前言

本参考手册面向应用开发人员，提供有关使用 SPC2168 微控制器储存器与外设的完整信息。

有关 SPC2168 器件的机械与电气特性，请参见该器件的数据手册。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
1	2019-07-09	-	Outdated	1. 初始版本。
2	2019-08-01	-	Outdated	1. 在章节 0 增加 ECAP 设计缺陷说明。
3	2019-08-15	-	Outdated	1. 更新图 23-1 和图 23-5。
4	2019-09-13	-	Outdated	1. 更新章节 1.3.2 中关于 SRAM 位置的描述以及相应的图。 2. 更新章节 3.4 中的 Note，时钟可在 GPIO11 被观测。 3. 在章节 10.2 中移除 TBSTS.CNTDIR 的错误声明。 4. 在章节 10.6 中更新 TZ0~TZ4 信号描述。 5. 在章节 10.7 中修复格式错误。 6. 修复表 10-85 中错误。
5	2019-10-10	-	Outdated	1. 更新表 4-3，重命名 PINMUXKEY 到 PINMUXREGKEY。 2. 更新表 4-5，修改 SMT 位段。 3. 增加表 4-6 和表 4-7。 4. 在章节 6.2.4 中移除 AESCTL1.RST。 5. 在章节 8.1 中更新 Timer 特性。 6. 在章节 9.1 中更新 WDT 特性。 7. 更新表 9-15 和表 9-16，重命名 WDTLOCK 到 WDTREGKEY。 8. 更新表 10-19，修改 DBGRUN 位段。 9. 更新表 10-133，修改 SOCA 位段。 10. 更新章节 0 中的 Note。 11. 更新表 11-20，修改 DBGRUN 位段。 12. 更新章节 12.9 中的上电时序要求。 13. 更新表 12-33，修改位段描述。 14. 更新表 12-55，修改 SHEN 位段。 15. 更新表 12-153，修改位段描述。 16. 在章节 13 中修改 DVDD 到 AVDD。 17. 在章节 13.4 中，将 PGAxCTL.INPOL 改为 PGAxCTL.CMSEL。 18. 在章节 13.7 中，移除 PGAxCTL.IBOOST。 19. 更新表 13-8 至表 13-19。 20. 在章节 17.4 中，修改 UARTIER.NRZE 为 UARTIER.NRZME。 21. 在章节 17.4.2 中，修改 UARTLSR.TXEMPTY 为 UARTLSR.TXDONE。

版本	日期	作者	状态	变更
				22. 在章节 17.4.3 中，移除表方法声明。 23. 更新表 17-26，修改 XMITIR 位段。 24. 更新表 18-67，表 18-77 和表 18-79，修改 I2CCTL_EN 为 I2CENABLE。 25. 更新表 19-1，修改 SFM 为 SFRM。 26. 在章节 19.4.3.2 中，修改 SSPCTL1.RXTOIE 为 SSPSTS.RXTOINT。 27. 更新章节 19.4.4.5 中的 Note，将 SS 改为 SFRM。 28. 修改章节 19.4.5 中的内容。 29. 更新表 19-7，将 SSP_CLK/SSP_FRM 改为 SSP_SCLK/SSP_SFRM。 30. 修改章节 20.3.1 中的内容。 31. 更新表 22-28，修改位段描述。
6	2019-11-28	-	Outdated	1. 更新表 23-12 和表 23-13，重命名 FIFODAV 为 FIFOEMPTY。
7	2020-11-21	-	Outdated	1. 增加表 1-2。 2. 修改章节 1.4 中内容。 3. 修改章节 4.3 中内容。 4. 更新表 10-89。 5. 更新图 12-2。 6. 修改章节 12.7 中内容。 7. 修改章节 12.10.2 中内容。 8. 更新章节 17.2 中 UART 特性。 9. 增加表 20-2。 10. 更新表 23-12。
8	2021-05-29	-	Outdated	1. 更新图 10-11。 2. 更新章节 23.1 和图 23-1。 3. 更新表 23-3。 4. 更新章节 3.9.11，修改 SIO 最高频率。 5. 更新图 10-11。 6. 增加表 10-5。 7. 更新图 10-17 和图 10-19。 8. 更新章节 1.3.2，修改 SRAM ECC 特性。 9. 更新表 10-19。 10. 增加章节 3.9.1。 11. 更新章节 3.9.2 和章节 3.9.4。 12. 更新章节 3.10。 13. 更新章节 19.4.3，为 SSP RXONLY 模式增加 Note。
9	2021-12-24	-	Outdated	1. 更新图 4-2。 2. 增加图 4-5。 3. 更新表 4-3 和表 4-5。 4. 更新表 10-55。 5. 更新章节 17.4.2.2，为接收中断增加 Note。

版本	日期	作者	状态	变更
				6. 更新章节 15.2。 7. 更新表 10-67 和表 10-69，修改位段 REDSRC 的描述。 8. 更新表 10-7，修改 TZSEL 寄存器的复位值。 9. 更新图 3-4 和图 3-5。 10. 更新章节 3.6。 11. 更新表 3-14 和表 3-15。 12. 更新表 17-27 和表 17-28，修改 UARTFOR.BYTECNT 位段宽度。 13. 更新表 14-1。 14. 更新图 10-4。 15. 更新章节 11，17 中的错别字。 16. 更新章节 10.6。 17. 更新章节 15.2。 18. 更新章节 17.4.5，删除 XMODE 相关描述。 19. 更新章节 16。 20. 更新图 4-6 和图 4-7。 21. 更新表 8-3 和表 8-4。 22. 更新章节 7.2。 23. 增加章节 25。 24. 更新章节 1.1。 25. 增加表 22-37 和表 22-38。 26. 更新表 22-39。
10	2022-07-05	-	Outdated	1. 增加图 4-1，图 4-3 和图 4-4。 2. 更新图 1-2。 3. 增加章节 13.9。 4. 增加章节 14.5 和章节 15.4。 5. 增加章节 16.3。 6. 更新章节 12.11。 7. 更新章节 12.7.2。 8. 更新章节 10.6 中信号封锁子模块结构的描述。 9. 更新示例说明在章节 15.2。
C/0	2024-09-11	Jiali Zhou	Released	1. 更正 DAC 和 buffer 数量。 2. 修改图 1-4、图 1-5。 3. 更正表 19-15 VAL 位段描述。 4. 修改图 10-2、图 10-3。 5. 修改图 15-2。 6. 修改图 13-4、图 13-5。 7. 更正图 10-14 TZSEL 的错误位段。 8. 修改图 4-1。 9. 修改 UART 发送中断产生逻辑的相关描述。 10. 修改表 19-7 以及章节 19.4.3.2。 11. 修改图 15-2。

版本	日期	作者	状态	变更
				12. 修改图 14-1 中 PWMCLK 为 ADCCLK，并同步修改相关描述。 13. 章节 20.5.1 添加注意事项：分区保护功能不包括 DRAM。 14. 修改表 17-14 DMAE 点位描述。

1 存储器和总线架构

1.1 系统架构

图 1-1：SPC2168 系统框图

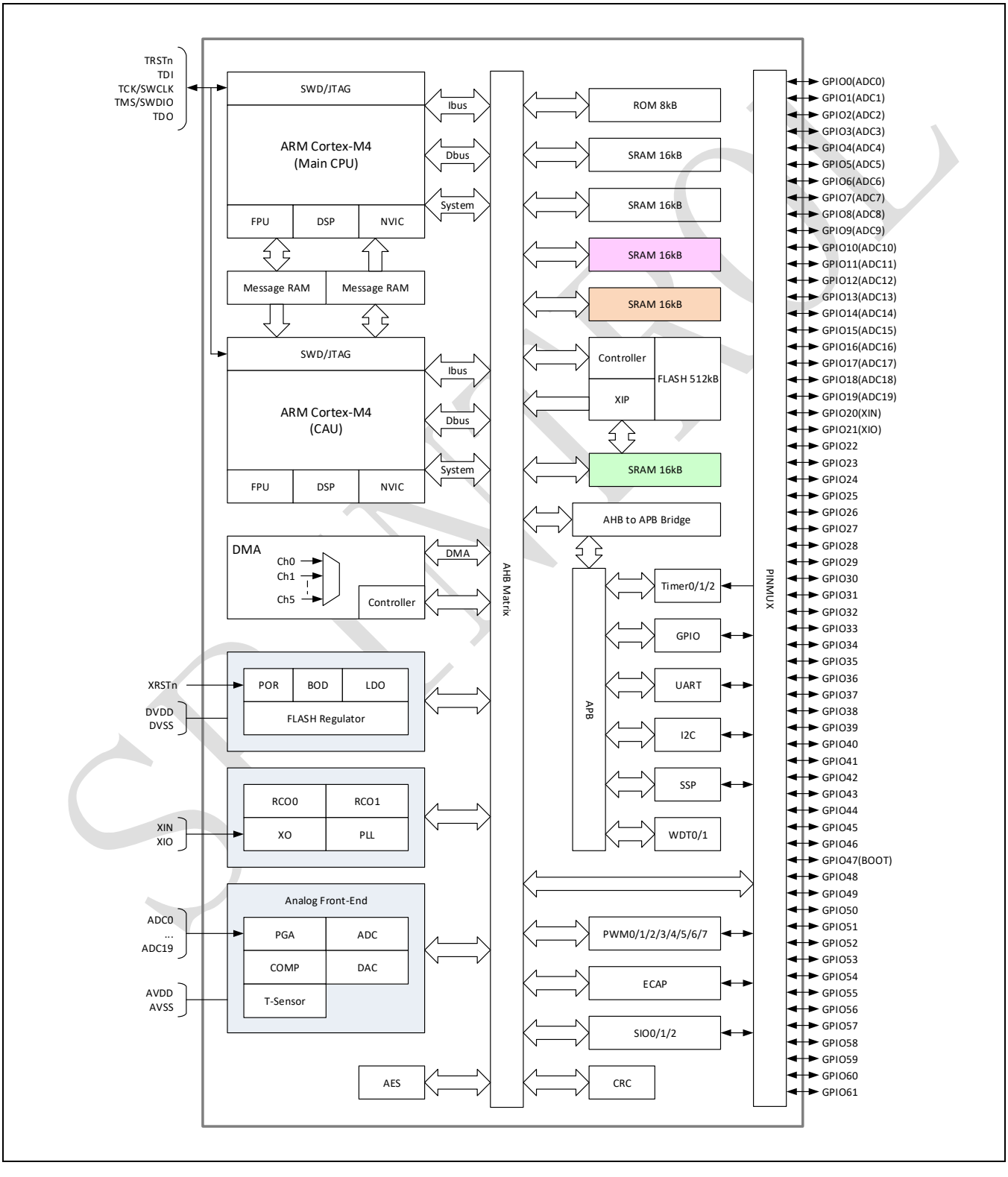
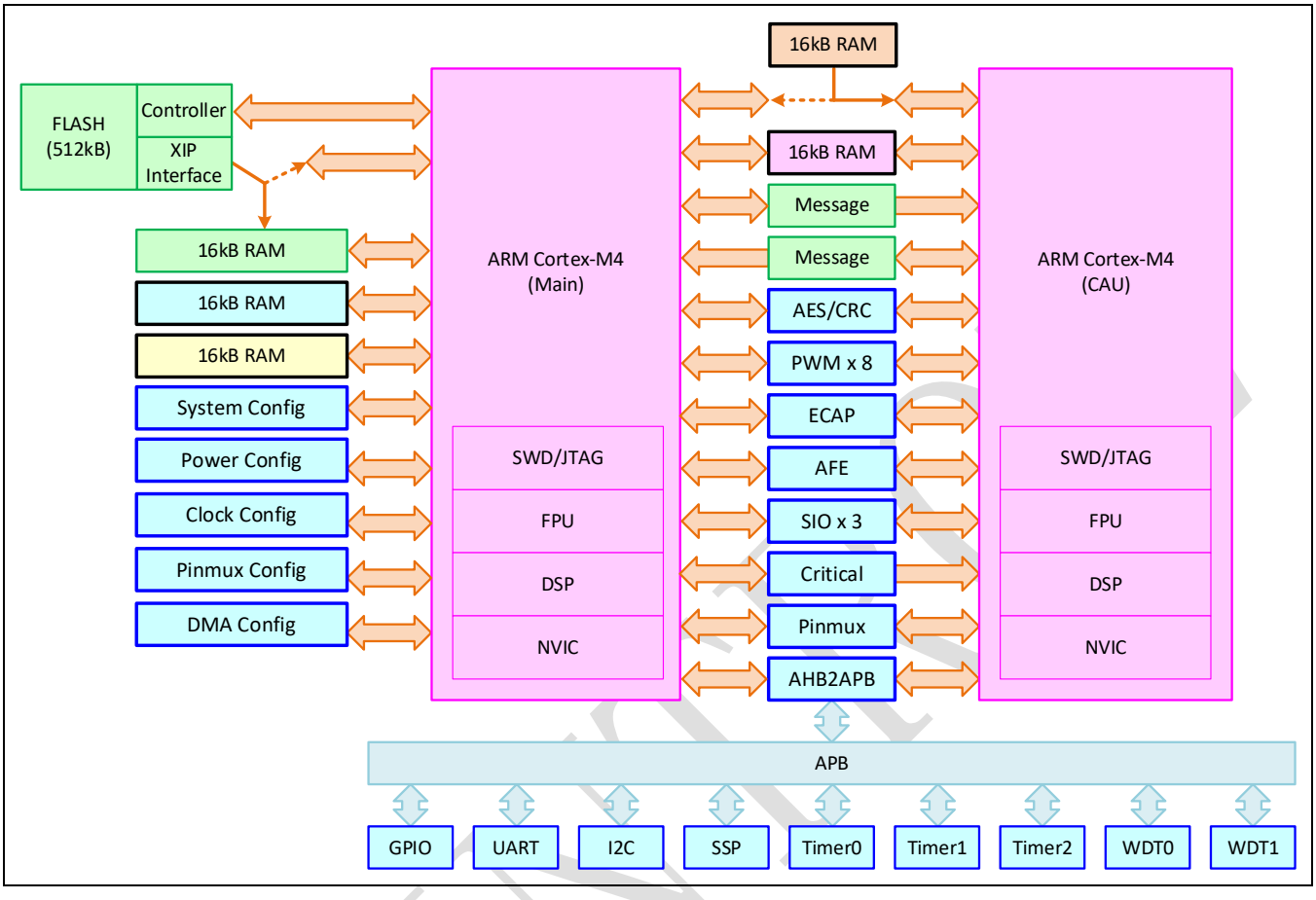


图 1-2：Main CPU 与 CAU 框图



在 SPC2168，主系统包括：

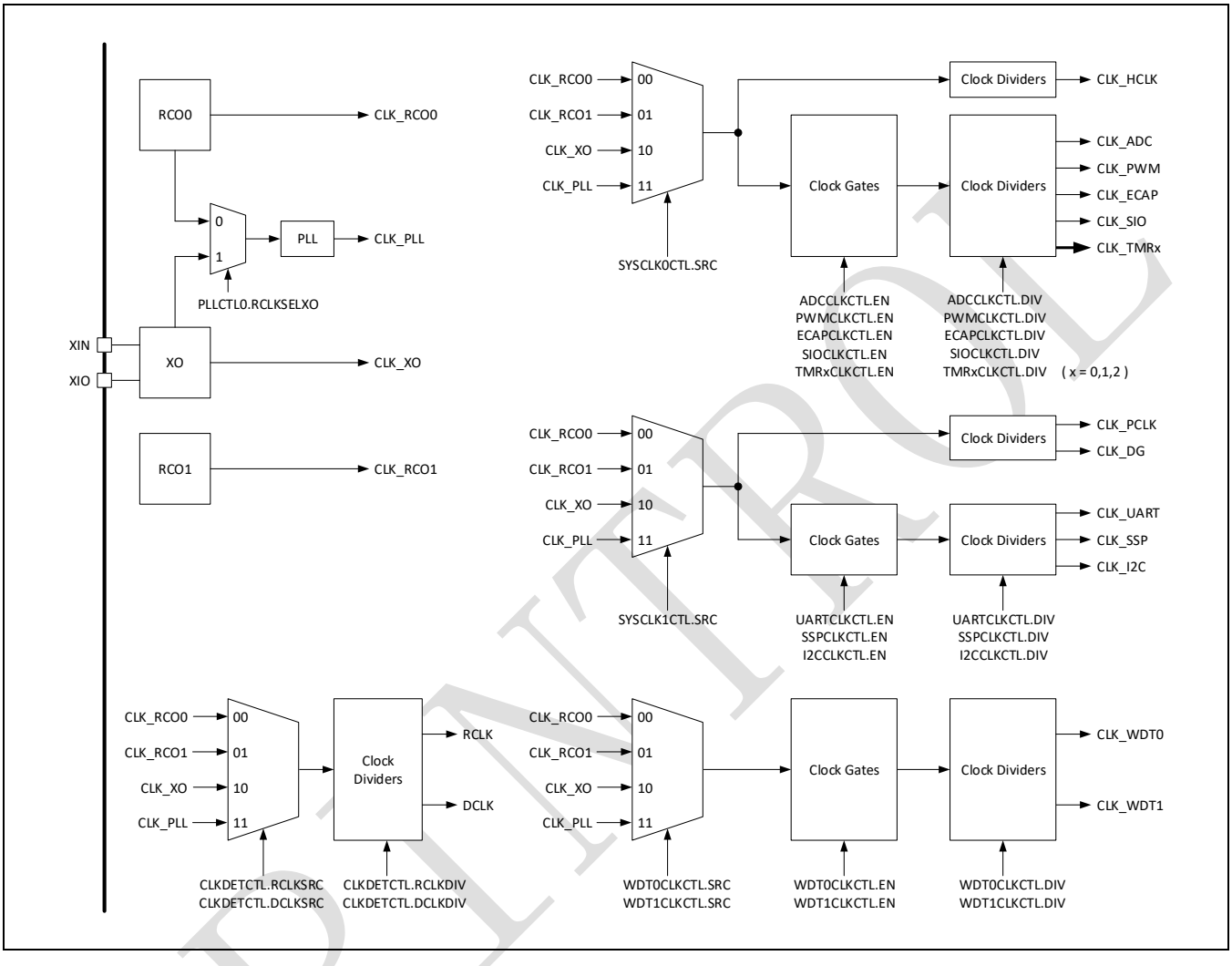
- 7 个总线主机
  - Cortex-M4 主核的 I 总线、D 总线以及系统总线
  - CAU（控制加速器单元）Cortex-M4 I 总线、D 总线以及系统总线
  - DMA
- 21 个总线从机
  - ROM
  - 5 块 16KB SRAM：其中一个可以用作 XIP 缓存，另外两个可以分配给 Main CPU 或 CAU
  - FLASH 控制器以及 XIP 接口，仅连接到 Main CPU
  - 2 个邮箱
  - 电源和时钟配置寄存器
  - DMA 控制寄存器
  - AHB 外设：AES、CRC、AFE、PWM、ECAP 和 SIO0/1/2
  - AHB 转 APB 桥，用于接各种 APB 外设

如图 1-1 所示，它们通过低延时的 AHB 总线架构互连，AHB-to-APB 桥连接所有的 APB 外设并实现 AHB 到 APB 的信号同步。APB 总线的最高速度限定为 50MHz。

如图 1-2 所示，存在一些区域，只能被 Main CPU 访问，CAU 无法访问。例如 Flash，System Config 等。

1.2 时钟树

图 1-3：SPC2168 时钟树



如图 1-3 所示，SPC2168 有如下 4 个根时钟：

- RCO0：校准为 32MHz
- RCO1：校准为 32MHz
- XO：支持片外无源晶体配合片上有源振荡电路（双管脚配置）或者外部有源时钟输入（单管脚配置，可选 XIN 或 XIO）
- PLL：提供最高 200MHz 的灵活时钟，其参考时钟可选为 RCO0 或者 XO

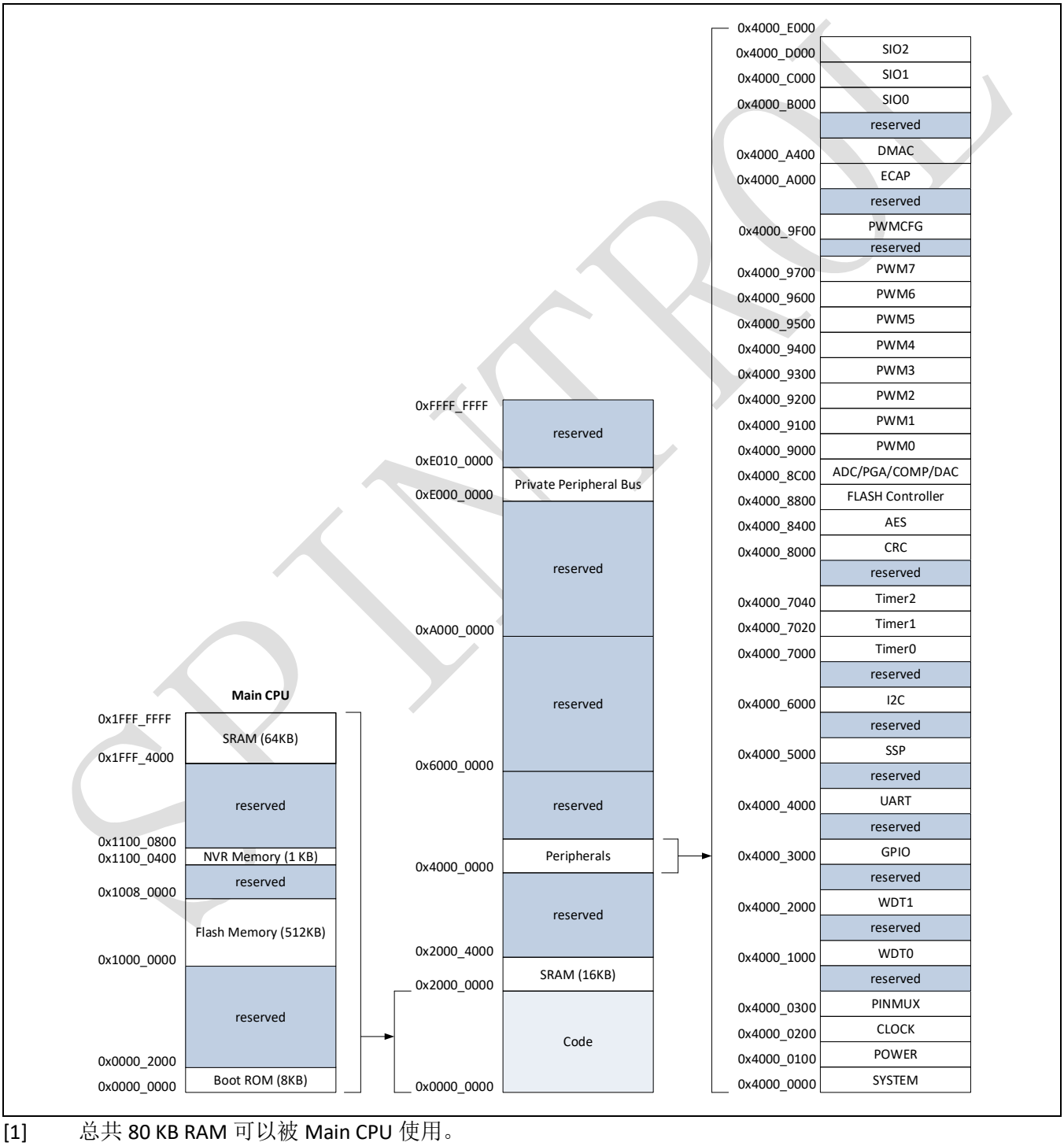
随后的平滑选择器可以实现无毛刺的时钟切换，然后通过对应的分频器和门控为各功能模块提供独立的时钟。

1.3 存储器映射

SPC2168 将 ROM、SRAM、寄存器和 I/O 端口分配在同一个 4GB 的线性地址空间内。数据字节在存储单元中采用小端模式存放。

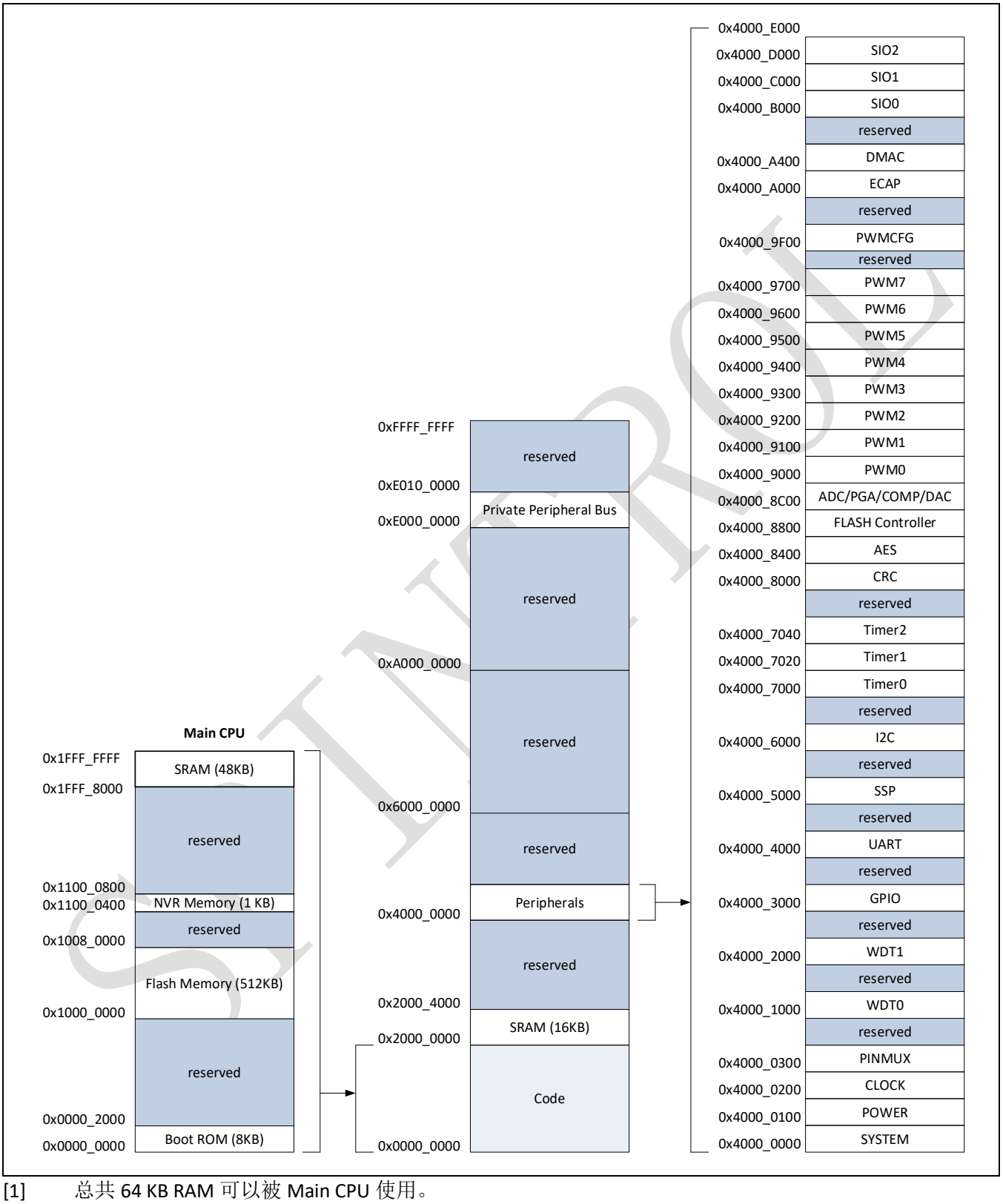
未分配给任何存储单元或者外设寄存器的地址视作“未使用的”空间。对这类地址的访问将产生总线错误。SPC2168 的存储器映射如图 1-4~图 1-7 所示。

图 1-4：缓存与 CAU 禁用时 SPC2168 内存映射（单核模式）



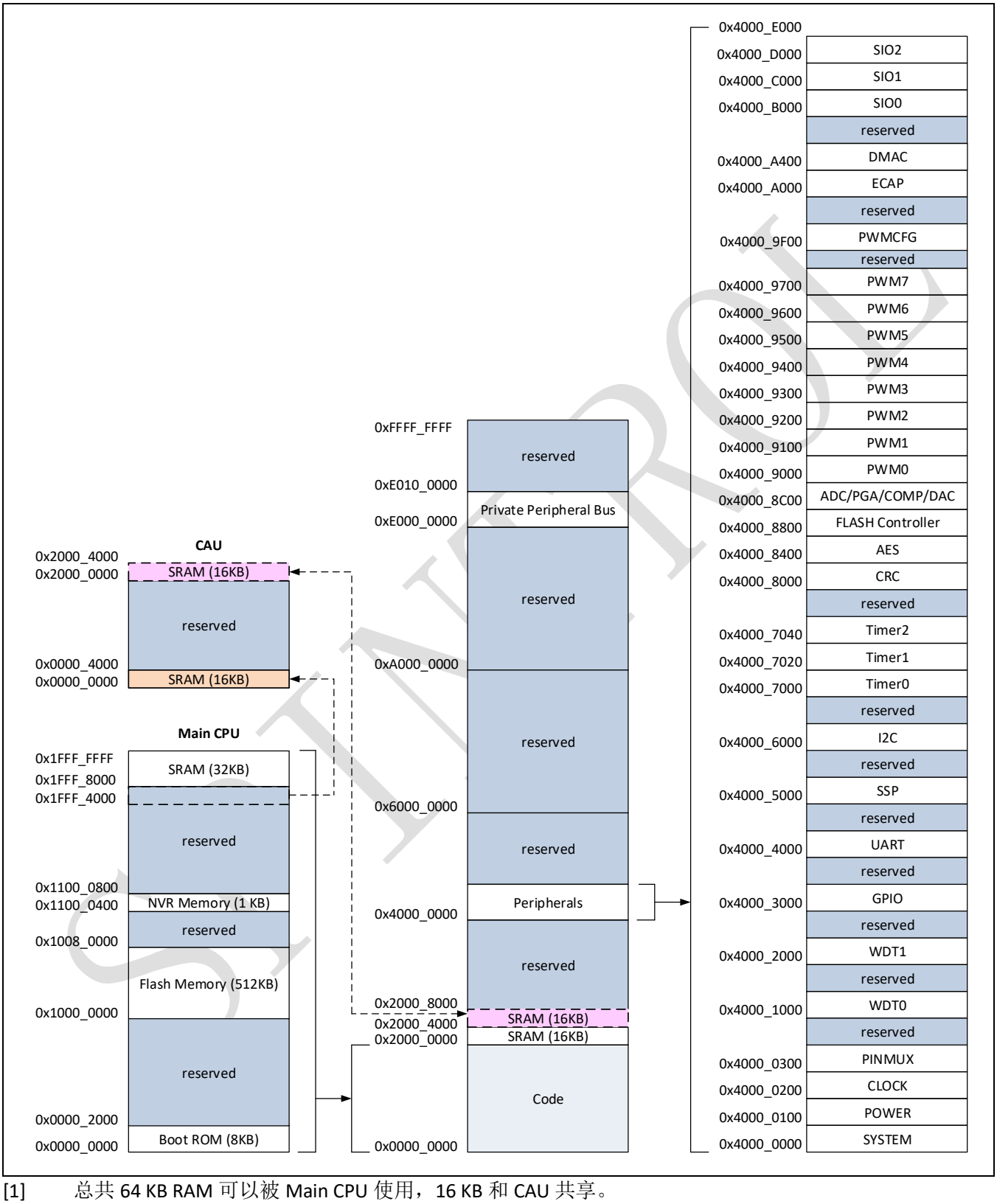
[1] 总共 80 KB RAM 可以被 Main CPU 使用。

图 1-5：缓存使能与 CAU 禁用时 SPC2168 内存映射（单核模式）



[1] 总共 64 KB RAM 可以被 Main CPU 使用。

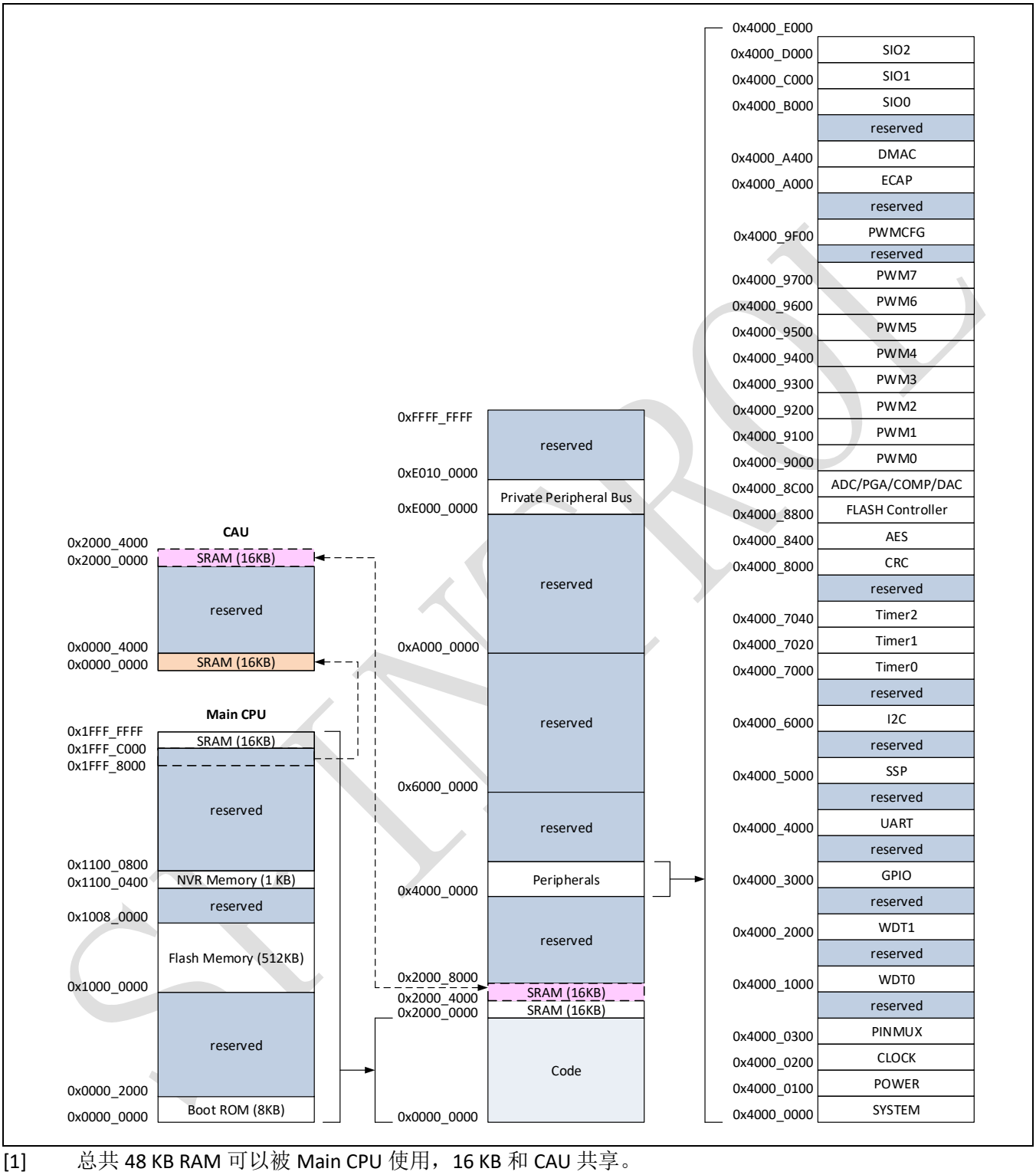
图 1-6：缓存禁用与 CAU 使能时 SPC2168 内存映射（双核模式）



[1] 总共 64 KB RAM 可以被 Main CPU 使用，16 KB 和 CAU 共享。

[2] 总共 32 KB RAM 可以被 CAU 使用，16 KB 和 Main CPU 共享。

图 1-7：缓存使能与 CAU 使能时 SPC2168 内存映射（双核模式）



[1] 总共 48 KB RAM 可以被 Main CPU 使用，16 KB 和 CAU 共享。

[2] 总共 32 KB RAM 可以被 CAU 使用，16 KB 和 Main CPU 共享。

表 1-1 给出了 SPC2168 各个外设模块的基地址。

表 1-1：外设模块基地址

外设模块	基地址	寄存器列表
SYSTEM	0x4000 0000	章节 22.9.1
POWER	0x4000 0100	章节 2.5
CLOCK	0x4000 0200	章节 3.11.1
GPIO	0x4000 3000	章节 4.4.1
PINMUX	0x4100 0300	章节 4.4.3
AES	0x4000 8400	章节 6.3.1
CRC	0x4000 8000	章节 7.4.1
TIMER0	0x4000 7000	章节 8.4.1
TIMER1	0x4000 7020	章节 8.4.1
TIMER2	0x4000 7040	章节 8.4.1
WDT0	0x4000 1000	章节 9.4.1
WDT1	0x4000 2000	章节 9.4.1
PWM0	0x4000 9000	章节 10.11.1
PWM1	0x4000 9100	章节 10.11.1
PWM2	0x4000 9200	章节 10.11.1
PWM3	0x4000 9300	章节 10.11.1
PWM4	0x4000 9400	章节 10.11.1
PWM5	0x4000 9500	章节 10.11.1
PWM6	0x4000 9600	章节 10.11.1
PWM7	0x4000 9700	章节 10.11.1
PWMCFG	0x4000 9F00	章节 10.11.1
ECAP	0x4000 A000	章节 11.7.1
ADC	0x4000 8C00	章节 12.13.1
PGA	0x4000 8C00	章节 13.10.1
COMP	0x4000 8C00	章节 14.6.1
UART	0x4000 4000	章节 17.5.1
I2C	0x4000 6000	章节 18.6.1
SSP	0x4000 5000	章节 19.5.1
FLASH	0x4000 8800	章节 20.8.1
DMAC	0x4000 A400	章节 23.4.1
MAILBOX	0x2000 8200	章节 24.4.1

1.3.1 嵌入式 Flash 存储器

SPC2168 集成了一个片上嵌入式 Flash 存储器，可以提供最大 512kB 空间用于存放程序的代码和数据。如图 1-1 所示，可以通过特别设计的 XIP（Execution-in-Place）接口读取 Flash 存储器的内容，还可以通过另外一个 AHB 接口的控制器来擦写 Flash 存储器。因此，当 SPC2168 启动时，Flash 存储器中的代码可以直接被本地执行，也可以被加载到 SRAM 后，再在 SRAM 中执行。可选的 16KB 缓存可被启用，以便加快 XIP 模式的执行速度。

出于数据安全性考虑，Flash 存储器和 ROM 都提供了可选的 ECC 功能，可以修正单比特错误和检测两比特错误。用户可以通过表 22-25 和表 22-26 所示的 MEMECCEN 寄存器来使能或者关闭这个功能。

Flash 存储器按照 1024 个扇区来组织，每个扇区包含 512 字节。Flash 存储器每次可以写入 32 个比特位（word，一个 4 字节字）。数据可以通过以下方式来擦除：512 字节的扇区擦除、4kB 的块擦除或者 Flash 整片擦除。

1.3.2 片上 SRAM

SPC2168 集成了多个 SRAM，用于存放代码和数据。这些 SRAM 可以按照字节、半字（16 位）或者字（32 位）的形式来访问，访问是零延迟的。根据控制加速器单元（CAU）和 XIP 缓存是否可用，整个 80KB SRAM 的地址可以被重新分配，如图 1-8 所示。

- XIP 缓存禁用，CAU 禁用（CAUCTL.RUN=0）

  在此情形下，整个 80KB 都可被 Main CPU 访问，访问地址为 0x1FFF4000 ~ 0x20007FFF。

- XIP 缓存禁用，CAU 使能（CAUCTL.RUN=1）

  在此情形下，48KB SRAM 被分配给 Main CPU，访问地址为 0x1FFF8000 ~ 0x20003FFF。另一个 16KB SRAM 既可以被 Main CPU 访问，访问地址为 0x20004000 ~ 0x20007FFF；也可以被 CAU 访问，访问地址为 0x20000000 ~ 0x20003FFF。剩下的 16KB SRAM，原本可以被 Main CPU 按照地址 0x1FFF4000 ~ 0x1FFF7FFF 进行访问，用于初始化 CAU 代码；现在被分配给 CAU 作为代码 RAM，并且 CAU 可以按照地址 0x0000 ~ 0x3FFF 进行访问。

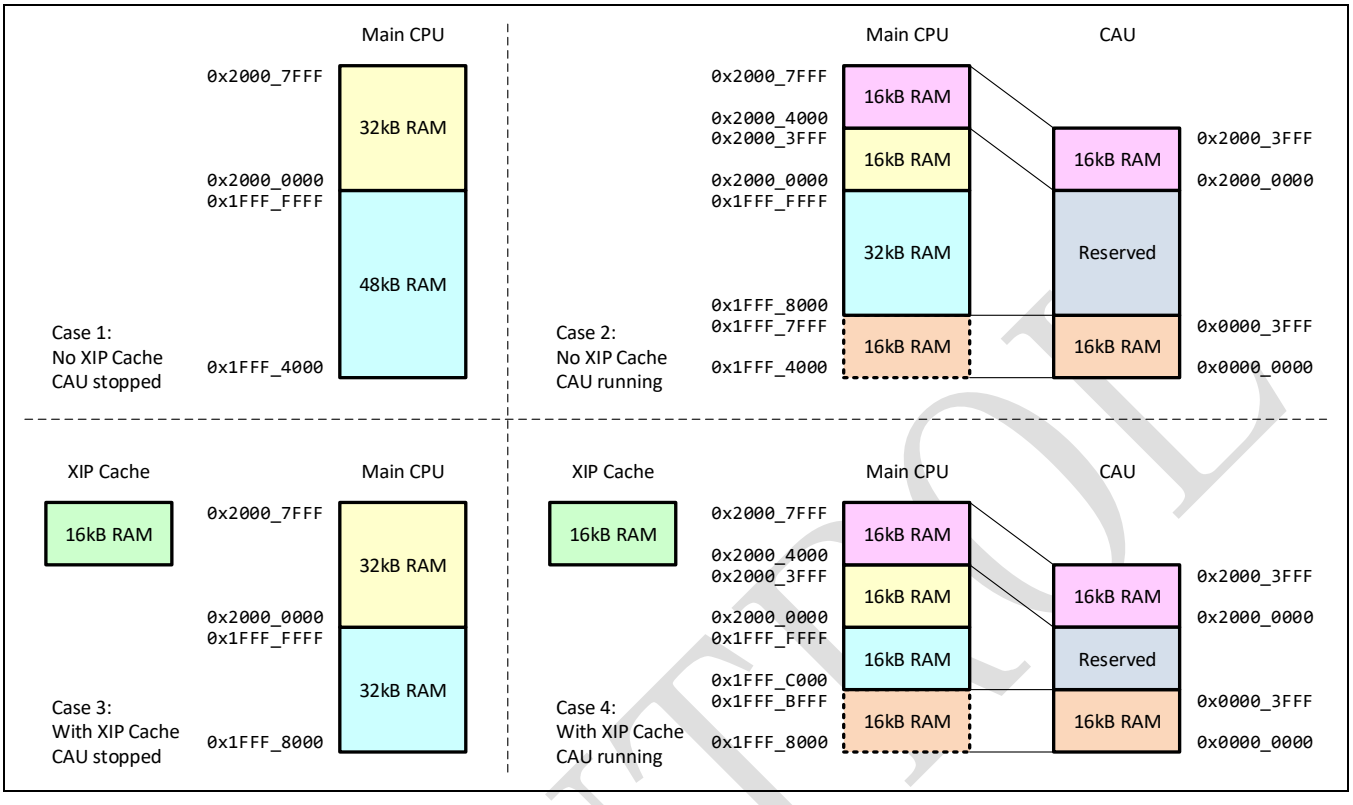
- XIP 缓存使能，CAU 禁用（CAUCTL.RUN=0）

  在此情形下，16KB SRAM 被用作 XIP 缓存；其他的 64KB 被分配给 Main CPU，访问地址为 0x1FFF8000 ~ 0x20007FFF。

- XIP 缓存使能，CAU 使能（CAUCTL.RUN=1）

  在这种情形下，16KB SRAM 被用作 XIP 缓存。32KB SRAM 被分配给 Main CPU，访问地址为 0x1FFFC000 ~ 0x20003FFF。另一个 16KB SRAM 既可以被 Main CPU 按照地址 0x20004000 ~ 0x20007FFF 进行访问，也可以被 CAU 按照地址 0x20000000 ~ 0x20003FFF 进行访问。剩下的 16KB SRAM，原本可以被 Main CPU 按照地址 0x1FFF8000 ~ 0x1FFFBFFF 进行访问，用于初始化 CAU 代码；现在被分配给 CAU 作为代码 RAM，并且 CAU 可以按照地址 0x0000 ~ 0x3FFF 进行访问。

图 1-8：SPC2168 SRAM 地址分配



另外，对于 SRAM，SPC2168 实现了 ECC 校验特性，用以增强数据安全。ECC 功能可以修正单比特错误和检测两比特错误。当使能该特性时，一旦数据出现 1-bit 纠错或者 2-bit 检验错误，就会产生一个中断通知 Cortex-M4 处理器。这个特性由 SYSTEM 模块的 MEMIF，MEMIC 以及 MEMIE 寄存器控制。寄存器的详细信息请参考表 22-19 到表 22-24。

1.4 启动引导配置

在 SPC2168，通过 BOOT 管脚和 TRSTn 管脚可以实现两种不同的启动模式：

- 从 Flash 存储器启动（BOOT 管脚为高电平，TRSTn 为任意电平）：在系统复位后，启动引导程序（boot loader）跳转到 Flash 存储器并从地址 0x1000 0000 开始执行。
- ISP（In System Programming）模式（BOOT 和 TRSTn 均为低电平）：启动引导程序使用 UART 接口对 Flash 存储器进行重新编程。在这个过程中，对于封装类型为 QFN52L 的芯片，GPIO38 被配置为 UART_TXD 功能，GPIO39 被配置为 UART_RXD 功能；对于其他封装类型的芯片，GPIO44 被配置为 UART_TXD 功能，GPIO45 被配置为 UART_RXD 功能。

注意： 当 BOOT 管脚为低电平时，必须保持 TRSTn 管脚为低电平，否则芯片将进入工程测试模式。

表 1-2：启动模式

启动模式选择管脚		启动模式
BOOT	TRSTn	
0	0	ISP 模式（对于封装类型为 QFN52L 的芯片，GPIO38 配置为 UART_TXD，而 GPIO39 配置为 UART_RXD；对于其他封装类型的芯片，GPIO44 配置为 UART_TXD，而 GPIO45 配置为 UART_RXD）
0	1	工程测试模式，该模式下芯片不能正常工作，**任何时候都不要选择该模式**
1	x	正常启动模式，从 Flash 存储器开始运行程序 特别说明：当 TRSTn 为高电平时，芯片调试接口处于有效状态

2 电源

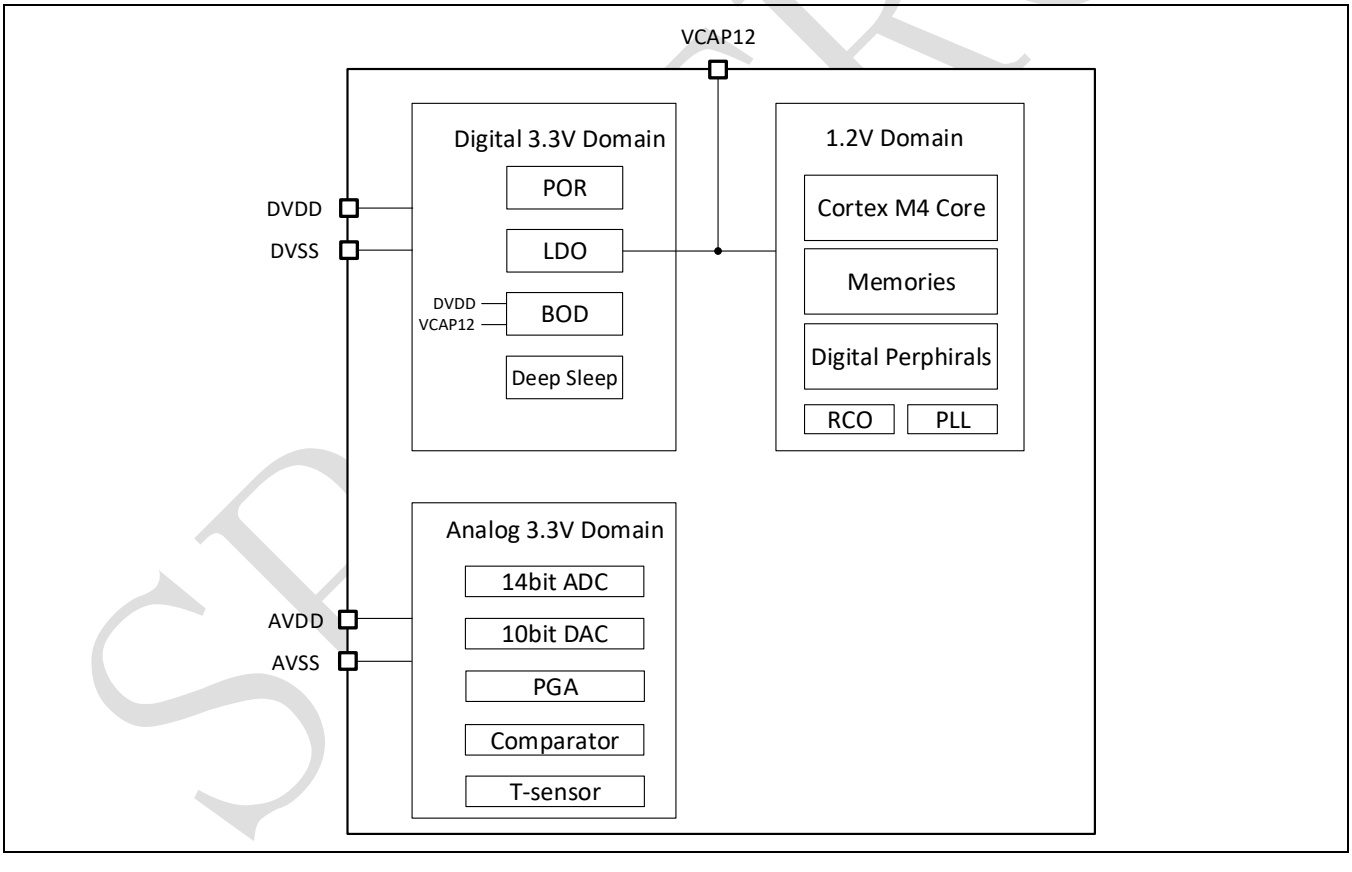
2.1 单电源供电

SPC2168 需要对 DVDD 管脚额定 3.3V 供电，片上 LDO 由 DVDD 产生额定 1.2V 电压，并输出到 VCAP12 管脚。上电顺序无特殊要求。

在上电过程中，片上 LDO 确保 VCAP12 上的负载电流低于 30mA。若检测到超过 30mA 的电流（例如由板上寄生短路引起），则 VCAP12 无法建立额定电压，系统无法正确上电。这个安全特性可以在诸如电源管脚连接错误等场景中保护芯片。

待 VCAP12 正常上电后，芯片退出复位状态。此时片上 LDO 监控 VCAP12 上的最大负载电流约为 500mA 到 1A。如果发生异常导致电流超出预定范围，则 VCAP12 掉电，并重新调整最大电流输出为 30mA。这个安全特性保护芯片不被烧毁。

图 2-1：电源系统总览



2.2 模拟前端供电

SPC2168 模拟前端供电管脚 AVDD 采用如章节 2.1 所述 DVDD 相同的供电。设计电路板时，需要注意在尽量靠近 AVDD 和 AVSS 管脚的地方放置去耦电容。

2.3 过压/欠压检测

BOD（Brown-out Detectors）模块用于将 3.3V/1.2V 额定电压域实际电压值与预设阈值加以比较，实现过压或者欠压的检测，检测结果存放于 BODIF 寄存器中，并可以根据 BODIE 寄存器配置决定是否产生中断。

BOD33CTL 和 BOD12CTL 寄存器用于启用或禁用 3.3V/1.2V 额定电压域的过压欠压检测并设定相应的阈值。

为了提高额外的灵活度，1.2V 电压域提供了两个 BOD 欠压检测。用户可以实现诸如低于某个值时产生中断报警、进一步低于某个值时产生复位的保护功能。

图 2-2 和图 2-3 给出了 3.3V/1.2V 额定电压域检测电路框图。

图 2-2：3.3V 额定电压域检测电路框图

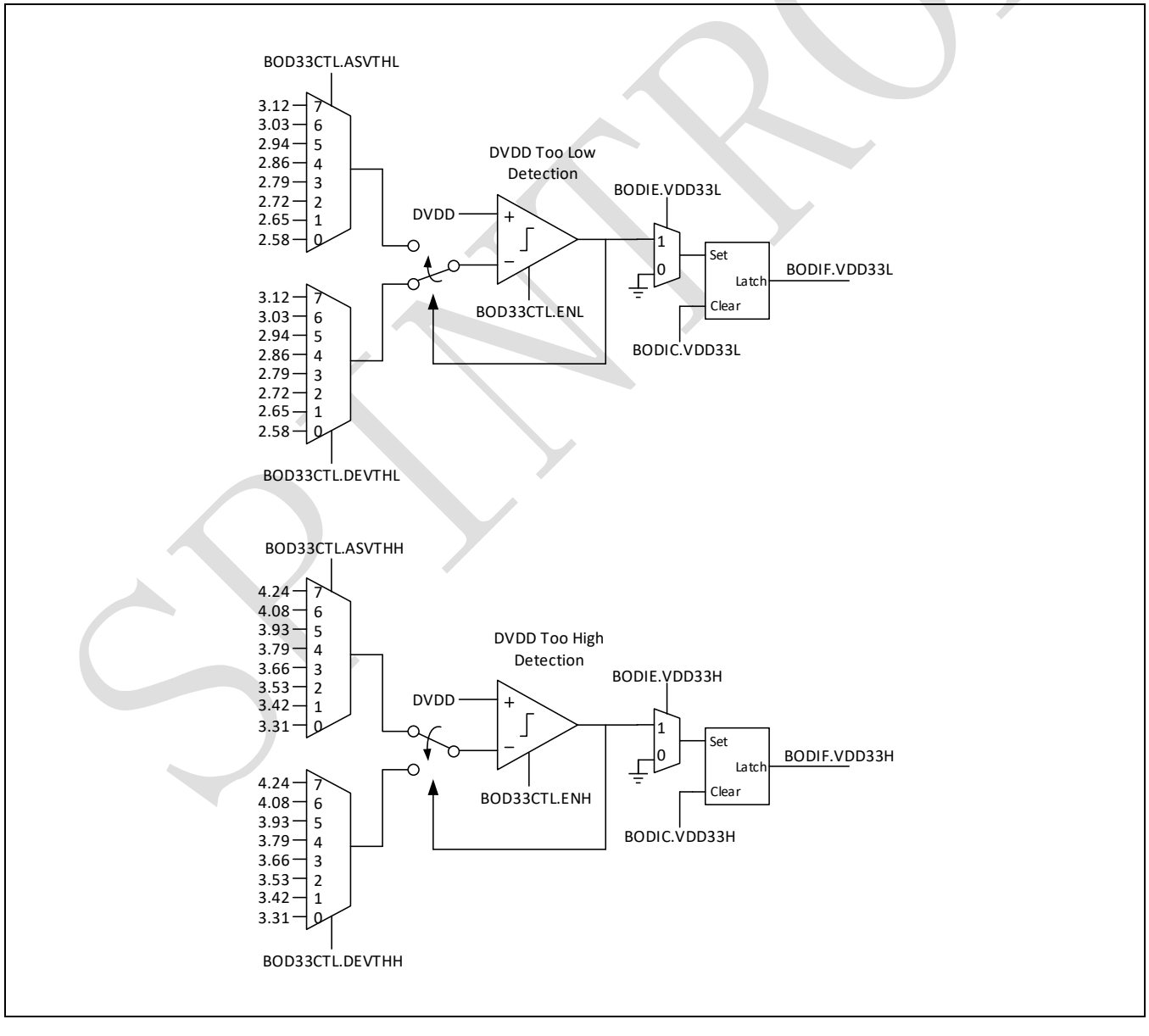
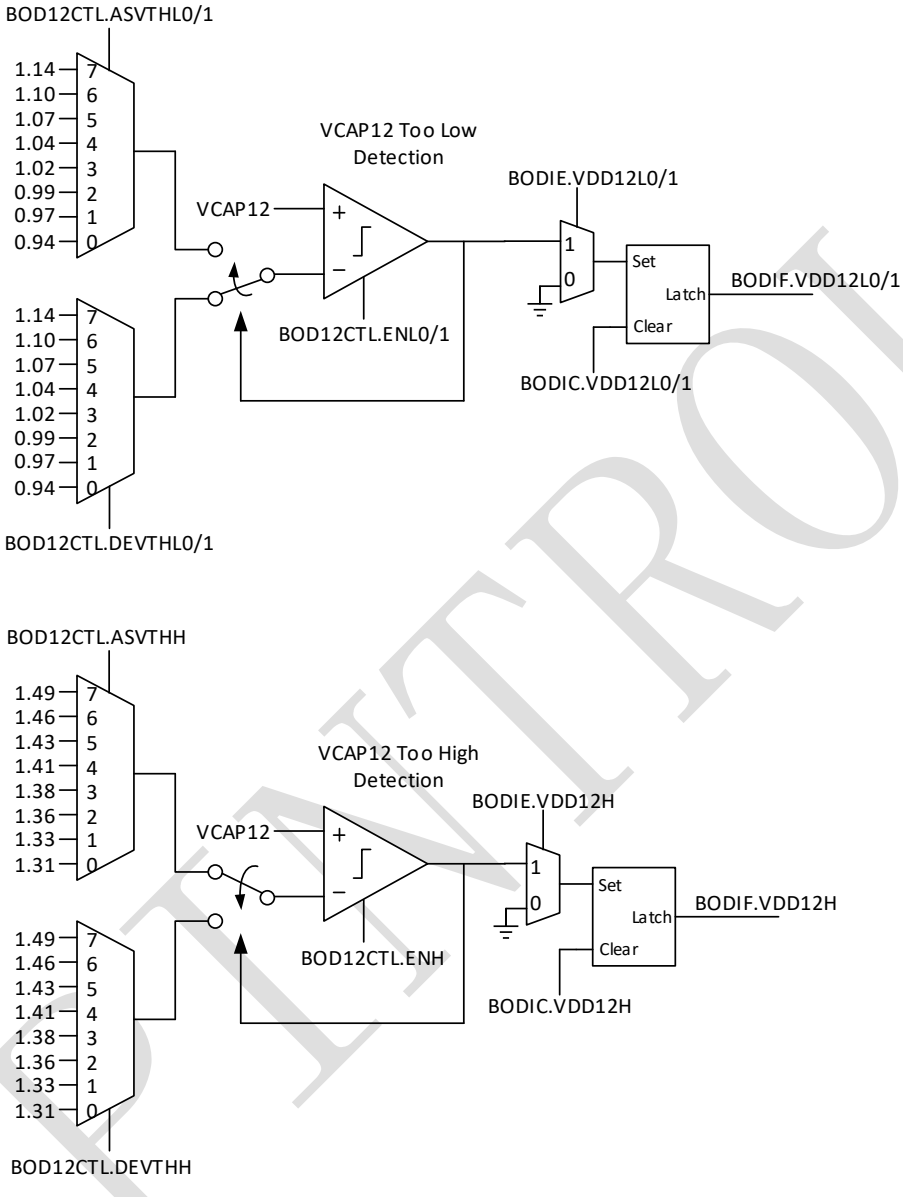


图 2-3：1.2V 额定电压域检测电路框图

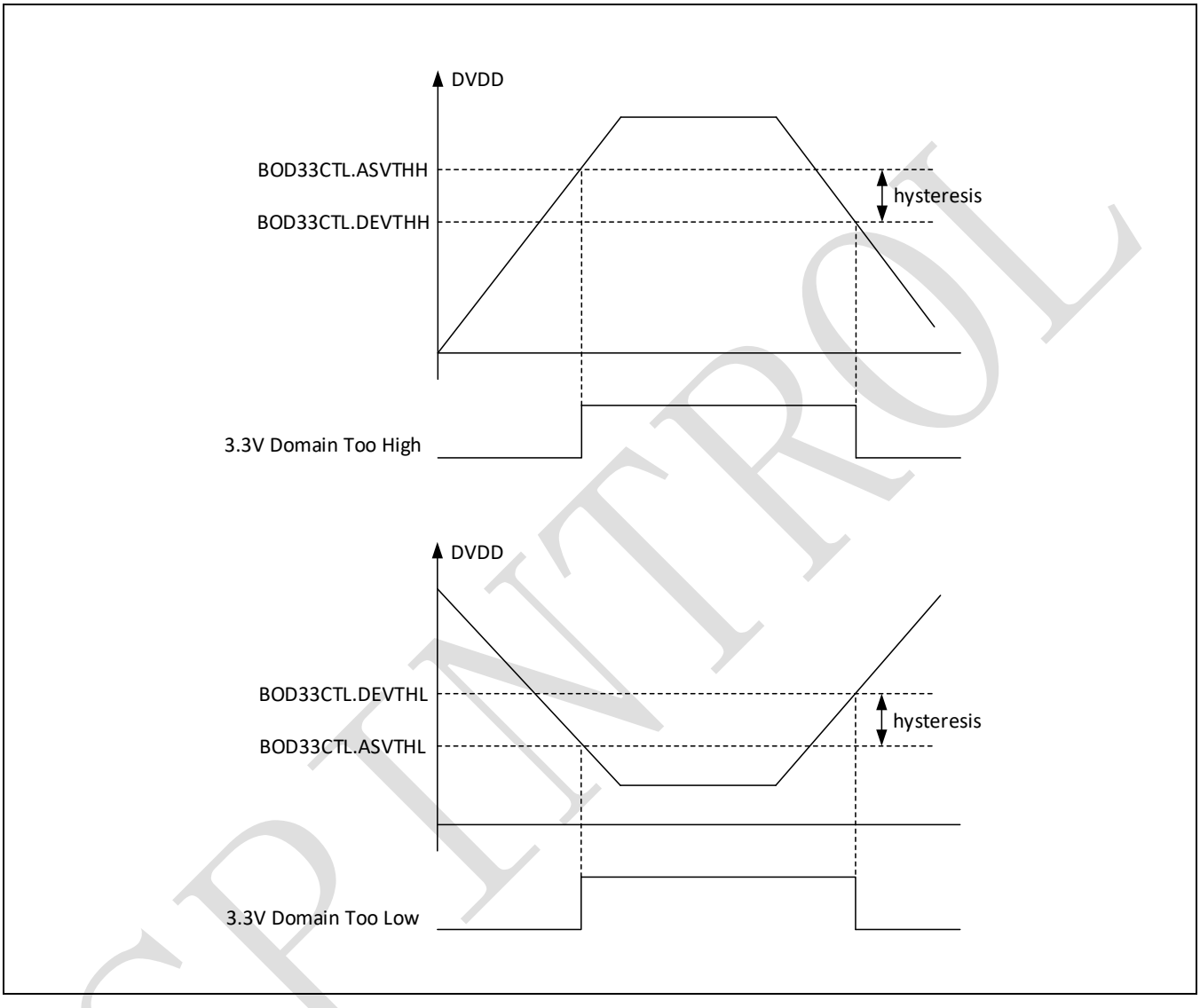


注意：　　按照如下说明来正确使用 BOD：

1. 过压检测时，BOD33CTL.ASVTHH/BOD12CTL.ASVTHH 必须大于或者等于 BOD33CTL.DEVTHH/BOD12CTL.DEVTHH。
2. 在欠压检测时，BOD33CTL.ASVTHL/BOD12CTL.ASVTHL0/ BOD12CTL.ASVTHL1 必须小于或者等于 BOD33CTL.DEVTHL/BOD12CTL.DEVTHL0/BOD12CTL.DEVTHL1。

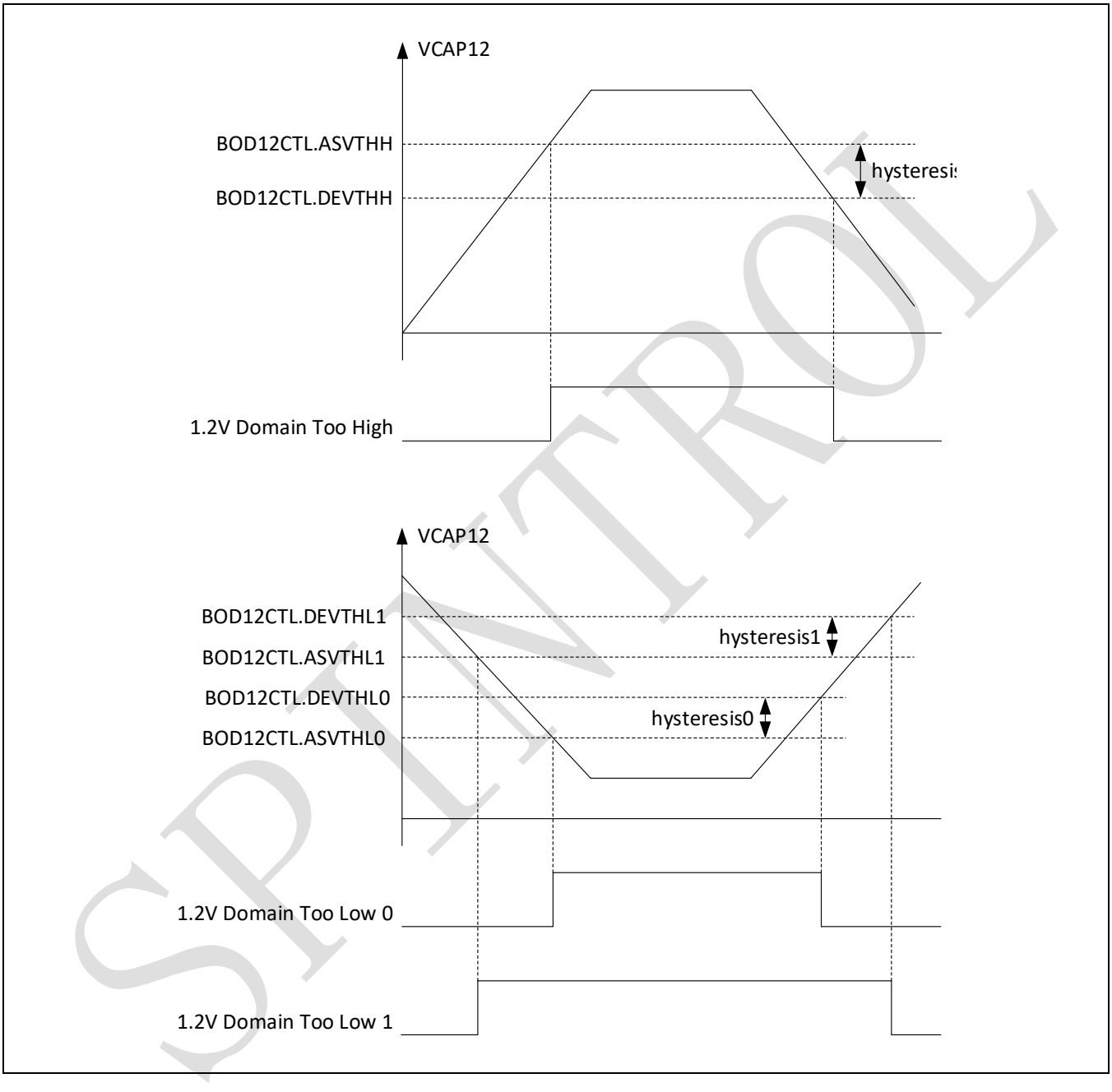
3.3V 额定电压域可以在 DVDD 电压上升并高于过压阈值时产生过压中断，或者在 DVDD 电压下降并低于欠压阈值时产生欠压中断。

图 2-4：DVDD 阈值



1.2V 额定电压域可以在 VCAP12 电压上升并高于过压阈值时产生过压中断，或者在 VCAP12 电压下降并低于欠压阈值时产生欠压中断。

图 2-5：VCAP12 阈值



2.4 深度睡眠模式

SPC2168 的片上 1.2V LDO 支持深度睡眠模式。用户可以通过向 DPSLPKEY 寄存器写 0x51EE9 来关闭 1.2V LDO，并使得整个系统进入深度睡眠模式。之后，可以通过 XRSTn 管脚置低来唤醒系统。

2.5 寄存器

2.5.1 Power 寄存器列表

表 2-1：POWER 模块基地址

外设模块	基地址
POWER	0x4000 0100

表 2-2：POWER 寄存器列表

寄存器	偏移地址	描述	复位值
PWRSTS	0x00	电源状态寄存器	0x00000001
LDOCTL*	0x04	LDO 控制寄存器	0x00000007
PORCTL*	0x0C	低压 POR 控制寄存器	0x00000000
BODIF	0x10	BOD 中断标志寄存器	0x00000000
BODIC	0x14	BOD 中断清除寄存器	0x00000000
BODIE*	0x18	BOD 中断使能寄存器	0x00000000
BODCTL*	0x1C	BOD 控制寄存器	0x00000007
BOD33CTL*	0x20	3.3V BOD 控制寄存器	0x00006E10
BOD12CTL*	0x24	1.2V BOD 控制寄存器	0x006E1010
DPSLPKEY*	0x28	深度睡眠模式使能寄存器	0x00000000
PWRREGKEY	0x2C	POWER 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 PWRREGKEY=0x1ACCE551 才可以改写。

2.5.2 Power 寄存器

表 2-3 到表 2-24 给出了 Power 模块各寄存器的定义。

表 2-3：电源状态寄存器（PWRSTS）位段定义

PWRSTS (Power Status Register) Offset: 0x0 Default: 0x00000001 Access: POWER -> PWRSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		RESERVED	RESERVED	RESERVED	RESERVED	RESERVED	FLASHREGRDY

表 2-4：电源状态寄存器（PWRSTS）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	RESERVED_5	RO	0x0	保留
4	RESERVED_4	RO	0x0	保留
3	RESERVED_3	RO	0x0	保留
2	RESERVED_2	RO	0x0	保留
1	RESERVED_1	RO	0x0	保留
0	FLASHREGRDY	RO	0x1	Flash 电源就绪标志 0：未就绪 1：就绪

表 2-5：LDO 控制寄存器（LDOCTL）位段定义

LDOCTL (LDO Control Register) Offset: 0x4 Default: 0x00000007 Access: POWER -> LDOCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				VREFTRIM			

表 2-6：LDO 控制寄存器（LDOCTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:0	VREFTRIM	RW	0x7	LDO 参考电压（出厂校准，不建议用户修改） 0000：1.06V 0001：1.08V 0010：1.10V 0011：1.12V 0100：1.14V 0101：1.16V 0110：1.18V 0111：1.20V 1000：1.22V 1001：1.24V 1010：1.26V 1011：1.28V 1100：1.30V 1101：1.32V 1110：1.34V 1111：1.36V

表 2-7：低压 POR 控制寄存器（PORCTL）位段定义

PORCTL (Low Power POR Control Register) Offset: 0xC Default: 0x00000000 Access: POWER -> PORCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					RESERVED	XRSTFILT	

表 2-8：低压 POR 控制寄存器（PORCTL）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	RESERVED_2	W1C	0x0	保留
1:0	XRSTFILT	RW	0x0	XRSTn 复位管脚消抖时间窗 00：500us 01：1ms 10：2ms 11：4ms

表 2-9：BOD 中断标志寄存器（BODIF）位段定义

BODIF (BOD Interrupt Flag Register) Offset: 0x10 Default: 0x00000000 Access: POWER -> BODIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		BODINT	VDD33H	VDD33L	VDD12H	VDD12L1	VDD12L0

表 2-10：BOD 中断标志寄存器（BODIF）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	BODINT	RO	0x0	锁存的 BOD 全局中断标志 0：BOD 中断未发生 1：BOD 中断曾发生 该位被清零前不会产生新的 BOD 中断。
4	VDD33H	RO	0x0	锁存的 VDD33H（VDD33 过压）事件标志 0：VDD33H 事件未发生 1：VDD33H 事件曾发生 该位被清零前新的 VDD33H 事件不会触发中断。
3	VDD33L	RO	0x0	锁存的 VDD33L（VDD33 欠压）事件标志 0：VDD33L 事件未发生 1：VDD33L 事件曾发生 该位被清零前新的 VDD33L 事件不会触发中断。
2	VDD12H	RO	0x0	锁存的 VDD12H（VDD12 过压）事件标志 0：VDD12H 事件未发生 1：VDD12H 事件曾发生 该位被清零前新的 VDD12H 事件不会触发中断。
1	VDD12L1	RO	0x0	锁存的 VDD12L1（VDD12 欠压）事件 1 标志 0：VDD12L1 事件未发生 1：VDD12L1 事件曾发生 该位被清零前新的 VDD12L1 事件不触发中断。
0	VDD12L0	RO	0x0	锁存的 VDD12L0（VDD12 欠压）事件 0 标志 0：VDD12L0 事件未发生 1：VDD12L0 事件曾发生 该位被清零前新的 VDD12L0 事件不触发中断。

表 2-11：BOD 中断清除寄存器（BODIC）位段定义

BODIC (BOD Interrupt Clear Register) Offset: 0x14 Default: 0x00000000 Access: POWER -> BODIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		BODINT	VDD33H	VDD33L	VDD12H	VDD12L1	VDD12L0

表 2-12：BOD 中断清除寄存器（BODIC）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	BODINT	W1C	0x0	BOD 全局中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.BODINT 标志位 本位段自动清 0
4	VDD33H	W1C	0x0	VDD33H 中断标志位清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.VDD33H 标志位 本位段自动清 0
3	VDD33L	W1C	0x0	VDD33L 中断标志位清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.VDD33L 标志位 本位段自动清 0
2	VDD12H	W1C	0x0	VDD12H 中断标志位清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.VDD12H 标志位 本位段自动清 0
1	VDD12L1	W1C	0x0	VDD12L1 中断标志位清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.VDD12L1 标志位 本位段自动清 0
0	VDD12L0	W1C	0x0	VDD12L0 中断标志位清除 0：写 0 无效，总是读回 0 1：写 1 清除 BODIF.VDD12L0 标志位 本位段自动清 0

表 2-13：BOD 中断使能寄存器（BODIE）位段定义

BODIE (BOD Interrupt Enable Register) Offset: 0x18 Default: 0x00000000 Access: POWER -> BODIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			VDD33H	VDD33L	VDD12H	VDD12L1	VDD12L0

表 2-14：BOD 中断使能寄存器（BODIE）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4	VDD33H	RW	0x0	VDD33H 中断使能 0：VDD33H 事件不触发中断 1：VDD33H 事件触发中断
3	VDD33L	RW	0x0	VDD33L 中断使能 0：VDD33L 事件不触发中断 1：VDD33L 事件触发中断
2	VDD12H	RW	0x0	VDD12H 中断使能 0：VDD12H 事件不触发中断 1：VDD12H 事件触发中断
1	VDD12L1	RW	0x0	VDD12L1 中断使能 0：VDD12L1 事件不触发中断 1：VDD12L1 事件触发中断
0	VDD12L0	RW	0x0	VDD12L0 中断使能 0：VDD12L0 事件不触发中断 1：VDD12L0 事件触发中断

表 2-15：BOD 控制寄存器（BODCTL）位段定义

BODCTL (BOD Control Register) Offset: 0x1C Default: 0x00000007 Access: POWER -> BODCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				VREFTRIM			

表 2-16：BOD 控制寄存器（BODCTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:0	VREFTRIM	RW	0x7	BOD 参考电压校准 0000：1.062V 0001：1.082V 0010：1.102V 0011：1.122V 0100：1.143V 0101：1.163V 0110：1.183V 0111：1.203V 1000：1.223V 1001：1.243V 1010：1.263V 1011：1.283V 1100：1.303V 1101：1.323V 1110：1.343V 1111：1.363V

表 2-17：3.3V BOD 控制寄存器（BOD33CTL）位段定义

BOD33CTL (3.3V BOD Control Register) Offset: 0x20 Default: 0x00006E10 Access: POWER -> BOD33CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED	DEVTHH			ASVTHH			ENH
7	6	5	4	3	2	1	0
RESERVED	DEVTHL			ASVTHL			ENL

表 2-18：3.3V BOD 控制寄存器（BOD33CTL）位段描述

位段	位段名	属性	复位值	描述
31:15	RESERVED_31_15	RO	0x0	保留
14:12	DEVTHH	RW	0x6	3.3V 电源下降时撤销 VDD33H 的阈值 000：3.31V 001：3.42V 010：3.53V 011：3.66V 100：3.79V 101：3.93V 110：4.08V 111：4.24V
11:9	ASVTHH	RW	0x7	3.3V 电源上升时报告 VDD33H 的阈值 000：3.31V 001：3.42V 010：3.53V 011：3.66V 100：3.79V 101：3.93V 110：4.08V 111：4.24V
8	ENH	RW	0x0	VDD33H BOD 使能 0：关闭 1：使能
7	RESERVED_7	RO	0x0	保留
6:4	DEVTHL	RW	0x1	3.3V 电源上升时撤销 VDD33L 的阈值 000：2.58V 001：2.65V 010：2.72V 011：2.79V

位段	位段名	属性	复位值	描述
				100：2.86V 101：2.94V 110：3.03V 111：3.12V
3:1	ASVTHL	RW	0x0	3.3V 电源下降时报告 VDD33L 的阈值 000：2.58V 001：2.65V 010：2.72V 011：2.79V 100：2.86V 101：2.94V 110：3.03V 111：3.12V
0	ENL	RW	0x0	VDD33L BOD 使能 0：关闭 1：使能

表 2-19：1.2V BOD 控制寄存器（BOD12CTL）位段定义

BOD12CTL (1.2V BOD Control Register) Offset: 0x24 Default: 0x006E1010
Access: POWER -> BOD12CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED	DEVTHH			ASVTHH			ENH
15	14	13	12	11	10	9	8
RESERVED	DEVTHL1			ASVTHL1			ENL1
7	6	5	4	3	2	1	0
RESERVED	DEVTHL0			ASVTHL0			ENL0

表 2-20：1.2V BOD 控制寄存器（BOD12CTL）位段描述

位段	位段名	属性	复位值	描述
31:23	RESERVED_31_23	RO	0x0	保留
22:20	DEVTHH	RW	0x6	1.2V 电源下降时撤销 VDD12H 的阈值 000：1.31V 001：1.33V 010：1.36V 011：1.38V 100：1.41V 101：1.43V 110：1.46V 111：1.49V

位段	位段名	属性	复位值	描述
19:17	ASVTHH	RW	0x7	1.2V 电源上升时报告 VDD12H 的阈值 000：1.31V 001：1.33V 010：1.36V 011：1.38V 100：1.41V 101：1.43V 110：1.46V 111：1.49V
16	ENH	RW	0x0	VDD12H BOD 使能 0：关闭 1：使能
15	RESERVED_15	RO	0x0	保留
14:12	DEVTHL1	RW	0x1	1.2V 电源上升时撤销 VDD12L1 的阈值 000：0.94V 001：0.97V 010：0.99V 011：1.02V 100：1.04V 101：1.07V 110：1.10V 111：1.14V
11:9	ASVTHL1	RW	0x0	1.2V 下降时报告 VDD12L1 的阈值 000：0.94V 001：0.97V 010：0.99V 011：1.02V 100：1.04V 101：1.07V 110：1.10V 111：1.14V
8	ENL1	RW	0x0	VDD12L1 BOD 使能 0：关闭 1：使能
7	RESERVED_7	RO	0x0	保留
6:4	DEVTHL0	RW	0x1	1.2V 电源上升时撤销 VDD12L0 的阈值 000：0.94V 001：0.97V 010：0.99V 011：1.02V 100：1.04V

位段	位段名	属性	复位值	描述
				101：1.07V 110：1.10V 111：1.14V
3:1	ASVTHL0	RW	0x0	1.2V 下降时报告 VDD12L0 的阈值 000：0.94V 001：0.97V 010：0.99V 011：1.02V 100：1.04V 101：1.07V 110：1.10V 111：1.14V
0	ENL0	RW	0x0	VDD12L0 BOD 使能 0：关闭 1：使能

表 2-21：深度睡眠模式使能寄存器（DPSLPKEY）位段定义

DPSLPKEY (Deep Sleep Enable Key Register) Offset: 0x28 Default: 0x00000000 Access: POWER -> DPSLPKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 2-22：深度睡眠模式使能寄存器（DPSLPKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x0	写 0x51ee9 令系统进入深度睡眠模式 写其余值无效

表 2-23：POWER 模块写使能寄存器（PWRREGKEY）位段定义

PWRREGKEY (Power Register Write-Allow Key Register) Offset: 0x2C Default: 0x1ACCE551 Access: POWER -> PWRREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 2-24：POWER 模块写使能寄存器（PWRREGKEY）位段描述

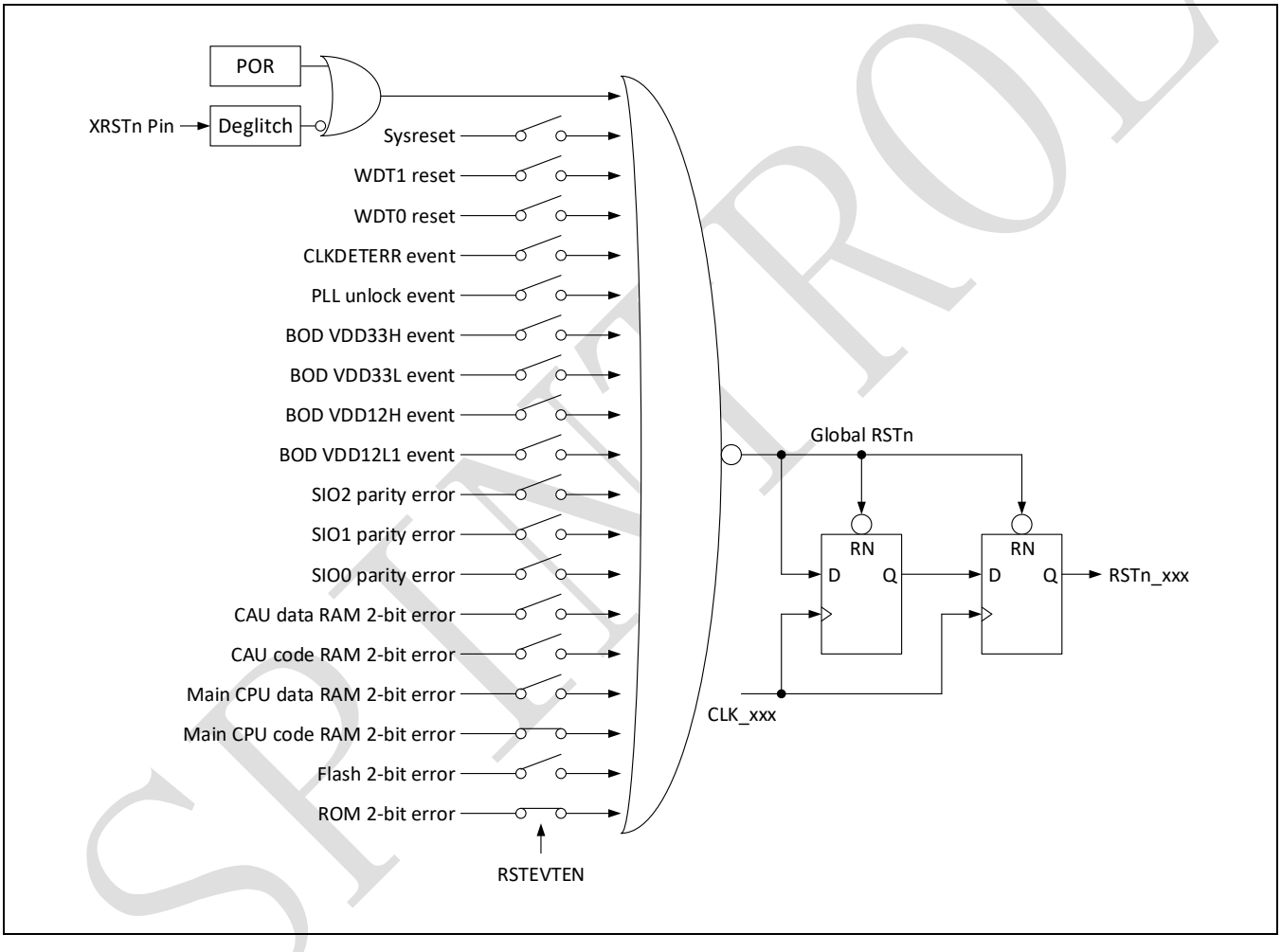
位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 POWER 寄存器

3 时钟与复位

3.1 复位架构

如图 3-1 所示。SPC2168 的低电平有效全局复位信号为如下信号的逻辑或（NOR）：上电复位（POR）模块，XRSTn 管脚，以及各种功能复位（包括存储错误、欠压过压错误、时钟错误、看门狗定时器超时以及 Cortex-M4 SYSRESET 复位请求等）。

图 3-1：复位信号框图



RSTEVTSTS 寄存器记录了各种功能复位事件供用户系统调试分析所用，并可以通过 RSTEVTCLR 寄存器对应比特位清除锁存的复位事件。此外，功能复位源可以通过 RSTEVTEN 寄存器的对应位选择是否使能并触发复位。具体可参见表 22-29 到表 22-34。

3.2 管脚复位

SPC2168 的 XRSTn 管脚提供了所有模块的全局复位，该管脚内部有上拉以确保芯片工作于正常状态。如表 3-1 所示，来自 XRSTn 管脚的信号会经过一个阈值可调整的除颤电路来消抖，仅当 XRSTn 保持为低的时间超过设定的阈值时才会触发有效的复位。默认的阈值为 500us。

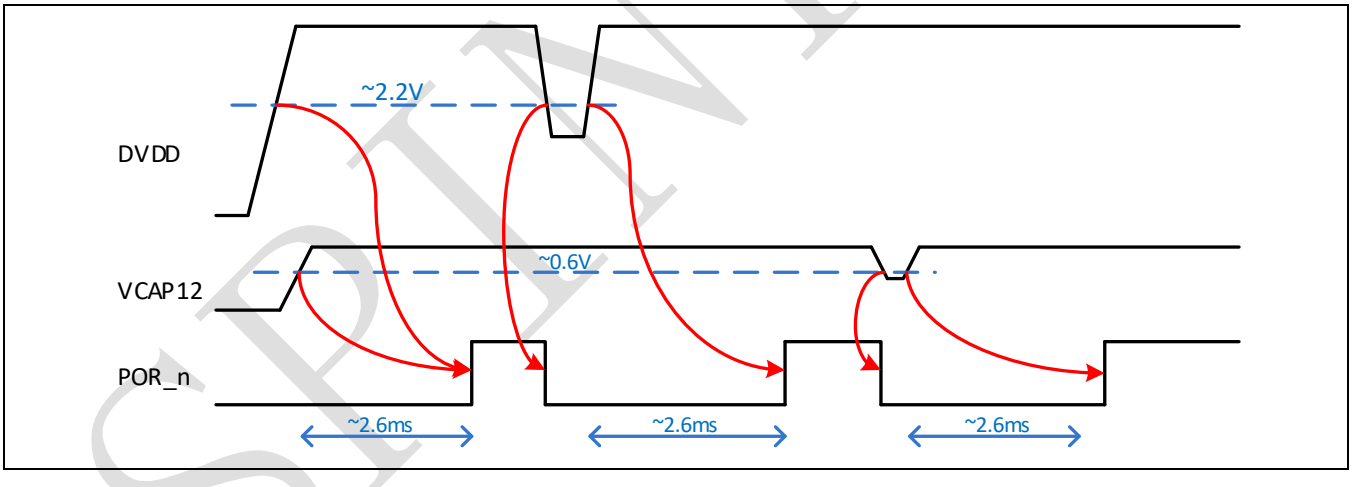
表 3-1：XRSTn 消抖阈值选项

PORCTL.XRSTFILT	阈值典型值
0	500us
1	1ms
2	2ms
3	4ms

3.3 上电复位

如图 3-2 所示，SPC2168 内建 POR 上电复位电路，确保芯片上电过程中时序无误，使芯片更易于使用。该复位在上电后会保持约 2.6ms，并且在 DVDD 低于 2.2V 或者 VCAP12 低于 0.6V 时会触发复位。

图 3-2：上电复位时序图

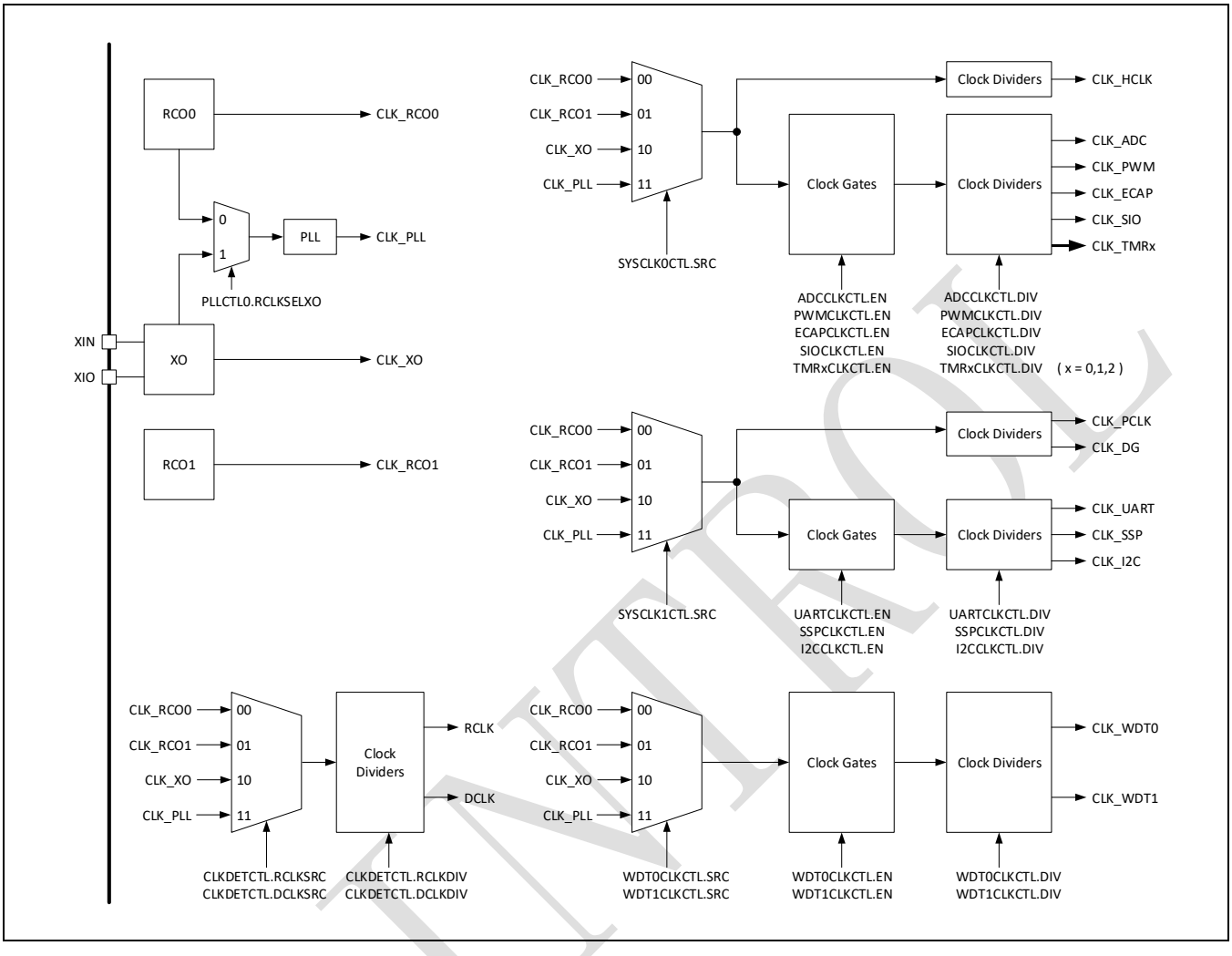


3.4 时钟结构

SPC2168 内置两个 RC 振荡器，并支持外部晶体或者有源时钟。片上还集成了 PLL 以提供从 25MHz 到 200MHz 的时钟，其输入参考时钟可以是 4MHz 到 56MHz。如图 3-3 所示，配合分频器和使能控制，SPC2168 可以非常灵活地实现对每个模块时钟的独立配置。

注意： 可以通过 GPIO11 来观测时钟，具体参见表 4-1。

图 3-3：时钟结构图



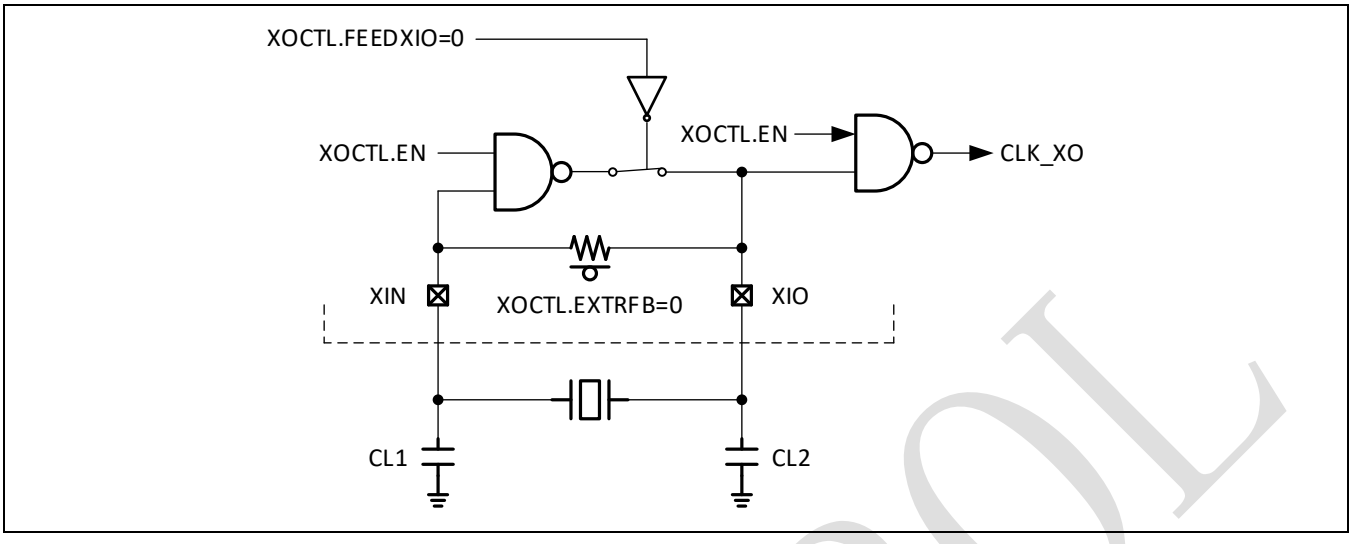
3.5 高频 RC 时钟

为了时钟安全，SPC2168 集成了两个 RC 振荡器。这两个 RC 时钟在出厂时都校准到 32MHz，而且都支持低频模式，以提供约 270kHz 的时钟来降低系统功耗。

3.6 内部晶振和外部时钟

SPC2168 支持内部晶振和外部时钟输入两种模式。如图 3-4 所示，对于内部晶振模式，电路板上需要在 XIN 和 XIO 两个管脚之间放置一个晶体。

图 3-4：内部晶振模式



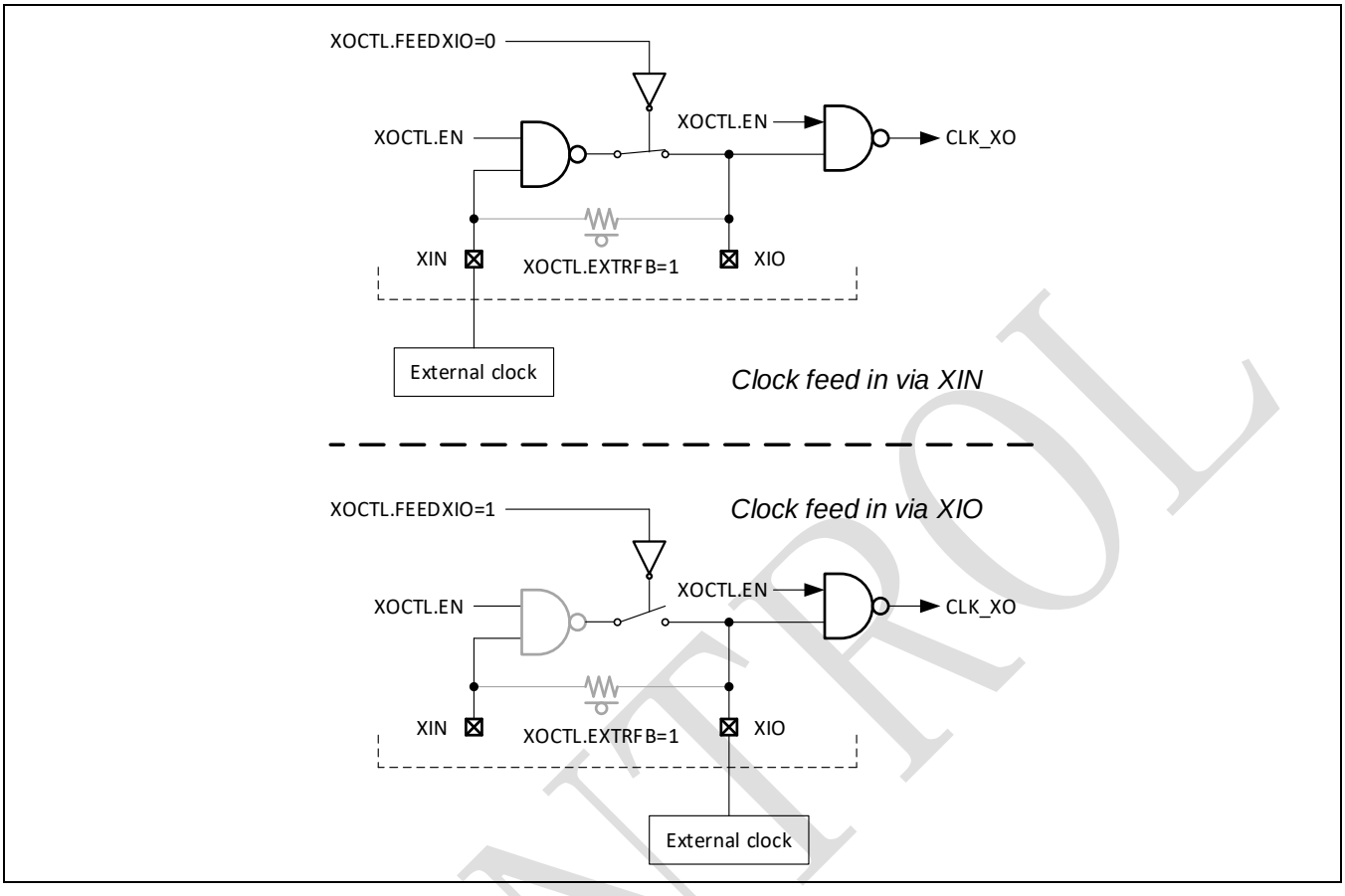
为了让内部反相器电路偏置在高增益区域，还需要在 XIN 和 XIO 之间放置一个典型值为 1MΩ 的反馈电阻。SPC2168 默认提供了内建反馈电阻以节约片外元件。请参考芯片数据手册外部时钟（XO）特性章节，来选择合适的晶体和负载电容（CL_eff）。其中，负载电容 CL_eff 为图 3-4 中 CL1 和 CL2 的串联值。

对于外部时钟模式，可以根据 XOCTL.FEEDXIO 的配置，通过 XIN 管脚（XIO 管脚保持悬空）或者 XIO 管脚（XIN 管脚可以用作常规 GPIO）送入时钟信号，详见图 3-5。

由于 XIN/XIO 管脚与 GPIO20/GPIO21 复用，必须在设定 XOCTL.EN 前先正确配置管脚通道。出于安全考虑，若 GPIO20/GPIO21 管脚通道没有被配置为有效的内部晶振或者外部时钟模式，用户将无法对 XOCTL.EN 置 1；反之，如果 XOCTL.EN 已经成功置 1（即 XO 时钟正常运行），此时用户将无法修改 GPIO20/GPIO21 的通道配置。

注意： 如果选择 CLK_XO 作为 PLL 输入参考时钟，外部时钟频率必须在 4MHz 到 56MHz 范围内。

图 3-5：外部时钟模式

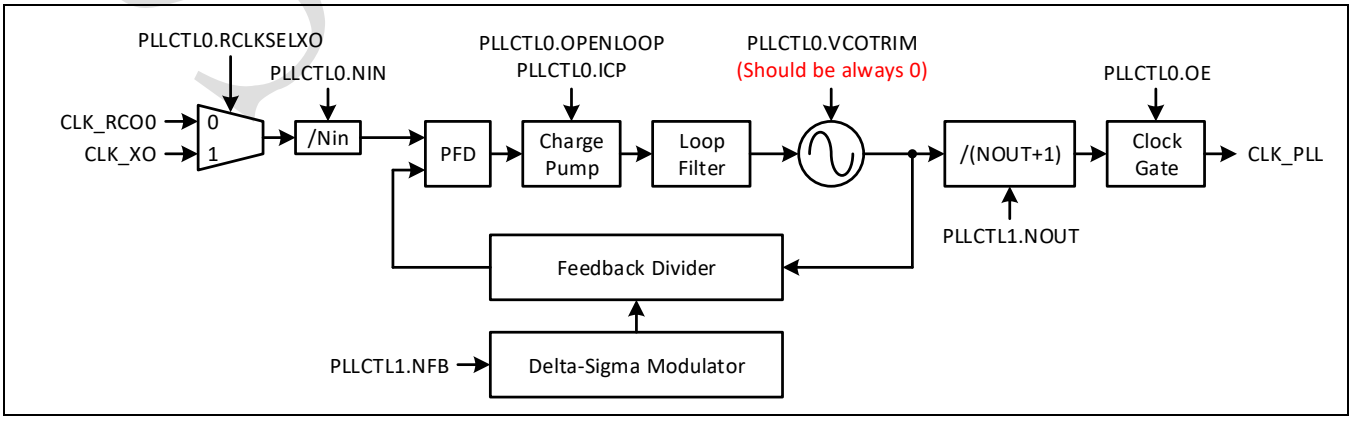


3.7 PLL 时钟

图 3-6 给出了 SPC2168 中所用 PLL（Phase-Locked Loop）的结构。它可以选择来自 RCO0 或者晶振的时钟作为输入参考。当环路锁定时，VCO 频率在 400MHz 到 600MHz 范围内，随后经过分频器进一步降至 25MHz 到 200MHz 范围。具体输出频率计算公式为：

$$f_{PLL} = \frac{f_{REF}}{NIN} \times \frac{NFB}{65536} \times \frac{1}{NOUT + 1}$$

图 3-6：PLL 结构总览



3.7.1 输入参考和分频

PLL 的输入参考时钟（Fin）为 4MHz 到 56MHz，最终送至鉴频鉴相器 PFD 的输入（Fpfd）为从 4MHz 到 8MHz。因此，输入分频比（Fin/Fpfd）应当如表 3-2 所示配置。

表 3-2：输入分频比设置

输入频率（MHz）	输入分频比
4~8	1
8~16	2
16~24	3
24~32	4
32~40	5
40~48	6
48~56	7

3.7.2 输出分频

VCO 的调频范围为 400MHz 到 600MHz，结合输出分频，最终时钟 CLK_PLL 的频率如表 3-3 所示。

表 3-3：输出时钟频率和分频比

输出频率			分频比（NOUT+1）
400MHz	~	600MHz	1
200MHz	~	300MHz	2
150MHz	~	200MHz	3
100MHz	~	150MHz	4
75MHz	~	100MHz	6
50MHz	~	75MHz	8
37.5MHz	~	50MHz	12
25MHz	~	37.5MHz	16

3.7.3 环路分频

基于由表 3-2 和表 3-3 得到的 NIN 和 NOUT，PLLCTL1 寄存器的 NFB 位段的值可以根据下面公式计算：

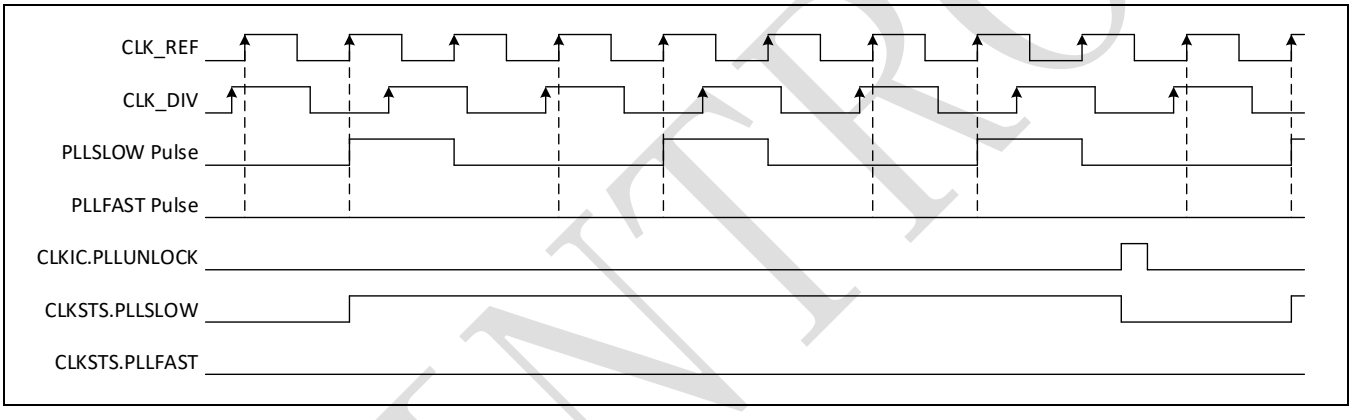
$$NFB = \frac{f_{PLL} \times (NOUT + 1)}{f_{REF} \,/\, NIN} \times 65536$$

3.7.4 PLL 失锁检测

SPC2168 PLL 的 PFD 中内嵌了环路失锁检测电路，以提供环路状态侦测信息。具体原理为：对于 PFD 的两个输入，如果在某一个时钟相邻上升沿之间，另一个时钟出现了两个及以上的上升沿，则产生 PLLFAST 或 PLLSLOW 脉冲，并更新 CLKSTS 寄存器。用户可以通过对 CLKIC.PLLUNLOCK 写 1 来清除这些标志位。图 3-7 给出了输入参考（FREF）快于分频器输出（FDIV）并使得 CLKSTS.PLLSLOW 置位的例子。

在 PLL 初始建立或者环路分频有很大改变的时候，FREF 和 FDIV 有较大差别，导致有不可避免的 PLLFAST 或者 PLLSLOW 事件被锁存到 CLKSTS 寄存器中。为了避免误报，使用时需要等待约 10us 以待环路锁定，随后对 CLKIC.PLLUNLOCK 写 1 以清除建立过程中可能误报的失锁，然后才能根据需要通过 CLKIE.PLLUNLOCK 来使能 PLL 失锁事件中断。一旦 PLL 锁定，将不应当再出现 CLKSTS.PLLFAST 或 CLKSTS.PLLSLOW 事件。

图 3-7：PLL 失锁检测时序图示例（FREF > FDIV）



3.8 PLL 和 RC 时钟稳压源

为了更好地将时钟生成电路与其他模块隔离，SPC2168 提供了专用于 PLL 和 RC 振荡器的稳压源。这也使得时钟产生电路的电源可以独立配置。

默认情况下，稳压源被旁路，时钟生成电路的电源轨直接与整个芯片的 1.2V 电源轨相连。

注意： 在使用 PLL 或者 RC 时钟稳压源之前，应确保稳压源已使能并且完全稳定（CLKSTS.CLKREGRDY=1）。

3.9 时钟选择和分频

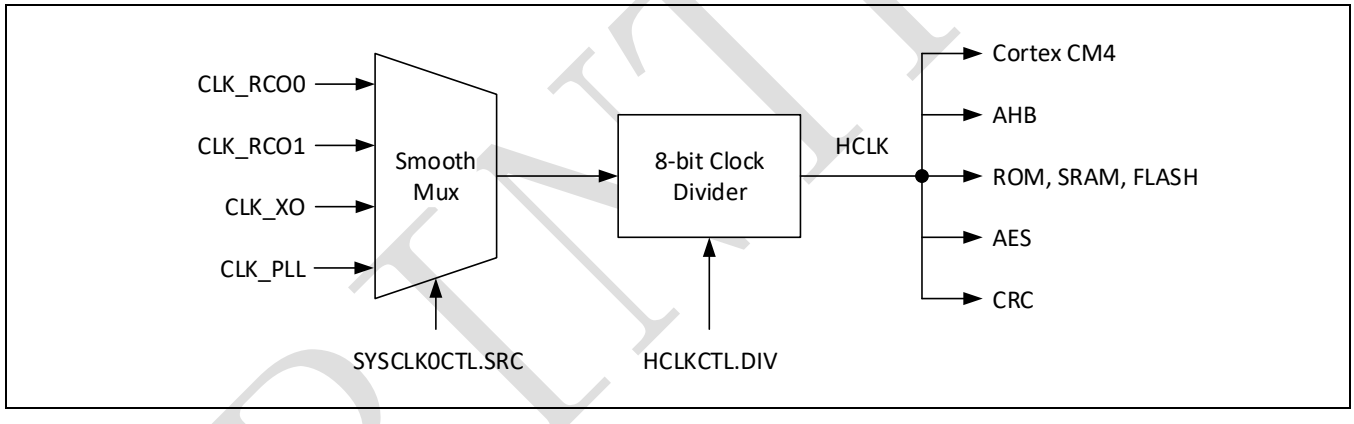
3.9.1 时钟平滑选择器

平滑选择器用于实现无毛刺的动态时钟切换。为了避免误切换到一个未开启的时钟并导致 CPU 卡死，直至目标时钟已经完全建立后，平滑选择器才会根据用户配置选择新的时钟，否则将一直选择当前时钟。如果当前选择的时钟源是 RCO0 且 RCO0 时钟失效，平滑选择器会选择 RCO1 作为当前时钟源，从而避免 CPU 卡死；于此同时，如果使能了时钟检测功能，将会产生一个时钟检测错误事件。该事件可以根据 CLKIE 和 CLKTZE 寄存器相应位的配置来触发 CPU 中断或者 PWM 锁定事件。

3.9.2 AHB 时钟（HCLK）

HCLK 是 SPC2168 最关键的时钟，它用于驱动 CPU、AHB 总线、存储单元、AES 和 CRC 模块。如图 3-8 所示，HCLK 由平滑选择器后经过分频产生，并保持常开。所允许的 HCLK 最高频率为 200MHz。

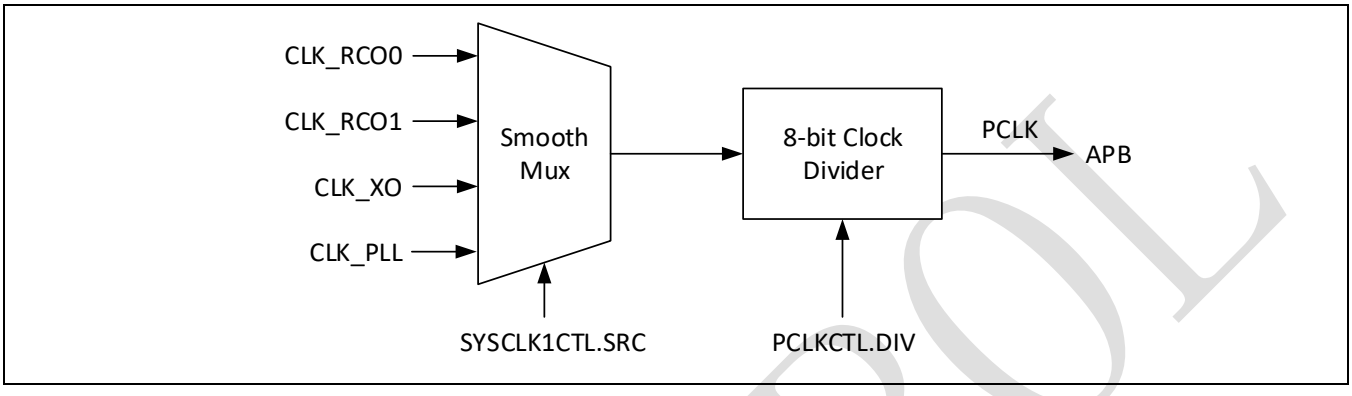
图 3-8：HCLK 控制框图



3.9.3 APB 时钟

如图 3-9 所示，APB 总线采用 PCLK 时钟，它同样由平滑选择后的常开时钟分频得到。所允许的 PCLK 最高频率是 50MHz。

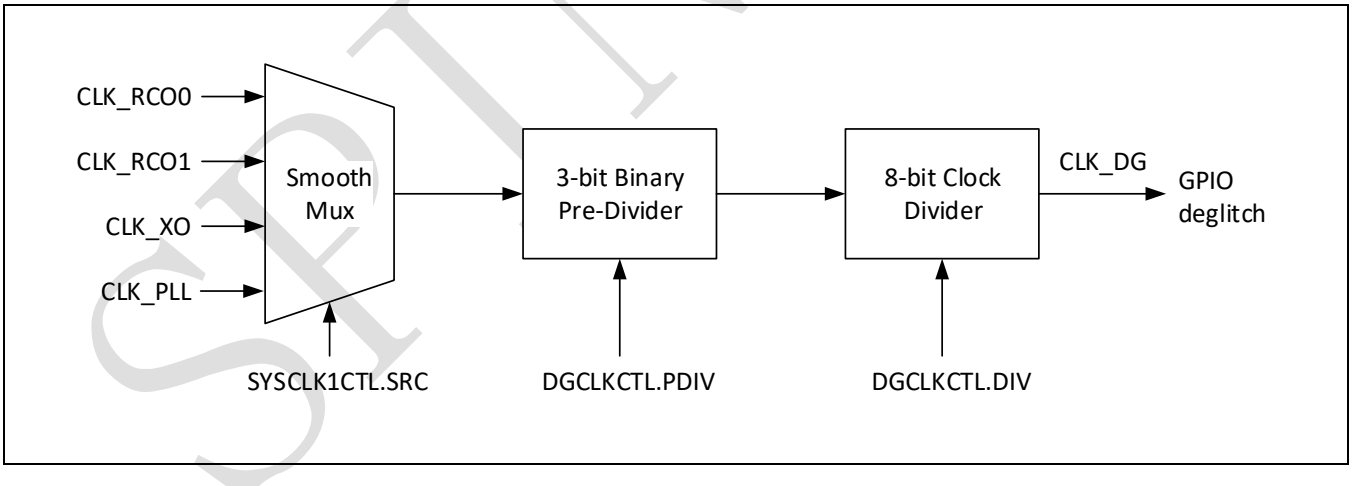
图 3-9：PCLK 控制框图



3.9.4 GPIO 消抖时钟

如图 3-10 所示，GPIO 消抖时钟由平滑选择后常开时钟，经过一次二进制分频和一次线性分频得到。它用于对 GPIO 输入重新采样和消抖以提高信号完整性。消抖滤波器会造成 3 个时钟周期的输入延迟。所允许的 CLK_DG 最高频率是 50MHz。

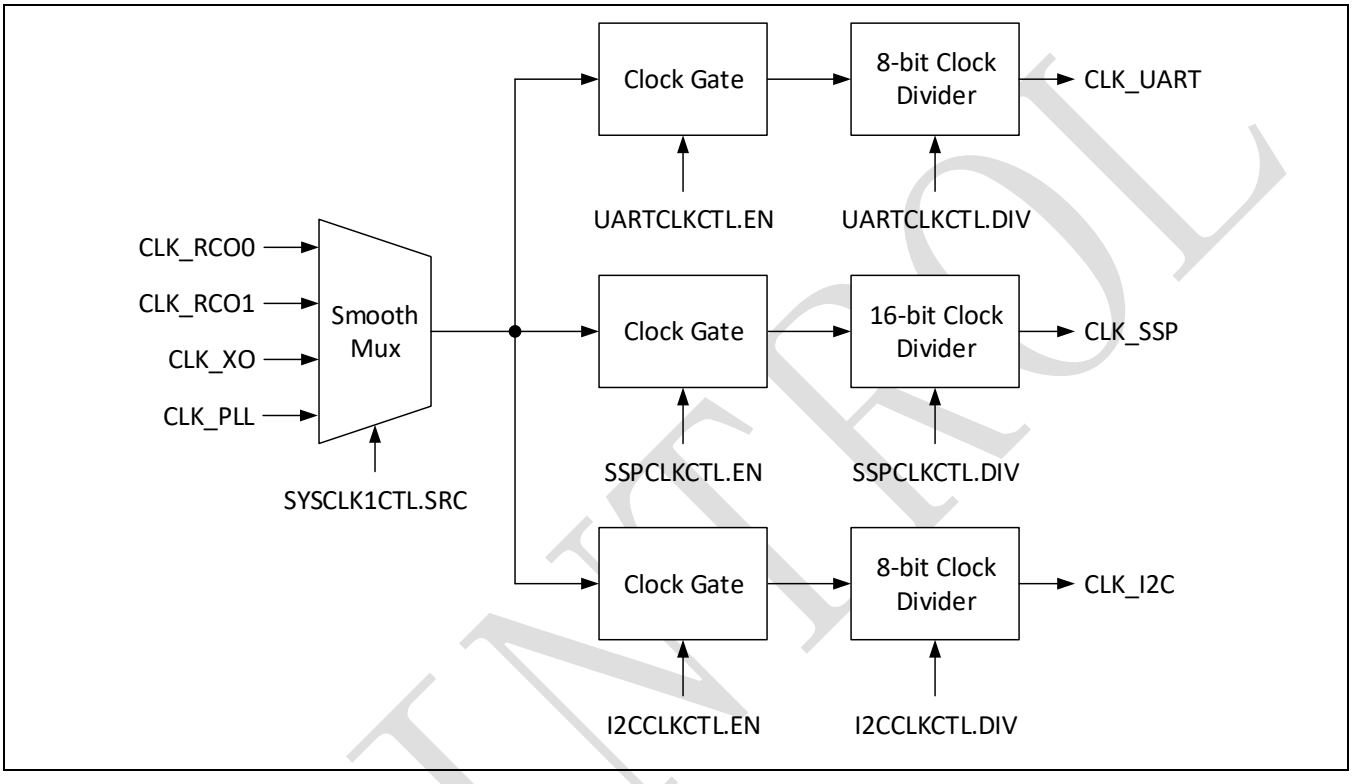
图 3-10：GPIO 消抖时钟控制框图



3.9.5 通讯外设时钟

SPC2168 提供了 SSP、UART 和 I2C 串行通讯接口。如图 3-11 所示，相应的时钟由平滑选择后的系统时钟 1（SYSCLK1）经过分频得到，并提供门控使能以降低系统功耗。所允许的 CLK_UART 最高频率是 200MHz，CLK_SSP 和 CLK_I2C 最高频率是 50MHz。

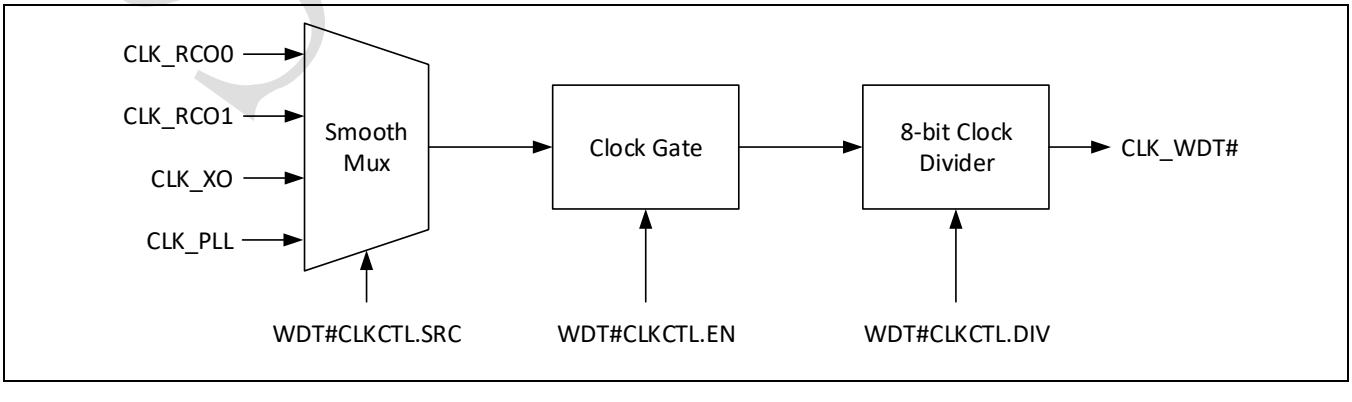
图 3-11：通讯外设时钟控制框图



3.9.6 看门狗定时器（WDT）时钟

SPC2168 提供了两个看门狗定时器。各自有独立的时钟，由相应的平滑选择、门控和分频产生，如图 3-12 所示。两个 WDT 时钟默认都是使能的。所允许的 WDT 时钟最高频率为 200MHz。

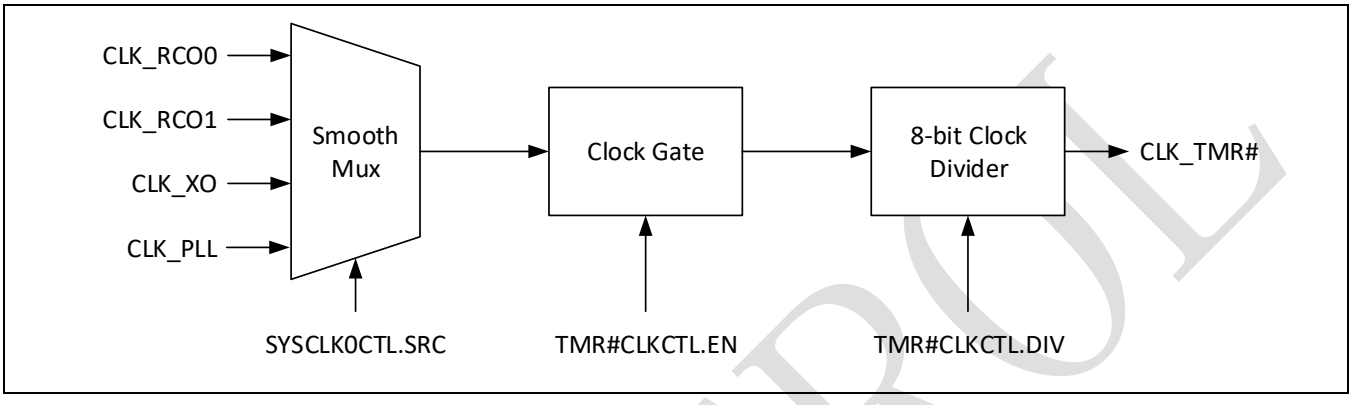
图 3-12：WDT 时钟控制框图



3.9.7 通用定时器时钟

SPC2168 提供了三个通用定时器。各自有独立的时钟，由平滑选择后的系统时钟 0（SYSCLK0）经过门控使能和分频后产生，如图 3-13 所示。所允许的通用定时器时钟最高频率是 200MHz。

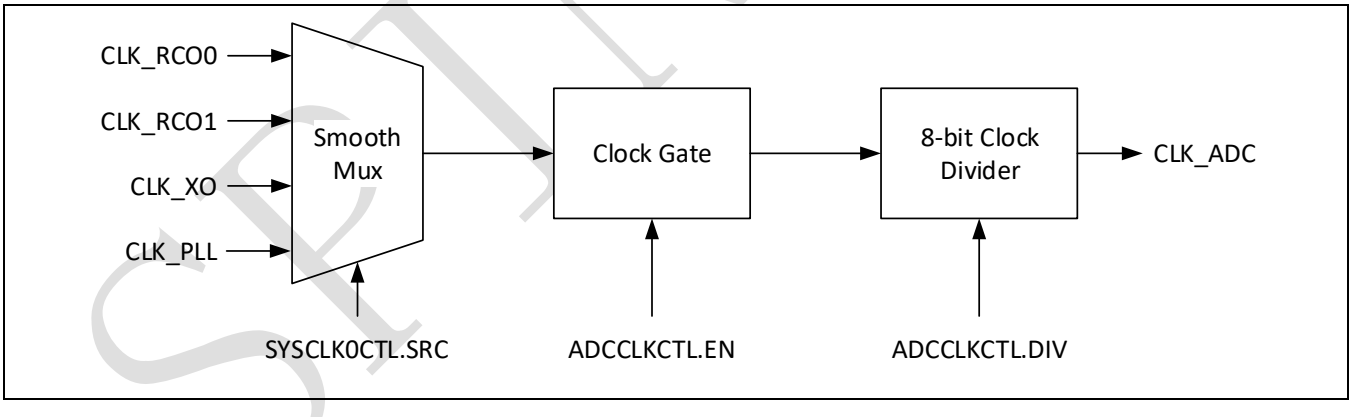
图 3-13：通用定时器时钟控制框图



3.9.8 ADC 时钟

如图 3-14 所示，ADC 时钟由平滑选择后的系统时钟 0（SYSCLK0）经过门控使能和分频产生。所允许的 ADC 时钟最高频率是 200MHz。

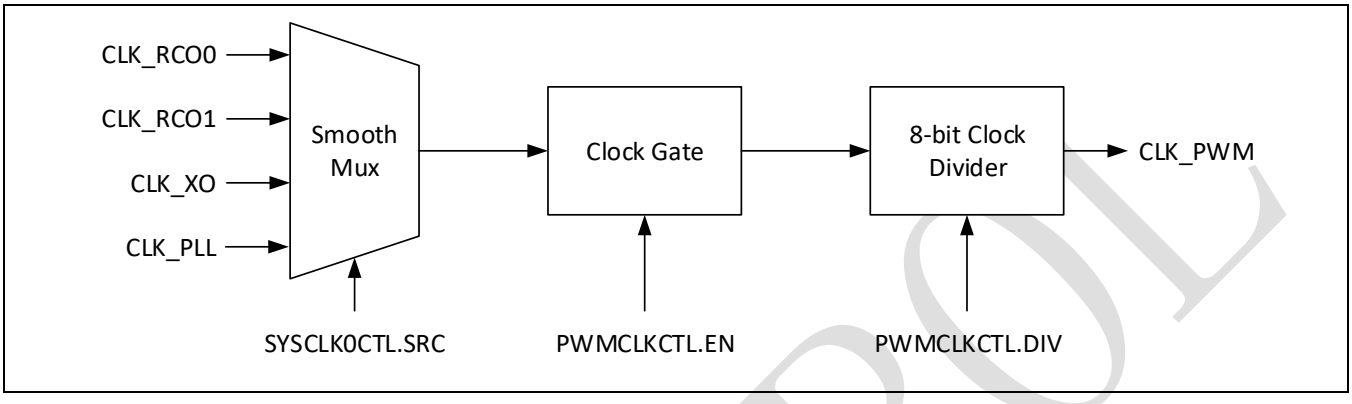
图 3-14：ADC 时钟控制框图



3.9.9 PWM 时钟

SPC2168 有 8 个 PWM 模块（编号为 0~7），并采用如图 3-15 所示的相同时钟，由平滑选择后的系统时钟 0（SYSCLK0）经过门控和分频产生。所允许的 PWM 时钟最高频率是 200MHz。

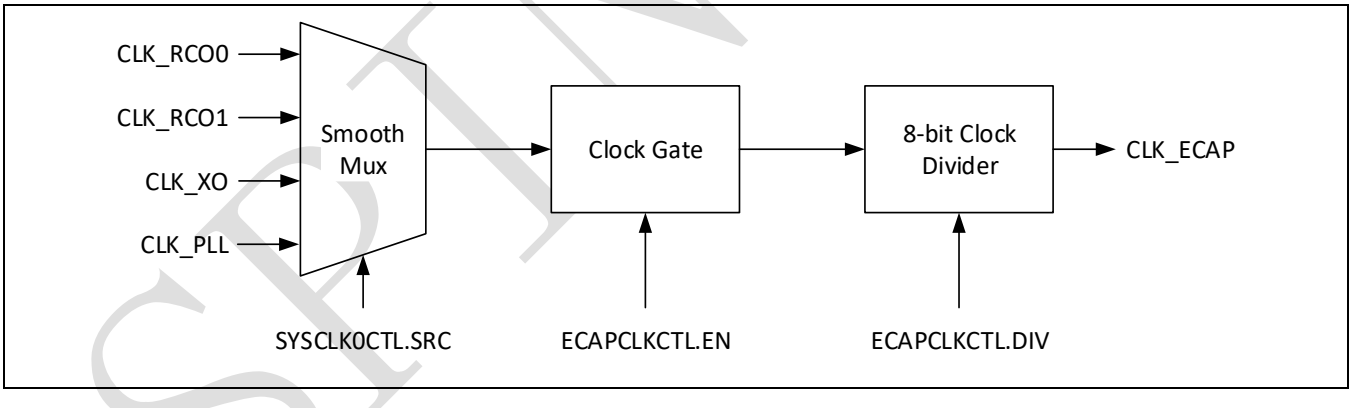
图 3-15：PWM 时钟控制框图



3.9.10 ECAP 时钟

如图 3-16 所示，ECAP 时钟由平滑选择后的系统时钟 0（SYSCLK0）经过门控和分频产生。所允许的 ECAP 时钟最高频率是 200MHz。

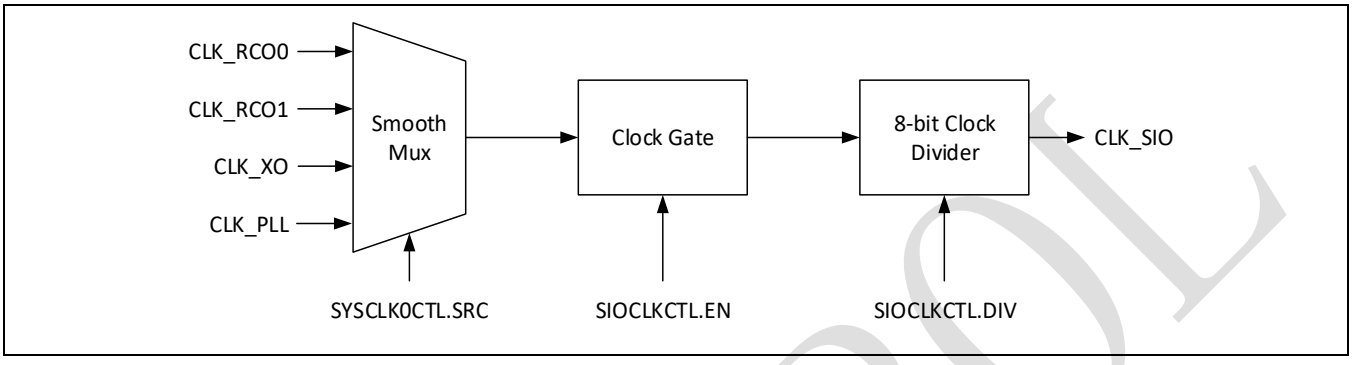
图 3-16：ECAP 时钟控制框图



3.9.11 SIO 时钟

如图 3-17 所示，SIO 时钟由平滑选择后的系统时钟 0（SYSCLK0）经过门控和分频产生。所允许的 SIO 时钟最高频率是 100MHz。

图 3-17：SIO 时钟控制框图



3.10 时钟安全

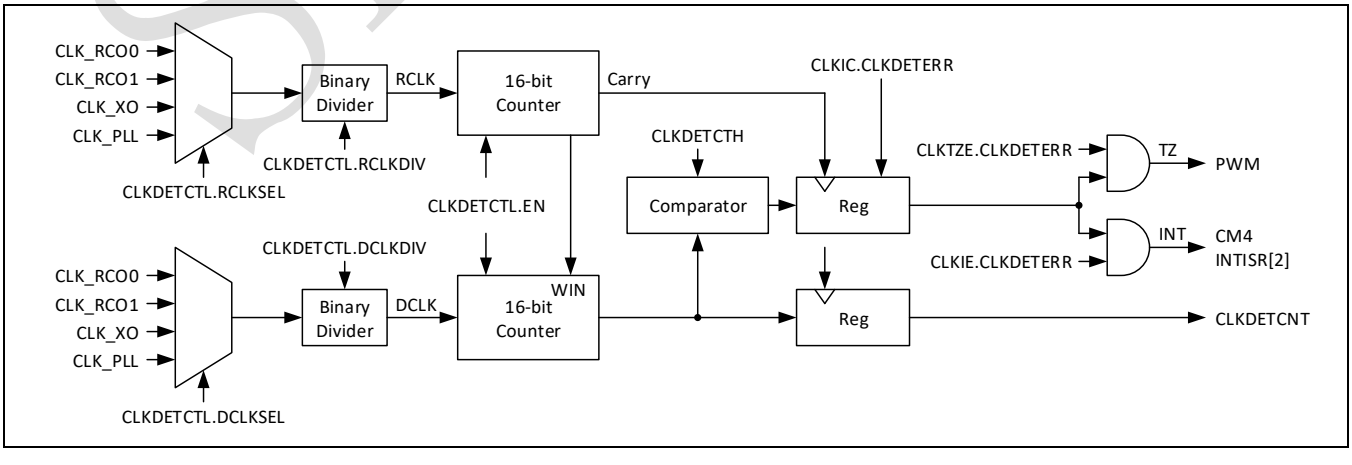
为了时钟安全，SPC2168 提供了时钟交叉检测模块。如图 3-18 所示，参考时钟和待检测时钟可以独立选择，随后各自经过一个二进制分频器得到 RCLK 和 DCLK。RCLK 用于产生观测窗口，在该窗口内 DCLK 保持计数。待观测窗口关闭后，DCLK 的最终计数值被锁存到 CLKDETCNT 寄存器，并与 CLKDETCTH 寄存器中的高低边界阈值进行比较，若计数越界则产生 CLKIF.CLKDETERR 标志位。计数值越界还可以根据 CLKIE 和 CLKTZE 寄存器相应位的配置来触发 CPU 中断或者 PWM 锁定事件。DCLK 可以通过配置 GPIO11.MUXSEL=3 或者 GPIO40.MUXSEL=3 后，在 GPIO11 管脚观测。

CLKDETCNT 的值可通过下面的公式计算：

$$\text{CLKDETCNT} = \frac{f_{DCLK}}{f_{RCLK}} \times 65536$$

其中，f_{DCLK}和f_{RCLK}分别是 DCLK 和 RCLK 的频率，如图 3-18 所示。

图 3-18：时钟交叉检测



3.11 寄存器

3.11.1 Clock 寄存器列表

表 3-4：CLOCK 模块基地址

外设模块	基地址
CLOCK	0x4000 0200

表 3-5：CLOCK 寄存器列表

寄存器	偏移地址	描述	复位值
CLKSTS	0x00	时钟状态寄存器	0x00000003
CLKREGCTL*	0x04	时钟稳压控制寄存器	0x0000001C
RCO0CTL*	0x08	RCO0 控制寄存器	0x00000579
RCO1CTL*	0x0C	RCO1 控制寄存器	0x00000579
XOCTL*	0x10	晶振控制寄存器	0x00000FF2
PLLCTL0*	0x14	PLL 控制寄存器 0	0x00000C40
PLLCTL1*	0x18	PLL 控制寄存器 1	0x0C320000
CLKDETCTL*	0x1C	时钟检测控制寄存器	0x00000000
CLKDETCTH*	0x20	时钟检测计数阈值寄存器	0x00000000
CLKDETCNT	0x24	时钟检测计数值寄存器	0x00000000
CLKIF	0x28	时钟中断标志寄存器	0x00000000
CLKIC	0x2C	时钟中断清除寄存器	0x00000000
CLKIE*	0x30	时钟中断使能寄存器	0x00000000
CLKTZE*	0x34	时钟封锁事件使能寄存器	0x00000000
SYSCLK0CTL*	0x38	SYSCLK0 时钟控制寄存器	0x00000000
HCLKCTL*	0x3C	HCLK 时钟控制寄存器	0x00000000
ADCCLKCTL*	0x40	ADC 时钟控制寄存器	0x00000000
PWMCLKCTL*	0x44	PWM 时钟控制寄存器	0x00000000
ECAPCLKCTL*	0x48	ECAP 时钟控制寄存器	0x00000000
TMR0CLKCTL*	0x4C	Timer 0 时钟控制寄存器	0x00000000
TMR1CLKCTL*	0x50	Timer 1 时钟控制寄存器	0x00000000
TMR2CLKCTL*	0x54	Timer 2 时钟控制寄存器	0x00000000
SIOCLKCTL*	0x58	SIO 时钟控制寄存器	0x00000000
SYSCLK1CTL*	0x5C	SYSCLK1 时钟控制寄存器	0x00000000
PCLKCTL*	0x60	PCLK 时钟控制寄存器	0x00000000
DGCLKCTL*	0x64	消抖时钟控制寄存器	0x00000200
UARTCLKCTL*	0x68	UART 时钟控制寄存器	0x00000000
SSPCLKCTL*	0x6C	SSP 时钟控制寄存器	0x00000000

寄存器	偏移地址	描述	复位值
I2CCLKCTL*	0x70	I2C 时钟控制寄存器	0x00000000
WDT0CLKCTL*	0x74	WDT0 时钟控制寄存器	0x00000100
WDT1CLKCTL*	0x78	WDT1 时钟控制寄存器	0x00000300
CLKREGKEY	0x7C	CLOCK 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 CLKREGKEY=0x1ACCE551 才可以改写。

3.11.2 Clock 寄存器

表 3-6 到表 3-69 给出了 CLOCK 模块各寄存器的定义。

表 3-6：时钟状态寄存器（CLKSTS）位段定义

CLKSTS (Clock Status Register) Offset: 0x0 Default: 0x00000003
Access: CLOCK -> CLKSTS.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						CLKREGRDY	CLKDETERR
7	6	5	4	3	2	1	0
PLLSLOW	PLLFAST	VCOFREQVLD	VCOFREQ	PLLRDY	XORDY	RCO1RDY	RCO0RDY

表 3-7：时钟状态寄存器（CLKSTS）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9	CLKREGRDY	RO	0x0	时钟稳压器就绪标志 0：未就绪 1：就绪
8	CLKDETERR	RO	0x0	CLKDET 模块检测到错误的标志 0：CLKDET 未检测到错误 1：CLKDET 检测到错误
7	PLLSLOW	RO	0x0	PLL 失锁且频率偏慢标志 可能的原因包括： （1） Fin/NIN*NFB/65536 高于 VCO 所能达到的 600MHz 上限频率 （2） VCO 未能起振 （3） 环路分频器功能异常，导致分频器没有输出，或者实际分频比高于预期值

位段	位段名	属性	复位值	描述
				0：PLL 从未发生过因为频率偏慢而失锁 1：PLL 失锁且频率偏慢
6	PLLFAST	RO	0x0	PLL 失锁且频率偏快标志 可能的原因包括： （1） Fin/NIN*NFB/65536 低于 VCO 所能达到的 400MHz 下限频率 （2） 未能成功提供输入参考时钟 （3） 环路分频器异常，导致实际分频比低于预期 0：PLL 从未发生过因为频率偏快而失锁 1：PLL 失锁且频率偏快
5	VCOFREQVLD	RO	0x0	VCOFREQ 位段读数有效标志 0：VCOFREQ 位段读数无效 1：VCOFREQ 位段读数有效
4	VCOFREQ	RO	0x0	VCO 频率标志位（用于校准） 0：VCO 偏慢 1：VCO 偏快
3	PLLRDY	RO	0x0	PLL 时钟就绪标志 0：未就绪 1：就绪
2	XORDY	RO	0x0	XO 时钟就绪标志 0：未就绪 1：就绪
1	RCO1RDY	RO	0x1	RCO1 时钟就绪标志 0：未就绪 1：就绪
0	RCO0RDY	RO	0x1	RCO0 时钟就绪标志 0：未就绪 1：就绪

表 3-8：时钟稳压控制寄存器（CLKREGCTL）位段定义

CLKREGCTL (Clock Regulator Control Register) Offset: 0x4 Default: 0x0000001C Access: CLOCK -> CLKREGCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		VREFTRIM				USEADCBG	EN

表 3-9：时钟稳压控制寄存器（CLKREGCTL）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5:2	VREFTRIM	RW	0x7	参考电压选择 0000：1.06V 0001：1.08V 0010：1.10V 0011：1.12V 0100：1.14V 0101：1.16V 0110：1.18V 0111：1.20V 1000：1.22V 1001：1.24V 1010：1.26V 1011：1.28V 1100：1.30V 1101：1.32V 1110：1.34V 1111：1.36V
1	USEADCBG	RW	0x0	使用 ADC 带隙（bandgap）基准提供偏置电流 0：偏置电流来自系统带隙基准 1：偏置电流来自 ADC 带隙基准
0	EN	RW	0x0	时钟稳压器使能 0：关闭 1：使能

表 3-10：RCO0 控制寄存器（RCO0CTL）位段定义

RCO0CTL (RCO0 Control Register) Offset: 0x8 Default: 0x00000579 Access: CLOCK -> RCO0CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				LOCALREG	FREQTRIM		
7	6	5	4	3	2	1	0
FREQTRIM						LFMODE	EN

表 3-11：RCO0 控制寄存器（RCO0CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11	LOCALREG	RW	0x0	本地时钟稳压器使能 0：RCO0 由全局电源 1.2V LDO 提供 1：RCO0 由本地电源 1.2V 时钟稳压器提供
10:2	FREQTRIM	RW	0x15E	RCO0 频率控制（出厂校准，不建议用户修改） 频率随控制字增加而提高
1	LFMODE	RW	0x0	RCO0 低频模式 0：常规模式（出厂校准到 32MHz） 1：低频模式，约 270kHz（不做校准）
0	EN	RW	0x1	RCO0 使能 0：关闭 1：使能

表 3-12：RCO1 控制寄存器（RCO1CTL）位段定义

RCO1CTL (RCO1 Control Register) Offset: 0xC Default: 0x00000579 Access: CLOCK -> RCO1CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				LOCALREG	FREQTRIM		
7	6	5	4	3	2	1	0
FREQTRIM						LFMODE	EN

表 3-13：RCO1 控制寄存器（RCO1CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11	LOCALREG	RW	0x0	本地时钟稳压器使能 0：RCO1 由全局电源 1.2V LDO 提供 1：RCO1 由本地电源 1.2V 时钟稳压器提供
10:2	FREQTRIM	RW	0x15E	RCO1 频率控制（出厂校准，不建议用户修改） 频率随控制字增加而提高
1	LFMODE	RW	0x0	RCO1 低频模式 0：常规模式（出厂校准到 32MHz） 1：低频模式，约 270kHz（不做校准）
0	EN	RW	0x1	RCO1 使能 0：关闭 1：使能

表 3-14：晶振控制寄存器（XOCTL）位段定义

XOCTL (Crystal Oscillator Control Register) Offset: 0x10 Default: 0x00000FF2 Access: CLOCK -> XOCTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_14							
23	22	21	20	19	18	17	16
RESERVED_31_14							
15	14	13	12	11	10	9	8
RESERVED_31_14		FEEDXIO	EXTRFB	PRECNT			
7	6	5	4	3	2	1	0
PRECNT				RESERVED_3_2		FASTEN	EN

表 3-15：晶振控制寄存器（XOCTL）位段描述

位段	位段名	属性	复位值	描述
31:14	RESERVED_31_14	RO	0x0	保留
13	FEEDXIO	RW	0x0	外部时钟模式时输入管脚选择 0：外部时钟由 XIN 输入（XIO 保持悬空） 1：外部时钟由 XIO 输入（XIN 可用作 GPIO）
12	EXTRFB	RW	0x0	反馈电阻选择 0：采用内置反馈电阻 1：采用外部反馈电阻
11:4	PRECNT	RW	0xFF	晶振时钟就绪前预计数 1024 x PRECNT 个周期
3:2	RESERVED_3_2	RW	0x0	保留
1	FASTEN	RW	0x1	噪声输入使能以加速起振 0：关闭 1：使能
0	EN	RW	0x0	晶振时钟使能 0：关闭 1：使能

表 3-16：PLL 控制寄存器 0（PLLCTL0）位段定义

PLLCTL0 (PLL Control Register 0) Offset: 0x14 Default: 0x00000C40 Access: CLOCK -> PLLCTL0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 LOCALREG	14 USEADCBG	13	12	11 ICP	10	9	8 OPENLOOP
7 RCLKSELXO	6	5	4	3 FCALWIN	2 FCALEN	1 OE	0 EN
	VCOTRIM						

表 3-17：PLL 控制寄存器 0（PLLCTL0）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	LOCALREG	RW	0x0	本地时钟稳压器使能 0：PLL 由全局电源 1.2V LDO 提供 1：PLL 由本地电源 1.2V 时钟稳压器提供
14	USEADCBG	RW	0x0	使用 ADC 带隙提供偏置电流 0：偏置电流来自系统带隙 1：偏置电流来自 ADC 带隙
13:9	ICP	RW	0x6	PLL 电荷泵设置 应当设定为 0.96*Fvco/Fpfd/(14-FREQTRIM)
8	OPENLOOP	RW	0x0	PLL 开环模式（用于校准 VCO） 0：闭环模式 1：开环模式
7	RCLKSELXO	RW	0x0	PLL 参考时钟选择 0：CLK_RCO0 1：CLK_XO
6:4	VCOTRIM	RW	0x0	必须保持写 0
3	FCALWIN	RW	0x0	PLL 频率校准时间窗（本功能为内部测试用） 在每一个标定周期，保持为 1 直到 VCOFREQVLD=1，然后再设定为 0. 0：时间窗关闭 1：时间窗开启
2	FCALEN	RW	0x0	PLL 频率校准使能（本功能为内部测试用） 0：关闭 1：使能
1	OE	RW	0x0	PLL 数字时钟输出使能 0：输出关闭 1：输出使能

位段	位段名	属性	复位值	描述
0	EN	RW	0x0	PLL 使能 0：关闭 1：使能

表 3-18：PLL 控制寄存器 1（PLLCTL1）位段定义

PLLCTL1 (PLL Control Register 1) Offset: 0x18 Default: 0x0C320000
Access: CLOCK -> PLLCTL1.all

31	30	29	28	27	26	25	24
RESERVED	NOUT				NIN		
23	22	21	20	19	18	17	16
NFB							
15	14	13	12	11	10	9	8
NFB							
7	6	5	4	3	2	1	0
NFB							

表 3-19：PLL 控制寄存器 1（PLLCTL1）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:27	NOUT	RW	0x1	输出分频比设定：Fvco/Fpll = NOUT+1
26:24	NIN	RW	0x4	输入分频比设定 000：无效 001：Fin(MHz) 位于[4, 8]则设为 1，即 Fin/Fpfd=1 010：Fin(MHz) 位于(8, 16]则设为 2，即 Fin/Fpfd=2 011：Fin(MHz) 位于(16, 24]则设为 3，即 Fin/Fpfd=3 100：Fin(MHz) 位于(24, 32]则设为 4，即 Fin/Fpfd=4 101：Fin(MHz) 位于(32, 40]则设为 5，即 Fin/Fpfd=5 110：Fin(MHz) 位于(40, 48]则设为 6，即 Fin/Fpfd=6 111：Fin(MHz) 位于(48, 56]则设为 7，即 Fin/Fpfd=7
23:0	NFB	RW	0x320000	环路分频比设定：Fvco/Fpfd = NFB/65536

表 3-20：时钟检测控制寄存器（CLKDETCTL）位段定义

CLKDETCTL (Clock Detection Control Register) Offset: 0x1C Default: 0x00000000 Access: CLOCK -> CLKDETCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					EN	DCLKDIV	
7	6	5	4	3	2	1	0
DCLKDIV	DCLKSEL		RCLKDIV			RCLKSEL	

表 3-21：时钟检测控制寄存器（CLKDETCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	EN	RW	0x0	时钟检测使能 0：关闭时钟检测，同时清空中断和错误标志 1：开启时钟检测
9:7	DCLKDIV	RW	0x0	待检测时钟的分频比 000：1 分频（不分频） 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频
6:5	DCLKSEL	RW	0x0	待检测时钟的选择 00：CLK_RCO0 01：CLK_RCO1 10：CLK_XO 11：CLK_PLL
4:2	RCLKDIV	RW	0x0	参考时钟的分频比 000：1 分频（不分频） 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频

位段	位段名	属性	复位值	描述
1:0	RCLKSEL	RW	0x0	参考时钟的选择 00：CLK_RCO0 01：CLK_RCO1 10：CLK_XO 11：CLK_PLL

表 3-22：时钟检测计数阈值寄存器（CLKDETCTH）位段定义

CLKDETCTH (Clock Detection Counter Threshold Register) Offset: 0x20 Default: 0x00000000 Access: CLOCK -> CLKDETCTH.all							
31	30	29	28	27	26	25	24
HI							
23	22	21	20	19	18	17	16
HI							
15	14	13	12	11	10	9	8
LO							
7	6	5	4	3	2	1	0
LO							

表 3-23：时钟检测计数阈值寄存器（CLKDETCTH）位段描述

位段	位段名	属性	复位值	描述
31:16	HI	RW	0x0	时钟检测计数上边界 计数值高于上边界时报错，并可产生 PWM 封锁
15:0	LO	RW	0x0	时钟检测计数下边界 计数值低于下边界时报错，并可产生 PWM 封锁

表 3-24：时钟检测计数值寄存器（CLKDETCNT）位段定义

CLKDETCNT (Clock Detection Counter Register) Offset: 0x24 Default: 0x00000000 Access: CLOCK -> CLKDETCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 3-25：时钟检测计数值寄存器（CLKDETCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	时钟检测计数终值

表 3-26：时钟中断标志寄存器（CLKIF）位段定义

CLKIF (Clock Interrupt Flag Register)　Offset: 0x28　Default: 0x00000000
Access: CLOCK -> CLKIF.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					CLKINT	CLKDETERR	PLLUNLOCK

表 3-27：时钟中断标志寄存器（CLKIF）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	CLKINT	RO	0x0	全局时钟中断标志 0：时钟中断未发生 1：时钟中断曾发生 该位被清零前不会产生新的中断。
1	CLKDETERR	RO	0x0	CLKDET 模块检测到时钟错误标志 0：CLKDETCNT 在 CLKDETCTH 指定的范围内 1：CLKDETCNT 超出 CLKDETCTH 指定的范围 该位段清零前，相同事件不会再触发中断。
0	PLLUNLOCK	RO	0x0	PLL 失锁标志，细节可查询 CLKSTS.PLLFAST 和 CLKSTS.PLLSLOW 0：PLL 锁定 1：PLL 失锁 该位段清零前，PLL 失锁不会再触发中断。

表 3-28：时钟中断清除寄存器（CLKIC）位段定义

CLKIC (Clock Interrupt Clear Register) Offset: 0x2C Default: 0x00000000 Access: CLOCK -> CLKIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					CLKINT	CLKDETERR	PLLUNLOCK

表 3-29：时钟中断清除寄存器（CLKIC）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	CLKINT	W1C	0x0	全局时钟中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 CLKIF.CLKINT 标志位 本位段自动清零
1	CLKDETERR	W1C	0x0	CLKDET 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 CLKIF.CLKDETERR 标志位 本位段自动清零
0	PLLUNLOCK	W1C	0x0	PLL 失锁中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 CLKIF.PLLUNLOCK、CLKSTS.PLLFAST、CLKSTS.PLLSLOW 标志位 本位段自动清零

表 3-30：时钟中断使能寄存器（CLKIE）位段定义

CLKIE (Clock Interrupt Enable Register) Offset: 0x30 Default: 0x00000000 Access: CLOCK -> CLKIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						CLKDETERR	PLLUNLOCK

表 3-31：时钟中断使能寄存器（CLKIE）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	CLKDETERR	RW	0x0	CLKDET 错误中断使能 0：CLKDET 错误不触发中断 1：CLKDET 错误触发中断
0	PLLUNLOCK	RW	0x0	PLL 失锁中断使能 0：PLL 失锁不触发中断 1：PLL 失锁触发中断

表 3-32：时钟封锁事件使能寄存器（CLKTZE）位段定义

CLKTZE (Clock Trip-zone Event Enable Register)　Offset: 0x34　Default: 0x00000000
Access: CLOCK -> CLKTZE.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						CLKDETERR	PLLUNLOCK

表 3-33：时钟封锁事件使能寄存器（CLKTZE）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	CLKDETERR	RW	0x0	CLKDET 错误封锁事件使能 0：CLKDET 错误不触发 PWM 封锁 1：CLKDET 错误触发 PWM 封锁
0	PLLUNLOCK	RW	0x0	PLL 失锁封锁事件使能 0：PLL 失锁不触发 PWM 封锁 1：PLL 失锁触发 PWM 封锁

表 3-34：SYSCLK0 控制寄存器（SYSCLK0CTL）位段定义

SYSCLK0CTL (SYSCLK0 Control Register) Offset: 0x38 Default: 0x00000000 Access: CLOCK -> SYSCLK0CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SRC		RESERVED
7	6	5	4	3	2	1	0
RESERVED							

表 3-35：SYSCLK0 控制寄存器（SYSCLK0CTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10:9	SRC	RW	0x0	时钟源选择 00：RCO0 01：RCO1 10：XO 11：PLL
8:0	RESERVED_8_0	RO	0x0	保留

表 3-36：HCLK 控制寄存器（HCLKCTL）位段定义

HCLKCTL (HCLK Control Register) Offset: 0x3C Default: 0x00000000 Access: CLOCK -> HCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
DIV							

表 3-37：HCLK 控制寄存器（HCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-38：ADC 时钟控制寄存器（ADCCLKCTL）位段定义

ADCCLKCTL (ADC Clock Control Register) Offset: 0x40 Default: 0x00000000 Access: CLOCK -> ADCCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-39：ADC 时钟控制寄存器（ADCCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-40：PWM 时钟控制寄存器（PWMCLKCTL）位段定义

PWMCLKCTL (PWM Clock Control Register) Offset: 0x44 Default: 0x00000000 Access: CLOCK -> PWMCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-41：PWM 时钟控制寄存器（PWMCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-42：ECAP 时钟控制寄存器（ECAPCLKCTL）位段定义

ECAPCLKCTL (ECAP Clock Control Register) Offset: 0x48 Default: 0x00000000 Access: CLOCK -> ECAPCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-43：ECAP 时钟控制寄存器（ECAPCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-44：Timer 0 时钟控制寄存器（TMR0CLKCTL）位段定义

TMR0CLKCTL (Timer 0 Clock Control Register) Offset: 0x4C Default: 0x00000000 Access: CLOCK -> TMR0CLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-45：Timer 0 时钟控制寄存器（TMR0CLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-46：Timer 1 时钟控制寄存器（TMR1CLKCTL）位段定义

TMR1CLKCTL (Timer 1 Clock Control Register) Offset: 0x50 Default: 0x00000000 Access: CLOCK -> TMR1CLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-47：Timer 1 时钟控制寄存器（TMR1CLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-48：Timer 2 时钟控制寄存器（TMR2CLKCTL）位段定义

TMR2CLKCTL (Timer 2 Clock Control Register) Offset: 0x54 Default: 0x00000000 Access: CLOCK -> TMR2CLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-49：Timer 2 时钟控制寄存器（TMR2CLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-50：SIO 时钟控制寄存器（SIOCLKCTL）位段定义

SIOCLKCTL (SIO Clock Control Register) Offset: 0x58 Default: 0x00000000 Access: CLOCK -> SIOCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-51：SIO 时钟控制寄存器（SIOCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK0 的分频比为（DIV+1）

表 3-52：SYSCLK1 控制寄存器（SYSCLK1CTL）位段定义

SYSCLK1CTL (SYSCLK1 Control Register) Offset: 0x5C Default: 0x00000000 Access: CLOCK -> SYSCLK1CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SRC		RESERVED
7	6	5	4	3	2	1	0
RESERVED							

表 3-53：SYSCLK1 控制寄存器（SYSCLK1CTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10:9	SRC	RW	0x0	时钟源选择 00：RCO0 01：RCO1 10：XO 11：PLL
8:0	RESERVED_8_0	RO	0x0	保留

表 3-54：PCLK 控制寄存器（PCLKCTL）位段定义

PCLKCTL (PCLK Control Register) Offset: 0x60 Default: 0x00000000 Access: CLOCK -> PCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
DIV							

表 3-55：PCLK 控制寄存器（PCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	DIV	RW	0x0	设定相对于 SYSCLK1 的分频比为（DIV+1）

表 3-56：消抖时钟控制寄存器（DGCLKCTL）位段定义

DGCLKCTL (Deglitch Clock Control Register) Offset: 0x64 Default: 0x00000200 Access: CLOCK -> DGCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					PDIV		
7	6	5	4	3	2	1	0
DIV							

表 3-57：消抖时钟控制寄存器（DGCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10:8	PDIV	RW	0x2	相对 SYSCLK1 的预分频 000：1 分频（不分频） 001：2 分频 010：4 分频 011：8 分频 100：16 分频 101：32 分频 110：64 分频 111：128 分频
7:0	DIV	RW	0x0	设置预分频后的进一步分频比为（DIV+1）

表 3-58：UART 时钟控制寄存器（UARTCLKCTL）位段定义

UARTCLKCTL (UART Clock Control Register) Offset: 0x68 Default: 0x00000000 Access: CLOCK -> UARTCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-59：UART 时钟控制寄存器（UARTCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK1 的分频比为（DIV+1）

表 3-60：SSP 时钟控制寄存器（SSPCLKCTL）位段定义

SSPCLKCTL (SSP Clock Control Register) Offset: 0x6C Default: 0x00000000 Access: CLOCK -> SSPCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							EN
15	14	13	12	11	10	9	8
DIV							
7	6	5	4	3	2	1	0
DIV							

