MSPM0 G 系列 80MHz 微控制器

Technical Reference Manual

Literature Number: ZHCUB41
JUNE 2023

目录

关于本手册

本手册介绍了 MSPM0G 系列器件的模块和外设。每个说明都给出了一般意义上的模块或外设。目前所展示的并没有涵盖器件上所有模块或外围设备的所有特性和功能。此外，模块或外设在不同器件上的具体实现可能有所不同。引脚功能、内部信号连接和操作参数都因器件不同而各异。有关这些详细信息，请参阅特定于器件的数据表。

命名惯例

本文档使用以下惯例。

- 十六进制数可以用后缀 h 或前缀 0x 显示。 例如、以下数字是十六进制的 40 (十进制的 64)：40h 或 0x40。
- 本文档中的寄存器如图所示、并在表中进行介绍。
  - 每个寄存器图都显示一个矩形、该矩形被划分为代表寄存器字段的字段。 每个字段都标有其位名称、其起始位和结束位编号、其读/写属性及以下默认复位值。 图例解释了用于属性的符号。
  - 寄存器图中的保留位可以有多种含义之一：
    - 未在器件上实现
    - 保留用于未来的器件扩展
    - 保留用于 TI 测试
    - 不支持的器件保留配置
  - 向保留位写入非默认值可能会导致意外行为、应避免此类行为。

术语表

TI 术语表　　本术语表列出并解释了术语、首字母缩略词和定义。

支持资源

TI E2E™ 支持论坛是工程师的重要参考资料，可直接从专家获得快速、经过验证的解答和设计帮助。搜索现有解答或提出自己的问题可获得所需的快速设计帮助。

链接的内容由各个贡献者“按原样”提供。这些内容并不构成 TI 技术规范，并且不一定反映 TI 的观点；请参阅 TI 的《使用条款》。

商标

This page intentionally left blank.

章节 1
架构

器件架构包括总线结构、平台存储器映射和引导配置。

1.1 架构概述

MSPM0 G 系列 MCU (MSPM0Gxx) 将 32 位计算性能与精密模拟相结合，可支持各种传感、接口、控制和辅助控制应用。该器件架构通过一个灵活且易于配置的电源管理和时钟系统支持高性能应用和低功耗应用。

MSPM0 G 系列器件还通过受 ECC 保护的闪存、奇偶校验保护的 SRAM、可用的双窗口看门狗计时器提供增强的稳健性，并支持 125°C 环境温度和 AEC-Q100 1 级认证。

本章介绍器件架构，包括关于电源域和总线组织、平台存储器映射和器件引导配置的概述。

1.2 总线结构

MSPM0Gxx 器件上有三个主要电源域：

- PD1（电源域 1），包含 CPU 子系统、存储器接口和高速外设
- PD0（电源域 0），包含低速低功耗外设
- 直接从电源为 IO、模拟模块和受限逻辑供电的电源电压 (VDD)

PD1 域支持更高的时钟速度以提高性能，并且在某些工作模式下被禁用，以更大限度地降低功耗。PD0 域支持超低功耗性能，并且始终在内核稳压器运行的工作模式下启用。

MSPM0Gxx 器件上有四条主要数据总线：

- AHB 总线矩阵，用于将 CPU 连接到器件存储器系统（ROM、SRAM 和闪存）和外设总线
- PD1（电源域 1）仅 CPU 外设总线，通过 MCLK 计时
- PD1（电源域 1）外设总线，通过 MCLK 计时
- PD0（电源域 0）外设总线，通过 ULPCLK 计时

CPU 和 DMA 控制器是器件中唯一的两个总线控制器。共享外设的 CPU 和 DMA 之间的仲裁发生在仅 CPU 的 PD1 外设总线和 CPU/DMA PD1 外设总线之间。DMA 无法访问仅 CPU PD1 外设总线或 CPU 总线矩阵（总线图中的绿色元件）上的外设。因此，在 DMA 处理 PD1 或 PD0 总线上的事务的同时，CPU 可以访问仅限 CPU 的 PD1 外设总线上的外设。

同样，只要 DMA 不访问 CPU 试图访问的同一存储器，CPU 就可以在 DMA 正在处理事务的同时通过 AHB 总线矩阵访问 SRAM 或闪存。存储器系统（SRAM 或闪存）的 CPU 和 DMA 之间的仲裁发生在存储器接口本身。CPU 和 DMA 之间的所有仲裁都是循环完成的。

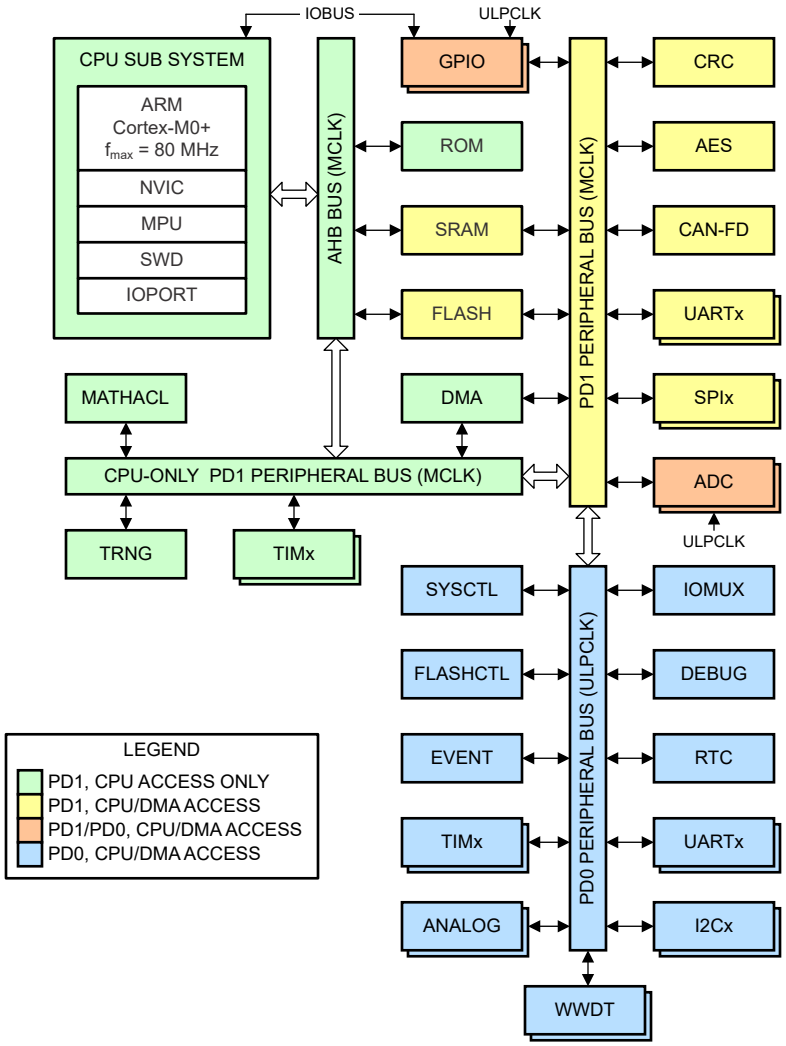


图 1-1. MSPM0Gxx 总线结构

备注

这是 MSPM0Gxx 器件上典型外设及其各自总线位置的通用图。并非所有器件都具有此处显示的所有外设选项。要确定给定器件上可用的外设，请参阅器件特定数据表。

GPIO 和 ADC 外设（总线图中的橙色元件）具有特殊功能，可实现从 CPU 快速访问寄存器以及在低功耗运行模式下运行。

- GPIO 外设通过两种机制连接到系统：PD1 外设总线和 Arm® Cortex®-M0+ 单周期高速 IO 总线。
  - 为了获得出色性能，从 CPU 到任何 GPIO 寄存器的访问通过单周期 IO 总线进行转换，从而在软件控制下实现 IO 的快速切换。
  - PD1 外设总线上也提供 GPIO DOUT 寄存器（数据输出），主要是为了使 DMA 可用于将值加载到 GPIO DOUT 寄存器。
  - 虽然到 GPIO 外设的总线接口在 PD1 电源域中（以获得出色读取/写入性能），但 GPIO 逻辑本身在 PD0 电源域中，因此它在核心稳压器处于运行状态的所有运行模式下都可用。
- ADC 外设通过 PD1 外设总线连接到系统，但在 PD0 电源域中包含功能逻辑。
  - 通过 PD1 外设总线处理 ADC 外设寄存器访问（以获得出色读取/写入性能）
  - ADC 转换逻辑位于 PD0 电源域中，以在禁用 PD1 后在某些低功耗模式下启用运行计时器触发的 ADC 转换，而无需 CPU 交互。

1.3 平台存储器映射

所有 MSPM0Gxx 器件共享一个通用的平台存储器映射。外设会获得一个固定地址空间，并且在系列中的所有器件上都具有相同的地址空间。存储器映射符合标准 Arm Cortex-M 存储器区域的要求。

表 1-1. 顶级存储器映射

存储器区域	起始地址	终止地址	说明
代码	0x0000.0000	0x1FFF.FFFF	闪存和 ROM
SRAM	0x2000.0000	0x3FFF.FFFF	SRAM
外设	0x4000.0000	0x5FFF.FFFF	全局外设存储器映射寄存器和全局不可执行数据存储器
子系统	0x6000.0000	0x7FFF.FFFF	本地 CPU 子系统存储器映射寄存器
系统 PPB	0xE000.0000	0xE00F.FFFF	Arm 专用外设总线

1.3.1 代码区域

代码区域包含用于存储可执行代码和数据的闪存。通过代码区从 CPU 访问闪存通过 AHB 总线矩阵直接处理到闪存读取接口。闪存在代码区域中被混叠到两个地址空间：一个返回 ECC 校正数据，另一个返回未校正 (原始) 数据。有关详细的闪存映射，请参阅节 6.2.3.1。

代码区域还包含用于 TI 器件引导代码和引导加载程序的只读存储器 (ROM)。ROM 仅在初始器件引导过程中可用。

1.3.2 SRAM 区域

SRAM 区域包含系统存储器 (SRAM)。SRAM 支持以最大 MCLK 频率 (80MHz) 访问零等待状态。CPU 对 SRAM 的访问直接通过连接到 SRAM 接口的 AHB 总线矩阵进行处理。SRAM 区域支持具有高达 1MB SRAM 的器件。请参阅器件特定的数据表，了解给定器件上存在的 SRAM 大小。

某些器件可能选择支持 SRAM 的奇偶校验或者奇偶校验与 ECC 校验。请参阅器件特定的数据表以确定器件是否支持具有 ECC 校验或奇偶校验的 SRAM。有关器件如何处理奇偶校验错误和 ECC 错误的信息，请参阅节 2.4.10。

奇偶校验

对于奇偶校验 (如果可用) ，每 8 个数据位提供 1 个奇偶校验位。奇偶校验能够检测相应 8 个数据位中的 single-bit 错误 (SED)。向经过奇偶校验的地址写入数据会根据新数据更新相应的奇偶校验位。从经过奇偶校验的地址读取数据会根据相应的奇偶校验位校验读取的数据。在读取时，如果数据与相应的奇偶校验位不匹配，则会产生一个奇偶校验错误。有关器件如何处理奇偶校验错误的信息，请参阅节 2.4.10。

ECC 校验

对于 ECC 校验 (如果可用) ，每 64 个数据位提供 8 个 ECC 位。ECC 能够纠正 single-bit 错误 (SEC) 并检测相应 64 个数据位中的 dual-bit 错误 (DED)。向经过 ECC 校验的地址写入数据会根据新数据更新相应的 ECC 代码。从经过 ECC 校验的地址读取数据会根据相应的 ECC 代码校验读取的数据。如果发现一个 single-bit 错误，则会自动纠正该错误并生成一个可纠正的 ECC 错误。如果发现 dual-bit 错误，则会生成一个不可纠正的 ECC 错误。

别名子区域

器件上的物理 SRAM 被混叠到整个 SRAM 区域中的多个地址子区域中，如表 1-2 所示。默认和未经过校验的地址子区域都映射到同一物理 SRAM 存储器。每个别名子区域之间的区别在于应用于访问的完整性检查类型。例如，将数据写入地址 0x2000.0000 (默认子区域) 将导致相同的数据出现在地址 0x2020.0000 (未校验的子区域) 中。

默认子区域 (0x2000.0000) 在所有 MSPM0 器件上都可用，并且在使用时提供器件上可用的最高级别的完整性检查。经过奇偶校验的子区域 (0x2010.0000) 在支持 ECC 校验和/或奇偶校验的器件上可用，并且始终使用奇偶校验来处理访问。未校验的子区域 (0x2020.0000) 在所有器件上都可用，并且在使用该子区域时不执行完整性检

查。奇偶校验/ECC 代码子区域 (0x2030.0000) 在具有 ECC 校验或奇偶校验的器件上可用，并且返回与读取的地址相对应的奇偶校验或 ECC 代码。

备注

为了提高稳健性，没有提供任何机制来禁用经过奇偶校验的 SRAM 地址空间的奇偶校验。要在不进行奇偶校验的情况下运行，请根据未校验的 SRAM 地址空间来链接应用程序。

表 1-2. SRAM 区域存储器映射

子区域	启动	结束	说明
默认值	0x2000.0000	0x200F.FFFF	器件上可用的最高完整性检查始终应用于该子区域中的访问： • 如果器件支持 ECC，则会对该子区域进行 ECC 校验。 • 如果器件仅支持奇偶校验（无 ECC），则会对该子区域进行奇偶校验，相关访问等效于对经过奇偶校验的子区域进行访问。 • 如果器件不支持 ECC 或奇偶校验，则不会对该子区域中的访问进行任何校验，该区域等效于未校验的子区域。
经过奇偶校验	0x2010.0000	0x201F.FFFF	如果器件支持奇偶校验，则对该子区域的访问将进行奇偶校验。
未校验	0x2020.0000	0x202F.FFFF	不对该子区域中的访问应用 ECC 校验或奇偶校验。
奇偶校验/ECC 代码	0x2030.0000	0x203F.FFFF	如果器件支持奇偶校验或 ECC，则可以通过该子区域直接访问奇偶校验或 ECC 代码： • 如果器件支持 ECC，则对 64 位边界内任何地址的访问都将返回与 ECC 代码相对应的 8 位（如果应用程序是根据默认区域进行链接的），而如果应用程序是根据经过奇偶校验的区域进行链接的，则返回奇偶校验位（每字节一位）。 • 如果器件仅支持奇偶校验，则在应用程序根据默认区域或经过奇偶校验的区域进行链接的情况下，对 32 位边界内任何地址的访问都将返回 4 个奇偶校验位（每字节一位）。 • 如果器件不支持 ECC 或奇偶校验，则对该区域的访问始终返回零。

备注

对支持 ECC 的器件使用 ECC 校验时，对 SRAM 的写入需要两个周期才能完成。读取访问只需要一个周期，不会产生任何额外的性能损失。

在支持奇偶校验或 ECC 的器件上，应用软件可以将物理 SRAM 的用途划分为用于 ECC 校验、奇偶校验或未校验的任意区域。例如，如果器件具有 32KB 的总 SRAM 存储器，并且支持 ECC 校验和奇偶校验，则可以将应用软件配置为根据两个子区域进行链接：一个子区域用于 ECC 校验，另一个子区域用于奇偶校验。

备注

不建议混合搭配针对相同存储器位置的 ECC 校验访问、奇偶校验访问和未校验访问，因为这样做会导致意外的奇偶校验/ECC 错误。例如，如果通过支持 ECC 的器件上经过奇偶校验的子区域写入 SRAM 存储器位置，然后通过默认（经过 ECC 校验的）子区域读取该位置，则可能会生成 ECC 错误，因为写入数据时会在 ECC/奇偶校验代码存储器中存储奇偶校验信息，而不是存储正确的 ECC 代码。

备注

在上电时或退出 SHUTDOWN 模式时，SRAM 内容可能是随机的。通过一个尚未写入的经过 ECC 校验/奇偶校验的区域读取 SRAM 位置时可能会导致检测到一个 ECC/奇偶校验错误，从而导致处理器出现硬故障。确保在读取之前先初始化所有 SRAM 位置。

1.3.3 外设区域

外设区域包含位于三条外设总线上的存储器映射外设。闪存也在外设区域中有别名。

表 1-3. 外设区域存储器映射

类型	启动	结束	说明
外设	0x4000.0000	0x40FF.FFFF	外设总线上外设的存储器映射寄存器
有别名的闪存	0x4100.0000	0x41FF.FFFF	请参阅 节 6.2.3.1

1.3.4 子系统区域

子系统区域包含特定于 CPU 子系统且无需全局访问的存储器映射寄存器。请参阅 CPU 子系统章节以了解子系统区域中的存储器映射寄存器。

1.3.5 系统 PPB 区域

系统 PPB 区域包含 Arm 专用外设总线上的存储器映射寄存器。这些寄存器与 CPU 紧密耦合，是存储器保护单元 (MPU)、SysTick 计时器以及 CPU 电源管理和复位功能等外设的接口。

1.4 启动配置

在 BOOTRST 之后，器件始终执行启动引导例程，以便在启动主应用程序之前将器件配置为运行。引导例程在主应用程序启动之前从只读存储器 (ROM) 中执行。有两个引导例程：引导配置例程 (BCR) 和引导加载程序 (BSL)。引导配置例程会设置器件安全策略，配置器件使其运行，并视需要启动 BSL。如果 BSL 由 BCR 启动，则可通过使用标准串行接口（UART 或 I2C）来使用 BSL 对器件存储器（闪存和 SRAM）进行编程和/或验证。

在启动例程成功完成执行后，CPU 被复位，并且通过无条件地从闪存的 0x0000.0000 和 0x0000.0004 中提取栈指针 (SP) 和复位矢量来一直启动应用程序。为了启用安全启动，此应用程序代码的单点入口由引导序列强制执行。无法引导至不同的存储器位置。

1.4.1 配置存储器 (NONMAIN)

NONMAIN 是闪存的专用区域，用于存储 BCR 和 BSL 用于引导器件的配置数据。该区域不用于任何其他目的。BCR 和 BSL 都有配置策略，这些策略可以保留为默认值（在开发和评估期间是典型值），也可以通过更改编程到 NONMAIN 闪存区域中的值来针对特定用途进行修改（在生产编程期间是典型值）。

BCR 和 BSL 配置数据结构都包含在 NONMAIN 闪存区域中的单个闪存扇区内。要更改引导配置中的任何参数，有必要擦除整个 NONMAIN 扇区，并使用所需的设置对 BCR 和 BSL 配置结构进行重新编程。

NONMAIN 闪存区域中的配置数据不受批量擦除命令的影响，但经由调试子系统邮箱 (DSSM) 通过 SWD 发送给 BCR 的出厂复位命令将其擦除并重新编程为出厂默认值。

NONMAIN 闪存也可通过使用 UART 或 I2C BSL 接口发送到 BSL 的出厂复位命令进行擦除。但是，与 DSSM 出厂复位不同，BSL 出厂复位不会在擦除之后将 TI 出厂默认值编程为 NONMAIN 存储器。因此，在终止 BSL 会话之前，连接到 MSPM0 目标的主机（通过 BSL 接口）负责使用有效配置重新编程 NONMAIN 存储器。

备注

如果通过 BSL 执行出厂复位命令，并且在 BSL 会话终止之前未将有效的 NONMAIN 配置编程回器件中，器件将假定下一个复位周期中处于最高限制状态，并且无法通过 SWD 或 BSL 访问器件。使用 BSL 出厂复位命令时，请务必确保重新编程有效的 NONMAIN 配置。

表 1-4 中提供了 NONMAIN 数据结构的地址范围。本节末尾提供了 NONMAIN 区域的详细细分。

表 1-4. NONMAIN 区域概述

NONMAIN 部分	起始地址	终止地址
BCR 配置	41C0.0000h	41C0.005Bh
BCR 配置 CRC	41C0.005Ch	41C0.005Fh

表 1-4. NONMAIN 区域概述 (continued)

NONMAIN 部分	起始地址	终止地址
BSL 配置	41C0.0100h	41C0.0153h
BSL 配置 CRC	41C0.0154h	41C0.0157h

1.4.1.1 由 CRC 支持的配置数据

NONMAIN 存储器中的 BCR 配置数据和 BSL 配置数据结构各自包含一个 CRC32 值，该值对应于相应结构的 CRC32 摘要。在器件启动过程中，BCR 将计算数据结构的 CRC 摘要，并将其与存储的 CRC 值进行比较，确认匹配后才会信任并使用这些结构中包含的数据。

BCR 配置 CRC 故障处理

如果 BCR 配置数据（包含 SWD 策略、BSL 启用/禁用策略以及闪存保护和完整性检查策略）在启动期间未能通过 CRC 检查，则会产生灾难性的启动错误并施加以下限制：

- 错误原因将作为引导诊断记录在 CFG-AP 中
- BSL 将不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用
- 如果启用了或使用密码启用了，则会执行待处理的 SWD 恢复出厂设置命令
- 如果已启用，则会执行待处理的 TI 失效分析流程条目
- 启动过程将最多重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

此 CRC 校验的好处是，在启动过程中可以明确地检测配置数据中是否存在任何位翻转，例如静态写保护配置（安全启动的支柱）。故障处理程序会明确阻止 BSL 和用户应用程序运行，唯一受支持的选项（SWD 恢复出厂设置和 TI FA）受 16 位模式匹配字段保护。

BSL 配置 CRC 故障处理

如果 BSL 配置数据（包含 BSL 密码和 BSL 策略）在 BSL 调用期间未通过 CRC 检查，则会导致灾难性的启动错误并施加以下限制：

- 错误原因作为引导诊断记录在 CFG-AP 中
- BSL 不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用
- 启动过程最多会重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

此 CRC 校验的好处是，在调用过程中可以明确地检测 BSL 配置数据中是否存在任何位翻转。故障处理程序会阻止 BSL 使用可能导致安全性丧失的无效数据启动。

TI 出厂修整数据 CRC 故障处理

除了用户指定的配置数据外，如果 TI 出厂修整在启动期间未能通过 CRC 检查，也会导致灾难性的启动错误并具有以下限制：

- 错误原因将作为引导诊断记录在 CFG-AP 中
- BSL 将不会被调用，即使它配置为要启用
- 用户应用程序不会启动
- 应用程序调试访问不会启用

- 如果已启用，则会执行待处理的 TI 失效分析流程条目
- 启动过程将最多重试 3 次
  - 如果第 2 次或第 3 次尝试通过，器件将正常启动
  - 如果第 3 次尝试仍未通过，则在下一次 BOR 或 POR 之前不会再进行启动尝试

1.4.1.2 16 位关键字段模式匹配

BCR 配置存储器中的关键策略（如 SWD 安全策略）会在 NONMAIN 存储器中实现为 16 位模式匹配字段，并具有以下特性：

- 需要精确的模式匹配，才能启用较低的安全状态
- 如果 16 位字段中的任何值与确切定义的模式不匹配，都会导致相应参数处于最高安全状态

这种行为可防止 single-bit 翻转导致器件进入比最初指定更低的安全状态。

1.4.2 引导配置例程 (BCR)

引导配置例程是 BOOTRST 之后在器件上运行的第一个固件。BCR 在启动时管理以下内容：

- 配置调试接口安全策略
- 视需要执行批量擦除
- 视需要执行恢复出厂设置
- 配置闪存静态写入保护策略
- 视需要验证部分或整个应用固件的完整性（使用 32 位 CRC）
- 视需要启动引导加载程序 (BSL)

1.4.2.1 串行线调试相关策略

串行线调试相关策略配置可通过器件的物理调试接口 (SWD) 获得的功能。默认情况下，TI 的 MSPM0 器件处于不受限制的状态。该状态支持轻松地进行生产编程、评估和开发。但是，不建议在大规模生产中使用这种不受限制的状态，因为它会留下很大的攻击面。为了在保持配置过程简单的同时满足各种需求，MSPM0 器件支持三个通用安全级别：无限制（级别 0）、自定义限制（级别 1）和完全限制（级别 2）。表 1-5 显示了三种通用安全级别，从限制性最低到限制性最高。

SWD 接口有 4 种主要用途需要考虑保护：

- 应用调试访问，其中包括：
  - 通过 AHB-AP 对处理器、存储器映射和外设进行完全访问
  - 通过 ET-AP 访问器件 EnergyTrace+ 状态信息
  - 通过 PWR-AP 访问器件电源状态控制以进行调试
- 批量擦除访问，其中包括：
  - 能够通过 SWD 发送一条命令来擦除 MAIN 内存区域，同时保持 NONMAIN 器件配置存储器不变
- 恢复出厂设置访问，其中包括：
  - 能够通过 SWD 发送命令来擦除 MAIN 存储器区域并将 NONMAIN 器件配置存储器恢复到 TI 出厂默认设置（0 级）
- TI 失效分析访问，其中包括：
  - TI 能够通过 SWD 启动失效分析回流（请注意，在向 TI 提供 FA 访问权限之前，TI FA 流始终强制恢复出厂设置；这确保在失效分析流程启动时，TI 没有任何机制来读取存储在器件闪存中的专有客户信息）

表 1-5. 通用安全级别

Level	场景	SW-DP 策略	应用调试策略	批量擦除策略	恢复出厂设置策略	TI FA 策略
0	无限制	EN	EN	EN	EN	EN
1	自定义限制	EN	EN、EN（具有密码）、DIS	EN、EN（具有密码）、DIS	EN、EN（具有密码）、DIS	EN、DIS
2	完全限制	DIS	无关（禁用 SW-DP 时无法访问）(1)			

(1)当 SW-DP 策略是**禁用 SW-DP** 时，从 SWD 接口的角度来看，批量擦除和恢复出厂设置策略是无关的。但是，如果启用了引导加载程序 (BSL)，则批量擦除和恢复出厂设置策略会影响通过 BSL 提供的功能。有关保护 BSL 的详细信息，请参阅 BSL 安全性部分。

1.4.2.1.1 SWD 安全级别 0

SWD 安全级别 0 是限制性最低的 SWD 安全状态。这是 TI 新器件的默认状态，也是器件在成功恢复出厂设置后的状态。此状态下对应用调试访问、整体擦除、恢复出厂设置和失效分析没有限制。

何时使用此状态

级别 0 非常适合原型设计和开发，因为它允许对器件存储器进行编程以及对处理器和外设进行调试。

何时不应使用此状态

大规模生产中不应使用级别 0。攻击者可以完全自由地读取器件存储器的内容、操纵器件的执行，并或许能更改闪存的内容（取决于闪存写保护方案）。

1.4.2.1.2 SWD 安全级别 1

SWD 安全级别 1 允许自定义安全配置。物理调试端口 (SW-DP) 保持启用状态，并且每个功能（应用调试、批量擦除命令、恢复出厂设置命令和 TI 失效分析）都可以单独启用、禁用或（在某些情况下）通过密码身份验证启用，从而提供相当大的灵活性来根据特定用例定制器件行为。

何时使用此状态

级别 1 非常适合受限的原型设计/开发场景以及大规模生产场景，在这些场景中，需要保留某些 SWD 功能（例如恢复出厂设置和 TI 失效分析），同时禁用其他功能（例如应用调试）。表 1-6 中给出了级别 1 自定义配置的常见示例。

表 1-6. 级别 1 配置示例

级别 1 场景	配置			
	应用调试	批量擦除	恢复出厂设置	TI FA
这种场景使用用户指定的密码限制了调试访问，但保留了恢复出厂设置和 TI 失效分析。此配置允许进行现场调试（使用密码），另外还允许通过恢复出厂设置来将器件恢复到默认的“级别 0”状态。	EN（具有密码）	DIS	EN	EN
此场景不允许进行调试。它允许恢复出厂设置，但只能通过用户指定的密码来使用。这提供了一种在现场访问器件的方法，方法是在密码已知时清除 MAIN 存储器内容并将器件恢复到“级别 0”状态。重要的是，即使恢复出厂设置密码被泄露，攻击者也无法读取 MAIN 闪存中的专有信息。	DIS	DIS	EN（具有密码）	EN
此场景不允许调试，也不允许 TI 失效分析。这可防止 TI 在器件上执行恢复出厂设置和进一步的 FA 活动，除非用户在将器件返回 TI 进行 FA 之前使用自己指定的密码执行恢复出厂设置。	DIS	DIS	EN（具有密码）	DIS

备注

对于大多数标准生产用例，建议使用级别 1 配置。对于不需要安全启动的应用，TI 建议在生产中使用级别 1，同时保持启用恢复出厂设置（使用密码）和 TI 失效分析。在此类配置中，用户（使用密码）或 TI（通过失效分析返回流程）或许能在器件配置后将器件恢复到限制性较低的状态。在需要最大安全启动保证的用例中，可以使用限制性较高的级别 1 或级别 2 进行生产，但需要权衡的是，器件在配置后可能无法恢复到限制性较低的状态。

何时不应使用此状态

如果需要对器件进行完全访问，则不应在原型设计期间使用级别 1；在这种情况下，应使用级别 0。

级别 1 也不应用于需要最高限制状态且不启用 SWD 功能的大规模生产场景；在这种情况下，应改为使用级别 2，因为它直接禁用整个 SWD 物理接口，并更大限度地减少误配置的可能性。

备注

如果器件配置为禁用应用调试和恢复出厂设置，则用户要恢复对器件的调试访问，唯一方法是用户应用代码提供了一种机制，可将 NONMAIN 配置更改为限制性较低的状态。如果 NONMAIN 通过静态写保护锁定，则状态不可逆，用户无法重新获得调试访问。

1.4.2.1.3 SWD 安全级别 2

SWD 安全级别 2 会将器件配置为最高限制状态。物理调试端口 (SW-DP) 被完全禁用，所有 SWD 可访问的功能（应用调试、批量擦除、恢复出厂设置和 TI 失效分析）都不能通过 SWD 访问，无论它们的配置如何。

当选择级别 2（SW-DP 禁用）时，应用调试配置和 TI 失效分析配置字段都是无关字段，不会影响器件配置。

如果 BSL 被禁用，则批量擦除和恢复出厂设置配置字段也是无关字段。但是，如果 BSL 被启用，那么 BSL 仍然使用批量擦除和恢复出厂设置配置字段来授权来自 BSL 接口的批量擦除或恢复出厂设置命令。

何时使用此状态

级别 2 只应用于大规模生产，那时无需进一步访问任何 SWD 功能且器件需要达到最大安全状态。

何时不应使用此状态

以下情况下不应使用级别 2：

- 未来可能需要通过 SWD 进行应用调试和/或重新编程
- 用户希望 TI 能够对器件执行失效分析
- 用户希望能够通过 SWD 发送批量擦除或恢复出厂设置命令来从闪存中删除专有信息

备注

器件配置为级别 2（SW-DP 禁用）后，就**无法**通过 SWD 进一步访问器件。要将器件恢复到级别 0 或级别 1 状态并恢复 SWD 访问，唯一方法是启用 BSL 和恢复出厂设置（允许发送 BSL 恢复出厂设置命令），或者用户应用程序代码中包含一种机制，可将 NONMAIN 配置更改为限制性较低的状态。在任何一种情况下，如果 NONMAIN 通过静态写保护锁定，则级别 2 状态不可逆，法重新获得 SWD 访问。

1.4.2.2 SWD 批量擦除和恢复出厂设置命令

BCR 借助于使用调试子系统邮箱 (DSSM) 从调试探针中通过 SWD 发送到器件的命令，提供批量擦除和恢复出厂设置功能。这些命令在 SWD 安全级别 2 中不可用，但它们在安全级别 0 和 1 中可供选用。当器件未配置为 SWD 安全级别 2 时，可以单独将批量擦除和出厂重置命令配置为启用、使用唯一的 128 位密码启用，或者禁用。默认情况下启用这两个命令。

SWD 批量擦除和恢复出厂设置 DSSM 命令取代了任何静态写保护策略。例如，如果 SWD 恢复出厂设置配置为启用或使用密码启用，则可以复位 BCR 配置数据，即使数据为静态受写保护也是如此。

SWD 批量擦除

SWD 批量擦除是指仅擦除 MAIN 闪存区域，通常包括用户应用程序。存储在 NONMAIN 闪存区域中的 BCR 和 BSL 策略不受批量擦除的影响。批量擦除适用于在保持器件配置本身不变的情况下擦除所有应用程序代码和数据。

要设置批量擦除命令模式和密码，请配置 NONMAIN 存储器中的 BOOTCFG3.MASSERASECMDACCESS 字段和 PWDMASSERASE 密码字段。

SWD 恢复出厂设置

SWD 恢复出厂设置是指擦除 MAIN 闪存区域，然后将非 NONMAIN 闪存区域复位为默认值。这种擦除对于完全复位 BCR 和 BSL 器件启动策略非常有用，同时还擦除应用程序代码和数据。

要设置恢复出厂设置命令模式和密码，请配置 NONMAIN 存储器中的 BOOTCFG3.FACTORYRESETCMDACCESS 字段和 PWDFACTORYRESET 密码字段。

1.4.2.3 闪存保护和完整性相关策略

闪存保护和完整性策略规定闪存的哪些扇区被锁定而无法修改，以及启动过程中在启动用户应用程序之前要检查哪些扇区的完整性。

1.4.2.3.1 锁定应用 (MAIN) 闪存

MSPM0 MCU 实现了一个静态写保护方案，以将 MAIN 闪存区域中用户定义的扇区锁定，从而防止在运行时针对相应扇区执行任何编程或擦除操作。所需的静态写保护方案配置为 NONMAIN 闪存区域中启动安全策略的一部分。

用途

静态写保护方案支持在闪存中存储一个用户定义的、具有以下特性的固定应用程序：

- 在编程结束并被锁定后，该应用程序就无法由应用程序代码或 ROM 引导加载程序进行修改
- 如果置于闪存的开头，该应用程序始终是 ROM 引导配置例程转换到执行用户应用程序时执行的第一个代码

MSPM0 静态写保护支持这两个特性，要实现安全启动映像管理器，必须满足这些特性。

功能

当引导配置例程将转换到执行 MAIN 闪存中的引导加载程序或用户应用程序代码时，在 NONMAIN 中配置为写入锁定的任何扇区都在功能上不可更改。如果应用程序代码或引导加载程序尝试对受静态保护的扇区进行任何编程或擦除，都会导致硬件闪存操作错误，并且扇区不会被修改。

静态写保护可防止应用程序代码或引导加载程序进行任何修改，但通过 SWD 接口发送的批量擦除或恢复出厂设置命令将被接受。如果不需要这种行为，可以使用唯一的密码来保护批量擦除或恢复出厂设置 SWD 命令，也可以禁用这两个命令（请参阅 SWD 策略）。要完全消除任何修改受静态写保护的 MAIN 闪存扇区的方法，必须禁用批量擦除和恢复出厂设置命令（或 SW-DP），并且 NONMAIN 引导配置存储器也必须具有静态写保护，以防止应用程序代码通过修改 NONMAIN 区域内容来更改底层写保护方案。下一节会对此进行介绍。

1.4.2.3.2 锁定配置 (NONMAIN) 闪存

MSPM0 MCU 实现了一个静态写保护机制，以在运行时锁定 NONMAIN 闪存区域，从而防止对该区域进行任何编程/擦除操作。写保护方案配置为 NONMAIN 闪存区域中启动安全策略的一部分。

用途

默认情况下，TI 的 NONMAIN 配置存储器（包含用户指定的启动安全策略和引导加载程序策略）不受写保护。这样一来，用户就可以在配置期间擦除 NONMAIN，并使用将用于大规模生产的用户指定策略重新编程。

在许多情况下，配置存储器最好在配置完毕后锁定。锁定配置存储器的好处是可以防止引导加载程序或应用程序代码本身对安全策略、引导加载程序策略和静态写保护策略进行任何未经授权的修改。在大多数应用中，大规模生产的器件无需修改配置存储器，即使在器件固件更新时也是如此。

功能

当配置为受保护时，整个 NONMAIN 区域都将被写锁定，并且在引导配置例程将执行传递给引导加载程序或 MAIN 闪存中的用户应用程序代码时在功能上不可更改。如果应用程序代码或引导加载程序尝试对 NONMAIN 进行任何编程或擦除，都会导致硬件闪存操作错误，并且扇区不会被修改。

静态写保护可防止应用程序代码或引导加载程序进行任何修改，但通过 SWD 接口发送的恢复出厂设置命令仍会被接受。如果不需要这种行为，可以使用唯一的密码来保护恢复出厂设置 SWD 命令，或者完全禁用该命令（请参阅 SWD 策略）。要完全消除任何修改 NONMAIN 配置存储器的方法，必须禁用恢复出厂设置命令和 TI FA（或 SW-DP）。

备注

当 NONMAIN 受到静态写保护并且禁用了恢复出厂设置命令和 TI FA（或 SW-DP）时，NONMAIN 相当于不可改变的只读存储器，并且不再能够通过任何方式更改器件配置。此外，如果任何 MAIN 存储器区域扇区配置了静态保护，则这些扇区也不能通过任何方式进行修改，可能会被视为不可更改。

1.4.2.3.3 静态写保护 NONMAIN 字段

对于 MAIN 闪存的前 32 个扇区，可以按扇区启用写保护。对于闪存的其余扇区，可以按每 8 个扇区启用写保护。要设置静态写保护策略，请配置 NONMAIN 存储器中的 FLASHSWP0 和 FLASHSWP1 字段。

备注

通过调试子系统邮箱 (DSSM) 发送到 BCR 的批量擦除和恢复出厂设置命令将覆盖指定的静态写保护策略。如果不需要此行为，请将批量擦除和恢复出厂设置命令配置为使用密码启用或者禁用。请注意，发送到 BSL 的批量擦除和恢复出厂设置命令将遵守指定的静态写保护策略（BSL 与应用程序代码具有相同的权限）。

1.4.2.4 应用程序 CRC 验证

BCR 支持在启动用户应用之前的引导过程中对包含在 MAIN 闪存区域中的应用代码和数据执行完整的 CRC32 完整性检查。这对于在执行部分或全部应用程序代码和数据之前确保这些代码和数据的完整性非常有用。

要在启动时启用 CRC32 完整性检查，必须将以下信息编程到 NONMAIN 闪存中的 BCR 配置中：

- CRC 校验的 32 位起始地址（APPCRCSTART.ADDRESS 字段）
- 应用 CRC 校验的应用程序长度（以字节为单位）（APPCRCLEN.LENGTH 字段）
- 要据以测试的预先计算的 32 位 CRC 值（APPCRC.DIGEST 字段）
- CRC 校验的已启用密钥值（BOOTCFG3.APPCRCMODE 字段）

如果应用程序 CRC 校验在引导时失败，则不会启动 MAIN 闪存中的应用程序。如果启用了引导加载程序，则进入它。如果未启用 BSL，引导会失败。

1.4.2.5 快速引导

通过启用快速引导模式，可以缩短 BCR 的执行时间。启用快速引导模式后，可使用以下方法加快引导过程：

- 限制 BSL 进入方法。启用快速引导模式后，只能使用 SYSCTL 寄存器调用方法和 DSSM 调用方法进入引导加载程序。未测试其他 BSL 调用条件（例如，基于引脚的调用）。
- 绕过应用程序 CRC 校验（即使启用了应用程序 CRC 校验）。

要启用快速引导模式，请在 NONMAIN 闪存的 BOOTCFG2.FASTBOOTMODE 字段中将快速引导模式设置为启用。

1.4.2.6 引导加载程序 (BSL) 启用/禁用策略

与串行线调试接口相反，引导加载程序 (BSL) 提供了一种通过标准串行接口（UART 或 I2C）对器件存储器进行编程和验证的方法。BSL 具有自己的配置策略，但由 BCR 确定是否启用了 BSL 以进行调用，还是要禁用 BSL（不可调用）。

由于 BSL 提供了一个额外的攻击面，如果它未在应用程序中使用，则可以在用户指定的启动安全策略中禁用它。如果在应用程序中使用 BSL，则在 BSL 配置策略中管理 BSL 安全设置（包括 BSL 访问密码）。

1.4.2.6.1 BSL 启用

通过在 NONMAIN 闪存的 BOOTCFG2.BSLMODE 字段中将引导加载程序模式设置为启用或禁用，可以启用或禁用引导加载程序 (BSL)。禁用 BSL 后，无法通过任何调用机制进入引导加载程序。

1.4.3 引导加载程序 (BSL)

引导加载程序 (BSL) 提供了一种通过标准 UART 或 I2C 串行接口对器件存储器进行编程和/或验证的方法。可通过串行接口访问的 BSL 主要特性包括：

- 闪存的编程和擦除
- 能够通过指向主闪存的指针返回固件版本号
- 指定硬件调用 GPIO 的能力
- 能够启用代码/数据读取（默认禁用）
- 能够返回代码/数据区域的 32 位 CRC（最小区域大小为 1KB）以验证编程
- 访问始终受到 256 位密码的保护
- 可配置的安全警报处理，用于抵抗蛮力攻击
- 支持 MAIN 闪存插件，以支持 UART 和 I2C 之外的其他接口

有关 BSL 功能的完整说明，请参阅 BSL 用户指南。

如果需要，可以通过在 NONMAIN 闪存的 BCR 配置中正确配置 BSL 模式来完全禁用 BSL。有关启用或禁用 BSL 的详细信息，请参阅 BSL 启用部分。

1.4.3.1 GPIO 调用

引导加载程序支持在 BOOTRST 之后通过使用 GPIO 进行硬件调用。NONMAIN 闪存中的 BSL 配置包含 GPIO 调用的焊盘、引脚和极性定义。器件由 TI 针对特定的 GPIO 和极性进行配置，但软件可以通过修改 NONMAIN 闪存中 BSL 配置中的 GPIO 引脚配置来更改此默认设置。

要指定 BSL_invoke 引脚的极性，请配置 NONMAIN 存储器中的 BSLCONFIG0.BSLIVK_LVL 字段。

要指定用于 BSL_invoke 的器件引脚，请在 NONMAIN 闪存中配置以下字段：

- 将 IOMUX PINCMx 索引存储到 BSLCONFIG0.BSLIVK_PAD_NUM 字段中
- 将 GPIO 端口（A 或 B）存储到 BSLCONFIG0.BSLIVK_GPIOPORT 字段中
- 将 GPIO 引脚（0 至 31）存储到 BSLCONFIG0.BSLIVK_GPIOPIN 字段中

请参阅特定于器件的数据表以确定默认的 BSL 调用 GPIO。

1.4.3.2 引导加载程序 (BSL) 安全策略

BSL 安全策略在调用时由引导加载程序解释，并包含以下参数：

- BSL 访问密码，如 节 1.4.3.2.1 所述
- BSL 读取策略，如 节 1.4.3.2.2 所述
- BSL 安全警报策略（篡改检测），如 节 1.4.3.2.3 所述

1.4.3.2.1 BSL 访问密码

对 BSL 的访问始终受到用户指定的 256 位密码保护。无法选择禁用密码。必须在调用后向 BSL 提供密码，才能获得权限来访问大多数的 BSL 功能。如果未提供密码，则允许的 BSL 命令只有*获取身份*和*启动应用程序*。

如果向 BSL 提供了错误的密码，BSL 会暂停 2 秒，随后可以再尝试发送正确的密码。在三次密码尝试失败后，安全警报功能将被激活（请参阅 节 1.4.3.2.3）。

1.4.3.2.2 BSL 读取策略

BSL 可选择支持出于调试和/或诊断目的读取器件存储器（在通过正确密码匹配获得 BSL 访问权限后）。默认情况下，为了安全起见，会禁用此功能，以防止他人从器件中提取敏感代码和/或数据。禁用 BSL 读取策略后，可通过 BSL 接口向主机提供的唯一信息是最小段长度为 1KB 的存储器段 CRC32 摘要。如果需要直接读取器件存储器，可在 BSL 配置中启用该功能。

1.4.3.2.3 BSL 安全警报策略

BSL 提供了一种警报机制，用于在怀疑发生篡改时采取措施。具体而言，如果在一个 BSL 会话期间 3 次将错误的密码传递给 BSL，则会激活安全警报，并且 BSL 可能会根据指定的安全警报策略以三种不同的方式之一进行响应：

1. 发出恢复出厂设置命令（擦除 MAIN 闪存并重置 NONMAIN 闪存区域）。
2. 禁用 BSL（保持 MAIN 闪存不变，但重新配置 NONMAIN 以阻止访问 BSL）。
3. 忽略（不修改配置并允许继续密码尝试）。

备注

选项 1 和 2 要求 NONMAIN 闪存区域未受静态写保护（请参阅 节 1.4.2.3.2）。

选择选项 1 时，配置为受静态写保护的任何 MAIN 存储器区域（请参阅 节 1.4.2.3.1）将不会在恢复出厂设置期间被擦除。

1.4.3.3 应用版本

BSL 支持通过 BSL 串行接口返回应用版本号。这使得 BSL 主机能够在不能读取固件的情况下查询固件版本。版本字段的长度为 32 位。要将应用版本命令链接到主闪存中编程的版本号，请在 NONMAIN 闪存中的 BSLAPPVER.ADDRESS 字段中对版本号的地址进行编程。只有当 BSLAPPVER.ADDRESS 中指定的地址对应于有效的闪存地址时，才会返回版本数据。

1.4.3.4 BSL 触发的批量擦除和恢复出厂设置

可以向 BSL 发送批量擦除或恢复出厂设置命令。这些命令的工作方式与 SWD 批量擦除和恢复出厂设置命令类似，但有几个关键字例外。

BSL 批量擦除

发送到 BSL 的批量擦除命令将擦除 MAIN 闪存。任何配置为静态写保护（通过 NONMAIN 配置存储器中的 FLASHSWP0 和 FLASHSWP1 字段）的 MAIN 闪存扇区都不会被擦除。NONMAIN 器件配置存储器不会被批量擦除所擦除。

BSL 恢复出厂设置

发送到 BSL 的恢复出厂设置命令将首先执行 BSL 批量擦除以擦除主闪存（不包括任何受静态写保护的扇区）。然后，它将额外擦除 NONMAIN 器件配置存储器。

仅当满足以下条件时，才接受 BSL 恢复出厂设置命令：

1. NONMAIN 存储器本身当前未配置为受静态写保护（NONMAIN 中的 BOOTCFG4.NONMAINSWP 字段设置为不受保护）
2. 恢复出厂设置命令未配置为禁用（NONMAIN 中的 BOOTCFG3.FACTORYRESETCMDACCESS 字段未设置为禁用）

在终止 BSL 会话之前，BSL 主机必须（通过 BSL 命令）将有效配置数据编程回 NONMAIN，否则器件可能进入不可恢复的状态。

备注

如果在 BSL 恢复出厂设置后未对 NONMAIN 进行编程，器件将假定下一个复位周期中处于最高限制状态，MAIN 闪存中的任何应用程序代码将不会启动，并且无法通过任何方式访问器件。为了防止锁定，请务必确保在 BSL 恢复出厂设置后将有效的配置数据编程到 NONMAIN 中。

1.5 NONMAIN 寄存器

表 1-7 列出了 NONMAIN 寄存器的存储器映射寄存器。表 1-7 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 1-7. NONMAIN 寄存器

偏移	缩写	寄存器名称	组	部分
41C00000h	BCRCONFIGID	BCR 结构的配置 ID		转到
41C00004h	BOOTCFG0	串行线调试 (SWD) 锁定策略。		转到
41C00008h	BOOTCFG1	BSL 调用引脚策略。		转到
41C0000Ch + 公式	PWDDEBUGLOCK[y]	SWD 命令和密码验证请求。		转到
41C00010h	BOOTCFG2	快速引导模式策略和 BSL 模式策略。		转到
41C00020h	BOOTCFG3	批量擦除和出厂复位模式策略。 这些策略会影响 SWD 启动和 BSL 启动的批量擦除和恢复出厂设置命令。 如果 SW-DP 被禁用（SWDP_MODE 被禁用），则不允许 SWD 启动的命令，因为 SW-DP 被完全禁用。 如果 BSL 被禁用（BSLMODE 被禁用），则这些设置对于 BSL 启动的命令是无关紧要的，因为 BSL 不允许被调用。		转到
41C00024h + 公式	PWDMASSERASE[y]	SWD 批量擦除命令密码（必须通过 DSSM 提供，以便验证批量擦除命令）。		转到
41C00034h + 公式	PWDFACTORYRESET[y]	SWD 恢复出厂设置命令密码（必须通过 DSSM 提供，以便验证恢复出厂设置命令）。		转到
41C00040h	FLASHSWP0	前 32kB 闪存的静态写保护策略。 受保护时，扇区将无法通过引导加载程序或应用程序代码进行编程或擦除。		转到
41C00048h	FLASHSWP1	闪存其他扇区的静态写保护策略。 受保护时，扇区将无法通过引导加载程序或应用程序代码进行编程或擦除。		转到
41C00050h	BOOTCFG4			转到
41C00060h	APPCRCSTART	应用 CRC 校验的起始地址（必须是 MAIN 闪存区域中的地址）。		转到
41C00064h	APPCRCLENGTH	要包括在应用 CRC 校验中的应用区域的长度（以字节为单位），从 APPCRCSTART 开始。		转到
41C00068h	APPCRC	引导期间要测试的预期应用 CRC 校验摘要 (CRC-32)。		转到
41C0006Ch	BOOTCRC	NONMAIN 存储器 BCR（引导配置）部分的 CRC 摘要 (CRC-32)。		转到
41C00080h	BSLCONFIGID	BSL 配置 ID。		转到
41C00084h	BSLPINCFG0	BSL UART 引脚配置。		转到
41C00088h	BSLPINCFG1	BSL I2C 引脚配置。		转到

表 1-7. NONMAIN 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
41C0010Ch	BSLCONFIG0	BSL 调用引脚配置和存储器读出策略。		转到
41C0011 0h + 公式	BSLPW[y]	256 位 BSL 访问密码。		转到
41C00130h	BSLPLUGINCFG	定义 MAIN 闪存中是否存在 BSL 插件及其类型。		转到
41C0013 4h + 公式	BSLPLUGINHOOK[y]	用于插件初始化、接收、发送和取消初始化函数的函数指针。		转到
41C00144h	PATCHHOOKID	备用 BSL 配置。		转到
41C00148h	SBLADDRESS	备用 BSL 的地址。		转到
41C0014Ch	BSLAPPVER	应用版本字的地址。		转到
41C00150h	BSLCONFIG1	BSL 安全配置。		转到
41C00154h	BSLCRC	NONMAIN 存储器 BSL_CONFIG 部分的 CRC 摘要 (CRC-32)。		转到

复杂的位访问类型经过编码可适应小型表单元。表 1-8 展示了适用于此部分中访问类型的代码。

表 1-8. NONMAIN 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值
寄存器数组变量		
i、j、k、l、m、n		当这些变量用于寄存器名称、偏移或地址时，它们指的是寄存器数组的值，其中寄存器是一组重复寄存器的一部分。寄存器组构成分层结构，数组用公式表示。
y		当该变量用于寄存器名称、偏移或地址时，它指的是寄存器数组的值。

1.5.1 BCRCONFIGID (偏移 = 41C00000h) [复位 = 00000001h]

图 1-2 展示了 BCRCONFIGID，表 1-9 中对此进行了介绍。

返回到汇总表。

BCR 结构的配置 ID

图 1-2. BCRCONFIGID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFIG																															
R/W-00000001h																															

表 1-9. BCRCONFIGID 字段说明

位	字段	类型	复位	说明
31-0	CONFIG	R/W	00000001h	BOOTCFG 的配置 ID

1.5.2 BOOTCFG0（偏移 = 41C00004h）[复位 = AABBAABBh]

图 1-3 展示了 BOOTCFG0，表 1-10 中对此进行了介绍。

返回到汇总表。

串行线调试 (SWD) 锁定策略。

图 1-3. BOOTCFG0

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

表 1-10. BOOTCFG0 字段说明

位	字段	类型	复位	说明
31-16	SWDP_MODE	R/W	AABBh	串行线调试端口 (SW-DP) 访问策略。该策略设置是否允许通过 SWD 引脚（连接到任何 DAP）与器件进行任何通信。禁用时，无论 DEBUGACCESS 字段的配置如何，都无法进行 SWD 通信。 5566h = SW-DP 已完全禁用，无法通过 SW-DP 访问器件（0x5566 和所有其他非 0xAABB 值）。 AABBh = SW-DP 已启用，器件访问由 NONMAIN 中的附加策略设置。
15-0	DEBUGACCESS	W	AABBh	用于访问 AHB-AP、ET-AP 和 PWR-AP 调试访问端口的调试访问策略。请注意，如果 SWDP_MODE 设置为 DISABLED，则会忽略该字段的值，调试端口将保持完全锁定。 5566h = 禁用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP（0x5566 和所有其他非 0xCCDD 或 0xAABB 值）。 AABBh = 启用通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP。 CCDDh = 只有在执行 BCR 之前通过 DSSM 提供正确密码时，才能通过 SWD 访问 AHB-AP、ET-AP 和 PWR-AP。

1.5.3 BOOTCFG1（偏移 = 41C00008h）[复位 = AABBAABBh]

图 1-4 展示了 BOOTCFG1，表 1-11 中对此进行了介绍。

返回到汇总表。

BSL 调用引脚策略。

图 1-4. BOOTCFG1

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

表 1-11. BOOTCFG1 字段说明

位	字段	类型	复位	说明
31-16	BSL_PIN_INVOKE	R/W	AABBh	引导加载程序 (BSL) 引脚调用方法启用/禁用策略。 5566h = 引导期间不检查 BSL_INVOKE 引脚（0x5566 和所有其他非 0xAABB 值）。 AABBh = 在引导期间检查 BSL_INVOKE 引脚。
15-0	TI_FA_MODE	R/W	AABBh	设置 TI 失效分析启用/禁用策略。如果启用，则允许通过 DSSM 重新测试请求，否则不允许。请注意，如果 SWDP_MODE 设置为禁用，则会忽略该字段并且无法进行失效分析。 5566h = 不允许进行 TI 失效分析（0x5566 和所有其他非 0xAABB 值）。 AABBh = 允许进行 TI 失效分析。

1.5.4 PWDDEBUGLOCK[y]（偏移 = 41C0000Ch + 公式）[复位 = FFFFFFFFh]

图 1-5 展示了 PWDDEBUGLOCK[y]，表 1-12 中对此进行了介绍。

返回到汇总表。

SWD 命令和密码验证请求。

偏移 = 41C0000Ch + (y * 4h)；其中 y = 0h 至 3h

图 1-5. PWDDEBUGLOCK[y]

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

表 1-12. PWDDEBUGLOCK[y] 字段说明

位	字段	类型	复位	说明
31-0	PW	R/W	FFFFFFFFh	密码

1.5.5 BOOTCFG2（偏移 = 41C0001Ch）[复位 = AABBFFFFh]

图 1-6 展示了 BOOTCFG2，表 1-13 中对此进行了介绍。

返回到汇总表。

快速引导模式策略和 BSL 模式策略。

图 1-6. BOOTCFG2

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

表 1-13. BOOTCFG2 字段说明

位	字段	类型	复位	说明
31-16	BSLMODE		AABBh	BSLMODE 配置引导加载程序启用/禁用策略。 5566h = BSL 被禁用（0x5566 和所有其他非 0xAABB 值）。 AABBh = 启用 BSL。
15-0	FASTBOOTMODE		FFFFh	FASTBOOTMODE 配置快速引导模式启用/禁用策略。 5566h = 禁用快速引导模式。会评估所有启用的 BSL 调用条件（0x5566 和所有其他非 0xAABB 值）。 AABBh = 启用快速引导模式。仅评估软件 BSL 调用条件。

1.5.6 BOOTCFG3（偏移 = 41C00020h）[复位 = AABBAABBh]

图 1-7 展示了 BOOTCFG3，表 1-14 中对此进行了介绍。

返回到汇总表。

批量擦除和出厂复位模式策略。这些策略会影响 SWD 启动和 BSL 启动的批量擦除和恢复出厂设置命令。如果 SW-DP 被禁用（SWDP_MODE 被禁用），则不允许 SWD 启动的命令，因为 SW-DP 被完全禁用。如果 BSL 被禁用（BSLMODE 被禁用），则这些设置对于 BSL 启动的命令是无关紧要的，因为 BSL 不允许被调用。

图 1-7. BOOTCFG3

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

表 1-14. BOOTCFG3 字段说明

位	字段	类型	复位	说明
31-16	FACTORYRESETCMDACCESS	R/W	AABBh	恢复出厂设置命令策略。 5566h = 不允许恢复出厂设置命令（0x5566 和所有其他非 0xAABB 或 0xCCDD 值）。 AABBh = 允许恢复出厂设置命令。 CCDDh = 仅当通过 DSSM 提供匹配密码时才允许恢复出厂设置命令。
15-0	MASSERASECMDACCESS	R/W	AABBh	批量擦除命令策略。 5566h = 不允许批量擦除命令（0x5566 和所有其他非 0xAABB 或 0xCCDD 值）。 AABBh = 允许批量擦除命令。 CCDDh = 仅当通过 DSSM 提供匹配密码时才允许批量擦除命令。

1.5.7 PWDMASSERASE[y]（偏移 = 41C00024h + 公式）[复位 = FFFFFFFFh]

图 1-8 展示了 PWDMASSERASE[y]，表 1-15 中对此进行了介绍。

返回到汇总表。

SWD 批量擦除命令密码（必须通过 DSSM 提供，以便验证批量擦除命令）。

偏移 = 41C00024h + (y * 4h)；其中 y = 0h 至 3h

图 1-8. PWDMASSERASE[y]

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

表 1-15. PWDMASSERASE[y] 字段说明

位	字段	类型	复位	说明
31-0	PW	R/W	FFFFFFFFh	密码

1.5.8 PWDFACTORYRESET[y]（偏移 = 41C00034h + 公式）[复位 = FFFFFFFFh]

图 1-9 展示了 PWDFACTORYRESET[y]，表 1-16 中对此进行了介绍。

返回到汇总表。

SWD 恢复出厂设置命令密码（必须通过 DSSM 提供，以便验证恢复出厂设置命令）。

偏移 = 41C00034h + (y * 4h)；其中 y = 0h 至 3h

图 1-9. PWDFACTORYRESET[y]

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

表 1-16. PWDFACTORYRESET[y] 字段说明

位	字段	类型	复位	说明
31-0	PW	R/W	FFFFFFFFh	密码

1.5.9 FLASHSWP0（偏移 = 41C00044h）[复位 = FFFFFFFFh]

图 1-10 展示了 FLASHSWP0，表 1-17 中对此进行了介绍。

返回到汇总表。

前 32kB 闪存的静态写保护策略。受保护时，扇区将无法通过引导加载程序或应用程序代码进行编程或擦除。

图 1-10. FLASHSWP0

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

表 1-17. FLASHSWP0 字段说明

位	字段	类型	复位	说明
31-0	MAINLOW	R/W	FFFFFFFFh	每个扇区 1 位（将某个位设置为 0 会禁用写入，设置为 1 会启用写入）。

1.5.10 FLASHSWP1（偏移 = 41C00048h）[复位 = FFFFFFFFh]

图 1-11 展示了 FLASHSWP1，表 1-18 中对此进行了介绍。

返回到汇总表。

闪存其他扇区的静态写保护策略。受保护时，扇区将无法通过引导加载程序或应用程序代码进行编程或擦除。

图 1-11. FLASHSWP1

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

表 1-18. FLASHSWP1 字段说明

位	字段	类型	复位	说明
31-0	MAINHIGH	R/W	FFFFFFFFh	每 8 个扇区 1 位。位 3:0，未像 FLASHSWP0 那样使用。（将某个位设置为 0 会禁用写入，设置为 1 会启用写入）

1.5.11 BOOTCFG4（偏移 = 41C0004Ch）[复位 = FFFFFFFFh]

图 1-12 展示了 BOOTCFG4，表 1-19 中对此进行了介绍。

返回到汇总表。

图 1-12. BOOTCFG4

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

表 1-19. BOOTCFG4 字段说明

位	字段	类型	复位	说明
31-16	APPCRCMODE	R/W	FFFFh	APPCRCMODE 启用或禁用 MAIN 闪存段的引导时间 CRC 校验。 5566h = 禁用引导时间 MAIN 闪存 CRC 校验。除非复位矢量或栈指针为空/未编程，否则 MAIN 闪存中的应用程序代码始终处于启动状态（0x5566 和所有其他非 0xAABB 值）。 AABBh = 启用引导时间 MAIN 闪存 CRC 校验。如果通过引导时间 CRC 校验，则启动 MAIN 闪存中的应用代码，除非复位矢量或栈指针为空（未编程）。如果未通过 CRC 校验，MAIN 闪存中的应用代码将不会启动，引导过程将失败。
15-1	RESERVED	R/W	0h	
0	NONMAINSWP	R/W	FFFFh	整个 NONMAIN 器件配置存储器的静态写保护策略。将位设置为 0 会禁用通过 SWD 启动的恢复出厂重置以外的所有方式对 NONMAIN 进行编程/擦除，设置为 1 会启用通过正常方式对 NONMAIN 进行编程/擦除。

1.5.12 APPCRCSTART（偏移 = 41C00050h）[复位 = FFFFFFFFh]

图 1-13 展示了 APPCRCSTART，表 1-20 中对此进行了介绍。

返回到汇总表。

应用 CRC 校验的起始地址（必须是 MAIN 闪存区域中的地址）。

图 1-13. APPCRCSTART

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

表 1-20. APPCRCSTART 字段说明

位	字段	类型	复位	说明
31-0	ADDRESS	R/W	FFFFFFFFh	应用 CRC 校验起始地址

1.5.13 APPCRCLENGTH（偏移 = 41C00054h）[复位 = FFFFFFFFh]

图 1-14 展示了 APPCRCLENGTH，表 1-21 中对此进行了介绍。

返回到汇总表。

要包括在应用 CRC 校验中的应用区域的长度（以字节为单位），从 APPCRCSTART 开始。

图 1-14. APPCRCLENGTH

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

表 1-21. APPCRCLENGTH 字段说明

位	字段	类型	复位	说明
31-0	长度	R/W	FFFFFFFFh	应用 CRC 校验源数据长度

1.5.14 APPCRC（偏移 = 41C00058h）[复位 = FFFFFFFFh]

图 1-15 展示了 APPCRC，表 1-22 中对此进行了介绍。

返回到汇总表。

引导期间要测试的预期应用 CRC 校验摘要 (CRC-32)。

图 1-15. APPCRC

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

表 1-22. APPCRC 字段说明

位	字段	类型	复位	说明
31-0	DIGEST	R/W	FFFFFFFFh	应用程序 CRC 校验预期摘要。

1.5.15 BOOTCRC（偏移 = 41C0005Ch）[复位 = 1879DAC3h]

图 1-16 展示了 BOOTCRC，表 1-23 中对此进行了介绍。

返回到汇总表。

NONMAIN 存储器 BCR（引导配置）部分的 CRC 摘要 (CRC-32)。

图 1-16. BOOTCRC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIGEST																															
R/W-1879DAC3h																															

表 1-23. BOOTCRC 字段说明

位	字段	类型	复位	说明
31-0	DIGEST	R/W	1879DAC3h	BCR 引导配置数据 CRC 摘要。

1.5.16 BSLCONFIGID（偏移 = 41C00100h）[复位 = 00000001h]

图 1-17 展示了 BSLCONFIGID，表 1-24 中对此进行了介绍。

返回到汇总表。

BSL 配置 ID。

图 1-17. BSLCONFIGID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONFIG																															
R/W-00000001h																															

表 1-24. BSLCONFIGID 字段说明

位	字段	类型	复位	说明
31-0	CONFIG	R/W	00000001h	BSL_CONFIG 的配置 ID。

1.5.17 BSLPINCFG0（偏移 = 41C00104h）[复位 = 02180217h]

图 1-18 展示了 BSLPINCFG0，表 1-25 中对此进行了介绍。

返回到汇总表。

BSL UART 引脚配置。

图 1-18. BSLPINCFG0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UARTTX_MUX_SEL								UARTTX_PAD_NUM							
R/W-02h								R/W-18h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UARTRX_MUX_SEL								UARTRX_PAD_NUM							
R/W-02h								R/W-17h							

表 1-25. BSLPINCFG0 字段说明

位	字段	类型	复位	说明
31-24	UARTTX_MUX_SEL	R/W	02h	UART TX IOMUX PINCM 多路复用器选择。
23-16	UARTTX_PAD_NUM	读/写	18h	UART TX IOMUX PINCM 寄存器。
15-8	UARTRX_MUX_SEL	R/W	02h	UART RX IOMUX PINCM 多路复用器选择。
7-0	UARTRX_PAD_NUM	R/W	17h	UART RX IOMUX PINCM 寄存器。

1.5.18 BSLPINCFG1（偏移 = 41C00108h）[复位 = 03020301h]

图 1-19 展示了 BSLPINCFG1，表 1-26 中对此进行了介绍。

返回到汇总表。

BSL I2C 引脚配置。

图 1-19. BSLPINCFG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2CSCL_MUX_SEL								I2CSCL_PAD_NUM							
R/W-03h								R/W-2h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CSDA_MUX_SEL								I2CSDA_PAD_NUM							
R/W-03h								R/W-1h							

表 1-26. BSLPINCFG1 字段说明

位	字段	类型	复位	说明
31-24	I2CSCL_MUX_SEL	R/W	03h	I2C SCL IOMUX PINCM 多路复用器选择。
23-16	I2CSCL_PAD_NUM	R/W	2h	I2C SCL IOMUX PINCM 寄存器。
15-8	I2CSDA_MUX_SEL	R/W	03h	I2C SDA IOMUX PINCM 多路复用器选择。
7-0	I2CSDA_PAD_NUM	R/W	1h	I2C SDA IOMUX PINCM 寄存器。

1.5.19 BSLCONFIG0（偏移 = 41C0010Ch）[复位 = FFFF1293h]

图 1-20 展示了 BSLCONFIG0，表 1-27 中对此进行了介绍。

返回到汇总表。

BSL 调用引脚配置和存储器读出策略。

图 1-20. BSLCONFIG0

31	30	29	28	27	26	25	24
READOUTEN							
R/W-FFFFh							
23	22	21	20	19	18	17	16
READOUTEN							
R/W-FFFFh							
15	14	13	12	11	10	9	8
保留		BSLIVK_GPIOPORT	BSLIVK_GPIOPIN				
R-0h		R/W-	R/W-12h				
7	6	5	4	3	2	1	0
BSLIVK_LVL	保留	BSLIVK_PAD_NUM					
R/W-1h	R-0h	R/W-13h					

表 1-27. BSLCONFIG0 字段说明

位	字段	类型	复位	说明
31-16	READOUTEN	R/W	FFFFh	设置 BSL 接口的存储器读出策略。 5566h = 无法通过 BSL 接口读取存储器（0x5566 和所有其他非 0xAABB 值）。 AABBh = 存储器内容可通过 BSL 接口读取。
15-14	保留	R	0h	
13	BSLIVK_GPIOPORT	R/W	0h	与用于 BSL_invoke 的焊盘对应的 BSL_invoke GPIO 端口索引。 0h = BSL_invoke 引脚位于 GPIO 端口 A 上。 1h = BSL_invoke 引脚位于 GPIO 端口 B 上。
12-8	BSLIVK_GPIOPIN	R/W	12h	与用于 BSL_invoke 的焊盘对应的 BSL_invoke GPIO 引脚索引。
7	BSLIVK_LVL	R/W	1h	将调用 BSL 的 BSL_invoke 输入逻辑电平。 0h = 低电平 1h = 高电平
6	RESERVED	R	0h	
5-0	BSLIVK_PAD_NUM	R/W	13h	与用于 BSL_invoke 的焊盘对应的 IOMUX PINCM 寄存器。

1.5.20 BSLPW[y]（偏移 = 41C00110h + 公式）[复位 = FFFFFFFFh]

图 1-21 展示了 BSLPW[y]，表 1-28 中对此进行了介绍。

返回到汇总表。

256 位 BSL 访问密码。

偏移 = 41C00110h + (y * 4h)；其中 y = 0h 至 7h

图 1-21. BSLPW[y]

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

表 1-28. BSLPW[y] 字段说明

位	字段	类型	复位	说明
31-0	PASSWORD	R/W	FFFFFFFFh	密码

1.5.21 BSLPLUGINCFG（偏移 = 41C00130h）[复位 = FFFFFFFFh]

图 1-22 展示了 BSLPLUGINCFG，表 1-29 中对此进行了介绍。

返回到汇总表。

定义 MAIN 闪存中是否存在 BSL 插件及其类型。

图 1-22. BSLPLUGINCFG

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAMEXISTS								FLASHEXISTS							
R/W-FFh								R/W-FFh							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLUGINTYPE															
R/W-FFFFh															

表 1-29. BSLPLUGINCFG 字段说明

位	字段	类型	复位	说明
31-24	SRAMEXISTS	R/W	FFh	闪存插件消耗的 SRAM，从 0x00 到 0xFF。
23-16	FLASHEXISTS	R/W	FFh	该字段指示闪存插件是否存在。0xBB - 存在闪存插件；0xFF（所有其他值）- 将仅使用 ROM 插件。
15-0	PLUGINTYPE	R/W	FFFFh	BSL 插件的类型代码。 1000h = 插件适用于 UART。 2000h = 插件适用于 I2C。 FFFFh = 用于所有其他值。任何其他具有有效挂钩的接口都将添加到插件列表中。

1.5.22 BSLPLUGINHOOK[y]（偏移 = 41C00134h + 公式）[复位 = FFFFFFFFh]

图 1-23 展示了 BSLPLUGINHOOK[y]，表 1-30 中对此进行了介绍。

返回到汇总表。

用于插件初始化、接收、发送和取消初始化函数的函数指针。

偏移 = 41C00134h + (y * 4h)；其中 y = 0h 至 3h

图 1-23. BSLPLUGINHOOK[y]

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

表 1-30. BSLPLUGINHOOK[y] 字段说明

位	字段	类型	复位	说明
31-0	BSLPLUGIN	R/W	FFFFFFFFh	BSL 插件挂钩的地址。字节 [3-0]：Init；字节 [7-4]：Receive；字节 [11-8]：Send；字节 [15-12]：Deinit

1.5.23 PATCHHOOKID（偏移 = 41C00144h）[复位 = FFFFFFFFh]

图 1-24 展示了 PATCHHOOKID，表 1-31 中对此进行了介绍。

返回到汇总表。

备用 BSL 配置。

图 1-24. PATCHHOOKID

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

表 1-31. PATCHHOOKID 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	ID	R/W	FFFFFFFFh	用于调用备用 BSL 的 ID 字段。 5566h = 不使用备用 BSL（0x5566 和所有其他非 0xAABB 值）。 AABBh = 使用备用 BSL。

1.5.24 SBLADDRESS（偏移 = 41C00148h）[复位 = FFFFFFFFh]

图 1-25 展示了 SBLADDRESS，表 1-32 中对此进行了介绍。

返回到汇总表。

备用 BSL 的地址。

图 1-25. SBLADDRESS

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

表 1-32. SBLADDRESS 字段说明

位	字段	类型	复位	说明
31-0	ADDRESS	R/W	FFFFFFFFh	备用 BSL 的地址（如果存在）。

1.5.25 BSLAPPVER（偏移 = 41C0014Ch）[复位 = FFFFFFFFh]

图 1-26 展示了 BSLAPPVER，表 1-33 中对此进行了介绍。

返回到汇总表。

应用版本字的地址。

图 1-26. BSLAPPVER

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

表 1-33. BSLAPPVER 字段说明

位	字段	类型	复位	说明
31-0	ADDRESS	R/W	FFFFFFFFh	应用版本字的地址（必须是要返回的有效闪存地址）。

1.5.26 BSLCONFIG1（偏移 = 41C00150h）[复位 = 0048FFFFh]

图 1-27 展示了 BSLCONFIG1，表 1-34 中对此进行了介绍。

返回到汇总表。

BSL 安全配置。

图 1-27. BSLCONFIG1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TARGETADDR																ALERTACTION															
R/W-0048h																R/W-FFFFh															

表 1-34. BSLCONFIG1 字段说明

位	字段	类型	复位	说明
31-16	TARGETADDR	R/W	0048h	用于 BSL I2C 通信的 I2C 目标地址。
15-0	ALERTACTION	R/W	FFFFh	在出现安全警报条件时采取的措施。 5566h = 忽略安全警报条件（0x5566 和所有其他非 0xAABB 或 0xCCDD 值）。 AABBh = 触发恢复出厂设置。请注意，如果 MAIN 或 NONMAIN 闪存中的扇区受写保护，它们将不受 BSL 恢复出厂设置的影响。 CCDDh = 重新配置 NONMAIN 区域以禁用 BSL。如果 NONMAIN 区域配置为受写保护，则不支持此操作。

1.5.27 BSLCRC（偏移 = 41C00154h）[复位 = 8C76DE95h]

图 1-28 展示了 BSLCRC，表 1-35 中对此进行了介绍。

返回到汇总表。

NONMAIN 存储器 BSL_CONFIG 部分的 CRC 摘要 (CRC-32)。

图 1-28. BSLCRC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIGEST																															
R/W-8C76DE95h																															

表 1-35. BSLCRC 字段说明

位	字段	类型	复位	说明
31-0	DIGEST	R/W	8C76DE95h	BSL 配置数据 CRC 摘要

1.6 出厂常量

所有器件都包含一个存储器映射出厂区域，该区域提供描述器件功能的只读数据以及任何出厂提供的修整信息，供应用软件使用。

FACTORY 存储器区域中提供的关键数据包括：

- 器件唯一 96 位标识
- 默认 BSL 引脚
- MAIN 区域闪存总大小（以 KB 为单位）
- DATA 区域闪存总大小（以 KB 为单位）（如果存在）
- 闪存存储体计数
- SRAM 存储器总大小（以 KB 为单位）
- 温度传感器校准值
- SYSPLL 启动参数（如果存在 SYSPLL，否则保留）

1.6.1 FACTORYREGION 寄存器

表 1-36 列出了 FACTORYREGION 寄存器的存储器映射寄存器。表 1-36 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 1-36. FACTORYREGION 寄存器

偏移	缩写	寄存器名称	组	部分
41C40000h	TRACEID	跟踪标识符		转到
41C40004h	DEVICEID	器件标识符		转到
41C40008h	USERID	器件型号标识符		转到
41C4000Ch	BSLPIN_UART			转到
41C40010h	BSLPIN_I2C			转到
41C40014h	BSLPIN_INVOKE			转到
41C40018h	SRAMFLASH			转到
41C4001Ch	PLLSTARTUP0_4_8MHZ			转到
41C40020h	PLLSTARTUP1_4_8MHZ	系统 PLL 参数 1 MMR - 闪存表查找数据		转到
41C40024h	PLLSTARTUP0_8_16MHZ			转到
41C40028h	PLLSTARTUP1_8_16MHZ	系统 PLL 参数 1 MMR - 闪存表查找数据		转到
41C4002Ch	PLLSTARTUP0_16_32MHZ			转到
41C40030h	PLLSTARTUP1_16_32MHZ	系统 PLL 参数 1 MMR - 闪存表查找数据		转到
41C40034h	PLLSTARTUP0_32_48MHZ			转到
41C40038h	PLLSTARTUP1_32_48MHZ	系统 PLL 参数 1 MMR - 闪存表查找数据		转到
41C4003Ch	TEMP_SENSE0	温度传感器室温校准代码。 这是温度传感器输出电压的 ADC 转换结果。 包含在 BOOTCRC 计算中。		转到
41C40040h	RESERVED0			转到
41C40044h	RESERVED1			转到
41C40048h	保留 2			转到
41C4004Ch	保留 3			转到
41C40050h	RESERVED4			转到
41C40054h	RESERVED5			转到
41C40058h	RESERVED6			转到

表 1-36. FACTORYREGION 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
41C4005Ch	RESERVED7			转到
41C40060h	RESERVED8			转到
41C40064h	RESERVED9			转到
41C40068h	RESERVED10			转到
41C4006Ch	RESERVED11			转到
41C40070h	RESERVED12			转到
41C40074h	RESERVED13			转到
41C40078h	RESERVED14			转到
41C4007Ch	BOOTCRC	BOOTCRC 会记录处于 OPEN 状态的所有位置（包括保留位置）的 32 位 CRC。		转到

复杂的位访问类型经过编码可适应小型表单元。表 1-37 展示了适用于此部分中访问类型的代码。

表 1-37. FACTORYREGION 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

1.6.1.1 TRACEID（偏移 = 41C40000h）[复位 = 00000000h]

图 1-29 展示了 TRACEID，表 1-38 中对此进行了介绍。

返回到汇总表。

已发货的每个器件的唯一值

图 1-29. TRACEID

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

表 1-38. TRACEID 寄存器字段说明

位	字段	类型	复位	说明
31-0	数据	R	0h	

1.6.1.2 DEVICEID（偏移 = 41C40004h）[复位 = 0BB8802Fh]

图 1-30 展示了 DEVICEID，表 1-39 中对此进行了介绍。

返回到汇总表。

器件标识符（特定于裸片版本）

图 1-30. DEVICEID

31	30	29	28	27	26	25	24
VERSION				PARTNUM			
R-0h				R-BB88h			
23	22	21	20	19	18	17	16
PARTNUM							
R-BB88h							
15	14	13	12	11	10	9	8
PARTNUM				制造商			
R-BB88h				R-17h			
7	6	5	4	3	2	1	0
制造商							ALWAYS_1
R-17h							R-1h

表 1-39. DEVICEID 字段说明

位	字段	类型	复位	说明
31-28	VERSION	R	0h	器件的修订版本
27-12	PARTNUM	R	BB88h	器件型号
11-1	制造商	R	17h	TI 的 JEDEC 银行和公司代码
0	ALWAYS_1	R	1h	该位始终为 1

1.6.1.3 USERID（偏移 = 41C40008h）[复位 = 80000000h]

图 1-31 展示了 USERID，表 1-40 中对此进行了介绍。

返回到汇总表。

根据连接格式，定义型号功能集

图 1-31. USERID

31	30	29	28	27	26	25	24
启动	MAJORREV			MINORREV			
R-1h	R-			R-			
23	22	21	20	19	18	17	16
变体							
R-							
15	14	13	12	11	10	9	8
部件							
R-							
7	6	5	4	3	2	1	0
部件							
R-							

表 1-40. USERID 字段说明

位	字段	类型	复位	说明
31	启动	R	1h	
30-28	MAJORREV	R	0h	单调增大的值，表示 SKU 的新修订足够重大，器件的用户可能必须修改 PCB 或软件设计
27-24	MINORREV	R	0h	单调增加的值，表示 SKU 的新修订版保留了与较小 minorrev 值的兼容性。如果使用了新功能，则可能会引入新功能，以至于较小的 minorrev 编号可能与较大的编号不兼容。
23-16	变体	R	0h	唯一标识器件型号的位模式
15-0	部件	R	0h	唯一标识器件的位模式

1.6.1.4 BSLPIN_UART (偏移 = 41C4000Ch) [复位 = 02180219h]

图 1-32 展示了 BSLPIN_UART，表 1-41 中对此进行了介绍。

返回到汇总表。

图 1-32. BSLPIN_UART

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

表 1-41. BSLPIN_UART 字段说明

位	字段	类型	复位	说明
31-24	UART_TXT_PF	R	2h	
23-16	UART_TXD_PAD	R	18h	
15-8	UART_RXD_PF	R	2h	
7-0	UART_RXD_PAD	R	19h	

1.6.1.5 BSLPIN_I2C（偏移 = 41C40010h）[复位 = 03020301h]

图 1-33 展示了 BSLPIN_I2C，表 1-42 中对此进行了介绍。

返回到汇总表。

图 1-33. BSLPIN_I2C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I2C_SCL_PF								I2C_SCL_PAD							
R-3h								R-2h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C_SDA_PF								I2C_SDA_PAD							
R-3h								R-1h							

表 1-42. BSLPIN_I2C 字段说明

位	字段	类型	复位	说明
31-24	I2C_SCL_PF	R	3h	
23-16	I2C_SCL_PAD	R	2h	
15-8	I2C_SDA_PF	R	3h	
7-0	I2C_SDA_PAD	R	1h	

1.6.1.6 BSLPIN_INVOKE（偏移 = 41C40014h）[复位 = 000001ABh]

图 1-34 展示了 BSLPIN_INVOKE，表 1-43 中对此进行了介绍。

返回到汇总表。

图 1-34. BSLPIN_INVOKE

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED							
R-							
15	14	13	12	11	10	9	8
GPIO_REG_SEL		GPIO_PIN_SEL					
R-		R-1h					
7	6	5	4	3	2	1	0
GPIO_LEVEL	BSL_PAD						
R-1h	R-2Bh						

表 1-43. BSLPIN_INVOKE 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15-14	GPIO_REG_SEL	R	0h	
13-8	GPIO_PIN_SEL	R	1h	
7	GPIO_LEVEL	R	1h	
6-0	BSL_PAD	R	2Bh	

1.6.1.7 SRAMFLASH（偏移 = 41C40018h）[复位 = 00200080h]

图 1-35 展示了 SRAMFLASH，表 1-44 中对此进行了介绍。

返回到汇总表。

图 1-35. SRAMFLASH

31	30	29	28	27	26	25	24
DATAFLASH_SZ						SRAM_SZ	
R-0h						R-20h	
23	22	21	20	19	18	17	16
SRAM_SZ							
R-20h							
15	14	13	12	11	10	9	8
保留		MAINNUMBANKS		MAINFLASH_SZ			
R-		R-0h		R-80h			
7	6	5	4	3	2	1	0
MAINFLASH_SZ							
R-80h							

表 1-44. SRAMFLASH 字段说明

位	字段	类型	复位	说明
31-26	DATAFLASH_SZ	R	0h	该字段的编码方式是，该字段的值是表示 KB 数的整数。 例如：如果该字段的值为 4，则为 4KB，如果值为 32，则为 32KB，依此类推。
25-16	SRAM_SZ	R	20h	该字段的编码方式是，该字段的值是表示 KB 数的整数。例如：如果该字段的值为 4，则为 4KB，如果值为 32，则为 32KB，依此类推。
15-14	保留	R	0h	
13-12	MAINNUMBANKS	R	0h	
11-0	MAINFLASH_SZ	R	80h	该字段的编码方式是，该字段的值是表示 KB 数的整数。例如：如果该字段的值为 4，则为 4KB，如果值为 32，则为 32KB，依此类推。

1.6.1.8 PLLSTARTUP0_4_8MHZ (偏移 = 41C4001Ch) [复位 = 00004496h]

图 1-36 展示了 PLLSTARTUP0_4_8MHZ，表 1-45 中对此进行了介绍。

返回到汇总表。

图 1-36. PLLSTARTUP0_4_8MHZ

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
CAPBOVERRIDE	CAPBVAL					CPCURRENT	
R/W-	R/W-					R/W-4h	
15	14	13	12	11	10	9	8
CPCURRENT				STARTTIMELP			
R/W-4h				R/W-12h			
7	6	5	4	3	2	1	0
STARTTIMELP		STARTTIME					
R/W-12h		R/W-16h					

表 1-45. PLLSTARTUP0_4_8MHZ 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23	CAPBOVERRIDE	R/W	0h	启用电容 B 覆盖 0h = 0 1h = 1
22-18	CAPBVAL	R/W	0h	电容 B 的覆盖值
17-12	CPCURRENT	R/W	4h	电荷泵电流
11-6	STARTTIMELP	R/W	12h	从低功耗退出到锁定时钟的启动时间，分辨率为 1us
5-0	STARTTIME	R/W	16h	从启用到锁定时钟的启动时间，分辨率为 1us

1.6.1.9 PLLSTARTUP1_4_8MHZ (偏移 = 41C40020h) [复位 = 0000805Ch]

图 1-37 展示了 PLLSTARTUP1_4_8MHZ，表 1-46 中对此进行了介绍。

返回到汇总表。

图 1-37. PLLSTARTUP1_4_8MHZ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED									LPFRESC						
R-									R/W-1h						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPFRESC	LPFRESA										LPFCAPA				
R/W-1h	R/W-2h										R/W-1Ch				

表 1-46. PLLSTARTUP1_4_8MHZ 字段说明

位	字段	类型	复位	说明
31-23	RESERVED	R	0h	
22-15	LPFRESC	R/W	1h	环路滤波器分辨率 C
14-5	LPFRESA	R/W	2h	环路滤波器分辨率 A
4-0	LPFCAPA	R/W	1Ch	环路滤波器电容 A

1.6.1.10 PLLSTARTUP0_8_16MHZ（偏移 = 41C40024h）[复位 = 000052CFh]

图 1-38 展示了 PLLSTARTUP0_8_16MHZ，表 1-47 中对此进行了介绍。

返回到汇总表。

图 1-38. PLLSTARTUP0_8_16MHZ

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
CAPBOVERRIDE	CAPBVAL					CPCURRENT	
R/W-	R/W-					R/W-5h	
15	14	13	12	11	10	9	8
CPCURRENT				STARTTIMELP			
R/W-5h				R/W-Bh			
7	6	5	4	3	2	1	0
STARTTIMELP		STARTTIME					
R/W-Bh		R/W-Fh					

表 1-47. PLLSTARTUP0_8_16MHZ 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23	CAPBOVERRIDE	R/W	0h	启用电容 B 覆盖 0h = 0 1h = 1
22-18	CAPBVAL	R/W	0h	电容 B 的覆盖值
17-12	CPCURRENT	R/W	5h	电荷泵电流
11-6	STARTTIMELP	R/W	Bh	从低功耗退出到锁定时钟的启动时间，分辨率为 1us
5-0	STARTTIME	R/W	Fh	从启用到锁定时钟的启动时间，分辨率为 1us

1.6.1.11 PLLSTARTUP1_8_16MHZ (偏移 = 41C40028h) [复位 = 00008030h]

图 1-39 展示了 PLLSTARTUP1_8_16MHZ，表 1-48 中对此进行了介绍。

返回到汇总表。

图 1-39. PLLSTARTUP1_8_16MHZ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED									LPFRESC						
R-									R/W-1h						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPFRESC	LPFRESA										LPFCAPA				
R/W-1h	R/W-1h										R/W-10h				

表 1-48. PLLSTARTUP1_8_16MHZ 字段说明

位	字段	类型	复位	说明
31-23	RESERVED	R	0h	
22-15	LPFRESC	R/W	1h	环路滤波器分辨率 C
14-5	LPFRESA	R/W	1h	环路滤波器分辨率 A
4-0	LPFCAPA	R/W	10h	环路滤波器电容 A

1.6.1.12 PLLSTARTUP0_16_32MHZ（偏移 = 41C4002Ch）[复位 = 0000520Ch]

图 1-40 展示了 PLLSTARTUP0_16_32MHZ，表 1-49 中对此进行了介绍。

返回到汇总表。

图 1-40. PLLSTARTUP0_16_32MHZ

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
CAPBOVERRIDE	CAPBVAL					CPCURRENT	
R/W-	R/W-					R/W-5h	
15	14	13	12	11	10	9	8
CPCURRENT				STARTTIMELP			
R/W-5h				R/W-8h			
7	6	5	4	3	2	1	0
STARTTIMELP		STARTTIME					
R/W-8h		R/W-Ch					

表 1-49. PLLSTARTUP0_16_32MHZ 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23	CAPBOVERRIDE	R/W	0h	启用电容 B 覆盖 0h = 0 1h = 1
22-18	CAPBVAL	R/W	0h	电容 B 的覆盖值
17-12	CPCURRENT	R/W	5h	电荷泵电流
11-6	STARTTIMELP	R/W	8h	从低功耗退出到锁定时钟的启动时间，分辨率为 1us
5-0	STARTTIME	R/W	Ch	从启用到锁定时钟的启动时间，分辨率为 1us

1.6.1.13 PLLSTARTUP1_16_32MHZ（偏移 = 41C40030h）[复位 = 00008030h]

图 1-41 展示了 PLLSTARTUP1_16_32MHZ，表 1-50 中对此进行了介绍。

返回到汇总表。

图 1-41. PLLSTARTUP1_16_32MHZ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED									LPFRESC						
R-									R/W-1h						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPFRESC	LPFRESA										LPFCAPA				
R/W-1h	R/W-1h										R/W-10h				

表 1-50. PLLSTARTUP1_16_32MHZ 字段说明

位	字段	类型	复位	说明
31-23	RESERVED	R	0h	
22-15	LPFRESC	R/W	1h	环路滤波器分辨率 C
14-5	LPFRESA	R/W	1h	环路滤波器分辨率 A
4-0	LPFCAPA	R/W	10h	环路滤波器电容 A

1.6.1.14 PLLSTARTUP0_32_48MHZ（偏移 = 41C40034h）[复位 = 0000518Ah]

图 1-42 展示了 PLLSTARTUP0_32_48MHZ，表 1-51 中对此进行了介绍。

返回到汇总表。

图 1-42. PLLSTARTUP0_32_48MHZ

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
CAPBOVERRIDE	CAPBVAL					CPCURRENT	
R/W-	R/W-					R/W-5h	
15	14	13	12	11	10	9	8
CPCURRENT				STARTTIMELP			
R/W-5h				R/W-6h			
7	6	5	4	3	2	1	0
STARTTIMELP		STARTTIME					
R/W-6h		R/W-Ah					

表 1-51. PLLSTARTUP0_32_48MHZ 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23	CAPBOVERRIDE	R/W	0h	启用电容 B 覆盖 0h = 0 1h = 1
22-18	CAPBVAL	R/W	0h	电容 B 的覆盖值
17-12	CPCURRENT	R/W	5h	电荷泵电流
11-6	STARTTIMELP	R/W	6h	从低功耗退出到锁定时钟的启动时间，分辨率为 1us
5-0	STARTTIME	R/W	Ah	从启用到锁定时钟的启动时间，分辨率为 1us

1.6.1.15 PLLSTARTUP1_32_48MHZ（偏移 = 41C40038h）[复位 = 00008030h]

图 1-43 展示了 PLLSTARTUP1_32_48MHZ，表 1-52 中对此进行了介绍。

返回到汇总表。

图 1-43. PLLSTARTUP1_32_48MHZ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED									LPFRESC						
R-									R/W-1h						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPFRESC	LPFRESA										LPFCAPA				
R/W-1h	R/W-1h										R/W-10h				

表 1-52. PLLSTARTUP1_32_48MHZ 字段说明

位	字段	类型	复位	说明
31-23	RESERVED	R	0h	
22-15	LPFRESC	R/W	1h	环路滤波器分辨率 C
14-5	LPFRESA	R/W	1h	环路滤波器分辨率 A
4-0	LPFCAPA	R/W	10h	环路滤波器电容 A

1.6.1.16 TEMP_SENSE0（偏移 = 41C4003Ch）[复位 = 00000000h]

图 1-44 展示了 TEMP_SENSE0，表 1-53 中对此进行了介绍。

返回到汇总表。

温度传感器室温校准代码。这是温度传感器输出电压的 ADC 转换结果。包含在 BOOTCRC 计算中。

图 1-44. TEMP_SENSE0

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

表 1-53. TEMP_SENSE0 字段说明

位	字段	类型	复位	说明
31-0	数据	R	0h	

1.6.1.17 RESERVED0（偏移 = 41C40040h）[复位 = 00000000h]

图 1-45 展示了 RESERVED0，表 1-54 中对此进行了介绍。

返回到汇总表。

图 1-45. RESERVED0

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

表 1-54. RESERVED0 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.18 RESERVED1（偏移 = 41C40044h）[复位 = 00000000h]

图 1-46 展示了 RESERVED1，表 1-55 中对此进行了介绍。

返回到汇总表。

图 1-46. RESERVED1

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

表 1-55. RESERVED1 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.19 RESERVED2（偏移 = 41C40048h）[复位 = 00000000h]

图 1-47 展示了 RESERVED2，表 1-56 中对此进行了介绍。

返回到汇总表。

图 1-47. 保留 2

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

表 1-56. RESERVED2 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.20 RESERVED3（偏移 = 41C4004Ch）[复位 = 00000000h]

图 1-48 展示了 RESERVED3，表 1-57 中对此进行了介绍。

返回到汇总表。

图 1-48. 保留 3

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

表 1-57. RESERVED3 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.21 RESERVED4（偏移 = 41C40050h）[复位 = 00000000h]

图 1-49 展示了 RESERVED4，表 1-58 中对此进行了介绍。

返回到汇总表。

图 1-49. RESERVED4

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

表 1-58. RESERVED4 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.22 RESERVED5（偏移 = 41C40054h）[复位 = 00000000h]

图 1-50 展示了 RESERVED5，表 1-59 中对此进行了介绍。

返回到汇总表。

图 1-50. RESERVED5

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

表 1-59. RESERVED5 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.23 RESERVED6（偏移 = 41C40058h）[复位 = 00000000h]

图 1-51 展示了 RESERVED6，表 1-60 中对此进行了介绍。

返回到汇总表。

图 1-51. RESERVED6

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

表 1-60. RESERVED6 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.24 RESERVED7（偏移 = 41C4005Ch）[复位 = 00000000h]

图 1-52 展示了 RESERVED7，表 1-61 中对此进行了介绍。

返回到汇总表。

图 1-52. RESERVED7

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

表 1-61. RESERVED7 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.25 RESERVED8（偏移 = 41C40060h）[复位 = 00000000h]

图 1-53 展示了 RESERVED8，表 1-62 中对此进行了介绍。

返回到汇总表。

图 1-53. RESERVED8

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

表 1-62. RESERVED8 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.26 RESERVED9（偏移 = 41C40064h）[复位 = 00000000h]

图 1-54 展示了 RESERVED9，表 1-63 中对此进行了介绍。

返回到汇总表。

图 1-54. RESERVED9

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

表 1-63. RESERVED9 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.27 RESERVED10（偏移 = 41C40068h）[复位 = 00000000h]

图 1-55 展示了 RESERVED10，表 1-64 中对此进行了介绍。

返回到汇总表。

图 1-55. RESERVED10

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

表 1-64. RESERVED10 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.28 RESERVED11（偏移 = 41C4006Ch）[复位 = 00000000h]

图 1-56 展示了 RESERVED11，表 1-65 中对此进行了介绍。

返回到汇总表。

图 1-56. RESERVED11

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

表 1-65. RESERVED11 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.29 RESERVED12（偏移 = 41C40070h）[复位 = 00000000h]

图 1-57 展示了 RESERVED12，表 1-66 中对此进行了介绍。

返回到汇总表。

图 1-57. RESERVED12

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

表 1-66. RESERVED12 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.30 RESERVED13（偏移 = 41C40074h）[复位 = 00000000h]

图 1-58 展示了 RESERVED13，表 1-67 中对此进行了介绍。

返回到汇总表。

图 1-58. RESERVED13

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

表 1-67. RESERVED13 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.31 RESERVED14（偏移 = 41C40078h）[复位 = 00000000h]

图 1-59 展示了 RESERVED14，表 1-68 中对此进行了介绍。

返回到汇总表。

图 1-59. RESERVED14

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

表 1-68. RESERVED14 字段说明

位	字段	类型	复位	说明
31-0	保留	R	0h	保留

1.6.1.32 BOOTCRC（偏移 = 41C4007Ch）[复位 = 00000000h]

图 1-60 展示了 BOOTCRC，表 1-69 中对此进行了介绍。

返回到汇总表。

BOOTCRC 会记录处于 OPEN 状态的所有位置（包括保留位置）的 32 位 CRC。

图 1-60. BOOTCRC

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

表 1-69. BOOTCRC 字段说明

位	字段	类型	复位	说明
31-0	数据	R	0h	

This page intentionally left blank.

