密钥所需的一切输入信息 ($HMAC_KEY$) 只存放在 eFuse 中且只允许被 HMAC 模块访问。这意味着只有 DS 硬件才能解密私钥密文，软件获取不到私钥明文。关于 eFuse 和 HMAC 的相关细节请参照章节 4 [eFuse 控制器 \(EFUSE\)](#) 和章节 18 [HMAC 加速器 \(HMAC\)](#)。

需要签名时，软件直接将输入消息 X 发送到 DS 外设。RSA 签名计算完成后，软件将读取签名结果 Z 。

为方便描述，这里约定几个符号，它们的作用域局限于本章。

- 1^s 表示一个完全由“1”组成的长度为 s 位的位串。
- $[x]_s$ 一个长度为 s 位的位串，要求 s 为 8 的整数倍。如果 x 是一个数 ($x < 2^s$)，那么其在位串中遵循小端字节序。 x 可以是一个变量，例如 $[Y]_{4096}$ ，或一个十六进制的常数，比如 $[0x0C]_8$ 。根据需要， $[x]_t$ 右边可以加上 $(s - t)$ 个 0，使字符串长度扩展成 s 位，得到 $[x]_s$ 。例如： $[0x05]_8 = 00000101$ ， $[0x05]_{16} = 0000010100000000$ ， $[0x0005]_{16} = 0000000000000101$ ， $[0x13]_8 = 00010011$ ， $[0x13]_{16} = 0001001100000000$ ， $[0x0013]_{16} = 0000000000010011$ 。
- \parallel 表示位串粘接操作符，用于将两个位串前后粘成一个较长的位串。

19.3.2 私钥运算符

私钥运算符 Y (私钥指数) 和 M (密钥模数) 由用户生成。它们具有特定的 RSA 密钥长度 (最大为 3072 位)。RSA 签名操作还额外需要两个运算符，即参数 \bar{r} 和 M' ，这两个参数需要软件由 Y 和 M 运算得到。

运算符 Y 、 M 、 \bar{r} 和 M' 与验证摘要一起由用户加密并以密文 C 的形式存储。密文 C 输入到 DS 模块之后先由硬件解密，然后参与 RSA 签名运算。如何生成 C 请参考章节 19.3.2。

DS 模块支持运算子长度为 $N = 32 \times x$ ($x \in \{1, 2, 3, \dots, 96\}$) 的 RSA 签名运算 $Z = X^Y \bmod M$ ，需要满足 RSA 计算的运算子长度要求，即 Z 、 X 、 Y 、 M 和 \bar{r} 的位宽必须相同，但必须为这 96 种中的其中一种，而 M' 的位宽始终是 32。更多 RSA 计算相关信息请参考章节 17 [RSA 加速器 \(RSA\)](#) 中的 17.3.1 [大数模幂运算](#) 部分。

19.3.3 软件需要做的准备工作

如果用户想使用 DS 模块进行数字签名，软件和硬件必须紧密配合才可以顺利完成，并且软件需要做一系列准备工作，如图 19-1 所示。图中左半边给出了在硬件开始 RSA 签名计算之前，软件需要做哪些准备工作。图中右半边展示了硬件在整个签名计算的过程中具体会做些什么。

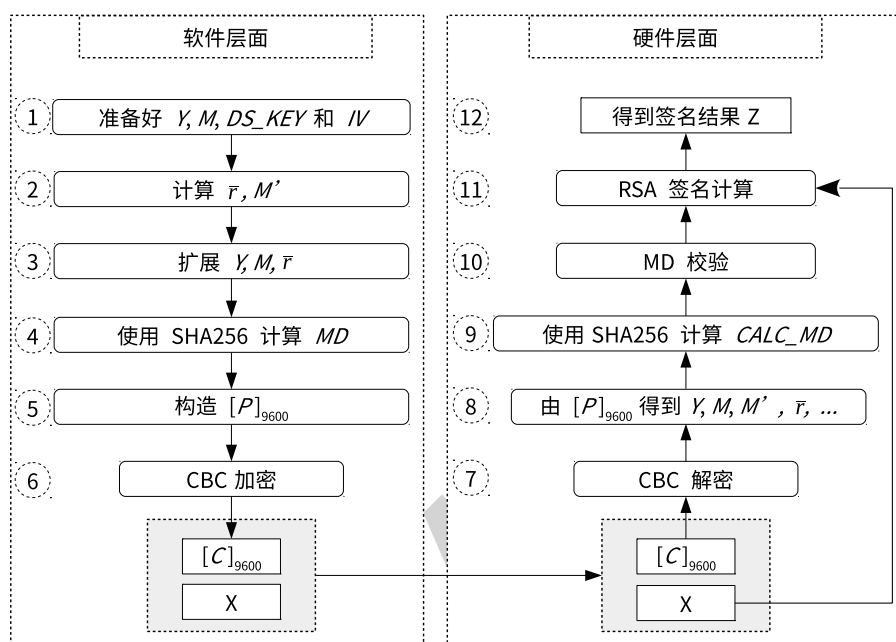


图 19-1. 软件准备工作与硬件工作流程

说明：

- 对于多次签名计算，软件准备工作（图 19-1 左侧）只需要进行一次，但对于每一次签名计算，硬件运算（图 19-1 右侧）都需要重新进行。

用户需要依照图 19-1 指定的步骤来计算 C 。详细的过程描述如下：

- 步骤 1:** 按照章节 19.3.2 所述准备好 RSA 私钥运算子 Y 和 M ，它们应符合运算子的长度要求。记 $[L]_{32} = \frac{N}{32}$ （比如，对于 RSA 3072， $[L]_{32} = [0x60]_{32}$ ）。还需要准备好 $[HMAC_KEY]_{256}$ ，并计算出 $[DS_KEY]_{256}$ ，即 $DS_KEY = HMAC\text{-}SHA256([HMAC_KEY]_{256}, 1^{256})$ 。另外还需要引入一个随机的 $[IV]_{128}$ ，它需要符合 AES-CBC 块加密算法的要求。有关 AES 更多信息，请参考章节 16 [AES 加速器 \(AES\)](#)。
- 步骤 2:** 根据 M 求解 \bar{r} 和 M' 。
- 步骤 3:** 扩展 Y 、 M 和 \bar{r} ，得到 $[Y]_{3072}$ 、 $[M]_{3072}$ 和 $[\bar{r}]_{3072}$ 。由于 Y 、 M 和 \bar{r} 的最大位宽为 3072，运算子位宽小于 3072 的运算子需要扩展至 3072，位宽等于 3072 则不需要扩展。
- 步骤 4:** 使用 SHA-256 计算 MD 校验码： $[MD]_{256} = SHA256([Y]_{3072} || [M]_{3072} || [\bar{r}]_{3072} || [M']_{32} || [L]_{32} || [IV]_{128})$
- 步骤 5:** 构造 $[P]_{9600} = ([Y]_{3072} || [M]_{3072} || [\bar{r}]_{3072} || [Box]_{384})$ 。这里 $[Box]_{384} = ([MD]_{256} || [M']_{32} || [L]_{32} || [\beta]_{64})$ ，其中 $[\beta]_{64}$ 是符合 PKCS#7 封装方式的追加码，由 8 个字节码 0x80 组成，即 $[0x0808080808080808]_{64}$ ，目的在于使 P 的长度为 128 位的整数倍。

- **步骤 6:** 计算密文形式的私钥参数 $C = [C]_{9600} = \text{AES-CBC-ENC}([P]_{9600}, [DS_KEY]_{256}, [IV]_{128})$, 长度为 1200 字节。 C 也可以表示为 $C = [C]_{9600} = ([\hat{Y}]_{3072} || [\hat{M}]_{3072} || [\hat{r}]_{3072} || [\hat{Box}]_{384})$, 其中 $[\hat{Y}]_{3072}$ 、 $[\hat{M}]_{3072}$ 、 $[\hat{r}]_{3072}$ 、 $[\hat{Box}]_{384}$ 是 C 的四个子参数, 分别对应 $[Y]_{3072}$ 、 $[M]_{3072}$ 、 $[r]_{3072}$ 、 $[Box]_{384}$ 的密文形式。

19.3.4 硬件工作流程

每次需要计算数字签名时, 都会触发硬件操作。硬件需要三个输入信息: 预先生成的私钥密文 C 、唯一的消息 X 、 IV 。

DS 模块的工作流程可以分为如下三个阶段:

1. 解析阶段, 即图 19-1 中的步骤 7 和步骤 8

解析过程是图 19-1 中步骤 6 的逆过程。DS 模块将调用 AES 硬件加速器以 CBC 块模式对输入的密文信息 C 进行解密, 获取明文信息。该过程可以表示为 $P = \text{AES-CBC-DEC}(C, DS_KEY, IV)$, 其中 IV 就是 $[IV]_{128}$, 由用户直接指定; $[DS_KEY]_{256}$ 由硬件 HMAC 提供, 由存储在 eFuse 中的 $HMAC_KEY$ 得到, 软件无法获取。

显然, DS 模块能够通过 P 解析出 $[Y]_{3072}$ 、 $[M]_{3072}$ 、 $[r]_{3072}$ 、 $[M']_{32}$ 、 $[L]_{32}$ 、MD 校验码和追加码 $[\beta]_{64}$, 这相当于步骤 5 的逆过程。

2. 校验阶段, 即图 19-1 中的步骤 9 和步骤 10

DS 模块会执行两种校验操作: MD 校验和填充 (padding) 校验。由于填充校验和 MD 校验同步进行, 因此填充校验不在图 19-1 中体现。

- MD 校验——DS 模块调用 SHA-256 进行哈希计算获取哈希值 $[CALC_MD]_{256}$ (即步骤 4), 然后将 $[CALC_MD]_{256}$ 与 MD 校验码 $[MD]_{256}$ 作比较, 当且仅当二者相同时, MD 校验通过。
- 填充校验——DS 模块将检查解析阶段解析出的追加码 $[\beta]_{64}$ 是否符合 PKCS#7 标准, 当且仅当符合标准时, 填充校验通过。

如果 MD 校验通过, DS 模块将执行后续计算; 否则 DS 模块拒绝执行。如果填充校验失败, 将生成警告信息, 但不会影响 DS 模块的后续操作。

3. 计算阶段, 即图 19-1 中的步骤 11 和步骤 12

DS 模块将把用户输入的 X , 以及解析得到的 Y 、 M 和 r 都视为大数, 结合解析得到的 M' , 构成了大数模幂运算 $X^Y \bmod M$ 的所有必要输入参数。大数模幂运算的运算长度由 L 的值唯一指定。DS 模块调用 RSA 硬件加速器完成大数模幂运算 $Z = X^Y \bmod M$, Z 为签名结果。

19.3.5 软件工作流程

每次需要计算数字签名时, 都应执行以下软件操作。输入消息是预先生成的私钥密文 C 、唯一的消息 X 、 IV 。这些软件步骤触发章节 19.3.4 中描述的硬件工作流程。下述流程基于一个假设: 软件已经调用了 HMAC 外设, 硬件上 HMAC 已经根据 $HMAC_KEY$ 计算出了 DS_KEY 。

1. **准备工作:** 准备好 C 、 X 、 IV 。具体方法请参考章节 19.3.3。
2. **启动 DS:** 对寄存器 `DS_SET_START_REG` 写 1。
3. **检查 DS_KEY 是否已经准备好:** 轮询寄存器 `DS_QUERY_BUSY_REG` 直到读到 0。

如果 `DS_QUERY_BUSY_REG` 超过 1 ms 还没读到 0, 则说明 HMAC 未被调用。此时, 软件应当读寄存器 `DS_QUERY_KEY_WRONG_REG`, 根据返回值判断具体是哪一种情况。

- 如果读到零值, 说明 HMAC 未被调用。

- 如果读到非零值 (1 ~ 15)，则说明 HMAC 被调用过，但是 DS 模块没有拿到 *DS_KEY*，原因可能是有其他程序的干扰。
4. **配置寄存器**：将 *IV* block 写入寄存器 *DS_IV_m_REG* (*m*: 0 ~ 3)。有关 *IV* block 的更多信息，请参考章节 16 *AES 加速器 (AES)*。
 5. **将 *X* 写入存储器 *DS_X_MEM***：将 X_i ($i \in \{0, 1, \dots, n-1\}$) 写入存储器 *DS_X_MEM*，容量为 96 个字 (word)，其中 $n = \frac{N}{32}$ 。每一个字刚好存放一个 *b* 进制数。存储器的低地址存放运算器的低位进制数，高地址存放运算器的高位进制数。当 *X* 的长度小于 96 个字时，存储器 *DS_X_MEM* 中有一部分空间未使用，该部分空间中的数据可以是任意值。
 6. **将 *C* 写入存储器**：即将 *C* 中的四个子参数分别写入对应的存储器。
 - 将 \hat{Y}_i ($i \in \{0, 1, \dots, 95\}$) 写入存储器 *DS_Y_MEM*。
 - 将 \hat{M}_i ($i \in \{0, 1, \dots, 95\}$) 写入存储器 *DS_M_MEM*。
 - 将 \hat{r}_i ($i \in \{0, 1, \dots, 95\}$) 写入存储器 *DS_RB_MEM*。
 - 将 \widehat{Box}_i ($i \in \{0, 1, \dots, 11\}$) 写入存储器 *DS_BOX_MEM*。

其中存储器 *DS_Y_MEM*、*DS_M_MEM*、*DS_RB_MEM* 的容量均为 96 个字，而存储器 *DS_BOX_MEM* 容量只有 12 个字。每一个字刚好存放一个 *b* 进制数。存储器的低地址存放运算器的低位进制数，高地址存放运算器的高位进制数。
 7. **启动计算**：对寄存器 *DS_SET_ME_REG* 写入 1。
 8. **等待运算结束**：轮询寄存器 *DS_QUERY_BUSY_REG* 直到读到 0。
 9. **检查校验结果**：读寄存器 *DS_QUERY_CHECK_REG*，根据返回值决定后续操作。
 - 如果返回值为 0，则说明填充校验通过，MD 校验通过，可以继续读取 *Z* 结果值。
 - 如果返回值为 1，则说明填充校验通过，但 MD 校验失败。*Z* 结果值全零无效，跳至步骤 11。
 - 如果返回值为 2，则说明填充校验通过失败，但 MD 校验通过，用户可以继续读取 *Z* 结果值。但仍需注意的是，数据填充不符合 PKCS#7 封装方式，这可能不是你想要的。
 - 如果返回值为 3，则说明填充校验失败，且 MD 校验失败。这种情况说明有致命错误发生。*Z* 结果值全零无效。跳至步骤 11。
 10. **读出运算结果**：从存储器 *DS_Z_MEM* 读出运算结果 Z_i ($i \in \{0, 1, \dots, n-1\}$)，其中 $n = \frac{N}{32}$ 。*Z* 以小端字节序存储在存储器中。
 11. **退出计算环境**：对寄存器 *DS_SET_FINISH_REG* 写入 1，然后轮询寄存器 *DS_QUERY_BUSY_REG* 直到读到 0。

DS 退出计算环境后，所有输入/输出寄存器和存储器中的数据都已经被抹除（清零）。

19.4 存储器列表

请注意，这里的起始地址和结束地址都是相对于基地址的地址偏移量（相对地址）。请参阅章节 3 系统和存储器中的表 3-4 获取 DS 模块的基地址。

名称	描述	大小（字节）	起始地址	结束地址	访问
DS_Y_MEM	存储器 Y	384	0x0000	0x017F	WO
DS_M_MEM	存储器 M	384	0x0200	0x037F	WO
DS_RB_MEM	存储器 \bar{r}	384	0x0400	0x057F	WO
DS_BOX_MEM	存储器 Box	48	0x0600	0x062F	WO
DS_X_MEM	存储器 X	384	0x0800	0x097F	WO
DS_Z_MEM	存储器 Z	384	0x0A00	0x0B7F	RO

19.5 寄存器列表

本小节的所有地址均为相对于数字签名基地址的地址偏移量（相对地址），具体基地址请见章节 3 [系统和存储器](#) 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
DS_IV_0_REG	IV block 数据	0x0630	WO
DS_IV_1_REG	IV block 数据	0x0634	WO
DS_IV_2_REG	IV block 数据	0x0638	WO
DS_IV_3_REG	IV block 数据	0x063C	WO
状态/控制寄存器			
DS_SET_START_REG	启动 DS 模块	0x0E00	WO
DS_SET_ME_REG	开始计算	0x0E04	WO
DS_SET_FINISH_REG	结束计算	0x0E08	WO
DS_QUERY_BUSY_REG	DS 模块状态	0x0E0C	RO
DS_QUERY_KEY_WRONG_REG	查询 <i>DS_KEY</i> 未准备好的原因	0x0E10	RO
DS_QUERY_CHECK_REG	查询校验结果	0x0814	RO
版本寄存器			
DS_DATE_REG	版本控制寄存器	0x0820	R/W

19.6 寄存器

本小节的所有地址均为相对于数字签名基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 19.1. DS_IV_ *m*_REG (*m*: 0-3) (0x0630+4**m*)

31	0
0x00000000	

Reset

DS_IV_*m***_REG** (*m*: 0-3) *IV* block 数据。(WO)

Register 19.2. DS SET START REG (0x0E00)

Diagram of the DS_SET_START register structure:

31																														1	0	
																															Reset	

The register is divided into two sections: **reserved** (bits 31-1) and **DS_SET_START** (bits 0-1).

DS_SET_START 写入 1 启动 DS 模块。(WO)

Register 19.3. DS SET ME REG (0x0E04)

[illegible]

DS_SET_ME 写入 1 以开始计算。(WO)

Register 19.4. DS SET FINISH REG (0x0E08)

[illegible]

DS_SET_FINISH 写入 1 以结束运算。(WO)

Register 19.5. DS_QUERY_BUSY_REG (0x0E0C)

(reserved)																																		DS_QUERY_BUSY																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																															1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DS_QUERY_BUSY 1: DS 模块正在忙; 0: DS 模块空闲。(RO)

Register 19.6. DS_QUERY_KEY_WRONG_REG (0x0E10)

(reserved)																															DS_QUERY_KEY_WRONG					
31																														4	3	0	Reset			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0	

DS_QUERY_KEY_WRONG 1-15: HMAC 被调用, 但 DS 模块未拿到 *DS_KEY* (最大值为 15);
0: HMAC 未被调用。(RO)

Register 19.7. DS_QUERY_CHECK_REG (0x0E14)

(reserved)																															DS_PADDING_BAD DS_MD_ERROR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31																														2	1	0	Reset																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DS_PADDING_BAD 1: 填充校验失败; 0: 填充校验通过。(RO)

DS_MD_ERROR 1: MD 校验失败; 0: MD 校验通过。(RO)

Register 19.8. DS_DATE_REG (0x0E20)

(reserved)																															DS_DATE								
31	30																																		0	Reset			
0	0																																		0x20200618				

DS_DATE 版本控制寄存器。(R/W)

20 时钟毛刺检测

20.1 概述

为提升 ESP32-C3 的安全性能，防止攻击者通过给外部晶振 XTAL_CLK 附加毛刺，使芯片进入异常状态，从而实施对芯片的攻击，ESP32-C3 搭载了毛刺检测模块 (CLK Glitch_Detect)，用于检测从外部晶振输入的 XTAL_CLK 是否携带毛刺，并在检测到毛刺后，产生数字系统复位信号，复位包括 RTC 在内的整个数字电路。

20.2 功能描述

20.2.1 时钟毛刺检测

ESP32-C3 的毛刺检测模块将对输入芯片的 XTAL_CLK 时钟信号进行检测，当时钟的脉宽 (a 或 b) 小于 3 ns 时，将认为检测到毛刺，触发毛刺检测信号，屏蔽输入的 XTAL_CLK 时钟信号。

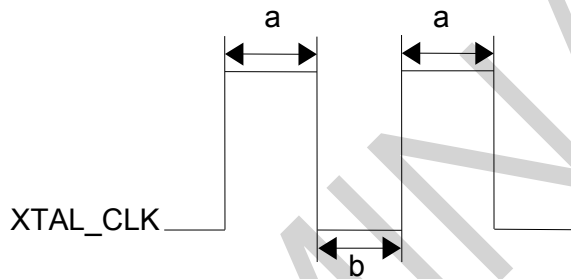


图 20-1. XTAL_CLK 脉宽

20.2.2 复位

当时钟毛刺检测电路检测到 XTAL_CLK 上有影响电路正常工作的毛刺之后，如果 `RTC_CNTL_GLITCH_RST_EN` 位为 1，将触发系统级复位。该位默认为开启复位状态。

21 随机数发生器 (RNG)

21.1 概述

ESP32-C3 内置一个真随机数发生器，其生成的 32 位随机数可作为加密等操作的基础。

21.2 主要特性

ESP32-C3 的随机数发生器可通过物理过程而非算法生成真随机数，所有生成的随机数在特定范围内出现的概率完全一样。

21.3 功能描述

系统可以从随机数发生器的寄存器 `RNG_DATA_REG` 中读取随机数，每个读到的 32 位随机数都是真随机数，噪声源为系统中的**热噪声**和**异步时钟**。

- **热噪声**可以来自 SAR ADC 或高速 ADC 或两者兼有。当芯片的 SAR ADC 或高速 ADC 工作时，就会产生比特流，并通过异或 (XOR) 逻辑运算作为随机数种子进入随机数生成器。
- `RTC20M_CLK` 是一种**异步时钟源**，会产生电路亚稳态。这种亚稳态也可以作为随机数种子²，进入随机数生成器。

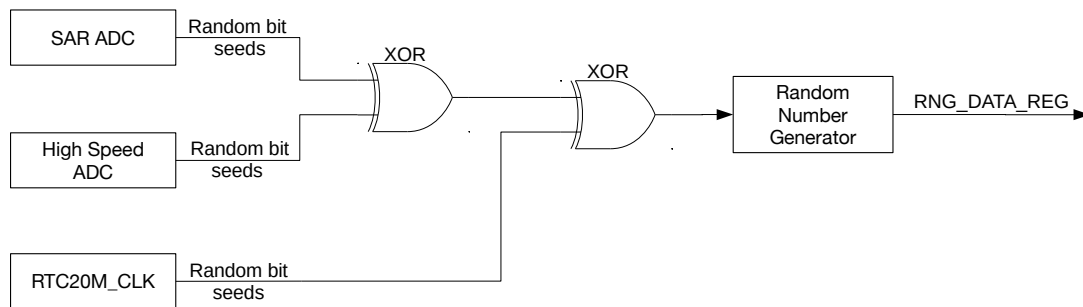


图 21-1. 噪声源

当 SAR ADC 打开时，每个 `RTC20M_CLK` (20 MHz) 时钟周期内（来自内部 RC 振荡器，详见 6 复位和时钟 章节），随机数发生器将获得 2 位的熵。因此，为了获得最大的熵值，建议读取 `RNG_DATA_REG` 寄存器时的速率不超过 1 MHz。

当高速 ADC 打开时，每个 APB 时钟周期（通常为 80 MHz）内，随机数发生器将获得 2 位的熵。因此，为了获得最大的熵值，建议读取 `RNG_DATA_REG` 寄存器时的速率不超过 5 MHz。

我们在仅打开高速 ADC 的状态下，以 5 MHz 的速率从 `RNG_DATA_REG` 读取了 2 GB 的数据样本，并使用 Dieharder 随机数测试套件（版本 3.31.1）对样本进行了测试。最终，样本通过了所有测试。

21.4 编程指南

在使用 ESP32-C3 的随机数生成器时，应该至少保证 SAR ADC 或高速 ADC¹ 或 `RTC20M_CLK` 处于使能状态²，否则可能会导致产生伪随机数，应注意避免。其中，

- SAR ADC 受控于 DIG ADC 控制器。详见 [28 片上传感器与模拟信号处理](#) 章节。
- 高速 ADC 在 Wi-Fi 或蓝牙开启时自动打开。
- RTC20M_CLK 可通过设置 [RTC_CNTL_CLK_CONF_REG](#) 寄存器中的 [RTC_CNTL_DIG_CLK20M_EN](#) 位使能。

说明:

1. 注意，在 Wi-Fi 开启时，极端情况下高速 ADC 有读值饱和的可能，这会降低熵值。因此，建议在 Wi-Fi 开启时，同时通过 DIG ADC1 控制器打开 SAR ADC 产生随机数。
2. RTC20M_CLK 时钟仅可以提高随机数发生器的熵值。然而，为了保证随机数发生器可以获得足够大的熵值，仍建议在使用随机数发生器时至少保证 SAR ADC 或高速 ADC 处于工作状态。

在使用随机数生成器时，请多次读取 [RNG_DATA_REG](#) 寄存器的值，直至获得足够多的随机数。在读取寄存器时，注意控制速率不要超过上方第 [21.3](#) 小节介绍。

21.5 寄存器列表

请注意，下表中的地址都是相对于随机数发生器基地址的地址偏移量（相对地址），详见章节 [3 系统和存储器](#) 中的表 [3-4](#)。

名称	描述	地址	访问
RNG_DATA_REG	随机数数据	0x00B0	只读

21.6 寄存器

请注意，这里的地址都是相对于随机数发生器基地址的地址偏移量（相对地址），相见章节 [3 系统和存储器](#) 中的表 [3-4](#)。

Register 21.1. RNG_DATA_REG (0x00B0)

31	0
0x00000000	
Reset	

RNG_DATA 随机数来源。(只读)

22 UART 控制器 (UART)

22.1 概述

嵌入式应用通常要求一个简单的并且占用系统资源少的方法来传输数据。通用异步收发传输器 (UART) 即可以满足这些要求，它能够灵活地与外部设备进行全双工数据交换。芯片中有两个 UART 控制器可供使用，并且兼容不同的 UART 设备。另外，UART 还可以用作红外数据交换 (IrDA) 或 RS485 调制解调器。

两个 UART 控制器分别有一组功能相同的寄存器。本文以 UART n 指代两个 UART 控制器， n 为 0、1。

UART 是一种以字符为导向的通用数据链，可以实现设备间的通信。异步传输的意思是不需要在发送数据上添加时钟信息。这也要求发送端和接收端的速率、停止位、奇偶校验位等都要相同，通信才能成功。

一个典型的 UART 帧开始于一个起始位，紧接着是有效数据，然后是奇偶校验位（可有可无），最后是停止位。芯片上的 UART 控制器支持多种字符长度和停止位。另外，控制器还支持软硬件流控和 GDMA，可以实现无缝高速的数据传输。开发者可以使用多个 UART 端口，同时又能保证很少的软件开销。

22.2 主要特性

UART 控制器具有如下特性：

- 支持三个可预分频的时钟源
- 可编程收发波特率
- 两个 UART 的发送 FIFO 以及接收 FIFO 共享 512 x 8-bit RAM
- 全双工异步通信
- 支持输入信号波特率自检功能
- 支持 5/6/7/8 位数据长度
- 支持 1/1.5/2/3 个停止位
- 支持奇偶校验位
- 支持 AT_CMD 特殊字符检测
- 支持 RS485 协议
- 支持 IrDA 协议
- 支持 GDMA 高速数据通信
- 支持 UART 唤醒模式
- 支持软件流控和硬件流控

22.3 UART 架构

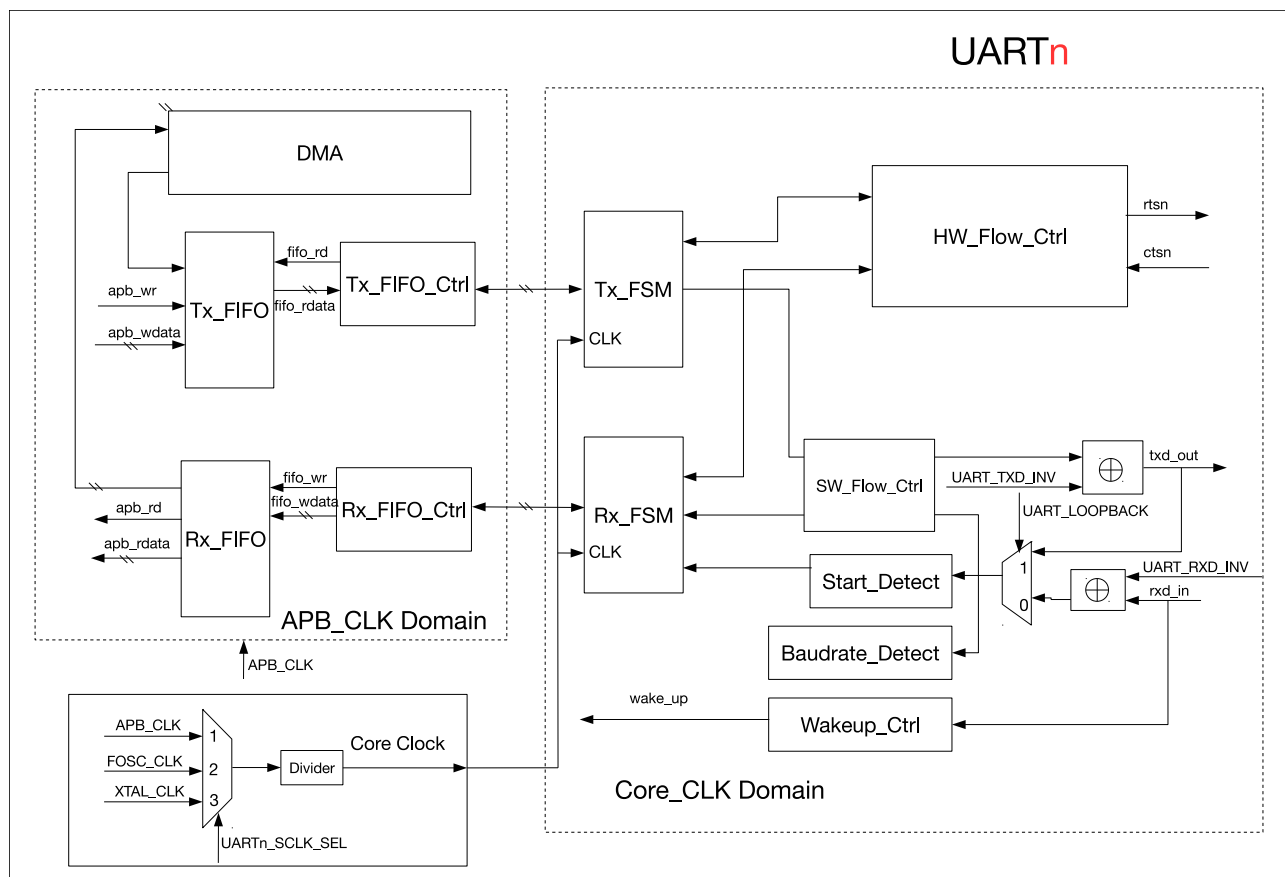


图 22-1. UART 基本架构图

图 22-1 为 UART 基本架构图。UART 模块工作在两个时钟域：APB_CLK 时钟域和 Core 时钟域。UART Core 有三个时钟源：80-MHz APB_CLK、FOSC_CLK 以及晶振时钟 XTAL_CLK（详情请参考章节 6 复位和时钟）。可以通过配置 `UART_SCLK_SEL` 来选择时钟源。分频器用于对时钟源进行分频，然后产生时钟信号来驱动 UART Core 模块。`UART_CLKDIV_REG` 将分频系数分成两个部分：`UART_CLKDIV` 用于配置整数部分，`UART_CLKDIV_FRAG` 用于配置小数部分。

UART 控制器可以分为两个功能块：发送块和接收块。

发送块包含一个发送 FIFO 用于缓存待发送的数据。软件可以通过 APB 总线向 Tx_FIFO 写数据，也可以通过 GDMA 将数据搬入 Tx_FIFO。Tx_FIFO_Ctrl 用于控制 Tx_FIFO 的读写过程，当 Tx_FIFO 非空时，Tx_FSM 通过 Tx_FIFO_Ctrl 读取数据，并将数据按照配置的帧格式转化成比特流。比特流输出信号 txd_out 可以通过配置 `UART_TXD_INV` 寄存器实现取反功能。

接收块包含一个接收 FIFO 用于缓存待处理的数据。输入比特流 rxd_in 可以输入到 UART 控制器。可以通过 `UART_RXD_INV` 寄存器实现取反。Baudrate_Detect 通过检测最小比特流输入信号的脉宽来测量输入信号的波特率。Start_Detect 用于检测数据的 START 位，当检测到 START 位之后，Rx_FSM 通过 Rx_FIFO_Ctrl 将帧解析后的数据存入 Rx_FIFO 中。软件可以通过 APB 总线读取 Rx_FIFO 中的数据也可以使用 GDMA 进行数据接收。

HW_Flow_Ctrl 通过标准 UART RTS 和 CTS (rtsn_out 和 ctsn_in) 流控信号来控制 rxd_in 和 txd_out 的数据流。SW_Flow_Ctrl 通过在发送数据流中插入特殊字符以及在接收数据流中检测特殊字符来进行数据流的控制。当 UART 处于 Light-sleep 状态（详情请参考章节 4 低功耗管理 (RTC_CNTL) [to be added later]）时，Wakeup_Ctrl

开始计算 rxd_in 的上升沿个数，当上升沿个数大于 (`UART_ACTIVE_THRESHOLD + 2`) 时产生 wake_up 信号给 RTC 模块，由 RTC 来唤醒芯片。

22.4 功能描述

22.4.1 时钟与复位

UART 为异步外设。其寄存器配置模块与 TX/RX FIFO 工作在 APB_CLK 时钟域，而控制 UART 发送与接收的 Core 模块工作在 UART Core 时钟域。UART Core 有三个时钟源：APB_CLK，FOSC_CLK 以及晶振时钟 XTAL_CLK，可通过配置 `UART_SCLK_SEL` 字段来选择时钟源。选择后的时钟源通过预分频器分频后进入 UART Core 模块。该预分频器支持小数分频，`UART_SCLK_DIV_NUM` 字段为整数部分，`UART_SCLK_DIV_B` 字段为小数部分的分子，`UART_SCLK_DIV_A` 为小数部分的分母。支持的分频范围为：1 ~ 256。

在 UART 波特率满足需求的情况下，通过预分频使 UART Core 模块工作在较小的时钟频率，从而减小 UART 外设的功耗。通常情况下，UART Core 模块时钟小于 APB_CLK 时钟，并且在满足 UART 波特率的情况下，UART Core 时钟分频系数可以配置到最大值。UART 也支持 UART Core 模块时钟大于 APB_CLK 时钟，此时，UART Core 模块时钟最大为 APB_CLK 的 3 倍。另外，UART TX/RX 的 Core 时钟可以被单独控制。置位 `UART_TX_SCLK_EN` 使能 UART TX 的 Core 时钟；置位 `UART_RX_SCLK_EN` 使能 UART RX 的 Core 时钟。

为确保配置寄存器的值成功从 APB_CLK 时钟域同步到 UART Core 时钟域，寄存器配置需要遵循一定的流程，详情请参考章节 22.5。

对整个 UART 的复位，需要遵循如下配置流程：

- 将 `SYSTEM_UART_MEM_CLK_EN` 置 1 打开 UART RAM 时钟；
- 将 `SYSTEM_UARTn_CLK_EN` 置 1 打开 UART_n APB_CLK；
- 将 `SYSTEM_UARTn_RST` 位清 0；
- 向寄存器 `UART_RST_CORE` 写 1；
- 向寄存器 `SYSTEM_UARTn_RST` 写 1；
- 将寄存器 `SYSTEM_UARTn_RST` 清 0；
- 向寄存器 `UART_RST_CORE` 清 0。

注：不推荐单独复位 UART APB 模块或者 UART Core 模块。

22.4.2 UART RAM

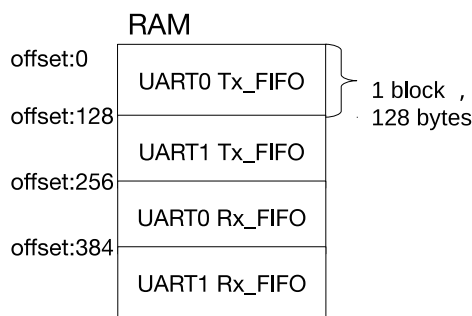


图 22-2. UART 共享 RAM 图

芯片中两个 UART 控制器共用 512x8-bit RAM 空间。如图 22-2 所示，RAM 以 block 为单位进行分配，1 block 为 128x8 bits。图 22-2 所示为默认情况下两个 UART 控制器的 Tx_FIFO 和 Rx_FIFO 占用 RAM 的情况。通过配置 `UART_TX_SIZE` 可以对 UART n 的 Tx_FIFO 进行扩展，通过配置 `UART_RX_SIZE` 可以对 UART n 的 Rx_FIFO 进行扩展，UART0 Tx_FIFO 可以从地址 0 扩展到整个 RAM 空间，UART1 Tx_FIFO 可以从地址 128 扩展到 RAM 的尾地址，UART0 Rx_FIFO 可以从地址 256 扩展到 RAM 的尾地址，UART1 Rx_FIFO 则不支持地址空间扩展，以 1 block 为单位进行扩展。需要注意的是所有 UART 的 FIFO 起始地址是固定的，因此前一个 UART 的 FIFO 空间向后扩展会占用后面 UART 的 FIFO 空间。比如，设置 UART0 的 `UART_TX_SIZE` 为 2，则 UART0 Tx_FIFO 的地址从 0 扩展到 255。这时，UART1 Tx_FIFO 的默认空间被占用，这时将不能使用 UART1 发送器功能。

当两个 UART 控制器都不工作时，可以通过置位 `UART_MEM_FORCE_PD` 来使 RAM 进入低功耗状态。

UART0 和 UART1 的 Tx_FIFO 可以通过置位 `UART_TXFIFO_RST` 来复位，UART0 和 UART1 的 Rx_FIFO 可以通过置位 `UART_RXFIFO_RST` 来复位。

对于 TX FIFO，可以通过 APB 总线或 GDMA 向其写入数据，硬件 Tx_FSM 自动从其中读取数据，数据将按照配置的帧格式转换成比特流；对于 RX FIFO，可以通过 APB 总线或 GDMA 读取其中的数据，并存储到内存，硬件 Rx_FSM 将接收到的比特流转换成字节并写入 RX FIFO。两个 UART 共享同一个 GDMA 通道。

配置 `UART_TXFIFO_EMPTY_THRHD` 可以设置 Tx_FIFO 空信号阈值，当存储在 Tx_FIFO 中的数据量小于 `UART_TXFIFO_EMPTY_THRHD` 时会产生中断 `UART_TXFIFO_EMPTY_INT`；配置 `UART_RXFIFO_FULL_THRHD` 可以设置 Rx_FIFO 满信号阈值，当储存在 Rx_FIFO 中的数据量大于 `UART_RXFIFO_FULL_THRHD` 会产生中断 `UART_RXFIFO_FULL_INT`。另外，当 Rx_FIFO 中储存的数据量超过其能存储的最大值时，会产生 `UART_RXFIFO_OVF_INT` 中断。

UART n 可以通过寄存器 `UART_FIFO_REG` 访问 FIFO。您可以写 `UART_RXFIFO_RD_BYTE` 将数据存入 TX FIFO，也可以读 `UART_RXFIFO_RD_BYTE` 获取 RX FIFO 中的数据。

22.4.3 波特率产生与检测

22.4.3.1 波特率产生

在 UART 发送或接收数据之前，需要配置寄存器来设置波特率。波特率发生器主要通过对输入时钟源的分频来实现，支持小数分频。`UART_CLKDIV_REG` 将分频系数分成两个部分：`UART_CLKDIV` 用于配置整数部分，`UART_CLKDIV_FRAG` 用于配置小数部分。在输入时钟为 80 MHz 的情况下，UART 能支持的最大波特率为 5 MBaud。

波特率分频器系数由 $\text{UART_CLKDIV} + (\text{UART_CLKDIV_FRAG}/16)$ 构成。也就是说，最终波特率为 $\text{INPUT_FREQ} / (\text{UART_CLKDIV} + (\text{UART_CLKDIV_FRAG}/16))$ 。例如，若 `UART_CLKDIV` = 694，`UART_CLKDIV_FRAG` = 7，则分频系数为 $(694 + 7/16) = 694.4375$ 。需要注意的是 INPUT_FREQ 为 UART Core 时钟。

`UART_CLKDIV_FRAG` 为 0 时，分频器为整数分频，每 `UART_CLKDIV` 个输入脉冲都会产生一个输出脉冲。

`UART_CLKDIV_FRAG` 不为 0 时，分频器为小数分频，输出波特率脉冲不完全统一。如图 22-3 所示，每 16 个输出脉冲，波特率发生器分频 $(\text{UART_CLKDIV} + 1)$ 个输入脉冲或 `UART_CLKDIV` 个输入脉冲。分频 $(\text{UART_CLKDIV} + 1)$ 个输入脉冲产生 `UART_CLKDIV_FRAG` 个输出脉冲，分频 `UART_CLKDIV` 个输入脉冲产生剩余的 $(16 - \text{UART_CLKDIV_FRAG})$ 个输出脉冲。

如图 22-3 所示，输出脉冲相互交错，使得输出时序更加统一。

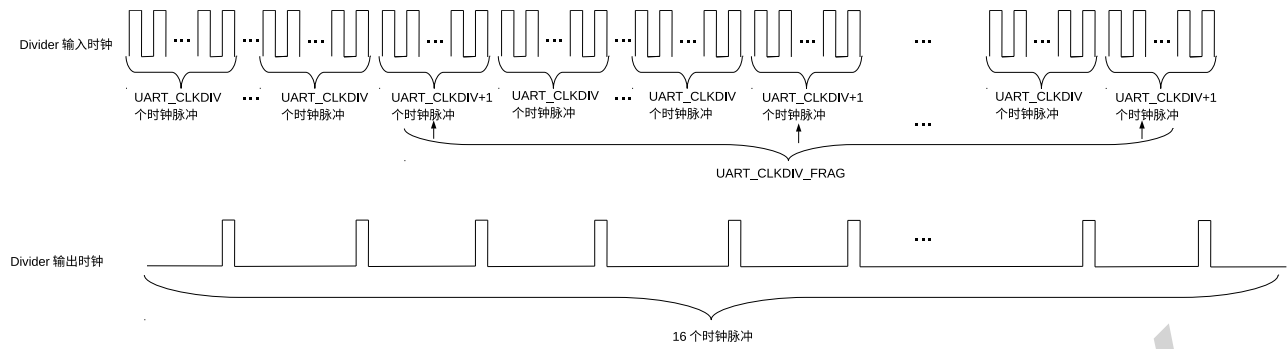


图 22-3. UART 控制器分频

为了支持 IrDA（详情见章节 22.4.6），IrDA 小数分频器会产生 $16 \times \text{UART_CLKDIV_REG}$ 分频的时钟用于 IrDA 数据传输。产生 IrDA 数据传输时钟的小数分频器原理与上述小数分频器一样，取 $\text{UART_CLKDIV}/16$ 作为分频值的整数部分，取 UART_CLKDIV 的低 4 比特作为小数部分。

22.4.3.2 波特率检测

置位 `UART_AUTOBAUD_EN` 可以开启 UART 波特率自检测功能。图 22-1 中的 `Baudrate_Detect` 可以滤除信号脉宽小于 `UART_GLITCH_FILT` 的噪声。

在 UART 双方进行通信之前可以通过发送几个随机数据让具有波特率检测功能的数据接收方进行波特率分析。`UART_LOWPULSE_MIN_CNT` 存储了最小低电平脉冲宽度，`UART_HIGHPULSE_MIN_CNT` 存储了最小高电平脉冲宽度，`UART_POSEDGE_MIN_CNT` 存储了两个上升沿之间的最小脉冲宽度，`UART_NEGEDGE_MIN_CNT` 存储了两个下降沿之间最小的脉冲宽度。软件可以通过读取这四个寄存器获取发送方的波特率。

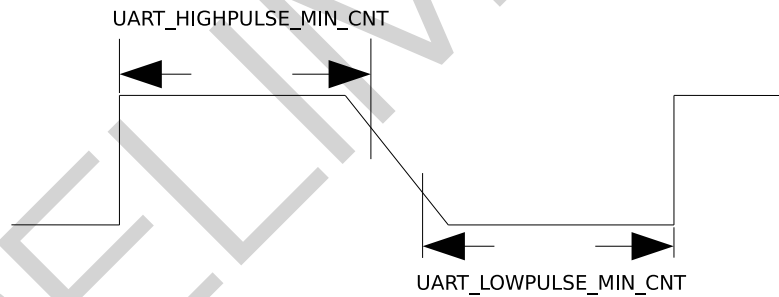


图 22-4. UART 信号下降沿较差时序图

波特率的计算分为三种情况：

1. 正常情况下，为防止因亚稳态在上升沿或下降沿附近采样数据错误而导致 `UART_LOWPULSE_MIN_CNT` 或者 `UART_HIGHPULSE_MIN_CNT` 不准确，单比特脉冲宽度可以通过将这两个值相加取平均消除误差。计算公式如下：

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{(\text{UART_LOWPULSE_MIN_CNT} + \text{UART_HIGHPULSE_MIN_CNT} + 2)/2}$$

2. 对于 UART 信号的下降沿信号比较差的情况，如图22-4所示，这时通过取 `UART_LOWPULSE_MIN_CNT` 与 `UART_HIGHPULSE_MIN_CNT` 的和平均得到的值不准确，可以通过 `UART_POSEDGE_MIN_CNT` 获取发送方波特率。计算公式如下：

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{(\text{UART_POSEDGE_MIN_CNT} + 1)/2}$$

3. 对于 UART 信号的上升沿信号比较差的情况，可以通过 `UART_NEGEDGE_MIN_CNT` 获取发送方波特率。计算公式如下：

$$B_{\text{uart}} = \frac{f_{\text{clk}}}{(\text{UART_NEGEDGE_MIN_CNT} + 1)/2}$$

22.4.4 UART 数据帧

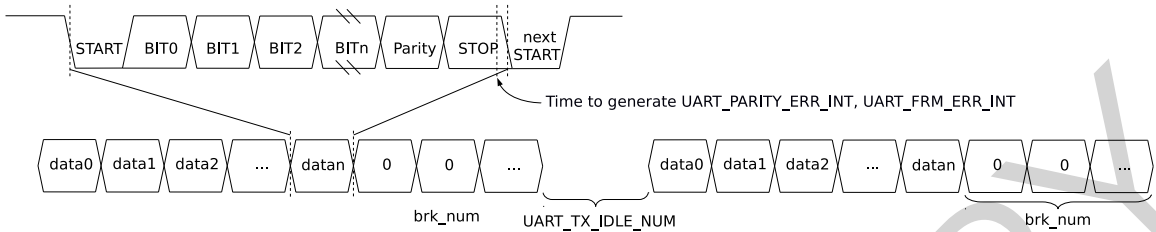


图 22-5. UART 数据帧结构

图 22-5 所示为基本数据帧格式，数据帧从 START 位开始以 STOP 位结束。START 占用 1 bit，STOP 位可以通过配置 `UART_STOP_BIT_NUM`、`UART_DL1_EN` 和 `UART_DLO_EN` 实现 1、1.5、2、3 位宽。START 为低电平，STOP 为高电平。

数据位宽 (BIT0 ~ BITn) 为 5 ~ 8 bit，可以通过 `UART_BIT_NUM` 进行配置。当置位 `UART_PARITY_EN` 时，数据帧会在数据之后添加一位奇偶校验位。`UART_PARITY` 用于选择奇校验或是偶校验。当接收器检测到输入数据的校验位错误时会产生 `UART_PARITY_ERR_INT` 中断，输入数据仍会存入 `Rx_FIFO`。当接收器检测到数据数据帧格式错误时会产生 `UART_FRM_ERR_INT` 中断，默认情况下，输入数据会被存入 `Rx_FIFO`。

`Tx_FIFO` 中数据都发送完成后会产生 `UART_TX_DONE_INT` 中断。置位 `UART_TXD_BRK` 时，`Tx_FIFO` 中数据发送完成后发送端会继续发送几个连续的特殊数据帧 NULL，在 NULL 数据帧，TX 数据线输出为低电平。NULL 数据帧的数量可由 `UART_TX_BRK_NUM` 进行配置。发送器发送完所有的 NULL 数据帧之后会产生 `UART_TX_BRK_DONE_INT` 中断。数据帧之间可以通过配置 `UART_TX_IDLE_NUM` 保持最小间隔时间。当一帧数据之后的空闲时间大于等于 `UART_TX_IDLE_NUM` 寄存器的配置值时则产生 `UART_TX_BRK_IDLE_DONE_INT` 中断。

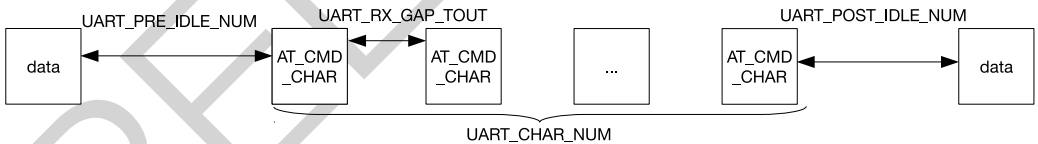


图 22-6. AT_CMD 字符格式

图 22-6 为一种特殊的 AT_CMD 字符格式。当接收器连续收到 AT_CMD_CHAR 字符且字符之间满足如下条件时将会产生 `UART_AT_CMD_CHAR_DET_INT` 中断。

- 接收到的第一个 AT_CMD_CHAR 与上一个非 AT_CMD_CHAR 之间间隔至少 `UART_PRE_IDLE_NUM` 个波特率周期。
- AT_CMD_CHAR 字符之间间隔小于 `UART_RX_GAP_TOUT` 个波特率周期。
- 接收的 AT_CMD_CHAR 字符个数必须大于等于 `UART_CHAR_NUM`。
- 接收到的最后一个 AT_CMD_CHAR 字符与下一个非 AT_CMD_CHAR 之间间隔至少 `UART_POST_IDLE_NUM` 个波特率周期。

22.4.5 RS485

UART 支持 RS485 协议, RS485 因使用差分信号传输数据, 相比于 RS232 具有更远的传输距离及更高的传输速率。RS485 有两线半双工及四线全双工模式, UART 模块采用两线半双工模式, 并支持侦听总线的功能。RS485 两线 multidrop 模式, 最大可支持 32 个 slave。

22.4.5.1 驱动控制

如图 22-7 所示, RS485 两线 multidrop 系统中, 需要一个外部 RS485 传输器实现单端信号与差分信号的转换。RS485 传输器包括一个驱动器与一个接收器。当 UART 不作为发送器时, 通过关闭驱动器来断开与差分传输线的连接。DE 为 1 时, 使能驱动器; DE 为 0 关闭驱动器。

UART 接收端通过外部接收器将差分信号转为单端信号。RE 作为接收器的使能控制信号, RE 为 0, 使能接收器; RE 为 1, 关闭接收器。如果 RE 被配置为 0, 从而允许 UART 保持侦听总线上的数据, 包括 UART 发送的数据。

DE 信号的控制分为软件控制和硬件控制两种方法。为减少软件的开销, DE 信号采用硬件来控制。图 22-7 所示, DE 与 UART 的 dtrn_out 相连 (详见 22.4.8.1 小节)。

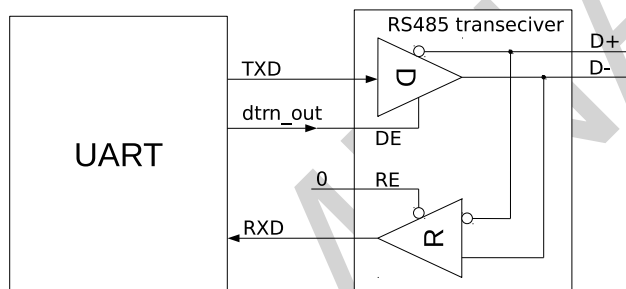


图 22-7. RS485 模式驱动控制结构图

22.4.5.2 转换延时

默认情况下, UART 处于接收状态。当从发送转为接收状态时, 为保证发送数据被稳定接收, RS485 协议推荐在发送停止位之后增加一个波特率的转换延时。UART 发送模块支持在 Start 位之前或在停止位之后增加一个波特率的延时。置位 `UART_DLO_EN`, 在 Start 位之前增加一个波特率周期延时; 置位 `UART_DL1_EN`, 在停止位之后增加一个波特率周期延时。

22.4.5.3 总线侦听

RS485 两线 multidrop 系统中, 当外部 RS485 传输器的 RE 被配置为 0 时, UART 支持侦听总线。默认情况下, 不允许 UART 在发送数据时接收数据。置位 `UART_RS485TX_RX_EN`, 允许在发送数据时接收数据, 配合外部 RS485 传输器的配置, UART 保持侦听传输总线。另外, 默认情况下, 不允许 UART 在接收数据时发送数据。置位 `UART_RS485RXBY_TX_EN`, 允许在接收数据时发送数据。

UART 支持侦听 UART 发送的数据。UART 处于发送状态下, 当侦听到 UART 发送的数据与 UART 接收的数据不同时, 触发 `UART_RS485_CLASH_INT` 中断; 侦听到发送的数据帧错误时, 触发 `UART_RS485_FRM_ERR_INT` 中断; 侦听到发送数据极性错误时, 触发 `UART_RS485_PARITY_ERR_INT` 中断。

22.4.6 IrDA

IrDA 数据协议由物理层、链路接入层和链路管理层三个基本层协议组成。UART 实现了其物理层协议。在 IrDA 编码模式下，支持最大信号速率到 115.2 Kbit/s，即 SIR 模式。如图 22-8 所示，IrDA 编码器将来自 UART 的非归零编码 (NRZ) 信号采用反向归零编码 (RZI) 并输出给外部驱动和红外 LED，用 3/16 Bit Time 的脉宽调制信号表示逻辑“0”，用低电平表示逻辑“1”。IrDA 解码器接收来自红外接收器的信号并输出为 UART 的 NRZ 编码。一般情况下，接收端信号空闲时为高电平，编码器输出极性与解码器输入极性相反。当检测到低脉冲表示接收到开始信号。

IrDA 使能时，一个比特被划分为 16 个时钟周期，在其第 9、10、11 个时钟周期中，当需要发送的比特为 0 时，IrDA 输出为高。

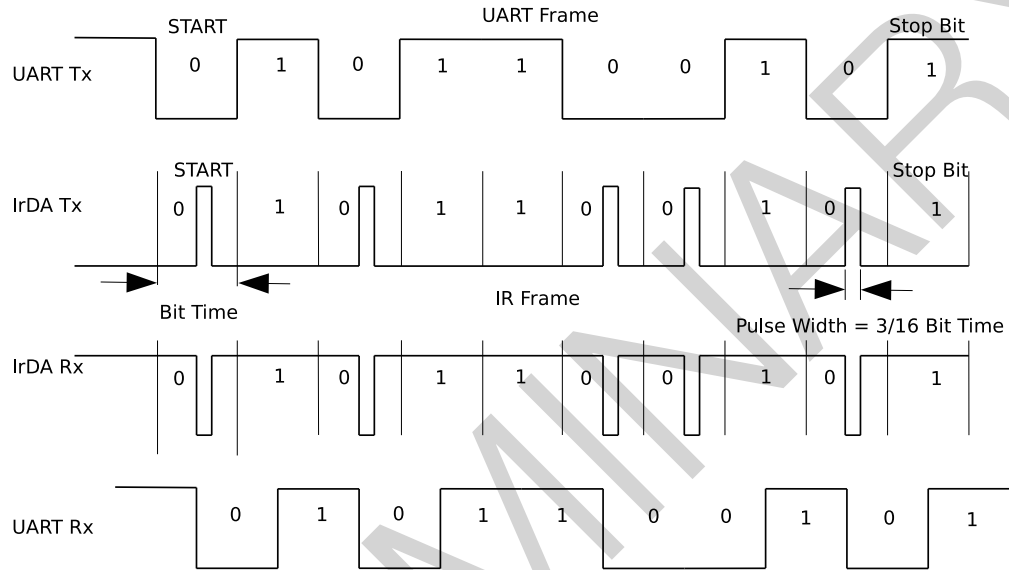


图 22-8. SIR 模式编解码时序图

IrDA 为半双工传输协议，不允许同时进行收发。如图22-9所示，置位 `UART_IRDA_EN` 使能 IrDA 功能。置位 `UART_IRDA_TX_EN`（拉高）使能 IrDA 发送数据，这时不允许 IrDA 接收数据；复位 `UART_IRDA_TX_EN`（拉低）使能 IrDA 接收数据，这时不允许 IrDA 发送数据。

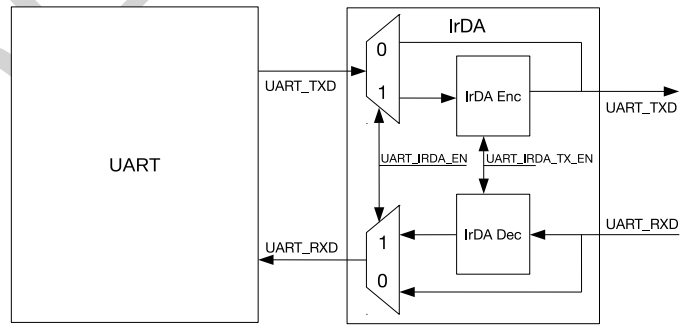


图 22-9. IrDA 编解码结构图

22.4.7 唤醒

UART0 和 UART1 支持唤醒功能。当 UART 处于 Light-sleep 状态时，Wakeup_Ctrl 开始计算 rxd_in 的上升沿个数，当上升沿个数大于 (`UART_ACTIVE_THRESHOLD` + 2) 时产生 wake_up 信号给 RTC 模块，由 RTC 来唤醒

芯片。

22.4.8 流控

UART 控制器有两种数据流控方式：硬件流控和软件流控。硬件流控主要通过输出信号 `rtsn_out` 以及输入信号 `dsmn_in` 进行数据流控制。软件流控主要通过发送数据流中插入特殊字符以及在接收数据流中检测特殊字符来实现数据流控功能。

22.4.8.1 硬件流控

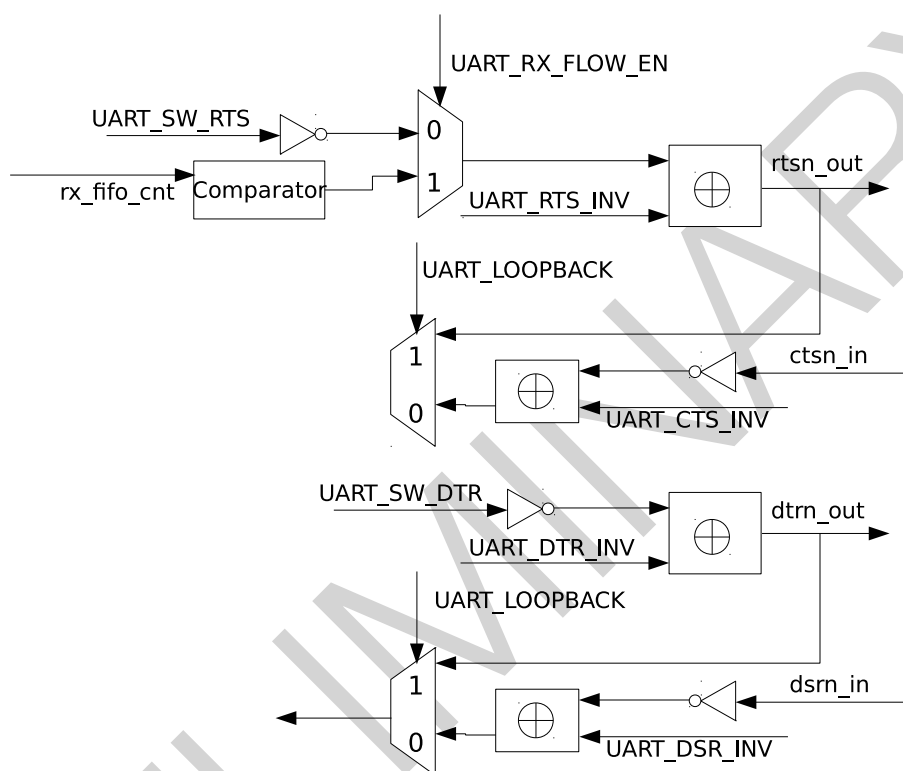


图 22-10. 硬件流控图

图 22-10 为 UART 硬件流控图。硬件流控的控制信号为输出信号 `rtsn_out` 及输入信号 `ctsn_in`。图 22-11 为两个 UART 之间硬件流控信号连接图。记 ESP32-C3 UART 为 IU0，External UART 为 EU0，下文将使用这两个标记来区分两个 UART。输出信号 `rtsn_out` (IU0) 为低电平表示允许对方 (EU0) 发送数据，`rtsn_out` (IU0) 为高电平表示通知对方 (EU0) 中止数据发送直到 `rtsn_out` (IU0) 恢复低电平。`rtsn_out` 输出信号的控制有两种方式。

- 软件控制：将 `UART_RX_FLOW_EN` 置 0 进入该模式。该模式下通过软件配置 `UART_SW_RTS` 改变 `rtsn_out` 的电平。
- 硬件控制：将 `UART_RX_FLOW_EN` 置 1 进入该模式。该模式下硬件会当 Rx_FIFO 中的数据大于 `UART_RX_FLOW_THRHD` 时拉高 `rtsn_out` 的电平。

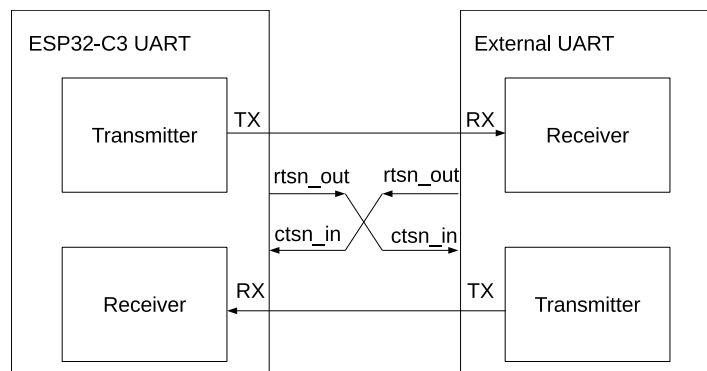


图 22-11. 硬件流控信号连接图

输入信号 `ctsn_in` (IU0) 为低电平表示允许发送端 (IU0) 发送数据；`ctsn_in` (IU0) 为高电平表示禁止发送端 (IU0) 发送数据。当 UART 检测到输入信号 `ctsn_in` (IU0) 的沿变化时会产生 `UART_CTS_CHG_INT` 中断。

UART 发送设备 (IU0) 输出信号 `dtrn_out` 为高电平表示发送数据已经准备完毕, 处于可用状态。`dtrn_out` 通过配置寄存器 `UART_SW_DTR` 产生。UART 接收设备 (IU0) 在检测到输入信号 `dstrn_in` 的沿变化时会产生 `UART_DSR_CHG_INT` 中断。软件在检测到中断后, 通过读取 `UART_DSRN` 可以获取 `dstrn_in` 的输入信号电平, `UART_DSRN` 为高电平时, 表示对方设备 (EU0) 处于可用状态。

对于 RS485 两线 multidrop 系统, 使用 `dtrn_out` 来收发转换。置位 `UART_RS485_EN` 使能 RS485 功能, `dtrn_out` 由硬件产生。数据开始发送时, `dtrn_out` 拉高, 使能外部驱动器; 数据最后一位发送完成后, `dtrn_out` 拉低, 关闭外部驱动器。注意, 当使能停止位之后增加一个波特率延时, `dtrn_out` 会在延时结束后才拉低。

置位 `UART_LOOPBACK` 即开启 UART 的回环测试功能。此时 UART 的输出信号 `txd_out` 和其输入信号 `rx_d_in` 相连, `rtsn_out` 和 `ctsn_in` 相连, `dtrn_out` 和 `dstrn_out` 相连。当接收的数据与发送的数据相同时表明 UART 能够正常发送和接收数据。

22.4.8.2 软件流控

软件流控不使用硬件的 CTS/RTS 控制线, 而是在发送数据流中嵌入 XON/XOFF 字符来通知对方是否可以使用数据发送来实现流控。将 `UART_SW_FLOW_CON_EN` 置 1 使能软件流控。

在使用软件流控后, 硬件会自动检测接收数据流中是否有 XON/XOFF 字符, 在检测到相应的字符后会产生 `UART_SW_XOFF_INT` 或 `UART_SW_XON_INT` 中断。在检测到接收数据流中有 XOFF 字符后, 发送器将会在发送完当前数据后停止发送; 在检测到接收数据流中有 XON 字符后, 将会使能发送器发送数据。另外, 软件可以通过置位 `UART_FORCE_XOFF` 来强制发送器停止发送数据, 发送器会在发送完当前字节后停止发送; 也可以通过置位 `UART_FORCE_XON` 来使能发送器发送数据。

软件可以根据 `Rx_FIFO` 中剩余空间大小决定流控字符的发送。置位 `UART_SEND_XOFF`, 发送器会在发送完当前数据之后插入一个 XOFF 字符, 该字符通过寄存器 `UART_XOFF_CHAR` 配置; 置位 `UART_SEND_XON`, 发送器会在发送完当前数据之后插入一个 XON 字符, 该字符通过寄存器 `UART_XON_CHAR` 配置。另外, 当 UART 接收 FIFO 中的数据量超过 `UART_XOFF_THRESHOLD` 时, 硬件会置位 `UART_SEND_XOFF`, UART 发送器会在发送完当前数据之后插入一个 XOFF 字符, 该字符通过寄存器 `UART_XOFF_CHAR` 配置。当 UART 接收 FIFO 中的数据量小于 `UART_XON_THRESHOLD` 时, 硬件会置位 `UART_SEND_XON`, UART 发送器会在发送完当前数据之后插入一个 XON 字符, 该字符通过寄存器 `UART_XON_CHAR` 配置。

22.4.9 GDMA 模式

ESP32-C3 中的两个 UART 接口通过通用主机控制器接口 (UHCI) 共用 1 组 GDMA TX/RX 通道。在 GDMA 模式下, 支持对 HCI 协议数据包的解析 (decoder) 及数据包封装 (encoder)。`UHCI_UART n _CE` 字段用于选择哪个串口占用 GDMA 通道。

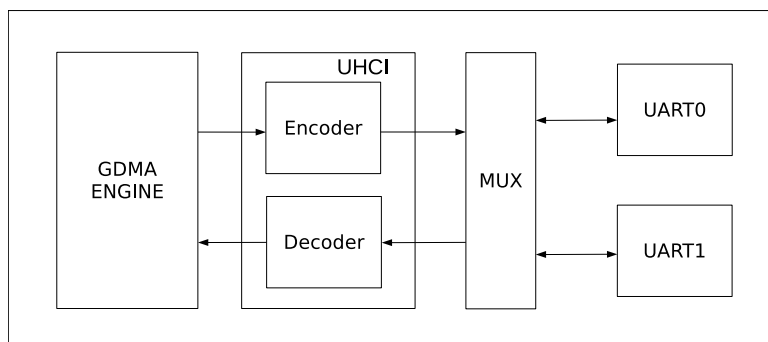


图 22-12. GDMA 模式数据传输

图 22-12 为 GDMA 方式数据传输图。在 GDMA Rx 通道接收数据前, 软件将接收链表准备好。`GDMA_INLINK_ADDR_CH n` 用于指向第一个接收链表描述符。置位 `GDMA_INLINK_START_CH n` 之后, 通用主机控制器接口 (UHCI) 会将 UART 接收到的数据传送给 Decoder。经过 Decoder 解析之后的数据在 GDMA 通道的控制下存入接收链表指定的 RAM 空间。