表 3-61：SSP 时钟控制寄存器（SSPCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:17	RESERVED_31_17	RO	0x0	保留
16	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
15:0	DIV	RW	0x0	设定相对于 SYSCLK1 的分频比为（DIV+1）

表 3-62：I2C 时钟控制寄存器（I2CCLKCTL）位段定义

I2CCLKCTL (I2C Clock Control Register) Offset: 0x70 Default: 0x00000000 Access: CLOCK -> I2CCLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							EN
7	6	5	4	3	2	1	0
DIV							

表 3-63：I2C 时钟控制寄存器（I2CCLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	EN	RW	0x0	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于 SYSCLK1 的分频比为（DIV+1）

表 3-64：WDT0 时钟控制寄存器（WDT0CLKCTL）位段定义

WDT0CLKCTL (WDT0 Clock Control Register) Offset: 0x74 Default: 0x00000100 Access: CLOCK -> WDT0CLKCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SRC		EN
7	6	5	4	3	2	1	0
DIV							

表 3-65：WDT0 时钟控制寄存器（WDT0CLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10:9	SRC	RW	0x0	时钟源选择 00：RCO0 01：RCO1 10：XO 11：PLL
8	EN	RW	0x1	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于时钟源的分频比为（DIV+1）

表 3-66：WDT1 时钟控制寄存器（WDT1CLKCTL）位段定义

WDT1CLKCTL (WDT1 Clock Control Register)　Offset: 0x78　Default: 0x00000300
Access: CLOCK -> WDT1CLKCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SRC		EN
7	6	5	4	3	2	1	0
DIV							

表 3-67：WDT1 时钟控制寄存器（WDT1CLKCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10:9	SRC	RW	0x1	时钟源选择 00：RCO0 01：RCO1 10：XO 11：PLL
8	EN	RW	0x1	时钟输出使能 0：关闭时钟 1：使能时钟
7:0	DIV	RW	0x0	设定相对于时钟源的分频比为（DIV+1）

表 3-68：CLOCK 模块写使能寄存器（CLKREGKEY）位段定义

CLKREGKEY (Clock Register Write-Allow Key Register) Offset: 0x7C Default: 0x1ACCE551 Access: CLOCK -> CLKREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 3-69：CLOCK 模块写使能寄存器（CLKREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 CLOCK 寄存器

4 通用 I/O（GPIO）

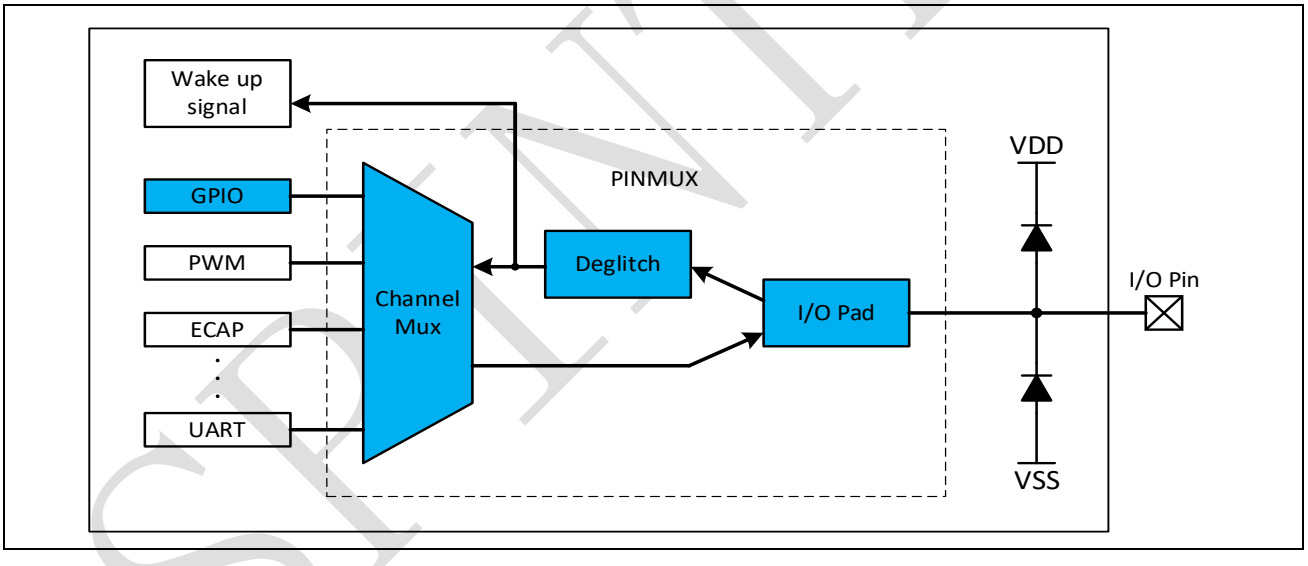
4.1 概述

SPC2168 具有 61 个 I/O 引脚（BOOT 引脚也可以用作 GPIO47，当然有一些限制）。这些 I/O 引脚是多路复用的，因而可以被配置为 GPIO 功能或者任意一个可选功能。每个 I/O 引脚都可以独立控制。

本章节详细描述 I/O 引脚的控制，包括引脚复用和 GPIO 相关的功能。I/O 控制的结构框图如图 4-1 所示，由两部分组成：

- PINMUX 模块
  - I/O Pad：上拉和下拉控制，输入/输出控制，施密特输入控制以及输出驱动强度控制
  - Deglitch：输入消抖控制
  - MUX：引脚多路复用
- GPIO 模块
  - GPIO 功能 IP 控制

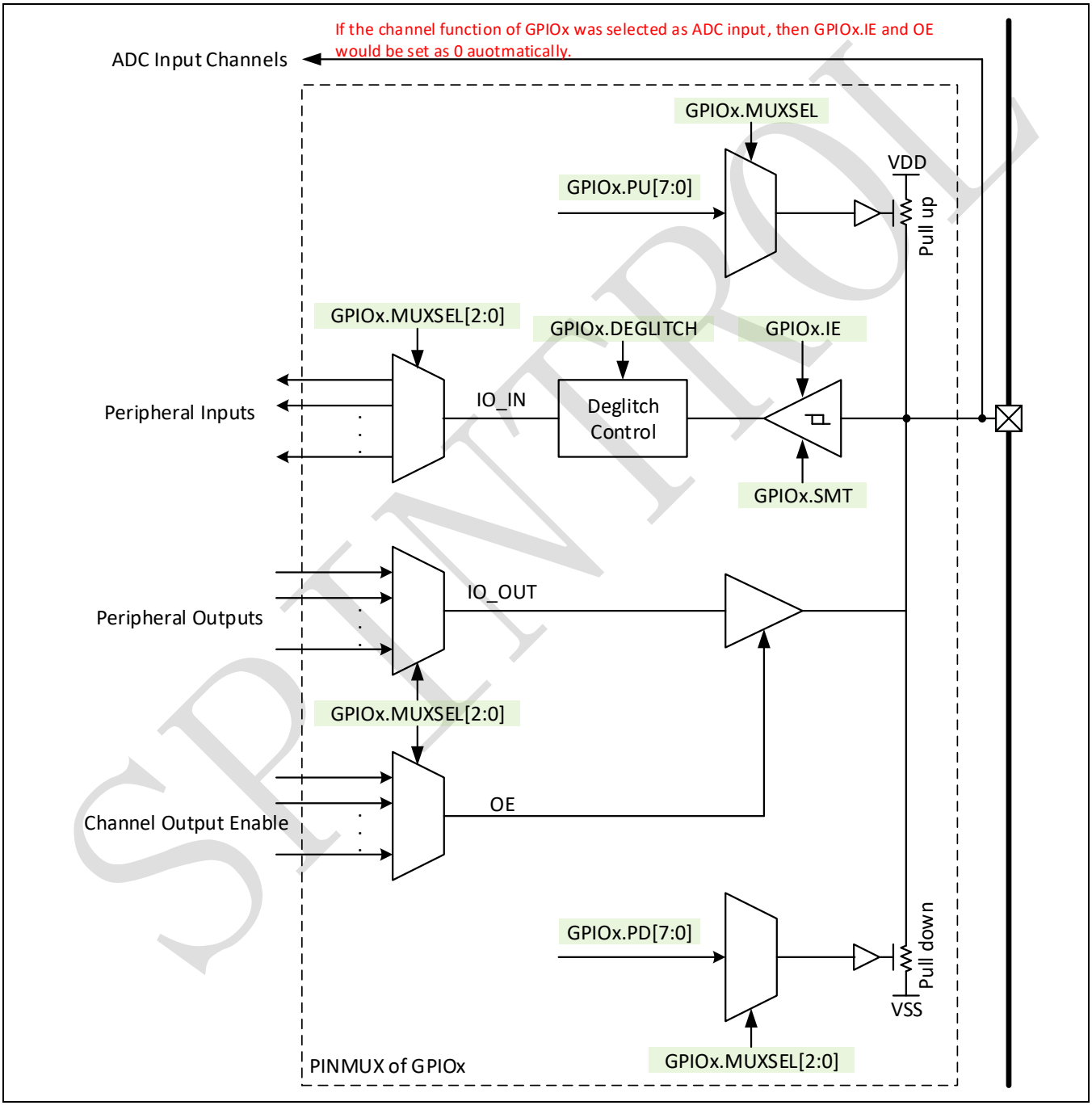
图 4-1：I/O 控制结构框图



4.2 引脚复用（PINMUX）

PINMUX 的结构框图如图 4-2 所示。每个 I/O 引脚的多路选择器支持 8 个功能通道。每个功能通道对应不同功能。当 I/O 引脚处于输入模式下，输出缓冲器禁用，并处于高阻态；可以通过配置毛刺信号抑制模块或者施密特触发器模块对输入信号进行过滤。

图 4-2：PINMUX 结构框图



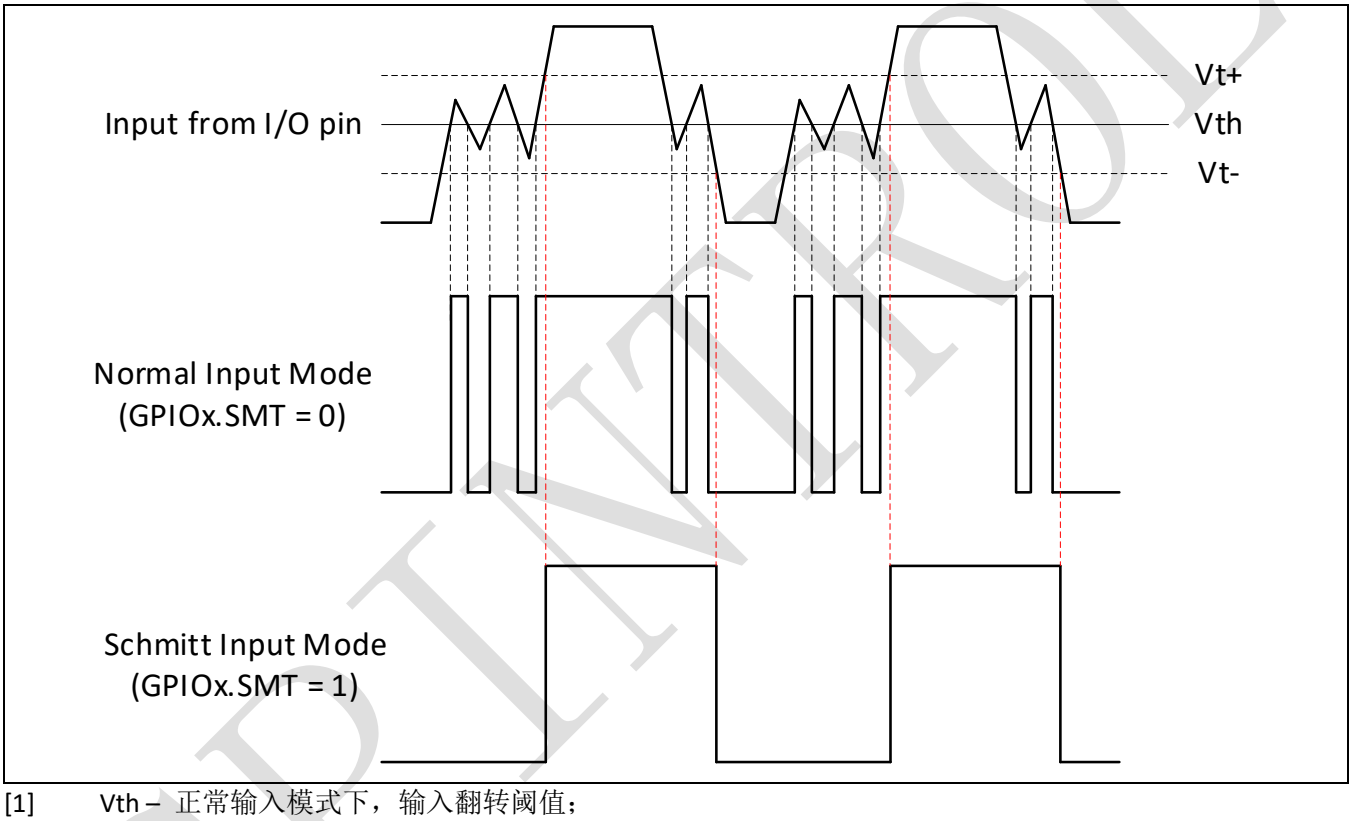
注意： 如果 GPIOx 端口的通道选择 ADC 输入功能，那么 GPIOx.IE 会被置 0，且输出功能会关闭。

当 GPIO20，GPIO21 配置为 XIN 与 XOUT 时，外部晶体和内部时钟电路直连，不存在图 4-2 中的输入输出关系。

4.2.1 施密特控制

施密特输入模式可以通过配置寄存器 GPIOx.SMT 来使能。如图 4-3 所示，施密特输入模式能够防止输入信号上小的波动（噪声）引起的误触发。

图 4-3：施密特输入模式



[1] Vth – 正常输入模式下，输入翻转阈值；

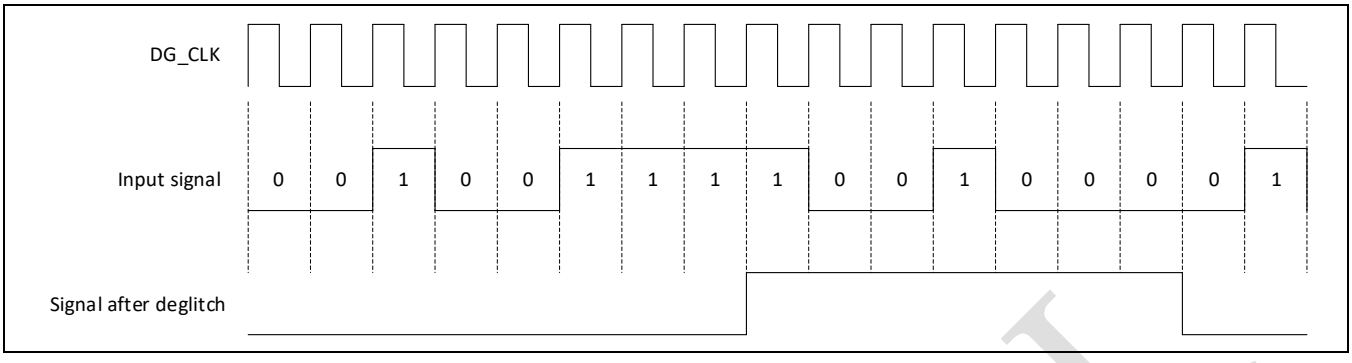
[2] Vt+ – 施密特输入模式下，输入上翻转阈值；Vt- – 施密特输入模式下，输入下翻转阈值。

[3] Vth 的典型值为 1.5V，Vt+的典型值为 1.67V，Vt-的典型值为 1.38V。

4.2.2 消抖控制

输入消抖功能可以通过配置寄存器 GPIOx.DEGLITCH 来使能。输入消抖的原理，如图 4-4 所示。当 Deglitch 模块的输入信号保持为某个值连续 3 个 DG_CLK 时钟周期不变时，Deglitch 模块的输出会变成该值，否则的话，Deglitch 模块的输出保持为原先的值不变。

图 4-4：输入消抖



4.2.3 I/O 可选功能

I/O 引脚各个功能通道对应的功能如表 4-1 所示。

表 4-1：I/O 引脚功能通道定义

引脚	通道（红色表示默认通道）							
	0	1	2	3	4	5	6	7
GPIO0	GPIO0	**ADC0**	COMP0H				SIO0_0	ECAPO
GPIO1	GPIO1	**ADC1**	COMP0L				SIO0_1	ECAPO
GPIO2	GPIO2	**ADC2**	COMP1H				SIO0_2	ECAPO
GPIO3	GPIO3	**ADC3**	COMP1L				SIO0_3	ECAPO
GPIO4	GPIO4	**ADC4**	COMP2H				SIO0_4	ECAPO
GPIO5	GPIO5	**ADC5**	COMP2L				SIO0_5	ECAPO
GPIO6	GPIO6	**ADC6**					SIO0_6	ECAPO
GPIO7	GPIO7	**ADC7**					SIO0_7	ECAPO
GPIO8	GPIO8	**ADC8**					SIO0_8	ECAPO
GPIO9	GPIO9	**ADC9**					SIO0_9	ECAPO
GPIO10	GPIO10	**ADC10**	COMP3H				SIO0_10	ECAPO
GPIO11	GPIO11	**ADC11**	COMP3L	DCLK			SIO0_11	ECAPO
GPIO12	GPIO12	**ADC12**	COMP4H				SIO0_12	ECAPO
GPIO13	GPIO13	**ADC13**	COMP4L				SIO0_13	ECAPO
GPIO14	GPIO14	**ADC14**	COMP5H				SIO0_14	ECAPO
GPIO15	GPIO15	**ADC15**	COMP5L				SIO0_15	ECAPO
GPIO16	GPIO16	**ADC16**	COMP6H				SIO0_16	ECAPO
GPIO17	GPIO17	**ADC17**	COMP6L				SIO0_17	ECAPO
GPIO18	GPIO18	**ADC18**	COMP7H				SIO0_0	ECAPO
GPIO19	GPIO19	**ADC19**	COMP7L				SIO0_1	ECAPO
GPIO20	**GPIO20**	XIN	SPI_SCLK	I2C_SCL	UART_TXD	PWMSYNCO	SIO0_2	ECAPO
GPIO21	**GPIO21**	XIO	SPI_SFRM	I2C_SDA	UART_RXD	PWMSOC	SIO0_3	ECAPO
GPIO22	**GPIO22**		SPI_MOSI	SPI_MISO			SIO0_4	ECAPO
GPIO23	**GPIO23**		SPI_MISO	SPI_MOSI			SIO0_5	ECAPO

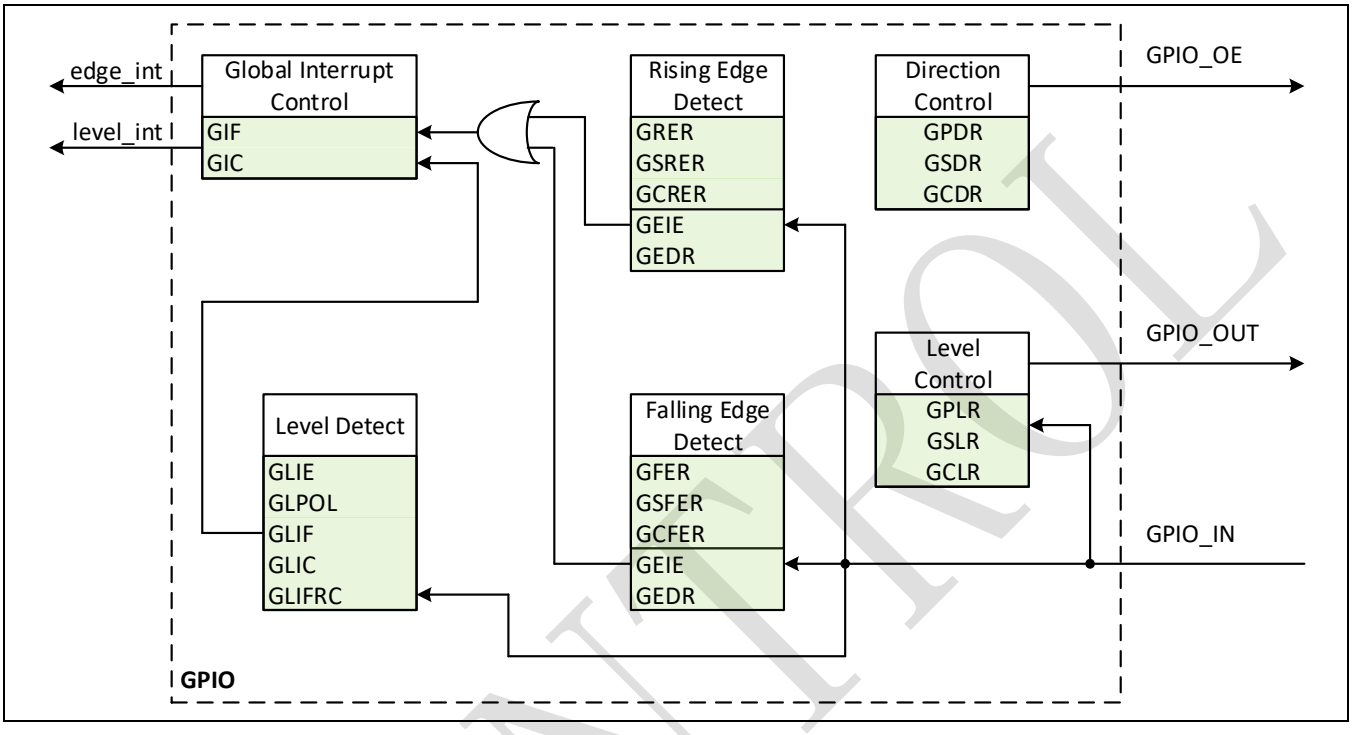
引脚	通道（红色表示默认通道）							
	0	1	2	3	4	5	6	7
GPIO24	GPIO24		COMP0H		PWM1A	PWM1A	SIO0_6	ECAPO
GPIO25	GPIO25		COMP0L		PWM2A	PWM1B	SIO0_7	ECAPO
GPIO26	GPIO26		COMP1H		PWM3A	PWM2A	SIO1_0	ECAPO
GPIO27	GPIO27		COMP1L		PWM1B	PWM2B	SIO1_1	ECAPO
GPIO28	GPIO28		COMP2H		PWM2B	PWM3A	SIO1_2	ECAPO
GPIO29	GPIO29		COMP2L		PWM3B	PWM3B	SIO1_3	ECAPO
GPIO30	GPIO30	I2C_SCL					SIO1_4	ECAPO
GPIO31	GPIO31	I2C_SDA					SIO1_5	ECAPO
GPIO32	GPIO32		COMP3H	PWMSYNCO	PWM4A	PWM4A	SIO1_6	ECAPO
GPIO33	GPIO33		COMP3L	PWMSOC	PWM5A	PWM4B	SIO1_7	ECAPO
GPIO34	GPIO34		COMP4H	PWM0A	PWM6A	PWM5A	SIO1_8	ECAPO
GPIO35	GPIO35		COMP4L	PWM0B	PWM4B	PWM5B	SIO1_9	ECAPO
GPIO36	GPIO36		COMP5H	PWM7A	PWM5B	PWM6A	SIO1_10	ECAPO
GPIO37	GPIO37		COMP5L	PWM7B	PWM6B	PWM6B	SIO1_11	ECAPO
GPIO38	GPIO38	SPI_SCLK	UART_TXD	UART_RXD	PWMSOCA		SIO1_12	ECAPO
GPIO39	GPIO39	SPI_SFRM	UART_RXD	UART_TXD	PWMSOCB		SIO1_13	ECAPO
GPIO40	GPIO40	SPI_MOSI	SPI_MISO	I2C_SCL	PWMSOCC		SIO1_14	ECAPO
GPIO41	GPIO41	SPI_MISO	SPI_MOSI	I2C_SDA	PWMSYNCO		SIO1_15	ECAPO
GPIO42	GPIO42	I2C_SCL		COMP6H	SPI_SCLK		SIO1_16	ECAPO
GPIO43	GPIO43	I2C_SDA		COMP6L	SPI_SFRM		SIO1_17	ECAPO
GPIO44	GPIO44	UART_TXD	UART_RXD	COMP7H	SPI_MOSI	SPI_MISO	SIO2_0	ECAPO
GPIO45	GPIO45	UART_RXD	UART_TXD	COMP7L	SPI_MISO	SPI_MOSI	SIO2_1	ECAPO
GPIO46	GPIO46						SIO2_2	ECAPO
GPIO47	GPIO47						SIO2_3	ECAPO
GPIO48	GPIO48	TCK/SWCK			COMP0H	COMP3H	SIO2_4	ECAPO
GPIO49	GPIO49	TMS/SWD			COMP0L	COMP3L	SIO2_5	ECAPO
GPIO50	GPIO50	TDI	CAU_SWCK		COMP1H	COMP4H	SIO2_6	ECAPO
GPIO51	GPIO51	TDO	CAU_SWD		COMP1L	COMP4L	SIO2_7	ECAPO
GPIO52	GPIO52			PWM0A	COMP2H	COMP5H	SIO2_8	ECAPO
GPIO53	GPIO53			PWM0B	COMP2L	COMP5L	SIO2_9	ECAPO
GPIO54	GPIO54			PWM7A	PWM4A	PWM4A	SIO2_10	ECAPO
GPIO55	GPIO55			PWM7B	PWM5A	PWM4B	SIO2_11	ECAPO
GPIO56	GPIO56				PWM4A	PWM4A	SIO2_12	ECAPO
GPIO57	GPIO57				PWM5A	PWM4B	SIO2_13	ECAPO
GPIO58	GPIO58				PWM6A	PWM5A	SIO2_14	ECAPO
GPIO59	GPIO59				PWM4B	PWM5B	SIO2_15	ECAPO
GPIO60	GPIO60				PWM5B	PWM6A	SIO2_16	ECAPO
GPIO61	GPIO61				PWM6B	PWM6B	SIO2_17	ECAPO

[1] 注意：GPIO47 为 BOOT 引脚，在芯片启动阶段不可被配置为其他功能，否则将导致 GPIO47 出现恒定电平，影响 ISP 模式或 Flash 启动模式的进入。

4.3 GPIO 功能 IP（GPIO）

GPIO 模块的结构框图，如图 4-5 所示。

图 4-5：GPIO 结构框图



GPIO 引脚被被封为两组：GPIO_PORT0 和 GPIO_PORT1。每组分为 32 比特宽度，支持位控制。GPIO_PORT1 中仅使用了低 30 位。

GPIO 引脚默认为输入模式。可以通过 GPIO 方向寄存器（GPDR0 / GPDR1）或者 GPIO 设置输出寄存器（GSDR0 / GSDR1）或者 GPIO 清除输出寄存器（GCDR0 / GCDR1）配置输入或者输出模式。GSDR 和 GCDR 是位控制寄存器，相应位写 0 无效，写 1 使能。

当 GPIO 被配置为输出，可以通过 GPIO 电平寄存器（GPLR0 / GPLR1）或者 GPIO 设置电平寄存器（GSLR0 / GSLR1）或者 GPIO 清除电平寄存器（GCLR0 / GCLR1）配置输出高电平或者低电平。GSLR 和 GCLR 是位控制寄存器，相应位写 0 无效，写 1 使能。

注意： 无论向 GPLR 寄存器或者 GPDR 寄存器写入任何值，读取的 GPLR 值只反映当前引脚的状态（高电平或者低电平）。

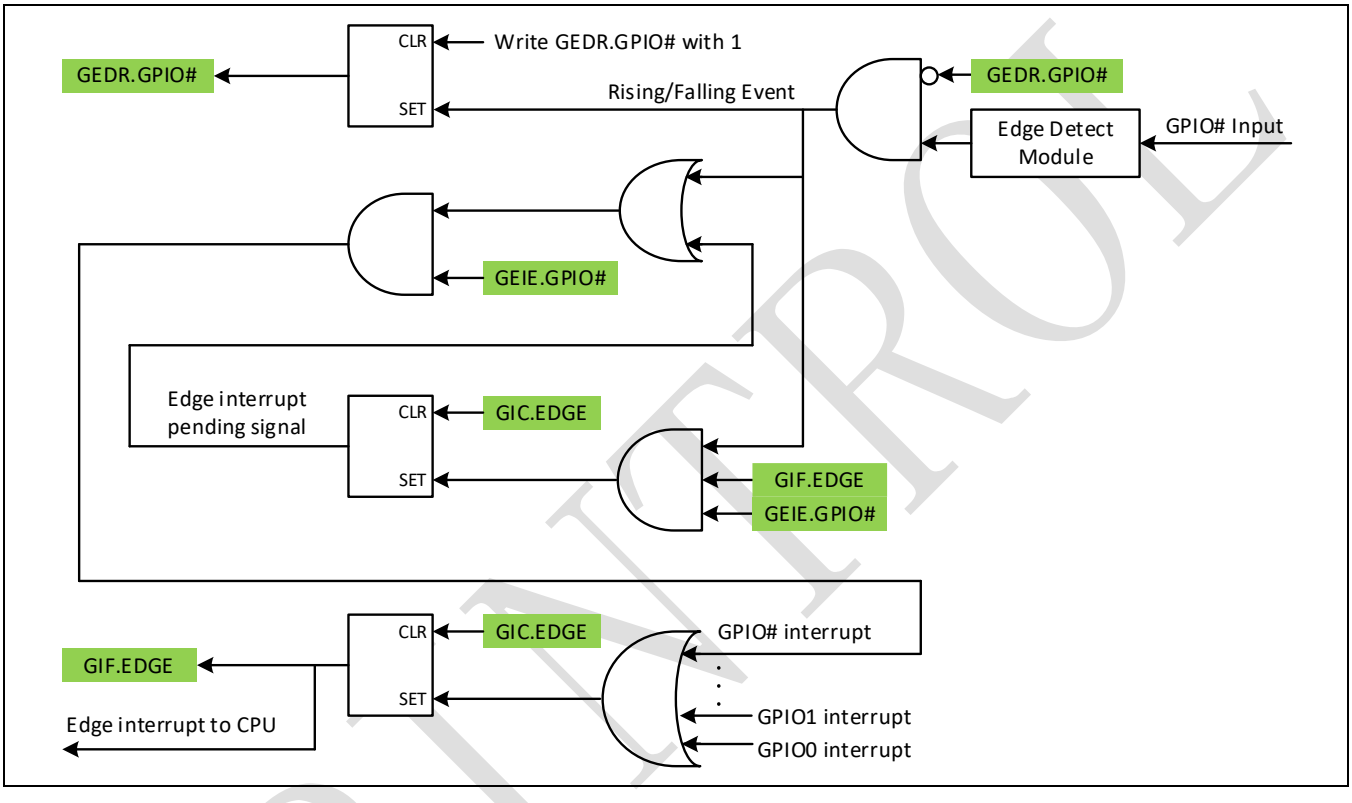
4.3.1 GPIO 边沿触发中断

可以通过配置 GPIO 上升沿检测使能寄存器（GRER0 / GRER1）或者 GPIO 下降沿检测使能寄存器（GFER0 / GFER1）使能边沿检测模块。GSRER，GCRER，GSFER 和 GCFER 是位控制寄存器，相应位写 0 无效，写 1 使能。

GPIO 边沿检测寄存器（GEDR0 / GEDR1）指示边沿检测状态。相应位写 1 清零该位，写 0 无效。边沿检测工作在 APB 时钟域下。

使能 GPIO 边沿检测中断寄存器（GEIE0 / GEIE1）中的相应位，当检测到相应的边沿，则 GEDR 相应位置位，同时 GPIO 全局中断标志寄存器中 GIF.EDGE 置位，触发 CPU 中断。如图 4-6 所示，向全局中断标志清除寄存器（GIC.EDGE）写 1 清除中断标志。

图 4-6：GPIO 边沿触发中断

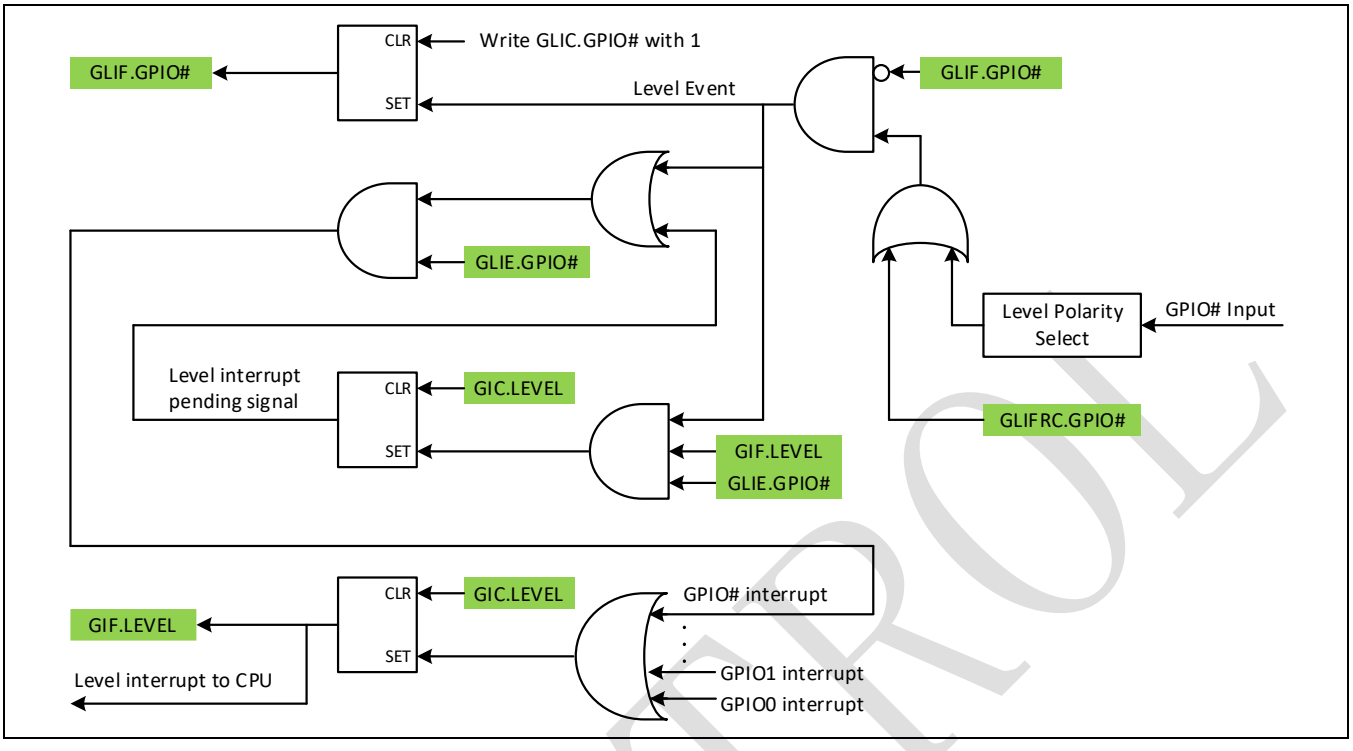


若 GIF.EDGE 为高电平，此时其他 GPIO 检测到边沿，并且 GEDR 相应位是低电平，则会引起中断挂起。清除 GIF.EDGE 则释放挂起的中断，并且程序会由于该挂起的中断再次进入中断服务程序。但是，若 GPIO 检测到边沿，GEDR 中相应位已经为高电平，则会忽略该事件。因此，中断服务程序中应当清除 GEDR 和 GIF.EDGE，确保新的边沿事件可以触发中断。

4.3.2 GPIO 电平触发中断

使能 GPIO 电平检测中断使能寄存器（GLIE0 / GLIE1）中相应位，可以打开电平中断响应。使能 GLIE 前应当先配置 GPIO 电平检测极性寄存器（GLIPOL0 / GLIPOL1）。GPIO 电平检测中断标志寄存器（GLIF0 / GLIF1）和 GPIO 中断标志寄存器中 GIF.LEVEL 指示当前检测状态。可以通过写 GLIFRC0 / GLIFRC1，强制产生电平检测中断事件。如图 4-7 所示，向电平全局中断标志清除寄存器（GLIC0 / GLIC1）相应位写 1 清除全局中断标志寄存器（GLIF0 / GLIF1）中的相应位。GIC.LEVEL 写 1 清除标志 GIF.LEVEL。GPIO 电平检测中断挂起和边沿检测中断挂起的动作相同。

图 4-7：GPIO 电平触发中断



4.4 寄存器

4.4.1 PINMUX 寄存器列表

表 4-2：PINMUX 模块基地址

外设模块	基地址
PINMUX	0x4000 0300

表 4-3：PINMUX 寄存器列表

寄存器	偏移地址	描述	复位值
GPIO0*	0x00	GPIO0 引脚控制寄存器	0x01000019
GPIO1*	0x04	GPIO1 引脚控制寄存器	0x01000019
GPIO2*	0x08	GPIO2 引脚控制寄存器	0x01000019
GPIO3*	0x0C	GPIO3 引脚控制寄存器	0x01000019
GPIO4*	0x10	GPIO4 引脚控制寄存器	0x01000019
GPIO5*	0x14	GPIO5 引脚控制寄存器	0x01000019
GPIO6*	0x18	GPIO6 引脚控制寄存器	0x01000019
GPIO7*	0x1C	GPIO7 引脚控制寄存器	0x01000019
GPIO8*	0x20	GPIO8 引脚控制寄存器	0x01000019
GPIO9*	0x24	GPIO9 引脚控制寄存器	0x01000019
GPIO10*	0x28	GPIO10 引脚控制寄存器	0x01000019
GPIO11*	0x2C	GPIO11 引脚控制寄存器	0x01000019
GPIO12*	0x30	GPIO12 引脚控制寄存器	0x01000019
GPIO13*	0x34	GPIO13 引脚控制寄存器	0x01000019
GPIO14*	0x38	GPIO14 引脚控制寄存器	0x01000019
GPIO15*	0x3C	GPIO15 引脚控制寄存器	0x01000019
GPIO16*	0x40	GPIO16 引脚控制寄存器	0x01000019
GPIO17*	0x44	GPIO17 引脚控制寄存器	0x01000019
GPIO18*	0x48	GPIO18 引脚控制寄存器	0x01000019
GPIO19*	0x4C	GPIO19 引脚控制寄存器	0x01000019
GPIO20*	0x50	GPIO20 引脚控制寄存器	0x09000818
GPIO21*	0x54	GPIO21 引脚控制寄存器	0x19001818
GPIO22*	0x58	GPIO22 引脚控制寄存器	0x01000018
GPIO23*	0x5C	GPIO23 引脚控制寄存器	0x01000018
GPIO24*	0x60	GPIO24 引脚控制寄存器	0x00000018
GPIO25*	0x64	GPIO25 引脚控制寄存器	0x00000018
GPIO26*	0x68	GPIO26 引脚控制寄存器	0x00000018
GPIO27*	0x6C	GPIO27 引脚控制寄存器	0x00000018
GPIO28*	0x70	GPIO28 引脚控制寄存器	0x00000018
GPIO29*	0x74	GPIO29 引脚控制寄存器	0x00000018
GPIO30*	0x78	GPIO30 引脚控制寄存器	0x03000218

寄存器	偏移地址	描述	复位值
GPIO31*	0x7C	GPIO31 引脚控制寄存器	0x03000218
GPIO32*	0x80	GPIO32 引脚控制寄存器	0x00000018
GPIO33*	0x84	GPIO33 引脚控制寄存器	0x00000018
GPIO34*	0x88	GPIO34 引脚控制寄存器	0x00000018
GPIO35*	0x8C	GPIO35 引脚控制寄存器	0x00000018
GPIO36*	0x90	GPIO36 引脚控制寄存器	0x00000018
GPIO37*	0x94	GPIO37 引脚控制寄存器	0x00000018
GPIO38*	0x98	GPIO38 引脚控制寄存器	0x09000818
GPIO39*	0x9C	GPIO39 引脚控制寄存器	0x05000418
GPIO40*	0xA0	GPIO40 引脚控制寄存器	0x09000818
GPIO41*	0xA4	GPIO41 引脚控制寄存器	0x09000818
GPIO42*	0xA8	GPIO42 引脚控制寄存器	0x03000218
GPIO43*	0xAC	GPIO43 引脚控制寄存器	0x03000218
GPIO44*	0xB0	GPIO44 引脚控制寄存器	0x05000418
GPIO45*	0xB4	GPIO45 引脚控制寄存器	0x03000218
GPIO46*	0xB8	GPIO46 引脚控制寄存器	0x01000018
GPIO47*	0xBC	GPIO47 引脚控制寄存器	0x01000018
GPIO48*	0xC0	GPIO48 引脚控制寄存器	0x03000018
GPIO49*	0xC4	GPIO49 引脚控制寄存器	0x03000018
GPIO50*	0xC8	GPIO50 引脚控制寄存器	0x07000018
GPIO51*	0xCC	GPIO51 引脚控制寄存器	0x05000018
GPIO52*	0xD0	GPIO52 引脚控制寄存器	0x00000018
GPIO53*	0xD4	GPIO53 引脚控制寄存器	0x00000018
GPIO54*	0xD8	GPIO54 引脚控制寄存器	0x00000018
GPIO55*	0xDC	GPIO55 引脚控制寄存器	0x00000018
GPIO56*	0xE0	GPIO56 引脚控制寄存器	0x00000018
GPIO57*	0xE4	GPIO57 引脚控制寄存器	0x00000018
GPIO58*	0xE8	GPIO58 引脚控制寄存器	0x00000018
GPIO59*	0xEC	GPIO59 引脚控制寄存器	0x00000018
GPIO60*	0xF0	GPIO60 引脚控制寄存器	0x00000018
GPIO61*	0xF4	GPIO61 引脚控制寄存器	0x00000018
PINMUXREGKEY	0xF8	PINMUX 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 PINMUXREGKEY =0x1ACCE551 才可以改写。

4.4.2 PINMUX 寄存器

表 4-4 到表 4-7 列出了 GPIO PINMUX 相关寄存器的定义。

表 4-4：GPIO#引脚控制寄存器（GPIO#）位段定义（# = 0 ~61）

GPIO# (GPIO# Pin Control Register) Offset: # * 4 Access: PINMUX -> GPIO#.all							
31	30	29	28	27	26	25	24
PU							
23	22	21	20	19	18	17	16
PD							
15	14	13	12	11	10	9	8
CHDFLTVAL							
7	6	5	4	3	2	1	0
STRENGTH		DEGLITCH	SMT	IE	MUXSEL		

表 4-5：GPIO#引脚控制寄存器（GPIO#）位段描述（# = 0 ~61）

位段	位段名	属性	复位值	描述
31:24	PU	RW	-	使能上拉，每一位代表一个通道 如果 PU[n]=1，则使能通道 n 上拉
23:16	PD	RW	-	使能下拉，每一位代表一个通道 如果 PD[n]=1，则使能通道 n 下拉
15:8	CHDFLTVAL	RW	-	解复用器输入通道默认值，每一位代表一个通道 如果 MUXSEL != n (n = 0 ~ 7)，那么通道 n 输入= CHDFLTVAL[n]
7:6	STRENGTH	RW	-	输出驱动强度 00：5mA 01：10mA 10：15mA 11：20mA
5	DEGLITCH	RW	-	输入尖峰脉冲过滤使能 0：禁用 1：使能
4	SMT	RW	-	施密特输入过滤使能 0：正常输入模式； 1：施密特输入模式。
3	IE	RW	-	输入使能 0：禁用输入，读取 GPIO 总是 0 1：使能输入
2:0	MUXSEL	RW	-	通道选择 每个 GPIO 引脚的通道定义如表 4-1 所示。

表 4-6：PINMUX 模块写使能寄存器（PINMUXREGKEY）位段定义

PINMUXREGKEY (PINMUX Register Write-Allow Key Register) Offset: 0xF8 Default: 0x1ACCE551 Access: PINMUX -> PINMUXREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 4-7：PINMUX 模块写使能寄存器（PINMUXREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 以解锁受保护的 PINMUX 寄存器

4.4.3 GPIO 寄存器列表

表 4-8：GPIO 模块基地址

外设模块	基地址
GPIO	0x4000 3000

表 4-9：GPIO 寄存器列表

寄存器	偏移地址	描述	复位值
GPLR0*	0x00	GPIO 引脚电平寄存器 0	0x00030000
GPLR1*	0x04	GPIO 引脚电平寄存器 1	0x000001CC
GPDR0*	0x0C	GPIO 引脚方向寄存器 0	0x00000000
GPDR1*	0x10	GPIO 引脚方向寄存器 1	0x00000000
GSLR0*	0x18	GPIO 引脚输出置 1 寄存器 0	0x00000000
GSLR1*	0x1C	GPIO 引脚输出置 1 寄存器 1	0x00000000
GCLR0*	0x24	GPIO 引脚输出清除寄存器 0	0x00000000
GCLR1*	0x28	GPIO 引脚输出清除寄存器 1	0x00000000
GRER0*	0x30	GPIO 上升沿检测使能寄存器 0	0x00000000
GRER1*	0x34	GPIO 上升沿检测使能寄存器 1	0x00000000
GFER0*	0x3C	GPIO 下降沿检测使能寄存器 0	0x00000000
GFER1*	0x40	GPIO 下降沿检测使能寄存器 1	0x00000000
GEDR0	0x48	GPIO 边沿检测状态寄存器 0	0x00000000
GEDR1	0x4C	GPIO 边沿检测状态寄存器 1	0x00000000
GSDR0*	0x54	GPIO 引脚按位设置输出寄存器 0	0x00000000
GSDR1*	0x58	GPIO 引脚按位设置输出寄存器 1	0x00000000
GCDR0*	0x60	GPIO 引脚按位设置输入寄存器 0	0x00000000
GCDR1*	0x64	GPIO 引脚按位设置输入寄存器 1	0x00000000
GSRER0*	0x6C	GPIO 上升沿检测按位使能寄存器 0	0x00000000
GSRER1*	0x70	GPIO 上升沿检测按位使能寄存器 1	0x00000000
GCRER0*	0x78	GPIO 上升沿检测按位关闭寄存器 0	0x00000000
GCRER1*	0x7C	GPIO 上升沿检测按位关闭寄存器 1	0x00000000
GSFER0*	0x84	GPIO 下降沿检测按位使能寄存器 0	0x00000000
GSFER1*	0x88	GPIO 下降沿检测按位使能寄存器 1	0x00000000
GCFER0*	0x90	GPIO 下降沿检测按位关闭寄存器 0	0x00000000
GCFER1*	0x94	GPIO 下降沿检测按位关闭寄存器 1	0x00000000
GEIE0*	0x9C	GPIO 边沿中断使能寄存器 0	0x00000000
GEIE1*	0xA0	GPIO 边沿中断使能寄存器 1	0x00000000
GLIF0*	0xA8	GPIO 电平中断标志寄存器 0	0x00000000
GLIF1*	0xAC	GPIO 电平中断标志寄存器 1	0x00000000
GLIE0*	0xB4	GPIO 电平中断使能寄存器 0	0x00000000
GLIE1*	0xB8	GPIO 电平中断使能寄存器 1	0x00000000
GLIC0	0xC0	GPIO 电平中断清除寄存器 0	0x00000000

寄存器	偏移地址	描述	复位值
GLIC1	0xC4	GPIO 电平中断清除寄存器 1	0x00000000
GLIFRC0*	0xCC	GPIO 电平中断强制寄存器 0	0x00000000
GLIFRC1*	0xD0	GPIO 电平中断强制寄存器 1	0x00000000
GLIPOL0*	0xD8	GPIO 电平中断极性寄存器 0	0x00000000
GLIPOL1*	0xDC	GPIO 电平中断极性寄存器 1	0x00000000
GIF	0xE4	GPIO 全局中断标志寄存器	0x00000000
GIC	0xE8	GPIO 全局中断清除寄存器	0x00000000
GPIOREGKEY	0xEC	GPIO 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 GPIOREGKEY=0x1ACCE551 才可以改写。

4.4.4 GPIO 寄存器

表 4-10 到表 4-91 列出了 GPIO 功能 IP 相关寄存器的定义。

表 4-10：GPIO 引脚电平寄存器 0（GPLR0）位段定义

GPLR0 (GPIO Pin Level Register 0) Offset: 0x0 Default: 0x00030000 Access: GPIO -> GPLR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-11：GPIO 引脚电平寄存器 0（GPLR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x30000	GPIOn 引脚电平（n = 0~31）。 0：写 0 设置引脚低电平；读 0 表示当前引脚为低电平 1：写 1 设置引脚高电平；读 1 表示当前引脚为高电平

表 4-12：GPIO 引脚电平寄存器 1（GPLR1）位段定义

GPLR1 (GPIO Pin Level Register 1) Offset: 0x4 Default: 0x000FFFC0 Access: GPIO -> GPLR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-13：GPIO 引脚电平寄存器 1（GPLR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	RW	0x1CC	GPIOn 引脚电平（n = 32~61）。 0：写 0 设置引脚低电平；读 0 表示当前引脚为低电平 1：写 1 设置引脚高电平；读 1 表示当前引脚为高电平

表 4-14：GPIO 引脚方向寄存器 0（GPDR0）位段定义

GPDR0 (GPIO Pin Direction Register 0) Offset: 0xC Default: 0x00000000 Access: GPIO -> GPDR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-15：GPIO 引脚方向寄存器 0（GPDR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	设置 GPIOn 方向（n = 0~31） 0：输入模式 1：输出模式

表 4-16：GPIO 引脚方向寄存器 1（GPDR1）位段定义

GPDR1 (GPIO Pin Direction Register 1) Offset: 0x10 Default: 0x00000000 Access: GPIO -> GPDR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-17：GPIO 引脚方向寄存器 1（GPDR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	RW	0x0	设置 GPIOn 方向（n = 32~61） 0：输入模式 1：输出模式

表 4-18：GPIO 引脚输出置 1 寄存器 0（GSLR0）位段定义

GSLR0 (GPIO Pin Output Set Register 0) Offset: 0x18 Default: 0x00000000 Access: GPIO -> GSLR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-19：GPIO 引脚输出置 1 寄存器 0（GSLR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1S	0x0	设置 GPIOn 输出高电平（n = 0~31） 0：写 0 无效，读总是 0 1：写 1，GPIOn 输出高，硬件自动清零该位。

表 4-20：GPIO 引脚输出置 1 寄存器 1（GSLR1）位段定义

GSLR1 (GPIO Pin Output Set Register 1) Offset: 0x1C Default: 0x00000000 Access: GPIO -> GSLR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-21：GPIO 引脚输出置 1 寄存器 1（GSLR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1S	0x0	设置 GPIOn 输出高电平（n = 32~61） 0：写 0 无效，读总是 0 1：写 1，GPIOn 输出高，硬件自动清零该位。

表 4-22：GPIO 引脚输出清除寄存器 0（GCLR0）位段定义

GCLR0 (GPIO Pin Output Clear Register 0) Offset: 0x24 Default: 0x00000000 Access: GPIO -> GCLR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-23：GPIO 引脚输出清除寄存器 0（GCLR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	设置 GPIOn 输出低电平（n = 0~31） 0：写 0 无效，读总是 0 1：写 1，GPIOn 输出低，硬件自动清零。

表 4-24：GPIO 引脚输出清除寄存器 1（GCLR1）位段定义

GCLR1 (GPIO Pin Output Clear Register 1) Offset: 0x28 Default: 0x00000000 Access: GPIO -> GCLR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-25：GPIO 引脚输出清除寄存器 1（GCLR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	设置 GPIOn 输出低电平（n = 32~61） 0：写 0 无效，读总是 0 1：写 1，GPIOn 输出低，硬件自动清零。

表 4-26：GPIO 上升沿检测使能寄存器 0（GRER0）位段定义

GRER0 (GPIO Rising Edge Detect Enable Register 0) Offset: 0x30 Default: 0x00000000 Access: GPIO -> GRER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-27：GPIO 上升沿检测使能寄存器 0（GRER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	使能 GPIOn 上升沿检测（n = 0~31） 0：关闭 GPIOn 上升沿检测 1：使能 GPIOn 上升沿检测

表 4-28：GPIO 上升沿检测使能寄存器 1（GRER1）位段定义

GRER1 (GPIO Rising Edge Detect Enable Register 1) Offset: 0x34 Default: 0x00000000 Access: GPIO -> GRER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-29：GPIO 上升沿检测使能寄存器 1（GRER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	RW	0x0	使能 GPIOn 上升沿检测（n = 32~61） 0：关闭 GPIOn 上升沿检测 1：使能 GPIOn 上升沿检测

表 4-30：GPIO 下降沿检测使能寄存器 0（GFER0）位段定义

GFER0 (GPIO Falling Edge Detect Enable Register 0) Offset: 0x3C Default: 0x00000000 Access: GPIO -> GFER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-31：GPIO 下降沿检测使能寄存器 0（GFER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	使能 GPIOn 下降沿检测（n = 0~31） 0：关闭 GPIOn 下降沿检测 1：使能 GPIOn 下降沿检测

表 4-32：GPIO 下降沿检测使能寄存器 1（GFER1）位段定义

GFER1 (GPIO Falling Edge Detect Enable Register 1) Offset: 0x40 Default: 0x00000000 Access: GPIO -> GFER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-33：GPIO 下降沿检测使能寄存器 1（GFER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	RW	0x0	使能 GPIOn 下降沿检测（n = 32~61） 0：关闭 GPIOn 下降沿检测 1：使能 GPIOn 下降沿检测

表 4-34：GPIO 边沿检测状态寄存器 0（GEDR0）位段定义

GEDR0 (GPIO Edge Detect Status Register 0) Offset: 0x48 Default: 0x00000000 Access: GPIO -> GEDR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-35：GPIO 边沿检测状态寄存器 0（GEDR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	GPIOn 边沿检测状态（n = 0~31） 0：读 0，未检测到边沿；写 0 无效 1：读 1，检测到边沿；写 1 清除检测状态。

表 4-36：GPIO 边沿检测状态寄存器 1（GEDR1）位段定义

GEDR1 (GPIO Edge Detect Status Register 1) Offset: 0x4C Default: 0x00000000 Access: GPIO -> GEDR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-37：GPIO 边沿检测状态寄存器 1（GEDR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	GPIOn 边沿检测状态（n = 32~61） 0：读 0，未检测到边沿；写 0 无效 1：读 1，检测到边沿；写 1 清除检测状态。

表 4-38：GPIO 引脚按位设置输出寄存器 0（GSDR0）位段定义

GSDR0 (GPIO Pin Bitwise Set Direction Register 0) Offset: 0x54 Default: 0x00000000 Access: GPIO -> GSDR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-39：GPIO 引脚按位设置输出寄存器 0（GSDR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1S	0x0	置位 GPDR0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 置位 GPDR0 寄存器的相应位段并设置 GPIOn 为输出，本位段自动清零。

表 4-40：GPIO 引脚按位设置输出寄存器 1（GSDR1）位段定义

GSDR1 (GPIO Pin Bitwise Set Direction Register 1) Offset: 0x58 Default: 0x00000000 Access: GPIO -> GSDR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-41：GPIO 引脚按位设置输出寄存器 1（GSDR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1S	0x0	置位 GPDR1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 置位 GPDR1 寄存器的相应位段并设置 GPIOn 为输出，本位段自动清零。

表 4-42：GPIO 引脚按位设置输入寄存器 0（GCDR0）位段定义

GCDR0 (GPIO Pin Bitwise Clear Direction Register 0) Offset: 0x60 Default: 0x00000000 Access: GPIO -> GCDR0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-43：GPIO 引脚按位设置输入寄存器 0（GCDR0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	清零 GPDR0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 清零 GPDR0 寄存器的相应位段并设置 GPIOn 为输入，本位段自动清零。

表 4-44：GPIO 引脚按位设置输入寄存器 1（GCDR1）位段定义

GCDR1 (GPIO Pin Bitwise Clear Direction Register 1) Offset: 0x64 Default: 0x00000000 Access: GPIO -> GCDR1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-45：GPIO 引脚按位设置输入寄存器 1（GCDR1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	清零 GPDR1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 清零 GPDR1 寄存器的相应位段并设置 GPIOn 为输入，本位段自动清零。

表 4-46：GPIO 上升沿检测按位使能寄存器 0（GSRER0）位段定义

GSRER0 (GPIO Bitwise Set Rising Edge Detect Enable Register 0) Offset: 0x6C Default: 0x00000000 Access: GPIO -> GSRER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-47：GPIO 上升沿检测按位使能寄存器 0（GSRER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1S	0x0	置位 GRER0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 置位 GRER0 的相应位段并使能 GPIOn 的上升沿检测，本位段自动清零。

表 4-48：GPIO 上升沿检测按位使能寄存器 1（GSRER1）位段定义

GSRER1 (GPIO Bitwise Set Rising Edge Detect Enable Register 1) Offset: 0x70 Default: 0x00000000 Access: GPIO -> GSRER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-49：GPIO 上升沿检测按位使能寄存器 1（GSRER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1S	0x0	置位 GRER1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 置位 GRER1 的相应位段并使能 GPIOn 的上升沿检测，本位段自动清零。

表 4-50：GPIO 上升沿检测按位关闭寄存器 0（GCRER0）位段定义

GCRER0 (GPIO Bitwise Clear Rising Edge Detect Enable Register 0) Offset: 0x78 Default: 0x00000000 Access: GPIO -> GCRER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-51：GPIO 上升沿检测按位关闭寄存器 0（GCRER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	清零 GRER0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 清零 GRER0 的相应位段并关闭 GPIOn 的上升沿检测，本位段自动清零。

表 4-52：GPIO 上升沿检测按位关闭寄存器 1（GCRER1）位段定义

GCRER1 (GPIO Bitwise Clear Rising Edge Detect Enable Register 1) Offset: 0x7C Default: 0x00000000 Access: GPIO -> GCRER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-53：GPIO 上升沿检测按位关闭寄存器 1（GCRER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	清零 GRER1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 清零 GRER1 的相应位段并关闭 GPIOn 的上升沿检测，本位段自动清零。

表 4-54：GPIO 下降沿检测按位使能寄存器 0（GSFER0）位段定义

GSFER0 (GPIO Bitwise Set Falling Edge Detect Enable Register 0) Offset: 0x84 Default: 0x00000000 Access: GPIO -> GSFER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-55：GPIO 下降沿检测按位使能寄存器 0（GSFER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1S	0x0	置位 GFER0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 置位 GFER0 的相应位段并使能 GPIOn 的下降沿检测，本位段自动清零。

表 4-56：GPIO 下降沿检测按位使能寄存器 1（GSFER1）位段定义

GSFER1 (GPIO Bitwise Set Falling Edge Detect Enable Register 1) Offset: 0x88 Default: 0x00000000 Access: GPIO -> GSFER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-57：GPIO 下降沿检测按位使能寄存器 1（GSFER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1S	0x0	置位 GFER1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 置位 GFER1 的相应位段并使能 GPIOn 的下降沿检测，本位段自动清零。

表 4-58：GPIO 下降沿检测按位关闭寄存器 0（GCFER0）位段定义

GCFER0 (GPIO Bitwise Clear Falling Edge Detect Enable Register 0) Offset: 0x90 Default: 0x00000000 Access: GPIO -> GCFER0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-59：GPIO 下降沿检测按位关闭寄存器 0（GCFER0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	清零 GFER0 寄存器位（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 清零 GFER0 的相应位段并关闭 GPIOn 的下降沿检测，本位段自动清零。

表 4-60：GPIO 下降沿检测按位关闭寄存器 1（GCFER1）位段定义

GCFER1 (GPIO Bitwise Clear Falling Edge Detect Enable Register 1) Offset: 0x94 Default: 0x00000000 Access: GPIO -> GCFER1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-61：GPIO 下降沿检测按位关闭寄存器 1（GCFER1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	清零 GFER1 寄存器位（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 清零 GFER1 的相应位段并关闭 GPIOn 的下降沿检测，本位段自动清零。

表 4-62：GPIO 边沿中断使能寄存器 0（GEIE0）位段定义

GEIE0 (GPIO Edge-Triggered Interrupt Enable Register 0) Offset: 0x9C Default: 0x00000000 Access: GPIO -> GEIE0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-63：GPIO 边沿中断使能寄存器 0（GEIE0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	GPIOn 边沿中断使能（n = 0~31） 0：关闭 GPIOn 边沿中断 1：使能 GPIOn 边沿中断

表 4-64：GPIO 边沿中断使能寄存器 1（GEIE1）位段定义

GEIE1 (GPIO Edge-Triggered Interrupt Enable Register 1) Offset: 0xA0 Default: 0x00000000 Access: GPIO -> GEIE1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-65：GPIO 边沿中断使能寄存器 1（GEIE1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_9	RO	0x0	保留
29:0	VAL	RW	0x0	GPIOn 边沿中断使能（n = 32~61） 0：关闭 GPIOn 边沿中断 1：使能 GPIOn 边沿中断

表 4-66：GPIO 电平中断标志寄存器 0（GLIF0）位段定义

GLIF0 (GPIO Level-Triggered Interrupt Flag Register 0) Offset: 0xA8 Default: 0x00000000 Access: GPIO -> GLIF0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-67：GPIO 电平中断标志寄存器 0（GLIF0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	GPIOn 电平中断标志（n = 0~31） 0：GPIOn 电平中断未发生 1：GPIOn 电平中断已发生

表 4-68：GPIO 电平中断标志寄存器 1（GLIF1）位段定义

GLIF1 (GPIO Level-Triggered Interrupt Flag Register 1) Offset: 0xAC Default: 0x00000000 Access: GPIO -> GLIF1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-69：GPIO 电平中断标志寄存器 1（GLIF1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_9	RO	0x0	保留
29:0	VAL	RO	0x0	GPIOn 电平中断标志（n = 32~61） 0：GPIOn 电平中断未发生 1：GPIOn 电平中断已发生

表 4-70：GPIO 电平中断使能寄存器 0（GLIE0）位段定义

GLIE0 (GPIO Level-Triggered Interrupt Enable Register 0) Offset: 0xB4 Default: 0x00000000 Access: GPIO -> GLIE0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-71：GPIO 电平中断使能寄存器 0（GLIE0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	GPIOn 电平中断使能（n = 0~31） 0：关闭 GPIOn 电平中断 1：使能 GPIOn 电平中断

表 4-72：GPIO 电平中断使能寄存器 1（GLIE1）位段定义

GLIE1 (GPIO Level-Triggered Interrupt Enable Register 1) Offset: 0xB8 Default: 0x00000000 Access: GPIO -> GLIE1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-73：GPIO 电平中断使能寄存器 1（GLIE1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_9	RO	0x0	保留
29:0	VAL	RW	0x0	GPIOn 电平中断使能（n = 32~61） 0：关闭 GPIOn 电平中断 1：使能 GPIOn 电平中断

表 4-74：GPIO 电平中断清除寄存器 0（GLIC0）位段定义

GLIC0 (GPIO Level-Triggered Interrupt Clear Register 0) Offset: 0xC0 Default: 0x00000000 Access: GPIO -> GLIC0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-75：GPIO 电平中断清除寄存器 0（GLIC0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1C	0x0	GPIOn 电平中断标志清除（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 清除相应标志位，硬件自动清零该位。

表 4-76：GPIO 电平中断清除寄存器 1（GLIC1）位段定义

GLIC1 (GPIO Level-Triggered Interrupt Clear Register 1) Offset: 0xC4 Default: 0x00000000 Access: GPIO -> GLIC1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-77：GPIO 电平中断清除寄存器 1（GLIC1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	W1C	0x0	GPIOn 电平中断标志清除（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 清除相应标志位，硬件自动清零该位。

表 4-78：GPIO 电平中断强制寄存器 0（GLIFRC0）位段定义

GLIFRC0 (GPIO Level-Triggered Interrupt Force Register 0) Offset: 0xCC Default: 0x00000000 Access: GPIO -> GLIFRC0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-79：GPIO 电平中断强制寄存器 0（GLIFRC0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	W1S	0x0	软件强制 GPIOn 电平中断（n = 0~31） 0：写 0 无效，总是读回 0 1：写 1 强制电平中断，硬件自动清零该位。

表 4-80：GPIO 电平中断强制寄存器 1（GLIFRC1）位段定义

GLIFRC1 (GPIO Level-Triggered Interrupt Force Register 1) Offset: 0xD0 Default: 0x00000000 Access: GPIO -> GLIFRC1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-81：GPIO 电平中断强制寄存器 1（GLIFRC1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_9	RO	0x0	保留
29:0	VAL	W1S	0x0	软件强制 GPIOn 电平中断（n = 32~61） 0：写 0 无效，总是读回 0 1：写 1 强制电平中断，硬件自动清零该位。

表 4-82：GPIO 电平中断极性寄存器 0（GLIPOL0）位段定义

GLIPOL0 (GPIO Level-Triggered Interrupt Polarity Register 0) Offset: 0xD8 Default: 0x00000000 Access: GPIO -> GLIPOL0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-83：GPIO 电平中断极性寄存器 0（GLIPOL0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	GPIOn 电平中断极性（n = 0~31） 0：GPIOn 低电平触发中断 1：GPIOn 高电平触发中断

表 4-84：GPIO 电平中断极性寄存器 1（GLIPOL1）位段定义

GLIPOL1 (GPIO Level-Triggered Interrupt Polarity Register 1) Offset: 0xDC Default: 0x00000000 Access: GPIO -> GLIPOL1.all							
31	30	29	28	27	26	25	24
RESERVED		VAL					
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 4-85：GPIO 电平中断极性寄存器 1（GLIPOL1）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:0	VAL	RW	0x0	GPIOn 电平中断极性（n = 32~61） 0：GPIOn 低电平触发中断 1：GPIOn 高电平触发中断

表 4-86：GPIO 全局中断标志寄存器（GIF）位段定义

GIF (GPIO Interrupt Flag Register) Offset: 0xE4 Default: 0x00000000 Access: GPIO -> GIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						LEVEL	EDGE

表 4-87：GPIO 全局中断标志寄存器（GIF）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	LEVEL	RO	0x0	GPIO 电平中断标志 0：电平中断未发生 1：电平中断已发生
0	EDGE	RO	0x0	GPIO 边沿中断标志 0：边沿中断未发生 1：边沿中断以发生

表 4-88：GPIO 全局中断清除寄存器（GIC）位段定义

GIC (GPIO Interrupt Flag Clear Register) Offset: 0xE8 Default: 0x00000000 Access: GPIO -> GIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						LEVEL	EDGE

表 4-89：GPIO 全局中断清除寄存器（GIC）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	LEVEL	W1C	0x0	GPIO 电平中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除相应标志位，硬件自动清零该位。
0	EDGE	W1C	0x0	GPIO 边沿中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除相应标志位，硬件自动清零该位。

表 4-90：GPIO 模块写使能寄存器（GPIOREGKEY）位段定义

GPIOREGKEY (GPIO Register Write-Allow Key Register) Offset: 0xEC Default: 0x1ACCE551 Access: GPIO -> GPIOREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 4-91：GPIO 模块写使能寄存器（GPIOREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 以解锁受保护的 GPIO 寄存器

5 中断（Interrupts）

5.1 嵌套向量中断控制器（NVIC）

嵌套向量中断控制器（NVIC）是 ARM Cortex-M4 内核的组成部分。NVIC 和 CPU 内核接口紧密配合，可以实现低延迟的中断处理和晚到中断的高效处理。

5.1.1 NVIC 特性

NVIC 包括下列特性：

- 具有多达 63 个中断向量（不包括 Cortex-M4 的 16 个中断）
- 16 个可编程优先级（使用了 4 位中断优先级）。优先级的数值越大，中断的优先级越低，因此优先级 0 是最高的中断优先级。
- 实现系统异常和外设中断的控制
- 支持中断末尾连锁（tail-chaining）和晚到处理
- 支持中断 lazy-stacking
- 支持不可屏蔽中断（NMI）

5.1.2 中断和异常向量

当一个异常事件发生并被 Cortex-M4 处理时，处理器需要知道异常处理函数的确切起始地址。因此，中断向量表的机制就被采用了。中断向量表是位于系统储存器的一个由 32 位数据组成的数组，每一个 32 位数据代表一个异常处理函数的起始地址。中断向量表的位置由 NVIC 中一个叫做 VTOR（地址 0xE000ED08）的可编程寄存器控制。在复位后，VTOR 的值被复位为 0；因此，在复位后，中断向量表的位置在地址 0。中断向量的地址是根据异常向量号乘以 4 来排列的。SPC2168 的中断向量表如表 5-1 所示。

表 5-1：SPC2168 的中断向量表

异常号	CMSIS 中断号	缩略语	优先级	描述	偏移地址
-	-	MSP	-	初始化主堆栈指针	0x0000_0000
1	-	Reset	-3，最高级	复位	0x0000_0004
2	-14	NMI	-2	不可屏蔽中断。 看门狗定时器 0 连接到 NMI 向量。	0x0000_0008
3	-13	Hard Fault	-1	所有类型的故障	0x0000_000C
4	-12	MemManage	可配置	存储器管理故障	0x0000_00010
5	-11	Bus Fault	可配置	总线系统故障	0x0000_0014
6	-10	Usage Fault	可配置	用法故障	0x0000_0018
7-10	-	-	-	保留	0x0000_001C –

异常号	CMSIS 中断号	缩略语	优先级	描述	偏移地址
					0x0000_002B
11	-5	SVCall	可配置	通过 SVC 指令调用的系统服务	0x0000_002C
12	-4	Debug Monitor	可配置	调试监控器	0x0000_0030
13	-	-	-	保留	0x0000_0034
14	-2	PendSV	可配置	可挂起的系统服务	0x0000_0038
15	-1	SysTick	可配置	系统节拍定时器中断	0x0000_003C
16	0	MEM	可配置	存储器中断	0x0000_0040
17	1	POWER	可配置	电源中断	0x0000_0044
18	2	CLOCK	可配置	时钟中断	0x0000_0048
19	3	WDT1	可配置	看门狗定时器 1 中断	0x0000_004C
20	4	GPIO_LEVEL	可配置	GPIO 电平触发中断	0x0000_0050
21	5	GPIO_EDGE	可配置	GPIO 边沿触发中断	0x0000_0054
22	6	SIO0A	可配置	SIO0 中断 A	0x0000_0058
23	7	SIO0B	可配置	SIO0 中断 B	0x0000_005C
24	8	SIO1A	可配置	SIO1 中断 A	0x0000_0060
25	9	SIO1B	可配置	SIO1 中断 B	0x0000_0064
26	10	SIO2A	可配置	SIO2 中断 A	0x0000_0068
27	11	SIO2B	可配置	SIO2 中断 B	0x0000_006C
28	12	UART	可配置	UART 中断	0x0000_0070
29	13	SSP	可配置	SSP 中断	0x0000_0074
30	14	I2C	可配置	I2C 中断	0x0000_0078
31	15	ADC0	可配置	ADC SOC0 中断	0x0000_007C
32	16	ADC1	可配置	ADC SOC1 中断	0x0000_0080
33	17	ADC2	可配置	ADC SOC2 中断	0x0000_0084
34	18	ADC3	可配置	ADC SOC3 中断	0x0000_0088
35	19	ADC4	可配置	ADC SOC4 中断	0x0000_008C
36	20	ADC5	可配置	ADC SOC5 中断	0x0000_0090
37	21	ADC6	可配置	ADC SOC6 中断	0x0000_0094
38	22	ADC7	可配置	ADC SOC7 中断	0x0000_0098
39	23	ADC8	可配置	ADC SOC8 中断	0x0000_009C
40	24	ADC9	可配置	ADC SOC9 中断	0x0000_00A0
41	25	ADC10	可配置	ADC SOC10 中断	0x0000_00A4
42	26	ADC11	可配置	ADC SOC11 中断	0x0000_00A8
43	27	ADC12	可配置	ADC SOC12 中断	0x0000_00AC
44	28	ADC13	可配置	ADC SOC13 中断	0x0000_00B0
45	29	ADC14	可配置	ADC SOC14 中断	0x0000_00B4
46	30	ADC15	可配置	ADC SOC15 中断	0x0000_00B8
47	31	ADCPPU0	可配置	ADC 后处理单元 0 中断	0x0000_00BC
48	32	ADCPPU1	可配置	ADC 后处理单元 1 中断	0x0000_00C0
49	33	ADCPPU2	可配置	ADC 后处理单元 2 中断	0x0000_00C4

异常号	CMSIS 中断号	缩略语	优先级	描述	偏移地址
50	34	ADCPPU3	可配置	ADC 后处理单元 3 中断	0x0000_00C8
51	35	ADCPPU4	可配置	ADC 后处理单元 4 中断	0x0000_00CC
52	36	ADCPPU5	可配置	ADC 后处理单元 5 中断	0x0000_00D0
53	37	ADCPPU6	可配置	ADC 后处理单元 6 中断	0x0000_00D4
54	38	ADCPPU7	可配置	ADC 后处理单元 7 中断	0x0000_00D8
55	39	PWM0	可配置	PWM0 中断	0x0000_00DC
56	40	PWM1	可配置	PWM1 中断	0x0000_00E0
57	41	PWM2	可配置	PWM2 中断	0x0000_00E4
58	42	PWM3	可配置	PWM3 中断	0x0000_00E8
59	43	PWM4	可配置	PWM4 中断	0x0000_00EC
60	44	PWM5	可配置	PWM5 中断	0x0000_00F0
61	45	PWM6	可配置	PWM6 中断	0x0000_00F4
62	46	PWM7	可配置	PWM7 中断	0x0000_00F8
63	47	PWM0TZ	可配置	PWM0 trip-zone 中断	0x0000_00FC
64	48	PWM1TZ	可配置	PWM1 trip-zone 中断	0x0000_0100
65	49	PWM2TZ	可配置	PWM2 trip-zone 中断	0x0000_0104
66	50	PWM3TZ	可配置	PWM3 trip-zone 中断	0x0000_0108
67	51	PWM4TZ	可配置	PWM4 trip-zone 中断	0x0000_010C
68	52	PWM5TZ	可配置	PWM5 trip-zone 中断	0x0000_0110
69	53	PWM6TZ	可配置	PWM6 trip-zone 中断	0x0000_0114
70	54	PWM7TZ	可配置	PWM7 trip-zone 中断	0x0000_0118
71	55	ECAP	可配置	ECAP 中断	0x0000_011C
72	56	TIMER0	可配置	通用定时器 0 中断	0x0000_0120
73	57	TIMER1	可配置	通用定时器 1 中断	0x0000_0124
74	58	TIMER2	可配置	通用定时器 2 中断	0x0000_0128
75	59	CRC	可配置	CRC 中断	0x0000_012C
76	60	AES	可配置	AES 中断	0x0000_0130
77	61	DMAC	可配置	DMA 中断	0x0000_0134
78	62	MAILBOX	可配置	Mail-Box 中断	0x0000_0138

5.1.3 NVIC 中断配置

NVIC 中断可以通过写 SETENA 寄存器来使能或者通过写 CLRENA 寄存器来关闭。通过这种方式，使能或者关闭一个中断不会影响其他中断的使能状态。SETENA/CLRENA 寄存器都是 32 比特位宽的，每一个比特位代表一个中断输入。

NVIC 中断挂起状态可以通过中断挂起设置寄存器（SETPEND）和中断挂起清除寄存器（CLRPEND）来设置。用户可以通过 CLRPEND 寄存器来清除一个当前挂起的异常，或者通过 SETPEND 寄存器来产生软中断。

每一个外中断都有一个关联的优先级寄存器，这个优先级寄存器最大 8 比特位宽，最小 3 比特位宽。基于优先级组的配置，每一个优先级寄存器可以被进一步分成抢占优先级（preempt

priority level）和子优先级（sub-priority level）。SPC2168 使用了 4 比特的中断优先级，因而支持 16 个可编程优先级。

更多关于 NVIC 的配置，请参阅 ARM 相关的文档。

5.2 SysTick 定时器

Cortex-M4 处理器包含了一个集成的系统定时器，叫做 SysTick。SYSTICK 定时器是一个 24 比特位宽的向下计数的计数器。一旦它计数到 0，计数器会从 RELOAD 寄存器中加载重装载值。只有清除 SYSTICK 控制和状态寄存器的使能位，该定时器才会停止。

当 SYSTICK 定时器计数从 1 变为 0 的时候，它会置“1” SYSTICK 控制和状态寄存器的 COUNTFLAG 位。COUNTFLAG 位可以通过下面的任意一种方式清除：

- 通过处理器读取 SYSTICK 控制和状态寄存器
- 通过向 SYSTICK 当前值寄存器中写入任意值以清除 SYSTICK 计数器值

SYSTICK 计数器可以被用来产生固定间隔的 SYSTICK 中断。这对操作系统 OS、任务以及资源管理常常是必要的。置“1” TICKINT 位可以使能 SYSTICK 中断的产生。

更多关于 SysTick 定时器的配置，请参阅 ARM 相关的文档。

6 先进加密标准模块（AES）

6.1 AES 模块概述

SPC2168 实现了一个 AES（Advanced Encryption Standard）模块，可以用来加密或者解密数据。AES 模块支持的加解密模式有 ECB，CBC，CTR，CCM*和 MMO。密码长度最高为 256 位。

主要特性如下：

- 支持 5 种加解密模式 ECB，CBC，CTR，CCM*，MMO 和旁路模式（Bypass）
- 支持 128 位，192 位和 256 位密码长度
- 支持局部码（Partial code）
- 通过读写寄存器导入或者导出数据
- AES 中断（输出 FIFO 空，输入 FIFO 满，计算结束）
- 分组加密模式的错误指示
- 独立的 4x32 位输入和输出 FIFO
- 特殊加密模式
  - CTR：支持计数器模值从 16 到 128
  - CCM*：支持 0 到（2^{32}-1）字节关联字符串和消息字符串
  - 支持 11~13 字节 Nonce，2~4 字节的 L（15 –Nonce 长度）

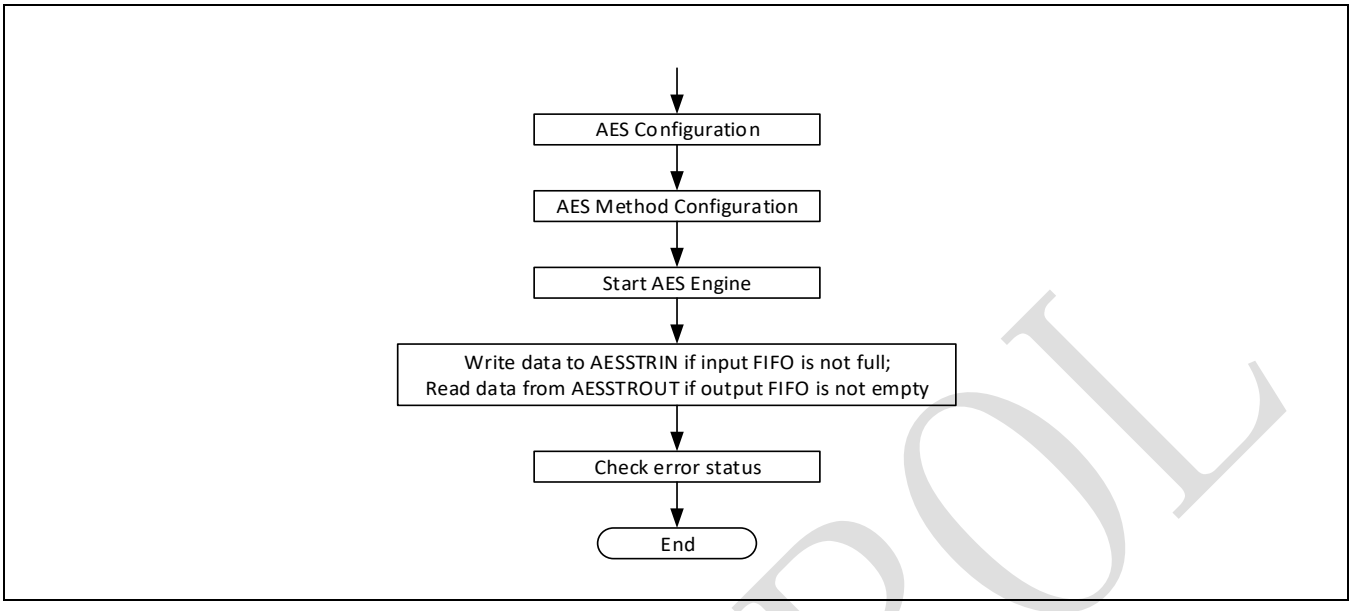
6.2 功能描述

AES 模块通过高效的硬件实现了 ECB，CBC，CTR，CCM*，MMO 分组加密模式和旁路（Bypass）模式。

6.2.1 AES 工作流程

AES 的工作流程如图 6-1 所示。

图 6-1：AES 工作流程图



AES 操作伪代码

```
AES_Config_Type aesConfig

aesConfig.mode <- AES_MODE_CBC
aesConfig.encDecSel <- AES_MODE_ENCRYPTION
aesConfig.keySize <- keysize
aesConfig.mStrLen <- length

for i=1 to keysize do
aesConfig.key[i] = key[i]

for i=1 to 4 do
aesConfig.initVect[i] = vector[i]

while j<length or k<length do
if AES input fifo not full do
feed the data plain_text[j]
j++

if AES output fifo not empty do
read the encryption data
k++
```

6.2.2 AES 配置

在启动 AES 模块之前，请按照如下步骤操作，确保 AES 模块被正确配置：

- 设置 AESCTL0.DECRYPTEN，选择加密模式还是解密模式。
- 设置 AESCTL0.MODE 位，配置 AES 分组加密模式（0 - ECB 模式，1 – CBC 模式，2 – CTR 模式，5 - CCM*模式，6 – MMO 模式以及 7 – 旁路模式）。
- 配置 AES 密码长度。AES 支持 3 钟密码长度：128 位，192 位和 256 位。通过设置 AESCTL0.KEYSIZE 来配置 AES 密码长度。

- 根据密码长度填充密码。AES 模块包含 8 个 32 位的密码寄存器，定义为 AESKEY0~7。
  - 当密码长度被配置为 128 位时，仅使用 AESKEY7/6/5/4
  - 当密码长度被配置为 192 位时，仅使用 AESKEY7/6/5/4/3/2
  - 当密码长度被配置为 256 位时，使用 AESKEY7/6/5/4/3/2/1/0
  - MMO 模式不支持 192 位和 256 位密码长度
  - Bypass 模式忽略密码长度
- 除了 CCM*模式，其他模式都通过 AESMSTRLEN 寄存器来设置输入字串数据长度。在 CCM*模式下，通过 AESASTRLEN 寄存器来设置关联字串数据长度，通过 AESMSTRLEN 寄存器来设置消息字串数据长度。
- 在 CTR 模式下，通过 AESCTL0.CTRMOD 来设置 CTR 模式的计数器模值。
- 在 CCM*加密模式或者 MMO 模式下
  - 如果需要 MIC/HASH 数据，置“1”AESCTL0.OUTMIC 位，那么 MIC/HASH 数据会追加在输出数据后面。
  - 如果只需要 MIC/HASH 数据，我们可以阻止已加密的数据进入输出 FIFO（通过置“1”AESCTL0.OUTMSG 位），然后从 AESOV3/2/1/0 获取 MIC/HASH 数据。
- 在 CCM*加密模式下，如果需要的话，可以置“1”AESCTL0.OUTHDR 位，那么会在输出数据流的开始输出数据 B0。
- 根据 AES 分组加密模式填充初始值。AES 模块包含 4 个 32 位的初始化向量 AESIV0/1/2/3。
  - ECB/MMO/BYPASS 模式
    不需要配置初始化向量
  - CTR 模式
    设置 AESIV0 = 初始计数值
    设置 AESIV1 = Nonce[31:0]
    设置 AESIV2 = Nonce[63:32]
    设置 AESIV3 = Nonce[95:64]
  - CCM*模式
    设置 AESIV0 = Nonce[31:0]
    设置 AESIV1 = Nonce[63:32]
    设置 AESIV2 = Nonce[95:64]
    设置 AESIV3[7:0] = Nonce[103:96]
    设置 AESIV3[15:8] = [15-（Nonce 长度）]

注意： Bypass 模式下，AES 模块会忽略输入数据；AES 模块会将输入数据不做修改传递到输出接口。

6.2.3 数据访问方法

AES 模块包含独立的 4 个 32 位的输入 FIFO 和输出 FIFO。CPU 可以访问这两个 FIFO。当输入 FIFO 非满时，CPU 可以向 AESSTRIN 寄存器中写入数据；当输出 FIFO 非空时，CPU 可以从 AESSTROUT 寄存器读取数据。当已传输的数据大小达到输入和输出的数据大小时，则 AES 模块的工作完成。

6.2.4 启动 AES 模块

在启动 AES 模块之前，需要先清除输入 FIFO 和输出 FIFO。置“1”AESCTL0.IFIFOCLR 位可以清除输入 FIFO；置“1”AESCTL0.OFIFOCLR 位可以清除输出 FIFO。通过置“1” AESCTL0.START 位可以启动 AES 模块进行工作。

6.2.5 中断请求

AES 模块具有三个中断源：输入 FIFO 满，输出 FIFO 空以及 AES 计算完成。每个中断都可以通过设置 AESIE/AESIC 寄存器来使能和关闭。

6.2.6 局部码（Partial code）支持

当输入数据大小不是 128 位的整数倍时，AES 模块会对输入数据进行自动填充。AES 模块针对不同的分组加密模式支持不同的填充方案，如表 6-1 所示。

表 6-1：填充方案

模式	描述
CCM*	对关联字串（A）和消息字串（M）都自动填充 0
MMO	自动填充“100…00”+ 2 字节长度信息
CBC	对局部码采用密文窃取（Cipher stealing）技术，这个技术是 Spintrol 专有的，不是根据 NIST 的技术标准
CTR	局部码不影响操作
ECB	检查是否需要局部码，当检测到需要局部码时则报错

6.2.7 错误状态检查

当 AES 工作完成后，AESSTS.ERRCODE 寄存器位段记录 AES 模块的错误状态。不同 AES 分组加密模式的错误码如表 6-2 所示。

表 6-2：不同 AES 分组加密模式的错误码

模式	ERRCODE[2]	ERRCODE [1]	ERRCODE [0]
ECB	N/A	数据长度不是 16 字节整数倍	输入数据长度小于 16 字节
CBC	N/A	N/A	
CTR	N/A	N/A	
CCM*	解密过程中 MIC 值不匹配	N/A	N/A
MMO	N/A	数据长度大于 2^{13}-1 字节	N/A
Bypass	N/A	N/A	N/A

6.2.8 输出向量

AES 模块的输出向量提供了一些有用的信息，例如 CBC 模式下最后一个密文块，CTR 模式下最后的计数值，CCM*模式下的 IC 值，以及 MMO 模式下的 ASH 值。对于不同的 AES 分组加密模式，AESOV3/2/1/0 寄存器记录这些有用的信息。表 6-3 列出了不同分组加密模式下，AES 输出向量所记录的信息。

表 6-3：AES 输出向量

分组加密模式	输出向量
ECB	N/A
CBC	最后一个密文快（cipher） AESOV0 = cipher[31:0] AESOV1 = cipher[63:32] AESOV2 = cipher[95:64] AESOV3 = cipher[127:96]
CTR	最后的计数值（counter） AESOV0 = counter[31:0] AESOV1 = counter[63:32] AESOV2 = counter[95:64] AESOV3 = counter[127:96]
CCM*	加密模式：MIC 值。如果 MIC 值小于 32 字节，总是输出向量高字节（MSB）被使用。 例如：8 字节 MIC AESOV2 = MIC[31:0] AESOV3 = MIC[63:32]
MMO	HASH 值 AESOV0 = HASH[31:0] AESOV1 = HASH[63:32] AESOV2 = HASH[95:64] AESOV3 = HASH[127:96]
Bypass	N/A

6.3 寄存器

6.3.1 AES 寄存器列表

表 6-4：AES 模块基地址

外设模块	基地址
AES	0x4000 8400

表 6-5：AES 寄存器列表

寄存器	偏移地址	描述	复位值
AESCTL0	0x00	AES 控制寄存器 0	0x00004010
AESSTS	0x08	AES 状态寄存器	0x00000081
AESASTRLEN	0x0C	AES 关联字串长度寄存器	0x00000000
AESMSTRLEN	0x10	AES 消息字串长度寄存器	0x00000000
AESSTRIN	0x14	AES 输入消息字寄存器	0x00000000
AESIV0	0x18	AES 初始向量寄存器 0	0x00000000
AESIV1	0x1C	AES 初始向量寄存器 1	0x00000000
AESIV2	0x20	AES 初始向量寄存器 2	0x00000000
AESIV3	0x24	AES 初始向量寄存器 3	0x00000000
AESKEY0	0x28	AES 密码寄存器 0	0x00000000
AESKEY1	0x2C	AES 密码寄存器 1	0x00000000
AESKEY2	0x30	AES 密码寄存器 2	0x00000000
AESKEY3	0x34	AES 密码寄存器 3	0x00000000
AESKEY4	0x38	AES 密码寄存器 4	0x00000000
AESKEY5	0x3C	AES 密码寄存器 5	0x00000000
AESKEY6	0x40	AES 密码寄存器 6	0x00000000
AESKEY7	0x44	AES 密码寄存器 7	0x00000000
AESSTROUT	0x48	AES 输出消息字寄存器	0x00000000
AESOV0	0x4C	AES 输出向量寄存器 0	0x00000000
AESOV1	0x50	AES 输出向量寄存器 1	0x00000000
AESOV2	0x54	AES 输出向量寄存器 2	0x00000000
AESOV3	0x58	AES 输出向量寄存器 3	0x00000000
AESIF	0x5C	AES 中断状态寄存器	0x00000000
AESIE	0x60	AES 中断使能寄存器	0x00000000
AESRAWIF	0x64	AES 中断原始状态寄存器	0x00000000
AESIC	0x68	AES 中断清除寄存器	0x00000000

6.3.2 AES 寄存器

表 6-6 到

表 6-57 提供了 AES 模块相关寄存器的详细信息。

表 6-6：AES 控制寄存器 0（AESCTL0）位段定义

AESCTL0 (AES Control Register 0) Offset: 0x0 Default: 0x00004010
Access: AES -> AESCTL0.all

31	30	29	28	27	26	25	24
RESERVED						CTRMOD	
23	22	21	20	19	18	17	16
CTRMOD					MODE		
15	14	13	12	11	10	9	8
DECRYPTEN	OUTMIC	MICLEN		KEYSIZE		RESERVED	RESERVED
7	6	5	4	3	2	1	0
RESERVED		OUTHDR	OUTMSG	OFIFOCLR	IFIFOCLR	RESERVED	START

表 6-7：AES 控制寄存器 0（AESCTL0）位段描述

位段	位段名	属性	复位值	描述
31:26	RESERVED_31_26	RO	0x0	保留
25:19	CTRMOD	RW	0x0	CTR 模式的计数器模值 000xxxx：模值 = 2^{128} 其他： 模值 = $2^{CTRMODE}$
18:16	MODE	RW	0x0	AES 运行模式 000：ECB 001：CBC 010：CTR 011：保留 100：保留 101：CCM* 110：MMO 111：BYPASS
15	DECRYPTEN	RW	0x0	解密操作，MMO 和 BYPASS 模式下该位 0：加密 1：解密
14	OUTMIC	RW	0x1	在 CCM*加密模式或者 MMO 模式下，在输出数据流后面附加 MIC/HASH 值 0：不附加 MIC/HASH 值 1：附加 MIC/HASH 值
13:12	MICLEN	RW	0x0	MIC 值长度 00：0 字节 01：4 字节 10：8 字节 11：16 字节
11:10	KEYSIZE	RW	0x0	密码长度

位段	位段名	属性	复位值	描述
				00：16 字节 01：32 字节 10：24 字节 11：无效选项
9	RESERVED_9	RW	0x0	保留
8	RESERVED_8	RW	0x0	保留
7:6	RESERVED_7_6	RO	0x0	保留
5	OUTHDR	RW	0x0	在 CCM*模式下，输出 B0 和 l(a) 0：不在输出数据流的开始输出 B0 和 l(a) 1：在输出数据流的开始输出 B0 和 l(a)
4	OUTMSG	RW	0x1	输出数据流到输出 FIFO 0：阻止输出数据流到输出 FIFO 1：允许输出数据流到输出 FIFO
3	OFIFOCLR	W1C	0x0	清空输出 FIFO 0：写 0 无效，总是读回 0 1：写 1 清空输出 FIFO，本位段自动清零
2	IFIFOCLR	W1C	0x0	清空输入 FIFO 0：写 0 无效，总是读回 0 1：写 1 清空输入 FIFO，本位段自动清零
1	RESERVED_1	RO	0x0	保留
0	START	W1S	0x0	启动 AES 0：写 0 无效，总是读回 0 1：写 1 启动 AES 工作，本位段自动清零

表 6-8：AES 状态寄存器（AESSTS）位段定义

AESSTS (AES Status Register) Offset: 0x8 Default: 0x00000081
Access: AES -> AESSTS.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				OFIFODEPTH			IFIFODEPTH
15	14	13	12	11	10	9	8
IFIFODEPTH		ERRCODE			RESERVED		
7	6	5	4	3	2	1	0
OFIFOEMPTY	OFIFORDY	RESERVED	IFIFOFULL	RESERVED			DONE

表 6-9：AES 状态寄存器（AESSTS）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:17	OFIFODEPTH	RO	0x0	输出 FIFO 深度

位段	位段名	属性	复位值	描述
16:14	IFIFODEPTH	RO	0x0	输入 FIFO 深度
13:11	ERRCODE	RO	0x0	AES 工作错误状态 000：无错误 001：ECB、CBC 和 CTR 模式下，输入数据流长度小于 16 字节 010：ECB 模式下，数据长度不是 16 字节的整数倍；或者 MMO 模式下，数据长度大于 2^{13}-1 字节 011：ECB 模式下，数据长度不是 16 字节的整数倍且小于 16 字节 100：在 CCM*解密模式时，MIC 值不匹配 其他：无效
10:8	RESERVED_10_8	RO	0x0	保留
7	OFIFOEMPTY	RO	0x1	输出 FIFO 空 0：输出 FIFO 非空 1：输出 FIFO 空
6	OFIFORDY	RO	0x0	输出 FIFO 读就绪 0：输出 FIFO 读未就绪 1：输出 FIFO 读就绪
5	RESERVED_5	RO	0x0	保留
4	IFIFOFULL	RO	0x0	输入 FIFO 满 0：输入 FIFO 非满 1：输入 FIFO 满
3:1	RESERVED_3_1	RO	0x0	保留
0	DONE	RO	0x1	AES 工作完成 0：AES 工作未完成 1：AES 工作完成

表 6-10：AES 关联字串长度寄存器（AESASTRLEN）位段定义

AESASTRLEN (AES Associate String Length Register) Offset: 0xC Default: 0x00000000 Access: AES -> AESASTRLEN.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-11：AES 关联字串长度寄存器（AESASTRLEN）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	关联字串大小

表 6-12：AES 消息字串长度寄存器（AESMSTRLEN）位段定义

AESMSTRLEN (AES Message String Length Register) Offset: 0x10 Default: 0x00000000
Access: AES -> AESMSTRLEN.all

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

表 6-13：AES 消息字串长度寄存器（AESMSTRLEN）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	消息字串大小

表 6-14：AES 输入消息字寄存器（AESSTRIN）位段定义

AESSTRIN (AES Input Message Word Register) Offset: 0x14 Default: 0x00000000 Access: AES -> AESSTRIN.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-15：AES 输入消息字寄存器（AESSTRIN）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	WO	0x0	输入消息字

表 6-16：AES 初始向量寄存器 0（AESIV0）位段定义

AESIV0 (AES Initial Vector Register 0) Offset: 0x18 Default: 0x00000000 Access: AES -> AESIV0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-17：AES 初始向量寄存器 0（AESIV0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	初始向量的字节 0~3

表 6-18：AES 初始向量寄存器 1（AESIV1）位段定义

AESIV1 (AES Initial Vector Register 1) Offset: 0x1C Default: 0x00000000 Access: AES -> AESIV1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-19：AES 初始向量寄存器 1（AESIV1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	初始向量的字节 4~7

表 6-20：AES 初始向量寄存器 2（AESIV2）位段定义

AESIV2 (AES Initial Vector Register 2) Offset: 0x20 Default: 0x00000000
Access: AES -> AESIV2.all

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

表 6-21：AES 初始向量寄存器 2（AESIV2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	初始向量的字节 8~11

表 6-22：AES 初始向量寄存器 3（AESIV3）位段定义

AESIV3 (AES Initial Vector Register 3) Offset: 0x24 Default: 0x00000000
Access: AES -> AESIV3.all

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

表 6-23：AES 初始向量寄存器 3（AESIV3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	初始向量的字节 12~15

表 6-24：AES 密码寄存器 0（AESKEY0）位段定义

AESKEY0 (AES Key Register 0) Offset: 0x28 Default: 0x00000000 Access: AES -> AESKEY0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-25：AES 密码寄存器 0（AESKEY0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 0~3

表 6-26：AES 密码寄存器 1（AESKEY1）位段定义

AESKEY1 (AES Key Register 1) Offset: 0x2C Default: 0x00000000 Access: AES -> AESKEY1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-27：AES 密码寄存器 1（AESKEY1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 4~7

表 6-28：AES 密码寄存器 2（AESKEY2）位段定义

AESKEY2 (AES Key Register 2) Offset: 0x30 Default: 0x00000000 Access: AES -> AESKEY2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-29：AES 密码寄存器 2（AESKEY2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 8~11

表 6-30：AES 密码寄存器 3（AESKEY3）位段定义

AESKEY3 (AES Key Register 3) Offset: 0x34 Default: 0x00000000 Access: AES -> AESKEY3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-31：AES 密码寄存器 3（AESKEY3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 12~15

表 6-32：AES 密码寄存器 4（AESKEY4）位段定义

AESKEY4 (AES Key Register 4) Offset: 0x38 Default: 0x00000000 Access: AES -> AESKEY4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-33：AES 密码寄存器 4（AESKEY4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 16~19

表 6-34：AES 密码寄存器 5（AESKEY5）位段定义

AESKEY5 (AES Key Register 5) Offset: 0x3C Default: 0x00000000 Access: AES -> AESKEY5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-35：AES 密码寄存器 5（AESKEY5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 20~23

表 6-36：AES 密码寄存器 6（AESKEY6）位段定义

AESKEY6 (AES Key Register 6) Offset: 0x40 Default: 0x00000000 Access: AES -> AESKEY6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-37：AES 密码寄存器 6（AESKEY6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 24~27

表 6-38：AES 密码寄存器 7（AESKEY7）位段定义

AESKEY7 (AES Key Register 7) Offset: 0x44 Default: 0x00000000 Access: AES -> AESKEY7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-39：AES 密码寄存器 7（AESKEY7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	密码字节 28~31

表 6-40：AES 输出消息字寄存器（AESSTROUT）位段定义

AESSTROUT (AES Output Message Word Register) Offset: 0x48 Default: 0x00000000 Access: AES -> AESSTROUT.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-41：AES 输出消息字寄存器（AESSTROUT）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出消息字

表 6-42：AES 输出向量寄存器 0（AESOV0）位段定义

AESOV0 (AES Output Vector Register 0) Offset: 0x4C Default: 0x00000000 Access: AES -> AESOV0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-43：AES 输出向量寄存器 0（AESOV0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出向量字节 0~3

表 6-44：AES 输出向量寄存器 1（AESOV1）位段定义

AESOV1 (AES Output Vector Register 1) Offset: 0x50 Default: 0x00000000 Access: AES -> AESOV1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-45：AES 输出向量寄存器 1（AESOV1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出向量字节 4~7

表 6-46：AES 输出向量寄存器 2（AESOV2）位段定义

AESOV2 (AES Output Vector Register 2) Offset: 0x54 Default: 0x00000000 Access: AES -> AESOV2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-47：AES 输出向量寄存器 2（AESOV2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出向量字节 8~11

表 6-48：AES 输出向量寄存器 3（AESOV3）位段定义

AESOV3 (AES Output Vector Register 3) Offset: 0x58 Default: 0x00000000 Access: AES -> AESOV3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 6-49：AES 输出向量寄存器 3（AESOV3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出向量字节 12~15

表 6-50：AES 中断状态寄存器（AESIF）位段定义

AESIF (AES Interrupt Status Register) Offset: 0x5C Default: 0x00000000 Access: AES -> AESIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					OFIFOEMPTY	IFIFOFULL	DONE

表 6-51：AES 中断状态寄存器（AESIF）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	OFIFOEMPTY	RO	0x0	AES 输出 FIFO 空中断状态 0：AES 输出 FIFO 空中断未发生 1：AES 输出 FIFO 空中断已发生
1	IFIFOFULL	RO	0x0	AES 输入 FIFO 满中断状态 0：AES 输入 FIFO 满中断未发生 1：AES 输入 FIFO 满中断已发生
0	DONE	RO	0x0	AES 工作完成中断状态 0：AES 工作完成中断未发生 1：AES 工作完成中断已发生

表 6-52：AES 中断使能寄存器（AESIE）位段定义

AESIE (AES Interrupt Enable Register) Offset: 0x60 Default: 0x00000000 Access: AES -> AESIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					OFIFOEMPTY	IFIFOFULL	DONE

表 6-53：AES 中断使能寄存器（AESIE）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	Reserved
2	OFIFOEMPTY	RW	0x0	使能输出 FIFO 空中断 0：关闭输出 FIFO 空中断 1：使能输出 FIFO 空中断
1	IFIFOFULL	RW	0x0	使能输入 FIFO 满中断 0：关闭输入 FIFO 满中断 1：使能输入 FIFO 满中断
0	DONE	RW	0x0	使能 AES 工作完成中断 0：关闭 AES 工作完成中断 1：使能 AES 工作完成中断

表 6-54：AES 中断原始状态寄存器（AESRAWIF）位段定义

AESRAWIF (AES Interrupt RAW status Register) Offset: 0x64 Default: 0x00000000 Access: AES -> AESRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					OFIFOEMPTY	IFIFOFULL	DONE

表 6-55：AES 中断原始状态寄存器（AESRAWIF）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	OFIFOEMPTY	RO	0x0	AES 输出 FIFO 空中断原始状态 0：AES 输出 FIFO 空中断未发生 1：AES 输出 FIFO 空中断已发生
1	IFIFOFULL	RO	0x0	AES 输入 FIFO 满中断原始状态 0：AES 输入 FIFO 满中断未发生 1：AES 输入 FIFO 满中断已发生
0	DONE	RO	0x0	AES 工作完成中断原始状态 0：AES 工作完成中断未发生 1：AES 工作完成中断已发生

表 6-56：AES 中断清除寄存器（AESIC）位段定义

AESIC (AES Interrupt Clear Register)　Offset: 0x68　Default: 0x00000000
Access: AES -> AESIC.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					OFIFOEMPTY	IFIFOFULL	DONE

表 6-57：AES 中断清除寄存器（AESIC）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	Reserved
2	OFIFOEMPTY	W1C	0x0	清除输出 FIFO 空中断状态位和原始状态位 0：写 0 无效，总是读回 0 1：写 1 清除输出 FIFO 空中断状态位和原始状态位，本位段自动清零
1	IFIFOFULL	W1C	0x0	清除输入 FIFO 满中断状态位和原始状态位 0：写 0 无效，总是读回 0 1：写 1 清除输入 FIFO 满中断状态位和原始状态位，本位段自动清零
0	DONE	W1C	0x0	清除 AES 工作完成中断状态位和原始状态位 0：写 0 无效，总是读回 0 1：写 1 清除 AES 工作完成中断状态位和原始状态位，本位段自动清零

7 循环冗余校验模块（CRC）

7.1 CRC 概述

循环冗余校验（Cyclic Redundancy Check，CRC）或者叫做多项式码校验是一种哈希函数，被设计用于检查原始计算数据的意外变化。因此，CRC 被用于数据传输或者存储过程中的错误检测方面。CRC 校验码最多为 32 比特。

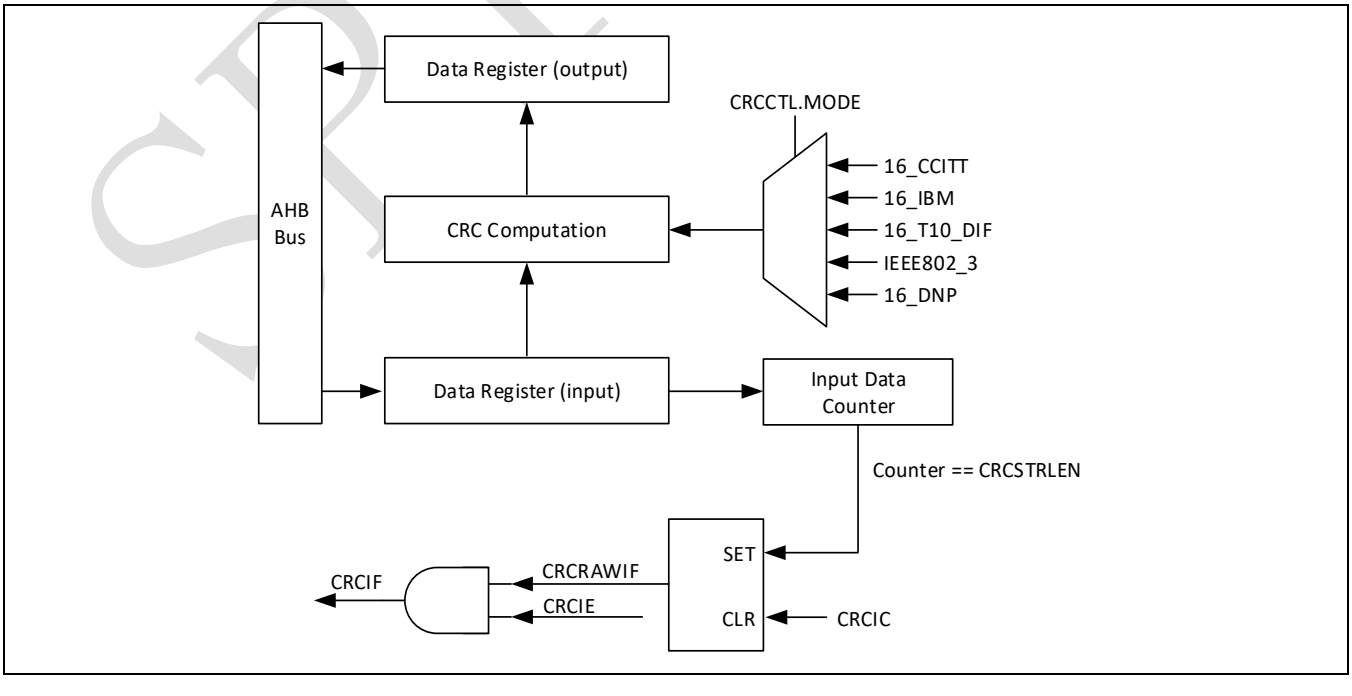
7.2 主要特性

一个标准的 AHB 总线从接口被用于配置 CRC 模块、接收数据比特流以及输出 CRC 结果。CRC 模块支持如下特性：

- 支持 32 位并行数据比特流输入，支持多大 32 位 CRC 值输出
- 支持多大 2^{32}（4294967296）字节长度的数据的 CRC 计算
- 支持以下 CRC 标准
  - CRC-16-CCITT，多项式为 $x^{16}+x^{12}+x^5+1$
  - CRC-16-IBM，多项式为 $x^{16}+x^{15}+x^2+1$
  - CRC-16-T10-DIF，多项式为 $x^{16}+x^{15}+x^{11}+x^9+x^8+x^7+x^5+x^4+x^2+1$
  - CRC-32-IEEE 802.3，多项式为 $X^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
  - CRC-16-DNP，多项式为 $x^{16}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+x^5+x^2+1$
  -

CRC 模块框图如图 7-1 所示。

图 7-1：CRC 模块框图



7.3 CRC 工作流程

用户可以根据以下步骤操作 CRC 模块：

步骤 1：关闭 CRC（设置 CRCCTL.EN 位为 0）

步骤 2：关闭中断（设置 CRCIE.DONE 位为 0）

步骤 3：清除中断标志（设置 CRCIC.DONE 为 1）

步骤 4：配置数据流长度（设置 CRCSTRLEN 寄存器）

步骤 5：配置 CRC 工作模式（设置 CRCCTL.MODE 位段)

步骤 6：使能中断（设置 CRCIE.DONE 位为 1)

步骤 7：使能 CRC（设置 CRCCTL.EN 位为 1)

步骤 8：写入数据流，等待中断发生（如果中断发生，转到步骤 9）

步骤 9：获取 CRC 计算结果（读 CRCRESULT 寄存器）

步骤 10：CRC 操作完成

注意： CRC 数据流输入寄存器一次接受一个字（32 位）数据。如果输入数据不是 4 字节对齐的，则会在数据流的开始填充 0。例如，如果数据流由 5 个字节构成，从低地址开始依次为：0xA1，0xA2，0xA3，0xA4，0xA5，那么下面两个字数据应该被写入数据流输入寄存器：

- 0xA1000000
- 0xA5A4A3A2

CRC 结果的位序为：

- 16 位 CRC：x_0~x_{15} [MSB->LSB]
- 32 位 CRC：x_0~x_{31} [MSB->LSB]

7.4 寄存器

7.4.1 CRC 寄存器列表

表 7-1：CRC 模块基地址

外设模块	基地址
CRC	0x4000 8000

表 7-2：CRC 寄存器列表

寄存器	偏移地址	描述	复位值
CRCIF	0x00	CRC 中断标志寄存器	0x00000000
CRCRAWIF	0x04	CRC 中断原始标志寄存器	0x00000000
CRCIC	0x08	CRC 中断清除寄存器	0x00000000
CRCIE	0x0C	CRC 中断使能寄存器	0x00000000
CRCCTL	0x10	CRC 控制寄存器	0x00000000
CRCSTRLEN	0x14	CRC 数据流长度寄存器	0x00000000
CRCSTRIN	0x18	CRC 数据流输入寄存器	0x00000000
CRCRESULT	0x1C	CRC 结果寄存器	0x00000000

7.4.2 CRC 寄存器

表 7-3 到表 7-18 提供了 CRC 模块相关寄存器的详细信息。

表 7-3：CRC 中断标志寄存器（CRCIF）位段定义

CRCIF (CRC Interrupt Flag Register) Offset: 0x0 Default: 0x00000000 Access: CRC -> CRCIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							DONE

表 7-4：CRC 中断标志寄存器（CRCIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	DONE	RO	0x0	CRC 计算完成中断标志 0：中断未发生 1：中断已发生

表 7-5：CRC 中断原始标志寄存器（CRCRAWIF）位段定义

CRCRAWIF (CRC Raw Interrupt Flag Register) Offset: 0x4 Default: 0x00000000 Access: CRC -> CRCRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							DONE

表 7-6：CRC 中断原始标志寄存器（CRCRAWIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	DONE	RO	0x0	CRC 计算完成中断原始标志 0：中断未发生 1：中断已发生

表 7-7：CRC 中断清除寄存器（CRCIC）位段定义

CRCIC (CRC Interrupt Clear Register) Offset: 0x8 Default: 0x00000000 Access: CRC -> CRCIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							DONE

表 7-8：CRC 中断清除寄存器（CRCIC）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	DONE	W1C	0x0	清除 CRCIF 和 CRCRAWIF 寄存器中的标志位 0：写 0 无效，总是读回 0 1：写 1 清除 CRCIF 和 CRCRAWIF 寄存器中的标志位，本位段自动清零

表 7-9：CRC 中断使能寄存器（CRCIE）位段定义

CRCIE (CRC Interrupt Enable Register) Offset: 0xC Default: 0x00000000 Access: CRC -> CRCIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							DONE

表 7-10：CRC 中断使能寄存器（CRCIE）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	DONE	RW	0x0	使能中断（由 CRCRAWIF 标志位触发） 0：关闭中断请求以及 CRCIF 寄存器中相关标志位的产生 1：使能中断请求以及 CRCIF 寄存器中相关标志位的产生

表 7-11：CRC 控制寄存器（CRCCTL）位段定义

CRCCTL (CRC Control Register) Offset: 0x10 Default: 0x00000000 Access: CRC -> CRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				MODE			EN

表 7-12：CRC 控制寄存器（CRCCRL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:1	MODE	RW	0x0	CRC 模式选择 000：$x^{16}+x^{12}+x^5+1$ (CRC-16-CCITT, CRC-CCITT) 001：$x^{16}+x^{15}+x^2+1$ (CRC-16, CRC-16-IBM, CRC-16-ANSI) 010：$x^{16}+x^{15}+x^{11}+x^9+x^8+x^7+x^5+x^4+x^2+x+1$ (CRC-16-T10-DIF) 011：$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$ (CRC-32-IEEE802.3) 100：$x^{16}+x^{13}+x^{12}+x^{11}+x^{10}+x^8+x^6+x^5+x^2+1$ (CRC-16-DNP) 101：保留 110：保留 111：保留
0	EN	RW	0x0	使能 CRC 计算 0：关闭 CRC 计算 1：使能 CRC 计算，CRC 计算完成后自动清零

表 7-13：CRC 数据流长度寄存器（CRCSTRLEN）位段定义

CRCSTRLEN (CRC Stream Length Register) Offset: 0x14 Default: 0x00000000 Access: CRC -> CRCSTRLEN.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 7-14：CRC 数据流长度寄存器（CRCSTRLEN）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	输入数据流长度 –1（以字节为单位）

表 7-15：CRC 数据流输入寄存器（CRCSTRIN）位段定义

CRCSTRIN (CRC Stream Input Register) Offset: 0x18 Default: 0x00000000
Access: CRC -> CRCSTRIN.all

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

表 7-16：CRC 数据流输入寄存器（CRCSTRIN）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	输入数据寄存器

表 7-17：CRC 结果寄存器（CRCRESULT）位段定义

CRCRESULT (CRC Result Register) Offset: 0x1C Default: 0x00000000
Access: CRC -> CRCRESULT.all

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

表 7-18：CRC 结果寄存器（CRCRESULT）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	CRC 计算结果

8 通用定时器

8.1 概述

SPC2168 内置 3 个 32 位通用定时器（General Purpose Timers，GPT），其特性如下：

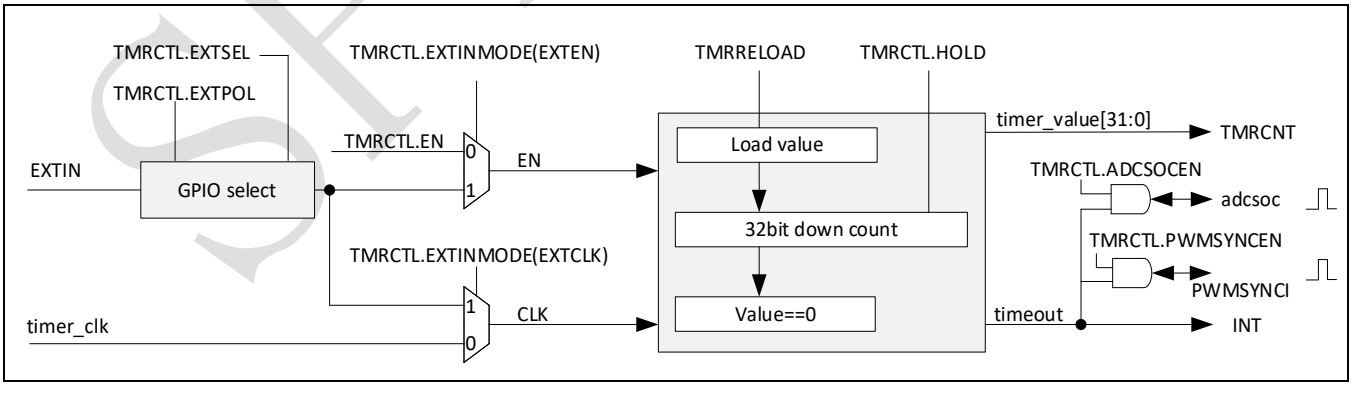
- APB 总线寄存器接口
- 专属 TIMER 时钟，参见图 3-13
- 32 位向下计数器
- 当计数到 0 时产生中断
- 当计数到 0 时产生 ADCSOC 事件
- 当计数到 0 时产生 PWMSYNCI 事件
- 捕获外部输入作为定时器的使能信号
- 捕获外部输入作为定时器的时钟源

8.2 功能描述

每一个通用定时器（GPT）都工作在周期性计时模式。如图 8-1 所示，当使能定时器之后，计数器加载 TMRRELOAD 寄存器的值，并开始向下计数直到 0。然后，计数器从 TMRRELOAD 寄存器的值重新开始，并向下计数直到 0。当计数器向下计数到 0 时，有如下动作发生：

- 如果使能了定时器中断，将会有中断产生，这个中断标志将会一直存在直到被手动清除
- 如果 TMRCTL.ADCSOCEN 使能，将会产生一个 ADCSOC 事件
- 如果 TMRCTL.PWMSYNCEN 使能，将会产生一个 PWMSYNCI 事件

图 8-1：通用定时器结构



外部信号输入脚（EXTIN）从低电平到高电平的转变，可以被用作定时器的使能信号；这个信号脚（EXTIN）也可以被用作外部时钟接入的 PIN 脚。通过寄存器 TMRCTL.EXTSEL 可以选择不同的 GPIO 作为外部输入信号脚（EXTIN）。详细信息，请参见表 8-3 和表 8-4。

在系统复位之后，定时器默认为关闭状态。

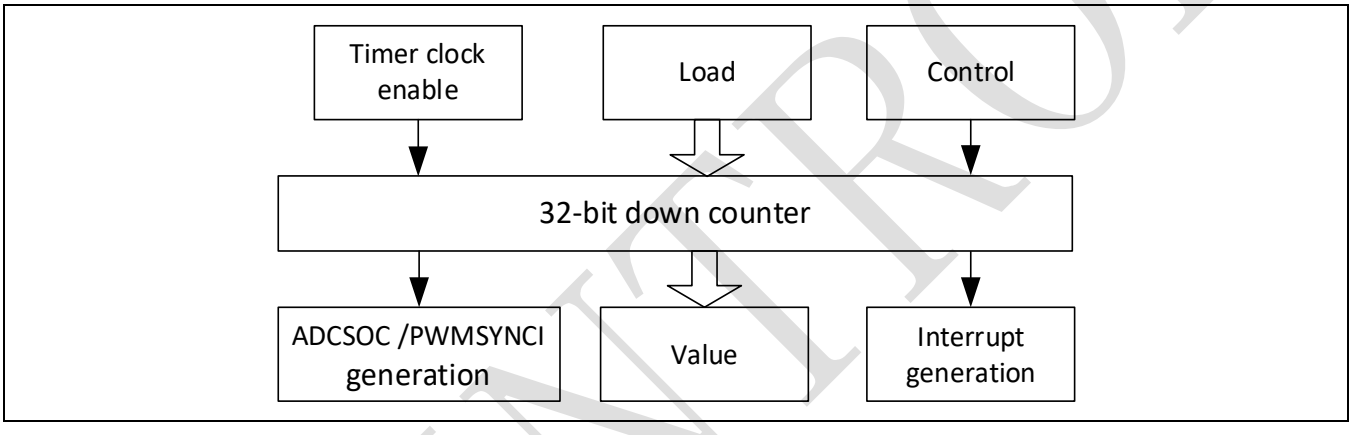
8.3 初始化流程

应当遵照下面的流程来启动定时器：

- 关闭定时器，并清除定时器中断
- 配置定时器重加载寄存器（TMRRELOAD）
- 配置定时器控制寄存器（TMRCTL）
- 使能定时器开始计数

定时器运行运行流程框图如图 8-2 所示。

图 8-2：通用定时器运行流程图



8.4 寄存器

8.4.1 TIMER 寄存器列表

表 8-1：TIMER 模块基地址

外设模块	基地址
TIMER0	0x4000 7000
TIMER1	0x4000 7020
TIMER2	0x4000 7040

表 8-2：TIMER 寄存器列表

寄存器	偏移地址	描述	复位值
TMRCTL	0x00	定时器控制寄存器	0x00000000
TMRCNT	0x04	定时器计数器值寄存器	0x00000000
TMRRELOAD	0x08	定时器超时重载寄存器	0x00000000
TMRIF	0x0C	定时器中断标志寄存器	0x00000000
TMRRAWIF	0x10	定时器中断原始标志寄存器	0x00000000
TMRIE	0x14	定时器中断使能寄存器	0x00000000
TMRIFRC	0x18	定时器中断强制寄存器	0x00000000
TMRIC	0x1C	定时器中断清除寄存器	0x00000000

8.4.2 TIMER 寄存器

表 8-3 到表 8-18 提供了定时器模块相关寄存器的详细信息。

表 8-3：定时器控制寄存器（TMRCTL）位段定义

TMRCTL (Timer Control Register) Offset: 0x0 Default: 0x00000000 Access: TIMER0 -> TMRCTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_13							
23	22	21	20	19	18	17	16
RESERVED_31_13							
15	14	13	12	11	10	9	8
RESERVED_31_13			EXTSEL				
7	6	5	4	3	2	1	0
EXTSEL	EXTPOL	EXTINMODE		PWMSYNCEN	ADCSOCEN	RESERVED_1	EN

表 8-4：定时器控制寄存器（TMRCTL）位段描述

位段	位段名	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12:7	EXTSEL	RW	0x0	外部输入源选择（GPIO 管脚）
6	EXTPOL	RW	0x0	外部输入极性 0：低有效 1：高有效
5:4	EXTINMODE	RW	0x0	外部输入模式。 00：关闭外部输入 01：无效 10：外部输入作为定时器时钟信号 11：外部输入作为定时器使能信号
3	PWMSYNCEN	RW	0x0	使能 PWMSYNCI 信号产生 0：不产生 PWMSYNCI 信号 1：当 TMRCNT 向下计数到 0 时，产生 PWMSYNCI
2	ADCSOCEN	RW	0x0	使能 ADCSOC 信号产生 0：不产生 ADCSOC 信号 1：当 TMRCNT 向下计数到 0 时，产生 ADCSOC
1	RESERVED_1	RW	0x0	保留
0	EN	RW	0x0	使能定时器 0：关闭定时器 1：使能定时器

表 8-5：定时器计数器值寄存器（TMRCNT）位段定义

TMRCNT (Timer Counter Value Register) Offset: 0x4 Default: 0x00000000 Access: TIMER -> TMRCNT.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 8-6：定时器计数器值寄存器（TMRCNT）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	定时器计数器的当前值

表 8-7：定时器超时重载寄存器（TMRRELOAD）位段定义

TMRRELOAD (Timer Reload Value Register) Offset: 0x8 Default: 0x00000000 Access: TIMER -> TMRRELOAD.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 8-8：定时器超时重载寄存器（TMRRELOAD）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	计数器超时重载值 向本寄存器写入数值时，也会设置 TMRCNT 寄

表 8-9：定时器中断标志寄存器（TMRIF）位段定义

TMRIF (Timer Interrupt Flag Register) Offset: 0xC Default: 0x00000000 Access: TIMER -> TMRIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 8-10：定时器中断标志寄存器（TMRIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	RO	0x0	定时器中断标志 0：中断没有产生。 1：中断已产生，并已经通知 CPU 在这个标志被清楚之前，不会有新的中断产生。

表 8-11：定时器中断原始标志寄存器（TMRRAWIF）位段定义

TMRRAWIF (Timer Raw Interrupt Flag Register) Offset: 0x10 Default: 0x00000000 Access: TIMER -> TMRRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 8-12：定时器中断原始标志寄存器（TMRRAWIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	RO	0x0	定时器原始中断状态 当 TMRCNT 向下计数到 0 时，本位段被置“1”。并且只能通过向 TMRIC 寄存器写 1 清除本位段。 0：中断未产生 1：中断已产生

表 8-13：定时器中断使能寄存器（TMRIE）位段定义

TMRIE (Timer Interrupt Enable Register) Offset: 0x14 Default: 0x00000000 Access: TIMER -> TMRIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 8-14：定时器中断使能寄存器（TMRIE）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	RW	0x0	使能定时器中断 这个位段不影响 TMRRAWIF 以及 PWMSYNCI/ ADCSOC 信号的产生。 0：不发送中断信号给 CPU 1：当 TMRRAWIF=1 时向 CPU 发送中断信号

表 8-15：定时器中断强制寄存器（TMRIFRC）位段定义

TMRIFRC (Timer Interrupt Force Register) Offset: 0x18 Default: 0x00000000 Access: TIMER -> TMRIFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 8-16：定时器中断强制寄存器（TMRIFRC）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	W1S	0x0	定时器软件强制中断控制位 0：写 0 无效，读总是读回 0 1：写 1 强制置位 TMRRAWIF 标志位 本位段会自动清零。

表 8-17：定时器中断清除寄存器（TMRIC）位段定义

TMRIC (Timer Interrupt Clear Register) Offset: 0x1C Default: 0x00000000 Access: TIMER -> TMRIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 8-18：定时器中断清除寄存器（TMRIC）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	W1C	0x0	定时器中断清除 0：写 0 无效，读总是读回 0 1：写 1 清除 TMRRAWIF 和 TMRIF 标志位 本位段会自动清零。

9 看门狗定时器（WDT）

9.1 概述

看门狗定时器（WDT）可以在系统运行失败（由于软件错误）的情况下，重新恢复系统的控制，以增加应用的可靠性。WDT 在计数器计数到一个给定的超时数值时，可以产生系统复位或者一个中断。SPC2168 有两个 32 位看门狗定时器。WDT 的寄存器可由 CPU 通过 AHB 总线进行控制。

每一个看门狗都有如下特性：

- APB 总线寄存器接口
- 专属 WDT 时钟，参见图 3-12
- 32 位向下计数器
- 当发生计数超时事件时，可配置产生复位或者中断

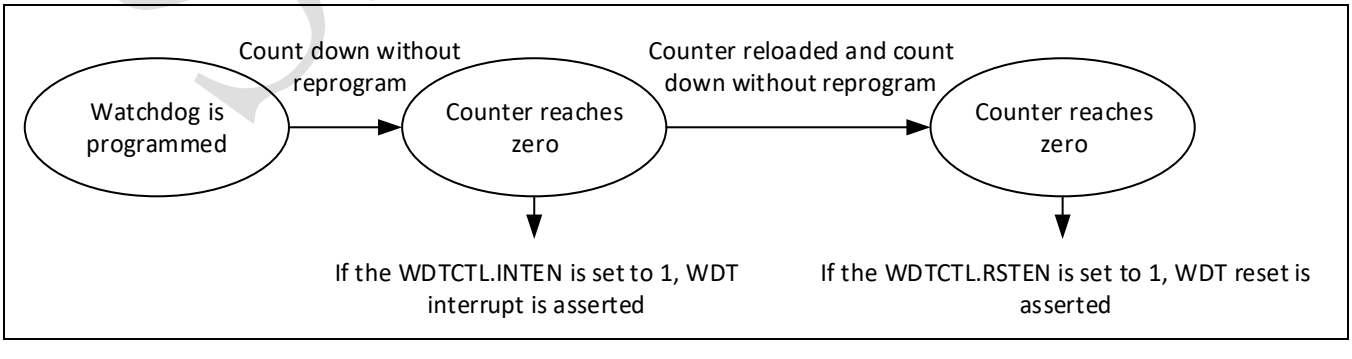
9.2 功能描述

看门狗计数器从一个预先设定的数值（超时值）向下计数到 0。这个超时值可以从 WDTLOAD 寄存器中获得。在系统复位之后和 WDT 使能之前，应当将超时值写入到 WDTLOAD 寄存器中。当计数器向下计数到 0 时，将会根据 WDTCTL 寄存器的设置产生系统复位或者中断。

出于安全方面的考虑，为了防止软件意外操作致使看门狗关闭，需要向 WDTREGKEY 寄存器写入 0x1ACCE551 后，才可以对看门狗其他寄存器进行写访问；当向 WDTREGKEY 寄存器写入任何其它数值后，对看门狗其他寄存器的写访问将无效。所以，在配置 WDT 之前，应该先使能其他寄存器的写访问。

图 9-1 是看门狗的操作流程框图。如果中断使能，WDT 的计数器也会使能，当超时发生时，会产生一个中断。然后，计数器从 WDTLOAD 寄存器重新加载数值，并继续向下计数的过程。如果在第二次超时发生时，上一次的中断没有被清除且系统复位功能被使能，那么 WDT 会产生一个系统复位请求。

图 9-1：看门狗工作流程



当向 WDTLOAD 写入新的数值时，计数器将会立即从新的数值重新开始计数。当向 WDTIC 写入任意数值时，中断会被清除且计数器重新开始计数。当这个重新开始计数与计数器计数到 0 发生在同一时刻时，这种情形下，WDT 不会产生中断。当 CPU 处于 HALTED 或者 LOCKUP 模式时，可以通过配置 WDTCTL 寄存器来关闭 WDT。

注意： WDT0 的中断请求信号被连接到 NMI 中断。

9.3 初始化流程

需要按照下面的操作流程来启动看门狗定时器：

- 配置 WDTREGKEY 寄存器，以使能其他寄存器的写访问
- 配置 WDTLOAD 寄存器，设置超时值
- 配置 WDTCTL.RSTEN 位段，使能复位功能
- 配置中断使能位，以启动计数器和使能中断
- 配置 WDTREGKEY 寄存器，关闭其他寄存器的写访问

9.4 寄存器

9.4.1 WDT 寄存器列表

表 9-1：WDT 模块基地址

外设模块	基地址
WDT0	0x4000 1000
WDT1	0x4000 2000

表 9-2：WDT 寄存器列表

寄存器	偏移地址	描述	复位值
WDTLOAD*	0x00	看门狗定时器超时加载寄存器	0x003D0900
WDTCNT	0x04	看门狗定时器计数器当前值寄存器	0xFFFFFFFF
WDTCTL*	0x08	看门狗定时器控制寄存器	0x0000000F
WDTIC*	0x0C	看门狗定时器中断清除寄存器	0x00000000
WDTRAWIF	0x10	看门狗定时器中断原始标志寄存器	0x00000000
WDTIF	0x14	看门狗定时器中断标志寄存器	0x00000000
WDTREGKEY	0x18	看门狗定时器写使能寄存器	0x00000000

注意：
1. 被*标记的寄存器只有在 WDTREGKEY=0x1ACCE551 时，才可以改写。
2. 如果 WDTREGKEY 的值不等于 0x1ACCE551，那么所有寄存器的读取值的 LSB 位都

为 1。

9.4.2 WDT 寄存器

表 9-3 到表 9-16 提供了 WDT 模块相关寄存器的详细信息。

表 9-3：看门狗定时器超时加载寄存器（WDTLOAD）位段定义

WDTLOAD (Watchdog Timer Load Register) Offset: 0x0 Default: 0x003D0900 Access: WDT -> WDTLOAD.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 9-4：看门狗定时器超时加载寄存器（WDTLOAD）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x3D0900	超时加载值 计数器从该加载值递减。当本寄存器被写入新的数值时，计数器将会立即从新的数值重新开始向下计数。 超时加载值的最小有效值为 1。

表 9-5：看门狗定时器计数器当前值寄存器（WDTCNT）位段定义

WDTCNT (Watchdog Timer Current Value Register) Offset: 0x4 Default: 0xFFFFFFFF Access: WDT -> WDTCNT.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 9-6：看门狗定时器计数器当前值寄存器（WDTCNT）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0xFFFFFFFF	向下计数的计数器当前值。

表 9-7：看门狗定时器控制寄存器（WDTCTL）位段定义

WDTCTL (Watchdog Timer Control Register) Offset: 0x8 Default: 0x0000000F Access: WDT -> WDTCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				LOCKUPRUN	HALTEDRUN	RSTEN	INTEN

表 9-8：看门狗定时器控制寄存器（WDTCTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	LOCKUPRUN	RW	0x1	在 CPU 内核处于 lockup 状态时，允许看门狗继续运行 0：在 CPU 内核处于 lockup 状态时，关闭看门狗 1：在 CPU 内核处于 lockup 状态时，使能看门狗
2	HALTEDRUN	RW	0x1	在 CPU 内核处于 halted 状态时，允许看门狗继续运行 0：在 CPU 内核处于 halted 状态时，关闭看门狗 1：在 CPU 内核处于 halted 状态时，使能看门狗
1	RSTEN	RW	0x1	使能看门狗系统复位请求 0：关闭系统复位请求 1：使能系统服务请求
0	INTEN	RW	0x1	使能中断 在本位段的上升沿，从 WDTLOAD 重载计数器的值。 0：关闭计数器和中断 1：使能计数器和中断

表 9-9：看门狗定时器中断清除寄存器（WDTIC）位段定义

WDTIC (Watchdog Timer Clear Interrupt Register) Offset: 0xC Default: 0x00000000 Access: WDT -> WDTIC.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 9-10：看门狗定时器中断清除寄存器（WDTIC）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	WO	0x0	中断清除寄存器 向本寄存器写入任意值都会清除看门狗中断，并且从 WDTLOAD 寄存器重新加载计数器的值。

表 9-11：看门狗定时器中断原始标志寄存器（WDTRAWIF）位段定义

WDTRAWIF (Watchdog Timer Raw Interrupt Status Register) Offset: 0x10 Default: 0x00000000 Access: WDT -> WDTRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							RAWINT

表 9-12：看门狗定时器中断原始标志寄存器（WDTRAWIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	RAWINT	RO	0x0	来自计数器的原始中断状态 这个寄存器表示来自计数器的原始中断状态。本位段的值会与 WDTCTL.INTEN 进行逻辑与操作，用于产生掩码后的中断信号，并传递到中断输出脚。 0：中断未发生 1：中断已发生

表 9-13：看门狗定时器中断标志寄存器（WDTIF）位段定义

WDTIF (Watchdog Timer Interrupt Status Register) Offset: 0x14 Default: 0x00000000 Access: WDT -> WDTIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							INT

表 9-14：看门狗定时器中断标志寄存器（WDTIF）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	INT	RO	0x0	来自计数器的掩码后的中断状态 这个寄存器表示来自计数器的掩码后的中断状态。本位段的值是 WDTRAWIF.RAWINT 和 WDTCTL.INTEN 的逻辑与，传递到中断输出脚的信号值是相同的。 0：中断未发生 1：中断已发生

表 9-15：看门狗定时器写使能寄存器（WDTREGKEY）位段定义

WDTREGKEY (Watchdog Timer Lock Register) Offset: 0x18 Default: 0x00000000 Access: WDT -> WDTREGKEY.all							
31	30	29	28	27	26	25	24
LCKCTL							
23	22	21	20	19	18	17	16
LCKCTL							
15	14	13	12	11	10	9	8
LCKCTL							
7	6	5	4	3	2	1	0
LCKCTL							LCKSTS

表 9-16：看门狗定时器写使能寄存器（WDTREGKEY）位段描述

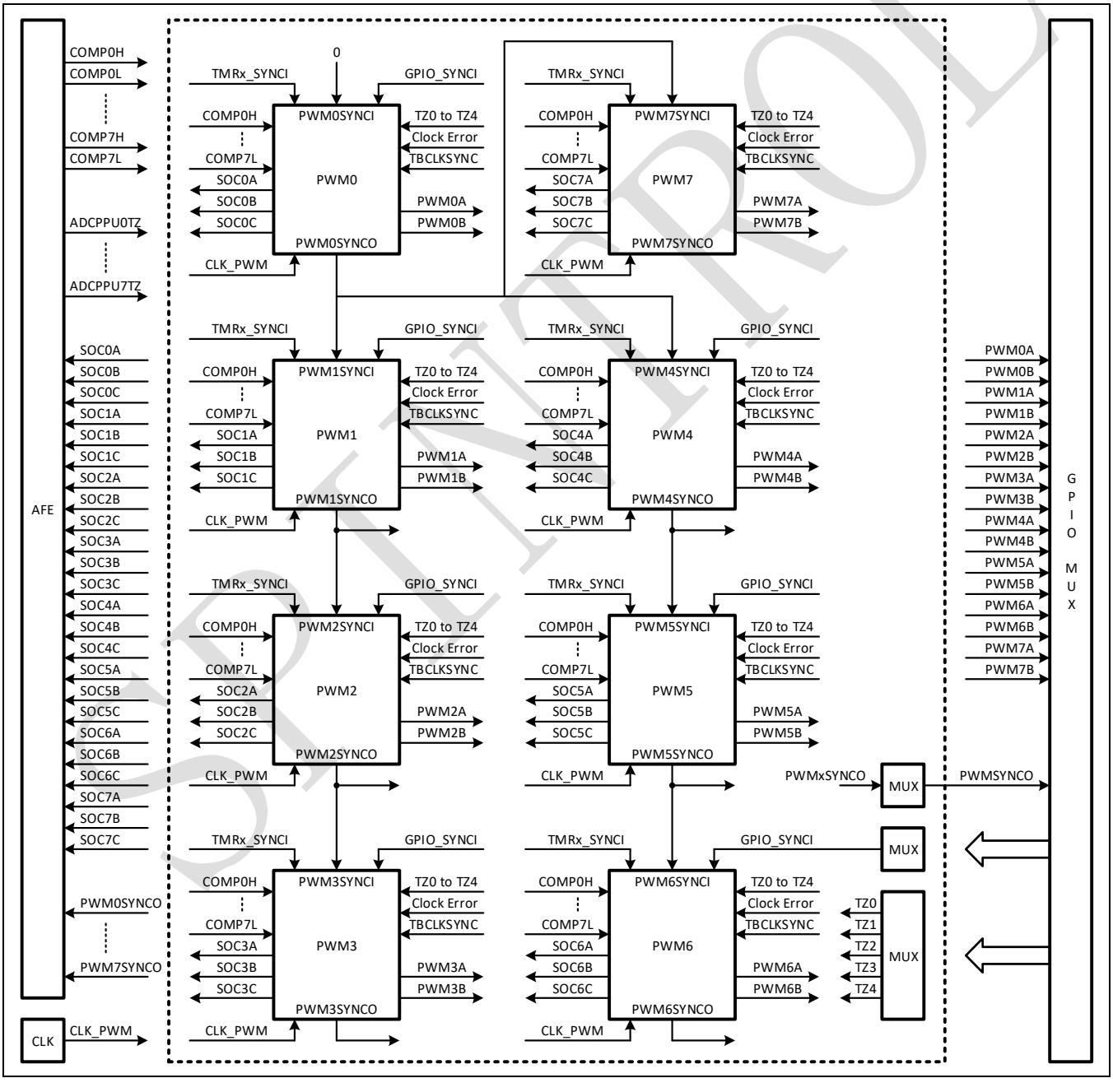
位段	位段名	属性	复位值	描述
31:1	LCKCTL	RW	0x0	使能寄存器写操作 向整个 WDTREGKEY 寄存器写入 0x1ACCE551，会使能其他寄存器的写访问；向整个 WDTREGKEY 寄存器写入其他值，会禁止对其他寄存器的写访问。
0	LCKSTS	RO	0x0	寄存器写访问锁定状态 0：对其他寄存器的写访问已使能（未锁定） 1：对其他寄存器的写访问已禁止（锁定）

10 PWM

10.1 PWM 概述

在诸如数控电机、开关电源、不间断电源等电力系统中，PWM 扮演了重要角色。SPC2168 提供了 8 个 PWM 模块，每个 PWM 模块有 PWMxA 和 PWMxB 两个输出。它们按照如图 10-1 所示方式串接在一起。

图 10-1：PWM 系统总览



每个 PWM 模块均支持如下特性：

- 周期可配置的 16-bit 时基计数
- 软件直接强制改变 PWM 输出
- 独立可配的相位控制
- 周期性的相位同步
- 上升沿延迟和下降沿延迟独立可配置的死区控制
- 异常事件下可配的周期性或者一次性封锁
- 封锁时可以配置 PWM 输出为高、低或高阻
- 数字比较或者封锁信号输入可产生原始封锁事件或者过滤后的封锁事件
- 所有事件可以用于产生 CPU 中断或者 ADC SOC（开始采样）信号

图 10-2：PWM 内部结构

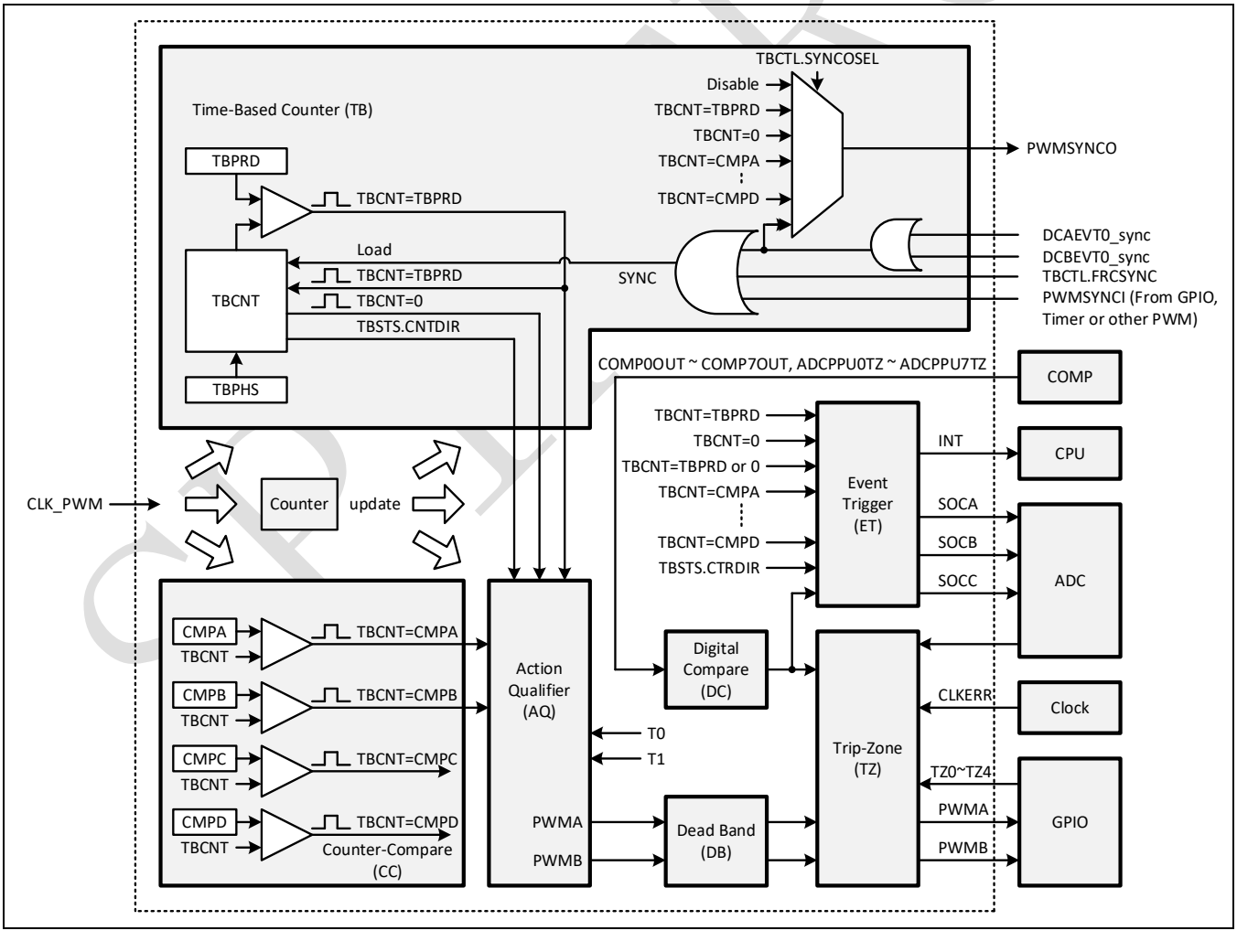


图 10-2 给出了单个 PWM 模块的内部结构，其关键信号包括：

- 同步信号（PWMSYNCI 和 PWMSYNCO）

用于实现多个 PWM 之间的相位同步。在 SPC2168 中，PWMSYNCI 可以来自于 GPIO 输入、通用定时器器或者是前级 PWM 的 PWMSYNCO 同步输出。

注意： 从同步信号输入到同步操作真正完成需要一个 PWM 时钟，因此当 TBCTL 寄存器的 TBDIVBIN 和 TBDIVLIN 位段同时为 0（即不分频）时，会有一拍时基时钟的延时。用户在设置 TBPHS 寄存器时需要考虑这个因素。
PWMCFG 模块中的 FRCSYNC 提供了更加灵活便捷的跨 PWM 同步方式。

- 封锁信号
  每个 PWM 有 5 个配置为来自 GPIO 的封锁信号（TZ0~TZ4）用于警示外部异常，另外来自 CLKDET 时钟交叉检测模块或者 PLL 的时钟错误事件、CPU 的 lockup 和 halted 事件也可以配置为 PWM 封锁信号。

注意： 来自 GPIO 的封锁信号极性可以配置，默认当 GPIO 为高时触发 PWM 封锁。

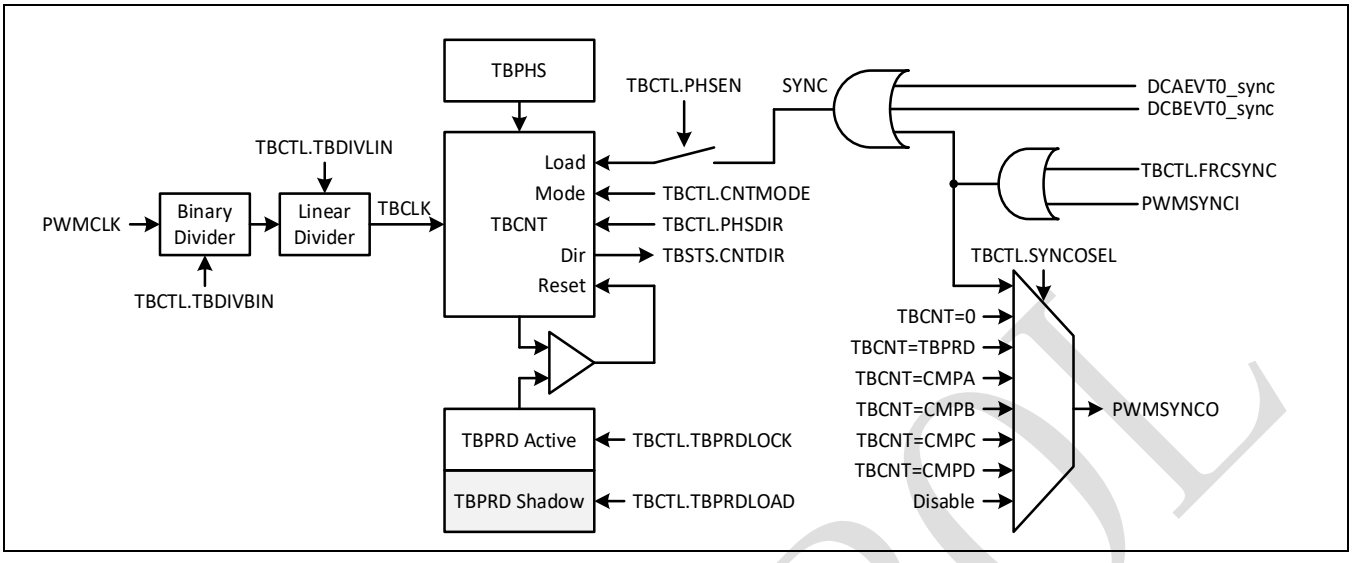
- 比较器结果
  来自 16 个外部比较器的结果可以产生数字比较事件。
- ADC SOC 信号（PWMxSOCA，PWMxSOCB 和 PWMxSOCC）
  每个 PWM 有 3 个 ADC SOC 信号。触发 ADC SOC 信号的事件是在事件触发（Event Trigger）子模块中配置的。
- PWM 输出（PWMxA 和 PWMxB）
  PWM 输出配置为与 GPIO 管脚复用，如表 4-1 所示。

10.2 时基产生（TB）子模块

时基产生（time-based）子模块负责 PWM 的时序控制，其关键信号如图 10-3 所示。该子模块有如下配置项：

- 设置时基计数时钟相对于 PWM 输入时钟的分频比
- 设置时基计数器（TBCNT）模式：
  - 向上计数
  - 向下计数
  - 上下计数
- 设置时基计数器（TBCNT）周期
- 设置相对于其他 PWM 的相位
- 产生 TBCNT=TBPRD 和 TBCNT=0 事件
- 设置同步触发事件以及事件发生时计数器采取的操作

图 10-3：PWM 时基产生子模块信号链



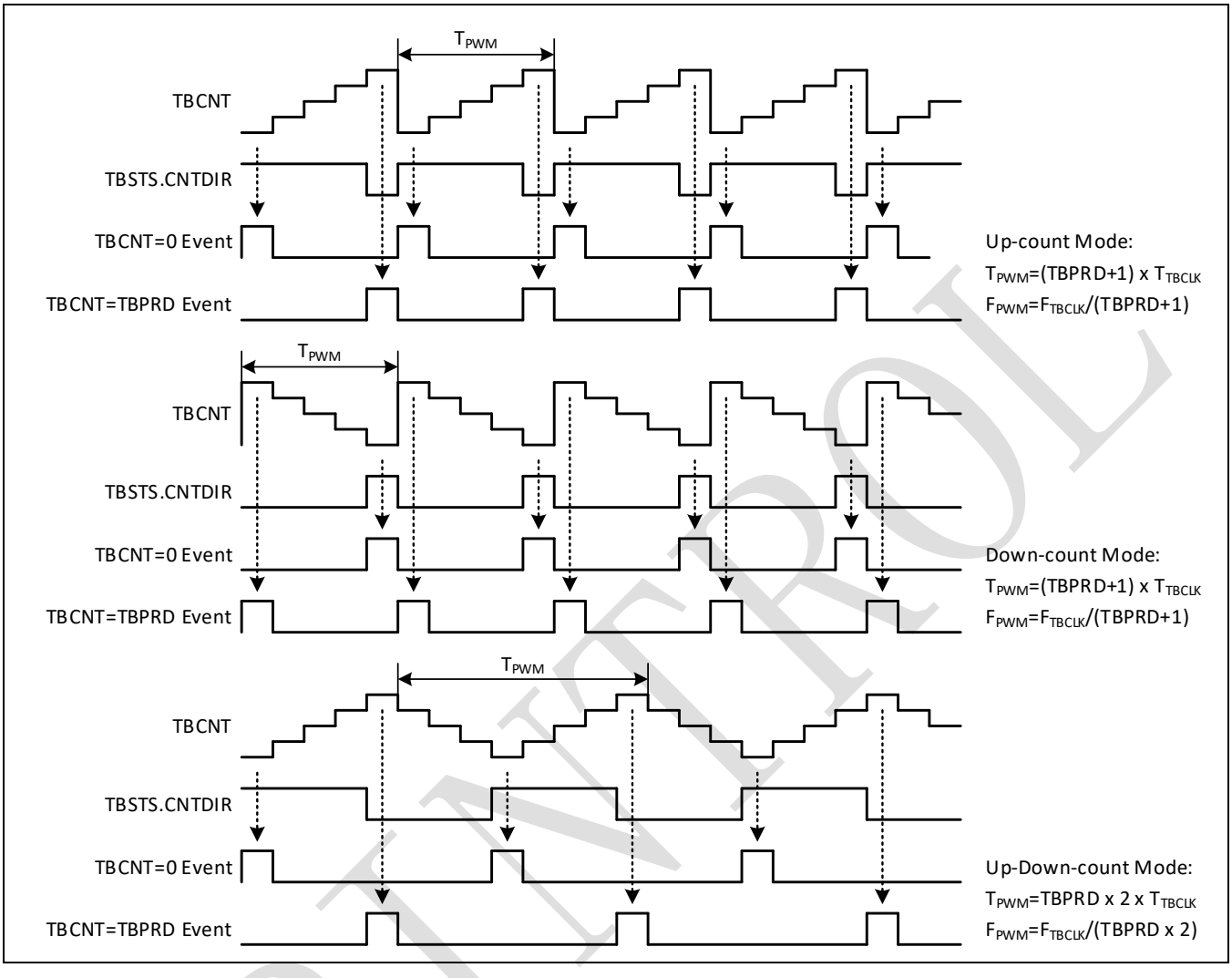
PWM 计数波形周期由计数模式（TBCTL.CNTMODE）和周期（TBPRD）寄存器共同决定。图 10-4 展示了向上计数、向下计数和上下计数三种情况下波形周期和 TBPRD 寄存器的关系。

向上计数模式时，时基计数器从 0 开始一直递增计数到 TBPRD，然后 TBCNT 复位到 0 重新开始递增计数。从 TBSTS 寄存器 CNTDIR 位段读回的值总是为 1。

向下计数模式时，时基计数器从 TBPRD 开始一直递减计数到 0，然后 TBCNT 复位到 TBPRD 重新开始递减计数。从 TBSTS 寄存器 CNTDIR 位段读回的值总是为 0。

上下计数模式时，时基计数器从 0 开始递增计数到 TBPRD，然后递减计数回 0，如此周而复始。从 TBSTS 寄存器 CNTDIR 位段读回的值，在从 0 到（TBPRD-1）递增计数时为 1，在从 TBPRD 到 1 递减计数时为 0。

图 10-4：时基计数波形

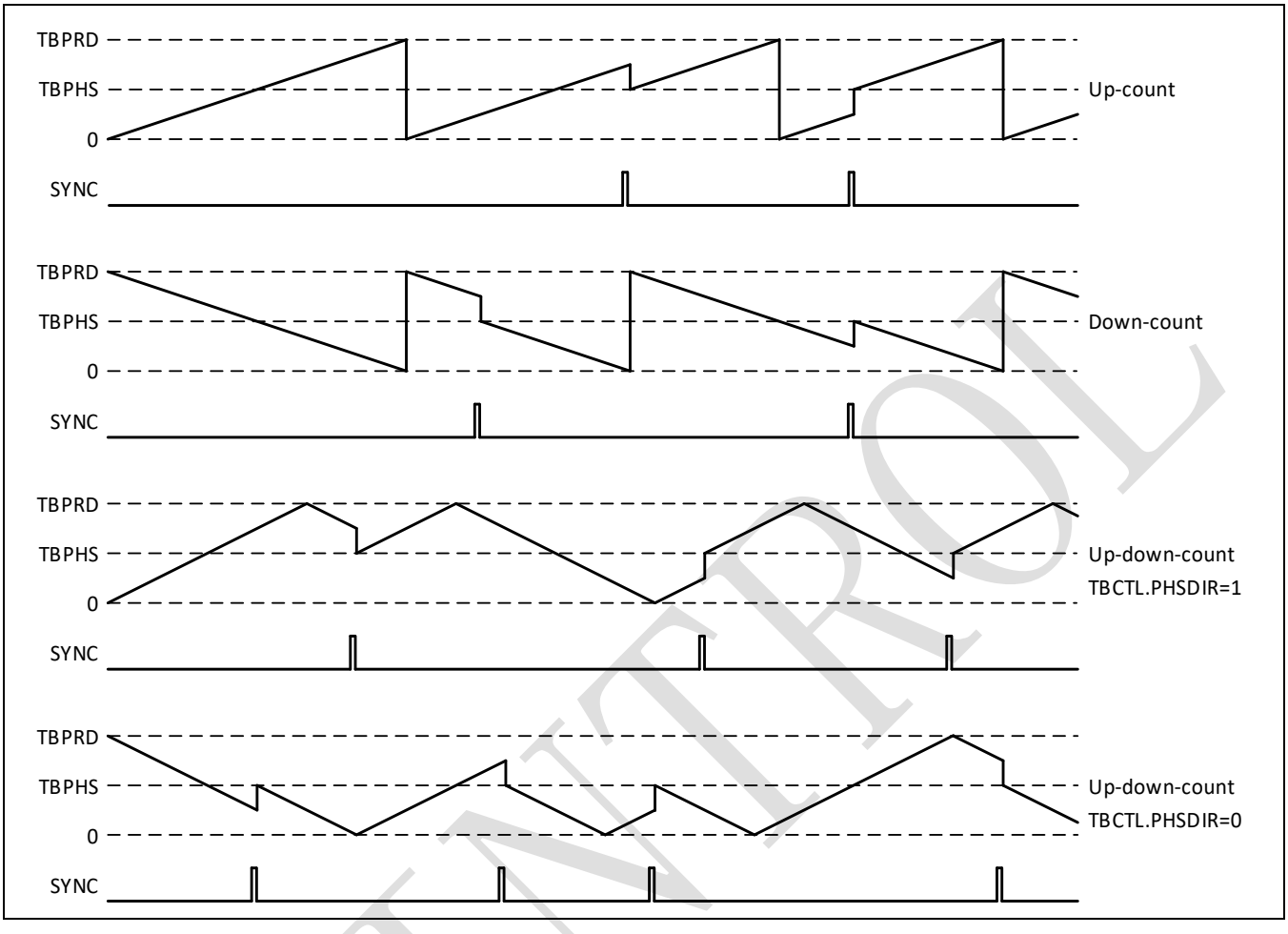


SPC2168 为 TBPRD 寄存器提供了影子寄存器。在该模式下，对 TBPRD 写操作只会先更新影子寄存器的值而不会立即更新实际值，然后根据 TBCTL.TBPRDLOAD 位段的设置，当 TBCNT=0 或者 TBCNT=TBPRD 时才从影子寄存器中更新。

相位同步功能使得 PWM 之间可以自动实现时基同步，可以通过设置 TBCTL.PHSEN 位段为 1 来开启这个功能。此时，一旦发生同步事件，TBCNT 会被更新为时基相位寄存器（TBPHS）的值。如图 10-5 所示，对于向上计数和向下计数模式，同步前后计数方向保持不变；对于上下计数方式，按照时基控制寄存器（TBCTL）的 PHSDIR 位段来决定同步后的计数方向。此外，如图 10-3 所示，同步事件可以是外部 PWMSYNCI 输入信号、数字比较同步信号 A 和 B、本地软件强制同步和全局软件强制同步。