电源管理和时钟单元 (PMCU) 是一种统一的系统模块，可为器件提供所有电源管理、时钟配置和复位控制功能。用于运行器件的所有电源管理单元 (PMU) 和时钟模块 (CKM) 策略均通过系统控制器 (SYSCTL) 中的存储器映射寄存器进行配置。

2.1 PMCU 概述

电源管理和时钟单元 (PMCU) 为器件提供所有电源、时钟、复位和系统控制服务。为提供此功能，PMCU 包含三个子模块：电源管理单元 (**PMU**)、时钟模块 (**CKM**) 和系统控制器 (**SYSCTL**)。

PMU 是一个模拟子模块，可为器件生成内部稳压电源并监控外部电源的状态。PMU 还包含由片上稳压器和模拟外设使用的电压和电流基准电路。

CKM 也是一模拟子模块，提供时钟源（内部和外部振荡器）并将这些时钟源呈现给 SYSCTL。SYSCTL 将这些时钟源分配给器件上的 CPU、总线和外设。

SYSCTL 是一个数字子模块，为 PMCU 中的所有功能提供控制逻辑。此外，SYSCTL 包含供软件用于配置电源管理和时钟、评估器件状态以及控制复位的存储器映射寄存器。SYSCTL 还提供在 SHUTDOWN 模式下保留的 4 字节通用存储器，可用于在 SRAM 和寄存器内容丢失时存储 SHUTDOWN 模式下的状态信息。

图 2-1 展示了 PMCU 与器件电源、时钟和信号之间的接口。软件对 PMCU 的配置始终通过 SYSCTL 子模块中的存储器映射寄存器完成。

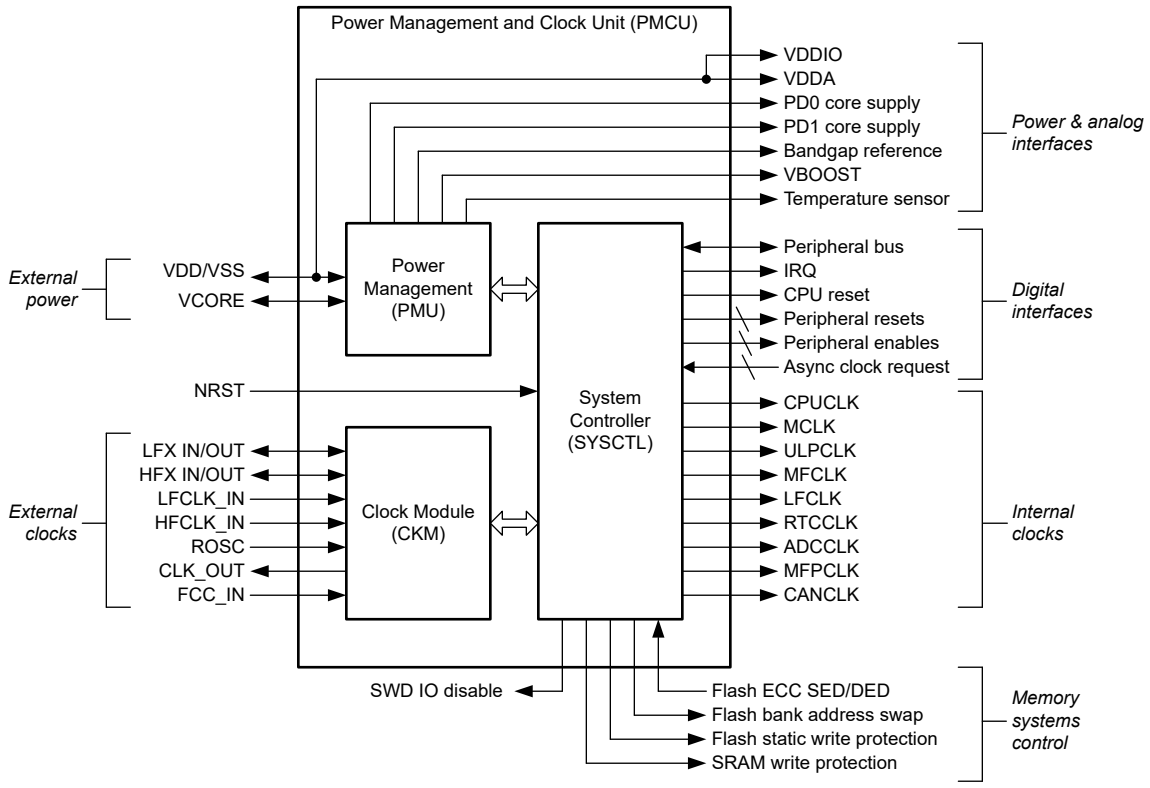


图 2-1. MSPM0Gxx PMCU 顶层

备注

并非所有器件都具有图 2-1 所示的所有 PMCU 特性。例如，并非所有器件都具有 LFXT（低频晶体振荡器）、HFXT（高频晶体振荡器）和 CANCLK。请参阅器件特定的数据表，了解给定器件的特性。

使用本指南

本章的 PMU、CKM 和 SYSCTL 几节详细介绍了每个子模块提供的功能。

快速入门一节介绍了 PMCU 的总体系统级工作原理以及如何针对不同应用场景配置 PMCU。

2.1.1 电源域

器件上提供了两个内核电源域：PD1 和 PD0。PD1 在 RUN 和 SLEEP 模式下始终通电，但在所有其他模式下会被禁用。在 RUN、SLEEP、STOP 和 STANDBY 模式下，PD0 始终处于通电状态。在 SHUTDOWN 模式下，PD1 和 PD0 都会被禁用。

- PD1 域包括 CPU 子系统、SRAM 存储器、PD1 外设和 PD1 外设总线，该总线通过 MCLK（包括 DMA，最高频率为 80MHz）运行。在 STOP 和 STANDBY 模式下禁用 PD1 时，CPU 寄存器、SRAM 和外设 MMR 配置寄存器将保持不变，以便在退出 STOP 或 STANDBY 模式时可立即使用它们恢复运行。
- PD0 域包括 PD0 外设和 PD0 总线段，该总线段通过 ULPCLK 运行（在 RUN 和 SLEEP 模式下的最高频率为 40MHz，在 STOP 模式下为 4MHz，在 STANDBY 模式下为 32kHz）。PD0 域在除 SHUTDOWN 模式之外的所有模式下均通电，可视为“常开”域。

器件特定的数据表中介绍了器件上哪些外设位于 PD1 中，哪些位于 PD0 中。

该器件还具有为 IO 和模拟外设供电的单个外部电源 (VDD) 域。

2.1.2 工作模式

该器件提供五种工作模式（功耗模式），可根据应用要求优化器件功耗。这些模式按照功耗从高到低排列如下：RUN、SLEEP、STOP、STANDBY 和 SHUTDOWN。图 2-2 展示了各种模式之间的交互。

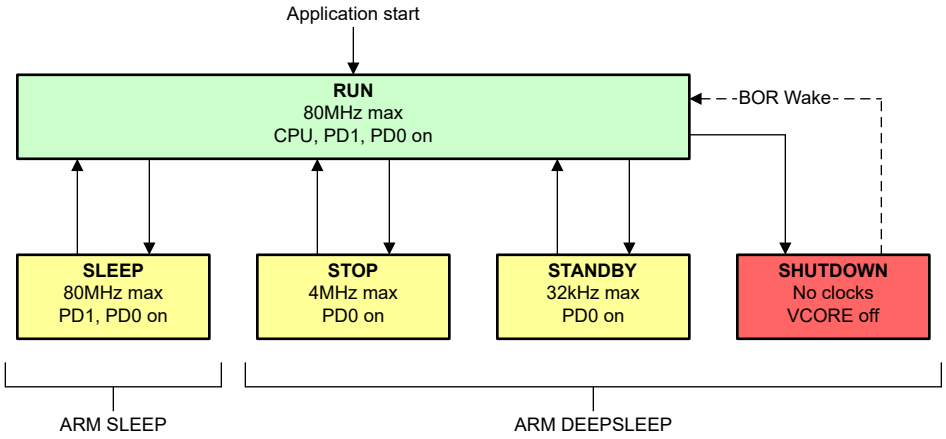


图 2-2. MSPM0Gxx 工作模式

节 2.1.2.6 展示了器件每个工作模式下可用的功能。有关如何将器件配置为特定工作模式的信息，请参阅工作模式选择一节。

工作模式概念

MSPM0 MCU 实施基于策略的电源和时钟管理方案。可以通过应用软件来配置策略，确定如何在每个工作模式下管理时钟，从而在给定应用的功耗和性能之间实现良好平衡。

配置每种模式的工作策略后，应用软件可以通过简单的寄存器命令进入和退出各种工作模式，SYSCTL 会根据软件定义的策略和软件选择的模式自动管理所有必要的 PMU 状态、振荡器和时钟的启用和禁用以及 SYSOSC 频率。

此外，还存在各种硬件触发的低功耗模式暂停机制，以便在受支持的外设发出相应请求时可以按需访问快速时钟，以及启用从低功耗模式触发 DMA 和 ADC 等功能。

策略驱动的工作模式方案与异步低功耗模式暂停机制相结合，使应用软件能够选择工作模式和相应的策略，以便后台活动具有尽可能低的功耗，而瞬态前台活动可以启动 DMA、启动快速时钟或使器件进入 RUN 状态（根据 IRQ）以进行突发处理。

2.1.2.1 RUN 模式

在 RUN 模式下，CPU 正在执行代码并且可以启用任何外设。

共有三个 RUN 模式策略选项：RUN0、RUN1 和 RUN2。

- **RUN0**：MCLK 和 CPUCLK 通过快速时钟源（SYSOSC、HFCLK 或 SYSPLL）运行。
- **RUN1**：MCLK 和 CPUCLK 通过 LFCLK (32kHz) 运行以降低有功功率，但 SYSOSC 会保持启用状态以用于 ADC、DAC、OPA 或 COMP 等模拟模块（在 HS 模式下）。
- **RUN2**：MCLK 和 CPUCLK 通过 LFCLK (32kHz) 运行，并且会完全禁用 SYSOSC 以节能。这是 CPU 运行时的最低功耗状态。

2.1.2.2 SLEEP 模式

在 SLEEP 模式下，会禁用 CPU（时钟选通），但在其他方面，器件配置与在 RUN 模式下相同。共有三个 SLEEP 模式策略选项：SLEEP0、SLEEP1 和 SLEEP2。SLEEPx 策略由进入 SLEEP 模式时的当前 RUNx 策略确定。

- **SLEEP0**：与 RUN0 相同，但会禁用 CPU。
- **SLEEP1**：与 RUN1 相同，但会禁用 CPU。
- **SLEEP2**：与 RUN2 相同，但会禁用 CPU。

2.1.2.3 STOP 模式

在 STOP 模式下，CPU、SRAM 和 PD1 外设被禁用并保留（如果适用）。PD0 外设的最高 ULPCLK 频率为 4MHz。SYSOSC 可以在更高的频率下运行以支持 ADC、OPA 或 HS COMP 运行，但 ULPCLK 将被 SYSCTL 自动限制为 4MHz SYSOSC 输出。高速振荡器（SYSPLL、HFXT、HFCLK_IN）被自动禁用。

DMA 可被触发。DMA 触发器可以唤醒 PD1 电源域以使 SRAM 和 DMA 可用于处理 DMA 传输，并以当前 MCLK 和 ULPCLK 速率处理 DMA 传输。传输完成后，SRAM 恢复为保留状态，并会自动禁用 PD1。

STOP 模式是支持 ADC、12 位 DAC 、OPA 和高速 COMP 运行的最低功耗模式。

STOP 模式有三个策略选项：STOP0、STOP1 和 STOP2。

- **STOP0**：当进入 STOP 模式时，SYSOSC 保持在当前频率下运行（32MHz、24MHz、16MHz 或 4MHz）。ULPCLK 始终由硬件自动限制为 4MHz，但 SYSOSC 不会受到干扰，支持 ADC、OPA 或 COMP 等模拟外设以一致的方式运行。
  - 注：如果从 RUN1 进入 STOP0（SYSOSC 启用，但 MCLK 来自 LFCLK），则 SYSOSC 与在 RUN1 中一样保持启用状态，ULPCLK 与在 RUN1 中一样保持在 32kHz。
  - 注：如果从 RUN2 进入 STOP0（SYSOSC 禁用并且 MCLK 来自 LFCLK），则 SYSOSC 与在 RUN2 中一样保持禁用状态，ULPCLK 与在 RUN2 中一样保持在 32kHz。
- **STOP1**：在 SYSOSC 处于运行状态的 STOP 模式下，SYSOSC 会从其当前频率换档至 4MHz 以实现最低功耗。SYSOSC 和 ULPCLK 均以 4MHz 频率运行。
- **STOP2**：SYSOSC 被禁用，ULPCLK 来自 LFCLK，频率为 32kHz。这是 STOP 模式下的最低功耗状态。

2.1.2.4 STANDBY 模式

在 STANDBY 模式下，CPU、SRAM 和 PD1 外设被禁用并保留。除了 ADC、12 位 DAC 和 OPA 外，PD0 外设的最高 ULPCLK 频率为 32kHz。高速振荡器（SYSPLL、HFXT、HFCLK_IN）和 SYSOSC 被禁用。

DMA 可被触发。DMA 触发器可以唤醒 PD1 电源域以使 SRAM 和 DMA 可用于处理 DMA 传输，并以当前 MCLK 和 ULPCLK 速率 (32kHz) 处理 DMA 传输。传输完成后，SRAM 恢复为保留状态，并会自动禁用 PD1。

在 STANDBY 模式下不支持 ADC、12 位 DAC、OPA 和高速 COMP 运行。

STANDBY 模式有 2 个策略选项：STANDBY0 和 STANDBY1。

- **STANDBY0**：所有 PD0 外设都接收 ULPCLK 和 LFCLK，而 RTC 接收 RTCCLK。

- **STANDBY1**：只有 TIMG0 和 TIMG1 接收 ULPCLK 或 LFCLK。RTC 继续接收 RTCCLK。STANDBY1 中的 TIMG0 或 TIMG1 中断、RTC 中断或 ADC 触发器始终会触发异步快速时钟请求以唤醒系统。其他 PD0 外设（例如 UART、I2C、GPIO 和 COMP）也可以在发生外部事件时通过异步快速时钟请求来唤醒系统，但不会在 STANDBY1 中主动为这些外设提供时钟。

2.1.2.5 SHUTDOWN 模式

在 SHUTDOWN 模式下，没有可用的时钟。内核稳压器被完全禁用，并且所有 SRAM 和寄存器内容都丢失，但 SYSCTL 中可用于存储状态信息的 4 字节通用存储器除外。BOR 和带隙电路被禁用。

该器件可通过支持唤醒功能的 IO、调试连接或 NRST 唤醒。

SHUTDOWN 模式的电流消耗是所有工作模式中最低的。退出 SHUTDOWN 模式会触发 BOR。

2.1.2.6 不同工作模式下支持的功能

表 2-1 提供了每种工作模式下支持的功能。

功能键：

- **EN**：该功能会在指定的模式下启用。
- **DIS**：该功能会在指定的模式下被禁用（时钟或电源门控），但该功能的配置会保留。
- **OPT**：该功能在指定的模式下是可选的，如果配置为启用，则保持启用状态。
- **NS**：该功能在指定的模式下不会自动禁用，但不受支持。
- **OFF**：该功能在指定的模式下会完全断电，不会保留任何配置信息。

备注

有关给定器件上每个外设实例的行为的完整列表，请参阅器件特定数据表的详细说明部分中的*不同工作模式下支持的功能*表。

表 2-1. MSPM0Gxx 不同工作模式下支持的功能

工作模式		运行			SLEEP			STOP			STANDBY		关断
		RUN0	RUN1	RUN2	SLEEP0	SLEEP1	SLEEP2	STOP0	STOP1	STOP2	STANDBY0	STANDBY1	
振荡器	SYSOSC	EN	EN	DIS	EN	EN	DIS	OPT[1]	EN	DIS	DIS	DIS	关闭
	LFOSC 或 LFXT	EN（LFOSC 或 LFXT）											关闭
	HFXT	OPT	DIS	DIS	OPT	DIS	DIS	DIS	DIS	DIS	DIS	DIS	关闭
	SYSPLL	OPT	DIS	DIS	OPT	DIS	DIS	DIS	DIS	DIS	DIS	DIS	关闭
时钟	CPUCLK	最大 80M	32k	32k	DIS								关闭
	MCLK 至 PD1	最大 80M	32k	32k	最大 80M	32k	32k	DIS					关闭
	ULPCLK 至 PD0	最大 40M	32k	32k	最大 40M	32k	32k	最大 4M[1]	4M	32k		DIS	关闭
	ULPCLK 至 TIMG0/1	最大 40M	32k	32k	最大 40M	32k	32k	最大 4M[1]	4M	32k			OFF
	RTCCLK	32 kHz											OFF
	MFCLK	OPT	DIS		OPT	DIS		OPT	DIS				关闭
	MFPCLK	OPT	DIS		OPT	DIS		OPT	DIS				关闭
	LFCLK	32k										DIS	关闭
	LFCLK 到 TIMG0/1	32k											OFF
	LFCLK 监视器	OPT											关闭
	MCLK 监测器	OPT										DIS	关闭

表 2-1. MSPM0Gxx 不同工作模式下支持的功能 (continued)

工作模式		运行			SLEEP			STOP			STANDBY		关断
		RUN0	RUN1	RUN2	SLEEP0	SLEEP1	SLEEP2	STOP0	STOP1	STOP2	STANDBY0	STANDBY1	
PMU	POR 监视器	EN											
	BOR 监视器	EN											OFF
	内核稳压器	全驱动						减速驱动			低驱动		关闭
核心功能	CPU	EN			DIS								关闭
	DMA	OPT						DIS (支持的触发器)					关闭
	闪存	EN						DIS					关闭
	SRAM	EN						DIS					关闭
外设	PD1 外设	OPT						DIS					关闭
	PD0 外设	OPT										OPT(2)	OFF
模拟	TRNG	OPT						关闭					
	ADC	OPT									NS (支持的触发器)		关闭
	12 位 DAC	OPT									NS		关闭
	OPA	OPT	NS		OPT	NS		OPT			NS		关闭
	GPAMP	OPT									NS		关闭
	COMP/8 位 DAC	OPT	OPT (仅 ULP)		OPT	OPT (仅 ULP)		OPT			OPT (仅 ULP)		关闭
	VREF	OPT									OPT (采样模式)		关闭
IOMUX 和 IO 唤醒		EN											具有唤醒功能的 DIS
唤醒源		不适用			任何 IRQ			PD0 IRQ					IOMUX、NRST

(1) 如果从 RUN1 进入 STOP0 (SYSOSC 启用，但 MCLK 来自 LFCLK) ，则 SYSOSC 将与在 RUN1 中一样保持启用状态，ULPCLK 将与在 RUN1 中一样保持在 32kHz。如果从 RUN2 进入 STOP0 (SYSOSC 禁用并且 MCLK 来自 LFCLK) ，则 SYSOSC 与在 RUN2 中一样会保持禁用状态，ULPCLK 与在 RUN2 中一样会保持在 32kHz。

(2) 在 STANDBY 模式下使用 STANDBY1 策略时，只有 TIMG0/1 和 RTC 有时钟。其他 PD0 外设可在发生外部活动时生成异步快速时钟请求，但不会主动配备时钟。

2.1.2.7 暂停低功耗模式

某些外设可以配置为暂停 STOP 或 STANDBY 模式，以便处理临时活动或处理事件。可以通过两种方式暂停 STOP 或 STANDBY 模式：

- 异步快速时钟请求
- DMA 触发器

使用异步快速时钟请求暂停 STOP 或 STANDBY 模式

异步快速时钟请求会暂停任何激活的低功耗模式，并以 32MHz 的频率运行 MCLK 和 ULPCLK 树 (来自 SYSOSC) 。如果 MCLK 来自频率为 32kHz 的 LFCLK 或来自频率低于 32MHz 的 SYSOSC，则在 RUN 和 SLEEP 模式下也可以使用异步快速时钟请求。当异步快速时钟请求暂停低功耗模式并更改时钟树配置以支持 32MHz 工作频率时，如果器件处于 STOP 或 STANDBY 模式，这些请求不会启用 PD1 电源域。此功能支持如下用例：

- 从 STANDBY1 中唤醒 RTC 和 TIMG0 或 TIMG1
- 按需 UART、I^2C 或 SPI 通信

- 从 STANDBY 模式进行计时器触发的 ADC 采样

使用 DMA 触发器暂停 STOP 或 STANDBY 模式

如果 DMA 触发器在 STOP 或 STANDBY 模式下生效，该低功耗模式将暂停，并会启用 PD1 电源域（包括 SRAM 和闪存）以处理 DMA 请求。与异步快速时钟请求不同，DMA 传输不会更改时钟树配置。STOP 或 STANDBY 模式下的 DMA 请求会以当前 ULPCLK 速率处理。

2.2 电源管理 (PMU)

电源管理单元 (PMU) 为器件生成稳压内核电源，并对外部电源进行监控。此外，还包含供 PMU 和其他模拟外设使用的带隙电压基准。

PMU 的主要特性包括：

- 支持在宽广电源电压范围（1.62V 至 3.6V）内运行器件
- 通过低压降线性稳压器产生内部内核逻辑电源，具有多种工作模式，可在低功耗模式下减小器件电流（由 SYSCTL 自动管理）
- 上电复位 (POR) 电源监测器
- 欠压复位 (BOR) 电源监测器具有四个可配置的阈值电压
- 带隙电压基准支持 BOR、内核稳压器和模拟外设
- 模拟多路复用器 VBOOST 单元可以提高模拟多路复用器性能
- 温度传感器连接到 ADC

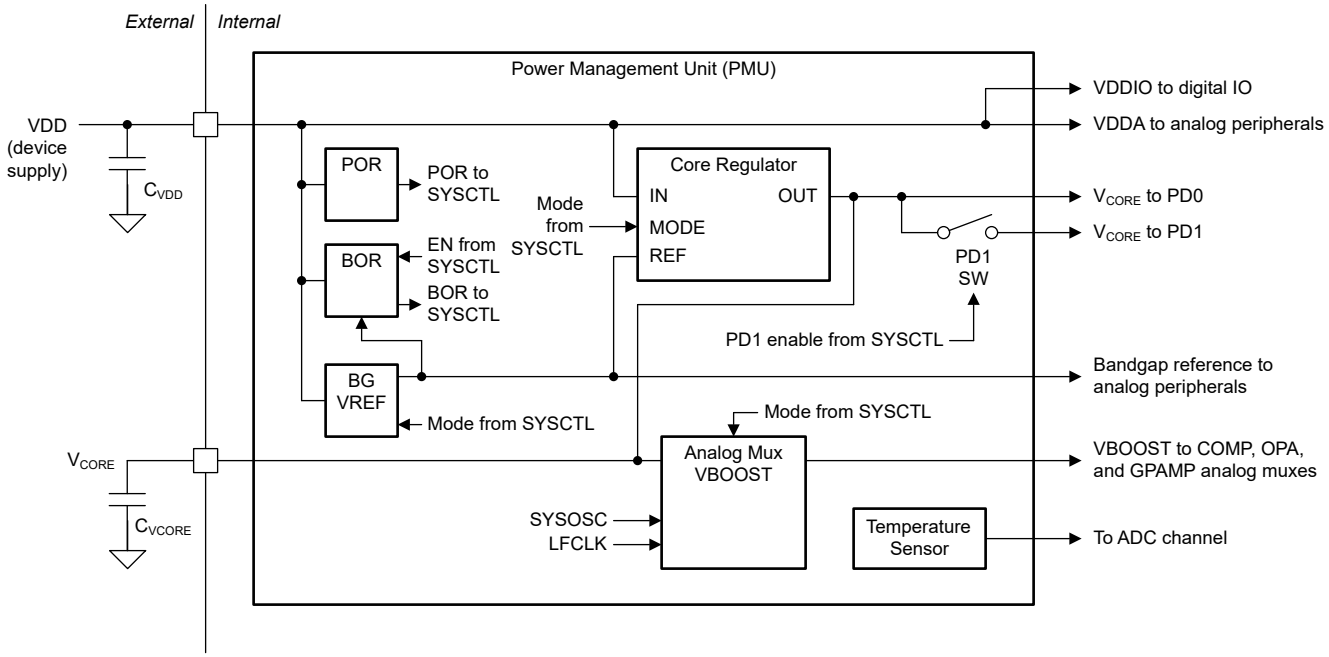


图 2-3. MSPM0Gxx PMU 方框图

2.2.1 电源

该器件由 VDD 和 VSS 连接进行供电。该器件支持在 1.62V 至 3.6V 的电源电压下运行，并会以 1.62V 电源电压启动。必须在所有 VDD 和 VSS 电源对之间放置一个去耦电容器 (C_{VDD})。有关 C_{VDD} 的正确值和容差，请参阅器件特定的数据表。具有 64 个或更少引脚的产品通常具有单个 VDD/VSS 电源对。

VDD 直接用于提供 IO 电源 (VDDIO) 和模拟电源 (VDDA)。VDDIO 和 VDDA 在内部连接到 VDD，因此无需额外的电源引脚。

2.2.2 内核稳压器

PMU 使用片上可配置低压降线性稳压器生成 1.35V 电源轨，为器件内核供电。通常，内核稳压器输出 (V_{CORE}) 为内核逻辑（包括 CPU、数字外设和器件存储器）供电。内核稳压器需要一个连接在器件 VCORE 引脚和 VSS（接地）之间的外部电容器 (C_{VCORE})。有关 C_{VCORE} 的正确值和容差，请参阅器件特定的数据表。

在除 SHUTDOWN 外的所有电源模式中，内核稳压器均处于运行状态。在所有其他功耗模式（RUN、SLEEP、STOP 和 STANDBY）中，稳压器的驱动强度会自动配置为支持每种模式的最大负载电流。这降低了使用低功耗模式时稳压器的静态电流，从而提高了低功耗性能。SYSCTL 会根据当前处于运行状态的电源模式自动配置稳压器以实现理想功耗。

2.2.3 电源监控器

PMU 提供两个电源监控器电路：

- 一个上电复位 (POR) 电路，用于指示外部电源已达到足够的电压来启动片上带隙基准和 BOR 电路
- 一个用户可编程的欠压复位 (BOR) 电路，用于确保外部电源保持足够的电压来支持器件的正确运行

2.2.3.1 上电复位 (POR) 监控器

上电复位 (POR) 监控器会监视外部电源 (VDD) 并为 SYSCTL 将 POR 违例设置为有效或使其无效。在冷上电期间，器件保持为 POR 状态，直到 VDD 超过 POR+ 阈值。一旦 VDD 超过 POR+，便会释放 POR 状态，并会启动带隙基准和 BOR 监测器电路。如果 VDD 降至 POR- 电平以下，则会发生 POR- 违例，并且器件再次保持为 POR 复位状态。

POR 监控器不会指示 VDD 已达到足以支持器件正确运行的电平，而是作为引导过程的第一步用于确定电源电压是否足以为带隙基准和 BOR 电路上电，然后再使用带隙基准和 BOR 电路来确定电源是否达到足以使器件正确运行的电平。

POR 监控器在包括 SHUTDOWN 模式在内的所有功耗模式下均处于活动状态，无法禁用。

2.2.3.2 欠压复位 (BOR) 监控器

欠压复位 (BOR) 监控器会监控外部电源 (VDD) 并使 SYSCTL 的 BOR 违例有效或无效。BOR 电路的主要作用是确保外部电源保持足够高的电压，以使包括内核稳压器在内的内部电路能够正常运行。BOR 阈值基准来自内部带隙电路。该阈值本身可编程，并且始终高于 POR 阈值。在冷启动期间，在 VDD 超过 POR+ 阈值后，带隙基准和 BOR 电路被启动。然后，器件保持在 BOR 状态，直到 VDD 超过 BOR0+ 阈值。一旦 VDD 通过 BOR0+，BOR 监控器便会释放器件以继续执行引导过程，并启动 PMU。

有四个可选的 BOR 阈值电平 (BOR0-BOR3)。在启动期间，BOR 阈值始终为 BOR0（最低值），以确保器件始终以指定的 VDD 最小值 (1.62V) 启动。启动后，软件可以选择性地重新配置 BOR 电路以使用不同（更高）的阈值电平 (BOR1-BOR3)。

当 BOR 阈值为 BOR0 时，BOR0 违例始终会向 SYSCTL 生成 BOR- 违例信号，从而生成 BOR 电平复位。当 BOR 阈值被重新配置为 BOR1、BOR2 或 BOR3 时，BOR 电路会生成一个 SYSCTL 中断，而不是使 BOR- 违例有效。这可用于向应用指示电源已降至某个电平以下，而不会导致复位。

要将 BOR 电平从默认值 (BOR0) 更改为其他值，首先在 SYSCTL 中 BORTHRESHOLD 寄存器的 LEVEL 字段中选择所需的值。然后，通过设置 BORCLRCMD 寄存器的 GO 位，激活在 LEVEL 字段中设置的阈值。可以通过测试 SYSSTATUS 寄存器中的 BORCURTHRESOLD 字段来验证该变化，该字段会返回与当前有效 BOR 阈值相对应的值。BOR 阈值变化大约需要 15 µs 才能完成，在此期间 BOR 电路对电源变化视而不见。

如果 BOR 处于中断模式（阈值电平为 BOR1-3）且电源电压降至相应的 BORx- 电平以下，则会生成中断，BOR 电路会自动将 BOR 阈值电平切换为 BOR0，以确保在 VDD 降至 BOR0- 以下时 BOR 违例有效。应用软件可以再次设置 BORCLRCMD 寄存器中的 GO 位，将 BOR 电平设置回 BORTHRESHOLD 寄存器 LEVEL 字段指定的电平。

BOR 监控器在 RUN、SLEEP、STOP 和 STANDBY 模式下处于运行状态，但在 SHUTDOWN 模式下会自动禁用。

2.2.3.3 电源变化期间的 POR 和 BOR 行为

当电源电压 (VDD) 降至 POR- 以下时，将清除整个器件状态。未降至 BOR0- 阈值以下的 VDD 微小变化不会导致 BOR- 违例，此时器件会继续运行。除了 BOR0 以外的 BORx 阈值（例如，BOR1-BOR3）的行为与 BOR0 的行为相同，但 BOR 电路配置为产生中断，而不是立即触发 BOR 复位。

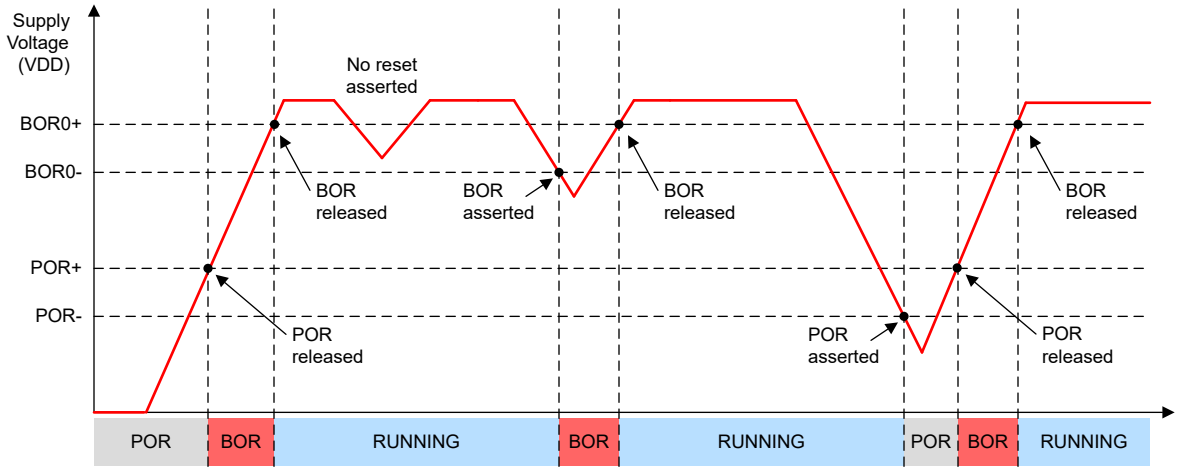


图 2-4. POR/BOR 与电源电压 (VDD) 间的关系

2.2.4 带隙基准

PMU 提供一个温度和电源电压稳定的带隙电压基准，此基准被器件用于内部功能，其中包括：

- 产生欠压复位电路阈值
- 设置内核稳压器的输出电压
- 推导片上模拟外设的片上 VREF 电平

带隙基准在 RUN、SLEEP、STOP 模式下启用。该基准在 STANDBY 模式下以采样模式运行，以降低功耗；在 SHUTDOWN 模式下被禁用。SYSCTL 会自动管理带隙状态；无需用户配置。

2.2.5 温度传感器

PMU 提供了一个可用于测量器件温度的基本温度传感器。温度传感器在内部连接到 ADC，必须使用 ADC 来执行温度测量。请参阅器件特定的数据表，确定在温度传感器测量时使用的正确内部 ADC 通道。

温度传感器输出一个与温度成线性关系的电压。温度系数 (TS_c) 是温度与电压关系的斜率（以 mV/C 为单位），在器件特定数据表的规格部分中给出。

每个器件的出厂常量存储器中提供了特定于单位的单点修整值 (TEMP_SENSE0.DATA)。该值表示出厂修整温度 (TS_{TRIM}) 下的温度传感器输出电压，采用 ADC 结果代码格式。TEMP_SENSE0.DATA 中的 ADC 结果代码基于 12 位采样模式以及 1.4V 内部电压基准。TS_{TRIM} 温度也在器件特定数据表的规格部分中给出。

可以使用以下参数计算器件的近似温度：

- TS_c，取自器件数据表
- TEMP_SENSE0.DATA，取自特定于单位的出厂常量存储器
- TS_{TRIM}，取自器件数据表
- V_{SAMPLE}（温度传感器在所需时间的电压样本，由 ADC 采集）

可通过方程式 1 中的线性关系计算温度，其中 V_{SAMPLE} 为当前温度传感器电压，V_{TRIM} 为 TS_{TRIM} 温度（源自 TEMP_SENSE0.DATA）下的工厂校准温度传感器电压：

$$T_{SAMPLE} = (1 / TS_c) * (V_{SAMPLE} - V_{TRIM}) + TS_{TRIM} \quad (1)$$

ADC_{CODE} 原始结果可以转换为等效电压 (V_{SAMPLE})，如方程式 2 中的关系所示，其中的 RES 表示以位为单位的 ADC 分辨率，VREF 表示 ADC 基准电压。

$$\mathbf{V_{SAMPLE}} = (VREF / 2^{RES}) * (ADC_{CODE} - 0.5) \quad (2)$$

示例

为了说明将温度传感器的 ADC 样本转换为某个近似器件温度的过程，下面给出了一个示例。

示例参数：

- TS_c = -2.04mV/C
- TEMP_SENSE0.DATA = 1857（基于 12 位模式和 1.4V 基准的 ADC 结果代码）
- TS_{TRIM} = 30C
- ADC_{CODE} = 1677（基于 12 位模式和 1.4V 基准的 ADC 结果代码）

首先，使用方程式 3 计算当前温度传感器样本电压：

$$\mathbf{V_{SAMPLE}} = (1.4\ V / 4096) * (1677 - 0.5) = \mathbf{0.5730\ V} \quad (3)$$

然后，使用相同的方法计算单点校准电压：

$$\mathbf{V_{TRIM}} = (1.4\ V / 4096) * (1857 - 0.5) = \mathbf{0.6345\ V} \quad (4)$$

然后，使用方程式 5 计算近似的温度：

$$\mathbf{T_{SAMPLE}} = (1 / -0.002044) * (0.5730\ V - 0.6345) + 30°C = \mathbf{60°C} \quad (5)$$

备注

在 STANDBY 和 SHUTDOWN 工作模式下，温度传感器不可用。

2.2.6 用于模拟多路复用器的 VBOOST

PMU 中的 VBOOST 电路会生成内部 VBOOST 电源，供器件上 COMP、GPAMP 和 OPA（如有）中的模拟多路复用器使用。VBOOST 电路可在外部电源电压 (VDD) 范围内实现一致的模拟多路复用器性能。

启用和禁用 VBOOST

SYSCTL 会根据以下参数自动管理 VBOOST 电路的使能请求：

- COMP、OPA 和 GPAMP 外设 PWREN 设置
- 任何 COMP 已启用的模式设置（FAST 与 ULP 模式）
- SYSCTL 中 GENCLKCFG 寄存器的 ANACPUMPCFG 控制位

在 SYSRST 之后，VBOOST 默认被禁用。在使用 COMP、OPA 或 GPAMP 之前，无需应用软件即可启用 VBOOST 电路。当 COMP、OPA 或 GPAMP 由应用软件启用时，SYSCTL 还会使 VBOOST 电路支持模拟外设。

VBOOST 电路具有从禁用状态转换到启用状态的启动时间要求（典型值为 12 µs）。如果在禁用 VBOOST 后启用 COMP、OPA 或 GPAMP，则在外设和 VBOOST 电路均就绪之前，相应的模拟外设就绪状态不会有效。如果 COMP、OPA 或 GPAMP 的启动时间小于 VBOOST 启动时间，则会延长外设启动时间以计入 VBOOST 启动时间。

或者，应用软件可以强制 VBOOST 电路始终启用 (ANACPUMPCFG=0x2) 或处于 RUN 或 SLEEP 模式 (ANACPUMPCFG=0x1)，以便在启用 COMP、OPA 或 GPAMP 时不会有额外的启动延迟。表 2-2 给出了 ANACPUMPCFG 控制和相应应用用例的行为。

表 2-2. 使用 ANACPUMPCFG 强制启用 VBOOST

VBOOST 控制 (ANACPUMPCFG)		行为	
值	模式	VBOOST 启用	应用用例
0x0	ONDEMAND	仅当启用 COMP、OPA 或 GPAMP 时，SYSCTL 才会自动启用 VBOOST。	当 COMP、OPA 或 GPAMP 的快速启动不重要时，该设置可在所有模式下提供最低功耗。
0x1	ONACTIVE	当器件处于 RUN 或 SLEEP 模式时，强制启用 VBOOST。如果启用了 COMP、OPA 或 GPAMP，VBOOST 也会在 STOP 或 STANDBY 模式下保持启用。	通过在 STOP 和 STANDBY 模式下自动禁用 VBOOST（当不启用需要 VBOOST 的外设时），此设置可提供低功耗。当应用软件在 RUN 模式下启用 COMP、OPA 或 GPAMP 时，VBOOST 会在退出 RUN 模式时自动重新启用，从而快速启动 COMP、OPA 或 GPAMP。
0x2	ONALWAYS	在除 SHUTDOWN 外的所有工作模式中，VBOOST 均被强制启用。	此设置可确保在快速 COMP、OPA 或 GPAMP 启动至关重要的应用中，COMP、OPA 和 GPAMP 绝不会由于 VBOOST 启动而产生额外的启动延迟。

图 2-5 展示了 VBOOST 使能、VBOOST 时钟选择和 VBOOST 时钟错误逻辑。

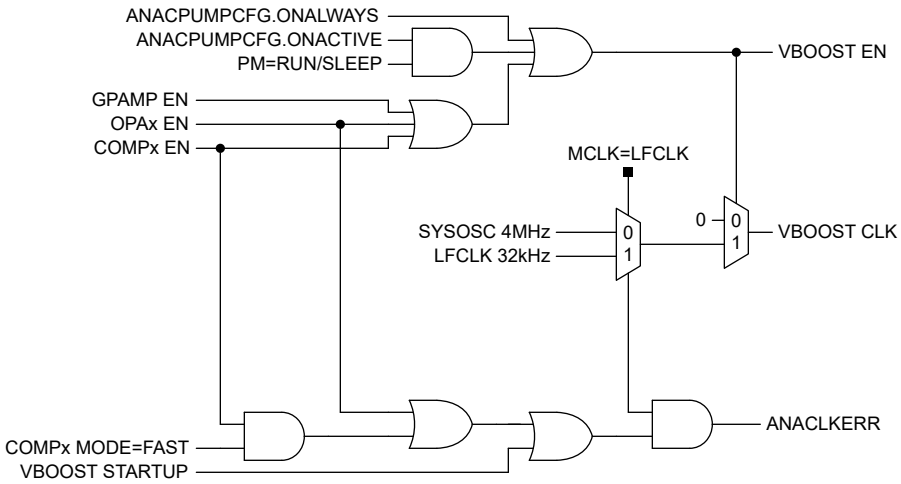


图 2-5. VBOOST 请求逻辑

VBOOST 时钟

VBOOST 电路需要一个功能时钟才能运行。VBOOST 根据当前使用的 MCLK 和 ULPCLK 树源通过 SYSOSC（4MHz 输出）或 LFCLK (32kHz) 计时。VBOOST 时钟由 SYSCTL 根据图 2-5 中显示的逻辑自动选择，并在此处进行介绍：

- 当 MCLK 和 ULPCLK 源自 LFCLK (32kHz) 时，VBOOST 也源自 32kHz LFCLK。
- 当 MCLK 和 ULPCLK 使用任何其他时钟源（例如，并非 LFCLK）时，VBOOST 使用 SYSOSC 4MHz 输出作为时钟源。根据定义，当 MCLK 和 ULPCLK 不使用 LFCLK 源时，SYSOSC 将一直启用。

某些 VBOOST 工作条件要求 VBOOST 时钟为 4MHz（源自 SYSOSC），而不是 32kHz（源自 LFCLK）。这类条件包括：

- OPA 运行或快速模式 COMP 运行
- VBOOST 正在启动（从禁用状态转换到启用状态）

应用软件必须确保当其中任一条件出现时，MCLK 和 ULPCLK 源自非 LFCLK。SYSCTL 不会更改 VBOOST 请求的当前系统时钟配置。在其中一种情况下，如果 MCLK 和 ULPCLK 树源自 LFCLK，SYSCTL 会使 SYSCTL 中 SYSSTATUS 寄存器的 ANACLKERR 状态有效，以向应用软件指示 VBOOST 时钟要求与当前 MCLK 和 ULPCLK 配置不匹配。

表 2-3 中给出了 VBOOST 的完整时钟要求。

表 2-3. VBOOST 时钟要求

VBOOST 请求		VBOOST 电路时钟要求	MCLK 和 ULPCLK 源要求(1)	支持的工作模式或策略
无有效请求（禁用 VBOOST）		不适用	不用考虑	不用考虑
VBOOST 从禁用状态启动		4MHz	不是 LFCLK	RUN0、SLEEP0、STOP0、STOP1
启用 OPA		4MHz	不是 LFCLK	RUN0、SLEEP0、STOP0、STOP1
启用 COMP	快速模式 (FAST)	4MHz	不是 LFCLK	RUN0、SLEEP0、STOP0、STOP1
	超低功耗模式 (ULP)(2)	4MHz 或 32kHz	不用考虑	RUN0、RUN1、RUN2、SLEEP0、SLEEP1、SLEEP2、STOP0、STOP1、STOP2、STANDBY0
GPAMP 使能(2)		4MHz 或 32kHz	不用考虑	RUN0、RUN1、RUN2、SLEEP0、SLEEP1、SLEEP2、STOP0、STOP1、STOP2、STANDBY0
在外设请求无效（由于 ANACPUMPCFG!=0x0）的情况下运行的 VBOOST(2)		4MHz 或 32kHz	不用考虑	RUN0、RUN1、RUN2、SLEEP0、SLEEP1、SLEEP2、STOP0、STOP1、STOP2、STANDBY0

(1) 应用软件必须确保按照上表中所述配置时钟系统，以支持每种情况都能实现正确的 VBOOST 运行。SYSCTL **不会在请求 VBOOST 时更改系统时钟配置**。如果 VBOOST 请求要求 VBOOST 时钟为 4MHz 且 MCLK 源自 LFCLK，SYSCTL 将使 ANACLKERR 有效，以向应用程序指示当前的系统时钟配置不能支持当前的 VBOOST 请求。

(2) 如果未启用 OPA 且未启用 COMP 并在 FAST 模式下配置，则 VBOOST **不需要 4MHz 时钟**并且 VBOOST 可以从 LFCLK 运行以降低功耗。但是，如果 VBOOST 先前被禁用（不存在请求）并且被启用的外设或应用软件请求 (ANACPUMPCFG!=0x0)，**VBOOST 启动需要 4MHz VBOOST 时钟**。VBOOST 启动完成后，如果在 FAST 模式下未启用 OPA 和 COMP，则 VBOOST 时钟可以是 32kHz，源自 LFCLK。在 VBOOST 启动期间将 MCLK 源更改为 LFCLK 会导致 VBOOST 错误运行和不一致的模拟外设性能。当 VBOOST 电路启动并准备就绪时，SYSCTL 中 SYSSTATUS 寄存器的 ANACPUMPGOOD 位被置位。

2.2.7 外设电源使能控制

除 SYSCTL 本身和 IOMUX 等基础结构外设外，器件上的所有外设均包含一个带有 KEY 和 ENABLE 字段的电源使能控制寄存器 (PWREN)。在由软件配置任何其他外设寄存器之前，必须通过将 ENABLE 位连同相应的 KEY 值一起写入外设的 PWREN 寄存器来启用外设本身。

当清除外设 ENABLE 位后，外设可被视为非活动状态，其余特定于外设的寄存器与外设总线相隔离，因此无法进行读取/写入操作。

备注

在设置 PWREN 寄存器中的 ENABLE | KEY 位以启用外设之后，至少等待 4 个 ULPCLK 时钟周期，然后再访问外设存储器映射寄存器的其余部分。在这 4 个周期内，外设总线接口上的总线隔离信号能够得到更新。

2.2.7.1 低功耗模式下自动禁用外设

进入 STOP 或 STANDBY 低功耗模式时，SYSCTL 将强制电源域 1 (PD1) 中的外设进入禁用状态。因此，这些外设在 STOP 或 STANDBY 模式下将无法使用。

大多数 PD1 外设在自动禁用后将保留其配置设置，这样在退出 STOP 或 STANDBY 模式时无需重新配置。有关哪些外设寄存器在 PD1 外设的 STOP 和 STANDBY 模式下被保留的详细信息，请参阅本指南中的外设专用章节。

如果在输出配置中将 PD1 外设多路复用到 IO 引脚（通过 IOMUX），则在进入 STOP 或 STANDBY 模式时，从外设到 IO 的最后一个有效输出状态（逻辑 0 或逻辑 1）将被锁存。这可防止在低功耗运行期间禁用外设时 SYSCTL 干扰外部电路。从 STOP 或 STANDBY 模式退出后，IO 再次连接到外设，因为外设重新启用。

2.3 时钟模块 (CKM)

时钟模块包含内部和外部 振荡器、时钟监测器以及时钟选择和控制逻辑。还提供了频率时钟计数器，用于根据 LFXT/LFCLK_IN 或 IO 引脚上提供的基准周期/脉冲来检查和/或校准高速时钟的频率。

2.3.1 振荡器

有一些内部和外部振荡器可以生成供系统使用的低频至高频时钟。CKM 包含器件中的所有振荡器，并使用它们来生成系统时钟。

内部振荡器

- LFOSC：低频振荡器（典型频率为 32kHz）
- SYSOSC：系统振荡器（出厂修整频率为 4MHz 或 32MHz，用户修整频率为 16MHz 或 24MHz）
- SYSPLL：具有可编程频率的系统 PLL

外部振荡器

- LFXT：低频、低功耗晶体振荡器（典型频率为 32kHz）
- HFXT：高频晶体振荡器（典型频率为 4-48MHz）

除了振荡器之外，还提供低频 (LFCLK_IN) 和高频 (HFCLK_IN) 数字时钟输入，以支持未使用 LFXT 或 HFXT 并且向器件提供直接数字时钟的情况。

2.3.1.1 内部低频振荡器 (LFOSC)

低频振荡器 (LFOSC) 是一种片上低功耗振荡器，出厂时频率已调整为 32.768kHz。在低频晶体振荡器 (LFXT) 不存在或未使用的情况下，LFOSC 可提供稳定的低频时钟源。

当在较低的温度范围内使用时，LFOSC 可提供更高的精度。相关详细信息，请参阅器件特定数据表。

在 BOOTRST 之后，LFOSC 默认处于活动状态，并作为 LFCLK 的时钟源。当 LFOSC 就绪时，LFOSC 启动监视器设置 CLKSTATUS 寄存器中的 LFOSCGOOD 位。

如果用户软件选择 LFXT 或 LFCLK_IN 作为 LFCLK 源，则会禁用 LFOSC，直至执行下一次 BOOTRST。

2.3.1.2 内部系统振荡器 (SYSOSC)

系统振荡器 (SYSOSC) 是一款精确且可配置的片上振荡器，其出厂修整频率为 32MHz（基础频率）和 4MHz（低频），支持用户修整的 24MHz 或 16MHz 工作频率。在不存在或未使用高频晶体振荡器 (HFXT) 或需要从低功耗模式快速唤醒的情况下，SYSOSC 可为系统提供灵活的高速时钟源。

SYSOSC 的主要特性包括：

- 使用可选频率校正环路 (FCL) 和基准电阻器时具有高精度
  - 频率校正环路可支持通过外部电阻器 (ROSC) 或内部电阻器进行校正，具体取决于器件功能。请参阅器件特定数据表，确定器件是否支持具有内部和/或外部电阻器的 FCL
- 从禁用状态快速启动
- 能够在一个时钟周期内从基础频率切换到低频或从低频切换到基础频率，而不会发生功能中断（换档）
  - 采用相位对齐的转换以尽可能减少对外设的干扰
  - 快速稳定至指定的精度
  - SYSCTL 可以在 STOP 模式下启动无缝换档频率切换以降低 SYSOSC 电流
- 次级输出包括可供 MFCLK 和 MFPCLK 使用的恒定 4MHz 频率
  - 当 f_{SYSOSC} = 32MHz 时，4MHz 输出是 SYSOSC 的 8 分频数字分频结果。当 f_{SYSOSC} = 4MHz 时，4MHz 输出直接来自 SYSOSC。24MHz 和 16MHz 的用户频率分别通过 /6 和 /4 数字分频器获得。SYSCTL 管理该输出端的数字分频器，以确保无论所选的 SYSOSC 频率如何，均可获得恒定的 4MHz 输出。
  - 通过 MFCLK 或 MFPCLK 使用此输出的外设在 RUN、SLEEP 和 STOP 模式下会看到一个连续的 4MHz 功能时钟，但在 STOP 模式下，启用换档后，由于 SYSOSC 以自身的 4MHz 频率运行而非对 32MHz 进行 8 分频，因此电流会降低。

默认情况下在欠压复位后，SYSOSC 在基础频率 (32MHz) 下处于活动状态，从而提供 MCLK。

2.3.1.2.1 SYSOSC 换档

SYSOSC 可以在单个 SYSOSC 时钟周期内从基础频率 (32MHz) 切换到低频 (4MHz)，或从低频切换回基础频率，为尽可能减少抖动，频率变化为相位对齐。此特性特别适合低功耗、低成本的微控制器应用，因为它允许使用单个振荡器产生两个频率：

- 高速 32MHz 时钟（用于在计算期间运行 CPU），仅在需要时启用
- 低速 4MHz 时钟（用于运行计时器、UART 接口和 I^2C 接口等外设），持续运行

当需要高速时钟时（例如，在 RUN 和 SLEEP 模式下），SYSOSC 可以在其基础频率下运行以提供高性能。在这种情况下，可以直接使用 SYSOSC 来提供 MCLK，以便 CPU、DMA 和由总线提供时钟的外设均可使用 MCLK。对于连续运行但不需要高时钟频率的外设，则通过采用 32MHz SYSOSC 并进行 8 分频的方式来提供 4MHz 的中频系统时钟 (MFCLK)。

当不需要 32MHz 的快速时钟时（例如，如果 CPU 已完成数据处理），则在 STOP 模式下，SYSOSC 可在一个周期内降档至 4MHz，而 MFCLK 会切换为直接由 SYSOSC 提供时钟，因此会保持持续运行的 4MHz 时钟，但会降低 SYSOSC 电流，并且无需使用单独的振荡器来提供 4MHz 时钟。当再次需要 CPU 时，可以将 SYSOSC 从 4MHz 切换回 32MHz。此转换也在一个 SYSOSC 时钟周期内发生。在升档过程中，MFCLK 切换回 SYSOSC/8 以保持 4MHz 恒定频率。降档请求期间的基础频率 (32MHz) 与低频 (4MHz) 之间的时序转换如图 2-6 所示。

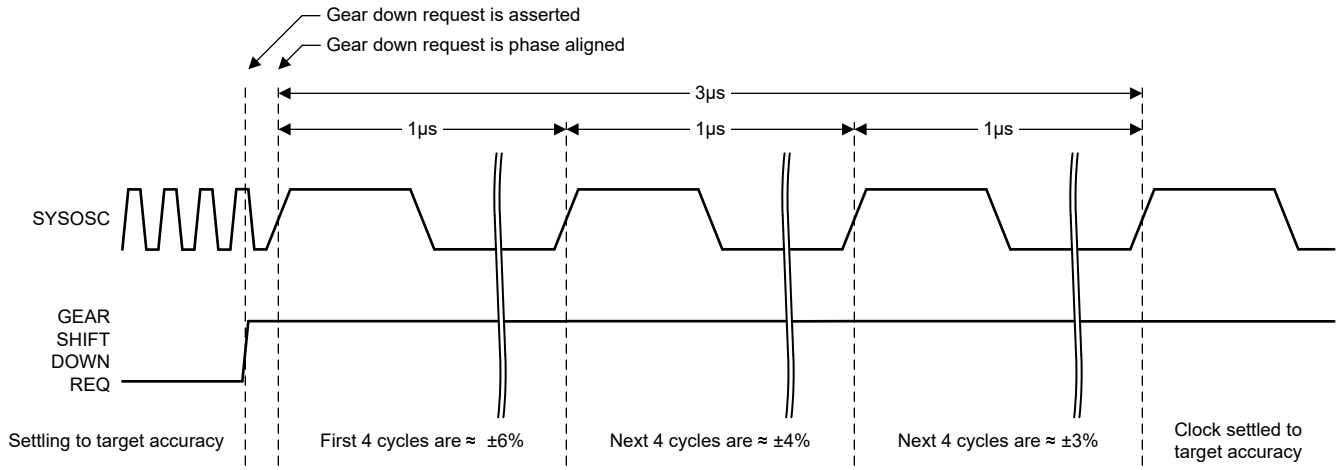


图 2-6. SYSOSC 降档时序

SYSOSC 换档模式与电源管理方案紧密结合，并由 SYSCTL 控制。换档模式可用于在 RUN 模式和 STOP 模式之间（降档）以及 STOP 模式和 RUN 模式之间（升档）切换。要在 STOP 模式下使用换档来降低 SYSOSC 电流，请设置 SYSOSCCFG 寄存器中的 USE4MHZSTOP 位。请注意，如果 SYSOSC 配置为在 RUN 和 SLEEP 模式以及 STOP 模式（无换档）下以 4MHz 的频率运行，则无需设置 USE4MHZSTOP 位即可在进入 STOP 模式时将 SYSOSC 保持在 4MHz。

2.3.1.2.2 SYSOSC 频率和用户修整

SYSOSC 在出厂时经过修整，可在 32MHz 或 4MHz 频率下运行。默认工作频率为基础频率 (32MHz)。如果 4MHz 的频率能为应用提供合适的性能，则可以在 4MHz 下持续运行 SYSOSC 以减小工作电流。要将 SYSOSC 频率更改为 4MHz，请将 SYSOSCCFG 寄存器中的 FREQ 字段设置为 4M。

备注

SYSOSCCFG 寄存器中的 FREQ 字段只能在 SYSOSC 为 MCLK 源时才可更改。当 MCLK 来自 LFCLK 或来自 HSCLK 时，请勿更改 SYSOSC FREQ 配置。

SYSOSC 还支持用户修整的 24MHz 或 16MHz 工作频率。不允许使用其他频率。要以用户修整的频率运行 SYSOSC，首先应确保 SYSOSC 以基础频率 (32MHz) 运行。然后，针对目标频率配置 SYSOSCTRIMUSER 寄存器。最后，在 SYSOSCCFG 寄存器的 FREQ 字段中选择 USER。请参阅修整过程以了解修整。

备注

以用户频率（16MHz 或 24MHz）运行时，将 SYSOSC 频率的选择更改为低频 (4MHz) 需要在 SYSOSC 切换到低频之前通过基础频率 (32MHz) 进行过渡。从低频切换到用户频率时也是如此。此更改完全在硬件中进行管理。以用户频率运行时，软件可以直接选择低频（反之亦然），但在更改频率时，硬件会通过基础频率进行过渡。如果启用了 MFCLK (4MHz)，在过渡期间将保持在 4MHz。

2.3.1.2.3 SYSOSC 频率校正环路

通过使用 SYSOSC 频率校正环路 (FCL) 功能可以提高 SYSOSC 频率精度。FCL 电路利用一个组装在 ROSC 引脚和 VSS 之间的内部电阻或外部电阻，通过为 SYSOSC 提供精密基准电流来稳定 SYSOSC 频率。可实现的总体频率精度取决于工作温度范围以及所选基准电阻器的容差和温度漂移。

备注

并非所有器件都支持使用内部电阻和外部电阻实现 FCL 运行。有些器件支持外部模式，有些器件支持内部模式，有些器件同时支持这两种模式。请参阅器件特定的数据表，确定给定器件的特性。

备注

由于 ROSC 中有基准电流，因此启用 FCL 后 SYSOSC 的功耗略微升高。从启动到获得目标精度的稳定时间也可能更长。请参阅器件特定的数据表以了解启动时间。

2.3.1.2.3.1 外部电阻器模式下的 SYSOSC FCL (ROSC)

本节介绍了选择外部 ROSC 电阻器的过程（包括计算可实现的 SYSOSC 精度）以及通过器件 SYSCTL 寄存器启用模式的过程。

使用 TI 推荐的 ROSC 电阻器时的 SYSOSC 总体频率精度

器件特定数据表包含将 TI 推荐的 ±0.1% 容差、±25ppm/°C、TCR 电阻器用于 ROSC 时，总体 SYSOSC 频率精度的规格。TI 推荐的电阻器支持多种应用，包括 UART 通信。如果包含 TI 推荐电阻器规格的器件特定数据表中的总体精度值满足应用和成本要求，则无需计算替代电阻器或替代温度范围的精度。

但是，如果需要更高精度或更低成本的电阻器，设计人员可以将更高或更低精度的电阻器用于 ROSC，以便在给定应用场景中平衡成本和频率精度。

使用设计人员选择的电阻器时的 SYSOSC 总体频率精度

假定一个理想（零误差、零温度漂移）基准电阻器，FCL 模式将 SYSOSC 电路精度提高到 f_{SYSOSC} 中指定的水平：器件特定数据表中 SYSOSC 规格表的 *“启用频率校正环路 (FCL) 并假设使用理想 ROSC 电阻器时的 SYSOSC 频率精度”* 部分。通过组合以下误差源来确定总频率误差，可确定 FCL 模式下可在给定外部电阻器和温度范围内实现的总体 SYSOSC 频率精度：

1. ROSC 电阻器容差（25°C 时）；这来自电阻器规格
2. ROSC 电阻器电阻温度系数 (TCR)；这来自电阻器规格
3. 所设计系统的最小和最大预期工作温度
4. 在所需温度范围内使用理想电阻器时的 SYSOSC 频率精度（摘自 MSPM0 器件特定数据表）

下表提供了一个示例，说明如何在以下典型应用场景中计算不同条件下的精度：

- 目标工作温度范围为 -40°C 至 85°C。
- 容差为 ±0.1% 并选择了 ±25ppm/°C TCR 的 ROSC 电阻器

表 2-4. 根据外部电阻器 (ROSC) 规格计算总体 SYSOSC FCL 频率精度

步进	说明	最小值	典型值	最大值	单位
1	定义应用的工作温度范围。	-40	25	85	℃

表 2-4. 根据外部电阻器 (ROSC) 规格计算总体 SYSOSC FCL 频率精度 (continued)

步进	说明	最小值	典型值	最大值	单位
2	选择目标 f_{SYSOSC} 频率（4MHz 或 32MHz）		32.00		MHz
3	根据步骤 1 中的温度范围，从器件特定数据表中查找相应的 f_{SYSOSC} 理想电阻器精度。	-0.80		0.93	%
4	根据步骤 2 中选择的频率和步骤 3 中的精度范围，计算使用理想电阻器时的 f_{SYSOSC} 最小值和最大值。	31.744		32.298	MHz
5	选择容差可接受的 ROSC 电阻器（本示例中使用的 TI 推荐的 ±0.1% 容差）	99.9	100	100.1	kΩ
6	记下 ROSC 电阻器的电阻温度系数 (TCR)。		±25		ppm/℃
7	根据步骤 1 中定义的温度范围，计算 25°C（标称值）时的最大温度变化。		| 25 - (-40) | = 65		℃
8	计算最大温度变化范围（步骤 7）和 TCR（步骤 6）下的最大 ROSC 漂移。	-0.16		0.16	%
9	计算最小和最大 ROSC 电阻，包括容差（步骤 5）和温度漂移（步骤 8）。	99.74		100.26	kΩ
10	使用步骤 9 中的总体 ROSC 电阻范围与标称电阻值，计算 ROSC 电阻器的总体精度（以百分比为单位）。	-0.26		0.26	%
11	通过按步骤 10 中计算出的 ROSC 精度降低步骤 4 中计算出的最小/最大 f_{SYSOSC}，计算最小和最大 f_{SYSOSC} 的变化。	31.66		32.38	MHz
12	如果需要，将步骤 11 中计算出的原始最小/最大 f_{SYSOSC} 转换为相对于步骤 2 中目标 f_{SYSOSC} 的百分比精度格式。	-1.06		1.20	%

启用具有外部电阻器 (ROSC) 的 FCL

要提高使用 FCL 时的 SYSOC 精度，请按以下过程操作：

1. 验证 ROSC 引脚上的所有数字功能是否在 IOMUX 中被禁用（这是上电后的默认状态）。
2. 验证是否在 ROSC 引脚与器件接地端 (VSS) 之间放置了一个满足应用精度要求的 ROSC 基准电阻。
3. 通过设置 SYSOSCFCLCTL 寄存器中的 SETUSEFCL 位即可启用 FCL 模式。
   a. 如果器件同时支持内部电阻器和外部电阻器 FCL 模式，那么除了设置 SETUSEFCL 位外，还要设置 SYSOSCFCLCTL 寄存器中的 SETUSEEXRES 位。
4. 启用 FCL 模式后，软件无法禁用该模式。在更改为 FCL 模式之前，需要 BOOTRST。

2.3.1.2.3.2 内部电阻模式下的 SYSOSC FCL

本节介绍了通过器件 SYSCTL 寄存器选择内部电阻器模式的过程。

对于支持内部电阻器 FCL 模式的器件，器件特定数据表包含了各种温度范围内 SYSOSC 总体频率精度的规格。如果器件特定数据表中的总体精度值满足应用和成本要求，则无需使用外部电阻器模式，并且 ROSC 引脚可用于标准功能。

启用具有内部电阻器的 FCL

要提高使用 FCL 时的 SYSOC 精度，请按以下过程操作：

1. 通过设置 SYSOSCFCLCTL 寄存器中的 SETUSEFCL 位即可启用 FCL 模式。
   a. 如果器件同时支持内部电阻器和外部电阻器 FCL 模式，则在设置 SETUSEFCL 位时不要设置 SYSOSCFCLCTL 寄存器中的 SETUSEEXRES 位。
2. 启用 FCL 模式后，软件无法禁用该模式。在更改为 FCL 模式之前，需要 BOOTRST。

2.3.1.2.4 SYSOSC 用户修整过程

如果需要，可以将 SYSOSC 修整为 16MHz 或 24MHz 的用户修整值。通过操作 SYSOSCTRIMUSER 中的字段来实现 16MHz 或 24MHz 的目标频率即可完成修整。要修整 SYSOSC，可以使用 CLK_OUT 单元将 SYSOC 引出至外部引脚，以便对其进行测量。请按照以下过程获得用户修整值。获得修整值后，可以将它们编程到器件的闪存中，以便稍后调用。必须单独修整每个器件的值以确保准确性。也可以在内部通过使用频率时钟计数器 (FCC) 对 SYSOSC 进行修整。

在禁用频率校正环路 (FCL) 的情况下使用 CLK_OUT 的修整过程（无 ROSC 电阻器）

1. 按照 CLK_OUT 一节中的说明启用 CLK_OUT 单元并选择 SYSOSC 作为时钟源
2. 将 SYSOSC 频率设置为 BASE 以启动（确保 SYSOSCCFG 寄存器中的 FREQ 字段设置为表示 BASE 的 0h），并使 FCL 模式保持禁用状态
3. 在 SYSOSCTRIMUSER 寄存器中对目标频率进行初始调优编程：
   a. 将 SYSOSCTRIMUSER 寄存器中的 RCOARSE 修整字段设置为中程
   b. 将 SYSOSCTRIMUSER 寄存器中的 RFINE 修整字段设置为中程
4. 通过在 SYSOSCCFG 寄存器的 FREQ 字段中选择 USER，将 SYSOSC 切换为用户修整的频率
5. 测量 CLK_OUT 引脚上的 SYSOSC 频率
6. 通过在 SYSOSCCFG 寄存器的 FREQ 字段中选择 BASE，将 SYSOSC 切换回 BASE 频率
7. 将 SYSOSCTRIMUSER 参数调整为接近目标频率：
   a. 在 SYSOSCTRIMUSER 寄存器的 FREQ 字段中设置目标频率（16MHz 或 24MHz）
   b. 根据步骤 7a 中的选择，调整修整参数以实现 16MHz 或 24MHz
      i. 增大 SYSOSCTRIMUSER 寄存器中 CAP 修整字段的值将减小 SYSOSC 范围。对于 24MHz，请选择 CAP=0；对于 16MHz，请选择 CAP=1
      ii. 增大 SYSOSCTRIMUSER 中 RCOARSE 修整字段的值将以大约 1MHz 的较大步长降低频率
      iii. 增大 SYSOSCTRIMUSER 中 RFINE 修整字段的值将以大约 100kHz 的较小步长降低频率
8. 重复步骤 4 至 7，直至达到所需的精度
9. 保存得到的 SYSOSCTRIMUSER 值以供调用

在启用频率校正环路 (FCL) 的情况下使用 CLK_OUT 的修整过程（存在 ROSC 电阻器）

1. 首先，完成 FCL 禁用调整过程（如上所述），以获取 SYSOSCTRIMUSER 寄存器中 CAP、RCOARSE 和 RFINE 修整字段的起始值
2. 根据步骤 1 中获得的值中将 RCOARSE 值增加 02h
3. 在 CLK_OUT 仍然启用的情况下（SYSOSC 作为 CLK_OUT 的源），确保 SYSOSCCFG 寄存器中的 FREQ 字段设置为表示 BASE 的 0h，并通过设置 SYSOSCFCLCTL 寄存器中的 SETUSEFCL 位启用 FCL 模式
4. 将 SYSOSCUSERTRIM 寄存器中的 RDIV 修整字段设置为中程
5. 通过在 SYSOSCCFG 寄存器的 FREQ 字段中选择 USER，将 SYSOSC 切换为用户修整的频率
6. 测量 CLK_OUT 引脚上的 SYSOSC 频率
7. 通过在 SYSOSCCFG 寄存器的 FREQ 字段中选择 BASE，将 SYSOSC 切换回 BASE 频率
8. 将修整参数调整为接近目标频率：
   a. 增大 SYSOSCTRIMUSER 寄存器中 RDIV 修整字段的值将以大约 50kHz 的较小步长增加频率
   b. 如果频率处于饱和状态（似乎未调整），则继续增大或减小 RDIV
9. 重复步骤 5 至 8，直至达到所需的精度
10. 保存得到的 SYSOSCTRIMUSER 值以供调用

备注

SYSOSCTRIMUSER 参数仅在切换到 USER 频率时生效。选择 USER 频率后，对 SYSOSCTRIMUSER 的更改将无效。要刷新 USER 值，首先切换到 BASE 频率，然后更新 SYSOSCTRIMUSER，再然后将 SYSOSC 切换回 BASE 频率。

2.3.1.2.5 禁用 SYSOSC

通过设置 SYSOSCCFG 寄存器中的 DISABLESTOP 位，可以在 STOP 模式下禁用 SYSOSC。这样做会强制 MCLK 在 STOP 模式下使用 LFCLK（这是 STOP2 策略）。这可在 STOP 模式下实现尽可能低的功耗，因为系统以 32kHz 的频率运行并且 SYSOSC 不消耗电流。退出 STOP 模式并进入 RUN 模式时，SYSCTL 将自动重新启用 SYSOSC，并将 MCLK 切换回 SYSOSC。

可通过设置 SYSOSCCFG 寄存器中的 DISABLE 位来手动禁用 SYSOSC。设置 SYSOSCCFG.DISABLE 后，系统将在所有电源模式下从 LFCLK 运行。

备注

SYSOSCCFG 中的 DISABLE 位和 DISABLESTOP 位互斥，不能同时进行设置。

使用不同的高频时钟作为 MCLK 的时钟源（如 HFCLK 或 PLL）时，无法禁用 SYSOSC。这是因为当 MCLK 源自 HFCLK 或 PLL 时，由 SYSCTL 逻辑使用 SYSOSC。

在 STANDBY 和 SHUTDOWN 模式下始终自动禁用 SYSOSC。

2.3.1.3 系统锁相环 (SYSPLL)

系统锁相环 (SYSPLL) 采用输入参考时钟 SYSPLLREF 并调节输入频率以产生用户指定的高频时钟（SYSPLLCLK0、SYSPLLCLK1 和 SYSPLLCLK2X）供器件使用。具体而言，SYSPLL 时钟输出可用作 MCLK 和 CANCLK 的时钟源。图 2-7 展示了 SYSPLL 的方框图。

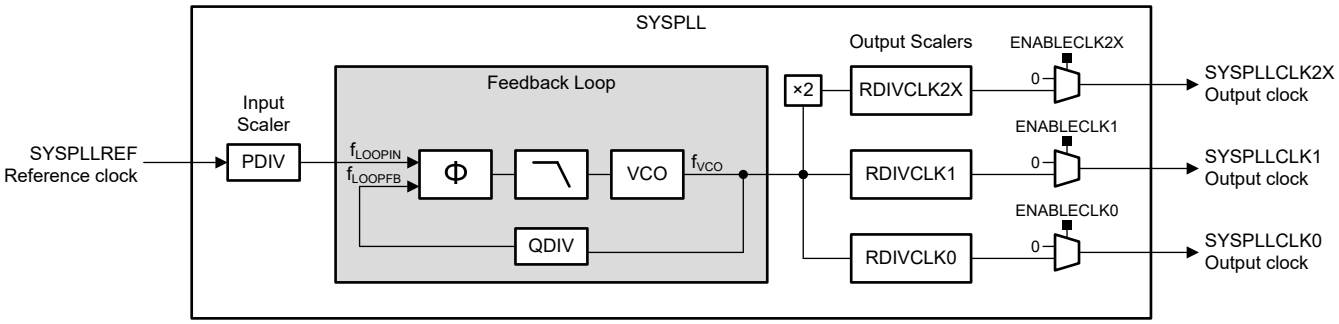


图 2-7. SYSPLL 方框图

备注

当启用 SYSPLL 时，不能禁用 SYSOSC。

2.3.1.3.1 配置 SYSPLL 输出频率

SYSPLL 接受 4-48MHz 的输入基准时钟。可用的基准时钟包括 SYSOSC 和 HFCLK。预分频器 PDIV 在 PLL 反馈环之前扩展所选的输入基准时钟。通过将 0x0 至 0x3 分别编程到 SYSPLLCFG1 寄存器的 PDIV 字段中，可将 PDIV 选作 /1、/2、/4 或 /8。

实际的分频器可通过方程式 6 中所示的 PDIV 寄存器设置计算得出。

$$SYSPLLREF_{DIV} = 2^{PDIV} \quad (6)$$

PLL 反馈环路将压控振荡器 (VCO) 输出设置为等于分频后的输入基准时钟 f_{LOOPIN} 乘以 QDIV 反馈分频器。QDIV 分频器是一个有效范围为 /2 至 /127 的整数分频器。通过将 0x01 至 0x7E（对应于 /2 到 /127）分别编程到 SYSPLLCFG1 寄存器的 QDIV 字段来选择所需的 QDIV 分频器。SYSPLLCFG1.QDIV=0x00 是无效配置。实际的反馈分频器可通过方程式 7 中所示的 QDIV 寄存器设置计算得出。

$$SYSPLLFB_{DIV} = QDIV + 1 \quad (7)$$

VCO 的输出频率 f_{VCO} 在方程式 8 中给出。

$$f_{VCO} = f_{SYSPLLREF} \times SYSPLLFB_{DIV} / SYSPLLREF_{DIV} \quad (8)$$

VCO 输出可提供三个独立的 SYSPLL 输出（SYSPLLCLK0、SYSPLLCLK1 和 SYSPLLCLK2X）。每个输出都有自己的分频器单元，可以生成多达 3 种不同的输出频率，供器件中的不同模块使用。第三个输出 (SYSPLLCLK2X) 还在分频器单元之前包含一个倍频器，用于提供更宽的输出频率范围和更低的功耗。

对于 SYSPLLCLK2X，输出分频器可以设置为 /1 到 /16，步长为 1。要设置 SYSPLLCLK2X 输出分频器，请将 0x0-0xF（对应于 /1 到 /16）分别编程到 SYSPLLCFG0 寄存器的 RDIVCLK2X 字段中。方程式 9 展示了如何根据给定的 RDIVCLK2X 寄存器设置计算有效的 SYSPLLCLK2X 分频器。

$$SYSPLLCLK2X_{DIV} = RDIVCLK2X + 1 \quad (9)$$

对于 SYSPLLCLK1 和 SYSPLLCLK0，输出分频器可设置为 /2 至 /32，步长为 2。要设置 SYSPLLCLK0 或 SYSPLLCLK1 输出分频器，请将 0x0-0xF（对应于 /2 至 /32）分别编程到 SYSPLLCFG0 寄存器中相应的 RDIVCLKx 字段中。方程式 10 展示了如何根据给定的 RDIVCLK0 设置计算有效的 SYSPLLCLK0 分频器，方程式 11 展示了如何根据给定的 RDIVCLK1 设置计算有效的 SYSPLLCLK1 分频器。

$$SYSPLLCLK0_{DIV} = 2 \times (RDIVCLK0+1) \quad (10)$$

$$SYSPLLCLK1_{DIV} = 2 \times (RDIVCLK1+1) \quad (11)$$

因此，SYSPLL 输出时钟频率由 f_{VCO} 和相应分频器的组合设置：

$$f_{SYSPLLCLK0} = f_{VCO} / SYSPLLCLK0_{DIV} \quad (12)$$

$$f_{SYSPLLCLK1} = f_{VCO} / SYSPLLCLK1_{DIV} \quad (13)$$

$$f_{SYSPLLCLK2X} = 2 \times f_{VCO} / SYSPLLCLK2X_{DIV} \quad (14)$$

启用和禁用 SYSPLL

配置后，通过置位 HSCLKEN 寄存器中的 SYSPLLEN 位来启用 SYSPLL。在启用 SYSPLL 之前，通过验证 CLKSTATUS 寄存器中的 SYSPLLOFF 位已置位，确保 SYSPLL 处于禁用状态。启用 SYSPLL 后，不得由应用软件禁用，直到 CLKSTATUS 寄存器中的 SYSPLLGOOD 或 SYSPLLOFF 位置位，表明 SYSPLL 已过渡至稳定的运行状态或稳定的死区状态。启用 SYSPLL 后，不得更改 SYSPLL 基准时钟选择。