在 GDMA Tx 通道发送数据前, 软件需要将发送链表和发送数据准备好, `GDMA_OUTLINK_ADDR_CH n` 用于指向第一个发送链表描述符。置位 `GDMA_OUTLINK_START_CH n` 之后, GDMA 引擎即从链表中指定的 RAM 地址读取数据, 并通过 Encoder 进行数据包封装, 然后经 UART 的发送模块串行发送出去。

HCI 的数据包格式为 (分隔符 + 数据 + 分隔符)。Encoder 用于在数据前后加上分隔符, 并将数据中和分隔符一样的数据用特殊字符替换。Decoder 用于去除数据包前后分隔符, 并将数据中的特殊字符进行替换为分隔符。数据前后的分隔符可以有连续多个。分隔符可由 `UHCI_SEPER_CHAR` 进行配置, 默认值为 0xC0。数据中与分隔符一样的数据可以用 `UHCI_ESC_SEQ0_CHAR0` (默认为 0xDB) 和 `UHCI_ESC_SEQ0_CHAR1` (默认为 0xDD) 进行替换。当数据全部发送完成后, 会产生 `GDMA_OUT_TOTAL_EOF_CH n _INT` 中断。当数据接收完成后, 会产生 `GDMA_IN_SUC_EOF_CH n _INT` 中断。

22.4.10 UART 中断

- `UART_AT_CMD_CHAR_DET_INT`: 当接收器检测到 AT_CMD 字符时触发此中断。
- `UART_RS485_CLASH_INT`: 在 RS485 模式下检测到发送器和接收器之间的冲突时触发此中断。
- `UART_RS485_FRM_ERR_INT`: 在 RS485 模式下检测到发送块发送的数据帧错误时触发此中断。
- `UART_RS485_PARITY_ERR_INT`: 在 RS485 模式下检测到发送块发送的数据校验位错误时触发此中断。
- `UART_TX_DONE_INT`: 当发送器发送完 FIFO 中的所有数据时触发此中断。
- `UART_TX_BRK_IDLE_DONE_INT`: 当发送器在最后一个数据发送后保持了最短的间隔时间时触发此中断。
- `UART_TX_BRK_DONE_INT`: 当发送 FIFO 中的数据发送完之后发送器完成了发送 NULL 则触发此中断。
- `UART_GLITCH_DET_INT`: 当接收器在起始位的中点处检测到毛刺时触发此中断。
- `UART_SW_XOFF_INT`: `UART_SW_FLOW_CON_EN` 置位时, 当接收器接收到 XOFF 字符时触发此中断。
- `UART_SW_XON_INT`: `UART_SW_FLOW_CON_EN` 置位时, 当接收器接收到 XON 字符时触发此中断。

- UART_RXFIFO_TOUT_INT: 当接收器接收一个字节的的时间大于 [UART_RX_TOUT_THRHD](#) 时触发此中断。
- UART_BRK_DET_INT: 当接收器在停止位之后检测到 NULL 时触发此中断。
- UART_CTS_CHG_INT: 当接收器检测到 CTSn 信号的沿变化时触发此中断。
- UART_DSR_CHG_INT: 当接收器检测到 DSRn 信号的沿变化时触发此中断。
- UART_RXFIFO_OVF_INT: 当接收器接收到的数据量多于 FIFO 的存储量时触发此中断。
- UART_FRM_ERR_INT: 当接收器检测到数据帧错误时触发此中断。
- UART_PARITY_ERR_INT: 当接收器检测到校验位错误时触发此中断。
- UART_TXFIFO_EMPTY_INT: 当发送 FIFO 中的数据量少于 [UART_TXFIFO_EMPTY_THRHD](#) 所指定的值时触发此中断。
- UART_RXFIFO_FULL_INT: 当接收器接收到的数据多于 [UART_RXFIFO_FULL_THRHD](#) 所指定的值时触发此中断。
- UART_WAKEUP_INT: UART 被唤醒时产生此中断。

22.4.11 UCHI 中断

- UHCI_APP_CTRL1_INT: 软件置位 [UHCI_APP_CTRL1_INT_RAW](#) 时触发此中断。
- UHCI_APP_CTRL0_INT: 软件置位 [UHCI_APP_CTRL0_INT_RAW](#) 时触发此中断。
- UHCI_OUTLINK_EOF_ERR_INT: 当检测到发送链表描述符中的 EOF 有错误时触发此中断。
- UHCI_SEND_A_REG_Q_INT: 当使用 `always_send` 发送一串短包, UHCI 发送了短包后触发此中断。
- UHCI_SEND_S_REG_Q_INT: 当使用 `single_send` 发送一串短包, UHCI 发送了短包后触发此中断。
- UHCI_TX_HUNG_INT: 当 UHCI 利用 GDMA Tx 通道从 RAM 中读取数据的时间过长时触发此中断。
- UHCI_RX_HUNG_INT: 当 UHCI 利用 GDMA Rx 通道接收数据的时间过长时触发此中断。
- UHCI_TX_START_INT: 当检测到分隔符时触发此中断。
- UHCI_RX_START_INT: 当分隔符已发送时触发此中断。

22.5 编程流程

22.5.1 寄存器类型

UART 的所有寄存器都处于 APB_CLK 时钟域。对于软件可配置的寄存器, 根据其作用的时钟域及同步处理, 将其分为三类: 立即寄存器, 同步寄存器及静态寄存器。立即寄存器作用于 APB_CLK 时钟域, 通过 APB 总线配置后立即生效; 同步寄存器作用于 Core 时钟域, 这些寄存器需要经过同步之后才能生效; 静态寄存器也作用于 Core 时钟域, 但这些寄存器不会在 UART 工作过程中动态修改。静态寄存器没有同步处理, 软件可以通过开关 UART TX/RX Core 时钟的方式保证 UART Core 时钟域采样到正确的配置信息。

22.5.1.1 同步寄存器

为了确保作用于 UART Core 时钟域的寄存器被正确采样, 他们中大多数都做了跨时钟域处理, 这部分即为同步寄存器。同步寄存器如表22-1所示。对这些寄存器的配置流程如下:

- 将 [UART_UPDATE_CTRL](#) 清 0 使能寄存器同步功能;

- 等待 `UART_REG_UPDATE` 为 0，确保上一次同步已经完成；
- 配置同步寄存器；
- 向 `UART_REG_UPDATE` 写 1，将配置的值同步到 Core 时钟域。

表 22-1. UART_n同步寄存器

寄存器	域名
UART_CLKDIV_REG	UART_CLKDIV_FRAG[3:0]
	UART_CLKDIV[11:0]
UART_CONF0_REG	UART_AUTOBAUD_EN
	UART_ERR_WR_MASK
	UART_TXD_INV
	UART_RXD_INV
	UART_IRDA_EN
	UART_TX_FLOW_EN
	UART_LOOPBACK
	UART_IRDA_RX_INV
	UART_IRDA_TX_EN
	UART_IRDA_WCTL
	UART_IRDA_TX_EN
	UART_IRDA_DPLX
	UART_STOP_BIT_NUM
	UART_BIT_NUM
	UART_PARITY_EN
	UART_PARITY
UART_FLOW_CONF_REG	UART_SEND_XOFF
	UART_SEND_XON
	UART_FORCE_XOFF
	UART_FORCE_XON
	UART_XONOFF_DEL
	UART_SW_FLOW_CON_EN
UART_TXBRK_CONF_REG	UART_RS485_TX_DLY_NUM[3:0]
	UART_RS485_RX_DLY_NUM
	UART_RS485RXBY_TX_EN
	UART_RS485TX_RX_EN
	UART_DL1_EN
	UART_DL0_EN
	UART_RS485_EN

22.5.1.2 静态寄存器

在作用于 UART Core 时钟域的寄存器中，有一部分寄存器不会在 UART 工作过程中动态修改，被认为是静态的，称为静态寄存器。静态寄存器没有做跨时钟域处理。静态寄存器的配置一定是 UART TX/RX 停止工作阶段，因此可以通过关闭 UART TX/RX 时钟的方式，保证配置寄存器的亚稳态不会被采样到。当打开 UART TX/RX 时钟打开时，软件配置的值已经稳定，从而确保配置的值被正确采样。表 22-2 列出了这些寄存器。对这些寄存器的配置流程如下：

- 根据当前停止工作的模块为 UART TX 还是 RX，将 `UART_TX_SCLK_EN` 或 `UART_RX_SCLK_EN` 清 0 关闭 UART Tx 或 RX 时钟；
- 配置静态寄存器；
- 向 `UART_TX_SCLK_EN` 或 `UART_RX_SCLK_EN` 写 1 打开 UART Tx 或 RX 时钟。

表 22-2. UART 静态寄存器

寄存器	域名
<code>UART_RX_FILT_REG</code>	<code>UART_GLITCH_FILT_EN</code>
	<code>UART_GLITCH_FILT[7:0]</code>
<code>UART_SLEEP_CONF_REG</code>	<code>UART_ACTIVE_THRESHOLD[9:0]</code>
<code>UART_SWFC_CONF0_REG</code>	<code>UART_XOFF_CHAR[7:0]</code>
<code>UART_SWFC_CONF1_REG</code>	<code>UART_XON_CHAR[7:0]</code>
<code>UART_IDLE_CONF_REG</code>	<code>UART_TX_IDLE_NUM[9:0]</code>
<code>UART_AT_CMD_PRECNT_REG</code>	<code>UART_PRE_IDLE_NUM[15:0]</code>
<code>UART_AT_CMD_POSTCNT_REG</code>	<code>UART_POST_IDLE_NUM[15:0]</code>
<code>UART_AT_CMD_GAPTOUT_REG</code>	<code>UART_RX_GAP_TOUT[15:0]</code>
<code>UART_AT_CMD_CHAR_REG</code>	<code>UART_CHAR_NUM[7:0]</code>
	<code>UART_AT_CMD_CHAR[7:0]</code>

22.5.1.3 立即寄存器

除表22-1与22-2 外的所有软件可配置寄存器作用于 APB_CLK 时钟域，即为立即寄存器，例如，中断及 FIFO 配置寄存器等。

22.5.2 具体步骤

图22-13 显示了 UART 模块的编程流程。主要包括：初始化、寄存器配置、启动 UART TX/RX 和数据传输结束。

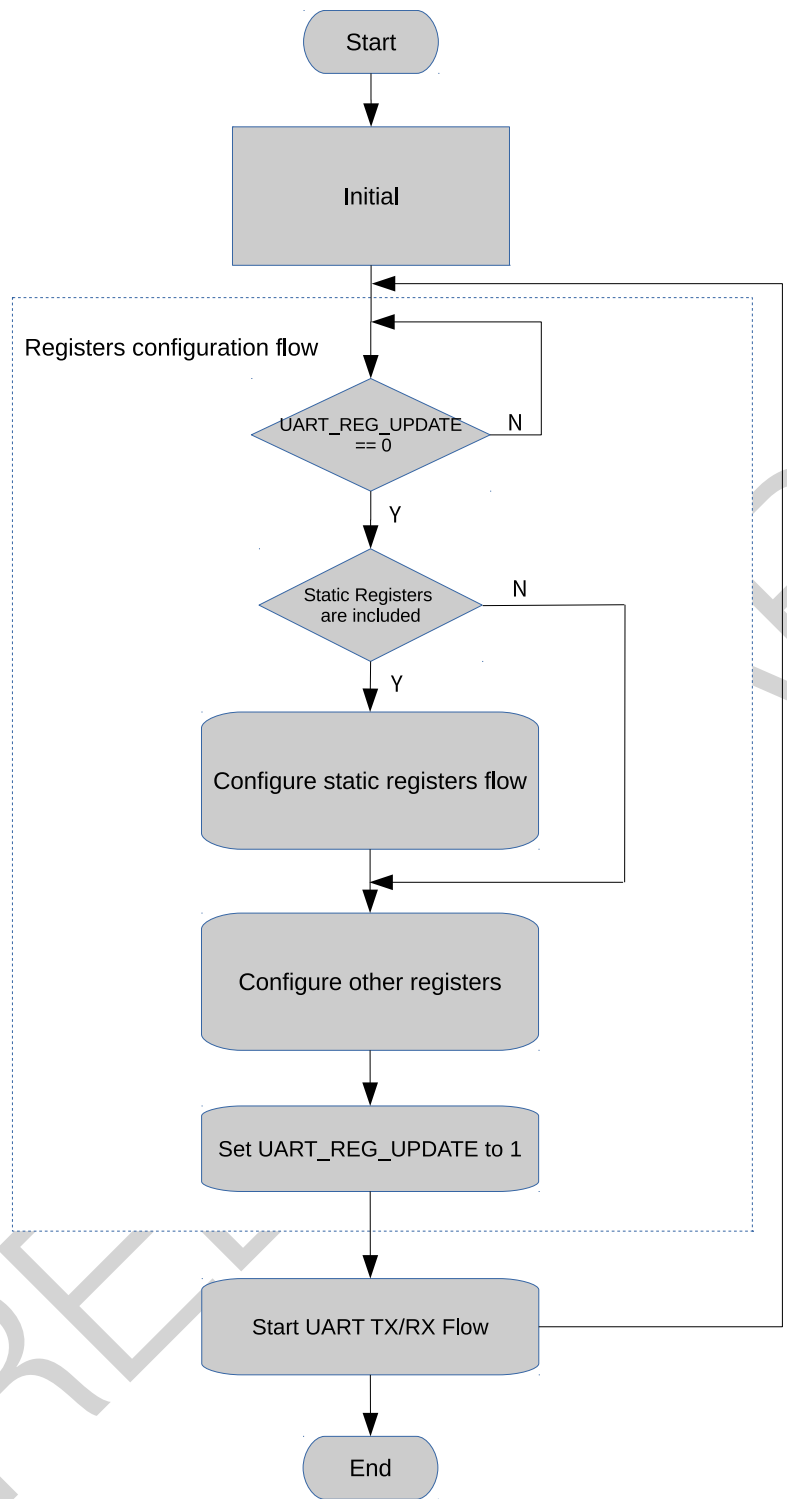


图 22-13. UART 编程流程

22.5.2.1 URAT_n 模块初始化

URAT_n 模块初始化流程如下：

- 将 `SYSTEM_UART_MEM_CLK_EN` 置 1 打开 UART RAM 时钟；
- 将 `SYSTEM_UARTn_CLK_EN` 置 1 打开 UART_n APB_CLK；
- 将寄存器 `SYSTEM_UARTn_RST` 清 0；

- 向寄存器 `UART_RST_CORE` 写 1；
- 向寄存器 `SYSTEM_UARTn_RST` 写 1；
- 将寄存器 `SYSTEM_UARTn_RST` 清 0；
- 将寄存器 `UART_RST_CORE` 清 0；
- 将 `UART_UPDATE_CTRL` 清 0 使能寄存器同步功能。

22.5.2.2 URAT_n 通信配置

URAT_n 通信配置流程如下：

- 等待 `UART_REG_UPDATE` 为 0，确保上一次同步已经完成；
- 如果配置寄存器中包含静态寄存器，配置流程参考 22.5.1.2 完成配置；
- 配置 `UART_SCLK_SEL` 选择时钟源；
- 配置 `UART_SCLK_DIV_NUM`、`UART_SCLK_DIV_A`、`UART_SCLK_DIV_B` 设置预分频器系数；
- 配置 `UART_CLKDIV`、`UART_CLKDIV_FRAG` 设置发送波特率；
- 配置 `UART_BIT_NUM` 设置数据长度；
- 配置 `UART_PARITY_EN`、`UART_PARITY` 设置奇偶校验；
- 可选步骤，根据应用不同存在差异...
- 向 `UART_REG_UPDATE` 写 1，将配置的值同步到 Core 时钟域。

22.5.2.3 启动 URAT_n

启动 UART_n TX 发送数据：

- 配置 `UART_TXFIFO_EMPTY_THRHD`，设置 TXFIFO 空阈值；
- 对 `UART_TXFIFO_EMPTY_INT_ENA` 置 0，关闭 `UART_TXFIFO_EMPTY_INT` 中断；
- 向 `UART_RXFIFO_RD_BYTE` 写入需要发送的数据；
- 置位 `UART_TXFIFO_EMPTY_INT_CLR`，清除 `UART_TXFIFO_EMPTY_INT` 中断；
- 置位 `UART_TXFIFO_EMPTY_INT_ENA`，使能 `UART_TXFIFO_EMPTY_INT` 中断；
- 检测 `UART_TXFIFO_EMPTY_INT`，等待发送数据结束。

启动 UART_n RX 数据接收：

- 配置 `UART_RXFIFO_FULL_THRHD`，设置 RXFIFO 满阈值；
- 置位 `UART_RXFIFO_FULL_INT_ENA`，使能 `UART_RXFIFO_FULL_INT` 中断；
- 检测 `UART_TXFIFO_FULL_INT`，等待 RXFIFO 接收数据满；
- 通过读 `UART_RXFIFO_RD_BYTE`，从 RXFIFO 中读出数据，并可通过 `UART_RXFIFO_CNT` 获得当前 RXFIFO 中的接收数据量。

22.6 寄存器列表

本小节的所有地址均为相对于 UART 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
FIFO 配置			
UART_FIFO_REG	FIFO 数据寄存器	0x0000	RO
UART_MEM_CONF_REG	UART 阈值和分配配置	0x0060	R/W
中断寄存器			
UART_INT_RAW_REG	原始中断状态	0x0004	R/WTC/SS
UART_INT_ST_REG	屏蔽中断状态	0x0008	RO
UART_INT_ENA_REG	中断使能位	0x000C	R/W
UART_INT_CLR_REG	中断清除位	0x0010	WT
配置寄存器			
UART_CLKDIV_REG	时钟分频配置	0x0014	R/W
UART_RX_FILT_REG	RX 滤波器配置	0x0018	R/W
UART_CONF0_REG	配置寄存器 0	0x0020	R/W
UART_CONF1_REG	配置寄存器 1	0x0024	R/W
UART_FLOW_CONF_REG	软件流控配置	0x0034	varies
UART_SLEEP_CONF_REG	睡眠模式配置	0x0038	R/W
UART_SWFC_CONF0_REG	软件流控字符配置	0x003C	R/W
UART_SWFC_CONF1_REG	软件流控字符配置	0x0040	R/W
UART_TXBRK_CONF_REG	帧结束空闲配置	0x0044	R/W
UART_IDLE_CONF_REG	帧结束空闲配置	0x0048	R/W
UART_RS485_CONF_REG	RS485 模式配置	0x004C	R/W
UART_CLK_CONF_REG	UART core 时钟配置	0x0078	R/W
状态寄存器			
UART_STATUS_REG	UART 状态寄存器	0x001C	RO
UART_MEM_TX_STATUS_REG	TX FIFO 写入、读取偏移地址	0x0064	RO
UART_MEM_RX_STATUS_REG	RX FIFO 写入、读取偏移地址	0x0068	RO
UART_FSM_STATUS_REG	UART 发送和接收状态	0x006C	RO
自动波特率检测寄存器			
UART_LOWPULSE_REG	自动波特率检测最短低电平脉冲持续时间寄存器	0x0028	RO
UART_HIGHPULSE_REG	自动波特率检测最短高电平脉冲持续时间寄存器	0x002C	RO
UART_RXD_CNT_REG	自动波特率检测沿变化计数寄存器	0x0030	RO
UART_POSPULSE_REG	自动波特率检测高电平脉冲寄存器	0x0070	RO
UART_NEGPULSE_REG	自动波特率检测低电平脉冲寄存器	0x0074	RO
AT 转义序列检测配置			
UART_AT_CMD_PRECNT_REG	序列发送前的时序配置	0x0050	R/W
UART_AT_CMD_POSTCNT_REG	序列发送后的时序配置	0x0054	R/W
UART_AT_CMD_GAPTOOUT_REG	超时配置	0x0058	R/W
UART_AT_CMD_CHAR_REG	AT 转义序列检测配置	0x005C	R/W

名称	描述	地址	访问
版本寄存器			
UART_DATE_REG	UART 版本控制寄存器	0x007C	R/W
UART_ID_REG	UART ID 寄存器	0x0080	varies

名称	描述	地址	访问
配置寄存器			
UHCI_CONF0_REG	UHCI 配置寄存器	0x0000	R/W
UHCI_CONF1_REG	UHCI 配置寄存器	0x0014	varies
UHCI_ESCAPE_CONF_REG	转义符配置	0x0020	R/W
UHCI_HUNG_CONF_REG	超时配置	0x0024	R/W
UHCI_ACK_NUM_REG	配置 UHCI ACK 值	0x0028	varies
UHCI_QUICK_SENT_REG	UHCI 快速发送配置寄存器	0x0030	varies
UHCI_REG_Q0_WORD0_REG	Q0_WORD0 快速发送寄存器	0x0034	R/W
UHCI_REG_Q0_WORD1_REG	Q0_WORD1 快速发送寄存器	0x0038	R/W
UHCI_REG_Q1_WORD0_REG	Q1_WORD0 快速发送寄存器	0x003C	R/W
UHCI_REG_Q1_WORD1_REG	Q1_WORD1 快速发送寄存器	0x0040	R/W
UHCI_REG_Q2_WORD0_REG	Q2_WORD0 快速发送寄存器	0x0044	R/W
UHCI_REG_Q2_WORD1_REG	Q2_WORD1 快速发送寄存器	0x0048	R/W
UHCI_REG_Q3_WORD0_REG	Q3_WORD0 快速发送寄存器	0x004C	R/W
UHCI_REG_Q3_WORD1_REG	Q3_WORD1 快速发送寄存器	0x0050	R/W
UHCI_REG_Q4_WORD0_REG	Q4_WORD0 快速发送寄存器	0x0054	R/W
UHCI_REG_Q4_WORD1_REG	Q4_WORD1 快速发送寄存器	0x0058	R/W
UHCI_REG_Q5_WORD0_REG	Q5_WORD0 快速发送寄存器	0x005C	R/W
UHCI_REG_Q5_WORD1_REG	Q5_WORD1 快速发送寄存器	0x0060	R/W
UHCI_REG_Q6_WORD0_REG	Q6_WORD0 快速发送寄存器	0x0064	R/W
UHCI_REG_Q6_WORD1_REG	Q6_WORD1 快速发送寄存器	0x0068	R/W
UHCI_ESC_CONF0_REG	转义序列配置寄存器 0	0x006C	R/W
UHCI_ESC_CONF1_REG	转义序列配置寄存器 1	0x0070	R/W
UHCI_ESC_CONF2_REG	转义序列配置寄存器 2	0x0074	R/W
UHCI_ESC_CONF3_REG	转义序列配置寄存器 3	0x0078	R/W
UHCI_PKT_THRES_REG	包长度配置寄存器	0x007C	R/W
中断寄存器			
UHCI_INT_RAW_REG	原始中断状态	0x0004	varies
UHCI_INT_ST_REG	屏蔽中断状态	0x0008	RO
UHCI_INT_ENA_REG	中断使能位	0x000C	R/W
UHCI_INT_CLR_REG	中断清除位	0x0010	WT
UHCI 状态寄存器			
UHCI_STATE0_REG	UHCI 接收状态	0x0018	RO
UHCI_STATE1_REG	UHCI 发送状态	0x001C	RO
UHCI_RX_HEAD_REG	UHCI 包报头寄存器	0x002C	RO
版本寄存器			
UHCI_DATE_REG	UHCI 版本控制寄存器	0x0080	R/W

22.7 寄存器

本小节的所有地址均为相对于 UART 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 22.1. UART_FIFO_REG (0x0000)

(reserved)																								UART_RXFIFO_RD_BYTE									
31																								8	7	0							
0 0																								0								Reset	

UART_RXFIFO_RD_BYTE UART_n 通过此寄存器访问 FIFO。(RO)

Register 22.2. UART_MEM_CONF_REG (0x0060)

(reserved)				UART_MEM_FORCE_PU				UART_MEM_FORCE_PD				UART_RX_TOUT_THRHD				UART_RX_FLOW_THRHD				UART_TX_SIZE				UART_RX_SIZE				(reserved)				
31					28	27	26	25					16	15					7	6					4	3					1	0
0	0	0	0	0	0	0	0xa				0x0				0x1				1				0				Reset					

UART_RX_SIZE 配置存储器分配给 RX FIFO 的空间大小。默认为 128 字节。(R/W)

UART_TX_SIZE 配置存储器分配给 TX FIFO 的空间大小。默认为 128 字节。(R/W)

UART_RX_FLOW_THRHD 配置使用硬件流控时接收数据的最大值。(R/W)

UART_RX_TOUT_THRHD 配置接收器接收一个字节所需时间的阈值，单位是比特时间（即传输一个比特所需的时间）。接收器接收一个字节所需时间超过阈值且 UART_RX_TOUT_EN 置 1 时触发 UART_RXFIFO_TOUT_INT 中断。(R/W)

UART_MEM_FORCE_PD 置位此位强制关闭 UART 存储器。(R/W)

UART_MEM_FORCE_PU 置位此位强制开启 UART 存储器。(R/W)

Register 22.3. UART_INT_RAW_REG (0x0004)

(reserved)																															UART_WAKEUP_INT_RAW UART_AT_CMD_CHAR_DET_INT_RAW UART_RS485_CLASH_INT_RAW UART_RS485_FRM_ERR_INT_RAW UART_TX_DONE_INT_RAW UART_TX_BRK_IDLE_DONE_INT_RAW UART_TX_BRK_DET_INT_RAW UART_GLITCH_DET_INT_RAW UART_SW_XON_INT_RAW UART_SW_XOFF_INT_RAW UART_BRK_DET_INT_RAW UART_DSR_CHG_INT_RAW UART_RXFIFO_OVF_INT_RAW UART_FRM_ERR_INT_RAW UART_PARITY_ERR_INT_RAW UART_TXFIFO_EMPTY_INT_RAW UART_RXFIFO_FULL_INT_RAW															
31																				20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0																

UART_RXFIFO_FULL_INT_RAW 接收器接收数据多于 UART_RXFIFO_FULL_THRHD 的值时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_TXFIFO_EMPTY_INT_RAW TX FIFO 中的数据少于 UART_TXFIFO_EMPTY_THRHD 的值时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_PARITY_ERR_INT_RAW 接收器检测到数据奇偶检验位错误时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_FRM_ERR_INT_RAW 接收器检测到数据帧错误时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_RXFIFO_OVF_INT_RAW 接收器接收数据超过 FIFO 的存储容量时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_DSR_CHG_INT_RAW 接收器检测到 DSRn 信号的沿变化时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_CTS_CHG_INT_RAW 接收器检测到 CTSn 信号的沿变化时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_BRK_DET_INT_RAW 接收器在停止位后检测到 0 时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_RXFIFO_TOUT_INT_RAW 接收器接收一个字节所需时间超过 UART_RX_TOUT_THRHD 时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_SW_XON_INT_RAW 接收器接收到 XON 字符且 UART_SW_FLOW_CON_EN 置 1 时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_SW_XOFF_INT_RAW 接收器接收到 XOFF 字符且 UART_SW_FLOW_CON_EN 置 1 时, 该原始中断位翻转至高电平。(R/WTC/SS)

UART_GLITCH_DET_INT_RAW 接收器在起始位的中点处检测到毛刺时, 该原始中断位翻转至高电平。(R/WTC/SS)

见下页...

Register 22.3. UART_INT_RAW_REG (0x0004)

接上页...

UART_TX_BRK_DONE_INT_RAW 发送器在发送完 TX FIFO 中所有数据后完成 NULL 字符的发送时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_TX_BRK_IDLE_DONE_INT_RAW 发送器发送完最后一个数据后的间隔时间达到阈值时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_TX_DONE_INT_RAW 发送器发完 FIFO 中的所有数据后，该原始中断位翻转至高电平。(R/WTC/SS)

UART_RS485_PARITY_ERR_INT_RAW RS485 模式下接收器检测到发送器回音的数据检验位错误时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_RS485_FRM_ERR_INT_RAW RS485 模式下接收器检测到发送器回音的数据帧错误时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_RS485_CLASH_INT_RAW RS485 模式下检测到发送器与接收器冲突时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_AT_CMD_CHAR_DET_INT_RAW 接收器检测到配置的 UART_AT_CMD_CHAR 时，该原始中断位翻转至高电平。(R/WTC/SS)

UART_WAKEUP_INT_RAW 输入 RXD 沿变化次数超过 Light-sleep 模式指定的 UART_ACTIVE_THRESHOLD 值时，该原始中断位翻转至高电平。(R/WTC/SS)

Register 22.4. UART_INT_ST_REG (0x0008)

[illegible]

UART_RXFIFO_FULL_INT_ST UART_RXFIFO_FULL_INT_ENA 置 1 时
UART_RXFIFO_FULL_INT_RAW 的状态位。(RO)

UART_TXFIFO_EMPTY_INT_ST	UART_TXFIFO_EMPTY_INT_ENA	置 1 时
UART_TXFIFO_EMPTY_INT_RAW 的状态位。(RO)		

UART_PARITY_ERR_INT_ST	UART_PARITY_ERR_INT_ENA	置	1	时
UART_PARITY_ERR_INT_RAW 的状态位。(RO)				

UART_FRM_ERR_INT_ST UART_FRM_ERR_INT_ENA 置 1 时 UART_FRM_ERR_INT_RAW 的状态位。(RO)

UART_RXFIFO_OVF_INT_ST	UART_RXFIFO_OVF_INT_ENA	置	1	时
UART_RXFIFO_OVF_INT_RAW 的状态位。(RO)				

UART_DSR_CHG_INT_ST UART_DSR_CHG_INT_ENA 置 1 时 UART_DSR_CHG_INT_RAW 的状态位。(RO)

UART_CTS_CHG_INT_ST UART_CTS_CHG_INT_ENA 置 1 时 UART_CTS_CHG_INT_RAW 的状态位。(RO)

UART_BRK_DET_INT_ST UART_BRK_DET_INT_ENA 置 1 时 UART_BRK_DET_INT_RAW 的状态位。(RO)

UART_RXFIFO_TOUT_INT_ST	UART_RXFIFO_TOUT_INT_ENA	置	1	时
UART RXFIFO TOUT INT RAW 的状态位。(RO)				

UART_SW_XON_INT_ST UART_SW_XON_INT_ENA 置 1 时 UART_SW_XON_INT_RAW 的状态位。
(RO)

UART_SW_XOFF_INT_ST UART_SW_XOFF_INT_ENA 置 1 时 UART_SW_XOFF_INT_RAW 的状态位。(RO)

UART_GLITCH_DET_INT_ST	UART_GLITCH_DET_INT_ENA	置	1	时
UART GLITCH DET INT RAW 的状态位。(RO)				

见下页...

Register 22.4. UART_INT_ST_REG (0x0008)

接上页...

UART_TX_BRK_DONE_INT_ST UART_TX_BRK_DONE_INT_ENA 置 1 时
UART_TX_BRK_DONE_INT_RAW 的状态位。(RO)

UART_TX_BRK_IDLE_DONE_INT_ST UART_TX_BRK_IDLE_DONE_INT_ENA 置 1 时
UART_TX_BRK_IDLE_DONE_INT_RAW 的状态位。(RO)

UART_TX_DONE_INT_ST UART_TX_DONE_INT_ENA 置 1 时 UART_TX_DONE_INT_RAW 的状态
位。(RO)

UART_RS485_PARITY_ERR_INT_ST UART_RS485_PARITY_INT_ENA 置 1 时
UART_RS485_PARITY_ERR_INT_RAW 的状态位。(RO)

UART_RS485_FRM_ERR_INT_ST UART_RS485_FRM_ERR_INT_ENA 置 1 时
UART_RS485_FRM_ERR_INT_RAW 的状态位。(RO)

UART_RS485_CLASH_INT_ST UART_RS485_CLASH_INT_ENA 置 1 时
UART_RS485_CLASH_INT_RAW 的状态位。(RO)

UART_AT_CMD_CHAR_DET_INT_ST UART_AT_CMD_CHAR_DET_INT_ENA 置 1 时
UART_AT_CMD_CHAR_DET_INT_RAW 的状态位。(RO)

UART_WAKEUP_INT_ST UART_WAKEUP_INT_ENA 置 1 时 UART_WAKEUP_INT_RAW 的状态位。
(RO)

Register 22.5. UART_INT_ENA_REG (0x000C)

(reserved)																					UART_WAKEUP_INT_ENA UART_AT_CMD_CHAR_DET_INT_ENA UART_RS485_CLASH_INT_ENA UART_RS485_FRM_ERR_INT_ENA UART_TX_DONE_INT_ENA UART_TX_BRK_ERR_INT_ENA UART_TX_GLITCH_DET_INT_ENA UART_SW_BRK_IDLE_DONE_INT_ENA UART_SW_XOFF_INT_ENA UART_RXFFO_INT_ENA UART_BRK_TOUT_ENA UART_CTS_DET_INT_ENA UART_DSR_CHG_INT_ENA UART_RXFFO_OVF_INT_ENA UART_FRM_ERR_INT_ENA UART_TXFFO_EMPTY_INT_ENA UART_RXFFO_FULL_INT_ENA																			
31																				20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset										

UART_RXFIFO_FULL_INT_ENA UART_RXFIFO_FULL_INT_ST 寄存器的使能位。(R/W)

UART_TXFIFO_EMPTY_INT_ENA UART_TXFIFO_EMPTY_INT_ST 寄存器的使能位。(R/W)

UART_PARITY_ERR_INT_ENA UART_PARITY_ERR_INT_ST 寄存器的使能位。(R/W)

UART_FRM_ERR_INT_ENA UART_FRM_ERR_INT_ST 寄存器的使能位。(R/W)

UART_RXFIFO_OVF_INT_ENA UART_RXFIFO_OVF_INT_ST 寄存器的使能位。(R/W)

UART_DSR_CHG_INT_ENA UART_DSR_CHG_INT_ST 寄存器的使能位。(R/W)

UART_CTS_CHG_INT_ENA UART_CTS_CHG_INT_ST 寄存器的使能位。(R/W)

UART_BRK_DET_INT_ENA UART_BRK_DET_INT_ST 寄存器的使能位。(R/W)

UART_RXFIFO_TOUT_INT_ENA UART_RXFIFO_TOUT_INT_ST 寄存器的使能位。(R/W)

UART_SW_XON_INT_ENA UART_SW_XON_INT_ST 寄存器的使能位。(R/W)

UART_SW_XOFF_INT_ENA UART_SW_XOFF_INT_ST 寄存器的使能位。(R/W)

UART_GLITCH_DET_INT_ENA UART_GLITCH_DET_INT_ST 寄存器的使能位。(R/W)

UART_TX_BRK_DONE_INT_ENA UART_TX_BRK_DONE_INT_ST 寄存器的使能位。(R/W)

UART_TX_BRK_IDLE_DONE_INT_ENA UART_TX_BRK_IDLE_DONE_INT_ST 寄存器的使能位。(R/W)

UART_TX_DONE_INT_ENA UART_TX_DONE_INT_ST 寄存器的使能位。(R/W)

UART_RS485_PARITY_ERR_INT_ENA UART_RS485_PARITY_ERR_INT_ST 寄存器的使能位。(R/W)

UART_RS485_FRM_ERR_INT_ENA UART_RS485_PARITY_ERR_INT_ST 寄存器的使能位。(R/W)

UART_RS485_CLASH_INT_ENA UART_RS485_CLASH_INT_ST 寄存器的使能位。(R/W)

UART_AT_CMD_CHAR_DET_INT_ENA UART_AT_CMD_CHAR_DET_INT_ST 寄存器的使能位。(R/W)

UART_WAKEUP_INT_ENA UART_WAKEUP_INT_ST 寄存器的使能位。(R/W)

Register 22.7. UART_CLKDIV_REG (0x0014)

(reserved)								UART_CLKDIV_FRAG												(reserved)								UART_CLKDIV																																			
31								24								23								20								19								12								11								0							
0 0 0 0 0 0 0 0								0x0								0 0 0 0 0 0 0 0								0x2b6								Reset																															

- UART_CLKDIV 分频系数的整数部分。(R/W)
- UART_CLKDIV_FRAG 分频系数的小数部分。(R/W)

Register 22.8. UART_RX_FILT_REG (0x0018)

(reserved)																								UART_GLITCH_FILT_EN		UART_GLITCH_FILT						
31																								9	8	7	0					
0 0																								0	0x8						Reset	

- UART_GLITCH_FILT 宽度小于该寄存器值的输入脉冲会被忽略。(R/W)
- UART_GLITCH_FILT_EN 置位此位，使能 RX 信号滤波器。(R/W)

Register 22.9. UART_CONF0_REG (0x0020)

<div>(reserved)</div> <div>UART_MEM_CLK_EN</div> <div>UART_AUTOBAUD_EN</div> <div>UART_ERR_WFL_MASK</div> <div>UART_CLK_EN</div> <div>UART_DTR_INV</div> <div>UART_RTS_INV</div> <div>UART_TXD_INV</div> <div>UART_DSR_INV</div> <div>UART_CTS_INV</div> <div>UART_RXD_INV</div> <div>UART_TXFIFO_RST</div> <div>UART_RXFIFO_RST</div> <div>UART_IRDA_EN</div> <div>UART_TX_FLOW_EN</div> <div>UART_LOOPBACK</div> <div>UART_IRDA_RX_INV</div> <div>UART_IRDA_TX_INV</div> <div>UART_IRDA_WCTL</div> <div>UART_IRDA_TX_EN</div> <div>UART_TXD_DPLX</div> <div>UART_SW_BRK</div> <div>UART_SW_DTR</div> <div>UART_STOP_RTS</div> <div>UART_STOP_BIT_NUM</div> <div>UART_BIT_NUM</div> <div>UART_PARITY_EN</div> <div>UART_PARITY</div>																														
31	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
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	Reset	

- UART_PARITY 配置奇偶检验方式。(R/W)
- UART_PARITY_EN 置位此位使能 UART 奇偶检验。(R/W)
- UART_BIT_NUM 设置数据长度。(R/W)
- UART_STOP_BIT_NUM 设置停止位的长度。(R/W)
- UART_SW_RTS 该寄存器用于配置软件流控使用的软件 RTS 信号。(R/W)
- UART_SW_DTR 该寄存器用于配置软件流控使用的软件 DTR 信号。(R/W)
- UART_TXD_BRK 置位此位，使能发送器在发完数据后发送 NULL。(R/W)
- UART_IRDA_DPLX 置位此位开启 IrDA 回环测试模式。(R/W)
- UART_IRDA_TX_EN IrDA 发送器的启动使能位。(R/W)
- UART_IRDA_WCTL 1'h1: IrDA 发送器的第 11 位与第 10 位相同。1'h0: 将 IrDA 发送器的第 11 位置 0。(R/W)
- UART_IRDA_TX_INV 置位此位翻转 IrDA 发送器的电平。(R/W)
- UART_IRDA_RX_INV 置位此位翻转 IrDA 接收器的电平。(R/W)
- UART_LOOPBACK 置位此位开启 UART 回环测试模式。(R/W)
- UART_TX_FLOW_EN 置位此位使能发送器的流控功能。(R/W)
- UART_IRDA_EN 置位此位使能 IrDA 协议。(R/W)
- UART_RXFIFO_RST 置位此位复位 UART RX FIFO。(R/W)
- UART_TXFIFO_RST 置位此位复位 UART TX FIFO。(R/W)
- UART_RXD_INV 置位此位翻转 UART RXD 信号电平。(R/W)
- UART_CTS_INV 置位此位翻转 UART CTS 信号电平。(R/W)
- UART_DSR_INV 置位此位翻转 UART DSR 信号电平。(R/W)
- UART_TXD_INV 置位此位翻转 UART TXD 信号电平。(R/W)
- UART_RTS_INV 置位此位翻转 UART RTS 信号电平。(R/W)
- UART_DTR_INV 置位此位翻转 UART DTR 信号电平。(R/W)

见下页...

Register 22.9. UART_CONF0_REG (0x0020)

接上页...

UART_CLK_EN 1'h1: 强制为寄存器开启时钟。1'h0: 仅在应用写寄存器时支持时钟。(R/W)

UART_ERR_WR_MASK 1'h1: 若数据错误, 接收器不再将数据存入 FIFO。1'h0: 若数据错误, 接收器仍存储。(R/W)

UART_AUTOBAUD_EN 波特率检测的使能信号。(R/W)

UART_MEM_CLK_EN UART 存储器门控使能信号。(R/W)

Register 22.10. UART_CONF1_REG (0x0024)

(reserved)										UART_RX_TOUT_EN UART_RX_FLOW_EN UART_RX_TOUT_FLOW_DIS UART_DIS_RX_DAT_OVF				UART_TXFIFO_EMPTY_THRHD				UART_RXFIFO_FULL_THRHD				
31										22		21	20	19	18	17		9		8	0	
0 0 0 0 0 0 0 0 0 0										0	0	0	0	0x60		0x60		Reset				

UART_RXFIFO_FULL_THRHD 接收器接收数据多于该寄存器的值时产生 UART_RXFIFO_FULL_INT 中断。(R/W)

UART_TXFIFO_EMPTY_THRHD TX FIFO 中的数据少于该寄存器的值时产生 UART_TXFIFO_EMPTY_INT 中断。(R/W)

UART_DIS_RX_DAT_OVF 关闭 UART RX 数据溢出检测。(R/W)

UART_RX_TOUT_FLOW_DIS 使用硬件流控时置位此位停止堆积 idle_cnt。(R/W)

UART_RX_FLOW_EN UART 接收器流控功能的使能位。(R/W)

UART_RX_TOUT_EN UART 接收器超时功能的使能位。(R/W)

Register 22.11. UART_FLOW_CONF_REG (0x0034)

(reserved)																										UART_SEND_XOFF				UART_SEND_XON				UART_FORCE_XOFF				UART_FORCE_XON				UART_XONOFF_DEL				UART_SW_FLOW_CTRL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
31																										6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


- UART_SW_FLOW_CON_EN** 置位此位使能软件流控，与 SW_XON 或 SW_XOFF 寄存器一起使用。
(R/W)
- UART_XONOFF_DEL** 置位此位移除接收数据中的流控字符。(R/W)
- UART_FORCE_XON** 置位此位让发送器继续发送数据。(R/W)
- UART_FORCE_XOFF** 置位此位让发送器停止发送数据。(R/W)
- UART_SEND_XON** 置位此位发送 XON 字符。此位由硬件自动清除。(R/W/SS/SC)
- UART_SEND_XOFF** 置位此位发送 XOFF 字符。此位由硬件自动清除。(R/W/SS/SC)

Register 22.12. UART_SLEEP_CONF_REG (0x0038)

(reserved)																				UART_ACTIVE_THRESHOLD											
31																				10	9										0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0xf0									Reset

- UART_ACTIVE_THRESHOLD** 输入 RXD 沿变化次数超过该寄存器的值时，UART 从 Light-sleep 模式唤醒。(R/W)

Register 22.13. UART_SWFC_CONF0_REG (0x003C)

(reserved)																UART_XOFF_CHAR								UART_XOFF_THRESHOLD																															
31																17								16								9								8								0							
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x13								0xe0								 Reset																							

UART_XOFF_THRESHOLD RX FIFO 中的数据超过该寄存器的值且 UART_SW_FLOW_CON_EN 置 1 时，发送 XOFF 字符。(R/W)

UART_XOFF_CHAR 存储 XOFF 流控字符。(R/W)

Register 22.14. UART_SWFC_CONF1_REG (0x0040)

(reserved)																UART_XON_CHAR				UART_XON_THRESHOLD				
31																17	16	9	8					0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x11				0x0				Reset

UART_XON_THRESHOLD RX FIFO 中的数据小于该寄存器的值且 UART_SW_FLOW_CON_EN 置 1 时，发送 XON 字符。(R/W)

UART_XON_CHAR 存储 XON 流控字符。(R/W)

Register 22.15. UART_TXBRK_CONF_REG (0x0044)

(reserved)																												UART_TX_BRK_NUM															
31																												7								0							
0 0																												0xa								Reset							

UART_TX_BRK_NUM 配置数据发完后待发 NULL 字符的数量。UART_TXD_BRK 置 1 时有意义。(R/W)

Register 22.16. UART_IDLE_CONF_REG (0x0048)

(reserved)												UART_TX_IDLE_NUM										UART_RX_IDLE_THRHD												
31											20	19											10	9										
0 0 0 0 0 0 0 0 0 0 0 0												0x100										0x100										Reset		

UART_RX_IDLE_THRHD 接收器接收一字节数据所需时间超过该寄存器的值时产生帧结束信号, 单位是比特时间 (即传输一个比特所需的时间)。 (R/W)

UART_TX_IDLE_NUM 配置两次数据传输的间隔时间, 单位是比特时间 (即传输一个比特所需的时间)。 (R/W)

Register 22.17. UART_RS485_CONF_REG (0x004C)

(reserved)																		UART_RS485_TX_DLY_NUM										UART_RS485_RX_DLY_NUM																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31											10	9											6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UART_RS485_EN 置位此位选择 RS485 模式。 (R/W)

UART_DL0_EN 置位此位, 延迟停止位 1 位。 (R/W)

UART_DL1_EN 置位此位, 延迟停止位 1 位。 (R/W)

UART_RS485TX_RX_EN 发送器在 RS485 模式下发送数据时, 置位此位使能接收器接收数据。 (R/W)

UART_RS485RXBY_TX_EN 1'h1: RS485 接收器线路繁忙时使能 RS485 发送器发送数据。 (R/W)

UART_RS485_RX_DLY_NUM 延迟接收器的内部数据信号。 (R/W)

UART_RS485_TX_DLY_NUM 延迟发送器的内部数据信号。 (R/W)

Register 22.18. UART_CLK_CONF_REG (0x0078)

(reserved)						UART_RX_SCLK_EN						UART_TX_SCLK_EN						UART_RST_CORE						UART_SCLK_EN						UART_SCLK_SEL						UART_SCLK_DIV_NUM						UART_SCLK_DIV_A						UART_SCLK_DIV_B					
31						26	25	24	23	22	21	20	19						12	11						6	5						0																				
0	0	0	0	0	0	1	1	0	1	3	0x1					0x0					0x0					Reset																											

- UART_SCLK_DIV_B 分频系数的分母。(R/W)
- UART_SCLK_DIV_A 分频系数的分子。(R/W)
- UART_SCLK_DIV_NUM 分频系数的整数部分。(R/W)
- UART_SCLK_SEL 选择 UART 时钟源。1: APB_CLK; 2: FOSC_CLK; 3: XTAL_CLK。(R/W)
- UART_SCLK_EN 置位此位，使能 UART TX/RX 使能。(R/W)
- UART_RST_CORE 向此位先写 1 后写 0，复位 UART TX/RX。(R/W)
- UART_TX_SCLK_EN 置位此位，使能 UART TX 时钟。(R/W)
- UART_RX_SCLK_EN 置位此位，使能 UART RX 时钟。(R/W)

Register 22.19. UART_STATUS_REG (0x001C)

UART_TXD				UART_RTSN				UART_DTRN				(reserved)				UART_TXFIFO_CNT				UART_RXD				UART_CTSN				UART_DSRN				(reserved)				UART_RXFIFO_CNT			
31	30	29	28	26	25											16	15	14	13	12	10	9											0						
1	1	1	0	0	0	0												1	1	0	0	0	0	0										Reset					

- UART_RXFIFO_CNT 存储 RX FIFO 中有效数据的字节数。(RO)
- UART_DSRN 该寄存器表示内部 UART DSR 信号的电平值。(RO)
- UART_CTSN 该寄存器表示内部 UART CTS 信号的电平值。(RO)
- UART_RXD 该寄存器表示内部 UART RXD 信号的电平值。(RO)
- UART_TXFIFO_CNT 存储 TX FIFO 中数据的字节数。(RO)
- UART_DTRN 此位表示内部 UART DTR 信号的电平。(RO)
- UART_RTSN 此位表示内部 UART RTS 信号的电平。(RO)
- UART_TXD 此位表示内部 UART TXD 信号的电平。(RO)

Register 22.20. UART_MEM_TX_STATUS_REG (0x0064)

(reserved)												UART_TX_RADDR												(reserved)		UART_APB_TX_WADDR									
31											21											11	10	9											0
0	0	0	0	0	0	0	0	0	0	0	0	0x0										0	0x0										Reset		

UART_APB_TX_WADDR 在软件通过 APB 总线写 TX FIFO 时存储 TX FIFO 的偏移地址。(RO)

UART_TX_RADDR 在 TX FSM 通过 Tx_FIFO_Ctrl 读取数据时存储 TX FIFO 的偏移地址。(RO)

Register 22.21. UART_MEM_RX_STATUS_REG (0x0068)

(reserved)												UART_RX_WADDR												(reserved)												UART_APB_RX_RADDR																																																																																															
31												21												20												11												10												9												0																																																											
0												0												0												0												0												0												0												0x100												0												0x100												Reset											

UART_APB_RX_RADDR 在软件通过 APB 总线读取 RX FIFO 数据时存储 RX FIFO 的偏移地址。
UART0 为 10'h100, UART1 为 10'h180。(RO)

UART_RX_WADDR 在 Rx_FIFO_Ctrl 写 RX FIFO 时存储 RX FIFO 的偏移地址。(RO)

Register 22.22. UART_FSM_STATUS_REG (0x006C)

(reserved)																								UART_ST_UTX_OUT				UART_ST_URX_OUT															
31																								8				7				4				3				0			
0 0																								0				0				Reset											

UART_ST_URX_OUT 接收器的状态寄存器。(RO)

UART_ST_UTX_OUT 发送器的状态寄存器。(RO)

360

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

360

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

360

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

Register 22.26. UART_POSPULSE_REG (0x0070)

Diagram of the UART_CR register structure:

- Bits 31-12: (reserved)
- Bits 11-0: UART_POSEDGE_MIN_CNT
- Bit 0: Reset

UART_POSEDGE_MIN_CNT 存储两个上升沿之间的最小输入时钟计数值。用于波特率检测。(RO)

Register 22.27. UART_NEGPULSE_REG (0x0074)

Diagram illustrating the structure of the `UART_NEDGE_MIN_CNT` register. The register is 32 bits wide, divided into three sections:

- Reserved:** Bits 31 down to 12 (20 bits).
- 1:** Bit 11 (1 bit).
- UART_NEDGE_MIN_CNT:** Bits 10 down to 0 (11 bits).

The `UART_NEDGE_MIN_CNT` field is shown with the value `0xffff`. The register is labeled `Reset` at the bottom right.

UART_NEGEDGE_MIN_CNT 存储两个下降沿之间的最小输入时钟计数值。用于波特率检测。(RO)

Register 22.28. UART_AT_CMD_PRECNT_REG (0x0050)

(reserved)																UART_PRE_IDLE_NUM															
31																0															
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x901															
																Reset															

UART_PRE_IDLE_NUM 配置接收器接收第一个 AT_CMD 字符前的空闲时间，单位是比特时间（即传输一个比特所需的时间）。(R/W)

362
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

Register 22.32. UART_DATE_REG (0x007C)

UART_DATE	
31	0
0x2008270	
Reset	

UART_DATE 版本控制寄存器。(R/W)

Register 22.33. UART_ID_REG (0x0080)

UART_REG_UPDATE		UART_UPDATE_CTRL		UART_ID	
31	30	29			0
0	1				0x000500
Reset					

UART_ID 配置 UART_ID。(R/W)

UART_UPDATE_CTRL 用于控制同步模式。在向域 UART_REG_UPDATE 写 1 同步配置寄存器至 UART Core 时钟域之前，该域必须配置为 0。(R/W)

UART_REG_UPDATE 软件向该字段写 1，将寄存器值同步到 UART Core 时钟域。该字段在同步完成后由硬件自清。(R/W/SC)

Register 22.34. UHCI_CONF0_REG (0x0000)

(reserved)													UHCI_UART_RX_BRK_EOF_EN UHCI_CLK_EN UHCI_ENCODE_CRC_EN UHCI_LEN_EOF_EN UHCI_UART_IDLE_EOF_EN UHCI_CRC_REC_EN UHCI_HEAD_EN UHCI_SEPER_EN (reserved) UHCI_UART1_CE UHCI_UART0_CE UHCI_RX_RST UHCI_TX_RST														
31													13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	

- UHCI_TX_RST** 向此位先写 1 再写 0 复位解码状态机。(R/W)
- UHCI_RX_RST** 向此位先写 1 再写 0 复位编码状态机。(R/W)
- UHCI_UART0_CE** 置位此位，将 HCI 和 UART0 相连。(R/W)
- UHCI_UART1_CE** 置位此位，将 HCI 和 UART1 相连。(R/W)
- UHCI_SEPER_EN** 置位此位，使用特殊字符分隔数据帧。(R/W)
- UHCI_HEAD_EN** 置位此位，用格式报头编码数据包。(R/W)
- UHCI_CRC_REC_EN** 置位此位，使能 UHCI 接收 16 位 CRC。(R/W)
- UHCI_UART_IDLE_EOF_EN** 若此位置 1，UHCI 在 UART 空闲时停止接收有效载荷。(R/W)
- UHCI_LEN_EOF_EN** 若此位置 1，UHCI 解码器接收字节数达到指定值时停止接收有效载荷数据。
UHCI_HEAD_EN 为 1 时，该值是 UCHI 数据包报头明确的有效负载长度；UHCI_HEAD_EN 为 0 时，该值为配置值。若此位置 0，UHCI 解码器在接收到 0xc0 后停止接收有效载荷数据。(R/W)
- UHCI_ENCODE_CRC_EN** 置位此位，在有效载荷末尾加 16 位 CCITT-CRC 开始数据完整性检测。
(R/W)
- UHCI_CLK_EN** 1'b1：强制为寄存器开启时钟。1'b0：仅在应用写寄存器时支持时钟。(R/W)
- UHCI_UART_RX_BRK_EOF_EN** 若此位置 1，UART 收到 NULL 帧后 UHCI 会停止接收有效载荷。
(R/W)

Register 22.37. UHCI_HUNG_CONF_REG (0x0024)

(reserved)								UHCI_RXFIFO_TIMEOUT_ENA				UHCI_RXFIFO_TIMEOUT_SHIFT				UHCI_RXFIFO_TIMEOUT				UHCI_TXFIFO_TIMEOUT_ENA				UHCI_TXFIFO_TIMEOUT_SHIFT				UHCI_TXFIFO_TIMEOUT			
31								24	23	22	20	19				12	11	10	8	7								0			
0 0 0 0 0 0 0 0								1	0			0x10				1		0		0x10							Reset				

UHCI_TXFIFO_TIMEOUT . 存储超时值。DMA 接收数据超时时产生 UHCI_TX_HUNG_INT 中断。(R/W)

UHCI_TXFIFO_TIMEOUT_SHIFT 用于配置计数最大值。(R/W)

UHCI_TXFIFO_TIMEOUT_ENA TX FIFO 接收数据超时的使能位。(R/W)

UHCI_RXFIFO_TIMEOUT 存储超时值。DMA 读取 RAM 数据超时时产生 UHCI_RX_HUNG_INT 中断。(R/W)

UHCI_RXFIFO_TIMEOUT_SHIFT 用于配置计数最大值。(R/W)

UHCI_RXFIFO_TIMEOUT_ENA DMA 发送数据超时的使能位。(R/W)

Register 22.38. UHCI_ACK_NUM_REG (0x0028)

(reserved)																												UHCL_ACK_NUM_LOAD		UHCL_ACK_NUM	
31																												4	3	2	0
0 0																												1	0x0		Reset

UHCI_ACK_NUM 软件流控中使用的 ACK 值。(R/W)

UHCI_ACK_NUM_LOAD 置位此位，加载 UHCI_ACK_NUM 配置的值。(WT)

Register 22.39. UHCI_QUICK_SENT_REG (0x0030)

(reserved)																												UHCI_ALWAYS_SEND_EN				UHCI_ALWAYS_SEND_NUM				UHCI_SINGLE_SEND_EN				UHCI_SINGLE_SEND_NUM																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
31																												8	7	6	4		3	2	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

- UHCI_SINGLE_SEND_NUM** 设定 single_send 寄存器。(R/W)
- UHCI_SINGLE_SEND_EN** 置位此位使能 single_send 模式发送短包。(R/W/SC)
- UHCI_ALWAYS_SEND_NUM** 设定 always_send 寄存器。(R/W)
- UHCI_ALWAYS_SEND_EN** 置位此位使能 always_send 模式发送短包。(R/W)

Register 22.40. UHCI_REG_Q0_WORD0_REG (0x0034)

UHCI_SEND_Q0_WORD0																
31															0	
0x000000																Reset

UHCI_SEND_Q0_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.41. UHCI_REG_Q0_WORD1_REG (0x0038)

UHCL_SEND_Q0_WORD1																															
31																															0
0x000000																															
Reset																															

UHCI_SEND_Q0_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.42. UHCI_REG_Q1_WORD0_REG (0x003C)

UHCI_SEND_Q1_WORD0	
31	0
0x000000	
Reset	

UHCI_SEND_Q1_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.43. UHCI_REG_Q1_WORD1_REG (0x0040)

UHCI_SEND_Q1_WORD1	
31	0
0x000000	
Reset	

UHCI_SEND_Q1_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.44. UHCI_REG_Q2_WORD0_REG (0x0044)

UHCI_SEND_Q2_WORD0	
31	0
0x000000	
Reset	

UHCI_SEND_Q2_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.45. UHCI_REG_Q2_WORD1_REG (0x0048)

UHCI_SEND_Q2_WORD1	
31	0
0x000000	
Reset	

UHCI_SEND_Q2_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.46. UHCI_REG_Q3_WORD0_REG (0x004C)

UHCI_SEND_Q3_WORD0	
31	0
0x000000	
Reset	

UHCI_SEND_Q3_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.47. UHCI_REG_Q3_WORD1_REG (0x0050)

UHCI_SEND_Q3_WORD1	
31	0
0x000000	
Reset	

UHCI_SEND_Q3_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.48. UHCI_REG_Q4_WORD0_REG (0x0054)

UHCI_SEND_Q4_WORD0	
31	0
0x000000	
Reset	

UHCI_SEND_Q4_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.49. UHCI_REG_Q4_WORD1_REG (0x0058)

UHCI_SEND_Q4_WORD1	
31	0
0x000000	
Reset	

UHCI_SEND_Q4_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.50. UHCI_REG_Q5_WORD0_REG (0x005C)

UHCI_SEND_Q5_WORD0	
31	0
0x000000	
Reset	

UHCI_SEND_Q5_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.51. UHCI_REG_Q5_WORD1_REG (0x0060)

UHCI_SEND_Q5_WORD1

31	0
0x000000	
Reset	

UHCI_SEND_Q5_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.52. UHCI_REG_Q6_WORD0_REG (0x0064)

UHCI_SEND_Q6_WORD0

31	0
0x000000	
Reset	

UHCI_SEND_Q6_WORD0 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.53. UHCI_REG_Q6_WORD1_REG (0x0068)

UHCI_SEND_Q6_WORD1

31	0
0x000000	
Reset	

UHCI_SEND_Q6_WORD1 在 UHCI_ALWAYS_SEND_NUM 或 UHCI_SINGLE_SEND_NUM 指定时用作快速发送寄存器。(R/W)

Register 22.54. UHCI_ESC_CONF0_REG (0x006C)

(reserved)								UHCI_SEPER_ESC_CHAR1								UHCI_SEPER_ESC_CHAR0								UHCI_SEPER_CHAR								
31								24	23							16	15							8	7							0
0	0	0	0	0	0	0	0	0xdc								0xdb								0xc0								Reset


- UHCI_SEPER_CHAR** 定义需编码的分隔符，默认为 0xc0。(R/W)
- UHCI_SEPER_ESC_CHAR0** 编码分隔符时定义 Slip 转义序列的第一个字符，分隔符默认为 0xdb。(R/W)
- UHCI_SEPER_ESC_CHAR1** 编码分隔符时定义 Slip 转义序列的第二个字符，分隔符默认为 0xdc。(R/W)

Register 22.55. UHCI_ESC_CONF1_REG (0x0070)

(reserved)								UHCI_ESC_SEQ0_CHAR1								UHCI_ESC_SEQ0_CHAR0								UHCI_ESC_SEQ0								
31								24	23							16	15							8	7							0
0	0	0	0	0	0	0	0	0xdd								0xdb								0xdb								Reset

- UHCI_ESC_SEQ0** 定义需编码的字符，默认为用作 Slip 转义序列第一个字符的 0xdb。(R/W)
- UHCI_ESC_SEQ0_CHAR0** 编码 UHCI_ESC_SEQ0 时定义 Slip 转义序列的第一个字符，UHCI_ESC_SEQ0 默认为 0xdb。(R/W)
- UHCI_ESC_SEQ0_CHAR1** 编码 UHCI_ESC_SEQ0 时定义 Slip 转义序列的第二个字符，UHCI_ESC_SEQ0 默认为 0xdd。(R/W)

Register 22.56. UHCI_ESC_CONF2_REG (0x0074)

(reserved)								UHCL_ESC_SEQ1_CHAR1								UHCL_ESC_SEQ1_CHAR0								UHCL_ESC_SEQ1											
31								24	23								16	15								8	7								0
0	0	0	0	0	0	0	0	0xde								0xdb								0x11								 Reset			

- UHCI_ESC_SEQ1** 定义需编码的字符，默认为用作流控字符的 0x11。(R/W)
- UHCI_ESC_SEQ1_CHAR0** 编码 UHCI_ESC_SEQ1 时定义 Slip 转义序列的第一个字符，UHCI_ESC_SEQ1 默认为 0xdb。(R/W)
- UHCI_ESC_SEQ1_CHAR1** 编码 UHCI_ESC_SEQ1 时定义 Slip 转义序列的第二个字符，UHCI_ESC_SEQ1 默认为 0xde。(R/W)

Register 22.57. UHCI_ESC_CONF3_REG (0x0078)

(reserved)								UHCI_ESC_SEQ2_CHAR1								UHCI_ESC_SEQ2_CHAR0								UHCI_ESC_SEQ2											
31								24	23								16	15								8	7								0
0	0	0	0	0	0	0	0	0xdf									0xdb								0x13								Reset		

- UHCI_ESC_SEQ2** 定义需编码的字符，默认为用作流控字符的 0x13。(R/W)
- UHCI_ESC_SEQ2_CHAR0** 编码 UHCI_ESC_SEQ2 时定义 Slip 转义序列的第一个字符，UHCI_ESC_SEQ2 默认为 0xdb。(R/W)
- UHCI_ESC_SEQ2_CHAR1** 编码 UHCI_ESC_SEQ2 时定义 Slip 转义序列的第二个字符，UHCI_ESC_SEQ2 默认为 0xdf。(R/W)

Register 22.58. UHCI_PKT_THRES_REG (0x007C)

(reserved)																UHCL_PKT_THRS																				
31																13	12																0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x80																Reset

- UHCI_PKT_THRS** UHCI_HEAD_EN 为 0 时配置包长度的最大值。(R/W)

374
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

UHCI TX START INT RAW 原始中断位。DMA 检测到分隔符时触发此中断。(R/WTC/SS)

UHCI_TX_HUNG_INT_RAW 原始中断位。DMA 读取 RAM 数据所需时间超过配置值时触发此中断。
(R/WTC/SS)

UHCI_SEND_S_REG_Q_INT_RAW 原始中断位。UHCI 使用 single_send 寄存器成功发送短包时触发此中断。(R/WTC/SS)

UHCI_SEND_A_REG_Q_INT_RAW 原始中断位。UHCI 使用 `always_send` 寄存器成功发送短包时触发此中断。(R/WTC/SS)

UHCI_OUT_EOF_INT_RAW 原始中断位。接收数据的 EOF 有错误时触发此中断。(R/WTC/SS)

UHCI_APP_CTRL0_INT_RAW 原始中断位，此位置 1 时触发中断，清零时清除中断。(R/W)

UHCI_APP_CTRL1_INT_RAW 原始中断位，此位置 1 时触发中断，清零时清除中断。(R/W)

Register 22.60. UHCI_INT_ST_REG (0x0008)

(reserved)																								UHCI_APP_CTRL1_INT_ST UHCI_APP_CTRL0_INT_ST UHCI_OUTLINK_EOF_ERR_INT_ST UHCI_SEND_A_REG_Q_INT_ST UHCI_SEND_S_REG_Q_INT_ST UHCI_TX_HUNG_INT_ST UHCI_RX_HUNG_INT_ST UHCI_TX_START_INT_ST UHCI_RX_START_INT_ST										
31																								9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

UHCI_RX_START_INT_ST UHCI_RX_START_INT_ENA 置 1 时 UHCI_RX_START_INT 中断的屏蔽中断位。(RO)

UHCI_TX_START_INT_ST UHCI_TX_START_INT_ENA 置 1 时 UHCI_TX_START_INT 中断的屏蔽中断位。(RO)

UHCI_RX_HUNG_INT_ST UHCI_RX_HUNG_INT_ENA 置 1 时 UHCI_RX_HUNG_INT 中断的屏蔽中断位。(RO)

UHCI_TX_HUNG_INT_ST UHCI_TX_HUNG_INT_ENA 置 1 时 UHCI_TX_HUNG_INT 中断的屏蔽中断位。(RO)

UHCI_SEND_S_REG_Q_INT_ST UHCI_SEND_S_REG_Q_INT_ENA 置 1 时 UHCI_SEND_S_REG_Q_INT 中断的屏蔽中断位。(RO)

UHCI_SEND_A_REG_Q_INT_ST UHCI_SEND_A_REG_Q_INT_ENA 置 1 时 UHCI_SEND_A_REG_Q_INT 中断的屏蔽中断位。(RO)

UHCI_OUTLINK_EOF_ERR_INT_ST UHCI_OUTLINK_EOF_ERR_INT_ENA 置 1 时 UHCI_OUTLINK_EOF_ERR_INT 中断的屏蔽中断位。(RO)

UHCI_APP_CTRL0_INT_ST UHCI_APP_CTRL0_INT_ENA 置 1 时 UHCI_APP_CTRL0_INT 中断的屏蔽中断位。(RO)

UHCI_APP_CTRL1_INT_ST UHCI_APP_CTRL1_INT_ENA 置 1 时 UHCI_APP_CTRL1_INT 中断的屏蔽中断位。(RO)

376
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

Register 22.62. UHCI_INT_CLR_REG (0x0010)

(reserved)																								UHCI_APP_CTRL1_INT_CLR UHCI_APP_CTRL0_INT_CLR UHCI_OUTLINK_EOF_ERR_INT_CLR UHCI_SEND_A_REG_Q_INT_CLR UHCI_SEND_S_REG_Q_INT_CLR UHCI_TX_HUNG_INT_CLR UHCI_RX_HUNG_INT_CLR UHCI_TX_START_INT_CLR UHCI_RX_START_INT_CLR										
31																								9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset						

