请注意赛普拉斯已正式并入英飞凌科技公司。
此封面页之后的文件标注有“赛普拉斯”的文件即该产品为此公司最初开发的。请注意作为英飞凌产品组合的部分，英飞凌将继续为新的及现有客户提供该产品。

文件内容的连续性
事实是英飞凌提供如下产品作为英飞凌产品组合的部分不会带来对于此文件的任何变更。未来的变更将在恰当的时候发生，且任何变更将在历史页面记录。

订购零件编号的连续性
英飞凌继续支持现有零件编号的使用。下单时请继续使用数据表中的订购零件编号。

PSoC 41XX_BLE/42XX_BLE 系列

PSoC® 4 BLE 架构
技术参考手册 （TRM）

文档编号：001-96125 版本 *A

August 16, 2019

赛普拉斯半导体公司
198 Champion Court
San Jose, CA 95134-1709

电话 （美国）：800.858.1810
电话 （国际）：408.943.2600
www.cypress.com

版权所有

© 赛普拉斯半导体公司，2014-2019 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“ **赛普拉斯** ”）的财产。本文件，包括其包含或引用的任何软件或固件（“ **软件** ”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可 （无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和 （2）在被软件 （由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规 格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统 （包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“ **非预期用途** ”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。

目录概要

章节 A: 概述 **17**

1. 简介 19
2. 入门 29
3. 文档结构 31

章节 B: CPU 系统 **35**

4. Cortex-M0 CPU 37
5. DMA 控制器模式 43
6. 中断 57

章节 C: 系统资源的子系统 （SRSS） **67**

7. I/O 系统 69
8. 时钟系统 83
9. 电源供应和监控 95
10. 芯片运行模式 99
11. 功耗模式 101
12. 看门狗定时器 107
13. 复位系统 111
14. 器件安全性 115

章节 D: 数字系统 **117**

15. 串行通信模块 （SCB） 119
16. 通用数字模块 （UDB） 161
17. 定时器、计数器和 PWM 203
18. 蓝牙低功耗子系统 （BLESS） 227

章节 E: 模拟系统 **245**

18. 高精度参考 247
19. SAR ADC 251
20. 低功耗比较器 283
21. 微型连续时间模块 （CTBm） 289
22. LCD 直接驱动 299
23. CapSense 311
24. 温度传感器 321

章节 F: 编程和调试 325
26. 编程与调试接口 327
27. 非易失性存储器编程 335

术语表 349
索引 362

目录

章节 A: 概述 17

文档修订记录 17

1. 简介 19

1.1 系统架构 19
1.2 特性 25
1.3 CPU 系统 25
1.3.1 处理器 25
1.3.2 中断控制器 26
1.3.3 直接存储器访问 26
1.4 存储器 26
1.4.1 闪存 26
1.4.2 SRAM 26
1.5 系统范围资源 26
1.5.1 时钟系统 26
1.5.2 电源系统 26
1.5.3 GPIO 26
1.6 蓝牙低功耗子系统 26
1.6.1 射频收发器 26
1.6.2 数字 PHY 调制解调器 26
1.6.3 链路层控制器 27
1.7 可编程数字模块 27
1.8 固定功能数字模块 27
1.8.1 定时器 / 计数器 /PWM 模块 27
1.8.2 串行通信模块 27
1.9 模拟系统 27
1.9.1 SAR ADC 27
1.9.2 微型连续时间模块 27
1.9.3 低功耗比较器 27
1.10 特殊功能的外设 27
1.10.1 LCD Segment 驱动 27
1.10.2 CapSense 27
1.10.2.1 IDAC 和比较器 27
1.11 编程和调试 27
1.12 器件特性总结 28

2. 入门 29

2.1 支持服务 29
2.2 产品升级 29
2.3 开发套件 29
2.4 应用笔记 29

3. 文档结构 31

3.1 主要部分 31
3.2 文档规范 31
3.2.1 寄存器约定 31
3.2.2 数字命名 31
3.2.3 测量单位 32
3.2.4 缩略语 32

章节 B: CPU 系统 35

系统架构 35

4. Cortex-M0 CPU 37

4.1 特性 37
4.2 框图 38
4.3 工作原理 38
4.4 地址映射 38
4.5 寄存器 39
4.6 工作模式 40
4.7 指令集 40
4.7.1 地址对齐 41
4.7.2 存储器中的字节顺序 41
4.8 Systick 定时器 41
4.9 调试 41

5. DMA 控制器模式 43

5.1 框图说明 43
5.1.1 触发源和复用 44
5.1.1.1 触发复用器 44
5.1.1.2 创建软件触发 46
5.1.2 挂起触发 46
5.1.3 输出触发 46
5.1.4 通道优先级 46
5.1.5 数据传输引擎 46
5.2 描述符 47
5.2.1 地址配置 47
5.2.2 传输大小 48
5.2.3 描述符链接 49
5.2.4 传输模式 49
5.2.4.1 每个触发器一个单数据元素 （操作码 0） 49
5.2.4.2 每次触发传输整个描述符 （操作码 1） 51
5.2.4.3 每个触发传输整个描述符链接 （操作码 2） 52
5.3 操作与时序 53
5.4 仲裁 54
5.5 寄存器列表 56

6. 中断 57

6.1 特性 57
6.2 工作原理 57
6.3 中断 / 异常事件及其操作 58
6.3.1 中断 / 异常事件处理 58
6.3.2 电平与边沿触发中断 58
6.3.3 异常事件向量表 59
6.4 异常事件源 59

6.4.1 复位异常事件 59
6.4.2 不可屏蔽中断 （NMI）异常事件 60
6.4.3 HardFault （硬故障）异常事件 60
6.4.4 管理程序调用 （SVCall）异常事件 60
6.4.5 PendSV 异常事件 60
6.4.6 SysTick 异常事件 60
6.5 中断源 60
6.6 异常事件优先级 62
6.7 使能和禁用中断 62
6.8 异常事件的状态 63
6.8.1 挂起异常事件 63
6.9 异常事件的堆栈使用情况 63
6.10 中断和低功耗模式 64
6.11 异常事件 — 初始化与配置 65
6.12 寄存器 65
6.13 相关文档 65

章节 C: 系统资源的子系统 （SRSS） 67

顶层架构 67

7. I/O 系统 69

7.1 特性 69
7.2 GPIO 接口概况 69
7.3 I/O 单元架构 70
7.3.1 数字输入缓冲区 72
7.3.2 数字输出驱动器 72
7.3.2.1 驱动模式 72
7.3.2.2 转换速率控制 74
7.4 GPIO-OVT 引脚 74
7.5 高速 I/O 矩阵 77
7.6 上电时的 I/O 状态 77
7.7 低功耗模式下的行为 78
7.8 输入和输出的同步 78
7.9 中断 78
7.10 外设连接 80
7.10.1 固件控制的 GPIO 80
7.10.2 模拟 I/O 80
7.10.2.1 AMUXBUS 连接和 DSI 80
7.10.3 LCD 驱动 80
7.10.4 CapSense 80
7.10.5 蓝牙低功耗子系统 （BLESS） 80
7.10.6 串行通信模块 （SCB） 81
7.11 端口限制 81
7.12 寄存器 81

8. 时钟系统 83

8.1 框图 83
8.2 时钟源 84
8.2.1 内部主振荡器 84
8.2.1.1 启动行为 85
8.2.1.2 IMO 频率扩展 86
8.2.1.3 编程时钟 （36 MHz） 86
8.2.2 内部低速振荡器 86

8.2.3 外部时钟 （EXTCLK）.....86
8.2.4 外部晶振 （ECO）.....87
8.2.4.1 ECO 负载电容调试.....87
8.2.5 监视晶体振荡器 （WCO）.....87
8.3 时钟分布.....88
8.3.1 HFCLK 输入选择.....88
8.3.2 LFCLK 输入选择.....88
8.3.3 ECO 分频器配置.....88
8.3.4 SYSCLK 预分频器配置.....89
8.3.5 外设时钟分频器配置.....89
8.3.6 外设时钟配置.....91
8.3.6.1 时钟生成.....91
8.4 低功耗模式操作.....92
8.5 寄存器列表.....93

9. 电源供应和监控 95

9.1 框图.....95
9.2 工作原理.....96
9.2.1 电压调节器汇总.....96
9.2.1.1 内核电压调节器.....96
9.2.1.2 RF 收发器电压调节器.....96
9.3 电压监控.....97
9.3.1 上电复位 （POR）.....97
9.3.1.1 欠压检测 （BOD）.....97
9.3.1.2 低压检测 （LVD）.....97
9.4 寄存器列表.....98

10. 芯片运行模式 99

10.1 启动模式.....99
10.2 用户模式.....99
10.3 特权模式.....99
10.4 调试模式.....99

11. 功耗模式 101

11.1 活动模式.....102
11.2 睡眠模式.....102
11.3 深度睡眠模式.....102
11.4 休眠模式.....103
11.5 停止模式.....103
11.6 功耗模式总结.....104
11.7 低功耗模式进入和退出.....105
11.8 寄存器列表.....106

12. 看门狗定时器 107

12.1 特性.....107
12.2 框图.....107
12.3 工作原理.....108
12.3.1 使能和禁用 WDT.....109
12.3.2 WDT 操作模式.....109
12.3.3 WDT 中断和低功耗模式.....110
12.3.4 WDT 复位模式.....110
12.4 寄存器列表.....110

13. 复位系统 111

13.1 复位源 111
13.1.1 上电复位 111
13.1.2 欠压复位 111
13.1.3 看门狗复位 111
13.1.4 软件复位 112
13.1.5 外部复位 112
13.1.6 保护故障复位 112
13.1.7 休眠模式唤醒复位 112
13.1.8 停止模式唤醒复位 112
13.2 识别复位源 112
13.3 寄存器列表 113

14. 器件安全性 115

14.1 特性 115
14.2 工作原理 115
14.2.1 器件安全性 115
14.2.2 闪存安全性 116

章节 D: 数字系统 117

系统架构 117

15. 串行通信模块 （SCB） 119

15.1 特性 119
15.2 串行外设接口 （SPI） 119
15.2.1 特性 119
15.2.2 概述 120
15.2.3 SPI 操作模式 121
15.2.3.1 Motorola SPI 121
15.2.3.2 Texas Instruments SPI 122
15.2.3.3 National Semiconductors SPI 124
15.2.4 通过使用 SPI 主设备为从设备提供时钟脉冲 125
15.2.5 Easy SPI 协议 125
15.2.5.1 EZ 地址的写操作 125
15.2.5.2 存储器阵列的写操作 125
15.2.5.3 存储器阵列的读操作 125
15.2.5.4 将 SCB 配置为 EZSPI 模式 126
15.2.6 SPI 寄存器 127
15.2.7 SPI 中断 128
15.2.8 使能和初始化 SPI 128
15.2.9 内部和外部时钟的 SPI 操作 130
15.2.9.1 非 EZ 工作模式 131
15.2.9.2 EZ 工作模式 131
15.3 UART 133
15.3.1 特性 133
15.3.2 概述 133
15.3.3 UART 工作模式 133
15.3.3.1 标准协议 133
15.3.3.2 SmartCard （ISO7816） 138
15.3.3.3 IrDA 139
15.3.4 UART 寄存器 140
15.3.5 UART 中断 141
15.3.6 使能和启动 UART 141

15.4 内部集成电路 （I2C） 143
15.4.1 特性 143
15.4.2 概述 143
15.4.3 术语和定义 144
15.4.3.1 时钟延长 144
15.4.3.2 总线仲裁 144
15.4.4 I2C 工作模式 144
15.4.4.1 写传输 145
15.4.4.2 读传输 145
15.4.5 Easy I2C （EZI2C）协议 146
15.4.5.1 存储器阵列的写操作 146
15.4.5.2 存储器阵列的读操作 146
15.4.6 I2C 寄存器 147
15.4.7 I2C 中断 148
15.4.8 使能和初始化 I2C 148
15.4.8.1 **将 I2C 接口配置为标准 （非 EZ）模式 148**
15.4.8.2 配置 EZI2C 模式 149
15.4.9 I2C 中的内部和外部时钟操作 149
15.4.9.1 I2C 非 EZ 工作模式 150
15.4.9.2 I2C EZ 工作模式 150
15.4.10 从睡眠模式中唤醒 151
15.4.11 主设备模式的传输示例 152
15.4.11.1 主设备发送 152
15.4.11.2 主设备接收 153
15.4.12 从设备模式的传输示例 154
15.4.12.1 从设备发送 154
15.4.12.2 从设备接收 155
15.4.13 EZ 从设备模式的传输示例 156
15.4.13.1 EZ 从设备发送 156
15.4.13.2 EZ 从设备接收 157
15.4.14 多主设备模式的传输示例 158
15.4.14.1 多主设备 - 从设备不被使能 158
15.4.14.2 多主设备 - 从设备被使能 159

16. 通用数字模块 （UDB） 161

16.1 特性 161
16.2 工作原理 162
16.2.1 PLD 162
16.2.1.1 PLD 宏单元 163
16.2.1.2 PLD 进位链路 164
16.2.1.3 PLD 配置 164
16.2.2 数据路径 164
16.2.2.1 概述 165
16.2.2.2 数据路径 FIFO 166
16.2.2.3 FIFO 状态 173
16.2.2.4 数据路径 ALU 173
16.2.2.5 数据路径输入和复用 176
16.2.2.6 CRC/PRS 支持 177
16.2.2.7 数据路径输出和复用 179
16.2.2.8 数据路径的并行输入和输出 181
16.2.2.9 数据路径链接 181
16.2.2.10 动态配置 RAM 182
16.2.3 状态和控制模块 183

16.2.3.1 状态和控制模式 185
16.2.3.2 控制寄存器操作 186
16.2.3.3 并行输入 / 输出模式 187
16.2.3.4 计数器模式 188
16.2.3.5 同步模式 189
16.2.3.6 状态和控制时序 189
16.2.3.7 辅助控制寄存器 189
16.2.3.8 状态和控制寄存器总结 190
16.2.4 复位和时钟控制模块 190
16.2.4.1 时钟控制 191
16.2.4.2 复位控制 192
16.2.4.3 UDB POR 初始化 197
16.2.5 UDB 寻址 198
16.2.6 系统总线访问一致性 198
16.2.6.1 同时进行系统总线访问 198
16.2.6.2 一致性累加器访问 （原子读和写操作） 198
16.3 端口适配器模块 199
16.3.1 PA 数据输入逻辑 199
16.3.2 PA 端口引脚的时钟复用器逻辑 200
16.3.3 PA 数据输出逻辑 200
16.3.4 PA 输出使能逻辑 201
16.3.5 PA 时钟复用器 202
16.3.6 PA 复位复用器 202

17. 定时器、计数器和 PWM 203

17.1 特性 203
17.2 框图 204
17.2.1 使能和禁用 TCPWM 模块中的计数器 204
17.2.2 时钟 204
17.2.3 基于触发输入的事件 206
17.2.4 输出信号 207
17.2.4.1 发生触发条件时的信号 207
17.2.4.2 中断 207
17.2.4.3 输出 208
17.2.5 功耗模式 208
17.3 各种操作模式 209
17.3.1 定时器模式 210
17.3.1.1 框图 210
17.3.1.2 工作原理 210
17.3.1.3 将计数器配置为定时器模式 212
17.3.2 捕获模式 213
17.3.2.1 框图 213
17.3.2.2 工作原理 213
17.3.2.3 将计数器配置为捕获模式 214
17.3.3 正交解码器模式 215
17.3.3.1 框图 215
17.3.3.2 工作原理 215
17.3.3.3 配置正交模式的计数器 217
17.3.4 脉宽调制模式 218
17.3.4.1 框图 218
17.3.4.2 工作原理 218
17.3.4.3 其他配置 220
17.3.4.4 终止 （kill）特性 220

17.3.4.5 将计数器配置为 PWM 模式 221
17.3.5 带死区时间模式的脉宽调制 222
17.3.5.1 框图 222
17.3.5.2 工作原理 222
17.3.5.3 将计数器配置为带死区时间的 PWM 模式 223
17.3.6 脉宽调制伪随机模式 224
17.3.6.1 框图 224
17.3.6.2 工作原理 224
17.3.6.3 将计数器配置为伪随机 PWM 模式 225
17.4 TCPWM 寄存器 226

18. 蓝牙低功耗子系统 （BLESS） 227

18.1 特性 227
18.2 框图 227
18.3 工作原理 228
18.3.1 LFCLK 初始化 228
18.3.2 无线 PHY 模块 228
18.3.2.1 电源 228
18.3.2.2 RF 初始化 229
18.3.3 链路层控制器 230
18.3.3.1 时钟 230
18.3.3.2 固件复位 231
18.3.3.3 BLE 功能模式和配置 231
18.3.4 功耗模式 232
18.3.4.1 深度睡眠模式 233
18.3.4.2 睡眠模式 233
18.3.4.3 空闲模式 233
18.3.4.4 发送模式 233
18.3.4.5 接收模式 233
18.3.5 模式切换 233
18.3.5.1 LL 睡眠模式进入 （具有自动唤醒功能） 233
18.3.5.2 LL 睡眠模式进入 （无自动唤醒功能） 234
18.3.5.3 手动退出睡眠模式 234
18.3.5.4 LL 深度睡眠模式进入 （具有自动唤醒功能） 234
18.3.5.5 LL 进入扩展深度睡眠模式 235
18.3.5.6 LL 深度睡眠模式手动退出 / 自动退出 236
18.3.5.7 LL 扩展深度睡眠模式手动退出 237
18.3.6 蓝牙 4.2 特性 — 数据长度扩展 238
18.3.7 蓝牙 4.2 特性 — 保密 1.2238
18.3.7.1 解析表 238
18.3.7.2 解析列表功能 239
18.3.7.3 处理不使用 RPA 的对等设备 240
18.3.7.4 处理未解析的自身 RPA 240
18.4 寄存器详细信息 241

章节 E: 模拟系统 245

系统架构 245

18. 高精度参考 247

18.1 特性 247
18.2 框 图 247
18.3 工作原理 248

18.3.1 高精度带隙 248
18.3.2 调整缓冲区 248
18.3.3 低功耗缓冲区 248
18.3.4 电流镜像 249
18.3.5 温度控制的电压发生器 249
18.3.6 温度控制的电流发生器 249
18.4 配置 249

19. SAR ADC 251

19.1 特性 251
19.2 框图 252
19.3 工作原理 253
19.3.1 SAR ADC 内核 253
19.3.1.1 单端模式和差分模式 253
19.3.1.2 输入电压范围 253
19.3.1.3 结果数据格式 253
19.3.1.4 负输入选择 254
19.3.1.5 分辨率 255
19.3.1.6 采集时间 255
19.3.1.7 SAR ADC 时钟 255
19.3.1.8 SAR ADC 时序 256
19.3.2 SARMUX 256
19.3.2.1 模拟路由 256
19.3.2.2 模拟互连 257
19.3.3 SARREF 263
19.3.3.1 参考电压选项 263
19.3.3.2 旁路电容 263
19.3.3.3 输入范围和参考 264
19.3.4 SARSEQ 264
19.3.4.1 求平均 265
19.3.4.2 范围检测 265
19.3.4.3 双缓冲 266
19.3.4.4 插入通道 266
19.3.5 中断 266
19.3.5.1 “扫描结束”中断 （EOS_INTR） 266
19.3.5.2 溢出中断 266
19.3.5.3 冲突中断 267
19.3.5.4 “插入转换结束”中断 （INJ_EOC_INTR） 267
19.3.5.5 范围检测中断 267
19.3.5.6 饱和检测中断 267
19.3.5.7 中断源的概述 267
19.3.6 触发器 267
19.3.6.1 DSI 触发配置 268
19.3.7 SAR ADC 状态 268
19.3.8 低功耗模式 268
19.3.9 系统操作 269
19.3.10 寄存器模式 271
19.3.10.1 SARMUX 模拟路由 271
19.3.10.2 全局 SARSEQ 配置 272
19.3.10.3 通道配置 272
19.3.10.4 通道使能 273
19.3.10.5 中断屏蔽 273
19.3.10.6 触发器 273

19.3.10.7 每当转换中断结束后，都将检索数据 273
19.3.10.8 插入转换 273
19.3.11 DSI 模式 274
19.3.11.1 固件模拟路由 275
19.3.11.2 DSI 模拟路由 275
19.3.11.3 全局 SARSEQ 配置 276
19.3.11.4 DSI 通道配置 276
19.3.11.5 中断 276
19.3.11.6 触发器 276
19.3.11.7 检索数据 277
19.3.11.8 DSI 输出使能 277
19.3.12 模拟路由配置示例 277
19.3.13 温度传感器配置 281
19.4 寄存器 282

20. 低功耗比较器 283

20.1 特性 283
20.2 框图 284
20.3 工作原理 284
20.3.1 输入配置 284
20.3.2 输出和中断配置 285
20.3.3 功耗模式和速度配置 286
20.3.4 迟滞 287
20.3.5 从低功耗模式唤醒 287
20.3.6 比较器时钟 287
20.3.7 偏移调整 287
20.4 寄存器汇总 288

21. 微型连续时间模块 （CTBm） 289

21.1 特性 289
21.2 框图 290
21.3 工作原理 290
21.3.1 功耗模式配置 291
21.3.2 输出驱动配置 291
21.3.3 补偿 292
21.3.4 开关控制 292
21.3.4.1 输入配置 293
21.3.4.2 输出配置 294
21.3.4.3 比较器模式 295
21.3.4.4 比较器配置 295
21.3.4.5 比较器中断 296
21.3.4.6 深度睡眠模式操作 296
21.4 寄存器汇总 298

22. LCD 直接驱动 299

22.1 特性 299
22.2 LCD segment 驱动概述 299
22.2.1 驱动模式 300
22.2.1.1 PWM 驱动模式 300
22.2.1.2 数字相关模式 305
22.2.2 建议使用的驱动模式 308
22.2.3 数字对比度控制 308
22.3 框图 309

22.3.1 工作原理 309
22.3.2 高速和低速主发生器 309
22.3.3 复用器及 LCD 引脚逻辑 310
22.3.4 显示数据寄存器 310
22.4 寄存器列表 310

23. CapSense 311

23.1 特性 311
23.2 框图 311
23.3 工作原理 312
23.4 CapSense CSD 感应 313
23.4.1 GPIO 单元中电容 - 电流转换器 313
23.4.2 CapSense 时钟发生器 315
23.4.3 Sigma Delta 转换器 315
23.5 CapSense CSD 屏蔽 317
23.5.1 CMOD 预充电 318
23.6 通用资源：IDAC 和比较器 319
23.7 寄存器列表 319

24. 温度传感器 321

24.1 特性 321
24.2 工作原理 321
24.3 温度传感器配置 322
24.4 算法 324
24.5 寄存器 324

章节 F: 编程和调试 325

系统架构 325

26. 编程与调试接口 327

26.1 特性 327
26.2 功能说明 327
26.3 串行线调试 （SWD）接口 328
26.3.1 SWD 时序的详细信息 329
26.3.2 ACK 的详细信息 329
26.3.3 反转 （Trn）周期的详细信息 329
26.4 Cortex-M0 调试和访问端口 （DAP） 330
26.4.1 调试端口 （DP）寄存器 330
26.4.2 访问端口 （AP）寄存器 330
26.5 编程 PSoC 4 器件 331
26.5.1 获取 SWD 端口 331
26.5.1.1 主要和辅助 SWD 引脚对 331
26.5.1.2 SWD 端口获取序列 331
26.5.2 SWD 编程模式入口 331
26.5.3 SWD 编程子程序执行 331
26.6 PSoC 4 SWD 调试接口 332
26.6.1 调试控制和配置寄存器 332
26.6.2 断点单元 （BPU） 332
26.6.3 数据观察点 （DWT） 332
26.6.4 调试 PSoC 4 器件 332
26.7 寄存器 333

27. 非易失性存储器编程 335

27.1 特性 335
27.2 功能说明 335
27.3 系统调用实现 336
27.4 阻塞和非阻塞的系统调用 336
27.4.1 执行系统调用 336
27.5 系统调用 337
27.5.1 芯片 ID 337
27.5.2 配置时钟 338
27.5.3 加载闪存字节 339
27.5.4 行写入 340
27.5.5 行编程 341
27.5.6 全部擦除 341
27.5.7 校验和 342
27.5.8 写保护 343
27.5.9 非阻塞行写入 344
27.5.10 非阻塞行编程 344
27.5.11 恢复非阻塞 345
27.6 系统调用状态 346
27.7 非阻塞系统调用伪代码 347

术语表 349

索引 365

章节 A：概述

本部分包括以下章节：

- 第 19 页上的简介章节
- 第 29 页上的入门章节
- 第 31 页上的文档结构章节

文档修订记录

版本	提交日期	变更者	变更说明
**	04/07/2015	RWEI	本文档版本号为 Rev**，译自英文版 001-92738 Rev*A。
*A	08/16/2019	YLIU	本文档版本号为 Rev*A，译自英文版 001-92738 Rev*D。

1. 简介

PSoC®4 是基于 ARM® Cortex®-M0 CPU 的可编程嵌入式系统控制器。它集成了可编程模拟模块、可编程互联、用户可编程的数字逻辑、常用的固定功能外设以及高性能的 ARM Cortex-M0 子系统。PSoC 4xxx-BL 系列基于支持蓝牙的 PSoC 4 架构。该系列 与 PSoC 4 构架的更多产品向上兼容。

PSoC 4 器件具有以下特性：

- 具有单周期乘法运算功能的高性能 32 位 Cortex-M0 CPU 内核
- BLE 射频和子系统
  - 片上 BLE 收发器
  - 链路层控制器符合蓝牙 4.2
- 固定功能和可配置数字模块
- 可编程数字逻辑
- 高性能模拟系统
- 灵活且可编程互联资源
- 支持电容式触摸感应 （CapSense®）功能
- 支持多种低功耗操作模式：睡眠、深度睡眠、休眠和停止模式
- 直接储存器访问 （DMA）

本文档详细描述了 PSoC 器件中的每个功能模块。从而帮助设计员创建系统级设计。

1.1 系统架构

图 1-1 和图 1-2 分别显示了 PSoC 41x7-BL4xx 架构和 PSoC 42x7-BL4xx 架构的主要组件。图 1-3 和图 1-4 分别显示了 PSoC 41x8-BL4xx 架构和 PSoC 42x8-BL4xx 架构的主要组件。图 1-5 和图 1-6 分别显示 PSoC 41x8-BL5xx 架构和 PSoC 42x8-BL5xx 架构的主要组件。

图 1-1. PSoC 41x7-BL4xx 系列框图

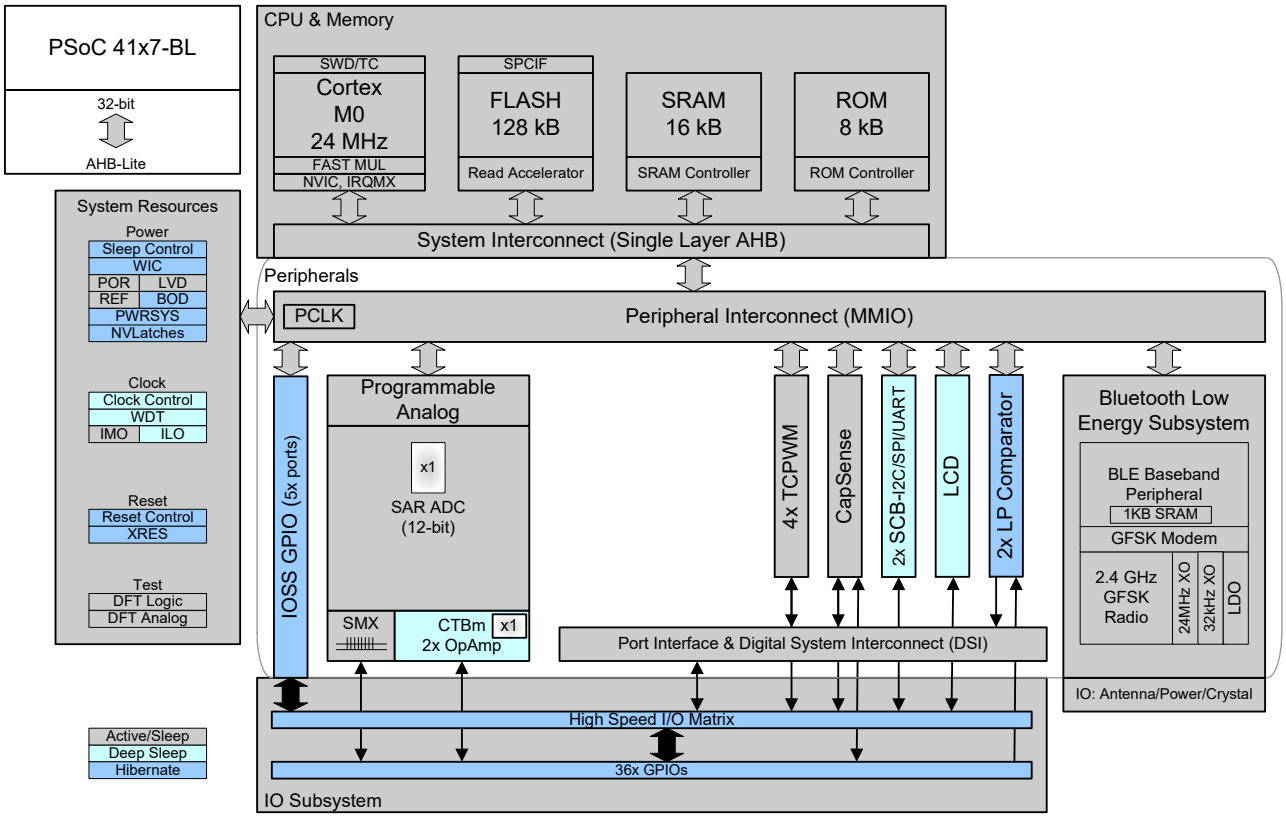


图 1-2. PSoC 42x7-BL4xx 系列框图

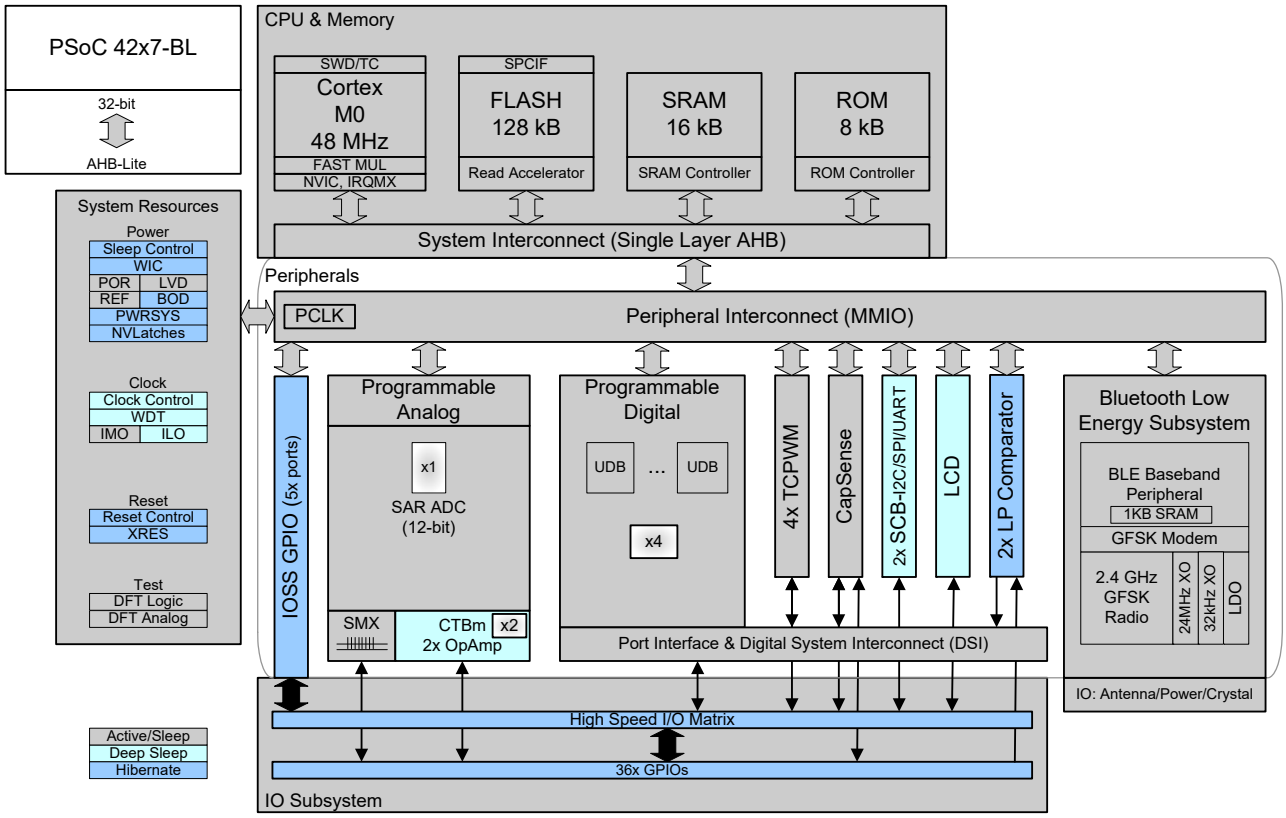


图 1-3. PSoC 41x8-BL4xx 系列框图

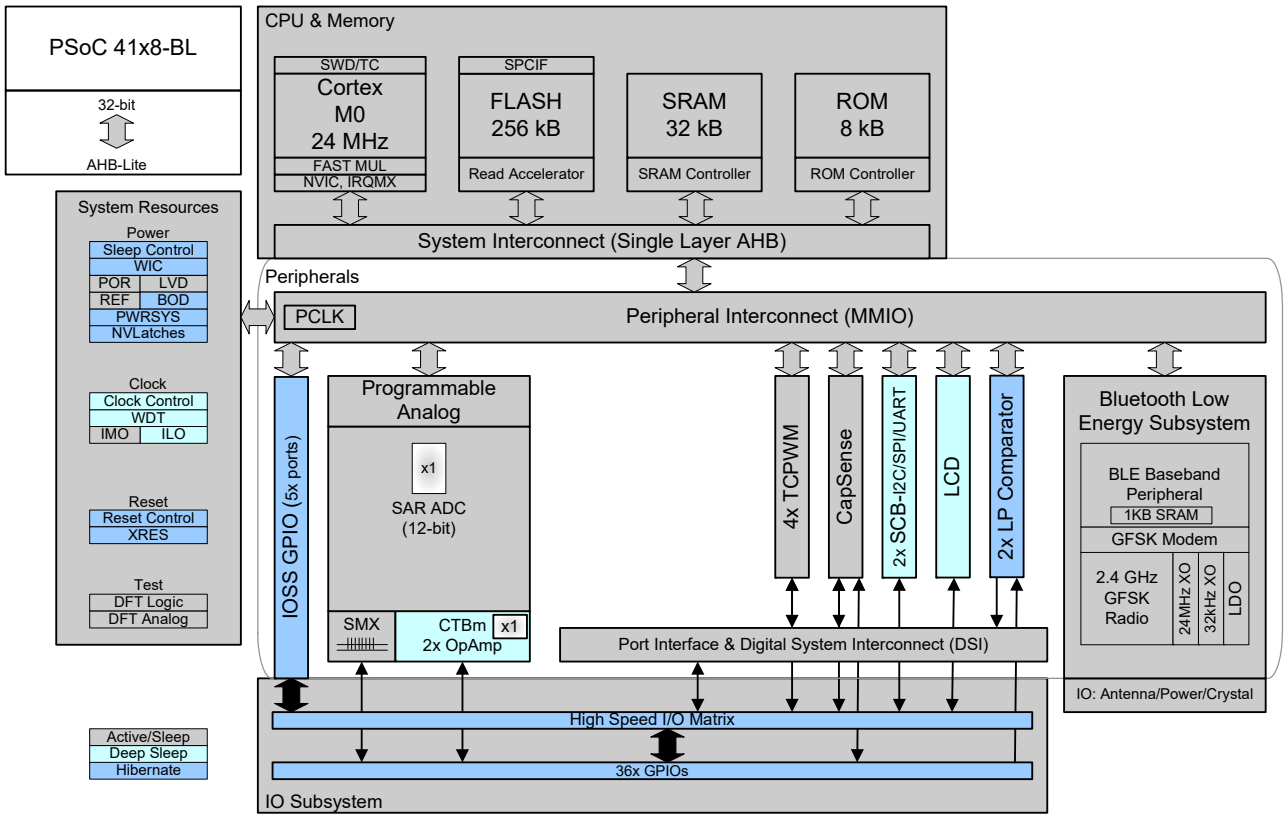


图 1-4. PSoC 42x8-BL4xx 系列框图

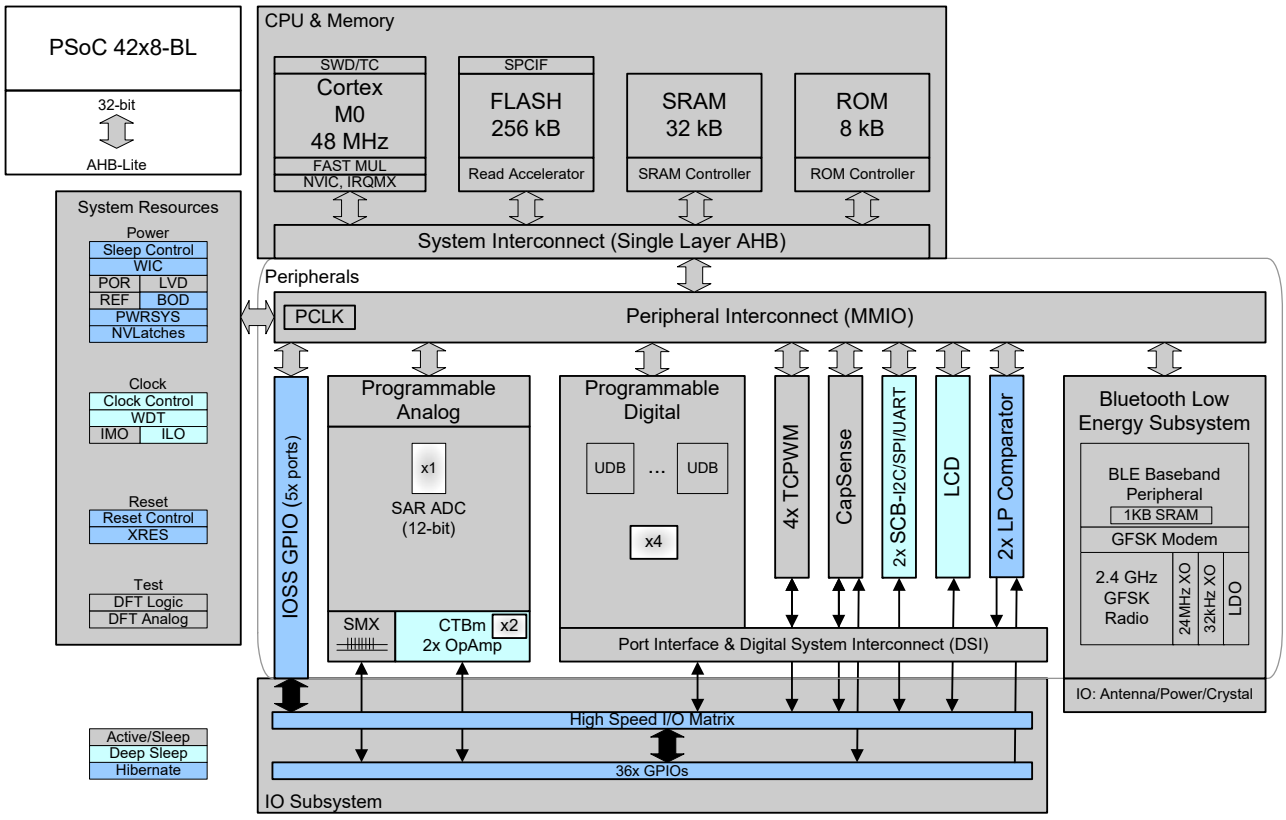


图 1-5. PSoC 41x8-BL5xx 系列框图

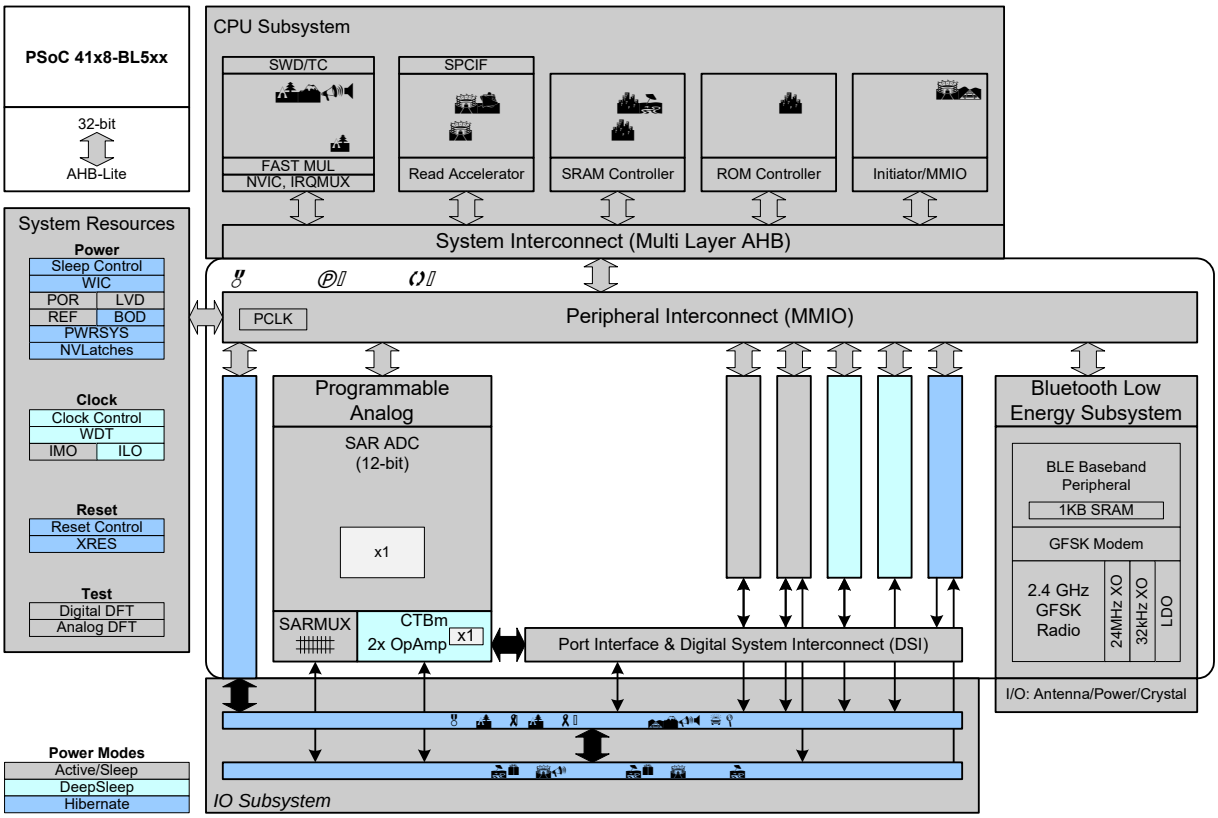
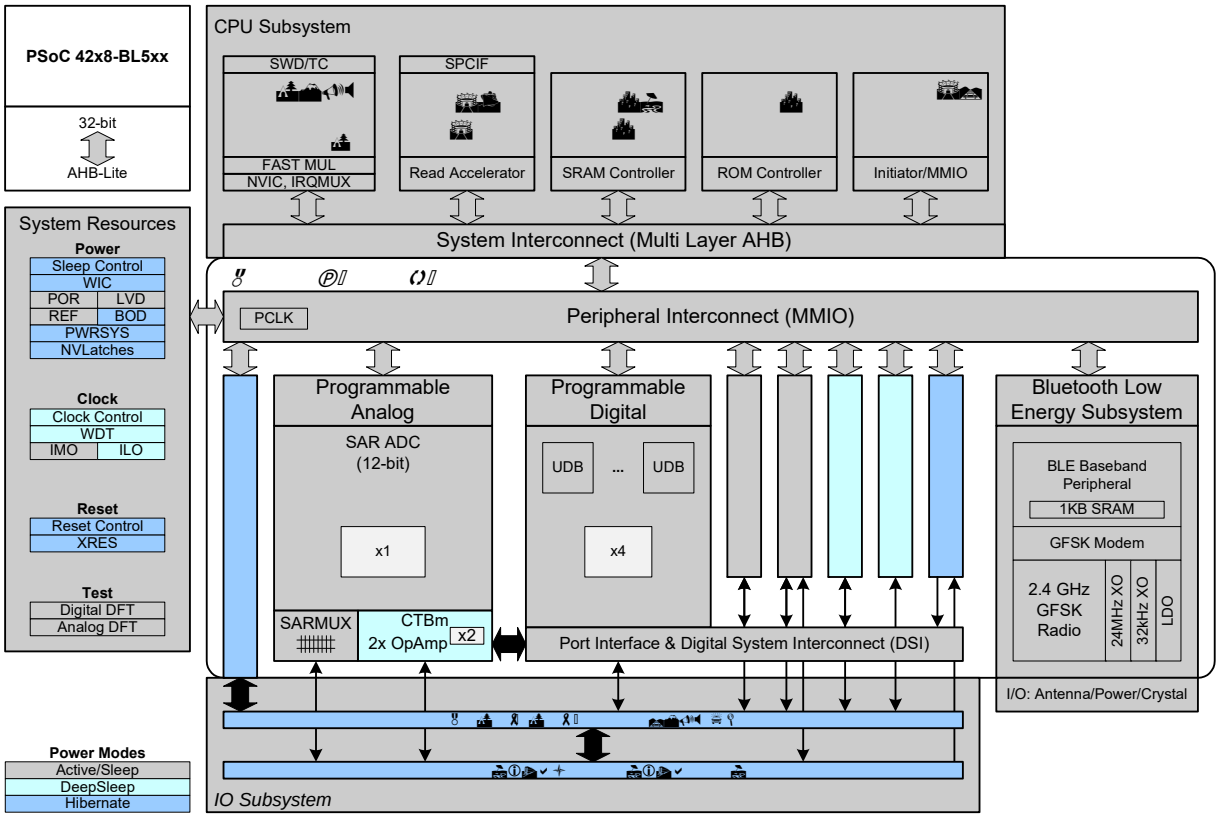


图 1-6. PSoC 42x8-BL5xx 系列框图



1.2 特性

PSoC 4xxx-BL 系列具有以下主要组件：

- BLE 射频和子系统
- 支持单周期乘法功能的32位Cortex-M0 CPU，运行频率为 48 MHz 时，该 CPU 的计算性能高达 43 DMIPS
- 支持最大 256 KB 的闪存和 32 KB 的 SRAM
- 支持直接储存器访问 （DMA）功能
- 四个独立的中心对齐脉冲宽度调制器 （PWM），支持互补、死区时间可编程的输出
- 12 位 SAR ADC （PSoC 42xx-BL 的采样率为 1 Msps 和 PSoC 41xx-BL 的采样率为 806 ksps）提供了一个能够自动切换扫描多个通道的硬件定序器
- 多达四个运算放大器，用于转换模拟信号，并能配置为比较器
- 两个低功耗比较器
- 两个串行通信模块（SCB），可将其配置为 SPI、UART、I^2C 和局部互联网络 （LIN）从设备等各种串行通信接口使用
- 多达四个可编程逻辑模块，即通用数字模块 （UDB）
- CapSense
- SEGMENT LCD 直接驱动
- 支持多种低功耗操作模式：睡眠、深度睡眠、休眠和停止模式
- 支持使用 SWD 接口进行编程和调试
- 全面支持 PSoC Creator™ IDE 工具

1.3 CPU 系统

1.3.1 处理器

PSoC 4 的主要组件是 32 位 Cortex-M0 CPU 内核。在 PSoC 42xx-BL 和 PSoC 41xx-BL 上，它的工作频率分别为 48 MHz 和 24 MHz。该内核通过扩展的时钟门控来优化低功耗操作。它使用了 16 位指令并执行 Thumb-2 指令子集。通过该指令集，可以将完全兼容的二进制代码导入更高性能的处理器，如 Cortex M3 和 M4。

CPU 还拥有一个 32 位的单周期硬件乘法器。

1.3.2 中断控制器

CPU子系统包括一个带有32个中断输入的嵌套向量中断控制器（NVIC）和一个能够从深度睡眠模式唤醒处理器的唤醒中断控制器（WIC）。PSoC 4 的 Cortex-M0 CPU 具有一个不可屏蔽中断（NMI）输入，该输入与数字路由相连，可实现通用操作。

1.3.3 直接存储器访问

通过可编程描述符链，DMA 能够在存储器映射区域执行数据传输（外设到外设、外设到存储器或存储器到外设）。

注意: DMA仅适用于PSoC 41x8-BL5xx和PSoC 42x8-BL5xx系列。

1.4 存储器

PSoC 4 存储器子系统包括闪存和 SRAM。此外，还提供了包含启动和配置子程序的监控 ROM。

1.4.1 闪存

PSoC 4 有一个闪存模块，该模块的闪存加速器与 CPU 紧密耦合在一起，用于缩短闪存模块的平均访问时间。通过闪存加速器，闪存的单周期访问时间平均为SRAM访问时间的85%。

1.4.2 SRAM

PSoC 4 提供了能够在休眠模式中保持数据的 SRAM。

1.5 系统范围资源

1.5.1 时钟系统

PSoC 4 器件的时钟系统包括内部时钟（内部主振荡器（IMO）和内部低速振荡器（ILO）），以及一个外部时钟接口、外部晶体振荡器（ECO）和监视晶体振荡器（WCO）。

精度误差为±2% 的IMO是器件中主要的内部时钟源。可以从主时钟频率生成多个分频时钟，以满足应用需求。

ILO 是一个精度较低的低功耗振荡器，并作为 LFCLK 的时钟源使用，用以为外设在深度睡眠模式下运行时生成时钟。ILO 时钟频率为 32 kHz，精度为 ±60 %。

可以使用一个频率范围为 0 ~ 48 MHz 的外部时钟源（替换 IMO）为功能模块提供时钟。

使用 ECO 来生成一个高精度的 24 MHz 时钟而不需要任何外部组件。它主要用于为 BLE 子系统（包括 Link Layer 引擎、数字 PHY 调制解调器及 RF 收发器）提供时钟。高精度 ECO 时钟还能作为 PSoC 4 器件的时钟源使用。

WCO 作为LFCLK 源使用。WCO 用于在深度睡眠模式下准确保持 广播事件及连接事件 的时间间隔。与 ILO 相同， WCO 在所有模式（休眠和停止模式除外）下均可用。

1.5.2 电源系统

PSoC 4 由单外部电源供电，其工作电压范围为 1.71 V 到 5.5 V。

除了默认的活动模式外，PSoC 4 还能在四种低功耗模式（即睡眠、深度睡眠、休眠和停止 模式）下运行。在活动模式下，CPU处于运行状态，所有逻辑均被供电。在睡眠模式下，CPU停止运行，所有其它外设仍正常工作。在深度睡眠模式下，CPU、 SRAM 和高速逻辑模块均暂停。主系统时钟关闭，低频时钟启用，并且低频外设仍然运行。在休眠模式下，低频时钟被关闭，低频外设停止运行。

系统中的多个内部电压调节器都可用，用于满足不同功耗模式的供电要求。

1.5.3 GPIO

PSoC 4 中的每个 GPIO 均具有以下特性：

- 支持八种强驱动模式
- 能够单独控制禁用输入和输出
- 支持保持模式，用于锁存先前状态
- 可选的转换速率
- 中断生成 — 边沿触发
- 支持 CapSense 和 LCD 驱动

PSoC 4 具有两个过压容限端口 , 这两个端口允许 I2C 满足快速模式断电的规范，以较低的 V_{DD} 运行时，该端口能够连接到更高电压的总线。

这些引脚组成了一个 8 位宽的端口。高速 I/O 矩阵用于复用连接某个 I/O 引脚上的多个信号。固定功能外设的引脚的位置是固定的。

1.6 蓝牙低功耗子系统

PSoC 4xxx 蓝牙低功耗（BLE）子系统集成了射频收发器、数字 PHY 调制解调器和链路层控制器。

1.6.1 射频收发器

射频收发器包括集成的 Balun。该 Balun 提供了一个单端射频端口引脚，这样能够通过匹配网络驱动一个 50 Ω 的天线终端。接收时，该模块将天线的射频信号转换为 1 MHz 的中间频率，并将模拟信号数字化为 10 位的数字信号。传输时，该模块从数字 PHY 提取调制后的 1 Mbps GFSK，然后将其转换为射频，并通过天线传输出去。

1.6.2 数字 PHY 调制解调器

传输时，该子模块会从链路层控制器中提取 1 Mbps 的串行数据，然后生成 GFSK 直接调制的数据并将其发送给 BLE 模拟部分。接收时，它从 BLE 模拟部分提取 1 MHz 的 IF ADC 数据，并使用数字解调器生成 1 Mbps 的串行数据。

1.6.3 链路层控制器

链路层控制器执行蓝牙低功耗链路层规范中指定的所有时序关键性能（包成帧 / 解帧、CRC 生成 / 检测、加密 / 解密、状态机和数据传输）；另外，它还将接口供给数字 PHY。使用中断、 FIFO 和寄存器来进行链路层硬件和固件之间的通信。

1.7 可编程数字模块

PSoC 42xx-BL 具有多达四个 UDB。每个 UDB 均包含一个结构化的数据路径逻辑和一个不受限制的 PLD 逻辑，它们之间具有灵活的互联。 UDB 阵列提供了一个切换型路由结构，即数字信号互联 （DSI）。通过 DSI，可以将信号从外设或端口路由到 UDB，也可以在各个 UDB 之间路由信号。

通过 PSoC 42xx-BL 中的 UDB 阵列，可以创建各个自定义的逻辑模块或额外定时器 /PWM 和通信接口，如 I^2C、SPI、I2S 和 UART。

注意： PSoC 41xx-BL 不支持 UDB。

1.8 固定功能数字模块

1.8.1 定时器 / 计数器 /PWM 模块

定时器 / 计数器 /PWM 模块包含四个用户可编程周期长度的 16 位计数器。可对这些计数器的功能进行同步化处理。每个模块都有捕获寄存器、周期寄存器和比较寄存器。该模块支持互补、死区可编程的输出。它还提供了用于强制输出进入未确定状态的停止 （Kill）输入。此外，该模块还支持中心对齐 PWM、时钟预分频器、伪随机 PWM 和正交解码器等特性。

1.8.2 串行通信模块

该器件具有两个 SCB。每个 SCB 可以实现一个串行通信接口，如 I^2C、UART、局部互联网络 （LIN）从设备或 SPI。

每个 SCB 的特性包括：

- 支持标准的 I^2C 多主设备和从设备功能
- 支持 Motorola、 Texas Instruments 和 National （MicroWire）模式下的标准 SPI 主设备和从设备功能
- 具有标准的 UART 发送器和接收器功能，从而可以支持 SmartCard reader （ISO7816）（智能卡读卡器（ISO7816））、 IrDA 协议和 LIN
- 该LIN从设备符合LIN版本1.3和LIN版本2.1/2.2规范的标准。
- 支持自带 32 字节缓冲区的 EzSPI 和 EzI^2C 功能模式

1.9 模拟系统

1.9.1 SAR ADC

PSoC 42xx-BL 具有一个采样速率为 1 Msps 的可配置 12 位 SAR ADC，PSoC 41xx-BL 也有一个相同的12位SAR ADC，其采样速率为 806 ksps。该 ADC 提供了三个内部参考电压（V_{DDA}、$V_{DDA}/2$ 和 V_{REF}），通过 GPIO 引脚还提供了一个外部参考电压。可以使用 SAR 硬件序列发生器扫描多个通道，无需 CPU 的干预。

1.9.2 微型连续时间模块

微型连续时间模块 （CTBm）在模拟子系统的入口和出口处提供了连续时间功能。 CTBm 具有两个可灵活配置的高性能运算放大器以及一个开关路由矩阵。运算放大器也可以在比较器模式中运行。PSoC 42xx-BL 具有两个 CTBm 模块，而 PSoC 41xx-BL 只有一个 CTBm 模块。

通过该模块，无需外部组件仍能够实现开环运算放大器、线性缓冲区和比较器功能。作为 PGA、电压缓冲区、滤波器和互阻抗放大器时，则需要使用外部组件。CTBm 模块可在活动、睡眠和深度睡眠模式下运行。

1.9.3 低功耗比较器

PSoC 4xxx-BL 有一对支持所有器件功耗模式的低功耗比较器。在低功耗模式下，通过该功能可禁用 CPU 和其它系统模块，但仍能监控外部电压。两个输入电压可以全部来自引脚，或一个来自引脚，另一个由内部信号通过 AMUXBUS 提供。

1.10 特殊功能的外设

1.10.1 LCD Segment 驱动

PSoC 4 具有一个能够驱动 4 个 COMMON 的 LCD 控制器，此外，还可以配置每个 GPIO，使之驱动 COMMON 或 SEGMENT。该控制器使用完整的数字方法 （数字关联和 PWM）驱动 LCD SEGMENT，而不需要生成内部 LCD 电压。

1.10.2 CapSense

PSoC 4 器件提供 CapSense 功能，从而您可以使用手指的电容属性来完成对按键和滑条的操作。通过使用 CapSense Sigma-Delta （CSD）模块， PSoC 4 中的所有 GPIO 引脚都支持 CapSense 功能。CSD 还提供了防水功能。

1.10.2.1 IDAC 和比较器

CapSense 模块包含两个 IDAC 和一个比较器，其参考电压为 1.2 V；在不使用CapSense的情况下，它们可用于通用目的。

1.11 编程和调试

PSoC 4 器件支持通过片上 SWD 接口对器件进行编程和调试操作。 PSoC Creator IDE 提供了全面集成的编程和调试支持。 SWD 接口与行业标准的第三方工具全面兼容。

1.12 器件特性总结

表 1-1 显示了 PSoC 41xx-BL/42xx-BL 器件总结。

表 1-1. PSoC 41xx-BL/42xx-BL 器件总结

特性	PSoC 41xx-BL	PSoC 42xx-BL
最大 CPU 频率	24 MHz	48 MHz
闪存	PSoC 41x7-BL：128 KB PSoC 41x8-BL：256 KB	PSoC 42x7-BL：128 KB PSoC 42x8-BL：256 KB
SRAM	PSoC 41x7-BL：16 KB PSoC 41x8-BL：32 KB	PSoC 42x7-BL：16 KB PSoC 42x8-BL：32 KB
最大 GPIO 数量	38	38
CapSense	支持	支持
LCD 驱动器	支持	支持
定时器、计数器、 PWM （TCPWM）	4	4
串行通信模块 （SCB）	2	2
通用数字模块 （UDB）	不支持	4
IDAC （CapSense 的一部分）	2	2
运算放大器	2	4
比较器	2	2
模数转换器 （ADC）	12 位 SAR， 806 ksps	12 位 SAR， 1 Msps
蓝牙	支持	支持

2. 入门

2.1 支持服务

可以访问 www.cypress.com/psoc4 以获取 PSoC® 4 产品的免费支持。资源包括培训讲座、论坛、应用笔记、PSoC 顾问、CRM 技术支持电子邮件、知识库以及应用支持工程师。

有关应用支持，请访问 www.cypress.com/support/ 或电话联系 1-800-541-4736。

2.2 产品升级

赛普拉斯免费提供 PSoC Creator 的定期升级以及版本更新。可以从分销商所提供的 DVD-ROM 中找到升级版本，也可以直接从 www.cypress.com/psoccreator 上下载升级版本。“ 文档 ” 章节中也提供系统文档的重要更新。

2.3 开发套件

赛普拉斯在线商店包含所有你成功开发 PSoC4 BLE 项目所需的开发套件、C 编译器和附件。请访问赛普拉斯的在线商店 www.cypress.com/cypress-store。在**产品**目录下面，通过点击 **Programmable System-on-Chip** 可以查找当前可用的器件系列。开发套件也可由 Digi-Key、 Avnet、 Arrow 以及 Future 公司提供。

2.4 应用笔记

请参见应用笔记 *AN91267 — PSoC 4 BLE 入门*以获取更多有关 PSoC 4 BLE 器件功能的信息，并了解如何使用 PSoC Creator 和 BLE 开发套件进行快速创建一个简单的 BLE 应用。

3. 文档结构

本文档包括以下部分：

- 章节 B：第 35 页上的 CPU 系统
- 章节 C：第 67 页上的系统资源的子系统 （SRSS）
- 章节 D：第 117 页上的数字系统
- 章节 E：第 245 页上的模拟系统
- 章节 F：第 325 页上的编程和调试

3.1 主要部分

为了便于使用，本文档中的信息被整理为各部分和章节。这些部分和章节根据器件的功能来区分。

- 部分 — 介绍了产品的系统架构、入门方法、相关规范和概述信息。
- 章节 — 介绍了有关部分主题的某一领域的特定章节。它们是针对集成电路的某些方面的详细实现和使用信息。
- 术语表 — 定义了本技术参考手册 （TRM）中使用的专业术语。术语表使用粗斜体字来显示。
- 寄存器技术参考手册 — 提供了技术参考手册中总结的所有器件寄存器的详细信息。这是一份附加文档。

3.2 文档规范

除了标题中使用的那些字体类型以外，本文档只使用四个明显的字体类型。

- 第一类型是*斜体字*，在谈及某些文档标题或文件名称时使用。
- 第二类型是***粗斜体字***，在谈及本文档中术语表所描述的某些术语时使用。
- 第三类型使用了 Times New Roman 字体，用以区分公式示例。
- 第四类型使用 `Courier New` 字体，用以区分代码示例。

3.2.1 寄存器约定

有关详细信息，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册*。

3.2.2 数字命名

十六进制数字中的所有字母均为大写，结尾带小写的 'h'（例如，'14h' 或 '3Ah'），此外，还可以使用 '0x' 前缀来表示*十六进制*数字 （根据 *C* 代码规范）。二进制数字在结尾带小写的 'b'（例如，'01010100b' 或 '01000011b'）。不带 'h' 或 'b' 的数字是*十进制数字*。

3.2.3 测量单位

下表列出了本文档中使用的测量单位。

表 3-1. 测量单位

缩略词	测量单位
bps	每秒比特数
°C	摄氏度
dB	分贝
dBm	分贝毫瓦
fF	飞法
G	千兆
Hz	赫兹
k	千， 1000
K	千， 2^10
KB	1024 个字节或大约一千个字节
Kbit	1024 位
kHz	千赫 （32.000）
kΩ	千欧
MHz	兆赫兹
MΩ	兆欧
µA	微安
µF	微法
µs	微秒
µV	微伏
µVrms	微伏均方根
mA	毫安
ms	毫秒
mV	毫伏
nA	纳安
ns	纳秒
nV	纳伏
Ω	欧姆
pF	皮法
pp	峰 - 峰值
ppm	百万分比
SPS	每秒采样数
σ	sigma：一个标准差
V	伏特

3.2.4 缩略语

下表列出了本文档中使用的缩略语

表 3-2. 缩略语

缩略语	定义
ABUS	模拟输出总线
AC	交流
ADC	模数转换器
ADV	广播

表 3-2. 缩略语 *（续）*

缩略语	定义
AES	高级加密标准
AHB	AMBA （高级微控制器总线结构）高性能总线，一种 ARM 数据传输总线
API	应用编程接口
APOR	模拟上电复位
BC	广播时钟
BLE	低功耗蓝牙，也称为 Bluetooth Smart
BLESS	BLE 子系统
BOD	欠压检测
BOM	物料表
BR	比特率
BRA	总线请求确认
BRQ	总线请求
CAN	控制器区域网络
CI	移入
CMP	比较
CO	移出
CPU	中央处理单元
CRC	循环冗余校验
CSD	电容触摸感应三角积分模块
CT	连续时间
CTB	连续时间模块
CTBm	微型连续时间模块
DAC	数模转换器
DAP	调试访问端口
DC	直流
DI	数字或数据输入
DMA	直接存储器访问
DNL	微分非线性
DO	数字或数据输出
DSI	数字信号接口
DSM	深度睡眠模式
DW	数据线
ECO	外部晶体振荡器
EEPROM	电可擦除可编程只读存储器
EMIF	外部存储器接口
FB	反馈
FIFO	先进先出
FSR	全量程范围
GAP	通用访问配置文件
GATT	通用属性配置文件
GFSK	高斯频移键控
GPIO	通用输入 / 输出
HCI	主机控制接口 （BLE 堆栈）
HFCLK	高频时钟
HSIOM	高速输入 / 输出矩阵

表 3-2. 缩略语 *（续）*

缩略语	定义
I^2C	内部集成电路
IDE	集成开发环境
ILO	内部低速振荡器
ITO	铟锡氧化物
IMO	内部主振荡器
INL	积分非线性
I/O	输入 / 输出
IOR	I/O 读取
IOW	I/O 写入
IRES	初次上电复位
IRA	中断请求确认
IRQ	中断请求
ISR	中断服务子程序
IVR	中断向量读取
L2CAP	逻辑链接控制和适配协议
LCD	液晶显示屏
LFCLK	低频率时钟
LIN	局部互联网络
LPCOMP	低功耗比较器
LRb	最后接收位
LRB	最后接收字节
LSb	最低有效位
LSB	最低有效字节
LUT	查找表
MISO	主入从出
MMIO	存储器映射输入 / 输出
MOSI	主出从入
MSb	最高有效位
MSB	最高有效字节
NMI	不可屏蔽中断
NVIC	嵌套向量中断控制器
PC	程序计数器
PCB	印刷电路板
PCH	程序计数器的高字节
PCL	程序计数器的低字节
PD	断电
PDU	协议数据单元
PGA	可编程增益放大器
PHY	物理层
PM	电源管理
PMA	PSoC 存储器仲裁器
POR	上电复位
PPOR	精确上电复位
PRS	伪随机序列
PSoC®	可编程片上系统
PSRR	电源抑制比

表 3-2. 缩略语 *（续）*

缩略语	定义
PSSDC	电源系统睡眠占空比
PWM	脉冲宽度调制器
RAM	随机访问存储器
RETI	中断返回
RF	射频
ROM	只读存储器
RMS	均方根
RW	读 / 写
SAR	逐次逼近寄存器
SC	开关电容
SCB	串行通信模块
SIE	串行接口引擎
SIO	特殊输入 / 输出
SE0	单端零
SNR	信噪比
SOF	帧头
SOI	指令起始
SP	堆栈指针
SPD	连续相位检测器
SPI	串行外设互连
SPIM	串行外设互连主设备
SPIS	串行外设互连从设备
SRAM	静态随机访问存储器
SROM	只读管理存储器
SSADC	单斜模数转换器
SSC	监控系统调用
SYSCLK	系统时钟
SWD	单线调试
TC	终端计数
TCPWM	定时器、计数器、脉冲宽度调制器
TD	数据传输描述符
UART	通用异步接收器 / 发送器
UDB	通用数字模块
USB	通用串行总线
USBIO	USB 输入 / 输出
WCO	监视晶体振荡器
WDT	监视定时器
WDR	监视复位
XRES	外部复位
XRES_N	外部复位，低电平有效

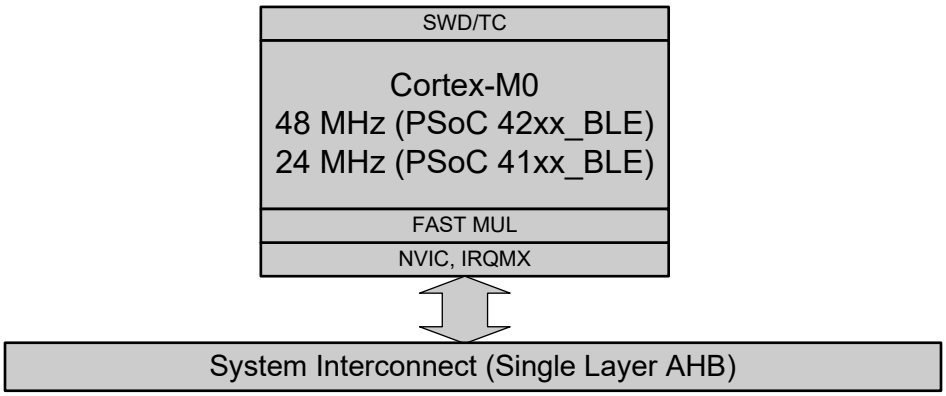
章节 B: CPU 系统

本部分包括下面章节：

- 第 37 页上的 Cortex-M0 CPU 章节
- 第 57 页上的中断章节

系统架构

CPU 系统框图



4. Cortex-M0 CPU

PSoC® 4 ARM Cortex-M0 内核是一个低功耗的 32 位 CPU。它拥有一个高效率的三级流水线、一个固定的 4 GB 存储器映射，另外还支持 ARMv6-M Thumb 指令集。Cortex-M0 还提供一个单周期的 32 位乘法指令和低延迟的中断处理程序。紧密与 CPU 内核相连的其他子系统包括：一个嵌套向量中断控制器 （NVIC）、一个 SYSTICK 定时器和调试。

本节介绍了 Cortex-M0 处理器的概述。更多有关信息，请参见 www.arm.com 网站中提供的 ARM Cortex-M0 用户指南或技术参考手册。

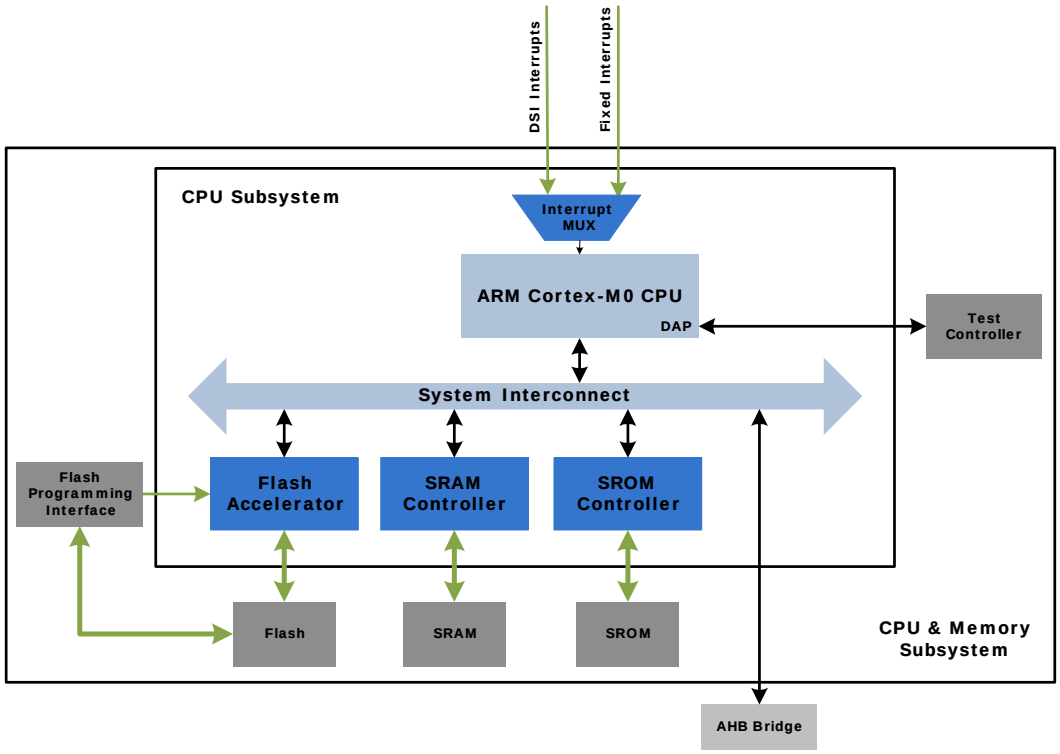
4.1 特性

PSoC 4 Cortex-M0 具有以下特性：

- 易于使用、编程和调试，因此可以从 8 位和 16 位处理器进行更简单的移植
- 工作速度可达 0.9 DMIPS/MHz ；这样允许增大执行速度或降低功耗
- PSoC41xx_BL 和 PSoC 42xx_BL 中的最大 CPU 时钟频率分别为 24 MHz 和 48 MHz。
- 支持 Thumb 指令集，这样可以提高代码密度和对存储器的利用率
- 支持中断和异常的 NVIC 单元，用于确保快速和确定的中断响应
- 广泛的调试支持包括：
  - SWD 端口
  - 断点
  - 观察点

4.2 框图

图 4-1. PSoC 4 CPU 子系统框图



4.3 工作原理

Cortex-M0 是一个 32 位的处理器，它包含一个 32 位数据路径、 32 位寄存器和一个 32 位存储器接口。它支持 Thumb 指令集中的大多数 16 位指令和 Thumb-2 指令集中的一些 32 位指令。

该处理器支持两种工作模式 （见第 40 页上的 “ 工作模式 ” 一节）。它有一个单周期的 32 位乘法指令。

4.4 地址映射

ARM Cortex-M0 具有固定的地址映射，因此可以通过简单的存储器访问指令访问存储器和外设。32 位（4 GB）地址空间可分为多个区域，如表 4-1 所示。请注意，可从代码区和 SRAM 区执行代码。

表 4-1. Cortex-M0 地址映射情况

地址范围	名称	使用说明
0x00000000 – 0x1FFFFFFF	代码	编程代码区域。您也可以将数据存储在此处，包括从地址 0 开始的异常向量表。
0x20000000 – 0x3FFFFFFF	SRAM	数据区域。您也可以执行该区域中的代码。
0x40000000 – 0x5FFFFFFF	外设	所有的外设寄存器。您不能执行该区域中的代码。
0x60000000 – 0xDFFFFFFF		未使用。
0xE0000000 – 0xE00FFFFF	PPB	CPU 内核的外设寄存器。
0xE0100000 – 0xFFFFFFFF	器件	PSoC 4 的专属空间

4.5 寄存器

Cortex-M0 具有 16 个 32 位寄存器，如表 4-2 中所示：

- R0 到 R12 — 通用寄存器。 R0 到 R7 可通过所有指令访问，其它寄存器可通过指令子集访问。
- R13 — 堆栈指针（SP）。有两个堆栈指针，一次只有一个指针可用。在线程模式下，CONTROL 寄存器指出了将要使用的堆栈指针：主堆栈指针 （MSP）还是进程堆栈指针 （PSP）。
- R14 — 链接寄存器。在调用函数过程中存储返回程序计数器。
- R15 — 程序计数器。通过写入到该寄存器内，可以控制程序流。

表 4-2. Cortex-M0 寄存器

名称	类型[a]	复位值	说明
R0-R12	RW	未定义	R0-R12 是用于数据操作的 32 位通用寄存器。
MSP （R13） PSP （R13）	RW	[0x00000000]	堆栈指针 （SP）是寄存器 R13。在线程模式下， CONTROL 寄存器的位 [1] 指出了所使用的堆栈指针： 0 = 主堆栈指针 （MSP）。这是复位值。 1 = 进程堆栈指针 （PSP）。 复位时，处理器将地址 0x00000000 中的值加载到 MSP。
LR （R14）	RW	未定义	链接寄存器 （LR）是寄存器 R14。它存储子程序、函数调用和异常情况的返回信息。
PC （R15）	RW	[0x00000004]	程序计数器（PC）是寄存器 R15。它包含当前程序地址。复位时，处理器将地址 0x00000004 中的值加载到 PC 内。复位时，该值的位 [0] 必须为 1 并被加载到 EPSR T 位内。
PSR	RW	未定义	程序状态寄存器 （PSR）组合： 应用程序状态寄存器 （APSR）。 执行程序状态寄存器 （EPSR）。 中断程序状态寄存器 （IPSR）。
APSR	RW	未定义	APSR 包含先前指令执行程序中条件标志的当前状态。
EPSR	RO	[0x00000004].0	复位时，会将寄存器 [0x00000004] 中的位 [0] 加载到 EPSR 内。
IPSR	RO	0	IPSR 包含当前 ISR 的异常编号。
PRIMASK	RW	0	PRIMASK 寄存器防止激活可配置优先级的异常情况。
CONTROL	RW	0	CONTROL 寄存器用于控制处理器处于线程模式时所使用的堆栈。

a. 说明程序在线程模式和处理模式下运行时的访问类型。调试访问会有所不同。

表 4-3 显示的是 PSR 位的分配方式。

表 4-3. Cortex-M0 PSR 位分配

位	PSR 寄存器	名称	使用情况
31	APSR	N	负向标志
30	APSR	Z	零标志
29	APSR	C	进位标志或借位标志
28	APSR	V	溢流标志

表 4-3. Cortex-M0 PSR 位分配

位	PSR 寄存器	名称	使用情况
27 – 25	–	–	保留
24	EPSR	T	Thumb 状态位。 T 位必须为 1。如果该位为 0，则尝试执行指令会导致 HardFault 异常。
23 – 6	–	–	保留
5 – 0	IPSR	N/A	当前 ISR 的异常编号： 0 = 线程模式 1 = 保留 2 = NMI 3 = HardFault 4 – 10 = 保留 11 = SVCall 12、 13 = 保留 14 = PendSV 15 = SysTick 16 = IRQ0 … 47 = IRQ31

通过使用 MSR 或 CPS 指令，可以设置或清除 PRIMASK 寄存器的位 0。如果该位为 ‘0’，将使能异常。如果该位为 ‘1’，将禁用所有可配置优先级的异常 （HardFault、 NMI 和复位除外）。第 57 页上的中断章节显示的是异常列表。

4.6 工作模式

Cortex-M0 处理器支持两种工作模式：

- 线程模式 — 由所有普通应用使用。在线程模式下，可以使用 MSP 或 PSP。CONTROL 寄存器的位 1 用于确定将使用的堆栈指针：
  - 0 = 当前的堆栈指针为 MSP
  - 1 = 当前的堆栈指针为 PSP
- 处理程序模式 — 用于执行异常处理程序。在该模式下，始终使用 MSP。

在线程模式下，通过使用 MSR 指令，可以设置 CONTROL 寄存器中的堆栈指针位。修改堆栈指针时，使用 MSR 指令后可立即使用 ISB 指令。这样就能确保 ISB 指令后的指令使用新的堆栈指针执行。

在处理程序模式下，对 CONTROL 寄存器进行写操作无效，这是因为 MSP 一直被使用。异常进入和返回机制会自动更新 CONTROL 寄存器。

4.7 指令集

Cortex-M0 执行 Thumb 指令集的一个版本，如表 4-4 中所示。有关详细内容，请参见 *Cortex-M0 通用用户指南*。

指令操作数可以是一个 ARM 寄存器、一个常量，或者是另一个指令特定参数。在操作数上执行指令，并经常将该结果存储在目标寄存器内。并非所有指令都能将 PC 或 SP 作为操作数或目标寄存器使用。

表 4-4. Thumb 指令集

助记符	简要说明
ADCS	进位加
ADD{S}[a]	添加
ADR	PC 相对地址到寄存器
ANDS	位元 AND
ASRS	算术右移
B{cc}	分支 { 条件 }
BICS	位清除
BKPT	断点
BL	分支链接
BLX	分支间接链接
BX	分支间接
CMN	负值比较
CMP	比较
CPSID	修改处理器状态，禁用中断
CPSIE	修改处理器状态，使能中断
DMB	数据存储器屏障
DSB	数据同步屏障
EORS	逻辑异或
ISB	指令同步屏障
LDM	加载多个寄存器，并且每次加载后都会递增地址
LDR	将 PC 相对的地址值加载到寄存器内
LDRB	将字加载到寄存器内
LDRH	将半字加载到寄存器内
LDRSB	将带符号的字节加载到寄存器内
LDRSH	将带符号的半字加载到寄存器内
LSLS	逻辑左移
LSRS	逻辑右移
MOV{S}[a]	移动
MRS	将数据从特殊寄存器内转移到通用寄存器内
MSR	将数据从通用寄存器内转移到特殊寄存器内
MULS	乘法，结果为 32 位数据
MVNS	位元 NOT
NOP	无操作
ORRS	逻辑 ‘ 或 ’
POP	从堆栈弹出寄存器
PUSH	将寄存器推入堆栈内
REV	字节反转字
REV16	紧密式半字的字节反转
REVSH	有符号半字的字节反转
RORS	向右旋转
RSBS	反向减法
SBCS	进位减

表 4-4. Thumb 指令集

助记符	简要说明
SEV	发送事件
STM	存储多个寄存器，并且每次存储后递增地址
STR	存储一个字
STRB	存储一个字节
STRH	存储半字
SUB{S}[a]	减
SVC	管理程序调用
SXTB	符号扩展字节
SXTH	符号扩展半字
TST	基于逻辑 AND 的测试
UXTB	对字节进行零扩展
UXTH	对半字进行零扩展
WFE	等待事件
WFI	等待中断

a. ‘S’ 限定符会使 ADD、 SUB 或 MOV 指令更新 APSR 条件标志。

4.7.1 地址对齐

对齐访问是一个操作，其中字对齐的地址用于一个字或许多字访问，半字对齐的地址用于半字访问。字节访问始终是对齐的。

Cortex-M0 处理器上的非对齐访问不受任何支持。尝试执行非对齐存储器访问操作会导致 HardFault 异常。

4.7.2 存储器中的字节顺序

PSoC 4 Cortex-M0 使用低位优先格式，其中最低有效字节被存储在最低地址，最高有效字节被存储在最高地址。

4.8 Systick 定时器

Systick 定时器和 NVIC 被集成在一个子系统中，用于生成 SYSTICK 中断。该中断可使用于实时系统中的任务管理。该定时器带有一个重载寄存器，其中可将 24 位作为一个倒计数值使用。Systick 定时器将 Cortex-M0 内部时钟作为源时钟使用。

4.9 调试

PSoC 4 包含一个基于 SWD 的调试接口；它具有四个断点（地址）比较器和两个观察点 （数据）比较器。

5. DMA 控制器模式

DMA 控制器提供了数据线 （DW）和直接存储器访问 （DMA）功能。 DMA 控制器具有以下特性：

- 支持多达 8 个 DMA 通道； 对于某个特定器件，请查询器件数据手册，以确定它支持的通道数量
- 每个通道支持四个优先级
- 支持字节、半字 （2 个字节）和字 （4 个字节）传输
- 每个通道支持三种工作模式
- 可配置中断生成
- 传输完成触发信号输出
- 最大传输数据元素为 65,536

DMA 控制器支持三种工作模式。这些工作模式在 DMA 控制器在单个触发器信号上运行的方式方面存在差别。通过这些工作模式，用户可以实现不同的 DMA 工作模式。这些工作模式分别为：

- 模式 0：每个触发传输单数据元素
- 模式 1：每个触发传输所有数据元素
- 模式 2：每个触发传输所有数据元素，并自动触发链式描述符

数据传输的各项特性 （如源地址位置、目标地址位置以及传输大小）都是由描述符结构指定的。每个通道具有独立的描述符结构。

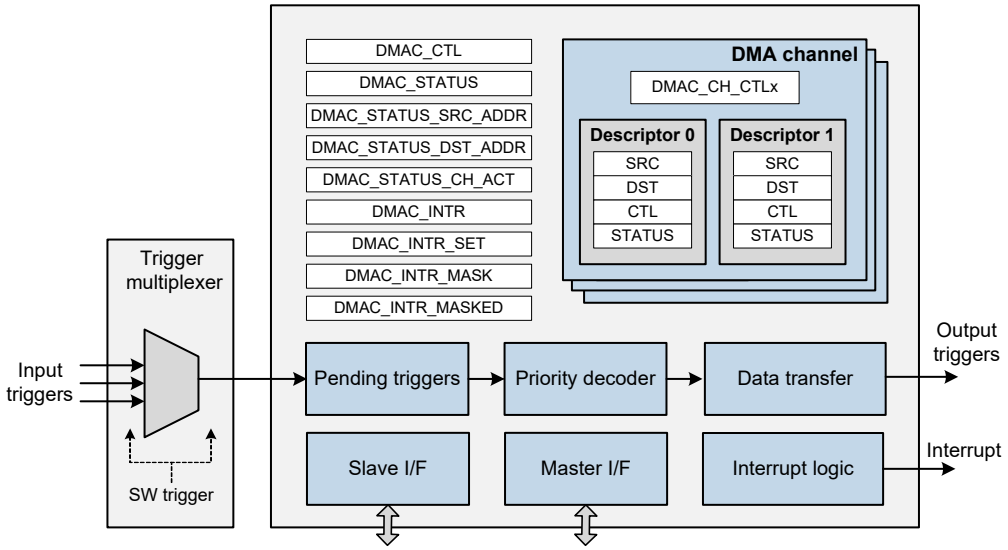
DMA 控制器支持活动 / 睡眠模式，但不支持深度睡眠和休眠功耗模式。

5.1 框图说明

DMA 组件使数据能够在存储器、外设和寄存器之间相互传输。这些传输不受 CPU 的控制。DMA 每次可传输多达 65,536 个数据元素。这些数据元素宽度可为 8 位、16 位或 32 位。通过来自 DMA 通道（包括 DMA 本身）、其他 DMA 通道、外设或 CPU 的外部触发器， DMA 可启动某个数据传输。 DMA 的最大优点是可以减轻 CPU 的数据传输负担。

图 5-1 显示的是 DMA 控制器的模块级别概述。

图 5-1. DMA 控制器模块框图



每个 DMA 通道具有两个描述符，用于配置传输专用的各种参数（如源地址、目标地址和数据宽度）。 DMA 通道通过触发事件启动传输。触发信号可来自器件中不同的外设（包括 DMA 本身）。

DMA 控制器具有两个总线接口：主设备接口和从设备接口。主设备 I/F 是一个 AHB-Lite 总线主设备，它允许 DMA 控制器将 AHB-Lite 数据传输到源位置和目标位置。DMA 是主设备接口中的总线主设备。通过该接口，可完成所有 DMA 传输。

通过从设备接口，可访问和重新配置 DMA 配置寄存器和描述符。从设备 I/F 是 AHB-Lite 总线从设备，它允许 PSoC 主 CPU 访问 DMA 控制器的控制 / 状态寄存器和描述符结构。一般情况下， CPU 是该总线的主设备。

接收某个触发器会激活 DMA 控制器中的状态机，经过特定的触发程序和处理过程后，它会根据描述符设置来启动数据传输操作。传输完成后会生成一个输出触发，可将它作为启动其他功能的触发条件或事件。

DMA 控制器中还包含一个中断逻辑模块。DMA 控制器中只有一个用于中断 CPU 的中断线。通过配置各个单独的 DMA 描述符，可在传输完成时激活该中断线。

5.1.1 触发源和复用

每个 DMA 通道都有一个与其相关的输入和输出触发。其中，输入触发可以来自任何外设、 CPU 或 DMA 通道本身。根据 5.2.4 传输模式小节的内容，可以使用输入触发器来触发DMA 传输。触发器输入处于 ‘ 逻辑高电平 ’ 状态时，它将触发 DMA 通道该 ‘ 逻辑高电平 ’ 的最小宽度为两个系统时钟周期。取消激活设置配置了触发器取消激活的性质。

传输操作完成时，触发器输出会发出确认信号。可以将该信号作为 DMA 通道的触发或数字互连的数字信号。触发输入可以来自不同的源，并通过 5.1.1.1 触发复用器路由。

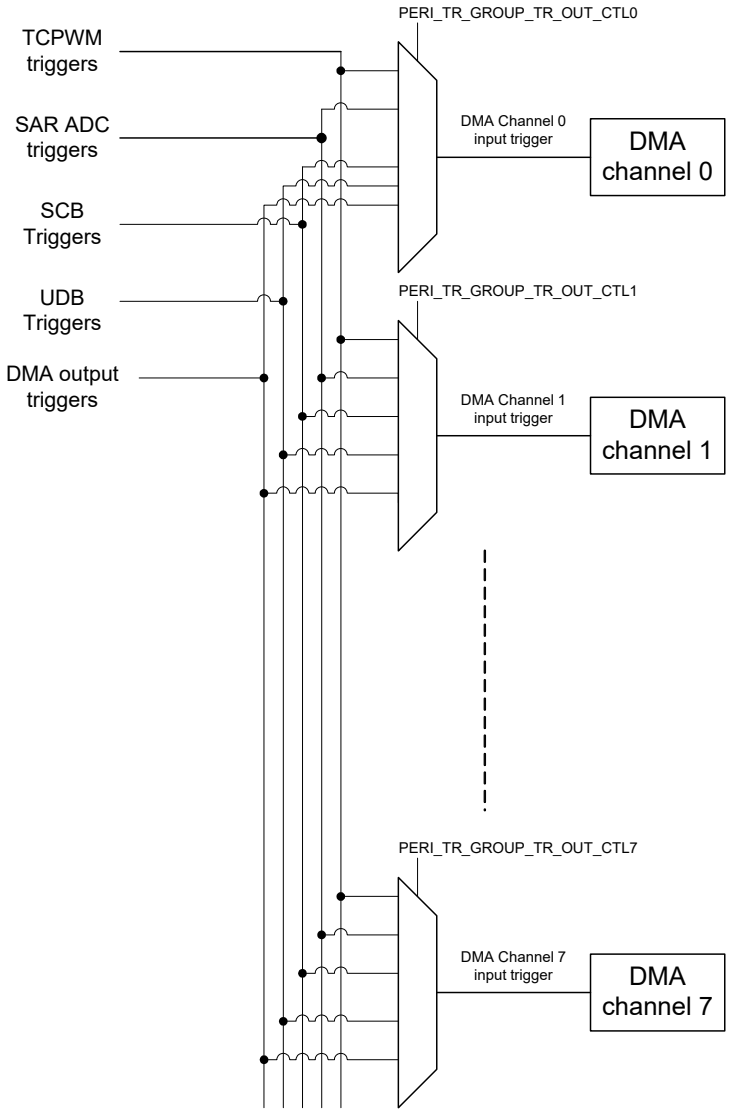
5.1.1.1 触发复用器

DMA 通道的触发输入可来自 PSoC 的不同外设源。通过触发复用器，可将 DMA 通道路由到单独的 DMA 通道的触发输入端。

在 DMA 触发器中，复用器被分为各个触发组。每个触发组由多个复用器组成，用于将数据写入到各个 DMA 通道的触发器输入端。

PSoC 4 实现一组单一的触发器 （触发器组 0），它为 DMA 提供触发器输入。触发输入选项可来自TCPWM、SAR ADC、SCB、 UDB 和 DMA 输出触发器。图 5-2 显示了触发复用器的实现。

图 5-2. 触发复用器的实现



通过配置 PERI_TR_GROUP_TR_OUT_CTLx[5:0] 寄存器，可以为单独的 DMA 通道选择触发源。表 5-1 提供了各个触发复用器。

表 5-1. 触发源

PERI_TR_GROUP_TR_OUT_CTLx[5:0]	触发源
0	软件触发被固定连接到逻辑 0
1	TCPWM 0 溢出
2	TCPWM 1 溢出
3	TCPWM 2 溢出
4	TCPWM 3 溢出
5	TCPWM 0 比较
6	TCPWM 1 比较

表 5-1. 触发源

PERI_TR_GROUP_TR_OUT_CTLx[5:0]	触发源
7	TCPWM 2 比较
8	TCPWM 3 比较
9	TCPWM 0 下溢
10	TCPWM 1 下溢
11	TCPWM 2 下溢
12	TCPWM 3 下溢
13	SAR ADC 输出
14	SCB0 TX
15	SCB0 RX
16	SCB1 TX
17	SCB1 RX

表 5-1. 触发源

PERI_TR_GROUP_TR_OUT_CTLx[5:0]	触发源
18	UDB DSI 请求 0
19	UDB DSI 请求 1
20	DMA 通道 0 触发
21	DMA 通道 1 触发
22	DMA 通道 2 触发
23	DMA 通道 3 触发
24	DMA 通道 4 触发
25	DMA 通道 5 触发
26	DMA 通道 6 触发
27	DMA 通道 7 触发

5.1.1.2 创建软件触发

每个 DMA 通道都有一个与其相关的触发输入和输出。该触发输入可以来自任何一个触发组，如第 44 页上的 “ 触发复用器 ” 一节所述。通过使用触发复用器设置中的触发输入选项 0，可以实现 DMA 通道的软件触发。当 PERI_TR_GROUP_TR_OUT_CTLx [5:0] 为 0 时， DMA 触发器被配置为软件触发器。然后，使用 PERI_TR_CTL 寄存器触发 DMA 通道。

5.1.2 挂起触发

如果在 DMA 通道工作时遇到触发事件，那么对应该触发的 DMA 通道将进入挂起状态。挂起触发将激活触发器局部存储在挂起位内，从而实现对它们的记录。该操作非常必要，因为可同时触发多个通道触发器，但每一次只能通过一个通道进行数据传输。通过该模块，可以使用电平敏感触发和沿触发。

挂起触发器被寄存在状态寄存器 （DMAC_STATUS_CH_ACT） 内。

5.1.3 输出触发

每个通道都有一个触发输出。该触发在两个系统时钟周期内处于高电平状态。在数据传输完成时将生成该触发。在系统级中，可将这些输出触发器连接到触发复用器组件上。通过该连接，可以将 DMA 控制器的输出触发连接到 DMA 控制器的输入触发上。换句话说，可通过完成单通道中的某个传输操作来激活其他通道，甚至重新激活同一通道。

请注意，DMA 输出触发器也被连接到数字系统互联（DSI），并且一些 DSI 信号被连接到触发器复用器输入。

5.1.4 通道优先级

如果有效的触发器使用了多个通道，那么可以根据通道优先级来确定可以访问数据传输引擎的通道。可以使用通道控制寄存器 （DMAC_CH_CTL） 的 PRIO 字段来设置每个通道的优先级，其中 ‘0’ 表示最高优先级， ‘3’ 表示最低优先级。优先级解码器根据通道优先级来确定优先级最高的有效通道。如果多个有效通道的优先级都为最高，则带有最低索引 ‘i’ 的通道被视为优先级最高的有效通道。

5.1.5 数据传输引擎

数据传输引擎用于将数据从源位置传输到目标位置。闲置时，数据传输引擎可以接收优先级最高的有效通道。该数据传输的配置由描述符指定。数据传输引擎实现了一个拥有以下状态的状态机。

- 状态 0 — 默认状态：这是 DMA 控制器的闲置状态。在该状态中，控制器需要等待触发条件，从而才能进行传输。
- 状态 1 — 负载描述符：在发生触发条件并确定好优先级时，数据传输引擎将进入负载描述符状态。在该状态中，有效描述符 （SRC、 DST 和 CTL） 将被加载到 DMA 控制器内，以开始进行传输。DMAC_STATUS、DMAC_STATUS_SRC_ADDR、DMAC_STATUS_DST_ADDR 以及 STATUS_CH_ACT 反映的是当前的有效状态。
- 状态 2 — 加载源位置的数据：数据传输引擎通过主设备 I/F 来加载源位置的数据。
- 状态 3 — 将数据存储在目标位置：数据传输引擎通过主设备 I/F 将数据存储到目标位置。

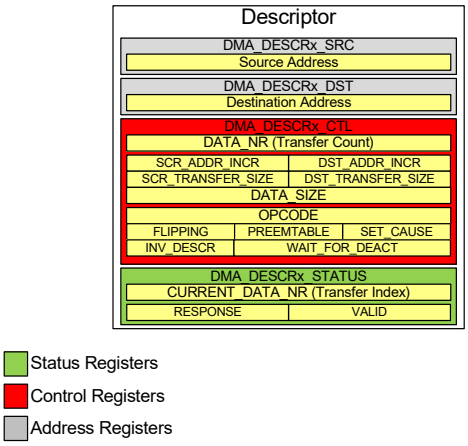
  根据传输模式，可以多次执行状态 2 和 3。
- 状态 4 — 存储描述符：数据传输引擎通过更新通道的描述符结构来反映数据传输，并将其存储在描述符状态寄存器内。
- 状态 5 — 等待触发器取消激活状态：如果触发器取消激活状态需要两个周期，那么在触发器保持有效状态两个周期后便满足该条件。如果该状态被设置为 ‘wait indefinitely’，则 DMA 控制器将保持在该状态下，直到触发信号进入低电平状态为止。
- 状态 6 — 存储描述符响应：在该相位中，将完成根据描述符的数据传输，并且生成中断（如果该中断被配置，已在该条件中生成）。DMAC_DESCR_PING_STATUS 或 DMAC_DESCR_PONG_STATUS 寄存器中的响应字段将被填充，并且其状态返回到状态 0。

5.2 描述符

通道中源地址和目标地址间的数据传输是由描述符配置的。DMA 中的每个通道都有两个称为 PING 和 PONG 的描述符（本文档将它们分别称为描述符 0 和描述符 1）。单个描述符指的是4个32位寄存器的组合，它包含相关通道的传输配置。

图 5-3 显示的是描述符的结构。

图 5-3. 描述符结构



5.2.1 地址配置

图5-4 演示了某个传输的地址配置的描述符设置的使用情况。

Source and Destination Address（源地址和目标地址）：源地址和目标地址在描述符中的相应寄存器内设置。它们为该传输的源位置和目标位置设置了基地址。通过配置描述符来传输单个元素时，该字段将保存该数据元素的源 / 目标地址。如果描述符在递增模式下使用源地址和 / 或目标地址来传输多个元素，该字段将保存被传输的第一个元素的地址。

Data Number (DATA_NR)（数据数量）：这是一个传输计数参数。 DATA_NR 是一个 16 位的编号，它确定了在一个描述符完成前将要传输的元素数量。在典型的使用中，它设置的是传输的缓冲器大小。

Source Address Increment (SCR_ADDR_INC)（源地址递增）：这是控制寄存器中的位设置，它确定了源地址是否在每个数据元素传输之间递增。当数据源为缓冲器并且需要从存储器中后续位置提取传输元素时，需要使能该特性。在这种情况下，源地址寄存器只设置基地址，并且该基地址上的后续传输均是递增的。地址递增的大小要根据第 48 页上的 5.2.2 传输大小章节中 SCR_TRANSFER_SIZE 设置确定。

Destination Address Increment (DST_ADDR_INC)（目标地址增量）：这是在控制寄存器中设置的位，通过它可以确定目标地址是否在每个元素传输之间递增。当数据目标为缓冲器，并且需要将每个传输元素传输到存储器中后续位置时，需要使能该特性。在这种情况下，目标地址寄存器只设置基地址，并且该基地址上的后续传输将递增。地址递增的大小要根据第 48 页上的 5.2.2 传输大小章节中的 DST_TRANSFER_SIZE 设置确定。

Invalidate Descriptor (INV_DESCR)（无效描述符）：设置该位时，描述符将传输所有数据元素，并清除描述符中的 VALID 位，从而使其无效。该特性会对 DMA_DESCRx_STATUS 寄存器中的 VALID 位产生影响。如果用户期待描述符在传输完成后无效，可以使用该设置。通过设置描述符的 STATUS 寄存器中的 VALID 位，可以使固件中的描述符反回到有效状态。

Preemptable (PREEMPTABLE)：如果禁用该位，工作模式所定义的当前传输则能够完成而不受干扰。如果使能该位，优先级更高的 DMA 通道可以阻止 / 中断工作模式所定义的当前传输。当阻止该通道时，该位将被挂起并在下次它的优先级最高时运行。

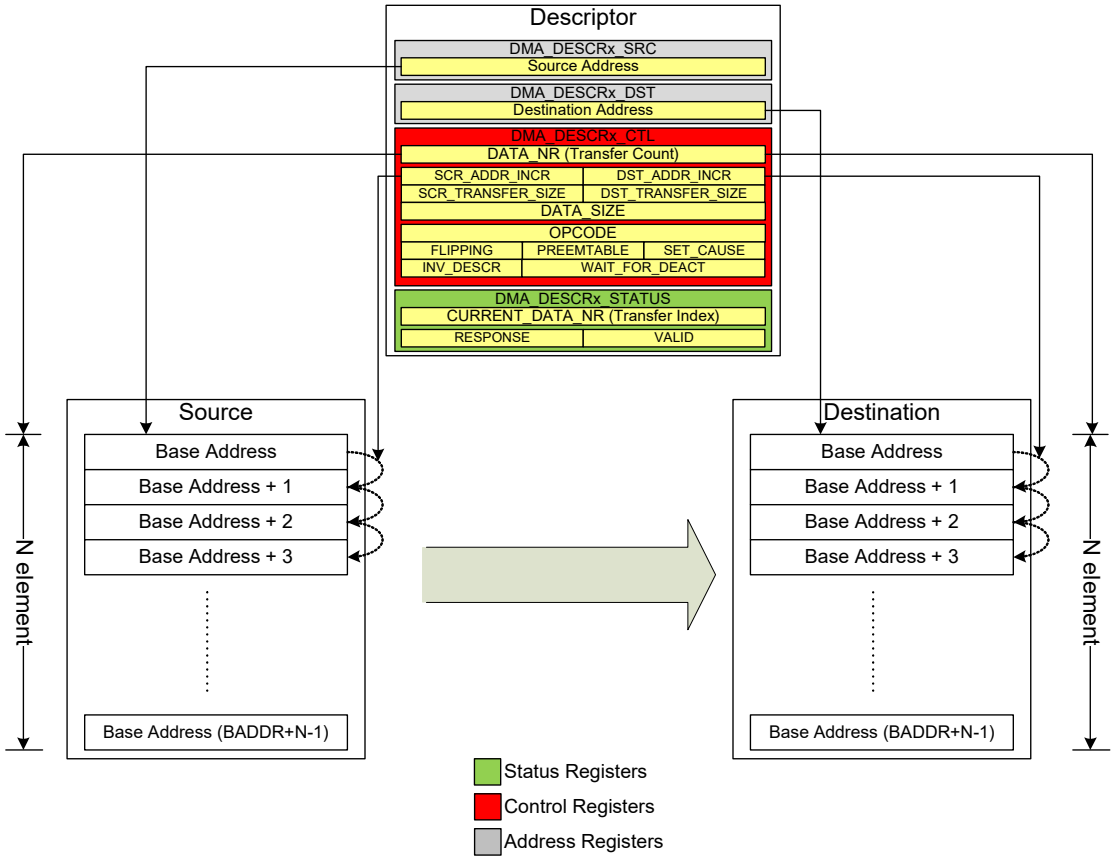
Setting Interrupt Cause (SET_CAUSE)（设置中断原因）：当描述符完成传输所有数据元素时，它将生成一个中断请求。该中断请求被所有 DMA 通道共享。设置该位会使与之相应的通道变成中断源。

Trigger Type (WAIT_FOR_DEACT)（触发类型）：基于描述符的 DMA 传输完成时，数据传输引擎会检测触发器是否取消激活状态。该步骤与数据传输引擎的状态 5 相对应。请参见第 46 页上的 5.1.5 数据传输引擎。根据 DMA 输入触发的类型，可以确定触发信号被取消激活的时间。触发器被激活时，DMA 传输将被激活，但这种传输只能在触发器处于取消激活状态后完成。可以使用该字段将控制器中数据的传输与生成该触发的源同步。

只有在完成一个描述符执行后使用该字段，它具有以下的四种设置值：

- 0 — 脉冲触发：无需等待取消激活。
- 1 — 电平敏感触发器等待四个SYSCLK周期：在四个周期内检测到电平触发器信号后，将禁用 DMA 触发。
- 2 — 电平敏感触发器等待八个SYSCLK周期：在八个周期内检测到电平触发器信号后，将启动 DMA 传输。
- 3 — 脉冲触发器将无限期等待，直到被取消激活为止。取消激活触发器信号后，将启动 DMA 传输。

图 5-4. DMA 传输：地址配置



5.2.2 传输大小

通过使用描述符中的传输 / 数据大小参数，可以为某个传输配置传输字宽度。这些设置被多样化为源传输大小、目标传输大小和数据大小。数据大小参数（DATA_SIZE）为该传输设置总线宽度。由SCR_TRANSFER_SIZE和DST_TRANSFER_SIZE 设置的源和目标传输大小，该值是 DATA_SIZE 或 32 位。可以将 DATA_SIZE 设置为 32 位、 16 位或 8 位。

大多数 PSoC 4 外设寄存器的数据带宽都是 4 字节（32 位），因此，当 DMA 将外设作为其源地址或目标地址时，SCR_TRANSFER_SIZE 或 DST_TRANSFER_SIZE 通常被设置为 32 位。不管需要移植的数据带宽多大， DMA 组件的源和目标地址的传输大小都必须与源和目标地址的编址宽度相匹配。DATA_SIZE 参数与实际数据的宽度相称。例如，如果将 16 位 PWM 用作 DMA 数据的目标地址，那么必须将 DST_TRANSFER_SIZE 设置为 32 位，以符合 PWM 寄存器的宽度匹配，因为 TCPWM 模块（以及大部分 PSoC 4 外设）的外设寄存器宽度始终为 32 位。但是，在该示例中，还可以将目标的 DATA_SIZE 参数设置为 16 位，因为 16 位 PWM 只使用大小为两字节的数据。 SRAM 和闪存的编址宽度都是 8 位、16 位或 32 位，并且可以选择任意源和目标地址的传输大小，以满足该应用的需要。

表 5-2 汇总了传输大小设置及其说明的组合。

表 5-2. 传输大小设置

DATA_SIZE	SCR_TRANSFER_SIZE	DST_TRANSFER_SIZE	典型用法	说明
8 位	8 位	8 位	存储器到存储器	无数据操作
8 位	32 位	8 位	外设到存储器	源位置中的高 24 位被清除
8 位	8 位	32 位	存储器到外设	目标位置中添加了高 24 位零
8 位	32 位	32 位	外设到外设	源位置中清除了高 24 位，并且标位置中添加了高 24 位零
16 位	16 位	16 位	存储器到存储器	无数据操作

表 5-2. 传输大小设置

DATA_SIZE	SCR_TRANSFER_SIZE	DST_TRANSFER_SIZE	典型用法	说明
16 位	32 位	16 位	外设到存储器	源位置中清除了高 16 位
16 位	16 位	32 位	存储器到外设	目标位置中添加了高 16 位零
16 位	32 位	32 位	外设到外设	源位置中清除了高 16 位，并且标位置中添加了高 16 位零
32 位	32 位	32 位	外设到外设	无数据操作

5.2.3 描述符链接

每个通道具有一个 PING 和一个 PONG 描述符，它带有用于相关传输的不同设置。有效描述由单独通道控制寄存器（DMAC_CH_CTL）中 PING_PONG 位的设置。通过 PING 和 PONG 描述符的功能，可以创建描述符的链接列表。这样不需要 CPU 干预也可以从一个传输配置转换到另一个配置。另外，两个描述符意味着当 PONG 寄存器有效时 CPU 可以随时修改 PING 寄存器，反之亦然。

当描述符中的 FLIPPING 位被使能时，它将被链接至 PING/PONG 的副本。可以将该字段与操作码 2 传输模式一起使用。因此，使能 PING 描述符中的 FLIPPING 位并配置该位，将其用于操作码 2 后，该通道将在 PING 描述符末尾自动执行 PONG 描述符。如果配置该位，使其用于操作码 0 或操作码 1，那么需要使用新的触发器来启动 PONG 描述符。

应该在各种传输模式下使用 PING PONG。

5.2.4 传输模式

在执行描述符期间，通道的操作是由操作码设置定义的。这三个操作码适用于 DMA 控制器的所有通道。

5.2.4.1 每个触发器一个单数据元素（操作码 0）

当配置操作码为 0 时，可以实现该模式。每次收到触发信号时，DMA 会将单个数据元素从源位置传输到目标位置中。可以将该功能与描述符中的其他设置 （如源和目标递增）一起使用。

图 5-5 显示的是该传输典型的使用情况。在该图中， ADC 数据寄存器作为源地址，其目标地址是一个外设寄存器 （如 PWM 比较）。该触发来自 ADC 的结束转换信号。当收到触发时，传输引擎将从 ADC 中加载数据，并将低 16 位数据存储在 PWM 寄存器内。因为描述符被重新运行，所以连续触发事件可导致相同的行为。注意：要掌握如何将源和目标地址的数据宽度设置为 32 位。这是因为对 PSoC 中外设寄存器进行的所有访问都必须是 32 位的。由于有效的数据宽度只有 16 位，因此必须保持 DATA_SIZE 为 16 位。

图 5-5. 操作码 0：各外设间简单的 DMA 传输

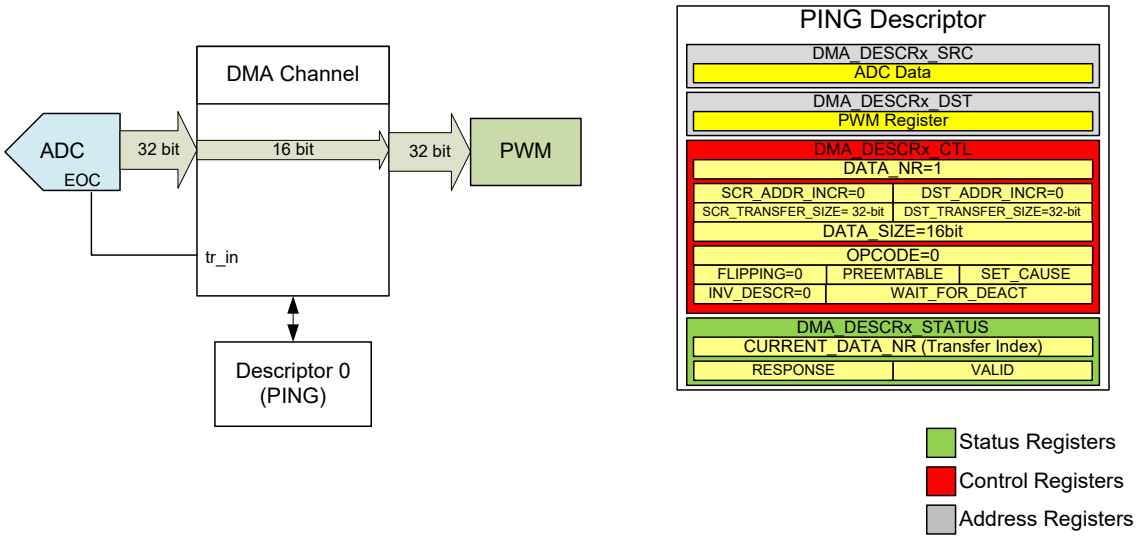


图 5-6 说明了 ADC 数据寄存器和缓冲器之间的数据传输的另一个使用情况。该情况显示的是一个 PING 描述符，从源位置（ADC）提取数据时，该寄存器会递增目标。当收到触发器时，传输引擎将从 ADC 位置加载数据，并将其存储到存储器缓冲区内（Sample 1 的存储位置）。后续触发器继续将 ADC 数据存储到 Sample 1 后连续的位置，直到 PING 描述符缓冲区（DATA_NR 字段）被填满为止。

图 5-6. 操作码 0：使用目标地址递增功能的数据传输

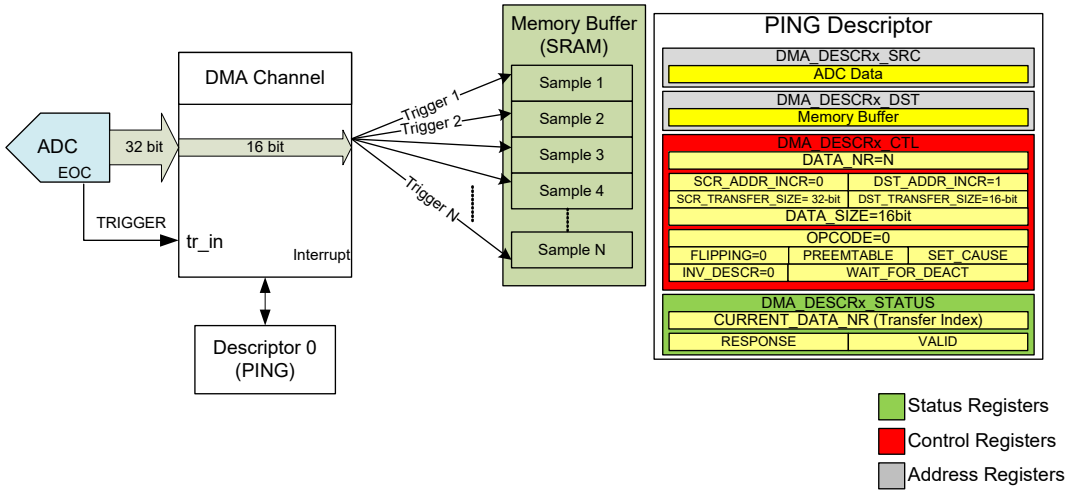
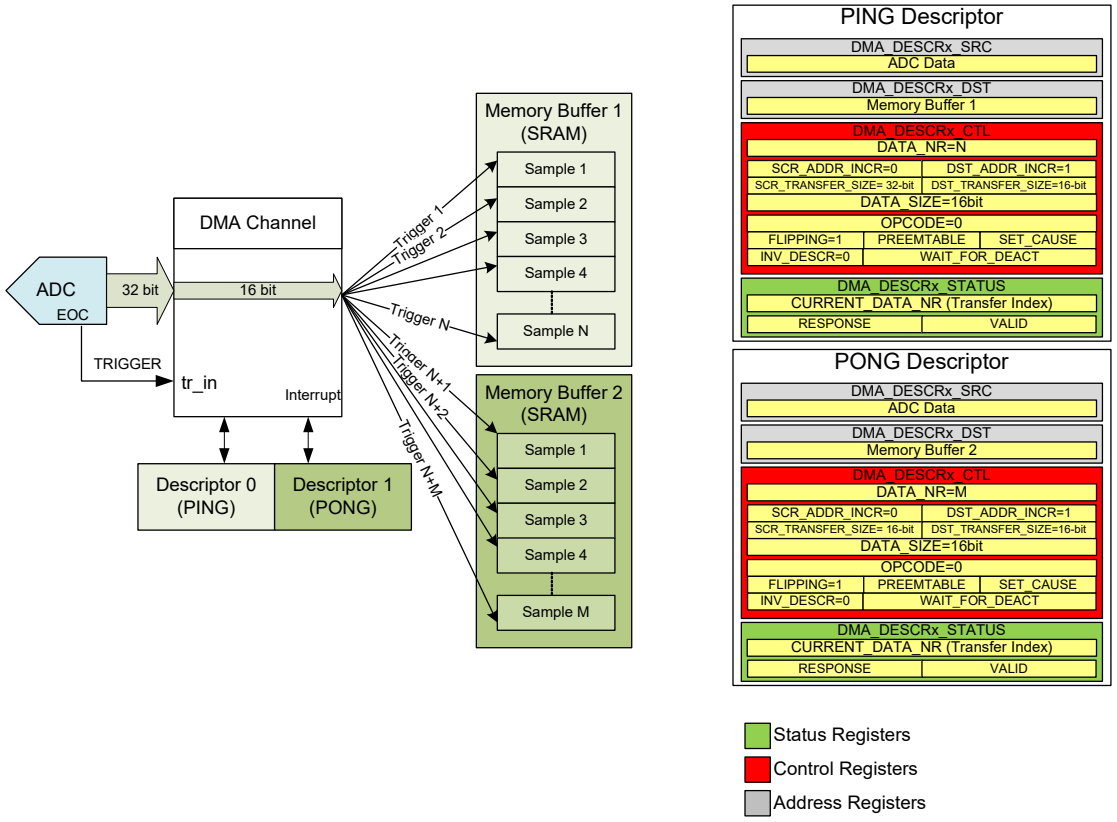


图 5-7 显示的是相同的使用情况。它演示了 PING 和 PONG 描述符的使用情况。完成 PING 描述符后，控制器将翻转，以执行 PONG 描述符。请注意，该翻转位是在描述符中被使能的；该操作使能了描述符的链接。如果翻转位未被使能，那么将重新运行相同的描述符。因此，该序列进行了两次缓冲器传输。另外请注意，传输器一次触发传输完一个元素后才能进行下一次触发传输。

图 5-7. 操作码 0：使用翻转功能的传输

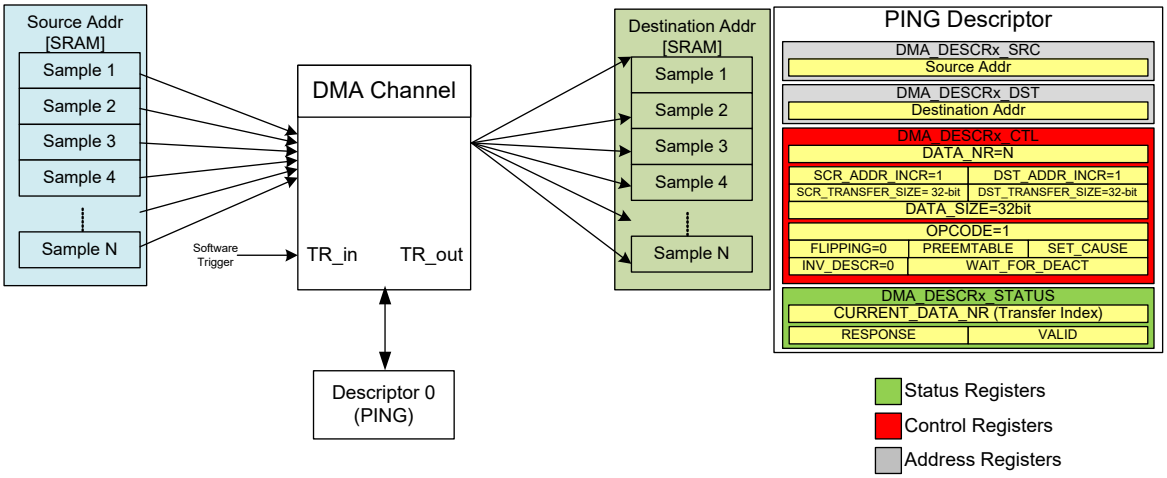


5.2.4.2 *每次触发传输整个描述符（操作码 1）*

在该操作模式下，在一个触发器中，DMA 能够从源位置将多个数据元素传输到目标位置。如果使用操作码 1，那么控制器将在单个触发器中执行整个描述符。在存储器至存储器缓冲器的传输中，该功能很有用。当遇到触发条件时，将连续进行传输，直到整个描述符被传输完成为止。

图 5-8 显示的是操作码 1 传输，即将源缓冲器中全部内容传输到目标缓冲器内。整个传输是单个 PING 描述符的一部分，并在执行单个触发器时完成。

图 5-8. 使用操作码 1 的 DMA 传输示例

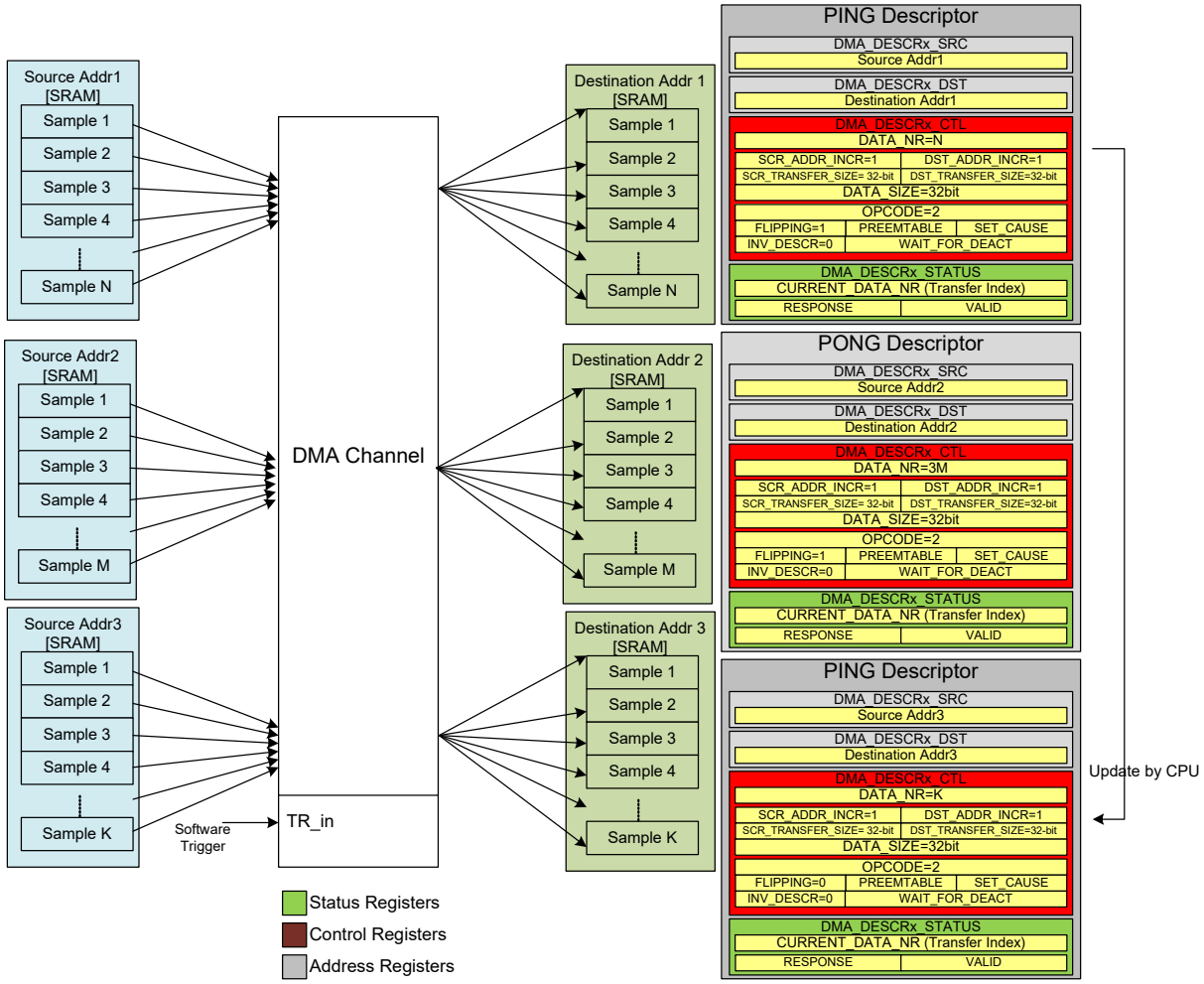


5.2.4.3 每个触发传输整个描述符链接（操作码 2）

操作码 2 始终与 FLIPPING 字段一起使用。将操作码 2 与 PING 描述符中的 FLIPPING 一起使用时，单个触发器可以执行 PING 描述符，并自动翻转到 PONG 描述符，从而能够按同样的方式执行该描述符。如果 PONG 描述符带有操作码 2，则从 PING 到 PONG 之间的周期将一直持续，直到 CPU 更改描述符或它们不再有效为止。

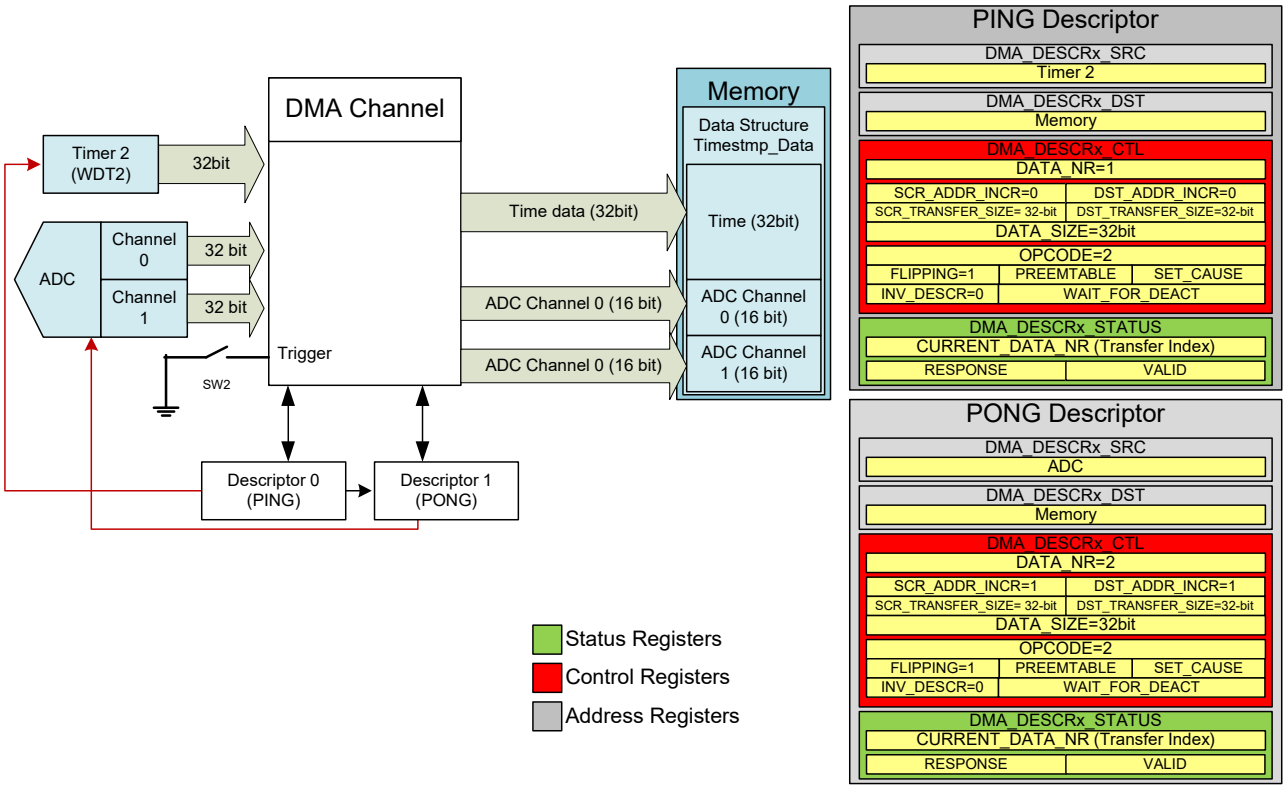
图 5-9 显示的是配置 PING 和 PONG 描述符，使之用于操作码 2 的情况。当执行 PING 寄存器的第二个迭代时， CPU 将禁用 FLIPPING。

图 5-9. 使用操作码 2 的 DMA 传输示例



通过制定操作码 2 的传输模式，可以实现不同的使用情况。图 5-10 显示了其中一种。在该图中，源地址数据可来自不同且不连续的存储位置。目标地址的数据结构：连续的存储位置。一个源地址为定时器 Timer 2（保持时序数据），另一个源地址为一个 ADC。这两个地址的数据被存储在存储器中连续的位置。

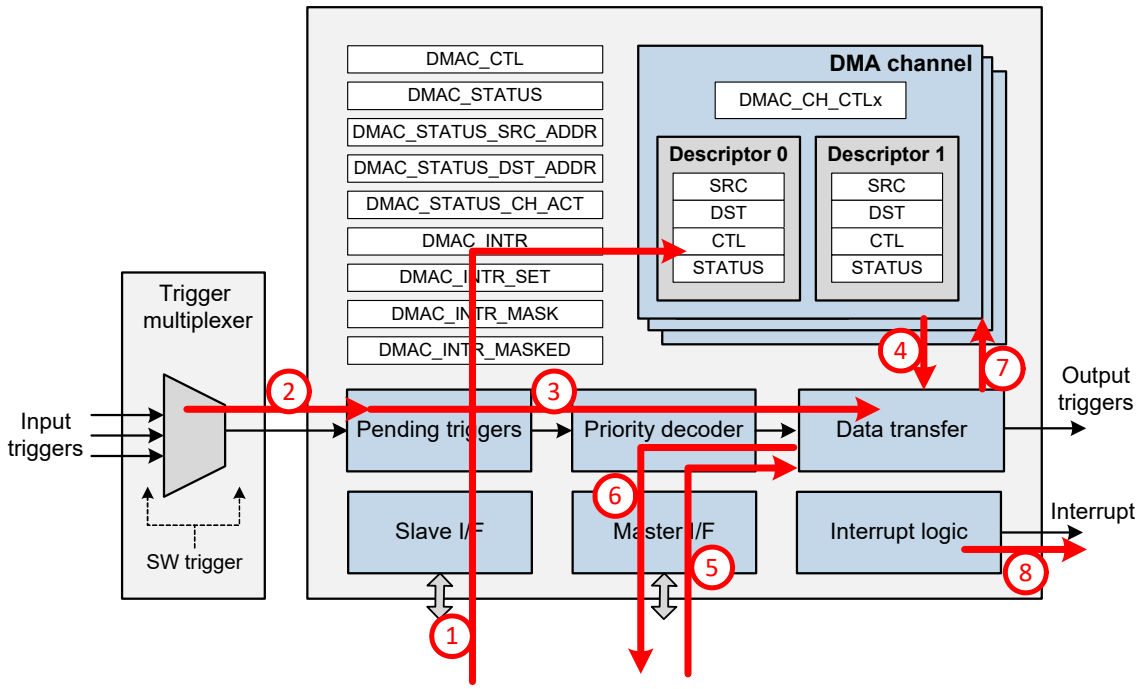
图 5-10. 操作码 2：从多个源地址传输到存储器



5.3 操作与时序

图 5-11 显示的是包含了一个触发器、数据或上面叠加了中断流的 DMA 控制器设计。

图 5-11. 操作流程



该流程图显示的是 DMA 控制器的数据传输的各个步骤。

1. 主 CPU 为特定的通道编程描述符结构。它也对 DMA 寄存器进行编程，从而为该通道选出特定的系统触发器。
2. 激活通道的系统触发器。
3. 通过优先级解码来确定优先级最高的有效通道。
4. 数据传输引擎接受有效的通道，并使用通道标识符来加载通道的描述符结构。描述符结构指定了通道的数据传输方式。
5. 数据传输引擎通过主设备 I/F 进行加载源位置中的数据。
6. 数据传输引擎通过主设备 I/F 将数据存储到目标位置。在单个元素 （操作码 0）传输中，步骤 5 和 6 只能执行一次。在多个元素描述符 （操作码 1 或 2）的传输中，可以按顺序地多次执行步骤5和6，以实现多次数据元素传输。
7. 数据传输引擎通过更新通道的描述符结构来反映数据传输，并将其存储在描述符的 SRAM 内。
8. 如果由描述符通道结构指定的所有数据传输已经完成，会生成一个中断 （这是可编程选项）。

DMA 控制器的数据传输步骤分三种：初始化、并发、或连续。

- 初始化：指的是步骤 1，用于编程描述符的结构。所有的描述符结构都执行该步骤。它由主 CPU 执行，并不能通过有效的通道触发器进行激活。
- 并发：包括步骤 2 和 3。需要对每个通道并行执行这两步。
- 连续：包括步骤4到8。需要对每个有效通道连续执行这些步骤。这样，您可以通过执行这些步骤所占用的时间来确定 DMA 控制器的吞吐量。时长包含两个因素：控制器加载和存储描述符的时间以及在总线设备上传输数据的时间。后续时间则取决于总线延迟（由仲裁器和桥接器组件确定）和目标存储器 / 外设。

传输单个数据元素时，需要 12 个时钟周期来完成一个完整的传输 （假定 AHB-Lite 总线不处于等待状态）。在该模式中，可按照下面的公式计算出完成一次传输所需的时钟周期：

时钟周期数量 = 12 + 加载等待状态 + 存储等待状态

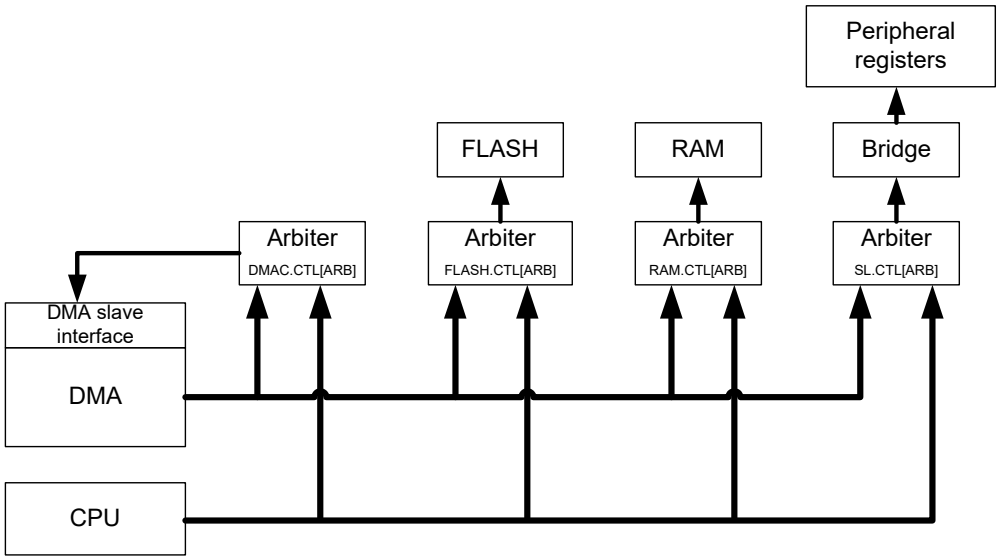
传输所有描述符或连接描述符链路时，需要 12 个时钟周期用于传输第一个数据元素。后续元素需要三个周期。该结果是在 AHB-Lite 走线不处于等待状态的条件下得出的。对于 ‘N’ 元素的传输，请按照下面公式计算所需时钟周期：

时钟周期数量 =（12 + 加载等待状态 + 存储等待状态）+（N–1） *（3 + 加载等待状态 + 存储等待状态）

5.4 仲裁

AHB 总线包含两个主设备：CPU 和 DMA 控制器。所有外设和存储器都通过从设备接口连接到总线上。其中，闪存存储器和 RAM 具有专用的从设备接口，这些接口具有自己的仲裁器。所有外设寄存器都是通过一个专用仲裁器中的桥接器连接到单个从设备接口的。DMA 控制器的从设备接口（用于访问 DMA 控制器的控制寄存器）通过其他从设备接口进行连接。图 5-12 显示的是该架构。

图 5-12. PSoC 4 总线架构

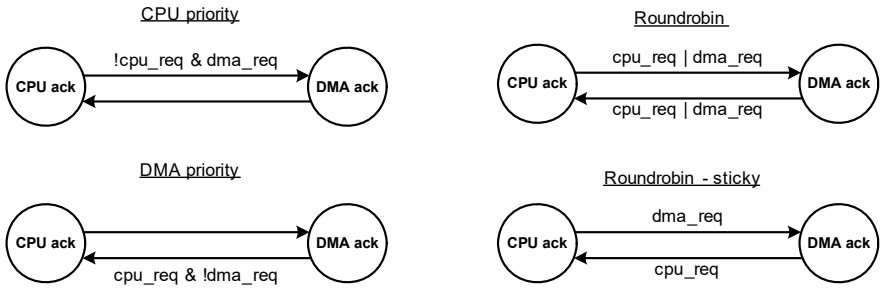


每个从设备可以使用以下任何一种仲裁方式：

- CPU 优先：CPU 的仲裁优先级始终最高。只有不存在 CPU 的请求时，才会允许 DMA 的访问。
- DMA 优先级：DMA 的仲裁优先级始终最高。只有不存在 DMA 请求时，才会允许 CPU 的访问。
- 轮循：执行每个请求时，仲裁优先级连续在 DMA 和 CPU 之间切换。每次请求 （CPU 或 DMA）中仲裁优先级都会切换。
- 粘性轮循 ： 该模式类似于轮循模式，但它仅在接收到更低优先级主设备的请求时切换优先级。例如，如果当前的优先级是 CPU，而 DMA 发出了一个请求，那么优先级将在下一个请求中交给 DMA。如果 DMA 没有发出任何请求，则 CPU 会保持当前的优先级。

下图详细介绍了各个仲裁模型。

图 5-13. 仲裁模型



5.5 寄存器列表

寄存器名称	说明	特性
DMAC_CTL	模块控制	用于使能 DMA 控制器的位。
DMAC_STATUS	模块状态	提供了有关 DMA 控制器的状态信息。
DMAC_STATUS_SRC_ADDR	当前源地址	提供了当前加载的源地址的相关信息。
DMAC_STATUS_DST_ADDR	当前目标地址	提供了当前加载的目标地址的相关信息。
DMAC_STATUS_CH_ACT	通道有效状态	软件通过读取该字段获取所有挂起通道的信息 （该信息处于挂起状态或位于数据传输引擎内）。
DMAC_CH_CTLx	通道控制寄存器	为通道 x 提供通道使能、 PING/PONG 和优先级设置等操作。
DMAC_DESCRx_PING_SRC	PING 源地址	通道 x 源位置的基地址。
DMAC_DESCRx_PING_DST	PING 目标地址	通道 x 目标位置的基地址。
DMAC_DESCRx_PING_CTL	PING 控制字	PING 描述符的所有控制设置。
DMAC_DESCRx_PING_STATUS	PING 状态字	有效性、应答和实时 Data_NR 索引状态。
DMAC_DESCRx_PONG_SRC	PONG 源地址	通道 x 源位置的基地址。
DMAC_DESCRx_PONG_DST	PONG 目标地址	通道 x 目标位置的基地址。
DMAC_DESCRx_PONG_CTL	PONG 控制字	PONG 描述符的所有控制设置。
DMAC_DESCRx_PONG_STATUS	PONG 状态字	有效性、应答和实时 Data_NR 索引状态。
DMAC_INTR	中断寄存器	
DMAC_INTR_SET	中断设置寄存器	读取时，该寄存器反映了中断请求寄存器的状态。
DMAC_INTR_MASK	中断屏蔽	INTR 寄存器中相应字段的屏蔽。
DMAC_INTR_MASKED	中断屏蔽寄存器	读取时，该寄存器反映了中断请求和屏蔽寄存器之间的按位逻辑与（AND）运算后的结果。该寄存器允许软件通过单次加载操作来读取所有屏蔽使能中断的原因，而不需要通过两次加载操作：一个用于中断原因，一个用于屏蔽。这样便简化了固件的开发过程。

6. 中断

PSoC® 4 中的 ARM Cortex-M0 (CM0) CPU 支持中断和异常事件。中断是指由 CPU 外围设备（如定时器、串行通信模块和端口引脚信号）生成的事件。异常事件是指由 CPU 生成的事件（如存储器访问故障和内部系统定时器事件）。中断和异常事件都会中止当前进行的程序流程，同时 CPU 将执行中断服务子程序或异常事件处理程序。器件为中断处理程序 /ISR 和异常事件处理程序提供了统一的异常向量表。

6.1 特性

PSoC 4 支持下面各项中断特性：

- 支持 32 个中断
- CPU 内核中集成了嵌套向量中断控制器（NVIC），从而得到较短的中断延迟
- 可将向量表保存在闪存或 SRAM 内
- 每个中断的可配置优先级范围为 0 到 3
- 支持电平触发和边沿触发的中断信号