备注

启用 SYSPLL 后，必须启用并以基频运行 SYSOSC，即使 HFCLK 被用作 SYSPLL 基准时钟也是如此。

SYSPLL 用法示例

要说明上述关系，以下列要求为例说明：

- 内部系统振荡器 (SYSOSC) 用作 SYSPLL 基准 (32MHz)
- SYSPLL 必须配置 80MHz 的输出频率来提供 MCLK，配置 40MHz 的输出频率来提供 CANCLK

为了实现这一点，可通过使用 PDIV 和 QDIV 将 VCO 配置为 80MHz。然后，SYSPLLCLK1 可以使用 /2 的输出分频器馈送至 CANCLK，SYSPLLCLK2X 可以使用 /2 的输出分频器馈送至 MCLK。

下面的步骤描述了如何配置 CKM 来以这种方式使用 SYSPLL：

1. 验证 SYSPLL 已禁用（CLKSTATUS 中的 SYSPLLOFF 已置位）
2. 确保 SYSOSC 以基频 (32MHz) 运行；即使 HFCLK 用作 SYSPLL 基准时钟而不是 SYSOSC，这也是 SYSPLL 运行的一项要求
3. 将 SYSOSC 设置为 SYSPLL 基准（确保 SYSPLLCFG0 寄存器中的 SYSPLLREF 位清零；这是复位后的默认状态）
4. 选择预分频器 PDIV 为 /2（将 SYSPLLCFG1.PDIV 设置为 0x01），将 f_{LOOPIN} 设置为 16MHz（32 除以 2）
5. 将 PLL 参数加载到 SYSPLLPARAM0 和 SYSPLLPARAM1 中以支持 f_{LOOPIN} 为 16MHz
6. 将反馈分频器 QDIV 设置为 5（将 SYSPLLCFG1.QDIV 设置为 4），得到 f_{VCO} =80MHz（16MHz 乘以 5）

7. 将 SYSPLLCLK1 和 SYSPLLCLK2X 的 SYSPLL 输出分频器设置为 /2（将 SYSPLLCFG0.RDIVCLK1 设置为 0x0，将 SYSPLLCFG0.RDIVCLK2X 设置为 0x1），使 SYSPLLCLK1 和 SYSPLLCLK2X 分别为 40MHz 和 80MHz
8. 通过置位 SYSPLLCFG0 寄存器中的 ENABLECLK1 和 ENABLECLK2X 位来启用 SYSPLLCLK1 和 SYSPLLCLK2X 输出
9. 启用并以基频（32MHz，这是复位后的默认状态）运行 SYSOSC 时，通过置位 HSCLKEN 寄存器中的 SYSPLLEN 来启用 SYSPLL
10. 通过在 CLKSTATUS 寄存器中测试 SYSPLLGOOD 来等待 SYSPLLGOOD 指示
11. 通过置位 SYSPLLCFG0 寄存器中的 MCLK2XVCO，选择 SYSPLLCLK2X 作为 HSCLK 多路复用器的 PLL 输出
12. 通过确保 HSCLKCFG 寄存器中的 HSCLKSEL 位清零（这是默认状态），选择 SYSPLL 作为 HSCLK 源
13. 通过置位 MCLKCFG 寄存器中的 USEHSCLK 位，选择高速时钟 (HSCLK) 作为 MCLK 的源。这将把 MCLK 从 SYSOSC 切换到 HSCLK。MCLK 现在以 80MHz 的频率从 SYSPLLCLK2X 运行。
14. 要配置 CANCLK，将 GENCLKCFG 寄存器中的 CANCLKSRC 位置位。

下面总结了本示例中使用的 SYSPLL 分频器值以供参考。

表 2-5. SYSPLL 分频器示例设置

参数	寄存器	位字段	位字段值	实际分频器
输入基准时钟分频器	SYSPLLCFG1	PDIV	0x1	/2
VCO 反馈环分频器	SYSPLLCFG1	QDIV	0x4	/5
输出时钟 1 分频器	SYSPLLCFG0	RDIVCLK1	0x0	/2
输出时钟 2X 分频器	SYSPLLCFG0	RDIVCLK2X	0x1	/2

调优指南

如果 PDIV、QDIV 和 RDIVCLKx 的多个组合可提供所需的输出频率，请考虑以下调优指南来确定最适合应用的值：

- 较低的 VCO 频率 (f_{VCO}) 可产生更低的功耗。请参阅器件数据表以了解 f_{VCO} 的允许范围。
- 更高的反馈环输入频率 (f_{LOOPIN}) 可实现更快的启动速度。例如，如果需要 80MHz 的输出频率，并且 f_{VCO} =40MHz，f_{VCO} = 40MHz 可通过将 PDIV 设置为 /8、将 QDIV 设置为 /10，从 SYSPLLREF 时钟 32MHz 得出。但是，这会使启动速度变慢，因为 f_{LOOPIN} <8MHz。通过将 PDIV 设置为 /4 并将 QDIV 设置为 /5 可以实现相同的结果，但是因为 f_{LOOPIN} 在本例中为 8MHz（32MHz 除以 4），可以使用较高输入范围的 SYSPLL 参数来加快启动速度。

2.3.1.3.2 加载 SYSPLL 查找参数

在使用 SYSPLL 之前，必须在 SYSPLLPARAM0 和 SYSPLLPARAM1 寄存器中配置若干调优参数。这些值由 PLL 反馈环路输入时钟频率 (f_{LOOPIN}) 决定。

SYSPLL 支持四个 f_{LOOPIN} 频率范围，如 SYSPLL 参数查找 所示。每个频率范围在 FACTORY 闪存区域中都有一个 64 位查找值，在启用 SYSPLL 之前，必须将该查找值从闪存复制到 SYSCTL 中的 SYSPLLPARAM0 和 SYSPLLPARAM1 寄存器中。

表 2-6. SYSPLL 参数查找

f_{LOOPIN}	查找地址 (PARAM0)	查找地址 (PARAM1)
32MHz ≤ FREQ ≤ 48MHz	0x41C4.0034	0x41C4.0038
16MHz ≤ FREQ < 32MHz	0x41C4.002C	0x41C4.0030
8MHz ≤ FREQ < 16MHz	0x41C4.0024	0x41C4.0028
4MHz ≤ FREQ < 8MHz	0x41C4.001C	0x41C4.0020

2.3.1.3.3 SYSPLL 启动时间

PLL 启动时间取决于 PLL 反馈环路输入频率 f_{LOOPIN}，以及 PLL 是在先前运行后启动（例如退出低功耗模式）还是在器件引导后首次启动。表 2-7 列出了 PLL 启动时间。

表 2-7. SYSPLL 启动时间

f_{LOOPIN}	SYSPLL 启动时间 (µs)	从低功耗模式退出时的 SYSPLL 启动时间 (µs)
32MHz ≤ FREQ	5	3
16MHz ≤ FREQ < 32MHz	15	12
8MHz ≤ FREQ < 16MHz	25	20
4MHz ≤ FREQ < 8MHz	40	35

2.3.1.4 低频晶体振荡器 (LFXT)

低频晶体振荡器 (LFXT) 是一种超低功耗晶体振荡器，支持标准的 32.768kHz 手表晶体。

要使用 LFXT，请在 LFXIN 引脚和 LFXOUT 引脚之间放置一个手表晶体。将这两个引脚上的负载电容器连接到电路接地 (VSS)。根据所用晶体的规格确定晶体负载电容器的大小。通过一个可编程驱动机制来支持多种晶体类型，此机制可实现驱动电流与特定晶体所需驱动强度之间的平衡。

LFXT 引脚 LFXIN 和 LFXOUT 与数字 IO 功能共享。要使用 LFXT，首先将 IOMUX 配置为在 LFXIN 和 LFXOUT 引脚上支持 LFXT 功能。配置 IOMUX 以在 LFXIN 和 LFXOUT 引脚上禁用各类数字 IO 功能。LFXT 默认为最高驱动强度（在 SYSCTL 中，LFCLKCFG.XT1DRIVE == 0x03），建议用于快速、可靠地启动振荡器。如果使用具有超低电容 (<3pF) 的晶体，则可以通过设置 LOWCAP 位 LFCLKCFG 来进一步降低 LFXT 的功耗（LOWCAP 默认情况下清零，以便与大多数晶体兼容）。

完成配置后，通过设置 SYSCTL 中 LFXTCTL 寄存器的 STARTLFXT 位来启动 LFXT。当振荡器成功启动时，LFXT 启动监视器会设置 SYSCTL 中 CLKSTATUS 寄存器的 LFXTGOOD 位。如果需要，可以降低晶体驱动强度以减少功耗。应用软件设置 STARTLFXT 后，内部 LFOSC 将被禁用以节省功耗，同时 LFXT 启动，预期 LFCLK 移至 LFXT。

要将 LFCLK 树切换为使用 LFXT 作为 32kHz 时钟源而不是 LFOSC，请设置 LFXTCTL 寄存器中的 SETUSELFXT 位，SYSCTL 会将 LFCLK 源持久切换为 LFXT。

启用 LFXT 后，内部 LFOSC 将永久禁用。LFOSC 无法重新启用，因此 LFCLK 无法切换回至 LFOSC，除非执行 BOOTRST。

备注

要使 MCLK 以 LFCLK 为时钟源，而 LFCLK 以 LFXT 为时钟源，请首先将 LFCLK 配置为使用 LFXT，然后将 MCLK 配置为使用 LFCLK。当 LFCLK 从 LFOSC 运行时，请勿将 MCLK 切换到 LFCLK，而应稍后将 LFCLK 切换到 LFXT。

备注

如果在启动时为 LFCLK 选择了 LFXT 但晶体无法正常启动，则需要执行 BOOTRST 以重新启用内部 LFOSC 并从 LFOSC 运行 LFCLK，否则系统将无法使用 32kHz LFCLK。在 LFXT 无法启动时，应用软件可能会利用备份存储器存储一个标志，指示晶体未正常启动，然后执行 BOOTRST。备份存储器通过 BOOTRST 保存，因此可以在 BOOTRST 周期后读取，以便在下一个启动序列中应用程序代码可以决定保持 LFCLK 来自 LFOSC 还是来自外部晶体（如果晶体出现硬件问题，从而阻止其启动）。

2.3.1.4.1 LFCLK_IN（数字时钟）

可以绕过 LFXT 电路，并将 32.768kHz 典型频率的数字时钟引入器件，用作 LFCLK 源而非 LFOSC 或 LFXT。要将 LFCLK 配置为使用数字时钟输入而非 LFXT 或 LFOSC，需首先配置 IOMUX，以便在适当的引脚上启用

LFCLK_IN 功能。当正确配置 IOMUX 且外部时钟源向 LFCLK_IN 输出 32kHz 时钟时，将设置 SYSCTL 中 EXLFCTL 寄存器的 SETUSEEXLF 位。

LFCLK_IN 与数字方波 CMOS 时钟输入兼容，且典型占空比应为 50%。

在设置 EXLFCTL 寄存器中的 SETUSEEXLF 之前，可以通过启用 LFCLK 监视器来检查 LFCLK_IN 上的有效时钟信号。默认情况下，如果未启动 LFXT，LFCLK 监视器将检查 LFCLK_IN。

一旦选择 LFCLK_IN 作为 LFCLK 源，就不可能在不经过 BOOTRST 的情况下改回为 LFOSC 或 LFXT。

备注

如果 MCLK 要以 LFCLK 为时钟源，而 LFCLK 以 LFCLK_IN 为时钟源，则首先将 LFCLK 配置为使用 LFCLK_IN，然后将 MCLK 配置为使用 LFCLK。当 LFCLK 从 LFOSC 运行时，不要将 MCLK 切换到 LFCLK，而应稍后将 LFCLK 切换到 LFCLK_IN。

备注

LFCLK_IN 和 LFXT 是互斥的，不得同时启用。如果在 LFXTCTL 寄存器中设置了 SETUSELFXT 位或 STARTLFXT 位，则不要设置 EXLFCTL 寄存器中的 SETUSEEXLF 位。

2.3.1.5 高频晶体振荡器 (HFXT)

高频晶体振荡器 (HFXT) 可与 4MHz 至 48MHz 范围内的标准晶体和谐振器搭配使用，为系统生成稳定的高速基准时钟。HFXT 可用于直接为主器件时钟树 (MCLK) 提供时钟，也可用作可生成更高频率的片上 PLL 的精密基准。此外，HFXT 与可从 PLL 或 SYSOSC 运行的主系统时钟异步，可作为采样时钟源直接提供给 ADC，也可作为功能时钟源提供给 CAN-FD 外设。

要使用 HFXT，必须在 HFXIN 和 HFXOUT 引脚之间填充一个晶体或谐振器。必须在连接到电路接地 (VSS) 的两个引脚上放置负载电容。晶体负载电容的大小必须根据所用晶体的规格而定。必须将 IOMUX 配置为在 HFXIN 和 HFXOUT 引脚上启用 HFXT 功能。配置 IOMUX 以在 HFXIN 和 HFXOUT 引脚上禁用任何数字 IO 功能。HFXT 频率范围必须通过配置 SYSCTL 中 HFCLKCLKCFG 寄存器中的 HFXTRSEL 位来设置。

可编程 HFXT 启动时间具有 64 µs 的分辨率。在启动 HFXT 之前，根据所需的晶体或谐振器规格，将适当的启动时间编程到 SYSCTL 中 HFCLKCLKCFG 寄存器的 HFXTTIME 字段中。

正确配置后，通过设置 SYSCTL 中 HSCLKEN 寄存器的 HFXTEN 位，即可启动 HFXT。当振荡器成功启动时，HFCLK 启动监视器会使 SYSCTL 内 CLKSTATUS 寄存器中的 HFCLKGOOD 位有效。

备注

当 HFXT 处于启用状态时，必须以基础频率启用 SYSOSC。

设置 HFXTEN 以启用 HFXT 后，应用软件必须验证 CLKSTATUS 寄存器中的 HFCLKGOOD 指示或 HFCLKOFF（关闭/死区）指示是否已由硬件置为有效，然后尝试通过清除 HFXTEN 来禁用 HFXT。当通过清除 HFXTEN 来禁用 HFXT 时，在硬件设置 CLKSTATUS 寄存器中的 HFCLKOFF 位之前，不得再次启用 HFXT。

要在接收到 HFCLKGOOD 状态后将 HFXT 用作 PLL 参考，请设置 SYSCTL 内 SYSPLLCFG0 寄存器中的 SYSPLLREF 位。如果选择 HFXT 作为 SYSPLL 的基准并且启用了 SYSPLL，那么必须先禁用 SYSPLL 并且必须设置 CLKSTATUS 寄存器中的 SYSPLLOFF 位，然后才能禁用 HFXT。

要在接收到 HFCLKGOOD 状态后直接将 HFXT 用作 MCLK 源，请首先设置 HSCLKCFG 寄存器中的 HSCLKSEL 位，选择 HFCLK 作为高速时钟源（而不是系统 PLL 输出）。然后，设置 MCLKCFG 寄存器中的 USEHSCLK 位，选择高速时钟源作为 MCLK 源。在设置 USEHSCLK 后，一定不能改变 HSCLKCFG 并且一定不能禁用 HFXT，直到通过清除 USEHSCLK 并验证已由硬件清除 CLKSTATUS 中的 HSCLKMUX 位来将 MCLK 源切换回 SYSOSC。

2.3.1.5.1 HFCLK_IN（数字时钟）

可以绕过 HFXT 电路并将 4MHz 至 48MHz 的典型频率数字时钟引入器件中，以用作 HFCLK 源，而不是使用 HFXT。要将 HFCLK 配置为使用数字时钟输入而不是 HFXT，请先配置 IOMUX 以在相应的引脚上启用 HFCLK_IN 功能。当 IOMUX 配置正确且时钟源正在向 HFCLK_IN 输出时钟时，设置 SYSCTL 内 HSCLKEN 寄存器中的 USEEXTHFCLK 位。

备注

当 HFCLK_IN 处于启用状态时，必须以基础频率启用 SYSOSC。

要在选择 HFCLK_IN 作为 HFCLK 源后将 HFCLK_IN 用作 PLL 基准，需设置 SYSCTL 内 SYSPLLCFG0 寄存器中的 SYSPLLREF 位。如果选择 HFCLK_IN 作为 SYSPLL 的基准（通过 HFCLK）并且 SYSPLL 处于启用状态，那么必须禁用 SYSPLL 并且必须设置 CLKSTATUS 寄存器中的 SYSPLLOFF 位，才能通过清除 HSCLKEN 中的 USEEXTHFCLK 位来禁用 HFCLK_IN。

要在选择 HFCLK_IN 作为 HFCLK 源后从 HFCLK_IN 获取 MCLK，请首先设置 HSCLKCFG 寄存器中的 HSCLKSEL 位，选择 HFCLK 作为高速时钟源（而不是系统 PLL 输出）。然后，设置 MCLKCFG 寄存器中的 USEHSCLK 位，选择高速时钟源作为 MCLK 源。在设置 USEHSCLK 后，一定不能改变 HSCLKCFG 并且一定不能禁用 HFCLK_IN，直到通过清除 USEHSCLK 并验证已由硬件清除 CLKSTATUS 中的 HSCLKMUX 位来将 MCLK 源切换回 SYSOSC。

HFCLK_IN 与数字方波 CMOS 时钟输入兼容，典型占空比应为 50%。

备注

HFCLK_IN 和 HFXT 是互斥的，不得同时启用。如果 HSCLKEN 中的 HFXTEN 位也已设置，请不要设置 USEEXTHFCLK 位。

2.3.2 时钟

CKM 接收振荡器输出并生成供器件使用的各种功能时钟。

时钟

- 系统时钟
  - MCLK：PD1 外设和 PD1 总线的主系统时钟
  - CPUCLK：CPU 时钟，源自 MCLK
  - ULPCLK：PD0 外设和 PD0 总线的主系统时钟，源自 MCLK
  - MFCLK：固定 4MHz 时钟，与 MCLK/ULPCLK 同步
  - MFPCLK：4MHz 固定时钟
  - LFCLK：固定 32kHz 时钟，与 MCLK/ULPCLK 同步
  - HFCLK：高频外部时钟
  - HSCLK：供 MCLK 使用的高速可配置时钟，来自 SYSPLL 或 HFCLK
- 外设专用时钟
  - ADCCLK：ADC 采样周期时钟
  - CANCLK：CAN-FD 模块功能时钟
  - RTCCLK：固定 32kHz 时钟直接连接 RTC
- 外部时钟
  - CLK_OUT：带有分频器的外部时钟输出，用于将时钟推送到外部电路

在 SHUTDOWN 模式下，所有时钟都被禁用。

除了上面列出的可配置时钟之外，还有几个时钟直接连接到模拟外设（请参阅节 2.3.2.13 部分）。

2.3.2.1 MCLK（主时钟）树

MCLK 是主系统时钟以及所有同步时钟（MCLK、CPUCLK、ULPCLK、MFCLK 和 LFCLK）的同步根点。它通常是系统中速度最高的时钟，在器件的完整温度范围内支持以高达 80MHz 的速度运行。MCLK 树是 CPUCLK（在运行模式下）、PD1 高速外设总线时钟（在运行和睡眠模式下）和 ULPCLK 低功耗总线时钟（在运行、睡眠、停止和待机模式下）的根时钟源。此外，4MHz MFCLK 和 32kHz LFCLK 输出与 MCLK 同步。

MCLK 到 PD1 外设的输出在运行和睡眠模式中启用，而在所有其他功耗模式下禁用。当在停止和待机模式下禁用 MCLK 到 PD1 的输出时，MCLK 树仍在运行，以便为 ULPCLK 提供时钟源并为 MFCLK 和 LFCLK 提供同步。

MCLK 源可通过无干扰时钟多路复用器进行选择，并且可以在运行时由用户软件动态更改。在进入停止和待机模式时或在异步快速时钟请求期间，也可以由硬件自动更改此源。

MCLK 的可用源包括：

- **HSCLK**（高速时钟），频率高达 80MHz，可通过以下方式获得时钟源：
  - **SYSPLLCLK0** 或 **SYSPLLCLK2X**（用于达到 80MHz 的最高 MCLK 速度）
  - **HFCLK** 适用于主时钟需要尽可能精确的应用
- **SYSOSC**，频率为 4、16、24 或 32MHz（复位后的默认时钟源）
- **SYSOSC**，频率为 4MHz，带 **MDIV** 分频器（适用于外设总线和 CPU 在 250kHz 和 4MHz 之间运行的应用）
- **LFCLK**，频率为 32kHz，适用于整个系统（包括 CPU）以 32kHz 的频率运行且峰值工作电流较低的应用

注：将 **SYSPLLCLK0** 或 **SYSPLLCLK2X** 设置为 **HSCLK** 的源，然后将 **HSCLK** 设置为 **MCLK** 的时钟源，否则器件可能处于不可预测的状态。

在运行和睡眠模式下使用 MCLK

启动后，默认情况下 MCLK 以 SYSOSC 为时钟源。决定要使用哪个振荡器作为 MCLK 的时钟源很重要，因为 MCLK 同时设置 CPUCLK 频率和 PD1 外设的总线时钟频率。因此，为 MCLK 选择的振荡器的精度和时钟速度不仅必须适合 CPU 运行，还必须适合于使用总线时钟作为其功能时钟的 PD1 外设运行。

为 MCLK 做出的时钟源和频率选择决策也会影响运行和睡眠模式下的 ULPCLK。有关 MCLK 和 ULPCLK 在运行和睡眠模式下如何相关的更多信息，请参阅 ULPCLK 部分。

在停止和待机模式下使用 MCLK

在停止和待机模式下，MCLK 到 PD1 外设的输出被禁用，但 ULPCLK（它是 PD0 外设的总线时钟）在停止模式下仍处于活动状态，而在待机模式下可选处于活动状态。有关 MCLK 源和 ULPCLK 在停止和待机模式下如何相关的更多信息，请参阅 ULPCLK 部分。

MCLK 源选择

应用软件可以通过在 SYSCTL 中适当地配置 MCLKCFG.USEHSCLK 和 HSCLKCFG.HSCLKSEL 寄存器位，将 MCLK 源从 SYSOSC 更改为 SYSPLL 或 HFCLK（即 HFXT 或 HFCLK_IN）。还可以通过设置 MCLKCFG.USELFCLK 来选择 LFCLK 作为所有模式下的 MCLK 源，从而在 CPU 和 PD1 外设运行时提供低峰值电流消耗。表 2-8 给出了在运行和睡眠模式下为 MCLK 选择不同时钟所需的适当寄存器位配置。

表 2-8. 运行和睡眠模式下的 MSPM0Gxx MCLK 源选择

所需的源	MCLKCFG.USEHSCLK	MCLKCFG.USELFCLK	HSCLKCFG.HSCLKSEL	SYSPLLCFG0.MCLK2XVCO
SYSOSC	0	0	不用考虑	不用考虑
SYSPLLCLK0	1	0	0	0
SYSPLLCLK2X	1	0	0	1
HFCLK	1	0	1	不用考虑
LFCLK	0	1	不用考虑	不用考虑

要在运行模式下将 MCLK 从 SYSOSC 切换到 LFCLK，请执行以下操作：

1. 如果启用了任何高速振荡器（SYSPLL、HFXT、HFCLK_IN），请先禁用它们，然后继续操作
2. 验证 MCLK 以 SYSOSC 为时钟源（CLKSTATUS.CURMCLKSEL 清零）
3. 如果 SYSOSC 未以基础频率运行，并且在将 MCLK 切换为 LFCLK 时要将 SYSOSC 保留为启用状态，则先将 SYSOSC 设置为基础频率，然后继续操作
4. 设置 MCLKCFG.USELFCLK 以将 MCLK 切换到 LFCLK 并使 SYSOSC 保留为启用状态，或设置 SYSOSCCFG.DISABLE 以将 MCLK 切换到 LFCLK 并禁用 SYSOSC

要在运行模式下将 MCLK 从 LFCLK 切换到 SYSOSC，请执行以下操作：

1. 验证 MCLK 以 LFCLK 为时钟源（设置了 CLKSTATUS.CURMCLKSEL）
2. 清除 MCLKCFG.USELFCLK 或 SYSOSCCFG.DISABLE，以其中哪个已设置为将 MCLK 切换到 LFCLK 为准

要将 MCLK 从 SYSOSC 切换到 HSCLK，请执行以下操作：

1. 验证 MCLK 以 SYSOSC 为时钟源（CLKSTATUS.HSCLKMUX 清零）。
2. 根据各自的要求启用所需的高速源（SYSPLL、HFXT、HFCLK_IN）。
3. 通过 HSCLKCFG.HSCLKSEL 控件选择所需的 HSCLK 源
4. 验证设置了 CLKSTATUS.HSCLKGOOD，以指示所选的 HSCLK 源有效。
5. 设置 MCLKCFG.USEHSCLK 以将 MCLK 切换到 HSCLK。

要将 MCLK 从 HSCLK 切换到 SYSOSC，请执行以下操作：

1. 验证 MCLK 以 HSCLK 为时钟源（设置了 CLKSTATUS.HSCLKMUX）。
2. 清除 MCLKCFG.USEHSCLK 以将 MCLK 切换到 SYSOSC。
3. 等待 CLKSTATUS.HSCLKMUX 清零，这表示 MCLK 现在以 SYSOSC 为时钟源。
4. 如果需要，禁用任何高速时钟源（SYSPLL 或 HFXT）

如表 2-8 所示，MCLKCFG 中的 USELFCLK 和 USEHSCLK 位是互斥的，不得同时设置。要将 MCLK 从 LFCLK 切换到 HSCLK，或从 HSCLK 切换到 LFCLK，MCLK 首先切换到 SYSOSC，然后再切换到最终时钟源。下面的过程描述了如何将 MCLK 从 HSCLK 切换到 LFCK，或从 LFCK 切换到 HSCLK。

要将 MCLK 从 HSCLK 切换到 LFCLK，请执行以下操作：

1. 验证 MCLK 以 HSCLK 为时钟源（设置了 CLKSTATUS.HSCLKMUX）。
2. 清除 MCLKCFG.USEHSCLK 以将 MCLK 切换到 SYSOSC
3. 等待 CLKSTATUS.HSCLKMUX 清零，这表示 MCLK 现在以 SYSOSC 为时钟源
4. 禁用任何已启用的高速源（SYSPLL、HFXT、HFCLK_IN）
5. 等待使 CLKSTATUS.HSCLKSOFF 有效，这表示高速时钟关闭
6. 设置 MCLKCFG.USELFCLK 以将 MCLK 切换到 LFCLK，同时使 SYSOSC 保留为启用状态，或设置 SYSOSCCFG.DISABLE 以将 MCLK 切换到 LFCLK 并禁用 SYSOSC

要将 MCLK 从 LFCLK 切换到 HSCLK，请执行以下操作：

1. 验证 MCLK 以 LFCLK 为时钟源（设置了 CLKSTATUS.CURMCLKSEL）
2. 清除 SYSOSCCFG.DISABLE 或 MCLKCFG.USELFCLK（根据之前设置的是哪个）
3. 等待 CLKSTATUS.CURMCLKSEL 清零，这表示 MCLK 现在以 SYSOSC 为时钟源
4. 根据各自的要求启用所需的高速振荡器（SYSPLL、HFXT、HFCLK_IN）
5. 通过 HSCLKCFG.HSCLKSEL 控件选择所需的 HSCLK 源
6. 验证设置了 CLKSTATUS.HSCLKGOOD，以指示所选的 HSCLK 源有效
7. 设置 MCLKCFG.USEHSCLK 以将 MCLK 切换到 HSCLK

备注

当 MCLK 当前以 HSCLK 为时钟源时（设置 CLKSTATUS 寄存器中的 HSCLKMUX 位），不得更改 HSCLK 源选择（不得更改 HSCLKCFG 寄存器中的 HSCLKSEL 位和 SYSPLLCFG0 寄存器中的 MCLK2XVCO 位）。要更改 HSCLK 源，首先使用上述过程将 MCLK 切换为 SYSOSC，重新配置 HSCLK 源，然后将 MCLK 切换回 HSCLK。

MCLK 分频器 (MDIV)

提供了一个 MCLK 源分频器 (MDIV)，以使 MCLK 在最低 SYSOSC 频率 (4MHz) 与 LFCLK 频率 (32kHz) 之间运行。MDIV 适用于峰值电流受限但仍要求时钟速度高于 32kHz 的应用。MDIV 支持对 4MHz SYSOSC 频率进行高达 16 分频，从而启用表 2-9 中给出的附加 MCLK 频率选项。例如，通过将 SYSOSC 设置为 4MHz 并将 MDIV 设置为 7（8 分频），可以获得 500kHz 的 MCLK 频率。

表 2-9 展示了使用 /2、/4、/8 和 /16 MDIV 配置实现的 MCLK 频率，但 MDIV 可以设置为 /2 和 /16 之间的任意整数分频器（MDIV 寄存器值分别为 0x1 到 0xF，0x0 将禁用 MDIV）。

表 2-9. 在 4MHz 和 32kHz 之间运行的典型 MCLK 配置

MCLK 源	MDIV	MCLK 频率
SYSOSC (4MHz)	0（已禁用）	4 MHz
	1 (/2)	2 MHz
	3 (/4)	1 MHz
	7 (/8)	500 kHz
	15 (/16)	250 kHz

要使用 MDIV 以低于 4MHz 的中频运行 MCLK，请执行以下步骤：

1. 禁用异步快速时钟请求（确保在 SYSOSCCFG 寄存器中设置了 BLOCKASYNCALL 位）
2. 确保选择 SYSOSC 作为 MCLK 源
3. 将 SYSOSC 频率设置为 4MHz（将 SYSOSCCFG 寄存器中的 FREQ 字段设置为 0x01）
4. 延迟 10 个 MCLK 周期
5. 在 MCLKCFG 寄存器的 MDIV 字段中设置所需的分频器值（从 1 到 15，分别对应于 /2 到 /16）

使用 MDIV 降低 MCLK 频率时，有几条规则适用：

- 当使用高速振荡器作为 MCLK 的时钟源（SYSPLL、HFCLK）时，不得使用 MDIV；如果选择 SYSPLL 或 HFCLK 作为 MCLK 时钟源，则必须禁用 MDIV (MCLKCFG.MDIV=0x00)
- 启用 MDIV 时，SYSOSC 频率必须保持在 4MHz
- 更改 SYSOSC 频率的异步请求必须保持被阻止状态

要禁用 MDIV，请执行以下操作：

1. 禁用 MDIV（将 MCLKCFG 寄存器中的 MDIV 字段设置为 0x0）
2. 等待 16 个 MCLK 周期，然后将 SYSOSC 频率从 4MHz 更改为另一个频率

备注

MDIV 不适用于 LFCLK，因为当选择 LFCLK 作为 MCLK 源时，MDIV 不在 LFCLK 路径中。如果为 MCLK 选择了 LFCLK，则必须禁用 MDIV。

2.3.2.2 CPUCLK（处理器时钟）

处理器时钟（CPU 时钟）始终直接来自 MCLK，并且在 RUN 模式下以 MCLK 频率运行。在所有其他电源模式下，CPUCLK 被禁用。

2.3.2.3 ULPCLK（低功耗时钟）

ULPCLK 是 PD0 电源域中的外设的总线时钟。ULPCLK 支持高达 40MHz 的工作频率，直接通过仅在 MCLK 来源于高速时钟（SYSPLL、HFXT 或 HFCLK_IN）时启用的时钟分频器 (UDIV) 获取自 MCLK 树。ULPCLK 频率取决于 MCLK 配置和所选的功率模式。

RUN 和 SLEEP 模式下的 ULPCLK 行为

PD0 电源域的频率限制为 40MHz（在 RUN 和 SLEEP 模式下）。因此，ULPCLK 必须始终保持 ≤40MHz。当 MCLK 配置为从 SYSOSC 或 LFCLK 运行时，SYSCTL 会自动禁用 UDIV，并且 $f_{ULPCLK}=f_{MCLK}$ 因为这些时钟源始终 ≤32MHz。

但是，当 MCLK 配置为从高速时钟（SYSPLL、HFXT 或 HFCLK_IN）运行时，硬件无法确保 ULPCLK ≤40MHz，因为 SYSCTL 不知道 MCLK 频率。在这种情况下，应用软件负责通过适当配置 UDIV 来确保在 RUN 和 SLEEP 模式下 ULPCLK ≤40MHz。默认情况下，ULPCLK 分频器设置为 2 分频 (MCLKCFG.UDIV==0x1)，因此可确保在不超过最高 MCLK 频率 (80MHz) 的频率下安全运行。如果 MCLK 配置为 ≤40MHz，则会通过将 MCLKCFG.UDIV 从 0x01 更改为 0x0（1 分频）将 ULPCLK 速度设置为等于 MCLK 速度。当 UDIV==0x0 时，MCLK==ULPCLK，对 PD0 外设的访问不会产生额外的延迟。

STOP 和 STANDBY 模式下的 ULPCLK 行为

SYSCTL 在 STOP 和 STANDBY 模式下自动禁用 UDIV。

- 在 STOP 模式下，MCLK 树（以及扩展的 ULPCLK）可以从频率为 4MHz 的 SYSOSC（如果 SYSOSCCFG.DISABLESTOP=0x0）或 32kHz 的 LFCLK（如果 SYSOSCCFG.DISABLESTOP=0x1）运行。当使用 SYSOSC 时 (SYSOSCCFG.DISABLESTOP=0x0)，SYSCTL 可以确保 ULPCLK 始终为 4MHz，即使 SYSOSC 以更高的频率运行（由于用户配置或由于来自外设的异步请求）也是如此。
- 在 STANDBY 模式下，MCLK 树（以及扩展的 ULPCLK）可以从 LFCLK 运行 (STANDBY0)，也可以禁用 (STANDBY1) 以便节能。在 STANDBY1 中，只有 TIMG0 和 TIMG1 计时器外设接收 ULPCLK。

表 2-10. MSPM0Gxx ULPCLK（按工作模式）

选择的功耗模式	配置	寄存器设置	ULPCLK 频率
RUN 或 SLEEP（最高 40MHz）	MCLK 源为 SYSOSC（RUN0、SLEEP0）	MCLKCFG.USEHSCLK=0x0、MCLKCFG.USELFCLK=0x0	根据 MCLK 配置，ULPCLK 来自 MCLK，其中：$f_{ULPCLK} = f_{MCLK}$
	MCLK 源为 HSCLK（SYSPLL、HFXT 或 HFCLK_IN）（RUN0、SLEEP0）	MCLKCFG.USEHSCLK=0x1、MCLKCFG.USELFCLK=0x0	根据 MCLK 配置，ULPCLK 来自 MCLK，其中：$f_{ULPCLK} = f_{MCLK}/UDIV$
	MCLK 源为 LFCLK（RUN1/2、SLEEP1/2）	MCLKCFG.USELFCLK=0x1 或 SYSOSCCFG.DISABLE=0x1	ULPCLK 来自 LFCLK，其中：$f_{ULPCLK} = f_{LFCLK} = 32kHz$
STOP（最高 4MHz）	STOP 模式，启用 SYSOSC (STOP0/1)	SYSOSCCFG.DISABLESTOP = 0x0	ULPCLK 来自 SYSOSC，其中 $f_{ULPCLK} = 4MHz$
	STOP 模式，禁用 SYSOSC (STOP2)	SYSOSCCFG.DISABLESTOP = 0x1	ULPCLK 来自 LFCLK，其中：$f_{ULPCLK} = f_{LFCLK} = 32kHz$
STANDBY（最高 32kHz）	STANDBY 模式，启用 ULPCLK 和 LFCLK (STANDBY0)	MCLKCFG.STOPCLKSTBY= 0x0	ULPCLK 来自 LFCLK，其中：$f_{ULPCLK} = f_{LFCLK} = 32kHz$
	STANDBY 模式，禁用 ULPCLK 和 LFCLK (STANDBY1)	MCLKCFG.STOPCLKSTBY= 0x1	对所有外设都禁用 ULPCLK，但 TIMG0 和 TIMG1 除外，这两者接收 $f_{ULPCLK} = f_{LFCLK} = 32kHz$
SHUTDOWN（关）	-	-	ULPCLK 关闭

2.3.2.4 MFCLK（中频时钟）

MFCLK 为器件上的各种外设提供一个连续 4MHz 时钟。MFCLK 4MHz 速率始终源自 SYSOSC。由于 SYSOSC 频率不是固定的（可以配置为 32MHz、24MHz、16MHz 或 4MHz），因此 SYSCTL 会自动将分频器应用于 SYSOSC，以将 MFCLK 保持在恒定的 4MHz 速率，而不管当前 SYSOSC 频率是多少。MFCLK 可供在运行、睡眠和停止功耗模式下需要恒定时钟源的计时器和串行接口等外设使用。

在 SYSRST 之后，MFCLK 最初处于禁用状态。通过设置 SYSCTL 中 MCLKCFG 寄存器的 USEMFTICK，可以在软件中启用 MFCLK。MFCLK 仅在运行、睡眠和停止功耗模式下处于活动状态，并且必须启用 SYSOSC，MFCLK 才能运行。

所有 MFCLK 边沿都与主系统时钟（MCLK 和 ULPCLK）同步，这意味着可以随时读取或写入由 MFCLK 计时的外设的寄存器，而无需任何特殊处理。

外设可以通过其各自的 CLKSEL 多路复用器，选择 MFCLK 作为其功能时钟源。并非所有外设都支持从 MFCLK 运行。

在停止模式下使用 MFCLK

当在停止模式下使用 MFCLK 时，可以将 SYSOSC 配置为在进入停止模式时自动切换至 4MHz（低频），并在退出停止模式而进入运行模式（换挡模式）时自动切换回之前选择的频率。由于 MFCLK 是 4MHz 时钟源，因此在停止模式下以 4MHz 运行 SYSOSC 可降低停止模式下的功耗。要将 SYSOSC 配置为换档模式，请参阅节 2.3.1.2.1。

有关使用 MFCLK 的要求

1. 使用 MFCLK 时，必须禁用 MDIV（MCLK 分频器）（设置为 /1）。可通过将 MCLKCFG 寄存器中的 MDIV 设置为 0x0 来禁用 MDIV。当 MCLKCFG.MDIV != 0 时，SYSCTL 硬件不允许 MFCLK 运行。
2. 使用 MFCLK 时，如果 MCLK 以 SYSOSC 以外的源为时钟源，则 MCLK 频率必须 ≥ 32MHz，MFCLK 才能正确运行。
3. 使用 MFCLK 时，如果 MCLK 以高速时钟 (HSCLK) 为时钟源，应用软件必须在将 MCLK 源从 SYSOSC 切换到 HSCLK 之前，通过设置 USEMFTICK 位来启用 MFCLK。当 MCLK 以 HSCLK 为时钟源时，不要更改 USEMFTICK 的状态。
4. 使用 MFCLK 时，如果 MCLK 以低频时钟 (LFCLK) 为时钟源，则不要启用 MFCLK。在将 MCLK 源从 SYSOSC 切换到 LFCLK 之前，应用软件必须通过设置 USEMFTICK 位来启用 MFCLK。设置 USEMFTICK 后，软件可以将 MCLK 切换到 LFCLK。在这种情况下，当 MCLK 以 LFCLK 为时钟源时，MFCLK 将暂停，并在 MCLK 切换回 SYSOSC 时恢复。
5. 当通过设置 MCLKCFG 寄存器中的 USEMFTICK 启用 MFCLK 时，硬件将其视为静态策略。不要清除 USEMFTICK。

当 MFCLK 配置为启用时，它仅在 SYSOSC 处于活动状态且 MCLK 不是以 LFCLK 为时钟源时才处于活动状态。当 MCLK 以 LFCLK 为时钟源时，硬件会自动停止 MFCLK。请注意，如果器件处于待机状态，则 MCLK 始终以 LFCLK 为时钟源，而 MFCLK 始终被硬件禁用。

异步快速时钟请求（如果已配置）将临时启用 SYSOSC 以处理特定的外设事件和活动。如果 MFCLK 配置为启用（设置了 USEMFTICK），则 MFTICK 将在外设使异步快速时钟请求有效时运行。

2.3.2.5 MFPCLK（中频精密时钟）

在运行、睡眠或停止模式下，MFPCLK 为外部时钟输出 (CLK_OUT) 多路复用器和 12 位 DAC 模块提供一个连续的 4MHz 时钟。与 MFCLK 还可产生供大多数外设使用的连续 4MHz 时钟不同，MFPCLK 与 MCLK/ULPCLK 异步，它既可以使用 SYSOSC 作为时钟源，也可以使用 HFCLK（HFXT 或 HFCLK_IN）作为时钟源，以提高 DAC 的精度，从而改善 DAC 的性能。

12 位 DAC 模块没有时钟选择多路复用器。通过配置 MFPCLK 来选择 12 位 DAC 时钟源。要选择 HFCLK（HFXT 或 HFCLK_IN）作为 MFPCLK 的时钟源，请设置 SYSCTL 中 GENCLKCFG 寄存器的 MFPCLKSRC。

要启用 MFPCLK，请设置 SYSCTL 中 GENCLKEN 寄存器的 MFPCLKEN 位。

备注

进入停止模式时，SYSCTL 会在停止模式下自动禁用 HFCLK。如果要在停止模式下使用 12 位 DAC，则 SYSOSC 必须是 MFPCLK 的时钟源。

2.3.2.6 LFCLK（低频时钟）

LFCLK 为器件上的各种外设提供连续的 32kHz 时钟。在 BOOTRST 之后，LFCLK 最初以内部 32kHz 振荡器 (LFOSC) 为时钟源。启动后，LFCLK 可由软件切换到低频晶体振荡器 (LFXT) 或低频数字时钟输入 (LFCLK_IN)。有关切换 LFCLK 源的说明，请参阅相应的振荡器部分。当更改 LFCLK 源时，将锁定更改并禁用 LFOSC 以减少功耗。在不执行 BOOTRST 的情况下，无法再次选择 LFOSC 作为 LFCLK 源。

备注

如果要选择 LFCLK 作为 MCLK 源，并且要将 LFXT 或 LFCLK_IN 用作 LFCLK 源，请在选择 LFCLK 作为 MCLK 的源之前，将 LFXT 或 LFCLK_IN 配置为 LFCLK 源。

LFCLK 在运行、睡眠、停止和待机功耗模式下处于活动状态。对于大多数处于待机模式下的外设，可以同时禁用 ULPCLK 和 LFCLK，从而实现尽可能低的待机模式功耗 (STANDBY1)。为此，请在进入待机模式之前，设置 SYSCTL 中 MCLKCFG 寄存器的 STOPCLKSTBY 位。在此状态下，RTC 和 TIMG0 以及 TIMG1 是仅有的计时外设。

LFCLK 是同步时钟。所有 LFCLK 边沿都与主系统时钟（MCLK 和 ULPCLK）同步，这意味着可以随时读取或写入由 LFCLK 计时的外设的寄存器，而无需任何特殊处理。RTC 是此规则的唯一例外情况，因为 RTC 接收异步 LFCLK，即 RTCCLK。

备注

当 MCLK/ULPCLK 不是来自 LFCLK 时（例如当它们来自 SYSOSC 时），依赖 LFCLK 运行的外设会看到低频时钟源的时钟边沿和相应的 LFCLK 边沿之间有 5 个 ULPCLK 周期同步延迟。当 MCLK/ULPCLK 频率恒定时，该延迟是恒定的，不会向 LFCLK 添加抖动。如果 MCLK/ULPCLK 频率发生变化，则同步延迟会按比例变化，这会导致 MCLK/ULPCLK 频率转换点处出现小的单周期 LFCLK 抖动。该抖动会改变一个 LFCLK 周期的占空比，但不会累积误差（LFCLK 周期数永远不会发生变化，从而确保外设的 LFCLK 时基准确）。常见的情况是 MCLK/ULPCLK 来自 SYSOSC，其频率为 32MHz，SYSOSC 偶尔会切换到 4MHz 来降低功耗。当 SYSOSC（根据定义，为 MCLK/ULPCLK）从 32MHz 切换到 4MHz 时，同步延迟从 0.16 µs 增加到 1.25 µs（增加 1.1 µs）。因此，在 SYSOSC 频率转换期间，一个 LFCLK 周期将比正常周期长 1.1 µs (3.6%)。当 SYSOSC 从 4MHz 切换回 32MHz 时，一个 LFCLK 周期会比正常周期短 1.1 µs，相移会恢复，误差不会长期累积。

外设可以通过其各自的 CLKSEL 多路复用器，选择 LFCLK 作为其功能时钟源。并非所有外设都支持从 LFCLK 运行。可以从 LFCK 运行主时钟 (MCLK)，在这种情况下，整个器件以 LFCLK 速率 (32kHz) 运行。

2.3.2.7 HFCLK（高频外部时钟）

高频外部时钟 (HFCLK) 是高频外部时钟选择多路复用器的输出，可以选择为高频振荡器 (HFXT) 输出或高频数字时钟输入 (HFCLK_IN)。

HFCLK 可用作以下时钟的源：

- HSCLK 源（用于提供 MCLK）
- SYSPLL 外部时钟
- CANCLK 源
- ADCCLK 源
- MFPCLK 源（带有可选分频器）

HFCLK 仅在运行和睡眠模式下可用。在所有其他模式下，SYSCTL 会自动禁用 HFCLK，以及 HFXT 本身（如果已启用）。

2.3.2.8 HSCLK（高速时钟）

高速时钟 (HSCLK) 是高速时钟选择多路复用器的输出，可通过设置 MCLKCFG 寄存器中的 USEHSCLK 位选择它来提供 MCLK。HSCLK 只是 MCLK 的一个选择选项；它不提供任何其他功能。HSCLK 可以配置为由 SYSPLL（SYSPLLCLK0 或 SYSPLLCLK2X）或 HFCLK（HFXT 或 HFCLK_IN）提供。默认情况下，HSCLK 由 SYSPLL 提供。要将 HSCLK 源更改为 HFCLK，请设置 HSCLKCFG 寄存器中的 HSCLKSEL 位。

HSCLK 仅在运行和睡眠模式下可用。在所有其他模式下，它与 SYSPLL 和 HFXT（如果已启用）一起由 SYSCTL 自动禁用。

如果 HSCLK 状态检查表明所选的 HSCLK 源未正确启动，即使软件请求，SYSCTL 也不会将 MCLK 切换到 HSCLK。

2.3.2.9 ADCCLK（ADC 采样周期时钟）

ADC 模块使用 ADCCLK 来设置 ADC 采样周期。给定 ADC 的 ADCCLK 由 CKM 提供给 ADC，但 ADCCLK 时钟选择在每个 ADC 外设配置寄存器内完成。有关配置 ADCCLK 的信息，请参阅 ADC 章节。ADCCLK 可被选为 ULPCLK、SYSOSC 或 HFCLK（HFXT 或 HFCLK_IN）。

备注

在包含多个 ADC 的器件上，每个 ADC 都映射了一个唯一的 ADCCLK，从而使各个 ADC 具有不同的采样时钟。

2.3.2.10 CANCLK（CAN-FD 功能时钟）

CANCLK 是由 HFCLK（HFXT 或 HFCLK_IN）或 SYSPLL (SYSPLLCLK1) 直接提供给 CAN-FD 模块的功能时钟。该时钟作为功能时钟提供给与主时钟 (MCLK) 异步的 CAN-FD 模块，以实现尽可能高的精度。通过配置 GENCLKCFG 寄存器中的 CANCLKSRC 位，可在 SYSCTL 中选择 CANCLK 源。CAN-FD 外设本身内提供了额外的 CAN-FD 时钟配置（有关更多详细信息，请参阅 CAN-FD 一章）。

2.3.2.11 RTCCLK（RTC 时钟）

RTCCLK 是实时时钟外设的唯一时钟源。由于 RTC 的重要计时性质，在 LFCLK 与主时钟 (MCLK) 同步之前会直接向 RTC 提供 LFCLK。此外，即使系统时钟树在 STANDBY 模式下完全受到门控以降低功耗，RTC 也能够继续计数。RTCCLK 无需特殊配置。由于 RTC 逻辑的始终与系统主时钟异步，因此在读取某些 RTC 寄存器时适用特殊规则。请参阅 RTC 一章以了解处理 RTC 寄存器的规则。

2.3.2.12 外部时钟输出 (CLK_OUT)

提供时钟输出单元，用于将数字时钟信号从器件发送到外部电路，或发送到频率时钟计数器。此特性可用于为外部电路计时，例如没有时钟源的外部 ADC。时钟输出单元有一组灵活的源可供选择，并包含一个可编程分频器。

CLK_OUT 的可用时钟源：

- SYSPLLCLK1
- HFCLK
- SYSOSC
- ULPCLK
- MFPCLK
- LFCLK

所选时钟源可以进行 1 分频（无分频）、2 分频、4 分频、6 分频、8 分频、10 分频、12 分频、14 分频或 16 分频，然后输出到引脚或频率时钟计数器。

要使用时钟输出单元，请执行以下操作：

1. 配置 IOMUX 以使用 CLK_OUT 选择器件引脚上的 CLK_OUT 功能。

2. 在 GENCLKCFG 寄存器的 EXCLKSRC 字段中选择所需的时钟源。
3. 如有必要，在 GENCLKCFG 寄存器的 EXCLKDIVVAL 字段中设置所需的时钟分频器，并通过设置 EXCLKDIVEN 位来启用分频器。
4. 通过设置 GENCLKEN 寄存器中的 EXCLKEN 位来启用外部时钟输出。

备注

选择 CLK_OUT 源作为 ULPCLK 或 MFPCLK 时，必须启用时钟分频器（必须设置 EXCLKDIVEN）。

备注

将 EXCLKEN 位清零以禁用 CLK_OUT 时，让时钟源运行 10 个时钟周期以稳定 EXCLKSRC 多路复用器。

备注

当禁用为 CLK_OUT 选择的时钟源时，如果在时钟源被禁用时 CLK_OUT 必须为逻辑低电平 (0)，则建议在禁用时钟源之前先禁用 CLK_OUT 功能。如果 CLK_OUT 保持启用，而 CLK_OUT 的时钟源被禁用，CLK_OUT 有可能停止在逻辑高电平 (1) 状态中。

备注

选择 CLK_OUT 作为 SYSPLLCLK1 时，SYSPLLCLK1 输出必须 ≤ 48MHz。根据特定器件和引脚的 IO 功能，可能存在进一步的速度限制；有关支持的 IO 速度的详细信息，请参阅器件特定数据表中的数字 IO 规格。

2.3.2.13 基础设施的直接时钟连接

器件中有多个直接时钟连接，可支持特定的模拟功能：

- SYSOSC 到 ADC
- SYSOSC 到 OPA（如果器件上存在 OPA）
- SYSOSC 和 LFCLK 到 PMU 模拟多路复用器 VBOOST 电路

与 ADC 的直接连接

除了接收 ADCCLK 以设置采样窗口外，ADC 模块还接收 SYSOSC 的直接输出。ADC 模块中的电荷泵逻辑使用到 ADC 的 SYSOSC 直接输出。可在任何频率下配置 SYSOSC 以支持此功能。ADC 支持在转换前自动请求 SYSOSC，因此无需应用软件来确保 SYSOSC 在触发 ADC 转换之前运行。

与 OPA 的直接连接

即使外设总线时钟 (ULPCLK) 以低于 OPA 要求的频率运行，SYSOSC 与 OPA 的直接时钟连接（如果存在）也能启用所有 OPA 工作模式。某些 OPA 模式要求 SYSOSC 以 32MHz 的频率运行。有关时钟的详细信息，请参阅 OPA 部分。应用软件负责将 SYSOSC 配置为支持所选 OPA 模式的频率。如果启用了 OPA 并将其配置为要求 SYSOSC 以特定频率运行的模式，并且软件尚未将 SYSOSC 配置为以该频率运行，则 SYSCTL 中 CLKSTATUS 寄存器的 OPAMPCLKERR 将有效，并且 OPA 无法正常工作。

与 PMU 模拟多路复用器 VBOOST 电路的直接连接

SYSOSC 和 LFCLK 都直接连接到 PMU 中的模拟多路复用器 VBOOST 电路，用于提高 COMP、OPA 和 GPAMP 使用的模拟多路复用器的性能。当外设需要 VBOOST 电路以确保正常运行时，会自动启用该电路。应用软件负责确保启用 SYSOSC 或 LFCLK 以支持 VBOOST 电路正确运行。如果 VBOOST 电路需要 SYSOSC 或 LFCLK，而时钟不可用，则会使 SYSCTL 中 CLKSTATUS 寄存器的 ANACLKERR 有效。SYSCTL 还可配置为在 ANACLKERR 有效时生成 SYSCTL 中断，以提醒应用，COMP、OPA 或 GPAMP 无法按预期正常运行。

2.3.3 时钟树

图 2-8 展示了 MSPM0Gxx 系列器件的顶层时钟树。该图显示了振荡器（源）和时钟（目标）之间的映射，以及选择多路复用器的 SYSCTL 寄存器位字段。请注意，并非所有器件都具有图 2-8 中显示的所有时钟系统特性。

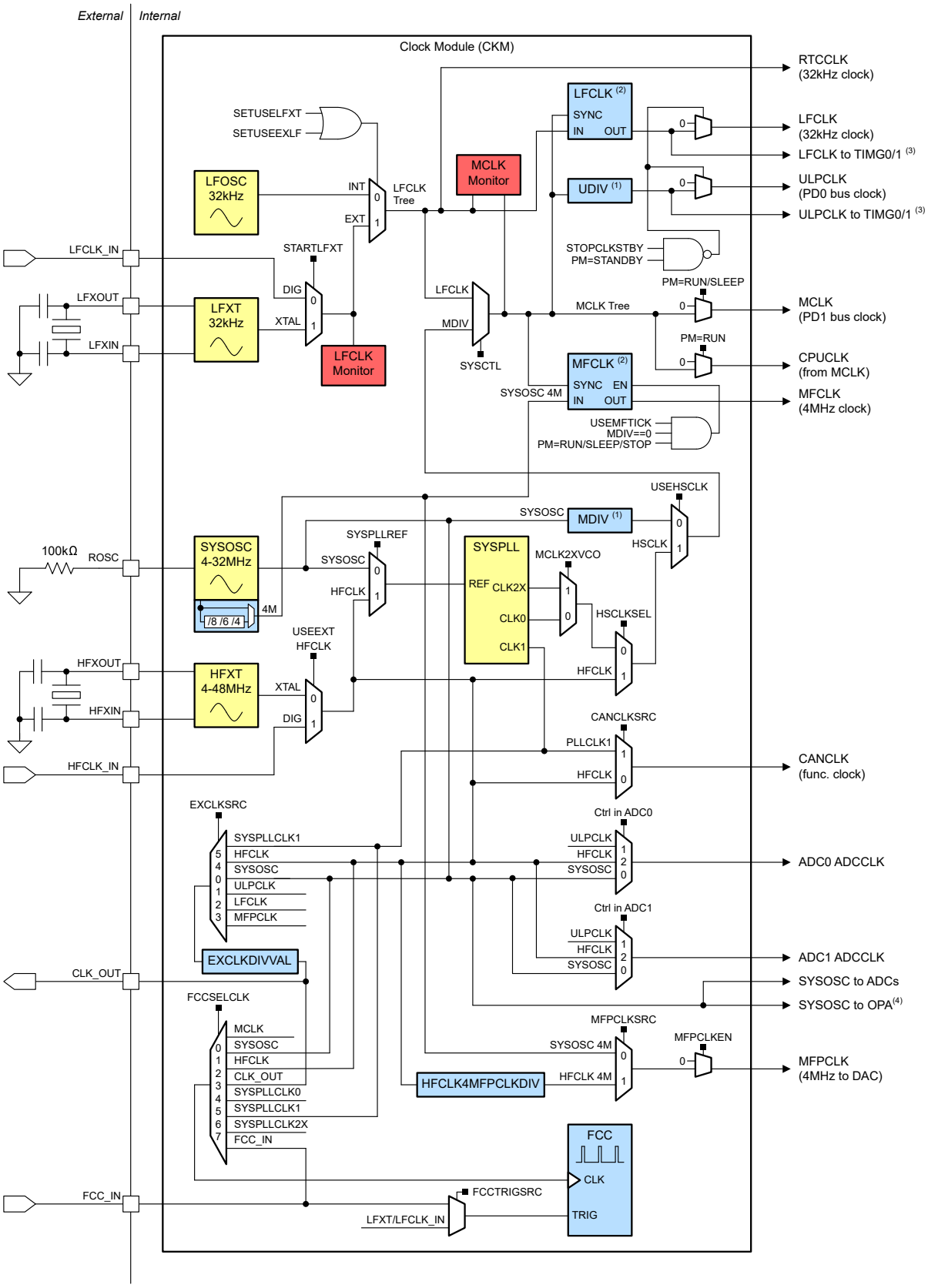


图 2-8. MSPM0Gxx 顶层时钟树

1. MDIV 和 UDIV 是支持特定用例的时钟分频器。有关 MDIV 和 UDIV 的具体设计注意事项，请分别查看 MCLK 和 ULPCLK 部分。
2. LFCLK 和 MFCLK 分别是固定频率 32kHz 和 4MHz 时钟，某些外设可选择这些时钟来确保恒定时钟速率，即使 MCLK 或 ULPCLK 更改源或速率也是如此。LFCK 和 MFCLK 始终相互同步，并与 MCLK 和 ULPCLK 同步。
3. TIMG0 和 TIMG1（通用计时器）接收未选通的 LFCLK 和 ULPCLK，这样当 STOPCLKSTBY 有效可将 LFCLK 和 ULPCLK 选通到所有其他外设时，TIMG0 和 TIMG1 能够在 STANDBY1 下继续运行，从而在 STANDY 模式下降低额外功耗。
4. SYSOSC 提供给 OPA 以供直接使用。OPA 不会自动请求 SYSOSC。应用软件负责确保在启用 OPA 之前启用 SYSOSC 并以所需的频率运行以支持正确的 OPA 操作。

2.3.3.1 外设时钟源选择

器件上的大多数外设都包含一个输入时钟选择多路复用器。这个多路复用器用于选择外设的功能时钟以及（可选）对其进行分频。图 2-9 展示了超集外设时钟选择多路复用器和可选的时钟分频器。请注意，并非每个外设都有图 2-9 所示的每个时钟源。例如，CRC、AES 和 DMA 等加速器依靠总线时钟运行。不能为这些外设选择 MFCLK 或 LFCLK。要确定外设的可用时钟源，请参阅特定外设的相关章节并查看其时钟输入选择。

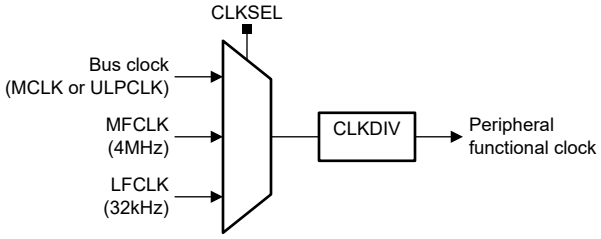


图 2-9. 外设时钟选择多路复用器和分频器

异常

还有几个外设具有独特的时钟选择方案，不使用上面显示的标准外设时钟多路复用器。通常，这是因为外设需要具有与系统其余部分异步的时钟源。出现这种情况的示例包括：

- 实时时钟 (RTC)，其中直接使用 RTCCLK，没有时钟选择选项
- 模数转换器 (ADC)，其中的 ADCCLK 有一个特殊选择多路复用器可直接获取异步时钟源（通过 ADC 控制寄存器中的特殊选项来选择 ADC 采样时钟）
- CAN-FD 控制器，其中的 CANCLK 可以是 PLL 输出或直接是 HFCLK（在 SYSCTL 中选择 CAN-FD 功能时钟源）
- DAC，其中直接使用 MFPCLK，没有时钟选择选项

2.3.4 时钟监控器

提供了多个硬件时钟监测器以确保 CKM 正常工作。时钟故障通过 SYSCTL 进行处理，并导致欠压复位（在发生致命故障时）或 SYSCTL 中断。

2.3.4.1 LFCLK 监视器

提供了一个低功耗模拟持续运行的时钟监视器，以确保 LFCLK 在其时钟源并非源自内部时正常运行（例如，当 LFCLK 以 LFXT 或 LFCLK_IN 而非 LFOSC 为时钟源时）。LFCLK 监视器仅用于检查时钟卡滞故障。它并不用于验证 LFCLK 的频率是否在特定容差范围内。

启用 LFCLK 监视器

除非 BOOTRST 的原因是 NRST 引脚，否则在 BOOTRST 之后启动时会禁用 LFCLK 监视器。

如果设置了 LFXTCTL 寄存器中的 STARTLFXT 位，则 LFCLK 监视器将监视 LFXT。如果 STARTLFXT 位保持清零，LFCLK 监视器将监视 LFCLK_IN。

在启用 LFCLK 监视器之前，请确保源时钟已启动并以 32kHz 的频率运行。如果设置了 STARTLFXT 并且监视器正在检查 LFXT，请等待 LFXTGOOD 指示，然后再启用 LFCLK 监视器。如果 STARTLFXT 清零并且监视器正在检查 LFCLK_IN 数字时钟输入，请确保外部时钟信号处于活动状态，然后再启用 LFCLK 监视器。如果在启动监视器时不存在有效时钟，则监视器将使故障有效。

可以通过设置 LFCLKCFG 寄存器中的 MONITOR 位来启用 LFCLK 连续监视器。

备注

LFCLK 监视器在启用后需要 100µs 才能变为活动状态。在从内部 LFOSC 切换 LFCLK 源之前，可以启用 LFCLK 监视器以监视 LFXT 或 LFCLK_IN。

LFCLK 监视器故障处理

如果检测到 LFCLK 卡滞故障，系统会根据系统时钟配置以两种方式之一进行响应：

- 如果 LFCLK 是 MCLK 的源 (例如，如果整个系统以源自 LFCLK 的 32kHz 频率运行) ，则 LFCLK 故障将视为致命故障，并使 BOOTRST 有效。在 BOOTRST 之后，MCLK 将来自 SYSOSC，频率为基频，而 LFCLK 将来自 LFOSC。
- 在发生故障时或发生故障后，如果 LFCLK 不是 MCLK 的源，LFCLK 故障会向处理器发出不可屏蔽中断 (NMI)，以便应用程序能够立即处理该故障并采取任何必要的措施。

LFXT 故障后回退到 LFOSC

如果检测到 LFXT 或 LFCLK_IN 故障，并且故障是由于系统级/PCB 级问题而导致外部时钟源无法可靠运行，可以回退到内部 32kHz LFOSC，而不是继续尝试使用 LFXT 或 LFCLK_IN。在这种情况下，建议按照以下过程操作：

1. 按照上面**启用 LFCLK 监视器**中给出的步骤，将器件配置为将 LFXT 或 LFCLK_IN 用作 LFCLK 源并启用 LFCLK 监视器。
2. 如果发生 LFCLK 监视器故障检测：
   a. 如果 MCLK 来自 LFCLK，则故障将自动生成 BOOTRST。
   b. 如果 MCLK 不是来自 LFCLK，则会生成 NMI。在 NMI 处理中，应用软件可以在检测到 LFCLK 故障时将 MCLK 切换到 LFCLK。此操作将触发硬件以自动生成 BOOTRST。可以通过将工作模式从 RUN0 更改为 RUN1/RUN2 或 STOP2/STANDBY 来将 MCLK 源切换为 LFCLK。
3. 在 BOOTRST 之后，MCLK 将来自 SYSOSC，频率为基频，而 LFCLK 将来自 LFOSC。应用软件可以检查导致复位的原因是致命时钟故障，并决定要采取的措施。
   a. 应用软件可能会再次尝试使用 LFXT/LFCLK_IN。如果 LFCLK 再次发生故障，应用软件可以返回上面的步骤 2，以再次将 LFCLK 切换回 LFOSC。
   b. 为了跟踪 LFCLK 故障的数量，应用软件可以将诊断信息存储在器件的关断存储器中。SHUTDNSTOREx 存储器位置通过 BOOTRST 保留。因此，应用软件可以跟踪发生了多少次 LFCLK 故障，如果该数量超过某个阈值，应用软件可以选择将 LFOSC 保留为 LFCLK 源，而不是再次尝试使用 LFXT/LFCLK_IN。

2.3.4.2 MCLK 监测器

数字时钟监视器可以与 MCLK 一起使用。如果在 1-12 个 LFCLK 周期的时段内没有 MCLK 活动，MCLK 监视器会将 MCLK 故障置为有效。MCLK 故障始终被视为对系统是致命的，并会生成一个 BOOTRST。

在配置并运行 LFCLK 后，可以启用 MCLK 监视器。要启用 MCLK 监视器，请设置 SYSCTL 中 MCLKCFG 寄存器的 MCLKDEADCHK 位。启用后，MCLK 监视器会在除 STANDBY1 和 SHUTDOWN 之外的所有工作模式下运行。

2.3.4.3 启动监视器

为应用软件提供的时钟启动监视器可用于确认 LFOSC、LFXT/LFCLK_IN、HFXT/HFCLK_IN、SYSPLL 和 HSCLK 源在由软件选择用于在系统中提供时钟之前处于活动状态。当一个时钟源已经成功启动并且准备就绪时，在 SYSCTL 的 CLKSTATUS 寄存器中会给出 GOOD 指示，并产生中断。启动监视器仅在进行了时钟系统配置的

相关更改时提供状态指示。当给出初始 GOOD 指示时，启动监视器不会持续监控时钟。对于 LFCLK 和 MCLK，则会进行持续监控。

2.3.4.3.1 LFOSC 启动监视器

LFOSC 在 BOOTRST 后自动启动。启动 LFOSC 需要一些时间。提供了一个启动监视器来向应用软件指示 LFOSC 启动何时完成，启动完成后，LFCLK 可供外设使用。当 LFSOSC 启动完成时，LFOSC 启动监视器会使 SYSCTL 中 CLKSTATUS 寄存器的 LFOSCGOOD 位有效，并使 LFOSCGOOD 中断有效以提示应用程序。有关 LFOSC 启动时间，请参阅器件特定数据表。

2.3.4.3.2 LFXT 启动监视器

LFXT 在启用后需要一段时间才能启动。提供了一个启动监视器来向应用软件指示 LFXT 已成功启动，此时可以选择 LFXT 作为 LFCLK 源。一旦 LFXT 启动完成，LFXT 启动监视器将使 SYSCTL 中 CLKSTATUS 寄存器的 LFXTGOOD 位有效，并且将使 LFXTGOOD 中断有效以提示应用程序。有关 LFXT 启动时间，请参阅器件特定数据表。

2.3.4.3.3 HFCLK 启动监视器

HFXT 在启用后需要一段时间才能启动。该器件提供了一个启动监视器来向应用软件指示是否已成功启动 HFXT，此时可以选择 HFCLK 来提供各种系统功能。HFCLK 启动监视器还支持检查 HFCLK_IN 数字时钟输入是否存在时钟卡滞故障。

要启用 HFCLK 启动监视器，请清除 SYSCTL（默认状态为禁用）内 HFCLKCLKCFG 寄存器中的 HFCLKFLTCHK 位。

启动 HFXT 时或选择 HFCLK_IN 作为 HFCLK 源时，则会清除 SYSCTL 内 CLKSTATUS 寄存器中的 HFCLKGOOD 和 HFCLKOFF 位。

在使用 HFXT 的情况下，在指定的 HFXT 启动时间结束后，会测试 HFXT 状态。如果 HFXT 成功启动，HFXT 启动监视器会使 CLKSTATUS 寄存器中的 HFCLKGOOD 位有效，同时也会使 HFCLKGOOD 中断有效。如果 HFXT 未在指定的启动时间内启动，则会设置 HFCLKOFF 位，指示 HFXT 在启动时发生故障。

在使用 HFCLK_IN 的情况下，在选择 HFCLK_IN 后，将执行时钟卡滞检查。如果时钟处于活动状态，则会设置 CLKSTATUS 寄存器中的 HFCLKGOOD 位，并且也会使 HFCLKGOOD 中断有效。如果 HFCLK_IN 信号卡住，则会设置 CLKSTATUS 寄存器中的 HFCLKOFF 位，指示 HFCLK_IN 发生故障。

如果需要，可以通过保持设置 SYSCTL 内 HFCLKCLKCFG 寄存器中的 HFCLKFLTCHK 位来使 HFCLK 启动监视器对 HFCLK 的检查保持禁用状态。

备注

在尝试进入停止或待机低功耗模式之前，HFCLK 必须处于稳定状态。在进入停止或待机模式之前，请确保已设置 HFCLKGOOD 位或 HFCLKOFF 位。

2.3.4.3.4 SYSPLL 启动监视器

SYSPLL 在启用后需要一段时间来启动和稳定。通过提供的启动监视器可以了解应用软件是否已成功启动 SYSPLL，在成功启动后则可以选择 SYSPLL 的时钟输出来提供各种系统功能。

当 SYSPLL 启动后，SYSCTL 的 CLKSTATUS 寄存器中的 SYSPLLGOOD 和 SYSPLLOFF 位将被清除。启动/稳定时间到期后将测试 SYSPLL 状态。如果 SYSPLL 成功启动，SYSPLL 启动监视器将使 CLKSTATUS 寄存器中的 SYSPLLGOOD 位生效，并且 SYSPLLGOOD 中断也将生效。如果 SYSPLL 未在指定时间内启动，则会设置 SYSPLLOFF 位，表示 SYSPLL 在启动时不工作。

备注

在尝试进入停止或待机低功耗模式之前，SYSPLL 必须处于稳定状态。在进入 STOP 或 STANDBY 模式之前，请确保已设置 SYSPLLGOOD 位或 SYSPLLOFF 位。

2.3.4.3.5 HSCLK 状态

HSCLK 可以来自 HFCLK 或 SYSPLL。SYSCTL 中的 CLKSTATUS 寄存器提供 HSCLKGOOD 和 HSCLKDEAD 指示，分别指示所选的 HSCLK 源以 GOOD 状态成功启动或以 DEAD 状态发生故障。如果未设置 HSCLKGOOD，即使收到请求，SYSCTL 也不会将 MCLK 切换到 HSCLK。

此外，CLKSTATUS 中提供了 HSCLKSOFF 位来指示所有 HSCLK 源（SYSPLL、HFCLK）是关闭（禁用）还是以 DEAD 状态启动。

2.3.5 频率时钟计数器 (FCC)

频率时钟计数器 (FCC) 可对器件上的各种振荡器和时钟进行灵活的系统内测试和校准。FCC 计算在已知固定触发周期（源自次级基准源）内所选源时钟上显示的时钟周期数，以估算源时钟的频率。

应用软件可以使用 FCC 来测量以下源振荡器和时钟（通过 GENCLKCFG 寄存器中的 FCCSELCLK 字段选择）的频率：

- MCLK
- SYSOSC
- HFCLK
- CLK_OUT
- SYSPLL（三个 SYSPLL 输出中的任一个）
- 外部 FCC 输入 (FCC_IN)

可以配置用于设置源时钟脉冲计数的触发时间的基准时钟（通过 GENCLKCFG 寄存器中的 FCCTRIGSRC 字段），并可通过以下方式进行驱动：

- 外部 FCC 输入 (FCC_IN)
- LFCLK
- LFOSC、LFXT 或 LFCLK_IN 多路复用器的输出 (32kHz)

可通过以下两种方式之一设置触发时间周期（通过 GENCLKCFG 寄存器中的 FCCLVLTRIG 字段）：

- 电平触发（基准时钟输入的一个上升沿到一个下降沿）。请注意，如果使用电平触发，则 LFCLK_IN 无法用作触发时钟源。
- 上升沿到上升沿触发，在定义数量的基准时钟的时钟周期内（可通过 GENCLKCFG 寄存器中的 FCCTRIGCNT 字段从 1 到 32 进行选择）

在电平触发模式下选择触发源作为外部 FCC 输入时，可通过在 FCC_IN 引脚上施加所需触发长度的逻辑高电平脉冲来设置用户指定的计数周期。

将触发源选作 LFXT 时，使用上升沿到上升沿触发将使 FCC 捕获在 LFXT 的 1 至 32 32.768kHz 时钟周期 (30.5 µs) 内发生的源时钟脉冲数。

FCC 计数器为 22 位，支持从 0 到 2^{22} – 1（或 4 194 303）的计数。

虽然外部 FCC 输入（FCC_IN 函数）可用作 FCC 时钟源或 FCC 触发输入，但在同一 FCC 捕获期间，它不能同时用于这两个函数。必须将其配置为 FCC 时钟源或 FCC 触发器。

2.3.5.1 使用 FCC

使用 FCC_IN 触发器的上升沿到上升沿触发模式

以下步骤说明了如何使用 FCC 在参考时钟设置的触发周期内对源时钟脉冲进行计数，同时选择 FCC_IN 引脚作为参考时钟，并选择 SYSOSC 作为源时钟。此示例可用于根据在外部提供给 FCC_IN 引脚的精确时钟源来校准 SYSOSC 频率。

1. 通过配置 GENCLKCFG 寄存器中的 FCCSELCLK 字段将源时钟设置为 SYSOSC。
2. 通过清除 GENCLKCFG 寄存器中的 FCCTRIGSRC 位将参考时钟设置为 FCC_IN。
3. 通过清除 GENCLKCFG 寄存器中的 FCCLVLTRIG 位来选择上升沿到上升沿触发。

4. 在 GENCLKCFG 寄存器的 FCCTRIGCNT 字段中选择所需的参考时钟周期数来对此期间的源时钟进行计数。
5. 确保在所需频率下启用 SYSOSC，并确保连接到 FCC_IN 的外部时钟源正常运行后再继续。
6. 将 GO 位和 KEY 字段写入 FCCCMD 寄存器，以便在下一个触发时钟周期开始 FCC 捕捉。
7. 轮询 CLKSTATUS 寄存器中的 FCCDONE 状态位。捕捉完成后，FCCDONE 将由硬件设置。FCCDONE 为只读状态，在开始新的捕捉时会由硬件自动清除。
8. 从 FCC 寄存器中的 22 位 DATA 字段提取计数结果。

使用 LFXT 触发器的上升沿到上升沿触发模式

以下步骤说明了如何使用 FCC 在参考时钟周期内对源时钟脉冲进行计数，同时选择 LFXT 作为参考时钟，并选择 SYSOSC 作为源时钟。此示例可用于根据精确的 32.768kHz 手表晶体来校准 SYSOSC 频率。

1. 通过配置 GENCLKCFG 寄存器中的 FCCSELCLK 字段将源时钟设置为 SYSOSC。
2. 通过设置 GENCLKCFG 寄存器中的 FCCTRIGSRC 位将参考时钟设置为 LFXT。
3. 通过清除 GENCLKCFG 寄存器中的 FCCLVLTRIG 位来选择上升沿到上升沿触发。
4. 在 GENCLKCFG 寄存器的 FCCTRIGCNT 字段中选择所需的参考时钟周期数来对此期间的源时钟进行计数。
5. 确保在所需频率下启用 SYSOSC，并确保 LFXT 正常运行后再继续。
6. 将 GO 位和 KEY 字段写入 FCCCMD 寄存器，以便在下一个触发时钟周期开始 FCC 捕捉。
7. 轮询 CLKSTATUS 寄存器中的 FCCDONE 状态位。捕捉完成后，FCCDONE 将由硬件设置。FCCDONE 为只读状态，在开始新的捕捉时会由硬件自动清除。
8. 从 FCC 寄存器中的 22 位 DATA 字段提取计数结果。如果 SYSOSC 以 32MHz 的频率运行并且 FCCTRIGCNT 设置为“0”（一个参考时钟周期），结果应该是在单个 32.768kHz 周期内计算的大约 976 个周期。
   a. 为了校准 SYSOSC 以便在 24MHz 频率下运行，必须调整 SYSOSC 用户修整值，直至计数到大约 732 个周期。
   b. 为了校准 SYSOSC 以便在 16MHz 频率下运行，必须调整 SYSOSC 用户修整值，直至计数到大约 488 个周期。

   通常，增加 FCCTRIGCNT 值会提高测量精度，但也会增加测量时间。

采用 FCC_IN 触发器和 HFCLK_IN 时钟的电平触发模式

以下步骤说明了如何使用 FCC 在一个外部参考脉冲窗口内对源时钟脉冲进行计数，并选择 HFCLK_IN 作为源时钟。此示例可用于根据外部信号驱动的固定脉冲宽度来测量外部时钟源的频率。

1. 通过配置 GENCLKCFG 寄存器中的 FCCSELCLK 字段将源时钟设置为 HFCLK。
2. 通过清除 GENCLKCFG 寄存器中的 FCCTRIGSRC 位将触发时钟设置为 FCC_IN 引脚功能。
3. 通过设置 GENCLKCFG 寄存器中的 FCCLVLTRIG 位来设置电平触发。
4. 确保 IOMUX 配置为 FCC_IN，HFCLK 配置为 HFCLK_IN，并由外部时钟提供 HFCLK_IN。
5. 将 GO 位和 KEY 字段写入 FCCCMD 寄存器，以便在 FCC_IN 变为逻辑高电平时开始 FCC 捕捉。请注意，如果在 GO 生效时 FCC_IN 已经是逻辑高电平，则会立即开始计数。如果使用电平模式，在设置 GO 时，FCC_IN 应该为低电平，在设置 GO 后，触发脉冲应该发送至 FCC_IN。
6. 轮询 CLKSTATUS 寄存器中的 FCCDONE 状态位。捕捉完成后，FCCDONE 将由硬件设置。FCCDONE 为只读状态，在开始新的捕捉时会由硬件自动清除。
7. 从 FCC 寄存器中的 22 位 DATA 字段提取计数结果。

2.3.5.2 FCC 频率计算和精度

如果触发时间已知，则可以在捕获后计算源时钟的频率。频率的计算方法是将捕获的源时钟周期数除以触发时间。例如，如果触发源是 32.768kHz 时钟，触发模式为上升沿到上升沿，周期计数为 1，则触发时间为一个 32.768kHz 时钟周期 (30.5 µs)。如果捕获的源计数返回为 122，则源时钟的频率计算方法是 122 除以 30.5 µs，得到大约 3.99MHz 的源时钟频率。

$$f_{source} = FCC.DATA / ((GENCLKCFG.FCCTRIGCNT+1) / f_{ref}) \quad (15)$$

FCC 的精度取决于触发时钟精度以及捕获的时钟周期总数。由于触发器与源时钟同步，FCC 固有误差为每次捕获 ≤2 个源时钟周期。因此，随着周期数的增加（随着触发时间的增加和/或源时钟频率的增加），这两个时钟周期的影响会减小。在 1 个 32.678kHz 周期 (FCCTRIGCNT=0) 和 32 个时钟周期 (FCCTRIGCNT=31) 捕获的各种源时钟频率下，FCC 的近似固有误差如表 2-11 所示。

表 2-11. FCC 误差

用例（源时钟频率）	FCC 触发时间	FCC 计数结果	FCC 计数不确定性	近似 FCC 固有不确定性误差
4MHz 源时钟	30.5 µs	122	2 个周期	1.6%
	976.6 µs	3906		0.05%
16MHz 源时钟	30.5 µs	488		0.4%
	976.6 µs	15625		0.01%
24MHz 源时钟	30.5 µs	732		0.27%
	976.6 µs	23437		0.01%
32MHz 源时钟	30.5 µs	976		0.20%
	976.6 µs	31250		0.01%
80MHz 源时钟	30.5 µs	2441		0.08%
	976.6 µs	78125		0.003%

备注

使用 FCC_IN 信号时，建议在 FCC_IN 引脚上具有 10ns 或更短的快速压摆率，以更大限度地降低测量不确定性。

2.4 系统控制器 (SYSCTL)

系统控制器 (SYSCTL) 包含用于管理 PMU 和 CKM 模拟电路的配置和状态的所有控制逻辑。SYSCTL 还提供复位管理、NRST 和 SWD 引脚复用控制、闪存存储体交换控制以及闪存 ECC 错误处理。

所有电源、时钟和复位配置均通过 SYSCTL 存储器映射寄存器接口完成。

2.4.1 复位和器件初始化

SYSCTL 可以管理器件复位级别和器件初始化。

2.4.1.1 复位级别

该器件有五个复位级别：

1. 上电复位 (POR)
2. 欠压复位 (BOR)
3. 引导复位 (BOOTRST)
4. 系统复位 (SYSRST)
5. CPU 复位 (CPURST)

图 2-10 给出了复位级别之间的关系。

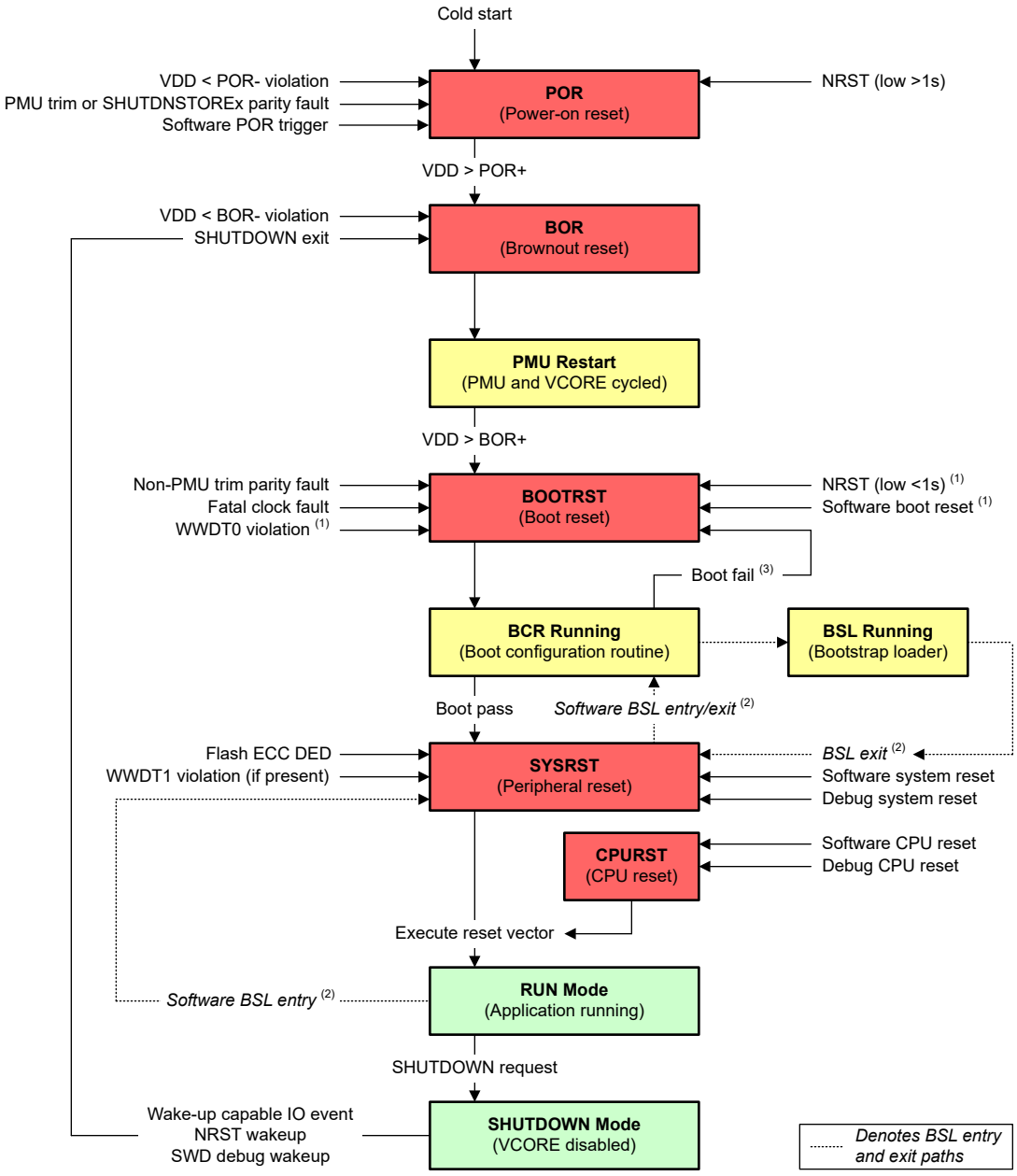


图 2-10. MSPM0 复位级别

(1) NRST（低电平 <1s）、软件引导复位或 WWDT0 违例触发的 BOOTRST 都会运行引导配置例程，但不会对 LFXT 或 LFCLK_IN 使用的任何 IO 引脚的 RTC、LFXT、LFCLK、LFCLK_IN 和 IOMUX 配置进行复位。这样 RTC 就可以通过外部复位触发器保持时间。