- UHCI_RX_START_INT_CLR** 置位此位清除 UHCI_RX_START_INT 中断。(WT)
- UHCI_TX_START_INT_CLR** 置位此位清除 UHCI_TX_START_INT 中断。(WT)
- UHCI_RX_HUNG_INT_CLR** 置位此位清除 UHCI_RX_HUNG_INT 中断。(WT)
- UHCI_TX_HUNG_INT_CLR** 置位此位清除 UHCI_TX_HUNG_INT 中断。(WT)
- UHCI_SEND_S_REG_Q_INT_CLR** 置位此位清除 UHCI_SEND_S_REQ_Q_INT 中断。(WT)
- UHCI_SEND_A_REG_Q_INT_CLR** 置位此位清除 UHCI_SEND_A_REQ_Q_INT 中断。(WT)
- UHCI_OUTLINK_EOF_ERR_INT_CLR** 置位此位清除 UHCI_OUTLINK_EOF_ERR_INT 中断。(WT)
- UHCI_APP_CTRL0_INT_CLR** 置位此位清除 UHCI_APP_CTRL0_INT 中断。(WT)
- UHCI_APP_CTRL1_INT_CLR** 置位此位清除 UHCI_APP_CTRL1_INT 中断。(WT)

Register 22.63. UHCI_STATE0_REG (0x0018)

(reserved)																										UHCI_DECODE_STATE			UHCI_RX_ERR_CAUSE		
31																										6	5	3	2	0	Reset
0 0																										0		0			

- UHCI_RX_ERR_CAUSE** 在 DMA 接收到错误帧时表示错误类型。3'b001: HCI 包校验和错误；3'b010: HCI 包序列号错误。3'b011: HCI 包 CRC 位错误；3'b100: 找到 0xc0 但接收的 HCI 包不完整；3'b101: 未找到 0xc0 但接收的 HCI 包完整；3'b110: CRC 检测错误。(RO)
- UHCI_DECODE_STATE** UHCI 解码器状态。(RO)

Register 22.64. UHCI_STATE1_REG (0x001C)

(reserved)																												UHCI_ENCODE_STATE	
31																											3	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																												Reset	

UHCI_ENCODE_STATE UHCI 编码器状态。(RO)

Register 22.65. UHCI_RX_HEAD_REG (0x002C)

UHCI_RX_HEAD																													
31																													0
0x000000																												Reset	

UHCI_RX_HEAD 存储当前接收包的报头 (RO)

Register 22.66. UHCI_DATE_REG (0x0080)

UHCI_DATE																													
31																													0
0x2007170																												Reset	

UHCI_DATE 版本控制寄存器。(R/W)

23 I2C 控制器 (I2C)

I2C (Inter-Integrated Circuit) 总线用于使 ESP32-C3 和多个外部设备进行通信。多个外部设备可以共用一个 I2C 总线。

23.1 概述

I2C 是一个两线总线，由 SDA 线和 SCL 线构成。这些线设置为漏极开漏 (open-drain) 输出。因此，I2C 总线上可以挂载多个外设，通常是和一个或多个主机以及一个或多个从机。但同一时刻只有一个主机能占用总线访问一个从机。

主机发出开始信号，则通讯开始：在 SCL 为高电平时拉低 SDA 线，主机将通过 SCL 线发出 9 个时钟脉冲。前 8 个脉冲用于传输 7 位地址和 1 个读写位。如果从机地址与该 7 位地址一致，那么从机可以通过在第 9 个脉冲拉低 SDA 线来应答。接下来，根据读 / 写标志位，主机和从机可以发送 / 接收更多的数据。根据应答位的逻辑电平决定是否停止发送数据。在数据传输中，SDA 线仅在 SCL 线为低电平时才发生变化。当主机完成通讯，发送一个停止标志：在 SCL 为高电平时，拉高 SDA 线。如果一次通信中主机既有写操作又有读操作，则主机需在读写操作变化前，发送一个重新开始信号、从机地址和读写标志位。重新开始信号不仅用于一次通信中切换方向，也用于切换设备模式（主机或从机模式）。

23.2 主要特性

I2C 具有以下几个特点：

- 支持主机模式和从机模式
- 支持多从机通信
- 支持标准模式 (100 Kbit/s)
- 支持快速模式 (400 Kbit/s)
- 支持 7 位以及 10 位地址寻址
- 支持拉低 SCL 时钟实现连续数据传输
- 支持可编程数字噪声滤波功能
- 支持从机地址和从机内存或寄存器地址的双寻址模式

23.3 I2C 架构

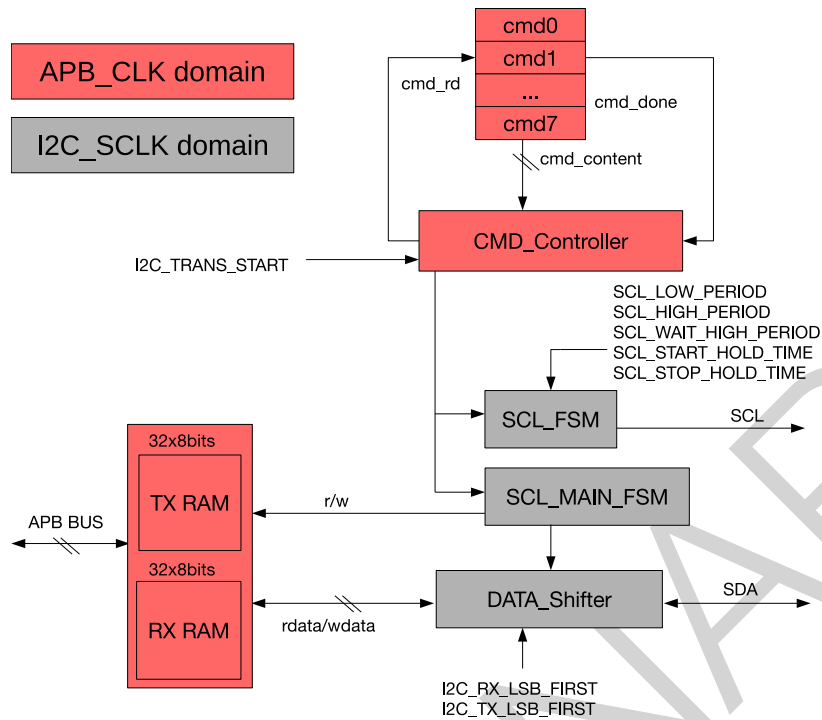


图 23-1. I2C 主机基本架构

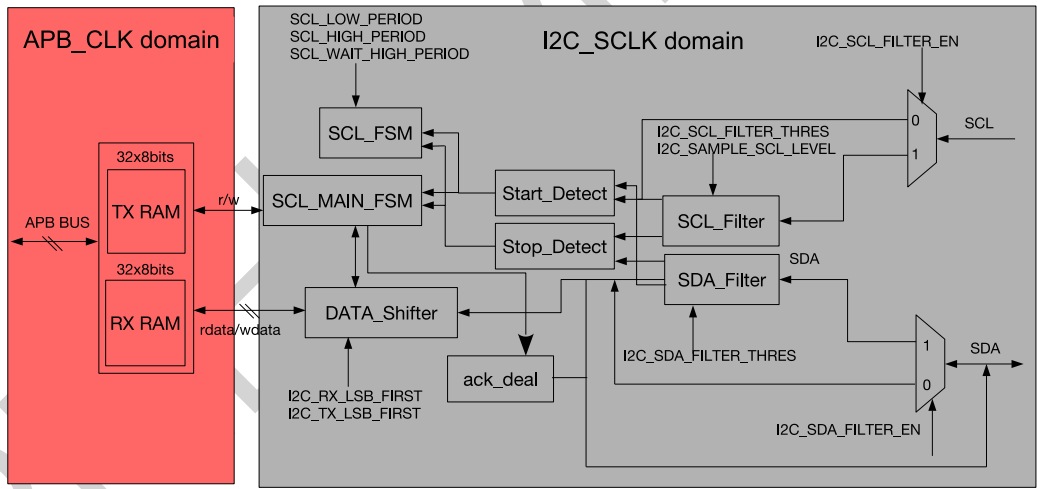


图 23-2. I2C 从机基本架构

I2C 控制器可以工作于主机模式或者从机模式，[I2C_MS_MODE](#) 用于模式选择。图 23-1 为 I2C 主机基本架构图，图 23-2 为 I2C 从机基本架构图。I2C 控制器内部包括的模块主要有：

- 接收和发送存储器 TX/RX RAM
- 命令控制器 CMD_Controller
- SCL 时钟控制器 SCL_FSM
- SDA 数据控制器 SCL_MAIN_FSM

- 串并转换器 DATA_Shifter
- SCL 滤波器 SCL_Filter
- SDA 滤波器 SDA_Filter

另外，还有产生 I2C 内部时钟的时钟模块，以及在 APB 总线和 I2C 模块之间同步的同步模块。

时钟模块的作用是进行时钟源选择，时钟开关和时钟分频。SCL_Filter 和 SDA_Filter 分别用于消除 SCL 及 SDA 输入信号上的噪声。同步模块用来同步不同时钟域之间信号的传输。

图 23-3 和图 23-4 是 I2C 协议的时序图和对应的参数表。SCL_FSM 用来产生满足 I2C 协议的时序序列。

SCL_MAIN_FSM 模块用来控制 I2C 指令的执行,和 SDA 线的序列。I2C 主机通过 CMD_Controller 产生 (R)START、STOP、WRITE、READ 和 END 指令。TX/RX RAM 分别用来存储 I2C 要发送和接收到的数据。DATA_Shifter 用来完成串行数据和并行数据之间的转换。

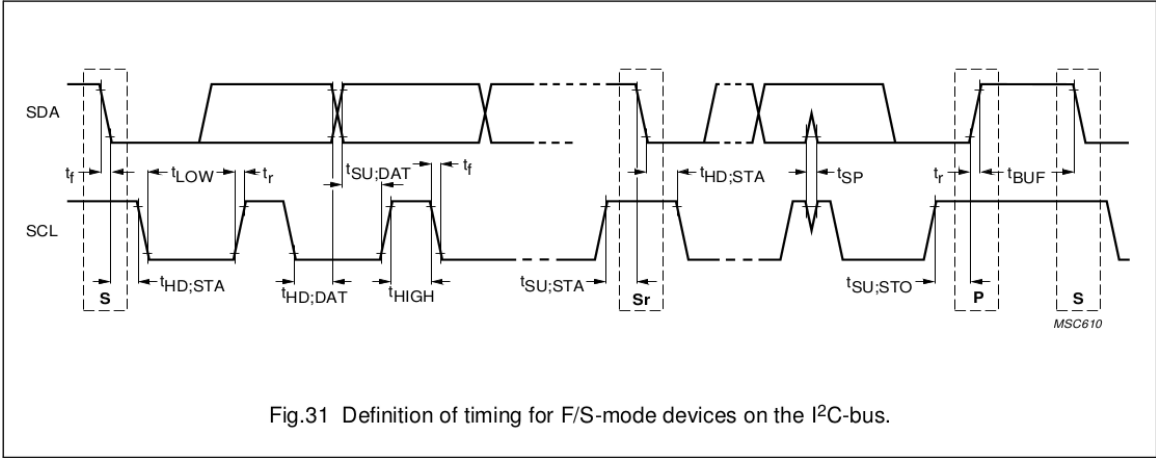


Fig.31 Definition of timing for F/S-mode devices on the I²C-bus.

图 23-3. I2C 协议时序 (引自 [The I2C-bus specification](#) Version 2.1 Fig.31)

PARAMETER	SYMBOL	STANDARD-MODE		FAST-MODE		UNIT
		MIN.	MAX.	MIN.	MAX.	
SCL clock frequency	f_{SCL}	0	100	0	400	kHz
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD;STA}$	4.0	—	0.6	—	μs
LOW period of the SCL clock	t_{LOW}	4.7	—	1.3	—	μs
HIGH period of the SCL clock	t_{HIGH}	4.0	—	0.6	—	μs
Set-up time for a repeated START condition	$t_{SU;STA}$	4.7	—	0.6	—	μs
Data hold time: for CBUS compatible masters (see NOTE, Section 10.1.3) for I ² C-bus devices	$t_{HD;DAT}$	5.0 0 ⁽²⁾	— 3.45 ⁽³⁾	— 0 ⁽²⁾	— 0.9 ⁽³⁾	μs μs
Data set-up time	$t_{SU;DAT}$	250	—	100 ⁽⁴⁾	—	ns
Rise time of both SDA and SCL signals	t_r	—	1000	$20 + 0.1C_b^{(5)}$	300	ns
Fall time of both SDA and SCL signals	t_f	—	300	$20 + 0.1C_b^{(5)}$	300	ns
Set-up time for STOP condition	$t_{SU;STO}$	4.0	—	0.6	—	μs
Bus free time between a STOP and START condition	t_{BUF}	4.7	—	1.3	—	μs

图 23-4. I2C 时序参数 (引自 [The I2C-bus specification](#) Version 2.1 Table5)

23.4 功能描述

需要注意的是，I2C 总线上其他主机或者从机的操作可能与 ESP32-C3 I2C 外设有所不同，具体请参考各个 I2C 设备的技术规格书。

23.4.1 时钟配置

寄存器配置和 TX/RX RAM 部分的时钟域为 APB_CLK，时钟范围是 1 ~ 80 MHz。I2C 主要逻辑部分，包括 SCL_FSM、SCL_MAIN_FSM、SCL_FILTER、SDA_FILTER 和 DATA_SHIFTER 都为 I2C_SCLK 时钟域。

用户可以通过配置 `I2C_SCLK_SEL` 选择 I2C_SCLK 的时钟源：XTAL_CLK 或 FOSC_CLK，`I2C_SCLK_SEL` 为 0 时选择时钟源 XTAL_CLK，`I2C_SCLK_SEL` 为 1 时选择时钟源 FOSC_CLK。配置 `I2C_SCLK_ACTIVE` 为高电平来打开 I2C_SCLK 的时钟源。选择后的时钟经过小数分频得到 I2C 的工作时钟 I2C_SCLK，分频系数为：

$$I2C_SCLK_DIV_NUM + 1 + \frac{I2C_SCLK_DIV_A}{I2C_SCLK_DIV_B}$$

XTAL_CLK 的频率是 40 MHz，FOSC_CLK 的频率是 17.5 MHz。根据时序参数的限制，分频后的 I2C_SCLK 的频率要满足大于 SCL 频率的 20 倍。

23.4.2 滤除 SCL 和 SDA 噪声

SCL_Filter 和 SDA_Filter 滤波器模块实现方式相同，用于滤除 SCL 及 SDA 输入信号上的噪声。通过配置 `I2C_SCL_FILTER_EN` 以及 `I2C_SDA_FILTER_EN` 寄存器可以开启或关闭滤波器。

以 SCL_Filter 为例，当使能 SCL_Filter 功能时，滤波器会连续采样输入信号 SCL，如果输入信号在连续 `I2C_SCL_FILTER_THRES` 个 I2C_SCLK 时钟周期内保持不变，则输入信号有效，否则输入信号无效。只有有效的输入信号才能通过滤波器。因此，SCL_Filter 和 SDA_Filter 滤波器会过滤脉冲宽度小于 `I2C_SCL_FILTER_THRES` 以及 `I2C_SDA_FILTER_THRES` 个 I2C_SCLK 时钟周期的线路毛刺。

23.4.3 SCL 时钟拉伸

从机模式下，可以通过拉低 SCL 线，给软件足够的时间处理数据。置位 `I2C_SLAVE_SCL_STRETCH_EN` 位使能 SCL 时钟拉伸功能，配置 `I2C_STRETCH_PROTECT_NUM` 字段来控制 SCL 拉伸后释放的时长以防出现时序错误。出现以下四种情况时从机会拉低 SCL 线：

1. 地址命中：从机模式下，从机地址与主机发送到 SDA 线上的地址相匹配，且读写标志位为 1。
2. 写满：从机模式下，I2C 控制器的 RX RAM 为满。注意，从机在接收少于 32 个字节时，可以不开启时钟拉伸功能；当要接收不少于 32 个字节时，可以通过 FIFO 阈值中断写 RAM 的乒乓操作，或者开始时钟拉伸功能，给软件提供处理时间。开启时钟拉伸功能时，必须将 `I2C_RX_FULL_ACK_LEVEL` 置 0，来保证功能正确，否则可能会出现不可预计的后果。
3. 读空：从机模式下，I2C 控制器要发送数据，但 TX RAM 为空。
4. 发送 ACK 时：从机模式下置位 `I2C_SLAVE_BYTE_ACK_CTL_EN`，从机会在发送 ACK 时拉低 SCL。软件在此阶段进行一些操作，如数据校验，并通过配置 `I2C_SLAVE_BYTE_ACK_LVL` 控制将要发送的 ACK 的电平高低。要注意的是，当出现从机接收的 RX RAM 满时，要发送的 ACK 电平将由 `I2C_RX_FULL_ACK_LEVEL` 而不是 `I2C_SLAVE_BYTE_ACK_LVL` 决定。此时同样需要将 `I2C_RX_FULL_ACK_LEVEL` 置 0，以保证 SCL 时钟拉伸功能的正常产生。

SCL 线拉低后，软件可读取 `I2C_STRETCH_CAUSE` 位获取 SCL 时钟拉伸的原因。置位 `I2C_SLAVE_SCL_STRETCH_CLR` 位关闭 SCL 时钟拉伸。

23.4.4 SCL 空闲时产生 SCL 脉冲

通常情况下，在 I2C 总线空闲时，SCL 线一直为高。ESP32-C3 I2C 支持在 I2C 主机处于空闲状态时，可编程配置产生 SCL 脉冲的功能。这个功能仅在 I2C 控制器作为主机时有效。置位 [I2C_SCL_RST_SLV_EN](#)，硬件会发送 [I2C_SCL_RST_SLV_NUM](#) 个 SCL 脉冲。一段时间后，软件读取到 [I2C_SCL_RST_SLV_EN](#) 的值为 0 后，再置位 [I2C_CONF_UPGATE](#)，来停止这个功能。

23.4.5 同步

I2C 的寄存器配置用 APB 时钟，I2C 主模块用 I2C_SCLK，这之间存在异步处理，需要增加同步的步骤将配置寄存器的值更新进入 I2C 主模块。步骤为先写配置寄存器，再向 [I2C_CONF_UPGATE](#) 位写 1。需要通过这种方法更新的配置寄存器详见表 23-1。

表 23-1. I2C 同步寄存器

配置寄存器	配置参数	地址
I2C_CTR_REG	I2C_SLV_TX_AUTO_START_EN	0x0004
	I2C_ADDR_10BIT_RW_CHECK_EN	
	I2C_ADDR_BROADCASTING_EN	
	I2C_SDA_FORCE_OUT	
	I2C_SCL_FORCE_OUT	
	I2C_SAMPLE_SCL_LEVEL	
	I2C_RX_FULL_ACK_LEVEL	
	I2C_MS_MODE	
	I2C_TX_LSB_FIRST	
	I2C_RX_LSB_FIRST	
	I2C_ARBITRATION_EN	
I2C_TO_REG	I2C_TIME_OUT_EN	0x000C
	I2C_TIME_OUT_VALUE	
I2C_SLAVE_ADDR_REG	I2C_ADDR_10BIT_EN	0x0010
	I2C_SLAVE_ADDR	
I2C_FIFO_CONF_REG	I2C_FIFO_ADDR_CFG_EN	0x0018
I2C_SCL_SP_CONF_REG	I2C_SDA_PD_EN	0x0080
	I2C_SCL_PD_EN	
	I2C_SCL_RST_SLV_NUM	
	I2C_SCL_RST_SLV_EN	
I2C_SCL_STRETCH_CONF_REG	I2C_SLAVE_BYTE_ACK_CTL_EN	0x0084
	I2C_SLAVE_BYTE_ACK_LVL	
	I2C_SLAVE_SCL_STRETCH_EN	
	I2C_STRETCH_PROTECT_NUM	
I2C_SCL_LOW_PERIOD_REG	I2C_SCL_LOW_PERIOD	0x0000
I2C_SCL_HIGH_PERIOD_REG	I2C_WAIT_HIGH_PERIOD	0x0038
	I2C_HIGH_PERIOD	
I2C_SDA_HOLD_REG	I2C_SDA_HOLD_TIME	0x0030
I2C_SDA_SAMPLE_REG	I2C_SDA_SAMPLE_TIME	0x0034
I2C_SCL_START_HOLD_REG	I2C_SCL_START_HOLD_TIME	0x0040
I2C_SCL_RSTART_SETUP_REG	I2C_SCL_RSTART_SETUP_TIME	0x0044

I2C_SCL_STOP_HOLD_REG	I2C_SCL_STOP_HOLD_TIME	0x0048
I2C_SCL_STOP_SETUP_REG	I2C_SCL_STOP_SETUP_TIME	0x004C
I2C_SCL_ST_TIME_OUT_REG	I2C_SCL_ST_TO_I2C	0x0078
I2C_SCL_MAIN_ST_TIME_OUT_REG	I2C_SCL_MAIN_ST_TO_I2C	0x007C
I2C_FILTER_CFG_REG	I2C_SCL_FILTER_EN	0x0050
	I2C_SCL_FILTER_THRES	
	I2C_SDA_FILTER_EN	
	I2C_SDA_FILTER_THRES	

23.4.6 漏级开路输出

SCL 及 SDA 线采用漏级开路的驱动方式。I2C 控制器有两种配置方式实现漏级开路驱动方式：

1. 置位 `I2C_SCL_FORCE_OUT`、`I2C_SDA_FORCE_OUT` 并配置相应 SCL 及 SDA PAD 的 `GPIO_PINn_PAD_DRIVER` 寄存器为漏级开路驱动。
2. 清零 `I2C_SCL_FORCE_OUT` 以及 `I2C_SDA_FORCE_OUT`。

SCL 和 SDA 配置成开漏方式时，从低电平转向高电平的时间会较长，这个转变时间由线上的上拉电阻以及电容共同决定。开漏模式下，I2C 输出频率的占空比受限于 SCL 上拉速度，主要受 SCL 的速度限制。

另外，在 `I2C_SCL_FORCE_OUT` 和 `I2C_SCL_PD_EN` 置 1 时，可以强制拉低 SCL 线；在 `I2C_SDA_FORCE_OUT` 和 `I2C_SDA_PD_EN` 置 1 时，可以强制拉低 SDA 线。

23.4.7 时序参数配置

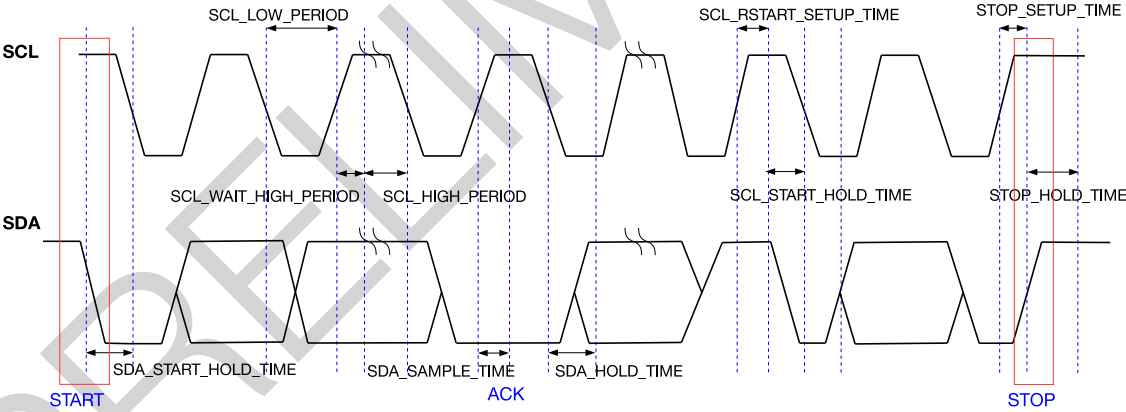


图 23-5. I2C 时序图

图 23-5 为实现 I2C 协议的 I2C 主机的时序图，图中的寄存器均用来配置时序参数。I2C 控制器的 START 位、STOP 位、数据保持时间、数据采样时间、SCL 上升沿等待时间等时序均可以通过图 23-5 中所示的寄存器进行配置。这些寄存器以模块时钟 (I2C_SCLK) 为单位，与各时序参数的对应关系为：

1. $t_{LOW} = (I2C_SCL_LOW_PERIOD + 1) \cdot T_{I2C_SCLK}$
2. $t_{HIGH} = (I2C_SCL_HIGH_PERIOD + 1) \cdot T_{I2C_SCLK}$
3. $t_{SU:STA} = (I2C_SCL_RSTART_SETUP_TIME + 1) \cdot T_{I2C_SCLK}$
4. $t_{HD:STA} = (I2C_SCL_START_HOLD_TIME + 1) \cdot T_{I2C_SCLK}$

5. $t_r = (I2C_SCL_WAIT_HIGH_PERIOD + 1) \cdot T_{I2C_SCLK}$
6. $t_{SU:STO} = (I2C_SCL_STOP_SETUP_TIME + 1) \cdot T_{I2C_SCLK}$
7. $t_{BUF} = (I2C_SCL_STOP_HOLD_TIME + 1) \cdot T_{I2C_SCLK}$
8. $t_{HD:DAT} = (I2C_SDA_HOLD_TIME + 1) \cdot T_{I2C_SCLK}$
9. $t_{SU:DAT} = (I2C_SCL_LOW_PERIOD - I2C_SDA_HOLD_TIME) \cdot T_{I2C_SCLK}$

根据在何种模式下有意义，下列时序寄存器可分为两组：

- 主机模式：

1. **I2C_SCL_START_HOLD_TIME**：生成 I2C 协议中的 START 位时，SDA 信号拉低到 SCL 信号拉低的时间间隔。该时间间隔为 $(I2C_SCL_START_HOLD_TIME + 1)$ 个模块时钟周期。仅控制器工作在主机模式时有意义。
2. **I2C_SCL_LOW_PERIOD**：SCL 低电平持续时间。SCL 低电平时间为 $(I2C_SCL_LOW_PERIOD + 1)$ 个模块时钟周期。但是如果外设拉低 SCL，I2C 控制器执行 END 命令拉低 SCL，或者控制器发生 SCL 时钟拉伸则可能会导致 SCL 低电平时间变长。仅控制器工作在主机模式时有意义。
3. **I2C_SCL_WAIT_HIGH_PERIOD**：等待 SCL 线拉高的模块时钟周期数。请确保在该时间内 SCL 线可以完成上拉。否则会导致 SCL 高电平持续时间不可预测。仅控制器工作在主机模式时有意义。
4. **I2C_SCL_HIGH_PERIOD**：SCL 线拉高后维持高电平的模块时钟周期数。仅控制器工作在主机模式时有意义。当 SCL 线在 $I2C_SCL_WAIT_HIGH_PERIOD + 1$ 个模块时钟内完成上拉，则 SCL 线的频率为：

$$f_{scl} = \frac{f_{I2C_SCLK}}{I2C_SCL_LOW_PERIOD + I2C_SCL_HIGH_PERIOD + I2C_SCL_WAIT_HIGH_PERIOD + 3}$$

- 主机模式和从机模式：

1. **I2C_SDA_SAMPLE_TIME**：SCL 上升沿到采样 SDA 线电平值的时间间隔。推荐设置在 SCL 高电平持续时间的中间值，以保证能够正确采样到 SDA 线上电平。控制器工作在主机模式及从机模式时都有意义。
2. **I2C_SDA_HOLD_TIME**：SDA 输出数据变化与 SCL 下降沿的时间间隔。控制器工作在主机模式及从机模式时都有意义。

根据时序参数的限制，对时序寄存器的配置范围也有约束。

1. $\frac{f_{I2C_SCLK}}{f_{SCL}} > 20$
2. $3 \times f_{I2C_SCLK} \leq (I2C_SDA_HOLD_TIME - 4) \times f_{APB_CLK}$
3. $I2C_SDA_HOLD_TIME + I2C_SCL_START_HOLD_TIME > SDA_FILTER_THRES + 3$
4. $I2C_SCL_WAIT_HIGH_PERIOD < I2C_SDA_SAMPLE_TIME < I2C_SCL_HIGH_PERIOD$
5. $I2C_SDA_SAMPLE_TIME < I2C_SCL_WAIT_HIGH_PERIOD + I2C_SCL_START_HOLD_TIME + I2C_SCL_RSTART_SETUP_TIME$
6. $I2C_STRETCH_PROTECT_NUM + I2C_SDA_HOLD_TIME > I2C_SCL_LOW_PERIOD$

23.4.8 超时控制

I2C 内部有三种超时控制，分别是对 SCL_FSM 状态的超时控制、SCL_MAIN_FSM 状态的超时控制和对 SCL 线状态的超时控制。其中前两种是一直打开的，第三种是可编程配置的。

当 SCL_FSM 长时间处于同一状态不变,且时间超过 $2^{I2C_SCL_ST_TO_I2C}$ 个时钟周期后,会触发 I2C_SCL_ST_TO_INT 中断,状态机会回到空闲状态。 $I2C_SCL_ST_TO_I2C$ 的配置值最大为 22,即最大在时间超过 2^{22} 个 I2C_SCLK 时钟周期后会产生超时中断。

当 SCL_MAIN_FSM 长时间处于同一状态不变,且时间超过 $2^{I2C_SCL_MAIN_ST_TO_I2C}$ 个模块时钟后,会触发 I2C_SCL_MAIN_ST_TO_INT 中断,状态机会回到空闲状态。 $I2C_SCL_MAIN_ST_TO_I2C$ 的配置值最大为 22,即最大在时间超过 2^{22} 个 I2C_SCLK 模块时钟后会产生超时中断。

使能 $I2C_TIME_OUT_EN$ 打开 SCL 线状态的超时控制。当 SCL 线状态长时间维持同一电平不变,且时间超过 $I2C_TIME_OUT_VALUE$ 后,会触发 I2C_TIME_OUT_INT 中断,I2C 总线回到空闲状态。

23.4.9 指令配置

I2C 控制器工作于主机模式时,CMD_Controller 会依次从八个命令寄存器中读出命令并按照命令来控制 SCL_FSM 及 SCL_MAIN_FSM。

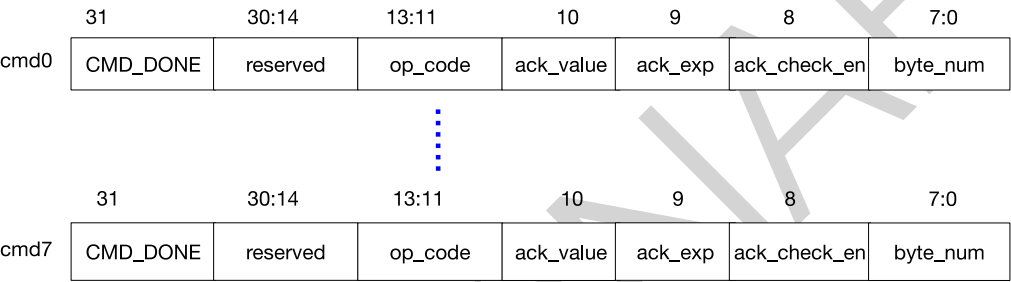


图 23-6. I2C 命令寄存器结构

命令寄存器只在 I2C 控制器工作于主机模式时才有效,其内部结构如图 23-6 所示。命令寄存器的参数为:

1. CMD_DONE: 命令执行完成标识。每条命令执行完硬件会将对应命令寄存器中的 CMD_DONE 置 1。软件可以通过读取每条命令的 CMD_DONE 位来判断该命令是否执行完毕。每次更新命令时,软件需要将 CMD_DONE 位清零。
2. op_code: 命令编码,共有五种命令:
 - RSTART: op_code 等于 6 时为 RSTART 命令,该命令指示 I2C 控制器发送 I2C 协议中的 START 位或 RSTART 位。
 - WRITE: op_code 等于 1 时为 WRITE 命令,该命令指示 I2C 控制器向从机发送从机地址、被访问的寄存器地址 (仅限双寻址模式)、数据。
 - READ: op_code 等于 3 时为 READ 命令,该命令指示 I2C 控制器从从机读取数据。
 - STOP: op_code 等于 2 时为 STOP 命令,该命令指示 I2C 控制器发送 I2C 协议中的 STOP 位。此命令也标识本次命令序列执行完成,CMD_Controller 将会停止取指令。软件再次启动 CMD_Controller 后,会重新从命令寄存器 0 开始去取指令。
 - END: op_code 等于 4 时为 END 命令,该命令指示 I2C 控制器将 SCL 信号拉低,暂停 I2C 通信。该命令也标识本次命令序列执行完成,CMD_Controller 将会停止取指令。软件在更新命令寄存器和 RAM 数据后可重新启动 CMD_Controller,继续进行 I2C 协议传输。再次启动后 CMD_Controller 会重新从命令寄存器 0 开始去取指令。
3. ack_value: 该位设置读操作时 I2C 控制器在 I2C 协议中的 ACK 位发送的电平值。RSTART、STOP、END、WRITE 命令中该位没有意义。

4. `ack_exp`: 该位用于设置写操作时 I2C 控制器在 I2C 协议中的 ACK 位期望接收的电平值。RSTART、STOP、END、READ 命令中该位没有意义。
5. `ack_check_en`: 该位使能写操作中 I2C 控制器检查从机发送的 ACK 位电平与命令中的 `ack_exp` 是否一致。如果接收的 ACK 值与 WRITE 命令中的 `ack_exp` 电平不一致时, I2C 主机会产生 `I2C_NACK_INT` 中断, 停止发送数据并产生 STOP。此位为 1 时, 检测从机发送的 ACK 位电平; 此位为 0 时, 不检测从机发送的 ACK 位电平。RSTART、STOP、END、READ 命令中该位没有意义。
6. `byte_num`: 读写数据的长度 (单位字节), 最大为 255, 最小为 1。RSTART、STOP、END 命令中 `byte_num` 无意义。

每次命令序列的执行都是从命令寄存器 0 开始, 到 STOP 或 END 命令结束。所以需要保证每个命令序列中必须有 STOP 或 END 命令。

一次完整的 I2C 协议传输应该起始于 START 命令, 结束于 STOP 命令。可通过 END 命令将一次 I2C 协议传输分为多个命令序列来完成。每个命令序列可以改变数据传输的方向、时钟频率、从机地址和数据长度等。这样可以弥补 RAM 大小不足的问题, 也可以实现更灵活的 I2C 通信。

23.4.10 TX/RX RAM 数据存储

TX/RX RAM 大小均为 32 x 8 位。TX RAM 和 RX RAM 均可以通过 FIFO 和直接地址 (non-FIFO) 两种方式访问。将 `I2C_NONFIFO_EN` 位设置成 0, 为 FIFO 方式; `I2C_NONFIFO_EN` 位设置成 1 时为直接地址方式。

TX RAM 用于存储 I2C 控制器需要发送的数据。在 I2C 通信的过程中, 当 I2C 控制器需要发送数据时 (不包括 ACK 位响应), 会依次读出 TX RAM 中的数据并串行输出到 SDA 线上。当 I2C 控制器工作于主机模式时, 所有需要发送给从机的数据都必须按照发送顺序依次存储在 TX RAM 中。包括被访问的从机地址、读写标志位、被访问的寄存器地址 (仅限双地址寻址模式下)、写数据。当 I2C 控制器工作于从机模式时, TX RAM 中只存放写数据。

TX RAM 可被 CPU 读写。CPU 可通过两种方式写 TX RAM: FIFO 访问和直接地址访问。FIFO 访问方式是通过固定地址 `I2C_DATA_REG` 写 TX RAM, 硬件自动进行 TX RAM 写地址自增。直接地址访问是通过地址段 (`I2C 基地址 + 0x100`) ~ (`I2C 基地址 + 0x17C`) 直接访问 TX RAM。TX RAM 的每一个字节占据一个字 (word) 的地址。因此, 第一个字节访问地址为 `I2C 基地址 + 0x100`, 第二字节访问地址为 `I2C 基地址 + 0x104`, 第三字节访问地址为 `I2C 基地址 + 0x108`, 以此类推。CPU 只可通过直接地址访问方式读 TX RAM, 读 TX RAM 的地址和写 TX RAM 的地址相同。

RX RAM 存储的是 I2C 通信过程中, I2C 控制器接收到的数据。当 I2C 控制器工作于从机模式时, 主机发送的从机地址及被访问的寄存器地址 (仅限双地址寻址模式下) 都不会存储在 RX RAM 中。软件可以在 I2C 通信结束后, 读出 RX RAM 的值。

RX RAM 只可被 CPU 读。CPU 可通过两种方式读 RX RAM: FIFO 访问和直接地址访问。FIFO 访问方式是通过固定地址 `I2C_DATA_REG` 读 RX RAM, 硬件自动完成 RX RAM 读地址自增。直接地址访问是通过地址段 (`I2C 基地址 + 0x180`) ~ (`I2C 基地址 + 0x1FC`) 直接访问 RX RAM。RX RAM 的每一个字节占据一个字的地址。因此, 第一个字节访问地址为 `I2C 基地址 + 0x180`, 第二字节访问地址为 `I2C 基地址 + 0x184`, 第三字节访问地址为 `I2C 基地址 + 0x188`, 以此类推。

在 FIFO 模式下可以对 TX RAM 进行乒乓操作, 来实现发送大于 32 个字节的数据。置位 `I2C_FIFO_PRT_EN`, 当 RAM 中剩下的待发送数据字节数小于 `I2C_TXFIFO_WM_THRHD` 时, 会产生 `I2C_TXFIFO_WM_INT` (主机) 中断。软件收到中断后, 继续向 `I2C_DATA_REG` (主机) 中写数。需要保证向 TX RAM 写数或更新数据的操作提前于 I2C 发送数据的动作, 否则会产生不可预计的后果。

在 FIFO 模式下也可以对 RX RAM 进行乒乓操作, 来实现接收大于 32 个字节的数据。置位 `I2C_FIFO_PRT_EN`, 将 `I2C_RX_FULL_ACK_LEVEL` 置 0。当 RAM 中收到的数据字节数大于等于 `I2C_RXFIFO_WM_THRHD` (从机)

时, 会产生 I2C_RXFIFO_WM_INT 中断。软件收到中断后, 从 I2C_DATA_REG (从机) 中读数。

23.4.11 数据转换

DATA_Shifter 模块用于串并转换, 当 I2C 发送数据时, 将 TX RAM 中的字节数据转化成比特流; 当 I2C 接收数据时, 将比特流转化成字节数据并存入 RX RAM。I2C_RX_LSB_FIRST 和 I2C_TX_LSB_FIRST 用于配置最高有效位或最低有效位的优先储存或传输。

23.4.12 寻址模式

除了 7 位寻址模式, ESP32-C3 I2C 还支持 10 位寻址模式和双寻址模式。10 位寻址和 7 位寻址可结合使用。

假设从机地址为 SLV_ADDR。ESP32-C3 I2C 控制器可以使用 7 位寻址 (SLV_ADDR[6:0]), 也可以使用 10 位寻址 (SLV_ADDR[9:0])。

对于主机模式而言, 7 位寻址下只要发送一个字节地址段 SLV_ADDR[6:0] 和读写标志。7 位寻址模式下, 有种特殊情况是广播寻址。在从机中, 将 I2C_ADDR_BROADCASTING_EN 置 1, 开启广播寻址模式。当接收到主机发送的地址为广播地址 (0x00) 且读写标志位为 0 时, 此时无论从机自己的地址是多少, 都会响应主机。

10 位寻址需要发送两字节地址段。第一个要发送的数是从机地址的第一个 7 位 slave_addr_first_7bits 和读写标志, slave_addr_first_7bits 的值应该配置为 (0x78 | SLV_ADDR[9:8])。第二个要发送的数是 slave_addr_second_byte, slave_addr_second_byte 的值为 SLV_ADDR[7:0]。在从机中, 可以通过配置 I2C_ADDR_10BIT_EN 寄存器开启 10 位寻址模式。I2C_SLAVE_ADDR 用于配置 I2C 从机地址。I2C_SLAVE_ADDR[14:7] 的值应配置为 SLV_ADDR[7:0], I2C_SLAVE_ADDR[6:0] 的值应配置为 (0x78 | SLV_ADDR[9:8])。由于 10 位从机地址比 7 位地址多一个字节, 所以 WRITE 命令对应的 byte_num 以及 RAM 中数据数量都相应增加 1。

控制器处于从机模式时, 还支持双地址访问方式。双地址的第一个地址是 I2C 从机地址, 第二个地址是 I2C 从机的内存地址。双地址模式下, RAM 必须采用 non-FIFO 方式访问。通过置位 I2C_FIFO_ADDR_CFG_EN 来使能双地址访问功能。

23.4.13 10 位寻址的读写标志位检查

在 10 位寻址模式下, 将 I2C_ADDR_10BIT_RW_CHECK_EN 置 1, 会对发送的第一个数 slave_addr_first_7bits 和读写标志做检查。当读写标志不是写, 此时与协议不符合, 会结束传输。若不打开此功能, 当读写标志不是写, 还能支持继续传输, 但可能出现传输错误。

23.4.14 启动控制器

对于主机, 配置成主机模式和命令寄存器等相关配置后, 通过向 I2C_TRANS_START 写 1, 启动主机解析, 执行命令序列。一组命令总是从命令寄存器 0 开始, 顺序执行到 STOP 或者 END 命令。当另一组命令需要从命令寄存器 0 开始执行时, 需要重新向 I2C_TRANS_START 写 1 来更新命令。

对于从机, 有两种启动方式:

- 置位 I2C_SLV_TX_AUTO_START_EN, 则从机会在被主机寻址后自动启动传输;
- 清零 I2C_SLV_TX_AUTO_START_EN, 且在每次传输前必须置位 I2C_TRANS_START。

23.5 编程示例

本节提供一些典型通信场景的编程示例。ESP32-C3 中有一个 I2C 控制器, 为了便于描述, 下文所有图示中的 I2C 主机和从机都假定为 ESP32-C3 I2C 外设控制器。

23.5.1 I2C 主机写入从机，7 位寻址，单次命令序列

23.5.1.1 场景介绍

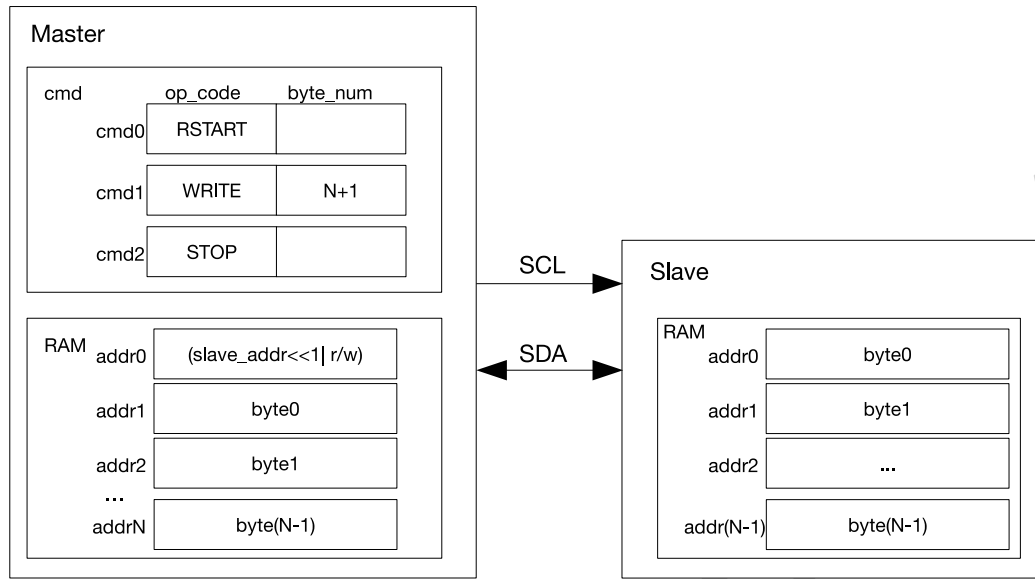


图 23-7. I2C 主机写 7 位寻址的从机

图 23-7 为 I2C 主机采用 7 位寻址写 N 个字节数据到 I2C 从机的 RAM。如图 23-7 所示，I2C 主机 RAM 中第一个字节数据为 7 位从机地址 + 1 位读写标志位，其中读写标志位为 0 时表示写操作，接下来的连续空间存储待发送的数据。cmd 框中包含了相应的命令序列。

对于主机，在软件配置好命令序列以及 RAM 数据后，置位 `I2C_TRANS_START` 寄存器启动控制器进行数据传输。控制器的行为可分为四步：

1. 等待 SCL 线为高电平，以避免 SCL 线被其他主机或者从机占用。
2. 执行 RSTART 命令发送 START 位。
3. 执行 WRITE 命令从 RAM 的首地址开始取出 N+1 个字节并依次发送给从机，其中第一个字节为地址。
4. 发送 STOP。当 I2C 主机完成 STOP 位的传输后，会产生 `I2C_TRANS_COMPLETE_INT` 中断。

23.5.1.2 配置示例

1. 设置 `I2C_MS_MODE`（主机）为 1，`I2C_MS_MODE`（从机）为 0。
2. 向 `I2C_CONF_UPGATE`（主机）和 `I2C_CONF_UPGATE`（从机）写 1 来同步寄存器。

3. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0 (主机)	RSTART	—	—	—	—
I2C_COMMAND1 (主机)	WRITE	ack_value	ack_exp	1	N+1
I2C_COMMAND2 (主机)	STOP	—	—	—	—

4. 参考章节 23.4.10，向 I2C 主机的 TX RAM 写入从机地址和要发送的数据。可选 FIFO 方式和直接访问方式。
5. 在 I2C_SLAVE_ADDR_REG (从机) 的 I2C_SLAVE_ADDR (从机) 设置 I2C 从机的地址。
6. 向 I2C_CONF_UPGATE (主机) 和 I2C_CONF_UPGATE (从机) 写 1 来同步寄存器。
7. 向 I2C_TRANS_START (主机) 和 I2C_TRANS_START (从机) 位写 1 开始传输。
8. I2C 从机比较 I2C 主机发送的从机地址和自己的 I2C_SLAVE_ADDR (从机)，当 I2C 主机 WRITE 命令中的 ack_check_en (主机) 配置为 1 时，I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0，则不会对 ACK 检测，会默认为匹配。
- 匹配：接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平一致，传输继续。
 - 不匹配：接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平不一致，I2C 主机产生 I2C_NACK_INT (主机) 中断，停止发送数据并且产生 STOP。
9. I2C 主机发送数据，并根据 ack_check_en (主机) 配置的不同进行或不进行 ACK 检测。
10. 若发送数据 N 大于 32 字节，在 FIFO 模式下可以对 I2C 主机的 TX RAM 进行乒乓操作，具体做法参照章节 23.4.10。
11. 若接收数据 N 大于 32 字节，在 FIFO 模式下可以对 I2C 从机的 RX RAM 进行乒乓操作，具体做法参照章节 23.4.10。
- 若接收数据 N 大于 32 字节，另一种处理方式是置位 I2C_SLAVE_SCL_STRETCH_EN (从机) 使能 SCL 时钟拉伸，同时将 I2C_RX_FULL_ACK_LEVEL 置 0。RX RAM 满时会产生 I2C_SLAVE_STRETCH_INT (从机) 中断。此时 I2C 从机会将 SCL 拉低，软件在此期间可以读取数据。等完成操作后再将 I2C_SLAVE_STRETCH_INT_CLR (从机) 置 1 清除中断，并将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1 释放 SCL 总线。
12. 当整个传输正常结束，I2C 主机执行 STOP 命令，并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.2 I2C 主机写入从机，10 位寻址，单次命令序列

23.5.2.1 场景介绍

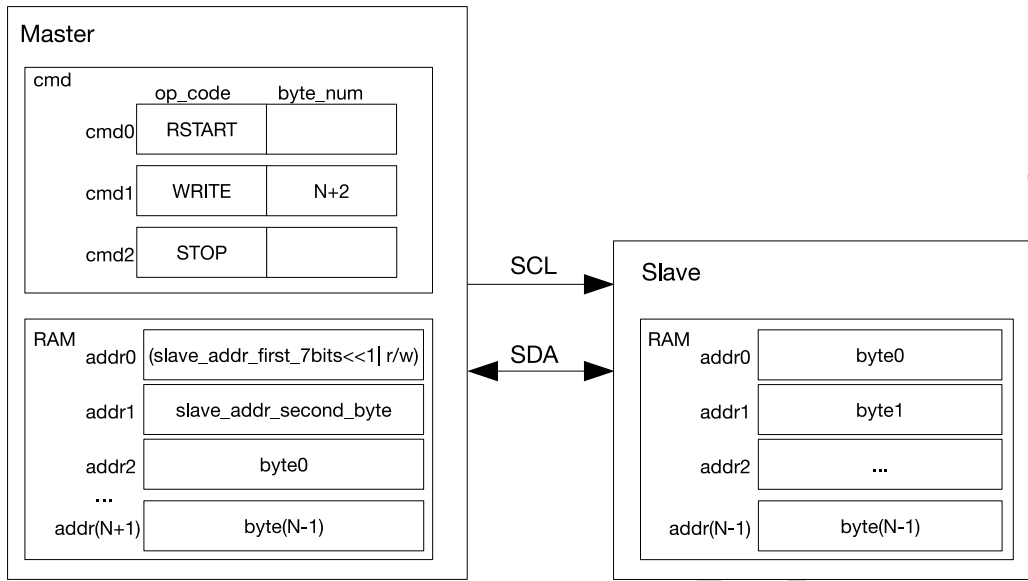


图 23-8. I2C 主机写 10 位寻址的从机

图 23-8 为 I2C 主机写 N 个字节到 10 位地址 I2C 从机的配置图。整个配置和传输过程都和 23.5.1 中类似，除了在传输的开始 I2C 主机在 10 位寻址模式下需要发送两字节地址段。由于 10 位从机地址比 7 位地址多一个字节，所以 WRITE 命令对应的 byte_num 以及 TX RAM 中数据数量都相应增加 1。

23.5.2.2 配置示例

1. 设置 `I2C_MS_MODE` (主机) 为 1, `I2C_MS_MODE` (从机) 为 0。
2. 向 `I2C_CONF_UPGATE` (主机) 和 `I2C_CONF_UPGATE` (从机) 写 1 来同步寄存器。
3. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
<code>I2C_COMMAND0</code> (主机)	RSTART	—	—	—	—
<code>I2C_COMMAND1</code> (主机)	WRITE	ack_value	ack_exp	1	N+2
<code>I2C_COMMAND2</code> (主机)	STOP	—	—	—	—

4. 在 `I2C_SLAVE_ADDR_REG` (从机) 的 `I2C_SLAVE_ADDR` (从机) 设置 I2C 从机。的 10 位从机地址，并将 `I2C_ADDR_10BIT_EN` (从机) 置 1 使能 10 位寻址模式。
5. 向 I2C 主机的 TX RAM 写入从机地址和要发送的数据, 第一个地址字节是 $((0x78 | I2C_SLAVE_ADDR[9:8]) \ll 1) | R/W$, 第二个地址字节是 `I2C_SLAVE_ADDR[7:0]`。之后就是要发送的数据，可选 FIFO 方式和直接访问方式。
6. 向 `I2C_CONF_UPGATE` (主机) 和 `I2C_CONF_UPGATE` (从机) 写 1 来同步寄存器。
7. 向 `I2C_TRANS_START` (主机) 和 `I2C_TRANS_START` (从机) 位写 1 开始传输。
8. I2C 从机比较 I2C 主机发送的从机地址和自己的 `I2C_SLAVE_ADDR` (从机)，当 I2C 主机 WRITE 命令中的

ack_check_en (主机) 配置为 1 时, I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0, 则不会对 ACK 检测, 会默认为匹配。

- 匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平一致, 传输继续。
- 不匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平不一致, I2C 主机产生 I2C_NACK_INT (主机) 中断, 停止发送数据并且产生 STOP。

9. I2C 主机发送数据, 并根据 ack_check_en (主机) 配置的不同进行或不进行 ACK 检测。

10. 若发送数据 N 大于 32 字节, 在 FIFO 模式下可以对 I2C 主机的 TX RAM 进行乒乓操作, 具体做法参照章节 23.4.10。

11. 若接收数据 N 大于 32 字节, 在 FIFO 模式下可以对 I2C 从机的 RX RAM 进行乒乓操作, 具体做法参照章节 23.4.10。

若接收数据 N 大于 32 字节, 另一种处理方式是置位 I2C_SLAVE_SCL_STRETCH_EN (从机) 使能 SCL 时钟拉伸, 同时将 I2C_RX_FULL_ACK_LEVEL 置 0。RX RAM 满时会产生 I2C_SLAVE_STRETCH_INT (从机) 中断。此时 I2C 从机将 SCL 拉低, 软件在此期间可以读取数据。等完成操作后再将 I2C_SLAVE_STRETCH_INT_CLR (从机) 置 1 清除中断, 并将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1 释放 SCL 总线。

12. 当整个传输正常结束, I2C 主机执行 STOP 命令, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.3 I2C 主机写入从机, 7 位双地址寻址, 单次命令序列

23.5.3.1 场景介绍

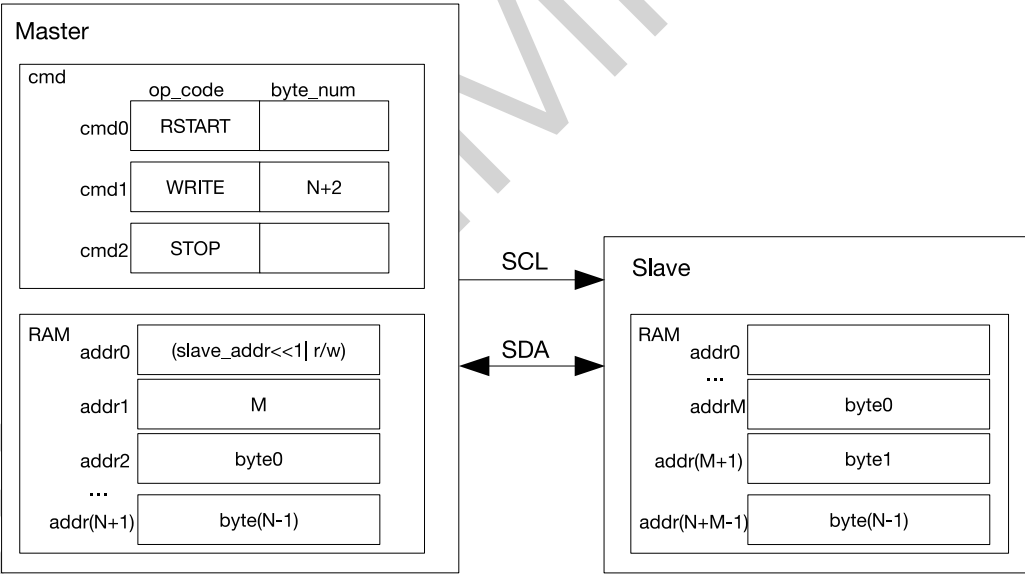


图 23-9. I2C 主机写 7 位双地址寻址从机

图 23-9 为 I2C 主机采用 7 位双寻址模式写 N 个字节数据到 I2C 从机的 RAM。整个配置和传输过程都和章节 23.5.1 中类似, 区别是传输的开始 I2C 主机在 7 位双寻址模式下需要发送两个字节。双地址的第一个地址是 7 位 I2C 从机地址, 第二个地址是 I2C 从机的内存地址 (即图 23-9 中的 addrM)。双地址模式下, RX RAM 必须采用 non-FIFO 方式访问。另一个区别是, I2C 从机将接收到的数据 byte0 ~ byte(N-1) 从 RX RAM 中的 addrM 开始依次存储, addrM 就是 I2C 主机发送的 I2C 内存地址。当超出地址 31 后会从地址 0 开始继续存储。

23.5.3.2 配置示例

- 1. 设置 I2C_MS_MODE (主机) 为 1, I2C_MS_MODE (从机) 为 0。
- 2. 设置 I2C_FIFO_ADDR_CFG_EN (从机) 为 1 来使能双寻址模式。
- 3. 向 I2C_CONF_UPGATE (主机) 和 I2C_CONF_UPGATE (从机) 写 1 来同步寄存器。
- 4. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0 (主机)	RSTART	—	—	—	—
I2C_COMMAND1 (主机)	WRITE	ack_value	ack_exp	1	N+2
I2C_COMMAND2 (主机)	STOP	—	—	—	—

- 5. 向 I2C 主机的 TX RAM 写入从机地址和要发送的数据, 可以用 FIFO 方式或直接访问方式。
- 6. 在 I2C_SLAVE_ADDR_REG (从机) 的 I2C_SLAVE_ADDR (从机) 设置 I2C 从机的地址。
- 7. 向 I2C_CONF_UPGATE (主机) 和 I2C_CONF_UPGATE (从机) 写 1 来同步寄存器。
- 8. 向 I2C_TRANS_START (主机) 和 I2C_TRANS_START (从机) 位写 1 开始传输。
- 9. I2C 从机比较 I2C 主机发送的从机地址和自己的 I2C_SLAVE_ADDR (从机), 当 I2C 主机 WRITE 命令中的 ack_check_en (主机) 配置为 1 时, I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0, 则不会对 ACK 检测, 会默认为匹配。
 - 匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平一致, 传输继续。
 - 不匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平不一致, I2C 主机产生 I2C_NACK_INT (主机) 中断, 停止发送数据并且产生 STOP。
- 10. I2C 从机接收到 I2C 主机发送的内存地址, 完成 RX RAM 的地址偏移。
- 11. I2C 主机发送数据, 并根据 ack_check_en (主机) 配置的不同进行或不进行 ACK 检测。
- 12. 若发送数据 N 大于 32 字节, 在 FIFO 模式下可以对 I2C 主机的 TX RAM 进行乒乓操作, 具体做法参照章节 23.4.10。
- 13. 若接收数据 N 大于 32 字节, 置位 I2C_SLAVE_SCL_STRETCH_EN (从机) 使能 SCL 时钟拉伸, 同时将 I2C_RX_FULL_ACK_LEVEL 置 0。RX RAM 满时会产生 I2C_SLAVE_STRETCH_INT (从机) 中断。此时 I2C 从机会将 SCL 拉低, 软件在此期间可以读取数据。等完成操作后再将 I2C_SLAVE_STRETCH_INT_CLR (从机) 置 1 清除中断, 并将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1 释放 SCL 总线。
- 14. 当整个传输正常结束, I2C 主机执行 STOP 命令, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.4 I2C 主机写入从机，7 位寻址，多次命令序列

23.5.4.1 场景介绍

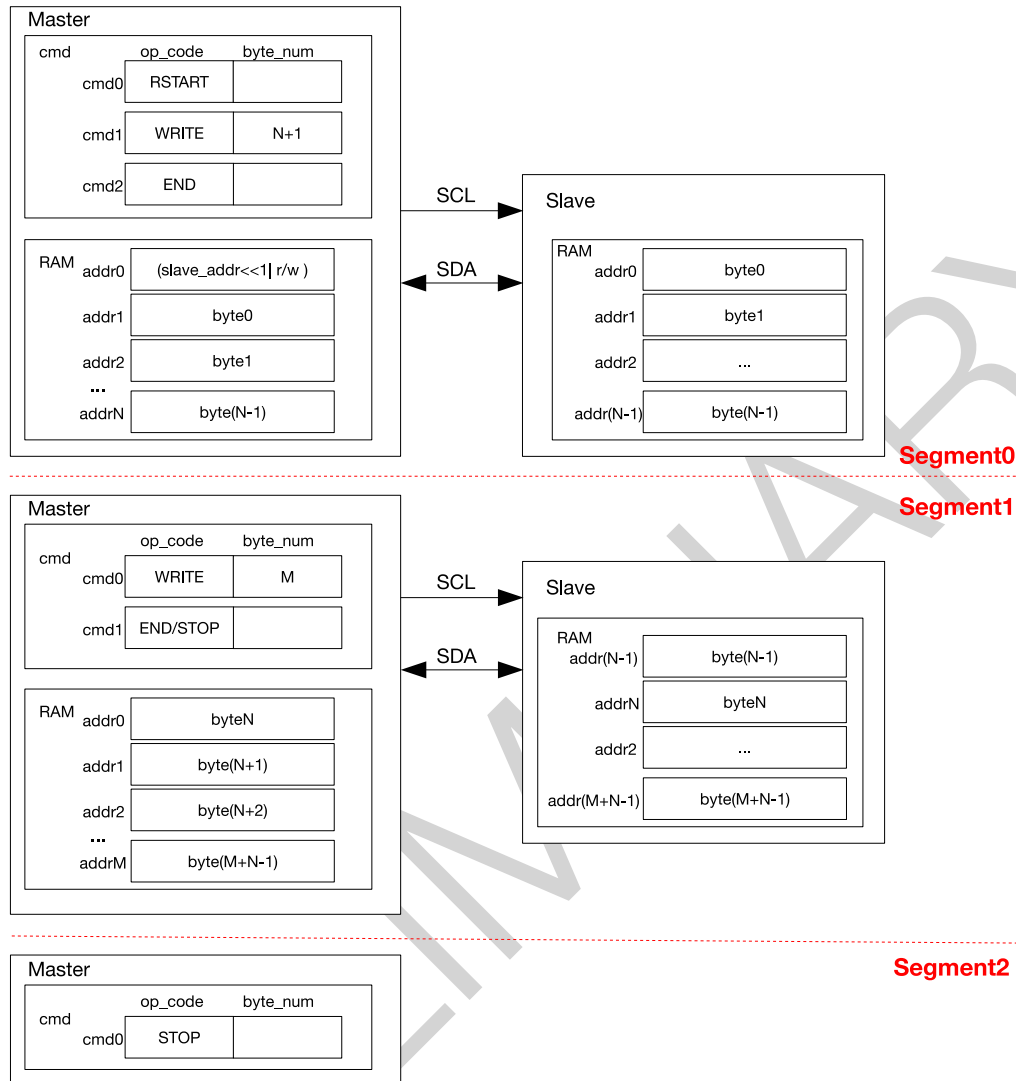


图 23-10. I2C 主机分段写 7 位寻址的从机

RAM 的大小只有 32 字节，对于大量的数据传输当 RAM 乒乓操作也不能满足要求时，建议使用多次命令序列进行分段传输。每次命令序列以 END 命令结尾，这样控制器会执行 END 命令拉低 SCL 线，软件此时可以更新命令序列寄存器和 RAM 的内容以用于下一次命令序列的传输。

以两段和三段传输为例，如图 23-10 所示为 I2C 主机分成三段或者两段写 I2C 从机。配置 I2C 主机的命令序列如第一段所示，并且在 I2C 主机的 RAM 中准备好数据，置位 `I2C_TRANS_START`，I2C 主机即开始数据传输。在执行到 END 命令后，I2C 主机会关闭 SCL 时钟，并将 SCL 线拉低来防止其他设备占用 I2C 总线。此时控制器产生 `I2C_END_DETECT_INT` 中断。

在检测到 `I2C_END_DETECT_INT` 中断后，软件可以更新命令序列以及 RAM 中的内容如第二段所示，并清除 `I2C_END_DETECT_INT` 中断。当第二段中 `cmd1` 为 STOP 时，不需要第三段，即为两段写 I2C 从机。置位 `I2C_TRANS_START` 后，I2C 主机继续发送数据，并在最后发送 STOP 位。当为三段写 I2C 从机时，I2C 主机在第二段发送完数据，并检测到 I2C 主机的 `I2C_END_DETECT_INT` 中断后，即可配置 `cmd` 如第三段所示。置位 `I2C_TRANS_START` 后，I2C 主机即产生 STOP 位，从而停止传输。

请注意，在两个分段之间，I2C 总线上的其他 I2C 主机设备不会占用总线。只有在发送了 STOP 信号后总线才会被释放。任何情况下，置位 I2C_FSM_RST 可复位 I2C 控制器，硬件自清 I2C_FSM_RST。

23.5.4.2 配置示例

- 1. 设置 I2C_MS_MODE（主机）为 1，I2C_MS_MODE（从机）为 0。
- 2. 向 I2C_CONF_UPGATE（主机）和 I2C_CONF_UPGATE（从机）写 1 来同步寄存器。
- 3. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0（主机）	RSTART	—	—	—	—
I2C_COMMAND1（主机）	WRITE	ack_value	ack_exp	1	N+1
I2C_COMMAND2（主机）	END	—	—	—	—

- 4. 参考章节 23.4.10，向 I2C 主机的 TX RAM 写入从机地址和要发送的数据。可选 FIFO 方式和直接访问方式。
- 5. 在 I2C_SLAVE_ADDR_REG（从机）的 I2C_SLAVE_ADDR（从机）设置 I2C 从机的地址。
- 6. 向 I2C_CONF_UPGATE（主机）和 I2C_CONF_UPGATE（从机）写 1 来同步寄存器。
- 7. 向 I2C_TRANS_START（主机）和 I2C_TRANS_START（从机）位写 1 开始传输。
- 8. I2C 从机比较 I2C 主机发送的从机地址和自己的 I2C_SLAVE_ADDR（从机），当 I2C 主机 WRITE 命令中的 ack_check_en（主机）配置为 1 时，I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0，则不会对 ACK 检测，会默认为匹配。
 - 匹配：接收的 ACK 值与 WRITE 命令中的 ack_exp（主机）电平一致，传输继续。
 - 不匹配：接收的 ACK 值与 WRITE 命令中的 ack_exp（主机）电平不一致，I2C 主机产生 I2C_NACK_INT（主机）中断，停止发送数据并且产生 STOP。
- 9. I2C 主机发送数据，并根据 ack_check_en（主机）配置的不同进行或不进行 ACK 检测。
- 10. 等到 I2C_END_DETECT_INT（主机）中断产生后，设置 I2C_END_DETECT_INT_CLR（主机）为 1 来清除中断。
- 11. 更新 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0（主机）	WRITE	ack_value	ack_exp	1	M
I2C_COMMAND1（主机）	END/STOP	—	—	—	—

- 12. 向 I2C 主机的 TX RAM 写入 M 个要发送的数据，可以用 FIFO 方式或直接访问方式。
- 13. 向 I2C_TRANS_START（主机）位写 1 开始传输，并重复步骤 9 的流程。
- 14. 若指令为 STOP，I2C 主机执行 STOP 命令结束传输，并产生 I2C_TRANS_COMPLETE_INT（主机）中断。
- 15. 若 I2C_COMMAND1（主机）的指令为 END，则重复步骤 10。
- 16. 更新 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND1 (主机)	STOP	—	—	—	—

- 17. 向 I2C_TRANS_START (主机) 位写 1 开始传输。
- 18. I2C 主机执行 STOP 命令结束传输，并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.5 I2C 主机读取从机，7 位寻址，单次命令序列