6.2 工作原理

图 6-1. PSoC 4 中断框图

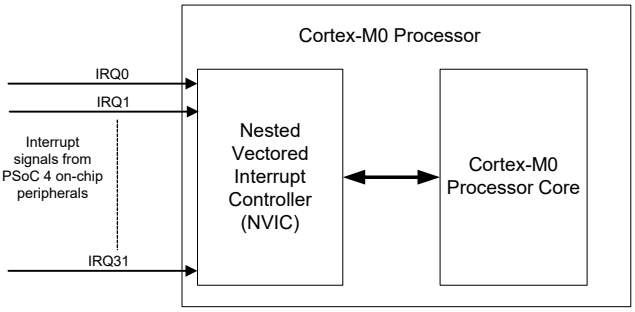


图 6-1 显示的是中断信号与 Cortex-M0 CPU 之间的交互。PSoC 4 一共有 32 个中断；这些中断均由 NVIC 处理。NVIC 分别用于使能 / 禁用单独中断、优先级处理，和与 CPU 内核通信。与 CPU 的外围设备所生成的中断不同，异常事件是由 CM0 内核生成的，因此，它们并未显示在图 6-1 中。

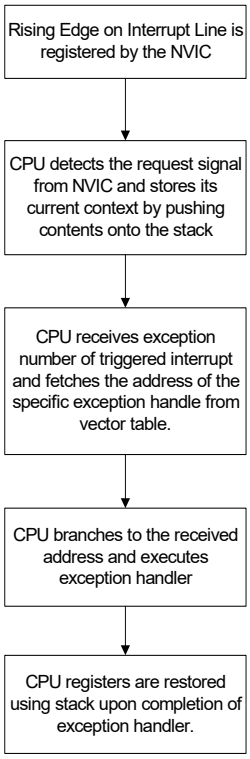
6.3 中断 / 异常事件及其操作

6.3.1 中断 / 异常事件处理

触发某个中断或异常事件时，会发生下面一系列事件：

1. 假设所有中断信号的初始状态均为低电平（闲置或非活动状态），并且处理器正在执行主代码，那么 NVIC 会记录所有中断信号线的上升沿。这时，该中断线处于挂起状态，直至 CPU 开始处理该中断为止。
2. 当检测到来自 NVIC 的中断请求信号时，CPU 会将其寄存器中的内容推放到堆栈上，以保存它的当前上下文内容。
3. 另外， CPU 还接收来自 NVIC 的触发中断的异常事件编号。所有中断和异常事件均有唯一的异常事件编号，如表 6-1 所示。通过使用该异常事件编号，CPU 可以从向量表中加载特定异常事件处理程序的地址。
4. 然后，CPU 转至该地址，并执行相应的异常事件处理器。
5. 在完成执行异常事件处理程序后，CPU寄存器会通过堆栈弹出操作恢复到它的原始状态，同时 CPU 也会恢复执行主代码。

图 6-2. 触发中断时的中断处理



如果在执行某个中断时，NVIC 接收到另一个中断请求，或它同时接收到多个中断请求， NVIC 会评估这些中断的优先级，然后向 CPU 发送优先级最高的中断所对应的异常事件编号。因此，优先级高的中断可以随时中断优先级低的ISR的执行。

异常事件的处理方法与中断的处理方法相同。每个异常事件均有唯一一个异常事件编号。 CPU 会根据该编号执行相应的异常事件处理程序。

6.3.2 电平与边沿触发中断

NVIC 支持中断信号线 （IRQ0 至 IRQ31）上的电平和边沿信号。中断的类型 （电平或边沿触发）是由中断源决定的。

图 6-3. 电平触发中断

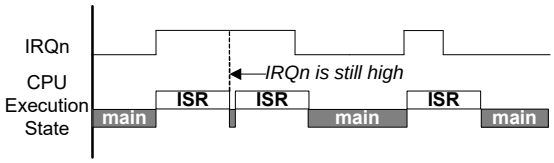


图 6-4. 边沿触发中断

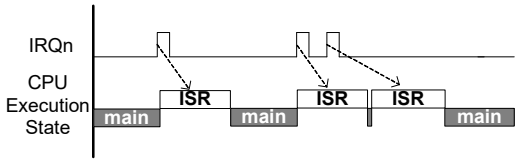


图 6-3 和图 6-4 分别显示的是工作中的电平中断和边沿中断。假设中断信号最初为非活动状态（逻辑低），那么以下各个事件序列说明了如何处理电平触发中断和边沿触发中断：

1. 在中断信号的上升沿事件中， NVIC 记录了中断请求。这时，中断处于挂起状态，即 CPU 尚未处理该中断请求。
2. 然后， NVIC 会将异常事件编号和中断请求信号一同发送到 CPU。当 CPU 开始执行中断服务子程序 （ISR）时，将清除中断的挂起状态。
3. 当 CPU 执行 ISR 时，一个或多个中断信号的上升沿将被记录，并被视为单一的挂起请求。在完成执行当前的 ISR 后，将再次处理挂起中断 （请查看图 6-4，了解边沿触发中断的相关信息）。
4. 执行完 ISR 后，如果中断信号仍为高电平，则该信号会处于挂起状态，并且重新执行 ISR。图 6-3 说明了电平触发中断的情况，只要中断信号为高电平，将始终执行 ISR。

6.3.3 异常事件向量表

异常事件向量表 （表 6-1）保存的是所有异常事件处理器的入口点地址。 CPU 会根据异常事件的编号来加载相应的地址。

表 6-1. 异常事件向量表

异常编号	异常事件	异常优先级	向量地址
–	初始堆栈指针值	不适用 （NA）	Base_Address — 可以为 0x00000000 （闪存存储器的起始地址），或者为 0x20000000 （SRAM 的起始地址）
1	复位	–3，最高优先级	Base_Address + 0x04
2	不可屏蔽中断 （NMI）	–2	Base_Address + 0x08
3	硬故障	–1	Base_Address + 0x0C
4-10	保留	NA	Base_Address + 0x10 至 Base_Address + 0x28
11	管理程序调用 （SVCall）	可配置 （0 - 3）	Base_Address + 0x2C
12-13	保留	NA	Base_Address + 0x30 至 Base_Address + 0x34
14	挂起管理程序 （PendSV）	可配置 （0 - 3）	Base_Address + 0x38
15	系统定时器 （SysTick）	可配置 （0 - 3）	Base_Address + 0x3C
16	外部中断 （IRQ0）	可配置 （0 - 3）	Base_Address + 0x40
…	…	可配置 （0 - 3）	…
47	外部中断 （IRQ31）	可配置 （0 - 3）	Base_Address + 0xBC

在表 6-1 中，没有将第一个字（4 字节）标记为异常事件编号零。这是因为异常事件表中的第一个字用于在器件复位时初始化主堆栈指针（MSP）的值。该值不被视为一个异常事件。在 PSoC 4 中，可通过配置向量表，使其位于闪存存储器（起始地址为 0x00000000），或位于 SRAM （起始地址为 0x20000000）。通过对 CPUSS_CONFIG 寄存器中的 VECT_IN_RAM 位字段 （位 0）进行写操作，可以实现该配置。当VECT_IN_RAM位字段为‘1’时，CPU能够从SRAM中的向量表位置获取异常事件处理程序的地址。当该位字段为 ‘0’ （复位状态）时，可通过使用闪存存储器中的向量表获取异常事件的地址。您必须将 VECT_IN_RAM 位字段设置为器件引导代码的一部分，以将向量表保存到 SRAM 内。将向量表存储在 SRAM 中的优点是：通过修改 SRAM 向量表的内容，可以动态更改异常事件处理程序的地址。然而，必须通过对闪存存储器进行的写操作才能更改非易失性闪存存储器的向量表。

如果未设置 CPUSS_SYSREQ 寄存器中的 NO_RST_OVR 位，那么在读取 0x00000000 和 0x00000004 闪存地址时会重新指向 SROM 的前八个字节，以加载堆栈指针和复位向量。要想读取 0x00000000 和 0x00000004 闪存地址，需要将 NO_RST_OVR 位设置为 ‘1’。堆栈指针向量将保持复位时堆栈指针所加载的地址。复位向量保持引导序列的地址。当器件完成复位时，执行该映射以使用 SROM 上堆栈指针和复位向量的默认地址。在复位过程中，先执行 SROM 中的启动代码，然后 CPU 跳转到闪存中的 0x00000004 地址，以执行闪存中的处理程序。复位时， VECT_IN_RAM 为 0，因此从未使用 SRAM 向量表中的复位异常事件地址。

此外，设置 CPUSS_SYSREQ 寄存器的 SYSREQ 位时，对闪存地址 0x00000008 的读取将指向 SROM（而不是闪存），以加载 NMI 向量地址。复位 CPUSS_SYSREQ 可以读取闪存中的地址 0x00000008。

6.4 异常事件源中说明了各个异常事件源（异常事件编号 1 到 15）。虽然表 6-1 中标记为 “ 保留 ” 的异常事件在向量表中都有自己的保留地址，但不会使用它们。6.5 中断源一节将介绍各个中断源 （异常事件编号 16 到 47）。

6.4 异常事件源

本节介绍了表 6-1 中所列出的不同异常事件源 （异常事件编号 1 到 15）。

6.4.1 复位异常事件

对于 PSoC 4，器件复位被视为一个异常事件。该事件始终处于有效状态，固定优先级为 –3 （即优先级最高）。器件复位可以由不同的原因 （如：上电复位 （POR）、 XRES 引脚上的外部复位信号或看门狗复位）引起。在复位器件时，将在监控只读存储器 （SROM）范围外执行用于配置器件的初始引导代码。 SROM 存储器中的引导代码和其它数据均由赛普拉斯编程，并且外部用户不能对其进行读 / 写访问。在完成 SROM 引导序列后，CPU 代码执行将跳转到闪存存储器。闪存存储器地址 0x00000004（表 6-1 中编号为 1 的异常事件）表示闪存存储器中启动代码的位置。 CPU 从该地址开始执行代码。注意，不能使用 SRAM 向量表中的复位异常事件地址，因为器件退出了复位状态后会选择闪存向量表。用于选用 SRAM 向量表的寄存器配置代码仅作为启动代码的一部分。在退出复位状态后，将执行启动代码，那时也将执行寄存器配置代码。

6.4.2 不可屏蔽中断 （NMI）异常事件

除了复位外，不可屏蔽中断 （NMI）是优先级最高的异常事件。该中断始终被使能，固定优先级为 –2。在 PSoC 4 器件中，可通过三种方式触发一个 NMI 异常事件：

- **由硬件信号触发的 NMI 异常事件（用户 NMI 异常事件）：** PSoC 4 可通过数字信号触发 NMI 异常事件。该数字信号指的是表 6-3 中的 irq_out[0]。由 irq_out[0] 触发的 NMI 异常事件将执行活动向量表 （闪存或 SRAM 向量表）所指向的 NMI 处理器。
- **通过置位NMIPENDSET位触发的NMI异常事件（用户NMI异常事件）：** 通过置位中断控制状态寄存器 （CM0_ICSR 寄存器）中的 NMIPENDSET 位，可以触发 NMI 异常事件。通过置位该位，可以执行活动向量表 （闪存或 SRAM 向量表）所指向的 NMI 处理程序。
- **系统调用 NMI 异常事件：** 该异常事件用于非易失性编程操作，如闪存写入操作和闪存校验和操作。通过置位 CPUSS_SYSREQ 寄存器中的 SYSCALL_REQ 位触发该异常事件。通过 SYSCALL_REQ 位触发的 NMI 异常事件始终执行 SROM 中的 NMI 异常事件处理程序代码。闪存或 SRAM 异常事件向量表不适用于系统调用 NMI 异常事件。不能对 SROM 中的 NMI 处理程序代码进行读 / 写操作，因为该代码包含了用户不能修改的非易失性编程子程序。

6.4.3 HardFault （硬故障）异常事件

HardFault 是正常或异常处理过程中因发生错误而导致的异常事件。该异常事件始终被使能。 HardFault 的固定优先级为 –1，即其优先级高于其他所有异常事件可配置的优先级。HardFault 异常事件是包含不同故障条件类型（包括执行未定义的指令和访问无效的存储器地址）的综合异常事件。 CM0 CPU 虽然不为HardFault异常处理程序提供故障状态的信息，但在软件能从故障恢复的情况下，它允许处理器返回异常事件，并持续运行。

6.4.4 管理程序调用 （SVCall）异常事件

当 CPU 执行应用代码的 SVC 指令时，会引起一直使能的管理程序调用 （SVCall）异常事件。应用软件使用 SVC 指令来调用一个底层的操作系统并提供一个处理操作。该调用操作被称为管理程序调用。SVC 指令允许应用生成一个管理程序调用，用以请求对系统的特权访问。请注意，PSoC 4 中的 CM0 将一个特权模式用于系统调用 NMI 异常事件（它与 SVCall 异常事件无关）。（有关特权模式的详细信息，请参考第 99 页上的芯片运行模式章节。）在器件的架构级别上，不支持 SVCall 的其他任何特权模式。因此，应用开发人员必须根据最终应用的要求来定义 SVCall 异常处理程序。

通过写入系统处理程序优先级寄存器 2 （SHPR2）中的两个位字段 PRI_11[31:30]，可配置 SVCall 异常的优先级 （范围为 0 到 3）。当执行 SVC 指令时， SVCall 异常将进入挂起状态，并等待 CPU 对其进行处理。通过使用系统处理器控制和状态寄存器 （SHCSR）中的 SVCALLPENDED 位，可以检查或修改 SVCall 异常的挂起状态。

6.4.5 PendSV 异常事件

PendSV 是另一种与 SVCall 相似的管理程序调用异常事件，该异常事件通常由软件生成。PendSV 始终被使能，并且其优先级是可配置的。通过置位控制状态寄存器 （CM0_ICSR）中的 PENDSVSET 位，可以触发 PendSV 异常事件。当置位该位时，PendSV 异常会进入挂起状态，并等待 CPU 的处理。通过设置中断控制状态寄存器（CM0_ICSR）中的 PENDSV-CLR 位，可以清除 PendSV 异常事件的挂起状态。通过写入到系统处理程序优先级寄存器 3 （CM0_SHPR3）中的两个位字段 PRI_14[23:22]，可以配置 PendSV 异常事件的优先级（范围为 0 到 3）。更多有关信息，请参考 ARMv6-M 架构参考手册。

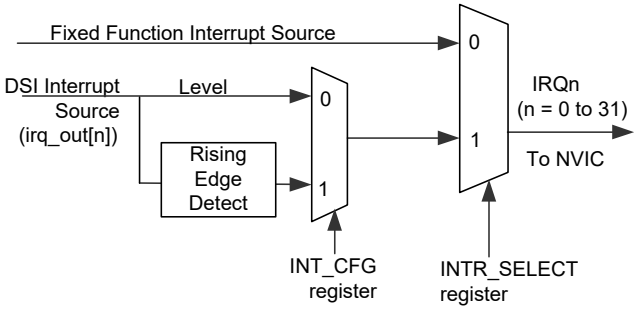
6.4.6 SysTick 异常事件

PSoC 4 中的 CM0 CPU 支持将系统定时器（也称为 SysTick）作为其内部架构的一部分。SysTick 为 RTOS 计时定时器、高速警报定时器、简单计数器等计时工具提供了一个简单的 24 位递减计数器。可配置 SysTick 定时器，使其计数值为零时将生成一个中断，即 SysTick 异常事件。通过置位 SysTick 控制和状态寄存器 （CM0_SYST_CSR）中的 TICKINT 位，可以使能该异常事件。通过写入到系统处理程序优先级寄存器 3 （SHPR3）中的两位字段 PRI_15[31:30] ，可以配置 SysTick 异常事件的优先级（范围为 0 到 3）。通过向中断控制状态寄存器 CM0_ICSR 中的 PENDSTSET 位写入 1，可以随时在软件中生成 SysTick 异常事件。同样，也可通过将 ‘1’ 写入到中断控制状态寄存器 CM0_ICSR 中的 PENDSTCLR 位来清除 SysTick 异常事件的挂起状态。

6.5 中断源

PSoC 4 支持来自外设的 32 个中断（IRQ0 - IRQ31 或异常编号 16 到 47）。表 6-3 介绍了每个中断的源。PSoC 4 为 32 个中断线都提供了灵活的中断源选项。图 6-5 显示了中断源的复用选项。每个中断均有两个源：固定功能的中断源和 DSI 中断源。通过 CPUSS_INTR_SELECT 寄存器，可以选用所需的中断源。

图 6-5. 中断源复用



注意: 不能对名称为（irq_out[n]）的 DSI 中断信号进行访问，但通过使用 DSI 中断路径路由数字信号， PSoC Creator IDE 可以简化该访问操作。因此，您不需要手动配置 DSI 路径。

固定功能中断包括来自片上外设的标准中断，如 TCPWM、串行通信模块、和 CSD 模块。通常通过对外设的不同状态进行 “ 或 ” 逻辑运算来生成固定功能中断。需要在 ISR 中读取外设状态寄存器，以检测生成中断的条件。固定功能中断通常是由电平触发的，所以需要通过在执行 ISR 时读取外设状态寄存器来清除该中断。如果执行 ISR 时未读取状态寄存器，该中断将保持激活状态，并且 CPU 将连续执行 ISR。中断源的第二类是 DSI 中断信号。共有八个 DSI 通道，每个通道被多路分解为 4 以跨越 Cortex M0 的 32 个中断源。片上的任何数字信号，如来自 UDB 的数字输出或引脚上的数字输入信号，都能作为 DSI 中断源使用。这样便能够更加灵活地选择中断源。您也可以通过上升沿检测电路路由 DSI 信号，如图 6-5 所示。该边沿检测电路将 DSI 线上的一个上升沿信号转换为宽度为两个系统时钟的边沿触发信号。这样确保在 DSI 线上信号的上升沿到来时，会立即触发中断。这对不能为 NVIC 生成合适的电平中断信号的中断源很有帮助。UDB_INT_CFG 寄存器用于选择直接 DSI 路径还是边沿检测路径。 由于 DSI 中断通道被多路分解，因此每次执行的 DSI 中断的最大数量被限制为 8。

请参考第 69 页上的 I/O 系统章节，了解 GPIO 中断的详细信息。

表 6-2. PSoC 4 中断源的列表

中断	Cortex-M0 异常编号	固定功能的中断源	DSI 中断源
NMI （请参见第 59 页上的 “ 异常事件源 ” 一节）	2	SYS_REQ	udb.interrupts[0]:4
IRQ0	16	GPIO 中断 — 端口 0	udb.interrupts[0]:0
IRQ1	17	GPIO 中断 — 端口 1	udb.interrupts[1]:0
IRQ2	18	GPIO 中断 — 端口 2	udb.interrupts[2]:0
IRQ3	19	GPIO 中断 — 端口 3	udb.interrupts[3]:0
IRQ4	20	GPIO 中断 — 端口 4	udb.interrupts[4]:0
IRQ5	21	GPIO 中断 — 端口 5	udb.interrupts[5]:0
IRQ6	22	GPIO 中断 — 所有端口[a]	udb.interrupts[6]:0
IRQ7	23	LPCOMP （低功耗比较器）	udb.interrupts[7]:0
IRQ8	24	WDT （看门狗定时器）	udb.interrupts[0]:1
IRQ9	25	SCB1 （串行通信模块 1）	udb.interrupts[1]:1
IRQ10	26	SCB2 （串行通信模块 2）	udb.interrupts[2]:1
IRQ11	27	CTBm 中断 （所有 CTBm）	udb.interrupts[3]:1
IRQ12	28	BLE 子系统中断	udb.interrupts[4]:1
IRQ13	29	SPCIF 中断	udb.interrupts[5]:1
IRQ14	30	LVD 中断	udb.interrupts[6]:1
IRQ15	31	SAR （逐次逼近 ADC）	udb.interrupts[7]:1
IRQ16	32	CSD （CapSense）	udb.interrupts[0]:2
IRQ17	33	TCPWM0 （定时器 / 计数器 /PWM 0）	udb.interrupts[1]:2
IRQ18	34	TCPWM1 （定时器 / 计数器 /PWM 1）	udb.interrupts[2]:2
IRQ19	35	TCPWM2 （定时器 / 计数器 /PWM 2）	udb.interrupts[3]:2
IRQ20	36	TCPWM3 （定时器 / 计数器 /PWM 3）	udb.interrupts[4]:2
IRQ21	37	< 仅用于 DSI>	udb.interrupts[5]:2
IRQ22	38	< 仅用于 DSI>	udb.interrupts[6]:2
IRQ23	39	< 仅用于 DSI>	udb.interrupts[7]:2
IRQ24	40	< 仅用于 DSI>	udb.interrupts[0]:3
IRQ25	41	< 仅用于 DSI>	udb.interrupts[1]:3
IRQ26	42	< 仅用于 DSI>	udb.interrupts[2]:3
IRQ27	43	< 仅用于 DSI>	udb.interrupts[3]:3

表 6-2. PSoC 4 中断源的列表

中断	Cortex-M0 异常编号	固定功能的中断源	DSI 中断源
IRQ28	44	< 仅用于 DSI>	udb.interrupts[4]:3
IRQ29	45	< 仅用于 DSI>	udb.interrupts[5]:3
IRQ30	46	< 仅用于 DSI>	udb.interrupts[6]:3
IRQ31	47	< 仅用于 DSI>	udb.interrupts[7]:3

a. 端口 6 没有专用中断向量；它使用向量 IRQ6。

6.6 异常事件优先级

当 CPU 需要处理多个异常事件时，可通过异常优先级实现异常仲裁。在 PSoC 4 中，能够灵活为不同异常事件选择优先级。除复位、NMI 和 HardFault 外，可为其它所有异常事件分配可配置的优先级。复位、NMI 和 HardFault 异常有自己的固定优先级，分别为 –3、–2 和 –1。在 PSoC 4 中，编号越低，优先级越高。因此，复位、NMI 和 HardFault 异常事件的优先级最高。其他异常事件的优先级可配置范围为 0 到 3。

PSoC 4 支持嵌套异常事件，这样优先级更高的异常事件可优先于 （中断）当前运行的异常事件处理。如果下一个异常与当前运行的异常有相同的优先级，那么该优先执行性能无效。处理完优先级高的异常后， CPU 会恢复执行优先级低的异常。PSoC 4 中的 CM0 CPU 支持嵌套多达四个异常事件。当 CPU 接收到两个或多个优先级相同的异常事件请求，将先执行编号最低的异常事件。

第 59 页上的 “ 异常事件源 ” 一节介绍了用于配置异常事件编号 1 到 15 的优先级的寄存器。

通过对中断优先级寄存器 （CM0_IPR）进行写操作，可以配置 32 个中断 （IRQ0 - IRQ31）的优先级。该组寄存器包括八个 32 位寄存器，其中每个寄存器用于存储四个中断的优先级值，如表 6-3 中所述。不使用该寄存器的其它位字段。

表 6-3. 中断优先级寄存器位的定义

位	名称	说明
7:6	PRI_N0	中断编号 N 的优先级。
15:14	PRI_N1	中断编号 N+1 的优先级。
23:22	PRI_N2	中断编号 N+2 的优先级。
31:30	PRI_N3	中断编号 N+3 的优先级。

6.7 使能和禁用中断

NVIC 提供了各种寄存器，用于单独使能和禁用软件中的 32 个中断。如果未使能某个中断，则 NVIC 将不会处理该中断线上的中断请求。中断设置使能寄存器 （CM0_ISER）和中断清除使能寄存器 （CM0_ICER）分别用于使能和禁用中断。这些寄存器的宽度均为 32 位，其中每个位相应于编号与其相同的中断。还可以通过软件读取这些寄存器，从而获取中断的使能状态。表 6-4 显示了两个寄存器的访问属性。请注意，向这些寄存器写入 ‘0’ 不起任何作用。

表 6-4. 中断使能 / 禁用寄存器

寄存器	操作	位值	说明
中断设置使能寄存器 （CM0_ISER）	写入	1	使能中断
		0	无作用
	读取	1	使能中断
		0	禁用中断
中断清除使能寄存器 （CM0_ICER）	写入	1	禁用中断
		0	无作用
	读取	1	使能中断
		0	禁用中断

CM0_ISER 和 CM0_ICER 寄存器仅适用于 IRQ0 到 IRQ31 的中断。不能通过这些寄存器使能或禁用编号为 1 到 11 的异常事件。这 15 个异常事件具有它们专有的使能和禁用支持，如第 59 页上的 “ 异常事件源 ” 一节所述。

Cortex-M0 （CM0） CPU 中的 PRIMASK 寄存器可作为全局异常使能寄存器使用。无论异常事件的使能情况如何，该寄存器都可以对所有可配置优先级的异常事件进行掩码。除表 6-1 中介绍的复位、NMI 和 HardFault 等异常事件外，其它所有的异常事件的优先级都是可配置的。可将它们的优先级配置为 0 到3，其中0是最高优先级，3是最低优先级。当置位PRIMASK 寄存器中的 PM 位 （位 0）时， CPU 不会处理任何优先级可配置的异常事件。但这些异常事件可以处于挂起状态，等到 PM 位被清除后， CPU 将处理这些事件。

6.8 异常事件的状态

每个异常事件都能处在下面各状态之一。

表 6-5. 异常事件的状态

异常事件的状态	含义
非活动状态	该异常事件处于非活动状态，并不被挂起。该异常事件被禁用，或者使能后的异常事件未被触发。
挂起状态	CPU/NVIC 接收异常事件请求，并且该异常事件正在等待 CPU 对其进行处理。
活动状态	CPU 正在处理异常事件，但该事件的处理程序操作尚未完成。优先级更高的异常事件可以中断优先级更低的异常事件的执行。在这种情况下，这两个异常事件都处于活动状态。
活动和挂起状态	处理器正在处理异常事件。与此同时，将挂起一个来自相同源的请求。

中断控制状态寄存器 （CM0_ICSR）包含用于描述各种异常状态的状态位。

- CM0_ICSR 中的 VECTACTIVE 位（[8:0]）用于存储当前执行的异常事件的编号。如果 CPU 尚未执行任何异常处理器 （CPU 处于线程模式），则该值为零。注意， VECTACTIVE 位字段中的值与中断编程状态寄存器 （IPSR）中位 [8:0] 内的值相同。IPSR 还用来存储有效的异常事件的编号。
- CM0_ICSR 寄存器中的 VECTPENDING 位（[20:12]）存储了优先级最高的挂起异常事件的编号。如果没有任何挂起异常事件，则该值为零。
- CM0_ICSR 寄存器中的 ISRPENDING 位（位 22）表示由 NVIC 生成的中断（IRQ0 到 IRQ31）是否处于挂起状态。

6.8.1 挂起异常事件

当外设为 NVIC 生成一个中断请求信号，或发生某个异常事件时，相应的异常事件将进入挂起状态。当 CPU 开始执行相应的异常处理器子程序时，该异常事件将从挂起状态转为活动状态。

通过使用独立的寄存器位来设置和清除中断的挂起状态，NVIC 允许软件挂起 32 个中断线。中断设置挂起寄存器（CM0_ISPR）和中断清除挂起寄存器 （CM0_ICPR）分别用于设置和清除中断线的挂起状态。这些寄存器的宽度均为 32 位，其中每个位相应于编号与其相同的中断。

表 6-6 显示了这两个寄存器的访问优先级。请注意，向这些寄存器写入 ‘0’ 不起任何作用。

表 6-6. 中断设置挂起 / 清除挂起寄存器

寄存器	操作	位值	说明
中断设置挂起寄存器（CM0_ISPR）	写入	1	使某个中断进入挂起状态
		0	无作用
	读取	1	中断处于挂起状态
		0	中断没有挂起
中断清除挂起寄存器（CM0_ICPR）	写入	1	清除某个挂起中断
		0	无作用
	读取	1	中断处于挂起状态
		0	中断没有挂起

在挂起位被置位时再次对同样位进行设置，则只执行一次 ISR。不管相应的中断是否被使能，仍会更新挂起位。如果未使能中断，则中断线不会转入挂起状态，直到通过写入 CM0_ISER 寄存器使能该中断为止。

请注意：CM0_ISPR 和 CM0_ICPR 寄存器仅用于 32 个外设中断（异常编号为 16 到 47）。不能使用它们来挂起编号为 1 到 11 的异常事件。这 15 个异常事件具有自己的挂起支持，如第 59 页上的 “ 异常事件源 ” 一节所述。

6.9 异常事件的堆栈使用情况

当 CPU 执行主代码（在线程模式中）并且发生某个异常请求时，CPU 会将其通用寄存器的状态信息存储到堆栈内。然后，在处理程序模式下， CPU 开始执行相应的异常处理程序。CPU 会将八个 32 位内部寄存器的内容推移到堆栈上。这些寄存器包括编程和状态寄存器 （PSR） 、返回地址、链接寄存器（LR 或 R14）、R12、R3、R2、R1 和 R0。Cortex-M0 具有两个堆栈指针 — MSP 和 PSP。一次只能激活一个堆栈指针。在线程模式下，控制寄存器中的活动堆栈指针位用于定义当前有效的堆栈指针。在处理器模式下， MSP 始终作为堆栈指针使用。Cortex-M0 中的堆栈指针始终向下移动并指向最后转移的数据的地址。

当 CPU 处于线程模式中，并且发生某个异常请求时，CPU 将使用控制寄存器中定义的堆栈指针来存储通用寄存器的内容。堆栈推移操作完成后， CPU 将进入处理程序模式，以执行异常处理程序。执行当前异常事件期间，如果出现优先级更高的异常事件，会使用 MSP 实现堆栈推移 / 弹出操作，这是因为 CPU 已经处于处理程序模式。有关详细信息，请参考第 37 页上的 Cortex-M0 CPU 章节中介绍的内容。

Cortex-M0 使用两项技术 （即末尾连锁和迟到）来缩短服务异常事件的延迟。对于外部用户，这些技术是不可见的。它们仅作为内部处理器架构的一部分。有关 “ 尾连锁 ” 和 “ 迟到 ” 机制的信息，请访问 ARM 信息中心。

6.10 中断和低功耗模式

在 PSoC 4 中，当生成外设中断请求时，器件可以从低功耗模式唤醒。当一个或更多唤醒源生成某个中断信号时，唤醒中断控制器 （WIC）模块会生成一个唤醒信号，从而使器件进入活动模式。器件进入活动模式后，将执行外设中断的 ISR。

由 CM0 CPU 执行的等待中断 （WFI）指令可使器件进入睡眠、深度睡眠和休眠等模式。第 101 页上的功耗模式章节将说明进入不同低功耗模式的序列。芯片的低功耗模式有三类固定功能的中断源：

- 可用于活动、深度睡眠和休眠等模式的固定功能中断源（例如， GPIO 中断，低功耗比较器）
- 仅用于活动模式和深度睡眠模式的固定功能中断源 （例如，看门狗定时器中断，串行通信模块中断和 BLE 子系统中断）
- 仅用于活动模式的功能固定中断源（其它所有的固定功能中断）

6.11 异常事件 — 初始化与配置

本节介绍了 PSoC 4 中初始化和配置异常事件的不同步骤。

1. 配置异常向量表位置：使用异常事件的第一步是配置向量表位置 — 确定该向量表是在闪存存储器中还是在 SRAM 中。通过将 ‘1’ （SRAM 向量表）或 ‘0’ （闪存向量表）写入到 CPUSS_CONFIG 寄存器中的 VECT_IN_RAM 位字段 （位 0）内，可以实现该配置。对该寄存器进行写操作被视为器件初始化代码的一部分。

   如果应用需要动态改变向量地址，建议 SRAM 中的向量表可用。如果该表位于闪存内，则需要通过对闪存进行写操作来修改向量表中的内容。默认情况下， PSoC Creator IDE 使用的是 SRAM 中的向量表。
2. 配置单独异常事件：下一步是对某个应用所需要的单独异常事件进行配置。
   a. 配置异常事件或中断源。该配置操作包括设置中断生成条件。根据所需的特定异常事件，寄存器的配置会存在差异。
   b. 定义异常处理程序功能并将该功能的地址写入异常向量表。表 6-1 介绍了异常向量表的格式；需要将异常处理程序地址写入表中合适的异常编号入口。
   c. 设置异常事件的优先级，如第 62 页上的 “ 异常事件优先级 ” 一节所述。
   d. 使能该异常事件，如第 62 页上的 “ 使能和禁用中断 ” 一节所述。

6.12 寄存器

表 6-7. 寄存器列表

寄存器名称	说明
CM0_ISER	中断设置使能寄存器
CM0_ICER	中断清除使能寄存器
CM0_ISPR	中断设置挂起寄存器
CM0_ICPR	中断清除挂起寄存器
CM0_IPR	中断优先级寄存器
CM0_ICSR	中断控制状态寄存器
CM0_AIRCR	应用中断和复位控制寄存器
CM0_SCR	系统控制寄存器
CM0_CCR	配置和控制寄存器
CM0_SHPR2	系统处理程序优先级寄存器 2
CM0_SHPR3	系统处理程序优先级寄存器 3
CM0_SHCSR	系统处理程序控制和状态寄存器
CM0_SYST_CSR	系统定时器控制和状态寄存器
CPUSS_CONFIG	CPU 子系统配置寄存器
CPUSS_SYSREQ	系统请求寄存器
CPUSS_INTR_SELECT	中断复用器选择寄存器
UDB_INT_CFG	UDB 子系统中断配置寄存器

6.13 相关文档

- ARMv6-M 架构参考手册 — 本文档介绍了 ARM Cortex-M0 架构，包括指令集、 NVIC 架构和 CPU 的寄存器说明。

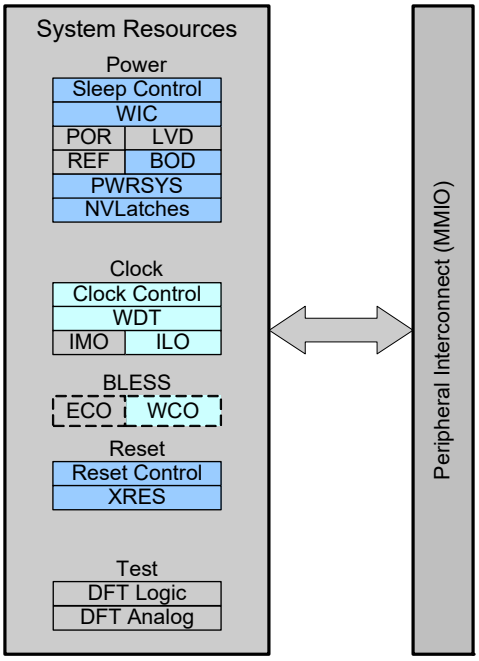
章节 C:　系统资源的子系统

本部分包括以下章节：

- 第 69 页上的 I/O 系统章节
- 第 83 页上的时钟系统章节
- 第 95 页上的电源供应和监控章节
- 第 99 页上的芯片运行模式章节
- 第 101 页上的功耗模式章节
- 第 107 页上的看门狗定时器章节
- 第 111 页上的复位系统章节
- 第 115 页上的器件安全性章节

顶层架构

系统范围资源框图



7. I/O 系统

本章节介绍了 PSoC® 4 I/O 系统以及其特性、架构、工作模式和中断。 PSoC 4 中的通用 I/O （GPIO）引脚被分为不同的端口。每个端口最多可包含 8 个 GPIO。 PSoC 4xxx-BL 最多有 36 个 GPIO，其中包括两个过压容限引脚，这些引脚被分布给六个端口。

7.1 特性

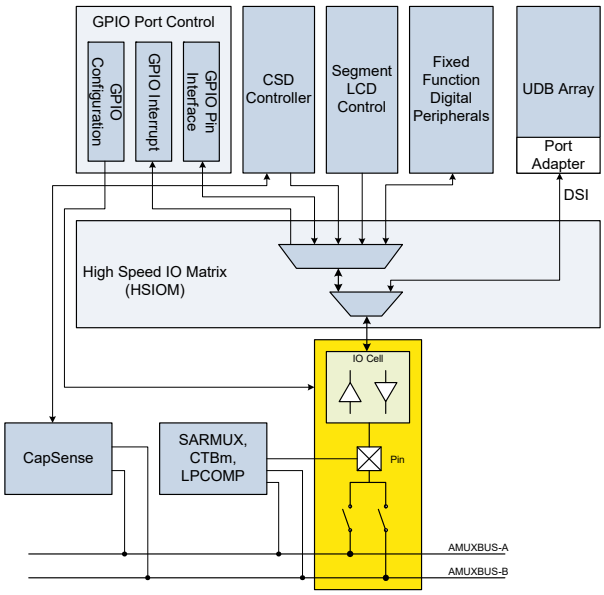
PSoC 4 GPIO 具有以下特性：

- 支持模拟 / 数字输入和输出
- 支持 8 种强驱动模式
- 两个过压容限 （OVT-GPIO）引脚，符合 I^2C 规范
- 支持在引脚上配置边沿触发中断 （包括上升沿、下降沿和双边沿等触发方式）
- 转换速率控制
- 保持模式，用于锁存前一种状态 （以便在深度睡眠模式下保持 I/O 状态）
- 可选择输入缓冲区模式：CMOS 输入或低电压 LVTTL 输入
- 支持 CapSense
- 支持 Segment LCD 驱动

7.2 GPIO 接口概况

PSoC 4 由多个模拟和数字外设组成。图 7-1 显示的是各个外设和引脚间的连接概况。

图 7-1. GPIO 接口概况

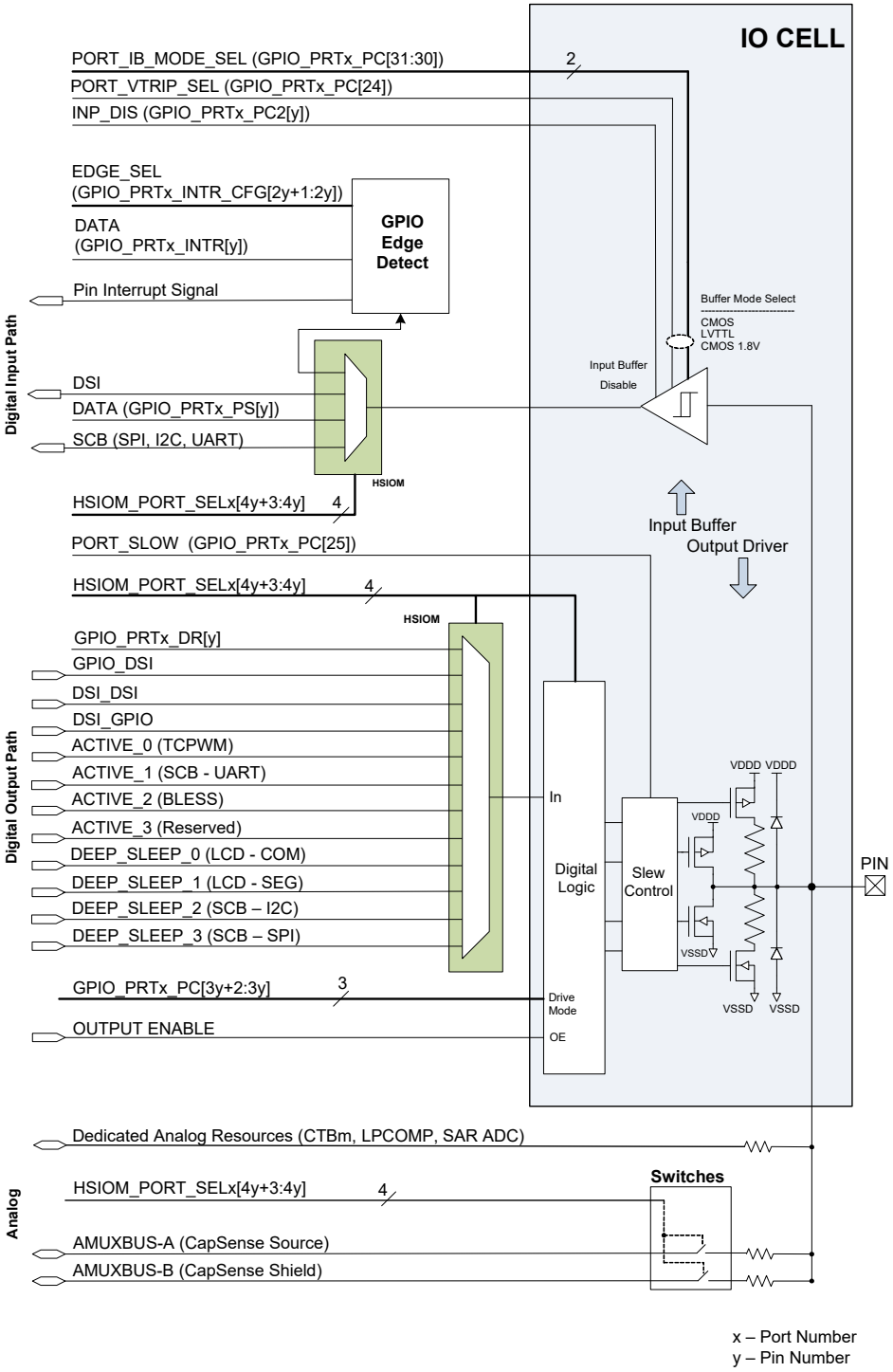


GPIO 引脚被连接到 I/O 单元。这些单元都配备了数字输入缓冲区（这样可提供高阻抗输入），以及数字输出信号的驱动器。数字外设通过高速 I/O 矩阵 （HSIOM）连接到 I/O 单元上。 HSIOM 包含多个复用器，用于将用户选定的器件连接到引脚上。HSIOM 还建立了数字系统互连 （DSI）与引脚间的连接。这样可以将引脚信号路由到已连接了 DSI 的外设 （如 UDB）上。模拟外设（如 SAR ADC、微型连续时间模块 （CTBm）、低功耗比较器 （LPCOMP）和 CapSense）可直接或通过 AMUX 总线连接到 GPIO 引脚上。

7.3 I/O 单元架构

图 7-2 显示的是 I/O 单元架构。它包括一个输入缓冲区和一个输出驱动器。所有 GPIO 单元都有这种架构。它连接着数字输入和输出信号的 HSIOM 复用器。模拟外设直接连接到引脚上。

图 7-2. PSoC 4 BLE 中的 I/O 单元架构



7.3.1 数字输入缓冲区

数字输入缓冲区为外部数字输入提供了高阻抗缓冲区。通过端口配置寄存器 2（GPIO_PRTx_PC2，其中 x 是端口编号）的 INP_DIS 位可使能 / 禁用该缓冲区。可以将该缓冲区配置为以下模式：

- CMOS
- LVTTL
- 1.8 V CMOS

这些缓冲区模式可通过端口配置寄存器的 PORT_VTRIP_SEL 位（GPIO_PRTx_PC[24]）和 PORT_IB_MODE_SEL 位（GPIO_PRTx_PC[31:30]）选定。

表 7-1. 输入缓冲区模式

PORT_VTRIP_SEL	PORT_IB_MODE_SEL	输入缓冲区模式
0b	0b 或 10b	CMOS
1b	0b 或 10b	LVTTL
x	1b 或 11b	1.8 V CMOS

欲了解每种模式的阈值，请参见器件数据手册。输入缓冲区的输出被连接至 HSIOM，以便路由到所选外设。通过对 HSIOM 端口选择寄存器（HSIOM_PORT_SELx）进行写操作，可以选择所需外设。 HSIOM 中的数字输入外设因引脚不同而不同，如图 7-2 中所示。请参阅器件数据手册，了解每个引脚的功能。

7.3.2 数字输出驱动器

引脚由数字输出驱动器驱动。该驱动器中包含的电路用于控制数字输出信号的不同驱动模式和转换速率。外设通过 HSIOM 连接到数字输出驱动器；通过写入到 HSIOM 端口选择寄存器（HSIOM_PORT_SELx）内，可以选择特定的外设。

在 PSoC 4 BLE 中，可以使用 V_{DDD} 供电电压来驱动 I/O。每个 GPIO 引脚都有 ESD 二极管，从而可维持引脚电压不超过 V_{DDD} 电源电压。请确保引脚电压不得超过 I/O 电源电压 / V_{DDD}，也不能低于 V_{SS}。有关 GPIO 电压的最大和最小绝对值，请参阅器件数据手册。通过外设的 DSI 信号或与输出引脚相关的数据寄存器（GPIO_PRTx_DR），可以启用或禁用数字输出驱动器。请参考 7.5 高速 I/O 矩阵一节，了解有关数据的外设源选择以及启用 / 禁用控制源选择的详细信息。

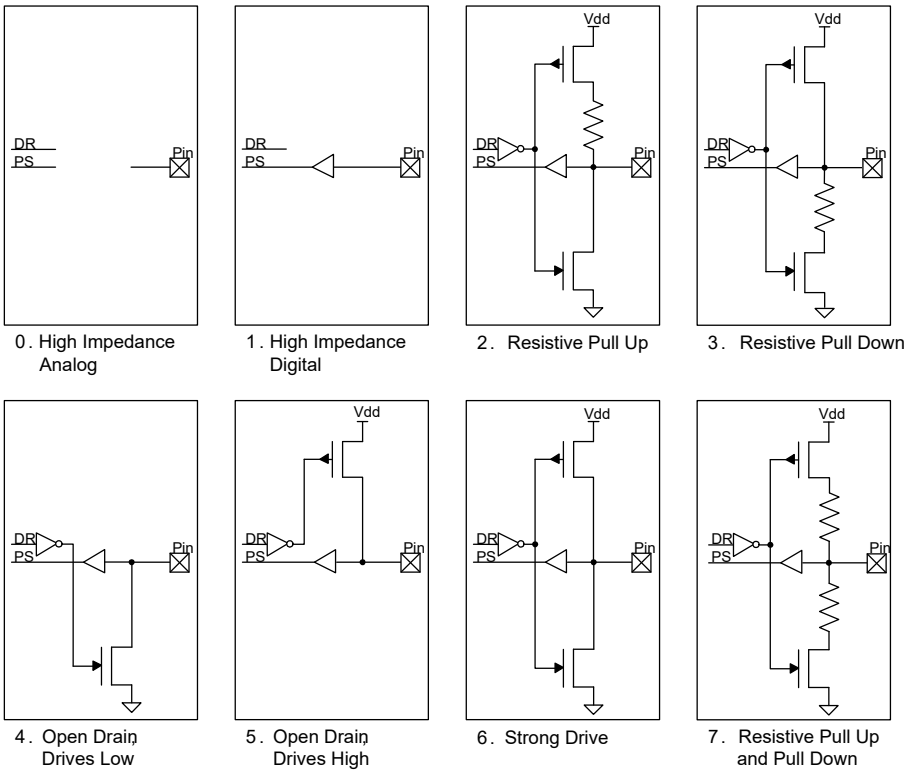
7.3.2.1 驱动模式

通过使用端口配置寄存器（GPIO_PRTx_PC），可将每个 I/O 单独配置为八种驱动模式中的一种。表 7-2 列出了各种驱动模式。图 7-2 是简单的输出驱动器框图，它显示了基于八种驱动模式的引脚视图。

表 7-2. 驱动模式设置

GPIO_PRTx_PC（‘x’ 表示端口编号，‘y’ 表示引脚编号）				
位	驱动模式	数值	数据 = 1	数据 = 0
3y+2: 3y	SEL‘y’	选择引脚 ‘y’ （0 ≤ y ≤ 7）的驱动模式		
	模拟高阻态	0	高阻态	高阻态
	数字高阻态	1	高阻态	高阻态
	电阻上拉	2	弱 1	强 0
	电阻下拉	3	强 1	弱 0
	开漏，驱动低电平	4	高阻态	强 0
	开漏，驱动高电平	5	强 1	高阻态
	强驱动	6	强 1	强 0
	电阻上拉和下拉	7	弱 1	弱 0

图 7-3. I/O 驱动模式框图



■ 模拟高阻态

模拟高阻态模式是默认的复位状态；在这种模式下，输出驱动器和数字输入缓冲区均被关闭。这种状态可以防止外部电压产生流入数字输入缓冲区的电流。对于悬空引脚或支持模拟电压的引脚，建议采用该驱动模式。模拟高阻态引脚不能作为数字输入使用。无论数据寄存器值如何，读取该引脚状态寄存器均返回 0x00 值。在低功耗模式下，为了实现最低器件电流，必须将不使用的 GPIO 配置为模拟高阻态模式。

■ 数字高阻态

数字高阻态模式是标准的高阻抗 （High Z）状态，建议将其用于数字输入。在这种状态中，为数字输入信号使能输入缓冲区。

■ 电阻上拉或电阻下拉

电阻模式在某种数据状态下可提供串联电阻，并在另一种数据状态下能够提供强驱动。在这些模式下，引脚可作为数字输入或数字输出使用。如果要求电阻上拉，请将 ‘1’ 写入到与引脚相应的数据寄存器位上。如果要求电阻下拉，请将 ‘0’ 写入到与引脚相应的数据寄存器位上。机械开关接口是这些驱动模式的常见应用。此外，还可以在这些电阻模式下将 PSoC 连接到开漏驱动线上。输入为开漏低电平时，使用电阻上拉；输入为开漏高电平时，则使用电阻下拉。

■ 开漏高电平驱动和开漏低电平驱动

开漏模式在某种数据状态下提供高阻抗，在另一种数据状态下提供强驱动。在这些模式下，引脚可作为数字输入或数字输出使用。因此，这些模式广泛使用于双向数字通信。当信号采用外部电阻下拉时，使用开漏高电平驱动模式；信号采用外部电阻上拉时，则使用开漏低电平驱动。开漏低电平驱动模式的常见应用是驱动 I^2C 总线信号线。

■ 强驱动

强驱动模式是各引脚的标准数字输出模式；在高电平和低电平状态下，它能提供 CMOS 输出强驱动。一般情况下，强驱动模式下的引脚不能作为输入使用。这种模式通常用于数字输出信号或用于驱动外部晶体管。

■ 电阻上拉和电阻下拉

在电阻上拉和电阻下拉模式下， GPIO 的串联电阻有两种输出状态，即逻辑 1 和逻辑 0。高电平数据被上拉时，低电平数据被下拉。当可能导致短路的其它信号驱动总线时，可使用该模式。

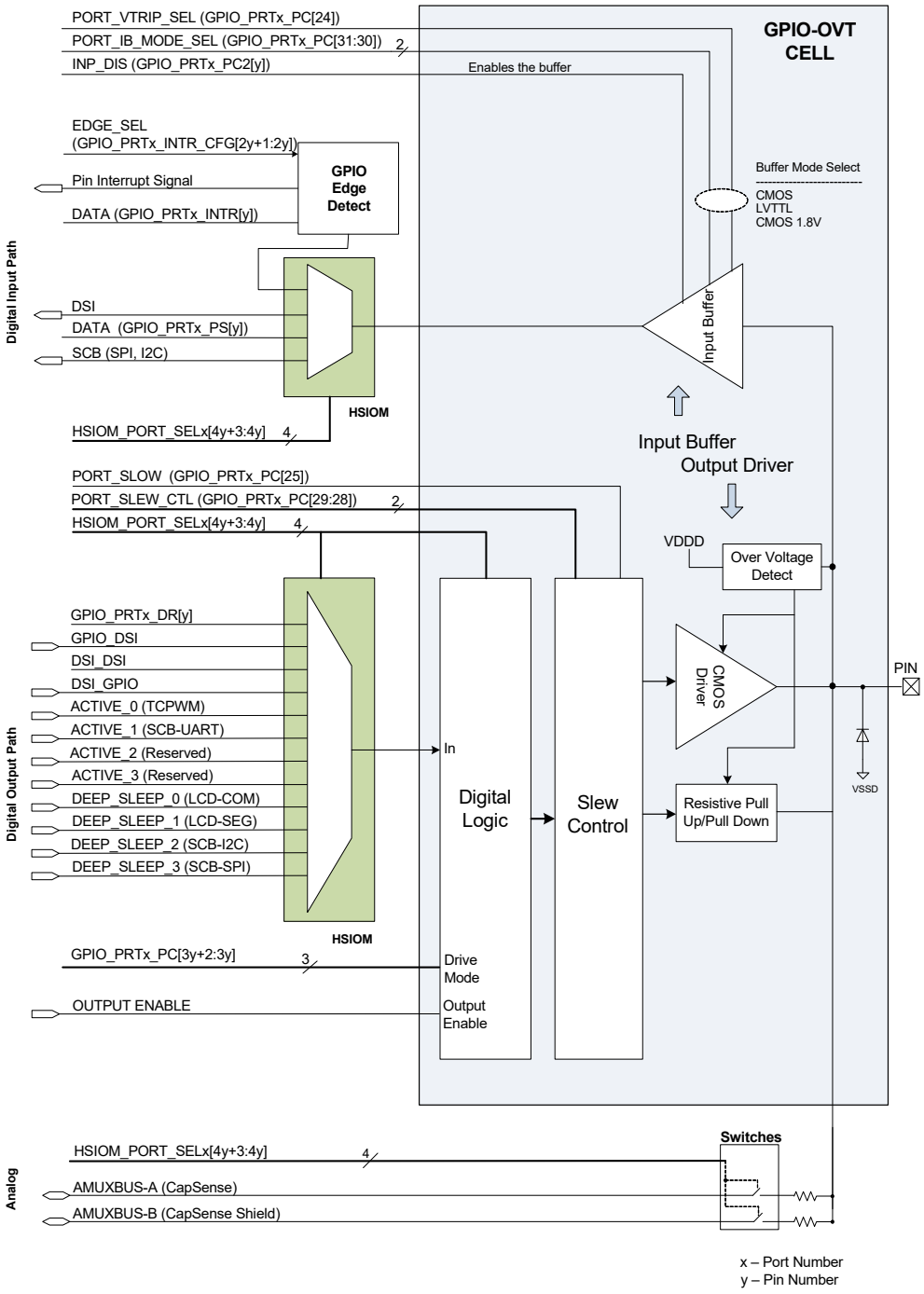
7.3.2.2 转换速率控制

GPIO 引脚在强驱动模式下具有快速和慢速输出速率转换选项。通过使用端口配置寄存器（GPIO_PRTx_PC[25]）的 PORT_SLOW 位，可以配置该选项。可以单独配置每个端口的转换速率。默认情况下，该位被清除，并且端口运行于快速转换速率模式。如果需要较低的转换速率，可以设置该位。转换速率越慢， EMI 和串扰则越少。因此，对低频信号或没有严格时序限制的信号，推荐使用慢速转换选项。

7.4 GPIO-OVT 引脚

PSoC 4 器件有两个过压容限 （OVT）引脚 — P5[0] 和 P5[1]。图 7-4 显示的是 GPIO-OVT 引脚架构。

图 7-4. PSoC 4 BLE 中的 GPIO-OVT 引脚架构



它与通用的 GPIO 相似，但具有以下各附加特性：

- 过压容差 — GPIO-OVT 单元使用硬件（图 7-4 中即为 “ 过压检测 ” 模块）来比较 VDDD 和引脚电压。如果引脚电压超过 VDDD 电压，则输出驱动器被禁用，并且该引脚变成三态。这样会生成引脚上可忽略的灌电流。此外，引脚和 VSSD 间还有 ESD 钳位二极管，可以限制负电压尖峰。

请注意，引脚上发生过压时，如果外部源的 VOH 和 VOL 规格不符合输入缓冲区的跳变点，则输入缓冲区数据无效。

- 与通用 GPIO 相比， OVT 引脚提供了更好的下拉驱动强度。
- 被配置为I^2C，并且它的各条信号线被路由到GPIO-OVT引脚时，则作为串行通信模块（SCB）使用。并且满足以下I^2C规范：
  - 加强型快速模式的 IOL 规范
  - 快速模式和加强型快速模式的迟滞以及最小下降时间规范

通过额外的转换速率控制功能，可以获得最小下降时间规范。通过使用端口配置寄存器 （GPIO_PRTx_PC[29:28]）的 PORT_SLEW_CTRL 位来配置该转换速率控制功能。表 7-3 显示的是 PORT_SLEW_CTRL 位的设置情况。

表 7-3. 转换速率控制

PORT_SLEW_CTRL (GPIO_PRTx_PC[29:28])	使用说明
00b	保留
01b	I^2C 增强型快速模式 （FM+）适用于外部电源电压超过 2.8 V 的 I^2C 总线
10b	保留
11b	I^2C 增强型快速模式 （FM+）适用于外部电源电压低于 2.8 V 的 I^2C 总线

请注意，要想使用 PORT_SLEW_CTRL 位，需要将驱动模式配置为开漏低电平驱动模式。如果选中其它驱动模式，则 PORT_SLEW_CTRL 不起任何作用。

7.5 高速 I/O 矩阵

高速 I/O 矩阵 （HSIOM）是一组将 GPIO 连接到器件中外设的高速开关。当多项功能共享 GPIO， HSIOM 将复用该引脚，并将其连接到用户选定的具体外设。通过 HSIOM_PORT_SELx 寄存器，可以选择所需外设。这是一个 32 位宽的寄存器。每个端口都使用该寄存器，每个引脚占用 4 位。该寄存器为每个引脚提供最多 16 个不同的选项，如表 7-4 所列。

表 7-4. PSoC 4 BLE HSIOM 端口设置

HSIOM_PORT_SELx （‘x’ 表示端口编号，‘y’ 表示引脚编号）			
位	**名称 （SEL‘y’）**	**数值**	**说明 （选择引脚 ‘y’ 的源 （0 ≤ y ≤ 7））**
4y+3 : 4y	DR	0	引脚是固件控制的通用 I/O，或者被连接到专用的硬件模块。
	DR_DSI	1	输出是由固件控制的，但 OE 通过 DSI 得到控制。
	DSI_DSI	2	输出和 OE 都由 DSI 控制。
	DSI_DR	3	输出是通过 DSI 控制， OE 则由固件控制。
	CSD_SENSE	4	引脚是 CSD 检测引脚 （在模拟模式下）。
	CSD_SHIELD	5	引脚是 CSD 屏蔽引脚 （在模拟模式下）。
	AMUXA	6	引脚被连接到 AMUXBUS-A。
	AMUXB	7	引脚被连接到 AMUXBUS-B。该模式还用于 CSD I/O 充电。如果 CSD I/O 充电是通过 CSD_CONTROL 使能，那么数字 I/O 驱动器会被连接到 csd_charge 信号 （引脚仍被连接到 AMUXBUS-B 上）。
	ACTIVE_0	8	特定于引脚的活动源 #0 （TCPWM）。
	ACTIVE_1	9	引脚特定的活动源 #1 （SCB-UART）。
	ACTIVE_2	10	特定于引脚活动源 #2 （BLESS、 EXT_CLK）
	ACTIVE_3	11	保留
	DEEP_SLEEP_0	12	特定于引脚的深度睡眠源 #0 （LCD - COM）
	DEEP_SLEEP_1	13	特定于引脚的深度睡眠源 #1 （LCD - SEG）
	DEEP_SLEEP_2	14	特定于引脚的深度睡眠源 #2 （SCB-I^2C、 SWD、 LPCOMP、唤醒）
	DEEP_SLEEP_3	15	特定于引脚深度睡眠源 #3 （SCB-SPI、 EXT_CLK）

注意: 活动和深度睡眠源由引脚决定。有关各引脚所支持特性的详细信息，请参考 器件数据手册中 “ 引脚分布 ” 部分的内容。

7.6 上电时的 I/O 状态

在上电过程中，所有 GPIO 均处于模拟高阻态，并且输入缓冲区均被禁用。器件运行期间，可以通过写入到相应的寄存器内来配置 GPIO。请注意，如果引脚支持调试访问端口 （DAP）的接口 （SWD 线），则上电时，这些引脚始终被配置为 SWD 线。但是，可以通过 HSIOM 禁用 DAP 接口或重新配置，以用于通用目的。然而，只在器件启动并开始执行代码后，才能重新配置该接口。

7.7 低功耗模式下的行为

表 7-5 显示的是低功耗模式下 GPIO 的状态。

表 7-5. 在低功耗模式下的 PSoC 4 BLE I/O

低功耗模式	状态
睡眠模式	■ 标准 GPIO 和 GPIO-OVT 引脚均处于活动状态，并能够通过 CapSense、TCPWM 和 SCB 等外设驱动，这些外设可在睡眠模式下工作。 ■ 输入缓冲区均处于活动状态，因此任意 I/O 引脚上的中断都能唤醒 CPU。
深度睡眠模式	■ 与深度睡眠模式下的外设连接的 GPIO 和 GPIO-OVT 引脚均可用。具有使能输出功能的其它引脚都处于冻结状态。 ■ 所有 I/O 引脚上的中断均可用。
休眠	■ 引脚输出状态被锁存，并保持为冻结状态。 ■ 所有 I/O 引脚上的中断均可用。请注意，不能将 GPIO 和 GPIO-OVT 引脚的缓冲输入配置为 1.8 V CMOS 模式。在休眠模式下，该模式不可用。
停止	■ GPIO 和 GPIO-OVT 引脚输出状态被锁存，并保持为冻结状态。 ■ 只有端口 P2[2] 上的中断可唤醒器件，因此不能将缓冲区输入配置为 1.8 V CMOS 模式。其它引脚上的输入缓冲区均无效。

7.8 输入和输出的同步

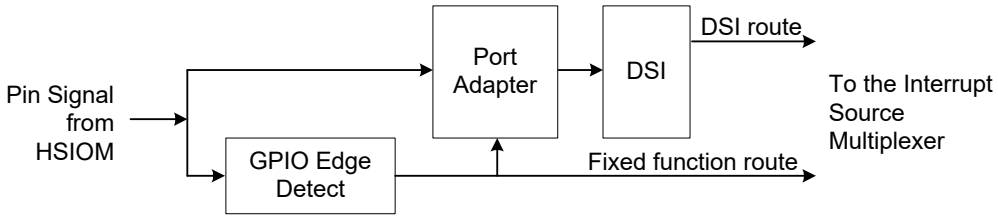
对于数字输入及输出信号，I/O 支持与内部时钟或作为时钟的数字信号进行同步。默认情况下使用 HFCLK 时钟，但仍可以使用其它时钟进行同步。

该特性可以通过使用 UDB 端口适配器实现。有关端口适配器的详细信息，请参见第 161 页上的通用数字模块 （UDB）章节。

7.9 中断

在 PSoC 4 器件中，所有端口引脚 （除端口 6 外）都能生成中断。通过以下三种引脚信号的路由方法，可以生成中断，如图 7-5 所示。

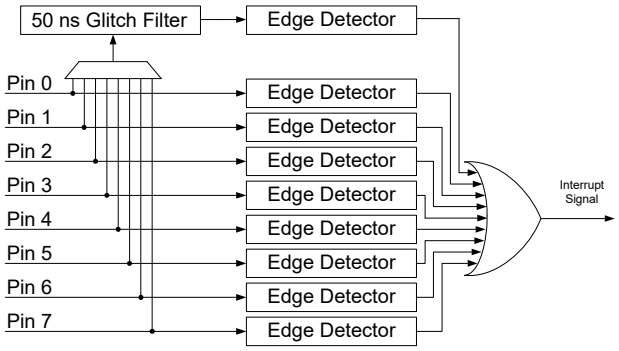
图 7-5. 中断信号路由



- 经过 “GPIO 边沿检测 ” 模块，然后直接连接至中断源复用器
- 经过 “GPIO 边沿检测 ” 模块，然后通过 DSI 连接至中断源复用器
- 引脚信号通过 DSI 连接至中断源复用器 （它没有经过 “GPIO 边沿检测 ” 模块）

图 7-6 显示的是 GPIO 边沿检测模块的架构。

图 7-6. GPIO 边沿检测模块架构



每个引脚上都有一个边沿检测器。无需重新对其进行任何配置即可检测上升沿、下降沿和双边沿。通过对端口中断配置寄存器 GPIO_PRTx_INTR_CFG 中的 EDGE_SEL 位字段进行写操作，可以配置边沿检测器，如表 7-6 中所示。

表 7-6. 边沿检测器配置

EDGE_SEL	配置
00	禁用中断
01	在上升沿上触发中断
10	在下降沿上触发中断
11	在双边沿上触发中断

不仅在每个引脚上使用了边沿检测器，窄脉冲滤波器输出上也使用了这种检测器。可将这种滤波器用在其中一个端口引脚上。通过写入到GPIO_PRTx_INTR_CFG寄存器中的FLT_-SEL 字段来选择该引脚，如表 7-7 所示。

表 7-7. 窄脉冲滤波器输入选择

FLT_SEL	所选引脚
000	选择引脚 0
001	选择引脚 1
010	选择引脚 2
011	选择引脚 3
100	选择引脚 4
101	选择引脚 5
110	选择引脚 6
111	选择引脚 7

对端口的边沿检测器的输出进行 OR（或）运算，然后将结果路由到中断控制器上 （即 CPU 系统中的 NVIC）。因此，每个端口只有一个中断向量。发生一个引脚中断时，需要知道引起中断的是哪个引脚。通过写入到端口中断状态寄存器 GPI-O_PRTx_INTR 内，可实现该操作。该寄存器不仅包含触发中断的引脚的信息，它还存储了引脚的状态。因此 CPU 只要通过单一的读取操作便能够读取这两种信息。另外，该寄存器还可以用于清除中断。向相应的中断状态位写入 ‘1’，即可清除引脚中断。必须清除中断状态位。否则，单个触发器会重复触发中断，或多个触发器只触发一次中断。稍后会在本节详细介绍该内容。另请注意，在读取端口中断控制状态寄存器时，如果相应端口上发生了中断，则不能正确检测该中断。因此，当使用 GPIO 中断时，建议仅在相应的中断服务子程序中（而不是在代码的其它任何部分）读取状态寄存器。表 7-8 显示的是端口中断状态寄存器的位字段。

表 7-8. 端口中断状态寄存器

GPIO_PRTx_INTR	说明
0000b 到 0111b	引脚 0 到引脚 7 的中断状态。将 ‘1’ 写入到相应位中可以清除该中断
1000b	窄脉冲滤波器输出信号的中断状态
10000b 到 10111	引脚 0 到引脚 7 的中断状态
11000b	窄脉冲滤波器的输出状态

边沿检测模块的输出信号被路由到中断源复用器上（如第 58 页上的图 6-3 所示），这样便可提供电平和上升沿检测选项。选中电平选项时，只要设置端口中断状态寄存器位，便可重复触发中断。选中上升沿检测选项时，如果端口中断状态寄存器未被清除，则只会触发一次中断。因此，使用边沿检测模块时，必须清除中断状态位。

通过固定功能模块路由中断信号时，每个端口上都有一个专用的中断向量。但是，当该信号通过 DSI 进行路由时，则中断向量非常灵活，它能够占用 NVIC 中任意 32 条中断线。有关详细信息，请参考第 57 页上的中断章节中介绍的内容。

当该信号被路由到 DSI （而旁路边沿检测模块）时，可以使用中断源复用器重新配置边沿检测选项。请注意，如果复用器被配置为电平触发，只要引脚信号为高电平，便能重复触发中断。选择这种路由方法时，推荐使用上升沿检测选项。

7.10 外设连接

7.10.1 固件控制的 GPIO

请查看表 7-4，了解由固件控制的 GPIO 的 HSIOM 设置。GPIO_PRTx_DR 是数据寄存器，用于读写 GPIO 的输出数据。对该寄存器进行写操作可将 GPIO 输出修改为所写入的值。请注意，读操作反映的是写入到该寄存器内的输出数据，并不是 GPIO 的当前状态。使用该寄存器时，在具有输入和输出 GPIO 的端口上，可以安全执行读取 - 修改 - 写入序列。

除了数据寄存器外，其它三个寄存器（GPIO_PRTx_DR_SET、GPIO_PRTx_DR_CLR 和 GPIO_PRTx_INV）分别用于设置、清除和反转端口上特定引脚的输出数据，并不会对其它引脚产生影响。将 ‘1’ 写入到这些寄存器可设置、清除或反转数据；将 ‘0’ 写入时不会影响引脚的状态。

GPIO_PRTx_PS 是 I/O 焊盘寄存器；读取该寄存器时，可获取 GPIO 的状态。而写入该寄存器时却不返回任何结果。

7.10.2 模拟 I/O

需要低阻抗路由路径的模拟资源（如 LPCOMP、SARMUX 和 CTBm）都有专用引脚。专用的模拟引脚可以直接连接到特定的模拟模块。它们有助于提高性能； 当使用这些模拟资源时，应优先使用这些专用引脚。有关这些专用引脚的详细信息，请参阅器件数据手册 。

为了将一个 GPIO 配置为专用模拟 I/O，应将其配置为高阻抗模拟模式（表 7-2），并且在特定的模拟资源中使能相应连接。通过与相应模拟资源相关联的寄存器，可以实现该操作。

为了将一个 GPIO 配置为连接至 AMUXBUS 的模拟引脚，需要将它配置为模拟高阻态模式，然后通过 HSIOM-_PORT_SELx 寄存器路由到 AMUXBUS。

7.10.2.1 AMUXBUS 连接和 DSI

在支持 DSI 连接的端口上，将某个引脚连接到 AMUXBUS A/B 时，除了需要配置 HSIOM_PORT_SELx 寄存器外，用户还要配置一对 DSI 信号。应将引脚的 DSI 输出和 DSI 输出使能信号设置为高电平，以使能 AMUXBUS 连接。这样能够使用 DSI 信号来控制某个引脚上的 AMUXBUS 连接，因此用户可以通过 DSI 实现 AMUXBUS 的硬件功能（即切换信号）。要正确将某个引脚配置为AMUXBUS输入，需要执行以下步骤：

1. 在设计中，将引脚的 DSI 输出和 DSI 输出使能信号连接至逻辑 ‘1’，以进行静态 AMUXBUS 连接。如果设计中要求动态 AMUXBUS 连接，可以将这些信号连接至 DSI 选定的信号上。可以在器件中进行该操作。
2. 配置 GPIO_PRTx_PC 寄存器，以便将引脚设置为高阻态模拟模式，从而可以通过禁用输入缓冲区来使能引脚上的该模拟连接。
3. 配置 HSIOM_PRT_SELx 寄存器，以将该引脚连接到 AMUXBUS A/B。

7.10.3 LCD 驱动

所有GPIO都具有驱动某个LCD的共模信号和段信号的能力。HSIOM_PORT_SELx寄存器用于选择LCD驱动使用的引脚。有关详细信息，请参考第 299 页上的 LCD 直接驱动章节中的内容。

7.10.4 CapSense

可将支持 CSD 的引脚配置为各种 CapSense Widget，如按键、滑条元件、触摸板元件或接近感应传感器。CapSense 还需要外部槽电容和屏蔽线路。表 7-9 显示了 CapSense 所需的 GPIO 和 HSIOM 设置。更多有关信息，请参考第 311 页上的 CapSense 章节中的内容。

表 7-9. CapSense 设置

CapSense 引脚	GPIO 驱动模式（GPIO_PRTx_PC）	数字输入缓冲区设置（GPIO_PRTx_PC2）	HSIOM 设置
传感器	模拟高阻态	禁用缓冲区	CSD_SENSE
屏蔽	模拟高阻态	禁用缓冲区	CSD_SHIELD
CMOD （正常操作）	模拟高阻态	禁用缓冲区	AMUXBUS A 或 CSD_COMP
CMOD （GPIO 预充电，仅用于选择 GPIO）	模拟高阻态	禁用缓冲区	AMUXBUS B 或 CSD_COMP
CSH TANK （GPIO 预充电，仅用于选择 GPIO）	模拟高阻态	禁用缓冲区	AMUXBUS B 或 CSD_COMP

7.10.5 蓝牙低功耗子系统（BLESS）

PSoC 4器件中的蓝牙子系统具有专用的引脚，用于连接32 kHz、24 MHz的晶振及天线。有关详细信息，请参考器件数据手册。

7.10.6 串行通信模块 （SCB）

可将 SCB 配置为 UART、I²C 和 SPI，并可将它连接到专用引脚上。有关 PSoC 4 的专用引脚的详细信息，请参阅器件数据手册。使用 UART 和 SPI 模式时， SCB 将控制该输入引脚的数字输出缓冲区驱动模式，从而使该引脚保持为高阻态。SCB 模块禁用 UART Rx 引脚上的输出缓冲区。如果该模块被配置为 SPI 主设备，它将禁用 MISO 引脚上的输出缓冲区。如果被配置为SPI从设备，则禁用MOSI引脚和选择线引脚上的输出缓冲区。这样会更改驱动模式的设置，该操作是通过使用GPIO_PRTx_PC寄存器来实现的。

7.11 端口限制

端口 4 和编号更高的端口上的引脚都没有端口适配器，因此存在以下限制：

- 不能通过 DSI 路由。因此，基于 UDB 的数字信号不能路由到这些端口的引脚上。
- 无输入 / 输出同步

但是，这些端口可以用于以下情况：

- 作为由固件控制的 GPIO 引脚使用
- 直接连接到 TCPWM、 SCB 或 CAN
- 作为 LCD 和 CapSense 引脚使用
- 中断生成

7.12 寄存器

表 7-10. I/O 寄存器

名称	说明
GPIO_PRTx_DR	端口输出数据寄存器
GPIO_PRTx_DR_SET	端口输出数据设置寄存器
GPIO_PRTx_DR_CLR	端口输出数据清除寄存器
GPIO_PRTx_DR_INV	端口输出数据反转寄存器
GPIO_PRTx_PS	端口引脚状态寄存器，用于读取 I/O 的逻辑引脚状态
GPIO_PRTx_PC	端口配置寄存器 — 用于配置输出驱动模式、输入阈值以及转换速率
GPIO_PRTx_PC2	端口辅助配置寄存器 — 用于配置 I/O 引脚的输入缓冲区
GPIO_PRTx_INTR_CFG	端口中断配置寄存器
GPIO_PRTx_INTR	端口中断状态寄存器
HSIOM_PORT_SELx	HSIOM 端口选择寄存器

注意：GPIO 寄存器名称中的 ‘x’ 表示端口编号。例如， GPIO_PTR1_DR 是端口 1 的输出数据寄存器。

8. 时钟系统

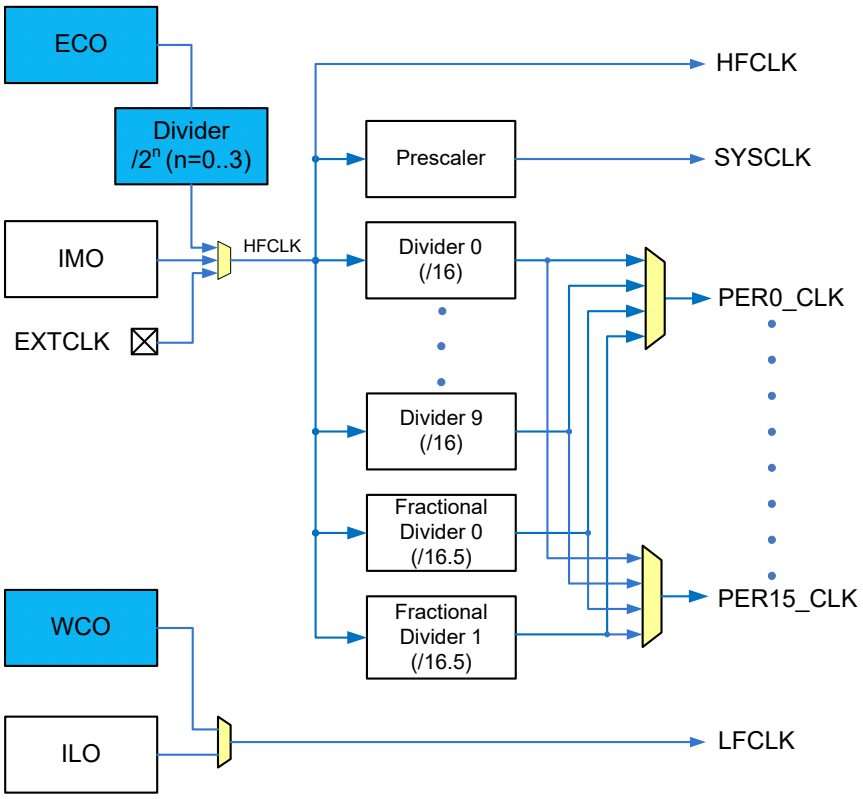
PSoC® 4 时钟系统包括以下时钟源：

- 两个内部时钟源：
  - 频率为 3 ~ 48 MHz 的内部主振荡器 （IMO），所有频率的调整后的精度误差为 ±2%
  - 32 kHz 内部低速振荡器 （ILO），调整后的频率的精度范围为 ±60% （可以使用 IMO 进行校准）
- 三个外部时钟源：
  - 通过使用 I/O 引脚的信号来生成的外部时钟 （EXTCLK）
  - 外部 24 MHz 晶振 （ECO）
  - 32 kHz 的外部监视晶振 （WCO）
- 从 IMO、 ECO 或外部时钟中选择频率高达 48 MHz 的高频时钟 （HFCLK）
- 来自 ILO 或 WCO 的低频时钟 （LFCLK）
- 系统时钟 （SYSCLK）专用预分频器，它的时钟源是频率高达 48 MHz 的 HFCLK
- 十个 16 位外设时钟分频器
- 两个小数分频器用于生成准确的时钟
- 16 个数字和模拟外设时钟

8.1 框图

图 8-1 提供 PSoC 4 器件中时钟系统的通用视图。

图 8-1. 时钟系统框图



该器件的 5 个时钟源包括 IMO、ECO、EXTCLK、WCO、和 ILO，如图 8-1 所示。 ECO 分频器对 ECO 时钟进行分频。HFCLK 复用器选择 EXTCLK、ECO 或 IMO 作为 HFCLK 源。LFCLK 复用器从 WCO 或 ILO 中选择 LFCLK。

8.2 时钟源

8.2.1 内部主振荡器

内部主振荡器在运行时不需要外部组件，并可输出一个稳定的时钟，其频率范围为 3 ~ 48 MHz，步长为 1 MHz。通过在寄存器 CLK_IMO_TRIM2 中设置频率、在寄存器 CLK_IMO_TRIM1 中设置IMO调整并最后在寄存器PWR_BG_TRIM4和PWR_BG_TRIM5 中设置带隙调整，可以选择频率。在 CLK_IMO_TRIM2 中设置的频率决定了 IMO 频率输出。表 8-1 提供了与 IMO 频率输出相对应的设置。除了在 CLK_IMO_TRIM2 中设置频率外，用户还需要加载 CLK_IMO_TRIM1、PWR_BG_TRIM4 和 PWR_BG_TRIM5 中各相应的调整值。选择频率时需要执行一个算法，从而确保没有任何中间状态被编程为高于 48 MHz 的频率。每个 PSoC 器件都有在制造过程中测量的 IMO 调整设置，以满足数据手册的要求；此调整被存储在 SFLASH 中的制造配置数据内。这些器件具有的 TRIM（调整）值与用户选定的频率相对应。SFLASH 中的 TRIM 值被加载到相应的调整寄存器（CLK_IMO_TRIM1、PWR_BG_TRIM4 和 PWR_BG_TRIM5）内。可以在启动时加载这些值，以实现所需的配置。固件可检索这些调整值，并重新配置器件，以运行时修改频率。

想要配置 IMO 频率，请遵循下面的算法：

- 如果 （(新频率 ≥ 43 MHz) 和 (就频率 ≥ 43 MHz)），
  - 那么要将CLK_IMO_TRIM2 的频率更改为一个更低的值，如 24 MHz
  - 将 CLK_IMO_TRIM1、PWR_BG_TRIM4 和 PWR_BG_TRIM5 应用于新频率
  - 等待 ≥ 5 µs
  - 将 CLK_IMO_TRIM2 更改成新频率
- 另外，如果 （新频率 > 旧频率），
  - 将 CLK_IMO_TRIM1、PWR_BG_TRIM4 和 PWR_BG_TRIM5 应用于新频率
  - 等待 ≥ 5 µs
  - 将 CLK_IMO_TRIM2 更改成新频率
- 另外
  - 将 CLK_IMO_TRIM2 更改成新频率
  - 等待 ≥ 5 个周期
  - 将 CLK_IMO_TRIM1、PWR_BG_TRIM4 和 PWR_BG_TRIM5 应用于新频率

表 8-1. IMO 频率配置

CLK_IMO_TRIM2						频率 （MHz）
位 5	位 4	位 3	位 2	位 1	位 0	
0	0	0	0	1	1	3
0	0	0	1	0	0	4
0	0	0	1	0	1	5
0	0	0	1	1	0	6
0	0	0	1	1	1	7
0	0	1	0	0	0	8
0	0	1	0	0	1	9
0	0	1	0	1	0	10
0	0	1	0	1	1	11
0	0	1	1	0	0	12
0	0	1	1	1	0	13
0	0	1	1	1	1	14
0	1	0	0	0	0	15
0	1	0	0	0	1	16
0	1	0	0	1	0	17
0	1	0	0	1	1	18
0	1	0	1	0	0	19
0	1	0	1	0	1	20
0	1	0	1	1	0	21
0	1	0	1	1	1	22
0	1	1	0	0	0	23
0	1	1	0	0	1	24
0	1	1	0	1	1	25
0	1	1	1	0	0	26
0	1	1	1	0	1	27
0	1	1	1	1	0	28
0	1	1	1	1	1	29
1	0	0	0	0	0	30
1	0	0	0	0	1	31
1	0	0	0	1	0	32
1	0	0	0	1	1	33
1	0	0	1	0	1	34
1	0	0	1	1	0	35
1	0	0	1	1	1	36
1	0	1	0	0	0	37
1	0	1	0	0	1	38
1	0	1	0	1	0	39
1	0	1	0	1	1	40
1	0	1	1	1	0	41
1	0	1	1	1	1	42
1	1	0	0	0	0	43
1	1	0	0	0	1	44
1	1	0	0	1	0	45
1	1	0	0	1	1	46
1	1	0	1	0	0	47
1	1	0	1	0	1	48

8.2.1.1 启动行为

复位后，IMO 的频率被配置为 24 MHz。在启动的 “ 引导 ” 过程中，可从闪存读取调整值，并配置 IMO，以满足数据手册指定的准确度。

8.2.1.2 IMO 频率扩展

IMO 能在扩频模式下运行，以减少在 IMO 中心工作频率中生成的噪声振幅。该模式使 IMO 频率在寄存器选择的四种分布之一不相同。四种分布包括：固定频率、三角波、伪随机及 DSI 输入。DSI 输入模式允许您使用数字信号来指定 IMO 频率分布的形状。分布模式由寄存器 CLK_IMO_SPREAD 位 SS_MODE 选择（如表 8-2 所示）。分布限制由寄存器 CLK_IMO_SPREAD 位 SS_RANGE 指定 （如表 8-3 所示）。所有扩展选项均为向下扩展，表示瞬间时钟频率值总是等于或低于配置频率。

表 8-2. IMO 扩频分布模式位 SS_MODE

名称	说明
SS_MODE[1:0]	IMO 扩频模式。定义扩频频率分布的形状。 0：关闭。 IMO 频率不变。 1：三角形。 IMO 频率根据中心频率形成一个三角形的分布。计数限制由位 SS_MAX 定义。 2：使用 LFSR 的伪随机序列。 IMO 频率根据中心频率形成一个伪随机分布。 3：DSI。通过使用 DSI 输入信号可确定 IMO 频率分布。

表 8-3. IMO 扩频分布范围位 SS_RANGE

名称	说明
SS_RANGE[1:0]	IMO 扩频的最大范围。定义了扩频计数器到达极限计数值时标称频率的扩展。 0：1%。扩频变化在极限计数值上的频率范围为 0 至 –1%。 1：2%。扩频变化在极限计数值上的频率范围为 0 至 –2%。 2：4%。扩频变化在极限计数值上的频率范围为 0 至 –4%。 3：保留。请勿使用。

CLK_IMO_SPREAD 寄存器中的 SS_MAX 字段设置了扩频计数器的最大计数值。当将 SS_MODE 设置为一个三角形扩频时，增大该值会延长三角形扩频整个周期时间。

IMO 扩频逻辑需要一个与它连接的时钟，以便获取 IMO 扩频性能。该逻辑将外设时钟 0 作为它的时钟。 IMO 扩频需要外设时钟 0 被路由到外设时钟分频器的相应时钟。该时钟的频率决定了扩频逻辑的速率以及频率的变化率。

8.2.1.3 编程时钟 （36 MHz）

IMO 模块具有一个 36 MHz 的输出，该输出可作为闪存编程模块的时钟使用。该时钟仅适用于闪存编程模块，并不能作为任意时钟分频器或时钟树的时钟源。

8.2.2 内部低速振荡器

内部低速振荡器运行时不需要使用外部器件，并会输出一个 32 kHz 的稳定时钟。ILO 是功耗相对低且精度较低的振荡器。可以使用精度和频率更高的时钟对 ILO 进行校准，从而提高该振荡器的精度。 ILO 在所有功耗模式（休眠和停止模式除外）下均可用。在该器件中， ILO 作为系统低频时钟 LFCLK 源使用。该 ILO 是一个相对不够准确 （过压和过温时精度为 ±60%）的振荡器，可用于生成低频率时钟。如果使用 IMO 进行校准，则在稳定的温度和电压条件下， ILO 的精度误差仅为 ±10%。通过使用 CLK_ILO_CONFIG 寄存器中的 ENABLE 位使能 / 禁用 ILO。

8.2.3 外部时钟 （EXTCLK）

外部时钟 （EXTCLK）是 MHz 范围的时钟，它可由指定的 PSoC 4 引脚上的信号生成。该时钟可取代 IMO，作为系统高频率时钟 HFCLK 源使用。外部时钟频率的可用范围为 0 ~ 48 MHz。该器件始终使用 IMO 启动，并且必须在用户模式下使能外部时钟。因此，不能通过由外部时钟提供脉冲的复位信号来启动器件。

当手动将引脚配置为 EXTCLK 的输入时，必须将引脚的驱动模式设置为数字高阻态，以使能数字输入缓冲区。有关更多信息，请参考第 69 页上的 I/O 系统章节。

8.2.4 外部晶振 （ECO）

不带外部组件的外部晶振用于生成一个高精度时钟。它主要用于给 BLE 子系统（包括 Link Layer 引擎、数字 PHY 调制解调器及 RF 收发器）提供时钟脉冲。高精度 ECO 时钟还能作为 PSoC 4 器件的时钟源使用。通过使用寄存器 BLE_BLERD_DBUS 的 XTAL_ENABLE 位使能及禁用 ECO。

8.2.4.1 ECO 负载电容调试

PSoC 4 器件包括一个可变的集成负载电容，用于调试生产线上的时钟频率。通过使用 BLE_BLERD_BB_XO_CAPTRIM 寄存器上的 X2 和 X1 位，可以单独控制 X1 和 X2 节点的负载电容，如表 8-4 所示。

表 8-4. 电容微调选择位 X2 和 X1

名称	说明
X2[7:0]	7:0 位：同时控制 X2 节点上的电容值。 6-0 位：微调控制 6-0 值：电容值 0：3.6900 pF 1：3.7911 pF 2：3.8922 pF . . . 127：16.4280 pF 7 位：粗调控制 0：不使用附加电容 1：使用 8.1 pF 的附加电容
X1[15:8]	7:0 位：同时控制 X1 节点上的电容值。 6-0 位：微调控制 6-0 值：电容值 0：3.6900 pF 1：3.7911 pF 2：3.8922 pF . . . 127：16.4280 pF 7 位：粗调控制 0：不使用附加电容 1：使用 8.1 pF 的附加电容

8.2.5 监视晶体振荡器 （WCO）

PSoC 器件包含一个振荡器，用于驱动 32.768 KHz 的监视晶振。它还作为 LFCLK 的源使用。WCO 用于在低功耗睡眠模式下准确保持广播事件及连接事件的时间间隔。与 ILO 相同，WCO 在所有模式（休眠和停止模式除外）下均可用。该时钟的功耗较低，因此它非常适合低功耗模式 （如深度睡眠模式）。通过 WCO_CONFIG 寄存器中的 ENABLE 位使能或禁用 WCO。

8.3 时钟分布

PSoC 4 时钟在整个器件中开发和分布，如图 8-1 所示。分布配置有以下的选项：

- HFCLK 输入选择
- LFCLK 输入选择
- ECO 分频器配置
- SYSCLK 预分频器配置
- 外设分频器配置

8.3.1 HFCLK 输入选择

PSoC 4 中的 HFCLK 共有三个输入选项：IMO、 EXTCLK 及 ECO。通过使用 CLK_SELECT 寄存器的 DIRECT_SEL 位选择 HFCLK 输入，如表 8-5 中所示。

表 8-5. HFCLK 输入选择位 DIRECT_SEL

名称	说明
DIRECT_SEL[2:0]	HFCLK 输入时钟选择 0：IMO。将 IMO 作为 HFCLK 的时钟源使用 1：EXTCLK。 EXTCLK 作为 HFCLK 源使用 2：ECO。 ECO 作为 HFCLK 源使用 3–7：保留。请勿使用

8.3.2 LFCLK 输入选择

PSoC 4 的 LFCLK 具有两个输入选项：ILO 和 WCO。通过使用 WDT_CONFIG 寄存器的 LFCLK_SEL 位选择 LFCLK，如表 8-6 所示。如果在 LFCLK 边沿附近更改了该选择，那么 LFCLK 选择可能出现故障。 LFCLK 为看门狗定时器 （WDT）提供时钟脉冲；因此，要确保 LFCLK 源的切换不会影响到 WDT。为了安全更改 LFCLK_SEL，需要等待 WDT_CTRLOW/WDT_CTRHIGH 更改，然后立即更改该设置。

表 8-6. LFCLK 输入选择位 LFCLK_SEL

名称	说明
LFCLK_SEL[1:0]	LFCLK 输入时钟选择 0：ILO。内部本机振荡器作为 LFCLK 源使用 1：WCO。监视晶振作为 LFCLK 源使用 2-3：保留。请勿使用

8.3.3 ECO 分频器配置

ECO 分频器允许将 ECO 时钟作为 HFCLK 使用前，器件会对该时钟进行分频。分频器可对 ECO 时钟进行 2 的 0 ~ 3 次方分频。通过使用寄存器 BLE_BLERD_XTAL_CLK_DIV_CONFI 位 SYSCLK_DIV 设置分频值，如表 8-7 所示。默认情况下，将 ECO 分频器设置为 0 （HFCLK = XTAL_CLK）。

表 8-7. ECO 分频器选择位 SYSCLK_DIV

名称	说明
SYSCLK_DIV[1:0]	ECO 时钟分频器。对频率为 24 MHz 的晶振时钟进行分频，以生成 HFCLK。 0：NO_DIV: HFCLK = XTAL_CLK/1 1：DIV_BY_2: HFCLK = XTAL_CLK/2 2：DIV_BY_4: HFCLK = XTAL_CLK/4 3：DIV_BY_8: HFCLK = XTAL_CLK/8

8.3.4 SYSCLK 预分频器配置

通过 SYSCLK 预分频器，将 HFCLK 作为 SYSCLK 使用之前，器件可对它进行分频，这样允许外设时钟和系统时钟之间为非整数倍。SYSCLK 时钟频率需要大于或等于器件中由 HFCLK 派生的所有其他时钟频率。SYSCLK 预分频器可对 HFCLK 进行 2 的 0 ~ 7 次方分频。通过使用寄存器 CLK_SELECT 的位 SYSCLK_DIV 设置预分频器分频值，如表 8-8 所示。预分频器最初配置的分频值是 1。

表 8-8. SYSCLK 预分频器分频值位 SYSCLK_DIV

名称	说明
SYSCLK_DIV[3:0]	SYSCLK 预分频器的分频值 0：SYSCLK = HFCLK 1：SYSCLK = HFCLK/2 2：SYSCLK = HFCLK/4 3：SYSCLK = HFCLK/8 4：SYSCLK = HFCLK/16 5：SYSCLK = HFCLK/32 6：SYSCLK = HFCLK/64 7：SYSCLK = HFCLK/128

8.3.5 外设时钟分频器配置

PSoC 4 共有 12 个时钟分频器，具体包括 10 个 16 位时钟分频器和 2 个 16.5 位的小数时钟分频器。小数时钟分频器可为时钟分频器提供 0..31/32 的小数分频。小数分频器的输出频率的公式为：Fout = Fin / （INT16_DIV + （FRAC5_DIV/32））。例如，16.5 位分频器对 48 MHz HFCLK 时钟进行 3 整数分频（INT16_DIV = 3，FRAC5_DIV = 0），这样便能生成 16 MHz 的时钟。16.5 位分频器可以对 48 MHz HFCLK 时钟进行 4 整数分频（INT16_DIV = 4，FRAC5_DIV = 0），从而生成 12 MHz 的时钟。 16.5 位分频器对 48 MHz HFCLK 时钟进行 3 整数分频 （INT16_DIV = 3）和 16 的小数分频 （FRAC5_DIV = 16），从而生成一个 13.7 MHz 的时钟。并非所有 13.7 MHz 时钟都有相同的周期。一半为 3 个 HFCLK 周期，另一半则为两个 HFCLK 周期。

如果需要使用一个高精度时钟 （如供给 UART/SPI 串行接口），则推荐使用小数分频器。因为时钟周期带有 1 个 HFCLK 周期的抖动，所以需要一个低抖动时钟时，避免使用小数分频器。

分别使用 PERI_DIV_16_CTLx 和 PERI_DIV_16_5_CTLx 寄存器配置 10 个非小数时钟分频器和 2 个小数时钟分频器。表 8-9 和表 8-10 显示了这些寄存器的配置情况。

通过使用 PERI_DIV_CMD 寄存器，可以使能所有分频器。该寄存器可作为所有 16 个整数分频器和 4 个小数分频器的指令寄

表 8-9. 外设时钟整数分频器配置寄存器 PERI_DIV_16_CTLx

位	名称	说明
0	ENABLE_x	分频器被使能。硬件通过 ENABLE 命令将该字段设置为 1 ；也可以通过 DISABLE 指令将该字段设置为 ‘0’ 。
23:8	INT16_DIV_x	分频值为 （1+INT16_DIV）的整数分频 。整数分频范围为 [2, 65,536]。

表 8-10. 外设时钟小数分频器配置寄存器 PERI_DIV_16_5_CTLx

位	名称	说明
0	ENABLE_x	分频器被使能。硬件通过ENABLE命令将该字段设置为1；也可以通过DISABLE指令将该字段设置为0。
7:3	FRAC5_DIV_x	小数分频值为 （FRAC5_DIV/32）。小数分频值范围为 [0, 31/32]。 请注意，因为某些时钟周期比其它时钟周期多一个 “clk_hf” 周期，所以小数分频会产生时钟抖动。
23:8	INT16_DIV_x	整数分频值为 （1+INT16_DIV）。整数分频的范围为 [1, 65,536]。

存器。 PERI_DIV_CMD 寄存器的结构如下所示。

Bit	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
Description	Enable	Disable															PA_SEL_TYPE		PA_SEL_DIV						SEL_TYPE		SEL_DIV					

SEL_TYPE 字段指定了所配置的分频器类型。对于 16 位整数分频器，该字段为 ‘1’ ；对于 16.5 位小数分频器，该字段被设置为 ‘2’ 。

SEL_DIV 字段指定了被配置的指定分频器数量。对于整数分频器，该字段的取值范围为 0 到 15。对于小数分频器，该字段的取值范围为 0 到 3。当 SEL_TYPE = 63 和 SEL_TYPE = 3 时，没有指定任何分频器。

（PA_SEL_TYPE， PA_SEL_DIV）字段对可使某一个分频器与其它分频器的相位对齐。 PA_SEL_DIV 指定了被相位对齐的分频器。任意已使能的分频器都可作为一个参考分频器。 PA_SEL_TYPE 指定了被相位对齐的分频器类型。当 PA_SEL_DIV = 63、 PA_SEL_TYPE = 3 时， HFCLK 可作为参考时钟。

例如，一个 48 MHz 的 HFCLK，用于生成一个 12 MHz 的分频时钟 A 和一个 8 MHz 的分频时钟 B。HF_CLK 时钟使用一个 16 位整数分频器 0 可生成时钟 A。这时，（（PA_SEL_TYPE， PA_SEL_DIV）为 （3， 63））， 并且 DIV_16_CTL0.INT16_DIV 为 3。时钟 A 使用一个整数分频器 1 可生成时钟 B。这时 （（PA_SEL_TYPE， PA_SEL_DIV）为 （1， 0）），并且 DIV_16_CTL1.INT16_DIV 为 5。这样可保证时钟 B 与时钟 A 相位对齐，因为两个时钟周期的最小公倍数为 12 个 HFCLK 周期。因此每经过 12 个 HFCLK 周期，时钟 A 和时钟 B 都会对齐一次。请注意，时钟 B 与时钟 A 的相位相互对齐，但它们仍将 HFCLK 作为参考时钟进行分频。

PSoC 中每个外设模块都有唯一一个与它相关联的外设时钟 （PER#_CLK）。每个外设时钟都有一个复用输入，可从任意一个现存的时钟分频器提取出输入时钟。

表 8-11 显示 外设输出与相应外设模块的映射情况 （如图 8-1 所示）。通过使用相应的 PERI_PCLK_CTLx 寄存器，可以将任意 12 个数字外设时钟中的一个映射到一个特定的数字外设内，如表 8-12 所示。

表 8-11. 外设时钟复用器输出映射

PERI#_CLK	外设
0	IMO （扩频）
1	SCB0
2	SCB1
3	CLOCK_PUMP
4	CSD （1）
5	CSD （2）
6	SAR
7	TCPWM0

表 8-11. 外设时钟复用器输出映射

PERI#_CLK	外设
8	TCPWM1
9	TCPWM2
10	TCPWM3
11	UDB0 （仅在 PSoC 42xx 中可用）
12	UDB1 （仅在 PSoC 42xx 中可用）
13	UDB2 （仅在 PSoC 42xx 中可用）
14	UDB3 （仅在 PSoC 42xx 中可用）
15	LCD

表 8-12. 可编程时钟控制寄存器 — PERI_PCLK_CTLx

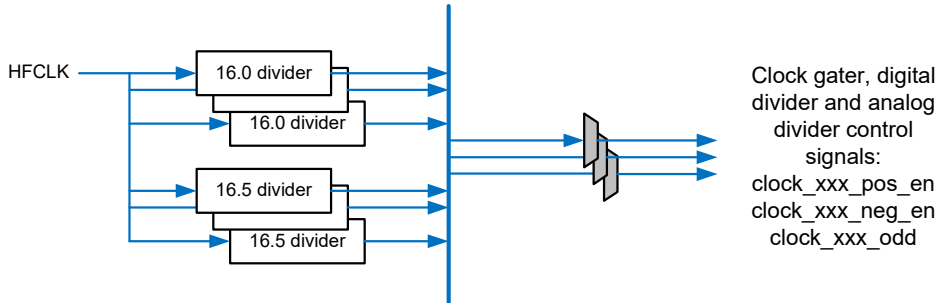
位	名称	说明
5:0	SEL_DIV	用于确定一个特定分频器 （该分频器的类型由 SEL_TYPE 指定）。如果 SEL_DIV 为 ‘4’ 、 SEL_TYPE 为 ‘1’ ，那么第五个 （将 ‘0’ 为第一个） 16 位时钟分频器将被路由到复用器输出端，供外设 clock_x 使用。同样，如果SEL_DIV为 ‘0’ 、SEL_TYPE为 ‘2’ ，那么第一个16.5位时钟分频器将被路由到复用器输出端。
7:6	SEL_TYPE	0：请勿使用 1：16.0 （整数）时钟分频器 2：16.5 （小数）时钟分频器 3：请勿使用

8.3.6 外设时钟配置

外设时钟分频器、时钟门控器、数字时钟分频器以及模拟时钟分频器生成最终外设使用的时钟。

外设时钟分频器生成以下 3 种控制信号：clock_xxx_pos_en、clock_xxx_neg_en 及 clock_xxx_odd。时钟门控器、数字时钟分频器以及模拟时钟分频器都使用这些信号和 HFCLK，以便生成所需的时钟信号。图 8-2 提供外设时钟分频器的概述。

图 8-2. 外设时钟分频器



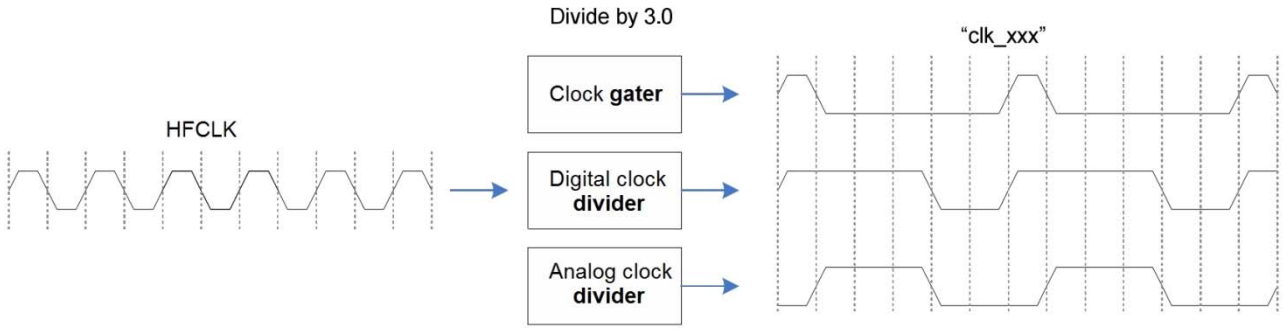
外设时钟分频器包括两层：

- 时钟分频器会根据一个整数分频器和一个可选小数时钟分频器值生成时钟信号。
- 时钟复用器从一个特定时钟分频器中选择信号，然后将三种时钟信号提供给特定的最终外设。

8.3.6.1 时钟生成

图 8-3 提供关于通过使用时钟门控器、数字时钟分频器及模拟时钟分频器生成时钟的概述。

图 8-3. 时钟的生成 （通过使用时钟门控器、数字时钟分频器及模拟时钟分频器）



- 时钟门控器：该组件用于门控输入时钟源。“Clock_xxx_pos_en” 信号用于使能时钟门控。该组件会生成单一的高脉冲，该脉冲的时间等于 HFCLK 高脉冲的时间。该逻辑通常使用于只有上升沿触发器的数字逻辑。
- 数字时钟分频器：该组件通过使用 “clock_xxx_pos_en” 及 “clock_xxx_neg_en” 生成时钟源。偶整数分频器生成的时钟均有 50% 占空比。奇整数分频器生成的时钟不具有 50% 的占空比；低电平时间将比高电平时间多 1 个周期。该组件使用于具有上升沿和下升沿触发器的数字逻辑 （比如某个 UDB 阵列）。
- 模拟时钟分频器：该组件通过使用 “clock_xxx_pos_en”，、“clock_xxx_neg_en” 及 “clock_xxx_odd” 信号生成时钟源。整数分频器 （偶数及奇数分频器）生成的时钟均有 50% 占空比。

8.4 低功耗模式操作

高频时钟 （包括 IMO、EXTCLK、ECO、HFCLK、SYSCLK）和外设时钟仅在活动模式和睡眠模式下运行。 ILO、 WCO 及 LFCLK 可在所有功耗模式 （休眠模式及停止模式除外）下运行。

8.5 寄存器列表

表 8-13. 时钟系统寄存器列表

寄存器名称	说明
CLK_IMO_TRIM1	IMO 调整寄存器 — 该寄存器包含 IMO 调整，允许对其频率进行细调。
CLK_IMO_TRIM2	IMO 频率选择寄存器 — 该寄存器控制 IMO 的频率范围，允许对其频率粗调。
PWR_BG_TRIM4	带隙调整寄存器 — 这些寄存器控制带隙参考的调整，允许进行器件中相关参考电压的操作。
PWR_BG_TRIM5	
CLK_IMO_SPREAD	IMO 扩频控制寄存器 — 该寄存器控制着 IMO 的扩频功能。
BLE_BLERD_DBUS	ECO 被使能。该寄存器包含用于禁用和使能 ECO 的位。
BLE_BLERD_BB_XO_CAP-TRIM	X1 及 X2 电容调整寄存器 — 该寄存器控制着 ECO 的 X1 及 X2 节点的电容值。
CLK_ILO_CONFIG	ILO 配置寄存器 — 该寄存器用于控制 ILO 的配置。
CLK_IMO_CONFIG	IMO 配置寄存器 — 该寄存器控制着 IMO 的配置。
CLK_SELECT	时钟选择 — 该寄存器控制着时钟树配置，用以选择不同的系统时钟源。
WDT_CONFIG	该寄存器用于选择 LFCLK 的时钟源。
WCO_CONFIG	WCO 使能。该寄存器用于禁用 / 使能外部监视晶振。
BLE_BLERD_XTAL_-CLK_DIV_CONFIG	ECO 分频器。该寄存器为 ECO 时钟选择分频值。
PERI_DIV_16_CTLx	外设时钟分频器控制寄存器 — 这些寄存器可配置外设时钟分频器，用以设置整数分频值并使能或禁用分频器。
PERI_DIV_16_5_CTLx	外设时钟小数分频器控制寄存器 — 这些寄存器可配置外设时钟分频器，用以设置小数分频值以及使能或禁用分频器。
PERI_PCLK_CTLx	可编程时钟控制寄存器 — 这些寄存器用于选择外设的输入时钟。

9. 电源供应和监控

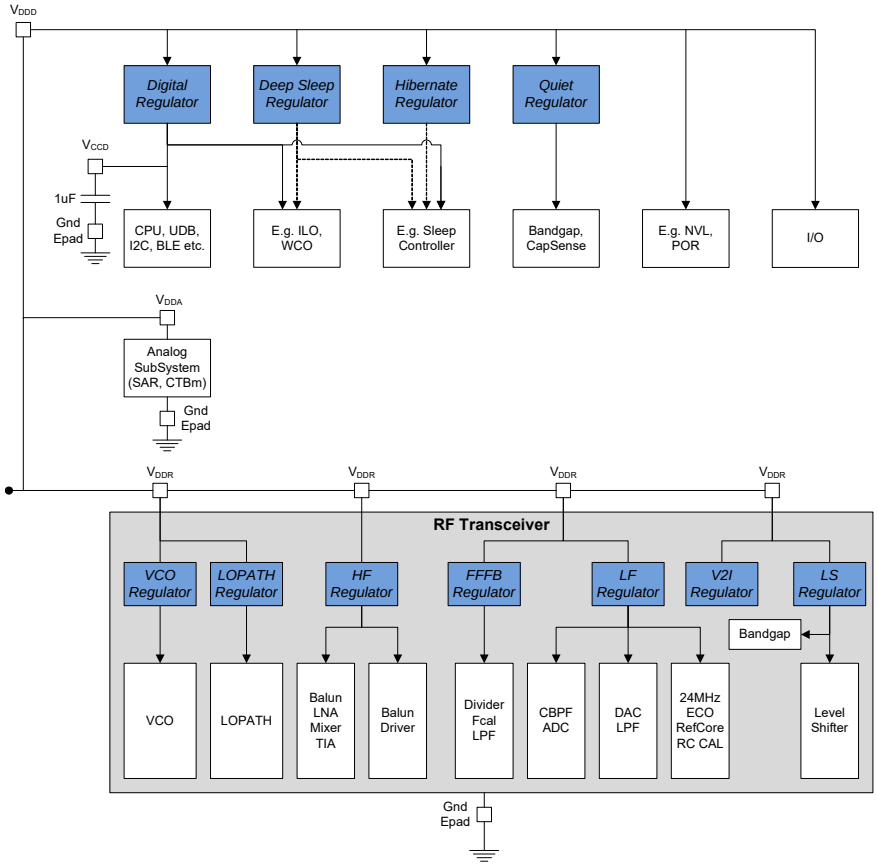
通过直接将电压为 1.9 V 到 5.5 V 的外部供电电压提供给数字供电 （VDDD）、模拟供电 （VDDA）和无线供电 （VDDR）引脚，可以使 PSoC® 4 运行。

请注意，根据器件封装中的可用性，PSoC 4 器件支持多种电源轨 — V_{DDA}、V_{DDD} 和 V_{DDIO}。这些电源轨相互独立，并且可以单独将它们连接到所需的独立电源。即使在直接供电电压模式下，只需要将 V_{DDD} 限制在 1.71 V 到 1.89 V 的范围内，而其他电源轨 （如 V_{DDA}/V_{DDIO}）可以在整个 1.71 V 到 5.50 V 的范围内。更多有关各种电源轨和它们的使用情况的信息，请参考器件数据手册。

根据不同的功耗模式和 BLE 状态，会提供不同的内部电压调节器。这些电压调节器包括：活动数字电压调节器、低噪声电压调节器、深度睡眠电压调节器以及休眠电压调节器，可使用这些电压调节器来支持内核功能。此外，VCO 电压调节器、LOPATH 电压调节器、HF 电压调节器、FFFB 电压调节器、LF 电压调节器、V2I 电压调节器和 LS 电压调节器支持不同的 BLE 功耗状态。

9.1 框图

图 9-1. 电源系统框图



电源系统包括独立的数字、模拟和无线供电引脚，它们分别被标志为 V_{DDD}、V_{DDA} 和 V_{DDR} 的，如图 9-1 所示。所有的接地引脚均被连接到芯片的 E-PAD。数字接地和 CapSense 接地也被连接到标志为 V_{SS} 和 V_{SSA} 的引脚。数字和模拟供电引脚共用了 ESD 源。为避免打开 ESD 器件，即使在暂时转换过程中，数字供电和模拟供电相差不能超过 300 mV。因此，先要给 V_{DDA} 供电，并要求使其电源上升（ramp up）到一定的值时才为 V_{DDD} 和 V_{DDR} 供电。

通过一个处于活动状态的数字电压调节器，可以将外部 V_{DDD} 供电电压调整到数字内核所需额定的 1.8 V。该电压调节器的输出引脚（V_{CCD}）具有特殊的电容要求，如图 9-1 所示。该活动状态数字调节器专门为内部电路供电，不能连接外部负载。

9.2 工作原理

图 9-1中的电压调节器为器件的不同区域供电。所有内核电压调节器都由 V_{DDD}电 源引脚供电。模拟电路则直接由 V_{DDA} 输入端供电。

9.2.1 电压调节器汇总

9.2.1.1 内核电压调节器

在活动或睡眠功耗模式下，活动数字电压调节器和低噪声电压调节器都处于使能状态。在深度睡眠和休眠功耗模式下，它们均被禁用（见表 9-1 和图 9-1）。深度睡眠和休眠状态电压调节器旨在满足器件在各低功耗模式下的功耗要求。

表 9-1. 不同功耗模式下的内核电压调节器状态

模式	活动电压调节器	低噪声电压调节器	深度睡眠电压调节器	休眠电压调节器
停止	关闭	关闭	关闭	关闭
休眠	关闭	关闭	关闭	启用
深度睡眠	关闭	关闭	启用	启用
睡眠	启用	启用	启用	启用
活动	启用	启用	启用	启用

活动数字电压调节器

在活动模式和睡眠模式下，活动数字电压调节器提供了主数字逻辑。它可以将电压范围为 1.9 V 到 5.5 V 的外部 VDDD 调整为数字内核所需的额定 1.8 V。该电压调节器的输出被连接到 VCCD 引脚，并且需要使用一个外部去耦电容（1 µF X5R）。

活动数字电压调节器仅在活动和睡眠功耗模式下可用。

低噪声电压调节器

在活动和睡眠模式中，该电压调节器为模拟电路（如带隙参考和电容式感测子系统）供电。这些模拟电路需要低噪声电源、无数字切换噪声以及无电源噪声。该电压调节器具有高电源抑制比。

低噪声电压调节器仅在活动和睡眠模式下可用。

深度睡眠状态电压调压器

该电压调节器将供电给在深度睡眠模式下保持上电状态的电路，如 ILO、WCO 和 SCB。除了休眠模式以外，深度睡眠电压调节器在所有功耗模式中都可用。在活动和睡眠功耗模式下，该电压调节器的主输出被连接至数字电压调节器的输出（VCCD）。该电压调节器还有一个独立的副本输出，它为ILO 和 WCO 振荡器提供了稳定的电压。在活动和睡眠模式下，该输出未连接至 VCCD。

休眠电压调节器

该电压调节器将供电给在休眠模式下保持上电的电路，如睡眠控制器、低功耗比较器以及 SRAM。休眠电压调节器在所有功耗模式中均可用。在活动和睡眠模式中，该电压调节器的输出被连接至数字电压调节器的输出端。在深度睡眠模式中，该调节器的输出被连接至深度睡眠电压调节器的输出端。

9.2.1.2 RF 收发器电压调节器

RF 收发器电压调节器供电给 RF 收发器中的不同模块，如图 9-1 所示，VDDR 将范围为 1.9 V 到 5.5 V 的输入电压提供给不同的电压调节器。RF 收发器电压调节器将从 VDDR 生成 1.8 V 的调节电压。

七个 RF 收发器电压调节器中的三个在所有 BLE 模式下均可用（BLE_DeepSleep 模式除外）。剩下的电压调节器仅在 BLE_RX 和 BLE_TX 模式下可用（见表 9-2）。有关 BLE 模式的更多信息，请参考第 227 页上的蓝牙低功耗子系统（BLESS）章节。

表 9-2. 在不同 BLE 模式下 RF 收发器调节器的状态

模式	LDO_LS	LDO_HF	LDO_VCO	LDO_LOPATH	LDO_FFFB	LDO_V2I	LDO_LF
BLE_DeepSleep	关闭	关闭	关闭	关闭	关闭	关闭	关闭
BLE_Sleep	启用	关闭	关闭	关闭	关闭	启用	启用
BLE_IDLE	启用	关闭	关闭	关闭	关闭	开启	启用
BLE_TX	启用	启用	启用	启用	启用	启用	启用
BLE_RX	启用	启用	启用	启用	启用	启用	启用

电平转换器电压调节器 （LDO_LS）

该电压调节器为 RF 收发器中的电平转换器和带隙生成器供电。电平转换器电压调节器是三个适用于所有模式（BLE_RX、BLE_TX、BLE_Idle 和 BLE_Sleep）的电压调节器中的一个 （BLE_DeepSleep 模式除外）。

高频率电压调节器 （LDO_HF）

该电压调节器给高频率无线电路（如低噪声放大器（LNA）、混频器和巴伦电路）供电。该电压调节器仅在 BLE_RX 和 BLE_TX 模式下可用。

LDO_FFFB

LDO_FFFB 为合成器模块中的电路供电 （如分频器和 VCO 校准模块）。同 LDO_HF 相似，该电压调节器仅在 BLE_RX 和 BLE_TX 模式下可用。

VCO 电压调节器 （LDO_VCO）

LDO_VCO 电压调节器为 RF 收发器的 VCO 部分供电。该电压调节器仅在 BLE_RX 和 BLE_TX 模式下可用。

高速锁存电压调节器 （LDO_LOPATH）

该电压调节器为 RF 收发器中的高速锁存供电。 LDO_LOPATH 仅在 BLE_RX 和 BLE_TX 模式下可用。

V-I 电压调节器 （LDO_V2I）

LDO_V2I 为 RF 收发器中的电压电流转换器电路供电。该电压调节器在所有模式下都可用 （BLE_DeepSleep 模式除外）。

低频率电压调节器 （LDO_LS）

LDO_LS 为 RF 收发器中的低频率电路供电。低频率电路包括 ADC、互补带通滤波器 （CBPF）、DAC、ECO 等。该电压调节器在所有模式下都可用 （BLE_DeepSleep 模式除外）。

9.3 电压监控

电压监控系统包括上电复位 （POR）、欠压检测 （BOD）和低压检测 （LVD）。

9.3.1 上电复位 （POR）

在初次电源上升期间，上电复位 （POR）电路提供了一个复位脉冲。POR 电路用于监控 V_{CCD} 电压。通常，在跳变点上，POR 电路对 VCCD 的监控不是很准确。在芯片的初次上电期间使用 POR 电路，然后将禁用它。

9.3.1.1 欠压检测 （BOD）

通过对器件进行复位， BOD 电路可使处于工作 / 保持状态的逻辑避免发生不安全的供电状态。 BOD 电路用于监控 V_{CCD} 电压。如果电压偏移低于确保安全操作的最低 V_{CCD} 电压，那么，BOD 电路将生成一个复位（有关详细信息，请参考器件数据手册 ）。系统不会退出复位状态，直到供电电压再次进入有效范围为止。

一个特别的寄存器（PWR_BOD_KEY）用于使固件能够区分器件的正常电源周期和欠压事件，此寄存器在由于 BOD 产生的复位后也不会被清除。然而，如果器件发生 POR 或 XRES，该寄存器将被清除。BOD 在所有功耗模式下均可用（停止模式除外）。

9.3.1.2 低压检测 （LVD）

LVD 电路监控外部电源电压，并准确地检测能量源的消耗。LVD 检测器将生成一个中断，以使系统采取预防措施。

LVD 仅在活动和睡眠功耗模式下可用。如果在深度睡眠模式下需要使用 LVD，那么要配置芯片，以便使用 WDT 作为唤醒源来定期将其从深度睡眠模式唤醒。另外 LVD 监控最好在活动模式下实现。 LVD 电路在安全工作电压范围内将以各个可编程的电平生成中断。通过 PWR_VMON_CONFIG 寄存器中的 LVD_SEL 字段，可以将 LVD 的触发点配置为介于 1.75 V 至 4.5 V 之间的范围。

使能 LVD 电路时，在初始建立时间内，可能发生错误中断。置位 PWR_VMON_CONFIG 寄存器中的 LVD_EN 位后，通过等待 1 µs 时间，固件可以屏蔽错误中断。使能 LVD 功能的推荐固件程序如下：

1. 要确保 PWR_INTR_MASK 寄存器中的 LVD 位为 0，以防止传输错误中断信号。
2. 设置 PWR_VMON_CFG 寄存器中 LVD_SEL 字段的所需跳变点。
3. 通过设置 PWR_VMON_CFG 寄存器中的 LVD_EN 位来使能 LVD。这时，可能导致错误 LVD 事件。
4. 等待至少 1 µs 以让电路稳定。
5. 将 ‘1’ 写入到 PWR_INTR 寄存器中的 LVD 位，以清除错误事件。如果发生 LVD 条件，该位将不被清除。
6. 通过 PWR_INTR_MASK 中的 LVD 位，取消屏蔽中断。

9.4 寄存器列表

表 9-3. 电源供应与监控寄存器列表

寄存器名称	说明
PWR_CONTROL	功耗模式控制寄存器 — 通过该寄存器，可以配置器件的功耗模式以及电压调节器的工作。
PWR_INTR	电源系统中断寄存器 — 该寄存器指出电源系统中断状态。
PWR_INTR_MASK	电源系统中断屏蔽寄存器 — 该寄存器控制着哪个中断会被传输到 CPU 的中断控制器内。
PWR_VMON_CONFIG	电源系统电压监控调整与配置 — 该寄存器包含了电压监控系统的调整与配置位。

10. 芯片运行模式

PSoC® 4 可以在四种不同的模式下执行固件。在这些模式下，指令在闪存和 ROM 内执行的位置以及硬件的特权级别也不一样。终端应用只使用其中的三种模式。在固件开发过程中，调试模式专门用于进行调试设计。

PSoC 4 支持以下工作模式：

- 启动模式
- 用户模式
- 特权模式
- 调试模式

10.1 启动模式

启动模式是一种工作模式，在这种模式下通过器件 SROM 中的硬编码 （固定）指令对器件进行配置。复位结束后，如果没有接收到调试序列，芯片将进入该模式。启动模式是一种特权模式。在这种模式下，中断被禁止，从而启动固件可以使器件连续进行操作而不被中断。在启动模式期间，器件从闪存加载硬件调整设置，以保证在上电过程中能正常运行。启动结束时，器件将进入用户模式，并开始执行闪存中的代码。闪存内的代码可能包括 PSoC Creator IDE 自动生成的指令，这些指令可进一步配置器件。

10.2 用户模式

用户模式是一种工作模式，在该模式下，可以执行闪存中存储的普通用户固件。在用户模式下，不能执行 SROM 中存储的代码。该模式下的固件执行操作包括：PSoC Creator IDE 自动生成的固件和用户编写的固件。自动生成的固件可以控制固件的启动以及部分正常操作。完成自己的任务后，启动过程会将控制权交给该模式。

10.3 特权模式

特权模式是一种工作模式，在该模式下，可以执行器件 ROM 中所存储的特殊子程序。用户不能更改这些子程序，它们用于执行不被中断或观察到的专有代码。在特权模式中，不能进行调试。

通过执行一个系统调用，CPU 可以切换到特权模式。更多有关如何执行一个系统调用的信息，请参考第 336 页上的 “ 执行系统调用 ” 一节。退出该模式后，器件将返回用户模式。

10.4 调试模式

调试模式是一种允许观察 PSoC 4 操作参数的工作模式。在开发过程中，该模式用于调试固件。器件复位过程中，在指定时间窗口内，如果器件连接到 SWD 调试器，将进入调试模式。在调试模式下，可以使用 IDE （如 PSoC Creator 或 ARM MDK）调试固件。调试模式只适用于开放模式（四种保护模式之一）下的器件。更多有关调试接口的信息，请参考第 327 页上的编程与调试接口章节的内容。

更多有关保护模式的信息，请参考第 115 页上的器件安全性章节。

11. 功耗模式

PSoC® 4 支持五种功耗模式，从而最小化某个具体应用的平均功耗。按功耗从大到小的顺序，功耗模式依次为：

- 活动模式
- 睡眠模式
- 深度睡眠模式
- 休眠模式
- 停止模式

活动、睡眠及深度睡眠模式是 ARM 所定义的标准功耗模式，这些模式由 ARM CPU 和指令集架构 （ISA）支持。休眠和停止模式是 PSoC 4 所支持的附加低功耗模式。与深度睡眠模式相同，可以通过固件进入这些模式。但被唤醒时，CPU 和所有外设都被复位。

可通过下列方法控制不同功耗模式下的功耗：

- 使能 / 禁用外设
- 启用 / 关闭内部电压调节器
- 启用 / 关闭时钟源
- 启用 / 关闭 PSoC 4 的部分组件

图 11-1 描述了各种功耗模式以及它们之间的转换状态。

图 11-1. 功耗模式转换状态框图

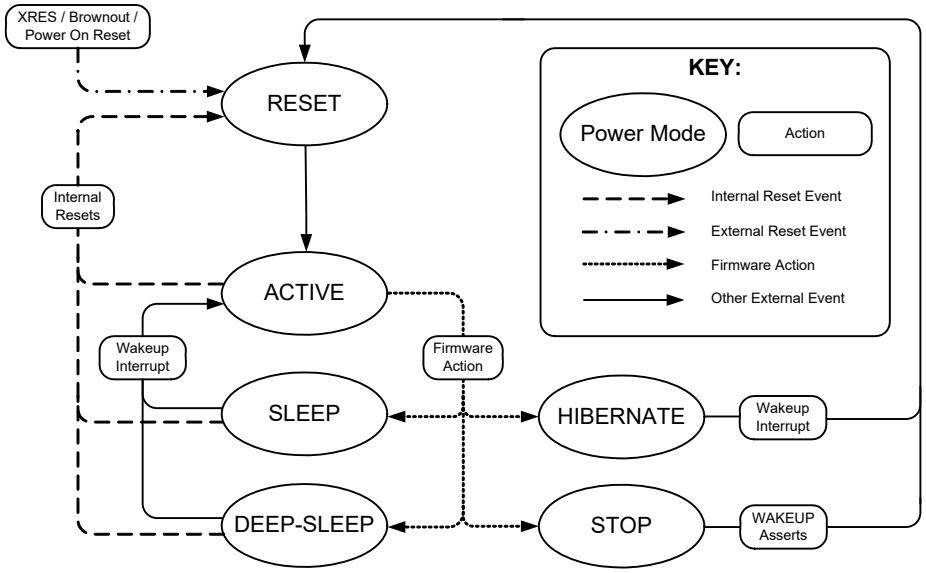


表 11-1 描述了 PSoC 4 所支持的各种功耗模式。

表 11-1. PSoC 4 功耗模式

功耗模式	说明	进入条件	唤醒源	活动时钟	唤醒事件	可用的电压调节器
活动模式	主要工作模式；所有外设都可用 （可编程）。	器件从其它功耗模式唤醒时，或发生内部和外部复位、欠压复位、上电复位等事件后	不适用	所有时钟 （可编程）	中断	所有电压调节器均可用。如果使用外部电压调节器，则可以禁用活动状态数字电压调节器。
睡眠模式	CPU 进入睡眠模式，SRAM 被暂停；所有外设均可用（可编程）。	手动写入寄存器	任意中断	所有时钟 （可编程）	中断	所有电压调节器均可用。如果使用外部电压调节器，则可以禁用活动状态数字电压调节器。
深度睡眠模式	所有内部电源都来自深度睡眠电压调节器。IMO 和高速外设均被关闭。只有低频（32 kHz）时钟可用。 通过低速、异步或低功耗模拟外设的中断，可以实现唤醒操作。	手动写入寄存器	GPIO 中断、低功耗比较器、SCB、看门狗定时器	ILO （32 kHz）、WCO （32 kHz）	中断	深度睡眠电压调节器和休眠电压调节器
休眠模式	仅保持 SRAM 和 UDB，所有内部供电电源（休眠供电电源除外）均被断开。通过引脚中断或低功耗比较器可以唤醒器件。	手动写入寄存器	GPIO 中断，低功耗比较器	无	复位（保持中断状态）	休眠状态电压调节器
停止模式	所有内部供电电源均被断开。仅保持 GPIO 状态。只能通过 XRES 或 WAKEUP 引脚唤醒器件。	手动写入寄存器	WAKEUP 引脚	无	复位	无

除了表 11-1 中所讲述的唤醒源外，可通过外部复位 （XRES）和欠压复位使器件从任意一种功耗模式转换为活动模式。有关 BLE-SS 状态功耗模式的更多信息，请参考 第 227 页上的蓝牙低功耗子系统 （BLESS）章节。

11.1 活动模式

活动模式是 PSoC 器件的基本功耗模式。在该模式下，器件中所有子系统 / 外设均可用。在该模式下，CPU 正在执行且所有外设都被供电。通过配置固件可以禁用未使用的特定外设，从而降低功耗。

11.2 睡眠模式

该功耗模式以 CPU 为中心。在该模式下，Cortex-M0 CPU 进入睡眠模式，且 CPU 时钟被禁用。为了降低功耗，通常器件要保持为睡眠模式，或 CPU 闲置后立即进入该模式。对于外设来说，该模式相当于活动模式。所有已使能的中断均能将器件从睡眠模式唤醒。

11.3 深度睡眠模式

在深度睡眠模式下，CPU、SRAM、UDB 以及高速逻辑均保持活动状态。高频时钟（包括 HFCLK 和 SYSCLK）被禁用。可选用内部低频（32 kHz）振荡器和监视晶体振荡器（WCO），使低频外设继续正常工作。无需时钟或使用外部接口（例如，I^2C 从设备）提供的时钟的数字外设仍持续运行。来自低速、异步或低功耗模拟外设的中断可以使器件从深度睡眠模式唤醒。CTBm 也可以在该模式下运行 （功耗及其带宽在这种情况下被降低）。有关功耗和 CTBm 带宽的更多信息，请参见器件数据手册。

表 11-3 中列出了可用的唤醒源。

11.4 休眠模式

这是 PSoC 4 中保持 SRAM 数据的最低功耗模式。通过关闭所有时钟和断开 CPU 及所有外设的电源（能够通过外部事件来唤醒系统的一些（异步）外设除外），可以达到最低功耗。请注意，在该模式下， CPU 和所有外设的状态不被保留。

该模式使用了容量有限的休眠状态电压调节器，从而能够实现极低功耗。因此，在休眠模式下，输入引脚上所有信号的最大频率将受到限制。所有 I/O 引脚的总切换率（所有输入和输出引脚信号的总频率）不能超过 10 kHz。

对于信号高速切换的系统，可以使用深度睡眠模式，而总功耗几乎不变。

只能通过引脚中断或低功耗比较器使器件从休眠模式中唤醒。从休眠模式中唤醒后将引起一个复位事件，而不是一个中断唤醒事件。从休眠模式唤醒时，CPU 和大多数外设都处于复位状态，且固件将从复位向量处开始执行。除非在进入休眠模式前固件已经明确冻结了 I/O 引脚，否则该复位会使它们处于三态。要想了解中断原因，请使用 PWR_STOP 寄存器中的 TOKEN 位，如第 105 页上的 “ 低功耗模式进入和退出 ” 一节中所述。

外部复位（XRES）将触发整个系统的重启事件。在这种情况下，器件重启后，将无法读取唤醒原因，且 I/O 引脚不会保留其 “ 冻结 ” 状态。

11.5 停止模式

在停止模式下， CPU、所有内部电压调节器以及所有外设均被关闭。从停止模式中唤醒是一个系统复位事件，并且只能通过 XRES 或 WAKEUP 引脚实现。进行复位后，I/O 引脚将处于三态（除非在进入停止模式之前，固件已经明确冻结了这些引脚）。要想了解中断原因，请使用 PWR_STOP 寄存器中的 TOKEN 位，如第 105 页上的 “ 低功耗模式进入和退出 ” 一节中所述。

外部复位（XRES）将触发整个系统的重启事件。在这种情况下，器件重启后，将无法读取唤醒原因，且 I/O 引脚不会保留其 “ 冻结 ” 状态。

11.6 功耗模式总结

表 11-3 说明了每个低功耗模式下可用的外设；表 11-3 则描述了每个功耗模式下可用的唤醒源。

表 11-2. 可用的外设

外设	活动模式	睡眠模式	深度睡眠模式	休眠模式	停止模式
CPU	启用	保持[a]	保持	关闭	关闭
SRAM	启用	保持	保持	保持	关闭
高速外设 （由 HFCLK 提供时钟的外设）	启用	启用	保持	关闭	关闭
CapSense	启用	启用	保持	关闭	关闭
通用数字模块 （UDB）	启用	启用	保持	关闭	关闭
低速外设 （由 ILO 提供时钟的外设）	启用	启用	启用 （可选）	关闭	关闭
内部主振荡器 （IMO）	启用	启用	关闭	关闭	关闭
内部低速振荡器 （32 kHz 的 ILO）	启用	启用	启用 （可选）	关闭	关闭
异步外设	启用	启用	启用	关闭	关闭
上电复位，欠压检测	启用	启用	启用	关闭	关闭
常规模拟外设	启用	启用	关闭	关闭	关闭
低功耗比较器	启用	启用	启用	启用	关闭
CTBm	启用	启用	启用	关闭	关闭
GPIO 输出状态	启用	启用	启用 / 冻结	冻结[b]	冻结

a. 外设的配置和状态被保留。当器件进入活动模式时，外设继续进行它的操作。
b. 系统中所有 GPIO 的配置、模式和状态都被锁定。器件进入活动模式前，将无法修改 GPIO 的状态。

表 11-3. 唤醒源

功耗模式	唤醒源	唤醒事件
睡眠	任意中断源	中断
	任意复位源	复位
深度睡眠	GPIO 中断	中断
	低功耗比较器	中断
	I2C 地址匹配	中断
	看门狗定时器	中断 / 复位
	XRES （外部复位引脚）[a]、欠压	复位
	CTBm	中断
	BLESS	中断
休眠	GPIO 中断	复位
	低功耗比较器	复位
	XRES （外部复位引脚）[a]、欠压	复位
停止	WAKEUP 引脚	复位
	XRES （外部复位引脚）[a]、欠压	复位

a. XRES 将触发整个系统的重启事件。所有状态 （包括冻结 GPIO）均被丢失。在这种情况下，器件重启后，无法读取唤醒源。

11.7 低功耗模式进入和退出

Cortex-M0 （CM0）的 “Wait For Interrupt” （等待中断， WFI）指令能够触发切换到睡眠、深度睡眠和休眠模式的事件。Cortex-M0 可以将进入低功耗模式的事件延迟到退出最低优先级的 ISR 为止 （如果置位了 CM0 系统控制寄存器中的 SLEEPONEXIT 位）。

切换到睡眠、深度睡眠和休眠模式的事件是由 CM0 系统控制寄存器 （CM0_SCR）中的 SLEEPDEEP 标志和系统资源电源子系统 （PWR_CONTROL）中的 HIBERNATE 标志控制的。

- WFI 指令被执行，并且 SLEEPDEEP = 0 和 HIBERNATE = x 时，将进入睡眠模式。
- WFI 指令被执行， SLEEPDEEP = 1 和 HIBERNATE = 0 时，器件将进入深度睡眠模式。
- WFI 指令被执行，且 SLEEPDEEP = 1 和 HIBERNATE = 1 时，器件将进入休眠模式。

PWR_CONTROL 寄存器中的 LPM READY 位显示的是深度睡眠和休眠状态电压调节器的状态。如果在电压调节器准备好前，固件尝试进入深度睡眠或休眠模式，那么 PSoC 4 会先进入睡眠模式，然后等待电压调节器准备好后才会进入深度睡眠或休眠模式。通过硬件，可以自动完成该操作。

在睡眠和深度睡眠模式下，可以选择外设 （请参见表 11-3），并且通过固件可以使能或禁用与其相应的中断。已使能的中断可以使系统从低功耗模式唤醒，然后再进入活动模式。另外，所有 RESET 都会使系统返回到活动模式。更多有关信息，请参见第 57 页上的中断章节和第 111 页上的复位系统章节。

通过 PWR_STOP 寄存器，可以将 GPIO 状态冻结在低功耗模式中。建议将该方法应用于休眠和停止模式，因为从这些模式中唤醒会使系统复位。通过使用系统资源电源子系统中的 PWR_STOP 寄存器，可以直接进入停止模式。这时，系统中所有低电压逻辑电源都被断开。在该模式下，只会保持 I/O 状态和 PWR_STOP 寄存器中的内容，并且 XRES 引脚或固定 WAKEUP 引脚的切换会导致唤醒 （复位）。
PWR_STOP 寄存器中的字段包括：

- TOKEN — 该字段包含一个能够在 STOP/WAKEUP 序列中保持的 8 位令牌。该令牌可供固件使用，这样可以区分 WAKEUP 事件和通用的 RESET 事件。请注意，使用 XRES 将器件从停止模式唤醒会复位该寄存器。
- UNLOCK — 必须向该字段写入 ‘0x3A’，以解锁停止模式。如果对该字段写入其他值，硬件将忽略 STOP 位。
- POLARITY — 该位用于设置 WAKEUP 引脚输入的极性。 WAKEUP 引脚输入同 POLARITY 位值匹配时，器件将被唤醒。
- FREEZE — 置位该位将冻结系统中所有 GPIO 的配置、模式和状态。
- STOP — 必须置位该位才能进入停止模式。

进入停止模式的推荐程序如下：

1. 写 TOKEN = < 任何特定于应用的值 >
2. 写 UNLOCK = 0x3A
3. 写 POLARITY = < 特定于应用的极性 >
4. 写 FREEZE = 1
5. 写 STOP = 1

建议在第三个写周期后另外加入两个 NOP 周期。切换 XRES 或 WAKEUP 引脚时，将退出停止模式。这两个事件将清除 PWR_STOP 寄存器中的 STOP 位，并触发 POR。唤醒事件不会清除 PWR_STOP 寄存器中的其他位，但 XRES 事件将清除所有位。

从停止或休眠模式中唤醒的推荐固件程序如下：

1. 为特定应用的分支可选读取 TOKEN。
2. 可以按照所需设置对各 I/O 驱动模式和输出数据寄存器进行写操作。数字输出端口的典型程序为：将引脚设置为输出，读取它的冻结值，然后将该值设置给输出数据寄存器。
3. 解冻 I/O。

11.8 寄存器列表

表 11-4. 功耗模式寄存器列表

寄存器名称	说明
CM0_SCR	系统控制 — 设置或返回系统控制数据。
PWR_CONTROL	功耗模式控制 — 控制器件的功耗模式选项，且允许观察当前状态。
PWR_STOP	停止功耗模式 — 控制进入 / 退出停止模式。

12. 看门狗定时器

当固件执行发生异常时，可以使用看门狗定时器（WDT）进行自动复位器件。WDT 由 ILO 或 WCO 生成的 LFCLK 提供时钟。如果定时器被使能，则固件必须定期刷新它，以避免发生复位。否则，定时器将停止并生成器件复位。在低功耗模式下，WDT 还可作为中断源或唤醒源使用。

12.1 特性

WDT 具有以下特性：

- 可配置时间间隔完成后会生成一个系统复位
- 在活动、睡眠和深度睡眠模式下，能够定期中断 （或唤醒）器件
- 提供两个 16 位和一个 32 位的独立计数器，可以将它们级联起来延长间隔

12.2 框图

图 12-1. 看门狗定时器框图

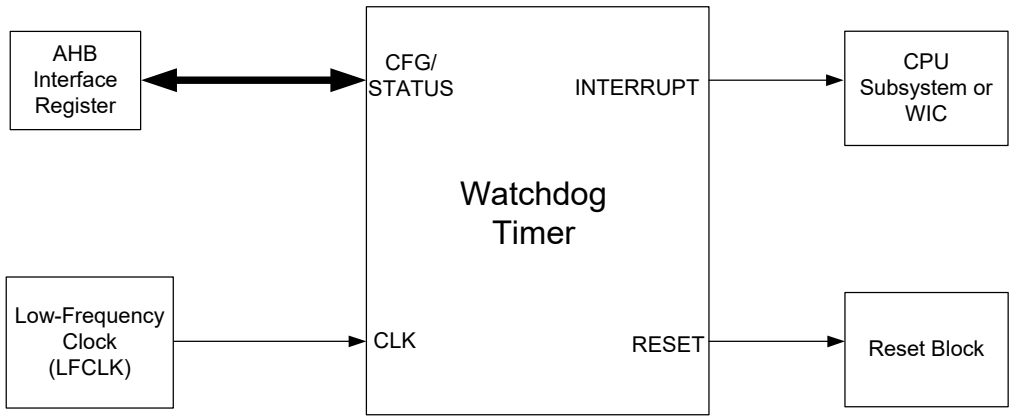
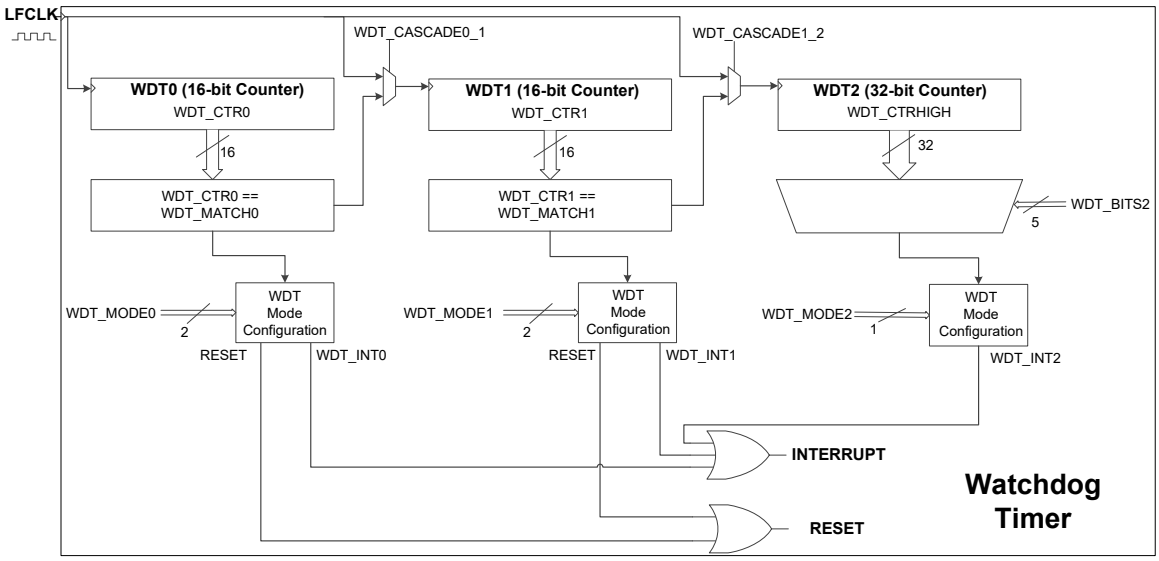