当 TBCTL.PHSEN=0 时，PWM 忽略输入的同步事件信号，但这些信号仍然会被送往 PWMSYNCO 输出给后级 PWM 模块用于其同步。

图 10-5：时基相位同步波形

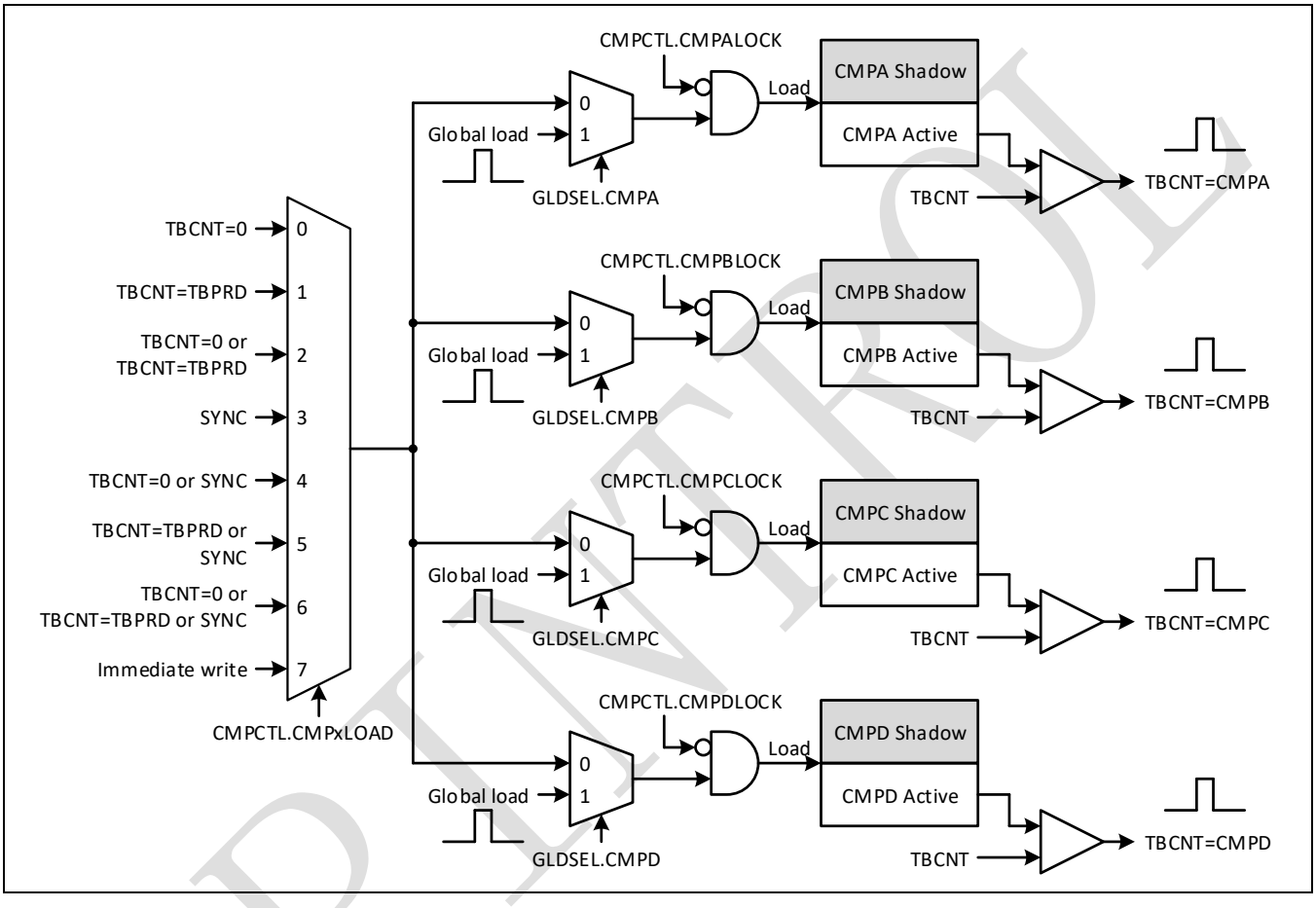


注意： 从同步信号输入到同步操作真正完成需要一个 PWM 时钟，因此当 TBCTL 寄存器的 TBDIVBIN 和 TBDIVLIN 位段同时为 0（即不分频）时，会有一拍时基时钟的延时。用户在设置 TBPHS 寄存器时需要考虑这个因素。
PWMCFG 模块中的 FRCSYNC 提供了更加灵活便捷的跨 PWM 同步方式。

10.3 计数比较（CC）子模块

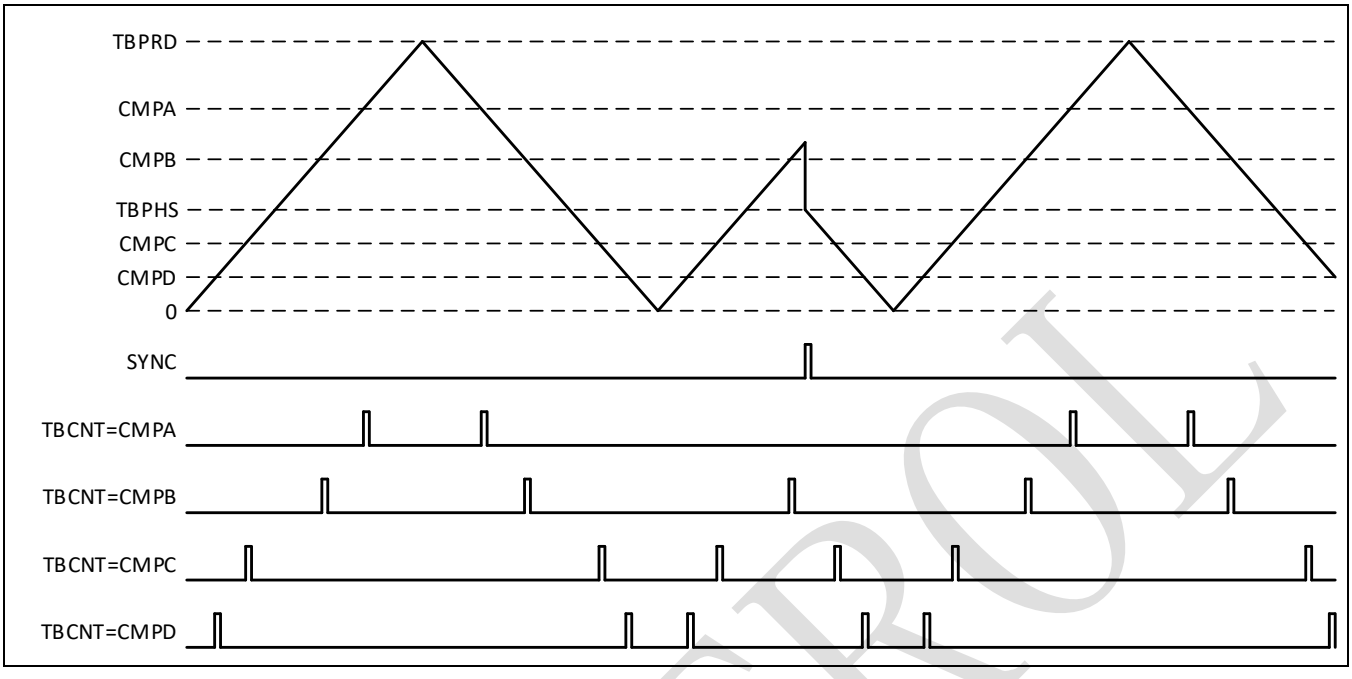
计数比较（counter-compare）子模块持续将 TBCNT 与计数比较寄存器 A（CMPA）、计数比较寄存器 B（CMPB）、计数比较寄存器 C（CMPC）以及计数比较寄存器 D（CMPD）的值进行比对。当 TBCNT 的值与这些寄存器的值相等时，产生相应的事件。

图 10-6：计数比较子模块结构



计数比较子模块的基本结构如图 10-6 所示。计数比较寄存器同样具备影子寄存功能，此时对计数比较寄存器的写操作会先更新影子寄存器的值，实际比较值会根据计数比较控制寄存器（CMPCTL）的 CMPxLOAD 位段定义的时间点更新。也可以按照表 10-14 和表 10-15 所示，通过全局影子到有效值更新时机控制寄存器（GLDSEL）的 CMPx 位段来配置影子功能。图 10-7 给出了计数比较子模块波形示例。

图 10-7：计数比较子模块波形



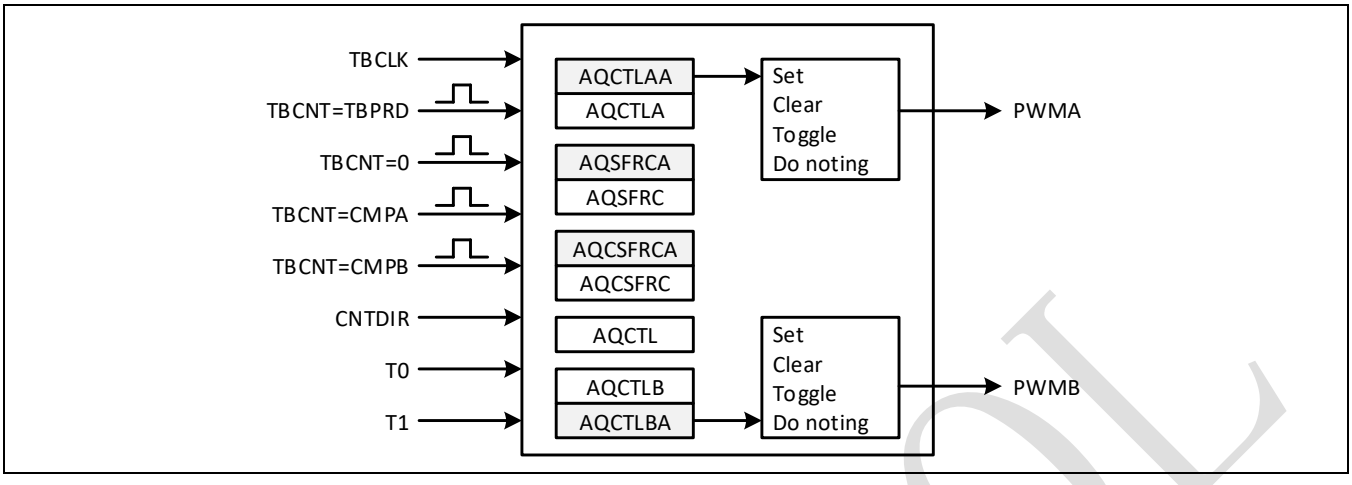
10.4 行为限定（AQ）子模块

行为限定（action-qualifier）子模块决定了哪些事件会转换为哪种行为并产生相应的 PWMA 和 PWMB 开关波形。

如图 10-8 所示，这些事件包括：

- TBCNT=TBPRD
- TBCNT=0
- TBCNT=CMPA
- TBCNT=CMPB
- T0/T1 事件（基于数字比较、封锁和同步输入事件产生）
- 软件强制事件

图 10-8：行为限定子模块输入输出



行为限定模块管理这些事件同时发生时的响应优先级，并基于 AQCTLA 和 AQCTLB 寄存器的配置在对应事件发生时做出如下操作：

- 输出置高
- 输出置低
- 输出翻转
- 不操作

可以通过 AQCTL.AQCTLALOAD、AQCTL.AQCTLBLOAD 和 AQSFRC.CSFLOAD 寄存器位段来选择影子寄存模式或是直接更新模式。也可以采用如图 10-6 所示 CMPx 寄存器类似的方法，通过 GLDSEL.AQCTLA、GLDSEL.AQCTLB 和 GLDSEL.AQCSFRC 寄存器位段配置全局影子寄存。

表 10-1 到表 10-3 列举了不同计数模式下的行为限定事件优先级。

表 10-1：向上计数时的行为限定事件优先级

优先级	事件
1（最高）	软件强制控制
2	TBCNT=TBPRD（PRD）
3	T0（T0U）
4	T1（T1U）
5	TBCNT=CMPB（CBU）
6	TBCNT=CMPA（CAU）
7（最低）	TBCNT=0（ZRO）

表 10-2：向下计数时的行为限定事件优先级

优先级	事件
1（最高）	软件强制控制
2	TBCNT=0（ZRO）
3	T0（T0D）
4	T1（T1D）
5	TBCNT=CMPB（CBD）
6	TBCNT=CMPA（CAD）
7（最低）	TBCNT=TBPRD（PRD）

表 10-3：上下计数时的行为限定事件优先级

优先级	事件	
	从 0 向上计数到 TBPRD-1 时	从 TBPRD 向下计数到 1 时
1（最高）	软件强制控制	软件强制控制
2	T0（T0U）	T0（T0D）
3	T1（T1U）	T1（T1D）
4	TBCNT=CMPB（CBU）	TBCNT=CMPB（CBD）
5	TBCNT=CMPA（CAU）	TBCNT=CMPA（CAD）
6（最低）	TBCNT=0（ZRO）	TBCNT=TBPRD（PRD）

图 10-9 和图 10-10 示例了行为限定模块产生的对称和非对称 PWM 波形。

图 10-9：上下计数时的 PWM 双边沿对称波形

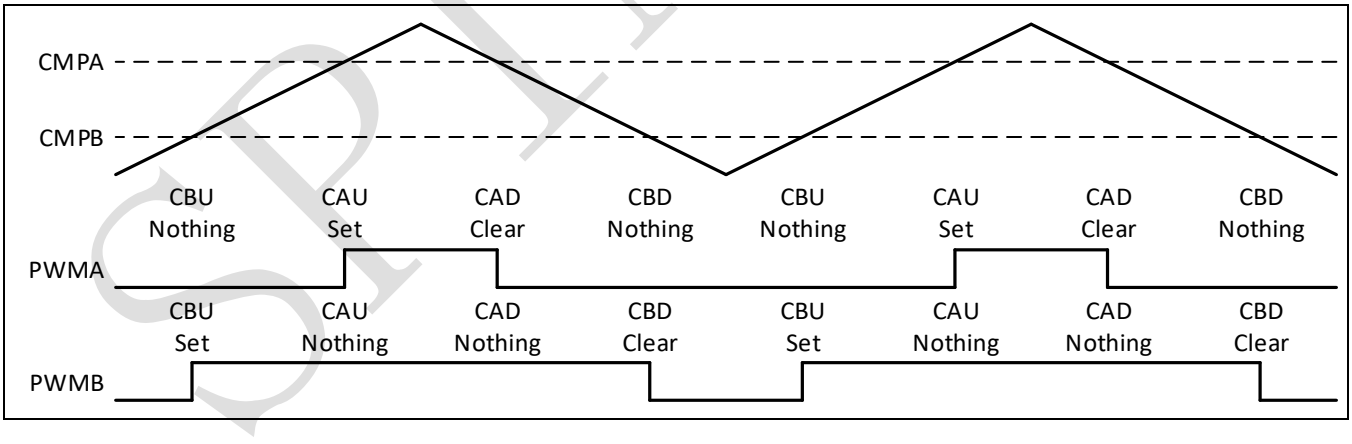
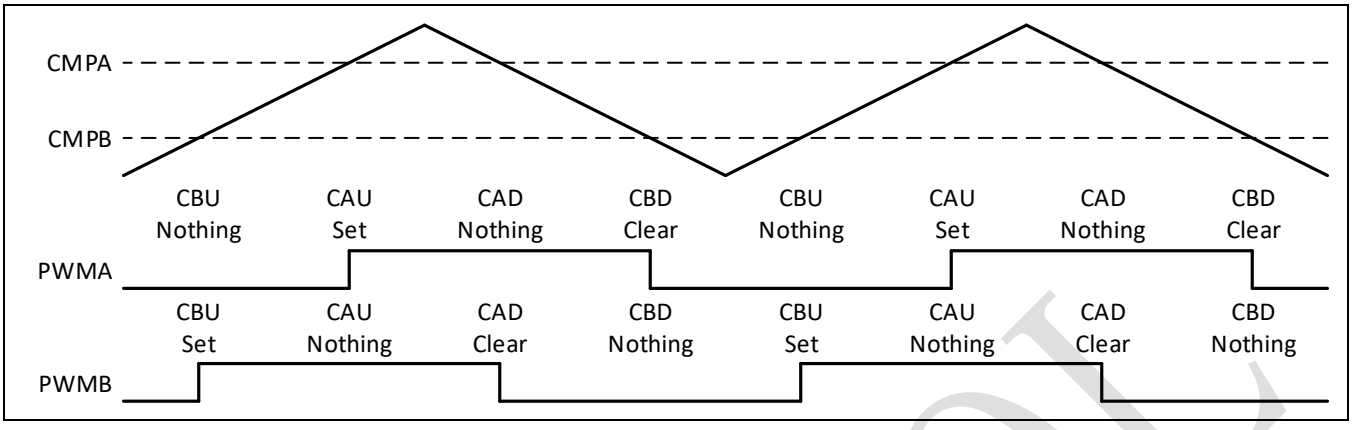


图 10-10：上下计数时的 PWM 双边沿非对称波形



10.5 死区控制（DB）子模块

死区（dead-band）控制子模块将行为限定模块输出的波形做进一步整形，包括对原始波形的上升沿和下降沿做独立的延时以及极性控制。

该模块的关键功能包括：

- 基于单路 PWMA 输入产生有相对死区的 PWMA 和 PWMB 信号对
- 对上升沿加可配置的延迟
- 对下降沿加可配置的延迟
- 支持半时钟周期延迟
- 调整输出信号极性
- 绕开死区控制信号整形，将行为限定模块输出的波形直接原样输出

图 10-11：死区控制子模块结构

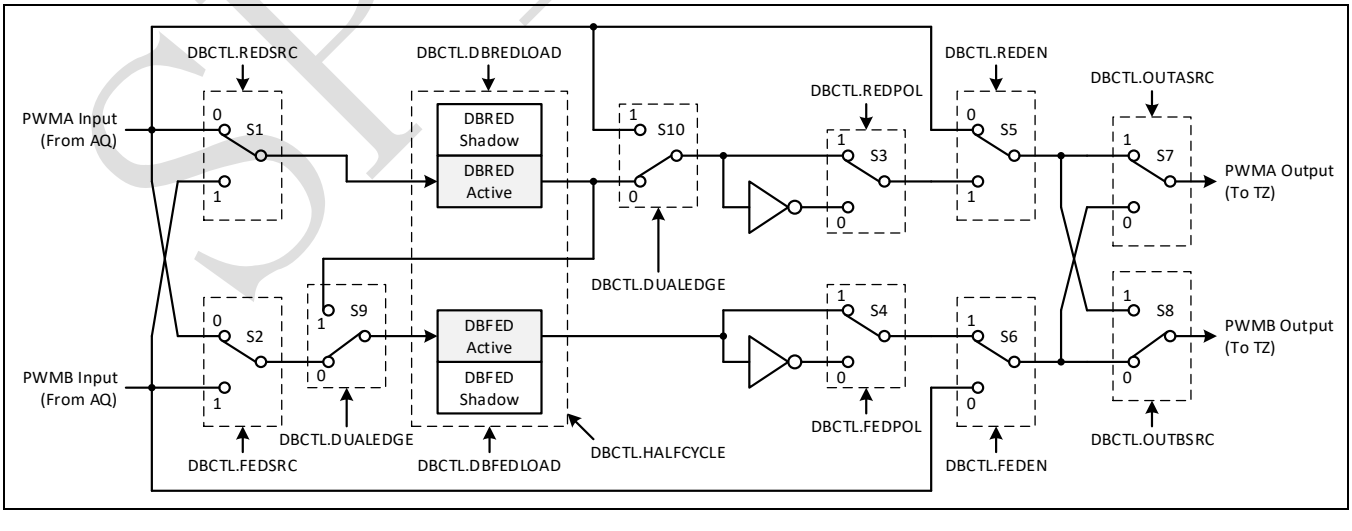


图 10-11 给出了死区控制模块的基本结构。死区控制寄存器（DBCTL）的 REDSRC 和 FEDSRC 位段设定了上升沿延迟通路和下降沿延迟通路的输入信号。随后 S3 到 S10 的各种开关组合可以产生灵活多变的波形。以 DBCTL.REDSRC=0 且 DBCTL.FEDSRC=0 为例，PWMA 被配置为两个延迟通路的输入，典型的死区控制工作模式如表 10-4 所示。

作为一种特殊情形，可以通过设置 DBCTL 寄存器的 DUALEDGE 位段实现双边沿延迟，即对同一路信号完成上升沿和下降沿的同时延迟。这个功能可以便于产生周期和占空比完全相同但存在指定相位差的两路波形。

表 10-4：典型的死区控制工作模式

模式	描述	REDPOL S3	FEDPOL S4	REDEN S5	FEDEN S6
1	同时将 PWMA 和 PWMB 原样输出	X	X	0	0
2	高有效的互补波形	1	0	1	1
3	低有效的互补波形	0	1	1	1
4	高有效的波形	0	0	1	1
5	低有效的波形	1	1	1	1
6	PWMA 输出 = PWMA 原始输入（无延迟） PWMB 输出 = PWMA 原始输入加下降沿延迟	0 或 1	0 或 1	0	1
7	PWMA 输出 = PWMA 原始输入加上升沿延迟 PWMB 输出 = PWMA 原始输入（无延迟）	0 或 1	0 或 1	1	0

图 10-12 示例了死区控制的典型波形。

图 10-12：死区控制典型波形

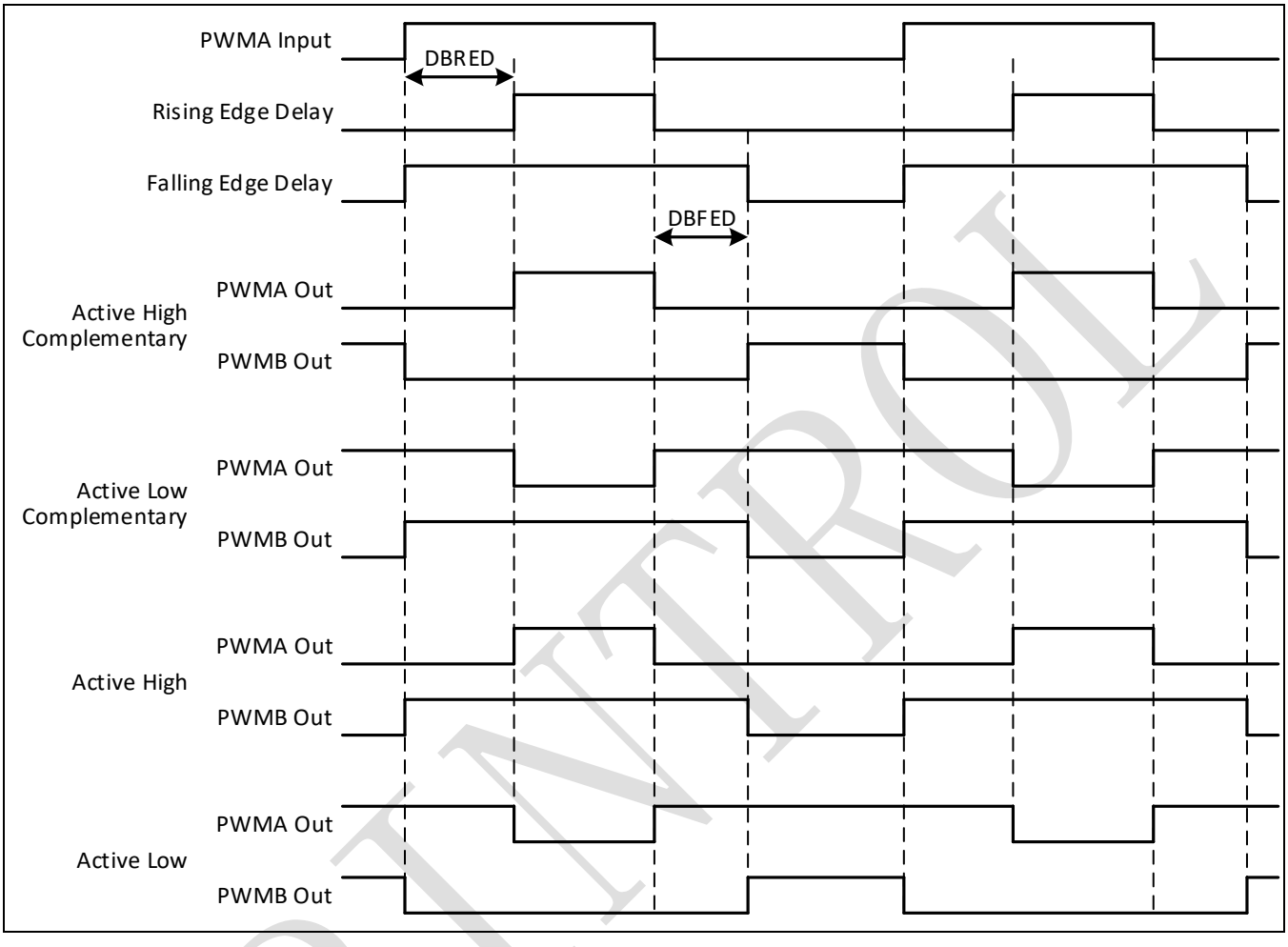
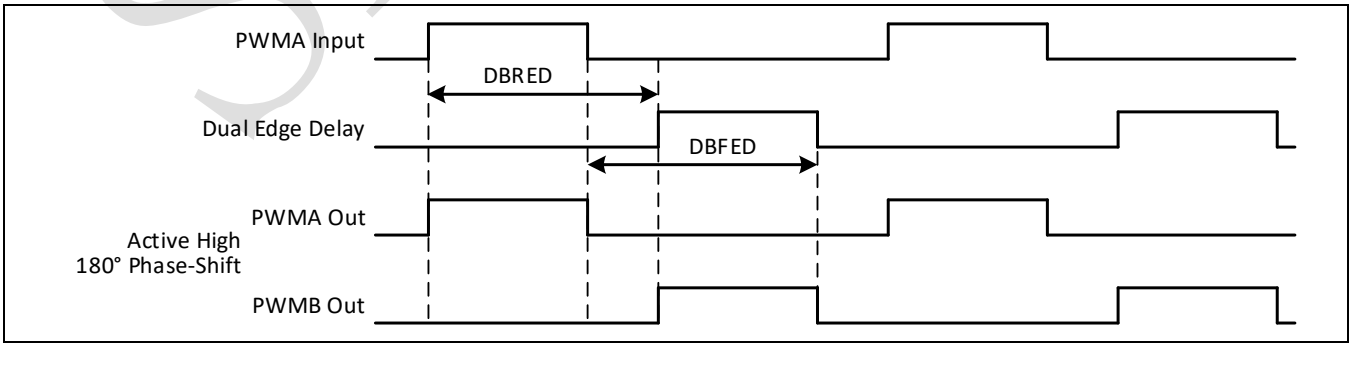


图 10-13 示例了通过使能双边沿延迟并设置 DBRED 和 DBFED 为波形周期的一半来产生 180° 相位差的两路波形。

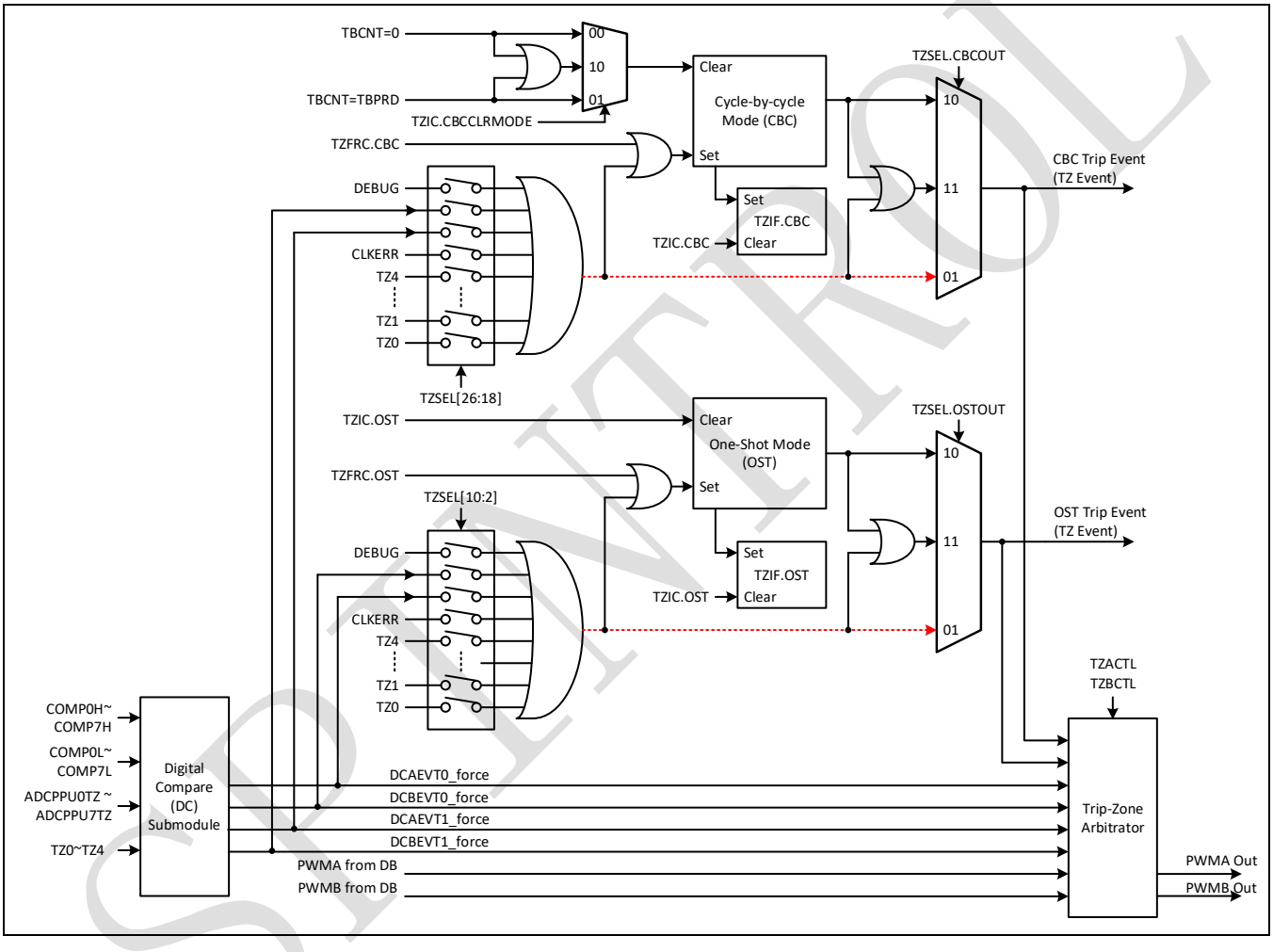
图 10-13：双边沿延迟死区控制波形



10.6 信号封锁（TZ）子模块

信号封锁（trip-zone）子模块处理当外部错误事件或封锁条件发生时的 PWM 输出。如图 10-14 所示，输入事件可以是来自 GPIO 的 TZ 信号、来自 PLL 失锁或者 CLKDET 时钟检测模块的时钟错误信号、CPU 调试信号、以及数字比较（DC）子模块的输出。可以根据 PWMCFG 模块中的 TZ#SRCCTL 寄存器选择 5 个 TZ 信号（TZ0~TZ4）具体对应的 GPIO 以及信号的极性。此外支持软件触发的封锁，并且可以配置各种封锁事件触发 CPU 中断。

图 10-14：信号封锁子模块结构



封锁事件发生时，PWMA 和 PMWB 输出可以被强制为如下几种情形：

- 置高
- 置低
- 高阻
- 维持原样不变

PWMA 输出的事件动作由寄存器 TZACTL 控制，这些事件动作的优先级为 TZU > TZD > DCAEVT0U > DCAEVT0D > DCAEVT1U > DCAEVT1D。这意味着当高优先级事件和低优先级事件同时

发生时，输出由高优先级事件的配置决定。例如，当 TZU 和 DCAEVT1U 事件同时发生时，TZU 寄存器字段决定 PWM 输出，而 DCAEVT1U 字段被忽略。

SPC2168 的 PWM 同时支持一次性封锁（OST）和周期性（CBC）封锁。一次性封锁通常用于诸如严重过流或者电路短路等重大异常，周期性封锁常用于限流。

当一次性封锁发生时，PWMA 和 PWMB 的输出会立刻按照 TZACTL 和 TZBCTL 寄存器相应位段的设置改变。同时，TZIF.OST 标志位置 1。如果用户通过封锁中断使能寄存器（TZIE）开启了中断，则同时会产生 PWM_TZINT 中断。TZIF.OST 标志位只能通过往 TZIC.OST 位段写 1 手动清除。

当周期性封锁发生时，PWMA 和 PWMB 的输出会立刻按照 TZACTL 和 TZBCTL 寄存器相应位段的设置改变。同时，TZIF.CBC 标志位置 1。如果用户通过封锁中断使能寄存器（TZIE）开启了中断，则同时会产生 PWM_TZINT 中断。根据 TZIC.CBCCLRPUL 位段的设置，PWM 会在 TBCNT 计数到 0 或者 TBPRD 的时候检查封锁条件是否已不存在，若封锁条件已不存在则自动撤销对输出的强制。但是，TZIF.CBC 标志位依然需要通过往 TZIC.CBC 位段写 1 来手动清除。如果清除 TZIF.CBC 标志位的时候封锁条件依然存在，则该标志位会立即又被置位。

注意： 图 10-14 中虚线所示的异步逻辑通路主要用于观测原始封锁事件信号（相应的 TZSEL.OSTOUT 或 TZSEL.CBCOUT 设定为 1）。实际应用中，必须选中锁存的通路来使能上述一次性和周期性封锁功能，即 TZSEL.OSTOUT 和 TZSEL.CBCOUT 必须设定为 2 或 3。

根据数字比较封锁事件选择寄存器（TZDCSEL）中 DCAH/DCAL 和 DCBH/DCBL 位段的设置，会产生数字比较事件 DCAEVT0/1 和 DCBEVT0/1。当数字比较事件发生时，PWMA 和 PWMB 的输出会立刻按照 TZACTL 和 TZBCTL 寄存器相应位段的设置改变。同时，TZIF.DCAEVT0/1 或 TZIF.DCBEVT0/1 标志位置 1。如果用户通过封锁中断使能寄存器（TZIE）开启了中断，则同时会产生 PWM_TZINT 中断。若数字比较事件消除，则自动撤销对输出的强制。但是，TZIF 寄存器中的相应标志位依然需要通过往 TZIC 相应位段写 1 来手动清除。如果清除标志位时数字比较事件依然存在，则标志位会立即又被置位。

图 10-15 示例了信号封锁中断产生逻辑。

图 10-15：信号封锁中断产生

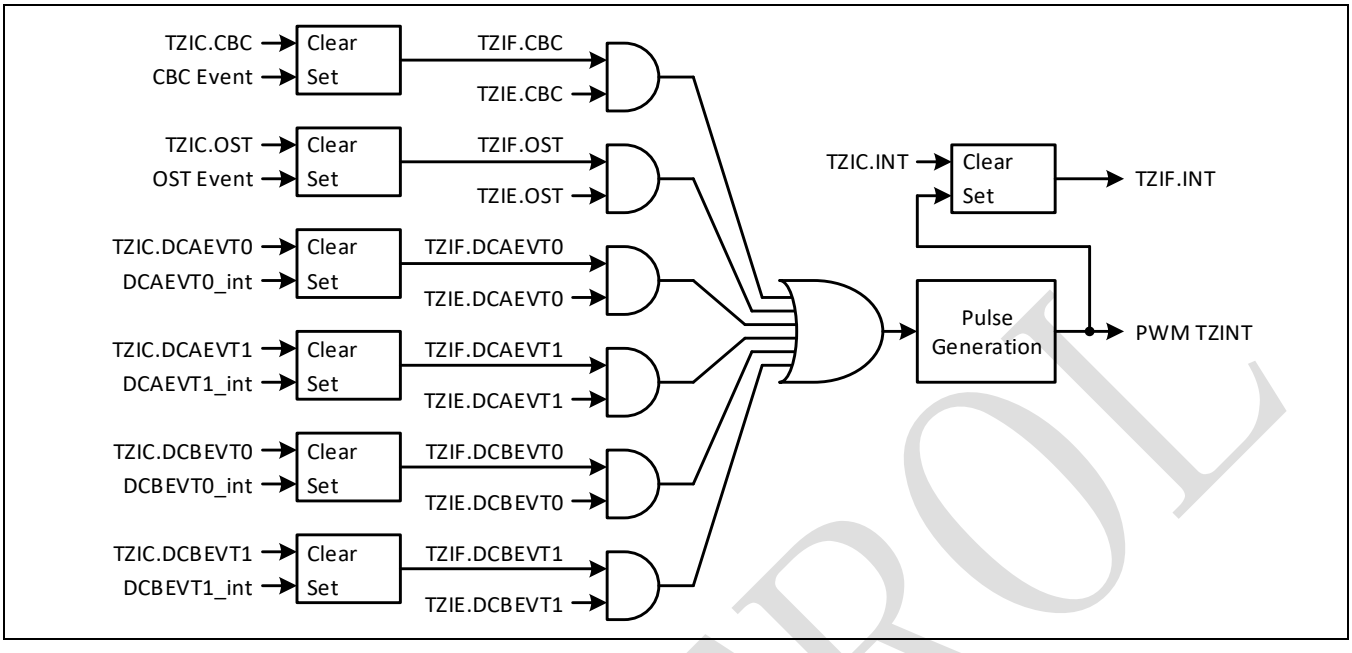


表 10-5 列举了影响 PWM 最终输出波形的各个事件的优先级。

表 10-5：影响 PWM 最终输出波形的事件优先级

优先级	事件
1（最高）	封锁（trip-zone）事件
2	通过 AQCSFRC 寄存器的软件持续强制
3	死区（Dead-Band）控制逻辑
4	通过 AQSFRC 寄存器的软件单次强制
5（最低）	行为限定（Action-Qualifier）控制逻辑

10.7 数字比较（DC）子模块

如图 10-16 所示，数字比较（digital compare）模块将来自 GPIO 的 TZ0~TZ4 事件、来自比较器（COMP）的 COMPxL 和 COMPxH 事件以及来自 ADC 后处理单元的 ADCPPU0TZ~ADCPPU7TZ 事件作为输入，根据用户配置产生原始的 DCAL、DCAH、DCBL、DCBH 信号，再从这四个信号的各种组合中产生原始的 DCAEVT0、DCAEVT1、DCBEVT0、DCBEVT1 事件。从中又可以选择一个事件，在一个时间窗内加以屏蔽，从而实现对事件噪声的过滤和消隐。具体如图 10-17 所示：在 TBCNT=TBPRD 或者 TBCNT=0 时，位移计数器根据 DCFOFFSET 寄存器重置计数值并开始递减计数，待计数到 0 时开启消隐窗口并保持 DCFWINDOW 寄存器所指定的时基周期。在消隐窗口这段时间内的事件会被忽略，而在消隐窗口前后的事件可以正常用于产生 SOC、SYNC 等信号。考虑到实际应用中模块间的串扰，用户可以使能其他 PWM 模块产生的消隐窗口来对本级原始数字比较事件加以屏蔽。DCFCTL 寄存器的 BLANKINV 位段可以实现对消隐窗口的反相。此外如图 10-18 所示，一旦位移计数到 0，即便上一个消隐窗口尚未结束，新的消隐窗口仍将开始。

图 10-16：数字比较子模块结构

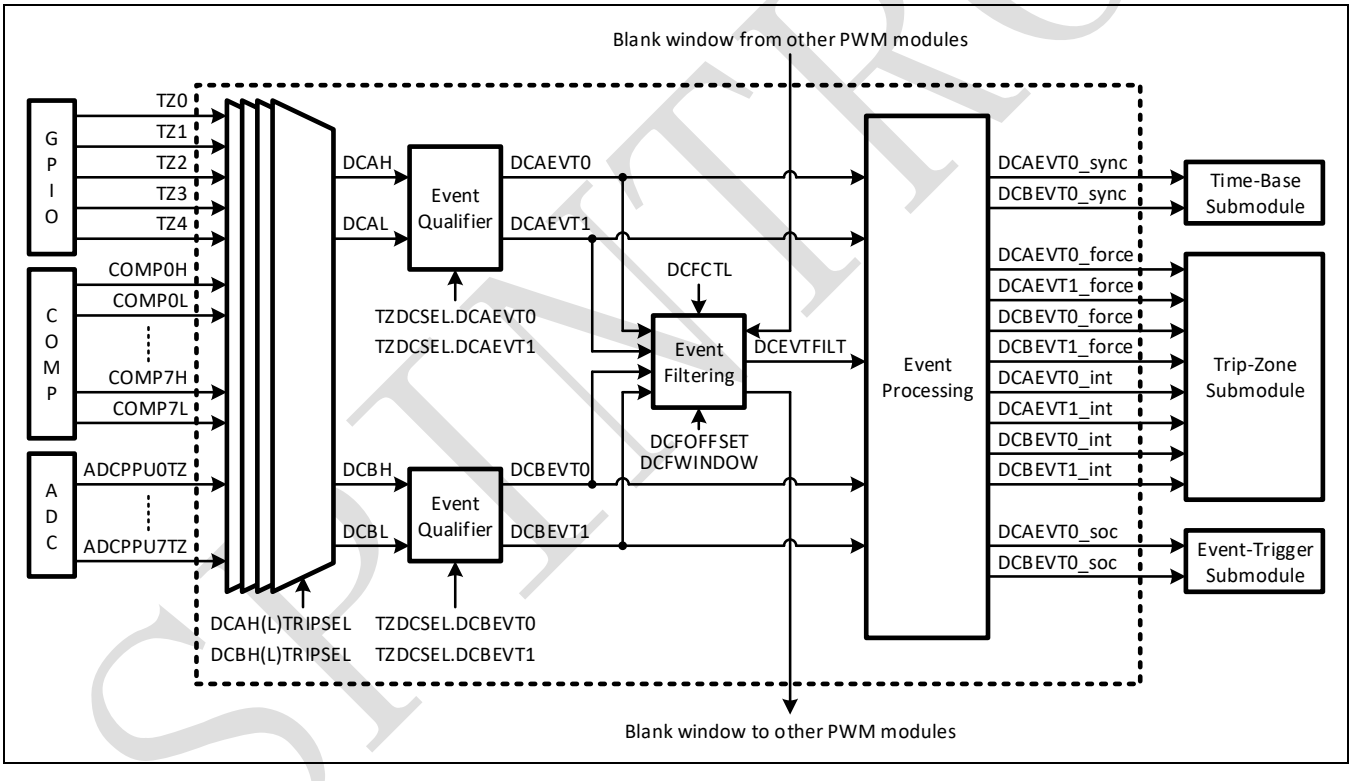


图 10-17：事件过滤逻辑

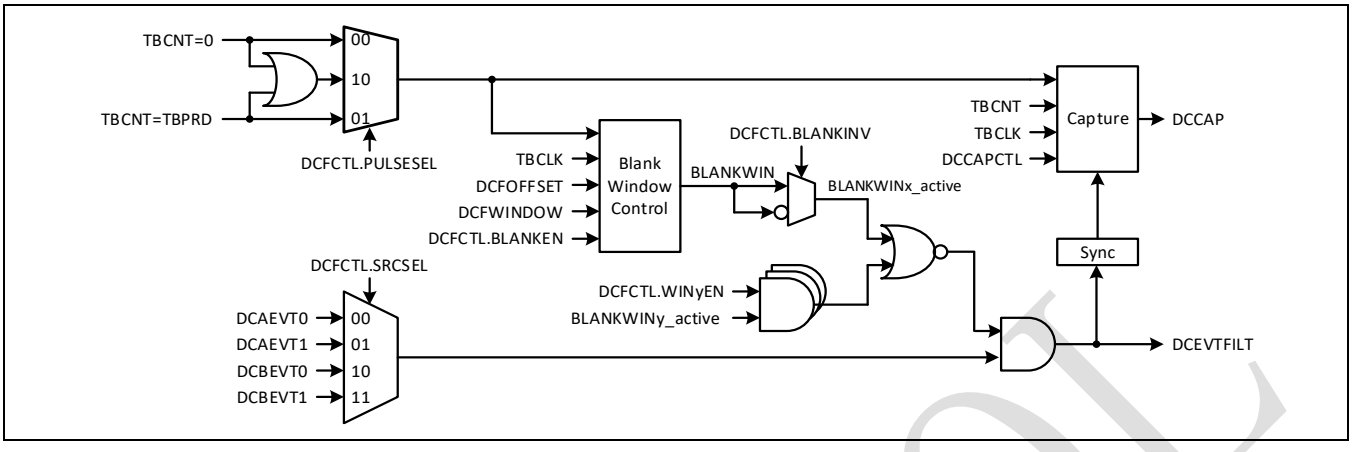
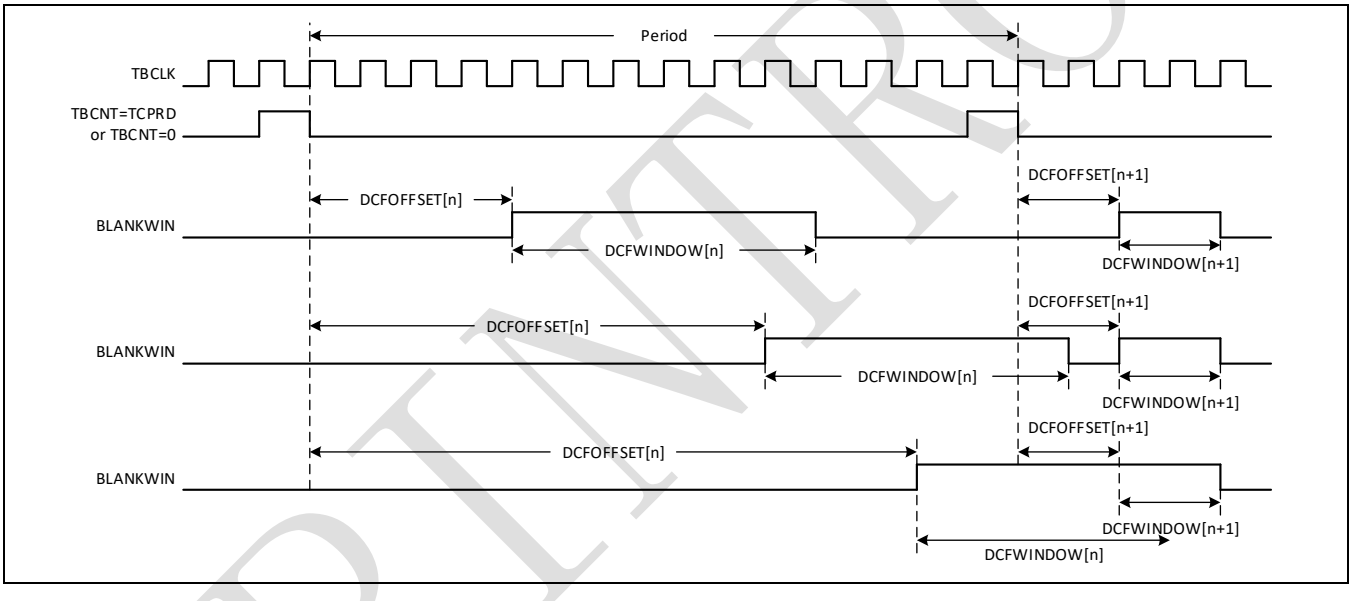


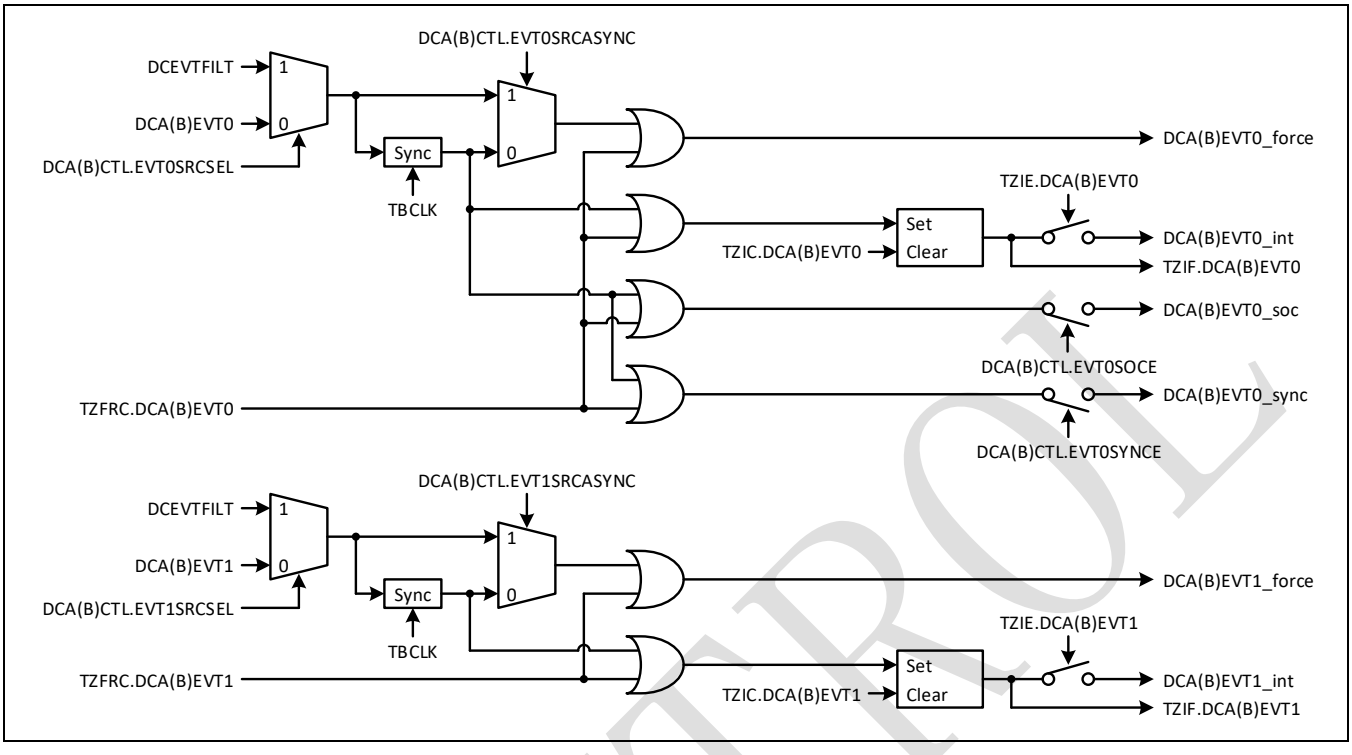
图 10-18：消隐窗口时序图



如图 10-19 所示，原始 DCAEVT0、DCAEVT1、DCBEVT0、DCBEVT1 信号和过滤得到的 DCEVTFILT 信号送至事件处理模块，根据 DCACTL 和 DCBCTL 寄存器的 EVT0SRCSEL 和 EVT1SRCSEL 位段，产生最终的 DCAEVT0、 DCAEVT1、DCBEVT0 和 DCBEVT1 事件。基于这些事件，可以进一步产生如下信号：

- 送往信号封锁子模块的强制信号
  DCAEVT0_force、DCAEVT1_force、DCBEVT0_force、DCBEVT1_force
- 送往时基产生子模块的同步信号
  DCAEVT0_sync、DCBEVT0_sync
- ADC 采样开始信号
  DCAEVT0_soc、DCBEVT0_soc
- 中断
  DCAEVT0_int、DCAEVT1_int、DCBEVT0_int、DCBEVT1_int

图 10-19：事件处理



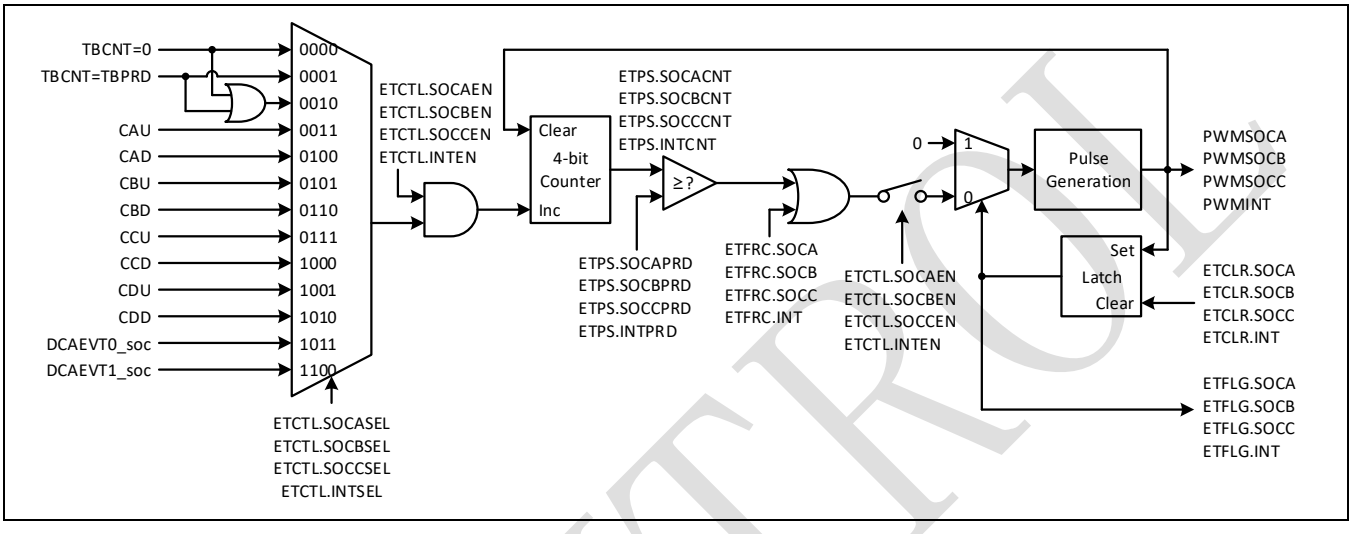
下面给出一个用 COMP0~COMP2 做相电流监测并产生消隐后的数字比较封锁信号来控制 PWMxA 和 PWMxB 输出的例子。

- 通过 DCAHTRIPSEL 寄存器设定 DCAH 事件为 COMP0H、COMP0L、COMP1H、COMP1L、COMP2H 和 COMP2L 信号的逻辑或
- 设定 TZDCSEL.DCAEVT0=4 来选择 DCAH=1 触发原始 DCAEVT0 事件
- 设定 DCFCTL.SRCSEL=0 来选择 DCAEVT0 作为滤波输入
- 设定 DCACTL.EVT0SRCSEL=1，选择滤波后的 DCEVTFILT 作为最终 DCAEVT0
- 设定 DCBCTL.EVT0SRCSEL=1，选择滤波后的 DCEVTFILT 作为最终 DCBEVT0
- 通过 TZACTL 寄存器的 DCAEVT0U 和 DCAEVT0D 位段设定 DCAEVT0 发生时 PWMxA 的输出
- 通过 TZBCTL 寄存器的 DCBEVT0U 和 DCBEVT0D 位段设定 DCBEVT0 发生时 PWMxB 的输出

10.8 事件触发（ET）子模块

事件触发（event-trigger）子模块管理时基产生子模块、计数比较子模块和数字比较子模块产生的各种事件，并根据配置在对应事件发生时产生送至 CPU 的中断请求脉冲或送至 ADC 的采样起始脉冲。

图 10-20：事件触发子模块结构



事件触发子模块的具体结构如图 10-20 所示。为了便于软件调试，除前述硬件信号外，SPC2168 也支持由软件强制触发的 SOCA、SOCB、SOCC 和 INT 脉冲。基于 4 位的计数器，脉冲产生可以有如下场景：

- 从不产生
- 每次事件到来均产生脉冲
- 每两次事件到来产生一个脉冲
- ……
- 每十五次事件到来产生一个脉冲

10.9 寄存器的全局重置

如图 10-21 所示，可以设置在全局重置脉冲发生时将所有影子寄存器的值更新到对应的寄存器中。通过将 GLDCTL0 寄存器的 GLDEN 位段置 1 可以开启这个功能。GLDSEL 寄存器中的各位段指定了每个带影子寄存功能的寄存器是在全局重置脉冲发生时还是在本地重置脉冲发生时根据影子寄存器更新有效值。

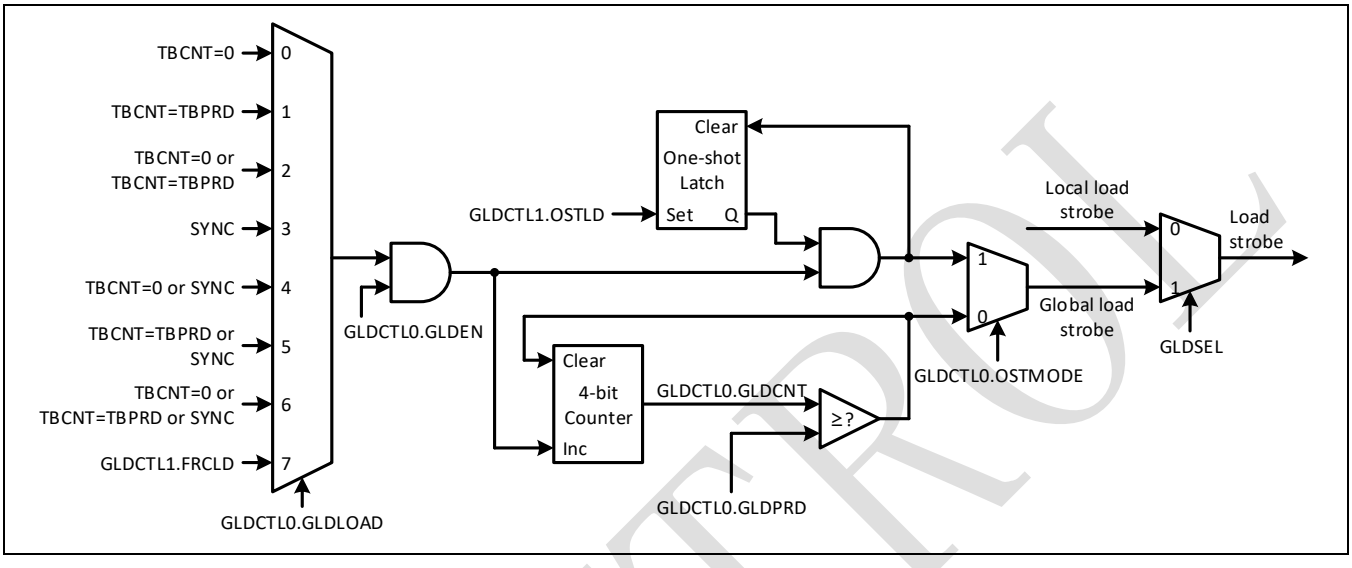
如果 GLDCTL0.OSTMODE=0，则为这种全局重置引入预分频功能，根据 GLDCTL0.GLDPRD 位段的值，每当发生相应数目的重置脉冲后才会真正做一次寄存器有效值的重置。

如果 GLDCTL0.OSTMODE=1，则开启单次重置功能。此时仅有 GLDCTL1.OSTLD 置 1 后的第一个全局重置脉冲会触发寄存器有效值的重置。该脉冲还会将 GLDCTL1.OSTLD 硬件清 0，直到软件

重新置 1 以使能下一个脉冲触发寄存器重置。在 GLDCTL1.OSTLD=0 期间，全局重置脉冲将被忽略，不会触发寄存器有效值的重置。

用户还可以通过将 GLDCTL1.FRCLD 位段置 1 来软件触发全局重置。

图 10-21：影子寄存器全局读入



10.10 寄存器同时写入

在某些应用中，用户期望能对不同 PWM 模块中相同名字的寄存器同时写入相同的值。SPC2168 提供了 PWMLINK 寄存器，由此实现对其他 PWM 模块中某个寄存器进行写操作的时候会同时更新本 PWM 模块中对应的寄存器。支持该特性的寄存器包括：TBPRD、CMPA、CMPB、CMPC、CMPD、DBRED、DBFED 和 GLDCTL1。

10.11 寄存器

10.11.1 PWM 寄存器列表

表 10-6：PWM 模块基地址

外设模块	基地址
PWM0	0x4000 9000
PWM1	0x4000 9100
PWM2	0x4000 9200
PWM3	0x4000 9300
PWM4	0x4000 9400
PWM5	0x4000 9500
PWM6	0x4000 9600
PWM7	0x4000 9700

表 10-7：PWM 寄存器列表

寄存器	偏移地址	描述	复位值
SHADOWSTS	0x00	影子状态寄存器	0x00000000
GLDCTL0*	0x04	全局影子值重置有效值控制寄存器 0	0x00000000
GLDCTL1	0x08	全局影子值重置有效值控制寄存器 1	0x00000000
GLDSEL*	0x0C	全局影子值重置有效值控制选择寄存器	0x00000000
PWMLINK*	0x10	PWM 链接寄存器	0xFFFFFFFF
TBCTL*	0x14	时基控制寄存器	0x00000006
TBPRD	0x18	时基周期寄存器	0x00000000
TBPRDA	0x1C	时基周期有效值寄存器	0x00000000
TBPHS	0x20	时基相位寄存器	0x00000000
TBCNT	0x24	时基计数寄存器	0x00000000
TBSTS	0x28	时基状态寄存器	0x00000100
TBSTSCLR	0x2C	时基状态清除寄存器	0x00000000
CMPCTL*	0x30	计数比较控制寄存器	0x00000000
CMPA	0x34	计数比较阈值 A 寄存器	0x00000000
CMPAA	0x38	计数比较阈值 A 有效值寄存器	0x00000000
CMPB	0x3C	计数比较阈值 B 寄存器	0x00000000
CMPBA	0x40	计数比较阈值 B 有效值寄存器	0x00000000
CMPC	0x44	计数比较阈值 C 寄存器	0x00000000
CMPCA	0x48	计数比较阈值 C 有效值寄存器	0x00000000
CMPD	0x4C	计数比较阈值 D 寄存器	0x00000000
CMPDA	0x50	计数比较阈值 D 有效值寄存器	0x00000000

寄存器	偏移地址	描述	复位值
AQCTL*	0x54	行为限定控制寄存器	0x00000000
AQCTLA	0x58	行为限定 A 路输出控制寄存器	0x00000000
AQCTLAA	0x5C	行为限定 A 路输出控制有效值寄存器	0x00000000
AQCTLB	0x60	行为限定 B 路输出控制寄存器	0x00000000
AQCTLBA	0x64	行为限定 B 路输出控制有效值寄存器	0x00000000
AQSFRC	0x68	行为限定软件强制寄存器	0x00000000
AQCSFRC	0x6C	行为限定软件持续强制寄存器	0x00000000
AQCSFRCA	0x70	行为限定软件持续强制有效值寄存器	0x00000000
DBCTL*	0x74	死区产生控制寄存器	0x00000070
DBCTLA	0x78	死区产生控制有效值寄存器	0x00000070
DBRED	0x7C	死区产生上升沿延迟量寄存器	0x00000000
DBREDA	0x80	死区产生上升沿延迟量有效值寄存器	0x00000000
DBFED	0x84	死区产生下降沿延迟量寄存器	0x00000000
DBFEDA	0x88	死区产生下降沿延迟量有效值寄存器	0x00000000
TZSEL*	0x8C	封锁事件选择寄存器	0x00030003
TZSTS	0x90	封锁状态寄存器	0x00000000
TZSTSCLR	0x94	封锁状态清除寄存器	0x00000000
TZDCSEL*	0x98	数字比较封锁事件寄存器	0x00000000
TZACTL	0x9C	封锁 A 路输出控制寄存器	0x00000000
TZBCTL	0xA0	封锁 B 路输出控制寄存器	0x00000000
TZIF	0xA4	封锁中断标志寄存器	0x00000000
TZIC	0xA8	封锁中断清除寄存器	0x00000000
TZIE*	0xAC	封锁中断使能寄存器	0x00000000
TZFRC	0xB0	封锁软件强制寄存器	0x00000000
DCALTRIPSEL*	0xB4	数字比较 DCAL 事件选择寄存器	0x00000000
DCAHTRIPSEL*	0xB8	数字比较 DCAH 事件选择寄存器	0x00000000
DCBLTRIPSEL*	0xBC	数字比较 DCBL 事件选择寄存器	0x00000000
DCBHTRIPSEL*	0xC0	数字比较 DCBH 事件选择寄存器	0x00000000
DCACTL*	0xC4	数字比较 A 控制寄存器	0x00000000
DCBCTL*	0xC8	数字比较 B 控制寄存器	0x00000000
DCFCTL*	0xCC	数字比较消隐滤波控制寄存器	0x00000010
DCFOFFSET	0xD0	数字比较消隐窗口偏移量寄存器	0x00000000
DCFOFFSETCNT	0xD4	数字比较消隐窗口偏移计数值寄存器	0x00000000
DCFWINDOW	0xD8	数字比较消隐窗口大小寄存器	0x00000000
DCFWINDOWCNT	0xDC	数字比较消隐窗口大小计数值寄存器	0x00000000
DCCAPCTL	0xE0	数字比较时基计数捕获控制寄存器	0x00000000
DCCAP	0xE4	数字比较时基计数捕获值寄存器	0x00000000

寄存器	偏移地址	描述	复位值
ETCTL*	0xE8	事件触发控制寄存器	0x0007B16F
ETPS	0xEC	事件触发预分频寄存器	0x00000000
ETFLG	0xF0	事件触发标志寄存器	0x00000000
ETCLR	0xF4	事件触发清除寄存器	0x00000000
ETFRC	0xF8	事件触发软件强制寄存器	0x00000000
PWMREGKEY	0xFC	PWM 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 PWMREGKEY=0x1ACCE551 才可以改写。

10.11.2 PWM 寄存器

表 10-8 到表 10-135 给出了 PWM 模块各寄存器的定义。

表 10-8：影子状态寄存器（SHADOWSTS）位段定义

SHADOWSTS (Shadow Status Register) Offset: 0x0 Default: 0x00000000 Access: PWM -> SHADOWSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					DBFED	DBRED	DBCTL
7	6	5	4	3	2	1	0
AQCSFRC	AQCTLB	AQCTLA	CMPD	CMPC	CMPB	CMPA	TBPRD

表 10-9：影子状态寄存器（SHADOWSTS）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	DBFED	RO	0x0	DBFED 寄存器影子状态 0：DBFEDA=DBFED，即已经用影子值重置有效值 1：DBFED 已经写入新的影子值，但尚未重置到有效值中
9	DBRED	RO	0x0	DBRED 寄存器影子状态 0：DBREDA=DBRED，即已经用影子值重置有效值 1：DBRED 已经写入新的影子值，但尚未重置到有效值中
8	DBCTL	RO	0x0	DBCTL 寄存器影子状态 0：DBCTLA=DBCTL，即已经用影子值重置有效值

位段	位段名	属性	复位值	描述
				1：DBCTL 已经写入新的影子值，但尚未重置到有效值中
7	AQCSFRC	RO	0x0	AQCSFRC 寄存器影子状态 0：AQCSFRCA=AQCSFRC，即已经用影子值重置有效值 1：AQCSFRC 已经写入新的影子值，但尚未重置到有效值中
6	AQCTLB	RO	0x0	AQCTLB 寄存器影子状态 0：AQCTLBA=AQCTLB，即已经用影子值重置有效值 1：AQCTLB 已经写入新的影子值，但尚未重置到有效值中
5	AQCTLA	RO	0x0	AQCTLA 寄存器影子状态 0：AQCTLAA=AQCTLA，即已经用影子值重置有效值 1：AQCTLA 已经写入新的影子值，但尚未重置到有效值中
4	CMPD	RO	0x0	CMPD 寄存器影子状态 0：CMPDA=CMPD，即已经用影子值重置有效值 1：CMPD 已经写入新的影子值，但尚未重置到有效值中
3	CMPC	RO	0x0	CMPC 寄存器影子状态 0：CMPCA=CMPC，即已经用影子值重置有效值 1：CMPC 已经写入新的影子值，但尚未重置到有效值中
2	CMPB	RO	0x0	CMPB 寄存器影子状态 0：CMPBA=CMPB，即已经用影子值重置有效值 1：CMPB 已经写入新的影子值，但尚未重置到有效值中
1	CMPA	RO	0x0	CMPA 寄存器影子状态 0：CMPAA=CMPA，即已经用影子值重置有效值 1：CMPA 已经写入新的影子值，但尚未重置到有效值中
0	TBPRD	RO	0x0	TBPRD 寄存器影子状态 0：TBPRDA=TBPRD，即已经用影子值重置有效值 1：TBPRD 已经写入新的影子值，但尚未重置到有效值中

表 10-10：全局影子值重置有效值控制寄存器 0（GLDCTL0）位段定义

GLDCTL0 (Global Shadow to Active Load Control Register 0)　Offset: 0x4　Default: 0x00000000
Access: PWM -> GLDCTL0.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED			OSTMODE	GLDCNT			
7	6	5	4	3	2	1	0
GLDPRD				GLDLOAD			GLDEN

表 10-11：全局影子值重置有效值控制寄存器 0（GLDCTL0）位段描述

位段	位段名	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12	OSTMODE	RW	0x0	单次全局重置模式 0：周期性持续重置模式，由 GLDCNT 对重置事件持续循环计数，当 GLDCNT=GLDPRD 时重置 1：单次重置模式，仅当 GLDCTL1.OSTLOAD=1 时重置事件有效
11:8	GLDCNT	RO	0x0	全局重置事件计数 记录了重置事件发生的次数
7:4	GLDPRD	RW	0x0	全局重置事件周期 本位段为 0 时将不产生重置脉冲；否则当 GLDCNT=GLDPRD 时产生重置脉冲。 OSTMODE=1 时本位段无效
3:1	GLDLOAD	RW	0x0	全局重置事件选择 000：TBCNT=0 001：TBCNT=TBPRD 010：TBCNT=0 or TBCNT=TBPRD 011：SYNC 事件 100：SYNC 事件或 TBCNT=0 101：SYNC 事件或 TBCNT=TBPRD 110：SYNC 事件或 TBCNT=0 或 TBCNT=TBPRD 111：GLDCTL1.FRCLOAD 位段被软件置 1
0	GLDEN	RW	0x0	全局重置使能 0：各寄存器影子值到有效值的重置独立控制，与 GLDSEL 寄存器设置无关 例：CMPA 的重置由 CMPCTL.CMPALOAD 决定 1：GLDSEL 可以全局控制影子值到有效值的重置

表 10-12：全局影子值重置有效值控制寄存器 1（GLDCTL1）位段定义

GLDCTL1 (Global Shadow to Active Load Control Register 1) Offset: 0x8 Default: 0x00000000 Access: PWM -> GLDCTL1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						FRCLOAD	OSTLOAD

表 10-13：全局影子值重置有效值控制寄存器 1（GLDCTL1）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	FRCLOAD	W1S	0x0	触发影子值到有效值的全局重置事件 0：写 0 无效，总是读回 0 1：写 1 产生一次全局重置事件； 本位段自动清零。
0	OSTLOAD	W1S	0x0	启动单次重置等待 0：写 0 无效，读到 0 表明没有等待中的重置 1：写 1 启动单次重置等待，当 GLDCTL0.GLDLOAD 位段指定的事件发生时，重置寄存器有效值，同时本位段自动清零。

表 10-14：全局影子值重置有效值控制选择寄存器（GLDSEL）位段定义

GLDSEL (Global Shadow to Active Load Select Register) Offset: 0xC Default: 0x00000000 Access: PWM -> GLDSEL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					DBFED	DBRED	DBCTL
7	6	5	4	3	2	1	0
AQCSFRC	AQCTLB	AQCTLA	CMPD	CMPC	CMPB	CMPA	TBPRD

表 10-15：全局影子值重置有效值控制选择寄存器（GLDSEL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	DBFED	RW	0x0	DBFED 全局重置选择 0：由 DBCTL.DBFEDLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
9	DBRED	RW	0x0	DBRED 全局重置选择 0：由 DBCTL.DBREDLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
8	DBCTL	RW	0x0	DBCTL 全局重置选择 0：由 DBCTL.DBCTLLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
7	AQCSFRC	RW	0x0	AQCSFRC 全局重置选择 0：由 AQCTL.AQCSFRCLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
6	AQCTLB	RW	0x0	AQCTLB 全局重置选择 0：由 AQCTL.AQCTLBLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
5	AQCTLA	RW	0x0	AQCTLA 全局重置选择 0：由 AQCTL.AQCTLALOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
4	CMPD	RW	0x0	CMPD 全局重置选择 0：由 CMPCTL.CMPDLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
3	CMPC	RW	0x0	CMPC 全局重置选择 0：由 CMPCTL.CMPCLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
2	CMPB	RW	0x0	CMPB 全局重置选择 0：由 CMPCTL.CMPBLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
1	CMPA	RW	0x0	CMPA 全局重置选择 0：由 CMPCTL.CMPALOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机
0	TBPRD	RW	0x0	TBPRD 全局重置选择 0：由 TBCTL.TBPRDLOAD 决定重置时机 1：由 GLDCTL 决定全局重置时机

表 10-16：PWM 链接控制寄存器（PWMLINK）位段定义

PWMLINK (PWM Link Control Register) Offset: 0x10 Default: 0xFFFFFFFF Access: PWM -> PWMLINK.all							
31	30	29	28	27	26	25	24
GLDCTL1				DBFED			
23	22	21	20	19	18	17	16
DBRED				CMPD			
15	14	13	12	11	10	9	8
CMPC				CMPB			
7	6	5	4	3	2	1	0
CMPA				TBPRD			

表 10-17：PWM 链接控制寄存器（PWMLINK）位段描述

位段	位段名	属性	复位值	描述
31:28	GLDCTL1	RW	0xF	往 PWMLINK.GLDCTL1 指定的 PWM 的 GLDCTL1 寄存器写值时同时写本 PWM 的 GLDCTL1 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
27:24	DBFED	RW	0xF	往 PWMLINK.DBFED 指定的 PWM 的 DBFED 寄存器写值时同时写本 PWM 的 DBFED 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
23:20	DBRED	RW	0xF	往 PWMLINK.DBRED 指定的 PWM 的 DBRED 寄存器写值时同时写本 PWM 的 DBRED 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器

位段	位段名	属性	复位值	描述
				0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
19:16	CMPD	RW	0xF	往 PWMLINK.CMPD 指定的 PWM 的 CMPD 寄存器写值时同时写本 PWM 的 CMPD 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
15:12	CMPC	RW	0xF	往 PWMLINK.CMPC 指定的 PWM 的 CMPC 寄存器写值时同时写本 PWM 的 CMPC 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
11:8	CMPB	RW	0xF	往 PWMLINK.CMPB 指定的 PWM 的 CMPB 寄存器写值时同时写本 PWM 的 CMPB 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
7:4	CMPA	RW	0xF	往 PWMLINK.CMPA 指定的 PWM 的 CMPA 寄存器写值时同时写本 PWM 的 CMPA 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器

位段	位段名	属性	复位值	描述
				0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM
3:0	TBPRD	RW	0xF	往 PWMLINK.TBPRD 指定的 PWM 的 TBPRD 寄存器写值时同时写本 PWM 的 TBPRD 寄存器 0000：链接到 PWM0 对应的寄存器 0001：链接到 PWM1 对应的寄存器 0010：链接到 PWM2 对应的寄存器 0011：链接到 PWM3 对应的寄存器 0100：链接到 PWM4 对应的寄存器 0101：链接到 PWM5 对应的寄存器 0110：链接到 PWM6 对应的寄存器 0111：链接到 PWM7 对应的寄存器 其他：不链接到任何 PWM

表 10-18：时基控制寄存器（TBCTL）位段定义

TBCTL (Time-Base Control Register) Offset: 0x14 Default: 0x00000006
Access: PWM -> TBCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				DBGRUN		TBDIVBIN	
15	14	13	12	11	10	9	8
TBDIVBIN	TBDIVLIN			FRCSYNC	SYNCOSEL		
7	6	5	4	3	2	1	0
TBPRDLOCK	TBPRDLOAD		PHSDIR	PHSEN	CNTMODE		CNTRUN

表 10-19：时基控制寄存器（TBCTL）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:18	DBGRUN	RW	0x0	CPU 停止时 PWM 时基计数器的行为 注意：JTAG/SWD 调试或者 CPU 错误异常都有可能导致 CPU 停止 00：完成下一拍计数后停止 01：完成整个计数周期后停止（向上计数模式为 TBCNT=TBPRD，其余模式计数至 TBCNT=0） 1x：保持正常计数
17:15	TBDIVBIN	RW	0x0	时基计数时钟（TBCLK）相对于 PWM 时钟的二进制预分频 总的分频比为 $2^{TBDIVBIN}$ x （TBDIVLIN+1） 0：1 分频（复位默认值） 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
14:12	TBDIVLIN	RW	0x0	时基计数时钟（TBCLK）预分频后进一步分频比 总的分频比为 $2^{TBDIVBIN}$ x （TBDIVLIN+1） 0：1 分频（复位默认值） 1：2 分频 2：3 分频 3：4 分频 4：5 分频 5：6 分频 6：7 分频 7：8 分频
11	FRCSYNC	W1S	0x0	软件强制同步 0：写 0 无效，总是读回 0 1：写 1 产生强制同步，本位段自动清零，且不受 PWMREGKEY 保护
10:8	SYNCOSEL	RW	0x0	同步输出选择 000：SYNCI 和软件强制的同步事件 001：TBCNT=0 事件 010：TBCNT=TBPRD 事件 011：TBCNT=CMPA 事件 100：TBCNT=CMPB 事件 101：TBCNT=CMPC 事件

位段	位段名	属性	复位值	描述
				110：TBCNT=CMPD 事件 111：关闭 SYNCO 输出
7	TBPRDLOCK	RW	0x0	TBPRD 有效值锁定 0：TBPRDA 存储的有效值根据 TBCTL.TBPRDLOAD 位段指定的时机重置 1：TBPRDA 存储的有效值维持不变
6:5	TBPRDLOAD	RW	0x0	TBPRD 有效值重置方式 TBCTL.TBPRDLOCK=1 时本位段无效 00：TBCNT=0 时用 TBPRD 重置 TBPRDA 01：SYNC 事件时用 TBPRD 重置 TBPRDA 10：SYNC 事件或 TBCNT=0 时用 TBPRD 重置 TBPRDA 11：写 TBPRD 寄存器时同时更新 TBPRDA
4	PHSDIR	RW	0x0	相位同步后的计数方向 本位段仅当 TBCNT 处于上下计数模式时有效（CNTMODE=2） 0：同步事件发生后向下计数 1：同步事件发生后向上计数
3	PHSEN	RW	0x0	相位同步使能 0：不用 TBPHS 寄存器更新 TBCNT 计数值 1：同步事件（SYNCI 输入、软件强制同步或数字比较同步事件）发生时，将 TBCNT 计数值更新为 TBPHS 寄存器的值
2:1	CNTMODE	RW	0x3	计数模式 00：向下计数 01：向上计数 10：上下计数 11：停止计数并维持原计数值（复位默认项）
0	CNTRUN	RW	0x0	计数开关控制 0：计数关闭 1：计数开始

表 10-20：时基周期寄存器（TBPRD）位段定义

TBPRD (Time-Base Period Register) Offset: 0x18 Default: 0x00000000 Access: PWM -> TBPRD.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-21：时基周期寄存器（TBPRD）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	时基计数周期

表 10-22：时基周期有效值寄存器（TBPRDA）位段定义

TBPRDA (Time-Base Period Active Register) Offset: 0x1C Default: 0x00000000 Access: PWM -> TBPRDA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-23：时基周期有效值寄存器（TBPRDA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	时基计数周期有效值 当 TBCTL.PRDLOAD=3，写 TBPRD 寄存器直接同时改变本寄存器值； 否则，根据 TBCTL.PRDLOAD 指定的方式从 TBPRD 寄存器重置本寄存器值。

表 10-24：时基相位寄存器（TBPHS）位段定义

TBPHS (Time-Base Phase Register) Offset: 0x20 Default: 0x00000000 Access: PWM -> TBPHS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-25：时基相位寄存器（TBPHS）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	同步事件发生时时基计数器待更新的相位 TBCTL.PHSEN=0：忽略同步事件，TBCNT 计数值维持不变。 TBCTL.PHSEN=1：同步事件发生时更新 TBCNT 为本寄存器值。

表 10-26：时基计数值寄存器（TBCNT）位段定义

TBCNT (Time-Base Counter Register) Offset: 0x24 Default: 0x00000000 Access: PWM -> TBCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-27：时基计数值寄存器（TBCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	时基计数器当前值

表 10-28：时基状态寄存器（TBSTS）位段定义

TBSTS (Time-Base Status Register) Offset: 0x28 Default: 0x00000100 Access: PWM -> TBSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							CNTDIR
7	6	5	4	3	2	1	0
CNTCMPD	CNTCMPC	CNTCMPB	CNTCMPA	CNTPRD	CNTZRO	CNTMAX	SYNCI

表 10-29：时基状态寄存器（TBSTS）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	CNTDIR	RO	0x1	时基计数器计数方向 0：时基计数器正递减计数 1：时基计数器正递增计数
7	CNTCMPD	RO	0x0	时基计数至 CMPD 标志 0：TBCNT 从未计数至 CMPD 1：TBCNT 曾计数至 CMPD
6	CNTCMPC	RO	0x0	时基计数至 CMPC 标志 0：TBCNT 从未计数至 CMPC 1：TBCNT 曾计数至 CMPC
5	CNTCMPB	RO	0x0	时基计数至 CMPB 标志 0：TBCNT 从未计数至 CMPB 1：TBCNT 曾计数至 CMPB
4	CNTCMPA	RO	0x0	时基计数至 CMPA 标志 0：TBCNT 从未计数至 CMPA 1：TBCNT 曾计数至 CMPA
3	CNTPRD	RO	0x0	时基计数至 TBPRD 标志 0：TBCNT 从未计数至 TBPRD 1：TBCNT 曾计数至 TBPRD
2	CNTZRO	RO	0x0	时基计数至 0 标志 0：TBCNT 从未计数至 0 1：TBCNT 曾计数至 0
1	CNTMAX	RO	0x0	时基计数至 0xFFFF 标志 0：TBCNT 从未计数至 0xFFFF 1：TBCNT 曾计数至 0xFFFF
0	SYNCI	RO	0x0	输入同步事件发生标志 0：输入同步事件从未发生 1：输入同步事件曾经发生

表 10-30：时基状态清除寄存器（TBSTSCLR）位段定义

TBSTSCLR (Time-Base Status Clear Register) Offset: 0x2C Default: 0x00000000 Access: PWM -> TBSTSCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 CNTCMPD	6 CNTCMPC	5 CNTCMPB	4 CNTCMPA	3 CNTPRD	2 CNTZRO	1 CNTMAX	0 SYNCI

表 10-31：时基状态清除寄存器（TBSTSCLR）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	CNTCMPD	W1C	0x0	时基计数至 CMPD 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
6	CNTCMPC	W1C	0x0	时基计数至 CMPC 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
5	CNTCMPB	W1C	0x0	时基计数至 CMPB 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
4	CNTCMPA	W1C	0x0	时基计数至 CMPA 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
3	CNTPRD	W1C	0x0	时基计数至 TBPRD 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
2	CNTZRO	W1C	0x0	时基计数至 0 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零

位段	位段名	属性	复位值	描述
1	CNTMAX	W1C	0x0	时基计数至 0xFFFF 状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零
0	SYNCI	W1C	0x0	输入同步事件发生状态清除 0：写 0 无效，总是读回 0 1：写 1 清除对应的状态标志 本位段自动清零

表 10-32：计数比较控制寄存器（CMPCTL）位段定义

CMPCTL (Counter-Compare Control Register)　Offset: 0x30　Default: 0x00000000
Access: PWM -> CMPCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
CMPDLOCK	CMPDLOAD			CMPCLOCK	CMPCLOAD		
7	6	5	4	3	2	1	0
CMPBLOCK	CMPBLOAD			CMPALOCK	CMPALOAD		

表 10-33：计数比较控制寄存器（CMPCTL）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	CMPDLOCK	RW	0x0	CMPD 有效值锁定 0：CMPDA 存储的有效值在 CMPCTL.CMPDLOAD 指定的时机用 CMPD 的值重置 1：CMPDA 存储的有效值维持不变
14:12	CMPDLOAD	RW	0x0	CMPD 有效值重置方式 CMPCTL.CMPDLOCK=1 时本位段无效 000：TBCNT=0 时用 CMPD 重置 CMPDA 001：TBCNT=TBPRD 时用 CMPD 重置 CMPDA 010：TBCNT=0 或 TBCNT=TBPRD 时用 CMPD 重置 CMPDA 011：SYNC 事件时用 CMPD 重置 CMPDA 100：SYNC 事件或 TBCNT=0 时用 CMPD 重置 CMPDA 101：SYNC 事件或 TBCNT=TBPRD 时用 CMPD 重置 CMPDA 110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 CMPD 重置 CMPDA

位段	位段名	属性	复位值	描述
				111：写 CMPD 寄存器时同时更新 CMPDA
11	CMPCLOCK	RW	0x0	CMPC 有效值锁定 0：CMPCA 存储的有效值在 CMPCTL.CMPCLOAD 指定的时机用 CMPC 的值重置 1：CMPCA 存储的有效值维持不变
10:8	CMPCLOAD	RW	0x0	CMPC 有效值重置方式 CMPCTL.CMPCLOCK=1 时本位段无效 000：TBCNT=0 时用 CMPC 重置 CMPCA 001：TBCNT=TBPRD 时用 CMPC 重置 CMPCA 010：TBCNT=0 或 TBCNT=TBPRD 时用 CMPC 重置 CMPCA 011：SYNC 事件时用 CMPC 重置 CMPCA 100：SYNC 事件或 TBCNT=0 时用 CMPC 重置 CMPCA 101：SYNC 事件或 TBCNT=TBPRD 时用 CMPC 重置 CMPCA 110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 CMPC 重置 CMPCA 111：写 CMPC 寄存器时同时更新 CMPCA
7	CMPBLOCK	RW	0x0	CMPB 有效值锁定 0：CMPBA 存储的有效值在 CMPCTL.CMPBLOAD 指定的时机用 CMPB 的值重置 1：CMPBA 存储的有效值维持不变
6:4	CMPBLOAD	RW	0x0	CMPB 有效值重置方式 CMPCTL.CMPBLOCK=1 时本位段无效 000：TBCNT=0 时用 CMPB 重置 CMPBA 001：TBCNT=TBPRD 时用 CMPB 重置 CMPBA 010：TBCNT=0 或 TBCNT=TBPRD 时用 CMPB 重置 CMPBA 011：SYNC 事件时用 CMPB 重置 CMPBA 100：SYNC 事件或 TBCNT=0 时用 CMPB 重置 CMPBA 101：SYNC 事件或 TBCNT=TBPRD 时用 CMPB 重置 CMPBA 110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 CMPB 重置 CMPBA 111：写 CMPB 寄存器时同时更新 CMPBA
3	CMPALOCK	RW	0x0	CMPA 有效值锁定 0：CMPAA 存储的有效值在 CMPCTL.CMPALOAD 指定的时机用 CMPA 的值重置 1：CMPAA 存储的有效值维持不变

位段	位段名	属性	复位值	描述
2:0	CMPALOAD	RW	0x0	CMPA 有效值重置方式 CMPCTL.CMPALOCK=1 时本位段无效 000：TBCNT=0 时用 CMPA 重置 CMPAA 001：TBCNT=TBPRD 时用 CMPA 重置 CMPAA 010：TBCNT=0 或 TBCNT=TBPRD 时用 CMPA 重置 CMPAA 011：SYNC 事件时用 CMPA 重置 CMPAA 100：SYNC 事件或 TBCNT=0 时用 CMPA 重置 CMPAA 101：SYNC 事件或 TBCNT=TBPRD 时用 CMPA 重置 CMPAA 110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 CMPA 重置 CMPAA 111：写 CMPA 寄存器时同时更新 CMPAA

表 10-34：计数比较阈值 A 寄存器（CMPA）位段定义

CMPA (Counter-Compare A Threshold Register) Offset: 0x34 Default: 0x00000000
Access: PWM -> CMPA.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-35：计数比较阈值 A 寄存器（CMPA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	计数比较阈值 A

表 10-36：计数比较阈值 A 有效值寄存器（CMPAA）位段定义

CMPAA (Counter-Compare A Threshold Active Register) Offset: 0x38 Default: 0x00000000 Access: PWM -> CMPAA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-37：计数比较阈值 A 有效值寄存器（CMPAA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	计数比较阈值 A 有效值 TBCNT 等于该值时，产生 TBCNT=CMPA 事件

表 10-38：计数比较阈值 B 寄存器（CMPB）位段定义

CMPB (Counter-Compare B Threshold Register) Offset: 0x3C Default: 0x00000000 Access: PWM -> CMPB.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-39：计数比较阈值 B 寄存器（CMPB）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	计数比较阈值 B

表 10-40：计数比较阈值 B 寄存器（CMPBA）位段定义

CMPBA (Counter-Compare B Threshold Active Register) Offset: 0x40 Default: 0x00000000 Access: PWM -> CMPBA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-41：计数比较阈值 B 寄存器（CMPBA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	计数比较阈值 B 有效值 TBCNT 等于该值时，产生 TBCNT=CMPB 事件

表 10-42：计数比较阈值 C 寄存器（CMPC）位段定义

CMPC (Counter-Compare C Threshold Register) Offset: 0x44 Default: 0x00000000 Access: PWM -> CMPC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-43：计数比较阈值 C 寄存器（CMPC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	计数比较阈值 C

表 10-44：计数比较阈值 C 有效值寄存器（CMPCA）位段定义

CMPCA (Counter-Compare C Threshold Active Register) Offset: 0x48 Default: 0x00000000 Access: PWM -> CMPCA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-45：计数比较阈值 C 有效值寄存器（CMPCA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	计数比较阈值 C 有效值 TBCNT 等于该值时，产生 TBCNT=CMPC 事件

表 10-46：计数比较阈值 D 寄存器（CMPD）位段定义

CMPD (Counter-Compare D Threshold Register) Offset: 0x4C Default: 0x00000000 Access: PWM -> CMPD.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-47：计数比较阈值 D 寄存器（CMPD）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	计数比较阈值 D

表 10-48：计数比较阈值 D 有效值寄存器（CMPDA）位段定义

CMPDA (Counter-Compare D Threshold Active Register) Offset: 0x50 Default: 0x00000000 Access: PWM -> CMPDA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-49：计数比较阈值 D 有效值寄存器（CMPDA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	计数比较阈值 D 有效值 TBCNT 等于该值时，产生 TBCNT=CMPD 事件

表 10-50：行为限定控制寄存器（AQCTL）位段定义

AQCTL (Action-Qualifier Control Register) Offset: 0x54 Default: 0x00000000 Access: PWM -> AQCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
T1SEL				T0SEL			
7	6	5	4	3	2	1	0
AQCTLBLOCK	AQCTLBLOAD			AQCTLALOCK	AQCTLALOAD		

表 10-51：行为限定控制寄存器（AQCTL）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:12	T1SEL	RW	0x0	T1 事件源选择 0000：DCAEVT0 0001：DCAEVT1 0010：DCBEVT0 0011：DCBEVT1 0100：TZ0 0101：TZ1 0110：TZ2 0111：TZ3 1000：TZ4 1001：SYNCI 其他：禁用 T1 事件
11:8	T0SEL	RW	0x0	T0 事件源选择 0000：DCAEVT0 0001：DCAEVT1 0010：DCBEVT0 0011：DCBEVT1 0100：TZ0 0101：TZ1 0110：TZ2 0111：TZ3 1000：TZ4 1001：SYNCI 其他：禁用 T0 事件
7	AQCTLBLOCK	RW	0x0	AQCTLB 有效值锁定 0：AQCTLBA 维持的有效值由 QCTL.AQCTLBLOAD 指定的时机用 AQCTLB 的值重置 1：AQCTLBA 存储的有效值维持不变
6:4	AQCTLBLOAD	RW	0x0	AQCTLB 有效值重置方式 AQCTL.AQCTLBLOCK=1 本位段无效 000：TBCNT=0 时用 AQCTLB 重置 AQCTLBA 001：TBCNT=TBPRD 时用 AQCTLB 重置 AQCTLBA 010：TBCNT=0 或 TBCNT=TBPRD 时用 AQCTLB 重置 AQCTLBA 011：SYNC 事件时用 AQCTLB 重置 AQCTLBA 100：SYNC 事件或 BCNT=0 时用 AQCTLB 重置 AQCTLBA 101：SYNC 事件或 BCNT=TBPRD 时用 AQCTLB 重置 AQCTLBA

位段	位段名	属性	复位值	描述
				110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 AQCTLB 重置 AQCTLBA 111：写 AQCTLB 寄存器时同时更新 AQCTLBA
3	AQCTLALOCK	RW	0x0	AQCTLA 有效值锁定 0：AQCTLAA 维持的有效值由 QCTL.AQCTLALOAD 指定的时机用 AQCTLA 的值重置 1：AQCTLAA 存储的有效值维持不变
2:0	AQCTLALOAD	RW	0x0	AQCTLA 有效值重置方式 AQCTL.AQCTLALOCK=1 本位段无效 000：TBCNT=0 时用 AQCTLA 重置 AQCTLAA 001：TBCNT=TBPRD 时用 AQCTLA 重置 AQCTLAA 010：TBCNT=0 或 TBCNT=TBPRD 时用 AQCTLA 重置 AQCTLAA 011：SYNC 事件时用 AQCTLA 重置 AQCTLAA 100：SYNC 事件或 BCNT=0 时用 AQCTLA 重置 AQCTLAA 101：SYNC 事件或 BCNT=TBPRD 时用 AQCTLA 重置 AQCTLAA 110：SYNC 事件、TBCNT=0 或 TBCNT=TBPRD 时用 AQCTLA 重置 AQCTLAA 111：写 AQCTLA 寄存器时同时更新 AQCTLAA

表 10-52：行为限定 A 路输出控制寄存器（AQCTLA）位段定义

AQCTLA (Action-Qualifier Output A Control Register)　Offset: 0x58　Default: 0x00000000
Access: PWM -> AQCTLA.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				T1D		T1U	
15	14	13	12	11	10	9	8
T0D		T0U		CBD		CBU	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	

表 10-53：行为限定 A 路输出控制寄存器（AQCTLA）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:18	T1D	RW	0x0	T1 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
17:16	T1U	RW	0x0	T1 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
15:14	T0D	RW	0x0	T0 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
13:12	T0U	RW	0x0	T0 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
11:10	CBD	RW	0x0	TBCNT=CMPB 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
9:8	CBU	RW	0x0	TBCNT=CMPB 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）

位段	位段名	属性	复位值	描述
7:6	CAD	RW	0x0	TBCNT=CMPA 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
5:4	CAU	RW	0x0	TBCNT=CMPA 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
3:2	PRD	RW	0x0	TBCNT=TBPRD 事件时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
1:0	ZRO	RW	0x0	TBCNT=0 事件时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）

表 10-54：行为限定 A 路输出控制有效值寄存器（AQCTLAA）位段定义

AQCTLAA (Action-Qualifier Output A Control Active Register)　Offset: 0x5C　Default: 0x00000000
Access: PWM -> AQCTLAA.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				T1D		T1U	
15	14	13	12	11	10	9	8
T0D		T0U		CBD		CBU	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	

表 10-55：行为限定 A 路输出控制有效值寄存器（AQCTLAA）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:18	T1D	RO	0x0	T1 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
17:16	T1U	RO	0x0	T1 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
15:14	T0D	RO	0x0	T0 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
13:12	T0U	RO	0x0	T0 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
11:10	CBD	RO	0x0	TBCNT=CMPB 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
9:8	CBU	RO	0x0	TBCNT=CMPB 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）

位段	位段名	属性	复位值	描述
7:6	CAD	RO	0x0	TBCNT=CMPA 事件发生且 TBCNT 递减计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
5:4	CAU	RO	0x0	TBCNT=CMPA 事件发生且 TBCNT 递增计数时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
3:2	PRD	RO	0x0	TBCNT=TBPRD 事件时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）
1:0	ZRO	RO	0x0	TBCNT=0 事件时的 A 路输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）

表 10-56：行为限定 B 路输出控制寄存器（AQCTLB）位段定义

AQCTLB (Action-Qualifier Output B Control Register)　Offset: 0x60　Default: 0x00000000
Access: PWM -> AQCTLB.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				T1D		T1U	
15	14	13	12	11	10	9	8
T0D		T0U		CBD		CBU	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	

表 10-57：行为限定 B 路输出控制寄存器（AQCTLB）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:18	T1D	RW	0x0	T1 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
17:16	T1U	RW	0x0	T1 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
15:14	T0D	RW	0x0	T0 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
13:12	T0U	RW	0x0	T0 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
11:10	CBD	RW	0x0	TBCNT=CMPB 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
9:8	CBU	RW	0x0	TBCNT=CMPB 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）

位段	位段名	属性	复位值	描述
7:6	CAD	RW	0x0	TBCNT=CMPA 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
5:4	CAU	RW	0x0	TBCNT=CMPA 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
3:2	PRD	RW	0x0	TBCNT=TBPRD 事件时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
1:0	ZRO	RW	0x0	TBCNT=0 事件时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）

表 10-58：行为限定 B 路输出控制有效值寄存器（AQCTLBA）位段定义

AQCTLBA (Action-Qualifier Output B Control Active Register)　Offset: 0x64　Default: 0x00000000
Access: PWM -> AQCTLBA.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				T1D		T1U	
15	14	13	12	11	10	9	8
T0D		T0U		CBD		CBU	
7	6	5	4	3	2	1	0
CAD		CAU		PRD		ZRO	

表 10-59：行为限定 B 路输出控制有效值寄存器（AQCTLBA）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19:18	T1D	RO	0x0	T1 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
17:16	T1U	RO	0x0	T1 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
15:14	T0D	RO	0x0	T0 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
13:12	T0U	RO	0x0	T0 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
11:10	CBD	RO	0x0	TBCNT=CMPB 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
9:8	CBU	RO	0x0	TBCNT=CMPB 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）

位段	位段名	属性	复位值	描述
7:6	CAD	RO	0x0	TBCNT=CMPA 事件发生且 TBCNT 递减计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
5:4	CAU	RO	0x0	TBCNT=CMPA 事件发生且 TBCNT 递增计数时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
3:2	PRD	RO	0x0	TBCNT=TBPRD 事件时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
1:0	ZRO	RO	0x0	TBCNT=0 事件时的 B 路输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）

表 10-60：行为限定软件强制寄存器（AQSFRC）位段定义

AQSFRC (Action-Qualifier Software Force Register) Offset: 0x68 Default: 0x00000000 Access: PWM -> AQSFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
CSFLOAD		OTSFB	ACTSFB		OTSFA	ACTSFA	

表 10-61：行为限定软件强制寄存器（AQSFRC）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:6	CSFLOAD	RW	0x0	AQCSFRC 寄存器有效值重置方式 00：TBCNT=0 时用 AQCSFRC 更新 AQCSFRCA 01：TBCNT=TBPRD 时用 AQCSFRC 更新 AQCSFRCA 10：TBCNT=0 或 TBCNT=TBPRD 时用 AQCSFRC 更新 AQCSFRCA 11：写 AQCSFRC 寄存器时同时更新 AQCSFRCA
5	OTSFB	W1S	0x0	B 路输出行为限定单次软件强制 0：写 0 无效，总是读回 0 1：写 1 产生单次行为限定软件强制事件 本位段自动清零
4:3	ACTSFB	RW	0x0	B 路行为限定单次软件强制时的输出 00：不变（禁止操作） 01：清零（PWMxB 输出低） 10：置高（PWMxB 输出高） 11：翻转（PWMxB 当前输出翻转）
2	OTSFA	W1S	0x0	A 路输出单次软件强制 0：写 0 无效，总是读回 0 1：写 1 产生单次行为限定软件强制事件 本位段自动清零
1:0	ACTSFA	RW	0x0	A 路行为限定单次软件强制时的输出 00：不变（禁止操作） 01：清零（PWMxA 输出低） 10：置高（PWMxA 输出高） 11：翻转（PWMxA 当前输出翻转）

表 10-62：行为限定软件持续强制寄存器（AQCSFRC）位段定义

AQCSFRC (Action-Qualifier Continuous Software Force Register) Offset: 0x6C Default: 0x00000000 Access: PWM -> AQCSFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				CSFB		CSFA	

表 10-63：行为限定软件持续强制寄存器（AQCSFRC）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:2	CSFB	RW	0x0	B 路软件持续强制 00：不强制 01：强制 B 路输出持续为低 10：强制 B 路输出持续为高 11：不强制
1:0	CSFA	RW	0x0	A 路软件持续强制 00：不强制 01：强制 B 路输出持续为低 10：强制 B 路输出持续为高 11：不强制

表 10-64：行为限定软件持续强制有效值寄存器（AQCSFRCA）位段定义

AQCSFRCA (Action-Qualifier Continuous Software Force Active Register) Offset: 0x70 Default: 0x00000000 Access: PWM -> AQCSFRCA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				CSFB		CSFA	

表 10-65：行为限定软件持续强制有效值寄存器（AQCSFRCA）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:2	CSFB	RO	0x0	B 路软件持续强制 00：不强制 01：强制 B 路输出持续为低 10：强制 B 路输出持续为高 11：不强制
1:0	CSFA	RO	0x0	A 路软件持续强制 00：不强制 01：强制 B 路输出持续为低 10：强制 B 路输出持续为高 11：不强制

表 10-66：死区产生控制寄存器（DBCTL）位段定义

DBCTL (Dead-Band Generator Control Register) Offset: 0x74 Default: 0x00000070
Access: PWM -> DBCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED					DBFEDLOCK	DBFEDLOAD	
15	14	13	12	11	10	9	8
DBREDLOCK	DBREDLOAD		DBCTLLOCK	DBCTLLOAD		HALFCYCLE	DUALEDGE
7	6	5	4	3	2	1	0
OUTBSRC	OUTASRC	FEDPOL	REDPOL	FEDSRC	REDSRC	FEDEN	REDEN

表 10-67：死区产生控制寄存器（DBCTL）位段描述

位段	位段名	属性	复位值	描述
31:19	RESERVED_31_19	RO	0x0	保留
18	DBFEDLOCK	RW	0x0	DBFED 有效值锁定 0：DBFEDA 维持的有效值由 DBCTL.DBFEDLOAD 指定的时机用 DBFED 的值重置 1：DBFEDA 存储的有效值维持不变
17:16	DBFEDLOAD	RW	0x0	DBFED 有效值重置方式 DBCTL.DBFEDLOCK=1 时本位段无效 00：TBCNT=0 时用 DBFED 重置 DBFEDA 01：TBCNT=TBPRD 时用 DBFED 重置 DBFEDA 10：TBCNT=0 或 TBCNT=TBPRD 时用 DBFED 重置 DBFEDA 11：写 DBFED 寄存器时同时更新 DBFEDA

位段	位段名	属性	复位值	描述
15	DBREDLOCK	RW	0x0	DBRED 有效值锁定 0：DBREDA 存储的有效值由 DBCTL.DBREDLOAD 指定的时机用 DBRED 的值重置 1：DBREDA 存储的有效值维持不变
14:13	DBREDLOAD	RW	0x0	DBRED 有效值重置方式 DBCTL.DBREDLOCK=1 时本位段无效 00：TBCNT=0 时用 DBRED 重置 DBREDA 01：TBCNT=TBPRD 时用 DBRED 重置 DBREDA 10：TBCNT=0 或 TBCNT=TBPRD 时用 DBRED 重置 DBREDA 11：写 DBRED 寄存器时同时更新 DBREDA
12	DBCTLLOCK	RW	0x0	DBCTL[9:0]有效值锁定 0：DBCTLA 存储的有效值由 DBCTL.DBCTLLOAD 指定的时机用 DBCTL[9:0]的值重置 1：DBCTLA 存储的有效值维持不变
11:10	DBCTLLOAD	RW	0x0	DBCTL[9:0]有效值重置方式 DBCTL.DBCTLLOCK=1 时本位段无效 00：TBCNT=0 时用 DBCTL[9:0]重置 DBCTLA 01：TBCNT=TBPRD 时用 DBCTL[9:0]重置 DBCTLA 10：TBCNT=0 或 TBCNT=TBPRD 时用 DBCTL[9:0] 重置 DBCTLA 11：写 DBCTL[9:0]时同时更新 DBCTLA
9	HALFCYCLE	RW	0x0	半时钟周期模式使能 0：全时钟周期模式，即死区控制计数时钟为时基计数时钟 TBCLK 1：半时钟周期模式，即死区控制计数时钟为 2 倍时基计数时钟 TBCLK
8	DUALEDGE	RW	0x0	双边沿延迟模式 0：上升沿延迟和下降沿延迟为两个独立通路 1：对同一个信号输入先做上升沿延迟，再做下降沿延迟
7	OUTBSRC	RW	0x0	B 路输出源 0：下降沿延迟通路作为 B 路输出 1：上升沿延迟通路作为 B 路输出
6	OUTASRC	RW	0x1	A 路输出源 0：下降沿延迟通路作为 A 路输出 1：上升沿延迟通路作为 A 路输出

位段	位段名	属性	复位值	描述
5	FEDPOL	RW	0x1	下降沿延迟通路输出极性 0：下降沿延迟通路输出取反以得到低电平有效的信号 1：下降沿延迟通路输出保持原始高电平有效的极性
4	REDPOL	RW	0x1	上升沿延迟通路输出极性 0：上升沿延迟通路输出取反以得到低电平有效的信号 1：上升沿延迟通路输出保持原始高电平有效的极性
3	FEDSRC	RW	0x0	下降沿延迟通路输入源 0：A 路输入作为下降沿延迟通路源 1：B 路输入作为下降沿延迟通路源
2	REDSRC	RW	0x0	上升沿延迟通路输入源 0：A 路输入作为上升沿延迟通路源 1：B 路输入作为上升沿延迟通路源
1	FEDEN	RW	0x0	下降沿延时通路使能 0：绕开下降沿延时通路，将行为限定子模块输出的信号直接送至下一级 1：启用下降沿延时通路
0	REDEN	RW	0x0	上升沿延时通路使能 0：绕开上升沿延时通路，将行为限定子模块输出的信号直接送至下一级 1：启用上升沿延时通路

表 10-68：死区产生控制有效值寄存器（DBCTLA）位段定义

DBCTLA (Dead-Band Generator Control Active Register) Offset: 0x78 Default: 0x00000070 Access: PWM -> DBCTLA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						HALFCYCLE	DUALEDGE
7	6	5	4	3	2	1	0
OUTBSRC	OUTASRC	FEDPOL	REDPOL	FEDSRC	REDSRC	FEDEN	REDEN

表 10-69：死区产生控制有效值寄存器（DBCTLA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9	HALFCYCLE	RO	0x0	半时钟周期模式使能 0：全时钟周期模式，即死区控制计数时钟为时基计数时钟 TBCLK 1：半时钟周期模式，即死区控制计数时钟为 2 倍时基计数时钟 TBCLK
8	DUALEDGE	RO	0x0	双边沿延迟模式 0：上升沿延迟和下降沿延迟为两个独立通路 1：对同一个信号输入先做上升沿延迟，再做下降沿延迟
7	OUTBSRC	RO	0x0	B 路输出源 0：下降沿延迟通路作为 B 路输出 1：上升沿延迟通路作为 B 路输出
6	OUTASRC	RO	0x1	A 路输出源 0：下降沿延迟通路作为 A 路输出 1：上升沿延迟通路作为 A 路输出
5	FEDPOL	RO	0x1	下降沿延迟通路输出极性 0：下降沿延迟通路输出取反以得到低电平有效的信号 1：下降沿延迟通路输出保持原始高电平有效
4	REDPOL	RO	0x1	上升沿延迟通路输出极性 0：上升沿延迟通路输出取反以得到低电平有效的信号 1：上升沿延迟通路输出保持原始高电平有效

位段	位段名	属性	复位值	描述
3	FEDSRC	RO	0x0	下降沿延迟通路输入源 0：A 路输入作为下降沿延迟通路源 1：B 路输入作为下降沿延迟通路源
2	REDSRC	RO	0x0	上升沿延迟通路输入源 0：A 路输入作为上升沿延迟通路源 1：B 路输入作为上升沿延迟通路源
1	FEDEN	RO	0x0	下降沿延时通路使能 0：绕开下降沿延时通路，将行为限定子模块输出的信号直接送至下一级 1：启用下降沿延时通路
0	REDEN	RO	0x0	上升沿延时通路使能 0：绕开上升沿延时通路，将行为限定子模块输出的信号直接送至下一级 1：启用上升沿延时通路

表 10-70：死区产生上升沿延迟量寄存器（DBRED）位段定义

DBRED (Dead-Band Generator Rising Edge Delay Register) Offset: 0x7C Default: 0x00000000
Access: PWM -> DBRED.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-71：死区产生上升沿延迟量寄存器（DBRED）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	上升沿延迟量

表 10-72：死区产生上升沿延迟量有效值寄存器（DBREDA）位段定义

DBREDA (Dead-Band Generator Rising Edge Delay Active Register) Offset: 0x80 Default: 0x00000000 Access: PWM -> DBREDA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-73：死区产生上升沿延迟量有效值寄存器（DBREDA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	上升沿延迟量

表 10-74：死区产生下降沿延迟量寄存器（DBFED）位段定义

DBFED (Dead-Band Generator Falling Edge Delay Register) Offset: 0x84 Default: 0x00000000 Access: PWM -> DBFED.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-75：死区产生下降沿延迟量寄存器（DBFED）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	下降沿延迟量

表 10-76：死区产生下降沿延迟量有效值寄存器（DBFEDA）位段定义

DBFEDA (Dead-Band Generator Falling Edge Delay Active Register) Offset: 0x88 Default: 0x00000000 Access: PWM -> DBFEDA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-77：死区产生下降沿延迟量有效值寄存器（DBFEDA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	下降沿延迟量

表 10-78：封锁事件选择寄存器（TZSEL）位段定义

TZSEL (Trip-Zone Event Select Register) Offset: 0x8C Default: 0x00030003 Access: PWM -> TZSEL.all							
31	30	29	28	27	26	25	24
RESERVED					DBGCBC	DCBEVT1	DCAEVT1
23	22	21	20	19	18	17	16
CLKERRCBC	TZ4CBC	TZ3CBC	TZ2CBC	TZ1CBC	TZ0CBC	CBCOUT	
15	14	13	12	11	10	9	8
RESERVED					DBGOST	DCBEVT0	DCAEVT0
7	6	5	4	3	2	1	0
CLKERROST	TZ4OST	TZ3OST	TZ2OST	TZ1OST	TZ0OST	OSTOUT	

表 10-79：封锁事件选择寄存器（TZSEL）位段描述

位段	位段名	属性	复位值	描述
31:27	RESERVED_31_27	RO	0x0	保留
26	DBGCBC	RW	0x0	选中（逻辑或）调试或异常等原因引起的 CPU 停止（lockup 或者 halted）作为周期性封锁事件 0：不选中 1：选中
25	DCBEVT1	RW	0x0	选中（逻辑或）数字比较事件 DCBEVT1 作为周期性封锁事件 0：不选中 1：选中
24	DCAEVT1	RW	0x0	选中（逻辑或）数字比较事件 DCAEVT1 作为周期性封锁事件

位段	位段名	属性	复位值	描述
				0：不选中 1：选中
23	CLKERRCBC	RW	0x0	选中（逻辑或）时钟错误作为周期性封锁事件 0：不选中 1：选中
22	TZ4CBC	RW	0x0	选中（逻辑或）TZ4 作为周期性封锁事件 0：不选中 1：选中
21	TZ3CBC	RW	0x0	选中（逻辑或）TZ3 作为周期性封锁事件 0：不选中 1：选中
20	TZ2CBC	RW	0x0	选中（逻辑或）TZ2 作为周期性封锁事件 0：不选中 1：选中
19	TZ1CBC	RW	0x0	选中（逻辑或）TZ1 作为周期性封锁事件 0：不选中 1：选中
18	TZ0CBC	RW	0x0	选中（逻辑或）TZ0 作为周期性封锁事件 0：不选中 1：选中
17:16	CBCOUT	RW	0x3	周期性封锁信号输出选择 00：不输出周期性封锁事件 01：仅选择异步通路，即各选中事件原始逻辑或 10：仅选择锁存通路 11：同时选择异步通路和锁存通路（逻辑或）
15:11	RESERVED_15_11	RO	0x0	保留
10	DBGOST	RW	0x0	选中（逻辑或）调试或异常等原因引起的 CPU 停止（lockup 或者 halted）作为一次性封锁事件 0：不选中 1：选中
9	DCBEVT0	RW	0x0	选中（逻辑或）DCBEVT0 作为一次性封锁事件 0：不选中 1：选中
8	DCAEVT0	RW	0x0	选中（逻辑或）DCAEVT0 作为一次性封锁事件 0：不选中 1：选中
7	CLKERROST	RW	0x0	选中（逻辑或）时钟错误作为一次性封锁事件 0：不选中 1：选中
6	TZ4OST	RW	0x0	选中（逻辑或）TZ4 作为一次性封锁事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
5	TZ3OST	RW	0x0	选中（逻辑或）TZ3 作为一次性封锁事件 0：不选中 1：选中
4	TZ2OST	RW	0x0	选中（逻辑或）TZ2 作为一次性封锁事件 0：不选中 1：选中
3	TZ1OST	RW	0x0	选中（逻辑或）TZ1 作为一次性封锁事件 0：不选中 1：选中
2	TZ0OST	RW	0x0	选中（逻辑或）TZ0 作为一次性封锁事件 0：不选中 1：选中
1:0	OSTOUT	RW	0x3	一次性封锁信号输出选择 00：不输出一次性封锁事件 01：仅选择异步通路，即各选中事件原始逻辑或 10：仅选择锁存通路 11：同时选择异步通路和锁存通路（逻辑或）

表 10-80：封锁状态寄存器（TZSTS）位段定义

TZSTS (Trip-Zone Status Register) Offset: 0x90 Default: 0x00000000 Access: PWM -> TZSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED						DBGCBC	DCBEVT1
15 DCAEVT1	14 CLKERRCBC	13 TZ4CBC	12 TZ3CBC	11 TZ2CBC	10 TZ1CBC	9 TZ0CBC	8 DBGOST
7 DCBEVT0	6 DCAEVT0	5 CLKERROST	4 TZ4OST	3 TZ3OST	2 TZ2OST	1 TZ1OST	0 TZ0OST

表 10-81：封锁状态寄存器（TZSTS）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	DBGCBC	RO	0x0	锁存的 CPU 停止（由调试或异常错误等原因引起）触发的周期性封锁标志 0：该封锁事件未发生 1：该封锁事件发生
16	DCBEVT1	RO	0x0	锁存的数字比较 DCBEVT1 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生

位段	位段名	属性	复位值	描述
15	DCAEVT1	RO	0x0	锁存的数字比较 DCAEVT1 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
14	CLKERRCBC	RO	0x0	锁存的时钟错误触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
13	TZ4CBC	RO	0x0	锁存的 TZ4 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
12	TZ3CBC	RO	0x0	锁存的 TZ3 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
11	TZ2CBC	RO	0x0	锁存的 TZ2 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
10	TZ1CBC	RO	0x0	锁存的 TZ1 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
9	TZ0CBC	RO	0x0	锁存的 TZ0 触发的周期封锁标志 0：该封锁事件未发生 1：该封锁事件发生
8	DBGOST	RO	0x0	锁存的 CPU 停止（由调试或异常错误等原因引起）触发的一次性封锁标志 0：该封锁事件未发生 1：该封锁事件发生
7	DCBEVT0	RO	0x0	锁存的数字比较 DCBEVT0 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
6	DCAEVT0	RO	0x0	锁存的数字比较 DCAEVT0 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
5	CLKERROST	RO	0x0	锁存的时钟错误触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生

位段	位段名	属性	复位值	描述
4	TZ4OST	RO	0x0	锁存的 TZ4 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
3	TZ3OST	RO	0x0	锁存的 TZ3 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
2	TZ2OST	RO	0x0	锁存的 TZ2 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
1	TZ1OST	RO	0x0	锁存的 TZ1 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生
0	TZ0OST	RO	0x0	锁存的 TZ0 触发的一次封锁标志 0：该封锁事件未发生 1：该封锁事件发生

表 10-82：封锁状态清除寄存器（TZSTSCLR）位段定义

TZSTSCLR (Trip-Zone Status Clear Register) Offset: 0x94 Default: 0x00000000 Access: PWM -> TZSTSCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED						DBGCBC	DCBEVT1
15	14	13	12	11	10	9	8
DCAEVT1	CLKERRCBC	TZ4CBC	TZ3CBC	TZ2CBC	TZ1CBC	TZ0CBC	DBGOST
7	6	5	4	3	2	1	0
DCBEVT0	DCAEVT0	CLKERROST	TZ4OST	TZ3OST	TZ2OST	TZ1OST	TZ0OST

表 10-83：封锁状态清除寄存器（TZSTSCLR）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	DBGCBC	W1C	0x0	清除由 CPU 停止（因为调试或错误异常等原因引起）触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
16	DCBEVT1	W1C	0x0	清除由 DCBEVT1 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
15	DCAEVT1	W1C	0x0	清除由 DCAEVT1 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
14	CLKERRCBC	W1C	0x0	清除由时钟错误触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
13	TZ4CBC	W1C	0x0	清除由 TZ4 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
12	TZ3CBC	W1C	0x0	清除由 TZ3 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
11	TZ2CBC	W1C	0x0	清除由 TZ2 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
10	TZ1CBC	W1C	0x0	清除由 TZ1 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
9	TZ0CBC	W1C	0x0	清除由 TZ0 触发的周期封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零

位段	位段名	属性	复位值	描述
8	DBGOST	W1C	0x0	清除由 CPU 停止（因为调试或错误异常等原因引起）触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
7	DCBEVT0	W1C	0x0	清除由 DCBEVT0 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
6	DCAEVT0	W1C	0x0	清除由 DCAEVT0 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
5	CLKERROST	W1C	0x0	清除由时钟错误触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
4	TZ4OST	W1C	0x0	清除由 TZ4 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
3	TZ3OST	W1C	0x0	清除由 TZ3 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
2	TZ2OST	W1C	0x0	清除由 TZ2 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
1	TZ1OST	W1C	0x0	清除由 TZ1 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零
0	TZ0OST	W1C	0x0	清除由 TZ0 触发的一次封锁标志 0：写 0 无效，总是读回 0 1：写 1 清除锁存的标志 本位段自动清零

表 10-84：数字比较封锁事件选择寄存器（TZDCSEL）位段定义

TZDCSEL (Trip-Zone Digital Compare Event Select Register) Offset: 0x98 Default: 0x00000000
Access: PWM -> TZDCSEL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				DCBEVT1			DCBEVT0
7	6	5	4	3	2	1	0
DCBEVT0		DCAEVT1			DCAEVT0		

表 10-85：数字比较封锁事件选择寄存器（TZDCSEL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	DCBEVT1	RW	0x0	数字比较封锁事件 DCBEVT1 选择 000：禁用事件 001：DCBL 为低 010：DCBL 为高 011：DCBH 为低 100：DCBH 为高 101：DCBL 为低且 DCBH 为高 110：DCBL 为高且 DCBH 为低 111：DCBL 为高且 DCBH 为高
8:6	DCBEVT0	RW	0x0	数字比较封锁事件 DCBEVT0 选择 000：禁用事件 001：DCBL 为低 010：DCBL 为高 011：DCBH 为低 100：DCBH 为高 101：DCBL 为低且 DCBH 为高 110：DCBL 为高且 DCBH 为低 111：DCBL 为高且 DCBH 为高
5:3	DCAEVT1	RW	0x0	数字比较封锁事件 DCAEVT1 选择 000：禁用事件 001：DCAL 为低 010：DCAL 为高 011：DCAH 为低 100：DCAH 为高 101：DCAL 为低且 DCAH 为高 110：DCAL 为高且 DCAH 为低 111：DCAL 为高且 DCAH 为高

位段	位段名	属性	复位值	描述
2:0	DCAEVT0	RW	0x0	数字比较封锁事件 DCAEVT0 选择 000：禁用事件 001：DCAL 为低 010：DCAL 为高 011：DCAH 为低 100：DCAH 为高 101：DCAL 为低且 DCAH 为高 110：DCAL 为高且 DCAH 为低 111：DCAL 为高且 DCAH 为高

表 10-86：封锁 A 路输出控制寄存器（TZACTL）位段定义

TZACTL (Trip-Zone Output A Control Register) Offset: 0x9C Default: 0x00000000
Access: PWM -> TZACTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED						DCAEVT1D	
15	14	13	12	11	10	9	8
DCAEVT1D	DCAEVT1U			DCAEVT0D			DCAEVT0U
7	6	5	4	3	2	1	0
DCAEVT0U		TZAD			TZAU		

表 10-87：封锁 A 路输出控制寄存器（TZACTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17:15	DCAEVT1D	RW	0x0	DCAEVT1 发生且 TBCNT 递减计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高 011：A 路输出翻转 1xx：A 路输出维持原样不变
14:12	DCAEVT1U	RW	0x0	DCAEVT1 发生且 TBCNT 递增计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高 011：A 路输出翻转 1xx：A 路输出维持原样不变
11:9	DCAEVT0D	RW	0x0	DCAEVT0 发生且 TBCNT 递减计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高

位段	位段名	属性	复位值	描述
				011：A 路输出翻转 1xx：A 路输出维持原样不变
8:6	DCAEVT0U	RW	0x0	DCAEVT0 发生且 TBCNT 递增计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高 011：A 路输出翻转 1xx：A 路输出维持原样不变
5:3	TZAD	RW	0x0	TZ 事件发生且 TBCNT 递减计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高 011：A 路输出翻转 1xx：A 路输出维持原样不变
2:0	TZAU	RW	0x0	TZ 事件发生且 TBCNT 递增计数时的封锁输出 000：A 路输出高阻 001：A 路输出置低 010：A 路输出置高 011：A 路输出翻转 1xx：A 路输出维持原样不变

表 10-88：封锁 B 路输出控制寄存器（TZBCTL）位段定义

TZBCTL (Trip-Zone Ouput B Control Register)　Offset: 0xA0　Default: 0x00000000
Access: PWM -> TZBCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED						DCBEVT1D	
15	14	13	12	11	10	9	8
DCBEVT1D	DCBEVT1U			DCBEVT0D			DCBEVT0U
7	6	5	4	3	2	1	0
DCBEVT0U		TZBD			TZBU		

表 10-89：封锁 B 路输出控制寄存器（TZBCTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17:15	DCBEVT1D	RW	0x0	DCBEVT1 发生且 TBCNT 递减计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转

位段	位段名	属性	复位值	描述
				1xx：B 路输出维持原样不变
14:12	DCBEVT1U	RW	0x0	DCBEVT1 发生且 TBCNT 递增计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转 1xx：B 路输出维持原样不变
11:9	DCBEVT0D	RW	0x0	DCBEVT0 发生且 TBCNT 递减计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转 1xx：B 路输出维持原样不变
8:6	DCBEVT0U	RW	0x0	DCBEVT0 发生且 TBCNT 递增计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转 1xx：B 路输出维持原样不变
5:3	TZBD	RW	0x0	TZ 事件发生且 TBCNT 递减计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转 1xx：B 路输出维持原样不变
2:0	TZBU	RW	0x0	TZ 事件发生且 TBCNT 递增计数时的封锁输出 000：B 路输出高阻 001：B 路输出置低 010：B 路输出置高 011：B 路输出翻转 1xx：B 路输出维持原样不变

表 10-90：封锁中断标志寄存器（TZIF）位段定义

TZIF (Trip-Zone Flag Register) Offset: 0xA4 Default: 0x00000000 Access: PWM -> TZIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	INT	DCBEVT1	DCBEVT0	DCAEVT1	DCAEVT0	CBC	OST

表 10-91：封锁中断标志寄存器（TZIF）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	INT	RO	0x0	全局封锁中断标志 0：封锁中断未发生 1：封锁中断发生 该位段清零前封锁事件不会产生新的中断。
5	DCBEVT1	RO	0x0	DCBEVT1 封锁事件标志 0：DCBEVT1 事件未发生 1：DCBEVT1 事件发生。
4	DCBEVT0	RO	0x0	DCBEVT0 封锁事件标志 0：DCBEVT0 事件未发生 1：DCBEVT0 事件发生。
3	DCAEVT1	RO	0x0	DCAEVT1 封锁事件标志 0：DCAEVT1 事件未发生 1：DCAEVT1 事件发生。
2	DCAEVT0	RO	0x0	DCAEVT0 封锁事件标志 0：DCAEVT0 事件未发生 1：DCAEVT0 事件发生。
1	CBC	RO	0x0	周期封锁事件标志 0：周期封锁事件未发生 1：周期封锁事件发生。
0	OST	RO	0x0	一次封锁事件标志 0：一次封锁事件未发生 1：一次封锁事件发生。

表 10-92：封锁中断清除寄存器（TZIC）位段定义

TZIC (Trip-Zone Clear Register) Offset: 0xA8 Default: 0x00000000 Access: PWM -> TZIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							CBCCLRMODE
7	6	5	4	3	2	1	0
CBCCLRMODE	INT	DCBEVT1	DCBEVT0	DCAEVT1	DCAEVT0	CBC	OST

表 10-93：封锁中断清除寄存器（TZIC）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8:7	CBCCLRMODE	RW	0x0	周期封锁标志自动清除方式 00：TBCNT=0 时自动清除 01：TBCNT=TBPRD 时自动清除 10：TBCNT=0 或 TBCNT=TBPRD 时自动清除 11：不自动清除
6	INT	W1C	0x0	全局封锁中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除中断和 TZIF.INT 标志 本位段自动清零。
5	DCBEVT1	W1C	0x0	DCBEVT1 事件标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.DCBEVT1 标志 本位段自动清零。
4	DCBEVT0	W1C	0x0	DCBEVT0 事件标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.DCBEVT0 标志 本位段自动清零。
3	DCAEVT1	W1C	0x0	DCAEVT1 事件标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.DCAEVT1 标志 本位段自动清零。
2	DCAEVT0	W1C	0x0	DCAEVT0 事件标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.DCAEVT0 标志 本位段自动清零。

位段	位段名	属性	复位值	描述
1	CBC	W1C	0x0	周期封锁事件标志手动清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.CBC 标志 本位段自动清零。
0	OST	W1C	0x0	一次封锁事件标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 TZIF.OST 标志 本位段自动清零。

表 10-94：封锁中断使能寄存器（TZIE）位段定义

TZIE (Trip-Zone Interrupt Enable Register) Offset: 0xAC Default: 0x00000000 Access: PWM -> TZIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		DCBEVT1	DCBEVT0	DCAEVT1	DCAEVT0	CBC	OST

表 10-95：封锁中断使能寄存器（TZIE）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	DCBEVT1	RW	0x0	DCBEVT1 封锁事件中断使能 0：关闭中断 1：使能中断
4	DCBEVT0	RW	0x0	DCBEVT0 封锁事件中断使能 0：关闭中断 1：使能中断
3	DCAEVT1	RW	0x0	DCAEVT1 封锁事件中断使能 0：关闭中断 1：使能中断
2	DCAEVT0	RW	0x0	DCAEVT0 封锁事件中断使能 0：关闭中断 1：使能中断
1	CBC	RW	0x0	周期封锁事件中断使能 0：关闭中断 1：使能中断
0	OST	RW	0x0	一次封锁事件中断使能 0：关闭中断 1：使能中断

表 10-96：封锁中断强制寄存器（TZFRC）位段定义

TZFRC (Trip-Zone Force Register) Offset: 0xB0 Default: 0x00000000 Access: PWM -> TZFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		DCBEVT1	DCBEVT0	DCAEVT1	DCAEVT0	CBC	OST

表 10-97：封锁中断强制寄存器（TZFRC）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	DCBEVT1	W1S	0x0	软件强制 DCBEVT1 封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生 DCBEVT1 封锁并置位 TZIF.DCBEVT1 本位段自动清零
4	DCBEVT0	W1S	0x0	软件强制 DCBEVT0 封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生 DCBEVT0 封锁并置位 TZIF.DCBEVT0 本位段自动清零
3	DCAEVT1	W1S	0x0	软件强制 DCAEVT1 封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生 DCAEVT1 封锁并置位 TZIF.DCAEVT1 本位段自动清零
2	DCAEVT0	W1S	0x0	软件强制 DCAEVT0 封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生 DCAEVT0 封锁并置位 TZIF.DCAEVT0 本位段自动清零
1	CBC	W1S	0x0	软件强制周期封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生周期封锁并置位 TZIF.CBC 本位段自动清零
0	OST	W1S	0x0	软件强制一次封锁事件 0：写 0 无效，总是读回 0 1：写 1 产生一次封锁并置位 TZIF.OST 本位段自动清零

表 10-98：数字比较 DCAL 事件选择寄存器（DCALTRIPSEL）位段定义

DCALTRIPSEL (Digital Compare AL Trip Select Register) Offset: 0xB4 Default: 0x00000000 Access: PWM -> DCALTRIPSEL.all							
31	30	29	28	27	26	25	24
COMP7H	COMP7L	COMP6H	COMP6L	COMP5H	COMP5L	COMP4H	COMP4L
23	22	21	20	19	18	17	16
COMP3H	COMP3L	COMP2H	COMP2L	COMP1H	COMP1L	COMP0H	COMP0L
15	14	13	12	11	10	9	8
ADCPPU7TZ	ADCPPU6TZ	ADCPPU5TZ	ADCPPU4TZ	ADCPPU3TZ	ADCPPU2TZ	ADCPPU1TZ	ADCPPU0TZ
7	6	5	4	3	2	1	0
RESERVED			TZ4	TZ3	TZ2	TZ1	TZ0

表 10-99：数字比较 DCAL 事件选择寄存器（DCALTRIPSEL）位段描述

位段	位段名	属性	复位值	描述
31	COMP7H	RW	0x0	选中（逻辑或）COMP7H 为 DCAL 事件 0：不选中 1：选中
30	COMP7L	RW	0x0	选中（逻辑或）COMP7L 为 DCAL 事件 0：不选中 1：选中
29	COMP6H	RW	0x0	选中（逻辑或）COMP6H 为 DCAL 事件 0：不选中 1：选中
28	COMP6L	RW	0x0	选中（逻辑或）COMP6L 为 DCAL 事件 0：不选中 1：选中
27	COMP5H	RW	0x0	选中（逻辑或）COMP5H 为 DCAL 事件 0：不选中 1：选中
26	COMP5L	RW	0x0	选中（逻辑或）COMP5L 为 DCAL 事件 0：不选中 1：选中
25	COMP4H	RW	0x0	选中（逻辑或）COMP4H 为 DCAL 事件 0：不选中 1：选中
24	COMP4L	RW	0x0	选中（逻辑或）COMP4L 为 DCAL 事件 0：不选中 1：选中
23	COMP3H	RW	0x0	选中（逻辑或）COMP3H 为 DCAL 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
22	COMP3L	RW	0x0	选中（逻辑或）COMP3L 为 DCAL 事件 0：不选中 1：选中
21	COMP2H	RW	0x0	选中（逻辑或）COMP2H 为 DCAL 事件 0：不选中 1：选中
20	COMP2L	RW	0x0	选中（逻辑或）COMP2L 为 DCAL 事件 0：不选中 1：选中
19	COMP1H	RW	0x0	选中（逻辑或）COMP1H 为 DCAL 事件 0：不选中 1：选中
18	COMP1L	RW	0x0	选中（逻辑或）COMP1L 为 DCAL 事件 0：不选中 1：选中
17	COMP0H	RW	0x0	选中（逻辑或）COMP0H 为 DCAL 事件 0：不选中 1：选中
16	COMP0L	RW	0x0	选中（逻辑或）COMP0L 为 DCAL 事件 0：不选中 1：选中
15	ADCPPU7TZ	RW	0x0	选中（逻辑或）ADCPPU7TZ 为 DCAL 事件 0：不选中 1：选中
14	ADCPPU6TZ	RW	0x0	选中（逻辑或）ADCPPU6TZ 为 DCAL 事件 0：不选中 1：选中
13	ADCPPU5TZ	RW	0x0	选中（逻辑或）ADCPPU5TZ 为 DCAL 事件 0：不选中 1：选中
12	ADCPPU4TZ	RW	0x0	选中（逻辑或）ADCPPU4TZ 为 DCAL 事件 0：不选中 1：选中
11	ADCPPU3TZ	RW	0x0	选中（逻辑或）ADCPPU3TZ 为 DCAL 事件 0：不选中 1：选中
10	ADCPPU2TZ	RW	0x0	选中（逻辑或）ADCPPU2TZ 为 DCAL 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
9	ADCPPU1TZ	RW	0x0	选中（逻辑或）ADCPPU1TZ 为 DCAL 事件 0：不选中 1：选中
8	ADCPPU0TZ	RW	0x0	选中（逻辑或）ADCPPU0TZ 为 DCAL 事件 0：不选中 1：选中
7:5	RESERVED_7_5	RO	0x0	保留
4	TZ4	RW	0x0	选中（逻辑或）TZ4 为 DCAL 事件 0：不选中 1：选中
3	TZ3	RW	0x0	选中（逻辑或）TZ3 为 DCAL 事件 0：不选中 1：选中
2	TZ2	RW	0x0	选中（逻辑或）TZ2 为 DCAL 事件 0：不选中 1：选中
1	TZ1	RW	0x0	选中（逻辑或）TZ1 为 DCAL 事件 0：不选中 1：选中
0	TZ0	RW	0x0	选中（逻辑或）TZ0 为 DCAL 事件 0：不选中 1：选中

表 10-100：数字比较 DCAH 事件选择寄存器（DCAHTRIPSEL）位段定义

DCAHTRIPSEL (Digital Compare AH Trip Select Register)　Offset: 0xB8　Default: 0x00000000
Access: PWM -> DCAHTRIPSEL.all

31	30	29	28	27	26	25	24
COMP7H	COMP7L	COMP6H	COMP6L	COMP5H	COMP5L	COMP4H	COMP4L
23	22	21	20	19	18	17	16
COMP3H	COMP3L	COMP2H	COMP2L	COMP1H	COMP1L	COMP0H	COMP0L
15	14	13	12	11	10	9	8
ADCPPU7TZ	ADCPPU6TZ	ADCPPU5TZ	ADCPPU4TZ	ADCPPU3TZ	ADCPPU2TZ	ADCPPU1TZ	ADCPPU0TZ
7	6	5	4	3	2	1	0
RESERVED			TZ4	TZ3	TZ2	TZ1	TZ0

表 10-101：数字比较 DCAH 事件选择寄存器（DCAHTRIPSEL）位段描述

位段	位段名	属性	复位值	描述
31	COMP7H	RW	0x0	选中（逻辑或）COMP7H 为 DCAH 事件 0：不选中 1：选中
30	COMP7L	RW	0x0	选中（逻辑或）COMP7L 为 DCAH 事件 0：不选中 1：选中
29	COMP6H	RW	0x0	选中（逻辑或）COMP6H 为 DCAH 事件 0：不选中 1：选中
28	COMP6L	RW	0x0	选中（逻辑或）COMP6L 为 DCAH 事件 0：不选中 1：选中
27	COMP5H	RW	0x0	选中（逻辑或）COMP5H 为 DCAH 事件 0：不选中 1：选中
26	COMP5L	RW	0x0	选中（逻辑或）COMP5L 为 DCAH 事件 0：不选中 1：选中
25	COMP4H	RW	0x0	选中（逻辑或）COMP4H 为 DCAH 事件 0：不选中 1：选中
24	COMP4L	RW	0x0	选中（逻辑或）COMP4L 为 DCAH 事件 0：不选中 1：选中
23	COMP3H	RW	0x0	选中（逻辑或）COMP3H 为 DCAH 事件 0：不选中 1：选中
22	COMP3L	RW	0x0	选中（逻辑或）COMP3L 为 DCAH 事件 0：不选中 1：选中
21	COMP2H	RW	0x0	选中（逻辑或）COMP2H 为 DCAH 事件 0：不选中 1：选中
20	COMP2L	RW	0x0	选中（逻辑或）COMP2L 为 DCAH 事件 0：不选中 1：选中
19	COMP1H	RW	0x0	选中（逻辑或）COMP1H 为 DCAH 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
18	COMP1L	RW	0x0	选中（逻辑或）COMP1L 为 DCAH 事件 0：不选中 1：选中
17	COMP0H	RW	0x0	选中（逻辑或）COMP0H 为 DCAH 事件 0：不选中 1：选中
16	COMP0L	RW	0x0	选中（逻辑或）COMP0L 为 DCAH 事件 0：不选中 1：选中
15	ADCPPU7TZ	RW	0x0	选中（逻辑或）ADCPPU7TZ 为 DCAH 事件 0：不选中 1：选中
14	ADCPPU6TZ	RW	0x0	选中（逻辑或）ADCPPU6TZ 为 DCAH 事件 0：不选中 1：选中
13	ADCPPU5TZ	RW	0x0	选中（逻辑或）ADCPPU5TZ 为 DCAH 事件 0：不选中 1：选中
12	ADCPPU4TZ	RW	0x0	选中（逻辑或）ADCPPU4TZ 为 DCAH 事件 0：不选中 1：选中
11	ADCPPU3TZ	RW	0x0	选中（逻辑或）ADCPPU3TZ 为 DCAH 事件 0：不选中 1：选中
10	ADCPPU2TZ	RW	0x0	选中（逻辑或）ADCPPU2TZ 为 DCAH 事件 0：不选中 1：选中
9	ADCPPU1TZ	RW	0x0	选中（逻辑或）ADCPPU1TZ 为 DCAH 事件 0：不选中 1：选中
8	ADCPPU0TZ	RW	0x0	选中（逻辑或）ADCPPU0TZ 为 DCAH 事件 0：不选中 1：选中
7:5	RESERVED_7_5	RO	0x0	保留
4	TZ4	RW	0x0	选中（逻辑或）TZ4 为 DCAH 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
3	TZ3	RW	0x0	选中（逻辑或）TZ3 为 DCAH 事件 0：不选中 1：选中
2	TZ2	RW	0x0	选中（逻辑或）TZ2 为 DCAH 事件 0：不选中 1：选中
1	TZ1	RW	0x0	选中（逻辑或）TZ1 为 DCAH 事件 0：不选中 1：选中
0	TZ0	RW	0x0	选中（逻辑或）TZ0 为 DCAH 事件 0：不选中 1：选中

表 10-102：数字比较 DCBL 事件选择寄存器（DCBLTRIPSEL）位段定义

DCBLTRIPSEL (Digital Compare BL Trip Select Register)　Offset: 0xBC　Default: 0x00000000
Access: PWM -> DCBLTRIPSEL.all

31	30	29	28	27	26	25	24
COMP7H	COMP7L	COMP6H	COMP6L	COMP5H	COMP5L	COMP4H	COMP4L
23	22	21	20	19	18	17	16
COMP3H	COMP3L	COMP2H	COMP2L	COMP1H	COMP1L	COMP0H	COMP0L
15	14	13	12	11	10	9	8
ADCPPU7TZ	ADCPPU6TZ	ADCPPU5TZ	ADCPPU4TZ	ADCPPU3TZ	ADCPPU2TZ	ADCPPU1TZ	ADCPPU0TZ
7	6	5	4	3	2	1	0
RESERVED			TZ4	TZ3	TZ2	TZ1	TZ0

表 10-103：数字比较 DCBL 事件选择寄存器（DCBLTRIPSEL）位段描述

位段	位段名	属性	复位值	描述
31	COMP7H	RW	0x0	选中（逻辑或）COMP7H 为 DCBL 事件 0：不选中 1：选中
30	COMP7L	RW	0x0	选中（逻辑或）COMP7L 为 DCBL 事件 0：不选中 1：选中
29	COMP6H	RW	0x0	选中（逻辑或）COMP6H 为 DCBL 事件 0：不选中 1：选中
28	COMP6L	RW	0x0	选中（逻辑或）COMP6L 为 DCBL 事件 0：不选中 1：选中
27	COMP5H	RW	0x0	选中（逻辑或）COMP5H 为 DCBL 事件

位段	位段名	属性	复位值	描述
				0：不选中 1：选中
26	COMP5L	RW	0x0	选中（逻辑或）COMP5L 为 DCBL 事件 0：不选中 1：选中
25	COMP4H	RW	0x0	选中（逻辑或）COMP4H 为 DCBL 事件 0：不选中 1：选中
24	COMP4L	RW	0x0	选中（逻辑或）COMP4L 为 DCBL 事件 0：不选中 1：选中
23	COMP3H	RW	0x0	选中（逻辑或）COMP3H 为 DCBL 事件 0：不选中 1：选中
22	COMP3L	RW	0x0	选中（逻辑或）COMP3L 为 DCBL 事件 0：不选中 1：选中
21	COMP2H	RW	0x0	选中（逻辑或）COMP2H 为 DCBL 事件 0：不选中 1：选中
20	COMP2L	RW	0x0	选中（逻辑或）COMP2L 为 DCBL 事件 0：不选中 1：选中
19	COMP1H	RW	0x0	选中（逻辑或）COMP1H 为 DCBL 事件 0：不选中 1：选中
18	COMP1L	RW	0x0	选中（逻辑或）COMP1L 为 DCBL 事件 0：不选中 1：选中
17	COMP0H	RW	0x0	选中（逻辑或）COMP0H 为 DCBL 事件 0：不选中 1：选中
16	COMP0L	RW	0x0	选中（逻辑或）COMP0L 为 DCBL 事件 0：不选中 1：选中
15	ADCPPU7TZ	RW	0x0	选中（逻辑或）ADCPPU7TZ 为 DCBL 事件 0：不选中 1：选中
14	ADCPPU6TZ	RW	0x0	选中（逻辑或）ADCPPU6TZ 为 DCBL 事件

位段	位段名	属性	复位值	描述
				0：不选中 1：选中
13	ADCPPU5TZ	RW	0x0	选中（逻辑或）ADCPPU5TZ 为 DCBL 事件 0：不选中 1：选中
12	ADCPPU4TZ	RW	0x0	选中（逻辑或）ADCPPU4TZ 为 DCBL 事件 0：不选中 1：选中
11	ADCPPU3TZ	RW	0x0	选中（逻辑或）ADCPPU3TZ 为 DCBL 事件 0：不选中 1：选中
10	ADCPPU2TZ	RW	0x0	选中（逻辑或）ADCPPU2TZ 为 DCBL 事件 0：不选中 1：选中
9	ADCPPU1TZ	RW	0x0	选中（逻辑或）ADCPPU1TZ 为 DCBL 事件 0：不选中 1：选中
8	ADCPPU0TZ	RW	0x0	选中（逻辑或）ADCPPU0TZ 为 DCBL 事件 0：不选中 1：选中
7:5	RESERVED_7_5	RO	0x0	保留
4	TZ4	RW	0x0	选中（逻辑或）TZ4 为 DCBL 事件 0：不选中 1：选中
3	TZ3	RW	0x0	选中（逻辑或）TZ3 为 DCBL 事件 0：不选中 1：选中
2	TZ2	RW	0x0	选中（逻辑或）TZ2 为 DCBL 事件 0：不选中 1：选中
1	TZ1	RW	0x0	选中（逻辑或）TZ1 为 DCBL 事件 0：不选中 1：选中
0	TZ0	RW	0x0	选中（逻辑或）TZ0 为 DCBL 事件 0：不选中 1：选中

表 10-104：数字比较 DCBH 事件选择寄存器（DCBHTRIPSEL）位段定义

DCBHTRIPSEL (Digital Compare BH Trip Select Register) Offset: 0xC0 Default: 0x00000000 Access: PWM -> DCBHTRIPSEL.all							
31	30	29	28	27	26	25	24
COMP7H	COMP7L	COMP6H	COMP6L	COMP5H	COMP5L	COMP4H	COMP4L
23	22	21	20	19	18	17	16
COMP3H	COMP3L	COMP2H	COMP2L	COMP1H	COMP1L	COMP0H	COMP0L
15	14	13	12	11	10	9	8
ADCPPU7TZ	ADCPPU6TZ	ADCPPU5TZ	ADCPPU4TZ	ADCPPU3TZ	ADCPPU2TZ	ADCPPU1TZ	ADCPPU0TZ
7	6	5	4	3	2	1	0
RESERVED			TZ4	TZ3	TZ2	TZ1	TZ0

表 10-105：数字比较 DCBH 事件选择寄存器（DCBHTRIPSEL）位段描述

位段	位段名	属性	复位值	描述
31	COMP7H	RW	0x0	选中（逻辑或）COMP7H 为 DCBH 事件 0：不选中 1：选中
30	COMP7L	RW	0x0	选中（逻辑或）COMP7L 为 DCBH 事件 0：不选中 1：选中
29	COMP6H	RW	0x0	选中（逻辑或）COMP6H 为 DCBH 事件 0：不选中 1：选中
28	COMP6L	RW	0x0	选中（逻辑或）COMP6L 为 DCBH 事件 0：不选中 1：选中
27	COMP5H	RW	0x0	选中（逻辑或）COMP5H 为 DCBH 事件 0：不选中 1：选中
26	COMP5L	RW	0x0	选中（逻辑或）COMP5L 为 DCBH 事件 0：不选中 1：选中
25	COMP4H	RW	0x0	选中（逻辑或）COMP4H 为 DCBH 事件 0：不选中 1：选中
24	COMP4L	RW	0x0	选中（逻辑或）COMP4L 为 DCBH 事件 0：不选中 1：选中
23	COMP3H	RW	0x0	选中（逻辑或）COMP3H 为 DCBH 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
22	COMP3L	RW	0x0	选中（逻辑或）COMP3L 为 DCBH 事件 0：不选中 1：选中
21	COMP2H	RW	0x0	选中（逻辑或）COMP2H 为 DCBH 事件 0：不选中 1：选中
20	COMP2L	RW	0x0	选中（逻辑或）COMP2L 为 DCBH 事件 0：不选中 1：选中
19	COMP1H	RW	0x0	选中（逻辑或）COMP1H 为 DCBH 事件 0：不选中 1：选中
18	COMP1L	RW	0x0	选中（逻辑或）COMP1L 为 DCBH 事件 0：不选中 1：选中
17	COMP0H	RW	0x0	选中（逻辑或）COMP0H 为 DCBH 事件 0：不选中 1：选中
16	COMP0L	RW	0x0	选中（逻辑或）COMP0L 为 DCBH 事件 0：不选中 1：选中
15	ADCPPU7TZ	RW	0x0	选中（逻辑或）ADCPPU7TZ 为 DCBH 事件 0：不选中 1：选中
14	ADCPPU6TZ	RW	0x0	选中（逻辑或）ADCPPU6TZ 为 DCBH 事件 0：不选中 1：选中
13	ADCPPU5TZ	RW	0x0	选中（逻辑或）ADCPPU5TZ 为 DCBH 事件 0：不选中 1：选中
12	ADCPPU4TZ	RW	0x0	选中（逻辑或）ADCPPU4TZ 为 DCBH 事件 0：不选中 1：选中
11	ADCPPU3TZ	RW	0x0	选中（逻辑或）ADCPPU3TZ 为 DCBH 事件 0：不选中 1：选中
10	ADCPPU2TZ	RW	0x0	选中（逻辑或）ADCPPU2TZ 为 DCBH 事件 0：不选中 1：选中

位段	位段名	属性	复位值	描述
9	ADCPPU1TZ	RW	0x0	选中（逻辑或）ADCPPU1TZ 为 DCBH 事件 0：不选中 1：选中
8	ADCPPU0TZ	RW	0x0	选中（逻辑或）ADCPPU0TZ 为 DCBH 事件 0：不选中 1：选中
7:5	RESERVED_7_5	RO	0x0	保留
4	TZ4	RW	0x0	选中（逻辑或）TZ4 为 DCBH 事件 0：不选中 1：选中
3	TZ3	RW	0x0	选中（逻辑或）TZ3 为 DCBH 事件 0：不选中 1：选中
2	TZ2	RW	0x0	选中（逻辑或）TZ2 为 DCBH 事件 0：不选中 1：选中
1	TZ1	RW	0x0	选中（逻辑或）TZ1 为 DCBH 事件 0：不选中 1：选中
0	TZ0	RW	0x0	选中（逻辑或）TZ0 为 DCBH 事件 0：不选中 1：选中

表 10-106：数字比较 A 控制寄存器（DCACTL）位段定义

DCACTL (Digital Compare A Control Register)　Offset: 0xC4　Default: 0x00000000
Access: PWM -> DCACTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		EVT1SRCASYNC	EVT1SRCSEL	EVT0SYNCE	EVT0SOCE	EVT0SRCASYNC	EVT0SRCSEL

表 10-107：数字比较 A 控制寄存器（DCACTL）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	EVT1SRCASYNC	RW	0x0	DCAEVT1 源信号通路选择 0：同步通路 1：异步通路
4	EVT1SRCSEL	RW	0x0	DCAEVT1 最终信号选择 0：选择 DCAEVT1 1：选择 DCEVTFILT
3	EVT0SYNCE	RW	0x0	DCAEVT0 SYNC 信号产生使能 0：不产生 DCAEVT0 SYNC 信号 1：产生 DCAEVT0 SYNC 信号
2	EVT0SOCE	RW	0x0	DCAEVT0 SOC 信号产生使能 0：不产生 DCAEVT0 SOC 信号 1：产生 DCAEVT0 SOC 信号
1	EVT0SRCASYNC	RW	0x0	DCAEVT0 源信号通路选择 0：同步通路 1：异步通路
0	EVT0SRCSEL	RW	0x0	DCAEVT0 最终信号选择 0：选择 DCAEVT0 1：选择 DCEVTFILT

表 10-108：数字比较 B 控制寄存器（DCBCTL）位段定义

DCBCTL (Digital Compare B Control Register) Offset: 0xC8 Default: 0x00000000 Access: PWM -> DCBCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		EVT1SRCASYNC	EVT1SRCSEL	EVT0SYNCE	EVT0SOCE	EVT0SRCASYNC	EVT0SRCSEL

表 10-109：数字比较 B 控制寄存器（DCBCTL）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	EVT1SRCASYNC	RW	0x0	DCBEVT1 源信号通路选择 0：同步通路 1：异步通路
4	EVT1SRCSEL	RW	0x0	DCBEVT1 最终信号选择 0：选择 DCBEVT1 1：选择 DCEVTFILT
3	EVT0SYNCE	RW	0x0	DCBEVT0 SYNC 信号产生使能 0：不产生 DCBEVT0 SYNC 信号 1：产生 DCBEVT0 SYNC 信号
2	EVT0SOCE	RW	0x0	DCBEVT0 SOC 信号产生使能 0：不产生 DCBEVT0 SOC 信号 1：产生 DCBEVT0 SOC 信号
1	EVT0SRCASYNC	RW	0x0	DCBEVT0 源信号通路选择 0：同步通路 1：异步通路
0	EVT0SRCSEL	RW	0x0	DCBEVT0 最终信号选择 0：选择 DCBEVT0 1：选择 DCEVTFILT

表 10-110：数字比较消隐滤波控制寄存器（DCFCTL）位段定义

DCFCTL (Digital Compare Filter Register) Offset: 0xCC Default: 0x00000010 Access: PWM -> DCFCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED		WIN7EN	WIN6EN	WIN5EN	WIN4EN	WIN3EN	WIN2EN
7	6	5	4	3	2	1	0
WIN1EN	WIN0EN	PULSESEL		BLANKINV	BLANKEN	SRCSEL	

表 10-111：数字比较消隐滤波控制寄存器（DCFCTL）位段描述

位段	位段名	属性	复位值	描述
31:14	RESERVED_31_14	RO	0x0	保留
13	WIN7EN	RW	0x0	使能 PWM7 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
12	WIN6EN	RW	0x0	使能 PWM6 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
11	WIN5EN	RW	0x0	使能 PWM5 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
10	WIN4EN	RW	0x0	使能 PWM4 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
9	WIN3EN	RW	0x0	使能 PWM3 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
8	WIN2EN	RW	0x0	使能 PWM2 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
7	WIN1EN	RW	0x0	使能 PWM1 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
6	WIN0EN	RW	0x0	使能 PWM0 产生的窗口对本级 PWM 信号消隐 0：关闭 1：使能（逻辑或）
5:4	PULSESEL	RW	0x1	消隐和捕捉脉冲起始对齐参考点选择

位段	位段名	属性	复位值	描述
				00：对齐到 TBCNT=0 01：对齐到 TBCNT=TBPRD 10：对齐到 TBCNT=0 或 TBCNT=TBPRD 11：无效选项
3	BLANKINV	RW	0x0	消隐窗口反相 0：保持原始效应窗口极性 1：消隐窗口取反
2	BLANKEN	RW	0x0	消隐窗口使能 0：关闭消隐窗口 1：使能消隐窗口
1:0	SRCSEL	RW	0x0	DCEVTFILT 消隐滤波信号源选择 00：对 DCAEVT0 消隐滤波 01：对 DCAEVT1 消隐滤波 10：对 DCBEVT0 消隐滤波 11：对 DCBEVT1 消隐滤波

表 10-112：数字比较消隐窗口偏移量寄存器（DCFOFFSET）位段定义

DCFOFFSET (Digital Compare Filter Offset Register) Offset: 0xD0 Default: 0x00000000
Access: PWM -> DCFOFFSET.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-113：数字比较消隐窗口偏移量寄存器（DCFOFFSET）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	消隐窗口偏移量 定义了从 DCFCTL.PULSESEL 指定的脉冲到消隐窗口开始的偏移量（以时基时钟 TBCLK 的周期数为单位）。

表 10-114：数字比较消隐窗口偏移计数值寄存器（DCFOFFSETCNT）位段定义

DCFOFFSETCNT (Digital Compare Filter Offset Counter Register) Offset: 0xD4 Default: 0x00000000 Access: PWM -> DCFOFFSETCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-115：数字比较消隐窗口偏移计数值寄存器（DCFOFFSETCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	消隐窗口偏移计数器 本位段只读，它记录了当前偏移计数值。由 DCFCTL.PULSESEL 指定的事件发生后，该计数器从 DCFOFFSET 开始递减计数一直到 0，然后维持，直到 DCFCTL.PULSESEL 指定的事件再次发生后，该计数器又从 DCFOFFSET 开始递减计数。

表 10-116：数字比较消隐窗口大小寄存器（DCFWINDOW）位段定义

DCFWINDOW (Digital Compare Filter Window Register) Offset: 0xD8 Default: 0x00000000 Access: PWM -> DCFWINDOW.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-117：数字比较消隐窗口大小寄存器（DCFWINDOW）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	以 TBCLK 周期度量的消隐窗口大小 当消隐窗口偏移计数器计到 0 后，消隐窗口开启，消隐窗口大小计数器开始递减计数，待递减计数到 0 后消隐窗口关闭。若消隐窗口开启时偏移计数器又计数到 0，则立刻重新开启新的消隐窗口。 消隐窗口允许跨 PWM 时基波形周期边界。 当本位段为 0 时，禁用消隐窗口。

表 10-118：数字比较消隐窗口大小计数寄存器（DCFWINDOWCNT）位段定义

DCFWINDOWCNT (Digital Compare Filter Window Counter Register) Offset: 0xDC Default: 0x00000000 Access: PWM -> DCFWINDOWCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-119：数字比较消隐窗口大小计数寄存器（DCFWINDOWCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	消隐窗口大小计数器 本位段只读，它记录了当前消隐窗口大小计数值。消隐窗口偏移计数器计数到 0 后，该计数器从 DCFWINDOW 开始递减计数，一直到 0 并停止，直到消隐窗口偏移计数器再次递减计数到 0，该计数器又再次从 DCFWINDOW 开始递减计数。

表 10-120：数字比较时基计数捕获控制寄存器（DCCAPCTL）位段定义

DCCAPCTL (Digital Compare Capture Control Register) Offset: 0xE0 Default: 0x00000000 Access: PWM -> DCCAPCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				CAPSTS	CAPCLR	DIRECT	CAPEN

表 10-121：数字比较时基计数捕获控制寄存器（DCCAPCTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	CAPSTS	RO	0x0	锁存的捕获状态 0：数字比较时基计数捕获未发生 1：数字比较时基计数捕获发生
2	CAPCLR	W1C	0x0	清除锁存的捕获状态 0：写 0 无效，总是读回 0 1：写 1 清除锁存的状态，该位段自动清零
1	DIRECT	RW	0x0	DCCAP 捕获值更新方式 0：影子模式 由 DCFCTL.PULSESEL 指定的 TBCNT=TBPRD 或 TBCNT=0 事件发生时，将更新 DCCAP 捕获的最新值更新到其影子寄存器。读取 DCCAP 总是返回影子值。 1：直接模式 读取 DCCAP 寄存器时总是返回最新有效值。
0	CAPEN	RW	0x0	TBCNT 时基计数捕获使能 0：关闭捕获 1：使能捕获

表 10-122：数字比较时基计数捕获值寄存器（DCCAP）位段定义

DCCAP (Digital Compare Counter Capture Register) Offset: 0xE4 Default: 0x00000000 Access: PWM -> DCCAP.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 10-123：数字比较时基计数捕获值寄存器（DCCAP）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	DCEVTFLT 上升沿时捕获的时基计数器的值 在影子模式下，捕获的值在 DCFCTL.PULSESEL 指定的 TBCNT=TBPRD 或 TBCNT=0 事件时更新到影子寄存器。读取 DCCAP 寄存器总是返回影子值。 在直接模式下，读取 DCCAP 寄存器总是返回最新捕获的有效值。

表 10-124：事件触发控制寄存器（ETCTL）位段定义

ETCTL (Event-Trigger Control Register) Offset: 0xE8 Default: 0x0007B16F Access: PWM -> ETCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED				SOCCEN	SOCCSEL		
15	14	13	12	11	10	9	8
SOCCSEL	SOCBEN	SOCBSEL				SOCAEN	SOCASEL
7	6	5	4	3	2	1	0
SOCASEL			INTEN	INTSEL			