23.5.5.1 场景介绍

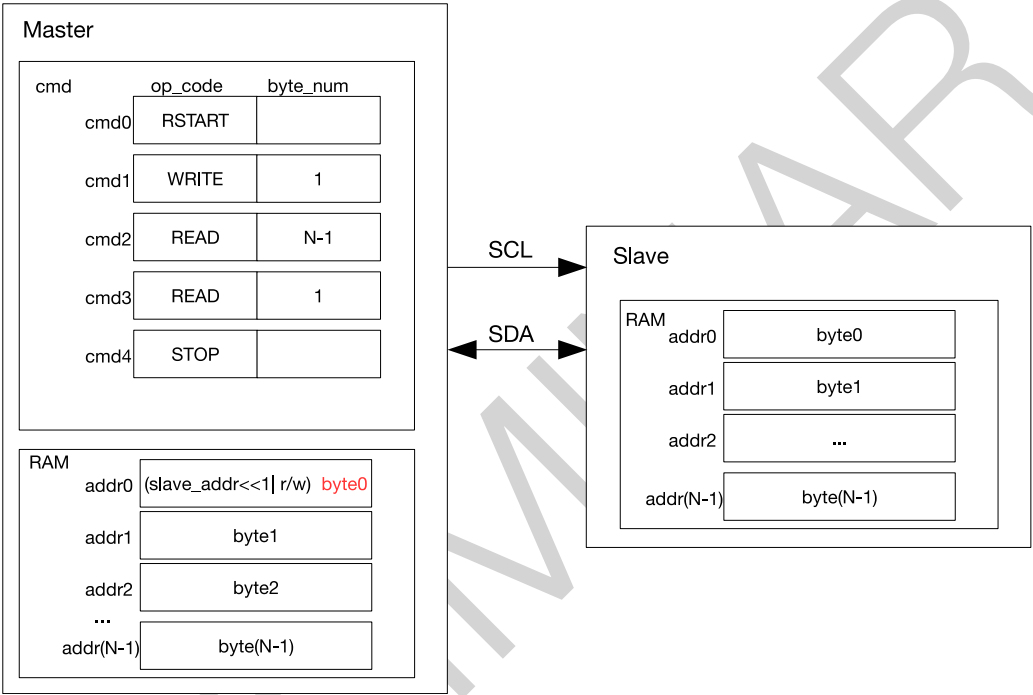


图 23-11. I2C 主机读 7 位寻址的从机

图 23-11 I2C 主机从 7 位寻址 I2C 从机读取 N 个字节数据的值。cmd1 为 WRITE 命令，I2C 主机会将 I2C 从机的地址发送出去。该命令发送的字节是 7 位 I2C 从机地址以及读写标志位。读写标志位为 1 表示读操作。I2C 从机在地址匹配成功之后即开始发送数据给 I2C 主机。I2C 主机根据 READ 命令中设置的 ack_value，在接收完一个字节的数据之后回复 ACK。

图 23-11 中 READ 分成两次，I2C 主机对 cmd2 中 N-1 个数据均回复 ACK，对 cmd3 中的数据即传输的最后一个数据回复 NACK，实际使用时可以根据需要进行配置。在存储接收的数据时，I2C 主机从 RX RAM 的首地址开始存储，即为图中红色 byte0 存储的位置。

23.5.5.2 配置示例

- 1. 设置 I2C_MS_MODE (主机) 为 1，I2C_MS_MODE (从机) 为 0。
- 2. 推荐将 I2C_SLAVE_SCL_STRETCH_EN (从机) 置 1，以便 I2C 从机 I2C 从机在需要发送数据时可以先将 SCL 拉低来给软件向 I2C 从机的 TX RAM 中写数提供时间，否则 I2C 从机需要在 I2C 主机开始传输前提前准备好数据。以下配置流程均按照 I2C_SLAVE_SCL_STRETCH_EN (从机) 为 1 的情况进行。

3. 向 `I2C_CONF_UPGATE` (主机) 和 `I2C_CONF_UPGATE` (从机) 写 1 来同步寄存器。
4. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
<code>I2C_COMMAND0</code> (主机)	RSTART	—	—	—	—
<code>I2C_COMMAND1</code> (主机)	WRITE	0	0	1	1
<code>I2C_COMMAND2</code> (主机)	READ	0	0	1	N-1
<code>I2C_COMMAND3</code> (主机)	READ	1	0	1	1
<code>I2C_COMMAND4</code> (主机)	STOP	—	—	—	—

5. 参考章节 23.4.10, 向 I2C 主机的 TX RAM 写入 I2C 从机地址。可选 FIFO 方式和直接访问方式。
6. 在 `I2C_SLAVE_ADDR_REG` (从机) 的 `I2C_SLAVE_ADDR` (从机) 设置 I2C 从机的地址。
7. 向 `I2C_CONF_UPGATE` (主机) 和 `I2C_CONF_UPGATE` (从机) 写 1 来同步寄存器。
8. 向 `I2C_TRANS_START` (主机) 写 1 开始 I2C 主机的传输。
9. 参考章节 23.4.14, 启动 I2C 从机的传输。
10. I2C 从机比较 I2C 主机发送的从机地址和自己的 `I2C_SLAVE_ADDR` (从机), 当 I2C 主机 WRITE 命令中的 `ack_check_en` (主机) 配置为 1 时, I2C 主机会在发送完每个字节之后进行 ACK 检测。若 `ack_check_en` 配置为 0, 则不会对 ACK 检测, 会默认为匹配。
 - 匹配: 接收的 ACK 值与 WRITE 命令中的 `ack_exp` (主机) 电平一致, 传输继续。
 - 不匹配: 接收的 ACK 值与 WRITE 命令中的 `ack_exp` (主机) 电平不一致, I2C 主机产生 `I2C_NACK_INT` (主机) 中断, 停止发送数据并且产生 STOP。
11. 等待 `I2C_SLAVE_STRETCH_INT` (从机), 读取 `I2C_STRETCH_CAUSE` 的值为 0, 此时 I2C 从机地址与 SDA 线上发送的地址相匹配, 且 I2C 从机要发送数据。
12. 参考章节 23.4.10, 向 I2C 从机的 TX RAM 写入要发送的数据。可选 FIFO 方式和直接访问方式。
13. 将 `I2C_SLAVE_SCL_STRETCH_CLR` (从机) 置 1, 释放 SCL 总线。
14. I2C 从机发送数据, I2C 主机会根据当前 READ 指令对应的 `ack_check_en` (主机) 配置的不同进行或不进行 ACK 检测。
15. 若 I2C 主机要读取的数超过 32 个, 当发送数据全部发完, 在 I2C 从机的 TX RAM 空时产生 `I2C_SLAVE_STRETCH_INT` (从机) 中断。此时 I2C 从机将 SCL 拉低, 软件在此期间可以继续向 I2C 从机的 TX RAM 填充数据, 也可以从 I2C 主机的 RX RAM 读取数据。等完成操作后再将 `I2C_SLAVE_STRETCH_INT_CLR` (从机) 置 1 清除中断, 将 `I2C_SLAVE_SCL_STRETCH_CLR` (从机) 置 1 释放 SCL 总线。
16. 当 I2C 主机接收最后一个数据时, 将 `ack_value` (主机) 设成 1, I2C 从机接收到 NACK 中断, 停止发送。
17. 当整个传输正常结束, I2C 主机执行 STOP 命令, 并产生 `I2C_TRANS_COMPLETE_INT` (主机) 中断。

23.5.6 I2C 主机读取从机，10 位寻址，单次命令序列

23.5.6.1 场景介绍

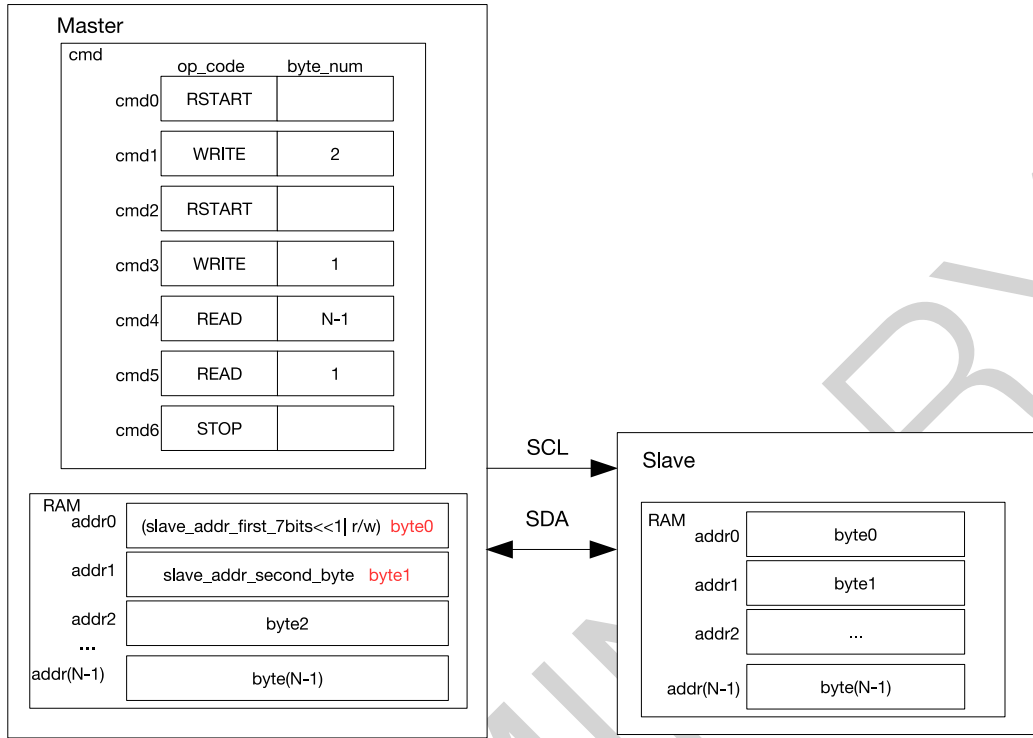


图 23-12. I2C 主机读 10 位寻址的从机

图 23-12 为 I2C 主机从 10 位寻址的 I2C 从机中读取数据的值。相比于 7 位寻址，I2C 主机的第一写命令的字节数为两个字节，相应 TX RAM 中存储两个字节的 I2C 从机 10 位地址，且第一个地址字节的 R/W 位为 W（主机）。之后再次发送 RSTART，并重复发送第一个地址字节，R/W 位为 R（从机）。之后 I2C 主机从 I2C 从机中读取数据。两个字节地址的配置方式与章节 23.5.2 的相同。

23.5.6.2 配置示例

1. 设置 `I2C_MS_MODE`（主机）为 1，`I2C_MS_MODE`（从机）为 0。
2. 推荐将 `I2C_SLAVE_SCL_STRETCH_EN`（从机）置 1，以便 I2C 从机 I2C 从机在需要发送数据时可以把 SCL 拉低来给软件向 I2C 从机的 TX RAM 中写数提供时间，否则 I2C 从机需要在 I2C 主机开始传输前提前准备好数据。以下配置流程均按照 `I2C_SLAVE_SCL_STRETCH_EN`（从机）为 1 的情况进行。
3. 向 `I2C_CONF_UPGATE`（主机）和 `I2C_CONF_UPGATE`（从机）写 1 来同步寄存器。
4. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0（主机）	RSTART	—	—	—	—
I2C_COMMAND1（主机）	WRITE	0	0	1	2
I2C_COMMAND2（主机）	RSTART	—	—	—	—
I2C_COMMAND3（主机）	WRITE	0	0	1	1
I2C_COMMAND4（主机）	READ	0	0	1	N-1
I2C_COMMAND5（主机）	READ	1	0	1	1

I2C_COMMAND6 (主机)	STOP	—	—	—	—
-------------------	------	---	---	---	---

- 在 I2C_SLAVE_ADDR_REG (从机) 的 I2C_SLAVE_ADDR (从机) 设置 I2C 从机的 10 位 I2C 从机地址, 并将 I2C_ADDR_10BIT_EN (从机) 置 1 使能 10bit 寻址模式。
- 向 I2C 主机的 TX RAM 写入从机地址和要发送的数据, 第一个地址字节是 $((0x78 | I2C_SLAVE_ADDR[9:8]) \ll 1) | R/W$, R/W 位为 W (主机); 第二个地址字节是 I2C_SLAVE_ADDR[7:0]。第三个字节是重复的第一个地址字节加上 R/W 位, 其中 R/W 位为 R (从机)。可选 FIFO 方式和直接访问方式。
- 向 I2C_CONF_UPGATE (主机) 和 I2C_CONF_UPGATE (从机) 写 1 来同步寄存器。
- 向 I2C_TRANS_START (主机) 写 1 开始 I2C 主机的传输。
- 参考章节 23.4.14, 启动 I2C 从机的传输。
- I2C 从机比较 I2C 主机发送的从机地址和自己的 I2C_SLAVE_ADDR (从机), 当 I2C 主机 WRITE 命令中的 ack_check_en (主机) 配置为 1 时, I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0, 则不会对 ACK 检测, 会默认为匹配。
 - 匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平一致, 传输继续。
 - 不匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平不一致, I2C 主机产生 I2C_NACK_INT (主机) 中断, 停止发送数据并且产生 STOP。
- I2C 主机发送 RSTART 命令, 并发送 TX RAM 里的第三个字节, 即为重复的地址字节和 R 位。
- I2C 从机重复执行步骤 10, 若地址匹配, 继续后面的步骤。
- 等待 I2C_SLAVE_STRETCH_INT (从机), 读取 I2C_STRETCH_CAUSE 的值为 0, 此时 I2C 从机地址与 SDA 线上发送的地址相匹配, 且 I2C 从机要发送数据。
- 参考章节 23.4.10, 向 I2C 从机的 TX RAM 写入要发送的数据。可选 FIFO 方式和直接访问方式。
- 将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1, 释放 SCL 总线。
- I2C 从机发送数据, I2C 主机会根据当前 READ 指令对应的 ack_check_en (主机) 配置的不同进行或不进行 ACK 检测。
- 若 I2C 主机要读取的数超过 32 个字节, 当发送数据全部发完, 在 I2C 从机的 TX RAM 空时产生 I2C_SLAVE_STRETCH_INT (从机) 中断。此时 I2C 从机会将 SCL 拉低, 软件在此期间可以继续向 I2C 从机的 TX RAM 填充数据, 也可以从 I2C 主机的 RX RAM 读取数据。等完成操作后再将 I2C_SLAVE_STRETCH_INT_CLR (从机) 置 1 清除中断, 将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1 释放 SCL 总线。
- 当 I2C 主机接收最后一个数据时, 将 ack_value (主机) 设成 1, I2C 从机接收到 NACK 中断, 停止发送。
- 当整个传输正常结束, I2C 主机执行 STOP 命令, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.7 I2C 主机读取从机，7 位双寻址，单次命令序列

23.5.7.1 场景介绍

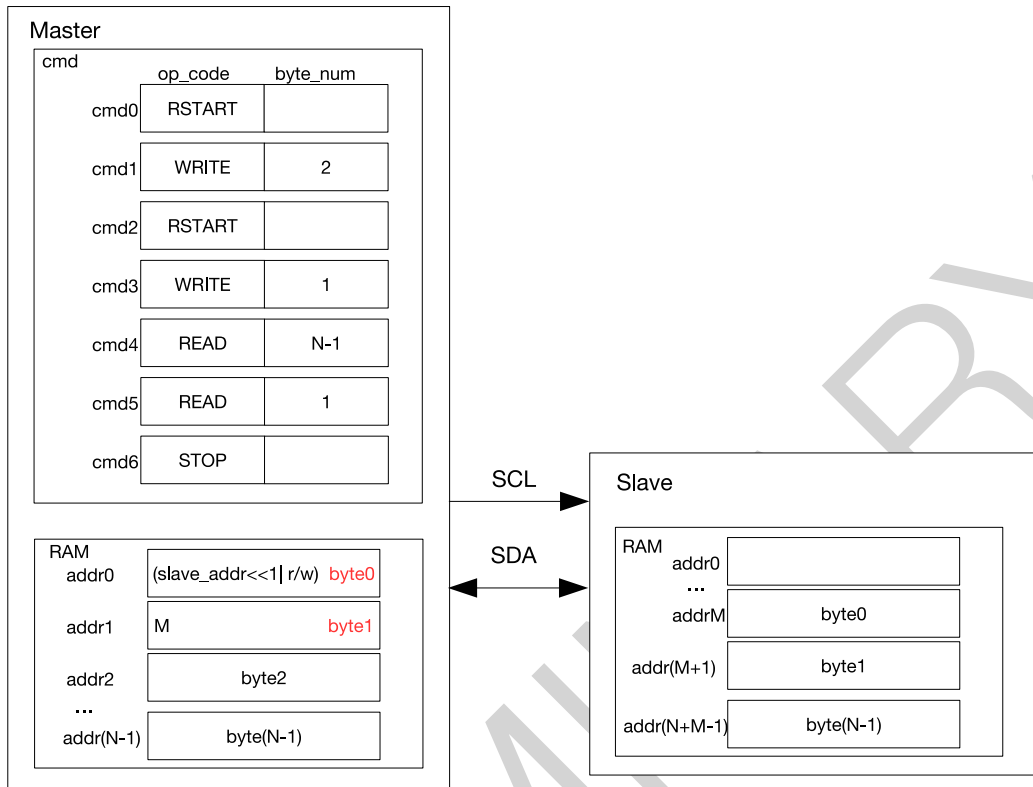


图 23-13. I2C 主机从 7 位寻址从机的 M 地址读取 N 个数据

图 23-13 为 I2C 主机从 I2C 从机中指定地址读取数据的值。I2C 主机在传输开始发送两个地址字节，第一个地址字节是 I2C 从机的 7 位地址加 R/W 位，R/W 位为 W（主机）；第二个地址字节是 I2C 从机的内存地址 M。之后再次发送 RSTART，并重复发送第一个地址字节，R/W 位变为 R（从机）。之后 I2C 主机从 I2C 从机的 AddrM 地址开始读取数据。

23.5.7.2 配置示例

1. 设置 `I2C_MS_MODE`（主机）为 1，`I2C_MS_MODE`（从机）为 0。
2. 推荐将 `I2C_SLAVE_SCL_STRETCH_EN`（从机）置 1，以便 I2C 从机 I2C 从机在需要发送数据时可以先将 SCL 拉低来给软件向 I2C 从机的 TX RAM 中写数提供时间，否则 I2C 从机需要在 I2C 主机开始传输前准备好数据。以下配置流程均按照 `I2C_SLAVE_SCL_STRETCH_EN`（从机）为 1 的情况进行。
3. 设置 `I2C_FIFO_ADDR_CFG_EN`（从机）为 1 来使能双寻址模式。
4. 向 `I2C_CONF_UPGATE`（主机）和 `I2C_CONF_UPGATE`（从机）写 1 来同步寄存器。
5. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0（主机）	RSTART	—	—	—	—
I2C_COMMAND1（主机）	WRITE	0	0	1	2
I2C_COMMAND2（主机）	RSTART	—	—	—	—

I2C_COMMAND3 (主机)	WRITE	0	0	1	1
I2C_COMMAND4 (主机)	READ	0	0	1	N-1
I2C_COMMAND5 (主机)	READ	1	0	1	1
I2C_COMMAND6 (主机)	STOP	—	—	—	—

6. 在 I2C_SLAVE_ADDR_REG(从机)的 I2C_SLAVE_ADDR(从机)设置 I2C 从机的 7 位地址, I2C_ADDR_10BIT_EN(从机)置 0 使能 7bit 寻址模式。
7. 参考章节 23.4.10, 向 I2C 主机的 TX RAM 写入从机地址和要发送的数据, 第一个地址字节是 (I2C_SLAVE_ADDR[6:0])«1) R/W, R/W 位为 W (主机); 第二个地址字节是 I2C 从机的内存地址 M。第三个字节是重复的第一个地址字节加上 R/W 位, 其中 R/W 位为 R (从机)。可选 FIFO 方式和直接访问方式。
8. 向 I2C_CONF_UPGATE (主机) 和 I2C_CONF_UPGATE (从机) 写 1 来同步寄存器。
9. 向 I2C_TRANS_START (主机) 写 1 开始 I2C 主机的传输。
10. 参考章节 23.4.14, 启动 I2C 从机的传输。
11. I2C 从机比较 I2C 主机发送的从机地址和自己的 I2C_SLAVE_ADDR (从机), 当 I2C 主机 WRITE 命令中的 ack_check_en (主机) 配置为 1 时, I2C 主机会在发送完每个字节之后进行 ACK 检测。若 ack_check_en 配置为 0, 则不会对 ACK 检测, 会默认为匹配。
 - 匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平一致, 传输继续。
 - 不匹配: 接收的 ACK 值与 WRITE 命令中的 ack_exp (主机) 电平不一致, I2C 主机产生 I2C_NACK_INT (主机) 中断, 停止发送数据并且产生 STOP。
12. I2C 从机接收到 I2C 主机发送的内存地址, 完成 TX RAM 的地址偏移。
13. I2C 主机发送 RSTART 命令, 并发送 TX RAM 里的第三个字节, 即为重复的地址字节和 R 位。
14. I2C 从机重复执行步骤 11, 若地址匹配, 继续后面的步骤。
15. 等待 I2C_SLAVE_STRETCH_INT (从机), 读取 I2C_STRETCH_CAUSE 的值为 0, 此时 I2C 从机地址与 SDA 线上发送的地址相匹配, 且 I2C 从机要发送数据。
16. 参考章节 23.4.10, 向 I2C 从机的 TX RAM 写入要发送的数据。可选 FIFO 方式和直接访问方式。
17. 将 I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1, 释放 SCL 总线。
18. I2C 从机发送数据, I2C 主机会根据当前 READ 指令对应的 ack_check_en (主机) 配置的不同进行或不进行 ACK 检测。
19. 若 I2C 主机要读取的数超过 32 个, 当发送数据全部发完, 在 I2C 从机的 TX RAM 空时产生 I2C_SLAVE_STRETCH_INT (从机) 中断。此时 I2C 从机将会将 SCL 拉低, 软件在此期间可以继续向 I2C 从机的 TX RAM 填充数据, 也可以从 I2C 主机的 RX RAM 读取数据。等完成操作后再将 I2C_SLAVE_STRETCH_INT_CLR (从机) 置 1, 清除中断; I2C_SLAVE_SCL_STRETCH_CLR (从机) 置 1, 释放 SCL 总线。
20. 当 I2C 主机接收最后一个数据时, 将 ack_value (主机) 设成 1, I2C 从机接收到 NACK 中断, 停止发送。
21. 当整个传输正常结束, I2C 主机执行 STOP 命令, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.5.8 I2C 主机读取从机，7 位寻址，多次命令序列

23.5.8.1 场景介绍

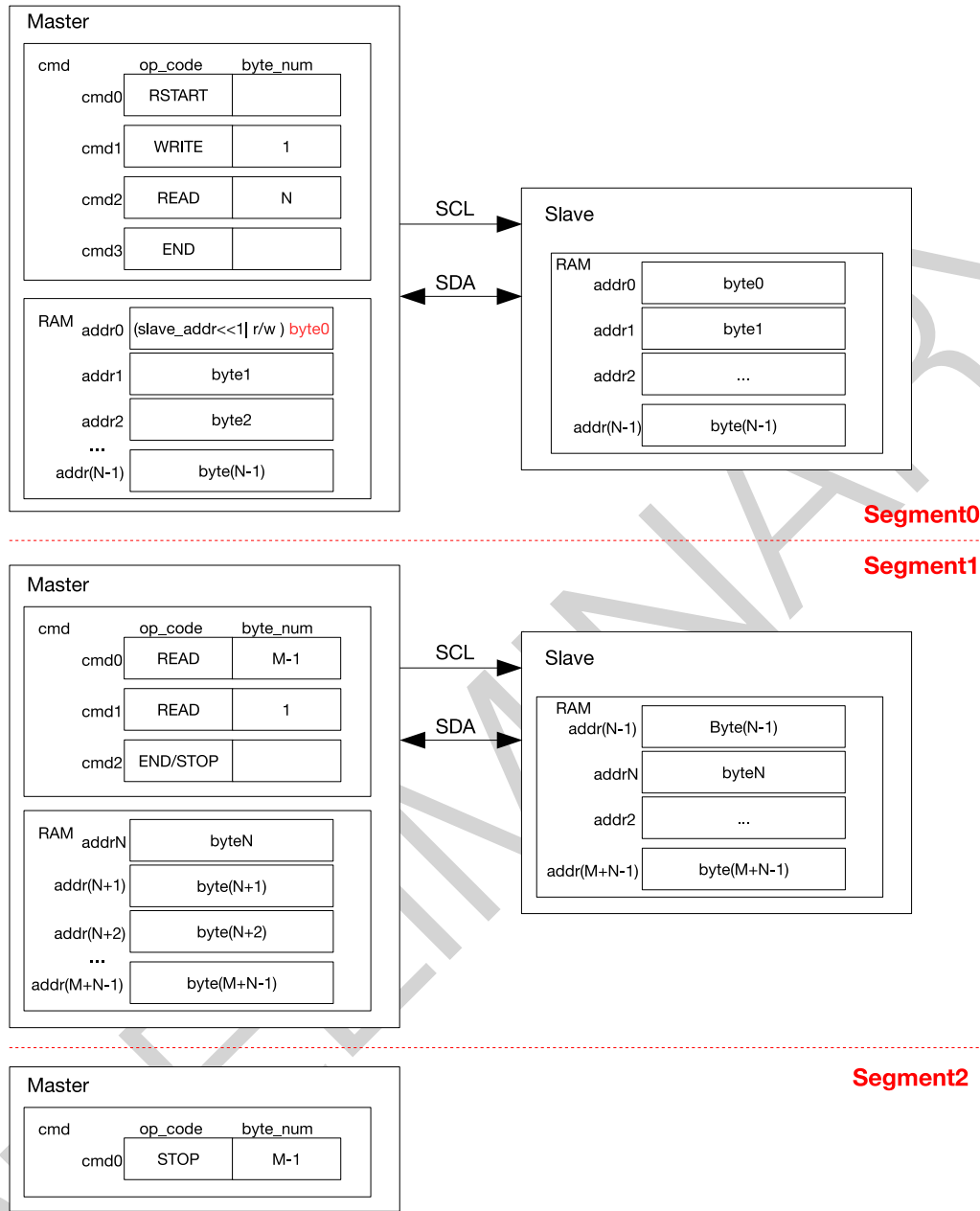


图 23-14. I2C 主机分段读 7 位寻址的从机

图 23-14 为 I2C 主机通过 END 命令分三段或者分两段，从 I2C 从机读取 N+M 个数据的流程图。

1. 第一段流程和章节 23-11 类似，只是最后一个命令变为 END。
2. 接着在 I2C 从机的 TX RAM 中准备好数据，置位 `I2C_TRANS_START`，当执行到 END 命令时，I2C 主机可以更新命令寄存器和 RAM 的内容，如第二段所示，并且清零其对应的 `I2C_END_DETECT_INT` 中断。当第二段中 cmd2 为 STOP 时，即两段读 I2C 从机，置位 `I2C_TRANS_START`，I2C 主机继续传输数据，最后发送 STOP 位来停止传输。
3. 当第二段中 cmd2 为 END 时，在 I2C 主机完成第二次数据传输，并检测到 I2C 主机的 `I2C_END_DETECT_INT`

中断后，配置 cmd 如第三段所示。置位 `I2C_TRANS_START`，I2C 主机发送 STOP 位停止传输。

23.5.8.2 配置示例

- 1. 设置 `I2C_MS_MODE`（主机）为 1，`I2C_MS_MODE`（从机）为 0。
- 2. 推荐将 `I2C_SLAVE_SCL_STRETCH_EN`（从机）置 1，以便 I2C 从机 I2C 从机在需要发送数据时可以先将 SCL 拉低来给软件向 I2C 从机的 TX RAM 中写数提供时间，否则 I2C 从机需要在 I2C 主机开始传输前提前准备好数据。以下配置流程均按照 `I2C_SLAVE_SCL_STRETCH_EN`（从机）为 1 的情况进行。
- 3. 向 `I2C_CONF_UPGATE`（主机）和 `I2C_CONF_UPGATE`（从机）写 1 来同步寄存器。
- 4. 配置 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
<code>I2C_COMMAND0</code> （主机）	RSTART	—	—	—	—
<code>I2C_COMMAND1</code> （主机）	WRITE	0	0	1	1
<code>I2C_COMMAND2</code> （主机）	READ	0	0	1	N
<code>I2C_COMMAND4</code> （主机）	END	—	—	—	—

- 5. 向 I2C 主机的 TX RAM 写入 I2C 从机地址，可选 FIFO 方式和直接访问方式。
- 6. 在 `I2C_SLAVE_ADDR_REG`（从机）的 `I2C_SLAVE_ADDR`（从机）设置 I2C 从机的地址。
- 7. 向 `I2C_CONF_UPGATE`（主机）和 `I2C_CONF_UPGATE`（从机）写 1 来同步寄存器。
- 8. 向 `I2C_TRANS_START`（主机）写 1 开始 I2C 主机的传输。
- 9. 参考章节 23.4.14，启动 I2C 从机的传输。
- 10. I2C 从机比较 I2C 主机发送的从机地址和自己的 `I2C_SLAVE_ADDR`（从机），当 I2C 主机 WRITE 命令中的 `ack_check_en`（主机）配置为 1 时，I2C 主机会在发送完每个字节之后进行 ACK 检测。若 `ack_check_en` 配置为 0，则不会对 ACK 检测，会默认为匹配。
 - 匹配：接收的 ACK 值与 WRITE 命令中的 `ack_exp`（主机）电平一致，传输继续。
 - 不匹配：接收的 ACK 值与 WRITE 命令中的 `ack_exp`（主机）电平不一致，I2C 主机产生 `I2C_NACK_INT`（主机）中断，停止发送数据并且产生 STOP。
- 11. 等待 `I2C_SLAVE_STRETCH_INT`（从机），读取 `I2C_STRETCH_CAUSE` 的值为 0，此时 I2C 从机地址与 SDA 线上发送的地址相匹配，且 I2C 从机要发送数据。
- 12. 参考章节 23.4.10，向 I2C 从机的 TX RAM 写入要发送的数据。可选 FIFO 方式和直接访问方式。
- 13. 将 `I2C_SLAVE_SCL_STRETCH_CLR`（从机）置 1，释放 SCL 总线。
- 14. I2C 从机发送数据，I2C 主机根据当前 READ 指令对应的 `ack_check_en`（主机）配置的不同进行或不进行 ACK 检测。
- 15. 若 I2C 主机一个 READ 指令要读取的数 N 或 M 超过 32 个字节，当 I2C 从机的 TX RAM 中发送数据全部发完，为空时产生 `I2C_SLAVE_STRETCH_INT`（从机）中断。此时 I2C 从机将会将 SCL 拉低，软件在此期间可以继续向 I2C 从机的 TX RAM 填充数据，也可以从 I2C 主机的 RX RAM 读取数据。等完成操作后再将 `I2C_SLAVE_STRETCH_INT_CLR`（从机）置 1 清除中断，将 `I2C_SLAVE_SCL_STRETCH_CLR`（从机）置 1 释放 SCL 总线。

16. 等到一次 READ 指令完成, I2C 主机执行 END 指令, I2C_END_DETECT_INT (主机) 中断产生后, 设置 I2C_END_DETECT_INT_CLR (主机) 为 1 来清除中断。
17. 更新 I2C 主机的指令寄存器, 有两种设置方式:

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0 (主机)	READ	ack_value	ack_exp	1	M
I2C_COMMAND1 (主机)	END	—	—	—	—

或者

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND0 (主机)	READ	0	0	1	M-1
I2C_COMMAND0 (主机)	READ	1	0	1	1
I2C_COMMAND1 (主机)	STOP	—	—	—	—

18. 向 I2C 从机的 TX RAM 写入 M 个字节要发送的数据, 若 M 大于 32, 重复步骤 12, 可以用 FIFO 方式或直接访问方式。
19. 向 I2C_TRANS_START (主机) 位写 1 开始传输, 并重复步骤 14 的流程。
20. 若最后一个指令为 STOP, 则当 I2C 主机接收最后一个数据时, 将 ack_value (主机) 设成 1, I2C 从机接收到 NACK 中断, 停止发送。I2C 主机执行 STOP 命令结束传输, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。
21. 若最后一个指令为 END, 则重复步骤 16, 并在完成后继续下面的步骤。
22. 更新 I2C 主机的指令寄存器。

指令寄存器	op_code	ack_value	ack_exp	ack_check_en	byte_num
I2C_COMMAND1 (主机)	STOP	—	—	—	—

23. 向 I2C_TRANS_START (主机) 位写 1 开始传输。
24. I2C 主机执行 STOP 命令结束传输, 并产生 I2C_TRANS_COMPLETE_INT (主机) 中断。

23.6 中断

- I2C_SLAVE_STRETCH_INT: 从机模式下, 当出现可触发时钟拉伸的事件时, 产生此中断。
- I2C_DET_START_INT: 主机或从机检测到 I2C START 位时, 触发此中断。
- I2C_SCL_MAIN_ST_TO_INT: 当 I2C 主状态机 SCL_MAIN_FSM 保持某个状态超过 I2C_SCL_MAIN_ST_TO_I2C[23:0] 个模块时钟周期时, 触发此中断。
- I2C_SCL_ST_TO_INT: 当 I2C 状态机 SCL_FSM 保持某个状态超过 I2C_SCL_ST_TO_I2C[23:0] 个模块时钟周期时, 触发此中断。
- I2C_RXFIFO_UDF_INT: 当 I2C 通过 APB 总线读取 RX FIFO, 但 RX FIFO 为空时, 触发该中断。
- I2C_TXFIFO_OVF_INT: 当 I2C 通过 APB 总线写 TX FIFO, 但 TX FIFO 为满时, 触发该中断。
- I2C_NACK_INT: 当 I2C 配置为主机时, 接收到的 ACK 与命令中期望的 ACK 值不一致时, 即触发该中断; 当 I2C 配置为从机时, 接收到的 ACK 值为 1 时即触发该中断。

- I2C_TRANS_START_INT: 当 I2C 发送一个 START 位时，即触发该中断。
- I2C_TIME_OUT_INT: 在传输过程中，当 I2C SCL 保持为高或为低电平的时间超过 [I2C_TIME_OUT_VALUE](#) 个模块时钟后，即触发该中断。
- I2C_TRANS_COMPLETE_INT: 当 I2C 检测到 STOP 位时，即触发该中断。
- I2C_MST_TXFIFO_UDF_INT: 当 I2C 主机的 TX FIFO 下溢时，触发此中断。
- I2C_ARBITRATION_LOST_INT: 当 I2C 主机的 SCL 为高电平，SDA 输出值与输入值不相等时，即触发该中断。
- I2C_BYTE_TRANS_DONE_INT: 当 I2C 发送或接收一个字节，即触发该中断。
- I2C_END_DETECT_INT: 当 I2C 主机命令的 op_code 为 END，且检测到 I2C END 状态时，触发此中断。
- I2C_RXFIFO_OVF_INT: 当 I2C RX FIFO 上溢时，触发此中断。
- I2C_TXFIFO_WM_INT: I2C TX FIFO 水标中断。当 [I2C_FIFO_PRT_EN](#) 为 1，且 TX FIFO 指针小于 [I2C_TXFIFO_WM_THRHD](#)[4:0] 时，触发此中断。
- I2C_RXFIFO_WM_INT: I2C RX FIFO 水标中断。当 [I2C_FIFO_PRT_EN](#) 为 1，且 RX FIFO 指针大于 [I2C_RXFIFO_WM_THRHD](#)[4:0] 时，触发此中断。

23.7 寄存器列表

本小节的所有地址均为相对于 **I2C 控制器** 基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
时序寄存器			
I2C_SCL_LOW_PERIOD_REG	配置 SCL 的低电平宽度	0x0000	R/W
I2C_SDA_HOLD_REG	配置 SCL 下降沿后的保持时间	0x0030	R/W
I2C_SDA_SAMPLE_REG	配置 SCL 上升沿后的采样时间	0x0034	R/W
I2C_SCL_HIGH_PERIOD_REG	配置 SCL 时钟的高电平宽度	0x0038	R/W
I2C_SCL_START_HOLD_REG	配置 START 命令产生时 SDA 下降沿和 SCL 下降沿之间的间隔时间	0x0040	R/W
I2C_SCL_RSTART_SETUP_REG	配置 SCL 上升沿和 SDA 下降沿之间的延迟	0x0044	R/W
I2C_SCL_STOP_HOLD_REG	配置 STOP 命令生成时 SCL 边沿的延迟	0x0048	R/W
I2C_SCL_STOP_SETUP_REG	配置 STOP 命令生成时 SDA 和 SCL 上升沿之间的间隔时间	0x004C	R/W
I2C_SCL_ST_TIME_OUT_REG	SCL 状态超时寄存器	0x0078	R/W
I2C_SCL_MAIN_ST_TIME_OUT_REG	SCL 主要状态超时寄存器	0x007C	R/W
配置寄存器			
I2C_CTR_REG	传输配置寄存器	0x0004	varies
I2C_TO_REG	超时控制寄存器	0x000C	R/W
I2C_SLAVE_ADDR_REG	从机地址配置寄存器	0x0010	R/W
I2C_FIFO_CONF_REG	FIFO 配置寄存器	0x0018	R/W
I2C_FILTER_CFG_REG	SCL 和 SDA 滤波配置寄存器	0x0050	R/W
I2C_CLK_CONF_REG	I2C 时钟配置寄存器	0x0054	R/W
I2C_SCL_SP_CONF_REG	电源配置寄存器	0x0080	varies
I2C_SCL_STRETCH_CONF_REG	配置 I2C 从机 SCL 时钟拉伸	0x0084	varies
状态寄存器			
I2C_SR_REG	描述 I2C 的工作状态	0x0008	RO
I2C_FIFO_ST_REG	FIFO 状态寄存器	0x0014	RO
I2C_DATA_REG	存储 RX FIFO 数据	0x001C	RO
中断寄存器			
I2C_INT_RAW_REG	原始中断状态	0x0020	R/ SS/ WTC
I2C_INT_CLR_REG	中断清除位	0x0024	WT
I2C_INT_ENA_REG	中断使能位	0x0028	R/W
I2C_INT_STATUS_REG	捕捉 I2C 通信事件的状态	0x002C	RO
命令寄存器			
I2C_COMD0_REG	I2C 命令寄存器 0	0x0058	varies
I2C_COMD1_REG	I2C 命令寄存器 1	0x005C	varies
I2C_COMD2_REG	I2C 命令寄存器 2	0x0060	varies
I2C_COMD3_REG	I2C 命令寄存器 3	0x0064	varies
I2C_COMD4_REG	I2C 命令寄存器 4	0x0068	varies
I2C_COMD5_REG	I2C 命令寄存器 5	0x006C	varies

名称	描述	地址	访问
I2C_COMD6_REG	I2C 命令寄存器 6	0x0070	varies
I2C_COMD7_REG	I2C 命令寄存器 7	0x0074	varies
版本寄存器			
I2C_DATE_REG	版本控制寄存器	0x00F8	R/W

23.8 寄存器

本小节的所有地址均为相对于 **I2C 控制器** 基地址的地址偏移量（相对地址），具体基地址请见章节 [3 系统和存储器](#) 中的表 3-4。

Register 23.1. I2C_SCL_LOW_PERIOD_REG (0x0000)

(reserved)																								I2C_SCL_LOW_PERIOD																							
31																								9								8								0							
0 0																								0								Reset															

I2C_SCL_LOW_PERIOD 用于配置主机模式下 SCL 低电平的保持时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.2. I2C_SDA_HOLD_REG (0x0030)

(reserved)																I2C_SDA_HOLD_TIME									
31																9	8	0							
0 0																0								Reset	

I2C_SDA_HOLD_TIME 用于配置 SCL 下降沿后的数据保持时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.3. I2C_SDA_SAMPLE_REG (0x0034)

(reserved)																								I2C_SDA_SAMPLE_TIME																							
31																								9								8								0							
0 0																								0								Reset															

I2C_SDA_SAMPLE_TIME 用于配置采样 SDA 的时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.4. I2C_SCL_HIGH_PERIOD_REG (0x0038)

(reserved)																I2C_SCL_WAIT_HIGH_PERIOD								I2C_SCL_HIGH_PERIOD																															
31																16								15								9								8								0							
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0								0								0																							
																																																Reset							

I2C_SCL_HIGH_PERIOD 用于配置 SCL 在主机模式下保持高电平的时间，以 I2C 控制器时钟周期数为单位。(R/W)

I2C_SCL_WAIT_HIGH_PERIOD 用于配置 SCL_FSM 等待 SCL 在主机模式下翻转至高电平的时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.5. I2C_SCL_START_HOLD_REG (0x0040)

(reserved)																I2C_SCL_START_HOLD_TIME																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
31																9	8					0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2C_SCL_START_HOLD_TIME 配置 START 命令产生时 SDA 下降沿和 SCL 下降沿的间隔时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.6. I2C_SCL_RSTART_SETUP_REG (0x0044)

(reserved)																I2C_SCL_RSTART_SETUP_TIME									
31																9	8	0							
0 0																8								Reset	

I2C_SCL_RSTART_SETUP_TIME 配置 RSTART 命令产生时 SCL 上升沿和 SDA 下降沿的间隔时间，以 I2C 控制器的时钟周期数为单位。(R/W)

Register 23.7. I2C_SCL_STOP_HOLD_REG (0x0048)

(reserved)																								I2C_SCL_STOP_HOLD_TIME																							
31																								9								8								0							
0 0																								8								Reset															

I2C_SCL_STOP_HOLD_TIME 配置 STOP 命令后的延迟，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.8. I2C_SCL_STOP_SETUP_REG (0x004C)

(reserved)																I2C_SCL_STOP_SETUP_TIME																	
31																9	8	0															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	Reset	

I2C_SCL_STOP_SETUP_TIME 配置 SCL 上升沿和 SDA 上升沿的间隔时间，以 I2C 控制器时钟周期数为单位。(R/W)

Register 23.9. I2C_SCL_ST_TIME_OUT_REG (0x0078)

(reserved)																															I2C_SCL_ST_TO_I2C																
31																									5						4	0															
0 0																									0x10						Reset																

I2C_SCL_ST_TO_I2C SCL_FSM 状态不变的最大时间，不能大于 23。(R/W)

411
反馈文档意见

ESP32-C3 TRM (预发布 v0.4)

I2C_SCL_MAIN_ST_TO_I2C SCL_MAIN_FSM 状态不变的最大时间，不能大于 23。(R/W)

Register 23.11. I2C_CTR_REG (0x0004)

(reserved)																I2C_ADDR_BROADCASTING_EN I2C_ADDR_10BIT_RW_CHECK_EN I2C_SLV_TX_AUTO_START_EN I2C_CONF_UPGATE I2C_FSM_RST I2C_ARBITRATION_EN I2C_CLK_EN I2C_RX_LSB_FIRST I2C_TX_LSB_FIRST I2C_TRANS_START I2C_MS_MODE I2C_RX_FULL_ACK_LEVEL I2C_SAMPLE_SCL_LEVEL I2C_SCL_FORCE_OUT I2C_SDA_FORCE_OUT															
31																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	Reset

- I2C_SDA_FORCE_OUT** 0: 直接输出; 1: 漏极开路输出。(R/W)
- I2C_SCL_FORCE_OUT** 0: 直接输出; 1: 漏极开路输出。(R/W)
- I2C_SAMPLE_SCL_LEVEL** 用于选择采样模式。0: SCL 为高电平时采样 SDA 数据; 1: SCL 为低电平时采样 SDA 数据。(R/W)
- I2C_RX_FULL_ACK_LEVEL** 用于配置主机在 I2C_RXFIFO_CNT 达到阈值时需发送的 ACK 电平值。(R/W)
- I2C_MS_MODE** 置位此位, 将 I2C 控制器配置为主机。清零此位, 将 I2C 控制器配置为从机。(R/W)
- I2C_TRANS_START** 置位此位, 开始发送 TX FIFO 中的数据。(WT)
- I2C_TX_LSB_FIRST** 用于控制待发送数据的发送顺序。0: 从最高有效位开始发送数据; 1: 从最低有效位开始发送数据。(R/W)
- I2C_RX_LSB_FIRST** 用于控制接收数据的存储顺序。0: 从最高有效位开始接收数据; 1: 从最低有效位开始接收数据。(R/W)
- I2C_CLK_EN** 用于控制 APB_CLK 时钟门控。0: APB_CLK 时钟门控使能, 以便节能; 1: APB_CLK 时钟一直开启。(R/W)
- I2C_ARBITRATION_EN** I2C 总线仲裁的使能位。(R/W)
- I2C_FSM_RST** 用于复位 SCL_FSM。(WT)
- I2C_CONF_UPGATE** 同步位。(WT)
- I2C_SLV_TX_AUTO_START_EN** 从机自动发送数据的使能位。(R/W)
- I2C_ADDR_10BIT_RW_CHECK_EN** 使能 10 位寻址模式的读写标志位检查功能, 检查读写标志是否符合协议。(R/W)
- I2C_ADDR_BROADCASTING_EN** 使能 7 位寻址模式的广播功能。(R/W)

Register 23.12. I2C_TO_REG (0x000C)

(reserved)																										I2C_TIME_OUT_EN		I2C_TIME_OUT_VALUE		
31																										6	5	4	0	
0 0																										0	0x10		Reset	

I2C_TIME_OUT_VALUE 用于配置接收一位数据的超时时间，以 APB 时钟周期为单位。(R/W)

I2C_TIME_OUT_EN 超时控制使能位。(R/W)

Register 23.13. I2C_SLAVE_ADDR_REG (0x0010)

I2C_ADDR_10BIT_EN																(reserved)																I2C_SLAVE_ADDR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
31	30															15	14															0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2C_SLAVE_ADDR I2C 控制器配置为从机时，该字段用于配置从机地址。(R/W)

I2C_ADDR_10BIT_EN 用于在主机模式下使能从机的 10 位寻址模式。(R/W)

Register 23.14. I2C_FIFO_CONF_REG (0x0018)

(reserved)																I2C_FIFO_PRT_EN					I2C_TX_FIFO_RST		I2C_RX_FIFO_RST		I2C_FIFO_ADDR_CFG_EN		I2C_NONFIFO_EN		I2C_TXFIFO_WM_THRHD		I2C_RXFIFO_WM_THRHD	
31																15	14	13	12	11	10	9	5		4	0						
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																1	0	0	0	0	0x4		0xb		Reset							

Reset

I2C_RXFIFO_WM_THRHD 直接访问模式下, RX FIFO 的水标阈值。I2C_FIFO_PRT_EN 为 1 且 RX FIFO 计数值大于 I2C_TXFIFO_WM_THRHD[4:0] 时, I2C_TXFIFO_WM_INT_RAW 位有效。(R/W)

I2C_TXFIFO_WM_THRHD 直接访问模式下, TX FIFO 的水标阈值。I2C_FIFO_PRT_EN 为 1 且 TX FIFO 计数值小于 I2C_TXFIFO_WM_THRHD[4:0] 时, I2C_TXFIFO_WM_INT_RAW 位有效。(R/W)

I2C_NONFIFO_EN 置位此位, 使能 APB 直接访问。(R/W)

I2C_FIFO_ADDR_CFG_EN 此位置 1 时, 从机接收地址字节的后一个字节为从机 RAM 中的偏移地址。(R/W)

I2C_RX_FIFO_RST 置位此位, 复位 RX FIFO。(R/W)

I2C_TX_FIFO_RST 置位此位, 复位 TX FIFO。(R/W)

I2C_FIFO_PRT_EN 直接访问模式下 FIFO 指针的控制使能位。该位控制 TX FIFO 和 RX FIFO 溢出、下溢、为满、为空时的有效位和中断。(R/W)

Register 23.15. I2C_FILTER_CFG_REG (0x0050)

(reserved)																								I2C_SDA_FILTER_EN I2C_SCL_FILTER_EN				I2C_SDA_FILTER_THRES I2C_SCL_FILTER_THRES								
31																								10	9	8	7	4				3	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0				0				Reset			

Reset

I2C_SCL_FILTER_THRES SCL 输入信号的脉冲宽度小于该字段的值时, I2C 控制器忽略此脉冲。该寄存器的值以 I2C 控制器时钟周期数为单位。(R/W)

I2C_SDA_FILTER_THRES SDA 输入信号的脉冲宽度小于该字段的值时, I2C 控制器忽略此脉冲。该寄存器的值以 I2C 控制器时钟周期数为单位。(R/W)

I2C_SCL_FILTER_EN SCL 的滤波使能位。(R/W)

I2C_SDA_FILTER_EN SDA 的滤波使能位。(R/W)

Register 23.16. I2C_CLK_CONF_REG (0x0054)

(reserved)										I2C_SCLK_ACTIVE I2C_SCLK_SEL		I2C_SCLK_DIV_B		I2C_SCLK_DIV_A		I2C_SCLK_DIV_NUM	
31	22	21	20	19	14	13	8	7	0								
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Reset

I2C_SCLK_DIV_NUM 分频系数的整数部分。(R/W)

I2C_SCLK_DIV_A 分频系数小数部分的分子。(R/W)

I2C_SCLK_DIV_B 分频系数小数部分的分母。(R/W)

I2C_SCLK_SEL 选择 I2C 控制器的时钟源。0: XTAL_CLK; 1: FOSC_CLK。(R/W)

I2C_SCLK_ACTIVE I2C 控制器的时钟开关。(R/W)

Register 23.17. I2C_SCL_SP_CONF_REG (0x0080)

(reserved)																I2C_SDA_PD_EN I2C_SCL_PD_EN		I2C_SCL_RST_SLV_NUM		I2C_SCL_RST_SLV_EN	
31	8	7	6	5	1	0															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset

I2C_SCL_RST_SLV_EN I2C 主机处于空闲状态时，置位此位发送 SCL 脉冲。脉冲数量为 I2C_SCL_RST_SLV_NUM[4:0]。(R/W/SC)

I2C_SCL_RST_SLV_NUM 配置主机模式下生成的 SCL 脉冲。I2C_SCL_RST_SLV_EN 为 1 时有效。(R/W)

I2C_SCL_PD_EN 降低 I2C SCL 输出功耗的使能位。0: 正常工作; 1: 不工作，降低功耗。将 I2C_SCL_FORCE_OUT 和 I2C_SCL_PD_EN 置 1 拉伸 SCL。(R/W)

I2C_SDA_PD_EN 降低 I2C SDA 输出功耗的使能位。0: 正常工作; 1: 不工作，降低功耗。将 I2C_SDA_FORCE_OUT 和 I2C_SDA_PD_EN 置 1 拉伸 SDA。(R/W)

Register 23.18. I2C_SCL_STRETCH_CONF_REG (0x0084)

(reserved)														I2C_SLAVE_BYTE_ACK_LVL I2C_SLAVE_BYTE_ACK_CTL_EN I2C_SLAVE_SCL_STRETCH_CLR I2C_SLAVE_SCL_STRETCH_EN				I2C_STRETCH_PROTECT_NUM																												
31														14														13	12	11	10	9	0													
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0														0														0	0	0	0	0														Reset

I2C_STRETCH_PROTECT_NUM 配置 SCL 时钟拉伸释放后的保护时间，通常设置为大于 SDA 的建立时间即可。(R/W)

I2C_SLAVE_SCL_STRETCH_EN SCL 时钟拉伸的使能位。0：关闭；1：使能。
I2C_SLAVE_SCL_STRETCH_EN 为 1，且出现可触发时钟拉伸的事件时，拉伸 SCL 时钟。
SCL 时钟拉伸的原因可见 I2C_STRETCH_CAUSE。(R/W)

I2C_SLAVE_SCL_STRETCH_CLR 置位此位，清除 CL 时钟拉伸。(WT)

I2C_SLAVE_BYTE_ACK_CTL_EN 使能从机控制 ACK 电平。(R/W)

I2C_SLAVE_BYTE_ACK_LVL I2C_SLAVE_BYTE_ACK_CTL_EN 置 1 时，设置 ACK 的电平值。(R/W)

Register 23.19. I2C_SR_REG (0x0008)

(reserved)		I2C_SCL_STATE_LAST		(reserved)		I2C_SCL_MAIN_STATE_LAST		I2C_TXFIFO_CNT		(reserved)		I2C_STRETCH_CAUSE		I2C_RXFIFO_CNT		(reserved)		I2C_SLAVE_ADDRESSED		I2C_BUS_BUSY		I2C_ARB_LOST		(reserved)		I2C_SLAVE_RW		I2C_RESP_REC	
31	30	28	27	26	24	23	18	17	16	15	14	13	8	7	6	5	4	3	2	1	0								
0	0	0	0	0	0	0	0	0	0x3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset																													

I2C_RESP_REC 主机模式或从机模式下接收的 ACK 电平值。0: ACK; 1: NACK。(RO)

I2C_SLAVE_RW 从机模式下, 0: 主机向从机写入数据; 1: 主机读取从机数据。(RO)

I2C_ARB_LOST I2C 控制器不控制 SCL 线时, 该寄存器变为 1。(RO)

I2C_BUS_BUSY 0: I2C 总线处于空闲状态; 1: I2C 总线正在传输数据。(RO)

I2C_SLAVE_ADDRESSED 配置成 I2C 从机、且主机发送地址与从机地址匹配时, 该位翻转为高电平。(RO)

I2C_RXFIFO_CNT 该字段为需发送数据的字节数。(RO)

I2C_STRETCH_CAUSE 从机模式下 SCL 时钟拉伸的原因。0: 主机开始读取数据时拉伸 SCL 时钟; 1: 从机模式下 I2C TX FIFO 读空时拉伸 SCL 时钟; 2: 从机模式下 I2C RX FIFO 写满时拉伸 SCL 时钟。(RO)

I2C_TXFIFO_CNT 该字段存储 RAM 接收数据的字节数。(RO)

I2C_SCL_MAIN_STATE_LAST 该字段为 I2C 控制器状态机的状态。0: 空闲; 1: 地址偏移; 2: ACK 地址; 3: 接收数据; 4: 发送数据; 5: 发送 ACK; 6: 等待 ACK (RO)

I2C_SCL_STATE_LAST 该字段为生成 SCL 的状态机状态。0: 空闲状态; 1: 开始; 2: 下降沿; 3: 低电平; 4: 上升沿; 5: 高电平; 6: 停止 (RO)

Register 23.20. I2C_FIFO_ST_REG (0x0014)

(reserved)			I2C_SLAVE_RW_POINT										(reserved)			I2C_TXFIFO_WADDR										I2C_TXFIFO_RADDR										I2C_RXFIFO_WADDR										I2C_RXFIFO_RADDR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
31	30	29																				22	21	20	19																				15	14																				10	9																				5	4																				0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
0	0	0																											0	0	0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																											0																										

- I2C_RXFIFO_RADDR** APB 总线读 RX FIFO 的偏移地址。(RO)
- I2C_RXFIFO_WADDR** I2C 控制器接收数据和写 RX FIFO 的偏移地址。(RO)
- I2C_TXFIFO_RADDR** I2C 控制器读 TX FIFO 的偏移地址。(RO)
- I2C_TXFIFO_WADDR** APB 总线写 TX FIFO 的偏移地址。(RO)
- I2C_SLAVE_RW_POINT** 从机模式下接收的数据。(RO)

Register 23.21. I2C_DATA_REG (0x001C)

(reserved)																I2C_FIFO_RDATA											
31																8	7	0									
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0										Reset	

- I2C_FIFO_RDATA** 从 RX FIFO 读取的数据。(RO)

Register 23.22. I2C_INT_RAW_REG (0x0020)

(reserved)																																I2C_GENERAL_CALL_INT_RAW	I2C_SLAVE_STRETCH_INT_RAW	I2C_DET_START_INT_RAW	I2C_SCL_MAIN_ST_TO_INT_RAW	I2C_SCL_ST_TO_INT_RAW	I2C_RXFIFO_UDF_INT_RAW	I2C_TXFIFO_OVF_INT_RAW	I2C_NACK_INT_RAW	I2C_TRANS_START_INT_RAW	I2C_TIME_OUT_INT_RAW	I2C_TRANS_COMPLETE_INT_RAW	I2C_MST_TXFIFO_UDF_INT_RAW	I2C_ARBITRATION_LOST_INT_RAW	I2C_BYTE_TRANS_DONE_INT_RAW	I2C_END_DETECT_INT_RAW	I2C_RXFIFO_WM_INT_RAW	I2C_TXFIFO_WM_INT_RAW
31																		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Reset																		

I2C_RXFIFO_WM_INT_RAW I2C_RXFIFO_WM_INT 的原始中断位。(R/SS/WTC)

I2C_TXFIFO_WM_INT_RAW I2C_TXFIFO_WM_INT 的原始中断位。(R/SS/WTC)

I2C_RXFIFO_OVF_INT_RAW I2C_RXFIFO_OVF_INT 的原始中断位。(R/SS/WTC)

I2C_END_DETECT_INT_RAW I2C_END_DETECT_INT 的原始中断位。(R/SS/WTC)

I2C_BYTE_TRANS_DONE_INT_RAW I2C_END_DETECT_INT 的原始中断位。(R/SS/WTC)

I2C_ARBITRATION_LOST_INT_RAW I2C_ARBITRATION_LOST_INT 的原始中断位。(R/SS/WTC)

I2C_MST_TXFIFO_UDF_INT_RAW I2C_TRANS_COMPLETE_INT 的原始中断位。(R/SS/WTC)

I2C_TRANS_COMPLETE_INT_RAW I2C_TRANS_COMPLETE_INT 的原始中断位。(R/SS/WTC)

I2C_TIME_OUT_INT_RAW I2C_TIME_OUT_INT 的原始中断位。(R/SS/WTC)

I2C_TRANS_START_INT_RAW I2C_TRANS_START_INT 的原始中断位。(R/SS/WTC)

I2C_NACK_INT_RAW I2C_SLAVE_STRETCH_INT 的原始中断位。(R/SS/WTC)

I2C_TXFIFO_OVF_INT_RAW I2C_TXFIFO_OVF_INT 的原始中断位。(R/SS/WTC)

I2C_RXFIFO_UDF_INT_RAW I2C_RXFIFO_UDF_INT 的原始中断位。(R/SS/WTC)

I2C_SCL_ST_TO_INT_RAW I2C_SCL_ST_TO_INT 的原始中断位。(R/SS/WTC)

I2C_SCL_MAIN_ST_TO_INT_RAW I2C_SCL_MAIN_ST_TO_INT 的原始中断位。(R/SS/WTC)

I2C_DET_START_INT_RAW I2C_DET_START_INT 的原始中断位。(R/SS/WTC)

I2C_SLAVE_STRETCH_INT_RAW I2C_SLAVE_STRETCH_INT 的原始中断位。(R/SS/WTC)

I2C_GENERAL_CALL_INT_RAW I2C_GENARAL_CALL_INT 的原始中断位。(R/SS/WTC)

Register 23.23. I2C_INT_CLR_REG (0x0024)

(reserved)																																I2C_GENERAL_CALL_INT_CLR I2C_SLAVE_STRETCH_INT_CLR I2C_DET_START_INT_CLR I2C_SCL_MAIN_ST_TO_INT_CLR I2C_SCL_ST_TO_INT_CLR I2C_RXFIFO_UDF_INT_CLR I2C_TXFIFO_UDF_INT_CLR I2C_NACK_INT_CLR I2C_TRANS_START_INT_CLR I2C_TIME_OUT_INT_CLR I2C_TRANS_COMPLETE_INT_CLR I2C_MST_TXFIFO_UDF_INT_CLR I2C_BYTE_TRANS_DONE_INT_CLR I2C_END_DETECT_INT_CLR I2C_RXFIFO_OVF_INT_CLR I2C_RXFIFO_WM_INT_CLR																			
31																18																17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0																0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- I2C_RXFIFO_WM_INT_CLR 置位此位，清除 I2C_RXFIFO_WM_INT 中断。(WT)
- I2C_TXFIFO_WM_INT_CLR 置位此位，清除 I2C_TXFIFO_WM_INT 中断。(WT)
- I2C_RXFIFO_OVF_INT_CLR 置位此位，清除 I2C_RXFIFO_OVF_INT 中断。(WT)
- I2C_END_DETECT_INT_CLR 置位此位，清除 I2C_END_DETECT_INT 中断。(WT)
- I2C_BYTE_TRANS_DONE_INT_CLR 置位此位，清除 I2C_END_DETECT_INT 中断。(WT)
- I2C_ARBTRATION_LOST_INT_CLR 置位此位，清除 I2C_ARBTRATION_LOST_INT 中断。(WT)
- I2C_MST_TXFIFO_UDF_INT_CLR 置位此位，清除 I2C_TRANS_COMPLETE_INT 中断。(WT)
- I2C_TRANS_COMPLETE_INT_CLR 置位此位，清除 I2C_TRANS_COMPLETE_INT 中断。(WT)
- I2C_TIME_OUT_INT_CLR 置位此位，清除 I2C_TIME_OUT_INT 中断。(WT)
- I2C_TRANS_START_INT_CLR 置位此位，清除 I2C_TRANS_START_INT 中断。(WT)
- I2C_NACK_INT_CLR 置位此位，清除 I2C_SLAVE_STRETCH_INT 中断。(WT)
- I2C_TXFIFO_OVF_INT_CLR 置位此位，清除 I2C_TXFIFO_OVF_INT 中断。(WT)
- I2C_RXFIFO_UDF_INT_CLR 置位此位，清除 I2C_RXFIFO_UDF_INT 中断。(WT)
- I2C_SCL_ST_TO_INT_CLR 置位此位，清除 I2C_SCL_ST_TO_INT 中断。(WT)
- I2C_SCL_MAIN_ST_TO_INT_CLR 置位此位，清除 I2C_SCL_MAIN_ST_TO_INT 中断。(WT)
- I2C_DET_START_INT_CLR 置位此位，清除 I2C_DET_START_INT 中断。(WT)
- I2C_SLAVE_STRETCH_INT_CLR 置位此位，清除 I2C_SLAVE_STRETCH_INT 中断。(WT)
- I2C_GENERAL_CALL_INT_CLR 置位此位，清除 I2C_GENARAL_CALL_INT 中断。(WT)

Register 23.24. I2C_INT_ENA_REG (0x0028)

(reserved)																															I2C_GENERAL_CALL_INT_ENA	I2C_SLAVE_STRETCH_INT_ENA	I2C_DET_START_INT_ENA	I2C_SCL_MAIN_ST_TO_INT_ENA	I2C_SCL_ST_TO_INT_ENA	I2C_RXFIFO_UDF_INT_ENA	I2C_TXFIFO_UDF_INT_ENA	I2C_NACK_INT_ENA	I2C_TRANS_START_INT_ENA	I2C_TIME_OUT_INT_ENA	I2C_TRANS_COMPLETE_INT_ENA	I2C_MST_TXFIFO_UDF_INT_ENA	I2C_ARBTRATION_LOST_INT_ENA	I2C_BYTE_TRANS_DONE_INT_ENA	I2C_END_DETECT_INT_ENA	I2C_RXFIFO_OVF_INT_ENA	I2C_RXFIFO_WM_INT_ENA
31																		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset																

- I2C_RXFIFO_WM_INT_ENA** I2C_RXFIFO_WM_INT 的使能位。(R/W)
- I2C_TXFIFO_WM_INT_ENA** I2C_TXFIFO_WM_INT 的使能位。(R/W)
- I2C_RXFIFO_OVF_INT_ENA** I2C_RXFIFO_OVF_INT 的使能位。(R/W)
- I2C_END_DETECT_INT_ENA** I2C_END_DETECT_INT 的使能位。(R/W)
- I2C_BYTE_TRANS_DONE_INT_ENA** I2C_END_DETECT_INT 的使能位。(R/W)
- I2C_ARBTRATION_LOST_INT_ENA** I2C_ARBTRATION_LOST_INT 的使能位。(R/W)
- I2C_MST_TXFIFO_UDF_INT_ENA** I2C_TRANS_COMPLETE_INT 的使能位。(R/W)
- I2C_TRANS_COMPLETE_INT_ENA** I2C_TRANS_COMPLETE_INT 的使能位。(R/W)
- I2C_TIME_OUT_INT_ENA** I2C_TIME_OUT_INT 的使能位。(R/W)
- I2C_TRANS_START_INT_ENA** I2C_TRANS_START_INT 的使能位。(R/W)
- I2C_NACK_INT_ENA** I2C_SLAVE_STRETCH_INT 的使能位。(R/W)
- I2C_TXFIFO_OVF_INT_ENA** I2C_TXFIFO_OVF_INT 的使能位。(R/W)
- I2C_RXFIFO_UDF_INT_ENA** I2C_RXFIFO_UDF_INT 的使能位。(R/W)
- I2C_SCL_ST_TO_INT_ENA** I2C_SCL_ST_TO_INT 的使能位。(R/W)
- I2C_SCL_MAIN_ST_TO_INT_ENA** I2C_SCL_MAIN_ST_TO_INT 的使能位。(R/W)
- I2C_DET_START_INT_ENA** I2C_DET_START_INT 的使能位。(R/W)
- I2C_SLAVE_STRETCH_INT_ENA** I2C_SLAVE_STRETCH_INT 的使能位。(R/W)
- I2C_GENERAL_CALL_INT_ENA** I2C_GENARAL_CALL_INT 的使能位。(R/W)

Register 23.25. I2C_INT_STATUS_REG (0x002C)

(reserved)																		I2C_GENERAL_CALL_INT_ST																		I2C_SLAVE_STRETCH_INT_ST																		I2C_DET_START_INT_ST																		I2C_SCL_MAIN_ST_TO_INT_ST																		I2C_SCL_ST_TO_INT_ST																		I2C_RXFIFO_UDF_INT_ST																		I2C_TXFIFO_OVF_INT_ST																		I2C_NACK_INT_ST																		I2C_TRANS_START_INT_ST																		I2C_TIME_OUT_INT_ST																		I2C_TRANS_COMPLETE_INT_ST																		I2C_MST_TXFIFO_UDF_INT_ST																		I2C_ARBTRATION_LOST_INT_ST																		I2C_BYTE_TRANS_DONE_INT_ST																		I2C_END_DETECT_INT_ST																		I2C_RXFIFO_OVF_INT_ST																		I2C_TXFIFO_WM_INT_ST																		I2C_RXFIFO_WM_INT_ST																	
31																		18																		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset																																																																																																																																																																																																																																																																																															
0																		0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																																																																										

I2C_RXFIFO_WM_INT_ST I2C_RXFIFO_WM_INT 的屏蔽状态位。(RO)

I2C_TXFIFO_WM_INT_ST I2C_TXFIFO_WM_INT 的屏蔽状态位。(RO)

I2C_RXFIFO_OVF_INT_ST I2C_RXFIFO_OVF_INT 的屏蔽状态位。(RO)

I2C_END_DETECT_INT_ST I2C_END_DETECT_INT 的屏蔽状态位。(RO)

I2C_BYTE_TRANS_DONE_INT_ST I2C_END_DETECT_INT 的屏蔽状态位。(RO)

I2C_ARBTRATION_LOST_INT_ST I2C_ARBTRATION_LOST_INT 的屏蔽状态位。(RO)

I2C_MST_TXFIFO_UDF_INT_ST I2C_TRANS_COMPLETE_INT 的屏蔽状态位。(RO)

I2C_TRANS_COMPLETE_INT_ST I2C_TRANS_COMPLETE_INT 的屏蔽状态位。(RO)

I2C_TIME_OUT_INT_ST I2C_TIME_OUT_INT 的屏蔽状态位。(RO)

I2C_TRANS_START_INT_ST I2C_TRANS_START_INT 的屏蔽状态位。(RO)

I2C_NACK_INT_ST I2C_SLAVE_STRETCH_INT 的屏蔽状态位。(RO)

I2C_TXFIFO_OVF_INT_ST I2C_TXFIFO_OVF_INT 的屏蔽状态位。(RO)

I2C_RXFIFO_UDF_INT_ST I2C_RXFIFO_UDF_INT 的屏蔽状态位。(RO)