图 12-2. 看门狗定时器内部框图



12.3 工作原理

除非固件对WDT进行定期处理，否则经过可编程间隔后，它会激活器件的中断或硬件复位。WDT拥有两个16位计数器（WDT0 和 WDT1）和一个 32 位计数器 （WDT2）。可以将这些计数器配置为独立或级联计数器。

可以配置 WDT0 和 WDT1，从而能在发生匹配事件时 （即计数器的值等于匹配值）生成中断。也可以对 WDT0 和 WDT1 计数器进行配置，以便在发生匹配事件时或在 3 个连续事件未得到处理 （不清除匹配事件中断）后生成复位。

根据 WDT_CONFIG 寄存器中 WDT_BITS2[4:0] 位的值，可以配置 WDT2 生成中断。当任意一个匹配值等于 WDT1 或 WDT0 值时， WDT2 不能生成系统复位或中断。WDT2 只能在计数器内 32 位中任意一位的上升沿上生成中断。 WDT_CONFIG 寄存器中的 WDT_BITS2[4:0] 位控制着用于生成中断的位。有关详细信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中介绍的 WDT_CONFIG 寄存器的内容。

WDT_MODEx 位可用于配置看门狗计数器，如上面所述。

通过图 12-2 中所示的级联配置，可以延长复位或中断间隔。请留意，级联两个 16 位计数器并不能提供一个 32 位计数器，但能够得到一个 16 位定期计数器和一个 16 位预分频器。例如，级联 WDT0 和 WDT1 时， WDT0 将作为 WDT1 的预分频器，并且预分频值将由 WDT_MATCH 寄存器中的 WDT_MATCH0[15:0] 位定义。 WDT1 的周期大小将由 WDT_MATCH 寄存器中的 WDT_MATCH1[31:16] 位定义。相似的逻辑也适用于 WDT1 和 WDT2 的级联。

当使用 WDT 来防止系统崩溃时，必须使用与 WDT 中断没有直接关联的代码来清除 WDT 中断位，从而实现复位 WDT。否则，就算固件的主要函数崩溃或处于无限循环状态， WDT 中断向量仍完整并定期刷新 WDT。

使用 WDT 来防止系统崩溃的最安全方法是：

- 配置看门狗复位周期，以便在该周期甚至固件的最大延迟路径中，固件能够至少复位一次看门狗。
- 通过在固件代码的主体中定期清除中断位，从而复位看门狗。如果将 WDT 配置为在匹配事件发生时生成复位，那么通过清除 WDTx 计数器来复位看门狗。通过设置 WDT_CONFIG 寄存器中的 WDT_RESETx 位，可以清除 WDTx 计数器。有关详细信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中介绍的 WDT_CONFIG 寄存器的内容。
- 如果将 WDT 作为复位源来防止系统崩溃，那么不应该在 WDT 中断服务子程序（ISR）中复位看门狗。因此，请不要使用相同的看门狗计数器来生成系统复位和中断。例如，如果使用 WDT0 来生成系统复位，从而防止系统崩溃，那么需要使用 WDT1 或 WDT2 来定期生成中断。

请按照以下各步骤将 WDT 作为一个周期性中断发生器使用：

1. 为 WDT0（或 WDT1）设置 WDT_CONFIG 寄存器中的 WDT_CLEAR0（或 WDT_CLEAR1）位，以便在发生匹配事件时对相应的看门狗计数器进行复位。
2. 将所需的匹配值写入到 WDT0/WDT1 的 WDT_MATCH 寄存器内，并将 WDT_BITS2 的值写入到 WDT2 的 WDT_CONFIG 寄存器内。
3. 通过清除 WDT_CONTROL 寄存器中的 WDT_INTx 位，可以清除所有待处理中断。
4. 通过配置 WDT_CONFIG 中的 WDT_MODEx 位来使能 WDT 中断。对于 WDT0 或 WDT1，将 WDT_CONFIG 中的 WDT_MODE0 或 WDT_MODE1 位配置为 ‘1’ （匹配时生成中断）或 ‘3’ （匹配时生成中断，并在尚未处理第三个匹配时生成复位）。对于 WDT2，要设置 WDT_CONFIG 寄存器中的 WDT_MODE2 位。
5. 使能 CM0_ISER 寄存器中的全局 WDT 中断 （有关详细信息，请参考第 57 页上的中断章节）。
6. 在 ISR 中，清除 WDT 中断。

更多有关中断的信息，请参见第 57 页上的中断章节。

对 WDT_MATCH 的更改需要经过三个 LFCLK 周期后才会生效。更改 WDT_MATCH 后，需要经过一个 LFCLK 周期后才能进入深度睡眠模式，以确保 WDT 更新了设置内容。

12.3.1 使能和禁用 WDT

通过设置 WDT_CONTROL 寄存器中的 WDT_ENABLEx 位，可以使能 WDT 计数器；如果想禁用它，则需要清除该位。使能或禁用 WDT 需要经过三个 LFCLK 周期后才能生效。因此，在这段时间内，不能多次更改 WDT_ENABLEx 位。

使能 WDT 后，建议不要对 WDT 配置寄存器 （WDT_CONFIG）和控制 （WDT_CONTROL）寄存器进行写操作。通过设置 CLK_SELECT 寄存器中的 WDT_LOCK[15:14] 位，可以防止 WDT 寄存器意外损坏。如果在运行 WDT 时，应用需要更新匹配值（WDT_MATCH），则必须清除 WDT_LOCK 位。需要进行两次写操作才能清除这两个 WDT_LOCK 位。写入 ‘1’ 会清除位 0，写入 ‘2’ 会清除位 1，写入 ‘3’ 会设置这两位，而写入 ‘0’ 则会使它们无效。有关详细信息，请参阅 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中介绍的 CLK_SELECT 寄存器的内容。

12.3.2 WDT 操作模式

通过使用 WDT0 和 WDT1 来生成复位，可以阻止系统进入无回应状态，或者通过使用这些定时器生成中断，这样可使系统从睡眠或深度睡眠模式唤醒。可以对 WDT_CONFIG 寄存器中的 WDT_MODEx[1:0] 位字段进行配置，以便在 WDT_CTRx 寄存器中存储的计数值等于 WDT_MATCH 寄存器中存储的匹配值时，能够选择所需要的操作。有关详细信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中介绍的 WDT_CTRHIGH、WDT_CTRLOW 和 WDT_MATCH 寄存器的内容。

表 12-1. WDT0 和 WDT1 模式

位字段名称	说明
WDT_MODE0[1:0] 或 WDT_MODE1[1:0]	发生匹配情况，即 （WDT_CTR0=WDT_MATCH0）或 （WDT_CTR1=WDT_MATCH1）时，看门狗计数器的操作如下： 00：无操作 01：激活 WDT_INT0 或 WDT_INT1 10：激活 WDT 复位 11：激活 WDT_INT0 或 WDT_INT1，并在尚未处理第三个中断后激活 WDT 复位

根据 WDT_BITS2[4:0] 寄存器位的状态，可以使用 WDT2 生成中断。**注意：**如果对看门狗进行配置，以便在每个 LFCLK 周期

表 12-2. WDT2 模式

位字段名称	说明
WDT_MODE2	0：无中断请求的自由运行计数器 1：自动运行计数器，将在由 WDT_CONFIG 寄存器中 WDT_BITS2 位所指定的位上升沿上生成中断请求

内都会生成中断，那么请确保清除看门狗中断 （设置 WDT_CONTROL 寄存器内的 WDT_INTx 位）后读取 WDT_CONTROL 寄存器的值。否则，会丢失下一个中断，也会在 LFCLK/2 时间内生成中断。

12.3.3 WDT 中断和低功耗模式

在活动模式下，看门狗计数器可将中断请求发送给 CPU ；在睡眠和深度睡眠模式下，则可将该中断请求发送给唤醒中断控制器 （WIC）。其工作原理如下：

- **活动模式：**在活动模式下，WDT 可以向 CPU 发送中断请求。CPU 确认该中断请求，并执行 ISR。进入 ISR 后，必须通过固件清除该中断。
- **睡眠或深度睡眠模式：**在该模式中，CPU 子系统被断电。因此，WDT 的中断请求将被直接发送到 WIC，以唤醒 CPU。CPU 确认中断请求，并执行 ISR。进入 ISR 后，必须通过固件清除该中断。

要想了解有关功耗模式的详细信息，请参阅第 101 页上的功耗模式章节。

12.3.4 WDT 复位模式

RES_CAUSE 寄存器中的 RESET_WDT 位表示由 WDT 生成的复位。该位将保持置位状态，直到清除该位或发生上电复位（POR）、欠压复位 （BOD）或外部复位 （XRES）为止。所有其它复位操作都不会对该位产生任何影响。更多信息，请参阅第 111 页上的复位系统章节。

12.4 寄存器列表

表 12-3. WDT 寄存器

寄存器名称	说明
WDT_CTRLOW	看门狗计数器 0 和 1
WDT_CTRHIGH	看门狗计数器 2
WDT_MATCH	看门狗计数器 0 和 1 的匹配值
WDT_CONFIG	包含 WDT 的配置位
WDT_CONTROL	控制 WDT 计数器的操作

13. 复位系统

PSoC® 4 支持多种复位类型，从而可保证器件上电时无错误运行，并且允许器件根据用户提供的外部硬件或内部软件复位信号进行复位。 PSoC 4 还包括用于使能复位检测的硬件。

复位系统包括以下复位源：

- 上电复位 （POR）— 供电电压上升时，器件将处于复位状态
- 欠压复位 （BOD）— 在操作过程中，如果供电电压低于标准范围，将产生复位
- 看门狗复位 （WRES）— 如果固件不能服务看门狗定时器，将产生器件复位
- 软件复位 （SRES）— 如果需要，可以使用固件进行复位器件
- 外部复位 （XRES） — 使用外部电信号来复位器件
- 保护故障复位 （PROT_FAULT）— 如果用户尝试进行未经授权操作时，将产生复位
- 休眠唤醒复位 — 使器件退出休眠低功耗模式
- 停止唤醒复位 — 使器件退出停止低功耗模式

13.1 复位源

下述内容描述了 PSoC 4 的各种复位源。

13.1.1 上电复位

上电复位用于在上电时系统复位。POR 会将器件锁定在复位状态，直到供电电压 V_{DDD} 满足数据手册中的规范为止。上电时，POR 自动被激活。

POR 事件不会设置复位源的状态位；但是，通过观察其它复位源的状态，可以在一定程度上推断出原因。如果未检测到其它复位事件，那么复位是由 POR、 BOD 或 XRES 导致的。

13.1.2 欠压复位

欠压复位将监控芯片数字电源电压 V_{CCD} ；如果 V_{CCD} 低于器件数据手册中指定的最小逻辑工作电压，将生成复位。BOD 在所有功耗模式下均可用 （停止模式除外）。

BOD 事件将不设置复位源状态位，但是在某些情况下，可以检测到它。在某些 BOD 事件中， V_{CCD} 会低于最小的逻辑工作电压，但仍大于最小逻辑保持电压。因此，可以通过检查逻辑保持来区分这些 BOD 事件和 POR 事件。详细信息，请查阅第 112 页上的 “ 识别复位源 ” 一节。

13.1.3 看门狗复位

如果看门狗定时器未在用户指定的时限内复位计数器，看门狗 （WRES）将产生复位，这表明用户程序发生了错误。通过置位 WDT_CONTROL 寄存器内的 WDT_ENABLEx 位，可以使能该功能。

发生看门狗复位时，RES_CAUSE 寄存器内的 RESET_WDT 状态位将被置位。该位保持为置位状态，直到被清除或发生 POR、XRES 或不可检测的 BOD 复位为止，例如：在一个器件的供电周期中。所有其它复位操作都不会对该位产生任何影响。

更多信息，请参考第 107 页上的看门狗定时器章节的内容。

13.1.4 软件复位

软件复位（SRES）是一个允许软件驱动复位的机制。将 ‘1’ 写入 SYSRESETREQ 位，这时 Cortex-M0 应用中断与复位控制寄存器（CM0_AIRCR）将强制器件进行复位。为了使能写操作，需要将数值 A05F 写入到 CM0_AIRCR 寄存器的前两个字节内。因此，复位操作需要写入 A05F0004。

发生软件复位时，RES_CAUSE 寄存器内的 RESET_SOFT 状态位将被置位。该位保持为置位状态，直到被清除或发生 POR、XRES 或不可检测的 BOD 复位为止，例如：在一个器件的供电周期中。所有其它复位操作都不会对该位产生任何影响。

13.1.5 外部复位

外部复位（XRES）是一种由用户提供的复位，产生后立即导致系统复位。XRES 引脚为***低电平有效信号***— 该引脚上的高电平电压不起任何作用，而低电平电压则会导致复位。在器件中，引脚处于高电平状态。在大多数器件中， XRES 均作为专用引脚使用。更多有关引脚分布的详细信息，请参见 器件数据手册中的 “ 引脚分布 ” 一节。

XRES 引脚处于活动状态时，将保持器件的复位状态。释放引脚后，器件将进行正常的启动序列。器件数据手册中的 “ 电气规范 ” 一节列出了各 XRES 的逻辑阈值和其它电气特性。

XRES 事件将不设置复位源的状态位。但是，通过观察其它复位源的状态，可以在一定程度上推断出原因。如果未检测到其它复位事件，那么，复位就是由 POR、不可检测的 BOD 或 XRES 导致的。

13.1.6 保护故障复位

保护故障复位 （PROT_FAULT）将检测未经授权的特权操作。如果检测到该类行为，它将产生器件复位。例如，执行特权代码时，遇到了调试断点。更多有关特权代码的信息，请参见第 99 页上的 “ 特权模式 ” 一节的内容。

发生保护故障时，RES_CAUSE 寄存器内的 RESET_PROT_FAULT 位被置位。该位保持为置位状态，直到被清除或发生 POR、XRES 或不可检测的 BOD 复位为止，例如：在一个器件的供电周期中。所有其它复位操作都不会对该位产生任何影响。

13.1.7 休眠模式唤醒复位

休眠模式唤醒复位将检测休眠唤醒源，并执行器件复位，以返回活动模式。休眠模式唤醒复位是由中断导致的。在休眠低功耗模式下，引脚和比较器中断均可用。执行休眠唤醒复位后， SRAM 和 UDB 寄存器的内容均被保持，但是此复位后的代码执行与其它复位源发生后的执行相同。

通过检查比较器和引脚的中断寄存器，可以检测休眠复位。休眠状态唤醒复位后，仍保持这些中断寄存器的状态。

更多信息，请查阅第 103 页上的 “ 休眠模式 ” 一节。

13.1.8 停止模式唤醒复位

停止模式唤醒复位将检测各个停止模式唤醒源，并执行一次器件复位，以返回到活动模式。停止模式唤醒复位是由 XRES 引脚或 WAKEUP 引脚导致的。执行停止模式唤醒复位后，将不会保持存储器中的任何内容；复位后的代码执行与其它复位源发生后的相同。

通过检查 PWR_STOP 寄存器中的 TOKEN 位字段 （位 0:7），可以检测某些停止模式的唤醒源。进入停止模式时，会向该位字段内填充一个关键字。如果使用 WAKEUP 引脚唤醒器件，该字段的内容将被保持。如果使用 XRES 引脚唤醒器件，则无法检测唤醒源。更多信息，请查阅第 103 页上的 “ 停止模式 ” 一节的内容。

13.2 识别复位源

器件退出复位状态时，知道最近或更早的复位源是很有用的。通常，通过器件的 RES_CAUSE 寄存器，可以获得该信息。该寄存器具有与几种复位源相对应的特定状态位。 RES_CAUSE 寄存器支持看门狗复位、软件复位以及保护故障复位等复位源检测。它并没记录 POR、BOD、XRES 或休眠和停止模式唤醒复位的发生。发生复位时，相应的状态位将被置位，并且在复位后仍保持置位状态，直到它被清除或保持丢失为止，如 POR 复位、外部复位或欠压检测。

通过检查配置为将器件从休眠模式中唤醒的比较器和引脚中断寄存器，可以检测休眠唤醒复位。通过检查PWR_STOP寄存器，可以检测由 WAKEUP 引脚事件导致的停止唤醒复位，如上面所述。无法检测到由 XRES 导致的停止唤醒复位。通过表 13-1 所示的 RES_CAUSE 状态，可以在一定程度上推断其他复位源。

表 13-1. 用于检测复位源的复位原因位

位	名称	描述
0	RESET_WDT	芯片发生了一次看门狗定时器复位。
3	RESET_PROT_FAULT	发生保护违法行为时就需要一次 RESET。
4	RESET_SOFT	Cortex-M0 通过它的 SYSRESETREQ 请求一次系统复位。

欠压事件可分为两个子类型：保持复位和无保持复位。如果 V_{CCD} 低于最小逻辑工作电压，但不小于最小逻辑保持电压，这时将发生 BOD 复位；寄存器中的内容会被保持。如果 V_{CCD} 低于最小工作电压和最小保持电压，将发生 BOD 复位，但不会保持寄存器中的内容。可以使用特别的寄存器 （PWR_BOD_KEY）来检测寄存器的保持内容。仅在通过固件进行写操作或发生无保持复位（如无保持 BOD、XRES 或 POR 事件）时， PWR_BOD_KEY 寄存器的值才被修改。通过固件初始化该寄存器，然后在启动代码的后续执行中进行检查该寄存器，从而确定是否发生了保持 BOD 复位。

如果这些方法都不能检测到复位源，那么复位源可能是下面无记录或无保持复位中的一种：无保持 BOD、 POR、 XRES 或停止唤醒复位。通过片上资源无法区分这些复位。

13.3 寄存器列表

表 13-2. 复位系统寄存器列表

寄存器名称	描述
WDT_CONTROL	看门狗定时器控制寄存器 — 通过该寄存器，可以配置器件看门狗定时器。
CM0_AIRCR	Cortex-M0 应用中断和复位控制寄存器 — 除了初始化软件复位功能外，寄存器还提供了 Cortex-M0 的其他功能 。
RES_CAUSE	复位源寄存器 — 该寄存器会捕捉最近发生复位的原因。
PWR_STOP	该寄存器用于控制停止功耗模式的进入和退出情况。

14. 器件安全性

PSoC® 4 为用户提供了多种方式来防止未授权的访问和复制。禁用调试性能以及使能闪存保护提供了高级别的安全性。在 PSoC 4200BLE 器件中，通过执行通用数字模块（即 UDB）（不是在固件中的自定义功能），可以提高安全性。与进行目标代码的逆向工程相比，对在 UDB 中实现的硬件设计进行逆向工程更加困难。

默认情况下，调试电路处于使能状态，并且只能通过固件禁用它们。如果调试电路被禁用，重新使能该电路的唯一方法是擦除整个器件，清除闪存保护，然后使用允许调试的新固件重新对该器件进行编程。此外，如果担心因器件恶意重新编程而造成欺诈性攻击的应用或通过启动和中断闪存编程序列来击败安全性的尝试，则可以永久性禁用所有器件接口。对于大多数应用，不建议永久性禁用接口，因为这样会使设计人员无法对器件进行访问。更多有关信息以及关于闪存行和芯片保护的讨论，请参阅 PSoC 4-BLE 的编程规范中介绍的内容。

注意：由于器件的最高安全级别被使能时将禁用所有编程、调试和测试接口，因此采用该安全级别的 PSoC 4 器件将不能再恢复到正常状态以进行故障分析。

14.1 特性

PSoC 4 器件的安全系统具有以下特性：

- 支持用户可选的保护级别。
- 在最高安全级别模式下，芯片被 “ 锁定 ”。这时无法对芯片进行测试或调试，也无法进行擦除操作。中断擦除操作是黑客使芯片处于未定义状态并打开观察的一种方式。
- 使用不可屏蔽中断（NMI）将使 CPU 在特权模式下运行。在特权模式下，NMI 保持确认状态，以防止中断指令的意外返回而造成安全漏洞。

另外， PSoC 4 还为单独闪存行数据提供保护。

14.2 工作原理

14.2.1 器件安全性

CPU 在通用用户模式或特权模式下运行，而器件则在 BOOT、OPEN、PROTECTED 以及 KILL 等四种保护模式下运行。每种模式均为 CPU 软件和调试提供了特定的功能。通过对 CPUSS_PROTECTION 寄存器进行写操作，可以更改器件运行模式。

- **BOOT 模式：**在该模式下，器件将退出复位状态。器件一直处于该模式，直到将其保护状态从监控闪存复制到保护控制寄存器 （CPUSS_PROTECTION）内为止。该操作完成前，调试访问端口仍被禁止。 BOOT 是一种过渡模式。该模式要求器件处于已配置的保护状态。在 BOOT 模式下， CPU 始终在特权模式下运行。
- **OPEN 模式：**这是出厂默认模式。 CPU 可以在用户模式或特权模式下运行。在用户模式下，可以对闪存进行编程，并且支持调试器功能。在特权模式下，访问受限。
- **PROTECTED 模式：**用户可以将 OPEN 模式转换为 PROTECTED 模式。在该模式下，禁止对用户代码或存储器进行的所有调试访问。尽管仍可以对大部分寄存器进行访问；但不能对寄存器进行调试来重新编程闪存。只有完成对闪存的擦除后才能将该模式转换回到 OPEN 模式。
- **KILL 模式：**用户可以将 OPEN 模式转换为 KILL 模式。通过该模式可以清除对用户代码或存储器进行的所有调试访问，但不能对闪存进行擦除。尽管仍可以访问大部分寄存器，但不能调试寄存器，使之重新编程闪存。器件无法退出 KILL 模式。在该模式下，器件不能退回进行故障分析。

14.2.2 闪存安全性

PSoC 4 器件包含了一个灵活的闪存保护系统，该系统控制着对闪存存储器进行的访问。该性能可以保护专有代码，另外它还可以防止对闪存内的 Bootloader 部分进行的意外写操作。

闪存存储器按行组织。您可以将两个保护级别的其中一个分配给每一行；请参考表 14-1。要想更改闪存保护级别，必须擦除整个闪存。

更多信息，请参考第 335 页上的非易失性存储器编程章节。

表 14-1. 闪存保护级别

保护设置	支持	不支持
无保护	外部读写操作， 内部读写操作	–
全面保护	外部读操作[a] 内部读操作	外部写操作， 内部写操作

a. 为了防止对器件进行的外部读取操作，您需要将器件保护设置更改为 PROTECTED （受保护）。

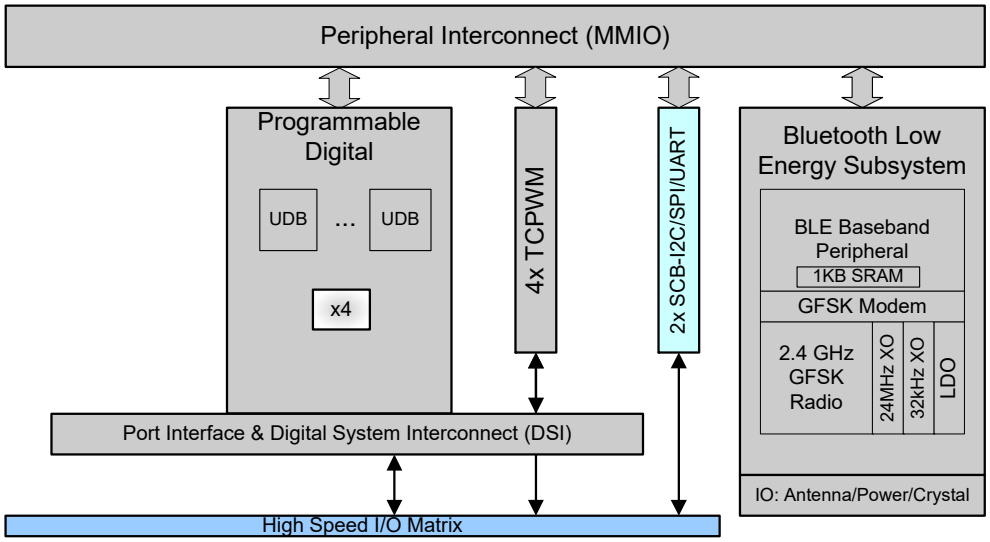
章节 D: 数字系统

本部分包括以下章节：

- 第 119 页上的串行通信模块 （SCB） 章节
- 第 161 页上的通用数字模块 （UDB） 章节
- 第 203 页上的定时器、计数器和 PWM 章节
- 第 227 页上的蓝牙低功耗子系统 （BLESS） 章节

系统架构

数字系统框图



15. 串行通信模块 （SCB）

PSoC® 4 的串行通信模块 （SCB）支持三种串行接口协议：SPI、 UART 和 I²C。在任何给定时间， SCB 只支持一种协议。PSoC 4 器件具有 两个 SCB。通过使用通用数字模块 （UDB），可实现串行外设接口 （SPI）和 UART 协议之间的其他实例。

15.1 特性

该模块支持以下特性：

- 标准 SPI 主设备和从设备功能，采用 Motorola、 Texas Instruments 和 National Semiconductor 的协议
- 标准 UART 功能，支持 SmartCard 读卡器、局部互联网络 （LIN）和 IrDA 协议
- 支持标准的 I²C 主 / 从设备功能
- 标准 LIN 从设备功能符合 LIN 版本 1.3 和 LIN 版本 2.1/2.2 规范的标准
- SPI 和 I²C 的 EZ 模式，允许器件进行正常操作而无需 CPU 的干预
- SPI 与 I²C 协议的低功耗 （深度睡眠模式）工作模式 （使用外部时钟）

下面各节将介绍每个协议的内容。

15.2 串行外设接口 （SPI）

串行外设接口 （SPI）是同步的串行接口协议。器件在主设备或从设备模式下运行。主设备启动数据传输。 SCB 为 SPI 支持 “ 一主多从 ” 的设备拓扑结构。多个从设备均有单独的从设备选择线。

当 PSoC 需要与一个或多个 SPI 从设备进行通信时，您可使用 SPI 主设备模式。当 PSoC 需要与 SPI 主设备进行通信时，您可以使用 SPI 从设备模式。

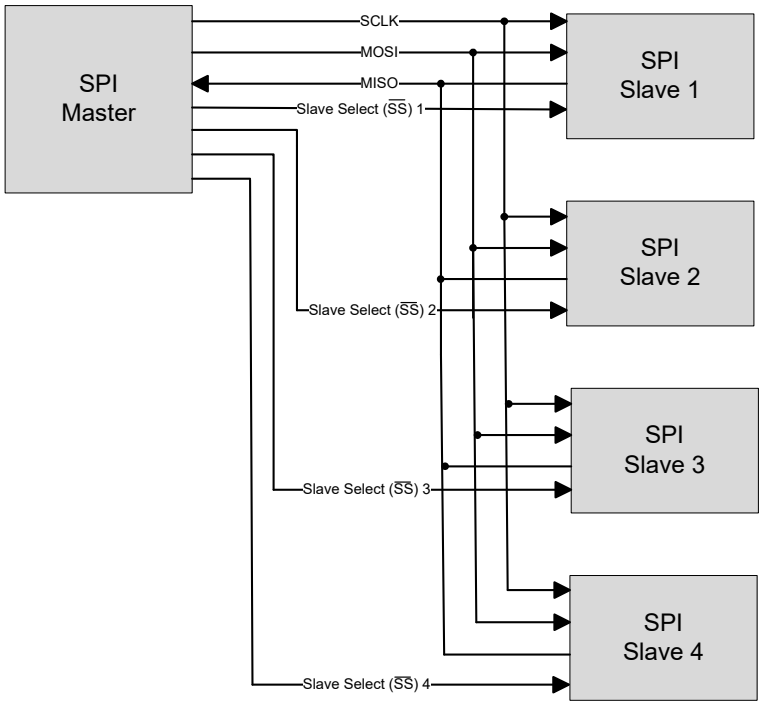
15.2.1 特性

- 支持主设备和从设备功能
- 支持以下三种 SPI 协议：
  - Motorola SPI — 模式 0、 1、 2 和 3
  - Texas Instruments SPI — 具有用于模式 1 的同步和前数据帧指示器
  - 针对模式 0 的 National Semiconductor （MicroWire） SPI
- 支持多达 4 个从设备选择线
- 可编程数据帧大小，取值范围为 4 到 16 位
- 中断或轮循 CPU 接口
- 可编程过采样
- 支持 EZ 操作模式 （Easy SPI 协议）
  - EZSPI 模式允许器件正常操作而无需 CPU 的干预
- 支持外部时钟的从设备操作：
  - 在该模式下，从设备会在活动、睡眠和深度睡眠等各种系统功耗模式中运行。

15.2.2 概述

图 15-1 提供了 SPI 主设备和四个从设备的示例。

图 15-1. SPI 示例



标准的 SPI 接口包含以下四个信号线。

- SCLK：串行时钟 （时钟信号从主设备输出并输入到从设备内）。
- MOSI：主出从入 （数据从主设备输出，向从设备输入）。
- MISO：主入从出 （数据向主设备输入，从从设备输出）。
- 从设备选择 （$\overline{SS}$）：一般为低电平有效信号 （从主设备输出，输入到从设备）。

一个简单的 SPI 数据传输操作包括以下步骤：主设备通过驱动其 $\overline{SS}$ 线路来选择从设备，然后驱动 MOSI 线路上的数据以及 SCLK 线路上的时钟。从设备使用 SCLK 的一个边沿 （根据配置）捕捉 MOSI 线路上的数据；它还可驱动 MISO 线路上的数据，该数据由主设备捕获。

默认情况下， SPI 接口支持 8 位 （一个字节）的数据帧大小。可将数据帧大小配置为 4 到 16 位范围内的任何值。串行数据可以按最高有效位 （MSB）优先或最低有效位 （LSB）优先的方式进行传输。

以下三种 SPI 协议的不同形式都受 SCB 的支持：

- Motorola SPI：这是原始的 SPI 协议。
- Texas Instruments SPI：这是原始 SPI 协议的一种变体，其中数据帧由 $\overline{SS}$ 线上的脉冲确定。
- National Semiconductors SPI：这是原始 SPI 协议的一种半双工变体。

15.2.3 SPI 操作模式

15.2.3.1 Motorola SPI

原始的 SPI 协议由 Motorola 定义。它是一种全双工协议。当 $\overline{SS}$ 线路保持为 ‘0’ 时，会发生多次数据传输。因此，从设备必须跟踪数据传输过程，以区分各单独数据帧。如果未传输数据，则 $\overline{SS}$ 线路为 ‘1’，并且 SCLK 通常处于低电平状态。

Motorola SPI 模式

根据 MOSI 和 MISO 线路上输出和采样数据的不同方式， Motorola SPI 协议具有四种模式。这些模式由时钟极性（CPOL）和时钟相位 （CPHA）决定。

时钟极性决定了 SCLK 线路在未传输数据时的值：CPOL = ‘0’ 表示当未传输数据时 SCLK 为 ‘0’。CPOL = ‘1’ 表示当未传输数据时 SCLK 为 ‘1’。

时钟相位决定何时输出和捕获数据。无论时钟边沿是上升沿还是下降沿，CPHA = 0 表示在前 （第一个）时钟边沿上采样 （捕捉）数据， CPHA = 1 表示在后 （第二个）时钟沿上采样数据。 CPHA = 0 时，在第一个时钟周期到来前，数据在建立时间内必须保持稳定状态。

- 模式 0：CPOL = 0， CPHA = 0：数据在 SCKL 的下降沿上被输出，并在 SCLK 的上升沿上被采样。
- 模式 1：CPOL = 0， CPHA = 1：数据在 SCKL 的上升沿上被输出，并在 SCLK 的下降沿上被采样。
- 模式 2：CPOL = 1， CPHA = 0：数据在 SCKL 的上升沿上被输出，并在 SCLK 的下降沿上被采样。
- 模式 3：CPOL = 1， CPHA = 1：数据在 SCKL 的下降沿上被输出，并在 SCLK 的上升沿上被采样。

图 15-2 描述了基于 CPOL 和 CPHA 的 MOSI/MISO 数据输出和采样情况。

图 15-2. SPI Motorola 的 4 种模式

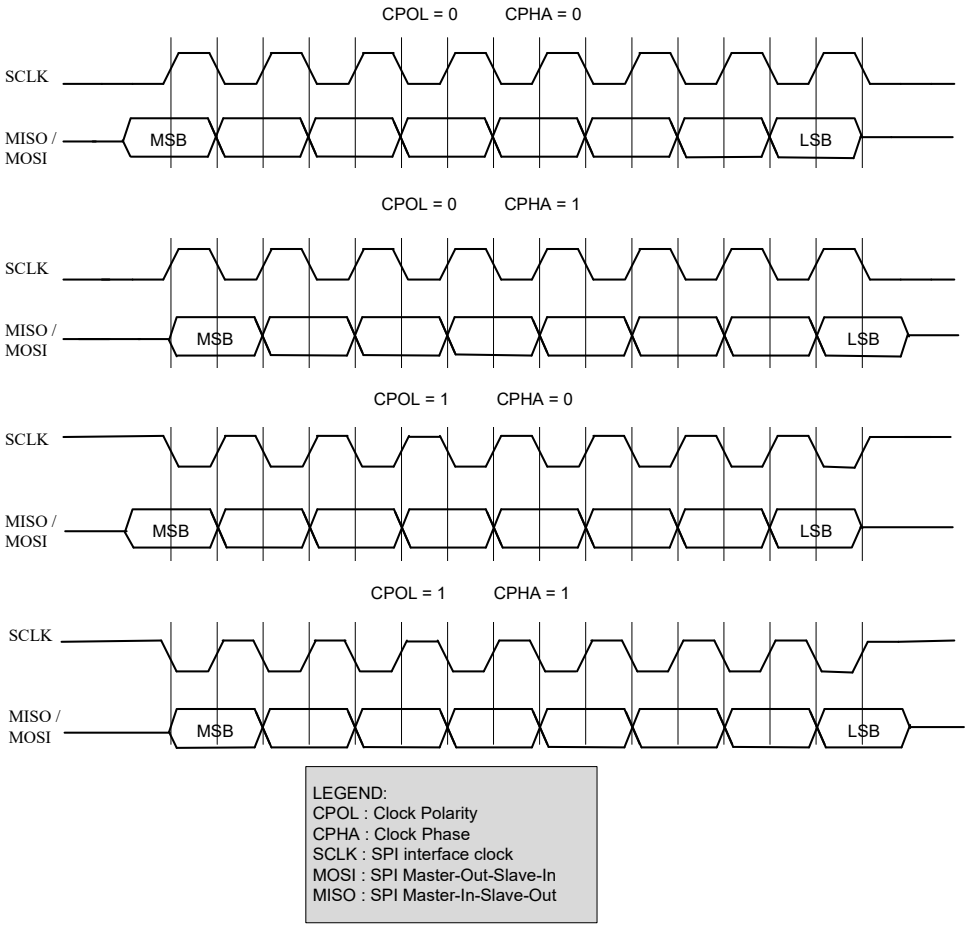
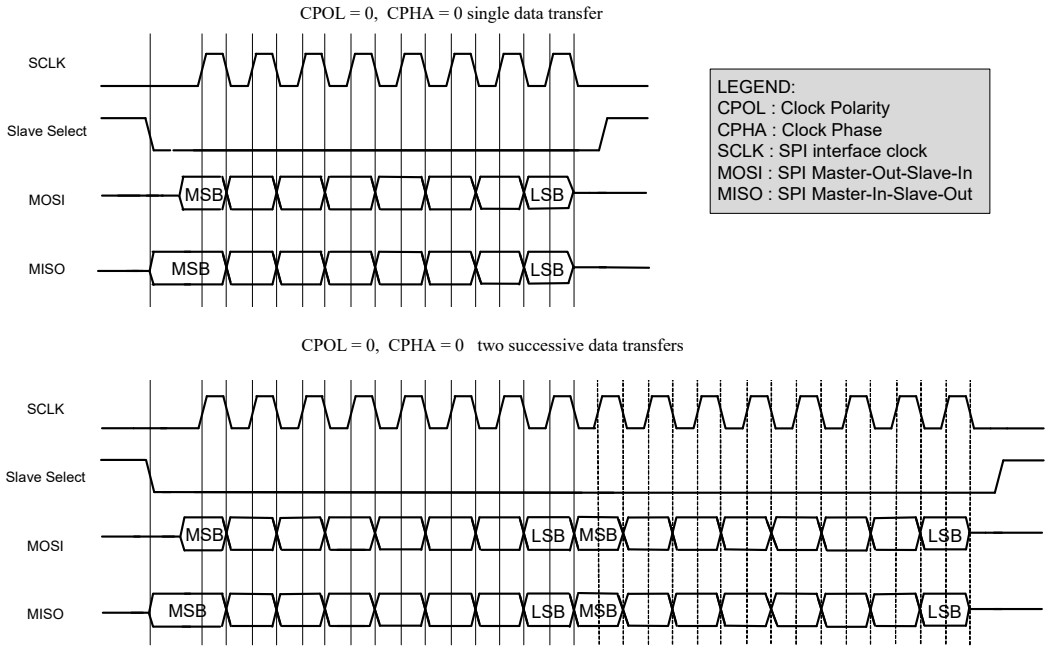


图 15-3 描述了在模式 0 （CPOL = 0， CPHA = 0）下，一个 8 位数据和两个连续 8 位数据的传输。

图 15-3. SPI Motorola 数据传输示例图



为 SPI Motorola 模式配置 SCB

要想将 SCB 配置为 SPI Motorola 模式，请按下列步骤设置各种寄存器位：

1. 将 ‘01’ 写入到 SCB_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI。
2. 将 ‘00’ 写入到 SCB_SPI_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI Motorola 模式。
3. 对 SCB_SPI_CTRL 寄存器中的 CPHA 和 CPOL 字段 （分别为位 2 和 3）进行写操作，从而选择 Motorola 操作模式。
4. 执行第 128 页上的 “ 使能和初始化 SPI” 一节中所描述的第 2 到第 4 步。

请注意， PSoC Creator 在 GUI 的帮助下自动执行这些所有操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册（TRM）*PSoC 4100-BL/4200-BL 系列: PSoC 4 BLE 寄存器数据参考手册（TRM）*。

15.2.3.2 Texas Instruments SPI

Texas Instruments 的 SPI 协议重新定义了 $\overline{SS}$ 信号的使用情况。与 Motorola SPI 的情况不同，它使用该信号表示开始传输数据，而不是通过低电平有效的从设备来选择信号。因此，从设备无需跟踪数据传输过程仍可以区分各单独数据帧。单比特传输周期中的高电平有效脉冲用于表示传输的开始。该脉冲可以与第一数据位传输同时或提前一个周期发生。TI SPI 协议仅支持模式 1 （CPOL = 0， CPHA = 1）：数据在 SCLK 的上升沿上被输出，并在 SCLK 的下降沿上被采样。

图 15-4 描述了一个 8 位数据传输和两个连续的 8 位数据传输。 SELECT 脉冲在传输第一个数据位前发生。请注意第二个数据传输中的 SELECT 脉冲与第一个数据传输中最后数据位同步传输。

图 15-4. SPI TI 数据传输示例图

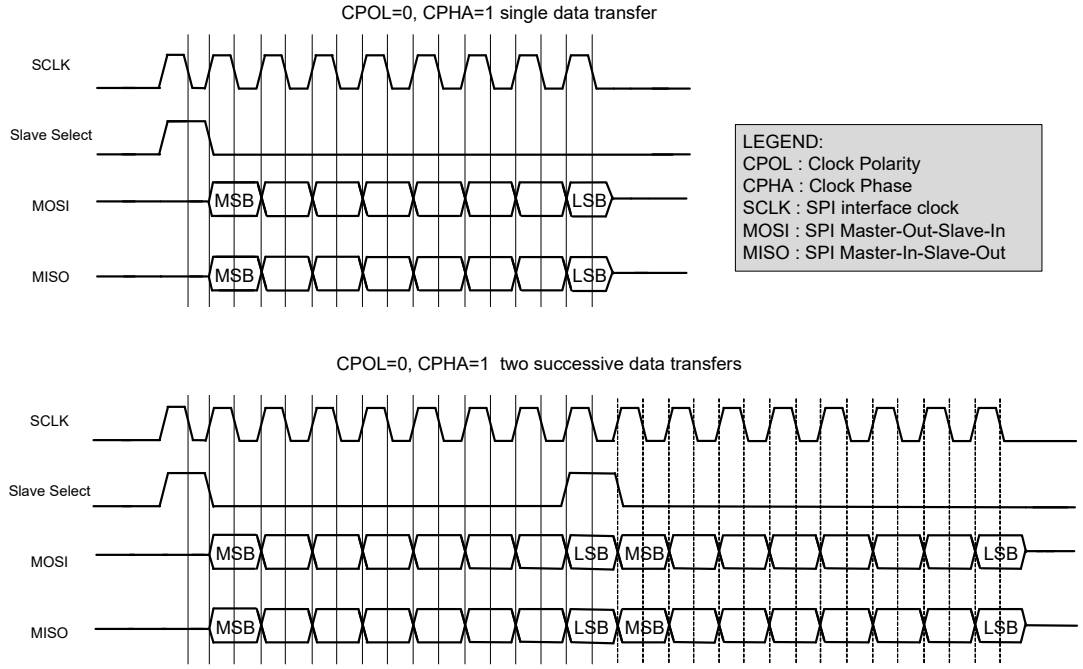
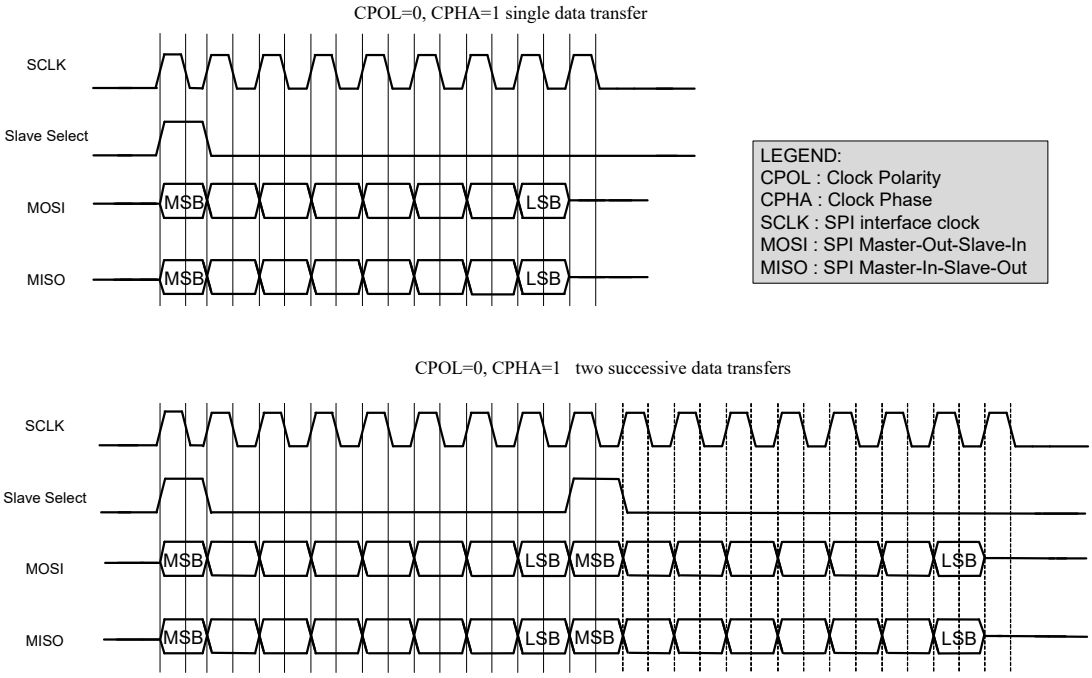


图 15-5 描述了一个 8 位数据传输和两个连续的 8 位数据传输。 SELECT 脉冲与数据帧的第一个数据位同步传输。

图 15-5. SPI TI 数据传输示例图



将 SCB 配置为 SPI TI 模式

要想将 SCB 配置为 SPI TI 模式，请按下列步骤置位各种寄存器位：

1. 将 ‘01’ 写入到 SCB_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI。
2. 将 ‘01’ 写入到 SCB_SPI_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI TI 模式。
3. 对 SCB_SPI_CTRL 寄存器中的 SELECT_PRECEDE 字段 （位 1）进行写操作 （1：表示 SELECT 脉冲优先于下个数据帧的第一位传输； 0：相反的），从而选择 TI 操作模式。
4. 执行第 128 页上的 “ 使能和初始化 SPI” 一节中所描述的第 2 到第 5 步。

请注意， PSoC Creator 在 GUI 的帮助下自动执行这些所有操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册（TRM）*PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册（TRM）*。

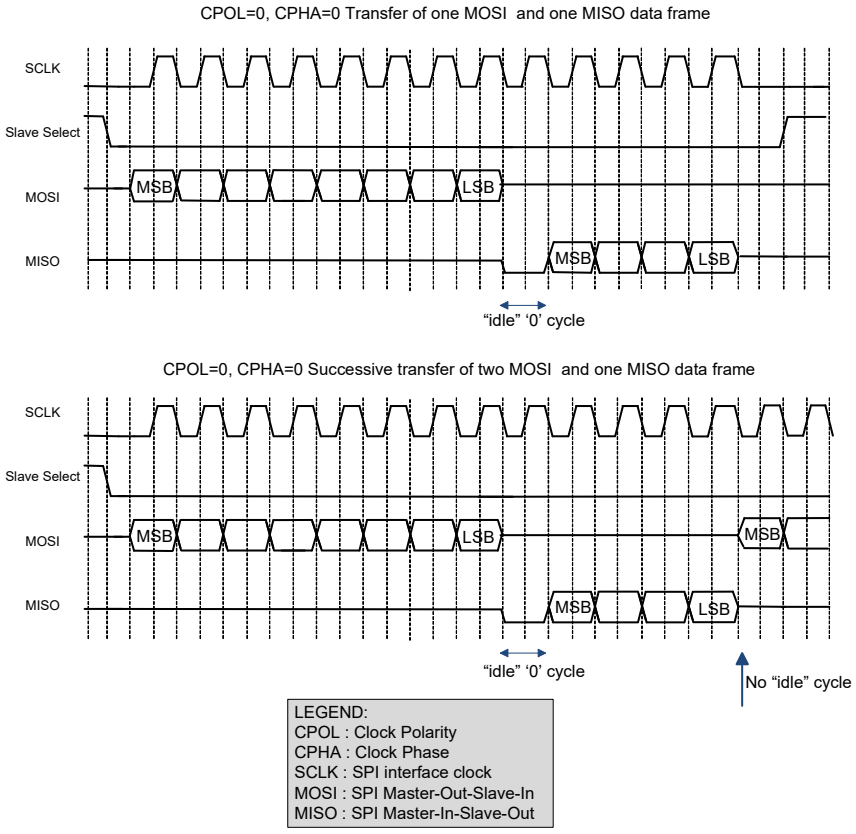
15.2.3.3 National Semiconductors SPI

National Semiconductors 的 SPI 协议是一种半双工协议。发送和接收交替进行，并非同时进行。发送的数据大小与接收的数据大小可能不同。使用单个 “ 空闲 ” 位传输周期来区分发送和接收过程。但连续的数据传输并不使用单个 “ 空闲 ” 位传输周期分开。

National Semiconductors SPI 协议仅支持模式 0：数据在 SCLK 的下降沿上输出，并在 SCLK 的上升沿上被捕获。

图 15-6 描述了一个数据传输和两个连续的数据传输。在这两种情况下，发送数据的传输大小为 8 位，接收数据的传输大小为 4 位。

图 15-6. SPI NS 数据传输示例图



将 SCB 配置为 SPI NS 模式

要想将 SCB 配置为 SPI NS 模式，请按下列步骤置位各种寄存器位：

1. 将 ‘01’ 写入到 SCB_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI。
2. 将 ‘10’ 写入到 SCB_SPI_CTRL 寄存器中的 MODE （位 [25:24]）来选择 SPI NS 模式。
3. 执行第 128 页上的 “ 使能和初始化 SPI” 一节中所描述的第 2 到第 5 步。

请注意， PSoC Creator 在组件定制器的帮助下自动执行这些所有操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册 （TRM）。

15.2.4 通过使用 SPI 主设备为从设备提供时钟脉冲

在一个正常的 SPI 主设备模式传输中，只有 SCB 被使能并且数据被传输时，才会生成 SCLK。可以改变生成 SCLK 的条件，以便只要使能 SCB 就可以在 SCLK 线路上始终生成一个时钟。从设备使用 SCLK 来执行 SPI 功能和其它功能时，会采用这种设置。要想实现该操作，需要将 ‘1’ 写入到 SCB_SPI_CTRL 寄存器的 SCLK_CONTINUOUS（位 5）内。

15.2.5 Easy SPI 协议

Easy SPI （EZSPI）协议是以 Motorola SPI 为基础的，并可以在任意模式（0、1、2、3）下运行。通过该协议可以在主设备和从设备之间进行通信，并不需要 CPU 对单独帧的干预。

EZSPI 协议定义了 8 位 EZ 地址，该地址索引位于从设备中的存储器阵列（支持 32 个宽度为 8 位的地址）。要想寻址这 32 个地址，需要使用 EZ 地址的低 5 位。所有 EZSPI 数据传输都有 8 位数据帧。

注意： SCB 有一个 FIFO 存储器（该存储器是一种 16 字 x 16 位的 SRAM）并使能了字节写入。 EZ 和非 EZ 功能的访问方式有所不同。在非 EZ 模式下， FIFO 被分成 TXFIFO 和 RXFIFO，各含 8 个条目，每个条目 16 位。每个 16 位宽度的条目用于调节可配置的数据宽度。在 EZ 模式下，该 FIFO 作为一个 32x8 位的 EZFIFO 使用，因为在该模式下只能使用一个固定的 8 位宽数据。

EZSPI 包括三种传输类型：将主设备中的 EZ 地址写入从设备内，将主设备中的数据写入已寻址的从设备存储器地址，主设备读取已寻址的从设备存储器地址。

15.2.5.1 EZ 地址的写操作

EZ地址的写操作开始于MOSI线上的一个指令字节（0x00），用于指出主设备需要写入EZ地址。然后，从设备将驱动MISO线上的回复字节，以确认是否收到该指令（0xFE 或 0xFF）。MOSI 线上的第二个字节为 EZ 地址。

15.2.5.2 存储器阵列的写操作

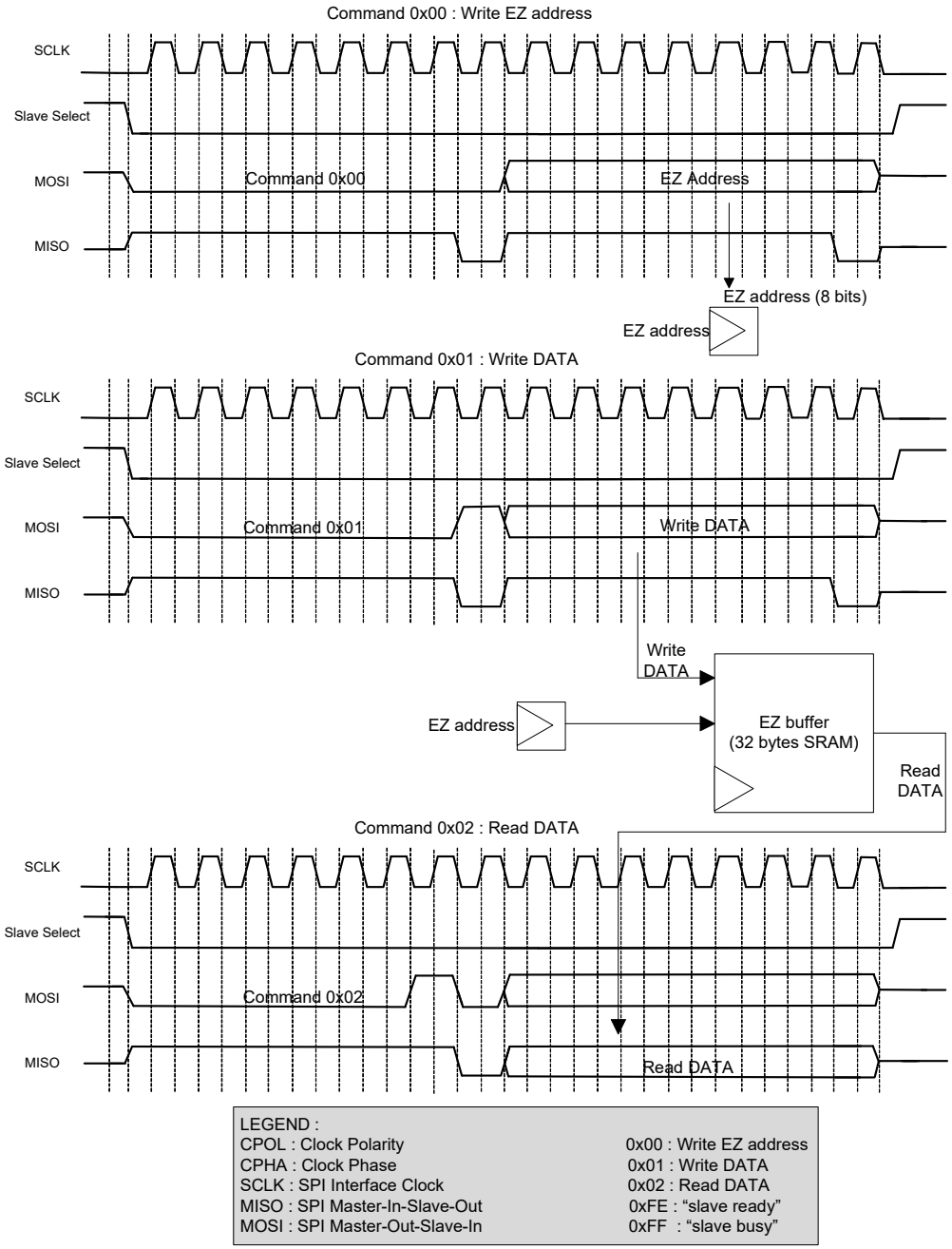
存储器阵列索引的写操作开始于 MOSI 线上的指令字节（0x01），以指出主设备将写入存储器阵列。然后，从设备将在 MISO 线上驱动一个回复字节，以确认是否收到该指令（0xFE 或 0xFF）。MOSI 线上的任何其它写数据字节将被写入到由通信 EZ 地址指出的存储器阵列地址内。各字节被写入到存储器阵列时，从设备自动递增 EZ 地址。 EZ 地址超过最大的存储器条目数量 （32）时，它将保持该数值，并且不会返回 0。

15.2.5.3 存储器阵列的读操作

存储器阵列索引的读操作开始于 MOSI 线上的指令字节（0x02），以指出主设备将读取存储器阵列。然后，从设备将在 MISO 线上驱动一个回复字节，以确认是否收到该指令（0xFE 或 0xFF）。MOSI 线上的任何其它读数据字节将从由通信 EZ 地址指出的存储器阵列地址中被读取。从存储器阵列中读取各字节时，从设备自动递增 EZ 地址。 EZ 地址超过最大的存储器条目数量 （32）时，它将保持该数值，并且不会返回 0。

图 15-7 显示的是 EZ 地址的写操作、存储器阵列的写操作和 EZSPI 协议中存储器阵列运行的读操作。

图 15-7. EZSPI 示例



15.2.5.4 将 SCB 配置为 EZSPI 模式

默认情况下，将 SCB 配置为非 EZ 工作模式。要想将 SCB 配置为 EZSPI 模式，请按下列步骤置位各个寄存器位：

1. 通过将 ‘1’ 写入到 SCB_CTRL 寄存器的 EZ_MODE 位 （位 10）上可选中 EZ 模式。
2. 通过向 SCB_SPI_CTRL 寄存器的 CONTINUOUS 位上写入 ‘1’，为发送器选择连续发送模式。
3. 执行第 128 页上的 “ 使能和初始化 SPI” 一节中所描述的第 2 到第 5 步。

请注意， PSoC Creator 在组件定制器的帮助下自动执行这些所有操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册 （TRM） *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册（TRM）*。

15.2.6 SPI 寄存器

通过使用表 15-1 中所列出的 32 位控制和状态寄存器集可以控制 SPI 接口。更多有关这些寄存器的信息，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册 （TRM）*。

表 15-1. SPI 寄存器

寄存器名称	操作
SCB_CTRL	用于使能 SCB、选择串行接口类型 （SPI、 UART、 I²C）、选择内部 / 外部时钟操作以及 EZ/ 非 EZ 操作模式。
SCB_STATUS	在 EZ 模式下，该寄存器指出外部时钟逻辑是否正在使用 EZ 存储器。
SCB_SPI_CTRL	用于将 SPI 配置为主设备或从设备，选择 SPI 协议 （Motorola、 TI、 National）以及 Motorola SPI 协议中基于时钟的子模式 （模式 0、 1、 2、 3），并选择 TI SPI 中的 SELECT （选择）信号的类型。
SCB_SPI_STATUS	表明 SPI 总线是否处于繁忙状态，并将 SPI 从设备 EZ 地址置于内部时钟模式。
SCB_TX_CTRL	指定数据帧的宽度；通过它还可以指定第一个传输位是 MSB （最高有效位）还是 LSB （最低有效位）。
SCB_RX_CTRL	它的功能与 SCB_TX_CTRL 寄存器的相同，但它用于接收器 FIFO。此外，它还决定是否在输入接口线路上使用中值滤波器。
SCB_TX_FIFO_CTRL	用于指定触发电平，移除发送器 FIFO 和移位寄存器，并执行冻结 （FREEZE）发送器 FIFO。
SCB_RX_FIFO_CTRL	它的功能与 SCB_TX_FIFO_CTRL 寄存器相同，但该寄存器使用于接收器。
SCB_TX_FIFO_WR	保存被写入到发送器 FIFO 内的数据帧。该寄存器的功能类似于 PUSH 操作。
SCB_RX_FIFO_RD	保存从接收器 FIFO 中读取的数据帧。读取某个数据帧后会从 FIFO 中移除它 — 它的性能类似于 POP 操作模式中的寄存器。当软件读取数据帧时，该寄存器会引起意外影响，即从 FIFO 中移除数据帧。
SCB_RX_FIFO_RD_SILENT	保存从接收器 FIFO 中读取的数据帧。读取某个数据帧后并不会在 FIFO 中移除它，与 PEEK 操作相似。
SCB_RX_MATCH	保持从设备地址和掩码值。
SCB_TX_FIFO_STATUS	指示存储在发送器 FIFO 中的字节数、硬件读取数据帧的位置 （读取指针）、新数据帧被编写的位置 （写入指针），并决定发送 FIFO 是否保存有效的数据。
SCB_RX_FIFO_STATUS	它的功能与 SCB_TX_FIFO_STATUS 寄存器相同，但它使用于接收器。
SCB_EZ_DATA	存储来自 EZ 寄存器位置中的数据

15.2.7 SPI 中断

SPI 支持内部和外部中断请求。本部分列出了内部中断事件。 PSoC Creator 生成所需要的中断服务子程序 （ISR），以处理缓冲器管理中断。通过将外部中断组件连接到 SPI 组件的中断输出 （使能外部中断），可以使用自定义 ISR。

SPI 预定义中断分为 TX 中断和 RX 中断。 TX 中断输出是所有 TX 中断源组的逻辑 OR 运算结果。任何已使能的 TX 中断源为 ‘true’ 时，该信号将变为高电平。RX 中断输出是所有 RX 中断源组的逻辑 OR 运算结果。任何已使能的 Rx 中断源为 ‘true’ 时，该信号将变为高电平。通过使用多种中断寄存器可以确定实际的中断源。

SPI 支持在发生以下事件时生成中断：

- 完成 SPI 主设备传输
- SPI 总线错误 — 在 SPI 传输过程中，意外取消选择 SPI 从设备
- 发生任何 EZSPI 传输后，取消选择 SPI 从设备
- 发生 EZSPI 写传输后，取消选择 SPI 从设备
- TX
  - TX FIFO 中的输入条目数量小于 SCB_TX_FIFO_CTRL 中 TRIGGER_LEVEL 指定的值
  - TX FIFO 未满
  - TX FIFO 为空
  - TX FIFO 上溢
  - TX FIFO 下溢
- RX
  - RX FIFO 已满
  - RX FIFO 非空
  - RX FIFO 上溢
  - RX FIFO 下溢
- 使用外部时钟源的 SPI
  - 从设备被选中时发送唤醒请求
  - 每次传输结束时检测到 SPI STOP 条件
  - 在写传输结束时，进行 SPI STOP 检测
  - 在读传输结束时，进行 SPI STOP 检测

注意 SPI 中断信号固定连线到 Cortex-M0 NVIC，但不能连接到外部引脚。

15.2.8 使能和初始化 SPI

必须按下列步骤编程 SPI：

1. 根据表 15-3，使用 SCB_SPI_CTRL 寄存器编程协议特定信息。该操作包括选择协议子模式和选择主设备、从设备的功能。EZSPI 仅能在从设备模式下使用。
2. 使用 SCB_TX_CTRL 和 SCB_RX_CTRL 寄存器编程通用的发送器和接收器信息，如表 15-4 中所示：
   a. 指定数据帧的宽度。 EZSPI 的数据帧的宽度始终为 8。
   b. 指定最高有效位或最低有效位作为发送 / 接收的第一位。 EZSPI 会始终先选择最高有效位。
3. 分别使用 SCB_TX_FIFO_CTRL 和 SCB_RX_FIFO_CTRL 寄存器来编程发送器和接收器 FIFO，如表 15-5 中所示。
   a. 设置触发电平。
   b. 清除发送器、接收器 FIFO 和移位寄存器。
   c. 冻结 TX 和 RX FIFO。
4. 通过编程 SCB_CTRL 寄存器使能 SCB 模块。另外，选择工作模式。这些寄存器位显示在表 15-2 中。

5. 使能该模块 （将 ‘1’ 写入 SCB_CTRL 寄存器中的 ENABLED 位）。使能该模块后，不该修改控制位。只有该模块被禁用后，才应该修改这些位。例如，修改工作模式（从 Motorola 模式变为 TI 模式）或从外部时钟操作转换为内部时钟操作。只有重新使能该模块后，这些修改才生效。请注意，重新使能该模块将导致重新初始化，而且将丢失相关状态（例如，FIFO 状态）。

表 15-2. SCB_CTRL 寄存器

位	名称	数值	说明
[25:24]	MODE	00	I^2C 模式
		01	SPI 模式
		10	UART 模式
		11	保留
31	ENABLED	0	SCB 模块被禁用
		1	使能 SCB 模块

表 15-3. SCB_SPI_CTRL 寄存器

位	名称	数值	说明
[25:24]	MODE	00	SPI Motorola 子模式。（这是 EZSPI 支持的唯一模式。）
		01	SPI Texas Instruments 子模式。
		10	SPI National Semiconductors 子模式。
		11	保留。
31	MASTER_MODE	0	从设备模式。（这是 EZSPI 支持的唯一模式。）
		1	主设备模式。

表 15-4. SCB_TX_CTRL/SCB_RX_CTRL 寄存器

位	名称	说明
[3:0]	DATA_ WIDTH	‘DATA_WIDTH + 1’ 指的是传输或接收数据帧中的位数。有效范围为 [3，15]。它不包括起始位、停止位和奇偶校验位。对于 EZSPI，该值为 ‘0b0111’。
8	MSB_FIRST	1：表示 MSB 优先 0= LSB 最优先对于 EZSPI，该值为 1。
9	MEDIAN	仅用于 SCB_RX_CTRL。 它决定是否在输入接口线路上使用数字三抽头中值滤波器。该滤波器会降低对错误的敏感性，但是它要求更高的过采样值。 1 = 使能 0 = 禁用

表 15-5. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL 寄存器

位	名称	说明
[3:0]	TRIGGER_LEVEL	触发电平。当发送器 FIFO 所输出的数据小于该字段的值或接收器 FIFO 所接收到的数据大于该字段的值时，在相应的情况下产生一个发送器或接收器的触发事件。
16	CLEAR	该位被设置为 ‘1’ 时，发送器或接收器 FIFO 和移位寄存器均被清除。
17	FREEZE	将该位设置为 ‘1’ 时，硬件对发送器或接收器 FIFO 进行读 / 写操作不会产生任何影响。冻结操作不会增加 TX 或 RX FIFO 的读 / 写指针。

15.2.9 内部和外部时钟的 SPI 操作

SCB 为 SPI 和 I^2C 功能支持从内部和外部提供时钟脉冲的操作。从内部提供时钟脉冲的操作使用由器件提供的时钟。外部时钟操作使用串行接口提供的时钟。通过外部时钟可以启用深度睡眠系统功耗模式中的操作。

内部时钟操作使用系统高频时钟 （HFCLK）。更多有关系统时钟的信息，请参考第 83 页上的时钟系统章节。该模式还支持过采样。对高频时钟进行过采样。通过 SCB_CTRL 寄存器的 OVS （位 [3:0]），可以指定过采样操作。

在 SPI 主设备模式下，过采样的有效范围为 4 到 16。因此，时钟速度为 48 MHz 时，最大比特率为 12 Mbps。然而，如果考虑输入 / 输出单元和路由延迟，则必须将过采样范围设置为 6 到 16，以确保正常操作。因此，最大比特率为 8 Mbps。**注意：**要想获取最大的比特率，需要在 SPI 主设备模式下将 LATE_MISO_SAMPLE 设置为 ‘1’ 。它的默认值为 ‘0’ 。

在 SPI 从设备模式下，不使用 SCB_CTRL 寄存器的 OVS 字段（位 [3:0]）。然而，根据接口时钟（SCLK），SCB 时钟有一个频率要求。该要求以一个比例值 （SCB 时钟 /SCLK）表示。该比例大小取决于两个字段：SCB_RX_CTRL 寄存器的 MEDIAN 位和 SCB_CTRL 寄存器的 LATE_MISO_SAMPLE 位。如果外部 SPI 主设备支持 MISO 延迟采样而且中值位被设置为 ‘0’，则最大数据速率可达 16 Mbps。如果外部 SPI 主设备不支持 MISO 延迟采样，最大数据速率被限制为 8 Mbps。根据这些位，最大比特率在表 15-6 中显示。

表 15-6. SPI 从设备的最大数据速率

外设时钟频率为 48 MHz 时的最大比特率	比率要求	SCB_RX_CTRL 的中值	SCB_CTRL 的 LATE_MISO_SAMPLE
8 Mbps	≥ 6	0	1
6 Mbps	≥ 8	1	1
4 Mbps	≥ 12	0	0
3 Mbps	≥ 16	1	0

从外部提供时钟脉冲的操作仅适用于：

- 从设备功能。
- EZ功能。EZ功能将模块SRAM作为存储器结构使用。非EZ功能将模块SRAM作为TX和RX FIFO使用；FIFO不支持外部时钟模式。
- Motorola 模式 0、 1、 2、 3。

外部时钟的 EZ 工作模式可支持 48 Mbps 的数据速率 （接口时钟频率为 48 MHz 时）。

使用 SCB_CTRL 寄存器中的以下两个字段，可以指定使用内部时钟模式还是外部时钟模式：

- **EC_AM_MODE**：指出SPI从设备选择使用内部时钟（ ‘0’ ）还是外部时钟（ ‘1’ ）。SPI从设备选择包含协议的第一部分。
- **EC_OP_MODE**：指出协议操作剩下部分 （SPI 从设备选择除外）使用内部时钟 （ ‘0’ ）还是外部时钟 （ ‘1’ ）。如上所述，外部时钟模式不支持非 EZ 功能。

通过这两个寄存器字段可以确定 SPI 的功能性能。需要根据活动、睡眠和深度睡眠系统功耗模式下所需要的性能来设置寄存器字段。如果设置错误，将引起一些系统功耗模式下错误的性能。表 15-7 和表 15-8 介绍的是SPI的设置（在EZ和非EZ模式下）。

15.2.9.1 非 EZ 工作模式

在非 EZ 模式下，有两种设置情况。由于非 EZ 功能不支持外部时钟操作 （无 FIFO 支持），因此始终需要将 EC_OP_MODE 设置为 ‘0’ 。但仍可以将 EC_AM_MODE 设置为 0 或 1。表 15-7 显示的是各种可能情况的概述。

表 15-7. 非 EZ 工作模式中的 SPI 操作

SPI （非 EZ）模式				
系统功耗模式	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
活动和睡眠	使用内部时钟选择。 使用内部时钟执行其它操作。	使用外部时钟选择。 使用内部时钟执行其它操作。 在活动模式中，唤醒中断源处于禁用状态 （MASK = 0）。 在睡眠模式下，用户可以配置 MASK 位。	不支持	不支持
深度睡眠	不支持	使用外部时钟选择：唤醒中断源处于使能状态 （MASK = 1）。 发送 0xFF。		
休眠 停止	在这些模式下， SCB 不可用 （请参考第 101 页上的功耗模式章节）			

EC_OP_MODE 为 ‘0’ 和 EC_AM_MODE 为 ‘0’ ：只在活动和睡眠系统功耗模式下，该设置才有效。整个模块的功能使用内部时钟。

EC_OP_MODE 为 ‘0’ 和 EC_AM_MODE 为 ‘1’ ：在活动和睡眠系统功耗模式下，该设置有效；在深度睡眠系统功耗模式下，模块被限于唤醒功能。 SPI 从设备选择通过外部时钟逻辑执行：在活动系统功耗模式下，内部和外部时钟逻辑都有效；在深度睡眠系统功耗模式下，只有外部时钟逻辑有效。外部时钟逻辑检测到从设备选择时，它会设置唤醒中断原位。通过该位可以生成用于唤醒 CPU 的中断。

- 在活动系统功耗模式下，CPU 和模块的内部时钟操作有效，并且唤醒中断源被禁用（相应的掩码位为 ‘0’ ）。但在睡眠模式下，根据应用要求，可以使能或禁用唤醒中断源位（掩码位为 ‘1’ 或 ‘0’ ）。睡眠模式中的其他操作与活动模式的相同。内部时钟操作将控制正在进行的 SPI 传输。
- 在深度睡眠系统功耗模式下，需要唤醒 CPU 并使能唤醒中断源位（掩码位为 ‘1’ ）。唤醒耗时较长，因此正在进行的 SPI 传输被否定确认 （ ‘1’ 位或 “0xFF” 字节被发送到 MISO 线），并且唤醒时，内部时钟逻辑将控制下一个 SPI 传输。

15.2.9.2 EZ 工作模式

EZ 模式具有三种设置情况。当 EC_OP_MODE 为 ‘0’ 时，可将 EC_AM_MODE 设置为 ‘0’ 或 ‘1’ ；当 EC_OP_MODE 为 ‘1’ 时，必须将 EC_AM_MODE 设置为 ‘1’ 。表 15-8 提供了可能情况的概述。灰色显示的单元表示可能发生但并不得推荐的设置。因为这些设置需要从外部时钟逻辑 （从设备选择）切换为内部时钟逻辑 （剩下操作）。 EC_AM_MODE = 0 和 EC_OP_MODE = 1 的组合无效，并且模块不会响应。

表 15-8. EZ 模式中的 SPI 操作

SPI， EZ 模式				
系统功耗模式	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
活动和睡眠	使用内部时钟选择。 使用内部时钟执行其它操作。	使用外部时钟选择。 使用内部时钟执行其它操作。 在活动模式中，唤醒中断源处于禁用状态 （MASK = 0）。 在睡眠模式下，用户可以配置 MASK 位。	无效	使用外部时钟选择。 使用外部时钟执行其它操作。
深度睡眠	不支持	使用外部时钟选择：唤醒中断源处于使能状态 （MASK = 1）。 发送 0xFF。		使用外部时钟选择。 使用外部时钟执行其它操作。
休眠模式 停止	在这些模式下， SCB 不可用 （请参考第 101 页上的功耗模式章节）			

EC_OP_MODE 为 0 和 EC_AM_MODE 为 0：只在活动和睡眠系统功耗模式下，该设置才有效。整个模块的功能使用内部时钟。

EC_OP_MODE 为 ‘0’ 和 EC_AM_MODE 为 ‘1’：在活动和睡眠系统功耗模式下，该设置有效；在深度睡眠系统功耗模式下，该设置限于唤醒功能。SPI 从设备选择通过外部时钟逻辑执行：在活动系统功耗模式下，内部和外部时钟逻辑均有效；在深度睡眠系统功耗模式下，只有外部时钟逻辑有效。外部时钟逻辑检测到从设备选择时，它会设置唤醒中断原位。通过该位可以生成用于唤醒 CPU 的中断。

- 在活动系统功耗模式下， CPU 和模块的内部时钟操作有效，而且唤醒中断源被禁用 （相应的掩码位为 ‘0’）。但在睡眠模式下，根据应用要求，可以使能或禁用唤醒中断源位 （掩码位为 ‘1’ 或 ‘0’）。睡眠模式的其余操作与活动模式的相同。内部时钟操作将控制正在进行的 SPI 传输。
- 在深度睡眠系统功耗模式下，需要唤醒CPU并使能唤醒中断源位 （掩码位为 ‘1’）。唤醒耗时较长，因此正在进行的 SPI 传输被否定确认 （‘1’ 位或 “0xFF” 字节被发送到 MISO 线），而且唤醒时，内部时钟逻辑将控制下一个 SPI 传输。

EC_OP_MODE 为 ‘1’ 和 EC_AM_MODE 为 ‘1’：在活动、睡眠和深度睡眠模式下，该设置均有效。 SCB 功能由外部时钟逻辑执行。请注意，该设置会引起对模块 SRAM 进行外部时钟的访问。这些访问会与器件的内部时钟访问发生冲突。这样会导致等待状态或总线错误的结果。通过 SCB_CTRL 寄存器的 FIFO_BLOCK 字段，可以指定生成等待状态 （‘1’）还是总线错误 （‘0’）。

15.3 UART

通用异步接收器 / 发送器（UART）协议是一个异步串行接口协议。UART 通信通常为点至点通信。UART 接口包括以下两个信号：

- TX：发送器输出
- RX：接收器输入

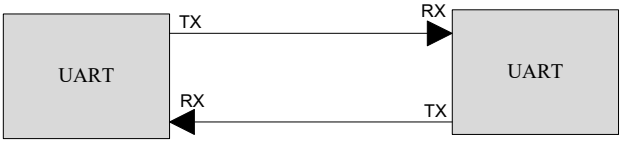
15.3.1 特性

- 异步发送器和接收器功能
- 最大数据速率可达 3 Mbps
- 支持 UART 协议
  - 标准 UART
  - SmartCard （ISO7816）读卡器。
  - IrDA
- 支持局部互连网络 （LIN）
  - 断点检测
  - 波特率检测
  - 冲突检测 （能够检测到一个驱动位值不被反映到总线上，表示另一个组件在驱动同一个总线）
- 多处理器模式
- 可编程数据帧大小，取值范围为 4 到 9 位
- 可编程停止位数，其取值范围为 1 到 4（以半位周期为单位）
- 支持奇偶校验 （奇校验和偶校验）
- 中断或轮循 CPU 接口
- 可编程过采样

15.3.2 概述

图 15-8 显示的是标准的 UART TX 和 RX。

图 15-8. UART 示例



典型的 UART 传输依次包括：一个起始位、多个数据位、可选的奇偶检验位和一个或多个停止位。起始位和停止位表示数据传输的开始和结束。奇偶校验位由发送器发送，接收器可以使用该位来检测单比特错误。由于该接口没有异步时钟，因此发送器和接收器使用自己的时钟。另外，它们还需要对位传输的周期一致。

支持三种不同的串行接口协议：

- 标准的 UART 协议
  - 多处理器模式
  - 局部互联网络 （LIN）
- SmartCard — 与 UART 相似，但可以发送否定确认（NACK）信号
- IrDA，是 UART 协议通过使用调制方案进行修改而得到的协议。

默认情况下， UART 支持 8 位的数据帧宽度。然而，可以将该值配置为 4 到 9 范围内的任意值。该值不包含起始位、停止位和奇偶校验位。停止位数量的取值范围可以为 1 到 4。可以使能或禁用奇偶校验位。如果使能该位，可以将奇偶校验位设置为奇校验或偶校验。只在标准UART和SmartCard UART模式下，才能使用奇偶校验位。在 IrDA UART 模式下，奇偶校验位自动被禁用。图 15-9 显示的是 SCB 的 UART 接口的默认配置。

注意：UART 接口不支持外部时钟操作。因此，UART 只能在活动和睡眠系统功耗模式下运行。

15.3.3 UART 工作模式

15.3.3.1 标准协议

典型的 UART 传输包括一个起始位，然后是多个数据位、可选的奇偶检验位和一个或多个停止位。起始位值始终为 ‘0’；数据位值取决于传输的数据；奇偶校验位值设置为可保证对数据位进行奇数或偶数校验的值；停止位值为 ‘1’。奇偶校验位由发送器生成，接收器可以使用该位来检测单比特传输错误。不传输数据时，TX 线为 ‘1’，与停止位的值相同。

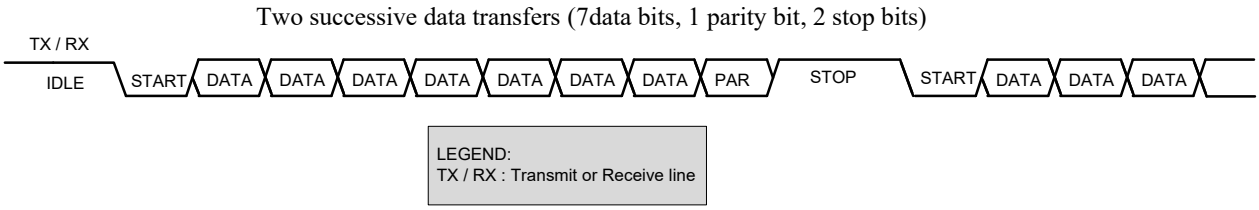
由于该接口没有时钟，所以发送器和接收器的单比特传输周期需要相互匹配。发送器和接收器自有的内部时钟。接收器时钟的运行频率高于位传输频率，因此该接收器可以对输入信号进行过采样。

通过将 TX 线上的值 ‘1’ 改为 ‘0’，可以将停止位切换为起始位。接收器可利用这种切换与发送器的时钟进行同步。在每次数据传输开始时进行同步，这样即使在发送器和接收器时钟间出现频率偏移时也可实现无差错的数据传输。要求的时钟精度取决于数据传输大小。

对于发送器和接收器，各连续数据传输间的停止周期或停止位数量通常一样，并在 1 至 3 个比特传输周期范围内。

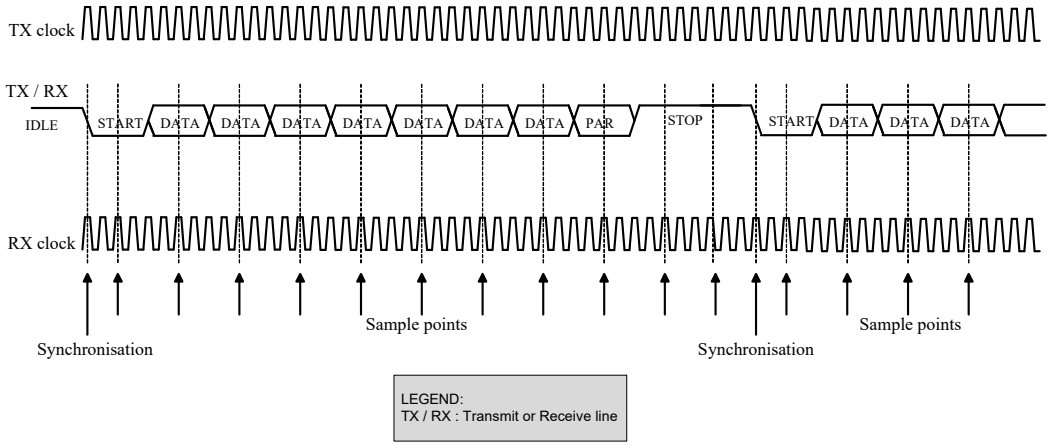
图 15-9 显示的是 UART 协议。

图 15-9. 标准 UART 协议示例



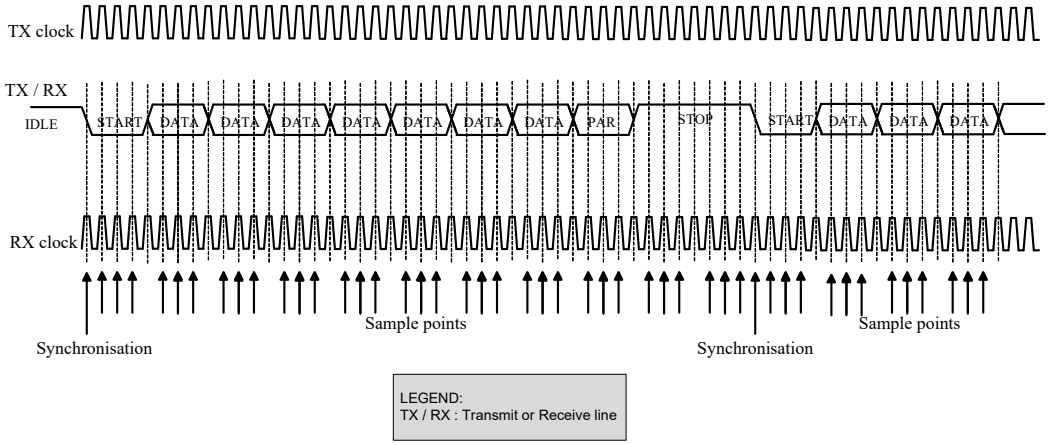
接收器过采样输入信号；位传输周期中间的采样点值被使用 （该传输周期在接收器时钟上进行）。请参考图 15-10。

图 15-10. 标准 UART 协议示例 （单样本采样）



或者，位传输周期中间附近 （该传输周期在接收器时钟上进行）的三个采样用于多数表决，以提高精度。请参考图 15-11。

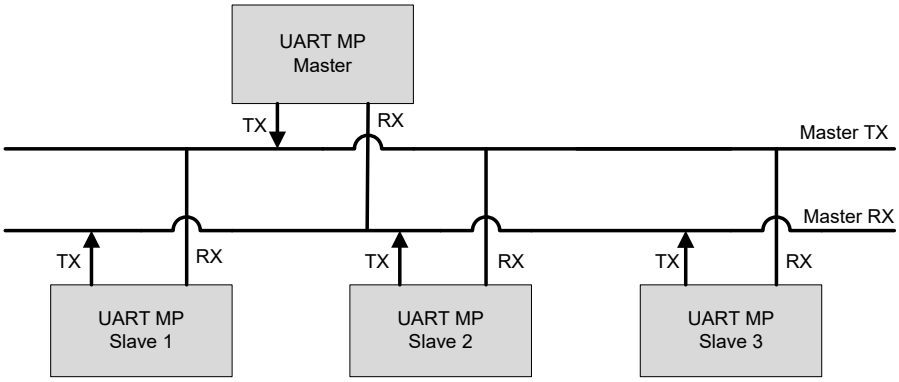
图 15-11. 标准 UART 协议 （多样本采样）



UART 多处理器模式

UART_MP （多处理器）模式被定义为 “ 单主多从 ” 拓扑结构，如图 15-12 所示。该模式还被视为 UART 9 位协议，因为数据域的宽度为 9 位。 UART_MP 是标准 UART 模式的一部分。

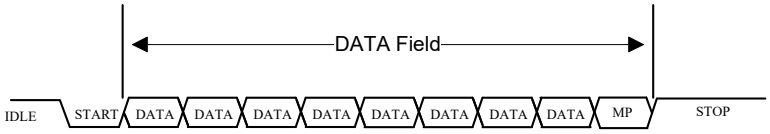
图 15-12. UART MP 模式总线连接



UART_MP 模式的主要特性是：

- 一个主设备与多个从设备接口 （多分支网络）。
- 每个从设备都通过独特的地址被识别。
- 使用 9 位数据域，将第 9 位作为地址 / 数据的标志（MP 位）。该位被设置为高电平时，表示地址字节；被设置为低电平时，则表示数据字节。数据帧显示在图 15-13。
- 奇偶校验位被禁用。

图 15-13. UART MP 数据帧

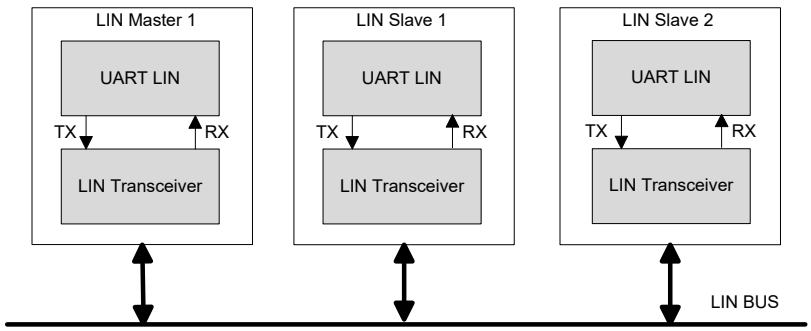


在 UART_MP 模式中， SCB 可作为主设备或从设备使用。需要将 SCB_TX_CTRL 和 SCB_RX_CTRL 寄存器的数据帧大小设置为 9 位。 SCB 作为 UART_MP 主设备使用时，固件将修改每个地址或数据帧的 MP 标志。当它作为 UART_MP 从设备使用时，需要将 SCB_UART_RX_CTRL 寄存器的 MP_MODE 字段设置为 ‘1’ 。另外，还需要为从设备地址和地址掩码设置 SCB_RX_MATCH 寄存器。SCB_CTRL 寄存器的 ADDR_ACCEPT 字段为 ‘1’ 时，匹配地址将被写入到 RX_FIFO 内。如果接收地址与它自己的地址不匹配，该接口将忽略后面的数据，直到接收到用于比较的下个地址为止。

UART 局部互联网络 （LIN）模式

SCB 将局部互联网络 （LIN）协议作为标准 UART 的一部分使用。使用一主多从的拓扑结构设计 LIN。 LIN 总线上有一个主设备节点和许多从设备节点。 SCB UART 支持 LIN 主设备和从设备功能。 LIN 规范定义了物理层 （第一层）和数据链路层 （第二层）。图 15-14 显示的是 UART_LIN 和 LIN 收发器。

图 15-14. UART_LIN 和 LIN 收发器

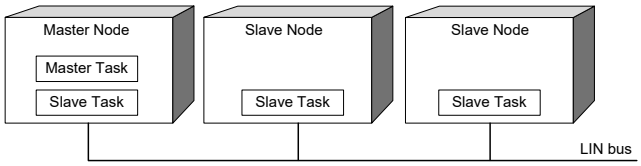


LIN 协议定义了两种任务：

- 主任务：发送数据包头，以启动 LIN 传输。
- 从任务：发送或接收响应。

主设备节点支持主任务和从任务；从设备节点仅支持从任务，如图 15-15 中所示。

图 15-15. LIN 总线节点和任务

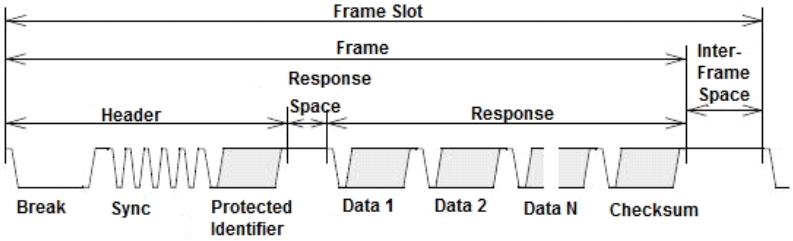


LIN 帧结构

LIN 基于预定时期内的帧传输。一帧分为头域和响应域，如图 15-16 中所示。

- 头域包括：
  - 断开字段 （至少 13 位周期，该值为 ‘0’ ）。
  - 同步字段 （0x55 字节帧）。通过使用同步字段可以将从任务的时钟同步到主任务的时钟。
  - 标识符字段 （该帧用于指定特殊从设备）。
- 响应域包括数据和校验和。

图 15-16. LIN 帧结构

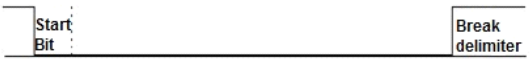


在 LIN 协议通信过程中，先发送最低有效位 （LSB）数据，最后发送最高有效位 （MSB）。起始位被编码为 ‘0’ ，停止位被编码为 ‘1’ 。以下章节将介绍 LIN 帧中的所有字节段。

中断域

每个新帧均以始终由主设备生成的中断域开始。中断域至少有 13 位时间 （其逻辑为零），最后是中断分隔符。图 15-17 显示了中断域结构。

图 15-17. LIN 中断域



同步域

这是主设备在头域中所传输的第二个域，它的值为 0x55。通过同步域可以使从设备和主设备的时钟同步，以便自动检测波特率。图 15-18 显示了 LIN 同步域的结构。

图 15-18. LIN 同步域



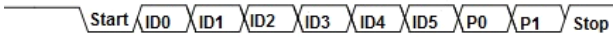
受保护的标识符 （PID）域

一个受保护的标识符域包括两个子域：帧标识符 （位 0-5）和奇偶校验 （位 6 和位 7）。图 15-19 显示了 PID 域结构。

- 帧标识符：分为 3 类
  - 0 至 59 的值 （0x3B）适用于带信号的帧
  - 60 （0x3C）和 61 （0x3D）用于储存诊断和配置数据
  - 62 （0x3E）和 63 （0x3F）被保留，以供将来进行协议的增强
- 奇偶校验：帧标识符位用于计算奇偶校验

图 15-19 显示的是 PID 域的结构。

图 15-19. PID 域



数据 . 在 LIN 中，每一帧可以带最少一个、最多 8 个字节的数据。此处，先发送最低有效位的数据字节，最后发送最高有效位的数据字节。

校验和

校验和是 LIN 帧中的最后字节字段。通过反转所有数据字节的带进位的 8 位总和，或反转所有数据字节和 PID 字段的带进位的 8 位总和，可以计算出该校验和。 LIN 帧中的校验和分两种，分别是：

- 传统校验和：它是通过计算所有数据字节得出的校验和 （在 LIN 1.x 从设备中使用）。
- 增强校验和：是通过计算所有数据字节以及受保护标识符得出的校验和 （在 LIN 2.x 从设备中使用）。

LIN 帧类型

帧类型是由用于传输帧的有效条件决定的。根据 LIN 规范，一共有五种不同的 LIN 帧类型。一个节点或集群不用支持所有帧类型。

无条件帧

这种帧会带信号和其帧标识符（在 0x00 至 0x3B 的范围内）。接收器会接收这些帧并使其适用于应用；帧的发送器会将响应发送给头域。

基于事件触发的帧

这种类型的帧用于增加 LIN 集群的响应能力，并不需要过多的总线带宽来对多个从设备轮询要求有关极少发生的事件。基于触发事件的帧带一个或多个无条件帧的响应。与事件触发帧相互关联的无条件帧应该：

- 具有相同的长度
- 使用相同的校验方法 （传统或增强）
- 将第一个数据字段保留在它的受保护标识符内
- 由不同的从设备节点发送
- 不能被直接列入相同的调度表内，这一点同基于触发事件的帧不一样

零星帧

零星帧用于将某些动态行为合并到调度表中，并不会影响该表的其他部分。零星帧具有一组无条件帧，这些无条件帧共享了帧插槽。需要传输零星帧时，必须检查信息，以确认无条件帧是否有被更新。如果没有任何信号被更新，那么不会传输帧，并且帧插槽为空。

诊断帧

诊断帧始终带有传输层的数据，并包含八个数据字节。

诊断帧的标识符是：

- 主设备请求帧 （0x3C），或
- 从设备响应帧 （0x3D）

发送主设备请求帧前，主设备会要求它的诊断模块检查该请求帧是否被发送或总线是否处于闲置状态。从设备响应帧的头域将被无条件发送。根据其诊断模块，从设备会进行发送和接收响应。

保留帧

这些帧被保留以供将来使用，它们的帧标识符为 0x3E 和 0x3F。

LIN 进入睡眠模式和唤醒指令

如果主设备发送了 “ 进入睡眠模式 ” 指令， LIN 协议可使 LIN 总线处于睡眠模式。 “ 进入睡眠模式 ” 指令是一个主设备请求帧 （ID = 0x3C），该请求帧的第一个字节段是 0x00，其余字节段被设置为 0xFF。收到 “ 进入睡眠模式 ” 指令后，从设备节点应用仍处于活动状态。这是应用特有的反应。如果 LIN 总线处于非活动状态的时间超过了 4 秒，则 LIN 从设备节点会自动进入睡眠模式。

如果控制总线的时间属于 250 µs 至 5 ms 的范围，那么在连接到 LIN 总线的任意节点上（是 LIN 主设备或任意 LIN 从设备）可以进行唤醒。每个从设备都会检测唤醒请求，并准备在 100 ms 内处理头域。主设备也会检测唤醒请求，当从设备节点处于活动状态时，主设备会开始发送头域。

要想支持 LIN，则需要使用一个专用 （片下）线驱动器 / 接收器。LIN 总线上的供电电压范围为 7 V ~ 18 V。一般情况下， LIN 线驱动器使用 SCB TX 线上提供的电压值驱动 LIN 线，并将 LIN 线上的电压值传输到 SCB RX 线上。通过将 SCB 中的 TX 和 RX 线进行比较，可以检测到总线冲突 （由 SCB_INTR_TX 寄存器中的 SCB_UART_ARB_LOST 字段指出）。

将 SCB 配置为标准的 UART 接口

要想将 SCB 配置为标准的 UART 接口，请按下列步骤置位各种寄存器位：

1. 通过将 ‘10’ 写入 SCB_CTRL 寄存器的 MODE 域 （位 [25:24]）内，将 SCB 配置为 UART 接口。
2. 通过将 ‘00’ 写入 SCB_UART_CTRL 寄存器的 MODE 域 （位 [25:24]）内，将 UART 接口配置为标准协议。
3. 要使能 UART MP 模式或 UART LIN 模式，将 ‘1’ 写入 SCB_UART_RX_CTRL 寄存器的 MP_MODE （位 10）或 LIN_MODE （位 12）内。
4. 执行第 141 页上的 “ 使能和启动 UART” 一节中所描述的第 2 到第 5 步。

请注意， PSoC Creator 在 GUI 的帮助下自动执行这些所有操作。更多有关这些寄存器的信息，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册 （TRM）*。

15.3.3.2 SmartCard （ISO7816）

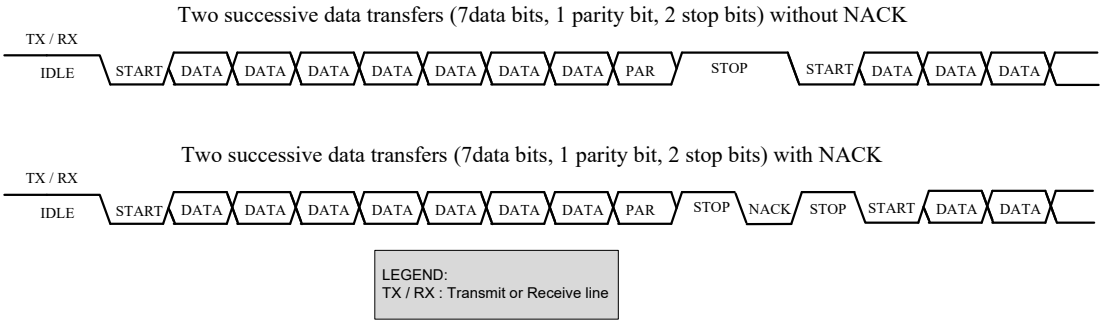
ISO7816 是使用 “ 单主单从 ” 拓扑结构的异步串行接口。 ISO7816 定义了读卡器 （主设备）和卡 （从设备）的功能。更多有关信息，请参考 ISO7816 规范中的内容。SCB 只支持主设备（读卡器）功能。通过该模块，基础物理层支持异步字符传输。通过在 UART_TX 和 UART_RX 控制模块间内部复用， UART_TX 线将与 SmartCard 的 I/O 线相连接。

SmartCard 传输与 UART 传输相似，另外可以将否定确认 （NACK）信号从接收器发送给发送器。NACK 始终为 ‘0’ 。主设备和从设备可能驱动同一条线 （甚至在不同时刻驱动）。

SmartCard 传输让发送器输出起始位和数据位（以及可选的奇偶校验位）。输出这些位后，它会释放总线，从而进入停止周期。释放后，线路的状态为 ‘1’ （即停止位的值）。一位的传输过程进入停止周期后，接收器可能在一个位传输周期时长内驱动线路上的 NACK（ ‘0’ 值）。观察到该 NACK 后，发送器将通过延长停止周期的方式做出反应，延长时间为一个位传输周期。要想让该协议奏效，停止期应大于一个位传输期。请注意，带 NACK 的数据传输比不带 NACK 的数据传输所用的时间长单比特传输期。通常，协议的实现使用的是带上拉电阻的三态驱动器，这样当线路不发送数据或发送停止位时，它的值将为 ‘1’ 。

图 15-20 显示的是 SmartCard 协议。

图 15-20. SmartCard 示例图



ISO7816 的通信波特率如下所示：

波特率 = f_{7816} × (D/F)

其中 f_{7816} 是时钟频率， F 是时钟频率转换整数， D 是波特率调整整数。

默认情况下，F = 372、D = f1 和最大时钟频率为 5 MHz。因此，最大波特率为 13.4 Kbps。一般情况下，选择 3.57 MHz 时钟。波特率的典型值为 9.6 Kbps。

将 SCB 配置为 UART SmartCard 接口

要将 SCB 配置为 UART SmartCard 接口，需要根据下列步骤设置各寄存器位；请注意， PSoC Creator 在 GUI 的帮助下自动执行所有的这些操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册（TRM）。

1. 通过将 ‘10’ 写入 SCB_CTRL 寄存器的 MODE （位 [25:24]）内，将 SCB 配置为 UART 接口。
2. 通过向 SCB_UART_CTRL 寄存器的 MODE （位 [25:24]）内写入 ‘01’，可将 UART 接口作为 SmartCard 协议运行。
3. 执行第 141 页上的 “ 使能和启动 UART” 一节中所描述的第 2 到第 5 步。

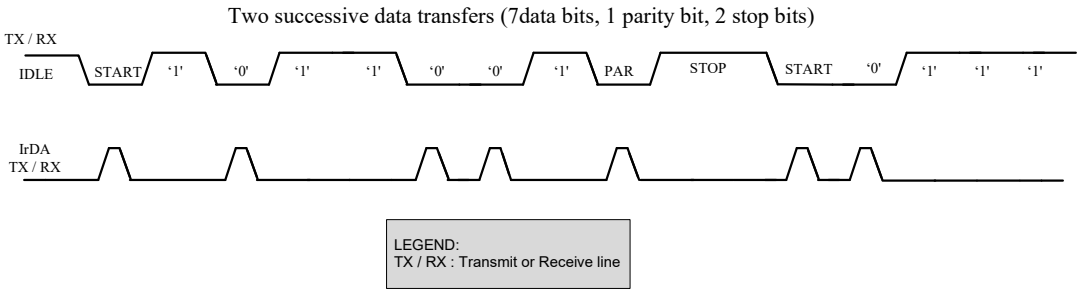
15.3.3.3 IrDA

通过使用 UART 接口，SCB 为红外数据关联性（IrDA）协议支持高达 115.2 Kbps 的数据速率。对于 IrDA 协议的基本物理层，它仅支持 115.2 Kbps 以下的数据速率。因此，系统使用该模块时，必须考虑如何使用其他可用资源来实现一个完整的 IrDA 通信系统。

IrDA 协议给 UART 信号添加了一个调制方案。在发送器端，对各位进行调制。在接收器端，对各位进行解调。调制方案使用的是归零反相 （RZI）格式。线路上短的 ‘1’ 脉冲会传输 ‘0’ 值的位，通过将线路保持逻辑 ‘0’ 传输 ‘1’ 值的位。对于这些数据速率（<= 115.2 Kbps），将使用 RZI 调制方案，而且脉冲的持续时间为位周期的 3/16。通过配置 SCB_CTRL 寄存器的 SCB_OVS 字段，将采样时钟频率设置为所选波特率的 16 倍。

配置相应模块的时钟频率，以得到 115.2 Kbps 以下的不同通信速度。另外，可用的速度还有 2.4 Kbps、9.6 Kbps、19.2 Kbps、38.4 Kbps 和 57.6 Kbps。IrDA 串行红外接口以 9.6 Kbps 的速度运行。图 15-21 显示的是 IrDA 对 UART 传输进行调制的方法。

图 15-21. IrDA 示例图



将 SCB 配置为 UART IrDA 接口

要将 SCB 配置为 UART IrDA 接口，需要根据下列步骤设置各寄存器位；请注意， PSoC Creator 在 GUI 的帮助下自动执行所有的这些操作。更多有关这些寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列: PSoC 4 BLE 寄存器数据参考手册（TRM）。

1. 通过将 ‘10’ 写入 SCB_CTRL 寄存器的 MODE （位 [25:24]）内，将 SCB 配置为 UART 接口。
2. 通过将 ‘10’ 写入 SCB_UART_CTRL 寄存器的 MODE （位 [25:24]）内，将 UART 接口配置为 IrDA 协议。
3. 通过将 ‘1’ 写入 SCB_RX_CTRL 寄存器的 MEDIAN （位 9）内，可以在输入接口线上使能中值滤波器。
4. 根据第 141 页上的 “ 使能和启动 UART” 一节中的内容配置 SCB。

15.3.4 UART 寄存器

通过使用表 15-9 中所列出的 32 位寄存器集可以控制 UART 接口。更多有关这些寄存器的信息，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器数据参考手册 （TRM）。*

表 15-9. UART 寄存器

寄存器名称	操作
SCB_CTRL	使能 SCB，可选择串行接口的类型 （SPI、 UART、 I^2C）
SCB_UART_CTRL	用于选择 UART 的子模式 （标准 UART、 SmartCard、 IrDA），还可用于局部环回控制。
SCB_UART_RX_STATUS	用于指定可确定位周期的 BR_COUNTER 值。该值用于设置 SCB 时钟的准确度。与 SCB_CTRL 寄存器的 OVS 位相比，该值提供的准确度更高。
SCB_UART_TX_CTRL	用于指定停止位的数量，使能奇偶校验，选择奇偶校验类型并使能收到 NACK 信号时的重新传输操作。
SCB_UART_RX_CTRL	执行与 SCB_UART_TX_CTRL 寄存器相同的功能，但还可以用于使能多个处理器模式、LIN 模式在检测到奇偶校验错误或帧错误时丢失数据。
SCB_TX_CTRL	用于指定数据帧的宽度；通过它还可以指定第一个传输位是 MSB（最高有效位）还是 LSB（最低有效位）。
SCB_RX_CTRL	它的功能与 SCB_TX_CTRL 寄存器的相同，但它用于接收器 FIFO。此外，它还决定是否在输入接口线路上使用中值滤波器。
SCB_UART_FLOW_CONTROL	用于配置 UART 发送器的流控。

15.3.5 UART 中断

UART 支持内部和外部中断请求。该部分列出了内部中断事件。 PSoC Creator 生成所需要的中断服务子程序 （ISR），以处理缓冲器管理中断。通过将外部中断组件连接到 UART 组件的中断输出 （使能外部中断），可以使用自定义 ISR。

可以将 UART 预定义中断分为 TX 中断和 RX 中断。 TX 中断输出是所有 TX 中断源组的逻辑 OR 运算结果。任何已使能的 TX 中断源为 true 时，该信号将变为高电平。 RX 中断输出是所有 RX 中断源组的逻辑 OR 运算结果。任何已使能的 Rx 中断源为 true 时，该信号将变为高电平。 UART 支持在发生以下事件时生成中断：

- TX
  - TX FIFO 中的输入条目数量小于 SCB_TX_FIFO_CTRL 中 TRIGGER_LEVEL 指定的值
  - TX FIFO 未满
  - TX FIFO 为空
  - TX FIFO 上溢
  - TX FIFO 下溢
  - TX 接收到 SmartCard 模式的 NACK
  - TX 完成
  - 仲裁丢失 （在 LIN 或 SmartCard 模式下）
- RX
  - RX FIFO 中的输入条目数量小于 SCB_RX_FIFO_CTRL 中 TRIGGER_LEVEL 指定的值
  - RX FIFO 已满
  - RX FIFO 非空
  - RX FIFO 上溢
  - RX FIFO 下溢
  - 收到的数据帧中包含的帧错误
  - 收到的数据帧中包含的奇偶校验错误
  - LIN 波特率检测完成
  - LIN 中断检测完成

15.3.6 使能和启动 UART

必须按下列步骤编程 UART：

1. 根据表 15-10，使用 SCB_UART_CTRL 寄存器编程协议特殊信息。该操作包括选择协议子模式和发送器 - 接收器功能等。
2. 使用 SCB_TX_CTRL 和 SCB_RX_CTRL 寄存器编程通用的发送器和接收器信息，如表 15-11 中所示。
   a. 指定数据帧的宽度。
   b. 指定最高有效位或最低有效位作为发送或接收的第一位。
3. 分别使用 SCB_TX_FIFO_CTRL 和 SCB_RX_FIFO_CTRL 寄存器来编程发送器和接收器 FIFO，如表 15-12 中所示。
   a. 设置触发电平。
   b. 清除发送器、接收器 FIFO 和移位寄存器。
   c. 冻结 TX 和 RX FIFO。
4. 通过编程 SCB_CTRL 寄存器使能 SCB 模块。另外，选择工作模式 （请参考表 15-13）。

5. 使能该模块 （将 ‘1’ 写入 SCB_CTRL 寄存器中的 ENABLED 位）。使能该模块后，不应修改控制位。只有禁用该模块后，才应该修改这些位。例如，修改工作模式 （从 SmartCard 改为 IrDA）。只在重新使能该模块后，这些修改才生效。请注意，重新使能该模块将导致重新初始化，而且将丢失相关状态 （例如， FIFO 状态）。

表 15-10. SCB_UART_CTRL 寄存器

位	名称	数值	说明
[25:24]	MODE	00	标准 UART
		01	SmartCard
		10	IrDA
		11	保留
16	LOOP_BACK	环回控制。通过设置该位， SCB UART 发送器能与相应的接收器进行通信。	

表 15-11. SCB_TX_CTRL/SCB_RX_CTRL 寄存器

位	名称	说明
[3:0]	DATA_ WIDTH	‘DATA_WIDTH + 1’ 是指所传输或接收到的数据帧中的位数量。有效范围为 [3，15]。它不包括起始位、停止位和奇偶校验位。
8	MSB_FIRST	1：表示 MSB 优先 0：表示 LSB 优先
9	MEDIAN	仅用于 SCB_RX_CTRL。 它决定是否在输入接口线路上使用数字三抽头中值滤波器。该滤波器会降低对错误的敏感性，但是它要求更高的过采样值。在 UART IrDA 模式中，该值应始终为 ‘1’ 。 1：被使能 0：被禁用

表 15-12. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL 寄存器

位	名称	说明
[3:0]	TRIGGER_LEVEL	触发电平。当发送器 FIFO 所输出的数据小于该字段的值或接收器 FIFO 所接收到的数据大于该字段的值时，在相应的情况下产生一个发送器或接收器的触发事件。
16	CLEAR	将该位置 1 时，发送器或接收器 FIFO 和移位寄存器都被清除 / 无效。
17	FREEZE	将该位置 1 时，硬件对发送器或接收器 FIFO 进行的读 / 写操作不会产生任何影响。冻结位将不会增加 TX 或 RX FIFO 的读 / 写指针。

表 15-13. SCB_CTRL 寄存器

位	名称	数值	说明
[25:24]	MODE	00	I^2C 模式
		01	SPI 模式
		10	UART 模式
		11	保留
31	ENABLED	0	SCB 模块被禁用
		1	SCB 模块被使能

15.4 内部集成电路 （I²C）

本部分介绍了 PSoC4 中 I²C 模块实现的情况。更多有关 I²C 协议规范的信息，请查阅 NXP 网站上的 I²C 总线规范。

15.4.1 特性

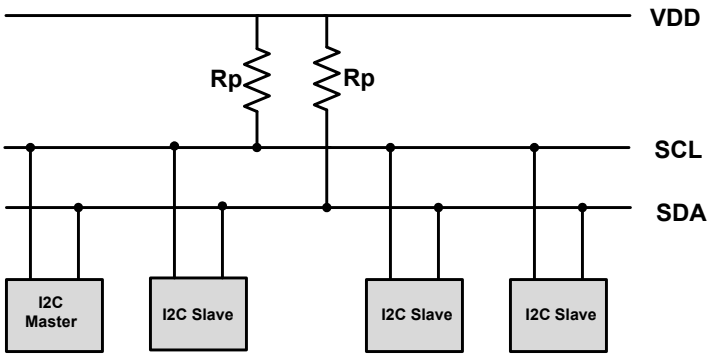
该模块支持以下特性：

- 主设备、从设备和主设备 / 从设备模式
- 慢速模式 （50 kbps）、标准模式 （100 kbps）、快速模式 （400 kbps）和超速模式 （1000 kbps）的数据速率
- 7 位或 10 位从设备寻址 （10 位寻址需要固件支持）
- 时钟延长和冲突检测
- 可编程 I²C 时钟信号 （SCL）的过采样
- 在 I²C 数据信号 （SDA）的输入路径上使用数字中值滤波器来减少错误
- 使用模拟干扰滤波器来实现无干扰的信号传输
- 中断或轮循 CPU 接口

15.4.2 概述

图 15-22 介绍了一个 I²C 通信网络示例。

图 15-22. I²C 接口框图



标准 I²C 总线是一个双线接口，包括：

- 串行数据 （SDA）线
- 串行时钟 （SCL）线

I²C 器件通过集电极开路或带有上拉电阻（Rp）的开漏输出级连接到这两条导线上。各器件间有简单的主设备 / 从设备的关系。可将各主设备和从设备作为发送器或接收器运行。通过使用唯一的 7 位地址可以寻址总线上连接的每一个从设备。 PSoC 4 还为 I²C 的 10 位地址匹配提供了固件支持。

15.4.3 术语和定义

表 15-14 介绍了 I²C 通信网络中常用的术语。

表 15-14. I²C 总线术语定义

术语	说明
发送器	是指将数据发送给总线的器件。
接收器	是指从总线接收数据的器件。
主设备	是指用于启动某个传输操作、生成时钟信号并终止传输的器件。
从设备	是指由主设备寻址的器件。
多个主设备	多个主设备可同时尝试控制总线，但不会破坏信息。
仲裁	该过程指的是如果多个主设备同时尝试控制总线，确保其中只有一个能够实现，并且获得仲裁的信息不被损坏。
同步	是指对两个或更多器件的时钟信号进行同步的过程。

15.4.3.1 时钟延长

从设备未能处理数据时，它可能在 SCL 线上驱动 '0'，以保持其低电平状态。由于实现了 I/O 信号接口，所以 SCL 线的值将为 '0'，该值完全独立于其它主设备或从设备在 SCL 线上驱动的值。该情况被视为时钟延展。另外，这也是唯一一个从设备驱动 SCL 线的情况。主设备监控 SCL 线，并检测 SCL 线上无法生成正向时钟脉冲 （ '1' ）的情况。主设备将在 SCL 线上延迟上升沿的生成，从而有效地与延长时钟的从设备同步。

15.4.3.2 总线仲裁

I²C 协议是一个多主设备、多从设备的接口。通过监控 SDA 线，可在主设备上实现总线仲裁。当主设备发现 SDA 线路的值与它在 SDA 线路上驱动的值不一样时，则检测到了总线冲突。例如，主设备 1 在 SDA 线上驱动数值 '1'，且主设备 2 在 SDA 线上驱动数值 '0'，那么由于执行了 I/O 信号接口，所以实际线路值将为 '0'。主设备 1 用于检测冲突情况，并取消总线控制权。主设备 2 不检测任何冲突，并保持总线控制权。

15.4.4 I²C 工作模式

I²C 是一个同步的单主设备、多主设备、多从设备串行接口。器件可在主设备、从设备或主设备 / 从设备模式下运行。在主设备 / 从设备模式下，寻址器件时，它将从主设备模式切换到从设备模式。在数据传输期间，只有一个主设备有效。有效的主设备将驱动 SCL 线上的时钟。表 15-15 显示的是 I²C 工作模式。

表 15-15. I²C 模式

模式	说明
从设备	仅限从设备操作 （默认）
主设备	仅限主设备操作
多个主设备	允许总线上使用多个主设备
多个主设备 - 从设备	可以同时执行从设备和多个主设备的操作

通过 I²C 总线进行的数据传输要遵循特定的格式。表 15-16 列出了参与 I²C 数据传输的常见总线事件。写传输和读传输两节说明了数据传输中 I²C 总线的位格式。

表 15-16. I²C 总线事件术语

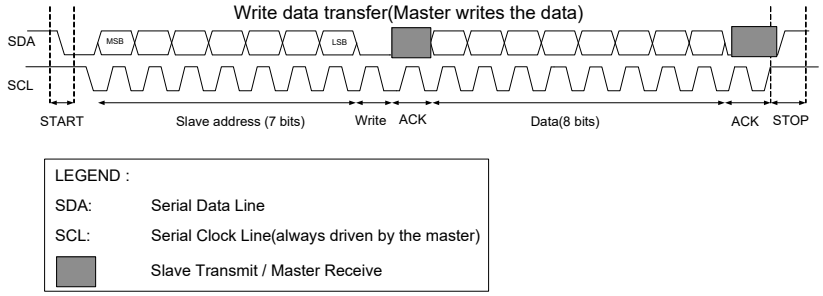
总线事件	说明
START	SCL 为高电平时， SDA 线的电平状态从高转为低。
STOP	SCL 为高电平时， SDA 线的电平状态从低转为高。
ACK	发送器发送每个字节后，接收器将 SDA 线置于低电平，并在时钟脉冲高电平期间保持低电平状态。这样会向发送器通知接收器已经正确接收字节。
NACK	发送器发送每个字节后，接收器并没有将 SDA 线拉低，并在时钟脉冲高电平期间保持它的高电平状态。这样会向发送器指示接收器已经正确接收字节。
重复 START	传输过程结束时，主设备将生成一个 START 事件，而不是 STOP 事件。
DATA	SCL 为低电平时， SDA 状态会发生改变 （数据被更改）； SCL 为高电平时， SDA 状态不会发生任何改变 （数据有效）。

在多个主设备模式下运行时，应始终检查总线是否处于繁忙状态；另一个主设备是否准备好与从设备进行通信。在该情况下，主设备必须等待完成当前的操作才能发出 START 信号（请查阅表 15-16、图 15-23 和图 15-24）。主设备寻找 STOP 信号，该信号表示主设备可以启动其数据传输。

在多主从设备模式下运行时，如果主设备在数据传输期间失去仲裁，则硬件将转为从设备模式，并且它接收到的字节会生成从设备地址中断，以便器件能够准备好响应总线上的其他主设备。对于所有这些模式，有两种传输，即读数据和写数据。在写传输中，主设备将数据发送给从设备；在读传输中，主设备接收来自从设备的数据。请在第 152 页上的 " 主设备模式的传输示例 " 一节、第 154 页上的 " 从设备模式的传输示例 " 一节和第 158 页上的 " 多主设备模式的传输示例 " 一节部分中查看写和读传输的示例。

15.4.4.1 写传输

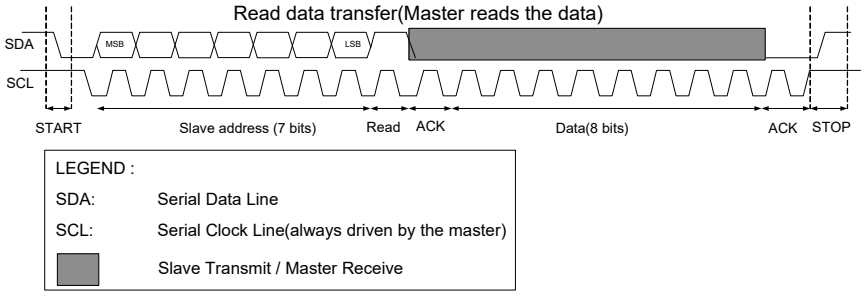
图 15-23. 主设备写数据传输



- 通常主设备在 I²C 总线上生成 START 状态，用于启动写传输。在发生 START 条件后，主设备将写入 7 位 I²C 从设备地址和写指示 （‘0’）。通过在第九位时间段内下拉数据线，被寻址的从设备将发送确认字节。
- 如果从设备地址不与任何从设备匹配或寻址设备不想确认请求，那么它不会通过将 SDA 线置于低电平来发送否定确认信号（NACK）。这样，由于采用了上拉电阻， SDA 线的值将为 ‘1’ 。
- 如果从设备不发送任何确认信号，主设备将通过 STOP 事件结束写传输。主设备也可以生成重复 START 条件，从而重试进行传输。
- 如果主设备接收到确认信号，它会将数据发送给总线。被寻址的从设备将发送一个确认信号来确认接收到被写入的每一个数据字节。收到该确认信号时，主设备将传输其它数据字节。
- 传输完成时，主设备将生成 STOP 条件。

15.4.4.2 读传输

图 15-24. 主设备读数据传输



- 通常，主设备会在 I²C 总线上生成 START 条件，以此启动读传输。在发生 START 条件后，主设备将写入 7 位 I²C 从设备地址和读指示 （‘1’）。通过在第九位时间段内将数据线下拉为低电平，寻址的从设备会传输确认信号。
- 如果从设备地址与连接从设备的地址不匹配或寻址器件不想确认请求，那么不会通过将 SDA 线置于低电平来发送否定确认信号 （NACK）。这样，由于采用了上拉电阻， SDA 线的值将为 ‘1’ 。
- 如果从设备不发送任何确认信号，主设备将通过 STOP 事件结束读传输。主设备也可以生成重复的 START 条件，重试传输。
- 如果从设备确认了地址，它将在确认信号后开始发送数据。主设备发送一个确认信号来确认接收到从设备发送的每个数据字节。收到该确认信号时，寻址的从设备将传输其它数据字节。
- 主设备会将 NACK 信号发送给从设备，以指示从设备停止发送数据字节。这样便完成了读传输操作。
- 传输完成时，主设备将生成一个 STOP 条件。

15.4.5 Easy I2C （EZI2C）协议

Easy I2C （EZI2C）协议是赛普拉斯根据 I²C 协议建立的独特的通信方案。它使用了标准 I²C 协议的软件封装器，并通过索引存储器传输与 I²C 从设备进行通信。这样， CPU 不需要干预每一个帧的操作。

EZI2C 协议定义了一个 8 位地址（8 位宽和 32 入口深度）用于索引存储器阵列，该存储器阵列位于从设备上。EZ 地址的低 5 位用于寻址 32 个入口。通过比较 START 事件中的 EZ 地址和 STOP 事件中的地址，可以计算出发送给 EZI2C 存储器阵列或从 EZI2C 存储器阵列收到的字节数量。

注意：I²C 模块具有一个硬件 FIFO 存储器，该存储器具有 16 位宽度和 16 入口深度，并具有字节写使能性能。EZ 和非 EZ 功能的访问方式有所不同。在非 EZ 模式下， FIFO 被分成 TXFIFO 和 RXFIFO 两类。每个有 8 个宽度为 16 位的地址。而在 EZ 模式下，FIFO 作为单存储器单元（32 个宽度为 8 位的地址）使用。

EZI2C 包括两种传输类型：将主设备中的数据写入到所寻址的从设备存储器空间内，以及主设备从所寻址的从设备存储器空间内读取数据。

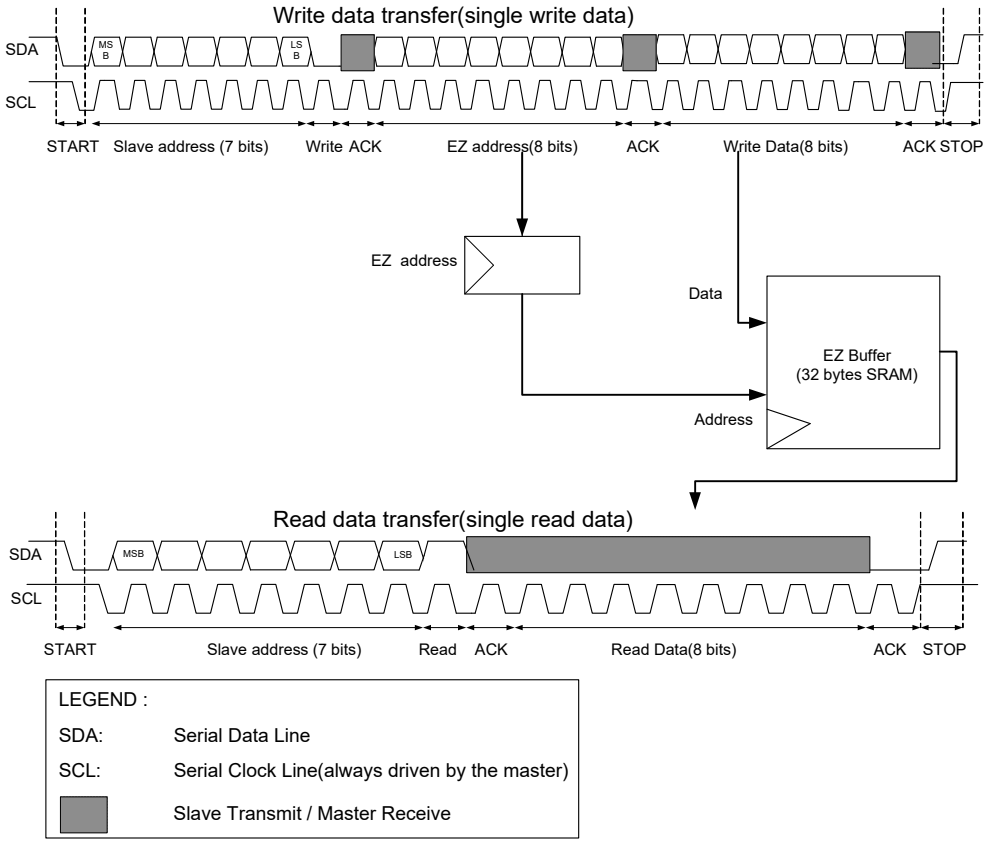
15.4.5.1 存储器阵列的写操作

通过 I²C 写传输对存储器阵列索引进行 EZ 写操作。使用第一个被传输的写数据，以将主设备中的 EZ 地址发送到从设备内。写数据的 5 个最低有效位可作为从设备中 “ 新 ” 的 EZ 地址使用。写传输中的其它写数据元素是写入存储器阵列中的字节。各字节被写入到存储器阵列时，从设备自动递增 EZ 地址。如果写入 EZI2C 缓冲区内的连续数据字节的数量超过了 EZI2C 缓冲区边界，那么后续字节会覆盖最后地址上存在的字节。

15.4.5.2 存储器阵列的读操作

通过 I²C 读数据传输对存储器阵列索引进行 EZ 读操作。 EZ 读操作依赖之前进行的 EZ 写操作 （EZ 写操作设置从设备上

图 15-25. EZI2C 写和读数据传输



15.4.6 I2C 寄存器

通过对表 15-17 中显示的配置、控制和状态寄存器集进行读 / 写来控制 I²C 接口。

表 15-17. I2C 寄存器

寄存器	功能
SCB_CTRL	用于使能 SCB 模块并选择串行接口类型 （SPI、 UART 或 I2C）。还可以用于选择内部 / 外部时钟模式以及 EZ/ 非 EZ 操作模式。
SCB_I2C_CTRL	选择模式 （主设备 / 从设备）并根据接收器 FIFO 状态发送 ACK 或 NACK 信号。
SCB_I2C_STATUS	指示总线繁忙状态检测、主 / 从设备的读 / 写传输状态，并存储 EZ 从设备地址。
SCB_I2C_M_CMD	使主设备生成 START、 STOP 和 ACK/NACK 等信号。
SCB_I2C_S_CMD	使从设备生成 ACK/NACK 信号。
SCB_STATUS	指示外部时钟逻辑是否使用 EZ 存储器。通过该位可以确定对 EZ 存储器进行的访问是否安全。
SCB_I2C_CFG	用于配置滤波器，从而可以去除 SDA 和 SCL 线上的干扰。
SCB_TX_CTRL	指定数据帧的宽度。通过它还可以指定第一个传输位是 MSB （最高有效位）还是 LSB （最低有效位）。
SCB_TX_FIFO_CTRL	用于指定触发电平、发送器 FIFO 和移位寄存器的移除，并执行冻结 （FREEZE）发送器 FIFO。
SCB_TX_FIFO_STATUS	指示存储在发送器 FIFO 中的字节数、硬件读取数据帧的位置（读取指针）、新数据帧被编写的起始位置（写入指针），并决定发送 FIFO 是否保存有效数据。
SCB_TX_FIFO_WR	保存被写入到发送器 FIFO 内的数据帧。该寄存器的功能类似于 PUSH 操作。
SCB_RX_CTRL	它的功能与 SCB_TX_CTRL 寄存器的相同，但它用于接收器 FIFO。此外，它还决定是否在输入接口线路上使用中值滤波器。
SCB_RX_FIFO_CTRL	它的功能与 SCB_TX_FIFO_CTRL 寄存器的相同，但它使用于接收器。
SCB_RX_FIFO_STATUS	它的功能与 SCB_TX_FIFO_STATUS 寄存器相同，但它使用于接收器。
SCB_RX_FIFO_RD	保存从接收器 FIFO 中读取的数据。读取某个数据帧后会从 FIFO 中移除它。它的性能类似于 POP 工作模式中的寄存器。当软件读取数据帧时，该寄存器会引起意外影响，即从 FIFO 中移除数据帧。
SCB_RX_FIFO_RD_SILENT	保存从接收器 FIFO 中读取的数据。读取某个数据帧时并不会删除来自 FIFO 的数据帧。它的性能类似于 PEEK 工作模式中的寄存器。
SCB_RX_MATCH	存储从设备地址，并可以用作从设备的地址屏蔽。
SCB_EZ_DATA	保存 EZ 存储器中的数据。

注意：有关 I²C 寄存器位的详细说明，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术餐口手册 （TRM）*。

15.4.7 I2C 中断

在下面条件下，固定功能 I²C 模块将生成中断。

- I2C 主设备
  - I2C 主设备仲裁失败
  - I2C 主设备收到 NACK
  - I2C 主设备收到 ACK
  - I2C 主设备发送 STOP
  - I2C 总线错误 （检测意外的 stop/start 条件）
- I2C 从设备
  - I2C 从设备仲裁失败
  - I2C 从设备收到 NACK
  - I2C 从设备收到 ACK
  - I2C 从设备收到 STOP
  - I2C 从设备收到 START
  - I2C 从设备地址匹配
  - I2C 总线错误 （检测意外的 stop/start 条件）
- TX
  - TX FIFO 中的输入条目数量小于 SCB_TX_FIFO_CTRL 中 TRIGGER_LEVEL 指定的值
  - TX FIFO 未满
  - TX FIFO 为空
  - TX FIFO 上溢
  - TX FIFO 下溢
- RX
  - RX FIFO 中的输入条目数量小于 SCB_RX_FIFO_CTRL 中 TRIGGER_LEVEL 指定的值
  - RX FIFO 已满
  - RX FIFO 非空
  - RX FIFO 上溢
  - RX FIFO 下溢
- I2C 外部时钟逻辑
  - 地址匹配时发生唤醒请求
  - 在每次传输结束时检测到 I2C STOP 条件
  - 在写传输结束时检测到 I2C STOP 条件
  - 在读传输结束时检测到 I2C STOP 条件

I2C 中断信号固定连接到 Cortex-M0 NVIC，并不能路由到外部引脚。

中断输出是所有可能中断源的逻辑 “ 或 ” （OR）。满足了任何已使能的中断条件时，将触发中断。通过使用中断状态寄存器可以确定实际的中断源。欲了解更多有关中断寄存器的信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器的技术参考手册。

15.4.8 使能和初始化 I2C

下面内容说明了如何配置 I2C 模块的标准 （非 EZ）模式和 EZI2C 模式。

15.4.8.1 将 I2C 接口配置为标准 （非 EZ）模式

必须按下列步骤编程 I2C 接口。

1. 根据表 15-18，使用 SCB_I2C_CTRL 寄存器编程协议特殊信息，包括选择主设备 - 从设备的功能。
2. 使用 SCB_TX_CTRL 和 SCB_RX_CTRL 寄存器编程通用的发送器和接收器信息，如表 15-19 中所示。
   a. 指定数据帧的宽度。
   b. 指定最高有效位作为发送 / 接收的第一位。
3. 分别使用SCB_TX_FIFO_CTRL和SCB_RX_FIFO_CTRL寄存器对发送器和接收器 FIFO 进行编程，如表 15-20 中介绍的内容：
   a. 设置触发电平。
   b. 清除发送器、接收器 FIFO 以及移位寄存器。
4. 通过编程 SCB_CTRL 寄存器使能 I2C 模块，并选择 I2C 模式。这些寄存器位显示在表 15-21 中。欲了解 I2C 寄存器的完整说明内容，请参见 *PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器的技术参考手册*。

表 15-18. SCB_I2C_CTRL 寄存器

位	名称	数值	说明
30	SLAVE_MODE	1	从设备模式
31	MASTER_MODE	1	主设备模式

表 15-19. SCB_TX_CTRL/SCB_RX_CTRL 寄存器

位	名称	说明
[3:0]	DATA_ WIDTH	‘DATA_WIDTH + 1’ 是指传输或接收数据帧中的位数。 I2C 的数值始终为 7。
8	MSB_FIRST	1：表示 MSB 最优先 （对于 I2C，始终优先传输该位）
		0：表示 LSB 最先
9	MEDIAN	仅用于 SCB_RX_CTRL。 它决定是否在输入接口线路上使用数字三抽头中值滤波器。该滤波器会降低对错误的敏感性，但是它要求更高的过采样值。 1 = 使能 0 = 禁用

表 15-20. SCB_TX_FIFO_CTRL/SCB_RX_FIFO_CTRL

位	名称	说明
[3:0]	TRIGGER_LEVEL	触发电平。当发送器 FIFO 所输出的数据小于该字段的值或接收器 FIFO 所接收到的数据大于该字段的值时，则在相应情况下产生一个发送器或接收器的触发事件。
16	CLEAR	该位被设置为 ‘1’ 时，发送器或接收器 FIFO 和移位寄存器均被清除。
17	FREEZE	将该位设置为 ‘1’ 时，硬件对发送器或接收器 FIFO 进行读 / 写操作不会产生任何影响。冻结操作不会增加 TX 或 RX FIFO 的读 / 写指针。

表 15-21. SCB_CTRL 寄存器

位	名称	数值	说明
[25:24]	MODE	00	I2C 模式
		01	SPI 模式
		10	UART 模式
		11	保留
31	ENABLED	0	SCB 模块被禁用
		1	SCB 模块被使能

15.4.8.2 配置 EZI2C 模式

要想将 I2C 模块配置为 EZI2C 模式，请设置下面各个 I2C 寄存器位：

1. 通过将 ‘1’ 写入 SCB_CTRL 寄存器的 EZ_MODE 位 （位 10）内，选择 EZI2C 模式。
2. 执行配置 EZI2C 模式中所描述的第 2 到第 4 步。
3. 设置 SCB_I2C_CTRL 寄存器中的 S_READY_ADDR_ACK （位 12）和 S_READY_DATA_ACK （位 13）。

15.4.9 I2C 中的内部和外部时钟操作

I2C 模块支持内部和外部时钟操作来生成所需的数据速率。内部时钟操作使用来自 PSoC 系统总线时钟的时钟信号。外部时钟操作使用用户所提供的时钟。在深度睡眠功耗模式下 （片上时钟无效），外部时钟的操作仅支持有限的功能。更多有关系统时钟的信息，请参考第 83 页上的时钟系统章节。

外部时钟操作仅适用于下面各场合：

- 从设备功能。
- EZ 功能。

由于 TX 和 RX FIFO 不支持外部时钟操作，因此该操作不适用于非 EZ 功能。

内部和外部时钟操作由 SCB_CTRL 寄存器的两个寄存器字段决定：

- **EC_AM_MODE （使用外部时钟源的地址匹配模式）**：表示 I2C 地址匹配操作使用内部时钟源 （‘0’）还是外部时钟源 （‘1’）。
- **EC_OP_MODE （外部时钟模式）**：表示剩下的协议操作 （除 I2C 地址匹配外）使用内部时钟 （‘0’）还是外部时钟 （‘1’）。如上所述，外部时钟模式不支持非 EZ 功能。

通过下面两个寄存器字段，可以指定 I2C 的功能。需要根据活动、睡眠和深度睡眠系统功耗模式下所需要的行为来设置寄存器字段。设置不当会引起具体功耗模式中的错误行为。表 15-22 和表 15-23 介绍的是 I2C 在 EZ 和非 EZ 模式下的设置。

15.4.9.1 I2C 非 EZ 工作模式

非EZ功能不支持外部定时操作，这是因为该模式并不支持FIFO。因此，在非EZ模式中，EC_OP_MODE应始终设置为‘0’。但仍可以将 EC_AM_MODE 设置为 0 或 1。表 15-22 显示的是各种可能情况的概述。EC_AM_MODE = 0 和 EC_OP_MODE = 1 的组合无效，并且模块不会响应。

EC_AM_MODE 为 ‘0’ 和 EC_OP_MODE 为 ‘0’。

只在活动 / 睡眠系统功耗模式下，该设置才有效。所有 I2C 功能均使用了内部时钟。

EC_AM_MODE 为 ‘1’ 和 EC_OP_MODE 为 ‘0’。

该设置适用于活动、睡眠和深度睡眠系统功耗模式。在活动、睡眠和深度睡眠系统功耗模式下， I2C 地址匹配由外部时钟逻辑实现。外部时钟逻辑进行地址匹配时，它会置位一个唤醒中断源位。通过该位，可以生成用于唤醒 CPU 的中断。

表 15-22. 非 EZ 模式中的 I2C 操作

I2C （非 EZ）模式				
系统功耗模式	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
活动和睡眠	使用内部时钟进行匹配地址。 使用内部时钟执行其它操作。	使用外部时钟进行匹配地址。 使用内部时钟执行其它操作。	不支持	
深度睡眠	不支持	使用外部时钟进行匹配地址。 使用内部时钟执行其它操作。		
休眠	在这些模式下， SCB 不可用 （请参考第 101 页上的功耗模式章节）。			
停止				

- 在活动系统功耗模式下，CPU 有效，但唤醒中断原被禁用（相应掩码位为 ‘0’）。外部时钟逻辑执行地址匹配，则内部时钟逻辑执行余下的 I2C 传输过程。
- 在睡眠模式下，根据应用要求，可以使能或禁用唤醒中断源。其余的操作与活动模式中的操作相同。
- 在深度睡眠模式下，CPU 被关闭；如果使能了唤醒中断源，当发生 I2C 活动时将唤醒 CPU。CPU 唤醒需要占用一段时间，正在进行的 I2C 传输可能被否定确认（NACK）或时钟被延长。在否定确认情况下，内部时钟逻辑将执行唤醒后的第一个 I2C 传输。在时钟延长情况下，内部时钟逻辑执行唤醒时正在进行 / 延长的传输。通过 SCB_I2C_CTRL 寄存器中的 S_NOT_READY_ADDR_NACK （位 14），可以指定外部时钟逻辑执行否定确认 （‘1’）还是时钟延展 （‘0’）。

15.4.9.2 I2C EZ 工作模式

EZ 模式具有三种设置情况。当 EC_OP_MODE 为 ‘0’ 时，EC_AM_MODE 可以设置为 ‘0’ 或 ‘1’ ；当 EC_OP_MODE 为 ‘1’ 时， EC_AM_MODE 必须设置为 ‘1’ 。表 15-23 提供了可能情况的概述。灰色显示的各单元格表示的是可能发生的设

置。但并不推荐这些设置，因为这些设置需要从外部时钟逻辑 （从设备选择）切换为内部时钟逻辑 （其余操作）。 EC_AM_-MODE = 0 和 EC_OP_MODE = 1 的组合无效，并且模块不会响应。

表 15-23. EZ 模式下的 I2C 操作

I2C， EZ 模式				
系统功耗模式	EC_OP_MODE = 0		EC_OP_MODE = 1	
	EC_AM_MODE = 0	EC_AM_MODE = 1	EC_AM_MODE = 0	EC_AM_MODE = 1
活动和睡眠	通过内部时钟进行地址匹配 通过内部时钟运行	通过外部时钟进行地址匹配 使用内部时钟执行其它操作	无效	使用外部时钟匹配地址 使用外部时钟执行其它操作
深度睡眠	不支持	通过外部时钟进行地址匹配 通过内部时钟运行		通过外部时钟进行地址匹配 使用外部时钟运行

- **EC_AM_MODE 为 ‘0’ 和 EC_OP_MODE 为 ‘0’**。只在活动 / 睡眠系统功耗模式下，该设置才有效。
- **EC_AM_MODE 为 ‘1’ 和 EC_OP_MODE 为 ‘0’**。该设置与 I2C 非 EZ 模式相同。
- **EC_AM_MODE 为 ‘1’ 和 EC_OP_MODE 为 ‘1’**。只有在活动和深度睡眠系统功耗模式下，该设置才有效。

I2C 模块的功能通过外部时钟实现。请注意，该设置会导致对模块 SRAM 进行外部时钟的访问。这些访问会与器件的内部时钟访问发生冲突。它会导致等待状态或总线错误。通过 SCB_CTRL 寄存器中的 FIFO_BLOCK 字段 （位 17），可以指定生成等待状态 （ ‘1’ ）还是总线错误 （ ‘0’ ）。