表 10-125：事件触发控制寄存器（ETCTL）位段描述

位段	位段名	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19	SOCCEN	RW	0x0	ADC 采样开始脉冲 C（SOCC）使能 0：关闭 1：使能
18:15	SOCCSEL	RW	0xF	SOCC 事件选择 0000：TBCNT=0 时触发事件 0001：TBCNT=TBPRD 时触发事件 0010：TBCNT=0 或 TBCNT=TBPRD 时触发事件 0011：TBCNT=CMPA 且 TBCNT 递增计数时触发 0100：TBCNT=CMPA 且 TBCNT 递减计数时触发 0101：TBCNT=CMPB 且 TBCNT 递增计数时触发 0110：TBCNT=CMPB 且 TBCNT 递减计数时触发 0111：TBCNT=CMPC 且 TBCNT 递增计数时触发 1000：TBCNT=CMPC 且 TBCNT 递减计数时触发 1001：TBCNT=CMPD 且 TBCNT 递增计数时触发 1010：TBCNT=CMPD 且 TBCNT 递减计数时触发 1011：DCAEVT0.SOC 事件触发 1100：DCBEVT0.SOC 事件触发 其他：不产生（禁止）SOCC 事件
14	SOCBEN	RW	0x0	ADC 采样开始脉冲 B（SOCB）使能 0：关闭 1：使能
13:10	SOCBSEL	RW	0xC	SOCB 事件选择 0000：TBCNT=0 时触发事件 0001：TBCNT=TBPRD 时触发事件 0010：TBCNT=0 或 TBCNT=TBPRD 时触发事件 0011：TBCNT=CMPA 且 TBCNT 递增计数时触发 0100：TBCNT=CMPA 且 TBCNT 递减计数时触发 0101：TBCNT=CMPB 且 TBCNT 递增计数时触发 0110：TBCNT=CMPB 且 TBCNT 递减计数时触发 0111：TBCNT=CMPC 且 TBCNT 递增计数时触发 1000：TBCNT=CMPC 且 TBCNT 递减计数时触发 1001：TBCNT=CMPD 且 TBCNT 递增计数时触发 1010：TBCNT=CMPD 且 TBCNT 递减计数时触发 1011：DCAEVT0.SOC 事件触发 1100：DCBEVT0.SOC 事件触发 其他：不产生（禁止）SOCB 事件

位段	位段名	属性	复位值	描述
9	SOCAEN	RW	0x0	ADC 采样开始脉冲 A（SOCA）使能 0：关闭 1：使能
8:5	SOCASEL	RW	0xB	SOCA 事件选择 0000：TBCNT=0 时触发事件 0001：TBCNT=TBPRD 时触发事件 0010：TBCNT=0 或 TBCNT=TBPRD 时触发事件 0011：TBCNT=CMPA 且 TBCNT 递增计数时触发 0100：TBCNT=CMPA 且 TBCNT 递减计数时触发 0101：TBCNT=CMPB 且 TBCNT 递增计数时触发 0110：TBCNT=CMPB 且 TBCNT 递减计数时触发 0111：TBCNT=CMPC 且 TBCNT 递增计数时触发 1000：TBCNT=CMPC 且 TBCNT 递减计数时触发 1001：TBCNT=CMPD 且 TBCNT 递增计数时触发 1010：TBCNT=CMPD 且 TBCNT 递减计数时触发 1011：DCAEVT0.SOC 事件触发 1100：DCBEVT0.SOC 事件触发 其他：不产生（禁止）SOCA 事件
4	INTEN	RW	0x0	中断脉冲使能 0：关闭 1：使能
3:0	INTSEL	RW	0xF	中断事件选择 0000：TBCNT=0 时触发事件 0001：TBCNT=TBPRD 时触发事件 0010：TBCNT=0 或 TBCNT=TBPRD 时触发事件 0011：TBCNT=CMPA 且 TBCNT 递增计数时触发 0100：TBCNT=CMPA 且 TBCNT 递减计数时触发 0101：TBCNT=CMPB 且 TBCNT 递增计数时触发 0110：TBCNT=CMPB 且 TBCNT 递减计数时触发 0111：TBCNT=CMPC 且 TBCNT 递增计数时触发 1000：TBCNT=CMPC 且 TBCNT 递减计数时触发 1001：TBCNT=CMPD 且 TBCNT 递增计数时触发 1010：TBCNT=CMPD 且 TBCNT 递减计数时触发 1011：DCAEVT0.SOC 事件触发 1100：DCBEVT0.SOC 事件触发 其他：不产生（禁止）中断事件

表 10-126：事件触发预分频寄存器（ETPS）位段定义

ETPS (Event-Trigger Prescale Register) Offset: 0xEC Default: 0x00000000 Access: PWM -> ETPS.all							
31	30	29	28	27	26	25	24
SOCCCNT				SOCCPRD			
23	22	21	20	19	18	17	16
SOCBCNT				SOCBPRD			
15	14	13	12	11	10	9	8
SOCACNT				SOCAPRD			
7	6	5	4	3	2	1	0
INTCNT				INTPRD			

表 10-127：事件触发预分频寄存器（ETPS）位段描述

位段	位段名	属性	复位值	描述
31:28	SOCCCNT	RO	0x0	SOCC 事件计数 该位段记录了 ETCTL.SOCCSEL 指定的 SOCC 事件发生的次数。当 SOCC 脉冲产生时，该位段自动清零。 0000：事件未发生 0001：事件发生了 1 次 0010：事件发生了 2 次 …… 1111：事件发生了 15 次
27:24	SOCCPRD	RW	0x0	SOCC 事件周期 当 ETCTL.SOCCEN=1 且 SOCCCNT =SOCCPRD 时产生 SOCC 脉冲。 0000：禁用 SOCC 计数器且不产生 SOCC 事件 0001：当 ETPS.SOCCCNT=1（每次）时产生 SOCC 脉冲 0010：当 ETPS.SOCCCNT=2（每 2 次）时产生 SOCC 脉冲 …… 1111：当 ETPS.SOCCCNT=15（每 15 次）时产生 SOCC 脉冲
23:20	SOCBCNT	RO	0x0	SOCB 事件计数 该位段记录了 ETCTL.SOCBSEL 指定的 SOCB 事件发生的次数。当 SOCB 脉冲产生时，该位段自动清零。 0000：事件未发生 0001：事件发生了 1 次 0010：事件发生了 2 次 …… 1111：事件发生了 15 次
19:16	SOCBPRD	RW	0x0	SOCB 事件周期 当 ETCTL.SOCBEN=1 且 SOCBCNT =SOCBPRD 时产生 SOCB 脉冲。 0000：禁用 SOCB 计数器且不产生 SOCB 事件

位段	位段名	属性	复位值	描述
				0001：当 ETPS.SOCBCNT=1（每次）时产生 SOCB 脉冲 0010：当 ETPS.SOCBCNT=2（每 2 次）时产生 SOCB 脉冲 …… 1111：当 ETPS.SOCBCNT=15（每 15 次）时产生 SOCB 脉冲
15:12	SOCACNT	RO	0x0	SOCA 事件计数 该位段记录了 ETCTL.SOCASEL 指定的 SOCA 事件发生的次数。当 SOCA 脉冲产生时，该位段自动清零。 0000：事件未发生 0001：事件发生了 1 次 0010：事件发生了 2 次 …… 1111：事件发生了 15 次
11:8	SOCAPRD	RW	0x0	SOCA 事件周期 当 ETCTL.SOCAEN=1 且 SOCACNT =SOCAPRD 时产生 SOCA 脉冲。 0000：禁用 SOCA 计数器且不产生 SOCA 事件 0001：当 ETPS.SOCACNT=1（每次）时产生 SOCA 脉冲 0010：当 ETPS.SOCACNT=2（每 2 次）时产生 SOCA 脉冲 …… 1111：当 ETPS.SOCACNT=15（每 15 次）时产生 SOCA 脉冲
7:4	INTCNT	RO	0x0	中断事件计数 该位段记录了 ETCTL.INTSEL 指定的中断事件发生的次数。当中断脉冲产生时，该位段自动清零。 0000：事件未发生 0001：事件发生了 1 次 0010：事件发生了 2 次 …… 1111：事件发生了 15 次
3:0	INTPRD	RW	0x0	中断事件周期 当 ETCTL.INTEN=1 且 INTCNT =INTPRD 时产生中断脉冲。 0000：禁用中断计数器且不产生中断事件 0001：当 ETPS.INTCNT=1（每次）时产生中断脉冲 0010：当 ETPS.INTCNT=2（每 2 次）时产生中断脉冲 …… 1111：当 ETPS.INTCNT=15（每 15 次）时产生中断脉冲

表 10-128：事件触发标志寄存器（ETFLG）位段定义

ETFLG (Event-Trigger Flag Register) Offset: 0xF0 Default: 0x00000000 Access: PWM -> ETFLG.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				SOCC	SOCB	SOCA	INT

表 10-129：事件触发标志寄存器（ETFLG）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	SOCC	RO	0x0	PWM 送至 ADC 的采样开始脉冲 SOCC 发生的标志 0：SOCC 脉冲未产生 1：SOCC 脉冲产生 本位段置 1 时不会阻止后续 SOCC 脉冲产生。
2	SOCB	RO	0x0	PWM 送至 ADB 的采样开始脉冲 SOCB 发生的标志 0：SOCB 脉冲未产生 1：SOCB 脉冲产生 本位段置 1 时不会阻止后续 SOCB 脉冲产生。
1	SOCA	RO	0x0	PWM 送至 ADC 的采样开始脉冲 SOCA 发生的标志 0：SOCA 脉冲未产生 1：SOCA 脉冲产生 本位段置 1 时不会阻止后续 SOCA 脉冲产生。
0	INT	RO	0x0	PWM 送至 CPU 的中断发生的标志 0：中断脉冲未产生 1：中断脉冲产生 本位段置 1 时将阻止后续中断脉冲发生。

表 10-130：事件触发标志清除寄存器（ETCLR）位段定义

ETCLR (Event-Trigger Clear Register) Offset: 0xF4 Default: 0x00000000 Access: PWM -> ETCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				SOCC	SOCB	SOCA	INT

表 10-131：事件触发标志清除寄存器（ETCLR）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	SOCC	W1C	0x0	SOCC 标志清除 0：写 0 无效，总是读回 0 1：写清除 ETFLG.SOCC 标志，本位段自动清零
2	SOCB	W1C	0x0	SOCB 标志清除 0：写 0 无效，总是读回 0 1：写清除 ETFLG.SOCB 标志，本位段自动清零
1	SOCA	W1C	0x0	SOCA 标志清除 0：写 0 无效，总是读回 0 1：写清除 ETFLG.SOCA 标志，本位段自动清零
0	INT	W1C	0x0	中断标志清除 0：写 0 无效，总是读回 0 1：写清除 ETFLG.INT 标志，本位段自动清零

表 10-132：事件触发软件强制寄存器（ETFRC）位段定义

ETFRC (Event-Trigger Force Register) Offset: 0xF8 Default: 0x00000000 Access: PWM -> ETFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				SOCC	SOCB	SOCA	INT

表 10-133：事件触发软件强制寄存器（ETFRC）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	SOCC	W1S	0x0	软件强制 SOCC 事件 ETCTL.SOCCEN=0 或 ETPS.SOCCPRD=0 时本位段无效。 0：写 0 无效，总是读回 0 1：写 1 强制产生一次 SOCC 事件，本位段自动清零
2	SOCB	W1S	0x0	软件强制 SOCB 事件 ETCTL.SOCBEN=0 或 ETPS.SOCBPRD=0 时本位段无效。 0：写 0 无效，总是读回 0 1：写 1 强制产生一次 SOCB 事件，本位段自动清零
1	SOCA	W1S	0x0	软件强制 SOCA 事件 ETCTL.SOCAEN=0 或 ETPS.SOCAPRD=0 时本位段无效。 0：写 0 无效，总是读回 0 1：写 1 强制产生一次 SOCA 事件，本位段自动清零
0	INT	W1S	0x0	软件强制中断事件 ETCTL.INTEN=0 或 ETPS.INTPRD=0 时本位段无效。 0：写 0 无效，总是读回 0 1：写 1 强制产生一次中断事件，本位段自动清零

表 10-134：PWM 模块写使能寄存器（PWMREGKEY）位段定义

PWMREGKEY (PWM Register Write-Allow Key Register) Offset: 0xFC Default: 0x1ACCE551 Access: PWM -> PWMREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 10-135：PWM 模块写使能寄存器（PWMREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 以解锁受保护的 PWM 寄存器

10.11.3 PWMCFG 寄存器列表

表 10-136：PWMCFG 模块基地址

外设模块	基地址
PWMCFG	0x4000 9F00

表 10-137：PWMCFG 寄存器列表

寄存器	偏移地址	描述	复位值
TZ0SRCCTL*	0x00	TZ0 信号源控制寄存器	0x0000007F
TZ1SRCCTL*	0x04	TZ1 信号源控制寄存器	0x0000007F
TZ2SRCCTL*	0x08	TZ2 信号源控制寄存器	0x0000007F
TZ3SRCCTL*	0x0C	TZ3 信号源控制寄存器	0x0000007F
TZ4SRCCTL*	0x10	TZ4 信号源控制寄存器	0x0000007F
FRCSYNC	0x14	全局 PWM 软件强制同步寄存器	0x00000000
GPIOSYNCCTL*	0x18	GPIO 触发 PWMSYNCI 信号源控制寄存器	0x0000007F
GPIOSYNCEN*	0x1C	GPIO 触发 PWMSYNCI 使能寄存器	0x00000009
TMR0SYNCEN*	0x20	Timer 0 触发 PWMSYNCI 使能寄存器	0x00000000
TMR1SYNCEN*	0x24	Timer 1 触发 PWMSYNCI 使能寄存器	0x00000000
TMR2SYNCEN*	0x28	Timer 2 触发 PWMSYNCI 使能寄存器	0x00000000
SYNCOCTL*	0x2C	PWMSYNCO 输出控制寄存器	0x00000007
SOCAOCTL*	0x30	PWMSOCA 输出控制寄存器	0x00000007
SOCBOCTL*	0x34	PWMSOCB 输出控制寄存器	0x00000007
SOCCOCTL*	0x38	PWMSOCC 输出控制寄存器	0x00000007
PWMCFGKEY	0x3C	PWMCFG 模块写使能寄存器	0x1ACCE551

注意： 由*标记的寄存器仅当 PWMCFGKEY=0x1ACCE551 时才可以改写。

10.11.4 PWMCFG 寄存器

表 10-138 到表 10-169 列举了 PWMCFG 模块各寄存器的定义。

表 10-138：TZ0 信号源控制寄存器（TZ0SRCCTL）位段定义

TZ0SRCCTL (TZ0 Source Control Register) Offset: 0x0 Default: 0x0000007F Access: PWMCFG -> TZ0SRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-139：TZ0 信号源控制寄存器（TZ0SRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	TZ0 极性 0：GPIO 输入低时触发 TZ0 1：GPIO 输入高时触发 TZ0
5:0	IOSEL	RW	0x3F	触发 TZ0 的 GPIO 编号

表 10-140：TZ1 信号源控制寄存器（TZ1SRCCTL）位段定义

TZ1SRCCTL (TZ1 Source Control Register) Offset: 0x4 Default: 0x0000007F Access: PWMCFG -> TZ1SRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-141：TZ1 信号源控制寄存器（TZ1SRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	TZ1 极性 0：GPIO 输入低时触发 TZ1 1：GPIO 输入高时触发 TZ1
5:0	IOSEL	RW	0x3F	触发 TZ1 的 GPIO 编号

表 10-142：TZ2 信号源控制寄存器（TZ2SRCCTL）位段定义

TZ2SRCCTL (TZ2 Source Control Register) Offset: 0x8 Default: 0x0000007F Access: PWMCFG -> TZ2SRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-143：TZ2 信号源控制寄存器（TZ2SRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	TZ2 极性 0：GPIO 输入低时触发 TZ2 1：GPIO 输入高时触发 TZ2
5:0	IOSEL	RW	0x3F	触发 TZ2 的 GPIO 编号

表 10-144：TZ3 信号源控制寄存器（TZ3SRCCTL）位段定义

TZ3SRCCTL (TZ3 Source Control Register) Offset: 0xC Default: 0x0000007F Access: PWMCFG -> TZ3SRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-145：TZ3 信号源控制寄存器（TZ3SRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	TZ3 极性 0：GPIO 输入低时触发 TZ3 1：GPIO 输入高时触发 TZ3
5:0	IOSEL	RW	0x3F	触发 TZ3 的 GPIO 编号

表 10-146：TZ4 信号源控制寄存器（TZ4SRCCTL）位段定义

TZ4SRCCTL (TZ4 Source Control Register) Offset: 0x10 Default: 0x0000007F Access: PWMCFG -> TZ4SRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-147：TZ4 信号源控制寄存器（TZ4SRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	TZ4 极性 0：GPIO 输入低时触发 TZ4 1：GPIO 输入高时触发 TZ4
5:0	IOSEL	RW	0x3F	触发 TZ4 的 GPIO 编号

表 10-148：全局 PWM 软件强制同步寄存器（FRCSYNC）位段定义

FRCSYNC (Global PWM Force Synchronization Register) Offset: 0x14 Default: 0x00000000 Access: PWMCFG -> FRCSYNC.all							
31	30	29	28	27	26	25	24
PWMCLK	RESERVED						
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
PWM7SYNC	PWM6SYNC	PWM5SYNC	PWM4SYNC	PWM3SYNC	PWM2SYNC	PWM1SYNC	PWM0SYNC

表 10-149：全局 PWM 软件强制同步寄存器（FRCSYNC）位段描述

位段	位段名	属性	复位值	描述
31	PWMCLK	W1S	0x0	软件强制同步所有 PWM 的时钟 0：写 0 无效，总是读回 0 1：写 1 产生脉冲同步所有 PWM 的时钟 本位段自动清零
30:8	RESERVED_30_8	RO	0x0	保留
7	PWM7SYNC	W1S	0x0	软件强制 PWM7 同步 0：写 0 无效，总是读回 0

位段	位段名	属性	复位值	描述
				1：写 1 产生一个同步脉冲给 PWM7 本位段自动清零
6	PWM6SYNC	W1S	0x0	软件强制 PWM6 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM6 本位段自动清零
5	PWM5SYNC	W1S	0x0	软件强制 PWM5 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM5 本位段自动清零
4	PWM4SYNC	W1S	0x0	软件强制 PWM4 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM4 本位段自动清零
3	PWM3SYNC	W1S	0x0	软件强制 PWM3 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM3 本位段自动清零
2	PWM2SYNC	W1S	0x0	软件强制 PWM2 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM2 本位段自动清零
1	PWM1SYNC	W1S	0x0	软件强制 PWM1 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM1 本位段自动清零
0	PWM0SYNC	W1S	0x0	软件强制 PWM0 同步 0：写 0 无效，总是读回 0 1：写 1 产生一个同步脉冲给 PWM0 本位段自动清零

表 10-150：GPIO 触发 PWMSYNCI 信号源控制寄存器（GPIOSYNCCTL）位段定义

GPIOSYNCCTL (GPIO PWMSYNCI Control Register) Offset: 0x18 Default: 0x0000007F Access: PWMCFG -> GPIOSYNCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 10-151：GPIO 触发 PWMSYNCI 信号源控制寄存器（GPIOSYNCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	PWMSYNCI 极性 0：GPIO 输入低时触发 PWMSYNCI 1：GPIO 输入高时触发 PWMSYNCI
5:0	IOSEL	RW	0x3F	触发 PWMSYNCI 的 GPIO 编号

表 10-152：GPIO 触发 PWMSYNCI 使能寄存器（GPIOSYNCEN）位段定义

GPIOSYNCEN (GPIO Force PWMSYNCI Enable Register) Offset: 0x1C Default: 0x00000009 Access: PWMCFG -> GPIOSYNCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
PWM7SYNC	PWM6SYNC	PWM5SYNC	PWM4SYNC	PWM3SYNC	PWM2SYNC	PWM1SYNC	PWM0SYNC

表 10-153：GPIO 触发 PWMSYNCI 使能寄存器（GPIOSYNCEN）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	PWM7SYNC	RW	0x0	使能 GPIO 触发 PWM7 同步 0：关闭 1：使能
6	PWM6SYNC	RW	0x0	使能 GPIO 触发 PWM6 同步 0：关闭 1：使能

位段	位段名	属性	复位值	描述
5	PWM5SYNC	RW	0x0	使能 GPIO 触发 PWM5 同步 0：关闭 1：使能
4	PWM4SYNC	RW	0x0	使能 GPIO 触发 PWM4 同步 0：关闭 1：使能
3	PWM3SYNC	RW	0x1	使能 GPIO 触发 PWM3 同步 0：关闭 1：使能
2	PWM2SYNC	RW	0x0	使能 GPIO 触发 PWM2 同步 0：关闭 1：使能
1	PWM1SYNC	RW	0x0	使能 GPIO 触发 PWM1 同步 0：关闭 1：使能
0	PWM0SYNC	RW	0x1	使能 GPIO 触发 PWM0 同步 0：关闭 1：使能

表 10-154：Timer 0 触发 PWMSYNCI 使能寄存器（TMR0SYNCEN）位段定义

TMR0SYNCEN (Timer 0 Force PWMSYNCI Enable Register) Offset: 0x20 Default: 0x00000000 Access: PWMCFG -> TMR0SYNCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 PWM7SYNC	6 PWM6SYNC	5 PWM5SYNC	4 PWM4SYNC	3 PWM3SYNC	2 PWM2SYNC	1 PWM1SYNC	0 PWM0SYNC

表 10-155：Timer 0 触发 PWMSYNCI 使能寄存器（TMR0SYNCEN）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	PWM7SYNC	RW	0x0	使能 Timer 0 触发 PWM7 同步 0：关闭 1：使能
6	PWM6SYNC	RW	0x0	使能 Timer 0 触发 PWM6 同步 0：关闭 1：使能
5	PWM5SYNC	RW	0x0	使能 Timer 0 触发 PWM5 同步 0：关闭 1：使能
4	PWM4SYNC	RW	0x0	使能 Timer 0 触发 PWM4 同步 0：关闭 1：使能
3	PWM3SYNC	RW	0x0	使能 Timer 0 触发 PWM3 同步 0：关闭 1：使能
2	PWM2SYNC	RW	0x0	使能 Timer 0 触发 PWM2 同步 0：关闭 1：使能
1	PWM1SYNC	RW	0x0	使能 Timer 0 触发 PWM1 同步 0：关闭 1：使能
0	PWM0SYNC	RW	0x0	使能 Timer 0 触发 PWM0 同步 0：关闭 1：使能

表 10-156：Timer 1 触发 PWMSYNCI 使能寄存器（TMR1SYNCEN）位段定义

TMR1SYNCEN (Timer 1 Force PWMSYNCI Enable Register) Offset: 0x24 Default: 0x00000000 Access: PWMCFG -> TMR1SYNCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
PWM7SYNC	PWM6SYNC	PWM5SYNC	PWM4SYNC	PWM3SYNC	PWM2SYNC	PWM1SYNC	PWM0SYNC

表 10-157：Timer 1 触发 PWMSYNCI 使能寄存器（TMR1SYNCEN）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	PWM7SYNC	RW	0x0	使能 Timer 1 触发 PWM7 同步 0：关闭 1：使能
6	PWM6SYNC	RW	0x0	使能 Timer 1 触发 PWM6 同步 0：关闭 1：使能
5	PWM5SYNC	RW	0x0	使能 Timer 1 触发 PWM5 同步 0：关闭 1：使能
4	PWM4SYNC	RW	0x0	使能 Timer 1 触发 PWM4 同步 0：关闭 1：使能
3	PWM3SYNC	RW	0x0	使能 Timer 1 触发 PWM3 同步 0：关闭 1：使能
2	PWM2SYNC	RW	0x0	使能 Timer 1 触发 PWM2 同步 0：关闭 1：使能
1	PWM1SYNC	RW	0x0	使能 Timer 1 触发 PWM1 同步 0：关闭 1：使能
0	PWM0SYNC	RW	0x0	使能 Timer 1 触发 PWM0 同步 0：关闭 1：使能

表 10-158：Timer 2 触发 PWMSYNCI 使能寄存器（TMR2SYNCEN）位段定义

TMR2SYNCEN (Timer 2 Force PWMSYNCI Enable Register) Offset: 0x28 Default: 0x00000000 Access: PWMCFG -> TMR2SYNCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 PWM7SYNC	6 PWM6SYNC	5 PWM5SYNC	4 PWM4SYNC	3 PWM3SYNC	2 PWM2SYNC	1 PWM1SYNC	0 PWM0SYNC

表 10-159：Timer 2 触发 PWMSYNCI 使能寄存器（TMR2SYNCEN）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	PWM7SYNC	RW	0x0	使能 Timer 2 触发 PWM7 同步 0：关闭 1：使能
6	PWM6SYNC	RW	0x0	使能 Timer 2 触发 PWM6 同步 0：关闭 1：使能
5	PWM5SYNC	RW	0x0	使能 Timer 2 触发 PWM5 同步 0：关闭 1：使能
4	PWM4SYNC	RW	0x0	使能 Timer 2 触发 PWM4 同步 0：关闭 1：使能
3	PWM3SYNC	RW	0x0	使能 Timer 2 触发 PWM3 同步 0：关闭 1：使能
2	PWM2SYNC	RW	0x0	使能 Timer 2 触发 PWM2 同步 0：关闭 1：使能
1	PWM1SYNC	RW	0x0	使能 Timer 2 触发 PWM1 同步 0：关闭 1：使能
0	PWM0SYNC	RW	0x0	使能 Timer 2 触发 PWM0 同步 0：关闭 1：使能

表 10-160：PWMSYNCO 输出控制寄存器（SYNCOCTL）位段定义

SYNCOCTL (PWMSYNCO Control Register) Offset: 0x2C Default: 0x00000007 Access: PWMCFG -> SYNCOCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		SRCSEL			POL	DURATION	

表 10-161：PWMSYNCO 输出控制寄存器（SYNCOCTL）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5:3	SRCSEL	RW	0x0	PWMSYNCO 信号源选择 000：选择 PWM0 的 PWMSYNCO 输出 001：选择 PWM1 的 PWMSYNCO 输出 010：选择 PWM2 的 PWMSYNCO 输出 011：选择 PWM3 的 PWMSYNCO 输出 100：选择 PWM4 的 PWMSYNCO 输出 101：选择 PWM5 的 PWMSYNCO 输出 110：选择 PWM6 的 PWMSYNCO 输出 111：选择 PWM7 的 PWMSYNCO 输出 其他：没有选择 PWMSYNCO 输出
2	POL	RW	0x1	PWMSYNCO 输出到 IO 管脚的极性 0：PWMSYNCO 有效时将管脚置低 1：WPMSYNCO 有效时将管脚置高
1:0	DURATION	RW	0x3	PWMSYNCO 输出到 IO 管脚的脉冲宽度 00：脉冲宽度为 4 个 PWM 时钟周期 01：脉冲宽度为 8 个 PWM 时钟周期 10：脉冲宽度为 16 个 PWM 时钟周期 11：脉冲宽度为 32 个 PWM 时钟周期

表 10-162：PWMSOCA 输出控制寄存器（SOCAOCTL）位段定义

SOCAOCTL (PWMSOCA Output Control Register) Offset: 0x30 Default: 0x00000007 Access: PWMCFG -> SOCAOCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SOCA7EN	SOCA6EN	SOCA5EN
7	6	5	4	3	2	1	0
SOCA4EN	SOCA3EN	SOCA2EN	SOCA1EN	SOCA0EN	POL	DURATION	

表 10-163：PWMSOCA 输出控制寄存器（SOCAOCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	SOCA7EN	RW	0x0	使能（逻辑或）PWM7SOCA 输出到 IO 管脚 0：关闭 1：使能
9	SOCA6EN	RW	0x0	使能（逻辑或）PWM6SOCA 输出到 IO 管脚 0：关闭 1：使能
8	SOCA5EN	RW	0x0	使能（逻辑或）PWM5SOCA 输出到 IO 管脚 0：关闭 1：使能
7	SOCA4EN	RW	0x0	使能（逻辑或）PWM4SOCA 输出到 IO 管脚 0：关闭 1：使能
6	SOCA3EN	RW	0x0	使能（逻辑或）PWM3SOCA 输出到 IO 管脚 0：关闭 1：使能
5	SOCA2EN	RW	0x0	使能（逻辑或）PWM2SOCA 输出到 IO 管脚 0：关闭 1：使能
4	SOCA1EN	RW	0x0	使能（逻辑或）PWM1SOCA 输出到 IO 管脚 0：关闭 1：使能
3	SOCA0EN	RW	0x0	使能（逻辑或）PWM0SOCA 输出到 IO 管脚 0：关闭 1：使能
2	POL	RW	0x1	PWMSOCA 输出到 IO 管脚的极性 0：PWMSOCA 有效时 IO 管脚置低 1：PWMSOCA 有效是 IO 管脚置低

位段	位段名	属性	复位值	描述
1:0	DURATION	RW	0x3	PWMSOCA 输出到 IO 管脚的脉冲宽度 00：脉冲宽度为 4 个 PWM 时钟周期 01：脉冲宽度为 8 个 PWM 时钟周期 10：脉冲宽度为 16 个 PWM 时钟周期 11：脉冲宽度为 32 个 PWM 时钟周期

表 10-164：PWMSOCB 输出控制寄存器（SOCBOCTL）位段定义

SOCBOCTL (PWMSOCB Output Control Register) Offset: 0x34 Default: 0x00000007
Access: PWMCFG -> SOCBOCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SOCB7EN	SOCB6EN	SOCB5EN
7	6	5	4	3	2	1	0
SOCB4EN	SOCB3EN	SOCB2EN	SOCB1EN	SOCB0EN	POL	DURATION	

表 10-165：PWMSOCB 输出控制寄存器（SOCBOCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	SOCB7EN	RW	0x0	使能（逻辑或）PWM7SOCB 输出到 IO 管脚 0：关闭 1：使能
9	SOCB6EN	RW	0x0	使能（逻辑或）PWM6SOCB 输出到 IO 管脚 0：关闭 1：使能
8	SOCB5EN	RW	0x0	使能（逻辑或）PWM5SOCB 输出到 IO 管脚 0：关闭 1：使能
7	SOCB4EN	RW	0x0	使能（逻辑或）PWM4SOCB 输出到 IO 管脚 0：关闭 1：使能
6	SOCB3EN	RW	0x0	使能（逻辑或）PWM3SOCB 输出到 IO 管脚 0：关闭 1：使能
5	SOCB2EN	RW	0x0	使能（逻辑或）PWM2SOCB 输出到 IO 管脚 0：关闭 1：使能

位段	位段名	属性	复位值	描述
4	SOCB1EN	RW	0x0	使能（逻辑或）PWM1SOCB 输出到 IO 管脚 0：关闭 1：使能
3	SOCB0EN	RW	0x0	使能（逻辑或）PWM0SOCB 输出到 IO 管脚 0：关闭 1：使能
2	POL	RW	0x1	PWMSOCB 输出到 IO 管脚的极性 0：PWMSOCB 有效时 IO 管脚置低 1：PWMSOCB 有效是 IO 管脚置低
1:0	DURATION	RW	0x3	PWMSOCB 输出到 IO 管脚的脉冲宽度 00：脉冲宽度为 4 个 PWM 时钟周期 01：脉冲宽度为 8 个 PWM 时钟周期 10：脉冲宽度为 16 个 PWM 时钟周期 11：脉冲宽度为 32 个 PWM 时钟周期

表 10-166：PWMSOCC 输出控制寄存器（SOCCOCTL）位段定义

SOCCOCTL (PWMSOCC Output Control Register)　Offset: 0x38　Default: 0x00000007
Access: PWMCFG -> SOCCOCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					SOCC7EN	SOCC6EN	SOCC5EN
7	6	5	4	3	2	1	0
SOCC4EN	SOCC3EN	SOCC2EN	SOCC1EN	SOCC0EN	POL	DURATION	

表 10-167：PWMSOCC 输出控制寄存器（SOCCOCTL）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	SOCC7EN	RW	0x0	使能（逻辑或）PWM7SOCC 输出到 IO 管脚 0：关闭 1：使能
9	SOCC6EN	RW	0x0	使能（逻辑或）PWM6SOCC 输出到 IO 管脚 0：关闭 1：使能
8	SOCC5EN	RW	0x0	使能（逻辑或）PWM5SOCC 输出到 IO 管脚 0：关闭 1：使能
7	SOCC4EN	RW	0x0	使能（逻辑或）PWM4SOCC 输出到 IO 管脚

位段	位段名	属性	复位值	描述
				0：关闭 1：使能
6	SOCC3EN	RW	0x0	使能（逻辑或）PWM3SOCC 输出到 IO 管脚 0：关闭 1：使能
5	SOCC2EN	RW	0x0	使能（逻辑或）PWM2SOCC 输出到 IO 管脚 0：关闭 1：使能
4	SOCC1EN	RW	0x0	使能（逻辑或）PWM1SOCC 输出到 IO 管脚 0：关闭 1：使能
3	SOCC0EN	RW	0x0	使能（逻辑或）PWM0SOCC 输出到 IO 管脚 0：关闭 1：使能
2	POL	RW	0x1	PWMSOCC 输出到 IO 管脚的极性 0：PWMSOCC 有效时 IO 管脚置低 1：PWMSOCC 有效是 IO 管脚置低
1:0	DURATION	RW	0x3	PWMSOCC 输出到 IO 管脚的脉冲宽度 00：脉冲宽度为 4 个 PWM 时钟周期 01：脉冲宽度为 8 个 PWM 时钟周期 10：脉冲宽度为 16 个 PWM 时钟周期 11：脉冲宽度为 32 个 PWM 时钟周期

表 10-168：PWMCFG 模块写使能寄存器（PWMCFG）位段定义

PWMCFGKEY (PWMCFG Register Write-Allow Key Register) Offset: 0x3C Default: 0x1ACCE551 Access: PWMCFG -> PWMCFGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 10-169：PWMCFG 模块写使能寄存器（PWMCFG）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 PWMCFG 寄存器

11 增强的捕获模块（ECAP）

11.1 概述

增强的捕获模块（ECAP）用于对外部输入事件的时序关系进行计时，应用场景包括：

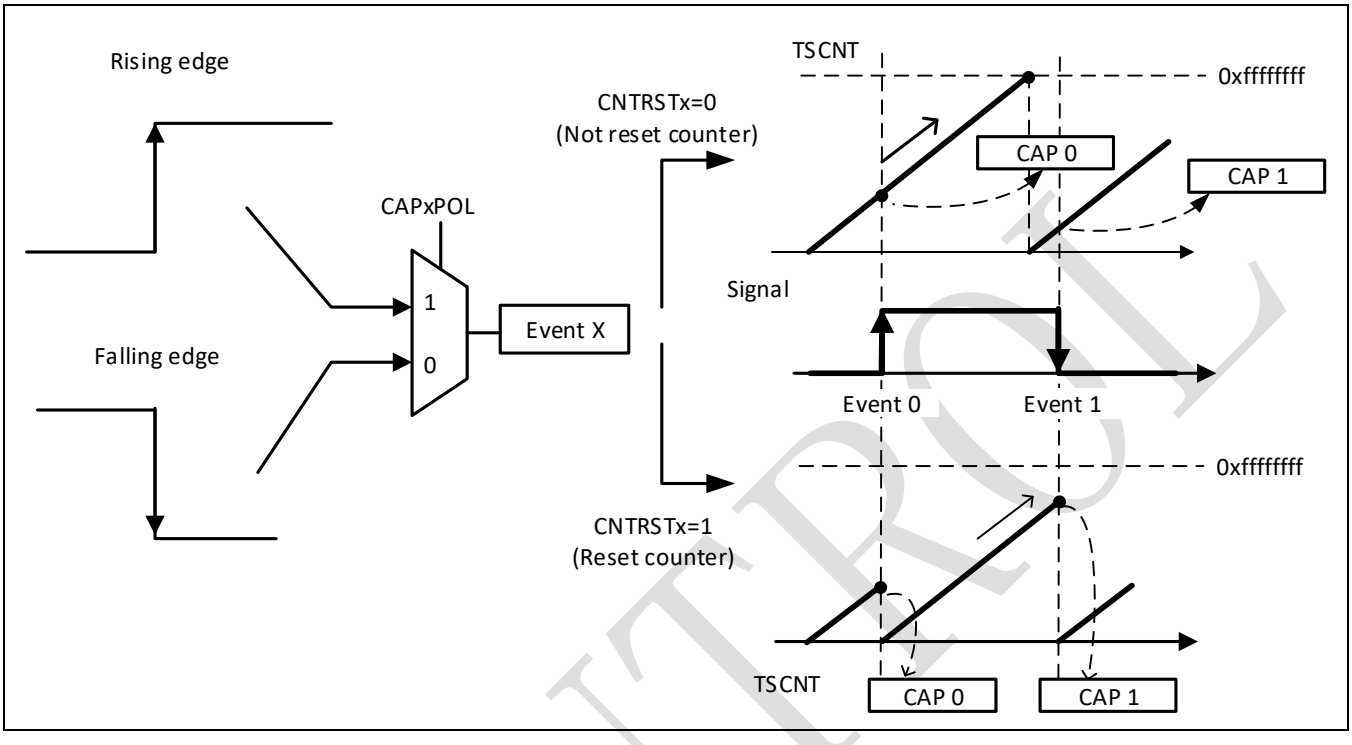
- 机电设备的速度测量（例如使用 Hall 传感器的电动车）
- 测量脉冲信号的周期和占空
- 对于占空比编码系统的解码

SPC2168 中的 ECAP 具备如下特性：

- 32 比特时基计数，最小 5ns 分辨率（最高 200MHz 时钟）
- 4 个事件触发的 32 比特时基计数捕获寄存器
- 最多 4 个时序事件的边沿极性选择
- 任何一个捕获事件都可以触发 CPU 中断
- 一次性捕获模式，在逐步完成最多 4 个捕获后停止
- 持续捕获模式，在最多 4 个捕获事件间持续轮流执行
- 绝对时间戳捕获
- 相对时间戳捕获
- 针对用户指定的输入端口完成上述功能
- 若不需要捕获功能，可以将 ECAP 配置为简单的 PWM 工作模式

11.2 功能描述

图 11-1：ECAP 基本工作方式



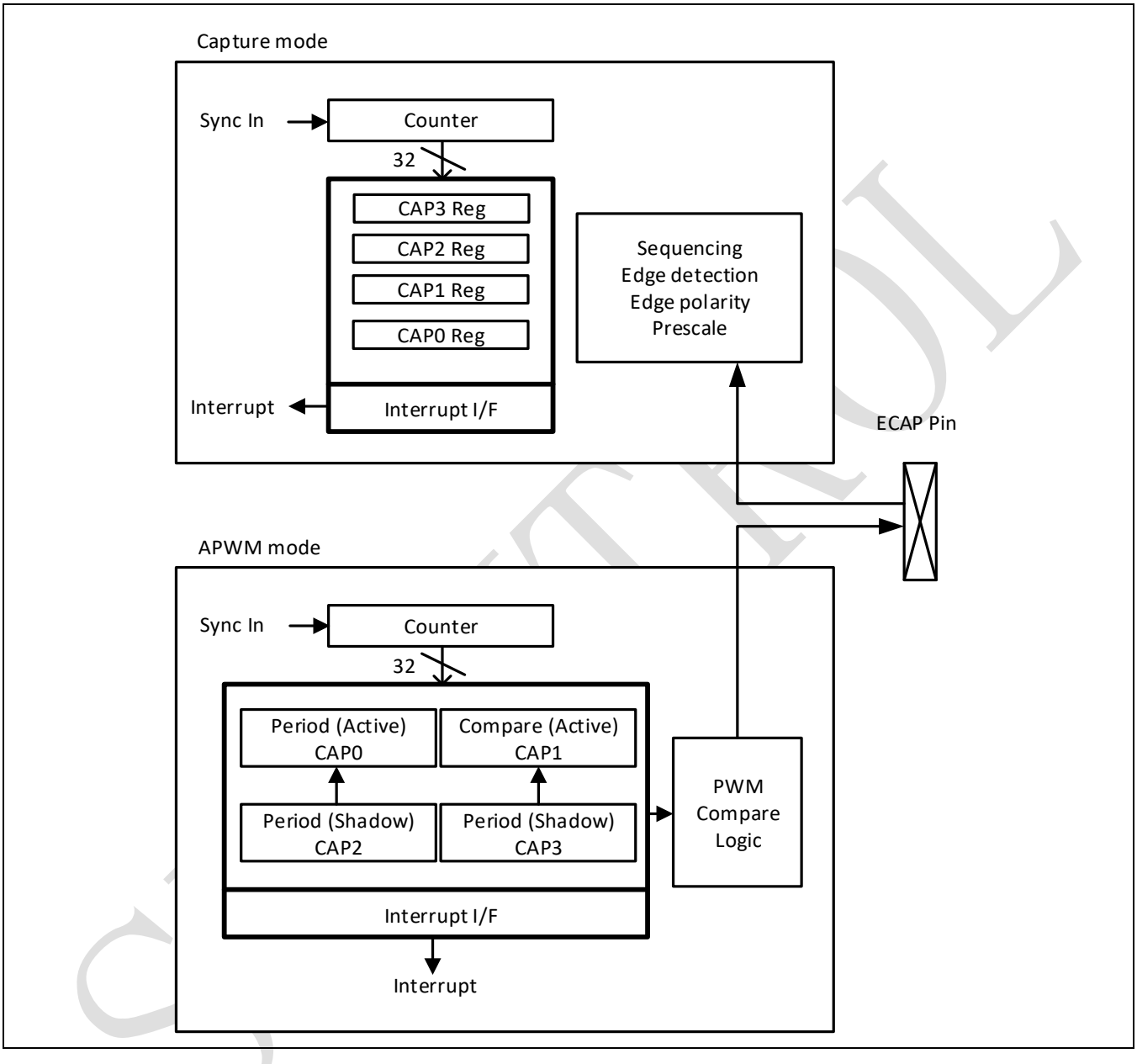
ECAP 模块根据用户对 CAPCTL.CAPxPOL 位段的配置，将输入信号的上升沿或者下降沿视作一个“事件”，并逐一编号成最多 4 个具有时序先后的捕获事件。

同时，用户通过 CAPCTL.CNTRSTx 位段设置每个捕获事件发生时，时间戳计数器是否复位。

通过两者的配置，用户可以实现诸如信号周期和占空计算等复杂的捕获和测量功能，图 11-1 示例了具体的工作模式。

11.3 捕获模式和辅助 APWM 模式

图 11-2：捕获模式和 APWM 模式



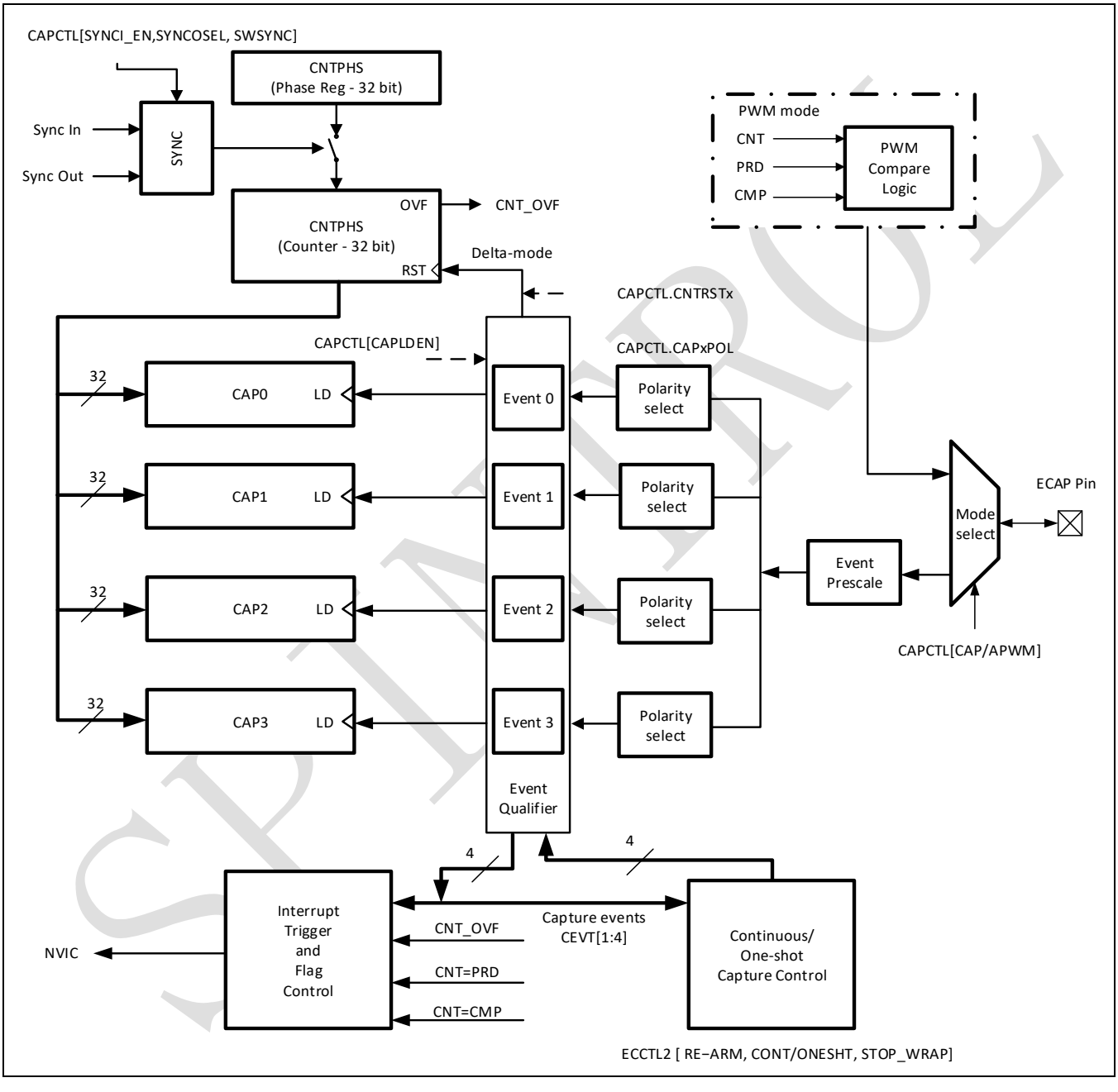
当 ECAP 模块不用作输入捕获时，用户可以将其用作有 32 位计数能力的单通道 PWM 模块。此时计数器处于向上计数模式，并提供非对称 PWM 波形。

此时，CAP0 和 CAP1 分别用作周期 PRD 和计数比较值 CMP 的有效寄存器，CAP2 和 CAP3 分别用作周期 PRD 和计数比较值 CMP 的影子寄存器。图 11-2 给出了捕获模式和 PWM 模式的顶层架构。

11.4 捕获模式详述

捕获模式的功能框图如图 11-3 所示。其中 ECAP 输入来自于由 CAPSRCCTL 寄存器指定的 GPIO 端口。

图 11-3：捕获模式功能框图



11.4.1 事件预分频

输入的待捕获事件（持续脉冲序列）可以进行预分频，比例为 N = CAPCTL.EVTDIV+1，这在高频输入作为输入事件时非常有用。图 11-4 和图 11-5 分别给出了预分频的控制逻辑和波形示例。

图 11-4：事件预分频控制

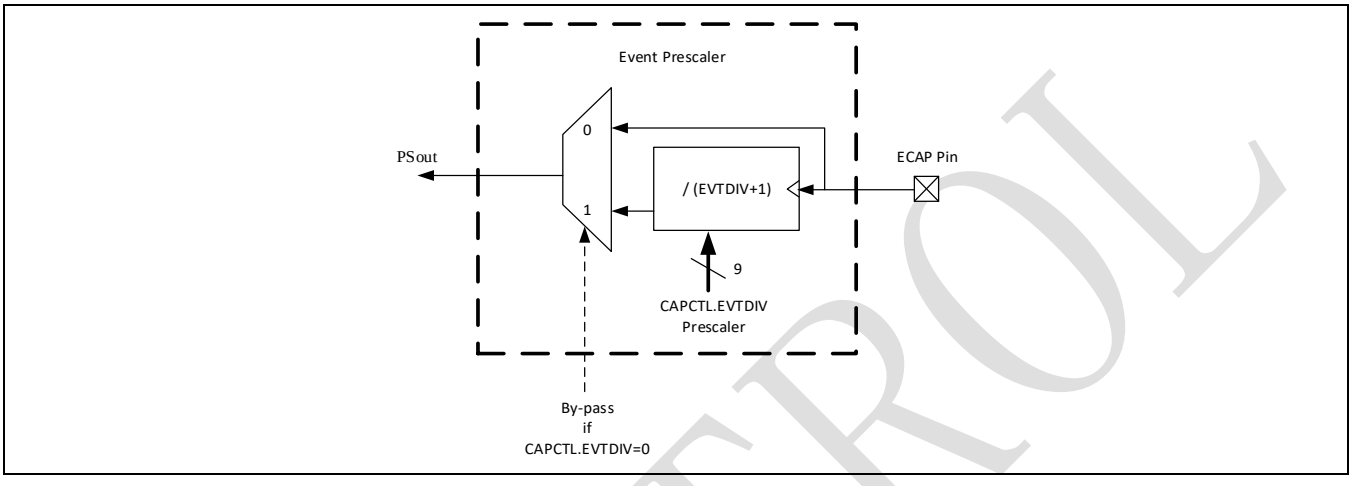
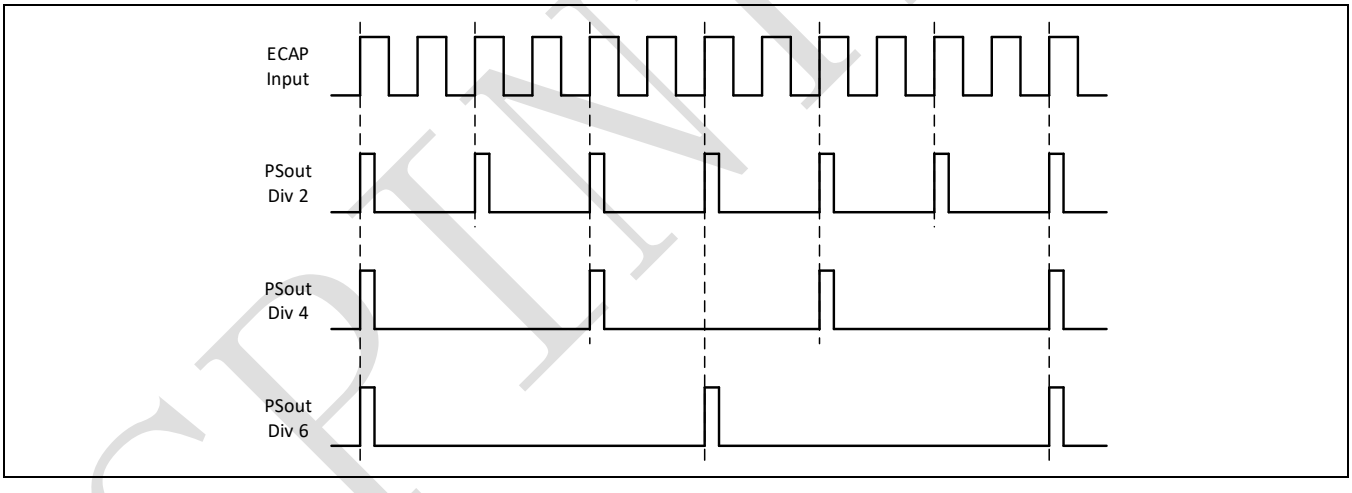


图 11-5：事件预分频波形



注意： ECAP 的事件分频在硬件上采用了类似时钟分频的电路结构，因此上电初始化需要额外的四个事件周期来初始化分频操作。在一次性捕获模式下，用户非常关心每个事件实际的导致的行为，此时初始化需求就会造成问题。因此用户可以在软件上通过主动翻转 GPIO 来生成用于初始化分频的 4 个事件来解决这个问题，详见 SDK 中的示例代码。

11.4.2 边沿选择和排序编号

每个事件（最多 4 个）对应的输入信号边沿（上升沿或下降沿）都通过独立的选择器配置：

- 每个边沿事件都通过模 4 计数器按顺序编号为 CEVT0~CEVT4

- 每个事件触发的捕获，其结果由模 4 计数器控制存入对应的 CAPx 寄存器

11.4.3 持续捕获和一次性捕获控制

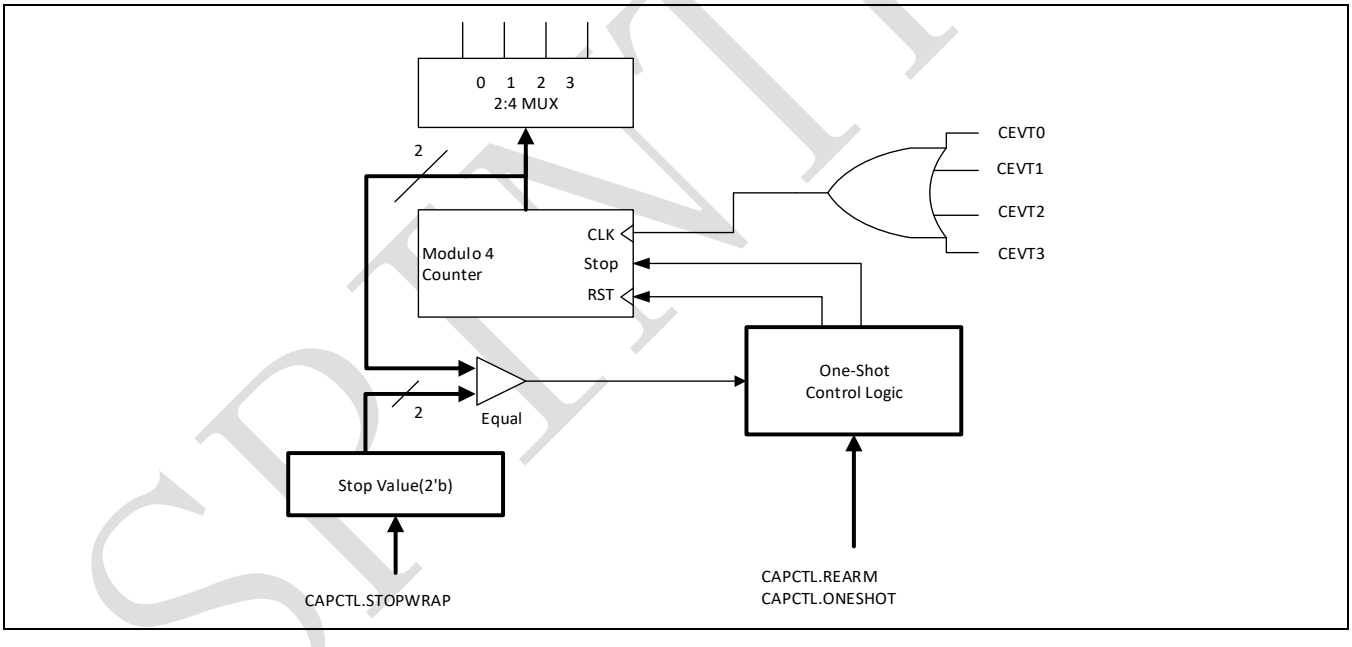
模 4（2 比特）计数器在 CEVT0-CEVT3 事件发生时逐一递增计数，它记录了当前事件的编号，并与 CAPCTL.STOPWARP 位段按照如下方式配合控制捕获的时序：

- 持续捕获模式下，模 4 事件计数器从 0 开始递增计数，一旦计数到 CAPCTL.STOPWARP 则自动归零，并继续周而复始地计数。
- 在一次性捕获模式下，模 4 事件计数器从 0 开始递增计数，一旦计数到 CAPCTL.STOPWARP 则停止计数，并阻止后续的捕获和对 CAP0~CAP3 寄存器的更新。

一次性计数模式下，可以通过软件方式将模 4 事件计数状态复位。模 4 事件计数复位后，ECAP 模块等待 1 到 4 个（由 CAPCTL.STOPWARP 指定）捕获事件，然后锁定模 4 事件计数和 CAP0~CAP3 的内容。新的模 4 事件计数软件复位可以让 ECAP 模块开始新一轮的捕获序列，并解锁 CAP0~CAP3 寄存器以存放新的捕获值。

持续捕获模式下，CAP0~CAP3 根据模 4 事件计数控制轮流不断存入最新的捕获值。

图 11-6：持续捕获和一次性捕获逻辑

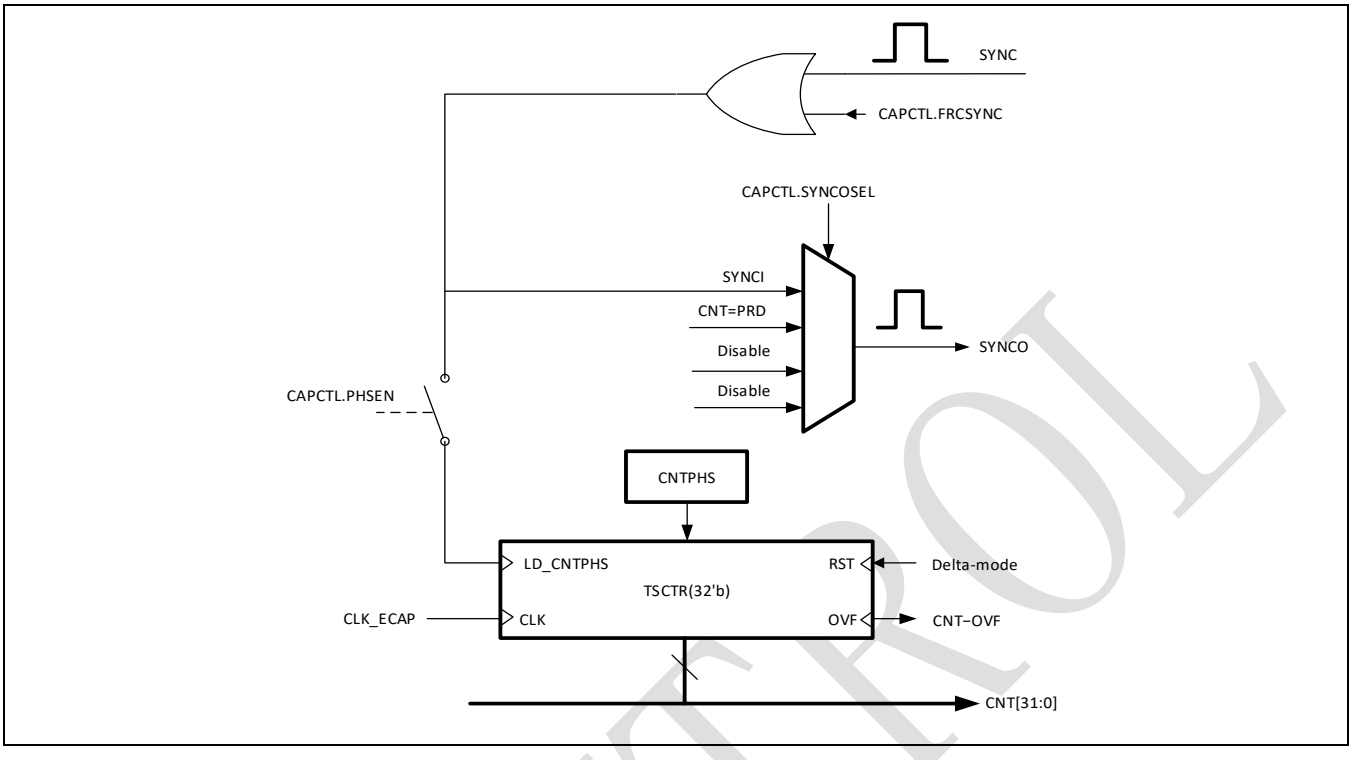


11.4.4 32 比特计数器和相位控制

32 位计数器提供了事件捕获的时间戳，并直接由 CLK_ECAP 作为计数时钟。此外，相位寄存器 CAPCNTPHS 指定了硬件或者软件同步发生时相对于其他计数器的相位偏移，这在需要指定跨模块相位偏移的 PWM 模式下非常有用。

在任意一个捕获事件发生时，均可以配置计数器复位，便于实现对时间差的捕获和计算。例如将上升沿时捕获并复位计数器，在下一个下降沿时再次捕获，则可以得到输入信号的占空。

图 11-7：时间戳计数器及其同步



11.4.5 CAP0-CAP3 寄存器

这 4 个 32 位计数器在对应的捕获事件发生时存入当时的时间戳计数器的计数值。在一次性捕获模式下，完成 CAPCTL.STOPWARP 指定次数的捕获后，CAP0~CAP3 自动停止捕获和更新。

在 APWM 模式下，CAP0 和 CAP1 分别用作周期 PRD 和计数比较值 CMP 的有效寄存器，CAP2 和 CAP3 分别用作周期 PRD 和计数比较值 CMP 的影子寄存器。

11.4.6 中断控制

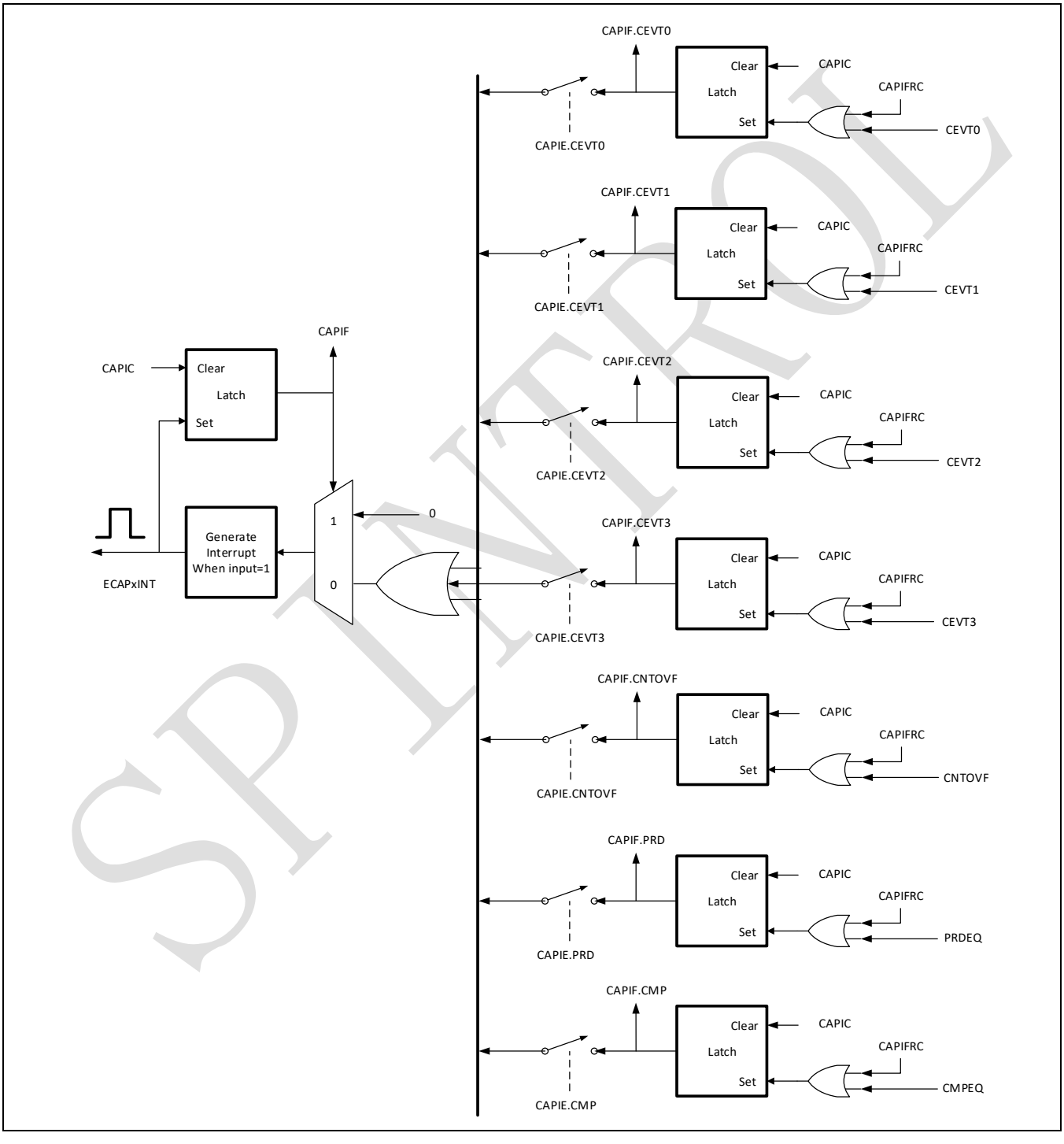
捕获事件（CEVT0-CEVT3）、APWM 事件（CNT=PRD 和 CNT=CMP）均可以触发中断。此外，计数溢出（0xFFFFFFFF->0x00000000）也可以作为独立的事件（CNTOVF）用来触发 CPU 中断。

这 7 个中断事件（CEVT0、CEVT1、CEVT2、CEVT3、CNTOVF、CNT=PRD 和 CNT=CMP）可以通过中断使能寄存器 CAPIE 独立配置是否触发 CPU 中断。

中断标志寄存器 CAPIF 记录了每个事件标志以及全局中断标志。当被使能的任何一个中断事件发生时，仅当对应的事件标志和全局中断标志都为 0 时才会产生新的中断脉冲到 NVIC。中断服务代码里必须通过中断清除寄存器 CAPIC 同时清除事件标志和全局中断标志才能保证后续事件能触发新的中断。保持事件标志为 1，将阻止后续同样的事件触发中断，但不影响其他事件触发中断；保持全局中断标志为 1，则阻止后续所有事件触发中断。用户可以通过 CAPIFRC 寄存器软件强制一个中断事件，用于代码调试。

注意： CEVT0、CEVT1、CEVT2、CEVT3 标志仅在捕获模式有效（CAPCTL.CAPAPWM = 0）；CNT=PRD、CNT=CMP 标志仅在 APWM 模式有效（CAPCTL.CAPAPWM = 1）；CNTOVF 标志在两种模式下均有效。

图 11-8：ECAP 模块的中断



11.4.7 影子寄存及其控制

在捕获模式下，不允许 CAP0 和 CAP1 从对应的影子寄存器 CAP2 和 CAP3 中读入值并更新；

在 APWM 模式下，影子模式被激活，用户可以选用如下任意一种写入方式：

- 直接写入：通过 CAP0 写入计数周期 PRD 和通过 CAP1 写入计数比较值 CMP，此时对应的 CAP2 和 CAP3 也立即做相应的更新。
- 影子写入：通过 CAP2 写入计数周期 PRD 的影子值和通过 CAP3 写入计数比较值 CMP 的影子值。此时对应的 CAP0 和 CAP1 保持不变。当计数到当前 CAP0（即当前有效 PRD）时，分别用 CAP2 和 CAP3 的值来更新 CAP0 和 CAP1。

11.5 APWM 模式详述

APWM 工作模式的要点如下：

- 时间戳计数器保持计数，并与 CMP 和 PRD 两个值进行比较
- CAP0 寄存器用作周期 PRD，CAP1 寄存器用作 CMP
- CAP2 和 CAP3 用作 CAP0 和 CAP1 的影子寄存器，并在 CNT=PRD 时更新 CAP0 和 CAP1
- 对 CAP0 和 CAP1 的写操作对应于 CMP 和 PRD 的直接写模式
- 对 CAP2 和 CAP3 的写操作对应于 CMP 和 PRD 的影子写模式
- 初始化时，必须通过写 CAP0 和 CAP1 指定 PRD 和 CMP。在之后，可以根据需求选用直接写或者影子写 PRD 和 CMP。

图 11-9：APWM 模式下 PRD 和 CMP 的有效值和影子值

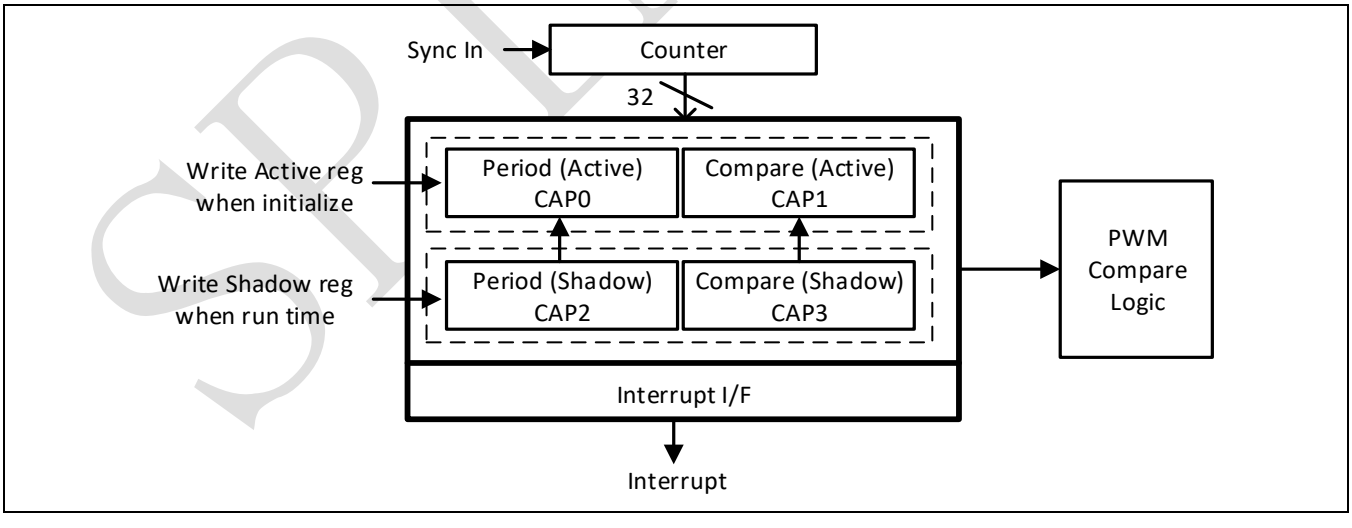
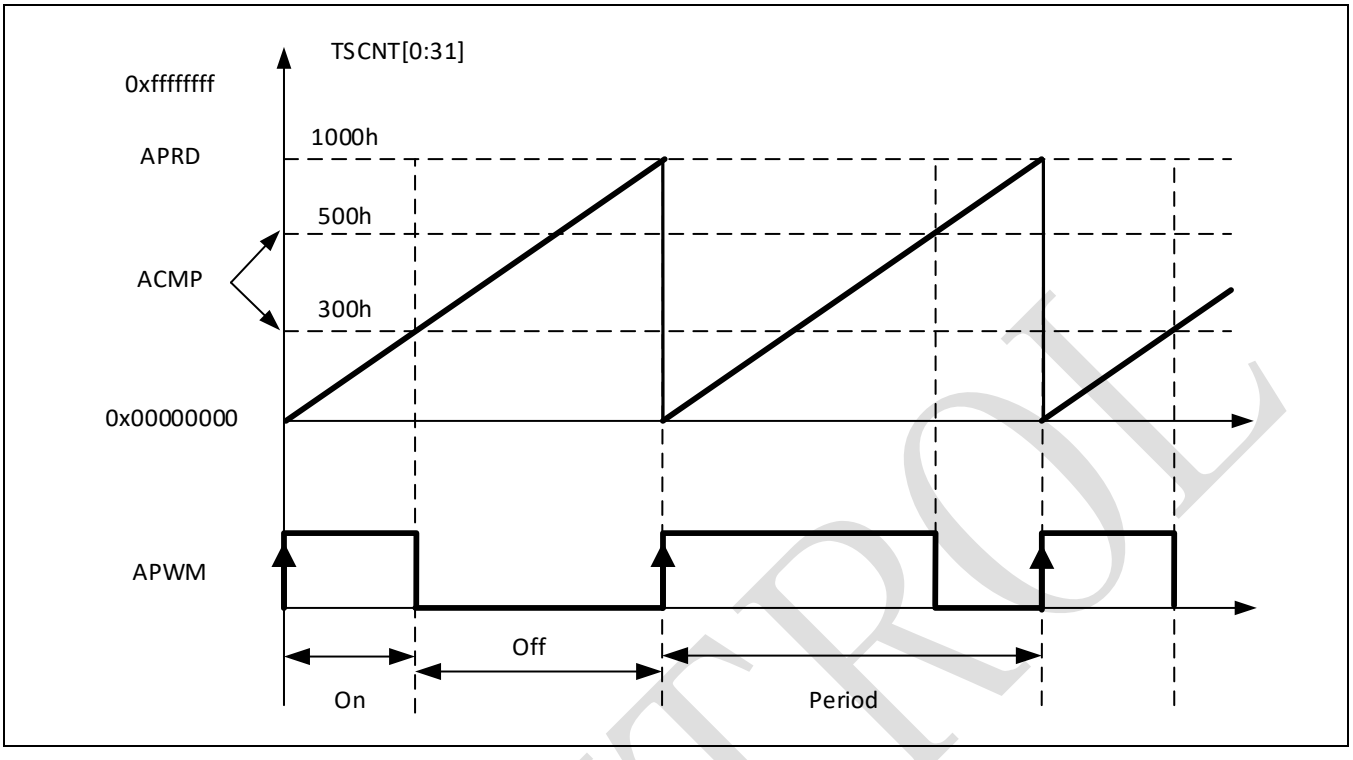


图 11-10：APWM 模式波形（极性为高有效）



11.6 ECAP 模块应用

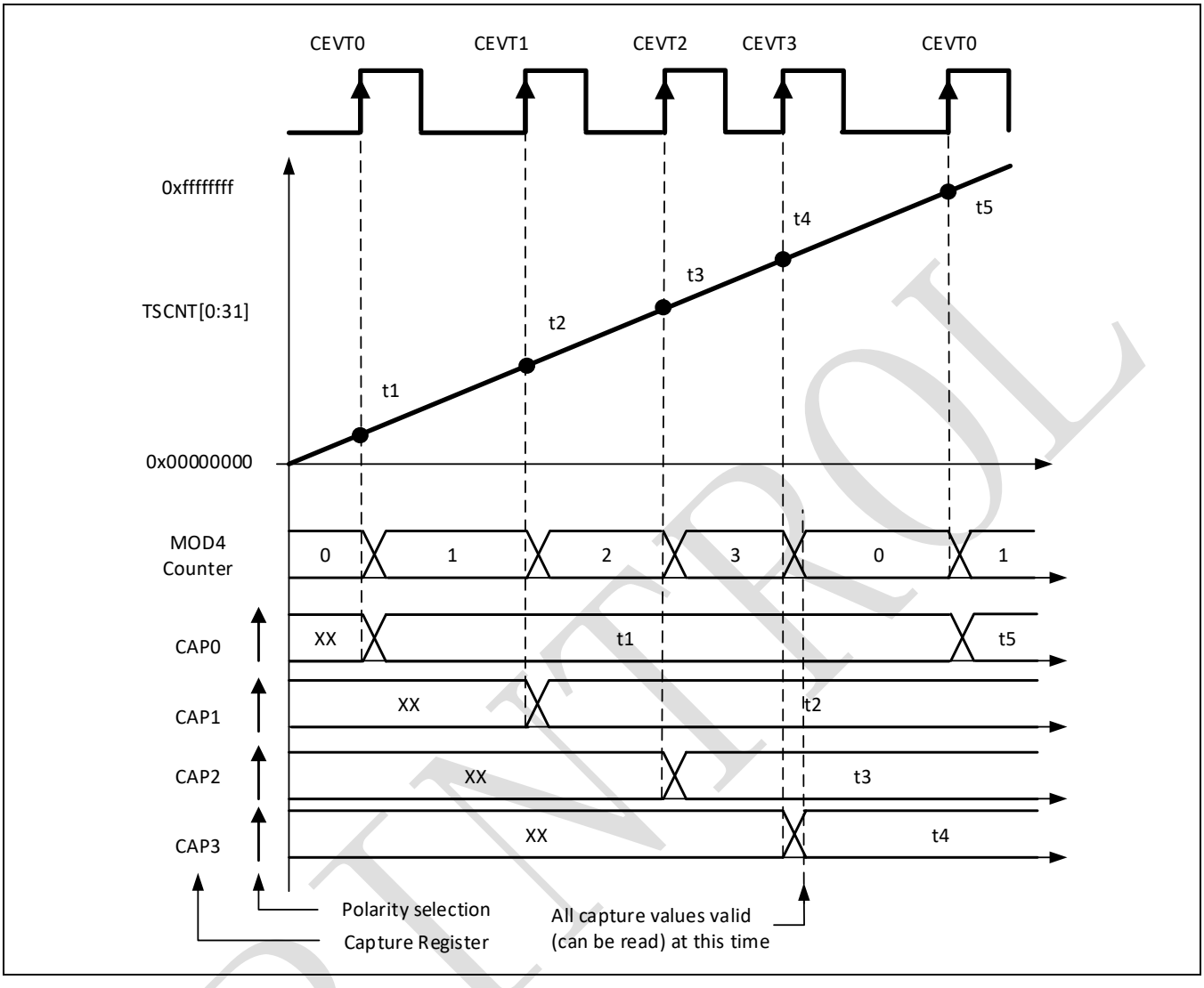
接下来的章节给出实际应用场景和具体代码的示例。为了可读性和使用方便，示例代码中使用了 SPC2168 ECAP “C” 头文件。

示例 1 –上升沿触发的绝对时间戳捕获

图 11-11 示例了持续捕获模式（模 4 事件计数自动归零持续计数）。图中，TSCNT 持续递增计而不复位，输入上升沿视作有效事件。

每个有效事件下，TSCNT 当前值被作为时间戳捕获到对应的 CAPx 寄存器，然后模 4 事件计数加 1。当 TSCNT 计数到 0xFFFFFFFF 时自动回到 0x00000000 继续计数（未在图 11-11）中示例给出）。同时，计数溢出标志 CNTOVF 被置位，如若使能，还将触发 CPU 中断。图中标记了各个有效事件的时间点，从中可以看出每到 CEVT3 时 4 个捕获已经依次完成，因此可以选择 CEVT3 作为中断触发源，便于读取所有 CAPx 寄存器的值。

图 11-11：上升沿检测和绝对时间戳捕获



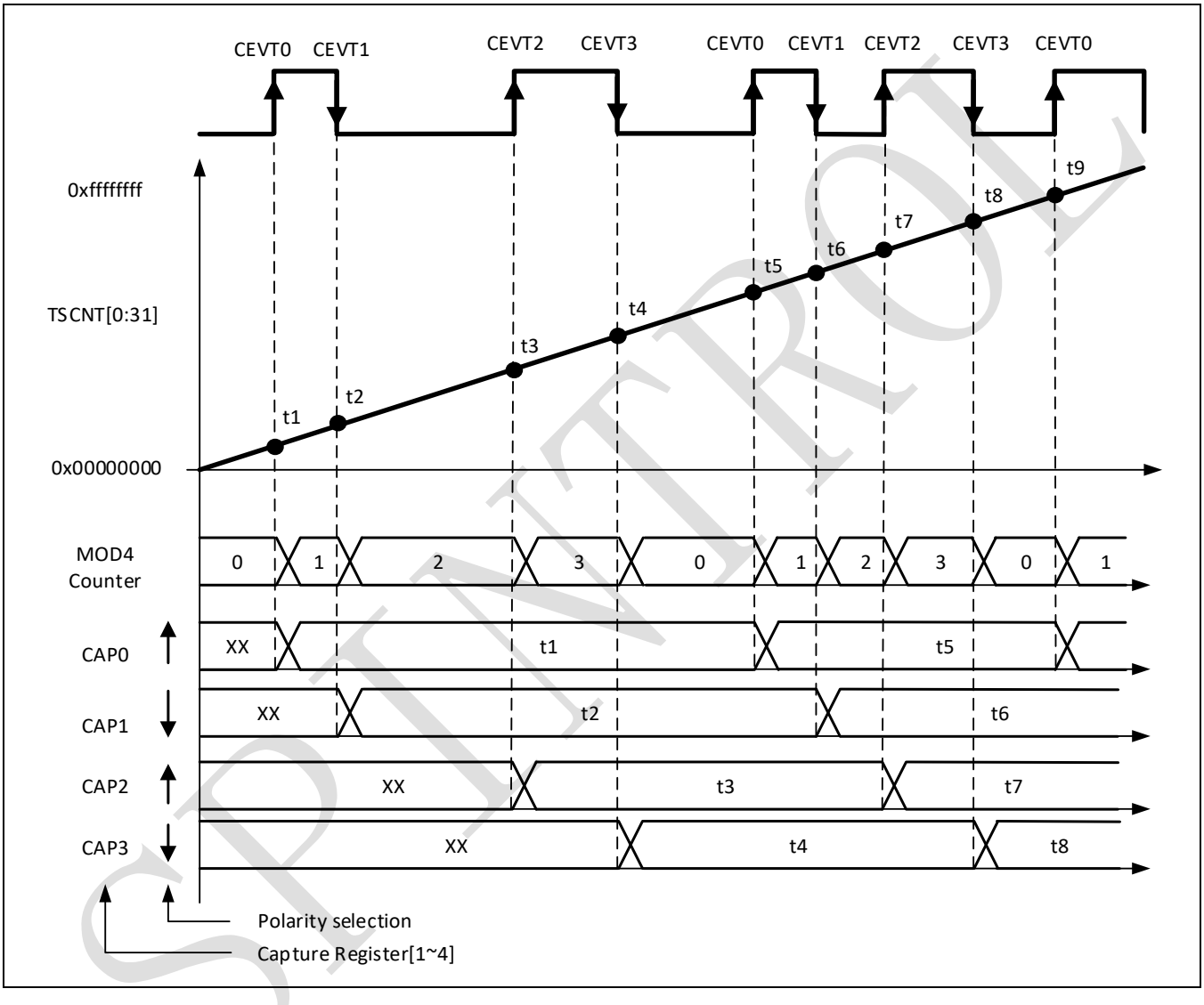
示例代码

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger
// Initialization Time
//======================
// Spintrol Co., Ltd.. All rights reserved.

ECAP->CAPCTL.bit.CAP0POL = EC_RISING;
ECAP->CAPCTL.bit.CAP1POL = EC_RISING;
ECAP->CAPCTL.bit.CAP2POL = EC_RISING;
ECAP->CAPCTL.bit.CAP3POL = EC_RISING;
ECAP->CAPCTL.bit.CNTRST0 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST1 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST2 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST3 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CAPLDEN = EC_ENABLE;
ECAP->CAPCTL.bit.EVTDIV = EC_DIV1;
ECAP->CAPCTL.bit.CAPAPWM = EC_CAP_MODE;
ECAP->CAPCTL.bit.MODE = EC_CONTINUOUS;
ECAP->CAPCTL.bit.SYNCOSEL = EC_SYNCO_DIS;
ECAP->CAPCTL.bit.SYNCIEN = EC_DISABLE;
ECAP->CAPCTL.bit.TSCNTRUN = EC_RUN; // Allow TSCNT to run

// Run Time (e.g. CEVT3 triggered ISR call)
//=========================================
TSt1 = ECAP->CAP0; // Fetch Time-Stamp captured at t1
TSt2 = ECAP->CAP1; // Fetch Time-Stamp captured at t2
TSt3 = ECAP->CAP2; // Fetch Time-Stamp captured at t3
TSt4 = ECAP->CAP3; // Fetch Time-Stamp captured at t4
Period1 = TSt2-TSt1; // Calculate 1st period
Period2 = TSt3-TSt2; // Calculate 2nd period
Period3 = TSt4-TSt3; // Calculate 3rd period
```

示例 2 – 由绝对时间戳捕获计算周期和占空

所示情形与上例几乎相同，只是捕获事件改为上升沿和下降沿交替，由此给出了周期和占空的信息。例如：周期 1 = $t_3 - t_1$，周期 2 = $t_5 - t_3$……。波形为高占空 = $t_2 - t_1$，波形为低占空 = $t_3 - t_2$。

图 11-12：通过绝对时间戳捕获计算周期和占空



示例代码

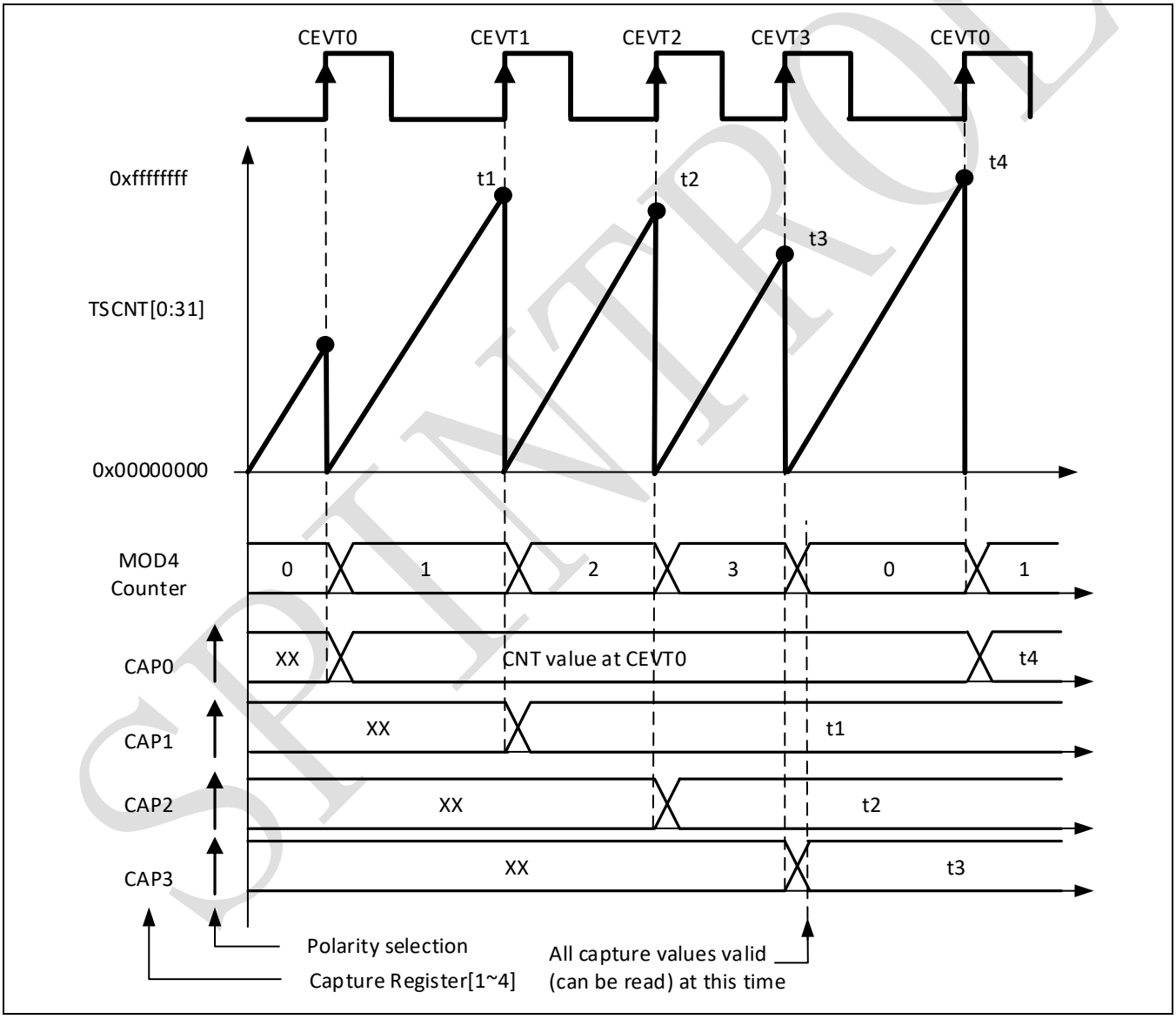
```
// Initialization Time
//=======================
// Spintrol Co., Ltd.. All rights reserved.

ECAP->CAPCTL.bit.CAP0POL = EC_RISING;
ECAP->CAPCTL.bit.CAP1POL = EC_FALLING;
ECAP->CAPCTL.bit.CAP2POL = EC_RISING;
ECAP->CAPCTL.bit.CAP3POL = EC_FALLING;
ECAP->CAPCTL.bit.CNTRST0 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST1 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST2 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CNTRST3 = EC_ABS_MODE;
ECAP->CAPCTL.bit.CAPLDEN = EC_ENABLE;
ECAP->CAPCTL.bit.EVTDIV = EC_DIV1;
ECAP->CAPCTL.bit.CAPAPWM = EC_CAP_MODE;
ECAP->CAPCTL.bit.MODE = EC_CONTINUOUS;
ECAP->CAPCTL.bit.SYNCOSEL = EC_SYNCO_DIS;
ECAP->CAPCTL.bit.SYNCIEN = EC_disable;
ECAP->CAPCTL.bit.TSCNTRUN = EC_RUN; // Allow TSCNT to run

// Run Time (e.g. CEVT3 triggered ISR call)
//=========================================
TSt1 = ECAP->CAP0; // Fetch Time-Stamp captured at t1
TSt2 = ECAP->CAP1; // Fetch Time-Stamp captured at t2
TSt3 = ECAP->CAP2; // Fetch Time-Stamp captured at t3
TSt4 = ECAP->CAP3; // Fetch Time-Stamp captured at t4
Period1 = TSt3-TSt1; // Calculate 1st period
DutyOnTime1 = TSt2-TSt1; // Calculate On time
DutyOffTime1 = TSt3-TSt2; // Calculate Off time
```

示例 3 – 通过计数器复位计算相对时间差

图 11-13 示例了 ECAP 如何从输入脉冲序列里获取时间差的信息。此处同样采用了持续捕获的模式并设置事件为上升沿有效。所不同的是，每次事件发生时，在捕获计数器当前值后都会复位计数器。

可以看出，这个方式的好处是 CAPx 寄存器的内容已经是所关心的周期信息而不需要 CPU 参与计算，即周期 1 = T_1，周期 2 = T_2 等。从图中还可以看出，CEVT0 适于作为中断触发源，因为此时 T_1、T_2、T_3、T_4 的信息都已经成功获得。

图 11-13：通过计数器复位捕获周期



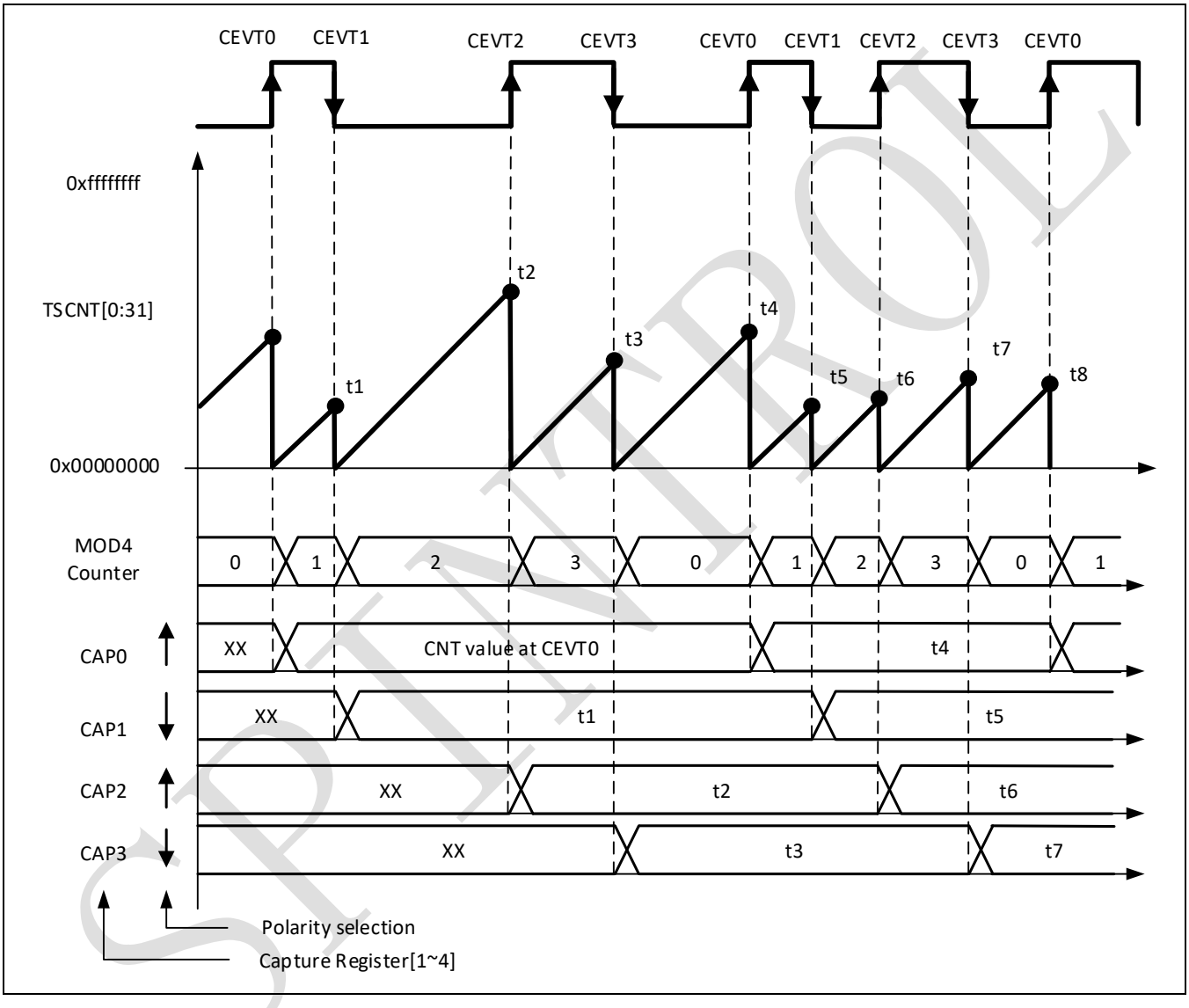
示例代码

```
// Initialization Time
//=======================
//Spintrol Co., Ltd.. All rights reserved.

ECAP->CAPCTL.bit.CAP0POL = EC_RISING;
ECAP->CAPCTL.bit.CAP1POL = EC_RISING;
ECAP->CAPCTL.bit.CAP2POL = EC_RISING;
ECAP->CAPCTL.bit.CAP3POL = EC_RISING;
ECAP->CAPCTL.bit.CNTRST0 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST1 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST2 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST3 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CAPLDEN = EC_ENABLE;
ECAP->CAPCTL.bit.EVTDIV = EC_DIV1;
ECAP->CAPCTL.bit.CAPAPWM = EC_CAP_MODE;
ECAP->CAPCTL.bit.MODE = EC_CONTINUOUS;
ECAP->CAPCTL.bit.SYNCOSEL = EC_SYNCO_DIS;
ECAP->CAPCTL.bit.SYNCIEN = EC_disable;
ECAP->CAPCTL.bit.TSCNTRUN = EC_RUN; // Allow TSCNT to run

// Run Time (e.g. CEVT0 triggered ISR call)
//=========================================
// Note: here Time-stamp directly represents the Period value.
Period3 = ECAP->CAP0; // Fetch Time-Stamp captured at T1
Period0 = ECAP->CAP1; // Fetch Time-Stamp captured at T2
Period1 = ECAP->CAP2; // Fetch Time-Stamp captured at T3
Period2 = ECAP->CAP3; // Fetch Time-Stamp captured at T4
```

示例 4 – 相对时间模式和双边沿检测

图 11-14 所示 ECAP 工作模式和上例几乎相同，只是捕获事件改为上升沿和下降沿交替，由此给出周期和占空信息。即周期 1 = T_1+T_2，周期 2 = T_3+T_4 等；波形高占空 = T_1，波形低占空 = T_2。

图 11-14：通过相对时间模式计数器复位捕获周期和占空



示例代码

```
// Initialization Time
//=======================
// Spintrol Co., Ltd.. All rights reserved

ECAP->CAPCTL.bit.CAP0POL = EC_RISING;
ECAP->CAPCTL.bit.CAP1POL = EC_FALLING;
ECAP->CAPCTL.bit.CAP2POL = EC_RISING;
ECAP->CAPCTL.bit.CAP3POL = EC_FALLING;
ECAP->CAPCTL.bit.CNTRST0 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST1 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST2 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CNTRST3 = EC_DELTA_MODE;
ECAP->CAPCTL.bit.CAPLDEN = EC_ENABLE;
ECAP->CAPCTL.bit.EVTDIV = EC_DIV1;
ECAP->CAPCTL.bit.CAPAPWM = EC_CAP_MODE;
ECAP->CAPCTL.bit.MODE = EC_CONTINUOUS;
ECAP->CAPCTL.bit.SYNCOSEL = EC_SYNCO_DIS;
ECAP->CAPCTL.bit.SYNCIEN = EC_disable;
ECAP->CAPCTL.bit.TSCNTRUN = EC_RUN; // Allow TSCNT to run

// Run Time (e.g. CEVT0 triggered ISR call)
//=========================================
// Note: here Time-stamp directly represents the Duty cycle values.
DutyOnTime1 = ECAP->CAP1; // Fetch Time-Stamp captured at T2
DutyOffTime1 = ECAP->CAP2; // Fetch Time-Stamp captured at T3
DutyOnTime2 = ECAP->CAP3; // Fetch Time-Stamp captured at T4
DutyOffTime2 = ECAP->CAP0; // Fetch Time-Stamp captured at T1
Period1 = DutyOnTime1 + DutyOffTime1;
Period2 = DutyOnTime2 + DutyOffTime2;
```

11.7 寄存器

11.7.1 ECAP 寄存器列表

表 11-1：ECAP 模块基地址

外设模块	基地址
ECAP	0x4000 A000

表 11-2：ECAP 寄存器列表

寄存器	偏移地址	描述	复位值
CAPSRCCTL	0x00	捕获信号输入控制寄存器	0x0000007F
CAPSYNCICTL	0x04	同步信号输入控制寄存器	0x0000007F
CAPTSCNT	0x08	时间戳计数寄存器	0x00000000
CAPCNTPHS	0x0C	时间戳计数相位寄存器	0x00000000
CAP0	0x10	捕获值 0 寄存器	0x00000000
CAP1	0x14	捕获值 1 寄存器	0x00000000
CAP2	0x18	捕获值 2 寄存器	0x00000000
CAP3	0x1C	捕获值 3 寄存器	0x00000000
CAPCTL	0x20	捕获控制寄存器	0x000C0000
CAPIF	0x24	捕获中断标志寄存器	0x00000000
CAPIE	0x28	捕获中断使能寄存器	0x00000000
CAPIC	0x2C	捕获中断清除寄存器	0x00000000
CAPIFRC	0x30	捕获中断强制寄存器	0x00000000
CAPREGKEY	0x34	ECAP 模块写使能寄存器	0x1ACCE551

注意： 由*标识的寄存器仅当 CAPREGKEY=0x1ACCE551 时才可以改写。

11.7.2 ECAP 寄存器

表 11-3 到表 11-30 给出了 ECAP 模块寄存器的定义。

表 11-3：捕获信号输入控制寄存器（CAPSRCCTL）位段定义

CAPSRCCTL (Capture Source Input Control Register) Offset: 0x0 Default: 0x0000007F Access: ECAP -> CAPSRCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 11-4：捕获信号输入控制寄存器（CAPSRCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	输入信号极性 0：输入低时有效 1：输入高时有效
5:0	IOSEL	RW	0x3F	作为捕获信号输入的 GPIO 端口号

表 11-5：同步信号输入控制寄存器（CAPSYNCICTL）位段定义

CAPSYNCICTL (Capture Synchronization Source Input Control Register) Offset: 0x4 Default: 0x0000007F Access: ECAP -> CAPSYNCICTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 11-6：同步信号输入控制寄存器（CAPSYNCICTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	同步信号极性 0：输入低时触发同步 1：输入高时触发同步
5:0	IOSEL	RW	0x3F	作为同步信号输入的 GPIO 端口号

表 11-7：时间戳计数寄存器（CAPTSCNT）位段定义

CAPTSCNT (Time-Stamp Counter Register) Offset: 0x8 Default: 0x00000000 Access: ECAP -> CAPTSCNT.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-8：时间戳计数寄存器（CAPTSCNT）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	32 位计数器，用作捕获时的时间戳

表 11-9：时间戳相位寄存器（CAPCNTPHS）位段定义

CAPCNTPHS (Counter Phase Offset Value Register) Offset: 0xC Default: 0x00000000 Access: ECAP -> CAPCNTPHS.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-10：时间戳相位寄存器（CAPCNTPHS）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	相位值

表 11-11：捕获值 0 寄存器（CAP0）位段定义

CAP0 (Capture Register 0) Offset: 0x10 Default: 0x00000000 Access: ECAP -> CAP0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-12：捕获值 0 寄存器（CAP0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	本寄存器可以用在以下几种场景： - 捕获事件发生时的时间戳计数值 - 软件强制捕获时间戳计数值（用于软件调试） - APWM 模式下用作 PRD（CAP2）有效值寄存器 注意：APWM 模式下，写 CAP0/CAP1 同时更新 CAP2/CAP3 的值，从而模拟直接写入模式。写 CAP2/CAP3 寄存器触发影子模式。

表 11-13：捕获值 1 寄存器（CAP1）位段定义

CAP1 (Capture Register 1) Offset: 0x14 Default: 0x00000000 Access: ECAP -> CAP1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-14：捕获值 1 寄存器（CAP1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	本寄存器可以用在以下几种场景： - 捕获事件发生时的时间戳计数值 - 软件强制捕获时间戳计数值（用于软件调试） - APWM 模式下用作 CMP（CAP3）有效值寄存器 注意：APWM 模式下，写 CAP0/CAP1 同时更新 CAP2/CAP3 的值，从而模拟直接写入模式。写 CAP2/CAP3 寄存器触发影子模式。

表 11-15：捕获值 2 寄存器（CAP2）位段定义

CAP2 (Capture Register 2) Offset: 0x18 Default: 0x00000000 Access: ECAP -> CAP2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-16：捕获值 2 寄存器（CAP2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	捕获模式下，记录捕获事件发生时时间戳计数值 APWM 模式下，用作周期 PRD 的影子寄存器。用户可以写本寄存器更改周期值，其有效值存于 CAP0，即 CAP2 是 CAP0 的影子寄存器。

表 11-17：捕获值 3 寄存器（CAP3）位段定义

CAP3 (Capture Register 3) Offset: 0x1C Default: 0x00000000 Access: ECAP -> CAP3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 11-18：捕获值 3 寄存器（CAP3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	捕获模式下，记录捕获事件发生时时间戳计数值 APWM 模式下，用作计数比较值 CMP 的影子寄存器。用户可以写本寄存器更改计数比较值，其有效值存于 CAP1，即 CAP3 是 CAP1 的影子寄存器。

表 11-19：捕获控制寄存器（CAPCTL）位段定义

CAPCTL (Capture Control Register) Offset: 0x20 Default: 0x000C0000 Access: ECAP -> CAPCTL.all							
31	30	29	28	27	26	25	24
RESERVED		DBGRUN		APWMPOL	APWMMODE	FRCSYNC	SYNCOSEL
23	22	21	20	19	18	17	16
SYNCOSEL	PHSEN	TSCNTRUN	REARM	STOPWRAP		ONESHOT	EVTDIV
15	14	13	12	11	10	9	8
EVTDIV							CAPLDEN
7	6	5	4	3	2	1	0
CNTRST3	CAP3POL	CNTRST2	CAP2POL	CNTRST1	CAP1POL	CNTRST0	CAP0POL

表 11-20：捕获控制寄存器（CAPCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29:28	DBGRUN	RW	0x0	由于调试或异常错误导致的 CPU 停止事件发生时的时间戳计数器的运行方式 00：立即停止 01：计数到 0 时停止 1x：保持正常计数
27	APWMPOL	RW	0x0	APWM 输出信号极性。本位段仅在 APWM 模式有效 0：输出低有效（比较值定义了输出低的时间） 1：输出高有效（比较值定义了输出高的时间）
26	APWMMODE	RW	0x0	捕获和 APWM 模式选择 0：捕获模式 在该模式下： - 关闭 CAPTSCNT 计数到 PRD 时的自动复位 - 关闭 CAP2/CAP3 对 CAP0/CAP1 的影子功能 - 允许用户使能 CAP0~CAP3 捕获时间戳计数 - CAPx 输入端口生效 1：APWM 模式 在该模式下： - CAPTSCNT 计数到 PRD 时复位为 0 继续计数 - 开启 CAP2/CAP3 对 CAP0/CAP1 的影子功能 - 禁止捕获时间戳计数到 CAP0~CAP3 寄存器 - APWMx 输出端口生效
25	FRCSYNC	W1S	0x0	软件强制时间戳计数同步 0：写 0 无效，总是读回 0 1：写 1 将 CAPTSCNT 重置为 CAPCNTPHS 本位段自动清零

位段	位段名	属性	复位值	描述
24:23	SYNCOSEL	RW	0x0	同步输出选择 0x：禁用同步输出 10：选择同步输入信号作为输出（直通） 11：选择 CNT = PRD 时间作为同步输出
22	PHSEN	RW	0x0	相位同步使能 0：关闭相位同步 1：使能相位同步，在输入同步事件或者软件强制的同步事件发生时，将 CAPTSCNT 重置为 CAPCNTPHS
21	TSCNTRUN	RW	0x0	时间戳计数器运行控制 0：停止计数 1：保持计数
20	REARM	W1S	0x0	一次性状态归位控制 在一次性捕获模式下，将阻止后续事件和中断的产生，直至状态归位 0：写 0 无效，总是读回 0 1：按如下顺序归位一次性捕获状态 1）复位 MOD4 捕获事件计数到 0 2）解锁 MOD4 计数 3）使能计数捕获
19:18	STOPWRAP	RW	0x3	一次性捕获模式下的停止值或持续捕获模式下的环回值 当 MOD4 计数到 STOPWRAP 时： - MOD4 计数停止（锁定） - 阻止后续的时机计数捕获 00：一次性捕获模式下事件 0 发生后即停止 持续捕获模式下事件 0 发生后重置事件计数 01：一次性捕获模式下事件 1 发生后即停止 持续捕获模式下事件 1 发生后重置事件计数 10：一次性捕获模式下事件 2 发生后即停止 持续捕获模式下事件 2 发生后重置事件计数 11：一次性捕获模式下事件 3 发生后即停止 持续捕获模式下事件 3 发生后重置事件计数
17	ONESHOT	RW	0x0	一次性捕获模式使能 本位段仅在捕获模式下有效。 0：持续捕获模式 1：一次性捕获模式使能
16:9	EVTDIV	RW	0x0	事件滤波预分频 每 EVTDIV+1 次事件触发一次捕获

位段	位段名	属性	复位值	描述
8	CAPLDEN	RW	0x0	使能 CAP0~CAP3 的捕获功能 0：关闭 1：使能
7	CNTRST3	RW	0x0	捕获事件 3 发生时复位时间戳计数 0：捕获事件 3 发生时不复位时间戳计数（绝对时间模式） 1：捕获事件 3 发生时复位时间戳计数（相对时间模式）
6	CAP3POL	RW	0x0	捕获事件 3 极性选择 0：输入下降沿时触发捕获事件 3 1：输入上升沿时触发捕获事件 3
5	CNTRST2	RW	0x0	捕获事件 2 发生时复位时间戳计数 0：捕获事件 2 发生时不复位时间戳计数（绝对时间模式） 1：捕获事件 2 发生时复位时间戳计数（相对时间模式）
4	CAP2POL	RW	0x0	捕获事件 2 极性选择 0：输入下降沿时触发捕获事件 2 1：输入上升沿时触发捕获事件 2
3	CNTRST1	RW	0x0	捕获事件 1 发生时复位时间戳计数 0：捕获事件 1 发生时不复位时间戳计数（绝对时间模式） 1：捕获事件 1 发生时复位时间戳计数（相对时间模式）
2	CAP1POL	RW	0x0	捕获事件 1 极性选择 0：输入下降沿时触发捕获事件 1 1：输入上升沿时触发捕获事件 1
1	CNTRST0	RW	0x0	捕获事件 0 发生时复位时间戳计数 0：捕获事件 0 发生时不复位时间戳计数（绝对时间模式） 1：捕获事件 0 发生时复位时间戳计数（相对时间模式）
0	CAP0POL	RW	0x0	捕获事件 0 极性选择 0：输入下降沿时触发捕获事件 0 1：输入上升沿时触发捕获事件 0

表 11-21：捕获中断标志寄存器（CAPIF）位段定义

CAPIF (Capture Interrupt Flag Register) Offset: 0x24 Default: 0x00000000 Access: ECAP -> CAPIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 INT	6 CMP	5 PRD	4 CNTOVF	3 CEVT3	2 CEVT2	1 CEVT1	0 CEVT0

表 11-22：捕获中断标志寄存器（CAPIF）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	INT	RO	0x0	全局中断标志 0：中断未触发 1：中断触发。本位段清零前，将不会产生新的中断。
6	CMP	RO	0x0	计数比较相等事件标志 本位段仅在 APWM 模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的计数比较相等事件将不会触发新的中断
5	PRD	RO	0x0	计数到 PRD 事件标志 本位段仅在 APWM 模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的计数到 PRD 事件将不会触发新的中断。
4	CNTOVF	RO	0x0	计数溢出事件（计数到 0xFFFFFFFF）标志 本位段在 APWM 模式和 CAP 模式下均有效。 0：事件未发生 1：事件发生。本位段清零前，新的计数溢出事件将不会触发新的中断。
3	CEVT3	RO	0x0	捕获事件 3 标志 本位段仅在捕获模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的捕获事件 3 将不会触发新的中断。

位段	位段名	属性	复位值	描述
2	CEVT2	RO	0x0	捕获事件 2 标志 本位段仅在捕获模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的捕获事件 2 将不会触发新的中断。
1	CEVT1	RO	0x0	捕获事件 1 标志 本位段仅在捕获模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的捕获事件 1 将不会触发新的中断。
0	CEVT0	RO	0x0	捕获事件 0 标志 本位段仅在捕获模式下有效。 0：事件未发生 1：事件发生。本位段清零前，新的捕获事件 0 将不会触发新的中断。

表 11-23：捕获中断使能寄存器（CAPIE）位段定义

CAPIE (Capture Interrupt Enable Register)　Offset: 0x28　Default: 0x00000000
Access: ECAP -> CAPIE.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	CMP	PRD	CNTOVF	CEVT3	CEVT2	CEVT1	CEVT0

表 11-24：捕获中断使能寄存器（CAPIE）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	CMP	RW	0x0	计数比较相等事件中断使能 0：关闭中断 1：使能中断
5	PRD	RW	0x0	计数到 PRD 事件中断使能 0：关闭中断 1：使能中断
4	CNTOVF	RW	0x0	计数溢出（计数到 0xFFFFFFFF）事件中断使能 0：关闭中断 1：使能中断

位段	位段名	属性	复位值	描述
3	CEVT3	RW	0x0	捕获事件 3 中断使能 0：关闭中断 1：使能中断
2	CEVT2	RW	0x0	捕获事件 2 中断使能 0：关闭中断 1：使能中断
1	CEVT1	RW	0x0	捕获事件 1 中断使能 0：关闭中断 1：使能中断
0	CEVT0	RW	0x0	捕获事件 0 中断使能 0：关闭中断 1：使能中断

表 11-25：捕获中断清除寄存器（CAPIC）位段定义

CAPIC (Capture Interrupt Clear Register) Offset: 0x2C Default: 0x00000000 Access: ECAP -> CAPIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 INT	6 CMP	5 PRD	4 CNTOVF	3 CEVT3	2 CEVT2	1 CEVT1	0 CEVT0

表 11-26：捕获中断清除寄存器（CAPIC）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	INT	W1C	0x0	全局中断清除 0：写 0 无效，总是读回 0 1：写 1 清除中断和标志位。本位段自动清零。
6	CMP	W1C	0x0	计数比较相等标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。
5	PRD	W1C	0x0	计数至 PRD 标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。
4	CNTOVF	W1C	0x0	计数溢出标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。

位段	位段名	属性	复位值	描述
3	CEVT3	W1C	0x0	捕捉时间 3 标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。
2	CEVT2	W1C	0x0	捕捉时间 2 标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。
1	CEVT1	W1C	0x0	捕捉时间 1 标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。
0	CEVT0	W1C	0x0	捕捉时间 0 标志清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位。本位段自动清零。

表 11-27：捕获中断强制寄存器（CAPIFRC）位段定义

CAPIFRC (Capture Interrupt Force Register) Offset: 0x30 Default: 0x00000000 Access: ECAP -> CAPIFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	CMP	PRD	CNTOVF	CEVT3	CEVT2	CEVT1	CEVT0

表 11-28：捕获中断强制寄存器（CAPIFRC）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	CMP	W1S	0x0	计数比较相等事件软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。
5	PRD	W1S	0x0	计数到 PRD 事件软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。
4	CNTOVF	W1S	0x0	计数溢出事件软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。

位段	位段名	属性	复位值	描述
3	CEVT3	W1S	0x0	捕获事件 3 软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。
2	CEVT2	W1S	0x0	捕获事件 2 软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。
1	CEVT1	W1S	0x0	捕获事件 1 软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。
0	CEVT0	W1S	0x0	捕获事件 0 软件强制 0：写 0 无效，总是读回 0 1：写 1 强制产生一个事件。本位段自动清零。

表 11-29：ECAP 模块写保护寄存器（CAPREGKEY）位段定义

CAPREGKEY (Capture Register Write-Allow Key Register) Offset: 0x34 Default: 0x1ACCE551 Access: ECAP -> CAPREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 11-30：ECAP 模块写保护寄存器（CAPREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 ECAP 寄存器

12 模数转换器（ADC）

12.1 ADC 概述

SPC2168 内嵌了一个 14 位逐次逼近型模数转换器（Successive Approximation A/D Converter，SAR-ADC）。其中，ADC 前端包含三个采样保持电路，他们可以依次运行，也可以同时运行。芯片有 20 个模拟输入通道可以送入 ADC 采样。ADC 的基准电压有内嵌和外灌两种方式。

用户很容易利用 ADC 对一个触发源产生一系列的转换。基本的使用原理是基于许多套转换的独立配置，我们把它称为转换状态配置（State-Of-Conversions，SOC）。

ADC 模块主要包含以下功能：

- 14 位分辨率转换内核以及三个采样保持电路（Sample-and-hold circuit，SH）
  - 支持高达 4 兆次/秒采样率
  - 支持多达 20 个模拟输入通道的采样
- 支持 3 个采样保持电路同时采样
- 模拟满幅度输入范围：3.657V（内部基准源），或者和外部输入基准源成比例（外灌输入管脚为 GPIO12）
- 16 套 SOC 配置，每套配置包含触发源，采样模式，通道配置，平均次数，采样时间
- 16 个结果寄存器（独立寻址）用于储存转换结果
- 多种触发源可供选择
  - 软件触发
  - 通用定时器 Timer0/1/2 触发
  - GPIO 外部触发
  - PWMxSOCA、PWMxSOCB 和 PWMxSOCC（x= 0 ~ 7）触发，时序可配
- 16 个独立的触发信号可以触发 16 个 NVIC 中断

12.2 ADC 架构

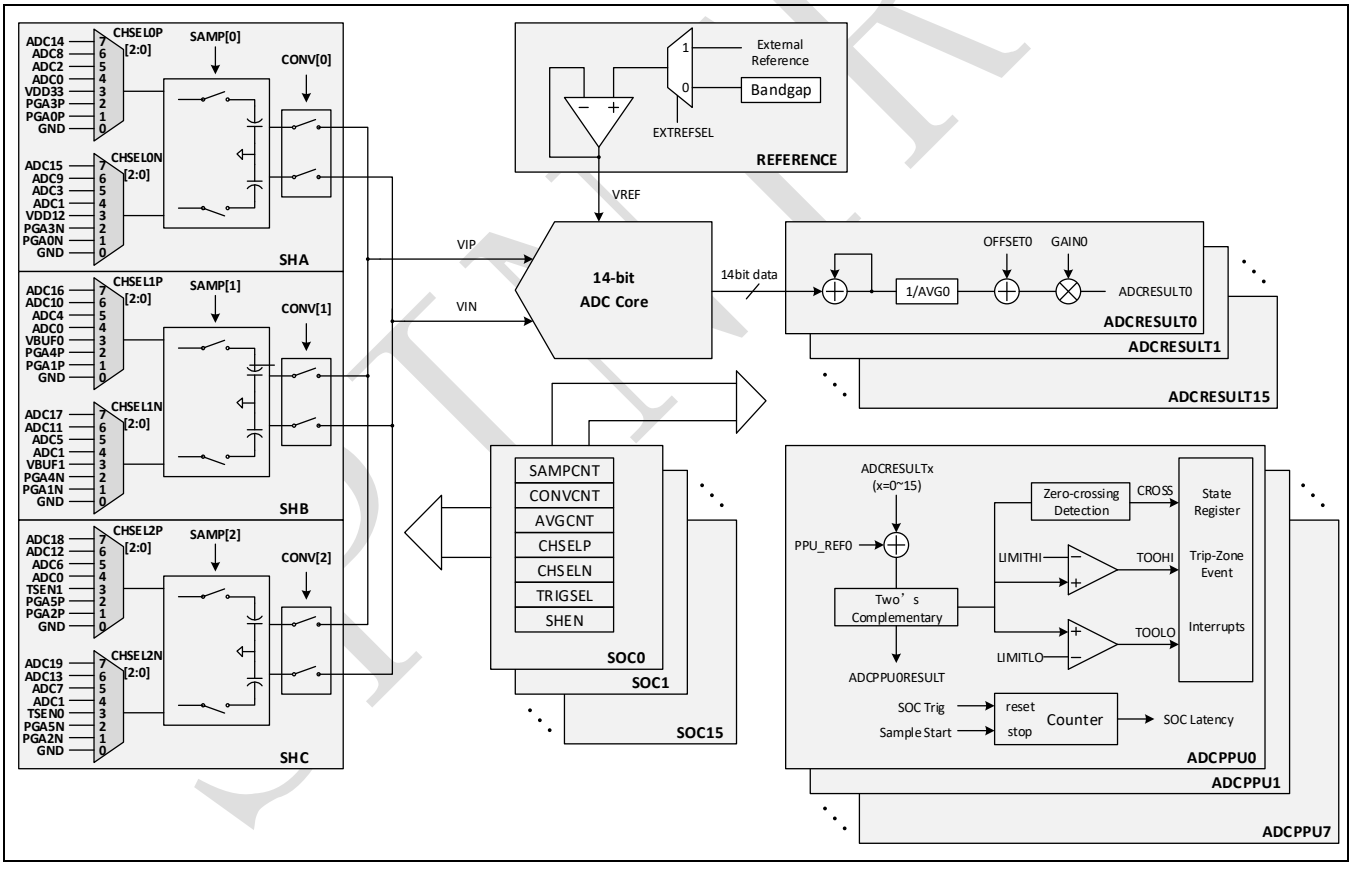
如图 12-1 所示，ADC 模拟电路包括：

- 3 个采样保持器（SHA，SHB，SHC）
- 1 个 ADC 内核
- ADC 基准电路

数字电路包括:

- 16 套 SOC 配置
- 16 个 ADC 结果寄存器（ADCRESULT），该结果已经经过平均，增益和误差校准处理。
- 8 个数字后处理单元（Post-processing units，PPU）
- 与芯片其它模块的接口（图中未展示）

图 12-1：ADC 内部模块框图



12.3 SOC 工作原理

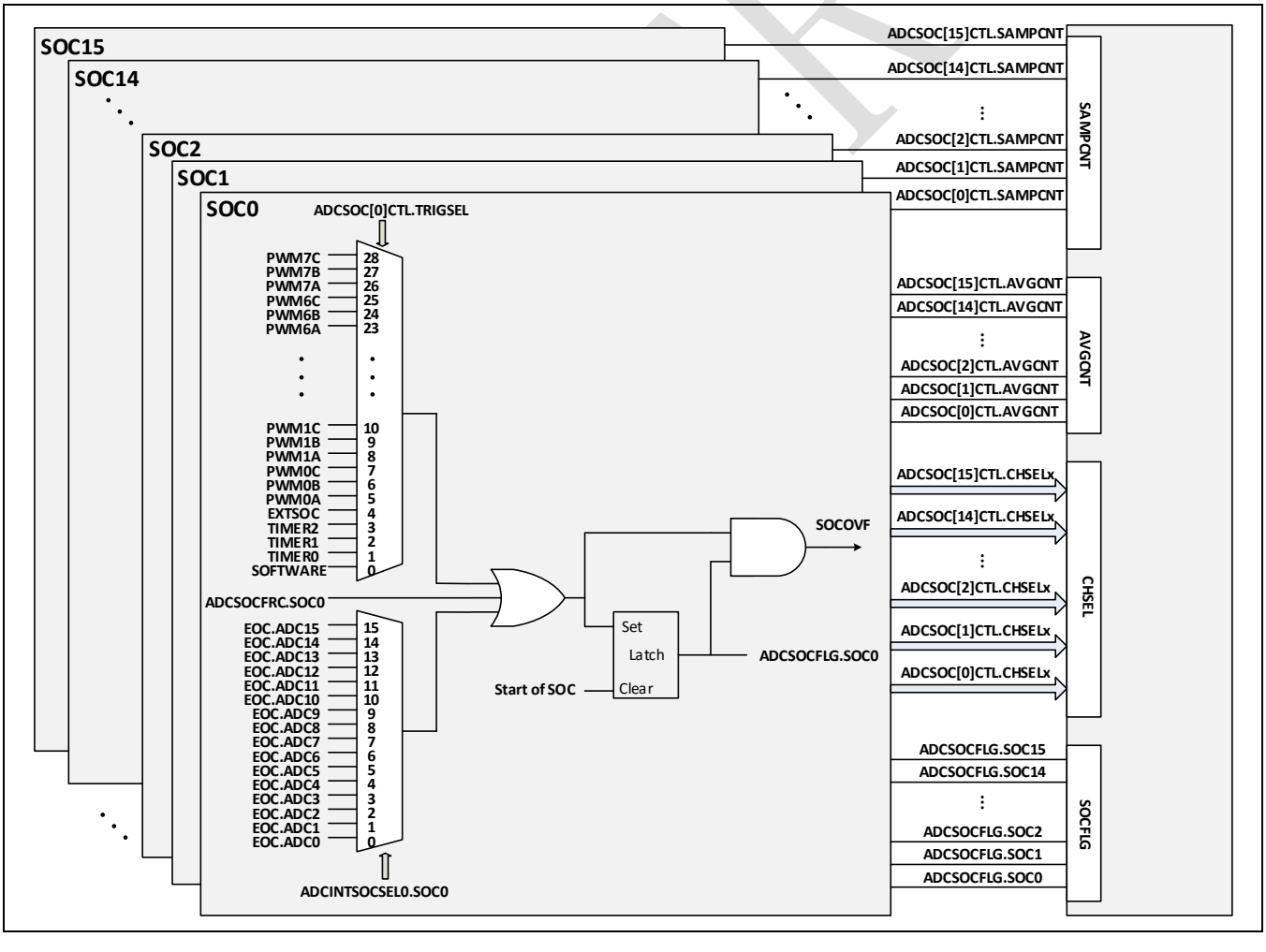
ADC 基于 SOC 的配置运行。每套 SOC 配置对应一个独立的转换。在一个 SOC 里面主要包含四部分设置：设定转换开始的触发源，选择需要转换的输入通道，平均化的次数以及采样时间窗口大小。除此以外，采样模式的控制也被包含在 SOC 的配置里面（这部分在章节 12.3.2 里面有详细描述）。如图 12-2 所示，每套 SOC 都已经被设置了触发源，通道选择，平均次数以及采样窗口大小。多个 SOC 也可以被设置成相同的触发源，输入通道，平均次数以及采样时间。这样，ADC 既支持利用不同触发源采样不同通道，也支持用相同触发源对同一通道过采样，或者用相同触发源对不同通道完成一系列转换。

SOC 的触发信号通过寄存器 ADCSOCCTL[x]的 TRIGSEL 位段和寄存器 ADCINTSOCSEL0~1 共同设置。软件也可以作为触发源，通过寄存器 ADCSOCFRC 强制触发 SOC。

SOC 的通道选择通过寄存器 ADCSOCCTL[x]的 SHEN 位段和 CHSELx（x=P 或者 N）共同设置。

SOC 的采样时间则是通过寄存器 ADCSOCCTL[x]的 SAMPCNT 位段设置。

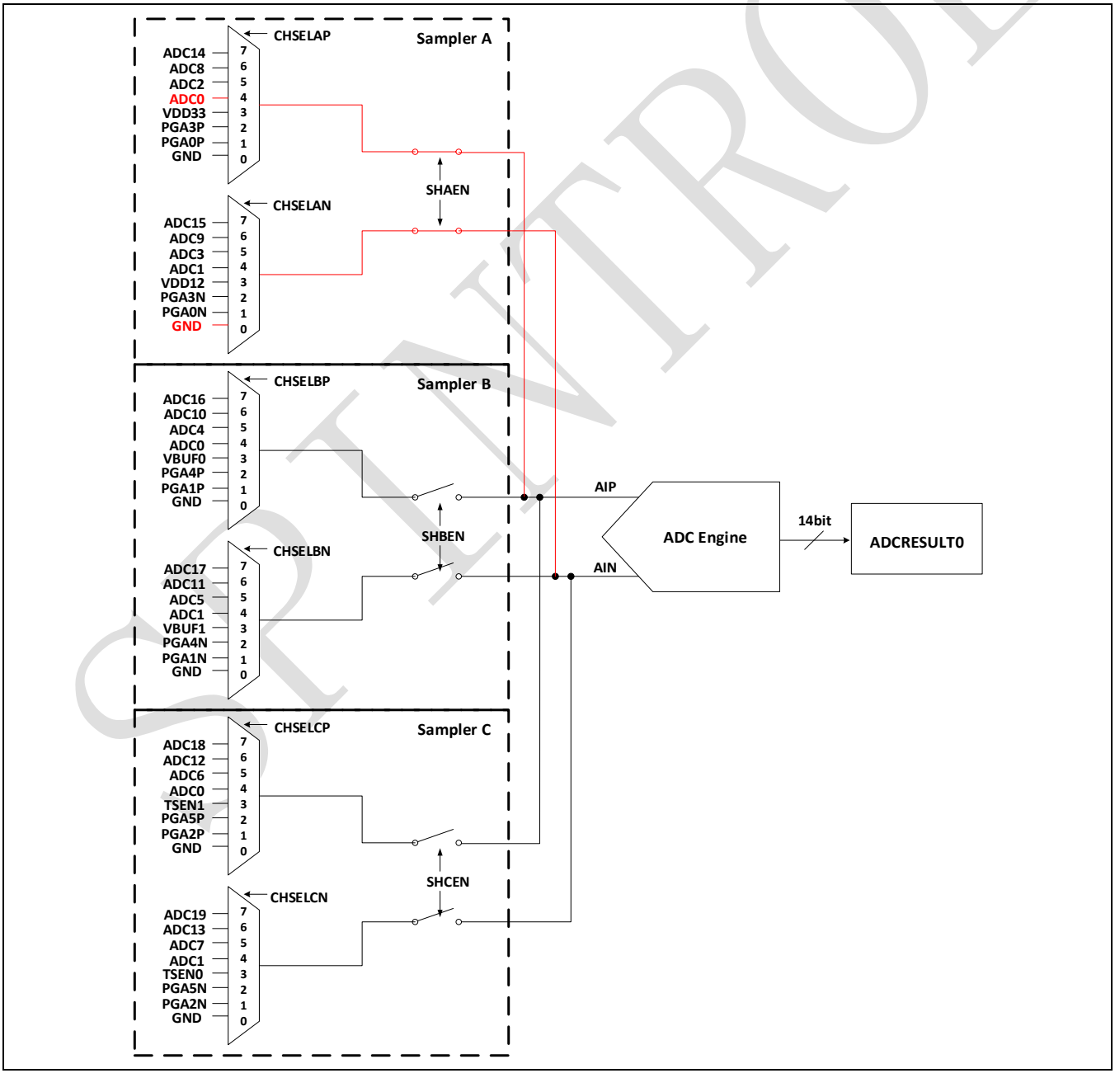
图 12-2：SOC 配置框图



示例 12.3.1 PWMSOC1A 触发 SOC0 转换 ADC0 输入

这个例子实现的是当 PWM1SOCA 的计数器到达周期设定值，触发 SOC 转换 ADC0 的输入信号。由于 16 套 SOC 配置设计完全相同，所以我们可以任意选择，在这里例子中，我们用 SOC0 的寄存器 ADCSOCCTL[0]配置。设置 SOC0 的触发源为 PWM1SOCA；选择平均次数为 4，这样 SOC0 会重复 4 次，并将平均值存入寄存器 ADCRESULT[0]；选择 ADC0 作为采样通道；设定采样时间为 4 个 ADC 周期，转换时间也为 4 个 ADC 周期。因此，我们在 ADCSOCCTL[0]中，设置位段 TRIGSEL 为 8，设置位段 AVGCNT 为 2（2 的 2 次方），设置位段 SHEN 为 1（只有采样保持器 A 工作），设置位段 CHSELP 为 4（选择 ADC0），设置位段 CHSELN 为 0（选择 GND），最后，分别设置位段 SAMPCNT 为 3 和 CONVCNT 为 3。

图 12-3：示例 12.3.1 的 ADC 通道选择



示例中的配置过程可以被拆分成如下代码：

示例 12.3.1

```
void  ADC_Example12_3_1(void)
{
  ADC->ADCSOCCTL[0].bit.SHEN    = 1;/* Enable Sampler A                    */
  ADC->ADCSOCCTL[0].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source  */
  ADC->ADCSOCCTL[0].bit.CHSELP  = 4;/* Sampler A Positive = ADC0           */
  ADC->ADCSOCCTL[0].bit.CHSELN  = 0;/* Sampler A Negative = GND            */
  ADC->ADCSOCCTL[0].bit.AVGCNT  = 2;/* Averaging 4 times (2^2)             */
  ADC->ADCSOCCTL[0].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock     */
  ADC->ADCSOCCTL[0].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock   */
}
```

假设 ADC 的时钟为 32MHz，周期为 31.25ns。所以采样时间最小为 125ns（SAMPCNT=3），转换时间最小为 156.25ns（CONVCNT=4）。只要 SOC0 收到 PWM1SOCA 事件触发就会转换 ADC0，并将结果存储在 ADCRESULT[0]里面。我们的 ADC 本质上是差分输入的，所以输出码值也是有正有负。在这个例子中，ADC 的正端输入为 ADC0，负端输入为内部 GND，这样转换结果是正的。如果我们在 ADC 正端接 GND，负端接其它 ADC 通道，那么转换结果就是负的。

示例 12.3.2　ADC0 过采样

如果 ADC0 需要被过采样 3 次，我们可以通过将 SOC1/SOC2 配置成和 SOC0 来一样实现。

示例 12.3.2

```
void  ADC_Example12_3_2(void)
{
  ADC->ADCSOCCTL[0].bit.SHEN    = 1;/* Enable Sampler A                    */
  ADC->ADCSOCCTL[0].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source  */
  ADC->ADCSOCCTL[0].bit.CHSELP  = 4;/* Sampler A Positive = ADC0           */
  ADC->ADCSOCCTL[0].bit.CHSELN  = 0;/* Sampler A Negative = GND            */
  ADC->ADCSOCCTL[0].bit.AVGCNT  = 2;/* Averaging 4 times (2^2)             */
  ADC->ADCSOCCTL[0].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock     */
  ADC->ADCSOCCTL[0].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock   */

  ADC->ADCSOCCTL[1].bit.SHEN    = 1;/* Enable Sampler A                    */
  ADC->ADCSOCCTL[1].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source  */
  ADC->ADCSOCCTL[1].bit.CHSELP  = 4;/* Sampler A Positive = ADC0           */
  ADC->ADCSOCCTL[1].bit.CHSELN  = 0;/* Sampler A Negative = GND            */
  ADC->ADCSOCCTL[1].bit.AVGCNT  = 2;/* Averaging 4 times (2^2)             */
  ADC->ADCSOCCTL[1].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock     */
  ADC->ADCSOCCTL[1].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock   */

  ADC->ADCSOCCTL[2].bit.SHEN    = 1;/* Enable Sampler A                    */
  ADC->ADCSOCCTL[2].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source  */
  ADC->ADCSOCCTL[2].bit.CHSELP  = 4;/* Sampler A Positive = ADC0           */
  ADC->ADCSOCCTL[2].bit.CHSELN  = 0;/* Sampler A Negative = GND            */
  ADC->ADCSOCCTL[2].bit.AVGCNT  = 2;/* Averaging 4 times (2^2)             */
  ADC->ADCSOCCTL[2].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock     */
  ADC->ADCSOCCTL[2].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock   */
}
```

按照之前的 SOC0 配置 SOC1 和 SOC2 之后，当 PWM1SOCA 事件触发后，ADC0 就会被转换三次，这三次的结果会分别储存在 ADCRESULT[0] ~ ADCRESULT[2]中。

示例 12.3.3 同样的触发事件转换不同的通道（ADC0，ADC2，ADC8）

示例 12.3.3

```
void   ADC_Example12_3_3(void)
{
   ADC->ADCSOCCTL[0].bit.SHEN        = 1;/* Enable Sampler A                              */
   ADC->ADCSOCCTL[0].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source   */
   ADC->ADCSOCCTL[0].bit.CHSELP = 4;/* Sampler A Positive = ADC0                  */
   ADC->ADCSOCCTL[0].bit.CHSELN = 0;/* Sampler A Negative = GND                   */
   ADC->ADCSOCCTL[0].bit.AVGCNT = 2;/* Averaging 4 times (2^2)                    */
   ADC->ADCSOCCTL[0].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock          */
   ADC->ADCSOCCTL[0].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock        */

   ADC->ADCSOCCTL[1].bit.SHEN        = 1;/* Enable Sampler A                              */
   ADC->ADCSOCCTL[1].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source   */
   ADC->ADCSOCCTL[1].bit.CHSELP = 5;/* Sampler A Positive = ADC2                  */
   ADC->ADCSOCCTL[1].bit.CHSELN = 0;/* Sampler A Negative = GND                   */
   ADC->ADCSOCCTL[1].bit.AVGCNT = 2;/* Averaging 4 times (2^2)                    */
   ADC->ADCSOCCTL[1].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock          */
   ADC->ADCSOCCTL[1].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock        */

   ADC->ADCSOCCTL[2].bit.SHEN        = 1;/* Enable Sampler A                              */
   ADC->ADCSOCCTL[2].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source   */
   ADC->ADCSOCCTL[2].bit.CHSELP = 6;/* Sampler A Positive = ADC8                  */
   ADC->ADCSOCCTL[2].bit.CHSELN = 0;/* Sampler A Negative = GND                   */
   ADC->ADCSOCCTL[2].bit.AVGCNT = 2;/* Averaging 4 times (2^2)                    */
   ADC->ADCSOCCTL[2].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock          */
   ADC->ADCSOCCTL[2].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock        */
}
```

按照代码所示配置之后，当 PWM1SOCA 事件触发，SOC0、SOC1、SOC2 顺序执行。相应地，ADC0 的结果被存储在 ADCRESULT[0]，ADC2 的结果被存储在 ADCRESULT[1]，ADC8 的结果被存储在 ADCRESULT[2]。结果的存储只与 SOC 对应，与转换什么通道，用什么触发源触发没有对应关系。

从示例 12.3.3 可以看到，如果应用需要用同一触发源采样三个不同的信号，只需要在 SOC0~2 中配置不同的通道选择，同时保持触发源相同即可。

12.3.1 触发源

每个 SOC 都需要配置触发源，触发源有多种选择。这些 SOC 可以配成相同的触发源，也可以配置成不同的触发源。下面是 SOC 可以选择的触发源类型：

- 软件触发
- 通用定时器 Timer0/1/2 触发
- 外部 GPIO 触发
- PWMxSOCA、PWMxSOCB 和 PWMxSOCC（x= 0 ~ 7）触发

表 12-1：触发源选择

TRIGSEL	触发源
其他	无效选择
28	PWM7SOCC
27	PWM7SOCB
26	PWM7SOCA
25	PWM6SOCC
24	PWM6SOCB
23	PWM6SOCA
22	PWM5SOCC
21	PWM5SOCB
20	PWM5SOCA
19	PWM4SOCC
18	PWM4SOCB
17	PWM4SOCA
16	PWM3SOCC
15	PWM3SOCB
14	PWM3SOCA
13	PWM2SOCC
12	PWM2SOCB
11	PWM2SOCA
10	PWM1SOCC
9	PWM1SOCB
8	PWM1SOCA
7	PWM0SOCC
6	PWM0SOCB
5	PWM0SOCA
4	EXTSOC
3	Timer 2
2	Timer 1
1	Timer 0
0	Software

表 12-1 列出了可用的触发源。如果 TRIGSEL 等于 4，则选择 EXTSOC（外部 SOC 事件）作为 SOC 触发源。有关这些触发器的配置详细信息，请参考 ADCSOCCTL [x]寄存器的位定义。有关如何选择 GPIO 引脚作为外部 SOC 事件源（ADCEXTSOCCTL）的信息，请参见表 12-38 和表 12-39。

12.3.2 ADC 采样模式和通道选择

SHEN 位段控制 ADC 的采样模式。如表 12-2 所示，当 SHEN 等于 0~3 时，所有的 SOC 的采样模式相同，都为顺序采样模式。当 SHEN 等于 4~7 时，只有 SOC0/SOC3/SOC6/SOC9/SOC12 有这样的控制，用于控制并行采样模式。

表 12-2：采样模式描述

ADCSOCCTL[x].SHEN[2:0] (x=0,3,6,9,12)	ADC 采样模式	ADCSOC[y]CTL.SHEN[1:0] (y=1,2,4,5,7,8,10,11,13,14,15)	ADC 采样模式
7	采样保持器 A，B，C 同时采样	/	/
6	采样保持器 A，C 同时采样	/	/
5	采样保持器 B，C 同时采样	/	/
4	采样保持器 A，B 同时采样	/	/
3	采样保持器 C 单独采样	3	采样保持器 C 单独采样
2	采样保持器 B 单独采样	2	采样保持器 B 单独采样
1	采样保持器 A 单独采样	1	采样保持器 A 单独采样
0	禁用	0	禁用

尽管 3 个采样器需要 3 对通道选择 CHSELxP 和 CHSELxN（x = A，B，C），但每个 ADCSOCCTL[x]寄存器中只有一对 CHSELP 和 CHSELN。寄存器 ADCSOCCTL[x]的 SHEN 位段表示了选择哪个采样保持器。也就是说，CHSELP 和 CHSELN 与 SHEN 相关联。

在顺序采样模式下，位段 SHEN 和位段 CHSELP 与 CHSELN 的对应关系如表 12-3 所示

表 12-3：顺序采样模式下 SOC 的通道选择

SHEN	CHSELAP	CHSELAN	CHSELBP	CHSELBN	CHSELCP	CHSELCN
3	禁用	禁用	禁用	禁用	CHSELP[2:0]	CHSELN[2:0]
2	禁用	禁用	CHSELP[2:0]	CHSELN[2:0]	禁用	禁用
1	CHSELP[2:0]	CHSELN[2:0]	禁用	禁用	禁用	禁用
0	禁用	禁用	禁用	禁用	禁用	禁用

每个采样保持器的通道选择如表 12-4 所示。

表 12-4：ADC 通道选择

Option	CHSELAP	CHSELAN	CHSELBP	CHSELBN	CHSELCP	CHSELCN
7	ADC14	ADC15	ADC16	ADC17	ADC18	ADC19
6	ADC8	ADC9	ADC10	ADC11	ADC12	ADC13
5	ADC2	ADC3	ADC4	ADC5	ADC6	ADC7
4	ADC0	ADC1	ADC0	ADC1	ADC0	ADC1
3	VDD33	VDD12	VBUF0	VBUF1	TSEN1	TSEN0
2	PGA3P	PGA3N	PGA4P	PGA4N	PGA5P	PGA5N
1	PGA0P	PGA0N	PGA1P	PGA1N	PGA2P	PGA2N
0	GND	GND	GND	GND	GND	GND

并行采样模式，位段 CHSELP 和 CHSELN 既取决于 SHEN 还取决于 SOC。例如，我们用 SOC0/1/2 作为一组，并行采样，他们相关的通道设置如表 12-5 所示。相应地，采样保持器 A 的设置在 ADCSOCCTL[0]（SOC0）中；采样保持器 B 的设置在 ADCSOCCTL[1]（SOC1）中；采样保持器 C 的设置在 ADCSOCCTL[2]（SOC2）中。

表 12-5：用 SOC0/SOC1/SOC2 三路并行采样

SHEN	CHSELAP	CHSELAN	CHSELBP	CHSELBN	CHSELCP	CHSELCN
7	ADCSOCCTL[0].CHSELP[2:0]	ADCSOCCTL[0].CHSELN[2:0]	ADCSOCCTL[1].CHSELP[2:0]	ADCSOCCTL[1].CHSELN[2:0]	ADCSOCCTL[2].CHSELP[2:0]	ADCSOCCTL[2].CHSELN[2:0]

我们也可以只用两路并行采样。例如 SOC4/SOC5 并行采样，他们对应的采样保持器为 B/C，设置如表 12-6 所示。这种情况下，采样保持器 A 将不会用到。但是两路并行采样的设置仍然需要在 ADCSOCCTL[3]的 SHEN 位段设置。采样器 B 的通道选择在 ADCSOCCTL[4]中。采样器 C 的通道选择在 ADCSOCCTL[5]中。

表 12-6：用 SOC4/SOC5 两路并行采样

SHEN	CHSELAP	CHSELAN	CHSELBP	CHSELBN	CHSELCP	CHSELCN
6	Disable	Disable	ADCSOCCTL[4].CHSELP[2:0]	ADCSOCCTL[4].CHSELN[2:0]	ADCSOCCTL[5].CHSELP[2:0]	ADCSOCCTL[5].CHSELN[2:0]

更多关于并行采样的描述见章节 12.5。

示例 12.3.4 ADC2，ADC4，ADC6 三路并行采样

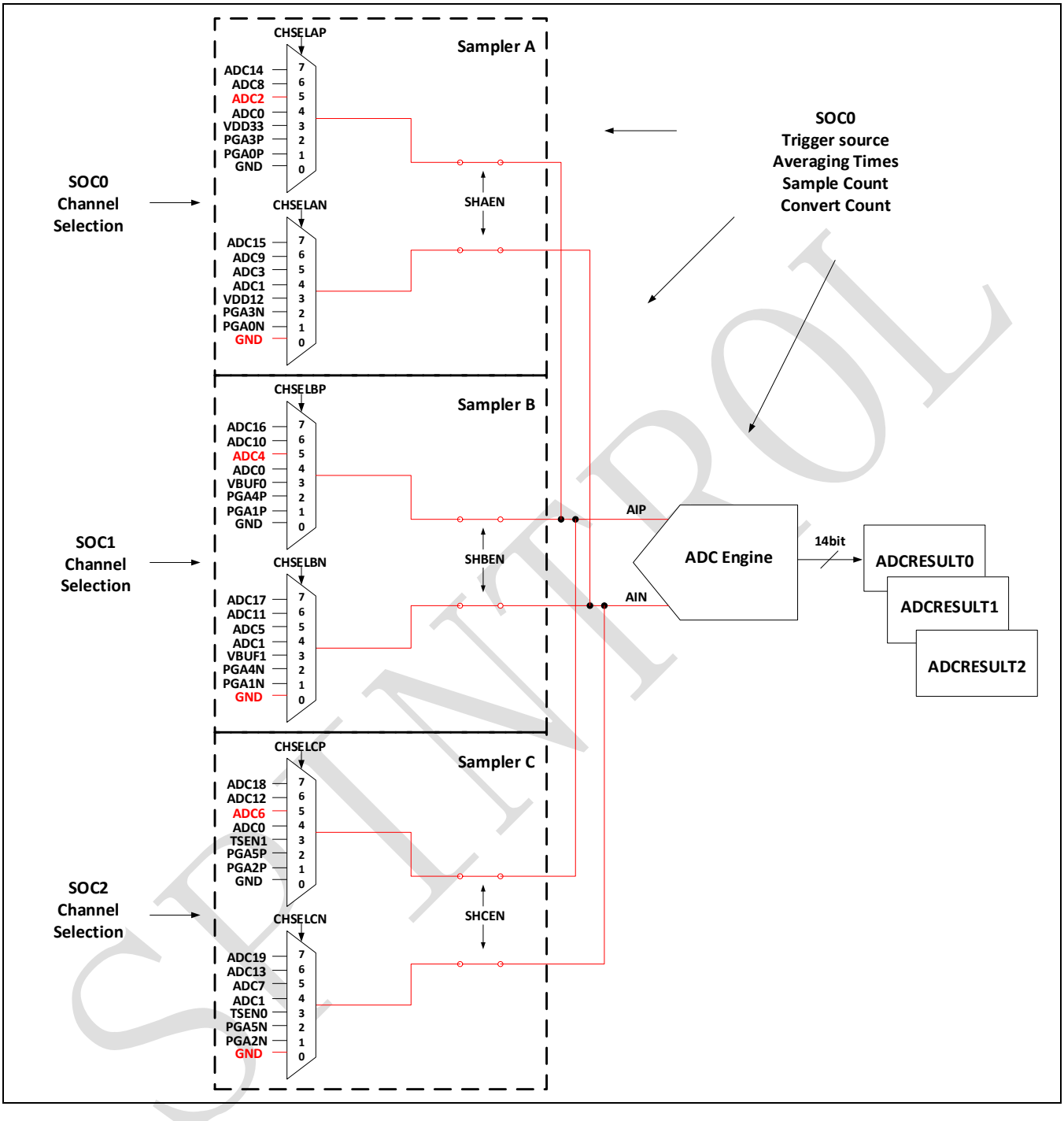
他们可以被分解程如下的代码：

示例 12.3.4

```
void  ADC_Example12_3_4(void)
{
  ADC->ADCSOCCTL[0].bit.SHEN     = 7;/* Enable Sampler A/B/C simultaneous     */
  ADC->ADCSOCCTL[0].bit.TRIGSEL = 8;/* Select PWM1SOCA as the trigger source  */
  ADC->ADCSOCCTL[0].bit.CHSELP = 5;/* Sampler A Positive = ADC2               */
  ADC->ADCSOCCTL[0].bit.CHSELN = 0;/* Sampler A Negative = GND                */
  ADC->ADCSOCCTL[0].bit.AVGCNT = 2;/* Averaging 4 times (2^2)                */
  ADC->ADCSOCCTL[0].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock       */
  ADC->ADCSOCCTL[0].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock       */

  ADC->ADCSOCCTL[1].bit.SHEN     = 1;/* Will be ignore                       */
  ADC->ADCSOCCTL[1].bit.TRIGSEL = 8;/* Will be ignore  */
  ADC->ADCSOCCTL[1].bit.CHSELP = 5;/* Sampler B Positive = ADC4               */
  ADC->ADCSOCCTL[1].bit.CHSELN = 0;/* Sampler B Negative = GND                */
  ADC->ADCSOCCTL[1].bit.AVGCNT = 2;/* Will be ignore                 */
  ADC->ADCSOCCTL[1].bit.SAMPCNT = 3;/* Will be ignore     */
  ADC->ADCSOCCTL[1].bit.CONVCNT = 4;/* Will be ignore      */