I2C_SCL_ST_TO_INT_ST I2C_SCL_ST_TO_INT 的屏蔽状态位。(RO)

I2C_SCL_MAIN_ST_TO_INT_ST I2C_SCL_MAIN_ST_TO_INT 的屏蔽状态位。(RO)

I2C_DET_START_INT_ST I2C_DET_START_INT 的屏蔽状态位。(RO)

I2C_SLAVE_STRETCH_INT_ST I2C_SLAVE_STRETCH_INT 的屏蔽状态位。(RO)

I2C_GENERAL_CALL_INT_ST I2C_GENARAL_CALL_INT 的屏蔽状态位。(RO)

423

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

- ## 反馈文档意见

反馈文档意见

反馈文档意见

423

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

反馈文档意见

423

ESP32-C3 TRM (预发布 v0.4)

反馈文档意见

反馈文档意见

Register 23.29. I2C_COMD3_REG (0x0064)

I2C_COMMAND3_DONE										(reserved)					I2C_COMMAND3															
31	30													14	13	0														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
																													Reset	

I2C_COMMAND3 命令寄存器 3 的内容，同 [I2C_COMMAND0](#)。(R/W)

I2C_COMMAND3_DONE 在 I2C 主机模式下完成命令 3 时，该位翻转为高电平。(R/W/SS)

Register 23.30. I2C_COMD4_REG (0x0068)

I2C_COMMAND4_DONE										(reserved)					I2C_COMMAND4															
31	30													14	13	0														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
																													Reset	

I2C_COMMAND4 命令寄存器 4 的内容，同 [I2C_COMMAND0](#)。(R/W)

I2C_COMMAND4_DONE 在 I2C 主机模式下完成命令 4 时，该位翻转为高电平。(R/W/SS)

Register 23.31. I2C_COMD5_REG (0x006C)

I2C_COMMAND5_DONE										(reserved)					I2C_COMMAND5															
31	30													14	13	0														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
																													Reset	

I2C_COMMAND5 命令寄存器 5 的内容，同 [I2C_COMMAND0](#)。(R/W)

I2C_COMMAND5_DONE 在 I2C 主机模式下完成命令 5 时，该位翻转为高电平。(R/W/SS)

Register 23.32. I2C_COMD6_REG (0x0070)

I2C_COMMAND6_DONE																(reserved)																I2C_COMMAND6																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
31	30													14	13	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

I2C_COMMAND6 命令寄存器 6 的内容, 同 [I2C_COMMAND0](#)。(R/W)

I2C_COMMAND6_DONE 在 I2C 主机模式下完成命令 6 时, 该位翻转为高电平。(R/W/SS)

Register 23.33. I2C_COMD7_REG (0x0074)

I2C_COMMAND7_DONE																(reserved)																I2C_COMMAND7															
31	30														14	13	0																														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											Reset																	

I2C_COMMAND7 命令寄存器 7 的内容, 同 [I2C_COMMAND0](#)。(R/W)

I2C_COMMAND7_DONE 在 I2C 主机模式下完成命令 7 时, 该位翻转为高电平。(R/W/SS)

Register 23.34. I2C_DATE_REG (0x00F8)

I2C_DATE															
310															
0x20070201Reset															

I2C_DATE 版本控制寄存器。(R/W)

24 USB 串口/JTAG 控制器 (USB_SERIAL_JTAG)

ESP32-C3 中包含一个 USB 串口/JTAG 控制器，可用于烧录芯片的外部 flash、读取程序输出的数据以及将调试器连接到正在运行的程序中。任何带有 USB 主机（下文将简称为“主机”）的计算机都可以实现上述功能，无需其他外部组件辅助。

24.1 概述

开发 ESP32 芯片通常使用两种方式和芯片通信：串口通信和 JTAG 调试端口通信。串口是一个双线接口，传统上用于将开发中的新固件烧录到 ESP32 上。由于大多数现代计算机上已没有兼容的串口，因此需要一个 USB 转串口集成电路或开发板来解决这一问题。固件烧录完成后，该端口即被用于监视程序中的调试输出数据，从而关注程序运行的总体状态。当程序运行中出现异常情况（程序崩溃）时，需使用 JTAG 调试端口检查程序及其变量的状态，并设置断点和观察点。此时便需要利用一个外部 JTAG 适配器使 SoC 与 JTAG 调试端口建立连接。

上述外部接口共需占用 6 个管脚，且在调试过程中，这些管脚便不能用于其他功能。然而，对于 ESP32-C3 这种小封装的设备，不能使用上述管脚会限制其设计。

为解决这一问题，同时尽可能减少对外部设备的需求，ESP32-C3 中包含了一个 USB 串口/JTAG 控制器，同时集成 USB-串口转换器和 USB-JTAG 适配器功能。由于该模块仅使用 USB1.1 所需的两条数据线直接连接外部 USB 主机，因此 ESP32-C3 仅需占用 2 个管脚用于调试。

24.2 特性

- USB 全速标准
- 固定功能。包含连接的 CDC-ACM（通信设备类抽象控制模型）和 JTAG 适配器功能
- 共 2 个 OUT 端点、3 个 IN 端点和 1 个控制端点 EP_0，可实现最大 64 字节的数据载荷
- 有内部 PHY，基本无需其他外部组件连接主机计算机
- CDC-ACM 的虚拟串行功能在大多数现代操作系统上可实现即插即用
- JTAG 接口可使用紧凑的 JTAG 指令实现与 CPU 调试内核的快速通信
- CDC-ACM 支持主机控制芯片复位和进入下载模式

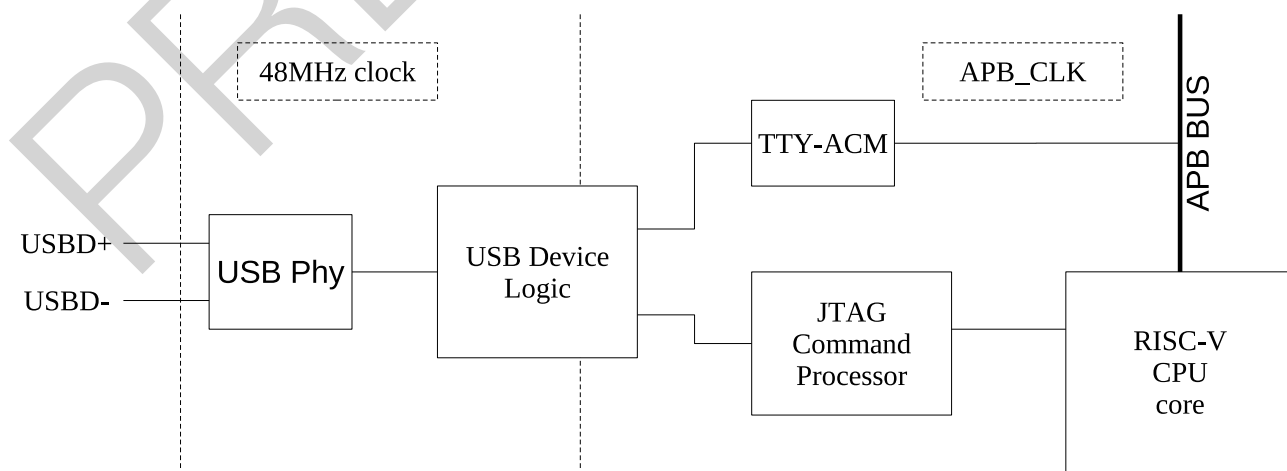


图 24-1. USB Serial/JTAG 高层框图

如图 24-1 所示，USB 串口/JTAG 控制器包含一个 USB PHY、USB 设备接口、JTAG 命令处理器、响应捕捉单元以及若干个 CDC_ACM 寄存器。PHY 和部分 USB 设备接口使用主 PLL 时钟产生的 48 MHz 时钟作为时钟源，除此以外的其他部分则使用 APB_CLK 作为时钟源。JTAG 命令处理器与 ESP32-C3 主处理器中的 JTAG 调试单元相连；CDC-ACM 寄存器则连接至 APB 总线，主 CPU 上运行的软件可对其进行读写访问。

请注意，USB 串口/JTAG 控制器为 USB 2.0 全速标准 (12 Mbps)，不支持 USB 2.0 标准的其他模式（如，480 Mbps 的高速模式）。

图 24-2 显示了 USB 串口/JTAG 控制器中 USB 部分的内部详细信息。USB 串口/JTAG 控制器由一个 USB 2.0 全速设备组成，包含 1 个控制端点、1 个虚拟中断端点、2 个批量输入端点和 2 个批量输出端点。这些端点共同组成了该复合型 USB 设备，具体可分为 CDC-ACM USB 设备和实现 JTAG 接口功能的供应商特定设备。JTAG 接口直接与芯片的 RISC-V CPU 的调试接口相连，可对运行在该 CPU 上的程序进行调试。同时，CDC-ACM 中包含一组寄存器，CPU 上运行的程序可对其进行读写操作。此外，芯片上的 ROM 启动代码可允许用户通过使用该接口重新烧录 flash。

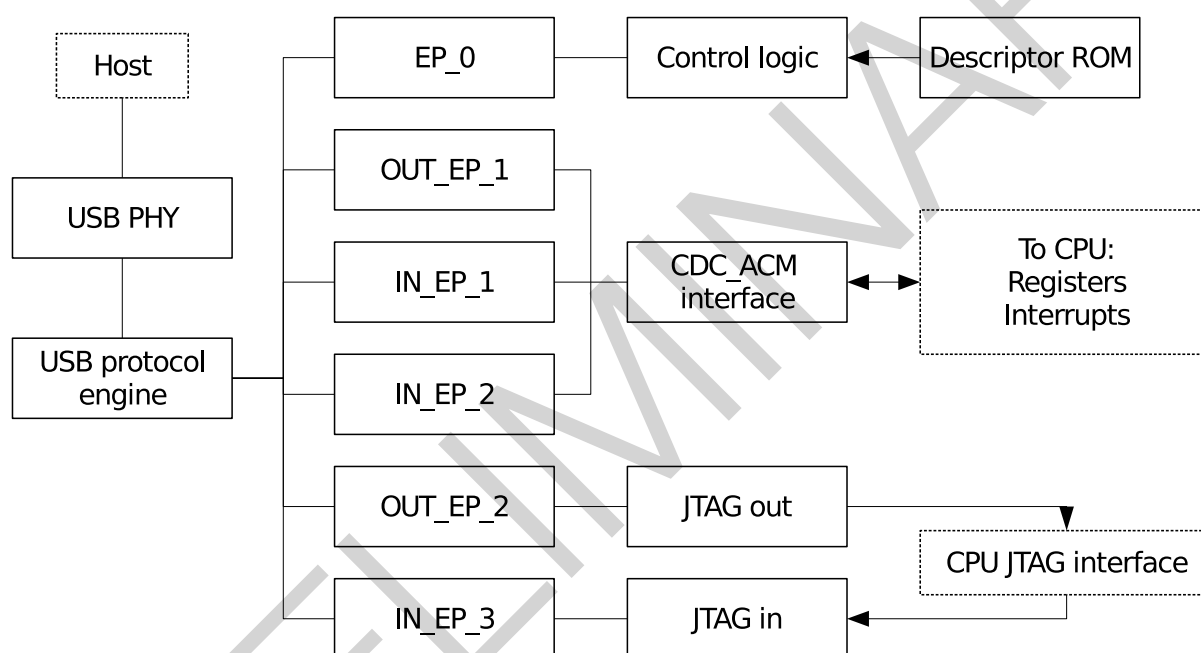


图 24-2. USB Serial/JTAG 框图

24.3 功能描述

USB 串口/JTAG 控制器一边与 USB 主机处理器连接，另一边与 CPU 调试硬件以及在 USB 端口上运行的软件连接。

24.3.1 CDC-ACM USB 接口描述

CDC-ACM 接口遵循标准 USB CDC-ACM 类别进行虚拟串口通信，包含一个虚中断端点（不会发送任何事件，无使用需求）以及一个批量输入端点 (Bulk IN) 和批量输出端点 (Bulk OUT) 进行数据接收和发送。这些端点一次可以处理最高 64 字节的数据包，实现高吞吐量。CDC-ACM 为标准的 USB 设备类型，主机一般无需任何特殊安装程序就能正常工作，也就是说，当一个 USB 调试设备正确连接至主机时，操作系统应能在片刻后显示新的串口信息。

CDC-ACM 接口可以接收以下标准 CDC-ACM 控制请求：

表 24-1. 标准 CDC-ACM 控制请求

命令	操作
SEND_BREAK	接收但忽略（虚拟命令）
SET_LINE_CODING	接收但忽略（虚拟命令）
GET_LINE_CODING	总是返回 9600 baud，无奇偶校验，8 个数据位，1 个停止位
SET_CONTROL_LINE_STATE	设置 RTC/DTR 线的状态，如表 24-2 所示

除了通用的通信之外，CDC-ACM 接口还可以复位 ESP32-C3 并选择使其进入下载模式，从而烧录新的固件。这一功能可通过设置虚拟串口的 RTS 和 DTR 线来实现。

表 24-2. CDC-ACM 中 RTS 和 DTR 的设置

RTS	DTR	操作
0	0	清除下载模式标志
0	1	置位下载模式标志
1	0	复位 ESP32-C3
1	1	无操作

请注意，当 ESP32-C3 复位时，如果下载模式标志已置位，则 ESP32-C3 重启时将直接进入下载模式；如果下载模式标志已清除，则 ESP32-C3 将从 flash 启动。具体操作流程，请参见章节 24.4。除此之外，也可以通过烧写相应 eFuse 来禁用上述功能，详细信息请参见章节 4 eFuse 控制器 (EFUSE)。

24.3.2 CDC-ACM 固件接口描述

由于 USB 串口/JTAG 控制器与 ESP32-C3 的内部 APB 总线相连，因此 CPU 可直接与该模块交互，主要对连接的主机上的虚拟串口进行读写操作。

CPU 向主机发送并从主机接收 0 ~ 64 字节大小的 USB CDC-ACM 串口数据包。主机已接收到足够多的 CDC-ACM 数据时，将向 CDC-ACM 的接收端点发送一个数据包，如果 USB 串行/JTAG 控制器中有空闲缓冲区，该缓冲区将接收这一数据包。反之，主机会定期检查 USB 串口/JTAG 控制器内是否有待向主机发送的数据包，如果有，主机将接收这个数据包。

固件可通过以下两种方式之一获知是否有来自主机的新数据：第一，只要缓冲区中还有来自主机的未读数据，[USB_SERIAL_JTAG_SERIAL_OUT_EP_DATA_AVAIL](#) 位将保持为 1；第二，如果有新的未读数据，[USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT](#) 中断也将被触发。

当新数据可用时，固件可通过重复从 [USB_SERIAL_JTAG_EP1_REG](#) 读取字节来获取该数据。读取每个字节后，可通过检查 [USB_REG_SERIAL_OUT_EP_DATA_AVAIL](#) 位来看是否还有其他可读取的数据，从而确定需要读取的总字节数。读取完所有数据后，USB 调试设备会自动做好准备，以接收来自主机的新数据包。

当固件需要发送数据时，可将待发送数据置于发送缓冲区并触发刷写，从而使主机以 USB 数据包的形式接收该数据。在此之前，需确保发送缓冲区有可用空间存储待发送数据。固件可通过读取 [USB_REG_SERIAL_IN_EP_DATA_FREE](#) 位检查发送缓冲区是否有可用空间：当该值为 1 时，发送缓冲区有可用空间。此时，固件可通过向 [USB_SERIAL_JTAG_EP1_REG](#) 寄存器写入字节从而向缓冲区中写入数据。

但是，数据写入后并不会立即触发向主机发送数据，还需对缓冲区执行刷写操作。刷写操作后，整个缓冲区可准备好被 USB 主机立即接收。可通过两种方式触发刷写：将第 64 个字节写入缓冲区后，USB 硬件会自动将缓

缓冲区刷写到主机；固件可通过向 `USB_REG_SERIAL_WR_DONE` 写入 1 来触发刷写。

不论以何种方式触发刷写操作，在此期间固件都无法向缓冲区写入数据，直到缓冲区中的所有数据都被主机读取完成。主机读取完成后，将触发 `USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT` 中断，此时可向缓冲区内写入新的 64 字节数据。

24.3.3 USB-JTAG 接口：JTAG 命令处理器

USB-JTAG 接口使用一种供应商特定接口类型实现命令解析的功能。它由两个端点组成：一个用于接收命令，一个用于发送响应。此外，一些对时效性要求不高的命令也可以作为控制请求发出。

JTAG 命令处理器内部包含一个全四线 JTAG 总线，包括发送信号到 RISC-V CPU 的 TCK、TMS 和 TDI 输出线，以及从 CPU 返回信号至 JTAG 响应捕捉单元的 TDO 线。这些信号都符合 IEEE 1149.1 JTAG 标准。此外，还有一条 SRST 线用于复位 ESP32-C3。

JTAG 命令处理器会将每个接收到的半字节 (4-bit) 解析为一条命令。由于 USB 以 8-bit 为一个字节接收数据，这就意味着每个字节中都包含两条命令。USB 命令处理器将先解析高 4-bit 字节，然后再解析低 4-bit 字节。这些命令用于控制内部 JTAG 总线的 TCK、TMS、TDI、SRST 线，以及向 JTAG 响应捕捉单元发出信号，说明需要捕捉 TDO 线（由 CPU 调试逻辑驱动）状态。

JTAG 总线中，TCK、TMS、TDI 和 TDO 线直接与 RISC-V CPU 的 JTAG 调试逻辑相连。上文提到的 SRST 线则与 ESP32-C3 数字电路中的复位逻辑相连，该线电平拉高，芯片将进行芯片复位。请注意，SRST 线并不会对 USB 串口/JTAG 控制器模块产生影响。

1 个半字节中可包含以下命令：

表 24-3. 半字节中的命令

位	3	2	1	0
CMD_CLK	0	cap	tms	tdi
CMD_RST	1	0	0	srst
CMD_FLUSH	1	0	1	0
CMD_RSV	1	0	1	1
CMD_REP	1	1	R1	R0

- **CMD_CLK**：将 TDI 和 TMS 设置为指示值，并在 TCK 上发出一个时钟脉冲。如果 CAP 位为 1，它将指示 JTAG 响应捕捉单元捕捉 TDO 线的状态。该指令构成了 JTAG 通信的基础。
- **CMD_RST**：将 SRST 线的状态设置为指示值。该命令可用于复位 ESP32-C3。
- **CMD_FLUSH**：指示 JTAG 响应捕捉单元对接收到的所有位的缓冲区进行刷写操作，以便主机可以读取这些位。请注意在某些情况下，一次 JTAG 通信会结束于第奇数个命令，即结束于第奇数个半字节。此时，可重复执行该命令直到获得偶数个半字节，使其组成整数个字节。
- **CMD_RSV**：该版本中保留。ESP32-C3 在接收到该命令时会自动忽略。
- **CMD_REP**：重复上一条指令（非 CMD_REP）一定的次数。该命令的目的是压缩多次重复 CMD_CLK 的命令流。因此，CMD_CLK 命令后可能跟随着多个 CMD_REP。一次 CMD_REP 命令产生的重复次数可表示为 $no_repetitions = (R1 \times 2 + R0) \times (4^{cmd_rep_count})$ ，其中 `cmd_rep_count` 表示该命令之前的 CMD_REP 数量。请注意，CMD_REP 仅用于重复 CMD_CLK 命令。也就是说，如果在 CMD_FLUSH 后使用该命令，USB 设备将无法响应，需进行 USB 复位后才可恢复正常。

24.3.4 USB-JTAG 接口：CMD_REP 使用示例

下列命令用于演示如何使用 CMD_REP 命令。请注意，该示例中每个命令为半字节，命令流的每个字节为 0x0D 0x5E 0xCF。

1. 0x0 (CMD_CLK: cap=0, tdi=0, tms=0)
2. 0xD (CMD_REP: R1=0, R0=1)
3. 0x5 (CMD_CLK: cap=1, tdi=0, tms=1)
4. 0xE (CMD_REP: R1=1, R0=0)
5. 0xC (CMD_REP: R1=0, R0=0)
6. 0xF (CMD_REP: R1=1, R0=1)

每一步骤的具体操作为：

1. TCK 上发出一个时钟脉冲，TDI 和 TMS 设置为 0。未捕捉到任何数据。
2. TCK 上再次发出 $(0 \times 2 + 1) \times (4^2) = 1$ 个时钟脉冲，其他设置与步骤 1 相同。
3. TCK 上发出一个时钟脉冲，TDI 和 TMS 设置为 0。捕捉到 TDO 线上的数据。
4. TCK 上再次发出 $(1 \times 2 + 0) \times (4^0) = 2$ 个时钟脉冲，其他设置与步骤 3 相同。
5. 未发生任何动作： $(0 \times 2 + 0) \times (4^1) = 0$ 。请注意，该步骤操作将增加下一步骤中的 cmd_rep_count 数值。
6. TCK 上再次发出 $(1 \times 2 + 1) \times (4^2) = 48$ 个时钟脉冲，其他设置与步骤 3 相同。

换言之，该命令流示例的操作结果等同于执行 2 次命令 1，然后执行 51 次命令 3。

24.3.5 USB-JTAG 接口：响应捕捉单元

响应捕捉单元首先读取内部 JTAG 总线的 TDO 线，并在命令处理器执行 CMD_CLK (cap=1) 命令时捕捉 TDO 线的值。然后它将这个值放入内部移位寄存器中，且在接收到 8-bit 时向 USB 缓冲区写入 1 个字节。这 8-bit 中的最低有效位即为最先从 TDO 线读取的值。

一旦接收到 64 字节 (512-bit) 数据或执行 CMD_FLUSH 命令后，响应捕捉单元将使缓冲区可被主机接收。请注意，USB 逻辑的接口为双缓冲。这样，只要 USB 的吞吐量充足，响应捕捉单元就可以随时接收更多数据，即当一个缓冲区等待发送给主机时，另一个缓冲区可以继续接收数据。当主机从缓冲区成功接收数据且响应捕捉单元对缓冲区执行刷写操作后，这两个缓冲区便可以交换位置。

同时，这也意味着一个命令流可导致最多 128 字节（若该命令流中有刷写命令，则该数字会减小）的捕捉数据生成，而不需要主机主动接收这些数据。如果还是生成了超过该阈值数量的捕捉数据，则命令流将被暂停，且在这些数据被读取之前设备不会接收其他命令。

另需注意，一般情况下，响应捕捉单元的逻辑会尽量不发送 0 字节响应。例如，当发送一系列 CMD_FLUSH 命令后并不会产生一系列 0 字节 USB 响应。但是，当前版本中一些特殊情况下也可能产生 0 字节响应，建议用户可直接忽略这些响应信息。

24.3.6 USB-JTAG 接口：控制传输请求

除命令处理器和响应捕捉单元之外，USB-JTAG 接口也可接收一些控制请求，具体如下表所示：

表 24-4. USB-JTAG 控制请求

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000b	0 (VEND_JTAG_SETDIV)	[divider]	接口	0	None
01000000b	1 (VEND_JTAG_SETIO)	[jobits]	接口	0	None
11000000b	2 (VEND_JTAG_GETTDO)	0	接口	1	[iostate]
10000000b	6 (GET_DESCRIPTOR)	0x2000	0	256	[jtag cap desc]

- VEND_JTAG_SETDIV: 设置使用的分频器。该命令将直接影响 TCK 时钟脉冲的持续时间。TCK 时钟脉冲来自于由内部分频器向下分频得到的内部 APB 时钟。该控制请求允许主机来设置这个内部分频器。请注意，该分频器在启动时的初始值为 2，即 TCK 时钟速率一般为 40 MHz。
- VEND_JTAG_SETIO: 跳过 JTAG 命令处理器直接将内部 TDI、TDO、TMS 和 SRST 线设置为指定值。这些指定值在 wValue 字段中以 11'b0, srst, trst, tck, tms, tdi 的格式编码。
- VEND_JTAG_GETTDO: 跳过 JTAG 响应捕捉单元直接读取内部 TDO 信号。该请求将返回 1 个字节，其中的最低有效位代表 TDO 线的状态。
- GET_DESCRIPTOR: 为标准 USB 请求，该请求也可使用供应商专用的 0x2000 wValue 获取 JTAG 功能描述符。该请求将返回一定字节，具体字节数所代表的 USB-JTAG 适配器功能如表 24-5 所示。这一固定结构允许主机软件自动支持未来新的硬件版本，无需再次更新。

ESP32-C3 包含的 JTAG 功能描述符如下表所示。请注意，所有 16-bit 值都为小端序存储。

表 24-5. JTAG 功能描述符

字节	数值	描述
0	1	JTAG 协议功能结构的版本
1	10	JTAG 协议功能长度
2	1	结构类型：1 代表高速功能结构类型
3	8	该高速功能结构长度
4 ~ 5	8000	以 10 KHz 为增量的 APB 速度，其基础速度为该值的一半
6 ~ 7	1	最小分频系数
8 ~ 9	255	最大分频系数

24.4 操作建议

使用 USB 串口/JTAG 控制器之前，几乎不需要多余的配置。除了主机操作系统已经完成的标准 USB 初始化之外，USB-JTAG 硬件本身不需要进行任何配置。而 CDC-ACM 虚拟串口在主机端也是即插即用的。

固件方面也几乎不需要初始化：USB 硬件是自初始化的，在其启动后，如果固件连接了一台主机并在 CDC-ACM 接口上监听，除非固件选择设置了中断处理程序，其他情况下无需任何特定设置就可以实现前文所述的数据交换。

需要注意的是，可能会出现主机未连接或 CDC-ACM 虚拟串口未打开的情况。在这种情况下，发送至主机的数据包永远无法被接收，发送缓冲区也永远不会为空。因而，对此进行检测以及执行超时操作便十分重要，这样才能清楚地检测到主机侧的端口是否关闭。

其次，需知 USB 设备依赖于产生 48 MHz USB PHY 时钟的 PLL 时钟和 APB 时钟。具体来说，正确的 USB 顺序操作要求 APB 时钟至少为 40 MHz，但在与低至 10 MHz APB 时钟的主机相连时 USB 也能工作。此时 USB 是否会出现其他问题则取决于主机的硬件和驱动条件，可能会有设备出现无法响应或在首次访问时消失的情况。

换言之，在 Light-sleep 模式下对 USB 串口/JTAG 控制器进行时钟门控操作时，APB 时钟将受到影响。除此之外，在 Deep-sleep 模式下，USB 串口/JTAG 控制器（及其连接的 RISC-V CPU）将完全断电。如果有设备需要在这两种模式下进行调试，最好使用一个外部 JTAG 调试器和串行接口。

CDC-ACM 接口还可用于复位芯片，使其进入或退出下载模式。产生正确的握手信号序列则有些复杂，因为大多数操作系统仅支持分别设置或重置 DTR 和 RTS，无法同时进行。此外，一些驱动程序（如，Windows 系统上的标准 CDC-ACM 驱动程序）须先设置 RTS 后才可设置 DTR，导致用户必须明确设置 RTS 才能传播 DTR 的值。推荐遵循以下程序进行设置：

复位芯片使其进入下载模式：

表 24-6. 复位芯片进入下载模式

操作	内部状态	备注
清除 DTR	RTS=?, DTR=0	初始化以获取数值
清除 RTS	RTS=0, DTR=0	-
设置 DTR	RTS=0, DTR=1	设置下载模式标志
清除 RTS	RTS=0, DTR=1	传播 DTR
设置 RTS	RTS=1, DTR=1	-
清除 DTR	RTS=1, DTR=0	复位芯片
设置 RTS	RTS=1, DTR=0	传播 DTR
清除 RTS	RTS=0, DTR=0	清除下载标志

复位 SoC 使其从 flash 启动：

表 24-7. 复位 SoC 进行启动

操作	内部状态	备注
清除 DTR	RTS=?, DTR=0	-
清除 RTS	RTS=0, DTR=0	清除下载标志
设置 RTS	RTS=1, DTR=0	复位 SoC
清除 RTS	RTS=0, DTR=0	退出复位

24.5 寄存器列表

本小节的所有地址均为相对于 USB Serial/JTAG 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
USB_SERIAL_JTAG_EP1_REG	CDC-ACM 输入/输出端点 FIFO 访问配置寄存器	0x0000	R/W
USB_SERIAL_JTAG_CONF0_REG	PHY 硬件配置寄存器	0x0018	R/W
USB_SERIAL_JTAG_TEST_REG	PHY 调试寄存器	0x001C	R/W
USB_SERIAL_JTAG_MISC_CONF_REG	时钟使能控制寄存器	0x0044	R/W
USB_SERIAL_JTAG_MEM_CONF_REG	存储器配置寄存器	0x0048	R/W
状态寄存器			
USB_SERIAL_JTAG_EP1_CONF_REG	CDC-ACM FIFO 配置与控制寄存器	0x0004	varies
USB_SERIAL_JTAG_JFIFO_ST_REG	JTAG FIFO 状态与控制寄存器	0x0020	varies
USB_SERIAL_JTAG_FRAM_NUM_REG	接收 SOF 帧索引寄存器	0x0024	RO
USB_SERIAL_JTAG_IN_EP0_ST_REG	输入端点状态信息寄存器	0x0028	RO
USB_SERIAL_JTAG_IN_EP1_ST_REG	CDC-ACM 输入端点状态信息寄存器	0x002C	RO
USB_SERIAL_JTAG_IN_EP2_ST_REG	CDC-ACM 中断输入端点状态信息寄存器	0x0030	RO
USB_SERIAL_JTAG_IN_EP3_ST_REG	JTAG 输入端点状态信息寄存器	0x0034	RO
USB_SERIAL_JTAG_OUT_EP0_ST_REG	输出端点状态信息寄存器	0x0038	RO
USB_SERIAL_JTAG_OUT_EP1_ST_REG	CDC-ACM 输出端点状态信息寄存器	0x003C	RO
USB_SERIAL_JTAG_OUT_EP2_ST_REG	JTAG 输出端点状态信息寄存器	0x0040	RO
中断寄存器			
USB_SERIAL_JTAG_INT_RAW_REG	中断原始状态寄存器	0x0008	R/ WTC/ SS
USB_SERIAL_JTAG_INT_ST_REG	中断状态寄存器	0x000C	RO
USB_SERIAL_JTAG_INT_ENA_REG	中断使能状态寄存器	0x0010	R/W
USB_SERIAL_JTAG_INT_CLR_REG	中断清除状态寄存器	0x0014	WT
版本寄存器			
USB_SERIAL_JTAG_DATE_REG	版本寄存器	0x0080	R/W

24.6 寄存器

本小节的所有地址均为相对于 USB Serial/JTAG 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 24.1. USB_SERIAL_JTAG_EP1_REG (0x0000)

(reserved)																								USB_SERIAL_JTAG_PDOWN_BYTE									
31																								8	7	0							
0 0																								0x0								Reset	

USB_SERIAL_JTAG_RDWR_BYTE 通过该字段向 UART Tx FIFO 中写入数据，或者从 UART Rx FIFO 中读取数据。USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT 置位时，用户可向 UART Tx FIFO 中写入数据（最大 64 字节）。当 USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 置位时，用户可通过查看 USB_SERIAL_JTAG_OUT_EP1_WR_ADDR 和 USB_SERIAL_JTAG_OUT_EP0_RD_ADDR 的值获知接收到的数据量，然后从 UART Rx FIFO 中读取这些数据。(R/W)

Register 24.2. USB_SERIAL_JTAG_CONF0_REG (0x0018)

(reserved)																USB_SERIAL_JTAG_USB_PAD_ENABLE USB_SERIAL_JTAG_PULLUP_VALUE USB_SERIAL_JTAG_DM_PULLDOWN USB_SERIAL_JTAG_DP_PULLUP USB_SERIAL_JTAG_DP_PULLDOWN USB_SERIAL_JTAG_PAD_PULLUP USB_SERIAL_JTAG_VREF_OVERRIDE USB_SERIAL_JTAG_VREFH USB_SERIAL_JTAG_VREFL USB_SERIAL_JTAG_EXCHG_PINS USB_SERIAL_JTAG_PHY_SEL															
31																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

USB_SERIAL_JTAG_PHY_SEL 选择使用内部 PHY 或外部 PHY。1' b0: 内部 PHY; 1' b1: 外部 PHY。(R/W)

USB_SERIAL_JTAG_EXCHG_PINS_OVERRIDE 使能软件控制 USB D+ 和 D- 管脚交换。(R/W)

USB_SERIAL_JTAG_EXCHG_PINS 交换 USB D+ 和 D- 管脚。(R/W)

USB_SERIAL_JTAG_VREFL 控制单端输入高阈值, 1.76 V ~ 2 V, 步长 80 mV。(R/W)

USB_SERIAL_JTAG_VREFH 控制单端输入低阈值, 0.8 V ~ 1.04 V, 步长 80 mV。(R/W)

USB_SERIAL_JTAG_VREF_OVERRIDE 使能软件控制输入阈值。(R/W)

USB_SERIAL_JTAG_PAD_PULL_OVERRIDE USB D+ 和 D- 管脚的上下拉电阻。(R/W)

USB_SERIAL_JTAG_DP_PULLUP USB D+ 管脚的上拉电阻。(R/W)

USB_SERIAL_JTAG_DP_PULLDOWN USB D+ 管脚的下拉电阻。(R/W)

USB_SERIAL_JTAG_DM_PULLUP USB D- 管脚的上拉电阻。(R/W)

USB_SERIAL_JTAG_DM_PULLDOWN USB D- 管脚的下拉电阻。(R/W)

USB_SERIAL_JTAG_PULLUP_VALUE 控制上拉数值。(R/W)

USB_SERIAL_JTAG_USB_PAD_ENABLE 使能 USB 填充功能。(R/W)

Register 24.3. USB_SERIAL_JTAG_TEST_REG (0x001C)

(reserved)																												USB_SERIAL_JTAG_TEST_TX_DM																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
																												USB_SERIAL_JTAG_TEST_TX_DP																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
																												USB_SERIAL_JTAG_TEST_USB_OE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
																												USB_SERIAL_JTAG_TEST_ENABLE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
31																											4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

USB_SERIAL_JTAG_TEST_ENABLE 使能测试 USB 填充功能。(R/W)

USB_SERIAL_JTAG_TEST_USB_OE 测试 USB oe 填充。(R/W)

USB_SERIAL_JTAG_TEST_TX_DP 测试 USB D+ 管脚的发送值。(R/W)

USB_SERIAL_JTAG_TEST_TX_DM 测试 USB D- 管脚的发送值。(R/W)

Register 24.4. USB_SERIAL_JTAG_MISC_CONF_REG (0x0044)

(reserved)																															USB_SERIAL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
31																														1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

USB_SERIAL_JTAG_CLK_EN 1'h1: 强制打开寄存器的时钟; 1'h0: 支持仅当应用程序向寄存器写入数据时打开时钟。(R/W)

Register 24.5. USB_SERIAL_JTAG_MEM_CONF_REG (0x0048)

(reserved)																												USB_SERIAL_JTAG			USB_SERIAL_JTAG		
31																											2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Reset			

USB_SERIAL_JTAG_USB_MEM_PD 置位关闭 USB 存储器。(R/W)

USB_SERIAL_JTAG_USB_MEM_CLK_EN 置位强制对 USB 存储器进行时钟分频。(R/W)

Register 24.6. USB_SERIAL_JTAG_EP1_CONF_REG (0x0004)

(reserved)																															USB_SERIAL_JTAG_SERIAL_OUT_EP_DATA_AVAIL				USB_SERIAL_JTAG_SERIAL_IN_EP_DATA_FREE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
31																												3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_SERIAL_JTAG_WR_DONE 置位表示已完成向 UART Tx FIFO 写入字节。然后，该位将保持为 0，直到 USB 主机读取 UART Tx FIFO 中的数据。(WT)

USB_SERIAL_JTAG_SERIAL_IN_EP_DATA_FREE 1'b1: UART Tx FIFO 不为空且可以写入数据。写入 USB_SERIAL_JTAG_WR_DONE 后，该位将保持为 1' b0，直到其中的数据已发送至 USB 主机。(RO)

USB_SERIAL_JTAG_SERIAL_OUT_EP_DATA_AVAIL 1'b1: UART Rx FIFO 中有数据。(RO)

Register 24.7. USB_SERIAL_JTAG_JFIFO_ST_REG (0x0020)

(reserved)																						USB_SERIAL_JTAG_OUT_FIFO_RESET USB_SERIAL_JTAG_IN_FIFO_RESET USB_SERIAL_JTAG_OUT_FIFO_FULL USB_SERIAL_JTAG_OUT_FIFO_EMPTY USB_SERIAL_JTAG_IN_FIFO_FULL USB_SERIAL_JTAG_IN_FIFO_EMPTY USB_SERIAL_JTAG_IN_FIFO_CNT																			
31																						10		9	8	7	6	5	4	3	2	1	0								
0 0																						0 0		0	1	0		0	1	0	Reset										

USB_SERIAL_JTAG_IN_FIFO_CNT FIFO 计数器中的 JTAG。(RO)

USB_SERIAL_JTAG_IN_FIFO_EMPTY 置位表示 FIFO 中的 JTAG 为空。(RO)

USB_SERIAL_JTAG_IN_FIFO_FULL 置位表示 FIFO 中的 JTAG 为满。(RO)

USB_SERIAL_JTAG_OUT_FIFO_CNT JTAG 输出 FIFO 计数器。(RO)

USB_SERIAL_JTAG_OUT_FIFO_EMPTY 置位表示 JTAG 输出 FIFO 为空。(RO)

USB_SERIAL_JTAG_OUT_FIFO_FULL 置位表示 JTAG 输出 FIFO 为满。(RO)

USB_SERIAL_JTAG_IN_FIFO_RESET 置位复位 FIFO 中的 JTAG。(R/W)

USB_SERIAL_JTAG_OUT_FIFO_RESET 置位复位 JTAG 输出 FIFO。(R/W)

Register 24.8. USB_SERIAL_JTAG_FRAM_NUM_REG (0x0024)

(reserved)																USB_SERIAL_JTAG_SOF_FRAME_INDEX																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																11																10																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

USB_SERIAL_JTAG_SOF_FRAME_INDEX 接收 SOF 帧的帧索引。(RO)

Register 24.9. USB_SERIAL_JTAG_IN_EP0_ST_REG (0x0028)

(reserved)																USB_SERIAL_JTAG_IN_EP0_RD_ADDR								USB_SERIAL_JTAG_IN_EP0_WR_ADDR								USB_SERIAL_JTAG_IN_EP0_STATE							
31																16	15								9	8							2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0								1		0			
																																		Reset					

- USB_SERIAL_JTAG_IN_EP0_STATE 输入端点 0 的状态。(RO)
- USB_SERIAL_JTAG_IN_EP0_WR_ADDR 输入端点 0 的写入数据地址。(RO)
- USB_SERIAL_JTAG_IN_EP0_RD_ADDR 输入端点 0 的读取数据地址。(RO)

Register 24.10. USB_SERIAL_JTAG_IN_EP1_ST_REG (0x002C)

(reserved)																USB_SERIAL_JTAG_IN_EP1_RD_ADDR								USB_SERIAL_JTAG_IN_EP1_WR_ADDR								USB_SERIAL_JTAG_IN_EP1_STATE							
31																16	15								9	8							2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0								1		0				
																																		Reset					

- USB_SERIAL_JTAG_IN_EP1_STATE 输入端点 1 的状态。(RO)
- USB_SERIAL_JTAG_IN_EP1_WR_ADDR 输入端点 1 的写入数据地址。(RO)
- USB_SERIAL_JTAG_IN_EP1_RD_ADDR 输入端点 1 的读取数据地址。(RO)

Register 24.11. USB_SERIAL_JTAG_IN_EP2_ST_REG (0x0030)

(reserved)																USB_SERIAL_JTAG_IN_EP2_RD_ADDR								USB_SERIAL_JTAG_IN_EP2_WR_ADDR								USB_SERIAL_JTAG_IN_EP2_STATE							
31																16	15								9	8							2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0							1		Reset						

- USB_SERIAL_JTAG_IN_EP2_STATE 输入端点 2 的状态。(RO)
- USB_SERIAL_JTAG_IN_EP2_WR_ADDR 输入端点 2 的写入数据地址。(RO)
- USB_SERIAL_JTAG_IN_EP2_RD_ADDR 输入端点 2 的读取数据地址。(RO)

Register 24.12. USB_SERIAL_JTAG_IN_EP3_ST_REG (0x0034)

(reserved)																USB_SERIAL_JTAG_IN_EP3_RD_ADDR								USB_SERIAL_JTAG_IN_EP3_WR_ADDR								USB_SERIAL_JTAG_IN_EP3_STATE							
31																16	15								9	8							2	1	0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0							1		Reset						

- USB_SERIAL_JTAG_IN_EP3_STATE 输入端点 3 的状态。(RO)
- USB_SERIAL_JTAG_IN_EP3_WR_ADDR 输入端点 3 的写入数据地址。(RO)
- USB_SERIAL_JTAG_IN_EP3_RD_ADDR 输入端点 3 的读取数据地址。(RO)

Register 24.13. USB_SERIAL_JTAG_OUT_EP0_ST_REG (0x0038)

(reserved)																USB_SERIAL_JTAG_OUT_EP0_RD_ADDR								USB_SERIAL_JTAG_OUT_EP0_WR_ADDR								USB_SERIAL_JTAG_OUT_EP0_STATE							
31																16	15								9	8							2	1	0	Reset			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0							0							

- USB_SERIAL_JTAG_OUT_EP0_STATE** 输出端点 0 的状态。(RO)
- USB_SERIAL_JTAG_OUT_EP0_WR_ADDR** 输出端点 0 的写入数据地址。当检测到 USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 时，输出端点 0 中有 USB_SERIAL_JTAG_OUT_EP0_WR_ADDR - 2 个字节数据。(RO)
- USB_SERIAL_JTAG_OUT_EP0_RD_ADDR** 输出端点 0 的读取数据地址。(RO)

Register 24.14. USB_SERIAL_JTAG_OUT_EP1_ST_REG (0x003C)

(reserved)																USB_SERIAL_JTAG_OUT_EP1_REC_DATA_CNT																USB_SERIAL_JTAG_OUT_EP1_RD_ADDR																USB_SERIAL_JTAG_OUT_EP1_WR_ADDR																USB_SERIAL_JTAG_OUT_EP1_STATE																															
31								23								22								16								15								9								8								2								1								0																							
0								0								0								0								0								0								0								0								0								0								0								Reset							

- USB_SERIAL_JTAG_OUT_EP1_STATE** 输出端点 1 的状态。(RO)
- USB_SERIAL_JTAG_OUT_EP1_WR_ADDR** 输出端点 1 的写入数据地址。当检测到 USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 时，输出端点 1 中有 USB_SERIAL_JTAG_OUT_EP1_WR_ADDR - 2 个字节数据。(RO)
- USB_SERIAL_JTAG_OUT_EP1_RD_ADDR** 输出端点 1 的读取数据地址。(RO)
- USB_SERIAL_JTAG_OUT_EP1_REC_DATA_CNT** 当接收到 1 个数据包时输出端点 1 中的数据计数器。(RO)

Register 24.15. USB_SERIAL_JTAG_OUT_EP2_ST_REG (0x0040)

(reserved)																USB_SERIAL_JTAG_OUT_EP2_RD_ADDR								USB_SERIAL_JTAG_OUT_EP2_WR_ADDR								USB_SERIAL_JTAG_OUT_EP2_STATE							
31																16	15								9	8							2	1	0	Reset			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0							0							

USB_SERIAL_JTAG_OUT_EP2_STATE 输出端点 2 的状态。(RO)

USB_SERIAL_JTAG_OUT_EP2_WR_ADDR 输出端点 2 的写入数据地址。当检测到 USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 时，输出端点 2 中有 USB_SERIAL_JTAG_OUT_EP2_WR_ADDR - 2 个字节数据。(RO)

USB_SERIAL_JTAG_OUT_EP2_RD_ADDR 输出端点 2 的读取数据地址。(RO)

Register 24.16. USB_SERIAL_JTAG_INT_RAW_REG (0x0008)

(reserved)																								USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_RAW USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_RAW USB_SERIAL_JTAG_USB_BUS_RESET_INT_RAW USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_RAW USB_SERIAL_JTAG_STUFF_ERR_INT_RAW USB_SERIAL_JTAG_CRC16_ERR_INT_RAW USB_SERIAL_JTAG_CRC5_ERR_INT_RAW USB_SERIAL_JTAG_PID_ERR_INT_RAW USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_RAW USB_SERIAL_JTAG_SOF_INT_RAW USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_RAW																			
31																								12								11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	Reset																		

USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_RAW JTAG 输入端口 2 接收到刷写命令时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_SOF_INT_RAW 接收到 SOF 帧时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT_RAW 串口输出端点接收到 1 个数据包时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_RAW 串口输入端点为空时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_PID_ERR_INT_RAW 检测到 PID 错误时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_CRC5_ERR_INT_RAW 检测到 CRC5 错误时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_CRC16_ERR_INT_RAW 检测到 CRC16 错误时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_STUFF_ERR_INT_RAW 检测到位填充错误时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_RAW 输入端点 1 接收到一个 IN 令牌时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_USB_BUS_RESET_INT_RAW 检测到 USB 总线复位时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_RAW 输出端点 1 接收到有效载荷为 0 的数据包时，原始中断位变为高电平。(R/WTC/SS)

USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_RAW 输出端点 2 接收到有效载荷为 0 的数据包时，原始中断位变为高电平。(R/WTC/SS)

Register 24.17. USB_SERIAL_JTAG_INT_ST_REG (0x000C)

(reserved)																								USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_ST USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_ST USB_SERIAL_JTAG_USB_BUS_RESET_INT_ST USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_ST USB_SERIAL_JTAG_STUFF_ERR_INT_ST USB_SERIAL_JTAG_CRC16_ERR_INT_ST USB_SERIAL_JTAG_CRC5_ERR_INT_ST USB_SERIAL_JTAG_PID_ERR_INT_ST USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_ST USB_SERIAL_JTAG_SOF_INT_ST USB_SERIAL_JTAG_IN_FLUSH_INT_ST											
31													12	11	10	9	8	7	6	5	4	3	2	1	0										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset										

USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_ST USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_SOF_INT_ST USB_SERIAL_JTAG_SOF_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT_ST USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_ST USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT 中断的原始中断状态位。(RO)

USB_SERIAL_JTAG_PID_ERR_INT_ST USB_SERIAL_JTAG_PID_ERR_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_CRC5_ERR_INT_ST USB_SERIAL_JTAG_CRC5_ERR_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_CRC16_ERR_INT_ST USB_SERIAL_JTAG_CRC16_ERR_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_STUFF_ERR_INT_ST USB_SERIAL_JTAG_STUFF_ERR_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_ST USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_USB_BUS_RESET_INT_ST USB_SERIAL_JTAG_USB_BUS_RESET_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_ST USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT 的原始中断状态位。(RO)

USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_ST USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT 的原始中断状态位。(RO)

Register 24.18. USB_SERIAL_JTAG_INT_ENA_REG (0x0010)

(reserved)																				USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_ENA USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_ENA USB_SERIAL_JTAG_USB_BUS_RESET_INT_ENA USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_ENA USB_SERIAL_JTAG_STUFF_ERR_INT_ENA USB_SERIAL_JTAG_CRC16_ERR_INT_ENA USB_SERIAL_JTAG_CRC5_ERR_INT_ENA USB_SERIAL_JTAG_PID_ERR_INT_ENA USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_ENA USB_SERIAL_JTAG_SOF_INT_ENA USB_SERIAL_JTAG_IN_FLUSH_INT_ENA											
31													12	11	10	9	8	7	6	5	4	3	2	1	0	Reset					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										

USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_ENA USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_SOF_INT_ENA USB_SERIAL_JTAG_SOF_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT_ENA USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_ENA USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_PID_ERR_INT_ENA USB_SERIAL_JTAG_PID_ERR_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_CRC5_ERR_INT_ENA USB_SERIAL_JTAG_CRC5_ERR_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_CRC16_ERR_INT_ENA USB_SERIAL_JTAG_CRC16_ERR_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_STUFF_ERR_INT_ENA USB_SERIAL_JTAG_STUFF_ERR_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_ENA USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_USB_BUS_RESET_INT_ENA USB_SERIAL_JTAG_USB_BUS_RESET_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_ENA USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT 的中断使能位。(R/W)

USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_ENA USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT 的中断使能位。(R/W)

Register 24.19. USB_SERIAL_JTAG_INT_CLR_REG (0x0014)

(reserved)																				USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_CLR USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_CLR USB_SERIAL_JTAG_USB_BUS_RESET_INT_CLR USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_CLR USB_SERIAL_JTAG_STUFF_ERR_INT_CLR USB_SERIAL_JTAG_CRC16_ERR_INT_CLR USB_SERIAL_JTAG_CRC5_ERR_INT_CLR USB_SERIAL_JTAG_PID_ERR_INT_CLR USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_CLR USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT_CLR USB_SERIAL_JTAG_SOF_INT_CLR USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_CLR											
31													12	11	10	9	8	7	6	5	4	3	2	1	0	Reset					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											

USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT_CLR 置位清除 USB_SERIAL_JTAG_JTAG_IN_FLUSH_INT 中断。(WT)

USB_SERIAL_JTAG_SOF_INT_CLR 置位清除 USB_SERIAL_JTAG_JTAG_SOF_INT 中断。(WT)

USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT_CLR 置 位 清 除
USB_SERIAL_JTAG_SERIAL_OUT_RECV_PKT_INT 中断。(WT)

USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT_CLR 置位清除 USB_SERIAL_JTAG_SERIAL_IN_EMPTY_INT 中断。(WT)

USB_SERIAL_JTAG_PID_ERR_INT_CLR 置位清除 USB_SERIAL_JTAG_PID_ERR_INT 中断。(WT)

USB_SERIAL_JTAG_CRC5_ERR_INT_CLR 置位清除 USB_SERIAL_JTAG_CRC5_ERR_INT 中断。
(WT)

USB_SERIAL_JTAG_CRC16_ERR_INT_CLR 置位清除 USB_SERIAL_JTAG_CRC16_ERR_INT 中
断。(WT)

USB_SERIAL_JTAG_STUFF_ERR_INT_CLR 置位清除 USB_SERIAL_JTAG_STUFF_ERR_INT 中
断。(WT)

USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT_CLR 置 位 清 除
USB_SERIAL_JTAG_IN_TOKEN_REC_IN_EP1_INT 中断。(WT)

USB_SERIAL_JTAG_USB_BUS_RESET_INT_CLR 置位清除 USB_SERIAL_JTAG_USB_BUS_RESET_INT
中断。(WT)

USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT_CLR 置 位 清 除
USB_SERIAL_JTAG_OUT_EP1_ZERO_PAYLOAD_INT 中断。(WT)

USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT_CLR 置 位 清 除
USB_SERIAL_JTAG_OUT_EP2_ZERO_PAYLOAD_INT 中断。(WT)

Register 24.20. USB_SERIAL_JTAG_DATE_REG (0x0080)

31		0	
0x2007300		Reset	

USB_SERIAL_JTAG_DATE 版本控制寄存器。(R/W)

25 双线汽车接口 (TWAI)

双线车载串口 (Two-wire Automotive Interface, TWAI®) 协议是一种多主机、多播的通信协议，具有检测错误、发送错误信号以及内置报文优先仲裁等功能。TWAI 协议适用于汽车和工业应用（可参见第 25.2 章）。

ESP32-C3 包含一个 TWAI 控制器，可通过外部收发器连接到 TWAI 总线。TWAI 控制器包含一系列先进的功能，用途广泛，可用于如汽车产品、工业自动化控制、楼宇自动化等。

25.1 主要特性

ESP32-C3 TWAI 控制器具有以下特性：

- 兼容 ISO 11898-1 协议（CAN 规范 2.0）
- 支持标准格式（11-bit 标识符）和扩展格式（29-bit 标识符）两种帧格式
- 支持 1 Kbit/s ~ 1 Mbit/s 位速率
- 支持多种操作模式
 - 正常模式
 - 只听模式（不影响总线）
 - 自测模式（发送数据时不需应答）
- 64-byte 接收 FIFO
- 特殊发送
 - 单次发送（发生错误时不会自动重新发送）
 - 自发自收（TWAI 控制器同时发送和接收报文）
- 接收滤波器（支持单滤波器和双滤波器模式）
- 错误检测与处理
 - 错误计数
 - 错误报警限制可配置
 - 错误代码捕捉
 - 仲裁丢失捕捉

25.2 功能性协议

25.2.1 TWAI 性能

TWAI 协议连接网络中的两个或多个节点，并允许各节点以延迟限制的形式进行报文交互。TWAI 总线具有以下性能：

单通道通信与不归零编码： TWAI 总线只有一根传输线进行单通道通信，因此为半双工通信。同步调整也在单通道中进行，因此不需其他通道（如时钟通道和使能通道）。TWAI 上报文的位流采用不归零编码 (NRZ) 方式。

位值：单通道可处于显性状态或隐性状态，显性状态的逻辑值为 0，隐性状态的逻辑值为 1。发送显性状态数据的节点总是比发送隐性状态数据的节点优先级高。总线上的其他物理功能（如，差分电平、单线）由其各自应用实现。

位填充：TWAI 报文的某些域已经过位填充。发送器在发送连续五个位的相同值（如显性数值或隐性数值）后，需自动插入一个互补位。同理，接收到 5 个连续位的接收器应将下一个位视为填充位。位填充应用于以下域：SOF、仲裁域、控制域、数据域和 CRC 序列（可参见第 25.2.2 章）。

多播：当各节点连接到同个总线上时，这些节点都将接收到相同的位。各节点上的数据将保持一致，除非发生总线错误（可参见第 25.2.3 章）。

多主机：任意节点都可发起数据传输。如果当前已有正在进行的数据传输，则节点将等待当前传输结束后再发起其数据传输。

报文优先级与仲裁：若两个或多个节点同时发起数据传输，TWAI 协议将确保其中一个节点获得总线的优先仲裁权。各节点所发送报文的仲裁域决定了哪个节点可以获得优先仲裁。

错误检测：各节点将积极检测总线上的错误，并通过向 TWAI 总线发送错误帧来广播检测到的错误。

故障限制：各节点都维护有错误计数器，该错误计数器依据 TWAI 协议规则增加或减少。当错误计数超过一定阈值时，对应节点将自动关闭并退出网络。

可配置位速率：单个 TWAI 总线的位速率是可配置的。但是，同个总线中的所有节点须以相同位速率工作。

发送器与接收器：不论何时，任意 TWAI 节点都可作为发送器和接收器。

- 产生报文的节点为发送器。且该节点将一直作为发送器，直到总线空闲或该节点失去仲裁。请注意，仲裁期间有多个节点作为发送器。
- 所有不属于发送器的节点都将成为接收器。

25.2.2 TWAI 报文

TWAI 节点使用报文发送数据，并在监测到总线上存在错误时向其他节点发送错误信号。报文有多帧类型，不同的帧类型具有不同的帧格式。

TWAI 协议有以下帧类型：

- 数据帧
- 远程帧
- 错误帧
- 过载帧
- 帧间距

TWAI 协议有以下帧格式：

- 标准格式 (SFF) 由 11-bit 标识符组成
- 扩展格式 (EFF) 由 29-bit 标识符组成

25.2.2.1 数据帧和远程帧

节点使用数据帧向其他节点发送数据，可负载 0~8 字节数据。节点使用远程帧向其他节点请求具有相同标识符的数据帧，因此远程帧中不包含任何数据字节。但是，数据帧和远程帧中包含许多相同域。下图 25-1 所示为不

同帧类型和不同帧格式中包含的域和子域。

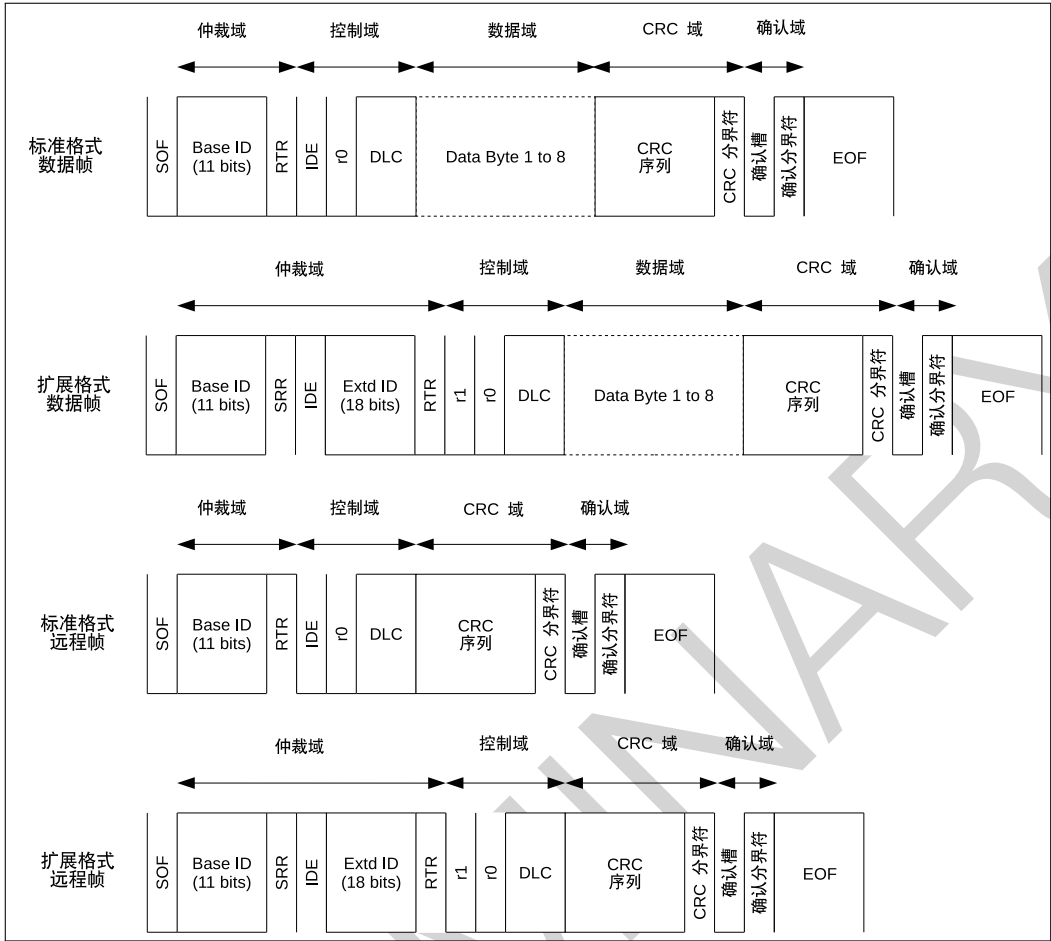


图 25-1. 数据帧和远程帧中的位域

仲裁域

当两个或多个节点同时发送数据帧和远程帧时，将根据仲裁域的位信息来决定总线上获得优先仲裁的节点。在发送仲裁域位信息时，如果一个节点在发送隐性位的同时检测到了一个显性位，这表示有其他节点优先于了这个隐性位。那么，这个发送隐性位的节点将丢失总线仲裁，应立即转为接收器。

仲裁域主要由获得最高优先发送权的帧标识符的有效位组成。根据显性位代表的逻辑值为 0，隐性位代表的逻辑值为 1，有以下规律：

- ID 值最小的帧将总是获得仲裁（逻辑 0 是显性位值）。
- 如果 ID 数值相同，由于数据帧的 RTR 位为显性位，数据帧将优先于远程帧。
- 如果 ID 的前 11 位相同，由于扩展帧的 SRR 位是隐性，因而标准格式帧将总优先于扩展格式帧。

控制域

控制域主要由数据长度代码 (DLC) 组成，DLC 表示一个数据帧中的负载的数据字节长度，或一个远程帧请求的数据字节长度。DLC 优先发送长度数值的最高有效位。

数据域

数据域中包含一个数据帧真实负载的数据字节。远程帧中不包含数据域。

CRC 域

CRC 域主要由 CRC 序列组成。CRC 序列是一个 15-bit 的循环冗余校验编码，根据数据帧或远程帧中位填充前的内容（从 SOF 到数据域末尾的所有内容）计算而来。

确认域

确认 (ACK) 域由确认槽和确认分界符组成，主要功能为：接收器向发送器报告已正确接收到有效报文。

表 25-1. 不同帧类型、帧格式下的域及子域信息

数据/远程帧	描述
SOF	帧起始 (SOF) 是一个用于同步总线上节点的单个显性位。
Base ID	基标识符 (ID.28 ~ ID.18) 是 SFF 的 11-bit 标识符，或者是 EFF 中 29-bit 标识符的前 11-bit。
RTR	远程发送请求位 (RTR) 显示当前报文是数据帧（显性）还是远程帧（隐性）。这意味着，当某个数据帧和一个远程帧有相同标识符时，数据帧始终优先于远程帧仲裁。
SRR	在 EFF 中发送替代远程请求位 (SRR)，以替代 SFF 中相同位置的 RTR 位。
IDE	标识符扩展位 (IED) 显示当前报文是 SFF（显性）还是 EFF（隐性）。这意味着，当某 SFF 帧和 EFF 帧有相同基标识符时，SFF 帧将始终优先于 EFF 帧仲裁。
Extd ID	扩展标识符 (ID.17 ~ ID.0) 是 EFF 中 29-bit 标识符的剩余 18-bit。
r1	r1（保留位 1）始终是显性位。
r0	r0（保留位 0）始终是显性位。
DLC	数据长度代码 (DLC) 为 4-bits，且为 0 ~ 8 中任一数值。数据帧使用 DLC 表示自身包含的数据字节长度。远程帧使用 DLC 表示从其他节点请求的数据字节长度。
数据字节	表示数据帧的数据负载量。该字节长度应与 DLC 的值匹配。首先发送数据字节的最高有效位 (MSB)。
CRC 序列	CRC 序列是一个 15-bit 的循环冗余校验编码。
CRC 分界符	CRC 分界符是紧随 CRC 序列的 1-bit 隐性位。
确认槽	确认槽用于接收器节点表示是否已成功接收数据帧或远程帧。发送器节点将在确认槽中发送一个隐性位，如果接收到的帧没有错误，则接收器节点将发送 1-bit 显性位替代隐性位。
确认分界符	确认分界符是紧随确认槽的 1-bit 隐性位。
EOF	帧结束 (EOF) 标志着数据帧或远程帧的结束，由七个隐性位组成。

25.2.2.2 错误帧和过载帧

错误帧

当某节点检测到总线错误时，将发送一个错误帧。错误帧由一个特殊的错误标志构成，该标志由某相同值的六个连续位组成，因而违反了位填充的规则。所以，当某节点检测到总线错误并发送错误帧时，其余节点也将相应地检测到一个填充错误并各自发送错误帧。也就是说，当发生总线错误时，通过上述过程可将该报文传递至总线上的所有节点。

当某节点检测到总线错误时，该节点将于下一个位发送错误帧。特例：如果总线错误类型为 CRC 错误，那么错误

帧将从确认分界符的下一个位开始（可参见第 25.2.3 章）。下图 25-2 所示为一个错误帧所包含的不同域：



图 25-2. 错误帧中的位域

表 25-2. 错误帧中的位域信息

错误帧	描述
错误标志	错误标志包括两种形式: 主动错误标志和被动错误标志，主动错误标志由 6 个显性位组成，被动错误标志由 6 个隐性位组成（被其他节点的显性位优先仲裁时除外）。处于主动错误状态的节点发送主动错误标志，处于被动错误状态的节点发送被动错误标志。
错误标志叠加	错误标志叠加域的主要目的是允许总线上的其他节点发送各自的主动错误标志。叠加域的范围是 0 ~ 6 位，结束标志是检测到第一个隐性位（如检测到分界符上的第一个位时）。
错误分界符	分界符域标志着错误/过载帧结束，由 8 个隐性位构成。

过载帧

过载帧与包含主动错误标志的错误帧有着相同的位信息。二者区别在于触发发送过载帧的条件。下图 25-3 所示为过载帧中包含的位域：

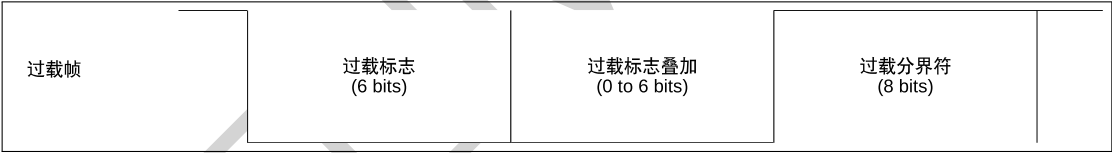


图 25-3. 过载帧中的位域

表 25-3. 过载帧中的位域信息

过载帧	描述
过载标志	由 6 个显性位构成。与主动错误标志相同。
过载标志叠加	允许其他节点发送过载标志的叠加，与错误标志叠加相似。
过载分界符	由 8 个隐性位构成。与错误分界符相同。

下列情况将触发发送过载帧：

- 1. 接收器内部要求延迟发送下一个数据帧或远程帧。
- 2. 在间歇域中的首个和第二个位上检测到显性位。
- 3. 如果在错误分界符的第八个（最后一个）位上检测到显性位。请注意，在这种情况下 TEC 和 REC 的值将不会增加（可参见第 25.2.3 章）。

由于上述情况发送过载帧时，须满足以下规定：

- 第 1 条情况下发送的过载帧只能从间歇域后的第一个位开始。
- 第 2、3 条情况下发送的过载帧须从检测到显性位的后一个位开始。
- 针对第 1 条情况，最多可生成两个过载帧。

25.2.2.3 帧间距

帧间距充当各帧之间的分隔符。数据帧和远程帧必须与前一帧用一个帧间距分隔开，不论前面的帧是何类型（数据帧、远程帧、错误帧、过载帧）。但是，错误帧和过载帧则无需与前一个帧分隔开。

下图 25-4 所示为帧间距中包含的域：

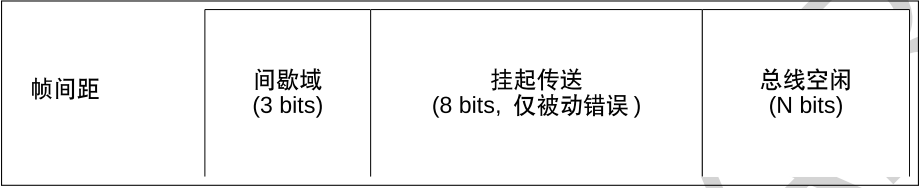


图 25-4. 帧间距中的域

表 25-4. 帧间距中的域信息

帧间距	描述
间歇域	间歇域由 3 个隐性位构成。
挂起传送	被动错误节点发送报文后，必须在帧间距中包含一个挂起传送域，由 8 个隐性位构成。主动错误节点中不含这个域。
总线空闲	总线空闲域长度任意。发送 SOF 时，总线空闲结束。若节点中有挂起传送，则 SOF 应在间歇域后的第一位发送。

25.2.3 TWAI 错误

25.2.3.1 错误类型

TWAI 中的总线错误包括以下类型：

位错误

当节点发送一个位值（显性位或隐性位）但检测到相反的位时（如，发送显性位时检测到了隐性位），就会发生位错误。但是，如果发送的位是隐性位，且位于仲裁域或确认槽或被动错误标志中，那么此时检测到显性位的话不会认定为位错误。