(2) 软件触发的引导加载程序 (BSL) 进入命令首先触发 SYSRST，然后运行引导配置例程 (BCR) 以在启动 BSL 之前验证 BSL 进入状态。在 BSL 执行结束后，将产生 SYSRST，并再次执行 BCR。BCR 完成后，将发出最终的 SYSRST，并启动应用程序。整个过程不会复位 RTC、LFXT/LFCLK/LFCLK_IN、由 LFXT 或 LFCLK_IN 使用的任何 IO 引脚的 IOMUX 配置以及 SYSOSC FCL 启用配置，因为在整个过程中只有 SYSRST 复位级别会生效。这样 RTC 就可以通过外部复位触发器保持时间。

(3) 如果在执行引导配置例程期间出现引导失败，则 SYSCTL 可以生成 BOOTRST 以再次从 BOOTRST 级别尝试引导过程。请参阅节 2.4.1.8。

备注

此图中未显示 SLEEP、STOP 和 STANDBY 工作模式。除非发生了导致复位级别生效或模式暂停的异常，否则这些模式源自并返回到 RUN 模式。

2.4.1.1.1 上电复位 (POR) 复位级别

上电复位 (POR) 是完整的器件复位。

以下条件会产生 POR：

- 器件上电（冷启动）
- POR- 电源监测器违例（VDD 降至 POR 电源监测器负向阈值以下）
- PMU 修整数据或关断存储器发生奇偶校验故障
- 软件通过 SYSCTL *(RESETLEVEL 0x03)* 触发 POR
- 处于 NRST 模式时，NRST 引脚保持低电平超过一秒钟

POR 总是会复位关断存储器、重新启用 NRST/SWD 引脚功能（如果已禁用）并触发 BOR。

2.4.1.1.2 欠压复位 (BOR) 复位电平

欠压复位 (BOR) 将器件电源管理单元 (PMU) 复位。由 VCORE 供电的所有稳压内核逻辑均被下电上电。

以下条件会生成 BOR：

- POR
- BOR0- 电源监测器违例（VDD 降至 BOR0- 电源监测器负向阈值以下）
- 退出关断模式（通过支持唤醒功能的 IO、NRST 或 SWD）

BOR 不会复位以下参数：

- 关断存储器 (SHUTDNSTOREx)
- NRST 状态，如果被软件禁用
- SWD 状态，如果被软件禁用
- 如果 BOR 的原因是退出 SHUTDOWN 模式（请参阅关断模式处理），则锁存 IO 引脚状态

当 VDD > BOR0+ 时，BOR 始终会触发 BOOTRST。

2.4.1.1.3 引导复位 (BOOTRST) 复位电平

引导复位 (BOOTRST) 会触发器件引导配置例程的执行并复位大多数内核逻辑，包括 RTC、LFCLK、LFXT 和 LFCLK_IN 配置以及 LFXT 或 LFCLKIN 使用的任何 IO 引脚的 IOMUX 配置（除非 BOOTRST 是由 NRST 或软件触发的 BOOTRST 引起）以及 SYSOSC FCL 模式（如果启用）。系统存储器 (SRAM) 也会进行下电上电，并且 SRAM 内容丢失。

下面的条件会生成 BOOTRST：

- BOR
- 非 PMU 调整数据上的奇偶校验故障
- 致命时钟故障（请参阅 LFCLK 监测器和 MCLK 监测器）
- WWDT0 违例
- 软件会通过 SYSCTL 触发 BOOTRST *(RESETLEVEL 0x01)*
- 当处于 NRST 模式时，NRST 引脚保持低电平的时间长于最短复位脉冲时间，但少于一秒
- BOOTRST 后引导失败（重新尝试失败的引导序列）

BOOTRST 不会复位以下参数：

- 关断存储器 (SHUTDNSTOREx)
- NRST 禁用状态，如果被软件禁用
- SWD 禁用状态，如果被软件禁用

- 如果 **BOOTRST 的原因是 NRST、WWDT0 违例或软件触发的 BOOTRST**，则 RTC、LFCLK 和 LFXT 和 LFCLK_IN 配置，包括 LFXT 和 LFCLK_IN 使用的任何 IO 引脚的 IOMUX 设置。因此，如果 LFCLK 配置为从 LFXT 或 LFCLK_IN 运行，则 NRST 引脚 BOOTRST 条件或软件触发的 BOOTRST 条件不会将 LFCLK 重置为默认内部振荡器 (LFOSC)。LFCLK 继续从 LFXT 或 LFCLK_IN 运行，无法重新配置。这使得 RTC 通过一个外部复位来保持其时基。所有其他 BOOTRST 条件触发 RTC、LFCLK、LFXT 和 LFCLK_IN 状态的复位，以及 LFXT 或 LFCLK_IN 使用的任何 IO 引脚的 IOMUX 配置。

在 BOOTRST 之后，如果引导配置例程成功完成，SYSRST 将始终被触发。如果引导配置例程未能成功完成，则会再次生成 BOOTRST，并从 BOOTRST 点再次尝试引导过程。引导过程尝试成功完成最多 3 次，之后器件状态将锁定，直到发生 BOR 或 POR 复位（请参阅节 2.4.1.8）。

2.4.1.1.4 系统复位 (SYSRST) 复位级别

系统复位会清除 CPU 和所有外设的状态，但下列情况除外。

以下条件会产生 SYSRST：

- 在 BOOTRST 后引导成功
- 引导加载程序 (BSL) 退出，之后始终执行引导配置例程 (BCR)
- 闪存 ECC 不可纠正 (DED) 错误
- WWDT1 违例（如果存在）
- CPU 锁定违例
- 软件通过 SYSCTL 触发 SYSRST *(RESETLEVEL 0x00)*
- 软件通过 SYSCTL 使用 BSL 进入命令触发 SYSRST *(RESETLEVEL 0x02)*
- 调试子系统触发系统复位

SYSRST 不会复位以下参数：

- 关断存储器 (SHUTDNSTOREx)
- NRST 状态（如果被软件禁用）
- SWD 状态（如果被软件禁用）
- RTC、LFCLK 和 LFXT/LFCLK_IN 配置，包括配置为供 LFXT/LFCLK_IN 使用的任何 IO 引脚的 IOMUX 设置
- SYSOSC 频率校正环路 (FCL)（如果由软件启用）

在大多数情况下，器件在 SYSRST 之后处于 RUN 模式，并且 CPU 会执行复位矢量并开始执行应用软件。这种情况有两个例外：

- 如果使用 BSL 进入请求触发了 SYSRST，则会依次运行 BCR 和 BSL。
- 如果是由于 BSL 退出请求而触发了 SYSRST，则会运行 BCR，然后再一次执行 SYSRST，之后应用软件将启动。

2.4.1.1.5 仅 CPU 复位 (CPURST) 复位电平

一个仅 CPU 复位只清除 CPU 逻辑的状态。外设状态不受 CPU 复位的影响。CPU 复位仅由软件通过 CPU AIRCR 本地寄存器或调试子系统生成。

2.4.1.2 POR 之后的初始条件

POR 之后，当引导过程完成且 CPU 启动应用程序时，初始器件条件如下：

- NRST 引脚配置为 NRST 模式
- 串行线调试 (SWD) IO 处于 SWD 模式
- 所有其他可配置 I/O 引脚均为高阻抗（高阻态）
- 外设模块按照本手册中相应章节所述进行复位。
- 器件处于运行模式。
- MCLK 由基础频率 (32MHz) 的内部 SYSOSC 提供
- LFCLK 由内部 LFOSC 提供（请注意，LFOSC 需要时间启动才能使用 LFCLK）
- MFCLK 禁用
- MFPCLK 禁用

- HFXT 和 SYSPLL 都禁用
- 外设禁用
- 任何配置为在引导时受写保护的闪存扇区都受写保护

2.4.1.3 NRST 引脚

冷启动后，NRST 引脚配置为 NRST 模式。NRST 引脚必须为高电平才能成功引导器件。NRST 上没有内部上拉电阻。外部电路（上拉电阻连接至 VDD，或复位控制电路）必须主动将 NRST 拉至高电平才能使器件启动。器件启动后，NRST 上持续时间短于 1 秒的低电平脉冲会触发 BOOTRST。如果 NRST 上的低电平脉冲保持时间超过 1 秒，则会触发 POR。

一些引脚数较少的器件支持将 NRST 引脚重新配置为 GPIO 引脚。请参阅器件特定的数据表中的引脚配置，了解 GPIO 是否与 NRST 共享功能。应用软件可以禁用 NRST 引脚的 NRST 功能，从而启用 GPIO 功能。要禁用 NRST，请将 EXRSTPIN 寄存器中的 DISABLE 位与 KEY 一起设置。然后将 IOMUX 配置为所需的功能。

禁用 NRST 引脚功能后，只能通过 POR 重新启用该功能。

备注

当 NRST 引脚与 I2C 开漏引脚共用时，用户系统必须确保在 I2C 总线上发生任何事务之前器件已上电并处于 I2C 模式。如果在此之前由于共享复位或 I2C SDA 线上的低电平信号而无意中复位了器件，则可能会导致器件复位。

为防止出现这种情况，应选择与 I2C 开漏 IO 共享的 NRST 引脚上的上拉电阻器，以便满足 I2C 上拉电阻器最小值和最大值要求。

2.4.1.4 SWD 引脚

所有器件上都有两个串行线调试 (SWD) 引脚：

- SWCLK（串行线时钟）
- SWDIO（串行线数据输入/输出）

冷启动后，SWD 引脚配置为 SWD 模式以允许建立调试连接。当不再需要调试支持时，可以在软件中将 SWD 引脚重新配置为通用 IO (GPIO)，以便在应用中使用这些引脚。要禁用 SWD 功能，请设置 SYSCTL 的 SWDCFG 寄存器中的 DISABLE 位以及 KEY 位。然后将 IOMUX 配置为所需的功能。

一旦禁用了 SWD 引脚功能，只能通过触发 POR 来重新启用这些功能。

2.4.1.5 在软件中生成复位

通过向 SYSCTL 发出适当的命令，软件可以生成软件 POR、软件 BOOTRST、具有引导加载程序 (BSL) 进入的软件 SYSRST 或软件 SYSRST。要发出复位，首先在 SYSCTL 的 RESETLEVEL 寄存器中选择所需的复位电平。然后，设置 RESETCMD 寄存器中的 GO 位与 KEY 值。

表 2-12. 软件生成的 SYSCTL 复位命令

等级	操作
0x0	软件 SYSRST
0x1	软件 BOOTRST
0x2	具有 BSL 进入的软件 SYSRST
0x3	软件 POR

通过设置 AIRCR 本地 CPU 寄存器的 SYSRESETREQ 位，还可以在 Cortex-M0+ CPU 内的软件中触发一个 CPU 专用复位 (CPURST)，此复位不会复位外设。更多信息，请参阅“CPU 子系统”一章。

从软件启动 BSL

软件触发的 BSL 进入 *(RESETLEVEL 0x02)* 是 SYSRST 的一个特殊情况，它为应用软件提供了一种机制来启动 ROM 引导加载程序 (BSL)。在运行模式下的正常软件执行期间，无法直接跳转到引导加载程序代码。当应用软件发出命令来通过软件触发进入 BSL *(RESETLEVEL 0x02)* 时，首先会生成一个 SYSRST，然后执行引导配置例程（用于身份验证），之后会启动 BSL（如果器件安全策略已将 BSL 配置为启用）。BSL 执行完成后，会发出第二个 SYSRST，并且 BCR 将执行。当 BCR 完成时，最终的 SYSTRST 会置为有效，以将系统控制返回到应用软件。不由 SYSRST 复位的任何系统配置将在整个过程中保持不变。因此，如果 RTC 可以在执行软件触发的 BSL 进入和退出期间继续运行而不中断。

2.4.1.6 复位原因

发生器件复位后，在硬件中会捕捉复位处理期间的最低级别复位原因，以便应用软件可以在启动应用程序时查询复位原因并采取任何适当的措施。最低级别复位原因会编码到 SYSCTL 复位原因寄存器的一个 5 位字段中。复位原因寄存器的内容总是在读取后被清除，如果读取后没有发生复位，则在读取后返回零。表 2-13 给出了复位原因编码。

表 2-13. 复位原因编码

复位				器件模块复位										
复位级别	原因 ID		复位原因	NRST/SWD 禁用	SHUTDN STOREx	内核稳压器	调试子系统	RTC、LFXT、LFCLK 状态	SRAM	BCR 执行	IOMUX	EVENT、DMA、FLASHCTL	外设	CPU
	0x00	0	自上次读取后无复位											
POR	0x01	1	VDD < POR- 违例	R	R	R	R	R	R	R	R	R	R	R
			PMU 修整奇偶校验故障											
			SHUTDNSTOREx 奇偶校验故障											
	0x02	2	NRST 引脚复位 (>1s)	R	R	R	R	R	R	R	R	R	R	R
	0x03	3	软件触发 POR	R	R	R	R	R	R	R	R	R	R	R
BOR	0x04	4	VDD < BOR- 违例			R	R	R	R	R	R	R	R	R
	0x05	5	从 SHUTDOWN 中唤醒			R	R	R	R	R	R(1)	R	R	R
	0x06	6	保留											
	0x07	7	保留											
BOOTRST	0x08	8	非 PMU 修整奇偶校验故障					R	R	R	R	R	R	R
	0x09	9	致命时钟故障					R	R	R	R	R	R	R
	0x0A	10	保留											
	0x0B	11	保留											
	0x0C	12	NRST 引脚复位 (<1s)						R	R	R(2)	R	R	R
	0x0D	13	软件触发 BOOTRST						R	R	R(2)	R	R	R
	0x0E	14	WWDT0 违例						R	R	R(2)	R	R	R
	0x0F	15	保留											

表 2-13. 复位原因编码 (continued)

复位				器件模块复位										
复位级别	原因 ID		复位原因	NRST/SWD 禁用	SHUTDN STOREx	内核稳压器	调试子系统	RTC、LFXT、LFCLK 状态	SRAM	BCR 执行	IOMUX	EVENT、DMA、FLASHCTL	外设	CPU
SYSRST	0x10	16	BSL 退出							R	R(2)	R	R	R
	0x11	17	BSL 进入							R	R(2)	R	R	R
	0x12	18	保留											
	0x13	19	WWDT1 违例								R(2)	R	R	R
	0x14	20	不可纠正的闪存 ECC 错误								R(2)	R	R	R
	0x15	21	CPULOCK 违例								R(2)	R	R	R
	0x16	22	保留											
	0x17	23	保留											
	0x18	24	保留											
	0x19	25	保留											
	0x1A	26	调试触发 SYSRST								R(2)	R	R	R
	0x1B	27	软件触发 SYSRST								R(2)	R	R	R
CPURST	0x1C	28	调试触发 CPURST											R
	0x1D	29	软件触发 CPURST											R
	0x1E	30	保留											
	0x1F	31	保留											

(1) 在退出 SHUTDOWN 模式的情况下，IOMUX 寄存器始终会复位，但 IO 本身会从进入 SHUTDOWN 的点开始保持其最后状态，直到用户清除 SYSCTL SHDNIOREL 寄存器中的 RELEASE 位。这使得应用软件能够在退出 SHUTDOWN 模式后释放 IO 之前重新配置 IOMUX 和任何相应的外设。请参阅关断模式处理和 IOMUX 唤醒。

(2) IOMUX 会复位，但如果启用了 LFXT 或 LFCLK_IN，那么这些焊盘的 IOMUX 设置不会复位。RTC、LFXT、LFCLK_IN 和 LFCLK 继续不间断运行。

如果同时出现两个复位原因，则优先选择并报告最低的原因复位 ID 值。例如，如果 WWDT0 违例（原因 0x12）与 VDD < BOR- 违例（原因 0x04）同时发生，则报告的复位原因为 BOR- 违例（原因 0x04），因为这是一个较低级别的复位，所以会清除器件状态的其他方面。

复位原因编码可在应用程序启动期间实现简单的软件处理。复位原因值可由应用软件读取并测试是否处于特定的值范围内，从而确定是否出现以下原因：

- **RESETCAUSE==0x00**：自上次读取后无复位
- **RESETCAUSE<0x04**：NRST/SWD 禁用状态已复位，可能需要重新配置
- **RESETCAUSE<0x04**：SHUTDNSTOREx 存储器已复位，可能需要重新配置
- **RESETCAUSE<0x08**：对经过稳压的 VCORE 域（包括 SRAM）进行了下电上电
- **RESETCAUSE<0x0B**：RTC 和低频时钟配置已复位，可能需要重新配置
- **RESETCAUSE<0x1C**：外设已复位，可能需要重新配置

以下示例说明了如何测试复位原因，以便在复位后启动应用程序时采取特定的操作：

```
// Read reset cause into SRAM variable
uint8_t cause = RESETCAUSE;

// Handle device re-configuration based on reset cause level
```

```
if (cause!=0)
{
    if (cause<0x04)
    {
        // NRST/SWD disable state was lost
        // SHUTDNSTOREx memory state was lost
        // PMU/VCORE domain state was lost
        // RTC/LFXT/LFCLK state was lost
    }
    if (cause<0x0B)
    {
        // RTC/LFXT/LFCLK state was lost
    }
    if (cause<0x1C)
    {
        // The peripherals were reset
    }
}
```

2.4.1.7 外设复位控制

器件上的每个外设都包含一个复位控制寄存器 (RSTCTL) 和一个状态寄存器 (STAT)。

STAT 寄存器是一个只读寄存器，包含一个用于指示外设是否被复位的 RESETSTKY 位。应用软件可以读取该位以确定外设是否已复位并需要重新配置。通过将 RESETSTKYCLR 位与 KEY 值一起写入 RSTCTL 寄存器即可清除 RESETSTKY 位。

应用软件还可以通过将 RESETASSERT 位与 KEY 值一起写入 RSTCTL 寄存器来强制外设复位。此操作会将外设重置为默认状态，并会设置 STAT 寄存器中的 RESETSTKY 位。

2.4.1.8 引导失败处理

如果在执行引导配置例程 (BCR) 期间引导失败，SYSCTL 将使 BOOTRST 有效，以尝试另一个引导。引导失败可能由以下原因引起：

- 引导配置数据完整性错误
- 器件修整完整性错误
- BCR 超时（由于任何其他原因，BCR 所需的时间明显长于预期）

硬件最多会尝试三次成功引导器件。如果第一次、第二次或第三次引导尝试成功，应用程序将正常启动。如果第三次尝试失败，则引导过程失败，不再尝试引导，并且应用程序软件未启动。

如果导致引导失败的原因是瞬态（临时）错误，则额外引导尝试的目的是允许器件正确引导。如果三次引导尝试不成功，则可能会出现稳态错误情况，并且不会启动应用程序以防止意外操作。

备注

如果器件因三次引导尝试失败而被锁定，并且发生了一个 BOR- 违例，则仍会生成一个 BOR 和 BOOTRST（根据定义）并进行一次引导尝试。在同样的条件下，如果器件完全断电（触发 POR- 违例），那么器件会再次尝试启动最多三次。

2.4.2 选择工作模式

通过以下方式配置器件工作模式：

1. 使用 SYSCTL 的 SYSOSCCFG 和 MCLKCFG 寄存器中的策略位（用于控制 SYSOSC 在 RUN、SLEEP 和 STOP 模式下的行为）
2. 使用 SYSCTL 的 PMODECFG 寄存器中的策略位（用于设置 STOP、STANDBY 或 SHUTDOWN 模式的深度睡眠级别）
3. 使用 SCR 本地 CPU 寄存器中的 SLEEPDEEP 策略位（用于选择 WFI 指令是触发 SLEEP 模式还是 STOP/STANDBY/SHUTDOWN 模式）
4. 使用 Arm WFI（等待中断）CPU 指令（进入配置的 SLEEP/STOP/STANDBY/SHUTDOWN 状态）

在进入禁用 CPU 的工作模式之前，请确保可将 CPU 从睡眠状态中唤醒的相应外设已配置为在发生所需事件时生成 CPU 中断。

有关每种工作模式的行为的详细说明，请参阅工作模式一节。

策略位配置

表 2-14 说明了如何为每种工作模式配置相关的策略位。所有值均以二进制格式表示。短横线 (-) 表示特定的策略位与指定的工作模式无关。

表 2-14. 工作模式策略位配置

工作模式策略控制		RUN			SLEEP(2)			STOP			STANDBY		SHUTDOWN
寄存器	位	RUN0	RUN1	RUN2	SLEEP0	SLEEP1	SLEEP2	STOP0	STOP1	STOP2(3)	STANDBY0	STANDBY1	
SYSOSCCFG	DISABLE(1)	0	0	1	0	0	1	-	-	(1)	-	-	-
	USE4MHZSTOP	-	-	-	-	-	-	0	1	0	-	-	-
	DISABLESTOP	-	-	-	-	-	-	0	0	1	-	-	-
MCLKCFG	USELFCLK(1)	0	1	-	0	1	-	0	0	-	-	-	-
	STOPCLKSTBY	-	-	-	-	-	-	-	-	-	0	1	-
PMODECFG	DSLEEP	-	-	-	-	-	-	00	00	00	01	01	10
SCR	SLEEPDEEP	0	0	0	0	0	0	1	1	1	1	1	1

(1) SYSOSCCFG.DISABLE 和 MCLKCFG.USELFCLK 策略位在配置后立即生效，因为这些位会影响 RUN 模式行为。其他策略位仅在 CPU 进入深度睡眠时生效。
(2) 除了禁用 CPUCLK 外，SLEEP 模式的行为始终与 RUN 模式相同。因此，SLEEP 模式的行为由 RUN 模式的配置决定。
(3) 可以通过在进入 DEEPSLEEP 之前设置 SYSOSCCFG 寄存器中的 DISABLESTOP 位或 DISABLE 位，来配置 STOP 模式的 STOP2 策略。如果已设置 DISABLESTOP 且已清除 DISABLE，则仅当请求 DEEPSLEEP 时才会禁用 SYSOSC。SYSOSC 继续在 RUN 和 SLEEP 模式下运行。如果已设置 DISABLE，则 DISABLESTOP 将变为无关状态，并且 SYSOSC 会立即被禁用，并在 STOP 模式下保持禁用状态。

进入 SLEEP 模式

进入 **SLEEP** 模式会禁用 CPU，但其他情况下会保持与 **RUN** 模式相同的配置。要进入 **SLEEP** 模式，请执行以下操作：

1. 通过清除 Cortex-M0+ SCR 本地寄存器中的 SLEEPDEEP 位，配置 Cortex-M0+ CPU 以用于 SLEEP 模式
2. 通过执行 WFI (等待中断) CPU 指令进入睡眠模式

进入 STOP 或 STANDBY 模式

要进入 **STOP** 或 **STANDBY** 模式，请执行以下操作：

1. 将 SYSCTL 中的 PMODECFG 寄存器配置为 0b00 (STOP) 或 0b01 (STANDBY)
2. 通过设置 Cortex-M0+ SCR 本地寄存器中的 SLEEPDEEP 位，配置 Cortex-M0+ CPU 以用于 DEEP SLEEP 模式
3. 通过执行 WFI (等待中断) CPU 指令进入睡眠模式

进入 SHUTDOWN 模式

要进入 **SHUTDOWN** 模式，请执行以下操作：

1. 将 SYSCTL 中的 PMODECFG 寄存器配置为 0b10 (SHUTDOWN)
2. 通过设置 Cortex-M0+ SCR 本地寄存器中的 SLEEPDEEP 位，配置 Cortex-M0+ CPU 以用于 DEEP SLEEP 模式
3. 通过执行 WFI (等待中断) CPU 指令进入睡眠模式

2.4.3 异步快速时钟请求

外设配置为使硬件 SYSCTL 请求异步有效以实现一个快速时钟源，即使器件在 STOP 或 STANDBY 模式下运行也是如此。这种机制非常适合 MCLK/ULPCLK 树通常源自 LFCLK（32kHz 时）或 SYSOSC（4MHz 时）的应用，但临时需要更快的时钟来快速处理外设事件（例如，计时器 IRQ 或 GPIO IRQ）或外设活动（例如串行通信或 ADC 转换）。

异步快速时钟请求对于器件以 STANDBY1 模式运行的情况也很有用。在 STANDBY1 下（STOPCLKSTBY 置位），对除 TIMG0 和 TIMG1 之外的所有外设禁用 ULPCLK 和 LFCLK，并将 TIMG0 和 TIMG1 及 RTC 保留为唯一时钟外设。为了将器件从总线时钟 (ULPCLK) 被禁用的状态下唤醒，TIMG0 和 TIMG1 或 RTC 中断请求会强制异步快速时钟请求将器件唤醒至 RUN 模式。如果其他外设支持检测异步事件（例如 GPIO、比较器和串行接口），它们也可以将器件从该状态唤醒。

异步快速时钟请求在请求时段内临时为外设提供一个 32MHz 总线时钟 (MCLK/ULPCLK)，该时钟源自 SYSOSC。MFCLK 如被允许使用，则在异步请求期间也会被启用。

异步快速时钟行为

配置后，SYSCTL 将通过以下方式响应外设快速时钟请求：

1. 如果器件当前处于 STOP 或 STANDBY 模式，低功耗状态将暂时挂起，从而支持以 SYSOSC 基频 (32MHz) 运行总线时钟 (ULPCLK)。
2. 如果 SYSOSC 被禁用，则会被强制启用；如果 SYSOSC 已经在运行，但频率与基频不同，则会被强制使用基频 (32MHz)
3. MCLK/ULPCLK 树被强制源自 SYSOSC，频率为 32MHz；如果器件处于 RUN 模式，CPUCLK 也会切换到 SYSOSC 频率（CPUCLK 始终源自 MCLK）
4. 如果 MFCLK 配置为使用，它将被激活

应用上述配置后，它将保持一段时间，即异步请求保持有效及额外 32 个 SYSOSC 周期（大约 1 µs）的时间。删除了该请求之后的 32 个 SYSOSC 周期，如果 CPU 在请求期间未更改配置，系统将返回到快速时钟请求之前存在的配置。

如果满足以下条件之一，则会忽略异步快速时钟请求，且不会对器件配置产生影响：

- MCLK 源自 SYSOSC，频率为基频 (32MHz)
- MCLK 源自 HSCLK（HFCLK 或 SYSPLL）
- 通过设置 SYSCTL 中 SYSOSCCFG 寄存器内的 BLOCKASYNCALL 位，可以全局阻止异步快速时钟请求

外设支持

RTC、TIMG0、TIMG1、GPIO、比较器、SPI、I2C、UART 和 ADC 外设均支持生成异步快速时钟请求。表 2-15 中提供了这些外设的用途、请求源和配置要求。

表 2-15. 针对异步快速时钟请求的外设支持

外设	用途	请求源	配置
RTC	从 RTC 事件中快速唤醒 CPU	RTC IRQ 至 CPU	当器件处于 STANDBY1 模式时，RTC IRQ 事件始终会生成异步快速时钟请求；由于主 ULPCLK 被禁用以降低功耗，因此唤醒器件时需要此功能。RTC 还可以通过清零 RTC CLKCFG 寄存器中的 BLOCKASYNC 位配置为在任何工作模式下生成异步快速时钟请求；这将提供极低延迟的 RTC 事件响应。
TIMG0 和 TIMG1	从 TIMG0/TIMG1 事件中快速唤醒 CPU	TIMG0 或 TIMG1 到 CPU 的 IRQ	当器件处于 STANDBY1 模式并且在 TIMG 寄存器中设置相应的 IMASK 中断时，来自 TIMG0 或 TIMG1 的 IRQ 事件会生成异步快速时钟请求。这是唤醒器件所必需的，因为 ULPCLK 被禁用以降低功耗。

表 2-15. 针对异步快速时钟请求的外设支持 (continued)

外设	用途	请求源	配置
GPIO	从 GPIO 事件中快速唤醒 CPU	GPIO 活动	GPIO 通过 GPIO 配置寄存器生成异步快速时钟请求。这非常适合需要将 GPIO 从 STANDBY 模式唤醒的应用，因为快速时钟请求会导致 GPIO 数字干扰滤波器以 32MHz 的速率运行。除了将 GPIO 寄存器配置为请求快速时钟，SYSOSCCFG 寄存器中的 BLOCKASYNCALL 位必须清零才能允许请求传播。
比较器	从比较器事件中快速唤醒	比较器事件	比较器事件会生成异步快速时钟请求，以提供延迟极低的比较器事件响应。必须禁用相应比较器 CLKCFG 寄存器中的 BLOCKASYNC 位。
SPI	暂时使用快速时钟生成位时钟	SPI 活动	当相应 SPI 外设的 CLKCFG 寄存器中的 BLOCKASYNC 位清零时，SPI 活动会生成异步快速时钟请求。
I2C	暂时使用快速时钟生成位时钟	I2C 活动	当相应 I2C 外设的 CLKCFG 寄存器中的 BLOCKASYNC 位清零时，I2C 活动会生成异步快速时钟请求。
UART	暂时使用快速时钟来生成波特率	UART 活动	当相应 UART 外设的 CLKCFG 寄存器中的 BLOCKASYNC 位清零时，UART 活动会生成异步快速时钟请求。
ADC	在低功耗模式下临时运行 SYSOSC 以支持计时器触发的 ADC 运行	ADC	如果在禁用 SYSOSC 时触发 ADC 转换，则会生成异步快速时钟请求以启用 SYSOSC（需要 SYSOSC 才能实现正确的 ADC 运行）。

快速 CPU 事件处理

除了外设事件和活动快速时钟请求触发器之外，SYSCTL 还可以配置为在向 CPU 发出任何 IRQ 请求时生成异步快速时钟请求。当系统以 LFCLK 速率 (32kHz) 运行时，这将提供极低延迟的中断处理，因为 IRQ 请求将以 SYSOSC 速率 (32MHz) 与 LFCLK 速率 (32kHz) 通过唤醒逻辑进行传播。当在 SYSCTL 中的 SYSOSCCFG 寄存器内设置 FASTCPUEVENT 位时，向 CPU 发出的任何中断请求也会生成快速时钟请求。

异步快速时钟请求逻辑

图 2-11 中提供了使快速时钟请求有效的逻辑。

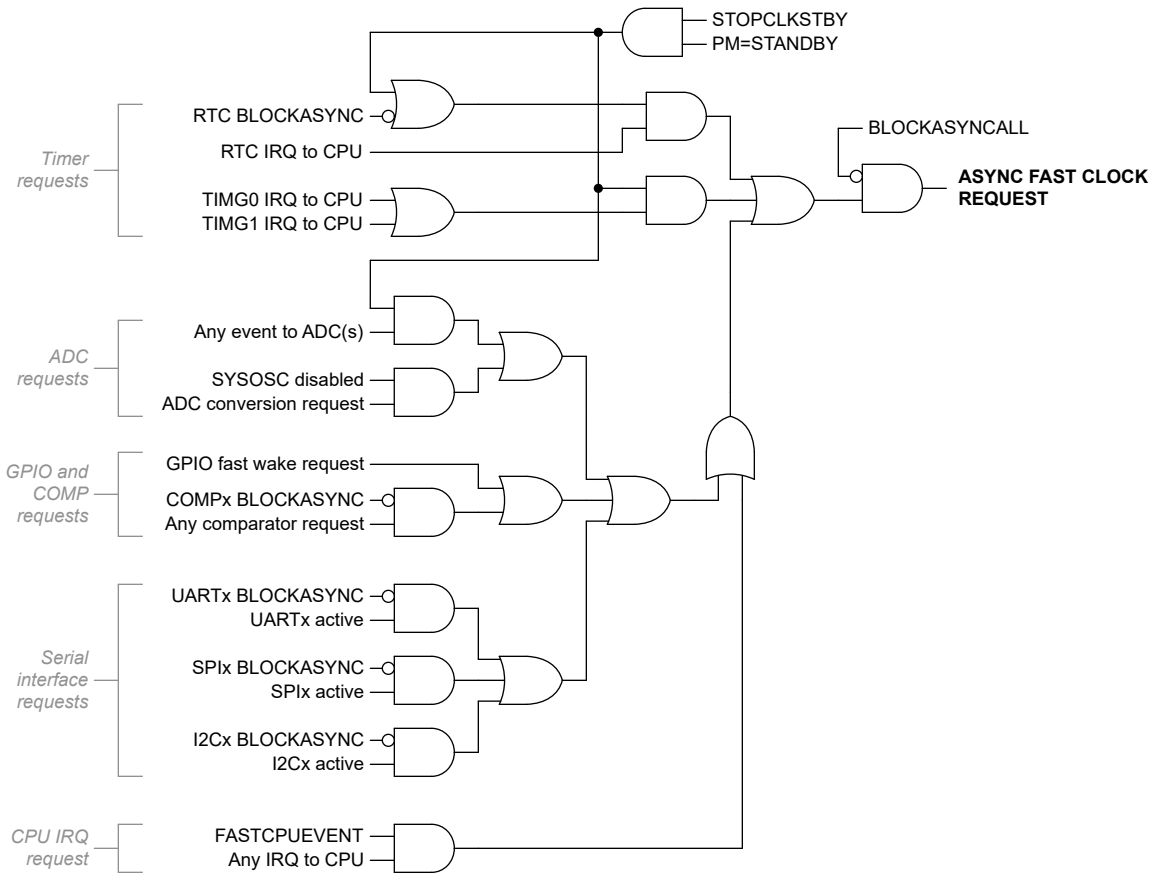


图 2-11. MSPM0Gxx 异步快速时钟请求逻辑

2.4.4 SRAM 写保护

某些应用程序需要将只读数据放入 SRAM。如果将代码放入 SRAM（用于零等待状态执行）或者将关键查询表放入 SRAM（用于零等待状态读取），则会发生这种情况。在这些情况下，尤其是从 SRAM 执行代码时，需要防止意外写入 SRAM 地址，以免在缓冲区溢出或堆栈溢出的情况下损坏可执行代码。同样，最好防止从不受写保护的 SRAM 地址执行。为了提高存储在 SRAM 中的数据的稳健性，SYSCTL 提供了一种写独占执行边界机制。

要使用此特性，首先需要将只读数据加载到所需的 SRAM 地址中，然后将 SRAM 地址范围配置为受写保护。要进行读取/执行（无写入）的 SRAM 内容应置于 SRAM 的上部。要进行读取/写入（无执行）的 SRAM 内容应置于 SRAM 的下部。然后，SRAMBOUNDRY 寄存器中可以写入所需的边界以将 SRAM 分为两个区域，下部区域为 RW，上部区域为 RX。

2.4.5 闪存等待状态

从 SYSOSC 或 LFCLK 运行 MCLK 时，SYSCTL 会自动管理闪存等待状态。如果 MCLK 不会切换到 SYSPLL、HFXT 或 HFCLK_IN，则无需等待状态配置。

如果要将 MCLK 配置为从 SYSPLL、HFXT 或 HFCLK_IN 等高速时钟源之一运行，则应用 MCLKCFG.FLASHWAIT 中的闪存等待状态配置。默认情况下，MCLKCFG.FLASHWAIT 设置为 0x2（2 个等待状态），支持在 80MHz 的最大 MCLK 频率下运行。如果将 MCLK 配置为从高速时钟运行，但 MCLK 频率允许在少于 2 个等待状态的情况下运行，则可以减少 MCLKCFG.FLASHWAIT。

请参阅器件特定数据表的*建议运行条件*部分，以确定等待状态为 0 或 1 时支持的最大时钟频率。

2.4.6 闪存存储体地址交换

具有多个闪存存储体的器件提供了一种将上部存储体地址空间与下部存储体地址空间交换的机制，从而在固件映像本身无需了解其所在存储体的情况下启用固件更新，以便能够在硬件上正确执行。

要将上部闪存存储体的地址与下部闪存存储体的地址交换，请在写入 KEY 值的同时设置 FLBANKSWAP 寄存器中的 USEUPPER 位。

交换闪存存储体地址空间时需要特别注意。请参阅本指南的非易失性存储器系统一章，了解使用闪存存储体地址交换时的注意事项。

2.4.7 SHUTDOWN 模式处理

在配置器件并进入 SHUTDOWN 模式时，内核稳压器将断电，器件寄存器内容和 SRAM 内容将丢失。退出 SHUTDOWN 模式会产生 BOR 级别的复位。在进入 SHUTDOWN 模式时可以使用两种机制保留器件状态：IO 锁存和小型关断存储器。

关断 IO 状态

数字 IO 引脚状态（输出低电平/高电平、上拉/下拉、高阻态、驱动配置）在进入 SHUTDOWN 模式时会锁存和保持。退出 SHUTDOWN 模式后，IO 保持上一状态，直到应用软件通过设置 SHDNIOREL 寄存器中的 RELEASE 位以及匹配的 KEY 值来释放该状态。退出 SHUTDOWN 模式时，应用软件必须先将 IO 重新配置为其正确状态，然后释放 IO。为了在启动时确定复位原因是否为退出 SHUTDOWN 模式，应用软件必须读取 SYSCTL 中的 RSTCAUSE 寄存器。

备注

退出 SHUTDOWN 模式时，**串行线调试 (SWD) 引脚也将保持锁定状态，直到应用软件设置 RELEASE 位**。因此，从 SHUTDOWN 模式唤醒时无法建立调试连接，直到应用软件释放 IO。

备注

退出 SHUTDOWN 模式时，引导加载程序 (BSL) 调用引脚必须保持在逻辑低电平，防止在退出 SHUTDOWN 模式期间意外进入 BSL。在退出 SHUTDOWN 模式期间进入 BSL 将阻止应用程序代码启动，BSL 接口将不可用，并且无法进行 SWD 连接，因为 IO 状态会在退出 SHUTDOWN 模式后保持锁存，直到应用软件释放 IO。

关断存储器

为了在进入 SHUTDOWN 模式之前保存应用程序状态信息，SYSCTL 中提供了 4 字节的关断存储器。这些存储器位置在 SHUTDOWN 模式下会保留，在退出 SHUTDOWN 模式后可由应用程序读取。要将数据保存到关断存储器中，请写入到 SYSCTL 中的 SHUTDNSTORE0-3 寄存器。

2.4.8 配置锁定

SYSCTL 中的配置寄存器可被锁定以防止写入，从而增加一层稳健性，防止在运行时对 PMCU 进行意外更改。要锁定配置寄存器以防止写入，请设置 SYSCTL 的 WRITELOCK 寄存器中的 ACTIVE 位。

除下列寄存器外，所有 SYSCTL 寄存器都受 WRITELOCK 功能保护：

- WRITELOCK
- PMODECFG
- FCC、FCCCMD
- FLBANKSWAP
- RSTCAUSE（读取以清除）、RESETLEVEL、RESETCMD
- BORTHRESHOLD、BORCLRCMD
- SHDNIOREL
- PMUOPAMP

- SHUTDNSTOREx

除了整个 SYSCTL 配置写入锁定特性外，许多 SYSCTL 寄存器还需要将一个 KEY 值与所需的配置数据一起写入才能使写入生效。

2.4.9 系统状态

软件可以通过读取 SYSCTL 中的 CLKSTATUS 和 SYSSTATUS 寄存器来查询 PMCU 各方面的状态。

检查时钟状态 (CLKSTATUS)

SYSCTL 中的 CLKSTATUS 寄存器是一个只读寄存器，用于指示时钟模块的当前配置和状态。CLKSTATUS 中提供的主要状态信息包括：

- 当前的 SYSOSC 频率
- 当前的 HSCLK 选择
- 当前的 LFCLK 选择
- 当前的 MCLK 选择
- HFCLK 和 SYSPLL 状态
- LFXT 状态
- LFOSC 状态
- HSCLK、HFCLK 和 SYSPLL 被禁用的状态指示
- 在外设请求了时钟但无法生成时钟时出现的错误指示

此状态信息可用于验证请求的时钟更改是否已成功完成，或在 SYSOSC 可能具有异步激活或外设发出频率请求的应用中检查真实的 SYSOSC 频率。

检查系统状态 (SYSSTATUS)

SYSCTL 中的 SYSSTATUS 寄存器是一个只读寄存器，用于指示闪存 ECC 错误（SED 和 DED）以及其他特定于外设的状态信息。SYSTATUS 中的 ECC 错误位具有粘性（即使将来的读取没有错误，这些位仍保持在发生 ECC 错误时的置位状态）。通过设置 SYSSTATUSCLR 寄存器中的 ALLECC 位以及 KEY 值可以将这些位复位（清除）。

2.4.10 错误处理

MSPM0 器件包含多种诊断机制，用于在运行时检测错误。表 2-16 列出了错误源及其相应的处理机制。

备注

并非所有 MSPM0 器件都支持所有诊断功能。例如，有些器件在存储器上没有 ECC/奇偶校验，有些器件没有双看门狗计时器。请始终参阅特定于器件的数据表，了解给定器件可以使用哪些诊断功能。在 PMCU 寄存器部分，还为每个 MCU 子系列提供了寄存器映射，详细说明了给定器件可用的特定寄存器。

表 2-16. 错误源和处理机制

错误源	出错了	处理机制
闪存（如果器件具有 ECC）	不可纠正的 ECC 错误（如果器件有 ECC）	• 对于 CPU 或 DMA 请求，会向处理器生成 FLASHDED 不可屏蔽中断或生成 SYSRST，具体取决于 FLASHECCRSTDIS 位的配置 • 在 SYSCTL 的 SYSSTATUS 寄存器中设置 FLASHDED 粘滞位
	可纠正的 ECC 错误（如果器件有 ECC）	• SYSCTL 中也会生成 FLASHSEC 中断 • 在 SYSCTL 的 SYSSTATUS 寄存器中设置 FLASHSEC 粘滞位

表 2-16. 错误源和处理机制 (continued)

错误源	出错了	处理机制
SRAM	不可纠正的 ECC 错误（如果器件有 ECC）	• 向处理器生成 SRAMDED 不可屏蔽中断
	可纠正的 ECC 错误（如果器件有 ECC）	• 向处理器生成 SYSCTL SRAMSED 中断
	奇偶校验错误（如果器件有奇偶校验）	• 如果请求来自 CPU，则向处理器生成不可屏蔽中断 • 如果请求来自 DMA，则生成 DMA 数据错误中断
	CPU 访问时的地址错误	• CPU 中生成硬故障
	DMA 访问时的地址错误	• DMA 控制器中生成 DMA 地址错误中断
	CAN SRAM 上的 ECC 错误（如果器件具有 CAN-FD）	• CAN-FD 外设中生成中断
SHUTDNSTOREx 存储器	奇偶校验错误	• 生成 POR
CKM	MCLK 失效	• 生成 BOOTRST
	LFCLK 失效（如果存在）	• 如果 LFCLK 作为 MCLK 的时钟源，则生成 BOOTRST • SYSCTL NMI 寄存器中生成 LFCLKFAIL 不可屏蔽中断。
CPUSS（如果器件具有 MPU）	存储器保护单元违例	• CPU 中生成硬故障
WWDT	WWDT0 违例	• 根据 WWDTLP0RSTDIS 位的配置，在 SYSCTL NMI 寄存器中生成 BOOTRST 或生成不可屏蔽中断
	WWDT1 违例（如果存在）	• 根据 WWDTLP1RSTDIS 位的配置，在 SYSCTL NMI 寄存器中生成 BOOTRST 或生成不可屏蔽中断
PMU	调整奇偶校验错误	• 生成 POR
	POR0- 电源错误	• 生成 POR
	BOR0- 电源错误	• 生成 BOR
	BOR1/2/3- 电源错误	• SYSCTL NMI 寄存器中生成 BORLVL 不可屏蔽中断
CPUSS	存储器保护单元违例（如果存在）	• CPU 中生成硬故障

可配置 NMI 触发器

错误源可以配置为触发不可屏蔽中断或不同的处理机制。SYSCTL 中的 SYSTEMCFG 寄存器可用于指定所需的错误处理机制。例如，WWDT0 可以配置为生成 BOOTRST 或 NMI，默认情况下生成 BOOTRST。有关可用的错误处理选项，请参阅相关器件子系列的 SYSTEMCFG 寄存器。

2.4.11 SYSCTL 事件

SYSCTL 模块包含两个事件发布者，不包含事件订阅者。一个事件发布者管理针对 CPU 子系统的 SYSCTL 中断请求 (IRQ)。另一个发布者管理针对 CPU 子系统的不可屏蔽中断以进行关键诊断。

SYSCTL 事件中汇总了 SYSCTL 事件。

表 2-17. SYSCTL 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	SYSCTL	CPU 子系统	静态路由	SYSCTL 中断寄存器	修复了从 SYSCTL 到 CPU 的中断路由
CPU 不可屏蔽中断 (NMI)	发布者	SYSCTL	CPU 子系统	静态路由	NMI 中断寄存器	修复了从 SYSCTL 到 CPU 的中断路由

2.4.11.1 CPU 中断事件 (CPU_INT)

SYSCTL 模块提供七个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 2-18 中列出了来自 SYSCTL 的 CPU 中断事件。

表 2-18. SYSCTL CPU 中断事件源

索引 (IIDX)	名称	说明
0	无	无中断挂起。
1	LFOSCGOOD	指示在启动期间 LFOSC 何时准备就绪，因为 LFOSC 需要大约 1ms 的启动时间。
2	ANACLKERR	表示模拟功能已启用并期望 SYSOSC 在特定频率下运行，但 SYSOSC 不可用或未在所需频率下运行。
3	FLASHSEC	表示检测到闪存单个位可纠正的错误
4	SRAMSEC	表示检测到 SRAM single-bit 可纠正的错误
5	LFXTGOOD	指示低频外部时钟（LFXT 振荡器或 LFCLK_IN 数字时钟）何时准备就绪。当启动时钟系统并等待 LFXT 或 LFCK_IN 在将 LFCLK 源从 LFOSC 切换到外部源之前准备好时，此指示很有用。
6	HFCLKGOOD	指示高频外部时钟（HFXT 振荡器或 HFCLK_IN 数字时钟）何时准备就绪。当启动时钟系统并等待 HFCLK 准备就绪，然后再将 MCLK 源切换到 HFCLK 或以 HFCLK 作为 SYSPLL 基准启动 SYSPLL 之前，该指示很有用。
7	SYSPLLGOOD	指示 SYSPLL 何时准备就绪且可供使用。当启动时钟系统并等待 SYSPLL 准备好后再将 MCLK 源切换到 SYSPLL 时，该指示很有用。
8	HSCLKGOOD	指示 HSCLK（由 HFCLK 或 SYSPLL 输出提供）何时准备就绪。该指示在从 STOP 或 STANDBY 模式唤醒时很有用（如果 HSCLK 配置为 RUN/SLEEP 模式下的 MCLK 源）。当从 STOP 或 STANDBY 唤醒时，MCLK 将从 SYSOSC 运行，直到 HSCLK 可用，此时 SYSCTL 自动将 MCLK 源切换到选定的 HSCLK 源。该中断会表明从 STOP/STANDBY 唤醒后，MCLK 已切换到所需的 HSCLK 源，现在可以使用由 MCLK 提供的时序敏感外设。

CPU 中断事件配置可通过 SYSCTL IIDX、IMASK、RIS、MIS、ISET 和 ICLR 事件管理寄存器进行管理。有关为 CPU 中断配置这些寄存器的指导，请参阅节 7.2.5。

2.4.11.2 不可屏蔽中断事件 (NMI)

SYSCTL 模块提供 x 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 2-18 中列出了来自 SYSCTL 的 CPU 中断事件。

表 2-19. 不可屏蔽中断事件表

索引 (IIDX)	名称	说明
0	无	无 NMI 挂起。
1	BORLVL	表示 VDD 已降至指定的 VBOR- 阈值以下。
2	WWDT0	发生 WWDT0 违例。
3	WWDT1	发生 WWDT1 违例。

表 2-19. 不可屏蔽中断事件表 (continued)

索引 (IIDX)	名称	说明
4	LFCLKFAIL	指示 LFXT 或 LFCLK_IN 时钟源已死。如果 LFCLK 不是作为 MCLK 源而是为外设（例如，RTC）提供源，此指示对于处理 LFCLK 错误很有用
5	FLASHDED	表示检测到闪存 double-bit 不可纠正的错误。
6	SRAMDED	表示检测到 SRAM double-bit 不可纠正的错误。

CPU 不可屏蔽中断事件配置由 NMIIIDX、NMIRIS、NMIISET 和 NMIICLR 事件管理寄存器管理。有关配置中断管理寄存器的指导，请参阅节 7.2.5。

2.5 快速入门参考

PMCU 旨在提供简单易用的电源管理、时钟和复位管理功能。本节介绍 PMCU 的基本工作原理，以及通过复位获取默认配置并针对特定应用进行优化的提示和技巧。

2.5.1 默认器件配置

器件的默认运行配置提供基本功能，无需修改即可适用于许多应用。

当外部电源（VDD 和 VSS）达到 1.62V 时，MSPM0Gxx 器件上电并释放复位，以执行应用代码。当释放应用代码用于执行时，器件处于 RUN 模式，MCLK 源自内部 SYSOSC，频率为 32MHz。CPUCLK 和 ULPCLK 也是 32MHz，源自 MCLK。LFCLK 自动启动，源自内部 LFOSC。在具有默认配置的 RUN 模式下，所有外设均可启用。DMA、CRC 和 AES 等外设以 MCLK 速率直接通过 MCLK 运行。其他外设（如计时器和串行接口）可根据其外设时钟选择，通过 32MHz 总线时钟或 32kHz 低频时钟 (LFCLK) 运行。

可通过进入 SLEEP、STOP、STANDBY 或 SHUTDOWN 模式来降低功耗。默认情况下，这些模式的运行如下：

- 在 SLEEP 模式下，CPUCLK 被禁用，但包括 DMA 在内的所有外设继续按照配置以 32MHz 或 32kHz 的频率运行。此电源模式非常适合使用 PD1 外设的情况，并且具有尽可能低的唤醒延迟比具有极低功耗更重要。
- 在 STOP 模式（默认为 STOP0）下，PD1 外设的 MCLK 源被禁用，PD1 外设被禁用并处于保留模式（不可使用）。默认 STOP 模式非常适合电源优化很重要但需要 ADC、OPA、DAC 或频率高于 32kHz 的时钟的情况。
  - 默认情况下，SYSOSC 将继续以 32MHz 的频率运行，但 MCLK 树将以 4MHz (SYSOSC/8) 的频率运行，仍处于运行状态的 PD0 外设的总线时钟 (ULPCLK) 将从 32MHz 更改为 4MHz。
  - 配置为通过 LFCLK 运行的 PD0 外设继续以 32kHz 的频率运行。
  - 所有 OPA 模式均可使用
  - ADC 的 SYSOSC 将始终以 32MHz 的频率进行采样
- 在 STANDBY 模式（默认为 STANDBY0）下，源自 LFCLK 的 MCLK 树以 32kHz 的频率运行，SYSOSC 被禁用。通过总线时钟运行的 PD0 外设更改为 32kHz。通过 LFCLK 运行的 PD0 外设以 32kHz 的频率继续运行，没有变化。

2.5.2 利用 MFCLK

当使用默认 PMCU 配置运行时，计时器和串行接口可以选择总线时钟 (MCLK/ULPCLK) 或 LFCLK 作为其时钟源。LFCLK 在运行、睡眠、停止和待机模式下始终为 32kHz，但 MCLK/ULPCLK 在停止模式下更改为 4MHz，而在待机模式下更改为 32kHz，这意味着，从总线时钟运行的外设在转换功耗模式时会看到其源时钟频率变化。

相比之下，MFCLK 的工作方式与 LFCLK 类似，因为它可为运行、睡眠和停止模式下的外设提供恒定频率时钟源。MFCLK 提供恒定的 4MHz，作为以 32kHz 频率运行的 LFCLK 的替代方案。MFCLK 的 4MHz 时基始终源自 SYSOSC。外设（尤其是可在停止模式下使用的 PD0 外设）可以选择 MFCLK 作为其时钟源，而不是 ULPCLK。MFCLK 在运行、睡眠和停止模式下将保持为 4MHz，非常适合 UART、I2C 和低功耗计时器等需要一致的时钟但要求时钟源大于 32kHz 的外设。

有关 MFCLK 使用的信息，请参阅 MFCLK 部分。

2.5.3 优化 STOP 模式下的功耗

STOP 模式为根据应用的特定功率和性能要求来定制器件提供了相当大的灵活性。在 STOP 模式下有几种方法可以降低功耗：

- 在 STOP 中启用 SYSOSC 换档模式以减小 SYSOSC 电流。在 STOP 模式下，MCLK 树（和 ULPCLK）始终被限制在最高 4MHz。默认情况下，SYSOSC 以 32MHz（基础频率）在 STOP 模式下运行，并进行 8 分频以满足 STOP 模式下 4MHz 的最高频率限制。或者，可以将 SYSOSC 进行换档，在 STOP 模式 (STOP1) 下以 4MHz 本机运行，从而节省 SYSOSC 电流。请参阅 SYSOSC 换档模式。
- 如果 32kHz 时钟足以运行所需的外设，则可以在 STOP 模式下以 32kHz 的频率运行来自 LFCLK 的 MCLK。可以禁用 SYSOSC，以便节能。要在 STOP 模式下禁用 SYSOSC 并从 LFCLK (STOP2) 运行，请参阅禁用 SYSOSC 和选择工作模式。

2.5.4 优化 STANDBY 模式下的功耗

在 STANDBY 模式下，如果只需要 RTC、TIMG0、TIMG1 或从 GPIO、比较器（低功耗模式）或串行接口异步快速唤醒，则可以将 ULPCLK 和 LFCLK 配置为在进入 STANDBY 模式时被禁用，仅保持 RTC、TIMG0 和 TIMG1 为运行状态 (STANDBY1)，从而实现尽可能低的功耗。请参阅 LFCLK 一节。在此状态下，RTC、TIMG0、TIMG1 或异步活动/事件将触发异步快速时钟请求以唤醒系统。

2.5.5 提高 MCLK 和 ULPCLK 精度

如果应用需要为高频外设提供高时钟精度，则可通过使用具有 HFXT 外部高频晶体并直接从 HFCLK 为 MCLK 树提供时钟源来实现最佳精度。通过直接从 HFCLK 提供 MCLK，PD1 和 PD0 中所有外设的总线时钟将为 HFCLK。

HFXT 支持高达 48MHz 的晶体频率，但由于使用 ULPCLK 的 PD0 外设限制为 40MHz，因此 40MHz 是以相同频率运行 PD1 和 PD0 外设时受支持的最高晶体频率。如果使用 48MHz 晶体，则 PD1 外设和 CPUCLK 可直接从晶体以 48MHz 的频率运行，但 PD0 外设必须以 MCLK/2 (24MHz) 的频率运行。

如果 CAN-FD 控制器或 ADC（用于采样窗口生成）需要精密时钟，并且还需要高 CPU 性（高 MCLK），也可以直接从 HFCLK 获取 CAN-FD 控制器 (CANCLK) 和 ADC 采样时钟 (ADCCLK) 并与 MCLK 异步，同时使用 PLL 以最大频率运行 MCLK 以获得最佳计算性能。

2.5.6 配置 MCLK 以获得最大速度

通过使用 SYSPLL 从 SYSOSC 基准或 HFXT 生成 80MHz 时钟，可获得出色的 CPU 计算性能。要将 SYSPLL 配置为生成 80MHz 输出源 MCLK，请参阅 SYSPLL 配置部分。

以 80MHz 运行 MCLK 还为 PD1 域中的 TIMA 和 TIMG 外设提供了理想的计时器分辨率 (12.5ns)。

2.5.7 低功耗模式下的高速时钟（SYSPLL、HFCLK）处理

停止和待机工作模式下不支持 SYSPLL 和 HFCLK（HFXT、HFCLK_IN）高速时钟源。如果启用了高速时钟源（SYSPLL、HFCLK），则进入停止模式或待机模式时，SYSCTL 会自动在进入停止或待机模式之前禁用 SYSPLL 和/或 HFCLK。从停止或待机模式退出并进入运行模式时，如果 SYSPLL 和/或 HFCLK 在进入低功耗模式之前处于启用状态，则 SYSCTL 将自动重新启用 SYSPLL 和/或 HFCLK。

在启用 SYSPLL 或 HFCLK 后首次进入停止或待机模式之前，以及从停止或待机模式唤醒之后，应用软件必须等待进入停止或待机模式，直到之前启用的任何高速时钟源完成启动。

应用软件必须在进入停止或待机模式之前检查以下内容：

- 如果 **SYSPLL 已启用**，则应用软件必须等待 SYSPLL 完成启动。当 SYSPLL 启动完成时，将设置 SYSCTL 内 CLKSTATUS 寄存器中的 SYSPLLGOOD 位。如果 SYSPLL 无法启动，则将设置 SYSPLLOFF 位。SYSPLLOFF 位指示 SYSPLL 在启动时发生故障或之前未启用。在尝试进入停止或待机模式之前，请确保已设置 SYSPLLGOOD 或 SYSPLLOFF。
- 如果 **HFCLK 已启用**，则应用软件必须等待 HFCLK 完成启动。当 HFCLK 启动完成时，则将设置 SYSCTL 内 CLKSTATUS 寄存器中的 HFCLKGOOD 位。如果 HFCLK 无法启动，则将设置 HFCLKOFF 位。HFCLKOFF

位指示 HFCLK 在启动时发生故障或之前未启用。在尝试进入停止或待机模式之前，确保已设置 HFCLKGOOD 或 HFCLKOFF。

如果 MCLK 配置为在进入停止或待机状态之前从 HSCLK 中获取，则在退出停止或待机状态时，MCLK 将在最初时从 SYSOSC 中获取，并且 CPU 将被释放以开始以 SYSOSC 频率执行代码。当高速时钟已启动并准备就绪时，SYSCTL 会自动将 MCLK 配置恢复到之前选择的高速时钟。当 MCLK 切换回高速时钟时，SYSCTL 将生成一个 HSCLK GOOD 中断，以提醒应用 MCLK 再次从高速时钟运行。

2.5.8 通过优化实现最低唤醒延迟

为了确保从 STOP 或 STANDBY 模式到 RUN 模式的唤醒延迟尽可能低，请在进入 STOP 或 STANDBY 模式之前将 MCLK 设置为 SYSOSC 并使 SYSOSC 以基础频率 (32MHz) 运行。SYSOSC 始终以基础频率启动，如果 SYSOSC 不需要更改为备用频率，则会尽可能降低延迟。

2.5.9 通过优化在 RUN/SLEEP 模式下实现最低峰值电流

在峰值电流受限的应用中，有两种方法可以降低 RUN 和 SLEEP 模式下的有功电流：

- 如果在 32kHz 频率下可以提供足够的性能，则从 LFCLK 运行 MCLK。在禁用 SYSOSC 的情况下，可以选择从 LFCLK 运行 MCLK。如果不需要快速处理事件，可禁用 SYSOSC 异步请求，以确保器件始终从 LFCLK 运行。这样便可在 CPU 仍在运行 (RUN2) 的情况下提供尽可能低的电流。请参阅 MCLK、SYSOSC 和选择工作模式部分
- 如果在 32kHz 频率下不能提供足够的性能，则可以选择从 SYSOSC 运行 MCLK，并将 SYSOSC 设置为低频率 (4MHz)。当 MCLK 从 SYSOSC 运行时，可以应用 MCLK 的 MDIV 分频器来降低电流消耗。例如，当从 4MHz 的 SYSOSC 运行 MCLK 时，可以通过将 MDIV 设置为 /16，将 MCLK 配置为以低至 250kHz 的频率运行。请参阅 MCLK 一节。

2.6 PMCU 寄存器

PMCU 的配置是通过 SYSCTL 中的存储器映射寄存器进行的，这些寄存器取决于器件系列，因为并非所有器件系列都具有所有特性。请使用特定器件系列的相关寄存器映射。

2.6.1 SYSCTL 寄存器

表 2-20 列出了 SYSCTL 寄存器的存储器映射寄存器。表 2-20 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 2-20. SYSCTL 寄存器

偏移	缩写	寄存器名称	部分
1020h	IIDX	SYSCTL 中断索引	节 2.6.1.1
1028h	IMASK	SYSCTL 中断屏蔽	节 2.6.1.2
1030h	RIS	SYSCTL 原始中断状态	节 2.6.1.3
1038h	MIS	SYSCTL 屏蔽中断状态	节 2.6.1.4
1040h	ISET	SYSCTL 中断设置	节 2.6.1.5
1048h	ICLR	SYSCTL 中断清除	节 2.6.1.6
1050h	NMIIIDX	NMI 中断索引	节 2.6.1.7
1060h	NMIRIS	NMI 原始中断状态	节 2.6.1.8
1070h	NMIISET	NMI 中断设置	节 2.6.1.9
1078h	NMIICLR	NMI 中断清除	节 2.6.1.10
1100h	SYSOSCCFG	SYSOSC 配置	节 2.6.1.11
1104h	MCLKCFG	主时钟 (MCLK) 配置	节 2.6.1.12
1108h	HSCLKEN	高速时钟 (HSCLK) 源启用/禁用	节 2.6.1.13
110Ch	HSCLKCFG	高速时钟 (HSCLK) 源选择	节 2.6.1.14
1110h	HFCLKCLKCFG	高频时钟 (HFCLK) 配置	节 2.6.1.15
1114h	LFCLKCFG	低频晶体振荡器 (LFXT) 配置	节 2.6.1.16
1120h	SYSPLLCFG0	SYSPLL 基准和输出配置	节 2.6.1.17
1124h	SYSPLLCFG1	SYSPLL 基准和反馈分频器	节 2.6.1.18
1128h	SYSPLLPARAM0	SYSPLL PARAM0 (从 FACTORY 区域加载)	节 2.6.1.19
112Ch	SYSPLLPARAM1	SYSPLL PARAM1 (从 FACTORY 区域加载)	节 2.6.1.20
1138h	GENCLKCFG	通用时钟配置	节 2.6.1.21
113Ch	GENCLKEN	通用时钟使能控制	节 2.6.1.22
1140h	PMODECFG	电源模式配置	节 2.6.1.23
1150h	FCC	频率时钟计数器 (FCC) 计数	节 2.6.1.24
1170h	SYSOSCTRIMUSER	SYSOSC 用户指定的修整	节 2.6.1.25
1178h	SRAMBOUNDARY	SRAM 写边界	节 2.6.1.26
1180h	SYSTEMCFG	系统配置	节 2.6.1.27
1200h	WRITELOCK	SYSCTL 寄存器写锁定	节 2.6.1.28
1204h	CLKSTATUS	时钟模块 (CKM) 状态	节 2.6.1.29
1208h	SYSSTATUS	系统状态信息	节 2.6.1.30
120Ch	DEDERRADDR	存储器 DED 地址	节 2.6.1.31
1220h	RSTCAUSE	复位原因	节 2.6.1.32
1300h	RESETLEVEL	应用触发的复位命令的复位电平	节 2.6.1.33
1304h	RESETCMD	执行应用触发的复位命令	节 2.6.1.34
1308h	BORTHRESHOLD	BOR 阈值选择	节 2.6.1.35
130Ch	BORCLRCMD	设置 BOR 阈值	节 2.6.1.36
1310h	SYSOSCFCLCTL	SYSOSC 频率校正环路 (FCL) ROSC 使能	节 2.6.1.37
1314h	LFXTCTL	LFXT 和 LFCLK 控制	节 2.6.1.38
1318h	EXLFCTL	LFCLK_IN 和 LFCLK 控制	节 2.6.1.39

表 2-20. SYSCTL 寄存器 (continued)

偏移	缩写	寄存器名称	部分
131Ch	SHDNIOREL	SHUTDOWN IO 释放控制	节 2.6.1.40
1320h	EXRSTPIN	禁用 NRST 引脚的复位功能	节 2.6.1.41
1324h	SYSSTATUSCLR	清除 SYSSTATUS 的粘滞位	节 2.6.1.42
1328h	SWDCFG	禁用 SWD 引脚上的 SWD 功能	节 2.6.1.43
132Ch	FCCCMD	频率时钟计数器开始捕捉	节 2.6.1.44
1380h	PMUOPAMP	GPAMP 控制	节 2.6.1.45
1400h	SHUTDNSTORE0	关断存储内存 (字节 0)	节 2.6.1.46
1404h	SHUTDNSTORE1	关断存储内存 (字节 1)	节 2.6.1.47
1408h	SHUTDNSTORE2	关断存储内存 (字节 2)	节 2.6.1.48
140Ch	SHUTDNSTORE3	关断存储内存 (字节 3)	节 2.6.1.49

复杂的位访问类型经过编码可适应小型表单元。表 2-21 展示了适用于此部分中访问类型的代码。

表 2-21. SYSCTL 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
RC	R C	读取 以清除
写入类型		
W	W	写入
W1C	W 1C	写入 1 以进行清除
W1S	W 1S	写入 1 以进行设置
复位或默认值		
-n		复位后的值或默认值

2.6.1.1 IIDX 寄存器（偏移 = 1020h）[复位 = X]

图 2-12 展示了 IIDX，表 2-22 中对此进行了介绍。

返回到汇总表。

SYSCTL 中断索引

图 2-12. IIDX 寄存器

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

表 2-22. IIDX 寄存器字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	X	
3-0	STAT	R	0h	SYSCTL 中断索引 (IIDX) 寄存器生成一个与最高优先级挂起中断源相对应的值。该值可用作中断服务例程中快速、确定性处理的地址偏移量。读取 IIDX 寄存器将清除 RIS 和 MIS 寄存器中相应的中断状态。 0h = 无中断挂起 1h = LFOSCGOOD 中断挂起 2h = 2 3h = 3 4h = 4 5h = 5 6h = 6 7h = 7 8h = 8

2.6.1.2 IMASK 寄存器（偏移 = 1028h）[复位 = X]

图 2-13 展示了 IMASK，表 2-23 中对此进行了介绍。

返回到汇总表。

SYSCTL 中断屏蔽

图 2-13. IMASK 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
HSCLKGOOD	SYSPLLGOOD	HFCLKGOOD	LFXTGOOD	SRAMSEC	FLASHSEC	ANACLKERR	LFOSCGOOD
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 2-23. IMASK 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	X	
7	HSCLKGOOD	R/W	0h	HSCLK GOOD 0h = 0 1h = 1
6	SYSPLLGOOD	R/W	0h	SYSPLL GOOD 0h = 0 1h = 1
5	HFCLKGOOD	R/W	0h	HFCLK GOOD 0h = 0 1h = 1
4	LFXTGOOD	R/W	0h	LFXT GOOD 0h = 0 1h = 1
3	SRAMSEC	R/W	0h	SRAM 单错校正 0h = 0 1h = 1
2	FLASHSEC	R/W	0h	闪存单错校正 0h = 0 1h = 1
1	ANACLKERR	R/W	0h	模拟时钟一致性错误 0h = 0 1h = 1
0	LFOSCGOOD	R/W	0h	启用或禁用 LFOSCGOOD 中断。LFOSCGOOD 表示已成功启动 LFOSC。 0h = 中断已禁用 1h = 中断已启用

2.6.1.3 RIS 寄存器（偏移 = 1030h）[复位 = X]

图 2-14 展示了 RIS，表 2-24 中对此进行了介绍。

返回到汇总表。

SYSCTL 原始中断状态

图 2-14. RIS 寄存器

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留							
R-X							
7	6	5	4	3	2	1	0
HSCLKGOOD	SYSPLLGOOD	HFCLKGOOD	LFXTGOOD	SRAMSEC	FLASHSEC	ANACLKERR	LFOSCGOOD
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 2-24. RIS 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	X	
7	HSCLKGOOD	R	0h	HSCLK GOOD 0h = 0 1h = 1
6	SYSPLLGOOD	R	0h	SYSPLL GOOD 0h = 0 1h = 1
5	HFCLKGOOD	R	0h	HFCLK GOOD 0h = 0 1h = 1
4	LFXTGOOD	R	0h	LFXT GOOD 0h = 0 1h = 1
3	SRAMSEC	R	0h	SRAM 单错校正 0h = 0 1h = 1
2	FLASHSEC	R	0h	闪存单错校正 0h = 0 1h = 1
1	ANACLKERR	R	0h	模拟时钟一致性错误 0h = 0 1h = 1
0	LFOSCGOOD	R	0h	LFOSCGOOD 中断的原始状态。 0h = 无中断待处理 1h = 中断待处理

2.6.1.4 MIS 寄存器（偏移 = 1038h）[复位 = X]

图 2-15 展示了 MIS，表 2-25 中对此进行了介绍。

返回到汇总表。

SYSCTL 屏蔽中断状态

图 2-15. MIS 寄存器

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留							
R-X							
7	6	5	4	3	2	1	0
HSCLKGOOD	SYSPLLGOOD	HFCLKGOOD	LFXTGOOD	SRAMSEC	FLASHSEC	ANACLKERR	LFOSCGOOD
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 2-25. MIS 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	X	
7	HSCLKGOOD	R	0h	HSCLK GOOD 0h = 0 1h = 1
6	SYSPLLGOOD	R	0h	SYSPLL GOOD 0h = 0 1h = 1
5	HFCLKGOOD	R	0h	HFCLK GOOD 0h = 0 1h = 1
4	LFXTGOOD	R	0h	LFXT GOOD 0h = 0 1h = 1
3	SRAMSEC	R	0h	SRAM 单错校正 0h = 0 1h = 1
2	FLASHSEC	R	0h	闪存单错校正 0h = 0 1h = 1
1	ANACLKERR	R	0h	模拟时钟一致性错误 0h = 0 1h = 1
0	LFOSCGOOD	R	0h	LFOSCGOOD 中断的屏蔽状态。 0h = 无中断待处理 1h = 中断待处理

2.6.1.5 ISET 寄存器（偏移 = 1040h）[复位 = X]

图 2-16 展示了 ISET，表 2-26 中对此进行了介绍。

返回到汇总表。

SYSCTL 中断设置

图 2-16. ISET 寄存器

31	30	29	28	27	26	25	24
保留							
W-X							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
HSCLKGOOD	SYSPLLGOOD	HFCLKGOOD	LFXTGOOD	SRAMSEC	FLASHSEC	ANACLKERR	LFOSCGOOD
W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h

表 2-26. ISET 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	X	
7	HSCLKGOOD	W1S	0h	HSCLK GOOD 0h = 0 1h = 1
6	SYSPLLGOOD	W1S	0h	SYSPLL GOOD 0h = 0 1h = 1
5	HFCLKGOOD	W1S	0h	HFCLK GOOD 0h = 0 1h = 1
4	LFXTGOOD	W1S	0h	LFXT GOOD 0h = 0 1h = 1
3	SRAMSEC	W1S	0h	SRAM 单错校正 0h = 0 1h = 1
2	FLASHSEC	W1S	0h	闪存单错校正 0h = 0 1h = 1
1	ANACLKERR	W1S	0h	模拟时钟一致性错误 0h = 0 1h = 1
0	LFOSCGOOD	W1S	0h	设置 LFOSCGOOD 中断。 0h = 写入 0h 不产生影响 1h = 设置中断

2.6.1.6 ICLR 寄存器（偏移 = 1048h）[复位 = X]

图 2-17 展示了 ICLR，表 2-27 中对此进行了介绍。

返回到汇总表。

SYSCTL 中断清除

图 2-17. ICLR 寄存器

31	30	29	28	27	26	25	24
保留							
W-X							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
HSCLKGOOD	SYSPLLGOOD	HFCLKGOOD	LFXTGOOD	SRAMSEC	FLASHSEC	ANACLKERR	LFOSCGOOD
W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h

表 2-27. ICLR 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	X	
7	HSCLKGOOD	W1C	0h	HSCLK GOOD 0h = 0 1h = 1
6	SYSPLLGOOD	W1C	0h	SYSPLL GOOD 0h = 0 1h = 1
5	HFCLKGOOD	W1C	0h	HFCLK GOOD 0h = 0 1h = 1
4	LFXTGOOD	W1C	0h	LFXT GOOD 0h = 0 1h = 1
3	SRAMSEC	W1C	0h	SRAM 单错校正 0h = 0 1h = 1
2	FLASHSEC	W1C	0h	闪存单错校正 0h = 0 1h = 1
1	ANACLKERR	W1C	0h	模拟时钟一致性错误 0h = 0 1h = 1
0	LFOSCGOOD	W1C	0h	清除 LFOSCGOOD 中断。 0h = 写入 0h 不产生影响 1h = 清除中断

2.6.1.7 NMIIIDX 寄存器（偏移 = 1050h）[复位 = X]

图 2-18 展示了 NMIIIDX，表 2-28 中对此进行了介绍。

返回到汇总表。

NMI 中断索引

图 2-18. NMIIIDX 寄存器

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

表 2-28. NMIIIDX 寄存器字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	X	
3-0	STAT	R	0h	NMI 中断索引 (NMIIIDX) 寄存器生成一个与最高优先级挂起 NMI 源相对应的值。该值可用作 NMI 服务例程中快速、确定性处理的地址偏移量。读取 NMIIIDX 寄存器将清除 NMIRIS 寄存器中相应的中断状态。 0h = 无 NMI 待处理 1h = BOR 阈值 NMI 待处理 2h = 2 3h = 3 4h = 4 5h = 5 6h = 6

2.6.1.8 NMIRIS 寄存器（偏移 = 1060h）[复位 = X]

图 2-19 展示了 NMIRIS，表 2-29 中对此进行了介绍。

返回到汇总表。

NMI 原始中断状态

图 2-19. NMIRIS 寄存器

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留							
R-X							
7	6	5	4	3	2	1	0
RESERVED		SRAMDED	FLASHDED	LFCLKFAIL	WWDT1	WWDT0	BORLVL
R-X		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 2-29. NMIRIS 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	X	
5	SRAMDED	R	0h	SRAM 双错检测 0h = 0 1h = 1
4	FLASHDED	R	0h	闪存双错检测 0h = 0 1h = 1
3	LFCLKFAIL	R	0h	LFXT-EXLF 监控失败 0h = 0 1h = 1
2	WWDT1	R	0h	看门狗 0 故障 0h = 0 1h = 1
1	WWDT0	R	0h	看门狗 0 故障 0h = 0 1h = 1
0	BORLVL	R	0h	BORLVL NMI 的原始状态 0h = 无中断待处理 1h = 中断待处理

2.6.1.9 NMIISET 寄存器（偏移 = 1070h）[复位 = X]

图 2-20 展示了 NMIISET，表 2-30 中对此进行了介绍。

返回到汇总表。

NMI 中断设置

图 2-20. NMIISET 寄存器

31	30	29	28	27	26	25	24
保留							
W-X							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
RESERVED		SRAMDED	FLASHDED	LFCLKFAIL	WWDT1	WWDT0	BORLVL
W-X		W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h	W1S-0h

表 2-30. NMIISET 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	X	
5	SRAMDED	W1S	0h	SRAM 双错检测 0h = 0 1h = 1
4	FLASHDED	W1S	0h	闪存双错检测 0h = 0 1h = 1
3	LFCLKFAIL	W1S	0h	LFXT-EXLF 监控失败 0h = 0 1h = 1
2	WWDT1	W1S	0h	看门狗 0 故障 0h = 0 1h = 1
1	WWDT0	W1S	0h	看门狗 0 故障 0h = 0 1h = 1
0	BORLVL	W1S	0h	设置 BORLVL NMI 0h = 写入 0h 不产生影响 1h = 设置中断

2.6.1.10 NMIICLR 寄存器（偏移 = 1078h）[复位 = X]

图 2-21 展示了 NMIICLR，表 2-31 中对此进行了介绍。

返回到汇总表。

NMI 中断清除

图 2-21. NMIICLR 寄存器

31	30	29	28	27	26	25	24
保留							
W-X							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
RESERVED		SRAMDED	FLASHDED	LFCLKFAIL	WWDT1	WWDT0	BORLVL
W-X		W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h	W1C-0h

表 2-31. NMIICLR 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	X	
5	SRAMDED	W1C	0h	SRAM 双错检测 0h = 0 1h = 1
4	FLASHDED	W1C	0h	闪存双错检测 0h = 0 1h = 1
3	LFCLKFAIL	W1C	0h	LFXT-EXLF 监控失败 0h = 0 1h = 1
2	WWDT1	W1C	0h	看门狗 0 故障 0h = 0 1h = 1
1	WWDT0	W1C	0h	看门狗 0 故障 0h = 0 1h = 1
0	BORLVL	W1C	0h	清除 BORLVL NMI 0h = 写入 0h 不产生影响 1h = 清除中断

2.6.1.11 SYSOSCCFG 寄存器（偏移 = 1100h）[复位 = X]

图 2-22 展示了 SYSOSCCFG，表 2-32 中对此进行了介绍。

返回到汇总表。

SYSOSC 配置

图 2-22. SYSOSCCFG 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留						FASTCPUEVENT	BLOCKASYNCALL
R/W-X						R/W-1h	R/W-0h
15	14	13	12	11	10	9	8
保留					DISABLE	DISABLESTOP	USE4MHZSTOP
R/W-X					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
保留						FREQ	
R/W-X						R/W-0h	

表 2-32. SYSOSCCFG 寄存器字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R/W	X	
17	FASTCPUEVENT	R/W	1h	FASTCPUEVENT 可用于在将 CPU 中断置为有效时使快速时钟请求有效，从而降低中断延迟。 0h = CPU 中断将不会使快速时钟请求有效 1h = CPU 中断将使快速时钟请求有效
16	BLOCKASYNCALL	R/W	0h	BLOCKASYNCALL 可用于屏蔽所有异步快速时钟请求，防止硬件在给定模式下运行时动态更改活动的时钟配置。 0h = 异步快速时钟请求由请求外设控制 1h = 阻止所有异步快速时钟请求
15-11	RESERVED	R/W	X	
10	DISABLE	R/W	0h	DISABLE 设置 SYSOSC 启用/禁用策略。SYSOSC 可在 RUN、SLEEP 和 STOP 模式下关闭，以便降低功耗。当 SYSOSC 禁用时，MCLK 和 ULPCLK 以 LFCLK 为时钟源。 0h = 不禁用 SYSOSC 1h = 立即禁用 SYSOSC，并使 MCLK 和 ULPCLK 以 LFCLK 为时钟源
9	DISABLESTOP	R/W	0h	DISABLESTOP 设置 SYSOSC STOP 模式启用/禁用策略。在 STOP 模式下运行时，可以自动禁用 SYSOSC。设置后，ULPCLK 将在 STOP 模式下从 LFCLK 运行，并将禁用 SYSOSC 以降低功耗。 0h = 在 STOP 模式下不禁用 SYSOSC 1h = 在 STOP 模式下禁用 SYSOSC 并使 ULPCLK 以 LFCLK 为时钟源
8	USE4MHZSTOP	R/W	0h	USE4MHZSTOP 设置 SYSOSC STOP 模式频率策略。进入 STOP 模式时，SYSOSC 频率可自动切换至 4MHz 以降低 SYSOSC 功耗。 0h = 在 STOP 模式下不将 SYSOSC 换档至 4MHz 1h = 在 STOP 模式下将 SYSOSC 换档至 4MHz
7-2	RESERVED	R/W	X	

表 2-32. SYSOSCCFG 寄存器字段说明 (continued)

位	字段	类型	复位	说明
1-0	FREQ	R/W	0h	系统振荡器的目标工作频率 (SYSOSC) 0h = 基频 (32MHz) 1h = 低频 (4MHz) 2h = 用户修整的频率（16MHz 或 24MHz）

2.6.1.12 MCLKCFG 寄存器（偏移 = 1104h）[复位 = X]

图 2-23 展示了 MCLKCFG，表 2-33 中对此进行了介绍。

返回到汇总表。

主时钟 (MCLK) 配置

图 2-23. MCLKCFG 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留	MCLKDEADCHK	STOPCLKSTBY	USELFCLK	保留			USEHSCLK
R/W-X	R/W-0h	R/W-0h	R/W-0h	R/W-X			R/W-0h
15	14	13	12	11	10	9	8
保留			USEMFTICK	FLASHWAIT			
R/W-X			R/W-0h	R/W-2h			
7	6	5	4	3	2	1	0
RESERVED		UDIV		MDIV			
R/W-X		R/W-1h		R/W-0h			

表 2-33. MCLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-23	保留	R/W	X	
22	MCLKDEADCHK	R/W	0h	MCLKDEADCHK 启用或禁用连续 MCLK 死区检查监视器。启用 MCLKDEADCHK 之前，LFCLK 必须正在运行。 0h = MCLK 死区检查监视器已禁用 1h = MCLK 死区检查监视器已启用
21	STOPCLKSTBY	R/W	0h	STOPCLKSTBY 设置待机模式策略（STANDBY0 或 STANDBY1）。设置后，所有处于待机模式的外设都禁用 ULPCLK 和 LFCLK，但 TIMG0 和 TIMG1 除外，它们会继续运行。只能通过异步快速时钟请求实现唤醒。 0h = 所有处于待机模式的 PD0 外设都运行 ULPCLK/LFCLK 1h = 除 TIMG0 和 TIMG1 外，所有处于待机模式的外设都禁用 ULPCLK/LFCLK
20	USELFCLK	R/W	0h	USELFCLK 设置 MCLK 源策略。设置 USELFCLK 以使用 LFCLK 作为 MCLK 源。请注意，设置 USELFCLK 不会禁用 SYSOSC，SYSOSC 仍可供模拟模块直接使用。 0h = MCLK 将不使用低频时钟 (LFCLK) 1h = MCLK 将使用低频时钟 (LFCLK)
19-17	保留	R/W	X	
16	USEHSCLK	R/W	0h	USEHSCLK 与 USELFCLK 一起设置 MCLK 源策略。设置 USEHSCLK 以使用 HSCLK（HFCLK 或 SYSPLL）作为 RUN 和 SLEEP 模式下的 MCLK 源。 0h = MCLK 将不使用高速时钟 (HSCLK) 1h = MCLK 将在 RUN 和 SLEEP 模式下使用高速时钟 (HSCLK)
15-13	RESERVED	R/W	X	
12	USEMFTICK	R/W	0h	USEMFTICK 指定是启用还是禁用外设的 4MHz 恒定速率时钟 (MFCLK)。启用时，必须禁用 MDIV（设置为 0h=/1）。 0h = 外设的 4MHz 速率 MFCLK 已禁用 1h = 外设的 4MHz 速率 MFCLK 已启用。

表 2-33. MCLKCFG 寄存器字段说明 (continued)

位	字段	类型	复位	说明
11-8	FLASHWAIT	R/W	2h	FLASHWAIT 指定当 MCLK 以 HSCLK 为时钟源时的闪存等待状态数。当 MCLK 以 SYSOSC 或 LFCLK 为时钟源时，FLASHWAIT 不产生影响。 0h = 未应用闪存等待状态 1h = 应用 1 个闪存等待状态 2h = 应用 2 个闪存等待状态
7-6	RESERVED	R/W	X	
5-4	UDIV	R/W	1h	当 MCLK 以 HSCLK 为时钟源时，UDIV 指定 ULPCLK 分频器。当 MCLK 以 SYSOSC 或 LFCLK 为时钟源时，UDIV 不产生影响。 0h = ULPCLK 未进行分频且等于 MCLK 1h = ULPCLK 为 MCLK/2（进行 2 分频） 2h = ULPCLK 为 MCLK/3（进行 3 分频）
3-0	MDIV	R/W	0h	当 MCLK 以 SYSOSC 为时钟源时，MDIV 可用于对 MCLK 频率进行分频。MDIV=0h 对应于 /1（不分频）。MDIV=1h 对应于 /2（2 分频）。MDIV=Fh 对应于 /16（16 分频）。MDIV 可设置为 /1 和 /16 之间的整数。

2.6.1.13 HSCLKEN 寄存器（偏移 = 1108h）[复位 = X]

图 2-24 展示了 HSCLKEN，表 2-34 对其进行了介绍。

返回到汇总表。

高速时钟 (HSCLK) 源启用/禁用

图 2-24. HSCLKEN 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED							USEEXTHFCLK
R/W-X							R/W-0h
15	14	13	12	11	10	9	8
RESERVED							SYSPLLEN
R/W-X							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							HFXTEN
R/W-X							R/W-0h

表 2-34. HSCLKEN 寄存器字段说明

位	字段	类型	复位	说明
31-17	保留	R/W	X	
16	USEEXTHFCLK	R/W	0h	USEEXTHFCLK 选择 HFCLK_IN 数字时钟输入作为 HFCLK 的源。禁用时，HFXT 是 HFCLK 源，并可以设置 HFXTEN。请勿同时设置 HFXTEN 和 USEEXTHFCLK。 0h = 使用 HFXT 作为 HFCLK 源 1h = 使用 HFCLK_IN 数字时钟输入作为 HFCLK 源
15-9	保留	R/W	X	
8	SYSPLLEN	R/W	0h	SYSPLLEN 可启用或禁用系统锁相环 (SYSPLL)。 0h = 禁用 SYSPLL 1h = 启用 SYSPLL
7-1	RESERVED	R/W	X	
0	HFXTEN	R/W	0h	HFXTEN 可启用或禁用高频晶体振荡器 (HFXT)。 0h = 禁用 HFXT 1h = 启用 HFXT

2.6.1.14 HSCLKCFG 寄存器（偏移 = 110Ch）[复位 = X]

图 2-25 展示了 HSCLKCFG，表 2-35 对其进行了介绍。

返回到汇总表。

高速时钟 (HSCLK) 源选择

图 2-25. HSCLKCFG 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED							HSCLKSEL
R/W-X							R/W-0h