  ADC->ADCSOCCTL[2].bit.SHEN     = 1;/* Will be ignore                       */
  ADC->ADCSOCCTL[2].bit.TRIGSEL = 8;/* Will be ignore  */
  ADC->ADCSOCCTL[2].bit.CHSELP = 5;/* Sampler C Positive = ADC6               */
  ADC->ADCSOCCTL[2].bit.CHSELN = 0;/* Sampler C Negative = GND                */
  ADC->ADCSOCCTL[2].bit.AVGCNT = 2;/* Will be ignore                 */
  ADC->ADCSOCCTL[2].bit.SAMPCNT = 3;/* Will be ignore      */
  ADC->ADCSOCCTL[2].bit.CONVCNT = 4;/* Will be ignore       */
}
```

在 SOC0 配置三路并行采样，所以 S0C0/SOC1/SOC2 的通道选择分别对应采样保持器 A/B/C。触发源，平均次数，采样时间和转换时间都在 SOC0 中配置。最后转换结果分别存储在 ADCRESULT[0]/[1]/[2]中，他们与采样保持器 SOC0/1/2 相对应。

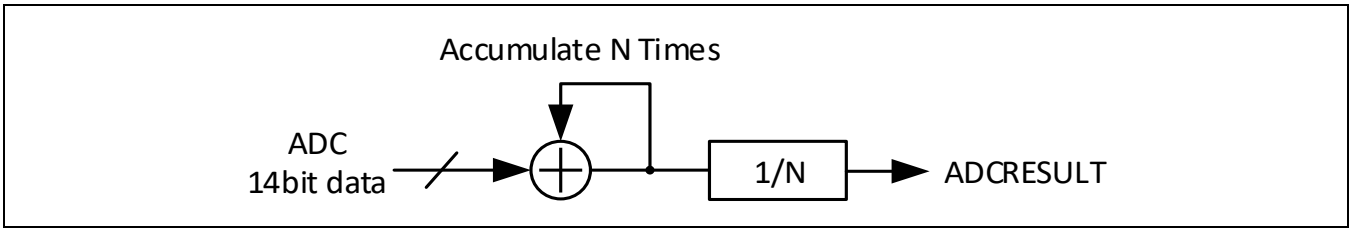
图 12-4：ADC2，ADC4 和 ADC6 三路并行采样



12.3.3 平均化处理

众所周知，采样系统中存在热噪声。这些热噪声来自输入信号、电容采样噪声、比较器噪声、基准电压噪声和电源噪声，等等。如果我们使用软件进行数据后处理，将会等待较长时间，且消耗 CPU 资源。SOC 中的平均化处理在硬件上完成了这一目的。这样，ADC 就可以连续采样多次，并在硬件上做平均处理，将平均后的结果存入 ADCRESULT[x]寄存器。

图 12-5：ADC 转换结果的平均化处理



每个 SOC 的控制寄存器 ADCSOCCTL[x]中的位段 AVGCNT 可以控制平均化次数。

表 12-7：平均化次数选择

AVGCNT[2:0]	平均化次数
7	128
6	64
5	32
4	16
3	8
2	4
1	2
0	1

12.3.4 ADC 采样和转换窗口

由于外部驱动器的内阻不同，为了将 ADC 的采样电容充电到足够的精度，需要不同的建立时间。 因此，对于每个 SOC 都有独立的采样窗口大小的配置。寄存器 ADCSOCCTL[x]中 SAMPCNT 有 8 位控制采样时间，CONVCNT 有 7 位控制转换时间。

$$Sample\_Time = (\text{Sampcnt} + 1) \times \text{ADC\_Clk\_Cycle}$$
$$Conversion\_Time = (\text{Convcnt} + 1) \times \text{ADC\_Clk\_Cycle}$$

采样时间建议至少 125 纳秒。转换时间必须大于 140 纳秒。这样总的信号处理时间最短为 265 纳秒（采样时间加转换时间之和）。不同 ADC 时钟下的采样窗口大小如表 12-8 所示。。

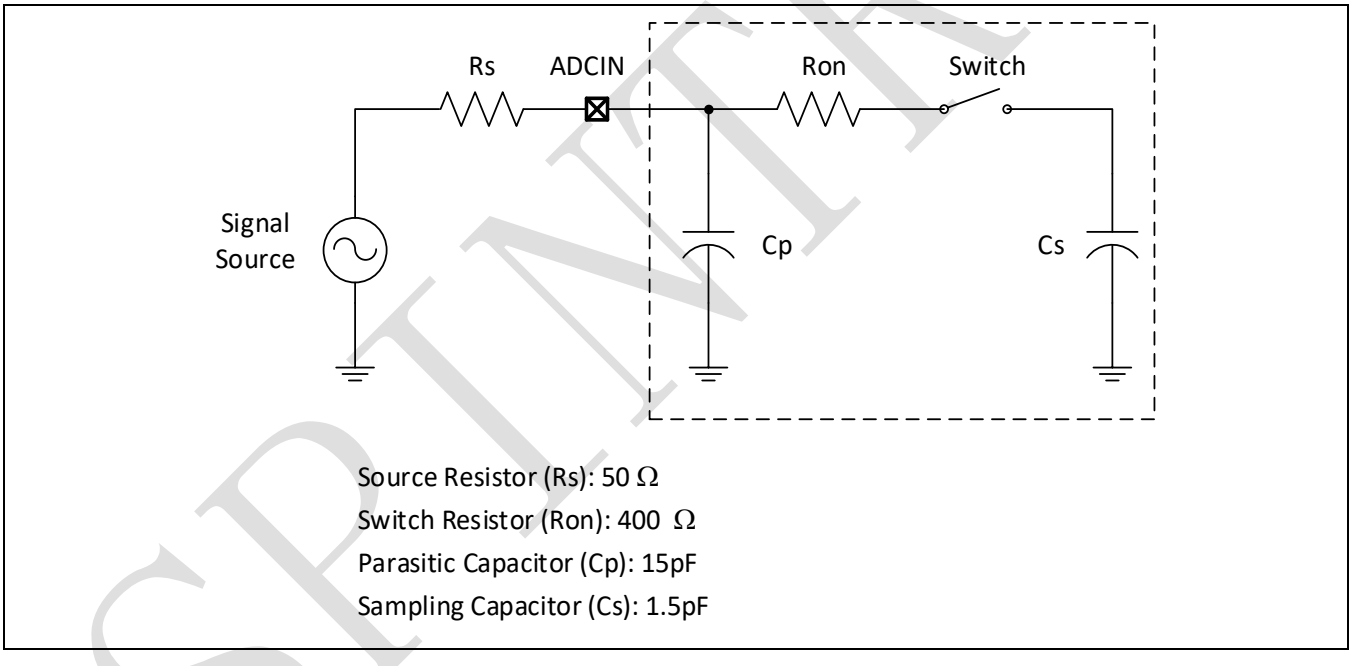
对于转换时间，不同 ADC 时钟周期和 CONVCNT 必须保证其大于 140 纳秒。例如，假设 ADC 时钟 32MHz（周期为 31.25 纳秒），那么 CONVCNT 必须大于等于 4，以确保大于 140 纳秒的转换时间。

表 12-8：不同 SAMPCNT 和 CONVCNT 对应的 ADC 信号处理时间

ADC 频率	SAMPCNT	采样时间（ns）	CONVCNT	转换时间（ns）	总的信号处理时间（ns）	采样率
32 MHz	3	125	4	156.25	281.25	3.56MHz
32 MHz	27	875	4	156.25	1031.15	970kHz
40 MHz	4	125	5	150	275	3.64MHz
40 MHz	34	875	5	150	1025	976kHz
64 MHz	7	125	8	140.625	265.625	3.76MHz
64 MHz	55	875	8	140.625	1015.625	985kHz

采样时间主要取决于信号源阻抗，通道上的寄生电阻和寄生电容，采样电容等等。如图 12-6 所示，ADC 输入通路可以表征为一个 RC 网络。假设 ADCIN 上面的信号 0 变化到 3.3V，那么采样电容上的信号建立到 12 位精度需要 70 纳秒时间。

图 12-6：ADCx 输入通路 RC 网络



12.4 ADC 转换优先级

当多个 SOC 同一时间触发，需要优先级来决定他们的执行。ADC 中有两套机制。默认优先级机制就是 Round-Robin 环。在这个机制中，所有 SOC 本质上优先级相同，他们的执行先后顺序取决于 Round-Robin 指针（RRPOINTER）。RRPOINTER 会指向寄存器 ADCSOCPRICTL 中最近执行的一个 SOC。因此，下一个执行的就是大于当前 RRPOINTER 数值的 SOC。这样一直顺序执行到 SOC15。SOC15 之后执行的将是 SOC0。默认情况下，RRPOINTER 等于 0。RRPOINTER 的值会被芯片复位，在 ADCCTL0.RST 有效或者 ADCSOCPRICTL 被改写后。

一个 Round-Robin 环优先级的例子被展示在图 12-7 中。

第一步，在复位后，SOC0 是优先级最高的 SOC。

第二步，SOC7 被触发，因此 SOC7 被立即执行转换。

第三步，指针 RRPOINTER 指向 SOC7。SOC8 因此变成了优先级最高的 SOC。此时如果 SOC2 和 SOC12 同时触发，那么 SOC12 将先被转换，此时 SOC2 将等候执行。

第四步，指针 RRPOINTER 指向 SOC12。之后，SOC2 被执行转换。

第五步，指针 RRPOINTER 指向 SOC2。SOC3 现在变为优先级最高的 SOC。

寄存器 ADCSOCPRICTL 中，PRIORITY 位段可以被用于赋予一部分 SOC 更高的优先级。当这些 SOC 被 PRIORITY 设置后，那么在当前 SOC 结束之后，ADC 会打断 Round-Robin 环的顺序，并立即执行这些高优先级 SOC。在这些 SOC 执行结束之后，Round-Robin 环从它被打断的地方继续执行，仍然按照之前的顺序方式，数值低的 SOC 先执行。

高优先级模式，必须从 SOC0 开始，然后可以增加需要优先的 SOC 数量。PRIORITY 的值表示了有几个 SOC 是高优先级 SOC。例如如果 PRIORITY 的值等于 4，那么 SOC0、SOC1、SOC2 和 SOC3 就是高优先级 SOC。并且，在这之中，SOC0 的优先级最高。

一个应用高优先级的例子被展示在图 12-8 中。

假设位段 PRIORITY 的值等于 4。

第一步，复位后，SOC4 是 Round-Robin 环中最先执行的 SOC。在这之后 SOC7 被触发，因此 SOC7 被立即执行转换。

第二步，指针 RRPOINTER 指向 SOC7。SOC8 因此变成了优先级最高的 SOC。

第三步，SOC2 和 SOC12 同时触发，SOC 会打断 Round-Robin 环，并立即执行。此时 SOC12 将等候执行。

第四步，指针 RRPOINTER 仍然停留在 SOC7，接下来 SOC12 将会被执行转换。

第五步，指针 RRPOINTER 指向 SOC12。SOC13 变成了 Round-Robin 环中最先执行的 SOC。

图 12-7：Round-Robin 环优先级示例

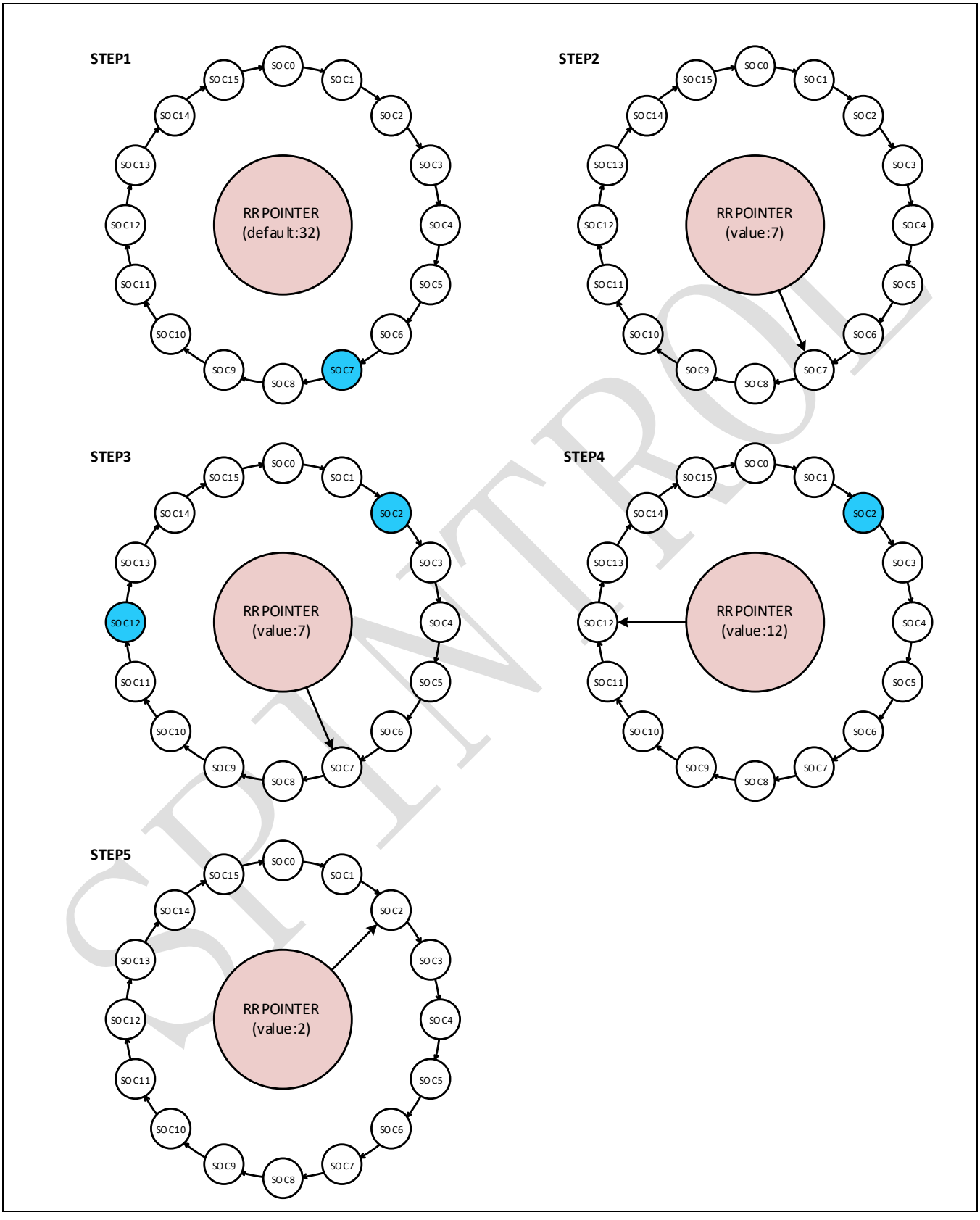
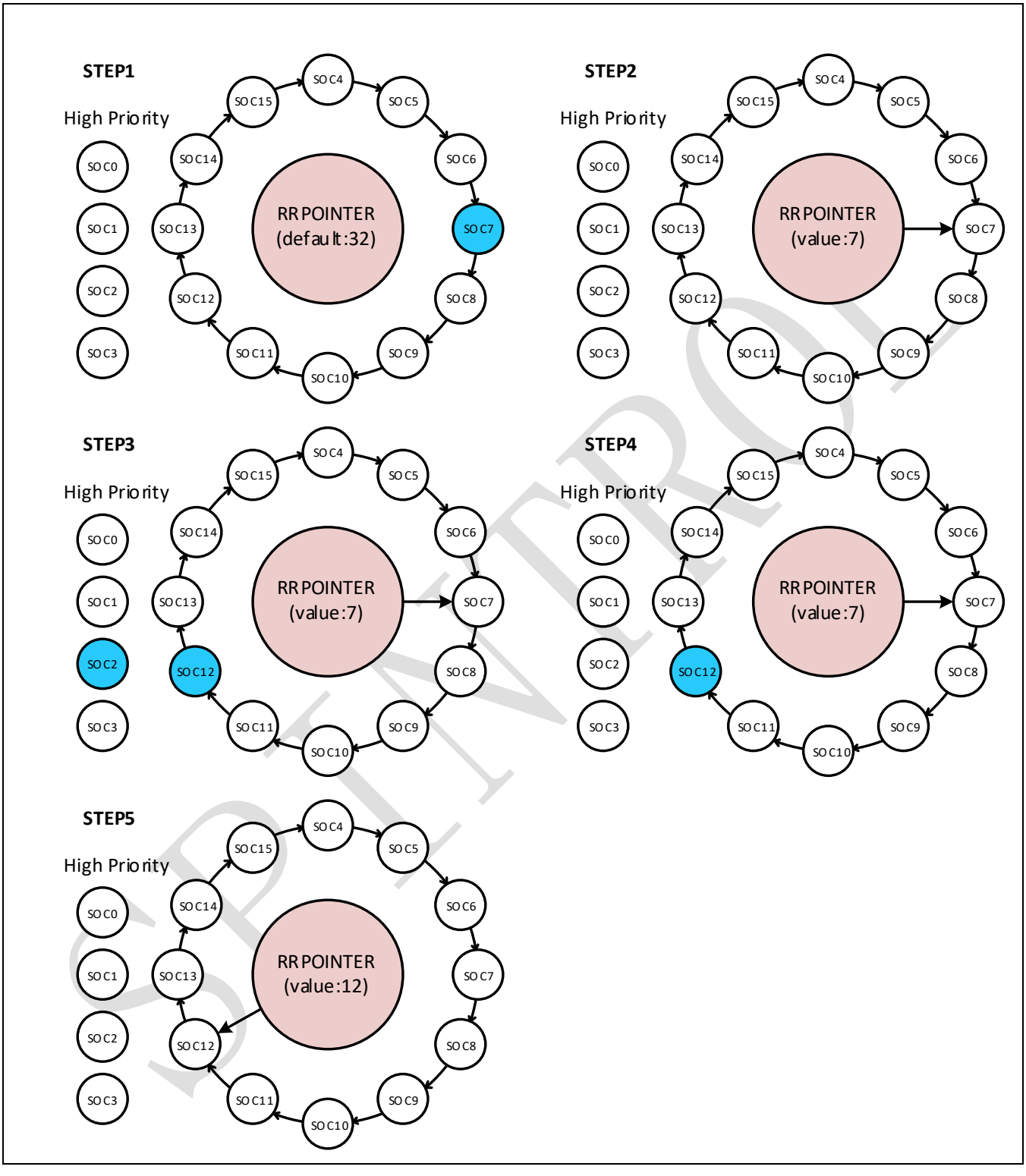


图 12-8：高优先级示例



12.5 同步采样模式

在一些应用中，希望不同信号之间的采样延迟最小化。SPC2168 的 ADC 包含三个采样保持器，因此它支持两路或者三路信号同时采样。同时采样模式可以在寄存器 ADCSOCCTL[x]的 SHEN 位段配置。

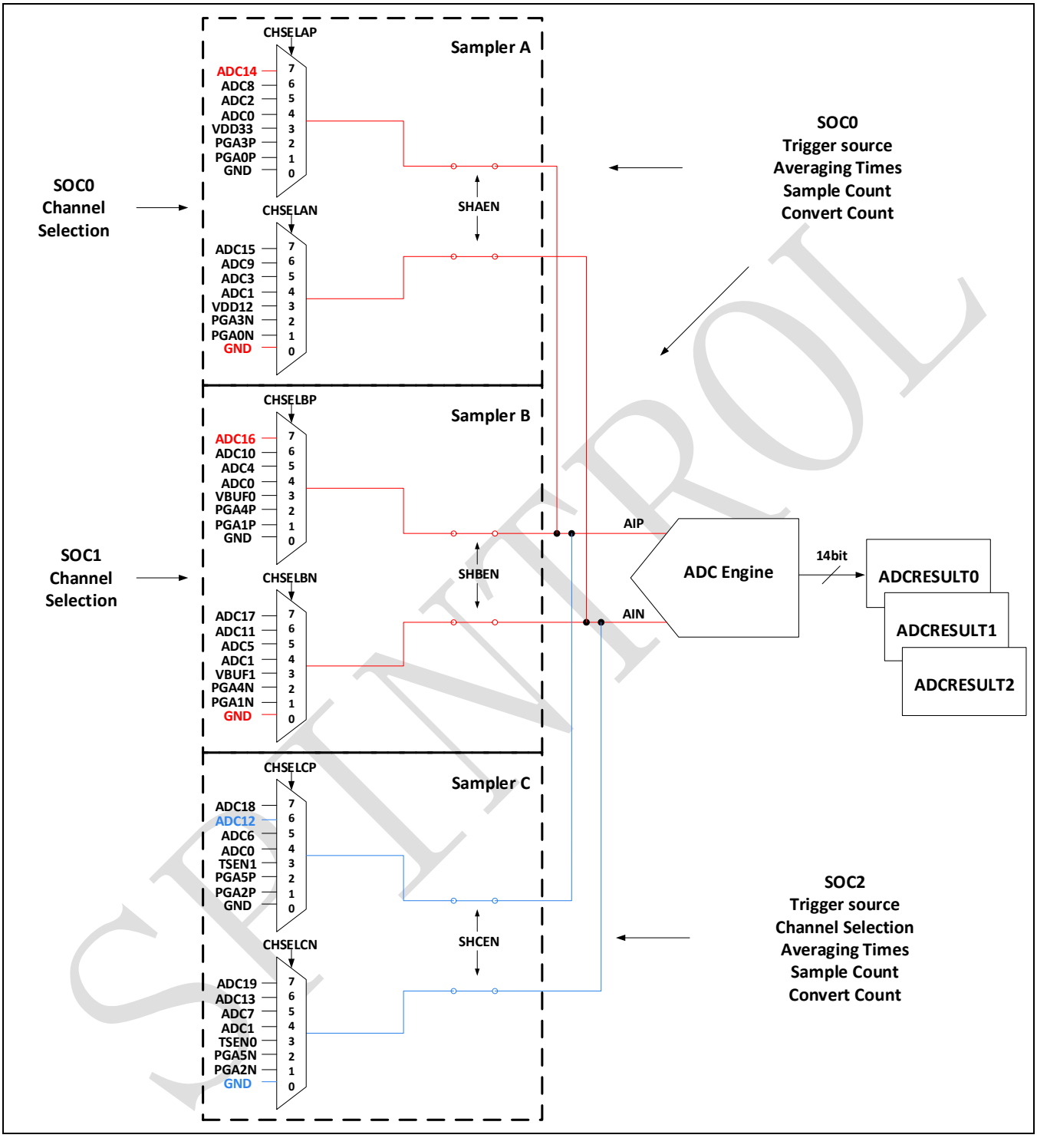
它的工作原理如下：

- ADC 将 16 个 SOC 分为五组。其中 SOC0~2 一组，SOC3~SOC5 一组，SCO6~SOC8 一组，SOC9~SOC11 一组，SOC12~SOC14 一组，共五组。
- 同时采样模式下，采样模式的配置、触发源、平均次数、采样时间和转换时间都在每组的第一个 SOC 设置（SOC0，SOC3，SOC6，SOC9，SOC12）。
- 对于采样模式下的通道选择，每组的 SOC[x]/SOC[x+1]/SOC[x+2]（x=0, 3, 6, 9, 12）分别对应采样保持器 A/B/C。
- 同时采样模式下，在采样结束以后，转换器会先转换采样保持器 A 的信号，然后转换采样保持器 B，最后转换采样保持器 C。
- 每组的 SOC[x]/SOC[x+1]/SOC[x+2]的转换结果被分别存储在对应的结果寄存器中 ADCRESULT[x]/ADCRESULT[x+1]/ADCRESULT[x+2]（x=0, 3, 6, 9, 12）。
- 当 SHEN 等于 4，此时只有采样保持器 A 和采样保持器 B 同时采样，采样保持器 C 可以独立的使用。同时采样的结果会存储在 ADCRESULT[x]/ADCRESULT[x+1]（x=0, 3, 6, 9, 12）中。
- 当 SHEN 等于 5，此时只有采样保持器 B 和采样保持器 C 同时采样，采样保持器 A 禁止使用。同时采样的结果会存储在 ADCRESULT[x+1]/ADCRESULT[x+2]（x=0, 3, 6, 9, 12）中。
- 当 SHEN 等于 6，此时只有采样保持器 A 和采样保持器 C 同时采样，采样保持器 B 可以独立的使用。同时采样的结果会存储在 ADCRESULT[x]/ADCRESULT[x+2]（x=0, 3, 6, 9, 12）中。
- 当 SHEN 等于 7，此时三个采样保持器 A/B/C 同时采样。同时采样的结果会存储在 ADCRESULT[x]/ADCRESULT[x+1]/ADCRESULT[x+2]（x=0, 3, 6, 9, 12）中。

表 12-9：同时采样模式控制

ADCSOCCTL[x].SHEN[2:0] (x=0,3,6,9,12)	SOCx	SOCx+1	SOCx+2	采样模式描述
7	采样保持器 A	采样保持器 B	采样保持器 C	采样保持器 A，B，C 同时采样
6	采样保持器 A	/	采样保持器 C	采样保持器 A，C 同时采样
5	/	采样保持器 B	采样保持器 C	采样保持器 B，C 同时采样
4	采样保持器 A	采样保持器 B	/	采样保持器 A，B 同时采样

图 12-9：SOC0 同时采样的信号流和配置说明



示例 12.5.1　PWM0SOCA 触发 SOC0 同时转换 ADC14 和 ADC16

示例 12.5.1

```
void  ADC_Example12_5_1(void)
{
  ADC->ADCSOCCTL[0].bit.SHEN     = 4;/* Enable Sampler A and B simultaneous     */
  ADC->ADCSOCCTL[0].bit.TRIGSEL = 5;/* Select PWM0SOCA as the trigger source  */
  ADC->ADCSOCCTL[0].bit.CHSELP = 7;/* Sampler A Positive = ADC14               */
  ADC->ADCSOCCTL[0].bit.CHSELN = 0;/* Sampler A Negative = GND                */
  ADC->ADCSOCCTL[0].bit.AVGCNT = 2;/* Averaging 4 times (2^2)                 */
  ADC->ADCSOCCTL[0].bit.SAMPCNT = 3;/* Sampling time = (3+1)*ADC_Clock      */
  ADC->ADCSOCCTL[0].bit.CONVCNT = 4;/* Conversion time = (4+1)*ADC_Clock       */

  ADC->ADCSOCCTL[1].bit.SHEN     = 1;/* Will be ignore                       */
  ADC->ADCSOCCTL[1].bit.TRIGSEL = 6;/* Will be ignore  */
  ADC->ADCSOCCTL[1].bit.CHSELP = 7;/* Sampler B Positive = ADC16              */
  ADC->ADCSOCCTL[1].bit.CHSELN = 0;/* Sampler B Negative = GND                */
  ADC->ADCSOCCTL[1].bit.AVGCNT = 2;/* Will be ignore                 */
  ADC->ADCSOCCTL[1].bit.SAMPCNT = 3;/* Will be ignore     */
  ADC->ADCSOCCTL[1].bit.CONVCNT = 4;/* Will be ignore      */

  ADC->ADCSOCCTL[2].bit.SHEN     = 3;/* Enable Sampler C single sampling       */
  ADC->ADCSOCCTL[2].bit.TRIGSEL = 9;/* Select PWM1SOCB as the trigger source  */
  ADC->ADCSOCCTL[2].bit.CHSELP = 6;/* Sampler C Positive = ADC12               */
  ADC->ADCSOCCTL[2].bit.CHSELN = 0;/* Sampler C Negative = GND                */
  ADC->ADCSOCCTL[2].bit.AVGCNT = 3;/* Averaging 8 times (2^3)                */
  ADC->ADCSOCCTL[2].bit.SAMPCNT = 5;/* Sampling time = (5+1)*ADC_Clock      */
  ADC->ADCSOCCTL[2].bit.CONVCNT = 5;/* Conversion time = (5+1)*ADC_Clock       */
}
```

如图 12-9 所示，当 PWM0SOCA 发出一个触发信号，ADC14 和 ADC16 会被同时采样。随后，ADC14 的采样信号会被转换并存储在 ADCRESULT[0]中。在这之后，ADC16 的采样信号会被转换并存储在 ADCRESULT[1]中。ADC16 转换结束后，会发转换结束的标志 EOC0，代表同时采样结束。除此以外，当 PWM1SOCB 发出一个触发信号，ADC12 会被采样，这套配置存储在 SOC2，结果将会存在 ADCRESULT[2]中。

同时采样模式下，优先级规则和顺序采样模式相同。章节 12.8 展示了同时采样模式的时序。

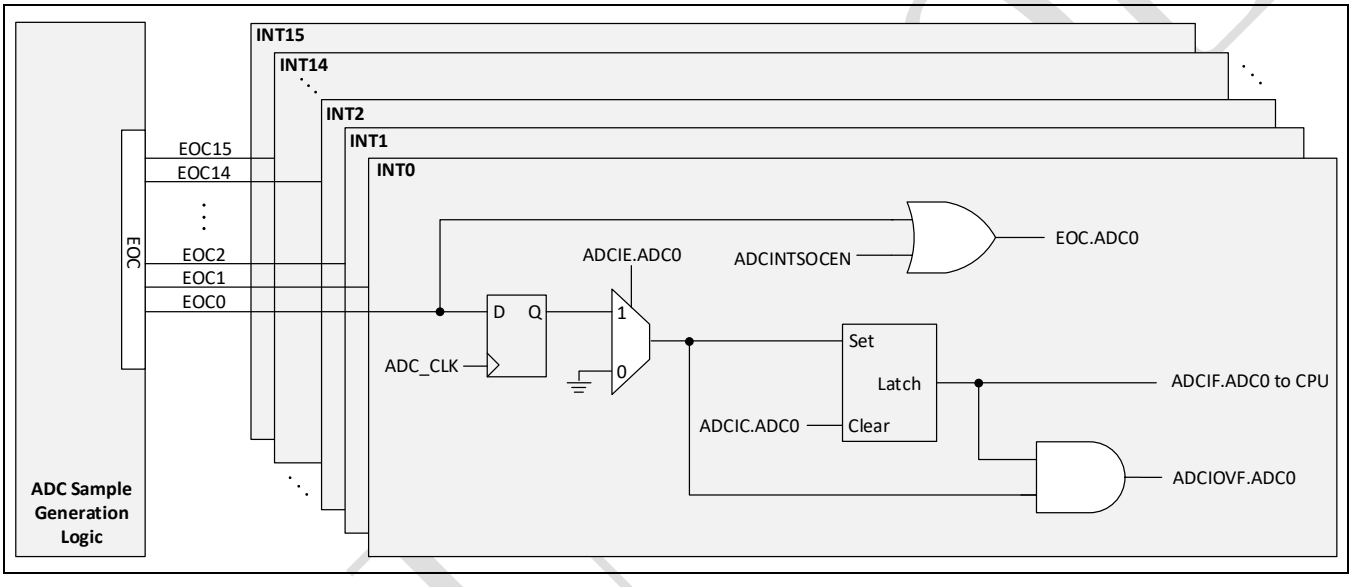
注意：　在同时采样模式下，只有不同采样保持器的信号才能达到同时采样的目的。如果两个信号属于同一采样保持器，则不能实现同时采样。例如 ADC2 可以和 ADC4 同时采样，ADC2 和 ADC14 则不行，因为他们属于被连接到了同一个采样保持器。

12.6 转换结束和产生中断原理

因为 ADC 有 16 套 SOC 配置，所以也有 16 个转换结束标志（EOC，End-of-Conversion）。顺序模式下，EOCx 与每个 SOCx 对应。同时采样模式，EOCx 与每组 SOCx（x=0, 3, 6, 9, 12）的第一个 SOC 对应。对于 SOC 的分组在章节 12.5 中有详细描述。EOCx 会在每次转换结束时发生。顺序模式下 EOCx 产生的时序图参见图 12-12。每个 EOCx 信号都可以被配置成 SOC 的触发源。并且，这个 EOCx 信号可以产生 ADC 中断。因此 16 个 EOCx 可以产生 16 个 ADC 中断。这些中断可以被送到 CPU。

图 12-10 展示了 ADC 中断结构的框图。

图 12-10：ADC 中断结构框图

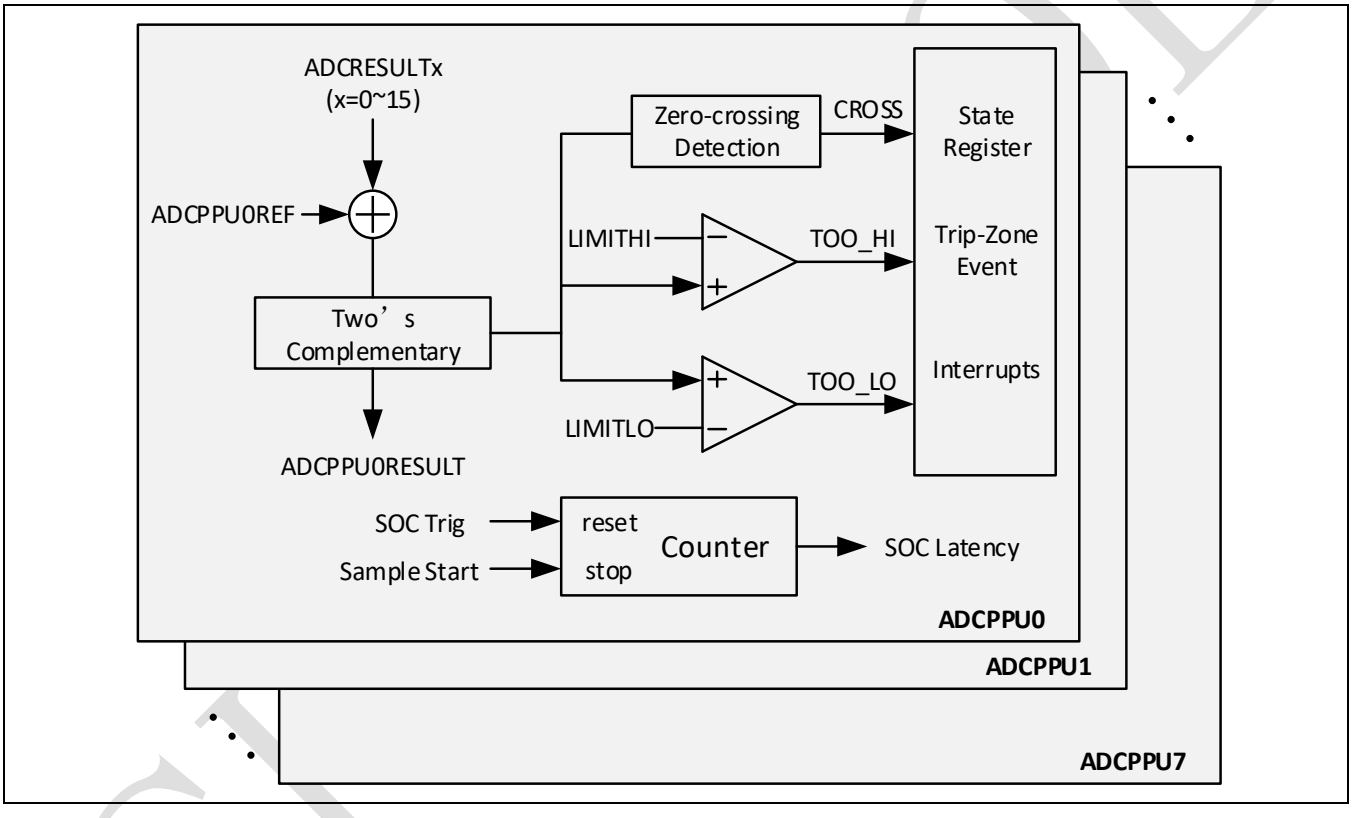


12.7 ADC 后处理单元

ADC 共有 8 个数字后处理单元（Digital Post-Processing Unit，PPU）。每一个后处理单元可以处理任意一个 SOC 的结果，只要通过寄存器 ADCPPUCTLx（x=0~7）的 DATASEL 位段选择即可。每个后处理单元都可以完成若干功能，例如减去一个基准值，判别过零点以及过高或者过低检测等等。不仅如此，这些过零和比较的检测还可以产生事件触发 PWM 封锁（trip-zone）或者中断信号。

后处理单元也可以记录 SOC 从被触发到实际开始执行的延迟。图 12-11 展示了每个后处理单元的架构。下面会分别介绍各个子功能块。

图 12-11：后处理单元构架



12.7.1 PPU 基准误差计算

在许多应用中，ADC 的结果需要减去一个基准值。数字后处理单元可以在硬件上自动实现该功能，从而减少输出延迟以及软件负担。

首先，配置寄存器 ADCPPUCTLx 的 DATASEL 位段，这样决定后处理单元对哪个 SOC 的结果做处理。这意味着 PPU 必须在 SOC 的结果出来以后才能运行。然后写一个基准值到寄存器 ADCPPUREFx（x=0~7）。数字后处理会自动地将 SOC 的结果减去该基准值并存储在 ADCPPURESULTx（x=0~7）中。这个减法会产生一个符号位扩展的 32 位数。我们只要设置寄存器 ADCPPUCTLx 的 POL 位段，就可以将这个值反向之后再存入 ADCPPURESULTx。

示例 12.7.1 用 PPU0 让 SOC0 的结果减去一个基准值

它可以被分解成以下代码：

示例 12.7.1

```
void  ADC_Example12_7_1(void)
{
  ADC->ADCPPU0CTL.bit.EN     = 1;/* Enable ADC PPU0     */
  ADC->ADCPPU0CTL.bit.DATASEL = 0;/* Select SOC0 result as post-processing data */
  ADC->ADCPPU0REF.all = 100;/* Set the reference value of PPU0 is 100  */

  ADC->ADCIC.bit.INT0 = 1;/* Clear SOC0 interrupt flag */
  ADC->ADCSOCFRC.bit.SOC0 = 1;/*Trig SOC0 to convert and PPU0 to post-processing */
  while(ADC->ADCIF.bit.INT0 != 1){}; /* wait for SOC0 result ready*/
}
```

首先使能 PPU0，然后让 PPU0 选择 SOC0 做处理。之后设定需要减去的基准值。最后当 SOC0 转换完成后，PPU0 就会处理 SOC0 的结果，减去设定基准值并存入 ADCPPURESULT[0]中。

12.7.2 PPU 过零和阈值检测

许多应用中，需要对于 ADC 的结果运行阈值检测。后处理单元可以实现这些功能，例如过零，或者过高过低检测。基于这些检测结果，硬件 PPU 单元可以自动地触发 PWM 封锁或者中断信号，而不需要额外的软件设置，从而达到减轻软件负担的目的。这个功能也可以用于安全方面，当 ADC 的转换结果超出范围，就会直接触发 PWM 封锁，而不需要通过 CPU 干预。

为了开启这个功能，首先应该配置 ADCPPUCTLx 的 DATASEL 位段，指明 PPUx 处理哪个 SOC 的结果。这样，SOC 一旦运行结束，PPU 就会立即执行。接着，给寄存器 ADCPPUTHHx 和 ADCPPUTHLx 赋值（过零检测则不需要进一步配置）。

当 SOC 转换结束之后（EOC 被触发），只要 ADCPPURESULTx 改变符号，寄存器 ADCPPUTZEx 的 XZRO 位段就会被置 1。或者对应的阈值被超过，寄存器 ADCPPUTZEx 的 TZHI 或 TZLO 位段就会被置 1。

这些检测结果可以产生事件触发中断，这些中断标志信号在 ADCPPUIFx 寄存器中。ADCPPUIEx 寄存器使能这些功能。相应地，寄存器 ADCPPUICx 可以清除中断标志。

示例 12.7.2　用 PPU0 对 SOC0 的结果做过零检测以及过高和过低检测

它可以被分解成以下代码：

示例 12.7.2

```
void ADC_Example12_7_2(void)
{
   ADC->ADCPPU0CTL.bit.EN          = 1;/* Enable ADC PPU0         */
   ADC->ADCPPU0IE.bit.XZRO         = 1;/* Enable zero-crossing interrupt of ADC PPU0 */
   ADC->ADCPPU0IE.bit.TZHI         = 1;/* Enable too high interrupt of ADC PPU0 */
   ADC->ADCPPU0IE.bit.TZLO         = 1;/* Enable too low interrupt of ADC PPU0 */

   ADC->ADCPPU0CTL.bit.DATASEL = 0;/* Select SOC0 result as post-processing data */
   ADC->ADCPPU0THH.all = 3000;/* Set PPU0 high threshold is 3000    */
   ADC->ADCPPU0THL.all = 1000;/* Set PPU0 low threshold is 1000    */

   ADC->ADCIC.bit.INT0 = 1;/* Clear SOC0 interrupt flag */
   ADC->ADCSOCFRC.bit.SOC0 = 1;/*Trig SOC0 to convert and PPU0 to post-processing */
   while(ADC->ADCIF.bit.INT0 != 1){}; /* wait for SOC0 result ready*/
}
```

首先使能 PPU0，然后选择 PPU0 处理 SOC0 的结果。设置高低检测阈值。最后触发 SOC0，那么在 SOC0 转换结束之后，PPU0 就会自动的完成检测，并产生标志信号。

12.7.3 采样延迟捕获

当多个控制环路同时触发多个 SOC，由于 SOC 自身有优先级，因此只有一个 SOC 可以立即执行，其它 SOC 都需要等待。这意味着这些不能立即执行的 SOC，存在一个从它被触发到实际执行采样的延迟。这在系统中叫做采样延迟误差。假使我们已经知道这个延迟的具体值，那么软件就可以利用插值技术，减小这个采样延迟带来的影响。

为了实现这一功能，需要设置 PPUx 寄存器 ADCPPUCTLx 中的 SOCSEL 位段，选择需要记录延迟的 SOC。从 SOC 触发到它被实际执行采样的延迟将以 ADC 时钟计数，并存储在 32 位寄存器 ADCPPUSOCDLYx 中。当 SOC 被触发以后，这个计数器被复位并开始计数。当 SOC 开始执行采样，计数器停止计数，并将当前值写入 ADCPPUSOCDLYx 寄存器中。

示例 12.7.3 用 PPU0 记录 SOC0 的采样延迟

它可以被分解成以下代码：

示例 12.7.3

```
void  ADC_Example12_7_3(void)
{
  ADC->ADCPPU0CTL.bit.EN      = 1;/* Enable ADC PPU0     */
  ADC->ADCPPU0CTL.bit.SOCSEL = 0;/* Select SOC0 to count sample delay */

  ADC->ADCIC.bit.INT0 = 1;     /* Clear SOC0 interrupt flag */
  ADC->ADCSOCFRC.bit.SOC0 = 1; /* Trig SOC0 and reset PPU0 delay capture counter */
  while(ADC->ADCIF.bit.INT0 != 1){}; /* wait for SOC0 result ready*/
}
```

使能 PPU0 的延迟捕获功能，并选择 SOC0 作为捕获对象。然后触发 SOC0，那么 PPU0 就会计算 SOC0 被触发到真正开始采样的延迟。

12.8 ADC 时序

图 12-12：顺序转换模式时序

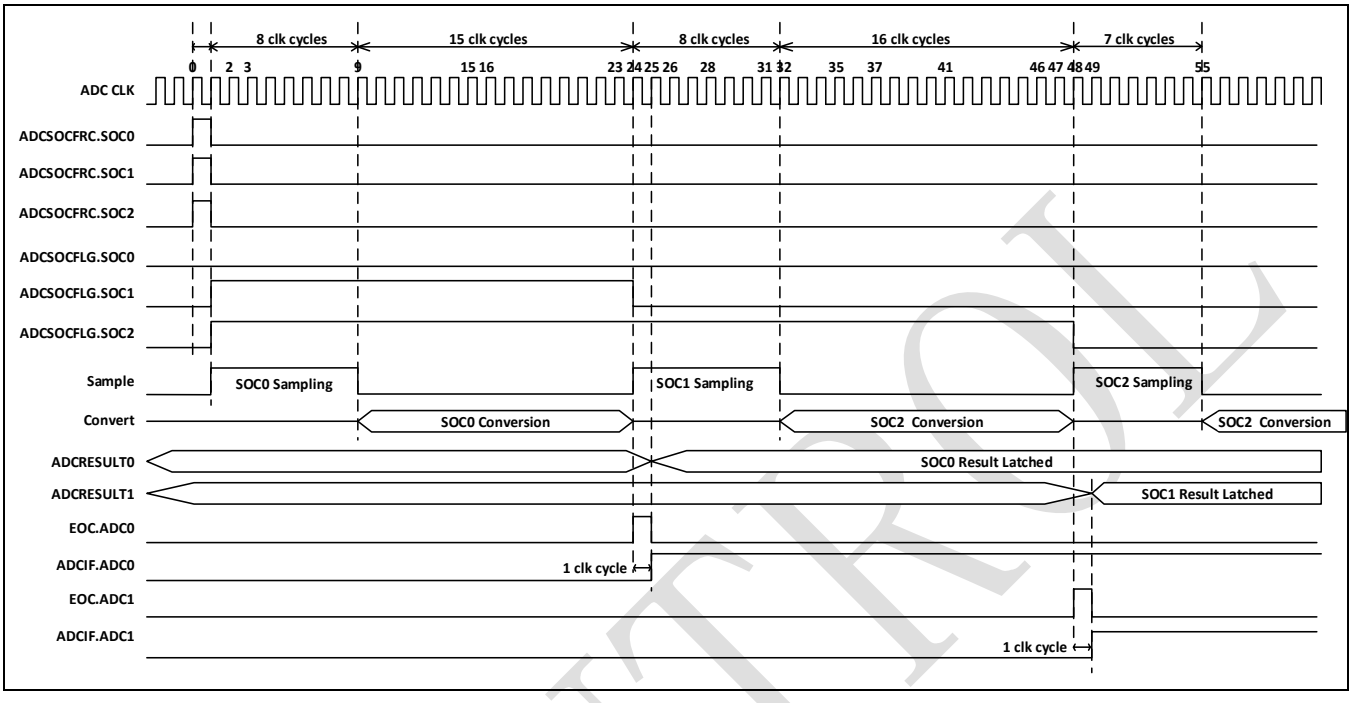
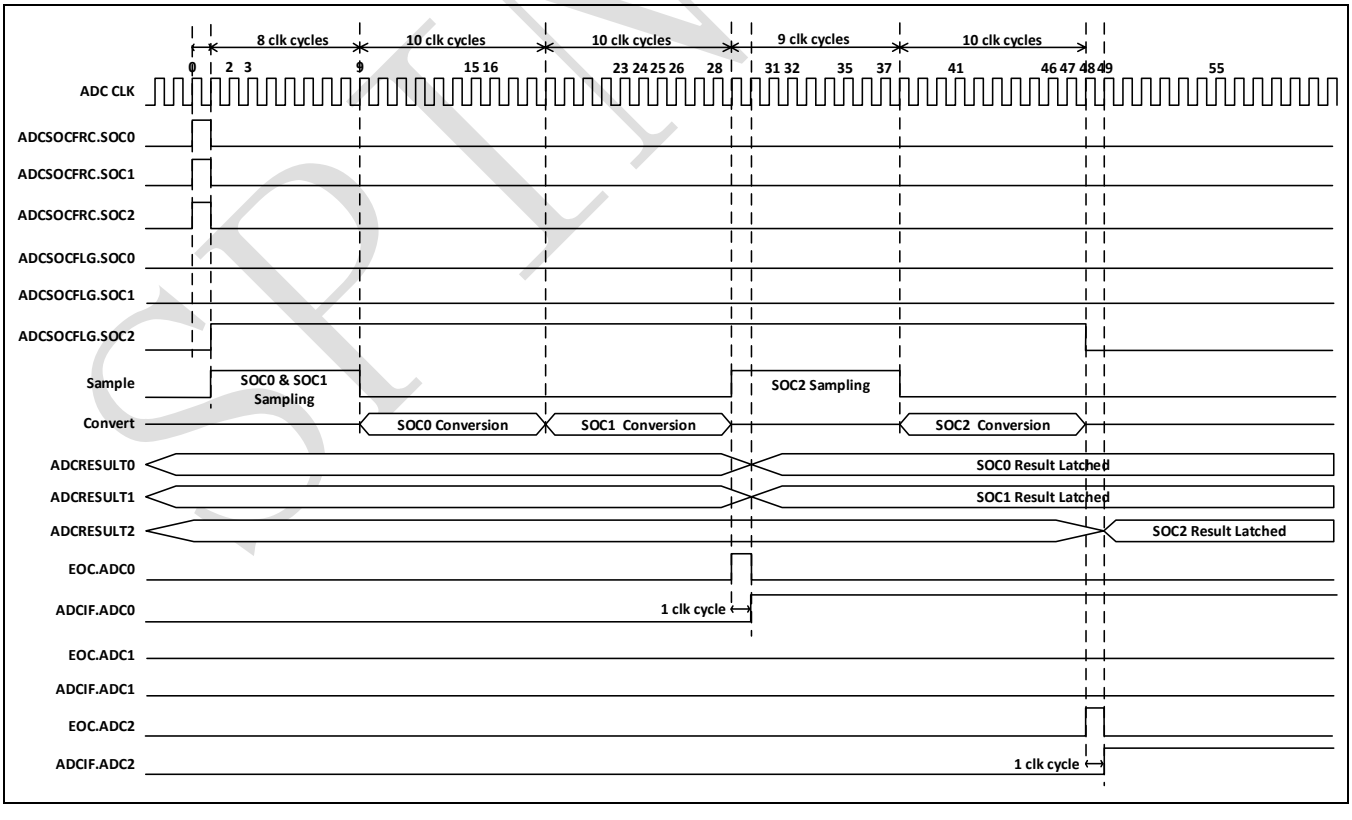


图 12-13：同时采样模式时序



12.9 上电顺序

当 ADC 开始启动，需要遵循下面的上电顺序：

第一步：使能 ADC 的带隙基准模块（bandgap）

第二步：等待 200 微秒以后，使能 ADC 的基准电压驱动模块

第三步：等待 200 微秒以后，使能 ADC 内部模拟电路

示例 12.9.1 ADC 使能上电

它可以被分解成以下代码：

示例 12.9.1

```
void   ADC_Example12_9_1(void)
{
   ADC->ADCBGCTL.bit.EN = 1;   /* Enable on Bandgap                */
   Delay_ms(1);
   ADC->ADCREFCTL.bit.EN = 1; /* Enable Reference Buffer          */
   Delay_ms(1);
   ADC->ADCCTL.bit.EN = 1; /* Enable ADC                */
   ADC->ADCCTL.bit.RST = 1; /* ADC logic state-machine reset, will be self-clear */
}
```

当关闭 ADC 时，步骤一至三的控制位可以同时关闭。ADC 的电源必须通过软件控制，他们和 MCU 系统电源是独立的。

在使用 ADC 时，需要将 ADC 的所有的寄存器复位，这个在软件上可以通过将 ADCCTL 的 RST 位段置 1 来实现。之后，RST 会自动清零。

12.10 ADC 校准

转换器存在固有的失调和增益误差。ADC 在出厂时，会在室温下校准这些误差。同时也允许用户在环境温度变化时，修改校准失调和增益的校准参数以达到更好的精度。一般情况下，除非环境剧烈变化，否则用户不需要修改，直接使用出场校准参数即可。ADC 会用这些参数校准每次转换结果，这个过程已经在硬件上处理。

12.10.1 出厂设置和硬件校准

在出厂时，ADC 的校准结果会被存入 Flash 中。在 SPC2168 启动的时候，内置的启动程序会将 Flash 的校准参数写入相应的 ADC 寄存器。这样在 ADC 正常运行时，这些寄存器的校准参数就可以用于硬件校准。

12.10.2 通道校准

如图 12-1 所示，每个通道有独立的失调（OFFSET0~15）和增益误差（GAIN0~15）。为此，在硬件上分配了独立的失调寄存器 ADCOFFSETx（x=0~15）和增益寄存器 ADCGAINx（x=0~15）。这允许用户根据每个通道的实际情况设定不同的校准参数。

通道校准公式为：

$$ADC\_RESULT[i] = (\text{ADC\_RAW\_CODE[i]} - \text{OFFSET[i]}) \times \text{GAIN[i]}$$

在上面的公式中，参数 i 的范围从 0 到 15。这个校准过程会在硬件上自动完成。

12.11 ADC 结果

12.11.1 满幅度测量范围

ADC 的满幅度测量范围从-FS 到+FS：

$$FS = 3.657\text{V}$$

但是，对于每个通道，其电压不能超过 AVDD。如果 AVDD 为 3.3V，则表示各通道输入电压不能超过 3.3V。

12.11.2 ADC 输出码值计算

如图 12-14 所示，ADC 的输入是差分的。每个端口的输入范围最大可以支持 0~3.657V。因此，量化范围是-3.657V 到 3.657V；数字码值范围是-8192 到 8191，如图 12-15 所示。

它的输出码值可以按照如下公式计算：

$$ADC\_RESULT = \frac{(AIP - AIN)}{3.657} \times 8192$$

小数部分会被截断。

图 12-14：ADC 内核示意图

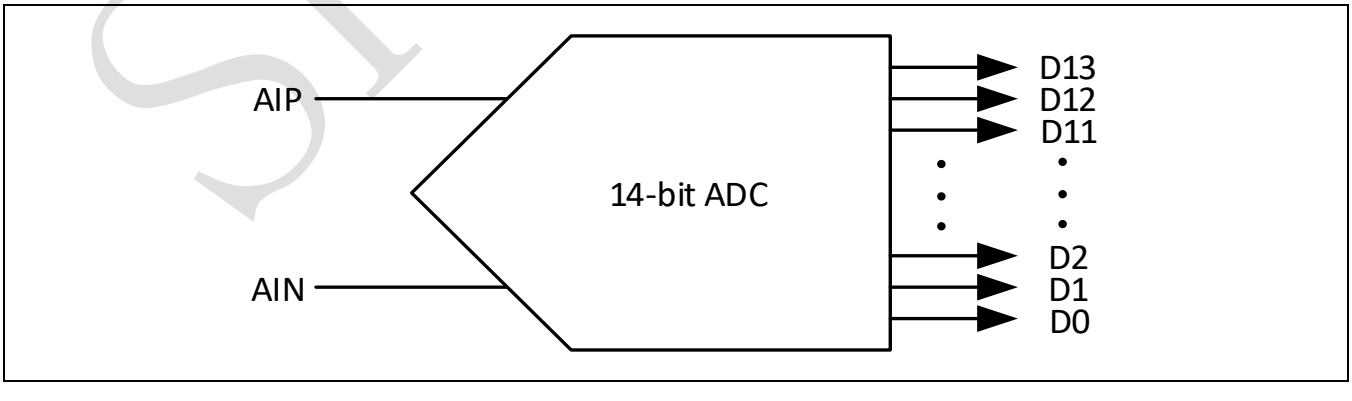
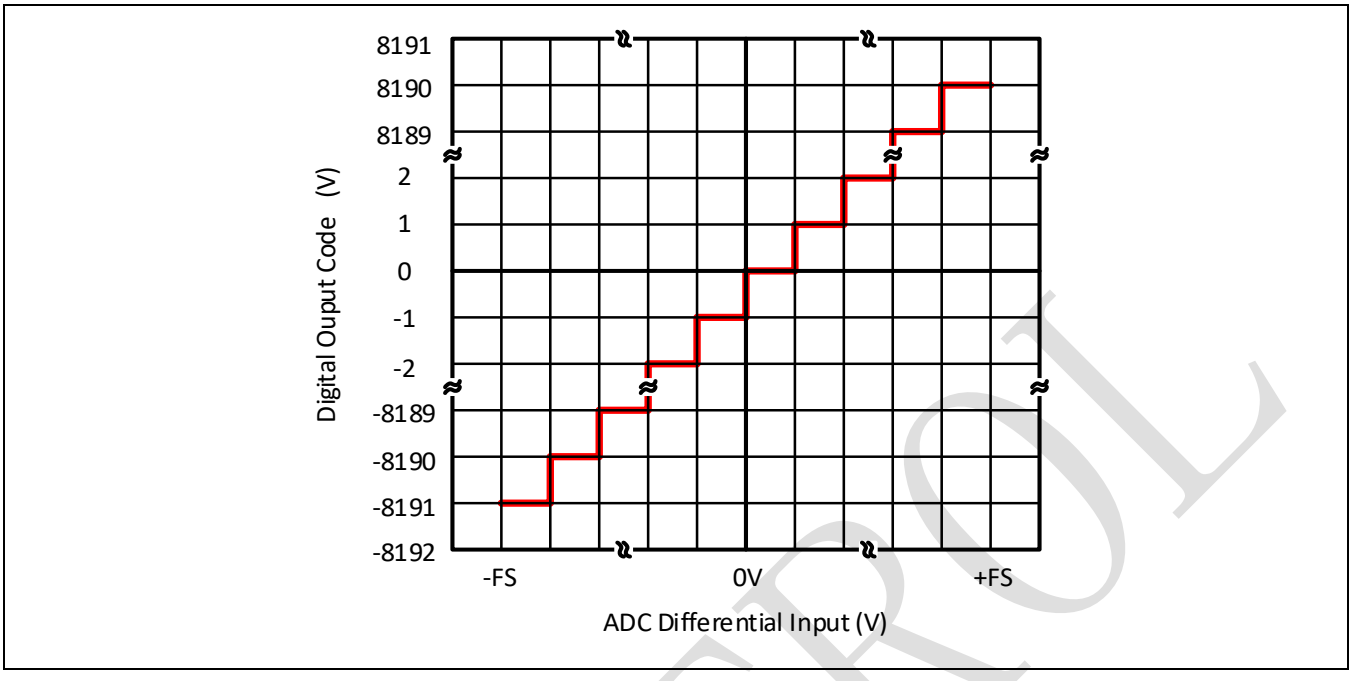


图 12-15：ADC 转换曲线



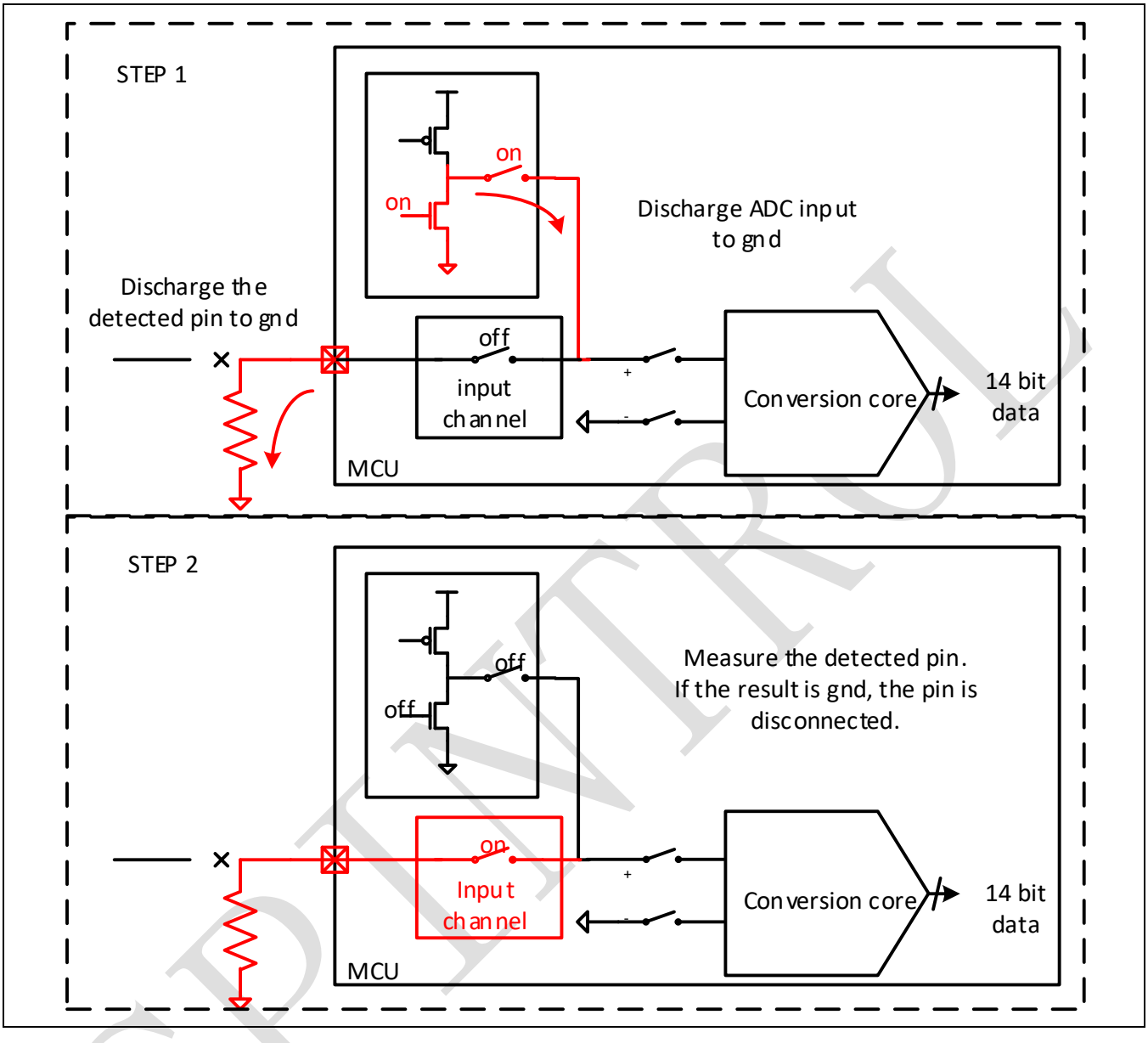
12.12 ADC 安全特性

12.12.1 模拟信号断路检测

如果传感器的模拟信号和 MCU 的连接断开，那么马达的位置、温度等等都不能被 MCU 正确的捕获，这将会引起严重事故。因此，ADC 的输入断路检测可以发现这个问题，从而有机会激活失效安全（fail-safe）程序。

断路探测功能可以通过配置 ADCCTL 寄存器来使能。ADC 中支持两种检测断路的场景，分别是预充电和预放电功能。

图 12-16：检测 ADC 输入 PIN 脚断路（放电）

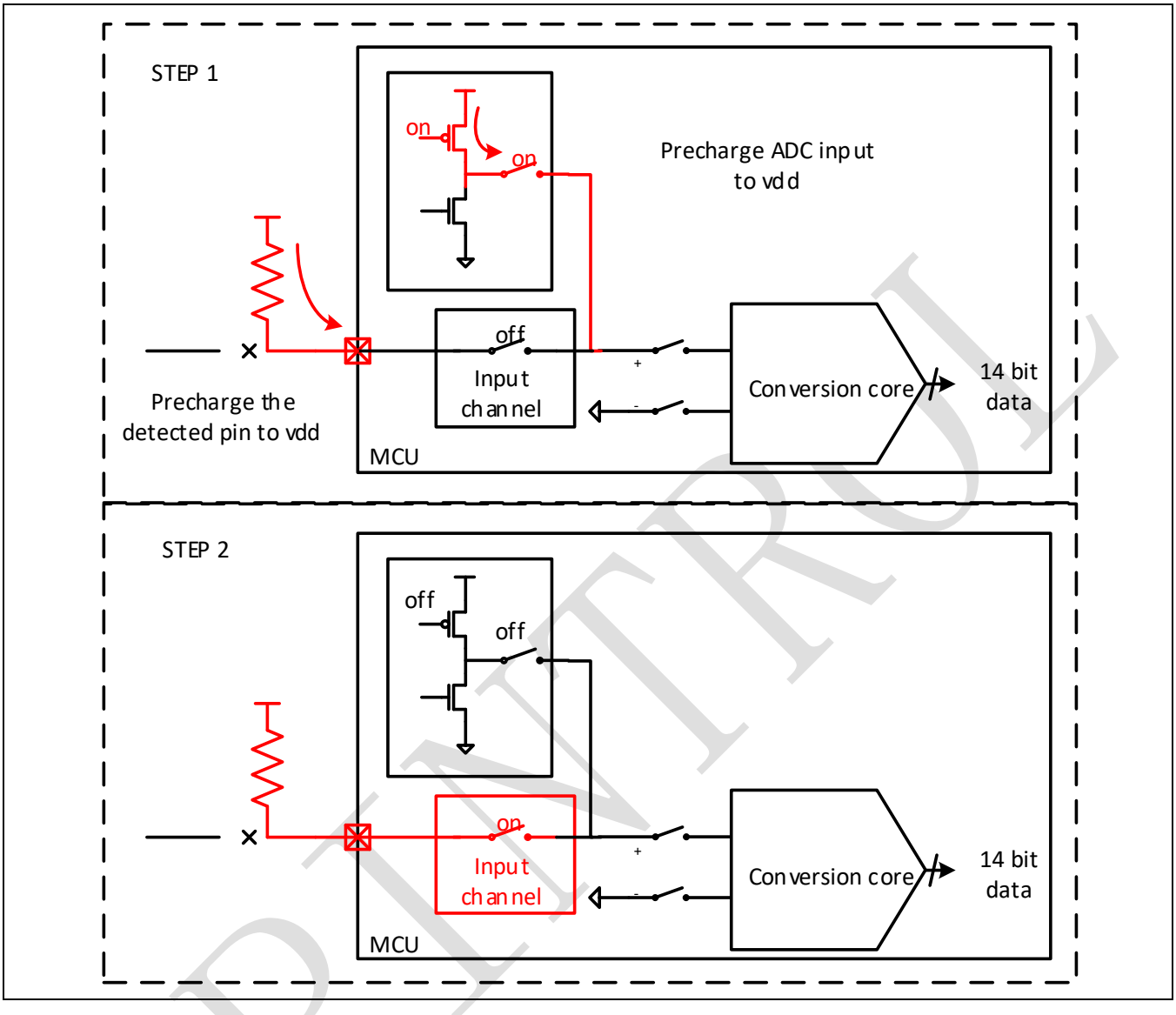


如图 12-16 所示，放电的过程如下所示，共需两步：

- 第一步，保持输入通道关闭。同时，分别将输入 PIN 脚节点和内部 ADC 输入采样节点放电到 GND。
- 第二步，使能输入通道开关。使用 ADC 测量输入 PIN 脚。如果 ADC 输出结果在 0V 附近，那么表明输入 PIN 总是在 0V 附近，此时没有外部驱动源。

  此时，我们认为输入 PIN 脚和驱动源是断开的。

图 12-17：检测 ADC 输入 PIN 脚断路（预充电）



如图 12-17 所示，预充电的过程如下所示，共需两步：

- 第一步，保持输入开关是关闭的。同时，分别将输入 PIN 脚节点和内部 ADC 输入采样节点预充电到 VDD。
- 第二步，输入开关闭合。使用 ADC 测量输入 PIN 脚电压。如果 ADC 输出结果在 3.3V（VDD=3.3V）附近。那么表明输入 PIN 总是在 VDD 附近，此时没有外部驱动源。

  此时，我们也认为输入 PIN 脚和驱动源是断开的。

12.12.2 输入信号短路检测

当来自传感器的模拟信号异常短路时，我们的 ADC 具有短路检测功能，这样可以给 MCU 提供失效安全处理的机会。

输入短路检测功能可以用 ADCCTL 寄存器设置。检测可以分为下面几步：

第一步，ADC 测量输入 PIN 脚电压，设为结果 DATA1。

第二步，给输入 PIN 接弱下拉 GND，ADC 测量输入 PIN 脚电压，设为结果 DATA2。

第三步，给输入 PIN 接弱上拉 VDD，ADC 测量输入 PIN 脚电压，设为结果 DATA3。

第四步，如果 DATA1= DATA2=DATA3，那么输入 PIN 脚短路。

注意： 只有当驱动源驱动能力很强的时候，才能测量输入 PIN 脚短路。假设 ADC 的分辨率为 1mV，ADC 内部的弱上拉或者弱下拉电流为 8uA，那么最大允许的驱动源阻抗为 1mV/8uA=125Ω。如果 ADC 输入驱动源能力较弱，它的输出阻抗大于这个值，那么就不能采用上述方法检测输入短路。

12.13 寄存器

12.13.1 ADC 寄存器列表

表 12-10：ADC 模块基地址

外设模块	基地址
ADC	0x4000 8C00

表 12-11：ADC 寄存器列表

寄存器	偏移地址	描述	复位值
ADCIF	0x00	ADC 中断标志寄存器	0x00000000
ADCIC	0x04	ADC 中断清除寄存器	0x00000000
ADCIOVF	0x08	ADC 中断溢出标志寄存器	0x00000000
ADCIOVFC	0x0C	ADC 中断溢出清除寄存器	0x00000000
ADCIE*	0x10	ADC 中断使能寄存器	0x00000000
ADCSOCPRICTL*	0x14	ADC SOC 优先级控制寄存器	0x00000000
ADCSOCFLG	0x18	ADC SOC 标志寄存器	0x00000000
ADCSOCFRC	0x1C	ADC SOC 软件强制寄存器	0x00000000
ADCSOCOVF	0x20	ADC SOC 溢出寄存器	0x00000000
ADCSOCOVFC	0x24	ADC SOC 溢出清除寄存器	0x00000000
ADCINTSOCEN*	0x28	ADC 中断触发 SOC 使能寄存器	0x00000000
ADCINTSOCSEL0*	0x2C	ADC 中断触发 SOC 选择寄存器 0	0x00000000
ADCINTSOCSEL1*	0x30	ADC 中断触发 SOC 选择寄存器 1	0x00000000
ADCEXTSOCCTL*	0x34	ADC 外部触发 SOC 控制寄存器	0x0000007F
ADCSOCCTL0*	0x38	ADC SOC0 控制寄存器	0x00000000
ADCSOCCTL1*	0x3C	ADC SOC1 控制寄存器	0x00000000
ADCSOCCTL2*	0x40	ADC SOC2 控制寄存器	0x00000000
ADCSOCCTL3*	0x44	ADC SOC3 控制寄存器	0x00000000
ADCSOCCTL4*	0x48	ADC SOC4 控制寄存器	0x00000000
ADCSOCCTL5*	0x4C	ADC SOC5 控制寄存器	0x00000000
ADCSOCCTL6*	0x50	ADC SOC6 控制寄存器	0x00000000
ADCSOCCTL7*	0x54	ADC SOC7 控制寄存器	0x00000000
ADCSOCCTL8*	0x58	ADC SOC8 控制寄存器	0x00000000
ADCSOCCTL9*	0x5C	ADC SOC9 控制寄存器	0x00000000
ADCSOCCTL10*	0x60	ADC SOC10 控制寄存器	0x00000000

寄存器	偏移地址	描述	复位值
ADCSOCCTL11*	0x64	ADC SOC11 控制寄存器	0x00000000
ADCSOCCTL12*	0x68	ADC SOC12 控制寄存器	0x00000000
ADCSOCCTL13*	0x6C	ADC SOC13 控制寄存器	0x00000000
ADCSOCCTL14*	0x70	ADC SOC14 控制寄存器	0x00000000
ADCSOCCTL15*	0x74	ADC SOC15 控制寄存器	0x00000000
ADCOFFSET0*	0x78	ADC 失调校准寄存器 0	0x00000000
ADCOFFSET1*	0x7C	ADC 失调校准寄存器 1	0x00000000
ADCOFFSET2*	0x80	ADC 失调校准寄存器 2	0x00000000
ADCOFFSET3*	0x84	ADC 失调校准寄存器 3	0x00000000
ADCOFFSET4*	0x88	ADC 失调校准寄存器 4	0x00000000
ADCOFFSET5*	0x8C	ADC 失调校准寄存器 5	0x00000000
ADCOFFSET6*	0x90	ADC 失调校准寄存器 6	0x00000000
ADCOFFSET7*	0x94	ADC 失调校准寄存器 7	0x00000000
ADCOFFSET8*	0x98	ADC 失调校准寄存器 8	0x00000000
ADCOFFSET9*	0x9C	ADC 失调校准寄存器 9	0x00000000
ADCOFFSET10*	0xA0	ADC 失调校准寄存器 10	0x00000000
ADCOFFSET11*	0xA4	ADC 失调校准寄存器 11	0x00000000
ADCOFFSET12*	0xA8	ADC 失调校准寄存器 12	0x00000000
ADCOFFSET13*	0xAC	ADC 失调校准寄存器 13	0x00000000
ADCOFFSET14*	0xB0	ADC 失调校准寄存器 14	0x00000000
ADCOFFSET15*	0xB4	ADC 失调校准寄存器 15	0x00000000
ADCGAIN0*	0xB8	ADC 增益校准寄存器 0	0x00008000
ADCGAIN1*	0xBC	ADC 增益校准寄存器 1	0x00008000
ADCGAIN2*	0xC0	ADC 增益校准寄存器 2	0x00008000
ADCGAIN3*	0xC4	ADC 增益校准寄存器 3	0x00008000
ADCGAIN4*	0xC8	ADC 增益校准寄存器 4	0x00008000
ADCGAIN5*	0xCC	ADC 增益校准寄存器 5	0x00008000
ADCGAIN6*	0xD0	ADC 增益校准寄存器 6	0x00008000
ADCGAIN7*	0xD4	ADC 增益校准寄存器 7	0x00008000
ADCGAIN8*	0xD8	ADC 增益校准寄存器 8	0x00008000
ADCGAIN9*	0xDC	ADC 增益校准寄存器 9	0x00008000
ADCGAIN10*	0xE0	ADC 增益校准寄存器 10	0x00008000
ADCGAIN11*	0xE4	ADC 增益校准寄存器 11	0x00008000

寄存器	偏移地址	描述	复位值
ADCGAIN12*	0xE8	ADC 增益校准寄存器 12	0x00008000
ADCGAIN13*	0xEC	ADC 增益校准寄存器 13	0x00008000
ADCGAIN14*	0xF0	ADC 增益校准寄存器 14	0x00008000
ADCGAIN15*	0xF4	ADC 增益校准寄存器 15	0x00008000
ADCOFFSETA*	0xF8	ADC SHA 失调校准寄存器	0x00000000
ADCOFFSETB*	0xFC	ADC SHB 失调校准寄存器	0x00000000
ADCOFFSETC*	0x100	ADC SHC 失调校准寄存器	0x00000000
ADCGAINA*	0x104	ADC SHA 增益校准寄存器	0x00008000
ADCGAINB*	0x108	ADC SHB 增益校准寄存器	0x00008000
ADCGAINC*	0x10C	ADC SHC 增益校准寄存器	0x00008000
ADCSTS	0x110	ADC 状态寄存器	0x00000000
ADCSTSCLR	0x114	ADC 状态清除寄存器	0x00000000
ADCCTL*	0x118	ADC 控制寄存器	0x00000008
ADCBGCTL*	0x11C	ADC 带隙基准控制寄存器	0x00000000
ADCREFCTL*	0x120	ADC 基准电压控制寄存器	0x00003D3C
ADCRAWCODEA	0x124	ADC SHA 转换原始结果寄存器	0x00000000
ADCRAWCODEB	0x128	ADC SHB 转换原始结果寄存器	0x00000000
ADCRAWCODEC	0x12C	ADC SHC 转换原始结果寄存器	0x00000000
ADCRESULT0	0x130	ADC 转换结果寄存器 0	0x00000000
ADCRESULT1	0x134	ADC 转换结果寄存器 1	0x00000000
ADCRESULT2	0x138	ADC 转换结果寄存器 2	0x00000000
ADCRESULT3	0x13C	ADC 转换结果寄存器 3	0x00000000
ADCRESULT4	0x140	ADC 转换结果寄存器 4	0x00000000
ADCRESULT5	0x144	ADC 转换结果寄存器 5	0x00000000
ADCRESULT6	0x148	ADC 转换结果寄存器 6	0x00000000
ADCRESULT7	0x14C	ADC 转换结果寄存器 7	0x00000000
ADCRESULT8	0x150	ADC 转换结果寄存器 8	0x00000000
ADCRESULT9	0x154	ADC 转换结果寄存器 9	0x00000000
ADCRESULT10	0x158	ADC 转换结果寄存器 10	0x00000000
ADCRESULT11	0x15C	ADC 转换结果寄存器 11	0x00000000
ADCRESULT12	0x160	ADC 转换结果寄存器 12	0x00000000
ADCRESULT13	0x164	ADC 转换结果寄存器 13	0x00000000
ADCRESULT14	0x168	ADC 转换结果寄存器 14	0x00000000

寄存器	偏移地址	描述	复位值
ADCRESULT15	0x16C	ADC 转换结果寄存器 15	0x00000000
ADCPPURESULT0	0x170	ADC PPU0 输出结果寄存器	0x00000000
ADCPPURESULT1	0x174	ADC PPU1 输出结果寄存器	0x00000000
ADCPPURESULT2	0x178	ADC PPU2 输出结果寄存器	0x00000000
ADCPPURESULT3	0x17C	ADC PPU3 输出结果寄存器	0x00000000
ADCPPURESULT4	0x180	ADC PPU4 输出结果寄存器	0x00000000
ADCPPURESULT5	0x184	ADC PPU5 输出结果寄存器	0x00000000
ADCPPURESULT6	0x188	ADC PPU6 输出结果寄存器	0x00000000
ADCPPURESULT7	0x18C	ADC PPU7 输出结果寄存器	0x00000000
ADCPPUSOCDLY0	0x190	ADC PPU0 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY1	0x194	ADC PPU1 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY2	0x198	ADC PPU2 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY3	0x19C	ADC PPU3 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY4	0x1A0	ADC PPU4 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY5	0x1A4	ADC PPU5 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY6	0x1A8	ADC PPU6 SOC 延迟捕获寄存器	0x00000000
ADCPPUSOCDLY7	0x1AC	ADC PPU7 SOC 延迟捕获寄存器	0x00000000
ADCPPUIF0	0x1B0	ADC PPU0 中断标志寄存器	0x00000000
ADCPPUIF1	0x1B4	ADC PPU1 中断标志寄存器	0x00000000
ADCPPUIF2	0x1B8	ADC PPU2 中断标志寄存器	0x00000000
ADCPPUIF3	0x1BC	ADC PPU3 中断标志寄存器	0x00000000
ADCPPUIF4	0x1C0	ADC PPU4 中断标志寄存器	0x00000000
ADCPPUIF5	0x1C4	ADC PPU5 中断标志寄存器	0x00000000
ADCPPUIF6	0x1C8	ADCPPU6 中断标志寄存器	0x00000000
ADCPPUIF7	0x1CC	ADCPPU7 中断标志寄存器	0x00000000
ADCPPUIC0	0x1D0	ADC PPU0 中断清除寄存器	0x00000000
ADCPPUIC1	0x1D4	ADC PPU1 中断清除寄存器	0x00000000
ADCPPUIC2	0x1D8	ADC PPU2 中断清除寄存器	0x00000000
ADCPPUIC3	0x1DC	ADC PPU3 中断清除寄存器	0x00000000
ADCPPUIC4	0x1E0	ADC PPU4 中断清除寄存器	0x00000000
ADCPPUIC5	0x1E4	ADC PPU5 中断清除寄存器	0x00000000
ADCPPUIC6	0x1E8	ADCPPU6 中断清除寄存器	0x00000000
ADCPPUIC7	0x1EC	ADCPPU7 中断清除寄存器	0x00000000

寄存器	偏移地址	描述	复位值
ADCPPUIE0*	0x1F0	ADC PPU0 中断使能寄存器	0x00000000
ADCPPUIE1*	0x1F4	ADC PPU1 中断使能寄存器	0x00000000
ADCPPUIE2*	0x1F8	ADC PPU2 中断使能寄存器	0x00000000
ADCPPUIE3*	0x1FC	ADC PPU3 中断使能寄存器	0x00000000
ADCPPUIE4*	0x200	ADC PPU4 中断使能寄存器	0x00000000
ADCPPUIE5*	0x204	ADC PPU5 中断使能寄存器	0x00000000
ADCPPUIE6*	0x208	ADC PPU6 中断使能寄存器	0x00000000
ADCPPUIE7*	0x20C	ADC PPU7 中断使能寄存器	0x00000000
ADCPPUTZE0*	0x210	ADC PPU0 封锁事件使能寄存器	0x00000000
ADCPPUTZE1*	0x214	ADC PPU1 封锁事件使能寄存器	0x00000000
ADCPPUTZE2*	0x218	ADC PPU2 封锁事件使能寄存器	0x00000000
ADCPPUTZE3*	0x21C	ADC PPU3 封锁事件使能寄存器	0x00000000
ADCPPUTZE4*	0x220	ADC PPU4 封锁事件使能寄存器	0x00000000
ADCPPUTZE5*	0x224	ADC PPU5 封锁事件使能寄存器	0x00000000
ADCPPUTZE6*	0x228	ADCPPU6 封锁事件使能寄存器	0x00000000
ADCPPUTZE7*	0x22C	ADCPPU7 封锁事件使能寄存器	0x00000000
ADCPPUCTL0*	0x230	ADC PPU0 控制寄存器	0x00000400
ADCPPUCTL1*	0x234	ADC PPU1 控制寄存器	0x00000400
ADCPPUCTL2*	0x238	ADC PPU2 控制寄存器	0x00000400
ADCPPUCTL3*	0x23C	ADC PPU3 控制寄存器	0x00000400
ADCPPUCTL4*	0x240	ADC PPU4 控制寄存器	0x00000400
ADCPPUCTL5*	0x244	ADC PPU5 控制寄存器	0x00000400
ADCPPUCTL6*	0x248	ADCPPU6 控制寄存器	0x00000400
ADCPPUCTL7*	0x24C	ADCPPU7 控制寄存器	0x00000400
ADCPPUREF0*	0x250	ADC PPU0 基准寄存器	0x00000000
ADCPPUREF1*	0x254	ADC PPU1 基准寄存器	0x00000000
ADCPPUREF2*	0x258	ADC PPU2 基准寄存器	0x00000000
ADCPPUREF3*	0x25C	ADC PPU3 基准寄存器	0x00000000
ADCPPUREF4*	0x260	ADC PPU4 基准寄存器	0x00000000
ADCPPUREF5*	0x264	ADC PPU5 基准寄存器	0x00000000
ADCPPUREF6*	0x268	ADC PPU6 基准寄存器	0x00000000
ADCPPUREF7*	0x26C	ADC PPU7 基准寄存器	0x00000000
ADCPPUTHH0	0x270	ADC PPU0 过高检测阈值寄存器	0x00000000

寄存器	偏移地址	描述	复位值
ADCPPUTHH1*	0x274	ADC PPU1 过高检测阈值寄存器	0x00000000
ADCPPUTHH2*	0x278	ADC PPU2 过高检测阈值寄存器	0x00000000
ADCPPUTHH3*	0x27C	ADC PPU3 过高检测阈值寄存器	0x00000000
ADCPPUTHH4*	0x280	ADC PPU4 过高检测阈值寄存器	0x00000000
ADCPPUTHH5*	0x284	ADC PPU5 过高检测阈值寄存器	0x00000000
ADCPPUTHH6*	0x288	ADC PPU6 过高检测阈值寄存器	0x00000000
ADCPPUTHH7*	0x28C	ADC PPU7 过高检测阈值寄存器	0x00000000
ADCPPUTHL0*	0x290	ADC PPU0 过低检测阈值寄存器	0x00000000
ADCPPUTHL1*	0x294	ADC PPU1 过低检测阈值寄存器	0x00000000
ADCPPUTHL2*	0x298	ADC PPU2 过低检测阈值寄存器	0x00000000
ADCPPUTHL3*	0x29C	ADC PPU3 过低检测阈值寄存器	0x00000000
ADCPPUTHL4*	0x2A0	ADC PPU4 过低检测阈值寄存器	0x00000000
ADCPPUTHL5*	0x2A4	ADC PPU5 过低检测阈值寄存器	0x00000000
ADCPPUTHL6*	0x2A8	ADC PPU6 过低检测阈值寄存器	0x00000000
ADCPPUTHL7*	0x2AC	ADC PPU7 过低检测阈值寄存器	0x00000000
TSENSCTL*	0x2B0	温度传感器控制寄存器	0x00000000
ADCREGKEY	0x2B4	ADC 模块写使能寄存器	0x1ACCE551

注意： 由*标识的寄存器仅当 ADCREGKEY=0x1ACCE551 时才可以改写。

12.13.2 ADC 寄存器

表 12-12 到表 12-359 列出了 ADC 模块相关寄存器的详细信息。

表 12-12：ADC 中断标志寄存器（ADCIF）位段定义

ADCIF (ADC Interrupt Flag Register) Offset: 0x0 Default: 0x00000000 Access: ADC -> ADCIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 INT15	14 INT14	13 INT13	12 INT12	11 INT11	10 INT10	9 INT9	8 INT8
7 INT7	6 INT6	5 INT5	4 INT4	3 INT3	2 INT2	1 INT1	0 INT0

表 12-13：ADC 中断标志寄存器（ADCIF）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	INT15	RO	0x0	ADC 中断 15 标志 0：ADC 中断 15 未发生 1：ADC 中断 15 已发生
14	INT14	RO	0x0	ADC 中断 14 标志 0：ADC 中断 14 未发生 1：ADC 中断 14 已发生
13	INT13	RO	0x0	ADC 中断 13 标志 0：ADC 中断 13 未发生 1：ADC 中断 13 已发生
12	INT12	RO	0x0	ADC 中断 12 标志 0：ADC 中断 12 未发生 1：ADC 中断 12 已发生
11	INT11	RO	0x0	ADC 中断 11 标志 0：ADC 中断 11 未发生 1：ADC 中断 11 已发生
10	INT10	RO	0x0	ADC 中断 10 标志 0：ADC 中断 10 未发生 1：ADC 中断 10 已发生
9	INT9	RO	0x0	ADC 中断 9 标志 0：ADC 中断 9 未发生 1：ADC 中断 9 已发生

位段	位段名	属性	复位值	描述
8	INT8	RO	0x0	ADC 中断 8 标志 0：ADC 中断 8 未发生 1：ADC 中断 8 已发生
7	INT7	RO	0x0	ADC 中断 7 标志 0：ADC 中断 7 未发生 1：ADC 中断 7 已发生
6	INT6	RO	0x0	ADC 中断 6 标志 0：ADC 中断 6 未发生 1：ADC 中断 6 已发生
5	INT5	RO	0x0	ADC 中断 5 标志 0：ADC 中断 5 未发生 1：ADC 中断 5 已发生
4	INT4	RO	0x0	ADC 中断 4 标志 0：ADC 中断 4 未发生 1：ADC 中断 4 已发生
3	INT3	RO	0x0	ADC 中断 3 标志 0：ADC 中断 3 未发生 1：ADC 中断 3 已发生
2	INT2	RO	0x0	ADC 中断 2 标志 0：ADC 中断 2 未发生 1：ADC 中断 2 已发生
1	INT1	RO	0x0	ADC 中断 1 标志 0：ADC 中断 1 未发生 1：ADC 中断 1 已发生
0	INT0	RO	0x0	ADC 中断 0 标志 0：ADC 中断 0 未发生 1：ADC 中断 0 已发生

表 12-14：ADC 中断清除寄存器（ADCIC）位段定义

ADCIC (ADC Interrupt Flag Clear Register) Offset: 0x4 Default: 0x00000000 Access: ADC -> ADCIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 INT15	14 INT14	13 INT13	12 INT12	11 INT11	10 INT10	9 INT9	8 INT8
7 INT7	6 INT6	5 INT5	4 INT4	3 INT3	2 INT2	1 INT1	0 INT0

表 12-15：ADC 中断清除寄存器（ADCIC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	INT15	W1C	0x0	清除 ADC 中断 15 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT15。本位段自动清零。
14	INT14	W1C	0x0	清除 ADC 中断 14 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT14。本位段自动清零。
13	INT13	W1C	0x0	清除 ADC 中断 13 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT13。本位段自动清零。
12	INT12	W1C	0x0	清除 ADC 中断 12 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT12。本位段自动清零。
11	INT11	W1C	0x0	清除 ADC 中断 11 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT11。本位段自动清零。
10	INT10	W1C	0x0	清除 ADC 中断 10 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT10。本位段自动清零。
9	INT9	W1C	0x0	清除 ADC 中断 9 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT9。本位段自动清零。
8	INT8	W1C	0x0	清除 ADC 中断 8 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT8。本位段自动清零。

位段	位段名	属性	复位值	描述
7	INT7	W1C	0x0	清除 ADC 中断 7 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT7。本位段自动清零。
6	INT6	W1C	0x0	清除 ADC 中断 6 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT6。本位段自动清零。
5	INT5	W1C	0x0	清除 ADC 中断 5 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT5。本位段自动清零。
4	INT4	W1C	0x0	清除 ADC 中断 4 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT4。本位段自动清零。
3	INT3	W1C	0x0	清除 ADC 中断 3 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT3。本位段自动清零。
2	INT2	W1C	0x0	清除 ADC 中断 2 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT2。本位段自动清零。
1	INT1	W1C	0x0	清除 ADC 中断 1 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT1。本位段自动清零。
0	INT0	W1C	0x0	清除 ADC 中断 0 标志 0：写 0 无效，总是读回 0 1：写 1 清除 ADCIF.INT0。本位段自动清零。

表 12-16：ADC 中断溢出标志寄存器（ADCIOVF）位段定义

ADCIOVF (ADC Interrupt Overflow Flag Register) Offset: 0x8 Default: 0x00000000 Access: ADC -> ADCIOVF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 INT15	14 INT14	13 INT13	12 INT12	11 INT11	10 INT10	9 INT9	8 INT8
7 INT7	6 INT6	5 INT5	4 INT4	3 INT3	2 INT2	1 INT1	0 INT0

表 12-17：ADC 中断溢出标志寄存器（ADCIOVF）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	INT15	RO	0x0	ADC 中断 15 溢出标志 0：未溢出 1：ADC 中断 15 溢出
14	INT14	RO	0x0	ADC 中断 14 溢出标志 0：未溢出 1：ADC 中断 14 溢出
13	INT13	RO	0x0	ADC 中断 13 溢出标志 0：未溢出 1：ADC 中断 13 溢出
12	INT12	RO	0x0	ADC 中断 12 溢出标志 0：未溢出 1：ADC 中断 12 溢出
11	INT11	RO	0x0	ADC 中断 11 溢出标志 0：未溢出 1：ADC 中断 11 溢出
10	INT10	RO	0x0	ADC 中断 10 溢出标志 0：未溢出 1：ADC 中断 10 溢出
9	INT9	RO	0x0	ADC 中断 9 溢出标志 0：未溢出 1：ADC 中断 9 溢出
8	INT8	RO	0x0	ADC 中断 8 溢出标志 0：未溢出 1：ADC 中断 8 溢出

位段	位段名	属性	复位值	描述
7	INT7	RO	0x0	ADC 中断 7 溢出标志 0：未溢出 1：ADC 中断 7 溢出
6	INT6	RO	0x0	ADC 中断 6 溢出标志 0：未溢出 1：ADC 中断 6 溢出
5	INT5	RO	0x0	ADC 中断 5 溢出标志 0：未溢出 1：ADC 中断 5 溢出
4	INT4	RO	0x0	ADC 中断 4 溢出标志 0：未溢出 1：ADC 中断 4 溢出
3	INT3	RO	0x0	ADC 中断 3 溢出标志 0：未溢出 1：ADC 中断 3 溢出
2	INT2	RO	0x0	ADC 中断 2 溢出标志 0：未溢出 1：ADC 中断 2 溢出
1	INT1	RO	0x0	ADC 中断 1 溢出标志 0：未溢出 1：ADC 中断 1 溢出
0	INT0	RO	0x0	ADC 中断 0 溢出标志 0：未溢出 1：ADC 中断 0 溢出

表 12-18：ADC 中断溢出清除寄存器（ADCIOVFC）位段定义

ADCIOVFC (ADC Interrupt Overflow Flag Clear Register) Offset: 0xC Default: 0x00000000 Access: ADC -> ADCIOVFC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 INT15	14 INT14	13 INT13	12 INT12	11 INT11	10 INT10	9 INT9	8 INT8
7 INT7	6 INT6	5 INT5	4 INT4	3 INT3	2 INT2	1 INT1	0 INT0

表 12-19：ADC 中断溢出清除寄存器（ADCIOVFC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	INT15	W1C	0x0	清除 ADC 中断 15 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
14	INT14	W1C	0x0	清除 ADC 中断 14 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
13	INT13	W1C	0x0	清除 ADC 中断 13 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
12	INT12	W1C	0x0	清除 ADC 中断 12 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
11	INT11	W1C	0x0	清除 ADC 中断 11 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
10	INT10	W1C	0x0	清除 ADC 中断 10 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
9	INT9	W1C	0x0	清除 ADC 中断 9 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
8	INT8	W1C	0x0	清除 ADC 中断 8 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。

位段	位段名	属性	复位值	描述
7	INT7	W1C	0x0	清除 ADC 中断 7 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
6	INT6	W1C	0x0	清除 ADC 中断 6 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
5	INT5	W1C	0x0	清除 ADC 中断 5 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
4	INT4	W1C	0x0	清除 ADC 中断 4 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
3	INT3	W1C	0x0	清除 ADC 中断 3 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
2	INT2	W1C	0x0	清除 ADC 中断 2 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
1	INT1	W1C	0x0	清除 ADC 中断 1 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。
0	INT0	W1C	0x0	清除 ADC 中断 0 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志。本位段自动清除。

表 12-20：ADC 中断使能寄存器（ADCIE）位段定义

ADCIE (ADC Interrupt Enable Register) Offset: 0x10 Default: 0x00000000 Access: ADC -> ADCIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 INT15	14 INT14	13 INT13	12 INT12	11 INT11	10 INT10	9 INT9	8 INT8
7 INT7	6 INT6	5 INT5	4 INT4	3 INT3	2 INT2	1 INT1	0 INT0

表 12-21：ADC 中断使能寄存器（ADCIE）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	INT15	RW	0x0	ADC 中断 15 使能 0：关闭 ADC 中断 15 1：使能 ADC 中断 15
14	INT14	RW	0x0	ADC 中断 14 使能 0：关闭 ADC 中断 14 1：使能 ADC 中断 14
13	INT13	RW	0x0	ADC 中断 13 使能 0：关闭 ADC 中断 13 1：使能 ADC 中断 13
12	INT12	RW	0x0	ADC 中断 12 使能 0：关闭 ADC 中断 12 1：使能 ADC 中断 12
11	INT11	RW	0x0	ADC 中断 11 使能 0：关闭 ADC 中断 11 1：使能 ADC 中断 11
10	INT10	RW	0x0	ADC 中断 10 使能 0：关闭 ADC 中断 10 1：使能 ADC 中断 10
9	INT9	RW	0x0	ADC 中断 9 使能 0：关闭 ADC 中断 9 1：使能 ADC 中断 9
8	INT8	RW	0x0	ADC 中断 8 使能 0：关闭 ADC 中断 8 1：使能 ADC 中断 8
7	INT7	RW	0x0	ADC 中断 7 使能 0：关闭 ADC 中断 7 1：使能 ADC 中断 7
6	INT6	RW	0x0	ADC 中断 6 使能 0：关闭 ADC 中断 6 1：使能 ADC 中断 6
5	INT5	RW	0x0	ADC 中断 5 使能 0：关闭 ADC 中断 5 1：使能 ADC 中断 5
4	INT4	RW	0x0	ADC 中断 4 使能 0：关闭 ADC 中断 4 1：使能 ADC 中断 4

位段	位段名	属性	复位值	描述
3	INT3	RW	0x0	ADC 中断 3 使能 0：关闭 ADC 中断 3 1：使能 ADC 中断 3
2	INT2	RW	0x0	ADC 中断 2 使能 0：关闭 ADC 中断 2 1：使能 ADC 中断 2
1	INT1	RW	0x0	ADC 中断 1 使能 0：关闭 ADC 中断 1 1：使能 ADC 中断 1
0	INT0	RW	0x0	ADC 中断 0 使能 0：关闭 ADC 中断 0 1：使能 ADC 中断 0

表 12-22：ADC SOC 优先级控制寄存器（ADCSOCPRICTL）位段定义

ADCSOCPRICTL (ADC Start of Conversion Priority Control Register) Offset: 0x14 Default: 0x00000000 Access: ADC -> ADCSOCPRICTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				PRIORITY			

表 12-23：ADC SOC 优先级控制寄存器（ADCSOCPRICTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3:0	PRIORITY	RW	0x0	SOC（开始采样）优先级 指定 SOC 高优先级模式和 Round-Robin 仲裁的截止点 0000：所有 SOC 的优先级遵循 Round-Robin 规则 0001：SOC0 为高优先级 SOC1-SOC15 遵循 Round-Robin 规则 0010：SOC0-SOC1 为高优先级 SOC2-SOC15 遵循 Round-Robin 规则 0011：SOC0-SOC2 为高优先级 SOC3-SOC15 遵循 Round-Robin 规则 0100：SOC0-SOC3 为高优先级 SOC4-SOC15 遵循 Round-Robin 规则 0101：SOC0-SOC4 为高优先级 SOC5-SOC15 遵循 Round-Robin 规则 0110：SOC0-SOC5 为高优先级 SOC6-SOC15 遵循 Round-Robin 规则 0111：SOC0-SOC6 为高优先级 SOC7-SOC15 遵循 Round-Robin 规则 1000：SOC0-SOC7 为高优先级 SOC8-SOC15 遵循 Round-Robin 规则 1001：SOC0-SOC8 为高优先级 SOC9-SOC15 遵循 Round-Robin 规则 1010：SOC0-SOC9 为高优先级 SOC10-SOC15 遵循 Round-Robin 规则 1011：SOC0-SOC10 为高优先级 SOC11-SOC15 遵循 Round-Robin 规则 1100：SOC0-SOC11 为高优先级 SOC12-SOC15 遵循 Round-Robin 规则 1101：SOC0-SOC12 为高优先级 SOC13-SOC15 遵循 Round-Robin 规则 1110：SOC0-SOC13 为高优先级 SOC14-SOC15 遵循 Round-Robin 规则 1111：SOC0-SOC14 为高优先级 SOC15 遵循 Round-Robin 规则

表 12-24：ADC SOC 标志寄存器（ADCSOCFLG）位段定义

ADCSOCFLG (ADC SOC Flag Register) Offset: 0x18 Default: 0x00000000 Access: ADC -> ADCSOCFLG.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SOC15	14 SOC14	13 SOC13	12 SOC12	11 SOC11	10 SOC10	9 SOC9	8 SOC8
7 SOC7	6 SOC6	5 SOC5	4 SOC4	3 SOC3	2 SOC2	1 SOC1	0 SOC0

表 12-25：ADC SOC 标志寄存器（ADCSOCFLG）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SOC15	RO	0x0	SOC15（开始采样）标志 0：SOC15 没有采样请求 1：接收到触发信号，SOC15 采样请求挂起 当采样开始时，本位段自动清零。
14	SOC14	RO	0x0	SOC14（开始采样）标志 0：SOC14 没有采样请求 1：接收到触发信号，SOC14 采样请求挂起 当采样开始时，本位段自动清零。
13	SOC13	RO	0x0	SOC13（开始采样）标志 0：SOC13 没有采样请求 1：接收到触发信号，SOC13 采样请求挂起 当采样开始时，本位段自动清零。
12	SOC12	RO	0x0	SOC12（开始采样）标志 0：SOC12 没有采样请求 1：接收到触发信号，SOC12 采样请求挂起 当采样开始时，本位段自动清零。
11	SOC11	RO	0x0	SOC11（开始采样）标志 0：SOC11 没有采样请求 1：接收到触发信号，SOC11 采样请求挂起 当采样开始时，本位段自动清零。
10	SOC10	RO	0x0	SOC10（开始采样）标志 0：SOC10 没有采样请求 1：接收到触发信号，SOC10 采样请求挂起 当采样开始时，本位段自动清零。
9	SOC9	RO	0x0	SOC9（开始采样）标志 0：SOC9 没有采样请求

位段	位段名	属性	复位值	描述
				1：接收到触发信号，SOC9 采样请求挂起 当采样开始时，本位段自动清零。
8	SOC8	RO	0x0	SOC8（开始采样）标志 0：SOC8 没有采样请求 1：接收到触发信号，SOC8 采样请求挂起 当采样开始时，本位段自动清零。
7	SOC7	RO	0x0	SOC7（开始采样）标志 0：SOC7 没有采样请求 1：接收到触发信号，SOC7 采样请求挂起 当采样开始时，本位段自动清零。
6	SOC6	RO	0x0	SOC6（开始采样）标志 0：SOC6 没有采样请求 1：接收到触发信号，SOC6 采样请求挂起 当采样开始时，本位段自动清零。
5	SOC5	RO	0x0	SOC5（开始采样）标志 0：SOC5 没有采样请求 1：接收到触发信号，SOC5 采样请求挂起 当采样开始时，本位段自动清零。
4	SOC4	RO	0x0	SOC4（开始采样）标志 0：SOC4 没有采样请求 1：接收到触发信号，SOC4 采样请求挂起 当采样开始时，本位段自动清零。
3	SOC3	RO	0x0	SOC3（开始采样）标志 0：SOC3 没有采样请求 1：接收到触发信号，SOC3 采样请求挂起 当采样开始时，本位段自动清零。
2	SOC2	RO	0x0	SOC2（开始采样）标志 0：SOC2 没有采样请求 1：接收到触发信号，SOC2 采样请求挂起 当采样开始时，本位段自动清零。
1	SOC1	RO	0x0	SOC1（开始采样）标志 0：SOC1 没有采样请求 1：接收到触发信号，SOC1 采样请求挂起 当采样开始时，本位段自动清零。
0	SOC0	RO	0x0	SOC0（开始采样）标志 0：SOC0 没有采样请求 1：接收到触发信号，SOC0 采样请求挂起 当采样开始时，本位段自动清零。

表 12-26：ADC SOC 软件强制寄存器（ADCSOCFRC）位段定义

ADCSOCFRC (ADC SOC Force Register) Offset: 0x1C Default: 0x00000000 Access: ADC -> ADCSOCFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SOC15	14 SOC14	13 SOC13	12 SOC12	11 SOC11	10 SOC10	9 SOC9	8 SOC8
7 SOC7	6 SOC6	5 SOC5	4 SOC4	3 SOC3	2 SOC2	1 SOC1	0 SOC0

表 12-27：ADC SOC 软件强制寄存器（ADCSOCFRC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SOC15	W1S	0x0	软件强制 SOC15（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC15 本位段自动清零。
14	SOC14	W1S	0x0	软件强制 SOC14（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC14 本位段自动清零。
13	SOC13	W1S	0x0	软件强制 SOC13（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC13 本位段自动清零。
12	SOC12	W1S	0x0	软件强制 SOC12（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC12 本位段自动清零。
11	SOC11	W1S	0x0	软件强制 SOC11（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC11 本位段自动清零。
10	SOC10	W1S	0x0	软件强制 SOC10（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC10 本位段自动清零。
9	SOC9	W1S	0x0	软件强制 SOC9（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC9

位段	位段名	属性	复位值	描述
				本位段自动清零。
8	SOC8	W1S	0x0	软件强制 SOC8（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC8 本位段自动清零。
7	SOC7	W1S	0x0	软件强制 SOC7（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC7 本位段自动清零。
6	SOC6	W1S	0x0	软件强制 SOC6（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC6 本位段自动清零。
5	SOC5	W1S	0x0	软件强制 SOC5（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC5 本位段自动清零。
4	SOC4	W1S	0x0	软件强制 SOC4（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC4 本位段自动清零。
3	SOC3	W1S	0x0	软件强制 SOC3（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC3 本位段自动清零。
2	SOC2	W1S	0x0	软件强制 SOC2（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC2 本位段自动清零。
1	SOC1	W1S	0x0	软件强制 SOC1（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC1 本位段自动清零。
0	SOC0	W1S	0x0	软件强制 SOC0（开始采样） 0：写 0 无效，总是读回 0 1：写 1 强制 SOC0 本位段自动清零。

表 12-28：ADC SOC 溢出寄存器（ADCSOCOVF）位段定义

ADCSOCOVF (ADC SOC Overflow Register) Offset: 0x20 Default: 0x00000000 Access: ADC -> ADCSOCOVF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SOC15	14 SOC14	13 SOC13	12 SOC12	11 SOC11	10 SOC10	9 SOC9	8 SOC8
7 SOC7	6 SOC6	5 SOC5	4 SOC4	3 SOC3	2 SOC2	1 SOC1	0 SOC0

表 12-29：ADC SOC 溢出寄存器（ADCSOCOVF）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SOC15	RO	0x0	SOC15（开始采样）溢出标志 本标志表示当一个 SOC15 中断已被挂起时，一个新的 SOC15 中断产生了。本标志不会阻止 SOC15 中断的处理。它只是表明错过了一个触发中断。 0：SOC15 中断未溢出 1：SOC15 中断溢出
14	SOC14	RO	0x0	SOC14（开始采样）溢出标志 0：SOC14 中断未溢出 1：SOC14 中断溢出
13	SOC13	RO	0x0	SOC13（开始采样）溢出标志 0：SOC13 中断未溢出 1：SOC13 中断溢出
12	SOC12	RO	0x0	SOC12（开始采样）溢出标志 0：SOC12 中断未溢出 1：SOC12 中断溢出
11	SOC11	RO	0x0	SOC11（开始采样）溢出标志 0：SOC11 中断未溢出 1：SOC11 中断溢出
10	SOC10	RO	0x0	SOC10（开始采样）溢出标志 0：SOC10 中断未溢出 1：SOC10 中断溢出
9	SOC9	RO	0x0	SOC9（开始采样）溢出标志 0：SOC9 中断未溢出 1：SOC9 中断溢出

位段	位段名	属性	复位值	描述
8	SOC8	RO	0x0	SOC8（开始采样）溢出标志 0：SOC8 中断未溢出 1：SOC8 中断溢出
7	SOC7	RO	0x0	SOC7（开始采样）溢出标志 0：SOC7 中断未溢出 1：SOC7 中断溢出
6	SOC6	RO	0x0	SOC6（开始采样）溢出标志 0：SOC6 中断未溢出 1：SOC6 中断溢出
5	SOC5	RO	0x0	SOC5（开始采样）溢出标志 0：SOC5 中断未溢出 1：SOC5 中断溢出
4	SOC4	RO	0x0	SOC4（开始采样）溢出标志 0：SOC4 中断未溢出 1：SOC4 中断溢出
3	SOC3	RO	0x0	SOC3（开始采样）溢出标志 0：SOC3 中断未溢出 1：SOC3 中断溢出
2	SOC2	RO	0x0	SOC2（开始采样）溢出标志 0：SOC2 中断未溢出 1：SOC2 中断溢出
1	SOC1	RO	0x0	SOC1（开始采样）溢出标志 0：SOC1 中断未溢出 1：SOC1 中断溢出
0	SOC0	RO	0x0	SOC0（开始采样）溢出标志 0：SOC0 中断未溢出 1：SOC0 中断溢出

表 12-30：ADC SOC 溢出清除寄存器（ADCSOCOVFC）位段定义

ADCSOCOVFC (ADC SOC Overflow Clear Register) Offset: 0x24 Default: 0x00000000 Access: ADC -> ADCSOCOVFC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SOC15	14 SOC14	13 SOC13	12 SOC12	11 SOC11	10 SOC10	9 SOC9	8 SOC8
7 SOC7	6 SOC6	5 SOC5	4 SOC4	3 SOC3	2 SOC2	1 SOC1	0 SOC0

表 12-31：ADC SOC 溢出清除寄存器（ADCSOCOVFC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SOC15	W1C	0x0	清除 SOC15 溢出标志 如果软件将本位段置 1 的同时，硬件尝试将 ADCSOCOVF 寄存器中的 SOC15 位段置 1，那么软件具有较高的优先级，ADCSOCOVF.SOC15 将会被清除。 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
14	SOC14	W1C	0x0	清除 SOC14 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
13	SOC13	W1C	0x0	清除 SOC13 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
12	SOC12	W1C	0x0	清除 SOC12 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
11	SOC11	W1C	0x0	清除 SOC11 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
10	SOC10	W1C	0x0	清除 SOC10 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
9	SOC9	W1C	0x0	清除 SOC9 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。

位段	位段名	属性	复位值	描述
8	SOC8	W1C	0x0	清除 SOC8 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
7	SOC7	W1C	0x0	清除 SOC7 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
6	SOC6	W1C	0x0	清除 SOC6 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
5	SOC5	W1C	0x0	清除 SOC5 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
4	SOC4	W1C	0x0	清除 SOC4 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
3	SOC3	W1C	0x0	清除 SOC3 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
2	SOC2	W1C	0x0	清除 SOC2 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
1	SOC1	W1C	0x0	清除 SOC1 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。
0	SOC0	W1C	0x0	清除 SOC0 溢出标志 0：写 0 无效，总是读回 0 1：写 1 清除溢出标志，本位段自动清零。

表 12-32：ADC 中断触发 SOC 使能寄存器（ADCINTSOCEN）位段定义

ADCINTSOCEN (ADC Interrupt Trigger SOC Enable Register) Offset: 0x28 Default: 0x00000000 Access: ADC -> ADCINTSOCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SOC15	14 SOC14	13 SOC13	12 SOC12	11 SOC11	10 SOC10	9 SOC9	8 SOC8
7 SOC7	6 SOC6	5 SOC5	4 SOC4	3 SOC3	2 SOC2	1 SOC1	0 SOC0

表 12-33：ADC 中断触发 SOC 使能寄存器（ADCINTSOCEN）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SOC15	RW	0x0	使能 ADC 中断触发 SOC15 0：忽略 ADCINTSOCSEL1.SOC15 1：触发源 ADCSOCCTL15.TRIGSEL 被 ADCINTSOCSEL1.SOC15 代替
14	SOC14	RW	0x0	使能 ADC 中断触发 SOC14 0：忽略 ADCINTSOCSEL1.SOC14 1：触发源 ADCSOCCTL14.TRIGSEL 被 ADCINTSOCSEL1.SOC14 代替
13	SOC13	RW	0x0	使能 ADC 中断触发 SOC13 0：忽略 ADCINTSOCSEL1.SOC13 1：触发源 ADCSOCCTL13.TRIGSEL 被 ADCINTSOCSEL1.SOC13 代替
12	SOC12	RW	0x0	使能 ADC 中断触发 SOC12 0：忽略 ADCINTSOCSEL1.SOC12 1：触发源 ADCSOCCTL12.TRIGSEL 被 ADCINTSOCSEL1.SOC12 代替
11	SOC11	RW	0x0	使能 ADC 中断触发 SOC11 0：忽略 ADCINTSOCSEL1.SOC11 1：触发源 ADCSOCCTL11.TRIGSEL 被 ADCINTSOCSEL1.SOC11 代替
10	SOC10	RW	0x0	使能 ADC 中断触发 SOC10 0：忽略 ADCINTSOCSEL1.SOC10 1：触发源 ADCSOCCTL10.TRIGSEL 被 ADCINTSOCSEL1.SOC10 代替
9	SOC9	RW	0x0	使能 ADC 中断触发 SOC9 0：忽略 ADCINTSOCSEL1.SOC9

位段	位段名	属性	复位值	描述
				1：触发源 ADCSOCCTL9.TRIGSEL 被 ADCINTSOCSEL1.SOC9 代替
8	SOC8	RW	0x0	使能 ADC 中断触发 SOC8 0：忽略 ADCINTSOCSEL1.SOC8 1：触发源 ADCSOCCTL8.TRIGSEL 被 ADCINTSOCSEL1.SOC8 代替
7	SOC7	RW	0x0	使能 ADC 中断触发 SOC7 0：忽略 ADCINTSOCSEL1.SOC7 1：触发源 ADCSOCCTL7.TRIGSEL 被 ADCINTSOCSEL1.SOC7 代替
6	SOC6	RW	0x0	使能 ADC 中断触发 SOC6 0：忽略 ADCINTSOCSEL1.SOC6 1：触发源 ADCSOCCTL6.TRIGSEL 被 ADCINTSOCSEL1.SOC6 代替
5	SOC5	RW	0x0	使能 ADC 中断触发 SOC5 0：忽略 ADCINTSOCSEL1.SOC5 1：触发源 ADCSOCCTL5.TRIGSEL 被 ADCINTSOCSEL1.SOC5 代替
4	SOC4	RW	0x0	使能 ADC 中断触发 SOC4 0：忽略 ADCINTSOCSEL1.SOC4 1：触发源 ADCSOCCTL4.TRIGSEL 被 ADCINTSOCSEL1.SOC4 代替
3	SOC3	RW	0x0	使能 ADC 中断触发 SOC3 0：忽略 ADCINTSOCSEL1.SOC3 1：触发源 ADCSOCCTL3.TRIGSEL 被 ADCINTSOCSEL1.SOC3 代替
2	SOC2	RW	0x0	使能 ADC 中断触发 SOC2 0：忽略 ADCINTSOCSEL1.SOC2 1：触发源 ADCSOCCTL2.TRIGSEL 被 ADCINTSOCSEL1.SOC2 代替
1	SOC1	RW	0x0	使能 ADC 中断触发 SOC1 0：忽略 ADCINTSOCSEL1.SOC1 1：触发源 ADCSOCCTL1.TRIGSEL 被 ADCINTSOCSEL1.SOC1 代替
0	SOC0	RW	0x0	使能 ADC 中断触发 SOC0 0：忽略 ADCINTSOCSEL1.SOC0 1：触发源 ADCSOCCTL0.TRIGSEL 被 ADCINTSOCSEL1.SOC0 代替

表 12-34：ADC 中断触发 SOC 选择寄存器 0（ADCINTSOCSEL0）位段定义

ADCINTSOCSEL0 (ADC Interrupt Trigger SOC Select Register 0) Offset: 0x2C Default: 0x00000000 Access: ADC -> ADCINTSOCSEL0.all							
31	30	29	28	27	26	25	24
SOC7				SOC6			
23	22	21	20	19	18	17	16
SOC5				SOC4			
15	14	13	12	11	10	9	8
SOC3				SOC2			
7	6	5	4	3	2	1	0
SOC1				SOC0			

表 12-35：ADC 中断触发 SOC 选择寄存器 0（ADCINTSOCSEL0）位段描述

位段	位段名	属性	复位值	描述
31:28	SOC7	RW	0x0	触发 SOC7 的 ADC 中断号
27:24	SOC6	RW	0x0	触发 SOC6 的 ADC 中断号
23:20	SOC5	RW	0x0	触发 SOC5 的 ADC 中断号
19:16	SOC4	RW	0x0	触发 SOC4 的 ADC 中断号
15:12	SOC3	RW	0x0	触发 SOC3 的 ADC 中断号
11:8	SOC2	RW	0x0	触发 SOC2 的 ADC 中断号
7:4	SOC1	RW	0x0	触发 SOC1 的 ADC 中断号
3:0	SOC0	RW	0x0	触发 SOC0 的 ADC 中断号

表 12-36：ADC 中断触发 SOC 选择寄存器 1（ADCINTSOCSEL1）位段定义

ADCINTSOCSEL1 (ADC Interrupt Trigger SOC Select Register 1) Offset: 0x30 Default: 0x00000000 Access: ADC -> ADCINTSOCSEL1.all							
31	30	29	28	27	26	25	24
SOC15				SOC14			
23	22	21	20	19	18	17	16
SOC13				SOC12			
15	14	13	12	11	10	9	8
SOC11				SOC10			
7	6	5	4	3	2	1	0
SOC9				SOC8			

表 12-37：ADC 中断触发 SOC 选择寄存器 1（ADCINTSOCSEL1）位段描述

位段	位段名	属性	复位值	描述
31:28	SOC15	RW	0x0	触发 SOC15 的 ADC 中断号
27:24	SOC14	RW	0x0	触发 SOC14 的 ADC 中断号
23:20	SOC13	RW	0x0	触发 SOC13 的 ADC 中断号
19:16	SOC12	RW	0x0	触发 SOC12 的 ADC 中断号
15:12	SOC11	RW	0x0	触发 SOC11 的 ADC 中断号
11:8	SOC10	RW	0x0	触发 SOC10 的 ADC 中断号
7:4	SOC9	RW	0x0	触发 SOC9 的 ADC 中断号
3:0	SOC8	RW	0x0	触发 SOC8 的 ADC 中断号

表 12-38：ADC 外部触发 SOC 控制寄存器（ADCEXTSOCCTL）位段定义

ADCEXTSOCCTL (ADC External SOC Input Control Register) Offset: 0x34 Default: 0x0000007F Access: ADC -> ADCEXTSOCCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	POL	IOSEL					

表 12-39：ADC 外部触发 SOC 控制寄存器（ADCEXTSOCCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	POL	RW	0x1	SOC 外部触发源极性 0：低电平触发 1：高电平触发
5:0	IOSEL	RW	0x3F	选择 SOC 外部触发源（GPIO 管脚号）

表 12-40：ADC SOC0 控制寄存器（ADCSOCCTL0）位段定义

ADCSOCCTL0 (ADC SOC0 Control Register) Offset: 0x38 Default: 0x00000000 Access: ADC -> ADCSOCCTL0.all							
31	30	29	28	27	26	25	24
SHEN			TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-41：ADC SOC0 控制寄存器（ADCSOCCTL0）位段描述

位段	位段名	属性	复位值	描述
31:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC） 100：使能 SHA 和 SHB 101：使能 SHB 和 SHC 110：使能 SHA 和 SHC 111：使能 SHA、SHB 和 SHC
28:24	TRIGSEL	RW	0x0	选择 SOC0 触发源 本位段可以被 ADCINTSOCSEL0.SOC0 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA

位段	位段名	属性	复位值	描述
				10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC0 的 SH 负输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC0 的 SH 正输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC0 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC0 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-42：ADC SOC1 控制寄存器（ADCSOCCTL1）位段定义

ADCSOCCTL1 (ADC SOC1 Control Register) Offset: 0x3C Default: 0x00000000
Access: ADC -> ADCSOCCTL1.all

31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-43：ADC SOC1 控制寄存器（ADCSOCCTL1）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL0.SHEN 设为 4、5 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC1 触发源 本位段可以被 ADCINTSOCSEL0.SOC1 覆盖。 当 ADCSOCCTL0.SHEN 设为 4、5 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC1 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL0.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC0 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC1 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL0.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC0 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC1 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC1 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-44：ADC SOC2 控制寄存器（ADCSOCCTL2）位段定义

ADCSOCCTL2 (ADC SOC2 Control Register) Offset: 0x40 Default: 0x00000000
Access: ADC -> ADCSOCCTL2.all

31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-45：ADC SOC2 控制寄存器（ADCSOCCTL2）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL0.SHEN 设为 5、6 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC2 触发源 本位段可以被 ADCINTSOCSEL0.SOC2 覆盖。 当 ADCSOCCTL0.SHEN 设为 5、6 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC2 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL0.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC0 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC2 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL0.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC0 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC2 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC2 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-46：ADC SOC3 控制寄存器（ADCSOCCTL3）位段定义

ADCSOCCTL3 (ADC SOC3 Control Register) Offset: 0x44 Default: 0x00000000 Access: ADC -> ADCSOCCTL3.all							
31	30	29	28	27	26	25	24
SHEN			TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-47：ADC SOC3 控制寄存器（ADCSOCCTL3）位段描述

位段	位段名	属性	复位值	描述
31:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC） 100：使能 SHA 和 SHB 101：使能 SHB 和 SHC 110：使能 SHA 和 SHC 111：使能 SHA、SHB 和 SHC
28:24	TRIGSEL	RW	0x0	选择 SOC3 触发源 本位段可以被 ADCINTSOCSEL0.SOC3 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA

位段	位段名	属性	复位值	描述
				10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC3 的 SH 负输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC3 的 SH 正输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC3 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC3 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-48：ADC SOC4 控制寄存器（ADCSOCCTL4）位段定义

ADCSOCCTL4 (ADC SOC4 Control Register) Offset: 0x48 Default: 0x00000000 Access: ADC -> ADCSOCCTL4.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-49：ADC SOC4 控制寄存器（ADCSOCCTL4）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL3.SHEN 设为 4、5 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC4 触发源 本位段可以被 ADCINTSOCSEL0.SOC4 覆盖。 当 ADCSOCCTL3.SHEN 设为 4、5 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC4 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL3.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC3 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC4 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL3.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC3 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC4 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC4 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-50：ADC SOC5 控制寄存器（ADCSOCCTL5）位段定义

ADCSOCCTL5 (ADC SOC5 Control Register) Offset: 0x4C Default: 0x00000000 Access: ADC -> ADCSOCCTL5.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-51：ADC SOC5 控制寄存器（ADCSOCCTL5）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL3.SHEN 设为 5、6 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC5 触发源 本位段可以被 ADCINTSOCSEL0.SOC5 覆盖。 当 ADCSOCCTL3.SHEN 设为 5、6 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC5 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL3.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC3 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC5 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL3.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC3 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC5 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC5 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-52：ADC SOC6 控制寄存器（ADCSOCCTL6）位段定义

ADCSOCCTL6 (ADC SOC6 Control Register) Offset: 0x50 Default: 0x00000000 Access: ADC -> ADCSOCCTL6.all							
31	30	29	28	27	26	25	24
SHEN			TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-53：ADC SOC6 控制寄存器（ADCSOCCTL6）位段描述

位段	位段名	属性	复位值	描述
31:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC） 100：使能 SHA 和 SHB 101：使能 SHB 和 SHC 110：使能 SHA 和 SHC 111：使能 SHA、SHB 和 SHC
28:24	TRIGSEL	RW	0x0	选择 SOC6 触发源 本位段可以被 ADCINTSOCSEL0.SOC6 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA

位段	位段名	属性	复位值	描述
				10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC6 的 SH 负输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC6 的 SH 正输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC6 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC6 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-54：ADC SOC7 控制寄存器（ADCSOCCTL7）位段定义

ADCSOCCTL7 (ADC SOC7 Control Register) Offset: 0x54 Default: 0x00000000 Access: ADC -> ADCSOCCTL7.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-55：ADC SOC7 控制寄存器（ADCSOCCTL7）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL6.SHEN 设为 4、5 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC7 触发源 本位段可以被 ADCINTSOCSEL0.SOC7 覆盖。 当 ADCSOCCTL6.SHEN 设为 4、5 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC7 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL6.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC6 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC7 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL6.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC6 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC7 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC7 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-56：ADC SOC8 控制寄存器（ADCSOCCTL8）位段定义

ADCSOCCTL8 (ADC SOC8 Control Register) Offset: 0x58 Default: 0x00000000
Access: ADC -> ADCSOCCTL8.all

31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-57：ADC SOC8 控制寄存器（ADCSOCCTL8）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL6.SHEN 设为 5、6 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC8 触发源 本位段可以被 ADCINTSOCSEL1.SOC8 覆盖。 当 ADCSOCCTL6.SHEN 设为 5、6 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC8 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL6.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC6 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC8 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL6.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC6 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC8 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC8 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-58：ADC SOC9 控制寄存器（ADCSOCCTL9）位段定义

ADCSOCCTL9 (ADC SOC9 Control Register) Offset: 0x5C Default: 0x00000000 Access: ADC -> ADCSOCCTL9.all							
31	30	29	28	27	26	25	24
SHEN			TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-59：ADC SOC9 控制寄存器（ADCSOCCTL9）位段描述

位段	位段名	属性	复位值	描述
31:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC） 100：使能 SHA 和 SHB 101：使能 SHB 和 SHC 110：使能 SHA 和 SHC 111：使能 SHA、SHB 和 SHC
28:24	TRIGSEL	RW	0x0	选择 SOC9 触发源 本位段可以被 ADCINTSOCSEL1.SOC9 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA

位段	位段名	属性	复位值	描述
				10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC9 的 SH 负输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC9 的 SH 正输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC9 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC9 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-60：ADC SOC10 控制寄存器（ADCSOCCTL10）位段定义

ADCSOCCTL10 (ADC SOC10 Control Register) Offset: 0x60 Default: 0x00000000 Access: ADC -> ADCSOCCTL10.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-61：ADC SOC10 控制寄存器（ADCSOCCTL10）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL9.SHEN 设为 4、5 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC10 触发源 本位段可以被 ADCINTSOCSEL1.SOC10 覆盖。 当 ADCSOCCTL9.SHEN 设为 4、5 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC10 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL9.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC9 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC10 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL9.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC9 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC10 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC10 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-62：ADC SOC11 控制寄存器（ADCSOCCTL11）位段定义

ADCSOCCTL11 (ADC SOC11 Control Register) Offset: 0x64 Default: 0x00000000 Access: ADC -> ADCSOCCTL11.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-63：ADC SOC11 控制寄存器（ADCSOCCTL11）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL9.SHEN 设为 5、6 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC11 触发源 本位段可以被 ADCINTSOCSEL1.SOC11 覆盖。 当 ADCSOCCTL9.SHEN 设为 5、6 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC11 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL9.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC9 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC11 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL9.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC9 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC11 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC11 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-64：ADC SOC12 控制寄存器（ADCSOCCTL12）位段定义

ADCSOCCTL12 (ADC SOC12 Control Register) Offset: 0x68 Default: 0x00000000 Access: ADC -> ADCSOCCTL12.all							
31	30	29	28	27	26	25	24
SHEN			TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-65：ADC SOC12 控制寄存器（ADCSOCCTL12）位段描述

位段	位段名	属性	复位值	描述
31:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC） 100：使能 SHA 和 SHB 101：使能 SHB 和 SHC 110：使能 SHA 和 SHC 111：使能 SHA、SHB 和 SHC
28:24	TRIGSEL	RW	0x0	选择 SOC12 触发源 本位段可以被 ADCINTSOCSEL1.SOC12 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA

位段	位段名	属性	复位值	描述
				10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC12 的 SH 负输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC12 的 SH 正输入端通道选择 当 SHEN=1，4，6 或 7，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 SHEN=5，本位段不使用。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC12 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC12 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-66：ADC SOC13 控制寄存器（ADCSOCCTL13）位段定义

ADCSOCCTL13 (ADC SOC13 Control Register) Offset: 0x6C Default: 0x00000000 Access: ADC -> ADCSOCCTL13.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-67：ADC SOC13 控制寄存器（ADCSOCCTL13）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL12.SHEN 设为 4、5 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC13 触发源 本位段可以被 ADCINTSOCSEL1.SOC13 覆盖。 当 ADCSOCCTL12.SHEN 设为 4、5 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC13 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL12.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC12 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC13 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL12.SHEN=4、5 或者 7，采样保持器选择 SHB（SOC12 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC13 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC13 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-68：ADC SOC14 控制寄存器（ADCSOCCTL14）位段定义

ADCSOCCTL14 (ADC SOC14 Control Register) Offset: 0x70 Default: 0x00000000 Access: ADC -> ADCSOCCTL14.all							
31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-69：ADC SOC14 控制寄存器（ADCSOCCTL14）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 当 ADCSOCCTL12.SHEN 设为 5、6 或者 7 时，本位段不使用。 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC14 触发源 本位段可以被 ADCINTSOCSEL1.SOC14 覆盖。 当 ADCSOCCTL12.SHEN 设为 5、6 或者 7 时，本位段不使用。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB

位段	位段名	属性	复位值	描述
				10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC 10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC14 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL12.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC12 的）。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC14 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 当 ADCSOCCTL12.SHEN=5、6 或者 7，采样保持器选择 SHC（SOC12 的）。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC14 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC14 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-70：ADC SOC15 控制寄存器（ADCSOCCTL15）位段定义

ADCSOCCTL15 (ADC SOC15 Control Register) Offset: 0x74 Default: 0x00000000
Access: ADC -> ADCSOCCTL15.all

31	30	29	28	27	26	25	24
RESERVED	SHEN		TRIGSEL				
23	22	21	20	19	18	17	16
CHSELN			CHSELP			AVGCNT	
15	14	13	12	11	10	9	8
AVGCNT	CONVCNT						
7	6	5	4	3	2	1	0
SAMPCNT							

表 12-71：ADC SOC15 控制寄存器（ADCSOCCTL15）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RO	0x0	保留
30:29	SHEN	RW	0x0	采样保持器（SH）使能 000：关闭所有的采样保持器 001：使能采样保持器 A（SHA） 010：使能采样保持器 B（SHB） 011：使能采样保持器 C（SHC）
28:24	TRIGSEL	RW	0x0	选择 SOC15 触发源 本位段可以被 ADCINTSOCSEL1.SOC15 覆盖。 00000：ADCTRIG0 – 软件触发 00001：ADCTRIG1 – CPU Timer 0 00010：ADCTRIG2 – CPU Timer 1 00011：ADCTRIG3 – CPU Timer 2 00100：ADCTRIG4 – 外部 GPIO 触发（EXTSOC） 00101：ADCTRIG5 – PWM0SOCA 00110：ADCTRIG6 – PWM0SOCB 00111：ADCTRIG7 – PWM0SOCC 01000：ADCTRIG8 – PWM1SOCA 01001：ADCTRIG9 – PWM1SOCB 01010：ADCTRIG10 – PWM1SOCC 01011：ADCTRIG11 – PWM2SOCA 01100：ADCTRIG12 – PWM2SOCB 01101：ADCTRIG13 – PWM2SOCC 01110：ADCTRIG14 – PWM3SOCA 01111：ADCTRIG15 – PWM3SOCB 10000：ADCTRIG16 – PWM3SOCC 10001：ADCTRIG17 – PWM4SOCA 10010：ADCTRIG18 – PWM4SOCB 10011：ADCTRIG19 – PWM4SOCC

位段	位段名	属性	复位值	描述
				10100：ADCTRIG20 – PWM5SOCA 10101：ADCTRIG21 – PWM5SOCB 10110：ADCTRIG22 – PWM5SOCC 10111：ADCTRIG23 – PWM6SOCA 11000：ADCTRIG24 – PWM6SOCB 11001：ADCTRIG25 – PWM6SOCC 11010：ADCTRIG26 – PWM7SOCA 11011：ADCTRIG27 – PWM7SOCB 11100：ADCTRIG28 – PWM7SOCC 其他：无效选项
23:21	CHSELN	RW	0x0	SOC15 的 SH 负输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 具体的通道选择请参见表 12-4。
20:18	CHSELP	RW	0x0	SOC15 的 SH 正输入端通道选择 当 SHEN=1，采样保持器选择 SHA。 当 SHEN=2，采样保持器选择 SHB。 当 SHEN=3，采样保持器选择 SHC。 具体的通道选择请参见表 12-4。
17:15	AVGCNT	RW	0x0	选择 ADC 转换结果的平均次数 000：不平均，每个 SOC 只触发一次 ADC 转换 001：每个 SOC 触发 2 次 ADC 转换，再做平均 010：每个 SOC 触发 4 次 ADC 转换，再做平均 011：每个 SOC 触发 8 次 ADC 转换，再做平均 100：每个 SOC 触发 16 次 ADC 转换，再做平均 101：每个 SOC 触发 32 次 ADC 转换，再做平均 110：每个 SOC 触发 64 次 ADC 转换，再做平均 111：每个 SOC 触发 128 次 ADC 转换，再做平均
14:8	CONVCNT	RW	0x0	SOC15 转换窗口大小为（CONVCNT+1）个 ADC 时钟周期
7:0	SAMPCNT	RW	0x0	SOC15 采样窗口大小为（SAMPCNT+1）个 ADC 时钟周期

表 12-72：ADC 失调校准寄存器 0（ADCOFFSET0）位段定义

ADCOFFSET0 (ADC Offset Trim Register 0) Offset: 0x78 Default: 0x00000000 Access: ADC -> ADCOFFSET0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-73：ADC 失调校准寄存器 0（ADCOFFSET0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT0 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-74：ADC 失调校准寄存器 1（ADCOFFSET1）位段定义

ADCOFFSET1 (ADC Offset Trim Register 1) Offset: 0x7C Default: 0x00000000 Access: ADC -> ADCOFFSET1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-75：ADC 失调校准寄存器 1（ADCOFFSET1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT1 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-76：ADC 失调校准寄存器 2（ADCOFFSET2）位段定义

ADCOFFSET2 (ADC Offset Trim Register 2) Offset: 0x80 Default: 0x00000000 Access: ADC -> ADCOFFSET2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-77：ADC 失调校准寄存器 2（ADCOFFSET2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT2 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-78：ADC 失调校准寄存器 3（ADCOFFSET3）位段定义

ADCOFFSET3 (ADC Offset Trim Register 3) Offset: 0x84 Default: 0x00000000 Access: ADC -> ADCOFFSET3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-79：ADC 失调校准寄存器 3（ADCOFFSET3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT3 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-80：ADC 失调校准寄存器 4（ADCOFFSET4）位段定义

ADCOFFSET4 (ADC Offset Trim Register 4) Offset: 0x88 Default: 0x00000000 Access: ADC -> ADCOFFSET4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-81：ADC 失调校准寄存器 4（ADCOFFSET4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT4 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-82：ADC 失调校准寄存器 5（ADCOFFSET5）位段定义

ADCOFFSET5 (ADC Offset Trim Register 5) Offset: 0x8C Default: 0x00000000 Access: ADC -> ADCOFFSET5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-83：ADC 失调校准寄存器 5（ADCOFFSET5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT5 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-84：ADC 失调校准寄存器 6（ADCOFFSET6）位段定义

ADCOFFSET6 (ADC Offset Trim Register 6) Offset: 0x90 Default: 0x00000000 Access: ADC -> ADCOFFSET6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-85：ADC 失调校准寄存器 6（ADCOFFSET6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT6 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-86：ADC 失调校准寄存器 7（ADCOFFSET7）位段定义

ADCOFFSET7 (ADC Offset Trim Register 7) Offset: 0x94 Default: 0x00000000 Access: ADC -> ADCOFFSET7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-87：ADC 失调校准寄存器 7（ADCOFFSET7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT7 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-88：ADC 失调校准寄存器 8（ADCOFFSET8）位段定义

ADCOFFSET8 (ADC Offset Trim Register 8) Offset: 0x98 Default: 0x00000000 Access: ADC -> ADCOFFSET8.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-89：ADC 失调校准寄存器 8（ADCOFFSET8）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT8 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-90：ADC 失调校准寄存器 9（ADCOFFSET9）位段定义

ADCOFFSET9 (ADC Offset Trim Register 9) Offset: 0x9C Default: 0x00000000 Access: ADC -> ADCOFFSET9.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-91：ADC 失调校准寄存器 9（ADCOFFSET9）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT9 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-92：ADC 失调校准寄存器 10（ADCOFFSET10）位段定义

ADCOFFSET10 (ADC Offset Trim Register 10) Offset: 0xA0 Default: 0x00000000 Access: ADC -> ADCOFFSET10.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-93：ADC 失调校准寄存器 10（ADCOFFSET10）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT10 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-94：ADC 失调校准寄存器 11（ADCOFFSET11）位段定义

ADCOFFSET11 (ADC Offset Trim Register 11) Offset: 0xA4 Default: 0x00000000 Access: ADC -> ADCOFFSET11.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

12-95：ADC 失调校准寄存器 11（ADCOFFSET11）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT11 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-96：ADC 失调校准寄存器 12（ADCOFFSET12）位段定义

ADCOFFSET12 (ADC Offset Trim Register 12) Offset: 0xA8 Default: 0x00000000 Access: ADC -> ADCOFFSET12.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-97：ADC 失调校准寄存器 12（ADCOFFSET12）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT12 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-98：ADC 失调校准寄存器 13（ADCOFFSET13）位段定义

ADCOFFSET13 (ADC Offset Trim Register 13) Offset: 0xAC Default: 0x00000000 Access: ADC -> ADCOFFSET13.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-99：ADC 失调校准寄存器 13（ADCOFFSET13）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT13 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-100：ADC 失调校准寄存器 14（ADCOFFSET14）位段定义

ADCOFFSET14 (ADC Offset Trim Register 14) Offset: 0xB0 Default: 0x00000000 Access: ADC -> ADCOFFSET14.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-101：ADC 失调校准寄存器 14（ADCOFFSET14）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT14 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-102：ADC 失调校准寄存器 15（ADCOFFSET15）位段定义

ADCOFFSET15 (ADC Offset Trim Register 15) Offset: 0xB4 Default: 0x00000000 Access: ADC -> ADCOFFSET15.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-103：ADC 失调校准寄存器 15（ADCOFFSET15）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	ADCRESULT15 的失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-104：ADC 增益校准寄存器 0（ADCGAIN0）位段定义

ADCGAIN0 (ADC Gain Trim Register 0) Offset: 0xB8 Default: 0x00008000 Access: ADC -> ADCGAIN0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-105：ADC 增益校准寄存器 0（ADCGAIN0）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT0 的增益校准=（VAL/32768）

表 12-106：ADC 增益校准寄存器 1（ADCGAIN1）位段定义

ADCGAIN1 (ADC Gain Trim Register 1) Offset: 0xBC Default: 0x00008000 Access: ADC -> ADCGAIN1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-107：ADC 增益校准寄存器 1（ADCGAIN1）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT1 的增益校准=（VAL/32768）

表 12-108：ADC 增益校准寄存器 2（ADCGAIN2）位段定义

ADCGAIN2 (ADC Gain Trim Register 2) Offset: 0xC0 Default: 0x00008000 Access: ADC -> ADCGAIN2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-109：ADC 增益校准寄存器 2（ADCGAIN2）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT2 的增益校准=（VAL/32768）

表 12-110：ADC 增益校准寄存器 3（ADCGAIN3）位段定义

ADCGAIN3 (ADC Gain Trim Register 3) Offset: 0xC4 Default: 0x00008000 Access: ADC -> ADCGAIN3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-111：ADC 增益校准寄存器 3（ADCGAIN3）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT3 的增益校准=（VAL/32768）

表 12-112：ADC 增益校准寄存器 4（ADCGAIN4）位段定义

ADCGAIN4 (ADC Gain Trim Register 4) Offset: 0xC8 Default: 0x00008000 Access: ADC -> ADCGAIN4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-113：ADC 增益校准寄存器 4（ADCGAIN4）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT4 的增益校准=（VAL/32768）

表 12-114：ADC 增益校准寄存器 5（ADCGAIN5）位段定义

ADCGAIN5 (ADC Gain Trim Register 5) Offset: 0xCC Default: 0x00008000 Access: ADC -> ADCGAIN5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-115：ADC 增益校准寄存器 5（ADCGAIN5）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT5 的增益校准=（VAL/32768）

表 12-116：ADC 增益校准寄存器 6（ADCGAIN6）位段定义

ADCGAIN6 (ADC Gain Trim Register 6) Offset: 0xD0 Default: 0x00008000 Access: ADC -> ADCGAIN6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-117：ADC 增益校准寄存器 6（ADCGAIN6）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT6 的增益校准=（VAL/32768）

表 12-118：ADC 增益校准寄存器 7（ADCGAIN7）位段定义

ADCGAIN7 (ADC Gain Trim Register 7) Offset: 0xD4 Default: 0x00008000 Access: ADC -> ADCGAIN7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-119：ADC 增益校准寄存器 7（ADCGAIN7）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT7 的增益校准=（VAL/32768）

表 12-120：ADC 增益校准寄存器 8（ADCGAIN8）位段定义

ADCGAIN8 (ADC Gain Trim Register 8) Offset: 0xD8 Default: 0x00008000 Access: ADC -> ADCGAIN8.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-121：ADC 增益校准寄存器 8（ADCGAIN8）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT8 的增益校准=（VAL/32768）

表 12-122：ADC 增益校准寄存器 9（ADCGAIN9）位段定义

ADCGAIN9 (ADC Gain Trim Register 9) Offset: 0xDC Default: 0x00008000 Access: ADC -> ADCGAIN9.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-123：ADC 增益校准寄存器 9（ADCGAIN9）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT9 的增益校准=（VAL/32768）

表 12-124：ADC 增益校准寄存器 10（ADCGAIN10）位段定义

ADCGAIN10 (ADC Gain Trim Register 10) Offset: 0xE0 Default: 0x00008000 Access: ADC -> ADCGAIN10.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-125：ADC 增益校准寄存器 10（ADCGAIN10）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT10 的增益校准=（VAL/32768）

表 12-126：ADC 增益校准寄存器 11（ADCGAIN11）位段定义

ADCGAIN11 (ADC Gain Trim Register 11) Offset: 0xE4 Default: 0x00008000 Access: ADC -> ADCGAIN11.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-127：ADC 增益校准寄存器 11（ADCGAIN11）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT11 的增益校准=（VAL/32768）

表 12-128：ADC 增益校准寄存器 12（ADCGAIN12）位段定义

ADCGAIN12 (ADC Gain Trim Register 12) Offset: 0xE8 Default: 0x00008000 Access: ADC -> ADCGAIN12.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-129：ADC 增益校准寄存器 12（ADCGAIN12）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT12 的增益校准=（VAL/32768）

表 12-130：ADC 增益校准寄存器 13（ADCGAIN13）位段定义

ADCGAIN13 (ADC Gain Trim Register 13) Offset: 0xEC Default: 0x00008000 Access: ADC -> ADCGAIN13.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-131：ADC 增益校准寄存器 13（ADCGAIN13）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT13 的增益校准=（VAL/32768）

表 12-132：ADC 增益校准寄存器 14（ADCGAIN14）位段定义

ADCGAIN14 (ADC Gain Trim Register 14) Offset: 0xF0 Default: 0x00008000 Access: ADC -> ADCGAIN14.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-133：ADC 增益校准寄存器 14（ADCGAIN14）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT14 的增益校准=（VAL/32768）

表 12-134：ADC 增益校准寄存器 15（ADCGAIN15）位段定义

ADCGAIN15 (ADC Gain Trim Register 15) Offset: 0xF4 Default: 0x00008000 Access: ADC -> ADCGAIN15.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-135：ADC 增益校准寄存器 15（ADCGAIN15）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	ADCRESULT15 的增益校准=（VAL/32768）

表 12-136：ADC SHA 失调校准寄存器（ADCOFFSETA）位段定义

ADCOFFSETA (ADC SHA Offset Trim Register) Offset: 0xF8 Default: 0x00000000 Access: ADC -> ADCOFFSETA.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-137：ADC SHA 失调校准寄存器（ADCOFFSETA）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	采样保持器 A（SHA）失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-138：ADC SHB 失调校准寄存器（ADCOFFSETB）位段定义

ADCOFFSETB (ADC SHB Offset Trim Register) Offset: 0xFC Default: 0x00000000 Access: ADC -> ADCOFFSETB.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-139：ADC SHB 失调校准寄存器（ADCOFFSETB）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	采样保持器 B（SHB）失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-140：ADC SHC 失调校准寄存器（ADCOFFSETC）位段定义

ADCOFFSETC (ADC SHC Offset Trim Register) Offset: 0x100 Default: 0x00000000 Access: ADC -> ADCOFFSETC.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-141：ADC SHC 失调校准寄存器（ADCOFFSETC）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	采样保持器 C（SHC）失调校准（有符号数），数值范围应当为[-8192, 8191]

表 12-142：ADC SHA 增益校准寄存器（ADCGAINA）位段定义

ADCGAINA (ADC SHA Gain Trim Register) Offset: 0x104 Default: 0x00008000 Access: ADC -> ADCGAINA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-143：ADC SHA 增益校准寄存器（ADCGAINA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	SHA 的增益校准=（VAL/32768）

表 12-144：ADC SHB 增益校准寄存器（ADCGAINB）位段定义

ADCGAINB (ADC SHB Gain Trim Register) Offset: 0x108 Default: 0x00008000 Access: ADC -> ADCGAINB.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-145：ADC SHB 增益校准寄存器（ADCGAINB）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	SHB 的增益校准=（VAL/32768）

表 12-146：ADC SHC 增益校准寄存器（ADCGAINC）位段定义

ADCGAINC (ADC SHC Gain Trim Register) Offset: 0x10C Default: 0x00008000 Access: ADC -> ADCGAINC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-147：增益校准寄存器（ADCGAINC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8000	SHC 的增益校准=（VAL/32768）

表 12-148：ADC 状态寄存器（ADCSTS）位段定义

ADCSTS (ADC Status Register) Offset: 0x110 Default: 0x00000000 Access: ADC -> ADCSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					CODECRDY	CODEBRDY	CODEARDY

表 12-149：ADC 状态寄存器（ADCSTS）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	CODECRDY	RO	0x0	ADCRAWCODEC 就绪标志 0：ADCRAWCODEC 未就绪 1：ADCRAWCODEC 就绪
1	CODEBRDY	RO	0x0	ADCRAWCODEB 就绪标志 0：ADCRAWCODEB 未就绪 1：ADCRAWCODEB 就绪
0	CODEARDY	RO	0x0	ADCRAWCODEA 就绪标志 0：ADCRAWCODEA 未就绪 1：ADCRAWCODEA 就绪

表 12-150：ADC 状态清除寄存器（ADCSTSCLR）位段定义

ADCSTSCLR (ADC Status Clear Register) Offset: 0x114 Default: 0x00000000 Access: ADC -> ADCSTSCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					CODECRDY	CODEBRDY	CODEARDY

表 12-151：ADC 状态清除寄存器（ADCSTSCLR）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	CODECRDY	W1C	0x0	清除 ADCRAWCODEC 就绪标志 0：写 0 无效，总是读回 0 1：写 1 清除标志。本位段自动清零。
1	CODEBRDY	W1C	0x0	清除 ADCRAWCODEB 就绪标志 0：写 0 无效，总是读回 0 1：写 1 清除标志。本位段自动清零。
0	CODEARDY	W1C	0x0	清除 ADCRAWCODEA 就绪标志 0：写 0 无效，总是读回 0 1：写 1 清除标志。本位段自动清零。

表 12-152：ADC 控制寄存器（ADCCTL）位段定义

ADCCTL (ADC Control Register)　Offset: 0x118　Default: 0x00000008
Access: ADC -> ADCCTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
TIECN		TIECP		TIEBN		TIEBP	
7	6	5	4	3	2	1	0
TIEAN		TIEAP		SYNCEDGE	SYNCEN	RST	EN

表 12-153：ADC 控制寄存器（ADCCTL）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:14	TIECN	RW	0x0	SHC 负输入端强制控制 00：关闭 01：关闭 10：强制 SHC 负输入端接低电平 11：强制 SHC 负输入端接高电平
13:12	TIECP	RW	0x0	SHC 正输入端强制控制 00：关闭 01：关闭 10：强制 SHC 正输入端接低电平 11：强制 SHC 正输入端接高电平

位段	位段名	属性	复位值	描述
11:10	TIEBN	RW	0x0	SHB 负输入端强制控制 00：关闭 01：关闭 10：强制 SHB 负输入端接低电平 11：强制 SHB 负输入端接高电平
9:8	TIEBP	RW	0x0	SHB 正输入端强制控制 00：关闭 01：关闭 10：强制 SHB 正输入端接低电平 11：强制 SHB 正输入端接高电平
7:6	TIEAN	RW	0x0	SHA 负输入端强制控制 00：关闭 01：关闭 10：强制 SHA 负输入端接低电平 11：强制 SHA 负输入端接高电平
5:4	TIEAP	RW	0x0	SHA 正输入端强制控制 00：关闭 01：关闭 10：强制 SHA 正输入端接低电平 11：强制 SHA 正输入端接高电平
3	SYNCEDGE	RW	0x1	控制同步的采样/转换时钟边沿 0：下降沿 1：上升沿
2	SYNCEN	RW	0x0	使能采样/转换使用本地时钟控制同步 0：关闭 1：使能
1	RST	W1C	0x0	ADC 数字逻辑复位 0：写 0 无效，总是读回 0 1：写 1 复位 ADC 数字部分。本位段自动清零。
0	EN	RW	0x0	ADC 使能 0：关闭 ADC 1：使能 ADC

表 12-154：ADC 带隙基准控制寄存器（ADCBGCTL）位段定义

ADCBGCTL (ADC Bandgap Control Register) Offset: 0x11C Default: 0x00000000 Access: ADC -> ADCBGCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							EN

表 12-155：ADC 带隙基准控制寄存器（ADCBGCTL）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	EN	RW	0x0	ADC 带隙基准（bandgap）使能 0：关闭 ADC bandgap 1：使能 ADC bandgap

表 12-156：ADC 基准电压控制寄存器（ADCREFCTL）位段定义

ADCREFCTL (ADC Reference Control Register) Offset: 0x120 Default: 0x00003D3C Access: ADC -> ADCREFCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED	VDDTRIM				VANATRIM		VDIGTRIM
7	6	5	4	3	2	1	0
VDIGTRIM	VREFTRIM					EXTREF	EN

表 12-157：ADC 基准电压控制寄存器（ADCREFCTL）位段描述

位段	位段名	属性	复位值	描述
31:15	RESERVED_31_15	RO	0x0	保留
14:11	VDDTRIM	RW	0x7	VDD 校准
10:9	VANATRIM	RW	0x2	VANA 校准
8:7	VDIGTRIM	RW	0x2	VDIG 校准
6:2	VREFTRIM	RW	0xF	VREF 校准
1	EXTREF	RW	0x0	使能外灌电压基准 0：关闭 1：使能
0	EN	RW	0x0	使能基准电压 0：关闭 1：使能

表 12-158：ADC SHA 转换原始结果寄存器（ADCRAWCODEA）位段定义

ADCRAWCODEA (ADC SHA Raw Code Register) Offset: 0x124 Default: 0x00000000 Access: ADC -> ADCRAWCODEA.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-159：ADC SHA 转换原始结果寄存器（ADCRAWCODEA）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	SHA 转换的原始结果（有符号数，范围为-8192 ~ 8191）

表 12-160：ADC SHB 转换原始结果寄存器（ADCRAWCODEB）位段定义

ADCRAWCODEB (ADC SHB Raw Code Register) Offset: 0x128 Default: 0x00000000 Access: ADC -> ADCRAWCODEB.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-161：ADC SHB 转换原始结果寄存器（ADCRAWCODEB）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	SHB 转换的原始结果（有符号数，范围为-8192 ~ 8191）

表 12-162：ADC SHC 转换原始结果寄存器（ADCRAWCODEC）位段定义

ADCRAWCODEC (ADC SHC Raw Code Register)　Offset: 0x12C　Default: 0x00000000
Access: ADC -> ADCRAWCODEC.all

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

表 12-163：ADC SHC 转换原始结果寄存器（ADCRAWCODEC）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	SHC 转换的原始结果（有符号数，范围为-8192 ~ 8191）

表 12-164：ADC 转换结果寄存器 0（ADCRESULT0）位段定义

ADCRESULT0 (ADC Result Register 0)　Offset: 0x130　Default: 0x00000000
Access: ADC -> ADCRESULT0.all

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

表 12-165：ADC 转换结果寄存器 0（ADCRESULT0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL0 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL0.SHEN 使能）下，本寄存器为 ADCSOCCTL0 配置的采样保持器 SHA 的转换结果。

表 12-166：ADC 转换结果寄存器 1（ADCRESULT1）位段定义

ADCRESULT1 (ADC Result Register 1) Offset: 0x134 Default: 0x00000000 Access: ADC -> ADCRESULT1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-167：ADC 转换结果寄存器 1（ADCRESULT1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL1 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL0.SHEN 使能）下，本寄存器为 ADCSOCCTL1 配置的采样保持器 SHB 的转换结果。

表 12-168：ADC 转换结果寄存器 2（ADCRESULT2）位段定义

ADCRESULT2 (ADC Result Register 2) Offset: 0x138 Default: 0x00000000 Access: ADC -> ADCRESULT2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-169：ADC 转换结果寄存器 2（ADCRESULT2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL2 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL0.SHEN 使能）下，本寄存器为 ADCSOCCTL2 配置的采样保持器 SHC 的转换结果。

表 12-170：ADC 转换结果寄存器 3（ADCRESULT3）位段定义

ADCRESULT3 (ADC Result Register 3) Offset: 0x13C Default: 0x00000000 Access: ADC -> ADCRESULT3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-171：ADC 转换结果寄存器 3（ADCRESULT3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL3 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL3.SHEN 使能）下，本寄存器为 ADCSOCCTL3 配置的采样保持器 SHA 的转换结果。

表 12-172：ADC 转换结果寄存器 4（ADCRESULT4）位段定义

ADCRESULT4 (ADC Result Register 4) Offset: 0x140 Default: 0x00000000 Access: ADC -> ADCRESULT4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-173：ADC 转换结果寄存器 4（ADCRESULT4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL4 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL3.SHEN 使能）下，本寄存器为 ADCSOCCTL4 配置的采样保持器 SHB 的转换结果。

表 12-174：ADC 转换结果寄存器 5（ADCRESULT5）位段定义

ADCRESULT5 (ADC Result Register 5) Offset: 0x144 Default: 0x00000000 Access: ADC -> ADCRESULT5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-175：ADC 转换结果寄存器 5（ADCRESULT5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL5 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL3.SHEN 使能）下，本寄存器为 ADCSOCCTL5 配置的采样保持器 SHC 的转换结果。

表 12-176：ADC 转换结果寄存器 6（ADCRESULT6）位段定义

ADCRESULT6 (ADC Result Register 6) Offset: 0x148 Default: 0x00000000 Access: ADC -> ADCRESULT6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-177：ADC 转换结果寄存器 6（ADCRESULT6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL6 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL6.SHEN 使能）下，本寄存器为 ADCSOCCTL6 配置的采样保持器 SHA 的转换结果。

表 12-178：ADC 转换结果寄存器 7（ADCRESULT7）位段定义

ADCRESULT7 (ADC Result Register 7) Offset: 0x14C Default: 0x00000000 Access: ADC -> ADCRESULT7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-179：ADC 转换结果寄存器 7（ADCRESULT7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL7 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL6.SHEN 使能）下，本寄存器为 ADCSOCCTL7 配置的采样保持器 SHB 的转换结果。

表 12-180：ADC 转换结果寄存器 8（ADCRESULT8）位段定义

ADCRESULT8 (ADC Result Register 8) Offset: 0x150 Default: 0x00000000 Access: ADC -> ADCRESULT8.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-181：ADC 转换结果寄存器 8（ADCRESULT8）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL8 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL6.SHEN 使能）下，本寄存器为 ADCSOCCTL8 配置的采样保持器 SHC 的转换结果。

表 12-182：ADC 转换结果寄存器 9（ADCRESULT9）位段定义

ADCRESULT9 (ADC Result Register 9) Offset: 0x154 Default: 0x00000000 Access: ADC -> ADCRESULT9.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-183：ADC 转换结果寄存器 9（ADCRESULT9）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL9 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL9.SHEN 使能）下，本寄存器为 ADCSOCCTL9 配置的采样保持器 SHA 的转换结果。

表 12-184：ADC 转换结果寄存器 10（ADCRESULT10）位段定义

ADCRESULT10 (ADC Result Register 10) Offset: 0x158 Default: 0x00000000 Access: ADC -> ADCRESULT10.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-185：ADC 转换结果寄存器 10（ADCRESULT10）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL10 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL9.SHEN 使能）下，本寄存器为 ADCSOCCTL10 配置的采样保持器 SHB 的转换结果。

表 12-186：ADC 转换结果寄存器 11（ADCRESULT11）位段定义

ADCRESULT11 (ADC Result Register 11) Offset: 0x15C Default: 0x00000000 Access: ADC -> ADCRESULT11.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-187：ADC 转换结果寄存器 11（ADCRESULT11）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL11 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL9.SHEN 使能）下，本寄存器为 ADCSOCCTL11 配置的采样保持器 SHC 的转换结果。

表 12-188：ADC 转换结果寄存器 12（ADCRESULT12）位段定义

ADCRESULT12 (ADC Result Register 12) Offset: 0x160 Default: 0x00000000 Access: ADC -> ADCRESULT12.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-189：ADC 转换结果寄存器 12（ADCRESULT12）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL12 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL12.SHEN 使能）下，本寄存器为 ADCSOCCTL12 配置的采样保持器 SHA 的转换结果。

表 12-190：ADC 转换结果寄存器 13（ADCRESULT13）位段定义

ADCRESULT13 (ADC Result Register 13) Offset: 0x164 Default: 0x00000000 Access: ADC -> ADCRESULT13.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-191：ADC 转换结果寄存器 13（ADCRESULT13）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL13 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL12.SHEN 使能）下，本寄存器为 ADCSOCCTL13 配置的采样保持器 SHB 的转换结果。

表 12-192：ADC 转换结果寄存器 14（ADCRESULT14）位段定义

ADCRESULT14 (ADC Result Register 14) Offset: 0x168 Default: 0x00000000 Access: ADC -> ADCRESULT14.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-193：ADC 转换结果寄存器 14（ADCRESULT14）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL14 中使能的采样保持器（SH）的转换结果。 在同时采样模式（通过 ADCSOCCTL12.SHEN 使能）下，本寄存器为 ADCSOCCTL14 配置的采样保持器 SHC 的转换结果。

表 12-194：ADC 转换结果寄存器 15（ADCRESULT15）位段定义

ADCRESULT15 (ADC Result Register 15)　Offset: 0x16C　Default: 0x00000000
Access: ADC -> ADCRESULT15.all

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

表 12-195：ADC 转换结果寄存器 15（ADCRESULT15）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	14 位 ADC 转换结果（有符号数） 在顺序采样模式下，本寄存器为 ADCSOCCTL15 中使能的采样保持器（SH）的转换结果。 不支持同时采样模式。

表 12-196：ADC PPU0 输出结果寄存器（ADCPPURESULT0）位段定义

ADCPPURESULT0 (ADCPPU0 Comparison Result Register) Offset: 0x170 Default: 0x00000000 Access: ADC -> ADCPPURESULT0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-197：ADC PPU0 输出结果寄存器（ADCPPURESULT0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-198：ADC PPU1 输出结果寄存器（ADCPPURESULT1）位段定义

ADCPPURESULT1 (ADCPPU1 Comparison Result Register) Offset: 0x174 Default: 0x00000000 Access: ADC -> ADCPPURESULT1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-199：ADC PPU1 输出结果寄存器（ADCPPURESULT1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-200：ADC PPU2 输出结果寄存器（ADCPPURESULT2）位段定义

ADCPPURESULT2 (ADCPPU2 Comparison Result Register) Offset: 0x178 Default: 0x00000000 Access: ADC -> ADCPPURESULT2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-201：ADC PPU2 输出结果寄存器（ADCPPURESULT2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-202：ADC PPU3 输出结果寄存器（ADCPPURESULT3）位段定义

ADCPPURESULT3 (ADCPPU3 Comparison Result Register) Offset: 0x17C Default: 0x00000000 Access: ADC -> ADCPPURESULT3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-203：ADC PPU3 输出结果寄存器（ADCPPURESULT3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-204：ADC PPU4 输出结果寄存器（ADCPPURESULT4）位段定义

ADCPPURESULT4 (ADCPPU4 Comparison Result Register) Offset: 0x180 Default: 0x00000000 Access: ADC -> ADCPPURESULT4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-205：ADC PPU4 输出结果寄存器（ADCPPURESULT4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-206：ADC PPU5 输出结果寄存器（ADCPPURESULT5）位段定义

ADCPPURESULT5 (ADCPPU5 Comparison Result Register) Offset: 0x184 Default: 0x00000000 Access: ADC -> ADCPPURESULT5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-207：ADC PPU5 输出结果寄存器（ADCPPURESULT5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-208：ADCPPU6 输出结果寄存器（ADCPPURESULT6）位段定义

ADCPPURESULT6 (ADCPPU6 Comparison Result Register) Offset: 0x188 Default: 0x00000000 Access: ADC -> ADCPPURESULT6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-209：ADCPPU6 输出结果寄存器（ADCPPURESULT6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-210：ADCPPU7 输出结果寄存器（ADCPPURESULT7）位段定义

ADCPPURESULT7 (ADCPPU7 Comparison Result Register) Offset: 0x18C Default: 0x00000000 Access: ADC -> ADCPPURESULT7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-211：ADCPPU7 输出结果寄存器（ADCPPURESULT7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	输出结果（有符号数）

表 12-212：ADC PPU0 SOC 延迟捕获寄存器（ADCPPUSOCDLY0）位段定义

ADCPPUSOCDLY0 (ADCPPU0 SOC Delay Register) Offset: 0x190 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-213：ADC PPU0 SOC 延迟捕获寄存器（ADCPPUSOCDLY0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU0CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-214：ADC PPU1 SOC 延迟捕获寄存器（ADCPPUSOCDLY1）位段定义

ADCPPUSOCDLY1 (ADCPPU1 SOC Delay Register) Offset: 0x194 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-215：ADC PPU1 SOC 延迟捕获寄存器（ADCPPUSOCDLY1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU1CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-216：ADC PPU2 SOC 延迟捕获寄存器（ADCPPUSOCDLY2）位段定义

ADCPPUSOCDLY2 (ADCPPU2 SOC Delay Register) Offset: 0x198 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-217：ADC PPU2 SOC 延迟捕获寄存器（ADCPPUSOCDLY2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU2CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-218：ADC PPU3 SOC 延迟捕获寄存器（ADCPPUSOCDLY3）位段定义

ADCPPUSOCDLY3 (ADCPPU3 SOC Delay Register) Offset: 0x19C Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-219：ADC PPU3 SOC 延迟捕获寄存器（ADCPPUSOCDLY3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU3CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-220：ADC PPU4 SOC 延迟捕获寄存器（ADCPPUSOCDLY4）位段定义

ADCPPUSOCDLY4 (ADCPPU4 SOC Delay Register) Offset: 0x1A0 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-221：ADC PPU4 SOC 延迟捕获寄存器（ADCPPUSOCDLY4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU4CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-222：ADC PPU5 SOC 延迟捕获寄存器（ADCPPUSOCDLY5）位段定义

ADCPPUSOCDLY5 (ADCPPU5 SOC Delay Register) Offset: 0x1A4 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-223：ADC PPU5 SOC 延迟捕获寄存器（ADCPPUSOCDLY5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU5CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-224：ADC PPU6 SOC 延迟捕获寄存器（ADCPPUSOCDLY6）位段定义

ADCPPUSOCDLY6 (ADCPPU6 SOC Delay Register) Offset: 0x1A8 Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-225：ADC PPU6 SOC 延迟捕获寄存器（ADCPPUSOCDLY6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU6CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-226：ADC PPU7 SOC 延迟捕获寄存器（ADCPPUSOCDLY7）位段定义

ADCPPUSOCDLY7 (ADCPPU7 SOC Delay Register) Offset: 0x1AC Default: 0x00000000 Access: ADC -> ADCPPUSOCDLY7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-227：ADC PPU7 SOC 延迟捕获寄存器（ADCPPUSOCDLY7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	ADCPPU7CTL.SOCSEL 位段选择的 SOC 从触发到执行的延迟

表 12-228：ADC PPU0 中断标志寄存器（ADCPPUIF0）位段定义

ADCPPUIF0 (ADCPPU0 Interrupt Flag Register) Offset: 0x1B0 Default: 0x00000000 Access: ADC -> ADCPPUIF0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-229：ADC PPU0 中断标志寄存器（ADCPPUIF0）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU0 中断标志 0：ADC PPU0 中断未发生 1：ADC PPU0 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU0 过零（zero-cross）状态 0：ADC PPU0 过零事件未发生 1：ADC PPU0 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU0 过高（too-high）检测状态 0：ADC PPU0 过高事件未发生 1：ADC PPU0 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU0 过低（too-low）检测状态 0：ADC PPU0 过低事件未发生 1：ADC PPU0 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-230：ADC PPU1 中断标志寄存器（ADCPPUIF1）位段定义

ADCPPUIF1 (ADCPPU1 Interrupt Flag Register) Offset: 0x1B4 Default: 0x00000000 Access: ADC -> ADCPPUIF1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-231：ADC PPU1 中断标志寄存器（ADCPPUIF1）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU1 中断标志 0：ADC PPU1 中断未发生 1：ADC PPU1 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU1 过零（zero-cross）状态 0：ADC PPU1 过零事件未发生 1：ADC PPU1 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU1 过高（too-high）检测状态 0：ADC PPU1 过高事件未发生 1：ADC PPU1 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU1 过低（too-low）检测状态 0：ADC PPU1 过低事件未发生 1：ADC PPU1 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-232：ADC PPU2 中断标志寄存器（ADCPPUIF2）位段定义

ADCPPUIF2 (ADCPPU2 Interrupt Flag Register) Offset: 0x1B8 Default: 0x00000000 Access: ADC -> ADCPPUIF2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-233：ADC PPU2 中断标志寄存器（ADCPPUIF2）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU2 中断标志 0：ADC PPU2 中断未发生 1：ADC PPU2 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU2 过零（zero-cross）状态 0：ADC PPU2 过零事件未发生 1：ADC PPU2 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU2 过高（too-high）检测状态 0：ADC PPU2 过高事件未发生 1：ADC PPU2 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU2 过低（too-low）检测状态 0：ADC PPU2 过低事件未发生 1：ADC PPU2 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-234：ADC PPU3 中断标志寄存器（ADCPPUIF3）位段定义

ADCPPUIF3 (ADCPPU3 Interrupt Flag Register) Offset: 0x1BC Default: 0x00000000 Access: ADC -> ADCPPUIF3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-235：ADC PPU3 中断标志寄存器（ADCPPUIF3）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU3 中断标志 0：ADC PPU3 中断未发生 1：ADC PPU3 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU3 过零（zero-cross）状态 0：ADC PPU3 过零事件未发生 1：ADC PPU3 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU3 过高（too-high）检测状态 0：ADC PPU3 过高事件未发生 1：ADC PPU3 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU3 过低（too-low）检测状态 0：ADC PPU3 过低事件未发生 1：ADC PPU3 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-236：ADC PPU4 中断标志寄存器（ADCPPUIF4）位段定义

ADCPPUIF4 (ADCPPU4 Interrupt Flag Register) Offset: 0x1C0 Default: 0x00000000 Access: ADC -> ADCPPUIF4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-237：ADC PPU4 中断标志寄存器（ADCPPUIF4）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU4 中断标志 0：ADC PPU4 中断未发生 1：ADC PPU4 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU4 过零（zero-cross）状态 0：ADC PPU4 过零事件未发生 1：ADC PPU4 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU4 过高（too-high）检测状态 0：ADC PPU4 过高事件未发生 1：ADC PPU4 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU4 过低（too-low）检测状态 0：ADC PPU4 过低事件未发生 1：ADC PPU4 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-238：ADC PPU5 中断标志寄存器（ADCPPUIF5）位段定义

ADCPPUIF5 (ADCPPU5 Interrupt Flag Register) Offset: 0x1C4 Default: 0x00000000 Access: ADC -> ADCPPUIF5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-239：ADC PPU5 中断标志寄存器（ADCPPUIF5）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU5 中断标志 0：ADC PPU5 中断未发生 1：ADC PPU5 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU5 过零（zero-cross）状态 0：ADC PPU5 过零事件未发生 1：ADC PPU5 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU5 过高（too-high）检测状态 0：ADC PPU5 过高事件未发生 1：ADC PPU5 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU5 过低（too-low）检测状态 0：ADC PPU5 过低事件未发生 1：ADC PPU5 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-240：ADC PPU6 中断标志寄存器（ADCPPUIF6）位段定义

ADCPPUIF6 (ADCPPU6 Interrupt Flag Register) Offset: 0x1C8 Default: 0x00000000 Access: ADC -> ADCPPUIF6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-241：ADC PPU6 中断标志寄存器（ADCPPUIF6）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU6 中断标志 0：ADC PPU6 中断未发生 1：ADC PPU6 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU6 过零（zero-cross）状态 0：ADC PPU6 过零事件未发生 1：ADC PPU6 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU6 过高（too-high）检测状态 0：ADC PPU6 过高事件未发生 1：ADC PPU6 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU6 过低（too-low）检测状态 0：ADC PPU6 过低事件未发生 1：ADC PPU6 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-242：ADC PPU7 中断标志寄存器（ADCPPUIF7）位段定义

ADCPPUIF7 (ADCPPU7 Interrupt Flag Register) Offset: 0x1CC Default: 0x00000000 Access: ADC -> ADCPPUIF7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-243：ADC PPU7 中断标志寄存器（ADCPPUIF7）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	RO	0x0	ADC PPU7 中断标志 0：ADC PPU7 中断未发生 1：ADC PPU7 中断已发生 在本位段清零前，不会有新的中断产生。
2	XZRO	RO	0x0	锁存的 ADC PPU7 过零（zero-cross）状态 0：ADC PPU7 过零事件未发生 1：ADC PPU7 过零事件已发生 在本位段清零前，新的事件不会产生中断。
1	TZHI	RO	0x0	锁存的 ADC PPU7 过高（too-high）检测状态 0：ADC PPU7 过高事件未发生 1：ADC PPU7 过高事件已发生 在本位段清零前，新的事件不会产生中断。
0	TZLO	RO	0x0	锁存的 ADC PPU7 过低（too-low）检测状态 0：ADC PPU7 过低事件未发生 1：ADC PPU7 过低事件已发生 在本位段清零前，新的事件不会产生中断。

表 12-244：ADC PPU0 中断清除寄存器（ADCPPUIC0）位段定义

ADCPPUIC0 (ADCPPU0 Interrupt Clear Register) Offset: 0x1D0 Default: 0x00000000 Access: ADC -> ADCPPUIC0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-245：ADC PPU0 中断清除寄存器（ADCPPUIC0）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU0 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU0IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU0 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU0IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU0 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU0IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU0 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU0IF.TZLO 本位段自动清零。

表 12-246：ADC PPU1 中断清除寄存器（ADCPPUIC1）位段定义

ADCPPUIC1 (ADCPPU1 Interrupt Clear Register) Offset: 0x1D4 Default: 0x00000000 Access: ADC -> ADCPPUIC1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-247：ADC PPU1 中断清除寄存器（ADCPPUIC1）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU1 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU1IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU1 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU1IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU1 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU1IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU1 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU1IF.TZLO 本位段自动清零。

表 12-248：ADC PPU2 中断清除寄存器（ADCPPUIC2）位段定义

ADCPPUIC2 (ADCPPU2 Interrupt Clear Register) Offset: 0x1D8 Default: 0x00000000 Access: ADC -> ADCPPUIC2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-249：ADC PPU2 中断清除寄存器（ADCPPUIC2）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU2 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU2IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU2 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU2IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU2 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU2IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU2 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU2IF.TZLO 本位段自动清零。

表 12-250：ADC PPU3 中断清除寄存器（ADCPPUIC3）位段定义

ADCPPUIC3 (ADCPPU3 Interrupt Clear Register) Offset: 0x1DC Default: 0x00000000 Access: ADC -> ADCPPUIC3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-251：ADC PPU3 中断清除寄存器（ADCPPUIC3）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU3 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU3IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU3 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU3IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU3 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU3IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU3 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU3IF.TZLO 本位段自动清零。

表 12-252：ADC PPU4 中断清除寄存器（ADCPPUIC4）位段定义

ADCPPUIC4 (ADCPPU4 Interrupt Clear Register) Offset: 0x1E0 Default: 0x00000000 Access: ADC -> ADCPPUIC4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-253：ADC PPU4 中断清除寄存器（ADCPPUIC4）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU4 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU4IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU4 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU4IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU4 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU4IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU4 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU4IF.TZLO 本位段自动清零。

表 12-254：ADC PPU5 中断清除寄存器（ADCPPUIC5）位段定义

ADCPPUIC5 (ADCPPU5 Interrupt Clear Register) Offset: 0x1E4 Default: 0x00000000 Access: ADC -> ADCPPUIC5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-255：ADC PPU5 中断清除寄存器（ADCPPUIC5）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADC PPU5 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADC PPU5 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADC PPU5 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADC PPU5 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZLO 本位段自动清零。

表 12-256：ADCPPU6 中断清除寄存器（ADCPPUIC6）位段定义

ADCPPUIC6 (ADCPPU6 Interrupt Clear Register) Offset: 0x1E8 Default: 0x00000000 Access: ADC -> ADCPPUIC6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-257：ADCPPU6 中断清除寄存器（ADCPPUIC6）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADCPPU6 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADCPPU6 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADCPPU6 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADCPPU6 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZLO 本位段自动清零。

表 12-258：ADCPPU7 中断清除寄存器（ADCPPUIC7）位段定义

ADCPPUIC7 (ADCPPU7 Interrupt Clear Register) Offset: 0x1EC Default: 0x00000000 Access: ADC -> ADCPPUIC7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				INT	XZRO	TZHI	TZLO

表 12-259：ADCPPU7 中断清除寄存器（ADCPPUIC7）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	INT	W1C	0x0	ADCPPU7 中断标志清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.INT 本位段自动清零。
2	XZRO	W1C	0x0	锁存的 ADCPPU7 过零（zero-cross）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.XZRO 本位段自动清零。
1	TZHI	W1C	0x0	锁存的 ADCPPU7 过高（too-high）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZHI 本位段自动清零。
0	TZLO	W1C	0x0	锁存的 ADCPPU7 过低（too-low）状态清除 0：写 0 无效，总是读回 0 1：写 1 清除 ADCPPU5IF.TZLO 本位段自动清零。

表 12-260：ADC PPU0 中断使能寄存器（ADCPPUIE0）位段定义

ADCPPUIE0 (ADCPPU0 Interrupt Enable Register) Offset: 0x1F0 Default: 0x00000000 Access: ADC -> ADCPPUIE0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-261：ADC PPU0 中断使能寄存器（ADCPPUIE0）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU0 过零事件作为 ADC PPU0 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU0 过高事件作为 ADC PPU0 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU0 过低事件作为 ADC PPU0 中断源 0：关闭 1：使能

表 12-262：ADC PPU1 中断使能寄存器（ADCPPUIE1）位段定义

ADCPPUIE1 (ADCPPU1 Interrupt Enable Register) Offset: 0x1F4 Default: 0x00000000 Access: ADC -> ADCPPUIE1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-263：ADC PPU1 中断使能寄存器（ADCPPUIE1）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU1 过零事件作为 ADC PPU1 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU1 过高事件作为 ADC PPU1 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU1 过低事件作为 ADC PPU1 中断源 0：关闭 1：使能

表 12-264：ADC PPU2 中断使能寄存器（ADCPPUIE2）位段定义

ADCPPUIE2 (ADCPPU2 Interrupt Enable Register) Offset: 0x1F8 Default: 0x00000000 Access: ADC -> ADCPPUIE2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-265：ADC PPU2 中断使能寄存器（ADCPPUIE2）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU2 过零事件作为 ADC PPU2 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU2 过高事件作为 ADC PPU2 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU2 过低事件作为 ADC PPU2 中断源 0：关闭 1：使能

表 12-266：ADC PPU3 中断使能寄存器（ADCPPUIE3）位段定义

ADCPPUIE3 (ADCPPU3 Interrupt Enable Register) Offset: 0x1FC Default: 0x00000000 Access: ADC -> ADCPPUIE3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-267：ADC PPU3 中断使能寄存器（ADCPPUIE3）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU3 过零事件作为 ADC PPU3 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU3 过高事件作为 ADC PPU3 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU3 过低事件作为 ADC PPU3 中断源 0：关闭 1：使能

表 12-268：ADC PPU4 中断使能寄存器（ADCPPUIE4）位段定义

ADCPPUIE4 (ADCPPU4 Interrupt Enable Register) Offset: 0x200 Default: 0x00000000 Access: ADC -> ADCPPUIE4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-269：ADC PPU4 中断使能寄存器（ADCPPUIE4）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU4 过零事件作为 ADC PPU4 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU4 过高事件作为 ADC PPU4 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU4 过低事件作为 ADC PPU4 中断源 0：关闭 1：使能

表 12-270：ADC PPU5 中断使能寄存器（ADCPPUIE5）位段定义

ADCPPUIE5 (ADCPPU5 Interrupt Enable Register) Offset: 0x204 Default: 0x00000000 Access: ADC -> ADCPPUIE5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-271：ADC PPU5 中断使能寄存器（ADCPPUIE5）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU5 过零事件作为 ADC PPU5 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU5 过高事件作为 ADC PPU5 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU5 过低事件作为 ADC PPU5 中断源 0：关闭 1：使能

表 12-272：ADC PPU6 中断使能寄存器（ADCPPUIE6）位段定义

ADCPPUIE6 (ADCPPU6 Interrupt Enable Register) Offset: 0x208 Default: 0x00000000 Access: ADC -> ADCPPUIE6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-273：ADC PPU6 中断使能寄存器（ADCPPUIE6）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU6 过零事件作为 ADC PPU6 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU6 过高事件作为 ADC PPU6 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU6 过低事件作为 ADC PPU6 中断源 0：关闭 1：使能

表 12-274：ADC PPU7 中断使能寄存器（ADCPPUIE7）位段定义

ADCPPUIE7 (ADCPPU7 Interrupt Enable Register) Offset: 0x20C Default: 0x00000000 Access: ADC -> ADCPPUIE7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-275：ADC PPU7 中断使能寄存器（ADCPPUIE7）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU7 过零事件作为 ADC PPU7 中断源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU7 过高事件作为 ADC PPU7 中断源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU7 过低事件作为 ADC PPU7 中断源 0：关闭 1：使能

表 12-276：ADC PPU0 封锁事件使能寄存器（ADCPPUTZE0）位段定义

ADCPPUTZE0 (ADCPPU0 Trip-Zone Event Enable Register) Offset: 0x210 Default: 0x00000000 Access: ADC -> ADCPPUTZE0.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-277：ADC PPU0 封锁事件使能寄存器（ADCPPUTZE0）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU0 过零事件作为 ADC PPU0 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU0 过高事件作为 ADC PPU0 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU0 过低事件作为 ADC PPU0 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-278：ADC PPU1 封锁事件使能寄存器（ADCPPUTZE1）位段定义

ADCPPUTZE1 (ADCPPU1 Trip-Zone Event Enable Register) Offset: 0x214 Default: 0x00000000 Access: ADC -> ADCPPUTZE1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-279：ADC PPU1 封锁事件使能寄存器（ADCPPUTZE1）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU1 过零事件作为 ADC PPU1 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU1 过高事件作为 ADC PPU1 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU1 过低事件作为 ADC PPU1 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-280：ADC PPU2 封锁事件使能寄存器（ADCPPUTZE2）位段定义

ADCPPUTZE2 (ADCPPU2 Trip-Zone Event Enable Register) Offset: 0x218 Default: 0x00000000 Access: ADC -> ADCPPUTZE2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-281：ADC PPU2 封锁事件使能寄存器（ADCPPUTZE2）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU2 过零事件作为 ADC PPU2 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU2 过高事件作为 ADC PPU2 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU2 过低事件作为 ADC PPU2 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-282：ADC PPU3 封锁事件使能寄存器（ADCPPUTZE3）位段定义

ADCPPUTZE3 (ADCPPU3 Trip-Zone Event Enable Register) Offset: 0x21C Default: 0x00000000 Access: ADC -> ADCPPUTZE3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-283：ADC PPU3 封锁事件使能寄存器（ADCPPUTZE3）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU3 过零事件作为 ADC PPU3 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU3 过高事件作为 ADC PPU3 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU3 过低事件作为 ADC PPU3 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-284：ADC PPU4 封锁事件使能寄存器（ADCPPUTZE4）位段定义

ADCPPUTZE4 (ADCPPU4 Trip-Zone Event Enable Register) Offset: 0x220 Default: 0x00000000 Access: ADC -> ADCPPUTZE4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-285：ADC PPU4 封锁事件使能寄存器（ADCPPUTZE4）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU4 过零事件作为 ADC PPU4 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU4 过高事件作为 ADC PPU4 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU4 过低事件作为 ADC PPU4 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-286：ADC PPU5 封锁事件使能寄存器（ADCPPUTZE5）位段定义

ADCPPUTZE5 (ADCPPU5 Trip-Zone Event Enable Register) Offset: 0x224 Default: 0x00000000 Access: ADC -> ADCPPUTZE5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-287：ADC PPU5 封锁事件使能寄存器（ADCPPUTZE5）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU5 过零事件作为 ADC PPU5 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU5 过高事件作为 ADC PPU5 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU5 过低事件作为 ADC PPU5 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-288：ADC PPU6 封锁事件使能寄存器（ADCPPUTZE6）位段定义

ADCPPUTZE6 (ADCPPU6 Trip-Zone Event Enable Register) Offset: 0x228 Default: 0x00000000 Access: ADC -> ADCPPUTZE6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-289：ADC PPU6 封锁事件使能寄存器（ADCPPUTZE6）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU6 过零事件作为 ADC PPU6 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU6 过高事件作为 ADC PPU6 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU6 过低事件作为 ADC PPU6 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-290：ADC PPU7 封锁事件使能寄存器（ADCPPUTZE7）位段定义

ADCPPUTZE7 (ADCPPU7 Trip-Zone Event Enable Register) Offset: 0x22C Default: 0x00000000 Access: ADC -> ADCPPUTZE7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					XZRO	TZHI	TZLO

表 12-291：ADC PPU7 封锁事件使能寄存器（ADCPPUTZE7）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	XZRO	RW	0x0	使能 ADC PPU7 过零事件作为 ADC PPU7 封锁（trip-zone）事件源 0：关闭 1：使能
1	TZHI	RW	0x0	使能 ADC PPU7 过高事件作为 ADC PPU7 封锁（trip-zone）事件源 0：关闭 1：使能
0	TZLO	RW	0x0	使能 ADC PPU7 过低事件作为 ADC PPU7 封锁（trip-zone）事件源 0：关闭 1：使能

表 12-292：ADC PPU0 控制寄存器（ADCPPUCTL0）位段定义

ADCPPUCTL0 (ADCPPU0 Control Register) Offset: 0x230 Default: 0x00000400
Access: ADC -> ADCPPUCTL0.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-293：ADC PPU0 控制寄存器（ADCPPUCTL0）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU0 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU0 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU0SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU0 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU0 周期性清除 1：使能 ADC PPU0 周期性清除
0	EN	RW	0x0	ADC PPU0 使能 0：关闭 ADC PPU0 1：使能 ADC PPU0

表 12-294：ADC PPU1 控制寄存器（ADCPPUCTL1）位段定义

ADCPPUCTL1 (ADCPPU1 Control Register) Offset: 0x234 Default: 0x00000400 Access: ADC -> ADCPPUCTL1.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-295：ADC PPU1 控制寄存器（ADCPPUCTL1）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU1 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU1 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU1SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU1 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU1 周期性清除 1：使能 ADC PPU1 周期性清除
0	EN	RW	0x0	ADC PPU1 使能 0：关闭 ADC PPU1 1：使能 ADC PPU1

表 12-296：ADC PPU2 控制寄存器（ADCPPUCTL2）位段定义

ADCPPUCTL2 (ADCPPU2 Control Register) Offset: 0x238 Default: 0x00000400 Access: ADC -> ADCPPUCTL2.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-297：ADC PPU2 控制寄存器（ADCPPUCTL2）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU2 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU2 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU2SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU2 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU2 周期性清除 1：使能 ADC PPU2 周期性清除
0	EN	RW	0x0	ADC PPU2 使能 0：关闭 ADC PPU2 1：使能 ADC PPU2

表 12-298：ADC PPU3 控制寄存器（ADCPPUCTL3）位段定义

ADCPPUCTL3 (ADCPPU3 Control Register) Offset: 0x23C Default: 0x00000400 Access: ADC -> ADCPPUCTL3.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-299：ADC PPU3 控制寄存器（ADCPPUCTL3）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU3 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU3 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU3SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU3 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU3 周期性清除 1：使能 ADC PPU3 周期性清除
0	EN	RW	0x0	ADC PPU3 使能 0：关闭 ADC PPU3 1：使能 ADC PPU3

表 12-300：ADC PPU4 控制寄存器（ADCPPUCTL4）位段定义

ADCPPUCTL4 (ADCPPU4 Control Register) Offset: 0x240 Default: 0x00000400 Access: ADC -> ADCPPUCTL4.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-301：ADC PPU4 控制寄存器（ADCPPUCTL4）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU4 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU4 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU4SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU4 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU4 周期性清除 1：使能 ADC PPU4 周期性清除
0	EN	RW	0x0	ADC PPU4 使能 0：关闭 ADC PPU4 1：使能 ADC PPU4

表 12-302：ADC PPU5 控制寄存器（ADCPPUCTL5）位段定义

ADCPPUCTL5 (ADCPPU5 Control Register) Offset: 0x244 Default: 0x00000400 Access: ADC -> ADCPPUCTL5.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-303：ADC PPU5 控制寄存器（ADCPPUCTL5）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU5 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU5 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU5SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU5 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU5 周期性清除 1：使能 ADC PPU5 周期性清除
0	EN	RW	0x0	ADC PPU5 使能 0：关闭 ADC PPU5 1：使能 ADC PPU5

表 12-304：ADC PPU6 控制寄存器（ADCPPUCTL6）位段定义

ADCPPUCTL6 (ADCPPU6 Control Register) Offset: 0x248 Default: 0x00000400 Access: ADC -> ADCPPUCTL6.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-305：ADC PPU6 控制寄存器（ADCPPUCTL6）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU6 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU6 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU6SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU6 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU6 周期性清除 1：使能 ADC PPU6 周期性清除
0	EN	RW	0x0	ADC PPU6 使能 0：关闭 ADC PPU6 1：使能 ADC PPU6

表 12-306：ADC PPU7 控制寄存器（ADCPPUCTL7）位段定义

ADCPPUCTL7 (ADCPPU7 Control Register) Offset: 0x24C Default: 0x00000400 Access: ADC -> ADCPPUCTL7.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED					POL	DATASEL	
7	6	5	4	3	2	1	0
DATASEL		SOCSEL				CBCEN	EN

表 12-307：ADC PPU7 控制寄存器（ADCPPUCTL7）位段描述

位段	位段名	属性	复位值	描述
31:11	RESERVED_31_11	RO	0x0	保留
10	POL	RW	0x1	输出结果极性选择 0：ADC PPU7 输出结果是基准值减去选定的 ADC 转换结果 1：ADC PPU7 输出结果是 ADC 转换结果减去基准值
9:6	DATASEL	RW	0x0	选择要处理的 ADC 转换结果
5:2	SOCSEL	RW	0x0	选择产生 ADCPPU7SOCDLY 的 SOC 信号
1	CBCEN	RW	0x0	使能 ADC PPU7 周期性（cycle-by-cycle）清除 0：关闭 ADC PPU7 周期性清除 1：使能 ADC PPU7 周期性清除
0	EN	RW	0x0	ADC PPU7 使能 0：关闭 ADC PPU7 1：使能 ADC PPU7

表 12-308：ADC PPU0 基准寄存器（ADCPPUREF0）位段定义

ADCPPUREF0 (ADCPPU0 Reference Register) Offset: 0x250 Default: 0x00000000 Access: ADC -> ADCPPUREF0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-309：ADC PPU0 基准寄存器（ADCPPUREF0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-310：ADC PPU1 基准寄存器（ADCPPUREF1）位段定义

ADCPPUREF1 (ADCPPU1 Reference Register) Offset: 0x254 Default: 0x00000000 Access: ADC -> ADCPPUREF1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-311：ADC PPU1 基准寄存器（ADCPPUREF1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-312：ADC PPU2 基准寄存器（ADCPPUREF2）位段定义

ADCPPUREF2 (ADCPPU2 Reference Register) Offset: 0x258 Default: 0x00000000 Access: ADC -> ADCPPUREF2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-313：ADC PPU2 基准寄存器（ADCPPUREF2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-314：ADC PPU3 基准寄存器（ADCPPUREF3）位段定义

ADCPPUREF3 (ADCPPU3 Reference Register) Offset: 0x25C Default: 0x00000000 Access: ADC -> ADCPPUREF3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-315：ADC PPU3 基准寄存器（ADCPPUREF3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-316：ADC PPU4 基准寄存器（ADCPPUREF4）位段定义

ADCPPUREF4 (ADCPPU4 Reference Register) Offset: 0x260 Default: 0x00000000 Access: ADC -> ADCPPUREF4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-317：ADC PPU4 基准寄存器（ADCPPUREF4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-318：ADC PPU5 基准寄存器（ADCPPUREF5）位段定义

ADCPPUREF5 (ADCPPU5 Reference Register) Offset: 0x264 Default: 0x00000000 Access: ADC -> ADCPPUREF5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-319：ADC PPU5 基准寄存器（ADCPPUREF5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-320：ADC PPU6 基准寄存器（ADCPPUREF6）位段定义

ADCPPUREF6 (ADCPPU6 Reference Register) Offset: 0x268 Default: 0x00000000 Access: ADC -> ADCPPUREF6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-321：ADC PPU6 基准寄存器（ADCPPUREF6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-322：ADC PPU7 基准寄存器（ADCPPUREF7）位段定义

ADCPPUREF7 (ADCPPU7 Reference Register) Offset: 0x26C Default: 0x00000000 Access: ADC -> ADCPPUREF7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-323：ADC PPU7 基准寄存器（ADCPPUREF7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	基准值（有符号数），数值范围应当为 [-8192, 8191]

表 12-324：ADC PPU0 过高检测阈值寄存器（ADCPPUTHH0）位段定义

ADCPPUTHH0 (ADCPPU0 Trip-Zone High-Side Threshold Register) Offset: 0x270 Default: 0x00000000 Access: ADC -> ADCPPUTHH0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-325：ADC PPU0 过高检测阈值寄存器（ADCPPUTHH0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-326：ADC PPU1 过高检测阈值寄存器（ADCPPUTHH1）位段定义

ADCPPUTHH1 (ADCPPU1 Trip-Zone High-Side Threshold Register) Offset: 0x274 Default: 0x00000000 Access: ADC -> ADCPPUTHH1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-327：ADC PPU1 过高检测阈值寄存器（ADCPPUTHH1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-328：ADC PPU2 过高检测阈值寄存器（ADCPPUTHH2）位段定义

ADCPPUTHH2 (ADCPPU2 Trip-Zone High-Side Threshold Register) Offset: 0x278 Default: 0x00000000 Access: ADC -> ADCPPUTHH2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-329：ADC PPU2 过高检测阈值寄存器（ADCPPUTHH2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-330：ADC PPU3 过高检测阈值寄存器（ADCPPUTHH3）位段定义

ADCPPUTHH3 (ADCPPU3 Trip-Zone High-Side Threshold Register) Offset: 0x27C Default: 0x00000000 Access: ADC -> ADCPPUTHH3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-331：ADC PPU3 过高检测阈值寄存器（ADCPPUTHH3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-332：ADC PPU4 过高检测阈值寄存器（ADCPPUTHH4）位段定义

ADCPPUTHH4 (ADCPPU4 Trip-Zone High-Side Threshold Register) Offset: 0x280 Default: 0x00000000 Access: ADC -> ADCPPUTHH4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-333：ADC PPU4 过高检测阈值寄存器（ADCPPUTHH4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-334：ADC PPU5 过高检测阈值寄存器（ADCPPUTHH5）位段定义

ADCPPUTHH5 (ADCPPU5 Trip-Zone High-Side Threshold Register) Offset: 0x284 Default: 0x00000000 Access: ADC -> ADCPPUTHH5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-335：ADC PPU5 过高检测阈值寄存器（ADCPPUTHH5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-336：ADC PPU6 过高检测阈值寄存器（ADCPPUTHH6）位段定义

ADCPPUTHH6 (ADCPPU6 Trip-Zone High-Side Threshold Register) Offset: 0x288 Default: 0x00000000 Access: ADC -> ADCPPUTHH6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-337：ADC PPU6 过高检测阈值寄存器（ADCPPUTHH6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-338：ADC PPU7 过高检测阈值寄存器（ADCPPUTHH7）位段定义

ADCPPUTHH7 (ADCPPU7 Trip-Zone High-Side Threshold Register) Offset: 0x28C Default: 0x00000000 Access: ADC -> ADCPPUTHH7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-339：ADC PPU7 过高检测阈值寄存器（ADCPPUTHH7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过高检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-340：ADC PPU0 过低检测阈值寄存器（ADCPPUTHL0）位段定义

ADCPPUTHL0 (ADCPPU0 Trip-Zone Low-Side Threshold Register) Offset: 0x290 Default: 0x00000000 Access: ADC -> ADCPPUTHL0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-341：ADC PPU0 过低检测阈值寄存器（ADCPPUTHL0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-342：ADC PPU1 过低检测阈值寄存器（ADCPPUTHL1）位段定义

ADCPPUTHL1 (ADCPPU1 Trip-Zone Low-Side Threshold Register) Offset: 0x294 Default: 0x00000000 Access: ADC -> ADCPPUTHL1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-343：ADC PPU1 过低检测阈值寄存器（ADCPPUTHL1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-344：ADC PPU2 过低检测阈值寄存器（ADCPPUTHL2）位段定义

ADCPPUTHL2 (ADCPPU2 Trip-Zone Low-Side Threshold Register) Offset: 0x298 Default: 0x00000000 Access: ADC -> ADCPPUTHL2.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-345：ADC PPU2 过低检测阈值寄存器（ADCPPUTHL2）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-346：ADC PPU3 过低检测阈值寄存器（ADCPPUTHL3）位段定义

ADCPPUTHL3 (ADCPPU3 Trip-Zone Low-Side Threshold Register) Offset: 0x29C Default: 0x00000000 Access: ADC -> ADCPPUTHL3.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-347：ADC PPU3 过低检测阈值寄存器（ADCPPUTHL3）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-348：ADC PPU4 过低检测阈值寄存器（ADCPPUTHL4）位段定义

ADCPPUTHL4 (ADCPPU4 Trip-Zone Low-Side Threshold Register) Offset: 0x2A0 Default: 0x00000000 Access: ADC -> ADCPPUTHL4.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-349：ADC PPU4 过低检测阈值寄存器（ADCPPUTHL4）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-350：ADC PPU5 过低检测阈值寄存器（ADCPPUTHL5）位段定义

ADCPPUTHL5 (ADCPPU5 Trip-Zone Low-Side Threshold Register) Offset: 0x2A4 Default: 0x00000000 Access: ADC -> ADCPPUTHL5.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-351：ADC PPU5 过低检测阈值寄存器（ADCPPUTHL5）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-352：ADC PPU6 过低检测阈值寄存器（ADCPPUTHL6）位段定义

ADCPPUTHL6 (ADCPPU6 Trip-Zone Low-Side Threshold Register) Offset: 0x2A8 Default: 0x00000000 Access: ADC -> ADCPPUTHL6.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-353：ADC PPU6 过低检测阈值寄存器（ADCPPUTHL6）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-354：ADC PPU7 过低检测阈值寄存器（ADCPPUTHL7）位段定义

ADCPPUTHL7 (ADCPPU7 Trip-Zone Low-Side Threshold Register) Offset: 0x2AC Default: 0x00000000 Access: ADC -> ADCPPUTHL7.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 12-355：ADC PPU7 过低检测阈值寄存器（ADCPPUTHL7）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	过低检测阈值（有符号数） 数值范围为[-16384, 16383]

表 12-356：温度传感器控制寄存器（TSENSCTL）位段定义

TSENSCTL (Temperature Sensor Control Register) Offset: 0x2B0 Default: 0x00000000 Access: ADC -> TSENSCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	DEMSEL				SWAPBJT	OUTINV	EN

表 12-357：温度传感器控制寄存器（TSENSCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6:3	DEMSEL	RW	0x0	选择电流分支，用于动态组件匹配
2	SWAPBJT	RW	0x0	交换 BJT 0：不交换 BJT 1：交换 BJT
1	OUTINV	RW	0x0	反转 T-Sensor 的输出 0 和输出 1 0：正常连接 1：反转 T-Sensor 的输出 0 和输出 1
0	EN	RW	0x0	温度传感器使能 0：关闭 1：使能

表 12-358：ADC 模块写使能寄存器（ADCREGKEY）位段定义

ADCREGKEY (ADC Register Write-Allow Key Register) Offset: 0x2B4 Default: 0x1ACCE551 Access: ADC -> ADCREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 12-359：ADC 模块写使能寄存器（ADCREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 以解锁受保护的 ADC 寄存器

13 可编程增益放大器（PGA）

13.1 PGA 概述

SPC2168 集成了六个可变增益运算放大器（Programmable-Gain Amplifiers，PGA），可以支持同时放大六个通道的信号。每个 PGA 有两种工作模式：一种是差分模式，最大增益可到 64，另一种是单端模式，最大增益可到 32。在单端模式下，每个 PGA 可以支持两个单端信号的放大。

13.2 PGA 结构

六个 PGA 编号为 0 到 5。如图 13-1 和图 13-2 所示，PGA 的每个正向和负向输入都有一个 8 选 1 MUX（称为 PGA MUX）。每个 PGA 均可以掉电并旁路。

图 13-1：PGA0 ~ PGA2 结构框图

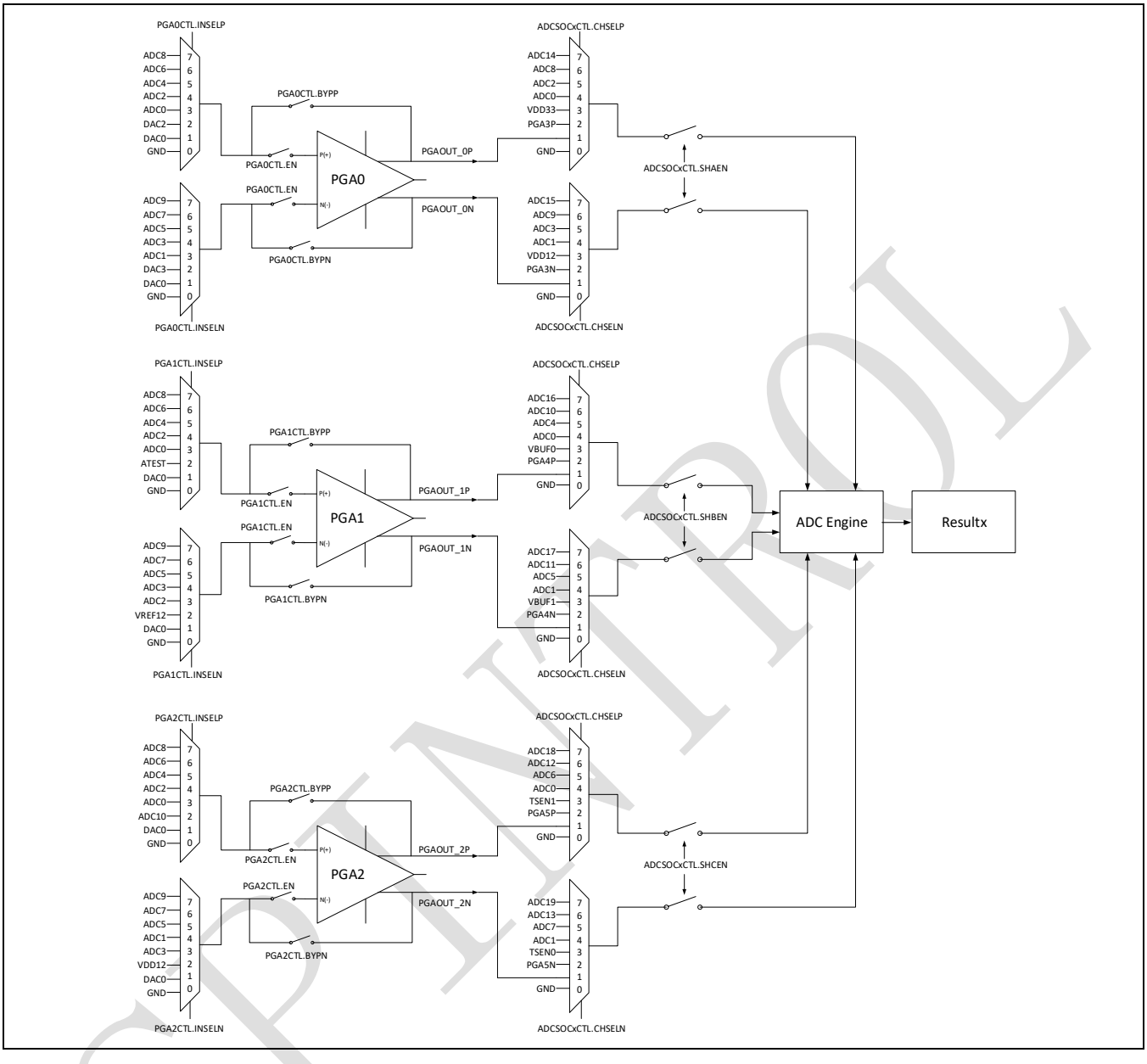
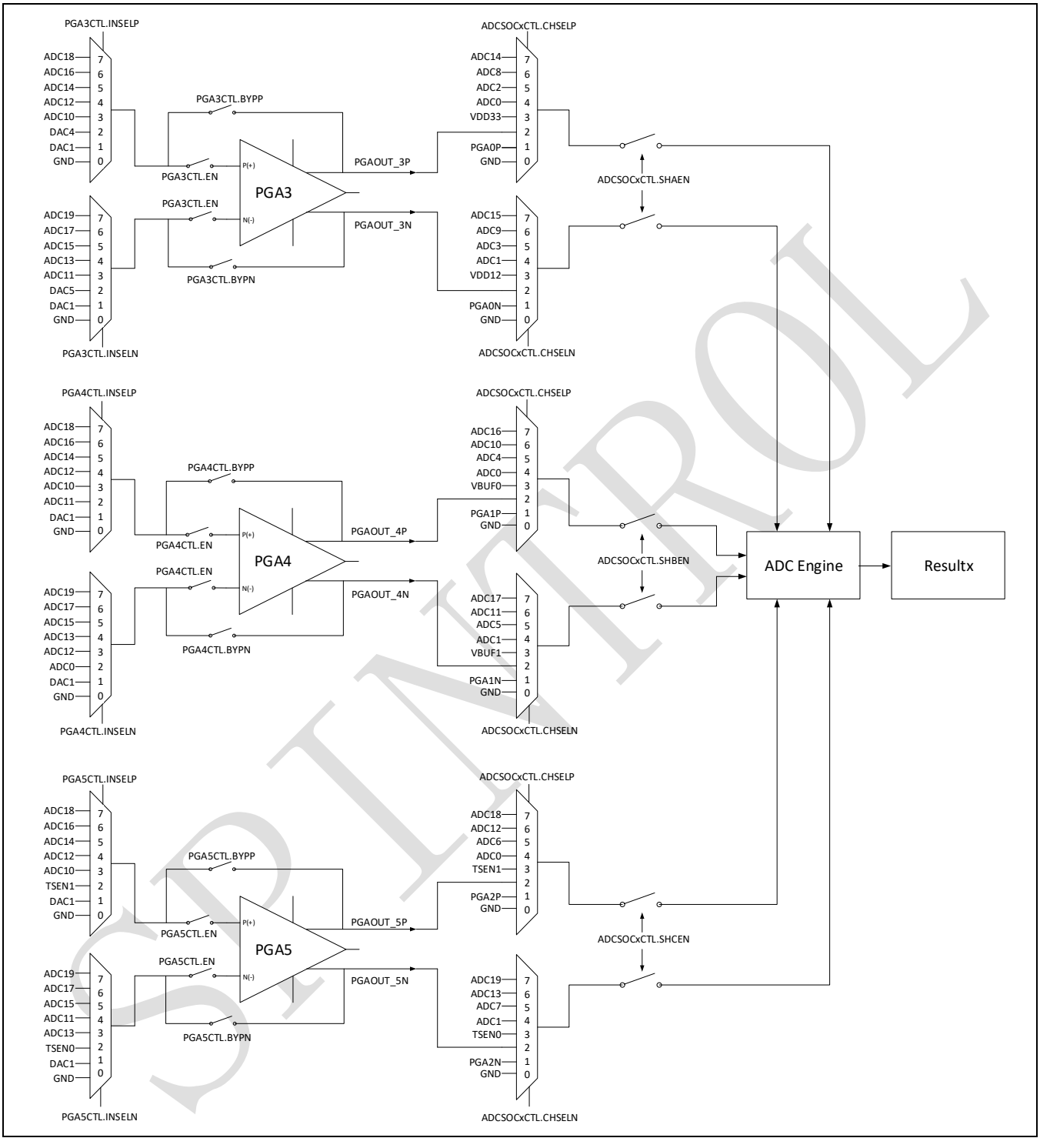


图 13-2：PGA3 ~ PGA5 结构框图



13.3 PGA 通道选择

每个 PGA 前面都有两个 8 选 1 的多路选择器（MUX），通过寄存器 PGAxCTL.INSELP 和 PGAxCTL.INSELN（x = 0，1，2，3，4，5）来选择 PGA 的正端和负端的输入。具体如下表所示：

表 13-1：PGA 0/1/2 MUX 输入选择

PGAxCTL.INSELP AND PGAxCTL.INSELN	in_p to PGA0	in_n to PGA0	in_p to PGA1	in_n to PGA1	in_p to PGA2	in_n to PGA2
7	ADC8	ADC9	ADC8	ADC9	ADC8	ADC9
6	ADC6	ADC7	ADC6	ADC7	ADC6	ADC7
5	ADC4	ADC5	ADC4	ADC5	ADC4	ADC5
4	ADC2	ADC3	ADC2	ADC3	ADC2	ADC1
3	ADC0	ADC1	ADC0	ADC2	ADC0	ADC3
2	DAC2	DAC3	ATEST	VREF12	ADC10	VDD12
1	DAC0	DAC0	DAC0	DAC0	DAC0	DAC0
0	GND	GND	GND	GND	GND	GND

表 13-2：PGA 3/4/5 MUX 输入选择

PGAxCTL.INSELP AND PGAxCTL.INSELN	in_p to PGA3	in_n to PGA3	in_p to PGA4	in_n to PGA4	in_p to PGA5	in_n to PGA5
7	ADC18	ADC19	ADC18	ADC19	ADC18	ADC19
6	ADC16	ADC17	ADC16	ADC17	ADC16	ADC17
5	ADC14	ADC15	ADC14	ADC15	ADC14	ADC15
4	ADC12	ADC13	ADC12	ADC13	ADC12	ADC11
3	ADC10	ADC11	ADC10	ADC12	ADC10	ADC13
2	DAC4	DAC5	ADC11	ADC0	TSEN1	TSEN0
1	DAC1	DAC1	DAC1	DAC1	DAC1	DAC1
0	GND	GND	GND	GND	GND	GND

示例 13.3.1 PGA 多路选择器使用示例

- 将温度传感器的差分输出连接到 PGA5
- 将 ADC10 输入电压（相对于 GND）连接到 PGA3
- 将 ADC8 和 DAC0 的差分信号连接到 PGA0

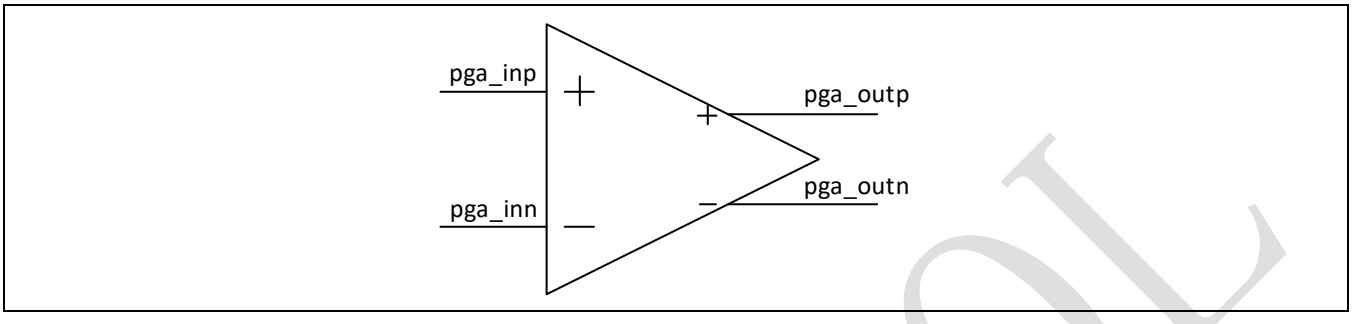
Example 13.3.1