15.4.10 从睡眠模式中唤醒

发生 I2C 地址匹配时，系统从睡眠或深度睡眠模式中唤醒。固定功能 I2C 模块在地址匹配后将执行下面两个操作中的某一个：地址 ACK （确认）或 NACK （否定确认）。

地址 ACK — I2C 从设备执行时钟延长，并等待状态，直到器件唤醒并确认地址为止。

地址 NACK — I2C 从设备立即否定确认 （NACK）地址。器件唤醒时间结束后，主设备必须再次轮询从设备。只有在从设备或多个主设备 - 从设备模式下，该选项才有效。

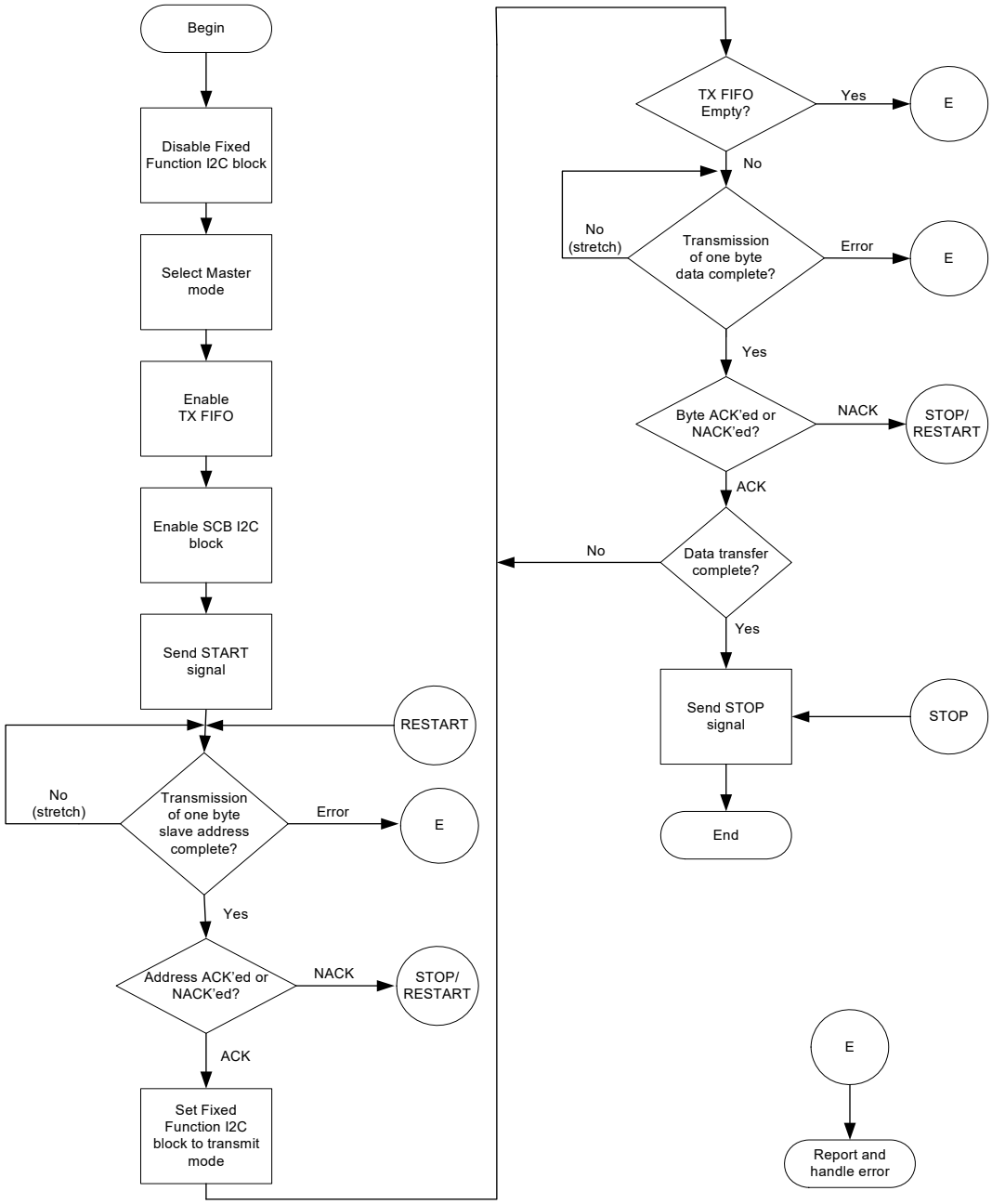
注意：必须使能 SCB_INTR_I2C_EC 寄存器中的中断位 WAKE_UP （位 0），以便在切换为睡眠模式时， I2C 可以通过从设备地址匹配来唤醒器件。

15.4.11 主设备模式的传输示例

在主设备模式下发送或接收数据。

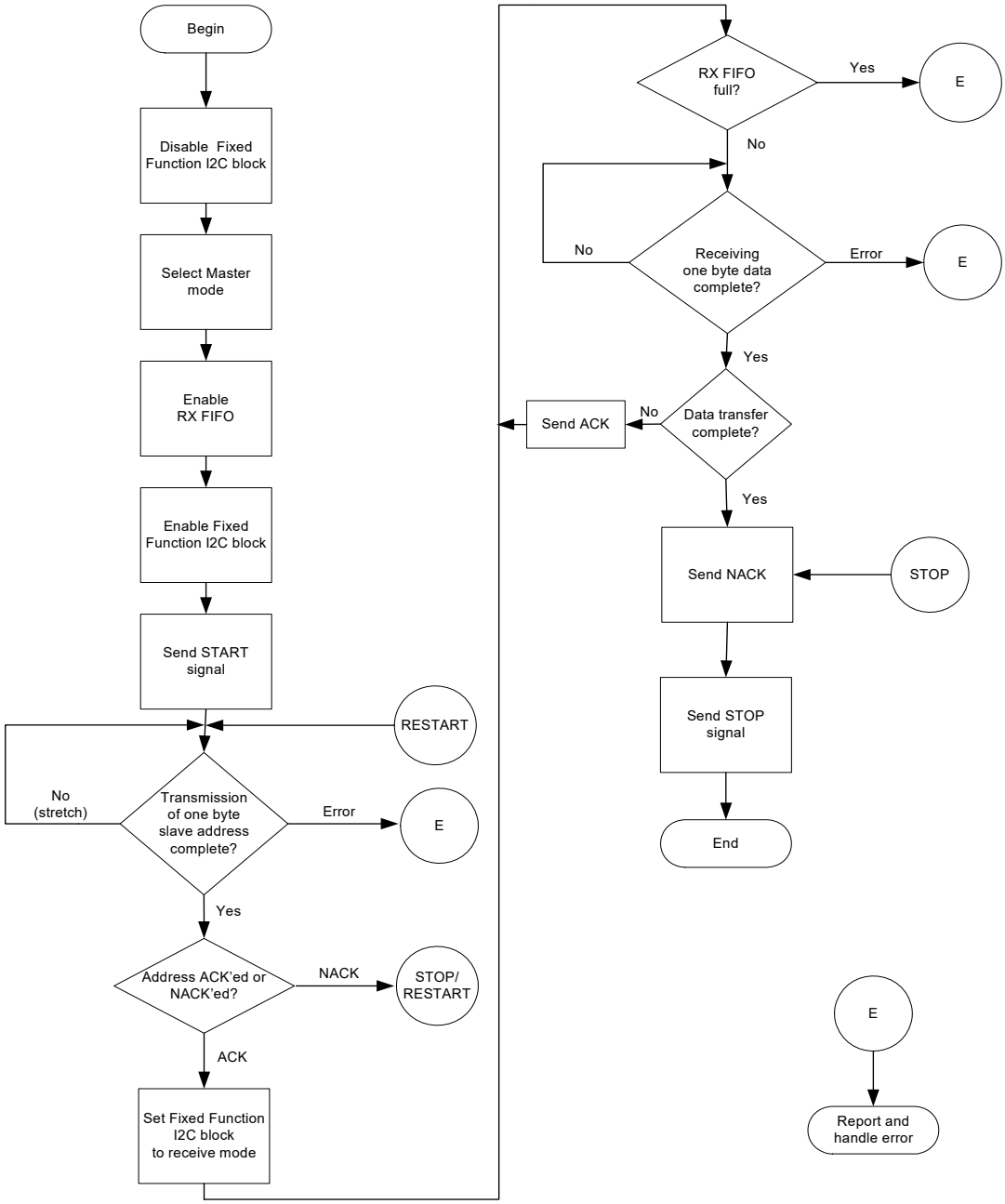
15.4.11.1 主设备发送

图 15-26. 单主设备模式写操作的流程图



15.4.11.2 主设备接收

图 15-27. 单主设备模式读操作的流程图

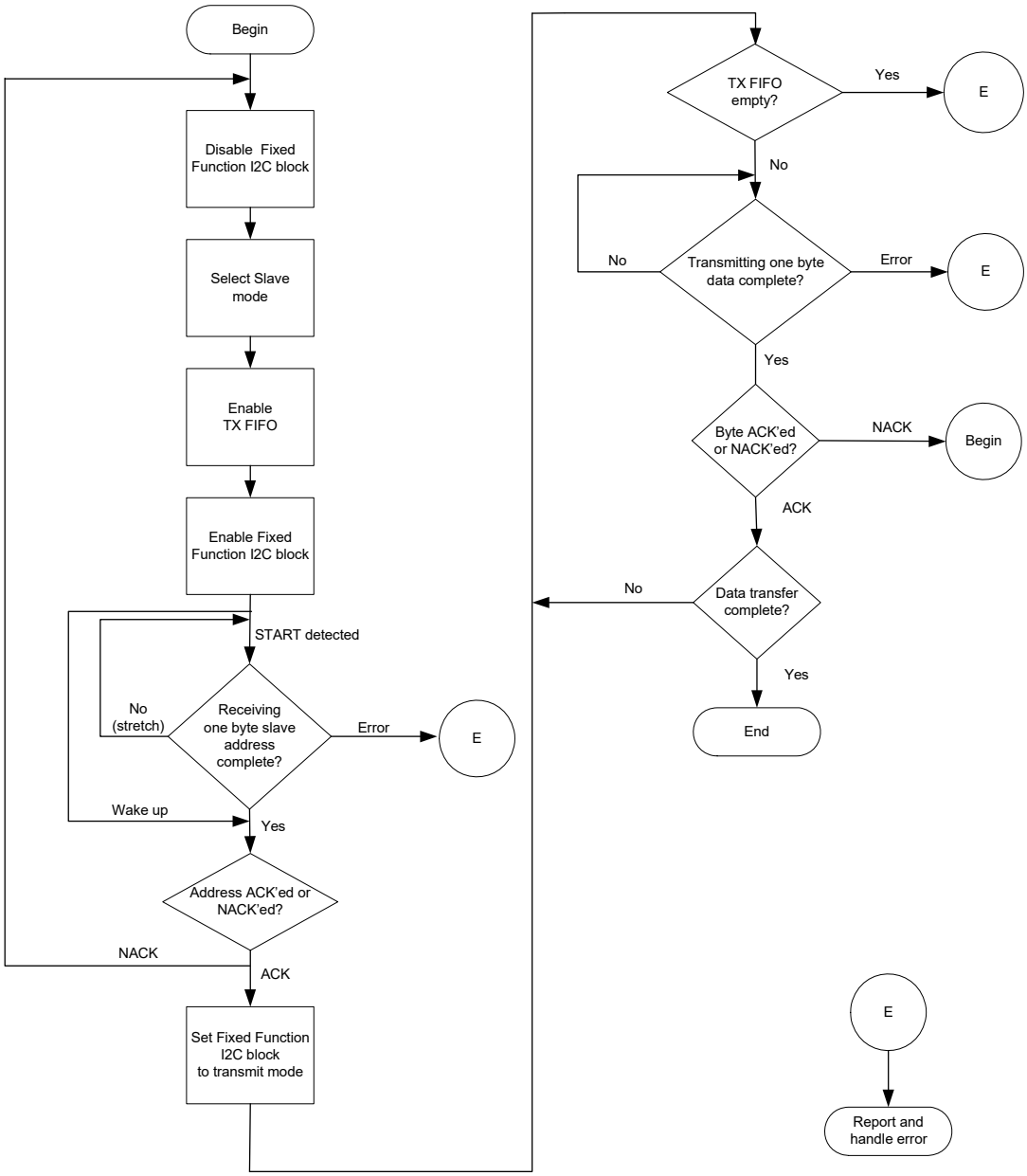


15.4.12 从设备模式的传输示例

在从设备模式下发送或接收数据。

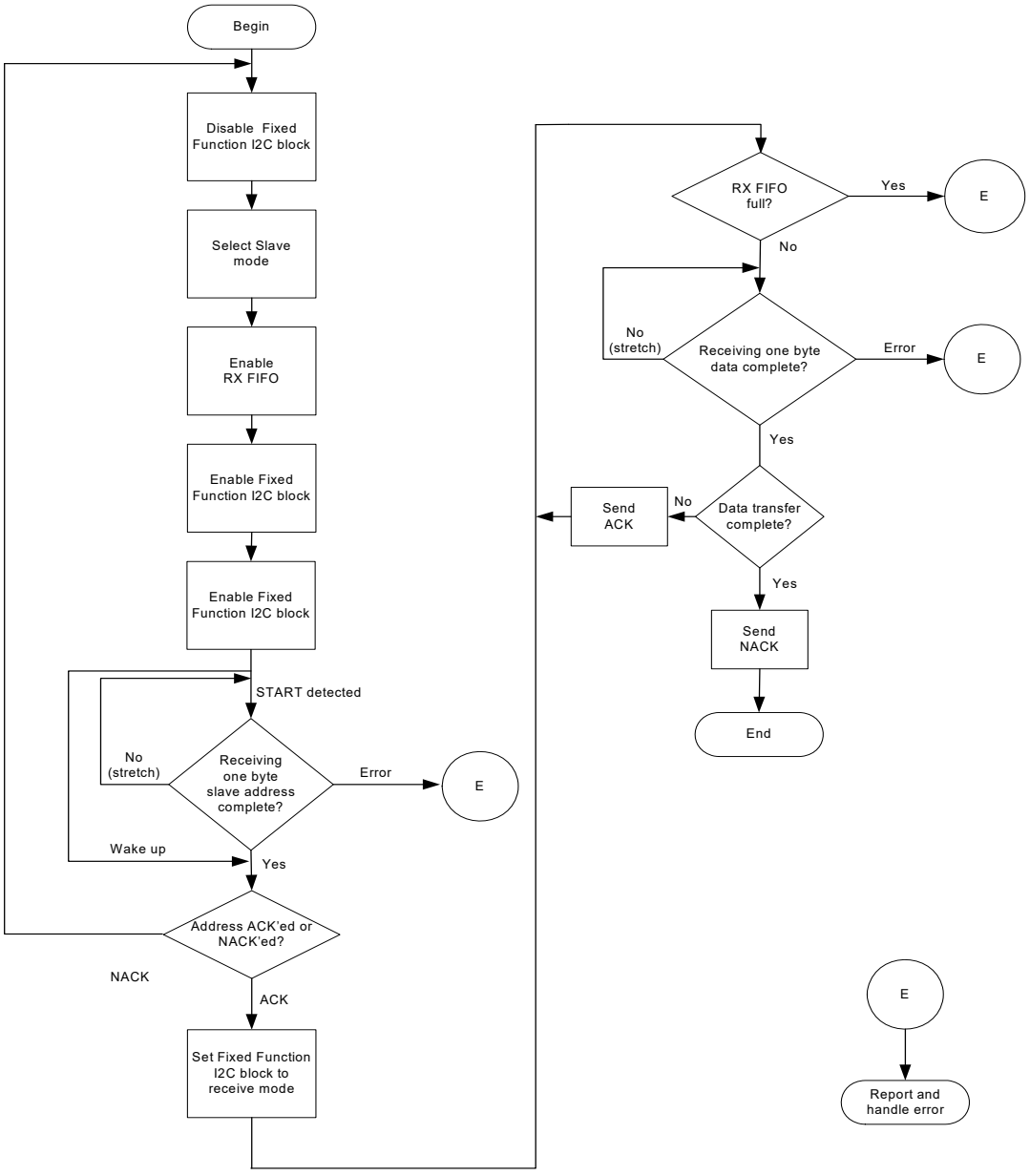
15.4.12.1 从设备发送

图 15-28. 从设备模式写操作的流程图



15.4.12.2 从设备接收

图 15-29. 从设备模式读操作的流程图

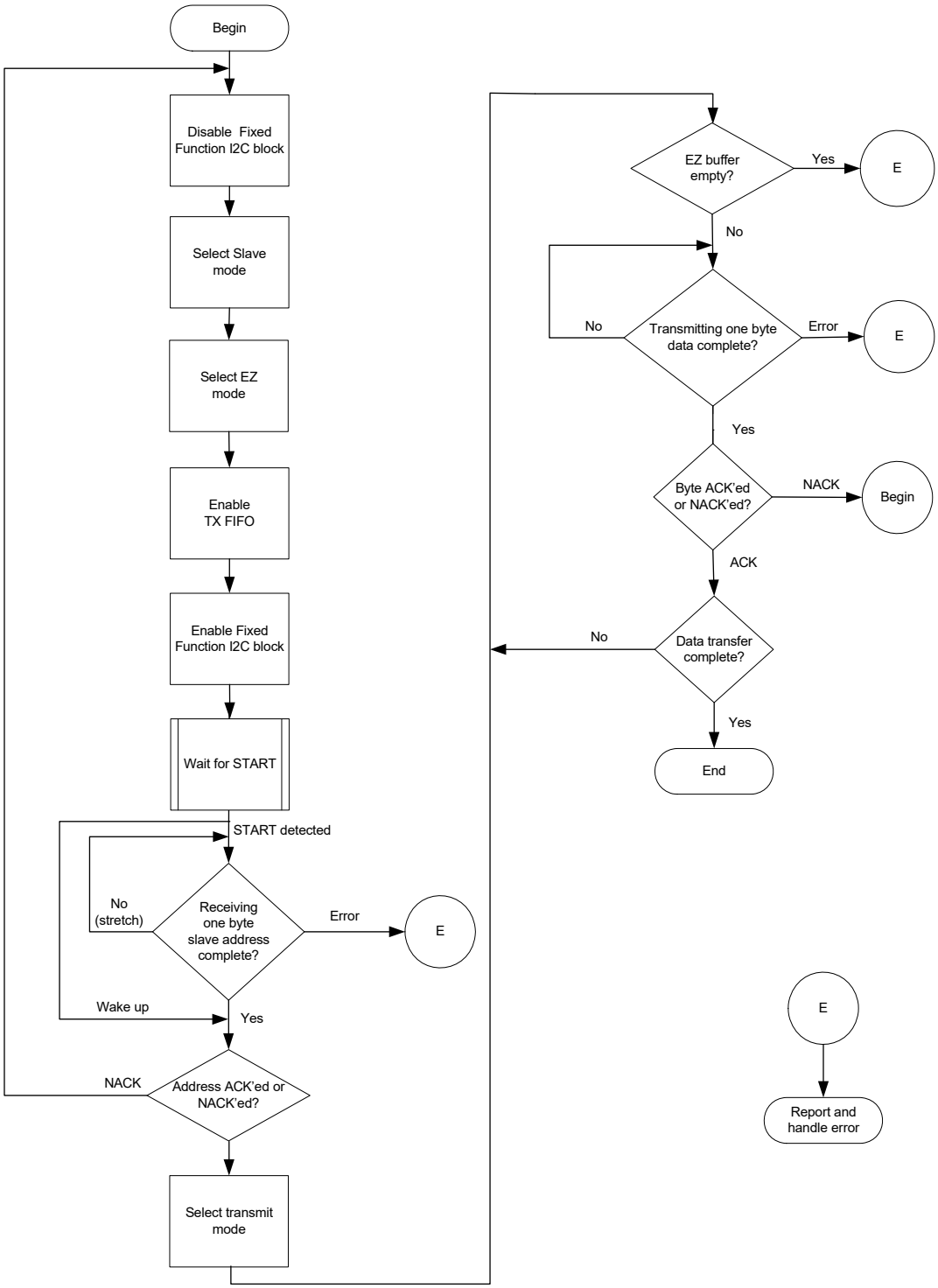


15.4.13 EZ 从设备模式的传输示例

在 EZ 从设备模式下发送或接收数据。

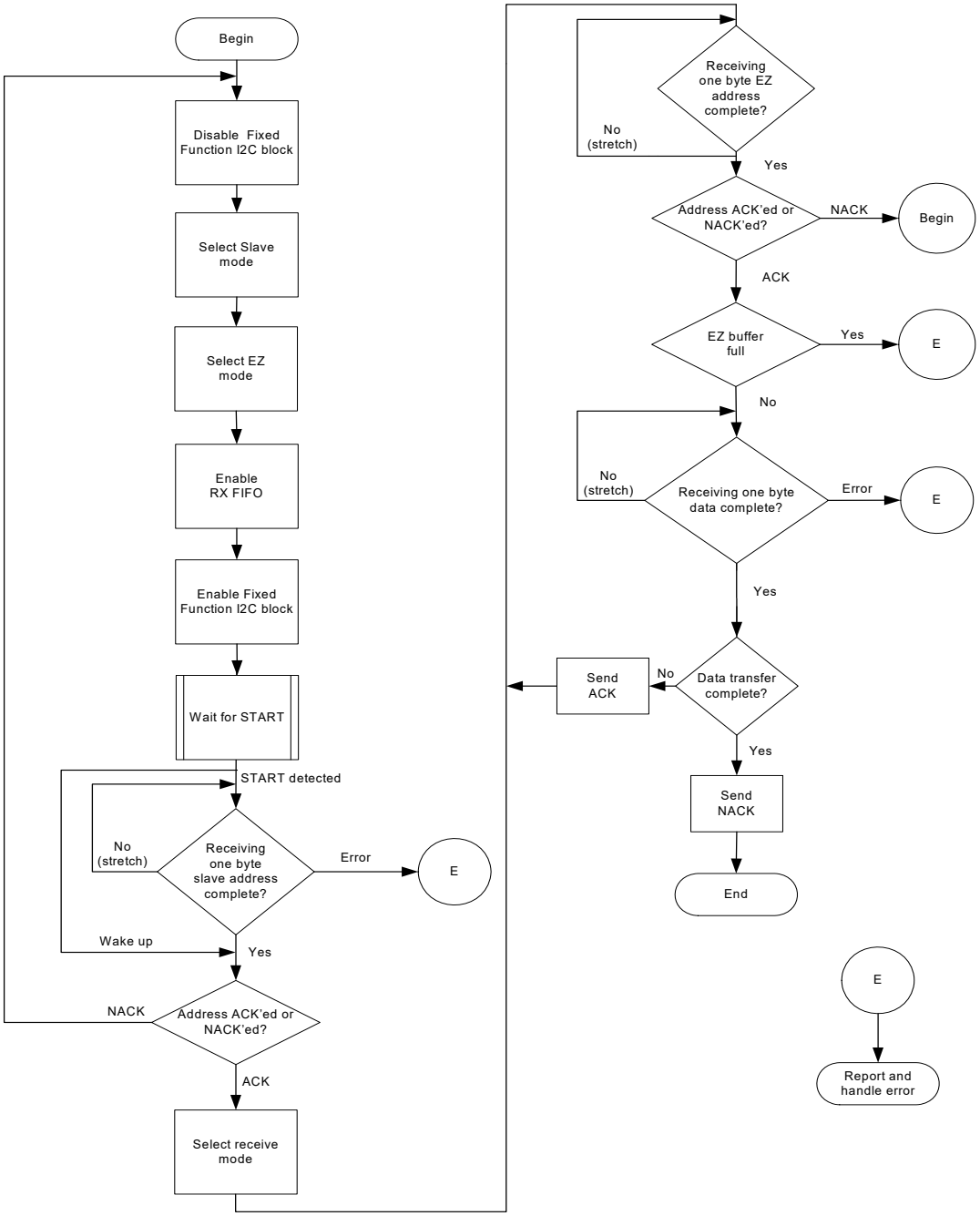
15.4.13.1 EZ 从设备发送

图 15-30. EZI2C 从设备模式写操作的流程图



15.4.13.2 EZ 从设备接收

图 15-31. EZI2C 从设备模式读操作的流程图

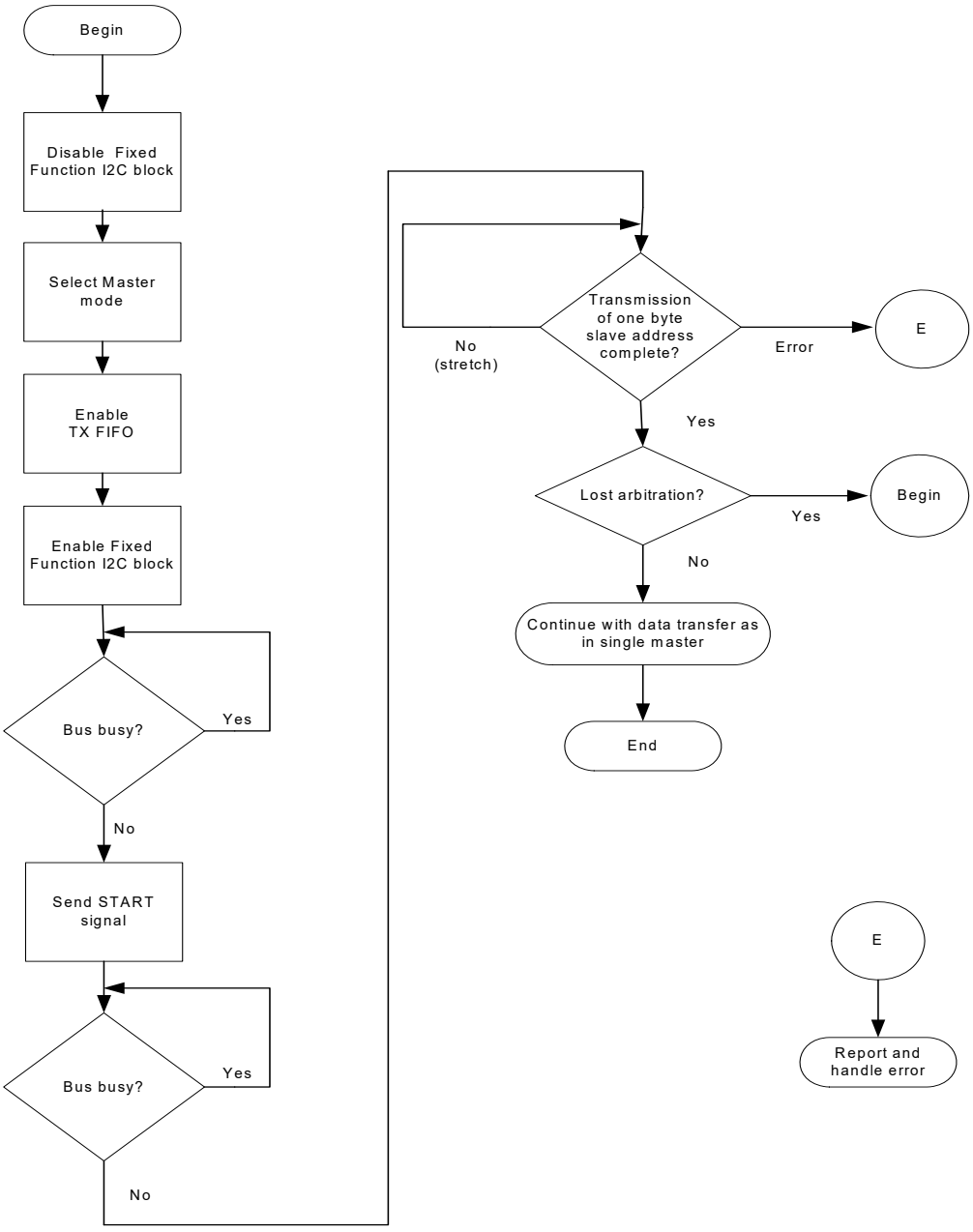


15.4.14 多主设备模式的传输示例

在多主设备模式下，使能或禁用从设备都能够传输数据。

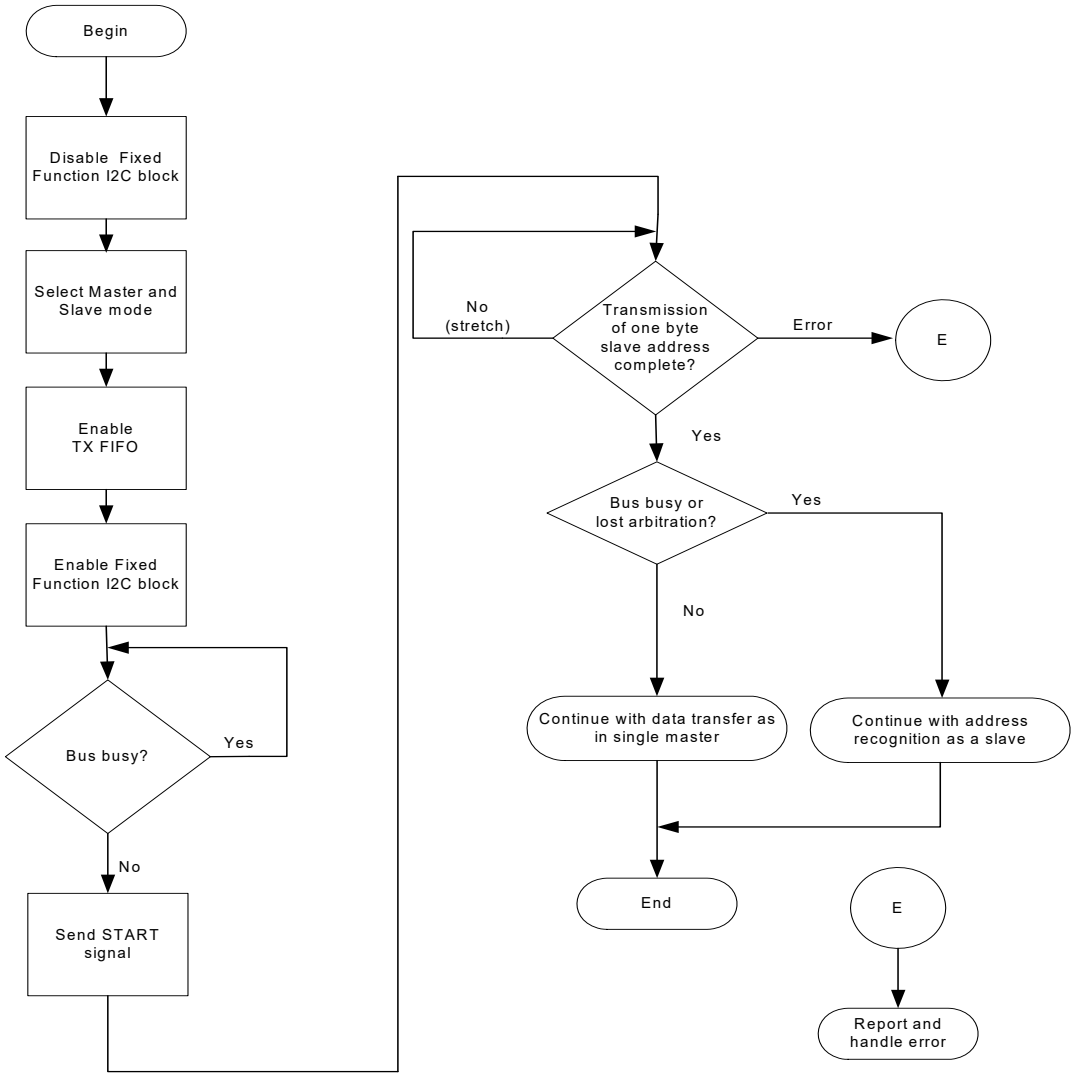
15.4.14.1 多主设备 - 从设备不被使能

图 15-32. 多主设备、从设备不被使能的流程图



15.4.14.2 多主设备 - 从设备被使能

图 15-33. 多主设备、从设备被使能的流程图



16. 通用数字模块 （UDB）

本章节介绍了 PSoC® 4 通用数字模块（UDB）的详细设计信息。UDB 架构在配置精细程度和高效实现两者之间取得了平衡；UDB 包含了可编程逻辑器件 （PLD）、结构逻辑 （数据路径）和灵活布线方案。

注意：某些 PSoC 4 器件系列不支持 UDB。更多详细信息，请参见器件数据手册。

16.1 特性

- PSoC 4 包含四个 UDB
- 为了实现最佳灵活性，每个 UDB 包含以下各组件：
  - 一个基于 ALU 的 8 位数据路径 （DP），包含许多寄存器、 FIFO 和一个 8 字指令存储区
  - 两个 PLD，每个都具有 12 个输入、 8 个乘积项和 4 个宏单元输出
  - 控制和状态模块
  - 时钟和复位模块
- 通过 UDB 阵列可灵活进行路由
- 各个 UDB 的部分可以共用或进行级联，以扩展使用更多功能。
- 可灵活实现多项数字功能，包括：定时器、计数器、 PWM （带死区发生器）、 UART、 SPI 和 CRC 生成 / 检查功能
- 可以通过寄存器与 CPU 进行通信

图 16-1 显示的是构成一个 UDB 的各个组件：两个 PLD 模块、一个数据路径以及控制、状态、时钟和复位模块。

图 16-1. 单个 UDB 的框图

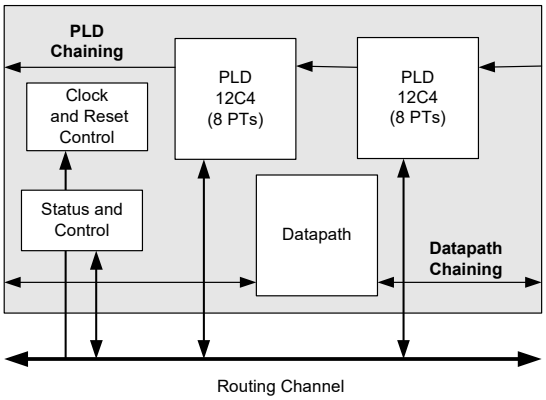
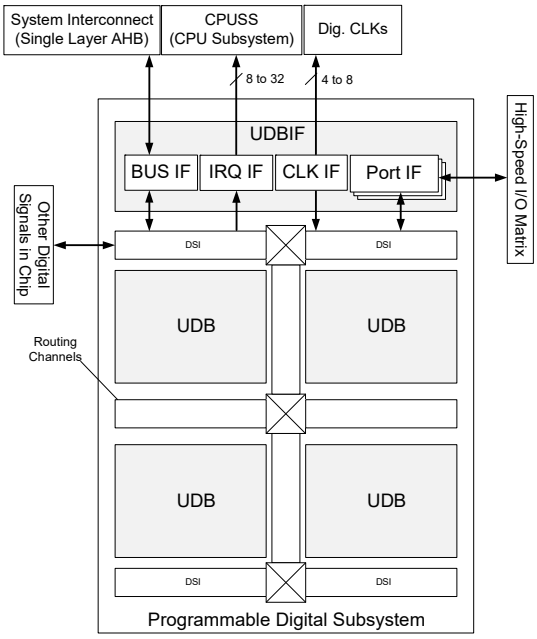


图 16-2 显示了如何将四个 UDB 连接到 PSoC 4 的其余部分。

图 16-2. PSoC 4 中的 UDB 阵列



16.2 工作原理

UDB 的主要组件包括：

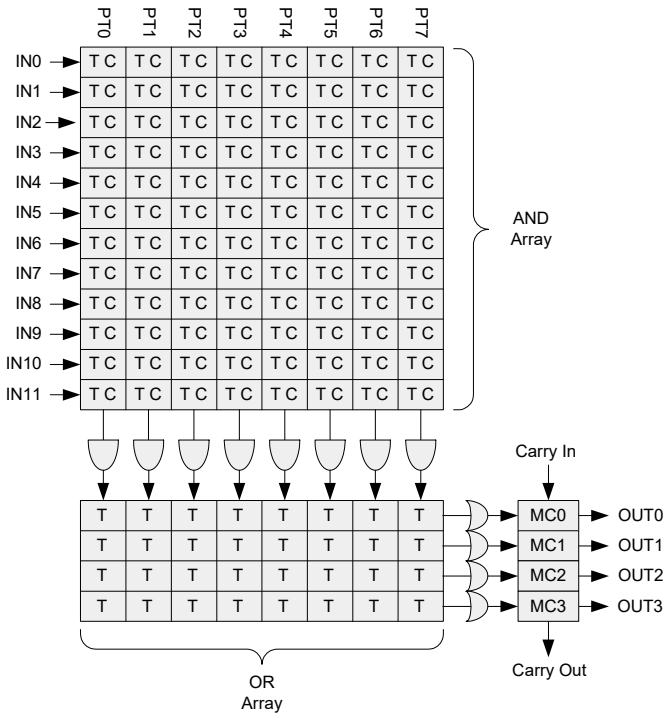
- **PLD（2 个）** — 这些模块从路由通道获取输入，并构成寄存或组合 “ 乘积和 ” 逻辑来实现状态机、数据路径操作的控制、条件输入和驱动输出。
- **数据路径** — 该模块包含一个动态可编程 ALU、四个寄存器、两个 FIFO、比较器和条件生成。
- **控制和状态模块** — 这些模块为CPU固件提供与UDB操作进行交互和同步的方式。
- **复位和时钟控制** — 这些模块为UDB中的其他模块提供了时钟选择和使能，以及复位选择。
- **级联信号** — PLD 和数据路径的级联信号允许链接相邻 UDB，以创建更高精度的功能。
- **路由通道** — 通过可编程开关阵列，将 UDB 连接到路由通道。该通道可以连接 UDB 中的所有模块，还可以连接阵列中的所有其他 UDB。
- **系统总线接口** — 每个UDB中的所有寄存器和RAM都被映射到系统地址空间内，并且 CPU 通过 8 位、 16 位和 32 位访问可以对这些寄存器和 RAM 进行访问。

16.2.1 PLD

每个 UDB 具有两个 “12C4” PLD。显示在图 16-3 内的 PLD 模块可用于实现状态机、执行输入或输出数据调整以及创建查找表 （LUT）。通过配置 PLD，还可以执行算术功能、定序数据路径和生成状态。通用 RTL 可以合成并映射到 PLD 模块。本节对 PLD 设计的概述进行了介绍。

该 PLD 具有 12 个输入，可用于驱动 ***AND*** 阵列中的 8 个乘积项 （PT）。在给定的乘积项中，可以选择输入的 “ 真值 ” （T）或 “ 补码 ” （C）。 PT 的输出就是 OR 阵列的输入。12C4 中的 ‘C’ 表示 OR 项在所有输入中均保持不变，并且每个 OR 输入可以通过编程方式访问任何 PT 或所有 PT。这种结构能够提供最大的灵活性，并确保所有输入和输出都是可交换的。

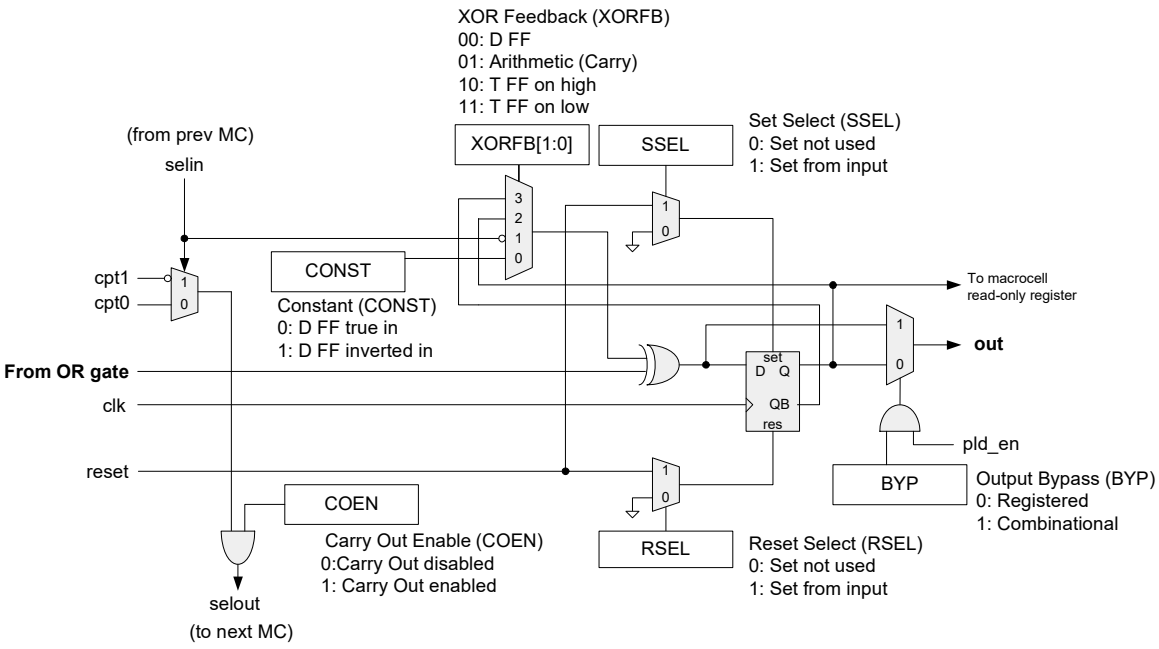
图 16-3. PLD 12C4 结构



16.2.1.1 PLD 宏单元

图 16-4 显示的是宏单元的架构。该输出会驱动路由阵列，并且能够被寄存或组合。寄存模式拥有包含 “ 真 ” 或反相输入的 D 触发器 （DFF）和处理输入高或低的反转触发器 （TFF）。可以设置或复位输出寄存器，以用于初始化，或在运行过程中通过路由信号对该寄存器进行异步设置或复位。

图 16-4. PLD 宏单元架构



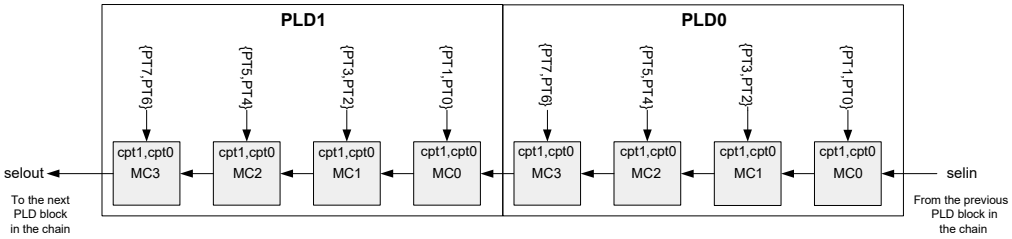
只读的 PLD 宏单元寄存器

CPU 可以将两个 PLD 中 8 个宏单元的输出作为 8 位只读寄存器进行访问。另外，它还可以将许多 UDB 中的宏单元作为 16 位或 32 位只读寄存器进行访问。请参见第 198 页上的 “UDB 寻址 ” 一节的内容。

16.2.1.2 PLD 进位链路

PLD 按照 UDB 地址顺序被级联在一起。如图 16-5 中所示，进位链路输入 “selin” 从链路中前一个 UDB 路由到两个 PLD 中各个宏单元，然后路由到下一个 UDB，成为进位链路输出 “selout”。为了支持算术功能的高效映射，特殊的乘积项被生成。可以在宏单元中同时使用这些乘积项和进位链路。

图 16-5. PLD 进位链路和特殊乘积项输入



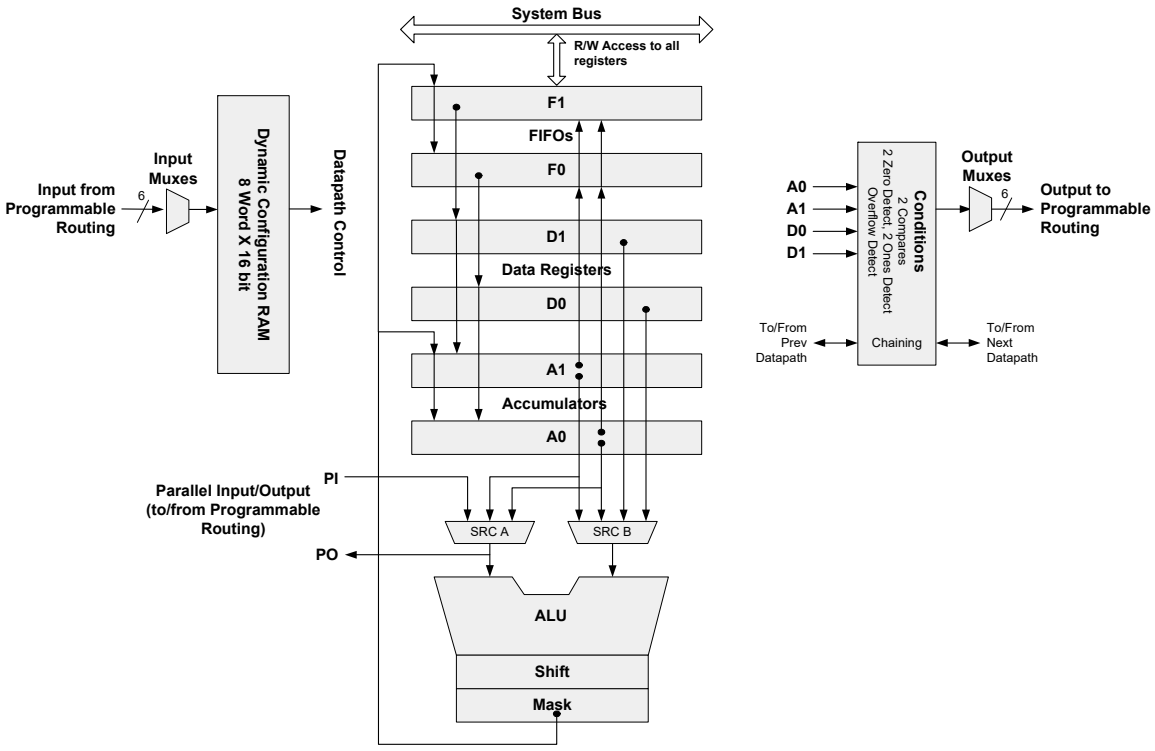
16.2.1.3 PLD 配置

通过访问一组 16 位或 32 位的寄存器，可以对 PLD 进行配置；请参见第 198 页上的 “UDB 寻址 ” 一节中的内容。

16.2.2 数据路径

如图 16-6 所示，数据路径模块包含一个 8 位的单周期 ALU 和相关的比较及条件生成电路。数据路径可被级联到相邻 UDB 中的数据路径，从而得到更高精度的功能。数据路径包含一个小型动态配置 RAM，用于动态选择将在给定周期内执行的操作。对数据路径进行优化，以实现典型的嵌入式功能，如定时器、计数器、 PWM、 PRS、 CRC、移位器和死区生成器。加法和减法功能支持数字 delta-sigma 操作。

图 16-6. 数据路径的顶层架构



16.2.2.1 概述

下面显示的是数据路径的主要特性：

动态配置

动态配置是指在序列发生器的控制下，按周期更改数据路径功能和内部连接的能力。该操作通过使用配置 RAM（它包含了 8 种独特的配置）实现。该 RAM 的地址输入可以来自与路由结构相连的任意模块 （通常是 PLD 逻辑、 I/O 引脚），也可以来自其他数据路径。

ALU

ALU 可以执行 8 个通用的功能：递增、递减、加法、减法、AND、OR、XOR 和 PASS。功能选择操作是由所配置的 RAM 按周期控制的。可以在 ALU 的输出上使用独立移位 （左移、右移、半字节交换）和掩码操作。

条件

每个数据路径都有两个带位掩码选项的比较器。通过配置这些比较器可以选择用于比较的多个数据路径寄存器输入。其他可检测的条件包括全零、全一和溢出检测。这些条件允许将主要的数据路径输出选择作为其他功能的输入路由到数字布线结构。

内置 CRC/PRS

数据路径包含对以下内容的内置支持：任意宽度和任意多项式的单周期循环冗余校验 （CRC）计算和伪随机序列 （PRS）生成。为得到超过 8 位的 CRC/PRS 宽度，可以在各数据路径之间链接信号。该特性可以通过动态方式控制，因此可以与其他功能交替进行。

可变 MSB

算术和移位功能的最高有效位可通过编程方式指定。可变 MSB 支持可变宽度 CRC/PRS 功能，而且通过与 ALU 输出掩码相结合，可实现任意宽度的定时器、计数器和移位模块。

输入 / 输出 FIFO

每个数据路径都包含两个 4 字节的 FIFO，这些 FIFO 可单独被配置为输入缓冲区 （CPU 写入到 FIFO，数据路径内部读取 FIFO）或输出缓冲区 （数据路径内部写入到 FIFO， CPU 读取 FIFO）。这些 FIFO 能够生成已满或空白的状态信号，可以路由这些信号，以便与序列发生器或中断进行交互。

链路

可配置数据路径，以便将条件和信号同相邻的数据路径连接起来。可以链接移位、进位、捕获和其他条件信号，以实现精度更高的算术、移位和 CRC/PRS 功能。

时间复用

在过采样或不需要最高时钟频率的应用中，数据路径中的单个 ALU 模块可以有效地被两组寄存器和条件生成器共用。ALU 和移位输出会被寄存起来，并可在后续周期中作为输入使用。使用示例支持在一个 （8 位）数据路径中实现 16 位功能，并支持在 CRC 生成操作和数据移位操作之间进行切换。

数据路径输入

该数据路径具有三个输入类型：配置、控制以及串行和并行数据。通过配置输入可以选择动态配置 RAM 地址。控制输入从 FIFO 加载数据寄存器，***累加器***并将输出捕获到 FIFO 内。串行数据输入包括 shift in 和 carry in。并行数据输入端口能够在路由过程中引入多达 8 位的数据。

数据路径输出

该数据路径可以生成多达 16 个信号。其中，一些信号为条件信号（例如，比较信号）、一些为状态信号（例如，FIFO 状态），剩下的信号为数据信号 （例如， shift out 信号）。这些信号被复用成 6 个数据路径输出，然后被驱动到路由矩阵内。默认情况下，各输出是单同步 （即管道式）的。这些输出还有一个组合输出选项。

数据路径的工作寄存器

每个数据路径模块有 6 个 8 位工作寄存器。CPU 可以对这些寄存器进行读 / 写操作：

表 16-1. 数据路径的工作寄存器

类型	名称	说明
累加器	A0， A1	累加器可以作为 ALU 的操作数，也可以作为结果。数据寄存器或 FIFO 的数据可被载入这些累加器内。累加器通常包含了某个功能 （如：计数、 CRC 或移位）的当前值。这些寄存器都是非保留寄存器；在睡眠模式下，这些寄存器的值将被丢失，在唤醒模式下则被复位为 0x00。
数据	D0， D1	数据寄存器通常包含某个功能的常数数据，如：PWM 比较值、定时器或 CRC 多项式。这些寄存器在睡眠间隔中会保留它们的值。
FIFO	F0， F1	两个 4 字节 FIFO 为缓冲的数据提供一个数据源和一个结果。可以将这两个 FIFO 都配置为输入缓冲区、输出缓冲区，或配置为一个输入缓冲区和一个输出缓冲区。状态信号显示这些寄存器的满 / 空状态。使用示例包括 SPI 或 UART 中的缓冲 TX 和 RX 数据，以及缓冲 PWM 比较和缓冲定时器周期数据。这些寄存器都是非保留寄存器；在睡眠模式下，这些寄存器的值将被丢失，在唤醒模式下则被复位为 0x00。

16.2.2.2 数据路径 FIFO

FIFO 模式和配置

每个 FIFO 具有多种操作模式和配置。

表 16-2. FIFO 模式和配置

模式	说明
输入 / 输出	在输入模式下， CPU 将数据写入到 FIFO 中，数据路径内部读取并使用该数据。在输出模式下，数据路径内部会将数据写入到 FIFO 内，然后 CPU 将读取并使用该数据。
单字节缓冲区	FIFO 作为无状态的单字节缓冲区运行。写入 FIFO 内的数据可以立即被读取，并且可以随时被覆盖。
电平 / 边沿	从数据路径内部加载 FIFO 可以通过电平或边沿触发实现。
正常 / 快速	从数据路径源加载 FIFO 的操作是在当前选定的数据路径时钟 （正常）或 HFCLK （快速）上进行采样的。这样能按系统的最高频率 （HFCLK）进行捕获，独立于数据路径时钟。
软件捕获	当使能该模式而且 FIFO 处于输出模式时， CPU 读取相应累加器 （F0 的 A0、 F1 的 A1），累加器值将被同步传输到 FIFO 内。然后可以立即从 FIFO 中读取该捕获值。如果链接功能被使能，通过数据路径进行原子读指令，多字节的值会跟着链路反馈到 MS 模块。
异步	当数据路径时钟与 HFCLK 异步时，FIFO 状态信号可以直接被路由到数据路径中其余部分，或与数据路径时钟做单次采样，或与异步数据路径时钟做两次采样后输出。
独立时钟极性	每个 FIFO 都有一个控制位，用于反转 FIFO 时钟信号的极性以将其跟随数据路径时钟信号。

图 16-7 显示的是输入 / 输出模式控制的各种可能的 FIFO 配置。 TX/RX 模式具有两个 FIFO，一个处于输入模式，另一个处于输出模式。 SPI 是该配置的主要示例。双捕获配置提供了 A0 和 A1 的独立捕获，或 A0/A1 的两个单独控制捕获。最后，双缓冲模式可以提供被缓冲的周期和比较，或两个独立的周期 / 比较。

图 16-7. FIFO 配置

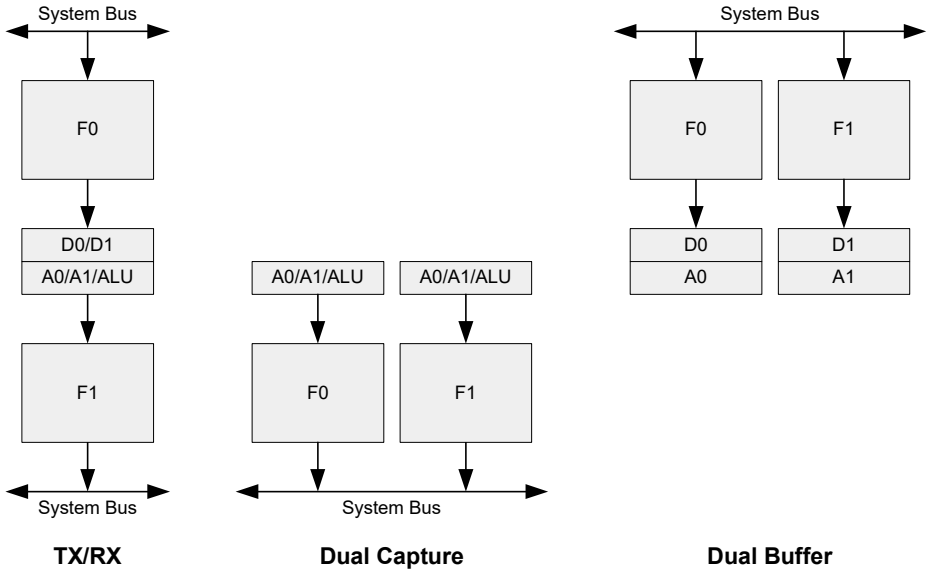
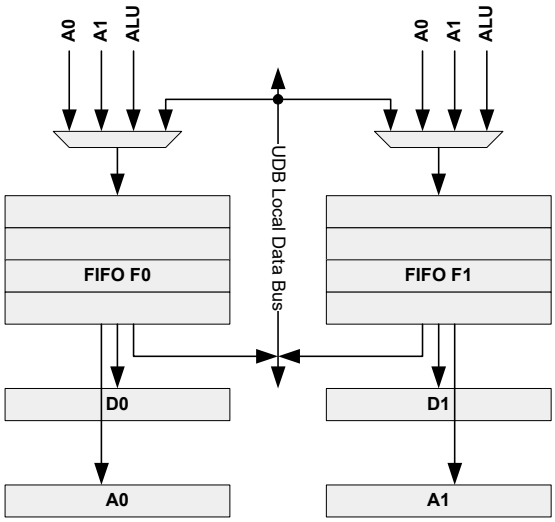


图 16-8 显示的是 FIFO 数据源和数据流的详细概况。

图 16-8. FIFO 数据源和数据流



FIFO 处于输入模式时，数据源为系统总线，数据流为 Dx 和 Ax 寄存器。而 FIFO 处于输出模式时，数据源为 Ax 寄存器和 ALU，数据流为系统总线。该复用器的选择是在 UDB 配置寄存器 CFG15 中静态设置的，用于 F0_INSEL[1:0] 或 F1_INSEL[1:0]，如表 16-3 中所示。

表 16-3. UDB CFG15 寄存器中的 FIFO 多路复用器设置

Fx_INSEL[1:0]	说明
00	**输入模式** — 系统总线对 FIFO 进行写操作， FIFO 输出目标为 Ax 或 Dx。
01	**输出 A0 模式** — FIFO 输入源为 A0， FIFO 输出目标为系统总线。
10	**输出 A1 模式** — FIFO 输入源为 A1， FIFO 输出目标为系统总线。
11	**输出 ALU 模式** — FIFO 输入源为 ALU 输出， FIFO 输出目标为系统总线。

FIFO 状态

每个 FIFO 会生成两个状态信号：“ 总线 ” 和 “ 模块 ”。这两个信号通过数据路径输出复用器被传送到 UDB 路由。可以使用 “ 总线 ” 状态来激活某个中断请求，以对 FIFO 进行读 / 写操作。“ 模块 ” 状态主要用于为 UDB 内部提供 FIFO 状态。状态位的含义取决于配置方向 （UDB CFG15 寄存器中 Fx_INSEL[1:0]）和 FIFO 电平位。 FIFO 电平位 （Fx_LVL）在工作寄存器空间中的辅助控制工作控制寄存器 （ACTL）内设置。表 16-4 显示了各个选项。

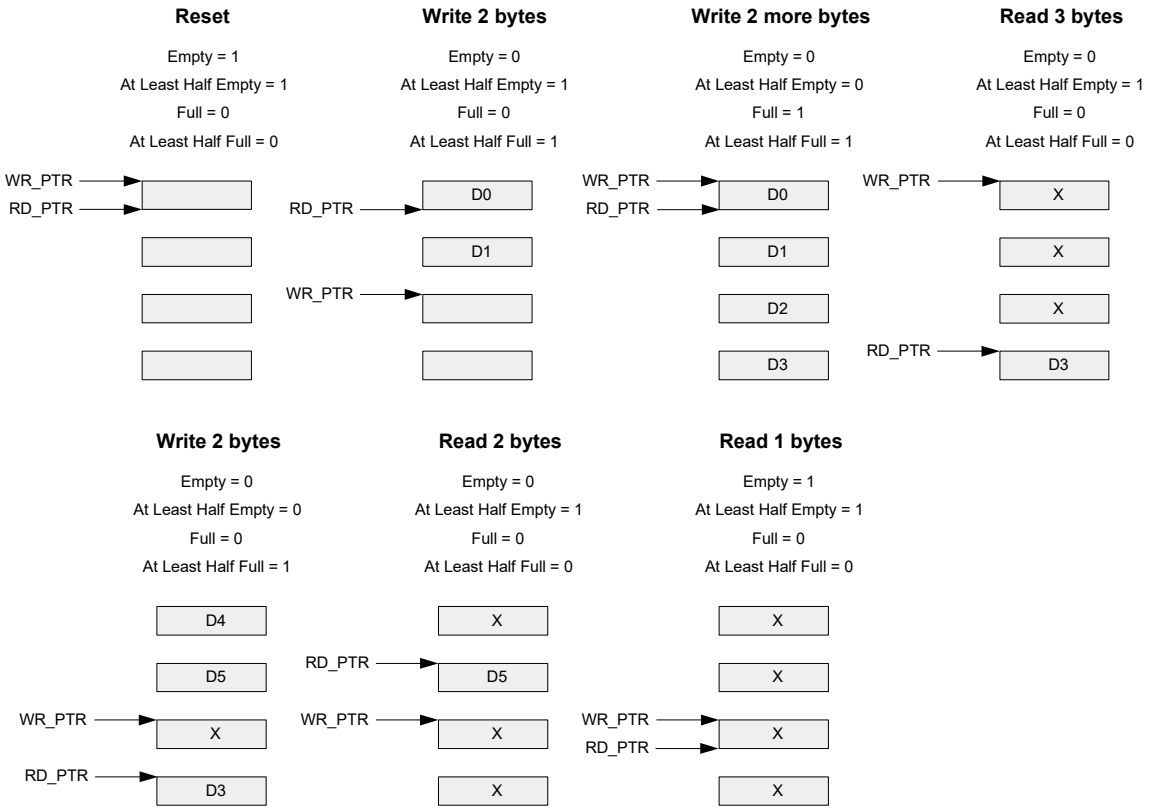
表 16-4. FIFO 状态选项

Fx_INSEL[1:0]	Fx_LVL	FIFO 状态	FIFO 状态信号	说明
输入	0	未满	总线状态	只有 FIFO 中至少有一个字节的空间时，输入才被激活。
输入	1	至少一半为空	总线状态	FIFO 中至少存在两个字节的空间时，输入才被激活。
输入	NA	空白	模块状态	FIFO 中没有字节 （FIFO 为空）时，输入才被激活。 FIFO 非空时，数据路径内部可能使用这些字节。FIFO 为空时，数据路径会处于闲置状态或生成一个欠载条件。
输出	0	非空	总线状态	FIFO 中至少存在一个字节可读时，输出才被激活。
输出	1	至少一半为空	总线状态	FIFO 中至少存在两个字节可读时，输出才被激活。
输出	NA	已满	模块状态	FIFO 已满时，输出才被激活。 FIFO 未满时，数据路径内部会将字节写到 FIFO 内。FIFO 已满时，数据路径会处于闲置状态或生成一个过载条件信号。

FIFO 操作

图 16-9 显示的是典型的读写序列和相关的状态生成。虽然该图指出读和写操作在不同的时间内发生，但读写操作还可以同时发生。

图 16-9. 详细的 FIFO 操作数据流

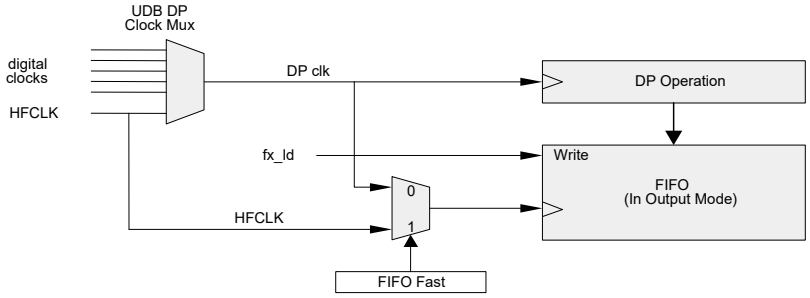


FIFO 快速模式 （FIFO FAST）

当将 FIFO 配置为输出时，FIFO 加载操作通常使用当前选择的数据路径时钟来采样写信号。如图 16-10 中所示，设置 FIFO 快速模式后，可以选择 HFCLK 用于该操作。与边沿敏感模式一起使用时，可以使累加器到 FIFO 的传输延迟从数据路径时钟分辨率下降为 HFCLK 分辨率 （该延迟可以为更高的值）。这样可使 CPU 读取 FIFO 中最小延迟的捕获结果。

图 16-10 指出了快速加载操作独立于当前所选择的数据路径时钟；但使用 HFCLK，功耗会更大。请注意， fx_ld 输入信号必须满足 HFCLK 时序要求，这可能需要进行局部的重新同步。

图 16-10. FIFO 快速配置数据流



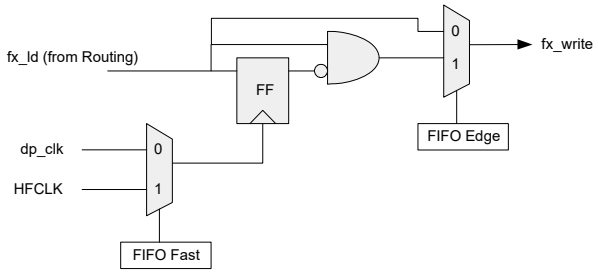
FIFO 电平 / 边沿写模式

在这两种模式下可以通过数据路径进行写 FIFO。在第一种模式下，各个累加器的数据被同步传输给 FIFO。该写操作的控制信号（fx_ld）通常由状态机或与数据路径时钟同步的条件生成。可以在输入加载控制为 ‘1’ 的任何周期内对 FIFO 执行写操作。

在第二种模式下， fx_ld 信号的上升沿出现时，通过 FIFO 可以捕获累加器的值。在该模式下，波形的占空比是任意的 （然而，它的宽度至少必须为一个数据路径时钟周期）。例如，通过将外部引脚输入作为触发器使用来捕获累加器的值。该模式的限制是在检测到另一个上升沿前，输入控制必须至少在一个周期内恢复为 ‘0’。

图 16-11 显示了 fx_ld 控制输入上的边沿检测选项。该选项的 1 位在 UDB 中为两个 FIFO 设置模式。请注意，边沿检测在已选定的 FIFO 时钟频率上被采样。

图 16-11. 内部 FIFO 写操作的边沿检测选项



FIFO 软件捕获模式

一个常见且重要的要求是：在普通操作期间允许 CPU 读取累加器中的内容。该操作是通过软件捕获实现的，并且通过设置 FIFO 捕获配置位（UDB CFG16 寄存器中的 FIFO_CAP 位）被触发。该位适用于 UDB 中的两个 FIFO，但仅在 FIFO 处于输出模式时，才能进行操作。使用软件捕获时，需要将 F0 设置为从 A0 加载，并将 F1 设置为从 A1 加载。

如图 16-12 中所示，读取累加器将触发该累加器对 FIFO 进行的写操作。该信号被链接，以便读取已给字节时同时会捕获所有链接 UDB 中的累加器。这样允许 CPU 可以同时读取 16 位或更多位。读取累加器后返回的数据将被忽略； FIFO 中的捕获值可以立即被读取。

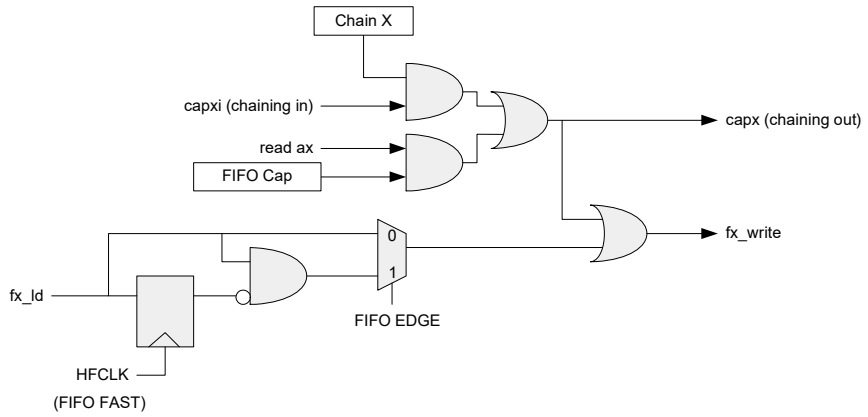
对生成 FIFO 加载的 fx_ld 信号与软件捕获信号进行 “OR” （或）运算；同时使用硬件和软件捕获时，则不可预测这些结果。据一般规则，这些功能是相互排斥的；然而，通过进行下列设置，可以同时使用硬件和软件捕获：

- 将 FIFO 捕获时钟模式设置为 FIFO FAST
- 将 FIFO 写入模式设置为 FIFO EDGE

进行这些设置后，硬件和软件捕获以相同的方式运行，在已给的 HFCLK 周期内，确认某个信号也会启动一个捕获。

在启动一个软件捕获前，建议先清除固件中的目标 FIFO（UDB ACTL 寄存器）。这样会将 FIFO 读和写指针初始化为已知状态。

图 16-12. 软件捕获配置



FIFO 控制位

在进行正常操作时， CPU 固件通过使用辅助控制寄存器 （ACTL）中的 4 位来控制 FIFO。

FIFO0 CLR 和 FIFO1 CLR 位用于复位或清理 FIFO。向这些位中的某一位写入 ‘1’ 后，可复位相应的 FIFO。为了继续进行 FIFO 操作，必须将该位回写成 ‘0’ 。如果该位保持激活状态，给定的 FIFO 将被禁用并作为无状态的一字节缓冲区运行。可以将数据写入到 FIFO 内；可以立即读取该数据并且可以随时覆盖掉该数据。使用 Fx INSEL[1:0]（UDB CFG15 寄存器）配置位的数据方向仍处于有效状态。

FIFO0 LVL 和 FIFO1 LVL 位控制了 4 字节 FIFO 确认总线状态处于被确认的等级（总线可以对 FIFO 进行读或写操作）。FIFO 总线状态的含义取决于所配置的方向，如表 16-5 中所示。

表 16-5. UDB ACTL 寄存器中 FIFO 等级的控制位

FIFOxLVL	输入模式 （总线正在写入 FIFO）	输出模式 （总线正在读取 FIFO）
0	未满 至少可以写入一个字节	非空 至少可以读取一个字节
1	至少一半为空 至少可以写入两个字节	至少一半为满 至少可以读取两个字节

FIFO 异步操作

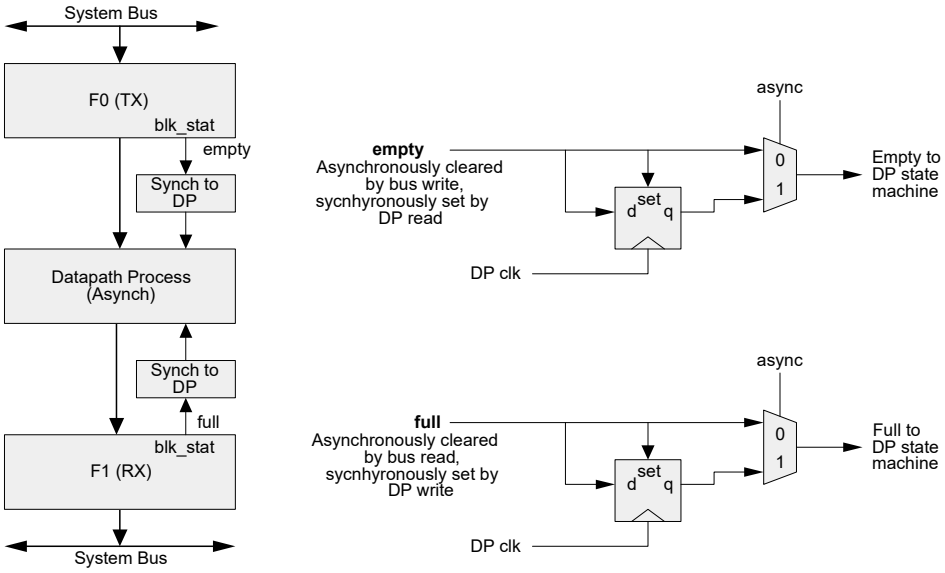
图 16-13 显示的是 FIFO 异步操作的概念。例如，假设 F0 被设置为输入模式和 F1 被设置为输出模式。这是 TX 和 RX 寄存器的典型配置。

在 TX 侧，数据路径状态机使用 “empty” （空）来确定是否有字节可用。“Empty” （空）与 DP 状态机是同步设置的，但被异步清除的 （由于进行了一个总线写操作）。被清除时，该状态与 DP 状态机同步。

在 RX 侧，数据路径状态机使用 “full” （满）来确定 FIFO 中是否还存在空间，以执行写操作。 Full （满）与 DP 状态机是同步设置的，但被异步清除的 （由于进行了一个总线读操作）。被清除时，该状态与 DP 状态机同步。

通过使用 UDB CFG16 寄存器的一个单 FIFO ASYNCH 位，可以使能该同步方式；被设置时，它适用于两个 FIFO。它仅适用于模块状态，因为它假设总线状态是通过中断程序自然同步的。

图 16-13. FIFO 异步操作



FIFO 上溢操作

通过使用 FIFO 状态信号可以安全实现内部（数据路径）和外部（CPU）读 / 写操作。而 FIFO 不具备下溢和上溢条件的内置保护。如果 FIFO 已满，并且发生后续写入操作 （上溢），则新的数据将覆盖掉 FIFO 中前部分的数据 （当前数据被输出，将读取下一个数据）。如果 FIFO 为空，并且发生后续读取操作（下溢），则读取的值是未定义值。无论下溢和上溢情况如何， FIFO 指针都将保持正确。

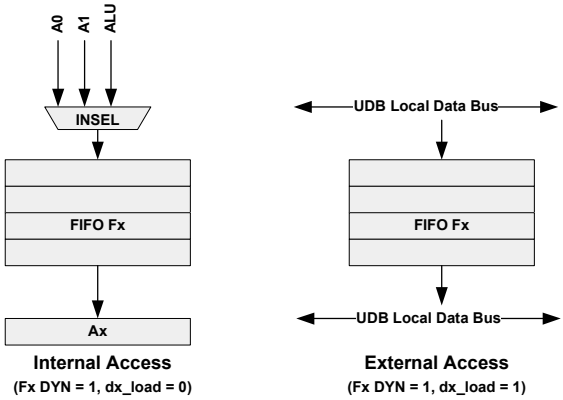
FIFO 时钟反转选项

每个 FIFO 的 UDB CFG16 寄存器都有一个名称为 Fx CK INV 的控制位，用于控制 FIFO 时钟的极性，使之与 DP 时钟的极性相对。默认情况下，在运行过程中， FIFO 的极性和 DP 时钟极性相同。置位该位时， FIFO 的运行极性与 DP 时钟的极性相对。这样便支持 “ 两个时钟沿 ” 的通信协议（如 SPI）。

FIFO 动态控制

一般情况下， FIFO 通过静态方式被配置为输入或输出模式。另外，可以将每一个 FIFO 配置为一种模式，其中方向是由路由信号动态控制的。通过使用每个 FIFO 中的一个配置位（UDB CFG17 寄存器的 Fx DYN 位）可以使能该模式。图 16-14 显示了动态 FIFO 模式的可用配置。

图 16-14. FIFO 动态模式



在内部访问模式下，数据路径可以对 FIFO 进行读 / 写操作。在该配置中，必须配置 Fx INSEL 位以选择 FIFO 写操作源。在该模式下， Fx INSEL = 00 （CPU 总线源）无效，因此该值必须为 01、 10 或 11 （A0、 A1 或 ALU）。请注意，只能对相应的累加器进行读访问；在该模式下，数据寄存器目标不可用。

在外部访问模式下，CPU 可以对 FIFO 进行读 / 写操作。通过使用数据路径路由信号可以在内部和外部访问之间进行动态切换。数据路径输入信号 d0_load 和 d1_load 都用于该控制操作。请注意，在动态控制模式下，不能使用 d0_load 和 d1_load 进行从 F0/F1 加载 D0/D1 寄存器。dx_load 信号可以由任何路由信号 （包括各常数）驱动。

在一个使用示例中，开始进行外部访问 （dx_load == 1）后， CPU 可以将一个或多个数据字节写入到 FIFO 内。然后切换到内部访问 （dx_load == 0），数据路径可以对数据进行各种操作。切换回外部访问时， CPU 可以读取计算的结果。

由于始终要将 Fx INSEL 设置为 01、 10 或 11 （A0、 A1 或 ALU）（表示处于正常操作的 “ 输出模式 ”），因此 FIFO 状态信号具有以下的含义 （独立于 Fx LVL 控制）。

表 16-6. FIFO 状态

状态信号	含义	Fx LVL = 0	Fx LVL = 1
fx_blk_stat	写状态	FIFO 已满	FIFO 已满
fx_bus_stat	读状态	FIFO 非空	至少一半已满

由于数据路径和 CPU 可以对 FIFO 进行读和写操作，因此不再将这些信号视为 “ 模块 ” 和 “ 总线 ” 状态。blk_stat 信号用于写状态， bus_stat 信号用于读状态。

16.2.2.3 FIFO 状态

有 4 个 FIFO 状态信号，每个 FIFO 使用其中两个：fifo0_bus_stat、 fifo0_blk_stat、 fifo1_bus_stat 和 fifo1_blk_stat。这些信号的含义取决于已给 FIFO 的方向，该方向由静态配置确定。

16.2.2.4 数据路径 ALU

ALU 内核包含 3 个独立的 8 位可编程功能，分别为：一个算术 / 逻辑单元、一个移位器单元和一个掩码单元。有关详细信息，请参考 UDB 数据路径架构框图 （图 16-6）。

算法和逻辑操作

表 16-7 中显示的是 ALU 功能，这些功能由配置 RAM 动态配置。

表 16-7. UDB DCFG 寄存器中的 ALU 功能

Func[2:0]	函数	操作
000	PASS	srca
001	INC	++srca
010	DEC	--srca
011	ADD	srca +srcb
100	SUB	srca – srcb
101	XOR	srca ^ srcb
110	AND	srca and srcb
111	OR	srca | srcb

srca = ‘A’ 输入源连接到 ALU， srcb = ‘B’ 输入源连接到 ALU。请参见图 16-6。

Carry In （进位输入）

carry in 作为算术操作。表 16-8 显示的是某些函数的默认 carry in 值。

表 16-8. Carry In （移入）函数

函数	操作	默认的 Carry In （移入）实现
INC	++srca	srca + 00h + ci，其中 ci 被强制设置为 1
DEC	--srca	srca + ffh + ci，其中 ci 被强制设置为 0
ADD	srca + srcb	srca + srcb + ci，其中 ci 被强制设置为 0
SUB	srca – srcb	srca + ~srcb + ci，其中 ci 被强制设置为 1

除了可用于进位操作的该默认算术模式外，还有其他三个进位选项。使用 CFG13 寄存器中的 CI SELA 和 CI SELB 配置位可以确定一个给定周期的 carry in。通过动态配置 RAM 可以按周期性选择 A 或 B 配置。在表 16-9 中定义了各选项。

表 16-9. UDB CFG13 中的其他 Carry In 函数

CI SEL A CI SEL B	Carry 模式	说明
00	默认	默认算术模式，如表 16-8 中所述。
01	寄存	Carry 标志 — 上一周期中的 carry 结果。该模式用于实现 add with carry（进位加）和 subtract with borrow（借位减）运算。可以在连续周期内使用该模式，以仿真一个双精度操作。
10	互连	在其他位置生成 Carry 信号，并将它路由到该输入端。该模式可用于实现可控制的计数器。
11	级联	Carry 信号从前一数据路径级联。该模式可用于实现两个或更多数据路径上精度更高的单周期操作。

使用路由进位时，它在每个算术函数的相应意义在表 16-10 中显示。请注意，对于递减和减法函数，该进位为低电平有效 （反转）。

表 16-10. 互连进位输入函数

函数	进位输入信号极性	进位输入有效	进位输入无效
INC	真	++srca	srca
DEC	反转	--srca	srca
ADD	真	(srca + srcb) + 1	srca + srcb
SUB	反转	(srca – srcb) – 1	(srca – srcb)

Carry Out （进位输出）

Carry out 是一个可选的数据路径输出，由静态可编程的当前定义 MSB 位置派生。该值还可以作为可选的 carry in 链接到下一个最高有效模块。请注意，对于递减和减法函数， carry out 被反转。

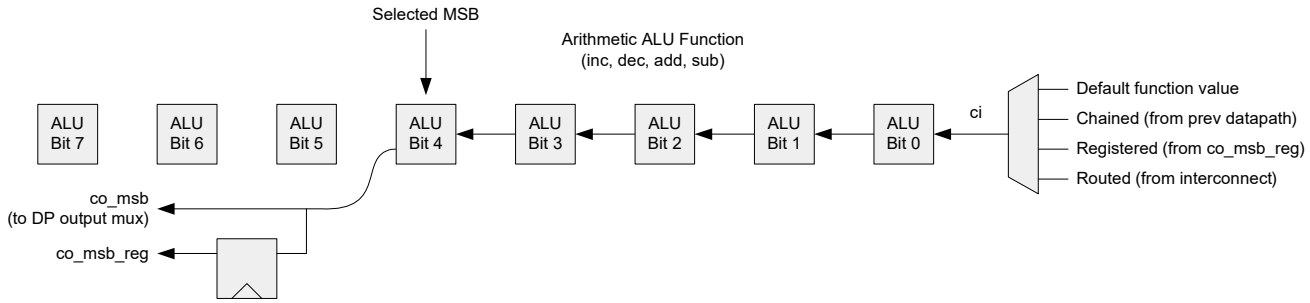
表 16-11. 进位输出函数

函数	进位输出信号极性	进位输出有效	进位输出无效
INC	真	++srca == 0	srca
DEC	反转	--srca == –1	srca
ADD	真	srca + srcb > 255	srca + srcb
SUB	反转	srca – srcb < 0	(srca – srcb)

进位结构

图 16-15 显示的是 carry in 选项和用于生成 carry out 的 MSB 选项。寄存的 carry out 值可以作为后续算术操作的 carry in 使用。在多个周期中，可以通过该特性来实现更高精度功能。

图 16-15. 进位操作



移位操作

根据表 16-12，移位操作独立于 ALU 操作。

表 16-12. UDB DCFG 寄存器中的移位操作函数

Shift[1:0]	函数
00	通过
01	向左移位
10	向右移位
11	半字节交换

Shift out 值可作为一个数据路径输出使用。 Shift out right （右移输出 — sor）和 shift out left （左移输出 — sol_msb）共享该输出选择。静态配置位 （UDB CFG15 寄存器中的 SHIFT SEL）用于确定作为数据路径输出使用的移位输出。没有发生移位时， sor 和 sol_msb 信号分别被定义为 ALU 功能的最低有效位和最高有效位。

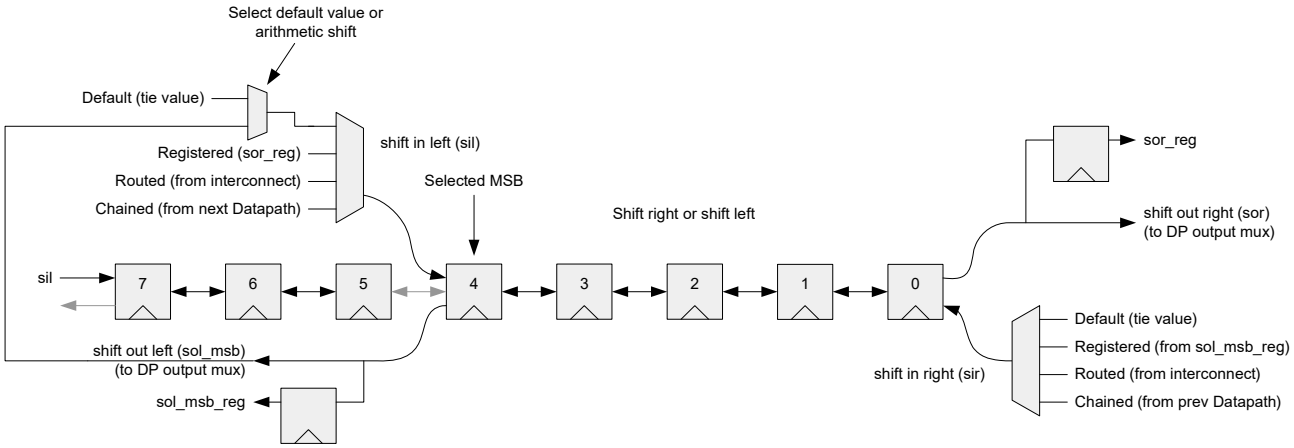
通过 SI SELA 和 SI SELB 配置位可以确定某个已给操作的 shift in 数据。通过动态配置 RAM 可以根据周期来选择 A 或 B 配置。Shift in 数据仅适用于左移和右移；它不适用于传送和半字节交换。表 16-13 显示的是左移和右移的选项以及使用情况。

表 16-13. UDB CFG15 寄存器中的移入函数

SI SEL A/ SI SEL B	移入源	说明
00	默认 / 算术	默认输入是 DEF SI 配置位的值 （固定为 1 或 0）。然而，如果设置 MSB SI 位，默认输入将为当前定义的最高有效位 （仅用于右移）。
01	寄存	移入值是由 （前一周期）寄存的当前移出值驱动。左移操作使用最后左移输出的值。右移操作使用最后右移输出的值。
10	互连	移入的数值是从路由通道 （SI 输入）中选择的。
11	级联	左移输入来自右侧相邻的数据路径，右移输出来自左侧相邻的数据路径

Shift out left 数据来自当前定义的 MSB 位置 （CFG14 寄存器中的 MSB_EN 和 MSB_SEL 位），并且从左边移入的数据 （在右移操作中）进入当前定义的 MSB 位置。可以寄存 shift out 数据 （左或右）并在后续周期内使用它。在多个周期中，可以使用该特性来实现更高精度的 shift in 操作。

图 16-16. 移位操作



请注意，由 MSB 选项隔离的各位仍然被移位。如该实例所示，位 7 通过右移移到 sil 值，位 5 通过左移移到位 4。从隔离位进行的右 shift out 或左 shift out 操作被抛弃。

ALU 掩码操作

通过 UDB 静态配置寄存器 （CFG9）中的 8 位掩码寄存器 （AMASK），可以定义掩码操作。在该操作中， ALU 的输出由掩码寄存器中的值掩码 （进行 AND 运算）。 ALU 掩码功能通常用于实现分辨率为 2 幂次的自由运行定时器和计数器。

16.2.2.5 数据路径输入和复用

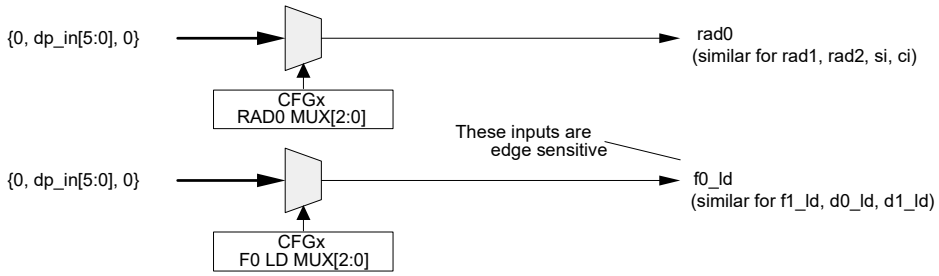
如表 16-14 中所示，数据路径共有 9 个输入，其中 6 个输入来自通道路由。这些输入包括配置 RAM 地址、FIFO 和数据寄存器加载控制信号，以及数据输入的 shift in 和 carry in 信号。

表 16-14. 数据路径输入

输入	说明
RAD2 RAD1 RAD0	异步动态配置的 RAM 地址。共有八个用户可编程的 16 位字。每个字都包含了当前周期的数据路径控制位。各指令的序列均由这些地址输入控制。
F0 LD F1 LD	在给定周期中激活它时，将使用一个 A0 或 A1 累加器数据或 ALU 输出的数据载入选定 FIFO。 Fx INSEL[1:0] 配置位用于选择输入源。该输入是边沿敏感的。。它在数据路径时钟上被采样；当检测到 ‘0’ 至 ‘1’ 的转换时，将在随后的时钟边缘上执行加载操作。
D0 LD D1 LD	在给定的周期中激活它时，将从相应的 FIFO Fx 加载 Dx 寄存器。该输入是边沿敏感的。它在数据路径时钟上被采样；当检测到 ‘0’ 至 ‘1’ 的转换时，将在随后的时钟边缘上执行加载操作。
SI	这是可用于左移输入或右移输入的数据输入值。
CI	它是 carry in select control （进位输入选择控制）被设置为 “routed carry” （互连进位）时使用的移入值。

如图 16-17 中所示，每个输入都有一个 6 输入 1 输出的复用器，因此所有输入都是可交换的。可以通过电平敏感或边沿敏感的方式来处理这些输入。 RAM 地址、 shift in 和 data in 值都是电平敏感的； FIFO 和数据寄存器加载信号则是边沿敏感的。

图 16-17. 数据路径输入选择



16.2.2.6 CRC/PRS 支持

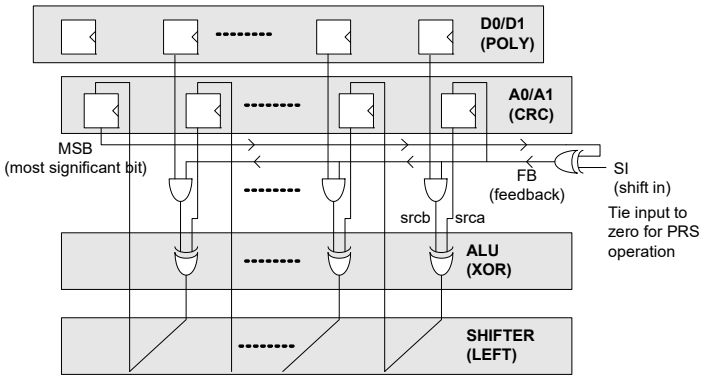
数据路径支持循环冗余校验 （CRC）和伪随机序列 （PRS）的生成。链接信号在各数据路径模块之间被路由，以支持超过 8 位的 CRC/PRS 位长度。

CRC/PRS 计算中最高有效模块的最高有效位 （MSB）被选中并被路由 （在各个模块上链接）到最低有效模块。然后最高有效位与数据输入 （SI 数据）进行 XOR 运算，以提供反馈 （FB）信号。这时反馈信号被路由 （在各模块上链接）到最高有效模块。该反馈值被使用于所有模块，以便传输使用当前累加器值将多项式（Data0 或 Data1 寄存器中）进行 XOR 运算后得到的值。

图 16-18 显示了 CRC 操作的结构配置。PRS 配置是相同的，但 shift in （SI）被连接到 ‘0’ 。在 PRS 配置中， D0 或 D1 包含了多项式的值，而 A0 或 A1 则包含了计算结果的初始 （种子）值和 CRC 余数。

为了使能 CRC 操作，必须将动态配置 RAM 中的 CFB_EN 位设置为 ‘1’ 。这样可以对 SRCB ALU 输入和 CRC 反馈信号进行 AND 运算。将该位置位为 ‘0’ 时，反馈信号被驱动为 ‘1’ 。这样可以进行普通的算术操作。按周期动态控制该位时，CRC/PRS 操作可以与其他算术操作交叉进行。

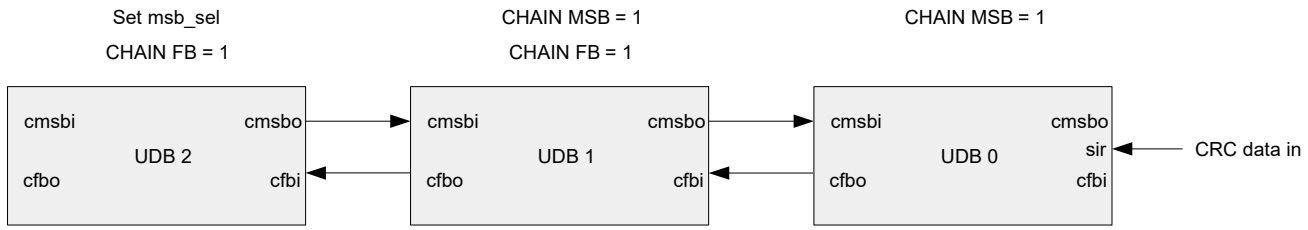
图 16-18. CRC 功能结构



CRC/PRS 链接

图 16-19 显示的是 3 个 UDB 上的 CRC/PRS 链接的示例。这种配置可支持 17 到 24 位操作。根据链接中数据路径的位置来设置链接控制位，如该图所示。

图 16-19. CRC/PRS 链接配置



CRC/PRS 反馈信号 （cfbo、 cfbi）如下链接：

- 如果已给模块为最低有效模块，那么反馈信号将从该模块的内置逻辑中生成。该逻辑从右边获取 shift in，并将该值与最高有效位信号进行 XOR 运算。（对于 PRS，“sir” 信号被连接到 ‘0’ 。）
- 如果给定模块不是最低有效模块，那么必须要设置 CHAIN FB 配置位，并且反馈信号被传输给链接中的前一个模块。

CRC/PRS MSB 信号 （cmsbo、 cmsbi）如下链接：

- 如果给定模块为最高有效模块，那么（根据多项式选择的）最高有效位将通过使用 UDB CFG14 寄存器中的 MSB_SEL 配置位进行配置。
- 如果给定模块不是最高有效模块，则必须设置UDB CFG14寄存器的CHAIN CMSB配置位，并且最高有效位信号被传输给链接中的下一个模块。

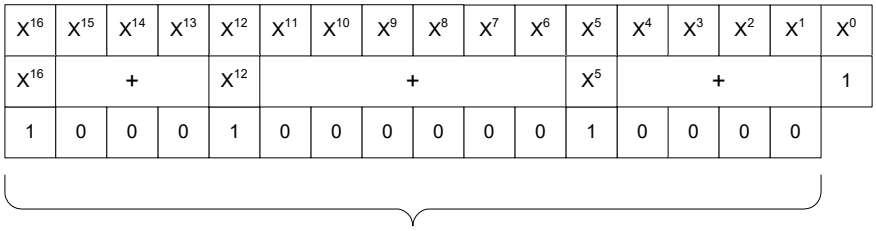
CRC/PRS 多项式规范

例如，要想配置多项式，以将其编程到相应的 D0/D1 寄存器内，那么需要考虑定义为 $x^{16} + x^{12} + x^5 + 1$ 的 CCITT CRC-16 多项式。图 16-20 中显示了从多项式获取数据格式的方式。

X^0 始终为 ‘1’，因此不需要进行编程。而对于多项式中剩余的每一项，一个 ‘1’ 被设置在显示对齐中合适的位置上。

注意：该多项式格式与通常指定的 Hex 格式稍微不同。例如， CCITT CRC16 多项式通常表示 1021H。要想转换到数据路径操作需要的格式，需要右移一个值，并将在最高有效位内添加一个 ‘1’ 。这时，加载到 D0 或 D1 寄存器内正确的多项式值将为 8810H。

图 16-20. CCITT CRC16 多项式格式



CRC/PRS 的示例配置

下面总结了 CRC/PRS 配置要求 （假设 D0 是多项式，且 CRC/PRS 是在 A0 中计算得到的）：

1. 选择一个合适的多项式，并将其写入到 D0 内。
2. 选择合适的种子值 （例如， CRC 全为 ‘0’ 、 PRS 全为 ‘1’ ），并将其写入到 A0 内。
3. 配置链接 （若需要）。
4. 根据该多项式内所定义的 MSB_SEL 静态配置寄存器位来选择 MSB 位置，并设置 UDB CFG14 寄存器的 MSB_EN 位。
5. 配置动态配置 RAM 字域：
   a. 将 D0 选择为 ALU “SRCB” （ALU B 输入源）
   b. 将 A0 选择为 ALU “SRCA” （ALU A 输入源）
   c. 为 ALU 功能选择 “XOR”
   d. 为 SHIFT 功能选择 “SHIFT LEFT”
   e. 选择 “CFB_EN”，以支持 CRC/PRS
   f. 将 ALU 选择为 A0 写入源

如果是一个 CRC 操作，则需要为路由中的输入数据配置 “shift in right”，并在每个时钟上提供输入。如果是一个 PRS 操作，请将 “shift in right” 连接到 ‘0’。

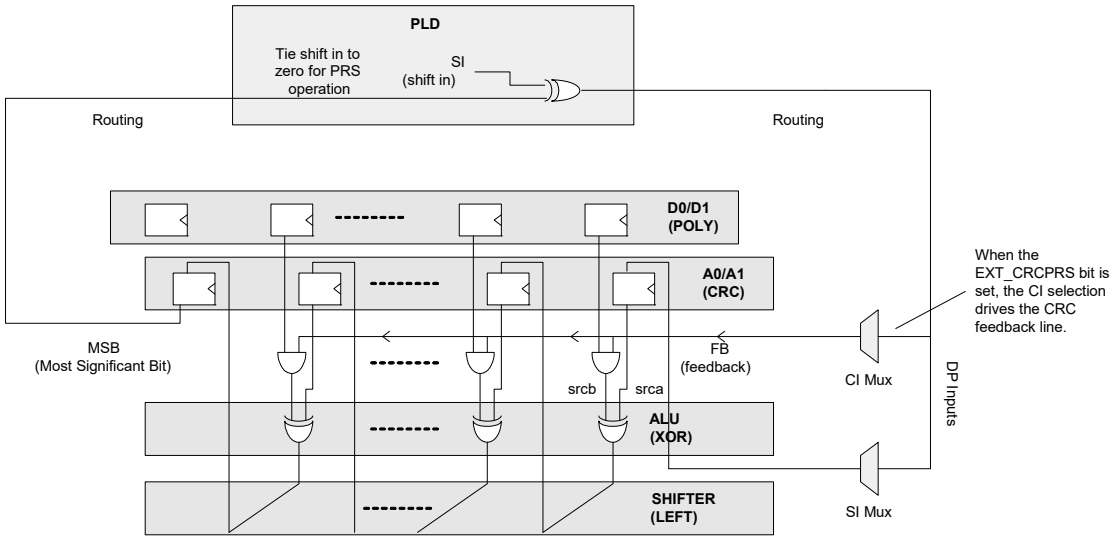
通过该配置给 UDB 提供时钟，会生成所需要的 CRC 或输出最高有效位。该位将被输出到 PRS 序列的布线。

外部 CRC/PRS 模式

通过设置静态配置位（UDB CFG16 寄存器的 EXT CRCPRS），可以支持 CRC 或 PRS 的外部计算。如图 16-21 中所示，CRC 反馈的计算是在 PLD 模块中进行的。设置该位时，可以从 CI （Carry In）数据路径输入选择复用器直接驱动 CRC 反馈信号，不用进行内部计算。该图显示的是支持 8 位 CRC 或 PRS 的简单配置。内置电路通常用于一般情况，但它还支持更复杂的配置，如在某个 UDB 中通过时分复用来实现 16 位 CRC/PRS 功能。

在该模式下，UDB DCFG0 寄存器的动态配置 RAM 位 CFB_EN 仍然控制着是否将 CRC 反馈信号与 SRCB ALU 输入进行 AND 运算。因此，如内置 CRC/PRS 操作，该功能可以与其他功能交叉使用 （若需要）。

图 16-21. 外部 CRC/PRS 模式



16.2.2.7 数据路径输出和复用

各个条件均由各寄存累加器值、 ALU 输出和 FIFO 状态生成。可以将这些条件驱动给数字路由，以使用于其他 UDB 模块，并作为各个中断或者将它们驱动到 I/O 引脚上。表 16-15 显示的是 16 种可能条件。

表 16-15. 数据路径条件生成

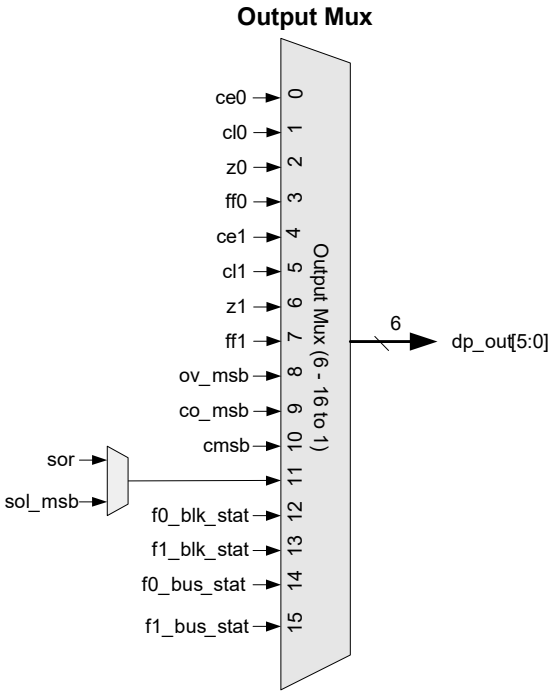
名称	条件	链接	说明
ce0	“ 等于 ” 比较	是	A0 == D0
cl0	“ 小于 ” 比较	是	A0 < D0
z0	“0” 值检测	是	A0 == 00h
ff0	“1” 值检测	是	A0 == FFh
ce1	“ 等于 ” 比较	是	A1 或 A0 == D1 或 A0 （动态选择）
cl1	“ 小于 ” 比较	是	A1 或 A0 < D1 或 A0 （动态选择）
z1	“0” 值检测	是	A1 == 00h
ff1	“1” 值检测	是	A1 == FFh
ov_msb	溢出	否	Carry(msb) ^ Carry(msb–1)
co_msb	进位	是	对 MSB 定义位进行进位
cmsb	CRC MSB	是	CRC/PRS 功能的最高有效位
So	移出	是	移位输出选择

表 16-15. 数据路径条件生成

名称	条件	链接	说明
f0_blk_stat	FIFO0 模块状态	否	定义取决于 FIFO 的配置情况
f1_blk_stat	FIFO1 模块状态	否	定义取决于 FIFO 的配置情况
f0_bus_stat	FIFO0 总线状态	否	定义取决于 FIFO 的配置情况
f1_bus_stat	FIFO1 总线状态	否	定义取决于 FIFO 的配置情况

共有 6 个数据路径输出端。如图 16-22 中所示，每个输出端都有一个 16-1 （16 输入 1 输出）的复用器，通过该复用器可以将 16 个信号中的任意一个路由到任意数据路径输出端。

图 16-22. 输出复用连接



比较器

共有两个比较器，其中一个具有固定源（比较器 0），而另一个则具有动态的可选源（比较器 1）。每个比较器均有一个 8 位静态编程的掩码寄存器，通过它可在一个指定的位字段中进行比较操作。默认情况下，该掩码无效（比较所有位），因此必须使能它。

可动态配置比较器 1 输入端。如表 16-16 所示，比较器 1 共有四种选项，均适用于 “ 小于 ” 和 “ 等于 ” 条件。UDB CFG12 寄存器中的 CMP SELA 和 CMP SELB 配置位决定了可能的比较配置。通过 UDB DCFG0 寄存器中的动态配置 RAM 位 CMP SEL，可以按周期选择 A 或 B 配置。

表 16-16. 比较配置

CMP SEL A CMP SEL B	比较器 1 的比较配置
00	将 A1 与 D1 进行比较
01	将 A1 与 A0 进行比较
10	将 A0 与 D1 进行比较
11	将 A0 与 A0 进行比较

比较器 0 和比较器 1 可被独立链接到前一个数据路径生成的条件（按照地址顺序）。通过UDB CFG14寄存器中的CHIAN0 和 CHAIN1 位可以确定链接比较器。图 16-23 显示的是比较相等链接，这便是该模块中的比较相等和前一模块中的链接输入进行 “AND” 运算的结果。

图 16-23. 比较 “ 相等 ” 链接

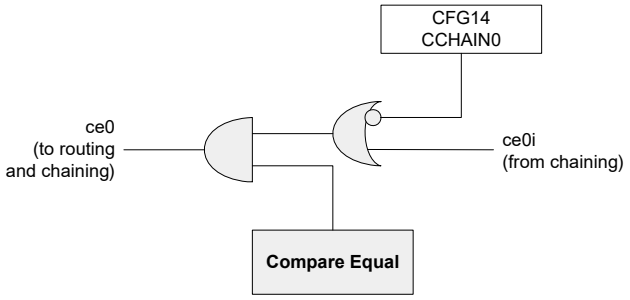
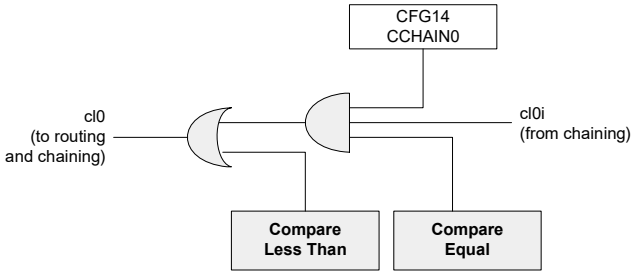


图 16-24 显示的是比较 “ 小于 ” 链接。在这种情况下，“ 小于 ” 由该模块中无条件的比较小于输出形成。它与该模块为相等的条件进行 “OR” 运算，并且前一模块中的链接输入被确认为 “ 小于 ”。

图 16-24. 比较 “ 小于 ” 链接



全零和全一检测

每个累加器均有专用的全零检测和全一检测条件。可以静态链接这些条件，如 UDB 配置寄存器中所指定的情况。通过 UDB 配置寄存器中所指定的信息，可确定是否链接这些条件。零检测的链接和相等比较是相同的概念。如果链接被使能，那么将对连续链接数据进行 “AND” （和）运算。

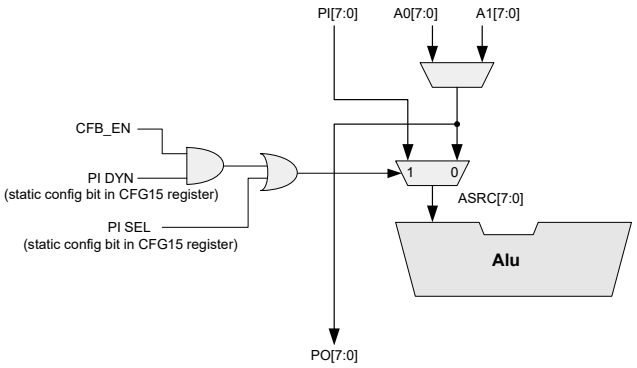
上溢

上溢是指对 carry（进位）移入 MSB（最高有效位）和 carry 移出 MSB 进行 “XOR” 运算。计算操作在当前定义的 MSB 实现，如 MSB_SEL 位所指定。不可链接该条件。然而，如果进位在各模块之间链接，而且在多精度功能的最高有效数据路径中执行计算操作，该操作将为有效的。

16.2.2.8 数据路径的并行输入和输出

如图 16-25 中所示，数据路径并行输入 （PI）和并行输出（PO）信号对于直接将路由数据传输给数据路径和从数据路径输出提供的能力有限。并行输出信号始终可以路由，并作为 A0 和 A1 之间的 ALU asrc 选择。

图 16-25. 数据路径的并行输入 / 输出

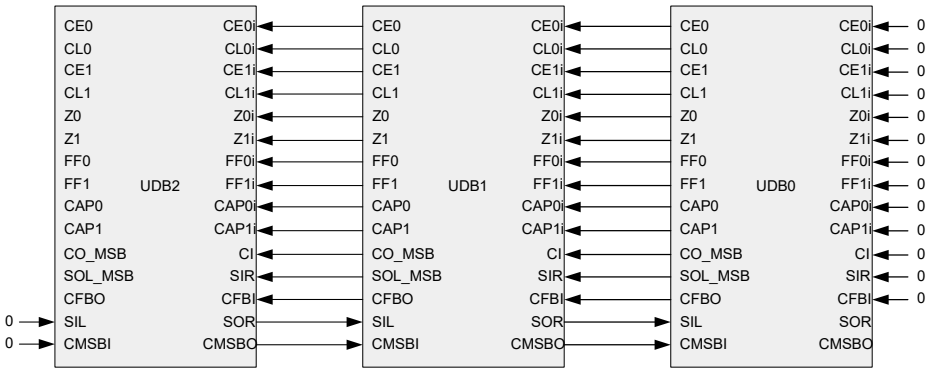


需要为 ALU 的输入选择并行输入。两个可用的选项为静态操作或动态操作。对于静态操作， UDB CFG15 寄存器中的 PI SEL 位会强制将 ALU asrc 作为 PI。UDB CFG15 寄存器中的 PI DYN 位用于使能 PI 动态操作。当该操作被使能，并假设 PI SEL 为 ‘0’ 时，CFB_EN（UDB DCFG0 寄存器）动态控制位可控制 PI 复用器。CFB_EN 主要用于使能 PRS/CRC 功能。

16.2.2.9 数据路径链接

每个数据路径模块都包含了一个 8 位的 ALU，用于将各进位、移位数据、捕获触发器和条件信号链接到最近的相邻数据路径，从而提供精度更高的算术功能和移位器。通过这些专用的链接信号，可以有效地实现单周期 16、 24 和 32 位功能，而不会发生通道路由资源的时序误差。另外，捕获链接还支持在链接模块中对累加器进行原子读操作。如图 16-26 所示，所有生成的条件和捕获信号链接的方向都是从最低有效模块到最高有效模块。向左移位操作也是从最低有效模块链接到最高有效模块。而向右移位操作是从最高有效模块链接到最低有效模块。反馈的 CRC/PRS 链接信号则从最低有效模块链接到最高有效模块。MSB 输出是从最高有效模块链接到最低有效模块。

图 16-26. 数据路径链接流

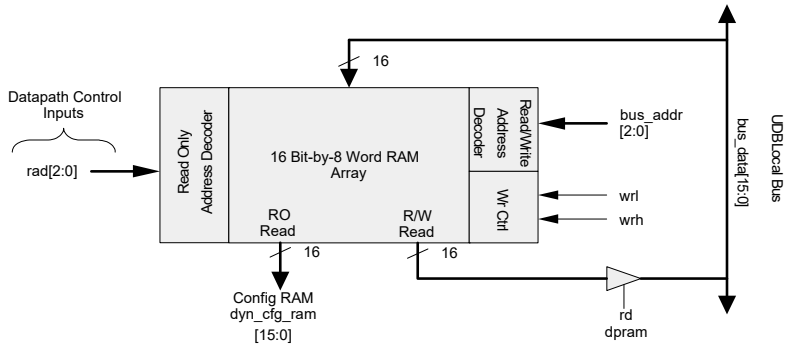


16.2.2.10 动态配置 RAM

每个数据路径包含一个大小为 8x16 位字的动态配置 RAM，如图 16-27 中所示。该 RAM 的功能是根据数据路径所选的时钟按周期控制数据路径配置位。该 RAM 具有同步的读和写端口，以便通过系统总线加载配置。

此外，还提供了另一个异步读端口作为快速路径使用，用以将这些 16 位字转为控制位，然后传送到数据路径。异步地址输入是从数据路径输入中选择的，它们可以从通道路由上的任何一个可能信号（如 I/O 引脚、PLD 输出、控制模块输出或其他数据路径输出）生成。异步读取路径的主要功能是为数据路径位提供快速单周期解码。

图 16-27. 配置 RAM I/O



下面显示的是动态配置 RAM 字的字段信息。下面介绍的是各个字段的使用情况。

寄存器	地址	15	14	13	12	11	10	9	8
CFGRAM	61h - 6Fh （奇数）	FUNC[2:0]			SRCA	SRCB[1:0]		SHIFT[1:0]	

寄存器	地址	7	6	5	4	3	2	1	0
CFGRAM	60h - 6Eh （偶数）	A0 WR SRC[1:0]		A1 WR SRC[1:0]		CFB EN	CI SEL	SI SEL	CMP SEL

表 16-17. 动态配置的快速参考

字段	位	参数	值
FUNC[2:0]	3	ALU 功能	000：PASS 001：INC SRCA 010：DEC SRCA 011：ADD 100：SUB 101：XOR 110：AND 111：OR
SRCA	1	ALU A 输入源	0：A0 1：A1
SRCB	2	ALU B 输入源	00：D0 01：D1 10：A0 11：A1
SHIFT[1:0]	2	SHIFT 功能	00：PASS 01：左移 10：右移 11：半字节交换
A0 WR SRC[1:0]	2	A0 写入源	00：无 01：ALU 10：D0 11：F0
A1 WR SRC[1:0]	2	A1 写入源	00：无 01：ALU 10：D1 11：F1
CFB EN	1	CRC 反馈使能	0：使能 1：禁用
CI SEL	1	Carry In 配置选择	0：ConfigA 1：ConfigB[a]
SI SEL	1	Shift In 配置选择	0：ConfigA 1：ConfigB[a]
CMP SEL	1	比较配置选择	0：ConfigA 1：ConfigB[a]

a. RAM 域为 CI、SI 和 CMP 选择两个预定义状态设置中的一个。请分别参见表 16-9、表 16-13 和表 16-16 中的内容。

16.2.3 状态和控制模块

图 16-28 显示的是状态和控制模块的顶层视图。控制寄存器驱动到路由，从而为 UDB 操作提供固件控制输入。固件通过在路由过程中读取状态寄存器的数值来监控 UDB 操作的状况。

图 16-29 显示的是状态和控制模块的详细视图。该模块的主要用途是协调 CPU 固件与内部 UDB 操作之间的交互。然而，由于该模块能够灵活地连接到路由矩阵，因此可将其配置为执行其他功能。

图 16-28. 状态和控制寄存器

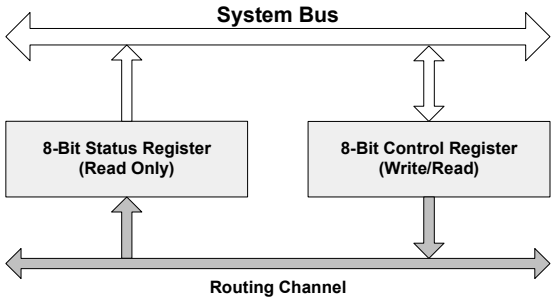
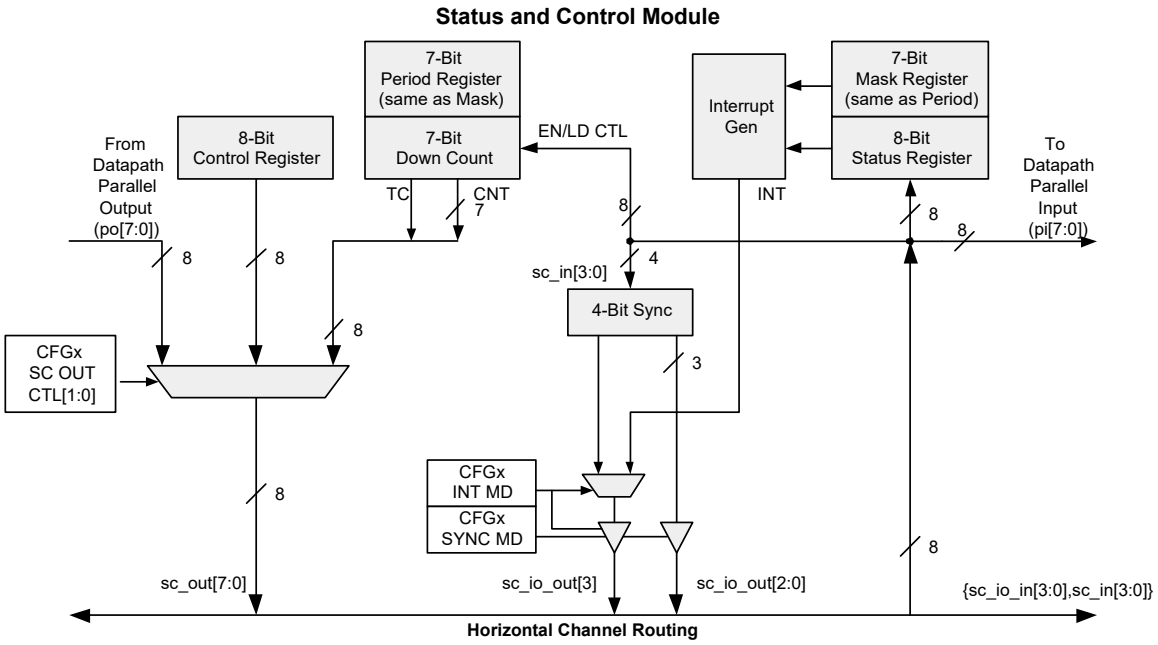


图 16-29. 状态和控制模块



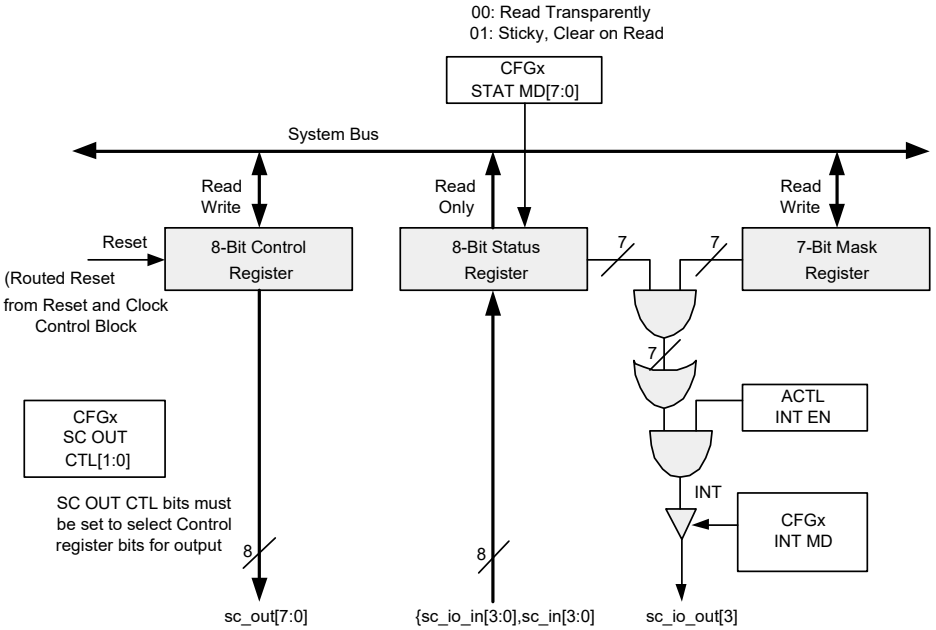
各操作模式包括：

- **状态输入** — 路由信号的状态可以为输入，可以捕获它作为状态并由 CPU 读取。
- **控制输出** — CPU 可以通过写入到控制寄存器内驱动路由状态。
- **并行输入** — 到数据路径并行输入。
- **并行输出** — 从数据路径并行输出。
- **寄存器模式** — 在该模式下，控制寄存器作为 7 位递减计数器运行，并且具有可编程周期和自动重新加载的特性。通过配置路由输入，可以控制计数器的使能和重新加载操作。当该模式被使能时，控制寄存器操作不可用。
- **同步模式** — 在该模式下，状态寄存器作为一个 4 位的双同步器运行。使能该模式时，状态寄存器操作不可用。

16.2.3.1 状态和控制模式

该模块运行于状态和控制模式时，它可作为状态寄存器、中断掩码寄存器和控制寄存器使用，如图 16-30 中所示的配置。

图 16-30. 状态和控制操作



状态寄存器操作

每个 UDB 都有一个 8 位的只读状态寄存器。该寄存器的输入来自数字路由结构中的所有信号。状态寄存器是非保存的寄存器，它在睡眠间隔内丢失状态，并在唤醒时复位为 0x00。可以独立编程每一位，以便让它能以两个方式中的一个运行，如表 16-18 中所示。

表 16-18. UDB CFG20 寄存器中的状态寄存器模式选择

STAT MD	说明
0	透明读取。读操作会返回路由信号的当前值
1	粘滞位，在读取时被清除。在输入中产生的高电平信号被采样和捕捉。读取寄存器时，它将被清除。

请注意，状态寄存器清除操作只适用于已设置的位。因此，尚未置位的其他位将继续捕获状态，从而过程可保持一致。

透明状态读取

默认情况下， CPU 读取寄存器时也会透明地读取相关路由的状态。该模式可用于在 UDB 中计算和内部寄存的瞬变状态。

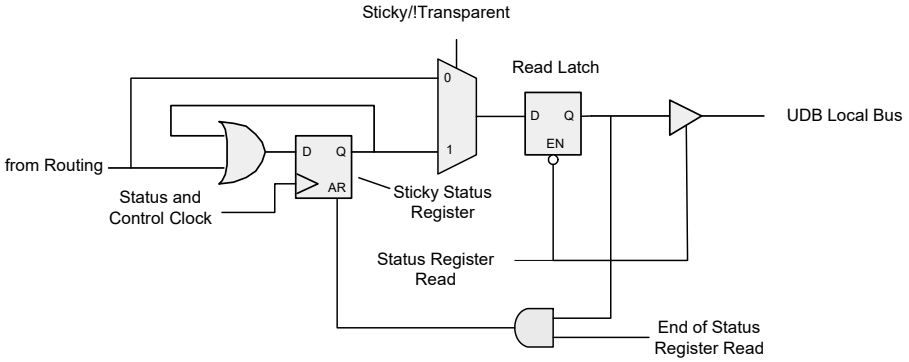
粘滞状态，在读取后被清除。

在该模式下，状态和控制时钟完成每个周期后，会对状态寄存器输入进行采样。如果信号在指定采样中为高电平，则在状态位上被捕获，并且不管输入的后续状态如何它都会保持为高电平。当 CPU 读取状态寄存器时，将清除该位。状态寄存器的清除不依赖于任何模式，即使在 UDB 时钟被禁用时也同样发生；它根据 HFCLK 作为读取操作的一部分而发生。

读取期间锁存状态

图 16-31 显示的是状态读取逻辑的结构。粘滞状态寄存器后紧跟一个锁存。在读周期期间，无论指定读操作中的等待状态数量是多少，该锁存都用于锁存状态寄存器的数据并保持其稳定状态。

图 16-31. 状态读取逻辑



中断生成

在几乎所有功能中，中断生成取决于状态位的设置情况。如图 16-31 中所示，该功能作为掩码状态和 OR 减少状态被内置到状态寄存器逻辑内。只有状态输入的低 7 位可以与内置中断生成电路一起使用。最高有效位通常作为中断输出使用，并可以通过数字路由被路由给中断控制器。在该配置中，状态寄存器的最高有效位作为中断位的状态被读取。

16.2.3.2 控制寄存器操作

每个 UDB 都有一个 8 位控制寄存器。它作为系统总线上的标准读 / 写寄存器运行，其中这些寄存器位的输出可以用于驱动数字路由结构。

控制寄存器是非保持寄存器；它在睡眠间隔内丢失内容，并在唤醒时复位到 0x00。

控制寄存器的工作模式

可以按位配置三种模式。该配置由 UDB CFG18 和 CFG198 寄存器内的两个 8 位寄存器 CTL_MD1[7:0] 和 CTL_MD0[7:0] 的位串联控制。例如，{CTL_MD1[0],CTL_MD0[0]} 控制着控制寄存器位 0 的模式，如表 16-19 所示。

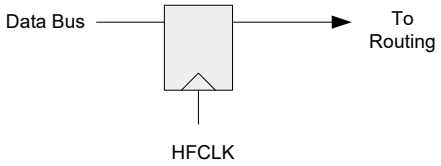
表 16-19. UDB CFG18 和 CFG19 寄存器中控制寄存器位 0 的模式

CTL MD	说明
00	直接模式
01	同步模式
10	双同步模式
11	脉冲模式

控制寄存器的直接模式

默认情况下为直接模式。如图 16-32 所示，当 CPU 写入到控制寄存器内时，该控制寄存器的输出将在该写周期内被直接驱动给路由。

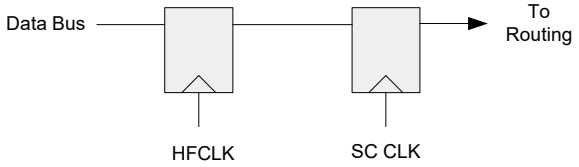
图 16-32. 控制寄存器的直接模式



控制寄存器的同步模式

如图 16-33 所示，在同步模式下控制寄存器输出由一个重新采样寄存器驱动。该寄存器由当前选定的状态和控制 （SC） 时钟提供脉冲。这样允许已选的 SC 时钟 （而不是 HFCLK） 控制输出的时序。

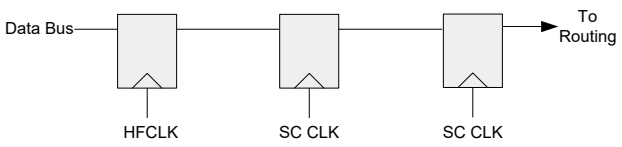
图 16-33. 控制寄存器的同步模式



控制寄存器的双同步模式

在双同步模式下，重新采样寄存器后，将添加由已选 SC 时钟提供脉冲的第二个寄存器，如图 16-34 中所示。这样，当 HFCLK 和 SC 时钟均为异步时，电路将能够稳健运行。

图 16-34. 控制寄存器的双同步模式



控制寄存器的脉冲模式

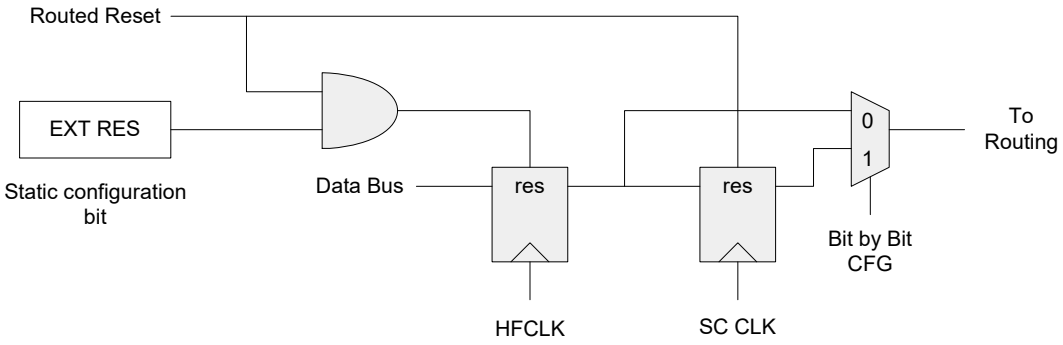
脉冲模式与同步模式相似，在该模式下，SC 时钟也能重新对控制位进行采样。在总线写周期后，脉冲在第一个 SC 时钟周期内开始。控制位的输出在一个完整的 SC 时钟周期内有效。在该时钟周期结束时，控制位被自动复位。

在该操作模式下，固件可以通过将 ‘1’ 写入到控制寄存器内来生成一个脉冲。写入‘1’后，它会被固件回读为‘1’，直到该脉冲结束为止。然后，该控制寄存器的值将被回读为‘0’。固件可以写入另一个 ‘1’，以启动另一个脉冲。只有在完成前一个脉冲后，才能生成新的脉冲。因此，脉冲生成的最大频率是所有其他 SC 时钟周期。

控制寄存器的复位

控制寄存器具有由 EXT RES 配置位控制的两种复位模式，如图 16-35 所示。当 EXT RES 位为 ‘0’ （默认）时，在同步模式或脉冲模式下，路由复位输入将复位同步输出，但不会复位实际的控制位。当 EXT RES 位为 ‘1’ 时，路由复位输入将复位控制位和同步输出。

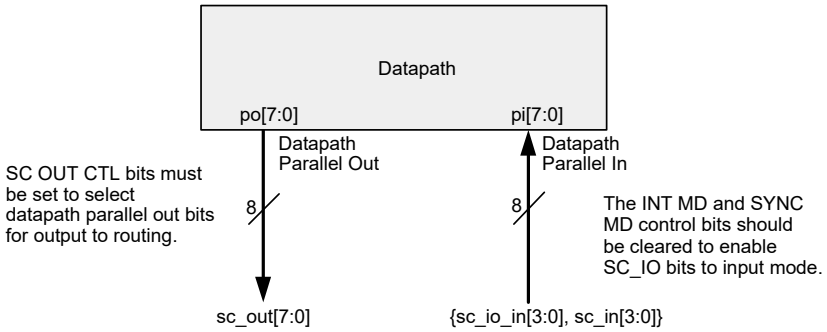
图 16-35. 控制寄存器的复位



16.2.3.3 *并行输入 / 输出模式*

在该模式下，状态和控制路由被连接到数据路径并行输入和并行输出信号，如图 16-36 所示。要想使能该模式，请通过设置 UDB CFG22 寄存器中的 SC OUT 配置位选择数据路径输出。该并行输入连接始终有效，但状态寄存器输入、计数器控制输入和中断输出共享这些路由连接。

图 16-36. 并行输入 / 输出模式



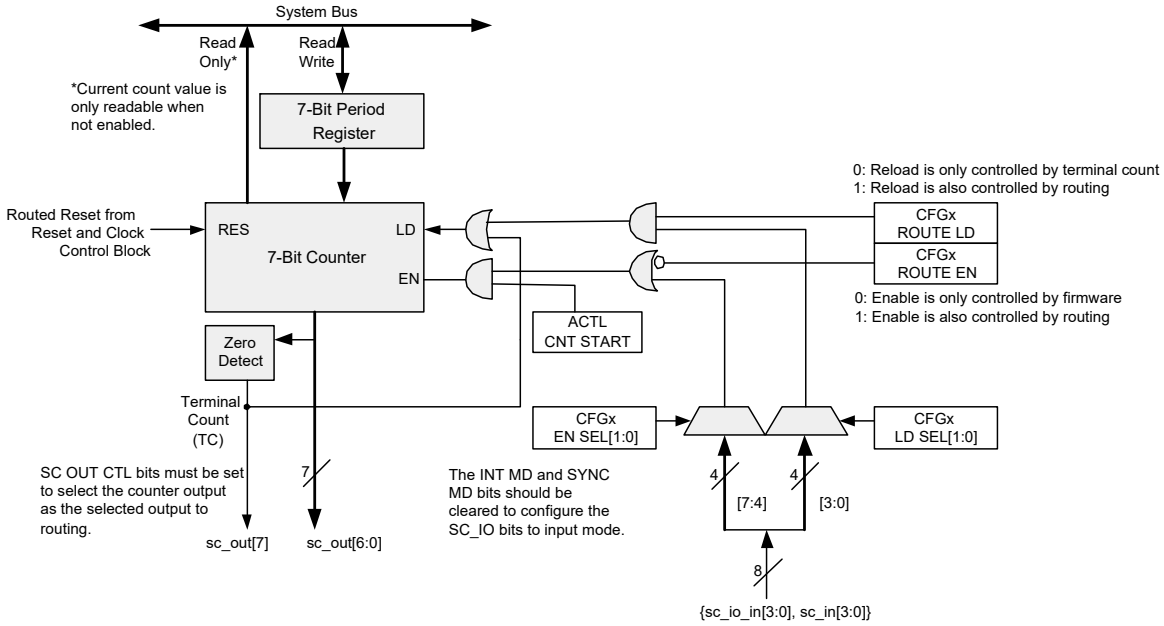
16.2.3.4 计数器模式

如图 16-37 所示，当模块处于计数器模式时， UDB 内部操作或固件应用程序可以使用 7 位递减计数器。该计数器具有下列特性：

- 一个 7 位读 / 写周期寄存器。
- 一个 7 位读 / 写计数寄存器。仅在该计数器被禁用时，才能访问它。
- 在终端计数上（0）自动将周期重新加载到计数寄存器内。
- 辅助工作控制寄存器内的固件控制位 （又称为 CNT START），用于启动和停止计数器。（这是一个覆盖的使能操作，必须进行设置才能执行可选路由的使能）。
- 路由中用于计数器使能和加载功能的可选动态控制的可选位：
  - EN，它是用于启动或停止计数的路由使能。
  - LD，它是用于强制重新加载周期的路由加载信号。当该信号被激活时，它将覆盖一个挂起的终端计数。激活期间，该信号是电平敏感型的，并会继续加载周期。
- 将 7 位计数作为 sc_out[6:0] 驱动到路由结构上。
- 将终端计数作为 sc_out[7] 驱动到路由结构上。
- 在默认模式下，终端计数被寄存。在备用模式下，终端计数被组合。
- 在默认模式下，必须激活路由使能信号 （若使用），使路由加载信号运行。在备用模式下，路由使能和路由加载信号相互独立。

要想使能计数器模式，必须将 UDB CFG22 寄存器中的 SC_OUT_CTL[1:0] 位设置为计数器输出。在该模式下，控制寄存器的普通操作无效。状态寄存器仍然可以用于读操作，但不应该用于生成中断，因为掩码寄存器作为计数器周期寄存器重用。周期寄存器作为保持寄存器实现，并能在睡眠间隔内保持其状态。在 N 个时钟的一个周期内，应该加载 N–1 周期值。 N = 1 （0 周期）不能作为时钟分频值，而且终端计数输出的结果始终为 1。同步模式的使用情况取决于是否使用动态控制输入（LD/EN）。如果没有使用这些输入，则同步模式不受到任何影响。如果使用这些输入， SYNC 模式无效。

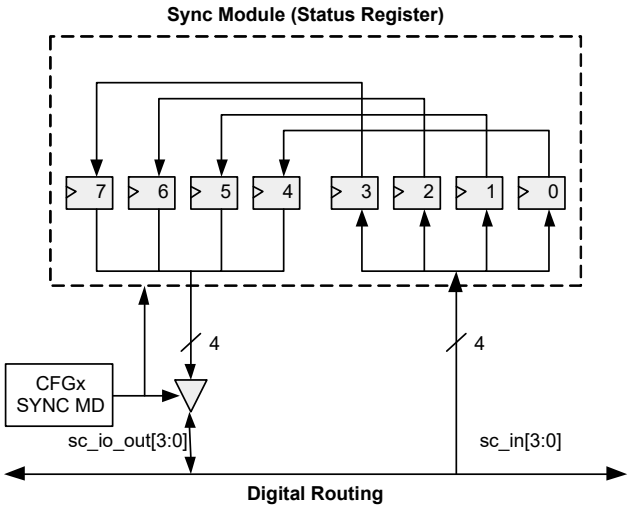
图 16-37. 计数器模式



16.2.3.5 同步模式

如图 16-38 所示，当 UDB CFG22 寄存器中 SYNC MD 位被置位时，状态寄存器可以作为 4 位双同步器运行，并由当前的 SC_CLK 时钟提供脉冲。该模式会用于实现异步信号（如 GPIO）的局部同步。被使能时，要同步的信号是从 SC_IN[3:0] 中选出的，输出被驱动到 SC_IO_OUT[3:0] 引脚，并且 SYNC MD 自动使 SC_IO 引脚进入输出模式。在该模式下，无论该模式的控制设置如何，状态寄存器的正常操作都不可用，并且状态粘滞位模式被强制关闭。在该模式下，该控制寄存器不受任何影响。您仍可以使用该计数器，但对它有一定的限制。在该模式下，不能使能计数器的动态输入（LD/EN）。

图 16-38. 同步模式



16.2.3.6 状态和控制时序

状态和控制寄存器需要选择某个时钟，用于执行以下的工作模式：

- 状态寄存器，其中将任意位设置为粘滞位，在读取模式下被清除。
- 计数器模式中的控制寄存器。
- 同步模式。

该模式的时钟在复位和时钟控制模块中分配。请参见第 190 页上的 “ 复位和时钟控制模块 ” 一节。

16.2.3.7 辅助控制寄存器

读 / 写辅助控制寄存器是一个特殊的寄存器，可用于控制 UDB 中固定的功能硬件。通过该寄存器，CPU 可以动态控制中断、FIFO 和计数器操作。寄存器位和其说明如下所示。

辅助控制寄存器							
7	6	5	4	3	2	1	0
		CNT START	INT EN	FIFO1 LVL	FIFO0 LVL	FIFO1 CLR	FIFO0 CLR

FIFO0 清除， FIFO1 清除

FIFO0 CLR 和 FIFO1 CLR 位用于复位相应 FIFO 的状态。将 ‘1’ 写入到这些位后，会清除相应 FIFO 的状态。必须将这些位回写为 ‘0’，以持续执行 FIFO 操作。当这些位被激活时， FIFO 会作为无状态的简单单字节缓冲区运行。

FIFO0 电平， FIFO1 电平

FIFO0 LVL 和 FIFO1 LVL 位控制 4 字节 FIFO 确认总线状态被激活的等级 （总线可以对 FIFO 进行读或写操作）。 FIFO 总线状态的意义取决于所配置的方向，如表 16-20 中所示。

表 16-20. FIFO 电平控制位

FIFOx LVL	输入模式 （总线正在写入 FIFO）	输出模式 （总线正在读取 FIFO）
0	未满 至少可以写入一个字节	非空 至少可以读取一个字节
1	至少一半为空 至少可以写入两个字节	至少一半为满 至少可以读取两个字节

中断使能

当状态寄存器的中断生成逻辑被使能时，INT EN 位将关断所引起的中断信号。

计数启动

CNT START 位可用于使能或禁用计数器 （仅在 SC_OUT_CTL[1:0] 位被配置为计数器输出模式使用时，该位才有效）。

16.2.3.8 状态和控制寄存器总结

表 16-21 总结了状态和控制寄存器的功能。请注意，寄存器可以是控制和掩码寄存器，也可以是计数和周期寄存器，具体情况取决于 UDB 的工作模式。

表 16-21. 状态、控制寄存器的功能汇总

模式	控制 / 计数	状态 / 同步	掩码 / 周期
控制	控制输出	状态输入或同步	状态掩码
计数	计数输出		计数周期[a]
状态	控制输出或计数输出	状态输入	状态掩码
同步		SYNC	NA[b]

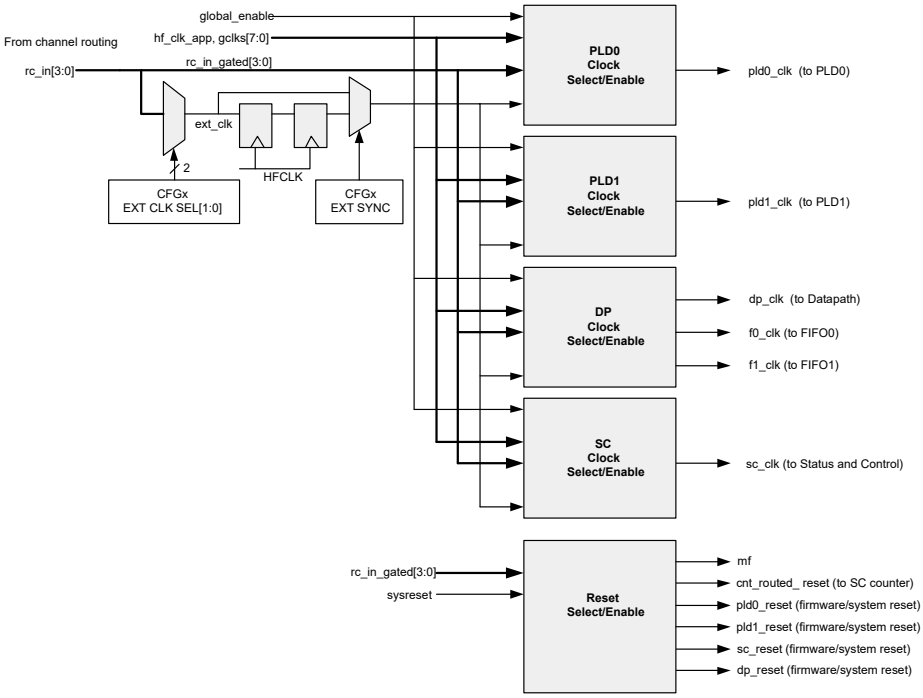
a. 在计数器模式下，掩码寄存器作为周期寄存器运行，不能执行掩码寄存器的功能。因此，计数器模式被使能时，中断输出不可用。
b. 在同步模式下，不能使用状态寄存器，因此掩码寄存器也不可用。但它可作为周期寄存器按计数模式使用。