填充错误

当检测到相同值的 6 个连续位时（违反位填充的编码规则），发生填充错误。

CRC 错误

接收器将根据接收到的数据帧和远程帧的有效位（无填充位流）计算 CRC 值。当接收器计算的值与接收到的数据帧和远程帧中的 CRC 序列不匹配时，会发生 CRC 错误。

格式错误

当某个报文中的固定格式位中包含非法位时，可检测到格式错误。比如，r1 和 r0 域必须固定为显性。

确认错误

当发送器无法在确认槽中检测到显性位时，将发生确认错误。

25.2.3.2 错误状态

每个节点 TIAI 控制器通过维护两个错误计数器来实现故障界定，计数数值决定错误状态。这两个错误计数器分别为：发送错误计数 (TEC) 和接收错误计数 (REC)。TIAI 包含以下错误状态。

主动错误

处于主动错误状态的节点可参与到总线交互中，且在检测到错误时可以发送主动错误标志。

被动错误

处于被动错误状态的节点可参与到总线交互中，但在检测到错误时只能发送一次被动错误标志。被动错误节点发送数据帧或远程帧后，需在后续的帧间距中增加挂起传送域。

离线

禁止处于离线状态的节点以任意方式干扰总线（如，不允许其进行数据传输）。

25.2.3.3 错误计数

TEC 和 REC 根据以下规则递增/递减。**请注意，一条报文传输中可应用多个规则。**

1. 当接收器检测到错误时，REC 数值将增加 1。当检测到的错误为发送主动错误标志或过载标志期间的位错误除外。
2. 发送错误标志后，当接收器第一个检测到的位是显性位时，REC 数值将增加 8。
3. 当发送器发送错误标志时，TEC 数值增加 8。但是，以下情况不适用于该规则：
 - 发送器为被动错误状态，因为在应答槽未检测到显性位而产生应答错误，且在发送被动错误标志时检测到显性位时，则 TEC 数值不应增加。
 - 发送器在仲裁期间因填充错误而发送错误标志，且填充位本该是隐性位但是检测到显性位，则 TEC 数值不应增加。
4. 若发送器在发送主动错误标志和过载标志时检测到位错误，则 TEC 数值增加 8。
5. 若接收器在发送主动错误标志和过载标志时检测到位错误，则 REC 数值增加 8。
6. 任意节点在发送主动/被动错误标志或过载标志后，节点仅能承载最多 7 个连续显性位。在（发送主动错误标志或过载标志时）检测到第 14 个连续显性位，或在被动错误标志后检测到第 8 个连续显性位后，发送器将使其 TEC 数值增加 8，而接收器将使其 REC 数值增加 8。每增加 8 个连续显性位的同时，（发送器的）TEC 和（接收器的）REC 数值也将增加 8。
7. 每当发送器成功发送报文后（接收到 ACK，且直到 EOF 完成未发生错误），TEC 数值将减小 1，除非 TEC 的数值已经为 0。
8. 当接收器成功接收报文后（确认槽前未检测到错误，且成功发送 ACK），则 REC 数值将相应减小。
 - 若 REC 数值位于 1 ~ 127 之间，则其值减小 1。
 - 若 REC 数值大于 127，则其值减小到 127。
 - 若 REC 数值为 0，则仍保持为 0。

- 9. 当一个节点的 TEC 和/或 REC 数值大于等于 128 时，该节点变为被动错误节点。导致节点发生上述状态切换的错误，该节点仍发送主动错误标志。请注意，一旦 REC 数值到达 128，后续任何增加该值的动作都是无效的，直到 REC 数值返回到 128 以下。
- 10. 当某节点的 TEC 数值大于等于 256 时，该节点将进入离线状态。
- 11. 当某被动错误节点的 TEC 和 REC 数值都小于等于 127，则该节点将变为主动错误节点。
- 12. 当离线节点在总线上检测到 128 次 11 个连续隐性位后，该节点可变为主动错误节点（TEC 和 REC 数值都重设为 0）。

25.2.4 TWAI 位时序

25.2.4.1 标称位

TWAI 协议允许 TWAI 总线以特定的位速率运行。但是，总线内的所有节点必须以统一位速率运行。

- 标称位速率为每秒发送比特数量。
- 标称位时间为 $1/\text{标称位速率}$ 。

每个标称位时间中含多个段，每段由多个时间定额 (Time Quanta) 组成。**时间定额**为最小时间单位，作为一种预分频时钟信号应用于各个节点中。下图 25-5 所示为一个标称位时间内所包含的段。

TWAI 控制器将在以一个时间定额的步长进行操作，每个时间定额中都会分析 TWAI 的总线状态。如果两个连续的时间定额中总线状态不同（隐性-显性，或反之），意味着有边沿产生。PBS1 和 PBS2 的交点将被视为采样点，采样的总线状态即为这个位的数值。

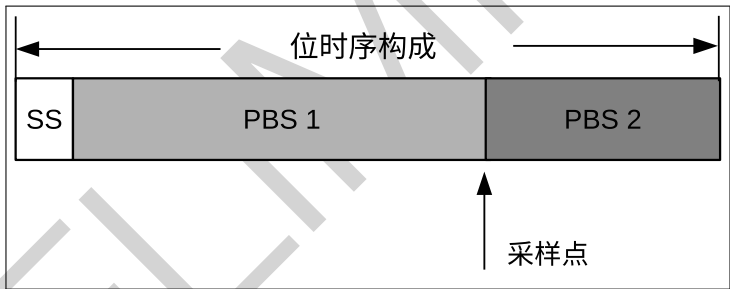


图 25-5. 标称位构成

表 25-5. 名义位时序中包含的段

段	描述
同步段 (SS)	SS（同步段）的长度为 1 个时间定额。若所有节点都同步，则位边沿发生时应位于该段内。
缓冲时期段 1 (PBS1)	PBS1 的长度可为 1 ~ 16 个时间定额，用于补偿网络中的物理延迟时间。可通过增加 PBS1 的长度来实现同步。
缓冲时期段 2 (PBS2)	PBS2 的长度可为 1 ~ 8 个时间定额，用于补偿节点中的信息处理时间。可通过缩短 PBS2 的长度来实现同步。

25.2.4.2 硬同步与再同步

由于时钟偏移和抖动，同一总线上节点的位时序可能会脱离相位段。因而，位边沿可能会偏移到同步段的前后。针对上述位边沿偏移的问题 TWAI 提供多种同步方式。设发生相位错误时位边沿偏移的 TQ（时间定额）数量为

“e”，该值与 SS 相关。

- 主动相位错误 ($e > 0$)：位边沿位于同步段之后采样点之前（即，边沿向后偏移）。
- 被动相位错误 ($e < 0$)：位边沿位于前个位的采样点之后同步段之前（即，边沿向前偏移）。

为解决相位错误，有两种同步方式，**硬同步与再同步**。**硬同步与再同步**遵守以下规则：

- 单个位时序中仅可执行一次同步。
- 同步仅可发生在隐性位到显性位的边沿上。

硬同步

总线空闲期间，硬同步发生在隐性位到显性位的变化边沿上（如总线空闲后的第一个 SOF 位）。此时，所有节点都将重启其内部时序，从而使该变化边沿位于重启位时序的同步段内。

再同步

非总线空闲期间，再同步发生在隐性位到显性位的变化边沿上。如果边沿上有主动相位错误 ($e > 0$)，则增加 PBS1 段的长度。如果边沿上有被动相位错误 ($e < 0$)，则减小 PBS2 段的长度。

PBS1/PBS2 具体增加和减小的时间定额取决于相位错误的绝对值，同时也受可配置的同步跳宽 (SJW) 数值限制。

- 当相位错误的绝对值小于等于 SJW 数值时，PBS1/PBS2 将增加/减小 e 个时间定额。该过程与硬同步具有相同效果。
- 当相位错误的绝对值大于 SJW 数值时，PBS1/PBS2 将增加/减小与 SJW 相同数值的时间定额。这意味着，在完全解决相位错误之前，可能需要执行多次位的再同步。

25.3 结构概述

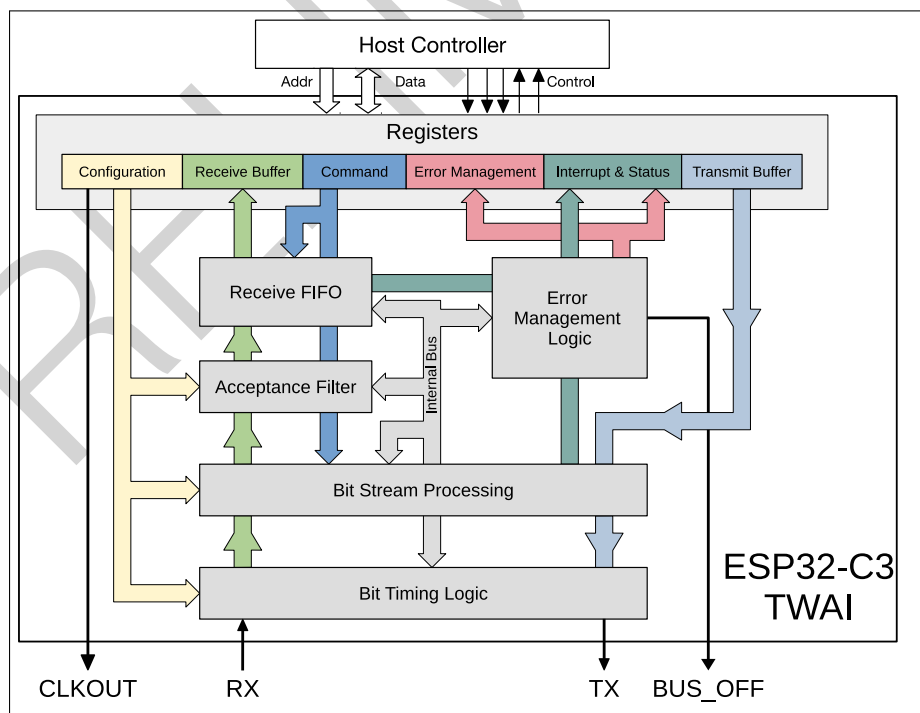


图 25-6. TWAI 概略图

TWAI 控制器的主要功能模块如图 25-6。

25.3.1 寄存器模块

ESP32-C3 的 CPU 使用 32-bit 字对齐地址访问外设。但对于 TWAI 控制器中的大部分寄存器，仅最低有效字节 (bits [7:0]) 数据有效。在这些寄存器中，bits [31:8] 数值在写入时被忽略，在读取时返回 0。

配置寄存器

配置寄存器存储 TWAI 控制器的各配置项，如位速率、操作模式、接收滤波器等。只有在 TWAI 控制器处于复位模式时，才可修改配置寄存器（可参见第 25.4.1 章）。

指令寄存器

CPU 通过指令寄存器驱动 TWAI 控制器执行任务，如发送报文或清除接收缓冲器。只有在 TWAI 控制器处于操作模式时，才可修改指令寄存器（可参见第 25.4.1 章）。

中断 & 状态寄存器

中断寄存器显示 TWAI 控制器中发生的事件（每个事件由一个单独的位表示）。状态寄存器显示 TWAI 控制器的当前状态。

错误管理寄存器

错误管理寄存器包括错误计数和捕捉寄存器。错误计数寄存器表示 TEC 和 REC 的数值。捕捉寄存器负责记录相关信息，如 TWAI 控制器在何处检测到总线错误，或何时丢失仲裁。

发送缓冲寄存器

发送缓冲器大小为 13 字节，用于存储 TWAI 的待发送报文。

接收缓冲寄存器

接收缓冲器大小为 13 字节，用于存储单个报文。接收缓冲器是读取接收 FIFO 的窗口，接收 FIFO 中的第一个报文将被映射到接收缓冲器中。

请注意，发送缓冲寄存器、接收缓冲寄存器和接收滤波配置寄存器的地址范围重叠，地址偏移涉及 0x0040 ~ 0x0070。上述地址范围内寄存器遵循以下规则：

- 当 TWAI 控制器处于复位模式时，该地址范围被映射到接收滤波配置寄存器。
- TWAI 控制器处于操作模式时：
 - 对地址范围的所有读取都映射到接收缓冲寄存器中。
 - 对地址范围的所有写入都映射到发送缓冲寄存器中。

25.3.2 位流处理器

位流处理 (BSP) 模块负责对发送缓冲器的数据进行帧处理 (如，位填充和附加 CRC 域) 并为位时序逻辑 (BTL) 模块生成位流。同时，BSP 模块还负责处理从 BTL 模块中接收的位流 (如，去填充和验证 CRC)，并将处理后报文写入接收 FIFO。BSP 还负责检测 TWAI 总线上的错误并将此类错误报告给错误管理逻辑 (EML)。

25.3.3 错误管理逻辑

错误管理逻辑 (EML) 模块负责更新 TEC 和 REC 数值，记录错误信息 (如，错误类型和错误位置)，更新控制器的错误状态，确保 BSP 模块发送正确的错误标志。此外，该模块还负责记录 TWAI 控制器丢失仲裁时的 bit 位置。

25.3.4 位时序逻辑

位时序逻辑 (BTL) 模块负责以预先配置的位速率发送和接受报文。BTL 模块还负责同步位时序，确保数据传输的稳定性。位速率由多个可编程的段组成，且用户可设置每个段的 TQ (时间定额) 长度，来调整传播延迟、控

制器处理时间等因素。

25.3.5 接收滤波器

接收滤波器是一个可编程的报文过滤单元，允许 TWAI 控制器根据报文的标识符域接收或拒绝该报文。通过接收滤波器的报文才能被存储到接收 FIFO 中。用户可配置接收滤波器的模式：单滤波器、双滤波器。

25.3.6 接收 FIFO

接收 FIFO 是大小为 64-byte 的缓冲器（位于 TWAI 控制器内部），负责存储通过接收滤波器的接收报文。接收 FIFO 中存储的报文大小可以不同（范围在 3 ~ 13 byte 之间）。当接收 FIFO 为满时（或剩余的空间不足以完全存储下一个接收报文），将触发溢出中断，后续的接收报文将丢失，直到接收 FIFO 中清除出足够的存储空间。接收 FIFO 中的第一条报文将被映射到 13-byte 的接收缓冲器中，直到该报文被清除（通过释放接收缓冲器指令）。清除后，接收 FIFO 将释放上一条已清除报文的空間，同时将窗口映射到接收 FIFO 中的下一条报文。

25.4 功能描述

25.4.1 模式

ESP32-C3 TWAI 控制器有两种工作模式：复位模式和操作模式。将 `TWAI_RESET_MODE` 位置 1，进入复位模式；置 0，进入操作模式。

25.4.1.1 复位模式

要修改 TWAI 控制器的各种配置寄存器，需进入复位模式。进入复位模式时，TWAI 控制器彻底与 TWAI 总线断开连接。复位模式下，TWAI 控制器将无法发送任何报文（包括错误信号）。任何正在进行的报文传输将立即被终止。同样的，TWAI 控制器在该模式下也将无法接收任何报文。

25.4.1.2 操作模式

进入操作模式后，TWAI 控制器与总线相连，并且写保护各配置寄存器，以确保控制器的配置在运行期间保持一致。操作模式下，TWAI 控制器可以发送和接收报文（包括错误信号），但具体取决于 TWAI 控制器配置于哪种运行子模式。TWAI 控制器支持以下三种子模式：

- **正常模式：** TWAI 控制器可以发送和接收包含错误信号在内的报文（如，错误帧和过载帧）。
- **自测模式：** 与正常模式相同，但在该模式下，TWAI 控制器发送报文时，即使在 CRC 域之后没有接收到应答信号，也不会产生应答错误。通常在单个 TWAI 控制器自测时使用该模式。
- **只听模式：** TWAI 控制器可以接收报文，但在 TWAI 总线上保持完全被动。因此，TWAI 控制器将无法发送任何报文、应答或错误信号。错误计数将保持冻结状态。该模式用于 TWAI 总线监控。

请注意，退出复位模式后（如，进入操作模式时），TWAI 控制器需等待 11 个连续隐性位出现，才能完全连接上 TWAI 总线（即，可以发送或接收报文）。

25.4.2 位时序

TWAI 控制器的工作位速率必须在控制器处于复位模式时进行配置。在寄存器

`TWAI_BUS_TIMING_0_REG` 和 `TWAI_BUS_TIMING_1_REG` 中配置位速率，这两个寄存器包含以下域：

下表 25-6 所示为 `TWAI_BUS_TIMING_0_REG` 包含的位域。

表 25-6. TWAI_BUS_TIMING_0_REG 的 bit 信息 (0x18)

Bit 31-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 1	Bit 0
保留	SJW.1	SJW.0	保留	BRP.12	BRP.1	BRP.0

说明:

- 预分频值 (BRP): TWAI 时间定额时钟由 APB 时钟分频得到, APB 时钟通常为 80 MHz。可通过以下公式计算分频数值, 其中 t_{Tq} 为时间定额的时钟周期, t_{CLK} 为 APB 时钟周期:

$$t_{Tq} = 2 \times t_{CLK} \times (2^{12} \times BRP.12 + 2^{11} \times BRP.11 + \dots + 2^1 \times BRP.1 + 2^0 \times BRP.0 + 1)$$

- 同步跳宽 (SJW): SJW 数值在 SJW.0 和 SJW.1 中配置, 计算公式为: $SJW = (2 \times SJW.1 + SJW.0 + 1)$ 。

下表 25-7 所示为 TWAI_BUS_TIMING_1_REG 包含的位域。

表 25-7. TWAI_BUS_TIMING_1_REG 的 bit (0x1c)

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	SAM	PBS2.2	PBS2.1	PBS2.0	PBS1.3	PBS1.2	PBS1.1	PBS1.0

说明:

- PBS1: 根据以下公式计算缓冲时期段 1 中的时间定额数量: $(8 \times PBS1.3 + 4 \times PBS1.2 + 2 \times PBS1.1 + PBS1.0 + 1)$ 。
- PBS2: 根据以下公式计算缓冲时期段 2 中的时间定额数量: $(4 \times PBS2.2 + 2 \times PBS2.1 + PBS2.0 + 1)$ 。
- SAM: 该值置 1 启动三点采样。用于低/中速总线, 有利于过滤总线上的尖峰信号。

25.4.3 中断管理

ESP32-C3 TWAI 控制器提供了八种中断, 每种中断由寄存器 TWAI_INT_RAW_REG 中的一个位表示。要触发某个特定的中断, 须设置 TWAI_INT_ENA_REG 中相应的使能位。

TWAI 控制器提供了以下八种中断:

- 接收中断
- 发送中断
- 错误报警中断
- 数据溢出中断
- 被动错误中断
- 仲裁丢失中断
- 总线错误中断
- 总线状态中断

只要在 TWAI_INT_RAW_REG 一个或多个中断位为 1, TWAI 控制器中的中断信号即为有效, 当 TWAI_INT_RAW_REG 中的所有位都被清除时, TWAI 控制器中的中断信号则失效。读操作访问寄存器 TWAI_INT_RAW_REG 后, 其中

的大多数中断位将自动清除。但是，只有通过 `TWAI_RELEASE_BUF` 指令位清除所有接收报文后，接收中断位才能被清除。

25.4.3.1 接收中断 (RXI)

当 TWAI 接收 FIFO 中有待读取报文时 (`TWAI_RX_MESSAGE_CNT_REG` > 0), 都会触发 RXI。 `TWAI_RX_MESSAGE_CNT_REG` 中记录的报文数量包括接收 FIFO 中的有效报文和溢出报文。直到通过 `TWAI_RELEASE_BUF` 指令位清除所有挂起接收报文后，RXI 才会失效。

25.4.3.2 发送中断 (TXI)

当发送缓冲器由锁定状态变为空闲状态，将会触发 TXI。此时软件可以将其他报文加载到发送缓冲器中等待发送。以下几种情况都会触发该中断：

- 报文发送已成功完成（如，应答未发现错误）。任何发送失败将自动重发。
- 单次发送已完成（`TWAI_TX_COMPLETE` 位指示发送成功与否）。
- 使用 `TWAI_ABORT_TX` 指令位终止报文发送。

25.4.3.3 错误报警中断 (EWI)

每当寄存器 `TWAI_STATUS_REG` 中 `TWAI_ERR_ST` 和 `TWAI_BUS_OFF_ST` 的位值改变时（如，从 0 变为 1 或反之），都会触发 EWI。根据 EWI 触发时 `TWAI_ERR_ST` 和 `TWAI_BUS_OFF_ST` 的值分成以下几种情况：

- 如果 `TWAI_ERR_ST` = 0 且 `TWAI_BUS_OFF_ST` = 0：
 - 如果 TWAI 控制器处于主动错误状态，则表示 TEC 和 REC 的值都返回到了 `TWAI_ERR_WARNING_LIMIT_REG` 所设的阈值之下。
 - 如果 TWAI 控制器此前正处于总线恢复状态，则表示此时总线恢复已成功完成。
- 如果 `TWAI_ERR_ST` = 1 且 `TWAI_BUS_OFF_ST` = 0：表示 TEC 或 REC 数值已超过 `TWAI_ERR_WARNING_LIMIT_REG` 所设的阈值。
- 如果 `TWAI_ERR_ST` = 1 且 `TWAI_BUS_OFF_ST` = 1：表示 TWAI 控制器已进入 BUS_OFF 状态（因 TEC >= 256）。
- 如果 `TWAI_ERR_ST` = 0 且 `TWAI_BUS_OFF_ST` = 1：表示 BUS_OFF 恢复期间，TWAI 控制器的 TEC 数值已低于 `TWAI_ERR_WARNING_LIMIT_REG` 所设的阈值。

25.4.3.4 数据溢出中断 (DOI)

每当接收 FIFO 中有溢出发生时，都会触发 DOI。DOI 表示接收 FIFO 已满且应立即进行读取，以防出现更多溢出报文。

只有导致接收 FIFO 溢出的第一条报文可触发 DOI（如，当接收 FIFO 从未满变为开始溢出时）。任意后续的溢出报文将不会再次重复触发 DOI。只有当所有接收的（有效报文或溢出）报文都被读取后，才能再次触发 DOI。

25.4.3.5 被动错误中断 (TXI)

每当 TWAI 控制器从主动错误变为被动错误，或反之，都会触发 EPI。

25.4.3.6 仲裁丢失中断 (ALI)

每当 TWAI 控制器尝试发送报文且丢失仲裁时，都会触发 ALI。TWAI 控制器丢失仲裁的 bit 位置将自动记录在仲裁丢失捕捉寄存器 (TWAI_ARB_LOST_CAP_REG) 中。仲裁丢失捕捉寄存器被清除（通过 CPU 读取该寄存器）之前，将不会再记录新发生的仲裁失败时的 bit 位置。

25.4.3.7 总线错误中断 (BEI)

每当 TWAI 控制器在 TWAI 总线上检测到错误时，都会触发 BEI。发生总线错误时，总线错误的类型和发生错误时的 bit 位置都将自动记录在错误捕捉寄存器 (TWAI_ERR_CODE_CAP_REG) 中。错误捕捉寄存器被清除（通过 CPU 的读取）之前，将不会再记录新的总线错误信息。

25.4.3.8 总线状态中断 (BSI)

每当 TWAI 控制器在收发总线数据状态与空闲状态之间切换时，都会触发 BSI。当该中断发生时，可通过读取 TWAI_STATUS_REG 寄存器 TWAI_RX_ST 和 TWAI_TX_ST 两个域来判断 TWAI 控制器当前的状态。

25.4.4 发送缓冲器与接收缓冲器

25.4.4.1 缓冲器概述

表 25-8. SFF 与 EFF 的缓冲器布局

标准格式 (SFF)		扩展格式 (EFF)	
TWAI 地址	内容	TWAI 地址	内容
0x40	TX/RX 帧信息	0x40	TX/RX 帧信息
0x44	TX/RX identifier 1	0x44	TX/RX identifier 1
0x48	TX/RX identifier 2	0x48	TX/RX identifier 2
0x4c	TX/RX data byte 1	0x4c	TX/RX identifier 3
0x50	TX/RX data byte 2	0x50	TX/RX identifier 4
0x54	TX/RX data byte 3	0x54	TX/RX data byte 1
0x58	TX/RX data byte 4	0x58	TX/RX data byte 2
0x5c	TX/RX data byte 5	0x5c	TX/RX data byte 3
0x60	TX/RX data byte 6	0x60	TX/RX data byte 4
0x64	TX/RX data byte 7	0x64	TX/RX data byte 5
0x68	TX/RX data byte 8	0x68	TX/RX data byte 6
0x6c	保留	0x6c	TX/RX data byte 7
0x70	保留	0x70	TX/RX data byte 8

表 25-8 所示为发送缓冲器和接收缓冲器的寄存器布局。发送和接收缓冲寄存器的访问地址范围相同，且只有当 TWAI 控制器处于操作模式时才可访问。CPU 的写入操作将访问发送缓冲寄存器，CPU 的读取操作将访问接收缓冲寄存器。发送缓冲器和接收缓冲器中存储报文（接收报文或待发送报文）的寄存器布局和域完全一致。

发送缓冲寄存器用于配置 TWAI 的待发送报文。CPU 可以在发送缓冲寄存器进行写入操作，指定报文的帧类型、帧格式、帧 ID 和帧数据（有效载荷）。一旦发送缓冲器配置完成后，CPU 可以将 TWAI_CMD_REG 中的 TWAI_TX_REQ 位置 1，以开始报文发送。

- 若是自发自收请求，变更为将 TWAI_SELF_RX_REQ 置 1。

- 若是单次发送，需要同时将 `TWAI_TX_REQ` 和 `TWAI_ABORT_TX` 置 1。

接收缓冲寄存器将映射接收 FIFO 中的第一条报文。CPU 可以对接收缓冲寄存器执行读取操作，获取第一条报文的帧类型、帧格式、帧 ID 和帧数据（有效载荷）。读取完接收缓冲寄存器中的报文后，CPU 通过将 `TWAI_CMD_REG` 中的 `TWAI_RELEASE_BUF` 位置 1 来清除接收缓冲寄存器，若接收 FIFO 中仍有待处理的报文，按照接收报文的先后次序依次将接收到的报文映射到接收缓冲寄存器。

25.4.4.2 帧信息

帧信息的长度为 1-byte，主要用于明确报文的帧类型、帧格式以及数据长度。下表 25-9 所示为帧信息域。

表 25-9. TX/RX 帧信息 (SFF/EFF); TWAI 地址 0x40

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	FF	RTR	X	X	DLC.3	DLC.2	DLC.1	DLC.0

说明：

1. FF: 主要明确某报文属于 EFF 还是 SFF。当 FF 位为 1 时，该报文为 EFF，当 FF 位为 0 时，该报文为 SFF。
2. RTR: 主要明确某报文是数据帧还是远程帧。当 RTR 位为 1 时，该报文为远程帧，当 RTR 位为 0 时，该报文为数据帧。
3. X: 无关 bit，可以是任意值。
4. DLC: 主要明确数据帧中的数据字节数量，或从远程帧中请求的数据字节数量。TWAI 数据帧的最大载荷为 8 个数据字节，因此 DLC 的数值范围应是 0 ~ 8。

25.4.4.3 帧标识符

若报文为 SFF，则对应的帧标识符域为 2-bytes (11-bits)；若报文为 EFF，则对应的帧标识符域为 4-bytes (29-bits)。

下表 Table 25-10 ~ 25-11 所示为 SFF (11-bits) 报文的帧标识符域。

表 25-10. TX/RX 标识符 1 (SFF); TWAI 地址 0x44

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3

表 25-11. TX/RX 标识符 2 (SFF); TWAI 地址 0x48

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.2	ID.1	ID.0	X ¹	X ²	X ²	X ²	X ²

说明：

1. 无关项。建议设置为与接收缓冲器兼容（设为 RTR），以防需使用自接收功能（或与自接收功能一起使用）。
2. 无关项。建议设置为与接收缓冲器兼容（设为 0），以防需使用自接收功能（或与自接收功能一起使用）。

下表 25-12 ~ 25-15 所示为 EFF (29-bits) 报文的帧标识符域。

表 25-12. TX/RX 标识符 1 (EFF); TWAI 地址 0x44

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

表 25-13. TX/RX 标识符 2 (EFF); TWAI 地址 0x48

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

表 25-14. TX/RX 标识符 3 (EFF); TWAI 地址 0x4c

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

表 25-15. TX/RX 标识符 4 (EFF); TWAI 地址 0x50

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ID.4	ID.3	ID.2	ID.1	ID.0	X ¹	X ²	X ²

说明:

1. 无关项。建议设置为与接收缓冲器兼容 (设为 RTR)，以防需使用自接收功能 (或与自接收功能一起使用)。
2. 无关项。建议设置为与接收缓冲器兼容 (设为 0)，以防需使用自接收功能 (或与自接收功能一起使用)。

25.4.4.4 帧数据

帧数据域包含发送或接收的数据帧，范围为 0 ~ 8 bytes。其中的有效字节数应与 DLC 相同。但是，如果 DLC 数值大于 8，则帧数据域的有效字节数仍为 8。远程帧中不包含数据载荷，因此不存在帧数据域。

比如，当发送 5 个字节的数据帧时，CPU 应在 DLC 域中写入数值 5，并将数据写入数据域 1 ~ 5 字节对应的寄存器。同样，当接收 DLC 为 5 的数据帧时，只有 1 ~ 5 数据字节中包含 CPU 可以读取的有效载荷数据。

25.4.5 接收 FIFO 和数据溢出

接收 FIFO 是一个 64-byte 的内部缓冲器，用于以先进先出的原则存储接收到的报文。一条接收报文可在接收 FIFO 中占 3 ~ 13 bytes 空间，且其中字节序与接收缓冲器的寄存器地址顺序相同。接收缓冲寄存器将被映射到接收 FIFO 中第一条报文。

当 TWAI 控制器接收到一条报文时，TWAI_RX_MESSAGE_COUNTER 的值将增加 1，最大值为 64。如果接收 FIFO 中有足够的剩余空间，报文内容将被写入到接收 FIFO 中。读取接收缓冲器中的消息后，通过将 TWAI_RELEASE_BUF 的位置 1，释放接收 FIFO 第一条报文所占的空间，TWAI_RX_MESSAGE_COUNTER 的值也将减小 1。然后，接收缓冲器将映射接收 FIFO 中的下一条报文。

当 TWAI 控制器接收到一条报文，但接收 FIFO 没有足够空间完整地存储这条接收报文时（不论是因为报文内容大小大于接收 FIFO 中的空闲空间，还是因为接收 FIFO 已满），便会发生数据溢出。

数据溢出发生时：

- 接收 FIFO 中剩余的空间将填满溢出报文的内容。如果接收 FIFO 已满，则无法存储溢出报文的任何内容。
- 接收 FIFO 首次发生数据溢出时，将触发数据溢出中断。
- 溢出报文仍将增加 `TWAI_RX_MESSAGE_COUNTER` 的值到最大值 64。
- 接收 FIFO 将在内部将溢出报文标记为无效。可使用 `TWAI_MISS_ST` 位，确认目前接收缓冲器映射的报文是有效报文还是溢出报文。

为了清除接收 FIFO 中的溢出报文，应重复调用 `TWAI_RELEASE_BUF`，直到 `TWAI_RX_MESSAGE_COUNTER` 为 0。这样可以读取接收 FIFO 中的所有有效报文，并清除所有溢出报文。

25.4.6 接收滤波器

接收滤波器允许 TWAI 控制器根据报文 ID 过滤接收报文（有时可以过滤报文的第一个数据字节和帧类型）。只有通过过滤的报文才能存储到接收 FIFO 中。接收滤波器的使用可以一定程度地减轻 TWAI 控制器的运行负荷（如，可减少使用接收 FIFO 和发生接收中断的次数），因为 TWAI 控制器将只需要操作过滤后的一小部分报文。

接收滤波的配置寄存器和发送/接收缓冲寄存器使用相同的访问地址空间，只有当 TWAI 控制器处于复位模式时，该地址段才被映射到接收滤波器的配置寄存器。

接收滤波器的配置寄存器由 32-bit 的 Code 值和 32-bit 的 Mask 值组成。Code 值将指定一种位排列模式，每条过滤报文中的位都必须匹配该模式，才能使该报文通过过滤。Mask 值可屏蔽 Code 值中的某些位（屏蔽位将被设置为“不相关”位）。如图 25-7 所示，为了使报文通过过滤，每条过滤报文的 ID 位都必须匹配 Code 值所设模式或者被 Mask 值屏蔽。

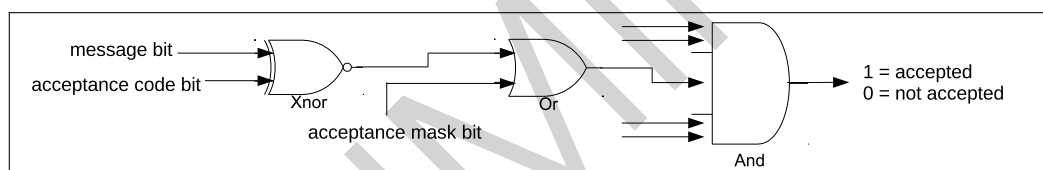


图 25-7. 接收滤波器

TWAI 控制器的接收滤波器允许 32-bit 的 Code 值和 Mask 值定义单个滤波器（单滤波模式），或两个滤波器（双滤波模式）。接收滤波器如何解析 32-bit 的 code 值和 mask 值，取决于滤波模式以及接收报文的格式（如，SFF 还是 EFF）。

25.4.6.1 单滤波模式

将 `TWAI_RX_FILTER_MODE` 的位置 1，可启动单滤波模式。此后，32-bit code/mask 的值将定义单个滤波器。

单个滤波器可过滤数据帧和远程帧中的以下位：

- SFF
 - 11-bit ID 整体
 - RTR bit
 - 数据字节 1 和数据字节 2
- EFF
 - 29-bit ID 整体
 - RTR bit

下图 25-8 所示为单滤波模式下如何解析 32-bit code/mask 的值。

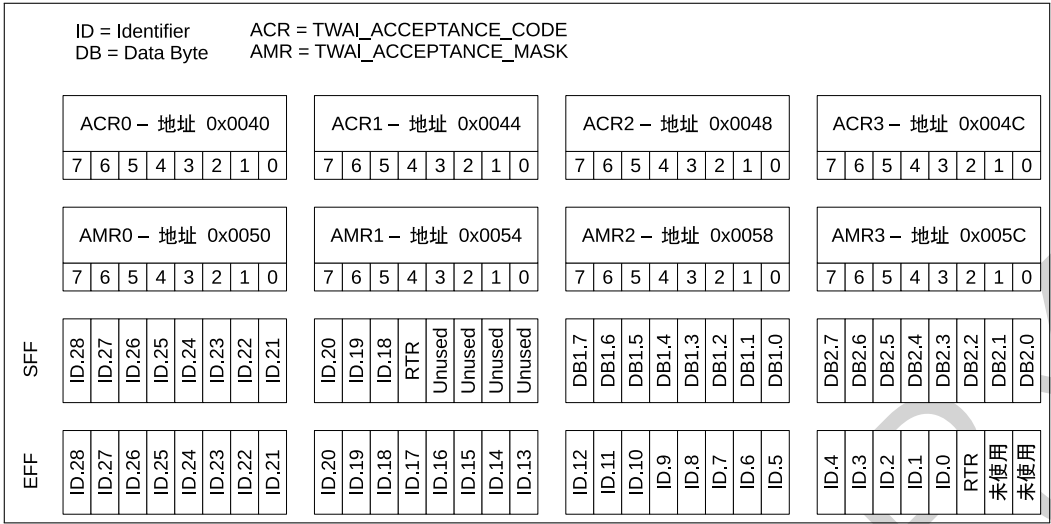


图 25-8. 单滤波模式

25.4.6.2 双滤波模式

将 `TWAI_RX_FILTER_MODE` 的位置 0，可启动双滤波模式。此后，32-bit code/mask 的值将定义两个滤波器之一，即滤波器 1 或滤波器 2。双滤波模式下，如果报文通过这两个滤波器中的至少一个，则表示该报文已成功通过过滤。

这两个滤波器可以过滤数据帧和远程帧中的以下位：

- SFF
 - 11-bit ID 整体
 - RTR bit
 - 数据字节 1 (仅适用于滤波器 1)
- EFF
 - 29-bit ID 的前 16-bit

下图 25-9 所示为双滤波模式下如何解析 32-bit code/mask 的值。

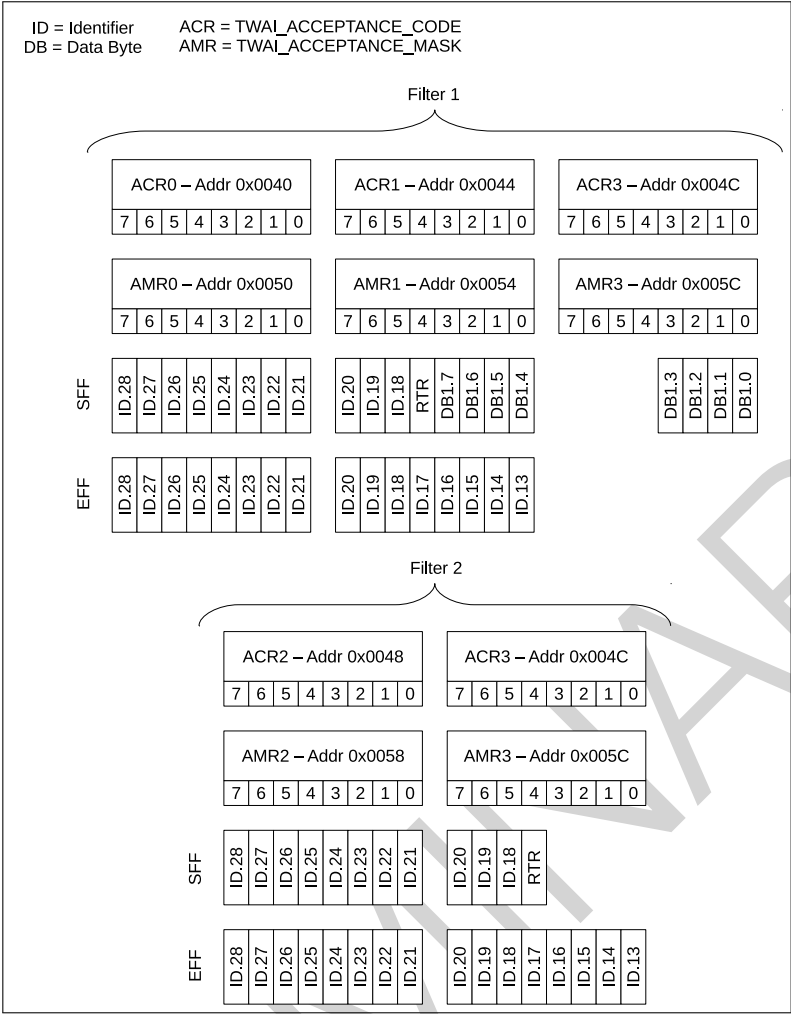


图 25-9. 双滤波模式

25.4.7 错误管理

TWAI 协议要求每个 TWAI 节点中都包含发送错误计数器 (TEC) 和接收错误计数器 (REC)。这两个错误计数的数值决定了 TWAI 控制器当前的错误状态 (如, 主动错误、被动错误、离线)。TWAI 控制器将 TEC 和 REC 的数值分别存储在 [TWAI_TX_ERR_CNT_REG](#) 和 [TWAI_RX_ERR_CNT_REG](#) 中, CPU 可随时进行读取。除了错误状态之外, TWAI 控制器还提供错误报警限制 (EWL) 的功能, 这个功能可在 TWAI 控制器进入被动错误状态之前, 提醒用户当前发生的严重总线错误。

TWAI 控制器的当前错误状态通过以下各数值和状态位体现, 即: TEC、REC、[TWAI_ERR_ST](#) 和 [TWAI_BUS_OFF_ST](#)。这些数值和状态位的变化也将触发中断, 从而提醒用户当前的错误状态变化 (可参见第 25.4.3 章)。下图 25-10 所示为错误状态、上述数值和状态位以及错误状态相关中断之间的关系。

25.4.7.1 错误报警限制

错误报警限制 (EWL) 为 TEC 和 REC 的可配置阈值, 若错误计数数值超过该阈值, 将触发 EWI 中断。EWL 将作为一个报警功能提示当前发生的严重 TWAI 总线错误, 且在 TWAI 控制器进入被动错误状态之前被触发。EWL 数值应在寄存器 [TWAI_ERR_WARNING_LIMIT_REG](#) 中进行配置, 配置同时 TWAI 控制器必须处于复位模式下。[TWAI_ERR_WARNING_LIMIT_REG](#) 默认数值为 96。

当 TEC 和/或 REC 数值大于等于 EWL 数值时, [TWAI_ERR_ST](#) 位将立即被置 1。同理, 当 TEC 和 REC 数值都

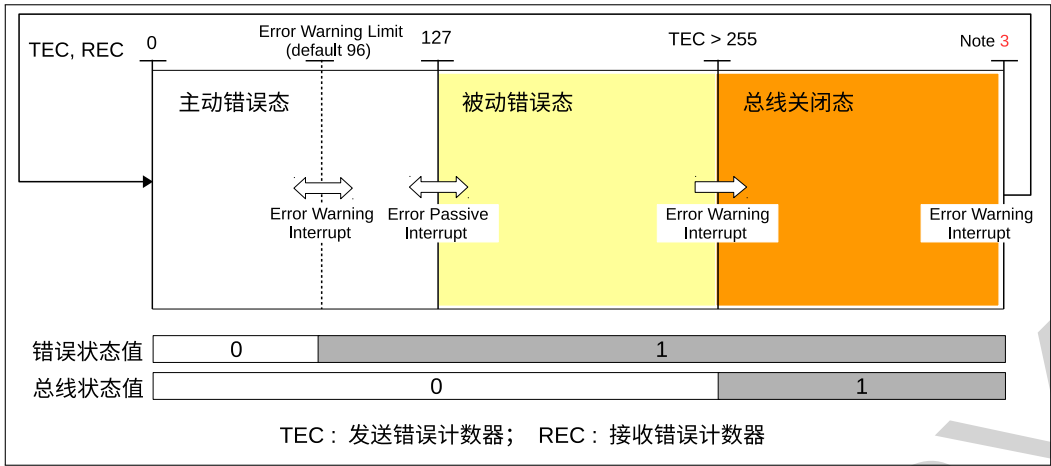


图 25-10. 错误状态变化

小于 EWL 数值时，[TWAI_ERR_ST](#) 位将立即复位为 0。只要 [TWAI_ERR_ST](#)（或 [TWAI_BUS_OFF_ST](#)）位值发生变化，便会触发错误报警中断。

25.4.7.2 被动错误

当 TEC 或 REC 数值大于 127 时，TWAI 控制器处于被动错误状态。同理，当 TEC 和 REC 数值都小于等于 127 时，TWAI 控制器进入主动错误状态。每当 TWAI 控制器从主动错误状态变为被动错误状态，或反之，都将触发被动错误中断。

25.4.7.3 离线状态与离线恢复

当 TEC 数值大于 255 时，TWAI 控制器将进入离线状态。进入离线状态后，TWAI 控制器将自动进行以下动作：

- REC 数值置为 0
- TEC 数值置为 127
- [TWAI_BUS_OFF_ST](#) 位置 1
- 进入复位模式

每当 [TWAI_BUS_OFF_ST](#) 位（或 [TWAI_ERR_ST](#) 位）数值发生变化时，都将触发错误报警中断。

为了返回主动错误状态，TWAI 控制器必须进行离线恢复。要启动离线恢复，首先需要退出复位模式，进入操作模式。然后要求 TWAI 控制器在总线上检测到 128 次 11 个连续隐性位。

每一次 TWAI 控制器检测到 11 个连续隐性位时，TEC 数值都将减小，以追踪离线恢复进程。当离线恢复完成后（TEC 数值从 127 减小到 0），[TWAI_BUS_OFF_ST](#) 位将自动复位为 0，从而触发错误报警中断。

25.4.8 错误捕捉

错误捕捉 (ECC) 功能允许 TWAI 控制器以错误代码的形式记录 TWAI 总线错误的错误类型和 bit 位置。当检测到一个 TWAI 总线错误时，总线错误中断将被触发，相应的错误代码将记录在 [TWAI_ERR_CODE_CAP_REG](#) 中。寄存器 [TWAI_ERR_CODE_CAP_REG](#) 中存储的当前错误代码被读取之前，后续的总线错误中断触发时，将不会再记录错误代码。

下表 25-16 所示为寄存器 [TWAI_ERR_CODE_CAP_REG](#) 中的域：

表 25-16. TWAI_ERR_CODE_CAP_REG 中的位信息 (0x30)

Bit 31-8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	ERRC.1	ERRC.0	DIR	SEG.4	SEG.3	SEG.2	SEG.1	SEG.0

说明:

- 错误代码 (ERRC): 表示总线错误的类型。00 代表位错误, 01 代表格式错误, 10 代表填充错误, 11 代表其他错误类型。
- 传输方向 (DIR): 表示总线错误发生时, TWAI 控制器处于发送器状态还是接收器状态。0 代表发送器, 1 代表接收器。
- 错误段 (SEG): 表示总线错误发生在 TWAI 报文的哪个段。

下表 25-17 所示为 SEG.0 ~ SEG.4 的位信息。

表 25-17. SEG.4 - SEG.0 的位信息

Bit SEG.4	Bit SEG.3	Bit SEG.2	Bit SEG.1	Bit SEG.0	描述
0	0	0	1	1	帧起始
0	0	0	1	0	ID.28 ~ ID.21
0	0	1	1	0	ID.20 ~ ID.18
0	0	1	0	0	bit SRTR
0	0	1	0	1	bit IDE
0	0	1	1	1	ID.17 ~ ID.13
0	1	1	1	1	ID.12 ~ ID.5
0	1	1	1	0	ID.4 ~ ID.0
0	1	1	0	0	bit RTR
0	1	1	0	1	保留位 1
0	1	0	0	1	保留位 0
0	1	0	1	1	数据长度代码
0	1	0	1	0	数据域
0	1	0	0	0	CRC 序列
1	1	0	0	0	CRC 分界符
1	1	0	0	1	确认槽
1	1	0	1	1	确认分界符
1	1	0	1	0	帧结束
1	0	0	1	0	间歇域
1	0	0	0	1	主动错误标志
1	0	1	1	0	被动错误标志
1	0	0	1	1	兼容显性位
1	0	1	1	1	错误分界符
1	1	1	0	0	过载标志

说明:

- Bit SRTR: 标准格式 RTR bit。

- Bit IDE: 标识符扩展位。0 表示标准格式。

25.4.9 仲裁丢失捕捉

仲裁丢失捕捉 (ALC) 功能允许 TWAI 控制器记录丢失仲裁的 bit 位置。当 TWAI 控制器丢失仲裁时, bit 位置将被记录在寄存器 `TWAI_ARB_LOST_CAP_REG` 中, 同时触发仲裁丢失中断。

后续的仲裁丢失中断触发时, bit 位置将不会被记录在 `TWAI_ARB_LOST_CAP_REG` 中, 直到 `TWAI_ERR_CODE_CAP_REG` 中的当前仲裁丢失捕捉被读取。

下表 25-18 所示为 `TWAI_ERR_CODE_CAP_REG` 中的位域; 下图 25-11 所示为一条 TWAI 报文的 bit 位置。

表 25-18. `TWAI_ARB_LOST_CAP_REG` 中的位信息 (0x2c)

Bit 31-5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
保留	BITNO.4	BITNO.3	BITNO.2	BITNO.1	BITNO.0

说明:

- 位号 (BITNO): 表示丢失仲裁的 TWAI 报文的第 n 个位。

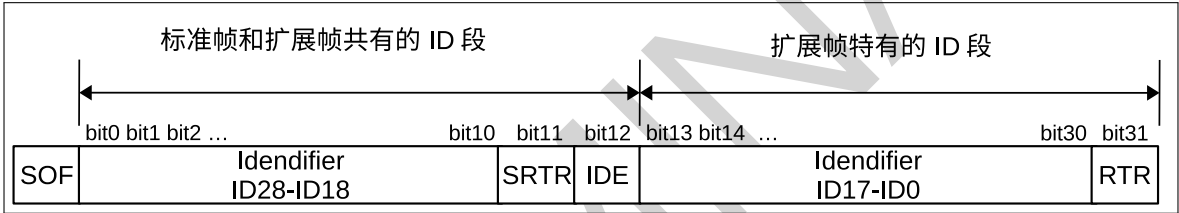


图 25-11. 丢失仲裁的 bit 位置

25.5 寄存器列表

请注意，“访问权限”一栏中，“I”用于区分第 25.4.1 中描述的工作模式，其中左侧为操作模式下的访问权限，右侧标红字体为复位模式下的访问权限。本小节的所有地址均为相对于双线汽车接口基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
TWAI_MODE_REG	模式寄存器	0x0000	R/W
TWAI_BUS_TIMING_0_REG	时序配置寄存器 0	0x0018	RO R/W
TWAI_BUS_TIMING_1_REG	时序配置寄存器 1	0x001C	RO R/W
TWAI_ERR_WARNING_LIMIT_REG	错误寄存器	0x0034	RO R/W
TWAI_DATA_0_REG	数据寄存器 0	0x0040	WO R/W
TWAI_DATA_1_REG	数据寄存器 1	0x0044	WO R/W
TWAI_DATA_2_REG	数据寄存器 2	0x0048	WO R/W
TWAI_DATA_3_REG	数据寄存器 3	0x004C	WO R/W
TWAI_DATA_4_REG	数据寄存器 4	0x0050	WO R/W
TWAI_DATA_5_REG	数据寄存器 5	0x0054	WO R/W
TWAI_DATA_6_REG	数据寄存器 6	0x0058	WO R/W
TWAI_DATA_7_REG	数据寄存器 7	0x005C	WO R/W
TWAI_DATA_8_REG	数据寄存器 8	0x0060	WO RO
TWAI_DATA_9_REG	数据寄存器 9	0x0064	WO RO
TWAI_DATA_10_REG	数据寄存器 10	0x0068	WO RO
TWAI_DATA_11_REG	数据寄存器 11	0x006C	WO RO
TWAI_DATA_12_REG	数据寄存器 12	0x0070	WO RO
TWAI_CLOCK_DIVIDER_REG	时钟分频寄存器	0x007C	不定
控制寄存器			
TWAI_CMD_REG	指令寄存器	0x0004	WO
状态寄存器			
TWAI_STATUS_REG	状态寄存器	0x0008	RO
TWAI_ARB_LOST_CAP_REG	仲裁丢失寄存器	0x002C	RO
TWAI_ERR_CODE_CAP_REG	错误捕获寄存器	0x0030	RO
TWAI_RX_ERR_CNT_REG	接收错误寄存器	0x0038	RO R/W
TWAI_TX_ERR_CNT_REG	发送错误寄存器	0x003C	RO R/W
TWAI_RX_MESSAGE_CNT_REG	接收数据寄存器	0x0074	RO
中断寄存器			
TWAI_INT_RAW_REG	中断寄存器	0x000C	RO
TWAI_INT_ENA_REG	中断使能寄存器	0x0010	R/W

25.6 寄存器

请注意“访问权限”一栏中，“I”左侧为操作模式下的访问权限，右侧标红字体为复位模式下的访问权限。本小节的所有地址均为相对于双线汽车接口基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 25.1. TWAI_MODE_REG (0x0000)

(reserved)																												TWAI_RX_FILTER_MODE TWAI_SELF_TEST_MODE TWAI_LISTEN_ONLY_MODE TWAI_RESET_MODE				
31																												4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Reset	

TWAI_RESET_MODE 配置 TWAI 控制器操作模式。1：复位模式；0：操作模式。(R/W)

TWAI_LISTEN_ONLY_MODE 置 1 进入只听模式，处于该模式下的节点只接收总线上数据，不产生应答信号，也不更新接收错误计数。(R/W)

TWAI_SELF_TEST_MODE 置 1 启动自测模式，此模式下发送节点发送完数据后无需应答信号反馈。该模式常配合自接自收指令测试某个节点。(R/W)

TWAI_RX_FILTER_MODE 配置滤波模式。0：双滤波模式；1：单滤波模式。(R/W)

Register 25.2. TWAI_BUS_TIMING_0_REG (0x0018)

(reserved)																TWAI_SYNC_JUMP_WIDTH (reserved)								TWAI_BAUD_PRESC							
31																16	15	14	13	12							0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0	0x0	0x00						Reset							

TWAI_BAUD_PRESC 预分频值，决定分频比例。(RO | R/W)

TWAI_SYNC_JUMP_WIDTH 同步跳宽 (SJW)，范围为 1 ~ 4 个时间定额。(RO | R/W)

Register 25.3. TWAI_BUS_TIMING_1_REG (0x001C)

(reserved)																								TWAI_TIME_SAMP		TWAI_TIME_SEG2		TWAI_TIME_SEG1																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
31																								8	7	6	4	3	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

TWAI_TIME_SEG1 缓冲时期段 1 的宽度。(RO | R/W)

TWAI_TIME_SEG2 缓冲时期段 2 的宽度。(RO | R/W)

TWAI_TIME_SAMP 采样点数目。0: 采样 1 次; 1: 采样三次。(RO | R/W)

Register 25.4. TWAI_ERR_WARNING_LIMIT_REG (0x0034)

(reserved)																				TWAI_ERR_WARNING_LIMIT													
31																				8	7	0											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x60												Reset	

TWAI_ERR_WARNING_LIMIT 错误报警阈值，当任一错误计数数值超过该阈值或者所有错误计数数值都小于该阈值时，将触发错误报警中断（使能信号有效情况下）。(RO | R/W)

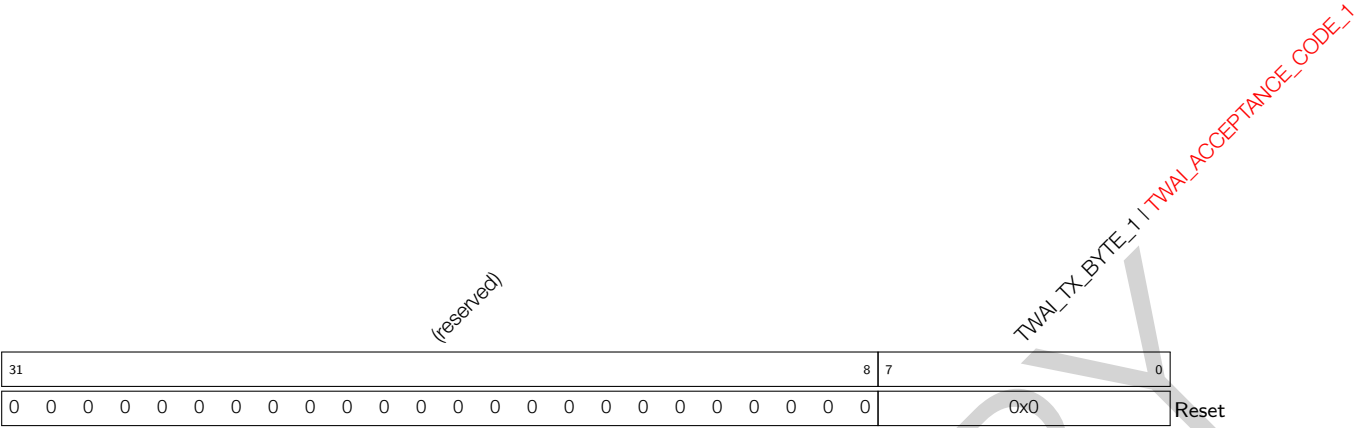
Register 25.5. TWAI_DATA_0_REG (0x0040)

(reserved)																								TWAI_TX_BYTE_0 TWAI_ACCEPTANCE_CODE_0								
31																								0								
0 0																								0x0								Reset

TWAI_TX_BYTE_0 操作模式下，存储着待发送数据的第 0 个字节内容。(WO)

TWAI_ACCEPTANCE_CODE_0 复位模式下，存储着滤波编码的第 0 个字节。(R/W)

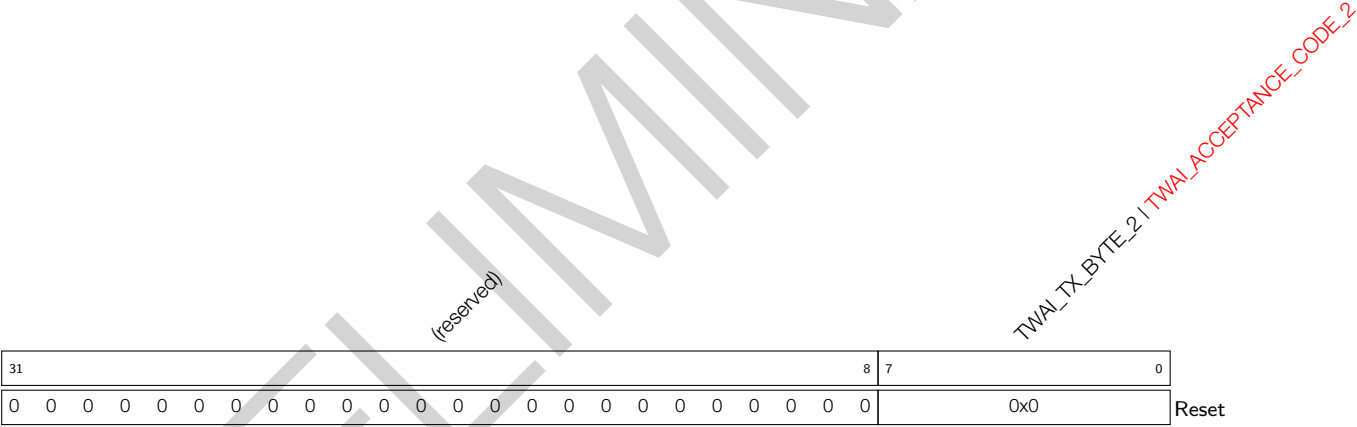
Register 25.6. TWAI_DATA_1_REG (0x0044)



TWAI_TX_BYTE_1 操作模式下，存储着待发送数据的第 1 个字节内容。(WO)

TWAI_ACCEPTANCE_CODE_1 复位模式下，存储着滤波编码的第 1 个字节。(R/W)

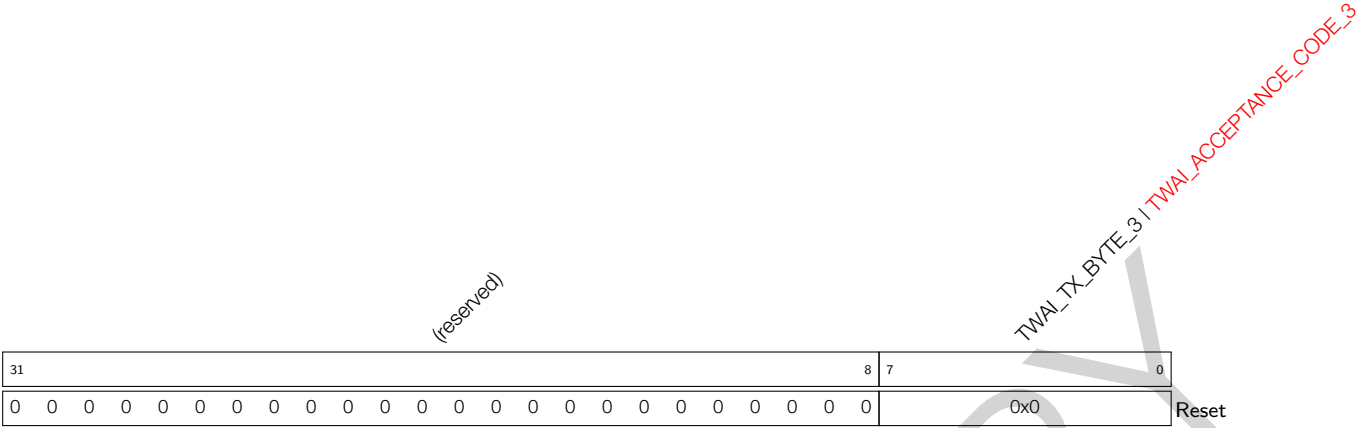
Register 25.7. TWAI_DATA_2_REG (0x0048)



TWAI_TX_BYTE_2 操作模式下，存储着待发送数据的第 2 个字节内容。(WO)

TWAI_ACCEPTANCE_CODE_2 复位模式下，存储着滤波编码的第 2 个字节。(R/W)

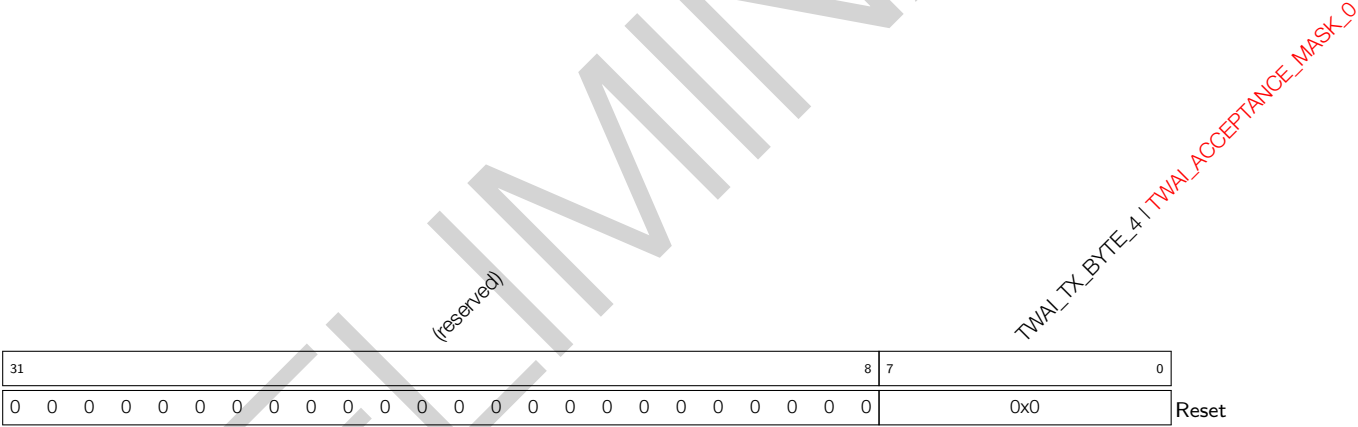
Register 25.8. TWAI_DATA_3_REG (0x004C)



TWAI_TX_BYTE_3 操作模式下，存储着待发送数据的第 3 个字节内容。(WO)

TWAI_ACCEPTANCE_CODE_3 复位模式下，存储着滤波编码的第 3 个字节。(R/W)

Register 25.9. TWAI_DATA_4_REG (0x0050)



TWAI_TX_BYTE_4 操作模式下，存储着待发送数据的第 4 个字节内容。(WO)

TWAI_ACCEPTANCE_MASK_0 复位模式下，存储着滤波编码的第 0 个字节。(R/W)

Register 25.12. TWAI_DATA_7_REG (0x005C)

(reserved)																								TWAI_TX_B[7:0]							
31																								0							
0 0																								0x0							
																								Reset							

TWAI_TX_BYTE_7 操作模式下，存储着待发送数据的第 7 个字节内容。(WO)

TWAI_ACCEPTANCE_MASK_3 复位模式下，存储着滤波编码的第 3 个字节。(R/W)

Register 25.13. TWAI_DATA_8_REG (0x0060)

[illegible]

TWAI_TX_BYTE_8 操作模式下，存储着待发送数据的第 8 个字节内容。(WO)

Register 25.14. TWAI_DATA_9_REG (0x0064)

(reserved)																												TWAI_TX_BYTE_9															
31																												7								0							
0 0																												0x0								Reset							

TWAI_TX_BYTE_9 操作模式下，存储着待发送数据的第 9 个字节内容。(WO)

Register 25.15. TWAI_DATA_10_REG (0x0068)

(reserved)																TWAI_TX_BYTE_10									
31																8	7	0							
0 0																0x0								Reset	

TWAI_TX_BYTE_10 操作模式下，存储着待发送数据的第 10 个字节内容。(WO)

Register 25.16. TWAI_DATA_11_REG (0x006C)

(reserved)																								TWAI_TX_BYTE_11																									
31																								8	7	0																							
0 0																								0x0																Reset									

TWAI_TX_BYTE_11 操作模式下，存储着待发送数据的第 11 个字节内容。(WO)

Register 25.17. TWAI_DATA_12_REG (0x0070)

(reserved)																								TWAI_TX_BYTE_12															
31																								7								0							
0 0																								0x0								Reset							

TWAI_TX_BYTE_12 操作模式下，存储着待发送数据的第 12 个字节内容。(WO)

Register 25.18. TWAI_CLOCK_DIVIDER_REG (0x007C)

(reserved)																TWAI_CLOCK_OFF				TWAI_CD			
31																9	8	7	0				
0 0																0	0x0				Reset		

TWAI_CD 配置输出时钟 CLKOUT 的分频系数。(R/W)

TWAI_CLOCK_OFF 复位模式下可配。1: 关闭输出的 CLKOUT 时钟；0: 打开 CLKOUT 时钟。(RO
I R/W)

Register 25.19. TWAI_CMD_REG (0x0004)

(reserved)																												TWAI_SELF_RX_REQ TWAI_CLR_OVERRUN TWAI_RELEASE_BUF TWAI_ABORT_TX TWAI_TX_REQ						
31																												5	4	3	2	1	0	Reset
0 0																												0	0	0	0	0		

- TWAI_TX_REQ** 置 1 驱动节点开始发送数据任务。(WO)
- TWAI_ABORT_TX** 置 1 取消当前还未开始的发送任务。(WO)
- TWAI_RELEASE_BUF** 置 1 释放接收缓冲器。(WO)
- TWAI_CLR_OVERRUN** 置 1 清除数据溢出状态。(WO)
- TWAI_SELF_RX_REQ** 自接自收命令。置 1 允许发送节点发送数据的同时接收总线上的数据。(WO)

Register 25.20. TWAI_STATUS_REG (0x0008)

(reserved)																				TWAI_MISS_ST TWAI_BUS_OFF_ST TWAI_ERR_ST TWAI_TX_ST TWAI_RX_ST TWAI_TX_COMPLETE TWAI_TX_BUF_ST TWAI_OVERRUN_ST TWAI_RX_BUF_ST										
31																				9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	Reset	

- TWAI_RX_BUF_ST** 若值为 1,表明接收缓冲器中数据不为空,至少有一个已经接收到的数据包。(RO)
- TWAI_OVERRUN_ST** 若值为 1, 表明接收 FIFO 中存储的数据已满, 产生了溢出。(RO)
- TWAI_TX_BUF_ST** 若值为 1, 表明发送缓冲器为空, 允许写入待发送数据。(RO)
- TWAI_TX_COMPLETE** 若值为 1, 表明成功从总线上接收到一个数据包。(RO)
- TWAI_RX_ST** 若值为 1, 表明节点正在从总线上接收数据。(RO)
- TWAI_TX_ST** 若值为 1, 表明节点正在往总线上发送数据。(RO)
- TWAI_ERR_ST** 若值为 1, 表明接收错误计数和发送错误计数中至少有一个数值大于等于寄存器 [TWAI_ERR_WARNING_LIMIT_REG](#) 中配置的数值。(RO)
- TWAI_BUS_OFF_ST** 若值为 1, 表明节点处于离线状态, 不再响应总线上的数据传输。(RO)
- TWAI_MISS_ST** 反映了从接收 FIFO 中取出数据包的完整状态。1: 当前数据包是缺失的; 0: 当前数据包是完整的。(RO)