```
void PGA_Example13_3_1(void)
{
   PGA->PGA5CTL.bit.INSELP=2;/* Select TSENS1 to PGA5_INP    */
   PGA->PGA5CTL.bit.INSELN=2;/* Select TSENS0 to PGA5_INN    */
   PGA->PGA3CTL.bit.INSELP=3;/* Select ADC10 to PGA3_INP     */
   PGA->PGA3CTL.bit.INSELN=0;/* Select GND to PGA3_INN        */
   PGA->PGA0CTL.bit.INSELP=7;/* Select ADC8 to PGA0_INP      */
   PGA->PGA0CTL.bit.INSELN=1;/* Select DAC0 to PGA0_INN       */
}
```

13.4 PGA 模式和增益

每个 PGA 都可以单独配置成差分模式或者单端模式。

图 13-3：PGA 输入和输出



如果配置成单端模式，可以只打开正端通路，pga_inp 做为输入，pga_outp 做为输出；也可以只打开负端通路，pga_inn 做为输入，pga_outn 做为输出；还可以配置成两个单端运放，也就是同时打开正端和负端通路。具体模式选择取决于寄存器 PGAxCTL.MODE 的配置，如下表所示：

表 13-3：PGA 模式选择

PGAxCTL.MODE	描述
0	差分模式
1	单端模式，只打开正端通路
2	单端模式，只打开负端通路
3	单端模式，同时打开正端和负端通路

PGA 单端模式和差分模式下的增益选择，如下表所示：

表 13-4：PGA 增益选择

PGAxCTL.GAINP/ PGAxCTL.GAINN	单端模式增益	差分模式增益
0	1	2
1	2	4
2	4	8
3	8	16
4	12	24
5	16	32
6	24	48
7	32	64

单端模式下，正端通路的增益是由寄存器 PGAxCTL.GAINP 设置，负端通路的增益是由寄存器 PGAxCTL.GAINN 设置，输出和输入关系如下所示：

$$\mathrm{OUTP} = \mathrm{GAINP} \times \mathrm{INP}$$

$$\mathrm{OUTN} = \mathrm{GAINN} \times \mathrm{INN}$$

PGA 的输入范围是轨到轨（rail-to-rail）的。但是在使用中要注意,为了防止输出管饱和,需要保证输出范围在 0.3V 到 AVDD-0.3V 之间。所以对于不同的增益，输入范围也是不同的，具体如下表所示：

表 13-5：PGA 单端模式在不同增益下的输入范围

PGAxCTL.GAINP(N)	增益	输入范围（V）	输出范围 (V)
0	1	0.3~3	0.3~3
1	2	0.15~1.5	0.3~3
2	4	0.075~0.75	0.3~3
3	8	0.0375~0.375	0.3~3
4	12	0.025~0.25	0.3~3
5	16	0.01875~0.1875	0.3~3
6	24	0.0125~0.125	0.3~3
7	32	0.009375~0.09375	0.3~3

在 PGA 差分模式下，需要将寄存器 PGAxCTL.GAINP 和 PGAxCTL.GAINN 设置为相同的值。假如都设为 Gain，那么差分输出的表达式如下：

$$\mathrm{OUTP} - \mathrm{OUTN} = \mathrm{Gain} \times (INP - INN)$$

输出共模电压可以设置为 INP，也可以设置为 INN，这是由寄存器 PGAxCTL.CMSEL 来设置的。该寄存器默认值为 0，选择 INN 做为输出共模电压。如果 PGAxCTL.CMSEL=1，选择 INP 做为输出共模电压。输出电压和输入电压关系如下所示：

如果 PGAxCTL.CMSEL = 0，

$$\mathrm{OUT\_CM} = \mathrm{INN}$$

$$\mathrm{OUTP} = \mathrm{INN} + (INP - INN) \times \frac{Gain}{2}$$

$$\mathrm{OUTN} = \mathrm{INN} - (INP - INN) \times \frac{Gain}{2}$$

如果 PGAxCTL.CMSEL = 1，

$$\mathrm{OUT\_CM} = \mathrm{INP}$$

$$\mathrm{OUTP} = \mathrm{INP} + (INP - INN) \times \frac{Gain}{2}$$

$$\mathrm{OUTN} = \mathrm{INP} - (INP - INN) \times \frac{Gain}{2}$$

示例 13.4.1　PGA 单端模式只打开正端通路

打开 PGA2 的正端通路，INP=0.5V，期望输出 OUTP=1V

- 设置 PGA2 为单端模式，只打开正端通路
- 设置单端增益为 2

Example 13.4.1

```
void PGA_Example13_4_1(void)
{
  PGA->PGA2CTL.bit.MODE=1;/* Set single mode with positive path only     */
  PGA->PGA2CTL.bit.GAINP=1;/* Set single mode gain =2 */
}
```

示例 13.4.2　PGA 单端模式同时打开正端和负端通路

PGA1 设置为单端模式，并且同时打开正端和负端通路。如果 INP 输入范围在 0.3V 到 0.5V，INN 输入范围在 0.06V 到 0.14V，需要将正端通路的增益设为 4，正端的输出在 1.2V 到 2V；将负端通路的增益设为 16，则负端输出在 0.96V 到 2.24V。

- 设置 PGA1 为单端模式，并且同时打开正端和负端通路
- 设置 PGA1 正端通路增益为 4
- 设置 PGA1 负端通路增益为 16

Example 13.4.2

```
void PGA_Example13_4_2(void)
{
PGA->PGA1CTL.bit.MODE=3;/* Set single mode with positive path and negative path    */
PGA->PGA1CTL.bit.GAINP=2;/* Set single mode positive path gain = 4 */
PGA->PGA1CTL.bit.GAINN=5;/* Set single mode negative path gain = 16 */
}
```

示例 13.4.3　PGA 差分模式

如果 AVDD=3.3V，增益为 4，INP=2.05V，INN=1.65V，INN 设置为输出共模电压，那么

OUTP = 1.65 + (2.05-1.65) * 4/2 = 2.45V

OUTN = 1.65 – (2.05-1.65) * 4/2 = 0.85V

- 设置 PGA1 为差分模式
- 设置 INN 为输出共模电压
- 设置差分增益为 4

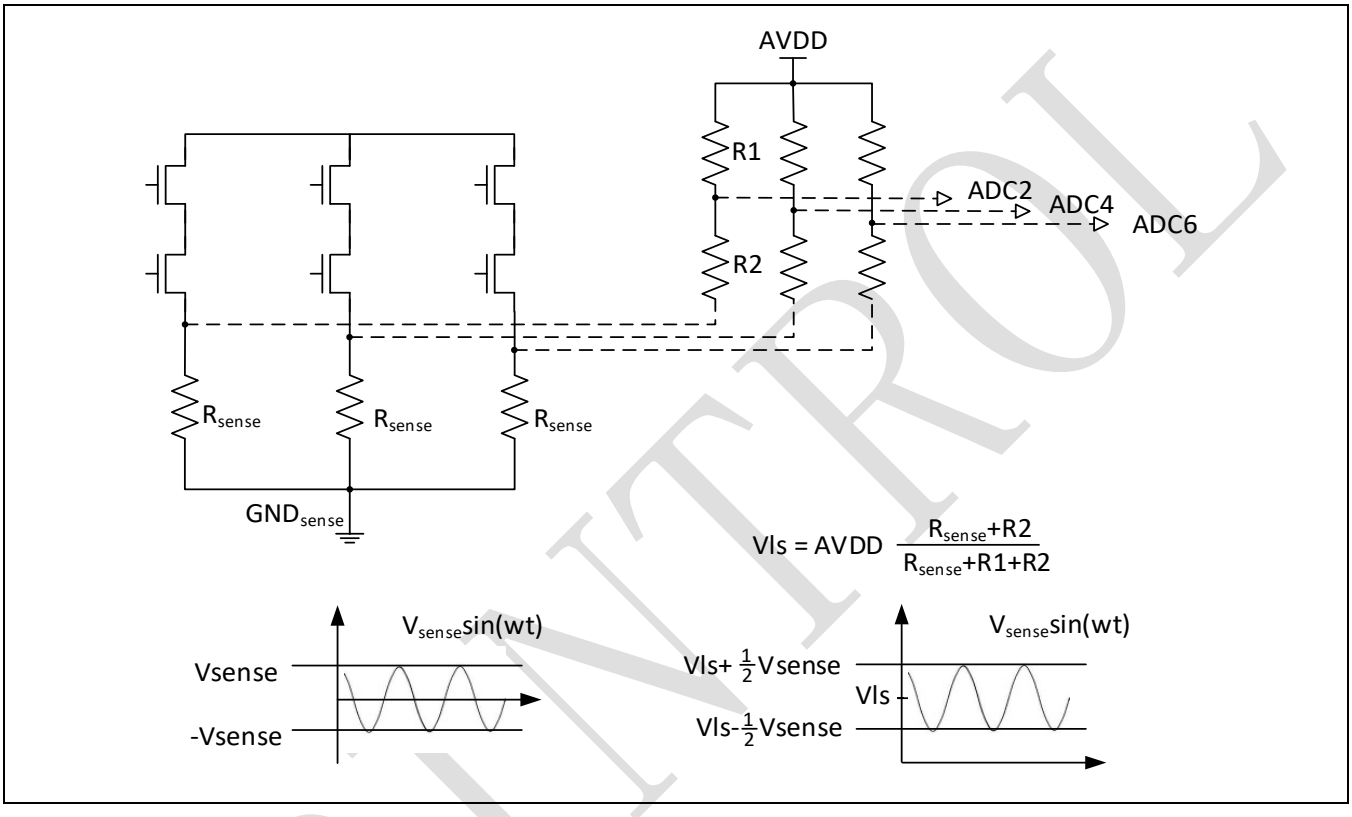
Example 13.4.3

```
void PGA_Example13_4_3(void)
{
PGA->PGA2CTL.bit.MODE=0;/* Set differential mode      */
PGA->PGA2CTL.bit.CMSEL=1;/* Use INN as output common mode voltage    */
PGA->PGA2CTL.bit.GAINP=1;/* Set differential mode gain = 4 */
PGA->PGA2CTL.bit.GAINN=1;/* Set differential mode gain = 4 */
}
```

13.5 PGA 单端模式在电流采样中应用

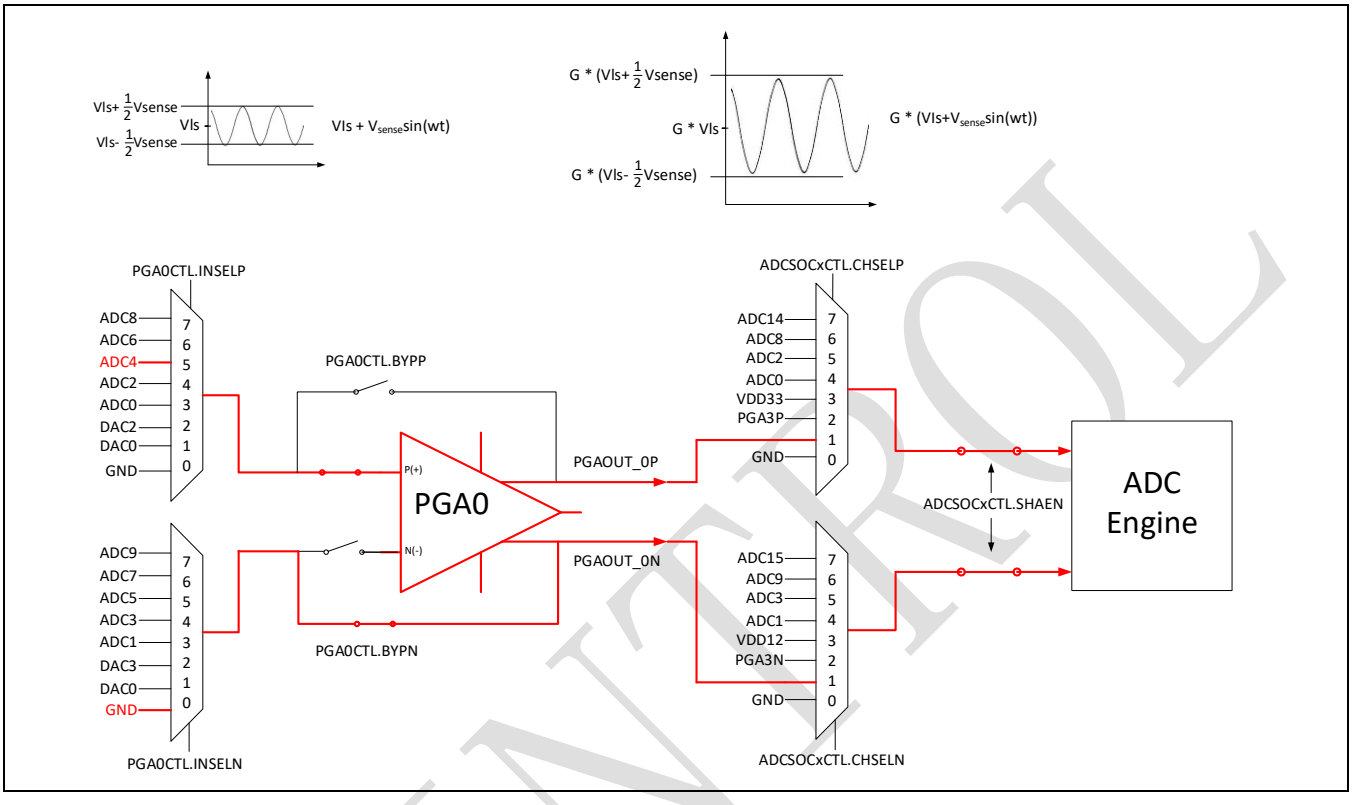
对于电机电流检测，如图 13-4 所示，假设使用 ADC2，ADC4，ADC6 三个通道来采样，相应的 PGA 配置如表 13-1 所示。需要注意的是，需要将 V_{sense} 信号电平上移来保证该电压不会是负压，然后再送给 PGA 做输入。

图 13-4：单端模式下采样电流示意图



如图 13-5 所示，PGA0 选择 ADC4 信号进行放大，并且将其输出传送到 ADC 输入。输入信号 Vls+Vsense*sin(wt)会被放大 G 倍。

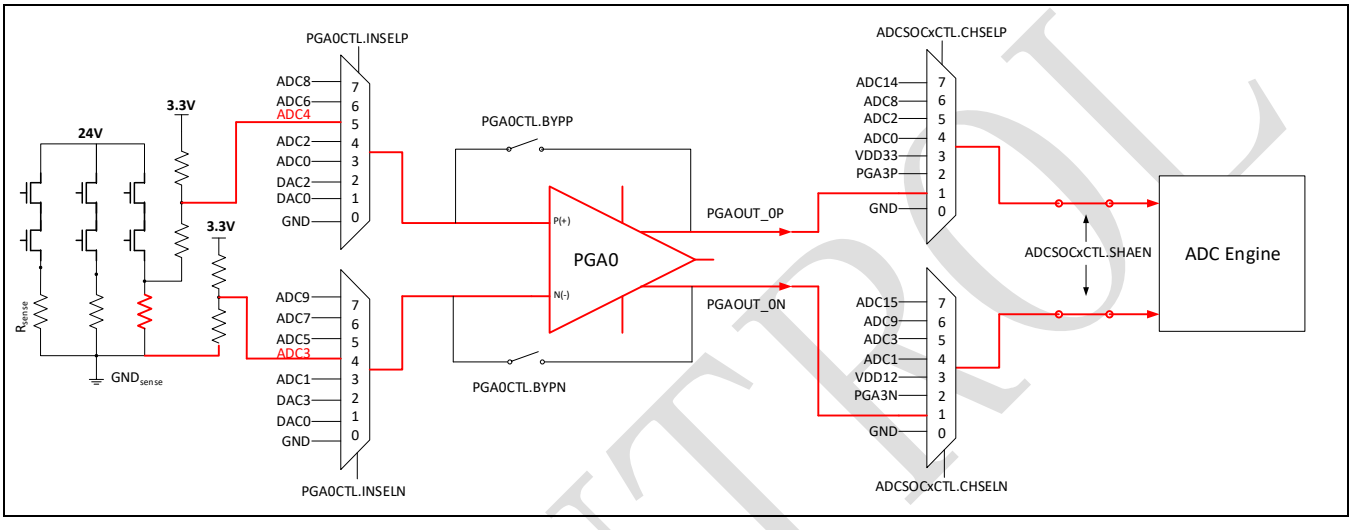
图 13-5：PGA0 选择 ADC4 作为输入（单端模式）



13.6 PGA 差分模式在电流采样中应用

电机控制通常通过测量驱动器场效应管的电流来实现，如图 13-6 所示。而电流是通过测量采样电阻 Rsense 的电压来测量的，产生的电压是正弦波，并且均值为 0V。由于采样电阻 Rsense 上的负电压不能传入芯片，所以通过使用由两个相同电阻 R 组成的电阻分压器将其电平上移 AVDD/2 再传给 PGA 输入。

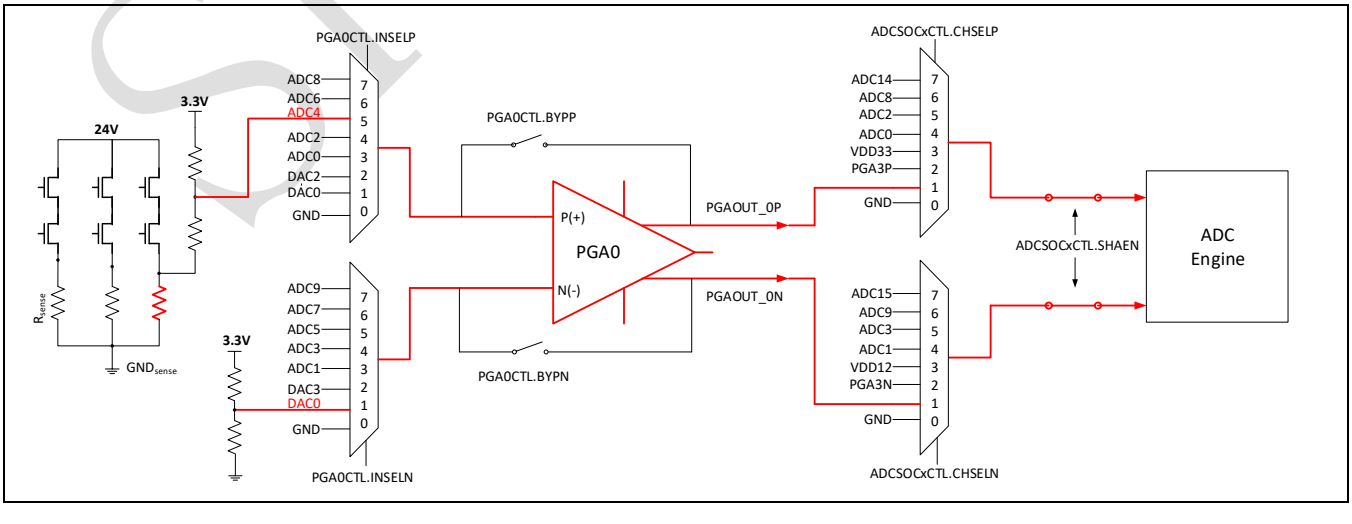
图 13-6：电机采样示意图



电平偏移以后的信号平均值为 AVDD/2，振幅等于采样电阻上电压振幅的一半。采样电阻 Rsense 的负端（GROUND）也需要做相同的电平上移。

如图 13-6 所示，PGA0 配置为差分模式，采样电阻的正端通过电平上移后通过 ADC4 送到 PGA0 的正端输入，采样电阻的负端通过电平上移后通过 ADC3 送到 PGA0 的负端输入。PGA0 的输出共模设置为负端输入，即等于 AVDD/2，这样有利于增大 PGA0 的输出信号范围。PGA0 输出差分信号等于输入差分信号乘以增益。

图 13-7：电机采样示意图（使用内部 DAC）

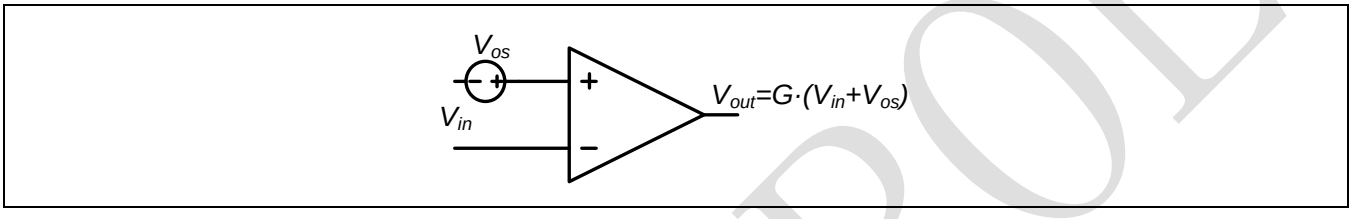


另外一种方式，可以取代外部的电阻分压产生 AVDD/2，可以用内部的 10 比特 DAC 做为 PGA0 的负端输入，并设置为输出共模电压，如图 13-7 所示。这种方式的优点是每一条采样支路上可以节省一个管脚。

13.7 PGA 偏移误差校准

由于芯片上晶体管的失配，每个放大器都有一个输入参考偏移量，如图 13-8 所示。PGA 输出等于输入与额外的电压 Vos 之和再乘以增益，其中 Vos 为输入参考偏移量，或者简单称为偏移量。

图 13-8：PGA 偏移误差

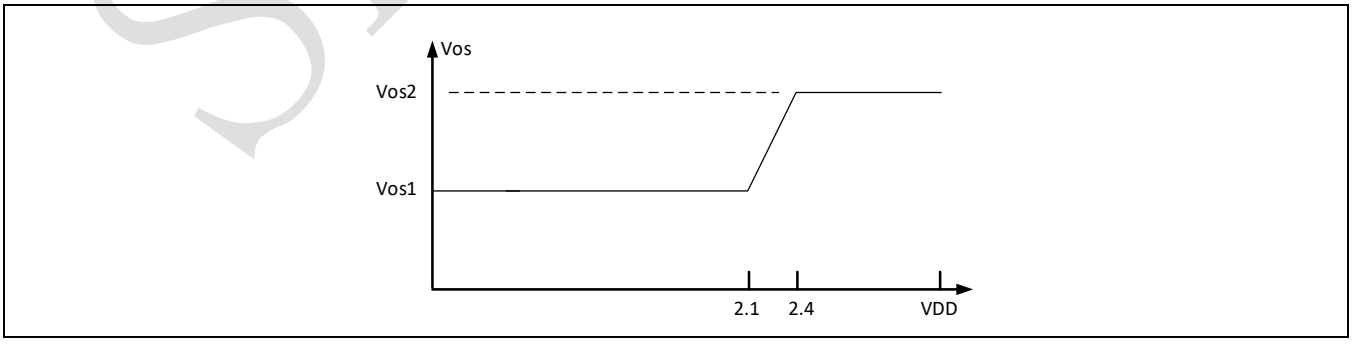


为了校准偏移量，偏移量必须和输入无关，即 VOS 不能依赖于输入的差分或者共模电压。SPC2168 放大器的偏移随输入共模电压的关系如图 13-9 所示。对于小于 2.1V 的输入共模电压，偏移量有一个值，对于输入共模大于 2.4V，偏移量有另一个值，并且在 2.1 和 2.4V 之间有一个过渡区。对于一般的应用来说，输入共模电压通常在 AVDD/2, 通常都是小于 2.1V，所以 VOS 是一个固定值，所以该偏移量可以被校准。

如果确定了偏移量，可以通过软件进行后处理，将测量的输出信号减去偏移量乘以增益，从而得到理想的放大信号信息。因此，偏移量标定是精确信号放大和测量的重要组成部分。

为了校准偏移量，在 PGA 的正负输入端送入相同的信号（图 13-9 中差分输入为 0）。然后，输出除以增益，就是该 PGA 的偏移量。建议将片内 10 位 DAC 的输出送入 PGA 输入（推荐使用 DAC1，相关配置见表 13-1），并且设置 DAC1 的输出小于 2.1V（推荐设为 AVDD/2）。PGA 将输出 $V_{OS}1$ 乘以增益。由于 $V_{OS}1$ 小于 2mV，在最大增益 64 内，PGA 差分输出小于 200mV，注意不要让 PGA 输出饱和。

图 13-9：偏移量随输入共模电压的变化



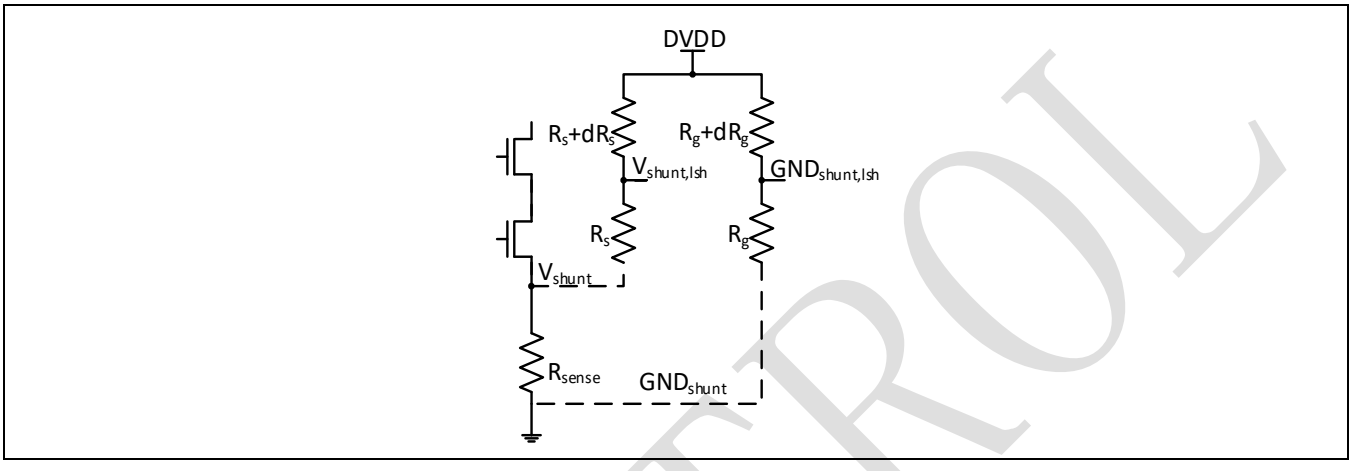
不建议使用外部电平移位器来做校准，因为电平移位器的失配更大。此外，由于使用相同的 DAC 输出应用于 PGA 的正端和负端输入，即使内部 DAC 有轻微的失配，差分输入信号也仍然为零。

对于 PGA 的三种不同的工作模式，对应三种不同的偏移量：差分模式下的 PGA、单端模式下只开通正端通路、单端模式下只开通负端通路。不同的偏移量应通过 ADC 后处理来确定。

13.8 电阻电平移位器偏移误差校准

从图 13-6 可以看出，每个电阻电平移位器由两个电阻组成。两个电阻阻值应相等，以便使信号向上移位 AVDD/2。这两个电阻通常会有不匹配，这种不匹配需要校准，以便确定 Rsense 两端的电压。

图 13-10：电平移位器电阻失配



为简单起见，但不降低讨论的一般性，以图 3-10 所示情况为例。V_{shunt} 和 GND_{shunt} 电压，理想中应该向上移位 AVDD/2，但是由于电阻不匹配，实际移位值会所差别。根据下面的表达式，可以计算出移位后的电压值：

$$V_{shunt,lsh} = V_{ddx}\frac{R_s}{2\cdot R_s + \delta R_s} + V_{shunt}\frac{R_s + \delta R_s}{2\cdot R_s + \delta R_s} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_s}{2}\right] + \frac{V_{shunt}}{2}\left[1 + \frac{\delta_s}{2}\right]$$

$$GND_{shunt,lsh} = V_{ddx}\frac{R_g}{2\cdot R_g + \delta R_g} + GND_{shunt}\frac{R_g + \delta R_g}{2\cdot R_g + \delta R_g} = \frac{V_{ddx}}{2}\left[1 - \frac{\delta_g}{2}\right] + \frac{GND_{shunt}}{2}\left[1 + \frac{\delta_g}{2}\right]$$

在这里，Rs/Rs=δ_s，Rg/Rg=δ_g。因此，移位后的差分输出电压为：

$$V_{ind} = V_{shunt,lsh} - GND_{shunt,lsh} = \frac{V_{ddx}}{2}\cdot\frac{\delta_g - \delta_s}{2} + \frac{V_{shunt} - GND_{shunt}}{2} + \frac{V_{shunt}}{2}\cdot\frac{\delta_s}{2}$$

这里，忽略了 $GND_{shunt}\delta_g/4$，因为 GND_{shunt} 和δ_g 都是小量。我们发现失配误差可以分为增益误差（V_{shunt} 和δ_s /2 乘积）和偏移误差（V_{shunt}=0 时的 V_{ind}）。增益误差项相当于 Rsense 电阻的变化。这两个项都可以在当电机不工作时通过测量电压来确定，具体步骤如下：

- 测量 AVDD。可以将 AVDD 通过 ADC 多路选择器送到 ADC 测量，配置如表 13-1 所示。
- 当电机不工作时，没有电流通过 Rsense，也就是 V_{shunt}=0，测量 $V_{shunt,lsh}$。通过上面 $V_{shunt,lsh}$ 的表达式可以计算出δ_s。
- 当电机不工作时，没有电流通过 Rsense，测量 V_{ind}。根据 V_{ind} 的表达式可以计算出$\delta_g - \delta_s$，从而计算出δ_g

13.9 上电顺序

当 PGA 开始启动，需要遵循下面的上电顺序：

第一步：使能 ADC 的带隙基准模块（bandgap）

第二步：使能 PGA 内部模拟电路

13.10 寄存器

13.10.1 PGA 寄存器列表

表 13-6：PGA 模块基地址

外设模块	基地址
PGA	0x4000 8C00

表 13-7：PGA 寄存器列表

寄存器	偏移地址	描述	复位值
PGA0CTL*	0x2C0	PGA0 控制寄存器	0x00000008
PGA1CTL*	0x2C4	PGA1 控制寄存器	0x00000008
PGA2CTL*	0x2C8	PGA2 控制寄存器	0x00000008
PGA3CTL*	0x2CC	PGA3 控制寄存器	0x00000008
PGA4CTL*	0x2D0	PGA4 控制寄存器	0x00000008
PGA5CTL*	0x2D4	PGA5 控制寄存器	0x00000008
PGAREGKEY	0x2E0	PGA 模块写使能寄存器	0x1ACCE551

注意： 由*标识的寄存器仅当 PGAREGKEY =0x1ACCE551 时才可以改写。

13.10.2 PGA 寄存器

表 13-8 到表 13-21 提供了 PGA 模块相关寄存器的详细信息。

表 13-8：PGA0 控制寄存器（PGA0CTL）位段定义

PGA0CTL (PGA0 Control Register) Offset: 0x2C0 Default: 0x00000000 Access: PGA -> PGA0CTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-9：PGA0 控制寄存器（PGA0CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA0 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA0 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA0 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA0 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式

位段	位段名	属性	复位值	描述
9:7	INSELN	RW	0x0	PGA0 负端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：DAC3 输出 011：ADC1 输入（来自 GPIO1） 100：ADC3 输入（来自 GPIO3） 101：ADC5 输入（来自 GPIO5） 110：ADC7 输入（来自 GPIO7） 111：ADC9 输入（来自 GPIO9）
6:4	INSELP	RW	0x0	PGA0 正端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：DAC2 输出 011：ADC0 输入（来自 GPIO0） 100：ADC2 输入（来自 GPIO2） 101：ADC4 输入（来自 GPIO4） 110：ADC6 输入（来自 GPIO6） 111：ADC8 输入（来自 GPIO8）
3	CMSEL	RW	0x0	PGA0 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA0 模式 00：差分模式 11：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA0 使能 0：关闭 PGA0 1：使能 PGA0

表 13-10：PGA1 控制寄存器（PGA1CTL）位段定义

PGA1CTL (PGA1 Control Register) Offset: 0x2C4 Default: 0x00000000 Access: PGA -> PGA1CTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-11：PGA1 控制寄存器（PGA1CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA1 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA1 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA1 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA1 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
9:7	INSELN	RW	0x0	PGA1 负端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：1.2V 电压

位段	位段名	属性	复位值	描述
				011：ADC2 输入（GPIO2） 100：ADC3 输入（GPIO3） 101：ADC5 输入（GPIO5） 110：ADC7 输入（GPIO7） 111：ADC9 输入（GPIO9）
6:4	INSELP	RW	0x0	PGA1 正端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：ATEST 输出电压 011：ADC0 输入（GPIO0） 100：ADC2 输入（GPIO2） 101：ADC4 输入（GPIO4） 110：ADC6 输入（GPIO6） 111：ADC8 输入（GPIO8）
3	CMSEL	RW	0x0	PGA1 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA1 模式 00：差分模式 01：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA1 使能 0：关闭 PGA1 1：使能 PGA1

表 13-12：PGA2 控制寄存器（PGA2CTL）位段定义

PGA2CTL (PGA2 Control Register) Offset: 0x2C8 Default: 0x00000000
Access: PGA -> PGA2CTL.all

31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-13：PGA2 控制寄存器（PGA2CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA2 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA2 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA2 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA2 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
9:7	INSELN	RW	0x0	PGA2 负端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：1.2V 数字电源

位段	位段名	属性	复位值	描述
				011：ADC3 输入（GPIO3） 100：ADC1 输入（GPIO1） 101：ADC5 输入（GPIO5） 110：ADC7 输入（GPIO7） 111：ADC9 输入（GPIO9）
6:4	INSELP	RW	0x0	PGA2 正端输入选择 000：AGND（模拟地） 001：DAC0（用于共模输入） 010：ADC10 输入（GPIO10） 011：ADC0 输入（GPIO0） 100：ADC2 输入（GPIO2） 101：ADC4 输入（GPIO4） 110：ADC6 输入（GPIO6） 111：ADC8 输入（GPIO8）
3	CMSEL	RW	0x0	PGA2 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA2 模式 00：差分模式 01：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA2 使能 0：关闭 PGA2 1：使能 PGA2

表 13-14：PGA3 控制寄存器（PGA3CTL）位段定义

PGA3CTL (PGA3 Control Register) Offset: 0x2CC Default: 0x00000008 Access: PGA -> PGA3CTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-15：PGA3 控制寄存器（PGA3CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA3 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA3 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA3 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA3 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
9:7	INSELN	RW	0x0	PGA3 负端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：DAC5 输出

位段	位段名	属性	复位值	描述
				011：ADC11 输入（GPIO11） 100：ADC13 输入（GPIO13） 101：ADC15 输入（GPIO15） 110：ADC17 输入（GPIO17） 111：ADC19 输入（GPIO19）
6:4	INSELP	RW	0x0	PGA3 正端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：DAC4 输出 011：ADC10 输入（GPIO10） 100：ADC12 输入（GPIO12） 101：ADC14 输入（GPIO14） 110：ADC16 输入（GPIO16） 111：ADC18 输入（GPIO18）
3	CMSEL	RW	0x0	PGA3 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA3 模式 00：差分模式 01：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA3 使能 0：关闭 PGA3 1：使能 PGA3

表 13-16：PGA4 控制寄存器（PGA4CTL）位段定义

PGA4CTL (PGA4 Control Register) Offset: 0x2D0 Default: 0x00000008 Access: PGA -> PGA4CTL.all							
31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-17：PGA4 控制寄存器（PGA4CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA4 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA4 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA4 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA4 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
9:7	INSELN	RW	0x0	PGA4 负端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：ADC0 输入（GPIO0）

位段	位段名	属性	复位值	描述
				011：ADC12 输入（GPIO12） 100：ADC13 输入（GPIO13） 101：ADC15 输入（GPIO15） 110：ADC17 输入（GPIO17） 111：ADC19 输入（GPIO19）
6:4	INSELP	RW	0x0	PGA4 正端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：ADC11 输入（GPIO11） 011：ADC10 输入（GPIO10） 100：ADC12 输入（GPIO12） 101：ADC14 输入（GPIO14） 110：ADC16 输入（GPIO16） 111：ADC18 输入（GPIO18）
3	CMSEL	RW	0x0	PGA4 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA4 模式 00：差分模式 01：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA4 使能 0：关闭 PGA4 1：使能 PGA4

表 13-18：PGA5 控制寄存器（PGA5CTL）位段定义

PGA5CTL (PGA5 Control Register) Offset: 0x2D4 Default: 0x00000008
Access: PGA -> PGA5CTL.all

31	30	29	28	27	26	25	24
RESERVED_31_18							
23	22	21	20	19	18	17	16
RESERVED_31_18						BYPN	BYPP
15	14	13	12	11	10	9	8
GAINN			GAINP			INSELN	
7	6	5	4	3	2	1	0
INSELN	INSELP			CMSEL	MODE		EN

表 13-19：PGA5 控制寄存器（PGA5CTL）位段描述

位段	位段名	属性	复位值	描述
31:18	RESERVED_31_18	RO	0x0	保留
17	BYPN	RW	0x0	PGA5 负端通路旁路（bypass） 0：不旁路 1：旁路
16	BYPP	RW	0x0	PGA5 正端通路旁路（bypass） 0：不旁路 1：旁路
15:13	GAINN	RW	0x0	PGA5 负端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
12:10	GAINP	RW	0x0	PGA5 正端通路增益 000：1x 单端模式，2x 差分模式 001：2x 单端模式，4x 差分模式 010：4x 单端模式，8x 差分模式 011：8x 单端模式，16x 差分模式 100：12x 单端模式，24x 差分模式 101：16x 单端模式，32x 差分模式 110：24x 单端模式，48x 差分模式 111：32x 单端模式，64x 差分模式
9:7	INSELN	RW	0x0	PGA5 负端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：T-sensor 输出 0

位段	位段名	属性	复位值	描述
				011：ADC13 输入（GPIO13） 100：ADC11 输入（GPIO11） 101：ADC15 输入（GPIO15） 110：ADC17 输入（GPIO17） 111：ADC19 输入（GPIO19）
6:4	INSELP	RW	0x0	PGA5 正端输入选择 000：AGND（模拟地） 001：DAC1（用于共模输入） 010：T-sensor 输出 1 011：ADC10 输入（GPIO10） 100：ADC12 输入（GPIO12） 101：ADC14 输入（GPIO14） 110：ADC16 输入（GPIO16） 111：ADC18 输入（GPIO18）
3	CMSEL	RW	0x0	PGA5 输出共模电压选择 该位只在差分模式下有效。 0：选择负端输入做为输出共模电压 1：选择正端输入做为输出共模电压
2:1	MODE	RW	0x0	PGA5 模式 00：差分模式 01：单端模式，只打开正端通路 10：单端模式，只打开负端通路 11：单端模式，同时打开正端和负端通路
0	EN	RW	0x0	PGA5 使能 0：关闭 PGA5 1：使能 PGA5

表 13-20：PGA 模块写使能寄存器（PGAREGKEY）位段定义

PGAREGKEY (PGA Register Write-Allow Key Register) Offset: 0x2E0 Default: 0x1ACCE551 Access: PGA -> PGAREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 13-21：PGA 模块写使能寄存器（PGAREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 PGA 寄存器

14 比较器（COMP）

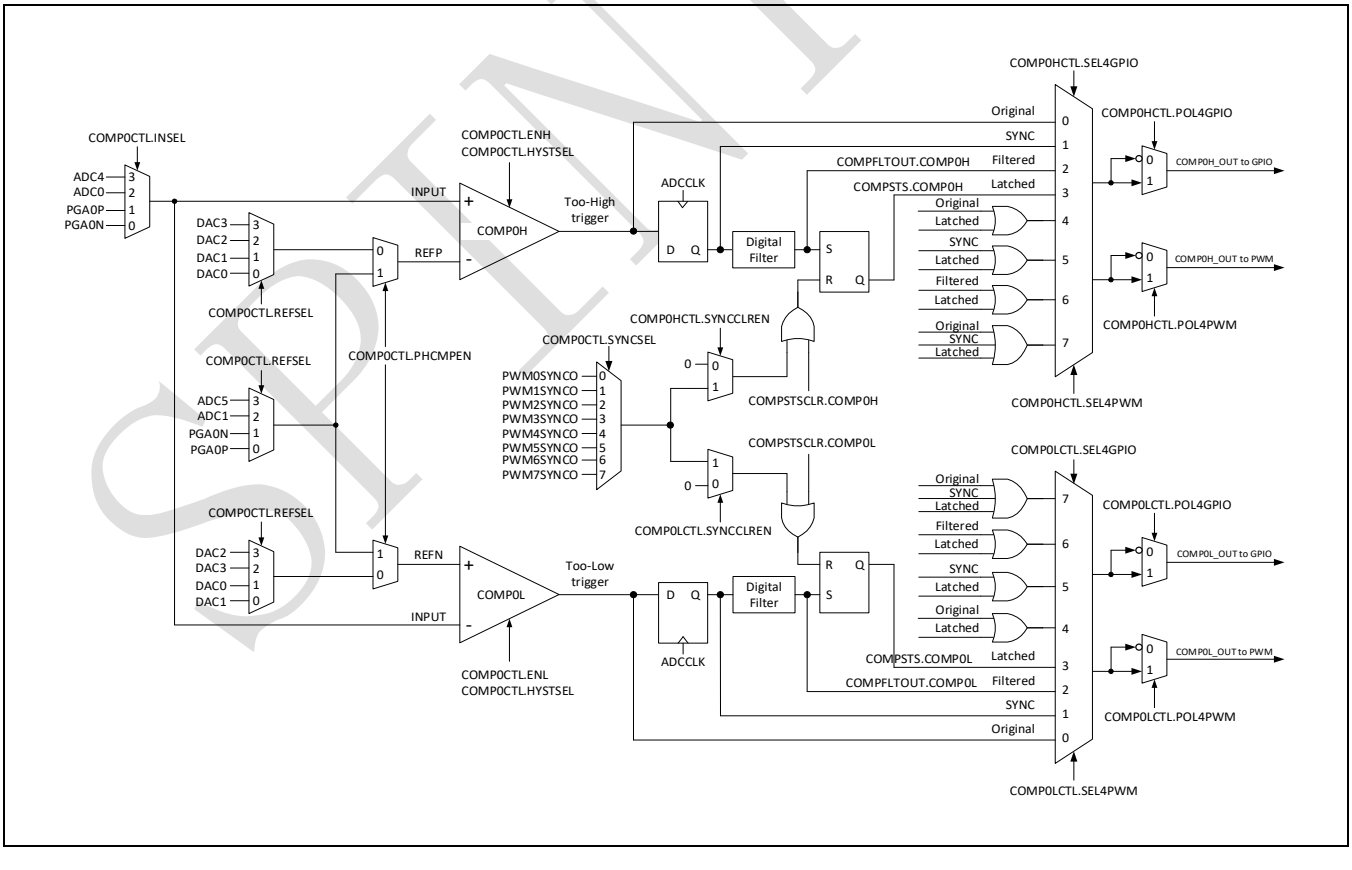
14.1 比较器概述

SPC2168 中的每个 PGA 的输出，会送到 2 个高速轨对轨输入比较器，用来检查电压是否过高或过低，从而实现过流检测，共 12 个比较器。另外还有 2 对比较器可用于其他应用，如双电机控制，或者支持 PFC。所以，SPC2168 一共有 16 个比较器。

14.2 比较器结构

图 14-1 给出了比较器 COMP0 的框图，包括两个比较器，其中，COMP0H 用于监测电压过高，COMP0L 用于检查电压过低，图中还包含比较器的输入选择器（MUX）。SPC2168 还具有相位比较器功能，当 COMP0CTL.PHCMPEN=1 时，此功能启用。举例来说，如果 INSEL=1，REFSEL=1，PGA0P 将被发送到 COMP0H 的正输入端和 COMP0L 的负输入端，PGA0N 将被发送到 COMP0H 的负输入端和 COMP0L 的正输入端。如果 COMP0CTL.PHCMPEN=0 时，比较器的参考电压来自四个 10 位 DAC 中的一个。每个比较器都有迟滞，迟滞的电压可以设置，最大 30mV，每个比较器的输出可以触发 PWM 封锁（Trip-zone），典型的延迟为 50ns。

图 14-1：比较器 COMP0 框图



14.3 比较器输入选择

8 对比较器（COMP0 ~ COMP7）的输入选择如表 14-1 所示。

表 14-1：比较器 COMP0 ~ COMP7 输入选择

COMP0 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC4	1	3	ADC5	0	3	DAC3	DAC2
2	ADC0	1	2	ADC1	0	2	DAC2	DAC3
1	PGA0P	1	1	PGA0N	0	1	DAC1	DAC0
0	PGA0N	1	0	PGA0P	0	0	DAC0	DAC1
COMP1 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC6	1	3	ADC7	0	3	DAC3	DAC2
2	ADC0	1	2	ADC2	0	2	DAC2	DAC3
1	PGA1P	1	1	PGA1N	0	1	DAC1	DAC0
0	PGA1N	1	0	PGA1P	0	0	DAC0	DAC1
COMP2 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC8	1	3	ADC9	0	3	DAC3	DAC2
2	ADC0	1	2	ADC3	0	2	DAC2	DAC3
1	PGA2P	1	1	PGA2N	0	1	DAC1	DAC0
0	PGA2N	1	0	PGA2P	0	0	DAC0	DAC1
COMP3 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC14	1	3	ADC15	0	3	DAC5	DAC4
2	ADC10	1	2	ADC11	0	2	DAC4	DAC5
1	PGA3P	1	1	PGA3N	0	1	DAC1	DAC0
0	PGA3N	1	0	PGA3P	0	0	DAC0	DAC1
COMP4 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC16	1	3	ADC17	0	3	DAC5	DAC4
2	ADC10	1	2	ADC12	0	2	DAC4	DAC5
1	PGA4P	1	1	PGA4N	0	1	DAC1	DAC0
0	PGA4N	1	0	PGA4P	0	0	DAC0	DAC1
COMP5 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC18	1	3	ADC19	0	3	DAC5	DAC4
2	ADC10	1	2	ADC13	0	2	DAC4	DAC5
1	PGA5P	1	1	PGA5N	0	1	DAC1	DAC0
0	PGA5N	1	0	PGA5P	0	0	DAC0	DAC1
COMP6 输入选择								
INSEL	INPUT	PHCOMPEN	REFSEL	REFP/N	PHCOMPEN	REFSEL	REFP	REFN
3	ADC16	1	3	ADC17	0	3	DAC3	DAC2

2	ADC12	1	2	ADC13	0	2	DAC2	DAC3
1	ADC6	1	1	ADC7	0	1	DAC1	DAC0
0	ADC2	1	0	ADC3	0	0	DAC0	DAC1
COMP7 输入选择								
INSEL	**INPUT**	**PHCOMPEN**	**REFSEL**	**REFP/N**	**PHCOMPEN**	**REFSEL**	**REFP**	**REFN**
3	ADC18	1	3	ADC19	0	3	DAC5	DAC4
2	ADC14	1	2	ADC15	0	2	DAC4	DAC5
1	ADC8	1	1	ADC9	0	1	DAC1	DAC0
0	ADC4	1	0	ADC5	0	0	DAC0	DAC1

示例 14.3.1　比较器输入选择示例

检测 PGA2 的输出，并做电压过高和电压过低的检测。

- 将 PGA2 的正端送到比较器 COMP2
- 设置 DAC1 的值，做为过高电压阈值
- 设置 DAC0 的值，做为过低电压阈值

Example 14.3.1

```
void COMP_Example14_3_1(void)
{
COMP->COMP2CTL.bit.INSEL=1;/* Set PGA2P as input to comparator */
COMP->COMP2CTL.bit.PHCMPEN=0;/* disable phase comparator function */
COMP->COMP2CTL.bit.REFSEL=1;/* Set DAC1 as REFP, and DAC0 as REFN */
COMP->COMP2CTL.bit.ENH=1;/* Enable too high comparator */
COMP->COMP2CTL.bit.ENL=1;/* Enable too low comparator */
}
```

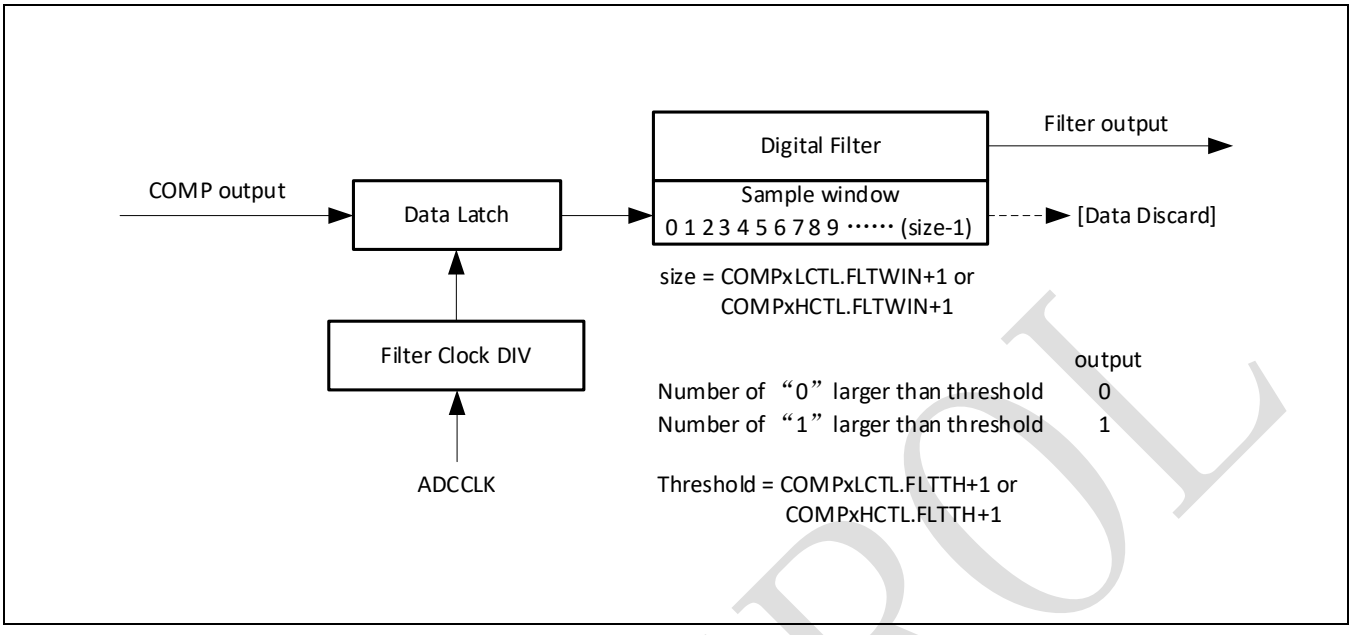
14.4　数字滤波器

比较器的输出被送到数字滤波器的采样窗口，采样时钟来自 ADCCLK，可由 COMPxLCTL.FLTDIV/COMPxHCTL.FLTDIV 来设置分频大小，窗口大小由 COMPxLCTL.FLTWIN/COMPxHCTL.FLTWIN 设置。通过 COMPxLCTL.FLTTH/ COMPxHCTL.FLTTH 设置阈值的大小，数字滤波器可以计算采样窗口内 0 和 1 发生的次数，如果 0 发生的次数大于之前设置的的阈值大小，则数字滤波器的输出会变为 0，同理，如果 1 发生的次数大于阈值，数字滤波器的输出会变为 1。

为了使滤波器工作正常，阈值应分别大于 COMPxLCTL.FLTWIN/2 和 COMPxHCTL.FLTWIN/2。

数字滤波器的行为描述如图 14-2 所示。

图 14-2：数字滤波器行为描述



14.5 上电顺序

当 COMP 开始启动，需要遵循下面的上电顺序：

第一步：使能 ADC 的带隙基准模块（bandgap）

第二步：使能 COMP 内部模拟电路

14.6 寄存器

14.6.1 COMP 寄存器列表

表 14-2：COMP 模块基地址

外设模块	基地址
COMP	0x4000 8C00

表 14-3：COMP 寄存器列表

寄存器	偏移地址	描述	复位值
COMPFLTOUT	0x2F0	比较器滤波输出寄存器	0x00000000
COMPSTS	0x2F4	比较器状态寄存器	0x00000000
COMPSTSCLR	0x2F8	比较器状态清除寄存器	0x00000000
COMP0CTL*	0x2FC	比较器 0 控制寄存器	0x00000000
COMP0LCTL*	0x300	COMP0L 控制寄存器	0x00000088
COMP0HCTL*	0x304	COMP0H 控制寄存器	0x00000088
COMP1CTL*	0x308	比较器 1 控制寄存器	0x00000000
COMP1LCTL*	0x30C	COMP1L 控制寄存器	0x00000088
COMP1HCTL*	0x310	COMP1H 控制寄存器	0x00000088
COMP2CTL*	0x314	比较器 2 控制寄存器	0x00000000
COMP2LCTL*	0x318	COMP2L 控制寄存器	0x00000088
COMP2HCTL*	0x31C	COMP2H 控制寄存器	0x00000088
COMP3CTL*	0x320	比较器 3 控制寄存器	0x00000000
COMP3LCTL*	0x324	COMP3L 控制寄存器	0x00000088
COMP3HCTL*	0x328	COMP3H 控制寄存器	0x00000088
COMP4CTL*	0x32C	比较器 4 控制寄存器	0x00000000
COMP4LCTL*	0x330	COMP4L 控制寄存器	0x00000088
COMP4HCTL*	0x334	COMP4H 控制寄存器	0x00000088
COMP5CTL*	0x338	比较器 5 控制寄存器	0x00000000
COMP5LCTL*	0x33C	COMP5L 控制寄存器	0x00000088
COMP5HCTL*	0x340	COMP5H 控制寄存器	0x00000088
COMP6CTL*	0x344	比较器 6 控制寄存器	0x00000000
COMP6LCTL*	0x348	COMP6L 控制寄存器	0x00000088
COMP6HCTL*	0x34C	COMP6H 控制寄存器	0x00000088
COMP7CTL*	0x350	比较器 7 控制寄存器	0x00000000
COMP7LCTL*	0x354	COMP7L 控制寄存器	0x00000088
COMP7HCTL*	0x358	COMP7H 控制寄存器	0x00000088
DAC0CTL*	0x35C	DAC0 控制寄存器	0x00000002
DAC0CODE	0x360	DAC0 码值寄存器	0x00000000
DAC0CODEA	0x364	DAC0 有效码值寄存器	0x00000000
RAMP0DLY*	0x368	RAMP0 延迟寄存器	0x00000000

寄存器	偏移地址	描述	复位值
RAMP0DLYA	0x36C	RAMP0 有效延迟寄存器	0x00000000
RAMP0DEC*	0x370	RAMP0 递减步长寄存器	0x00000000
RAMP0DECA	0x374	RAMP0 有效递减步长寄存器	0x00000000
RAMP0MAX*	0x378	RAMP0 最大值寄存器	0x00000000
RAMP0MAXA	0x37C	RAMP0 有效最大值寄存器	0x00000000
RAMP0CNT	0x380	RAMP0 计数寄存器	0x00000000
DAC1CTL*	0x384	DAC1 控制寄存器	0x00000002
DAC1CODE	0x388	DAC1 码值寄存器	0x00000000
DAC1CODEA	0x38C	DAC1 有效码值寄存器	0x00000000
RAMP1DLY*	0x390	RAMP1 延迟寄存器	0x00000000
RAMP1DLYA	0x394	RAMP1 有效延迟寄存器	0x00000000
RAMP1DEC*	0x398	RAMP1 递减步长寄存器	0x00000000
RAMP1DECA	0x39C	RAMP1 有效递减步长寄存器	0x00000000
RAMP1MAX	0x3A0	RAMP1 最大值寄存器	0x00000000
RAMP1MAXA	0x3A4	RAMP1 有效最大值寄存器	0x00000000
RAMP1CNT	0x3A8	RAMP1 计数寄存器	0x00000000
DAC2CTL*	0x3AC	DAC2 控制寄存器	0x00000002
DAC2CODE	0x3B0	DAC2 码值寄存器	0x00000000
DAC2CODEA	0x3B4	DAC2 有效码值寄存器	0x00000000
DAC3CTL*	0x3B8	DAC3 控制寄存器	0x00000002
DAC3CODE	0x3BC	DAC3 码值寄存器	0x00000000
DAC3CODEA	0x3C0	DAC3 有效码值寄存器	0x00000000
DAC4CTL*	0x3C4	DAC4 控制寄存器	0x00000002
DAC4CODE	0x3C8	DAC4 码值寄存器	0x00000000
DAC4CODEA	0x3CC	DAC4 有效码值寄存器	0x00000000
DAC5CTL*	0x3D0	DAC5 控制寄存器	0x00000002
DAC5CODE	0x3D4	DAC5 码值寄存器	0x00000000
DAC5CODEA	0x3D8	DAC5 有效码值寄存器	0x00000000
DACBUF0CTL*	0x3F4	DAC Buffer 0 控制寄存器	0x00000000
DACBUF1CTL*	0x3F8	DAC Buffer 1 控制寄存器	0x00000000
COMPREGKEY	0x3FC	COMP 模块写使能寄存器	0x1ACCE551

注意： 由*标识的寄存器仅当 COMPREGKEY=0x1ACCE551 时才可以改写。

14.6.2 COMP 寄存器

表 14-4 到表 14-127 提供了比较器模块寄存器的详细信息。

表 14-4：比较器滤波输出寄存器（COMPFLTOUT）位段定义

COMPFLTOUT (Comparator Filter Output Register) Offset: 0x2F0 Default: 0x00000000 Access: COMP -> COMPFLTOUT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 COMP7H	14 COMP7L	13 COMP6H	12 COMP6L	11 COMP5H	10 COMP5L	9 COMP4H	8 COMP4L
7 COMP3H	6 COMP3L	5 COMP2H	4 COMP2L	3 COMP1H	2 COMP1L	1 COMP0H	0 COMP0L

表 14-5：比较器滤波输出寄存器（COMPFLTOUT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	COMP7H	RO	0x0	COMP7H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
14	COMP7L	RO	0x0	COMP7L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
13	COMP6H	RO	0x0	COMP6H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
12	COMP6L	RO	0x0	COMP6L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
11	COMP5H	RO	0x0	COMP5H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
10	COMP5L	RO	0x0	COMP5L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
9	COMP4H	RO	0x0	COMP4H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
8	COMP4L	RO	0x0	COMP4L 滤波后不带锁存的输出

位段	位段名	属性	复位值	描述
				0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
7	COMP3H	RO	0x0	COMP3H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
6	COMP3L	RO	0x0	COMP3L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
5	COMP2H	RO	0x0	COMP2H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
4	COMP2L	RO	0x0	COMP2L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
3	COMP1H	RO	0x0	COMP1H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
2	COMP1L	RO	0x0	COMP1L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值
1	COMP0H	RO	0x0	COMP0H 滤波后不带锁存的输出 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
0	COMP0L	RO	0x0	COMP0L 滤波后不带锁存的输出 0：输入低于预先设定的低阈值 1：输入高于预先设定的低阈值

表 14-6：比较器状态寄存器（COMPSTS）位段定义

COMPSTS (Comparator Status Register) Offset: 0x2F4 Default: 0x00000000 Access: COMP -> COMPSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 COMP7H	14 COMP7L	13 COMP6H	12 COMP6L	11 COMP5H	10 COMP5L	9 COMP4H	8 COMP4L
7 COMP3H	6 COMP3L	5 COMP2H	4 COMP2L	3 COMP1H	2 COMP1L	1 COMP0H	0 COMP0L

表 14-7：比较器状态寄存器（COMPSTS）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	COMP7H	RO	0x0	锁存的 COMPFLTOUT.COMP7H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
14	COMP7L	RO	0x0	锁存的 COMPFLTOUT.COMP7L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
13	COMP6H	RO	0x0	锁存的 COMPFLTOUT.COMP6H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
12	COMP6L	RO	0x0	锁存的 COMPFLTOUT.COMP6L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
11	COMP5H	RO	0x0	锁存的 COMPFLTOUT.COMP5H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
10	COMP5L	RO	0x0	锁存的 COMPFLTOUT.COMP5L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
9	COMP4H	RO	0x0	锁存的 COMPFLTOUT.COMP4H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
8	COMP4L	RO	0x0	锁存的 COMPFLTOUT.COMP4L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
7	COMP3H	RO	0x0	锁存的 COMPFLTOUT.COMP3H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
6	COMP3L	RO	0x0	锁存的 COMPFLTOUT.COMP3L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
5	COMP2H	RO	0x0	锁存的 COMPFLTOUT.COMP2H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
4	COMP2L	RO	0x0	锁存的 COMPFLTOUT.COMP2L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值

位段	位段名	属性	复位值	描述
3	COMP1H	RO	0x0	锁存的 COMPFLTOUT.COMP1H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
2	COMP1L	RO	0x0	锁存的 COMPFLTOUT.COMP1L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
1	COMP0H	RO	0x0	锁存的 COMPFLTOUT.COMP0H 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值
0	COMP0L	RO	0x0	锁存的 COMPFLTOUT.COMP0L 状态 0：输入低于预先设定的高阈值 1：输入高于预先设定的高阈值

表 14-8：比较器状态清除寄存器（COMPSTSCLR）位段定义

COMPSTSCLR (Comparator Status Clear Register) Offset: 0x2F8 Default: 0x00000000 Access: COMP -> COMPSTSCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 COMP7H	14 COMP7L	13 COMP6H	12 COMP6L	11 COMP5H	10 COMP5L	9 COMP4H	8 COMP4L
7 COMP3H	6 COMP3L	5 COMP2H	4 COMP2L	3 COMP1H	2 COMP1L	1 COMP0H	0 COMP0L

表 14-9：比较器状态清除寄存器（COMPSTSCLR）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	COMP7H	W1C	0x0	清除锁存的 COMP7H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP7H 本位段自动清零 0。
14	COMP7L	W1C	0x0	清除锁存的 COMP7L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP7L 本位段自动清零 0。
13	COMP6H	W1C	0x0	清除锁存的 COMP6H 状态 0：写 0 无效，读总是读回 0

位段	位段名	属性	复位值	描述
				1：写 1 清除 COMPSTS.COMP6H 本位段自动清零 0。
12	COMP6L	W1C	0x0	清除锁存的 COMP6L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP6L 本位段自动清零 0。
11	COMP5H	W1C	0x0	清除锁存的 COMP5H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP5H 本位段自动清零 0。
10	COMP5L	W1C	0x0	清除锁存的 COMP5L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP5L 本位段自动清零 0。
9	COMP4H	W1C	0x0	清除锁存的 COMP4H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP4H 本位段自动清零 0。
8	COMP4L	W1C	0x0	清除锁存的 COMP4L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP4L 本位段自动清零 0。
7	COMP3H	W1C	0x0	清除锁存的 COMP3H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP3H 本位段自动清零 0。
6	COMP3L	W1C	0x0	清除锁存的 COMP3L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP3L 本位段自动清零 0。
5	COMP2H	W1C	0x0	清除锁存的 COMP2H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP2H 本位段自动清零 0。
4	COMP2L	W1C	0x0	清除锁存的 COMP2L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP2L 本位段自动清零 0。

位段	位段名	属性	复位值	描述
3	COMP1H	W1C	0x0	清除锁存的 COMP1H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP1H 本位段自动清零 0。
2	COMP1L	W1C	0x0	清除锁存的 COMP1L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP1L 本位段自动清零 0。
1	COMP0H	W1C	0x0	清除锁存的 COMP0H 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP0H 本位段自动清零 0。
0	COMP0L	W1C	0x0	清除锁存的 COMP0L 状态 0：写 0 无效，读总是读回 0 1：写 1 清除 COMPSTS.COMP0L 本位段自动清零 0。

表 14-10：比较器 0 控制寄存器（COMP0CTL）位段定义

COMP0CTL (Comparator 0 Control Register) Offset: 0x2FC Default: 0x00000000 Access: COMP -> COMP0CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-11：比较器 0 控制寄存器（COMP0CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
8:7	REFSEL	RW	0x0	COMP0 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC2 作为过高参考电平，DAC3 作为过低参考电平 11：DAC3 作为过高参考电平，DAC2 作为过低参考电平 当 PHCMPEN=1 00：PGA0 正端输出 01：PGA0 负端输出 10：ADC1 输入（来自 GPIO1） 11：ADC5 输入（来自 GPIO5）
6:5	INSEL	RW	0x0	COMP0 输入选择 00：PGA0 负端输出 01：PGA0 正端输出

位段	位段名	属性	复位值	描述
				10：ADC0 输入（来自 GPIO0） 11：ADC4 输入（来自 GPIO4）
4:3	HYSTSEL	RW	0x0	COMP0 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP0 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP0H 使能 0：关闭 COMP0H 1：使能 COMP0H
0	ENL	RW	0x0	COMP0L 使能 0：关闭 COMP0L 1：使能 COMP0L

表 14-12：COMP0L 控制寄存器（COMP0LCTL）位段定义

COMP0LCTL (COMP0L Control Register) Offset: 0x300 Default: 0x00000088
Access: COMP -> COMP0LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-13：COMP0L 控制寄存器（COMP0LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP0L 状态 0：PWMSYNC 不影响 COMP0L 状态 1：PWMSYNC 清除锁存的 COMP0L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP0L 输出极性 0：当 COMP0L 事件发生时，输出低电平 1：当 COMP0L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP0L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP0L 输出极性 0：当 COMP0L 事件发生时，输出低电平 1：当 COMP0L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP0L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-14：COMP0H 控制寄存器（COMP0HCTL）位段定义

COMP0HCTL (COMP0H Control Register) Offset: 0x304 Default: 0x00000088 Access: COMP -> COMP0HCTL.all							
31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-15：COMP0H 控制寄存器（COMP0HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP0H 状态 0：PWMSYNC 不影响 COMP0H 状态 1：PWMSYNC 清除锁存的 COMP0H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP0H 输出极性 0：当 COMP0H 事件发生时，输出低电平 1：当 COMP0H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP0H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP0H 输出极性 0：当 COMP0H 事件发生时，输出低电平 1：当 COMP0H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP0H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-16：比较器 1 控制寄存器（COMP1CTL）位段定义

COMP1CTL (Comparator 1 Control Register)　Offset: 0x308　Default: 0x00000000
Access: COMP -> COMP1CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-17：比较器 1 控制寄存器（COMP1CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出

位段	位段名	属性	复位值	描述
8:7	REFSEL	RW	0x0	COMP1 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC2 作为过高参考电平，DAC3 作为过低参考电平 11：DAC3 作为过高参考电平，DAC2 作为过低参考电平 当 PHCMPEN=1 00：PGA1 正端输出 01：PGA1 负端输出 10：ADC2 输入（来自 GPIO2） 11：ADC7 输入（来自 GPIO7）
6:5	INSEL	RW	0x0	COMP1 输入选择 00：PGA1 负端输出 01：PGA1 正端输出 10：ADC0 输入（来自 GPIO0） 11：ADC6 输入（来自 GPIO6）
4:3	HYSTSEL	RW	0x0	COMP1 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP1 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP1H 使能 0：关闭 COMP1H 1：使能 COMP1H
0	ENL	RW	0x0	COMP1L 使能 0：关闭 COMP1L 1：使能 COMP1L

表 14-18：COMP1L 控制寄存器（COMP1LCTL）位段定义

COMP1LCTL (COMP1L Control Register) Offset: 0x30C Default: 0x00000088
Access: COMP -> COMP1LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-19：COMP1L 控制寄存器（COMP1LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP1L 状态 0：PWMSYNC 不影响 COMP1L 状态 1：PWMSYNC 清除锁存的 COMP1L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP1L 输出极性 0：当 COMP1L 事件发生时，输出低电平 1：当 COMP1L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP1L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP1L 输出极性 0：当 COMP1L 事件发生时，输出低电平 1：当 COMP1L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP1L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-20：COMP1H 控制寄存器（COMP1HCTL）位段定义

COMP1HCTL (COMP1H Control Register)　Offset: 0x310　Default: 0x00000088
Access: COMP -> COMP1HCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-21：COMP1H 控制寄存器（COMP1HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP1H 状态 0：PWMSYNC 不影响 COMP1H 状态 1：PWMSYNC 清除锁存的 COMP1H 状态

位段	位段名	属性	复位值	描述
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP1H 输出极性 0：当 COMP1H 事件发生时，输出低电平 1：当 COMP1H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP1H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP1H 输出极性 0：当 COMP1H 事件发生时，输出低电平 1：当 COMP1H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP1H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-22：比较器 2 控制寄存器（COMP2CTL）位段定义

COMP2CTL (Comparator 2 Control Register) Offset: 0x314 Default: 0x00000000 Access: COMP -> COMP2CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-23：比较器 2 控制寄存器（COMP2CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
8:7	REFSEL	RW	0x0	COMP2 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC2 作为过高参考电平，DAC3 作为过低参考电平 11：DAC3 作为过高参考电平，DAC2 作为过低参考电平 当 PHCMPEN=1 00：PGA2 正端输出 01：PGA2 负端输出 10：ADC3 输入（来自 GPIO3） 11：ADC9 输入（来自 GPIO9）
6:5	INSEL	RW	0x0	COMP2 输入选择 00：PGA2 负端输出 01：PGA2 正端输出

位段	位段名	属性	复位值	描述
				10：ADC0 输入（来自 GPIO0） 11：ADC8 输入（来自 GPIO8）
4:3	HYSTSEL	RW	0x0	COMP2 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP2 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP2H 使能 0：关闭 COMP2H 1：使能 COMP2H
0	ENL	RW	0x0	COMP2L 使能 0：关闭 COMP2L 1：使能 COMP2L

表 14-24：COMP2L 控制寄存器（COMP2LCTL）位段定义

COMP2LCTL (COMP2L Control Register)　Offset: 0x318　Default: 0x00000088
Access: COMP -> COMP2LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-25：COMP2L 控制寄存器（COMP2LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP2L 状态 0：PWMSYNC 不影响 COMP2L 状态 1：PWMSYNC 清除锁存的 COMP2L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP2L 输出极性 0：当 COMP2L 事件发生时，输出低电平 1：当 COMP2L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP2L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP2L 输出极性 0：当 COMP2L 事件发生时，输出低电平 1：当 COMP2L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP2L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-26：COMP2H 控制寄存器（COMP2HCTL）位段定义

COMP2HCTL (COMP2H Control Register) Offset: 0x31C Default: 0x00000088 Access: COMP -> COMP2HCTL.all							
31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-27：COMP2H 控制寄存器（COMP2HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP2H 状态 0：PWMSYNC 不影响 COMP2H 状态 1：PWMSYNC 清除锁存的 COMP2H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP2H 输出极性 0：当 COMP2H 事件发生时，输出低电平 1：当 COMP2H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP2H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP2H 输出极性 0：当 COMP2H 事件发生时，输出低电平 1：当 COMP2H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP2H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-28：比较器 3 控制寄存器（COMP3CTL）位段定义

COMP3CTL (Comparator 3 Control Register)　Offset: 0x320　Default: 0x00000000
Access: COMP -> COMP3CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-29：比较器 3 控制寄存器（COMP3CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出

位段	位段名	属性	复位值	描述
8:7	REFSEL	RW	0x0	COMP3 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC4 作为过高参考电平，DAC5 作为过低参考电平 11：DAC5 作为过高参考电平，DAC4 作为过低参考电平 当 PHCMPEN=1 00：PGA3 正端输出 01：PGA3 负端输出 10：ADC11 输入（来自 GPIO11） 11：ADC15 输入（来自 GPIO15）
6:5	INSEL	RW	0x0	COMP3 输入选择 00：PGA3 负端输出 01：PGA3 正端输出 10：ADC10 输入（来自 GPIO10） 11：ADC14 输入（来自 GPIO14）
4:3	HYSTSEL	RW	0x0	COMP3 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP3 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP3H 使能 0：关闭 COMP3H 1：使能 COMP3H
0	ENL	RW	0x0	COMP3L 使能 0：关闭 COMP3L 1：使能 COMP3L

表 14-30：COMP3L 控制寄存器（COMP3LCTL）位段定义

COMP3LCTL (COMP3L Control Register) Offset: 0x324 Default: 0x00000088 Access: COMP -> COMP3LCTL.all							
31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-31：COMP3L 控制寄存器（COMP3LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP3L 状态 0：PWMSYNC 不影响 COMP3L 状态 1：PWMSYNC 清除锁存的 COMP3L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP3L 输出极性 0：当 COMP3L 事件发生时，输出低电平 1：当 COMP3L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP3L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP3L 输出极性 0：当 COMP3L 事件发生时，输出低电平 1：当 COMP3L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP3L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-32：COMP3H 控制寄存器（COMP3HCTL）位段定义

COMP3HCTL (COMP3H Control Register)　Offset: 0x328　Default: 0x00000088
Access: COMP -> COMP3HCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-33：COMP3H 控制寄存器（COMP3HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP3H 状态 0：PWMSYNC 不影响 COMP3H 状态 1：PWMSYNC 清除锁存的 COMP3H 状态

位段	位段名	属性	复位值	描述
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP3H 输出极性 0：当 COMP3H 事件发生时，输出低电平 1：当 COMP3H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP3H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP3H 输出极性 0：当 COMP3H 事件发生时，输出低电平 1：当 COMP3H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP3H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-34：比较器 4 控制寄存器（COMP4CTL）位段定义

COMP4CTL (Comparator 4 Control Register) Offset: 0x32C Default: 0x00000000 Access: COMP -> COMP4CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-35：比较器 4 控制寄存器（COMP4CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
8:7	REFSEL	RW	0x0	COMP4 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC4 作为过高参考电平，DAC5 作为过低参考电平 11：DAC5 作为过高参考电平，DAC4 作为过低参考电平 当 PHCMPEN=1 00：PGA4 正端输出 01：PGA4 负端输出 10：ADC12 输入（来自 GPIO12） 11：ADC17 输入（来自 GPIO17）
6:5	INSEL	RW	0x0	COMP4 输入选择 00：PGA4 负端输出 01：PGA4 正端输出

位段	位段名	属性	复位值	描述
				10：ADC10 输入（来自 GPIO10） 11：ADC16 输入（来自 GPIO16）
4:3	HYSTSEL	RW	0x0	COMP4 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP4 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP4H 使能 0：关闭 COMP4H 1：使能 COMP4H
0	ENL	RW	0x0	COMP4L 使能 0：关闭 COMP4L 1：使能 COMP4L

表 14-36：COMP4L 控制寄存器（COMP4LCTL）位段定义

COMP4LCTL (COMP4L Control Register)　Offset: 0x330　Default: 0x00000088
Access: COMP -> COMP4LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-37：COMP4L 控制寄存器（COMP4LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP4L 状态 0：PWMSYNC 不影响 COMP4L 状态 1：PWMSYNC 清除锁存的 COMP4L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP4L 输出极性 0：当 COMP4L 事件发生时，输出低电平 1：当 COMP4L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP4L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP4L 输出极性 0：当 COMP4L 事件发生时，输出低电平 1：当 COMP4L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP4L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-38：COMP4H 控制寄存器（COMP4HCTL）位段定义

COMP4HCTL (COMP4H Control Register) Offset: 0x334 Default: 0x00000088 Access: COMP -> COMP4HCTL.all							
31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-39：COMP4H 控制寄存器（COMP4HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP4H 状态 0：PWMSYNC 不影响 COMP4H 状态 1：PWMSYNC 清除锁存的 COMP4H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP4H 输出极性 0：当 COMP4H 事件发生时，输出低电平 1：当 COMP4H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP4H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP4H 输出极性 0：当 COMP4H 事件发生时，输出低电平 1：当 COMP4H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP4H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-40：比较器 5 控制寄存器（COMP5CTL）位段定义

COMP5CTL (Comparator 5 Control Register)　Offset: 0x338　Default: 0x00000000
Access: COMP -> COMP5CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-41：比较器 5 控制寄存器（COMP5CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出

位段	位段名	属性	复位值	描述
8:7	REFSEL	RW	0x0	COMP5 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC4 作为过高参考电平，DAC5 作为过低参考电平 11：DAC5 作为过高参考电平，DAC4 作为过低参考电平 当 PHCMPEN=1 00：PGA5 正端输出 01：PGA5 负端输出 10：ADC13 输入（来自 GPIO13） 11：ADC19 输入（来自 GPIO19）
6:5	INSEL	RW	0x0	COMP5 输入选择 00：PGA5 负端输出 01：PGA5 正端输出 10：ADC10 输入（来自 GPIO10） 11：ADC18 输入（来自 GPIO18）
4:3	HYSTSEL	RW	0x0	COMP5 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP5 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP5H 使能 0：关闭 COMP5H 1：使能 COMP5H
0	ENL	RW	0x0	COMP5L 使能 0：关闭 COMP5L 1：使能 COMP5L

表 14-42：COMP5L 控制寄存器（COMP5LCTL）位段定义

COMP5LCTL (COMP5L Control Register) Offset: 0x33C Default: 0x00000088
Access: COMP -> COMP5LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-43：COMP5L 控制寄存器（COMP5LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP5L 状态 0：PWMSYNC 不影响 COMP5L 状态 1：PWMSYNC 清除锁存的 COMP5L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP5L 输出极性 0：当 COMP5L 事件发生时，输出低电平 1：当 COMP5L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP5L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP5L 输出极性 0：当 COMP5L 事件发生时，输出低电平 1：当 COMP5L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP5L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-44：COMP5H 控制寄存器（COMP5HCTL）位段定义

COMP5HCTL (COMP5H Control Register)　Offset: 0x340　Default: 0x00000088
Access: COMP -> COMP5HCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-45：COMP5H 控制寄存器（COMP5HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP5H 状态 0：PWMSYNC 不影响 COMP5H 状态 1：PWMSYNC 清除锁存的 COMP5H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP5H 输出极性 0：当 COMP5H 事件发生时，输出低电平 1：当 COMP5H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP5H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP5H 输出极性 0：当 COMP5H 事件发生时，输出低电平 1：当 COMP5H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP5H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-46：比较器 6 控制寄存器（COMP6CTL）位段定义

COMP6CTL (Comparator 6 Control Register) Offset: 0x344 Default: 0x00000000 Access: COMP -> COMP6CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-47：比较器 6 控制寄存器（COMP6CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
8:7	REFSEL	RW	0x0	COMP6 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC2 作为过高参考电平，DAC3 作为过低参考电平 11：DAC3 作为过高参考电平，DAC2 作为过低参考电平 当 PHCMPEN=1 00：ADC3 输入（来自 GPIO3） 01：ADC7 输入（来自 GPIO7） 10：ADC13 输入（来自 GPIO13） 11：ADC17 输入（来自 GPIO17）
6:5	INSEL	RW	0x0	COMP6 输入选择 00：ADC2 输入（来自 GPIO2） 01：ADC6 输入（来自 GPIO6）

位段	位段名	属性	复位值	描述
				10：ADC12 输入（来自 GPIO12） 11：ADC16 输入（来自 GPIO16）
4:3	HYSTSEL	RW	0x0	COMP6 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP6 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP6H 使能 0：关闭 COMP6H 1：使能 COMP6H
0	ENL	RW	0x0	COMP6L 使能 0：关闭 COMP6L 1：使能 COMP6L

表 14-48：COMP6L 控制寄存器（COMP6LCTL）位段定义

COMP6LCTL (COMP6L Control Register)　Offset: 0x348　Default: 0x00000088
Access: COMP -> COMP6LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-49：COMP6L 控制寄存器（COMP6LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP6L 状态 0：PWMSYNC 不影响 COMP6L 状态 1：PWMSYNC 清除锁存的 COMP6L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP6L 输出极性 0：当 COMP6L 事件发生时，输出低电平 1：当 COMP6L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP6L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP6L 输出极性 0：当 COMP6L 事件发生时，输出低电平 1：当 COMP6L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP6L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-50：COMP6H 控制寄存器（COMP6HCTL）位段定义

COMP6HCTL (COMP6H Control Register) Offset: 0x34C Default: 0x00000088 Access: COMP -> COMP6HCTL.all							
31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-51：COMP6H 控制寄存器（COMP6HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP6H 状态 0：PWMSYNC 不影响 COMP6H 状态 1：PWMSYNC 清除锁存的 COMP6H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP6H 输出极性 0：当 COMP6H 事件发生时，输出低电平 1：当 COMP6H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP6H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP6H 输出极性 0：当 COMP6H 事件发生时，输出低电平 1：当 COMP6H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP6H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-52：比较器 7 控制寄存器（COMP7CTL）位段定义

COMP7CTL (Comparator 7 Control Register)　Offset: 0x350　Default: 0x00000000
Access: COMP -> COMP7CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				SYNCSEL			REFSEL
7	6	5	4	3	2	1	0
REFSEL	INSEL		HYSTSEL		PHCMPEN	ENH	ENL

表 14-53：比较器 7 控制寄存器（COMP7CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:9	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出

位段	位段名	属性	复位值	描述
8:7	REFSEL	RW	0x0	COMP7 参考电平选择 当 PHCMPEN=0 00：DAC0 作为过高参考电平，DAC1 作为过低参考电平 01：DAC1 作为过高参考电平，DAC0 作为过低参考电平 10：DAC4 作为过高参考电平，DAC5 作为过低参考电平 11：DAC5 作为过高参考电平，DAC4 作为过低参考电平 当 PHCMPEN=1 00：ADC5 输入（来自 GPIO5） 01：ADC9 输入（来自 GPIO9） 10：ADC15 输入（来自 GPIO15） 11：ADC19 输入（来自 GPIO19）
6:5	INSEL	RW	0x0	COMP7 输入选择 00：ADC4 输入（来自 GPIO4） 01：ADC8 输入（来自 GPIO8） 10：ADC14 输入（来自 GPIO14） 11：ADC18 输入（来自 GPIO18）
4:3	HYSTSEL	RW	0x0	COMP7 迟滞电压选择 00：0 mV 01：12 mV 10：24 mV 11：36 mV
2	PHCMPEN	RW	0x0	COMP7 相位比较功能使能 0：正常模式 1：相位比较模式
1	ENH	RW	0x0	COMP7H 使能 0：关闭 COMP6H 1：使能 COMP6H
0	ENL	RW	0x0	COMP7L 使能 0：关闭 COMP7L 1：使能 COMP7L

表 14-54：COMP7L 控制寄存器（COMP7LCTL）位段定义

COMP7LCTL (COMP7L Control Register) Offset: 0x354 Default: 0x00000088
Access: COMP -> COMP7LCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-55：COMP7L 控制寄存器（COMP7LCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP7L 状态 0：PWMSYNC 不影响 COMP7L 状态 1：PWMSYNC 清除锁存的 COMP7L 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP7L 输出极性 0：当 COMP7L 事件发生时，输出低电平 1：当 COMP7L 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP7L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

位段	位段名	属性	复位值	描述
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP7L 输出极性 0：当 COMP7L 事件发生时，输出低电平 1：当 COMP7L 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP7L 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-56：COMP7H 控制寄存器（COMP7HCTL）位段定义

COMP7HCTL (COMP7H Control Register)　Offset: 0x358　Default: 0x00000088
Access: COMP -> COMP7HCTL.all

31	30	29	28	27	26	25	24
RESERVED		FLTRST	FLTTH				
23	22	21	20	19	18	17	16
FLTWIN					FLTDIV		
15	14	13	12	11	10	9	8
FLTDIV							SYNCCLREN
7	6	5	4	3	2	1	0
POL4GPIO	SEL4GPIO			POL4PWM	SEL4PWM		

表 14-57：COMP7H 控制寄存器（COMP7HCTL）位段描述

位段	位段名	属性	复位值	描述
31:30	RESERVED_31_30	RO	0x0	保留
29	FLTRST	W1C	0x0	用滤波器输入复位其内部寄存器 0：写 0 无效，读总是读回 0 1：写 1 会用滤波器输入填充其内部寄存器 本位段自动清零。
28:24	FLTTH	RW	0x0	滤波器阈值为（FLTTH+1）个 1 或者 0
23:19	FLTWIN	RW	0x0	滤波器窗口大小为（FLTWIN+1）个采样数据
18:9	FLTDIV	RW	0x0	滤波器时钟分频系数为（FLTDIV+1）

位段	位段名	属性	复位值	描述
8	SYNCCLREN	RW	0x0	使能 PWMSYNC 信号清除锁存的 COMP7H 状态 0：PWMSYNC 不影响 COMP7H 状态 1：PWMSYNC 清除锁存的 COMP7H 状态
7	POL4GPIO	RW	0x1	输出到 GPIO 的 COMP7H 输出极性 0：当 COMP7H 事件发生时，输出低电平 1：当 COMP7H 事件发生时，输出高电平
6:4	SEL4GPIO	RW	0x0	输出到 GPIO 的 COMP7H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出
3	POL4PWM	RW	0x1	输出到 PWM 模块的 COMP7H 输出极性 0：当 COMP7H 事件发生时，输出低电平 1：当 COMP7H 事件发生时，输出高电平
2:0	SEL4PWM	RW	0x0	输出到 PWM 模块的 COMP7H 输出选择 000：原始输出 001：ADC 时钟的同步输出 010：数字滤波器输出 011：锁存的数字滤波器输出 100：原始输出或者锁存的数字滤波器输出 101：ADC 时钟的同步输出或者锁存的数字滤波器输出 110：数字滤波器输出或者锁存的数字滤波器输出 111：原始输出或者 ADC 时钟的同步输出或者锁存的数字滤波器输出

表 14-58：DAC0 控制寄存器（DAC0CTL）位段定义

DAC0CTL (DAC0 Control Register) Offset: 0x35C Default: 0x00000002 Access: COMP -> DAC0CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				DBGRUN		COMPHSEL	
7	6	5	4	3	2	1	0
COMPHSEL	RAMPLOAD	RAMPEN	SYNCSEL			CODELOAD	EN

表 14-59：DAC0 控制寄存器（DAC0CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:10	DBGRUN	RW	0x0	CPU halted 或者 lockup 时的 Ramping 行为 00：立即停止 Ramping 01：在 PWM 同步事件发生时停止 Ramping 1x：自由运行
9:7	COMPHSEL	RW	0x0	COMPH 选择 注意：本位段仅用于 RAMP 功能 000：COMP0H 001：COMP1H 010：COMP2H 011：COMP3H 100：COMP4H 101：COMP5H 110：COMP6H 111：COMP7H
6	RAMPLOAD	RW	0x0	DAC Ramp 加载模式 0：Shadow 模式，RAMP 计数器从影子寄存器中加载 RAMPMAX 1：直接模式，RAMP 计数器直接加载最新的 RAMPMAX
5	RAMPEN	RW	0x0	DAC Ramping 使能 0：关闭 Ramping 1：使能 Ramping。DAC0CODEA 的值等于 RAMP0CNT 的高 10 位。
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出

位段	位段名	属性	复位值	描述
				011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC0CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-60：DAC0 码值寄存器（DAC0CODE）位段定义

DAC0CODE (DAC0 Code Register) Offset: 0x360 Default: 0x00000000
Access: COMP -> DAC0CODE.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-61：DAC0 码值寄存器（DAC0CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	VAL	RW	0x0	DAC 码值 当 DAC0CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC0CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-62：DAC0 有效码值寄存器（DAC0CODEA）位段定义

DAC0CODEA (DAC0 Active Code Register) Offset: 0x364 Default: 0x00000000 Access: COMP -> DAC0CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-63：DAC0 有效码值寄存器（DAC0CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-64：RAMP0 延迟寄存器（RAMP0DLY）位段定义

RAMP0DLY (RAMP0 Delay Shadow Register) Offset: 0x368 Default: 0x00000000 Access: COMP -> RAMP0DLY.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-65：RAMP0 延迟寄存器（RAMP0DLY）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	从 PWMSYNC 到 Ramping 开始的延迟时间（shadow）

表 14-66：RAMP0 有效延迟寄存器（RAMP0DLYA）位段定义

RAMP0DLYA (RAMP0 Delay Active Register) Offset: 0x36C Default: 0x00000000 Access: COMP -> RAMP0DLYA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-67：RAMP0 有效延迟寄存器（RAMP0DLYA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	从 PWMSYNC 到 Ramping 开始的延迟时间（active）

表 14-68：RAMP0 递减步长寄存器（RAMP0DEC）位段定义

RAMP0DEC (RAMP0 Decrement Shadow Register) Offset: 0x370 Default: 0x00000000 Access: COMP -> RAMP0DEC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-69：RAMP0 递减步长寄存器（RAMP0DEC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	Ramping 减少的步长（shadow）

表 14-70：RAMP0 有效递减步长寄存器（RAMP0DECA）位段定义

RAMP0DECA (RAMP0 Decrement Active Register) Offset: 0x374 Default: 0x00000000 Access: COMP -> RAMP0DECA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-71：RAMP0 有效递减步长寄存器（RAMP0DECA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	Ramping 减少的步长（active）

表 14-72：RAMP0 最大值寄存器（RAMP0MAX）位段定义

RAMP0MAX (RAMP0 Maximum Value Shadow Register) Offset: 0x378 Default: 0x00000000 Access: COMP -> RAMP0MAX.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-73：RAMP0 最大值寄存器（RAMP0MAX）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	Ramping 复位时的初始最大值（shadow）

表 14-74：RAMP0 有效最大值寄存器（RAMP0MAXA）位段定义

RAMP0MAXA (RAMP0 Maximum Value Active Register) Offset: 0x37C Default: 0x00000000 Access: COMP -> RAMP0MAXA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-75：RAMP0 有效最大值寄存器（RAMP0MAXA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	Ramping 复位时的初始最大值（active）

表 14-76：RAMP0 计数值寄存器（RAMP0CNT）位段定义

RAMP0CNT (RAMP0 Count Register) Offset: 0x380 Default: 0x00000000 Access: COMP -> RAMP0CNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-77：RAMP0 计数值寄存器（RAMP0CNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	RAMP0 计数值

表 14-78：DAC1 控制寄存器（DAC1CTL）位段定义

DAC1CTL (DAC1 Control Register) Offset: 0x384 Default: 0x00000002
Access: COMP -> DAC1CTL.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				DBGRUN		COMPHSEL	
7	6	5	4	3	2	1	0
COMPHSEL	RAMPLOAD	RAMPEN	SYNCSEL			CODELOAD	EN

表 14-79：DAC1 控制寄存器（DAC1CTL）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11:10	DBGRUN	RW	0x0	CPU halted 或者 lockup 时的 Ramping 行为 00：立即停止 Ramping 01：在 PWM 同步事件发生时停止 Ramping 1x：自由运行
9:7	COMPHSEL	RW	0x0	COMPH 选择 注意：本位段仅用于 RAMP 功能 000：COMP0H 001：COMP1H 010：COMP2H 011：COMP3H 100：COMP4H 101：COMP5H 110：COMP6H 111：COMP7H
6	RAMPLOAD	RW	0x0	DAC Ramp 加载模式 0：Shadow 模式，RAMP 计数器从影子寄存器中加载 RAMPMAX 1：直接模式，RAMP 计数器直接加载最新的 RAMPMAX
5	RAMPEN	RW	0x0	DAC Ramping 使能 0：关闭 Ramping 1：使能 Ramping。DAC1CODEA 的值等于 RAMP1CNT 的高 10 位。
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出

位段	位段名	属性	复位值	描述
				011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC1CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-80：DAC1 码值寄存器（DAC1CODE）位段定义

DAC1CODE (DAC1 Code Register) Offset: 0x388 Default: 0x00000000
Access: COMP -> DAC1CODE.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-81：DAC1 码值寄存器（DAC1CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	VAL	RW	0x0	DAC 码值 当 DAC1CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC1CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-82：DAC1 有效码值寄存器（DAC1CODEA）位段定义

DAC1CODEA (DAC1 Active Code Register) Offset: 0x38C Default: 0x00000000 Access: COMP -> DAC1CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-83：DAC1 有效码值寄存器（DAC1CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-84：RAMP1 延迟寄存器（RAMP1DLY）位段定义

RAMP1DLY (RAMP1 Delay Shadow Register) Offset: 0x390 Default: 0x00000000 Access: COMP -> RAMP1DLY.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-85：RAMP1 延迟寄存器（RAMP1DLY）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	从 PWMSYNC 到 Ramping 开始的延迟时间（shadow）

表 14-86：RAMP1 有效延迟寄存器（RAMP1DLYA）位段定义

RAMP1DLYA (RAMP1 Delay Active Register) Offset: 0x394 Default: 0x00000000 Access: COMP -> RAMP1DLYA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-87：RAMP1 有效延迟寄存器（RAMP1DLYA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	从 PWMSYNC 到 Ramping 开始的延迟时间（active）

表 14-88：RAMP1 递减步长寄存器（RAMP1DEC）位段定义

RAMP1DEC (RAMP1 Decrement Shadow Register) Offset: 0x398 Default: 0x00000000 Access: COMP -> RAMP1DEC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-89：RAMP1 递减步长寄存器（RAMP1DEC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	Ramping 减少的步长（shadow）

表 14-90：RAMP1 有效递减步长寄存器（RAMP1DECA）位段定义

RAMP1DECA (RAMP1 Decrement Active Register) Offset: 0x39C Default: 0x00000000 Access: COMP -> RAMP1DECA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-91：RAMP1 有效递减步长寄存器（RAMP1DECA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	Ramping 减少的步长（active）

表 14-92：RAMP1 最大值寄存器（RAMP1MAX）位段定义

RAMP1MAX (RAMP1 Maximum Value Shadow Register) Offset: 0x3A0 Default: 0x00000000 Access: COMP -> RAMP1MAX.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-93：RAMP1 最大值寄存器（RAMP1MAX）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x0	Ramping 复位时的初始最大值（shadow）

表 14-94：RAMP1 有效最大值寄存器（RAMP1MAXA）位段定义

RAMP1MAXA (RAMP1 Maximum Value Active Register) Offset: 0x3A4 Default: 0x00000000 Access: COMP -> RAMP1MAXA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-95：RAMP1 有效最大值寄存器（RAMP1MAXA）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	Ramping 复位时的初始最大值（active）

表 14-96：RAMP1 计数值寄存器（RAMP1CNT）位段定义

RAMP1CNT (RAMP1 Count Register) Offset: 0x3A8 Default: 0x00000000 Access: COMP -> RAMP1CNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 14-97：RAMP1 计数值寄存器（RAMP1CNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RO	0x0	RAMP1 计数值

表 14-98：DAC2 控制寄存器（DAC2CTL）位段定义

DAC2CTL (DAC2 Control Register) Offset: 0x3AC Default: 0x00000002 Access: COMP -> DAC2CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			SYNCSEL			CODELOAD	EN

表 14-99：DAC2 控制寄存器（DAC2CTL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC2CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-100：DAC2 码值寄存器（DAC2CODE）位段定义

DAC2CODE (DAC2 Code Register) Offset: 0x3B0 Default: 0x00000000 Access: COMP -> DAC2CODE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-101：DAC2 码值寄存器（DAC2CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保持
9:0	VAL	RW	0x0	DAC 码值 当 DAC2CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC2CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-102：DAC2 有效码值寄存器（DAC2CODEA）位段定义

DAC2CODEA (DAC2 Active Code Register) Offset: 0x3B4 Default: 0x00000000 Access: COMP -> DAC2CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-103：DAC2 有效码值寄存器（DAC2CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-104：DAC3 控制寄存器（DAC3CTL）位段定义

DAC3CTL (DAC3 Control Register) Offset: 0x3B8 Default: 0x00000002 Access: COMP -> DAC3CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			SYNCSEL			CODELOAD	EN

表 14-105：DAC3 控制寄存器（DAC3CTL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 100：PWM6 同步输出 101：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC3CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-106：DAC3 码值寄存器（DAC3CODE）位段定义

DAC3CODE (DAC3 Code Register) Offset: 0x3BC Default: 0x00000000 Access: COMP -> DAC3CODE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-107：DAC3 码值寄存器（DAC3CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	VAL	RW	0x0	DAC 码值 当 DAC3CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC3CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-108：DAC3 有效码值寄存器（DAC3CODEA）位段定义

DAC3CODEA (DAC3 Active Code Register) Offset: 0x3C0 Default: 0x00000000 Access: COMP -> DAC3CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-109：DAC3 有效码值寄存器（DAC3CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-110：DAC4 控制寄存器（DAC4CTL）位段定义

DAC4CTL (DAC4 Control Register) Offset: 0x3C4 Default: 0x00000002 Access: COMP -> DAC4CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			SYNCSEL			CODELOAD	EN

表 14-111：DAC4 控制寄存器（DAC4CTL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 110：PWM6 同步输出 111：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC4CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-112：DAC4 码值寄存器（DAC4CODE）位段定义

DAC4CODE (DAC4 Code Register) Offset: 0x3C8 Default: 0x00000000 Access: COMP -> DAC4CODE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-113：DAC4 码值寄存器（DAC4CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保持
9:0	VAL	RW	0x0	DAC 码值 当 DAC4CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC4CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-114：DAC4 有效码值寄存器（DAC4CODEA）位段定义

DAC4CODEA (DAC4 Active Code Register) Offset: 0x3CC Default: 0x00000000 Access: COMP -> DAC4CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-115：DAC4 有效码值寄存器（DAC4CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-116：DAC5 控制寄存器（DAC5CTL）位段定义

DAC5CTL (DAC5 Control Register) Offset: 0x3D0 Default: 0x00000002 Access: COMP -> DAC5CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			SYNCSEL			CODELOAD	EN

表 14-117：DAC5 控制寄存器（DAC5CTL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:2	SYNCSEL	RW	0x0	PWM 同步输出源选择 000：PWM0 同步输出 001：PWM1 同步输出 010：PWM2 同步输出 011：PWM3 同步输出 100：PWM4 同步输出 101：PWM5 同步输出 100：PWM6 同步输出 101：PWM7 同步输出
1	CODELOAD	RW	0x1	DAC 码值加载模式 0：DAC 码值在 PWMSYNC 事件时更新 1：DAC 码值在写 DAC5CODE 时立即更新
0	EN	RW	0x0	DAC 使能 0：关闭 1：使能

表 14-118：DAC5 码值寄存器（DAC5CODE）位段定义

DAC5CODE (DAC5 Code Register) Offset: 0x3D4 Default: 0x00000000 Access: COMP -> DAC5CODE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-119：DAC5 码值寄存器（DAC5CODE）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	VAL	RW	0x0	DAC 码值 当 DAC5CTL.CODELOAD=0 时，写本位段，只会影响影子码值（shadow code）。 当 DAC5CTL.CODELOAD=1 时，写本位段，会同时影响影子码值（shadow code）和有效码值（active code）。

表 14-120：DAC5 有效码值寄存器（DAC5CODEA）位段定义

DAC5CODEA (DAC5 Active Code Register) Offset: 0x3D8 Default: 0x00000000 Access: COMP -> DAC5CODEA.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 14-121：DAC5 有效码值寄存器（DAC5CODEA）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	CODEA	RO	0x0	有效的 DAC 码值

表 14-122：DAC Buffer 0 控制寄存器（DACBUF0CTL）位段定义

DACBUF0CTL (DAC Buffer 0 Control Register) Offset: 0x3F4 Default: 0x00000000 Access: COMP -> DACBUF0CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				OE	INSEL		EN

表 14-123：DAC Buffer 0 控制寄存器（DACBUF0CTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	OE	RW	0x0	输出到 GPIO10 管脚使能 0：关闭 1：使能
2:1	INSEL	RW	0x0	DAC buffer 0 输入选择 00：DAC0 01：DAC1 10：DAC2 11：DAC3
0	EN	RW	0x0	DAC buffer 0 使能 0：关闭 1：使能

表 14-124：DAC Buffer 1 控制寄存器（DACBUF1CTL）位段定义

DACBUF1CTL (DAC Buffer 1 Control Register) Offset: 0x3F8 Default: 0x00000000 Access: COMP -> DACBUF1CTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED				OE	INSEL		EN

表 14-125：DAC Buffer 1 控制寄存器（DACBUF1CTL）位段描述

位段	位段名	属性	复位值	描述
31:4	RESERVED_31_4	RO	0x0	保留
3	OE	RW	0x0	输出到 GPIO13 管脚使能 0：关闭 1：使能
2:1	INSEL	RW	0x0	DAC buffer 1 输入选择 00：DAC0 01：DAC1 10：DAC4 11：DAC5
0	EN	RW	0x0	DAC buffer 1 使能 0：关闭 1：使能

表 14-126：COMP 模块写使能寄存器（COMPREGKEY）位段定义

COMPREGKEY (COMP Register Write-Allow Key Register) Offset: 0x3FC Default: 0x1ACCE551 Access: COMP -> COMPREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 14-127：COMP 模块写使能寄存器（COMPREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 COMP 寄存器

15 数模转换器（DAC）

15.1 DAC 概述

SPC2168 包含 6 个 10 位 DAC 和 2 个可以驱动容性负载的缓冲器（Buffer）。DAC 输出可以配置成直流电压和交流波形，如三角波、方波等。DAC 的输出可由下面表达式计算：

$$\mathrm{VOUT} = \frac{AVDD}{2^{10}} * CODE \quad (AVDD\text{典型值为 } 3.3V)$$

15.2 DAC 结构

DAC 和 DAC Buffer 的框图如图 15-1 所示。

DAC 有两组寄存器：CODE 和 CODEA，其中 CODEA 是只读寄存器，可以直接控制 10 比特 DAC 的输出。CODE 是用户可写的影子（shadow）寄存器，其值可以立即或者由下一个 PWMSYNC 事件同步，再送到 CODEA 中，这取决于 DACxCTL.CODELOAD 的设置。如果 DACxCTL.CODELOAD=0（影子模式），用户改变 CODE 值，要等到下一个 PWMSYNC 信号来临后，才能传送到 CODEA 中。如果 DACxCTL.CODELOAD=1（直接模式），用户改变 CODE 值，会直接送到 CODEA 中，DAC 的输出也会立即变化。

比较器电压过高阈值和电压过低阈值分别由两个 DAC 产生，如何选择 DAC 输入请参考表 14-1。

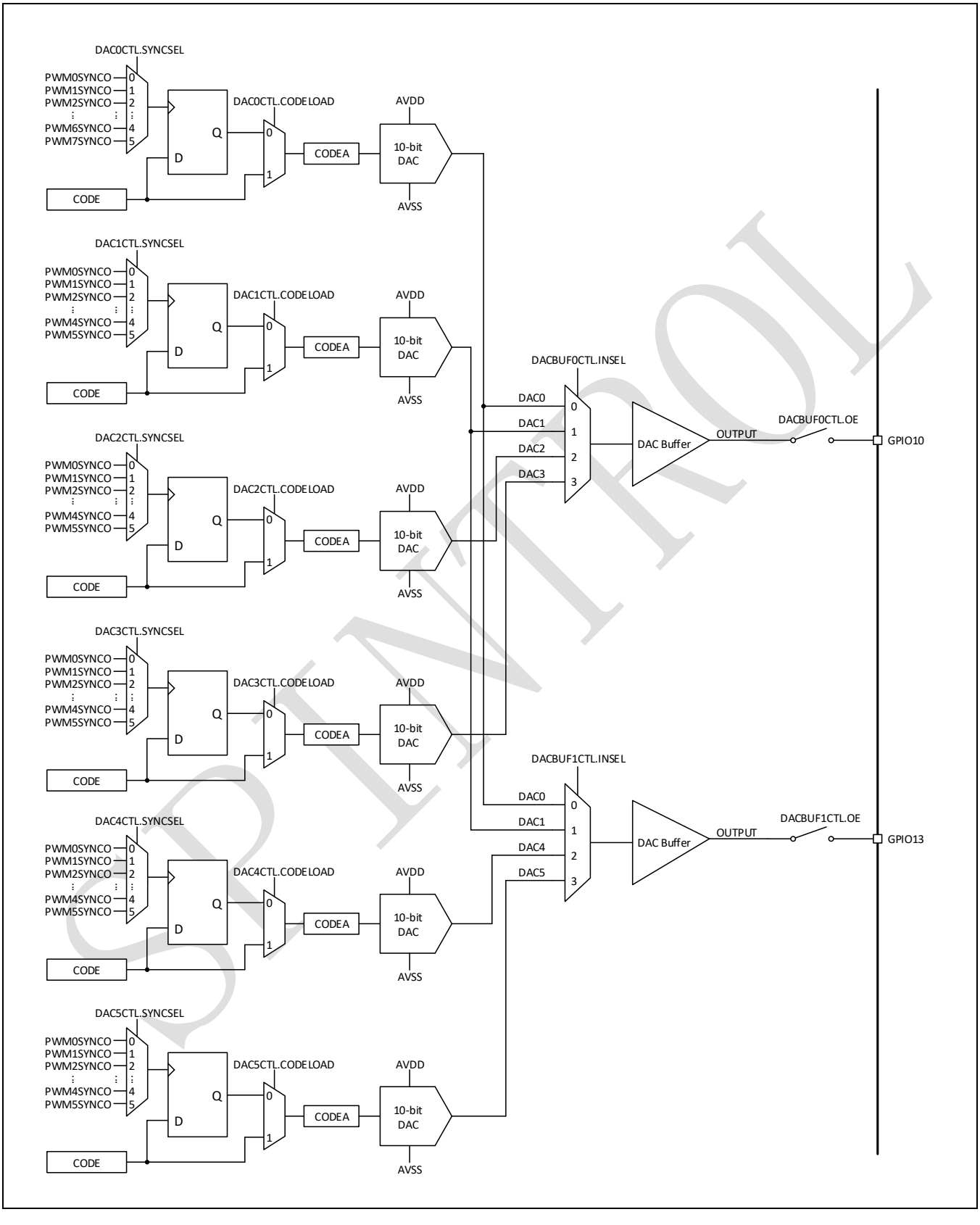
SPC2168 集成了两个 DAC Buffer，每个 DAC 的输出均可以被送到 DAC Buffer 的输入，这取决于 DACBUFCTL.INSEL 的配置，如表 15-1 所示。

表 15-1：DAC Buffer 输入选择

DACBUF0CTL.INSEL	DAC buffer 0 输出	DACBUF1CTL.INSEL	DAC buffer 1 输出
0	DAC0	0	DAC0
1	DAC1	1	DAC1
2	DAC2	2	DAC4
3	DAC3	3	DAC5

DAC Buffer 的输出可以直接送到 ADC 中（如表 12-4 所示），DAC Buffer 的输出也可以送到芯片的 ADC10 或者 ADC13 管脚，分别由 DACBUF0CTL.OE 和 DACBUF0CTL.OE 来配置。

图 15-1：DAC 和 DAC Buffer 框图



示例 15.2.1　DAC 和 DAC buffer 配置

设置 DAC0 输出为 1.65V（假如 AVDD=3.3V），并送到 DAC Buffer，再用 ADC 去测量 Buffer 输出，并且把 Buffer 输出送到 ADC10 管脚。

- 使能 DAC0，设置输出为 1.65V
- 使能 DAC buffer，并且选择 DAC0 做为输入
- 设置 ADC 输入选择器，选择 DAC Buffer 做为输入
- 把 DAC buffer 的输出送到 ADC10 管脚

Example 15.2.1