表 2-35. HSCLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	X	
0	HSCLKSEL	R/W	0h	HSCLKSEL 选择 HSCLK 源（SYSPLL 或 HFCLK）。 0h = HSCLK 以 SYSPLL 为时钟源 1h = HSCLK 以 HFCLK 为时钟源

2.6.1.15 HFCLKCLKCFG 寄存器（偏移 = 1110h）[复位 = X]

图 2-26 展示了 HFCLKCLKCFG，表 2-36 对其进行了介绍。

返回到汇总表。

高频时钟 (HFCLK) 配置

图 2-26. HFCLKCLKCFG 寄存器

31	30	29	28	27	26	25	24
RESERVED			HFCLKFLTCHK	保留			
R/W-X			R/W-1h	R/W-X			
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED		HFXTRSEL		保留			
R/W-X		R/W-0h		R/W-X			
7	6	5	4	3	2	1	0
HFXTTIME							
R/W-0h							

表 2-36. HFCLKCLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	X	
28	HFCLKFLTCHK	R/W	1h	HFCLKFLTCHK 可启用或禁用 HFCLK 启动监控器。 0h = 未检查 HFCLK 启动 1h = 检查 HFCLK 启动
27-14	保留	R/W	X	
13-12	HFXTRSEL	R/W	0h	HFXT 范围选择 0h = 4MHz ≤ HFXT 频率 ≤ 8MHz 1h = 8MHz < HFXT 频率 ≤ 16MHz 2h = 16MHz < HFXT 频率 ≤ 32MHz 3h = 32MHz < HFXT 频率 ≤ 48MHz
11-8	保留	R/W	X	
7-0	HFXTTIME	R/W	0h	HFXTTIME 以 64us 分辨率指定 HFXT 启动时间。如果启用 HFCLK 启动监控器 (HFCLKFLTCHK)，则在该时间到期后将检查 HFXT。 0h = 最短启动时间（大概为零） FFh = 最长启动时间（大概为 16.32ms）

2.6.1.16 LFCLKCFG 寄存器（偏移 = 1114h）[复位 = X]

图 2-27 展示了 LFCLKCFG，表 2-37 对其进行了介绍。

返回到汇总表。

低频晶体振荡器 (LFXT) 配置

图 2-27. LFCLKCFG 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
RESERVED							LOWCAP
R/W-X							R/W-0h
7	6	5	4	3	2	1	0
RESERVED			MONITOR	RESERVED		XT1DRIVE	
R/W-X			R/W-0h	R/W-X		R/W-3h	

表 2-37. LFCLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	X	
8	LOWCAP	R/W	0h	LOWCAP 控制低功耗 LFXT 模式。当 LFXT 负载电容小于 3pf 时，可以设置 LOWCAP 以降低功耗。 0h = LFXT 低电容模式已禁用 1h = LFXT 低电容模式已启用
7-5	RESERVED	R/W	X	
4	MONITOR	R/W	0h	MONITOR 启用或禁用 LFCLK 监控器，该监控器持续检查 LFXT 或 LFCLK_IN 是否存在时钟卡滞故障。 0h = 时钟监控器已禁用 1h = 时钟监控器已启用
3-2	RESERVED	R/W	X	
1-0	XT1DRIVE	R/W	3h	XT1DRIVE 选择低频晶体振荡器 (LFXT) 驱动强度。 0h = 更低驱动和电流 1h = 较低驱动和电流 2h = 较高驱动和电流 3h = 更高驱动和电流

2.6.1.17 SYSPLLCFG0 寄存器（偏移 = 1120h）[复位 = X]

图 2-28 展示了 SYSPLLCFG0，表 2-38 中对其进行了介绍。

返回到汇总表。

SYSPLL 基准和输出配置

图 2-28. SYSPLLCFG0 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED				RDIVCLK2X			
R/W-X				R/W-0h			
15	14	13	12	11	10	9	8
RDIVCLK1				RDIVCLK0			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED	ENABLECLK2X	ENABLECLK1	ENABLECLK0	RESERVED		MCLK2XVCO	SYSPLLREF
R/W-X	R/W-0h	R/W-0h	R/W-0h	R/W-X		R/W-0h	R/W-0h

表 2-38. SYSPLLCFG0 寄存器字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	X	
19-16	RDIVCLK2X	R/W	0h	RDIVCLK2X 为 SYSPLLCLK2X 输出设置最终分频器。 0h = SYSPLLCLK1 进行 1 分频 1h = SYSPLLCLK1 进行 2 分频 2h = SYSPLLCLK1 进行 3 分频 3h = SYSPLLCLK1 进行 4 分频 4h = SYSPLLCLK1 进行 5 分频 5h = SYSPLLCLK1 进行 6 分频 6h = SYSPLLCLK1 进行 7 分频 7h = SYSPLLCLK1 进行 8 分频 8h = SYSPLLCLK1 进行 9 分频 9h = SYSPLLCLK1 进行 10 分频 Ah = SYSPLLCLK1 进行 11 分频 Bh = SYSPLLCLK1 进行 12 分频 Ch = SYSPLLCLK1 进行 13 分频 Dh = SYSPLLCLK1 进行 14 分频 Eh = SYSPLLCLK1 进行 15 分频 Fh = SYSPLLCLK1 进行 16 分频

表 2-38. SYSPLLCFG0 寄存器字段说明 (continued)

位	字段	类型	复位	描述
15-12	RDIVCLK1	R/W	0h	RDIVCLK1 为 SYSPLLCLK1 输出设置最终分频器。 0h = SYSPLLCLK1 进行 2 分频 1h = SYSPLLCLK1 进行 4 分频 2h = SYSPLLCLK1 进行 6 分频 3h = SYSPLLCLK1 进行 8 分频 4h = SYSPLLCLK1 进行 10 分频 5h = SYSPLLCLK1 进行 12 分频 6h = SYSPLLCLK1 进行 14 分频 7h = SYSPLLCLK1 进行 16 分频 8h = SYSPLLCLK1 进行 18 分频 9h = SYSPLLCLK1 进行 20 分频 Ah = SYSPLLCLK1 进行 22 分频 Bh = SYSPLLCLK1 进行 24 分频 Ch = SYSPLLCLK1 进行 26 分频 Dh = SYSPLLCLK1 进行 28 分频 Eh = SYSPLLCLK1 进行 30 分频 Fh = SYSPLLCLK1 进行 32 分频
11-8	RDIVCLK0	R/W	0h	RDIVCLK0 为 SYSPLLCLK0 输出设置最终分频器。 0h = SYSPLLCLK0 进行 2 分频 1h = SYSPLLCLK0 进行 4 分频 2h = SYSPLLCLK0 进行 6 分频 3h = SYSPLLCLK0 进行 8 分频 4h = SYSPLLCLK0 进行 10 分频 5h = SYSPLLCLK0 进行 12 分频 6h = SYSPLLCLK0 进行 14 分频 7h = SYSPLLCLK0 进行 16 分频 8h = SYSPLLCLK0 进行 18 分频 9h = SYSPLLCLK0 进行 20 分频 Ah = SYSPLLCLK0 进行 22 分频 Bh = SYSPLLCLK0 进行 24 分频 Ch = SYSPLLCLK0 进行 26 分频 Dh = SYSPLLCLK0 进行 28 分频 Eh = SYSPLLCLK0 进行 30 分频 Fh = SYSPLLCLK0 进行 32 分频
7	RESERVED	R/W	X	
6	ENABLECLK2X	R/W	0h	ENABLECLK2X 可启用或禁用 SYSPLLCLK2X 输出。 0h = SYSPLLCLK2X 已禁用 1h = SYSPLLCLK2X 已启用
5	ENABLECLK1	R/W	0h	ENABLECLK1 可启用或禁用 SYSPLLCLK1 输出。 0h = SYSPLLCLK1 已禁用 1h = SYSPLLCLK1 已启用
4	ENABLECLK0	R/W	0h	ENABLECLK0 可启用或禁用 SYSPLLCLK0 输出。 0h = SYSPLLCLK0 已禁用 1h = SYSPLLCLK0 已启用
3-2	RESERVED	R/W	X	
1	MCLK2XVCO	R/W	0h	MCLK2XVCO 选择 SYSPLL 输出，此输出发送至 HSCLK 多路复用器以供 MCLK 使用。 0h = SYSPLLCLK0 输出发送到 HSCLK 多路复用器 1h = SYSPLLCLK2X 输出发送到 HSCLK 多路复用器
0	SYSPLLREF	R/W	0h	SYSPLLREF 选择系统 PLL (SYSPLL) 基准时钟源。 0h = SYSPLL 基准为 SYSOSC 1h = SYSPLL 基准为 HFCLK

2.6.1.18 SYSPLLCFG1 寄存器 (偏移 = 1124h) [复位 = X]

图 2-29 展示了 SYSPLLCFG1，表 2-39 中对其进行了介绍。

返回到汇总表。

SYSPLL 基准和反馈分频器

图 2-29. SYSPLLCFG1 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	QDIV							RESERVED						PDIV	
R/W-X	R/W-0h							R/W-X						R/W-0h	

表 2-39. SYSPLLCFG1 寄存器字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	X	
14-8	QDIV	R/W	0h	QDIV 选择 SYSPLL 反馈路径分频器。 0h = 一分频不是有效的 QDIV 选项 1h = 反馈路径进行 2 分频 7Eh =反馈路径进行 127 分频 (0x7E)
7-2	RESERVED	R/W	X	
1-0	PDIV	R/W	0h	PDIV 选择 SYSPLL 基准时钟预分频器。 0h = SYSPLLREF 未分频 1h = SYSPLLREF 进行 2 分频 2h = SYSPLLREF 进行 4 分频 3h = SYSPLLREF 进行 8 分频

2.6.1.19 SYSPLLPARAM0 寄存器（偏移 = 1128h）[复位 = X]

图 2-30 展示了 SYSPLLPARAM0，表 2-40 中对其进行了介绍。

返回到汇总表。

SYSPLL PARAM0（从 FACTORY 区域加载）

图 2-30. SYSPLLPARAM0 寄存器

31	30	29	28	27	26	25	24
CAPBOVERRIDE	RESERVED			CAPBVAL			
R/W-0h	R/W-X			R/W-0h			
23	22	21	20	19	18	17	16
RESERVED		CPCURRENT					
R/W-X		R/W-0h					
15	14	13	12	11	10	9	8
RESERVED		STARTTIMELP					
R/W-X		R/W-0h					
7	6	5	4	3	2	1	0
RESERVED		STARTTIME					
R/W-X		R/W-0h					

表 2-40. SYSPLLPARAM0 寄存器字段说明

位	字段	类型	复位	说明
31	CAPBOVERRIDE	R/W	0h	CAPBOVERRIDE 控制针对电容 B 的覆盖 0h = 电容 B 覆盖已禁用 1h = 电容 B 覆盖已启用
30-29	保留	R/W	X	
28-24	CAPBVAL	R/W	0h	电容 B 的覆盖值
23-22	保留	R/W	X	
21-16	CPCURRENT	R/W	0h	电荷泵电流
15-14	RESERVED	R/W	X	
13-8	STARTTIMELP	R/W	0h	从低功耗模式退出到锁定时钟的启动时间，分辨率为 1us
7-6	RESERVED	R/W	X	
5-0	STARTTIME	R/W	0h	从启用到锁定时钟的启动时间，分辨率为 1us

2.6.1.20 SYSPLLPARAM1 寄存器（偏移 = 112Ch）[复位 = X]

图 2-31 展示了 SYSPLLPARAM1，表 2-41 中对其进行了介绍。

返回到汇总表。

SYSPLL PARAM1（从 FACTORY 区域加载）

图 2-31. SYSPLLPARAM1 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPFRESC								RESERVED						LPFRESA	
R/W-0h								R/W-X						R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPFRESA								RESERVED			LPFCAPA				
R/W-0h								R/W-X			R/W-0h				

表 2-41. SYSPLLPARAM1 寄存器字段说明

位	字段	类型	复位	说明
31-24	LPFRESC	R/W	0h	环路滤波器分辨率 C
23-18	RESERVED	R/W	X	
17-8	LPFRESA	R/W	0h	环路滤波器分辨率 A
7-5	RESERVED	R/W	X	
4-0	LPFCAPA	R/W	0h	环路滤波器电容 A

2.6.1.21 GENCLKCFG 寄存器（偏移 = 1138h）[复位 = X]

图 2-32 展示了 GENCLKCFG，表 2-42 中对此进行了介绍。

返回到汇总表。

通用时钟配置

图 2-32. GENCLKCFG 寄存器

31	30	29	28	27	26	25	24
保留			FCCTRIGCNT				
R/W-X			R/W-0h				
23	22	21	20	19	18	17	16
ANACPUMPCFG		FCCLVLTRIG	FCCTRIGSRC	FCCSELCLK			
R/W-0h		R/W-0h	R/W-0h	R/W-0h			
15	14	13	12	11	10	9	8
HFCLK4MFPCLKDIV				RESERVED		MFPCLKSRC	CANCLKSRC
R/W-0h				R/W-X		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
EXCLKDIVEN	EXCLKDIVVAL			保留	EXCLKSRC		
R/W-0h	R/W-0h			R/W-X	R/W-0h		

表 2-42. GENCLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	X	
28-24	FCCTRIGCNT	R/W	0h	FCCTRIGCNT 指定触发窗口中的触发时钟周期数。可以指定 FCCTRIGCNT=0h（1 个触发时钟周期）至 1Fh（32 个触发时钟周期）。
23-22	ANACPUMPCFG	R/W	0h	ANACPUMPCFG 选择模拟多路复用器电荷泵 (VBOOST) 使能方法。 0h = 根据 COMP、GPAMP 或 OPA 的请求启用 VBOOST 1h = 当器件处于 RUN 或 SLEEP 模式或当启用 COMP/GPAMP/OPA 时启用 VBOOST 2h = 始终启用 VBOOST
21	FCCLVLTRIG	R/W	0h	FCCLVLTRIG 选择频率时钟计数器 (FCC) 触发模式。 0h = 上升沿到上升沿触发 1h = 电平触发
20	FCCTRIGSRC	R/W	0h	FCCTRIGSRC 选择频率时钟计数器 (FCC) 触发源。 0h = FCC 触发器为外部引脚 1h = FCC 触发器为 LFCLK
19-16	FCCSELCLK	R/W	0h	FCCSELCLK 选择频率时钟计数器 (FCC) 时钟源。 0h = FCC 时钟为 MCLK 1h = FCC 时钟为 SYSOSC 2h = FCC 时钟为 HFCLK 3h = FCC 时钟为 CLK_OUT 选择 4h = FCC 时钟为 SYSPLLCLK0 5h = FCC 时钟为 SYSPLLCLK1 6h = FCC 时钟为 SYSPLLCLK2X 7h = FCC 时钟为 FCCIN 外部输入

表 2-42. GENCLKCFG 寄存器字段说明 (continued)

位	字段	类型	复位	说明
15-12	HFCLK4MFPCLKDIV	R/W	0h	当 HFCLK 用作 MFPCLK 源时，HFCLK4MFPCLKDIV 选择应用于 HFCLK 的分频器。可以选择从 /1 到 /16 的整数分频器。 0h = HFCLK 在用于 MFPCLK 之前未分频 1h = HFCLK 在用于 MFPCLK 之前进行 2 分频 2h = HFCLK 在用于 MFPCLK 之前进行 3 分频 3h = HFCLK 在用于 MFPCLK 之前进行 4 分频 4h = HFCLK 在用于 MFPCLK 之前进行 5 分频 5h = HFCLK 在用于 MFPCLK 之前进行 6 分频 6h = HFCLK 在用于 MFPCLK 之前进行 7 分频 7h = HFCLK 在用于 MFPCLK 之前进行 8 分频 8h = HFCLK 在用于 MFPCLK 之前进行 9 分频 9h = HFCLK 在用于 MFPCLK 之前进行 10 分频 Ah = HFCLK 在用于 MFPCLK 之前进行 11 分频 Bh = HFCLK 在用于 MFPCLK 之前进行 12 分频 Ch = HFCLK 在用于 MFPCLK 之前进行 13 分频 Dh = HFCLK 在用于 MFPCLK 之前进行 14 分频 Eh = HFCLK 在用于 MFPCLK 之前进行 15 分频 Fh = HFCLK 在用于 MFPCLK 之前进行 16 分频
11-10	RESERVED	R/W	X	
9	MFPCLKSRC	R/W	0h	MFPCLKSRC 选择 MFPCLK (中频精密时钟) 源。 0h = MFPCLK 以 SYSOSC 为时钟源 1h = MFPCLK 以 HFCLK 为时钟源
8	CANCLKSRC	R/W	0h	CANCLKSRC 选择 CANCLK 源。 0h = CANCLK 源为 HFCLK 1h = CANCLK 源为 SYSPLLCLK1
7	EXCLKDIVEN	R/W	0h	EXCLKDIVEN 启用或禁用 CLK_OUT 外部时钟输出块的分频器功能。 0h = 时钟分频器已禁用 (直通，未应用 EXCLKDIVVAL) 1h = 时钟分频器已启用 (应用 EXCLKDIVVAL)
6-4	EXCLKDIVVAL	R/W	0h	EXCLKDIVVAL 为 CLK_OUT 外部时钟输出块中的分频器选择分频器值。 0h = CLK_OUT 源具有 2 分频 1h = CLK_OUT 源具有 4 分频 2h = CLK_OUT 源具有 6 分频 3h = CLK_OUT 源具有 8 分频 4h = CLK_OUT 源具有 10 分频 5h = CLK_OUT 源具有 12 分频 6h = CLK_OUT 源具有 14 分频 7h = CLK_OUT 源具有 16 分频
3	RESERVED	R/W	X	
2-0	EXCLKSRC	R/W	0h	EXCLKSRC 为 CLK_OUT 外部时钟输出块选择源。ULPCLK 和 MFPCLK 要求启用 CLK_OUT 分频器 (EXCLKDIVEN) 0h = CLK_OUT 为 SYSOSC 1h = CLK_OUT 为 ULPCLK (必须启用 EXCLKDIVEN) 2h = CLK_OUT 为 LFCLK 3h = CLK_OUT 为 MFPCLK (必须启用 EXCLKDIVEN) 4h = CLK_OUT 为 HFCLK 5h = CLK_OUT 为 SYSPLLCLK1 (SYSPLLCLK1 必须 ≤48MHz)

2.6.1.22 GENCLKEN 寄存器（偏移 = 113Ch）[复位 = X]

图 2-33 展示了 GENCLKEN，表 2-43 中对此进行了介绍。

返回到汇总表。

通用时钟使能控制

图 2-33. GENCLKEN 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
保留			MFPCLKEN	保留			EXCLKEN
R/W-X			R/W-0h	R/W-X			R/W-0h

表 2-43. GENCLKEN 寄存器字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	X	
4	MFPCLKEN	R/W	0h	MFPCLKEN 启用中频精密时钟 (MFPCLK)。 0h = MFPCLK 已禁用 1h = MFPCLK 已启用
3-1	保留	R/W	X	
0	EXCLKEN	R/W	0h	EXCLKEN 启用 CLK_OUT 外部时钟输出块。 0h = CLK_OUT 块已禁用 1h = CLK_OUT 块已启用

2.6.1.23 PMODECFG 寄存器（偏移 = 1140h）[复位 = X]

图 2-34 展示了 PMODECFG，表 2-44 中对此进行了介绍。

返回到汇总表。

电源模式配置

图 2-34. PMODECFG 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														DSLEEP	
R/W-X														R/W-0h	

表 2-44. PMODECFG 寄存器字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	X	
1-0	DSLEEP	R/W	0h	DSLEEP 根据 CPU 发出的 DEEPSLEEP 请求选择要进入的工作模式。 0h = 进入 STOP 模式 1h = 进入待机模式 2h = 进入关断模式 3h = 保留

2.6.1.24 FCC 寄存器（偏移 = 1150h）[复位 = X]

图 2-35 展示了 FCC，表 2-45 中对此进行了介绍。

返回到汇总表。

频率时钟计数器 (FCC) 计数

图 2-35. FCC 寄存器

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

表 2-45. FCC 寄存器字段说明

位	字段	类型	复位	说明
31-22	保留	R	X	
21-0	数据	R	0h	频率时钟计数器 (FCC) 计数值。

2.6.1.25 SYSOSCTRIMUSER 寄存器（偏移 = 1170h）[复位 = X]

图 2-36 展示了 SYSOSCTRIMUSER，表 2-46 中对此进行了介绍。

返回到汇总表。

SYSOSC 用户指定的修整

图 2-36. SYSOSCTRIMUSER 寄存器

31	30	29	28	27	26	25	24
保留				RDIV			
R/W-X				R/W-0h			
23	22	21	20	19	18	17	16
RDIV				RESFINE			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
保留		RESCOARSE					
R/W-X		R/W-0h					
7	6	5	4	3	2	1	0
保留	CAP			保留		FREQ	
R/W-X	R/W-0h			R/W-X		R/W-0h	

表 2-46. SYSOSCTRIMUSER 寄存器字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	X	
28-20	RDIV	R/W	0h	RDIV 指定频率校正环路 (FCL) 电阻器修整。该值随目标频率而变化。
19-16	RESFINE	R/W	0h	RESFINE 指定电阻器微调。该值随目标频率而变化。
15-14	RESERVED	R/W	X	
13-8	RESCOARSE	R/W	0h	RESCOARSE 指定电阻器粗调。该值随目标频率而变化。
7	RESERVED	R/W	X	
6-4	CAP	R/W	0h	CAP 指定 SYSOSC 电容器修整。该值随目标频率而变化。
3-2	RESERVED	R/W	X	
1-0	FREQ	R/W	0h	FREQ 指定 SYSOSC 的目标用户修整频率。 0h = 保留 1h = 16MHz 用户频率 2h = 24MHz 用户频率 3h = 保留

2.6.1.26 SRAMBOUNDARY 寄存器（偏移 = 1178h）[复位 = X]

图 2-37 展示了 SRAMBOUNDARY，表 2-47 对其进行了介绍。

返回到汇总表。

SRAM 写边界

图 2-37. SRAMBOUNDARY 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED												ADDR															保留				
R/W-X												R/W-0h															R/W-X				

表 2-47. SRAMBOUNDARY 寄存器字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	X	
19-5	ADDR	R/W	0h	SRAM 边界配置。该字段中配置的值具有以下作用：对小于或等于该值的地址的 SRAM 访问将仅为 RW。对大于该值的地址的 SRAM 访问将仅为 RX。值 0 无效（系统将没有堆栈）。如果设置为 0，系统的行为就像整个 SRAM 为 RWX 一样。可以配置任何非零值，包括值 = SRAM 大小。
4-0	RESERVED	R/W	X	

2.6.1.27 SYSTEMCFG 寄存器（偏移 = 1180h）[复位 = X]

图 2-38 展示了 SYSTEMCFG，表 2-48 中对此进行了介绍。

返回到汇总表。

系统配置

图 2-38. SYSTEMCFG 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
保留					FLASHECCRSTDIS	WWDTLP1RSTDIS	WWDTLP0RSTDIS
R/W-X					R/W-1h	R/W-1h	R/W-0h

表 2-48. SYSTEMCFG 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	密钥值 1Bh (27) 必须与要更新的内容一起写入 KEY。读取为 0 1Bh = 发出写入
23-3	保留	R/W	X	
2	FLASHECCRSTDIS	R/W	1h	FLASHECCRSTDIS 指定闪存 ECC 双错检测 (DED) 是将触发 SYSRST 或还是 NMI。 0h = 闪存 ECC DED 将触发 SYSRST 1h = 闪存 ECC DED 将触发 NMI
1	WWDTLP1RSTDIS	R/W	1h	WWDTLP1RSTDIS 指定 WWDT 错误事件是将触发 SYSRST 还是 NMI。 0h = WWDTLP1 错误事件将触发 SYSRST 1h = WWDTLP1 错误事件将触发 NMI
0	WWDTLP0RSTDIS	R/W	0h	WWDTLP0RSTDIS 指定 WWDT 错误事件是将触发 BOOTRST 还是 NMI。 0h = WWDTLP0 错误事件将触发 BOOTRST 1h = WWDTLP0 错误事件将触发 NMI

2.6.1.28 WRITELOCK 寄存器（偏移 = 1200h）[复位 = X]

图 2-39 展示了 WRITELOCK，表 2-49 中对此进行了介绍。

返回到汇总表。

SYSCTL 寄存器写锁定

图 2-39. WRITELOCK 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
保留							ACTIVE
R/W-X							R/W-0h

表 2-49. WRITELOCK 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	X	
0	有效	R/W	0h	ACTIVE 控制关键 SYSCTL 寄存器是否受写保护。 0h = 允许写入可锁定寄存器 1h = 不允许写入可锁定寄存器

2.6.1.29 CLKSTATUS 寄存器（偏移 = 1204h）[复位 = X]

图 2-40 展示了 CLKSTATUS，表 2-50 中对此进行了介绍。

返回到汇总表。

时钟模块 (CKM) 状态

图 2-40. CLKSTATUS 寄存器

31	30	29	28	27	26	25	24
ANACLKERR	OPAMPCLKERR	SYSPLLBLKUPD	HFCLKBLKUPD	保留		FCCDONE	FCLMODE
R-0h	R-0h	R-0h	R-0h	R-X		R-0h	R-0h
23	22	21	20	19	18	17	16
LFCLKFAIL	RESERVED	HSCLKGOOD	HSCLKDEAD	保留		CURMCLKSEL	CURHSCLKSEL
R-0h	R-X	R-0h	R-0h	R-X		R-0h	R-0h
15	14	13	12	11	10	9	8
RESERVED	SYSPLLOFF	HFCLKOFF	HSCLKSOFF	LFOSCGOOD	LFXTGOOD	SYSPLLGOOD	HFCLKGOOD
R-X	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
LFCLKMUX		保留	HSCLKMUX	保留		SYSOSCFREQ	
R-0h		R-X	R-0h	R-X		R-0h	

表 2-50. CLKSTATUS 寄存器字段说明

位	字段	类型	复位	说明
31	ANACLKERR	R	0h	当器件时钟配置不支持已启用的模拟外设模式且模拟外设可能无法按预期正常工作时，将设置 ANACLKERR。 0h = 未检测到模拟时钟错误 1h = 检测到模拟时钟错误
30	OPAMPCLKERR	R	0h	当器件时钟配置不支持已启用的 OPA 模式且 OPA 可能无法按预期正常工作时，将设置 OPAMPCLKERR。 0h = 未检测到 OPA 时钟生成错误 1h = 检测到 OPA 时钟生成错误
29	SYSPLLBLKUPD	R	0h	SYSPLLBLKUPD 指示是否阻止对 SYSPLLCFG0/1 和 SYSPLLPARAM0/1 的写入。 0h = 允许写入 SYSPLLCFG0/1 和 SYSPLLPARAM0/1 1h = 阻止写入 SYSPLLCFG0/1 和 SYSPLLPARAM0/1
28	HFCLKBLKUPD	R	0h	HFCLKBLKUPD 指示是否阻止对 HFCLKCLKCFG 寄存器的写入。 0h = 允许写入 HFCLKCLKCFG 1h = 阻止写入 HFCLKCLKCFG
27-26	RESERVED	R	X	
25	FCCDONE	R	0h	FCCDONE 指示频率时钟计数器捕捉是否完成。 0h = FCC 捕捉未完成 1h = FCC 捕捉已完成
24	FCLMODE	R	0h	FCLMODE 指示是否启用了 SYSOSC 频率校正环路 (FCL)。 0h = SYSOSC FCL 已禁用 1h = SYSOSC FCL 已启用
23	LFCLKFAIL	R	0h	LFCLKFAIL 指示连续 LFCLK 监控器是否检测到 LFXT 或 LFCLK_IN 时钟卡滞故障。 0h = 未检测到 LFCLK 故障 1h = 检测到 LFCLK 卡滞故障
22	RESERVED	R	X	

表 2-50. CLKSTATUS 寄存器字段说明 (continued)

位	字段	类型	复位	说明
21	HSCLKGOOD	R	0h	如果为 HSCLK 选择的时钟源成功启动，则 HSCLKGOOD 由硬件设置。 0h = HSCLK 源未正确启动 1h = HSCLK 源正确启动
20	HSCLKDEAD	R	0h	如果为 HSCLK 选择的源已启动但未成功启动，则 HSCLKDEAD 由硬件设置。 0h = HSCLK 源未启动或未正确启动 1h = HSCLK 源未正确启动
19-18	RESERVED	R	X	
17	CURMCLKSEL	R	0h	CURMCLKSEL 指示 MCLK 当前是否以 LFCLK 为时钟源。 0h = MCLK 不以 LFCLK 为时钟源 1h = MCLK 以 LFCLK 为时钟源
16	CURHSCLKSEL	R	0h	CURHSCLKSEL 指示 HSCLK 的当前时钟源。 0h = HSCLK 当前以 SYSPLL 为时钟源 1h = HSCLK 当前以 HFCLK 为时钟源
15	保留	R	X	
14	SYSPLLOFF	R	0h	SYSPLLOFF 指示 SYSPLL 是否已禁用或在启动时发生故障。当 SYSPLL 启动时，SYSPLLOFF 由硬件清零。在 SYSPLL 启动之后，如果 SYSPLL 启动监控器确定 SYSPLL 未正确启动，则设置 SYSPLLOFF。 0h = SYSPLL 已正确启动并已启用 1h = SYSPLL 已禁用或在启动时发生故障
13	HFCLKOFF	R	0h	HFCLKOFF 指示 HFCLK 是否已禁用或在启动时发生故障。当 HFCLK 启动时，HFCLKOFF 由硬件清零。HFCLK 启动后，如果 HFCLK 启动监控器确定 HFCLK 未正确启动，则设置 HFCLKOFF。 0h = HFCLK 已正确启动并已启用 1h = HFCLK 已禁用或在启动时发生故障
12	HSCLKSOFF	R	0h	当高速时钟源（SYSPLL、HFCLK）被禁用或发生故障时，设置 HSCLKSOFF。它是 HFCLKOFF 和 SYSPLLOFF 的逻辑与。 0h = SYSPLL、HFCLK 或两者已正确启动并保持启用 1h = SYSPLL 和 HFCLK 均关闭或发生故障
11	LFOSCGOOD	R	0h	LFOSCGOOD 指示 LFOSC 启动是否已完成以及 LFOSC 是否准备就绪可供使用。 0h = LFOSC 未就绪 1h = LFOSC 已就绪
10	LFXTGOOD	R	0h	LFXTGOOD 指示 LFXT 是否正确启动。LFXT 启动时，LFXTGOOD 由硬件清零。启动稳定时间到期后，将测试 LFXT 状态。如果 LFXT 成功启动，则设置 LFXTGOOD 位，否则它保持清零。 0h = LFXT 未正确启动 1h = LFXT 已正确启动
9	SYSPLLGOOD	R	0h	SYSPLLGOOD 指示 SYSPLL 是否正确启动。当 SYSPLL 启动时，SYSPLLGOOD 由硬件清零。启动稳定时间到期后，将测试 SYSPLL 状态。如果 SYSPLL 成功启动，则设置 SYSPLLGOOD 位，否则它保持清零。 0h = SYSPLL 未正确启动 1h = SYSPLL 已正确启动
8	HFCLKGOOD	R	0h	HFCLKGOOD 指示 HFCLK 已正确启动。当 HFXT 启动或选择 HFCLK_IN 作为 HFCLK 源时，如果检测到有效的 HFCLK，该位将由硬件设置，如果 HFCLK 未在预期范围内运行，则该位将被清零。 0h = HFCLK 未正确启动 1h = HFCLK 已正确启动

表 2-50. CLKSTATUS 寄存器字段说明 (continued)

位	字段	类型	复位	说明
7-6	LFCLKMUX	R	0h	LFCLKMUX 指示 LFCLK 是以内部 LFOSC、低频晶振 (LFXT) 还是 LFCLK_IN 数字时钟输入为时钟源。 0h = LFCLK 以内部 LFOSC 为时钟源 1h = LFCLK 以 LFXT (晶振) 为时钟源 2h = LFCLK 以 LFCLK_IN (外部数字时钟输入) 为时钟源
5	RESERVED	R	X	
4	HSCLKMUX	R	0h	HSCLKMUX 表示 MCLK 当前是否以高速时钟 (HSCLK) 为时钟源。 0h = MCLK 不以 HSCLK 为时钟源 1h = MCLK 以 HSCLK 为时钟源
3-2	RESERVED	R	X	
1-0	SYSOSCFREQ	R	0h	SYSOSCFREQ 指示当前的 SYSOSC 工作频率。 0h = SYSOSC 处于基频 (32MHz) 1h = SYSOSC 处于低频 (4MHz) 2h = SYSOSC 处于用户修整的频率 (16MHz 或 24MHz) 3h = 保留

2.6.1.30 SYSSTATUS 寄存器（偏移 = 1208h）[复位 = X]

图 2-41 展示了 SYSSTATUS，表 2-51 中对此进行了介绍。

返回到汇总表。

系统状态信息

图 2-41. SYSSTATUS 寄存器

31	30	29	28	27	26	25	24
REBOOTATTEMPTS		保留					
R-0h		R-X					
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留	SHDNIOLOCK	SWDCFGDIS	EXTRSTPINDIS	保留			MCAN0READY
R-X	R-0h	R-0h	R-0h	R-X			R-0h
7	6	5	4	3	2	1	0
保留	PMUIREFGOOD	ANACPUMPGOOD	BORLVL	BORCURTHRESHOLD		FLASHSEC	FLASHDED
R-X	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 2-51. SYSSTATUS 寄存器字段说明

位	字段	类型	复位	说明
31-30	REBOOTATTEMPTS	R	0h	REBOOTATTEMPTS 指示在用户应用启动之前进行的引导尝试次数。
29-15	保留	R	X	
14	SHDNIOLOCK	R	0h	SHDNIOLOCK 指示 IO 是否因关断而锁定 0h = IO 不因关断而锁定 1h = IO 因关断而锁定
13	SWDCFGDIS	R	0h	SWDCFGDIS 指示用户是否禁用了 SWD 端口 0h = SWD 端口已启用 1h = SWD 端口已禁用
12	EXTRSTPINDIS	R	0h	EXTRSTPINDIS 指示用户是否禁用了外部复位引脚 0h = 外部复位引脚已启用 1h = 外部复位引脚已禁用
11-9	保留	R	X	
8	MCAN0READY	R	0h	MCAN0READY 指示 MCAN0 外设是否准备就绪。 0h = MCAN0 未就绪 1h = MCAN0 已就绪
7	RESERVED	R	X	
6	PMUIREFGOOD	R	0h	当 PMU 电流基准就绪时，PMUIREFGOOD 由硬件设置。 0h = IREF 未就绪 1h = IREF 已就绪
5	ANACPUMPGOOD	R	0h	当 VBOOST 模拟多路复用器电荷泵就绪时，ANACPUMPGOOD 由硬件设置。 0h = VBOOST 未就绪 1h = VBOOST 已就绪
4	BORLVL	R	0h	BORLVL 指示是否发生 BOR 事件并且 BOR 阈值是否已由硬件切换为 BOR0。 0h = 未发生 BOR 违例 1h = 发生 BOR 违例且 BOR 阈值切换为 BOR0

表 2-51. SYSSTATUS 寄存器字段说明 (continued)

位	字段	类型	复位	说明
3-2	BORCURTHRESHOLD	R	0h	BORCURTHRESHOLD 指示有源欠压复位电源监测器配置。 0h = 默认最小阈值；BOR0- 违例触发 BOR 1h = BOR1- 违例生成 BORLVL 中断 2h = BOR2- 违例生成 BORLVL 中断 3h = BOR3- 违例生成 BORLVL 中断
1	FLASHSEC	R	0h	FLASHSEC 指示是否检测到并校正了闪存 ECC single-bit 错误 (SEC)。 0h = 未检测到闪存 ECC single-bit 错误 1h = 检测到并校正了闪存 ECC single-bit 错误
0	FLASHDED	R	0h	FLASHDED 指示是否检测到闪存 ECC 双位错误 (DED)。 0h = 未检测到闪存 ECC 双位错误 1h = 检测到闪存 ECC 双位错误

2.6.1.31 DEDERRADDR 寄存器（偏移 = 120Ch）[复位 = 00000000h]

图 2-42 展示了 DEDERRADDR，表 2-52 对其进行了介绍。

返回到汇总表。

存储器 DED 地址

图 2-42. DEDERRADDR 寄存器

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

表 2-52. DEDERRADDR 寄存器字段说明

位	字段	类型	复位	说明
31-0	ADDR	R	0h	MEMORY DED 错误的地址。

2.6.1.32 RSTCAUSE 寄存器（偏移 = 1220h）[复位 = X]

图 2-43 展示了 RSTCAUSE，表 2-53 中对此进行了介绍。

返回到汇总表。

复位原因

图 2-43. RSTCAUSE 寄存器

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

表 2-53. RSTCAUSE 寄存器字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	X	
4-0	ID	RC	0h	ID 是一个“读取以清除”字段，指示自上次读取以来的最低级别复位原因。 0h = 自上次读取后无复位 1h = POR- 违例，SHUTDNSTOREx 或 PMU 修整奇偶校验故障 2h = NRST 触发的 POR（保持时间 >1s） 3h = 软件触发的 POR 4h = BOR0- 违例 5h = 关断模式退出 8h = 非 PMU 修整奇偶校验故障 9h = 致命时钟故障 Ah = 软件触发的 BOOTRST Ch = NRST 触发的 BOOTRST（保持时间 <1s） 10h = BSL 退出 11h = BSL 进入 12h = WWDT0 违例 13h = WWDT1 违例 14h = 闪存不可纠正的 ECC 错误 15h = CPULOCK 违例 1Ah = 调试触发的 SYSRST 1Bh = 软件触发的 SYSRST 1Ch = 调试触发的 CPURST 1Dh = 软件触发的 CPURST

2.6.1.33 RESETLEVEL 寄存器 (偏移 = 1300h) [复位 = X]

图 2-44 展示了 RESETLEVEL，表 2-54 中对此进行了介绍。

返回到汇总表。

应用触发的复位命令的复位电平

图 2-44. RESETLEVEL 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														等级	
R/W-X														R/W-0h	

表 2-54. RESETLEVEL 寄存器字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	X	
2-0	等级	R/W	0h	当 RESETCMD 设置为生成软件触发的复位时，LEVEL 用于指定要发出的复位类型。 0h = 发出 SYSRST (只适用于 CPU 和外设) 1h = 发出 BOOTRST (CPU、外设和引导配置例程) 2h = 发出 SYSRST 并进入引导加载程序 (BSL) 3h = 发出上电复位 (POR) 4h = 发出 SYSRST 并退出引导加载程序 (BSL)

2.6.1.34 RESETCMD 寄存器（偏移 = 1304h）[复位 = X]

图 2-45 展示了 RESETCMD，表 2-55 中对此进行了介绍。

返回到汇总表。

执行应用触发的复位命令

图 2-45. RESETCMD 寄存器

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

表 2-55. RESETCMD 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 E4h (228) 与 GO 一起写入 KEY 才能触发复位。 E4h = 发出复位
23-1	保留	W	X	
0	搜索	W	0h	执行 RESETLEVEL.LEVEL 中指定的复位。必须与 KEY 一起写入。 1h = 发出复位

2.6.1.35 BORTHRESHOLD 寄存器（偏移 = 1308h）[复位 = X]

图 2-46 展示了 BORTHRESHOLD，表 2-56 中对此进行了介绍。

返回到汇总表。

BOR 阈值选择

图 2-46. BORTHRESHOLD 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
R/W-X															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保留														等级	
R/W-X														R/W-0h	

表 2-56. BORTHRESHOLD 寄存器字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	X	
1-0	等级	R/W	0h	LEVEL 指定所需的 BOR 阈值和 BOR 模式。 0h = 默认最小阈值；BOR0- 违例触发 BOR 1h = BOR1- 违例生成 BORLVL 中断 2h = BOR2- 违例生成 BORLVL 中断 3h = BOR3- 违例生成 BORLVL 中断

2.6.1.36 BORCLRCMD 寄存器（偏移 = 130Ch）[复位 = X]

图 2-47 展示了 BORCLRCMD，表 2-57 中对此进行了介绍。

返回到汇总表。

设置 BOR 阈值

图 2-47. BORCLRCMD 寄存器

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

表 2-57. BORCLRCMD 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 C7h (199) 与 GO 一起写入 KEY 才能触发清除和 BOR 阈值变化。 C7h = 发出清除
23-1	保留	W	X	
0	搜索	W	0h	GO 清除任何先前的 BOR 违例状态指示，并尝试将有效 BOR 模式更改为 BORTHRESHOLD 寄存器的 LEVEL 字段中指定的模式。 1h = 发出清除

2.6.1.37 SYSOSCFCLCTL 寄存器（偏移 = 1310h）[复位 = X]

图 2-48 展示了 SYSOSCFCLCTL，表 2-58 中对此进行了介绍。

返回到汇总表。

SYSOSC 频率校正环路 (FCL) ROSC 使能

图 2-48. SYSOSCFCLCTL 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
RESERVED						SETUSEEXRES	SETUSEFCL
W-X						W-0h	W-0h

表 2-58. SYSOSCFCLCTL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 2Ah (42) 与 SETUSEFCL 一起写入 KEY 才能启用 FCL。 2Ah = 发出命令
23-2	RESERVED	W	X	
1	SETUSEEXRES	W	0h	设置 SETUSEEXRES 以指定外部电阻将用于 FCL。必须在 ROSC 引脚上安装适当的电阻器。该状态将锁定，直到下一个 BOOTRST。 1h = 启用 SYSOSC 外部电阻
0	SETUSEFCL	W	0h	设置 SETUSEFCL 以在 SYSOSC 中启用频率校正环路。一旦启用，该状态将锁定，直到下一个 BOOTRST。 1h = 启用 SYSOSC FCL

2.6.1.38 LFXTCTL 寄存器（偏移 = 1314h）[复位 = X]

图 2-49 展示了 LFXTCTL，表 2-59 中对其进行了介绍。

返回到汇总表。

LFXT 和 LFCLK 控制

图 2-49. LFXTCTL 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
保留							
W-X							
15	14	13	12	11	10	9	8
保留							
W-X							
7	6	5	4	3	2	1	0
RESERVED						SETUSELFXT	STARTLFXT
W-X						W-0h	W-0h

表 2-59. LFXTCTL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将密钥值 91h (145) 与 STARTLFXT 或 SETUSELFXT 一起写入 KEY，以设置相应的位。 91h = 发出命令
23-2	RESERVED	W	X	
1	SETUSELFXT	W	0h	设置 SETUSELFXT 以将 LFCLK 切换为 LFXT。一旦设置，SETUSELFXT 将保持设置状态，直到下一个 BOOTRST。 0h = 0 1h = 使用 LFXT 作为 LFCLK 源
0	STARTLFXT	W	0h	设置 STARTLFXT 以启动低频晶体振荡器 (LFXT)。一旦设置，STARTLFXT 将保持设置状态，直到下一个 BOOTRST。 0h = LFXT 未启动 1h = 启动 LFXT

2.6.1.39 EXLFCTL 寄存器（偏移 = 1318h）[复位 = X]

图 2-50 展示了 EXLFCTL，表 2-60 对其进行了介绍。

返回到汇总表。

LFCLK_IN 和 LFCLK 控制

图 2-50. EXLFCTL 寄存器

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

表 2-60. EXLFCTL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	密钥值 36h (54) 必须和 SETUSEEXLF 一起写入 KEY，才能设置 SETUSEEXLF。 36h = 发出命令
23-1	保留	W	X	
0	SETUSEEXLF	W	0h	设置 SETUSEEXLF 以将 LFCLK 切换到 LFCLK_IN 数字时钟输入。一旦设置，SETUSEEXLF 将保持设置状态，直到下一个 BOOTRST。 1h = 使用 LFCLK_IN 作为 LFCLK 源

2.6.1.40 SHDNIOREL 寄存器（偏移 = 131Ch）[复位 = X]

图 2-51 展示了 SHDNIOREL，表 2-61 中对此进行了介绍。

返回到汇总表。

SHUTDOWN IO 释放控制

图 2-51. SHDNIOREL 寄存器

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

表 2-61. SHDNIOREL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 91h 与 RELEASE 一起写入 KEY 才能设置 RELEASE。 91h = 发出命令
23-1	保留	W	X	
0	RELEASE	W	0h	将 RELEASE 设置为在关断模式退出后释放 IO。 1h = 释放 IO

2.6.1.41 EXRSTPIN 寄存器（偏移 = 1320h）[复位 = X]

图 2-52 展示了 EXRSTPIN，表 2-62 中对此进行了介绍。

返回到汇总表。

禁用 NRST 引脚的复位功能

图 2-52. EXRSTPIN 寄存器

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

表 2-62. EXRSTPIN 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 1Eh 与 DISABLE 一起写入才能禁用复位功能。 1Eh = 发出命令
23-1	保留	W	X	
0	DISABLE	W	0h	设置 DISABLE 以禁用 NRST 引脚的复位功能。一旦设置，该配置将锁定，直到下一个 POR。 0h = NRST 引脚的复位功能已启用 1h = NRST 引脚的复位功能已禁用

2.6.1.42 SYSSTATUSCLR 寄存器（偏移 = 1324h）[复位 = X]

图 2-53 展示了 SYSSTATUSCLR，表 2-63 中对此进行了介绍。

返回到汇总表。

清除 SYSSTATUS 的粘滞位

图 2-53. SYSSTATUSCLR 寄存器

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

表 2-63. SYSSTATUSCLR 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 CEh (206) 与 ALLECC 一起写入 KEY 才能清除 ECC 状态。 CEh = 发出命令
23-1	保留	W	X	
0	ALLECC	W	0h	设置 ALLECC 以清除所有与 ECC 相关的 SYSSTATUS 指示器。 1h = 清除 ECC 错误状态

2.6.1.43 SWDCFG 寄存器（偏移 = 1328h）[复位 = X]

图 2-54 展示了 SWDCFG，表 2-64 中对此进行了介绍。

返回到汇总表。

禁用 SWD 引脚上的 SWD 功能

图 2-54. SWDCFG 寄存器

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

表 2-64. SWDCFG 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须将键值 62h (98) 与 DISBALE 一起写入 KEY 才能禁用 SWD 功能。 62h = 发出命令
23-1	保留	W	X	
0	DISABLE	W	0h	设置 DISABLE 以禁用 SWD 引脚上的 SWD 功能，从而允许将 SWD 引脚用作 GPIO。 1h = 禁用 SWD 引脚上的 SWD 功能

2.6.1.44 FCCCMD 寄存器（偏移 = 132Ch）[复位 = X]

图 2-55 展示了 FCCCMD，表 2-65 中对此进行了介绍。

返回到汇总表。

频率时钟计数器开始捕捉

图 2-55. FCCCMD 寄存器

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

表 2-65. FCCCMD 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	必须写入键值 0Eh (14) 和 GO 才能开始捕捉。 0Eh = 发出命令
23-1	保留	W	X	
0	搜索	W	0h	设置 GO 以使用频率时钟计数器 (FCC) 开始捕捉。 1h = 1

2.6.1.45 PMUOPAMP 寄存器（偏移 = 1380h）[复位 = X]

图 2-56 展示了 PMUOPAMP，表 2-66 中对此进行了介绍。

返回到汇总表。

GPAMP 控制

图 2-56. PMUOPAMP 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留				CHOPCLKMODE		CHOPCLKFREQ	
R/W-X				R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
保留	OUTENABLE	RRI		NSEL		PCHENABLE	ENABLE
R/W-X	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h

表 2-66. PMUOPAMP 寄存器字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	X	
11-10	CHOPCLKMODE	R/W	0h	CHOPCLKMODE 选择 GPAMP 斩波模式。 0h = 禁用斩波 1h = 常规斩波 2h = ADC 辅助斩波 3h = 保留
9-8	CHOPCLKFREQ	R/W	0h	CHOPCLKFREQ 选择 GPAMP 斩波时钟频率 0h = 16kHz 1h = 8kHz 2h = 4kHz 3h = 2kHz
7	RESERVED	R/W	X	
6	OUTENABLE	R/W	0h	设置 OUTENABLE 以将 GPAMP 输出信号连接到 GPAMP_OUT 引脚 0h = GPAMP_OUT 信号未连接到 GPAMP_OUT 引脚 1h = GPAMP_OUT 信号连接到 GPAMP_OUT 引脚
5-4	RRI	R/W	0h	RRI 选择轨至轨输入模式。 0h = PMOS 输入对 1h = NMOS 输入对 2h = 轨到轨模式 3h = 轨到轨模式
3-2	NSEL	R/W	0h	NSEL 选择 GPAMP 负通道输入。 0h = GPAMP_OUT 引脚连接到负通道 1h = GPAMP_IN- 引脚连接到负通道 2h = GPAMP_OUT 信号连接到负通道 3h = 未选择通道
1	PCHENABLE	R/W	0h	设置 PCHENABLE 以启用正通道输入。 0h = 正通道已禁用 1h = GPAMP_IN+ 连接到正通道

表 2-66. PMUOPAMP 寄存器字段说明 (continued)

位	字段	类型	复位	说明
0	ENABLE	R/W	0h	设置 ENABLE 以打开 GPAMP。 0h = GPAMP 已禁用 1h = GPAMP 已启用

2.6.1.46 SHUTDNSTORE0 寄存器（偏移 = 1400h）[复位 = X]

图 2-57 展示了 SHUTDNSTORE0，表 2-67 中对此进行了介绍。

返回到汇总表。

关断存储内存（字节 0）

图 2-57. SHUTDNSTORE0 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
数据							
R/W-0h							

表 2-67. SHUTDNSTORE0 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	X	
9	PARITYERR	R	0h	SHUTDNSTORE0 的奇偶校验错误
8	奇偶校验	R/W	0h	SHUTDNSTORE0 的奇偶校验
7-0	数据	R/W	0h	关断存储字节 0

2.6.1.47 SHUTDNSTORE1 寄存器（偏移 = 1404h）[复位 = X]

图 2-58 展示了 SHUTDNSTORE1，表 2-68 中对此进行了介绍。

返回到汇总表。

关断存储内存（字节 1）

图 2-58. SHUTDNSTORE1 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
数据							
R/W-0h							

表 2-68. SHUTDNSTORE1 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	X	
7-0	数据	R/W	0h	关断存储字节 1

2.6.1.48 SHUTDNSTORE2 寄存器（偏移 = 1408h）[复位 = X]

图 2-59 展示了 SHUTDNSTORE2，表 2-69 中对此进行了介绍。

返回到汇总表。

关断存储内存（字节 2）

图 2-59. SHUTDNSTORE2 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
数据							
R/W-0h							

表 2-69. SHUTDNSTORE2 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	X	
7-0	数据	R/W	0h	关断存储字节 2

2.6.1.49 SHUTDNSTORE3 寄存器（偏移 = 140Ch）[复位 = X]

图 2-60 展示了 SHUTDNSTORE3，表 2-70 中对此进行了介绍。

返回到汇总表。

关断存储内存（字节 3）

图 2-60. SHUTDNSTORE3 寄存器

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
数据							
R/W-0h							

表 2-70. SHUTDNSTORE3 寄存器字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	X	
7-0	数据	R/W	0h	关断存储字节 3

This page intentionally left blank.

章节 3
CPU

CPU 子系统 (MCPUSS) 包括 Arm Cortex-M0+ 处理器、中断逻辑以及预取和高速缓存逻辑。

3.1 概述

MSP CPU 子系统 (MCPUSS) 包含中央处理单元 (CPU) 本身及其相关的支持逻辑和只读存储器 (ROM)。构成 MCPUSS 的功能块包括：

- Arm Cortex-M0+ 32 位 CPU 及其内部外设
- CPU 总线分离器/路由器
- 中断管理逻辑和 DEEPSLEEP 进入/退出逻辑
- 非易失性存储器系统预取和缓存逻辑
- 用于访问调试子系统 (DEBUGSS) 的 CPU 调试接口
- 处理器跟踪存储器
- 只读存储器 (ROM) (用于 BCR 和 BSL)

MCPUSS 的顶层架构如图 3-1 所示。

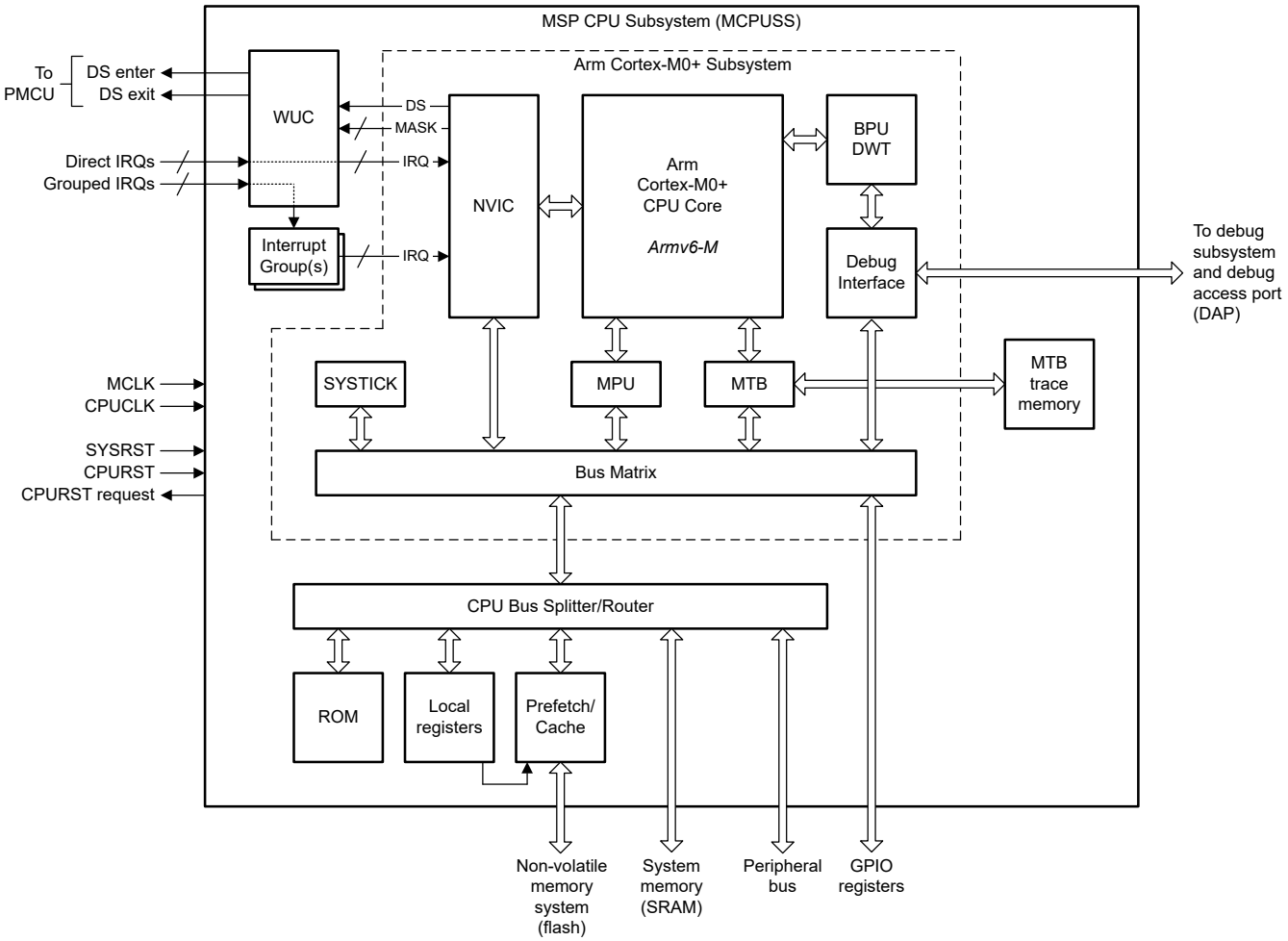


图 3-1. MSPM0Gxx MCPUSS 顶层图

3.2 Arm Cortex-M0+ CPU

MCPUSS 包含一个高能效 Arm Cortex-M0+ CPU，可实现 Armv6-M 指令集架构 (ISA)，并支持 32kHz 至 80MHz 的 CPU 时钟速度。Cortex-M0+ 是冯·诺依曼式 32 位处理器，具有 2 级超低功耗流水线和一个到 GPIO 寄存器的单周期访问端口，可实现高效的 GPIO 操作。

MSPM0Gxx 器件上的 Cortex-M0+ 实现具有以下特性：

- 高达 80MHz 的执行频率

- 小端字节序（最低字节地址位置的最低有效字节）
- 支持 32 位字指令提取
- 单周期 32*32 乘法指令
- 紧密集成的存储器保护单元 (MPU)
- 用户和特权执行模式
- 集成 24 位系统计时器 (SYSTICK)
- 用于调试的四个硬件断点和两个硬件观察点
- 微跟踪缓冲器
- 重置所有寄存器支持
- 矢量表偏移支持

Cortex-M0+ 架构可实现出色的代码密度、确定性中断处理以及与 Arm Cortex-M 系列中的其他处理器架构的向上兼容性。

本节对 Arm Cortex-M0+ 进行了一般性概述，以便对处理器的特性有基本的了解。有关使用 Arm Cortex-M0+ 处理器进行开发的详细信息，请参阅“Arm Cortex-M0+ 器件通用用户指南”。

3.2.1 CPU 寄存器文件

Arm Cortex-M0+ 处理器指令在 CPU 寄存器文件中的寄存器上运行。处理器包含一个由 16 个标准寄存器和 3 个专用寄存器组成的寄存器文件，如图 3-2 所示。

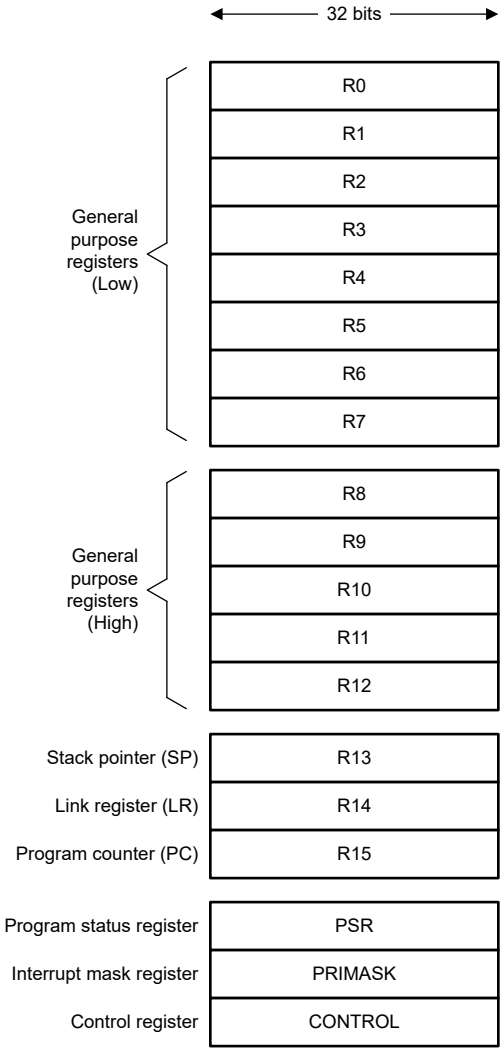


图 3-2. CPU 寄存器

通用寄存器 (R0-R12)

该处理器提供 13 个通用寄存器 R0-R12，用于对数据进行操作。寄存器 R0 至 R7（低位寄存器）可通过指定通用寄存器的所有指令访问。寄存器 R8 至 R12（高位寄存器）不能通过 16 位指令访问，但可以通过指定通用寄存器的任何 32 位指令访问。

栈指针寄存器 (R13)

栈指针位于 R13 中，包含主栈指针 (MSP) 或进程栈指针 (PSP)。当处理器以处理程序模式运行时，始终使用主栈指针 (MSP)。当处理器以线程模式运行时，可以使用 MSP 或进程栈指针 (PSP)，具体取决于 CONTROL 寄存器中 SPSEL 位的配置。

在 CPURST 之后，处理器会自动无条件地从主闪存的第一个地址 (0x0000.0000) 提取默认栈指针作为主栈指针 (MSP)。

连接寄存器 (R14)

R14 用作连接寄存器，包含函数调用和异常的返回值。连接寄存器必须设置后方可使用，因为它不会复位为任何已知值。在特权模式和非特权模式下均可访问。

程序计数器寄存器 (R15)

程序计数器寄存器 (R15) 包含下一条待执行指令的地址。在特权和非特权模式下可访问 PC。

在 CPURST 之后，处理器会自动无条件地从主闪存的第二个字 (0x0000.0004) 提取默认 PC。

专用寄存器

专用寄存器包括程序状态寄存器 (PSR)、中断屏蔽寄存器 (PRIMASK) 和控制寄存器 (CONTROL)。专用寄存器通常通过 CPS、MRS 和 MSR 系统指令进行访问。

- **程序状态寄存器 (PSR)：**PSR 是应用状态 (APSR)、中断状态 (IPSR) 和执行状态 (EPSR) 寄存器的组合。应用软件可以使用 MRS 和 MSR 指令访问 PSR，访问完整的 PSR 或一个或多个寄存器的组合，但有一些限制。可以使用表 3-1 中给出的助记符，通过 MRS 和 MSR 指令访问 PSR 寄存器。
  - 应用状态寄存器 (APSR) 包含 N、Z、C 和 V 标志，这些标志被处理器用来评估条件分支指令。这些位分别位于 PSR 的 BIT31、BIT30、BIT29 和 BIT28 中。
  - 当处于处理程序模式时，中断状态寄存器 (IPSR) 会报告目前正在执行异常的当前编号。在线程模式下，它读取为零。处理器忽略对该寄存器的写入。异常编号字段显示在 PSR 的 BIT0 到 BIT5 中。
  - 执行状态寄存器 (EPSR) 包含 T 位 (BIT24)，它定义了处理器是否处于 Thumb 状态。该位无法由软件读取或写入，但供处理器使用。
- **中断屏蔽寄存器 (PRIMASK)：**PRIMASK 寄存器 (PM) 的 BIT0 可用于屏蔽所有优先级可配置的处理器中断（请参阅节 3.3）。这可以被视为全局外设中断屏蔽控制。处理器忽略对 PRIMASK 的非特权写入。将 PM 清零会启用中断；将 PM 置位可禁用中断。CPS 指令可用于更改 PRIMASK 寄存器中的 PM 位值。
- **控制寄存器 (CONTROL)：**通过分别清零或置位 nPRIV 位 (BIT0)，控制寄存器可用于定义在线程模式下执行的代码是特权级还是非特权级。通过分别清零或置位 SPSEL 位 (BIT1)，控制寄存器还可用于选择所用的 R13 栈指针作为主栈指针 (MSP) 或进程栈指针 (PSP)。CPURST 将 CONTROL 寄存器清零。处理器忽略对 CONTROL 寄存器的非特权写入。当进入异常并从异常返回时，处理器会自动更新 SPSEL 栈指针选择位。请注意，软件必须在写入 CONTROL 后执行 ISB 边界指令，以确保所有更改都在处理器执行下一条应用指令之前生效。

表 3-1. 程序状态寄存器 (PSR) 访问助记符

助记符	包括子寄存器
APSR	APSR
IPSR	IPSR
EPSR	EPSR

表 3-1. 程序状态寄存器 (PSR) 访问助记符 (continued)

助记符	包括子寄存器
IAPSR	IPSR 和 APSR
EAPSR	EPSR 和 APSR
XPSR	APSR、IPSR、EPSR
IEPSR	IPSR 和 EPSR

3.2.2 堆栈行为

Arm Cortex-M0+ 处理器采用一种完整的降序堆栈协议。堆栈指针寄存器 (SP) 始终指示最后一个堆栈数据的位置。将新数据添加到调用堆栈时，SP 值将递减，而数据将写入由 SP 递减之后指示的位置。

Arm Cortex-M0+ 支持使用两个指针来管理两个独立的堆栈：主堆栈 (MSP) 和进程堆栈 (PSP)。

3.2.3 执行模式和特权等级

该处理器支持两种主要的执行模式：

- **线程模式**（用于执行应用软件）
- **处理程序模式**（用于处理处理器异常和外设中断）

默认情况下，处理器在复位后处于线程模式。如果向处理器发出异常，处理器将在处理程序模式下处理该异常，并在处理程序执行完成后恢复为线程模式。根据处理器内部 CONTROL 寄存器的配置，在线程模式下运行的代码可以配置为特权级或非特权级。在处理程序模式下运行的代码始终以特权身份执行。

一般而言，以特权身份执行的代码可以完全控制处理器配置，包括对 MPU、 SysTick、NVIC 和 SCB 的控制。只有特权代码才能更改线程模式下运行的代码的特权等级。

在非特权状态下以线程模式执行的代码无法访问前面提到的资源（MPU、SysTick、NVIC、SCB），如果已配置并启用 MPU，还可能受 MPU 限制。

3.2.4 地址空间和支持的数据大小

MSPM0 器件采用一种具有 32 位字节可寻址地址空间的平面存储器映射。字节地址是介于 0 至 2^{32}-1 的无符号数。

地址空间

处理器认为地址空间包含 2^{30} 个 32 位字，每个字都采用字对齐（4 字节对齐）方式。指针始终为 32 位，堆栈操作（例如，push、pop）使堆栈指针递增 4 个地址（4 个字节）。如果处理器的地址计算结果从 32 位存储器空间上溢或下溢，则会绕回。

处理器的指令获取始终采用 16 位半字对齐方式。

处理器的数据读取必须采用自然对齐方式（例如，字必须为字对齐，半字必须为半字对齐，等等）。

支持的数据大小

处理器支持 8 位字节、16 位半字和 32 位字数据大小。支持有符号和无符号数据，其中，有符号数据以 32 位二进制补码格式存储在 CPU 寄存器中。Armv6-M 指令集没有本机指令支持对 64 位双字数据进行操作。

当数据小于 32 位时，从存储器到 CPU 寄存器的加载操作可以是有符号或无符号操作。将无符号半字数据或字节数据加载到 CPU 寄存器时，该值会使用零自动扩展为 32 位。将有符号半字数据或字节数据加载到 CPU 寄存器时，该值会使用符号自动扩展为 32 位。

从 CPU 寄存器到存储器的存储具有不可知的符号。

所有指令和数据访问都使用小端字节序。

3.3 中断和异常

外设中断异常和系统异常会暂停处理器的正常执行流程，以便处理器可用于处理事件。

以下情况可能会导致正常执行流程中断：

- CPURST
- 不可屏蔽中断（NMI，来自 SYSCTL 的软件触发或硬件错误信号）
- 系统中的故障异常（硬故障）
- 管理程序调用指令执行 (SVCall)
- 设置挂起的管理程序服务请求 (PendSV)
- SysTick 异常
- 已启用的外设中断 (IRQ)
- 断点指令（用于调试）

异常状态

在任何给定时间点，处理器的每个异常源都将处于以下状态之一：

- **非活动**（未激活、未挂起）
- **挂起**（等待处理器提供服务）
- **活动**（由处理器主动提供服务）
- **活动和挂起**（当同一个源产生另一个异常时，由处理器主动提供服务）

异常优先级划分、进入和退出

异常由处理器和嵌套矢量中断控制器 (NVIC) 进行优先级排序。每个异常都有一个固定的优先级（复位、NMI、硬故障）或者一个可配置的优先级（SVCall、PendSV、SysTick、外设 IRQ）。具有可配置优先级的异常可由在特权模式下运行的应用软件禁用。具有固定优先级的异常无法禁用。

处理器异常模式支持占先、尾链和延迟到达功能，以提升异常处理性能：

- 在**占先**模式下，如果在执行优先级较低的异常时挂起优先级较高的异常，则优先级较高的异常将优先于运行中的处理程序，以处理优先级较低的异常。
- 在**尾链**模式下，如果在异常处理程序完成时一个入口有效的异常处于挂起状态，那么应用上下文不会从堆栈中恢复，并且会立即向挂起的异常提供控制。
- 在**延迟到达**模式下，如果在进入较低优先级的异常期间发生较高优先级的异常，则在处理器状态保存到堆栈后，会优先处理较高优先级的异常。在较高优先级的异常处理完成后，较低优先级的异常（仍处于挂起状态）将根据尾链程序进行处理。

如果满足以下所有条件，则会发出异常进入：

- 例外处于**挂起**状态
- 挂起异常的优先级高于异常屏蔽寄存器 (PRIMASK) 设置的限制。
- 处理器当前处于线程模式（不处理异常）或新挂起异常的优先级高于当前正在处理的异常（导致占先）

处理器异常是矢量异常。发生异常时，当前处理器状态会压入事件发生时处于活动状态的堆栈中，执行会被导引至矢量表中与要处理的异常所对应的入口点地址。

如果异常尾链接到已完成的前一个处理程序，则无需将任何状态压入堆栈，并且该异常可以立即被导引至中断服务例程。同样，如果异常的优先级高于先前启动进入但未完成进入的异常，则无需再次保存上下文（延迟到达）。

异常处理程序完成后，如果没有需要处理的异常挂起，处理器将从堆栈中弹出状态，并将处理器恢复到发生异常时的先前状态。

3.3.1 外设中断 (IRQ)

外设中断功能由器件上的几个组件管理：

- 嵌套矢量中断控制器 (NVIC)
- 一个或多个中断组 (INT_GROUP)
- 唤醒控制器 (WUC)

MSPM0 器件包含一个带有 Cortex-M0+ CPU 的 Arm 嵌套矢量中断控制器 (NVIC)，用于管理外设中断。NVIC 操作与处理器紧密集成，支持多达 32 个本机外设中断源。

除了 NVIC 外，器件上还可以存在中断分组模块，以便能够将 32 个以上的外设中断连接到 NVIC。可能需要占先功能的高优先级中断源会直接映射到 NVIC，其工作方式与正常的 NVIC 中断类似。通常不需要占先功能的低优先级中断源会映射到一个中断分组模块，然后该模块的输出会映射到 NVIC 作为一个本机 NVIC 中断源。此布线布局如图 3-3 所示。

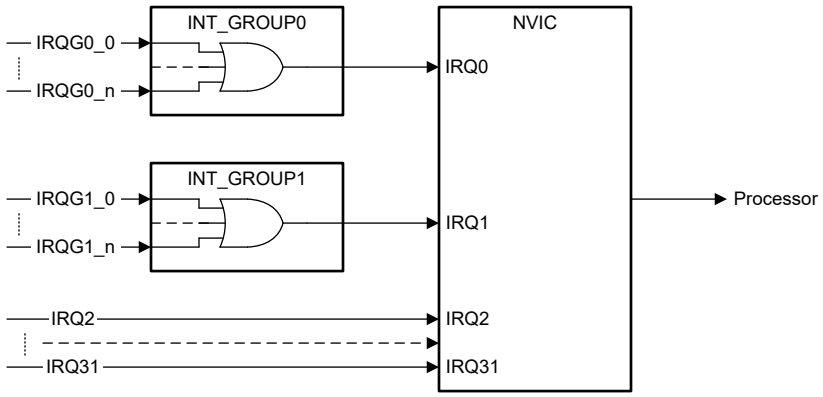


图 3-3. 外设中断层次结构

如果 PD1 电源域在停止或待机模式下断电，唤醒控制器 (WUC) 将确定是否需要为 PD1 电源域（包含处理器）供电，以处理外设中断。

3.3.1.1 嵌套矢量中断控制器 (NVIC)

嵌套矢量中断控制器 (NVIC) 是一个业界通用的 Arm 组件，可将外设中断（位于处理器外部）连接到 CPU。NVIC 支持连接多达 32 个本机外设中断源。

NVIC 通过系统专用外设总线 (PPB) 区域中的存储器映射寄存器进行配置。请参阅表 3-2 中的 NVIC 寄存器列表。随器件提供的软件开发套件 (SDK) 支持 NVIC 的标准 Arm Cortex 微控制器软件接口标准 (CMSIS) 寄存器访问定义。访问任何 NVIC 寄存器时，应用软件必须使用 32 位对齐的字大小事务。

除了将外设中断连接到处理器之外，NVIC 还支持对每个中断的优先级进行编程。

启用和禁用中断

可以通过 NVIC 中的中断设置使能 (ISER) 和中断清除使能 (ICER) 寄存器来读取、设置和清除外设中断的启用状态。32 个中断映射到 ISER 和 ICER 寄存器，其中的中断 0 位于每个寄存器的 BIT0 位置 (LSB)，中断 31 位于每个寄存器的 BIT31 位置 (MSB)。要启用中断，请设置 ISER 寄存器中相应的使能位。向 ISER 写入“0”无效。可以通过读取 ISER 寄存器来确定启用了哪些中断。读取时，“1”表示启用中断；“0”表示禁用。要禁用中断，请设置 ICER 寄存器中相应的使能位。向 ICER 写入“0”无效。

备注

除了在 NVIC 中启用外设中断，通常还需要配置相应外设的中断配置。大多数外设有多个中断源，这些中断源在外设中合并在一起以产生单个 NVIC 中断。各个外设中断的屏蔽在外设的中断管理寄存器内完成。

在 NVIC 已禁用某个中断的情况下，如果该中断由相应的外设设置为有效状态，则 NVIC 中断将进入挂起状态，但不会中断处理器。如果中断在有效状态时（处理程序正在运行时）被禁用，它将保持有效状态，直到异常处理程序返回结果或发生复位，但不会进一步激活。

备注

如果外设向 NVIC 发出中断，但在 NVIC 中禁用了该外设的中断，则由于唤醒控制器 (WUC) 正在为处理器保留事件，该器件可能会保持比预期更高的功率模式。为了避免这种情况，请确保直接在外设上屏蔽外设中断，而不是只在 NVIC 上屏蔽中断。

设置和清除中断挂起状态

可以通过 NVIC 中的中断设置挂起 (ISPR) 和中断清除挂起 (ICPR) 寄存器来读取、设置和清除中断挂起状态。32 个中断映射到 ISPR 和 ICPR 寄存器，其中的中断 0 位于每个寄存器的 BIT0 位置 (LSB)，中断 31 位于每个寄存器的 BIT31 位置 (MSB)。要读取某个中断是否为挂起状态，请读取 ISPR 或 ICPR。读取时，“1”表示中断为挂起状态；“0”表示非挂起状态。要通过软件将中断设置为挂起状态，请设置 ISPR 寄存器中的相应位。向 ISPR 写入“0”无效。要清除中断挂起状态，请设置 ICPR 寄存器中的相应位。向 ICPR 写入“0”无效。请注意，如果外设中断条件仍然存在，即使挂起状态被清除也将由硬件再次设置。

设置中断优先级

NVIC 上的中断具有可编程的优先级。可能有四个优先级。设置优先级的方法是对 NVIC 中的八个 IPRx 寄存器进行编程。每个优先级字段的长度为 8 位，因此每个 32 位寄存器可配置 4 个中断的优先级。Arm Cortex-M0+ 仅实现每个 8 位优先级字段的最高有效 2 位（提供 4 个优先级）。较低的优先级值具有较高的优先级。系统异常（复位、NMI、硬故障）分别具有固定的优先级 -3、-2 和 -1。因此，这些异常的优先级总是高于外设中断。外设中断优先级可编程为 0、64、128 或 192，其中 0 为最高优先级，192 为最低优先级。

如果处理器当前正在处理一个异常，那么这个异常只能被一个更高优先级的异常抢占。如果处于挂起状态下的多个异常都分配到相同的优先级，则首先处理具有最低异常编号的异常。

备注

当某个中断为有效（正在处理）或启用状态时，应用软件不得改变相应中断的优先级。这样做会导致不可预知的行为。

表 3-2. Arm Cortex-M0+ NVIC 寄存器

地址	寄存器	CMSIS	说明
0xE000.E100	NVIC_ISER	NVIC->ISER[0]	中断设置使能寄存器
0xE000.E180	NVIC_ICER	NVIC->ICER[0]	中断清除使能寄存器
0xE000.E200	NVIC_ISPR	NVIC->ISPR[0]	中断设置挂起寄存器
0xE000.E280	NVIC_ICPR	NVIC->ICPR[0]	中断清除挂起寄存器
0xE000.E400	NVIC_IPR0	NVIC->IP[0]	中断优先级寄存器 (0-3)
0xE000.E404	NVIC_IPR1	NVIC->IP[1]	中断优先级寄存器 (4-7)
0xE000.E408	NVIC_IPR2	NVIC->IP[2]	中断优先级寄存器 (8-11)
0xE000.E40C	NVIC_IPR3	NVIC->IP[3]	中断优先级寄存器 (12-15)
0xE000.E410	NVIC_IPR4	NVIC->IP[4]	中断优先级寄存器 (16-19)
0xE000.E414	NVIC_IPR5	NVIC->IP[5]	中断优先级寄存器 (20-23)
0xE000.E418	NVIC_IPR6	NVIC->IP[6]	中断优先级寄存器 (24-27)
0xE000.E41C	NVIC_IPR7	NVIC->IP[7]	中断优先级寄存器 (28-31)

3.3.1.2 中断组

为了支持 32 个以上外设中断源到 NVIC 的映射，某些 MSPM0 器件在 MCPUSS 中包含中断分组逻辑 (INT_GROUP)，以组合多个中断来提供一个本机 NVIC 中断。

INT_GROUP 中断分组使用 MSPM0 事件管理寄存器结构，其主要区别在于，中断组的所有外设中断源始终为未屏蔽（始终启用），因此除了外设中断配置和 NVIC 配置外，不需要额外的启用配置。IMASK 寄存器本身是只读的，并且通过硬接线来启用中断组的所有源。图 3-4 展示了 INT_GROUP 结构。

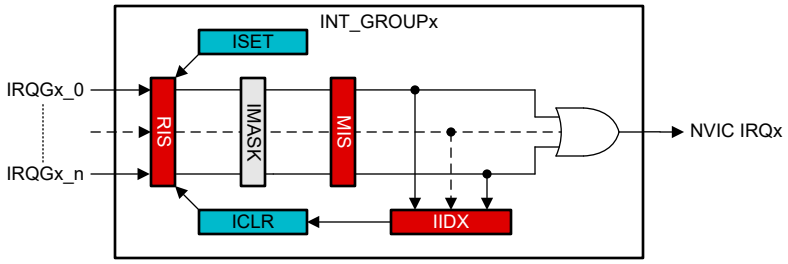


图 3-4. 中断组 (INT_GROUP)

由于不需要屏蔽控制，应用软件只需与中断索引 (IIDX) 寄存器连接，即可有效处理通过 INT_GROUP 寄存到 NVIC 的外设中断。

应用软件可以读取 INT_GROUP 中的 IIDX 寄存器，以确定和清除该组中最高优先级的挂起外设中断。读取 IIDX 将返回与设置中断的最高优先级外设相对应的索引。读取操作还将同时清除与读取所返回索引的最高优先级中断相对应的 RIS 和 MIS 位。从 IIDX 寄存器读取的值随后可用于选择语句，如下所示。

```
void GROUP_HANDLER(void)
{
  switch(IIDX)
  {
    case 0:         // no IRQ pending
      break;
    case 1:         // IRQ[0]
      do_peripheral_1_ISR();
      break;
    case 2:         // IRQ[1]
      do_peripheral_2_ISR();
      break;
    default:        // out of range
      illegal();
  }
}
```

使用

由于组合到一个 INT_GROUP 中的多个外设中断会提供一个 NVIC 中断，因此无法让一个组中的外设中断优先于中断组的活动处理程序的执行。例如，假设 WWDT0 中断请求线路和 PMCU 中断请求线路连接到 INT_GROUP0，而 INT_GROUP0 提供 NVIC 外设中断 0。如果 WWDT0 中断被置为有效，INT_GROUP0 将向 NVIC 发出中断请求。如果没有更高优先级的中断处于活动状态，NVIC 会将处理器引导至 INT_GROUP0 处理程序。然后，应用软件可以读取 INT_GROUP0 IIDX 寄存器，以确定在 NVIC 上触发 INT_GROUP0 中断的是 WWDT0，并且软件可以跳转到 WWDT0 处理程序函数。

如果在处理器仍处于处理程序模式来处理 WWDT0 函数（实际上是 INT_GROUP0 处理程序的一部分）期间，PMCU 将其中断线路置为有效，则 PMCU 中断不能优先于 WWDT0 处理程序。当 WWDT0 处理程序完成时，INT_GROUP0 处理程序将返回。此时，处理器将看到 INT_GROUP0 请求再次置为有效（这次是由于 PMCU），并将第二个条目尾链到中断处理程序。这时，应用软件将读取 IIDX 并确定 PMCU 是向 NVIC 发出 INT_GROUP0 中断的原因。

如果有两个或多个的外设中断挂起到一个中断组，软件可以设置中断处理的优先级，方法是先读取 RIS 或 MIS 寄存器以测试哪个外设中断置为有效，然后执行软件确定的优先级。或者，如果使用中断索引 (IIDX) 寄存器，则中断组硬件将根据索引顺序返回最高优先级索引。

3.3.1.3 唤醒控制器 (WUC)

唤醒控制器 (WUC) 负责当处理器在 STOP 或 STANDBY 工作模式下断电时监控是否产生了中断。在这些模式下，整个 PD1 电源域采用电源门控，因此处理器和 NVIC 不可用于检查中断。当处理器进入 STOP 或 STANDBY 模式时，WUC 会保留一份已启用的 NVIC 外设中断源的副本。如果发出已启用的中断，WUC 将与 PMCU 握手以使器件退出 STOP 或 STANDBY 模式，以便 CPU 能够处理中断。当处理器启动时，WUC 将捕捉中断状态并将其

呈现给 NVIC 和处理器，这样即使在处理器完成上电以便处理中断之前外设的原始中断状态已被移除，处理器也会看到中断。

WUC 在进入或退出低功耗模式时不需要由应用软件进行任何配置，并且其运行是透明的。

3.3.2 中断和异常表

MSPM0 器件会在不同器件之间共享通用中断和异常映射。特定器件无法实现所有中断源，但如果外设是两个器件共用的，则它将具有到这两个器件上 NVIC 的相同映射。

有关特定器件支持的完整中断列表，请参阅特定于器件的数据表。

Arm Cortex-M0+ 中断矢量表的长度为 48 个字（192 字节）。表 3-3 中给出了带有矢量表地址的完整平台中断和异常表。

表 3-3. MSPM0 平台处理器中断和异常表

异常编号	NVIC 编号(1)	优先级组	异常或中断	矢量表地址	矢量说明
-	-	-	-	0x0000.0000	栈指针
1	-	-3	复位	0x0000.0004	复位向量
2	-	-2	NMI	0x0000.0008	NMI 处理程序
3	-	-1	硬故障	0x0000.000C	硬故障处理程序
4	-	-	被保留	0x0000.0010	-
5	-	-	被保留	0x0000.0014	-
6	-	-	被保留	0x0000.0018	-
7	-	-	被保留	0x0000.001C	-
8	-	-	被保留	0x0000.0020	-
9	-	-	被保留	0x0000.0024	-
10	-	-	被保留	0x0000.0028	-
11	-	可选	SVCall	0x0000.002C	管理程序调用处理程序
12	-	-	被保留	0x0000.0030	-
13	-	-	被保留	0x0000.0034	-
14	-	可选	PendSV	0x0000.0038	挂起管理程序处理程序
15	-	可选	SysTick	0x0000.003C	SysTick 处理程序
16	0	可选	INT_GROUP0	0x0000.0040	组合外设组 0 处理程序（请参阅下面的 INT_GROUP0）
17	1	可选	INT_GROUP1	0x0000.0044	组合外设组 1 处理程序（请参阅下面的 INT_GROUP1）
18	2	可选	TIMG8	0x0000.0048	计时器 TIMG8 中断处理程序
19	3	可选	UART3	0x0000.004C	UART3 中断处理程序
20	4	可选	ADC0	0x0000.0050	ADC0 中断处理器
21	5	可选	ADC1	0x0000.0054	ADC1 中断处理程序
22	6	可选	CANFD0	0x0000.0058	CAN-FD 控制器中断处理程序
23	7	可选	DAC0	0x0000.005C	DAC0 中断处理程序
24	8	可选	被保留	0x0000.0060	
25	9	可选	SPI0	0x0000.0064	SPI0 中断处理器
26	10	可选	SPI1	0x0000.0068	SPI1 中断处理程序
27	11	可选	被保留	0x0000.006C	-
28	12	可选	被保留	0x0000.0070	-
29	13	可选	UART1	0x0000.0074	UART1 中断处理器
30	14	可选	UART2	0x0000.0078	UART2 中断处理程序
31	15	可选	UART0	0x0000.007C	UART0 中断处理器

表 3-3. MSPM0 平台处理器中断和异常表 (continued)