Register 25.21. TWAI_ARB_LOST_CAP_REG (0x002C)

(reserved)																																TWAI_ARB_LOST_CAP																
31																															5	4	0															
0 0																															0x0																Reset	

TWAI_ARB_LOST_CAP 记录着发送节点仲裁丢失的 bit 位置。(RO)

Register 25.22. TWAI_ERR_CODE_CAP_REG (0x0030)

(reserved)																TWAI_ERR_CODE_CAP_REG																																										
31																8	7	6	5	4																																						
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0x0																																										
Reset																																																										

TWAI_ECC_SEGMENT 记录错误发生的位置，详见表 25-16。(RO)

TWAI_ECC_DIRECTION 记录错误时节点的数据传输方向。1：接收数据时发生错误；0：发送数据时发生错误。(RO)

TWAI_ECC_TYPE 记录错误类别：00：位错误；01：格式错误；10：填充错误；11：其他错误。(RO)

Register 25.23. TWAI_RX_ERR_CNT_REG (0x0038)

(reserved)																TWAI_RX_ERR_CNT																															
31																7																0															
0 0																0x0																Reset															

TWAI_RX_ERR_CNT 接收错误计数，数值变化发生在接收状态下。(RO | R/W)

Register 25.24. TWAI_TX_ERR_CNT_REG (0x003C)

(reserved)																								TWAI_TX_ERR_CNT									
31																								8	7	0							
0 0																								0x0								Reset	

TWAI_TX_ERR_CNT 发送错误计数，数值变化发生在发送状态下。(RO | R/W)

Register 25.25. TWAI_RX_MESSAGE_CNT_REG (0x0074)

(reserved)																								TWAI_RX_MESSAGE_COUNTER																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																								
31																								7	6	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
0 0																								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TWAI_RX_MESSAGE_COUNTER 存储着接收 FIFO 中数据包的个数。(RO)

Register 25.27. TWAI_INT ENA_REG (0x0010)

(reserved)																TWAI_BUS_STATE_INT_ENA			
																TWAI_BUS_ERR_INT_ENA			
																TWAI_ARB_LOST_INT_ENA			
																TWAI_ERR_PASSIVE_INT_ENA			
																(reserved)			
																TWAI_OVERRUN_INT_ENA			
																TWAI_ERR_WARN_INT_ENA			
																TWAI_TX_INT_ENA			
																TWAI_RX_INT_ENA			
31									9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

- TWAI_RX_INT_ENA 置 1 使能接收中断。(R/W)
- TWAI_TX_INT_ENA 置 1 使能发送中断。(R/W)
- TWAI_ERR_WARN_INT_ENA 置 1 使能错误报警中断。(R/W)
- TWAI_OVERRUN_INT_ENA 置 1 使能数据溢出中断。(R/W)
- TWAI_ERR_PASSIVE_INT_ENA 置 1 使能被动错误中断。(R/W)
- TWAI_ARB_LOST_INT_ENA 置 1 使能仲裁丢失中断。(R/W)
- TWAI_BUS_ERR_INT_ENA 置 1 使能总线错误中断。(R/W)
- TWAI_BUS_STATE_INT_ENA 置 1 使能总线状态中断。(R/W)

26 LED PWM 控制器 (LEDC)

26.1 概述

LED PWM 控制器用于生成控制 LED 的脉冲宽度调制信号 (PWM)，具有占空比自动渐变等专门功能。该外设也可生成 PWM 信号用作其他用途。

26.2 特性

LED PWM 控制器具有如下特性：

- 六个独立的 PWM 生成器（即六个通道）
- 四个独立定时器，可实现小数分频
- 占空比自动渐变（即 PWM 信号占空比可逐渐增加或减小，无须处理器干预），渐变完成时产生中断
- PWM 输出信号相位可调节
- 低功耗模式 (Light-sleep mode) 下可输出 PWM 信号
- PWM 最大精度为 14 位

四个定时器具有相同的功能和运行方式，下文将四个定时器统称为定时器 x （ x 的范围是 0 到 3）。六个 PWM 生成器的功能和运行方式也相同，下文将统称为 PWM n （ n 的范围是 0 到 5）。

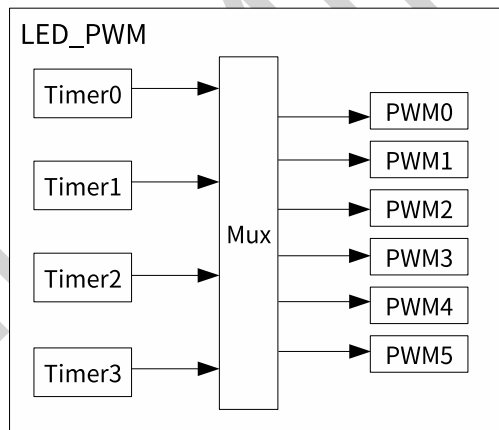


图 26-1. LED PWM 控制器架构

26.3 功能描述

26.3.1 架构

图 26-1 为 LED PWM 控制器的架构。

四个定时器可独立配置（可配置时钟分频器和计数器最大值），每个定时器内部有一个时基计数器（即基于基准时钟周期计数的计数器）。每个 PWM 生成器在四个定时器中择一，以该定时器的计数值为基准生成 PWM 信号。

图 26-2 为定时器和 PWM 生成器的主要功能块。

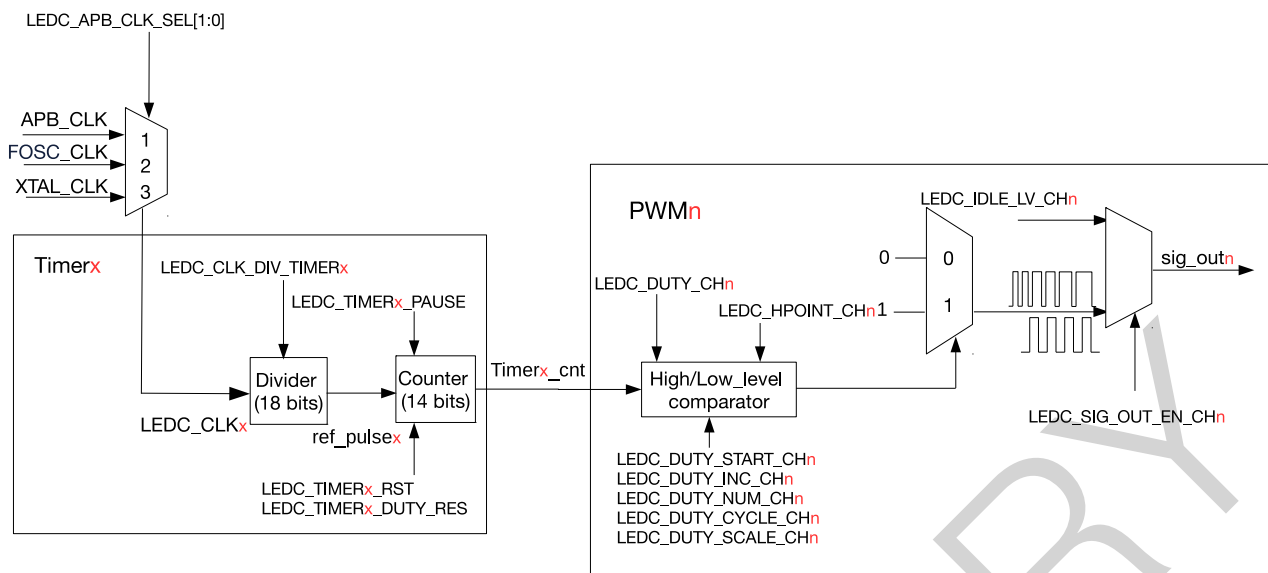


图 26-2. 定时器和 PWM 生成器功能块

26.3.2 定时器

LED PWM 控制器的每个定时器内部都有一个时基计数器。图 26-2 中时基计数器使用的时钟信号称为 ref_pulse_x。所有定时器使用同一个时钟源信号 LEDC_CLK_x，该时钟源信号经分频器分频后产生 ref_pulse_x 供计数器使用。

26.3.2.1 时钟源

软件配置 LED PWM 寄存器由 APB_CLK 驱动。更多关于 APB_CLK 的信息，详见章节 6 复位和时钟。要使用 LED PWM 控制器，需使能 LED PWM 的 APB_CLK 时钟信号，该时钟信号可通过置位 SYSTEM_PERIP_CLK_EN0_REG 寄存器的 SYSTEM_LEDC_CLK_EN 使能，通过软件置位 SYSTEM_PERIP_RST_EN0_REG 寄存器的 SYSTEM_LEDC_RST 位复位。更多信息，请参阅章节 13 系统寄存器 (SYSREG) 的表 13-1。

LED PWM 控制器的定时器有三个时钟源信号可以选择：APB_CLK、FOSC_CLK 和 XTAL_CLK（更多关于时钟源的信息详见章节 6 复位和时钟）。为 LEDC_CLK_x 选择时钟源信号的配置如下：

- APB_CLK：将 LEDC_APB_CLK_SEL[1:0] 置 1
- FOSC_CLK：将 LEDC_APB_CLK_SEL[1:0] 置 2
- XTAL_CLK：将 LEDC_APB_CLK_SEL[1:0] 置 3

之后，LEDC_CLK_x 信号会进入时钟分频器。

26.3.2.2 时钟分频器配置

LEDC_CLK_x 信号传输到时钟分频器，产生 ref_pulse_x 信号供计数器使用。ref_pulse_x 的频率等于 LEDC_CLK_x 的频率经 LEDC_CLK_DIV_TIMER_x 分频系数分频后的结果（见图 26-2）。

LEDC_CLK_DIV_TIMER_x 分频系数为小数分频，因此其值可为非整数。分频系数 LEDC_CLK_DIV_TIMER_x 可根据下列等式通过 LEDC_CLK_DIV_TIMER_x 字段配置：

$$\text{LEDC_CLK_DIV_TIMER}_x = A + \frac{B}{256}$$

- 整数部分 A 为 LEDC_CLK_DIV_TIMER_x 字段的高 10 位（即 LEDC_TIMER_x_CONF_REG[21:12]）

- 小数部分 B 为 `LEDC_CLK_DIV_TIMERx` 字段的低 8 位 (即 `LEDC_TIMERx_CONF_REG[11:4]`)

小数部分 B 为 0 时, `LEDC_CLK_DIV_TIMERx` 的值为整数 (整数分频)。也就是说, 每 A 个 `LEDC_CLKx` 时钟周期产生一个 `ref_pulsex` 时钟脉冲。

小数部分 B 不为 0 时, `LEDC_CLK_DIV_TIMERx` 的值非整数。时钟分频器按照 A 个 `LEDC_CLKx` 时钟周期和 $(A+1)$ 个 `LEDC_CLKx` 时钟周期轮流进行非整数分频。这样一来, `ref_pulsex` 时钟脉冲的平均频率便会是理想值 (非整数分频的频率)。每 256 个 `ref_pulsex` 时钟脉冲中:

- 有 B 个以 $(A+1)$ 个 `LEDC_CLKx` 时钟周期分频
- 有 $(256-B)$ 个以 A 个 `LEDC_CLKx` 时钟周期分频
- 以 $(A+1)$ 个 `LEDC_CLKx` 时钟周期分频的时钟脉冲均匀分布在以 A 分频的时钟脉冲中

图 26-3 展示了 `LEDC_CLK_DIV_TIMERx` 分频系数非整数时, `LEDC_CLKx` 时钟周期和 `ref_pulsex` 时钟脉冲的关系。

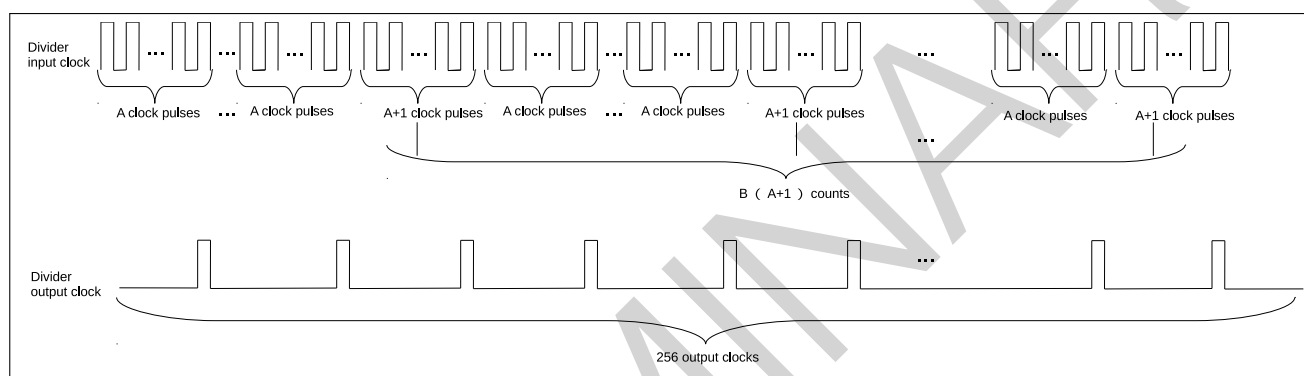


图 26-3. `LEDC_CLK_DIV_TIMERx` 非整数时的分频

在运行时改变定时器时钟的分频系数, 需先置位 `LEDC_CLK_DIV_TIMERx` 字段, 然后置位 `LEDC_TIMERx_PARA_UP` 字段应用新配置。新配置会在计数器下次溢出时生效。`LEDC_TIMERx_PARA_UP` 字段由硬件自动清除。

26.3.2.3 14 位计数器

每个定时器有一个以 `ref_pulsex` 为基准时钟的 14 位时基计数器 (见图 26-2)。`LEDC_TIMERx_DUTY_RES` 字段用于配置 14 位计数器的最大值。因此, PWM 信号的最大精确度为 14 位。计数器最大可计数至 $2^{\text{LEDC_TIMERx_DUTY_RES}-1}$, 然后溢出重新从 0 开始计数。软件可以读取、复位、暂停计数器。

计数器可在每次溢出时触发 (`LEDC_TIMERx_OVF_INT`) 中断, 这个中断为硬件自动产生, 不需要配置。计数器也可配置为在溢出 `LEDC_OVF_NUM_CHn + 1` 次时触发 `LEDC_OVF_CNT_CHn_INT` 中断, 该中断配置步骤如下:

1. 配置 `LEDC_TIMER_SEL_CHn` 为 PWM 生成器选择该计数器
2. 置位 `LEDC_OVF_CNT_EN_CHn` 使能计数器
3. 把 `LEDC_OVF_NUM_CHn` 的值设为计数器触发中断的溢出次数减 1
4. 置位 `LEDC_OVF_CNT_CHn_INT_ENA` 使能溢出中断
5. 置位 `LEDC_TIMERx_DUTY_RES` 使能定时器, 等待 `LEDC_OVF_CNT_CHn_INT` 中断产生

如图 26-2 所示, PWM 生成器输出信号 `sig_outn` 的频率取决于定时器的时钟源 `LEDC_CLKx`、时钟分频系数 `LEDC_CLK_DIV_TIMERx` 以及计数器的计数范围 `LEDC_TIMERx_DUTY_RES`:

$$f_{\text{PWM}} = \frac{f_{\text{LEDC_CLK}_x}}{\text{LEDC_CLK_DIV}_x \cdot 2^{\text{LEDC_TIMER}_x\text{_DUTY_RES}}}$$

在运行时改变计数器的最大值，需先置位 `LEDC_TIMERx_DUTY_RES` 字段，然后置位 `LEDC_TIMERx_PARA_UP` 字段。新的配置在计数器下一次溢出时生效。如果重新配置 `LEDC_OVF_CNT_EN_CHn` 字段，也需置位 `LEDC_PARA_UP_CHn` 应用新配置。总之，更改配置时都需置位 `LEDC_PARA_UP_CHn` 应用新配置。`LEDC_TIMERx_PARA_UP` 字段由硬件自动清除。

26.3.3 PWM 生成器

要生成 PWM 信号，PWM 生成器 (PWM_n) 需选择一个定时器 (Timer_x)。每个 PWM 生成器均可通过置位 `LEDC_TIMER_SEL_CHn` 单独配置，在四个定时器中选择一个输出 PWM 信号。

如图 26-2 所示，每个 PWM 生成器主要包括一个高低电平比较器和两个选择器。PWM 生成器将定时器的 14 位计数值 (Timer_x_cnt) 与高低电平比较器的值 Hpoint_n 和 Lpoint_n 比较。如果定时器的计数值等于 Hpoint_n 或 Lpoint_n，PWM 信号可以输出高低电平：

- 如果 Timer_x_cnt == Hpoint_n，则 sig_out_n 为 1。
- 如果 Timer_x_cnt == Lpoint_n，则 sig_out_n 为 0。

图 26-4 展示了如何使用 Hpoint_n 和 Lpoint_n 生成占空比固定的 PWM 信号。

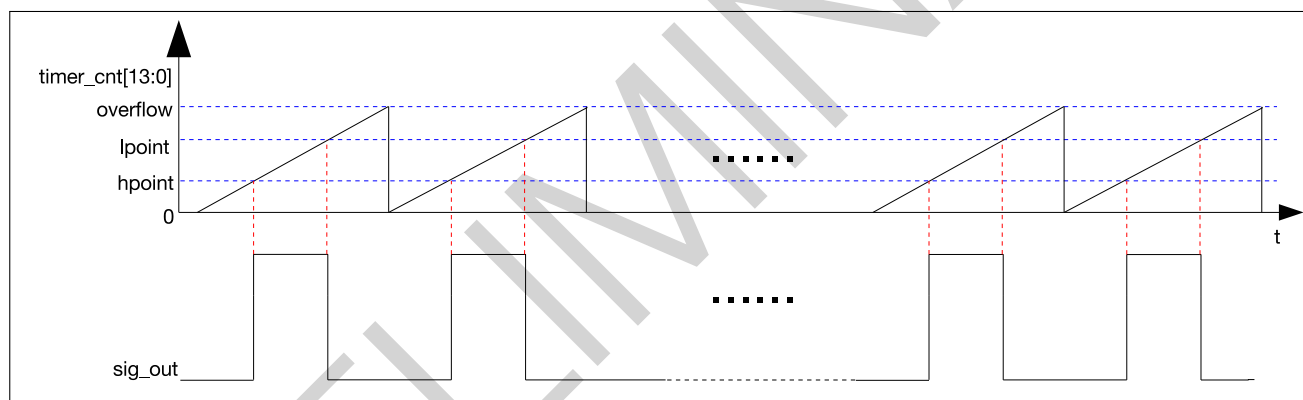


图 26-4. LED PWM 输出信号图

每当所选定时器的计数器溢出时，PWM 生成器 (PWM_n) 的 Hpoint_n 值更新为 `LEDC_HPOINT_CHn`。Lpoint_n 的值同样在计数器每次溢出时更新，为 `LEDC_DUTY_CHn[18:4]` 和 `LEDC_HPOINT_CHn` 的和。通过配置 `LEDC_DUTY_CHn[18:4]` 和 `LEDC_HPOINT_CHn` 两个字段，可设置 PWM 输出的相对相位和占空比。

置位 `LEDC_SIG_OUT_EN_CHn`，开启 PWM 信号 (sig_out_n) 输出；清除 `LEDC_SIG_OUT_EN_CHn`，关闭 PWM 信号输出，输出信号 sig_out_n 输出恒定电平，电平值为 `LEDC_IDLE_LV_CHn`。

`LEDC_DUTY_CHn[3:0]` 通过周期性改变 PWM 输出信号 sig_out_n 的占空比实现微调。如 `LEDC_DUTY_CHn[3:0]` 不为 0，那么 sig_out_n 每 16 个周期中，有 `LEDC_DUTY_CHn[3:0]` 个周期的 PWM 脉冲占空比要比 (16 - `LEDC_DUTY_CHn[3:0]`) 个周期的脉冲占空比多一个定时器的计数周期。比如，如果 `LEDC_DUTY_CHn[18:4]` 设为 10，`LEDC_DUTY_CHn[3:0]` 设为 5，则 16 个周期中，有 5 个周期的 PWM 脉冲占空比为 11，剩余 11 个周期的 PWM 脉冲占空比为 10。16 个周期的平均占空比为 10.3125。

如果重新配置 `LEDC_TIMER_SEL_CHn`、`LEDC_HPOINT_CHn`、`LEDC_DUTY_CHn[18:4]` 和 `LEDC_SIG_OUT_EN_CHn` 字段，需置位 `LEDC_PARA_UP_CHn` 应用新配置。新配置在计数器下次溢出时生效。`LEDC_TIMERx_PARA_UP` 字段由硬件自动清除。

26.3.4 占空比渐变

PWM 生成器可以渐变 PWM 输出信号的占空比，即由一种占空比逐渐变为为另一种占空比。如果开启占空比渐变功能，Lpoint n 的值会在计数器溢出固定次数后递增或递减。图 26-5 展示了占空比渐变功能。

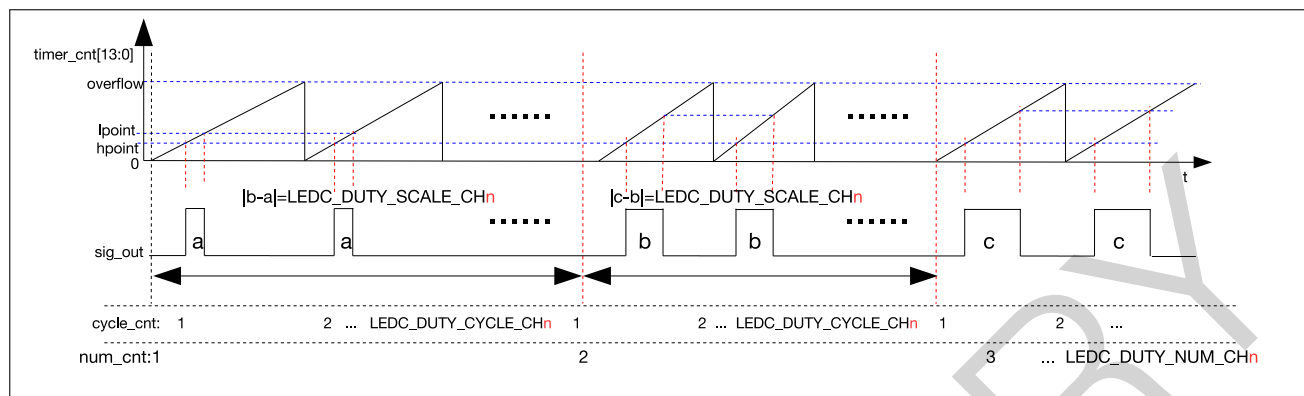


图 26-5. 输出信号占空比渐变图

占空比渐变功能可通过以下寄存器字段配置：

- **LEDC_DUTY_CH n** 用于设置 Lpoint n 的初始值。
- **LEDC_DUTY_START_CH n** 置 1 或清零，使能或关闭占空比渐变功能。
- **LEDC_DUTY_CYCLE_CH n** 用于设置 Lpoint n 在计数器溢出多少次时递增或递减。也就是说，Lpoint n 会在计数器溢出 **LEDC_DUTY_CYCLE_CH n** 次时递增或递减。
- **LEDC_DUTY_INC_CH n** 置 1 或清零，Lpoint n 递增或递减。
- **LEDC_DUTY_SCALE_CH n** 用于设置 Lpoint n 递增或递减的值。
- **LEDC_DUTY_NUM_CH n** 用于设置占空比渐变停止前，Lpoint n 递增或递减的最大次数。

如果重新配置 **LEDC_DUTY_CH n** 、**LEDC_DUTY_START_CH n** 、**LEDC_DUTY_CYCLE_CH n** 、**LEDC_DUTY_INC_CH n** 、**LEDC_DUTY_SCALE_CH n** 和 **LEDC_DUTY_NUM_CH n** 字段，需置位 **LEDC_PARA_UP_CH n** 应用新配置。**LEDC_PARA_UP_CH n** 置位后，新配置立即生效。**LEDC_TIMER x _PARA_UP** 字段由硬件自动清除。

26.3.5 中断

- **LEDC_OVF_CNT_CH n _INT**: 定时器计数器溢出 (**LEDC_OVF_NUM_CH n** + 1) 次且寄存器 **LEDC_OVF_CNT_EN_CH n** 置 1 时触发中断。
- **LEDC_DUTY_CHNG_END_CH n _INT**: PWM 生成器渐变完成后触发中断。
- **LEDC_TIMER x _OVF_INT**: 定时器达到最大计数值时触发中断。

26.4 寄存器列表

本小节的所有地址均为相对于 LED PWM 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
LEDC_CH0_CONF0_REG	通道 0 的配置寄存器 0	0x0000	varies
LEDC_CH0_CONF1_REG	通道 0 的配置寄存器 1	0x000C	varies
LEDC_CH1_CONF0_REG	通道 1 的配置寄存器 0	0x0014	varies
LEDC_CH1_CONF1_REG	通道 1 的配置寄存器 1	0x0020	varies
LEDC_CH2_CONF0_REG	通道 2 的配置寄存器 0	0x0028	varies
LEDC_CH2_CONF1_REG	通道 2 的配置寄存器 1	0x0034	varies
LEDC_CH3_CONF0_REG	通道 3 的配置寄存器 0	0x003C	varies
LEDC_CH3_CONF1_REG	通道 3 的配置寄存器 1	0x0048	varies
LEDC_CH4_CONF0_REG	通道 4 的配置寄存器 0	0x0050	varies
LEDC_CH4_CONF1_REG	通道 4 的配置寄存器 1	0x005C	varies
LEDC_CH5_CONF0_REG	通道 5 的配置寄存器 0	0x0064	varies
LEDC_CH5_CONF1_REG	通道 5 的配置寄存器 1	0x0070	varies
LEDC_CONF_REG	LEDC 全局配置寄存器	0x00D0	R/W
高位点寄存器			
LEDC_CH0_HPOINT_REG	通道 0 的高位点寄存器	0x0004	R/W
LEDC_CH1_HPOINT_REG	通道 1 的高位点寄存器	0x0018	R/W
LEDC_CH2_HPOINT_REG	通道 2 的高位点寄存器	0x002C	R/W
LEDC_CH3_HPOINT_REG	通道 3 的高位点寄存器	0x0040	R/W
LEDC_CH4_HPOINT_REG	通道 4 的高位点寄存器	0x0054	R/W
LEDC_CH5_HPOINT_REG	通道 5 的高位点寄存器	0x0068	R/W
占空比寄存器			
LEDC_CH0_DUTY_REG	通道 0 的初始占空比	0x0008	R/W
LEDC_CH0_DUTY_R_REG	通道 0 的当前占空比	0x0010	RO
LEDC_CH1_DUTY_REG	通道 1 的初始占空比	0x001C	R/W
LEDC_CH1_DUTY_R_REG	通道 1 的当前占空比	0x0024	RO
LEDC_CH2_DUTY_REG	通道 2 的初始占空比	0x0030	R/W
LEDC_CH2_DUTY_R_REG	通道 2 的当前占空比	0x0038	RO
LEDC_CH3_DUTY_REG	通道 3 的初始占空比	0x0044	R/W
LEDC_CH3_DUTY_R_REG	通道 3 的当前占空比	0x004C	RO
LEDC_CH4_DUTY_REG	通道 4 的初始占空比	0x0058	R/W
LEDC_CH4_DUTY_R_REG	通道 4 的当前占空比	0x0060	RO
LEDC_CH5_DUTY_REG	通道 5 的初始占空比	0x006C	R/W
LEDC_CH5_DUTY_R_REG	通道 5 的当前占空比	0x0074	RO
定时器寄存器			
LEDC_TIMER0_CONF_REG	定时器 0 配置	0x00A0	varies
LEDC_TIMER0_VALUE_REG	定时器 0 的当前计数器值	0x00A4	RO
LEDC_TIMER1_CONF_REG	定时器 1 配置	0x00A8	varies
LEDC_TIMER1_VALUE_REG	定时器 1 的当前计数器值	0x00AC	RO

名称	描述	地址	访问
LEDC_TIMER2_CONF_REG	定时器 2 配置	0x00B0	varies
LEDC_TIMER2_VALUE_REG	定时器 2 的当前计数器值	0x00B4	RO
LEDC_TIMER3_CONF_REG	定时器 3 配置	0x00B8	varies
LEDC_TIMER3_VALUE_REG	定时器 3 的当前计数器值	0x00BC	RO
中断寄存器			
LEDC_INT_RAW_REG	原始中断状态	0x00C0	R/WTC/SS
LEDC_INT_ST_REG	屏蔽中断状态	0x00C4	RO
LEDC_INT_ENA_REG	中断使能位	0x00C8	R/W
LEDC_INT_CLR_REG	中断清除位	0x00CC	WT
版本寄存器			
LEDC_DATE_REG	版本控制寄存器	0x00FC	R/W

26.5 寄存器

本小节的所有地址均为相对于 LED PWM 控制器基地址的地址偏移量（相对地址），具体基地址请见章节 3 系统和存储器中的表 3-4。

Register 26.1. LEDC_CH n _CONF0_REG (n : 0-5) (0x0000+20* n)

(reserved)																LEDC_OVF_CNT_RESET_CH ⁿ LEDC_OVF_CNT_EN_CH ⁿ				LEDC_OVF_NUM_CH ⁿ				LEDC_PARA_UP_CH ⁿ LEDC_IDLE_LV_CH ⁿ LEDC_SIG_OUT_EN_CH ⁿ LEDC_TIMER_SEL_CH ⁿ					
31															17	16	15	14					5	4	3	2	1	0	
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																0 0		0				0 0		0 0		0 0			
																												Reset	

LEDC_TIMER_SEL_CH n 用于选择通道 n 的定时器。

- 0: 选择定时器 0
- 1: 选择定时器 1
- 2: 选择定时器 2
- 3: 选择定时器 3 (R/W)

LEDC_SIG_OUT_EN_CH n 置位此位，使能通道 n 的信号输出。(R/W)

LEDC_IDLE_LV_CH n 控制通道 n 不工作时（LEDC_SIG_OUT_EN_CH n 为 0 时）的输出电平。(R/W)

LEDC_PARA_UP_CH n 用于更新通道 n 的下列字段，由硬件自动清除。(WT)

- LEDC_HPOINT_CH n
- LEDC_DUTY_START_CH n
- LEDC_SIG_OUT_EN_CH n
- LEDC_TIMER_SEL_CH n
- LEDC_DUTY_NUM_CH n
- LEDC_DUTY_CYCLE_CH n
- LEDC_DUTY_SCALE_CH n
- LEDC_DUTY_INC_CH n
- LEDC_OVF_CNT_EN_CH n

见下页...

Register 26.1. LEDC_CH n _CONF0_REG (n : 0-5) (0x0000+20* n)

接上页...

LEDC_OVF_NUM_CH n 用于配置定时器溢出次数的最大值减 1。通道 n 的定时器溢出次数达到 (LEDC_OVF_NUM_CH n +1) 次时，触发 LEDC_OVF_CNT_CH n _INT 中断。(R/W)

LEDC_OVF_CNT_EN_CH n 用于计算通道 n 选择的定时器溢出的次数。(R/W)

LEDC_OVF_CNT_RESET_CH n 置位此位，复位通道 n 的定时器溢出计数器。(WT)

Register 26.2. LEDC_CH n _CONF1_REG (n : 0-5) (0x000C+20* n)

LEDC_DUTY_START_CH _n LEDC_DUTY_INC_CH _n			LEDC_DUTY_NUM_CH _n																	LEDC_DUTY_CYCLE_CH _n										LEDC_DUTY_SCALE_CH _n											
31	30	29	20																	19	10										9	0									
0	1	0x0																	0x0										0x0										Reset		

LEDC_DUTY_SCALE_CH n 用于配置渐变时占空比的步长变化。(R/W)

LEDC_DUTY_CYCLE_CH n 通道 n 占空比每隔 LEDC_DUTY_CYCLE_CH n 周期变化一次。(R/W)

LEDC_DUTY_NUM_CH n 用于控制占空比变化的次数。(R/W)

LEDC_DUTY_INC_CH n 决定了通道 n 输出信号的占空比是递增还是递减。1：递增；0：递减。(R/W)

LEDC_DUTY_START_CH n 此位置 1 时，LEDC_CH n _CONF1_REG 中的其他字段在定时器下次溢出时生效。(R/W/SC)

Register 26.3. LEDC_CONF_REG (0x00D0)

LEDC_CLK_EN			(reserved)																									LEDC_APB_CLK_SEL			
31	30	2																									1	0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

LEDC_APB_CLK_SEL 用于设置 4 个定时器共同的时钟源。1：APB_CLK；2：FOSC_CLK；3：XTAL_CLK。(R/W)

LEDC_CLK_EN 用于控制时钟。

1：强制开启寄存器时钟。0：仅在应用写寄存器时支持时钟。(R/W)

Register 26.4. LEDC_CH_n_HPOINT_REG (*n*: 0-5) (0x0004+20**n*)

(reserved)														LEDC_HPOINT_CH ⁿ																		
31														14	13																	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00	Reset

LEDC_HPOINT_CH_n 该通道所选定时器计数值达到该字段的值时，输出信号翻转为高电平。(R/W)

Register 26.5. LEDC_CH_n_DUTY_REG (*n*: 0-5) (0x0008+20**n*)

(reserved)														LEDC_DUTY_CHn																		
31														19	18																	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x000																	Reset	

LEDC_DUTY_CH_n 通过控制低位点改变输出信号占空比。该通道所选定时器达到低位点时，输出信号翻转为低电平。(R/W)

Register 26.6. LEDC_CH_n_DUTY_R_REG (*n*: 0-5) (0x0010+20**n*)

(reserved)														LEDC_DUTY_R_CH ⁿ																		
31														19	18	0																
0 0 0 0 0 0 0 0 0 0 0 0 0 0														0x000																	Reset	

LEDC_DUTY_R_CH_n 存储通道 *n* 输出信号的当前占空比。(RO)

Register 26.7. LEDC_TIMER_x_CONF_REG (x: 0-3) (0x00A0+8*x)

(reserved)						LEDC_TIMER _x _PARA_UP (reserved) LEDC_TIMER _x _RST LEDC_TIMER _x _PAUSE						LEDC_CLK_DIV_TIMER _x								LEDC_TIMER _x _DUTY_RES			
31	26				25	24	23	22	21	4								3	0				
0	0	0	0	0	0	0	0	1	0	0x000								0x0				Reset	

- LEDC_TIMER_x_DUTY_RES** 用于控制定时器 _x 计数器的计数范围。(R/W)
- LEDC_CLK_DIV_TIMER_x** 用于配置定时器 _x 分频器的分频系数。低 8 位为小数部分。(R/W)
- LEDC_TIMER_x_PAUSE** 用于暂停定时器 _x 的计数器。(R/W)
- LEDC_TIMER_x_RST** 用于复位定时器 _x。复位后计数器为 0。(R/W)
- LEDC_TIMER_x_PARA_UP** 置位此位, 更新 LEDC_CLK_DIV_TIMER_x 和 LEDC_TIMER_x_DUTY_RES。
(WT)

Register 26.8. LEDC_TIMER_x_VALUE_REG (x: 0-3) (0x00A4+8*x)

(reserved)														LEDC_TIMERx_CNT																		
31														14	13	0																
0 0 0 0 0 0 0 0 0 0 0 0 0 0														0	0																	Reset

- LEDC_TIMER_x_CNT** 存储定时器 _x 的当前计数器值 (RO)

Register 26.9. LEDC_INT_RAW_REG (0x00C0)

(reserved)																																LEDC_OVF_CNT_CH5_INT_RAW LEDC_OVF_CNT_CH4_INT_RAW LEDC_OVF_CNT_CH3_INT_RAW LEDC_OVF_CNT_CH2_INT_RAW LEDC_OVF_CNT_CH1_INT_RAW LEDC_DUTY_CHNG_END_CH0_INT_RAW LEDC_DUTY_CHNG_END_CH1_INT_RAW LEDC_DUTY_CHNG_END_CH2_INT_RAW LEDC_DUTY_CHNG_END_CH3_INT_RAW LEDC_DUTY_CHNG_END_CH4_INT_RAW LEDC_TIMER3_OVF_INT_RAW LEDC_TIMER2_OVF_INT_RAW LEDC_TIMER1_OVF_INT_RAW LEDC_TIMER0_OVF_INT_RAW															
31																16																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset															

- LEDC_TIMER_x_OVF_INT_RAW** 定时器 x 达到最大计数值时触发中断。(R/WTC/SS)
- LEDC_DUTY_CHNG_END_CH_n_INT_RAW** 通道 n 的原始中断位。占空比渐变结束时触发。(R/WTC/SS)
- LEDC_OVF_CNT_CH_n_INT_RAW** 通道 n 的原始中断位。ovf_cnt 达到 LEDC_OVF_NUM_CH_n 的值时触发。(R/WTC/SS)

Register 26.10. LEDC_INT_ST_REG (0x00C4)

(reserved)																LEDC_OVF_CNT_CH5_INT_ST LEDC_OVF_CNT_CH4_INT_ST LEDC_OVF_CNT_CH3_INT_ST LEDC_OVF_CNT_CH2_INT_ST LEDC_OVF_CNT_CH1_INT_ST LEDC_DUTY_CHNG_END_CH0_INT_ST LEDC_DUTY_CHNG_END_CH1_INT_ST LEDC_DUTY_CHNG_END_CH2_INT_ST LEDC_DUTY_CHNG_END_CH3_INT_ST LEDC_DUTY_CHNG_END_CH4_INT_ST LEDC_DUTY_CHNG_END_CH5_INT_ST LEDC_TIMER3_OVF_INT_ST LEDC_TIMER2_OVF_INT_ST LEDC_TIMER1_OVF_INT_ST LEDC_TIMER0_OVF_INT_ST																
31																16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset		

- LEDC_TIMER_x_OVF_INT_ST** LEDC_TIMER_x_OVF_INT_ENA 置 1 时，LEDC_TIMER_x_OVF_INT 中断的屏蔽中断状态位。(RO)
- LEDC_DUTY_CHNG_END_CH_n_INT_ST** LEDC_DUTY_CHNG_END_CH_n_INT_ENA 置 1 时，LEDC_DUTY_CHNG_END_CH_n_INT 中断的屏蔽中断状态位。(RO)
- LEDC_OVF_CNT_CH_n_INT_ST** LEDC_OVF_CNT_CH_n_INT_ENA 置 1 时，LEDC_OVF_CNT_CH_n_INT 中断的屏蔽中断状态位。(RO)

Register 26.11. LEDC_INT_ENA_REG (0x00C8)

(reserved)																LEDC_OVF_CNT_CH5_INT_ENA LEDC_OVF_CNT_CH4_INT_ENA LEDC_OVF_CNT_CH3_INT_ENA LEDC_OVF_CNT_CH2_INT_ENA LEDC_OVF_CNT_CH1_INT_ENA LEDC_OVF_CNT_CH0_INT_ENA LEDC_DUTY_CHNG_END_CH5_INT_ENA LEDC_DUTY_CHNG_END_CH4_INT_ENA LEDC_DUTY_CHNG_END_CH3_INT_ENA LEDC_DUTY_CHNG_END_CH2_INT_ENA LEDC_DUTY_CHNG_END_CH1_INT_ENA LEDC_DUTY_CHNG_END_CH0_INT_ENA LEDC_TIMER3_OVF_INT_ENA LEDC_TIMER2_OVF_INT_ENA LEDC_TIMER1_OVF_INT_ENA LEDC_TIMER0_OVF_INT_ENA																
31																16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset	

- LEDC_TIMER_x_OVF_INT_ENA** LEDC_TIMER_x_OVF_INT 中断的使能位。(R/W)
- LEDC_DUTY_CHNG_END_CH_n_INT_ENA** LEDC_DUTY_CHNG_END_CH_n_INT 中断的使能位。(R/W)
- LEDC_OVF_CNT_CH_n_INT_ENA** LEDC_OVF_CNT_CH_n_INT 中断的使能位。(R/W)

Register 26.12. LEDC_INT_CLR_REG (0x00CC)

(reserved)																																LEDC_OVF_CNT_CH5_INT_CLR LEDC_OVF_CNT_CH4_INT_CLR LEDC_OVF_CNT_CH3_INT_CLR LEDC_OVF_CNT_CH2_INT_CLR LEDC_OVF_CNT_CH1_INT_CLR LEDC_OVF_CNT_CH0_INT_CLR LEDC_DUTY_CHNG_END_CH5_INT_CLR LEDC_DUTY_CHNG_END_CH4_INT_CLR LEDC_DUTY_CHNG_END_CH3_INT_CLR LEDC_DUTY_CHNG_END_CH2_INT_CLR LEDC_DUTY_CHNG_END_CH1_INT_CLR LEDC_DUTY_CHNG_END_CH0_INT_CLR LEDC_TIMER3_OVF_INT_CLR LEDC_TIMER2_OVF_INT_CLR LEDC_TIMER1_OVF_INT_CLR LEDC_TIMER0_OVF_INT_CLR																	
31																16																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0																0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- LEDC_TIMER_x_OVF_INT_CLR** 置位此位，清除 LEDC_TIMER_x_OVF_INT 中断。(WT)
- LEDC_DUTY_CHNG_END_CH_n_INT_CLR** 置位此位，清除 LEDC_DUTY_CHNG_END_CH_n_INT 中断。(WT)
- LEDC_OVF_CNT_CH_n_INT_CLR** 置位此位，清除 LEDC_OVF_CNT_CH_n_INT 中断。(WT)

Register 26.13. LEDC_DATE_REG (0x00FC)

LEDC_LEDC_DATE	
31	0
0x19061700	
Reset	

LEDC_LEDC_DATE 版本控制寄存器。(R/W)

27 红外遥控 (RMT)

27.1 概述

RMT（红外收发器）是一个红外发送和接收控制器，支持多种红外协议。RMT 模块可以实现将模块内置 RAM 中的脉冲编码转换为信号输出，或将模块的输入信号转换为脉冲编码存入 RAM 中。此外，RMT 模块可以选择是否对输出信号进行载波调制，也可以选择是否对输入信号进行解调和去噪处理。

RMT 共有四个通道，可独立用于信号发送或接收，编码为 0~3。0~1 通道专门用于信号发送；2~3 通道专门用于信号接收，发送通道和接收通道分别有一组功能相同的寄存器。为了方便叙述，以 n 表示各个发送通道，以 m 表示各个接收通道。

27.2 主要特性

- 两个通道支持发送
- 两个通道支持接收
- 可编程配置多个通道同时发送
- RMT 的四个通道共享 192 x 32-bit 的 RAM
- 发送脉冲支持载波调制
- 接收脉冲支持滤波和载波解调
- 乒乓发送模式
- 乒乓接收模式
- 发射器支持持续发送

27.3 功能描述

27.3.1 RMT 架构

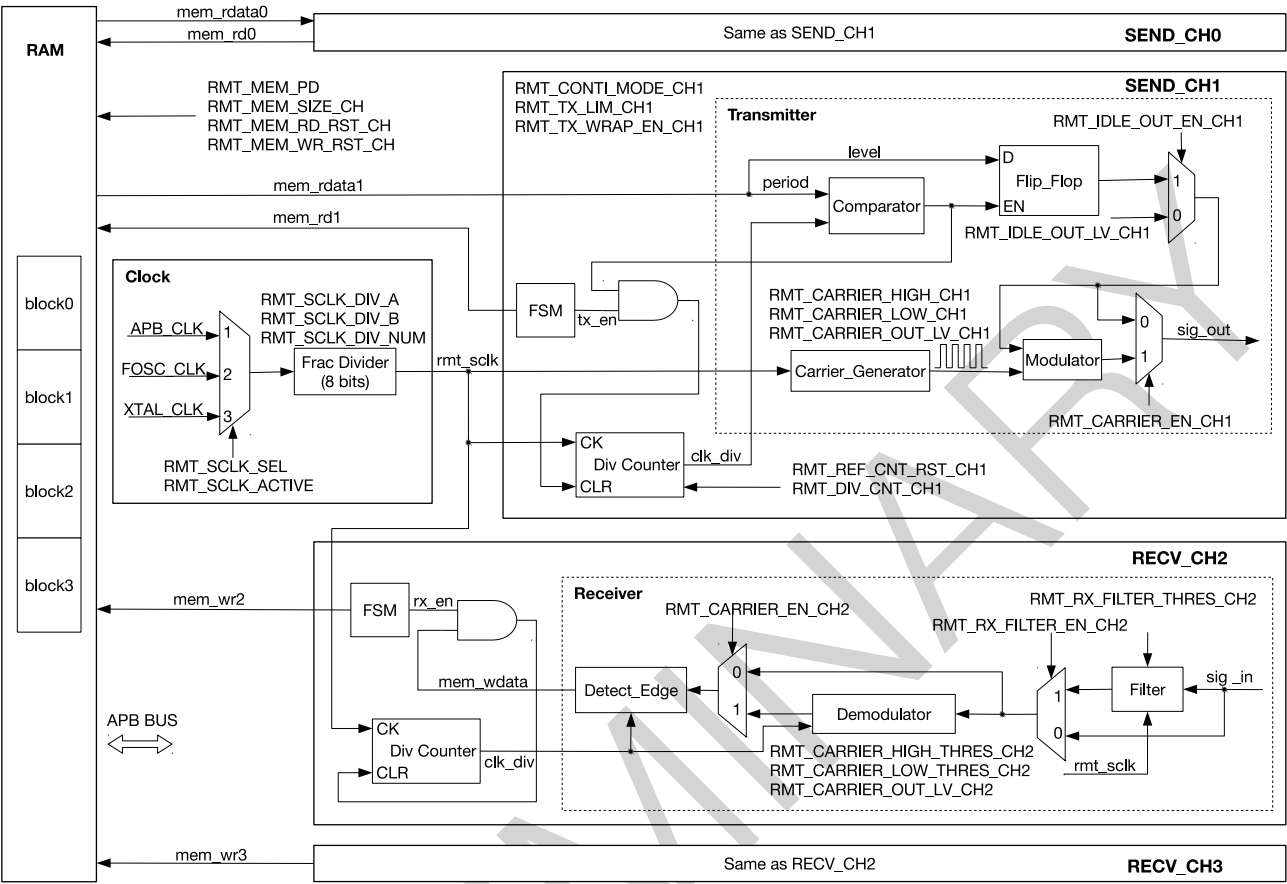


图 27-1. RMT 结构框图

RMT 模块有四个独立的通道，其中两个为发送通道，两个为接收通道。发送通道内部有各自的一个时钟分频计数器、状态机和发射器，接收通道内部有各自的一个时钟分频计数器、状态机和接收器。四个通道共享一块 192 x 32 bit 的 RAM。

27.3.2 RMT RAM

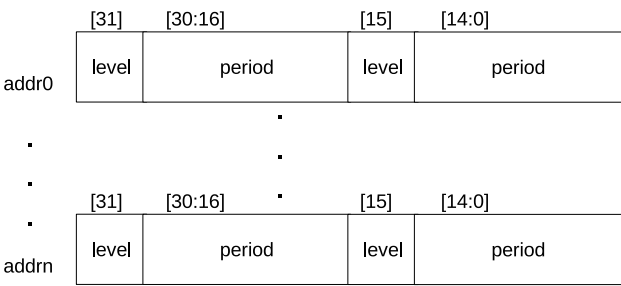


图 27-2. RAM 中脉冲编码结构

RAM 中脉冲编码结构如图 27-2 所示。每个脉冲编码为 16 位，由 level 与 period 两部分组成。其中 level 表示输入或输出信号的逻辑电平值（0 或 1），period 表示该电平信号持续的时钟（图 27-1 clk_div）周期数。Period 为

0 是传输结束标志。非结束标志的 period 的数值需要满足与 APB 时钟和 RMT 时钟相关的限制条件：

$$3 \times T_{apb_clk} + 5 \times T_{rmt_sclk} < period \times T_{clk_div}(1)$$

RAM 按照 48 x 32 位分成四个 block。默认情况下每个通道只能使用一个 block（固定为通道 0 使用 block 0，通道 1 使用 block 1，以此类推）。

当发送通道 n 或接收通道 m 单次传输的脉冲编码数大于一个 block 时，可以：

- 置位 RMT_MEM_TX_WRAP_EN_CH n/m 使能乒乓操作；
- 或通过配置 RMT_MEM_SIZE_CH n/m 寄存器，允许该通道占用多个 block。

当设置 RMT_MEM_SIZE_CH $n/m > 1$ 时，通道 n/m 将占用 block $(n/m) \sim \text{block } (n/m + \text{RMT_MEM_SIZE_CH}_{n/m} - 1)$ 的存储空间。通道 $n/m + 1 \sim n/m + \text{RMT_MEM_SIZE_CH}_{n/m} - 1$ 因为对应的 RAM block 被占用而无法使用。

注意，每个通道使用 RAM 的空间是根据地址从低到高进行映射的，因此通道 0 可以通过配置 RMT_MEM_SIZE_CH0 寄存器来使用通道 1、2、3 的 RAM 空间，但是通道 3 不能使用通道 0、1 或 2 的 RAM 空间。因此 RMT_MEM_SIZE_CH n 的最大值不应超过 $(4 - n)$ ，RMT_MEM_SIZE_CH m 的最大值不应超过 $(2 - m)$ 。

RAM 可被 APB 总线及通道的发射器或接收器访问，为了防止接收通道访问 RAM 和 APB 访问时发生冲突，用户可以通过配置 RMT_MEM_OWNER_CH m 来决定当前 RAM 的使用权。当接收通道发生越权访问时会产生 RMT_CH m _OWNER_ERR 标志信号。

APB 总线访问 RAM 有 FIFO 和直接地址 (NONFIFO) 两种模式：

- RMT_FIFO_MASK 置 1 时，选择直接地址模式；
- RMT_FIFO_MASK 置 0 时选择 FIFO 模式。

在 FIFO 模式下，APB 通过固定地址 RMT_CH n/m DATA_REG 向 RAM 写数据或从 RAM 读数据。在直接地址模式下，APB 向连续地址段写入数据，或从连续地址段读出数据。发送通道 n 对应的写地址段的首地址是：RMT 基地址 + 0x800 + $(n - 1) \times 48$ ，第 2 个数据的访问地址是 RMT 基地址 + 0x800 + $(n - 1) \times 48 + 0x4$ ，以此类推，后面的地址依次加上 0x4。接收通道 m 对应的读地址段的首地址是：RMT 基地址 + 0x860 + $(m - 1) \times 48$ ，第 2 个数据的访问地址是 RMT 基地址 + 0x860 + $(m - 1) \times 48 + 0x4$ ，以此类推，后面的地址依次加上 0x4。

当 RMT 模块不工作时，可以通过配置 RMT_MEM_FORCE_PD 寄存器使 RAM 工作于低功耗模式。

27.3.3 时钟

用户可以通过配置 RMT_SCLK_SEL 选择 RMT 的时钟源：APB_CLK、FOSC_CLK 或 XTAL_CLK，配置 RMT_SCLK_ACTIVE 为高电平来打开 RMT 的时钟。选择后的时钟经过小数分频得到 RMT 的工作时钟（图 27-1 rmt_sclk），分频系数为：

$$RMT_SCLK_DIV_NUM + 1 + RMT_SCLK_DIV_A/RMT_SCLK_DIV_B$$

更多信息，请参考章节 6 复位和时钟。RMT_DIV_CNT_CH n/m 用于配置 RMT 通道内部的时钟分频器的分频系数，除 0 表示 256 分频外，其它分频数等同于 RMT_DIV_CNT_CH n/m 的值。时钟分频器可以通过配置 RMT_REF_CNT_RST_CH n/m 进行复位。时钟分频器的分频时钟可供计数器使用。

27.3.4 发射器

27.3.4.1 普通发送模式

当 `RMT_TX_START_CH n` 置为 1 时, 通道 n 的发射器开始从通道对应 RAM block 的起始地址, 按照地址从低到高依次读取脉冲编码进行发送。当遇到结束标志 (period 等于 0) 时, 发射器将结束发送返回空闲状态, 并产生 `RMT_CH n _TX_END_INT` 中断。配置 `RMT_TX_STOP_CH n` 可以使发射器立刻停止发送并进入空闲状态。发射器空闲状态发送的电平由结束标志中的 level 段或者是 `RMT_IDLE_OUT_LV_CH n` 决定。用户可以配置 `RMT_IDLE_OUT_EN_CH n` 来选择这两种方式。这些配置都需要用置位 `RMT_CONF_UPDATE_CH n` 的方法来更新进入发射器, 详情见第 27.3.6 小节。

27.3.4.2 乒乓发送模式

当发送的脉冲编码较多时, 可通过置位 `RMT_MEM_TX_WRAP_EN_CH n` 使能乒乓操作。在乒乓操作模式下, 发射器会循环从通道对应的 RAM 区域取出脉冲编码进行发送, 直至遇到结束标识为止。例如, 当 `RMT_MEM_SIZE_CH n` = 1 时, 发射器将从 $48 * n$ 地址开始发送, 然后对应 RAM 的地址递增。发完 $(48 * (n + 1) - 1)$ 地址的数据后, 下次继续从 $48 * n$ 地址开始递增发送数据, 依此类推, 遇到结束标识时停止发送。`RMT_MEM_SIZE_CH n` > 1 的情形下, 乒乓操作同样适用。

每当发射器发送的脉冲编码数大于等于 `RMT_TX_LIM_CH n` 时, 会产生 `RMT_CH n _TX_THR_EVENT_INT` 中断。在乒乓模式下, 可以设置 `RMT_TX_LIM_CH n` 为每个通道对应 RAM 空间的一半或几分之一。软件在检测到 `RMT_CH n _TX_THR_EVENT_INT` 中断之后, 可以更新已使用过的 RAM 区域的脉冲编码, 从而实现乒乓操作。

发送乒乓模式用到的 `RMT_MEM_TX_WRAP_EN_CH n` 、`RMT_MEM_SIZE_CH n` 和 `RMT_TX_LIM_CH n` 参数, 都需要用置位 `RMT_CONF_UPDATE_CH n` 的方法来更新进入发射器。详情见第 27.3.6 小节。

27.3.4.3 发送加载波

此外, 发射器还可以对输出信号进行载波调制, 置位 `RMT_CARRIER_EN_CH n` 可以使能该功能。载波的波形可配置。一个载波周期中高电平持续时间为 $(RMT_CARRIER_HIGH_CH n + 1)$ 个 `rmt_sclk` 时钟周期, 低电平持续的时间为 $(RMT_CARRIER_LOW_CH n + 1)$ 个 `rmt_sclk` 时钟周期。置位 `RMT_CARRIER_OUT_LV_CH n` 时在输出信号高电平上加载波信号, 清零 `RMT_CARRIER_OUT_LV_CH n` 时在输出信号低电平上加载波信号。同时, 在进行载波调制时, 载波可以一直加载在输出信号上, 也可以仅加载在有效的脉冲编码 (RAM 中的数据) 上。通过配置 `RMT_CARRIER_EFF_EN_CH n` 寄存器, 可以选择这两种模式。`RMT_CARRIER_EFF_EN_CH n` 设置为 0 时, 在所有信号上加载波, 设置为 1 时在有效信号上加载波。

上述载波调制中所需的配置参数, 都需要用置位 `RMT_CONF_UPDATE_CH n` 的方法来更新进入发射器, 详情见第 27.3.6 小节。

27.3.4.4 持续发送模式

置位 `RMT_TX_CONTI_MODE_CH n` 可以使能发射器的持续发送模式。置位该寄存器后, 发射器会循环发送 RAM 中的脉冲编码。持续发送模式下, 如果遇到结束标志, 会重新开始发送第一个数据; 如果没有结束标志, 会在发送到最后数据处回卷, 再重新开始发送第一个数据。配置 `RMT_TX_LOOP_CNT_EN_CH n` 后, 发射器每遇到一次结束标志, 循环发送的次数会加 1。当该次数达到 `RMT_TX_LOOP_NUM_CH n` 设定的值时, 会产生 `RMT_CH n _TX_LOOP_INT` 中断。持续发送模式下, 如果遇到的结束标志类型是 `period[11:0]` 为 0, 那么这个结束标志前一个数据的 `period` 需要满足:

$$6 \times T_{apb_clk} + 12 \times T_{rmt_sclk} < period \times T_{clk_div}(2)$$

而其它数据的 period，满足上述的关系式 (1) 即可。

上述所需的配置参数，都需要用置位 RMT_CONF_UPDATE_CH n 的方法来更新进入发射器，详情见第 27.3.6 小节。

27.3.4.5 多通道同时发送

首先配置 RMT_TX_SIM_CH n 用于选择同步发送的通道，然后置位 RMT_TX_SIM_EN 可以使能发射器多个通道同步发送的功能，将所选同步发送通道的 RMT_TX_START_CH n 置为 1，当最后一个通道完成配置时，此时多个通道会同时启动发送。由于硬件条件的限制，两条通道不能保证完全同时开始发送，两条通道发送的时间间隔在 $3 \times T_{clk_div}$ 以内。RMT_TX_SIM_EN 需要用置位 RMT_CONF_UPDATE_CH n 的方法来更新进入发射器，详情见第 27.3.6 小节。

27.3.5 接收器

27.3.5.1 普通接收模式

RMT_RX_EN_CH m 置为 1 时接收器开始工作，置为 0 时会停止接收。接收器会从信号的第一个跳变沿开始计数，并检测信号电平及其持续的时钟周期数，将其按照脉冲编码的格式存入 RAM 中。当信号在一个电平下持续的时钟周期数超过 RMT_IDLE_THRES_CH m 时，接收器结束接收过程，返回空闲状态，并产生 RMT_CH m _RX_END_INT 中断。需要将 RMT_IDLE_THRES_CH m 设置为超过接收电平的最大时钟周期数，否则会导致将正常的接收电平误判为进入空闲状态的后果。当接收数据存满了接收通道设置的 RAM 空间时，会停止接收，在非乒乓模式下会产生 RMT_CH n _ERR_INT 中断（由 RAM 满事件触发）。

上述所需的配置参数，都需要用置位 RMT_CONF_UPDATE_CH m 的方法来更新进入接收器，详情见第 27.3.6 小节。

27.3.5.2 乒乓接收模式

当接收的脉冲编码较多时，可通过置位 RMT_MEM_RX_WRAP_EN_CH m 使能通道 m 的乒乓操作。在乒乓操作模式下，接收器会将接收到的脉冲编码循环存入通道对应的 RAM 区域。当信号在一个电平下持续的时钟周期数超过 RMT_IDLE_THRES_CH m 时，接收器结束接收过程，返回空闲状态，并产生 RMT_CH m _RX_END_INT 中断。例如，当 RMT_MEM_SIZE_CH m = 1 时，接收器将从 $48 * m$ 地址开始接收，然后对应 RAM 的地址递增。收完 $(48 * (m + 1) - 1)$ 地址的数据后，下次继续从 $48 * m$ 地址开始递增接收数据，以此类推，遇到信号在一个电平下持续的时钟周期数超过 RMT_IDLE_THRES_CH m 时停止接收。RMT_MEM_SIZE_CH m > 1 的情形下，乒乓操作同样适用。

每当接收器接收的脉冲编码数大于等于 RMT_RX_LIM_CH m 时，会产生 RMT_CH m _RX_THR_EVENT_INT 中断。在乒乓模式下，可以设置 RMT_RX_LIM_CH m 为每个通道对应 RAM 空间的一半或几分之一。软件在检测到 RMT_CH m _RX_THR_EVENT_INT 中断之后，可以回收已使用过的 RAM 区域的脉冲编码，从而实现乒乓操作。

上述所需的配置参数，都需要用置位 RMT_CONF_UPDATE_CH m 的方法来更新进入发射器，详情见第 27.3.6 小节。

27.3.5.3 接收滤波

每个通道都可以通过置位 RMT_RX_FILTER_EN_CH m 使能接收器对输入信号进行滤波的功能。滤波器的功能为连续采样输入信号，如果输入信号在连续 RMT_RX_FILTER_THRES_CH m 个 rmt_sclk 时钟周期内保持不变，则

输入信号有效，否则输入信号无效。只有有效的输入信号才能通过滤波器。因此，滤波器会滤除脉冲宽度小于 RMT_RX_FILTER_THRES_CH n 个 rmt_sclk 时钟周期的线路毛刺。

上述所需的配置参数，都需要用置位 RMT_CONF_UPDATE_CH m 的方法来更新进入发射器，详情见第 27.3.6 小节。

27.3.5.4 接收去载波

此外，接收器还可以对输入信号或滤波后的输出信号进行去载波调制，置位 RMT_CARRIER_EN_CH m 可以使能该功能。去载波分为去除高电平载波和低电平载波两种类型：

- 置位 RMT_CARRIER_OUT_LV_CH m ，设置去除高电平载波；
- 清零 RMT_CARRIER_OUT_LV_CH m ，设置去除低电平载波。

配置 RMT_CARRIER_HIGH_THRES_CH m 和 RMT_CARRIER_LOW_THRES_CH m 来设置去载波的高电平阈值和低电平阈值。当信号的高电平持续时间小于 RMT_CARRIER_HIGH_THRES_CH m 个 clk_div 分频时钟周期，或者信号的低电平持续时间小于 RMT_CARRIER_LOW_THRES_CH m 个 clk_div 分频时钟周期，就会被认为是载波被滤除。

上述所需的配置参数，都需要用置位 RMT_CONF_UPDATE_CH m 的方法来更新进入接收器，详情见第 27.3.6 小节。

27.3.6 配置参数更新

RMT 的配置寄存器均需要配置各自通道的 RMT_CONF_UPDATE_CH n/m 寄存器位来更新进入各自通道，更新方法是向 RMT_CONF_UPDATE_CH n/m 写入高电平。发送通道和接收通道需要通过这种方法更新的配置参数列在下表中。

表 27-1. 更新配置参数

配置寄存器	配置参数
发送通道	
RMT_CH n CONF0_REG	RMT_CARRIER_OUT_LV_CH n
	RMT_CARRIER_EN_CH n
	RMT_CARRIER_EFF_EN_CH n
	RMT_DIV_CNT_CH n
	RMT_TX_STOP_CH n
	RMT_IDLE_OUT_EN_CH n
	RMT_IDLE_OUT_LV_CH n
	RMT_TX_CONTI_MODE_CH n
RMT_CH n CARRIER_DUTY_REG	RMT_CARRIER_HIGH_CH n
	RMT_CARRIER_LOW_CH n
RMT_CH n TX_LIM_REG	RMT_TX_LOOP_CNT_EN_CH n
	RMT_TX_LOOP_NUM_CH n
	RMT_TX_LIM_CH n
RMT_CH n TX_SIM_REG	RMT_TX_SIM_EN
接收通道	
RMT_CH m CONF0_REG	RMT_CARRIER_OUT_LV_CH m
	RMT_CARRIER_EN_CH m

接上页

配置寄存器	配置参数
	RMT_IDLE_THRES_CH m
	RMT_DIV_CNT_CH m
RMT_CH m CONF1_REG	RMT_RX_FILTER_THRES_CH m
	RMT_RX_EN_CH m
RMT_CH m RX_CARRIER_RM_REG	RMT_CARRIER_HIGH_THRES_CH m
	RMT_CARRIER_LOW_THRES_CH m
RMT_CH m RX_LIM_REG	RMT_RX_LIM_CH m
RMT_REF_CNT_RST_REG	RMT_REF_CNT_RST_CH m

27.3.7 中断

- RMT_CH n/m _ERR_INT：当通道 n/m 发生读写数据不正确，或内存空满错误时，即触发此中断。
- RMT_CH n _TX_THR_EVENT_INT：发射器每发送 [RMT_CH \$n\$ _TX_LIM_REG](#) 的数据，即触发一次此中断。
- RMT_CH m _RX_THR_EVENT_INT：接收器每接收 [RMT_CH \$m\$ _RX_LIM_REG](#) 的数据，即触发一次此中断。
- RMT_CH n _TX_END_INT：当发射器停止发送信号时，即触发此中断。
- RMT_CH m _RX_END_INT：当接收器停止接收信号时，即触发此中断。
- RMT_CH n _TX_LOOP_INT：发射器处于循环发送模式时，当循环次数达到 [RMT_TX_LOOP_NUM_CH \$n\$](#) 的值后，会产生此中断。

27.4 寄存器列表

本小节的所有地址均为相对于 RMT 基地址的地址偏移量（相对地址），具体基地址请见章节 3 [系统和存储器](#) 中的表 3-4。

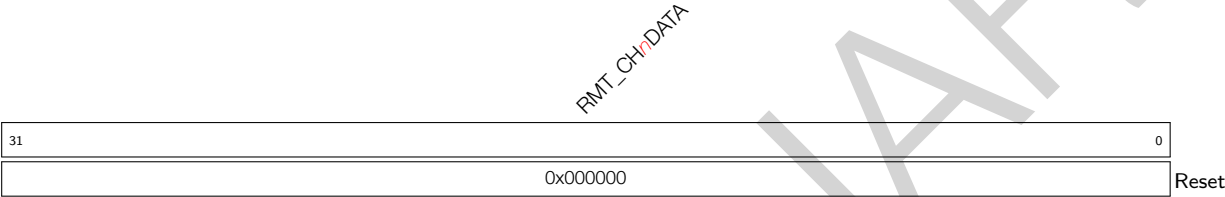
名称	描述	地址	访问
FIFO 读/写寄存器			
RMT_CH0DATA_REG	通道 0 通过 APB FIFO 进行读写操作时用到的数据寄存器	0x0000	RO
RMT_CH1DATA_REG	通道 1 通过 APB FIFO 进行读写操作时用到的数据寄存器	0x0004	RO
RMT_CH2DATA_REG	通道 2 通过 APB FIFO 进行读写操作时用到的数据寄存器	0x0008	RO
RMT_CH3DATA_REG	通道 3 通过 APB FIFO 进行读写操作时用到的数据寄存器	0x000C	RO
配置寄存器			
RMT_CH0CONF0_REG	通道 0 的配置寄存器 0	0x0010	varies
RMT_CH1CONF0_REG	通道 1 的配置寄存器 0	0x0014	varies
RMT_CH2CONF0_REG	通道 2 的配置寄存器 0	0x0018	R/W
RMT_CH2CONF1_REG	通道 2 的配置寄存器 1	0x001C	varies
RMT_CH3CONF0_REG	通道 3 的配置寄存器 0	0x0020	R/W
RMT_CH3CONF1_REG	通道 3 的配置寄存器 1	0x0024	varies
RMT_SYS_CONF_REG	RMT APB 配置寄存器	0x0068	R/W
RMT_REF_CNT_RST_REG	RMT 时钟分频器复位寄存器	0x0070	WT
Status registers			
RMT_CH0STATUS_REG	通道 0 的状态寄存器	0x0028	RO
RMT_CH1STATUS_REG	通道 1 的状态寄存器	0x002C	RO
RMT_CH2STATUS_REG	通道 2 的状态寄存器	0x0030	RO
RMT_CH3STATUS_REG	通道 3 的状态寄存器	0x0034	RO
中断寄存器			
RMT_INT_RAW_REG	原始中断状态寄存器	0x0038	R/ WTC/ SS
RMT_INT_ST_REG	屏蔽中断状态寄存器	0x003C	RO
RMT_INT_ENA_REG	中断使能寄存器	0x0040	R/W
RMT_INT_CLR_REG	中断清零寄存器	0x0044	WT
载波占空比寄存器			
RMT_CH0CARRIER_DUTY_REG	通道 0 的占空比配置寄存器	0x0048	R/W
RMT_CH1CARRIER_DUTY_REG	通道 1 的占空比配置寄存器	0x004C	R/W
RMT_CH2_RX_CARRIER_RM_REG	通道 2 的去载波寄存器	0x0050	R/W
RMT_CH3_RX_CARRIER_RM_REG	通道 3 的去载波寄存器	0x0054	R/W
TX 事件配置寄存器			
RMT_CH0_TX_LIM_REG	通道 0 的 TX 事件配置寄存器	0x0058	varies
RMT_CH1_TX_LIM_REG	通道 1 的 TX 事件配置寄存器	0x005C	varies
RMT_TX_SIM_REG	RMT TX 同步发送寄存器	0x006C	R/W