16.2.4 复位和时钟控制模块

复位和时钟模块主要用于从可用的全局系统时钟或 HFCLK 为每个 PLD、数据路径及状态和控制模块中选择一个时钟。它还为 UDB 模块提供动态以及基于固件的复位。如图 16-39 所示，共有四个时钟控制模块和一个复位模块。通过路由矩阵（RC_IN[3:0]）可以使用四个输入。每个时钟控制模块可以从这些路由输入中选择一个时钟使能源。另外，通过一个复用器可以选择路由输入中的一个作为外部时钟源使用。如该图所示，可以选择性地同步外部时钟源选择。共有六个供给每个 UDB 组件使用的时钟：四个 UDB 外部时钟、一个 HFCLK 和一个所选的外部时钟 （ext clk）。可以将任意路由输入信号 （rc_in）作为电平敏感型或边沿敏感型使能使用。该模块的复位功能为 PLD 模块和 SC 计数器提供了路由复位，并且为每个模块提供了固件复位能力，以此支持重新配置。

复位和时钟控制的 HFCLK 输入不同于系统 HFCLK。之所以将该时钟称为 “hf_clk_app”，是因为关闭它的方式与其他 UDB 外设时钟相类似，并且它还用于 UDB 操作。系统 HFCLK 仅适用于 I/O 访问操作，并且进行访问操作时它被自动关断。 数据路径时钟生成器生成了 3 个时钟：一个用于数据路径，另外两个则用于 FIFO （每个 FIFO 使用一个时钟）。

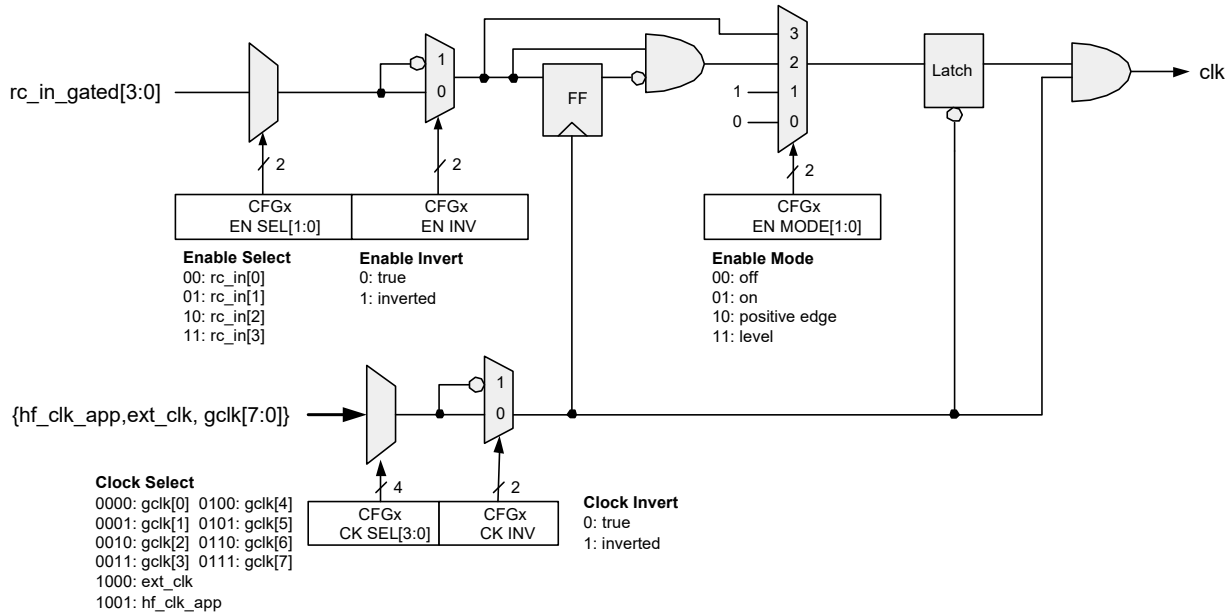
图 16-39. 复位和时钟控制



16.2.4.1 时钟控制

图 16-40 显示的是时钟选择和使能电路的一个实例。每个 UDB 包含了四个电路，其中两个用于 PLD 模块 （每个 PLD 使用一个）、一个用于数据路径，另外一个用于状态和控制模块。该电路的主要组件包括：全局时钟选择复用器、时钟反转、时钟使能选择复用器、时钟使能反转和边沿检测逻辑。

图 16-40. 时钟选择 / 使能控制



时钟选择

四个 UDB 外设时钟 （请参见第 83 页上的时钟系统章节）（gclk[0] 至 gclk[3]）被路由给所有 UDB ； 剩余的四个时钟（gclk[4] 至 gclk[7]）在 PSoC 4 器件系列中不受支持。可以从这些时钟选择一个。 UDB 外设时钟是用户选择时钟分频器的输出。另外，还可以选择 HFCLK，在系统中该时钟的频率最高。被称为 “hf_clk_app” 的信号可从系统 HFCLK 单独路由得到。另外，可以选择外部路由信号作为时钟输入使用，以支持直接提供时钟脉冲的功能（如 SPI）。由于应用功能被映射到各 UDB 上任意边缘，所以每个 UDB 子组件模块的单独时钟选择支持高精度编程。

时钟反转

可以选择性地反转所选时钟。由于存在半周期时序路径，因此最大工作频率受限。当内部时钟被反转，并且其频率与 HFCLK 相同时，同时总线写操作和内部写操作（例如计数器正在计数时写入一个新的计数值）都不受支持。该限制会影响 A0、 A1、 D0、 D1 和处于计数器模式的控制寄存器。

时钟使能选择

可以将该时钟使能信号路由到任意同步信号，并且可以从路由矩阵上四个输入中选择一个 （这些输入均用于此模块）。

时钟使能反转

可以选择性地反转时钟使能信号。通过该特性，可以通过任何极性来使能生成时钟。

时钟使能模式

默认情况下，该时钟使能处于关闭状态。配置目标模块操作后，通过使用 UDB CFG24 寄存器中的 EN MODE[1:0] 位，软件可以将该模式设置为以下模式，如图 16-40 所示。

表 16-22. UDB CFG24 寄存器中的时钟使能模式

时钟使能模式	说明
OFF	时钟被关闭。
ON	时钟被打开。选定的全局时钟可以自由运行。
上升沿	检测到时钟使能输入的上升沿时，将生成门控时钟。使能输入的最大频率是选定的全局时钟的二分频。
电平	当时钟使能输入为高电平 （‘1’）时，将生成时钟。

时钟使能的使用情况

时钟使能信号通常使用于以下两种情况：

固件使能 — 假设几乎所有功能都需要一个固件时钟使能，用于启动和停止功能。由于映射到 UDB 矩阵的功能边界是任意的，因此它可以跨多个 UDB 和 / 或 UDB 部分；必须有自动使能给定功能的方式。该情况通常通过控制寄存器中的某一位实现（该寄存器被路由至一个或多个时钟使能输入）。该情况还支持应用要求同时使能多个无关模块的场合。

仿真局部时钟生成 — 通过该特性，局部时钟会由 UDB 生成，并通过使用同步时钟使能方案（而不是由一个 UDB 直接给另一个提供时钟）分配到阵列中的其他 UDB。使用时钟使能模式的上升沿特性可以消除时钟使能波形占空比的限制。

特殊的 FIFO 时序

数据路径 FIFO 具有特殊的时序注意事项。默认情况下，FIFO 时钟和数据路径时钟的配置相同。但 FIFO 还包含了各个特殊的控制位，用于修改时钟配置：

- 根据选定的数据路径时钟极性，可以反转 FIFO 时钟。
- 如果在 UDB CFG16 寄存器中设置了 FIFO FAST 模式，那么 HFCLK 将覆盖 FIFO 通常使用的数据路径时钟选择。

16.2.4.2 复位控制

复位控制具有两种模式，分别为：兼容模式和备用模式。这两种模式由每个 UDB CFG31 寄存器中的 ALT RES 位控制。当该位为 ‘0’ 时，可实现兼容模式。该位为 ‘1’ 时，则实现备用模式。

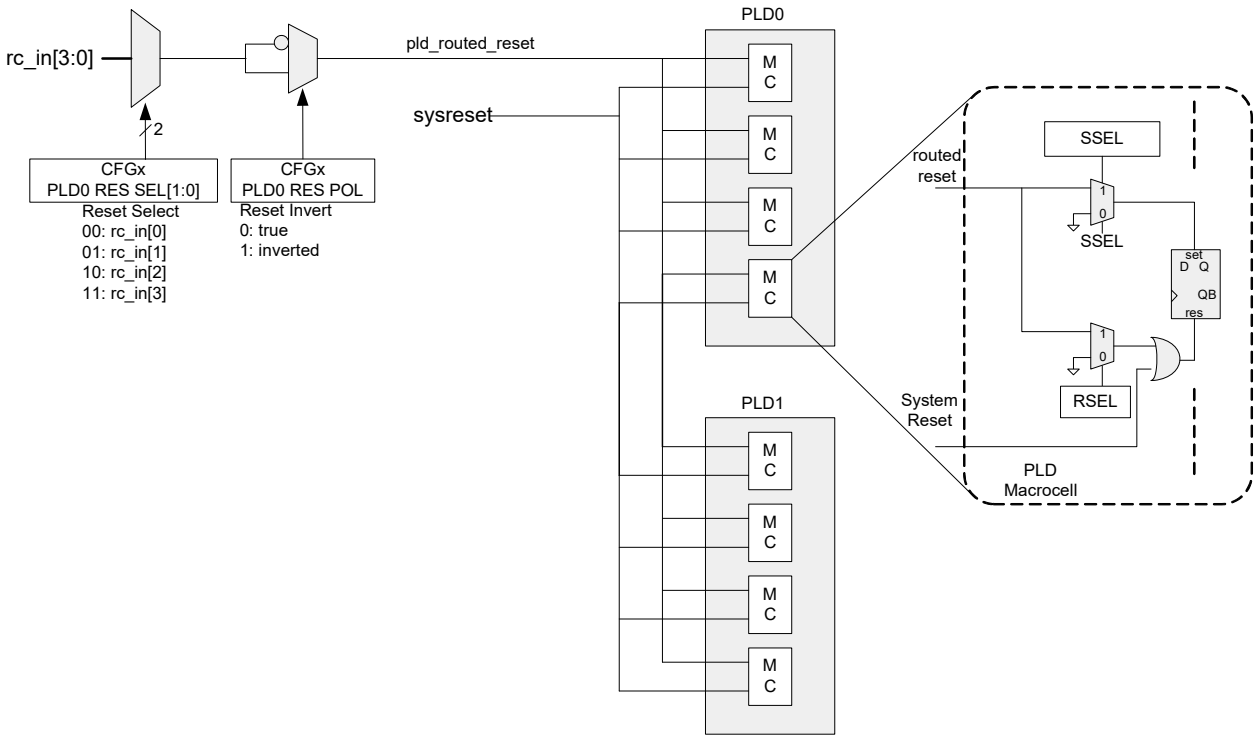
兼容复位方案

该方案包括一个路由复位，用于动态复位模块的嵌入式状态（该状态适用于各 PLD 宏单元和 SC 计数器）。

兼容 PLD 复位控制

图 16-41 显示的是使用路由动态复位的兼容 PLD 复位系统。

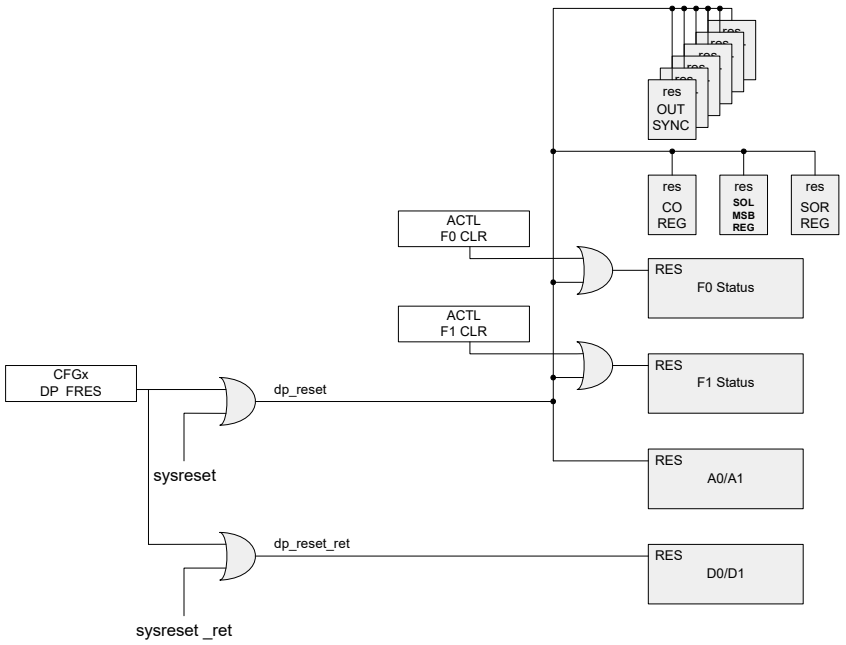
图 16-41. 兼容 PLD 复位结构



兼容的数据路径复位控制

图 16-42 显示的是使用固件复位的兼容数据路径复位系统。固件复位会异步清除 DP 输出寄存器、进位和移出标志、 FIFO 状态、累加器以及数据寄存器。请注意， D0 和 D1 寄存器作为保持寄存器使用，它们在睡眠间隔中会保持其状态。 FIFO 数据是未知的，因为它是基于 RAM 的数据。

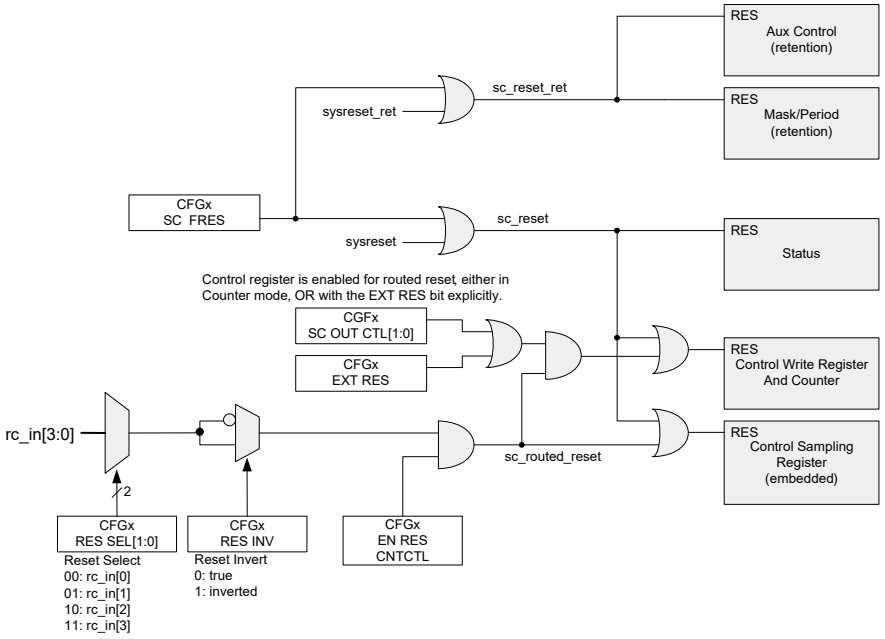
图 16-42. 兼容的数据路径复位结构



兼容的状态和控制复位控制

图 16-43 显示了兼容的状态和控制模块复位。掩码 / 周期和辅助控制寄存器都是保持寄存器。

图 16-43. 兼容的状态和控制复位控制



备用复位方案

表 16-23 显示了兼容复位方案和备用复位方案间的差异。

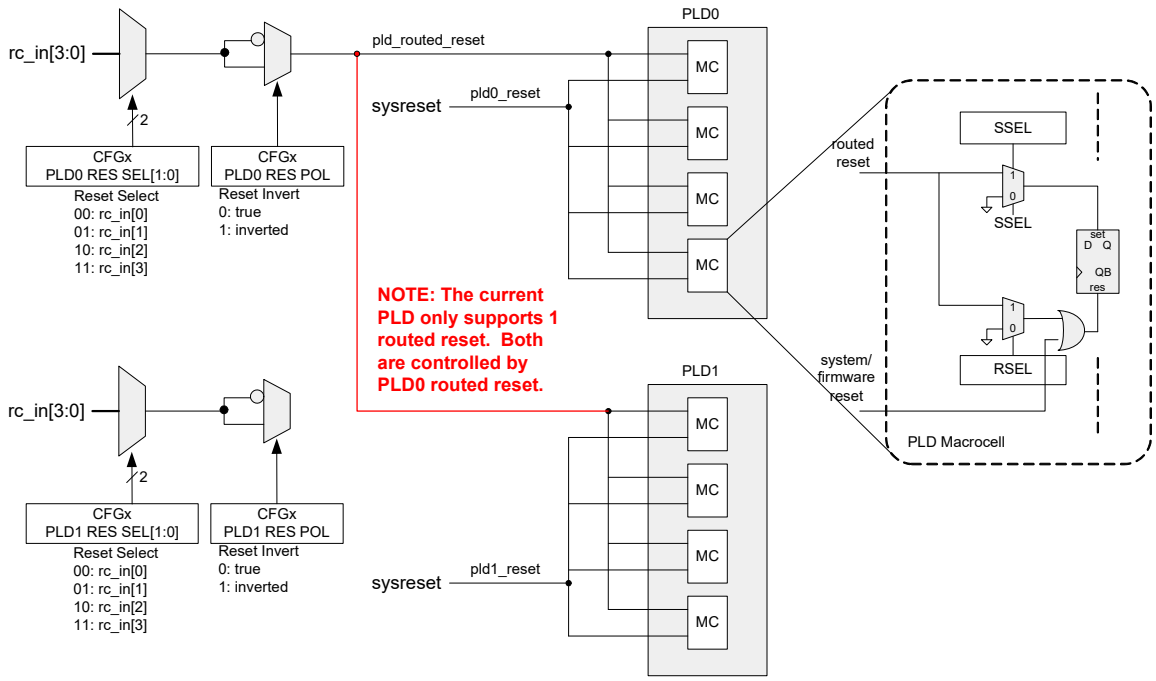
表 16-23. 复位方案

特性	兼容	备用
精度	UDB 中的所有模块共用了一个路由复位	每个 UDB 组件模块都可以选择一个单独的复位
状态寄存器	没有路由复位功能	可以使用已选定的 SC 路由复位
数据路径	没有路由复位功能	可选的，可以使用已选定的 DP 路由复位

备用的 PLD 复位控制

图 16-44 显示了备用的 PLD 复位系统。虽然每个 PLD 都有自己的用于单独复位的规定，但 PLD 模块不支持该规定。因此，在备用复位方案中， PLD0 复位控制设置适用于两个 PLD。

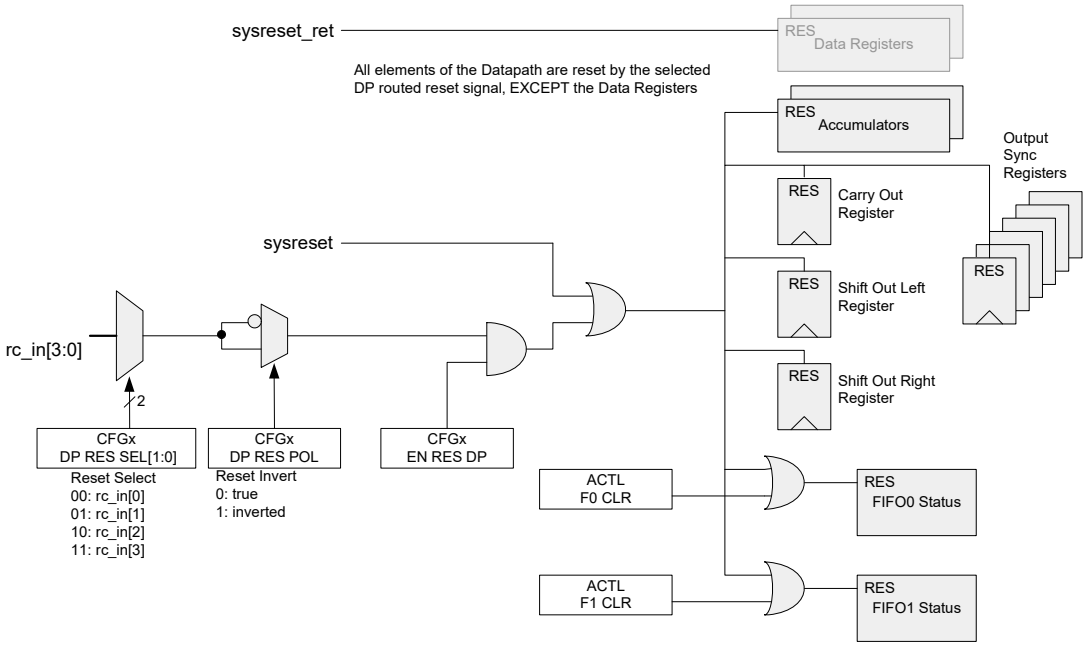
图 16-44. 备用的 PLD 复位结构



备用的数据路径复位控制

图 16-45 显示的是备用的数据路径复位系统。数据路径的路由复位适用于所有数据路径状态，作为保持寄存器实现的数据寄存器除外。

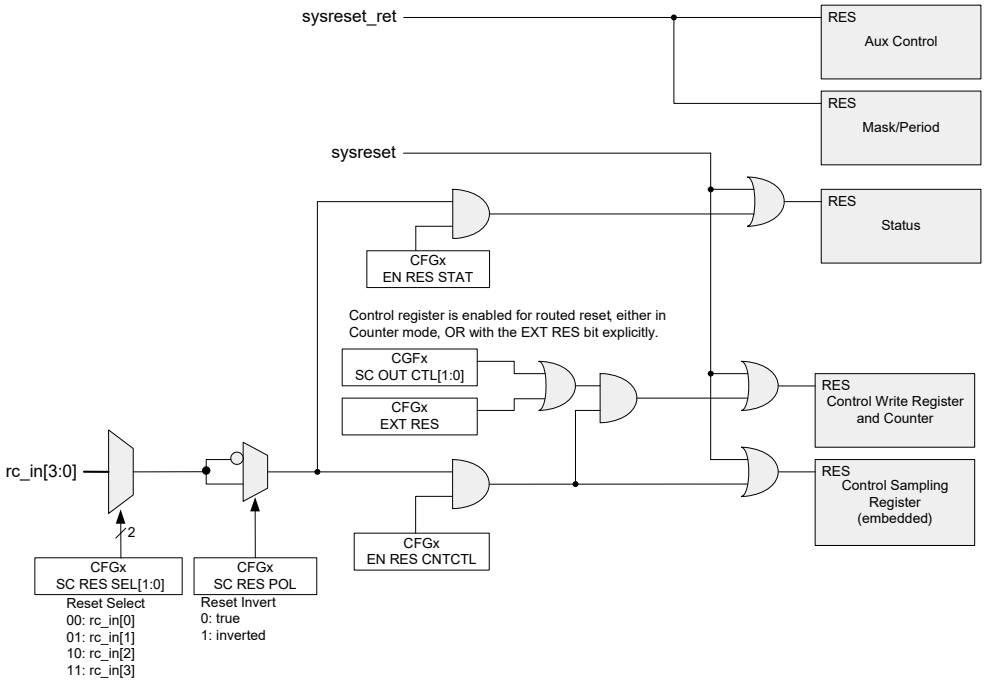
图 16-45. 备用的数据路径复位结构



备用的状态和控制复位控制

图 16-46 显示的是备用的状态和控制模块复位。掩码 / 周期和辅助控制寄存器都是保持寄存器。

图 16-46. 备用状态和控制复位控制



16.2.4.3 UDB POR 初始化

寄存器和状态初始化

表 16-24. UDB POR 状态初始化

状态元素	状态元素	POR 状态
配置锁存	CFG 0 – 31	0
Ax、 Dx、 CTL、 ACTL、 MASK	累加器、数据寄存器、辅助控制寄存器、掩码寄存器	0
ST、宏单元	只读状态和宏单元寄存器	0
DP CFG RAM 和 Fx （FIFO）	数据路径配置 RAM 和 FIFO RAM	未知
PLD RAM	PLD 配置 RAM	未知

路由初始化

上电复位 （POR）时，输入和输出路由的状态如下：

- 从 UDB 驱动到路由矩阵中的所有输出均保持为 ‘0’ 。
- 从路由驱动到 UDB 输入内的所有输出都为 ‘0’ 。

这样可以避免路由中发生的冲突驱动状态，而且初始配置会按独立序列依次发生。

16.2.5 UDB 寻址

可以通过多个地址空间对 UDB 进行访问，例如，工作寄存器（A0、A1、D0、D1、FIFO 等）和配置寄存器的 8 位、16 位和 32 位访问。

- 8 位工作寄存器 — 通过该地址空间可以访问 UDB 中的单个工作寄存器。
- 16位连续工作寄存器 — 通过该地址空间可以访问两个连续UDB中的同一个工作寄存器，例如UDB n的D0和UDB n+ 1的D0
- 已配对的 16 位工作寄存器 — 通过该地址空间可以访问同一个 UDB 中的两个工作寄存器，如 A0 和 A1。
- 32 位工作寄存器 — 通过该地址空间可以访问四个 UDB 中相同的工作寄存器，如 A1。
- 8 位、 16 位或 32 位配置寄存器 — 通过该地址空间可以访问单个 UDB 中的配置寄存器。

16.2.6 系统总线访问一致性

UDB 寄存器具有两种访问模式：

- 系统总线访问模式，在该模式下 CPU 对 UDB 寄存器进行读或写操作。
- UDB 内部访问模式，在该模式下 UDB 函数会更新或使用寄存器的内容。

16.2.6.1 同时进行系统总线访问

表 16-25 列出了可能同时访问的事件和所需的行为：

表 16-25. 同时进行的系统总线访问

寄存器	UDB 写入 总线写入	UDB 写入 总线读取	UDB 读取 总线写入	UDB 读取 总线读取
Ax	未定义结果	不允许直接进行[a、b]	UDB 读取先前的数值	当前值通过这两个操作读取
Dx				
Fx	不受支持（UDB 和总线必须为相反访问）	如果使用 FIFO 状态标志，在同一个位置上不能同时进行读 / 写操作		不受支持（UDB 和总线必须为相反访问）
ST	NA，总线不进行写操作	总线读取先前的数值	NA， UDB 不进行读操作	
CTL	NA， UDB 不进行写操作		UDB 读取先前的数值	当前值通过这两个操作读取
CNT	未定义结果	不允许直接进行[c]		
ACTL	NA， UDB 不进行写操作			
掩码				
PER				
宏单元 （RO）	NA，总线不进行写操作	不允许直接进行[d]	NA，总线不进行写操作	

a. 通过使用 FIFO 的软件捕获特性，可以安全读取 Ax 寄存器。
b. 只有 FIFO 才能动态写入到 Dx 寄存器内。编程该模式时，不允许直接读取 Dx 寄存器。
c. 只有 CNT 寄存器被禁用时，才能安全读取它的数据。另外，通过将输出路由到 SC 寄存器 （透传模式），也可以动态读取 CNT 值。
d. 通过将宏单元寄存器位路由到状态寄存器 （透传模式），也可以进行安全读取操作。

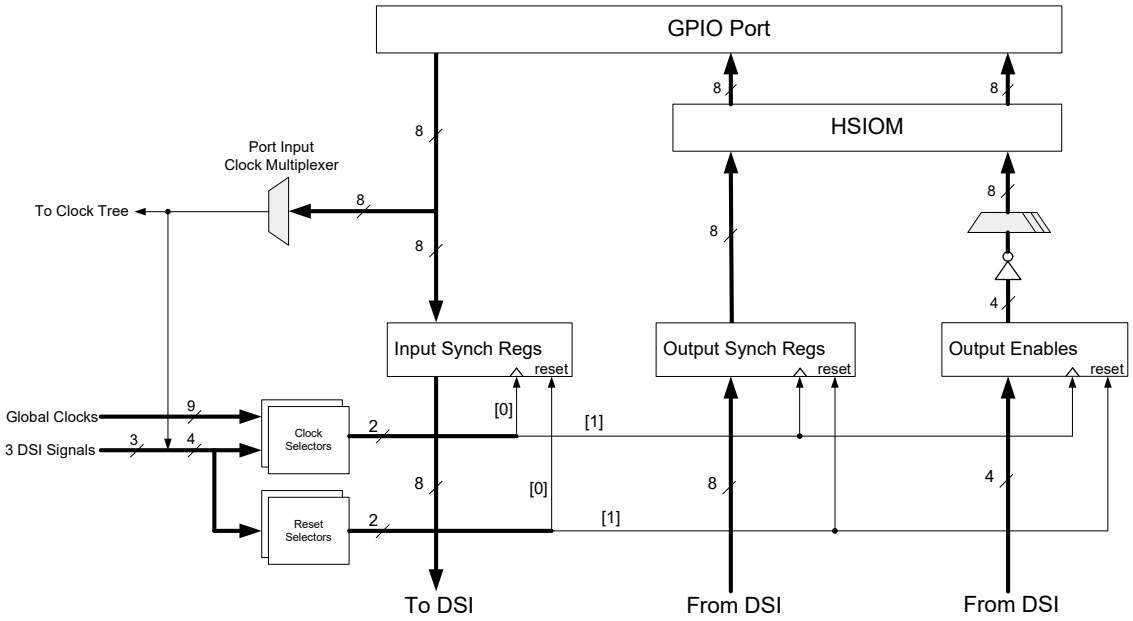
16.2.6.2 一致性累加器访问 （原子读和写操作）

UDB 累加器是数据计算的主要目标。因此，在正常操作期间直接读取这些寄存器会得到未定义的结果，如表 16-25 所示。然而，原子读操作的内置支持 （其格式为软件捕获）在各链接模块上实现。在该使用模型中，对最低有效累加器的读取操作会将所有链接模块中的数据传输到相应的 FIFO 内。通过编程方式可以对累加器进行原子写操作。可以对输入 FIFO 单独执行写操作，然后将最后一次写入 FIFO 内的状态信号路由到所有相关模块，同时将 FIFO 数据传输到 Dx 或 Ax 寄存器内。

16.3 端口适配器模块

端口适配器模块扩展了 UDB，以便通过高速 I/O 矩阵 （HSIOM）连接到 GPIO，如第 77 页上的 “ 高速 I/O 矩阵 ” 一节中所介绍。 HSIOM 对各寄存器进行放置，以便将 DSI 信号快速路由到 GPIO 输出和输出使能。 HSIOM 还允许多个模块共用 GPIO，例如，端口数据寄存器和外设 （如 I2C）。图 16-47 显示的是一个顶层视图。

图 16-47. 端口适配器框图



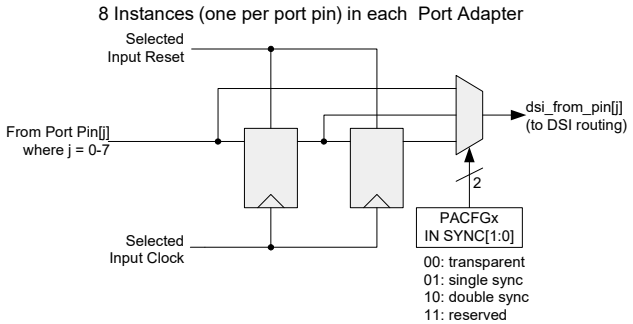
每个 8 位 GPIO 端口都有一个端口适配器 （PA）。 GPIO 输入数据具有 8 个输入， GPIO 输出数据具有 8 个输出，以及 8 个输出使能 （OE）连接。 PA 中的寄存器用于同步输入、输出和输出使能。另一个特性是端口输入时钟复用器。通过该复用器可以选择作为时钟使用的一个端口输入。该时钟可以在 PA 中局部使用并路由至全局时钟 （请参见第 83 页上的时钟系统章节）。

两个可编程的时钟选择器均可用，用以为输入和输出同步寄存器提供单独的时钟。 OE 寄存器和输出寄存器使用同一个时钟。另外，与可编程的时钟选择器相同，两个可编程的复位选择器均可用。

16.3.1 PA 数据输入逻辑

图 16-48 显示的是数据输入逻辑的结构。各输入来自输入 / 输出端口上的每一个引脚。可以对该信号进行单同步或双同步，或者旁路同步过程，以得到异步输入。对选定的端口输入时钟进行同步。该电路的输出连接到 DSI 路由。

图 16-48. GPIO 输入逻辑的详细内容



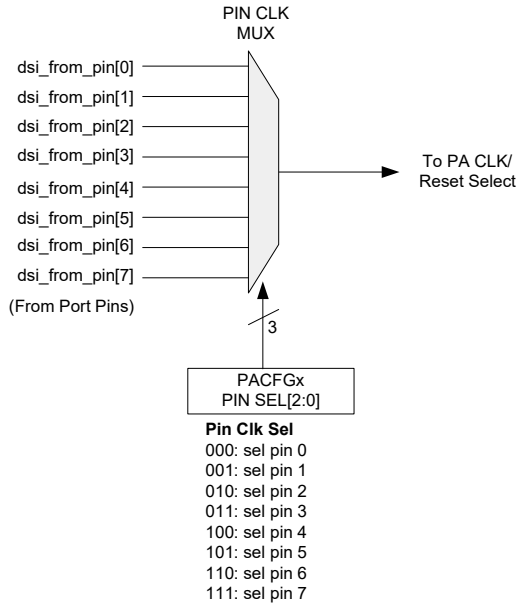
16.3.2 PA 端口引脚的时钟复用器逻辑

图 16-49 显示的是端口引脚复用器。每个端口具有八个数据输入信号，其中一个被作为时钟使用。该时钟被路由，以作为：

- 端口适配器中的可编程时钟
- UDB 时钟树源
- 端口适配器中的可编程复位
- 作为端口适配器中的时钟使能使用。

请注意，选定的信号不能通过同步器传送，并与模块中的其他时钟域异步。将该信号用于各个选定功能时，必须十分慎重。

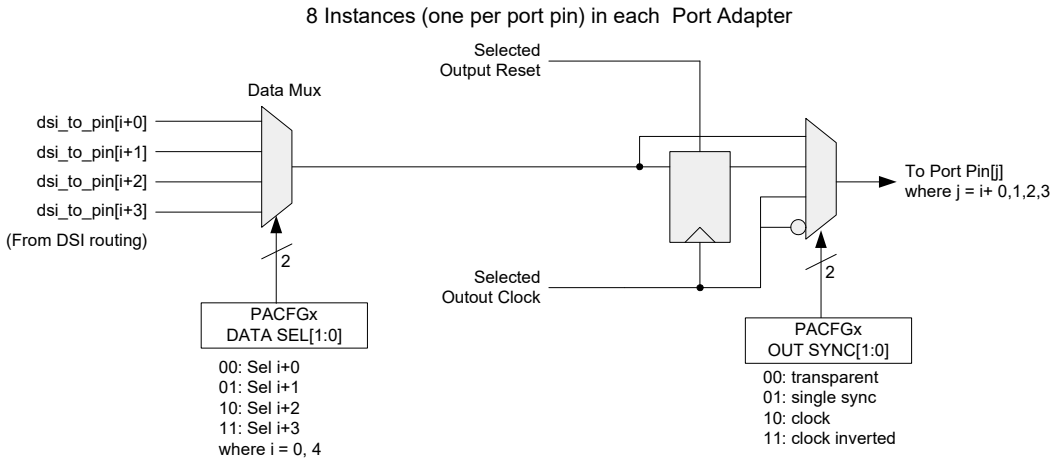
图 16-49. GPIO 引脚选择的详细内容



16.3.3 PA 数据输出逻辑

图 16-50 显示的是数据输出逻辑的结构。输出（通过 HSIOM）被发送给 I/O 端口上的各个引脚。可以对该信号进行单同步，或者旁路同步过程，以得到异步输出。通过其他选项可以输出所选的时钟或输出时钟的反转版本。

图 16-50. GPIO 输出数据逻辑的详细内容



16.3.4 PA 输出使能逻辑

图 16-51 显示的是输出使能 （OE）逻辑。该电路和数据输出共用了相关的时钟和复位。该连接是唯一的，就是说四个 DSI 输出与 OE 相连，但这些输出被复用到 I/O 端口引脚上的四个 OE 连接，如图 16-52 所示。

图 16-51. GPIO 输出使能 （OE）同步逻辑

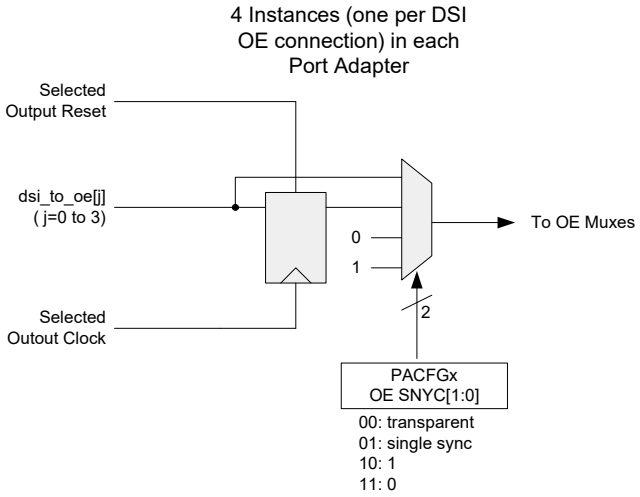
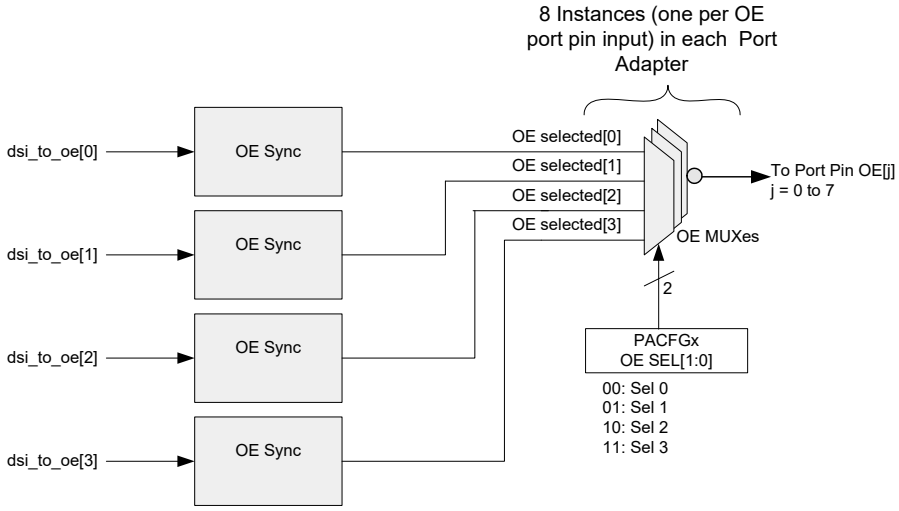


图 16-52. GPIO 输出使能 （OE）复用器

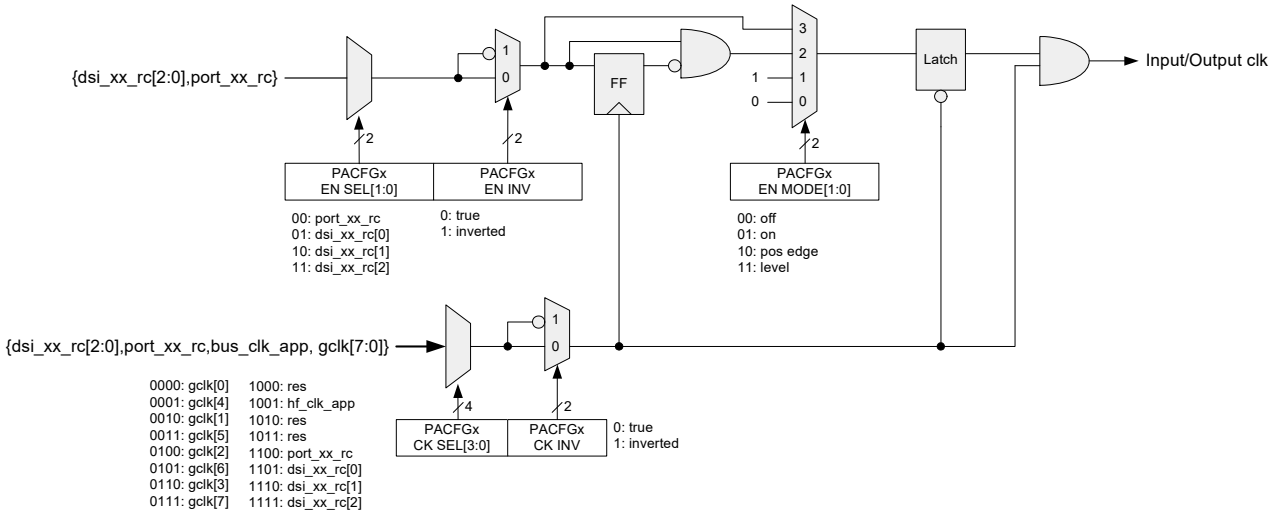


请注意，由于 OE 信号在各端口上为低电平有效，因此在 OE 同步逻辑和 OE 复用器之间的路径中存在一个附加的反转。

16.3.5 PA 时钟复用器

图 16-53 显示的是 PA 时钟复用器的结构。如上面所述，每个 PA 都有两个可编程的时钟选择器，用于为端口输入、输出和输出使能 （OE）提供单独的时钟。

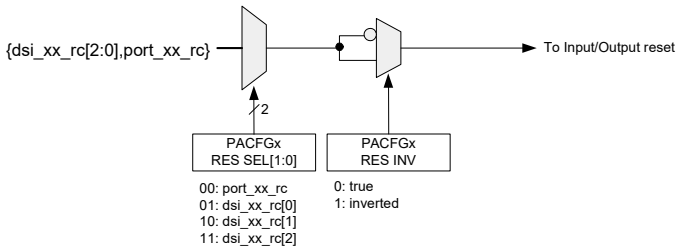
图 16-53. PA 时钟复用器的详细内容



16.3.6 PA 复位复用器

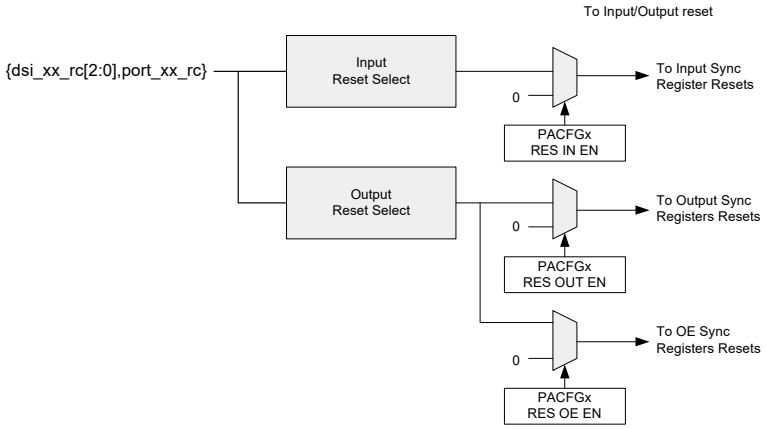
PA 复位复用器的结构如图 16-54 中所示。

图 16-54. PA 复位复用器的详细内容



如图 16-55 所示，复位选择逻辑被重复，一个用于输入，一个用于控制输出和输出使能。每个复位都有一个单独使能，适用于相应类别中的所有 8 位。

图 16-55. PA 复位系统



17. 定时器、计数器和 PWM

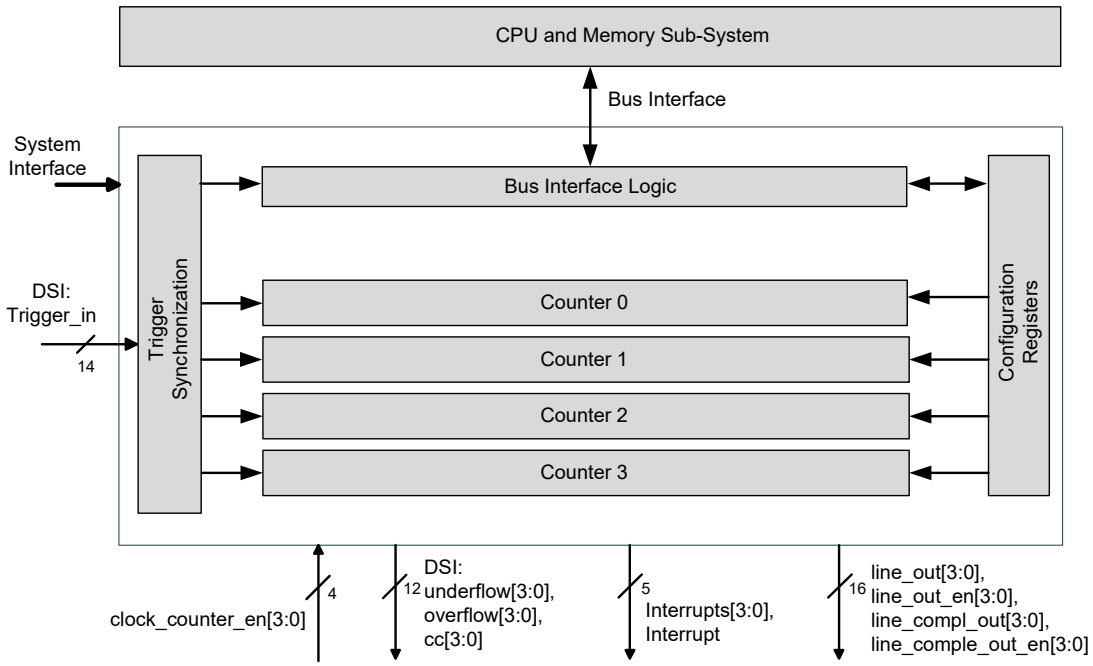
PSoC® 4 中的定时器、计数器和脉宽调制器 （TCPWM）模块能够实现 16 位定时器、计数器、脉宽调制器 （PWM）和正交解码器等功能。该模块用于测量输入信号的周期和脉宽 （定时器），捕获特定事件发生的次数 （计数器），生成 PWM 信号或对正交信号进行解码。本节介绍了 TCPWM 模块的特性、工作原理以及操作模式。

17.1 特性

- 四个 16 位定时器、计数器或脉宽调制器 （PWM）
- TCPWM 模块支持下面各种工作模式：
  - 定时器
  - 捕获
  - 正交解码
  - 脉宽调制
  - 伪随机 PWM
  - 带死区时间的 PWM
- 支持多种计数模式：递增、递减和递增 / 递减
- 时钟预分频 （1、 2、 4、 …、 64、 128 分频）
- 比较 / 捕获值和周期值的双缓冲
- 发生以下条件时将生成中断：
  - 终值计数 — 达到计数器寄存器中的最终值
  - 捕获 / 比较 — 计数值被捕获到捕获 / 比较寄存器或计数器的值等于比较值
- 同步计数器 — 计数器在相同的时间内可重新加载、启动、停止和计数
- 供给每个计数器的 DSI 输出信号，以用于说明下溢、上溢以及捕获 / 比较匹配事件
- PWM 的互补线路输出
- 根据上升沿、下降沿、双边沿和电平触发等选项，可以从 14 个 DSI 信号选择每个 TCPWM 的启动、重载、停止、计数和捕获事件信号

17.2 框图

图 17-1. TCPWM 框图



该模块具有以下接口：

- 总线接口：将模块连接到 CPU 子系统。
- 带有 DSI 的 I/O 信号接口：路由通用数字模（UDB）和 TCPWM 模块的信号。它包括输入触发信号（如：重新加载、启动、停止、计数和捕获）和输出信号（如：上溢（OV），下溢（UN）和捕获/比较（CC））。任意GPIO都可作为输入触发信号使用。
- 中断：根据终止计数（TC）或 CC 条件提供每个计数器的中断请求信号，并提供四个中断请求信号经过逻辑 OR 后生成的组合中断信号。
- 系统接口：包括控制信号，如来自系统资源子系统的时钟和复位信号。

通过对 TCPWM 寄存器进行写操作，可以配置 TCPWM 模块。有关该模块所需的所有寄存器的详细信息，请参见第 226 页上的 “TCPWM 寄存器 ” 一节。

17.2.1 使能和禁用 TCPWM 模块中的计数器

通过设置控制寄存器 TCPWM_CTRL 中的 COUNTER_ENABLED 字段 （位 0），可以使能计数器。

注意：使能计数器前，需要先对它进行配置。否则，各个寄存器将被更新为新的数值。禁用计数器后，寄存器中的值仍被保留，直到再次使能 （或重新配置）该计数器为止。状态寄存器在计数器被禁用后均被清零。

17.2.2 时钟

TCPWM通过系统接口接收HFCLK，用于对模块中所有事件进行同步化。使能计数器时所生成的计数器使能信号（counter_en）对 HFCLK 进行门控，从而为特定计数器提供时钟 （counter_clock）。此外，输出触发信号 （本节中后面部分将介绍）也与 HFCLK 同步。

时钟预分频：可对counter_clock进行预分频，分频值可为1、2、4、…、64、128。通过修改计数器控制（TCPWM_CNT_CTRL）寄存器中的 GENERIC 字段，可以实现该预分频操作，如表 17-1 所示。

表 17-1. 预分频计数器时钟的位字段设置

GENERIC[10:8]	说明
0	1 分频
1	2 分频
2	4 分频
3	8 分频
4	16 分频
5	32 分频
6	64 分频
7	128 分频

注意： 在正交模式和 PWM-DT 模式下，则不能对时钟进行预分频。

17.2.3 基于触发输入的事件

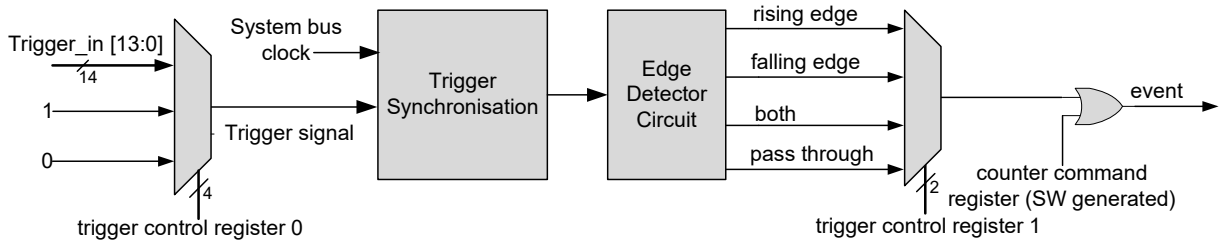
这些事件可由硬件或软件触发。

- 重载
- 启动
- 停止
- 计数
- 捕获 / 切换

硬件可根据信号电平、上升沿、下降沿或双边沿来触发事件。图 17-2 显示的是任何事件触发信号可选择的边沿检测类型。触发控制寄存器0（TCPWM_CNT_TR_CTRL0）选择14个触发输入中的某一个作为事件信号，该输入包含常量‘0’和‘1’信号。

通过配置触发控制寄存器 1（TCPWM_CNT_TR_CTRL1），可以选择边沿（上升沿、下降沿或双边沿）或电平（高电平或低电平）来触发事件。可分别为每个触发事件选择边沿 / 电平配置。另外，通过写入计数器指令寄存器（TCPWM_CMD），固件可以生成一个事件，如图 17-2 所示。

图 17-2. 触发信号的边沿检测



在 TCPWM 模块的各种模式下，由这些触发源引起的事件会有不同的定义。

- **重载：**重载事件初始化并启动计数器。
  - 在递增计数模式下，计数寄存器（TCPWM_CNT_COUNTER）被初始化为‘0’。
  - 在递减计数模式下，计数器使用存储在 TCPWM_CNT_PERIOD 寄存器中的周期值进行初始化。
  - 在递增 / 递减计数模式下，计数寄存器被初始化为 1。
  - 在正交模式下，重载事件作为正交索引事件。索引 / 重载事件表示一个完整的旋转，并且可用于同步化正交解码。
- **启动：**启动事件用于启动计数操作。在发生停止事件后，或在软件将计数寄存器重新初始化为某个值后，可使用该事件。请注意，发生该事件时，计数寄存器不会被初始化。
  - 在正交模式下，启动事件作为正交相位输入 phiB，这在第 215 页上的“正交解码器模式”一节一节中有详细说明。
- **计数：**根据计数器的配置，计数事件使计数器进行递增或递减。
  - 在正交模式下，计数事件作为正交相位输入 phiA。
- **停止：**停止事件用于停止计数器的递增或递减操作。启动事件会重新启动计数操作。
  - 在脉宽调制模式中，停止事件作为终止事件。终止事件会禁用所有 PWM 输出信号线。
- **捕获：**捕获事件会将计数寄存器中的值复制到捕获寄存器内，并将捕获寄存器中的值复制到缓冲捕获寄存器内。在 PWM 模式下，捕获事件作为切换事件。它将捕获 / 比较寄存器和周期寄存器中的值切换到缓冲寄存器内的相应数值。通过该特性，可调制脉冲宽度和频率。

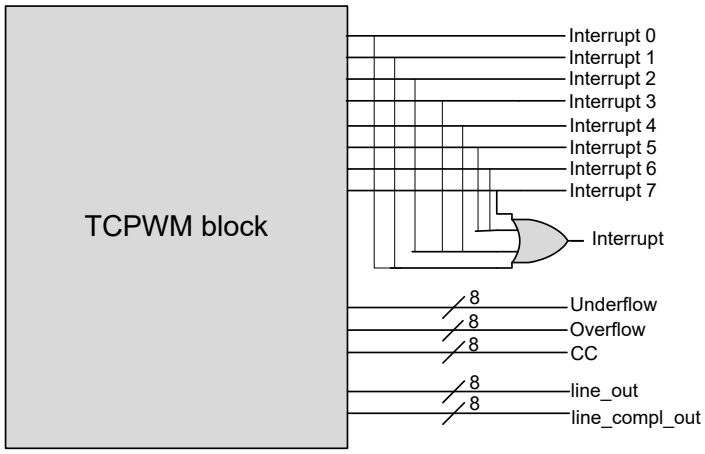
注意：

- 所有触发输入会与 HFCLK 同步。
- 在正交模式下，边沿检测由计数器时钟执行的。在其他 5 种模式下，可以使用 HFCLK 门控进行边沿检测。

17.2.4 输出信号

TCPWM 模块生成几种输出信号，如图 17-3 所示。

图 17-3. TCPWM 输出信号



17.2.4.1 发生触发条件时的信号

- 当计数器进行递增，并且计数寄存器达到周期值时，计数器会生成内部上溢 （OV）条件。
- 当计数器进行递减，并且计数寄存器达到零值时，计数器会生成内部下溢 （UN）条件。
- 当计数器正在运行并发生下面任何一个条件时， TCPWM 都会生成捕获 / 比较 （CC）条件：
  - 计数器值等于比较值。
  - 发生捕获事件 — 发生捕获事件时， TCPWM_CNT_COUNTER 寄存器中的值将被复制到捕获寄存器内，然后捕获寄存器中的值将被复制到缓冲捕获寄存器内。

注意：上述条件发生时，这些信号将保持为逻辑高电平一个 HFCLK 周期。为了保证运行过程可靠，该触发事件的信号频率应该小于 HFCLK 频率的 1/4。例如，如果 HFCLK 的工作频率为 24 MHz，则触发事件的信号频率需要小于 6 MHz。

17.2.4.2 中断

TCPWM 模块从计数器提供一个专用的中断输出信号。在发生 TC 条件或 CC 条件时可以生成中断。这些条件的准确定义因特定模式不同而异。此外，还对来自四个 TCPWM 中的所有四个中断输出信号进行 “OR” 计算，从而生产单一中断输出信号。

四个寄存器用于处理该模块中的中断，如表 17-2 所示。

表 17-2. 中断寄存器

中断寄存器	位	名称	说明
TCPWM_CNT_INTR （中断请求寄存器）	0	TC	当检测到终端计数时，该位被设置为 ‘1’ 。写入 ‘1’ 可清零该位。
	1	CC_MATCH	当计数值与捕获 / 比较寄存器中的值相匹配时，该位将被置为 ‘1’ 。写入 ‘1’ 可清零该位。
TCPWM_CNT_INTR_SET （中断设置请求寄存器）	0	TC	写入 ‘1’ 可以设置中断请求寄存器中的相对应位。读取该寄存器时，它将反映中断请求寄存器的状态。
	1	CC_MATCH	写入 ‘1’ 可以设置中断请求寄存器中的相对应位。读取该寄存器时，它将反映中断请求寄存器的状态。
TCPWM_CNT_INTR_MASK （中断屏蔽寄存器）	0	TC	同中断请求寄存器中 TC 位相对应的屏蔽位。
	1	CC_MATCH	同中断请求寄存器中 CC_MATCH 位相对应的屏蔽位。
TCPWM_CNT_INTR_MASKED （中断屏蔽请求寄存器）	0	TC	相对应的 TC 请求和屏蔽位的逻辑 AND 运算。
	1	CC_MATCH	相对应的 CC_MATCH 请求和屏蔽位的逻辑 AND 运算。

17.2.4.3 输出

TCPWM 有两个输出信号，line_out 和 line_compl_out（ line_out 的互补）。请注意，如果需要，通过配置 TCPWM_CNT_TR_CTRL2 寄存器，可使用 OV、 UN 和 CC 条件驱动 line_out 和 line_compl_out （请查看表 17-3）。

表 17-3. OV、 UN 和 CC 条件的输出信号线配置

字段	位	数值	事件	说明
CC_MATCH_MODE 默认值为 3	1:0	0	将 line_out 设置为 ‘1’	在发生比较匹配（CC）事件时配置输出信号线
		1	将 line_out 清零	
		2	反转 line_out	
		3	保持不变	
OVERFLOW_MODE 默认值为 3	3:2	0	将 line_out 设置为 ‘1’	在发生上溢 （OV）事件时配置输出信号线
		1	将 line_out 清零	
		2	反转 line_out	
		3	保持不变	
UNDERFLOW_MODE默认值为3	5:4	0	将 line_out 设置为 ‘1’	在发生下溢 （UN）事件时配置输出信号线
		1	将 line_out 清零	
		2	反转 line_out	
		3	保持不变	

17.2.5 功耗模式

TCPWM 模块可在活动模式和睡眠模式下工作。 TCPWM 模块由 V_{CCD} 供电。在深度睡眠模式下，配置寄存器和其他逻辑仍被供电，以保持配置寄存器的状态。更多详细信息，请参考表 17-4。

表 17-4. TCPWM 模块的功耗模式

功耗模式	模块状态
活动	在活动模式下，该模块正常工作，时钟正常运行且电源打开。
睡眠	所有计数器时钟均可用，但不能访问总线接口。
深度睡眠	在这种模式下，仍为该模块供电，但不提供总线时钟，因此逻辑无法工作。所有配置寄存器将保持其状态。
休眠	在这种模式下，该模块的电源被关闭。不能保持配置寄存器的状态。
停止	在这种模式下，该模块的电源被关闭。不能保持配置寄存器的状态。

17.3 各种操作模式

计数器模块支持六种操作模式，如表 17-5 所示。计数器控制寄存器 （TCPWM_CNTx_CTRL）中的 MODE [26:24] 字段用于将计数器配置为特定的操作模式。

表 17-5. 工作模式配置

模式	MODE 位字段 [26:24]	说明
定时器	000	实现一个定时器或计数器。在每个计数器时钟周期内，如果检测到计数事件，计数器就加 ‘1’ 或减 ‘1’。
捕获	010	使用捕获输入实现定时器或计数器。在每个计数器时钟周期内，如果检测到计数事件，计数器就加 ‘1’ 或减 ‘1’。如果发生捕获事件，计数器值将被复制到捕获寄存器中。
正交解码器	011	根据选定的（X1、X2 或 X4）编码结构的两个相位输入执行正交解码器。执行正交解码器时可使计数器递增或递减计数。
PWM	100	通过 8 位时钟预分频器和缓冲比较 / 周期寄存器实现边沿 / 中心对齐 PWM。
PWM-DT	101	通过可配置的 8 位死区时间 （在两个输出上）和缓冲比较 / 周期寄存器实现边沿 / 中心对齐 PWM。
PWM-PR	110	使用一个 16 位线性反馈移位寄存器 （LFSR）产生一个伪随机 PWM。

通过设置 TCPWM_CNT_CTRL 寄存器中的 UP_DOWN_MODE[17:16] 字段，可将计数器配置为递增、递减或递增 / 递减计数模式，如表 17-6 所示。

表 17-6. 计数模式配置

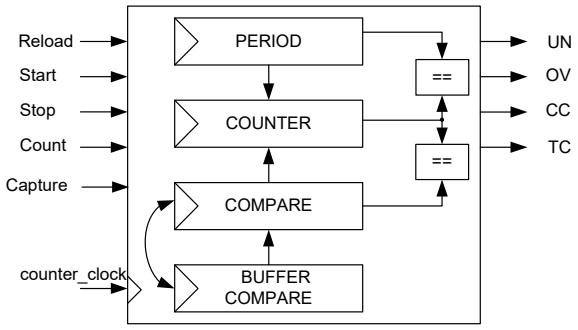
计数模式	UP_DOWN_MODE[17:16]	说明
递增计数模式	00	计数器递增，直到达到周期值为止。当计数器达到周期值时，将生成终值计数 （TC）条件。
递减计数模式	01	计数器从周期值开始递减，直到数值等于 0 为止。当计数器的值为 0 时，将生成 TC 状态。
递增 / 递减计数模式 0	10	计数器递增，达到周期值时开始递减，直到达到 0 为止。只有达到 0 时，才会生成 TC 状态。
递增 / 递减计数模式 1	11	与递增 / 递减计数模式 0 相似，但在计数器达到 0 或周期值时，都会生成一个 TC 条件。

17.3.1 定时器模式

定时器模式通常用于测量某个事件发生的时长或测量两个事件间的时间差。

17.3.1.1 框图

图 17-4. 定时器模块框图



17.3.1.2 工作原理

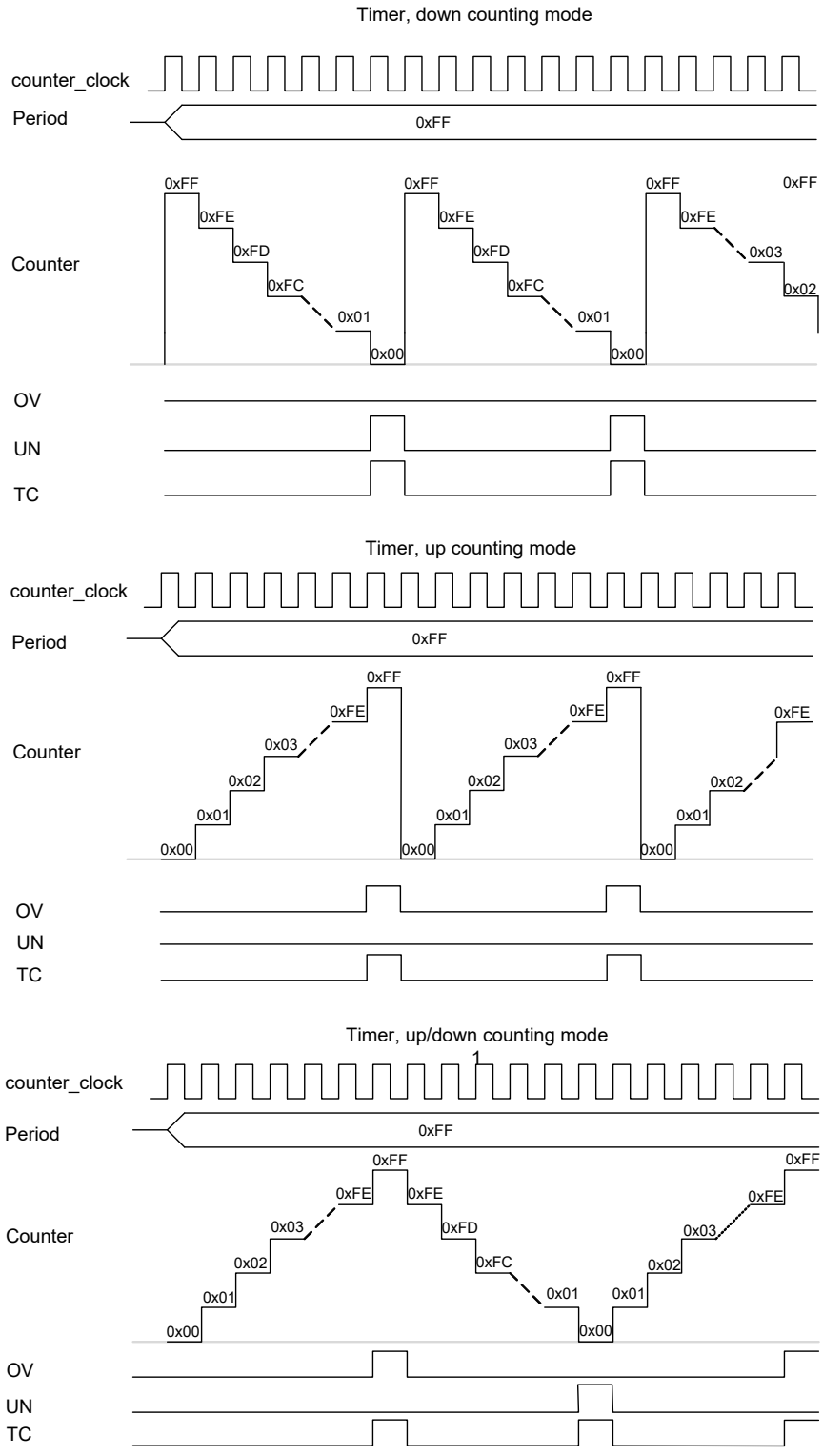
可以配置定时器，使之能够在递增、递减或递增 / 递减计数模式下进行计数。也可以将该模块配置为连续模式或单触发模式。下面介绍的是定时器的工作原理：

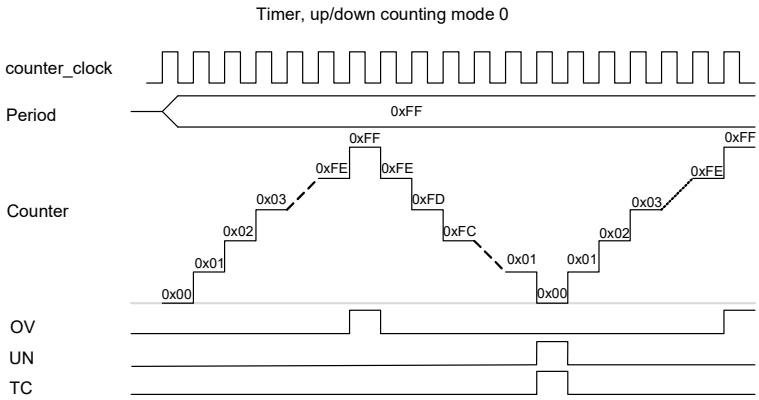
- 定时器是一个递增、递减和递增 / 递减计数器。
  - 当前的计数值被存储在计数寄存器 （TCPWM_CNTx_COUNTER）内。
    注意：在计数器运行过程中不能对该寄存器进行写操作。
  - 定时器的周期值被存储在周期寄存器内。
- 可在不同的计数模式下重新初始化计数器：
  - 在递增计数模式下，计数值达到周期值之后，计数寄存器会自动重新加载 ‘0’ 。
  - 在递减计数模式下，计数寄存器的值达到零之后，计数寄存器会重新加载周期寄存器中的值。
  - 在递增 / 递减计数模式下，计数寄存器的值达到终值时，它的值不被更新。当计数值等于零或周期值时，计数方向才发生改变。
- 当计数寄存器中的值等于比较寄存器中的值时，会生成 CC 条件。发生该条件时，如果通过计数器控制（TCPWM_CNT_CTRL）寄存器中的 AUTO_RELOAD_CC 位字段使能了比较寄存器和缓冲比较寄存器，则这两个寄存器的值将被切换。可通过该条件生成中断请求。

图 17-5 显示的是计数器在四种不同的计数模式下的定时器操作模式。周期寄存器存储最大的计数器值。

- 在递增计数模式下，如果周期值为 A，则会引起 A+1 个计数器周期 （0 到 A）。
- 在递减计数模式中， A 的周期值会导致 A+1 个计数器周期 （A 到 0）。
- 在两种递增 / 递减计数模式 （模式 0 和 1）中， A 的周期值需要 2*A 个计数器周期 （1 到 A 并 A 回至 0）。

图 17-5. 各种计数模式下的定时器时序图





注意: OV 和 UN 信号在 HFCLK 的一个周期内保持为逻辑高电平状态，如 第 207 页上的 “ 发生触发条件时的信号 ” 一节所示。本节中的所有图表都假设 HFCLK 与计数器时钟相同。

17.3.1.3 将计数器配置为定时器模式

下面各步骤介绍的是配置定时器操作模式的计数器以及受影响的寄存器位。

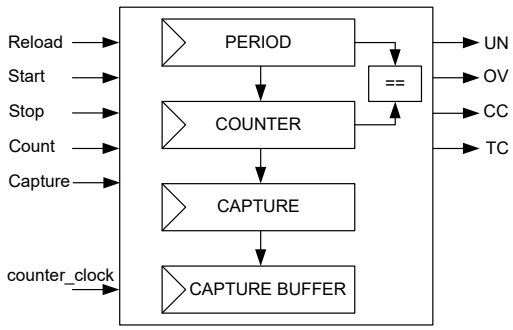
1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相对应位写入零，可以禁用该计数器。
2. 通过向 TCPWM_CNT_CTRL 寄存器中的 MODE[26:24] 字段写入 ‘000’ 选择 PWM 模式。
3. 设置 TCPWM_CNT_PERIOD 寄存器中所需要的 16 位周期值。
4. 设置 TCPWM_CNT_CC 寄存器中的 16 位比较值以及 TCPWM_CNT_CC_BUFF 寄存器中的缓冲比较值。
5. 如果需要每次发生 CC 条件时切换寄存器的值，请设置 TCPWM_CNT_CTRL 寄存器中的 AUTO_RELOAD_CC 字段。
6. 通过对 TCPWM_CNT_CTRL 寄存器中的 GENERIC[15:8] 字段进行写操作，设置时钟预分频值，如表 17-1 所示。
7. 通过对 TCPWM_CNT_CTRL 寄存器中的 UP_DOWN_MODE[17:16] 字段进行写操作，设置计数的方向，如表 17-6 所示。
8. 通过向 TCPWM_CNT_CTRL 寄存器的 ONE_SHOT[18] 字段写入 0 或 1，可分别将定时器配置为连续模式或单触发模式。
9. 设置 TCPWM_CNT_TR_CTRL0 寄存器以选择事件 （重载、启动、停止、捕获和计数）的触发源。
10. 设置 TCPWM_CNT_TR_CTRL1 寄存器以选择引起发生事件 （重载、启动、停止、捕获和计数）的触发边沿。
11. 需要时，请根据 TC 或 CC 条件设置中断，如第 207 页上的 “ 中断 ” 一节所示。
12. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 ‘1’，可以使能该计数器。如果硬件启动信号未被使能，则固件 （TCPWM_CMD 寄存器）必须提供一个启动触发信号，用以启动计数器。

17.3.2 捕获模式

在捕获模式下，通过固件对指令寄存器 （TCPWM_CMD）进行写操作或通过捕获触发输入信号，都可随时捕获计数器的值。该模式用于测量周期值和脉冲宽度。

17.3.2.1 框图

图 17-6. 捕获模式框图



17.3.2.2 工作原理

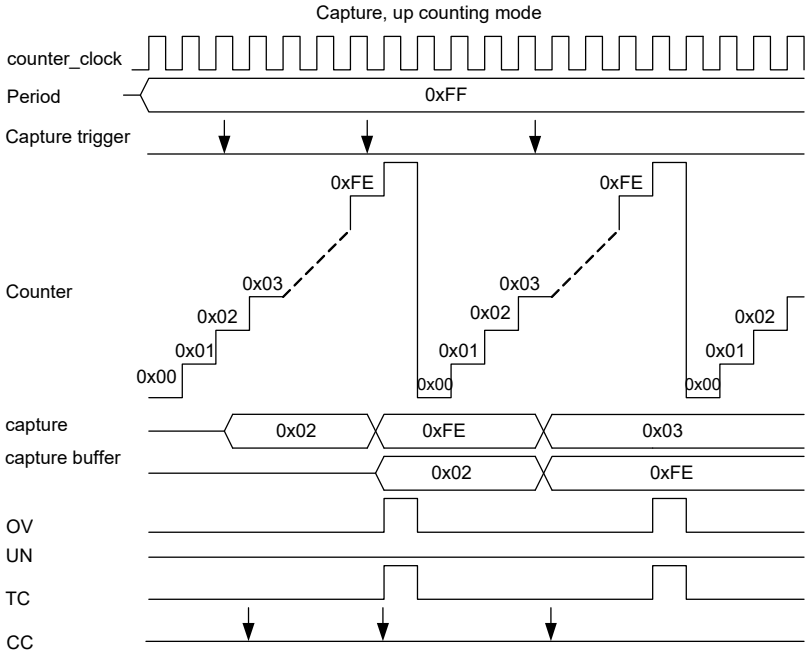
通过配置计数器控制 （TCPWM_CNT_CTRL）寄存器中的 UP_DOWN_MODE[17:16] 位字段，可将计数器设置为递增、递减或递增 / 递减计数模式。

在捕获模式下可进行以下操作：

- 发生由硬件或软件生成的捕获事件时，当前的计数寄存器值被复制到捕获寄存器 （TCPWM_CNT_CC）内，并且捕获寄存器中的值被复制到缓冲捕获寄存器 （TCPWM_CNT_CC_BUFF）内。
- 当计数器的值被复制到捕获寄存器内时， CC 输出信号上将生成一个脉冲。该条件还可用于生成中断请求。

图 17-7 显示了在递增计数模式下的捕获行为。

图 17-7. 在递增计数捕获模式下的计数器时序图



在该图中可以观察到：

- 周期寄存器包含最大的计数值。
- 当计数器达到周期值时，会生成内部上溢 （OV）和 TC 状态。
- 只有在边沿上或通过软件才能生成捕获事件。使用触发控制寄存器 1 配置边沿检测。
- 按照下面的设置处理单个时钟周期内的多个捕获事件：
  - 偶数捕获事件 — 尚未观察到任何事件
  - 奇数捕获事件 — 观察到单一事件

当捕获信号频率超过了 counter_clock 频率时，可观察到上述现象。

17.3.2.3 将计数器配置为捕获模式

下面各步骤介绍的是配置捕获操作模式下的计数器以及受影响的寄存器位。

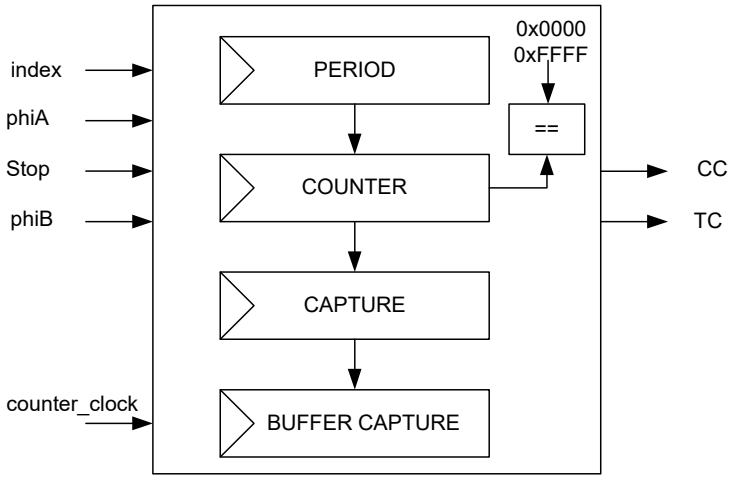
1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相对应位写入零，可以禁用该计数器。
2. 通过将 ‘010’ 写入到 TCPWM_CNT_CTRL 寄存器的 MODE[26:24] 字段内选择捕获模式。
3. 设置 TCPWM_CNT_PERIOD 寄存器中所需要的 16 位周期值。
4. 通过对 TCPWM_CNT_CTRL 寄存器中的 GENERIC[15:8] 字段进行写操作，设置时钟预分频值，如表 17-1 所示。
5. 通过对 TCPWM_CNT_CTRL 寄存器中的 UP_DOWN_MODE[17:16] 字段进行写操作，设置计数的方向，如表 17-6 所示。
6. 通过向TCPWM_CNT_CTRL寄存器中的ONE_SHOT[18]字段内写入0或1，可分别将计数器配置为连续模式或单触发模式。
7. 设置 TCPWM_CNT_TR_CTRL0 寄存器以选择事件 （重载、启动、停止、捕获和计数）的触发源。
8. 设置 TCPWM_CNT_TR_CTRL1 寄存器以选择引发事件 （重载、启动、停止、捕获和计数）的边沿。
9. 如果需要，根据 TC 或 CC 条件设置中断，如第 207 页上的 “ 中断 ” 一节所示。
10. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 ‘1’，可以使能该计数器。如果硬件启动信号未被使能，则固件 （TCPWM_CMD 寄存器）必须提供一个启动触发信号，用以启动计数器。

17.3.3 正交解码器模式

正交解码器可用于确定旋转式设备 （如伺服电机、音量控制轮和电脑鼠标）的速度和位置。正交解码器信号可作为解码器的 phiA 和 phiB 输入使用。

17.3.3.1 框图

图 17-8. 正交模式框图



17.3.3.2 工作原理

只有 counter_clock 为正交解码器提供时钟。该模块支持 X1、 X2 和 X4 三种子模式。可以通过计数器控制（TCPWM_CNT_CTRL）寄存器中的 QUADRATURE_-MODE[21:20] 字段控制这些编码模式。该模式使用了双缓冲的捕获寄存器。

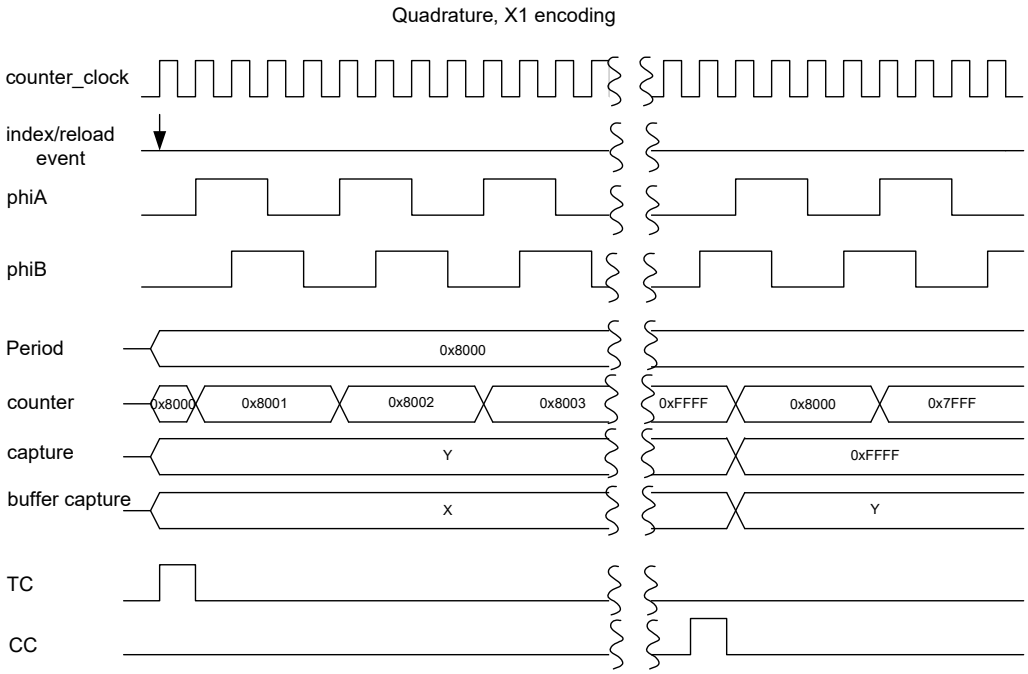
下面介绍的是正交模式下的操作：

- 正交相位phiA和phiB：计数方向可由phiA和phiB间的相位关系确定。这些相位分别连接至计数和启动触发输入，作为解码器的硬件输入使用。
- 正交索引信号：该信号连接至重载信号，作为硬件输入使用。该事件会生成 TC 条件，如图 17-9 所示。

  发生 TC 条件时，计数器将被设置为 0x0000 （在递增计数模式下）或周期值 （在递减计数模式下）。

  注意：在递减计数模式下，建议使用 0x8000 周期值 （中点值）。
- 当计数寄存器值达到 0x0000 或 0xFFFF 时，会在 CC 输出信号上生成一个脉冲。在 CC 的情况下，计数寄存器的值被设置为 0x8000。
- 在 TC 或 CC 的情况下：
  - 计数寄存器的值被复制到捕获寄存器中
  - 捕获寄存器的值被复制到缓冲捕获寄存器中
  - 可以使用该条件生成中断请求
- 使用捕获寄存器中的值可确定引发事件的条件，并确定是否：
  - 发生了计数器下溢 （该值为 0）
  - 发生了计数器上溢出 （该值为 0xFFFF）
  - 发生了索引 /TC 事件 （该值为非 0 或非 0xFFFF）
- 通过读取计数器状态 （TCPWM_CNTx_STATUS）寄存器中的 DOWN 位字段，可以确定当前的计数方向。数值 ‘0’ 表示前一个递增操作，数值 ‘1’ 表示前一个递减操作。图 17-9 说明了 X1 解码模式下的正交操作状况。
  - phiA的上升沿分别在phiB等于 ‘0’ 和phiB等于 ‘1’ 时递增和递减计数器。
  - 发生索引 / 重载事件时，使用周期值初始化计数寄存器。
  - 索引事件初始化计数器时，会生成计数终值事件。可以使用该事件生成中断。
  - 当计数寄存器达到 0xFFFF 值 （最大计数寄存器值）时，该计数寄存器的值将被复制到捕获寄存器内，并且计数寄存器将被初始化为周期值 （0x8000）。

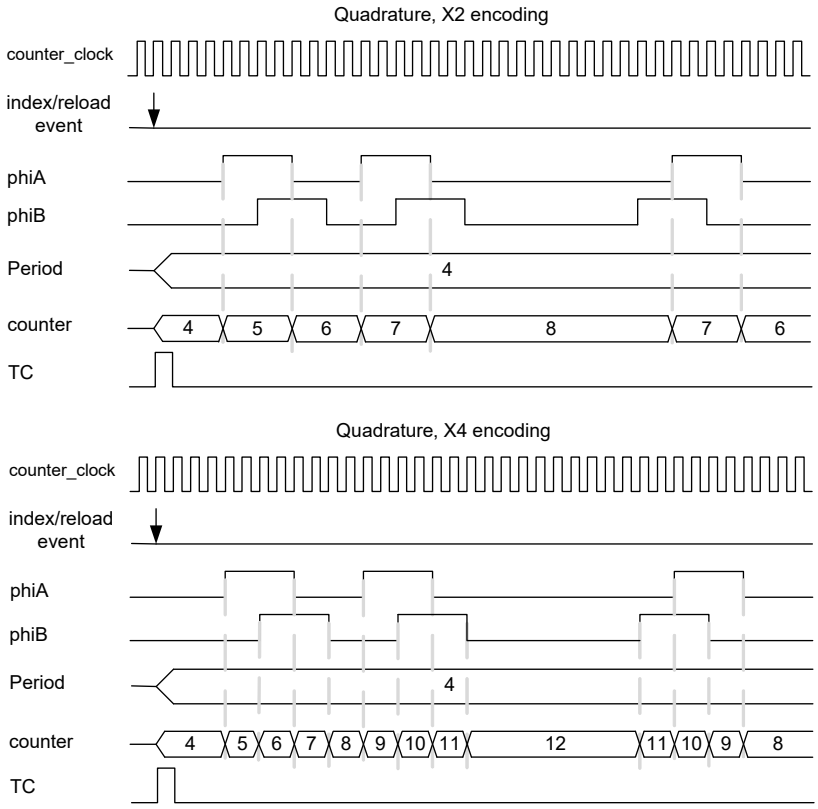
图 17-9. X1 正交解码模式的时序图



可根据 counter_clock 信号检测正交相位。在单个 counter_clock 周期内，只能对相位的值进行一次修改。X2 和 X4 正交编码模式的计数速度分别比 X1 编码模式下的计数速度快一倍和三倍。

图 17-10 说明了 X2 和 X4 解码模式下的正交模式状况。

图 17-10. X2 和 X4 正交解码模式的时序图



17.3.3.3 配置正交模式的计数器

下面各步骤介绍的是配置正交模式下的计数器以及受影响的寄存器位。

1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相对应位写入零，可以禁用该计数器。
2. 通过向 TCPWM_CNT_CTRL 寄存器中的 MODE[26:24] 字段写入 ‘011’，可以选择正交模式。
3. 设置 TCPWM_CNT_PERIOD 寄存器中所需要的 16 位周期值。
4. 对 TCPWM_CNT_CTRL 寄存器中的 QUADRATURE_MODE[21:20] 字段进行写操作，以便设置所需要的编码模式。
5. 设置 TCPWM_CNT_TR_CTRL0 寄存器可以选择事件 （索引和停止）的触发源。
6. 设置 TCPWM_CNT_TR_CTRL1 寄存器可以选择引发事件 （索引和停止）的边沿。
7. 需要时，请根据 TC 或 CC 条件设置中断，如第 207 页上的 “ 中断 ” 一节所示。
8. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 ‘1’，可以使能该计数器。

17.3.4 脉宽调制模式

PWM 模式也被称为数字比较器模式。比较输出是一个 PWM 信号，其周期值取决于周期寄存器的值，而其占空比则取决于比较和周期寄存器的值。

在左对齐和右对齐模式下：PWM 周期 = （周期值 / 计数器时钟频率）

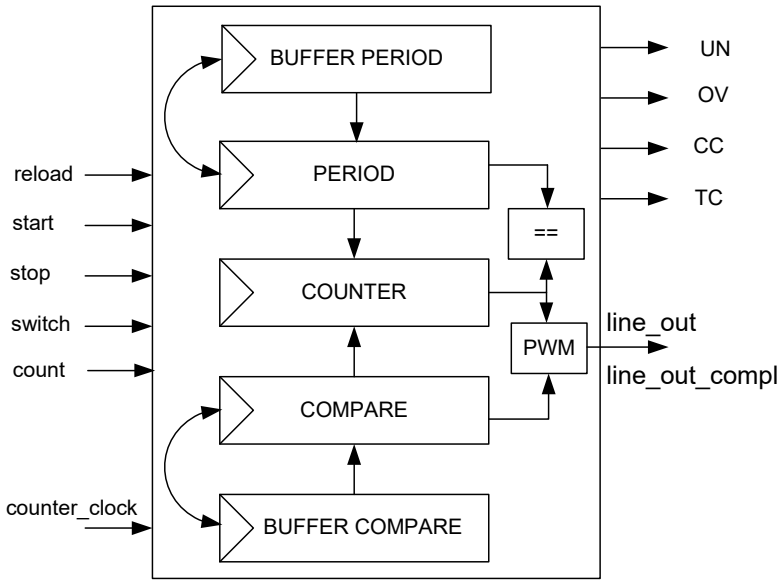
在中心对齐模式下：PWM 周期 = （2 × （周期值 / 计数器时钟频率））

在左对齐和右对齐模式下：占空比 = （比较值 / 周期值）

在中心对齐模式下：占空比 = （（周期值 - 比较值） / 周期值）

17.3.4.1 框图

图 17-11. PWM 模式框图



17.3.4.2 工作原理

PWM模式可输出左对齐、右对齐、中心对齐或非对称的PWM信号。通过 TCPWM_CNT_CTRL 寄存器中的 UP_DOWN_-MODE [17:16] 位选择计数器的递增、递减、递增 / 递减计数模式，可以得到所需要的输出对齐，如表 17-6 所示。

CC、OV 和 UN 信号均能够控制 PWM 输出信号线。通过配置 TCPWM_CNT_TR_CTRL2 寄存器，这些信号可以翻转输出信号线，或将其设置为逻辑 ‘0’ 或逻辑 ‘1’ 。通过配置信号影响输出信号线的方式，可以得到所需要的 PWM 输出对齐。

建议按照以下方法修改占空比：

- 使用新值更新缓冲周期寄存器和缓冲比较寄存器。
- 在TC条件下，如果发生了某个有效的切换事件，周期寄存器和比较寄存器将自动被更新为缓冲周期寄存器和缓冲比较寄存器的值。计数器控制寄存器中的 AUTO_RE-LOAD_CC 和 AUTO_RELOAD_PERIOD 字段均被设置为 ‘1’ 。当检测到某个切换事件时，该事件将被保存，直到发生下一个 TC 事件为止。“ 通过 ” 信号 （设置事件检测时所选择的）不能触发切换事件。
- 发生下一个 TC 事件前，需要通过有效的切换事件完成对缓冲周期寄存器和缓冲比较寄存器的更新；否则，切换操作不会影响寄存器的更新，如 图 17-13 所示。

在中心对齐模式下，需要进行的操作为：下溢 = 清除，上溢 = 设置，CC = 反转

在发生重载事件时，计数寄存器被初始化，并在合适的模式下开始计数。每次计数，都会将计数寄存器的值和比较寄存器的值进行比较，用以在发生匹配时生成 CC 信号。

图 17-12 显示了有缓冲周期寄存器和比较寄存器的中心对齐脉宽调制 （递增 / 递减计数模式 0）。

图 17-12. 中心对齐 PWM 的时序图

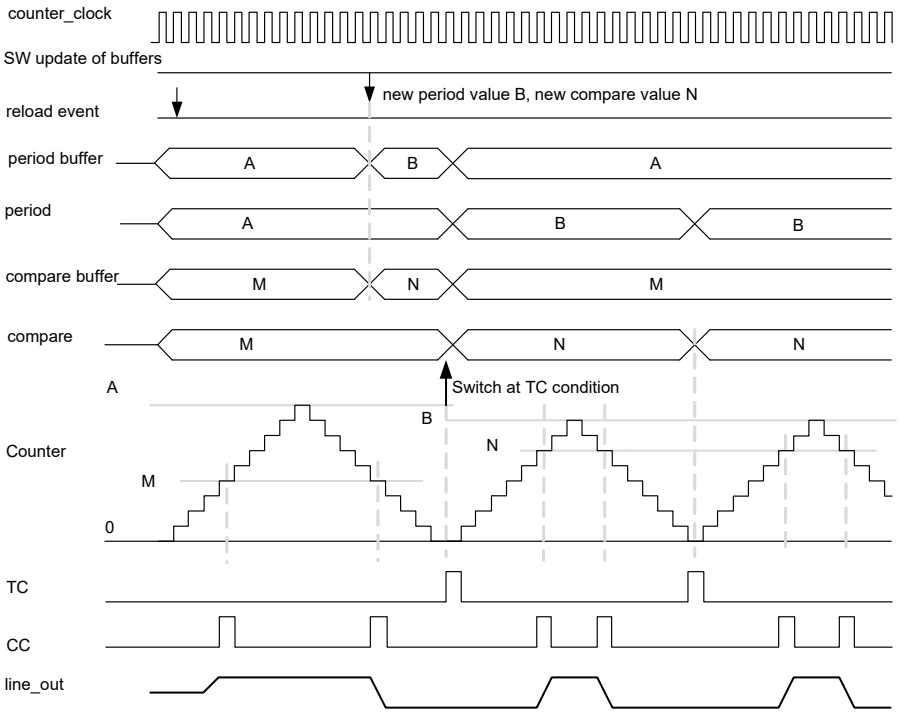
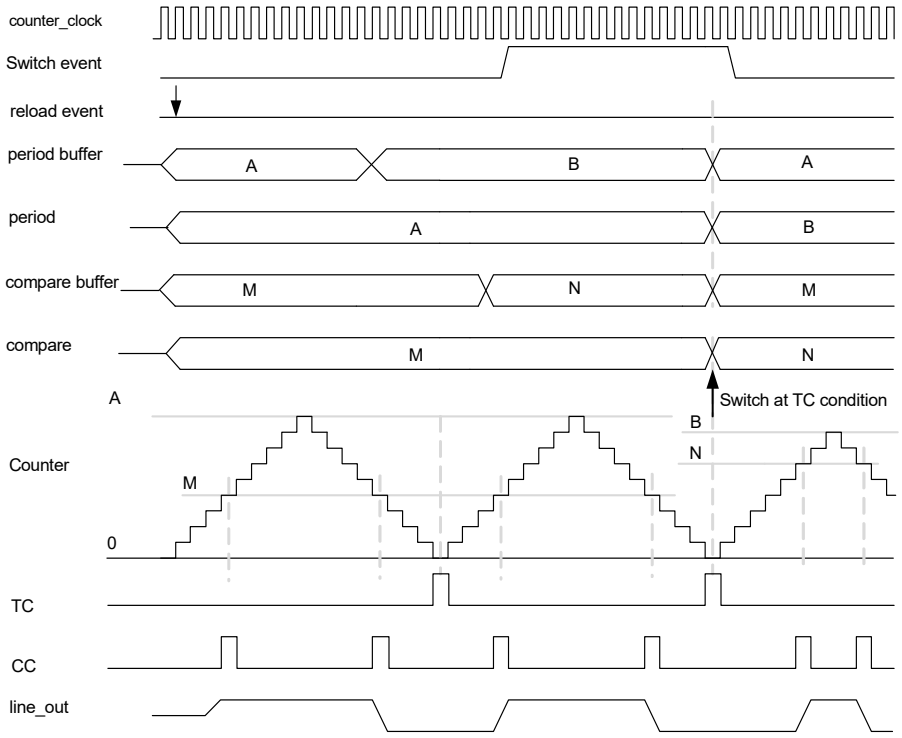


图 17-12 显示的是中心对齐 PWM，其中切换事件是由软件生成的：

- 仅在更新了周期缓冲寄存器和比较缓冲寄存器后，软件才会生成切换事件。
- 由于第二个 PWM 脉冲的更新被延迟 （达到计数终值后），所以第一个 PWM 脉冲将被重复。
- 请注意，切换事件完成后，硬件将在 TC 条件下自动取消该事件。

图 17-13. 中心对齐脉宽调制的时序图 （软件切换事件）



17.3.4.3 *其他配置*

- 对于不对称 PWM，应使用递增 / 递减计数模式 1。当计数器达到 ‘0’ 或周期值时，会发生 TC 事件。想要创建某个非对称 PWM，需要满足下面的条件：在每次发生 TC 事件 （计数器达到 0 或周期值）时更改比较寄存器的值，同时，只能在发生其他 TC 事件 （只在计数器达到 0）时更改周期寄存器。确保周期值在一个 PWM 周期内保持不变。
- 对于左对齐 PWM，使用递增计数模式；配置 OV 条件以将输出信号线设置为 ‘1’ 并配置 CC 条件以将输出信号线复位为 ‘0’ 。请参见表 17-3。
- 对于右对齐 PWM，使用递减计数模式；配置 UN 条件以将输出信号线复位为 ‘0’ 并配置 CC 条件以将输出信号线设置为 ‘1’ 。请参见表 17-3。

17.3.4.4 *终止 （kill）特性*

使用终止 （Kill）特性，可以立即禁用两个输出信号线。可以编程该事件，从而通过修改计数器控制寄存器中的 PWM_STOP_ON_KILL 和 PWM_SYNC_KILL 字段便可停止计数器，如表 17-7 所示。

表 17-7. 使能终止 （Kill）事件停止计数器特性的字段设置

PWM_STOP_ON_KILL 字段	说明
0	终止 （kill）事件暂时禁用 PWM 输出信号线，但计数器仍然运行。
1	终止 （kill）事件暂时禁用 PWM 输出信号线，同时也停止计数器的操作。

可将终止 （kill）事件编程为异步事件或同步事件，如表 17-8 所示。

表 17-8. 同步 / 异步终止 （Kill）的字段设置

PWM_SYNC_KILL 字段	说明
0	只要存在异步终止 （kill）事件，它便一直有效。该事件需要 “ 通过 ” 模式。
1	同步终止 （kill）事件禁用 PWM 输出信号线，直到发生下一个 TC 事件为止。该事件需要上升沿模式。

在同步终止 （kill）事件中，发生下一个 TC 条件前，不能启动 PWM。移除终止 （kill）输入后，如果要立即重启 PWM，那么应该使用异步终止 （kill）事件 （请参见表 17-8）。通过所生成的停止 （stop）事件，可以禁用两个输出信号线。在这种情况下，重载事件应该在下降沿检测模式下使用相同的触发输入信号。

17.3.4.5 将计数器配置为 PWM 模式

下面介绍的是配置 PWM 操作模式的计数器以及受影响的寄存器位的各个步骤。

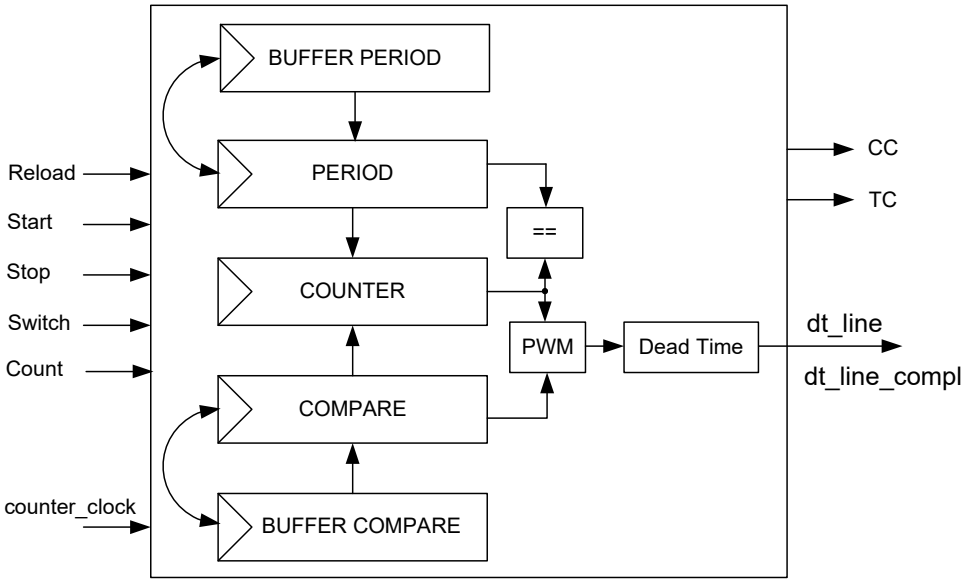
1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相对应位写入零，可以禁用该计数器。
2. 通过向 TCPWM_CNT_CTRL 寄存器中的 MODE[26:24] 字段内写入 ‘100’ 选择 PWM 模式。
3. 通过对 TCPWM_CNT_CTRL 寄存器中的 GENERIC[15:8] 字段进行写操作，设置时钟预分频值，如表 17-1 所示。
4. 根据需求，分别将 TCPWM_CNT_PERIOD 寄存器中的 16 位周期值和 TCPWM_CNT_PERIOD_BUFF 寄存器中的缓冲周期值设置为切换值。
5. 根据需求，分别将 TCPWM_CNT_CC 寄存器中的 16 位比较值和 TCPWM_CNT_CC_BUFF 寄存器中的缓冲比较值设置为切换值。
6. 通过对 TCPWM_CNT_CTRL 寄存器中的 UP_DOWN_MODE[17:16] 字段进行写操作设置计数方向，可将其配置为左对齐、右对齐或中心对齐的 PWM，如表 17-6 所示。
7. 根据需求，设置 TCPWM_CNT_CTRL 寄存器中的 PWM_STOP_ON_KILL 和 PWM_SYNC_KILL 字段。
8. 通过设置 TCPWM_CNT_TR_CTRL0 寄存器选择事件 （重载、启动、终止 （Kill）、切换和计数）的触发源。
9. 通过设置 TCPWM_CNT_TR_CTRL1 寄存器选择引起事件 （重载、启动、终止 （Kill）、切换和计数）的边沿。
10. 通过 TCPWM_CNT_TR_CTRL2 寄存器控制 line_out 和 line _out_compl 在发生 CC、 OV 和 UN 时置位、复位或翻转。
11. 需要时，请根据 TC 或 CC 条件设置中断，如第 207 页上的 “ 中断 ” 一节所示。
12. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 ‘1’，可以使能该计数器。如果硬件启动信号未被使能，则固件 （TCPWM_CMD 寄存器）必须提供一个启动触发信号，用以启动计数器。

17.3.5 带死区时间模式的脉宽调制

死区时间用于延迟 ‘line_out’ 和 ‘line_out_compl’ 信号的转换。它通过特定的时间间隔分离这两个信号的转换边沿。两条互补输出信号线 ‘dt_line’ 和 ‘dt_line_compl’ 是由这两条信号线生成的。在死区期间，比较输出和互补比较输出均在固定时长内保持逻辑 0 状态。通过死区特性，可以生成两个非重叠的 PWM 脉冲。使用该特性，最多可生成长达 255 个时钟周期的死区时间。

17.3.5.1 框图

图 17-14. PWM-DT 模式框图



17.3.5.2 工作原理

带有死区时间模式的 PWM 操作如下：

- 在PWM line_out的上升沿上，根据UN、OV和CC事件，死区时间模块将 dt_line 和 dt_line_compl 设置为 ‘0’。
- 根据寄存器中配置的周期加载并计数死区周期。
- 死区周期完成后， dt_line 被设置为 ‘1’。
- 在PWM line_out的下降沿上，根据UN、OV和CC事件，死区时间模块将 dt_line 和 dt_line_compl 设置为 ‘0’。
- 根据寄存器中配置的周期加载并计数死区周期。
- 死区周期完成后， dt_line_compl 被设置为 1。
- 零死区周期不会对 dt_line 产生任何影响，它与 line_out 相同。
- 当死区时间的持续时长等于或超过了脉冲的宽度时，脉冲将被移除。

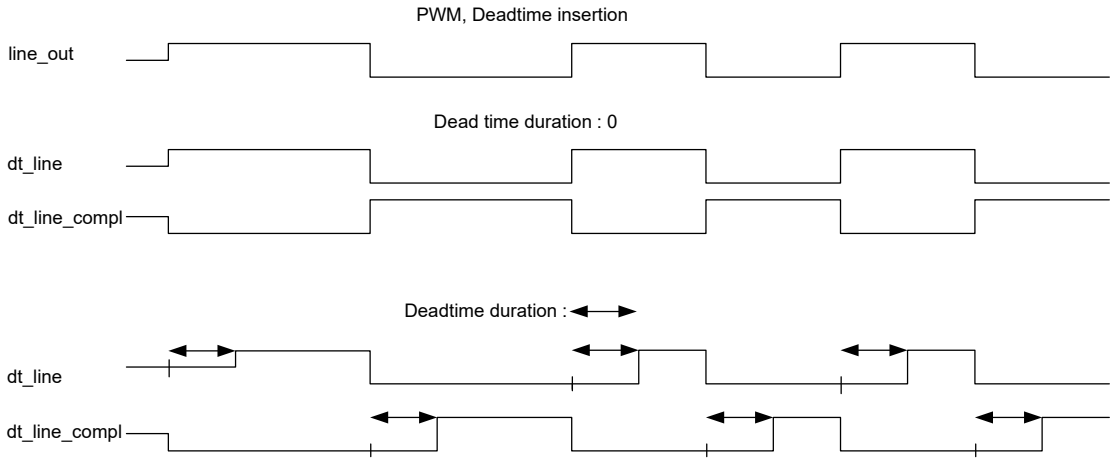
该模式遵循 PWM 模式并支持 PWM 模式的下述可用特性：

- 多种输出对齐模式
- 两个互补输出信号线（dt_line 和 dt_line_compl）分别由 PWM “line_out” 和 “line _out_compl” 生成
  - 可将停止（stop）/ 终止（kill）事件配置为同步模式和异步模式
  - 可对比较和缓冲比较寄存器以及周期和缓冲周期寄存器进行有条件切换事件

该模式不支持时钟预分频。

图 17-15 显示了如何根据 PWM 输出信号线 “line_out” 生成互补输出信号线 “dt_line” 和 “dt_line_compl”。

图 17-15. 带和不带死区时间的 PWM 时序图



17.3.5.3 *将计数器配置为带死区时间的 PWM 模式*

下面各步骤介绍的是配置 PWM （带死区时间）操作模式的计数器以及受影响的寄存器位：

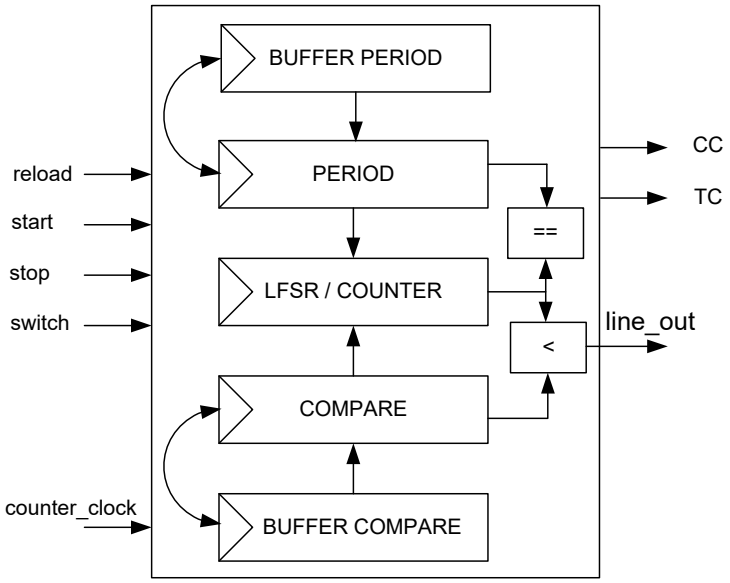
1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相对应位写入零，可以禁用该计数器。
2. 通过将 '101' 写入到 TCPWM_CNT_CTRL 寄存器中的 MODE[26:24] 字段选择带死区时间模式的 PWM。
3. 通过对 TCPWM_CNT_CTRL 寄存器中的 GENERIC[15:8] 字段进行写操作设置所需要的死区时间，如表 17-1 所示。
4. 根据需求，分别将 TCPWM_CNT_PERIOD 寄存器中的 16 位周期值和 TCPWM_CNT_PERIOD_BUFF 寄存器中的缓冲周期值设置为切换值。
5. 根据要求，分别将 TCPWM_CNT_CC 寄存器中的 16 位比较值和 TCPWM_CNT_CC_BUFF 寄存器中的缓冲比较值设置为切换值。
6. 通过对TCPWM_CNT_CTRL寄存器中的 UP_DOWN_MODE[17:16]字段进行写操作设置计数方向，可将其配置为左对齐、右对齐或中心对齐的 PWM，如表 17-6 所示。
7. 根据要求，设置 TCPWM_CNT_CTRL 寄存器中的 PWM_STOP_ON_KILL 和 PWM_SYNC_KILL 字段，如第 218 页上的 " 脉宽调制模式 " 一节所示。
8. 通过设置 TCPWM_CNT_TR_CTRL0 寄存器选择事件 （重载、启动、终止 （Kill）、切换和计数）的触发源。
9. 通过设置 TCPWM_CNT_TR_CTRL1 寄存器选择引起事件 （重载、启动、终止 （Kill）、切换和计数）的边沿。
10. 通过 TCPWM_CNT_TR_CTRL2 寄存器控制 dt_line 和 dt_line_compl 在发生 CC、 OV 和 UN 时进行置位、复位或翻转。
11. 需要时，请根据 TC 或 CC 条件设置中断，如第 207 页上的 " 中断 " 一节所示。
12. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 '1'，可以使能该计数器。如果硬件启动信号未被使能，则固件 （TCPWM_CMD 寄存器）必须提供一个启动触发信号启动计数器。

17.3.6 脉宽调制伪随机模式

该模式使用线性反馈移位寄存器 （LFSR）。 LFSR 是一个移位寄存器，其输入位是其前一个状态的线性功能。

17.3.6.1 框图

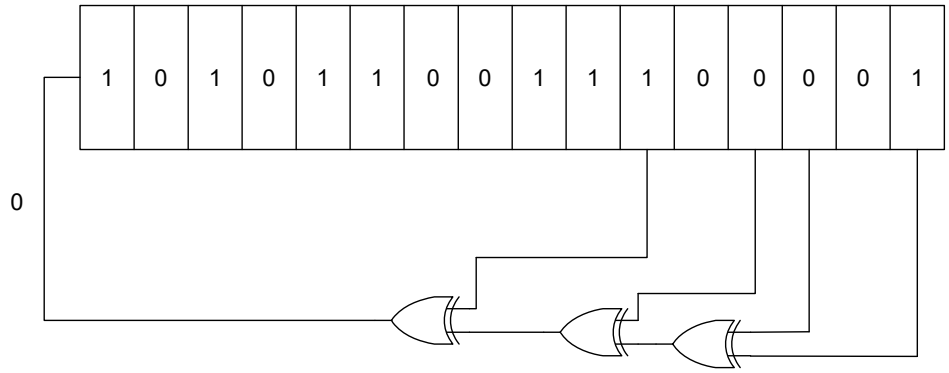
图 17-16. PWM-PR 模式的框图



17.3.6.2 工作原理

通过计数器寄存器，并使用多项式 $x^{16}+x^{14}+x^{13}+x^{11}+1$，可以执行 LFSR，如图 17-17 所示。它以伪随机序列生成 [1, 0xFFFF] 范围内所有的数值。注意：应该使用非零值初始化计数器寄存器。

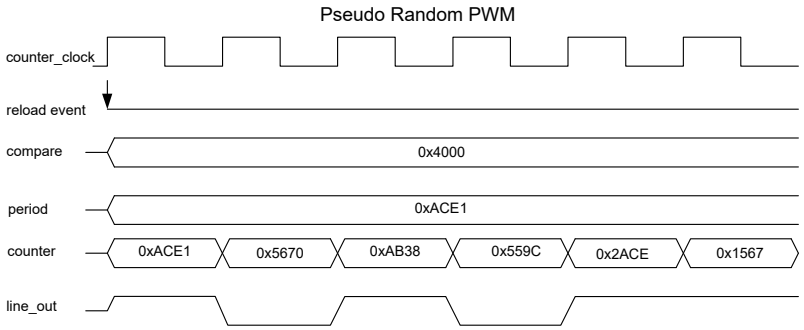
图 17-17. 使用计数器寄存器生成伪随机序列



该过程包括以下步骤：

- 当计数器寄存器中低15位的值小于比较寄存器中的值（如果计数器 [14:0] < 比较器 [15:0]）时， PWM 输出信号线（‘line_out’）将被驱动为 ‘1’。等于或大于 ‘0x8000’ 的比较值，都会使 PWM 输出信号线等于 ‘1’。如果比较值等于 ‘0’，则会使 PWM 输出信号线等于 ‘0’。
- 重载事件与启动事件相似，但它并不会初始化计数器。
- 当计数器的值等于周期值时，将生成计数终值 （TC）事件。 LFSR 为特定的初始值生成可预测的计数器值模式。在 LFSR 的计数重复了 ‘n’ 次后，可以通过该可预测性计算计数器的值。计算得出的计数器值可作为周期值使用，并且在重复 ‘n’ 次后将生成 TC 条件。
- 发生TC时，切换/捕获事件会有条件地切换比较和周期寄存器对 （即通过设置计数器控制寄存器中的 AUTO_RELOAD_CC 和 AUTO_RELOAD_PERIOD 字段）。
- 通过编程终止 （kill）事件可以禁止计数器，如前一节所述。
- 通过设置计数器控制寄存器中的 ONE_SHOT 字段，可以配置单触发模式。达到计数终值时，硬件将停止计数器。
- 在该模式下不会发生下溢、上溢和触发条件等事件。
- 当计数器运行，并且它的值等于比较值时，将发生 CC 事件。图 17-18 描述了伪随机噪声的状况。
- 如果比较值等于0x4000，那么能够生成50%的占空比（仅将 16 位计数器中低 15 位的值和比较寄存器的值进行比较）。

图 17-18. 伪随机 PWM 的时序图



捕获 / 切换输入信号可能切换比较寄存器和比较缓冲寄存器间的值，也可能切换周期寄存器和周期缓冲寄存器间的值。通过使用触发输入信号来控制调制操作，该功能可调制两个不同的比较值。

注意: 捕获 / 切换输入信号只能由某个边沿（上升沿、下降沿或两者）触发。该输入信号被记录，直到到达下一个计数终值为止。

17.3.6.3 将计数器配置为伪随机 PWM 模式

下面介绍的是将计数器配置为伪随机 PWM 操作模式的流程以及受影响的寄存器位。

1. 通过向 TCPWM_CTRL 寄存器的 COUNTER_ENABLED 字段的相应位写入零，可以禁用该计数器。
2. 通过向 TCPWM_CNT_CTRL 寄存器的 MODE[26:24] 字段写入 ‘110’ 选择伪随机 PWM 模式。
3. 根据需求，并为了切换各个值，分别设置 TCPWM_CNT_PERIOD 寄存器中所需要的周期值（16 位）和 TCPWM_CNT_PERIOD_BUFF 寄存器的缓冲周期值。
4. 为了切换各个值，分别设置 TCPWM_CNT_CC 寄存器的 16 位比较值和 TCPWM_CNT_CC_BUFF 寄存器的缓冲比较值。
5. 根据要求，设置 TCPWM_CNT_CTRL 寄存器中的 PWM_STOP_ON_KILL 和 PWM_SYNC_KILL 字段。
6. 通过设置 TCPWM_CNT_TR_CTRL0 寄存器选择引起事件 （重载、启动、终止 （Kill）和切换）的触发源。
7. 通过设置 TCPWM_CNT_TR_CTRL1 寄存器选择引起事件 （重载、启动、终止 （Kill）和切换）的边沿。
8. 通过 TCPWM_CNT_TR_CTRL2 寄存器控制 line_out 和 line _out_compl 在发生 CC、 OV 和 UN 时置位、复位或翻转。
9. 需要时，请根据 TC 或 CC 条件设置中断，如第 207 页上的 “ 中断 ” 一节所示。
10. 通过向 TCPWM_CTRL 寄存器 COUNTER_ENABLED 字段的相对应位写入 ‘1’，可以使能该计数器。

17.4 TCPWM 寄存器

表 17-9. TCPWM 寄存器列表

寄存器	说明	功能
TCPWM_CTRL	TCPWM 控制寄存器	使能计数器模块
TCPWM_CMD	TCPWM 指令寄存器	生成软件事件
TCPWM_INTR_CAUSE	TCPWM 计数器中断源寄存器	指定组合中断信号源
TCPWM_CNTx_CTRL	计数器控制寄存器	用于配置计数器模式、编码模式、单触发模式、切换性能、终止（kill）功能、死区时间、时钟预分频和计数方向
TCPWM_CNTx_STATUS	计数器状态寄存器	读取计数方向、死区时间和时钟预分频；检查计数器是否运行
TCPWM_CNTx_COUNTER	计数寄存器	包含 16 位计数器的值
TCPWM_CNTx_CC	计数器比较 / 捕获寄存器	捕获计数器的值或将需要比较的值与计数器的值进行比较
TCPWM_CNTx_CC_BUFF	计数器缓冲比较 / 捕获寄存器	计数器 CC 寄存器的缓冲寄存器；切换比较值
TCPWM_CNTx_PERIOD	计数器周期寄存器	包含计数器的上限值
TCPWM_CNTx_PERIOD_BUFF	计数器缓冲周期寄存器	计数器周期寄存器的缓冲寄存器；切换周期值
TCPWM_CNTx_TR_CTRL0	计数器触发控制寄存器 0	为特定计数器事件选择触发器
TCPWM_CNTx_TR_CTRL1	计数器触发控制寄存器 1	为特定计数器输入信号确定边沿检测
TCPWM_CNTx_TR_CTRL2	计数器触发控制寄存器 2	在发生 CC、 OV 和 UN 条件时控制计数器输出信号线
TCPWM_CNTx_INTR	中断请求寄存器	检测到 TC 或 CC 条件时设置该寄存器中的位
TCPWM_CNTx_INTR_SET	中断设置请求寄存器	设置中断请求寄存器中的相对应位
TCPWM_CNTx_INTR_MASK	中断屏蔽寄存器	中断请求寄存器的屏蔽
TCPWM_CNTx_INTR_MASKED	中断屏蔽请求寄存器	中断请求和屏蔽寄存器的按位 AND 运算

注意 ：寄存器名称中的 ‘x’ 表示 TCPWM 的数量。例如， TCPWM0 的中断屏蔽寄存器是 TCPWM_CNT0_INTR_MASK。

18. 蓝牙低功耗子系统 （BLESS）

低功耗蓝牙 （BLE）子系统 （BLESS）将执行低功耗蓝牙链路层和物理层，如蓝牙版本 4.2 规范中第六卷所示。该子系统能够同时执行蓝牙版本 4.2 规范中所定义的多种程序，并保持一个连接。本节介绍了 BLESS 的特性、实现以及操作模式。

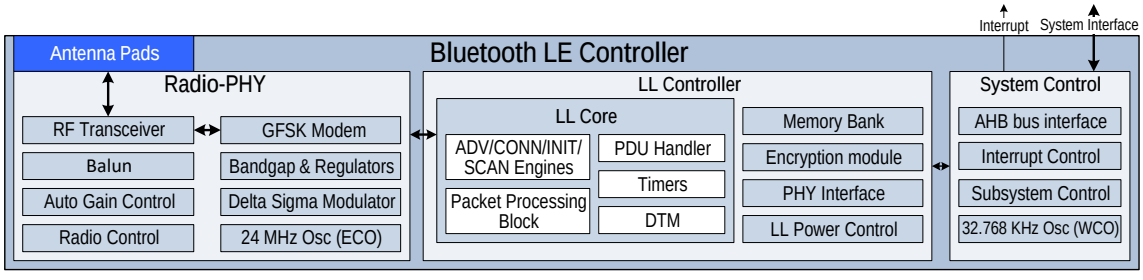
18.1 特性

该子系统支持以下特性：

- 链路层
  - 蓝牙版本 4.0 规范中所有必要和可选的 LE 链路层特性，以及蓝牙版本 4.2 规范中多种可选的 LE 链路层特性。
  - 可作为主设备和从设备使用。
  - 支持所有数据包类型和控制 PDU。
  - 重复和白名单过滤。
  - AES-128（高级加密标准）的加密 / 解密功能使用 CCM（计数器和密码分组链接（CBC）- 消息认证码（MAC））模式。
  - 支持蓝牙版本 4.2 性能，如低占空比广播、安全配对的低功耗 ping 机制和 L2CAP 面向连接信道。
- 具有功能齐全的低功耗蓝牙收发器。
- 物理层
  - GFSK delta-sigma 调制器以 1 Mbps 符号速率设置 1 MHz ~ 5 GHz 范围。
  - 不同模拟组件的自动校准。
  - 频率为 12 MHz 的 10 位 DAC 和带有 1 MHz IF、频率为 12 MHz 的 9 位 I/Q ADC。
  - 自动增益控制。
  - 将平衡－不平衡变换器 （巴伦）集成在单端 RF 引脚上。
  - 高精度的 24 MHz 和 32.768 kHz 振荡器。

18.2 框图

图 18-1. BLE 子系统框图



18.3 工作原理

BLESS 包括以下主要模块：

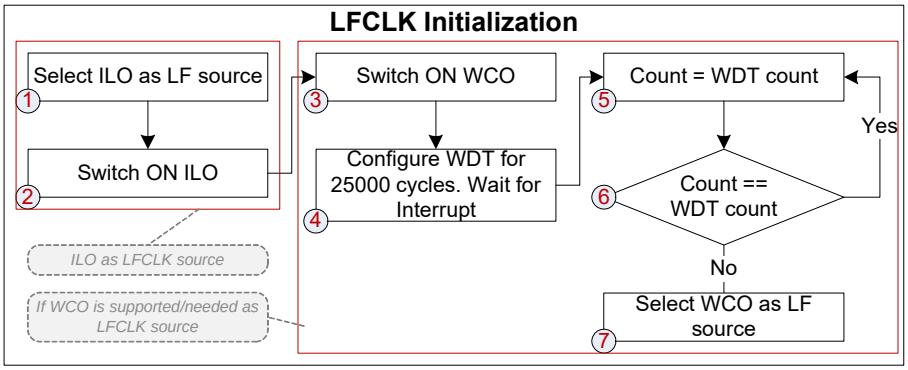
- 无线通信 PHY 模块（RF-PHY）：该模块实现蓝牙通信的 2.4 GHz RF 收发器、 1 Mbps 串行数据 GFSK 调制解调器、一个用于设置信道频率的 delta sigma 调制器、为接收器的动态范围和灵敏度自动增益的控制模块、电压调节器、校准和过滤的控制逻辑模块以及 24 MHz 外部晶体振荡器 （ECO）。
- 链路层（LL）控制器：该控制器为不同的蓝牙 LE 程序（广播、扫描、发起和连接）、模式 （主设备和从设备）、时序关键性能（数据包组帧 / 解帧、CRC 生成 / 检测、加密 / 解密、数据包传输和协议计时参考）和直接检测模式实现了状态机。
- 系统控制模块：该模块包括 AHB 总线接口逻辑，中断控制逻辑，控制子系统时钟、复位和功耗模式的子系统控制器以及一个频率为 32.768 kHz 的监视晶体振荡器（WCO）。该振荡器由链路层低功耗状态机和计时逻辑使用，并用于为系统生成一个准确的低频率时钟。

18.3.1 LFCLK 初始化

LFCLK 源有两种：WCO 和内部低速振荡器 （ILO）。在系统中将该时钟作为低频率时钟使用，并用于初始化 RF。如果 LFCLK 源是 WCO，那么它被额外地用于 BLESS 低功耗模式功能。所有不支持 WCO 的解决方案都不需要初始化 WCO，并且不支持 BLESS 低功耗模式功能，并要将 ILO 作为 LFCLK 源使用，以便初始化 RF。

为了初始化 LFCLK，需要在加电复位 （POR） 、外部复位（XRES）、欠压检测 （BOD）或其他系统复位事件（在该期间内 ILO 和 WCO 都被关闭）发生后执行以下序列。更多信息，请参考第 88 页上的 LFCLK 输入选择 的内容。

图 18-2. LFCLK 初始化



1. 通过写入WDT_CONFIG寄存器上的LF_CLK_SEL字段，将 ILO 选择为 LF 时钟源。
2. 通过设置 CLK_ILO_CONFIG 寄存器上的 ENABLE 字段打开系统资源的 ILO。 ILO 生成一个时钟需要 2 ms 的时间。该时钟的误差比较大，并只有在不支持 BLESS 低功耗模式操作的情况下才能使用它。

如果在解决方案中支持或需要 WCO，则需要执行以下子序列。

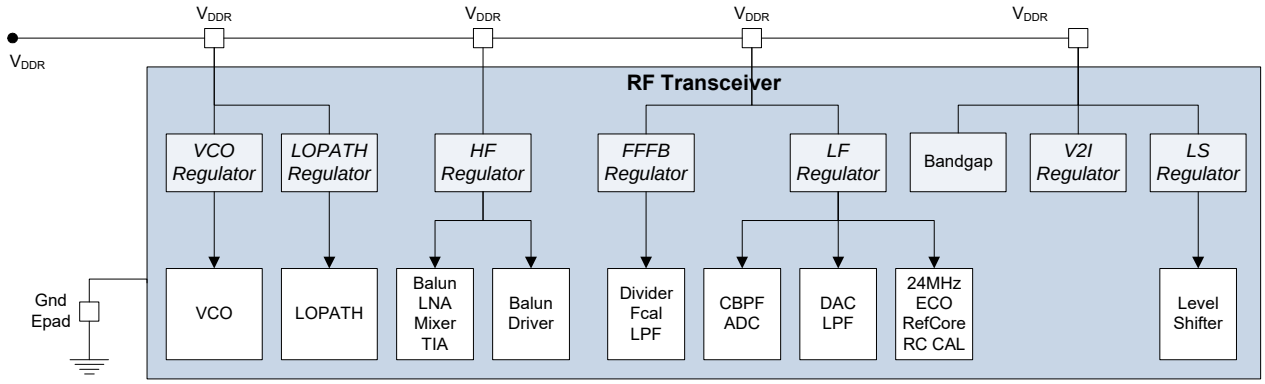
3. 通过设置 BLE_BLESS_WCO_CONFIG 寄存器上的 ENABLE 字段打开 BLESS 的 WCO。WCO 需要占用 0.5 秒的时间来生成一个时钟。
4. 对运行于 ILO 时钟的看门狗定时器 （WDT）进行相关配置，以在计数 1.7 秒的最差情况 （因 ILO 的不准确度造成）下进行计数。请参考第 107 页上的看门狗定时器章节以了解配置内容。
5. WDT 到期时，在下一个 WDT 的计数值递增时要将 LFCLK 从 ILO 切换到 WCO。可以通过持续监控相应的 WDT 计数器值来执行该操作。
6. 当该值改变时，通过写入 WDT_CONFIG 寄存器上的 LF_CLK_SEL 字段来切换时钟源。

18.3.2 无线 PHY 模块

18.3.2.1 电源

无线 PHY 模块具有电压调节器，可将电源供给模块中不同的子模块，如图 18-3 所示。该电压调节器网络仅给无线 PHY 模块供电。有关系统级电源，请参考第 95 页上的电源供应和监控章节。

图 18-3. BLE 子系统框图



V_{DDR} 将 1.9 V 到 5.5 V 的输入电压提供给不同的电压调节器。 RF 收发器电压调节器将 V_{DDR} 电压调整为 1.8 V 电压。

有关功耗模式和电压调节器状态的信息，请参考 表 18-1。第 232 页上的功耗模式 显示的是 BLESS 功耗模式的详细信息。

表 18-1. 在不同 BLE 模式下 RF 收发器调节器的状态

模式	LDO_LS	LDO_HF	LDO_VCO	LDO_LOPATH	LDO_FFFB	LDO_V2I	LDO_LF
BLE_DeepSleep	关闭	关闭	关闭	关闭	关闭	关闭	关闭
BLE_Sleep	启用	关闭	关闭	关闭	关闭	启用	启用
BLE_IDLE	启用	关闭	关闭	关闭	关闭	启用	启用
BLE_TX	启用	启用	启用	启用	启用	启用	启用
BLE_RX	启用	启用	启用	启用	启用	启用	启用

电平转换器电压调节器 （LDO_LS）

该电压调节器为 RF 收发器中的电平转换器和带隙供电。

高频率电压调节器 （LDO_HF）

该电压调节器给高频率无线电路（如低噪声放大器（LNA）、混频器和平衡－不平衡变换器电路）供电。

LDO_FFFB

前馈电压调节器给合成器模块（如分频器和 VCO 校准模块）中的电路供电。

VCO 电压调节器 （LDO_VCO）

使用 LDO_VCO 电压调节器为 RF 收发器的 VCO 部分供电。

LO 路径电压调节器 （LDO_LOPATH）

使用该电压调节器为本地振荡器路径供电。

V-I 电压调节器 （LDO_V2I）

使用 LDO_V2I 为 RF 收发器的电压电流转换器电路供电。

低频率电压调节器 （LDO_LF）

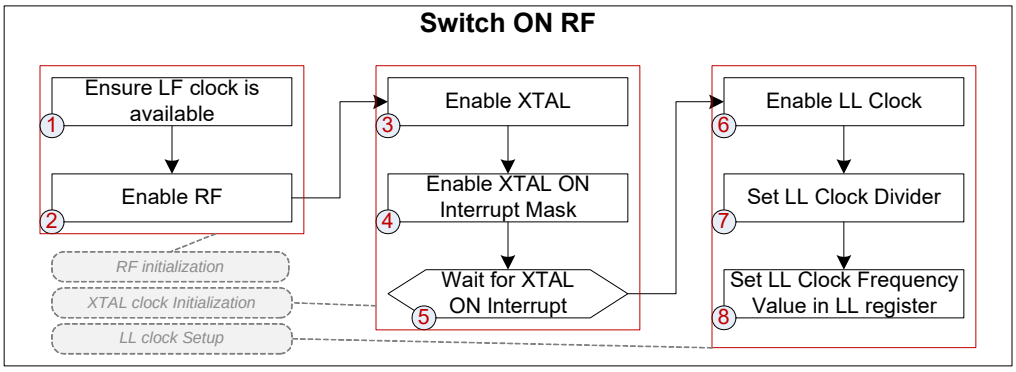
LDO_LF 为 RF 收发器中的低频率电路供电。低频率电路包括 ADC、互补带通滤波器 （CBPF）、 DAC、 ECO 等。

18.3.2.2 RF 初始化

带隙 （BG）和电平转换器电压调节器 （LDO_LS）

控制 RF 电压调节器的信号来自系统电域上的寄存器。因此，给系统供电时，才能控制 RF 调节器。有关给系统供电的模块和模式的详细信息，请参见第 95 页上的电源供应和监控章节和第 101 页上的功耗模式章节。

图 18-4. RF 和 LL 时钟初始化



- 为了打开 RF，LF 时钟源必须有效（ILO 或 WCO）。如果 WCO 没有被选择或者无效，则不支持 BLESS 低功耗模式。
- 为了将控制信号输入给 RF 调节器，需要设置 BLE_BLESS_RF_CONFIG 寄存器中的 RF_ENABLE 字段，打开 RF 带隙。这样会打开带隙发生器，然后是 LDO_LS。

ECO 初始化

在图 18-4 的中间显示了子序列。

- 当 BG 和 LDO_LS 被打开时，通过设置 BLE_BLERD_DBUS 寄存器中的 XTAL_ENABLE 字段可以使能 24 MHz ECO。
- 使用一个中断来表示时钟是否可用。通过设置 BLE_BLESS_LL_DSM_CTRL 寄存器中的 XTAL_ON_INTR_MASK 字段可以使能该中断。
- 中断状态被捕获在 BLE_BLESS_LL_DSM_INTR_STAT 寄存器的 XTAL_ON_INTR 字段中。

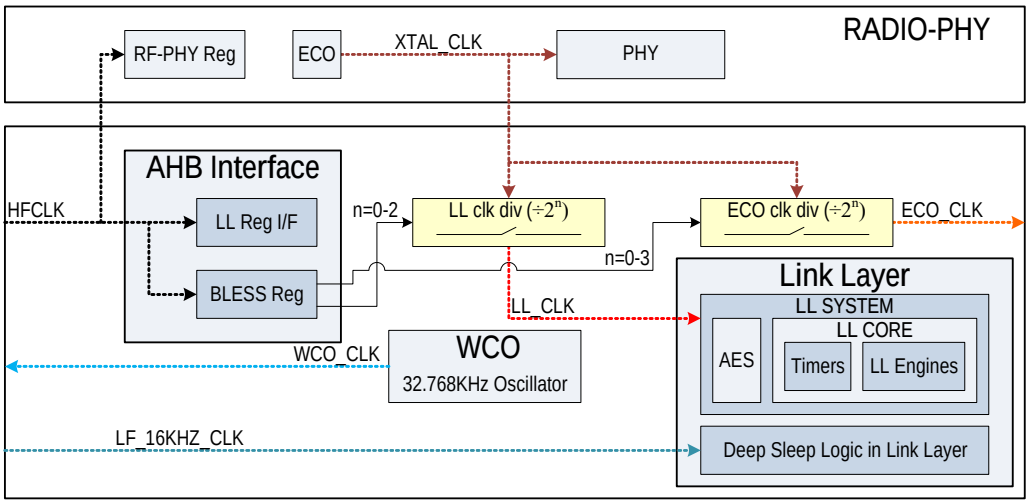
18.3.3 链路层控制器

本部分介绍了有关链路层的配置和高级功能模式的详细信息。

18.3.3.1 时钟

图 18-5 显示 BLESS 的时钟架构。

图 18-5. BLESS 时钟网络



BLESS 具有三个异步时钟区域，如图 18-5 所示。

XTAL_CLK：. XTAL_CLK是从ECO中派生的24 MHz时钟。该时钟为 RF-PHY 控制器、 GFSK 调制器和发送逻辑提供时钟脉冲。在 LL 控制器中，一个分频时钟 （LL_CLK）为链路层协议逻辑提供时钟脉冲，其他分频时钟（ECO_CLK）则作为系统中的外部时钟源使用。

- LL_CLK：在图 18-4 的右侧显示了子序列。
  - 为链路层LL_CLK时钟进行的主要控制是通过BLE_BLESS_LL_CLK_EN 寄存器中的 CLK_EN 位实现的。该位被使能时，链路层固件指令和控制位会进一步控制该时钟；这些指令和位能够使LL进入睡眠模式或深度睡眠模式，也能唤醒它。
  - 通过可编程分频器进行 1 次、 2 次和 4 次分频后，XTAL_CLK 会生成 LL_CLK。因此， LL_CLK 的频率可以是 24 MHz、12 MHz 或 6 MHz。可以在 BLE_BLESS_XTAL_CLK_DIV_CONFIG 寄存器的 LLCLK_DIV 字段中设置分频值。
  - 必须通过 BLE_BLELL_POC_REG_TIM_CTRL 寄存器的BB_CLK_FREQ_MINUS_1字段来设置相应的频率。表 18-2 显示了映射情况。

表 18-2. LL_CLK 频率和寄存器设置

LL_CLK 频率	LLCLK_DIV	BB_CLK_FREQ_MINUS_1
24 MHz	2'b00	4'd23
12 MHz	2'b01	4'd11
06 MHz	2'b10	4'd5

- ECO_CLK：通过可编程分频器1次、2次、4次和8次分频后，XTAL_CLK 会生成 ECO_CLK。因此，ECO_CLK 的频率可以是 24 MHz、 12 MHz、 6 MHz 或 3 MHz。可以在 BLE_BLESS_XTAL_CLK_DIV_CONFIG 寄存器的 SYSCLK_DIV 字段中设置该分频值。有关该时钟的系统级使用情况，请参见第 87 页上的外部晶振 （ECO）。

HFCLK：. 该时钟用于为 RF-PHY、子系统寄存器和 LL AHB 接口提供时钟脉冲。该时钟的频率范围为 3 ~ 48 MHz。有关该时钟系统的详细信息，请参见第 83 页上的时钟系统章节。

LF_16KHZ_CLK：. LF_16KHZ_CLK 是一个 16.384 kHz 精准时钟或是一个从系统低频率时钟 LFCLK 派生的 16 kHz 时钟。有关复用的详细信息，请参见第 87 页上的监视晶体振荡器 （WCO）和第 88 页上的时钟分布 。

- 16.384 kHz：LFCLK 源是 WCO。在该模式中，链路层使用该时钟来保持链路层低功耗模式中的高精度BLE协议计时。在BLESS深度睡眠模式进入扩展的深度睡眠模式时，BLE_BLESS_LF_CLK_CTRL 寄存器的固件控制位 （即 DISABLE_LF_CLK 字段）会禁止该时钟，这时无需低功耗模式下的 BLE 协议计时。
- 16 kHz：LFCLK源是ILO。该时钟不精确，并且BLESS低功耗模式下不支持 BLE 协议计时。在 BLESS 深度睡眠模式进入扩展的深度睡眠模式时， BLE_BLESS_LF_CLK_CTRL 寄存器的固件控制位（即 DISABLE_LF_CLK 字段）会禁止该时钟。

18.3.3.2 固件复位

通过使用链路层软件复位请求可以复位链路层。这样可以使所有寄存器和状态机复位为它们的默认状态。请注意，复位操作不会修改 FIFO 的内容。 FIFO 的内容变得无效，并且要重新初始化 FIFO。

18.3.3.3 BLE 功能模式和配置

该部分详细介绍了各种 BLE 功能模式。链路层的操作可通过状态机进行定义，具有以下主要模式：

- 待机
- 广播
- 扫描 （主动或被动）
- 发起
- 连接 （主设备或从设备）
- 直接测试模式 （DTM）
- 加密 / 解密

待机状态

该状态是默认的功能状态。在这种状态中，链路层不会传送或接收任何数据包。根据相关功能的要求，可以从该状态或者切换为广播、扫描或启动状态中的任何一种。还可以从其他状态进入待机状态。

广播状态

广播状态允许链路层发送广播信道数据包；它还能响应对端器件的回答（该回答是由这些广播信道数据包触发的）。处于这种状态下的器件被称为广播者，可以从待机状态中进入该状态。某个器件进入该状态，可被发现或连接。还允许该器件在相邻区域中广播数据给其他器件。

扫描状态

处于这种状态的链路层有助于器件监听来自当前正在广播的对等器件中的广播信道数据包。监听是为了将该状态分为以下两种：

- 主动扫描
- 被动扫描

使用被动扫描操作是为了简单地监听其他器件的存在。主动扫描允许器件将扫描请求回应给正在广播的器件，以得到对等器件的进一步数据，可能会进入连接阶段。

处于这种状态下的器件被称为扫描者，可以从待机状态进入这种状态。

启动状态

在启动状态下的链路层会监听来自特定对等器件的广播信道数据包，然后发起一个与对等器件的连接。在这种状态下的器件被称为启动器，可以从待机状态进入这种状态。

连接状态

链路层协议最终的过程状态是连接状态。可以从广播状态或

启动状态进入这种状态；将处于这种状态的器件称为正在连接。

这种状态有两种情况：

- 作为主设备
- 作为从设备

如果器件从广播状态进入这种状态，那么它作为从设备；如果器件从启动状态进入这种状态，那么它作为主设备。

直接测试模式

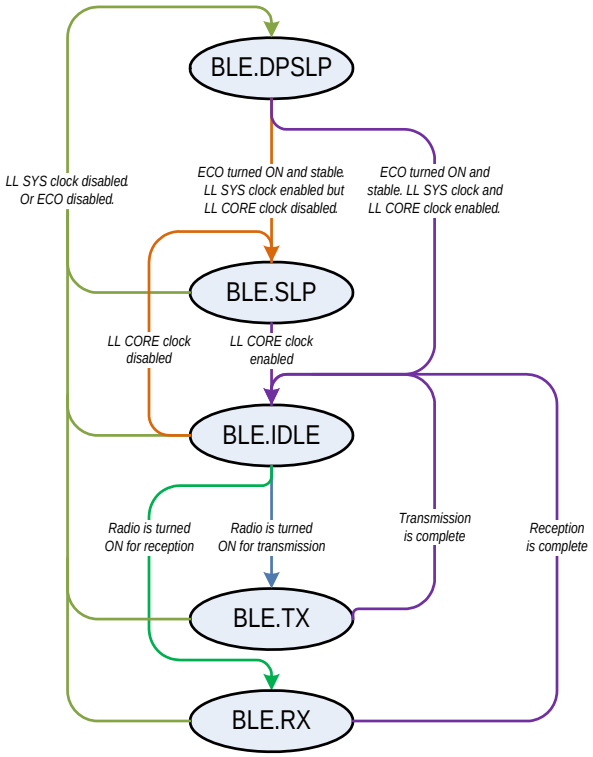
通过该模式可以测试物理层（无线数字 /RF）。它允许测试者发一个命令给控制器的物理层，如发送或接收一个预定义测试数据包的序列。然后，测试者可以对接收到的数据包进行分析，用以确定物理层是否按规范工作。测试者还可以测量来自接收到的数据包的多个 RF 参数，以便确定物理层与 RF 规范的兼容性。