```
Void COMP_Example15_2_1(void)
{
COMP->DAC0CODE.all=512;/* Set DAC0 output to 1.65V */
COMP->DAC0CTL.bit.CODELOAD=1;/* direct mode, dac code immediately update */
COMP->DAC0CTL.bit.EN=1;/* Enable DAC0 */
COMP->DACBUF0CTL.bit.INSEL=0;/* Send DAC0 to DAC buffer input */
ADC->ADCBGCTL.bit.EN=1;/* Enable ADC Bandgap */
COMP->DACBUF0CTL.bit.EN=1;/* Enable DAC buffer */
ADC->ADCSOCCTL[0].bit.SHEN=2;/* Enable Sampler B */
ADC->ADCSOCCTL[0].bit.CHSELP=3;/* Sampler B Positive =    DAC Buffer 0 */
ADC->ADCSOCCTL[0].bit.CHSELN=0;/* Sampler B Negative =    GND */
COMP->DACBUF0CTL.bit.OE=1;/* Send DAC buffer output to ADC10 pin */
}
```

15.3 Ramp 发生器

Ramp 发生器可以为过高电压比较器产生一个下降的斜坡参考电压（REFP），如图 14-1 中所示。只有 DAC0 和 DAC1 有这个功能。

图 15-2 为 Ramp 发生器的框图。Ramp 功能有两组寄存器，一组是只读寄存器：RAMPDLYA、RAMPDECA 和 RAMPMAXA，它们可以直接控制 Ramp 功能；另外一组是可写影子（shadow）寄存器：RAMPDLY、RAMPDEC 和 RAMPMAX。这些可写影子寄存器，可以在某些情况下传送到只读寄存器中，例如 RAMPEN 或者 PWMSYNC 的上升沿，如图 15-2 所示。

RAMPCNT 的高 10 位会传送到 CODEA，用来控制 DAC 的输出。RAMPMAXA 为最初的 DAC 值，RAMPDLYA 控制延迟计数器。每一个 PWMCLK 周期，RAMPDLYA 会自动减一，直到减少到 0。RAMPDECA 控制 DAC 减少的步长，每当 RAMPDLYA 减少到 0，RAMPCNT 会自动减少 RAMPDECA。

当使能 Ramp 发生器（RAMPEN=1）后，RAMPCNT 首先读取 RAMPMAX 的数值，在 PWMSYNC 触发之前，该数值保持不变。每个 PWMCLK，RAMPDLYA 自动减少 1。当减少到 0，RAMPCNT 将会减少 RAMPDECA。之后 RAMPDLYA 会在下一个 PWMCLK 重新读取初始值，重复之前步骤。所以，DAC 的输出会在每（RAMPDLYA+1）个 PWMCLK 的周期下，减少 RAMPDECA。

如果 RAMPLOAD=0，Ramp 发生器处于 shadow 模式。当 COMPxH 有效时，RAMPMAX 不会立即传送到 RAMPMAXA，RAMPCNT 会由之前的 RAMPMAXA 来控制，在下个 PWMSYNC 的上升沿，更新后的 RAMPMAX 才会传送到 RAMPMAXA。如果 RAMPLOAD=1，Ramp 发生器处于直接模式。当 COMPxH 有效时，RAMPMAX 值会直接传送到 RAMPMAXA 中，并控制 RAMPCNT。

图 15-2：Ramp 发生器框图

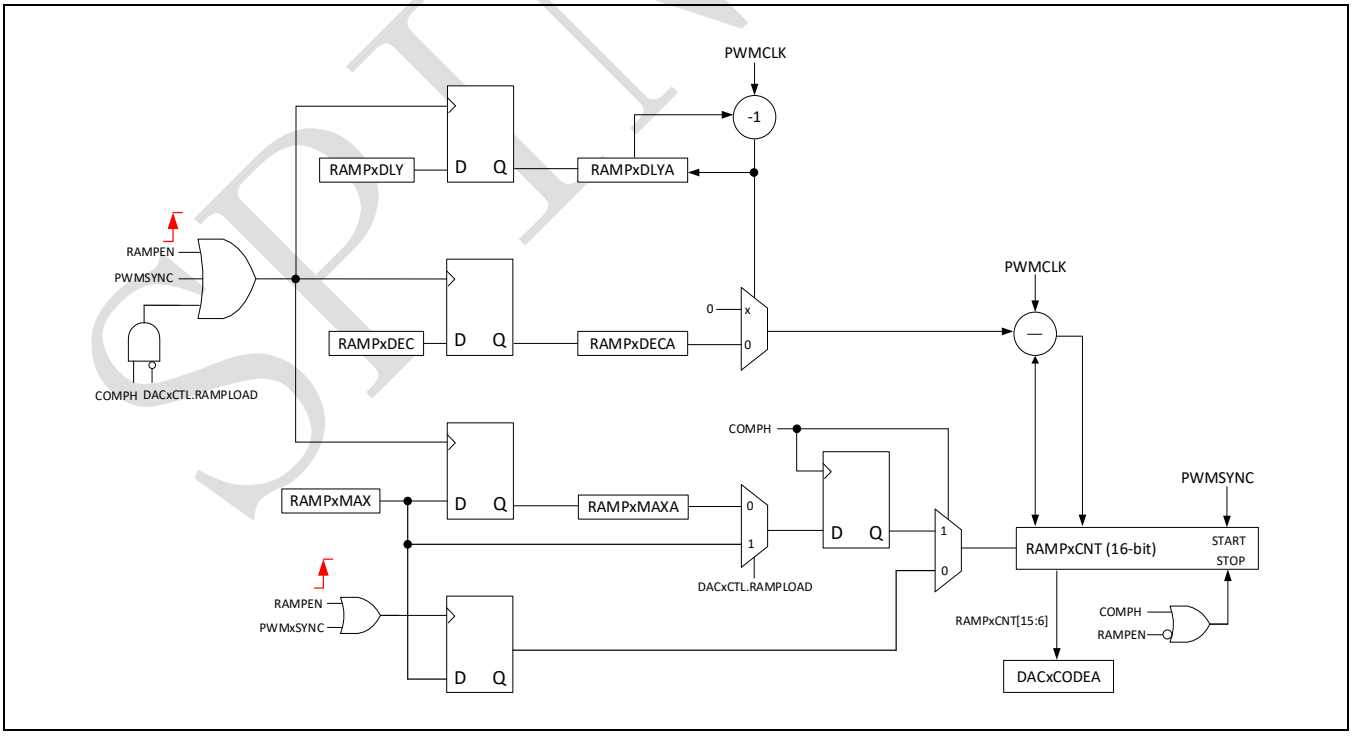
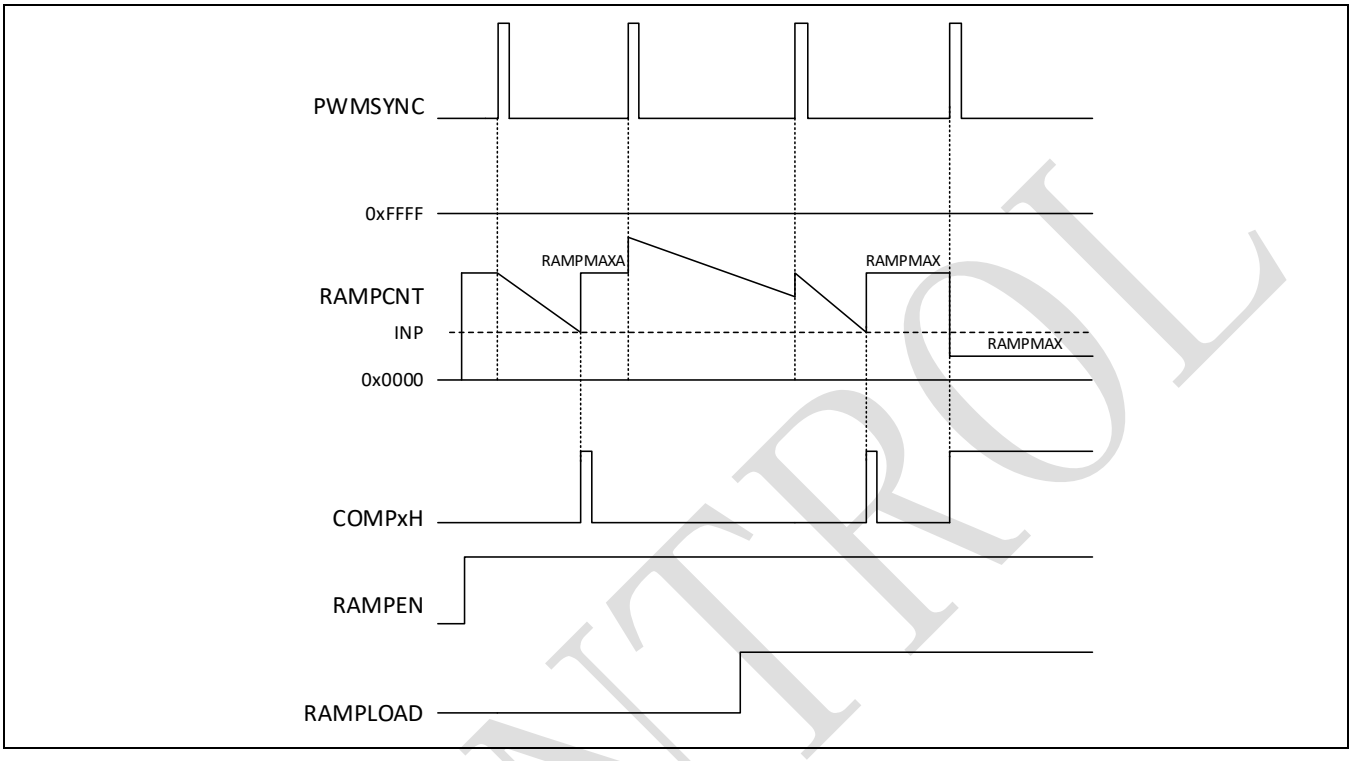


图 15-3 显示了 DAC 输出（由 RAMPCNT 控制）在 PWMSYNC 触发和 COMPxH 发生时的波形，假设 COMPxH 的正端输入是固定值（INP 为固定值）。

图 15-3：Ramp 发生器波形



15.4 上电顺序

当 DAC 或 DAC Buffer 开始启动，需要遵循下面的上电顺序：

第一步：使能 ADC 的带隙基准模块（bandgap）

第二步：使能 DAC 或 DAC Buffer 内部模拟电路

15.5 寄存器

关于 DAC 寄存器的定义，请参考章节 14.6。

16 温度传感器（T-Sensor）

16.1 T-Sensor 概述

SPC2168 实现了一个内部温度传感器用于测量芯片的结温。温度传感器被接入 ADC 采样保持器 C 的选择通道，如图 12-1 所示。因此，可以通过 ADC 对温度进行测量。

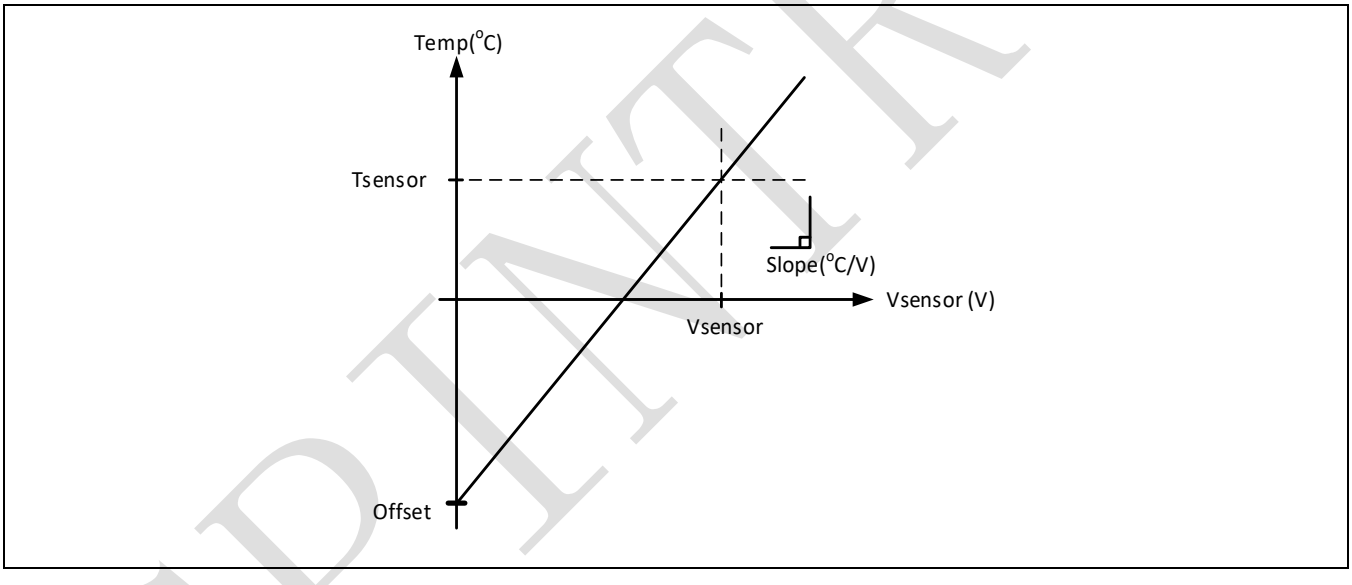
16.2 转换函数

温度传感器输出电压和温度成正比。如图 16-1 所示，温度可以用下列公式计算：

$$\text{Temperature} = \text{Slope} \times \text{Vsensor} + \text{Offset}$$

其中，Slope = 4238.97 ºC/V，Offset = -283.36 ºC。

图 16-1：温度传感器转换曲线



因此，用温度传感器测量温度的步骤如下：

第一步，测量温度传感器的输出电压 Vsensor，这个电压可以通过 ADC 测量结果换算后得到。需要注意的是，ADC 的测量结果需要使用将 ADC 自身的增益和失调误差校准后的结果。

第二步，用上面的公式和参数计算得到当前温度值。

注意： 1 个 ADC 码值对应 1.9ºC。为了满足温度传感器稳定时间的要求，ADC 采样时间至少应为 2us。SDK 中提供的函数 ADC_CalculateTemperature()可以直接得到温度值。

16.3 上电顺序

当 T-Sensor 开始启动，需要遵循下面的上电顺序：

第一步：使能 ADC 的带隙基准模块（bandgap）

第二步：等待 200 微秒以后，使能 ADC 的基准电压驱动模块

第三步：等待 200 微秒以后，使能 ADC 内部模拟电路

16.4 精度提升

温度传感器的准确度将随着内部器件固有的失配而退化。由于温度传感器总是用于过温保护场景，不需要高精度，因此，在出厂的时候，温度传感器没有被标定。

然而，SPC2168 使用动态元件匹配（Dynamic Element Matching, DEM）来修正温度传感器的失配，从而让温度传感器获得更好的精度。SDK 中提供的函数 ADC_CalculateTemperature()已经采用该方法获取温度值。

16.5 寄存器

温度传感器相关寄存器的定义请参见表 12-356 和表 12-357。

17 通用异步收发器（UART）

17.1 UART 概述

SPC2168 实现了一个通用异步收发器（UART）端口。这个端口包含了一个 UART 和一个低速串行红外发送编码器和接收解码器，符合 IrDA 串行红外规范。

UART 端口用来执行：对从外围设备接收到的数据字符进行串到并的转换，对从 CPU 接收到的数据字符执行并到串的转换。

CPU 能够从线路状态寄存器（UARTLSR）中读到一个完整的 UART 状态。状态信息包括传输的类型和状态、错误状态 - 如奇偶校验，溢出，帧中断或者间断（break）中断。

UART 端口可以工作在 FIFO 模式下，也可以工作在非 FIFO 模式下。在 FIFO 模式下，一个 64 字节的 TXFIFO（Transmit FIFO）保存来自 CPU 的数据直到它被发送到串行链路上；一个 64 字节的 RXFIFO（Receive FIFO）缓存来自串行链路上的数据直到它被 CPU 读取。在非 FIFO 模式下，TXFIFO 和 RXFIFO 是被绕过的，使用 THR 寄存器（Transmit Holding Register）和 RBR 寄存器（Receive Buffer Register）作为替代。

UART 包括一个可编程的波特率产生器，能够对输入时钟进行 1~（2^{16}-1）分频。它能够产生一个 16 倍的时钟信号来能够驱动内部发送和接收逻辑。软件可以通过编写中断来满足它的需求，这样可以最小化处理通信链路所需的计算量。UART 的工作既可以通过软件轮询来控制也可以通过中断来驱动。

UART 端口支持波特率有：9600，19.2K，38.4K，57.6K，115.2K，230K，460K，921K，1.8M 以及 3.6M。

17.2 UART 主要特性

UART 模块具有以下特性：

- 在串行数据上添加或者删除标准异步通信位（开始位，停止位，校验位）
- 独立控制的发送、接收、线路状态和数据集（data-set）中断
- 可编程串行接口
  - 5 - 8 个数据位
  - 偶检验、奇校验或者无校验检测
  - 支持 1 个、1.5 个以及 2 个停止位生成
  - 最大支持 12.5 Mbps 波特率
  - 错误的开始位检测
- 64 字节 TXFIFO
- 64 字节 RXFIFO
- 完备的状态报告功能
- 产生和检测线路间断（break）的功能
- 内部诊断功能，包括：

- 回环（loopback）控制：用于通信链路故障隔离
- 间断、检验位以及帧错误仿真

– 完全优先处理的中断系统控制

– 串行红外异步接口，符合红外数据协会（IrDA）的标准

17.3 UART 信号描述

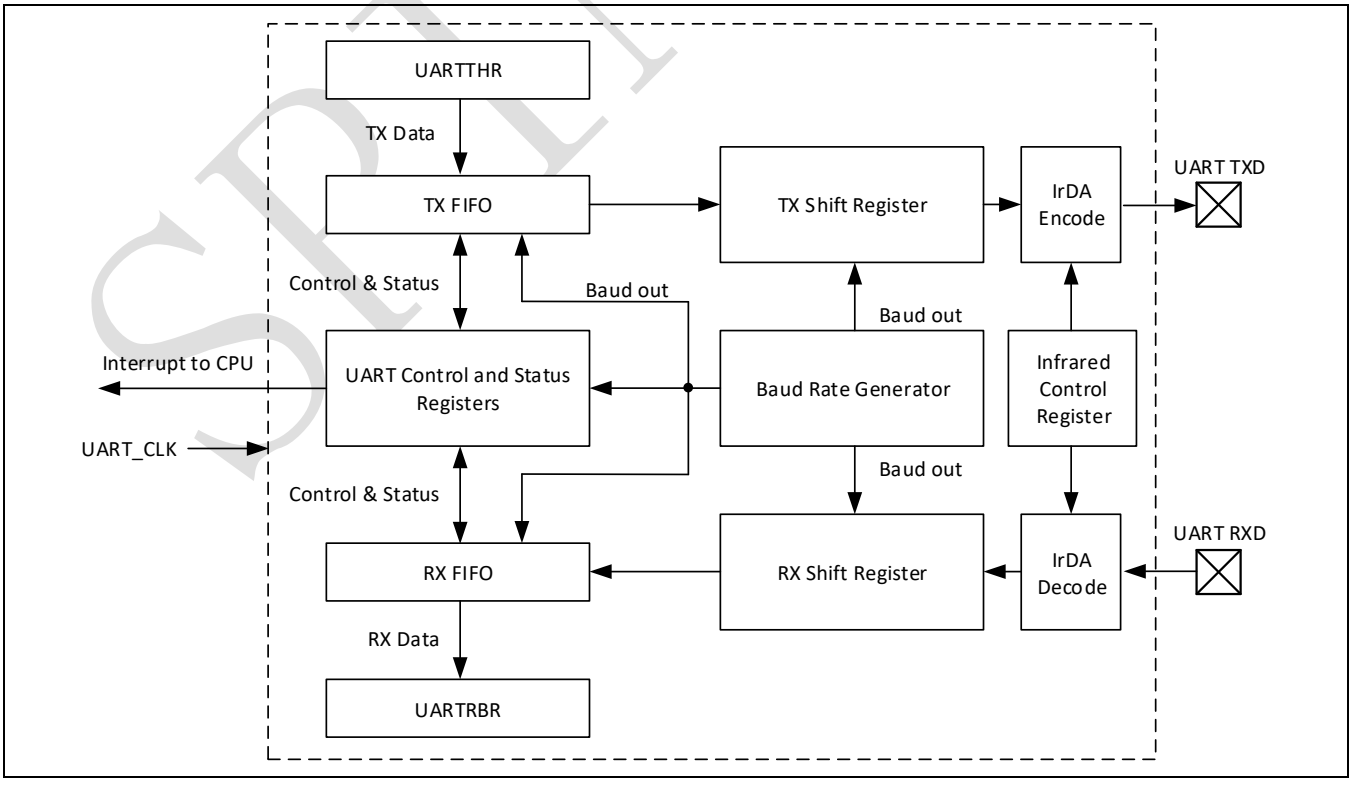
表 17-1 列出并描述了连接到 UART 模块的外部信号。传输数字 CMOS 电平信号的引脚是通过 GPIO 连接起来的。更多详细信息，请参考 GPIO 引脚配置寄存器。

表 17-1：UART 信号描述

名称	类型	描述
RXD	输入	**串行输入** 接收移位寄存器（Receive Shift register）的串行数据输入。在红外模式下，该信号被连接到红外接收器的输入端。
TXD	输出	**串行输出** 串行数据输出，复位操作后，TXD 信号被设置为逻辑“1”。在红外模式下，该信号被连接到红外发射器的输出端。

UART 模块的框图如图 17-1 所示。

图 17-1：UART 模块框图



17.4 UART 操作

图 17-2 展示了一个 UART 数据帧的格式。

图 17-2：UART 数据帧格式示例

Start Bit	Data <0>	Data <1>	Data <2>	Data <3>	Data <4>	Data <5>	Data <6>	Data <7>	Parity Bit	Stop Bit 1	Stop Bit 2
TXD or RXD pin											
	LSB							MSB			

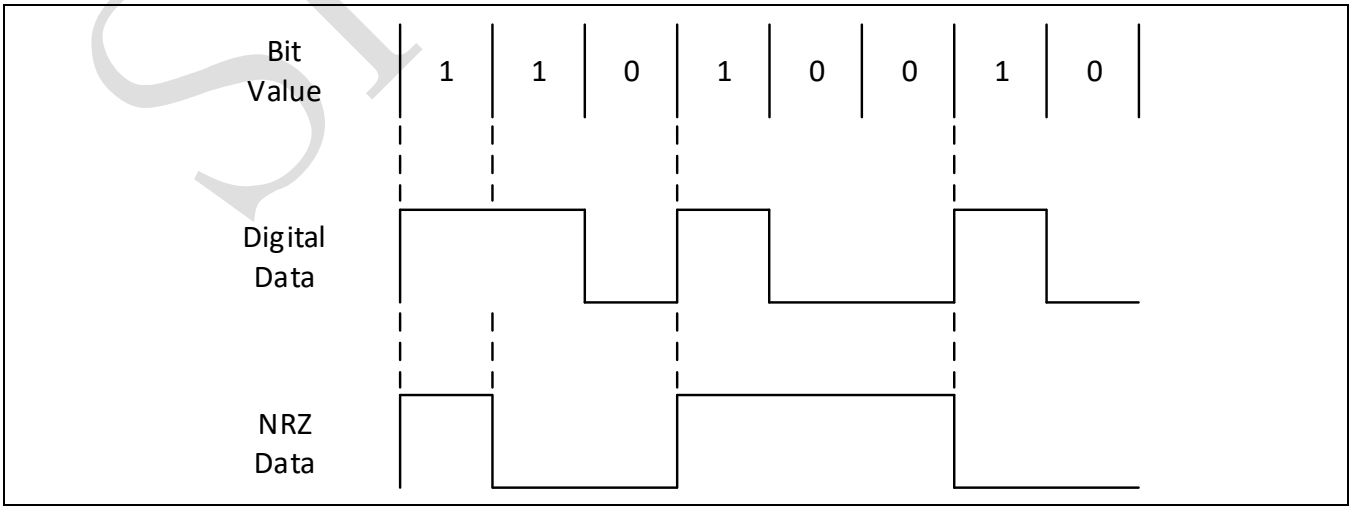
接收数据采样计数的频率是波特率数值的 16 倍。16 倍频的时钟是由波特率生成器产生的。每个比特位在中间采样 3 次。图 17-2 中带深色背景标记的比特位是可选的，可以通过软件编程来控制。

每个数据帧的长度在 9~11 位，取决于可编程的数据位大小以及是否使能奇偶校验。一个数据帧开始传输是通过传输一个开始位（指高电平到低电平的转换）。开始位后紧随着 8 位数据（LSB 先传输）。数据位后面是可选择的校验位。校验位被置“1”：偶校验被使能且数据字节包含奇数个 1，或者奇校验被使能且数据字节包含偶数个 1。数据帧的结束是一个停止位。停止位是指逻辑 1 持续一个比特位的时间。

UART 模块有 2 个 FIFO：一个用于发送，一个用于接收。TXFIFO 是 64 字节深度，8 位宽度；RXFIFO 是 64 字节深度，11 位宽度，其中 3 位用于追踪错误。

UART 可以使用 NRZ 编码来表示单个比特位的值。将 UARTIER.NRZME 置“1”可以使能 NRZ 编码。比特 1 代表线路电平转换，比特 0 代表没有线路电平转换。图 17-3 展示了数据字节 0b0100_1011 的 NRZ 编码。这个字节的 LSB 位首先传输。

图 17-3：NRZ 编码示例 –0b0100_1011



17.4.1 Reset

UART 在复位后是关闭的。要想使能 UART，软件必须编程 GPIO 引脚配置寄存器，然后将 UARTIER.UUE 位段置“1”。当 UART 被使能后，接收器等待一个帧的开始位；如果在 THR 寄存器中的数据是有效的，发送器进行数据发送。要被发送的数据可以在 UART 单元使能之前被写入到 THR 寄存器。在 FIFO 模式下，数据先被从 FIFO 发送到 THR 寄存器，然后再发送到管脚上。

当 UART 单元被关闭时，发送器或者接收器完成当前的字节传输后，会停止发送或者接收更多的数据。在 FIFO 中的数据不会被清除，当 UART 使能时，这些数据的传输会重新开始。

17.4.2 FIFO 操作

UART 模块有一个 TXFIFO 和一个 RXFIFO，每个 FIFO 能够保存 64 字节的数据。编程 I/O 是一种将数据移入或移出 FIFO 的方法。在编程 I/O 模式下，使用查询模式。

17.4.2.1 FIFO 中断模式操作

UARTMCR.OUT2 位段是 UART 中断的全局使能位，只有将该位段置“1”，才能使能 UART 中断。

17.4.2.2 接收中断

若要让一个接收中断可以发生，RXFIFO 和接收中断必须使能。当 FIFO 到达触发阈值时，UARTIIR.IID 位段的值会改变，表明接收数据是有效的。当 FIFO 掉到触发阈值以下时，UARTIIR.IID 位段的值会改变，表明在等待下一个中断。软件读取 UARTIIR.IID 位段的值，可以确定中断的产生原因。

接收线路状态（receive-line-status）中断有最高的优先级；接收数据有效（received-data-available）中断的优先级要低一些。只有当字符数据在被放入 FIFO 前出错，线路状态中断（line-status interrupt）才会产生。

当一个字符从移位寄存器传输到 RXFIFO 时，UARTLSR.DR 位段会被置“1”。当 FIFO 为空时，UARTLSR.DR 位段被清零。

注意： 如果使能了接收中断但是没有使能 RXFIFO，那么只要 UARTRBR 寄存器中接收到数据就会产生一个接收中断。

17.4.2.3 接收字符超时中断

当 RXFIFO 和接收超时中断被使能时，并且以下条件存在时，接收超时中断会发生：

- 最近一次收到的字符是在接收 4 个连续字符所需时间之前接收到的
- 最近一次 FIFO 读取是接收 4 个连续字符所需时间之前执行的

在从 RXFIFO 中读取一个字符之后或者接收到一个新的开始位，超时中断被清除并且超时计时器复位。如果超时中断没有发生，当接收到一个新的字符或者读取 RXFIFO 时，超时计时器复位。

17.4.2.4 发送中断

当 FIFO 使能且发送中断使能时，才会产生发送中断；当 FIFO 关闭时，写一次 THR 寄存器就会产生一个新的发送中断。当 TXFIFO 至少是半空时，会发生发送数据请求中断（transmit data-request interrupt）。当 THR 寄存器被写或者 UARTIIR 寄存器被读时，中断被清除。

17.4.2.5 清除尾部字节（trailing bytes）

CPU 必须消除尾部字节。接收超时中断发出信号通知尾部字节的存在。当处理接收超时中断服务时，CPU 使用下面的过程：

1. 读取 UARTLSR 寄存器，检查错误。
2. 通过 UARTIER.RTOIE 位段关闭接收器超时中断。
3. 从 UART FIFO 中读取数据。
4. 读取 UARTLSR 寄存器，检查错误，然后转到步骤 3。如果 UARTLSR.DR 位段的值为“1”，转到步骤 5。
5. 当 FIFO 中不在有数据，通过 UARTIER.RTOIE 位段重新使能接收超时中断。
6. 完成。

17.4.2.6 FIFO 查询操作模式

当 FIFO 使能后，清除 UARTIER 寄存器的位[4:0]，会让 UART 端口工作于 FIFO 查询模式。接收器和发送器被分别控制，可以单独一个或者两个都处于查询模式。在查询模式下，软件通过 UARTLSR 寄存器检查接收器和发送器的状态。为了接收和传输数据的服，CPU 通过查询以下位段：

- 接收数据服务 – CPU 检查 UARTLSR.DR 位段，当一或更多的字节保留在 RXFIFO 或者 RBR 寄存器中时，UARTLSR.DR 位段被置“1”。
- 发送数据服务 – CPU 检查 UARTLSR.TDRQ 位段，当发送器需要数据时，UARTLSR.TDRQ 位段被置“1”。

CPU 也可以检查 UARTLSR.TXDONE 位段，当 TXFIFO 为空时，UARTLSR.TXDONE 位段被置“1”。

17.4.3 自动波特率检测

UART 模块支持自动波特率检测。当使能自动波特率检测后，UART 对开始位脉冲持续的时钟周期进行计数，然后将这个计数值写入到 UARTACR 寄存器中，用于计算波特率。当 UARTACR 寄

存器被写后，可以产生自动波特率锁定（auto-baud-lock）中断（如果该中断被使能）。UART 可以根据波特率的值，自动编程分频锁存寄存器（Divisor Latch Registers）。如果需要的话，CPU 可以读取 UARTACR 寄存器的值，并使用这个信息计算波特率以及编程 UARTDLL 和 UARTDLH 寄存器。在波特率被编程之后，CPU 通过预定的字符（通常是 AT 或 at）被正确的接收进行验证。

如果 UART 要去编程分频锁存寄存器，软件可以基于公式来计算波特率。在大多数商业电子产品中见到的波特率值被称为“普通的”波特率，如表 17-2 所示。表 17-2 所示的波特率都可以通过 UART 来编程。基于公式来计算波特率的方法适用于波特率较高的情形。但是，如果远程发送器的实际波特率误差大于 1%时，在波特率低于 28.8kbps 的情形下，通信可能会失败。

当波特率被检测到后，自动波特率电路可以通过清楚 UARTACR.CNTVAL 位段来关闭自身。如果要重新使能自动波特率检测，将 UARTABR.ABE 位段再次置“1”即可。

注意： 在发送或者接收数据时，改变波特率是不被允许的。在 IrDA（串行红外）模式下，自动波特率检测是不被支持的。

17.4.4 可编程的波特率产生器

UART 包含了一个可编程波特率产生器。它将一个固定的输入时钟除进行分频，用于产生想要的波特率。波特率采用 UART 外设时钟来计算。

波特率产生器的输出频率是波特率的 16 倍。2 个 8 位的分频锁存寄存器（UARTDLL 和 UARTDLH）以二进制格式存储 16 位的分频值。在初始化期间，需要加载这些分频值，以确保波特率产生器正确地工作。如果任意一个分频锁存寄存器被写入 0x0，16 倍频的时钟会停止。被移入或移出 UART 的数据的波特率由下面的式给出：

$$\text{BaudRate} = \frac{UART\_CLK}{16 \times Divisor}$$

例如，如果分频值是 24，且 UART 时钟为 24 MHz，波特率为 62500bps。

注意： 在发送或者接收数据时，改变波特率是不被允许的。
UART 配置完成且使能之后，才允许配置自动波特率功能。

参考表 17-2，表中是一系列推荐的波特率值，基于分频值（UARTDLH:UARTDLL）。

复位后，分频值为 0x0002。在发送或者接收数据时，不允许改变波特率（写寄存器 UARTDLH 和 UARTDLL）。

表 17-2：推荐的波特率

推荐的波特率（bps）
9600
19200

推荐的波特率（bps）
38400
57600
115200
230400
460800
921600
1842000
3686400

17.4.5 串行红外异步接口

UART 模块包含一个慢速串行红外（SIR）接口，支持双向无线红外通信。SIR 接口提供一个发送编码器和接收解码器，支持符合 IrDA 串行红外标准的物理链路。

SIR 接口不包含实际的 IR LED 驱动或者接收放大器。连接到 SIR 接口的 I/O 引脚只有数字 CMOS 电平信号。SIR 支持双向通信，但是不能实现全双工通信，因为发送 LED 的反射会进入到接收器。SIR 支持的频率高达 115.2kbps。

注意： 只有在 UART 模块完全空闲或关闭时，SIR 接口才可以打开或者关闭。运行中的更改可能会产生问题，不建议这样做。

SIR 调制技术支持 7 位或者 8 位字符数据，以及一个可选的校验位。在数据位前面的是一个开始位（值为 0），以一个停止位结束。编码方案是在在每个零值比特位的中间发送一个 1.6us（或者 3/16 位宽）的脉冲；如果比特位的值为 1，则不发送脉冲。每个零值比特位的脉冲必须发生，即使他们是连续的，之间没有边沿。图 17-4 展示了一个发送和接收操作的示例。

图 17-4：IR 发送和接收示例

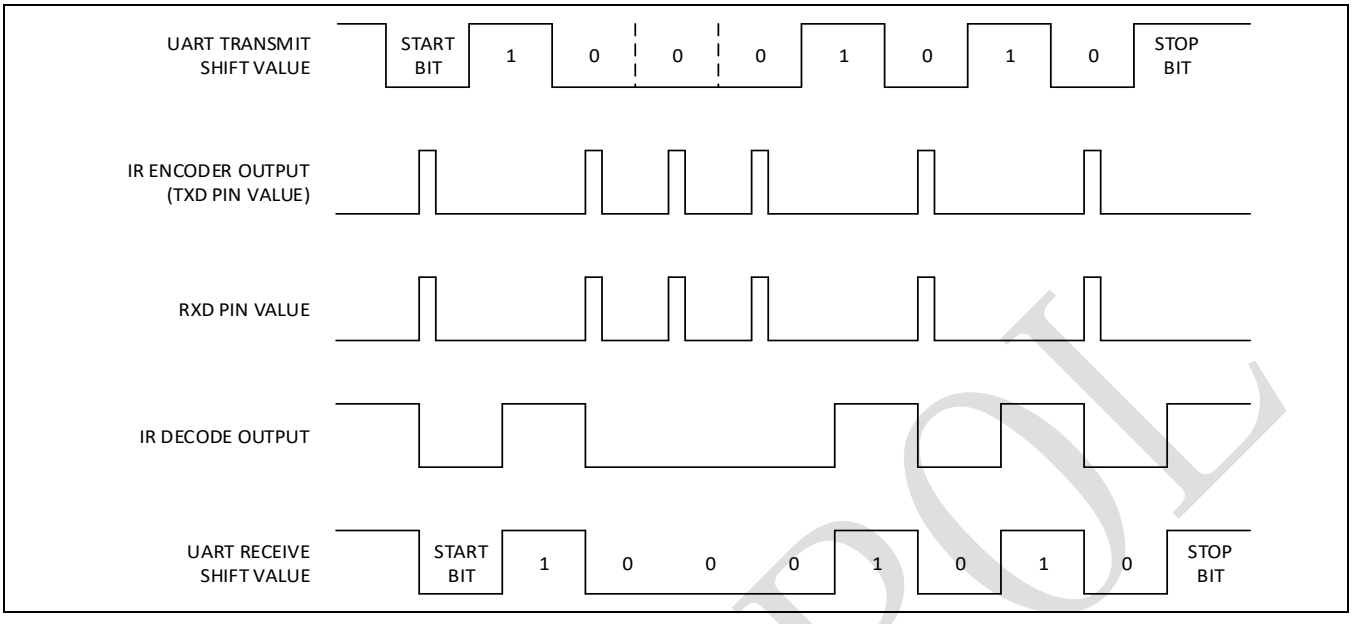
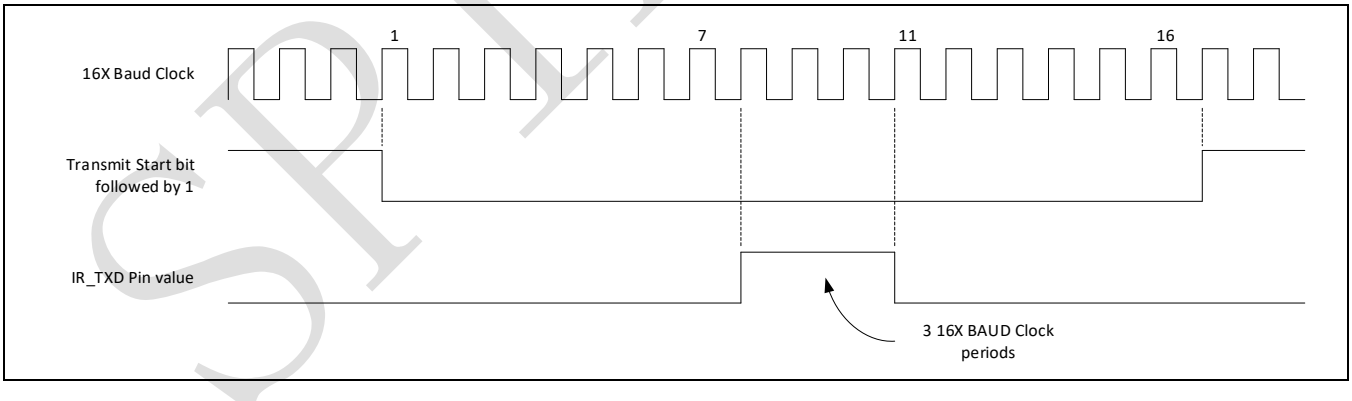


图 17-4 中的最上面一行是从 UART 发出的一个异步发送。第二行是 IR 编码器在 TXD 引脚上产生的脉冲。在开始位和任意一个值为 0 的数据位中间产生一个 1.6us 的脉冲。第三行是 RXD 输入引脚上接收到的值。第四行接收解码器的输出。当检测到一个脉冲后，接收解码器会将接收数据线拉低。最下面一行展示了 UART 接收器如何中断解码器的行为。最后一行与第一行是相同的，但是它被移动了半个比特位的周期。

每个零值比特位的脉冲宽度是一个比特位时间的 3/16，如图 17-5 所示。

图 17-5：零值比特位的脉冲



为了防止发送器 LED 反射反馈到接收器，当 IR 发送编码器发送数据时，需要关闭 IR 接收解码器；当 IR 接收解码器接收数据时，需要关闭 IR 发送编码器。UARTISR.RCVEIR 和 UARTISR.XMITIR 不可以被同时置“1”。

注意： 确切的说，这个限制是为了确保没有反馈；没有滤波器来删除这些反馈数据。

17.5 寄存器

17.5.1 UART 寄存器列表

表 17-3：UART 模块基地址

外设模块	基地址
UART	0x4000 4000

表 17-4：UART 寄存器列表

寄存器	偏移地址	描述	复位值
UARTRBR	0x00	UART 接收缓存寄存器	0x00000000
UARTTHR	0x00	UART 发送保持寄存器	0x00000000
UARTDLL	0x00	UART 分频系数低字节寄存器	0x00000002
UARTDLH	0x04	UART 分频系数高字节寄存器	0x00000000
UARTIER	0x04	UART 中断使能寄存器	0x00000000
UARTIIR	0x08	UART 中断识别寄存器	0x00000001
UARTFCR	0x08	UART FIFO 控制寄存器	0x00000000
UARTLCR	0x0C	UART 线路控制寄存器	0x00000000
UARTMCR	0x10	UART 调制解调器控制寄存器	0x00000000
UARTLSR	0x14	UART 线路状态寄存器	0x00000060
UARTISR	0x20	UART 红外选择寄存器	0x00000000
UARTFOR	0x24	UART 接收 FIFO 占用寄存器	0x00000000
UARTABR	0x28	UART 自动波特率控制寄存器	0x00000000
UARTACR	0x2C	UART 自动波特率计数寄存器	0x00000000

17.5.2 UART 寄存器

表 17-5 到表 17-32 提供了 UART 模块相关寄存器的详细信息。

表 17-5：UART 接收缓存寄存器（UARTRBR）位段定义

UARTRBR (UART Receive Buffer Register) Offset: 0x0 Default: 0x00000000 Access: UART -> UARTRBR.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 17-6：UART 接收缓存寄存器（UARTRBR）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	在非 FIFO 模式下，本寄存器保存从 UART 接收移位寄存器（Receive Shift Register）接收到的字节。如果使用本寄存器小于 8 比特，那么比特位是右对齐并且最高位（MSB）是零。读本寄存器会清空寄存器以及清零 UARTLSR.DR 位段。在 FIFO 模式下，UARTRBR 锁存 FIFO 前面的数据字节。

表 17-7：UART 发送保持寄存器（UARTTHR）位段定义

UARTTHR (UART Transmit Holding Register) Offset: 0x0 Default: 0x00000000 Access: UART -> UARTTHR.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 17-8：UART 发送保持寄存器（UARTTHR）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	WO	0x0	在非 FIFO 模式下，本寄存器保存要被发送的下一个数据字节。当发送移位寄存器（Transmit Shift Register，TSR）为空时，那么本寄存器的内容被加载到 TSR 并将 UARTLSR.TDRQ 位段置“1”。 在 FIFO 模式下，对本寄存器的写操作会将数据放到 FIFO 的顶端。当 TSR 为空时，在 FIFO 前部的数据被加载到 TSR 中。

表 17-9：UART 分频系数低字节寄存器（UARTDLL）位段定义

UARTDLL (UART Divisor Latch Low Byte Register) Offset: 0x0 Default: 0x00000002 Access: UART -> UARTDLL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
VAL							

表 17-10：UART 分频系数低字节寄存器（UARTDLL）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	VAL	RW	0x2	产生波特率的分频系数低字节数据

表 17-11：UART 分频系数高字节寄存器（UARTDLH）位段定义

UARTDLH (UART Divisor Latch High Byte Register) Offset: 0x4 Default: 0x00000000 Access: UART -> UARTDLH.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
VAL							

表 17-12：UART 分频系数高字节寄存器（UARTDLH）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	VAL	RW	0x0	产生波特率的分频系数高字节数据

表 17-13：UART 中断使能寄存器（UARTIER）位段定义

UARTIER (UART Interrupt Enable Register) Offset: 0x4 Default: 0x00000000 Access: UART -> UARTIER.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	UUE	NRZME	RTOIE	RESERVED	RLSE	TIE	RAVIE

表 17-14：UART 中断使能寄存器（UARTIER）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	DMAE	RW	0x0	使能 DMA 请求 0：关闭 DMA 1：使能 DMA
6	UUE	RW	0x0	使能 UART 单元 0：关闭 UART 单元 1：使能 UART 单元
5	NRZME	RW	0x0	使能 NRZM 编码 NRZM 编码/解码功能只能用于 UART 模式，不能用在红外模式下。如果串行红外接收器或者发送器使能，NRZM 编码功能就会关闭。

位段	位段名	属性	复位值	描述
				0：关闭 NRZM 编码 采用通用的电平编码，低电平为 0，高电平为 1。 1：使能 NRZM 编码 低电平时，信号保持不变，高电平时，信号翻转。
4	RTOIE	RW	0x0	使能接收器超时中断（触发源 UARTIIR.TOD） 0：关闭接收器数据超时中断 1：使能接收器数据超时中断
3	RESERVED_3	RW	0x0	保留
2	RLSE	RW	0x0	使能接收器线路状态中断（触发源 UARTIIR.IID） 0：关闭接收器线路状态中断 1：使能接收器线路状态中断
1	TIE	RW	0x0	使能发送数据请求中断（触发源 UARTIIR.IID） 0：关闭 TXFIFO 数据请求中断 1：使能 TXFIFO 数据请求中断
0	RAVIE	RW	0x0	使能接收器数据有效中断（触发源 UARTIIR.IID） 0：关闭接收器数据有效（达到阈值）中断 1：使能接收器数据有效（达到阈值）中断

表 17-15：UART 中断识别寄存器（UARTIIR）位段定义

UARTIIR (UART Interrupt Identification Register)　Offset: 0x8　Default: 0x00000001
Access: UART -> UARTIIR.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
FIFOSTS		RESERVED	ABL	TOD	IID		NIP

表 17-16：UART 中断识别寄存器（UARTIIR）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:6	FIFOSTS	RO	0x0	FIFO 模式使能状态 00：非 FIFO 模式 01：保留 10：保留 11：FIFO 模式
5	RESERVED_5	RO	0x0	保留

位段	位段名	属性	复位值	描述
4	ABL	RO	0x0	自动波特率锁定 0：自动波特率电路未编程分频系数锁存寄存器 1：自动波特率电路已编程分频系数锁存寄存器
3	TOD	RO	0x0	检测到超时中断 0：没有挂起的超时中断 1：超时中断挂起（仅 FIFO 模式有效）
2:1	IID	RO	0x0	中断触发源 00：无效选项 01：TX 请求数据 10：接收到的数据有效 11：接收错误（溢出、校验、帧、间断、FIFO 错误）
0	NIP	RO	0x1	无中断挂起 0：有中断挂起（低有效） 1：无中断挂起

表 17-17：UART FIFO 控制寄存器（UARTFCR）位段定义

UARTFCR (UART FIFO Control Register) Offset: 0x8 Default: 0x00000000 Access: UART -> UARTFCR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RXTH		BUS32	RESERVED	TXTH	CLRTF	CLRRF	TRFIFOE

表 17-18：UART FIFO 控制寄存器（UARTFCR）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:6	RXTH	WO	0x0	接收中断触发阈值 当 RXFIFO 中的字节个数等于阈值（本位段的值）且接收数据有效中断使能（UARTIER.RAVIE=1）时，会产生一个中断并设置 UARTIIR 寄存器的相应位段。 00：FIFO 中有 1 个或更多个字节时产生中断 01：FIFO 中有 8 个或更多个字节时产生中断 10：FIFO 中有 16 个或更多个字节时产生中断 11：FIFO 中有 32 个或更多个字节时产生中断

位段	位段名	属性	复位值	描述
5	BUS32	WO	0x0	BUS 宽度 0：8 位 1：32 位
4	RESERVED_4	WO	0x0	保留
3	TXTH	WO	0x0	发送中断触发阈值 0：当 FIFO 半空时，产生中断请求 1：当 FIFO 为空时，产生中断请求
2	CLRTF	W1C	0x0	清空 TXFIFO 当本位段置“1”，会清空 TXFIFO 中的所有字节。如果 UARTIER.TIE 位的值为 1，UARTLSR.TDRQ 位段会置“1”并产生一个发送数据请求中断。但是，发送移位寄存器（LSR）不会被清空，它会完成当前的发送。 0：写 0 无效，读总是读回 0 1：写 1 清空 TXFIFO 本位段自动清零。
1	CLRRF	W1C	0x0	清空 RXFIFO 当本位段置“1”，会清空 RXFIFO 中的所有字节。UARTLSR.DR 位段会被清零。FIFO 中的所有错误位以及 UARTLSR.FIFOE 位段会被清零。在 UARTLSR 寄存器中的任何错误位，OE，PE，FE 或者 BI，已经被置“1”的话，仍然保持为“1”。接收移位寄存器（RSR）不会被清空。如果 UARTIIR.IID 的值为“接收数据有效”，那么 UARTIIR.IID 会被清零。 0：写 0 无效，读总是读回 0 1：写 1 清空 TXFIFO 本位段自动清零。
0	TRFIFOE	WO	0x0	使能 TXFIFO 和 RXFIFO 本位段（TRFIFO）使能或者关闭 TXFIFO 和 RXFIFO。当 TRFIFOE 位为“1”，两个 FIFO 都会被使能（FIFO 模式）。当 TRFIFOE 位清零，两个 FIFO 都会被关闭（非 FIFO 模式）。向本位段写 0，会清空两个 FIFO 里的所有数据。当从 FIFO 模式变到非 FIFO 模式或者反过来，FIFO 中的数据会被自动清空。当要写本寄存器的其他位段时，本位段必须为“1”，否则的话，其他位段不会被写入。 0：FIFO 被关闭 1：FIFO 被使能

表 17-19：UART 线路控制寄存器（UARTLCR）位段定义

UARTLCR (UART Line Control Register) Offset: 0xC Default: 0x00000000 Access: UART -> UARTLCR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 DLAB	6 SB	5 STKYP	4 EPS	3 PEN	2 STB	1 WLS	0

表 17-20：UART 线路控制寄存器（UARTLCR）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	DLAB	RW	0x0	分频寄存器访问锁定位 在读或者写操作期间，必须置“1”，以便可以访问波特率产生器的分频系数锁存寄存器（Divisor Latch registers）。当要访问 RBR 寄存器，THR 寄存器或者 UARTIER 寄存器时，必须清零。 0：访问 UARTRBR，UARTTHR 和 UARTIER 1：访问分频系数锁存寄存器（UARTDLL 和 UARTDLH）
6	SB	RW	0x0	设置间断（Break） 发送一个 break 信号给接收的 UART。只在 TXD 引脚上起作用，对发送逻辑没有影响。在 FIFO 模式下，要等到发送器空闲时（UARTLSR.TXDONE=1）才可以设置和清除 SB 位。 0：TXD 输出没有影响 1：强制 TXD 输出为 0
5	STKYP	RW	0x0	粘滞校验（Sticky Parity） 本位段在 PEN=0 时，会被忽略。 0：校验位没有影响 1：强制校验位的值与 EPS 位的值相反
4	EPS	RW	0x0	偶校验选择 本位段在 PEN=0 时，会被忽略。 0：发送或者检查奇校验 1：发送或者检查偶校验

位段	位段名	属性	复位值	描述
3	PEN	RW	0x0	使能校验位 0：无校验位 1：使能检验位
2	STB	RW	0x0	停止位 0：1 位停止位 1：1.5 位停止位如果数据位为 5 位 2 位停止位如果数据位为 6/7/8 位
1:0	WLS	RW	0x0	每次传输的数据位长度 00：5 位字符 01：6 位字符 10：7 位字符 11：8 位字符

表 17-21：UART 调制解调器控制寄存器（UARTMCR）位段定义

UARTMCR (UART Modem Control Register) Offset: 0x10 Default: 0x00000000
Access: UART -> UARTMCR.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		RESERVED	RESERVED	GIE	RESERVED	RESERVED	RESERVED

表 17-22：UART 调制解调器控制寄存器（UARTMCR）位段描述

位段	位段名	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	RESERVED_5	RW	0x0	保留
4	RESERVED_4	RW	0x0	保留
3	GIE	RW	0x0	使能全局中断 0：关闭 UART 中断到 CPU 1：使能 UART 中断到 CPU
2	RESERVED_2	RW	0x0	保留
1	RESERVED_1	RW	0x0	保留
0	RESERVED_0	RW	0x0	保留

表 17-23：UART 线路状态寄存器（UARTLSR）位段定义

UARTLSR (UART Line Status Register) Offset: 0x14 Default: 0x00000060 Access: UART -> UARTLSR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7 FIFOERR	6 TXDONE	5 TDRQ	4 BI	3 FE	2 PE	1 OE	0 DR

表 17-24：UART 线路状态寄存器（UARTLSR）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	FIFOERR	RO	0x0	FIFO 错误状态 在非 FIFO 模式下，本位段总是 0。在 FIFO 模式下，当 FIFO 中的任一数据有至少一个校验错误，帧错误，或者 break 标志，FIFOERR 位段置“1”。CPU 读 UARTLSR 寄存器不会复位本位段。当所有的出错的数据都被从 FIFO 中读出后，FIFOERR 位段复位。 0：非 FIFO 模式或者 RXFIFO 中没有错误 1：RXFIFO 中至少有一个数据有错误
6	TXDONE	RO	0x1	发送完成 当 THR（ransmit Holding Register）和 TSR（Transmit Shift Register）都为空时，本位段置“1”。当 UARTTHR 或者 TSR 包含一个数据时，本位段清零。在 FIFO 模式下，TXFIFO 和 TSR 都为空时，本位段置“1”。 0：TSR、THR 或者 FIFO 中有数据 1：发送器中所有数据都被移出了（Shift out）
5	TDRQ	RO	0x1	发送数据请求 表示 UART 准备好接受一个新的数据用于发送。另外，当 UARTIER.TIE=1 时，本位段可以导致 UART 发给 CPU 一个中断。当一个字符数据从 THR 传输到 TSR 时，TDRQ 位段置“1”。当加载数据到 THR 时，本位段清零。在 FIFO 模式下，当 FIFO 中字符数据的一半已经加载到 TSR 中或者 UARTFCR.CLRTF 位段置“1”时，TDRQ 位段置“1”。当 FIFO 已有一半以上的数据，本位段清零。当

位段	位段名	属性	复位值	描述
				加载到 FIFO 中的数据超过 64 个字符时，超过的字符会丢失掉。 0：THR 或者 FIFO 有数据等待被移位出去 1：TXFIFO 中只有一半或者小于一半的数据
4	BI	RO	0x0	接收间断（break）中断 当接收数据的输入低电平保持时间比全“字”的传输时间（起始位+数据位+校验位+停止位的总时间）还长时，BI 位段置“1”。当 CPU 读 UARTLSR 寄存器时，BI 位段被清零。在 FIFO 模式下，只需一个等于 0x00 的字符被加载到 FIFO 中，不管 break 信号的长度。BI 展示了在 FIFO 前部字符的 break 状态，而不是最近接收到的字符。 0：未接收到 break 信号 1：已接收到 break 信号
3	FE	RO	0x0	接收帧错误 帧错误 FE 是指接收到的字符没有有效的停止位。当检测到最后一个数据位或者校验位之后的比特位为 0 时，FE 置“1”。当 CPU 读 UARTLSR 寄存器，FE 被清零。在一个帧错误之后，UART 会重新同步。为了重新同步，UART 会假设帧错误是由于下一个开始位，因此，UART 会采样这个开始位两次，然后再读到数据里。在 FIFO 模式下，FE 表示的是 FIFO 前部字符的帧错误，而不是最近接收到的字符。 0：没有帧错误 1：检测到无效的停止位
2	PE	RO	0x0	接收校验错误 表示接收到的数据字符没有正确的偶校验或奇校验位（由 UARTLCR.EPS 位来选择）。一旦检测到一个检验错误，PE 立即被置“1”。当 CPU 读 UARTLSR 寄存器时，本位段被清零。在 FIFO 模式下，PE 表示的是 FIFO 前部字符的校验错误，而不是最近接收到的字符。 0：没有检验错误 1：发生了检验错误
1	OE	RO	0x0	接收溢出错误 在非 FIFO 模式下，表示在下一个字符被接收之前，RBR（Receive Buffer Register）中的数据没有被 CPU 读取。新的字符会丢失。在 FIFO 模式下，OE 表示 FIFO 的 64 字节空间已满，最近接

位段	位段名	属性	复位值	描述
				收到的字节已被丢弃。一旦检测到一个溢出信号，OE 立即被置“1”。当 CPU 读 UARTLSR 寄存器时，本位段被清零。 0：没有数据溢出 1：接收数据已溢出
0	DR	RO	0x0	接收数据就绪（例如，非空） 当一个完整的字符被接收到并被传输到 RBR 或者 FIFO 时，本位段置“1”。在非 FIFO 模式下，当 RBR 被读取时，DR 位段被清零。在 FIFO 模式下，如果 FIFO 为空（最后一个字符被从 UATRBR 中读取）或者使用 UARTFCR.CLRRF 将 FIFO 复位，DR 位段被清零。 0：没有接收到数据 1：UARTRBR 或者 FIFO 中的数据是有效的

表 17-25：UART 红外选择寄存器（UARTISR）位段定义

UARTISR (UART Infrared Selection Register) Offset: 0x20 Default: 0x00000000
Access: UART -> UARTISR.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			RXPL	TXPL	RESERVED	RCVEIR	XMITIR

表 17-26：UART 红外选择寄存器（UARTISR）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4	RXPL	RW	0x0	接收数据极性 0：SIR 解码器将正脉冲作为 0 1：SIR 解码器将负脉冲作为 0
3	TXPL	RW	0x0	发送数据极性 0：SIR 编码器产生正脉冲当数据位为 0 时 1：SIR 编码器产生负脉冲当数据位为 0 时
2	RESEVED_2	RW	0x0	保留
1	RCVEIR	RW	0x0	使能接收器 SIR 当 RCVEIR 置“1”时，来自 RXD 引脚的信号再被喂给 UART 之前，由 IrDA 解码器进行处理。如果

位段	位段名	属性	复位值	描述
				RCVEIR 被清零，所有进入到 IrDA 解码器的时钟都被阻止，RXD 引脚上的被直接喂给 UART。 0：接收器工作在 UART 模式 1：接收器工作在红外模式
0	XMITIR	RW	0x0	使能发送器 SIR 当 XMITIR 置“1”时，UART 正常的 TXD 输出在被喂给设备管脚之前，由 IrDA 编码器进行处理。如果 XMITIR 被清零，所有进入到 IrDA 编码器的时钟都被阻止，UART 的 TXD 信号直接连接到设备管脚。 0：TXD 信号被直接连接 1：UART TXD 的输出由 IrDA 编码器处理

表 17-27：UART 接收 FIFO 占用寄存器（UARTFOR）位段定义

UARTFOR (UART Receive FIFO Occupancy Register) Offset: 0x24 Default: 0x00000000 Access: UART -> UARTFOR.all							
31	30	29	28	27	26	25	24
RESERVED_31_7							
23	22	21	20	19	18	17	16
RESERVED_31_7							
15	14	13	12	11	10	9	8
RESERVED_31_7							
7	6	5	4	3	2	1	0
RESERVED_31_7	BYTECNT						

表 17-28：UART 接收 FIFO 占用寄存器（UARTFOR）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6:0	BYTECNT	RO	0x0	RXFIFO 中保留的字节数（0~64）

表 17-29：UART 自动波特率控制寄存器（UARTABR）位段定义

UARTABR (UART Auto-Baud Control Register) Offset: 0x28 Default: 0x00000000 Access: UART -> UARTABR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					ABUP	ABLIE	ABE

表 17-30：UART 自动波特率控制寄存器（UARTABR）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	ABUP	RW	0x0	编程 DLR（Divisor Latch Registers）的方法 0：软件编程 DLR 1：UART 编程 DLR
1	ABLIE	RW	0x0	使能自动波特率锁定中断（触发源 UARTIIR.ABL） 0：关闭自动波特率锁定中断 1：使能自动波特率锁定中断
0	ABE	RW	0x0	使能自动波特率检测 0：关闭自动波特率检测 1：使能自动波特率检测

表 17-31：UART 自动波特率计数寄存器（UARTACR）位段定义

UARTACR (UART Auto-Baud Count Register) Offset: 0x2C Default: 0x00000000 Access: UART -> UARTACR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
CNTVAL							
7	6	5	4	3	2	1	0
CNTVAL							

表 17-32：UART 自动波特率计数寄存器（UARTACR）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	CNTVAL	RO	0x0	一个开始位脉冲的 UART 时钟周期数

18 内部集成电路（I2C）模块

18.1 I2C 概述

SPC2168 实现了一个硬件 I2C 模块，该模块为 MCU 和任何通过 I2C 串行总线相连的 I2C 总线兼容设备之间提供了一个接口。连接在 I2C 总线上的外部部件，通过两线 I2C 接口能够串行地向 MCU 发送数据或者从 MCU 接收数据。

I2C 总线是多主机总线，因而，I2C 模块支持多主机模式，允许多个能够控制总线的设备与其连接。每个 I2C 设备可以通过其唯一的地址来识别，而且根据其功能，I2C 设备既可以作为发送器，也可以作为接收器。除了作为发送器或接收器之外，连接在 I2C 总线上的设备在执行数据传输时也可以被视为主机或者从机。需要注意的是，主机是在总线上发起数据传输并生成时钟信号以允许该数据传输的设备。在数据传输过程中，任何被这个主机寻址的设备都被视为从机。

18.2 I2C 主要特性

SPC2168 I2C 模块具有以下特性：

- 兼容 I2C 标准 2.1
- 支持三种速度模式：标准模式（100kb/s），快速模式（400kb/s）和高速模式（2Mb/s）
- 时钟同步
- 支持 7 位或者 10 位地址寻址
- 支持 7 位或者 10 位地址模式下的混合格式传输
- 两个独立的发送和接收 FIFO，每个 FIFO 为 16 x 32 位

18.3 I2C 信号描述

表 18-1：I2C 信号描述

信号名称	类型	描述
I2C_SCL	输入/输出	I2C serial clock line（漏极开路）
I2C_SDA	输入/输出	I2C serial data line（漏极开路）

18.4 I2C 功能描述

I2C 模块通过软件可以被配置为：

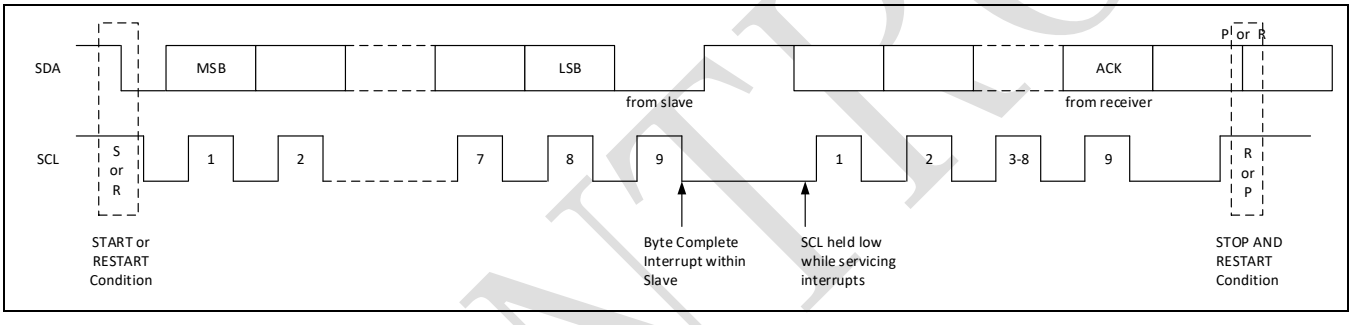
- 仅 I2C 主机模式，与其他 I2C 主机通信
- 仅 I2C 从机模式，与一个以上 I2C 主机通信

主机负责生成时钟并控制数据传输。从机负责向主机发送数据或从主机接收数据。数据的应答由接收数据的设备发送，该设备可以是主机，也可以是从机。如前所述，I2C 协议也允许多个主机存在于 I2C 总线上，并采用仲裁程序来确定总线所有权。

每个从机都有一个由系统设计者确定的唯一的地址。当一个主机想要和一个从机通信时，主机会发送一个开始/重启（START/RESTART）信号，紧接着是从机的地址和一个控制位（读/写），该控制位用以确定主机是向从机传输数据还是从从机接收数据。然后，从机在地址之后发送一个应答（ACK）脉冲。

假如主机（发送器）向从机（接收器）写，接收器接收一个字节的数据；该传输持续到主机通过停止（STOP）信号来终止传输。假如主机（接收器）从从机读，从机（发送器）向主机发送一个字节的数据；随后主机使用 ACK 信号来应答，此次传输将继续进行，直到主机在接收到最后一个字节后使用 NACK 信号来终止传输。最后主机发出停止（STOP）信号或者在发出重启（RESTART）信号后寻址另外一个从机设备。图 18-1 描述了上述行为。

图 18-1：I2C 总线数据传输



I2C 是一个同步串行接口。SDA 线是一个双向信号线，只有当 SCL 线为低电平时才会改变（产生 STOP、START 和 RESTART 信号除外）。输出驱动器是漏极开路或者集电极开路，从而可以在总线上实现“线与”功能。总线可以承载的设备最大数量仅受 400pF 的最大电容规格限制。数据以字节包的形式传输。

注意： 向 FIFO 中放入数据会产生一个 START 信号，清空 FIFO 会产生一个 STOP 信号。更多信息，请参考章节 18.4.1。

18.4.1 START 和 STOP 信号的产生

当 I2C 模块作为主机运行时，将数据写入发送 FIFO 会导致在 I2C 总线上产生一个 START 信号，而发送 FIFO 为空时则会导致在 I2C 总线上产生一个 STOP 信号。

当作为从机运行时，根据 I2C 协议，I2C 模块不会产生 START 和 STOP 信号。然而，如果接收到一个读请求，该 I2C 模块会将 SCL 线拉低直道要读取的数据已经提供给它。此操作会拖住 I2C 总线，直到从机开始提供要读取的数据或者通过将 I2CENABLE.EN 寄存器位清零来关闭 I2C 从机。

18.4.2 混合格式传输

SPC2168 I2C 模块支持在 7 位和 10 位寻址模式下的读/写交替传输。但是，不支持混合寻址模式下的传输，即 7 位寻址模式下传输数据后，紧接着进行 10 位寻址模式的数据传输，反之亦然。

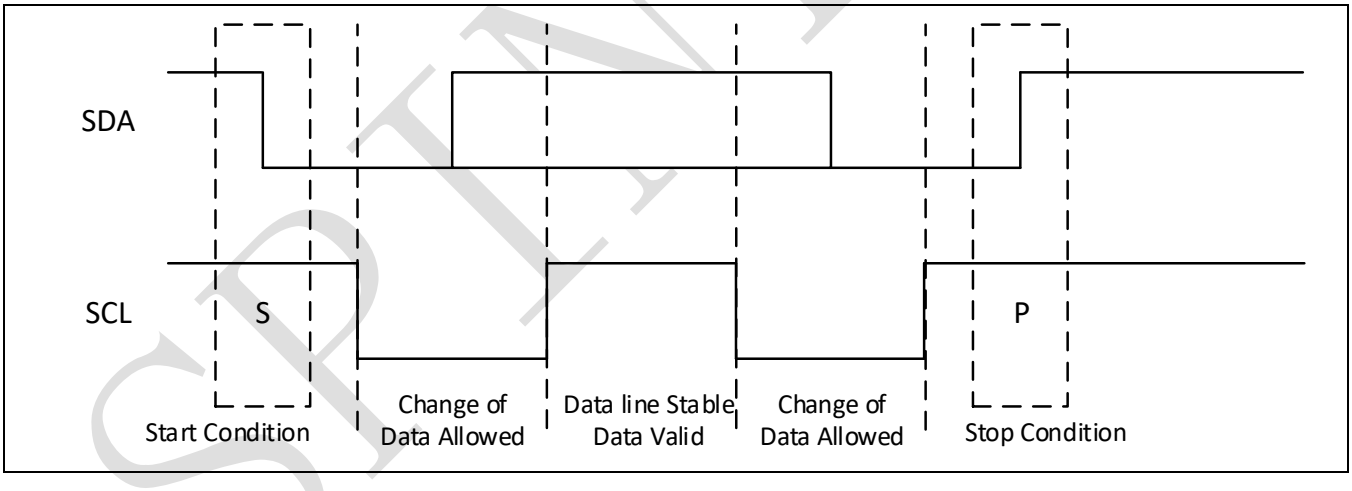
在主机模式下，通过置“1”I2CCTL.RESTARTEN 寄存器位，来初始化混合格式的传输。当完成一次传输时，I2C 模块会检查发送 FIFO 并执行下一次传输。若当前传输方向与上一次传输不同不同，则混合格式传输会被用来启动本次传输。如果当前 I2C 传输完成时，发送 FIFO 为空，则会发出一个 STOP 信号；下一次传输则会在一个 START 信号后面启动。

18.5 I2C 协议

18.5.1 START 和 STOP 信号

当总线空闲时，SCL 和 SDA 信号线都通过总线上的外部上拉电阻拉高。当主机想要在总线上开始一次传输时，主机会发出 START 信号。START 信号被定义为当 SCL 为高电平“1”时，SDA 信号由高电平变到低电平；当主机必须终止传输时，它会发出一个 STOP 信号。STOP 信号被定义为当 SCL 为高电平“1”时，SDA 信号由低电平变到高电平。图 18-2 展示了 START 和 STOP 信号的时序。当数据在总线上被被传输时，在 SCL 为“1”时，SDA 线必须保持稳定。

图 18-2：START 和 STOP 信号



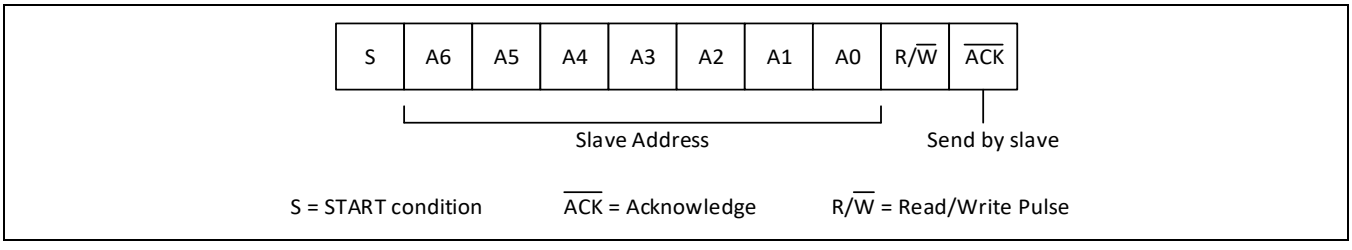
18.5.2 地址格式

I2C 模块支持两种寻址格式：7 位地址和 10 位地址格式。

7 位地址格式

在 7 位寻址格式中，第一个字节的前 7 位（位[7:1]，即图 18-3 中的 A6~A0）设置寻址地址，最后一位（位 0）是读写控制位，如图 18-3 所示。当位 0（R/W）被设置为“0”时，意味着主机向从机写数据；当位 0（R/W）被设置为“1”时，意味着主机从从机读数据。

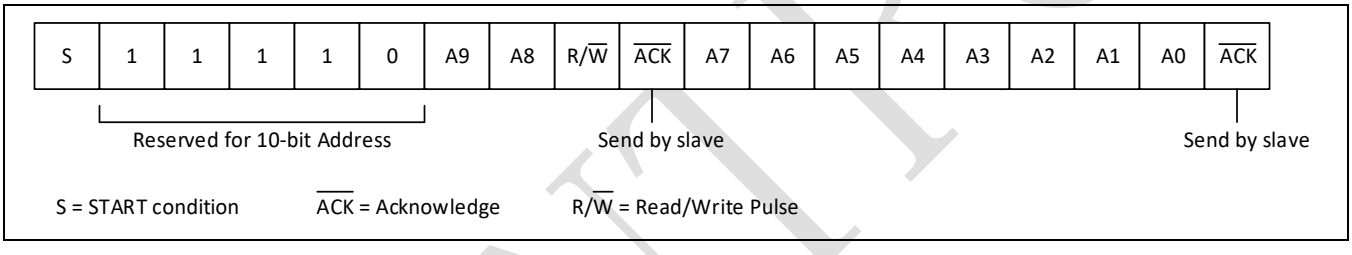
图 18-3：7 位地址格式



10 位地址格式

在 10 位寻址格式中，10 位地址被分成两次传输。第一个被传输的字节的位定义为：前五位（位[7:3]）用于通知从机这是一个 10 位地址传输，接下来的俩位（位[2:1]）用于设置从机地址的位[9:8]，最后一位（位 0）是读/写控制位。第二个被传输的字节用来设置从机地址的位[7:0]。10 位地址格式如图 18-4 所示。

图 18-4：10 位地址格式



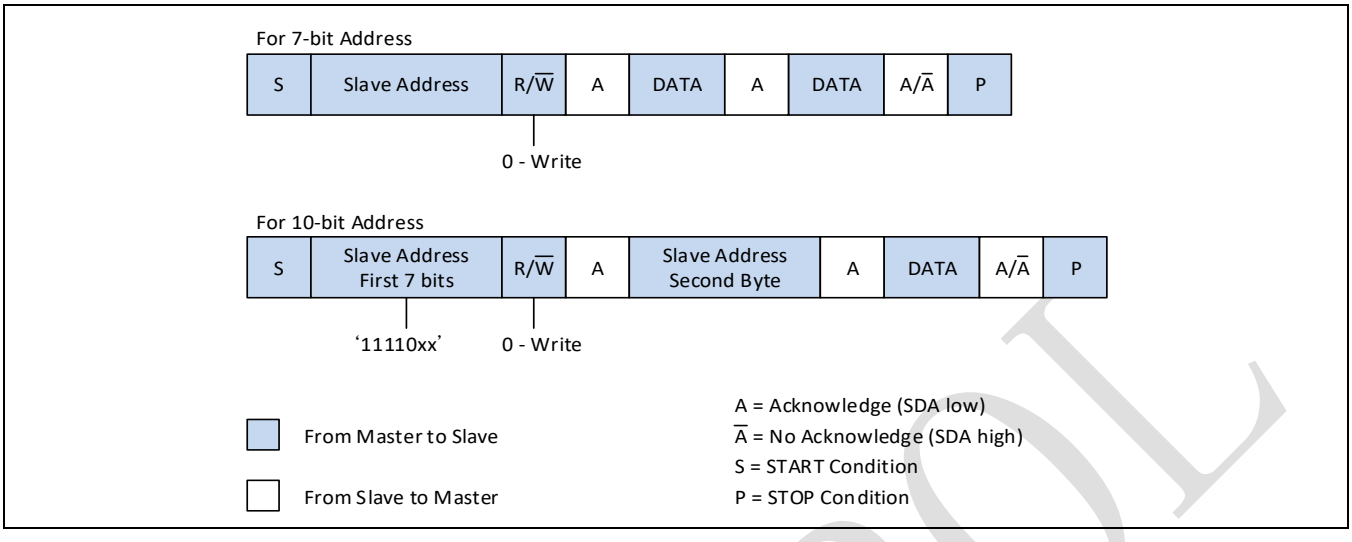
18.5.3 收发协议

主机可以发起向总线传输数据（主机作为发送器）或者从总线接收数据（主机作为接收器）。从机负责响应来至主机的请求，向总线发送数据（从机作为发送器）或者从总线接收数据（从机作为接收器）。

主机-发送器和从机-接收器

所有的数据以字节为单位进行传输，每次数据传输的字节数没有限制。在主机发送地址和读写控制位或者主机向从从机发送一个字节数据后，从机-接收器必须用 ACK 信号响应。当从机-接收器用 ACK 脉冲信号响应时，主机会发出一个 STOP 信号中止传输；于此同时，从机必须将 SDA 线保持在高电平，以便主机可以中止传输。

图 18-5：主机-发送器协议

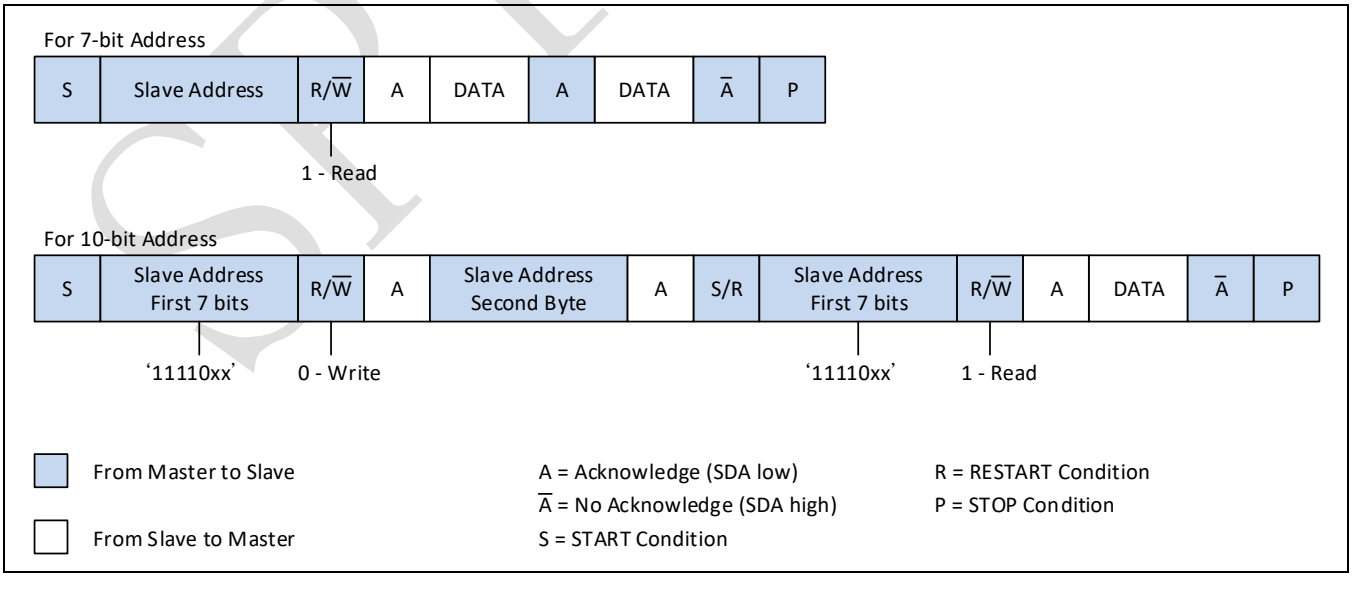


主机-接收器和从机-发送器

假如主机接收数据如图 18-6 所示的那样，那么主机在接收到一个字节的数据（最后一个字节除外）后，会向从机-发送器发送一个 ACK 脉冲信号作为响应。主机-接收器利用这种方式通知从机-发送器这是最后一个字节数据。在检测到 NACK 信号后，从机-发送器会放弃 SDA 线，以便主机可以发出一个 STOP 信号。

当主机拒绝通过发送一个 STOP 信号来放弃总线时，主机可以发出一个 RESTART 信号。这个 RESTART 信号和 START 信号是等同的，唯一的区别是 RESTART 信号是在 ACK 脉冲信号后发生的。此时，如果工作在主机模式，那么 I2C 模块可以继续和同一个从机进行通信而且可以改变传输方向。

图 18-6：主机-接收器协议



18.6 寄存器

18.6.1 I2C 寄存器列表

表 18-2：I2C 模块基地址

外设模块	基地址
I2C	0x4000 6000

表 18-3：I2C 寄存器列表

寄存器	偏移地址	描述	复位值
I2CCTL	0x00	I2C 控制寄存器	0x0000007F
I2CMASTERADDR	0x04	I2C 主机地址寄存器	0x00001055
I2CSLVADDR	0x08	I2C 从机地址寄存器	0x00000055
I2CHSMADDR	0x0C	I2C 高速主机模式地址寄存器	0x00000001
I2CDATACMD	0x10	I2C 数据和命令寄存器	0x00000000
I2CSSHCNT	0x14	I2C 标准速度模式时钟高电平计数寄存器	0x000001F4
I2CSSLCNT	0x18	I2C 标准速度模式时钟低电平计数寄存器	0x0000024C
I2CFSHCNT	0x1C	I2C 快速模式时钟高电平计数寄存器	0x0000004B
I2CFSLCNT	0x20	I2C 快速模式时钟低电平计数寄存器	0x000000A3
I2CHSHCNT	0x24	I2C 高速模式时钟高电平计数寄存器	0x00000008
I2CHSLCNT	0x28	I2C 高速模式时钟低电平计数寄存器	0x00000014
I2CIF	0x2C	I2C 中断标志寄存器	0x00000000
I2CIE	0x30	I2C 中断使能寄存器	0x000008FF
I2CRAWIF	0x34	I2C 中断原始标志寄存器	0x00000000
I2CRXTH	0x38	I2C 接收 FFIO 阈值寄存器	0x00000000
I2CTXTH	0x3C	I2C 发送 FIFO 阈值寄存器	0x00000000
I2CINTCLR	0x40	I2C 中断清除寄存器	0x00000000
I2CRXUDFCLR	0x44	I2C RXUDF 中断清除寄存器	0x00000000
I2CRXOVFCLR	0x48	I2C RXOVF 中断清除寄存器	0x00000000
I2CTXOVFCLR	0x4C	I2C TXOVF 中断清除寄存器	0x00000000
I2CRDREQCLR	0x50	I2C RDREQ 清除寄存器	0x00000000
I2CTXABRTCLR	0x54	I2C TXABRT 中断清除寄存器	0x00000000
I2CRXDONECLR	0x58	I2C RXDONE 中断清除寄存器	0x00000000
I2CACTCLR	0x5C	I2C ACT 中断清除寄存器	0x00000000
I2CSTOPDETCLR	0x60	I2C STOPDET 中断清除寄存器	0x00000000
I2CSTARTDETCLR	0x64	I2C STARTDET 中断清除寄存器	0x00000000
I2CGENCALLCLR	0x68	I2C GENCALL 中断清除寄存器	0x00000000
I2CENABLE	0x6C	I2C 使能寄存器	0x00000000
I2CSTS	0x70	I2C 状态寄存器	0x00000006
I2CTFLVL	0x74	I2C 发送 FIFO 水平寄存器	0x00000000
I2CRFLVL	0x78	I2C 接收 FIFO 水平寄存器	0x00000000

寄存器	偏移地址	描述	复位值
I2CSDAHOLD	0x7C	I2C SDA 保持时间寄存器	0x00000001
I2CTXABRTSRC	0x80	I2C 发送中止源寄存器	0x00000000
I2CSDASETUP	0x94	I2C SDA 建立时间寄存器	0x00000064
I2CACKGC	0x98	I2C 应答广播寻址寄存器	0x00000001
I2CENSTS	0x9C	I2C 使能状态寄存器	0x00000000
I2CFSSPKLEN	0xA0	I2C 快速模式尖峰抑制寄存器	0x00000006
I2CHSSPKLEN	0xA4	I2C 高速模式尖峰抑制寄存器	0x00000002

18.6.2 I2C 寄存器

表 18-4 到表 18-79 提供了 I2C 模块相关寄存器的详细信息。

表 18-4：I2C 控制寄存器（I2CCTL）位段定义

I2CCTL (I2C Control Register) Offset: 0x0 Default: 0x0000007F Access: I2C -> I2CCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	SLVDIS	RESTARTEN	ADDRSTS	SLVADDR10B	SPEED		MASTER

表 18-5：I2C 控制寄存器（I2CCTL）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	SLVDIS	RW	0x1	关闭 I2C 从机模式 注意：本位段的值需要与 MASTER 位段的值保持一致 0：使能从机模式 1：关闭从机模式
5	RESTARTEN	RW	0x1	主机模式下，决定 RESTART 信号是否需要发送。一些旧标准的 I2C 从机不支持处理 RESTART 信号。 当 RESTART 信号被禁止时，主机被禁止执行以下操作： - 在一个传输内改变方向 - 发送开始字节（START BYTE） - 工作在高速模式 - 7 位寻址模式下的混合格式传输

位段	位段名	属性	复位值	描述
				- 10 位寻址模式时的读操作 - 在一次传输中，发送多字节时，用 STOP 信号和随后的 START 信号代替 RESTART 信号，将操作分解为多个 I2C 传输。 如果执行上述操作，会导致 I2CRAWIF.TXABRT 寄存器位被置“1”。 0：关闭 1：使能
4	ADDRSTS	RO	0x1	I2CMASTERADDR.MASTERADDR10B 寄存器位的只读备份。 0：7 位寻址模式 1：10 位寻址模式
3	SLVADDR10B	RW	0x1	从机模式下的寻址模式 0：7 位寻址 I2C 忽略 10 位寻址的传输；对于 7 位寻址，只有 I2CSLVADDR 寄存器值的低 7 位用于比较。 1：10 位寻址 I2C 只对 10 位寻址模式并且地址与 I2CSLVADDR 寄存器的值匹配的传输进行响应
2:1	SPEED	RW	0x3	I2C 主机模式下的速度模式 00：不可用，若写 0，硬件会自动设置为 3 01：标准速度模式（100 kbit/s） 10：快速模式（400 kbit/s） 11：高速模式（3.4 Mbit/s）
0	MASTER	RW	0x1	使能 I2C 主机模式 注意：本位段的值需要与 SLVDIS 位段的值保持一致 0：关闭 I2C 主机模式 1：使能 I2C 主机模式

表 18-6：I2C 主机地址寄存器（I2CMASTERADDR）位段定义

I2CMASTERADDR (I2C Master Address Register) Offset: 0x4 Default: 0x00001055 Access: I2C -> I2CMASTERADDR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED			MASTERADDR10B	SPECIAL	GCORSTART	TARADDR	
7	6	5	4	3	2	1	0
TARADDR							

表 18-7：I2C 主机地址寄存器（I2CMASTERADDR）位段描述

位段	位段名	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12	MASTERADDR10B	RW	0x1	主机模式寻址模式 0：7 位寻址模式 1：10 位寻址模式
11	SPECIAL	RW	0x0	使能特殊命令 Special command enable 0：关闭特殊命令，忽略 GCORSTART 位段设置 1：使能特殊命令（由 GCORSTART 位段指定）
10	GCORSTART	RW	0x0	特殊命令选择 0：广播请求 发起广播请求后，只有写操作可以执行。尝试发起读操作会导致 I2CRAWIF.TXABRT 寄存器位置“1”。I2C 会停留在广播请求模式直到 SPECIAL 位被清除。 1：开始字节（Start Byte）
9:0	TARADDR	RW	0x55	主机传输的目标地址 本位段在发起广播请求时会被忽略。 若要产生开始字节，CPU 只需要向本位段写一次。 本位段的值不可以与 I2CSLVADDR 的值相同。

表 18-8：I2C 从机地址寄存器（I2CSLVADDR）位段定义

I2CSLVADDR (I2C Slave Address Register) Offset: 0x8 Default: 0x00000055 Access: I2C -> I2CSLVADDR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						VAL	
7	6	5	4	3	2	1	0
VAL							

表 18-9：I2C 从机地址寄存器（I2CSLVADDR）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9:0	VAL	RW	0x55	I2C 工作在从机模式时的从机地址 本寄存器只有在 I2C 接口处于关闭状态时（I2CENABLE=0）才可以写入。在其他时刻写本寄存器都是无效的。 注意：从机地址不可以为保留的地址：0x00 ~ 0x07，或者 0x78 ~ 0x7f。当 I2CSLVADDR 或者 I2CMASTERADDR 被设置为保留地址时，不能够保证 I2C 模块会正确工作。

表 18-10：I2C 高速主机模式地址寄存器（I2CHSMADDR）位段定义

I2CHSMADDR (I2C High Speed Master Mode Code Address Register) Offset: 0xC Default: 0x00000001 Access: I2C -> I2CHSMADDR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					VAL		

表 18-11：I2C 高速主机模式地址寄存器（I2CHSMADDR）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2:0	VAL	RW	0x1	高速模式下，8 位主机代码（格式为 0b00001xxx）的低 3 位 本寄存器只有在 I2C 接口处于关闭状态时（I2CENABLE=0）才可以写入。在其他时刻写本寄存器都是无效的。

表 18-12：I2C 数据和命令寄存器（I2CDATACMD）位段定义

I2CDATACMD (I2C Data Buffer and Command Register) Offset: 0x10 Default: 0x00000000 Access: I2C -> I2CDATACMD.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							CMD
7	6	5	4	3	2	1	0
DATA							

表 18-13：I2C 数据和命令寄存器（I2CDATACMD）位段描述

位段	位段名	属性	复位值	描述
31:9	RESERVED_31_9	RO	0x0	保留
8	CMD	RW	0x0	主机模式 0：写请求 1：读请求 从机模式 0：在接收器模式下，总是读回 0；在发送器模式下，需要将本位段设置为 0 1：在接收器模式下，写 1 无效；在发送器模式下，写 1 会导致 TXABRT 中断。
7:0	DATA	RW	0x0	I2C 总线上发送或者接受的数据 向本寄存器写数据，会使数据被发送。 读本寄存器，会获取接收到的数据。

表 18-14：I2C 标准速度模式时钟高电平计数寄存器（I2CSSHCNT）位段定义

I2CSSHCNT (Standard Speed I2C Clock SCL High Count Register) Offset: 0x14 Default: 0x000001F4 Access: I2C -> I2CSSHCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-15：I2C 标准速度模式时钟高电平计数寄存器（I2CSSHCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x1F4	标准速度模式下，SCL 高电平持续时间 例如，SCL 高电平持续 I2CSSHCNT 个时钟周期。 最小的有效值为 6，如果写入的值小于 6，硬件会自动重新设为 6。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-16：I2C 标准速度模式时钟低电平计数寄存器（I2CSSLCNT）位段定义

I2CSSLCNT (Standard Speed I2C Clock SCL Low Count Register) Offset: 0x18 Default: 0x0000024C Access: I2C -> I2CSSLCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-17：I2C 标准速度模式时钟低电平计数寄存器（I2CSSLCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x24C	标准速度模式下，SCL 低电平持续时间 例如，SCL 低电平持续 I2CSSLCNT 个时钟周期。 最小的有效值为 8，如果写入的值小于 8，硬件会自动重新设为 8。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-18：I2C 快速模式时钟高电平计数寄存器（I2CFSHCNT）位段定义

I2CFSHCNT (Fast Speed I2C Clock SCL High Count Register) Offset: 0x1C Default: 0x0000004B Access: I2C -> I2CFSHCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-19：I2C 快速模式时钟高电平计数寄存器（I2CFSHCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x4B	快速模式下，SCL 高电平持续时间 例如，SCL 高电平持续 I2CFSHCNT 个时钟周期。 最小的有效值为 6，如果写入的值小于 6，硬件会自动重新设为 6。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-20：I2C 快速模式时钟低电平计数寄存器（I2CFSLCNT）位段定义

I2CFSLCNT (Fast Speed I2C Clock SCL Low Count Register) Offset: 0x20 Default: 0x000000A3 Access: I2C -> I2CFSLCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-21：I2C 快速模式时钟低电平计数寄存器（I2CFSLCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0xA3	快速模式下，SCL 低电平持续时间 例如，SCL 低电平持续 I2CFSLCNT 个时钟周期。 最小的有效值为 8，如果写入的值小于 8，硬件会自动重新设为 8。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-22：I2C 高速模式时钟高电平计数寄存器（I2CHSHCNT）位段定义

I2CHSHCNT (High Speed I2C Clock SCL High Count Register) Offset: 0x24 Default: 0x00000008 Access: I2C -> I2CHSHCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-23：I2C 高速模式时钟高电平计数寄存器（I2CHSHCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x8	高速模式下，SCL 高电平持续时间 例如，SCL 高电平持续 I2CHSHCNT 个时钟周期。 最小的有效值为 6，如果写入的值小于 6，硬件会自动重新设为 6。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-24：I2C 高速模式时钟低电平计数寄存器（I2CHSLCNT）位段定义

I2CHSLCNT (High Speed I2C Clock SCL Low Count Register) Offset: 0x28 Default: 0x00000014 Access: I2C -> I2CHSLCNT.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-25：I2C 高速模式时钟低电平计数寄存器（I2CHSLCNT）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x14	高速模式下，SCL 低电平持续时间 例如，SCL 低电平持续 I2CHSLCNT 个时钟周期。 最小的有效值为 8，如果写入的值小于 8，硬件会自动重新设为 8。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-26：I2C 中断标志寄存器（I2CIF）位段定义

I2CIF (I2C Interrupt Flag Register) Offset: 0x2C Default: 0x00000000 Access: I2C -> I2CIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				GENCALL	STARTDET	STOPDET	ACT
7	6	5	4	3	2	1	0
RXDONE	TXABRT	RDREQ	TXDREQ	TXOVF	RXDAV	RXOVF	RXUDF

表 18-27：I2C 中断标志寄存器（I2CIF）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11	GENCALL	RO	0x0	广播寻址接收并应答标志 0：未接收到广播寻址 1：接收到广播寻址
10	STARTDET	RO	0x0	START 或者 RESTART 信号标志 0：未检测到 START 或者 RESTART 信号 1：检测到 START 或者 RESTART 信号
9	STOPDET	RO	0x0	STOP 信号标志 0：未检测到 STOP 信号 1：检测到 STOP 信号
8	ACT	RO	0x0	I2C 总线活动状态标志 0：未检测到 I2C 总线活动 1：检测到 I2C 总线活动
7	RXDONE	RO	0x0	I2C 作为从机-发送器时，外部主机接收数据完成标志 0：从机发送被主机应答，主机将继续接收数据 1：从机发送未被主机应答，主机完成数据接收
6	TXABRT	RO	0x0	发送中止标志 0：未发生发送中止 1：发送已中止，发送 FIFO 会保持在冲刷状态，直到 I2CTXABRTCLR 寄存器被读取
5	RDREQ	RO	0x0	I2C 作为从机时，外部主机读请求标志 0：未接收到主机的读请求 1：已接收到主机的读请求。处理器必须响应该中断，并将请求的数据写入到 I2CDATACMD 寄存器

位段	位段名	属性	复位值	描述
4	TXDREQ	RO	0x0	发送请求标志 该标志用来指示发送 FIFO 水平小于或等于 I2CTXTH 寄存器的阈值，以便请求更多的数据。 0：发送 FIFO 水平大于阈值 1：发送 FIFO 水平小于或等于阈值 当发送 FIFO 水平高于阈值时，本位段自动清零
3	TXOVF	RO	0x0	发送 FIFO 上溢标志 该标志用来指示当发送 FIFO 已有 15 个数据时，试图再写 I2CDATACMD 寄存器 0：发送 FIFO 未上溢 1：发送 FIFO 上溢
2	RXDAV	RO	0x0	接收数据有效标志 该标志用来指示接收 FIFO 水平达到或者高于 I2CRXTH 寄存器的阈值，以便数据可以被读取。 0：接收 FIFO 水平低于阈值 1：接收 FIFO 水平等于或者高于阈值 当接收 FIFO 水平低于阈值时，本位段自动清零
1	RXOVF	RO	0x0	接收 FIFO 上溢标志 当接收 FIFO 已有 16 个数据时，此时又从外部 I2C 设备接收到一个字节数据，I2C 模块会做出应答，但是在接收 FIFO 满之后，任何接收到的数据都会丢失。 0：接收 FIFO 未上溢 1：接收 FIFO 上溢
0	RXUDF	RO	0x0	接收 FIFO 下溢标志 该标志用于指示当接收 FIFO 为空时，处理器尝试去读 I2CDATACMD 寄存器。 0：接收 FIFO 未下溢 1：接收 FIFO 下溢

表 18-28：I2C 中断使能寄存器（I2CIE）位段定义

I2CIE (I2C Interrupt Enable Register) Offset: 0x30 Default: 0x000008FF Access: I2C -> I2CIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				GENCALL	STARTDET	STOPDET	ACT
7	6	5	4	3	2	1	0
RXDONE	TXABRT	RDREQ	TXDREQ	TXOVF	RXDAV	RXOVF	RXUDF

表 18-29：I2C 中断使能寄存器（I2CIE）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11	GENCALL	RW	0x1	使能广播寻址接收并应答（GENCALL）中断 0：关闭 GENCALL 中断 1：使能 GENCALL 中断
10	STARTDET	RW	0x0	使能 START 或者 RESTART 信号（STARTDET）中断 0：关闭 STARTDET 中断 1：使能 STARTDET 中断
9	STOPDET	RW	0x0	使能 STOP 信号（STOPDET）中断 0：关闭 STOPDET 中断 1：使能 STOPDET 中断
8	ACT	RW	0x0	使能 I2C 总线活动（ACT）中断 0：关闭 ACT 中断 1：使能 ACT 中断
7	RXDONE	RW	0x1	使能接收完成（RXDONE）中断 0：关闭 RXDONE 中断 1：使能 RXDONE 中断
6	TXABRT	RW	0x1	使能发送中止（TXABRT）中断 0：关闭 TXABRT 中断 1：使能 TXABRT 中断
5	RDREQ	RW	0x1	使能读请求（RDREQ）中断 0：关闭 RDREQ 中断 1：使能 RDREQ 中断
4	TXDREQ	RW	0x1	使能发送请求（TXDREQ）中断 0：关闭 TXDREQ 中断 1：使能 TXDREQ 中断

位段	位段名	属性	复位值	描述
3	TXOVF	RW	0x1	使能发送 FIFO 上溢（TXOVF）中断 0：关闭 TXOVF 中断 1：使能 TXOVF 中断
2	RXDAV	RW	0x1	使能接收数据有效（RXDAV）中断 0：关闭 RXDAV 中断 1：使能 RXDAV 中断
1	RXOVF	RW	0x1	使能接收 FIFO 上溢（RXOVF）中断 0：关闭 RXOVF 中断 1：使能 RXOVF 中断
0	RXUDF	RW	0x1	使能接收 FIFO 下溢（RXUDF）中断 0：关闭 RXUDF 中断 1：使能 RXUDF 中断

表 18-30：I2C 中断原始标志寄存器（I2CRAWIF）位段定义

I2CRAWIF (I2C Raw Interrupt Flag Register) Offset: 0x34 Default: 0x00000000 Access: I2C -> I2CRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED				GENCALL	STARTDET	STOPDET	ACT
7	6	5	4	3	2	1	0
RXDONE	TXABRT	RDREQ	TXDREQ	TXOVF	RXDAV	RXOVF	RXUDF

表 18-31：I2C 中断原始标志寄存器（I2CRAWIF）位段描述

位段	位段名	属性	复位值	描述
31:12	RESERVED_31_12	RO	0x0	保留
11	GENCALL	RO	0x0	广播寻址接收并应答原始标志 0：未接收到广播寻址 1：接收到广播寻址
10	STARTDET	RO	0x0	START 或者 RESTART 信号标志 0：未检测到 START 或者 RESTART 信号 1：检测到 START 或者 RESTART 信号
9	STOPDET	RO	0x0	STOP 信号标志 0：未检测到 STOP 信号 1：检测到 STOP 信号
8	ACT	RO	0x0	I2C 总线活动状态标志 0：未检测到 I2C 总线活动 1：检测到 I2C 总线活动

位段	位段名	属性	复位值	描述
7	RXDONE	RO	0x0	I2C 作为从机-发送器时，外部主机接收数据完成标志 0：从机发送被主机应答，主机将继续接收数据 1：从机发送未被主机应答，主机完成数据接收
6	TXABRT	RO	0x0	发送中止标志 0：未发生发送中止 1：发送已中止，发送 FIFO 会保持在冲刷状态，直到 I2CTXABRTCLR 寄存器被读取
5	RDREQ	RO	0x0	I2C 作为从机时，外部主机读请求标志 0：未接收到主机的读请求 1：已接收到主机的读请求。处理器必须响应该中断，并将请求的数据写入到 I2CDATACMD 寄存器
4	TXDREQ	RO	0x0	发送请求标志 该标志用来指示发送 FIFO 水平小于或等于 I2CTXTH 寄存器的阈值，以便请求更多的数据。 0：发送 FIFO 水平大于阈值 1：发送 FIFO 水平小于或等于阈值 当发送 FIFO 水平高于阈值时，本位段自动清零
3	TXOVF	RO	0x0	发送 FIFO 上溢标志 该标志用来指示当发送 FIFO 已有 15 个数据时，试图再写 I2CDATACMD 寄存器 0：发送 FIFO 未上溢 1：发送 FIFO 上溢
2	RXDAV	RO	0x0	接收数据有效标志 该标志用来指示接收 FIFO 水平达到或者高于 I2CRXTH 寄存器的阈值，以便数据可以被读取。 0：接收 FIFO 水平低于阈值 1：接收 FIFO 水平等于或者高于阈值 当接收 FIFO 水平低于阈值时，本位段自动清零
1	RXOVF	RO	0x0	接收 FIFO 上溢标志 当接收 FIFO 已有 16 个数据时，此时又从外部 I2C 设备接收到一个字节数据，I2C 模块会做出应答，但是在接收 FIFO 满之后，任何接收到的数据都会丢失。 0：接收 FIFO 未上溢 1：接收 FIFO 上溢
0	RXUDF	RO	0x0	接收 FIFO 下溢标志

位段	位段名	属性	复位值	描述
				该标志用于指示当接收 FIFO 为空时，处理器尝试去读 I2CDATACMD 寄存器。 0：接收 FIFO 未下溢 1：接收 FIFO 下溢

表 18-32：I2C 接收 FFIO 阈值寄存器（I2CRXTH）位段定义

I2CRXTH (I2C Receive FIFO Threshold Register) Offset: 0x38 Default: 0x00000000 Access: I2C -> I2CRXTH.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
KEY				VAL			

表 18-33：I2C 接收 FFIO 阈值寄存器（I2CRXTH）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:4	KEY	WO	0x0	写位段[3:0]时，必须向本位段写 0，否则位段[3:0]的值总是 0xF。 本位段总是读回 0。
3:0	VAL	RW	0x0	接收 FIFO 阈值（用于触发 RXDAV 中断） 0000：1 个数据 0001：2 个数据 0010：3 个数据 0011：4 个数据 0100：5 个数据 0101：6 个数据 0110：7 个数据 0111：8 个数据 1000：9 个数据 1001：10 个数据 1010：11 个数据 1011：12 个数据 1100：13 个数据 1101：14 个数据 1110：15 个数据 1111：16 个数据

表 18-34：I2C 发送 FIFO 阈值寄存器（I2CTXTH）位段定义

I2CTXTH (I2C Transmit FIFO Threshold Register) Offset: 0x3C Default: 0x00000000 Access: I2C -> I2CTXTH.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
KEY				VAL			

表 18-35：I2C 发送 FIFO 阈值寄存器（I2CTXTH）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:4	KEY	WO	0x0	写位段[3:0]时，必须向本位段写 0，否则位段[3:0]的值总是 0xF。 本位段总是读回 0。
3:0	VAL	RW	0x0	发送 FIFO 阈值（用于触发 TXDREQ 中断） 0000：无数据 0001：1 个数据 0010：2 个数据 0011：3 个数据 0100：4 个数据 0101：5 个数据 0110：6 个数据 0111：7 个数据 1000：8 个数据 1001：9 个数据 1010：10 个数据 1011：11 个数据 1100：12 个数据 1101：13 个数据 1110：14 个数据 1111：15 个数据

表 18-36：I2C 中断清除寄存器（I2CINTCLR）位段定义

I2CINTCLR (Clear Combined and Individual Interrupt Register) Offset: 0x40 Default: 0x00000000 Access: I2C -> I2CINTCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-37：I2C 中断清除寄存器（I2CINTCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器会清除所有中断标志位以及 I2CTXABRTSRC 寄存器。本位段不会清除硬件可清除的中断，只清除软件可清除中断。 I2CTXABRTSRC.STARTNORESTART 寄存器位是清除 I2CTXABRTSRC 寄出器的一个例外。

表 18-38：I2C RXUDF 中断清除寄存器（I2CRXUDFCLR）位段定义

I2CRXUDFCLR (Clear RXUDF Interrupt Register) Offset: 0x44 Default: 0x00000000 Access: I2C -> I2CRXUDFCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-39：I2C RXUDF 中断清除寄存器（I2CRXUDFCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 RXUDF 标志

表 18-40：I2C RXOVF 中断清除寄存器（I2CRXOVFCLR）位段定义

I2CRXOVFCLR (Clear RXOVF Interrupt Register) Offset: 0x48 Default: 0x00000000 Access: I2C -> I2CRXOVFCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-41：I2C RXOVF 中断清除寄存器（I2CRXOVFCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 RXOVF 标志

表 18-42：I2C TXOVF 中断清除寄存器（I2CTXOVFCLR）位段定义

I2CTXOVFCLR (Clear TXOVF Interrupt Register) Offset: 0x4C Default: 0x00000000 Access: I2C -> I2CTXOVFCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-43：I2C TXOVF 中断清除寄存器（I2CTXOVFCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 TXOVF 标志

表 18-44：I2C RDREQ 中断清除寄存器（I2CRDREQCLR）位段定义

I2CRDREQCLR (Clear RDREQ Interrupt Register) Offset: 0x50 Default: 0x00000000 Access: I2C -> I2CRDREQCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-45：I2C RDREQ 中断清除寄存器（I2CRDREQCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 RDREQ 标志

表 18-46：I2C TXABRT 中断清除寄存器（I2CTXABRTCLR）位段定义

I2CTXABRTCLR (Clear TXABRT Interrupt Register) Offset: 0x54 Default: 0x00000000 Access: I2C -> I2CTXABRTCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-47：I2C TXABRT 中断清除寄存器（I2CTXABRTCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 TXABRT 标志和 I2CTXABRTSRC 寄存器。同时，也会将发送 FIFO 从冲刷/复位状态中释放，允许写发送 FIFO。 I2CTXABRTSRC.STARTNORESTART 寄存器位是清除 I2CTXABRTSRC 寄出器的一个例外

表 18-48：I2C RXDONE 中断清除寄存器（I2CRXDONECLR）位段定义

I2CRXDONECLR (Clear RXDONE Interrupt Register) Offset: 0x58 Default: 0x00000000 Access: I2C -> I2CRXDONECLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-49：I2C RXDONE 中断清除寄存器（I2CRXDONECLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 RXDONE 标志

表 18-50：I2C ACT 中断清除寄存器（I2CACTCLR）位段定义

I2CACTCLR (Clear ACT Interrupt Register) Offset: 0x5C Default: 0x00000000 Access: I2C -> I2CACTCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-51：I2C ACT 中断清除寄存器（I2CACTCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	如果 I2C 不在活动时，读本寄存器清除 ACT 标志。如果 I2C 总线仍然保持活动，ACT 标志会被继续设置。 如果 I2C 模块被关闭且总线上不在有活动，ACT 标志会被硬件自动清除。

表 18-52：I2C STOPDET 中断清除寄存器（I2CSTOPDETCLR）位段定义

I2CSTOPDETCLR (Clear STOPDET Interrupt Register) Offset: 0x60 Default: 0x00000000 Access: I2C -> I2CSTOPDETCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-53：I2C STOPDET 中断清除寄存器（I2CSTOPDETCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 STOPDET 标志

表 18-54：I2C STARTDET 中断清除寄存器（I2CSTARTDETCLR）位段定义

I2CSTARTDETCLR (Clear STARTDET Interrupt Register) Offset: 0x64 Default: 0x00000000 Access: I2C -> I2CSTARTDETCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-55：I2C STARTDET 中断清除寄存器（I2CSTARTDETCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 STARTDET 标志

表 18-56：I2C GENCALL 中断清除寄存器（I2CGENCALLCLR）位段定义

I2CGENCALLCLR (Clear GENCALL Interrupt Register) Offset: 0x68 Default: 0x00000000 Access: I2C -> I2CGENCALLCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							CLR

表 18-57：I2C GENCALL 中断清除寄存器（I2CGENCALLCLR）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	CLR	RO	0x0	读本寄存器清除 GENCALL 标志

表 18-58：I2C 使能寄存器（I2CENABLE）位段定义

I2CENABLE (I2C Enable Register) Offset: 0x6C Default: 0x00000000 Access: I2C -> I2CENABLE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							EN

表 18-59：I2C 使能寄存器（I2CENABLE）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	EN	RW	0x0	控制是否使能 I2C 当使能或者关闭 I2C 时，存在 2 个 CLK_I2C 的延迟。 当 I2C 被关闭时，会发生以下情形： - 发送 FIFO 和接收 FIFO 被冲刷清空。 - I2CIF 寄存器中的状态位仍然有效直到 I2C 进入 IDLE 状态。 - 如果 I2C 正处于发送状态，那么 I2C 会在当前传输完成后停止并删除发送 FIFO 的内容。 - 如果 I2C 正处于接收状态，那么 I2C 会在当前字节后停止当前传输并发送 NACK 信号。 0：关闭 I2C（保持 FIFO 在擦除状态）。当 I2C 处于活动状态时，软件可以关闭 I2C。然而，重要的是，要确保 I2C 被恰当地关闭。 1：使能 I2C

表 18-60：I2C 状态寄存器（I2CSTS）位段定义

I2CSTS (I2C Status Register) Offset: 0x70 Default: 0x00000006 Access: I2C -> I2CSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED	SACT	MACT	RFF	RFNE	TFE	TFNF	ACT

表 18-61：I2C 状态寄存器（I2CSTS）位段描述

位段	位段名	属性	复位值	描述
31:7	RESERVED_31_7	RO	0x0	保留
6	SACT	RO	0x0	从机 FSM（有限状态机）活动状态 0：从机 FSM 处于 IDLE 状态（空闲状态） 1：从机 FSM 未处于 IDLE 状态（活动状态）
5	MACT	RO	0x0	主机 FSM 活动状态 0：主机 FSM 处于 IDLE 状态（空闲状态） 1：主机 FSM 未处于 IDLE 状态（活动状态）
4	RFF	RO	0x0	接收 FIFO 完全满 0：接收 FIFO 有空的位置 1：接收 FIFO 完全满
3	RFNE	RO	0x0	接收 FIFO 非空 0：接收 FIFO 为空 1：接收 FIFO 非空（包含一个或多个数据）
2	TFE	RO	0x1	发送 FIFO 完全空 0：发送 FIFO 非空（包含一个或多个有效数据） 1：发送 FIFO 完全空
1	TFNF	RO	0x1	发送 FIFO 非满 0：发送 FIFO 满 1：发送 FIFO 非满（包含空的位置）
0	ACT	RO	0x0	I2C 活动状态（位段 SACT 和 MACT 的逻辑或） 0：主机 FSM 和从机 FSM 都处于空闲状态 1：主机 FSM 或者从机 FSM 处于活动状态

表 18-62：I2C 发送 FIFO 水平寄存器（I2CTFLVL）位段定义

I2CTFLVL (I2C Transmit FIFO Level Register) Offset: 0x74 Default: 0x00000000 Access: I2C -> I2CTFLVL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			VAL				

表 18-63：I2C 发送 FIFO 水平寄存器（I2CTFLVL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:0	VAL	RO	0x0	发送 FIFO 中有效数据个数

表 18-64：I2C 接收 FIFO 水平寄存器（I2CRFLVL）位段定义

I2CRFLVL (I2C Receive FIFO Level Register) Offset: 0x78 Default: 0x00000000 Access: I2C -> I2CRFLVL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			VAL				

表 18-65：I2C 接收 FIFO 水平寄存器（I2CRFLVL）位段描述

位段	位段名	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4:0	VAL	RO	0x0	接收 FIFO 中有效数据个数

表 18-66：I2C SDA 保持时间寄存器（I2CSDAHOLD）位段定义

I2CSDAHOLD (I2C SDA Hold-Time Register) Offset: 0x7C Default: 0x00000001 Access: I2C -> I2CSDAHOLD.all							
31	30	29	28	27	26	25	24
RESERVED_31_16							
23	22	21	20	19	18	17	16
RESERVED_31_16							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 18-67：I2C SDA 保持时间寄存器（I2CSDAHOLD）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15:0	VAL	RW	0x1	配置 SDA 信号保持时间（以 CLK_I2C 时钟周期为单位） 例如，如果需要的保持时间为 1000ns，CLK_I2C 频率为 10MHz，那么本寄存器的值建议配置为 11。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的

表 18-68：I2C 发送中止源寄存器（I2CTXABRTSRC）位段定义

I2CTXABRTSRC (I2C Transmit Abort Source Register) Offset: 0x80 Default: 0x00000000 Access: I2C -> I2CTXABRTSRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 SLVRDINTX	14 SARBLOST	13 SLVFLUSHTF	12 MARBLOST	11 MASTERDIS	10 RD10BNORESTART	9 STARTNORESTART	8 HSNORESTART
7 STARTACKDET	6 HSACKDET	5 GCREAD	4 GCNACK	3 TXDATANACK	2 ADDR10B2NACK	1 ADDR10B1NACK	0 ADDR7BNACK

表 18-69：I2C 发送中止源寄存器（I2CTXABRTSRC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	SLVRDINTX	RO	0x0	I2C 角色：从机-发送器 当从机响应读请求向远程主机发送数据时，用户向 I2CDATACMD 寄存器的 CMD 位段（位 8）写 1，会造成本位段置“1”。
14	SARBLOST	RO	0x0	I2C 角色：从机-发送器 当从机在向远程主机发送数据时丢失总线控制权，会造成本位段置“1”。与此同时，I2CTXABRTSRC.MARBLOST 寄存器位也会置“1”。 注意：即使从机不在控制总线，总线也有可能出故障。这是一种防故障检查。例如，在数据传输期间，SCL 信号由低变到高，如果数据总线上的数据不是想要传输的，那么 I2C 将不再控制总线。
13	SLVFLUSHTF	RO	0x0	I2C 角色：从机-发送器 当从机接收到一个读命令且发送 FIFO 中存在一些数据时，本位段会置“1”，以便从机产生一个 TXABRT 中断去冲刷发送 FIFO 中的旧数据。
12	MARBLOST	RO	0x0	I2C 角色：主机/从机-发送器 本位段在下列情况下会被置“1”： - 主机失去仲裁 - 当 I2CTXABRTSRC.SARBLOST 位段被置“1”，从机-发送器失去仲裁
11	MASTERDIS	RO	0x0	I2C 角色：主机-发送器/接收器 当用户试图去启动一次主机操作而主机模式被关闭时，本位段会被置“1”。

位段	位段名	属性	复位值	描述
10	RD10BNORESTART	RO	0x0	I2C 角色：主机-接收器 当重启信号功能被关闭（I2CCTL.RESTARTEN=0）而主机发送一个 10 位寻址的读命令时，本位段会被置“1”。
9	STARTNORESTART	RO	0x0	I2C 角色：主机-发送器/接收器 当重启信号功能被关闭（I2CCTL.RESTARTEN=0）而用户试图去发送一个 START Byte 时，本位段会被置“1”。 按照下面的配置可以避免这个错误： (1) I2CCTL.RESTARTEN=1 (2) I2CMASTERADDR.SPECIAL=0 或者 I2CMASTERADDR.GCORSTART=0 否则的话，在通过读 I2CTXABRTCLR 清除本位段后，下一个时钟周期本位段还会被置“1”。
8	HSNORESTART	RO	0x0	I2C 角色：主机-发送器/接收器 当重启信号功能被关闭（I2CCTL.RESTARTEN=0）而用户试图在高速模式下传输数据，本位段会被置“1”。
7	STARTACKDET	RO	0x0	I2C 角色：主机-发送器/接收器 当主机发送一个 START Byte 并且 START Byte 被应答后（错误的行为），本位段被置“1”。
6	HSACKDET	RO	0x0	I2C 角色：主机-发送器/接收器 当主机工作在高速模式下且高速主机地址被应答（错误的行为）时，本位段被置“1”。
5	GCREAD	RO	0x0	I2C 角色：主机-发送器 当主机发送一个广播寻址（General Call）但是用户在广播请求后紧接着从总线去读数据（I2CDATACMD.CMD=1）时，本位段被置“1”。
4	GCNACK	RO	0x0	I2C 角色：主机-发送器 当主机发送一个广播寻址（General Call）但是总线上没有从机应答时，本位段被置“1”。
3	TXDATANACK	RO	0x0	I2C 角色：主机-发送器 当主机已经接受到地址应答，但是在地址之后发送的数据字节并没有收到远程从机的应答，本位段被置“1”。

位段	位段名	属性	复位值	描述
2	ADDR10B2NACK	RO	0x0	I2C 角色：主机-发送器/接收器 当主机工作在 10 位寻址模式下，而且 10 位地址的第二个字节没有被从机应答时，本位段被置“1”。
1	ADDR10B1NACK	RO	0x0	I2C 角色：主机-发送器/接收器 当主机工作在 10 位寻址模式下，而且 10 位地址的第一个字节没有被从机应答时，本位段被置“1”。
0	ADDR7BNACK	RO	0x0	I2C 角色：主机-发送器/接收器 当主机工作在 7 位寻址模式下，而且发送地址后没有被从机应答时，本位段被置“1”。

表 18-70：I2C SDA 建立时间寄存器（I2CSDASETUP）位段定义

I2CSDASETUP (I2C SDA Setup Register) Offset: 0x94 Default: 0x00000064 Access: I2C -> I2CSDASETUP.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
VAL							

表 18-71：I2C SDA 建立时间寄存器（I2CSDASETUP）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	VAL	RW	0x64	配置 SDA 信号建立时间（以 CLK_I2C 时钟周期为单位） 例如，如果需要的建立时间为 1000ns，而 CLK_I2C 的频率为 10MHz，本寄存器的值建议设置为 11。

表 18-72：I2C 应答广播寻址（I2CACKGC）位段定义

I2CACKGC (I2C ACK General Call Register) Offset: 0x98 Default: 0x00000001 Access: I2C -> I2CACKGC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							ACKGC

表 18-73：I2C 应答广播寻址（I2CACKGC）位段描述

位段	位段名	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	ACKGC	RW	0x1	对广播寻址（General Call）的响应 0：用 NACK 响应 General Call 1：用 ACK 响应 General Call

表 18-74：I2C 使能状态寄存器（I2CENSTS）位段定义

I2CENSTS (I2C Enable Status Register) Offset: 0x9C Default: 0x00000000 Access: I2C -> I2CENSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED					SLVRDLOST	SLVDISONBUSY	EN

表 18-75：I2C 使能状态寄存器（I2CENSTS）位段描述

位段	位段名	属性	复位值	描述
31:3	RESERVED_31_3	RO	0x0	保留
2	SLVRDLOST	RO	0x0	由于 I2C 关闭导致的从机已接收的数据丢失 0：在 I2CENABLE 寄存器从 1 变为 0 之前，从机-接收器工作已完成 1：在 I2CENABLE 寄存器从 1 变为 0 之前，从机-接收器工作（已从 I2C 传输中接收了至少一个字节数据）被中止

位段	位段名	属性	复位值	描述
1	SLVDISONBUSY	RO	0x0	当从机处于忙状态时被关闭 0：在从机工作期间，I2CENABLE 寄存器的值保持稳定 1：I2CENABLE 寄存器的值从 1 变为 0，当 I2C 从机正在从远程主机接收： （a） 从机-发送器的地址 （b） 从机-接收器的地址和数据
0	EN	RO	0x0	I2C 使能状态 0：I2C 完全处于关闭状态 可以安全地读 SLVRDLOST 和 SLVDISONBUSY 位段 1：I2C 处于使能状态

表 18-76：I2C 快速模式尖峰抑制寄存器（I2CFSSPKLEN）位段定义

I2CFSSPKLEN (I2C Fast Speed Spike Suppresion Limit Register) Offset: 0xA0 Default: 0x00000006 Access: I2C -> I2CFSSPKLEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
VAL							

表 18-77：I2C 快速模式尖峰抑制寄存器（I2CFSSPKLEN）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	VAL	RW	0x6	快速模式下，SCL 或者 SDA 信号线上能被尖峰抑制逻辑过滤掉的最长尖峰持续时间 最小值为 1，如果写入 0，会被硬件自动设为 1。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

表 18-78：I2C 高速模式尖峰抑制寄存器（I2CHSSPKLEN）位段定义

I2CHSSPKLEN (I2C High Speed Spike Suppresion Limit Register) Offset: 0xA4 Default: 0x00000002 Access: I2C -> I2CHSSPKLEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
VAL							

表 18-79：I2C 高速模式尖峰抑制寄存器（I2CHSSPKLEN）位段描述

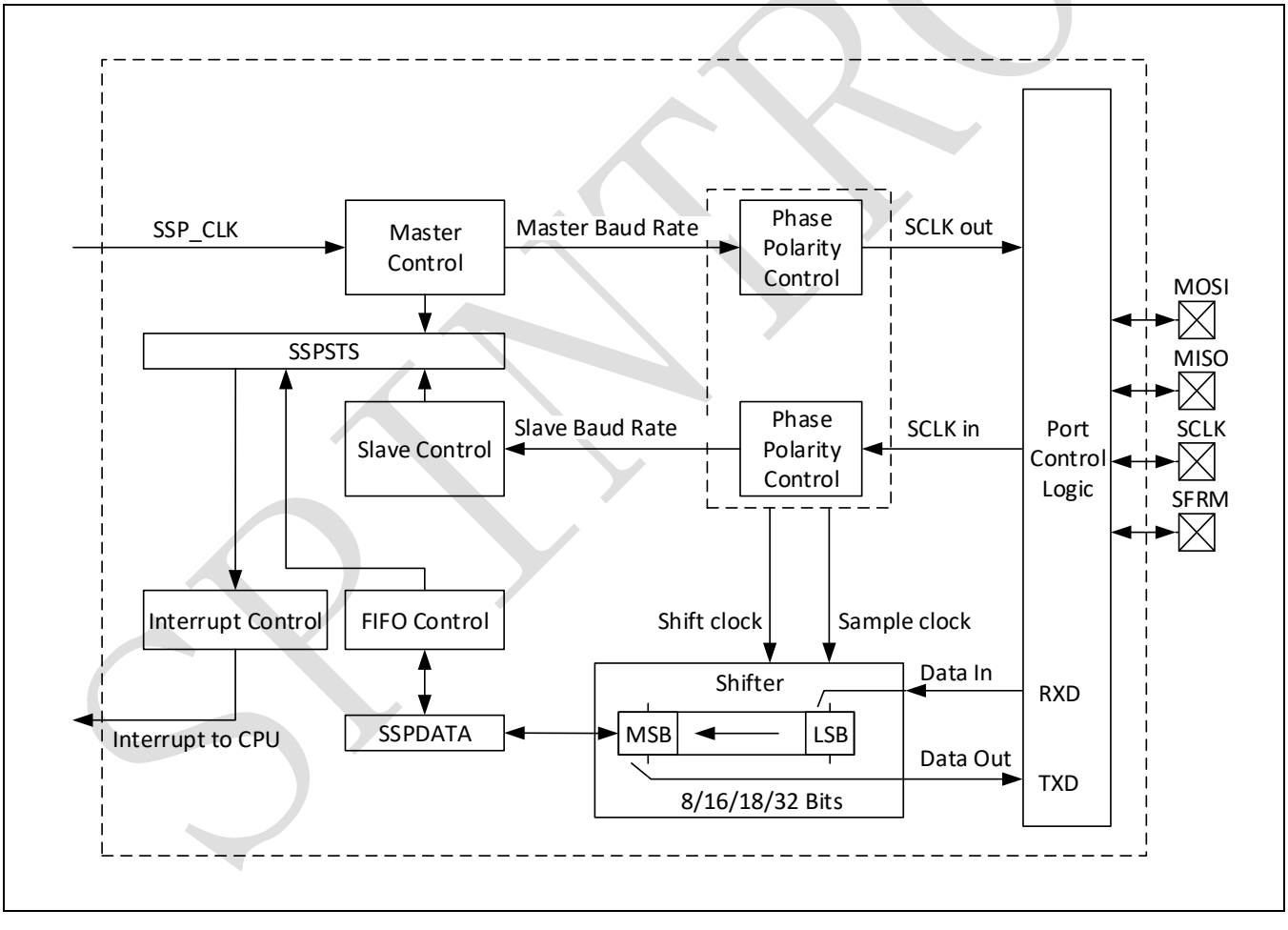
位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7:0	VAL	RW	0x2	高速模式下，SCL 或者 SDA 信号线上能被尖峰抑制逻辑过滤掉的最长尖峰持续时间 最小值为 1，如果写入 0，会被硬件自动设为 1。 本寄存器只有在 I2CENABLE=0 时才可以写入。在其他时刻写本寄存器都是无效的。

19 同步串行端口（SSP）

19.1 SSP 概述

SPC2168 的 SSP 接口是一个同步全双工串行数据传输端口。SSP 接口可以用来连接多种外部接口，如 flash，ADC，CODEC 等。SSP 接口可以配置工作在主机模式（相连的外设设备作为从机）或者从机模式（相连的外设设备作为主机）。通过可配置的时钟，SSP 端口支持不同的通信速率。串行数据帧的长度可以被配置为 8，16，18 或者 32 位。发送数据和接收数据分别有一个独立的 FIFO。两个 FIFO 都是 16 x 32 位的。两个 FIFO 可以被 CPU 装载或者清空。SSP 模块可以通过查询模式或者中断请求模式来控制。SSP 模块主要由控制逻辑、状态和数据寄存器、移位逻辑、FIFO 控制逻辑、主从控制逻辑以及端口控制逻辑组成，如图 19-1 所示。

图 19-1：SSP 模块框图



19.2 SSP 主要特性

SPC2168 SSP 模块具有以下特性：

- 支持 Motorola SPI
- 数据传输速率高达 50 Mbps
- 支持主从模式
- 支持全双工
- 支持只收模式
- 可编程的数据帧长度：1~32 位
- 可编程的片选信号（SFRM）极性选择
- 数据位发送顺序为 MSB 位优先
- 可编程的时钟极性和相位
- 两个独立的发送 FIFO（TXFIFO）和接收 FIFO（RXFIFO），每个 FIFO 的大小为 16 x 32 位，都支持压缩模式（Packed mode）

19.3 SSP 信号描述

表 19-1 描述了 SPC2168 SSP 接口的总线信号。

表 19-1：SSP 信号描述

信号名	类型	描述
MOSI	输入/输出	SPI模块数据传输信号 主机模式：发送数据 从机模式：接收数据
MISO	输入/输出	SPI模块数据传输信号 主机模式：接收数据 从机模式：发送数据
SCLK	输入/输出	SPI 模块时钟信号 主机模式：发送数据时，输出时钟信号 从机模式：接收时钟信号
SFRM	输入/输出	SPI 模块片选信号 主机模式：发送数据时，输出片选信号 从机模式：接收片选信号

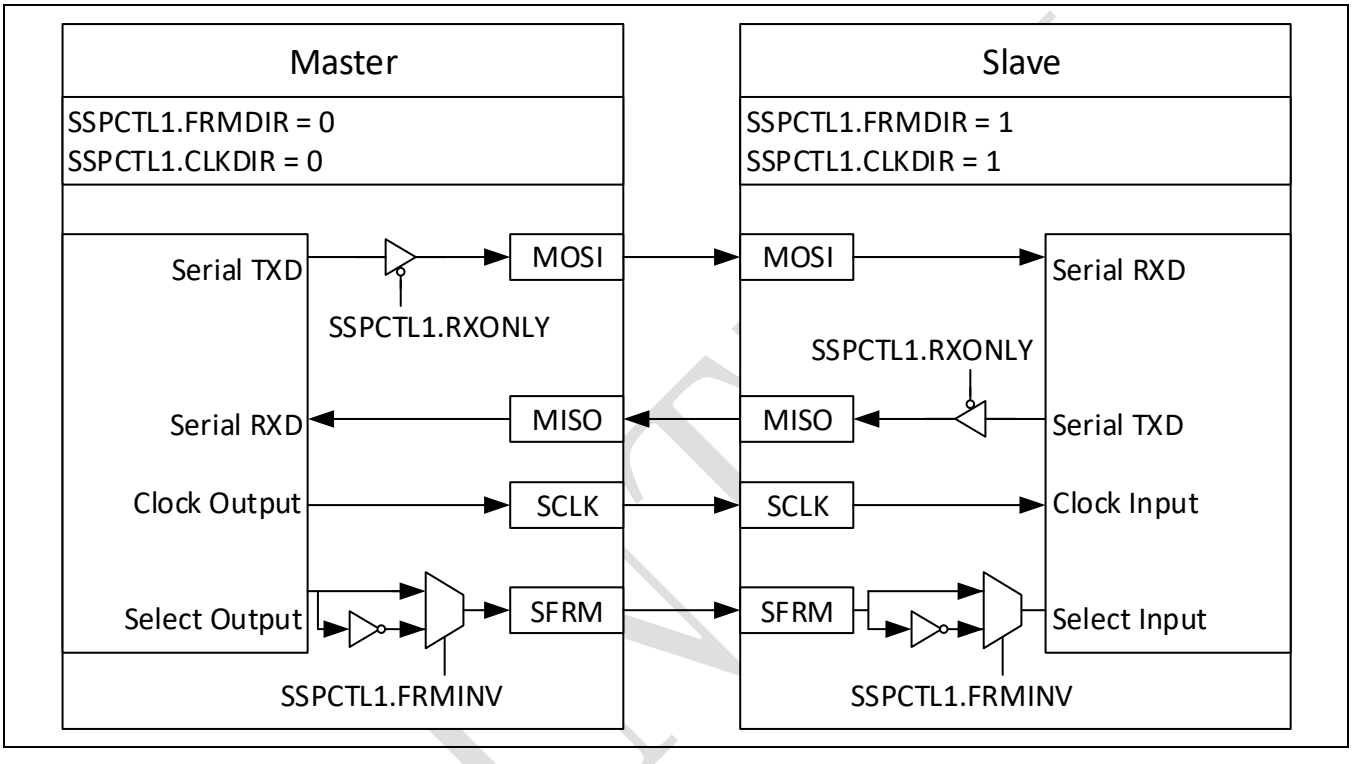
19.4 SSP 功能描述

本章节介绍 SSP 的各种功能操作，包括连接、波特率产生、FIFO 操作、数据帧格式和中断。

19.4.1 功能操作介绍

SSP 可以被配置为主机或者从机。主机发送信号 SCLK 和 SFRM 给从机，从机基于信号 SCLK 和 SFRM 收发数据。主机数据口（MOSI/MISO）与从机名称相同的数据口相连。主机通过 MOSI 发送数据，从 MISO 接收数据；从机从 MOSI 接收数据，通过 MISO 发送数据。SSP 主机和从机之间通信的典型连接如图 19-2 所示。

图 19-2：SSP 主机-从机连接示意图



数据传输存在三种可能的方式：

- 主机发送数据；从机发送无效数据（Dummy data）
- 主机发送数据；从机也发送数据
- 主机发送无效数据；从机发送数据

当发送无效数据时，如果不想从 TXD 信号接口向外部 MOSI 口发送数据，可以把 SSPCTL1.RXONLY 寄存器位置“1”。

19.4.2 波特率产生

SSP 的波特率时钟由 SSP 时钟（SSPCLK）产生。SSPCLK 由 SYSCLK1 分频得到，如图 3-11 所示。SSP 时钟分频器是 16 位的。

19.4.3 FIFO 操作

SPC2168 SSP 拥有两个独立的 FIFO，分别用于发送和接收。CPU 可以装载和清空这些 FIFO。CPU 每次访问可以传输一个 FIFO 条目（FIFO Entry），而且这些访问必须是 32 位宽度的。CPU 写数据到 TXFIFO 是 32 位宽度的，但是串行化逻辑会忽略超过 FIFO 数据大小（SSPCTL0.ESIZESEL 和 SSPCTL0.SIZESEL）的高位数据。CPU 从 RXFIFO 读数据也是 32 位宽度的，FIFO 数据大小不够 32 位的，会在高位补 0。

从 CPU 的角度看来，TXFIFO 和 RXFIFO 共用同一个 32 位宽的位置。对于数据发送，SSP 接口从 TXFIFO 中获取数据，将数据串行化并通过 TXD 信号接口将数据发送到外部设备；对于数据接收，通过 RXD 信号接口将串行数据转换为并行数据并写入到 RXFIFO 中。取决于来自 CPU 的访问是读还是写操作，FIFO 数据寄存器会自动定位到 RXFIFO 或者 TXFIFO。从存储器映射的角度来看，TXFIFO 和 RXFIFO 共享相同的地址。

假若可编程的 FIFO 触发阈值被超过了，会产生一个中断服务请求，用来通知 CPU 清空 RXFIFO 或重新装满 TXFIFO。

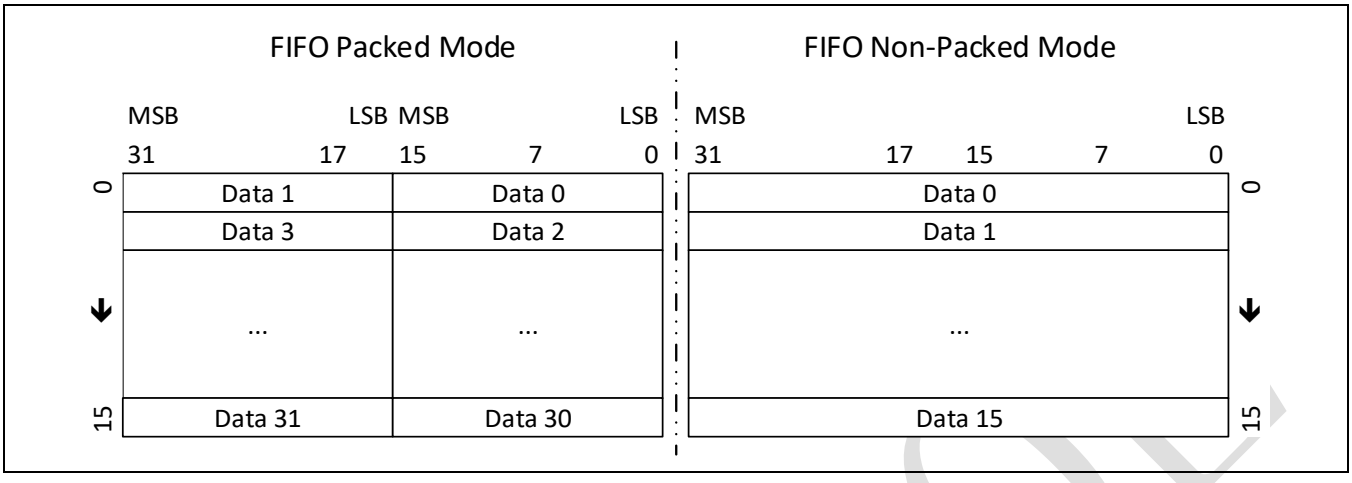
注意： 如果 SSP 被配置为从机且工作在 Receive Only 模式（SSPCTL1.RXONLY=1），那么在接收数据的时候没有必要去写 TXFIFO（半双工）。但是，写 TXFIFO 仍然会发送数据。

19.4.3.1 FIFO 存储的并行数据格式

样本数据（sample data）样本的大小可以是 8，16，18 或者 32 位。FIFO 中的数据样本是以 32 位形式存储或者以 16 位形式存储（压缩模式）。对于 32 位形式存储或者 16 位形式存储，存储的数据样本都是右对齐的，LSB 位在位 0。对于读操作，未使用的位会被填充 0；对于写操作，未使用的位不用去关心。SSP 模块中的逻辑会自动格式化在 TXFIFO 中的数据，以便样本数据可以在 TXD 信号上按选定的帧格式被恰当地发送。

当 FIFO 工作在压缩模式（packed mode）时，如图 19-3 所示，每一个 FIFO 是 32 x 16 位的，总共有 32 个数据样本。每一个数据样本在长度上可以是 8 位或者 16 位。当数据被串行化并被发送时，位 15~0 先被发送，紧接着发送位 31~16。对压缩模式来说，最佳做法是把 FIFO 的一个条目（entry）看做两个样本（sample）。因此，当 CPU 一次写或者读 32 位数据时，每次写或者读需要传输 2 个样本。整个 FIFO（32 位）都必须以这种模式来读或者写。此外，FIFO 的触发阈值也需要以 32 位读写来计算，不能基于 16 位。

图 19-3：FIFO 压缩和非压缩模式



读取 SSPSTS.RFODDSTS 和 SSPSTS.TFODDSTS 可以判断 FIFO 中的样本数据是偶数个还是奇数个。这些数据位有助于在 FIFO 压缩模式下正确地读取数据。

19.4.3.2 RXFIFO 中尾部字节（trailing bytes）

当 RXFIFO 中的数据样本个数小于 RXFIFO 的触发阈值且没有接收到其他数据，那么这些剩余字节数据就被称作 RXFIFO 尾部字节。RXFIFO 中的尾部字节是通过 CPU 查询模式来处理的。根据 SSPSTS.RNE 寄存器位段的指示，读取所有的尾部字节。

19.4.3.3 FIFO 阈值

每个 FIFO 都有一个可配置的触发阈值，用来触发中断。当 RXFIFO 中的条目个数超过了 RXFIFO 的触发阈值（SSPCTL1.RFTH 的值），会产生一个中断（如果该中断被使能）去通知 CPU 清空 RXFIFO。当 TXFIFO 中的条目个数小于或者等于 TXFIFO 的触发阈值（SSPCTL1.TFTH 的值）加 1，会产生一个中断（如果该中断被使能）去通知 CPU 将 TXFIFO 填满。

CPU 可以查询 SSP 状态寄存器来判断 FIFO 中有多少数据或者 FIFO 是满还是空。软件负责选择合适的 RXFIFO/TXFIFO 触发阈值，以防止 RXFIFO 上溢或者 TXFIFO 下溢。

19.4.4 数据帧格式

19.4.4.1 时钟相位和极性控制

SSPCTL1.CLKPOL 位段控制时钟的极性，用来指定时钟是高有效还是低有效，对传输格式没有显著影响。SSPCTL1.CLKPHS 位段控制时钟相位，用来从两个基础的传输格式中选择一个。

对主机和与之通信的从机来说，时钟相位和极性必须是相同的。在一些情况下，时钟相位和极性在多个传输中间会被改变，以允许一个主机设备和不同需求的主机之间进行通信。

时钟信号 SCLK 只在传输数据时才翻转（而不是一直在运行）。

19.4.4.2 SFRM 极性控制

SSPCTL1.FRMINV 位段控制片选信号的极性选择。通常情况下，空闲时，SFRM 为高电平；传输数据时，SFRM 为低电平。

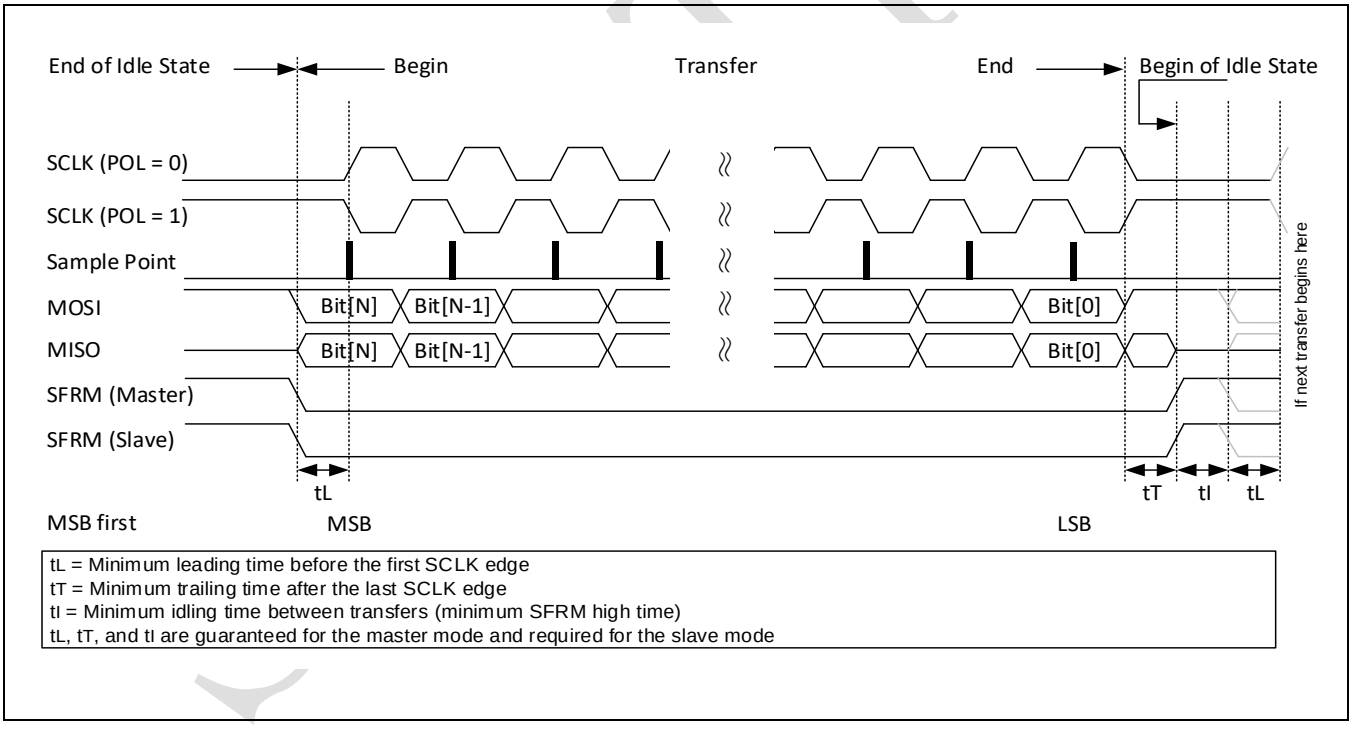
19.4.4.3 PHS = 0 时的帧格式

当 SSP 端口被关闭或者处于空闲模式，SFRM 信号为高电平，MOSI 和 MISO 为高阻态，SCL 为低电平（SSPCTL1.CLKPOL = 0）或者高电平（SSPCTL1.CLKPOL = 1），如图 19-4 所示。

当发送数据准备就绪且 SSPCTL1.CLKPHS 为低电平时，主机和从机的第一个 SCLK 边沿用来采样第一个数据位，SFRM 变低（比 SCLK 第一个边沿早半个时钟周期）并在传输期间保持为低电平。SFRM 在最后一个 SCLK 边沿之后半个时钟周期变高。

发送数据的 MSB 位会在第一个 SCLK 边沿之前半个时钟周期被驱动到 MOSI/MISO 上。发送数据剩下的位会在偶数时钟边沿被驱动到 MOSI/MISO 上。在 SCLK 的奇数边沿会采样 MOSI/MISO 的数据位并移位到 LSB，为了下个时钟边沿的 MOSI/MISO 数据位，MSB 会被移位出去并被锁住。每个数据帧可以传输 1~32 位数据。

图 19-4：SSP 时钟格式 0（PHS = 0）



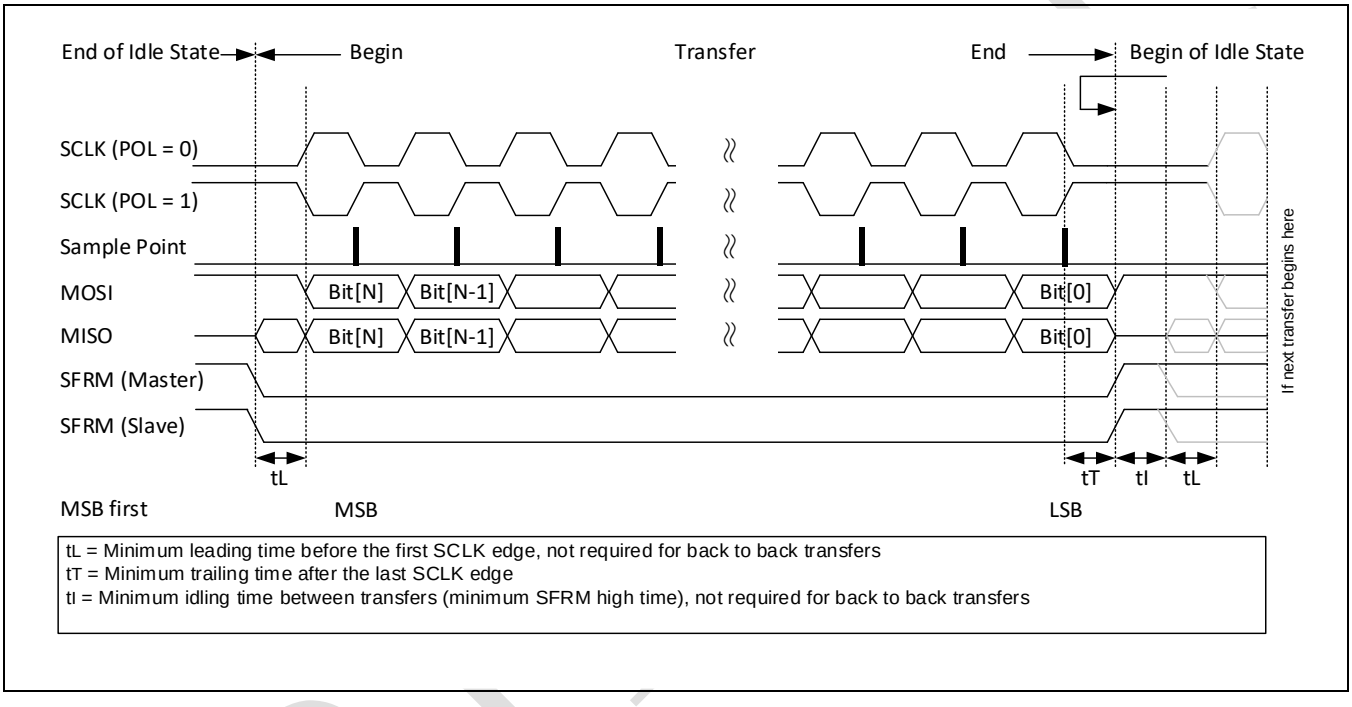
19.4.4.4 PHS = 1 时的帧格式

当 SSP 端口被关闭或者处于空闲模式，SFRM 信号为高电平，MOSI 和 MISO 为高阻态，SCL 为低电平（SSPCTL1.CLKPOL = 0）或者高电平（SSPCTL1.CLKPOL = 1），如图 19-5 所示。

当发送数据准备就绪且 SSPCTL1.CLKPHS 为高电平时，主机和从机的第二个 SCLK 边沿用来采样第一个数据位，SFRM 变低（比 SCLK 第一个边沿早半个时钟周期）并在传输期间保持为低电平。SFRM 在最后一个 SCLK 边沿之后半个时钟周期变高。

发送数据的 MSB 位会在第一个 SCLK 边沿之前半个时钟周期被驱动到 MOSI/MISO 上。发送数据剩下的位会在奇数时钟边沿被驱动到 MOSI/MISO 上。在 SCLK 的偶数边沿会采样 MOSI/MISO 的数据位并移位到 LSB，为了下个时钟边沿的 MOSI/MISO 数据位，MSB 会被移位出去并被锁住。每个数据帧可以传输 1~32 位数据。

图 19-5：SSP 时钟格式 1（PHS = 1）



19.4.4.5 Back to back 传输模式

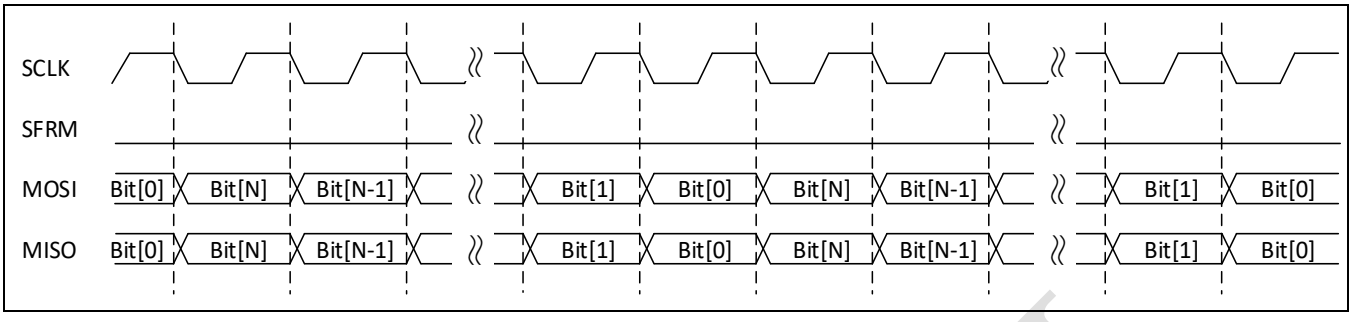
对于 back-to-back 传输，开始和结束与单帧传输时类似的，但是 SFRM 信号在每个数据字传输之间不会被禁能。发送器和接收器要配置为相同的数据字大小，而且内部会跟踪数据帧的开始和结束。传输过程中不存在死亡位（“dead” bits），一个数据帧的 LSB 位传输完后，紧接着传输下一帧的 MSB 位。

为了使能 back to back 模式，需要确保在传输数据时，TXFIFO 中至少有一个数据。如果不想使能 back to back 模式，FIFO 压缩模式应当关闭，并且在当前数据传输完成之前，TXFIFO 应当为空。

SSP 可以作为主机或者从机，当时时钟和片选信号的方向必须是相同的。因此，SSPCTL1.CLKDIR 和 SSPCTL1.FRMDIR 位段必须都是“1”或者“0”。

注意： 当片选信号被禁能时，SSP 的输入时钟不可以是有效的。因此，当 SSP 为从机时，SSPCTL1.SLVCLKSEL 位必须置“1”。

图 19-6：Motorola SPI 帧协议（多帧传输）



19.4.5 SSP 中断

SSP 状态寄存器（SSPSTS）包含溢出错误标志位以及 TXFIFO 和 RXFIFO 服务请求标志位。每一个硬件检测的事件都会产生一个中断请求去通知中断控制器。SSPSTS 寄存器也包含一些标志位用来指示 SSP 接口是否忙于发送数据，TXFIFO 是否非满以及 RXFIFO 是否非空。会有相应的中断信号被送到中断控制器。这些事件可以产生下面的中断：接收器超时、RXFIFO 上溢、RXFIFO 服务请求以及 TXFIFO 服务请求。

产生中断的位段会保持为“1”，直到这些位被写“1”清零。一旦一个状态位被清除，相应的中断也被清除。可读/写的位段被称作状态位（状态位被称为粘滞位，一旦被硬件置“1”，只能通过向该位写“1”清零），只读的位段被称作标志位。向粘滞位写“1”会清零该位，写“0”无效。只读位被置“1”后，可以被硬件自动清零，对只读位写无效的。一些产生中断请求的位段在控制寄存器（SSPCTL0/ SSPCTL1）中有相应的掩码位。

SSPSTS 中的所有位都是只读的，除了 RFOVF，TFUDF 和 RXTO 这些位是可读/写的。

SSPSTS 可读/写的位段的复位值是 0。当 SSPCTL0.EN 位段被清零时，所有 SSPSTS 位段的值都会返回到复位值。

19.5 寄存器

19.5.1 SSP 寄存器列表

表 19-2：SSP 模块基地址

外设模块	基地址
SSP	0x4000 5000

表 19-3：SSP 寄存器列表

寄存器	偏移地址	描述	复位值
SSPCTL0	0x00	SSP 控制寄存器 0	0x00000000
SSPCTL1	0x04	SSP 控制寄存器 1	0x00000000
SSPSTS	0x08	SSP 状态寄存器	0x0000F004
SSPFRC	0x0C	SSP 中断强制寄存器	0x00000000
SSPDATA	0x10	SSP 数据寄存器	0x00000000
SSPTO	0x28	SSP 超时时间寄存器	0x00000000

19.5.2 SSP 寄存器

表 19-4 到表 19-15 提供了 SSP 模块相关寄存器的详细信息。

表 19-4：SSP 控制寄存器 0（SSPCTL0）位段定义

SSPCTL0 (SSP Control Register 0)　Offset: 0x0　Default: 0x00000000
Access: SSP -> SSPCTL0.all

31	30	29	28	27	26	25	24
RESERVED	RESERVED	FPACKEN	RESERVED		RESERVED		
23	22	21	20	19	18	17	16
TFINTMSK	RFINTMSK	RESERVED	ESIZESEL	RESERVED			
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
EN	RESERVED	FRMSEL		SIZESEL			

表 19-5：SSP 控制寄存器 0（SSPCTL0）位段描述

位段	位段名	属性	复位值	描述
31	RESERVED_31	RW	0x0	保留
30	RESERVED_30	RO	0x0	保留
29	FPACKEN	RW	0x0	使能 FIFO 压缩模式 0：关闭 FIFO 压缩模式 1：使能 FIFO 压缩模式
28:27	RESERVED_28_27	RO	0x0	保留

位段	位段名	属性	复位值	描述
26:24	RESERVED_26_24	RW	0x0	保留
23	TFINTMSK	RW	0x0	TXFIFO 下溢中断掩码 0：TFUDF 事件产生中断请求 1：TFUDF 事件不产生中断请求
22	RFINTMSK	RW	0x0	RXFIFO 上溢中断掩码 0：RFOVF 事件产生中断请求 1：RFOVF 事件不产生中断请求
21	RESERVED_21	RO	0x0	保留
20	ESIZESEL	RW	0x0	扩展数据大小选择 0：在 SIZESEL 位段值前补 0，设置 SIZESEL 的范围为 1~16 位 1：在 SIZESEL 位段值前补 1，设置 SIZESEL 的范围为 17~32 位
19:8	RESERVED_19_8	RO	0x0	保留
7	EN	RW	0x0	使能 SSP 模块 0：关闭 SSP 模块 1：使能 SSP 模块
6	RESERVED_6	RO	0x0	保留
5:4	FRMSEL	RW	0x0	帧格式选择 00：Motorola SPI 其他值：无效
3:0	SIZESEL	RW	0x0	数据帧大小 0000：1 位（ESIZESEL=0）或 17 位（ESIZESEL=1） 0001：2 位（ESIZESEL=0）或 18 位（ESIZESEL=1） 0010：3 位（ESIZESEL=0）或 19 位（ESIZESEL=1） 0011：4 位（ESIZESEL=0）或 20 位（ESIZESEL=1） 0100：5 位（ESIZESEL=0）或 21 位（ESIZESEL=1） 0101：6 位（ESIZESEL=0）或 22 位（ESIZESEL=1） 0110：7 位（ESIZESEL=0）或 23 位（ESIZESEL=1） 0111：8 位（ESIZESEL=0）或 24 位（ESIZESEL=1） 1000：9 位（ESIZESEL=0）或 25 位（ESIZESEL=1） 1001：10 位（ESIZESEL=0）或 26 位（ESIZESEL=1） 1010：11 位（ESIZESEL=0）或 27 位（ESIZESEL=1） 1011：12 位（ESIZESEL=0）或 28 位（ESIZESEL=1） 1100：13 位（ESIZESEL=0）或 29 位（ESIZESEL=1） 1101：14 位（ESIZESEL=0）或 30 位（ESIZESEL=1） 1110：15 位（ESIZESEL=0）或 31 位（ESIZESEL=1） 1111：16 位（ESIZESEL=0）或 32 位（ESIZESEL=1）

表 19-6：SSP 控制寄存器 1（SSPCTL1）位段定义

SSPCTL1 (SSP Control Register 1) Offset: 0x4 Default: 0x00000000
Access: SSP -> SSPCTL1.all

31	30	29	28	27	26	25	24
TXTRITIME	TXTRIEN	RESERVED	SLVCLKSEL	RESERVED		CLKDIR	FRMDIR
23	22	21	20	19	18	17	16
RXONLY	TRAIL	DMATXEN	DMARXEN	RESERVED	RESERVED		FRMINV
15	14	13	12	11	10	9	8
RESERVED	RESERVED	RFTH				TFTH	
7	6	5	4	3	2	1	0
TFTH		RESERVED	CLKPHS	CLKPOL	RESERVED	TFIE	RFIE

表 19-7：SSP 控制寄存器 1（SSPCTL1）位段描述

位段	位段名	属性	复位值	描述
31	TXTRITIME	RW	0x0	TXD 三态输出时刻选择 0：SSP_TXD 输出在 LSB 位开始后的 1/2 时钟周期变为三态 1：SSP_TXD 输出在 LSB 位结束时的时钟边沿变为三态
30	TXTRIEN	RW	0x0	使能 TXD 三态输出 0：SSP_TXD 输出信号不是三态 1：当不发送数据时，SSP_TXD 输出为三态
29	RESERVED_29	RW	0x0	保留
28	SLVCLKSEL	RW	0x0	从机时钟控制 0：当 SSPCTL1.CLKDIR=1 时，时钟输入 SSP_SCLK 无效 1：当 SSPCTL1.CLKDIR=1 时，时钟输入 SSP_SCLK 有效
27:26	RESERVED_27_26	RO	0x0	保留
25	CLKDIR	RW	0x0	SSP 时钟信号 SSP_SCLK 方向 0：主机模式，SSP 端口驱动时钟信号 SSP_SCLK 1：从机模式，SSP 端口接收时钟信号 SSP_SCLK
24	FRMDIR	RW	0x0	SSP 片选信号 SSP_SFRM 方向 0：主机模式，SSP 端口驱动片选信号 SSP_SFRM 1：从机模式，SSP 端口接收片选信号 SSP_SFRM
23	RXONLY	RW	0x0	只收模式 0：收发模式 1：只收模式，SSP_TXD 输出高阻
22	TRAIL	RW	0x0	尾部字节控制 0：尾随字节由 CPU 处理 1：尾随字节由 DMA 处理

位段	位段名	属性	复位值	描述
21	DMATXEN	RW	0x0	DMA 发送服务请求使能 0：禁用 1：使能
20	DMARXEN	RW	0x0	DMA 接收服务请求使能 0：禁用 1：使能
19	RESERVED_19	RW	0x0	保留
18:17	RESERVED_18_17	RO	0x0	保留
16	FRMINV	RW	0x0	反转片选信号极性 0：SSP_SFRM 空闲时为高电平，低电平有效 1：SSP_SFRM 空闲时为低电平，高电平有效
15:14	RESERVED_15_14	RW	0x0	保留
13:10	RFTH	RW	0x0	RXFIFO 中断触发阈值 本位段的值应该为预设的阈值减 1。
9:6	TFTH	RW	0x0	TXFIFO 中断触发阈值 本位段的值应该为预设的阈值减 1。
5	RESERVED_5	RO	0x0	保留
4	CLKPHS	RW	0x0	Motorola SPI SSP_SCLK 相位配置 0：在帧传输开始后的第一个 SSP_SCLK 边沿开始采样数据 1：在帧传输开始后的第二个 SSP_SCLK 边沿开始采样数据
3	CLKPOL	RW	0x0	Motorola SPI SSP_SCLK 极性配置 0：SSP_SCLK 空闲时为低电平 1：SSP_SCLK 空闲时为高电平
2	RESERVED_2	RW	0x0	保留
1	TFIE	RW	0x0	使能 TXFIFO 中断 0：关闭 TXFIFO 中断 1：使能 TXFIFO 中断
0	RFIE	RW	0x0	使能 RXFIFO 中断 0：关闭 RXFIFO 中断 1：使能 RXFIFO 中断

表 19-8：SSP 状态寄存器（SSPSTS）位段定义

SSPSTS (SSP Status Register) Offset: 0x8 Default: 0x0000F004 Access: SSP -> SSPSTS.all							
31	30	29	28	27	26	25	24
RFODDSTS	TFODDSTS	RESERVED					
23	22	21	20	19	18	17	16
RESERVED	SLVCLKSTS	TFUDF	RESERVED	RXTO	RESERVED		
15	14	13	12	11	10	9	8
RFLVL				TFLVL			
7	6	5	4	3	2	1	0
RFOVF	RFS	TFS	BUSY	RNE	TNF	RESERVED	

表 19-9：SSP 状态寄存器（SSPSTS）位段描述

位段	位段名	属性	复位值	描述
31	RFODDSTS	RO	0x0	RXFIFO 奇数样本数据状态 本位段仅在 FIFO 压缩模式使能时使用。 0：RXFIFO 条目有 2 个样本数据 1：RXFIFO 条目有一个样本数据
30	TFODDSTS	RO	0x0	TXFIFO 奇数样本数据状态 总的样本数据个数为 TFLVL*2+TFODDSTS 本位段仅在 FIFO 压缩模式使能时使用。 0：TXFIFO 条目有偶数个样本数据 1：TXFIFO 条目有奇数个样本数据
29:24	RESERVED_29_24	RO	0x0	保留
23	RESERVED_23	RW	0x0	保留
22	SLVCLKSTS	RO	0x0	从机时钟状态 0：从机 SSP 时钟准备就绪 1：从机 SSP 忙于同步信号
21	TFUDF	RW	0x0	TXFIFO 下溢标志 0：读到 0 表明没有 TXFIFO 下溢；写 0 无效 1：读到 1 表明发生 TXFIFO 下溢；写 1 清除该位
20	RESERVED_20	RO	0x0	保留
19	RXTO	RW	0x0	接收器超时标志 0：读到 0 表明没有接收超时；写 0 无效 1：读到 1 表明发生接收超时；写 1 清除该位
18:16	RESERVED_18_16	RO	0x0	保留
15:12	RFLVL	RO	0xF	RXFIFO 水平，即数据（32 位）个数 本位段的值为 RXFIFO 中的条目个数减 1。

位段	位段名	属性	复位值	描述
11:8	TFLVL	RO	0x0	TXFIFO 水平，即数据（32 位）个数 本位段的值为 TXFIFO 中的条目个数
7	RFOVF	RW	0x0	RXFIFO 上溢标志 0：读到 0 表明没有 RXFIFO 上溢；写 0 无效 1：读到 1 表明发生 RXFIFO 上溢；写 1 清除该位
6	RFS	RO	0x0	RXFIFO 服务请求标志 0：RXFIFO 水平等于或者小于 RXFIFO 阈值（RFTH）或者 SSP 已被关闭 1：RXFIFO 水平超过 RXFIFO 阈值（RFTH）
5	TFS	RO	0x0	TXFIFO 服务请求标志 0：TXFIFO 水平超过阈值（TFTH+1）或者 SSP 已被关闭 1：TXFIFO 水平等于或者小于阈值（TFTH+1）
4	BUSY	RO	0x0	SSP 忙状态 0：SSP 处于空闲状态或者被关闭 1：SSP 正在发送或者接收数据
3	RNE	RO	0x0	RXFIFO 非空 0：RXFIFO 空 1：RXFIFO 非空
2	TNF	RO	0x1	TXFIFO 非满 0：TXFIFO 满 1：TXFIFO 非满
1:0	RESERVED_1_0	RO	0x0	保留

表 19-10：SSP 中断强制寄存器（SSPFRC）位段定义

SSPFRC (SSP Interrupt Test Register) Offset: 0xC Default: 0x00000000 Access: SSP -> SSPFRC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RFOVF	RFREQ	TFREQ	RESERVED				

表 19-11：SSP 中断强制寄存器（SSPFRC）位段描述

位段	位段名	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	保留
7	RFOVF	RW	0x0	强制 RXFIFO 上溢中断（不可屏蔽） 0：写 0 释放中断强制 1：写 1 强制 RXFIFO 上溢中断 本位段需要手动清零。
6	RFREQ	RW	0x0	强制 RXFIFO 服务请求中断（不可屏蔽） 0：写 0 释放中断强制 1：写 1 强制 RXFIFO 服务请求中断 本位段需要手动清零。
5	TFREQ	RW	0x0	强制 TXFIFO 服务请求中断（不可屏蔽） 0：写 0 释放中断强制 1：写 1 强制 TXFIFO 服务请求中断 本位段需要手动清零。
4:0	RESERVED_4_0	RO	0x0	保留

表 19-12：SSP 数据寄存器（SSPDATA）位段定义

SSPDATA (SSP Data Register) Offset: 0x10 Default: 0x00000000 Access: SSP -> SSPDATA.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 19-13：SSP 数据寄存器（SSPDATA）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RW	0x0	要写入 TXFIFO 的数据或者从 RXFIFO 读取的数据

表 19-14：SSP 超时时间寄存器（SSPTO）位段定义

SSPTO (SSP Time Out Register) Offset: 0x28 Default: 0x00000000 Access: SSP -> SSPTO.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 19-15：SSP 超时时间寄存器（SSPTO）位段描述

位段	位段名	属性	复位值	描述
31:24	RESERVED_31_24	RO	0x0	保留
23:0	VAL	RW	0x0	超时间隔定义为 CLK_SSP*（VAL+1）个时钟周期

20 嵌入式 Flash 存储器（FLASH）

20.1 Flash 主要特性

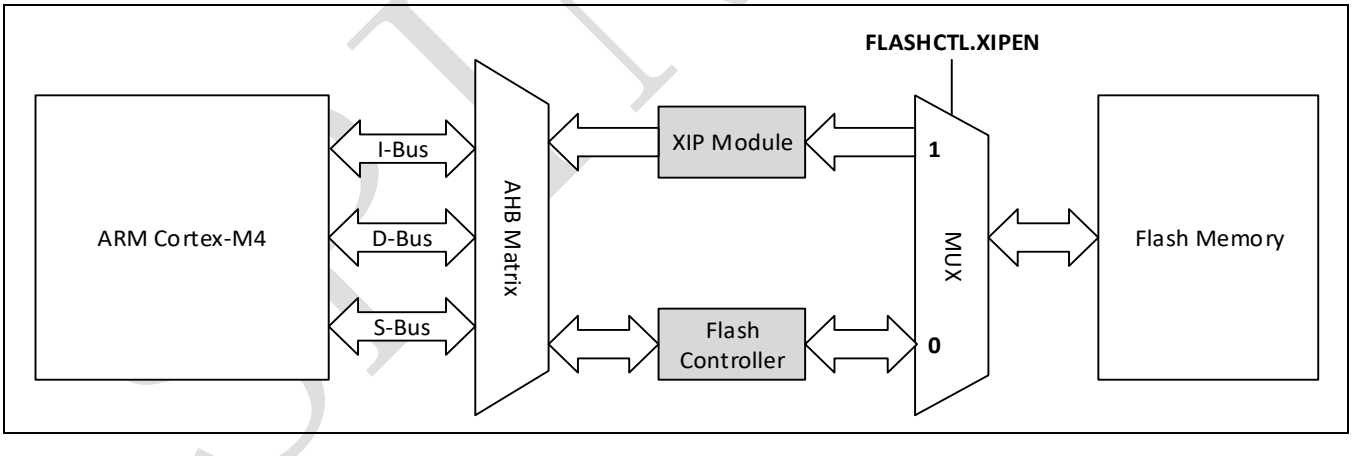
SPC2168 Flash 存储器模块的主要特性如下：

- 存储器最高容量可达 512KB
- 存储器构成：
  - 主存储区块：包含 1024 扇区，每个扇区包含 128 x 32 比特位
  - NVR 存储区块：包含 2 个扇区，每个扇区包含 128 x 32 比特位
- 擦写寿命：最少 100000 次擦写周期 @ T_J=85 ºC
- 数据保存时间：超过 10 年 @ T_J=85 ºC
- 误码检查和校正（Error Checking and Correction, ECC）

Flash 存储器接口特性：

- XIP（Execute-In-Place）读接口
- Flash 控制器接口：支持读/编程/擦除操作
- 读出/写入保护
- 分区保护

图 20-1：Flash 存储器访问接口



20.2 Flash 架构

Flash 存储器由一个主存储区和一个 NVR 存储区构成。主存储区由 1024 个扇区构成，每个扇区包含 128 个基本存储器单元（32-bit）；NVR 存储区由 2 个扇区构成，每个扇区包含 128 个基本存储器单元（32-bit）。具体信息如图 20-2 和表 20-1 所示。

Flash 存储器的基本存储单元为 32 比特位宽（字），用于存储代码和数据常量。

NVR 存储区由两部分构成：

- OTP 存储区 – 该存储区实现 OTP 的方法是通过向该区域第一个存储器单元中写入数据 0x4C4F434B 实现的；如果第一个存储器单元的数据不等于 0x4C4F434B，那么该存储区就只能被当做普通的数据存储区使用。
- 配置字（Configuration Words）存储区 – 该区域的配置字用于实现分区加密保护功能以及 WDT 的开机使能等。

对 Flash 主存储区和 NVR 存储区的写操作是通过 Flash 控制器接口实现的，如图 20-1 所示。编程和擦除操作所需要的高电压是由 Flash 模块内部产生的。

Flash 主存储区有两种保护模式防止非法的访问（读/写/擦除）：

- 扇区写入保护
- 读保护

注意： 在使用 Flash 控制器接口对存储器进行操作之前，必须将 XIP 接口关闭（FLASHCTL.XIPEN = 0）。也就是说，在同一时刻，Flash 控制器接口和 XIP 接口只能有一个接口处于工作状态。

图 20-2：Flash 模块逻辑结构

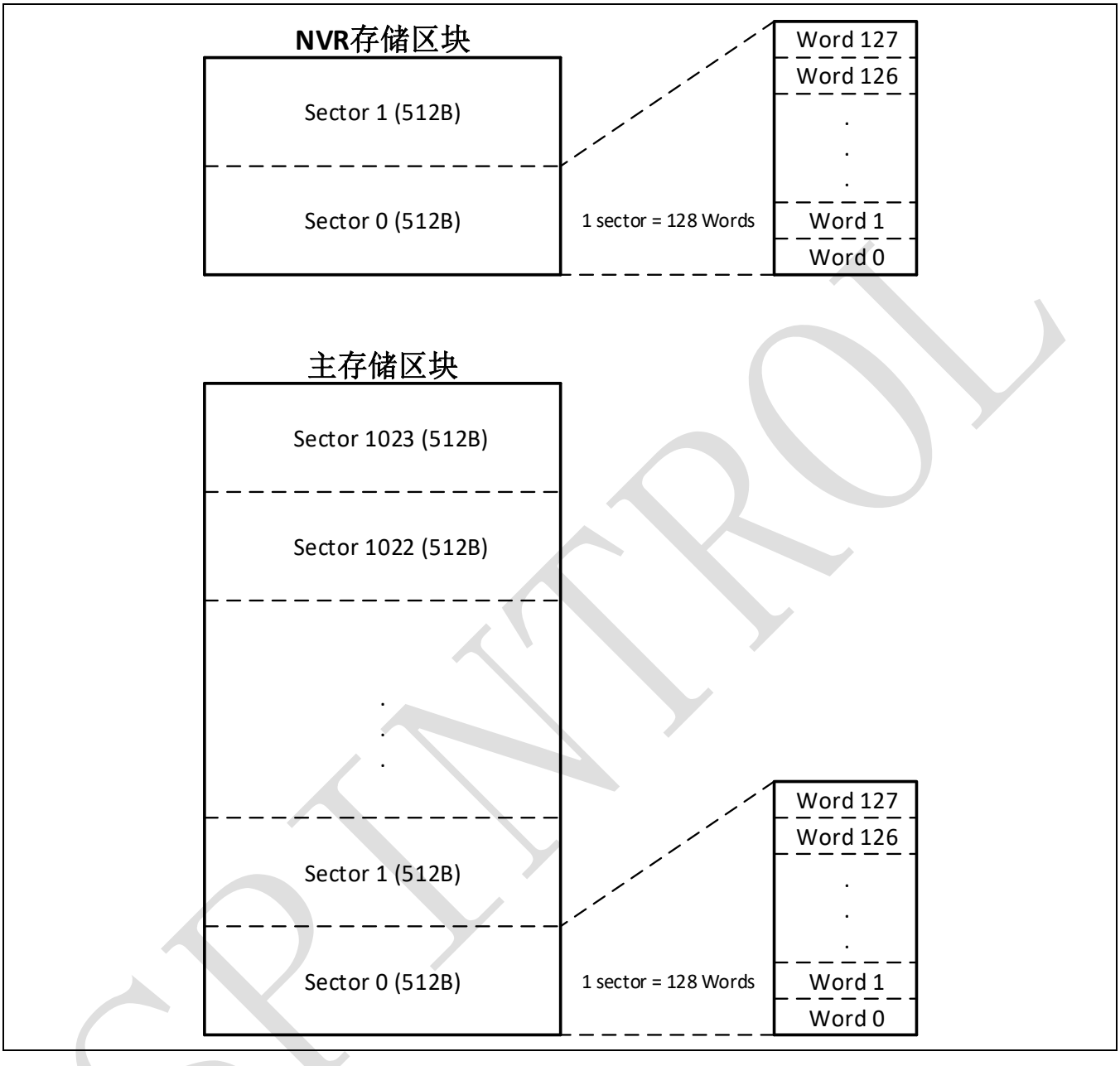


表 20-1：Flash 模块构成

区块	名称	起始地址	大小
主存储区	扇区 0	0x1000 0000	512 bytes
	扇区 1	0x1000 0200	512 bytes
	扇区 2	0x1000 0400	512 bytes
	扇区 3	0x1000 0600	512 bytes
	.	. . .	. . .
	扇区 1023	0x1007 FE00	512 bytes
NVR 存储区	OTP 存储区	0x11000400	128 Words
	配置字存储区	0x11000600	128 Words
Flash 存储器接口寄存器	FLASHCTL	0x4000 8800	4 bytes
	FLASHADDR	0x4000 8804	4 bytes
	FLASHDIN	0x4000 8808	4 bytes
	FLASHDOUT	0x4000 880C	4 bytes
	保留	0x4000 8810	28 bytes
	FLASHWP0	0x4000 882C	4 bytes
	FLASHWP1	0x4000 8830	4 bytes
	FLASHWP2	0x4000 8834	4 bytes
	FLASHWP3	0x40008838	4 bytes
	FLASHREGKEY	0x4000 883C	4 bytes

20.3 读操作

CPU 内核可以通过 XIP 接口和 Flash 控制器接口从 Flash 模块中读取数据。

20.3.1 XIP 接口读操作

通过 XIP 接口，Flash 存储器可以被看做一个普通的存储空间，能够以字节、半字或者字的方式直接通过地址访问。XIP 接口的主要任务就是产生读 Flash 存储器所需要的控制信号。Cortex-M4 内核使用 XIP 接口在 I-Code 总线上取指令，在 D-Code 总线上取数据。

读访问操作可能会存在延迟（Latency）的情况。这个延迟是指一个正在进行的读操作所需要的等待状态（Wait state）数。等待状态数是系统时钟 SYSCLK 周期和 Flash 存储器访问时间的比值。具体信息如表 20-2 所示。

表 20-2：Flash 存储器读访问操作延迟

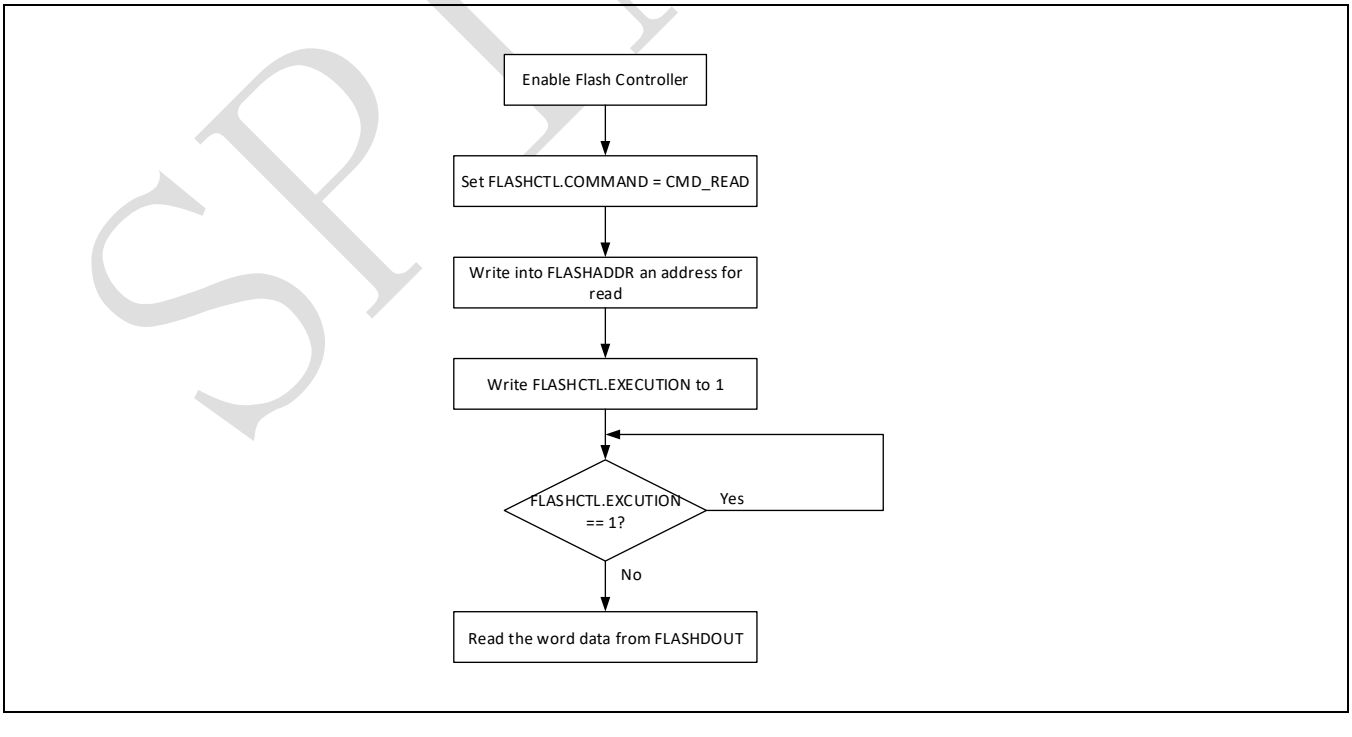
系统时钟频率	等待状态数
0 < SYSCLK ≤ 25 MHz	0
25 MHz < SYSCLK ≤ 50 MHz	1
50 MHz < SYSCLK ≤ 75 MHz	2
75 MHz < SYSCLK ≤ 100 MHz	3
100 MHz < SYSCLK ≤ 125 MHz	4
125 MHz < SYSCLK ≤ 150 MHz	5
150 MHz < SYSCLK ≤ 175 MHz	6
175 MHz < SYSCLK ≤ 200 MHz	7

20.3.2 Flash 控制器接口读操作

CPU 内核也可以通过 Flash 控制器接口从 Flash 存储器中读取数据，但是只能以字（32 比特）的方式读取。该读操作的过程如下：

- 设置 FLASHCTL 寄存器 COMMAND 位的值为 CMD_READ
- 设置 FLASHADDR 寄存器的值为要读取的存储器单元的地址
- 将 FLASHCTL 寄存器的 EXECUTION 位的值置“1”
- 等待 EXECUTION 位的值复位
- 从 FLASHDOUT 寄存器中读取字（32 比特）数据

图 20-3：Flash 存储器读操作流程



20.4 编程和擦除操作

Flash 存储器的编程和擦除操作是通过 Flash 控制器接口实现的。Flash 控制器接口包含 9 个 32 位的寄存器：

- Flash 控制寄存器（FLASHCTL）
- Flash 地址寄存器（FLASHADDR）
- Flash 数据输入寄存器（FLASHDIN）
- Flash 数据输出寄存器（FLASHDOUT）
- Flash 写保护寄存器（FLASHWP0 ~ FLASHWP3）
- Flash 控制器写使能寄存器（FLASHREGKEY）

20.4.1 使能 Flash 控制器

在芯片复位后，Flash 控制器处于关闭状态，可以通过将 FLASHCTL.XIPEN 设置为 0 来使能 Flash 控制器。然而，芯片上电后，FLASHCTL 寄存器处于写保护状态。因此，需要先向 FLASHREGKEY 寄存器中写入解锁值（0x1ACCE551）以便使能对这些被写保护的寄存器的写访问。

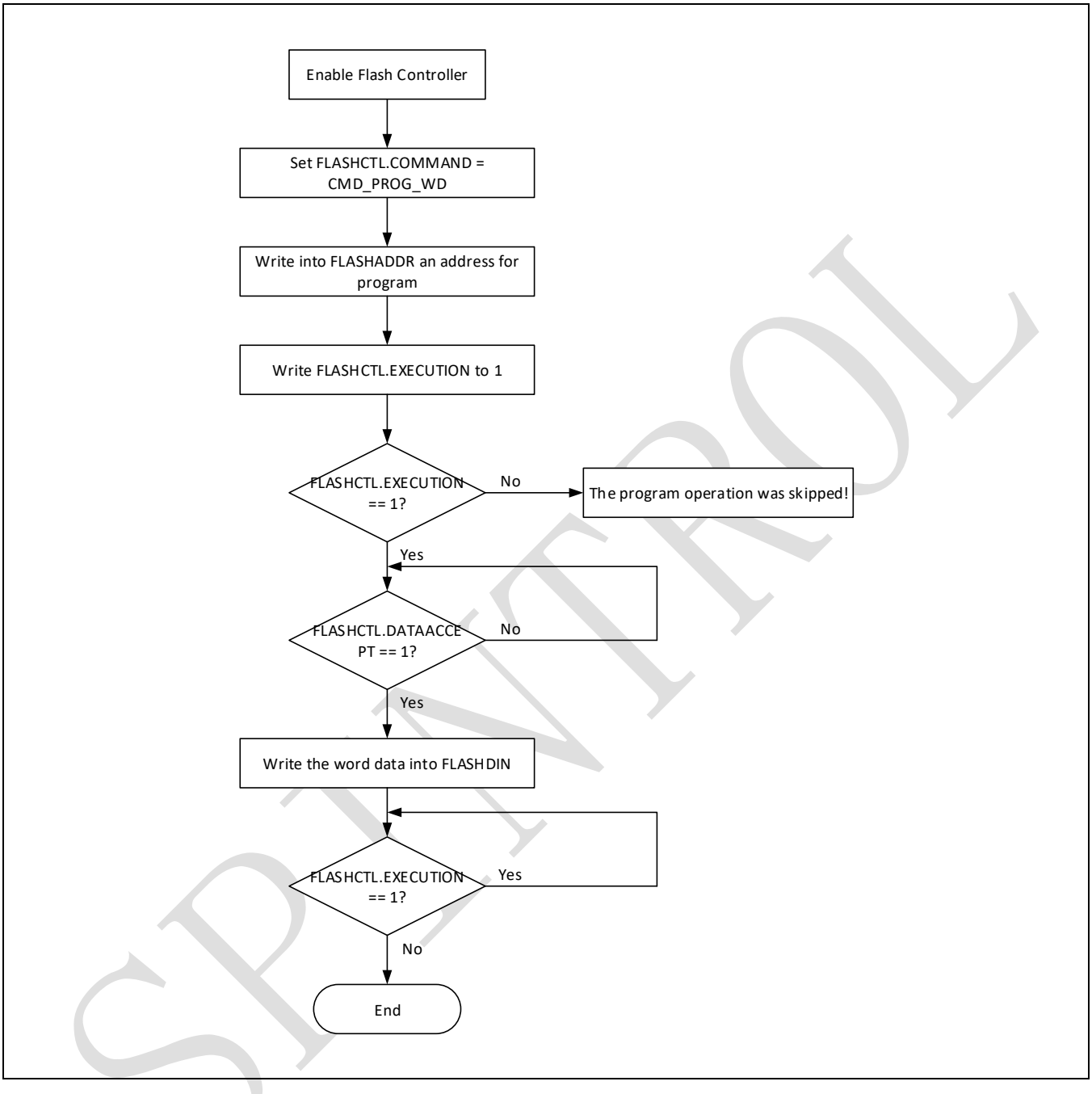
20.4.2 Flash 存储器编程操作

Flash 存储器的编程操作粒度可以到一个字（32 比特）。编程操作的过程如下：

- 设置 FLASHCTL 寄存器的 COMMAND 位的值为 CMD_PROG_WD
- 设置 FLASHADDR 寄存器的值为要编程的存储器单元的地址
- 将 FLASHCTL 寄存器的 EXECUTION 位的值置“1”，然后再检查 EXCUTION 位的值是否被置“1”；如果未被置“1”，则说明该存储器单元不可以被编程
- 等待 FLASHCTL 寄存器的 DATAACCEPT 位的值被置“1”
- 向 FLASHDIN 寄存器中写入一个 32 位数据
- 等待 FLASHCTL 寄存器的 EXECUTION 位的值复位

如果要编程的主存储器单元被设置了写保护（通过设置 FLASHWPx 寄存器），那么 FLASHCTL 寄存器的 EXECUTION 位的值就不能够被置“1”，Flash 控制器会忽略对该存储器单元的编程操作。

图 20-4：Flash 存储器编程操作流程



20.4.3 Flash 存储器擦除操作

Flash 存储器的主存储区块可以按扇区擦除或者整片擦除，而 NVR 存储区块只能够按扇区擦除。

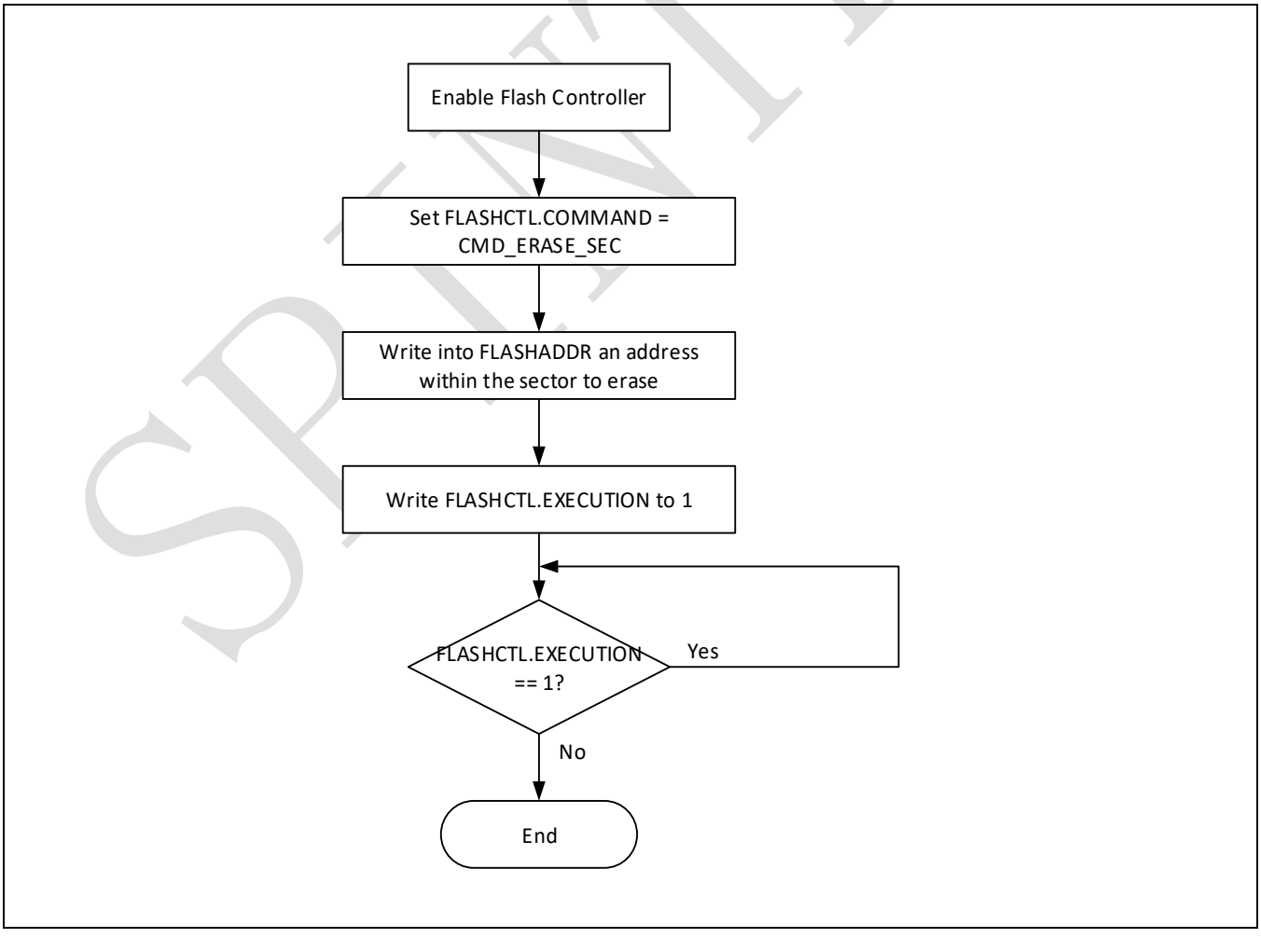
扇区擦除

Flash 存储器的扇区擦除可以通过 Flash 控制器的扇区擦除特性来实现。擦除一个扇区的过程如下：

- 设置 FLASHCTL 寄存器的 COMMAND 位的值为 CMD_ERASE_SEC
- 设置 FLASHADDR 寄存器的值为要擦出的扇区地址
- 将 FLASHCTL 寄存器的 EXECUTION 位的值置“1”
- 等待 EXECUTION 位的值复位

如果要擦除的扇区被设置了写保护（通过设置 FLASHWPx 寄存器），那么 FLASHCTL 寄存器的 EXECUTION 位的值就不能够被置“1”，Flash 控制器会忽略对该扇区的擦除操作。

图 20-5：Flash 存储器扇区擦除操作流程

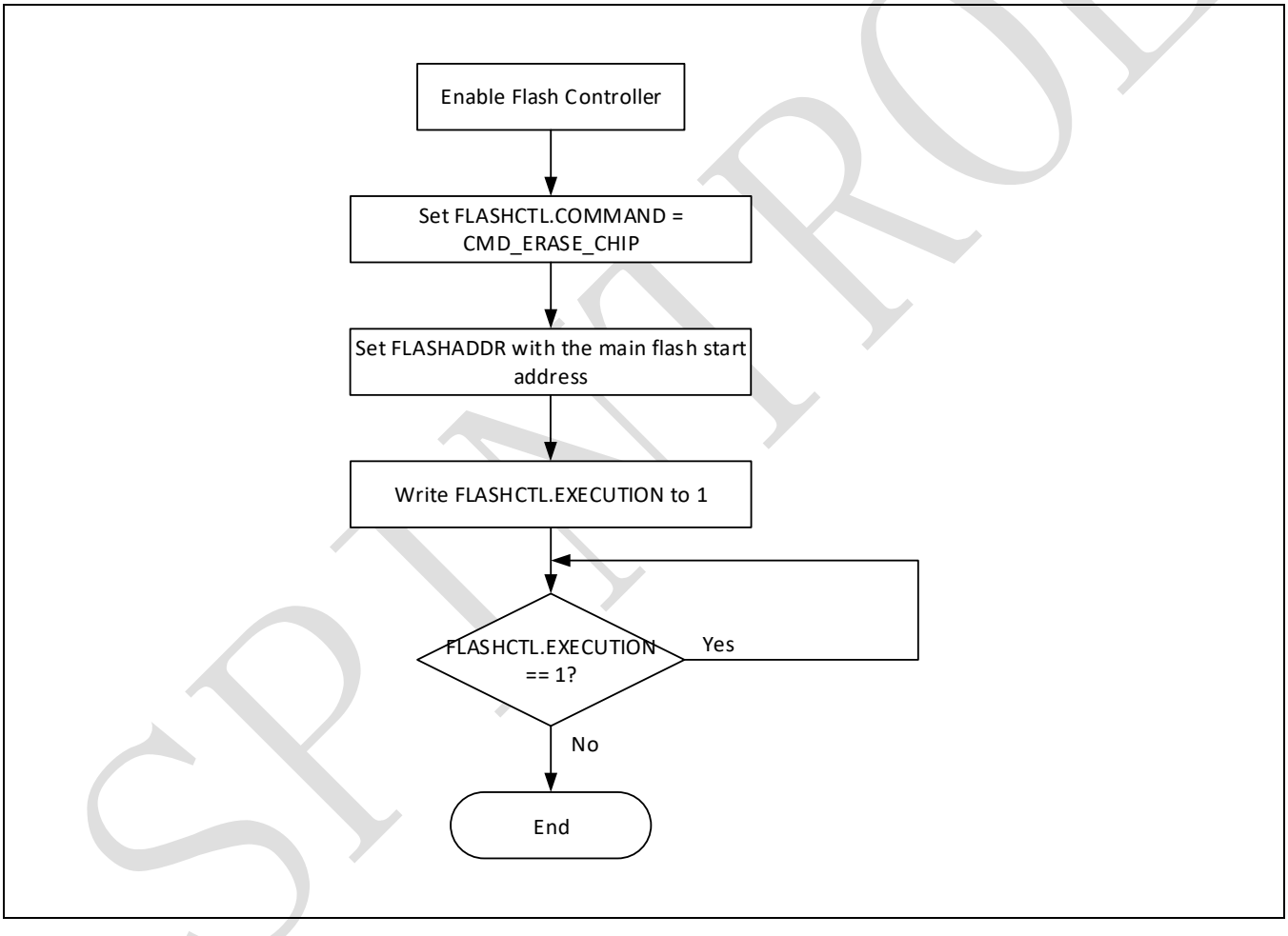


整片擦除

整片擦除指令用于擦除 Flash 主存储区的所有扇区。NVR 存储区不会受到整片擦除指令的影响。推荐的整片擦除过程如下：

- 设置 FLASHCTL 寄存器的 COMMAND 位的值为 CMD_ERASE_CHIP
- 设置 FLASHADDR 寄存器的值为 Flash 主存储区的起始地址
- 将 FLASHCTL 寄存器的 EXECUTION 位的值置“1”
- 等待 EXECUTION 位的值复位

图 20-6：Flash 存储器整片擦除操作流程



20.5 保护机制

20.5.1 分区保护

Flash 主存储区可以被划分为四个代码分区（Flash 分区）。分区保护技术是一种先进的代码保护机制，能够保证多个用户安全地共享同一个芯片的内部资源。一旦一个 Flash 分区被使能后：

- 其他 Flash 分区对该 Flash 分区的访问会被严重的限制。运行在其他 Flash 分区的代码只能够跳转到该 Flash 分区执行程序，而对该 Flash 分区的读、编程以及擦除操作都是被禁止的。
- 通过芯片的调试接口访问该 Flash 分区也是不被允许的。

分区保护功能可以通过设置在 NVR 存储区的配置字（Configuration Words）来激活。当配置字设置完成后，需要芯片再次复位启动后，新的配置字才会被芯片重新加载并起作用。

关闭分区保护的方法如下：

- 首先，擦除整个配置字区域（扇区）。这样做的结果是，配置字的值都会变成 0xFFFFFFFF。在这种情形下，分区保护还是处于使能状态。
- 然后，复位芯片，配置字会被重新加载，此时分区保护功能就被关闭了。

注意： 分区保护功能不包括 DRAM。

擦除配置字区域会自动触发 Flash 主存储区的整片擦除操作。

20.5.2 写保护

Flash 主存储区的扇区可以被保护起来，以免遭程序跑飞而导致的非法写入操作。Flash 主存储区的写保护的可以实现单个扇区的保护。如果对一个使能了写保护的扇区进行编程或者擦除操作，该操作会被忽略掉。

扇区的写保护功能可以通过将 FLASHWPx 寄存器相应位的值置“1”来激活；而将 FLASHWPx 寄存器相应位的值清零会关闭相应扇区的写保护功能。

20.6 Flash 时序设置

为了防止 Flash 存储器出现异常，对 Flash 的时序是有要求的。Flash 的时序是一系列事件之间时序关系的数值。在 XIP 接口和 Flash 控制器中设计有一个有限状态机（FSM），该有限状态机通过对时钟周期的计数判断是否超时，从而保证时序的要求。为了方便用户设置 Flash 时序，在 SDK 中提供了相应的函数 FLASH_SetTiming。该函数以 CPU 内核时钟（HCLK）频率作为输入参数，可以自行设置 Flash 相关的时序参数。

无论在何时，只要想增加 HCLK 的频率，就必须在改变 HCLK 频率前调用 FLASH_SetTiming 函数，这是增加 Flash 时序参数的数值所要求的。

无论在何时，只要想降低 HCLK 的频率，就必须在改变 HCLK 频率后调用 FLASH_SetTiming 函数，这是降低 Flash 时序参数的数值所要求的。

20.7 配置字(Configuration Words)

跟分区保护功能相关的配置字总共有 28 个，每个分区有 7 个。这些配置字由终端用户根据应用的需求进行设置。

配置字的构成及其功能描述如表 20-3 所示。配置字的值可以通过 XIP 接口或者 Flash 控制器接口按照表 20-3 所示的地址进行读取。新写入的配置字需要在芯片复位并重新加载后才能生效。

表 20-3：配置字的构成及其描述

地址	配置字名称	配置字描述
0x11000600	ZONE0_INFO_LOCK	锁定分区 0 的配置字(0x11000600 – 0x1100063F) 0xFFFFFFFF：分区 0 的配置字可以被编程 其他值：分区 0 的配置字不可以被编程
0x11000604	ZONE0_ENCRYPT	Flash 分区 0 代码的加密状态 0xDEC0DE：Flash 分区 0 的代码已被加密 其他值：Flash 分区 0 的代码是原始代码
0x11000608	ZONE0_DECRYPT	Flash 分区 0 代码的解密状态 0xAA621623：成功解密 Flash 分区 0 的代码 0x1E55051：未成功解密 Flash 分区 0 的代码 其他值：无效值
0x1100060C	ZONE0_FLASH_PROT	Flash 分区 0 保护字段 0xFFFFFFFF：关闭 Flash 分区 0 保护 其他值：使能 Flash 分区 0 保护
0x11000610	ZONE0_FLASH_ADDR	Flash 分区 0 起始地址
0x11000614	ZONE0_RAM_PROT	RAM 分区 0 保护字段 0xFFFFFFFF：关闭 RAM 分区 0 保护 其他值：使能 RAM 分区 0 保护
0x11000618	ZONE0_RAM_ADDR	RAM 分区 0 起始地址
0x1100061C ~ 0x1100063F	Reserved for ZONE0	分区 0 保留配置字段
0x11000640	ZONE1_INFO_LOCK	锁定分区 1 的配置字(0x11000640 – 0x1100067F) 0xFFFFFFFF：分区 1 的配置字可以被编程 其他值：分区 1 的配置字不可以被编程
0x11000644	ZONE1_ENCRYPT	Flash 分区 1 代码的加密状态 0xDEC0DE：Flash 分区 1 的代码已被加密 其他值：Flash 分区 1 的代码是原始代码
0x11000648	ZONE1_DECRYPT	Flash 分区 1 代码的解密状态 0xAA621623：成功解密 Flash 分区 1 的代码 0x1E55051：未成功解密 Flash 分区 1 的代码 其他值：无效值
0x1100064C	ZONE1_FLASH_PROT	Flash 分区 1 保护字段 0xFFFFFFFF：关闭 Flash 分区 1 保护

地址	配置字名称	配置字描述
		其他值：使能 Flash 分区 1 保护
0x11000650	ZONE1_FLASH_ADDR	Flash 分区 1 起始地址
0x11000654	ZONE1_RAM_PROT	RAM 分区 1 保护字段 0xFFFFFFFF：关闭 RAM 分区 1 保护 其他值：使能 RAM 分区 1 保护
0x11000658	ZONE1_RAM_ADDR	RAM 分区 1 起始地址
0x1100065C ~ 0x1100067F	Reserved for ZONE1	分区 1 保留配置字段
0x11000680	ZONE2_INFO_LOCK	锁定分区 2 的配置字(0x11000680 – 0x110006BF) 0xFFFFFFFF：分区 2 的配置字可以被编程 其他值：分区 2 的配置字不可以被编程
0x11000684	ZONE2_ENCRYPT	Flash 分区 2 代码的加密状态 0xDEC0DE：Flash 分区 2 的代码已被加密 其他值：Flash 分区 2 的代码是原始代码
0x1100688	ZONE2_DECRYPT	Flash 分区 2 代码的解密状态 0xAA621623：成功解密 Flash 分区 2 的代码 0x1E55051：未成功解密 Flash 分区 2 的代码 其他值：无效值
0x1100068C	ZONE2_FLASH_PROT	Flash 分区 2 保护字段 0xFFFFFFFF：关闭 Flash 分区 2 保护 其他值：使能 Flash 分区 2 保护
0x11000690	ZONE2_FLASH_ADDR	Flash 分区 2 起始地址
0x11000694	ZONE2_RAM_PROT	RAM 分区 2 保护字段 0xFFFFFFFF：关闭 RAM 分区 2 保护 其他值：使能 RAM 分区 2 保护
0x11000698	ZONE2_RAM_ADDR	RAM 分区 2 起始地址
0x1100069C ~ 0x110006BF	Reserved for ZONE2	分区 2 保留配置字段
0x110006C0	ZONE3_INFO_LOCK	锁定分区 3 的配置字(0x110006C0 – 0x110006FF) 0xFFFFFFFF：分区 3 的配置字可以被编程 其他值：分区 3 的配置字不可以被编程
0x110006C4	ZONE3_ENCRYPT	Flash 分区 3 代码的加密状态 0xDEC0DE：Flash 分区 3 的代码已被加密 其他值：Flash 分区 3 的代码是原始代码
0x110006C8	ZONE3_DECRYPT	Flash 分区 3 代码的解密状态 0xAA621623：成功解密 Flash 分区 3 的代码 0x1E55051：未成功解密 Flash 分区 3 的代码 其他值：无效值
0x110006CC	ZONE3_FLASH_PROT	Flash 分区 3 保护字段 0xFFFFFFFF：关闭 Flash 分区 3 保护 其他值：使能 Flash 分区 3 保护
0x110006D0	ZONE3_FLASH_ADDR	Flash 分区 3 起始地址
0x110006D4	ZONE3_RAM_PROT	RAM 分区 3 保护字段 0xFFFFFFFF：关闭 RAM 分区 3 保护

地址	配置字名称	配置字描述
		其他值：使能 RAM 分区 3 保护
0x110006D8	ZONE3_RAM_ADDR	RAM 分区 3 起始地址
0x110006DC ~ 0x110006FF	Reserved for ZONE3	分区 3 保留配置字段
0x11000700	WDT_ENABLE	Watchdog 开机使能字段 0xFFFFFFFF：关闭 Watchdog 当芯片启动时 其他值：使能 Watchdog 当芯片启动时
0x11000704	CACHE_ENABLE	Flash 缓存（Cache）开机使能字段 0x1ACCE551：使能 Flash cache 当芯片启动时 其他值：关闭 Flash cache 当芯片启动时
0x11000708 ~ 0x110007FF	保留	未使用

20.8 寄存器

20.8.1 Flash 寄存器列表

表 20-4：Flash 控制器模块基地址

外设模块	基地址
FLASH	0x4000 8800

表 20-5：Flash 控制器寄存器列表

寄存器	偏移地址	描述	复位值
FLASHCTL*	0x00	Flash 控制寄存器	0x00000000
FLASHADDR	0x04	Flash 地址寄存器	0x00000000
FLASHDIN	0x08	Flash 数据输入寄存器	0x00000000
FLASHDOUT	0x0C	Flash 数据输出寄存器	0x00000000
FLASHWP0*	0x2C	Flash 写保护寄存器 0	0x00000000
FLASHWP1*	0x30	Flash 写保护寄存器 1	0x00000000
FLASHWP2*	0x34	Flash 写保护寄存器 2	0x00000000
FLASHWP3*	0x38	Flash 写保护寄存器 3	0x00000000
FLASHREGKEY	0x3C	Flash 控制器写使能寄存器	0x1ACCE551

注意： 被‘*’标记的寄存器只有当 FLASHREGKEY=0x1ACCE551 时才允许写操作。

20.8.2 Flash 寄存器

表 20-6 到表 20-23 列出了 Flash 模块相关寄存器的详细信息。

表 20-6：Flash 控制寄存器（FLASHCTL）位段定义

FLASHCTL (Flash Control Register) Offset: 0x0 Default: 0x00000000 Access: FLASH -> FLASHCTL.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
WITHCACHE	XIPEN	EXECUTION	DATAACCEPT	COMMAND			

表 20-7：Flash 控制寄存器（FLASHCTL）位段描述

位段	位段名称	属性	复位值	描述
31:8	RESERVED_31_8	RO	0x0	Reserved
7	WITHCACHE	RO	0x0	XIP 模块缓存选项 0：XIP 缓存不可用 1：XIP 缓存可用
6	XIPEN	RW	0x0	XIP 模块使能 0：关闭 XIP 模块 1：使能 XIP 模块
5	EXECUTION	RW	0x0	命令处于执行状态 0：FLASH FSM 处于空闲状态 1：FLASH FSM 处于执行状态
4	DATAACCEPT	RO	0x0	当这个位为 0 时，FLASHDIN 寄存器的值不可以被改变
3:0	COMMAND	RW	0x0	Flash 命令 0001：读 0101：字编程 1000：扇区擦除 1010：整片擦除

表 20-8：Flash 地址寄存器（FLASHADDR）位段定义

FLASHADDR (Flash Address Register) Offset: 0x4 Default: 0x00000000 Access: FLASH -> FLASHADDR.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 20-9：Flash 地址寄存器（FLASHADDR）位段描述

位段	位段名称	属性	复位值	描述
31:0	VAL	RW	0x0	Flash 地址

表 20-10：Flash 数据输入寄存器（FLASHDIN）位段定义

FLASHDIN (Flash Data Input Register) Offset: 0x8 Default: 0x00000000 Access: FLASH -> FLASHDIN.all							
31	30	29	28	27	26	25	24
FLASHDIN							
23	22	21	20	19	18	17	16
FLASHDIN							
15	14	13	12	11	10	9	8
FLASHDIN							
7	6	5	4	3	2	1	0
FLASHDIN							

表 20-11：Flash 数据输入寄存器（FLASHDIN）位段描述

位段	位段名称	属性	复位值	描述
31:0	FLASHDIN	RW	0x0	Flash 输入数据

表 20-12：Flash 数据输出寄存器（FLASHDOUT）位段定义

FLASHDOUT (Flash Data Output Register) Offset: 0xC Default: 0x00000000 Access: FLASH -> FLASHDOUT.all							
31	30	29	28	27	26	25	24
FLASHDOUT							
23	22	21	20	19	18	17	16
FLASHDOUT							
15	14	13	12	11	10	9	8
FLASHDOUT							
7	6	5	4	3	2	1	0
FLASHDOUT							

表 20-13：Flash 数据输出寄存器（FLASHDOUT）位段描述

位段	位段名称	属性	复位值	描述
31:0	FLASHDOUT	RO	0x0	Flash 输出数据

表 20-14：Flash 写保护寄存器 0（FLASHWP0）位段定义

FLASHWP0 (Flash Write Protect Register 0) Offset: 0x2C Default: 0x00000000 Access: FLASH -> FLASHWP0.all							
31	30	29	28	27	26	25	24
UNIT31	UNIT30	UNIT29	UNIT28	UNIT27	UNIT26	UNIT25	UNIT24
23	22	21	20	19	18	17	16
UNIT23	UNIT22	UNIT21	UNIT20	UNIT19	UNIT18	UNIT17	UNIT16
15	14	13	12	11	10	9	8
UNIT15	UNIT14	UNIT13	UNIT12	UNIT11	UNIT10	UNIT9	UNIT8
7	6	5	4	3	2	1	0
UNIT7	UNIT6	UNIT5	UNIT4	UNIT3	UNIT2	UNIT1	UNIT0

表 20-15：Flash 写保护寄存器 0（FLASHWP0）位段描述

位段	位段名称	属性	复位值	描述
31	UNIT31	RW	0x0	Flash 单元 31 写保护 （0x1001_F000 → 0x1001_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
30	UNIT30	RW	0x0	Flash 单元 30 写保护 （0x1001_E000 → 0x1001_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
29	UNIT29	RW	0x0	Flash 单元 29 写保护 （0x1001_D000 → 0x1001_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
28	UNIT28	RW	0x0	Flash 单元 28 写保护 （0x1001_C000 → 0x1001_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
27	UNIT27	RW	0x0	Flash 单元 27 写保护 （0x1001_B000 → 0x1001_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
26	UNIT26	RW	0x0	Flash 单元 26 写保护 （0x1001_A000 → 0x1001_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
25	UNIT25	RW	0x0	Flash 单元 25 写保护 （0x1001_9000 → 0x1001_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
24	UNIT24	RW	0x0	Flash 单元 24 写保护 （0x1001_8000 → 0x1001_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
23	UNIT23	RW	0x0	Flash 单元 23 写保护 （0x1001_7000 → 0x1001_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
22	UNIT22	RW	0x0	Flash 单元 22 写保护 （0x1001_6000 → 0x1001_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
21	UNIT21	RW	0x0	Flash 单元 21 写保护 （0x1001_5000 → 0x1001_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
20	UNIT20	RW	0x0	Flash 单元 20 写保护 （0x1001_4000 → 0x1001_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
19	UNIT19	RW	0x0	Flash 单元 19 写保护 （0x1001_3000 → 0x1001_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
18	UNIT18	RW	0x0	Flash 单元 18 写保护 （0x1001_2000 → 0x1001_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
17	UNIT17	RW	0x0	Flash 单元 17 写保护 （0x1001_1000 → 0x1001_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
16	UNIT16	RW	0x0	Flash 单元 16 写保护 （0x1001_0000 → 0x1001_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
15	UNIT15	RW	0x0	Flash 单元 15 写保护 （0x1000_F000 → 0x1000_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
14	UNIT14	RW	0x0	Flash 单元 14 写保护 （0x1000_E000 → 0x1000_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
13	UNIT13	RW	0x0	Flash 单元 13 写保护 （0x1000_D000 → 0x1000_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
12	UNIT12	RW	0x0	Flash 单元 12 写保护 （0x1000_C000 → 0x1000_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
11	UNIT11	RW	0x0	Flash 单元 11 写保护 （0x1000_B000 → 0x1000_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
10	UNIT10	RW	0x0	Flash 单元 10 写保护 （0x1000_A000 → 0x1000_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
9	UNIT9	RW	0x0	Flash 单元 9 写保护 （0x1000_9000 → 0x1000_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
8	UNIT8	RW	0x0	Flash 单元 8 写保护 （0x1000_8000 → 0x1000_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
7	UNIT7	RW	0x0	Flash 单元 7 写保护 （0x1000_7000 → 0x1000_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
6	UNIT6	RW	0x0	Flash 单元 6 写保护 （0x1000_6000 → 0x1000_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
5	UNIT5	RW	0x0	Flash 单元 5 写保护 （0x1000_5000 → 0x1000_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
4	UNIT4	RW	0x0	Flash 单元 4 写保护 （0x1000_4000 → 0x1000_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
3	UNIT3	RW	0x0	Flash 单元 3 写保护 （0x1000_3000 → 0x1000_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
2	UNIT2	RW	0x0	Flash 单元 2 写保护 （0x1000_2000 → 0x1000_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
1	UNIT1	RW	0x0	Flash 单元 1 写保护 （0x1000_1000 → 0x1000_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
0	UNIT0	RW	0x0	Flash 单元 0 写保护 （0x1000_0000 → 0x1000_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

表 20-16：Flash 写保护寄存器 1（FLASHWP1）位段定义

FLASHWP1 (Flash Write Protect Register 1) Offset: 0x30 Default: 0x00000000 Access: FLASH -> FLASHWP1.all							
31	30	29	28	27	26	25	24
UNIT63	UNIT62	UNIT61	UNIT60	UNIT59	UNIT58	UNIT57	UNIT56
23	22	21	20	19	18	17	16
UNIT55	UNIT54	UNIT53	UNIT52	UNIT51	UNIT50	UNIT49	UNIT48
15	14	13	12	11	10	9	8
UNIT47	UNIT46	UNIT45	UNIT44	UNIT43	UNIT42	UNIT41	UNIT40
7	6	5	4	3	2	1	0
UNIT39	UNIT38	UNIT37	UNIT36	UNIT35	UNIT34	UNIT33	UNIT32

表 20-17：Flash 写保护寄存器 1（FLASHWP1）位段描述

位段	位段名称	属性	复位值	描述
31	UNIT63	RW	0x0	Flash 单元 63 写保护 （0x1003_F000 → 0x1003_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
30	UNIT62	RW	0x0	Flash 单元 62 写保护 （0x1003_E000 → 0x1003_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
29	UNIT61	RW	0x0	Flash 单元 61 写保护 （0x1003_D000 → 0x1003_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
28	UNIT60	RW	0x0	Flash 单元 60 写保护 （0x1003_C000 → 0x1003_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
27	UNIT59	RW	0x0	Flash 单元 59 写保护 （0x1003_B000 → 0x1003_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
26	UNIT58	RW	0x0	Flash 单元 58 写保护 （0x1003_A000 → 0x1003_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
25	UNIT57	RW	0x0	Flash 单元 57 写保护 （0x1003_9000 → 0x1003_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
24	UNIT56	RW	0x0	Flash 单元 56 写保护 （0x1003_8000 → 0x1003_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
23	UNIT55	RW	0x0	Flash 单元 55 写保护 （0x1003_7000 → 0x1003_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
22	UNIT54	RW	0x0	Flash 单元 54 写保护 （0x1003_6000 → 0x1003_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
21	UNIT53	RW	0x0	Flash 单元 53 写保护 （0x1003_5000 → 0x1003_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
20	UNIT52	RW	0x0	Flash 单元 52 写保护 （0x1003_4000 → 0x1003_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
19	UNIT51	RW	0x0	Flash 单元 51 写保护 （0x1003_3000 → 0x1003_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
18	UNIT50	RW	0x0	Flash 单元 50 写保护 （0x1003_2000 → 0x1003_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
17	UNIT49	RW	0x0	Flash 单元 49 写保护 （0x1003_1000 → 0x1003_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
16	UNIT48	RW	0x0	Flash 单元 48 写保护 （0x1003_0000 → 0x1003_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
15	UNIT47	RW	0x0	Flash 单元 47 写保护 （0x1002_F000 → 0x1002_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
14	UNIT46	RW	0x0	Flash 单元 46 写保护 （0x1002_E000 → 0x1002_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
13	UNIT45	RW	0x0	Flash 单元 45 写保护 （0x1002_D000 → 0x1002_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
12	UNIT44	RW	0x0	Flash 单元 44 写保护 （0x1002_C000 → 0x1002_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
11	UNIT43	RW	0x0	Flash 单元 43 写保护 （0x1002_B000 → 0x1002_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
10	UNIT42	RW	0x0	Flash 单元 42 写保护 （0x1002_A000 → 0x1002_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
9	UNIT41	RW	0x0	Flash 单元 41 写保护 （0x1002_9000 → 0x1002_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
8	UNIT40	RW	0x0	Flash 单元 40 写保护 （0x1002_8000 → 0x1002_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
7	UNIT39	RW	0x0	Flash 单元 39 写保护 （0x1002_7000 → 0x1002_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
6	UNIT38	RW	0x0	Flash 单元 38 写保护 （0x1002_6000 → 0x1002_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
5	UNIT37	RW	0x0	Flash 单元 37 写保护 （0x1002_5000 → 0x1002_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
4	UNIT36	RW	0x0	Flash 单元 36 写保护 （0x1002_4000 → 0x1002_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
3	UNIT35	RW	0x0	Flash 单元 35 写保护 （0x1002_3000 → 0x1002_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
2	UNIT34	RW	0x0	Flash 单元 34 写保护 （0x1002_2000 → 0x1002_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
1	UNIT33	RW	0x0	Flash 单元 33 写保护 （0x1002_1000 → 0x1002_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
0	UNIT32	RW	0x0	Flash 单元 32 写保护 （0x1002_0000 → 0x1002_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

表 20-18：Flash 写保护寄存器 2（FLASHWP2）位段定义

FLASHWP2 (Flash Write Protect Register 2) Offset: 0x34 Default: 0x00000000 Access: FLASH -> FLASHWP2.all							
31	30	29	28	27	26	25	24
UNIT95	UNIT94	UNIT93	UNIT92	UNIT91	UNIT90	UNIT89	UNIT88
23	22	21	20	19	18	17	16
UNIT87	UNIT86	UNIT85	UNIT84	UNIT83	UNIT82	UNIT81	UNIT80
15	14	13	12	11	10	9	8
UNIT79	UNIT78	UNIT77	UNIT76	UNIT75	UNIT74	UNIT73	UNIT72
7	6	5	4	3	2	1	0
UNIT71	UNIT70	UNIT69	UNIT68	UNIT67	UNIT66	UNIT65	UNIT64

表 20-19：Flash 写保护寄存器 2（FLASHWP2）位段描述

位段	位段名称	属性	复位值	描述
31	UNIT95	RW	0x0	Flash 单元 95 写保护 （0x1005_F000 → 0x1005_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
30	UNIT94	RW	0x0	Flash 单元 94 写保护 （0x1005_E000 → 0x1005_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
29	UNIT93	RW	0x0	Flash 单元 93 写保护 （0x1005_D000 → 0x1005_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
28	UNIT92	RW	0x0	Flash 单元 92 写保护 （0x1005_C000 → 0x1005_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
27	UNIT91	RW	0x0	Flash 单元 91 写保护 （0x1005_B000 → 0x1005_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
26	UNIT90	RW	0x0	Flash 单元 90 写保护 （0x1005_A000 → 0x1005_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
25	UNIT89	RW	0x0	Flash 单元 89 写保护 （0x1005_9000 → 0x1005_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
24	UNIT88	RW	0x0	Flash 单元 88 写保护 （0x1005_8000 → 0x1005_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
23	UNIT87	RW	0x0	Flash 单元 87 写保护 （0x1005_7000 → 0x1005_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
22	UNIT86	RW	0x0	Flash 单元 86 写保护 （0x1005_6000 → 0x1005_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
21	UNIT85	RW	0x0	Flash 单元 85 写保护 （0x1005_5000 → 0x1005_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
20	UNIT84	RW	0x0	Flash 单元 84 写保护 （0x1005_4000 → 0x1005_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
19	UNIT83	RW	0x0	Flash 单元 83 写保护 （0x1005_3000 → 0x1005_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
18	UNIT82	RW	0x0	Flash 单元 82 写保护 （0x1005_2000 → 0x1005_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
17	UNIT81	RW	0x0	Flash 单元 81 写保护 （0x1005_1000 → 0x1005_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
16	UNIT80	RW	0x0	Flash 单元 80 写保护 （0x1005_0000 → 0x1005_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
15	UNIT79	RW	0x0	Flash 单元 79 写保护 （0x1004_F000 → 0x1004_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
14	UNIT78	RW	0x0	Flash 单元 78 写保护 （0x1004_E000 → 0x1004_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
13	UNIT77	RW	0x0	Flash 单元 77 写保护 （0x1004_D000 → 0x1004_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
12	UNIT76	RW	0x0	Flash 单元 76 写保护 （0x1004_C000 → 0x1004_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
11	UNIT75	RW	0x0	Flash 单元 75 写保护 （0x1004_B000 → 0x1004_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
10	UNIT74	RW	0x0	Flash 单元 74 写保护 （0x1004_A000 → 0x1004_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
9	UNIT73	RW	0x0	Flash 单元 73 写保护 （0x1004_9000 → 0x1004_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
8	UNIT72	RW	0x0	Flash 单元 72 写保护 （0x1004_8000 → 0x1004_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
7	UNIT71	RW	0x0	Flash 单元 71 写保护 （0x1004_7000 → 0x1004_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
6	UNIT70	RW	0x0	Flash 单元 70 写保护 （0x1004_6000 → 0x1004_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
5	UNIT69	RW	0x0	Flash 单元 69 写保护 （0x1004_5000 → 0x1004_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
4	UNIT68	RW	0x0	Flash 单元 68 写保护 （0x1004_4000 → 0x1004_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
3	UNIT67	RW	0x0	Flash 单元 67 写保护 （0x1004_3000 → 0x1004_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
2	UNIT66	RW	0x0	Flash 单元 66 写保护 （0x1004_2000 → 0x1004_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
1	UNIT65	RW	0x0	Flash 单元 65 写保护 （0x1004_1000 → 0x1004_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
0	UNIT64	RW	0x0	Flash 单元 64 写保护 （0x1004_0000 → 0x1004_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

表 20-20：Flash 写保护寄存器 3（FLASHWP3）位段定义

FLASHWP3 (Flash Write Protect Register 3) Offset: 0x38 Default: 0x00000000 Access: FLASH -> FLASHWP3.all							
31	30	29	28	27	26	25	24
UNIT127	UNIT126	UNIT125	UNIT124	UNIT123	UNIT122	UNIT121	UNIT120
23	22	21	20	19	18	17	16
UNIT119	UNIT118	UNIT117	UNIT116	UNIT115	UNIT114	UNIT113	UNIT112
15	14	13	12	11	10	9	8
UNIT111	UNIT110	UNIT109	UNIT108	UNIT107	UNIT106	UNIT105	UNIT104
7	6	5	4	3	2	1	0
UNIT103	UNIT102	UNIT101	UNIT100	UNIT99	UNIT98	UNIT97	UNIT96

表 20-21：Flash 写保护寄存器 3（FLASHWP3）位段描述

位段	位段名称	属性	复位值	描述
31	UNIT127	RW	0x0	Flash 单元 127 写保护 （0x1007_F000 → 0x1007_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
30	UNIT126	RW	0x0	Flash 单元 126 写保护 （0x1007_E000 → 0x1007_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
29	UNIT125	RW	0x0	Flash 单元 125 写保护 （0x1007_D000 → 0x1007_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
28	UNIT124	RW	0x0	Flash 单元 124 写保护 （0x1007_C000 → 0x1007_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
27	UNIT123	RW	0x0	Flash 单元 123 写保护 （0x1007_B000 → 0x1007_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
26	UNIT122	RW	0x0	Flash 单元 122 写保护 （0x1007_A000 → 0x1007_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
25	UNIT121	RW	0x0	Flash 单元 121 写保护 （0x1007_9000 → 0x1007_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
24	UNIT120	RW	0x0	Flash 单元 120 写保护 （0x1007_8000 → 0x1007_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
23	UNIT119	RW	0x0	Flash 单元 119 写保护 （0x1007_7000 → 0x1001_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
22	UNIT118	RW	0x0	Flash 单元 118 写保护 （0x1007_6000 → 0x1007_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
21	UNIT117	RW	0x0	Flash 单元 117 写保护 （0x1007_5000 → 0x1007_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
20	UNIT116	RW	0x0	Flash 单元 116 写保护 （0x1007_4000 → 0x1007_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
19	UNIT115	RW	0x0	Flash 单元 115 写保护 （0x1007_3000 → 0x1007_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
18	UNIT114	RW	0x0	Flash 单元 114 写保护 （0x1007_2000 → 0x1007_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
17	UNIT113	RW	0x0	Flash 单元 113 写保护 （0x1007_1000 → 0x1007_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
16	UNIT112	RW	0x0	Flash 单元 112 写保护 （0x1007_0000 → 0x1007_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
15	UNIT111	RW	0x0	Flash 单元 111 写保护 （0x1006_F000 → 0x1006_FFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
14	UNIT110	RW	0x0	Flash 单元 110 写保护 （0x1006_E000 → 0x1006_EFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
13	UNIT109	RW	0x0	Flash 单元 109 写保护 （0x1006_D000 → 0x1006_DFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
12	UNIT108	RW	0x0	Flash 单元 108 写保护 （0x1006_C000 → 0x1006_CFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
11	UNIT107	RW	0x0	Flash 单元 107 写保护 （0x1006_B000 → 0x1006_BFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
10	UNIT106	RW	0x0	Flash 单元 106 写保护 （0x1006_A000 → 0x1006_AFFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
9	UNIT105	RW	0x0	Flash 单元 105 写保护 （0x1006_9000 → 0x1006_9FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
8	UNIT104	RW	0x0	Flash 单元 104 写保护 （0x1006_8000 → 0x1006_8FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
7	UNIT103	RW	0x0	Flash 单元 103 写保护 （0x1006_7000 → 0x1006_7FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
6	UNIT102	RW	0x0	Flash 单元 102 写保护 （0x1006_6000 → 0x1006_6FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
5	UNIT101	RW	0x0	Flash 单元 101 写保护 （0x1006_5000 → 0x1006_5FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

位段	位段名称	属性	复位值	描述
4	UNIT100	RW	0x0	Flash 单元 100 写保护 （0x1006_4000 → 0x1006_4FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
3	UNIT99	RW	0x0	Flash 单元 99 写保护 （0x1006_3000 → 0x1006_3FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
2	UNIT98	RW	0x0	Flash 单元 98 写保护 （0x1006_2000 → 0x1006_2FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
1	UNIT97	RW	0x0	Flash 单元 97 写保护 （0x1006_1000 → 0x1006_1FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除
0	UNIT96	RW	0x0	Flash 单元 96 写保护 （0x1006_0000 → 0x1006_0FFF） 0：可以被编程或者擦除 1：不可以被编程或者扇区擦除

表 20-22：Flash 控制器写使能寄存器（FLASHREGKEY）位段定义

FLASHREGKEY (Flash Register Write-Allow Key Register) Offset: 0x3C Default: 0x1ACCE551 Access: FLASH -> FLASHREGKEY.all							
31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

表 20-23：Flash 控制器写使能寄存器（FLASHREGKEY）位段描述

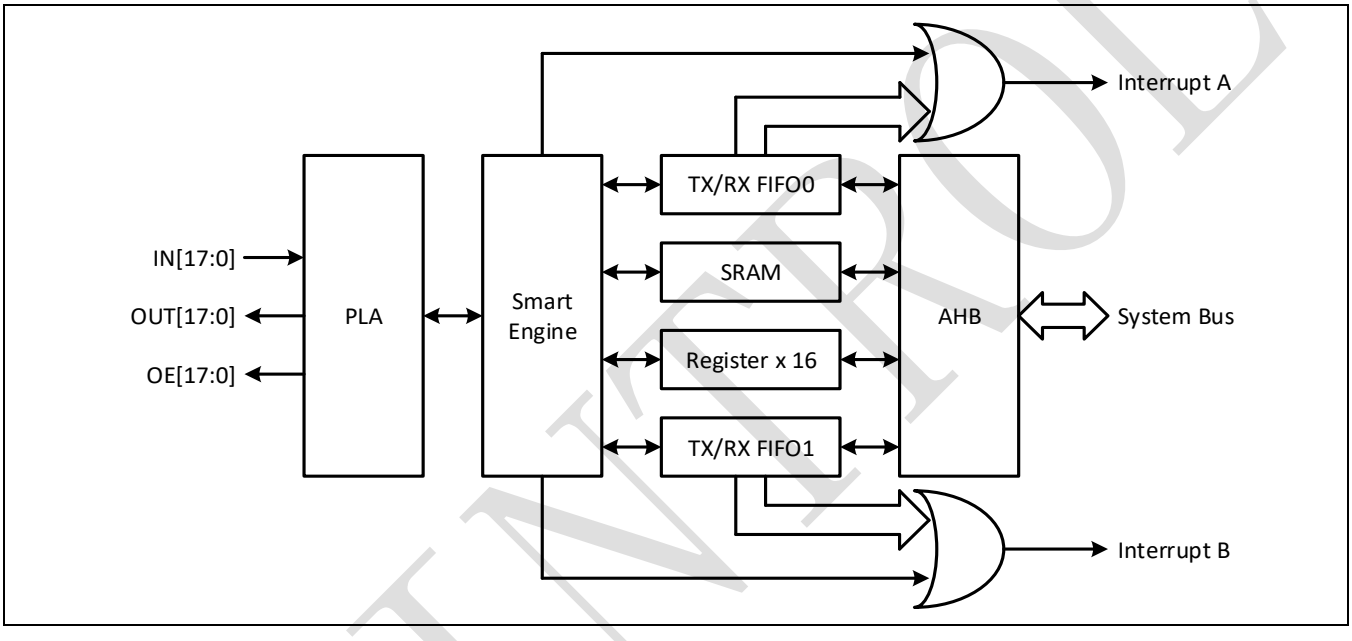
位段	位段名称	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 Flash 寄存器

21 SIO

21.1 概述

SIO 是 Spintrol 的一项专利技术。它具有可变编程的特性，通过编程可以将 SIO 模块转换为预定义的通信模块。SPC2168 集成了 3 个 SIO 模块，每个 SIO 模块支持多达 18 个 I/O 通道，如表 4-1 所示。

图 21-1：SIO 架构原理图



SIO 模块的架构如图 21-1 所示。它的基本组成包括两个 TX FIFO、两个 RX FIFO、16 个寄存器、一个接在 AHB 总线上的 SRAM、一个可编程逻辑阵列（PLA）和一个智能引擎（Smart Engine）。每个 SIO 模块可以向 CPU 内核提供两个中断信号。可编程逻辑阵列可以提供快速并行的逻辑处理性能，而智能引擎可以胜任复杂协议的状态控制以及传输工作。通过利用这两个模块的优点，SPC2168 中的 SIO 模块可以实现下面的使用案例：

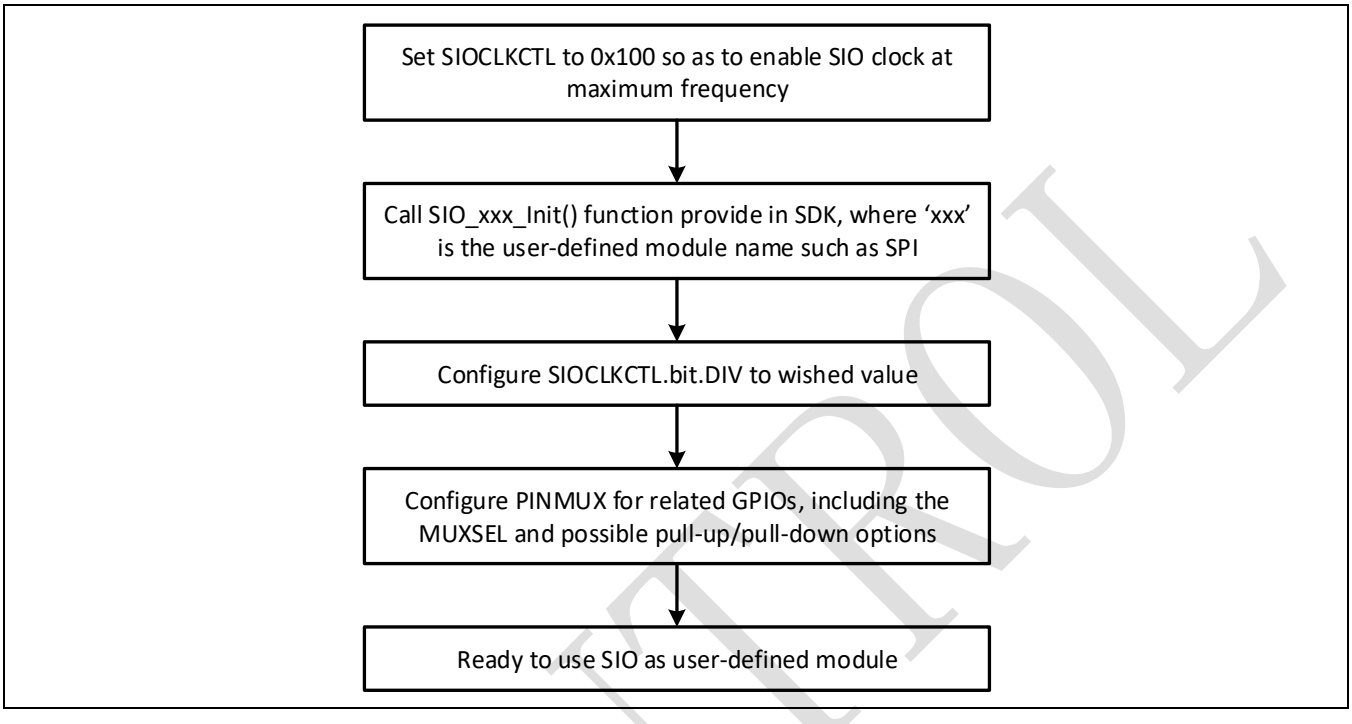
- 重新定义在 SPC2168 上某个接口的管脚
  例如，在 SIO 模块上实现的一个 UART 接口，使用 GPIO40/GPIO41 作为 TXD/RXD，通过更新 SIO 的固件，将可以将 UART 接口切换为使用 GPIO30/GPIO31 作为 TXD/RXD。
- 重新定义在 SPC2168 上某个接口的整个功能
  例如，一个 SIO 模块被初始定义为 I2C 功能，但是通过更新该 SIO 的固件，就可以将该 SIO 模块重新定义为 UART 功能。
- 为 SPC2168 上的标准接口增加功能

  例如，用户可以在标准 UART 的基础上定义新的数据格式和通信协议，这个新接口的功能可以通过更新 SIO 的固件来实现。

21.2 SIO 初始化

图 21-2 所示流程介绍了将 SIO 模块初始化为一个用户自定义模块的过程。

图 21-2：SIO 初始化流程



Example 21.2.1