异常编号	NVIC 编号(1)	优先级组	异常或中断	矢量表地址	矢量说明
32	16	可选	TIMG0	0x0000.0080	计时器 TIMG0 中断处理程序
33	17	可选	TIMG6	0x0000.0084	计时器 TIMG6 中断处理程序
34	18	可选	TIMA0	0x0000.0088	计时器 TIMA0 中断处理程序
35	19	可选	TIMA1	0x0000.008C	计时器 TIMA1 中断处理程序
36	20	可选	TIMG7	0x0000.0090	计时器 TIMG7 中断处理程序
37	21	可选	TIMG12	0x0000.0094	计时器 TIMG12 中断处理程序
38	22	可选	被保留	0x0000.0098	-
39	23	可选	被保留	0x0000.009C	-
40	24	可选	I2C0	0x0000.00A0	I2C0 中断处理器
41	25	可选	I2C1	0x0000.00A4	I2C1 中断处理器
42	26	可选	被保留	0x0000.00A8	-
43	27	可选	被保留	0x0000.00AC	-
44	28	可选	AES	0x0000.00B0	AES 加速器中断处理程序
45	29	可选	被保留	0x0000.00B4	-
46	30	可选	RTC	0x0000.00B8	RTC 中断处理程序
47	31	可选	DMA	0x0000.00BC	DMA 中断处理器

(1) 如果多个 NVIC 中断具有相同的组优先级，NVIC 编号还指示相对中断优先级。但是，一个中断不会优先于另一个具有相同组优先级的中断的活动处理程序，即使该中断具有更高（数值较小）的 NVIC 位置。为了实现占先，新的中断必须配置为一个更高的优先级组（数值较小）。

不可屏蔽的中断(NMI)

CPU 实现了一个不可屏蔽中断，此中断处理必须由处理器立即处理的关键中断。NMI 中断源由 SYSCTL 管理。请参阅“PMCU”一章内“SYSCTL”部分中的相应 NMI 信息。

INT_GROUP0 外设中断组

如果组中的任何外设具有挂起的中断，INT_GROUP0 外设中断组会向 NVIC0 发出中断（异常 16）。表 3-4 中给出了映射到 INT_GROUP0 的外设中断。

表 3-4. INT_GROUP0 中断

优先级	IIDX 索引	中断	说明
0	1	WWDT0	WWDT0 中断处理器
1	2	WWDT1	WWDT1 中断处理程序
2	3	DEBUGSS	调试子系统中断处理程序
3	4	FLASHCTL	闪存控制器中断处理程序
4	5	WUC FSUB0	通用事件订阅者 0 中断处理程序
5	6	WUC FSUB1	通用事件订阅者 1 中断处理程序
6	7	PMCU (SYSCTL)	PMCU（系统控制器）中断处理程序
7	8	保留	-

INT_GROUP1 外设中断组

如果组中的任何外设具有挂起的中断，INT_GROUP1 外设中断组会向 NVIC1 发出中断（异常 17）。表 3-5 中给出了映射到 INT_GROUP1 的外设中断。

表 3-5. INT_GROUP1 中断

优先级	IIDX 索引	中断	说明
0	1	GPIO0	GPIO0 中断处理器
1	2	GPIO1	GPIO1 中断处理程序
2	3	COMP0	COMP0 中断处理器
3	4	COMP1	COMP1 中断处理程序
4	5	COMP2	COMP2 中断处理程序
5	6	TRNG	TRNG 中断处理程序
6	7	保留	-
7	8	保留	-

3.3.3 处理器锁定方案

有几种异常情况会导致处理器进入锁定状态。在 MSPM0 器件上，处理器锁定被视为致命故障，始终会触发 SYSRST 以清除锁定条件并重新启动系统。

如果在处理器处理优先级为 -1 或更高（数值更低）的异常时发生 SVC（监控器调用）或故障情况，处理器将进入锁定状态。在正常运行条件下，处理器会将此类故障视为意外故障。

以下示例是可以触发处理器中锁定状态的条件：

- 处理器无法在复位时获取堆栈指针或复位矢量
- 处理器无法获取 NMI 矢量
- 处理器无法获取硬故障矢量
- 当处理器已在处理优先级为 -1 或 -2 的异常（硬故障或 NMI）时，会发生存储器故障
- 当处理器已在处理优先级为 -1 或 -2 的异常（硬故障或 NMI）时，会发生监控器调用 (SVC)
- 当处理器已在处理优先级为 -1 或 -2 的异常（硬故障或 NMI）时，会获取使用故障或未定义指令
- 当处理器已在处理优先级为 -1 或 -2 的异常（硬故障或 NMI）时，会执行 BKPT 指令

3.4 CPU 外设

Arm Cortex-M0+ 包含紧密耦合的外设，可实现系统时序和存储器保护。

3.4.1 系统控制模块 (SCB)

系统控制块 (SCB) 提供系统实现信息和系统控制功能，以及系统异常配置、控制和报告。

可通过系统专用外设总线 (PPB) 区域中的存储器映射寄存器配置 SCB。请参阅表 3-6 中的 SCB 寄存器列表。随器件提供的软件开发套件 (SDK) 对于 SCB 支持标准 Arm Cortex 微控制器软件接口标准 (CMSIS) 寄存器访问定义。访问任何 SCB 寄存器时，应用软件必须使用 32 位对齐的字大小事务。

表 3-6. Arm Cortex-M0+ 系统控制块寄存器

地址	寄存器	CMSIS	说明
0xE000.ED00	CPUID	SCB->CPUID	指示 CPU 类型和版本的只读寄存器
0xE000.ED04	ICSR	SCB->ICSR	提供特定的中断控制和状态
0xE000.ED08	VTOR	SCB->VTOR	用于指定从 0x0000.0000 开始的矢量表偏移量
0xE000.ED0C	AIRCR	SCB->AIRCR	用于发出 CPU 复位请求 (SYSRESETREQ)
0xE000.ED10	SCR	SCB->SCR	用于控制低功耗模式行为的系统控制寄存器
0xE000.ED14	CCR	SCB->CCR	指示处理器行为的只读寄存器
0xE000.ED1C	SHPR2	SCB->SHP[2]	用于配置 SVCall 系统处理程序的优先级
0xE000.ED20	SHPR3	SCB->SHP[3]	用于配置 SysTick 和 PendSV 系统处理程序的优先级

有关系统控制块寄存器配置的详细信息，请参阅 Arm Cortex-M0+ 器件通用用户指南的 SCB 部分。

3.4.2 系统时钟周期计时器 (SysTick)

系统时钟周期计时器 (SysTick) 是由 MCLK 提供时钟的紧密耦合计时器，可用于系统计时。SysTick 可用于 RUN 和 SLEEP 模式，但不可用于 STOP、STANDBY 和 SHUTDOWN 模式。

可通过系统专用外设总线 (PPB) 区域中的存储器映射寄存器配置 SysTick。请参阅表 3-7 中的 SysTick 配置寄存器列表。随器件提供的软件开发套件 (SDK) 对于 SysTick 支持标准 Arm Cortex 微控制器软件接口标准 (CMSIS) 寄存器访问定义。

SysTick 计时器有多种不同的使用方式，包括：

- 作为 RTOS 计时器，它以可编程的速率（例如 100Hz）触发并调用 SysTick 例程
- 作为高速警报计时器
- 作为简单计数器，用于测量完成时间以及在应用中使用的时间
- 作为超时计数器，用于确认例程在指定的时间段内未超时

SysTick 计时器是一种简单的 24 位向下计数器，从重载值开始递减至零。达到零后，SysTick 将在下一个时钟周期重新加载编程到重载值寄存器 (SYST_RVR) 中的值，然后再次开始递减至零。

当 SysTick 计数器达到零时会生成 SysTick 事件，此时将设置 COUNTFLAG 状态标志。读取 SYST_CSR 寄存器将清除 COUNTFLAG 状态位。写入到当前值寄存器 (SYST_CVR) 会清除该寄存器和 COUNTFLAG 状态，但不会对 CPU 产生中断。读取 SYST_CVR 会返回访问时的计数器值。

表 3-7. SysTick 寄存器

地址	寄存器	CMSIS	说明
0xE000.E010	SYST_CSR	SysTick->CTRL	控制和状态寄存器，用于启用/禁用 SysTick 及其相关异常，以及检查 COUNTFLAG 状态
0xE000.E014	SYST_RVR	SysTick->LOAD	重载寄存器，用于对计数器重载值进行编程以设置 MCLK 周期中的 SysTick 间隔
0xE000.E018	SYST_CVR	SysTick->VAL	返回 SysTick 计数器的当前值；写入到该寄存器会将计数器清零并清除 SYST_CSR 中的 COUNTFLAG
0xE000.E01C	SYST_CALIB	SysTick->CALIB	未执行

应用软件只能使用 32 位字对齐的字访问方式访问 SysTick 寄存器。要初始化 SysTick，请按照以下步骤操作：

1. 将所需的重载值编程到 SYST_RVR 中（例如，如果需要每 1000 个 MCLK 周期生成一个标志，请编程为 999）
2. 通过写入到 SYST_CVR 寄存器来清除当前值
3. 对 SYST_CSR 寄存器进行编程以启用 SysTick

备注

当 CPU 暂停以进行调试时，SysTick 计数器不会递减。

3.4.3 存储器保护单元 (MPU)

存储器保护单元 (MPU) 可用于根据一组可由编程人员定义的访问权限策略，来检查处理器进行的所有存储器访问。当与 Arm Cortex-M0+ 的特权/非特权执行模式结合使用时，MPU 支持仅限特权代码访问特定的存储器位置。如果非特权代码访问非受限区域，则将继续执行，就好像 MPU 不存在一样。但是，如果非特权代码发出对受限区域的访问，则会在处理器中生成硬故障。还可以限制特权代码和非特权代码进行访问（也即，无法通过处理器进行访问）。

可通过系统专用外设总线 (PPB) 区域中的存储器映射寄存器配置 MPU。有关 MPU 配置寄存器的列表，请参阅表 3-8。随器件提供的软件开发套件 (SDK) 对于 MPU 支持标准 Arm Cortex 微控制器软件接口标准 (CMSIS) 寄存器访问定义。

由于 MPU 会检查来自处理器的所有存储器访问（包括对闪存、SRAM 和外设的访问），因此，它非常适合提高涉及 RTOS 的线程应用程序的可靠性和稳健性。MPU 可用于限制各个线程的存储器访问，包括建立堆栈边界和限制对特定外设的访问。可以保护 RTOS 使用的关键状态数据，以防止非特权线程进行修改。

备注

MPU 仅监控对源自处理器的对于存储器映射的访问。MPU 不限制 DMA 控制器的访问。

备注

使用 MPU 保护闪存区域并不会防止闪存控制器读取或修改闪存。这是因为闪存控制器可以直接控制闪存存储体以执行读取验证、编程和擦除操作。在闪存控制器本身内部提供了专用的写保护功能，当对闪存进行写保护时，应使用这些机制。还可以将 MPU 配置为限制对闪存控制器寄存器的访问，从而防止 CPU 直接配置或启动闪存命令。

MPU 提供了一种机制，可将器件存储器映射划分为 8 个区域（编号为 0-7）加上一个默认的后台区域。可以为每个区域配置其自己的访问权限和存储器属性，如果需要，可以将区域配置为重叠。在区域重叠的情况下，对存在于多个区域中的位置的存储器访问受制于编号最大的区域的属性。

当 MPU 未启用（MPU_CTRL 寄存器中的 ENABLE 位清零）时，器件使用默认存储器映射，并且 CPU 可以访问存储器映射，就像 MPU 不存在一样。

启用 MPU 以供使用后，始终允许访问矢量表和系统控制空间，但能否访问任何其他位置将取决于以下条件：

- 区域配置
- 访问是特权访问还是非特权访问
- 特权软件默认存储器映射访问控制配置 (PRIVDEFENA)

由于 Arm Cortex-M0+ 是单总线 CPU 架构，因此 MPU 的指令提取和数据访问之间没有界限。处理它们的方式是相同的。

备注

如果启用 MPU（设置了 ENABLE 位），则必须设置 PRIVDEFENA 以授予特权软件对存储器映射的访问权限，或者必须配置并启用至少一个存储器区域，以便 CPU 继续运行而不会出现硬故障。如果将 PRIVDEFENA 清零，未定义和启用任何区域，并且启用了 MPU，则整个存储器映射受到限制，任何访问都会生成故障。

表 3-8. MPU 寄存器

地址	寄存器	CMSIS	说明
0xE000.ED90	MPU_TYPE	MPU->TYPE	表示 MPU 存在的类型寄存器。
0xE000.ED94	MPU_CTRL	MPU->CTRL	用于启用和配置 MPU 的 MPU 控制寄存器。
0xE000.ED98	MPU_RNR	MPU->RNR	区域选择寄存器，供您选择使用 MPU_RBAR 和 MPU_RASR。
0xE000.ED9C	MPU_RBAR	MPU->RBAR	区域基地址配置寄存器。
0xE000.EDA0	MPU_RASR	MPU->RASR	区域大小和存储器属性寄存器。

有关 MPU 寄存器配置的详细信息，请参阅《Arm Cortex-M0+ 器件通用用户指南》的 MPU 部分。

3.5 只读存储器 (ROM)

MCPUSS 的一个只读存储器中包含引导配置例程 (BCR) 和引导加载程序 (BSL) 的可执行代码。在执行 BOOTRST 之后或在执行 SYSRST 并经过 BSL 进入/退出之后，该 ROM 将激活。该 ROM 在应用程序启动后会自动禁用，无法由应用软件访问。

3.6 CPUSS 寄存器

表 3-9 列出了 CPUSS 寄存器的存储器映射寄存器。表 3-9 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 3-9. CPUSS 寄存器

偏移	缩写	寄存器名称	组	部分
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	IIDX	中断索引	CPU_INT_GROUP0	转到
1108h	IMASK	中断屏蔽	CPU_INT_GROUP0	转到
1110h	RIS	原始中断状态	CPU_INT_GROUP0	转到
1118h	MIS	已屏蔽中断状态	CPU_INT_GROUP0	转到
1120h	ISET	中断设置	CPU_INT_GROUP0	转到
1128h	ICLR	中断清除	CPU_INT_GROUP0	转到
1130h	IIDX	中断索引	CPU_INT_GROUP1	转到
1138h	IMASK	中断屏蔽	CPU_INT_GROUP1	转到
1140h	RIS	原始中断状态	CPU_INT_GROUP1	转到
1148h	MIS	已屏蔽中断状态	CPU_INT_GROUP1	转到
1150h	ISET	中断设置	CPU_INT_GROUP1	转到
1158h	ICLR	中断清除	CPU_INT_GROUP1	转到
1160h	IIDX	中断索引	CPU_INT_GROUP2	转到
1168h	IMASK	中断屏蔽	CPU_INT_GROUP2	转到
1170h	RIS	原始中断状态	CPU_INT_GROUP2	转到
1178h	MIS	已屏蔽中断状态	CPU_INT_GROUP2	转到
1180h	ISET	中断设置	CPU_INT_GROUP2	转到
1188h	ICLR	中断清除	CPU_INT_GROUP2	转到
1190h	IIDX	中断索引	CPU_INT_GROUP3	转到
1198h	IMASK	中断屏蔽	CPU_INT_GROUP3	转到
11A0h	RIS	原始中断状态	CPU_INT_GROUP3	转到
11A8h	MIS	已屏蔽中断状态	CPU_INT_GROUP3	转到
11B0h	ISET	中断设置	CPU_INT_GROUP3	转到

表 3-9. CPUSS 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
11B8h	ICLR	中断清除	CPU_INT_GROUP3	转到
11C0h	IIDX	中断索引	CPU_INT_GROUP4	转到
11C8h	IMASK	中断屏蔽	CPU_INT_GROUP4	转到
11D0h	RIS	原始中断状态	CPU_INT_GROUP4	转到
11D8h	MIS	已屏蔽中断状态	CPU_INT_GROUP4	转到
11E0h	ISET	中断设置	CPU_INT_GROUP4	转到
11E8h	ICLR	中断清除	CPU_INT_GROUP4	转到
1300h	CTL	预取/缓存控制		转到

复杂的位访问类型经过编码可适应小型表单元。表 3-10 显示了适用于此部分中访问类型的代码。

表 3-10. CPUSS 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

3.6.1 EVT_MODE（偏移 = 10E0h）[复位 = 00000001h]

图 3-5 展示了 EVT_MODE，表 3-11 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 3-5. EVT_MODE

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

表 3-11. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1-0	INT_CFG	R	1h	事件线模式选择 0h = 中断或事件线禁用。 1h = 由软件处理的事件。软件必须清除关联的 RIS 标志。 2h = 由硬件处理的事件。硬件（另一个模块）会自动清除关联的 RIS 标志。

3.6.2 DESC（偏移 = 10FCh）[复位 = 27110000h]

图 3-6 展示了 DESC，表 3-12 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 3-6. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-2711h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-0h				R-				R-0h				R-0h			

表 3-12. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	2711h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本
3-0	MINREV	R	0h	IP 的次要版本

3.6.3 IIDX（偏移 = 1100h）[复位 = 00000000h]

图 3-7 展示了 IIDX，表 3-13 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 3-7. IIDX

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

表 3-13. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 1h = 中断 0 2h = 中断 1 3h = 中断 2 4h = 中断 3 5h = 中断 4 6h = 中断 5 7h = 中断 6 8h = 中断 7

3.6.4 IMASK（偏移 = 1108h）[复位 = 000000FFh]

图 3-8 展示了 IMASK，表 3-14 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 3-8. IMASK

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

表 3-14. IMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	FFh	屏蔽相应的中断 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.5 RIS（偏移 = 1110h）[复位 = 00000000h]

图 3-9 展示了 RIS，表 3-15 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 3-9. RIS

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

表 3-15. RIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT 的原始中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.6 MIS（偏移 = 1118h）[复位 = 00000000h]

图 3-10 展示了 MIS，表 3-16 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 3-10. MIS

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

表 3-16. MIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT0 的屏蔽中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.7 ISET（偏移 = 1120h）[复位 = 00000000h]

图 3-11 展示了 ISET，表 3-17 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 3-11. ISET

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

表 3-17. ISET 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	设置 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.8 ICLR (偏移 = 1128h) [复位 = 00000000h]

图 3-12 展示了 ICLR，表 3-18 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 3-12. ICLR

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

表 3-18. ICLR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	清除 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.9 IIDX（偏移 = 1130h）[复位 = 00000000h]

图 3-13 展示了 IIDX，表 3-19 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 3-13. IIDX

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

表 3-19. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 1h = 中断 0 2h = 中断 1 3h = 中断 2 4h = 中断 3 5h = 中断 4 6h = 中断 5 7h = 中断 6 8h = 中断 7

3.6.10 IMASK（偏移 = 1138h）[复位 = 000000FFh]

图 3-14 展示了 IMASK，表 3-20 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 3-14. IMASK

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

表 3-20. IMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	FFh	屏蔽相应的中断 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.11 RIS（偏移 = 1140h）[复位 = 00000000h]

图 3-15 展示了 RIS，表 3-21 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 3-15. RIS

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

表 3-21. RIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT 的原始中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.12 MIS (偏移 = 1148h) [复位 = 00000000h]

图 3-16 展示了 MIS，表 3-22 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 3-16. MIS

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

表 3-22. MIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT0 的屏蔽中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.13 ISET（偏移 = 1150h）[复位 = 00000000h]

图 3-17 展示了 ISET，表 3-23 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 3-17. ISET

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

表 3-23. ISET 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	设置 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.14 ICLR (偏移 = 1158h) [复位 = 00000000h]

图 3-18 展示了 ICLR，表 3-24 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 3-18. ICLR

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

表 3-24. ICLR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	清除 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.15 IIDX（偏移 = 1160h）[复位 = 00000000h]

图 3-19 展示了 IIDX，表 3-25 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 3-19. IIDX

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

表 3-25. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 1h = 中断 0 2h = 中断 1 3h = 中断 2 4h = 中断 3 5h = 中断 4 6h = 中断 5 7h = 中断 6 8h = 中断 7

3.6.16 IMASK（偏移 = 1168h）[复位 = 000000FFh]

图 3-20 展示了 IMASK，表 3-26 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 3-20. IMASK

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

表 3-26. IMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	FFh	屏蔽相应的中断 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.17 RIS（偏移 = 1170h）[复位 = 00000000h]

图 3-21 展示了 RIS，表 3-27 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 3-21. RIS

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

表 3-27. RIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT 的原始中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.18 MIS (偏移 = 1178h) [复位 = 00000000h]

图 3-22 展示了 MIS，表 3-28 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 3-22. MIS

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

表 3-28. MIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT0 的屏蔽中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.19 ISET（偏移 = 1180h）[复位 = 00000000h]

图 3-23 展示了 ISET，表 3-29 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 3-23. ISET

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

表 3-29. ISET 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	设置 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.20 ICLR (偏移 = 1188h) [复位 = 00000000h]

图 3-24 展示了 ICLR，表 3-30 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 3-24. ICLR

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

表 3-30. ICLR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	清除 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.21 IIDX（偏移 = 1190h）[复位 = 00000000h]

图 3-25 展示了 IIDX，表 3-31 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 3-25. IIDX

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

表 3-31. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 1h = 中断 0 2h = 中断 1 3h = 中断 2 4h = 中断 3 5h = 中断 4 6h = 中断 5 7h = 中断 6 8h = 中断 7

3.6.22 IMASK（偏移 = 1198h）[复位 = 000000FFh]

图 3-26 展示了 IMASK，表 3-32 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 3-26. IMASK

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

表 3-32. IMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	FFh	屏蔽相应的中断 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.23 RIS（偏移 = 11A0h）[复位 = 00000000h]

图 3-27 展示了 RIS，表 3-33 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 3-27. RIS

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

表 3-33. RIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT 的原始中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.24 MIS（偏移 = 11A8h）[复位 = 00000000h]

图 3-28 展示了 MIS，表 3-34 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 3-28. MIS

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

表 3-34. MIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT0 的屏蔽中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.25 ISET（偏移 = 11B0h）[复位 = 00000000h]

图 3-29 展示了 ISET，表 3-35 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 3-29. ISET

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

表 3-35. ISET 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	设置 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.26 ICLR（偏移 = 11B8h）[复位 = 00000000h]

图 3-30 展示了 ICLR，表 3-36 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 3-30. ICLR

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

表 3-36. ICLR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	清除 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.27 IIDX（偏移 = 11C0h）[复位 = 00000000h]

图 3-31 展示了 IIDX，表 3-37 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 3-31. IIDX

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

表 3-37. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 1h = 中断 0 2h = 中断 1 3h = 中断 2 4h = 中断 3 5h = 中断 4 6h = 中断 5 7h = 中断 6 8h = 中断 7

3.6.28 IMASK（偏移 = 11C8h）[复位 = 000000FFh]

图 3-32 展示了 IMASK，表 3-38 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 3-32. IMASK

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

表 3-38. IMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	FFh	屏蔽相应的中断 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.29 RIS（偏移 = 11D0h）[复位 = 00000000h]

图 3-33 展示了 RIS，表 3-39 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 3-33. RIS

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

表 3-39. RIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT 的原始中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.30 MIS（偏移 = 11D8h）[复位 = 00000000h]

图 3-34 展示了 MIS，表 3-40 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 3-34. MIS

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

表 3-40. MIS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	INT	R	0h	INT0 的屏蔽中断状态 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.31 ISET（偏移 = 11E0h）[复位 = 00000000h]

图 3-35 展示了 ISET，表 3-41 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 3-35. ISET

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

表 3-41. ISET 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	设置 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.32 ICLR（偏移 = 11E8h）[复位 = 00000000h]

图 3-36 展示了 ICLR，表 3-42 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 3-36. ICLR

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

表 3-42. ICLR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	W	0h	
7-0	INT	W	0h	清除 RIS 寄存器中的 INT 1h = 中断 0 2h = 中断 1 4h = 中断 2 8h = 中断 3 10h = 中断 4 20h = 中断 5 40h = 中断 6 80h = 中断 7

3.6.33 CTL（偏移 = 1300h）[复位 = 00000007h]

图 3-37 展示了 CTL，表 3-43 中对此进行了介绍。

返回到汇总表。

闪存预取和缓存控制寄存器。

图 3-37. CTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED					LITEN	ICACHE	PREFETCH
R/W-0h					R/W-1h	R/W-1h	R/W-1h

表 3-43. CTL 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2	LITEN	R/W	1h	字面量缓存和预取使能。 该位是 ICACHE/PREFETCH 位的子集，即只有分别设置了 ICACHE 或 PREFETCH 位时，才会发生字面量缓存或字面量预取 启用时，CPUSS 内的缓存和预取结构将会缓存和预取字面量 禁用时，CPUSS 内的缓存和预取结构不会缓存和预取字面量 0h = 字面量缓存禁用 1h = 字面量缓存启用
1	ICACHE	R/W	1h	用于启用/禁用闪存访问时的指令缓存。 0h = 禁用指令缓存。 1h = 启用指令缓存。
0	PREFETCH	R/W	1h	用于启用/禁用到闪存的指令预取。 0h = 禁用指令预取。 1h = 启用指令预取。

3.7 WUC 寄存器

表 3-44 列出了 WUC 寄存器的存储器映射寄存器。表 3-44 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 3-44. WUC 寄存器

偏移	首字母缩写	寄存器名称	部分
400h	FSUB_0	订阅者端口 0	FSUB_0 寄存器（偏移 = 400h）[复位 = 00h]
404h	FSUB_1	订阅者端口 1	FSUB_1 寄存器（偏移 = 404h）[复位 = 00h]

3.7.1 FSUB_0 寄存器（偏移 = 400h）[复位 = 00h]

图 3-38 展示了 FSUB_0，表 3-45 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 3-38. FSUB_0 寄存器

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

表 3-45. FSUB_0 寄存器字段说明

位	字段	类型	复位	说明
31-4	保留	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 连接至通道 ID = CHANID。 0h = 值 0 表示事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

3.7.2 FSUB_1 寄存器（偏移 = 404h）[复位 = 00h]

图 3-39 展示了 FSUB_1，表 3-46 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 3-39. FSUB_1 寄存器

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

表 3-46. FSUB_1 寄存器字段说明

位	字段	类型	复位	说明
31-4	保留	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 连接至通道 ID = CHANID。 0h = 值 0 表示事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

章节 4
DMA

直接存储器存取 (DMA) 控制器模块可将数据从一个地址传输到另一个地址，而无需 CPU 干预。本章对 DMA 控制器的运行进行了说明。

4.1 DMA 概述

DMA 控制器无需 CPU 干预即可将数据从源地址传输到目标地址。例如，DMA 控制器可用于将数据从 ADC 转换存储器移动到 SRAM。

器件可提供多达 16 个 DMA 通道。因此，根据 DMA 通道数量的不同，在这一章中有些特性并不对适用于所有器件。请参阅器件特定的数据表以了解 DMA 的实际通道数。

通过使用 DMA 控制器可增加外设模块的吞吐量。通过使 CPU 保持在睡眠模式，无需将其唤醒就可以从一个外设中移动数据，它也会减少系统功耗。

DMA 控制器的功能包括：

- 多达 16 个独立的传输通道
- 可配置的 DMA 通道优先级
- 字节（8 位）、短字（16 位）、字（32 位）和长字（64 位）或混合字节和字传输能力
- 传输计数器块大小支持传输高达 64k 的任何类型数据
- 可配置的 DMA 传输触发选择
- 六种灵活的寻址模式
- 单一或块传输模式

在图 4-1 中给出了 DMA 控制器的结构图。

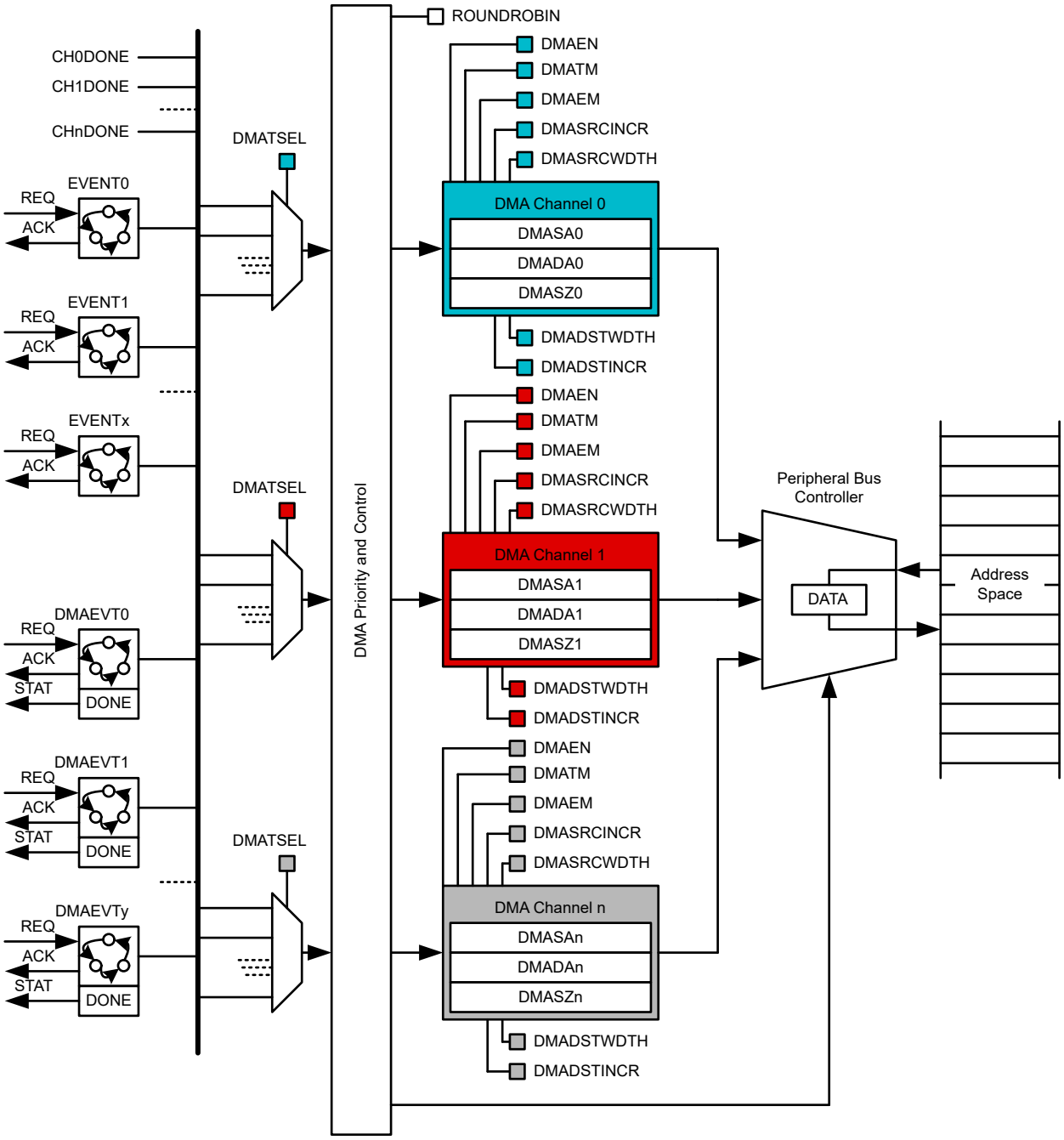


图 4-1. DMA 方框图

4.2 DMA 操作

DMA 控制器由用户软件配置。DMA 的建立和操作将在下面的部分进行讨论。

4.2.1 寻址模式

DMA 控制器有六种寻址模式，每个 DMA 通道的寻址模式都可独立配置。例如，通道 0 可以在两个固定的地址间传输，而通道 1 可在地址的两个块间传输。在图 4-2 中给出了寻址模式。

这些寻址模式是：

1. 固定的地址到固定的地址

2. 固定的地址到地址块
3. 地址块到固定的地址
4. 地址块到地址块
5. 将数据填充到地址块
6. 数据表到特定地址

上面显示的寻址模式 1-4 只需通过 DMASRCINCR 和 DMADSTINCR 控制位进行配置。DMASRCINCR 位选择在每次传输结束后源地址是否不变、增加还是减少。DMADSTINCR 位选择在每次传输结束后目标地址是否不变、增加还是减少。

上面显示的寻址模式 5 和 6 也通过 DMASRCINCR 和 DMADSTINCR 控制位进行配置，同时借助 DMAEM 等附加参数来利用 DMA 的扩展模式。有关如何在填充模式和表格模式下正确配置和使用 DMA 的更多详细信息，请参阅节 4.2.4.1 和节 4.2.4.2。

传输可以是字节到字节、短字到短字、字到字、长字到长字，也可以是这四种方式的任意组合。在以 (short/long) 字到字节的方式进行传输时，仅传输源数据的低字节。将 (long) 字传输到短字时，仅传输源数据的低短字。在以字节到 (short/long) 字的方式进行传输时，在传输时目标字的高字节被清零。将短字传输到 (long) 字时，高位短字在传输发生时清零。将字传输到长字时，高位字被清零。通过将多个源字节传输组合到一个目标 (短) 字或反向，不支持打包或解包。

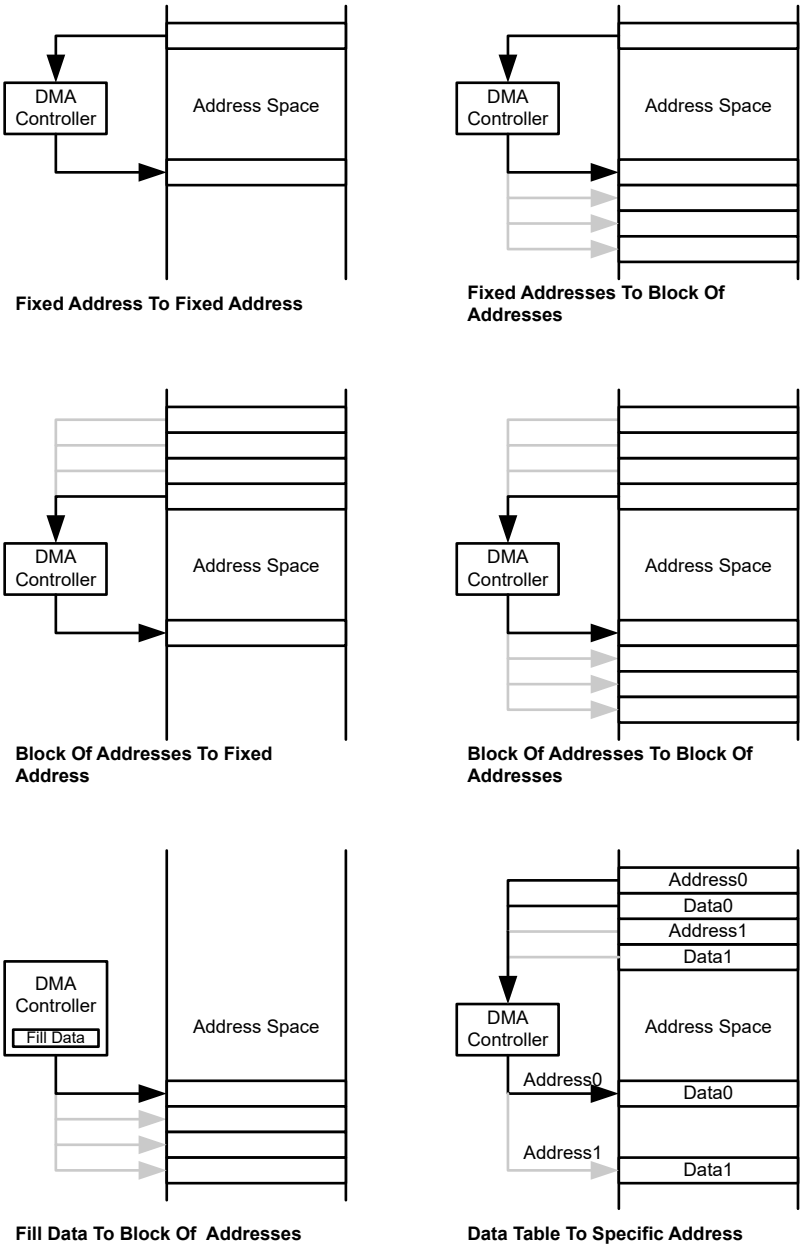


图 4-2. DMA 寻址模式

4.2.2 通道类型

有两种类型的 DMA 通道：基本 (BASIC) 通道和全功能 (FULL) 通道。BASIC 通道仅支持单字或单字节传输或者块传输，而 FULL 通道支持重复的单字或单字节传输和重复块传输，并具有提前中断请求生成以及扩展表和填充模式等附加功能。

最高优先级的 DMA 通道（从 DMA0 开始）是 FULL 通道，其余优先级通道是 BASIC 通道。

备注

请参阅特定于器件的数据表，以确定有多少个 BASIC 通道和 FULL 通道可用。

表 4-1 展示了基本和全功能 DMA 通道支持的功能。

表 4-1. 基本和全功能 DMA 通道的功能比较

DMA 功能	全功能通道	基本通道
重复模式	✓	-
早期 IRQ 通知	✓	-
块突发模式	✓	✓
跨步模式	✓	✓
内部通道作为触发源	✓	✓
扩展模式 (表格和填充模式)	✓	-

4.2.3 传输模式

如表 4-2 所示，DMA 控制器有四种可通过 DMATM 位选择的传输模式。每个通道都可以独立配置其传输模式。例如，通道 0 可配置为重复块传输模式，而通道 1 配置为块传输模式，通道 2 采用单字或单字节传输模式。传输模式独立于寻址模式进行配置。 任何寻址模式都可以使用任何传输模式。

可以通过 DMADSTWDTH 和 DMASRCWDTH 控制位来选择可传输的四种类型的数据。源位置和目标位置可以是字节数据、短字数据、字数据或长字数据。也可以进行字节到字节、短字到短字、字到字、长字到长字或任意组合的传输。

此外，所有传输模式都支持跨步模式，在该模式下，DMA 源和目标可以递增到更高的值来支持数据重组。

表 4-2. DMA 传输模式

DMATM	传输模式	说明	通道类型
0h	单字或单字节传输	每次传输都需要一个单独的触发。DMASZx 传输完成后会自动清除 DMAEN。	基本型
1h	块传输	一个整块将会在一个触发后传输。在块传输结束时 DMAEN 会被自动清零。	基本型
2h	重复单字或单字节传输	每次传输都需要一个单独的触发。 DMAEN 保持启用。	全功能
3h	重复块传输	一个整块将会在一个触发后传输。DMAEN 保持启用。	全功能

4.2.3.1 单字或单字节传输

在单字或单字节传输模式 (DMATM = 0h) 下，每次的字节、半字、字或长字传输都需要一个单独的触发器。单字或单字节传输模式在基本和全功能 DMA 通道中可用。

DMASZx 寄存器可以定义要进行的传输次数。DMADSTINCR 和 DMASRCINCR 位用来选择在每次传输结束后目标地址和源地址是否增加或减少。如果 DMASZx = 0，则不发生传输。

DMASAx、DMADAx 和 DMASZx 寄存器在每次传输后递增或递减。DMADSTWDTH 将指示目标地址在每个传输周期内是递增还是递减 1、2、4 或 8。DMASRCWDTH 和源地址也是如此。当 DMASZx 寄存器递减到零时，设置相应的 RIS 标志。

当 DMASZx 递减至零时，将会自动清除 DMAEN 位，必须重新设置该位才能进行下一次传输。

4.2.3.2 块传输

在块传输模式 (DMATM = 1h) 中，将会在一个触发后开始传输一整块数据。块传输模式仅在基本 DMA 通道中可用。

DMASZx 寄存器用来定义块的大小，且 DMADSTINCR 和 DMASRCINCR 位会选择在每次块传输后目标地址和源地址会递增还是会递减。如果 DMASZx = 0，则不发生传输。

DMASAx、DMADAx 和 DMASZx 寄存器被复制到临时寄存器中。在每次块传输结束后，DMASAx 和 DMADAx 的临时值都会递增或递减。DMADSTWDTH 将指示目标地址在每个传输周期内是递增还是递减 1、2、4 或 8。DMASRCWDTH 和源地址也是如此。在每次块传输结束后，DMASZx 寄存器中的值会递减，并且会指示该块中还剩余多少数据。当 DMATM = 01 时，DMAEN 位将会被自动清零，当 DMASZx 减至 0 时必须重新设置才能进行下一次传输。

DMAEN 位在块传输完成后被清零，并且必须再次设置才能触发另一个块传输。块传输一旦开始，在块传输期间发生的另一个触发信号将被忽略。

4.2.3.3 重复单字或单字节传输

在重复单字或单字节传输模式 (DMATM = 2h) 下，DMA 控制器在 DMAEN=1 时保持启用状态，每次发生触发时都会进行传输。重复单字或单字节传输模式仅在全功能 DMA 通道中可用。

DMASAx、DMADAx 和 DMASZx 寄存器被复制到临时寄存器中。在每次传输结束后，DMASAx 和 DMADAx 的临时值都会递增或递减。DMASZx 寄存器在每个寄存器之后递减。DMADSTWDTH 将指示目标地址在每个传输周期内是递增还是递减 1、2、4 或 8。DMASRCWDTH 和源地址也是如此。当 DMASZx 寄存器的值递减至零时，将会从其临时寄存器重新加载，并设置相应的 RIS 标志。

备注

当使用重复单字或单字节传输模式时，DMA 不支持通过禁用通道（暂停）然后重新启用通道（继续）来暂停和继续传输。

4.2.3.4 重复块传输

在重复块传输模式 (DMATM = 11) 下，DMAEN 位在块传输完成后保持置位。一个重复块传输完成后的下一次触发将触发另一次块传输。重复块传输模式仅在全功能 DMA 通道中可用。

DMASAx、DMADAx 和 DMASZx 寄存器被复制到临时寄存器中。在每次块传输结束后，DMASAx 和 DMADAx 的临时值都会递增或递减。DMADSTWDTH 将指示目标地址在每个传输周期内是递增还是递减 1、2、4 或 8。DMASRCWDTH 和源地址也是如此。在每次块传输结束后，DMASZx 寄存器中的值会递减，并且会指示该块中还剩余多少数据。

4.2.3.5 跨步模式

所有传输模式都支持“跨步”模式，在该模式下，DMA 源和目标可以在传输后递增到更高的值（而不是 +1）。这种模式有助于重新组织源和目标之间的数据顺序。

要支持增量跨步，请将 DMADSTINCR 和/或 DMASRCINCR 设置为 STRIDE_n，其中的 n 是目标和/或源增量的数量。实际增量分别基于 DMADSTWDTH 和/或 DMASRCWDTH 定义。例如，如果外部 ADC 数据以六字 SPI 帧的形式传输到 MCU，则在块传输期间 DMADSTINCR 可设置为 STRIDE_6，以便目标地址增加 6，并整理数据以使处理更轻松。

4.2.4 扩展模式

在完整通道中，DMA 控制器有 2 种可通过 DMAEM 位进行选择的扩展模式，如表 4-3 所示。每个通道均可针对扩展模式单独配置。例如，通道 0 可在表格模式下配置，而通道 1 可在填充模式下配置。

表 4-3. DMA 扩展模式

DMAEM	扩展模式	说明
00、01	正常模式	操作由 DMATM 定义
10	填充模式	用于将预定义数据模式填充到存储器中
11	表格模式	用于帮助配置外设控制寄存器表

4.2.4.1 填充模式

在填充模式 (DMAEM = 10b) 下，DMA 控制器采用预定义的 FILL 模式，并将该模式写入用户定义的存储器区段。DMATM 位被忽略，并且使用的自动传输模式为“块传输”。

DMASAx 寄存器用作 FILL 模式数据。DMASRCINCR 位字段用于指示 FILL 模式数据是应保持恒定，还是应随着每个写入周期而递增/递减。此特性允许使用顺序模式（例如，0、1、2、3、⋯）填充存储器块。DMASRCWDTH 位字段指示 FILL 模式数据的增量幅度。请参阅表 4-4，了解如何在填充模式下使用 DMASRCWDTH。

表 4-4. 填充模式下的 DMASRCWDTH

DMASRCWDTH	FILL 模式数据增量值
0	±1
1	±2
2	±4
3	±8

目标寄存器以及位字段 DMADAx、DMADSTINCR 和 DMADSTWDTH 均按预期运行，并影响 FILL 模式在存储器中的写入位置和写入方式。

4.2.4.2 表模式

在表模式 (DMAEM = 11b) 下，DMA 控制器从源执行 2 次读取，并向确定的目标执行 1 次写入。此特性可用于解释地址和数据表，并且使用 DMA 无需 CPU 干预即可将数据高效地编程到关联的地址。使用表模式可以解析地址和数据表，只需在一次块传输中配置外设存储器映射寄存器。

DMASRCWDTH 位字段应设置为“3”（64 位模式），DMADSTWDTH 位字段应设置为“2”（32 位模式）。DMASRCINCR 位字段可以设置为 10b 以递减传输后的源地址，设置为 11b 以递增源地址。DMASZ 设置为表示表中的条目数，对于块传输模式，DMATM 应设置为“01”。DMADAx 和 DMADSTINCR 在表模式中被忽略，可以视为“无关”值。

DMASAx 寄存器需要用表的起始地址进行编程，该地址需要与 64 位数据对齐（即 DMASAx[2:0] = "000"）。存储在表中的地址需要位于 64 位数据的低位字上 (ADDR[2:0] = "000")，而数据需要位于 64 位数据的高位字上 (ADDR[2:0] = "100")。表 4-5 是存储器中与 DMA 表模式兼容的表示例。

表 4-5. 与 DMA 表模式兼容的增量表示例

表地址	表数据
0x0000	地址 0
0x0004	数据 0
0x0008	地址 1
0x000C	数据 1

4.2.5 初始化 DMA 传输

每个 DMA 通道都能根据它的触发源使用 DMATSEL 进行独立配置。只有当 DMACTLx.DMAEN 位为 0 时，才应修改 DMATSEL 位；否则，可能会发生不可预知的 DMA 触发。

有关可用触发的列表及其对应的 DMATSEL 值，请参阅特定于器件的数据表。

当选择触发时，必须确保触发还没有发生，或传输将不会发生。

DMA 通道可在另一个通道上的活动完成时在内部触发，以支持级联。当 DMA 通道的 DMASZ 计数器达到零时，活动完成。对于无需中断或事件配置即可检索、传输和/或错误检查数据的应用，这非常有用。

设置 DMATINT 位，以便根据 DMATSEL 触发源在内部触发下一个 DMA 通道。一旦发生 DMATSEL 触发，下一个 DMA 通道便开始自动执行。

例如，如果 UART 数据通过 DMA 通道 0 接收并发送到 SRAM，并且 DMATSEL 设置为 UART RX，则当 UART 完成接收数据时，可在内部触发 DMA 通道 1。如果 DMA 通道 1 配置为将数据从 SRAM 传输到 CRC，则 DMA 传输将在接收到 UART 数据后触发。在这种情况下，DMA 通道从通道 0 级联到通道 1。

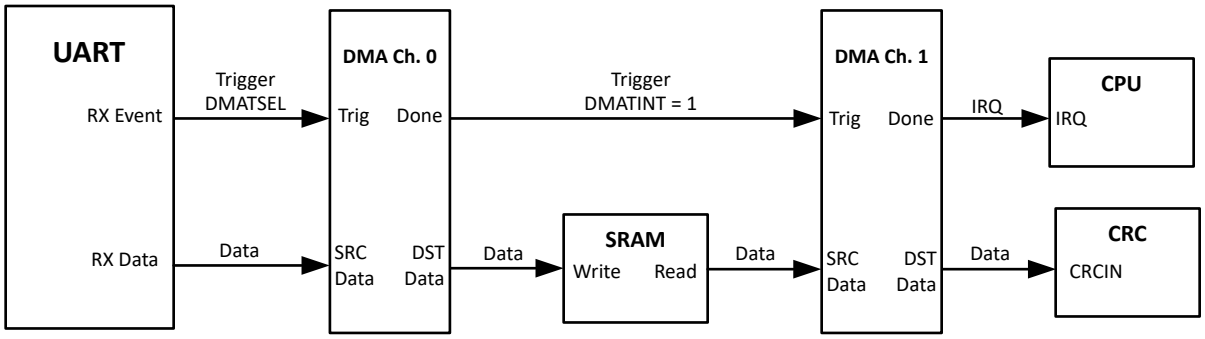


图 4-3. DMA 级联通道

4.2.6 停止 DMA 传输

可以通过清除 DMAEN 位来停止正在进行的 DMA 块传输。正在进行的传输周期完成后，DMA 将停止，所有通道寄存器将保持当前状态。再次设置 DMAEN 位后，可以按照最初的配置继续进行块传输。

备注

正在进行的单次传输不能被中断。

4.2.7 通道的优先级

默认的 DMA 通道优先级是 DMA0 到 DMA15。如果两个或三个触发同时发生或者挂起，拥有最高优先级的通道将会首先完成传输（单一或块传输），然后是第二优先级的通道，最后是第三优先级的通道。如果一个较高优先级的通道被触发，进行中的传输中将不会被暂停。一直到进行中的传输完成后较高优先级的传输才开始。

DMA 通道的优先级由 ROUNDROBIN 位配置。当 ROUNDROBIN 位被置位时，完成一个传输的通道的优先级会变为最低。这些通道的优先级的顺序总保持相同，例如，对于三个通道而言为 DMA0-DMA1-DMA2。当 ROUNDROBIN 位被清零时，通道的优先级回到默认优先级。

表 4-6. 轮询 DMA 优先级示例

当前 DMA 优先级	传输发生	新的 DMA 优先级
DMA0 - DMA1 - DMA2	DMA1	DMA2 - DMA0 - DMA1
DMA2 - DMA0 - DMA1	DMA2	DMA0 - DMA1 - DMA2
DMA2 - DMA0 - DMA1	DMA0	DMA1 - DMA2 - DMA0

4.2.8 突发块模式

DMA 模块支持突发块模式，用于在可配置的传输数量后暂停活动通道，以便为其他待处理通道提供服务。通过将 DMAPRIO.BURSTSZ 设置为 8、16、32 或无限次传输，可配置突发块大小。如果在突发块之后有一个更高优先级的通道处于挂起状态，则 DMA 将执行更高优先级的通道，并在更高优先级的通道完成后在挂起的通道上恢复。如果没有其他通道挂起，优先级逻辑会将控制权重新分配给下一次突发的块传输。

4.2.9 DMA 与系统中断结合使用

系统中断服务子程序将会被 DMA 传输打断。如果中断服务例程或其他例程必须在没有中断的情况下运行，则必须在执行该例程前禁用 DMA 控制器。

4.2.10 DMA 控制器中断

每个 DMA 通道都有自己的 RIS 标志。当相应的 DMASZx 寄存器计数为零时，在任何模式下都会设置每个 RIS 标志。如果设置了相应的 MASK 和 RIS 位，则会生成中断请求。

所有 RIS 标志都经过优先级排序（其中 DMA0 的优先级最高）并结合在一起，以获得单个中断矢量。具有最高优先级的中断会在 IIDX 寄存器中生成一个数字。可评估该数字或将其添加到程序计数器 (PC)，以便自动进入相应的软件程序。

对 IIDX 寄存器的任何访问（读取或写入）都会自动复位最高挂起中断标志。如果另一个中断标志被置位，则另一个中断将会在最初的中断服务结束后立即产生。例如，假设 DMA0 有最高的优先级。如果在中断服务例程访问 IIDX 寄存器时设置了 DMA0-RIS 和 DMA2-RIS 标志，则 DMA0-RIS 会自动复位。执行中断服务例程后，DMA2-RIS 会生成另一个中断。

4.2.11 DMA 触发事件状态

DMA 控制器支持专用 DMA 事件。有关 DMA 事件触发协议的详细信息，请参阅节 7.2.2。具体思路是，DMA 可以向事件触发外设通知已分配 DMA 通道的状态。这将允许触发外设在重复传输完成后自行发出中断，而不是 DMA 发出中断事件。优点是 DMA 中断服务例程不需要跟踪通道的已分配功能。因此，DMA 触发外设中断服务例程将处理 DMA 传输的完成。

状态将反映 DMASZx 寄存器的值。如果最后一次 DMA 传输导致大小递减到零，则 DMA 将返回状态 1，表示传输结束。否则状态将为 0。

此外，DMA 模块可以向 CPU 生成提前中断请求，以指示传输将在可配置的传输次数（1、2、4、8、32、64、半 DMASZ）内完成。

通过将 DMAPREIRQ 设置为所需的传输次数来启用早期 IRQ 事件。当 DMA 达到传输次数时，将设置相应 DMA 通道的 PREIRQ 中断。

早期 DMA 中断生成可用于：

- 减少时序关键型应用中的中断延迟，在此类应用中，让 DMA 在 DMA 传输完成之前抢先生成 IRQ 很有用
- 在安排 CPU 完成其他任务时，用作“进度通知”
- 传输神经网络层的权重，并通知 CPU 完成对 IP 的软件配置写入
- 实现乒乓缓冲器（通过将 DMAPREIRQ 设置为一半）

备注

此特性仅在支持重复的通道上可用。

4.2.12 DMA 工作模式支持

DMA 支持在 RUN 模式以及 SLEEP、STOP 和 STANDBY 低功耗模式下触发传输。更多详细信息，请参阅以下部分。

4.2.12.1 在 RUN 模式下传输

在 RUN 模式下，系统可以完全正常运行。CPU 以及所有其他外设和资源都可用，因此在 RUN 模式下对 DMA 功能没有限制。

4.2.12.2 在 SLEEP 模式下传输

在 SLEEP 模式下仅暂停 CPU。所有其他外设和资源都与在 RUN 模式下一样可用，因此在 SLEEP 模式下对 DMA 功能没有限制。所有可以在 RUN 模式下触发 DMA 传输的外设也可以在 SLEEP 模式下触发 DMA 传输。

4.2.12.3 在 STOP 模式下传输

在 STOP 模式下会暂停 CPU，并且 ULPCLK 工作频率限制为 4MHz。适用时，PD1 外设将被禁用并处于保留模式。只有 PD0 外设正常工作，因此只有 PD0 外设能够在 STOP 模式下触发 DMA 传输。DMA 本身位于 PD1 中，因此在 STOP 模式下处于保留状态。事件管理器将检测到 DMA 触发事件并请求 PMU 进入“暂停 STOP”状态。更多有关此状态的信息，请参阅节 2.1.2.7。暂停 STOP 模式后，DMA 可以完全正常工作，能够处理待处理的 DMA 触发请求。一旦 DMA 传输完成，DMA 将确认待处理的触发事件，并且事件子系统会从 PMU 移除功耗模式请求。如果 PMU 没有其他待处理的请求，则 SoC 将转换回正常的 STOP 模式。

4.2.12.4 在 STANDBY 模式下传输

在 STANDBY 模式下会暂停 CPU，并且 ULPCLK 工作频率限制为 32kHz。适用时，PD1 外设将被禁用并处于保留模式。只有 PD0 外设正常工作，因此只有 PD0 外设能够在 STANDBY 模式下触发 DMA 传输。DMA 本身位于

PD1 中，因此在 STANDBY 模式下处于保留状态。事件管理器将检测到 DMA 触发事件并请求 PMU 进入“暂停 STANDBY”状态。更多有关此状态的信息，请参阅节 2.1.2.7。暂停 STANDBY 模式后，DMA 可以完全正常工作，能够处理待处理的 DMA 触发请求。一旦 DMA 传输完成，DMA 将确认待处理的触发事件，并且事件子系统会从 PMU 移除功耗模式请求。如果 PMU 没有其他待处理的请求，则 SoC 将转换回正常的 STANDBY 模式。

4.2.13 DMA 地址和数据错误

DMA 本身能够标记地址或数据错误。源地址或目标地址错误可能是由于访问受保护或不存在的存储器范围所致。如果发生地址错误，中断索引 IIDX[j].STAT 会标记 DMA 地址错误 (11h)。使用 ADDERR 位可以屏蔽、设置和清除地址错误中断。

备注

DMA 本身不执行范围检查。如果 DMA 传输在受保护的存储器范围内发生，则与受保护或不存在的存储器范围重叠的 DMA 事务的每个字节的目标数据将报告零 (0h)。

如果存在 ECC 或奇偶校验错误，则 SRAM 或闪存中可能会发生数据错误。如果发生数据错误，中断索引 IIDX[j].STAT 会标记 DMA 数据错误 (12h)。使用 DATERR 位可以屏蔽、设置和清除数据错误中断。

4.2.14 中断和事件支持

DMA 模块包含一个事件发布者和两个事件订阅者。

- 一个事件发布者 (CPU_INT) 通过静态事件路由来管理到 CPU 子系统的 DMA 中断请求 (IRQ)。
- 第二个和第三个事件 (GEN_EVENT) 用于通过通用路由设置通用事件发布者和订阅者。

表 4-7 中汇总了 DMA 事件。

表 4-7. DMA 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	DMA	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 TIMx 到 CPU 的中断路由
通用发布者事件	发布者	DMA	其他外设	通用路由	GEN_EVENT 和 FPUB_1 寄存器	从 DMA 到其他外设的可配置中断路由
通用订阅者事件	订阅者	其他外设	DMA	通用路由	FSUB_0	从其他外设到 DMA 的可配置中断路由
通用订阅者事件	订阅者	其他外设	DMA	通用路由	FSUB_1	从其他外设到 DMA 的可配置中断路由

4.3 DMA 寄存器

表 4-8 列出了 DMA 寄存器的存储器映射寄存器。表 4-8 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 4-8. DMA 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者端口 0		转到
404h	FSUB_1	订阅者端口 1		转到
444h	FPUB_1	发布者端口 0		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT	转到
1058h	IMASK	中断屏蔽	GEN_EVENT	转到
1060h	RIS	原始中断状态	GEN_EVENT	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT	转到
1070h	ISET	中断设置	GEN_EVENT	转到
1078h	ICLR	中断清除	GEN_EVENT	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	DMAPRIO	DMA 通道优先级控制		转到
1110h + 公式	DMATCTL[j]	DMA 触发选择		转到
1200h + 公式	DMACTL[j]	DMA 通道控制		转到
1204h + 公式	DMASA[j]	DMA 通道源地址		转到
1208h + 公式	DMADA[j]	DMA 通道目标地址		转到
120Ch + 公式	DMASZ[j]	DMA 通道大小		转到

复杂的位访问类型经过编码可适应小型表单元。表 4-9 显示了适用于此部分中访问类型的代码。

表 4-9. DMA 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值
寄存器数组变量		

表 4-9. DMA 访问类型代码 (continued)

访问类型	代码	说明
i、j、k、l、m、n		当这些变量用于寄存器名称、偏移或地址时，它们指的是寄存器数组的值，其中寄存器是一组重复寄存器的一部分。寄存器组构成分层结构，数组用公式表示。
y		当该变量用于寄存器名称、偏移或地址时，它指的是寄存器数组的值。

4.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 4-4 展示了 FSUB_0，表 4-10 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 4-4. FSUB_0

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

表 4-10. FSUB_0 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	CHANID	R/W	0h	0 = 已断开连接。 1-255 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 FFh = 请参阅器件数据表，因为实际允许的最大值可能小于 255。

4.3.2 FSUB_1（偏移 = 404h）[复位 = 00000000h]

图 4-5 展示了 FSUB_1，表 4-11 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 4-5. FSUB_1

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

表 4-11. FSUB_1 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	CHANID	R/W	0h	0 = 已断开连接。 1-255 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 FFh = 请参阅器件数据表，因为实际允许的最大值可能小于 255。

4.3.3 FPUB_1（偏移 = 444h）[复位 = 00000000h]

图 4-6 展示了 FPUB_1，表 4-12 中对此进行了介绍。

返回到汇总表。

发布者端口

图 4-6. FPUB_1

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

表 4-12. FPUB_1 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	CHANID	R/W	0h	0 = 已断开连接。 1-255 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 FFh = 请参阅器件数据表，因为实际允许的最大值可能小于 255。

4.3.4 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 4-7 展示了 PDBGCTL，表 4-13 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 4-7. PDBGCTL

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

表 4-13. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设将阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

4.3.5 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 4-8 展示了 IIDX，表 4-14 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、...IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS [RIS] 和 MIS [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-8. IIDX

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

表 4-14. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 没有位置位意味着没有挂起的中断请求 01h = DMA 通道 0 大小计数器达到零 (DMASZ=0)。 02h = DMA 通道 1 大小计数器达到零 (DMASZ=0)。 03h = DMA 通道 2 大小计数器达到零 (DMASZ=0)。 04h = DMA 通道 3 大小计数器达到零 (DMASZ=0)。 05h = DMA 通道 4 大小计数器达到零 (DMASZ=0)。 06h = DMA 通道 5 大小计数器达到零 (DMASZ=0)。 07h = DMA 通道 6 大小计数器达到零 (DMASZ=0)。 08h = DMA 通道 7 大小计数器达到零 (DMASZ=0)。 09h = DMA 通道 8 大小计数器达到零 (DMASZ=0)。 0Ah = DMA 通道 9 大小计数器达到零 (DMASZ=0)。 0Bh = DMA 通道 10 大小计数器达到零 (DMASZ=0)。 0Ch = DMA 通道 11 大小计数器达到零 (DMASZ=0)。 0Dh = DMA 通道 12 大小计数器达到零 (DMASZ=0)。 0Eh = DMA 通道 13 大小计数器达到零 (DMASZ=0)。 0Fh = DMA 通道 14 大小计数器达到零 (DMASZ=0)。 10h = DMA 通道 15 大小计数器达到零 (DMASZ=0)。 11h = DMA 通道 0 的 PRE-IRQ 事件。 12h = DMA 通道 1 的 PRE-IRQ 事件。 13h = DMA 通道 2 的 PRE-IRQ 事件。 14h = DMA 通道 3 的 PRE-IRQ 事件。 15h = DMA 通道 4 的 PRE-IRQ 事件。 16h = DMA 通道 5 的 PRE-IRQ 事件。 17h = DMA 通道 6 的 PRE-IRQ 事件。 18h = DMA 通道 7 的 PRE-IRQ 事件。 19h = DMA 地址错误，SRC 地址不可访问。 1Ah = DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。

4.3.6 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 4-9 展示了 IMASK，表 4-15 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX [IIDX] 以及 MIS [MIS] 中。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-9. IMASK

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R/W-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 4-15. IMASK 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R/W	0h	
25	DATAERR	R/W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R/W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R/W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R/W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R/W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R/W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R/W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R/W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-15. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	R/W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	R/W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R/W	0h	DMA 通道 15 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
14	DMACH14	R/W	0h	DMA 通道 14 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
13	DMACH13	R/W	0h	DMA 通道 13 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
12	DMACH12	R/W	0h	DMA 通道 12 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
11	DMACH11	R/W	0h	DMA 通道 11 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
10	DMACH10	R/W	0h	DMA 通道 10 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
9	DMACH9	R/W	0h	DMA 通道 9 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
8	DMACH8	R/W	0h	DMA 通道 8 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
7	DMACH7	R/W	0h	DMA 通道 7 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
6	DMACH6	R/W	0h	DMA 通道 6 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
5	DMACH5	R/W	0h	DMA 通道 5 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
4	DMACH4	R/W	0h	DMA 通道 4 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
3	DMACH3	R/W	0h	DMA 通道 3 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
2	DMACH2	R/W	0h	DMA 通道 2 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
1	DMACH1	R/W	0h	DMA 通道 1 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-15. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	R/W	0h	DMA 通道 0 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

4.3.7 RIS（偏移 = 1030h）[复位 = 00000000h]

图 4-10 展示了 RIS，表 4-16 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK [IMASK] 位未启用，也可以通过向 ICLR [ICLR] 寄存器位写入 1 来清除此寄存器中设置的标志。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-10. RIS

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 4-16. RIS 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	DATAERR	R	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-16. RIS 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	R	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	R	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
14	DMACH14	R	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
13	DMACH13	R	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
12	DMACH12	R	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
11	DMACH11	R	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
10	DMACH10	R	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
9	DMACH9	R	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
8	DMACH8	R	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
7	DMACH7	R	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
6	DMACH6	R	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
5	DMACH5	R	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
4	DMACH4	R	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
3	DMACH3	R	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
2	DMACH2	R	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
1	DMACH1	R	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断

表 4-16. RIS 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	R	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断

4.3.8 MIS（偏移 = 1038h）[复位 = 00000000h]

图 4-11 展示了 MIS，表 4-17 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK [IMASK] 和 RIS [RIS] 寄存器的与运算结果。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-11. MIS

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 4-17. MIS 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	DATAERR	R	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
17	PREIRQCH1	R	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-17. MIS 字段说明 (continued)

位	字段	类型	复位	说明
16	PREIRQCH0	R	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
14	DMACH14	R	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
13	DMACH13	R	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
12	DMACH12	R	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
11	DMACH11	R	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
10	DMACH10	R	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
9	DMACH9	R	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
8	DMACH8	R	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
7	DMACH7	R	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
6	DMACH6	R	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
5	DMACH5	R	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
4	DMACH4	R	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
3	DMACH3	R	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
2	DMACH2	R	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
1	DMACH1	R	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
0	DMACH0	R	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断

4.3.9 ISET（偏移 = 1040h）[复位 = 00000000h]

图 4-12 展示了 ISET，表 4-18 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此也会设置相关的 RIS [RIS] 位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS [MIS] 位。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-12. ISET

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
W-0h						W-0h	W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 4-18. ISET 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	W	0h	
25	DATAERR	W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-18. ISET 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
14	DMACH14	W	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
13	DMACH13	W	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
12	DMACH12	W	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
11	DMACH11	W	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
10	DMACH10	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
9	DMACH9	W	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
8	DMACH8	W	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
7	DMACH7	W	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
6	DMACH6	W	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
5	DMACH5	W	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
4	DMACH4	W	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
3	DMACH3	W	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
2	DMACH2	W	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
1	DMACH1	W	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断

表 4-18. ISET 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断

4.3.10 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 4-13 展示了 ICLR，表 4-19 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-13. ICLR

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
W-0h						W-0h	W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 4-19. ICLR 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	W	0h	
25	DATAERR	W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
17	PREIRQCH1	W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-19. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
16	PREIRQCH0	W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	W	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
14	DMACH14	W	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
13	DMACH13	W	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
12	DMACH12	W	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
11	DMACH11	W	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
10	DMACH10	W	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
9	DMACH9	W	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
8	DMACH8	W	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
7	DMACH7	W	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
6	DMACH6	W	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
5	DMACH5	W	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
4	DMACH4	W	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
3	DMACH3	W	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
2	DMACH2	W	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
1	DMACH1	W	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
0	DMACH0	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断

4.3.11 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 4-14 展示了 IIDX，表 4-20 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、...IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS [RIS] 和 MIS [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-14. IIDX

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

表 4-20. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 没有位置位意味着没有挂起的中断请求 01h = DMA 通道 0 大小计数器达到零 (DMASZ=0)。 02h = DMA 通道 1 大小计数器达到零 (DMASZ=0)。 03h = DMA 通道 2 大小计数器达到零 (DMASZ=0)。 04h = DMA 通道 3 大小计数器达到零 (DMASZ=0)。 05h = DMA 通道 4 大小计数器达到零 (DMASZ=0)。 06h = DMA 通道 5 大小计数器达到零 (DMASZ=0)。 07h = DMA 通道 6 大小计数器达到零 (DMASZ=0)。 08h = DMA 通道 7 大小计数器达到零 (DMASZ=0)。 09h = DMA 通道 8 大小计数器达到零 (DMASZ=0)。 0Ah = DMA 通道 9 大小计数器达到零 (DMASZ=0)。 0Bh = DMA 通道 10 大小计数器达到零 (DMASZ=0)。 0Ch = DMA 通道 11 大小计数器达到零 (DMASZ=0)。 0Dh = DMA 通道 12 大小计数器达到零 (DMASZ=0)。 0Eh = DMA 通道 13 大小计数器达到零 (DMASZ=0)。 0Fh = DMA 通道 14 大小计数器达到零 (DMASZ=0)。 10h = DMA 通道 15 大小计数器达到零 (DMASZ=0)。 11h = DMA 通道 0 的 PRE-IRQ 事件。 12h = DMA 通道 1 的 PRE-IRQ 事件。 13h = DMA 通道 2 的 PRE-IRQ 事件。 14h = DMA 通道 3 的 PRE-IRQ 事件。 15h = DMA 通道 4 的 PRE-IRQ 事件。 16h = DMA 通道 5 的 PRE-IRQ 事件。 17h = DMA 通道 6 的 PRE-IRQ 事件。 18h = DMA 通道 7 的 PRE-IRQ 事件。 19h = DMA 地址错误，SRC 地址不可访问。 1Ah = DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。

4.3.12 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 4-15 展示了 IMASK，表 4-21 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX [IIDX] 以及 MIS [MIS] 中。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-15. IMASK

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R/W-0h						R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 4-21. IMASK 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R/W	0h	
25	DATAERR	R/W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R/W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R/W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R/W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R/W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R/W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R/W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R/W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-21. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	R/W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	R/W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R/W	0h	DMA 通道 15 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
14	DMACH14	R/W	0h	DMA 通道 14 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
13	DMACH13	R/W	0h	DMA 通道 13 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
12	DMACH12	R/W	0h	DMA 通道 12 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
11	DMACH11	R/W	0h	DMA 通道 11 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
10	DMACH10	R/W	0h	DMA 通道 10 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
9	DMACH9	R/W	0h	DMA 通道 9 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
8	DMACH8	R/W	0h	DMA 通道 8 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
7	DMACH7	R/W	0h	DMA 通道 7 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
6	DMACH6	R/W	0h	DMA 通道 6 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
5	DMACH5	R/W	0h	DMA 通道 5 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
4	DMACH4	R/W	0h	DMA 通道 4 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
3	DMACH3	R/W	0h	DMA 通道 3 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
2	DMACH2	R/W	0h	DMA 通道 2 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
1	DMACH1	R/W	0h	DMA 通道 1 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-21. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	R/W	0h	DMA 通道 0 中断信号。大小计数器达到零 (DMASZ=0)。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

4.3.13 RIS（偏移 = 1060h）[复位 = 00000000h]

图 4-16 展示了 RIS，表 4-22 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK [IMASK] 位未启用，也可以通过向 ICLR [ICLR] 寄存器位写入 1 来清除此寄存器中设置的标志。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-16. RIS

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 4-22. RIS 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	DATAERR	R	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-22. RIS 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	R	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	R	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
14	DMACH14	R	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
13	DMACH13	R	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
12	DMACH12	R	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
11	DMACH11	R	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
10	DMACH10	R	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
9	DMACH9	R	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
8	DMACH8	R	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
7	DMACH7	R	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
6	DMACH6	R	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
5	DMACH5	R	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
4	DMACH4	R	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
3	DMACH3	R	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
2	DMACH2	R	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断
1	DMACH1	R	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断

表 4-22. RIS 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	R	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 未发生中断 1h = 已发生中断

4.3.14 MIS（偏移 = 1068h）[复位 = 00000000h]

图 4-17 展示了 MIS，表 4-23 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK [IMASK] 和 RIS [RIS] 寄存器的与运算结果。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-17. MIS

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 4-23. MIS 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	DATAERR	R	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	R	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	R	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	R	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	R	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	R	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	R	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	R	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
17	PREIRQCH1	R	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-23. MIS 字段说明 (continued)

位	字段	类型	复位	说明
16	PREIRQCH0	R	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	R	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
14	DMACH14	R	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
13	DMACH13	R	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
12	DMACH12	R	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
11	DMACH11	R	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
10	DMACH10	R	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
9	DMACH9	R	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
8	DMACH8	R	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
7	DMACH7	R	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
6	DMACH6	R	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
5	DMACH5	R	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
4	DMACH4	R	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
3	DMACH3	R	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
2	DMACH2	R	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
1	DMACH1	R	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断
0	DMACH0	R	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 中断未发生或被屏蔽掉 1h = 已发生中断

4.3.15 ISET（偏移 = 1070h）[复位 = 00000000h]

图 4-18 展示了 ISET，表 4-24 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此也会设置相关的 RIS [RIS] 位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS [MIS] 位。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-18. ISET

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
W-0h						W-0h	W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 4-24. ISET 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	W	0h	
25	DATAERR	W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-24. ISET 字段说明 (continued)

位	字段	类型	复位	说明
17	PREIRQCH1	W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
16	PREIRQCH0	W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
14	DMACH14	W	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
13	DMACH13	W	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
12	DMACH12	W	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
11	DMACH11	W	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
10	DMACH10	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
9	DMACH9	W	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
8	DMACH8	W	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
7	DMACH7	W	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
6	DMACH6	W	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
5	DMACH5	W	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
4	DMACH4	W	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
3	DMACH3	W	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
2	DMACH2	W	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断
1	DMACH1	W	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断

表 4-24. ISET 字段说明 (continued)

位	字段	类型	复位	说明
0	DMACH0	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 设置中断

4.3.16 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 4-19 展示了 ICLR，表 4-25 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。
注意：DMACH 的编号取决于器件。请参阅具体器件的数据表，了解实现的通道编号。

图 4-19. ICLR

31	30	29	28	27	26	25	24
RESERVED						DATAERR	ADDRERR
W-0h						W-0h	W-0h
23	22	21	20	19	18	17	16
PREIRQCH7	PREIRQCH6	PREIRQCH5	PREIRQCH4	PREIRQCH3	PREIRQCH2	PREIRQCH1	PREIRQCH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 4-25. ICLR 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	W	0h	
25	DATAERR	W	0h	DMA 数据错误，SRC 数据可能已损坏（PAR 或 ECC 错误）。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
24	ADDRERR	W	0h	DMA 地址错误，SRC 地址不可访问。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
23	PREIRQCH7	W	0h	通道 7 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
22	PREIRQCH6	W	0h	通道 6 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
21	PREIRQCH5	W	0h	通道 5 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
20	PREIRQCH4	W	0h	通道 4 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
19	PREIRQCH3	W	0h	通道 3 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
18	PREIRQCH2	W	0h	通道 2 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
17	PREIRQCH1	W	0h	通道 1 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位

表 4-25. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
16	PREIRQCH0	W	0h	通道 0 的预 IRQ。大小计数器达到预 IRQ 阈值。 0h = 将中断屏蔽位清零 1h = 将中断屏蔽位置位
15	DMACH15	W	0h	DMA 通道 15 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
14	DMACH14	W	0h	DMA 通道 14 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
13	DMACH13	W	0h	DMA 通道 13 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
12	DMACH12	W	0h	DMA 通道 12 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
11	DMACH11	W	0h	DMA 通道 11 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
10	DMACH10	W	0h	DMA 通道 10 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
9	DMACH9	W	0h	DMA 通道 9 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
8	DMACH8	W	0h	DMA 通道 8 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
7	DMACH7	W	0h	DMA 通道 7 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
6	DMACH6	W	0h	DMA 通道 6 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
5	DMACH5	W	0h	DMA 通道 5 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
4	DMACH4	W	0h	DMA 通道 4 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
3	DMACH3	W	0h	DMA 通道 3 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
2	DMACH2	W	0h	DMA 通道 2 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
1	DMACH1	W	0h	DMA 通道 1 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断
0	DMACH0	W	0h	DMA 通道 0 中断信号，指示大小计数器达到零 (DMASZ=0)。 0h = 写入 0 不产生影响 1h = 清除中断

4.3.17 EVT_MODE（偏移 = 10E0h）[复位 = 00000009h]

图 4-20 展示了 EVT_MODE，表 4-26 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 4-20. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED				EVT1_CFG		INT0_CFG	
R/W-				R-2h		R-1h	

表 4-26. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	EVT1_CFG	R	2h	通用事件 GEN_EVENT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	中断事件 CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

4.3.18 DESC（偏移 = 10FCh）[复位 = 2511F000h]

图 4-21 展示了 DESC，表 4-27 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 4-21. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-2511h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-Fh				R-				R-0h				R-0h			

表 4-27. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	2511h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	Fh	DMA 的功能集：DMA 通道编号减一（例如 0 -> 1 通道、2 -> 3 通道、15 -> 16 通道）。 0h = 最小值（1 通道） Fh = 尽可能高的值（16 通道）
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

4.3.19 DMAPRIO（偏移 = 1100h）[复位 = 00000000h]

图 4-22 展示了 DMAPRIO，表 4-28 中对此进行了介绍。

返回到汇总表。

DMA 通道优先级控制

图 4-22. DMAPRIO

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

表 4-28. DMAPRIO 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R/W	0h	
17-16	BURSTSZ	R/W	0h	定义在重新评估优先级之前块传输的突发大小 0h = 无突发大小，整个块传输一次性完成，无中断 1h = 突发大小为 8，在 8 次传输后块传输被中断并重新评估优先级 2h = 突发大小为 16，在 16 次传输后块传输被中断并重新评估优先级 3h = 突发大小为 32，在 32 次传输后块传输被中断并重新评估优先级
15-1	RESERVED	R/W	0h	
0	ROUNDROBIN	R/W	0h	轮循。该位启用轮循 DMA 通道优先级。 0h = 循环优先级被禁用，DMA 通道优先级被固定：DMA0-DMA1-DMA2-...-DMA16 1h = 循环优先级被启用，DMA 通道优先级随每次传输而变化

4.3.20 DMATCTL[j]（偏移 = 1110h + 公式）[复位 = 00000000h]

图 4-23 展示了 DMATCTL[j]，表 4-29 中对此进行了介绍。

返回到汇总表。

DMA 触发控制

偏移 = 1110h + (j * 4h)；其中 j = 0h 至 Fh

图 4-23. DMATCTL[j]

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

表 4-29. DMATCTL[j] 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7	DMATINT	R/W	0h	通过内部通道进行 DMA 触发 0h = DMATSEL 会将外部触发选择定义为传输触发。 1h = DMATSEL 会将内部通道定义为传输触发选择。0 -> 通道 0 完成、1 -> 通道 1 完成、...
6	RESERVED	R/W	0h	
5-0	DMATSEL	R/W	0h	DMA 触发选择 注：请参阅器件的数据表来查看具体的触发映射。 00h = 软件触发请求 3Fh = 尽可能高的值

4.3.21 DMACTL[j]（偏移 = 1200h + 公式）[复位 = 00000000h]

图 4-24 展示了 DMACTL[j]，表 4-30 中对此进行了介绍。

返回到汇总表。

DMA 通道控制

偏移 = 1200h + (j * 10h)；其中 j = 0h 至 Fh

图 4-24. DMACTL[j]

31	30	29	28	27	26	25	24
RESERVED		DMATM		RESERVED		DMAEM	
R/W-		R/W-0h		R/W-		R/W-0h	
23	22	21	20	19	18	17	16
DMADSTINCR				DMASRCINCR			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED		DMADSTWDTH		RESERVED		DMASRCWDTH	
R/W-		R/W-0h		R/W-		R/W-0h	
7	6	5	4	3	2	1	0
RESERVED	DMAPREIRQ			RESERVED		DMAEN	DMAREQ
R/W-	R/W-0h			R/W-		R/W-0h	R/W-0h

表 4-30. DMACTL[j] 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29-28	DMATM	R/W	0h	DMA 传输模式寄存器 注：重复单个 (2h) 和重复块 (3h) 传输仅在完整通道配置中可用。请参阅具体器件的数据表，了解哪个通道编号具有完整或基本功能。在基本通道配置中，只能设置单个 (0h) 和块 (1h) 传输的值。 0h = 单个传输。每次传输都需要一次新的触发。当 DMASZ 向下计数至零时，会生成一个事件并且 DMAEN 被清除。 1h = 块传输。每次触发后，传输 DMASZ 中定义的完整块。传输完成后，会生成一个事件并且 DMAEN 被清除。 2h = 重复单个传输。每次传输都需要一次新的触发。当 DMASZ 向下计数至零时，会生成一个事件。在最后一次传输之后，DMASA、DMADA、DAMSZ 寄存器恢复到其初始值并且 DMAEN 保持启用状态。 3h = 重复块传输。每次触发后，传输 DMASZ 中定义的完整块。在最后一次传输之后，DMASA、DMADA、DAMSZ 寄存器恢复到其初始值并且 DMAEN 保持启用状态。
27-26	RESERVED	R/W	0h	
25-24	DMAEM	R/W	0h	DMA 扩展模式 注：扩展传输模式仅在完整通道配置中可用。请参阅具体器件的数据表，了解哪个通道编号具有完整或基本功能。在基本通道配置中，该寄存器是只读寄存器，读数为 0x0。 0h = 正常模式与从 SRC 到 DST 的传输相关 2h = 填充模式会将 SA 寄存器内容作为数据复制到 DA 3h = 表模式将从 SA 读取地址和数据值并将数据写入地址

表 4-30. DMACTL[j] 字段说明 (continued)

位	字段	类型	复位	说明
23-20	DMADSTINCR	R/W	0h	DMA 的目标增量。每次传输时，该位选择目标地址 DMADA 的自动递增或递减。DMADA 的变化量基于 DMADSTWDTH 中的定义。例如，字传输的增量 1 (+1) 将使 DMADA 递增 4。 0h = 地址不变 (+0) 2h = 递减 1 (-1 * DMADSTWDTH) 3h = 递增 1 (+1 * DMADSTWDTH) 8h = 跨步大小为 2 (+2 * DMADSTWDTH) 9h = 跨步大小 3 (+3 * DMADSTWDTH) Ah = 跨步大小为 4 (+4 * DMADSTWDTH) Bh = 跨步大小为 5 (+5 * DMADSTWDTH) Ch = 跨步大小为 6 (+6 * DMADSTWDTH) Dh = 跨步大小为 7 (+7 * DMADSTWDTH) Eh = 跨步大小为 8 (+8 * DMADSTWDTH) Fh = 跨步大小为 9 (+9 * DMADSTWDTH)
19-16	DMASRCINCR	R/W	0h	DMA 源增量。每次传输时，该位选择源地址 DMASA 的自动递增或递减。DMASA 的变化量基于 DMASRCWDTH 中的定义。例如，字传输的增量 1 (+1) 将使 DMASA 递增 4。 0h = 地址不变 (+0) 2h = 递减 1 (-1 * DMASRCWDTH) 3h = 递增 1 (+1 * DMASRCWDTH) 8h = 跨步大小为 2 (+2 * DMASRCWDTH) 9h = 跨步大小 3 (+3 * DMASRCWDTH) Ah = 跨步大小为 4 (+4 * DMASRCWDTH) Bh = 跨步大小为 5 (+5 * DMASRCWDTH) Ch = 跨步大小为 6 (+6 * DMASRCWDTH) Dh = 跨步大小为 7 (+7 * DMASRCWDTH) Eh = 跨步大小为 8 (+8 * DMASRCWDTH) Fh = 跨步大小为 9 (+9 * DMASRCWDTH)
15-14	RESERVED	R/W	0h	
13-12	DMADSTWDTH	R/W	0h	DMA 目标宽度。该位选择字节、半字、字或长字作为目标。 0h = 目标数据宽度为字节 (8 位) 1h = 目标数据宽度为半字 (16 位) 2h = 目标数据宽度为字 (32 位) 3h = 目标数据宽度为长字 (64 位)
11-10	RESERVED	R/W	0h	
9-8	DMASRCWDTH	R/W	0h	DMA 源宽度。该位选择字节、半字、字或长字作为源数据宽度。 0h = 源数据宽度为字节 (8 位) 1h = 源数据宽度为半字 (16 位) 2h = 源数据宽度为字 (32 位) 3h = 源数据宽度为长字 (64 位)
7	RESERVED	R/W	0h	
6-4	DMAPREIRQ	R/W	0h	启用早期 IRQ 事件。这可以帮助软件更快地对 DMA 完成事件做出反应，或者允许在通道完成之前进行一些额外的配置。 注意：该寄存器仅在完整通道配置中可用。请参阅具体器件的数据表，了解哪个通道编号具有完整或基本功能。在基本配置中，该寄存器是只读值，读数始终为 0x0。 0h = 预 IRQ 事件被禁用。 1h = 当 DMASZ=1 时发出预 IRQ 事件 2h = 当 DMASZ=2 时发出预 IRQ 事件 3h = 当 DMASZ=4 时发出预 IRQ 事件 4h = 当 DMASZ=8 时发出预 IRQ 事件 5h = 当 DMASZ=32 时发出预 IRQ 事件 6h = 当 DMASZ=64 时发出预 IRQ 事件 7h = 当 DMASZ 达到原始传输大小的一半大小点时发出预 IRQ 事件
3-2	RESERVED	R/W	0h	

表 4-30. DMACTL[j] 字段说明 (continued)

位	字段	类型	复位	说明
1	DMAEN	R/W	0h	DMA 启用 0h = DMA 通道被禁用 1h = DMA 通道被启用
0	DMAREQ	R/W	0h	DMA 请求。软件控制的 DMA 启动。DMAREQ 自动复位。 0h = 默认读取值 1h = DMA 传输请求（启动 DMA）

4.3.22 DMASA[j]（偏移 = 1204h + 公式）[复位 = 00000000h]

图 4-25 展示了 DMASA[j]，表 4-31 中对此进行了介绍。

返回到汇总表。

DMA 通道源地址

偏移 = 1204h + (j * 10h)；其中 j = 0h 至 Fh

图 4-25. DMASA[j]

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

表 4-31. DMASA[j] 字段说明

位	字段	类型	复位	说明
31-0	ADDR	R/W	0h	DMA 通道源地址 0h = 最小值 FFFFFFFFh = 尽可能高的值

4.3.23 DMADA[j]（偏移 = 1208h + 公式）[复位 = 00000000h]

图 4-26 展示了 DMADA[j]，表 4-32 中对此进行了介绍。

返回到汇总表。

DMA 通道目标地址

偏移 = 1208h + (j * 10h)；其中 j = 0h 至 Fh

图 4-26. DMADA[j]

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

表 4-32. DMADA[j] 字段说明

位	字段	类型	复位	说明
31-0	ADDR	R/W	0h	DMA 通道目标地址 0h = 最小值 FFFFFFFFh = 尽可能高的值

4.3.24 DMASZ[j]（偏移 = 120Ch + 公式）[复位 = 00000000h]

图 4-27 展示了 DMASZ[j]，表 4-33 中对此进行了介绍。

返回到汇总表。

DMA 通道大小

偏移 = 120Ch + (j * 10h)；其中 j = 0h 至 Fh

图 4-27. DMASZ[j]

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

表 4-33. DMASZ[j] 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	尺寸	R/W	0h	DMA 通道大小（以传输次数表示） 0h = 最小值 FFFFh = 尽可能高的值

This page intentionally left blank.