加密 / 解密

通过该模式可以加密一个纯文本数据或解密一个被加密的数据。蓝牙低功耗采用 AES-128 （高级加密标准）加密 / 解密功能，该功能使用了 CCM（ 计数器和密码分组链接（CBC）- 消息认证码 （MAC））。

18.3.4 功耗模式

图 18-6. BLE 子系统功耗模式



BLESS 支持以下五种功能功耗模式：深度睡眠模式（DSM）、睡眠模式 （SM）、休闲模式、发送模式和接收模式。一系列的时钟门控逻辑能够实现功耗模式。一个主时钟门（LL DSM 时钟门）逻辑提供了 LL_SYS 时钟，该时钟为子模块提供时钟脉冲，该子模块包含一个辅助时钟门（LL SM 时钟门）。辅助时钟门为 CORE 模块和全部其子模块提供了 LL_CORE 时钟。如果 LFCLK 源是 WCO，那么在深度睡眠模式下，为电源控制模块提供时钟脉冲的 lpo_16khz_clk 会保持时序参考。表 18-3 列出了 BLE 系统功耗模式的共存情况。

表 18-3. BLE 系统功耗模式共存表

BLESS 功耗模式	说明	系统功耗模式			应用情况
		DPSLP	SLP	ACT	
BLE.DSM	无线通信处于关闭状态。逻辑模块被供电。需要使能 WCO 来保持 LL 时序。可以使能被门控的 ECO 来执行快速唤醒。	✔	✔	✔	在长事件间隔中，BLE 会进入这种状态，这样可以最大限度节省电源。使用低频率时钟进行计时。根据其他外设活动，系统可以处于活动、睡眠或深度睡眠模式。
BLE.SM	无线通信处于关闭状态。逻辑模块被供电。可以使能 WCO。使能 ECO，但禁止提供给 LL 内核和数字调制解调器逻辑的时钟。		✔	✔	在短事件间隔中，BLE 会进入这种状态，这样可以节省电源。对于非 BLE 应用，BLE 也会进入这种状态以执行 AES-CCM 加密 / 解密操作。根据其他外设活动，系统可以处于活动或睡眠模式。
BLE.IDLE	无线通信处于关闭状态。在 ECO 给 LL 和 RD 逻辑提供时钟脉冲时，将激活逻辑模块。没有进行数据发送 / 接收操作。		✔	✔	在收发数据包之前以及之后，BLE 都会进入这种状态。在该状态中执行所有 FIFO 和寄存器配置。根据 BLE 或外设活动，系统可能处于活动或睡眠模式。
BLE.TX	LL 和 RD 逻辑处于活动状态。启动无线通信并进行无线发送。		✔	✔	进行无线发送时，BLE 进入这种状态。根据外设活动，系统可以处于活动或睡眠模式。
BLE.RX	LL 和 RD 逻辑处于活动状态。启动无线通信，并进行无线接收。		✔	✔	进行无线接收过程中，BLE 进入该状态。根据外设活动，系统可能处于活动或睡眠模式。

18.3.4.1 深度睡眠模式

深度睡眠模式是 BLESS 支持的最低功耗功能模式。在该模式下，无线功能被关闭。发送和接收完数据包后在广播、扫描和连接间隔期间将进入这种模式，这样可以最大限度地节省电源。LL_SYS 时钟被门控，或者 ECO 被关闭，这样能够节省电源。在需要链路层低功耗模式时序参考 / 唤醒逻辑的 BLE 应用中，必须将 LFCLK 设置为 WCO，并打开该时钟。在扩展深度睡眠模式下，如果 LF_16KHZ_CLK 被门控和 WCO 被关闭，可以将 LFCLK 设置为 ILO。根据 BLE 或外设活动，系统可以处于活动、睡眠或深度睡眠模式中。

18.3.4.2 睡眠模式

在睡眠模式下，无线功能被关闭。模块保持了所有配置内容。在各事件之间的短间隔内会进入这种模式。另外，这种模式也用于执行非 BLE AES-CCM 加密 / 解密活动。进入和退出该模式是由 CPU 控制的。对 LL CORE 时钟进行门控，以节省电源。根据 BLE 或外设活动，系统可以处于活动模式或者睡眠模式。

18.3.4.3 空闲模式

空闲模式是发送和接收模式的前置和后置状态。在这种状态下，无线功能被关闭，但链路层逻辑的链路层时钟处于使能状态，这样 CPU 可以启动协议状态机、配置和读取 FIFO 与寄存器。根据 BLE 或外设活动，系统可以处于活动模式或者睡眠模式。

18.3.4.4 发送模式

当无线模块开始传输数据时，将进入这种模式。调制器和无线发送器被打开。RF-PHY 接收来自链路层的速率为 1 Mbps 的串行数据，并将 2.4 GFSK 调制的数据发送到天线端口上。这时，BLE 会从空闲模式转入发送模式。根据 BLE 或外设活动，系统可以处于活动模式或者睡眠模式。

18.3.4.5 接收模式

当无线模块开始接收数据时，将转入这种模式。解调器和射频接收器被打开。RF-PHY 对来自 RF 模拟模块的 1 Mbps 数据进行解析，并在解调后将其转发给链路层控制器。根据 BLE 或外设活动情况，系统可以处于活动模式或者睡眠模式。

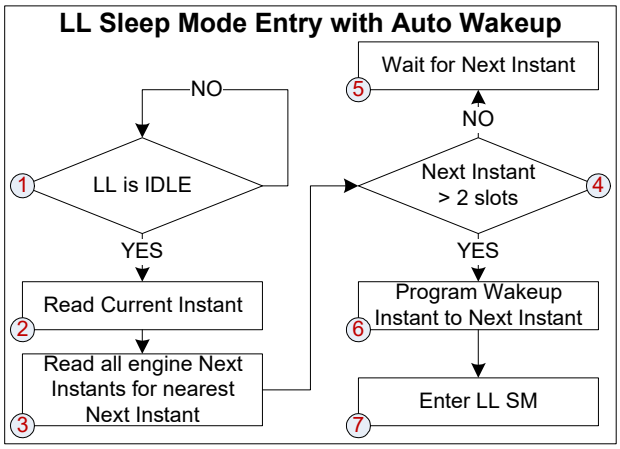
18.3.5 模式切换

本部分介绍了有关 BLESS 的各种模式切换序列。

18.3.5.1 LL 睡眠模式进入（具有自动唤醒功能）

LL 引擎处于空闲状态，并且下一个事件过程 （下一个事件）在至少两个 slot（1.25 ms）后，将进入这种模式。图 18-7 显示了该流程。

图 18-7. LL 睡眠模式进入 （具有自动唤醒功能）

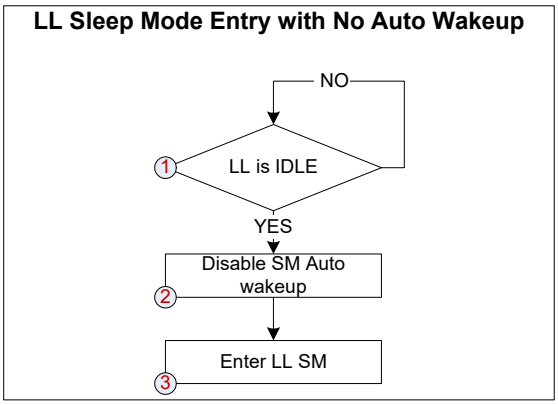


1. 通过读取BLE_BLELL_CLOCK_CONFIG寄存器的LLH_IDLE 字段检查 LL 是否处于空闲状态。
2. 如果LL处于空闲状态，那么会从BLE_BLELL_TIM_CONTROL_L 寄存器中读取当前事件。
3. 从相应的寄存器 BLE_BLELL_ADV_NEXT_INSTANT、BLE_BLELL_INIT_NEXT_INSTANT、BLE_BLELL_SCAN_NEXT_INSTANT 和 BLE_BLELL_NEXT_CE_INSTANT 中读取所有引擎后续的事件，并选择最接近的一个事件。
4. 检查下一个事件是否在小于两个 slot 后。
5. 如果只在不到两个 slot 后，则等待下一个事件。
6. 如果下一个事件在多于两个 slot 后，将编程该值，以便唤醒 BLE_BLELL_WAKEUP_CONTROL 寄存器中的事件。
7. 通过门控 LL SM 时钟门控器，实现对 CLK_CORE 的门控，从而进入睡眠模式。

18.3.5.2 LL 睡眠模式进入 （无自动唤醒功能）

如果无可预见的 BLE 活动，并且 LL 尚未准备好进入深度睡眠模式，可以转入该状态。要将 AES-CCM 使用于非低功耗蓝牙应用中时，还可以转入这种模式。图 18-8 显示了该流程。

图 18-8. LL 睡眠模式进入 （无自动唤醒功能）



1. 通过读取BLE_BLELL_CLOCK_CONFIG寄存器的LLH_IDLE 字段检查 LL 是否处于空闲状态。
2. 通过将 ‘0’ 写入到 BLE_BLELL_CLOCK_CONFIG.SM_AUTO_WKUP_EN字段内，可禁用睡眠模式自动唤醒功能。
3. 通过门控 LL SM 时钟门控器，实现对 CLK_CORE 的门控，从而进入睡眠模式。

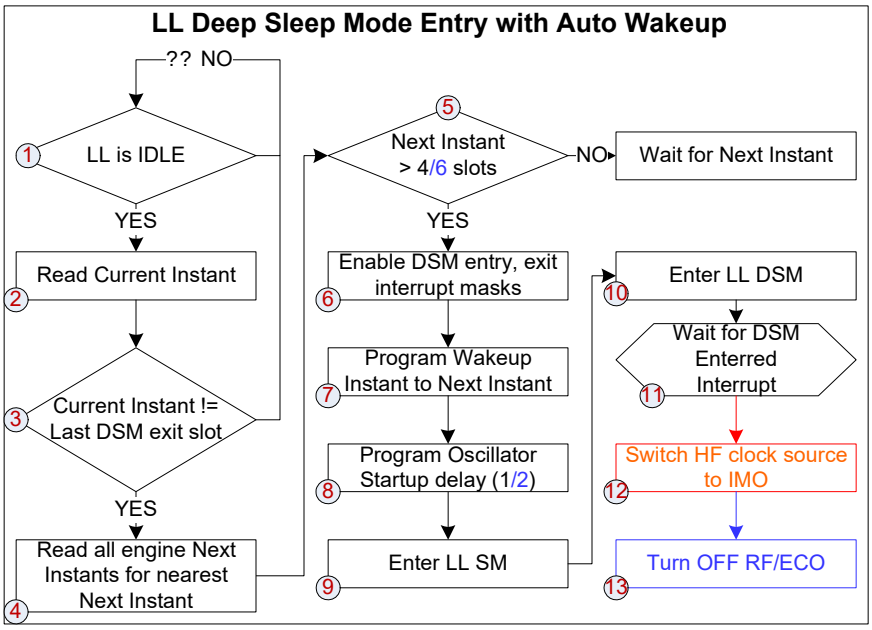
18.3.5.3 手动退出睡眠模式

需要使能 LL SM 时钟门控器来使能 CLK_CORE，这样才能手动退出无自动唤醒功能的 LL 睡眠模式。

18.3.5.4 LL 深度睡眠模式进入 （具有自动唤醒功能）

如果 ECO 计划保持 “ 打开 ” 状态或它至少在六个 slot （3.75 ms）后，或如果 ECO 计划被关闭，那么仅当 LL 引擎处于空闲状态并下一个事件至少在四个 slot （2.5 ms）后时，系统才能进入这种状态。图 18-9 显示了该流程；如果系统被调度进入深度睡眠模式或者 ECO 计划被关闭，那么会执行以红色显示的步骤。如果 ECO 计划被关闭，将执行以蓝色显示的步骤。

图 18-9. LL 深度睡眠模式进入 （具有自动唤醒功能）的流程图

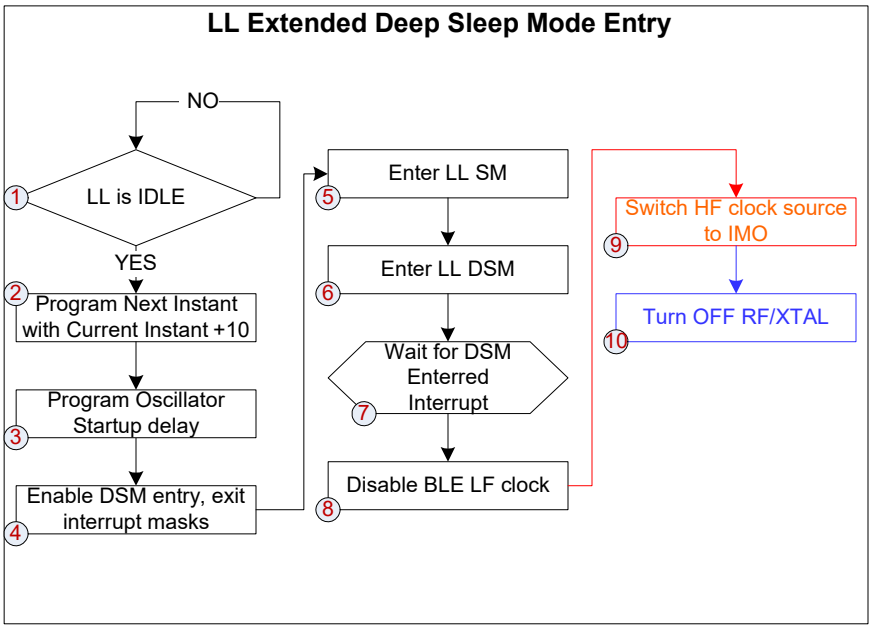


1. 通过读取BLE_BLELL_CLOCK_CONFIG寄存器的LLH_IDLE 字段检查 LL 是否处于空闲状态。
2. 如果LL处于空闲状态，那么会从BLE_BLELL_TIM_CONTROL_L 寄存器中读取当前事件。
3. 检查当前事件的 slot 与最后深度睡眠退出的 slot 是否相同。如果 slot 一样，那么将不允许进入 DSM 模式，因为进入该模式会影响 DSM 模式下 slot 的计数值。请转到第一步。
4. 如果 slot 不一样，那么从相应的寄存器 BLE_BLELL_ADV_NEXT_INSTANT、BLE_BLELL_INIT_NEXT_INSTANT、BLE_BLELL_SCAN_NEXT_INSTANT 和 BLE_BLELL_NEXT_CE_INSTANT 中读取所有引擎的下一个事件，并选择最近的那个事件。
5. 如果 RF/ECO 预计保持为打开状态，请检查下一个事件是否在小于 4 个 slot 后；或者如果 RF/ECO 预计被关闭，那么请检查下一个事件是否在 6 个 slot 后。如果下一事件在小于 4 或 6 个 slot 后，那么将等待下一个事件。
6. 通过在 BLE_BLESS_LL_DSM_CTRL 寄存器中设置 DSM_ENTERED_INTR_MASK 和 DSM_EXITED_INTR_MASK 字段来使能 DSM 进入并退出中断掩码。
7. 如果下一事件在大于 4 或 6 个 slot 后，那么会编程该值，以便唤醒BLE_BLELL_WAKEUP_CONTROL寄存器中的事件。
8. 编程 BLE_BLELL_WAKEUP_CONFIG.OSC_STARTUP_DELAY 寄存器字段中的振荡器启动延迟值：一个 slot，如果预计保持 RF/ECO 为打开，或两个 slot，如果预计关闭 RF/ECO
9. 通过门控 LL SM 时钟门控器，实现对 CLK_CORE 的门控，从而进入睡眠模式。
10. 通过将 ENTER_DSM 操作码指令写入到 BLE_BLELL_COMMAND_REGISTER 内，可以进入深度睡眠模式。
11. 等待通知已成功进入 DSM 模式的中断被确认，以确保成功完成 DSM 进入的切换。 BLE_BLESS_LL_DSM_INTR_STAT.DSM_ENTERED_INTR寄存器字段捕获了这种状态。
12. 如果将 HF 时钟设置为 ECO，将执行红色显示的步骤。这时， HF 时钟源被切换为 IMO。
13. 如果预计 RF/ECO 被关闭，则执行蓝色显示的步骤。

18.3.5.5 LL 进入扩展深度睡眠模式

预计在较长的时间 （几分钟或更长的）内不会使用 BLE 功能，或不需要 BLE 计时时，可以进入这种状态。所有链路层数据和时序的信息都会变得无效。图 18-10 显示了该流程。如果预定系统将进入深度睡眠模式或 ECO 被预设为关闭状态，那么会执行红色显示的步骤。如果预定设置 ECO 为关闭状态，则执行蓝色显示的步骤。

图 18-10. LL 进入扩展深度睡眠模式的流程图

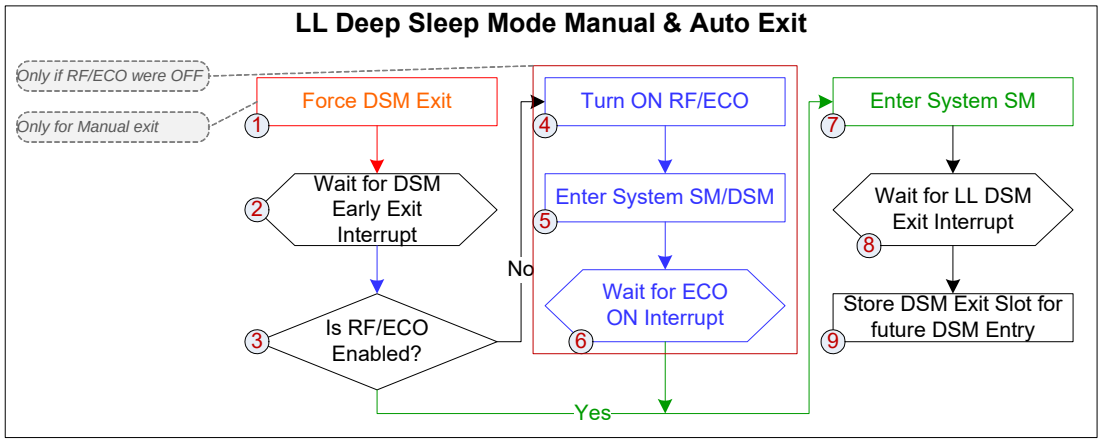


1. 通过读取BLE_BLELL_CLOCK_CONFIG寄存器的LLH_IDLE 字段检查 LL 是否处于空闲状态。
2. 如果 LL 处于空闲状态，那么从 BLE_BLELL_TIM_CONTROL_L 寄存器中读取当前的事件，并将当前事件值 + 10，以便唤醒 BLE_BLELL_WAKEUP_CONTROL 寄存器中的事件。
3. 编程 BLE_BLELL_WAKEUP_CONFIG.OSC_STARTUP_DELAY 寄存器字段中的振荡器启动延迟加 2 个 slot
4. 通过在 BLE_BLESS_LL_DSM_CTRL 寄存器中设置 DSM_ENTERED_INTR_MASK 和 DSM_EXITED_INTR_MASK 字段来使能 DSM 进入并退出中断掩码。
5. 通过门控 LL SM 时钟门控器，实现对 CLK_CORE 的门控，从而进入睡眠模式。
6. 通过将 ENTER_DSM 操作码指令写入到 BLE_BLELL_COMMAND_REGISTER 内，可以进入深度睡眠模式。
7. 等待通知已成功进入 DSM 模式的中断被确认，以确保成功完成 DSM 输入的切换。 BLE_BLESS_LL_DSM_INTR_STAT.DSM_ENTERED_INTR寄存器字段捕获了这种状态。
8. 通过设置 BLE_BLESS_LF_CLK_CTRL.DISABLE_LF_CLK 寄存器字段来禁用 BLE LF 时钟（即 LF_16KHZ_CLK）。这样可以防止运行低功耗状态机。
9. 如果将 HF 时钟设置为 ECO，将执行红色显示的步骤。这时， HF 时钟源被切换为 IMO。
10. 如果预计 RF/ECO 被关闭，则执行蓝色显示的步骤。

18.3.5.6 LL 深度睡眠模式手动退出 / 自动退出

为了手动从 LL 深度睡眠模式唤醒，可以退出 DSM 模式，这样会开始唤醒序列。在自动唤醒模式下，内部超时会开始唤醒序列。图 18-11 显示了该流程；如果 ECO 被关闭，将执行蓝色显示的步骤。如果 CPU 不用执行其他操作，并且唤醒序列很长，则执行绿色显示的步骤。

图 18-11. 手动和自动退出 LL 深度睡眠模式的流程图



1. 如果需要手动退出，那么设置 BLE_BLESS_LL_DSM_CTRL.DSM_EXIT 寄存器字段，以便强制退出 DSM 模式。
2. 对于手动和自动退出，需要等待 DSM 早些退出中断。BLE_BLESS_LL_DSM_INTR_STAT.DSM_ENTERED_INTR 寄存器字段捕获了这种状态。
3. 检查 RF/ECO 是否已经被使能。

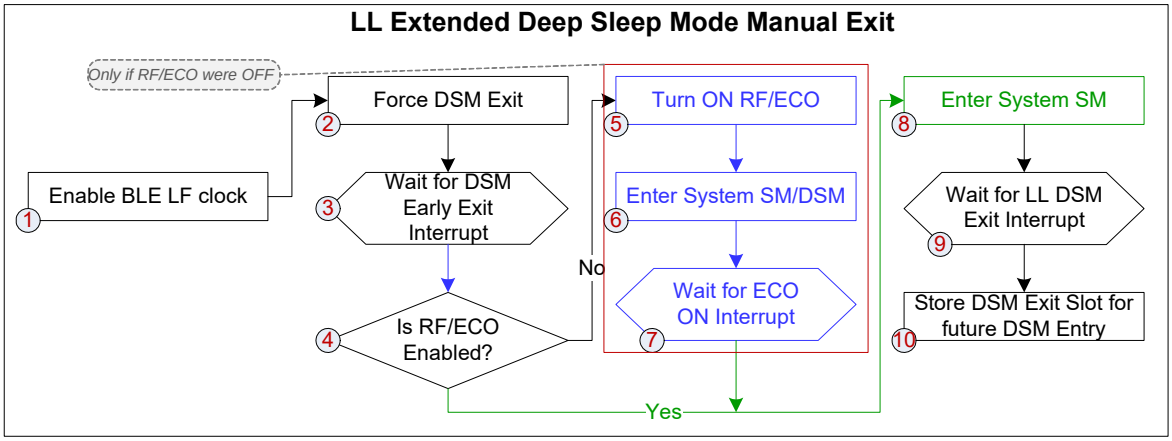
如果 RF/ECO 被关闭，则需要执行下面的子序列 （第四步到第六步）。

4. 使能 RF/ECO。
5. 如果CPU不需要执行其他操作，那么通过设置BLE_BLESS_LL_DSM_INTR.XTAL_ON_INTR_MASK寄存器字段使能ECO ON 中断 （如果以前未使能它）后，将进入系统睡眠模式或深度睡眠模式。
6. 等待 ECO ON 中断被确认；如果在第五步中系统已经进入低功耗模式，将唤醒系统。在 BLE_BLESS_LL_DSM_INTR_STAT.XTAL_ON_INTR 寄存器字段中反射了这种状态。
7. 发生ECO ON中断后，或如果在第三步中执行 ‘n’ 检查时ECO被使能，并CPU无需执行其他操作，那么将进入睡眠模式。
8. 等待 LL DSM 退出的中断被确认；如果在第七步中系统已经进入睡眠模式，将唤醒系统。BLE_BLELL_EVENT_INTR.DSM_INTR 中捕获了这种状态。
9. 退出时，会存储当前 slot 值 （即 DSM 退出 slot），以使用于将来 DSM 进入调用。

18.3.5.7 LL 扩展深度睡眠模式手动退出

为了手动从 LL 扩展深度睡眠模式唤醒，必须使能 LF 时钟，并退出 DSM 模式，这样会开始唤醒序列。图 18-12 显示该流程；如果 RF/ECO 被关闭，则执行蓝色显示的步骤。如果 CPU 不用执行其他操作，并且唤醒序列很长，则执行绿色显示的步骤。

图 18-12. 手动退出 LL 扩展深度睡眠模式的流程图



1. 通过清除 BLE_BLESS_LF_CLK_CTRL.DISABLE_LF_-CLK 寄存器字段，可使能 BLE LF 时钟 （即 LF_16KHZ_CLK）。这样能够开始低功耗状态机运行。
2. 设置 BLE_BLESS_LL_DSM_CTRL.DSM_EXIT 寄存器字段，以便强制退出 DSM 模式。
3. 对于手动和自动退出，需要等待 DSM 早些退出中断。BLE_BLESS_LL_DSM_INTR_STAT.DSM_EN-TERED_INTR 寄存器字段捕获了这种状态。
4. 检查 RF/ECO 是否已经被使能。

如果 RF/ECO 被关闭，则需要执行下面的子序列 （第五步到第七步）。

5. 使能 RF/ECO。
6. 如果 CPU 不需要执行其他操作，那么通过设置 BLE_B-LESS_LL_DSM_INTR.XTAL_ON_INTR_MASK寄存器字段使能 ECO ON 中断 （如果以前未使能它）后，将进入系统睡眠模式或深度睡眠模式。
7. 等待 ECO ON 中断被确认；如果在第五步中系统已经进入低功耗模式，将唤醒系统。在 BLE_BLESS_LL_DS-M_INTR_STAT.XTAL_ON_INTR 寄存器字段中反射了这种状态。
8. 发生 ECO ON 中断后，或如果在第三步中执行 ‘n’ 检查时 ECO 被使能，并 CPU 无需执行其他操作，那么将进入睡眠模式。
9. 等待 LL DSM 退出中断被确认；如果在第七步中系统已经进入低功耗模式，将唤醒系统。BLE_-BLELL_EVENT_INTR.DSM_INTR 中捕获了这种状态。
10. 退出时，会存储当前 slot 值 （即 DSM 退出 slot），以使用于将来 DSM 进入调用。

18.3.6 蓝牙 4.2 特性 — 数据长度扩展

数据长度扩展是 BLE 4.2 功能，用于将 LL 数据 PDU 的最大支持长度从 27 个八位字节增加到 251 个八位字节。通过设置 BLE_BLELL_LL_CONTROL_DLE，可在子系统中启用该功能。使能该功能后，在连接期间的发送和接收数据包长度将增加到 251 个字节。加密模块还支持 251 字节的数据包加密和解密。

注意：该功能仅适用于 PSoC 41x8-BL5xx 和 PSoC 42x8-BL5xx 系列。

18.3.7 蓝牙 4.2 特性 — 保密 1.2

保密 1.2 是一种 BLE 4.2 功能，其中私有地址的生成和解析都在 BLE 链路层中实现。该功能在 BLESS 硬件和链路层固件之间进行分区。

通过设置 BLE_BLELL_LL_CONTROL_PRIV_1_2，可在子系统中启用该功能。

注意：该功能仅适用于 PSoC 41x8-BL5xx 和 PSoC 42x8-BL5xx 系列。

18.3.7.1 解析表

链路层维护了本地和对等 IRK 值对的一组记录，被称为解析列表。解析列表包含以下字段。

- 对等识别地址：对等设备的公共或静态地址。
- 局部识别解析密钥 （IRK）：是与关联的对等设备共享的并用于生成自解析私有地址 （RPA）的密钥。
- 对等 IRK：是由关联的对等设备共享并用于解析对等 RPA 的密钥。

解析列表在固件和硬件之间实现，所实现的硬件分辨率列表如表 18-4 中所示。固件将维护主机层检测到并使用的解析列表，但固件将使用硬件端解析列表来检索和管理操作列表。

表 18-4. 解析列表

Valid Entry	Peer Privacy level	Self Privacy Level	Peer Whitelist status	Peer ID addr type	Peer RPA valid	Rcvd self RPA valid	TX self RPA valid	INIT specific device	Self TX Addr type	Peer idnty addr 48 bits	Peer RPA 48 bits	Rcvd Self RPA 48 bits	TX self RPA 48 bits

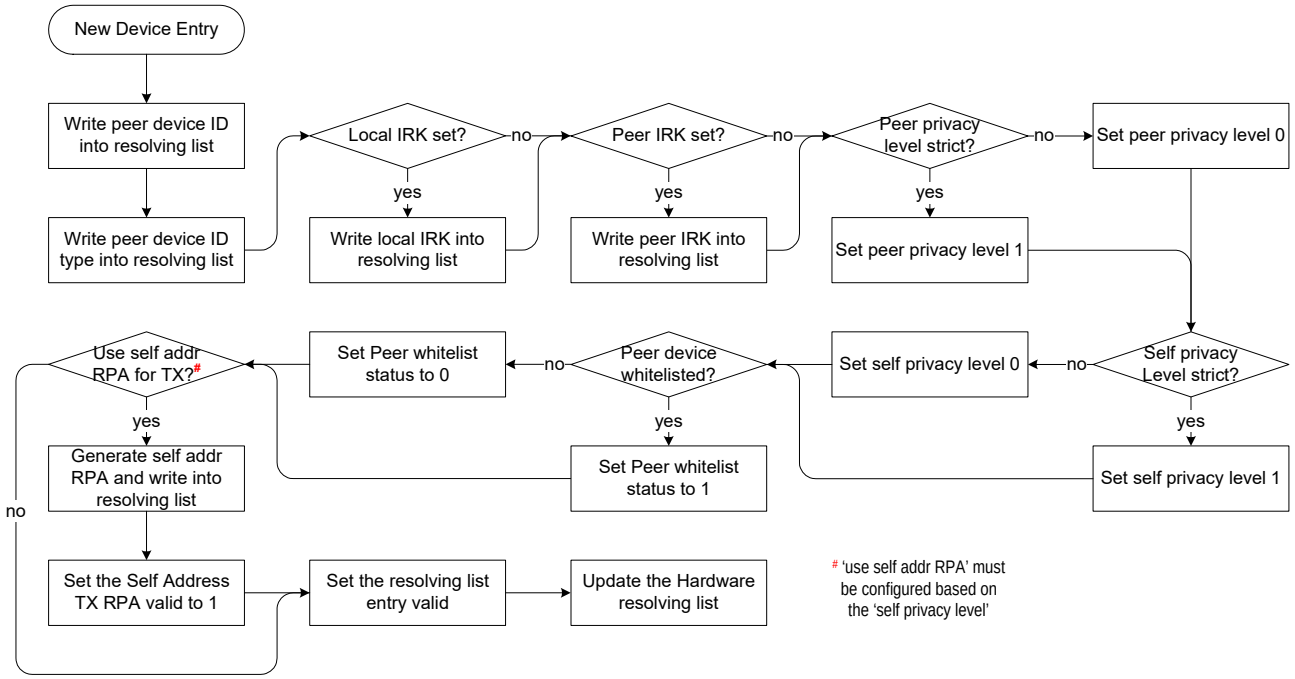
18.3.7.2 解析列表功能

以下是应由链路层固件管理的并与保密相关的通用流程。

将设备添加到解析表内

要将设备添加到解析列表内，固件必须将 DevA 添加到硬件解析列表中，设置适当的标志，并填充设备识别地址和 IRK。请参见图 18-13，了解固件程序指南。

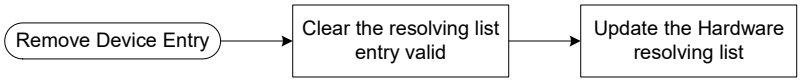
图 18-13. 将设备添加到解析表内



从解析表移除一个设备

要从解析列表中删除设备，固件必须将硬件解析列表中的设备地址的 BLE_BLELL_RSLV_LIST_ENABLE_VALID_ENTRY 位设置为 ‘0’。此时，整行的内容都变为无效，并且可以重新使用该行来存储另一个设备的条目。请参见图 18-14，了解固件程序指南。

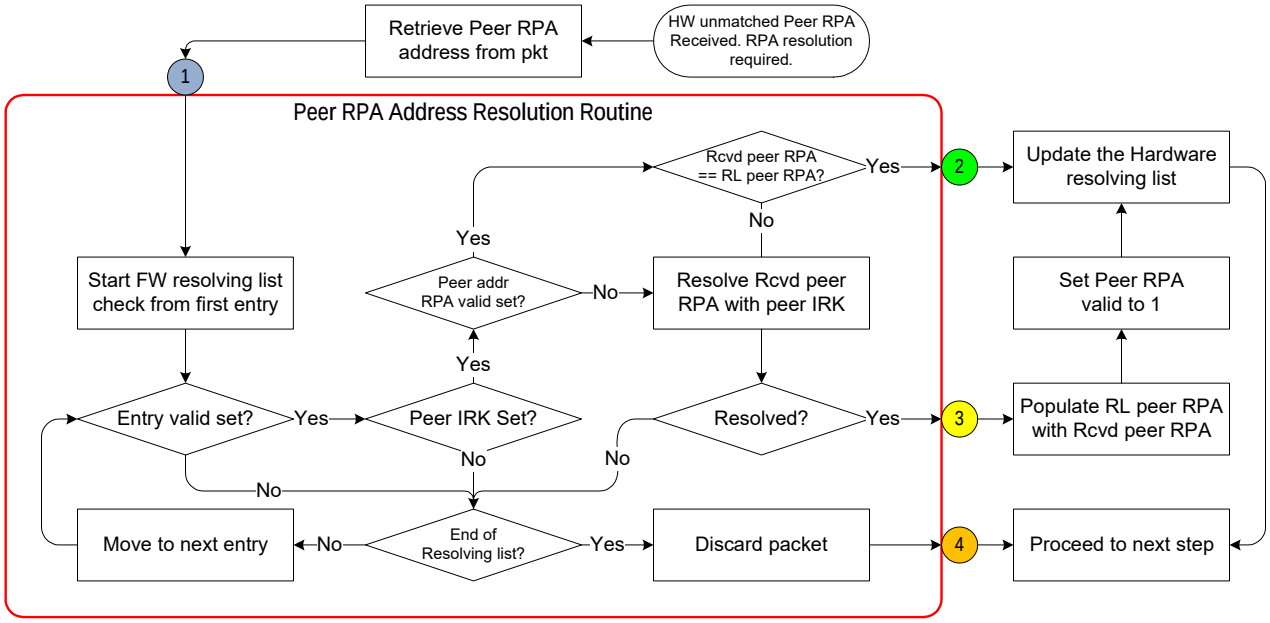
图 18-14. 从解析表移除一个设备



处理未解析的对等 RPA

当接收到具有对等地址 RPA 的数据包时，将 RPA 与所有有效行的 PEER_RPA 字段进行比较（设置 BLE_BLELL_RSLV_LIST_ENABLE_VALID_ENTRY 和 BLE_BLELL_RSLV_LIST_ENABLE_PEER_ADDR_RPA_VAL 位）。如果没有任何有效的条目，则表示该地址仍未被解析。硬件会中断固件并提供接收到的对等 RPA 以进行解析。请参见图 18-15，了解对等地址解析的固件程序指南。

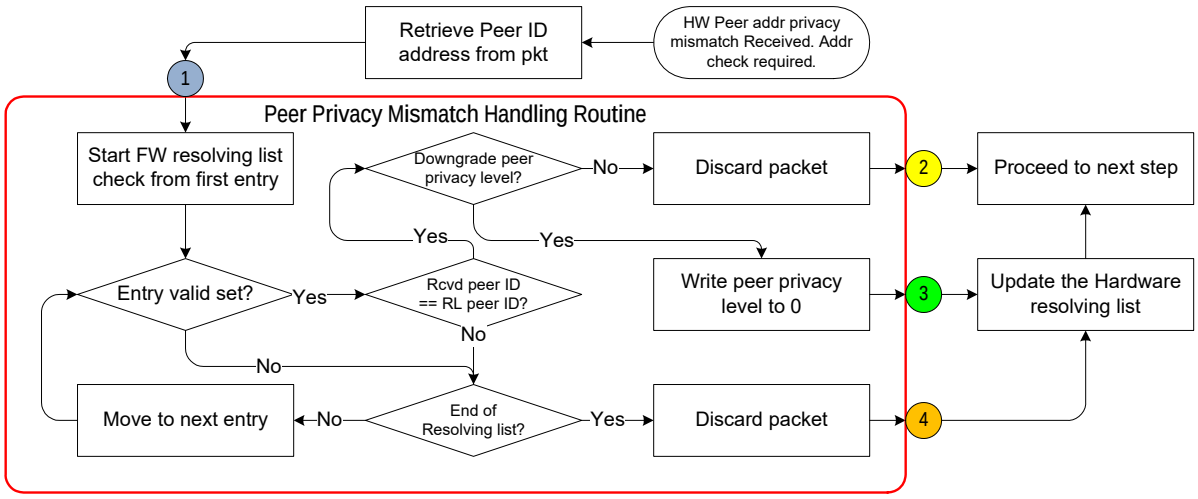
图 18-15. 对等 RPA 地址解析



18.3.7.3 处理不使用 RPA 的对等设备

当接收到具有对等识别地址的数据包但是所需的类型是 RPA（对等隐私级别被设置为 1）时，将报告对等地址隐私不匹配。固件必须检查解析列表以确保接收的对等 ID 地址相匹配，同时降低隐私级别以允许接收具有对等识别地址的数据包，或者拒绝保密违规的数据包。保留该选项，以允许将来在处理此类设备时更改规范。

图 18-16. 对等隐私界别不匹配的处理流程

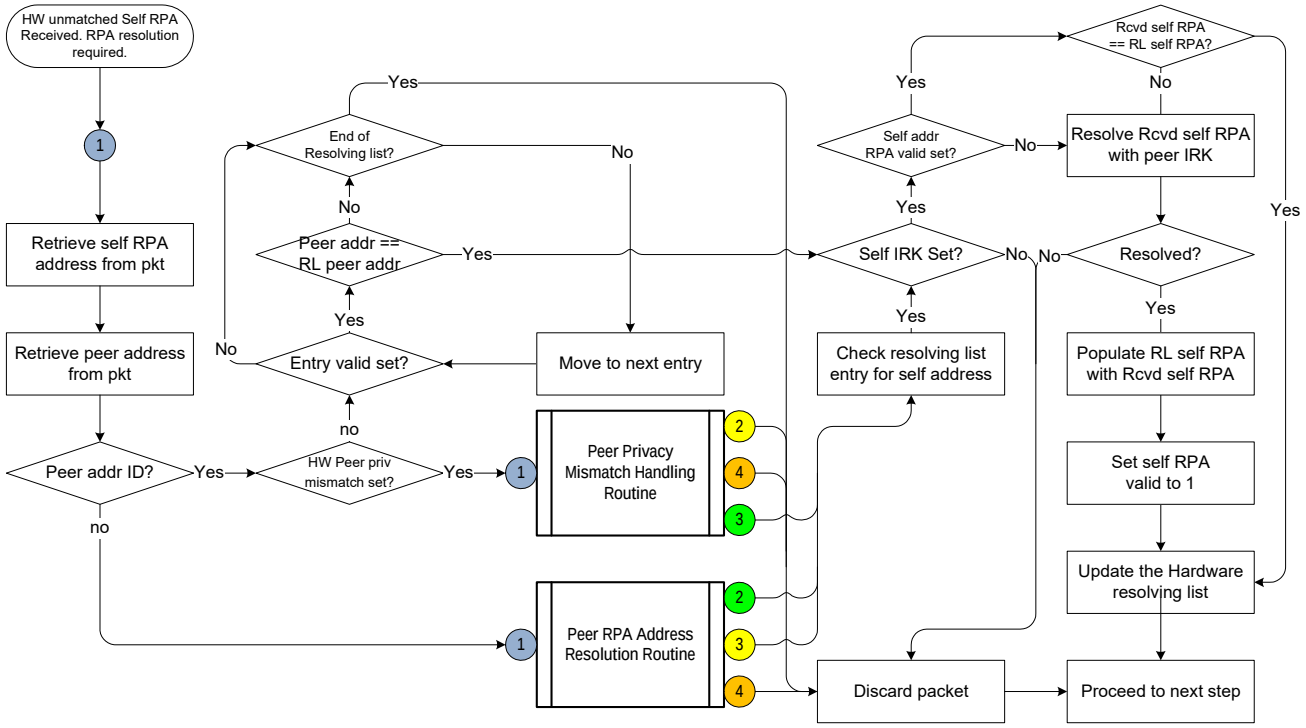


18.3.7.4 处理未解析的自身 RPA

当接收到具有自身地址 RPA 的数据包时，如果条目与对等地址 （ID 或 RPA）相匹配，则将接收到的自身 RPA 与匹配行的自身 RPA 字段进行比较。如果自身 RPA 不匹配或者对等地址与自身 RPA 都不匹配，则会引发自身地址不匹配中断 （以及相应的对等地址中断）。

固件首先必须匹配对等地址 （基于前面的部分）。

图 18-17. 不匹配的自身 RPA 地址解析



18.4 寄存器详细信息

表 18-5 提供了本章节中所介绍的全部寄存器字段的详细信息。欲了解更多信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器的技术参考手册。

表 18-5. 寄存器列表

寄存器	位	字段名称	位说明
BLE_BLELL_ADV_NEXT_INSTANT	15:0	ADV_NEXT_INSTANT	显示了在参考内部参考时钟的情况下广播事件的下一个启动。
BLE_BLELL_CLOCK_CONFIG	7	LLH_IDLE	指示硬件是否正在进行任意的数据发送 / 接收操作。固件通过使用该信息决定将硬件编程为深度睡眠模式。 1 — LL 硬件处于休闲状态。 0 — LL 硬件处于繁忙状态。在这种情况下，即使固件提供了一条进入 DSM 的命令，LL 硬件也不会进入深度睡眠模式。（硬件生成一个 dsm 退出中断，用于通知固件：未能成功进入 DSM 模式）。
	10	SM_AUTO_WKUP_EN	使能睡眠模式自动唤醒使能功能。 1：使能； 0：禁用。 在 *wakeup_instant – sm_offset_to_wakeup_instant* 时使能硬件，使其自动从深度睡眠唤醒。 *wakeup_insant* 是上述*唤醒控制寄存器*中的字段。 *sm_offset_to_wakeup_instant* 值是唤醒配置寄存器中的字段。
BLE_BLELL_COMMAND_REGISTER	7:0	COMMAND	从固件发送到链路层控制器的 8 位命令。有关命令及其操作码的信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册。该命令会使链路层硬件启动 / 停止一个操作。
BLE_BLELL_NEXT_CE_INSTANT	15:0	NEXT_CE_INSTANT	16 位内部参考时钟值，在该值中会发生下一个连接事件。读取寄存器前，需要将连接索引写入到连接索引寄存器内。
BLE_BLELL_EVENT_INTR	5	DSM_INTR	**读取：** 深度睡眠模式退出中断。链路层硬件退出深度睡眠模式时，会设置该位。 **写入：** 清除深度睡眠模式退出中断。将寄存器中的该位设置为 1，以清除中断源。
BLE_BLELL_INIT_NEXT_INSTANT	15:0	INIT_NEXT_INSTANT	显示的是相对于分辨率为 625 µs 的内部参考时钟的时刻，这时将开始下一个初始扫描事件。

表 18-5. 寄存器列表

寄存器	位	字段名称	位说明
BLE_-BLELL_POC_REG_TIM_CTRL	7:3	BB_CLK_FREQ_MINUS_1	LLH 时钟配置。在该寄存器上对设计中时钟输入的时钟频率进行配置。这样可以派生一个 1 MHz 的时钟。
BLE_BLELL_SCAN_NEXT_INSTANT	15:0	NEXT_SCAN_INSTANT	显示的是相对于分辨率为 625 µs 的内部参考时钟的时刻，这时将开始下一个扫描事件。
BLE_BLELL_TIM_-CONTROL_L	15:0	TIM_REF_CLOCK	16 位内部参考时钟。该时钟为自由运行的时钟，并由一个周期为 0.625 ms 的脉冲递增。它作为一个参考时钟使用，以根据协议派生出全部时序。
BLE_BLELL_WAKEUP_CONFIG	7:0	OSC_STARTUP_DELAY	振荡器稳定 / 启动延迟。这是 X.Y 格式，其中 X 是 625 µs 的 BT“slot”（时隙）数量， Y 是频率为 16.384 kHz 时钟输入的时钟周期数量。打开 RF 振荡器后，它需要一个频率为 16.384 kHz 的时钟用以保持其稳定性。在该周期内，假设时钟不稳定，所以直到该周期完成，控制器才会打开内部逻辑的时钟。这意味着，从深度睡眠模式唤醒要考虑到发生唤醒前的延迟。 Osc_startup_delay[7:5] 是 625 µs 的 slot （时隙）的数量 Osc_startup_delay[4:0] 是频率为 16 KHz 时钟的时钟周期数量 （警告：支持 Osc_startup_delay [4:0] 的最小值为 1 和最大值为 9。因此，可编程范围为 1 到 9）
BLE_BLELL_WAKEUP_CONTROL	15:0	WAKEUP_INSTANT	在参考内部 16 位时钟参考时，必须使硬件从深度睡眠模式唤醒。当需要控制器操作 （如广播者、扫描者）时，固件会根据下一个最接近的事件对其进行计算。固件在相应的 *_NEXT_INSTANT 寄存器上读取程序的下一个事件。只有在时钟控制寄存器上使能硬件自动从深度睡眠模式唤醒功能时，才能使用该值。 注意：建议进入 DSM 时，对 “wakeup_instant” 进行编程，以便使要唤醒的实际时长比参考时钟至少多两个计数值 （两个 625 us 的 slot） 。实际需要唤醒的时长为 “wakeup_instant – dsm_offset_to_wakeup_instant – osc_startup_delay”，该时长要大于 “ 参考时钟 + 2”
BLE_BLERD_DBUS	15	XTAL_ENABLE	使能晶振。高电平有效。
BLE_BLESS_LF_-CLK_CTRL	0	DISABLE_LF_CLK	设置为 1 时，阻止 LF 时钟输入连接到链路层。可对扩展的 DSM 模式执行该操作，其中需要冻结 DSM 状态机以防止默认自动退出。
BLE_BLESS_LL_-CLK_EN	0	clk_en	将该位设置为 ‘1’ 以使能提供给链路层的时钟。
BLE_BLESS_LL_DSM_CTRL	0	DSM_EXIT	链路层处于深度睡眠模式时，固件可通过设置该位来唤醒链路层。
	1	DSM_ENTERED_INTR_MASK	当 DSM 进入中断被禁用时，则屏蔽它。
	2	DSM_EXITED_INTR_MASK	当 DSM 退出中断被禁用时，则屏蔽它。
	3	XTAL_ON_INTR_MASK	当晶振稳定中断被禁用时，则屏蔽它。
BLE_BLESS_LL_DSM_INTR_STAT	0	DSM_ENTERED_INTR	将固件请求发送到 LL 使 LL 进入状态机（该状态机使用 LF 时钟）时，LL 会进入深度睡眠模式并会激活该中断。可通过将数值 “1” 写入到该位置来清除中断。
	1	DSM_EXITED_INTR	通过将固件请求发送到 LL 来要求 LL 退出深度睡眠模式（该模式正在使用 LF 时钟）时，LL 会退出深度睡眠模式，并且在深度睡眠时钟门控器被打开时它会激活该中断。可通过将数值 “1” 写入到该寄存器字段来清除中断。
	8	XTAL_ON_INTR	使能晶振稳定信号上升沿中断。可通过将数值 “1” 写入到该寄存器字段来清除中断。
BLE_BLESS_RF_-CONFIG	0	RF_ENABLE	使能 RF 振荡器带隙。 1：带隙被使能 0：带隙被禁用
BLE_BLESS_WCO_-CONFIG	31	ENABLE	WCO 振荡器的主设备使能。
BLE_BLESS_XTAL_-CLK_DIV_CONFIG	1:0	SYSCLK_DIV	系统时钟预分频器值。对频率为 24 MHz 的晶振时钟进行分频，以生成系统时钟。 0：NO_DIV: SYSCLK = XTALCLK/1 1：DIV_BY_2: SYSCLK = XTALCLK/2 2：DIV_BY_4: SYSCLK = XTALCLK/4 3：DIV_BY_8: SYSCLK = XTALCLK/8
	3:2	LLCLK_DIV	链路层时钟预分频器的值。对频率为 24 MHz 的晶振时钟进行分频，以生成链路层时钟。 0：NO_DIV: LLCLK = XTALCLK/1 1：DIV_BY_2: LLCLK = XTALCLK/2 2：DIV_BY_4: LLCLK = XTALCLK/4 3：DIV_BY_8: LLCLK = XTALCLK/8

表 18-5. 寄存器列表

寄存器	位	字段名称	位说明
WDT_CONFIG	1:0	WDT_MODE0	发生匹配情况 （即 WDT_CTR0 = WDT_MATCH0）时的看门狗计数器操作。
		NOTHING	无操作
		INT	设置 WDT_INTx
		复位	设置 WDT 复位
		INT_THEN_RESET	设置 WDT_INTx，并在未处理的第三个中断后设置 WDT 复位
	2	WDT_CLEAR0	当 WDT_CTR0 = WDT_MATCH0 时，清除看门狗计数器。换句话说， WDT_CTR0 将对 LFCLK 进行 (WDT_MATCH0+1) 分频。 0：自由运行计数器 1：匹配时进行清除
	3	WDT_CASCADE0_1	级联看门狗计数器 0、 1。 WDT_CTR0 = WDT_MATCH0 后，计数器 1 会递增。 0：独立计数器 1：级联计数器
	9:8	WDT_MODE1	发生匹配情况 （即 WDT_CTR1=WDT_MATCH1）时的看门狗计数器操作。
		NOTHING	无操作
		INT	设置 WDT_INTx
		复位	设置 WDT 复位
		INT_THEN_RESET	设置 WDT_INTx，并在未处理的第三个中断后设置 WDT 复位
	10	WDT_CLEAR1	当 WDT_CTR1 = WDT_MATCH1 时，会清除看门狗计数器。换句话说， WDT_CTR1 将 LFCLK 值除以 (WDT_MATCH1+1)。 0：自由运行计数器 1：匹配时进行清除
	11	WDT_CASCADE1_2	对看门狗计数器 1、2 进行级联。 WDT_CTR1 = WDT_MATCH1 后，计数器 2 将递增计数。因此，可以对三个 WDT 计数器进行级联。 0：独立计数器 1：级联计数器
	16	WDT_MODE2	看门狗计数器 2 模式。
		NOTHING	无中断请求的自由运行计数器。
		INT	带计数器 2 中的特殊位切换时，自由运行计数器会发生中断请求 （请参考 WDT_BITS2）。
	28:24	WDT_BITS2	用于观察 WDT_INT2 的位： 0：当 WDT_CTR2 的位 0 被切换 （每一次切换发生一个中断）时会设置它 .. 31：当 WDT_CTR2 的位 31 被切换 （每 2^31 次切换发生一个中断）时会设置它
	31:30	LFCLK_SEL	选择 LFCLK 源： 0：ILO — 内部 R/C 振荡器 1：WCO — 内部晶体振荡器 2-3：保留 — 不使用 并非所有产品都支持全部时钟源。选择不受支持的时钟源可能会引起不良操作。为了安全更改 LFCLK_SEL，需要等待 WDT_CTRLOW/WDT_CTRHIGH 更改，然后立即更改该设置。

表 18-5. 寄存器列表

寄存器	位	字段名称	位说明
RSLV_LIST_ENABLE（4.2）[a]	0	VALID_ENTRY	指示索引是否有效。
	1	PEER_ADDR_IRK_SET	指示所列出的对等设备是否已共享其 IRK。 0 — 接收已收到的数据包中的识别地址。如果列表中列有有效的对等设备 RPA，则接收所收到的数据包中的 RPA。 1 — 只接收所收到的数据包中的对等设备 RPA（如果列表中罗列了该 RPA）。已收到的数据包中的识别地址被报告为隐私未匹配。
	2	SELF_ADDR_IRK_SET_RX	指示是否与所列出的对等设备共享局部 IRK。 0 — 接收已收到的数据包中的自识别地址。如果列表中列出了有效的自身 RPA，则接收所收到的数据包中的 RPA。 1 — 只接收所收到的数据包中的自身 RPA（如果列表中罗列了该 RPA）。已收到的数据包中的自识别地址被报告为隐私未匹配。
	3	WHITELISTED_PEER	指示列出的对等设备是否在白名单中。
	4	PEER_ADDR_TYPE	指示所列出的对等设备的地址类型。
	5	PEER_ADDR_RPA_VAL	表示列表中的对等设备 RPA 有效。
	6	SELF_ADDR_RXD_RPA_VAL	表示列表中接收到的自身 RPA 有效。
	7	SELF_ADDR_TX_RPA_VAL	表示列表中被发送的自身 RPA 有效。
	8	SELF_ADDR_INIT_RPA_SEL	禁用 Initiator 白名单时，该位指示将从其接收 ADV 数据包的特定设备。
	9	SELF_ADDR_IRK_SET_RX	指示用于 SCANA 和 INITA 的 TX 地址类型。 0 — SCAN_REQ/CONN_REQ 数据包的 SCANA/INITA 中使用自身识别地址 1 — 解析列表中 RSLV_LIST_TX_INIT_RPA 字段 （带有上面 SELF_ADDR_TX_RPA_VAL 中的相关有效位）中提供的自身 RPA 地址被用于 SCAN_REQ/CONN_REQ 数据包的 SCANA/INITA 内
LL_CONTROL（4.2）[a]	0	PRIV_1_2	使能保密 1.2 功能
	1	DLE	使能 DTM、连接和加密模块中的数据长度扩展功能。

a. 只有蓝牙 4.2 设备支持这些寄存器。

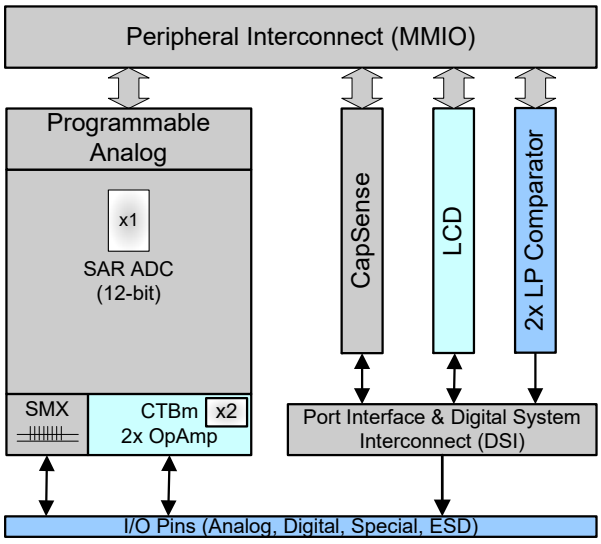
章节E: 模拟系统

本部分包括以下章节：

- 第 247 页上的高精度参考章节
- 第 251 页上的 SAR ADC 章节
- 第 283 页上的低功耗比较器章节
- 第 289 页上的微型连续时间模块 （CTBm）章节
- 第 299 页上的 LCD 直接驱动章节
- 第 311 页上的 CapSense 章节
- 第 321 页上的温度传感器章节

系统架构

模拟系统框图



18. 高精度参考

PSoC® 4 具有一个高精度参考模块，它可以为整个芯片创建多个参考用的偏置电压和电流。在器件温度范围内，该模块为内部主振荡器 （IMO）电路和闪存模块提供了随温度变化的参考电压，以分别为它们保证准确的 IMO 输出频率和无错误的闪存读 / 写操作。

18.1 特性

高精度参考模块具有以下特性：

- 带隙电路，可生成 1.024 V 和 2.4 µA 的参考电源
- 调整缓冲区，可从带隙电路的输入生成 1.2 V、 1.024 V 和 0.8 V 等多种输出电压
- 多个快速和慢速的低功耗缓冲区，它们不仅提高了各种参考输出的驱动能力，并且还可以将噪声彼此隔离开
- 多个快速和慢速的电流镜像电路
- 随温度变化的参考电压，用于闪存存储器
- 随温度变化的参考电流，用于 IMO

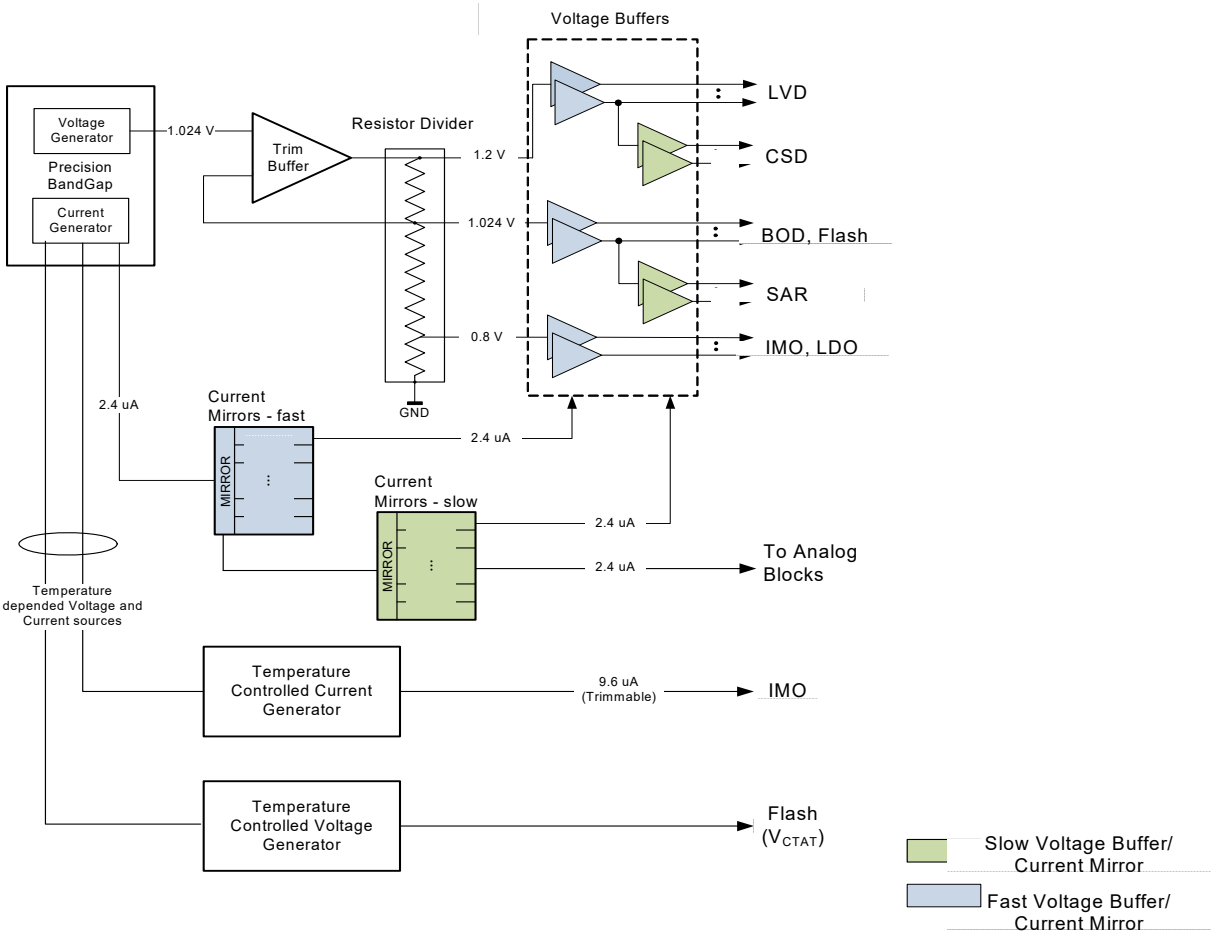
18.2 框 图

图 18-1 显示了参考电压框图。

高精度参考主要包括以下模块：

- 一个高精度带隙模块，用于生成精确的参考电压和参考电流
- 一个调整缓冲区，它为各种应用程序生成不同的输出参考电压并调整 1.024 V 的输出电压
- 它是一组快速低功耗缓冲区和慢速低功耗缓冲区。它不仅提高了各种参考输出的驱动能力，而且还可以将噪声彼此隔离开
- 一组快速元件和慢速元件，可分别在快速域和慢速域中创建多个参考电流的副本
- 一个由温度控制的电压发生器模块，用于闪存系统
- 一个由温度控制的电流源，用于 IMO

图 18-1. 参考电压框图



18.3 工作原理

本节详细介绍了主要组件的工作原理。

18.3.1 高精度带隙

该电路是高精度参考模块所生成的所有参考电压的源。它提供了经过第二阶曲率纠正后的 1.024 V 参考电压和 2.4 µA 的参考电流。参考电压被路由到调整缓冲区，而参考电流则被输送给电流镜像电路。

18.3.2 调整缓冲区

调整缓冲区是一个运算放大器网络，它从带隙电路中获取输入信号，然后生成三种不同的参考电压（分别为 1.2 V、1.024 V 和 0.8 V）。参考电压在反馈网络中的电阻阵列处分接，从而产生高输出阻抗。因此，必须使用缓冲区来驱动参考电压。

18.3.3 低功耗缓冲区

PSoC 4 具有多个低功耗缓冲区，并分为两组：快速和慢速。这些缓冲区从调整缓冲区电路获取输入，并驱动到目标模块。快速缓冲区在 9 us 的时间内可以达到最终值，差值为 1%。而慢速缓冲区则需要 40 µs 的时间。多缓冲区可以确保低参考线电容，从而降低建立时间。快速电压缓冲区用于输出给对系统启动过程起着重要作用的模块的参考电压。它们包括 IMO、闪存、低压差电压调节器 （LDO）、低压检测 （LVD）以及欠压检测 （BOD）电路。

快速缓冲区的输出被驱动给慢速缓冲区。这样可以确保将相关未启动模块引起的额外负载与快速缓冲区驱动的模块隔离开。慢速缓冲区驱动 SAR ADC 和 CapSense CSD 等功能模块。

快速缓冲区始终随着带隙模块的使能而被使能，而用户通过使用 PWR_BG_CONFIG 寄存器中的 VREF_EN 位可以单独使能或禁用慢速缓冲区。

18.3.4 电流镜像

通过使用电流镜像电路可以从带隙电路中生成 2.4 µA 参考电流的多个副本。与电压缓冲区相似，电流镜像也分为两种类型：快速和慢速。快速电流镜像电路经过 9 µs 的建立时间可以达到最终电压值，差值为 1% ； 而慢速电流镜像电路则需要 40 µs 才能达到。通过使用快速电流镜像可以为快速电压缓冲区提供偏置电压。慢速电流镜像的各个输出用于驱动 SAR、CTBm、CSD 和 LPCOMP 等模拟模块。快速电流镜像中的 2.4 µA 源电流被输送给闪存模块。

18.3.5 温度控制的电压发生器

由该模块生成的偏置信号会根据温度来控制闪存存储器的参考电压。它从高精度带隙模块中接收输入。在器件温度范围内，随温度变化的参考电压 （V_{CTAT}）将补偿在闪存存储器模块中所生成的所需泵电压，从而正常执行读写操作。

18.3.6 温度控制的电流发生器

该模块为 IMO 生成随温度变化的参考电流，以便在器件的工作温度范围内保持它的时钟频率差值为 ±2%。

表 18-1. 参考电压

参考电压	缓冲速度	说明
1.2 V	快速	LVD 模块的参考电压
1.2 V	慢速	CapSense 模块的参考电压
1.024 V	快速	BOD 模块的参考电压
1.024 V	快速	闪存模块的偏置参考电压用于读取闪存内的数据
1.024 V	慢速	SAR ADC 模块的参考电压
0.8 V	快速	比较器阈值，适用于 IMO 中的张弛振荡器
0.8 V	快速	LDO 模块中 VCCD 和 VCCA 电压调节器的参考电压
VCTAT	快速	闪存正电压 （VPOS）泵的参考电压，该参考电压会随温度不同而变化

表 18-2. 参考电流

参考电流	缓冲速度	说明
2.4 µA	快速模式	LCD 驱动、 BOD 以及闪存模块的参考电流，并为快速电压缓冲区提供偏置电压
2.4 µA	慢速	模拟模块 （SAR、CapSense、IDAC、LPCOMP 和 CTBm）的参考电流，并为慢速电压缓冲区提供偏置电压
9.6 µA	快速	IMO 模块的参考电流，支持基于温度的可编程补偿

18.4 配置

在上电过程中，高精度参考模块被初始化为保存在非易失性锁存器 （NVL）和 SFLASH 中的默认调整设置。这些设置在制造过程中已经过编程，并不需要进行字段调整。

19. SAR ADC

PSoC® 4 具有一个逐次逼近寄存器模数转换器（SAR ADC）。SAR ADC 的设计适用于各种需要中等分辨率以及高数据速率的应用。它包含以下模块 （请参见图 19-1）：

- SARMUX
- SAR ADC 内核
- SARREF
- SARSEQ

SAR ADC 内核是一个快速 12 位 ADC，在 PSoC 42xx-BL 系列和 PSoC 41xx-BL 系列中它的采样率分别为 1 Msps 和 806 Msps。SAR ADC 模块的前面是 SARMUX，它可将外部引脚和内部信号（如 AMUXBUS-A/-B、CTBm、温度传感器输出）路由到 SAR ADC 的八个内部通道。SARREF 用于选择多个参考电压。定序器控制器 SARSEQ 用于控制 SARMUX 和 SAR ADC，从而自动扫描所有被使能的通道 （无需 CPU 的干预）并进行预处理 （如求输出数据的平均值）。

第九个通道为插入通道，用于对引脚和信号进行不频繁和偶然的采样 （如内部温度传感器）。

每个通道的结果都是双缓冲的，并且会配置一个完整的扫描操作在扫描结束时生成中断。此外，数据还可传输到可编程数字模块 （UDB）中以进一步处理而无需 CPU 的干预。也可以配置定序器来标志溢流、冲突和饱和等可触发中断的错误。

为了提供更大的灵活性，还可以通过使用 UDB 来控制几乎所有模拟开关，其中包括 SARMUX 中的开关。这样能够通过 UDB 实现另一个定序器。

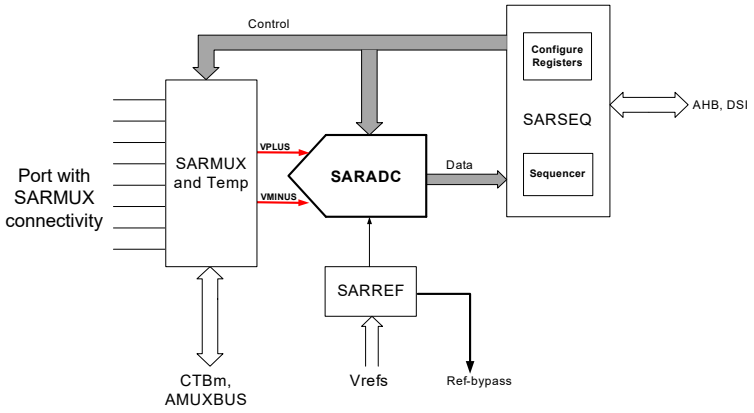
19.1 特性

- 可以在整个器件的供电范围内进行操作
- 最大采样率达 1 Msps
- 具有 8 个可独立配置通道及一个插入通道
- 每个通道具有以下特性：
  - 外部引脚 （8 个单端通道和四个差分通道）或内部信号 （AMUXBUS/CTBm/ 温度传感器）的输入
  - 可编程采集时间
  - 可选的 8 位、 10 位和 12 位分辨率
  - 单端或差分测量
  - 求平均
  - 结果数据是双缓冲的
  - 可对结果数据进行左对齐或右对齐
- 由定时器、引脚或 UDB 触发扫描
  - 单触发周期性或连续模式
- 支持硬件求平均
  - 第一阶累加
  - 对从 2 至 256 样本 （2 的幂）进行求平均
- 结果表示为 16 位有符号的扩展值
- 可选的参考电压

  - 内部参考电压 V_{DDA} 和 $V_{DDA}/2$
  - 使用缓冲区的内部 1.024 V 参考电压
  - 外部参考电压
- 中断生成
  - 完成了扫描转换操作
  - 每个通道的饱和检测和超出范围 （可配置）检测
  - 扫描结果溢出
  - 冲突检测
- 可配置插入通道
  - 可以在两个扫描序列之间交错进行 （紧接）
  - 可选的采样时间、分辨率、单端或差分、平均值
- 用于处理可编程数字模块中的数据以减轻 CPU 负载的选项
- 用于控制可编程数字模块开关的选项
- 用于控制 SAR ADC 和可编程数字模块开关的选项
  - 实现一个备用 SAR 定序器
  - 能够实现 1 Msps 的采样率
- 低功耗模式
  - ADC 内核和参考电压具有专用的低功耗模式

19.2 框图

图 19-1. 框图



19.3 工作原理

本节将介绍了以下内容：

- SAR ADC 内核、SARMUX、SARREF 和 SARSEQ 等模块的简介
- SAR ADC 系统资源：中断、低功耗模式以及 SAR ADC 状态
- 系统操作模式
  - 寄存器模式
  - DSI 模式
- 配置示例

19.3.1 SAR ADC 内核

PSoC 4 SAR ADC 内核是一个 12 位 SAR ADC。该 ADC 的最大采样率为 1 Msps。 SAR ADC 内核具有以下特性：

- 基于全差分架构模式，另外也支持单端模式
- 12 位分辨率和可选的备用分辨率 （8 位或 10 位）
- 可编程采集时间
- 可编程功耗模式 （100%、 50%、 25%）
- 支持单一和连续转换模式

19.3.1.1 单端模式和差分模式

PSoC 4 SAR ADC 可在单端模式和差分模式下运行。它的设计基于一个全差分架构，并得到优化，从而在差分操作模式下可提供 12 位的精度。它为处于 $-V_{REF}$ 到 $+V_{REF}$ 范围内的差分输入提供了全范围输出 （0 至 4095）。通过确定负输入，可在单端模式下配置 SAR ADC。可以使用通道配置寄存器（SAR_CHANx_CONFIG）配置为差分模式或单端模式。

负输入的单端模式选项包括：V_{SSA}、V_{REF} 或已连接至 SARMUX 的八个引脚中任意引脚的外部输入。更多有关引脚的信息，请参见该器件的数据手册。该模式是由全局配置寄存器 SAR_CTRL 配置的。当 Vminus 连接到这些 SARMUX 引脚时，该单端模式便相当于差分模式。然而，如果每个差分对中的奇数引脚连接到通用的备用接地引脚，这些转换操作便是 11 位的，因为测量的信号值 （SARMUX.vplus）不能低于接地值。

为了实现分辨率为 12 位的单端转换，需要将 V_{REF} 连接至 SAR ADC 的负输入；这样输入的电压范围为 0 到 $2 \times V_{ref}$。

请注意，仅在单端模式下才能使用温度传感器。它将 SAR_CTRL [11:9] 覆盖为 0。差分转换不适用于温度传感器，因此它的结果是 “ 未定义 ” 的。

19.3.1.2 输入电压范围

所有输入都要处于从 V_{SSA} 至 V_{DDA} 的范围内。输入电压范围也受 V_{REF} 的限制。如果负输入上的电压为 Vn，并且 ADC 参考电压为 V_{REF}，那么正输入的电压范围将为 $Vn \pm V_{REF}$。该标准适用于单端模式和差分模式。在单端模式下， Vn 连接着 V_{SSA}、 V_{REF} 或某个外部输入。

请注意：$Vn \pm V_{REF}$ 需要处于从 V_{SSA} 到 V_{DDA} 的范围内。例如，若负输入连接到V_{SSA}，则正输入的电压范围为0至V_{REF}，并非 $-V_{REF}$ 至 V_{REF}。这是因为该信号不能低于 V_{SSA}。由于正输入信号的摆幅不能低于 V_{SS}（这样只会生成一个 11 位的结果），因此只有 ADC 范围的一半是可用的。

19.3.1.3 结果数据格式

可从以下两个方面对结果数据格式进行配置：

- 有符号 / 无符号
- 左 / 右对齐

当结果为有符号时，转换操作的最高有效位将以符号扩展到 16 位，且扩展结果为 MSB。对于一个无符号的转换，使用零将结果扩展为 16 位。通过 SAR_SAMPLE_CTRL [3:2]，可以将结果数据格式分别配置为差分转换和单端转换。

采样值在结果寄存器的 16 位中可以是右对齐或左对齐。在默认情况下，data[11:0] 中的数据是右对齐的，并且被扩展为 16 位（若需要）。分辨率低与左对齐的结合会使较低有效位变为 0。

针对 12、 10、 8 位转换将有符号和无符号、左对齐和右对齐结合起来，结果数据格式可如下显示：

表 19-1. 结果数据格式

对齐	有符号 / 无符号	分辨率	结果寄存器															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
右对齐	无符号	12	–	–	–	–	11	10	9	8	7	6	5	4	3	2	1	0
		10	–	–	–	–	–	–	9	8	7	6	5	4	3	2	1	0
		8	–	–	–	–	–	–	–	–	7	6	5	4	3	2	1	0

表 19-1. 结果数据格式

对齐	有符号 / 无符号	分辨率	结果寄存器															
			15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
右对齐	有符号	12	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0
		10	9	9	9	9	9	9	9	8	7	6	5	4	3	2	1	0
		8	7	7	7	7	7	7	7	7	7	6	5	4	3	2	1	0
左对齐	–	12	11	10	9	8	7	6	5	4	3	2	1	0	–	–	–	–
		10	9	8	7	6	5	4	3	2	1	0	–	–	–	–	–	–
		8	7	6	5	4	3	2	1	0	–	–	–	–	–	–	–	–

19.3.1.4 负输入选择

负输入连接选择会影响电压范围、信噪比以及有效分辨率 （请参考表 19-2）。在单端模式下， SAR ADC 的负输入可连接至 V_{SSA}、 V_{REF}，或连接至与 SARMUX 互连的八个引脚中任意一个引脚。

表 19-2. 负向输入选择的比较

单端 / 差分	有符号 / 无符号	SARMUX Vminus	SARMUX Vplus 范围	结果寄存器	最大信噪比
单端	不适用[a]	V_{SSA}	$+V_{REF}$ V_{SSA} = 0	0x7FF 0x000	更好
单端	无符号	V_{REF}	+2 × V_{REF} V_{REF} V_{SSA} = 0	0xFFF 0x800 0	良好
单端	有符号	V_{REF}	+2 × V_{REF} V_{REF} V_{SSA} = 0	0x7FF 0x000 0x800	良好
单端	无符号	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0xFFF 0x800 0	最佳
单端	有符号	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0x7FF 0x000 0x800	最佳
差分	无符号	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0xFFF 0x800 0	最佳
差分	有符号	Vx	Vx + V_{REF} Vx Vx – V_{REF}	0x7FF 0x000 0x800	最佳

a. 在 Vminus 连接到 V_{SSA} 的单端模式下，因为在所有 PSoC 4 引脚上电压的摆幅不能低于 V_{SSA}，转换的分辨率为 11 位。因此，全局配置位 SINGLE_ENDED_SIGNED （SAR_SAMPLE_CTRL[2]）将被忽略，并且结果始终为 （0x000-0x7FF）。

为了能够实现 12 位分辨率的单端转换，需要将 V_{REF} 连接至 SAR ADC 的负输入端，这样输入的电压范围便为 0 到 2 × V_{REF}。

请注意，Vminus 连接到与 SARMUX 相连的引脚时，单端转换操作便为差分模式。然而，如果每个差分对中的奇数引脚连接到通用的备用接地引脚，这些转换操作便是 11 位的，因为测量的信号值 （SARMUX.vplus）不能低于接地值。

19.3.1.5 分辨率

PSoC 4 支持 12 位分辨率 （默认）以及一个可选的备用分辨率：每个通道可选 8 位或 10 位。分辨率影响到转换时间：

转换时间 （sar_clk） = 分辨率 （位） + 2

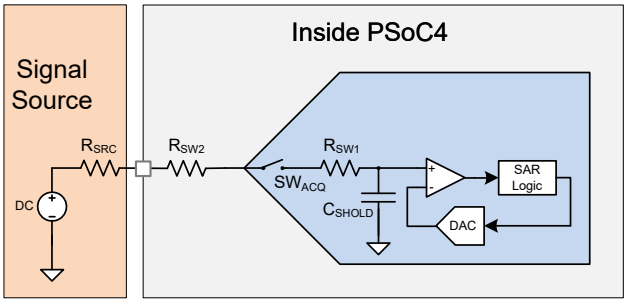
采集和转换的总时间 （sar_clk） = 采集时间 + 分辨率 （位） + 2

12 位转换以及采集时间为 4 时，所需的 sar_clk 周期数为 18。例如，如果 sar_clk 为 18 MHz，则需要使用 18 sar_clk 进行转换，此时转换速率将为 1 Msps。分辨率越低，转换速率越高。

19.3.1.6 采集时间

采集时间是指 SAR ADC 中采样以及保持（S/H）电路用于达到稳定状态的时间。采集时间后，输入信号源会断开与 SAR ADC 内核的连接，并使用 S/H 电路的输出进行转换。每个通道都可以选择四个采集时间选项中的一个，即在全局配置寄存器 SAR_SAMPLE_TIME01 和 SAR_SAMPLE_TIME23 中所定义的从 4 到 1023 的 SAR 时钟周期。

图 19-2. 采集时间



采集时间要足够长，以通过路由路径的电阻给 ADC 的内部保持电容充电，如图 19-2 所示。采集时间的推荐值为：

$t_{ACQ} \geq 9 \times (R_{SRC} + R_{SW2} + R_{SW1}) \times C_{SHOLD}$

其中：

C_{SHOLD} ~= 10 pF

$R_{SW2} + R_{SW1}$ = ~ 500 至 1000 Ω，具体情况取决于路由路径 （具体请参见第 256 页上的模拟路由 ）。

R_{SRC} = 信号源的串联电阻

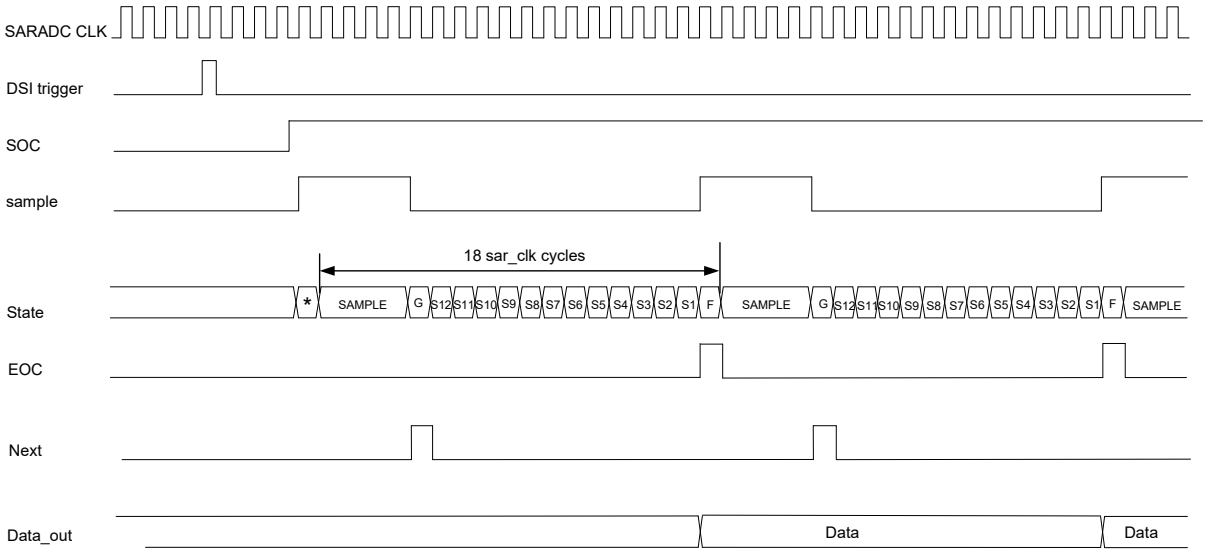
19.3.1.7 SAR ADC 时钟

SAR ADC 时钟频率必须介于 1 MHz 和 18 MHz 之间（对于 PSoC 42xx-BL ）或 1 MHz 和 14.5 MHz 之间（对于 PSoC 41xx-BL）。HFCLK 时钟源经过时钟分频器可得到该频率。请注意， SAR ADC 不支持小数分频。要想获取 1 Msps 的采样率，需要使用一个 18 MHz SAR ADC 时钟。此时，必须将系统时钟 （HFCLK）设置为 36 MHz 而不是 48 MHz。对于 PSoC 41xx-BL 器件，需要将 IMO 设为 29 MHz ，以获取 806 ksps 的采样率。当最小采集时间为 4 个时钟周期时 （频率为 18 MHz），需要 18 个时钟周期来完成 12 位 ADC 转换操作。 10 位和 8 位转换则分别需要 16 和 14 个时钟周期。请注意，频率为 18 MHz 时，最小采集时间为 4 个时钟周期，该数据是根据 SAR 模块 （图 19-2 中的 R_{SW1} 和 C_{SHOLD}）所支持的最小采集时间 （194 ns）得到的。

19.3.1.8 SAR ADC 时序

如图 19-3 所示，在生成 ‘开始转换’ （SOC）前，会出现一个 sar_clk 延迟。12 位分辨率的转换需要 14 个时钟周期来完成（每一位需要一个 sar_clk，再加上处于 G 和 F 状态的两个超额 sar_clk）。当采集时间默认为 4 个 sar_clk 周期时， ADC 的采集及转换的总时间需要 18 个 sar_clk 时钟周期。采样（采集）后，将输出下一个脉冲（or dsi_sample_done）。可以将 SARMUX 连接到其它引脚和信号；通过定序器可以自动实现该操作 （请参考第 264 页上的 SARSEQ，了解详细信息）。

图 19-3. SAR ADC 时序



19.3.2 SARMUX

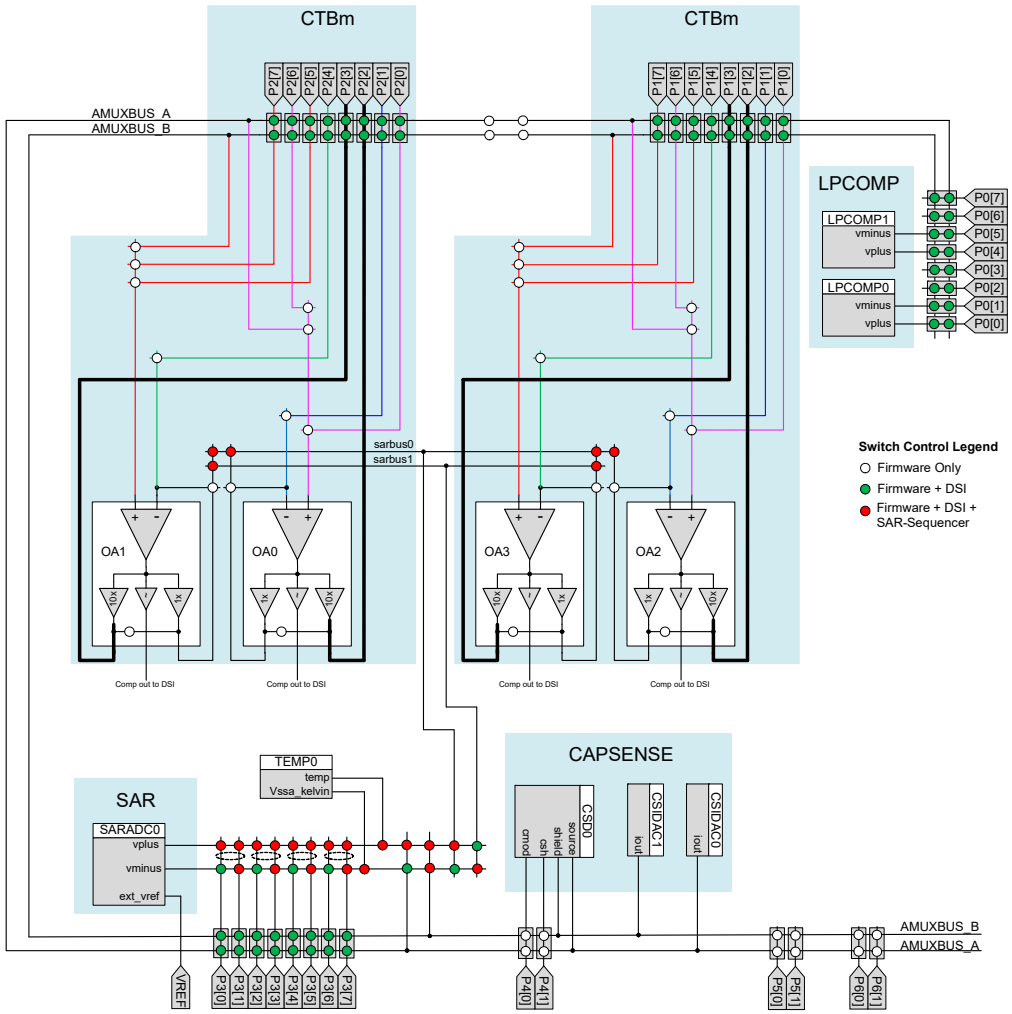
SARMUX 是一个专用的模拟可编程复用器。 SARMUX 的主要特性如下：

- 导通电阻：600 Ω （最大值）
- 内部温度传感器
- 由定序器控制器模块 （SARSEQ）或 UDB 控制。
- 内部电荷泵：
  - 如果 V_{DDA} < 4.0 V，应打开电荷泵，以降低开关电阻
  - 如果 V_{DDA} ≥ 4.0 V，将关闭电荷泵，并输出 V_{DDA} 电压。
- 多个输入：
  - 从各引脚的模拟信号 （端口 2）
  - 温度传感器输出
  - 通过 sarbus0/1 的 CTBm 输出 （不足以按 1 Msps 的采样频率进行采样）
  - AMUXBUS_A/_B （不足以按 1 Msps 的采样率进行采样）

19.3.2.1 模拟路由

SARMUX 有多个可由 SARSEQ 模块 （定序器控制器）或 DSI 控制的开关。定序器和 DSI 都是硬件控制方式，它们可被 SAR_MUX_SWITCH_HW_CTRL 寄存器中的硬件控制位屏蔽。不同的控制方法会有不同的开关控制能力。请参见图 19-4。

图 19-4. SARMUX 开关与控制能力



定序器控制：各个开关由 SARSEQ 模块中的定序器控制。配置好每个通道的模拟路由后，能够以循环方式自动扫描多个通道，无需 CPU 的干预。并不是所有开关都可由定序器控制；请参考图 19-4。相应的寄存器包括：SAR_CHANx_CONFIG、SAR_MUX_SWITCH0、SAR_CTRL 以及 SAR_MUX_SWITCH_HW_CTRL。有关在寄存器模式中可用的具体配置，请参见第 275 页上的固件模拟路由。

固件控制：可编程寄存器直接定义 VPLUS/VMINUS 接口。它可控制 SARMUX 中的所有开关；请参考图 19-4。例如，在固件控制模式中，可以对任何两个引脚或信号进行差分测量，而不仅仅是测量两个相邻的引脚（如定序器控制方法中介绍的内容）。然而，采集多个通道时则需要 CPU 的干预。相应的寄存器包括：SAR_MUX_SWITCH0、SAR_MUX_SWITCH_HW_CTRL 以及 SAR_CTRL。有关在寄存器模式中可用的具体配置，请参见第 275 页上的固件模拟路由。

DSI 控制：各个开关由 UDB 中的 DSI 信号控制。在自定义逻辑设计中，它可作为第二个定序器工作。DSI 能控制大多数开关。因此，它可对任何两个引脚和信号间进行差分测量，并进行固件控制。可以在 DSI 模式中查看可用的具体配置。请参见第 271 页上的 SARMUX 模拟路由。

19.3.2.2 模拟互连

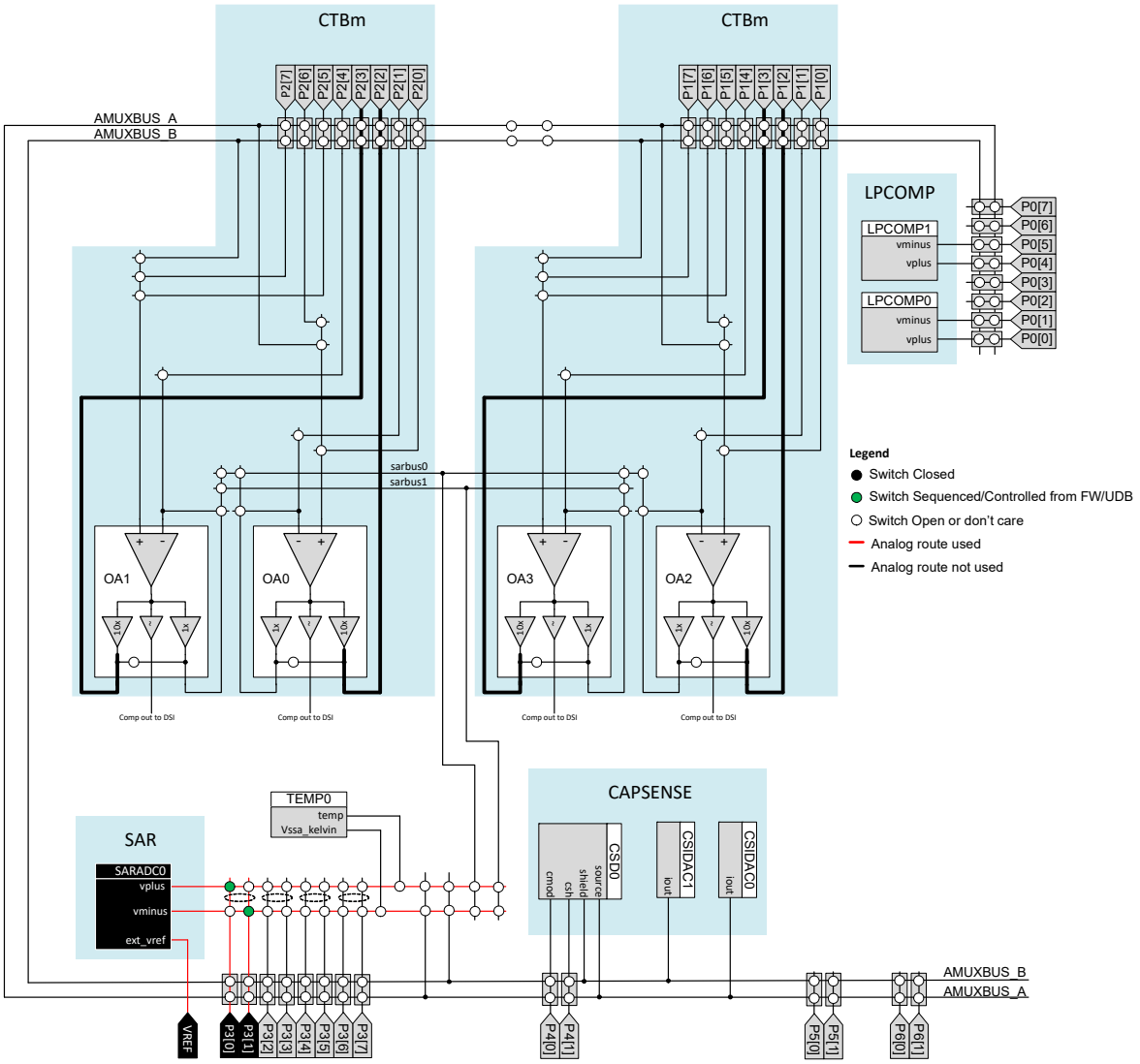
PSoC 4 模拟互连的灵活性很强。通过 SARMUX，可将 SAR ADC 连接至多个输入端，包括外部引脚以及内部信号。例如，它可以连接到一个相邻模块，如 CTBm。它也可以通过 AMUXBUS_A/_B 连接到其它引脚（端口 2 除外），但扫描性能却被降低（需要较多的寄生耦合，较长的 RC 时间来进行扫描）。

下面将介绍几种情况，加深用户对模拟互联的了解。

来自外部引脚的输入

图 19-5 显示的是如何通过各开关，将支持 SARMUX 的两个 GPIO 作为差分对 （Vpuls/Vminus）连接至 SAR ADC。这两个开关可由定序器、固件或 DSI 控制。这些引脚被安排在各相邻对中；例如，在 SARMUX 端口 P3[0] 和 P3[1]、P3[2] 和 P3[3]，等等。如果您需要使用未被配对成差分对的各个引脚，（如 P3[1] 和 P3[2]），那么定序器便不适用；请使用固件或 DSI。

图 19-5. 来自外部引脚的输入

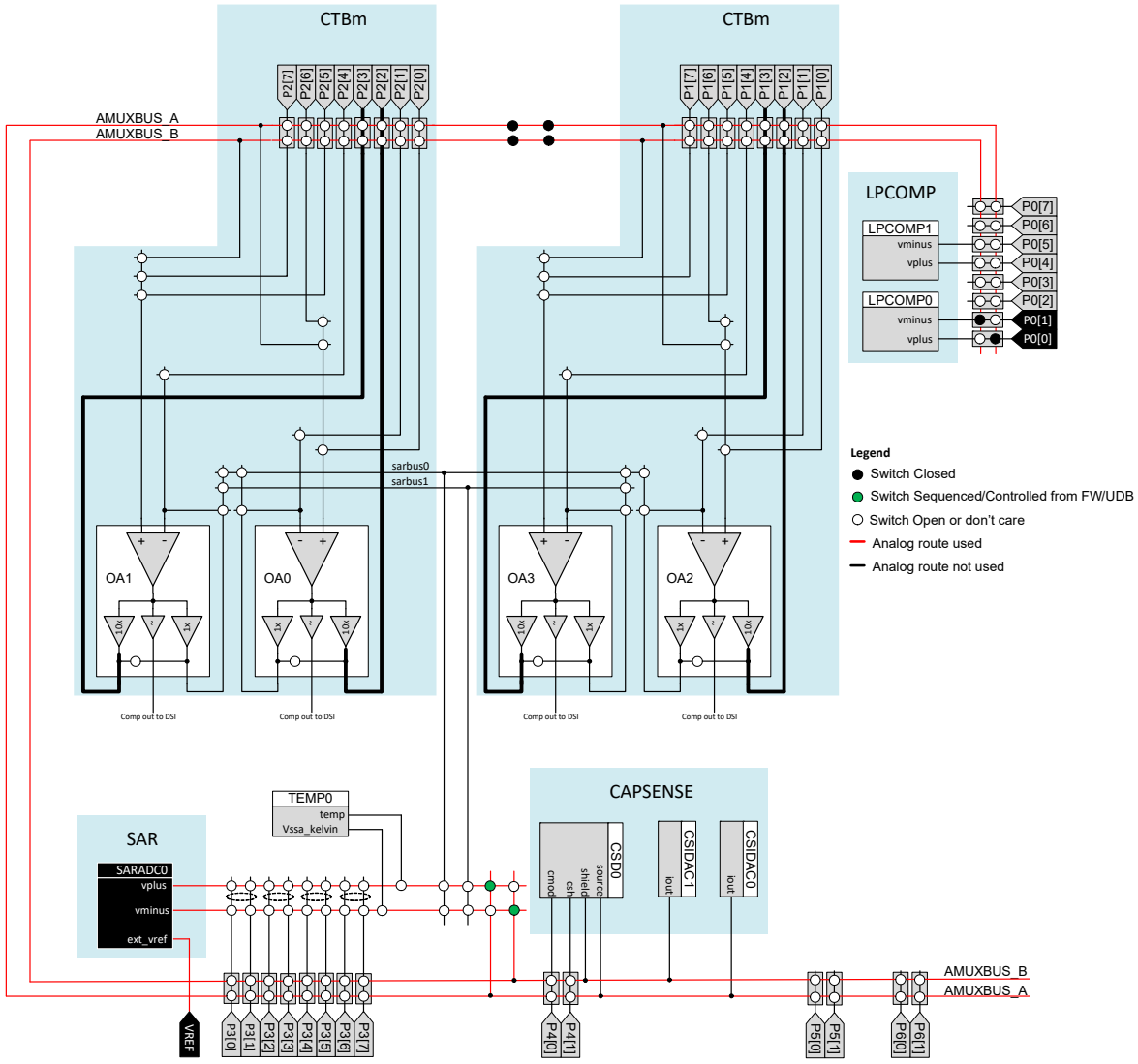


来自模拟总线的输入（AMUXBUS_A/B）

图 19-6 显示的是不支持 SARMUX 连接的两个引脚如何作为差分对连接至 ADC。需要使用额外的开关将这些差分对连接到 AMUXBUS_A 和 AMUX-BUS_B 两个引脚上，然后将 AMUXBUS_A 和 AMUXBUS_B 连接到 ADC 上。

额外的开关会降低扫描性能（由于需要较多的寄生耦合，较长的 RC 时间来进行扫描）— 这样便不能按 1 Msps 的采样率进行采样。不建议使用外部信号。请使用专用的 SARMUX 端口 （若可以）。

图 19-6. 来自模拟总线的输入



通过 sarbus 与 CTBm 输出连接的输入

通过 sarbus 0/1，可以将 SAR ADC 连接至 CTBm 输出。图 19-7 显示了如何将一个运算放大器 （被配置为一个跟随器）输出连接到单端 SAR ADC。负端连接至 V_{REF}。图 19-8 显示的是如何将两个运算放大器输出作为一个差分对连接至 SAR ADC。必须先将运算放大器输出连接至 sarbus 0/1，然后再将 SAR ADC 输入与 sarbus 0/1 相连接。因为还有额外的开关，所以不足以按 1Msps 的采样率进行采样。然而，该片上运算放大器在很多应用中非常有用。

图 19-7. 通过 sarbus，将 CTBm 输出作为输入

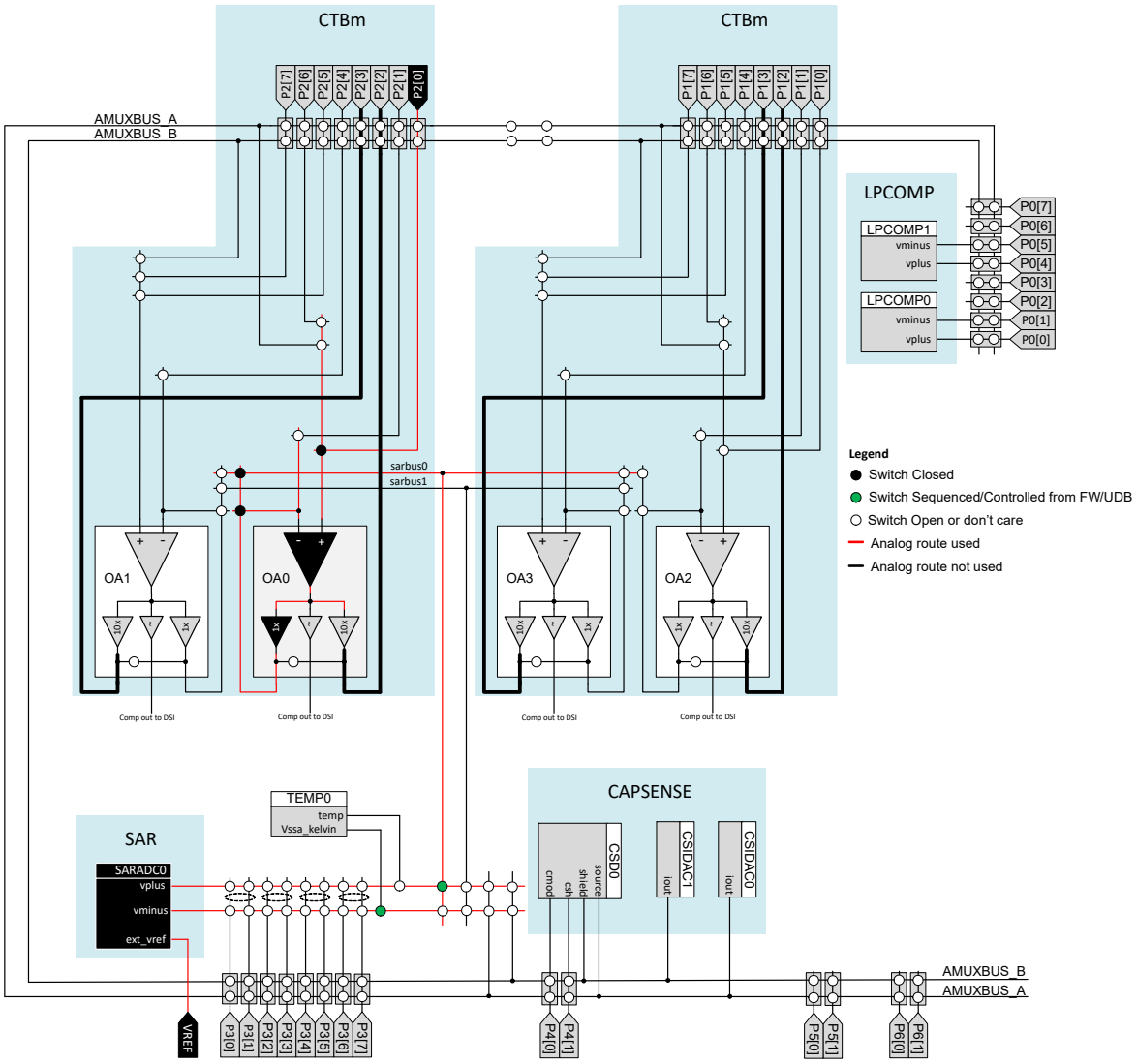
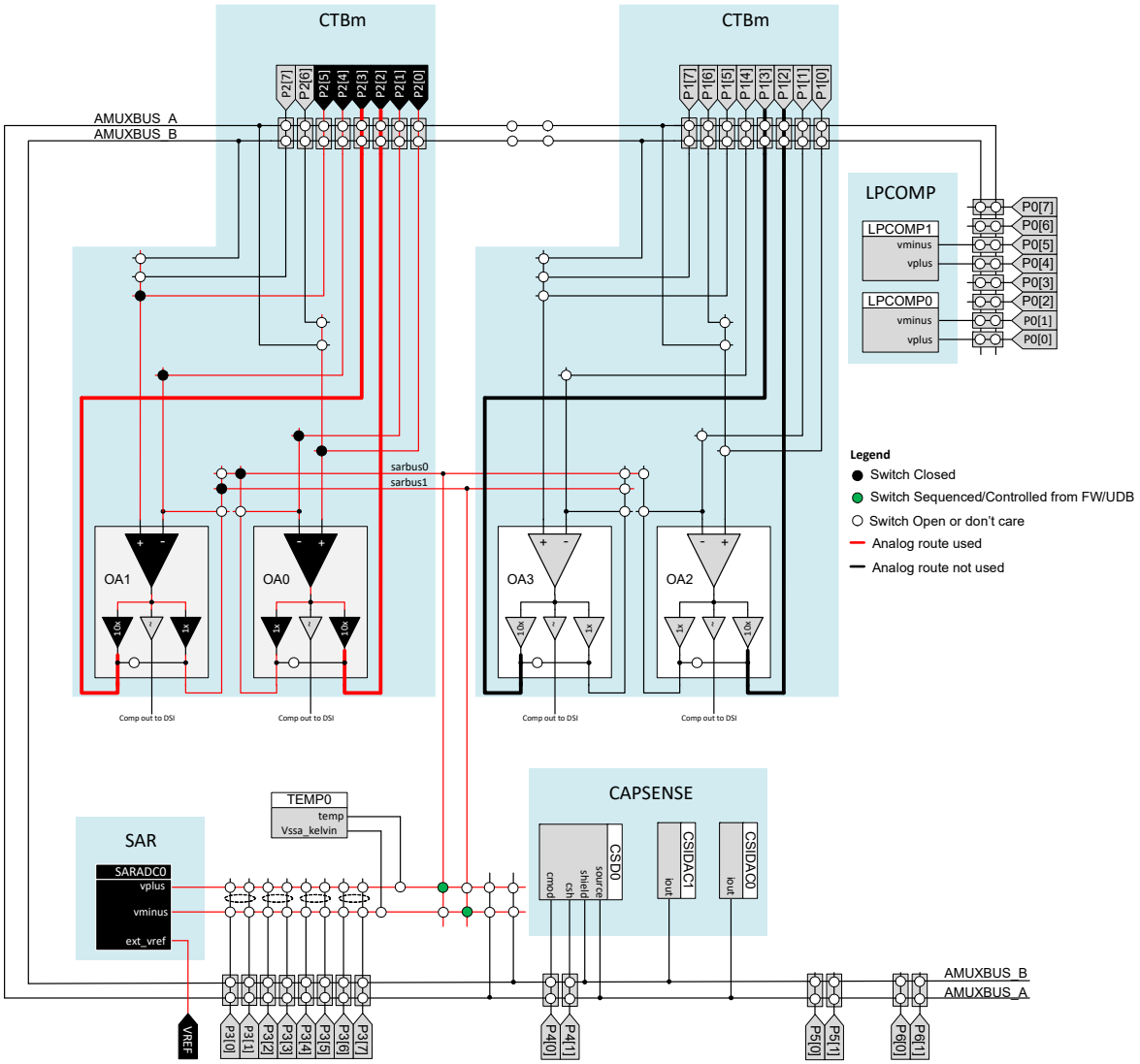


图 19-8. 通过 sarbus0 和 sarbus1，将 CTBm 输出作为输入

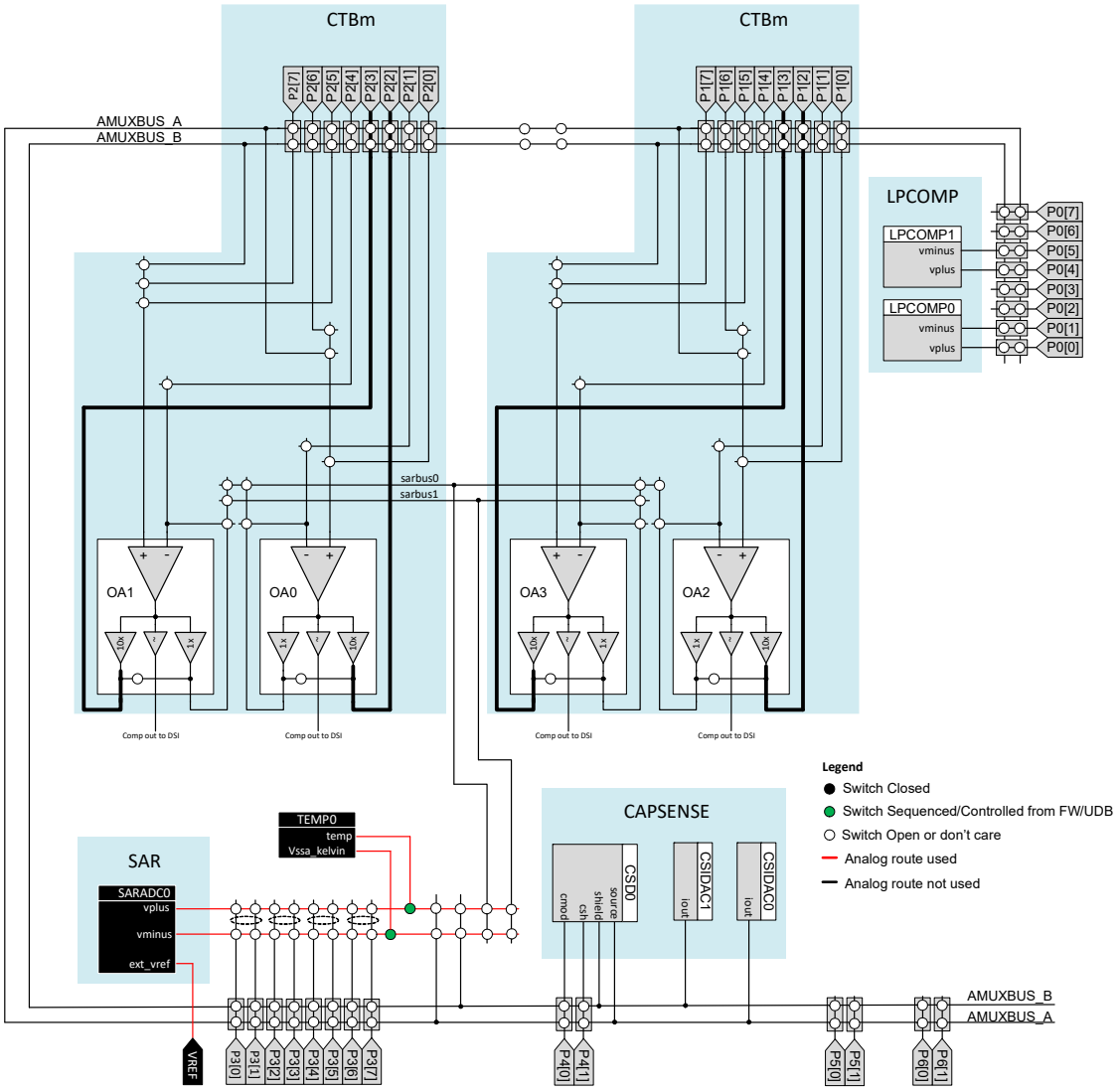


来自温度传感器的输入

一个片上温度传感器可适用于温度感应以及基于温度的校准。请注意，对于温度传感器，差分转换不可用（转换结果未定义），因此总是在单端模式下使用它。

如图 19-9 所示，通过开关可将温度传感器路由到 SAR ADC 的正输入上，该操作可由定序器、固件或 DSI 控制。置位 MUX_FW_TEMP_VPLUS 位（即 SAR_MUX_SWITCH0[17]）可以使能温度传感器，并将其输出连接到 SAR ADC 的 VPLUS；但清除该位时，可通过切断其偏置电流来禁用温度传感器。

图 19-9. 来自温度传感器的输入

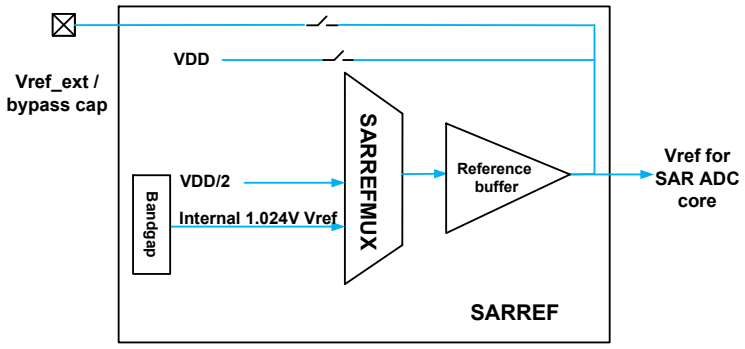


19.3.3 SARREF

SARREF 的主要特性如下：

- 参考电压选项：V_{DDA}、$V_{DDA}/2$、1.024 V 带隙 （±1%）以及外部参考电压
- 参考缓冲区以及旁路电容，用于加强内部参考驱动能力

图 19-10. SARREF 框图



19.3.3.1 参考电压选项

SAR ADC 的参考电压选项包括 SARREF 中的参考复用器和开关。这样的选项允许将下面各个电压连接在一起：V_{DDA}、$V_{DDA}/2$、来自带隙的 1.024 V 内部参考电压，以及与一个外部 Vref/SAR 旁路引脚相连接的外部 V_{REF} （有关详细信息，请参见器件数据手册）。 SARREF 内的参考复用器的控制是在全局配置寄存器 SAR_CTRL [6:4] 内进行的。

19.3.3.2 旁路电容

来自带隙的 1.024 V 或 $V_{DDA}/2$ 等内部参考电压都是通过参考缓冲区得到缓冲的。该参考可能会路由至外部 Vref/SAR 旁路引脚，该引脚上的外部电容可用于过滤掉参考信号中存在的内部噪声。如果不使用外部参考旁路电容，则 SAR ADC 采样率不可超过 100 ksps （在 12 位分辨率时）。例如，当不使用旁路电容并且内部 V_{REF} 为 1.024 V 时， SAR ADC 时钟的最大频率为 1.6 MHz。使用外部参考电压时，推荐使用一个外部电容。通过设置 SAR_CTRL [7]，旁路电容得到使能。表 19-3 列出了不同的参考模式以及在 12 位连续模式操作条件下的最大频率 / 采样率。

表 19-3. 参考模式

参考模式	参考 SAR_CTRL [6:4]	旁路电容 SAR_CTRL[7]	缓冲区	最大频率	最大采样率 （在 12 位分辨率时）
不使用旁路电容时的1.024 V内部参考电压V_{REF}	4	0	有	1.6 MHz	100 ksps
使用旁路电容时的 1.024 V 内部参考电压 V_{REF}	4	1	有	18 MHz	1 Msps
外部参考电压 V_{REF} （低阻抗路径）	5	X	无	18 MHz	1 Msps
不使用旁路电容时的 $V_{DDA}/2$	6	0	有	1.6 MHz	100 ksps
使用旁路电容时的 $V_{DDA}/2$	6	1	有	18 MHz	1 Msps
V_{DDA}	7	X	无	9 MHz	500 ksps

1.024 V 内部 V_{REF} 的启动时间会因旁路电容大小不同而不同。表 19-4 列出了两个最常见的旁路电容值以及对应的启动时间。如果在各扫描操作间或退出睡眠 / 深度睡眠状态后进行扫描时更改了参考选项，需要确保 SAR ADC 开始采样时，1.024 V 的内部 V_{REF} 处于稳定状态。在最坏的情况下，稳定时间 （V_{REF} 完全放电时）与启动时间一样长。

表 19-4. 旁路电容值

内部 V_{REF} 的启动时间	最大时间规范
使用 1 µF 大小的外部电容时参考电压的启动时间	2 ms
使用 100 nF 大小的外部电容时参考电压的启动时间	200 µs

19.3.3.3 输入范围和参考

所有输入的电压要处于 V_{SSA} 和 V_{DDA} 之间的范围内。该 ADC 输入电压的范围受 V_{REF} 选择的限制。如果负输入上的电压为 Vn，并且 ADC 参考电压为 V_{REF}，那么正输入的电压范围将为 Vn ± V_{REF}。只要正负输入端间的电压都处于 V_{SSA} 到 V_{DDA} 的范围内，该标准便适用于单端模式和差分模式。

19.3.4 SARSEQ

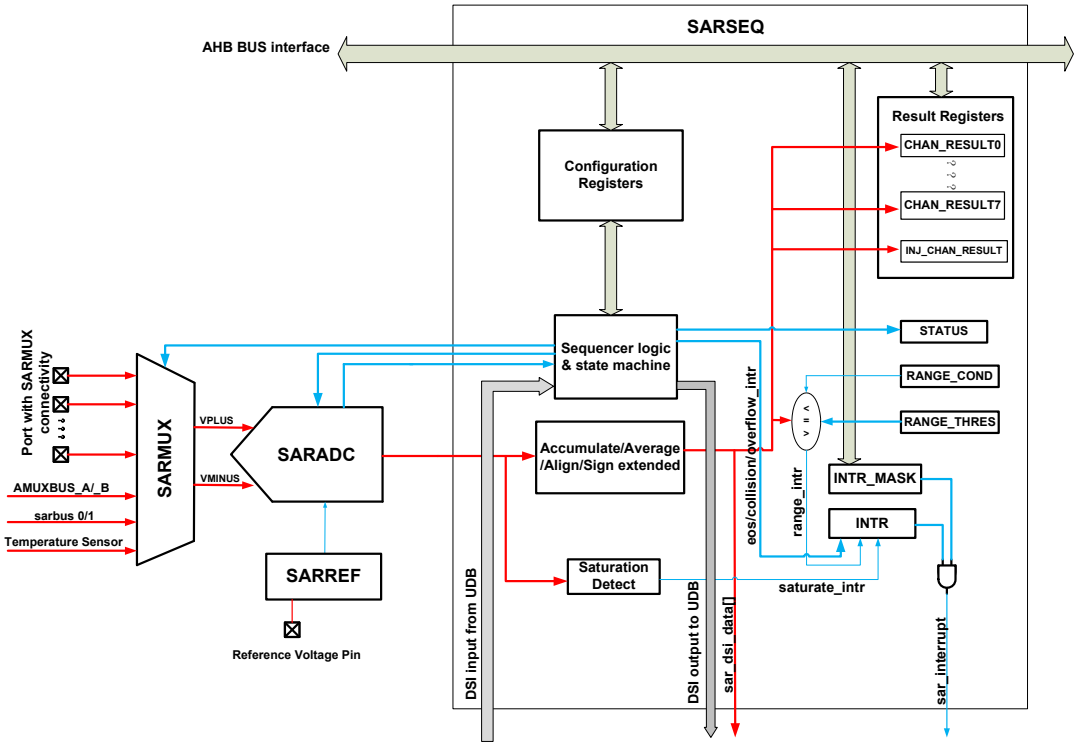
SARSEQ 是一个专用的定序器控制器，它会自动控制输入复用器对通道逐个进行扫描，并将其结果放置在寄存器的一个阵列中，这样每个寄存器便存储每一个通道的扫描结果。

- 自动控制 SARMUX 模拟路由而无需 CPU 的干预
- 控制 SAR ADC 内核 （如分辨率、采集时间和参考电压）
- 接收 SAR ADC 输出的数据，并对该数据进行预处理（求平均值、范围检测）
- 结果被双缓冲，因此处理下一个扫描时，CPU 能够安全读取最终扫描操作的结果。

SARSEQ 的特性如下：

- 可单独使能 8 个通道，使之自动进行扫描而无需 CPU 的干预
- 第九个通道（插入通道）用于在一个自动扫描中插入偶发信号
- 每个通道具有以下特性：
  - 单端模式或差分模式
  - 外部引脚（8 个单端通道和四个差分通道）或内部信号（AMUXBUS/CTBm/ 温度传感器）的输入
  - 最多支持四个可编程采集时间选项
  - 分辨率默认为 12 位，另外还有可选的备用分辨率 （8 位或 10 位）
  - 对结果进行求平均值
- 扫描触发
  - 单触发周期模式或连续模式
  - 由 GPIO 引脚上的任何数字信号或输入触发
  - 由固定功能模块的内部 UDB 触发
  - 由软件触发
- 支持硬件求平均
  - 第一阶累加
  - 求 2 到 256 个样本 （2 的幂）的平均值
  - 结果使用 16 位表达式
- 输出数据的双缓冲
  - 对结果进行左对齐或右对齐
  - 结果被保存在工作寄存器和结果寄存器内
- 中断生成
  - 完成了扫描转换操作
  - 在所有控制模式下的通道饱和检测
  - 每个通道的超范围 （可配置）检测
  - 扫描结果溢出
  - 冲突检测
- 可配置插入通道
  - 可以在两个扫描序列之间交错进行 （紧接）
  - 可选的采样时间、分辨率、单端或差分模式、求平均值

图 19-11. SARSEQ 框图



19.3.4.1 *求平均*

SARSEQ 模块具有一个 20 位的累加器和移位寄存器，用于进行求平均值运算。有符号扩展后进行求平均值。全局配置寄存器 SAR_SAMPLE_CTRL 指定求平均操作的详细信息。

在寄存器控制模式下，通道配置寄存器 SAR_CHAN_CONFIG 具有一个用于使能求平均功能的使能位（AVG_EN）。在 DSI 控制模式下， dsi_cfg_average 信号会使能求平均操作。

在全局配置寄存器中， AVG_CNT （SAR_SAMPLE_CTRL [6:4]）将根据以下公式指定采样数 （N）：

N = 2^(AVG_CNT+1)， N 范围 = [2..256]

例如，如果 AVG_CNT （SAR_SMAPLE_CTRL [6:4]）= 3，则 N = 16。

AVG_SHIFT 位 （即 SAR_SAMPLE_CTRL[7]）用于移位结果，从而提取平均值；如果使能了求平均操作，应设置该位。

如果配置了某个通道用以进行求平均，那么每次扫描时，SARSEQ 将会使用指定通道的 N 个连续样本。由于转换结果是 12 位的，并且 N 的最大值是 256 （左移 8 位），因此 20 位的累加器不会发生溢出现象。

如果设置了 SAR_SAMPLE_CTRL 寄存器中的 AVG_SHIFT，那么 SAR 定序器将实现有符号扩展，然后进行累加。累加结果将右移 AVG_CNT + 1 位，以进行求平均运算。如果不置位该位，会强制该结果向右转移，以确保该位的宽度不超过 16 位。通过 （0， AVG_CNT - 3）的最大值实现右移 — 如果采样数超过 16 个 （AVG_CNT >3），则累加结果被右移 （AVG_CNT - 3）位；如果 AVG_CNT < 3，则该结果不被移位。请注意，在这种情况下，平均结果大于所需结果；推荐置位 AVG_SHIFT。该模式总是使用被选中的 ADC 分辨率（12、10 或 8 位）。

19.3.4.2 *范围检测*

通过 SARSEQ 支持的范围检测，可以在无 CPU 干预的情况下自动将结果值与两个可编程阈值进行比较。范围检测是由 SAR_RANGE_THRES 寄存器定义的。 RANGE_LOW 字段（即 SAR_RANGE_THRES [15:0]）的值定义了下限阈值，而 RANGE_HIGH 字段（即 SAR_RANGE_THRES [31:16]）则定义了它的上限阈值。

SAR_RANGE_COND 位用于定义触发某个通道的可屏蔽范围检测中断 （RANGE_INTR）的条件。可以选择以下条件：

0：结果 < RANGE_LOW （低于阈值）

1：RANGE_LOW ≤ 结果 < RANGE_HIGH （在范围内）

2：RANGE_HIGH ≤ 结果 （超出范围）

3：结果 < RANGE_LOW || RANGE_HIGH <= 结果 （在范围外）

更多有关信息，请参考第 267 页上的范围检测中断 。

19.3.4.3 双缓冲

通过使用双缓冲，固件可以在处理下一个扫描操作时读取已完成的扫描结果。将 SAR ADC 的结果写入到一组工作寄存器内，直到扫描结束为止。此时数据将被复制到第二组寄存器内，这样数据可被用户应用读取。这样会为固件提供足够的时间，以在当前扫描结束前能够读取前一个扫描结果。使用 16 个寄存器对插入通道除外的所有输入通道进行双缓冲。由于插入通道一般不是正常通道扫描的一部分，因此不需要对它进行双缓冲。

19.3.4.4 插入通道

通过设置 SAR_INJ_CHAN_CONFIG 寄存器，可以使用与正常通道转换相同的方法来配置插入通道的转换。该寄存器支持：

- 引脚或信号选择
- 单端或差分选择
- 选择 12 位分辨率或全局指定 SUB_RESOLUTION 分辨率
- 从 4 个全局指定采样时间选项中选择一个
- 平均值选择

它支持将相同的中断使用于正常通道（溢出中断除外）。

- 可屏蔽的转换结束中断 INJ_EOC_INTR
- 可屏蔽的范围检测中断 INJ_RANGE_INTR
- 可屏蔽的饱和检测中断 INJ_SATURATE_INTR
- 可屏蔽的冲突检测中断 INJ_COLLISION_INTR

SAR_INTR、SAR_INTR_MASK、SAR_INTR_MASKED 以及 SAR_INTR_SET 是相应的寄存器。

第 272 页上的全局 SARSEQ 配置 、第 272 页上的通道配置和第 266 页上的中断 中详细介绍了这些特性。

紧接

通过置位起始或使能位 INJ_START_EN（SAR_INJ_CHAN_CONFIG [31]），可以触发插入通道的转换。推荐通过设置 INJ_TAILGATING = 1（SAR_INJ_CHAN_CONFIG [30]）来选择紧接方式。可在正常通道正在进行的扫描操作结束时扫描插入通道，而不会发生任何冲突。然而，如果当前未进行任何扫描或者 SAR ADC 处于闲置状态，那么 INJ_START_EN 位会允许在正常通道下一次扫描结束时扫描插入通道。

插入通道的转换完成后，将设置 “ 转换结束 ” 中断 （INJ_EOC_INTR）并清除 INJ_START_EN 位。插入通道的转换数据被放置在 SAR_INJ_RESULT 寄存器中。与 SAR_CHAN_RESULT 相似，各寄存器包含 “valid” （= INJ_EOC_INTR）、范围检测、饱和检测中断等的镜像位以及冲突中断 （INJ_COLLISSION_INTR）的镜像位。

19.3.5 中断

可以通过以下不同事件来生成一个中断：

- 扫描结束 — 扫描完所有被使能的通道。
- 溢出 — 在读取先前结果前已更新了结果寄存器。
- 冲突 — SAR ADC 正在处理先前的触发时，收到新的触发。
- 插入转换结束 — 完成了插入通道的转换。
- 范围检测 — 通道结果满足阈值。
- 饱和检测 — 通道结果等于设置分辨率的最小或最大值。

本部分详细描述了每个中断。每个中断具有 SAR_INTR_MASK寄存器中的一个中断屏蔽位。将中断屏蔽置为低电平时，相应的中断源将被忽略。如果中断屏蔽位为高电平，并且相应的中断源处于挂起状态，将生成 SAR 中断。

执行中断时，中断服务子程序（ISR）在读取数据后，通过将 “1” 写入中断位来清除中断源。

SAR_INTR_MASKED 寄存器是中断源和中断屏蔽间的逻辑 “ 与 ” （AND）的结果。这样会为固件提供一个便利的方法来确定中断源。

要想进行验证和调试，需要通过一个设置位 （如 SAR_INTR_SET 寄存器中的 EOS_SET）来触发每个中断。这样会允许固件生成中断而不用发生实际事件。

19.3.5.1 “ 扫描结束 ” 中断 （EOS_INTR）

扫描完成后，将生成 “ 扫描结束 ” 中断（EOS_INTR）。从 RESULT （结果）寄存器采集数据后，固件会清除该中断。

此外，通过设置 SAR_SAMPLE_CTRL [31] 中的 EOS_DSI_OUT_EN 位，可以在 DSI 总线上将 EOS_INTR 中断发送出去。在 DSI 总线上，将在两个系统时钟周期内保持 EOS_INTR 信号。针对扫描进行的最后一个通道 （若选中），这些周期与 data_valid 信号一致。

通过将 SAR_INTR_MASK 寄存器中的 EOS_MASK 位设置为 0，可以屏蔽 EOS_INTR。 SAR_INTR_MASKED 寄存器的 EOS_MASKED 位是对中断标志和中断屏蔽进行逻辑 “ 与 ” （AND）运算得到的结果。将 ‘1’ 写入到 SAR_INTR_SET 寄存器中的 EOS_SET 位可以设置用于调试和验证的 EOS_INTR。

19.3.5.2 溢出中断

如果新的扫描操作已结束，并且硬件尝试将 EOS_INTR 和 EOS_INTR 设置为高电平 （固件不够快地清除它），那么硬件会生成一个溢出中断 （OVERFLOW_INTR）。这通常意味着当前的扫描完成前，固件不可读取先前的扫描结果。在这种情况下，旧数据将被覆盖。

通过将 SAR_INTR_MASK 寄存器中的 OVERFLOW_MASK 位置0，可以屏蔽OVERFLOW_INTR。SAR_INTR_MASKED 寄存器的OVERFLOW_MASKED位是中断标志和中断屏蔽经

过逻辑 “ 与 ”（AND）运算的结果，这样便于固件使用。将 ‘1’ 写入到SAR_INTR_SET寄存器中的OVERFLOW_SET 位可以设置用于调试和验证的 OVERFLOW_INTR。

19.3.5.3 冲突中断

SARSEQ 忙碌时，通过使用以前触发器触发扫描操作，可能会生成一个新的触发器。因此，当前扫描完成前，新触发器的扫描将一直被延迟。此时，要注意通知固件新采样无效。可通过冲突中断来实现。当接收到一个新的触发 （而不是连续触发）时，便会生成该中断。

有两种冲突中断：一种用于 DSI 触发器（DSI_COLLISION_INTR），另一种用于插入通道（INJ_COLLISION_INTR）。这些中断允许固件确定哪个触发器与正在执行的扫描相冲突。

在电平模式下使用 DSI 触发器时，不会设置 DSI_COLLISION_INTR。

通过将 SAR_INTR_MASK 寄存器中的相应位设置为 ‘0’，可以屏蔽三个冲突中断。SAR_INTR_MASKED寄存器的相应位是中断标志和中断屏蔽间的逻辑 “ 与 ”（AND）的结果。将数值 ‘1’ 写入到 SAR_INTR_SET 寄存器中的相应位，可以设置用于调试和验证的冲突中断。

19.3.5.4 “插入转换结束”中断（INJ_EOC_INTR）

完成插入通道的转换后，将生成 “ 插入转换结束 ” 中断（INJ_EOC_INTR）。从 INJ_RESULT 寄存器采集数据后，固件会清除该中断。

请注意，如果插入通道扫描紧接着一个扫描，将生成 EOS_INTR，同时也开始进行插入通道转换。该插入通道不被视为扫描的一部分。

通过将 SAR_INTR_MASK 寄存器中的 INJ_EOC_MASK 位设置为 ‘0’，可以屏蔽 INJ_EOC_INTR。 SAR_INTR_MASKED 寄存器的 INJ_EOC_MASKED 位是对中断标志和中断屏蔽进行逻辑 “ 与 ” （AND）运算得到的结果。将 ‘1’ 写入到 SAR_INTR_SET 寄存器中的 INJ_EOC_SET 位，可以设置用于调试和验证的 INJ_EOC_INTR。

19.3.5.5 范围检测中断

实现求平均值、调整以及符号扩展 （如适用）后即可设置范围检测中断标志。这意味着不用等待整个扫描过程完成，即可确定通道转换是否超出范围。该阈值的数据格式需要与结果数据格式相同。

通过将SAR_RANGE_INTR_MASK寄存器中的指定位设置为 ‘0’，可以屏蔽指定通道的范围检测中断。SAR_RANGE_INTR_MASKED 寄存器反映的是中断请求和屏蔽寄存器间的按位进行逻辑与运算（AND）后得到的结果。如果该值为非零值，则输入到 NVIC 的 SAR 中断信号为高电平。

SAR_RANGE_INTR_SET 可用于调试 / 验证。写入 ‘1’ 值，从而设置中断请求寄存器中的相应位；读取它时，该寄存器反映的是中断请求寄存器。

每个通道都有一个范围检测中断 （RANGE_INTR 和 INJ_RANGE_INTR）。

19.3.5.6 饱和检测中断

饱和检测适用于所有转换操作。该特性将检测采样值是否等于指定分辨率的最小值或最大值，并且为相应通道设置一个可屏蔽中断标志。这样会允许固件在 SAR ADC 饱和时采取行动（如清除结果）。转换结束后并且实现求平均前会立即测试采样值。因此当数据寄存器中的平均值不等于最小值或最大值时，将设置该中断。

当为通道选择 10 位或 8 位的分辨率时，将在 10 位或 8 位的数据上进行饱和检测。

扫描结束和实现求平均值前，应立即设置饱和中断标志以能够快速响应饱和。通过将 SAR_SATURATE_INTR_MASK 寄存器中的指定位设置为 ‘0’，可以屏蔽指定通道的饱和检测中断。SAR_SATURATE_INTR_MASKED 寄存器反映的是中断请求和屏蔽寄存器之间按位进行逻辑与运算 （AND）后的结果。如果该值为非零值，则输入到 NVIC 的 SAR 中断信号为高电平。

SAR_SARTURATE_INTR_SET 可用于调试 / 验证操作。写入 ‘1’ 值，从而设置中断请求寄存器中的相应位；读取它时，该寄存器反映的是中断请求寄存器。

19.3.5.7 中断源的概述

INTR_CAUSE 寄存器包含所有挂起 SAR 中断的概述。它允许 ISR 确定导致中断的原因。该寄存器包含 SAR_INTR_MASKED 的镜像副本。此外，它还有两位，它们汇总了所有通道的范围和饱和检测中断。它对 RANGE_INTR_MASKED 和 SATURATE_INTR_MASKED 寄存器中所有位进行逻辑 “ 或 ” 运算 （OR）（INJ_RANGE_INTR 和 INJ_SATURATE_INTR 除外）。

19.3.6 触发器

可通过以下两种方法触发扫描：

- 通过 DSI 接口 （dsi_trigger）传输周期性触发器。该触发器连接到 TCPWM 的输出端，还可将它连接到 UDB 或任意一个 GPIO 引脚。UDB 可以作为一个状态机使用，以查找一定的事件序列。
- 通过置位 SAR_SAMPLE_CTRL 寄存器中的 CONTINUOUS（连续）位，可以激活连续触发器。在该模式下，扫描完成后， SARSEQ 会立即启动下一次扫描，所以 SARSEQ 总是处于 BUSY （繁忙）状态。因此，必须忽略其它所有触发器。

尽管不需要硬件，但这两个触发器仍相互独立。当 DSI 触发器与连续触发器一致时，将同时有效处理这两个触发器 （可能为 DSI 触发器设置冲突中断）。

对于连续触发器，定序器通知 SAR ADC 开始进行采样 （已给的定序器处于闲置状态）前，只需要一个 SAR ADC 时钟周期。对于 DSI 触发器，这取决于触发配置的设置。

19.3.6.1 DSI 触发配置

- DSI 同步化

SARSEQ 的 DSI 接口以系统时钟频率 （clk_sys）运行；请参见第 83 页上的时钟系统章节了解详细信息。如果正在输入的 DSI 触发器信号与 AHB 时钟不同步，则需要通过双反转该信号 （默认）使其同步。但是如果 DSI 触发信号已经与 AHB 时钟同步，则可旁路这双反转操作。SAR_SAMPLE_CTRL 寄存器中的配置位 （DSI_SYNC_TRIGGER）控制双反转的旁路。DSI_SYNC_TRIGGER 影响到 DSI 脉冲触发信号所需要的触发宽度 （TW）和触发的时间间隔 （TI）。

- DSI 触发电平

DSI 触发可以是脉冲触发或电平触发；这取决于 SAR_SAMPLE_CTRL 寄存器中的配置位 DSI_TRIGGER_LEVEL。如果是电平触发，只要 DSI 触发信号保持为高电平， SAR 将启动新的扫描操作。当 DSI 触发信号是一个脉冲输入时，如果在 DSI 触发信号上检测到一个上升沿，则新的扫描将被触发。

- 传输时间

如果生成了 ‘dsi_trigger’，则通知 SAR ADC 开始进行采样前，需要一段传输时间。根据 DSI_SYNC_TRIGGER 和 DSI_TRIGGER_LEVEL 配置的不同，传输时间也不一样；表 19-5 显示的是传输的最大时间。两个触发脉冲时间间隔应该长于传输时间，否则第二个触发脉冲将被忽略。

SAR 被禁用 （ENABLED = 0）时，会忽略 DSI 触发器。

表 19-5. DSI 触发器的最长时间

DSI_TRIGGER 的最长传输时间	旁路同步 DSI_SYNC_TRIGGER = 0	使能同步 DSI_SYNC_TRIGGER = 1 （默认）
脉冲触发：DSI_TRIGGER_LEVEL = 0 （默认）	1 clk_sys+2 clk_sar	3 clk_sys+2 clk_sar
电平触发：DSI_TRIGGER_LEVEL=1	2 clk_sar	2 clk_sys+2 clk_sar

表 19-6. 触发信号要求

触发规范	要求
触发宽度 （TW）	TW 应足够大，以锁定触发器。如果 DSI_SYNC_TRIGGER = 1，则 TW ≥ 两个 clk_sys 周期。如果 DSI_SYNC_TRIGGER = 0，则 TW ≥ 一个 SAR 时钟周期。
触发的时间间隔 （TI）	与传输时间相比， DSI 脉冲触发信号的触发间隔应更长（如表 19-5 中所指定），否则，第二个触发脉冲将被忽略。

19.3.7 SAR ADC 状态

通过 SAR_STATUS 寄存器中的 BUSY 和 CUR_CHAN 字段，可以观察到 SAR 的当前状态。无论 SAR 正在对某个通道进行采样还是转换， BUSY （忙碌）位都会保持高电平； CUR_CHAN [4:0] 位表示正在采样的当前通道的编号 （通道 16 指示插入通道）。 SW_VREF_NEG 位表示 SAR ADC 中当前开关的状态，包括 DSI 控制和寄存器控制 （该开关将 V_{REF} 输入短路连接到 NEG 引脚）。

如果在终端扫描过程中被采样的WORK（工作）数据可用，那么会置位CHAN_WORK_VALID寄存器中的CHAN_WORK_VALID位。当 CHAN_RESULT_VALID 寄存器中的 CHAN_RESULT_VALID 被置位时，便意味着 RESULT （结果）数据是可用的，并且相应的 CHAN_WORK_VALID 位被清除。 SAR_AVG_STAT 寄存器中的 CUR_AVG_ACCU 和 CUR_AVG_CNT 字段指示了当前平均累加器的内容以及用于实现求平均的当前采样计数器值 （递减计数）。

SAR_MUX_SWITCH_STATUS 寄存器提供 MUX_SWITCH0 寄存器的当前开关状态。这些状态寄存器有助于调试 SAR。

19.3.8 低功耗模式

SAR ADC 的电流消耗可分为两个部分：SAR ADC 内核和 SARREF。可以通过某些方法来降低 SAR ADC 内核的功耗。最简单的方法是降低触发器频率，即降低每秒中进行的转换次数。另一个方法是对不要求高精度的通道采用较低的分辨率。这样，

最多可减少转换时间共 18 个周期中的四个（对于 8 位分辨率和最小的采样时间）。另外，SAR ADC 还提供了用于控制其总功耗的 ICONT_LV[1:0] 配置位。需要观察每个功耗设置的最大时钟速率。

表 19-7. 低功耗的 ICONT_LV

ICONT_LV[1:0]	SAR ADC 内核的相对功耗 [%]	最大频率 [MHz]	最短的采样时间 [周期]	最大采样率 （在 12 位分辨率时） [ksps]
0	100	18	4	1000
1	50	9	3	529
2	133	18	4	1000
3	25	4.5	2	281

除了控制 SAR ADC 内核的功耗外，还可以配置 VREF 缓冲器（若使用）的功耗。请注意，为了在不使用外部旁路电容时能够获得全 VDDA 范围 （1.7 V 到 5.5 V）， VREF 缓冲器必须运行于 2x 功耗模式。然而，不使用外部旁路电容时所支持的最大采样率仍是 100 ksps。为了达到 1 Msps 的采样率，需要使用外部旁路电容和一个 18 MHz 的时钟。更多详细信息，请参见表 19-8。