```
void   SIO_Example21_2_1(void)
{
  /* Enable SIO clock at maximum frequency to minimize initialization time */
  CLOCK->SIOCLKCTL.all = 0x100;

  /* Initialize the SIO PLA and smart engine to make the SIO as a UART */
  SIO_UART_Init();

  /* Configure SIO clock frequency as SYSCLK1/4 */
  CLOCK->SIOCLKCTL.bit.DIV = 0x3;

  /* Configure GPIO36 and GPIO37 to SIO channel with pull-up enabled for UART */
  PINMUX->GPIO36.bit.MUXSEL   = GPIO36_BIT_MUXSEL_SIO0_14;
  PINMUX->GPIO36.bit.PU       |= 1 << GPIO36_BIT_MUXSEL_SIO0_14;
  PINMUX->GPIO37.bit.MUXSEL   = GPIO37_BIT_MUXSEL_SIO0_15;
  PINMUX->GPIO37.bit.PU       |= 1 << GPIO37_BIT_MUXSEL_SIO0_15;

  /* Ready to use SIO as UART */
}
```

21.3 寄存器

请参考相应的基于 SIO 的用户自定义模块的应用手册。

22 系统控制模块

22.1 概述

系统控制模块提供：

- 唯一设备 ID 寄存器（64 位）
- 存储器错误中断控制寄存器
- 存储器 ECC 使能寄存器
- 存储器锁定状态寄存器
- 复位事件控制寄存器
- 系统信息寄存器

22.2 唯一设备 ID 寄存器（64 位）

唯一设备 ID 非常适合于：

- 用作序列号
- 用作密钥以提高 Flash 存储器中代码的安全性。例如，在对 Flash 存储器编程之前，用户可以将其与软件加密原语和协议相结合来使用。
- 激活安全启动过程等

64 位唯一设备 ID 为每一颗 SPC2168 分配了独有的标识号，且无法由用户更改。

22.3 存储器错误中断控制寄存器

这些寄存器用来使能、标识以及清除存储器错误中断。

22.4 存储器 ECC 使能寄存器

该寄存器使能 ROM 和 Flash 存储器的 ECC 特性。

22.5 存储器锁定状态寄存器

该寄存器记录了 Flash 和 RAM 存储器分区保护状态以及配置字（Configuration Words）锁定状态。

22.6 复位事件控制寄存器

这些寄存器用来使能、标识以及清除复位事件。

22.7 系统信息寄存器

该寄存器提供了 Flash 存储器大小、RAM 大小、ADC 有效位宽以及时钟频率选项信息。

22.8 CAU 控制寄存器

该寄存器用来使能 CAU 模块的运行以及 CAU 的 Debug 接口。

22.9 寄存器

22.9.1 系统寄存器列表

表 22-1：SYSTEM 模块基地址

外设模块	基地址
SYSTEM	0x4000 0000

表 22-2：SYSTEM 寄存器列表

寄存器	偏移地址	描述	复位值
CID0	0x00	芯片 ID 寄存器 0	0x00000000
CID1	0x04	芯片 ID 寄存器 1	0x00000000
UID0	0x08	唯一设备 ID 寄存器 0	0x00000000
UID1	0x0C	唯一设备 ID 寄存器 1	0x00000000
RND0	0x10	随机数寄存器 0	0x00000000
RND1	0x14	随机数寄存器 1	0x00000000
REV0	0x18	版本信息寄存器 0	0x636D1611
REV1	0x1C	版本信息寄存器 1	0x201812A0
MEMIF	0x20	存储器错误中断标志寄存器	0x00000000
MEMIC	0x24	存储器错误中断清除寄存器	0x00000000
MEMIE*	0x28	存储器错误中断使能寄存器	0x00000000
MEMECCEN*	0x2C	存储器 ECC 使能寄存器	0x00000007
MEMLOCKSTS	0x30	存储器锁定状态寄存器	0x00000000
RSTEVTSTS	0x34	复位事件状态寄存器	0x00000000
RSTEVTCLR	0x38	复位事件清除寄存器	0x00000000
RSTEVTEN*	0x3C	复位事件使能寄存器	0x00000005
SYSINFO	0x40	系统信息寄存器	0x000003F3
CAUCTL*	0x44	CAU 控制寄存器	0x00000000
SYSREGKEY	0x48	SYSTEM 模块写使能寄存器	0x1ACCE551

注意： 由*标识的寄存器仅当 SYSREGKEY=0x1ACCE551 时才可以改写。

22.9.2 系统寄存器

表 22-3 到表 22-40 给出了 SYSTEM 模块各寄存器的定义。

表 22-3：芯片 ID 寄存器 0（CID0）位段定义

CID0 (Chip ID Register 0) Offset: 0x0 Default: 0x00000000 Access: GLOBAL -> CID0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-4：芯片 ID 寄存器 0（CID0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位芯片 ID 的低 32 位

表 22-5：芯片 ID 寄存器 1（CID1）位段定义

CID1 (Chip ID Register 1) Offset: 0x4 Default: 0x00000000 Access: GLOBAL -> CID1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-6：芯片 ID 寄存器 1（CID1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位芯片 ID 的高 32 位

表 22-7：唯一设备 ID 寄存器 0（UID0）位段定义

UID0 (Unique ID Register 0) Offset: 0x8 Default: 0x00000000 Access: GLOBAL -> UID0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-8：唯一设备 ID 寄存器 0（UID0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位唯一设备 ID 的低 32 位

表 22-9：唯一设备 ID 寄存器 1（UID1）位段定义

UID1 (Unique ID Register 1) Offset: 0xC Default: 0x00000000 Access: GLOBAL -> UID1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-10：唯一设备 ID 寄存器 1（UID1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位唯一设备 ID 的高 32 位

表 22-11：随机数寄存器 0（RND0）位段定义

RND0 (Random Number Register 0) Offset: 0x10 Default: 0x00000000 Access: SYSTEM -> RND0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-12：随机数寄存器 0（RND0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位随机数的低 32 位

表 22-13：随机数寄存器 1（RND1）位段定义

RND1 (Random Number Register 1) Offset: 0x14 Default: 0x00000000 Access: SYSTEM -> RND1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-14：随机数寄存器 1（RND1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x0	64 位随机数的高 32 位

表 22-15：版本信息寄存器 0（REV0）位段定义

REV0 (Revision Information Register 0) Offset: 0x18 Default: 0x636D1611 Access: SYSTEM -> REV0.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-16：版本信息寄存器 0（REV0）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x636D1611	64 位版本号的低 32 位

表 22-17：版本信息寄存器 1（REV1）位段定义

REV1 (Revision Information Register 1) Offset: 0x1C Default: 0x201812A0 Access: SYSTEM -> REV1.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 22-18：版本信息寄存器 1（REV1）位段描述

位段	位段名	属性	复位值	描述
31:0	VAL	RO	0x201812A0	64 位版本号的高 32 位

表 22-19：存储器错误中断标志寄存器（MEMIF）位段定义

MEMIF (Memory Error Interrupt Flag Register) Offset: 0x20 Default: 0x00000000 Access: SYSTEM -> MEMIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 MEMINT	14 SIO2PERR	13 SIO1PERR	12 SIO0PERR	11 CDRAM2ERR	10 CDRAM1ERR	9 CIRAM2ERR	8 CIRAM1ERR
7 MDRAM2ERR	6 MDRAM1ERR	5 MIRAM2ERR	4 MIRAM1ERR	3 FLASH2ERR	2 FLASH1ERR	1 ROM2ERR	0 ROM1ERR

表 22-20：存储器错误中断标志寄存器（MEMIF）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	MEMINT	RO	0x0	全局存储器错误中断标志 0：存储器错误中断未发生 1：存储器错误中断已发生 本位段清零前，新的存储器错误事件不会触发新的中断给 CPU。
14	SIO2PERR	RO	0x0	SIO2 存储奇偶校验错误标志 0：校验错误未发生 1：校验错误已发生 本位段清零前，新的校验错误事件不会触发新的中断。
13	SIO1PERR	RO	0x0	SIO1 存储奇偶校验错误标志 0：校验错误未发生 1：校验错误已发生 本位段清零前，新的校验错误事件不会触发新的中断。
12	SIO0PERR	RO	0x0	SIO0 存储奇偶校验错误标志 0：校验错误未发生 1：校验错误已发生 本位段清零前，新的校验错误事件不会触发新的中断。
11	CDRAM2ERR	RO	0x0	CAU DRAM ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误 本位段清零前，新的 2 比特错误事件不会触发新的中断。

位段	位段名	属性	复位值	描述
10	CDRAM1ERR	RO	0x0	CAU DRAM ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。
9	CIRAM2ERR	RO	0x0	CAU IRAM ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误 本位段清零前，新的 2 比特错误事件不会触发新的中断。
8	CIRAM1ERR	RO	0x0	CAU IRAM ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。
7	MDRAM2ERR	RO	0x0	Main DRAM ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误 本位段清零前，新的 2 比特错误事件不会触发新的中断。
6	MDRAM1ERR	RO	0x0	Main DRAM ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。
5	MIRAM2ERR	RO	0x0	Main IRAM ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误 本位段清零前，新的 2 比特错误事件不会触发新的中断。
4	MIRAM1ERR	RO	0x0	Main IRAM ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。
3	FLASH2ERR	RO	0x0	FLASH 存储器 ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误

位段	位段名	属性	复位值	描述
				本位段清零前，新的 2 比特错误事件不会触发新的中断。
2	FLASH1ERR	RO	0x0	FLASH 存储器 ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。
1	ROM2ERR	RO	0x0	ROM 存储器 ECC 检测到 2 比特错误标志 0：未检测到 2 比特错误 1：检测到 2 比特错误 本位段清零前，新的 2 比特错误事件不会触发新的中断。
0	ROM1ERR	RO	0x0	ROM 存储器 ECC 检测到 1 比特错误标志 0：未检测到 1 比特错误 1：检测到 1 比特错误 本位段清零前，新的 1 比特错误事件不会触发新的中断。

表 22-21：存储器错误中断清除寄存器（MEMIC）位段定义

MEMIC (Memory Error Interrupt Clear Register) Offset: 0x24 Default: 0x00000000 Access: SYSTEM -> MEMIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 MEMINT	14 SIO2PERR	13 SIO1PERR	12 SIO0PERR	11 CDRAM2ERR	10 CDRAM1ERR	9 CIRAM2ERR	8 CIRAM1ERR
7 MDRAM2ERR	6 MDRAM1ERR	5 MIRAM2ERR	4 MIRAM1ERR	3 FLASH2ERR	2 FLASH1ERR	1 ROM2ERR	0 ROM1ERR

表 22-22：存储器错误中断清除寄存器（MEMIC）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	MEMINT	W1C	0x0	全局存储器错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除中断和标志位，本位段自动清零
14	SIO2PERR	W1C	0x0	SIO2 存储奇偶校验错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
13	SIO1PERR	W1C	0x0	SIO1 存储奇偶校验错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
12	SIO0PERR	W1C	0x0	SIO0 存储奇偶校验错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
11	CDRAM2ERR	W1C	0x0	CAU DRAM ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
10	CDRAM1ERR	W1C	0x0	CAU DRAM ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
9	CIRAM2ERR	W1C	0x0	CAU IRAM ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
8	CIRAM1ERR	W1C	0x0	CAU DRAM ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
7	MDRAM2ERR	W1C	0x0	Main DRAM ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0

位段	位段名	属性	复位值	描述
				1：写 1 清除标志位，本位段自动清零
6	MDRAM1ERR	W1C	0x0	Main DRAM ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
5	MIRAM2ERR	W1C	0x0	Main IRAM ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
4	MIRAM1ERR	W1C	0x0	Main IRAM ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
3	FLASH2ERR	W1C	0x0	FLASH 存储 ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
2	FLASH1ERR	W1C	0x0	FLASH 存储 ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
1	ROM2ERR	W1C	0x0	ROM 存储 ECC 检测 2 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零
0	ROM1ERR	W1C	0x0	ROM 存储 ECC 检测 1 比特错误清除 0：写 0 无效，总是读回 0 1：写 1 清除标志位，本位段自动清零

表 22-23：存储器错误中断使能寄存器（MEMIE）位段定义

MEMIE (Memory Error Interrupt Enable Register) Offset: 0x28 Default: 0x00000000 Access: SYSTEM -> MEMIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED	SIO2PERR	SIO1PERR	SIO0PERR	CDRAM2ERR	CDRAM1ERR	CIRAM2ERR	CIRAM1ERR
7	6	5	4	3	2	1	0
MDRAM2ERR	MDRAM1ERR	MIRAM2ERR	MIRAM1ERR	FLASH2ERR	FLASH1ERR	ROM2ERR	ROM1ERR

表 22-24：存储器错误中断使能寄存器（MEMIE）位段描述

位段	位段名	属性	复位值	描述
31:15	RESERVED_31_15	RO	0x0	保留
14	SIO2PERR	RW	0x0	SIO2 存储奇偶校验错误中断使能 0：关闭中断 1：使能中断
13	SIO1PERR	RW	0x0	SIO1 存储奇偶校验错误中断使能 0：关闭中断 1：使能中断
12	SIO0PERR	RW	0x0	SIO0 存储奇偶校验错误中断使能 0：关闭中断 1：使能中断
11	CDRAM2ERR	RW	0x0	CAU DRAM ECC 2 比特错误中断使能 0：关闭中断 1：使能中断
10	CDRAM1ERR	RW	0x0	CAU DRAM ECC 1 比特错误中断使能 0：关闭中断 1：使能中断
9	CIRAM2ERR	RW	0x0	CAU IRAM ECC 2 比特错误中断使能 0：关闭中断 1：使能中断
8	CIRAM1ERR	RW	0x0	CAU IRAM ECC 1 比特错误中断使能 0：关闭中断 1：使能中断
7	MDRAM2ERR	RW	0x0	Main DRAM ECC 2 比特错误中断使能 0：关闭中断 1：使能中断

位段	位段名	属性	复位值	描述
6	MDRAM1ERR	RW	0x0	Main DRAM ECC 1 比特错误中断使能 0：关闭中断 1：使能中断
5	MIRAM2ERR	RW	0x0	Main IRAM ECC 2 比特错误中断使能 0：关闭中断 1：使能中断
4	MIRAM1ERR	RW	0x0	Main IRAM ECC 1 比特错误中断使能 0：关闭中断 1：使能中断
3	FLASH2ERR	RW	0x0	Flash 存储 ECC 2 比特错误中断使能 0：关闭中断 1：使能中断
2	FLASH1ERR	RW	0x0	Flash 存储 ECC 1 比特错误中断使能 0：关闭中断 1：使能中断
1	ROM2ERR	RW	0x0	ROM 存储 ECC 2 比特错误中断使能 0：关闭中断 1：使能中断
0	ROM1ERR	RW	0x0	ROM 存储 ECC 1 比特错误中断使能 0：关闭中断 1：使能中断

表 22-25：存储器 ECC 使能寄存器（MEMECCEN）位段定义

MEMECCEN (Memory ECC Enable Register) Offset: 0x2C Default: 0x00000003 Access: SYSTEM -> MEMECCEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED						FLASHECC	ROMECC

表 22-26：存储器 ECC 使能寄存器（MEMECCEN）位段描述

位段	位段名	属性	复位值	描述
31:2	RESERVED_31_2	RO	0x0	保留
1	FLASHECC	RW	0x1	FLASH ECC 使能 0：关闭 FLASH ECC 1：使能 FLASH ECC
0	ROMECC	RW	0x1	ROM ECC 使能 0：关闭 ROM ECC 1：使能 ROM ECC

表 22-27：存储器锁定状态寄存器（MEMLOCKSTS）位段定义

MEMLOCKSTS (Memory Lock Register) Offset: 0x30 Default: 0x00000000 Access: GLOBAL -> MEMLOCKSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED			RAM3	RAM2	RAM1	RAM0	FLASHOTP
7	6	5	4	3	2	1	0
USER3INFO	USER2INFO	USER1INFO	USER0INFO	FLASH3	FLASH2	FLASH1	FLASH0

表 22-28：存储器锁定状态寄存器（MEMLOCKSTS）位段描述

位段	位段名	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12	RAM3	RO	0x0	RAM 区域 3 锁定状态 0：可以读写数据和取指执行 1：不能读写数据，只能取指执行

位段	位段名	属性	复位值	描述
11	RAM2	RO	0x0	RAM 区域 2 锁定状态 0：可以读写数据和取指执行 1：不能读写数据，只能取指执行
10	RAM1	RO	0x0	RAM 区域 1 锁定状态 0：可以读写数据和取指执行 1：不能读写数据，只能取指执行
9	RAM0	RO	0x0	RAM 区域 0 锁定状态 0：可以读写数据和取指执行 1：不能读写数据，只能取指执行
8	FLASHOTP	RO	0x0	FLASH OTP 锁定状态 0：FLASH OTP 区域可以读取、擦除和重新写入 1：FLASH OTP 区域只读
7	USER3INFO	RO	0x0	多区域安全机制用户 3 信息锁定状态 0：允许读取、擦除和重写用户 3 信息 1：用户 3 信息只读
6	USER2INFO	RO	0x0	多区域安全机制用户 2 信息锁定状态 0：允许读取、擦除和重写用户 2 信息 1：用户 2 信息只读
5	USER1INFO	RO	0x0	多区域安全机制用户 1 信息锁定状态 0：允许读取、擦除和重写用户 1 信息 1：用户 1 信息只读
4	USER0INFO	RO	0x0	多区域安全机制用户 0 信息锁定状态 0：允许读取、擦除和重写用户 0 信息 1：用户 0 信息只读
3	FLASH3	RO	0x0	RAM 区域 3 锁定状态 0：不能读写数据，只能取指执行 1：可以读写数据和取指执行
2	FLASH2	RO	0x0	RAM 区域 2 锁定状态 0：不能读写数据，只能取指执行 1：可以读写数据和取指执行
1	FLASH1	RO	0x0	RAM 区域 1 锁定状态 0：不能读写数据，只能取指执行 1：可以读写数据和取指执行
0	FLASH0	RO	0x0	RAM 区域 0 锁定状态 0：不能读写数据，只能取指执行 1：可以读写数据和取指执行

表 22-29：复位事件状态寄存器（RSTEVTSTS）位段定义

RSTEVTSTS (Reset Event Status Register) Offset: 0x34 Default: 0x00000000 Access: SYSTEM -> RSTEVTSTS.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED					SYSRESET	WDT1RST	WDT0RST
15	14	13	12	11	10	9	8
CLKDETERR	PLLUNLOCK	VDD33H	VDD33L	VDD12H	VDD12L1	VDD12L0	SIO2ERR
7	6	5	4	3	2	1	0
SIO1ERR	SIO0ERR	CDRAMERR	CIRAMERR	MDRAMERR	MIRAMERR	FLASHERR	ROMERR

表 22-30：复位事件状态寄存器（RSTEVTSTS）位段描述

位段	位段名	属性	复位值	描述
31:19	RESERVED_31_19	RO	0x0	保留
18	SYSRESET	RO	0x0	CPU SYSRESETREQ 事件触发的复位状态 0：复位事件未发生 1：复位事件发生
17	WDT1RST	RO	0x0	WDT1 复位事件触发的复位状态 0：复位事件未发生 1：复位事件发生
16	WDT0RST	RO	0x0	WDT0 复位事件触发的复位状态 0：复位事件未发生 1：复位事件发生
15	CLKDETERR	RO	0x0	时钟检测模块检测到错误事件触发的复位状态 0：复位事件未发生 1：复位事件发生
14	PLLUNLOCK	RO	0x0	PLL 失锁错误事件触发的复位状态 0：复位事件未发生 1：复位事件发生
13	VDD33H	RO	0x0	VDD33H 过压事件触发的复位状态 0：复位事件未发生 1：复位事件发生
12	VDD33L	RO	0x0	VDD33L 欠压事件触发的复位状态 0：复位事件未发生 1：复位事件发生
11	VDD12H	RO	0x0	VDD12H 过压事件触发的复位状态 0：复位事件未发生 1：复位事件发生

位段	位段名	属性	复位值	描述
10	VDD12L1	RO	0x0	VDD12L 欠压事件 1 触发的复位状态 0：复位事件未发生 1：复位事件发生
9	VDD12L0	RO	0x0	VDD12L 欠压事件 0 触发的复位状态 0：复位事件未发生 1：复位事件发生
8	SIO2ERR	RO	0x0	SIO2 存储奇偶校验错误触发的复位状态 0：复位事件未发生 1：复位事件发生
7	SIO1ERR	RO	0x0	SIO1 存储奇偶校验错误触发的复位状态 0：复位事件未发生 1：复位事件发生
6	SIO0ERR	RO	0x0	SIO0 存储奇偶校验错误触发的复位状态 0：复位事件未发生 1：复位事件发生
5	CDRAMERR	RO	0x0	CAU DRAM ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生
4	CIRAMERR	RO	0x0	CAU IRAM ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生
3	MDRAMERR	RO	0x0	Main DRAM ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生
2	MIRAMERR	RO	0x0	Main IRAM ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生
1	FLASHERR	RO	0x0	FLASH 存储 ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生
0	ROMERR	RO	0x0	ROM 存储 ECC 2 比特错误触发的复位状态 0：复位事件未发生 1：复位事件发生

表 22-31：复位事件状态清除寄存器（RSTEVTCLR）位段定义

RSTEVTCLR (Reset Event Status Clear Register) Offset: 0x38 Default: 0x00000000 Access: SYSTEM -> RSTEVTCLR.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED					SYSRESET	WDT1RST	WDT0RST
15	14	13	12	11	10	9	8
CLKDETERR	PLLUNLOCK	VDD33H	VDD33L	VDD12H	VDD12L1	VDD12L0	SIO2ERR
7	6	5	4	3	2	1	0
SIO1ERR	SIO0ERR	CDRAMERR	CIRAMERR	MDRAMERR	MIRAMERR	FLASHERR	ROMERR

表 22-32：复位事件状态清除寄存器（RSTEVTCLR）位段描述

位段	位段名	属性	复位值	描述
31:19	RESERVED_31_19	RO	0x0	保留
18	SYSRESET	W1C	0x0	CPU SYSRESETREQ 事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
17	WDT1RST	W1C	0x0	WDT1 复位事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
16	WDT0RST	W1C	0x0	WDT0 复位事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
15	CLKDETERR	W1C	0x0	时钟检测模块检测到错误事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
14	PLLUNLOCK	W1C	0x0	PLL 失锁错误事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
13	VDD33H	W1C	0x0	VDD33H 过压事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
12	VDD33L	W1C	0x0	VDD33L 欠压事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
11	VDD12H	W1C	0x0	VDD12H 过压事件触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零

位段	位段名	属性	复位值	描述
10	VDD12L1	W1C	0x0	VDD12L 欠压事件 1 触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
9	VDD12L0	W1C	0x0	VDD12L 欠压事件 0 触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
8	SIO2ERR	W1C	0x0	SIO2 存储奇偶校验错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
7	SIO1ERR	W1C	0x0	SIO1 存储奇偶校验错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
6	SIO0ERR	W1C	0x0	SIO0 存储奇偶校验错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
5	CDRAMERR	W1C	0x0	CAU DRAM ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
4	CIRAMERR	W1C	0x0	CAU IRAM ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
3	MDRAMERR	W1C	0x0	Main DRAM ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
2	MIRAMERR	W1C	0x0	Main IRAM ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
1	FLASHERR	W1C	0x0	FLASH 存储 ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零
0	ROMERR	W1C	0x0	ROM 存储 ECC 2 比特错误触发的复位状态清除 0：写 0 无效，总是读回 0 1：写 1 清除锁存的复位状态，本位段自动清零

表 22-33：复位事件使能寄存器（RSTEVTEN）位段定义

RSTEVTEN (Reset Event Enable Register) Offset: 0x3C Default: 0x00000005 Access: SYSTEM -> RSTEVTEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15 CLKDETERR	14 PLLUNLOCK	13 VDD33H	12 VDD33L	11 VDD12H	10 VDD12L1	9 VDD12L0	8 SIO2ERR
7 SIO1ERR	6 SIO0ERR	5 CDRAMERR	4 CIRAMERR	3 MDRAMERR	2 MIRAMERR	1 FLASHERR	0 ROMERR

表 22-34：复位事件使能寄存器（RSTEVTEN）位段描述

位段	位段名	属性	复位值	描述
31:16	RESERVED_31_16	RO	0x0	保留
15	CLKDETERR	RW	0x0	CLKDET 错误事件触发复位使能 0：不触发复位 1：触发复位
14	PLLUNLOCK	RW	0x0	PLL 失锁事件触发复位使能 0：不触发复位 1：触发复位
13	VDD33H	RW	0x0	VDD33H 过压事件触发复位使能 0：不触发复位 1：触发复位
12	VDD33L	RW	0x0	VDD33L 欠压事件触发复位使能 0：不触发复位 1：触发复位
11	VDD12H	RW	0x0	VDD12H 过压事件触发复位使能 0：不触发复位 1：触发复位
10	VDD12L1	RW	0x0	VDD12L1 欠压事件触发复位使能 0：不触发复位 1：触发复位
9	VDD12L0	RW	0x0	VDD12L0 欠压事件触发复位使能 0：不触发复位 1：触发复位
8	SIO2ERR	RW	0x0	SIO2 存储奇偶校验错误触发复位使能 0：不触发复位 1：触发复位

位段	位段名	属性	复位值	描述
7	SIO1ERR	RW	0x0	SIO1 存储奇偶校验错误触发复位使能 0：不触发复位 1：触发复位
6	SIO0ERR	RW	0x0	SIO0 存储奇偶校验错误触发复位使能 0：不触发复位 1：触发复位
5	CDRAMERR	RW	0x0	CAU DRAM ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位
4	CIRAMERR	RW	0x0	CAU IRAM ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位
3	MDRAMERR	RW	0x1	Main DRAM ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位
2	MIRAMERR	RW	0x1	Main IRAM ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位
1	FLASHERR	RW	0x1	FLASH 存储 ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位
0	ROMERR	RW	0x1	ROM 存储 ECC 2 比特错误触发复位使能 0：不触发复位 1：触发复位

表 22-35：系统信息寄存器（SYSINFO）位段定义

SYSINFO (System Information Register)　Offset: 0x40　Default: 0x000001F3
Access: SYSTEM -> SYSINFO.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED						WITHCAU	ADCBIT
7	6	5	4	3	2	1	0
ADCBIT	CLKOPT			RESERVED		FLASHSIZE	

表 22-36：系统信息寄存器（SYSINFO）位段描述

位段	位段名	属性	复位值	描述
31:10	RESERVED_31_10	RO	0x0	保留
9	WITHCAU	RO	0x0	CAU 选项 0：CAU 不可用 1：CAU 可用
8:7	ADCBIT	RO	0x3	ADC 转换结果数据宽度 00：14 比特，其中低 4 比特始终是 0（等效于 10 比特精度） 01：14 比特，其中低 3 比特始终是 0（等效于 11 比特精度） 10：14 比特，其中低 2 比特始终是 0（等效于 12 比特精度） 11：14 比特
6:4	CLKOPT	RO	0x7	时钟选项 000：PLL 输出最小分频比是 12，不支持 XO 001：PLL 输出最小分频比是 6，不支持 XO 010：PLL 输出最小分频比是 4，不支持 XO 011：PLL 输出最小分频比是 12，支持 XO 100：PLL 输出最小分频比是 6，支持 XO 101：PLL 输出最小分频比是 4，支持 XO 110：PLL 输出最小分频比是 3，支持 XO 111：无限制
3:2	RESERVED_3_2	RO	0x0	保留
1:0	FLASHSIZE	RO	0x3	FLASH 存储器大小 00：64KB 01：128KB 10：256KB 11：512KB

表 22-37：CAU 控制寄存器（CAUCTL）位段定义

CAUCTL (CAU Control Register)　Offset: 0x44　Default: 0x00000000
Access: SYSTEM -> CAUCTL.all

31	30	29	28	27	26	25	24
RESERVED						RUN	DBGIFBLK
23	22	21	20	19	18	17	16
DBGIFKEY							
15	14	13	12	11	10	9	8
DBGIFKEY							
7	6	5	4	3	2	1	0
DBGIFKEY							

表 22-38：CAU 控制寄存器（CAUCTL）位段描述

位段	位段名	属性	复位值	描述
31:26	RESERVED_31_26	RO	0x0	保留
25	RUN	RW	0x0	CAU 使能 0：CAU 禁用 1：CAU 使能
24	DBGIFBLK	RW	0x0	阻止 CAU 调试接口 一旦阻止，只能通过上电复位来释放 0：不阻止调试接口 1：阻止调试接口
23:0	DBGIFKEY	RW	0x0	写入 0xACCE51，芯片调试接口仅供 CAU 使用 写入 0xACCE52，芯片调试接口配置为双路 SWD 接口，分别供主 CPU 和 CAU 使用

表 22-39：SYSTEM 模块写使能寄存器（SYSREGKEY）位段定义

SYSREGKEY (System Register Write-Allow Key Register)　Offset: 0x48　Default: 0x1ACCE551
Access: SYSTEM -> SYSREGKEY.all

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

表 22-40：SYSTEM 模块写使能寄存器（SYSREGKEY）位段描述

位段	位段名	属性	复位值	描述
31:0	KEY	RW	0x1ACCE551	写入 0x1ACCE551 解锁受保护的 SYSTEM 寄存器

23 直接存储器访问控制器(DMA)

23.1 DMA 概述

使用直接存储器访问（DMA）的目的是为了在外设和内存、内存和内存之间提供高速数据传输。数据可以被 DMA 快速地移动，而不需要 CPU 的参与。这样可以节省 CPU 的资源，用于其它操作。

SPC2168 的 DMA 控制器共有 6 个通道，用于管理一个或者多个外设对存储器的访问请求。DMA 控制器有一个仲裁器，用以处理不同优先级的 DMA 请求。DMA 控制器通过总线矩阵与 DRAM 互联，通过 AHB-to-APB 桥与 UART 和 SSP 外设互联。

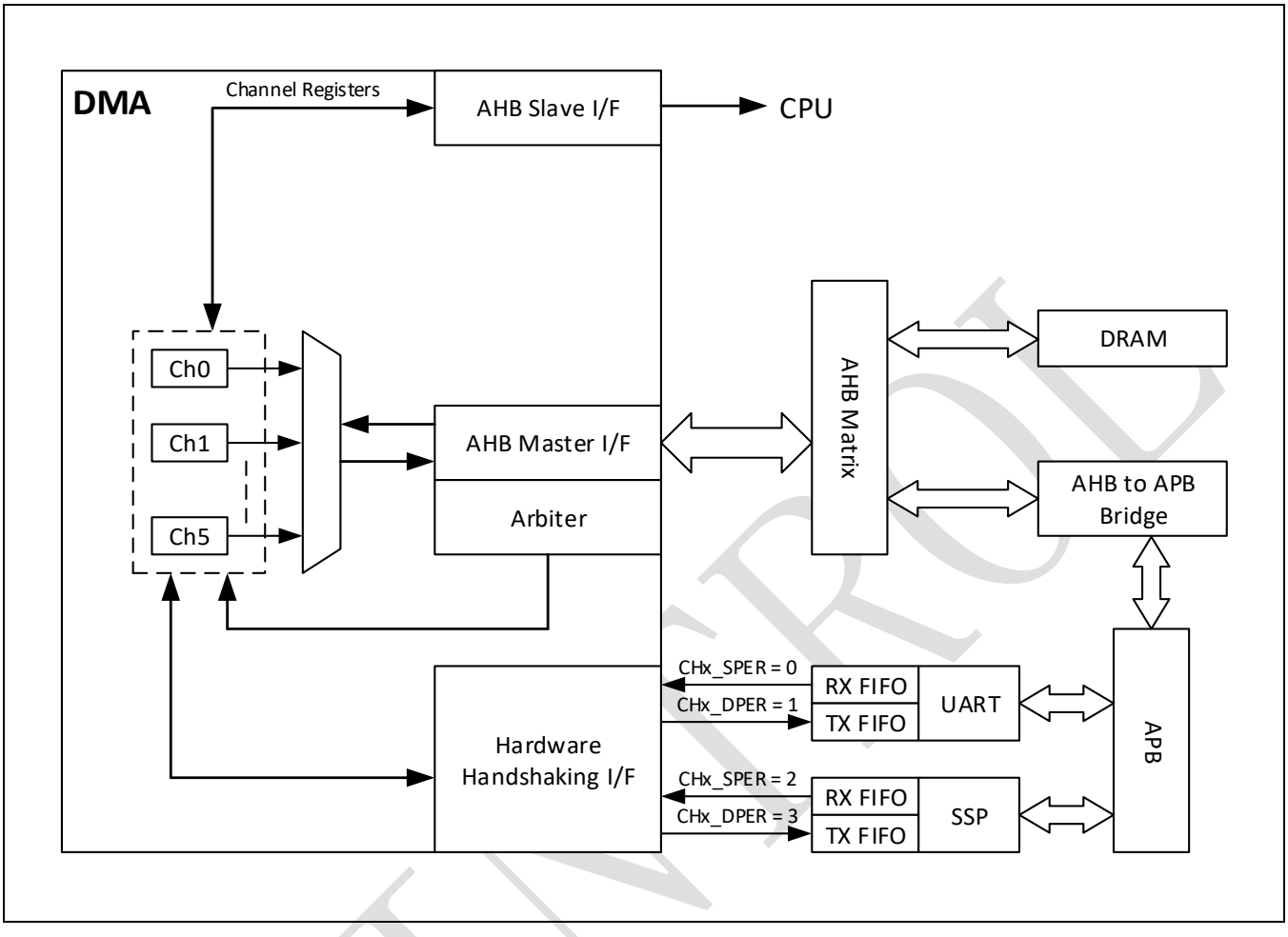
23.2 DMA 特性

SPC2168 DMA 控制器有如下特性：

- 6 个独立、可配置的 DMA 通道，每个通道的 FIFO 深度为 16 x 32 比特
- 非 AHB 外设可以通过硬件握手机制发起一个 DMA 传输请求
- 通道优先级可编程
- 支持 Memory-to-memory 传输
- 支持 Peripheral-to-memory 和 memory-to-peripheral 传输
- 支持数据块传输，数据块大小可配置，最大为 4095 字节
- 源端和目标端的并发传输（burst transaction）的大小可编程
- 源端和目标端的地址递增方式可编程：
  - 地址增加到下一个字对齐的地址
  - 地址降低到下一个字对齐的地址
  - 地址固定不变
- 两个带有标志位的中断源：
  - 传输完成中断
  - 错误中断

DMA 模块的框图如图 23-1 所示。

图 23-1：DMA 模块框图



注意： DMA 可以访问的 DRAM 地址范围为 0x2000 0000 ~ 0x2000 3FFF。

23.3 DMA 操作

23.3.1 基本定义

以下术语是本章节用到的 DMA 概念的定义。

- **源端（Source）** – DMA 读取数据的设备，DMA 从源端设备读取数据，并将数据存储到通道的 FIFO 中。
- **目标端（Destination）** – DMA 将通道 FIFO 中的数据（之前从源端设备读取的）写入的设备。
- **通道（Channel）** – 源端设备和目标端设备之间数据传输的路径。
- **块（Block）** – DMA 数据块。DMA 数据块的大小就是块的大小。对于 DMA 和存储器之间的传输，一个数据块被直接拆分成一序列的并发传输（burst transfer）和单个传输（single transfer）；对于 DMA 和外设之间的传输，一个数据块被拆分成一序列 DMA 事务（tansaction），一个 DMA 事务由并发事务和单个事务构成。
- **事务（Transaction）** – DMA 传输的基本单元。事务只有当与 DMA 进行数据传输的设备是非存储器设备时，才有意义。事务有两种类型：
  - 单个事务（single transaction）：单个事务的数据长度总是 1，并且会被转换成一个 AHB 传输；
  - 并发事务（burst transaction）：并发事务的数据长度由 DMA 的配置决定，并发事务会被转换成一序列的 AHB 并发传输和 AHB 单个传输。

23.3.2 硬件握手接口

DMA 硬件握手接口用在 DMA 事务中，用来控制并发事务和单个事务流。当 DMA 通道和外设都被使能，硬件握手接口能够自动完成数据传输。

在 SPC2168 中，APB 总线外设（UART 和 SSP）通过硬件握手接口和 DMA 连接。UART/SSP 模块和 DMA 通道的连接关系如表 23-1 所示。

表 23-1：DMA 模块的 UART 和 SSP 硬件握手接口

DMACHxCTL3.DPER/ DMACHxCTL3.SPER	硬件握手接口
0	UART_RX
1	UART_TX
2	SSP_RX
3	SSP_TX

通过 AHB 总线与 DMA 直接相连的模块，例如 DRAM 和 SIO 等，和 DMA 之间没有握手接口。一旦 DMA 通道被使能，数据传输会立即进行，不需要等待 DMA 事务请求。

示例：为 UART 选择通道 2 和 3 作为硬件握手接口

```
DMACH3->DMACHCTL3.bit.SPER = 0x0; // Select channel 3 for UART_RX
DMACH2->DMACHCTL3.bit.DPER = 0x1; // Select channel 2 for UART_TX
```

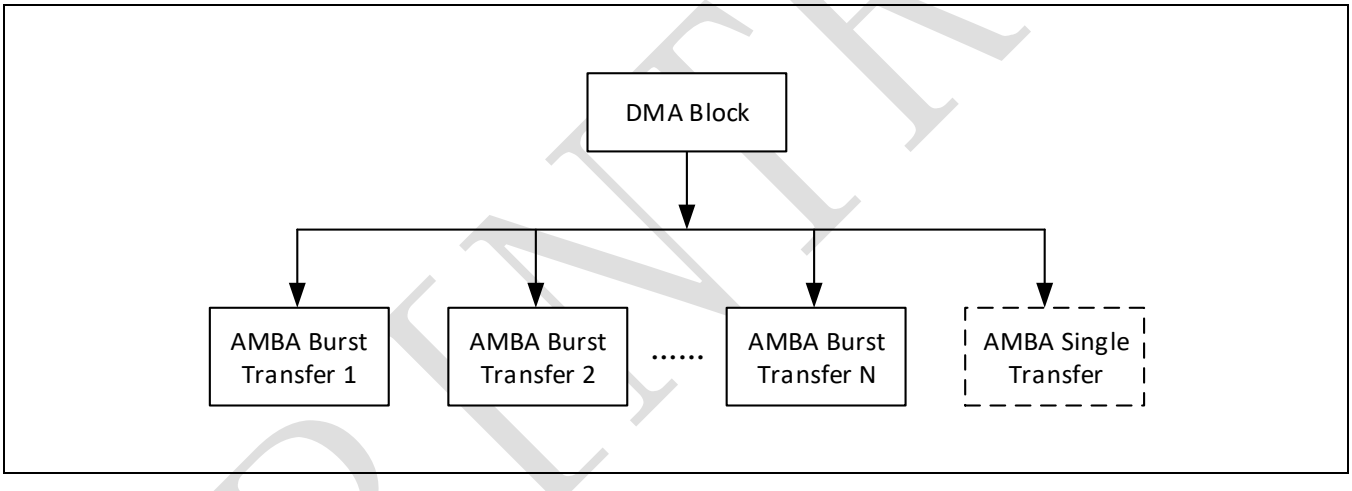
示例：为 SSP 选择通道 4 和 5 作为硬件握手接口

```
DMACH4->DMACHCTL3.bit.DPER = 0x3; // Select channel 4 for SSP_TX
DMACH5->DMACHCTL3.bit.SPER = 0x2; // Select channel 5 for SSP_RX
```

23.3.3 Memory-to-memory 传输操作

DMA 只支持单个数据块的传输，数据块的大小可以通过寄存器位 DMACHCTL1.BLKTS 来配置。对于 DMA 和存储器之间的传输，一个数据块的传输会被直接拆分成一序列的并发传输和单个传输，如图 23-2 所示。

图 23-2：DMA 和存储器之间的传输拆分

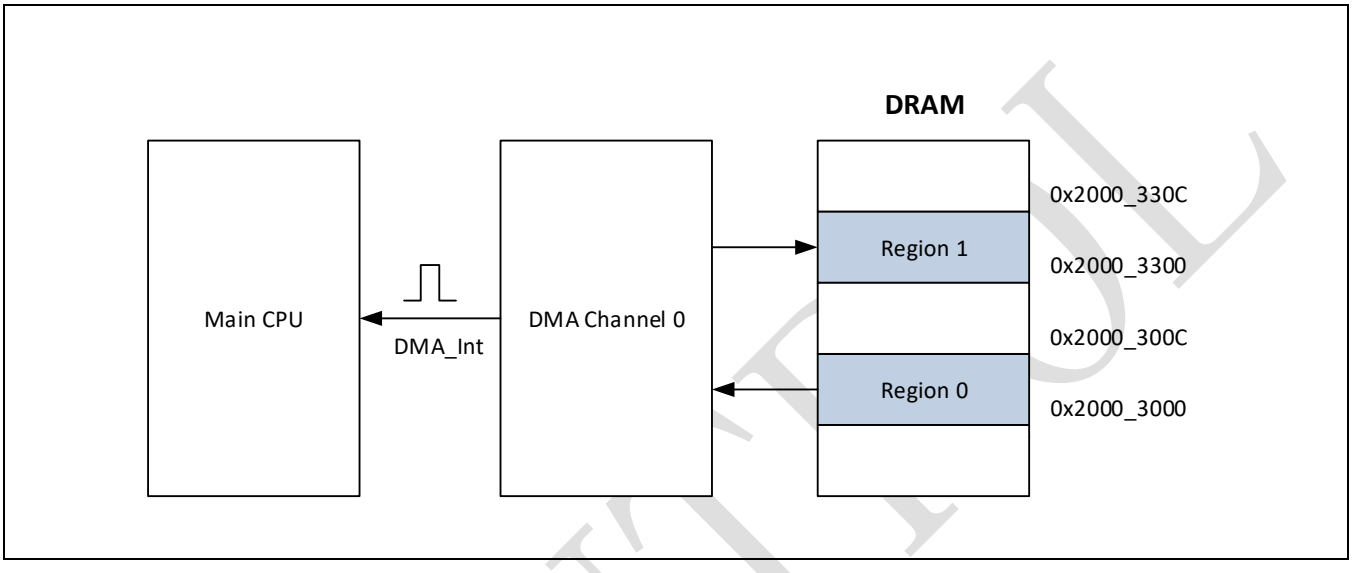


当 DMA 通道中配置的数据块大小不是并发传输长度的整数倍时，如图 23-2 所示，在一些列并发传输后，需要一个单个传输来完成数据块的传输。

示例：DRAM 不同区域之间的 DMA 传输

DMA 通道 0 的配置：传输类型是 memory-to-memory，数据块大小是 0xC，并且源端和目标端的地址步进方式都是递增模式。使能传输完成中断，这样在传输结束时会产生一个中断通知 Main CPU。

图 23-3：两个 DRAM 区域之间的 DMA 传输



示例代码

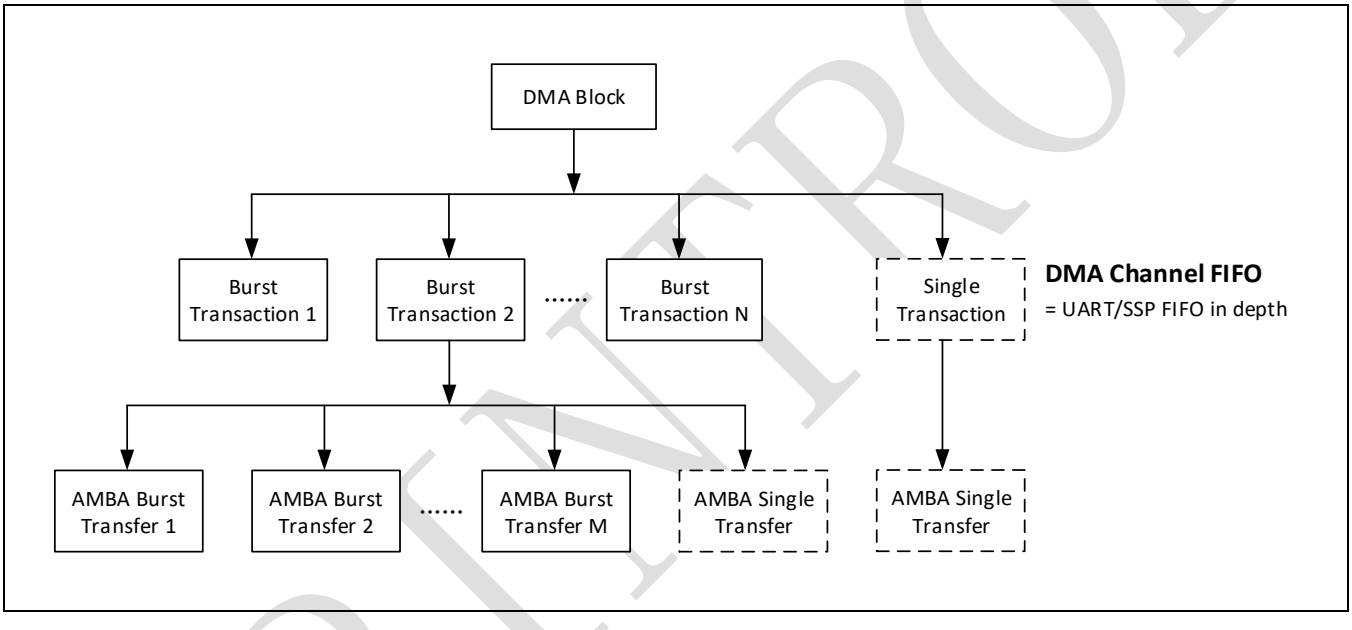
```
// Channel 0 configuration
DMAC->DMATCIE.all   = 0x0101;          // Enable channel 0 tansfer complete int
DMAC->DMAEN.all     = 0x1;            // DMA enable
DMACH0->DMACHSA.all = 0x20003000;      // Source address
DMACH0->DMACHDA.all = 0x20003300;      // Destination address
DMACH0->DMACHCTL0.bit.IE    = 0x1;    // Channel 0 INT enable
DMACH0->DMACHCTL0.bit.DINC  = 0x0;     // Destination address increment
DMACH0->DMACHCTL0.bit.SINC  = 0x0;     // Source address increment
DMACH0->DMACHCTL0.bit.TTFC  = 0x0;     // Memory to memory
DMACH0->DMACHCTL1.bit.BLKTS = 0xC;     // Block size = 12
DMACH0->DMACHEN.all         = 0x0101;  // Enable channel 0


// Run Time (Channel transfer complete triggered ISR call)
//=========================================
if(DMAC->DMATCIF.bit.CH0 == 1)
{
  DMAC->DMATCIC.all = 0x1; // Clear int
}
```

23.3.4 Memory-to-peripheral 以及 Peripheral-to-memory 传输操作

Memory-to-peripheral 和 peripheral-to-memory 都是非存储器传输。对于 DMA 和外设之间的传输，数据块被拆分成多个事务（单个事务或者并发事务），每个事务都是被来自于外设的请求初始化的，然后，每个事务会被转换成到非存储器外设的 AHB 传输（单个传输或者并发传输）。

必须要设置 DMA 控制器并发事务的长度，可以通过寄存器 DMACHCTL0.SSIZE/DSIZE 分别配置源端和目标端的并发事务长度。并发事务长度是指每次 DMA 请求的数据项的数量（例如：UART FIFO 的条目数量）。此外，并发事务的长度应该考虑 DMA FIFO 大小与源端和目标端外设 FIFO 大小之间的关系。

图 23-4：DMA 和非存储器外设之间的传输拆分

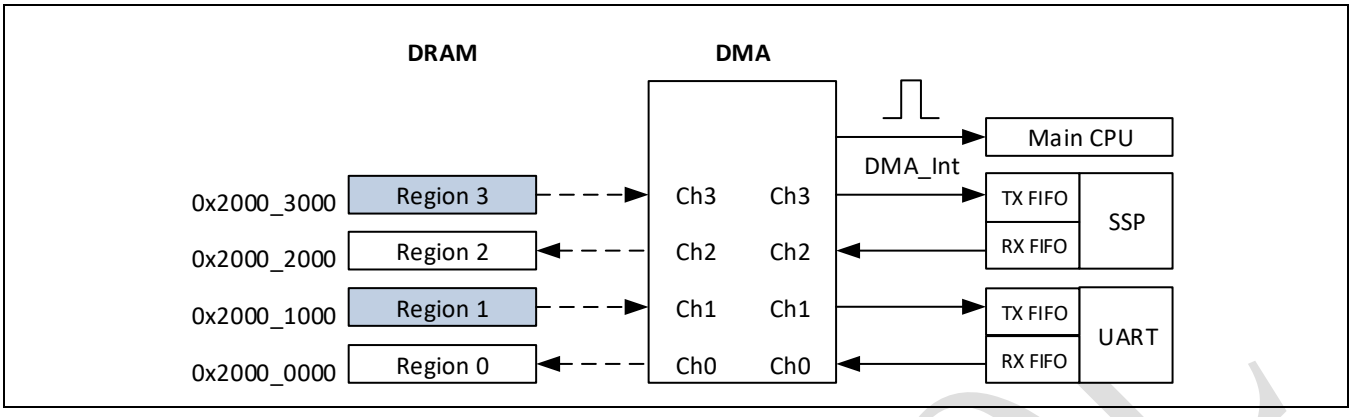


在 SPC2168 中，UART_TX 和 SSP_TX 使用 memory-to-peripheral 操作方式完成 DMA 传输；UART_RX 和 SSP_RX 使用 peripheral-to-memory 操作方式完成 DMA 传输。根据 UART 和 SSP 外设的 FIFO 大小配置 DMACHCTL.SSIZE 和 DMACHCTL.DSIZE 寄存器，从而避免 FIFO 上溢和下溢。

当 UART 和 SSP 不使用 DMA 传输时，尽管硬件握手接口是存在的，但是它不占用 DMA 通道资源，这些通道可以被用作 memory-to-memory 的传输。

下面给出一个示例，DMA 通道 0，1，2，3 被配置用于 DRAM 与 UART/SSP 之间的数据传输。数据存储在 DRAM 中。在每个通过数据传输结束时，都会产生一个中断通知 Main CPU。

图 23-5：DRAM 与 UART/SSP 之间的 DMA 传输



示例代码：存储器和 UART/SSP 之间 DMA 传输的配置

```
// Configure UART
UART->UARTFCR.bit.TRFIFOE    = 0x1;   // FIFO enable
UART->UARTFCR.bit.ITL        = 0x3;   // UART RX FIFO threshold = 8 words
UART->UARTFCR.bit.TIL        = 0x1;   // UART TX FIFO threshold = 0 words
UART->UARTFCR.bit.BUS        = 0x1;   // 32-bit bus
UART->UARTIER.bit.DMAE       = 0x1;   // Enable UART DMA
UART->UARTIER.bit.UUE        = 0x1;   // Enable UART

// Configure channel 0 for UART_RX
DMACH0->DMACHSA.all = 0x40004000;     // UARTRBR address
DMACH0->DMACHDA.all = 0x20000000;     // DRAM region 0 address
DMACH0->DMACHCTL0.bit.TT     = 0x2;   // Peripheral to memory
DMACH0->DMACHCTL0.bit.SSIZE  = 0x2;   // Transaction size = 8 words
DMACH0->DMACHCTL0.bit.SINC   = 0x2;   // Keep source address as UARTRBR register
DMACH0->DMACHCTL0.bit.DINC   = 0x0;   // Increase to next address in DRAM region
0
DMACH0->DMACHCTL0.bit.IE     = 0x1;   // Channel 0 interrupt enable
DMACH0->DMACHCTL1.bit.BLKTS  = 0x10;  // Block size = 16 words
DMACH0->DMACHCTL2.bit.SSWHS  = 0x0;   // Hardware handshaking
DMACH0->DMACHCTL2.bit.SHSPOL = 0x0;   // Active high
DMACH0->DMACHCTL3.bit.SPER   = 0x0;   // Channel 0 select UART RX as source

// Configure channel 1 for UART_TX
DMACH1->DMACHSA.all = 0x20001000;     // DRAM region 1 address
DMACH1->DMACHDA.all = 0x40004000;     // UARTTHR address
DMACH1->DMACHCTL0.bit.TT     = 0x1;   // Memory to peripheral
DMACH1->DMACHCTL0.bit.DSIZE  = 0x3;   // Transaction size = 16 words
DMACH1->DMACHCTL0.bit.SINC   = 0x0;   // Increase to next address in DRAM region
1
DMACH1->DMACHCTL0.bit.DINC   = 0x2;   // Keep destination address as UARTTHR
DMACH1->DMACHCTL0.bit.IE     = 0x1;   // Channel 1 interrupt enable
DMACH1->DMACHCTL1.bit.BLKTS  = 0x10;  // Block size = 16 words
DMACH1->DMACHCTL2.bit.DSWHS  = 0x0;   // Hardware handshaking
DMACH1->DMACHCTL2.bit.DSHPOL = 0x0;   // Active high
DMACH1->DMACHCTL3.bit.DPER   = 0x1;   // Channel 1 select UART TX as destionation

// Configure SSP FIFO
SSP->SSPCTL1.bit.DMARXEN     = 0x1;   // SSP RX DMA enable
SSP->SSPCTL1.bit.DMATXEN     = 0x1;   // SSP TX DMA enable
SSP->SSPCTL1.bit.RFTH        = 0x7;   // SSP RX FIFO threshold >= 8 words
```

```
SSP->SSPCTL1.bit.TFTH       = 0x8;   // SSP TX FIFO threshold <= 8 words

// Configure channel 2 for SSP_RX
DMACH2->DMACHSA.all = 0x40005010;     // SSPDATA register address
DMACH2->DMACHDA.all = 0x20002000;     // DRAM region 2 address
DMACH2->DMACHCTL0.bit.TT     = 0x2;   // Peripheral to Memory
DMACH2->DMACHCTL0.bit.SSIZE  = 0x2;   // Transaction size = 8 words
DMACH2->DMACHCTL0.bit.SINC   = 0x2;   // Keep source address as SSPDATA register
DMACH2->DMACHCTL0.bit.DINC   = 0x0;   // Increase to next address in DRAM region
2
DMACH2->DMACHCTL0.bit.IE     = 0x1;   // Enable channel 2 interrupt
DMACH2->DMACHCTL1.bit.BLKTS  = 0x8;   // Block size = 8 words
DMACH2->DMACHCTL2.bit.SSWHS  = 0x0;   // Hardware handshaking
DMACH2->DMACHCTL2.bit.SHSPOL = 0x0;   // Active high
DMACH2->DMACHCTL3.bit.SPER   = 0x2;   // Channel 2 select SSP RX

// Configure channel 3 for SSP_TX
DMACH3->DMACHSA.all = 0x2003000;      // DRAM region 3 address
DMACH3->DMACHDA.all = 0x40005010;     // SSPDATA register address
DMACH3->DMACHCTL0.bit.TT     = 0x1;   // Memory to Peripheral
DMACH3->DMACHCTL0.bit.SINC   = 0x0;   // Increase to next address in DRAM region
3
DMACH3->DMACHCTL0.bit.DINC   = 0x2;   // Keep destination address as SSPDATA
DMACH3->DMACHCTL0.bit.DSIZE  = 0x2;   // Transaction size = 8 words
DMACH3->DMACHCTL0.bit.IE     = 0x1;   // Enable channel 3 interrupt
DMACH3->DMACHCTL1.bit.BLKTS  = 0x11;  // Block size = 17 words
DMACH3->DMACHCTL2.bit.DSWHS  = 0x0;   // Hardware handshaking
DMACH3->DMACHCTL2.bit.DSHPOL = 0x0;   // Active high
DMACH3->DMACHCTL3.bit.DPER   = 0x3;   // Channel 3 select SSP TX

DMAC->DMATCIE.all            = 0xf0f; // Enable channel 0-3 transfer complete
int
DMAC->DMAEN.all           = 0x1;   // DMA enable
DMAC->DMACHEN.all         = 0xf0f; // Channel 0/1/2/3 enable

// Run Time (Channel transfer complete triggered ISR call)
//=========================================
while(DMAC->DMATCIF.all != 0xf);
DMAC->DMATCIC.all= 0xf;             // Clear the interrupt
```

23.3.5 Peripheral-to-peripheral 传输操作

SPC2168 为 peripheral-to-peripheral 类型的传输（UART_RX 到 SSP_TX 或者 SSP_RX 到 UART_TX）提供了物理通信通道。

下面的示例代码展示了通过 DMA 通道 0 实现 SSP_RX 向 UART_TX 传输数据。需要注意的是：要小心地选择 SSP 和 UART 通信的波特率，从而避免 FIFO 上溢和下溢。

示例代码

```
// Configure UART
UART->UARTFCR.bit.TRFIFOE    = 0x1;   // FIFO enable
UART->UARTFCR.bit.TIL        = 0x1;   // UART TX FIFO threshold = 0 words
UART->UARTFCR.bit.BUS        = 0x1;   // 32-bit bus
UART->UARTIER.bit.DMAE       = 0x1;   // Enable UART DMA
```

示例代码

```
UART->UARTIER.bit.UUE       = 0x1;   // Enable UART

// Configure SSP FIFO
SSP->SSPCTL1.bit.DMARXEN     = 0x1;   // SSP RX DMA enable
SSP->SSPCTL1.bit.RFTH       = 0x7;   // SSP RX FIFO threshold >= 8 words

// Configure channel 0 for SSP_RX to UART_TX
DMACH0->DMACHSA.all = 0x40005010;     // SSPDATA register address
DMACH0->DMACHDA.all = 0x40004000;     // UARTTHR address
DMACH0->DMACHCTL0.bit.TT     = 0x3;   // Peripheral to peripheral
DMACH0->DMACHCTL0.bit.SSIZE  = 0x2;   // Transaction size = 8 words
DMACH0->DMACHCTL0.bit.SINC   = 0x2;   // Keep source address as SSPDATA register
DMACH0->DMACHCTL0.bit.DINC   = 0x2;   // Keep destination address as UARTTHR
DMACH0->DMACHCTL0.bit.IE     = 0x1;   // Channel 0 interrupt enable
DMACH0->DMACHCTL1.bit.BLKTS  = 0x10;  // Block size = 16 words
DMACH0->DMACHCTL2.bit.SSWHS  = 0x0;   // Hardware handshaking
DMACH0->DMACHCTL2.bit.SHSPOL = 0x0;   // Active high
DMACH0->DMACHCTL3.bit.SPER   = 0x2;   // Select SSP RX as source
DMACH0->DMACHCTL3.bit.DPER   = 0x1;   // Select UART TX as destionation

DMAC->DMATCIE.all           = 0x101; // Enable channel 0 transfer complete int
DMAC->DMAEN.all            = 0x1;   // DMA enable
DMAC->DMACHEN.all           = 0x101; // Channel 0 enable

// Run Time (Channel transfer complete triggered ISR call)
//=========================================
while(DMAC->DMATCIF.bit.CH0 != 0x1);
DMAC->DMATCIC.all= 0x1;              // Clear the interrupt
```

23.4 寄存器

23.4.1 DMACH 寄存器列表

表 23-2：DMACH 模块基地址

外设模块	基地址
DMACH0	0x4000 A400
DMACH1	0x4000 A458
DMACH2	0x4000 A4B0
DMACH3	0x4000 A508
DMACH4	0x4000 A560
DMACH5	0x4000 A5B8

表 23-3：DMACH 寄存器列表

寄存器	偏移地址	描述	复位值
DMACHSA	0x00	DMA 通道源端地址寄存器	0x00000000
DMACHDA	0x08	DMA 通道目标端地址寄存器	0x00000000
DMACHCTL0	0x18	DMA 通道控制寄存器 0	0x00304825
DMACHCTL1	0x1C	DMA 通道控制寄存器 1	0x00000002
DMACHCTL2	0x40	DMA 通道控制寄存器 2	0x00000E00
DMACHCTL3	0x44	DMA 通道控制寄存器 3	0x00000004

23.4.2 DMACH 寄存器

表 23-4 和表 23-15 提供了 DMACH 模块相关的寄存器细节。

表 23-4：DMA 通道源端地址寄存器（DMACHSA）位段定义

DMACHSA (DMA Channel Source Address Register) Offset: 0x0 Default: 0x00000000 Access: DMACHx -> DMACHSA.all							
31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

表 23-5：DMA 通道源端地址寄存器（DMACHSA）位段描述

位段	位段名称	属性	复位值	描述
31:0	ADDR	RW	0x0	当前 DMA 传输的源端地址 写入本寄存器的数值必须是一个 32 位对齐的地址 在每次源端传输之后，会根据 DMACHCTL0.SINC 来更新本寄存器的值。

表 23-6：DMA 通道目标端地址寄存器（DMACHDA）位段定义

DMACHDA (DMA Channel Destination Address Register) Offset: 0x8 Default: 0x00000000 Access: DMACHx -> DMACHDA.all							
31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

表 23-7：DMA 通道目标端地址寄存器（DMACHDA）位段描述

位段	位段名称	属性	复位值	描述
31:0	ADDR	RW	0x0	当前 DMA 传输的目标端地址 写入本寄存器的数值必须是一个 32 位对齐的地址 在每次源端传输之后，会根据 DMACHCTL0.DINC 来更新本寄存器的值。

表 23-8：DMA 通道控制寄存器 0（DMACHCTL0）位段定义

DMACHCTL0 (DMA Channel Control Register 0) Offset: 0x18 Default: 0x00304825
Access: DMACHx -> DMACHCTL0.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED		TT		RESERVED			SSIZE
15	14	13	12	11	10	9	8
SSIZE		DSIZE			SINC		DINC
7	6	5	4	3	2	1	0
DINC	SWIDTH			DWIDTH			IE

表 23-9：DMA 通道控制寄存器 0（DMACHCTL0）位段描述

位段	位段名称	属性	复位值	描述
31:22	RESERVED_31_22	RO	0x0	保留
21:20	TT	RW	0x3	传输类型 00：Memory-to-Memory 01：Memory-to-Peripheral 10：Peripheral-to-Memory 11：Peripheral-to-Peripheral（例如：UART_RX 到 SSP_TX 或者 SSP_RX 到 UART_TX）
19:17	RESERVED_19_17	RO	0x0	保留
16:14	SSIZE	RW	0x1	源端并发事务大小 这个数值和 AHB HBURST 信号无关 000：1 个字 001：4 个字 010：8 个字 011：16 个字 100：32 个字 101：64 个字 110：128 个字 111：256 个字
13:11	DSIZE	RW	0x1	目标端并发事务大小 这个数值和 AHB HBURST 信号无关 000：1 个字 001：4 个字 010：8 个字 011：16 个字 100：32 个字 101：64 个字 110：128 个字 111：256 个字

位段	位段名称	属性	复位值	描述
10:9	SINC	RW	0x0	每次传输之后，源端地址递增方式 00：递增到下一个字对齐的地址 01：递减到下一个字对齐的地址 10：地址固定不变 11：无效
8:7	DINC	RW	0x0	每次传输之后，目标端地址递增方式 00：递增到下一个字对齐的地址 01：递减到下一个字对齐的地址 10：地址固定不变 11：无效
6:4	SWIDTH	RO	0x2	总是读回 0x2，表示源端传输的位宽为 32-bit
3:1	DWIDTH	RO	0x2	总是读回 0x2，表示目标端传输的位宽为 32-bit
0	IE	RW	0x1	中断使能 该位不影响中断原始标志位的产生 0：关闭中断 1：使能中断

表 23-10：DMA 通道控制寄存器 1（DMACHCTL1）位段定义

DMACHCTL1 (DMA Channel Control Register 1)　Offset: 0x1C　Default: 0x00000002
Access: DMACHx -> DMACHCTL1.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED			RESERVED	BLKTS			
7	6	5	4	3	2	1	0
BLKTS							

表 23-11：DMA 通道控制寄存器 1（DMACHCTL1）位段描述

位段	位段名称	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12	RESERVED_12	RW	0x0	保留
11:0	BLKTS	RW	0x2	数据块大小，指的是完成一个数据块传输所需的单个事务的数量 该位段必须在通道使能之前设定

表 23-12：DMA 通道控制寄存器 2（DMACHCTL2）位段定义

DMACHCTL2 (DMA Channel Control Register 2) Offset: 0x40 Default: 0x00000E00
Access: DMACHx -> DMACHCTL2.all

31	30	29	28	27	26	25	24
RESERVED_31_20							
23	22	21	20	19	18	17	16
RESERVED_31_20				SHSPOL	DHSPOL	RESERVED_17_12	
15	14	13	12	11	10	9	8
RESERVED_17_12				SSWHS	DSWHS	FIFOEMPTY	SUSPEND
7	6	5	4	3	2	1	0
PRIORITY			RESERVED_4_0				

表 23-13：DMA 通道控制寄存器 2（DMACHCTL2）位段描述

位段	位段名称	属性	复位值	描述
31:20	RESERVED_31_20	RO	0x0	保留
19	SHSPOL	RW	0x0	源端握手接口极性 0：高电平有效 1：低电平有效
18	DSHPOL	RW	0x0	目标端握手接口极性 0：高电平有效 1：低电平有效
17:12	RESERVED_17_12	RO	0x0	保留
11	SSWHS	RW	0x1	源端握手类型选择 如果源端是存储器，该位段被忽略 0：硬件握手 1：软件握手
10	DSWHS	RW	0x1	目标端握手类型选择 如果目的端是存储器，该位段被忽略 0：硬件握手 1：软件握手
9	FIFOEMPTY	RO	0x1	通道 FIFO 中存在数据标志 该位段可以和 SUSPEND 位段联合使用，可以保证在不丢失数据的情况下安全关闭通道 0：通道 FIFO 为空 1：通道 FIFO 有数据
8	SUSPEND	RW	0x0	通道挂起 挂起所有 DMA 源端的数据传输直到该位段被清除。不保证完成当前的传输事务。 该位段还可以和 FIFOEMPTY 位段联合使用，以达到在不丢失数据的情况下安全关闭通道 0：通道正在使用中

位段	位段名称	属性	复位值	描述
				1：挂起通道
7:5	PRIORITY	RW	0x0	通道优先级 000：第六优先级（最低） 001：第五优先级 010：第四优先级 011：第三优先级 100：第二优先级 101：第一优先级（最高） 其它：无效
4:0	RESERVED_4_0	RO	0x0	保留

表 23-14：DMA 通道控制寄存器 3（DMACHCTL3）位段定义

DMACHCTL3 (DMA Channel Control Register 3)　Offset: 0x44　Default: 0x00000004
Access: DMACHx -> DMACHCTL3.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED			DPER		RESERVED		SPER
7	6	5	4	3	2	1	0
SPER	RESERVED		RESERVED			FIFOTH	RESERVED

表 23-15：DMA 通道控制寄存器 3（DMACHCTL3）位段描述

位段	位段名称	属性	复位值	描述
31:13	RESERVED_31_13	RO	0x0	保留
12:11	DPER	RW	0x0	通过硬件握手接口进行通信的目标端外设 如果 DMACHCT2.DSWHS=1，则忽略该位段。 01：硬件握手接口是 UART_TX 11：硬件握手接口是 SSP_TX 其它：无效
10:9	RESERVED_10_9	RO	0x0	保留
8:7	SPER	RW	0x0	通过硬件握手接口进行通信的源端外设 如果 DMACHCT2.SSWHS=1，则忽略该位段。 00：硬件握手接口是 UART_RX 10：硬件握手接口是 SSP_RX 其它：无效
6:5	RESERVED_6_5	RO	0x0	保留
4:2	RESERVED_4_2	RW	0x1	保留

位段	位段名称	属性	复位值	描述
1	FIFOTH	RW	0x0	FIFO 阈值 用来确定在触发一次并发事务之前，FIFO 中还需要多少空间或者数据 0：只要 FIFO 中有数据就传输 1：目标端在 FIFO 中的数据大于或等于 8 个字后开始传输；源端在 FIFO 中的数据大于 8 个字后开始传输 例外情形是在并发事务或者块传输结束时。
0	RESERVED_0	RW	0x0	保留

23.4.3 DMAC 寄存器列表

表 23-16：DMAC 模块基地址

外设模块	基地址
DMAC	0x4000 A400

表 23-17：DMAC 寄存器列表

寄存器	偏移地址	描述	复位值
DMATCRAWIF	0x2C0	DMA 传输完成中断原始标志寄存器	0x00000000
DMAERRRAWIF	0x2E0	DMA 错误中断原始标志寄存器	0x00000000
DMATCIF	0x2E8	DMA 传输完成中断标志寄存器	0x00000000
DMAERRIF	0x308	DMA 错误中断标志寄存器	0x00000000
DMATCIE	0x310	DMA 传输完成中断使能寄存器	0x00000000
DMAERRIE	0x330	DMA 错误中断使能寄存器	0x00000000
DMATCIC	0x338	DMA 传输完成中断清除寄存器	0x00000000
DMAERRIC	0x358	DMA 错误中断清除寄存器	0x00000000
DMAIF	0x360	DMA 全局中断标志寄存器	0x00000000
DMAEN	0x398	DMA 使能寄存器	0x00000000
DMACHEN	0x3A0	DMA 通道使能寄存器	0x00000000

23.4.4 DMAC 寄存器

表 23-18 和表 23-39 提供了 DMAC 相关的寄存器细节。

表 23-18：DMA 传输完成中断原始标志寄存器（DMATCRAWIF）位段定义

DMATCRAWIF (DMA Transfer Complete Raw Interrupt Flag Register) Offset: 0x2C0 Default: 0x00000000 Access: DMAC -> DMATCRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-19：DMA 传输完成中断原始标志寄存器（DMATCRAWIF）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	RW[1]	0x0	通道 5 传输完成中断原始标识 0：传输没有完成 1：传输已完成
4	CH4	RW[1]	0x0	通道 4 传输完成中断原始标识 0：传输没有完成 1：传输已完成
3	CH3	RW[1]	0x0	通道 3 传输完成中断原始标识 0：传输没有完成 1：传输已完成
2	CH2	RW[1]	0x0	通道 2 传输完成中断原始标识 0：传输没有完成 1：传输已完成
1	CH1	RW[1]	0x0	通道 1 传输完成中断原始标识 0：传输没有完成 1：传输已完成
0	CH0	RW[1]	0x0	通道 0 传输完成中断原始标识 0：传输没有完成 1：传输已完成

[1] 这些寄存器位段可写仅仅是为了给软件测试提供方便，在正常操作中，不推荐写这些寄存器位段。

表 23-20：DMA 错误中断原始标志寄存器（DMAERRRAWIF）位段定义

DMAERRRAWIF (DMA Error Raw Interrupt Flag Register) Offset: 0x2E0 Default: 0x00000000 Access: DMAC -> DMAERRRAWIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-21：DMA 错误中断原始标志寄存器（DMAERRRAWIF）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	RW[1]	0x0	通道 5 错误中断原始标识 0：错误未发生 1：错误已发生
4	CH4	RW[1]	0x0	通道 4 错误中断原始标识 0：错误未发生 1：错误已发生
3	CH3	RW[1]	0x0	通道 3 错误中断原始标识 0：错误未发生 1：错误已发生
2	CH2	RW[1]	0x0	通道 2 错误中断原始标识 0：错误未发生 1：错误已发生
1	CH1	RW[1]	0x0	通道 1 错误中断原始标识 0：错误未发生 1：错误已发生
0	CH0	RW[1]	0x0	通道 0 错误中断原始标识 0：错误未发生 1：错误已发生

[1] 这些寄存器位段可写仅仅是为了给软件测试提供方便，在正常操作中，不推荐写这些寄存器位段。

表 23-22：DMA 传输完成中断标志寄存器（DMATCIF）位段定义

DMATCIF (DMA Transfer Complete Interrupt Flag Register) Offset: 0x2E8 Default: 0x00000000 Access: DMAC -> DMATCIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-23：DMA 传输完成中断标志寄存器（DMATCIF）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	RO	0x0	通道 5 传输完成中断标志 0：中断未发生 1：中断已发生
4	CH4	RO	0x0	通道 4 传输完成中断标志 0：中断未发生 1：中断已发生
3	CH3	RO	0x0	通道 3 传输完成中断标志 0：中断未发生 1：中断已发生
2	CH2	RO	0x0	通道 2 传输完成中断标志 0：中断未发生 1：中断已发生
1	CH1	RO	0x0	通道 1 传输完成中断标志 0：中断未发生 1：中断已发生
0	CH0	RO	0x0	通道 0 传输完成中断标志 0：中断未发生 1：中断已发生

表 23-24：DMA 错误中断标志寄存器（DMAERRIF）位段定义

DMAERRIF (DMA Error Interrupt Flag Register) Offset: 0x308 Default: 0x00000000 Access: DMAC -> DMAERRIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-25：DMA 错误中断标志寄存器（DMAERRIF）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	RO	0x0	通道 5 错误中断标志 0：中断未发生 1：中断已发生
4	CH4	RO	0x0	通道 4 错误中断标志 0：中断未发生 1：中断已发生
3	CH3	RO	0x0	通道 3 错误中断标志 0：中断未发生 1：中断已发生
2	CH2	RO	0x0	通道 2 错误中断标志 0：中断未发生 1：中断已发生
1	CH1	RO	0x0	通道 1 错误中断标志 0：中断未发生 1：中断已发生
0	CH0	RO	0x0	通道 0 错误中断标志 0：中断未发生 1：中断已发生

表 23-26：DMA 传输完成中断使能寄存器（DMATCIE）位段定义

DMATCIE (DMA Transfer Complete Interrupt Enable Register) Offset: 0x310 Default: 0x00000000 Access: DMAC -> DMATCIE.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED		CH5WE	CH4WE	CH3WE	CH2WE	CH1WE	CH0WE
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-27：DMA 传输完成中断使能寄存器（DMATCIE）位段描述

位段	位段名称	属性	复位值	描述
31:14	RESERVED_31_14	RO	0x0	保留
13	CH5WE	W1S	0x0	CH5 位段写使能 0：禁止 1：使能，此位段自动清零
12	CH4WE	W1S	0x0	CH4 位段写使能 0：禁止 1：使能，此位段自动清零
11	CH3WE	W1S	0x0	CH3 位段写使能 0：禁止 1：使能，此位段自动清零
10	CH2WE	W1S	0x0	CH2 位段写使能 0：禁止 1：使能，此位段自动清零
9	CH1WE	W1S	0x0	CH1 位段写使能 0：禁止 1：使能，此位段自动清零
8	CH0WE	W1S	0x0	CH0 位段写使能 0：禁止 1：使能，此位段自动清零
7:6	RESERVED_7_6	RO	0x0	保留
5	CH5	RW	0x0	通道 5 传输完成中断使能 0：关闭 1：使能
4	CH4	RW	0x0	通道 4 传输完成中断使能 0：关闭 1：使能

位段	位段名称	属性	复位值	描述
3	CH3	RW	0x0	通道 3 传输完成中断使能 0：关闭 1：使能
2	CH2	RW	0x0	通道 2 传输完成中断使能 0：关闭 1：使能
1	CH1	RW	0x0	通道 1 传输完成中断使能 0：关闭 1：使能
0	CH0	RW	0x0	通道 0 传输完成中断使能 0：关闭 1：使能

表 23-28：DMA 错误中断使能寄存器（DMAERRIE）位段定义

DMAERRIE (DMA Error Interrupt Enable Register)　Offset: 0x330　Default: 0x00000000
Access: DMAC -> DMAERRIE.all

31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED		CH5WE	CH4WE	CH3WE	CH2WE	CH1WE	CH0WE
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-29：DMA 错误中断使能寄存器（DMAERRIE）位段描述

位段	位段名称	属性	复位值	描述
31:14	RESERVED_31_14	RO	0x0	保留
13	CH5WE	W1S	0x0	CH5 位段写使能 0：禁止 1：使能，此位段自动清零
12	CH4WE	W1S	0x0	CH4 位段写使能 0：禁止 1：使能，此位段自动清零
11	CH3WE	W1S	0x0	CH3 位段写使能 0：禁止 1：使能，此位段自动清零
10	CH2WE	W1S	0x0	CH2 位段写使能 0：禁止 1：使能，此位段自动清零

位段	位段名称	属性	复位值	描述
9	CH1WE	W1S	0x0	CH1 位段写使能 0：禁止 1：使能，此位段自动清零
8	CH0WE	W1S	0x0	CH0 位段写使能 0：禁止 1：使能，此位段自动清零
7:6	RESERVED_7_6	RO	0x0	保留
5	CH5	RW	0x0	通道 5 错误中断使能 0：关闭 1：使能
4	CH4	RW	0x0	通道 4 错误中断使能 0：关闭 1：使能
3	CH3	RW	0x0	通道 3 错误中断使能 0：关闭 1：使能
2	CH2	RW	0x0	通道 2 错误中断使能 0：关闭 1：使能
1	CH1	RW	0x0	通道 1 错误中断使能 0：关闭 1：使能
0	CH0	RW	0x0	通道 0 错误中断使能 0：关闭 1：使能

表 23-30：DMA 传输完成中断清除寄存器（DMATCIC）位段定义

DMATCIC (DMA Transfer Complete Interrupt Clear Register) Offset: 0x338 Default: 0x00000000 Access: DMAC -> DMATCIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-31：DMA 传输完成中断清除寄存器（DMATCIC）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	W1C	0x0	通道 5 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零
4	CH4	W1C	0x0	通道 4 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零
3	CH3	W1C	0x0	通道 3 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零
2	CH2	W1C	0x0	通道 2 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零
1	CH1	W1C	0x0	通道 1 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零
0	CH0	W1C	0x0	通道 0 传输完成中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标志位，该位段自动清零

表 23-32：DMA 错误中断清除寄存器（DMAERRIC）位段定义

DMAERRIC (DMA Error Interrupt Clear Register) Offset: 0x358 Default: 0x00000000 Access: DMAC -> DMAERRIC.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-33：DMA 错误中断清除寄存器（DMAERRIC）位段描述

位段	位段名称	属性	复位值	描述
31:6	RESERVED_31_6	RO	0x0	保留
5	CH5	W1C	0x0	通道 5 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零
4	CH4	W1C	0x0	通道 4 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零
3	CH3	W1C	0x0	通道 3 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零
2	CH2	W1C	0x0	通道 2 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零
1	CH1	W1C	0x0	通道 1 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零
0	CH0	W1C	0x0	通道 0 错误中断清除 0：写 0 无效，总是读回 0 1：写 1 清除 RAWIF 和 IF 标识位，该位段自动清零

表 23-34：DMA 全局中断标志寄存器（DMAIF）位段定义

DMAIF (DMA Combined Interrupt Flag Register) Offset: 0x360 Default: 0x00000000 Access: DMAC -> DMAIF.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED			ERROR	RESERVED			TC

表 23-35：DMA 全局中断标志寄存器（DMAIF）位段描述

位段	位段名称	属性	复位值	描述
31:5	RESERVED_31_5	RO	0x0	保留
4	ERROR	RO	0x0	全局错误中断标志 此位段值是 DMAERRIF 寄存器位逻辑或的结果 0：没有错误中断产生 1：有错误中断产生
3:1	RESERVED_3_1	RO	0x0	保留
0	TC	RO	0x0	全局传输完成中断标志 此位段值是 DMATCIF 寄存器位逻辑或的结果 0：没有传输完成中断产生 1：有传输完成中断产生

表 23-36：DMA 使能寄存器（DMAEN）位段定义

DMAEN (DMA Enable Register) Offset: 0x398 Default: 0x00000000 Access: DMAC -> DMAEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED							
7	6	5	4	3	2	1	0
RESERVED							EN

表 23-37：DMA 使能寄存器（DMAEN）位段描述

位段	位段名称	属性	复位值	描述
31:1	RESERVED_31_1	RO	0x0	保留
0	EN	RW	0x0	DMA 使能 当清零该位时，如果仍有通道在进行数据传输，那么该位段读回的值会保持为 1，直到所有通道上的传输终止，此时该位段读回的值才为 0 0：关闭 DMA 1：使能 DMA

表 23-38：DMA 通道使能寄存器（DMACHEN）位段定义

DMACHEN (DMA Channel Enable Register) Offset: 0x3A0 Default: 0x00000000 Access: DMAC -> DMACHEN.all							
31	30	29	28	27	26	25	24
RESERVED							
23	22	21	20	19	18	17	16
RESERVED							
15	14	13	12	11	10	9	8
RESERVED		CH5WE	CH4WE	CH3WE	CH2WE	CH1WE	CH0WE
7	6	5	4	3	2	1	0
RESERVED		CH5	CH4	CH3	CH2	CH1	CH0

表 23-39：DMA 通道使能寄存器（DMACHEN）位段描述

位段	位段名称	属性	复位值	描述
31:14	RESERVED_31_14	RO	0x0	保留
13	CH5WE	W1S	0x0	CH5 位段写使能 0：禁止 1：使能，此位段自动清零
12	CH4WE	W1S	0x0	CH4 位段写使能 0：禁止 1：使能，此位段自动清零
11	CH3WE	W1S	0x0	CH3 位段写使能 0：禁止 1：使能，此位段自动清零
10	CH2WE	W1S	0x0	CH2 位段写使能 0：禁止 1：使能，此位段自动清零
9	CH1WE	W1S	0x0	CH1 位段写使能 0：禁止 1：使能，此位段自动清零
8	CH0WE	W1S	0x0	CH0 位段写使能 0：禁止 1：使能，此位段自动清零
7:6	RESERVED_7_6	RO	0x0	保留
5	CH5	RW	0x0	通道 5 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能

位段	位段名称	属性	复位值	描述
4	CH4	RW	0x0	通道 4 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能
3	CH3	RW	0x0	通道 3 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能
2	CH2	RW	0x0	通道 2 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能
1	CH1	RW	0x0	通道 1 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能
0	CH0	RW	0x0	通道 0 使能 当传输完成之后，此位段自动清零 0：关闭 1：使能

24 邮箱（Mailbox）

24.1 邮箱概述

SPC2168 邮箱用来完成 Main CPU 和 CAU 之间的消息通信。当 Main CPU 已经将消息发送到邮箱时，它可以触发一个中断去通知 CAU 接收这个消息；在 CAU 收到消息后，它可以通过清除由 Main CPU 触发的中断来通知 Main CPU，反之亦然。

24.2 邮箱架构

如图 24-1 所示，邮箱包含两个消息 RAM（Message RAM）和两个中断触发单元。表 24-1 显示了消息 RAM 和中断寄存器的详细信息。

24.2.1 消息 RAM

有两个消息 RAM，用于 Main CPU 与 CAU 之间的数据共享和通信。消息 RAM 总是同时映射到 Main CPU 和 CAU 的存储器空间，并且可以按字节，半字和字进行访问。每个消息 RAM 的大小为 64 x 32 比特。

- CAU 到 Main CPU 消息 RAM

  CAU 可以使用该存储器块将数据传递到 Main CPU。 该存储器块，对 CAU 来说，是可读和可写的；对 Main CPU 来说，也是可读的，但是 Main CPU 的写操作会被忽略。

- Main CPU 到 CAU 消息 RAM

  Main CPU 可以使用该存储器块将数据传递到 CAU。该消息 RAM，对 Main CPU 来说，是可读和可写的；对 CAU 来说，可以执行读操作，但是 CAU 写操作会被忽略。

24.2.2 软件中断

Main CPU 和 CAU 之间有两个用于事件指示的中断触发单元。

- CAU 到 Main CPU 的软件中断

  CAU 可以在任何时刻通过向 C2MINTCTL 寄存器写入任意数据（0xCA17CE1 除外），来向 Main CPU 发出软件中断。该中断可以通过下面的方式清除：
  - Main CPU 向 C2MINTCTL 寄存器写入任意数据
  - CAU 向 C2MINTCTL 寄存器写入数据 0xCA17CE1

- Main CPU 到 CAU 的软件中断

  Main CPU 可以在任何时刻通过向 M2CINTCTL 寄存器写入任意数据（0xCA17CE1 除外），来向 CAU 发出软件中断。该中断可以通过下面的方式清除：
  - CAU 向 M2CINTCTL 寄存器写入任意数据
  - Main CPU 向 M2CINTCTL 寄存器写入数据 0xCA17CE1

图 24-1：邮箱架构

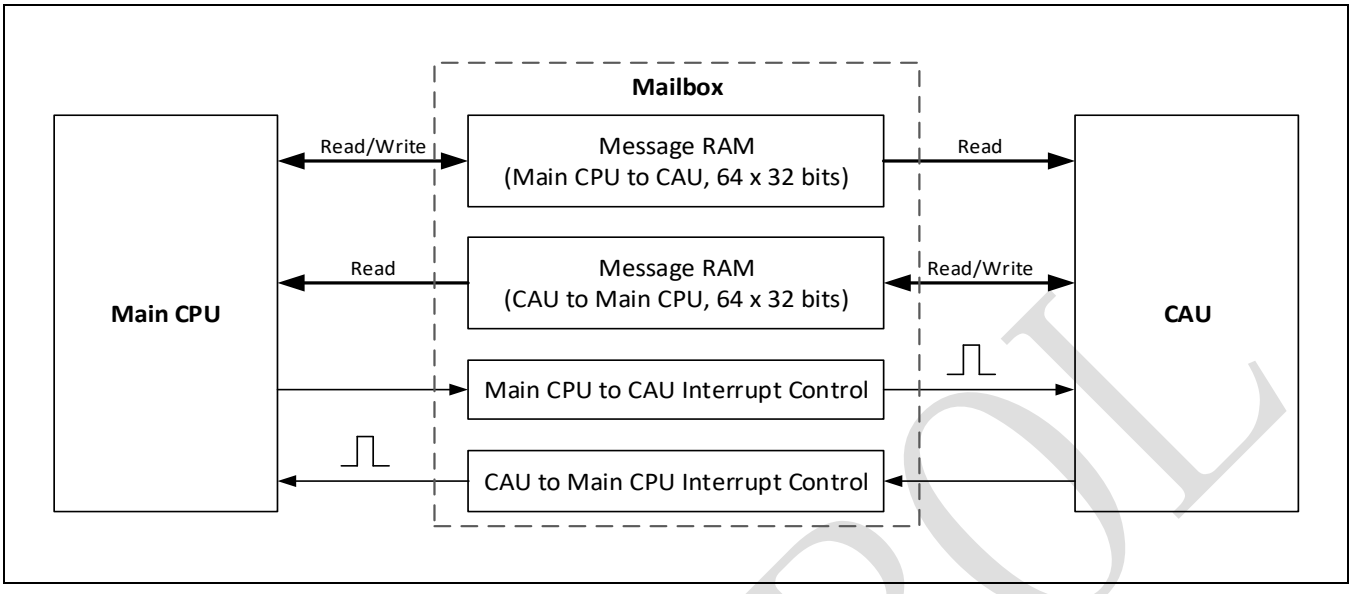


表 24-1：邮箱构成

模块	名称	基地址	大小
消息 RAM	Main CPU 到 CAU	0x2000 8000	256 字节
	CAU 到 Main CPU	0x2000 8100	256 字节
中断控制寄存器	M2CINTCTL	0x2000 8200	4 字节
	C2MINTCTL	0x2000 8204	4 字节

24.3 邮箱操作流程

图 24-2 显示了从 Main CPU 到 CAU 的数据传输的邮箱操作流程。

步骤 1：Main CPU 将消息数据写入消息 RAM（Main CPU 到 CAU）

步骤 2：Main CPU 通过向 M2CINTCTL 寄存器写入“1”（或其他任意值）来向 CAU 发出中断

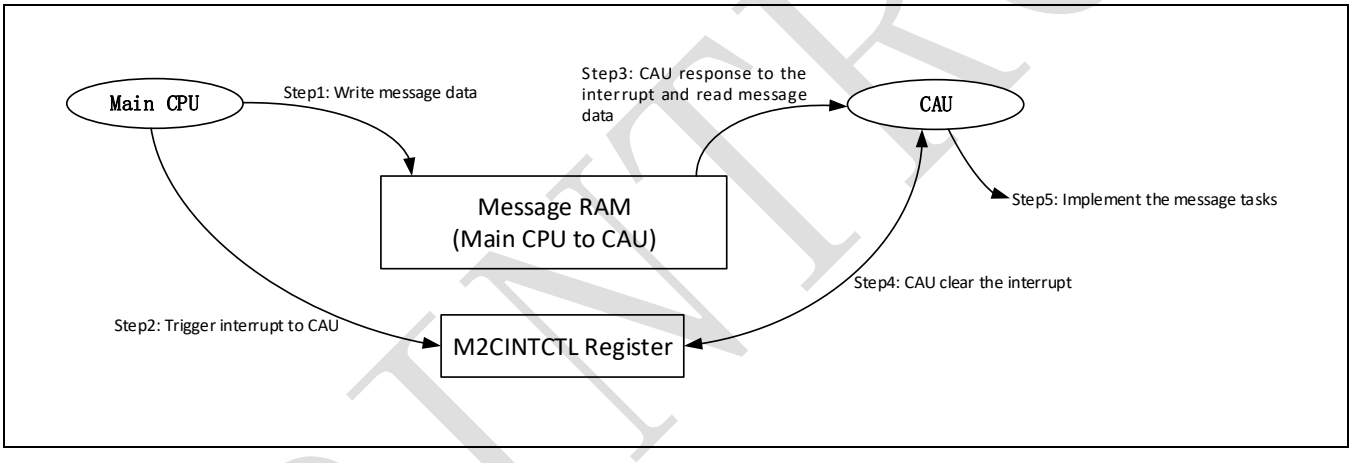
步骤 3：CAU 响应中断并从消息 RAM（Main CPU 到 CAU）读取消息数据

步骤 4：CAU 通过向 M2CINTCTL 寄存器写入“1”（或其他任意值）来清除中断

步骤 5：CAU 解析消息数据并执行相应的任务

在退出中断服务程序之前，必须清除由 Main CPU 或 CAU 发出的中断。CAU 向 Main CPU 发送数据的操作流程是相同的。

图 24-2：邮箱操作流程 – Main CPU 发送数据到 CAU



24.4 寄存器

24.4.1 邮箱寄存器列表

表 24-2：邮箱模块基地址

外设模块	基地址
MAILBOX	0x2000 8200

表 24-3：邮箱寄存器列表

寄存器	偏移地址	描述	复位值
M2CINTCTL	0x00	Main CPU 触发 CAU 中断控制寄存器	0x00000000
C2MINTCTL	0x04	CAU 触发 Main CPU 中断控制寄存器	0x00000000

24.4.2 邮箱寄存器

表 24-4 和表 24-7 提供了邮箱模块相关的寄存器细节。

表 24-4：Main CPU 触发 CAU 中断控制寄存器（M2CINTCTL）位段定义

M2CINTCTL (Main CPU Trigger CAU Interrupt Control Register) Offset: 0x0 Default: 0x00000000 Access: MAILBOX -> M2CINTCTL.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 24-5：Main CPU 触发 CAU 中断控制寄存器（M2CINTCTL）位段描述

位段	位段名称	属性	复位值	描述
31:0	VAL	RW	0x0	Main CPU 触发 CAU 中断的产生与清除控制 产生： Main CPU 对该寄存器写入任意数据（0xCA17CE1 除外）会产生一个中断到 CAU 清除： Main CPU 向该寄存器写入 0xCA17CE1 或者 CAU 向该寄存器写入任意数据将清除 Main CPU 到 CAU 的中断

表 24-6：CAU 触发 Main CPU 中断控制寄存器（C2MINTCTL）位段定义

C2MINTCTL (CAU Trigger Main CPU Interrupt Control Register) Offset: 0x4 Default: 0x00000000 Access: MAILBOX -> C2MINTCTL.all							
31	30	29	28	27	26	25	24
VAL							
23	22	21	20	19	18	17	16
VAL							
15	14	13	12	11	10	9	8
VAL							
7	6	5	4	3	2	1	0
VAL							

表 24-7：CAU 触发 Main CPU 中断控制寄存器（C2MINTCTL）位段描述

位段	位段名称	属性	复位值	描述
31:0	VAL	RW	0x0	CAU 触发 Main CPU 中断的产生与清除控制 产生： CAU 对该寄存器写入任意数据（0xCA17CE1 除外）会产生一个中断到 Main CPU 清除： CAU 向该寄存器写入 0xCA17CE1 或者 Main CPU 向该寄存器写入任意数据将清除 CAU 到 Main CPU 的中断

25 调试行为（Debug Behavior）

本章描述一些外设的调试行为。

25.1 看门狗定时器

当 CPU 进入 HALT 模式时，看门狗定时器（WDT）可以暂停运行或者保持正常运行。可以通过 WDTCTL 寄存器的 HALTEDRUN 位段来控制 WDT 的调试功能，如表 25-1 所示。

表 25-1：WDT 调试行为

WDTCTL.HALTEDRUN	描述
0	当 CPU 进入 HALT 模式，WDT 计数器立即停止
1	当 CPU 进入 HALT 模式，WDT 计数器保持正常计数

25.2 通用定时器

通用定时器没有特殊的调试特性。

25.3 PWM

当 CPU 进入 HALT 模式时，PWM 可以暂停运行或者继续保持运行。可以通过 TBCTL 寄存器的 DBGRUN 位段来控制 PWM 的调试功能，如表 25-2 所示。

表 25-2：PWM 调试行为

TBCTL.DBGRUN	描述
0	当 CPU 进入 HALT 模式，PWM 时基计数器在完成下一拍计数后停止
1	当 CPU 进入 HALT 模式，PWM 时基计数器在完成整个计数周期后停止（向上计数模式为 TBCNT=TBPRD，其余模式计数至 TBCNT=0）
2	当 CPU 进入 HALT 模式，PWM 时基计数器保持正常计数

25.4 ECAP

当 CPU 进入 HALT 模式时，ECAP 可以暂停运行或者继续保持运行。可以通过 CAPCTL 寄存器的 DBGRUN 位段来控制 ECAP 的调试功能，如表 25-3 所示。

表 25-3：ECAP 调试行为

TBCTL.DBGRUN	描述
0	当 CPU 进入 HALT 模式，ECAP 时间戳计数器立即停止
1	当 CPU 进入 HALT 模式，ECAP 时间戳计数器在计数到 0 时停止
2	当 CPU 进入 HALT 模式，ECAP 时间戳计数器保持正常计数

25.5 UART

外部调试器可以通过调试接口读 UART 模块的寄存器。需要注意的是，Debug 读操作可以产生和 CPU 读操作相同的效果。表 25-4 列出了一些寄存器的调试行为。

表 25-4：UART 调试行为

寄存器	描述
UARTRBR	在非 FIFO 模式下，Debug 读该寄存器会清空 UARTRBR 寄存器。在 FIFO 模式下，Debug 读该寄存器会获得 RXFIFO 最前面的数据字节，与此同时，UARTFOR 寄存器的值也会减 1。
UARTIIR.NIP UARTIIR.IID UARTIIR.ABL	Debug 读 UARTIIR 寄存器会清除这些寄存器位段。

25.6 SSP

外部调试器可以通过调试接口读 SSP 模块的寄存器。需要注意的是，Debug 读操作可以产生和 CPU 读操作相同的效果。表 25-5 列出了一些寄存器的调试行为。

表 25-5：SSP 调试行为

寄存器	描述
SSPDATA	如果 RXFIFO 非空，Debug 读该寄存器会获得 RXFIFO 最前面的数据，与此同时，SSPSTS.RXLVL 寄存器位段的值也会减 1。

25.7 I2C

外部调试器可以通过调试接口读 I2C 模块的寄存器。需要注意的是，Debug 读操作可以产生和 CPU 读操作相同的效果。表 25-5 列出了一些寄存器的调试行为。

表 25-6：I2C 调试行为

寄存器	描述
I2CDATACMD	如果 RXFIFO 非空且 I2CDATACMD.CMD = 1，Debug 读该寄存器会获得 RXFIFO 最前面的数据，与此同时，I2CRFLVL 寄存器的值也会减 1。
I2CINTCLR	Debug 读该寄存器会清除所有中断标志位以及 I2CTXABRTSRC 寄存器。 I2CTXABRTSRC.STARTNORESTART 寄存器位是清除 I2CTXABRTSRC 寄出器的一个例外。
I2CRXUDFCLR	Debug 读该寄存器会清除 RXUDF 标志
I2CRXOVFCLR	Debug 读该寄存器会清除 RXOVF 标志
I2CTXOVFCLR	Debug 读该寄存器会清除 TXOVF 标志
I2CRDREQCLR	Debug 读该寄存器会清除 RDREQ 标志
I2CTXABRTCLR	Debug 读该寄存器会清除 TXABRT 标志和 I2CTXABRTSRC 寄存器

寄存器	描述
	I2CTXABRTSRC.STARTNORESTART 寄存器位是清除 I2CTXABRTSRC 寄出器的一个例外。
I2CRXDONECLR	Debug 读该寄存器会清除 RXDONE 标志
I2CACTCLR	如果 I2C 不在活动时，Debug 读该寄存器会清除 ACT 标志
I2CSTOPDETCLR	Debug 读该寄存器会清除 STOPDET 标志
I2CSTARTDETCLR	Debug 读该寄存器会清除 STARTDET 标志
I2CGENCALLCLR	Debug 读该寄存器会清除 GENCALL 标志