数学加速器 (MATHACL) 是一组硬件加速的 32 位数学函数，用于提高系统计算吞吐量。在 CPU 上运行的软件通过写入 MATHACL 寄存器来调用 MATHACL。

5.1 概述

MATHACL 可减轻 CPU 执行的数学计算工作量，从而提高效率和 CoreMark 性能。MATHACL 中提供了以下硬件功能：

- 正弦/余弦 (SINCOS)
- 反正切 (ATAN2)
- 平方根 (SQRT)
- 除法 (DIV)
- 与 32 位结果相乘 (MPY32)
- 对 32 位结果进行平方 (SQUARE32)
- 与 64 位结果相乘 (MPY64)
- 对 64 位结果进行平方 (SQUARE64)
- 乘法累加 (MAC)
- 平方累加 (SAC)

5.2 数据格式

MATCHACL 函数中使用的操作数是 32 位定点数。操作数可以表示为无符号 32 位整数、有符号 32 位整数、无符号 32 位数和有符号 32 位数。

表 5-1 汇总了 MATHACL 中使用的数据格式。

表 5-1. MATHACL 数据格式

数据格式	数据类型	有符号格式	数据范围
无符号 32 位整数	Uint32_t	二进制补码	0 至 2^{32} - 1
有符号 32 位整数	int32_t	二进制补码	-2^{31} 至 2^{31} - 1
无符号 32 位数字	UQm.n	有符号幅度	0 至 $(2^{32})/2^{n}$
有符号 32 位数字	SQm.n	有符号幅度	$(-2^{31})/2^{n}$ 至 $(2^{31})/2^{n}$

5.2.1 无符号 32 位整数

无符号 32 位整数是一个介于 0 到 2^{32}-1 之间的 32 位正数，表示为类型 uint32_t。

例如，数字 123456 等于 0b 00000000 00000001 11100010 01000000 或 0x0001E240。

5.2.2 有符号 32 位整数

有符号 32 位整数是一个介于 -2^{31} 和 2^{31} – 1 之间的 32 位正数或负数，表达为 int32_t 类型。要转换十进制的有符号 32 位整数，请执行以下操作：

- 将该整数转换为无符号 32 位整数
- 将该数字反转

- 向最低有效位 (LSB) 添加 1

例如，数字 -123456 等于 0b 11111111 11111110 00011101 11000000 或 0xFFFE1DC0。

5.2.3 无符号 32 位数字

无符号 32 位数字包含使用 Q 数字格式的整数部分和小数部分。它表示为 UQ**m**.**n**，其中：

- **m** 是整数部分的位数
- **n** 是小数部分的位数

m 和 n 部分因操作数类型而异。例如，UQ16.16 格式的操作数的有符号数组成部分如表 5-2 中所示

表 5-2. 无符号数组成部分

说明	值	位数	位
整数部分	I	m = 16	31:16
小数部分	F	n = 16	15:0

请按以下方法生成无符号数，例如 UQ16.16 格式的 4.75：

1. 将整数部分 (I = 4) 表示为 m 位数字 (0x0004)
2. 将小数部分 (F = 0.75) 表示为 n 位数字 (0xC000)
3. 将生成的十六进制值表示为无符号值 (0x0100C000)

5.2.4 有符号 32 位数字

有符号 32 位数字包括一个使用 Q 数字格式的有符号幅度部分、整数部分和小数部分。它表示为 SQ**m**.**n**，其中：

- S 是符号位
- **m** 是整数部分的位数
- **n** 是小数部分的位数

m 和 n 部分因操作数类型而异。例如，SQ15.16 格式的操作数的有符号数组成部分如表 5-3 中所示。

表 5-3. 有符号数的组成部分

说明	值	位数	位
符号位	+ 或 -	S = 1	31
整数部分	I	m = 15	30:16
小数部分	F	n = 16	15:0

要生成有符号数，例如采用 SQ15.16 格式中的 -1.5：

1. 将整数部分 (I = 1) 表示为 m 位数字 (0x0001)
2. 将小数部分 (F = 0.5) 表示为 n 位数字 (0x8000)
3. 将生成的十六进制值表示为无符号值 (0x00018000)
4. 如果数字是有符号的，则取二进制补码 (0xFFFE7FFF)

5.3 基本操作

电源使能

启动 MATHACL 之前，请将该密钥写入 PWREN 寄存器，并设置 ENABLE 位来启动 MATHACL 电源。复位 MATHACL 之前，请将密钥写入 RSTCTL 寄存器，设置 RESETSTKYCLR 位和 RESETASSERT 位。

MATHACL 操作

所有 MATHACL 操作都遵循类似的代码序列来调用函数、开始计算操作并读取结果：

1. 执行单次写入操作向 CTL 寄存器的 FUNC、QVAL、OPTYPE、SFACTOR 和 NUMITER 字段进行写入。有关被调用的函数需要哪些字段，请参阅节 5.4。
2. 根据操作中所需的操作数来触发函数开始计算。
   a. 对于两个操作数函数，按顺序写入 OP1 和 OP2 来触发该函数开始计算。
   b. 对于单个操作数函数，写入 OP2 来触发函数开始计算。
3. 要读取计算结果，请读取 RES1 和 RES2 寄存器来检查新数据何时返回。或者，可以轮询 STATUS.BUSY 标志，直到其为 0，则表示计算完成。

备注

在当前操作进行时，不应更改 MATHACL 控制寄存器，否则将导致未定义的行为。

MATHACL 时序

每次 MATHACL 计算的时序取决于小数位数 (n)、SINCOS、ATAN2 和 SQRT 函数中的迭代次数 (NUMITER)，或软件开销。有关硬件周期数（如果有）的详细信息，请参阅 节 5.4 中每个函数示例。

5.4 配置详细信息及示例

5.4.1 正弦和余弦 (SINCOS)

正弦和余弦 (SINCOS) 函数是使用 CORDIC 算法计算的。该算法依赖于将角度表示为每单位格式的角度和，并通过可配置的迭代次数以递增方式计算结果。给定角度的正弦和余弦是根据 CTL.NUMITER 字段中指定的迭代次数同时计算的，并存储在 RES1 和 RES2 寄存器中。

表 5-4 展示了 SINCOS 的配置寄存器。

表 5-4. 正弦/余弦配置寄存器

寄存器（位字段）	值	说明
OP1	用户值	每单位角度（SQ0.31 格式）。每单位角度的计算公式为 n = 角度/180，其中 n 是 [-1,1) 之间的每单位角度，角度介于 [-180,180) 度之间。
CTL.FUNC	1h	SINCOS
CTL.NUMITER	用户值	迭代次数
RES1	32 位结果	角度的余弦（SQ0.31 格式）
RES2	32 位结果	角度的正弦（SQ0.31 格式）

迭代次数 (NUMITER) 越高，计算结果越准确，但计算时间越长。

状态、错误和溢出

SINCOS 没有状态、错误或溢出位。

配置

要执行 SINCOS，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 1h。
   b. 将 NUMITER 设置为计算的迭代次数。
2. 在 OP2 中设置单位角度（SQ0.31 格式）以开始计算。
3. 以 RES1（SQ0.31 格式）读取角度的余弦。
4. 以 RES2（SQ0.31 格式）读取角度的正弦。

5.4.2 反正切 (ATAN2)

反正切函数 (ATAN2) 计算 (y/x) 的反正切，其中 y 和 x 是标准化坐标。ATAN2 算法通过对组成角求总和来表示产生的角，即递增累积 y 坐标值为 0 之前的各组成角。计算反正切的迭代次数是使用 CTL.NUMITER 字段指定的，单位角度的最终结果以带符号数的形式存储在 RES1 中。

在执行 ATAN2 函数之前，应将 x 和 y 坐标归一化为单位向量，可以使用各种类型的归一化算法来完成此操作。图 5-1 展示了一个归一化算法。

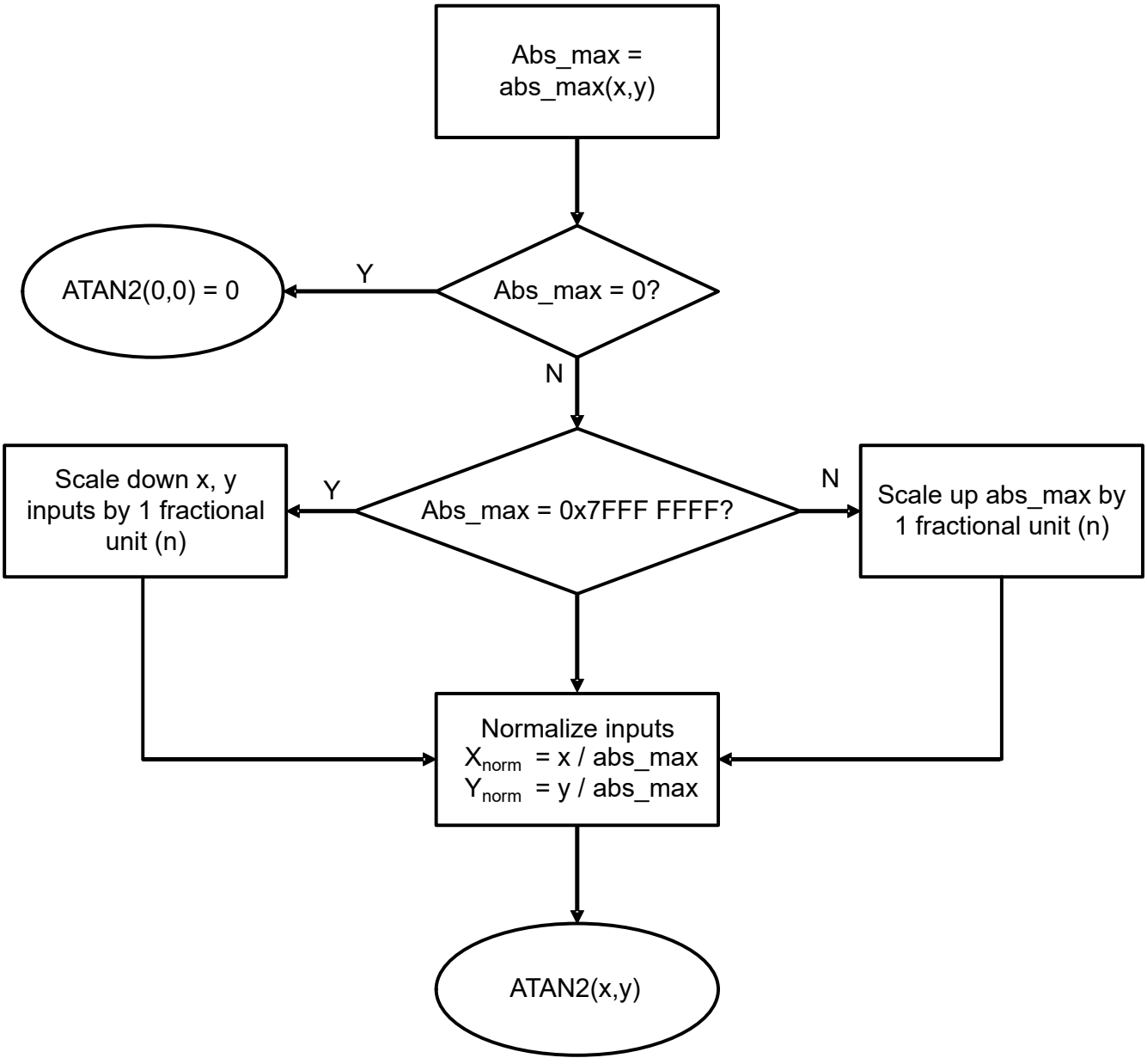


图 5-1. ATAN2 输入的归一化算法

表 5-5 展示了 ATAN2 函数的配置寄存器。

表 5-5. ATAN2 配置寄存器

寄存器 (位字段)	值	说明
OP1	用户值	X 坐标 (SQ0.31 格式)

表 5-5. ATAN2 配置寄存器 (continued)

寄存器 (位字段)	值	说明
OP2	用户值	Y 坐标 (SQ0.31 格式)
CTL.FUNC	2h	ATAN2
CTL.NUMITER	用户值	迭代次数
RES1	32 位值	每单位角度 (SQ0.31 格式)。每单位角度的计算公式为 n = 角度/180，其中 n 是 [-1,1) 之间的每单位角度，角度介于 [-180,180) 度之间。

迭代次数 (NUMITER) 越高，计算结果越准确，但计算时间越长。

状态、错误和溢出

ATAN2 没有状态、错误或溢出位。

配置

要执行 ATAN2，请执行以下操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 2h。
   b. 将 NUMITER 设置为计算的迭代次数。
2. 在 OP1 中设置归一化的 x 坐标 (SQ0.31 格式)。
3. 在 OP2 中设置归一化的 y 坐标 (SQ0.31 格式) 以开始计算。
4. 读取 RES1 中的归一化反正切结果 (SQ0.31 格式)。

5.4.3 平方根 (SQRT)

平方根函数 (SQRT) 计算一个数 (被开方数) 的平方根。平方根根据 CTL.NUMITER 字段中的迭代次数递增计算。迭代次数越多，计算结果越精确，但计算时间越长。

被开方数必须是 UQm.n 数据类型，并分为两个部分：未缩放的数字 (UN) 和缩放的数字 (SN)。未缩放的数字是数字的底，而缩放的数字介于 1.0 和 2.0 之间 (格式为 UQ2.30)。缩放的数字 (SN) 和未缩放的数字 (UN) 之间的关系是比例因子 (SFACTOR)，如方程式 16 所示。

$$Scaled\ number\ \left(SN\right) = \frac{Unscaled\ number\ (UN)}{2^{SFACTOR}} \tag{16}$$

为了从 UQm.n 格式的被开方数计算 SN 和 SFACTOR，可以使用以下算法。

```
//Compute scale factor and scaled number from radicand

UN = floor(radicand);          //unsigned number
SFACTOR = 0;                   //scale factor
count = 2;                     //scaled number must be between 1 and 2
do {
   SFACTOR++;                  //incease SFACTOR
   count <<= 1;                //multiply by 2
} while (count < UN)           //repeat until UN is less than count

SN = radicand / (count >> 1); //scale number = radicand / 2^SN
```

例如，如果 10.375 是被开方数，未缩放的数字为 10，SFACTOR = 3 ($10.375/2^3 = 1.297$)，而 1.297 是缩放的数字，介于 1.0 和 2.0 之间。

表 5-6 显示了用于 SQRT 函数的配置寄存器。

表 5-6. SQRT 配置寄存器

寄存器 (位字段)	值	说明
OP1	用户值	采用 UQ2.30 格式的缩放数字 (SN)。缩放的数字从未缩放的数字缩放，而缩放的数字必须介于 1.0 和 2.0 之间。
CTL.FUNC	5h	SQRT
CTL.SFACTOR	用户值	比例因子 n。缩放的数字 (Sn) = (未缩放的数字) /2^n。这是一个有符号整数，以二进制补码表示正值或负值。
CTL.NUMITER	用户值	迭代次数
RES1	32 位结果	UQ16.16 格式的未缩放数字的平方根。

迭代次数 (NUMITER) 越高，计算结果越准确，但计算时间越长。

状态、错误和溢出

SQRT 没有状态、错误或溢出位。

配置

要执行 SQRT，请执行以下操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 5h。
   b. 将 NUMITER 设置为计算的迭代次数。
   c. 将 SFACTOR 设置为比例因子 n。
2. 在 OP2 中设置缩放的数字 (UQ2.30 格式) 以开始计算。
3. 读取 RES1 中的平方根结果 (UQ16.16 格式)。

5.4.4 除法 (DIV)

除法函数 (DIV) 使用已知的被除数和除数进行计算。

$$Dividend = Divisor*Quotient + Remainder \tag{17}$$

$$mod(Remainder) < mod(Divisor) \tag{18}$$

表 5-7 显示了 DIV 函数可以执行的数据类型。

表 5-7. 支持的 DIV 数据类型

数据类型 (被除数和除数)	数据类型 (商)	数据类型 (余数)
Uint32_t	Uint32_t	Uint32_t
Int32_t	Int32_t	Int32_t
SQm.n	SQm.n	不适用
UQm.n	UQm.n	不适用

表 5-8 显示了 DIV 函数的配置寄存器。

表 5-8. DIV 配置寄存器

寄存器 (位字段)	值	说明
OP1	用户值	被除数
OP2	用户值	分频系数
CTL.FUNC	4h	DIV
CTL.QVAL	用户值	小数位数 (仅限无符号或有符号数数据类型)

表 5-8. DIV 配置寄存器 (continued)

寄存器 (位字段)	值	说明
CTL.OPTYPE	0 = 无符号操作数 1 = 有符号操作数	操作数符号
RES1	32 位结果	商 (与被除数和除数数据类型相同)
RES2	32 位结果	余数 (与被除数和除数数据类型相同)

备注

OP1 和 OP2 的数据类型必须相同，否则会出现使用错误的结果。

有符号整数 (int32_t) 计算需要 4 个周期来计算 OP2 写入的 RES1 和 RES2 结果。表 5-9 提供了 int32_t 的 DIV 示例。

表 5-9. int32_t 数据类型的除法示例

被除数	分频系数	商	余数
15 (0x0000000F)	2 (0x00000002)	7 (0x00000007)	1 (0x00000001)
-15 (0xFFFFFFF1)	2 (0x00000002)	-7 (0xFFFFFFF9)	-1 (0xFFFFFFFF)
15 (0x0000000F)	-2 (0xFFFFFFFE)	-7 (0xFFFFFFF9)	1 (0x00000001)
-15 (0xFFFFFFF1)	-2 (0xFFFFFFFE)	7 (0x00000007)	-1 (0xFFFFFFFF)

有符号整数 (SQm.n) 计算需要 n 个周期来计算 OP2 写入的 RES1 和 RES2 结果，其中，n 表示小数位数。表 5-10 提供了 SQ15.16 的 DIV 示例。

表 5-10. SQ15.16 数据类型的除法示例

被除数	分频系数	商	余数
7.5 (0x00078000)	4.5 (0x00048000)	1.666657 (0x0001AAAA)	不适用
-7.5 (0xFFF88000)	4.5 (0x00048000)	-1.666657 (0xFFFE5556)	不适用
7.5 (0x00078000)	-4.5 (0xFFFB8000)	-1.666657 (0xFFFE5556)	不适用
-7.5 (0xFFF88000)	-4.5 (0xFFFB8000)	1.666657 (0x0001AAAA)	不适用

状态、错误和溢出

如果除数为 0，则设置 STATUS.ERR 标志。

如果操作结果超过 32 位，则设置 STATUS.OVF 表示发生了溢出。该设置状态将保持，直到通过向 CLR.CLR_OVF 写入 1 而被清除为止。

对于有符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0x7FFFFFFF) 和最小负结果 (0x80000000)。

对于无符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0xFFFFFFFF)。

配置

要执行 DIV，请执行以下操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设为 4h。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
   c. 将 QVAL 设置为有符号数 (SQm.n) 或无符号数 (UQm.n) 的小数位数 (n)
2. 在 OP1 中设置被除数。
3. 在 OP2 中设置除数来开始计算。

4. 在 RES1 中读取商，在 RES2 中读取余数。

5.4.5 乘法

乘法函数给出两个数字相乘的结果。乘法函数存在多种变体，包括与 32 位和 64 位结果进行乘法和平方运算。

5.4.5.1 Multiply32 (MPY32)

Multiply32 (MPY32) 函数返回将两个 32 位数字与 RES1 中的 32 位结果相乘的结果。

MPY32 函数可以支持表 5-11 中所示的数据类型。

表 5-11. MPY32 支持的数据类型

数据类型（被乘数和乘数）	数据类型（乘积）	条件
Uint32_t	Uint32_t	乘积小于 2^{32} - 1
Int32_t	Int32_t	乘积大于 -2^{31} 且小于 2^{31} - 1
SQm.n	SQm.n	乘积没有超出所选数据格式可表示的范围
UQm.n	UQm.n	

表 5-12 展示了 MPY32 函数的操作数和说明。

表 5-12. MPY32 配置寄存器

寄存器（位字段）	值	说明
OP1	用户值	被乘数
OP2	用户值	乘数
CTL.FUNC	6h	MPY32
CTL.QVAL	用户值	小数位数（仅限无符号或有符号数数据类型）
CTL.OPTYPE	0h = 无符号操作数 1h = 有符号操作数	操作数符号
RES1	32 位乘积	产品

备注

OP1 和 OP2 的数据类型必须相同，否则会出现使用错误的结果。

有符号整数 (int32_t) 计算需要 1 个周期来计算 RES1 中来自 OP2 写入的结果。表 5-13 提供了 int32_t 的 MPY32 示例。

表 5-13. 使用有符号整数 (int32_t) 的 MPY32 示例

被乘数	乘数	产品
15 (0x0000000F)	2 (0x00000002)	30 (0x0000001E)
-15 (0xFFFFFFF1)	2 (0x00000002)	-30 (0xFFFFFFE2)
15 (0x0000000F)	-2 (0xFFFFFFFE)	-30 (0xFFFFFFE2)
-15 (0xFFFFFFF1)	-2 (0xFFFFFFFE)	30 (0x0000001E)

有符号整数 (SQm.n) 计算需要 n 个周期来计算 RES1 中来自 OP2 写入的结果，其中，n 表示小数位数。表 5-14 提供了 SQ15.16 的 MPY32 示例。

表 5-14. 使用 SQ15.16 有符号数的 MPY32 示例

被乘数	乘数	产品
1.5 (0x00018000)	2.5 (0x00028000)	3.75 (0x0003C000)
-1.5 (0xFFFE8000)	2.5 (0x00028000)	-3.75 (0xFFFCC000)
1.5 (0x00018000)	-2.5 (0xFFFD8000)	-3.75 (0xFFFCC000)

表 5-14. 使用 SQ15.16 有符号数的 MPY32 示例 (continued)

被乘数	乘数	产品
-1.5 (0xFFFE8000)	-2.5 (0xFFFD8000)	3.75 (0x0003C000)

状态、错误和溢出

如果操作结果超过 32 位，则设置 STAT.OVF 以指示发生了溢出。将保持该设置状态，直到通过在 CLR 寄存器中写入设置 CLR_OVF=1 来清除为止。

对于有符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则设置 STAT.OVF 位，并且结果将饱和至最大正结果 (0x7FFFFFFF)。

对于无符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0xFFFFFFFF)。

配置

要执行 MPY32，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 6h。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
   c. 将 QVAL 设置为有符号数 (SQm.n) 或无符号数 (UQm.n) 的小数位数 (n)
2. 在 OP1 中设置乘数。
3. 在 OP2 中设置被乘数以开始计算。
4. 读取 RES1 中的乘积结果。

5.4.5.2 Square32 (SQUARE32)

Square32 (SQUARE32) 函数是 MPY32 函数的变体，对操作数 OP2 执行平方运算，并将 32 位结果存储在 RES1 中。

SQUARE32 函数可以执行表 5-15 中所示的数据类型。

表 5-15. 支持的 SQUARE32 数据类型

数据类型 (基数)	数据类型 (结果)	条件
Uint32_t	Uint32_t	结果 < 2^{32} - 1
Int32_t	Int32_t	结果 > -2^{31} 且 < 2^{31} - 1
SQm.n	SQm.n	结果不超出所选数据格式可表示的范围
UQm.n	UQm.n	

表 5-12 显示了 SQUARE32 函数的操作数和说明。

表 5-16. SQUARE32 配置寄存器

寄存器 (位字段)	值	说明
OP2	用户值	进制
CTL.FUNC	7h	MPY32
CTL.QVAL	用户值	小数位数 (仅限无符号或有符号数数据类型)
CTL.OPTYPE	0h = 无符号操作数 1h = 有符号操作数	操作数符号
RES1	32 位平方结果	平方结果

有符号整数 (int32_t) 计算需要 1 个周期来计算 RES1 中来自 OP2 写入的结果。表 5-17 提供了 int32_t 数据类型的 SQUARE32 示例。

表 5-17. 使用 int32_t 的 SQUARE32 示例

被乘数/乘数	方形
4 (0x00000004)	16 (0x00000010)
-4 (0xFFFFFFFB)	16 (0x00000010)

有符号整数 (SQm.n) 计算需要 n 个周期来计算 RES1 中来自 OP2 写入的结果，其中，n 表示小数位数。表 5-18 提供了 SQ15.16 的除法示例。

表 5-18. 使用 SQ15.16 有符号数的 SQUARE32 示例

被乘数/乘数	方形
1.5 (0x00018000)	2.25 (0x00024000)
-1.5 (0xFFFE8000)	2.25 (0x00024000)

状态、错误和溢出

如果操作结果超过 32 位，则设置 STAT.OVF 以指示发生了溢出。将保持该设置状态，直到通过在 CLR 寄存器中写入设置 CLR_OVF=1 来清除为止。

对于有符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则设置 STAT.OVF 位，并且结果将饱和至最大正结果 (0x7FFFFFFF)。

对于无符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0xFFFFFFFF)。

配置

要执行 SQUARE32，请执行以下操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 6h。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
   c. 将 QVAL 设置为有符号数 (SQm.n) 或无符号数 (UQm.n) 的小数位数 (n)
2. 在 OP2 中设置底数以开始计算。
3. 读取 RES1 中的乘积结果。

5.4.5.3 Multiply64 (MPY64)

Multiply64 (MPY64) 函数返回将两个 32 位数字与 RES1 和 RES2 中的 64 位结果相乘的结果。

MPY64 函数可以执行表 5-11 中所示的乘法类型。

表 5-19. MPY64 支持的数据类型

数据类型（被乘数和乘数）	数据类型（乘积）	条件
Uint32_t	Uint64_t	乘积小于 2^{64} - 1
Int32_t	Int64_t	乘积大于 -2^{63} 且小于 2^{63} - 1

表 5-12 展示了该乘法函数的操作数和说明。

表 5-20. MPY64 配置寄存器

寄存器（位字段）	值	说明
OP1	用户值	被乘数
OP2	用户值	乘数
CTL.FUNC	8h	MPY64

表 5-20. MPY64 配置寄存器 (continued)

寄存器 (位字段)	值	说明
CTL.OPTYPE	0h = 无符号操作数 1h = 有符号操作数	操作数符号
RES1	乘积的低 32 位	乘积 (64 位)
RES2	乘积的高 32 位	

备注

OP1 和 OP2 的数据类型必须相同，否则会出现使用错误的结果。

状态、错误和溢出

MPY64 没有状态、错误或溢出位。

配置

要执行 MPY64，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 8h。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
2. 在 OP1 中设置乘数。
3. 在 OP2 中设置被乘数以开始计算。
4. 读取 RES1 和 RES2 中的乘积结果。

5.4.5.4 Square64 (SQUARE64)

Square64 函数 (SQUARE64) 是 MPY32 函数的变体，对操作数 OP2 执行平方运算，并将 64 位结果存储在 RES1 和 RES2 中。

SQUARE64 函数可以执行表 5-11 中所示的乘法类型。

表 5-21. 支持的 SQUARE64 数据类型

数据类型 (基数)	数据类型 (平方结果)	条件
Uint32_t	Uint64_t	结果 < 2^{64} - 1
Int32_t	Int64_t	结果 > -2^{63} 且 < 2^{63} - 1

表 5-12 展示了 SQUARE64 函数的操作数和说明。

表 5-22. SQUARE64 配置寄存器

寄存器 (位字段)	值	说明
OP1	用户值	进制
CTL.FUNC	9h	MPY64
CTL.OPTYPE	0h = 无符号操作数 1h = 有符号操作数	操作数符号
RES1	乘积的低 32 位	乘积 (64 位)
RES2	乘积的高 32 位	

状态、错误和溢出

SQUARE64 没有状态、错误或溢出位。

配置

要执行 SQUARE64，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 8h。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
2. 在 OP2 中设置底数以开始计算。
3. 读取 RES1 和 RES2 中的乘积结果。

5.4.6 乘法累加 (MAC)

乘法累加 (MAC) 函数在结果寄存器 RES1 和 RES2 内多次迭代累加 OP1 和 OP2 的乘积，直到 MATHACL 为另一个函数重新编程。MATHACL 仅支持 32 位乘数和被乘数操作数大小，以及 64 位累加结果大小。

图 5-2 显示了 MAC 模式下 MATHACL 的操作。

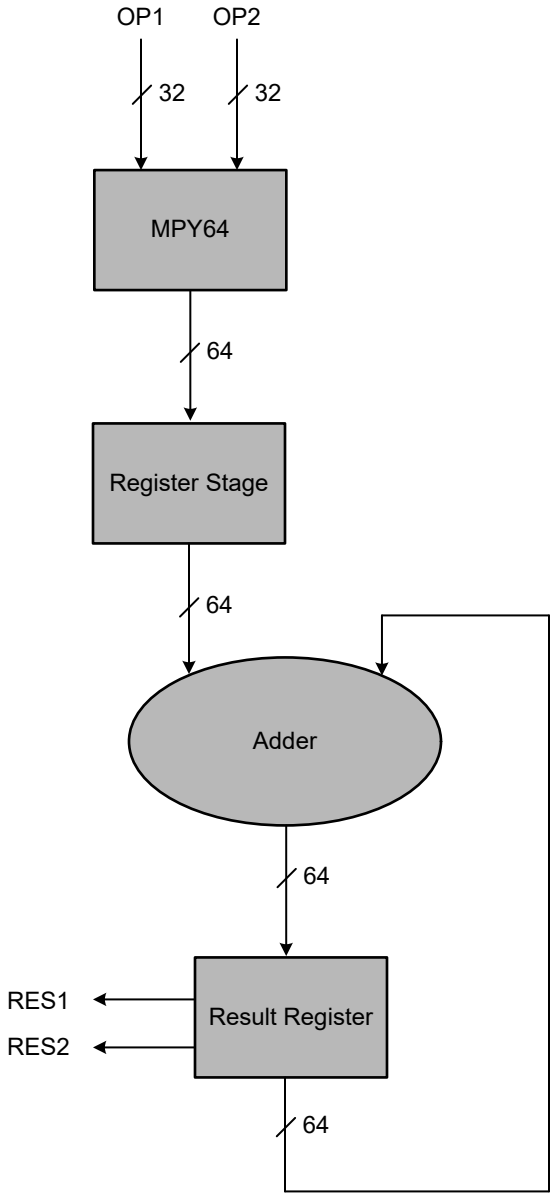


图 5-2. MAC 操作

内部 MAC 操作执行以下步骤：

- OP1 和 OP2 相乘，得到 64 位乘积
  - 如果使用 SQm.n 或 UQm.n 数据类型，则乘积的小数位数 (n) 与操作数相同
- 乘积结果存储在寄存器阶段，以得到正确时序
- 寄存器值被馈送到一个 64 位加法器中，而当前结果寄存器值则作为另一个输入（结果寄存器+=寄存器阶段）
- 加法运算的结果存储在 RES1 和 RES2 中

备注

在启动 MAC 操作之前将 0 写入 RES1 和 RES2 以获得正确的结果。

MAC 函数可以执行表 5-23 中所示的以下数据类型的乘法累加。

表 5-23. 支持的 MAC 数据类型

数据类型（被乘数和乘数）	数据类型（乘积）	条件
Uint32_t	Uint64_t	MAC 结果 < 2^{64} - 1
Int32_t	Int64_t	MAC 结果 > -2^{63} 且 < 2^{63} - 1
SQm.n	SQ(32+m).n	MAC 结果不超出所选数据格式可表示的范围
UQm.n	UQ(32+m).n	

表 5-24 显示了用于 MAC 函数的配置寄存器。

表 5-24. MAC 配置寄存器

寄存器（位字段）	值	说明
OP1	用户值	被乘数
OP2	用户值	乘数
CTL.FUNC	Ah	MAC
CTL.QVAL	用户值	小数位数（仅限无符号或有符号数数据类型）
CTL.OPTYPE	0 = 无符号操作数 1 = 有符号操作数	操作数符号
RES1	结果的低 32 位	乘法累加结果（64 位）
RES2	结果的高 32 位	

备注

OP1 和 OP2 的数据类型必须相同，否则会出现使用错误的结果。

写入 OP2 后 2 个周期后，可以读取 MAC 操作的最终结果。

状态、错误和溢出

如果操作结果超过 64 位，则设置 STAT.OVF 以指示发生了溢出。该设置状态将保持，直到通过向 CLR.CLR_OVF 写入 1 而被清除为止。

对于有符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0x7FFF_FFFF_FFFF_FFFF) 和最小负结果 (0x8000_0000_0000_0000)。

对于无符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0xFFFF_FFFF_FFFF_FFFF)。

配置

要配置 MAC 的运行，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设为 Ah。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。

c. 将 QVAL 设置为有符号数 (SQm.n) 或无符号数 (UQm.n) 的小数位数 (n)
2. 通过将 RES1 和 RES2 设置为 0 来清除结果寄存器。
3. 在 OP1 中设置乘数。
4. 在 OP2 中设置被乘数以开始计算。
5. 读取 RES1 和 RES2 中的 64 位乘积结果。
6. 重复步骤 3-5，继续对 RES1 和 RES2 中的数据进行乘法运算和累加。

5.4.7 平方累加 (SAC)

平方累加 (SAC) 函数会在多次迭代期间累积结果寄存器 RES1 和 RES2 内 OP2 操作数的平方结果，直到为另一个函数重新编程 MATHACL。它是 MAC 函数的变体，运行细节保持不变。

备注

在开始 SAC 操作之前将 0 写入 RES1 和 RES2，以获得正确的结果。

SAC 函数可以执行表 5-25 中所示的以下平方累加数据类型。

表 5-25. 支持的 SAC 数据类型

数据类型（基数）	数据类型（结果）	条件
Uint32_t	Uint64_t	SAC 结果应 < 2^{64} - 1
Int32_t	Int64_t	SAC 结果应 > -2^{63} 且 < 2^{63} - 1
SQm.n	SQ(32+m).n	SAC 结果不应超出所选数据格式可表示的范围
UQm.n	UQ(32+m).n	

表 5-26 展示了 SAC 函数的配置寄存器。

表 5-26. SAC 配置寄存器

寄存器（位字段）	值	说明
OP1	用户值	进制
CTL.FUNC	Bh	SAC
CTL.QVAL	用户值	小数位数（仅限无符号或有符号数数据类型）
CTL.OPTYPE	0 = 无符号操作数 1 = 有符号操作数	操作数符号
RES1	结果的低 32 位	平方累加结果（64 位）
RES2	结果的高 32 位	

写入 OP2 后 2 个周期后，可以读取 SAC 操作的最终结果。

状态、错误和溢出

如果操作结果超过 64 位，则设置 STAT.OVF 以指示发生了溢出。该设置状态将保持，直到通过向 CLR.CLR_OVF 写入 1 而被清除为止。

对于有符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0x7FFF_FFFF_FFFF_FFFF) 和最小负结果 (0x8000_0000_0000_0000)。

对于无符号运算，如果在溢出的情况下启用了结果饱和 (CTL.SATEN = 1)，则结果将饱和至最大正结果 (0xFFFF_FFFF_FFFF_FFFF)。

配置

要配置 SAC 的运行，请按以下步骤操作：

1. 在 CTL 寄存器中：
   a. 将 FUNC 设置为 Bh。
   b. 如果数据类型为无符号，则将 OPTYPE 设置为 0；如果数据类型为有符号，则设置为 1。
   c. 将 QVAL 设置为有符号数 (SQm.n) 或无符号数 (UQm.n) 的小数位数 (n)
2. 通过将 RES1 和 RES2 设置为 0 来清除结果寄存器。
3. 在 OP2 中设置底数以开始计算。
4. 读取 RES1 和 RES2 中的 64 位平方结果。
5. 重复步骤 3-4，继续对 RES1 和 RES2 中的数据进行平方和累加。

5.5 MATHACL 寄存器

表 5-27 列出了 MATHACL 寄存器的存储器映射寄存器。表 5-27 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 5-27. MATHACL 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1100h	CTL	控制寄存器		转到
1118h	OP2	操作数 2 寄存器。		转到
111Ch	OP1	操作数 1 寄存器		转到
1120h	RES1	结果 1 寄存器		转到
1124h	RES2	结果 2 寄存器		转到
1130h	状态	状态寄存器		转到
1140h	STATUSCLR	状态标志清除寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 5-28 显示了适用于此部分中访问类型的代码。

表 5-28. MATHACL 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
RH	R H	读取 由硬件置位或清零
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-*n*		复位后的值或默认值

5.5.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 5-3 展示了 PWREN，表 5-29 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 5-3. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 5-29. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

5.5.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 5-4 展示了 RSTCTL，表 5-30 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 5-4. RSTCTL

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

表 5-30. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

5.5.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 5-5 展示了 STAT，表 5-31 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 5-5. STAT

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

表 5-31. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

5.5.4 CTL（偏移 = 1100h）[复位= 00000000h]

图 5-6 展示了 CTL，表 5-32 中对此进行了介绍。

返回到汇总表。

控制寄存器

图 5-6. CTL

31	30	29	28	27	26	25	24
RESERVED			NUMITER				
R/W-0h			R/W-0h				
23	22	21	20	19	18	17	16
RESERVED	SATEN	SFACTOR					
R/W-0h	R/W-0h	R/W-0h					
15	14	13	12	11	10	9	8
保留			QVAL				
R/W-0h			R/W-0h				
7	6	5	4	3	2	1	0
RESERVED		OPTYPE	FUNC				
R/W-0h		R/W-0h	R/W-0h				

表 5-32. CTL 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28-24	NUMITER	R/W	0h	迭代次数，适用于函数迭代计算的情况，例如，正弦/余弦/atan2/sqrt。 注意：值 0 被解释为 31。
23	RESERVED	R/W	0h	
22	SATEN	R/W	0h	饱和使能 该位在 DIV、SQUARE32、MPY32、MAC 和 SAC 函数之间共享。 启用后，在出现溢出事件时它将使结果饱和到最大值。 禁用后，结果将溢出到未知值。 0h = Saturation 已禁用 1h = Saturation 已启用
21-16	SFACTOR	R/W	0h	比例因数。对于 SQRT 函数，输入操作数需要位于一个范围内。如果没有，则必须将其缩放至 $2^{+/-n}$。此字段应使用值 "n" 写入。
15-13	RESERVED	R/W	0h	

表 5-32. CTL 字段说明 (continued)

位	字段	类型	复位	说明
12-8	QVAL	R/W	0h	指示操作数中的小数位，范围为 0 到 31。适用于 DIV 函数。 0h = Q0 操作数 1h = Q1 操作数 2h = Q2 操作数 3h = Q3 操作数 4h = Q4 操作数 5h = Q5 操作数 6h = Q6 操作数 7h = Q7 操作数 8h = Q8 操作数 9h = Q9 操作数 Ah = Q10 操作数 Bh = Q11 操作数 Ch = Q12 操作数 Dh = Q13 操作数 Eh = Q14 操作数 Fh = Q15 操作数 10h = Q16 操作数 11h = Q17 操作数 12h = Q18 操作数 13h = Q19 操作数 14h = Q20 操作数 15h = Q21 操作数 16h = Q22 操作数 17h = Q23 操作数 18h = Q24 操作数 19h = Q25 操作数 1Ah = Q26 操作数 1Bh = Q27 操作数 1Ch = Q28 操作数 1Dh = Q29 操作数 1Eh = Q30 操作数 1Fh = Q31 操作数
7-6	RESERVED	R/W	0h	
5	OPTYPE	R/W	0h	操作数类型，可以是有符号的，也可以是无符号的。适用于 DIV 函数。 0h = 无符号操作数 1h = 有符号操作数。
4-0	FUNC	R/W	0h	ULP_ADCHP 使能转换。 0h = 无操作 1h = 正弦和余弦操作 2h = 以 x 和 y 值作为操作数的反正切。 4h = 除法，操作数是分子、分母和除法类型。结果是商和提醒。 5h = DO 平方根。操作数是需要计算的平方根数。超出范围的数字需要缩小 2 次方 2n 才能使其在范围内。 6h = 32 位乘法结果 7h = 32 位平方结果 8h = 64 位乘法结果 9h = 64 位乘法结果 Ah = 乘法和累加运算 Bh = 平方和累加运算

5.5.5 OP2（偏移 = 1118h）[复位 = 00000000h]

图 5-7 展示了 OP2，表 5-33 中对此进行了介绍。

返回到汇总表。

操作数 2 寄存器

图 5-7. OP2

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

表 5-33. OP2 字段说明

位	字段	类型	复位	说明
31-0	数据	R/W	0h	操作数 2 寄存器

5.5.6 OP1（偏移 = 111Ch）[复位 = 00000000h]

图 5-8 展示了 OP1，表 5-34 中对此进行了介绍。

返回到汇总表。

操作数 1 寄存器

图 5-8. OP1

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

表 5-34. OP1 字段说明

位	字段	类型	复位	说明
31-0	数据	R/W	0h	操作数 1 寄存器

5.5.7 RES1（偏移 = 1120h）[复位= 00000000h]

图 5-9 展示了 RES1，表 5-35 中对此进行了介绍。

返回到汇总表。

结果 1 寄存器。

图 5-9. RES1

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

表 5-35. RES1 字段说明

位	字段	类型	复位	说明
31-0	数据	RH/W	0h	结果 1 寄存器

5.5.8 RES2（偏移 = 1124h）[复位 = 00000000h]

图 5-10 展示了 RES2，表 5-36 中对此进行了介绍。

返回到汇总表。

结果 2 寄存器

图 5-10. RES2

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

表 5-36. RES2 字段说明

位	字段	类型	复位	说明
31-0	数据	RH/W	0h	结果 2 寄存器

5.5.9 STATUS（偏移 = 1130h）[复位 = 00000000h]

图 5-11 展示了 STATUS，表 5-37 中对此进行了介绍。

返回到汇总表。

状态寄存器

图 5-11. STATUS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							BUSY
R-0h							RH-0h
7	6	5	4	3	2	1	0
RESERVED				ERR		OVF	UF
R-0h				RH-0h		R-0h	R-0h

表 5-37. STATUS 字段说明

位	字段	类型	复位	说明
31-9	保留	R	0h	
8	BUSY	RH	0h	MATHACL 繁忙位。 0h = 计算已完成。 1h = 计算进行中
7-4	RESERVED	R	0h	
3-2	ERR	RH	0h	错误输入/输出。 0h = 计算无误。 1h = DIVBY0 错误
1	OVF	R	0h	用于 MPY32、SQUARE32、DIV、MAC 和 SAC 函数的溢出位 此位将在溢出时设置，并将保留其值，直到通过将 1 写入 CLR.CLR_OVF 的方式将其清除 0h = 溢出错误。
0	UF	R	0h	下溢标志 0h = 无下溢错误。 1h = 下溢错误。

5.5.10 STATUSCLR（偏移 = 1140h）[复位= 00000000h]

图 5-12 展示了 STATUSCLR，表 5-38 中对此进行了介绍。

返回到汇总表。

用于清除 STATUS 寄存器中标志的清除寄存器

图 5-12. STATUSCLR

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

表 5-38. STATUSCLR 字段说明

位	字段	类型	复位	说明
31-3	保留	W	0h	
2	CLR_ERR	W	0h	向该位写入 1 将 STATUS.ERR 字段清零 0h = 写入 0 没有任何影响 1h = 将 STATUS.ERR 清零
1	CLR_OVF	W	0h	向该位写入 1 将 STATUS.OVF 位清零 0h = 写入 0 没有任何影响 1h = 将 STATUS.OVF 清零
0	CLR_UF	W	0h	向该位写入 1 将 STATUS.UF 位清零 0h = 写入 0 没有任何影响 1h = 将 STATUS.UF 清零

This page intentionally left blank.

章节 6

NVM（闪存）

非易失性存储器 (NVM) 系统提供用于存储可执行代码和数据的非易失性闪存。

6.1 NVM 概述

非易失性存储器系统提供用于存储可执行代码和数据的系统内可编程闪存。本章介绍非易失性存储器系统提供的全部功能。

6.1.1 关键特性

非易失性存储器系统的主要特性包括：

- 在整个电源电压范围内支持电路内编程和擦除操作
- 内部编程电压生成
- 64 位闪存字大小 (存在可选 ECC 时为 72 位)
- 静态写入保护 (在引导时锁存并一直保持，直到 BOR 或 POR)
- 动态写保护 (可在运行时配置)
- 扇区 (1KB) 和组 (高达 256KB) 擦除
- 自动硬件预校验，以延长闪存组寿命
- 编程/擦除的自动硬件后验证
- 可选 ECC 保护 (SECDED)
- 可选组地址交换模式 (用于无缝双映像固件更新)
- 用于高效编程的可选编程寄存器高速缓存 (2、4 或 8 个闪存字)

备注

要确定器件是否具有上述任何可选功能，请查看相应器件数据表中的非易失性存储器系统详细说明。

6.1.2 系统组成部分

非易失性存储器系统由如下三个组件组成：

- 一个或多个闪存存储体 (用于存储代码和数据)
- 闪存控制器 (用于管理闪存存储体上的所有编程/擦除操作)
- 读取接口 (用于将闪存存储体连接到 CPU 和外设总线)

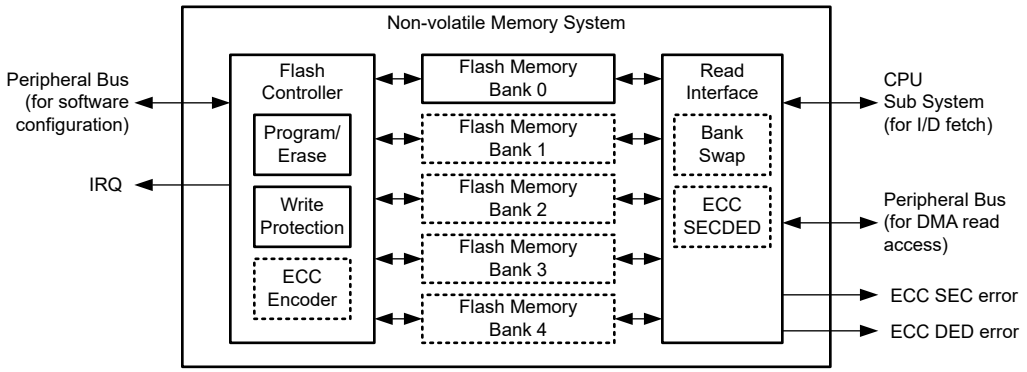


图 6-1. 非易失性存储器系统组件

6.1.3 术语

本节定义了重要的闪存存储体术语，用作本章其余部分的参考。

表 6-1. NVM 系统术语

术语	定义	大小
闪存字	闪存编程和读取操作的基本数据大小 (也是针对系统的读取总线宽度)	64 个数据位 (含 ECC 为 72 位)
字线	扇区内的一组闪存字，在扇区擦除前具有最大编程操作限制	16 个闪存字 (128 个数据字节，可选 16 个 ECC 字节)

表 6-1. NVM 系统术语 (continued)

术语	定义	大小
扇区	一起擦除的一组字线 (闪存的最小擦除分辨率)	8 个字线 (1024 个数据字节，可选 128 个 ECC 字节)
存储体	在一次操作中可以批量擦除的一组扇区。在同一时间针对给定的存储体只能进行一个读取、编程、擦除或验证操作。	变量
区域	在存储体中以逻辑形式分配的闪存区域。	变量

6.2 闪存存储体结构

闪存用于存储应用代码和数据、器件引导配置以及 TI 在工厂预编程的参数。闪存分为一个或多个存储体，每个存储体中的存储器进一步映射到一个或多个逻辑存储区域，并分配有系统地址空间以供应用程序使用。

6.2.1 存储体

非易失性存储器系统支持多达 5 个闪存存储体 (枚举为 BANK0 到 BANK4) 。存在的闪存存储体数量取决于器件。要确定特定器件的存储体方案，请查看特定器件数据表的详细说明部分。大多数器件都会实现一个闪存存储体 (BANK0)。

在具有单个闪存存储体的器件上，一个正在进行的编程/擦除操作将暂停对闪存的所有读取请求，直到操作完成并且闪存控制器已经释放了对存储体的控制。在具有多个闪存存储体的器件上，一个存储体上的编程/擦除操作也将暂停向正在执行编程/擦除操作的存储体发出的读取请求，但不会暂停向任何其他存储体发出的读取请求。因此，存在多个存储体可实现以下应用案例：

- 双映像固件更新 (应用程序可以从一个闪存存储体中执行代码，同时将第二个映像编程到第二个对称的闪存存储体，而不会暂停应用程序的执行)
- EEPROM 仿真 (应用程序可以从一个闪存存储体中执行代码，第二个闪存存储体用于写入数据，而不会暂停应用程序的执行)

6.2.2 闪存区域

根据每个组中存储器所支持的功能，每个组中的存储器映射到一个或多个逻辑区域。有四个区域：FACTORY、NONMAIN、MAIN 和 DATA。

表 6-2. 闪存区域

闪存区域	区域内容	可执行	使用者	编程者
FACTORY	器件 ID 和其他参数	否	应用程序	仅限 TI (不可修改)
NONMAIN	器件引导配置 (BCR 和 BSL)	否	引导 ROM	TI、用户
MAIN	应用程序代码和数据	是	应用程序	用户
DATA	数据或 EEPROM 仿真	否	应用	用户

具有一个存储体的器件在 BANK0 (唯一存在的存储体) 上实现 FACTORY、NONMAIN 和 MAIN 区域，并且数据区域不可用。具有多个存储体的器件也在 BANK0 上实现 FACTORY、NONMAIN 和 MAIN 区域，但包括可实现 MAIN 或 DATA 区域的其他存储体 (BANK1 至 BANK4) 。

有关只读 FACTORY 区域内容的详细说明，请参阅节 1.6。

6.2.3 寻址

闪存区域被分配给系统存储器映射中的地址空间。

NONMAIN、DATA 和 FACTORY 区域被分配给外设地址空间 (0x4000.0000)，因为它们不包含任何可执行代码。CPU 不应从该区域提取可执行指令。

MAIN 区域被分配给代码地址空间 (0x0000.0000) 和外设地址空间 (0x4000.0000)。建议始终通过代码地址空间执行指令和数据提取，因为这样可提供出色性能。CPU 不应从该区域提取可执行指令。

ECC

在支持纠错码 (ECC) 的器件上，所有存储器区域的 ECC 代码也被分配给地址空间，软件可以将 ECC 代码读取为数据以用于诊断目的。也可以在没有应用 ECC 校正的情况下读取任何存储器区域的内容。

6.2.3.1 闪存映射

表 6-3 中给出了系统地址空间分配，此类分配对于所有器件都是一致的。

表 6-3. 闪存区域存储器映射

区域	读取类型	ECC 行为	基址
NONMAIN	数据读取	已更正	0x41C0.0000
		未更正	0x41C1.0000
		ECC 代码	0x41C2.0000
MAIN	指令提取或数据读取	已更正	0x0000.0000
		未更正	0x0040.0000
	数据读取	已更正	0x4100.0000
		未更正	0x4140.0000
		ECC 代码	0x4180.0000
数据	数据读取	已更正	0x41D0.0000
		未更正	0x41E0.0000
		ECC 代码	0x41F0.0000
FACTORY	数据读取	已更正	0x41C4.0000
		未更正	0x41C5.0000
		ECC 代码	0x41C6.0000

NONMAIN、DATA 和 FACTORY 数据读取仅通过外设总线和外设地址空间进行处理。MAIN 区域可以通过 CPU 总线矩阵或通过外设总线访问，具体取决于使用的是代码地址空间还是外设地址空间。建议使用代码地址空间访问 CPU (指令提取或数据读取) ，因为此类访问不会跨越外设总线，因此不会与 DMA 竞争对外设总线的控制权。有关总线互连的详细说明，请参阅总线架构部分。

在具有 ECC 的器件上，对 ECC 代码地址的访问将针对所访问的整个 64 位闪存字返回 8 位 ECC 值。在没有 ECC 的器件上，对具有相同偏移量的已更正和未更正的 ECC 地址空间的访问将读取相同内容，并且 ECC 代码地址读取为 0x0。

6.2.4 存储器组织示例

图 6-2 是一个具有 64KB MAIN 区域的单组配置示例。NONMAIN 和 FACTORY 区域也包含在具有 MAIN 区域的单组 (BANK0) 中。其 MAIN 区域大小 ≤128KB 的大多数器件都实现了单组配置。

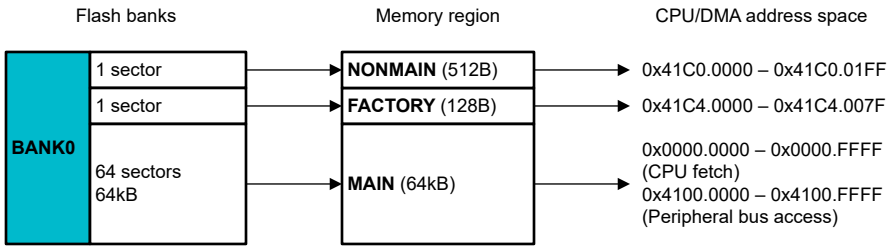


图 6-2. 存储器组织示例 - 单组配置

图 6-3 是一个三组配置的示例，其中 512KB 的 MAIN 区域跨 BANK0 和 BANK1 进行拆分，并在 BANK2 中提供一个 16KB 的 DATA 区域。与单组示例一样，BANK0 中包含 NONMAIN 和 FACTORY 区域。此示例支持在 DATA 区域中进行 EEPROM 仿真，而不会停止对 MAIN 的提取，它还支持双映像应用，其中，可以写入 BANK0

MAIN 而不会停止对 BANK1 MAIN 的提取 (反之亦然) 。其 MAIN 区域大小 ≥256KB 的大多数器件都实现了多组配置。

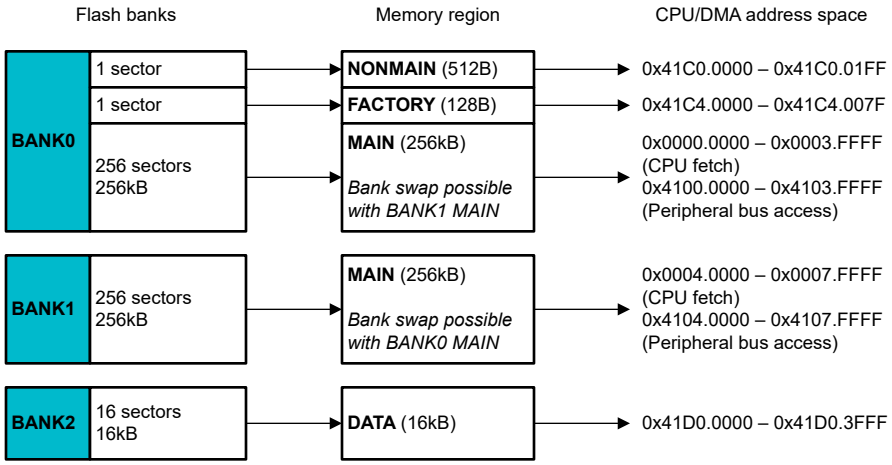


图 6-3. 存储器组织示例 - 多组配置

6.3 闪存控制器

闪存控制器管理在非易失性存储器系统上执行的所有编程、擦除和验证操作。它在器件存储器映射的外设区域中包含存储器映射寄存器，必须由软件配置这些寄存器才能对闪存执行操作。

TI 为闪存控制器提供软件抽象，作为软件开发套件 (SDK) 的 DriverLib 层的一部分。使用软件对闪存进行操作时，建议使用 DriverLib 抽象层，但不强制这样做。要使用 DriverLib 软件抽象层对闪存执行操作，请查看与本文档分开提供的软件开发套件 (SDK) 文档。要使用对闪存控制器的低级寄存器访问直接对闪存进行操作，请详细查看本节的其余部分。

备注

复位后，FLASHCTL 寄存器可能并不总是配置为默认值。如果引导配置例程 (BCR) 或自举加载程序 (BSL) 在引导期间对闪存执行操作，则可能会发生这种情况。为一项操作配置 FLASHCTL 寄存器时，请确保正确配置与该操作相关的所有寄存器。

6.3.1 闪存控制器命令概述

执行闪存操作的具体做法是为所需命令配置 CMDTYPE 和 CMDCTL 寄存器以及必须为特定命令配置的任何其他寄存器，然后将 0x01 写入 CMDEXEC 寄存器以启动该命令。

在 CMDEXEC 中设置 0x01 后，该命令的操作开始执行。正在执行操作时，大多数配置寄存器会被阻止写入，直到该操作完成。某些寄存器 (例如，屏蔽寄存器) 可以在操作完成时在硬件控制下更改状态。闪存控制器通过设置 STATCMD 寄存器中的 CMDDONE 位来指示命令的操作已完成。闪存控制器还向 CPU 子系统提供中断矢量，以便在操作完成时指示“DONE”状态。

必须从以下两个位置执行设置 CMDEXEC 位并等待 CMDDONE 响应的软件序列：从器件 SRAM 或从作为操作对象的闪存存储体之外的其他存储体，因为闪存控制器将控制正在运行的闪存存储体。在闪存控制器执行命令时对作为操作对象的闪存存储体的读取是不可预测的。

闪存控制器在 CMDTYPE 寄存器的 COMMAND 字段中提供五个基本命令来操作闪存。表 6-4 介绍了这些命令。

表 6-4. 闪存控制器命令

命令	说明
NOOP	无操作 (默认设置) 。
PROGRAM	为闪存选择一个编程操作。
ERASE	为闪存选择一个擦除操作。

表 6-4. 闪存控制器命令 (continued)

命令	说明
READVERIFY	选择一个独立的读取验证操作。
BLANKVERIFY	选择一个独立的空白验证操作。

6.3.2 NOOP 命令

如果不使用闪存控制器，最好将 COMMAND 字段设置为 NOOP，以防止在意外设置 CMDEXEC 中的 VAL 位时对闪存执行任何意外操作。执行 NOOP 命令对闪存没有影响。

6.3.3 PROGRAM 命令

PROGRAM 命令用于写入 (编程) 闪存。具体而言，编程操作的目的是将一个或多个闪存字中的闪存位从非确定性擦除状态配置为确定性编程状态。一旦使用 PROGRAM 命令对一个字节进行了编程，除非使用 ERASE 命令擦除相应扇区，否则不能对这个字节重新编程。

所有器件都支持一次对 64 个数据位 (加上具有 ECC 的器件上的 8 个 ECC 位) 进行单个闪存字编程，并可以控制将编程操作的范围限制为 64 位闪存字内的特定字节。

一些器件还支持多字编程模式，在该模式下可以通过单个命令操作写入 2 个、4 个或 8 个闪存字。多字编程 (如果可用) 可以在需要对多个字进行编程时 (例如，在生产编程或固件更新期间) 显著加快编程速度。请参阅器件特定的数据表，确定是否支持多字编程以及 (如果支持) 提供的闪存字缓冲器数量。

6.3.3.1 编程位屏蔽行为

闪存控制器提供了一个编程验证机制来延长闪存存储体的使用寿命。在编程操作期间，闪存控制器使用 CMDDATAx 寄存器作为编程位掩码来指示闪存字中的哪些特定位需要编程脉冲。因此，在开始编程操作之前加载到 CMDDATAx 寄存器中的数据将在编程操作期间和/或完成时从 CMDDATAx 寄存器中丢失。如果要再次对相同的数据进行编程，必须通过软件将需要编程的正确数据值重新加载到 CMDDATAx 寄存器中。

6.3.3.2 编程少于一个闪存字

通常，对闪存进行编程的最简单方法是一次编程一个闪存字 (如果存在 ECC，则为 64 位加 8 个 ECC 位)。可以使用 32 位、16 位或 8 位 (字节) 分辨率对闪存进行编程，但这样做时必须特别小心，确保以下几点：

1. 在具有 ECC 的器件上，必须正确处理 ECC 位以防止出现意外的 ECC 错误。
2. 必须监控应用于给定字线的编程操作数量，确保不违反擦除前的最大字线编程限制。
3. 一旦已经编程一个字节，在尝试对这个字节重新编程之前，必须擦除包含这个字节的扇区。

要编程少于一个闪存字，必须配置 CMDBYTEN 寄存器，以便在开始编程操作之前指示要对闪存字中的哪些字节进行编程。CMDBYTEN 中的每个位对应于要编程的闪存字中的一个字节 (包括 ECC 位)。

处理 ECC

在具有 ECC 的器件上，一次对 64 位数据进行编程可以确保同时正确地编程对应于 64 位数据字的 8 个 ECC 位。这样做可以防止在编程后 CPU 或 DMA 读取存储器位置时发生 ECC 错误。

如果计划使用 ECC 并且需要进行部分编程，一种方法是屏蔽 (不编程) ECC 位，直到对闪存字的所有 64 位编程完成，此时也可以对 ECC 位进行编程。清除 CMDBYTEN 寄存器中的 BIT8 (0x100) 即可屏蔽 ECC 位的编程。这样做可以防止在每次执行编程操作以重新编程 ECC 位来匹配新的 64 位数据时，必须擦除整个扇区的情况。但是，在这种情况下，如果读取已部分编程的字，但尚未写入 ECC 位，则会导致 ECC 错误。为了避免 ECC 错误，软件必须等到写入完整的 64 位闪存字和 8 个 ECC 位，或者从未校正的地址空间中读取数据。

擦除前每个字线的最大编程操作数

器件数据表指明了在需要擦除包含字线的扇区之前每条字线的编程操作数量的最大限制。超过此最大值可能会导致字线内的数据损坏。

如果执行 16 位或更大的编程操作，并且在擦除扇区之前没有对 16 位位置进行多次编程，则**始终不会达到最大限制，因此不需要考虑该限制**。

如果执行 8 位（字节）编程操作，则必须考虑每条字线的最大编程限制，并且不能超过该限制。在 ECC 位置执行的编程操作与其他编程操作无关时将计入写入次数，直到需要进行擦除。

6.3.3.3 目标数据对齐（仅限使用单闪存字编程的器件）

对于仅支持单字编程的器件，只有 CMDDATA0、CMDDATA1 和 CMDECC0 寄存器用于加载要编程到闪存中的数据。CMDDATA0 中始终加载目标数据的 BIT31-BIT0，CMDDATA1 中始终加载目标数据的 BIT63-BIT32。如果直接指定且不自动计算 ECC 数据，则会将该数据加载到 CMDECC0 的 BIT7-BIT0 中。不使用其他 CMDDATAx 或 CMDECCx 寄存器，也不使用 CMDDATAINDEX。如果要编程的数据位少于 64 位，请参阅上面的特殊处理要求部分以了解如何编程少于一个闪存字。

单字编程操作必须采用闪存字（64 位）对齐方式。这意味着 CMDADDR 中指定的目标系统地址必须与 0b000 边界对齐（例如，CMDADDR 中的 3 个 LSB 必须为零）。

6.3.3.4 目标数据对齐（使用多字编程的器件）

对于支持 2 字、4 字或 8 字编程的器件，可以使用两种方法加载要编程的数据：直接模式或索引模式。编程者必须选择更适合应用要求的模式。

将数据加载到支持多字编程的器件上的 CMDDATAx 和 CMDECCx 寄存器时，还需满足其他对齐规则，即使不使用多字编程特性而仅使用单字编程。

- 单字编程操作必须使 CMDADDR（目标系统地址）与 0b000 边界对齐（例如，CMDADDR 中的 3 个 LSB 必须为零）。
- 双字编程操作必须使 CMDADDR（目标系统地址）与 0b0000 边界对齐（例如，CMDADDR 中的 4 个 LSB 必须为零）。
- 四字编程操作必须使 CMDADDR（目标系统地址）与 0b0.0000 边界对齐（例如，CMDADDR 中的 5 个 LSB 必须为零）。
- 八字编程操作必须使 CMDADDR（目标系统地址）与 0b00.0000 边界对齐（例如，CMDADDR 中的 6 个 LSB 必须为零）。

直接加载数据

为了配置可以直接加载数据的编程操作，应根据器件支持的闪存字数、目标地址对齐方式和目标数据大小将目标数据加载到相应的 CMDDATAx 寄存器中。例如，如果要在支持四字编程的器件上启动四字编程操作，则应在 CMDDATA0-CMDDATA7 寄存器中填充目标数据。如果直接指定 ECC（而不是由闪存控制器自动计算），则相应的 CMDECCx 寄存器中还需要填充每个要编程的数据字的 ECC 值。

索引式数据加载

可以只将 CMDDATA0-CMDDATA1 寄存器与索引寄存器 (CMDDATAINDEX) 结合使用来指示所加载数据的闪存字偏移量，而不必单独将数据缓冲到所有 CMDDATAx 寄存器中。通过这种方法，可以针对加载的每个字调整索引，并且每个目标数据字都可以写入到相同的 64 位空间 (CMDDATA0-CMDDATA1)。应用索引后，硬件会将加载的数据映射到相应的 CMDDATAx 寄存器中。例如，如果要在支持四字编程的器件上启动四字编程操作，则 CMDDATA1:0 寄存器将加载 4 次目标数据，并且在每个新字加载到 CMDDATA1:0 之前，CMDDATAINDEX 都将递增 1。

对齐规则

下表给出了每种器件配置（2、4 或 8 个字）的对齐规则，并说明如何在单字、双字、四字或八字编程操作中将目标数据放置在 CMDDATAx、CMDECCx 和 CMDDATAINDEX 寄存器中。

表 6-5. 支持 2 个闪存字编程的器件的数据加载对齐

直接加载寄存器	索引式加载索引	1 个字对齐到 0b000	2 个字对齐到 0b0000	4 个字对齐到 0b0.0000	8 个字对齐到 0b00.0000
CMDDATA1:0 / CMDECC0	CMDINDEX = 0	如果目标地址在 0b0000 中结束，则为目标数据字 0	目标数据字 0	不支持	不支持
CMDDATA3:2 / CMDECC1	CMDINDEX = 1	如果目标地址在 0b1000 中结束，则为目标数据字 0	目标数据字 1		

表 6-6. 支持 4 个闪存字编程的器件的数据加载对齐

直接加载寄存器	索引式加载索引	1 个字对齐到 0b000	2 个字对齐到 0b0000	4 个字对齐到 0b0.0000	8 个字对齐到 0b00.0000
CMDDATA1:0 / CMDECC0	CMDINDEX = 0	如果目标地址在 0b0.0000 中结束，则为目标数据字 0	如果目标地址在 0b0.0000 中结束，则为目标数据字 0	目标数据字 0	不支持
CMDDATA3:2 / CMDECC1	CMDINDEX = 1	如果目标地址在 0b0.1000 中结束，则为目标数据字 0	如果目标地址在 0b0.0000 中结束，则为目标数据字 1	目标数据字 1	
CMDDATA5:4 / CMDECC2	CMDINDEX = 2	如果目标地址在 0b1.0000 中结束，则为目标数据字 0	如果目标地址在 0b1.0000 中结束，则为目标数据字 0	目标数据字 2	
CMDDATA7:6 / CMDECC3	CMDINDEX = 3	如果目标地址在 0b1.1000 中结束，则为目标数据字 0	如果目标地址在 0b1.0000 中结束，则为目标数据字 1	目标数据字 3	

表 6-7. 支持 8 个闪存字编程的器件的数据加载对齐

直接加载寄存器	索引式加载索引	1 个字对齐到 0b000	2 个字对齐到 0b0000	4 个字对齐到 0b0.0000	8 个字对齐到 0b00.0000
CMDDATA1:0 / CMDECC0	CMDINDEX = 0	如果目标地址在 0b00.0000 中结束，则为目标数据字 0	如果目标地址在 0b00.0000 中结束，则为目标数据字 0	如果目标地址在 0b00.0000 中结束，则为目标数据字 0	目标数据字 0
CMDDATA3:2 / CMDECC1	CMDINDEX = 1	如果目标地址在 0b00.1000 中结束，则为目标数据字 0	如果目标地址在 0b00.0000 中结束，则为目标数据字 1	如果目标地址在 0b00.0000 中结束，则为目标数据字 1	目标数据字 1
CMDDATA5:4 / CMDECC2	CMDINDEX = 2	如果目标地址在 0b01.0000 中结束，则为目标数据字 0	如果目标地址在 0b01.0000 中结束，则为目标数据字 0	如果目标地址在 0b00.0000 中结束，则为目标数据字 2	目标数据字 2
CMDDATA7:6 / CMDECC3	CMDINDEX = 3	如果目标地址在 0b01.1000 中结束，则为目标数据字 0	如果目标地址在 0b01.0000 中结束，则为目标数据字 1	如果目标地址在 0b00.0000 中结束，则为目标数据字 3	目标数据字 3
CMDDATA9:8 / CMDECC4	CMDINDEX = 4	如果目标地址在 0b10.0000 中结束，则为目标数据字 0	如果目标地址在 0b10.0000 中结束，则为目标数据字 0	如果目标地址在 0b10.0000 中结束，则为目标数据字 0	目标数据字 4
CMDDATA11:10 / CMDECC5	CMDINDEX = 5	如果目标地址在 0b10.1000 中结束，则为目标数据字 0	如果目标地址在 0b10.0000 中结束，则为目标数据字 1	如果目标地址在 0b10.0000 中结束，则为目标数据字 1	目标数据字 5
CMDDATA13:12 / CMDECC6	CMDINDEX = 6	如果目标地址在 0b11.0000 中结束，则为目标数据字 0	如果目标地址在 0b11.0000 中结束，则为目标数据字 0	如果目标地址在 0b10.0000 中结束，则为目标数据字 2	目标数据字 6
CMDDATA15:14 / CMDECC7	CMDINDEX = 7	如果目标地址在 0b11.1000 中结束，则为目标数据字 0	如果目标地址在 0b11.0000 中结束，则为目标数据字 1	如果目标地址在 0b10.0000 中结束，则为目标数据字 3	目标数据字 7

6.3.3.5 执行 PROGRAM 操作

要对闪存进行编程，请执行以下操作：

1. 在 CMDTYPE 寄存器中选择命令：
   a. 将 CMDTYPE 寄存器中的 COMMAND 字段设置为 PROGRAM。

b. 将 CMDTYPE 寄存器中的 SIZE 字段设置为所需的大小 (1、2、4 或 8 个闪存字) 。如果器件不支持多字编程，请选择 ONEWORD。如果器件支持多字编程，并且需要多字编程，请选择小于或等于目标器件所支持最大大小的所需值。硬件不会检查 SIZE 字段的配置是否无效；软件必须确保器件支持选择选项。请注意，SECTOR 和 BANK 大小不适用于 PROGRAM 操作，只适用于擦除操作。

2. 在 CMDCTL 寄存器中配置编程命令：
   a. 在具有 ECC 的器件上，默认情况下，闪存控制器将在 PROGRAM 操作期间从数据中生成所需的 ECC 位。或者，软件可以覆盖硬件 ECC 代码生成，并通过设置 CMDCTL 寄存器中的 ECCGENOVR 位手动提供待编程的 ECC 代码。
3. 在 CMDADDR 和 CMDBYTEN 寄存器中选择目标编程地址：
   a. 将目标系统地址加载到 CMDADDR 寄存器中，以指示开始编程的基地址。目标地址必须是一个闪存字地址 (64 位对齐) 。闪存控制器会将系统地址转换为适用的闪存区域、存储体 ID 和存储体地址。如果需要，操作完成后，可以从 STATADDR 寄存器中读取闪存区域、存储体 ID 和存储体地址。在多字编程中，STATADDR 将指示存储体 ID 和编程的最终地址。
   b. 如果需要进行子字编程 (少于完整的 64 或 72 位闪存字的编程) ，请配置 CMDBYTEN 寄存器，以设置已寻址闪存字内待编程的字节。CMDBYTEN 中的每个位对应于已寻址闪存字中待编程的一个字节，包括 ECC 字节。例如，在对闪存字的数据字节进行编程时，可以通过将 CMDBYTEN 中的位 8 清零来屏蔽 ECC 代码的编程。请注意，在必须应用扇区擦除之前，每个字线都有允许的最大编程操作数 (有关最大数量，请参阅器件特定数据表) 。
4. 将待编程的数据加载到 CMDDATAx 寄存器中：
   a. 对于单个闪存字编程操作 (64 位或 72 位，具体取决于是否存在 ECC) ，按照对齐要求将数据加载到 CMDDATAx 寄存器中 (对于仅支持单字编程的器件，无论目标地址如何，始终使用 CMDDATA0 和 CMDDATA1) 。
   b. 对于多字编程 (如果可用且已选择) ，根据第 1 步中指定的对齐规则和多字编程操作数，将数据加载到 CMDDATAx 寄存器中。
   c. 如按上述设置了 CMDCTL 寄存器中的 ECCGENOVR (禁用硬件 ECC 代码生成) ，则将 ECC 数据写入 CMDDATAECC0 寄存器 (对于单字编程) ，也可以根据需要写入其他 CMDDATAECCx 寄存器 (对于多字编程) 。
   d. 请注意，在编程操作期间，CMDDATA 寄存器被闪存控制器用作位屏蔽寄存器；操作完成后，写入这些寄存器的值将被闪存控制器覆盖。
5. 确保将写保护方案配置为允许写入目标地址 (有关配置写保护的其他信息，请参阅本指南的写保护部分) 。
6. 通过向 CMDEXEC 寄存器写入 0x1 来执行编程操作。
7. 监控完成编程操作：
   a. 可以轮询 STATCMD 寄存器以确定编程操作的状态。一旦命令启动，将由硬件设置 CMDINPROGRESS 位。操作终止后，将设置 CMDDONE 位。
   b. 设置 CMDDONE 后，CMDPASS 位将同时复位或置位，用于指示该操作是成功完成还是失败。如果在受保护区域尝试编程，则使 FAILWEPROT 位有效。如果编程操作无法在最大编程脉冲数限制范围内成功完成，则将使 FAILVERIFY 有效。有关最长编程时间，请参阅器件特定数据表。
8. 完成编程操作后，闪存控制器将配置多个设置：
   a. 所有动态写保护寄存器均设置为受保护状态 (以防止意外编程)
   b. 所有数据寄存器均设置为 1。
   c. 所有编程字节使能均清除为 0。
9. 对闪存进行编程后，处理器的高速缓存和预取逻辑中可能存在过时的数据。在读取已编程的位置之前，建议先刷新 CPU 子系统中的高速缓存。

6.3.4 ERASE 命令

擦除命令用于擦除闪存的各个扇区 (针对 MAIN、NONMAIN 或 DATA 区域) 或整个存储体 (仅针对 MAIN 区域) 。从这个已擦除状态中，稍后可以使用 PROGRAM 命令根据需要将位编程为“0”状态或“1”状态。在扇区对齐的情况下，无法以低于一个扇区 (1KB) 的分辨率进行擦除。对于具有多个存储体的器件，必须对所有存储体执行存储体擦除，以擦除器件上的整个 MAIN 区域。

备注

擦除之后，扇区的存储器内容在编程之前并不确定。擦除后，擦除的位并不总是读为 1。必须先使用 PROGRAM 命令对存储器位置成功编程，然后才能将存储器位置视为确定性。

6.3.4.1 擦除扇区屏蔽行为

闪存控制器提供了一种擦除验证机制来延长闪存存储体的使用寿命。CMDWEPROTx 寄存器用作擦除掩码，并在执行擦除操作期间由闪存控制器处理。在所有擦除操作结束时，CMDWEPROTx 寄存器设置为完全受保护状态以防止意外编程，并且在尝试另一次编程或擦除操作之前，必须对其重新配置。

6.3.4.2 执行 ERASE 操作

要擦除闪存的一个扇区或存储体，请执行以下步骤：

1. 在 CMDTYPE 寄存器中选择命令：
   a. 将 CMDTYPE 寄存器中的 COMMAND 字段设置为 ERASE。
   b. 将 CMDTYPE 寄存器的 SIZE 字段设置为 SECTOR 或 BANK。ERASE 命令不支持 SECTOR 或 BANK（例如，ONEWORD）以外的大小。在发出 ERASE 命令之前，软件负责检查配置。
2. 在 CMDADDR 寄存器中选择目标擦除地址：
   a. 将目标系统地址存储到 CMDADDR 寄存器中，以指示要擦除的扇区或存储体的基地址。在启用写保护的情况下执行存储体擦除（仅擦除未受保护的扇区）时，请确保写入 CMDADDR 的地址位于未受保护的扇区中。闪存控制器会将系统地址转换为适用的闪存区域、存储体 ID 和存储体地址。如果需要，操作完成后，可以从 STATADDR 寄存器中读取闪存区域、存储体 ID 和存储体地址。
3. 确保将写保护方案配置为允许写入目标扇区（有关配置写保护的其他信息，请参阅本指南的写保护部分）。
4. 通过向 CMDEXEC 寄存器写入 0x1 来执行擦除操作。
5. 监控擦除操作是否完成：
   a. 可以轮询 STATCMD 寄存器以确定擦除操作的状态。一旦命令启动，将由硬件设置 CMDINPROGRESS 位。操作终止后，将设置 CMDDONE 位。设置 CMDDONE 后，CMDPASS 位将同时复位或置位，用于指示该操作是成功完成还是失败。
   b. 如果在受保护区域尝试擦除，则使 FAILWEPROT 位有效。
   c. 如果擦除操作无法在最大擦除脉冲数限制范围内成功完成，则使 FAILVERIFY 有效。
6. 完成擦除操作后，闪存控制器将配置几个设置来防止意外编程：
   a. 所有动态写保护寄存器均设置为受保护状态。

6.3.5 READVERIFY 命令

READVERIFY 命令是指读取验证命令，可用于读取闪存位置，并将读取到的数据与预加载到闪存控制器 CMDDATA 寄存器中的数据进行比较。该命令可应用于单个闪存字、多个闪存字（如果器件支持多字编程）、整个扇区或整个存储体。对整个扇区或存储体执行读取验证时，将重用 CMDDATAx 中的数据。

6.3.5.1 执行 READVERIFY 操作

要执行读取验证命令，请执行以下步骤：

1. 在 CMDTYPE 寄存器中选择命令：
   a. 将 CMDTYPE 寄存器中的 COMMAND 字段设置为 READVERIFY。
   b. 将 CMDTYPE 寄存器中的 SIZE 字段设置为所需的大小。
2. 在 CMDCTL 寄存器中配置读取验证命令：
   a. 如果需要手动提供 ECC 位和数据，请设置 CMDCTL 寄存器中的 ECCGENOVR 位。如果 ECCGENOVR 清零，闪存控制器将根据提供的比较数据生成 ECC 位以进行比较。
3. 在 CMDADDR 寄存器中选择要验证的目标地址：
   a. 将目标系统地址加载到 CMDADDR 寄存器中，以指示要验证的基地址。闪存控制器会将系统地址转换为适用的闪存区域、存储体 ID 和存储体地址。如果需要，操作完成后，可以从 STATADDR 寄存器中读取闪存区域、存储体 ID 和存储体地址。
4. 将待验证的数据加载到 CMDDATAx 寄存器中：

a. 对于单字验证，请将待验证的数据写入 CMDDATA0 和 CMDDATA1 寄存器。对于多字验证，如果目标器件上提供多字验证，请将待验证的数据写入 CMDDATA0 和 CMDDATA1 之外的相应 CMDDATAx 寄存器。

5. 在 CMDBYTEN 寄存器中配置字节使能设置：
   a. 任何设置为逻辑“0”的 CMDBYTEN 位都将屏蔽相关数据字节，使其在执行 READVERIFY 命令期间无法进行比较。这可用于验证小于一个闪存字 (少于 64 位) 的数据。
6. 通过向 CMDEXEC 寄存器写入 0x1 来执行读取验证操作。
7. 监控是否完成读取验证操作：
   a. 可以轮询 STATCMD 寄存器以确定擦除操作的状态。一旦命令启动，将由硬件设置 CMDINPROGRESS 位。操作终止后，将设置 CMDDONE 位。设置 CMDDONE 后，CMDPASS 位将同时复位或置位，以指示读取验证是否通过。
   b. 如果从闪存读取的任何数据与 CMDDATAx 中加载的预期数据不匹配，则将设置 STATCMD 中的 FAILVERIFY 位。
8. 完成读取验证操作后，闪存控制器将配置多个设置：
   a. 所有动态写保护寄存器均设置为受保护状态 (以防止意外编程)。
   b. 所有数据寄存器均设置为“1”。
   c. 所有编程字节使能均清除为“0”。

6.3.6 BLANKVERIFY 命令

应用软件可以使用空白验证 (BLANKVERIFY) 命令来验证闪存字是否为空白。空白闪存字定义为已使用 ERASE 命令成功擦除且尚未使用 PROGRAM 命令编程到撤出该未擦除状态的闪存字。

擦除后，在使用 PROGRAM 命令对闪存字进行编程之前，闪存字不处于确定性状态。这意味着应用软件不能期望擦除后被擦除的位读回为“1”。必须使用 PROGRAM 命令成功地对存储器位置进行编程，然后才能将该存储器位置的读取视为确定性的，并由应用软件使用。

由于无法通过直接读取该位置来确定闪存字是否处于已擦除状态 (因为被擦除的位置在读取时将返回非确定性数据)，因此可使用 BLANKVERIFY 命令来测试闪存字是否处于空白状态，这种状态表示尚未将其编程为撤出已擦除状态。

BLANKVERIFY 命令一次只能应用于单个闪存字。

6.3.6.1 执行 BLANKVERIFY 操作

要执行空白验证操作，请执行以下步骤：

1. 在 CMDTYPE 寄存器中选择命令：
   a. 将 CMDTYPE 寄存器中的 COMMAND 字段设置为 BLANKVERIFY。
   b. 将 CMDTYPE 寄存器中的 SIZE 字段设置为一个字。
2. 在 CMDADDR 寄存器中选择要验证的目标地址：
   a. 将目标系统地址存储到 CMDADDR 寄存器中，以指示要验证的基地址。闪存控制器会将系统地址转换为适用的闪存区域、存储体 ID 和存储体地址。如果需要，操作完成后，可以从 STATADDR 寄存器中读取闪存区域、存储体 ID 和存储体地址。指定闪存字中的所有 8 个数据字节以及相应的 ECC 字节将由 BLANKVERIFY 进行检查。
3. 通过向 CMDEXEC 寄存器写入 0x1 来执行空白验证操作。
4. 监控是否完成空白验证操作：
   a. 可以轮询 STATCMD 寄存器以确定擦除操作的状态。一旦命令启动，将由硬件设置 CMDINPROGRESS 位。操作终止后，将设置 CMDDONE 位。设置 CMDDONE 后，CMDPASS 位将同时复位或置位，以指示空白验证是否通过。
   b. 如果未擦除闪存位置，则将设置 STATCMD 中的 FAILVERIFY 位。
5. 完成空白验证操作后，闪存控制器将配置多个设置：
   a. 所有动态写保护寄存器均设置为受保护状态 (以防止意外编程)
   b. 所有数据寄存器均设置为“1”。
   c. 所有编程字节使能均清除为“0”。

6.3.7 命令诊断

闪存控制器会更新几个软件可读寄存器来传达有关已启动操作的信息。

6.3.7.1 状态命令

STATCMD 寄存器是一个只读寄存器，提供有关已启动或已完成操作的诊断信息。CMDINPROGRESS 位表示当前正在进行某个操作。CMDDONE 位表示某操作已经完成。这些位可由软件轮询来确定操作期间闪存控制器的状态。

6.3.7.2 地址转换

STATADDR 寄存器是一个只读寄存器，可通过读取该寄存器来确定闪存控制器指向的当前存储体 ID、区域 ID 和存储体地址。在某些命令执行期间，这些值会递增，在这种情况下，命令完成后出现的值表示闪存控制器触及的最后一个地址。

6.3.7.3 脉冲计数

STATPCNT 寄存器是只读寄存器，可以通过读取该寄存器来确定在编程或擦除操作期间应用的当前脉冲计数。

6.3.8 使用存储体 ID、区域 ID 和存储体地址覆盖系统地址

通常，闪存控制器命令通过将系统存储器映射地址加载到 CMDADDR 寄存器中来定位到特定的闪存位置。推荐采用此方法为 PROGRAM、ERASE、READVERIFY 或 BLANKVERIFY 命令指定目标地址。在此工作模式下，闪存控制器会自动将系统地址转换为相应的存储体 ID、区域 ID 和存储体地址，这些信息用于对闪存执行命令。应用软件不需要单独指定这些项目；只需要系统地址。

但是，在某些情况下，可能需要直接指定目标闪存存储体、区域以及该指定存储体/区域中的地址。例如，如果希望在对器件执行批量擦除操作时擦除整个存储体，应用软件实际上不需要知道要擦除的存储体的系统地址，只需要为闪存控制器指定存储体 ID 和区域 ID 来擦除存储体。

要使用闪存控制器在地址覆盖模式下执行命令，请设置 CMDCTL 寄存器中的 ADDRXLATEOVR 位，并在执行命令之前指定存储体 ID、区域和存储体地址。要回到系统寻址模式，请清除 ADDRXLATEOVR。默认会清除 ADDRXLATEOVR (支持系统地址操作) 。