表 19-8. SAR VREF 功耗选项

PWR_CTRL_VREF [1:0]	要求外部旁路电容	相对 VREF 功耗 [%]	最大频率 [MHz]	最短的采样时间 [周期]	最大采样率 （在 12 位分辨率时） [ksps]	VDDA 范围
0	是	100	18	4	1000	1.7 V - 5.5 V
0	否	100	1.6	2	100	2.7 V - 5.5 V
2	否	200	1.6	2	100	1.7 V - 5.5 V
1 或 3	无效设置 — 不使用					

有了外部 VREF 便不需要使用 VREF 缓冲器和旁路电容，因此能够降低 SAR ADC 模块的总功耗。

19.3.9 系统操作

通过置位 ENABLED 位 （SAR_CTRL [31]）来使能 SAR 模拟模块后，请使用 SARSEQ 根据下列步骤开始进行 ADC 转换：

1. 设置 SAR ADC 控制模式：19.3.10 寄存器模式 或 19.3.11 DSI 模式
2. 通通过定序器 / 固件 /DSI 设置 SARMUX 模拟路由 （引脚 / 信号选择）
3. 设置全局 SARSEQ 转换配置
4. 对每个通道源 （如引脚地址）进行配置
5. 使能各通道
6. 设置触发器类型
7. 设置中断屏蔽
8. 启动触发源
9. 每个转换中断结束后，都要检索数据
10. 进行插入转换 （如果需要）

寄存器模式是指使用寄存器控制 SARMUX 和 SAR ADC 转换； DSI 模式是指使用来自 UDB 的 DSI 进行控制转换。表 19-9 显示了这两种控制模式间的主要区别。可通过设置 DSI_MODE 位 （SAR_CTRL [29]）使能 DSI 模式。

表 19-9. 各种控制模式间的差别

控制模式	寄存器	DSI
DSI_MODE	0	1
SARMUX 控制	定序器控制寄存器： SAR_CHANx_CONFIG、 SAR_MUX_SWITCH0、 SAR_MUX-_HW_SWITCH_CTRL SAR_CTRL 固件控制寄存器： SAR_MUX_SWITCH0、 SAR_MUX_HW_SWITCH_CTRL、 SAR_CTRL	DSI 控制信号：dsi_out、 dsi_oe、 dsi_swctrl、 dsi_sw_negvref 固件控制寄存器：SAR_MUX_SWITCH0、 SAR_MUX_HW_SWITCH_CTRL、 SAR_CTRL
全局配置	全局配置寄存器： SAR_CTRL、 SAR_SAMPLE_CTRL、 SAR_SAMPLE01、 SAR_SAMPLE23、 SAR_RANGE_THES、 SAR_RANGE_COND	全局配置寄存器： SAR_CTRL、 SAR_SAMPLE_CTRL、 SAR_SAMPLE01、 SAR_SAMPLE23、 SAR_RANGE_THES、 SAR_RANGE_COND
通道配置	通道配置寄存器： CHAN_CONFIG、 CHAN_EN、 INJ_CHAN_CONFIG	使用以下 DSI 信号： dsi_cfg_st_sel、 dsi_cfg_average、 dsi_cfg_resolution、 dsi_cfg_differential （CHAN_CONFIG、CHAN_EN、INJ_CHAN_CON-FIG 均被忽略）
触发器	全适用 固件触发 （SAR_START_CTRL[0]） DSI 触发 （dsi_trigger） 连续触发 （SAR_SAMPLE_CTRL [0]）	全适用 固件触发 （SAR_START_CTRL[0]） DSI 触发 （dsi_trigger） 连续触发 （SAR_SAMPLE_CTRL [0]）
中断	全适用	全适用 （DIS 信号上只有 EOS_INTR、 RANGE_INTR、 SATU-RATE_INTR 输出）
DSI 输出	支持	支持
结果数据	8 个通道结果寄存器以及一个插入通道结果寄存器	只有通道 0 结果寄存器可用
插入	支持	不支持
求平均	支持在一个引脚 / 信号上进行求平均	支持对不同引脚 / 信号求平均值

19.3.10 寄存器模式

使用寄存器配置 SAR ADC；这是最通用的方法。有关寄存器位定义的详细信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器计数参考手册。

19.3.10.1 SARMUX 模拟路由

在寄存器模式下，可以通过两种方法来控制 SARMUX 模拟路由，即：定序器和固件。

定序器控制

请注意，需要将 MUX_SWITCH_HW_CTRL 寄存器中的相应硬件控制位以及 MUX_SWITCH0 寄存器中的固件控制位全部设置为 ‘1’。请确保 SWITCH_DISABLE = 0；设置 SWITCH_DISABLE 以禁止定序器控制。

对于定序器控制，通过端口和引脚地址的组合可以指定通道转换的引脚或内部信号。 PORT_ADDR 位是 SAR_CHANx_CONFIG [6:4]，并且 PIN_ADDR 位为 SAR_CHANx_CONFIG [2:0]。表 19-10 显示的是 PORT_ADDR 和 PIN_ADDR 设置情况，以及相应的 SARMUX 选择。未使用的端口 / 引脚被保留给 PSoC 4 系列中的其他产品。

表 19-10. PORT_ADDR 和 PIN_ADDR

PORT_ADDR	PIN_ADDR	描述
0	0…7	SARMUX 的 8 个专用引脚
1	X	sarbus0[a]
1	X	sarbus1[a]
7	0	温度传感器
7	2	AMUXBUS-A
7	3	AMUXBUS-B

a. sarbus0 和 sarbus1 连接至 CTBm 模块的输出端 （该 CTBm 模块包含 opamp0/1）。更多信息，请参阅第 289 页上的微型连续时间模块 （CTBm）章节。PORT_ADDR = 1 时，无论 PIN_ADDR 的值如何，sarbus0 都会连接至 SAR ADC 的正端；当差分模式被使能且 PORT_ADDR=1 时，sarbus1 只能连接至 SAR ADC 的负端。

对于差分转换，负端的连接取决于正端的连接，它由 PORT_ADDR 和 PIN_ADDR 定义。通过设置 DIFFERENTIAL_EN，该通道会在偶数 / 奇数引脚对上进行差分转换（此引脚对在忽略 PIN_ADDR [0] 的情况下由引脚地址指定）。P3.0/P3.1，P3.2/P3.3，P3.4/P3.5， P3.6/P3.7 均为可用的差分对，用于定序器控制。固件或 DSI 可实现更加灵活的模拟。

对于单端转换，NEG_SEL 字段（即 SAR_CTRL [11:9]）用于决定连接至负输入的信号。在差分模式下，这些位都被忽略。负输入的选择影响到输入电压范围和有效分辨率。更多详细信息，请参考第 254 页上的负输入选择 。这些选项包括：V_{SSA}、V_{REF} 或来自带有 SARMUX 连接的八个引脚中任意一个引脚的外部输入。要想将负输入连接至 V_{REF}，必须置位 SAR_HW_CTRL_NEGVREF （SAR_CTRL[13]），因为 MUX_SWITCH_HW_CTRL 寄存器没有该硬件控制位。

固件控制

在默认情况下，SARMUX 会运行于固件控制模式。通过设置 SAR_MUX_SWITCH0 [29:0] 中的相应位，可单独控制 SAR ADC 的 VPLUS （正）和 VMINUS （负）输入。清除硬件开关控制寄存器中的相应位 （SAR_MUX_SWITCH_HW_CTR[n] = 0）。否则，硬件控制方法 （定序器 /DSI）将控制 SARMUX 模拟路由。

SAR_CTRL 寄存器的 SWITCH_DISABLE 位用于禁止 SAR 定序器使能路由开关。请注意，固件控制模式总能够关闭开关而不受该位的影响；但推荐将该位设置为 ‘1’。

NEG_SEL 字段 （SAR_CTRL [11:9]）决定了在单端模式下连接至 SAR ADC 负端 （Vminus）的信号。在差分模式下，这些位都被忽略。在单端模式下，使用定序器控制时，必须设置这些位。使用硬件控制时，将忽略 NEG_SEL 并设置 SAR_MUX_SWITCH0，以控制负输入。在特殊情况下，当 SAR_MUX_SWITCH0 未将内部 V_{REF} 连接至 Vminus 时，则需要将 NEG_SEL 设置为 ‘7’。负输入的选择影响到输入电压范围、信噪比和有效分辨率。更多详细信息，请参考第 254 页上的负输入选择 。

19.3.10.2 全局 SARSEQ 配置

一些适用于所有通道的选项是全局配置。在某些情况下，通道配置具有用于选用哪个全局配置部分的位。全局配置适用于寄存器控制模式和 DSI 控制模式。

SAR_CTRL、SAR_SAMPLE_CTRL、SAR_SAMPLE01、SAR_SAMPLE23、SAR_RANGE_THES 以及 SAR_RANGE_COND 都是全局配置寄存器。正在进行扫描时，通常不会更改这些配置。如果更改了使用过程中的配置设置，将产生 “ 未定义 ” 的结果。可以更改未使用的配置设置，而不会影响正在进行的扫描操作。

表 19-11. 全局配置寄存器

配置	控制寄存器	详细的参考信息
参考电压选择	SAR_CTRL[6:4]	19.3.3.1 参考电压选项
有符号 / 无符号选择	SAR_SAMPLE_CTRL [3:2]	19.3.1.3 结果数据格式
数据的左 / 右对齐	SAR_SAMPLE_CTRL [1]	19.3.1.3 结果数据格式
单端模式下的负输入选择	SAR_CTRL[11:9]	19.3.1.4 负输入选择
分辨率	SAR_SAMPLE_CTRL[0][a]	19.3.1.5 分辨率
数据采集时间	SAR_SAMPLE_TIME01 [25:0] SAR_SAMPLE_TIME32 [25:0]	19.3.1.6 采集时间
平均计数	SAR_SAMPLE_CTRL[7:4]	19.3.4.1 求平均
范围检测	SAR_RANGE_THRES [31:0] SAR_RANGE_COND [31:30]	19.3.4.2 范围检测

a. 需要通过 SAR_CHAN_CONFIG 寄存器中的 SAR_RESOLUTION 位来使能备用分辨率。如果备用分辨率未被使能，则 ADC 不再考虑由 SAR_SAMPLE_CTRL 寄存器设置的分辨率，在 12 位分辨率下运行。

19.3.10.3 通道配置

通道配置包括：

- 差分或单端模式的选项
- 全局配置选项包括：采样时间、分辨率、求平均使能
- DSI 输出使能

作为一般规则，只能在各扫描之间更改通道配置 （和全局配置相同）。但如果在扫描过程中未使能通道，则可随意更改该通道的配置而不会影响正在进行的扫描。如果违反此规则，将产生不确定的结果。。被自使能的通道是唯一例外；可在扫描过程中更改被使能的通道，则更改的内容将在下一次扫描时生效。更改使能的通道，可能会改变采样率。

表 19-12. 通道配置寄存器

配置	寄存器	详细的参考信息
单端 / 差分	SAR_CHANx_CONFIG [8]	19.3.1.1 单端模式和差分模式
采集时间选择	SAR_CHANx_CONFIG [13:12]	19.3.1.6 采集时间
分辨率选择	SAR_CHANx_CONFIG [9]	19.3.1.5 分辨率
求平均使能	SAR_CHANx_CONFIG [10]	19.3.4.1 求平均
DSI 输出使能	SAR_CHANx_CONFIG [30]	19.3.11.8 DSI 输出使能

可通过 SUB_RESOLUTION 位（SAR_SAMPLE_CTRL[0]）来选择所使用的备用分辨率（8 位或 10 位）。分辨率（SAR_CHANx_CONFIG [9]）用于决定使用默认的 12 位分辨率还是备用分辨率。当使能了求平均功能时，会忽略 SUB_RESOLUTION；这时分辨率将被调整为最大值（12 位）。

表 19-13. 分辨率

求平均功能	SUB_RESOLUTION SAR_SAMPLE_CTRL[0]	寄存器模式的分辨率 SAR_CHANx_CONFIG [9]	通道分辨率
禁用	0	1	8 位
禁用	1	1	10 位
禁用	0	0	12 位
禁用	1	0	12 位
启用	X	X	12 位

19.3.10.4 通道使能

通过 CHAN_EN 寄存器可以单独使能各个通道。发生下一个触发器时，所有使能的通道均被扫描。触发后，可立即更新通道使能，以为下一次扫描做准备。这不会影响到正在进行的扫描。请注意，这是一个例外；扫描过程中不应更改其它所有配置（全局或通道）。

19.3.10.5 中断屏蔽

共有六个中断源；每个都有一个中断屏蔽：

- “扫描结束”中断
- 溢出中断
- 冲突中断
- “插入转换结束”中断
- 范围检测中断
- 饱和检测中断

每个中断均有一个中断请求寄存器（INTR、SATURATE_INTR、RANGE_INTR）、一个软件中断设置寄存器（INTR_SET、SATURATE_INTR_SET、RANGE_INTR_SET）、一个中断屏蔽寄存器（INTR_MASK、SATURATE_INTR_MASK、RANGE_INTR_MASK）以及一个中断请求屏蔽结果寄存器（INTR_MASKED、SATURATE_INTR_MASKED、RANGE_INTR_MASKED）。也可以通过添加一个中断源寄存器来取得所有正在挂起的 SAR 中断概述，并允许 ISR 只要通过读取该寄存器即可确定中断源。

更多有关信息，请参考 19.3.5 中断。

19.3.10.6 触发器

可使用以下三种方法来启动 A/D 转换：

- 固件触发：SAR_START_CTRL[0]
- DSI 触发：dsi_trigger
- 连续触发：SAR_SAMPLE_CTRL [16]

更多有关信息，请参考 19.3.6 触发器。

19.3.10.7 每当转换中断结束后，都将检索数据

请确保每次扫描结束后都要读取结果寄存器中的数据；否则数据会因下次扫描的配置而被改变。

16 位数据寄存器用于为多达 8 个通道实现双缓冲（插入通道没有双缓冲功能）。双缓冲是指每个通道都有一个工作寄存器以及一个结果寄存器。采样该通道后，数据立即被写入到工作寄存器中。然后，当该扫描中全部使能通道的采样过程完成后，数据将从工作寄存器复制到结果寄存器内。

相应的工作数据有效时，将设置 CHAN_WORK_VALID 位，即在当前扫描过程中已对它进行了采样。扫描完成后，相应的 CHAN_RESULT_VALID 将被置位。置位 CHAN_RESULT_VALID 位时，相应的 CHAN_WORK_VALID 位将被清除。

为了便于固件，SAR_CHAN_WORK 寄存器中的位 [31] 是 SAR_CHAN_WORK_VALID 寄存器中相应位的镜像位。SAR_CHAN_RESULT 寄存器中的位 [29]、位 [30] 和位 [31] 都是 SAR_SATURATE_INTR、SAR_RANGE_INTR 和 SAR_CHAN_RESULT_VALID 寄存器中相应位的镜像位。请注意，这里所映射的中断位都是原始（未屏蔽）中断位。它有助于固件只要读取数据寄存器，即可检查数据的有效性。

若使能 DSI 输出，它会允许 UDB 处理 SARSEQ 结果数据，并且通道编号允许对不同通道内的数据采用不同的处理方法。有关详细说明，请参见 19.3.11.8 DSI 输出使能。

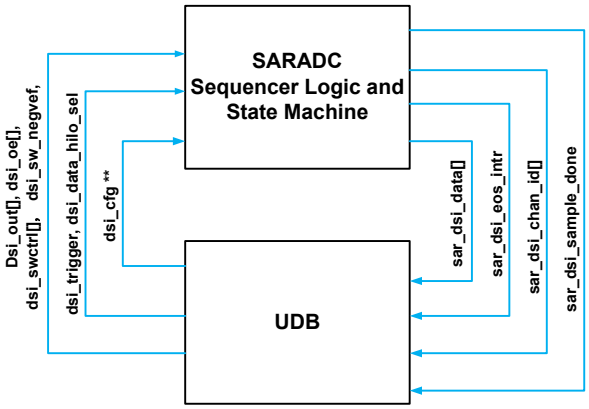
19.3.10.8 插入转换

通过置位起始位 INJ_START_EN（INJ_CHAN_CONFIG [31]），可以触发插入通道。要想避免正常自动扫描的冲突，推荐通过置位 INJ_CHAN_CONFIG [30] 使能紧接。被使能时，INJ_START_EN 会允许在下一次正常通道扫描结束时进行插入通道扫描。更多有关信息，请参考 19.3.4.4 插入通道。

19.3.11 DSI 模式

在 DSI 控制模式下，除全局配置（例如，中断屏蔽、范围检测设置以及触发器）外，几乎所有 SAR ADC 配置均可由 DSI 信号实现。DSI 模式和寄存器模式间的主要区别是：DSI 模式允许硬件动态控制 ADC 配置。图 19-12 是 SAR ADC 框图（图 19-1）的子集，它指定了 DSI 输入和输出信号。

图 19-12. DSI 控制模式框图



通过置位 SAR_CTRL 寄存器中的 DSI_MODE 位，可以选择 DSI 控制模式。在该模式下，SARSEQ 会忽略 CHAN_EN、CHAN_CONFIG、和 INJ_CHAN_CONFIG 中的所有通道配置。反而，它会使用由 DSI 信号传输的配置。

以下各 DSI 信号将被使用。

表 19-14. DSI 信号

信号	宽度	描述
sar_dsi_sample_done	1	用于指示 SAR ADC 采样操作已完成的脉冲。可以更改开关，以选择需要转换的下一个信号（与 SAR ADC 下个输出相同）
sar_dsi_chan_id_valid	1	通道 ID 的有效信号
sar_dsi_chan_id	4	常规模式：通道 ID （正在转换的通道 ID）（早期） DSI 控制模式： [0] = 饱和检测中断 [1] = 范围检测中断 （发生数据输出时，该信号为有效）
sar_dsi_data_valid	1	数据值的有效信号
sar_dsi_data	12	某个通道转换 （并求平均，若可用）的结果；内部平均结果为 16 位宽。 如果 dsi_data_hilo_sel = 0，则 sar_dsi_data[11:0] = sar_data[11:0]。 如果dsi_data_hilo_sel = 1，则sar_dsi_data[7:0] = sar_data[15:8]，且sar_dsi_data[11:8] = <未定义>。
sar_dsi_eos_intr	1	用于表示 SARSEQ 已完成所有使能通道的扫描操作的 “ 结束扫描 ” 中断。
dsi_out	8	dsi_out[0]=1， P3.0 连接至 ADC dsi_out[1]=1， P3.1 连接至 ADC … dsi_out[7]=1， P3.7 连接至 ADC **注意**：MUX_SWITCH0 的配置决定了该引脚连接到 Vplus 还是 Vminus。
dsi_oe	4	dsi_oe[0] = 1， AMUXBUSA 连接至 ADC dsi_oe[1] = 1， AMUXBUSB 连接至 ADC dsi_oe[2] = 1， opamp0 输出连接至 ADC dsi_oe[3] = 1， opamp1 输出连接至 ADC **注意**：MUX_SWITCH0 的配置决定了该信号连接到 Vplus 还是 Vminus。

表 19-14. DSI 信号

信号	宽度	描述
dsi_swctrl[0]	1	该信号控制着 SARMUX 模拟信号的转换情况，并将 Vssa_kelvin 连接至 Vminus
dsi_swctrl[1]	1	该信号控制着 SARMUX 模拟信号的转换情况，并将 temp_sens 连接至 Vplus
dsi_sw_negvref	1	该信号控制着 SAR ADC 内部信号的转换情况，并将 V_{REF} 输入连接至 NEG 输入
dsi_cfg_st_sel	2	DSI 控制模式的配置控制：选择 4 个全局采样时间中的一个
dsi_cfg_average	1	DSI 控制模式的配置控制：使能求平均功能
dsi_cfg_resolution	1	DSI 控制模式的配置控制：0 = 12 位分辨率 1 = 使用全局配置的其它分辨率 （8 位或 10 位）
dsi_cfg_differential	1	DSI 控制模式的配置控制：0 = 单端模式， 1 = 差分模式
dsi_trigger	1	对所有使能通道进行 SARSEQ 扫描的触发器
dsi_data_hilo_sel	1	为 sar_dsi_data[7:0] 选择高或低字节输出。该信号是完全异步的（它影响着 sar_dsi_data 信号而不需要使用任何有关的时钟）。

19.3.11.1 固件模拟路由

在 DSI 模式下，通过 DSI 信号和固件，可以实现模拟路由。无论寄存器的配置如何，固件控制始终有效，并且它在寄存器模式下也保持有效状态。请参考 19.3.2.1 模拟路由，了解有关固件控制的详细信息。

19.3.11.2 DSI 模拟路由

来自 UDB 模块的 DSI 信号用于控制 SARMUX 开关。在 DSI 控制模式下，SARSEQ 不会输出任何来自定序器使能的开关。图 19-4 显示 DSI 可以控制所有开关 （除用于测试设计的 DTP 开关外）。因此，在该模式下， SAR ADC 的负输入和正输入可连接至任何开关。

除了 DSI 信号，寄存器中的相应硬件和固件控制位都需要置位。这些寄存器和信号包括 SAR_MUX_SWITCH0 [n] = 1 以及 SAR_MUX_SWITCH_HW_CTRL[n] = 1。当 V_{REF} 连接至负输入时，除了 DSI 信号以外，应设置 SAR_CTRL [11:9] = 7 （固件控制字段）以及 SAR_CTRL [13] = 1 （硬件控制位）。

在单端模式下，通过 dsi_swctrl[0] 和 dsi_sw_neg v_{REF}， DSI 信号能够控制 SAR ADC 的负端。如果置位了 NEG_SEL （SAR_CTRL[11:9]），则仅有 NEG_SEL=7 可用；其它数值都被忽略。

表 19-15 显示了 DSI 信号。

表 19-15. DSI 模拟路由

信号	宽度	描述
dsi_out	8	dsi_out[0]=1， P3.0 连接至 ADC dsi_out[1]=1， P3.1 连接至 ADC … dsi_out[7]=1， P3.7 连接至 ADC **注意**：该引脚连接至 vplus 还是 vminus，均由 MUX_SWITCH0 配置决定的。
dsi_oe	4	dsi_oe[0] = 1， AMUXBUSA 连接至 ADC dsi_oe[1] = 1， AMUXBUSB 连接至 ADC dsi_oe[2] = 1， sarbus0 输出连接至 ADC dsi_oe[3] = 1， sarbus1 输出连接至 ADC **注意**：该信号连接至 Vplus 还是 Vminus 是由 MUX_SWITCH0 配置决定的。
dsi_swctrl[0]	1	该信号控制着 SARMUX 模拟信号的转换情况，并将 V_{SSA} 连接至 Vminus
dsi_swctrl[1]	1	该信号控制着 SARMUX 模拟信号的转换情况，并将温度传感器连接至 Vplus
dsi_sw_negvref	1	该信号控制着 SAR ADC 内部信号的转换情况，并将 V_{REF} 输入连接到 NEG 输入端

19.3.11.3 全局 SARSEQ 配置

全局配置适用于寄存器模式以及 DSI 控制模式。更多详细信息，请参考 19.3.10.2 全局 SARSEQ 配置。

19.3.11.4 DSI 通道配置

对于 DSI 控制模式，只有通道 0 可用。通过 DSI 信号，可执行通道 0 配置，如表 19-16 所示。 CHAN_EN 和 CHAN_CONFIG 与 INJ_CHAN_CONFIG 中的通道配置均被忽略。

通过设置 DSI_SYNC_CONFIG，可以使 dsi_cfg_* 信号与 SAR 时钟域（实际的 clk_hf）同步。SAR 在低频率下运行时，可能需要旁路同步。

表 19-16. DSI 通道配置

信号	宽度	配置	说明
dsi_cfg_st_sel	2	数据采集时间	DSI 控制模式的配置控制：选择 4 个全局采样时间中的一个
dsi_cfg_average	1	求平均使能	DSI 控制模式的配置控制：使能求平均功能
dsi_cfg_resolution	1	分辨率	DSI 控制模式的配置控制： 0：12 位分辨率 1：使用全局配置分辨率位 SUB_RESOLUTION （8 位或 10 位）
dsi_cfg_differential	1	差分 / 单端	DSI 控制模式的配置控制： 0：单端模式 1：差分模式

19.3.11.5 中断

有关 SAR ADC 中断的介绍，请查阅第 273 页上的中断屏蔽 。所有中断屏蔽均正常工作于寄存器控制模式。并非所有中断都是在 DSI 模式中生成的； SATURATE_INTR、 RANGE_INTR 以及 EOS_INTR 是通过 DSI 信号发送的。

- 数据以及SATURATE_INTR都是在dsi_chan_id[0]上输出的；另外SATURATE_INTR[0]被设置为DSI控制模式，因为在DSI模式中只有通道 0 可用。
- 数据以及RANGE_INTR都是在dsi_chan_id[1]上输出；RANGE_INTR[0]在DSI控制模式下设置，因为在DSI模式中只有通道 0 可用。
- 通道使能都被忽略，即每次触发只能执行一次转换。为每个转换生成 EOS_INTR 中断。
- 始终通过 DSI 信号 sar_dsi_eos_intr （dsi_data_valid 的副本）发送 EOS_INTR 中断。

表 19-17 列出了由 DSI 信号发送的中断。

表 19-17. DSI 信号中断

信号	宽度	说明
sar_dsi_chan_id	4	寄存器模式：通道 ID （正在转换的通道的 ID）。 DSI 控制模式： [0] = 饱和检测中断； [1] = 范围检测中断 （发生数据输出时，该信号有效）。
sar_dsi_eos_intr	1	用于表示 SARSEQ 已完成对所有使能通道进行的扫描操作的 “ 扫描结束 ” 中断。

19.3.11.6 触发器

DSI 控制模式通常与 DSI 触发器一起使用。但其它触发源（如固件触发和连续触发）也受支持。触发配置与寄存器控制模式中的配置相同。更多有关信息，请参考第 267 页上的触发器 。

针对 DSI 触发，配置设置 （dsi_cfg_*）以及开关设置应稳定，该设置不能迟于 dsi_trigger 被传送的周期。它们应保持稳定状态，直到 sar_dsi_sample_done 的上升沿到来为止。

19.3.11.7 检索数据

结果数据和通道编号都在 sar_dsi_data 上输出。它相当于在寄存器控制模式下将 dsi_out_en 置高。更多有关信息，请参考 19.3.11.8 DSI 输出使能。每当转换完成后，会将数据写入到 CHAN_WORK0 和 CHAN_RESULT0 寄存器中。

19.3.11.8 DSI 输出使能

如果置位了 DSI_OUT_EN 位（即 SAR_CHANx_CONFIG[31]），结果数据和通道编号也在 DSI 总线（sar_dsi_data、sar_dsi_chan_id）上输出，然后被存储在正常结果寄存器中。它允许 UDB 处理 SARSEQ 结果数据，并且通道数量允许对不同通道采用不同的处理方法。

在 DSI 总线上发出的数据格式与在结果寄存器中存储的数据格式相同。但在默认情况下，只输出 12 个最低有效位（LSB）；除非需要多于 12 位，否则不推荐使用左对齐。要想获取 8 个高最低有效位（LSB），需要将 dsi_data_hilo_sel 输入设置为‘1’。如果想从结果寄存器获取全 16 位数据，首先要将 dsi_data_hilo_sel 设置为 ‘0’，从而获得低 12 位数据，然后设置 dsi_data_hilo_sel = 1 以获得高 8 位数据。需要处理额外数据，以便处理数据重叠现象。

SAR ADC 完成通道采样后，会发出通道编号（sar_dsi_chan_id）。该通道的编号能够自身触发 UDB，以驱动某些 GPIO 引脚（这些引脚可以为片外器件供电或断电）。这样会允许随后的一个通道在同一扫描操作中对模拟输入引脚进行扫描（该操作需要一个较长的采样时间）。

请注意，数据在转换完成的一个周期后被输出。在 DSI 总线上两个系统时钟周期内维持通道编号、数据以及它们各自的有效位。

表 19-18. DSI 输出信号

信号	宽度	说明
sar_dsi_sample_done	1	用于指示 SAR ADC 采样操作已完成的脉冲。可以更改开关，以选择需要转换的下一个信号（与 SAR ADC 下个输出相同）
sar_dsi_chan_id_valid	1	通道 ID 的有效信号
sar_dsi_chan_id	4	常规模式：通道 ID（正在转换的通道 ID）（早期） DSI 控制模式： [0] = 饱和检测中断 [1] = 范围检测中断（发生数据输出时，该信号为有效）
sar_dsi_data_valid	1	数据值的有效信号
sar_dsi_data	12	一个通道的转换（和求平均，若可用）结果。内部平均值为 16 位宽。 如果 dsi_data_hilo_sel = 0，则 sar_dsi_data[11:0] = sar_data[11:0] 如果 dsi_data_hilo_sel = 1，则 sar_dsi_data[7:0] = sar_data[15:8]，且 sar_dsi_data[11:8] = < 未定义 >
sar_dsi_eos_intr	1	用于表示 SARSEQ 已完成所有使能通道的扫描操作的 “ 结束扫描 ” 中断。
dsi_data_hilo_sel	1	为 sar_dsi_data[7:0] 选择高或低字节输出。该信号是完全异步的（它对 sar_dsi_data 信号产生影响而不需要任何相关时钟）

19.3.12 模拟路由配置示例

表 19-19 显示的是定序器控制、固件控制和 DSI 控制的引脚和信号选项。

表 19-19. 模拟路由配置示例

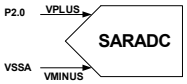
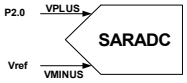
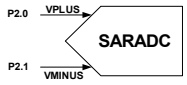
	定序器控制	固件控制	DSI 控制
	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 0 （CTRL[30]） PORT_ADDR = 0 （CHANx_CONFIG[6:4]） PIN_ADDR = 0 （CHANx_CONFIG[2:0]） NEG_SEL = 0 （CTRL [11:9]） MUX_SWITCH0[0] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[16]= 1	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 1 （CTRL[30]） MUX_SWITCH0[0] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[0] = 0 MUX_SWITCH_HW_CTRL[16] = 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 0 dsi_out [0] =1 dsi_swctrl[0]=1 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH0 [16] = 1
	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 0 （CTRL[30]） PORT_ADDR = 0 （CHANx_CONFIG[6:4]） PIN_ADDR = 0 （CHANx_CONFIG[2:0]） NEG_SEL = 7 （CTRL [11:9]） MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0]=1 HW_CTRL_NEGVREF =1 （CTRL[13]）	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 1 （CTRL[30]） MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] =0 NEG_SEL = 7 （CTRL [11:9]） HW_CTRL_NEGVREF =0 （CTRL[13]）	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 0 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 dsi_out [0] =1 dsi_sw_negvref =1 HW_CTRL_NEGVREF =1 （CTRL[13]）
	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） SWITCH_DISABLE = 0 （CTRL[30]） PORT_ADDR = 0 （CHANx_CONFIG[6:4]） PIN_ADDR = 0 或 PIN_ADDR = 1 （CHANx_CONFIG[2:0]） MUX_SWITCH0[0] = 1 MUX_SWITCH0[9] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH_HW_CTRL[1] = 1	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） SWITCH_DISABLE = 1 （CTRL[30]） MUX_SWITCH0[0] = 1 MUX_SWITCH0[9] = 1 MUX_SWITCH_HW_CTRL[0] = 0 MUX_SWITCH_HW_CTRL[1] = 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 1 dsi_out [0] =1 dsi_out [1] =1 MUX_SWITCH0[0] = 1 MUX_SWITCH_HW_CTRL[0] = 1 MUX_SWITCH0 [9] = 1 MUX_SWITCH_HW_CTRL[1]=1

表 19-19. 模拟路由配置示例 <Italic> （续）

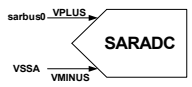
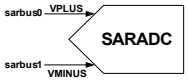
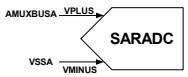
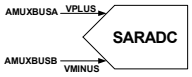
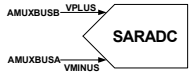
	定序器控制	固件控制	DSI 控制
	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 0 （CTRL[30]） PORT_ADDR = 1 （CHANx_CONFIG[6:4]） NEG_SEL = 0 （CTRL [11:9]） MUX_SWITCH0[22] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[22] =1 MUX_SWITCH_HW_CTRL[16] =1 **注意**：为了能够控制端口 / 引脚，不支持将 sarbus1 连接至 VPLUS	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） SWITCH_DISABLE = 1 （CTRL[30]） MUX_SWITCH0[22] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[22] = 0 MUX_SWITCH_HW_CTRL[16] = 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 0 dsi_oe [2] =1 dsi_swctrl[0]=1 MUX_SWITCH0 [16] = 1 MUX_SWITCH0[22] = 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH_HW_CTRL[22] =1

表 19-19. 模拟路由配置示例 <Italic> （续）

	定序器控制	固件控制	DSI 控制
	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 0** （CTRL[30]） PORT_ADDR = 1 （CHANx_CONFIG[6:4]） MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22]=1 MUX_SWITCH_HW_CTRL[23]=1	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） SWITCH_DISABLE = 1 （CTRL[30]） MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22] = 0 MUX_SWITCH_HW_CTRL[23] = 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 1 dsi_oe [2] = 1 dsi_oe [3] = 1 MUX_SWITCH0[22] = 1 MUX_SWITCH0[25] = 1 MUX_SWITCH_HW_CTRL[22]=1 MUX_SWITCH_HW_CTRL[23]=1
	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 0** （CTRL[30]） PORT_ADDR = 7 （CHANx_CONFIG[6:4]） PIN_ADDR = 2 （CHANx_CONFIG[2:0]） **NEG_SEL = 0** （CTRL [11:9]） MUX_SWITCH0[18] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[16]= 1	DIFFERENTIAL_EN = 0 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 1** （CTRL[30]） MUX_SWITCH0[18] = 1 MUX_SWITCH0[16] = 1 MUX_SWITCH_HW_CTRL[18]= 0 MUX_SWITCH_HW_CTRL[16]= 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 0 dsi_oe [0] = 1 dsi_swctrl[0]=1 MUX_SWITCH0[18] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[16]=1 MUX_SWITCH0 [16] = 1
	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 0** （CTRL[30]） PORT_ADDR = 7 （CHANx_CONFIG[6:4]） PIN_ADDR = 2 （CHANx_CONFIG[2:0]） MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[19]= 1	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 1** （CTRL[30]） MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 0 MUX_SWITCH_HW_CTRL[19]= 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 1 dsi_oe [0] = 1 dsi_oe [1] = 1 MUX_SWITCH0[18] = 1 MUX_SWITCH0[21] = 1 MUX_SWITCH_HW_CTRL[18]= 1 MUX_SWITCH_HW_CTRL[19]= 1
	不支持。 端口 / 引脚控制的差分对是固定的	DIFFERENTIAL_EN = 1 （CHANx_CONFIG[8]） **SWITCH_DISABLE = 1** （CTRL[30]） MUX_SWITCH0[19] = 1 MUX_SWITCH0[20] = 1 MUX_SWITCH_HW_CTRL[18] =0 MUX_SWITCH_HW_CTRL[19] = 0	DSI_MODE = 1 （CTRL[29]） dsi_cfg_differential = 1 dsi_oe [0] = 1 dsi_oe [1] = 1 MUX_SWITCH0[19] = 1 MUX_SWITCH0[20] = 1 MUX_SWITCH_HW_CTRL[18] =1 MUX_SWITCH_HW_CTRL[19] = 1

19.3.13 温度传感器配置

一个片上温度传感器可适用于温度感应以及基于温度的校准。对于温度传感器，差分转换不可用（转换结果为未定义）。因此，该转换始终运行于单端模式。内部参考电压为 1.024 V。

可通过三种方法将某个引脚或信号连接到 SAR ADC。表 19-20 列出了将温度传感器连接至 SAR ADC 的方法。置位 MUX_FW_TEMP_VPLUS 位（即 SAR_MUX_SWITCH0[17]）可使能温度传感器，并且可将它的输出连接到 SAR ADC 的 VPLUS 上；如果清除该位，将通过切断温度传感器的偏置电流来禁用它。

表 19-20. 将温度传感器连接至 SAR ADC

控制方法	设置
定序器	DIFFERENTIAL_EN = 0 （SAR_CHANx_CONFIG[8]） VREF_SEL = 0 （SAR_CTRL[6:4]） PORT_ADDR = 7 （SAR_CHANx_CONFIG[6:4]） PIN_ADDR = 0 （SAR_CHANx_CONFIG[2:0]） SWITCH_DISABLE = 0 （SAR_CTRL[30]） SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16] = 1 SAR_MUX_SWITCH_HW_CTRL[17] = 1 NEG_SEL = 0 （SAR_CTRL [11:9]）被覆盖为 0[a]
固件	DIFFERENTIAL_EN = 0 （SAR_CHANx_CONFIG[8]） VREF_SEL = 0 （SAR_CTRL[6:4]） SWITCH_DISABLE = 1 （SAR_CTRL[30]） SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16]= 0 SAR_MUX_SWITCH_HW_CTRL[17]= 0 NEG_SEL = 0 （SAR_CTRL [11:9]）被覆盖为 0[a]
DSI	SWITCH_DISABLE = 1 （SAR_CTRL[30]） VREF_SEL = 0 （SAR_CTRL[6:4]） 设置 DSI 信号： dsi_cfg_differential = 1 dsi_swctrl[1] = 1 dsi_swctrl[0] = 1 SAR_MUX_SWITCH0[16] = 1 SAR_MUX_SWITCH0[17] = 1 SAR_MUX_SWITCH_HW_CTRL[16] = 1 SAR_MUX_SWITCH_HW_CTRL[17] = 1 NEG_SEL = 0 （SAR_CTRL [11:9]）被覆盖为 0[a]

a. 对于温度传感器，需要将 NEL_SEG （SAR_CTRL [11:9]）覆盖为 0。

19.4 寄存器

名称	偏移地址	数量	宽度	说明
SAR_CTRL	0x0000	1	32	全局配置寄存器 模拟控制寄存器
SAR_SAMPLE_CTRL	0x0004	1	32	全局配置寄存器 采样控制寄存器
SAR_SAMPLE_TIME01	0x0010	1	32	全局配置寄存器 采样时间规范 ST0 和 ST1
SAR_SAMPLE_TIME23	0x0014	1	32	全局配置寄存器 采样时间规范 ST2 和 ST3
SAR_RANGE_THRES	0x0018	1	32	全局范围检测阈值寄存器
SAR_RANGE_COND	0x001C	1	32	全局范围检测模式寄存器
SAR_CHAN_EN	0x0020	1	32	通道使能位
SAR_START_CTRL	0x0024	1	32	启动控制寄存器 （固件触发）
SAR_CHAN_CONFIG	0x0080	8	32	通道配置寄存器
SAR_CHAN_WORK	0x0100	8	32	通道工作数据寄存器
SAR_CHAN_RESULT	0x0180	8	32	通道结果数据寄存器
SAR_CHAN_WORK_VALID	0x0200	1	32	通道工作数据寄存器的有效位
SAR_CHAN_RESULT_VALID	0x0204	1	32	通道结果数据寄存器的有效位
SAR_STATUS	0x0208	1	32	内部 SAR 寄存器的当前状态 （用于调试）
SAR_AVG_STAT	0x020C	1	32	当前的平均值状态 （用于调试）
SAR_INTR	0x0210	1	32	中断请求寄存器
SAR_INTR_SET	0x0214	1	32	中断设置请求寄存器
SAR_INTR_MASK	0x0218	1	32	中断屏蔽寄存器
SAR_INTR_MASKED	0x021C	1	32	中断屏蔽请求寄存器：如果该值为非零值，则输入到 NVIC 的 SAR 中断信号为高电平。读取时，该寄存器反映中断请求和屏蔽寄存器之间的按位逻辑与 （AND）运算
SAR_SATURATE_INTR	0x0220	1	32	饱和中断请求寄存器
SAR_SATURATE_INTR_SET	0x0224	1	32	饱和中断设置请求寄存器
SAR_SATURATE_INTR_MASK	0x0228	1	32	饱和中断屏蔽寄存器
SAR_SATURATE_INTR_MASKED	0x022C	1	32	饱和中断屏蔽请求寄存器
SAR_RANGE_INTR	0x0230	1	32	范围检测中断请求寄存器
SAR_RANGE_INTR_SET	0x0234	1	32	范围检测中断设置请求寄存器
SAR_RANGE_INTR_MASK	0x0238	1	32	范围检测中断屏蔽寄存器
SAR_RANGE_INTR_MASKED	0x023C	1	32	范围中断屏蔽请求寄存器
SASR_INTR_CAUSE	0x0240	1	32	中断源寄存器
SAR_INJ_CHAN_CONFIG	0x0280	1	32	插入通道配置寄存器
SAR_INJ_RESULT	0x0290	1	32	插入通道结果寄存器
SAR_MUX_SWITCH0	0x0300	1	32	SARMUX 固件开关控制
SAR_MUX_SWITCH_CLEAR0	0x0304	1	32	SARMUX 固件开关控制清除
SAR_MUX_SWITCH_HW_CTRL	0x0340	1	32	SARMUX 硬件开关控制
SAR_MUX_SWITCH_STATUS	0x0348	1	32	SARMUX 开关状态
SAR_PUMP_CTRL	0x0380	1	32	开关泵控制

20. 低功耗比较器

PSoC® 4 器件具有两个低功耗比较器。这些比较器可以在所有的系统功耗模式（停止模式除外）下快速进行模拟信号比较。有关器件的各种功耗模式的详细信息，请参见第 101 页上的功耗模式章节的内容。可将正向和负向输入连接到专用的 GPIO 引脚或 AMUXBUS-A/AMUXBUS-B。 CPU 通过状态寄存器读取该比较器输出。该输出能作为中断源或唤醒源使用，此外，它还可以反馈到 DSI 进行处理或路由到 GPIO。

20.1 特性

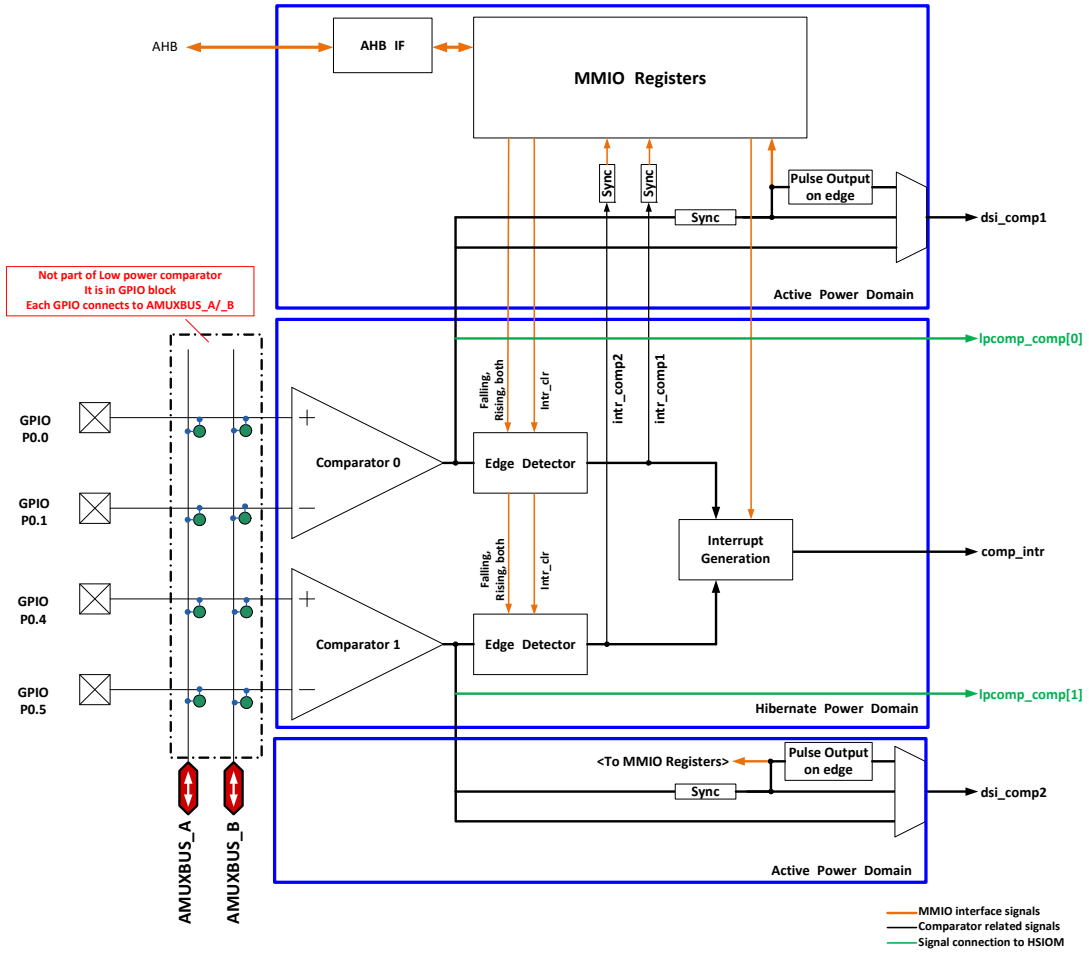
PSoC 4 比较器具有以下特性：

- 可配置的正向和负向输入
- 可编程的功耗和速度模式
- 支持超低功耗模式 （< 4 µA）
- 可选的 10 mV 输入迟滞
- 低输入偏移电压 （调整后该电压低于 4 mV）
- 支持在深度睡眠 / 休眠模式下的唤醒源

20.2 框图

图 20-1 显示了低功耗比较器的框图。

图 20-1. 低功耗比较器框图



20.3 工作原理

下面各部分介绍了 PSoC 4 低功耗比较器的操作情况，包括输入配置、功耗和速度模式、输出和中断配置、迟滞、从低功耗模式唤醒、比较器时钟以及偏移调整。

20.3.1 输入配置

可用的比较器输入包括：

- 来自专用输入引脚的正向和负向输入。
- 从任何引脚输入 AMUXBUS 的正向和负向输入（在深度睡眠模式下不可用）。
- 一个来自外部引脚的输入和一个来自内部生成信号的输入。可以将两个输入连接到比较器的正向或负向输入。通过模拟 AMUXBUS，将内部生成的信号连接到比较器输入端。
- 来自内部生成信号的正向和负向输入。从内部产生的信号通过 AMUXBUS-A/AMUXBUS-B 连接到比较器输入。

从图 20-1 可见，P0.0 和 P0.1 连接到比较器 0 的正向和负向输入； P0.4 和 P0.5 连接到比较器 1 的输入端。另外还要注意：AMUXBUS 网络并不直接连接到比较器输入。因此，比较器连接是通过相应的输入引脚被路由到 AMUXBUS 网络的。将 AMUXBUS 用于连接比较器输入时，这些输入引脚不可用于其他目的。在 AMUXBUS 被用于比较器输入连接的设计中，它们应该保持为开路状态。请注意，在深度睡眠和休眠模式下 AMUXBUS 连接不可用。如果需要在深度睡眠或休眠模式下进行操作，则必须将低功耗比较器连接到专用引脚上。该限制还包括任何内部生成信号的路由，因为它们也通过 AMUXBUS 进行连接。请参见第 69 页上的 I/O 系统章节，以了解 GPIO 到 AMUXBUS A/B 的连接或比较器输入的 GPIO 设置。

20.3.2 输出和中断配置

在 LPCOMP_CONFIG 寄存器中（表 20-1），OUT1 位 [6] 和 OUT2 位 [14] 分别提供了比较器 0 和比较器 1 的输出。将比较器输出锁存到LPCOMP_CONFIG寄存器中的OUTx位前，必须使这些输出与 SYSCLK 同步。每个比较器的输出被连接到一个对应的边沿检测器模块。该模块会确定触发中断的边沿。通过 LPCOMP_CONFIG 寄存器中的 INTTYPE1 位 [5:4] 和 INTTYPE2 位 [13:12]，可以配置边沿选择和中断使能。通过使用 INTTYPEx 位，可以将中断类型设置为禁用、上升沿、下降沿或双边沿，如表 20-1 所示。

通过 HSIOM，可以将每个比较器的输出直接路由到 GPIO 引脚上。在 HSIOM 中，比较器输出作为深度睡眠模式的源 2 连接。有关 HSIOM 的详细信息，请参考第 77 页上的高速 I/O 矩阵 。请参考器件数据手册，以详细了解有关支持低功耗比较器输出的引脚的信息。这些引脚上的输出是直接从比较器输出的，并且它们不同步。因为这些输出可作为引脚的深度睡眠源使用，所以比较器输出也可用于深度睡眠模式。请注意，HSIOM 在休眠模式下不可用。因此，受支持引脚上的比较器输出在休眠模式下亦不可用。

另外，通过 DSI （即图 20-1 中的 dsi_comp1/dsi_comp2 信号），可以将比较器输出单独路由到引脚或其他模块。可将连接至 DSI 的比较器输出配置为异步输出、与 SYSCLK 同步的输出，或比较器输出上升沿上的同步脉冲输出等选项。LPCOMP_CONFIG 寄存器的 DSI_BYPASS1 位 [16] 和 DSI_LEVEL1 位 [17] 用于将比较器 0 的输出配置为 DSI。同样，LPCOMP_CONFIG 寄存器中的 DSI_BYPASS2 位 [20] 和 DSI_LEVEL2 位 [21] 用于将比较器 1 的输出配置为 DSI。更多详细信息，请参见表 20-1 中的内容。

在某个边沿上，比较器会触发中断（图 20-1 中的intr_comp1/intr_comp2 信号）。比较器 0 和比较器 1 的中断请求分别被寄存在 LPCOMP_INTR 寄存器的 COMP1 位 [0] 和 COMP2 位 [1] 上。比较器 0 和比较器 1 共享了一个通用中断 （即图 20-1 中的 comp_intr 信号）。该中断是由两个中断经过逻辑 OR 运算得到的，并且它还被映射为 CPU NVIC 中低功耗比较器模块的中断。更多详细信息，请参考第 57 页上的中断章节。如果在设计中使用了这两个比较器，则必须在中断服务子程序中读取LPCOMP_INTR寄存器的COMP1和/或COMP2位，以确定触发中断的比较器。另外，通过使用 LPCOMP_INTR_MASK 寄存器的 COMP1_MASK 位 [0] 和 COMP2_MASK 位 [1]，可以屏蔽发给 CPU 的比较器 0 和比较器 1 中断。其中，CPU 只负责处理屏蔽中断。处理中断后，固件需要将 ‘1’ 写入到 LPCOMP_INTR 寄存器的 COMP1 和 COMP2 位，以清除该中断。如果未清除该中断，那么下一个比较事件将不会触发中断，并且 CPU 将无法处理该事件。在活动和睡眠模式下， dsi_comp1/dsi_comp2 输出会被路由到 UDB 映射中断，以单独处理每个比较器的触发。但在深度睡眠和休眠模式下，不能实现 UDB/DSI 路由。

LPCOMP 中断（comp1_intr/comp2_intr）与 SYSCLK 同步。清除 dsi_comp1/dis_comp2 和 comp1_intr/comp2_intr 的操作是同步进行的。

在活动和睡眠模式下，通过 UDB 中的 DSI 路由可以将 dsi_comp1/dsi_comp2 路由到 GPIO 或其他模块（同步或异步）。UDB DSI 输出有一个附加的同步器。有关 DSI 信号同步的详细信息，请参见第 161 页上的通用数字模块 （UDB）章节。在深度睡眠和休眠模式下，由于 UDB 被断电，所以不能进行上述路由操作。此外，如果将 dsi_comp1/dsi_comp2 路由到 UDB 以进行后续的处理操作，那么，时序则由用户的算法和同步器的选择决定。

通过使用 LPCOMP_INTR_SET 寄存器中的 [1:0] 位，可以激活一个软件调试的中断。

在深度睡眠和休眠模式下，可以通过比较器边沿事件来激活唤醒中断控制器 （WIC），进而唤醒 CPU。因此， LPCOMP 在低功耗模式下仍能监控一个指定的信号。

表 20-1. LPCOMP_CONFIG 寄存器中的输出和中断配置

寄存器 [Bit_Pos]	Bit_Name	说明
LPCOMP_CONFIG[6]	OUT1	比较器 0 的当前 / 瞬间输出值
LPCOMP_CONFIG[14]	OUT2	比较器 1 的当前 / 瞬间输出值
LPCOMP_CONFIG[5:4]	INTTYPE1	设置触发 IRQ 的比较器 0 边沿 00：禁用 01：上升沿 10：下降沿 11：双边沿
LPCOMP_CONFIG[13:12]	INTTYPE2	设置触发 IRQ 的比较器 1 边沿 00：禁用 01：上升沿 10：下降沿 11：双边沿

表 20-1. LPCOMP_CONFIG 寄存器中的输出和中断配置

寄存器 [Bit_Pos]	Bit_Name	说明
LPCOMP_CONFIG[16]	DSI_BYPASS1	比较器 0 旁路 DSI 输出的输出同步 0：输出与 SYSCLK 同步 1：旁路 / 输出异步
LPCOMP_CONFIG[17]	DSI_LEVEL1	比较器 0 的 DSI 输出电平 0：脉冲输出 — 在上升沿上生成宽度为两个 SYSCLK 周期的脉冲 1：电平
LPCOMP_CONFIG[20]	DSI_BYPASS2	比较器 1 旁路 DSI 输出的输出同步 0：输出与 SYSCLK 同步 1：旁路 / 输出异步
LPCOMP_CONFIG[21]	DSI_LEVEL2	比较器 1 的 DSI 输出电平 0：脉冲输出 — 在上升沿上生成宽度为两个 SYSCLK 周期的脉冲 1：电平
LPCOMP_INTR[0]	COMP1	比较器 0 中断：硬件在比较器 0 触发时设置该中断。通过写入 ‘1’ 来清除中断
LPCOMP_INTR[1]	COMP2	比较器 2 中断：硬件在比较器 1 触发时设置该中断。通过写入 ‘1’ 来清除中断
LPCOMP_INTR_SET[0]	COMP1	通过写入 ‘1’ 来触发比较器 0 的软件中断
LPCOMP_INTR_SET[1]	COMP2	通过写入 ‘1’ 来触发比较器 1 的软件中断

20.3.3 功耗模式和速度配置

低功耗比较器能够在以下三种功耗模式下运行：

- 快速模式
- 慢速模式
- 超低功耗模式

通过 LPCOMP_CONFIG 寄存器中的 MODE1 位 [1:0]，可以配置比较器 0 的功耗或速度模式。通过同一个寄存器中的 MODE2 位 [9:8]，可以配置比较器 1 的功耗或速度模式。功耗和响应时间会根据所选功耗模式而有所不同：在快速模式下，功耗最高，但响应速度最快；在超低功耗模式下，功耗最低，但响应速度最慢。请参见 器件数据手册，了解在各种功耗模式下响应时间和功耗的规范。

通过 LPCOMP_CONFIG 寄存器中的 ENABLE1 位 [7] 和 ENABLE2 位 [15]，可以使能 / 禁用比较器，如表 20-2 所示。

注意：当使能比较器时，如果更改功耗模式，比较器输出可能会发生短时脉冲。为了避免发生这种情况，请在更改功耗模式前禁用比较器。

表 20-2. 比较器功耗模式选择位 MODE1 和 MODE2

寄存器 [Bit_Pos]	Bit_Name	说明
LPCOMP_CONFIG[1:0]	MODE1	比较器 0 功耗模式选择位 00：慢速工作模式 （功耗较低） 01：快速工作模式 （功耗较高） 10：超低功耗工作模式 （功耗最低）
LPCOMP_CONFIG[9:8]	MODE2	比较器 1 功耗模式选择位 00：慢速工作模式 （功耗较低） 01：快速工作模式 （功耗较高） 10：超低功耗工作模式 （功耗最低）
LPCOMP_CONFIG[7]	ENABLE1	比较器 0 使能位 0：禁用比较器 0 1：使能比较器 0
LPCOMP_CONFIG[15]	ENABLE2	比较器 1 使能位 0：禁用比较器 1 1：使能比较器 1

20.3.4 迟滞

对于比较相邻信号或缓慢更改信号的应用，迟滞有助于避免信号嘈杂时在比较器输出上产生的振荡。对于这些应用，在比较器模块中将使能固定的 10 mV 迟滞。

通过配置 LPCOMP_CONFIG 寄存器中的 HYST1 位 [2] 和 HYST2 位 [10]，可以使能或禁用 10 mV 迟滞电平，如表 20-3 所示。

表 20-3. 迟滞控制位 HYST1 和 HYST2

寄存器 [Bit_Pos]	Bit_Name	说明
LPCOMP_CONFIG[2]	HYST1	使能 / 禁用比较器 0 的 10 mV 迟滞 - 0：使能迟滞 - 1：禁用迟滞
LPCOMP_CONFIG[10]	HYST2	使能 / 禁用比较器 1 的 10 mV 迟滞 - 0：使能迟滞 - 1：禁用迟滞

20.3.5 从低功耗模式唤醒

比较器可以在器件的低功耗模式 （睡眠、深度睡眠和休眠模式）下运行。比较器输出中断可将器件从睡眠、深度睡眠和休眠模式唤醒。 为了从低功耗模式中唤醒器件，必须通过 LPCOMP_CONFIG 寄存器使能比较器，而且不能将 LPCOMP_CONFIG 寄存器中的 INTTYPEx 位设置为禁用，另外需要置位 LPCOMP_INTR_MASK 寄存器中相对应的比较器的 INTR_MASKx 位。在深度睡眠和休眠模式下，不可用的性能包括：

- 使用 AMUXBUS 连接进行的比较
- 通过 DSI 路由比较器输出

在深度睡眠和休眠模式下，发生在比较器 0 或比较器 1 输出的比较事件均可生成唤醒中断。应该根据要求配置 LPCOMP_CONFIG 寄存器中的 INTTYPEx 位，以使相应的比较器将器件从低功耗模式唤醒。通过使用 LPCOMP_INTR_MASK 寄存器中的屏蔽位，可以选择 CPU 处理一个还是两个比较器中断。

20.3.6 比较器时钟

比较器将系统主时钟 SYSCLK 作为中断同步时钟使用。

20.3.7 偏移调整

比较器的偏移在出厂前被调整为 4.0 mV 以下。调整过程分两步：依次在通用模式下的电压为 0.1 V 和 V_{DD}–0.1 V 时进行调整。对于输入电压范围为 0.1 V 到 V_{DD}–0.1 V 的输入，偏移电压必须低于 10.0 mV。对于正常操作模式，不建议使用超过 10.0 mV 的偏移电压。

可以在特定的输入共模电压下准确调整偏移。通过配置 LPCOMP_TRIM1/2/3/4 寄存器，可以进行比较器偏移调整。LPCOMP_TRIM1 和 LPCOMP_TRIM2 用于调整比较器 0。LPCOMP_TRIM3 和 LPCOMP_TRIM4 用于调整比较器 1。LPCOMP_TRIM1 和 LPCOMP_TRIM3 的 TRIMA 位 [4:0] 以及 LPCOMP_TRIM2 和 LPCOMP_TRIM4 的 TRIMB 位 [3:0] 是用于更改调整值的位字段。 TRIMA 位用于粗调偏移，而 TRIMB 位用于微调偏移。只在比较器的慢速工作模式下才能使用 TRIMB 位来纠正偏移。

可以使用任何标准比较器偏移调整程序进行调整。通过使用以下方法可以提高参考 / 共模电压输入中的偏移电压。

1. 对比较器输入进行外部短接，并将参考电压 V_{ref} 连接到输入。
2. 设置比较器 （以便进行比较），关闭迟滞并检查输出。
3. 如果该输出为高电平，则偏移将为正向。否则，该偏移为负向。请按照以下步骤调整偏移：
   a. 调校 TRIMA 位 [4:0]，直到输出方向发生偏转为止。TRIMA 位 [3:0] 控制偏移量，TRIMA 位 [4] 则控制偏移的极性 （其中， ‘1’ 表示正偏移， ‘0’ 表示负偏移）。
   b. 当完成调校 TRIMA 位时，应一直调校 TRIMB 位 [3:0]，直到输出再次转换方向为止。TRIMB 位的调校只能在比较器的慢速模式下进行。 TRIMB 位 [3] 控制着偏移的极性。增加 TRIMB 位 [2:0] 会减少偏移。
   c. 完成 3-b 步骤后， TRIMA 和 TRIMB 位的值将为该特定 V_{ref} 最接近的调整值。

20.4 寄存器汇总

表 20-4. 低功耗比较器寄存器汇总

寄存器	功能
LPCOMP_ID	包含 LPCOMP 控制器 ID 及版本编号的信息
LPCOMP_CONFIG	LPCOMP 配置寄存器
LPCOMP_INTR	LPCOMP 中断寄存器
LPCOMP_INTR_SET	LPCOMP 中断设置寄存器
LPCOMP_INTR_MASK	LPCOMP 中断请求屏蔽寄存器
LPCOMP_INTR_MASKED	LPCOMP 屏蔽中断输出寄存器
LPCOMP_TRIM1	存储比较器 0 的调整字段
LPCOMP_TRIM2	存储比较器 0 的调整字段
LPCOMP_TRIM3	存储比较器 1 的调整字段
LPCOMP_TRIM4	存储比较器 1 的调整字段

21. 微型连续时间模块 （CTBm）

微型连续时间模块 （CTBm）在芯片内提供了离散运算放大器 （opamps），以用于连续时间信号链路。每个 CTBm 模块包括一个用于输入 / 输出配置的开关矩阵、两个相同的运算放大器 （也可以将它们配置为两个比较器）、每个运算放大器具有一个电荷泵，另外还包括用于比较器输出路由、开关控制和中断的数字接口。 PSoC 4100BL/4200BL 系列具有两个 CTBm 模块 — 四个离散运算放大器。另外，该 CTBm 模块还可以在深度睡眠的功耗模式下工作。

21.1 特性

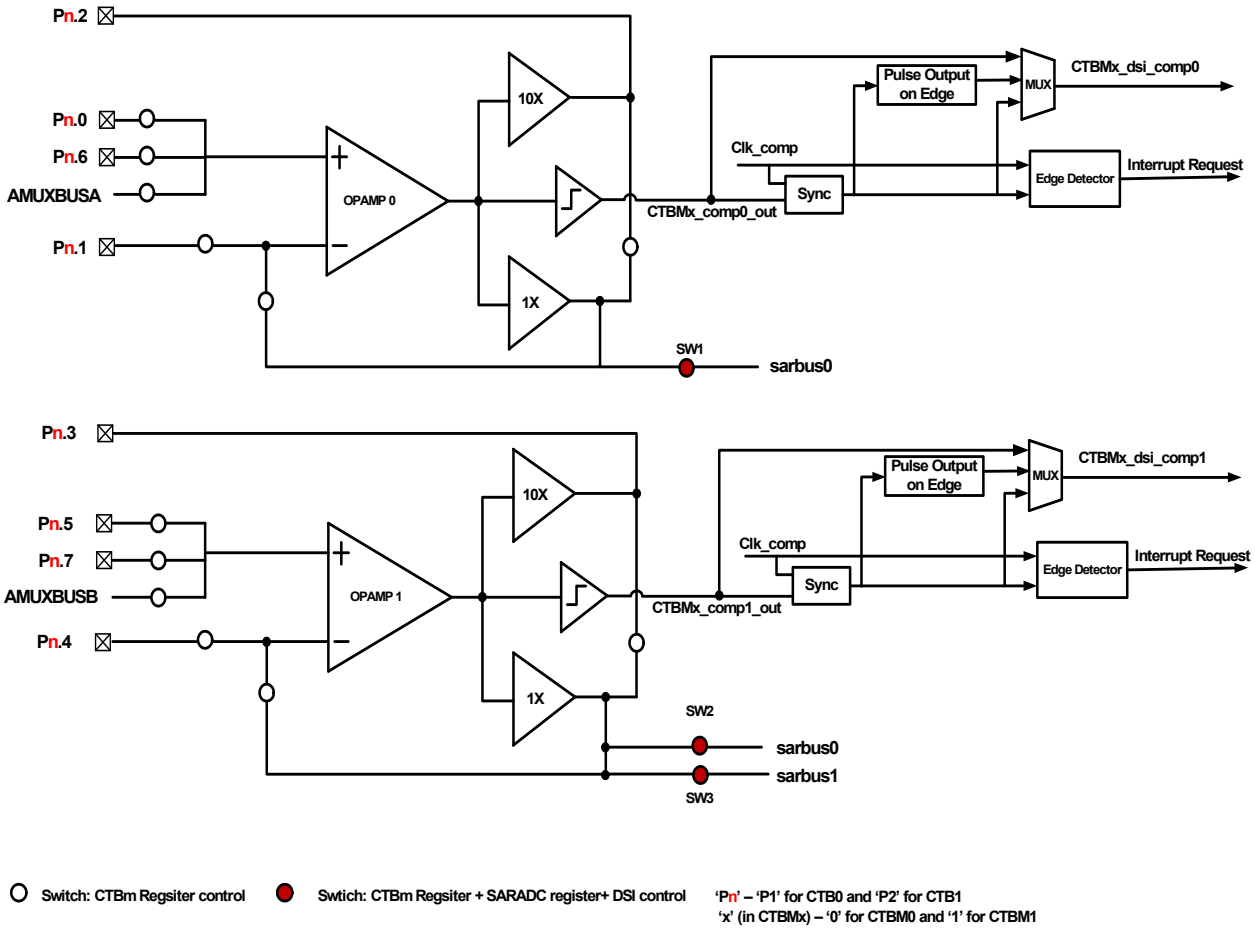
PSoC 4 CTBm 模块中的运算放大器具有以下特性：

- 离散、高性能并且高度可配置的片上放大器
- 可编程功耗、带宽、补偿和输出驱动强度
- 可选输出电流的驱动能力为 1 mA 或 10 mA
- 对于 20 pF 负载，增益带宽为 6 MHz
- 调整偏移低于 1 mV
- 支持运算放大器的 Follower 模式
- 比较器模式带有可选的 10 mV 迟滞
- SAR 输入的缓冲器 / 前置放大器
- 对于 1 mA 负载，轨至轨范围为 V_{SS} 至 V_{DDA}，误差为 ±0.2 V
- 对于 10 mA 负载，轨至轨范围为 V_{SS} 至 V_{DDA}，误差为 ±0.5 V
- 对于 50 pF 负载，转换速率为 4 V/µs
- 支持器件的深度睡眠模式

21.2 框图

图 21-1 显示的是 PSoC 4 器件中 CTBmx — CTMBm0 和 CTBm1 模块的框图。

图 21-1. CTBm 框图



21.3 工作原理

如框图所示，每个 CTBm 由两个相同的运算放大器和一个开关连接阵列组成。每个运算放大器包括一个输入级和三个输出级，它们共享了一个公用输入级（如图 21-1 中所示）；每次只能选中其中一个输出级。输出级可以作为 A 级（1X）、AB 级（10X）或比较器运行。另外，还可以配置其他特性，如：功耗与速度、补偿和开关连接控制。

为了使用 CTBm 模块，先设置外部组件，如电阻 （若需要）。然后，通过设置 CTBMx_CTB_CTRL [31] 位使能该模块。每个运算放大器中应有一个电荷泵，以获取轨至轨输入范围和最小的失真共模输入。通过在 CTBMx 中设置 opamp0 的 CTBMx-_OA_RES0_CTRL [11] 位和 opamp1 的 CTBMx_OA_RES1_CTRL [11] 位，可以使能相应的电荷泵。

使能运算放大器和电荷泵后，请按照下面的步骤设置放大器：

1. 配置功耗模式
2. 配置输出强度
3. 配置补偿
4. 配置输入开关

5. 配置输出开关，特别是在需要将运算放大器输出连接至 SAR ADC 的场合

请按照下面步骤设置比较器：

1. 配置功耗模式
2. 配置输入开关
3. 根据要求配置比较器输出电路、中断生成以及 DSI 输出等
4. 配置迟滞并使能比较器

21.3.1 功耗模式配置

运算放大器可以在下面三个功率模式下运行，即：低功耗、中等功耗以及高功耗。CTBm 可以通过调整运算放大器的参考电流来调整功耗。通过 CTBMx_OA_RESy_CTRL 内的 PWR_MODE 位 [1:0]，可以配置功耗模式。转换速率和增益带宽在高功耗模式下为最大值，而在低功耗模式下为最小值。请注意，功耗模式的配置也会影响到 1X 模式下的最大输出驱动能力（I_{OUT}）。更多详细信息，请参考表 21-1。欲了解各种功耗模式下的增益带宽、转换速率以及 I_{OUT} 的规范，请参阅 器件的数据手册 内容。

注意： ‘y’ 表示 CTBMx 中的 Opamp1/0 （y = 1 表示 Opamp1； y = 0 表示 Opamp0）。

21.3.2 输出驱动配置

可将每个运算放大器的输出驱动器配置为内部驱动器 （A 级 /1X 驱动器）或外部驱动器 （AB 级 /10X 驱动器）。1X 和 10X 驱动器是相互排斥的，即不能同时启用。1X 输出驱动器适合于以较高速度驱动较小的片上的电容性和电阻性负载。10X 输出驱动器则适用于驱动较大的片外的电容性和电阻性负载。1X 驱动器输出连接至 sarbus 0/1，10X 驱动器输出连接至外部引脚。每个驱动器都有低等、中等和高等三种功耗模式，如表 21-1 所示。

表 21-1. 输出驱动器与功耗模式

功耗模式 I_{OUT} 驱动能力	CTBMx_OA_RESy_CTRL[1:0]			
	00 （禁用）	01 （低功耗）	10 （中功耗）	11 （高功耗）
外部驱动器 （10X）	关闭	10 mA	10 mA	10 mA
内部驱动器 （1X）	关闭	100 mA	400 mA	1 mA

通过 CTBMx_OA_RESy_CTRL[2] 位，可以选择 10X 或 1X 的输出能力 （0: 1X， 1: 10X）。如果运算放大器的输出被连接到 SAR ADC，则建议选择 1X 输出驱动器。如果运算放大器的输出被连接到某个外部引脚，则建议选择 10X 输出驱动器。在特殊情况下，如果需要通过 1X 输出驱动器将输出连接至外部引脚，或通过 10X 输出驱动器将输出连接至内部负载 （例如， SAR ADC），可以将 CTBMx_OAy_SW [21] 置为 ‘1’。然而，赛普拉斯不能保证此情况下的性能。

表 21-2 总结了用于配置运算放大器输出驱动强度和功耗模式的各个位。

表 21-2. CTBM 寄存器中的输出强度和功耗模式配置

寄存器 [Bit_Pos]	位名称	描述
CTBMx_CTB_CTRL[31]	使能	CTBMx 功耗模式选择 0: CTBM 被禁用 1：CTBM 被使能
CTBMx_OA_RES0_CTRL [11]	OA0_PUMP_EN	Opamp0 泵使能位 0： Opamp0 泵在 CTBMx 中被禁用 1：Opamp0 泵在 CTBMx 中被使能
CTBMx_OA_RES1_CTRL [11]	OA1_PUMP_EN	Opamp1 泵使能位 0：Opamp1 泵在 CTBMx 中被禁用 1：Opamp1 泵在 CTBMx 中被使能
CTBMx_OA_RES0_CTRL [1:0]	OA0_PWR_MODE	Opamp0 功耗模式选择位 00：Opamp0 在 CTBMx 中被关闭 01：Opamp0 在 CTBMx 中为低功耗模式 10：Opamp0 在 CTBMx 中为中功耗模式 11：Opamp0 在 CTBMx 中为高功耗模式
CTBMx_OA_RES1_CTRL [1:0]	OA1_PWR_MODE	Opamp1 功耗模式选择位 00：Opamp1 在 CTBMx 中被关闭 01：Opamp1 在 CTBMx 中为低功耗模式 10：Opamp1 在 CTBMx 中为中功耗模式 11：Opamp1 在 CTBMx 中为高功耗模式
CTBMx_OA_RES0_CTRL [2]	OA0_DRIVE_STR_SEL	Opamp0 输出驱动强度选择位 0：Opamp0 在 CTBMx 中的输出驱动强度为 1X 1：Opamp0 在 CTBMx 中的输出驱动强度为 10X
CTBMx_OA_RES1_CTRL [2]	OA1_DRIVE_STR_SEL	Opamp1 输出驱动强度选择位 0：Opamp1 在 CTBMx 中的输出驱动强度为 1X 1：Opamp1 在 CTBMx 中的输出驱动强度为 10X

21.3.3 补偿

每个运算放大器还有一个可编程补偿的电容模块，允许根据输出负载来优化运算放大器性能的稳定性。通过相应的 CTBMx-_OAy_COMP_TRIM 寄存器，可以控制每个运算放大器的补偿，如在表 21-3 中所示。请注意，器件数据手册中的所有增益带宽和转换速率规范可用于所有补偿调整。

表 21-3. CTBMx （CTBM0 和 CTBM1）中的 Opampy （Opamp0 或 Opamp1）补偿位

寄存器 [Bit_Pos]	位名称	描述
CTBMx_OAy_COMP_TRIM[1:0]	OAy_COMP_TRIM	Opampy 补偿调整位 00：无补偿 01：CTBMx 中有最小补偿，高速度和低稳定性 10：CTBMx 中有中度补偿，平衡的速度和稳定性 11：CTBMx 中有最大补偿，低速度和高稳定性

21.3.4 开关控制

每个 CTBm 均有许多用于配置运算放大器的输入和输出的开关。通过配置 CTBm 寄存器（CTBMx_OA0_SW、CTBMx_OA1_SW），可以控制几乎所有开关，除了三个用于将运算放大器输出通过 sarbus0 和 sarbus1 连接至 SAR ADC 的开关以外。需要通过 SAR ADC 寄存器、 CTBm 寄存器和 DSI 信号来控制这三个开关。

通过设置 CTBMx_OAy_SW 寄存器中的相应位，可以关闭开关；而清除这些位时会使相应开关被打开。向 CTBMx_OAy_SW_CLEAR 写入 ‘1’ 时，可以清除 CTBMx_OAy_SW 中的相应位。更多有关这些开关和它们所使能的连接的信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册。

21.3.4.1 输入配置

模拟开关为运算放大器的正和负输入选择提供了一些选项。这些开关会连接至运算算放大器输入 （来自外部引脚或 AMUX 总线），或构成一个局部反馈回路 （用于缓冲功能）。每个运算放大器上的开关都将连接两个 AMUXBUS 线中的一个：Opamp0 和 Opamp1 分别连接至每个 CTBm 模块中的 AMUXBUS-A 和 AMUXBUS-B。

注意：要确保正输入和负输入路径中只有一个开关被关闭；否则，不同的输入源将被互相短接。

- 正输入：模拟开关为每个 CTBm 的 opamp0 和 opamp1 提供了三个正输入选项，分别为两个外部引脚和一个 AMUXBUS 线。更多有关信息，请参考表 21-4。

表 21-4. 正输入选择

	正输入	开关控制位	描述
Opamp0	AMUXBUSA	CTBMx_OA0_SW [0]	0：开断； 1：闭合
	Pn.0[a]	CTBMx_OA0_SW [2]	0：开断； 1：闭合
	Pn.6	CTBMx_OA0_SW [3]	0：开断； 1：闭合
Opamp1	AMUXBUSB	CTBMx_OA1_SW [0]	0：开断； 1：闭合
	Pn. 5	CTBMx_OA1_SW [1]	0：开断； 1：闭合
	Pn.7	CTBMx_OA1_SW [4]	0：开断； 1：闭合

a. Pn = P1 （对于 CTBm0）以及 P25 （对于 CTBm1）

- 负输入：模拟开关为每个 CTBm 的 opamp0 和 opamp1 提供了两个负输入选项，即为一个外部引脚或输出反馈 （具体由 CTBMx_OAy_SW 寄存器控制）。表 21-5 显示的是控制位的详细信息。

表 21-5. 负输入选择

	负输入	开关控制位	描述
Opamp0	Pn.1[a]	CTBMx_OA0_SW [8]	0：开断； 1：闭合
	Opamp0 输出反馈经过 1X 输出驱动器	CTBMx_OA0_SW [14]	0：开断； 1：闭合
Opamp1	Pn.4	CTBMx_OA1_SW [8]	0：开断； 1：闭合
	Opamp1 输出反馈经过 1X 输出驱动器	CTBMx_OA1_SW [14]	0：开断； 1：闭合

a. Pn = P1 （对于 CTBm0）以及 P25 （对于 CTBm1）

21.3.4.2 输出配置

每个运算放大器的输出被直接连接到一个固定引脚上；不要求其它设置。通过三个开关 （SW1/2/3） ，可以选择将它连接至 sarbus0 或 sarbus1。每个 CTBm 的 Opamp0 输出可以连接至 sarbus0，而 opamp1 输出可以连接至 sarbus0 或 sarbus1。通过 sarbus0 和 sarbus1 可以将运算放大器输出连接至 SAR ADC 输入复用器。连接至 sarbus 的这三个输出路由开关由 SAR ADC 寄存器、 CTBm 寄存器和 DSI 信号一起控制；其他开关则仅由 CTBm 寄存器控制。

下面的真值表（、和）显示了三个开关的控制逻辑。PORT_ADDR、PIN_ADDR 和 DIFFERENTIAL_EN 分别源于 SAR_CHANx-_CONFIG [6:4]、SAR_CHANx_CONFIG [2:0] 和 SAR_CHANx_CONFIG [2:0]。PORT_ADDR =0 或 PIN_ADDR = 0 都会导致 SW[n] = 0。使用 SAR 寄存器或 DSI 信号来控制开关时，需要设置 CTBMx_SW_HW_CTRL 位 [2] 或 [3]。如果 CTBMx-_OAy_SW[18]/[19] = 0 并且 SW[n] = 0， CTBMx_OAy_SW[18]/[19] 可以屏蔽其他控制位。

CTBMx__SW_STATUS [30:28] 寄存器显示了 SW1/2/3 的当前开关状态。

表 21-6. SW1 控制逻辑的真值表

PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[2]	dsi_out[2]	CTBMx_OA0_SW[18]	SW1
X	X	X	X	0	0
X	0	1	0	1	0
0	X	1	0	1	0
X	X	X	1	1	1
X	X	0	X	1	1
1	2	X	X	1	1

表 21-7. SW2 控制逻辑的真值表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[3]	dsi_out[3]	CTBMx_OA0_SW[18]	SW2
X	X	X	X	X	0	0
X	X	0	1	0	1	0
X	0	X	1	0	1	0
1	X	X	X	0	1	0
X	X	X	0	X	1	1
X	X	X	X	X	1	1
0	1	3	X	X	1	1

表 21-8. SW3 控制逻辑的真值表

DIFFERENTIAL_EN	PORT_ADDR	PIN_ADDR	CTBMx_SW_HW_CTRL[3]	dsi_out[3]	CTBMx_OA0_SW[18]	SW3
X	X	X	X	X	0	0
X	X	0	1	0	1	0
X	0	X	1	0	1	0
0	X	X	X	0	1	0
X	X	X	0	X	1	1
X	X1	X	X	X	1	1
1	1	2	X	X	1	1

21.3.4.3 比较器模式

通过置位相应的 CTBMx_OA_RESy_CTRL[4] 位，可以将每个运算放大器配置为比较器。请注意，使能比较器后会完全禁用补偿电容，并会关闭 A 级 （1X）和 AB 级 （10X）输出驱动器。该比较器具有以下特性：

- 可选的 10 mV 输入迟滞
- 可配置的功耗 / 速度
- 可选的 DSI 输出同步
- 偏移被调整到低于 1 mV 的值
- 可配置的边沿检测 （上升沿 / 下降沿 / 两者 / 禁用）

21.3.4.4 比较器配置

可以单向 （从低到高）使能 ‘10 mV ±5%’ 迟滞。通过设置 CTBMx_OA_RESy_CTRL[5]，可以使能输入迟滞。通过设置 CTBMx_OA_RESy_CTRL [1:0]，可以将每个 CTBm 模块中的两个比较器配置为低、中等和高等三种功耗模式。根据不同的功耗模式，响应时间和功耗也不一样；快速模式下的功耗最高，而超低功耗模式下的功耗最低。本数据手册中提供了有关功耗和响应时间的详细规范。

比较器输出可以同步连接至 DSI。通过 CTBMx_CTBM_OA_RESxy_CTRL[6]，可以配置比较器时钟（系统 AHB 时钟）的同步。

比较器 0 和比较器 1 的输出状态分别被存储于 CTBMx_COMP_STAT[0] 和 CTBMx_COMP_STAT[16] 中。

表 21-9 总结了用于配置 CTBM 模块中的比较器模式的各个位。

表 21-9. 比较器模式和配置寄存器的设置

寄存器 [位 _ 位置]	位名称	描述
CTBMx_OA_RESyy_CTRL[4]	OAy_COMP_EN	Opampy 比较器使能位 0：CTBMx 中 opampy 内的比较器模式被禁用 1：CTBMx 中 opampy 内的比较器模式被使能
CTBMx_OA_RESy_CTRL[5]	OAy_HYST_EN	Opampy 比较器迟滞使能位 0：CTBMx 中 opampy 内的迟滞被禁用 1：CTBMx 中 opampy 内的迟滞被使能
CTBMx_OA_RESy_CTRL[6]	OAy_BYPASS_DSI_SYNC	Opampy 旁路比较器输出与 DSI （触发器）输出的同步 0：同步 （电平或脉冲） 1：旁路
CTBMx_OA_RESy_CTRL[7]	OAy_ DSI_LEVEL	Opampy 比较器 DSI （触发器）输出同步电平 0：脉冲 1：电平

21.3.4.5 比较器中断

将比较器输出连接至边沿检测器模块，通过该模块可以检测生成中断的边沿 （禁用 / 上升沿 / 下降沿 / 双边沿）。通过使用 CTBMx_OA_RESy_CTRL[9:8] 位，可以配置该中断。

每个比较器均有独立的 IRQ。 CTBMx_INTR [0] 用于 comparator0 IRQ， CTBMx_INTR [1] 用于 comparator1 IRQ。虽然各个 CTBM 中的每个比较器具有不同的 IRQ 位，但它们都共享了被映射到 CPU NVIC 中的单个 CTBM ISR。有关详细信息，请参考第 57 页上的中断章节中介绍的内容。通过轮询 CTBMx_INTR 位，您可以检查到底是 CTBM 中的哪个比较器触发了该 ISR。

每个中断具有一个 CTBMx_INTR_MASK 寄存器中的中断屏蔽位。通过将中断屏蔽位设置为低电平，可以忽略相应的中断源。如果CTBMx_INTR寄存器中的中断标志和CTBMx_INTR_MASK寄存器中的相应中断屏蔽进行逻辑AND得到的结果为 ‘1’，将生成对 NVIC 的 CTBm 比较器中断。

向 CTBMx_INTR 位 [1:0] 写入 ‘1’，可以清除相应中断。

为方便固件，还可以在 CTBMx_INTR_MASKED 寄存器中对中断标志和中断屏蔽进行交叉 （逻辑 AND）。

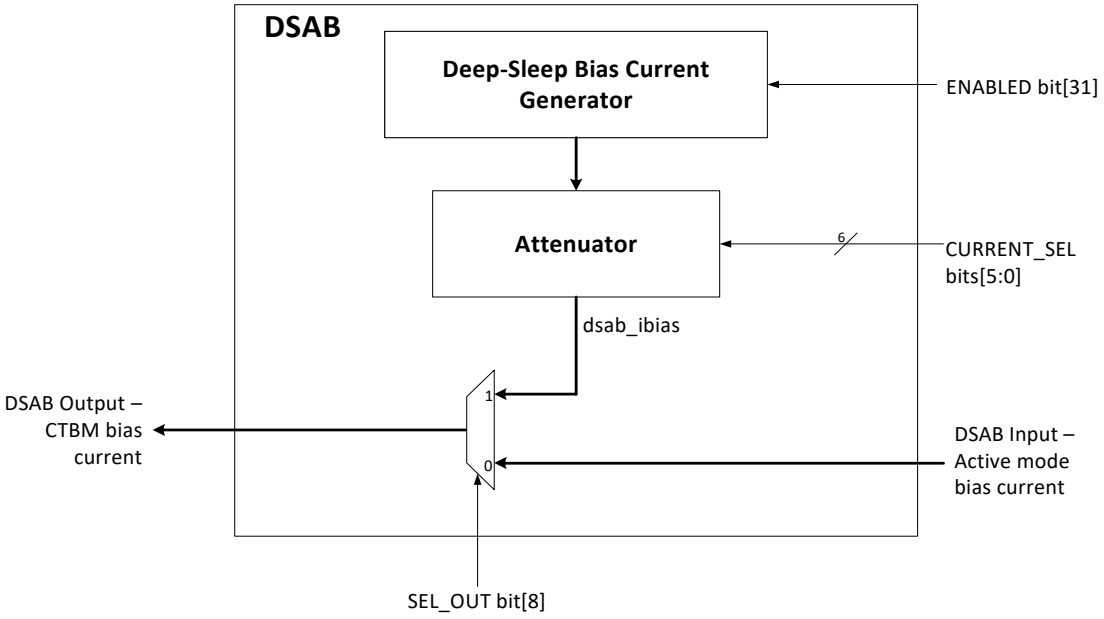
为了便于进行验证和调试， CTBMx_INTR_SET 寄存器为每个中断提供了一个设置位。这样，固件不需要真正的比较器开关事件仍能生成中断。

21.3.4.6 深度睡眠模式操作

在深度睡眠模式下，提供偏置电流、参考电压和 IMO 时钟的模块被关闭。因此，CTBm 功能（它的操作取决于偏置电流和 IMO 时钟）也随之无效。更多有关各种功耗模式和它们可用模块的信息，请参考第 101 页上的功耗模式章节。为了在深度睡眠模式下支持 CTBm 功能，需要通过一个被称为深度睡眠放大器偏置（DSAB）的特殊模块来生成交流偏置电流。这样， CTBm 中的运算放大器能够在深度睡眠模式下工作。

图 21-2 显示的是 DSAB 模块的架构。该模块将活动模式偏置电流作为输入电流接收。它会输出偏置电流，该电流被输入到运算放大器偏置电路。在活动模式下，DSAB 模块作为一个通过模块运行，并将偏置电流从输入路由至输出。在深度睡眠模式下，如果被使能， DSAB 将生成交流偏置电流，并且输出被衰减为一个用户指定的值，然后为该输出上的 CTBm 模块提供偏置电流。如果禁用了 DSAB 模块，输出会始终连接至输入偏置电流，而且在深度睡眠模式下不会生成交流偏置电流。DSAB 模块被禁用时，运算放大器不会在深度睡眠模式下工作。 PASS_DSAB_DSAB_CTRL 寄存器中的 ENABLED 位 [31] 用于使能 / 禁用该模块；而 CURRENT_SEL 位 [5:0] 用于选择输出偏置电流的值。该值被选为 CURRENT_SEL × 0.075 uA （±5%）。可以通过 SEL_OUT 位 [8] 选择可路由至 CTBm 偏置的两个偏置电流的其中一个。表 21-10 总结了 PASS_DSAB_DSAB_CTRL 寄存器的位配置设置情况。

图 21-2. 深度睡眠模式下的放大器偏置框图



该特性对要求根据运算放大器电路进行的设计非常有用，因此可以在低功耗模式下 （如深度睡眠模式）保持活动状态，从而节省电源。例如，对于要求运算放大器始终处于使能状态的电池运行系统（如心率监视器），如果芯片其余部分可以进入深度睡眠模式并且只在需要时才会唤醒的话，则可以节省大量的功耗。请注意，DSAB 模块提供的偏置电流不能满足活动模式偏置电流的准确度和稳定性。另外，DSAB 模块不会生成替代时钟。因此，开关以及与运算放大器相关的电荷泵都不可用。因此，运算放大器的最高输入共模电压被限于 V_{DDA} – 1.3 V。另外，由于开关泵不可用 （在电压低于 3.3 V 条件下运行时需要使用它们来实现模拟开关），当电源电压下降至低于 3.3 V 水平时，模拟开关的导通电阻会超过正常规范。信号速度低时，模拟开关需要使用更高的导通电阻值。所以，模拟开关的阻力变得过高前， V_{DDA} 可以降低至 ~ 2.8 V。这是电源电压的最低值。但是，在深度睡眠模式下使用运算放大器时，建议将 V_{DDA} 设置为 3.3 V 或更高的值。有关深度睡眠模式下的运算放大器规范的详细信息，请参考器件数据手册。

要想在深度睡眠模式下使能运算放大器，需要设置 CTBMx-_CTBM_CTB_CTRL 寄存器的 DEEPSLEEP_ON 位 [30]。这样便可以在深度睡眠模式下使能 CTBMx 的两个运算放大器。DSAB 模块被使能后， CTBm 才能在深度睡眠模式下运行。

表 21-10. DSAB 和 CTBM 深度睡眠配置寄存器设置

寄存器 [位 _ 位置]	位名称	描述
PASS_DSAB_DSAB_CTRL [5:0]	CURRENT_SEL	dsab_ibias 的电流选择； dsab_ibias = CURRENT_SEL × 0.075 μA （±5%）
PASS_DSAB_DSAB_CTRL [8]	SEL_OUT	CTBm 的偏置电流选择 0：旁路 DSAB，并使用活动模式中的偏置电流 1：将 dsab_ibias 作为 CTBm 偏置电流使用
PASS_DSAB_DSAB_CTRL [31]	ENABLED	使能 / 禁用 DSAB 偏置发生器 0：DSAB 模块被禁用，并且 CTBm 偏置电流被连接到活动模式中的偏置电流 1：DSAB 模块被使能，并且 CTBm 偏置电流是由 SEL_OUT 信号控制的
CTBMx_CTBM_CTB_CTRL [30]	DEEPSLEEP_ON	使能 / 禁用深度睡眠模式中的 CTBMx 功能 0：使能 1：禁用

21.4 寄存器汇总

表 21-11. 寄存器总结

名称	描述
CTBMx_CTRL	全局 CTBm 模块使能
CTBMx_OA_RES0_CTRL	Opamp0 控制寄存器
CTBMx_OA_RES1_CTRL	Opamp1 控制寄存器
CTBMx_COMP_STAT	比较器状态
CTBMx_INTR	中断请求寄存器
CTBMx_INTR_SET	中断请求设置寄存器
CTBMx_INTR_MASK	中断请求屏蔽
CTBMx_INTR_MASKED	中断请求被屏蔽
CTBMx_OA0_SW	Opamp0 开关控制
CTBMx_OA0_SW_CLEAR	Opamp0 开关控制清除
CTBMx_OA1_SW	Opamp1 开关控制
CTBMx_OA1_SW_CLEAR	Opamp1 开关控制清除
CTBMx_SW_HW_CTRL	CTBm 硬件控制使能
CTBMx_SW_STATUS	CTBm 总线开关控制状态
CTBMx_OA0_OFFSET_TRIM	Opamp0 调整控制
CTBMx_OA0_SLOPE_OFFSET_TRIM	Opamp0 调整控制
CTBMx_OA0_COMP_TRIM	Opamp0 调整控制
CTBMx_OA1_OFFSET_TRIM	Opamp1 调整控制
CTBMx_OA1_SLOPE_OFFSET_TRIM	Opamp1 调整控制
CTBMx_OA1_COMP_TRIM	Opamp1 调整控制
PASS_DSAB_DSAB_CTRL	DSAB 控制寄存器
PASS_DSAB_TRIM	IBIAS 调整寄存器

22. LCD 直接驱动

PSoC® 4 液晶显示屏（LCD）驱动系统是一种可灵活配置的外设，通过它可以使 PSoC 器件直接驱动 STN 及 TN segment LCD。

22.1 特性

PSoC 4 LCD segment 驱动模块具有以下特性：

- 支持多达 32 个 segment 及 4 个 common
- 支持 A 型 （标准）和 B 型 （低功耗）驱动波形
- 可将任意一个 GPIO 配置为 common 或 segment
- 支持以下 5 种驱动模式：
  - 数字相关模式
  - 1/2 偏压下的 PWM 驱动模式
  - 1/3 偏压下的 PWM 驱动模式
  - 1/4 偏压下的 PWM 驱动模式
  - 1/5 偏压下的 PWM 驱动模式
- 在数字相关模式下，能够使用 1.8 V 的 V_{DD} 来驱动 3 V 的显示屏
- 能够在活动、睡眠及深度睡眠模式下工作
- 支持数字对比度控制

22.2 LCD segment 驱动概述

一个分段的 LCD 显示屏由以下部分构成：悬浮于两组电极之间的液晶材料、多个偏振和反射层。这两组电极分别称为共模信号（COM）或背板和段式电极 （SEG）。从电气角度来看，可将一个 LCD segment 视为电容负载。可将 COM/SEG 电极视为 segment 矩阵中的行和列。通过改变对应的 COM/SEG 对的均方根 （RMS）电压，可以控制 LCD segment 的不透明度。

本章节中使用了以下各术语 / 电压来介绍 LCD 驱动：

- **V_{RMSOFF}**：表示 LCD 驱动器对准备关闭的各 segment 可执行的电压值。
- **V_{RMSON}**：表示 LCD 驱动器对打算打开的各 segment 可执行的电压值。
- **识别率（D）**：表示 LCD 驱动器可执行的 V_{RMSON} 及 V_{RMSOFF} 的比率。该比率取决于应用于 LCD 屏幕的波形类型。识别率越高，则对比度越高。

液晶材料不能长期在直流电压下工作。因此，所有应用于屏幕的波形都必须在各 segment（打开或关闭）上产生 0 V 的直接电流。通常，LCD 驱动器将波形应用于（通过多种电压之间的切换生成的）COM 和 SEG 电极。文档使用下列术语定义这些波形：

- **占空比**：当一个驱动器驱动了 ‘M’ 个 COM 电极时，便可以确定它运行时的占空比为 1/M。在 1/M 的时间内，各个 COM 电极被有效驱动。
- **偏压**：当驱动器的波形使用 （1/B）× VDRV 的电压阶跃时，则可以确定它的偏压为 1/B。 VDRV 是系统中最高的驱动电压（等于 PSoC 4 中的 V_{DD} 电压）。在 PWM 驱动模式下， PSoC 4 支持 1/2、 1/3、 1/4 以及 1/5 的偏压。
- **帧**：一帧表示驱动所有 segment 所需的时间。执行一个帧期间，驱动器通过 common 信号顺序循环。测量整个帧时，所有 segment 将接收 0 V 直流 （非零的 RMS 电压）。

在所有驱动模式下， PSoC 4 均支持两种不同的驱动波形类型，包括：

- **A 类波形**：在这种波形中，驱动器将一帧分成 M 个子帧。‘M’ 为 COM 电极的数量。在执行一个帧期间，仅对各个 COM 标记一次。例如， COM[i] 会在子帧 i 中被标记。
- **B 类波形**：驱动器将一帧分成 2M 个子帧。这两个子帧是互为相反的。在执行一个帧期间，对各个 COM 标记两次。例如，COM[i] 会在子帧 i 及 M+i 中被标记。由于 B 类波形的每一帧中都包含少量的转换，所以它的节能特性特别显著。

22.2.1 驱动模式

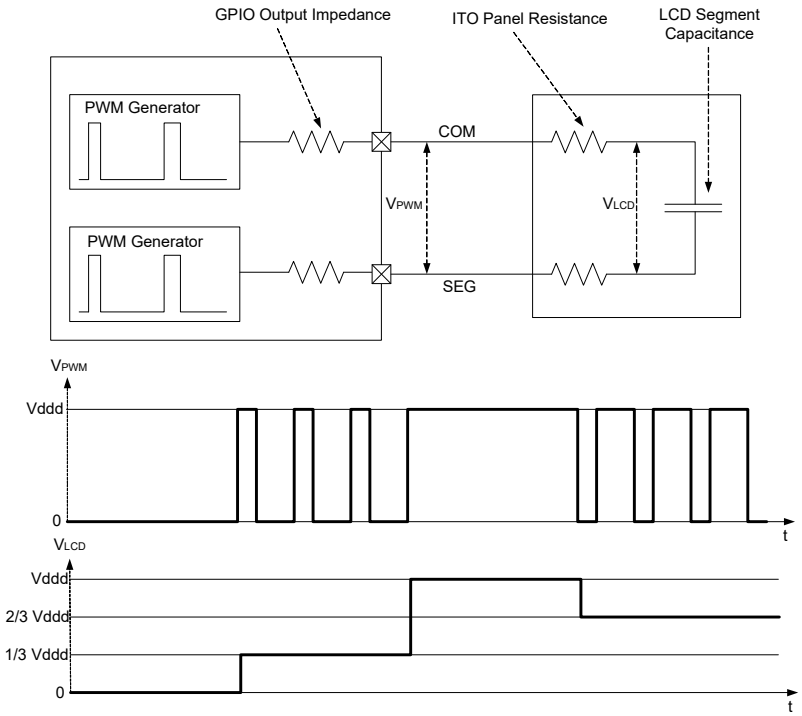
PSoC 4 支持以下驱动模式。

- 1/2 偏压下的 PWM 驱动模式
- 1/3 偏压下的 PWM 驱动模式
- 1/4 偏压下的 PWM 驱动模式 （在时钟输入信号为高频的条件下）
- 1/5 偏压下的 PWM 驱动模式 （在时钟输入信号为高频的条件下）
- 数字相关模式

22.2.1.1 PWM 驱动模式

在 PWM 驱动模式下，通过使用 PWM 输出信号以及 LCD 显示屏的内置电阻和电容来生成多个电压驱动信号，如图 22-1 所述。

图 22-1. PWM 在 1/3 偏压下驱动



驱动电气的输出波形是一个 PWM 波形。通过使用铟锡氧化物 （ITO）屏幕电阻和 segment 电容来过滤 PWM， LCD segment 的电压是模拟电压，如图 22-1 所示。该图显示了在 1/3 偏压下波形的生成过程 （四个 common，电压阶跃为 1/3 V_{DD}）。

PWM 是由 ILO（频率为 32 kHz、低速操作）或 IMO（高速操作）提供时钟。所生成的模拟电压通常在超低频率下（~ 50 Hz）驱动 segment LCD。

图 22-2 及图 22-3 分别说明了在 1/2 偏压及 1/4 占空比下 COM 和 SEG 电极的 A 类和 B 类波形。其中只绘制了 COM0/COM1 及SEG0/SEG1，用于演示。同样，图22-4及图22-5分别说明了在1/3偏压及1/4占空比下的COM和SEG电极的A类和B类波形。

图 22-2. PWM1/2 的 A 类波形示例

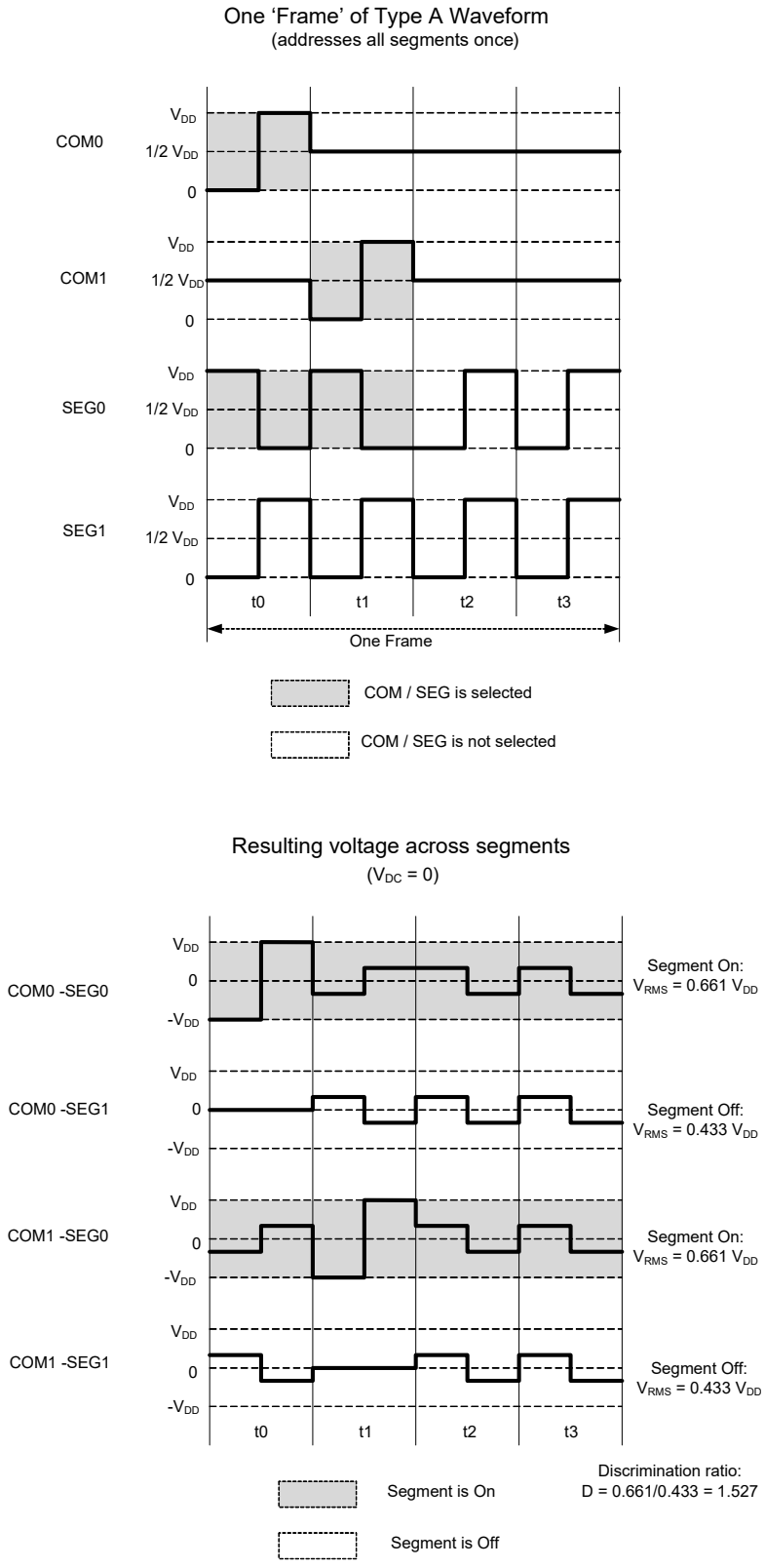


图 22-3. PWM1/2 的 B 类波形示例

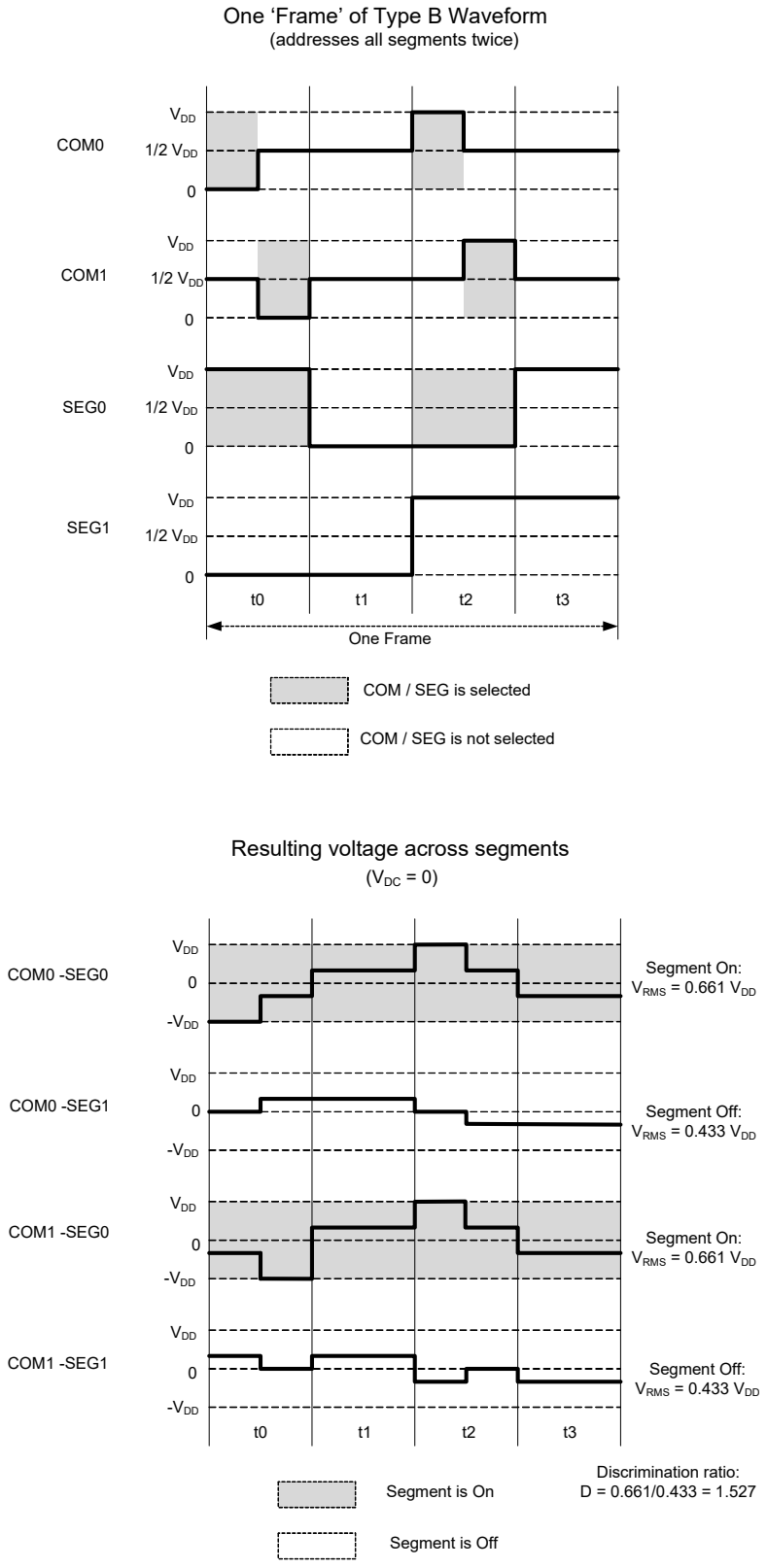


图 22-4. PWM1/3 的 A 类波形示例

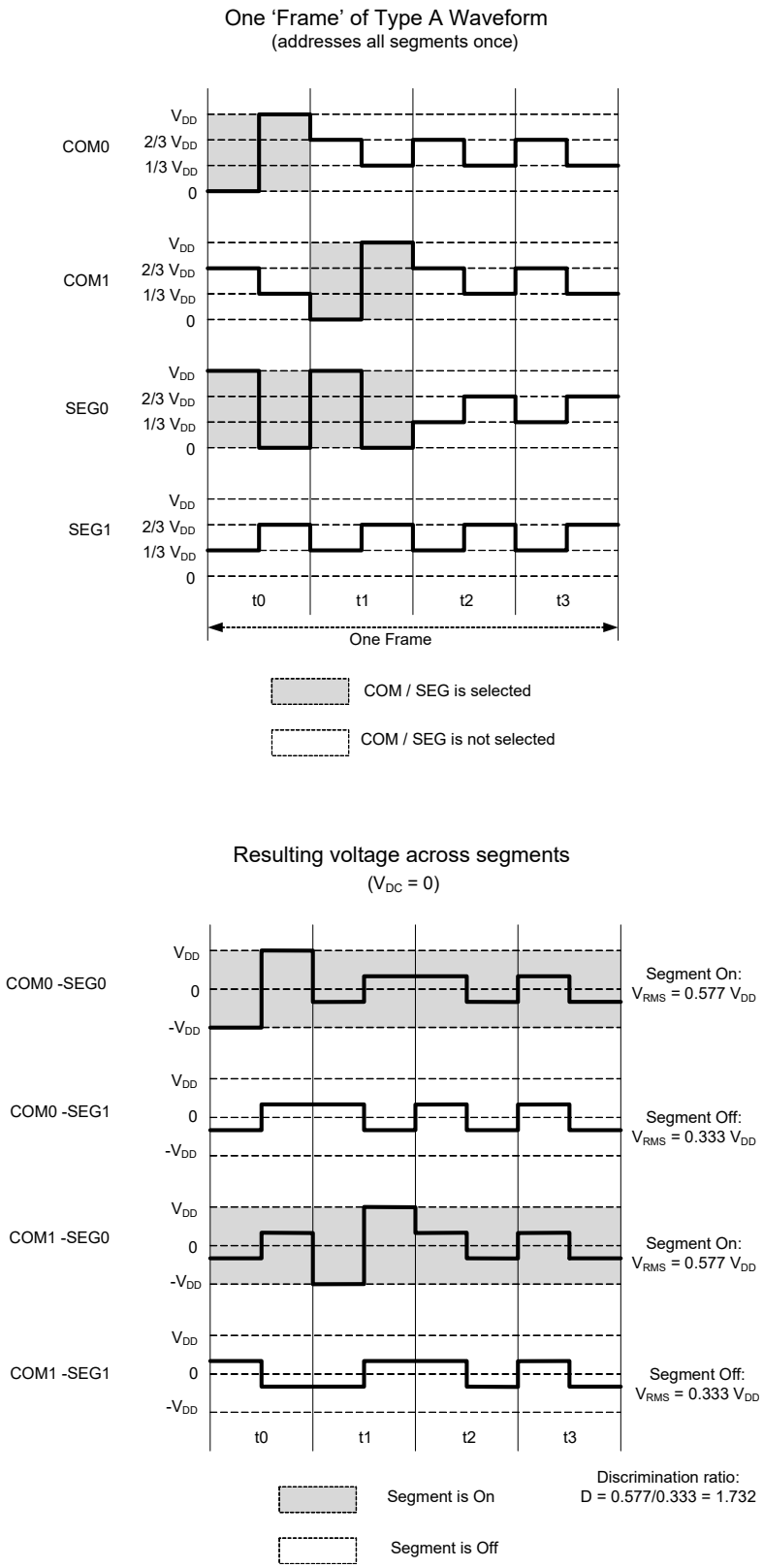
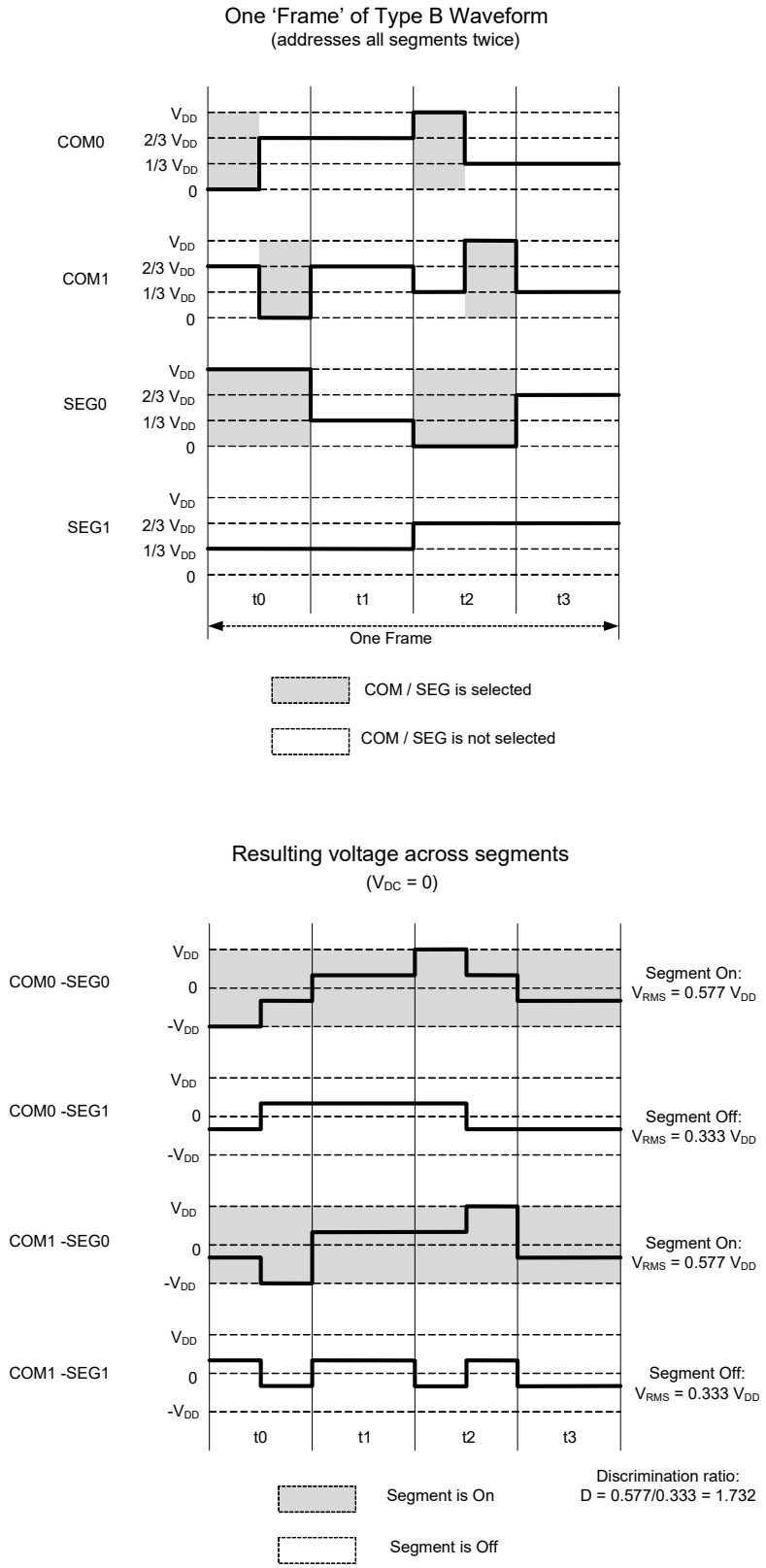


图 22-5. PWM1/3 的 B 类波形示例



通过下列公式，可以容易计算出打开 （ON）或关闭 （OFF） segment 的有效 RMS 电压：

$$V_{RMS(OFF)} = \sqrt{\frac{2(B-2)^2 + 2(M-1)}{2M}} x\left(\frac{V_{DRV}}{B}\right)$$

公式 22-1

$$V_{RMS(ON)} = \sqrt{\frac{2B^2 + 2(M-1)}{2M}} x\left(\frac{V_{DRV}}{B}\right)$$

公式 22-2

其中， B 是偏压，M 是占空比 （即为 COM 的数量）。

例如，如果 COM 的数量是 4，那么在 1/2 及 1/3 偏压下的识别率 （D）分别为 1.528 及 1.732。如果 COM 的数量是 2 和 3，则 1/3 偏压所提供的识别率会更好。因此，与 1/2 偏压相比， 1/3 偏压提供了更高的对比度，并建议将其使用于大多数应用中。1/4 和 1/5 偏压仅适用于 LCD 的高速操作。特别是使用率高的 COM 设计 （多于 4 个 COM）时，它们会提供更好的识别率。

当使用 LCD 的低速操作时， PWM 信号会由 32 kHz ILO 派生得到。为了使用 32 kHz PWM 驱动具有可接受波纹及上升 / 下降时间的低电容显示屏，需要使用 100 k-1 MΩ 的附加外部串联电阻。频率大于 ~1 MHz 的 PWM 无需外部电阻。理想的 PWM 频率取决于显示屏的电容及 ITO 路由线的内部 ITO 电阻。

1/2 偏压模式只需要对 COM 信号进行脉冲宽度调制，这也是它的优势，而 SEG 信号则需要使用逻辑电平，如图 22-2 和图 22-3 所示。

22.2.1.2 数字相关模式

数字相关模式并非在轨与轨之间生成偏压，而是利用 LCD 显示屏的特性。 LCD segment 的对比度由各 segment 的 RMS （均方根）电压决定。使用这种方法，可根据任意给定的 COM 和 SEG 信号对间的相关系数来确定对应的 LCD segment 处于打开状态还是关闭状态。因此，通过将 COM 信号在其非活动子帧间隔中的基准驱动频率翻倍， COM 和 SEG 驱动信号的相位关系可发生变化，以打开和关闭各 segment。这种情况与改变信号的直流电平的 PWM 驱动方法不同。图 22-8 及图 22-9 显示的是描述操作原则的波形示例。

图 22-6. 数字相关 A 类波形

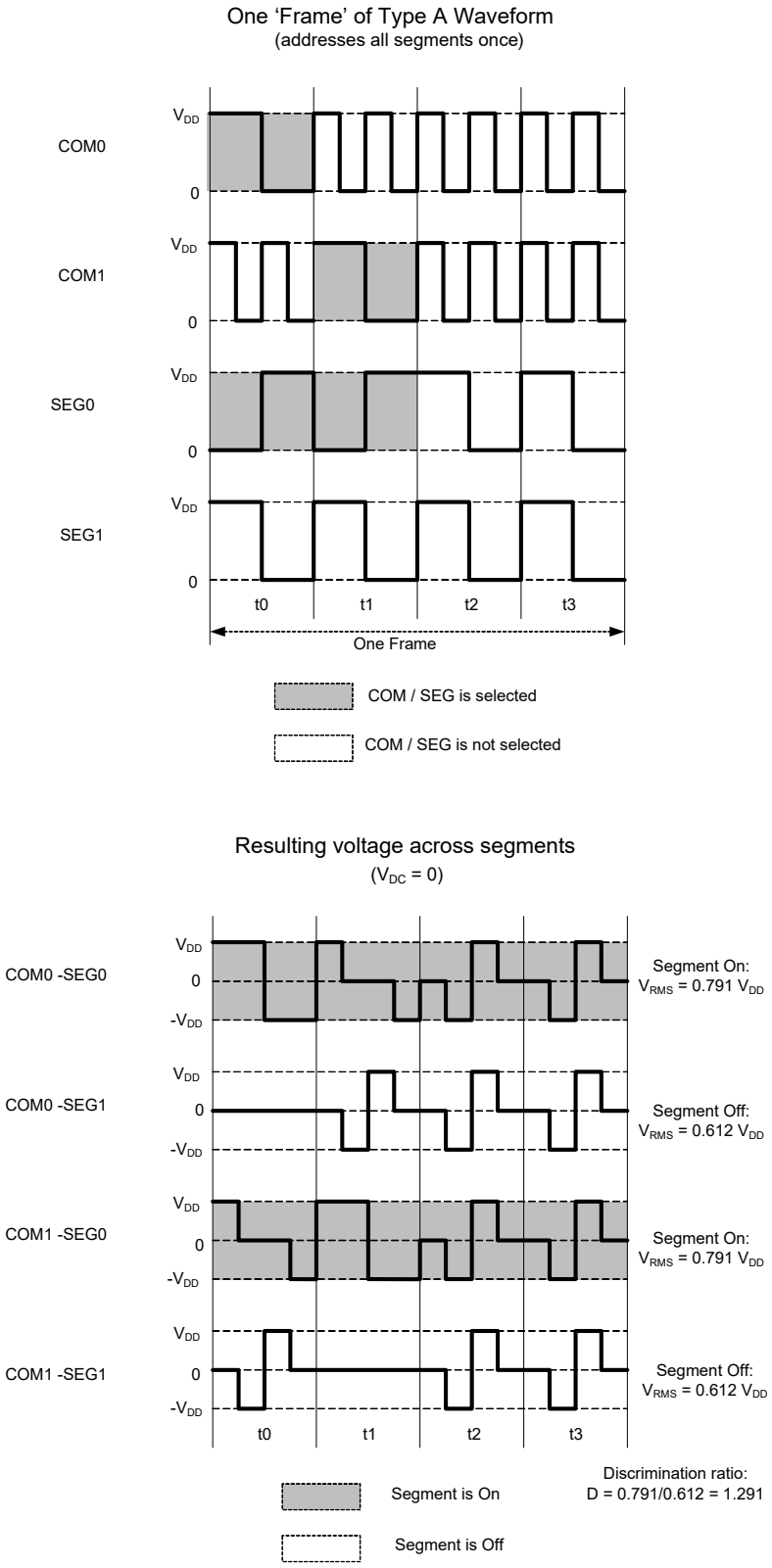
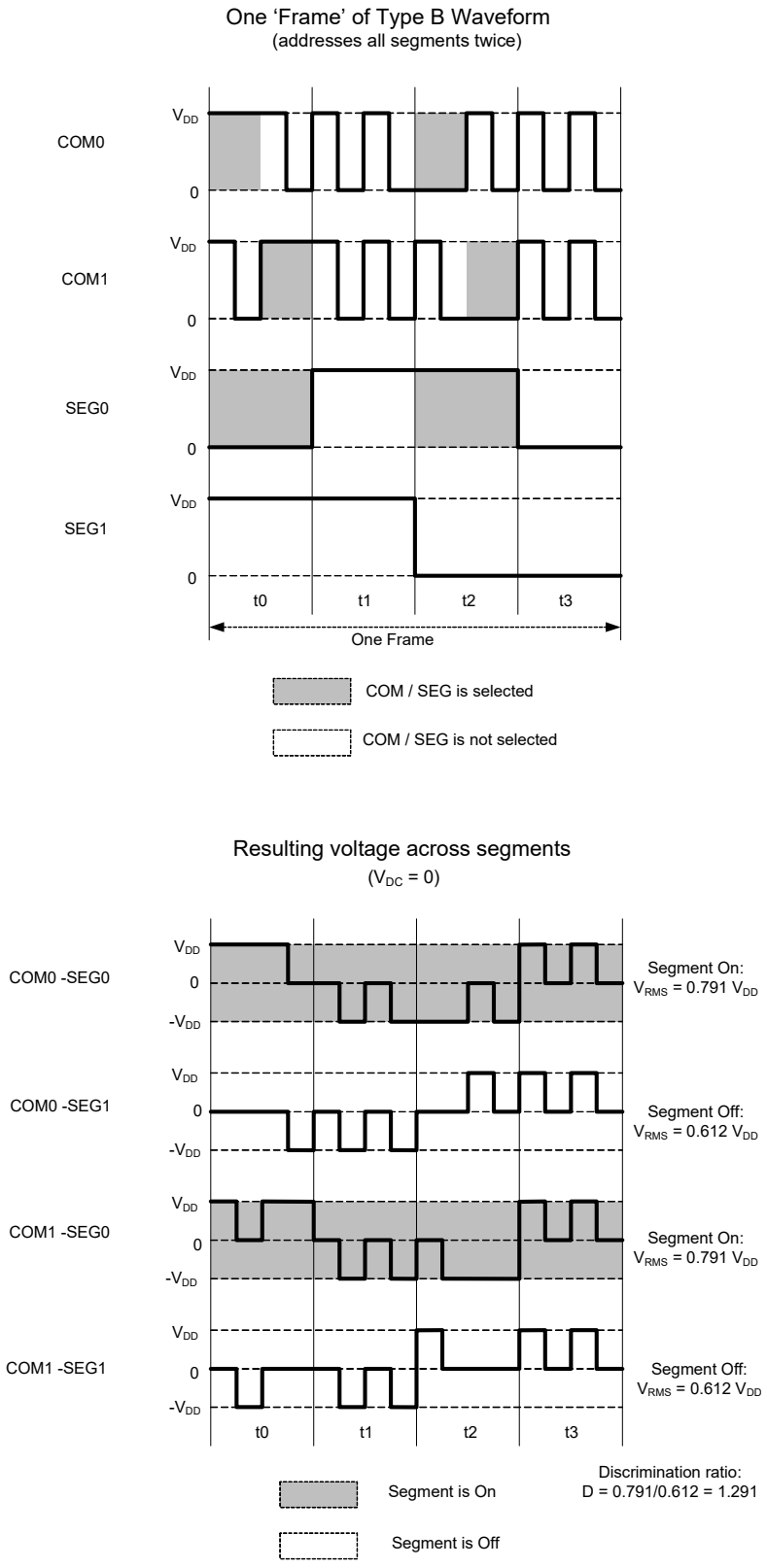


图 22-7. 数字相关 B 类波形



可使用以下公式计算出用于打开或关闭 segment 的 RMS 电压：

$$V_{RMS(OFF)} = \sqrt{\frac{(M-1)}{2M}} x(V_{DD})$$

$$V_{RMS(ON)} = \sqrt{\frac{2+(M-1)}{2M}} x(V_{DD})$$

其中，B 是偏压，M 是占空比（即 COM 的数量）。这样可使四个 COM 的识别率（D）等于 1.291。数字相关模式还能够使用 1.8 V 的 V_{DD} 来驱动 3 V 的显示屏。

22.2.2 建议使用的驱动模式

与数字相关模式相比，PWM 驱动模式具有较高的识别率，如 22.2.1.1 PWM 驱动模式及 22.2.1.2 数字相关模式所示。因此，数字相关方法的对比度小于 PWM 方法的对比度，但由于数字相关方法的波形切换频率较低，所以它的功耗也较低。

数字相关模式对 TN 显示屏提供了较低但仍可接受的对比度，但在对比度较高的 STN 显示屏上，对比度和视角便没有明显区别。因为每种模式都各有优劣，所以对使用情况提出如下建议：

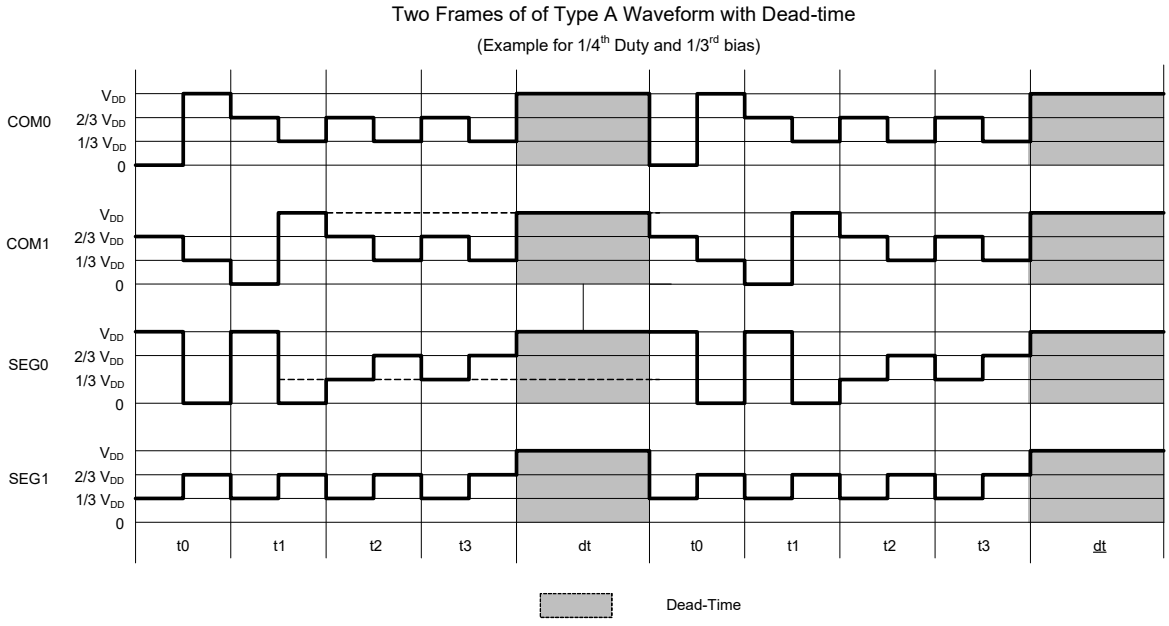
表 22-1. 推荐使用的驱动模式

显示屏类型	深度睡眠模式	睡眠 / 活动模式	注意
带有四个 COM 的 TN 液晶显示屏	数字相关模式	1/3偏压下的PWM驱动模式	进入深度睡眠状态或唤醒状态前，固件必须切换各种 LCD 的驱动模式。
带有四个 COM 的 STN 液晶显示屏	数字相关模式		STN 液晶显示屏使用 PWM 驱动时不会带来对比度上的优势。
带有 8 个或 16 个 COM 的 STN 液晶显示屏	不支持	1/4 偏压和 1/5 偏压下的 PWM 驱动模式	仅在高速 LCD 模式下得到支持。低速时钟会使 PWM 不能在高复用率的条件下运行。

22.2.3 数字对比度控制

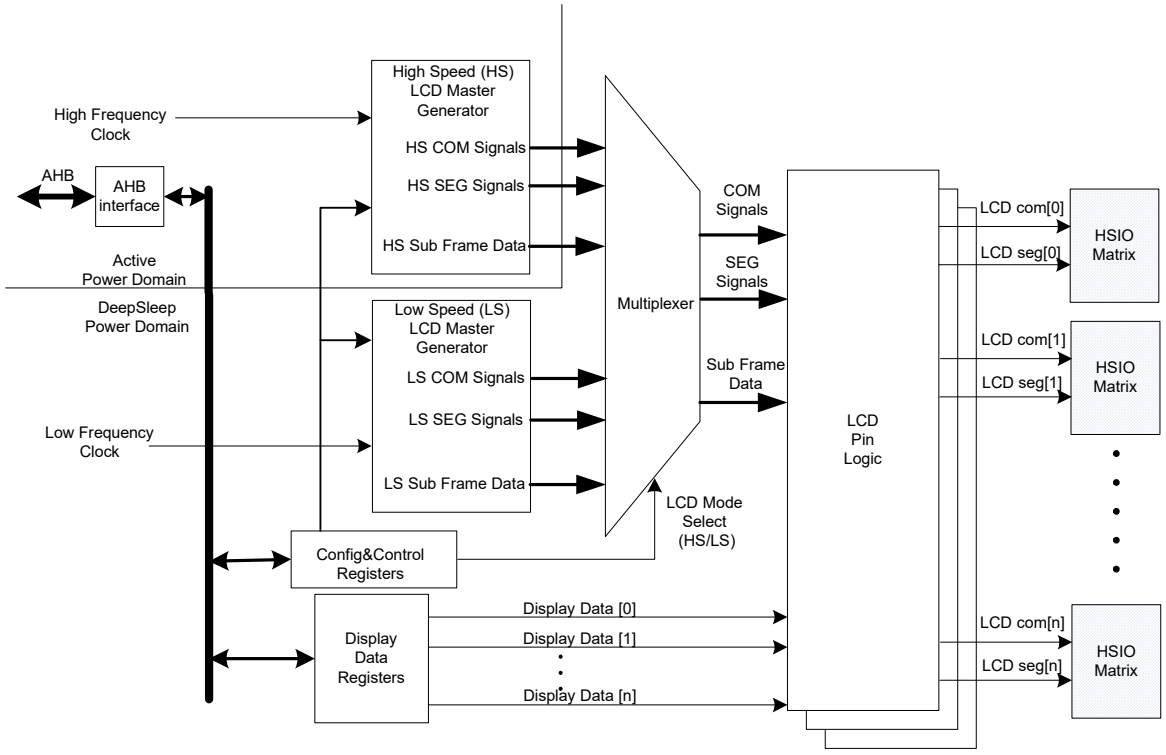
在所有驱动模式下，均可使用数字对比度控制来更改各 segment 的对比度。该方法通过缩短各 segment 的驱动时间来降低对比度。通过在各帧后插入一个死区时间间隔来完成该操作。在死区时间内，所有 COM 和 SEG 信号都被驱动为逻辑 “1” 。在分辨率较好的条件下，可以控制死区时间。图 22-8 说明了在 1/3 偏压和 1/4 占空比的情况下所实现的死区时间对比度控制方法。

图 22-8. 死区时间对比度控制



22.3 框图

图 22-9. LCD 直接驱动系统框图



22.3.1 工作原理

LCD 控制器模块包括两个发生器：其中一个使用高速时钟源 HFCLK，另一个使用由 ILO 生成的低速时钟源 （32 kHz）。这两个发生器分别被称为高速LCD主发生器和低速LCD主发生器。它们都支持 PWM 驱动模式及数字相关驱动模式。若 PWM 驱动模式使用了低速发生器，则需要使用外部电阻，如 第 300 页上的 PWM 驱动模式 所示。

通过固件配置，复用器将选择一个发生器输出，用以驱动 LCD。LCD 引脚逻辑模块将 COM 及 SEG 输出从发生器路由到相应的 I/O 矩阵。任何 GPIO 均可作为 COM 或 SEG 使用。该 COM 或 SEG 的可配置引脚分配会在 GPIO 和 I/O 矩阵上进行，请参考第 77 页上的高速 I/O 矩阵 。这两个发生器共用了相同的配置寄存器。这些存储器映射 I/O 寄存器通过使用 AHB 接口连接到系统总线 （AHB）。

LCD 控制器可在活动、睡眠及深度睡眠三种器件功耗模式下运行。在活动和睡眠模式下能够进行高速操作。在活动、睡眠及深度睡眠模式下能够进行低速操作。在休眠及停止模式下，LCD 控制器被断电。

22.3.2 高速和低速主发生器

高速和低速主发生器几乎相同。唯一的差别是高速主发生器具有较大频率的分频器，用于生成帧周期及子帧周期。这是因为高速模块 （HFCLK）的时钟是从 IMO 派生的，而 IMO 的频率则通常是供给低速模块的 ILO （32 kHz）频率的 30 到 100 倍。高速发生器在活动电压域内运行，而低速发生器则在深度睡眠电压域内运行。另外，还提供了一组配置寄存器，用以控制高速和低速模块。每个主发生器都具有以下特性及特点：

- 用于配置 A 类型或 B 类型驱动波形的模块的寄存器位 （LCD_CONTROL 寄存器中的 LCD_MODE 位）。
- 用于选择 COM 数量的寄存器位 （LCD_CONTROL 寄存器中的 COM_NUM 字段）。可用值分别是 2、 3 和 4。
- 通过使能操作模式配置位，可以选择以下模式：
  - 数字相关模式
  - 1/2 偏压下的 PWM 驱动模式
  - 1/3 偏压下的 PWM 驱动模式
  - 1/4 偏压下的 PWM 驱动模式（在低速发生器中不受支持）
  - 在 1/5 偏压下的 PWM 驱动模式（在低速发生器中不受支持）
  - 关闭 / 禁用模式。通常，两个发生器中的一个被配置为 Off （关闭）

  LCD_CONTROL 寄存器中的 OP_MODE 和 BIAS 字段用于选择驱动模式。

- 具有一个用于生成子帧时序的计数器。LCD_DIVIDER 寄存器中的 SUBFR_DIV 字段用于确定各子帧的持续时间。如果写入该计数器的分频值为 C，那么子帧持续时间为 4 × (C+1)。低速发生器具有一个 8 位的计数器。该计数器通过 32 kHz ILO 时钟生成一个最大为 8 ms 的半子帧周期。高速发生器具有一个 16 位的计数器。
- 具有一个用于生成死区时间周期的计数器。这些计数器与子帧周期计数器具有相同的位数量，并且还使用了相同的时钟。LCD_DIVIDER 寄存器中的 DEAD_DIV 字段控制着死区时间周期。

22.3.3 复用器及 LCD 引脚逻辑

复用器选择高速或低速主发生器模块的输出信号，并将该信号传输给 LCD 引脚逻辑。该选项由配置及控制寄存器控制。LCD 引脚逻辑通过使用复用器的子帧信号来选择显示数据。该引脚逻辑将被复制，以供给各个 LCD 引脚。

22.3.4 显示数据寄存器

每个 LCD segment 引脚都是一个 LCD 端口的一部分，各自都有独立的 LCD_DATAnx 显示屏数据寄存器。这样的 LCD 端口在器件中共有 8 个。请注意，这些端口不是真正的引脚端口，而是 LCD 硬件中有效的端口 / 连接，用于将各 segment 映射到各 common。所配置的每一个 LCD segment 都被作为这些 LCD 端口中的一个引脚使用。 LCD_DATAnx 寄存器是 32 位宽寄存器，用于存储设计中所有被使能的 SEG-COM 组合的 ON/OFF 数据。LCD_DATA0x 寄存器用于保持了 COM0 到 COM3 的 SEG-COM 数据。每个 LCD_DATA0x 寄存器的位 [4i+3:4i]（ ‘i’ 表示引脚编号）显示了端口 [x] 及 COM[3、2、1、0] 组合中的引脚 [i] 的 ON/OFF 状态，如表 22-2 所示。应根据每一帧显示的数据对 LCD_DATAnx 寄存器进行编程。显示数据寄存器是存储器映射 I/O （MMIO），并且可以通过 AHB 从设备接口访问它。

表 22-2. LCD_DATA0x 寄存器的 SEG-COM 映射情况 （每个 SEG 表示 LCD 端口上的一个引脚）

BITS[31:28] = PIN_7[3:0]				BITS[27:24] = PIN_6[3:0]			
PIN_7-COM3	PIN_7-COM2	PIN_7-COM1	PIN_7-COM0	PIN_6-COM3	PIN_6-COM2	PIN_6-COM1	PIN_6-COM0
BITS[23:20] = PIN_5[3:0]				**BITS[19:16] = PIN_4[3:0]**			
PIN_5-COM3	PIN_5-COM2	PIN_5-COM1	PIN_5-COM0	PIN_4-COM3	PIN_4-COM2	PIN_4-COM1	PIN_4-COM0
BITS[15:12] = PIN_3[3:0]				**BITS[11:8] = PIN_2[3:0]**			
PIN_3-COM3	PIN_3-COM2	PIN_3-COM1	PIN_3-COM0	PIN_2-COM3	PIN_2-COM2	PIN_2-COM1	PIN_2-COM0
BITS[7:3] = PIN_1[3:0]				**BITS[3:0] = PIN_0[3:0]**			
PIN_1-COM3	PIN_1-COM2	PIN_1-COM1	PIN_1-COM0	PIN_0-COM3	PIN_0-COM2	PIN_0-COM1	PIN_0-COM0

22.4 寄存器列表

表 22-3. LCD 直接驱动寄存器列表

寄存器名称	说明
LCD_ID	该寄存器包含了 LCD 控制器 ID 及版本编号的信息
LCD_DIVIDER	该寄存器用于控制子帧及死区时间
LCD_CONTROL	该寄存器用于配置高速和低速发生器
LCD_DATA0x	COM0 到 COM3 的 LCD 端口引脚数据寄存器； x 表示端口编号，共有 8 个可用端口

23. CapSense

PSoC® 4 使用了一种电容式触摸感应方法，即 CapSense® Sigma Delta（CSD）。CapSense Sigma Delta 触摸感应方法能够提供行业中最佳的信噪比（SNR）。CSD 是硬件技术和固件技术的结合。本章节介绍的是在 PSoC 4 中如何实现 CSD 硬件。

请参见 PSoC 4 CapSense 设计指南的内容，了解有关基本 CSD 操作、可用的 CapSense 设计工具、容易使用的 PSoC Creator 组件、使用调谐器 GUI 来调校性能以及 PCB 布局设计注意事项等详细信息。