名称	描述	地址	访问
RX 事件配置寄存器			
RMT_CH2_RX_LIM_REG	通道 2 RX 事件配置寄存器	0x0060	R/W
RMT_CH3_RX_LIM_REG	通道 3 RX 事件配置寄存器	0x0064	R/W
版本寄存器			
RMT_DATE_REG	版本控制寄存器	0x00CC	R/W

27.5 寄存器

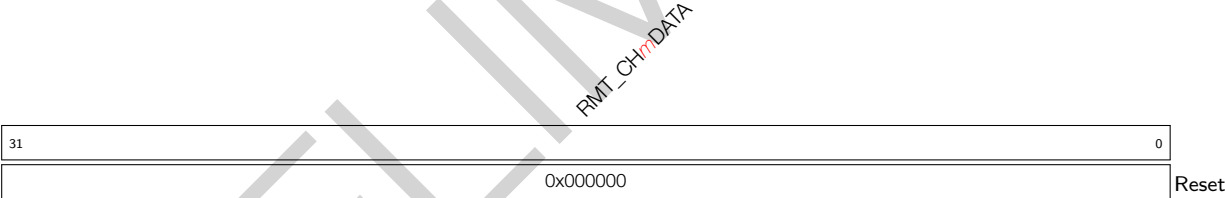
本小节的所有地址均为相对于 RMT 基地址的地址偏移量（相对地址），具体基地址请见章节 [3 系统和存储器](#) 中的表 3-4。

Register 27.1. RMT_CH n DATA_REG ($n = 0, 1$) (0x0000, 0x0004)



RMT_CH n DATA 通道 n 通过 APB FIFO 进行读写操作时用到的数据寄存器。(RO)

Register 27.2. RMT_CH m DATA_REG ($m = 2, 3$) (0x0008, 0x000C)



RMT_CH m DATA 通道 m 通过 APB FIFO 进行读写操作时用到的数据寄存器。(RO)

Register 27.4. RMT_CH m CONF0_REG ($m = 2, 3$) (0x0018, 0x0020)

(reserved)		RMT_CARRIER_OUT_LV_CH m		RMT_CARRIER_EN_CH m		(reserved)		RMT_MEM_SIZE_CH m		RMT_IDLE_THRES_CH m								RMT_DIV_CNT_CH m	
31	30	29	28	27	26	25	23	22		8	7								0
0	0	1	1	0	0	0x1				0x7ff								0x2	Reset

RMT_DIV_CNT_CH m 配置通道 m 的时钟分频器。(R/W)

RMT_IDLE_THRES_CH m 配置接收阈值。接收器长时间检测不到信号变化，且持续的时间大于 RMT_IDLE_THRES_CH m 的值，则接收器停止接收过程。(R/W)

RMT_MEM_SIZE_CH m 配置通道 m 可用的最大 RAM Block 数量。(R/W)

RMT_CARRIER_EN_CH m 通道 m 的载波调制使能控制位。1：对输出信号进行载波调制和解调；0：禁止对输出信号进行载波调制和解调。(R/W)

RMT_CARRIER_OUT_LV_CH m 配置通道 m 的载波调制方式。(R/W)

1'h0：载波加载在低电平上；

1'h1：载波加载在高电平上。

Register 27.5. RMT_CH m CONF1_REG ($m = 2, 3$) (0x001C, 0x0024)

(reserved)																RMT_CONF_UPDATE_CH ^m				(reserved)				RMT_MEM_RX_WRAP_EN_CH ^m				RMT_RX_FILTER_THRES_CH ^m				RMT_RX_FILTER_EN_CH ^m				RMT_MEM_OWNER_CH ^m				RMT_APB_MEM_RST_CH ^m				RMT_MEM_WR_RST_CH ^m				RMT_RX_EN_CH ^m			
31																	15	14	13	12					5	4	3	2	1	0																					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0xf	0	1	0	0	0	0	Reset																									

- RMT_RX_EN_CH m** 置位此位，使能通道 m 的接收器，接收器开始接收数据。(R/W)
- RMT_MEM_WR_RST_CH m** 置位此位，则通道 m 通过接收器访问的 RAM 写地址将被复位。(WT)
- RMT_APB_MEM_RST_CH m** 置位此位，则通道 m 通过 APB FIFO 访问的读/写 RAM 地址将被复位。(WT)
- RMT_MEM_OWNER_CH m** 标志通道 m 的 RAM 使用权。(R/W/SC)
- 1'h1: 接收器有权使用该通道的 RAM;
 - 1'h0: APB 总线有权使用该通道的 RAM。
- RMT_RX_FILTER_EN_CH m** 通道 m 的接收器滤波功能使能位。(R/W)
- RMT_RX_FILTER_THRES_CH m** 接收模式下，忽略宽度小于 RMT_RX_FILTER_THRES_CH m 个 rmt_sclk 周期的输入脉冲。(R/W)
- RMT_MEM_RX_WRAP_EN_CH m** 置位此位，使能通道 m 的乒乓接收模式。在这种模式下，如果待接收的数据长度大于该通道的 RAM Block 长度，则接收器将继续把接收数据存入 RAM 的第一个地址，依次循环。(R/W)
- RMT_CONF_UPDATE_CH m** 通道 m 的同步位。(WT)

Register 27.6. RMT_SYS_CONF_REG (0x0068)

RMT_CLK_EN		(reserved)		RMT_SCLK_ACTIVE		RMT_SCLK_SEL		RMT_SCLK_DIV_B		RMT_SCLK_DIV_A		RMT_SCLK_DIV_NUM		RMT_MEM_FORCE_PU		RMT_MEM_FORCE_PD		RMT_APB_FIFO_MASK	
31	30	27	26	25	24	23	18	17	12	11	4	3	2	1	0				
0	0	0	0	0	1	0x1	0x0		0x0		0x1		0	0	0	0	Reset		

RMT_APB_FIFO_MASK 1'h1: 直接访问 RAM; 1'h0: 通过 FIFO 访问 RAM。(R/W)

RMT MEM CLK FORCE ON 置位此位，使能 RMT 的 RAM 时钟。(R/W)

RMT_MEM_FORCE_PD 置位此位，关闭 RMT RAM。(R/W)

RMT_MEM_FORCE_PU 1: 禁用 RMT RAM 的 Light-sleep 低功耗模式; 0: RMT 处于 Light-sleep 模式时, 关闭 RMT RAM。(R/W)

RMT_SCLK_DIV_NUM 小数分频器的整数部分。(R/W)

RMT SCLK DIV A 小数分频器的分子。(R/W)

RMT SCLK DIV B 小数分频器的分母。(R/W)

RMT_SCLK_SEL 设置 rmt_sclk 的时钟源：1: APB_CLK；2: FOSC_CLK；3: XTAL_CLK。(R/W)

RMT_SCLK_ACTIVE rmt_sclk 控制开关。(R/W)

RMT_CLK_EN RMT 寄存器的时钟门控使能位。1：打开寄存器的驱动时钟；0：关闭寄存器的驱动时钟。(R/W)

Register 27.7. RMT_REF_CNT_RST_REG (0x0070)

[illegible]

RMT REF CNT RST CH0 复位通道 0 的时钟分频器。(WT)

RMT_REF_CNT_RST_CH1 复位通道 1 的时钟分频器。(WT)

RMT_REF_CNT_RST_CH2 复位通道 2 的时钟分频器。(WT)

RMT REF CNT RST CH3 复位通道 3 的时钟分频器。(WT)

Register 27.8. RMT_CH n STATUS_REG ($n = 0, 1$) (0x0028, 0x002C)

RMT_APB_MEM_RADDR_CH _n				RMT_APB_MEM_WR_ERR_CH _n				RMT_APB_MEM_WADDR_CH _n				RMT_STATE_CH _n				RMT_MEM_RADDR_EX_CH _n																											
31				24				23				22				21				20				12				11				9				8				0			
0x0				0				0				0				0				0				0				0				0				0				0			

Register 27.10. RMT_INT_RAW_REG (0x0038)

(reserved)																												RMT_CH1_TX_LOOP_INT_RAW	RMT_CH0_TX_LOOP_INT_RAW	RMT_CH3_RX_THR_EVENT_INT_RAW	RMT_CH2_RX_THR_EVENT_INT_RAW	RMT_CH1_TX_THR_EVENT_INT_RAW	RMT_CH0_TX_THR_EVENT_INT_RAW	RMT_CH3_ERR_INT_RAW	RMT_CH2_ERR_INT_RAW	RMT_CH1_ERR_INT_RAW	RMT_CH0_ERR_INT_RAW	RMT_CH3_RX_END_INT_RAW	RMT_CH2_RX_END_INT_RAW	RMT_CH1_TX_END_INT_RAW	RMT_CH0_TX_END_INT_RAW
31														14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													

RMT_CH0_TX_END_INT_RAW [RMT_CH0_TX_END_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH1_TX_END_INT_RAW [RMT_CH1_TX_END_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH2_RX_END_INT_RAW [RMT_CH2_RX_END_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH3_RX_END_INT_RAW [RMT_CH3_RX_END_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH0_ERR_INT_RAW [RMT_CH0_ERR_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH1_ERR_INT_RAW [RMT_CH1_ERR_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH2_ERR_INT_RAW [RMT_CH2_ERR_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH3_ERR_INT_RAW [RMT_CH3_ERR_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH0_TX_THR_EVENT_INT_RAW [RMT_CH0_TX_THR_EVENT_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH1_TX_THR_EVENT_INT_RAW [RMT_CH1_TX_THR_EVENT_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH2_RX_THR_EVENT_INT_RAW [RMT_CH2_RX_THR_EVENT_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH3_RX_THR_EVENT_INT_RAW [RMT_CH3_RX_THR_EVENT_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH0_TX_LOOP_INT_RAW [RMT_CH0_TX_LOOP_INT](#) 的原始中断位。(R/WTC/SS)

RMT_CH1_TX_LOOP_INT_RAW [RMT_CH1_TX_LOOP_INT](#) 的原始中断位。(R/WTC/SS)

Register 27.11. RMT_INT_ST_REG (0x003C)

(reserved)														RMT_CH1_TX_LOOP_INT_ST RMT_CH0_TX_LOOP_INT_ST RMT_CH3_RX_THR_EVENT_INT_ST RMT_CH2_RX_THR_EVENT_INT_ST RMT_CH1_TX_THR_EVENT_INT_ST RMT_CH0_TX_THR_EVENT_INT_ST RMT_CH3_ERR_INT_ST RMT_CH2_ERR_INT_ST RMT_CH1_ERR_INT_ST RMT_CH0_ERR_INT_ST RMT_CH3_RX_END_INT_ST RMT_CH2_RX_END_INT_ST RMT_CH1_TX_END_INT_ST RMT_CH0_TX_END_INT_ST															
31														14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

RMT_CH0_TX_END_INT_ST [RMT_CH0_TX_END_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH1_TX_END_INT_ST [RMT_CH1_TX_END_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH2_RX_END_INT_ST [RMT_CH2_RX_END_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH3_RX_END_INT_ST [RMT_CH3_RX_END_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH0_ERR_INT_ST [RMT_CH0_ERR_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH1_ERR_INT_ST [RMT_CH1_ERR_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH2_ERR_INT_ST [RMT_CH2_ERR_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH3_ERR_INT_ST [RMT_CH3_ERR_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH0_TX_THR_EVENT_INT_ST [RMT_CH0_TX_THR_EVENT_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH1_TX_THR_EVENT_INT_ST [RMT_CH1_TX_THR_EVENT_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH2_RX_THR_EVENT_INT_ST [RMT_CH2_RX_THR_EVENT_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH3_RX_THR_EVENT_INT_ST [RMT_CH3_RX_THR_EVENT_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH0_TX_LOOP_INT_ST [RMT_CH0_TX_LOOP_INT](#) 的屏蔽中断状态位。(RO)

RMT_CH1_TX_LOOP_INT_ST [RMT_CH1_TX_LOOP_INT](#) 的屏蔽中断状态位。(RO)

Register 27.12. RMT_INT_ENA_REG (0x0040)

(reserved)														RMT_CH1_TX_LOOP_INT_ENA RMT_CH0_TX_LOOP_INT_ENA RMT_CH3_RX_THR_EVENT_INT_ENA RMT_CH2_RX_THR_EVENT_INT_ENA RMT_CH1_TX_THR_EVENT_INT_ENA RMT_CH0_TX_THR_EVENT_INT_ENA RMT_CH3_ERR_INT_ENA RMT_CH2_ERR_INT_ENA RMT_CH1_ERR_INT_ENA RMT_CH0_ERR_INT_ENA RMT_CH3_RX_END_INT_ENA RMT_CH2_RX_END_INT_ENA RMT_CH1_TX_END_INT_ENA RMT_CH0_TX_END_INT_ENA															
31														14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reset
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

- RMT_CH0_TX_END_INT_ENA [RMT_CH0_TX_END_INT](#) 的中断使能位。(R/W)
- RMT_CH1_TX_END_INT_ENA [RMT_CH1_TX_END_INT](#) 的中断使能位。(R/W)
- RMT_CH2_RX_END_INT_ENA [RMT_CH2_RX_END_INT](#) 的中断使能位。(R/W)
- RMT_CH3_RX_END_INT_ENA [RMT_CH3_RX_END_INT](#) 的中断使能位。(R/W)
- RMT_CH0_ERR_INT_ENA [RMT_CH0_ERR_INT](#) 的中断使能位。(R/W)
- RMT_CH1_ERR_INT_ENA [RMT_CH1_ERR_INT](#) 的中断使能位。(R/W)
- RMT_CH2_ERR_INT_ENA [RMT_CH2_ERR_INT](#) 的中断使能位。(R/W)
- RMT_CH3_ERR_INT_ENA [RMT_CH3_ERR_INT](#) 的中断使能位。(R/W)
- RMT_CH0_TX_THR_EVENT_INT_ENA [RMT_CH0_TX_THR_EVENT_INT](#) 的中断使能位。(R/W)
- RMT_CH1_TX_THR_EVENT_INT_ENA [RMT_CH1_TX_THR_EVENT_INT](#) 的中断使能位。(R/W)
- RMT_CH2_RX_THR_EVENT_INT_ENA [RMT_CH2_RX_THR_EVENT_INT](#) 的中断使能位。(R/W)
- RMT_CH3_RX_THR_EVENT_INT_ENA [RMT_CH3_RX_THR_EVENT_INT](#) 的中断使能位。(R/W)
- RMT_CH0_TX_LOOP_INT_ENA [RMT_CH0_TX_LOOP_INT](#) 的中断使能位。(R/W)
- RMT_CH1_TX_LOOP_INT_ENA [RMT_CH1_TX_LOOP_INT](#) 的中断使能位。(R/W)

Register 27.13. RMT_INT_CLR_REG (0x0044)

(reserved)														RMT_CH1_TX_LOOP_INT_CLR RMT_CH0_TX_LOOP_INT_CLR RMT_CH3_RX_THR_EVENT_INT_CLR RMT_CH2_RX_THR_EVENT_INT_CLR RMT_CH1_TX_THR_EVENT_INT_CLR RMT_CH0_TX_THR_EVENT_INT_CLR RMT_CH3_ERR_INT_CLR RMT_CH2_ERR_INT_CLR RMT_CH1_ERR_INT_CLR RMT_CH0_ERR_INT_CLR RMT_CH3_RX_END_INT_CLR RMT_CH2_RX_END_INT_CLR RMT_CH1_TX_END_INT_CLR RMT_CH0_TX_END_INT_CLR																
31														14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset

- RMT_CH0_TX_END_INT_CLR RMT_CH0_TX_END_INT 的中断清除位。(WT)
- RMT_CH1_TX_END_INT_CLR RMT_CH1_TX_END_INT 的中断清除位。(WT)
- RMT_CH2_RX_END_INT_CLR RMT_CH2_RX_END_INT 的中断清除位。(WT)
- RMT_CH3_RX_END_INT_CLR RMT_CH3_RX_END_INT 的中断清除位。(WT)
- RMT_CH0_ERR_INT_CLR RMT_CH0_ERR_INT 的中断清除位。(WT)
- RMT_CH1_ERR_INT_CLR RMT_CH1_ERR_INT 的中断清除位。(WT)
- RMT_CH2_ERR_INT_CLR RMT_CH2_ERR_INT 的中断清除位。(WT)
- RMT_CH3_ERR_INT_CLR RMT_CH3_ERR_INT 的中断清除位。(WT)
- RMT_CH0_TX_THR_EVENT_INT_CLR RMT_CH0_TX_THR_EVENT_INT 的中断清除位。(WT)
- RMT_CH1_TX_THR_EVENT_INT_CLR RMT_CH1_TX_THR_EVENT_INT 的中断清除位。(WT)
- RMT_CH2_RX_THR_EVENT_INT_CLR RMT_CH2_RX_THR_EVENT_INT 的中断清除位。(WT)
- RMT_CH3_RX_THR_EVENT_INT_CLR RMT_CH3_RX_THR_EVENT_INT 的中断清除位。(WT)
- RMT_CH0_TX_LOOP_INT_CLR RMT_CH0_TX_LOOP_INT 的中断清除位。(WT)
- RMT_CH1_TX_LOOP_INT_CLR RMT_CH1_TX_LOOP_INT 的中断清除位。(WT)

Register 27.14. RMT_CH_{*n*}CARRIER_DUTY_REG (*n* = 0, 1) (0x0048, 0x004C)

RMT_CARRIER_HIGH_CH ⁿ																RMT_CARRIER_LOW_CH ⁿ														
31																16	15												0	
0x40																0x40														Reset

- RMT_CARRIER_LOW_CH_{*n*} 配置通道 *n* 载波的低电平时钟周期。(R/W)
- RMT_CARRIER_HIGH_CH_{*n*} 配置通道 *n* 载波的高电平时钟周期。(R/W)

Register 27.15. RMT_CH_m_RX_CARRIER_RM_REG ($m = 2, 3$) (0x0050, 0x0054)

RMT_CARRIER_HIGH_THRES_CH _m																RMT_CARRIER_LOW_THRES_CH _m																
31															16	15															0	
0x00																0x00																Reset

RMT_CARRIER_LOW_THRES_CH_m 载波调制模式下, 通道 m 低电平周期为 RMT_CARRIER_LOW_THRES_CH_m + 1。 (R/W)

RMT_CARRIER_HIGH_THRES_CH_m 载波调制模式下, 通道 m 高电平周期为 RMT_CARRIER_HIGH_THRES_CH_m + 1。 (R/W)

Register 27.16. RMT_CH_n_TX_LIM_REG ($n = 0, 1$) (0x0058, 0x005C)

(reserved)												RMT_LOOP_COUNT_RESET_CH _n RMT_TX_LOOP_CNT_EN_CH _n RMT_TX_LOOP_NUM_CH _n												RMT_TX_LIM_CH _n											
31											21	20	19	18									9	8											0
0 0 0 0 0 0 0 0 0 0												0	0	0								0x80												Reset	

RMT_TX_LIM_CH_n 配置通道 n 发送脉冲编码数量的上限值。 (R/W)

RMT_TX_LOOP_NUM_CH_n 配置持续发送模式下最大循环发送次数。 (R/W)

RMT_TX_LOOP_CNT_EN_CH_n 置位此位, 使能循环次数计数。 (R/W)

RMT_LOOP_COUNT_RESET_CH_n 重置持续发送模式下的循环计数器。 (WT)

Register 27.17. RMT_TX_SIM_REG (0x006C)

(reserved)																															RMT_TX_SIM_EN RMT_TX_SIM_CH1 RMT_TX_SIM_CH0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
31																															3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

- RMT_TX_SIM_CH0** 置位此位，使能通道 0 与其它启用的通道同步开始发送数据。(R/W)
- RMT_TX_SIM_CH1** 置位此位，使能通道 1 与其它启用的通道同步开始发送数据。(R/W)
- RMT_TX_SIM_EN** 置位此位，多个通道开始同步发送数据。(R/W)

Register 27.18. RMT_CH_m_RX_LIM_REG ($m = 2, 3$) (0x0060, 0x0064)

(reserved)																				RMT_CH _m _RX_LIM_REG							
31																		9	8								0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x80							Reset

- RMT_RX_LIM_CH_m** 配置通道 m 最大可接收的脉冲编码数量。(R/W)

Register 27.19. RMT_DATE_REG (0x00CC)

(reserved)				RMT_RMT_DATE																								
31				28	27																							0
0	0	0	0	0x2006231																								Reset

- RMT_DATE** 版本控制寄存器。(R/W)

28 片上传感器与模拟信号处理

28.1 概述

ESP32-C3 搭载了以下片上传感器和模拟信号处理设备：

- 两个 12 位逐次逼近型模拟数字转换器 (SAR ADC)：SAR ADC1 和 SAR ADC2，共支持六个通道的模拟信号检测；
- 温度传感器：用于测量 ESP32-C3 芯片内部温度。

28.2 SAR ADC

28.2.1 概述

ESP32-C3 内置了两个 12 位的 SAR ADC，可测量最多来自六个管脚的模拟信号以及内部电压等内部信号。SAR ADC 由两个专用控制器控制：

- DIG ADC 控制器：可驱动 [Digital_Reader0](#) 和 [Digital_Reader1](#) 分别对 SAR ADC1 和 SAR ADC2 的通道电压进行采样。DIG ADC 控制器支持高性能多通道扫描和 DMA 连续转换。
- PWDET 控制器：用于 RF 功率检测。注意，此控制器仅供 RF 内部使用。

28.2.2 特性

- 两个 SAR ADC 分别有各自独立的 ADC Reader 模块 ([Digital_Reader0](#) 和 [Digital_Reader1](#))。因此两个 SAR ADC 可独立工作。
- 支持 12 位采样分辨率
- 支持采集最多六个管脚上的模拟电压
- DIG ADC 控制器：
 - 配有单次采样和多通道扫描控制模块，分别支持单次采样模式和多通道扫描模式。
 - 支持单次采样模式和多通道扫描模式同时工作。
 - 在多通道扫描模式下，支持自定义扫描通道顺序
 - 提供两个滤波器，滤波系数可配
 - 支持阈值监控，采样值大于设置的高阈值或小于设置的低阈值将产生中断
 - 支持 DMA
- PWDET 控制器：用于 RF 功率检测（仅供内部使用）

28.2.3 功能描述

SAR ADC 的主要元件与连接情况见图 [28-1](#)。

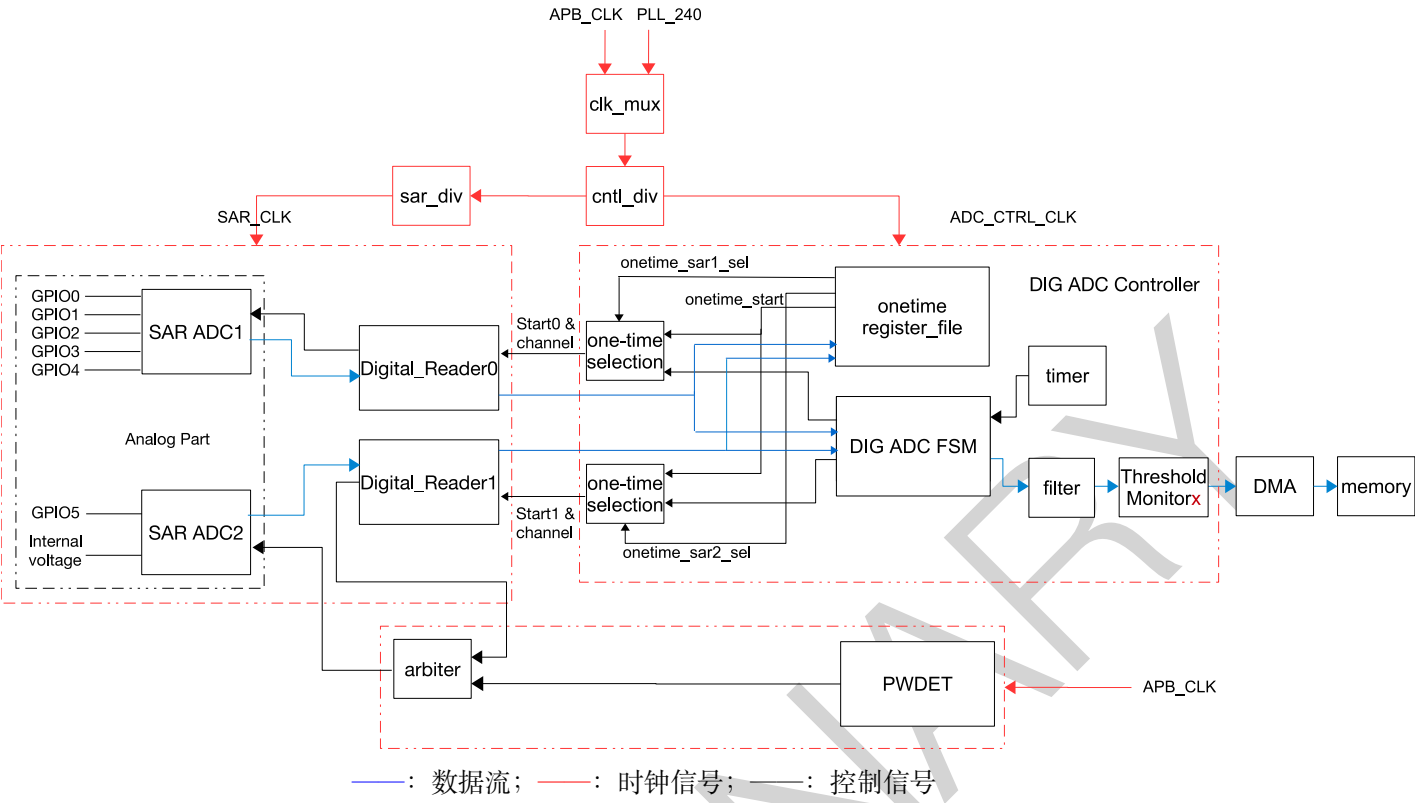


图 28-1. SAR ADC 的功能概况

如图 28-1 所示，SAR ADC 模块主要包括以下元件：

- SAR ADC1：可对五个通道进行电压检测；
- SAR ADC2：可对一个通道进行电压检测，也可对内部电压等信号进行检测；
- 时钟管理：对时钟源进行选择和分频：
 - 时钟源：可选择 APB_CLK 或 PLL_240；
 - 分频时钟：
 - * SAR_CLK：SAR ADC1、SAR ADC2、Digital_Reader0 和 Digital_Reader1 的工作时钟；其中控制 SAR_CLK 分频的 sar_div 的分频系数至少是 2 分频；
 - * ADC_CTRL_CLK：DIG ADC FSM 的工作时钟。
- 仲裁器 (arbiter)：用于选择使用 ADC2 的控制器，可以是 DIG ADC 控制器或 PWDET 控制器。
- Digital_Reader0：由 DIG ADC FSM 驱动，读取 SAR ADC1 的数值；
- Digital_Reader1：由 DIG ADC FSM 驱动，读取 SAR ADC2 的数值；
- DIG ADC FSM：生成整个 ADC 采样过程中所需的各种信号，下文简称 FSM。
- Threshold Monitor：阈值监控器 1 和阈值监控器 2。可在采样值大于设定的高阈值，或小于设定的低阈值时触发中断。

以下小节将详细介绍各个元件。

28.2.3.1 输入信号

SAR ADC 需首先通过内部多路器选择待测量的模拟管脚或内部信号，然后才能采样模拟信号。表 28-1 列出了所有可能需要经过 SAR ADC1 或 SAR ADC2 处理的模拟信号。

表 28-1. SAR ADC 的信号输入

信号名称	通道编号	ADC 选择
GPIO0	0	SAR ADC1
GPIO1	1	
GPIO2	2	
GPIO3	3	
GPIO4	4	
GPIO5	0	SAR ADC2
内部电压	n/a	

28.2.3.2 ADC 转换和衰减

SAR ADC 转换模拟信号时，转换分辨率（12 位）电压范围为 0 mV ~ V_{ref} 。其中， V_{ref} 为 SAR ADC 内部参考电压。因此，转换结果 (data) 可以使用以下公式转换成模拟电压输出 V_{data} ：

$$V_{data} = \frac{V_{ref}}{4095} \times data$$

如需转换大于 V_{ref} 的电压，信号输入 SAR ADC 前可进行衰减。衰减可配置为 0 dB、2.5 dB、6 dB 和 12 dB。

28.2.3.3 DIG ADC 控制器

DIG ADC 控制器使用快速时钟，实现了采样速率大幅提升。该控制器最高支持 12 位采样分辨率，同时支持软件触发的单次采样和专用定时器触发的多通道扫描。更多参数和性能信息见 [《ESP32-C3 系列芯片技术规格书》](#) 中的 ADC 特性章节。

软件驱动的单次采样的具体配置如下：

- 选择需要进行单次采样的 SAR ADC：
 - 置位 APB_SARADC1_ONETIME_SAMPLE 选择对 SAR ADC1 进行单次采样；
 - 置位 APB_SARADC2_ONETIME_SAMPLE 选择对 SAR ADC2 进行单次采样。
- 配置 APB_SARADC_ONETIME_CHANNEL 选择采样通道；
- 配置 APB_SARADC_ONETIME_ATTEN 选择衰减；
- 配置 APB_SARADC_ONETIME_START 启动单次采样；
- 采样结束即触发 APB_SARADC_ADC x _DONE_INT_RAW 中断。软件检测该中断后，可在 APB_SARADC_AD Cx _DATA 寄存器内读到采样值。这里的 x 可以是 1 或 2。为 1 时表示 SAR ADC1；为 2 时表示 SAR ADC2。

如果选择专用定时器驱动的多通道扫描，可采用如下配置。注，在多通道扫描模式下，扫描序列可根据样式表的描述进行，样式表可配置。

- 配置 APB_SARADC_TIMER_TARGET 设置 DIG ADC 定时器的触发周期。当计数器计数到配置周期数的 2 倍时，触发采样。计数器的工作时钟见章节 28.2.3.4；

- 配置 `APB_SARADC_TIMER_EN` 使能定时器；
- 定时器超时则将驱动 DIG ADC FSM 根据样式表进行采样；
- 数据通过 DMA 自动存储到内存空间中，扫描完成将产生中断。

说明：

单次采样不能和多通道扫描共用一个 SAR ADC。因此，当多通道扫描的样式表包含某个 SAR ADC 的采样指令时，该 SAR ADC 不能配置为单次采样。

28.2.3.4 DIG ADC 时钟

用户可配置 `APB_SARADC_CLK_SEL` 选择 DIG ADC 控制器的工作时钟：

- 1：选择 `PLL_240M` 的分频时钟 `ADC_CTRL_CLK`；
- 2：选择 `APB_CLK`。

如果选择使用 `ADC_CTRL_CLK`，用户可配置 `APB_SARADC_CLKM_DIV_NUM` 选择分频系数。注意，由于 SAR ADC 有速度限制，所以 `Digital_Reader0`（包括 SAR ADC1）和 `Digital_Reader1`（包括 SAR ADC2）的工作时钟是 `SAR_CLK`，`SAR_CLK` 频率会影响采样精度，频率越低采样精度越高。`SAR_CLK` 由 `ADC_CTRL_CLK` 经过专用分频器分频所得。分频系数通过 `APB_SARADC_SAR_CLK_DIV` 配置。ADC 每采样一个数据需要 25 个 `SAR_CLK` 时钟周期数，所以最大采样速率受到 `SAR_CLK` 的频率限制。更多时钟信息，见章节 6 [复位和时钟](#)。

28.2.3.5 DMA 支持

DIG ADC 控制器允许通过外设 DMA 实现直接内存访问，由 DIG ADC 专用定时器产生触发信号。用户可通过软件配置 `APB_SARADC_APB_ADC_TRANS` 将 DMA 的数据通路切换到 DIG ADC。关于 DMA 的具体配置，请参考章节 2 [通用 DMA 控制器 \(GDMA\)](#)。

28.2.3.6 DIG ADC FSM

概述

图 28-2 展示了 DIG ADC FSM 的工作原理。

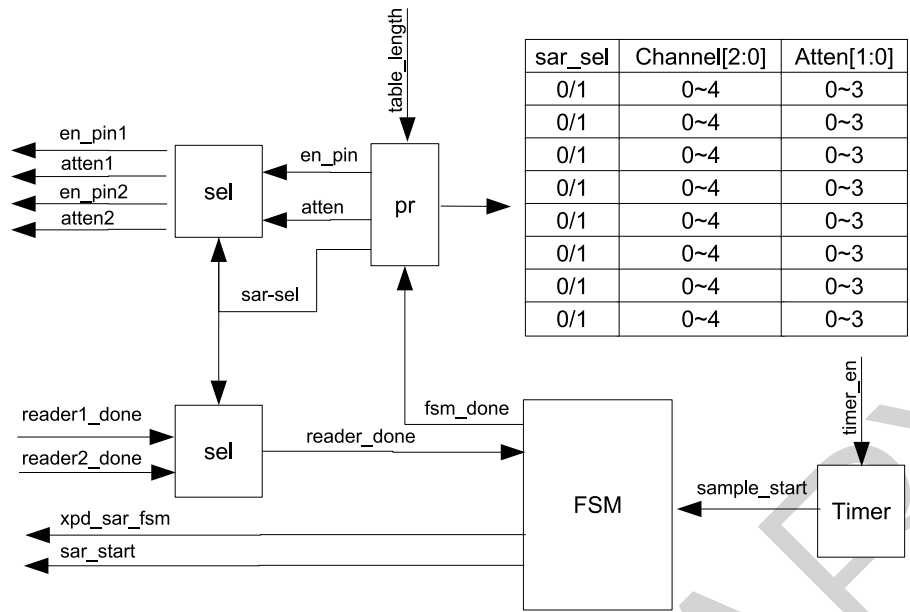


图 28-2. DIG ADC FSM 概况

其中，

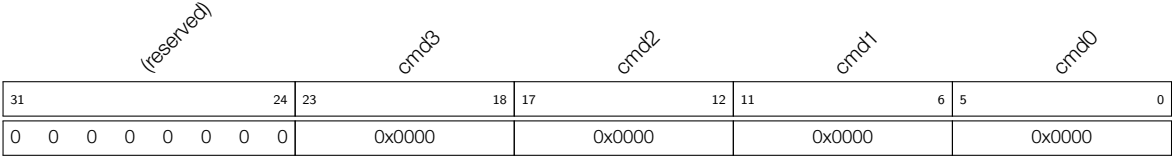
- Timer：表示 DIG ADC 的专用定时器，可产生 sample_start 信号；
- pr：样式表指针，FSM 将根据该指针指向的样式配置，发送相应信号。

相关执行过程如下：

- 置位 [APB_SARADC_TIMER_EN](#) 使能 DIG ADC 的专用定时器。定时器超时将触发 sample_start 信号驱动 FSM 模块开始采样；
- FSM 模块收到 sample_start 信号后，执行以下操作：
 - 开启 SAR ADC 电源；
 - 根据当前 pr 指向的样式，选择 SAR ADC1 或 SAR ADC2 用作工作 ADC，同时配置 ADC 通道以及衰减；
 - 根据配置信息，输出相应的 en_pad（使能管脚）以及 atten（衰减）信号到模拟端；
 - 发起 sar_start 信号，开启采样。
- FSM 收到 ADC Reader（Digital_Reader0 或 Digital_Reader1）返回的 reader_done 信号后，
 - 结束采样；
 - 将数据传输给滤波器 (filter)，然后阈值监控器 (threshold monitor) 通过 DMA 将数据传输给内存（见图 [28-1](#)）；
 - 更新样式表指针 pr，等待下一次采样。注意，如果指针 pr 小于 APB_SARADC_SAR_PATT_LEN (table_length)，则 pr = pr + 1；否则 pr 将被清零。

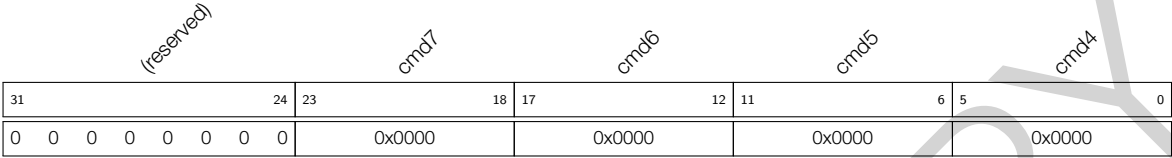
样式表结构

DIG ADC FSM 包含一个样式表，由 APB_SARADC_SAR_PATT_TAB1_REG 和 APB_SARADC_SAR_PATT_TAB2_REG 两个寄存器组成，如图 [28-3](#) 和图 [28-4](#) 所示：



cmd *x* 表示样式表中的样式，即样式 0 ~ 样式 3。

图 28-3. APB_SARADC_SAR_PATT_TAB1_REG 与样式 0 - 3



cmd *x* 表示样式表中的样式，即样式 4 ~ 样式 7。

图 28-4. APB_SARADC_SAR_PATT_TAB2_REG 与样式 4 - 7

每个寄存器包含四个样式，每个样式长度为六位，共包括三个字段，分别存储了选择的工作 ADC、通道和衰减信息，具体见表 28-5。



图 28-5. 样式表中的样式结构

atten 衰减配置信息。0: 0 dB; 1: 2.5 dB; 2: 6 dB; 3: 12 dB。

ch_sel 扫描通道选择信息，更多信息见表 28-1。

sar_sel ADC 选择信息。0: 选择 SAR ADC1; 1: 选择 SAR ADC2。

多通道扫描配置示例

例如，希望实现如下所示的多通道扫描方式：

- 扫描 SAR ADC1 的通道 2，且衰减配置为 12 dB；
- 扫描 SAR ADC2 的通道 0，且衰减配置为 2.5 dB。

则具体的配置如下：

- 配置第一个样式 cmd0，如下图所示：

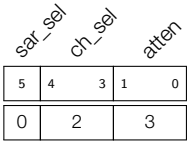


图 28-6. cmd0 配置示例

atten 配置该字段的值为 3，即衰减配置为 12 dB。

ch_sel 配置该字段的值为 2，即选择通道 2（见表 28-1）。

sar_sel 配置该位为 0，即选择 SAR ADC1。

- 配置第二个样式 cmd1，如下图所示：

sar_sel		ch_sel		atten	
5	4	3	1	0	
1	0	1			

图 28-7. cmd1 配置示例

atten 配置该字段的值为 1，即衰减配置为 2.5 dB。

ch_sel 配置该字段的值为 0，即选择通道 0（见表 28-1）。

sar_sel 配置该位为 1，即选择 SAR ADC2。

- 配置 APB_SARADC_SAR_PATT_LEN 为 1，即选择使用上述配置好的样式表 0 和样式表 1；
- 使能定时器，则 DIG ADC 控制器将根据上述样式配置，周期性采样 SAR ADC1 的通道 2 和 SAR ADC2 的通道 0。

DMA 数据格式

ADC 最终向 DMA 传递 32 位数据，如下图所示：

reserved																sar_sel																ch_sel																reserved																data																																																															
31																17																16																15																13																12																11																0															
xx																x																xxx																x																x																x																																															

图 28-8. DMA 数据格式

data ADC 转换结果，12 位

ch_sel 通道信息，3 位

sar_sel ADC 选择信息，1 位

28.2.3.7 ADC 滤波器

DIG ADC 控制器支持滤波功能，提供两个滤波器。两个滤波器均可配置给任一 SAR ADC 的任一通道，然后对目标通道的采样数据进行滤波。滤波公式如下所示：

$$data_{cur} = \frac{(k-1)data_{prev}}{k} + \frac{data_{in}}{k} - 0.5$$

- $data_{cur}$ ：滤波后数据
- $data_{in}$ ：ADC 采样值
- $data_{prev}$ ：上次滤波数据
- k ：滤波系数

配置滤波器如下：

- 配置 APB_SARADC_FILTER_CHANNEL x 设置滤波器 x 作用的 ADC 通道

- 配置 APB_SARADC_FILTER_FACTOR_x 设置滤波器 _x 的滤波系数

注意，这里的 _x 为滤波器编号：_x 为 0 表示滤波器 0；为 1 表示滤波器 1。

28.2.3.8 阈值监控

DIG ADC 包含两个阈值监控器，可配置到 SAR ADC1 和 SAR ADC2 的任意通道上。当 ADC 采样值大于设定的高阈值，则触发高阈值中断；若采样值小于设定的低阈值，则触发低阈值中断。

阈值监控配置如下：

- 配置 APB_SARADC_THRES_x_EN 使能阈值监控 _x 的功能；
- 配置 APB_SARADC_THRES_x_LOW 设置低阈值；
- 配置 APB_SARADC_THRES_x_HIGH 设置高阈值；
- 配置 APB_SARADC_THRES_x_CHANNEL 设置监控的 SAR ADC 以及通道。

注意，这里的 _x 为阈值监控器编号：_x 为 0 表示阈值监控器 0；为 1 表示阈值监控器 1。

28.2.3.9 SAR ADC2 仲裁器

SAR ADC2 可选两种控制器：DIG ADC 控制器或 PWDET 控制器。为防止出现冲突，同时提升 SAR ADC2 的使用效率，ESP32-C3 提供了 SAR ADC2 的访问仲裁。仲裁器有公平仲裁和固定优先级仲裁两种模式可供选择。

- 公平仲裁模式，即循环优先级仲裁。清零 APB_SARADC_ADC_ARB_FIX_PRIORITY 位即可进入公平仲裁模式。
- 固定优先级仲裁模式下，配置 APB_SARADC_ADC_ARB_APB_PRIORITY 位 (DIG ADC 控制器) 和 APB_SARADC_ADC_ARB_WIFI_PRIORITY 位 (PWDET 控制器) 可分别配置对应控制器的优先级。值越大，优先级越高。

仲裁器规定，无论低优先级控制器是否已开始转换数据，高优先级控制器均可随时开始自己的数据转换。如果 ADC 已经接受低优先级控制器的转换请求，开始转换数据，但高优先级控制器也需要转换数据，则可以中断或终止低优先级控制器的数据转换，开始高优先级控制器的数据转换。如果已经有高优先级控制器正在进行数据转换，而低优先级控制器则无法启动数据转换。

转换中断或转换启动失败返回的转换数据无效，因此在返回的转换结果中增加数据标记位，表示转换是否有效。

- DIG ADC 控制器的数据标记位为 DMA 数据类型的 {sar_sel, ch_sel[2:0]} 位，见图 28-8
 - 4'b1111：数据转换中断
 - 4'b1110：数据转换启动失败
 - 对应的通道号：转换数据有效
- PWDET 控制器的数据标记位为采样结果的高两位
 - 2'b10：数据转换中断
 - 2'b01：数据转换启动失败
 - 2'b00：转换数据有效

除了上述两种模式以外，用户可以通过配置 `APB_SARADC_ADC_ARB_GRANT_FORCE` 屏蔽仲裁器，配置 `APB_SARADC_ADC_ARB_WIFI_FORCE` 或 `APB_SARADC_ADC_ARB_APB_FORCE` 决定授权控制器。

28.3 温度传感器

28.3.1 概述

ESP32-C3 搭载了温度传感器可以实时监测芯片内部温度。

28.3.2 特性

温度传感器的主要特性包括：

- 支持软件触发，且一旦触发后，可持续读取数据
- 可根据使用环境配置温度偏移，提高测试精度
- 测量范围可调节

28.3.3 功能描述

温度传感器可由软件启动，具体配置如下：

- 置位 `APB_SARADC_TSENS_PU`，温度传感器上电；
- 等待 `APB_SARADC_TSENS_XPD_WAIT` 个时钟周期后，温度传感器的复位释放，开始测量环境温度；
- 等待一段时间后（输出值会随着测量时间的增加而逐渐线性逼近真实的温度值），直接从 `APB_SARADC_TSENS_OUT` 中即可持续获取温度值。

温度传感器的输出值需要使用转换公式转换成实际的温度值 (°C)。转换公式如下：

$$T(^{\circ}C) = 0.4386 * VALUE - 27.88 * offset - 20.52$$

其中 `VALUE` 即温度传感器的输出值，`offset` 由温度偏移 `TSENS_DAC` 决定。用户可根据实际使用环境（测量温度范围），结合表 28-2 通过 I2C 寄存器 `I2C_SARADC_TSENS_ADC` 配置 `TSENS_DAC`。

表 28-2. 温度传感器的温度偏移

TSENS_DAC	温度偏移 (°C)	测量范围 (°C)
5	-2	50 ~ 125
13 或 7	-1	20 ~ 100
15	0	-10 ~ 80
11 或 14	1	-30 ~ 50
10	2	-40 ~ 20

28.4 中断

- `APB_SARADC_ADC1_DONE_INT`：SAR ADC1 完成一次转换，即触发此中断；
- `APB_SARADC_ADC2_DONE_INT`：SAR ADC2 完成一次转换，即触发此中断；
- `APB_SARADC_THRESx_HIGH_INT`：超过阈值监控器 `x` 的高阈值，即触发此中断；

- APB_SARADC_THRES_x_LOW_INT: 超过阈值监控器 _x 的低阈值, 即触发此中断。

28.5 寄存器列表

本小节的所有地址均为相对于 ADC 控制器基地址的地址偏移量 (相对地址), 具体基地址请见章节 3 系统和存储器 中的表 3-4。

名称	描述	地址	访问
配置寄存器			
APB_SARADC_CTRL_REG	SAR ADC 控制寄存器 1	0x0000	R/W
APB_SARADC_CTRL2_REG	SAR ADC 控制寄存器 2	0x0004	R/W
APB_SARADC_FILTER_CTRL1_REG	滤波控制寄存器 1	0x0008	R/W
APB_SARADC_SAR_PATT_TAB1_REG	样式表寄存器 1	0x0018	R/W
APB_SARADC_SAR_PATT_TAB2_REG	样式表寄存器 2	0x001C	R/W
APB_SARADC_ONETIME_SAMPLE_REG	单次采样配置寄存器	0x0020	R/W
APB_SARADC_APB_ADC_ARB_CTRL_REG	SAR ADC2 仲裁器配置寄存器	0x0024	R/W
APB_SARADC_FILTER_CTRL0_REG	滤波控制寄存器 0	0x0028	R/W
APB_SARADC_1_DATA_STATUS_REG	SAR ADC1 采样数据寄存器	0x002C	RO
APB_SARADC_2_DATA_STATUS_REG	SAR ADC2 采样数据寄存器	0x0030	RO
APB_SARADC_THRES0_CTRL_REG	采样阈值控制寄存器 0	0x0034	R/W
APB_SARADC_THRES1_CTRL_REG	采样阈值控制寄存器 1	0x0038	R/W
APB_SARADC_THRES_CTRL_REG	采样阈值控制寄存器	0x003C	R/W
APB_SARADC_INT_ENA_REG	SAR ADC 中断使能寄存器	0x0040	R/W
APB_SARADC_INT_RAW_REG	SAR ADC 原始中断寄存器	0x0044	RO
APB_SARADC_INT_ST_REG	SAR ADC 中断状态寄存器	0x0048	RO
APB_SARADC_INT_CLR_REG	SAR ADC 中断清除寄存器	0x004C	WO
APB_SARADC_DMA_CONF_REG	SAR ADC DMA 配置寄存器	0x0050	R/W
APB_SARADC_APB_ADC_CLKM_CONF_REG	SAR ADC 时钟控制寄存器	0x0054	R/W
APB_SARADC_APB_TSENS_CTRL_REG	温度传感器控制寄存器 1	0x0058	varies
APB_SARADC_APB_TSENS_CTRL2_REG	温度传感器控制寄存器 2	0x005C	R/W
APB_SARADC_CALI_REG	SAR ADC 校准寄存器	0x0060	R/W
APB_SARADC_APB_CTRL_DATE_REG	版本控制寄存器	0x03FC	R/W

28.6 寄存器

本小节的所有地址均为相对于 ADC 控制器基地址的地址偏移量 (相对地址), 具体基地址请见章节 3 系统和存储器 中的表 3-4。

Register 28.1. APB_SARADC_CTRL_REG (0x0000)

APB_SARADC_WAIT_ARB_CYCLE		(reserved)		APB_SARADC_XPD_SAR_FORCE		(reserved)		APB_SARADC_SAR_PATT_P_CLEAR		(reserved)		APB_SARADC_SAR_PATT_LEN		(reserved)		APB_SARADC_SAR_CLK_DIV		(reserved)		APB_SARADC_SAR_CLK_GATED		(reserved)		APB_SARADC_START		APB_SARADC_START_FORCE					
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
1	0	0	0	0	0	0	0	0	0	0	0	0	0	7	4	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
																															Reset

- APB_SARADC_START_FORCE** 0: 选择使用 FSM 启动 SAR ADC 采样; 1: 选择使用软件启动 SAR ADC 采样。(R/W)
- APB_SARADC_START** 写 1 选择使用软件启动 SAR ADC。仅当 **APB_SARADC_START_FORCE** = 1 时有效。(R/W)
- APB_SARADC_SAR_CLK_GATED** SAR ADC 的时钟门控使能位。(R/W)
- APB_SARADC_SAR_CLK_DIV** SAR ADC 的时钟分频系数。(R/W)
- APB_SARADC_SAR_PATT_LEN** 配置需要使用的样式数量。0~7 分别代表样式表的第 0~7 个样式。(R/W)
- APB_SARADC_SAR_PATT_P_CLEAR** 清除 DIG ADC1 控制器样式表指针。(R/W)
- APB_SARADC_XPD_SAR_FORCE** 强制选择 XPD SAR。(R/W)
- APB_SARADC_WAIT_ARB_CYCLE** SAR_DONE 信号发出后至仲裁信号稳定需等待的时钟周期。(R/W)

Register 28.2. APB_SARADC_CTRL2_REG (0x0004)

(reserved)								APB_SARADC_TIMER_EN																APB_SARADC_TIMER_TARGET																(reserved)								APB_SARADC_SAR2_INV																APB_SARADC_SAR1_INV																APB_SARADC_MAX_MEAS_NUM																APB_SARADC_MEAS_NUM_LIMIT															
31								25								24	23													12				11	10	9	8													1	0																																																												
0								0								0	0	10										0				0	0	255												0	Reset																																																																

- APB_SARADC_MEAS_NUM_LIMIT** 使能 SAR ADC 最大转换次数限制。(R/W)
- APB_SARADC_MAX_MEAS_NUM** 设置最大转换次数。(R/W)
- APB_SARADC_SAR1_INV** 写 1 反转 SAR ADC1 数据。(R/W)
- APB_SARADC_SAR2_INV** 写 1 反转 SAR ADC2 数据。(R/W)
- APB_SARADC_TIMER_TARGET** 设置 SAR ADC 定时器目标，即定时器的触发周期。(R/W)
- APB_SARADC_TIMER_EN** 使能 SAR ADC 定时器触发。(R/W)

Register 28.3. APB_SARADC_FILTER_CTRL1_REG (0x0008)

APB_SARADC_FILTER_FACTOR0																												APB_SARADC_FILTER_FACTOR1																												(reserved)																											
31	29	28	26	25																								0																																																							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Reset																																																					

- APB_SARADC_FILTER_FACTOR1** 设置 SAR ADC 滤波器 1 的滤波系数。(R/W)
- APB_SARADC_FILTER_FACTOR0** 设置 SAR ADC 滤波器 0 的滤波系数。(R/W)

Register 28.4. APB_SARADC_SAR_PATT_TAB1_REG (0x0018)

(reserved)								APB_SARADC_SAR_PATT_TAB1																									
31								24	23																								0
0	0	0	0	0	0	0	0	0x0000																								Reset	

APB_SARADC_SAR_PATT_TAB1 样式表的第 0~3 个样式。每个样式占六位。(R/W)

Register 28.5. APB_SARADC_SAR_PATT_TAB2_REG (0x001C)

(reserved)								APB_SARADC_SAR_PATT_TAB2																							0		
31								24	23																								0
0	0	0	0	0	0	0	0	0x0000																							Reset		

APB_SARADC_SAR_PATT_TAB2 样式表的第 4~7 个样式，每个样式占六位。(R/W)

Register 28.6. APB_SARADC_ONETIME_SAMPLE_REG (0x0020)

APB_SARADC1_ONETIME_SAMPLE				APB_SARADC2_ONETIME_SAMPLE				APB_SARADC_ONETIME_START				APB_SARADC_ONETIME_CHANNEL				(reserved)																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
31	30	29	28					25	24	23	22																										0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0	0	0	13				0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0</

APB_SARADC_ONETIME_ATTEN 配置单次采样的衰减。(R/W)

APB_SARADC_ONETIME_CHANNEL 配置单次采样的通道。(R/W)

APB_SARADC_ONETIME_START 启动 SAR ADC 单次采样。(R/W)

APB_SARADC2_ONETIME_SAMPLE 使能 SAR ADC2 单次采样。(R/W)

APB_SARADC1_ONETIME_SAMPLE 使能 SAR ADC1 单次采样。(R/W)

Register 28.9. APB_SARADC_1_DATA_STATUS_REG (0x002C)

(reserved)																APB_SARADC_APB_SARADC1_DATA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																17																16																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

APB_SARADC_ADC1_DATA SAR ADC1 的转换数据。(RO)

Register 28.10. APB_SARADC_2_DATA_STATUS_REG (0x0030)

(reserved)																APB_SARADC_ADC2_DATA																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
31																17																16																0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0																0															

APB_SARADC_ADC2_DATA SAR ADC2 的转换数据。(RO)

Register 28.11. APB_SARADC_THRES0_CTRL_REG (0x0034)

(reserved)																APB_SARADC_THRES0_LOW																																APB_SARADC_THRES0_HIGH																(reserved)																APB_SARADC_THRES0_HIGH															
31	30															18	17													5	4	3																																																															
0	0														0x1fff														0	13												Reset																																																					

APB_SARADC_THRES0_CHANNEL 配置 SAR ADC 阈值监控器 0 需要监控的通道。(R/W)

APB_SARADC_THRES0_HIGH 配置 SAR ADC 阈值监控器 0 的高阈值。(R/W)

APB_SARADC_THRES0_LOW 配置 SAR ADC 阈值监控器 0 的低阈值。(R/W)

Register 28.14. APB_SARADC_INT_ENA_REG (0x0040)

Bit	Field	Reset
31	APB_SARADC_ADC1_DONE_INT_ENA	0
30	APB_SARADC_ADC2_DONE_INT_ENA	0
29	APB_SARADC_THRES0_HIGH_INT_ENA	0
28	APB_SARADC_THRES0_LOW_INT_ENA	0
27	APB_SARADC_THRES1_HIGH_INT_ENA	0
26	APB_SARADC_THRES1_LOW_INT_ENA	0
25-0	(reserved)	0

APB_SARADC_THRES1_LOW_INT_ENA APB_SARADC_THRES1_LOW_INT 的中断使能位。(R/W)

APB_SARADC_THRES0_LOW_INT_ENA **APB_SARADC_THRES0_LOW_INT** 的中断使能位。(R/W)

APB_SARADC_THRES1_HIGH_INT_ENA **APB_SARADC_THRES1_HIGH_INT** 的中断使能位。(R/W)

APB_SARADC_THRES0_HIGH_INT_ENA **APB_SARADC_THRES0_HIGH_INT** 的中断使能位。(R/W)

APB_SARADC_ADC2_DONE_INT_ENA **APB_SARADC_ADC2_DONE_INT** 的中断使能位。(R/W)

APB_SARADC_ADC1_DONE_INT_ENA **APB_SARADC_ADC1_DONE_INT** 的中断使能位。(R/W)

APB_SARADC_ADC1_DONE_INT_RAW
APB_SARADC_ADC2_DONE_INT_RAW
APB_SARADC_THRES0_HIGH_INT_RAW
APB_SARADC_THRES0_LOW_INT_RAW
APB_SARADC_THRES1_HIGH_INT_RAW
APB_SARADC_THRES1_LOW_INT_RAW

APB_SARADC_THRES1_LOW_INT_RAW APB_SARADC_THRES1_LOW_INT 的原始中断位。(RO)

APB_SARADC_THRES0_LOW_INT_RAW APB_SARADC_THRES0_LOW_INT 的原始中断位。(RO)

APB_SARADC_THRES1_HIGH_INT_RAW APB_SARADC_THRES1_HIGH_INT 的原始中断位。
(RO)

APB_SARADC_THRES0_HIGH_INT_RAW APB_SARADC_THRES0_HIGH_INT 的原始中断位。
(RO)

APB_SARADC_ADC2_DONE_INT_RAW APB_SARADC_ADC2_DONE_INT 的原始中断位。(RO)

APB_SARADC_ADC1_DONE_INT_RAW APB_SARADC_ADC1_DONE_INT 的原始中断位。(RO)

Register 28.16. APB_SARADC_INT_ST_REG (0x0048)

[illegible]

APB_SARADC_THRES1_LOW_INT_ST APB_SARADC_THRES1_LOW_INT 的中断状态位。(RO)

APB_SARADC_THRES0_LOW_INT_ST **APB_SARADC_THRES0_LOW_INT** 的中断状态位。(RO)

APB_SARADC_THRES1_HIGH_INT_ST **APB_SARADC_THRES1_HIGH_INT** 的中断状态位。(RO)

APB_SARADC_THRES0_HIGH_INT_ST **APB_SARADC_THRES0_HIGH_INT** 的中断状态位。(RO)

APB_SARADC_ADC2_DONE_INT_ST APB_SARADC_ADC2_DONE_INT 的中断状态位。(RO)

APB_SARADC_ADC1_DONE_INT_ST APB_SARADC_ADC1_DONE_INT 的中断状态位。(RO)

Register 28.19. APB_SARADC_APB_ADC_CLKM_CONF_REG (0x0054)

(reserved)										APB_SARADC_CLK_SEL				APB_SARADC_CLK_EN				APB_SARADC_CLKM_DIV_A				APB_SARADC_CLKM_DIV_B				APB_SARADC_CLKM_DIV_NUM			
31								23	22	21	20	19	14			13	8			7				0					
0	0	0	0	0	0	0	0	0	0	0	0	0x0			0x0			4						Reset					

APB_SARADC_CLKM_DIV_NUM ADC 时钟的整数分频系数。分频系数 = $\text{APB_SARADC_CLKM_DIV_NUM} + \text{APB_SARADC_CLKM_DIV_B}/\text{APB_SARADC_CLKM_DIV_A}$ 。
(R/W)

APB_SARADC_CLKM_DIV_B 小数分频系数中的分子。(R/W)

APB_SARADC_CLKM_DIV_A 小数分频系数中的分母。(R/W)

APB_SARADC_CLK_EN 使能 SAR ADC 寄存器时钟。(R/W)

APB_SARADC_CLK_SEL 置位此位，使能 PLL_240。(R/W)

Register 28.20. APB_SARADC_APB_TSENS_CTRL_REG (0x0058)

(reserved)								APB_SARADC_TSENS_PU				APB_SARADC_TSENS_CLK_DIV				APB_SARADC_TSENS_IN_INV				(reserved)				APB_SARADC_TSENS_OUT				
31								23	22	21					14	13	12					8	7					0
0	0	0	0	0	0	0	0	0	0	0	6				0	0	0	0	0	0	0x0				Reset			

APB_SARADC_TSENS_OUT 温度传感器的输出值。(RO)

APB_SARADC_TSENS_IN_INV 反转温度传感器的输入值。(R/W)

APB_SARADC_TSENS_CLK_DIV 温度传感器的时钟分频系数。(R/W)

APB_SARADC_TSENS_PU 温度传感器上电。(R/W)

Register 28.21. APB_SARADC_APB_TSENS_CTRL2_REG (0x005C)

(reserved)																APB_SARADC_TSENS_CLK_SEL						(reserved)										APB_SARADC_TSENS_XPD_WAIT					
31															16	15	13	12	10														0				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2						Reset												

APB_SARADC_TSENS_XPD_WAIT 温度传感器的复位释放前，需要等待的时间。(R/W)

APB_SARADC_TSENS_CLK_SEL 选择温度传感器时钟。0: FOSC_CLK; 1: XTAL_CLK。(R/W)

Register 28.22. APB_SARADC_CALI_REG (0x0060)

(reserved)																APB_SARADC_CALL_CFG															
31															17	16															0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x8000														Reset	

APB_SARADC_CALI_CFG 配置 SAR ADC 校准系数。(R/W)

Register 28.23. APB_SARADC_APB_CTRL_DATE_REG (0x03FC)

APB_SARADC_DATE																															
310																															
0x2007171Reset																															

APB_SARADC_DATE 版本控制寄存器。(R/W)

29 相关文档和资源

相关文档

- [《ESP32-C3 技术规格书》](#) – 提供 ESP32-C3 芯片的硬件技术规格。
- 证书
<http://espressif.com/zh-hans/support/documents/certificates>
- 文档更新和订阅通知
<http://espressif.com/zh-hans/support/download/documents>

开发者社区

- [《ESP32-C3 ESP-IDF 编程指南》](#) – ESP-IDF 开发框架的文档中心。
- ESP-IDF 及 GitHub 上的其它开发框架
<http://github.com/espressif>
- ESP32 论坛 – 工程师对工程师 (E2E) 的社区，您可以在这里提出问题、解决问题、分享知识、探索观点。
<http://esp32.com/>
- *The ESP Journal* – 分享乐鑫工程师的最佳实践、技术文章和工作随笔。
<http://blog.espressif.com/>
- SDK 和演示、App、工具、AT 等下载资源
<http://espressif.com/zh-hans/support/download/sdks-demos>

产品

- ESP32-C3 系列芯片 – ESP32-C3 全系列芯片。
<http://espressif.com/zh-hans/products/socs?id=ESP32-C3>
- ESP32-C3 系列模组 – ESP32-C3 全系列模组。
<http://espressif.com/zh-hans/products/modules?id=ESP32-C3>
- ESP32-C3 系列开发板 – ESP32-C3 全系列开发板。
<http://espressif.com/zh-hans/products/devkits?id=ESP32-C3>
- ESP Product Selector (乐鑫产品选型工具) – 通过筛选性能参数、进行产品对比快速定位您所需要的产品。
<http://products.espressif.com/#/product-selector?language=zh>

联系我们

- 商务问题、技术支持、电路原理图 & PCB 设计审阅、购买样品 (线上商店)、成为供应商、意见与建议
<http://espressif.com/zh-hans/contact-us/sales-questions>

词汇列表

外设相关词汇

AES	AES 加速器
BOOTCTRL	芯片 Boot 控制
DS	数字签名
DMA	DMA 控制器
eFuse	eFuse 控制器
HMAC	HMAC 加速器
I2C	I2C 控制器
I2S	I2S 控制器
LEDC	LED 控制 PWM
MCPWM	电机控制 PWM
PCNT	脉冲计数器控制器
RMT	红外遥控
RNG	随机数生成器
RSA	RSA 加速器
SDHOST	SD/MMC 主机控制器
SHA	SHA 加速器
SPI	SPI 控制器
SYSTIMER	系统定时器
TIMG	定时器组
TWAI	双线汽车接口
UART	UART 控制器
ULP 协处理器	超低功耗协处理器
USB OTG	USB On-The-Go
WDT	看门狗定时器

寄存器相关词汇

ISO	隔离。当模块断电时，其输出的引脚将处于未知状态（某些中间电压）。“ISO”寄存器将使其输出引脚隔离在一个确定的电压，从而不会影响其他未掉电的工作模块的状态。
NMI	不可屏蔽中断。
REG	寄存器。
R/W	读/写，软件可读写这些位。
RO	只读，软件只能读这些位。
SYSREG	系统寄存器。
WO	只写，软件只能写这些位。

修订历史

日期	版本	发布说明
2021-10-28	v0.4	<p>新增以下章节：</p> <ul style="list-style-type: none"> • 章节 8 中断矩阵 (INTMATRIX) • 章节 14 辅助调试 (Debug Assist) • 章节 23 I2C 控制器 (I2C) • 章节 28 片上传感器与模拟信号处理 • 章节 29 相关文档和资源 <p>更新以下章节：</p> <ul style="list-style-type: none"> • 章节 4 eFuse 控制器 (EFUSE) • 章节 27 红外遥控 (RMT)
2021-08-05	v0.3	<p>新增以下章节：</p> <ul style="list-style-type: none"> • 章节 9 系统定时器 (SYSTIMER) • 章节 11 看门狗定时器 (WDT) • 章节 12 XTAL32K 看门狗定时器 (XTWDT) • 章节 13 系统寄存器 (SYSREG) • 章节 18 HMAC 加速器 (HMAC) • 章节 19 数字签名 (DS) • 章节 24 USB 串口/JTAG 控制器 (USB_SERIAL_JTAG) • 章节 27 红外遥控 (RMT) <p>更新以下章节：</p> <ul style="list-style-type: none"> • 章节 4 eFuse 控制器 (EFUSE) • 章节 5 IO MUX 和 GPIO 交换矩阵 (GPIO, IO MUX) • 章节 7 芯片 Boot 控制 • 章节 25 双线汽车接口 (TWAI)
2021-05-27	v0.2	<p>新增以下章节：</p> <ul style="list-style-type: none"> • 章节 2 通用 DMA 控制器 (GDMA) • 章节 4 eFuse 控制器 (EFUSE) • 章节 10 定时器组 (TIMG) • 章节 22 UART 控制器 (UART) • 章节 26 LED PWM 控制器 (LEDC) <p>更新章节 5 IO MUX 和 GPIO 交换矩阵 (GPIO, IO MUX)</p> <p>调整章节顺序</p>
2021-04-07	v0.1	首次发布



www.espressif.com

免责声明和版权公告

本文档中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

本文档可能引用了第三方的信息，所有引用的信息均为“按现状”提供，乐鑫不对信息的准确性、真实性做任何保证。

乐鑫不对本文档的内容做任何保证，包括内容的适销性、是否适用于特定用途，也不提供任何其他乐鑫提案、规格书或样品在他处提到的任何保证。

乐鑫不对本文档是否侵犯第三方权利做任何保证，也不对使用本文档内信息导致的任何侵犯知识产权的行为负责。本文档在此未以禁止反言或其他方式授予任何知识产权许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。

文档中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

版权归 © 2021 乐鑫信息科技（上海）股份有限公司。保留所有权利。