23.1 特性

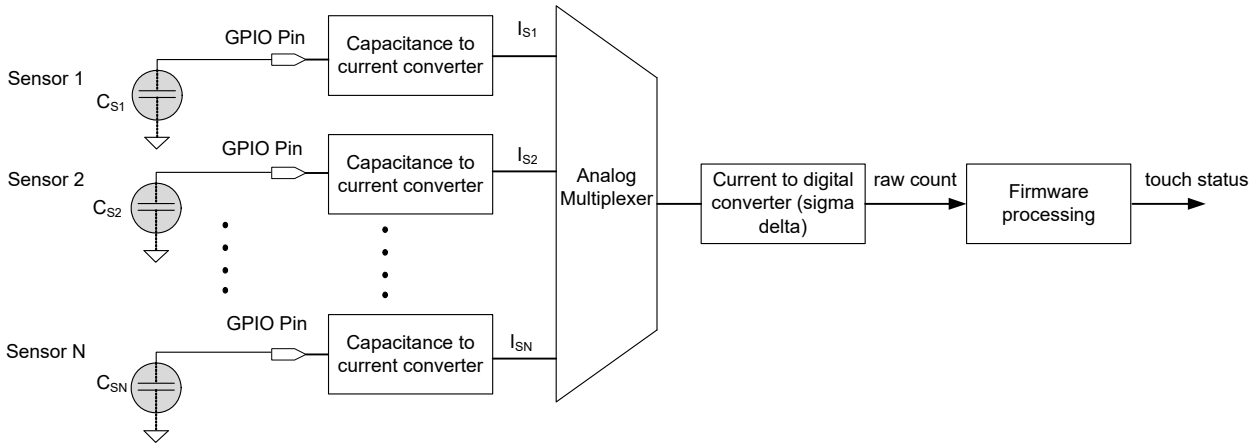
PSoC 4 CapSense 具有以下特性：

- 强大的感应技术
- CSD 操作提供了最佳的 SNR
- 在不同覆盖材料和厚度的条件下仍能提供高性能感应
- SmartSense™ 自动调校技术
- 支持多达 35 个传感器
- 具有大范围的接近感应
- 所有 GPIO 都支持基于屏蔽信号的防水功能
- 低功耗
- 同时使用两个 IDAC，可提高扫描速度和 SNR
- 任何 GPIO 引脚都可用于感应或屏蔽
- 伪随机序列（PRS）时钟源可降低电磁干扰（EMI）
- 专用的充电槽电容，用来在屏蔽线路上快速转移电荷
- GPIO 单元预充电，支持快速初始化外部槽电容

23.2 框图

图 23-1 显示的是 CSD 系统框图。

图 23-1. CapSense 模块 框图



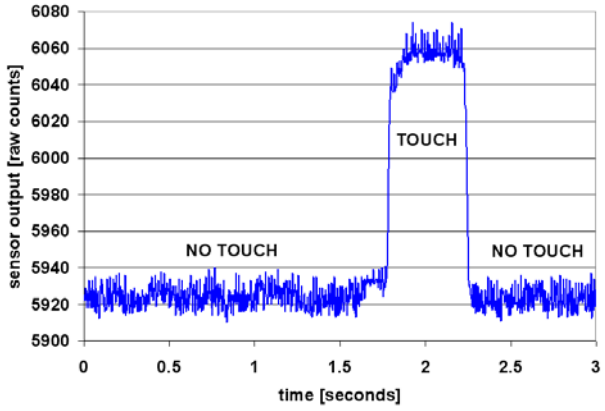
23.3 工作原理

采用 CSD 时，每个 GPIO 均有一个开关电容电路，用于将传感器电容转换为等效电流。然后，模拟复用器会选择其中一个电流信号并将其送到电流 - 数字转换器。电流 - 数字转换器的工作原理与 Delta Sigma ADC 的工作原理相似。

电流 - 数字转换器的输出计数值 （被称为原始计数值）是一个与传感器电容大小成正比的数字值。

图 23-2 显示的是一段时间内的初始计数值图。当手指触摸传感器时，原始计数值会随着传感器电容值增加而增大。通过将初始计数值与某个预定的阈值进行对比，固件中的逻辑可判定传感器是否处于激活状态 （即存在手指触摸）。

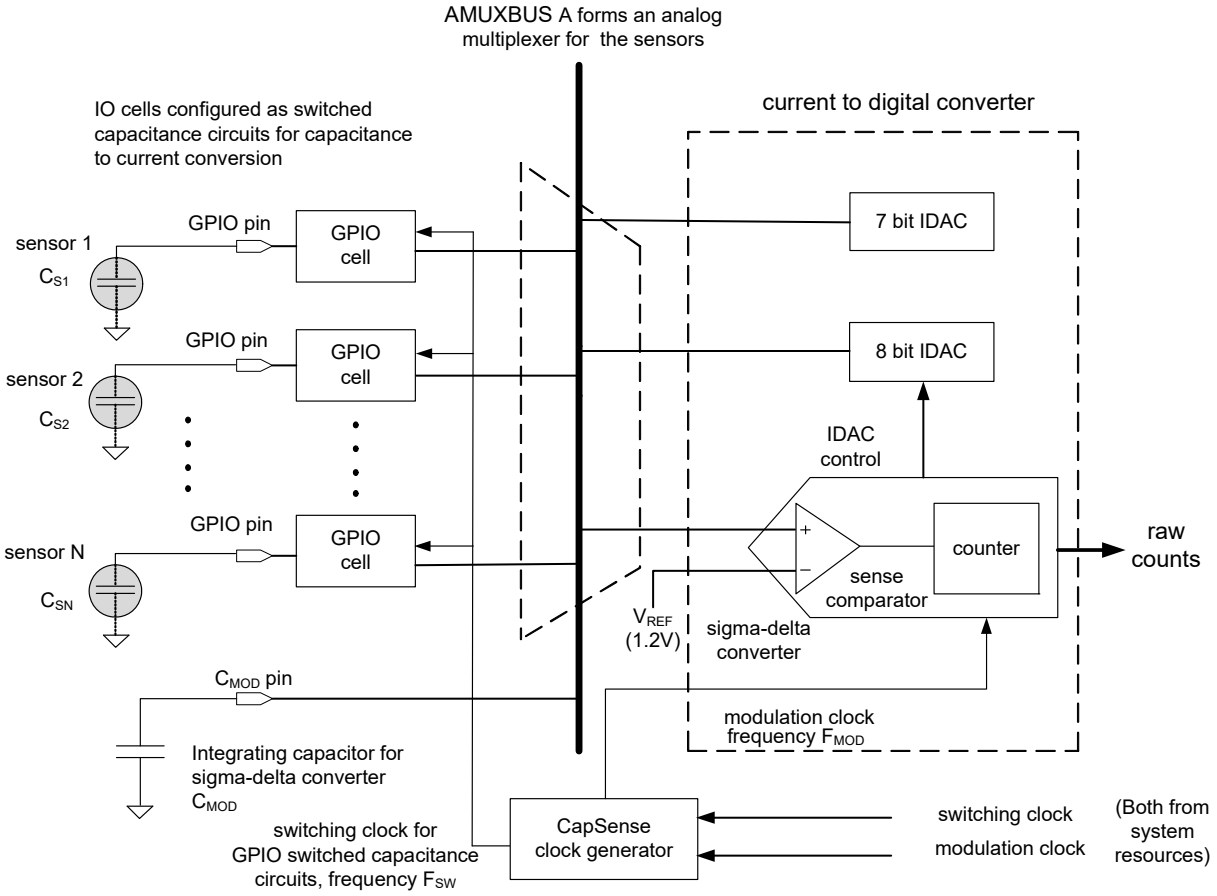
图 23-2. 原始计数值与时间



23.4 CapSense CSD 感应

图 23-3 显示的是 PSoC 4 CapSense 硬件的框图。

图 23-3. PSoC 4 CapSense CSD 感应



23.4.1 GPIO 单元中电容 - 电流转换器

在 CapSense CSD 系统中， GPIO 单元被配置为将传感器电容转换为等效电流的开关电容电路。图 23-4 显示的是 PSoC 4 GPIO 单元结构的简单框图。

PSoC 4 具备两条模拟复用器总线，其中：AMUXBUS A 适用于 CSD 感应； AMUXBUS B 适用于 CSD 屏蔽。 GPIO 开关电容电路有两种配置：向 AMUXBUS A 提供电流或从 AMUX-BUS A 接收灌电流。图 23-5 显示的是通过向 AMUXBUS A 提供电流的开关电容配置。

图 23-4. PSoC 4 中的 GPIO 单元

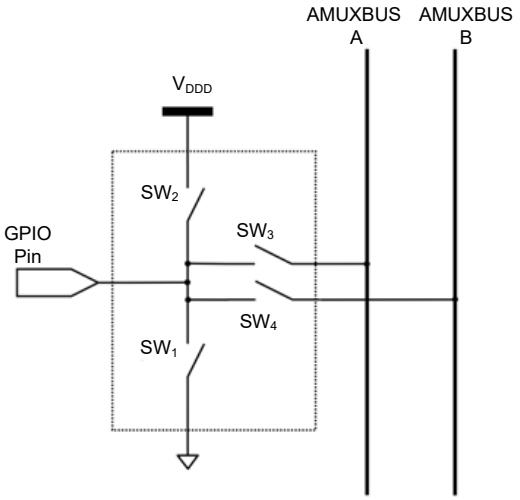
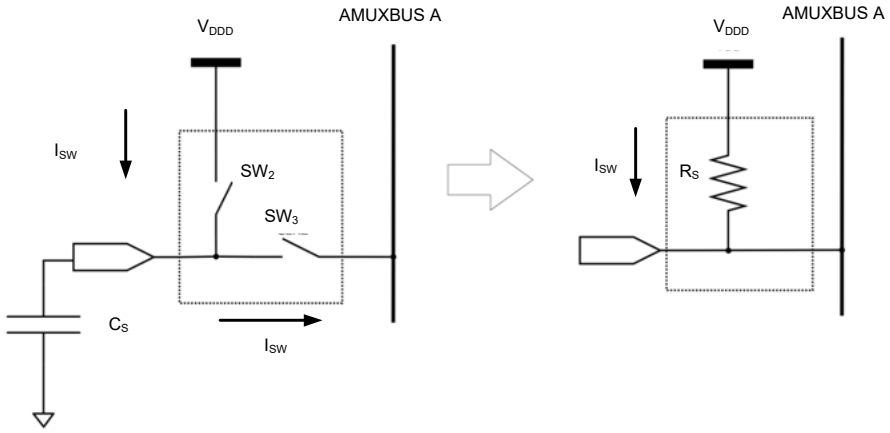


图 23-5. 向 AMUXBUS A 提供电流



两个非重叠、非重相位时钟的频率 F_{SW}（请参考图 23-3）控制着开关 SW_2 和 SW_3。连续切换 SW_2 和 SW_3 会构成一个等效的电阻 R_S，如图 23-5 所示。等效的 R_S 电阻值为：

$$R_S = \frac{1}{C_S F_{SW}}$$

公式 23-1

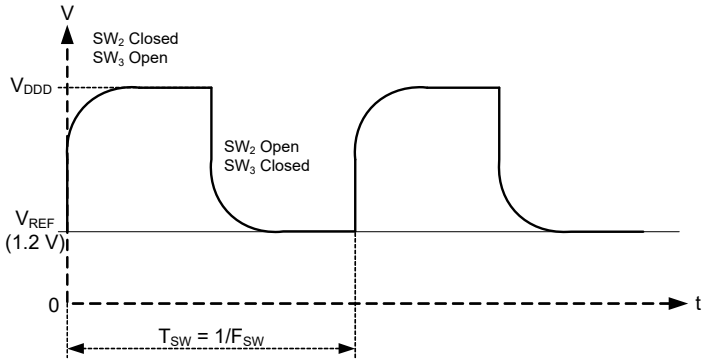
其中：

C_S = 传感器电容

F_{SW} = 开关时钟频率

Sigma Delta 转换器使 AMUXBUS A 的电压保持为一个常量 V_{REF}（在第 315 页上的 Sigma Delta 转换器 中详细介绍了该过程）。图 23-6 显示的是传感器电容的电压波形。

图 23-6. 传感器电容上的电压



通过公式 23-3，可以计算得出提供给 AMUXBUS A 的平均电流。

$$I_S = C_S F_{SW}(V_{DDD} - V_{REF})$$

公式 23-2

图 23-7 显示了从 AMUXBUS A 接收电流的开关电容配置。图 23-8 则显示的是 C_S 上的电压波形结果。

图 23-7. 从 AMUXBUS A 接收电流

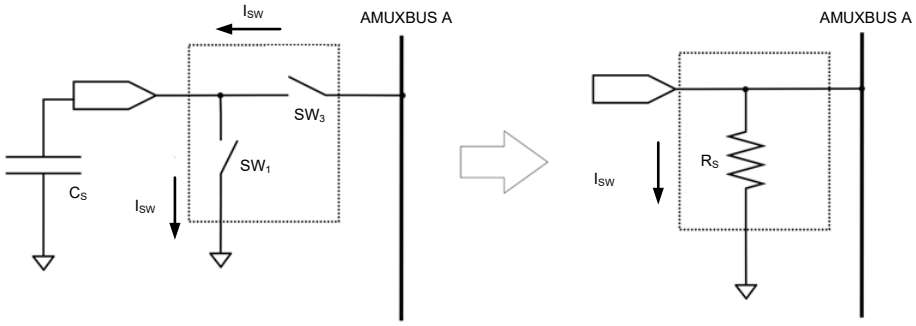
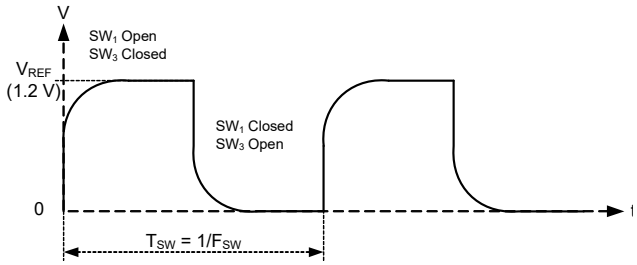


图 23-8. 传感器电容上的电压



通过公式23-4，可以计算出从AMUXBUS A输出的平均电流。

$$I_S = C_S F_{SW} V_{REF}$$

公式 23-3

Sigma Delta 转换器每次扫描一个传感器。通过 AMUXBUS A，可以选择一个 GPIO 单元，并将该单元连接至 Sigma Delta 转换器的引脚上，如图 23-3 所示。AMUXBUS A 和 GPIO 单元开关共同组成该模拟复用器（请参见图 23-4 中的 SW_3）。AMUXBUS A 可以连接到所有支持 CSD 的 PSoC 4 引脚上。欲了解 CSD 引脚的信息，请参考器件数据手册 中的内容。

请参见第 69 页上的 I/O 系统章节，了解如何为感应、屏蔽和连接 C_{MOD} 配置 GPIO 单元。

23.4.2 CapSense 时钟发生器

该模块和系统资源中的外设时钟生成开关时钟 F_{SW} 和调制时钟 F_{MOD}，如图 23-3 所示。有关详细信息，请参见第 83 页上的时钟系统章节。

GPIO单元切换电容电路时，需要使用开关时钟。Sigma Delta 转换器使用调制时钟执行时序操作。

使用系统资源中的两个外设时钟（CSDCLK0 和 CSDCLK1）生成所需的频率。有关详细信息，请参见第 83 页上的时钟系统章节。CSDCLK0 生成调制时钟，另外 CSDCLK1 生成开关时钟。

然而，最终的开关时钟频率取决于 CapSense 的时钟发生器。它具有以下输出选项：

- 直接：直接使用可编程时钟分频器的输出。想要选择该选项，请在 CSD_CONFIG 寄存器 ‘1’ 中设置 BYPASS_SEL 位。
- 二分频。对时钟频率进行二分频。想要选择该选项，请在 CSD_CONFIG寄存器中清除PRS_SELECT和BYPASS_SEL 位。
- 伪随机序列 （PRS）：通过在更大的范围内扩展开关频率，可以降低 CapSense 系统中的 EMI。想要选择该选项，请在CSD_CONFIG寄存器中设置PRS_SELECT位，并清除 BYPASS_SEL 位。通过在同一个寄存器中使用 PRS_12_8 位，可以选择 8 位伪随机序列或 12 位伪随机序列。具体操作为：通过设置 PRS_12_8 位，可以选择 12 位伪随机序列；通过清除该位选择 8 位 PRS。

如果已经选择了 PRS，则最大开关频率为

$$F_{SW(maximum)} = \frac{F_{in}}{2}$$

公式 23-4

其中：F_{in} 是 CSDCLK1 的输出频率。最小频率为：

$$F_{SW(minimum)} = \frac{F_{in}}{\text{PRS length-1}}$$

公式 23-5

其中：PRS 的长度为 12 位或 8 位。平均开关频率为：

$$F_{SW(average)} = \frac{F_{in}}{4}$$

公式 23-6

CSD_CONFIG 寄存器中的 PRS_CLEAR 位可用于清除 PRS；设置该位时，它会强制伪随机发生器返回到初始状态。

23.4.3 Sigma Delta 转换器

Sigma Delta 转换器将输入电流转换为一个相应的数字计数值。它包含一个比较器、一个参考电压 V_{REF}、一个计数器以及两个拉电流 / 灌电流数模转换器（IDAC），如图 23-3 所示。

Sigma delta 调制器以打开 / 关闭方式来控制 8 位 IDAC 的电流。该 IDAC 被称为调制 IDAC。7 位 IDAC （即称为补偿 IDAC）始终为打开或关闭状态。

Sigma delta 转换器可在单 IDAC 模式或双 IDAC 模式下运行。在单 IDAC 模式下，补偿 IDAC 始终处于关闭状态。在双 IDAC 模式下，补偿 IDAC 始终处于打开状态。

Sigma Delta 转换器还需要一个外部积分电容 C_{MOD}，如图 23-1 所示。C_{MOD} 的推荐值为 2.2 nF。PSoC 4 具有一个专用的 C_{MOD} 引脚。欲了解更详细的信息，请参考 器件数据手册 中的引脚分布一节的内容。

Sigma Delta 调制器保持 C_{MOD} 上的电压为 V_{REF}。它在下列某种模式下工作：

- IDAC 拉电流模式：如果开关电容电路从 AMUXBUS A 接收电流，那么，IDAC 将为 AMUXBUS A 提供电流，以使其电压平衡。
- IDAC 灌电流模式：在该模式下，IDAC 从 C_{MOD} 接收电流，同时开关电容电路会为 C_{MOD} 提供电流。

在这两种情况下，随着 C_{MOD} 上电压发生的微小变化，将调制 IDAC 电流切换为打开或关闭状态，从而保持 C_{MOD} 的电压等于 V_{REF}。

Sigma Delta 转换器的工作范围为 8 位到 16 位分辨率。在单 IDAC 模式下，原始计数值与传感器电容值成正比。如果 ‘N’ 是 Sigma Delta 转换器的分辨率，并且 I_{MOD} 是调制 IDAC 电流的值，则使用公式 16-7 可计算出 IDAC 拉电流模式下相应的原始计数值。

$$Rawcount = 2^N \frac{V_{REF} F_{SW}}{I_{MOD}} C_S$$

公式 23-7

同样， IDAC 灌电流模式下原始计数的近似值为：

$$Rawcount = 2^N \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S$$

公式 23-8

在这两种情况下，原始计数值均与传感器电容值 C_S 成正比。通过固件来处理原始计数值，以检测触摸。您可以使用双 IDAC 模式下的两种 IDAC，以提高 CapSense 的性能。

在该双 IDAC 模式下，补偿 IDAC 始终处于打开状态。如果 I_{COMP} 是补偿 IDAC 电流，则通过下面的公式可计算出 IDAC 拉电流模式下的原始计数值：

$$Rawcount = 2^N \frac{V_{REF} F_{SW}}{I_{MOD}} C_S - 2^N \frac{I_{COMP}}{I_{MOD}}$$

公式 23-9

IDAC 灌电流模式下的原始计数值可通过公式 16-10 计算得出。

$$Rawcount = 2^N \frac{(V_{DD} - V_{REF}) F_{SW}}{I_{MOD}} C_S - 2^N \frac{I_{COMP}}{I_{MOD}}$$

公式 23-10

请注意，原始计数值始终为正值。

应将硬件参数 （如：I_{COMP}、 I_{MOD} 和 F_{SW}）调试为最佳值，用以执行可靠的触摸测试。欲了解调试过程的详细信息，请参考 PSoC 4 CapSense 设计指南中的内容。

CSD_CONFIG、CSD_COUNTER 和 CSD_IDAC 寄存器控制着 Sigma delta 转换器的操作。CSD_CONFIG 寄存器中的各重要位包括：

- CSD_CONFIG 中的 ENABLE 位：它是 CSD 模块的主设备使能位。必须将该位设置为 ‘1’，以使 CSD 操作有效。
- CSD_CONFIG 寄存器中的 POLARITY 位：用于选择 IDAC 灌电流模式或者 IDAC 拉电流模式。 0 值表示 IDAC 拉电流模式， 1 值表示 IDAC 灌电流模式。
- CSD_CONFIG 寄存器中的 SENSE_COMP_BW 位：用于选择感应比较器的带宽。设置该位可提供高带宽，清除它则提供的是低带宽。建议在 CSD 操作中使用高带宽。
- CSD_CONFIG 中的 SENSE_COMP_EN 位：用于启动感应比较器电路。 ‘0’ 表示感应比较器的电源被关闭 。 ‘1’ 表示感应比较器的电源被打开。
- SENSE_EN 位：使能 Sigma delta 调制器输出。另外，它还会启动各 IDAC。

必须准确配置 IDAC，以用于 CSD 操作有关详细信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中介绍的 CSD_IDAC 寄存器的内容。

CSD_COUNTER 寄存器用于对当前选定的传感器进行采样，并读取结果。每当以调制时钟频率对比较器进行采样时，如果采样结果为 ‘1’，则该寄存器中的 16 位计数器字段会增加。每次开始进行一个新感应操作时，固件通常会将 ‘0’ 写入到该字段内。CSD_COUNTER 寄存器中的 16 位周期字段用于初始化将电容值转换为数字值的操作。将一个非零值写入到该寄存器中会初始感应操作。通过固件写入到该字段的值可以确定计数器字段对比较器输出进行采样的周期。

开始进行 CSD 操作前，必须正确配置时钟、GPIO、IDAC 和 Sigma delta 调制器等项。每个调制时钟周期过后，周期字段都会减少。当它的值达到 0 时，计数器字段会停止递增。这时，字段值等于传感器中电容值相应的原始计数值。

23.5 CapSense CSD 屏蔽

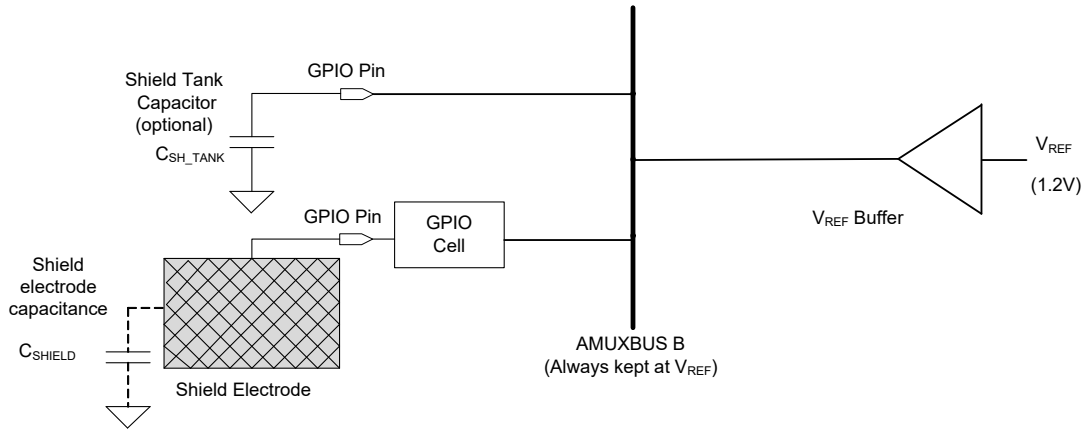
PSoC 4 CapSense 支持用于防水和接近感应性能的屏蔽电极。对于防水性能，屏蔽电极总是保持为与传感器相同的电位。PSoC 4 CapSense 具有一个屏蔽电路，该电路会使用传感器开关信号的副本来驱动屏蔽电极 （请参考第 313 页上的 GPIO 单元中电容 - 电流转换器 ），这样可以避免传感器与屏蔽电极间潜在的差异。欲了解屏蔽的基本信息，请参考 PSoC 4 CapSense 设计指南的内容。

在感应电路中， Sigma Delta 转换器保持使 AMUXBUS A 的电压等于 V_{REF} （请参考第 315 页上的 Sigma Delta 转换器 ）。通过切换 AMUXBUS A 与电源轨 （是 V_{DD} 或接地，取决于其配置）之间的传感器， GPIO 单元可生成传感器波形。屏蔽电路也使用类似的方式工作； AMUXBUS B 电压始终保持为 V_{REF}。GPIO 单元会切换 AMUXBUS B 与电源轨（等于 V_{DD} 或接地，该配置与传感器配置相同）之间的屏蔽。该过程会生成屏蔽电极上的传感器开关波形的副本。

根据保持 AMUXBUS B 的电压为 V_{REF} 的方法，有以下两种不同的配置。

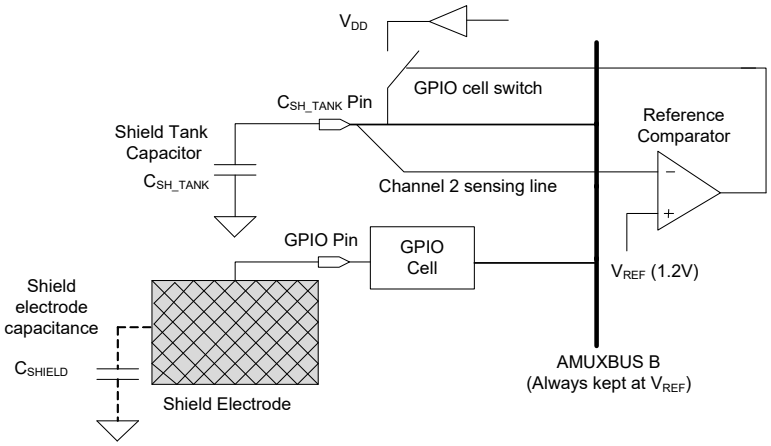
- 使用 V_{REF} 缓冲区驱动屏蔽：在此配置中，会使用一个电压缓冲区将 AMUXBUS B 驱动为 V_{REF}，如图 23-9 所示。推荐使用外部电容 C_{SH_TANK} 来降低开关跃变。设置 CSD_CONFIG 寄存器中 REBUF_OUTSEL 位可以将缓冲区输出连接至 AMUXBUS B。同一寄存器中的 REFBUF_DRV 位字段可用来设置缓冲区的驱动能力。将 ‘0’ 写入到该字段可禁用该缓冲区；将 1、2 和 3 写入到该字段可分别选择低电流、中电流和高电流驱动模式。

图 23-9. 使用 V_{REF} 缓冲区驱动屏蔽



- 使用 GPIO 单元预充电驱动屏蔽：该配置需要一个外部电容 C_{SH_TANK}，如图 23-10 所示。通过特殊的 GPIO 单元和一个参考比较器，可以给电容 C_{SH_TANK} 充电，这样 AMUXBUS B 会保持其电压为 V_{REF}。参考比较器始终监控着 CSH_TANK 电容的电压，并控制 GPIO 单元开关，以维持其电压为 V_{REF}。使用专用的感应线路 （称为通道 2 感应线路），可将参考比较器连接至 C_{SH_TANK} 电容上，如图 23-10 所示。

图 23-10. 使用 GPIO 预充电驱动屏蔽



GPIO 单元预充电功能仅适用于 C_{SH_TANK} 固定引脚。欲了解更详细的信息，请参考 器件数据手册中器件引脚分布一节的内容。

CSD_CONFIG 寄存器中的 COMP_MODE 位可选择参考缓冲区预充电或 GPIO 预充电模式：0 值表示参考缓冲区预充电，1 值表示 GPIO 预充电。

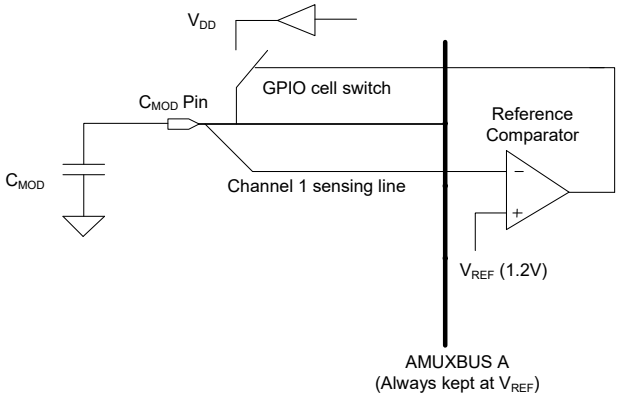
23.5.1 C_{MOD} 预充电

第一次使能 CapSense 硬件时， C_{MOD} 上的电压为 0 V。然后，Sigma Delta 转换器会缓慢地为 C_{MOD} 充电，使其电压达到 V_{REF}。充电电流来自各 IDAC （在 IDAC 拉电流模式中）或传感器开关电容电路（在 IDAC 灌电流模式中）。然而，由于 C_{MOD} 是一个较大的电容，因此，充电过程相当缓慢。

C_{MOD} 预充电指的是快速初始化 C_{MOD} 上的电压，并使之达到 V_{REF} 的过程。使用预充电方式可缩短 Sigma Delta 转换器开始工作所需要的时间。 C_{MOD} 预充电有两种选项。

- 使用 V_{REF} 缓冲区进行预充电：使能屏蔽时，V_{REF} 缓冲区输出始终连接至 AMUXBUS B（图 23-9）。要想使用 V_{REF} 缓冲区进行预充电，应先将 C_{MOD} 连接至 AMUXBUS B。预充电结束后，将 C_{MOD} 连接至 AMUXBUS A，使 Sigma Delta 能够正常操作。屏蔽被禁用时，在预充电期间 V_{REF} 缓冲区输出始终连接着 AMUXBUS A，然后再断开连接。
- 使用 GPIO 单元进行预充电：在该配置中，通过一个特殊的 GPIO 单元和一个参考比较器，为电容 C_{MOD} 充电，使之电压达到 V_{REF}。GPIO 单元预充电功能仅适用于 C_{MOD} 固定引脚。欲了解更详细的信息，请参考 器件数据手册中的引脚分布一节的内容。用于该目的的比较器与预充电 CSH_TANK 的参考比较器相同。通过 CSD_CONFIG 寄存器中的 COMP_PIN 位，可以选择连接至参考比较器的电容。如果该位的值为 0，被指定为 “ 通道 1” 的感应线路用于将 C_{MOD} 连接至参考比较器，如图 23-11 所示；如果该位的值为 1，则通道 2 感应线路用来将 CSH_TANK 连接到参考比较器，如图 23-10 所示。请注意：必须正确配置 GPIO 单元，这样才能进行 GPIO 单元预充电。

图 23-11. GPIO 单元预充电



使用 GPIO 单元进行预充电的速度比使用 V_{REF} 缓冲区的速度快。因此，推荐使用 GPIO 进行预充电。然而，如果您对初始化 CapSense 的时效性要求不高，请使用 V_{REF} 缓冲区进行预充电。

通道 1 感应线路还用来将 C_{MOD} 连接至 Sigma delta 调制器中的感应比较器。通过将 CSD_CONFIG 寄存器中的 SENSE_INSEL 位设置为 ‘1’，可以使能该选项。清除该位可使 C_{MOD} 通过 AMUXBUS A 连接至感应比较器。

23.6 通用资源：IDAC 和比较器

如果触摸感应时没有使用 CapSense 模块，则感应比较器和两个 IDAC 可作为通用模拟模块使用。

您可以使用 AMUXBUS A 将任何支持 CSD 的 GPIO 连接至感应比较器的非反相输入端。反相输入被连接到 1.2 V V_{REF} （请参见图 23-3）。 AMUXBUS A 还可以作为比较器输入的模拟复用器使用。可以使用 CSD_CONFIG 寄存器中的 SENSE_COMP_EN、 SENSE_COMP_BW 和 ENABLE 位控制感应比较器，如第 315 页上的 Sigma Delta 转换器 中所介绍。

如果需要将 AMUXBUS 用于其他目的，CSD_CONFIG 寄存器中的 SENSE_INSEL 位可用于将感应比较器的非反相输入连接至固定的 C_{MOD} 引脚，如第 318 页上的 CMOD 预充电 中的介绍。比较器的输出可以连接到多个 GPIO，请参见第 69 页上的 I/O 系统章节了解详细信息。

8 位 IDAC 的工作电流范围为 0~306 µA（1.2 µA/ 位）或 0~612 µA（2.4 µA/ 位）。7 位 IDAC 的工作电流范围为 0 到 152.4 µA（1.2 µA/ 位）或 0 到 304.8 µA （2.4 µA/ 位）。

8 位和 7 位 IDAC 都可以使用 AMUXBUS A 和 AMUXBUS B 连接到 GPIO。另外，还可以将这两个 IDAC 连接到单个 AMUXBUS。IDAC 可以在下面三个不同的模式下工作：仅 CSD 模式、通用 （GP）模式和 CSD 与通用模式。表 23-1 展示了在每个模式下如何将 IDAC1 和 IDAC2 连接到 AMUXBUS A 和 AMUXBUS B。

表 23-1. IDAC 模式

模式	AMUXBUS A	AMUXBUS B
CSD 模式	IDAC 灌 / 拉电流 （电压为 1.2 V）	没有连接 IDAC
通用模式	8 位 IDAC 灌 / 拉电流	7 位 IDAC 灌 / 拉电流
CSD 和通用 （GP）模式	8 位 IDAC 灌 / 拉电流 （电压为 1.2 V）	7 位 IDAC 灌 / 拉电流

有关详细信息，请参见 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册 （TRM）中介绍的 CSD_IDAC 寄存器的内容。通过 CSD_CONFIG 寄存器可以使能 IDAC，并设置极性，如第 315 页上的 Sigma Delta 转换器 所介绍的。有关如何将 GPIO 连接至 AMUXBUS A 和 AMUXBUS B 的详细信息，请参考第 69 页上的 I/O 系统章节。

23.7 寄存器列表

表 23-2. CapSense 寄存器列表

寄存器名称	说明
CSD_CONFIG	该寄存器用于配置和控制 CSD 模块及其资源。
CSD_IDAC	该寄存器用于控制 IDAC 电流的设置。
CSD_COUNTER	该寄存器用于初始化对选定电容传感器进行的采样，并读取转换结果。
CSD_STATUS	该寄存器允许观察 CSD 模块中的关键信号。
CSD_INTR	它是 CSD 中断请求寄存器。

24. 温度传感器

PSoC® 4 使用片上温度传感器来测量内部 Die （裸片）温度。传感器包含一个在二极管配置中连接的晶体管。

24.1 特性

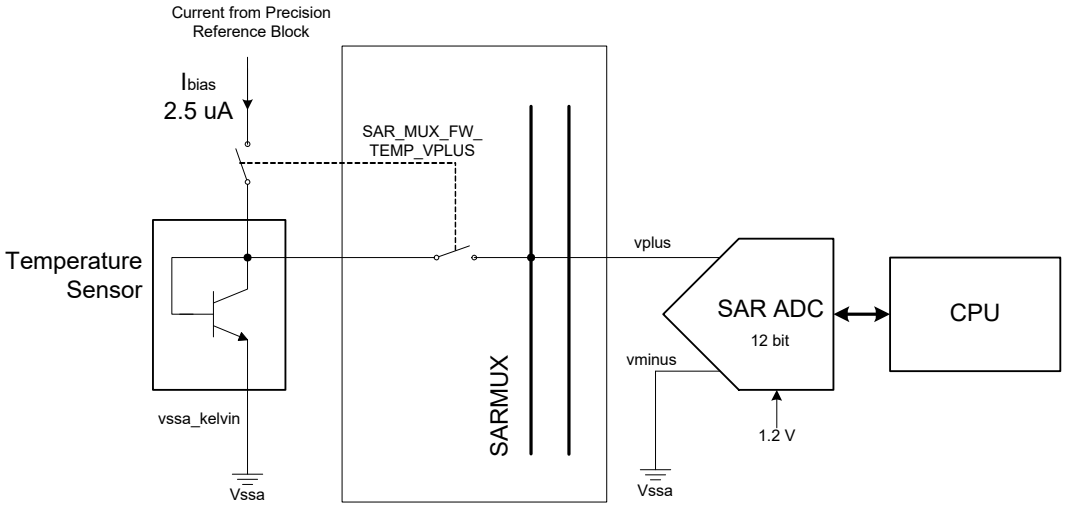
温度传感器的特性包括：

- 温度范围为 –40 °C 到 +85 °C，精度为 ± 5°C
- 当使用一个 12 位 SAR ADC，参考电压为 1.024 V 时，分辨率为 0.5° Celsius/LSB （尚未放大）
- 10 µs 的建立时间

24.2 工作原理

温度传感器包括一个外形为二极管的单端双极结型晶体管 （BJT）。在恒定集电极电流和零集电极 - 基极电压条件下，温度对该二极管的基极 - 发射电压 （V_{BE}）产生的影响非常大。可利用该属性计算裸片 （die）的温度，具体是通过使用 SAR ADC 测量晶体管的 V_{BE}，如图 24-1 所示。

图 24-1. 温度感应机制



通过使用 SAR ADC 可以测量传感器的模拟输出电压 （V_{BE}）。可以通过以下公式使用 ADC 结果计算 Die 温度 （单位为 °C）：

$$Temp = (A \times SAR_{out} + 2^{10}xB) + T_{adjust}$$

公式 24-1

- 温度是指单位为 °C 的斜率补偿温度，它的表示形式为 32 位 （16.16）定点数。
- ‘A’ 是一个16位的乘数常量。通过使用PSoC 4系列的数据计算两点间斜率，可以确定A值。可以按照以下公式计算该值：

$$A = (\text{signed int})\left(2^{16}\left(\frac{100°C-(-40°C)}{SAR_{100°C}-SAR_{-40°C}}\right)\right)$$ **公式 24-2**

其中：

SAR_{100C} 表示温度为 100°C 时 ADC 的计数值

SAR_{-40C} 表示温度为 –40°C 时 ADC 的计数值

常量 ‘A’ 被存储在 SFLASH_SAR_TEMP_MULTIPLIER 寄存器内。

- ‘B’ 是一个 16 位的偏移值。针对每一个裸片（die）计算出 B 值。计算操作所涉及的因素包括加工技术的差异和芯片上实际的偏置电流 （I_{bias}）。可以按照以下公式计算出该值：

$$B = (\text{unsigned int})\left(2^{6}x100°C-\left(\frac{A\times SAR_{100C}}{2^{10}}\right)\right)$$ **公式 24-3**

其中：

SAR_{100C} 表示温度为 100°C 时的 ADC 计数值

常量 ‘B’ 被存储在 SFLASH_SAR_TEMP_OFFSET 寄存器内。

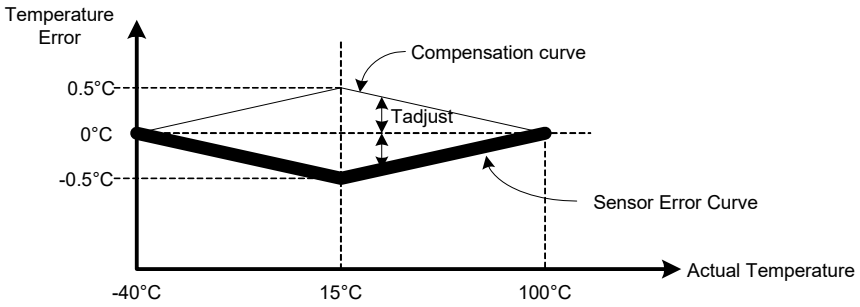
- T_{adjust} 是斜率修正因子 （单位为 °C）。通过使用斜率修正因子可以修正双斜率的温度传感器。根据得到的结果 （未经过斜率修正）计算该值， $T_{initial} = (A\times SAR_{out}+ 2^{10}\times B)$。如果该值大于中心值 （15°C），则可以按照下面公式计算 T_{adjust}。

$$T_{adjust} = \left(\frac{0.5°C}{100°C-15°C}\times(100°C\times 2^{16}-T_{initial})\right)$$ **公式 24-4**

如果该值小于中心值，则可以按照下面公式计算 Tadjust。

$$T_{adjust} = \left(\frac{0.5°C}{40°C+15°C}\times(40°C\times 2^{16}-T_{initial})\right)$$ **公式 24-5**

图 24-2. 温度误差补偿

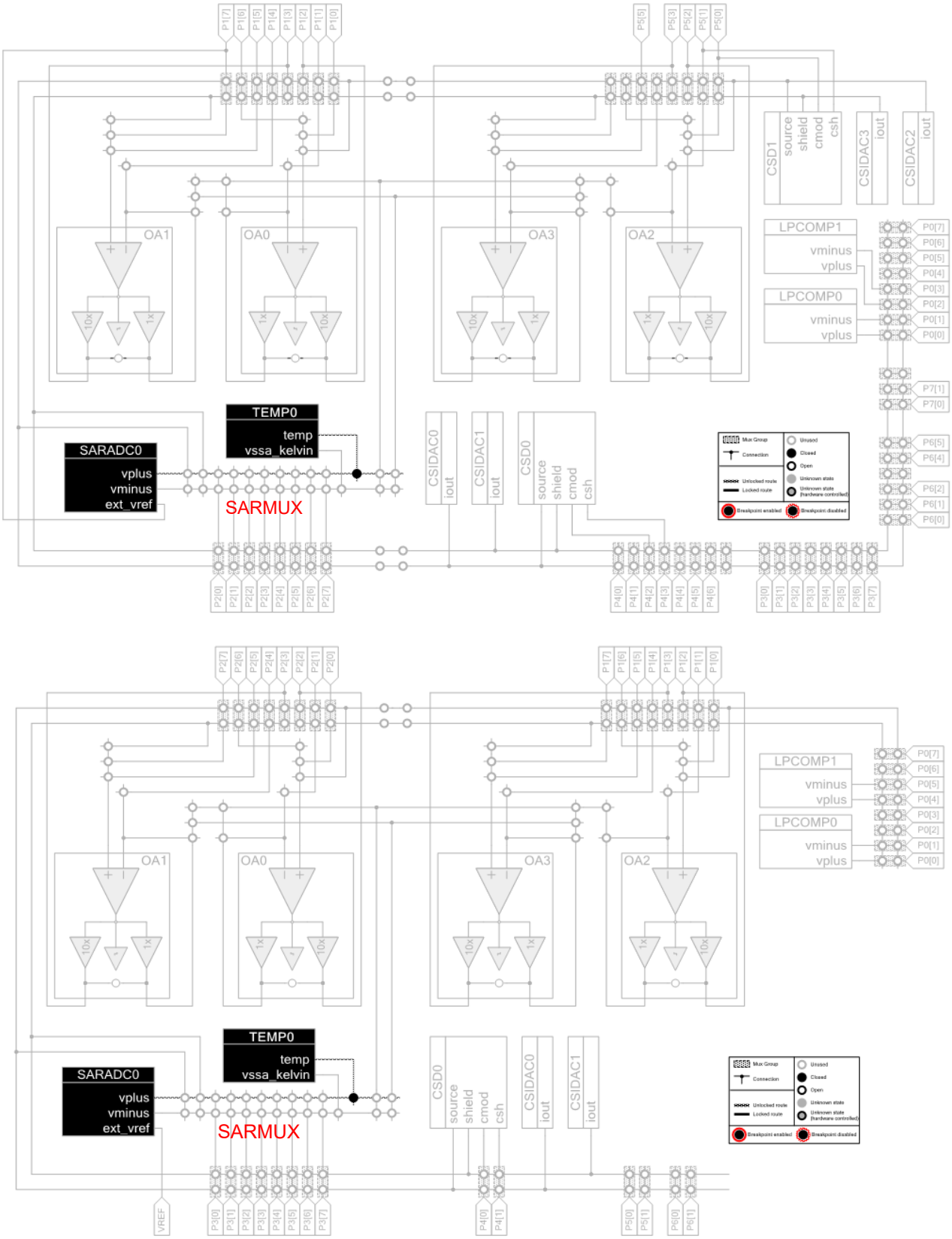


注意： A 和 B 是工厂校准过程中存储在闪存内的 16 位常量。请留意，只有 SAR ADC 在分辨率为 12 位和参考电压为 1.024 V 的条件下运行时，这些常量才有效。

24.3 温度传感器配置

如图 24-3 所示，通过专用开关可以将温度传感器输出路由到 SAR ADC 的正输入端。这些开关可以由序列发生器、固件或数字系统互连 （DSI）控制。通过传输来自高精度参考模块的偏置电流，并将传感器输出连接到 SAR ADC 的正输入端，可以使用开关的控制信号（图 24-1 所示的 SAR_MUX_FW_TEMP_VPLUS）使能温度传感器。SAR_MUX_FW_TEMP_VPLUS 控制位是 SAR_MUX_SWITCH0 寄存器的一部分。通过使用 SAR_MUX_SWITCH_STATUS 寄存器可以读取开关状态。

图 24-3. 将温度传感器输出路由到 SAR ADC



24.4 算法

1. 启用 SARMUX 和 SAR ADC。
2. 将 SAR ADC 配置为单端模式，其中 $V_{NEG} = V_{SS}$， V_{REF} = 1.024 V，分辨率为 12 位，并且结果为右对齐。
3. 启用温度传感器。
4. 从 SAR ADC 中获取数字输出。
5. 分别从 SFLASH_SAR_TEMP_MULTIPLIER 和 SFLASH_SAR_TEMP_OFFSET 上提取 ‘A’ 值和 ‘B’ 值。
6. 使用线性公式 （公式 24-1）计算裸片温度。

   例如， A = 0xBC4B 和 B = 0x65B4。假设在给定的温度条件下， SAR ADC 的输出 （V_{BE}）为 0x595。

   这时，固件会进行以下计算：

   a. 将 A 乘以 V_{BE}：0xBC4B × 0x595 = $(–17333)_{10} \times (1429)_{10} = (–24768857)_{10}$

   b. 将 B 乘以 1024：0x65B4 × 0x400 = $(26036)_{10} \times (1024)_{10} = (26660864)_{10}$

   c. 将第一步和第二步得到的结果相加，从而得到 $T_{initial}$ 值：$(–24768857)_{10} + (26660864)_{10} = (1892007)_{10}$ = 0x1CDEA7

   d. 使用 $T_{initial}$ 值来计算 T_{adjust}：$T_{initial}$ 是高 16 位乘以 2^{16}，即为 0x1C00 = $(1835008)_{10}$。该值超过 15°C （0x1C — 高 16 位）。使用公式 4 计算 T_{adjust}。得到的结果为 0x6C6C = $(27756)_{10}$

   e. 将 T_{adjust} 加上 $T_{initial}$：$(1892007)_{10} + (27756)_{10} = (1919763)_{10}$ = 0x1D4B13

   f. 温度的整数部分为高 16 位值 = 0x001D = $(29)_{10}$

   g. 温度的小数部分为低 16 位值 = 0x4B13 = $(0.19219)_{10}$

   h. 将 f 和 g 步骤的结果结合起来，可得到：Temp = 29.19219 °C ~ 29.2°C

24.5 寄存器

名称	说明
SAR_MUX_SWITCH0	该寄存器包含了 SAR_MUX_FW_TEMP_VPLUS 字段，用于将温度传感器连接到 SAR MUX 终端。
SAR_MUX_SWITCH_STATUS	该寄存器提供了连接到 SAR MUX 的温度传感器开关的状态。
SFLASH_SAR_TEMP_MULTIPLIER	如公式 24-1 所定义的乘数常量 ‘A’。
SFLASH_SAR_TEMP_OFFSET	如公式 24-1 所定义的常量 ‘B’。

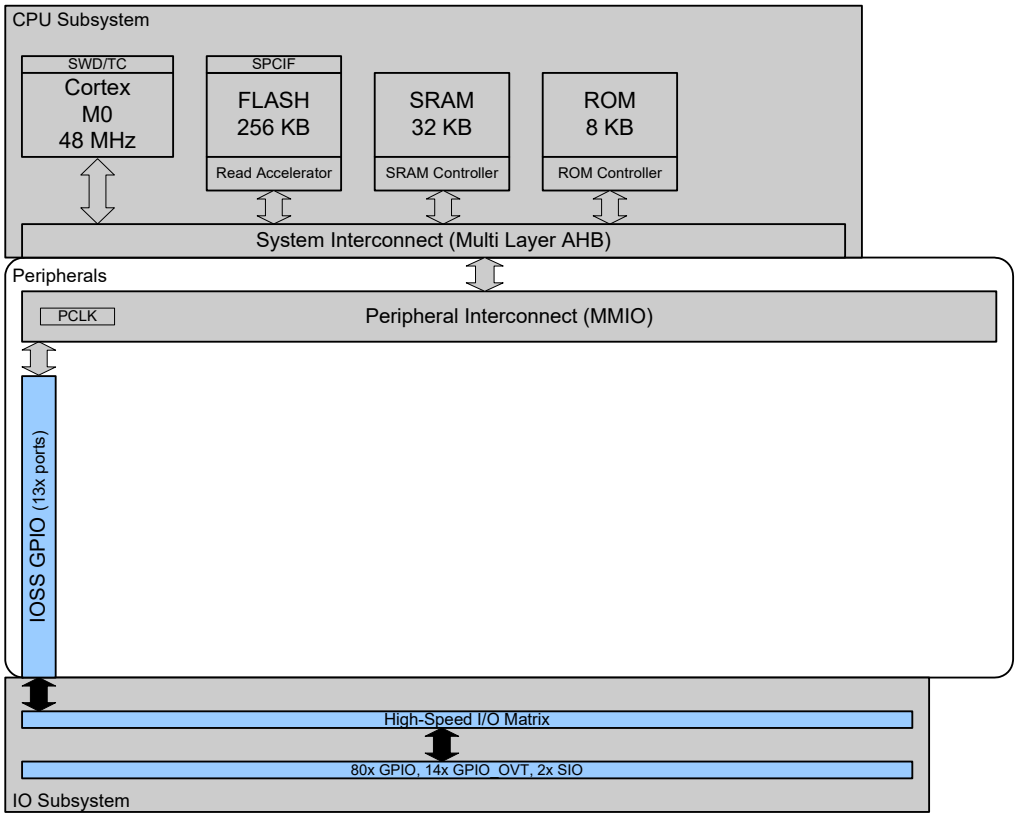
章节 F:　　编程和调试

本部分包括以下章节：

- 第 327 页上的编程与调试接口章节
- 第 335 页上的非易失性存储器编程章节

系统架构

编程和调试框图



26. 编程与调试接口

PSoC® 4 编程与调试接口为外部器件提供了一个用于编程或调试的通信网关。外部器件可以是赛普拉斯供应的编程器和调试器，也可以是支持编程和调试功能的第三方器件。串行线调试 （SWD）接口用作外部器件与 PSoC 4 间的通信协议。

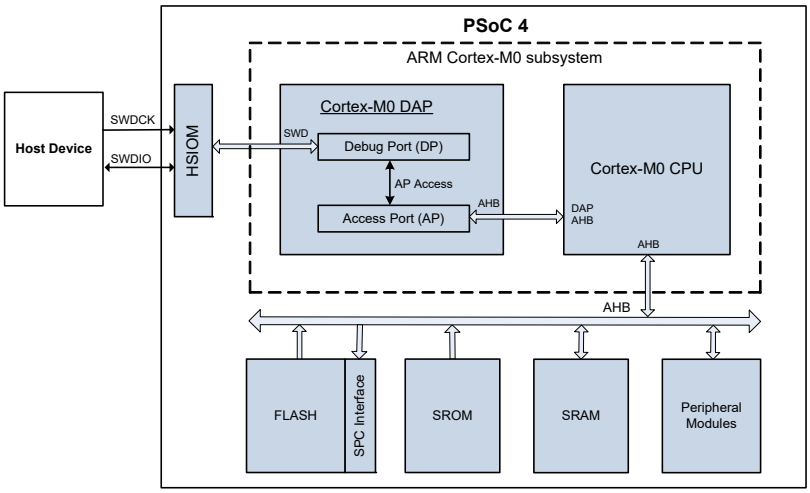
26.1 特性

- 通过 SWD 接口进行编程和调试
- 调试过程中使用四个硬件断点和两个硬件观察点
- 调试过程中能对系统中所有存储器和寄存器 （包括内核处于运行或停止状态下的 Cortex-M0 寄存器组）进行读和写访问

26.2 功能说明

图 26-1 显示的是 PRoC 4 中编程和调试接口的框图。 Cortex-M0 调试和访问端口 （DAP）作为编程和调试接口使用。外部编程器或调试器（即 “ 主机 ”）通过使用 SWD 接口的两个引脚（双向数据引脚（SWDIO）和主机驱动的时钟引脚（SWDCK））与 PSoC 4 “ 目标器件 ” 的 DAP 进行通信。SWD 物理端口引脚（SWDIO 和 SWDCK）通过高速 I/O 矩阵（HSIOM）与 DAP 通信。有关 HSIOM 的详细信息，请参考第 69 页上的 I/O 系统章节。

图 26-1. 编程与调试接口



DAP 使用 ARM 指定的高级和高性能总线 （AHB）接口与 Cortex-M0 CPU 通信。 AHB 是 PSoC 4 器件的内部系统互连协议，用于加快由 AHB 主设备进行的存储器和外设寄存器访问。该器件有两个 AHB 主设备 — ARM CM0 CPU 内核和 DAP。外部器件可通过 DAP 有效地控制整个器件，以实现编程和调试操作。

26.3 串行线调试 （SWD）接口

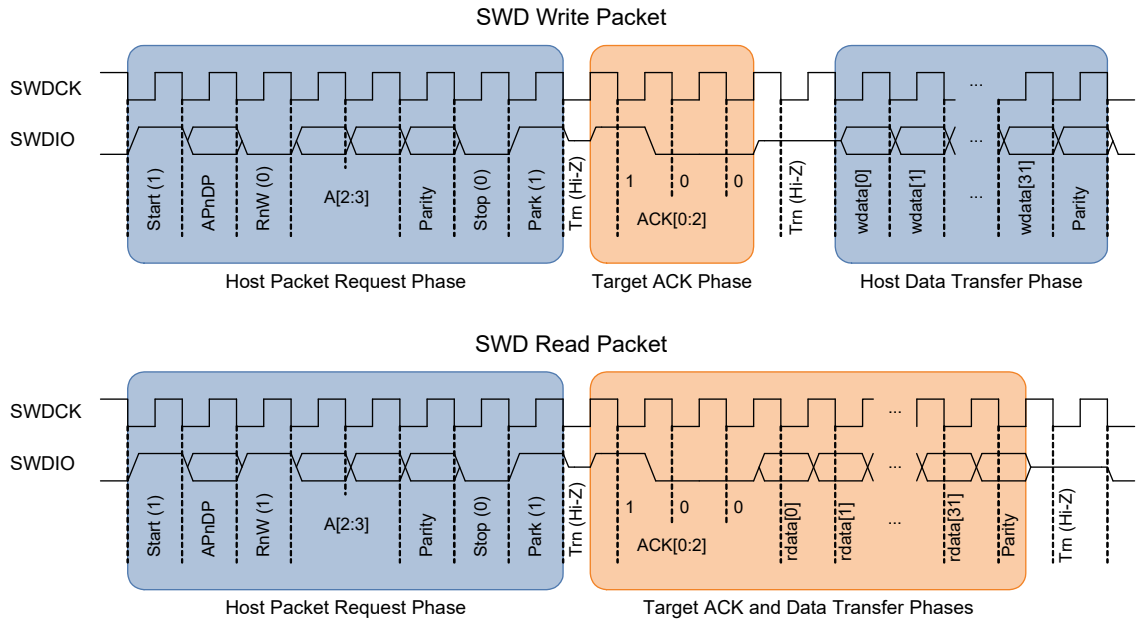
PSoC 4 的 Cortex-M0 支持通过 SWD 接口进行的编程和调试。 SWD 协议是基于数据包的串行数据操作协议。在引脚等级上，它使用一个双向数据信号 （SWDIO）和一个单向时钟信号 （SWDCK）。主机编程器始终驱动时钟信号线，而主机或目标器件将驱动数据线。完整的数据传输 （一个 SWD 数据包）需要 46 个时钟周期，并包括三个阶段：

- **主机数据包请求阶段** — 主机向 PSoC4 目标器件发送一个请求。
- **目标器件确认响应阶段** — PSoC4 目标器件给主机发送一个确认信息。
- **数据传输阶段** — 根据传输方向，主机或目标器件将数据写入到总线上。

当 SWDIO 线的控制权从主机转交给目标器件 （或反过来转交）时，在一个反转周期 （Trn）内，没有任何器件驱动该信号线，它会浮动于高阻态 （Hi-Z）。根据转交情况，该周期可能等于 0.5 或 1.5 个时钟周期。

图 26-2 显示了读取和写入 SWD 数据包的时序框图。

图 26-2. SWD 写入和读取数据包的时序框图



下面介绍的是 SWD 读取和写入数据包的传输序列：

1. 主机数据包请求阶段：SWDIO 由主机驱动
   a. 起始位启动传输；其状态始终为逻辑 1。
   b. “AP not DP” （APnDP）位确定传输类型是 AP 访问 （1b1）还是 DP 访问 （ 1b0）。
   c. “Read not Write” 位 （RnW）控制着数据传输的方向。 1b1 表示从目标器件 ‘ 读取 ’ ， 1b0 表示 ‘ 写入 ’ 目标器件。
   d. 地址位（A[3:2]）是 AP 或 DP（取决于 APnDP 位值）的寄存器选择位。请参见表 26-3 和表 26-4，了解相关的定义。**注意**：地址位的最低有效位 （LSB）先被传送。
   e. 奇偶校验位包含 APnDP、RnW 和 ADDR 位的奇偶校验。它是一个偶数校验位，这意味着，当将该位与其他位进行 XOR 运算时，得到的结果将为 ‘0’ 。
      如果奇偶位不正确，PSoC 4 将忽略包头；这样，将不会有 ACK 响应 （ACK = 3b111）。应该中止该编程操作，并通过下面的器件复位过程再尝试进行编程。
   f. 停止位始终为逻辑 0。
   g. 停留位始终为逻辑 1。
2. 目标器件确认响应阶段：SWDIO 由目标器件驱动
   a. ACK[2:0] 位表示从目标器件到主机的响应，并指出响应结果是成功还是失败。请参见表 26-1，了解相关的定义。
      注意：ACK 位的最低有效位 （LSB）先被传送。
3. 数据传输阶段：SWDIO 由目标器件或主机驱动 （取决于传输方向）
   a. 将需要读取或写入的数据写入到总线内 （先写入最低有效位 （LSB））。
   b. 数据的奇偶校验位指示需要读取或写入的数据的奇偶情况。它是一个偶校验位，这意味着，当将该位与数据位进行 XOR 运算时，得到的结果将为 0。

如果奇偶校验位指出一个数据错误，那么，需要对它进行纠正。对于一个读取数据包，如果主机检测到一个奇偶错误，它必须中止编程过程，然后重新启动该操作。对于写入数据包，如果目标检测到奇偶错误，它将在下一个数据包中生成 FAULT ACK 响应。

根据 SWD 协议，在 SWDIO 的状态为低电平时，主机可以在某两个数据包间生成不限定的 SWDCK 时钟周期数。如果时钟不能自由运行，或要求时钟在空闲模式下自由运行，则建议在某两个SWD数据包传输之间生成三个或更多的虚拟时钟周期。

SWDIO 的状态为高电平时，在 50 个或更多的时钟周期内给 SWDCK 线提供时钟脉冲，可以复位 SWD 接口。要想返回空闲状态，只要在 SWDIO 为低电平时给 SWDCK 线提供一个周期的脉冲。

26.3.1 SWD 时序的详细信息

根据通信的方向，SWDIO 线的写入和读取时间会有所不同。主机在主机数据包请求阶段内驱动 SWDIO 线。如果主机向目标写入数据，它也会在数据传输阶段内驱动 SWDIO 线。当主机驱动 SWDIO 线时，它会在 SWDCK 的下降沿上写入新的位，同时，目标会在 SWDCK 的上升沿上读取该位。在目标器件应答响应阶段期间，目标器件会驱动 SWDIO 线。如果目标器件正在读取数据，则它也会在数据传输阶段期间驱动 SWDIO 线。当目标器件驱动 SWDIO 线时，它会在 SWDCK 的上升沿上写入新位，同时，主机会在 SWDCK 的下降沿上读取该位。

表26-1和图26-2介绍了SWDIO位的写入和读取的时序情况。

表 26-1. SWDIO 位写入和读取时序

SWD 数据包相位	SWDIO 边沿	
	下降	上升
主机数据包请求	主机写入	目标读取
主机数据传输		
目标确认响应	主机读取	目标写入
目标数据传输		

26.3.2 ACK 的详细信息

应答 （ACK）位字段用于通知上一次传输的状态。 OK ACK 表示上一次数据包传输已成功。一个WAIT响应要求一个数据阶段。FAULT 状态表示需要立即中止编程过程。表 26-2 显示的是 ACK 位字段解码的详细信息。

表 26-2. SWD 传输确认响应解码

响应	ACK[2:0]
OK	3b001
WAIT	3b010
FAULT	3b100
NO ACK	3b111

下面介绍的是 WAIT 和 FAULT 响应行为的详情：

- 对于 WAIT 响应，如果数据操作为读取，主机应在数据阶段内忽略数据的读取。目标不驱动该线，所以主机也不需检查奇偶位。
- 对于 WAIT 响应，如果数据操作为写入，PSoC 4 会忽略数据相位。但是，主机仍要发送所需写入的数据，以完成数据包。同时，主机还要发送与该数据相应的奇偶位。
- WAIT响应表示PSoC 4正在处理前一数据操作。主机最多可以尝试四个连续的 WAIT 响应，以确定是否接收到 OK 响应。如果失败，那么需要中止编程操作，并重新启动该操作。
- 对于 FAULT 响应，也需要中止编程操作，然后通过器件复位重新启动该操作。

26.3.3 反转 （Trn）周期的详细信息

数据包请求阶段和 ACK 阶段间有一个反转周期； ACK 阶段和数据阶段 （主机写入传输情况下）间也有一个反转周期，如图 26-2 所示。根据 SWD 协议，主机和目标器件都使用 Trn 周期来更改其相应 SWDIO 线的驱动模式。在数据包请求阶段结束后的第一个 Trn 周期内，目标器件在 SWDCK 上升沿到来时开始驱动 SWDIO 线上的 ACK 数据。这样确保主机能在下一个下降沿上读取 ACK 数据。因此，第一个 Trn 周期的长度仅为半个时钟周期。SWD 数据包的第二个 Trn 周期等于一个半时钟周期。在 Trn 周期内，不建议使用主机和 PSoC 4 驱动 SWDIO 线。

26.4 Cortex-M0 调试和访问端口 （DAP）

Cortex-M0 编程和调试接口包括一个调试端口 （DP）和一个访问端口 （AP）。这两个端口组合起来，形成了 DAP 端口。调试端口将执行用于使能与主机通信的 SWD 接口协议的状态机。它还包含用于配置访问端口、DAP 标识代码等的寄存器。访问端口包含多个寄存器，允许外部器件能够通过 DAP-AHB 接口访问 Cortex-M0。通常，DP 寄存器用于一次性配置或错误检测，AP 寄存器用于执行编程和调试操作。有关 DAP 的完整架构的详细信息，请参阅 *ARM® 调试接口 v5 架构规格*。

26.4.1 调试端口 （DP）寄存器

表 26-3 显示的是用于编程和调试的 Cortex-M0 DP 寄存器及其相应的 SWD 地址位选择。对于 DP 寄存器访问，APnDP 位始终为 ‘0’。两个地址位 （A[3:2]）用于选择不同的 DP 寄存器。请注意，根据访问操作是读取操作还是写入操作，可以通过同样的地址位访问不同的 DP 寄存器。更多有关所有 DP 寄存器的详细信息，请参阅 *ARM® 调试接口 v5 架构规格*。

表 26-3. 主要调试端口 （DP）寄存器

寄存器	APnDP	地址 A[3:2]	RnW	全名	寄存器功能
ABORT	0 （DP）	2b00	0 （W）	AP 中止寄存器	该寄存器用于强制中止一个 DAP，以及清除错误和粘滞标志条件。
IDCODE	0 （DP）	2b00	1 （R）	标识代码寄存器	该寄存器保存 Cortex-M0 CPU 的 SWD ID （0x0BB11477）。
CTRL/STAT	0 （DP）	2b01	X （R/W）	控制和状态寄存器	该寄存器用于控制 DP，并包含有关 DP 的状态信息。
SELECT	0 （DP）	2b10	0 （W）	AP 选择寄存器	该寄存器用于选择当前的 AP。在 PSoC 4 中，只有一个与 DAP AHB 相连的 AP。
RDBUFF	0 （DP）	2b11	1 （R）	读取缓冲区寄存器	该寄存器保存最后一次 AP 读取操作的结果。

26.4.2 访问端口 （AP）寄存器

表 26-4 介绍了用于编程和调试的主要 Cortex-M0 AP 寄存器及其相应的 SWD 地址位选择。对于 AP 寄存器访问，APnDP 位始终为 ‘1’。两个地址位 （A[3:2]）用于选择不同的 AP 寄存器。

表 26-4. 主要访问端口 （AP）寄存器

寄存器	APnDP	地址 A[3:2]	RnW	全名	寄存器功能
CSW	1 （AP）	2b00	X （R/W）	控制和状态字寄存器 （CSW）	通过该寄存器，可以配置并控制通过存储器访问端口对已连接的存储器系统 （即为 PSoC 4 存储器映像）进行的访问
TAR	1 （AP）	2b01	X （R/W）	传输地址寄存器	该寄存器用于指定需要读取或写入的 32 位存储器地址
DRW	1 （AP）	2b11	X （R/W）	数据读 / 写寄存器	该寄存器保存需要读取或写入 TAR 寄存器所指定的地址的 32 位数据

26.5 编程 PSoC 4 器件

按照下面的顺序，可以对 PSoC 4 进行编程。请参阅 PSoC 4 BLE 编程规范，了解编程时所需要的编程算法、时序规范、硬件配置的详细信息。

1. 获取 PSoC 4 中的 SWD 端口。
2. 进入编程模式。
3. 执行器件编程子程序，如芯片 ID 检查、闪存编程、闪存验证以及校验和验证。

26.5.1 获取 SWD 端口

26.5.1.1 主要和辅助 SWD 引脚对

编程器件的第一步是获取 PSoC 4 中的 SWD 端口。有关 SWD 引脚的信息，请参考 器件数据手册 中的内容。

如果在器件中的两对 SWD 引脚均可用，则通过监控闪存区中的 SWD_CONFIG 寄存器，可以选择两对 SWD 引脚中的一对用于编程和调试。请注意，每次编程或调试会话期间，只能使用一对 SWD 引脚。器件的默认工厂设置是主要的 SWD 引脚对。如要选用辅助的 SWD 引脚对，则需要编程器件，使之同时使用主要引脚对和配置用于使能辅助引脚对的十六进制文件。这样，就能使用辅助 SWD 引脚对。

26.5.1.2 SWD 端口获取序列

编程器件的第一步是为主机获取目标的 SWD 端口。主机先通过激活外部复位（XRES）引脚执行器件复位。移除 XRES 信号后，主机必须在获取窗口中向器件发送一个 SWD 连接序列，以连接至 DAP 中的 SWD 接口。下面提供了该序列的伪代码。

代码 1. SWD 端口获取伪代码

```
ToggleXRES(); // Toggle XRES pin to reset device

//Execute ARM's connection sequence to acquire SWD-port
do
{
	SWD_LineReset(); //perform a line reset (50+ SWDCK clocks with SWDIO high)
	ack = Read_DAP ( IDCODE, out ID); //Read the IDCODE DP register

}while ((ack != OK) && time_elapsed < 1.5 ms); //retry connection until OK ACK or timeout

if (time_elapsed >= 1.5 ms) return FAIL; //check for acquire time out

if (ID != CM0_ID) return FAIL; //confirm SWD ID of Cortex-M0 CPU. (0x0BB11477)
```

在伪代码中， SWD_LineReset() 是用于复位调试访问端口的标准 ARM 命令。它使用超过 49 个 SWDCK 时钟周期，并且还需要 SWDIO 处于高电平状态。必须发送了至少一个 SWDCK 时钟周期，且将 SWDIO 置位为低电平来完成数据操作。该序列将对编程器和芯片进行同步化。Read_DAP() 指的是对调试端口中 IDCODE 寄存器进行的读取操作。需要重复线复位序列和 IDCODE 的读取操作，直至接收到 IDCODE 读取操作的 OK ACK 或发生超时 （1.5 ms）为止。如果在时间窗口中接收到 OK ACK，并且 IDCODE 的读取操作与 Cortex-M0 DAP 的相应读取操作相匹配，这样才能获取 SWD 端口。

26.5.2 SWD 编程模式入口

获取 SWD 端口后，主机必须在特定的时间窗口中进入器件编程模式。在测试模式控制寄存器 （MODE 寄存器）下设置 TEST_MODE 位（位 31）便能完成该操作。进入器件编程模式前，还要配置调试端口。有关进入编程模式的时序规范和伪代码的信息，请参考 PSoC 4 BLE 编程规范文档。

26.5.3 SWD 编程子程序执行

当器件进入编程模式时，外部编程器可启动发送 SWD 数据包序列，以执行编程操作，如擦除闪存、编程闪存、校验和验证，等等。第 335 页上的非易失性存储器编程章节中介绍了编程子程序。有关调用编程程序的正确序列的信息，请参考 PSoC 4 器件编程规范、PSoC 4-BLE 编程规范、PSoC 4100M、PSoC 4200M、PSoC 4200D、PSoC 4400 和 PSoC 4000S 器件编程规范。

26.6 PSoC 4 SWD 调试接口

Cortex-M0 DAP 拥有两类调试功能，包括：侵入性调试和非侵入性调试。侵入性调试包括中止编程和逐步编程，断点以及数据观察点。非侵入性调试包括指令地址配置和器件存储器访问。这些存储器包含闪存存储器、 SRAM 和其他外设寄存器。

DAP 有三个主要的调试子系统：

- 调试控制和配置寄存器
- 断点单元 （BPU） — 提供断点支持
- 调试观察点 （DWT）— 提供观察点支持。 Cortex-M0 调试中不支持跟踪功能。

请参阅 *ARMv6-M 架构参考手册*，了解有关调试架构的详细信息。

26.6.1 调试控制和配置寄存器

调试控制和配置寄存器用于执行固件的调试。下面介绍的是寄存器及其主要功能。请参阅 *ARMv6-M 架构参考手册* ，了解有关这些寄存器的完整位级定义。

- 调试中止控制和状态寄存器 （CM0_DHCSR）— 该寄存器包含用于使能调试、中止 CPU、单步调试等操作的控制位。它还包含处理器调试状态的状态位。
- 调试故障状态寄存器 （CM0_DFSR）— 该寄存器用于说明引起调试事件的原因。它包括由 CPU 停止、断点事件或观察点事件引起的调试事件。
- 调试内核寄存器选择器寄存器 （CM0_DCRSR）— 通过使用该寄存器，可以在 Cortex-M0 CPU 中选择必须由外部调试器读取或写入的通用寄存器。
- 调试内核寄存器数据寄存器 （CM0_DCRDR）— 该寄存器用于存储对寄存器 （由 CM0_DCRSR 寄存器选择）进行读取或写入操作的数据。
- 调试异常和监控控制寄存器 （CM0_DEMCR）— 该寄存器包含各个使能位，分别用于全局调试观察点（DWT）模块使能、复位向量捕捉和硬故障异常捕捉。

26.6.2 断点单元 （BPU）

BPU 提供了提取指令时的断点功能。PSoC 4 中的 Cortex-M0 DAP 支持多达四个硬件断点。除了硬件断点外，通过 Cortex-M0 中的 BKPT 指令，可以创建任意软件断点的数量。BPU 具有两类寄存器。

- 断点控制寄存器 （CM0_BP_CTRL）用于使能 BPU 和存储受调试系统支持的硬件断点数量 （对于 PSoC 4 中的 CM0 DAP，该数量为 4）。
- 每个硬件断点具有一个断点比较寄存器（CM0_BP_COMPx）。它包含断点的使能位，比较地址值以及触发断点调试事件的匹配条件。该寄存器的典型使用情况是：当某个指令提取地址与断点的比较地址相互匹配时，将生成一个断点，并中止处理器。

26.6.3 数据观察点 （DWT）

DWT 为数据地址访问或编程计数器 （PC）指令地址提供观察点支持。 PSoC 4 中的 Cortex-M0 不支持跟踪功能。DWT 支持两个观察点。它还通过 PC 样本寄存器提供外部编程计数器采样。该样本寄存器用于编程计数器的非侵入粗调配置。下面将介绍 DWT 中的最重要寄存器。

- 观察点比较（CM0_DWT_COMPx）寄存器存储了观察点比较器所使用的比较值，以生成观察点事件。每个观察点均有相应的 DWT_COMPx 寄存器。
- 观察点掩码 （CM0_DWT_MASKx）寄存器存储了忽略掩码，这些忽略掩码适用于相关观察点中的地址范围匹配。
- 观察点功能（CM0_DWT_FUNCTIONx）寄存器存储了触发观察点事件的条件。这些事件可以是编程计数器观察点事件，也可以是数据地址读取 / 写入访问观察点事件。发生相应的观察点事件时，状态位将被置位。
- 观察点比较器 PC 采样寄存器（CM0_DWT_PCSR）存储了编程计数器的当前值。该寄存器用于对编程计数器寄存器进行的粗调、非侵入性配置。

26.6.4 调试 PSoC 4 器件

通过访问调试控制和配置寄存器、BPU 中的寄存器以及 DWT 中的寄存器，主机可以调试 PSoC 4 目标器件。通过 SWD 接口对所有寄存器进行访问； Cortex-M0 DAP 中的 SWD 调试端口 （SW-DP）将 SWD 数据包转换为通过 DAP-AHB 接口进行的适当寄存器访问。

调试 PSoC 4 目标器件的第一步是获取 SWD 端口。获取序列包括 SWD 线复位序列和通过 SWD 接口进行的 DAP SWDID 读取操作。从目标器件读取到正确的 CM0 DAP SWDID 时，可以获取 SWD 端口。如果要使调试操作发生在 SWD 接口上，请勿将相应的引脚用于其他任何目的。请参考 第 69 页上的 I/O 系统章节以了解如何配置 SWD 端口引脚上的各位，从而决定将这些引脚仅使用于 SWD 接口，还是允许将它们使用于其他功能，如 LCD 和 GPIO。如果需要调试，请勿将 SWD 端口引脚使用于其他目的。如果只需要编程，则可以将 SWD 引脚使用于其他目的。

获取了 SWD 端口后，外部调试器将通过置位 DHCSR 寄存器中的 C_DEBUGEN 位来使能调试操作。然后，通过对调试系统中的相应寄存器进行写操作，可以执行不同的调试操作，如逐步调试、中止、断点配置以及观察点配置。

对目标器件的调试过程还受整个器件的保护设置的影响，如 第 115 页上的器件安全性章节中所述。因此，只有 OPEN（打开）保护模式支持器件调试。此外，当器件进入休眠或停止模式时，外部调试器会与目标器件断连。器件进入活动模式后，必须恢复该连接。当器件从活动模式转换为睡眠或深度睡眠模式时，外部调试器与目标器件间的连接不被断开。当器件从深度睡眠模式或睡眠模式恢复到活动模式时，调试器无需再次启动连接序列，就能恢复其操作。

26.7 寄存器

表 26-5. 寄存器列表

寄存器名称	说明
CM0_DHCSR	调试停止控制和状态寄存器
CM0_DFSR	调试故障状态寄存器
CM0_DCRSR	调试内核寄存器选型寄存器
CM0_DCRDR	调试内核寄存器数据寄存器
CM0_DEMCR	调试异常和监控控制寄存器
CM0_BP_CTRL	断点控制寄存器
CM0_BP_COMPx	断点比较寄存器
CM0_DWT_COMPx	观察点比较寄存器
CM0_DWT_MASKx	观察点掩码寄存器
CM0_DWT_FUNCTIONx	观察点功能寄存器
CM0_DWT_PCSR	观察点比较器 PC 样本寄存器

27. 非易失性存储器编程

非易失性存储器编程指的是对 PSoC® 4 器件中的闪存存储器进行编程。本章节介绍了属于器件编程的各项功能，如擦除、写入、编程和校验和计算等。用户可借助赛普拉斯公司提供的编程器或其它第三方编程器，使用这些功能将应用的十六进制文件的数据写入到 PSoC 4 器件中。此外，还可以使用这些功能来执行引导加载（bootload）操作，使 CPU 更新部分闪存存储器。

27.1 特性

- 支持通过调试和访问端口 （DAP）及 Cortex-M0 CPU 进行编程
- 支持由 Cortex-M0 CPU 执行的阻塞和非阻塞的闪存编程和擦除操作

27.2 功能说明

闪存编程相关的所有操作均是通过调用系统函数来实现的。在特权操作模式下在 SROM 外执行这些系统调用。用户无权读取或修改 SROM 代码。 DAP 或 CM0 CPU 通过将函数操作码和函数参数写入系统性能控制器接口 （SPCIF）输入寄存器内，然后请求 SROM 执行函数来调用系统函数。根据函数操作码，系统性能控制器 （SPC）将执行 SROM 中的相应系统调用，并更新 SPCIF 状态寄存器。DAP 或 CPU 将读取该状态寄存器，以便获得函数执行成功 / 失败的结果。执行函数时，SROM 中的代码与 SPCIF 接口交互，以进行实际的闪存编程操作。

使用编程 - 擦除 - 编程 （PEP）序列编程 PSoC 4 闪存。先将所有闪存单元编程为已知状态，然后擦除它们，再对选定的位进行编程。该序列可使存储电荷得到平衡，从而延长闪存的使用寿命。写入闪存时需要进行两个操作：先将数据复制到页锁存缓冲区中，然后使用闪存写函数将该数据传输到闪存内。

通过使用 SWD 协议将各命令发送到调试和访问端口 （DAP）中，外部编程器可以编程 PSoC 4 中的闪存存储器。有关带有外部编程器的 PSoC 4 器件的编程序列，请参考 PSoC 4-BLE 器件编程规范中介绍的内容。通过 AHB 接口访问相关寄存器，CM0 CPU 也可以对闪存存储器进行编程。这类编程在引导加载操作期间通常用于更新闪存存储器的部分，或其他应用要求，如更新存储在闪存存储器内的查找表。 DAP 或 CPU 对闪存存储器进行的所有写操作都是通过 SPCIF 实现。

注意：写入闪存的操作需要 20 ms。在这段时间内不应该复位器件，否则部分闪存会发生意外更改。复位源 （请参考第 111 页上的复位系统章节）包括 XRES 引脚、软件复位以及看门狗；需要确保不会意外激活这些复位源。另外，需要配置低电压检测电路，使它生成中断而不是复位。

注意：PSoC 4 含有一个用户监控闪存 （SFlash），该闪存可用于存储应用特定的信息。这些行不属于十六进制文件，可以选择是否编程这些行。

27.3 系统调用实现

一个系统调用函数包括以下部分：

- 操作码：一个唯一的 8 位操作码
- 参数：所有系统调用必须有两个 8 位参数。这些参数被称为 key1 和 key2，具体如下定义：

  key1 = 0xB6

  key2 = 0xD3 + 操作码

  这两个参数均被传送，以确保不会无意中启动用户系统调用函数。如果参数 key1 和 key2 不正确， SROM 将不会执行函数并返回错误代码。除了这两个参数以外，根据被调用的特定函数，可能还需要其它参数。
- 返回值：完成执行某些系统调用后，将返回一个值，如芯片 ID 或一个校验和。
- 完成状态：每个系统调用将返回一个 CPU 或 DAP 可读取的 32 位状态，以验证成功或确定失败的原因。

27.4 阻塞和非阻塞的系统调用

根据执行性质，系统调用函数可以划分为阻塞或非阻塞。执行阻塞系统调用时， CPU 不能执行任何其它任务。从某个过程调用阻塞系统函数时，CPU 会跳转到 SROM 中相应的代码位置。执行完成后，将恢复原始线程执行。非阻塞系统调用允许 CPU 同时执行其它代码。另外，通过一个中断向 CPU 报告中间系统调用已经完成。

只在 CPU 启动系统调用时，才能使用非阻塞系统调用。在编程模式下，DAP 仅使用系统调用，CPU 则在该过程中停止。

三种非阻塞系统调用分别为非阻塞行写入、非阻塞行编程以及恢复非阻塞。所有其它系统调用都是阻塞系统调用。

由于 CPU 在对闪存进行擦除或编程时不能执行闪存中的代码，因此只能通过在 SRAM 外执行的代码才能调用非阻塞系统调用函数。如果从闪存存储器内调用非阻塞函数，那么执行闪存提取时，将返回未定义结果，同时发生总线错误，并且触发一个硬故障。

系统性能控制器 （SPC）模块能够生成擦除和编程闪存存储器时所需的正确序列高电压脉冲。从 SRAM 调用非阻塞函数时，完成写入或编程操作中的每个子操作后， SPC 定时器将触发相应的中断。从 SPC 中断服务子程序 （ISR）中调用恢复非阻塞函数，以确保能够完成系统调用中的后续步骤。执行非阻塞写入或编程操作时，CPU 只能执行 SRAM 中的代码，因此应将 SPC ISR 放置在 SRAM 内。调用非阻塞编程函数时，SPC 中断将被触发一次；进行非阻塞写操作时，SPC 中断将被触发三次。执行非阻塞编程操作时， SPC ISR 中的恢复非阻塞函数被调用一次，而执行非阻塞写操作时，该函数将被调用三次。

本章后面的内容将介绍用于非阻塞写系统调用以及在 SRAM 外执行用户代码的伪代码。

27.4.1 执行系统调用

下面介绍了启动一个系统调用的流程：

1. 设置函数参数：输入函数参数（key1、key2、其它参数）的两种方法如下：
   a. 将函数参数写入到 CPUSS_SYSARG 寄存器内：该方法用于从 CPUSS_SYSARG 寄存器读取函数的参数。必须根据相应的系统调用表所指定的序列将各个参数写入到 32 位 CPUSS_SYSARG 寄存器内。
   b. 将函数参数写入到 SRAM 内：该方法用于从 SRAM 读取函数参数。首先，应该按照所指定的序列将这些参数写入到 SRAM 的连续地址内。然后，将 SRAM 的起始地址（即第一个参数的地址）写入到 CPUSS_SYSARG 寄存器内。该起始地址应该为字对齐 （32 位）地址。系统调用通过该地址可以获得参数。
2. 通过使用系统调用的操作码来指定系统调用并启动它：将 8 位操作码写入到 CPUSS_SYSREQ 寄存器的 SYSCALL_COMMAND 位（[15:0]）内。这个操作码被放置在低八位 [7:0] 内， 0x00 被写入到高八位 [15:8] 内。为了启动该系统调用，需要设置 CPUSS_SYSREG 寄存器中的 SYSCALL_REQ 位 （31）。设置该位会触发一个不可屏蔽的中断，该中断使 CPU 跳转到操作码参数参照的 SROM 代码。
3. 等待系统调用完成执行：系统调用开始执行时，它将设置 CPUSS_SYSREQ 寄存器中的 PRIVILEGED 位。只有通过系统调用才能设置该位，CPU或DAP不能设置它。DAP 应该连续轮询 CPUSS_SYSREG 寄存器中的 PRIVILEGED 位和 SYSCALL_REQ 位，以检查是否已经完成了系统调用。系统调用完成时，这两位均被清除。执行时间最多占用 1 秒。如果超过 1 秒钟这两位未被清除，则该操作被视为失败并被中止，而不再进行后面的步骤。请注意，CPU 应用代码在执行系统调用期间不能轮询这些位，这一点与 DAP 不同。这是因为在系统调用期间， CPU 在 SROM 外执行代码。从 SROM 返回执行结果后，应用代码只能检查最终函数成功 / 失败状态。
4. 检查完成状态：清除 PRIVILEGED 位和 SYSCALL_REQ 位以指出系统调用已完成后，需要通过读取 CPUSS_SYSARG 寄存器来检查系统调用的状态。若从 CPUSS_SYSARG 寄存器上读取的 32 位值为 0xAXXXXXXX （其中 ‘X’ 表示无需关注的十六进制值），那么系统调用操作已成功。如果系统调用失败，则状态代码将为 0xF00000YY，其中 ‘YY’ 表示失败的原因。有关状态代码的完整列表和说明，请查看表 27-1 中的内容。
5. 检索返回值：对于返回值为芯片 ID 和校验和等数值的系统调用， CPU 或 DAP 需要读取 CPUSS_SYSREG 和 CPUSS_SYSARG 寄存器来提取所返回的值。

27.5 系统调用

表 27-1 列出了 PSoC 4 所支持的所有系统调用、其功能描述以及在各个器件保护模式中的可用性。有关器件保护设置的更多信息，请参考第 115 页上的器件安全性章节中的内容。请注意， CPU 不能调用某些系统调用，如下表所示。随后是各个系统调用的详细内容。

表 27-1. 系统调用列表

系统调用	说明	DAP 访问			CPU 访问
		开放	保护	锁定	
芯片 ID	返回器件的芯片 ID、系列 ID 和修订 ID	✔	✔	–	✔
加载闪存字节	将数据加载到页锁存缓冲区内，以便稍后编程到闪存行中，以 1 字节为单位	✔	–	–	✔
行写入	擦除闪存行，然后将页锁存缓冲区中的数据编程到该闪存行内	✔	–	–	✔
行编程	将页锁存缓冲区中的数据编程到闪存行内	✔	–	–	✔
全部擦除	擦除闪存阵列中的所有用户代码和监控闪存区中的闪存行级保护数据	✔	–	–	
校验和	对整个闪存存储器 （用户和监控区）或单个闪存行进行计算校验和	✔	✔	–	✔
写保护	通过该系统调用可以将闪存行级保护设置和芯片级保护设置编程到监控闪存区 （行 0）内	✔	✔	–	
非阻塞行写入	擦除闪存行，然后将页锁存缓冲区中的数据编程到该闪存行内。在编程 / 擦除脉冲期间，用户可以执行 SRAM 中的代码。该函数仅适用于 CPU 访问	–	–	–	✔
非阻塞行编程	将页锁存缓冲区中的数据编程到闪存行内。在编程 / 擦除脉冲期间，用户可以执行 SRAM 中的代码。该函数仅适用于 CPU 访问	–	–	–	✔
恢复非阻塞	恢复非阻塞行写入或非阻行塞编程。该函数仅适用于 CPU 访问	–	–	–	✔

27.5.1 芯片 ID

该函数 返回一个 12 位系列 ID、一个 16 位芯片 ID、一个 8 位版本 ID 和当前的器件保护模式。这些值被返回到 CPUSS_SYSARG 和 CPUSS_SYSREQ 寄存器内。各参数被传送到 CPUSS_SYSARG 和 CPUSS_SYSREQ 寄存器内。

参数

地址	要写入的 'c' 值	说明
CPUSS_SYSARG 寄存器		
位 [7:0]	0xB6	Key1
位 [15:8]	0xD3	Key2
位 [31:16]	0x0000	不使用
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0000	芯片 ID 操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [7:0]	芯片 ID 低位	有关不同器件型号的芯片 ID 值，请查看器件数据手册。
位 [15:8]	芯片 ID 高位	
位 [19:16]	次版本 ID	请参见 PSoC 4-BLE 编程规范，了解这些值
位 [23:20]	主版本 ID	
位 [27:24]	0xXX	未使用（无需关注）
位 [31:28]	0xA	成功状态代码
CPUSS_SYSREQ 寄存器		
位 [11:0]	系列 ID	对于 PSoC 4200 BLE 和 PSoC 4100 BLE 系列，系列 ID 是 0x09E。
位 [15:12]	芯片保护	请参见第 115 页上的器件安全性章节
位 [31:16]	0xXXXX	不使用

27.5.2 配置时钟

该函数用于初始化执行闪存编程和擦除时所需的时钟。该 API 用以确保在调用闪存写入和闪存擦除 API 函数前，已经将电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）设置为 48 MHz 的 IMO。如果 IMO 是电荷泵时钟源，且它的频率不是 48 MHz，则闪存写入和闪存擦除 API 对闪存不起任何作用，并且返回 " 无效泵时钟频率 " 状态。

请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xE8	Key2
位 [31:16]	0xXXXX	无需关注
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数（key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0015	配置时钟操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.3 加载闪存字节

通过使用该函数 可以将要编程为一个闪存行的数据加载到页锁存缓冲区内。加载大小的范围介于 1 字节到闪存行中的最大字节数，即是 128 个字节 （对于 41x7-BL 和 42x7-BL 器件）或 256 个字节 （对于 PSoC 41x8-BL 和 PSoC 42x8-BL 器件）。数据被加载到页锁存缓冲区内，开始于 “Byte Addr” 输入参数所指定的地址。加载到页锁存缓冲区内的数据被保持，直到执行编程操作为止，该操作会清除页锁存内容。该函数的参数 （包括加载到页锁存缓冲区内的数据）将被写入 SRAM 内； SRAM 数据的起始地址被写入 CPUSS_SYSARG 寄存器内。请注意，参数的起始地址应该是字对齐的地址。

参数

地址	要写入的值	说明
SRAM 地址 — 32’hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xD7	Key2
位 [23:16]	字节地址	页锁存缓冲区中写入数据的起始地址 0x00 — 锁存缓冲区的字节 0 0x7F — 锁存缓冲区的字节 127 （适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件） 0xFF — 锁存缓冲区的字节 255 （适用于 PSoC 41x8-BL 和 PSoC 42x8-BL 器件）
位 [31:24]	闪存宏选择	0x00 — 闪存宏 0 0x01 — 闪存宏 1 （有关器件中的闪存宏数量，请参考第 37 页上的 Cortex-M0 CPU 章节）
SRAM 地址 — 32’hYY + 0x04		
位 [7:0]	加载大小	要写入到页锁存缓冲区内的字节数量。 0x00 — 1 个字节 0x7F — 字节 128（适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件） 0xFF — 字节 256 （适用于 PSoC 41x8-BL 和 PSoC 42x8-BL 器件）
位 [15:8]	0xXX	无需关注的参数
位 [23:16]	0xXX	无需关注的参数
位 [31:24]	0xXX	无需关注的参数
SRAM 地址 — 从 （32’hYY + 0x08）到 （32’hYY + 0x08 + 加载大小）		
字节 0	数据字节 [0]	要加载的第一个数据字节
.	.	.
.	.	.
字节 （加载大小 – 1）	数据字节 [加载大小 – 1]	要加载的最后数据字节
CPUSS_SYSARG 寄存器		

地址	要写入的值	说明
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数（key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0004	加载闪存字节操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.4 行写入

该函数先擦除寻址闪存行的内容，然后将页锁存缓冲区中的数据编程到该闪存行内。如果页锁存缓冲区中的所有数据都为‘0’，将跳过编程操作。该函数的参数被存储在 SRAM 内。存储参数的起始地址被写入到 CPUSS_SYSARG 寄存器内。行被编程后，该函数将清除页锁存缓冲区的内容。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟 （clk_pump）和 HF 时钟 （clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。调用该函数前，先调用加载闪存字节函数。只有相应的闪存行不受写保护时，该函数才能执行写操作。

请注意，在执行闪存写操作前，该系统调用会禁用 36 MHz IMO 输出。可以使用 36 MHz IMO 输出为模拟开关泵或 CTBm 泵提供时钟源。如果使用 36 MHz IMO 输出，在系统调用完成后，必须手动重新使能它。特别是要对 CLK_IMO_CONFIG EN_CLK36 和 FLASHPUMP_SEL 进行复位。

更多有关信息，请参考 PSoC 4100-BL/4200-BL 系列：PSoC 4 BLE 寄存器技术参考手册中 CLK_IMO_CONFIG 寄存器的内容。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xD8	Key2
位 [31:16]	行 ID	要写入的行号 0x0000 — 行 0
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数 （key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0005	写入行的操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.5 行编程

该函数用于将页锁存缓冲区中的数据编程到寻址闪存行内。如果页锁存缓冲区中的所有数据都为 ‘0’，将跳过编程操作。调用该函数前，必须擦除这一行。在该行被编程后，页锁存缓冲区的内容将被清除。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。调用该函数前，先调用加载闪存字节函数。调用该函数前，必须擦除这一行。只在相应的闪存行不受写保护的情况下，该函数才能执行编程操作。

参数

地址	要写入的值	说明
SRAM 地址：32’hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xD9	Key2
位 [31:16]	行 ID	要编程的行号 0x0000 — 行 0
CPUSS_SYSARG 寄存器		
位 [31:0]	32’hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数 （key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0006	编程行操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.6 全部擦除

通过该函数可以擦除闪存主阵列中的所有用户代码和每个闪存宏的监控闪存行 0 中的行级保护数据。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。只在芯片保护模式为 OPEN （开放），且 DAP 处于编程模式时，才能从 DAP 调用该 API。如果芯片保护模式为 PROTECTED （保护）， DAP 必须使用写保护 API 将保护设置改为 OPEN（打开）。将保护设置从 PROTECTED 修改为 OPEN 时，会自动执行全部擦除操作。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xDD	Key2
位 [31:16]	0xXXXX	无需关注
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数 （key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x000A	擦除所有操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.7 校验和

通过使用该函数可以读取整个闪存存储器或一个闪存行，并返回在闪存区中读取到的每个字节的 24 位总和。在整个闪存上执行校验和时，也对用户代码和监控闪存区进行校验和。在某个闪存行上执行校验和时，闪存行数量将被作为一个参数传入。通过参数的字节 2 和字节 3，可以选择在整个闪存存储器还是在一个用户代码闪存行上执行校验和。

参数

地址	要写入的值	说明
CPUSS_SYSARG 寄存器		
位 [7:0]	0xB6	Key1
位 [15:8]	0xDE	Key2
位 [31:16]	行 ID	选择执行校验和操作的闪存行号 行号 — 16 位闪存行号 或 0x8000 — 对整个闪存存储器进行校验和
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x000B	校验和操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:24]	0xX	未使用 （无需关注）
位 [23:0]	校验和	选定的闪存区中 24 位校验和的值

27.5.8 写保护

通过该函数可以对监控闪存行中的闪存行级保护设置和器件保护设置进行编程。可以独立编程器件中每个闪存宏的闪存行级保护设置。每行具有一个保护位。保护字节的总数等于闪存行的数量除以 8。芯片级保护设置 （1 字节）被存储在监控闪存行 0 中最后字节地址上的闪存宏 0 内。监控闪存行的大小等于用户代码闪存行的大小。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。通过使用加载闪存字节函数，可以将闪存宏的闪存保护字节加载到相应的页锁存缓冲区内。加载函数的起始地址参数应该为 ‘0’ 。闪存宏的数量是需要编程的数量；要加载的字节数量是该宏中闪存保护字节的数量。

然后调用写保护函数，从而将页锁存缓冲区中的闪存保护字节编程为相应闪存宏的监控行。在存储器件保护设置的闪存宏 0 内，器件级保护设置作为参数，并被传送到 CPUSS_SYSARG 寄存器内。

参数

地址	要写入的值	说明
CPUSS_SYSARG 寄存器		
位 [7:0]	0xB6	Key1
位 [15:8]	0xE0	Key2
位 [23:16]	器件保护字节	仅适用于闪存宏 0 的参数 0x01 — OPEN （开放）模式 0x02 — PROTECTED （保护）模式 0x04 — KILL （锁定）模式
位 [31:24]	闪存宏选择	0x00 — 闪存宏 0 0x01 — 闪存宏 1
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x000D	写保护操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:24]	0xX	未使用 （无需关注）
位 [23:0]	0x000000	

27.5.9 非阻塞行写入

CM0 CPU 使用非阻塞方式写入闪存行时，该函数将被使用。这样可以确保在执行写操作时，CPU 能够执行 SRAM 中的代码。有关非阻塞系统调用的说明，请参考第 336 页上的阻塞和非阻塞的系统调用 。

非阻塞行写入的系统调用具有三个阶段：预编程、擦除、编程。在预编程的步骤中，闪存行中的所有位都被写入 ‘1’，以准备执行擦除操作。擦除操作将清除该行中的所有位，然后通过编程操作将新数据写到该行内。

当执行每个阶段时， CPU 可以执行 SRAM 中的代码。当启动非阻塞行写入的系统调用时，除了用于完成非阻塞写操作的恢复非阻塞函数外，用户不能调用任何其它系统调用函数。完成每个阶段后，SPC 都会触发中断。发生该中断时，将调用恢复非阻塞系统函数。

注意：在进行非阻塞行写入操作期间，固件绝不能使器件进入睡眠模式。这样做将复位页锁存缓冲区，并会将所有零值写入到闪存内。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。调用该函数之前，先调用用加载闪存字节函数，旨在加载用于编程行的数据字节。另外，只能调用 SRAM 中的非阻塞行写入函数。这是因为执行闪存擦除编程操作时， CM0 CPU 不能执行闪存中的代码。如果从闪存存储器调用该函数，执行闪存提取操作时，结果为未定义并会返回一个总线错误，而且会触发一个硬故障。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xDA	Key2
位 [31:16]	行 ID	要写入的行号 0x0000 — 行 0
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数 （key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0007	非阻塞行写入的操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.10 非阻塞行编程

CM0 CPU 使用非阻塞方式编程闪存行时，该函数将被使用。这样可以确保在执行编程操作时， CPU 能够执行 SRAM 中的代码。有关非阻塞系统调用的说明，请参考第 336 页上的阻塞和非阻塞的系统调用 。执行编程操作时， CPU 可以执行 SRAM 中的代码。当调用非阻塞行编程的系统函数时，除了用于完成非阻塞写操作的恢复非阻塞函数外，用户不能调用任何其它系统函数。

与非阻塞行写入的系统调用函数不同，编程系统调用只有一个阶段。因此，当调用非阻塞行编程的系统函数时，仅需要从 SPC 中断调用恢复非阻塞函数一次。

使用要求：在调用该函数前，先调用配置时钟 API。通过配置时钟 API，可确保电荷泵时钟（clk_pump）和 HF 时钟（clk_hf）均被设为频率为 48 MHz 的 IMO。请注意： 该函数仅适用于 PSoC 41x7-BL 和 PSoC 42x7-BL 器件。调用该函数之前，先调

用用加载闪存字节函数，旨在加载用于编程行的数据字节。另外，只能从 SRAM 调用非阻塞行编程函数。这是因为执行闪存编程操作时， CM0 CPU 不能执行闪存中的代码。如果从闪存存储器调用该函数，执行闪存提取操作时，结果将为未定义并会返回一个总线错误，而且将触发一个硬故障。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xDB	Key2
位 [31:16]	行 ID	要写入的行号 0x0000 — 行 0
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数（key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0008	非阻塞行编程的操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.5.11 恢复非阻塞

使用该函数可以完成其它擦除和编程阶段。这些阶段通过使用非阻塞行写入和非阻塞行编程的系统调用来启动。调用非阻塞行写入后，需要从 SPC ISR 调用该函数三次，调用非阻塞行编程后，则需要从 SPC ISR 调用该函数一次。完成编程或擦除操作的所有阶段前，不能执行其它系统调用。有关使用非阻塞函数的流程的更多信息，请参考第 336 页上的阻塞和非阻塞的系统调用 。

参数

地址	要写入的值	说明
SRAM 地址：32'hYY （32 位宽、字对齐的 SRAM 地址）		
位 [7:0]	0xB6	Key1
位 [15:8]	0xDC	Key2
位 [31:16]	0xXXXX	无需关注。 SROM 不使用这些位
CPUSS_SYSARG 寄存器		
位 [31:0]	32'hYY	SRAM 的 32 位字对齐地址，用于存储第一个函数参数 （key1）
CPUSS_SYSREQ 寄存器		
位 [15:0]	0x0009	恢复非阻塞操作码
位 [31:16]	0x8000	设置 SYSCALL_REQ 位

返回值

地址	返回值	说明
CPUSS_SYSARG 寄存器		
位 [31:28]	0xA	成功状态代码
位 [27:0]	0xXXXXXXX	未使用 （无需关注）

27.6 系统调用状态

每个系统调用结束后，都会将一个状态代码覆写为 CPUSS_SYSARG 寄存器中的参数。成功状态的代码为 0xAXXXXXXX，其中 ‘X’ 表示无需关注的值或返回数据 （如果系统调用返回某一值）。失败状态的代码为 0xF00000XX，其中 ‘XX’ 表示失败的代码。

表 27-2. 系统调用的状态代码

状态代码 （CPUSS_SYSARG 寄存器内的 32 位值）	说明
AXXXXXXXh	成功 — “X” 表示无需关注的值。除非 API 直接向 CPUSS_SYSARG 寄存器返回参数，否则 SROM 会返回 ‘0’ 。
F0000001h	无效的芯片保护模式 — 在当前的芯片保护模式下，不能使用该 API。
F0000003h	无效的页锁存地址 — 表示页锁存缓冲区超出了边界，或者所提供的地址大小大于页地址。
F0000004h	无效地址 — 所提供的行 ID 或字节地址不处于可用的存储器范围内。
F0000005h	受保护的行 — 所提供的行 ID 是一个受保护的行。
F0000007h	恢复过程已完成 — 所有非阻塞 API 操作已完成。执行下一个非阻塞 API 前，不能调用恢复 API。
F0000008h	挂起恢复 — 启动了一个非阻塞 API。在调用任何其它 API 之前，必须通过调用一个恢复 API 完成该操作。
F0000009h	正在进行系统调用 — 正在调用一个恢复 API 或非阻塞 API。在尝试进行下一个恢复过程之前，必须触发 SPC ISR。
F000000Ah	零校验和失败 — 所计算的校验和不是零。
F000000Bh	无效的操作码 — 操作码不是一个有效的 API 操作码。
F000000Ch	关键操作码失配 — 所提供的操作码与 key1 和 key2 不匹配。
F000000Eh	无效的起始地址 — 起始地址大于所提供的结束地址。
F0000012h	无效的泵时钟频率 — 在对闪存进行写入 / 擦除操作前，必须将 IMO 设为 48 MHz，HF 时钟源设为 IMO 时钟源。

27.7 非阻塞系统调用伪代码

本节提供的伪代码演示了如何设置一个非阻塞的系统调用并在进行闪存编程操作期间在 SRAM 外执行代码。

```
#define REG(addr)          (*((volatile uint32 *) (addr)))
#define CM0_ISER_REG            REG( 0xE000E100 )
#define CPUSS_CONFIG_REG           REG( 0x40100000 )
#define CPUSS_SYSREQ_REG           REG( 0x40100004 )
#define CPUSS_SYSARG_REG           REG( 0x40100008 )

/* Note: Use the right macro according to your device */
#define ROW_SIZE_128      (128)
#define ROW_SIZE_256      (256)
#define ROW_SIZE        (ROW_SIZE_128)

/*Variable to keep track of how many times SPC ISR is triggered */
__ram int iStatusInt = 0x00;

__flash int main(void)
{
  DoUserStuff();

  /*CM0 interrupt enable bit for spc interrupt enable */
  CM0_ISER_REG |= 0x00000040;

  /*Set CPUSS_CONFIG.VECS_IN_RAM because SPC ISR should be in SRAM */
  CPUSS_CONFIG_REG |= 0x00000001;

  /*Call non-blocking write row API */
  NonBlockingWriteRow();

  /*End Program */
  while(1);
}
__sram void SpcIntHandler(void)
{
  /* Write key1, key2 parameters to SRAM */
  REG( 0x20000000 ) = 0x0000DCB6;

  /*Write the address of key1 to the CPUSS_SYSARG reg */
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x09 to the CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000009;

  /* Number of times the ISR has triggered */
  iStatusInt ++;
}
__sram void NonBlockingWriteRow(void)
{
  int iter;

  /*Load the Flash page latch with data to write*/
  * Write key1, key2, byte address, and macro sel parameters to SRAM
  */
  REG( 0x20000000 ) = 0x0000D7B6;
```

```
  /* Write load size param (128/256 bytes) to SRAM */
  #if (ROW_SIZE == ROW_SIZE_128)
     REG( 0x20000004 ) = 0x0000007F;
  #elif (ROW_SIZE == ROW_SIZE_256)
     REG( 0x20000004 ) = 0x000000FF;
  #endif  /* #if (ROW_SIZE == ROW_SIZE_128) */

  for(i = 0; i < ROW_SIZE/4; i += 1)
  {
     REG( 0x20000008 + i*4 ) = 0xDADADADA;
  }

  /*Write the address of the key1 param to CPUSS_SYSARG reg*/
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x04 to CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000004;

  /*Perform Non-Blocking Write Row on Row 200 as an example.
  * Write key1, key2, row id to SRAM row id = 0xC8 -> which is row 200
  */
  REG( 0x20000000 ) = 0x00C8DAB6;

  /*Write the address of the key1 param to CPUSS_SYSARG reg */
  CPUSS_SYSARG_REG = 0x20000000;

  /*Write the API opcode = 0x07 to CPUSS_SYSREQ.COMMAND
  * register and assert the sysreq bit
  */
  CPUSS_SYSREQ_REG = 0x80000007;

  /*Execute user code until iStatusInt equals 3 to signify
  * 3 SPC interrupts have happened. This should be 1 in case
  * of non-blocking program System Call
  */
  while( iStatusInt != 0x03 )
  {
     DoOtherUserStuff();
  }

  /* Get the success or failure status of System Call*/
  syscall_status = CPUSS_SYSARG_REG;
}
```

在代码中，通过将 0x01 写入 CPUSS_CONFIG 寄存器内，可以将 CM0 异常表配置为处于 SRAM 内。 SRAM 异常表将 SPC 中断的向量地址作为定义在 SRAM 内的 *SpcIntHandler()* 函数使用。有关将 CM0 异常表配置在 SRAM 内的详细信息，请参考第 57 页上的中断章节的内容。非阻塞编程系统调用的伪代码是相同的，但函数操作码和各个参数不一样，且 iStatusInt 变量应被轮询 1 次，而不是 3 次。这是因为执行非阻塞编程系统调用时，只能触发 SPC ISR 一次。

术语表

术语表这一节介绍了本技术参考手册所使用的术语。术语表中的所有术语以***粗斜体字***显示。

A

累加器	位于 CPU 中并用于存储中间结果的寄存器。如果没有累加器，则需要将各项计算（加、减、移位等等）的结果写入到主存储器内，然后读回它们。由于累加器通常具有算术和逻辑单元 （ALU）的直接输入 / 输出路径，所以访问主存储器的速度总是比访问累加器的速度慢。
高电平有效信号	1. 它是一种逻辑信号，其激活状态为逻辑 1。 2. 它是一种逻辑信号，逻辑 1 状态作为两种状态中较高的电压状态。
低电平有效信号	1. 它是一种逻辑信号，它的激活状态为逻辑 0。 2. 它是一种逻辑信号，逻辑 1 状态作为两种状态中较低的电压状态：反转逻辑。
地址	确定存储器位置（RAM、ROM 或寄存器）的标签或编号，在该位置上存储了一单位大小的信息。
算法	通过限定步骤解决数学问题的程序，这些步骤通常涉及到一个重复操作。
环境温度	一个指定区域的空气温度，特别是 PSoC 器件周围的区域。
模拟	请参见***模拟信号***。
模拟模块	是基本的可编程运算放大器电路。它们是 SC （切换电容）和 CT （连续时间）模块。这些模块内部互联，能够提供 ADC、 DAC、多极滤波器、放大器等多种功能。
模拟输出	该输出可驱动各个电源范围之间的任何电压，而不仅是逻辑 1 或逻辑 0 的电压水平。
模拟信号	在连续时间内连续变化的信号。相反，数字信号是在连续时间内分离变化的信号。
模数转换器 （ADC）	是将模拟信号转换为相应量级的数字信号的器件。通常， ADC 可以将某个电压值转换成一个数字值。*数模转换器 （DAC）*可以用于执行逆向操作。
AND	请参见*布尔代数*。
API （应用编程接口）	一系列的软件程序，包括计算机应用与低层服务和函数之间的接口 （例如，用户模块和库） 。应用编程接口 （API）用作程序员创建软件应用使用的基本模块。

数组	数组，也称为向量或列表，是电脑编程中最简单的数据结构之一。数组存储数目固定、大小相等并且通常属于同一数据类型的数据元素。与关联数组不同，可通过使用一组连续的整数索引对单个元素进行访问。大多数高级编程语言都将数组作为内置数据类型使用。某些数组是多维的，也就是说它们是由固定数量的整数索引，例如，包含两个整数的组。可是，最常见的数组是一维和二维数组。另外，数组还可以是使用几个通用形式进行连接的电容或电阻的组合。
汇编	一个特定处理器的机器语言的符号表示法。通过汇编程序可将汇编语言转换为机器代码。虽然通常使用宏，但每个汇编代码行一般产生一个机器指令。汇编语言被视为低层语言，相反， C 被视为高层语言。
异步	其数据被立即确认或作出响应的信号，与任何时钟信号无关。
衰减	信号的强度衰减。它是由信号在到达检测器时能量被吸收和分散而导致。它不包括由几何扩展的衰减。衰减的单位通常为分贝 （dB）。

B

带隙参考	一个稳定电压的参考设计将 V_T 温度正系数与 V_{BE} 温度负系数相互匹配，从而生成零温度系数（理想的）参考。
带宽	1. 指的是消息或信息处理系统的频率范围 （单位为 Hz）。 2. 放大器 （或吸收器）在其频谱区内会有大量增益 （或损失）。有时它表示的更为具体，例如半峰全宽。
偏置	1. 数值与参考值之间的系统偏差。 2. 一组数据的平均值偏离参考值的幅度。 3. 施加于器件的电力、机械力、磁场或其他力 （场），用于建立运行该器件所需的参考电平。
偏置电流	用于为放大器产生稳定操作的常量低电平直流电流。在某些情况下，可调整该电流值，以更改放大器的带宽。
二进制	是以 2 为基数的数字系统名称。现在，以 10 为基数的数字系统是最常用的数字系统。数字系统的基数表示系统中一个数字里的特定位置所存在的数字数量。例如，在二进制中，每个位置包含两个值 （0 或 1）的其中一个。在十进制数字系统中，每个位置包含十个值 （0、1、2、3、4、5、6、 7、 8 和 9）的其中一个。
位	二进制数系统中的单一数字。因此，一位仅能具有 ‘0’ 或 ‘1’ 值。每 8 个位组成一个字节。由于 PSoC M8CP 是一个 8 位的微控制器，因此 PSoC 器件的原始数据单元大小为一个字节。
比特率 （BR）	位流中每个时间单位内处理的位数，通常使用比特每秒 （bps）为单位。
模块	1. 用于执行单项功能的功能单元，如振荡器。 2. 按目标功能进行配置的功能单元，例如，数字 PSoC 模块或模拟 PSoC 模块。

	在数学和计算机科学中，布尔代数或布尔格是捕获了集合运算交集、并集、补集和逻辑运算 AND（与）、OR（或）和 NOT（非）的根本性质的一个代数结构。布尔代数也定义了如何操作布尔等式的一组定理。例如，通过这些定理可以简化布尔等式。这样可减少实现等式所需的逻辑元素数量。
布尔代数	布尔代数的运算符可以使用多种方式来表示。通常可以将它们简写为 AND、OR 和 NOT。在介绍电路时，也可以使用 NAND（NOT AND）、NOR（NOT OR）、XNOR（非 NOT OR）和 XOR（非 OR）等符号。对于 OR，数学家经常使用 ‘+’ 号（例如，A+B），而对于 AND，他们使用 • 号（例如，A*B）（在某些情况下，那些运算与代数运算中的加法和乘法类似）。另外，数学家通过在被否定的表达式上面划线来表示 NOT（例如，~A、A_、!A）。
先开后合	是指建立新的连接状态（“合”）前需要先进入断开状态（“开”）的因素。
广播网络	是指在整个微控制器中路由，并可由多个模块或系统访问的信号。
缓冲区	1. 它是用来补偿数据从一个器件传输至另一个器件时速度差的数据存储区。通常指保留用于 I/O 操作的区域，可在此区域内读取或写入数据。 2. 往往将数据发送到外部器件前或者从外部器件接收数据前，会留出一部分存储器空间用于存储数据。 3. 它是一个用于降低系统的输出阻抗的放大器。
总线	1. 某个命名的网络连接。将网络捆绑到总线中，便于使用类似的路由模式路由网络。 2. 用于执行通用功能并携带类似数据的一组信号。通常使用向量符号来表示。例如，地址 [7:0]。 3. 作为一组相关器件的通用连接的一个或多个导体。
字节	是包含 8 个位的数据存储单位。

C

C	是一种高层编程语言。
电容	电容是由平行的、中间有绝缘介质隔离的两片导体构成，用来衡量当电势差变化时保持电荷的能力。电容的测量单位为法拉（Farads）。
捕获	通过使用软件或硬件来提取信息，而不是将数据手工输入到计算器文件内。
链路	连接两个或两个以上的 8 位数字模块，以组成 16、24 乃至 32 位的功能。通过链路，一个模块可以为其他模块产生比较（Compare）、进位（Carry）、使能（Enable）、捕获（Capture）和门（Gate）等信号。
校验和	可通过将每个数据字添加到总和来生成一组数据的校验和。实际的校验和可以是结果之和，或者是要添加到总和以生成预定值的值。
清除	将某一位或某寄存器的值强制设置为逻辑 ‘0’。
时钟	是指生成具有固定频率和占空比的周期信号的器件。有时，时钟可以用来同步化各个不同的逻辑模块。
时钟生成器	用于生成时钟信号的电路。

CMOS	使用以互补方式连接的 *MOS* 晶体管构建的逻辑门。 CMOS 是互补金属氧化物半导体的缩略语。
比较器	指两个输入电平同时满足预定振幅要求时会生成输出电压或电流的电气电路。
编译器	将高级语言 （例如 C 语言）转换成机器语言的程序。
配置	在计算机系统中，功能性单元的安排取决于其性质、数量以及主要特性。该配置与硬件、软件、固件以及文档有关。它会影响系统的性能。
配置空间	在 PSoC 器件中，当 CPU_F 寄存器中的 XIO 位设置为 ‘1’ 时，可以访问寄存器空间。
crowbar	是一种过压保护的电路。当输出电压超出预定的电压值，这个电路会快速将低电阻（通常为 SCR）放在信号和电源轨范围内。
CPUSS	CPU 子系统
晶体振荡器	由压电晶体控制频率的振荡器。通常情况下，与其他电路组件相比，压电晶体对环境温度的灵敏度更低。
循环冗余校验 （CRC）	用于检测数据通信中的错误的计算方法，通常使用线性反馈移位寄存器执行。相似的计算方法可用于其他用途，如数据压缩。

D

数据总线	计算机使用以将信息从存储器位置传输到中央处理单元 （CPU）或反向传输信息的双向信号组。更为普遍的是，用来传送数字功能之间数据的信息组。
数据流	用于表示传输的信息的一串数字编码信号。
数据传输	使用通道的信号来将数据从一个位置发送到其他位置。
调试器	允许用户分析正在开发系统操作的软件和硬件系统。调试器通常允许开发人员逐步执行固件操作，设置断点及分析存储器。
死区	两个或多个信号都没有处于活动状态或切换状态的一段时间。
十进制	是以 10 为基数的数字系统。它使用符号 0、1、2、3、4、5、6、7、8 和 9（名为数字）、小数点以及 ‘+’ （加号）和 ‘-’ （减号）两个符号表示数字。
默认值	指的是系统所假定、使用或在没有用户指令的情况下执行的预定义初始、原始或特定的设置、条件、数值或行动。
器件	除非另有说明，否则本手册所提到的器件均是 PSoC 器件。
裸片	一个非封装集成电路 （IC），通常从晶圆切割。
数字	是一个信号或功能，它的幅值使用两个离散值 ‘0’ 或 ‘1’ 的其中一个来描述。
数字模块	可用作计数器、定时器、串行接收器、串行发送器、CRC 发生器、伪随机数发生器或 SPI 的 8 位逻辑模块。

数字逻辑	用于处理表达式的方法。表达式包含了说明电路或系统的行为二状态变量。
数模转换器 （DAC）	可将数字信号转换为相应量级的模拟信号的器件。*模数转换器 （ADC）*可以用来执行逆向操作。
直接访问	根据独立于它们相关位置的序列，使用表示数据的物理位置的地址来获取存储器件中的数据或将数据写入存储器件中的能力。
占空比	是时钟周期的*高电平时间*与其*低电平时间*的关系，表示为一个百分比。

E

外部复位 （XRES_N）	传入 PSoC 器件的高电平有效信号。这导致 CPU 和各个模块的所有操作都停止，并返回到预定义状态。

F

下降沿	从逻辑 1 到逻辑 0 的跃变。它也被称为负向沿。
反馈	将一个 （通常为活动）器件的输出部分或输出的处理部分返回到输入。
滤波器	信号的某些频率组件衰减的器件或过程。
固件	是指被嵌入到硬件器件并由 CPU 执行的软件。最终用户可以执行该软件，但不能修改它。
标志	确定条件或事件 （例如，一个字符通知传输操作终止）的任意指示器。
闪存	是可电编程和电擦除、*易失性*的技术，它为用户提供 EPROM 的可编程功能和数据存储，以及系统内可擦除功能。非易失性表示在断电时，数据仍被保留。
闪存组	是闪存模块在一个单一闪存组中以 ‘0’ 开始的一组闪存 ROM 模块。一个闪存组也有其自己的模块范围的保护信息。
闪存模块	能够通过单个编程操作进行编程的最小闪存 ROM 空间以及受保护的最小闪存空间。闪存模块容量为 64 个字节。
触发器	它是一个具有两种稳定状态和两个输入端 （或两种输入信号类型）的器件，一个输入端与一种状态相对应。电路处于任意一种状态，直到使用相应的信号使它切换为另一种状态。
频率	是指执行周期性功能时每个时间单位内的周期数或发生事件的次数。

G

增益	分别为输出电流、电压或功率与输入电流、电压或功率之间的比率。增益的单位通常使用分贝（dB）。

门	1. 对于具有一个输出通道和一个或多个输入通道的器件，除开关跃变外，输出通道状态完全取决于输入通道状态。 2. 多种组合逻辑元素的其中一个最少具有两个输入（例如，AND、OR、NAND 和 NOR（请参见*布尔代数*一节））。
接地	1. 与周围地面具有相同电位的电中性线。 2. 直流电源的负端。 3. 电气系统的参考点。 4. 电路或设备与地面或接触地面的某些导电体之间的导电路径。

H

硬件	它是一个使用于计算机或嵌入式系统所有物理部分的综合术语。硬件不同于其自身所包含的或操作的数据和软件 （为硬件提供完成任务所需的指令）。
硬件复位	是由电路触发的复位，例如 POR、看门狗复位或外部复位。硬件复位将器件返回到它首次被上电的状态。因此，所有寄存器均被设为 POR 值 （如本文档中的寄存器表所示）。
十六进制	它是一个十六进制数字系统 （通常被简写并称为 hex），通常使用数字 0 到 9 以及字母 A 到 F 来表示。由于很容易将四个位元映射为十六进制数字，它在计算机领域是一个很有用的系统。因此，用户可以使用两个连续的十六进制数字来表示每一个字节。二进制、十六进制和十进制表示法对比如下： bin = hex = dec 0000b = 0x0 = 0 0001b = 0x1 = 1 0010b = 0x2 = 2 ... 1001b = 0x9 = 9 1010b = 0xA = 10 1011b = 0xB = 11 ... 1111b = 0xF = 15 这样，十进制数字 79 的二进制表示法为 0100 1111b，它的十六进制表示法则为 4Fh （0x4F）。
高电平时间	针对一个周期性数字信号，信号在一个周期内具有 ‘1’ 值的时长。

I

I^2C	由飞利浦半导体公司 （现为 NXP 半导体）生产的两线式串行计算机总线。I^2C 是互联集成电路。它用于连接嵌入式系统中的低速外设。原始系统创建于 20 世纪 80 年代初期，当时只作为电池控制接口，但后来被用作构建控制电子器件的简单的内部总线系统。I^2C 仅使用两个双向引脚（时钟和数据引脚），两者均在 +5 V 的电压条件下运行并采用电阻上拉。在标准模式下，总线速率为 100 Kbps ；而在快速模式下，总线速率为 400 Kbps。

闲置状态	当用户消息不被传输时但可以立即使用服务的状态。
阻抗	1. 是电流上的阻力，该阻力由电路中的电阻、电容或电感产生。 2. 供给电流的总负阻抗。请注意，该阻抗是由已给电路中的电阻、感抗和容抗组合决定的。
输入	在器件、过程或通道中用于接收数据的一点。
输入 / 输出 （I/O）	是用于将数据引入到系统或从系统中提取数据的器件。
指令	它是在编程语言中 （例如 C 或汇编语言）指定一个操作并识别它的操作数 （若有）的表达方式。
指令助记符	是表示各汇编语言指令的操作码的一组缩略语 （例如， ADD、 SUBB、 MOV）。
集成电路 （IC）	它是将电阻、电容、二级管和*晶体管*等组件集成在半导体单一芯片表面的器件。
接口	使两个系统或器件连接或有相互作用的方式。
中断	流程暂停 （例如，执行计算机程序），由流程外部事件导致，并且暂停后可以恢复流程。
中断服务子程序 （ISR）	M8CP 收到硬件中断时常规代码执行转入的代码模块。可以存在多个中断源，每个中断源均有各自的优先级和单个 ISR 代码模块。每个 ISR 代码模块均以 RETI 指令结束，该指令将器件返回到离开常规程序执行的程序点。

J

抖动	1. 从它的理想位置跃变的时序错位。在串行数据流中出现的典型损坏。 2. 一个或多个信号特性的突发和意外变化，例如连续脉冲之间的间隔、连续周期的振幅或连续周期的频率或相位。

L

延迟	将一个信号传输到一个给定的电路或网络所需的时间或延迟。
最低有效位 （LSb）	在二进制数中表示最低有效值（通常为右边的位）的二进制数字或位。为了区分位和字节，在 LSb 中的位 （bit）上使用了小写字母 “b”。
最低有效字节 （LSB）	它是一个多字节中的字节，用于表示最低有效值 （通常为右边的字节） 。为了区分位和字节，在 LSB 中的字节 （byte）使用大写字母 “B”。
线性反馈移位寄存器 （LFSR）	它是一个移位寄存器，其数据输入是寄存器链中两个或两个以上元素经过 *XOR* 运算后得到的结果。
负载	操作过程所需求的电力，其表现形式为功耗 （瓦特）、电流 （安培）或电阻 （欧姆）。
逻辑函数	是一个对数字数据执行数字运算，然后返回一个数字值的数学函数。

查询表 （LUT）	执行若干逻辑函数的逻辑模块。通过使用选择线选中逻辑函数，并将它应用于模块的输入。例如：可以使用具有 4 个选择线的两引脚 LUT 对两个输入引脚上执行 16 个逻辑函数中的任意一个，从而产生一个单一的逻辑输出。LUT 是一个组合的器件，因此输入 / 输出之间是连续关系，即不被采样。
低电平时间	针对一个周期数字信号，在一个周期内信号值为 ‘0’ 的时长。
低压检测 （LVD）	是指在 V_{DDD} 降低到选定阈值以下时，可检测 V_{DDD} 并实现系统中断的电路。

M

M8CP	8 位 Harvard 架构微处理器。微处理器通过连接闪存、 SRAM 和寄存器空间来协调 PSoC 器件内部的所有活动。
宏	编程语言宏是一个抽象概念。它根据一系列预定义的规则替换一定的文本模式。当遇到宏的实例名称，解释器或编译器将自动使用宏内容来替换宏的实例名称。因此，如果使用了一个宏五次，且宏的定义需要 10 个字节的代码空间，则总共需要 50 个字节的代码空间。
掩码	1. 用于掩饰、隐藏信息、或防止信息从信号派生。掩码通常是与其他信号（例如噪声、静态、拥塞或其他干扰形式）相交互的结果。 2. 可使用位模式来保留或抑制计算和数据处理系统中其他位模式的段式。
主设备	用于控制两个器件间数据交换时序的器件。或者，以脉冲宽度级联器件时，主设备是用来控制级联器件与外部接口之间数据交换时序的器件。受控制的器件被称为*从设备*。
微控制器	主要用于控制系统和产品的集成电路器件。除 CPU 外，微控制器通常还包含存储器、定时电路和 I/O 电路。这是为了能够使用最小器件数目实现一个控制器，从而能实现最大程度的微型化。这样会降低控制器的体积和成本。微控制器通常不用于通用计算功能，这些功能由微处理器进行处理。
助记符	用于协助存储器的工具。助记符不仅通过重复，而且还通过创建易记结构和数据列表之间的关联来记住实际情况。一个具有两到四个字符的字符串，表示微处理器指令。
模式	软件或硬件的不同操作方式。例如，数字 PSoC 模块可以处于计数器模式或定时器模式。
调制	用于对载波信号 （通常为正弦波信号）上的信息进行编码的一系列技术。执行调制的器件被称为调制器。
调制器	在载波上附加信号的器件。
MOS	是金属氧化物半导体的缩略语。
最高有效位 （MSb）	在二进制数中表示最高有效值（通常为左边的位）的二进制数字或位。为了区分位和字节，在 MSb 字母中的位 （bit）使用小写字母 “b”。
最高有效字节 （MSB）	多字节中表示最高有效值的字节 （通常为左边的字节）。为了区分位和字节，在 MSB 字母中的字节 （byte）使用大写字母 “B”。

复用器 （mux）	1. 一个使用二进制值，或地址来选择多个输入信号中的一个，然后将选定输入信号的数据传递至输出信号中的逻辑功能。 2. 允许不同的输入（或输出）信号在不同的时间使用同样的线路，并由外部信号控制的技术。使用复用器可以节省线路和 I/O 端口数量。

N

NAND	请参见 *布尔代数*。
负向沿	从逻辑 1 到逻辑 0 的跃变。它也被称为下降沿。
网络	是器件之间的连接。
半字节	由四个位，就是一个字节的一半构成的组。
噪声	1. 指的是一种干扰，它会影响信号，并且可使信号携带的信息失真。 2. 如电压、电流或数据等任何实体中一种或多种特性发生的随机变化。
NOR	请参见 *布尔代数*。
NOT	请参见 *布尔代数*。

O

OR	请参见 *布尔代数*。
振荡器	可受晶控，并用于生成时钟频率的电路。
输出	电子信号或由模拟或数字模块生成的信号。

P

并行	是指数字数据每次能发送多个位，其中每个同步位通过一个独立的线路进行传送的通信方式。
参数	是已给模块的特性。这些特性已经被定性，或可由设计人员定义。
参数模块	是指存储器中执行 SSC 指令前用于存储参数的位置。
奇偶校验	用于测试传输数据的技术。通常，将一个二进制数字添加到数据中，以便求所有二进制数据奇数之和 （奇校验）或偶数之和 （偶校验）。
路径	1. 是由计算机执行的指令逻辑序列。 2. 电路中的电子信号流。
挂起中断	一个被触发但未得到处理的中断，这可能是因为处理器正在忙着处理其他中断，或全局中断被禁用。

相位	是两个信号 （通常为具有相同频率的信号）之间的关系。此关系决定信号之间的延迟。信号间的延迟可使用时间或角 （度）来测量。
引脚	是硬件组件上的终端。它也被称为接脚。
引脚分配	引脚号分配：PSoC 器件的逻辑输入和输出与它们在印刷电路板 （PCB）封装中相对物理位置之间的关系。引脚分配包括原理图与 PCB 设计 （两者均是计算机生成的文件）之间链接的引脚号，也包括引脚名称。
端口	一组引脚，通常有八个。
正向沿	从逻辑 0 到逻辑 1 的跃变。它也被称为上升沿。
发布的中断	一个由硬件检测，但可通过屏蔽位使能或禁用的中断。未屏蔽的发布中断变成了挂起中断。
上电复位 （POR）	当电压下降至预设电压以下时强迫 PSoC 器件复位的电路。这是一种*硬件复位*的类型。
程序计数器	指令指针（又称程序计数器）是电脑处理器中的寄存器，用于指出在 CPU 正在处理指令的存储位置。根据特定机器的详细内容，它会存储将被执行的指令地址，或将被执行的下一个指令地址。
协议	是一组规则。特别是控制网路通信的规则。
PSoC®	赛普拉斯的可编程片上系统 （PSoC®）器件。
PSoC 模块	*请参见 模拟模块*和*数字模块*。
PSoC Creator™	赛普拉斯的新一代可编程片上系统技术的软件。
脉冲	是指信号特性 （例如，相位或频率）的快速变化。它从基线的值变为更高或更低的值，然后快速返回到基线的值。
脉冲宽度调制器（PWM）	占空比形式表示的输出，它随着应用测量对象的不同而变化。

R

RAM	随机存取存储器的缩略语。数据存储器件，可以对该器件进行读写操作。
寄存器	具有特定容量 （例如一位或一个字节）的存储器件。
复位	使系统返回已知状态的方法。请参见*硬件复位*和*软件复位*。
电阻	导体的电流电阻，其单位为欧姆 （Ω）。
版本 ID	PSoC 器件的唯一标识符。
波纹分频器	是触发器所构成的一个异步波纹计数器。将时钟信号提供到计数器的第一段内。包含 n 个触发器的 n 位二进制计数器可以从 0 到 2^n - 1 进行计数的二进制值。
上升沿	*请参见 正向沿*。

ROM	只读存储器的缩略语。数据存储器件，可以对该器件进行读操作但无法进行写操作。
子程序	是一个代码模块。它由其他代码模块调用，另外，还具有通用或频繁使用方式。
布线	是根据参考库中的设计规则而物理上连接各对象。
短脉冲	在数字电路中，由于信号的非零上升沿和下降沿，窄脉冲未达到一个有效高电平或低电平。例如，在异步时钟间进行切换，或者是信号经过两个独立的路径输入同一个电路造成竞争状态时，将发生短脉冲。在这些竞争状态下可能有不同的延迟，然后形成毛刺或者此时一个触发器的输出成为亚稳定状态。

S

采样	指的是将模拟信号转换为一系列数字值或反转的过程。
原理图	它是用于详细描述一个系统中各元件（例如电子电路的元件或计算机中逻辑图的元素）的示意图、绘制图或草图。
种子值	是加载到线性反馈移位寄存器或随机数发生器中的初始值。
串行	1. 是指所有事件在其中连续发生的流程。 2. 表示在单个器件或通道中两个或多个相关活动的连续发生。
设置	将某一位或某一寄存器的值强制设置为逻辑 1。
建立时间	输入从一个值改为另一个值后，输出信号或值变为稳定状态需要的时长。
移位	是字位置中位的移位，可移到左边或右边。例如，如果十六进制值 0x24 向左边移位 1，它将成为 0x48。如果十六进制值 0x24 向右边移位 1，则它便成为 0x12。
移位寄存器	按顺序向左或向右转移一个字以便输出串行数据流的存储器件。
符号位	是带符号二进制数的最高有效二进制数字或位。如果将它设置为逻辑 1，该位将表示负值。
信号	用于传递信息的可测试传送能量。使用于各电子设备时，它是任何被传送的电脉冲。
芯片 ID	PSoC 芯片唯一的标识符。
时滞	是在并行发送中，同时传送的位到达目标的时间差别。
从设备	允许另一个器件控制两个器件之间数据交换的时序的器件。或者，以脉冲宽度级联器件时，从设备是允许另一个器件控制级联器件与外部接口之间数据交换的时序的器件。控制器件被称为主设备。
软件	是一系列有关数据处理系统操作的电脑程序、过程和相关文档的结合（例如，编译器、库子程序、手册和电路图）。通常，软件首先被编写为源代码，然后被转换为适合器件执行的二进制格式代码。

软件复位	是软件执行的部分复位，从而将部分系统返回已知的状态。软件复位时，则将 M8CP 恢复到一个已知的状态，而不是恢复 PSoC 模块、系统、外设或寄存器的状态。软件复位时，要将 CPU 寄存器（CPU_A、CPU_F、CPU_PC、CPU_SP 和 CPU_X）的值设为 0x00。因此，代码执行将从闪存地址 0x0000 开始。
SRAM	静态随机存取存储器的缩略语。可以高速存储和检索数据的存储器件。之所以使用术语 “ 静态 ”，是因为在将某一值加载到 SRAM 单元时，该值仍保持不变，直至它被明确更改，或直至器件断电为止。
SROM	只读管理存储器的缩略语。 SROM 存储用以引导器件、校准电路和执行闪存操作的代码。使用常规用户代码访问 SROM 函数，这些代码在闪存中运行。
堆栈	堆栈是按照后入先出 （LIFO）的原理运作的数据结构，即最后输入到堆栈的项目也是第一个被取出的项目。
堆栈指针	堆栈可能位于计算机中模块里的存储器单元内。它的底部被放在单元的一个固定位置上，而变量堆栈指针则被放在当前顶部单元中。
状态机	是一个功能的实际实现过程 （可由硬件或软件实现），它可以包括一组需要按序列进入的状态。
粘滞	它是指寄存器中的一位，该位能保持它的值，直到发生导致其转换的事件为止。
停止位	是特征或模块带有的信号，用于使接收器件准备好接收下一个特征或模块。
开关	电路上信号的控制或布线，用以执行逻辑或算术操作，或在网络中各个特定点之间传输数据。
开关相位	根据开关电容控制一个给定的开关（PHI1 或 PHI2）的时钟。PSoC 的 SC 模块具有两组开关。一组通常在 PHI1 运行期间被关闭，则在 PHI2 运行期间被打开。另一组在 PHI1 运行期间被打开，则在 PHI2 运行期间被关闭。在正常操作中，可以控制这些开关。如果 PHI1 和 PHI2 时钟被反转，则可在反转模式下控制它们。
同步	1. 指的是一个信号，其数据未被确认或做出响应，直到时钟信号的下一个边沿有效为止。 2. 是指其操作由时钟信号进行同步的系统。

T

抽头	是指器件中两个模块通过以串行方式来连接某些模块 / 组件（如移位寄存器或电阻电压分频器）的连接。
终端计数	计数器倒计至零的状态。
阈值	系统或传感器在考虑下可检测的信号的最低值。
Thumb-2	Thumb-2 是一组高效强大的指令集。从易用性、代码大小和性能的角度来说，它能够带来显著利益。除32位指令外，通过添加16位指令，Thumb-2指令集是先前的16位Thumb指令集的超级集合。
晶体管	晶体管是一个固态半导体器件，用于放大增益和切换。它具有三个终端：向一个终端提供的小电流或电压控制其它两个终端的电流。它是现代电器的重要组成部分。在数字电路中，可将晶体管做为快速电子开关使用；另外，合适的晶体管组合可作为逻辑门、 RAM 型存储器和其他器件使用。在模拟电路，基本上将晶体管作为放大器使用。

三态	指的是一种功能，其输出有三种状态：0、1 和 Z（高阻态）。该功能不会驱动 Z 状态下的任何值，在许多方面，可将其视为与其余电路断开，允许另一个输出驱动相同的*网络*。

U

UART	UART 或通用异步接收器 - 发送器在数据并行位和串行位之间进行转换。
用户	使用 PSoC 器件或阅读本手册的人。
用户模块	负责全面管理和配置低级模拟和数字 PSoC 模块的预构建、预测试硬件 / 固件外围功能。此外，用户模块还针对外设功能提供高级 *API （应用编程接口）*。
用户空间	寄存器映射的组 0 空间。在执行常规程序和初始化期间，很可能会对该组中的寄存器进行修改。程序初始化阶段，很可能对组 1 中的寄存器进行了修改。

V

V_{DDD}	电力网名称，意为 “ 电压漏极 ”。最正极的电源信号。电压通常为 5 V 或 3.3 V。
易失性	如果不属于合适的范围，便不能保证保持相同的值或电平。
V_{SS}	电源网络名称，意为 “ 电压源 ”。最负极的电源信号。

W

看门狗定时器	一个必须定期刷新的定时器。如果未定期刷新它，则 CPU 会在指定时间期间后复位。
波形	是一个信号的表示法，以振幅和时间图显示。

X

XOR	请参见 *布尔代数*。

索引

C

Cortex-M0
- 寄存器 39
- 指令集 40
- 特性 37

I

I/O 系统
- CapSense 80
- LCD 驱动能力 80
- 寄存器汇总 81
- 开漏模式 73
- 强驱动模式 73
- 模拟 I/O 80
- 特性 69
- 电阻模式 73
- 简介 69
- 转换速率控制 74

I/O 系统中的转换速率控制 74

I/O 驱动模式
- 开漏 73
- 强 73
- 数字高阻态 73
- 模拟高阻态 73
- 电阻 73

L

LCD 驱动
- I/O 系统能力 80

P

PSoC
- 活动模式 102
- 编程和调试 27

S

SAR ADC
- 说明 251

SWD 接口
- 编程和调试接口 328

Z

上电复位 111

介绍
- 复位 111
- 时钟生成 83

休眠唤醒复位 112

休眠模式 103

保护故障复位 112

修订记录 17

停止唤醒复位 112

内部主振荡器 84

内部低速振荡器 86

内部电压调节器 95

升级 29

复位
- 介绍 111
- 识别源 112

复位源
- 说明 111

外部复位 112

寄存器
- Cortex-M0 39

寄存器汇总
- I/O 系统 81

工作原理
- 看门狗定时器 108

开发套件 29

异常事件
- HardFault（硬故障） 60
- NMI 60
- PendSV 60
- SVCall 60
- SysTick 60
- 复位 59

按键和滑条创建的 GPIO 引脚 80

振荡器
- 内部 PSoC 84

支持服务 29

数字高阻态驱动模式 73

文档
- 修订记录 17
- 术语表 349

时钟分布 88

时钟源
- 分布 88

时钟系统
- 介绍 83

术语表 349
框图
    看门狗定时器电路 107
    编程和调试接口 327
概况，文档
    修订记录 17
模拟 I/O 80
模拟高阻态驱动模式 73
欠压复位 111
活动模式
    PSoC 102
特性
    I/O 系统 69
    看门狗定时器 107
电压调节器
    内部 95
看门狗复位 111
看门狗定时器
    中断 110
    使能 109
    工作原理 108
    工作模式 109
    特性 107
    禁用 109
睡眠模式 102
简介
    I/O 系统 69
系统调用
    概述 336
编程和调试
    PSoC 27
识别复位源 112
说明
    逐次逼近寄存器模数转换器 251
软件复位 112