示例 - 使用 ADDRXLATEOVR 擦除存储体

要通过指定存储体 ID 和区域而不是系统地址来擦除 BANK0 的 MAIN 区域，请执行节 6.3.4.2 中的步骤，但将其中的步骤 2 替换为下面给出的替代步骤：

1. 设置 CMDCTL 中的 ADDRXLATEOVR 位以启用地址转换覆盖模式
2. 通过将 CMDCTL 寄存器中的 BANKSEL 字段设置为 0x1 来指定 BANK0
3. 通过将 CMDCTL 寄存器的 REGIONSEL 字段设置为 0x1 来指定 MAIN 区域
4. 将 CMDADDR 寄存器设置为 0x0000.0000

6.3.9 FLASHCTL 事件

闪存控制器包含一个事件发布者，不包含事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理到 CPU 子系统的 FLASHCTL 中断请求 (IRQ)。

表 6-8 总结了 FLASHCTL 事件。

表 6-8. FLASHCTL 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	FLASHCTL	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 FLASHCTL 到 CPU 的中断路由

6.3.9.1 CPU 中断事件发布者

FLASHCTL 模块提供一个中断源，可配置为产生 CPU 中断事件。表 6-9 中给出了 FLASHCTL 中断条件。

表 6-9. FLASHCTL CPU 中断条件

索引 (IIDX)	名称	说明
0	DONE	表示 FLASHCTL 操作已完成

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关配置这些寄存器的指导，请参阅节 7.2.5。

6.4 写保护

闪存控制器提供两个写保护机制 (一个为静态，另一个为动态) ，这些机制并行 (逻辑“或”) 用于在任何尝试的编程或擦除操作期间保护用户指定的扇区。如果对受写保护的闪存扇区发出编程或擦除操作，该操作将终止，并在 STATCMD 寄存器中报告 FAILWEPROT 错误。

6.4.1 写保护分辨率

静态和动态写保护机制的写保护分辨率取决于受保护的闪存存储体和存储器区域。

表 6-10. 按区域划分的写保护分辨率

闪存存储体	存储器区域	写保护分辨率
0	NONMAIN	512B (整个区域)
	MAIN	前 32KB (32 个扇区) 为 1KB (1 个扇区) 剩余扇区为 8KB (8 个扇区)
1-4	MAIN	8KB (8 个扇区)
	数据	1KB (1 个扇区)

6.4.2 静态写保护

不可变的 ROM 引导代码必须在器件引导期间配置并锁存静态写保护方案，然后才能执行闪存中的主应用程序。通过静态写保护机制将闪存扇区配置为受写保护后，除非重新启动，否则无法在运行时通过软件取消该保护。因此，受静态写保护的扇区可被视为在引导后不可变。这种类型的保护非常适合用于保护双映像应用程序中的自定义引导加载程序，或者将 ROM 引导代码中的安全信任根扩展到主闪存区域的某个部分，以便使用锁定的公共密钥来启用安全启动管理器。

配置静态写保护方案的方法是对 NONMAIN 闪存区域中的相应位进行编程，这样便可在主应用程序启动前由引导代码读取相应的位。此外，还可以对 NONMAIN 闪存扇区本身进行静态写保护，从而获得一种完全永久且不能修改的静态写保护机制。如果 NONMAIN 扇区连同任何其他闪存扇区一起被配置为受静态保护，则所有受静态保护的闪存扇区均可在功能上视为只读存储器，因此无法以任何方式对其进行更新。

器件所有存储体上的所有闪存扇区均可配置静态写保护。本文档的节 1.4 提供了关于配置静态写保护以及 NONMAIN 区域其余部分的说明。

6.4.3 动态写保护

动态写保护方案旨在通过软件在运行时进行配置。该方案为应用软件提供了一种简单的方法来指定扇区，以防止被闪存控制器发出的任何编程或擦除操作修改。与静态写保护机制不同，动态机制不可锁定，因此不提供任何级别的数据安全性。

动态写保护有两个主要用途。首先，它进一步提高了稳健性，可防止应用中涉及固件更新或 EEPROM 仿真的系统内编程的指定扇区被意外编程或擦除。其次，它提供了一种方法来简化需要存储体擦除的情况，但在发出存储体擦除命令时，有少量扇区不应擦除。一个示例是在单存储体器件上运行的应用，其中大多数 MAIN 区域扇区用于存储可执行映像，但少数扇区用于存储器件特定数据，固件更新期间不应擦除这些数据。在这种情况下，可通过动态写保护来保护包含器件特定数据的扇区，并可发出存储体擦除命令来擦除所有其他扇区。这样做的好处是，使用单个命令 (存储体擦除) 就可以擦除大部分 MAIN 区域，而不是使用各项逐扇区命令 (这些命令具有更长的整体擦除时间并使用更多的能量) 。

通过在闪存控制器中设置 CMDWEPROTx 寄存器来配置动态写保护方案。CMDWEPROTx 寄存器一次覆盖一个闪存存储体。这意味着，必须在知道将对哪个闪存存储体尝试应用编程或擦除操作的情况下，对这些寄存器进行

配置。请注意，在所有编程和擦除操作结束时，CMDWEPROTx 寄存器将复位至一个受保护的状态。启动新操作之前，必须通过软件重新配置这些寄存器。

6.4.3.1 为 MAIN 区域配置保护

CMDWEPROTA 寄存器用于配置 BANK0 MAIN 区域的前 32 个扇区 (32KB) 的动态写保护。每个位对应于主区域的一个扇区，从 MAIN 区域的开头开始。CMDWEPROTA 仅适用于对 BANK0 的 MAIN 区域的下部 32 个扇区（扇区 0-31）的操作。在应用于其他扇区的编程/擦除操作期间，不使用它。

CMDWEPROTB 寄存器也用于配置 MAIN 区域的动态写保护。有两种应用 CMDWEPROTB 的模式，具体取决于编程/擦除操作是应用到 BANK0 还是 BANK1-4。在对 BANK0 进行编程/擦除操作时，CMDWEPROTB 从 BIT4 开始以 8 扇区增量（每 8KB 1 位）保护扇区 32-255。CMDWEPROTB 中的 BIT0-BIT3 会被忽略。下部 32 个扇区（BANK0 的 0-31 扇区）受 CMDWEPROTA 保护。在对 BANK1-4 进行编程/擦除操作时，CMDWEPROTB 从 CMDWEPROTB 中的 BIT0 开始，以 8 扇区增量（每 8 扇区 1 位）保护扇区 0-255。

CMDWEPROTC 寄存器用于配置 MAIN 区域的动态写保护。对于 BANK0-4，CMDWEPROTC 以 8 扇区增量（每 8 个扇区 1 位）保护扇区 256-511。

6.4.3.2 为 NONMAIN 区域配置保护

CMDWEPROTNM 寄存器可保护 NONMAIN 区域不受编程和擦除的影响。每个扇区提供一个保护位。器件只有一个 NONMAIN 扇区，它将位于 BANK0 中。

6.5 读取接口

读取接口提供 CPU 子系统的读取路径（用于提取指令/数据）、外设总线的读取路径（供 DMA 控制器或 CPU 使用）、主存储体地址交换以及针对 ECC SEC 或 DED 错误的检测和报告。

6.5.1 存储体地址交换

包含多个存储体的器件支持地址空间内存储体的 MAIN 区域交换。这种机制可将两个版本的应用固件编程到器件中，而无需固件知道其所在的物理存储体。

表 6-11 提供了为一个器件请求存储体交换前后的映射示例，此器件具有 512KB 主闪存，拆分成 2 个存储体（每个为 256KB）。

表 6-11. 存储体地址交换转换

存储体和区域	交换之前的地址空间	交换后的地址空间
BANK0 MAIN	0x0000.0000 – 0x0003.FFFF	0x0004.0000 – 0x0007.FFFF
BANK1 MAIN	0x0004.0000 – 0x0007.FFFF	0x0000.0000 – 0x0003.FFFF

器件复位后，下部存储体的 MAIN 区域始终映射到最低地址空间。应用软件负责确定是否应用存储体地址交换。存储体地址交换控制包含在 SYSCTL 模块中；请参阅 SYSCTL 一章，了解寄存器和位定义。在地址交换期间，应用软件必须满足以下限制条件：

1. 软件必须在发出存储体交换命令之前禁用中断。
2. 软件必须在发出存储体交换命令后轮询存储体交换状态，然后才能继续执行。
3. 如果交换命令和轮询状态例程从闪存执行，它们必须位于两个存储体中的同一个位置，以便在存储体交换后从中断的位置恢复执行。如果存储体交换和状态轮询是从 SRAM（而不是闪存）完成的，则此限制不适用。

6.5.2 ECC 错误处理

读取接口纠正 64 位闪存字中的一位错误 (SEC)，并检测 64 位闪存字中的双位错误 (DED)。对于“全 1”方案或“全 0”方案，将忽略 ECC 检查。

6.5.2.1 single-bit（可纠正）错误

在将请求的数据返回到 CPU 子系统或外设总线 (DMA) 之前，将自动纠正 single-bit 错误。不报告由于调试访问而发生的错误。

6.5.2.2 双位 (不可纠正) 错误

双位错误可检测但不可纠正，并指示给 SYSCTL。SYSCTL 可配置为生成不可屏蔽中断 (NMI) 或复位，因为不可纠正的错误可能是致命错误。不报告由于调试访问而发生的错误。

6.6 FLASHCTL 寄存器

表 6-12 列出了 FLASHCTL 寄存器的存储器映射寄存器。表 6-12 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 6-12. FLASHCTL 寄存器

偏移	缩写	寄存器名称	组	部分
1020h	IIDX	中断索引寄存器		转到
1028h	IMASK	中断屏蔽寄存器		转到
1030h	RIS	原始中断状态寄存器		转到
1038h	MIS	屏蔽中断状态寄存器		转到
1040h	ISET	中断集寄存器		转到
1048h	ICLR	中断清除寄存器		转到
1100h	CMDEXEC	命令执行寄存器		转到
1104h	CMDTYPE	命令类型寄存器		转到
1108h	CMDCTL	命令控制寄存器		转到
1120h	CMDADDR	命令地址寄存器		转到
1124h	CMDBYTEN	命令编程字节使能寄存器		转到
112Ch	CMDDATAINDEX	命令数据索引寄存器		转到
1130h	CMDDATA0	命令数据寄存器 0		转到
1134h	CMDDATA1	命令数据寄存器 1		转到
1138h	CMDDATA2	命令数据寄存器 2		转到
113Ch	CMDDATA3	命令数据寄存器位 127:96		转到
1140h	CMDDATA4	命令数据寄存器 4		转到
1144h	CMDDATA5	命令数据寄存器 5		转到
1148h	CMDDATA6	命令数据寄存器 6		转到
114Ch	CMDDATA7	命令数据寄存器 7		转到
1150h	CMDDATA8	命令数据寄存器 8		转到
1154h	CMDDATA9	命令数据寄存器 9		转到
1158h	CMDDATA10	命令数据寄存器 10		转到
115Ch	CMDDATA11	命令数据寄存器 11		转到
1160h	CMDDATA12	命令数据寄存器 12		转到
1164h	CMDDATA13	命令数据寄存器 13		转到
1168h	CMDDATA14	命令数据寄存器 14		转到
116Ch	CMDDATA15	命令数据寄存器 15		转到
1170h	CMDDATA16	命令数据寄存器 16		转到
1174h	CMDDATA17	命令数据寄存器 17		转到
1178h	CMDDATA18	命令数据寄存器 18		转到
117Ch	CMDDATA19	命令数据寄存器 19		转到
1180h	CMDDATA20	命令数据寄存器 20		转到
1184h	CMDDATA21	命令数据寄存器 21		转到
1188h	CMDDATA22	命令数据寄存器 22		转到
118Ch	CMDDATA23	命令数据寄存器 23		转到
1190h	CMDDATA24	命令数据寄存器 24		转到
1194h	CMDDATA25	命令数据寄存器 25		转到
1198h	CMDDATA26	命令数据寄存器 26		转到
119Ch	CMDDATA27	命令数据寄存器 27		转到

表 6-12. FLASHCTL 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
11A0h	CMDDATA28	命令数据寄存器 28		转到
11A4h	CMDDATA29	命令数据寄存器 29		转到
11A8h	CMDDATA30	命令数据寄存器 30		转到
11ACh	CMDDATA31	命令数据寄存器 31		转到
11B0h	CMDDATAECC0	命令数据寄存器 ECC 0		转到
11B4h	CMDDATAECC1	命令数据寄存器 ECC 1		转到
11B8h	CMDDATAECC2	命令数据寄存器 ECC 2		转到
11BCh	CMDDATAECC3	命令数据寄存器 ECC 3		转到
11C0h	CMDDATAECC4	命令数据寄存器 ECC 4		转到
11C4h	CMDDATAECC5	命令数据寄存器 ECC 5		转到
11C8h	CMDDATAECC6	命令数据寄存器 ECC 6		转到
11CCh	CMDDATAECC7	命令数据寄存器 ECC 7		转到
11D0h	CMDWEPROTA	命令写擦除保护 A 寄存器		转到
11D4h	CMDWEPROTB	命令写擦除保护 B 寄存器		转到
11D8h	CMDWEPROTC	命令写擦除保护 C 寄存器		转到
1210h	CMDWEPROTNM	命令写擦除保护非主寄存器		转到
13B4h	CFGPCNT	脉冲计数器配置寄存器		转到
13D0h	STATCMD	命令状态寄存器		转到
13D4h	STATADDR	地址状态寄存器		转到
13D8h	STATPCNT	脉冲计数状态寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 6-13 显示了适用于此部分中访问类型的代码。

表 6-13. FLASHCTL 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

6.6.1 IIDX (偏移 = 1020h) [复位 = 00000000h]

图 6-4 展示了 IIDX，表 6-14 中对此进行了介绍。

返回到汇总表。

中断索引 (IIDX) 寄存器提供最高优先级挂起和已启用中断的索引。

图 6-4. IIDX

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

表 6-14. IIDX 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	保留
0	STAT	R	0h	对应于最高优先级挂起中断源的索引。该值可用作中断服务例程中快速、确定性处理的地址偏移量。读取 IIDX 寄存器将清除 RIS 和 MIS 寄存器中相应的中断状态。 0h (R/W) = 无中断挂起 1h (R/W) = DONE 中断挂起

6.6.2 IMASK (偏移 = 1028h) [复位 = 00000000h]

图 6-5 展示了 IMASK，表 6-15 中对此进行了介绍。

返回到汇总表。

中断屏蔽 (IMASK) 寄存器保持当前中断屏蔽设置。

图 6-5. IMASK

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

表 6-15. IMASK 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	保留
0	DONE	R/W	0h	启用或禁用 DONE 中断。 0h (R/W) = 中断被屏蔽 1h (R/W) = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置

6.6.3 RIS (偏移 = 1030h) [复位 = 00000000h]

图 6-6 展示了 RIS，表 6-16 中对此进行了介绍。

返回到汇总表。

原始中断状态 (RIS) 寄存器保持当前的原始中断状态。

图 6-6. RIS

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

表 6-16. RIS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	保留
0	DONE	R	0h	DONE 中断的原始状态。 0h (R/W) = 未发生中断 1h (R/W) = 已发生中断

6.6.4 MIS (偏移 = 1038h) [复位 = 00000000h]

图 6-7 展示了 MIS，表 6-17 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态 (MIS) 寄存器保持当前屏蔽中断状态。

图 6-7. MIS

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

表 6-17. MIS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	保留
0	DONE	R	0h	DONE 中断的屏蔽状态。 0h (R/W) = 未发生屏蔽中断 1h (R/W) = 已发生屏蔽中断

6.6.5 ISET (偏移 = 1040h) [复位 = 00000000h]

图 6-8 展示了 ISET，表 6-18 中对此进行了介绍。

返回到汇总表。

中断设置 (ISET) 寄存器可用于将中断设置为软件挂起。

图 6-8. ISET

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

表 6-18. ISET 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	保留
0	DONE	W	0h	设置 DONE 中断。 0h (R/W) = 写入 0 没有任何影响 1h (R/W) = 设置 RIS 位

6.6.6 ICLR (偏移 = 1048h) [复位 = 00000000h]

图 6-9 展示了 ICLR，表 6-19 中对此进行了介绍。

返回到汇总表。

中断清除 (ICLR) 寄存器可用于清除挂起的中断。

图 6-9. ICLR

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

表 6-19. ICLR 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	保留
0	DONE	W	0h	清除 DONE 中断。 0h (R/W) = 写入 0 没有任何影响 1h (R/W) = 将 RIS 位清零

6.6.7 CMDEXEC (偏移 = 1100h) [复位 = 00000000h]

图 6-10 展示了 CMDEXEC，表 6-20 中对此进行了介绍。

返回到汇总表。

命令执行寄存器
启动 CMDTYPE 寄存器中指定命令的执行。在写入 1 之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。硬件在完成命令处理后清除此寄存器。

图 6-10. CMDEXEC

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

表 6-20. CMDEXEC 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	保留
0	值	R/W	0h	命令执行值启动 CMDTYPE 寄存器中指定命令的执行。 0h (R/W) = 命令在硬件中不执行 1h (R/W) = 命令在硬件中执行

6.6.8 CMDTYPE (偏移 = 1104h) [复位 = 00000000h]

图 6-11 展示了 CMDTYPE，表 6-21 中对此进行了介绍。

返回到汇总表。

命令类型寄存器
指定硬件要执行的命令类型。在将 CMDEXEC 写入 1 之后，以及在硬件设置 STATCMD.DONE 以指示命令执行已完成之前，阻止此寄存器写入。

图 6-11. CMDTYPE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	尺寸			保留	命令		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		

表 6-21. CMDTYPE 字段说明

位	字段	类型	复位	说明
31-7	保留	R/W	0h	保留
6-4	尺寸	R/W	0h	命令大小 0h (R/W) = 在 1 个闪存字上运行 1h (R/W) = 在 2 个闪存字上运行 2h (R/W) = 在 4 个闪存字上运行 3h (R/W) = 在 8 个闪存字上运行 4h (R/W) = 在闪存扇区上运行 5h (R/W) = 在整个闪存存储体上运行
3	RESERVED	R/W	0h	保留
2-0	命令	R/W	0h	命令类型 0h (R/W) = 无操作 1h (R/W) = 编程 2h (R/W) = 擦除 3h (R/W) = 读取验证 - 执行独立的读取验证操作。 6h (R/W) = 空白验证 - 检查闪存字是否处于已擦除状态。此命令只能与 CMDTYPE.SIZE = ONEWORD 一起使用

6.6.9 CMDCTL (偏移 = 1108h) [复位 = 00000000h]

图 6-12 展示了 CMDCTL，表 6-22 中对此进行了介绍。

返回到汇总表。

命令控制寄存器 此寄存器配置状态机与命令执行相关的特定功能。在将 CMDEXEC 写入 1 之后，以及在硬件设置 STATCMD.DONE 以指示命令执行已完成之前，阻止此寄存器写入。

图 6-12. CMDCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED		RESERVED	SSERASEDIS	保留		ECCGENOVR	ADDRXLATEOVR
R/W-0h		R/W-	R/W-0h	R/W-		R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
保留		RESERVED	REGIONSEL				RESERVED
R/W-		R/W-0h	R/W-0h				R/W-
7	6	5	4	3	2	1	0
RESERVED			BANKSEL	RESERVED			
R/W-			R/W-0h	R/W-			

表 6-22. CMDCTL 字段说明

位	字段	类型	复位	说明
31-22	保留	R/W	0h	保留
21	保留	R/W	0h	
20	SSERASEDIS	R/W	0h	禁用阶梯擦除。如已设置，则所有擦除脉冲将使用默认的 VHV 修整电压设置。默认情况下，此位已复位，这意味着 VHV 电压将在连续擦除脉冲期间发生阶跃。阶跃数、阶跃电压、开始电压和结束电压均为硬接线。 0h (R/W) = 启用 1h (R/W) = 禁用
19-18	保留	R/W	0h	
17	ECCGENOVR	R/W	0h	覆盖编程 ECC 数据的硬件生成。使用写入到 CMDDATAECC* 的数据。 0h (R/W) = 不覆盖 1h (R/W) = 覆盖
16	ADDRXLATEOVR	R/W	0h	覆盖 CMDADDR 中的地址从系统地址到相应存储体地址和存储体 ID 的硬件地址转换。设置后，CMDADDR 将直接用作存储体地址，CMDCTL.REGIONSEL 将直接用作区域 ID，而 CMDCTL.BANKSEL 将直接用作存储体 ID (如果该器件包含多个存储体)。 0h (R/W) = 不覆盖 1h (R/W) = 覆盖
15-14	RESERVED	R/W	0h	
13	RESERVED	R/W	0h	被保留
12-9	REGIONSEL	R/W	0h	存储体区域 如果设置了 CMDCTL.ADDRXLATEOVR，则可以向此字段写入特定区域 ID，以指示应将操作应用于哪个区域。 1h (R/W) = 主区域 2h (R/W) = 非主区域
8-5	RESERVED	R/W	0h	

表 6-22. CMDCTL 字段说明 (continued)

位	字段	类型	复位	说明
4	BANKSEL	R/W	0h	存储体选择 如果设置了 CMDCTL.ADDRXLATEOVR，则可以向此字段写入特定存储体 ID，以指示应将操作应用于哪个存储体。 1h (R/W) = 存储体 0 2h (R/W) = 存储体 1 4h (R/W) = 存储体 2 8h (R/W) = 存储体 3 10h (R/W) = 存储体 4
3-0	RESERVED	R/W	0h	

6.6.10 CMDADDR (偏移 = 1120h) [复位 = 00000000h]

图 6-13 展示了 CMDADDR，表 6-23 中对此进行了介绍。

返回到汇总表。

命令地址寄存器：
此寄存器构成命令的目标地址。用例如下：
1) 对于单字编程，此地址指示要编程的闪存存储体字。
2) 对于多字编程，此地址指示编程的第一个闪存存储体地址。对于后续字，该地址会递增。
3) 对于扇区擦除，此地址指示要擦除的扇区。
4) 对于存储体擦除，该地址指示要擦除的存储体。
5) 对于读取验证，该地址指示遵循上面列出的编程/擦除。
请注意，写入此寄存器的地址将提交到闪存地址转换接口用于转换，转换后的地址将用于访问存储体。但是，如果设置了 CMDCTL.ADDRXLATEOVR 位，那么写入此寄存器的地址将直接用作存储体地址。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。

图 6-13. CMDADDR

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

表 6-23. CMDADDR 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	0h	地址值 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.11 CMDBYTEN (偏移 = 1124h) [复位 = 00000000h]

图 6-14 展示了 CMDBYTEN，表 6-24 中对此进行了介绍。

返回到汇总表。

命令编程字节使能寄存器：
此寄存器构成编程数据的每字节使能。对于要编程的数据字节，必须将 1 写入到此寄存器中的相应位。通常，所有位都写入 1，从而实现完整闪存字的编程。但是，将某些位保留为 0 可实现对闪存字的 8 位、16 位、32 位或 64 位部分的编程。
此外，如果相应 CMDBYTEN 位为 0，读取验证命令会在比较中忽略从闪存读取的数据字节。
对于 64 位闪存字大小的器件，CMDBYTEN 寄存器使用 BIT7-0 启用每个数据字节，使用 BIT8 启用 ECC 代码字节。
对于 128 位闪存字大小的器件，CMDBYTEN 寄存器使用 BIT15-0 启用每个数据字节，使用 BIT17-16 启用 ECC 代码字节。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
在完成所有命令后，此寄存器全部写入 0。

图 6-14. CMDBYTEN

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

表 6-24. CMDBYTEN 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R/W	0h	保留
17-8	RESERVED	R/W	0h	
7-0	值	R/W	0h	命令字节使能值。在此寄存器中放入每个闪存字字节值的 1 位。 0h = [VAL] 的最小值 0003FFFFh = [VAL] 的最大值

6.6.12 CMDDATAINDEX (偏移 = 112Ch) [复位 = 00000000h]

图 6-15 展示了 CMDDATAINDEX，表 6-25 中对此进行了介绍。

返回到汇总表。

命令编程数据索引寄存器：
当多个数据寄存器可用于多字编程时，可以使用指向其中一个数据寄存器的索引写入此寄存器。完成对 CMDDATA* 的写入后，数据将写入按此寄存器中的值索引的物理数据寄存器。
最多可以存在 8 个数据寄存器，因此该寄存器可以用 0x0 写入 0x7。如果存在的数据寄存器少于 8 个，则在为 CMDDATA* 寄存器编制索引时，将忽略此寄存器的连续 MSB 位。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。

图 6-15. CMDDATAINDEX

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

表 6-25. CMDDATAINDEX 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	保留
2-0	值	R/W	0h	数据寄存器索引 0h = [VAL] 的最小值 7h = [VAL] 的最大值

6.6.13 CMDDATA0 (偏移 = 1130h) [复位 = FFFFFFFFh]

图 6-16 展示了 CMDDATA0，表 6-26 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 0
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 0 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-16. CMDDATA0

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

表 6-26. CMDDATA0 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.14 CMDDATA1 (偏移 = 1134h) [复位 = FFFFFFFFh]

图 6-17 展示了 CMDDATA1，表 6-27 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 1
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 0 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-17. CMDDATA1

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

表 6-27. CMDDATA1 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.15 CMDDATA2 (偏移 = 1138h) [复位 = FFFFFFFFh]

图 6-18 展示了 CMDDATA2，表 6-28 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 2
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 1 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-18. CMDDATA2

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

表 6-28. CMDDATA2 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.16 CMDDATA3 (偏移 = 113Ch) [复位 = FFFFFFFFh]

图 6-19 展示了 CMDDATA3，表 6-29 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 3
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 1 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-19. CMDDATA3

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

表 6-29. CMDDATA3 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.17 CMDDATA4 (偏移 = 1140h) [复位 = FFFFFFFFh]

图 6-20 展示了 CMDDATA4，表 6-30 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 4
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 2 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-20. CMDDATA4

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

表 6-30. CMDDATA4 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。T 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.18 CMDDATA5 (偏移 = 1144h) [复位 = FFFFFFFFh]

图 6-21 展示了 CMDDATA5，表 6-31 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 5
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 2 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-21. CMDDATA5

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

表 6-31. CMDDATA5 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.19 CMDDATA6 (偏移 = 1148h) [复位 = FFFFFFFFh]

图 6-22 展示了 CMDDATA6，表 6-32 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 6
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 3 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-22. CMDDATA6

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

表 6-32. CMDDATA6 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.20 CMDDATA7 (偏移 = 114Ch) [复位 = FFFFFFFFh]

图 6-23 展示了 CMDDATA7，表 6-33 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 7
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 3 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-23. CMDDATA7

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

表 6-33. CMDDATA7 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.21 CMDDATA8 (偏移 = 1150h) [复位 = FFFFFFFFh]

图 6-24 展示了 CMDDATA8，表 6-34 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 8
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 4 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-24. CMDDATA8

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

表 6-34. CMDDATA8 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.22 CMDDATA9 (偏移 = 1154h) [复位 = FFFFFFFFh]

图 6-25 展示了 CMDDATA9，表 6-35 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 9
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 4 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-25. CMDDATA9

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

表 6-35. CMDDATA9 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.23 CMDDATA10 (偏移 = 1158h) [复位 = FFFFFFFFh]

图 6-26 展示了 CMDDATA10，表 6-36 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 10
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 5 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-26. CMDDATA10

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

表 6-36. CMDDATA10 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.24 CMDDATA11 (偏移 = 115Ch) [复位 = FFFFFFFFh]

图 6-27 展示了 CMDDATA11，表 6-37 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 11
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 5 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-27. CMDDATA11

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

表 6-37. CMDDATA11 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.25 CMDDATA12 (偏移 = 1160h) [复位 = FFFFFFFFh]

图 6-28 展示了 CMDDATA12，表 6-38 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 12
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 6 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-28. CMDDATA12

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

表 6-38. CMDDATA12 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.26 CMDDATA13 (偏移 = 1164h) [复位 = FFFFFFFFh]

图 6-29 展示了 CMDDATA13，表 6-39 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 13
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 6 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-29. CMDDATA13

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

表 6-39. CMDDATA13 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.27 CMDDATA14 (偏移 = 1168h) [复位 = FFFFFFFFh]

图 6-30 展示了 CMDDATA14，表 6-40 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 14
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 7 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-30. CMDDATA14

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

表 6-40. CMDDATA14 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.28 CMDDATA15 (偏移 = 116Ch) [复位 = FFFFFFFFh]

图 6-31 展示了 CMDDATA15，表 6-41 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 15
此寄存器包含命令的数据。
此寄存器代表闪存字数据寄存器 7 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有 NoWrapper 命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-31. CMDDATA15

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

表 6-41. CMDDATA15 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.29 CMDDATA16 (偏移 = 1170h) [复位 = FFFFFFFFh]

图 6-32 展示了 CMDDATA16，表 6-42 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 16
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 4 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-32. CMDDATA16

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

表 6-42. CMDDATA16 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.30 CMDDATA17 (偏移= 1174h) [复位= FFFFFFFFh]

图 6-33 展示了 CMDDATA17，表 6-43 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 17
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 4 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-33. CMDDATA17

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

表 6-43. CMDDATA17 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.31 CMDDATA18 (偏移= 1178h) [复位= FFFFFFFFh]

图 6-34 展示了 CMDDATA18，表 6-44 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 18
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 4 的位 95:64。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-34. CMDDATA18

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

表 6-44. CMDDATA18 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.32 CMDDATA19 (偏移= 117Ch) [复位= FFFFFFFFh]

图 6-35 展示了 CMDDATA19，表 6-45 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 19
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 4 的位 127:96。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-35. CMDDATA19

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

表 6-45. CMDDATA19 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.33 CMDDATA20 (偏移= 1180h) [复位= FFFFFFFFh]

图 6-36 展示了 CMDDATA20，表 6-46 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 20
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 5 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-36. CMDDATA20

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

表 6-46. CMDDATA20 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.34 CMDDATA21 (偏移 = 1184h) [复位= FFFFFFFFh]

图 6-37 展示了 CMDDATA21，表 6-47 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 21
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 5 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-37. CMDDATA21

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

表 6-47. CMDDATA21 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.35 CMDDATA22 (偏移 = 1188h) [复位= FFFFFFFFh]

图 6-38 展示了 CMDDATA22，表 6-48 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 22
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 5 的位 95:64。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-38. CMDDATA22

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

表 6-48. CMDDATA22 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.36 CMDDATA23 (偏移 = 118Ch) [复位 = FFFFFFFFh]

图 6-39 展示了 CMDDATA23，表 6-49 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 23
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 5 的位 127:96。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-39. CMDDATA23

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

表 6-49. CMDDATA23 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.37 CMDDATA24 (偏移 = 1190h) [复位 = FFFFFFFFh]

图 6-40 展示了 CMDDATA24，表 6-50 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 24
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 6 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-40. CMDDATA24

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

表 6-50. CMDDATA24 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.38 CMDDATA25 (偏移 = 1194h) [复位 = FFFFFFFFh]

图 6-41 展示了 CMDDATA25，表 6-51 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 25
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 6 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-41. CMDDATA25

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

表 6-51. CMDDATA25 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.39 CMDDATA26 (偏移 = 1198h) [复位 = FFFFFFFFh]

图 6-42 展示了 CMDDATA26，表 6-52 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 26
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 6 的位 95:64。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-42. CMDDATA26

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

表 6-52. CMDDATA26 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.40 CMDDATA27 (偏移 = 119Ch) [复位 = FFFFFFFFh]

图 6-43 展示了 CMDDATA27，表 6-53 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 27
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 6 的位 127:96。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-43. CMDDATA27

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

表 6-53. CMDDATA27 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.41 CMDDATA28 (偏移 = 11A0h) [复位 = FFFFFFFFh]

图 6-44 展示了 CMDDATA28，表 6-54 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 28
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 7 的位 31:0。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-44. CMDDATA28

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

表 6-54. CMDDATA28 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.42 CMDDATA29 (偏移 = 11A4h) [复位 = FFFFFFFFh]

图 6-45 展示了 CMDDATA29，表 6-55 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 29
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 7 的位 63:32。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-45. CMDDATA29

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

表 6-55. CMDDATA29 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.43 CMDDATA30 (偏移 = 11A8h) [复位 = FFFFFFFFh]

图 6-46 展示了 CMDDATA30，表 6-56 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 30
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 7 的位 95:64。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-46. CMDDATA30

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

表 6-56. CMDDATA30 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.44 CMDDATA31 (偏移 = 11ACh) [复位 = FFFFFFFFh]

图 6-47 展示了 CMDDATA31，表 6-57 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 31
此寄存器构成命令的数据。
对于 DATAWIDTH == 128：此寄存器代表闪存字数据寄存器 7 的位 127:96。
将 1 写入到 CMDEXEC 寄存器之后以及在闪存包装程序硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有闪存包装程序命令后全部写入 1。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。

图 6-47. CMDDATA31

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

表 6-57. CMDDATA31 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	在此字段中放入一个 32 位数据值。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.45 CMDDATAECC0 (偏移 = 11B0h) [复位 = 0000FFFFh]

图 6-48 展示了 CMDDATAECC0，表 6-58 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 0
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 0。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-48. CMDDATAECC0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-58. CMDDATAECC0 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.46 CMDDATAECC1 (偏移 = 11B4h) [复位 = 0000FFFFh]

图 6-49 展示了 CMDDATAECC1，表 6-59 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 1
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 1。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-49. CMDDATAECC1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-59. CMDDATAECC1 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.47 CMDDATAECC2 (偏移 = 11B8h) [复位 = 0000FFFFh]

图 6-50 展示了 CMDDATAECC2，表 6-60 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 2
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 2。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-50. CMDDATAECC2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-60. CMDDATAECC2 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.48 CMDDATAECC3 (偏移 = 11BCh) [复位 = 0000FFFFh]

图 6-51 展示了 CMDDATAECC3，表 6-61 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 3
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 3。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-51. CMDDATAECC3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-61. CMDDATAECC3 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.49 CMDDATAECC4 (偏移 = 11C0h) [复位 = 0000FFFFh]

图 6-52 展示了 CMDDATAECC4，表 6-62 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 4
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 4。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-52. CMDDATAECC4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-62. CMDDATAECC4 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.50 CMDDATAECC5 (偏移 = 11C4h) [复位 = 0000FFFFh]

图 6-53 展示了 CMDDATAECC5，表 6-63 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 5
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 5。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-53. CMDDATAECC5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-63. CMDDATAECC5 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.51 CMDDATAECC6 (偏移 = 11C8h) [复位 = 0000FFFFh]

图 6-54 展示了 CMDDATAECC6，表 6-64 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 6
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 6。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-54. CMDDATAECC6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-64. CMDDATAECC6 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.52 CMDDATAECC7 (偏移 = 11CCh) [复位 = 0000FFFFh]

图 6-55 展示了 CMDDATAECC7，表 6-65 中对此进行了介绍。

返回到汇总表。

命令数据寄存器 7
此寄存器构成命令数据的 ECC 部分。此寄存器中的这个 ECC 数据涵盖闪存数据寄存器 7。
可以覆盖硬件 ECC 生成，并可使用在其他地方开发的 ECC 数据。在此寄存器中放入 ECC 数据。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
CMDDATA* 寄存器的用例如下：
1) 编程 - 这些寄存器包含要编程的数据。
2) 擦除 - 不使用这些寄存器。
3) 读取验证 - 这些寄存器包含要验证的数据。
此寄存器用于聚合在编程运行期间不需要额外编程脉冲的位的掩码，并将在完成所有命令后全部写入 1。

图 6-55. CMDDATAECC7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																VAL1								VAL0							
R/W-0h																R/W-FFh								R/W-FFh							

表 6-65. CMDDATAECC7 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	保留
15-8	VAL1	R/W	FFh	数据的位 127:64 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值
7-0	VAL0	R/W	FFh	数据的位 63:0 的 ECC 数据放置在这里。 0h = 最小值 FFh = 最大值

6.6.53 CMDWEPROTA (偏移 = 11D0h) [复位 = FFFFFFFFh]

图 6-56 展示了 CMDWEPROTA，表 6-66 中对此进行了介绍。

返回到汇总表。

命令写擦除保护 A 寄存器
此寄存器可以保护主区域的前 32 个扇区不受编程或擦除的影响，使用 1 位保护每个扇区。如果主区域小于 32 个扇区，则此寄存器为整个区域提供保护。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此外，此寄存器用于聚合在存储体擦除运行期间不需要额外擦除脉冲的扇区的掩码，并将在完成所有命令后全部写入 1。

图 6-56. CMDWEPROTA

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

表 6-66. CMDWEPROTA 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	每个位保护 1 个扇区。位 [0]：如果为 1，则保护闪存的扇区 0 不受编程和擦除的影响。位 [1]：如果为 1，则保护闪存的扇区 1 不受编程和擦除的影响。: : 位 [31]：如果为 1，则保护闪存的扇区 31 不受编程和擦除的影响。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.54 CMDWEPROTB (偏移 = 11D4h) [复位 = FFFFFFFFh]

图 6-57 展示了 CMDWEPROTB，表 6-67 中对此进行了介绍。

返回到汇总表。

命令写擦除保护 B 寄存器
此寄存器可以保护主区域扇区不受编程和擦除的影响。每个位对应一组 8 个扇区。如何应用这些保护位有 3 种情况：
1.单存储体系统：
如果只有一个闪存存储体，则通过 CMDWEPROTA 寄存器保护前 32 个扇区。因此，CMDWEPROTB 中的位提供的保护从扇区 32 开始。
2.多存储体系统，存储体 0：
如果有多个闪存存储体，则通过 CMDWEPROTA 寄存器保护存储体 0 的前 32 个扇区。因此，只有 CMDWEPROTB 的位 4 和更高的位才适用于存储体 0。位 4 和更高位的保护将从扇区 32 开始。对于存储体 0，忽略 WEPROTB 的位 3:0。
3.多存储体系统，存储体 1-N：
对于多存储体系统中除存储体 0 之外的存储体，CMDWEPROTA 无效，因此 CMDWEPROTB 中的位将从扇区 0 开始保护这些存储体。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此外，此寄存器用于聚合在存储体擦除运行期间不需要额外擦除脉冲的扇区的掩码，并将在完成所有命令后全部写入 1。

图 6-57. CMDWEPROTB

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

表 6-67. CMDWEPROTB 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	每个位保护一组 8 个扇区。当一个位为 1 时，可以保护闪存中关联的 8 个扇区不受编程和擦除影响。使用此寄存器最多可保护 256 个扇区。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.55 CMDWEPROTC (偏移 = 11D8h) [复位 = FFFFFFFFh]

图 6-58 展示了 CMDWEPROTC，表 6-68 中对此进行了介绍。

返回到汇总表。

命令写擦除保护 C 寄存器
此寄存器可以保护主区域扇区不受编程和擦除的影响。每个位对应一组 8 个扇区。
该寄存器扩展了 CMDWEPROTB 寄存器的保护位，以涵盖大于 32*8=256 扇区的存储体。将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此外，此寄存器用于聚合在存储体擦除运行期间不需要额外擦除脉冲的扇区的掩码，并将在完成所有命令后全部写入 1。

图 6-58. CMDWEPROTC

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

表 6-68. CMDWEPROTC 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	每个位保护一组 8 个扇区。当一个位为 1 时，可以保护闪存中关联的 8 个扇区不受编程和擦除影响。请注意，受此寄存器保护的扇区从闪存中的扇区 256 开始，而受 CMDWEPROTB 寄存器保护的扇区在该扇区结束。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.56 CMDWEPROTNM (偏移 = 1210h) [复位 = FFFFFFFFh]

图 6-59 展示了 CMDWEPROTNM，表 6-69 中对此进行了介绍。

返回到汇总表。

命令写擦除保护非主寄存器
此寄存器可以保护非主区域扇区不受编程和擦除的影响。每个位对应 1 个扇区。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。
此外，此寄存器用于聚合在存储体擦除运行期间不需要额外擦除脉冲的扇区的掩码，并将在完成所有命令后全部写入 1。

图 6-59. CMDWEPROTNM

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

表 6-69. CMDWEPROTNM 字段说明

位	字段	类型	复位	说明
31-0	值	R/W	FFFFFFFFh	每个位保护 1 个扇区。位 [0]：如果为 1，则保护非主区域的扇区 0 不受编程和擦除的影响。位 [1]：如果为 1，则保护非主区域的扇区 1 不受编程和擦除的影响。: : 位 [31]：如果为 1，则保护非主的扇区 31 不受编程和擦除的影响。 0h = [VAL] 的最小值 FFFFFFFFh = [VAL] 的最大值

6.6.57 CFGPCNT (偏移 = 13B4h) [复位 = 00000000h]

图 6-60 展示了 CFGPCNT，表 6-70 中对此进行了介绍。

返回到汇总表。

脉冲计数器配置寄存器
此寄存器可以进一步配置用于编程和擦除操作的最大脉冲计数。
将 1 写入到 CMDEXEC 寄存器之后以及在硬件设置 STATCMD.DONE 之前阻止此寄存器写入。

图 6-60. CFGPCNT

31	30	29	28	27	26	25	24
MAXERSPCNTVAL							
R/W-0h							
23	22	21	20	19	18	17	16
MAXERSPCNTVAL				保留			MAXERSPCNTOVR
R/W-0h				R/W-0h			R/W-0h
15	14	13	12	11	10	9	8
保留				MAXPCNTVAL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
MAXPCNTVAL				保留			MAXPCNTOVR
R/W-0h				R/W-0h			R/W-0h

表 6-70. CFGPCNT 字段说明

位	字段	类型	复位	描述
31-20	MAXERSPCNTVAL	R/W	0h	使用此值覆盖擦除的最大脉冲计数。如果 MAXERSPCNTOVR = 0，则忽略此字段。如果 MAXERSPCNTOVR = 1，则此值将用于覆盖擦除的最大脉冲计数。 0h = 最小值 FFFh = 最大值
19-17	保留	R/W	0h	保留
16	MAXERSPCNTOVR	R/W	0h	覆盖擦除的硬接线最大脉冲计数。如果设置，则 MAXERSPCNTVAL 中的值将用作擦除操作的最大脉冲计数。默认情况下，此位为 0，并使用硬接线最大脉冲计数。 0h = 使用硬接线 (默认) 值来实现最大脉冲计数 1h = 使用 MAXERSPCNTVAL 字段的值作为最大擦除脉冲计数
15-12	保留	R/W	0h	保留
11-4	MAXPCNTVAL	R/W	0h	使用此值覆盖最大脉冲计数器。如果 MAXPCNTOVR = 0，则忽略此字段。如果 MAXPCNTOVR = 1 且 MAXERSPCNTOVR = 0，则此值将用于覆盖编程和擦除的最大脉冲计数。完整最大值为 {4'h0, MAXPCNTVAL}。如果 MAXPCNTOVR = 1 且 MAXERSPCNTOVR = 1，则此值将仅用于覆盖编程的最大脉冲计数。完整最大值为 {4'h0, MAXPCNTVAL}。 0h = 最小值 FFh = 最大值
3-1	保留	R/W	0h	保留
0	MAXPCNTOVR	R/W	0h	覆盖硬接线最大脉冲计数。如果未设置 MAXERSPCNTOVR，则单独设置此值将覆盖编程和擦除的最大脉冲计数。如果设置了 MAXERSPCNTOVR，则此位将仅控制编程的最大脉冲计数设置。默认情况下，此位为 0，并使用硬接线最大脉冲计数。 0h = 使用硬接线 (默认) 值来实现最大脉冲计数 1h = 使用 MAXPCNTVAL 字段的值作为最大脉冲计数

6.6.58 STATCMD (偏移 = 13D0h) [复位 = 00000000h]

图 6-61 展示了 STATCMD，表 6-71 中对此进行了介绍。

返回到汇总表。

命令状态寄存器 此寄存器包含有关命令执行完成和错误的状态。

图 6-61. STATCMD

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留			FAILMISC	RESERVED			RESERVED
R-0h			R-0h	R-0h			R-
7	6	5	4	3	2	1	0
FAILMODE	FAILILLADDR	FAILVERIFY	FAILWEPROT	保留	CMDINPROGRESS	CMDPASS	CMDDONE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 6-71. STATCMD 字段说明

位	字段	类型	复位	说明
31 - 13	RESERVED	R	0h	保留
12	FAILMISC	R	0h	由于出现除写/擦除保护违例或验证错误以外的其他错误，命令失败。如果添加了需要状态位的新故障机制，则这是一个额外位。 0h = 无故障 1h = 故障
11-9	保留	R	0h	被保留
8	保留	R	0h	
7	FAILMODE	R	0h	因为存储体已设置为 READ 以外的模式，命令失败。除非所有存储体都处于 READ 模式，否则无法启动编程和擦除命令。 0h = 无故障 1h = 故障
6	FAILILLADDR	R	0h	由于使用了非法地址，命令失败 0h = 无故障 1h = 故障
5	FAILVERIFY	R	0h	由于验证错误，命令失败 0h = 无故障 1h = 故障
4	FAILWEPROT	R	0h	由于写/擦除保护扇区违例，命令失败 0h = 无故障 1h = 故障
3	RESERVED	R	0h	保留
2	CMDINPROGRESS	R	0h	命令正在进行 0h = 完成 1h = 正在进行
1	CMDPASS	R	0h	命令通过 - 当 CMD_DONE 字段为 1 时有效 0h = 未通过 1h = 通过

表 6-71. STATCMD 字段说明 (continued)

位	字段	类型	复位	说明
0	CMDDONE	R	0h	命令完成 0h = 未完成 1h = 完成

6.6.59 STATADDR (偏移 = 13D4h) [复位 = 00010000h]

图 6-62 展示了 STATADDR，表 6-72 中对此进行了介绍。

返回到汇总表。

当前地址计数器值 对状态机当前地址进行读取访问的只读寄存器。存储体 ID、区域 ID 和地址存储在此寄存器中，并在命令执行期间根据需要递增。

图 6-62. STATADDR

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

表 6-72. STATADDR 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	保留
25-21	BANKID	R	0h	当前存储体 ID 在此寄存器中存储了一个存储体指示符，表示状态机正在运行的当前存储体。每个存储体有 1 位。 1h (R/W) = 存储体 0 2h (R/W) = 存储体 1 4h (R/W) = 存储体 2 8h (R/W) = 存储体 3 10h (R/W) = 存储体 4
20-16	REGIONID	R	1h	当前区域 ID 在此寄存器中存储了一个区域指示符，表示状态机正在运行的当前闪存区域。 1h (R/W) = 主区域 2h (R/W) = 非主区域 4h (R/W) = 修正区域 8h (R/W) = 工程区域
15-0	BANKADDR	R	0h	当前存储体地址 此寄存器中存储了存储体偏移地址。 0h = 最小值 FFFFh = 最大值

6.6.60 STATPCNT (偏移 = 13D8h) [复位 = 00000000h]

图 6-63 展示了 STATPCNT，表 6-73 中对此进行了介绍。

返回到汇总表。

当前脉冲计数寄存器：只读寄存器为用于编程/擦除操作的状态机当前脉冲计数值授予读取访问权限。

图 6-63. STATPCNT

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

表 6-73. STATPCNT 字段说明

位	字段	类型	复位	说明
31-12	RESERVED	R	0h	保留
11-0	PULSECNT	R	0h	当前脉冲计数器值 0h = 最小值 FFFh = 最大值

This page intentionally left blank.

章节 7
事件

事件管理器提供外设到外设、外设到 DMA 和外设到 CPU (IRQ) 事件连接。

7.1 事件概述

事件管理器将数字事件从一个实体（例如外设）传输到另一个实体（例如，另一个外设、DMA 或 CPU）。事件管理器通过一组定义的事件发布者（发生器）和订阅者（接收器）实现事件传输，这些事件发布者和订阅者通过包含固定（静态）路由和可编程路由组合的事件结构进行互连。

事件管理器传输的事件包括：

- 作为中断请求 (IRQ) 传输到 CPU 的外设事件
  - 示例：RTC 中断会发送到 CPU
- 作为 DMA 触发器传输到 DMA 的外设事件
  - 示例：传输到 DMA、请求 DMA 传输的 UART 数据接收触发器
- 传输到另一个外设以直接触发硬件中操作的外设事件
  - 示例：TIMx 计时器外设将周期性事件发布到 ADC 订阅者端口，ADC 使用该事件触发采样开始

除了提供事件传输逻辑外，如果事件需要更改器件的电源和/或时钟配置以正确处理事件，则事件管理器还会与电源管理和时钟单元 (PMCU) 连接。例如，如果外设生成针对 DMA 的事件，并且器件处于 STOP 或 STANDBY 工作模式（禁用 DMA），则事件管理器将与 PMCU 握手以暂停低功耗工作模式状态，并会启用 DMA，以便可以处理 DMA 传输。

事件管理器配置取决于器件，因为不同的器件支持不同的外设。请参阅器件特定的数据表以了解特定于器件的事件实现。

7.1.1 事件发布者

事件发布者是事件结构上所传播事件的来源。外设包含用于通过发布端口 FPUB_x 将 CPU 中断、DMA 触发和通用事件发布到事件结构的事件发布者。发布者行为通过标准化事件管理寄存器进行配置。

7.1.2 事件订阅者

事件订阅者包括在处理器、DMA 和某些外设中（请参阅节 7.1.4）。事件订阅者通过订阅端口 FSUB_x 来订阅事件。事件订阅者使模块能够订阅由事件发布者发布到事件结构的事件，并对这些事件执行预定义操作。

7.1.3 事件结构路由

通过事件结构的路由有三种不同类型，用于将发布者连接到订阅者：CPU 中断事件、DMA 触发事件和通用事件。

7.1.3.1 CPU 中断事件路由 (CPU_INT)

CPU 中断事件路由是一个事件发布者（在外设模块内部）和一个用于传播 CPU 中断的事件订阅者（CPU 子系统）之间的固定点对点连接。

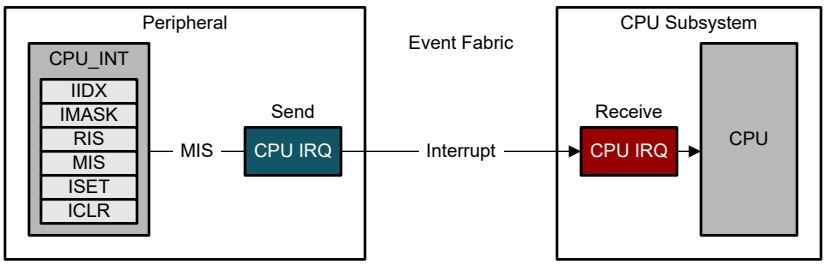


图 7-1. CPU 中断（固定事件路由）

对于每个能够产生 CPU 中断的外设，都会获得一条从外设的屏蔽中断状态 (MIS) 寄存器到 CPU 子系统的中断管理逻辑的固定路由。

如果软件未清除外设的事件管理寄存器内的中断请求，则该请求将一直挂起到 CPU 子系统。请参阅节 7.2.5.3，了解关于使用事件管理寄存器来设置和清除中断状态的指导。

7.1.3.2 DMA 触发事件路由 (DMA_TRIGx)

DMA 路由是外设与 DMA 控制器之间的固定路由，（可选）通过附加边带信号将 DMA 完成条件从 DMA 控制器传递回触发外设，以指示 DMA 活动何时运行到结束。DMA 触发路由如图 7-2 所示。

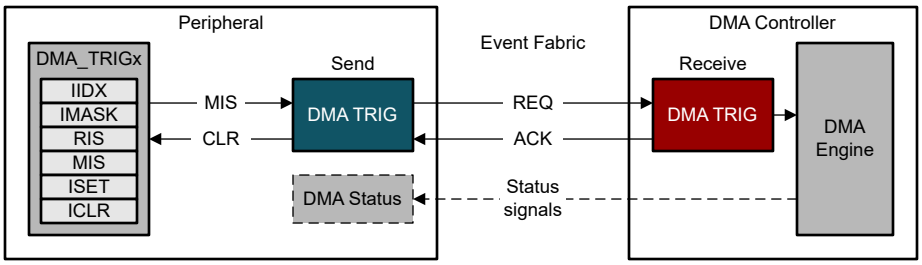


图 7-2. DMA 路由

大多数能够生成 DMA 触发的外设还有一组额外的事件管理寄存器（除了用于 CPU 中断和任何 GEN_EVENTx 通用路由发布者的 CPU_INT 寄存器外）。可使用这些寄存器来选择用于生成 DMA 触发的外设条件。

当 DMA 接收到触发信号时，DMA 确认请求，外设清除该请求。DMA 还确认已清除的请求，之后外设可以发出新的请求。

DMA 路由还可以包含状态信号（针对特定外设），以向触发外设指示 DMA 传输序列已完成。例如，可以设置 DMA，以便根据 UART TX DMA 触发，从 SRAM 缓冲区传输 *N* 个字节到 UART TX 数据寄存器中。每次从 UART 触发时，DMA 都会确认传输成功。在第 *N* 个字节上，DMA 将向 UART 发送一个完整状态信号，UART 可使用该信号将传输完成中断传播到 CPU。

特殊情况

某些外设（例如 12 位 DAC）不会实现用于管理其 DMA 触发的事件管理寄存器组。在这些情况下，外设会实现特定的 DMA 配置逻辑，这样管理寄存器就不需要与 DMA 连接。图 7-3 展示了未实现事件管理寄存器时的模型。有关在这种情况下如何配置 DMA 通道的指导，请参阅本文档的外设特定部分。

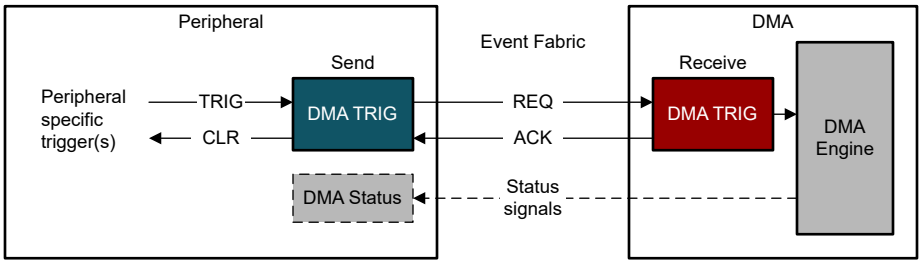


图 7-3. 无事件管理寄存器的 DMA 路由

7.1.3.3 通用事件路由 (GEN_EVENTx)

通用路由是点对点 (1:1) 路由或一分二 (1:2) 分离器路由，其中发布事件的外设使用多个可用的通用路由通道之一来将事件发布到另一个实体（如果是分离器路由，则为多个实体）。在这里，实体可以是另一个外设、通用 DMA 触发事件或通用 CPU 事件，如图 7-4 所示。

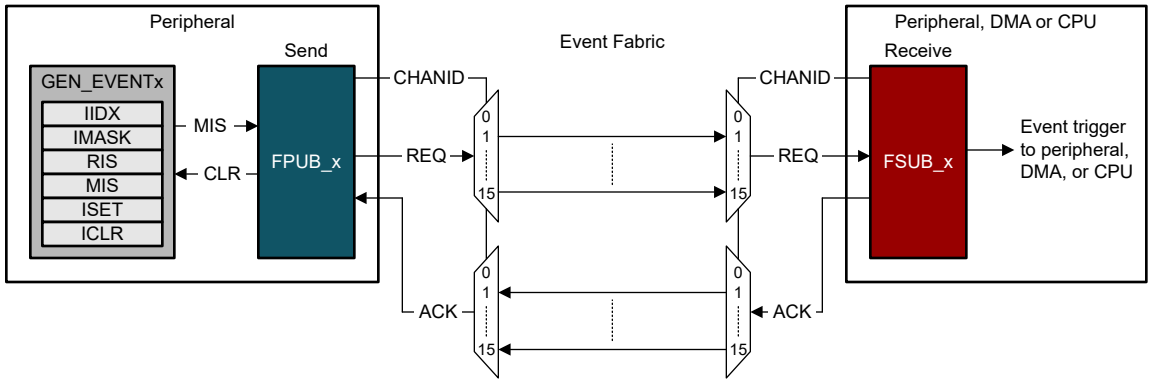


图 7-4. 通用路由

能够生成通用事件的外设具有一组（或多组）额外的 GEN_EVENTx 事件管理寄存器（除了用于 CPU 中断或 DMA DMA_TRIGx 的 CPU_INT 寄存器，如果存在的话）。这些寄存器可用于选择要用于发布通用事件的外设条件。配置后，该事件将广播到 FPUB_x 寄存器选择的通用路由通道。另一个外设、DMA 或 CPU 可以通过将其通用订阅者端口 (FSUB_x) 配置为侦听发布外设所连接的同一通用路由通道来订阅此事件。

通用路由通道可以配置为一个订阅者（1:1 路由）或两个订阅者（1:2 分离器路由），具体取决于选择的通道。有关可用通用路由通道及其类型（1:1 或 1:2）的完整列表，请参阅器件数据表。通用路由通道一次只能使用一个发布外设进行配置。外设订阅 1:1 通用路由通道后，除非先断开最初连接的外设，否则任何其他外设都无法选择要订阅的通道。具有分离器功能 (1:2) 的通用路由通道恰好支持两个外设订阅该通道，之后硬件将阻止添加订阅者的额外尝试，直到两个连接的外设都与分离器通道断开连接。

每个外设类型都在以下方面具有独特的功能：可以生成要发布的事件，以及可以在外设内触发的订阅事件。查看本指南中与相关外设相对应的一章，了解给定外设上发布者和订阅者端口的功能。

7.1.4 事件路由映射

每种外设类型的事件功能如图 7-5 所示。UART、SPI 和 I2C 等外设生成静态路由到 CPU 的 CPU 中断事件，它们还生成路由到 DMA 的 DMA 触发事件。GPIO 和 ADC 等外设也会生成 CPU 中断事件，但它们还支持生成和接收通过一个通用通道进行路由的事件。例如，通过使用通用事件通道，可以通过将 GPIO FPUB_x 和 ADC FSUB_0 连接到同一通用事件通道，直接从 GPIO 事件启动 ADC 转换。

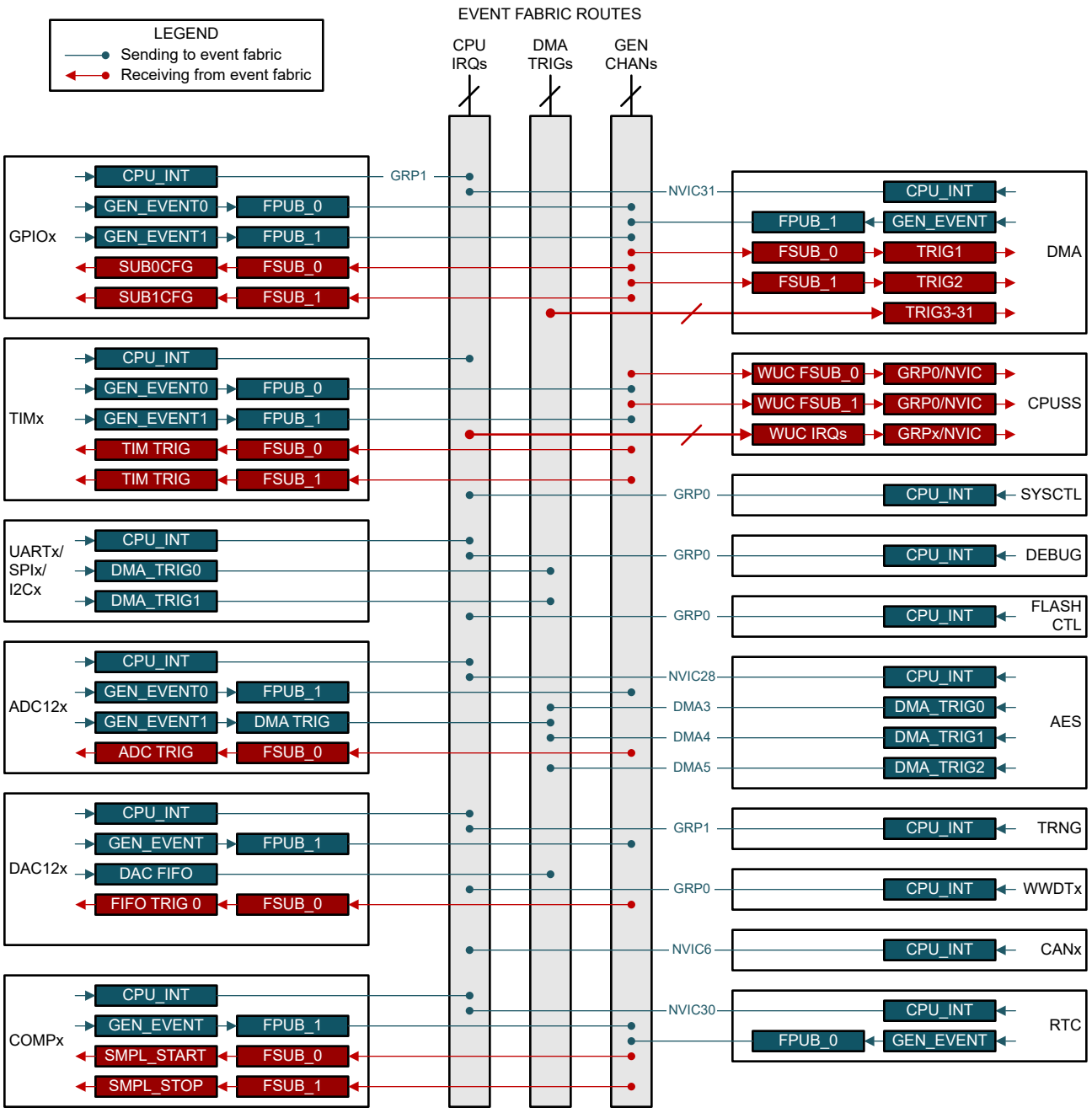


图 7-5. 事件映射

7.1.5 事件传播延迟

通用路由通道在发布实体和订阅实体之间实现了四路硬件握手。此握手需要四个 ULPCLK 周期才能完成：

1. 发布者向订阅者发出请求
2. 订阅者向发布者确认
3. 发布者向订阅者发出的请求无效
4. 订阅者向发布者确认无效

如果发布外设发送两个请求，并且第一个请求尚未清除握手，则第二个请求将被丢弃。

7.2 事件操作

本节介绍如何配置外设以使用事件管理器。请注意，事件管理器本身不包含任何配置寄存器。所有事件配置均通过发布和订阅外设完成。

7.2.1 CPU 中断

外设中断请求 (IRQ) 通过事件管理器传播到 CPU 子系统。外设中断请求使用固定路由，但除此之外，CPU 子系统还可提供两个通用事件订阅者端口，可用于通过通用路由触发 CPU 中断。有关给定器件的中断分配的完整列表，请参阅器件特定的数据表。

标准 CPU 中断请求

固定路由中断不需要特殊配置。中断可通过外设的 CPU_INT 事件管理寄存器（IIDX、IMASK、RIS、MIS、ISET 和 ICLR）以及通过 CPU 子系统中断配置来管理（请参阅节 3.3）。

基于通用事件的 CPU 中断请求

CPU 子系统包含两个通用事件用户端口 (FSUB_x)，可用于从器件的任何通用事件通道中获取 CPU 中断。这可用于支持特殊情况，在此类情况下，外设上的特定功能向 CPU 子系统生成专用中断，该中断独立于该外设的标准中断机制并作为其补充。

考虑 GPIO 外设，它具有标准中断请求以及 2 个发布者，可以根据 GPIO 中定义的状态向任何通用事件通道路由 GPIO 事件。例如，可以需要将大多数 GPIO 事件配置为提供标准中断，而单个特定 GPIO 事件通过通用路由提供第二个专用 CPU 中断。这使得应用软件能够使 GPIO 具有两个完全独立的中断处理程序。

要将事件管理器配置为从通用路由触发 CPU 中断，请执行以下步骤：

1. 配置生成事件的外设的 GEN_EVENT 寄存器，以选择所需的外设状态作为事件发生器。
2. 通过要使用的通用路由通道 ID 配置生成事件的外设的 FPUB_x 寄存器。此通道不得被另一个外设使用。
3. 配置唤醒控制器 (WUC) 的 FSUB_x 寄存器，该寄存器捕获要转发到 CPU 子系统的通用路由通道事件。
4. 配置 CPU 子系统中断管理以启用 GENSUBx 中断。

请注意，当通过通用路由生成 CPU 中断时，通用事件逻辑将作为四次事件握手的一部分自动清除挂起的中断请求。应用软件将无法从外设寄存器中读取中断原因，并且不需要将任何中断状态位清零。软件只能读取 FSUB_x 通用事件生成的中断。这减少了中断开销。

7.2.2 DMA 触发

DMA 触发通过事件管理器传播到 DMA。大多数 DMA 触发使用固定路由，但 DMA 提供了两个通用事件订阅者端口，可用于触发通过通用路由通道的 DMA 传输。有关给定器件的 DMA 触发分配的完整列表，请参阅器件特定数据表。

标准 DMA 触发

要确定器件上的特定外设是否提供从外设直接到 DMA 的固定 DMA 触发 (DMA_TRIGx)，请查看器件特定数据表中详细说明部分的 DMA 触发表。某些外设可以有多个 DMA 触发（例如，在串行通信外设上启用 TX 触发和 RX 触发）。

要选择触发静态 DMA 路由的特定外设事件，请配置与目标 DMA 路由对应的外设的 DMA_TRIGx 事件管理寄存器组（IIDX、IMASK、RIS、MIS、ISET 和 ICLR）。要确定哪个 DMA_TRIGx 寄存器组与哪个 DMA 触发相对应，请查看本指南中相应外设的相关章节，或查看 节 7.1.4。

某些外设（例如 12 位 DAC）不会实现用于管理 DMA 触发的 DMA_TRIGx 寄存器组。在这些情况下，通过特定于外设的配置寄存器完成 DMA 触发配置。

基于通用事件的 DMA 触发

DMA 包含两个通用事件订阅者端口 (FSUB_x)，可用于从器件的任何通用事件通道中获取 DMA 触发。这可用于启用外设上的特定功能生成 DMA 触发信号的特殊情况。例如，可能需要从计时器触发 DMA 传输。

要将事件管理器配置为从通用路由触发 DMA 通道，请执行以下步骤：

1. 配置生成事件的外设的 GEN_EVENTx 寄存器，以选择所需的外设状态作为事件发生器。
2. 通过要使用的通用事件通道 ID 配置生成事件的外设的 FPUB_x 寄存器。此通道不得被另一个外设使用。
3. 配置 DMA 的 FSUB_x 寄存器，该寄存器捕获通用事件通道事件用作 DMA 中的触发。
4. 根据 DMA 一章中的配置指令来配置 DMA。

7.2.3 外设间事件

外设间事件使一个外设中的条件能够在第二个（或第三个）外设中触发操作，完全在硬件中触发，无需任何 CPU 交互。该器件提供一定数量的通用路由通道，这些通道可以通过包含发布者和订阅者端口的外设进行发布或订阅。在建立配置之前，请执行以下步骤：

1. 查看器件特定的数据表，确定目标器件上可用的通用路由通道数和通道类型。根据所需的功能选择相应的通道类型（点对点或分离器），并确定用于连接的通道编号（该通道不得事先被其他外设占用）。
2. 查看要连接的外设的发布者和订阅者功能。有些外设具有多个发布者端口和/或多个订阅者端口，而有些外设没有发布者端口或订阅者端口。要了解外设的可用端口，请查看本指南中的外设参考章节，或查看节 7.1.4 中的通用事件通道连接。

一旦确定要使用的通道并且已知所连接外设的发布者端口和订阅者端口后，请使用以下步骤建立事件连接。此示例中将配置一个由计时器触发的 ADC 应用，使用 TIMG0 将事件发布到通用通道 1，并由 ADC0 订阅通用通道 1 作为转换开始触发。

1. 配置 TIMG0 的 GEN_EVENTx 事件管理寄存器，以根据相应的计时器事件（例如归零事件）设置事件请求。
2. 将 0x1 存储到 TIMG0 的 FPUB_0 寄存器中，以便将 GEN_EVENTx 寄存器选择的 TIMG0 事件发布到通用路由通道 1。通道 1 不能正在被另一个外设使用。
3. 将 0x1 存储到 ADC0 的 FSUB_0 寄存器中，以便 ADC0 侦听计时器发布到通道 1 的事件。
4. 根据节 10.2.8 中的配置说明，将 ADC0 配置为从订阅者端口进行触发。
5. 配置并启用 TIMG0。

7.2.4 扩展的模块说明寄存器

DESC_EX 寄存器是事件管理器中的只读寄存器，可由应用软件读取，以确定给定器件上可用的点对点（单）通用路由通道和分离器（双）通用路由通道数量。

7.2.5 使用事件寄存器

事件管理寄存器组是一组标准寄存器，由所有能够生成事件（CPU 中断、DMA 触发或通用事件）的外设实现。外设中的每个事件生成器都包含自己的事件管理寄存器组。例如，如果外设支持生成 CPU 中断和 DMA 触发，它将具有一个用于 CPU 中断的事件管理寄存器组（组名称为 CPU_INT），以及另一个用于 DMA 触发的事件管理寄存器组（组名称为 DMA_TRIG）。

事件管理寄存器用于：

- 配置用于生成事件（屏蔽）的外设条件
- 报告原始和屏蔽的外设事件状态
- 通过软件设置或清除外设事件状态

在给定外设的“寄存器”部分中，“组”列显示组名称，以指示映射到每个事件管理寄存器组的功能。请参阅表 7-1，了解哪些事件管理组映射到外设“寄存器”部分中组名称的特定功能。

表 7-1. 事件管理组功能和映射

组名称（在寄存器中）	功能
CPU_INT	CPU 中断（到 CPU 子系统的固定路由）
DMA_TRIGx	DMA 触发（到 DMA 控制器的固定路由）

表 7-1. 事件管理组功能和映射 (continued)

组名称 (在寄存器中)	功能
GEN_EVENT	通用事件 (用于其他模块到模块连接的可编程路由)

7.2.5.1 事件寄存器

事件管理寄存器组包含 6 个标准寄存器：RIS、IMASK、MIS、ISET、ICLR 和 IIDX，如表 7-2 所示。这些事件寄存器相互连接，如图 7-6 所示。

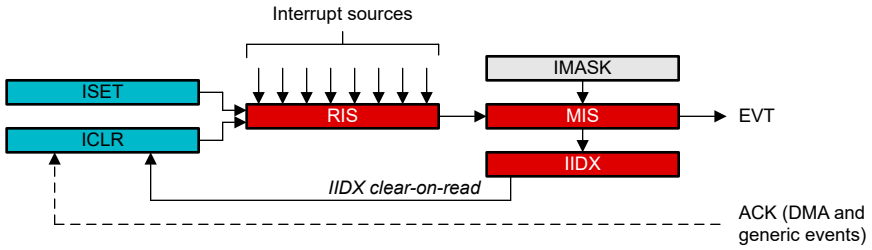


图 7-6. 事件管理寄存器关系

生成事件的外设将包含一个或多个连接到原始中断状态 (RIS) 寄存器的中断源信号。软件可以随时轮询 RIS 以检查原始中断状态。软件也可以通过写入 ICLR 寄存器中的相应位来清除 RIS 寄存器中的挂起中断。RIS 和 IMASK 寄存器通过 MIS 寄存器 (屏蔽中断状态) 中的逐位 AND 功能组合在一起。要取消屏蔽中断，请设置 IMASK 寄存器中的相应位。取消屏蔽后，RIS 和 MIS 寄存器将指示挂起的中断，并生成一个事件。IIDX 寄存器也将使用最高优先级挂起中断的索引进行更新。

如果 CPU 中断 (CPU_INT) 具有 CPU 中断事件路由，则读取 IIDX 寄存器将清除 RIS 和 MIS 寄存器中的最高优先级挂起中断，并将最高优先级挂起中断的索引返回给应用软件。

如果发生硬件事件 DMA 触发路由 (DMA_TRIGx) 或通用事件路由 (GEN_EVENTx)，硬件四次握手将向 ICLR 机制发送 ACK 信号，该机制将清除 RIS 和 MIS 寄存器中的挂起中断。

表 7-2. 标准化事件管理寄存器 (用于 CPU_INT、DMA_TRIGx、GEN_EVENTx 配置)

寄存器	说明	读/写	功能
RIS	原始中断状态	R	指示当前挂起的中断状态，每个中断条件提供一个位。如果中断条件不再存在，则写入 ICLR 将清除 RIS 寄存器中的相应位。
IMASK	中断屏蔽	RW	由应用软件用来配置哪些中断条件传播到一个事件中，每个中断条件提供一个位。
MIS	屏蔽中断状态	R	向软件和硬件指示当前挂起的屏蔽中断状态，每个中断条件提供一个位。MIS 是 RIS 和 IMASK 寄存器的逐位 AND。如果中断条件不再存在，则写入 ICLR 将清除 RIS 寄存器中的相应位。如果 RIS 被清除，MIS 寄存器中的相应位也会被自动清除。
ISET	软件中断设置控制	W	由应用软件用来强制设置用于诊断的中断条件。写入 ISET 将设置 RIS 寄存器中的相应位。如果在 IMASK 中启用了中断条件，则也会设置 MIS 寄存器中的相应位。向 ISET 中某个位的位置写入“1”会设置相应的中断状态。
ICLR	软件中断清除控制	W	由应用软件用来清除 RIS 中的挂起中断状态。向 ICLR 中某个位的位置写入“1”会清除相应的中断状态。如果在 IMASK 中启用了中断，则在 RIS 清除时，MIS 中相应位的位置也会自动清除。如果中断条件仍然存在，清除状态没有任何影响，RIS 将保持设置。
IIDX	挂起中断索引	R	由应用软件用来读取最高优先级的挂起中断，同时清除 RIS 和 MIS 中最高优先级的中断状态。如果没有未屏蔽的中断处于挂起状态 (MIS=0)，那么读取 IIDX 会返回 0，否则它返回一个索引值，指示最高优先级的挂起中断。

7.2.5.2 配置事件

要配置用于触发事件的外设中断源，请设置与所需事件相对应的 IMASK 寄存器中所需中断源相对应的位。设置 IMASK 中的位会使 RIS 寄存器中的原始中断状态传播到 MIS 寄存器。当 MIS 寄存器中的中断状态位被置位时（由于 IMASK 寄存器中的中断未屏蔽，并且 RIS 寄存器中的原始中断待处理），将生成一个事件。

由于应用软件可以通过读取 IIDX 或 MIS 寄存器来确定中断原因，因此可以为 CPU 中断事件启用多个中断源。

对于 DMA 触发器和通用事件发布者等硬件事件，在 IMASK 中只应取消屏蔽一个中断源。

7.2.5.3 响应应用软件中的 CPU 中断

如果事件产生 CPU 中断，应用软件可以通过读取 IIDX 寄存器或读取 MIS 并写入 ICLR 寄存器来确定触发事件生成的外设中断。

使用 CPU_INT IIDX 寄存器的 CPU IRQ 中断服务例程

应用软件可以读取 CPU_INT 组中的 IIDX 寄存器，以确定和清除最高优先级的挂起中断。读取 IIDX 将返回与最高优先级中断相对应的索引，该中断已设置且未被屏蔽。读取操作还将同时清除与读取所返回索引的最高优先级中断相对应的 RIS 和 MIS 位。从 IIDX 寄存器读取的值随后可用于选择语句，如下所示。

```
void ISR_IIDX(void)
{
  switch(IIDX)
  {
    case 0:         // no IRQ pending
      break;
    case 1:         // IRQ[0]
      do_irq0();
      break;
    case 2:         // IRQ[1]
      do_irq1();
      break;
    default:        // out of range
      illegal();
  }
}
```

使用 CPU_INT MIS 和 ICLR 寄存器的 CPU IRQ 中断服务例程

或者，应用软件可以读取 MIS 寄存器来确定设置了哪些位，然后使用 ICLR 寄存器来清除挂起的中断状态位。

```
void ISR(void)
{
  uint32_t pending = MIS;
  ICLR = pending;           // clear pending IRQ
  if (pending & 0x01)       // IRQ[0]
  {
      do_irq0();
  }
  if (pending & 0x02)       // IRQ[1]
  {
      do_irq1();
  }
}
```

7.2.5.4 硬件事件处理

如果是源自 DMA 触发 (DMA_TRIG) 或通用事件 (GEN_EVENT) 的事件，则不使用 IIDX 寄存器。在生成事件的外设和订阅事件的硬件实体（例如 DMA 或辅助外设）之间执行四次事件握手。四次事件握手将自动清除 RIS 和 MIS 寄存器中相应的中断状态位。

This page intentionally left blank.

章节 8
IOMUX

IOMUX 通过数字输入输出 (IO) 功能控制所有器件引脚的配置，包括：数字功能选择、反相控制、驱动强度（如果适用）、上拉或下拉电阻（如果适用），以及唤醒配置（如果适用于某些 IO，用于从关断模式唤醒）。

8.1 IOMUX 概述

IOMUX 可管理数字 IO 的配置。IOMUX 配置的主要功能包括：

- 选择将哪个外设复用到每个数字 IO 引脚（例如，GPIO 或 UART 外设）
- 数字输入路径配置
  - 迟滞控制
  - 输入路径启用/禁用
  - 输入逻辑反转控制
- 数字输出路径配置
  - 驱动强度控制
  - 输出连接启用/禁用
  - 输出逻辑反转（与输入逻辑反转共享控制）
  - 逻辑高电平至高阻态输出转换（用于开漏型接口）
  - 禁用连接到 IO 的外设时保持“最后状态”
- 唤醒配置（用于从 SHUTDOWN 模式唤醒）
  - 从 PINCM 寄存器的唤醒状态位读取唤醒源
  - 唤醒比较电平
  - 释放 SHDNIOREL
  - 唤醒启用/禁用
- 上拉和下拉电阻控制

8.1.1 IO 类型和模拟共享

IOMUX 用于管理要在数字 IO 上使用的外设函数的选择。它还为输出驱动器、输入路径和从 SHUTDOWN 模式唤醒的唤醒逻辑提供控制。

数字 IO 类型

给定器件上可以包含多种数字 IO 类型。每种数字 IO 类型都支持不同的功能。表 8-1 列出了每个 IO 类型包含的功能。请参阅器件特定数据表，了解在给定封装引脚上使用哪种 IO 类型。

表 8-1. 按 IO 类型分类的数字 IO 功能

IO 结构	反转控制	驱动强度控制	迟滞控制	上拉电阻器	下拉电阻器	唤醒逻辑
标准驱动	是			是	是	
带唤醒功能的标准驱动	是			是	是	是
高驱动	是	是		是	是	是
高速	是	是		是	是	
5V 容限开漏	是		是		是	是

请注意，IOMUX 不支持 SPI POCI 引脚上的反转控制和伪开漏（输出高电平转换为高阻抗）设置。此外，IOMUX 不支持连接到 HSIO 引脚的 SPI SCLK 引脚上的反转控制。

与模拟功能共享的数字 IO

器件上的某些引脚将仅为数字引脚，并且不会将任何模拟功能连接到引脚。除了数字 IO 功能外，其他引脚还可以连接一个或多个模拟功能。从不在 IOMUX 内选择模拟功能；它们始终在相应的模拟外设内进行配置。模拟外设不了解 IOMUX 或与之交互。

通常，在同时具有数字功能的引脚上使用模拟功能时，该引脚的 IOMUX 配置应保持默认（高阻态）状态，以免干扰模拟功能的正常运行。但是，如果模拟外设也与引脚交互，则可以在引脚上使 IOMUX 有效，前提是应用软件可确保这些功能之间不会发生冲突。例如，可以在 ADC 在同一 IO 上运行转换的同时使能 IO 上的上拉或下拉电阻器。但是，无效配置会在模拟外设驱动 IO 的同时启用 IO 上的输出驱动器（例如，DAC 或 OPA 输出）。这会导致 IO 冲突。

应用软件负责确保 IOMUX 设置不会与任何可在共享焊盘上启用的模拟外设功能相冲突。

IO 切片

全功能 IO 引脚的混合信号 IO 引脚切片图如图 8-1 所示。并非所有引脚都具有模拟功能、唤醒逻辑、驱动强度控制以及上拉或下拉电阻器。有关特定引脚支持哪些功能的详细信息，请参阅特定于器件的数据表。

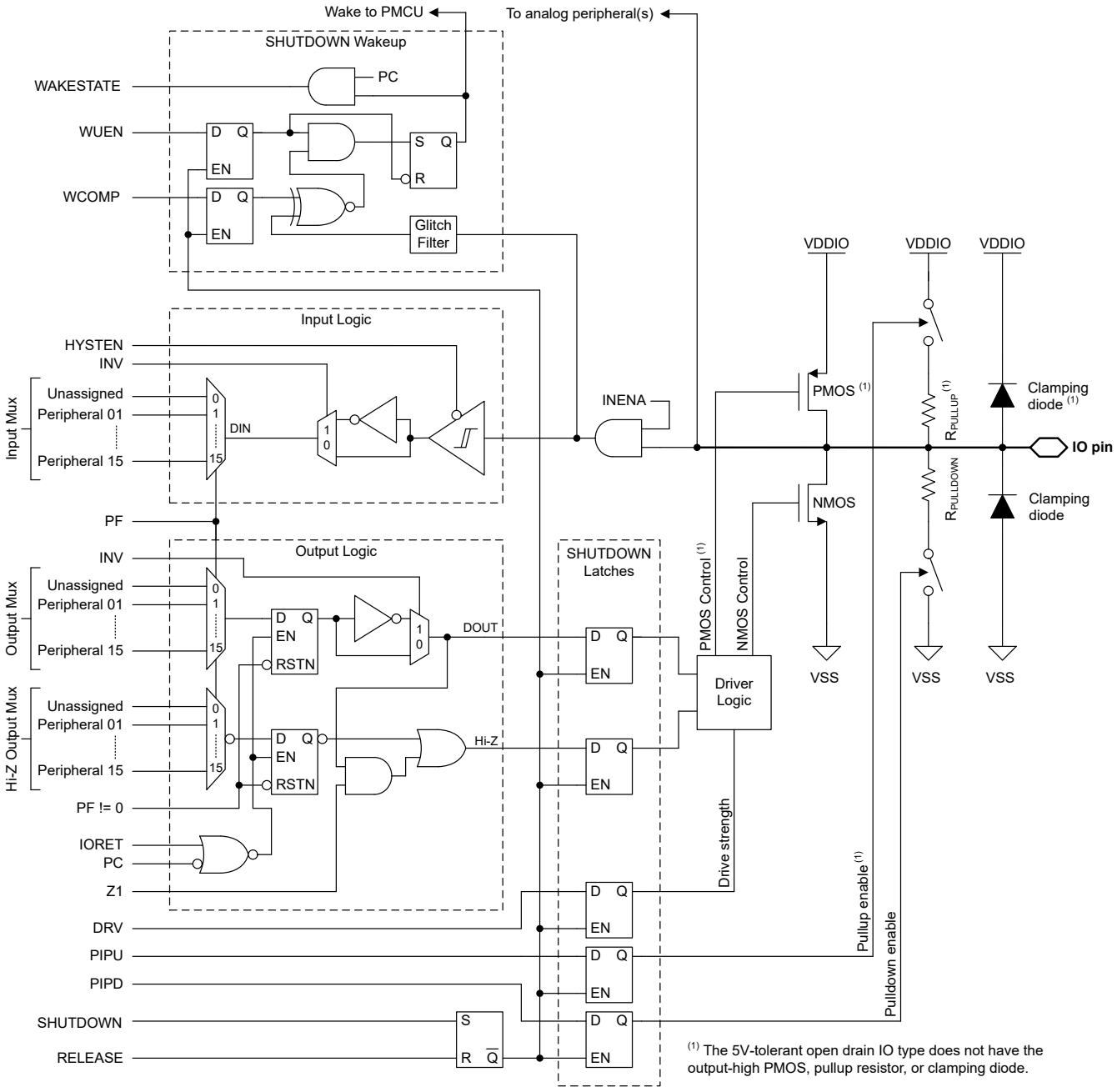


图 8-1. 超集 IO 切片

默认 IOMUX 状态

BOOTRST 之后所有数字 IO 的 IOMUX 引脚片的初始状态如下：

- 数字 IO 处于高阻抗状态

- 外设功能选择字段 (PF) 被清零（未选择外设功能）并且外设连接 (PC) 和输入使能状态被清零（禁用）
- 反转逻辑被禁用
- 高阻态输出高电平模式被禁用
- 上拉/下拉电阻（如果存在）被禁用
- 输入迟滞控制（如果存在）被禁用以降低功耗
- 驱动强度控制（如果存在）复位
- 唤醒逻辑（如果存在）被禁用

备注

SWD 引脚是上述默认状态的一个特例。SWD 调试引脚默认配置为 SWD 模式，可以在启动后切换到备用设置。请参阅 PMCU 一章 SYSCTL 部分中的节 2.4.1.4。

8.2 IOMUX 运行

器件上的每个数字 IO 在 IOMUX 外设寄存器空间中都有一个专用的 32 位 PINCM 寄存器，用于配置相应 IO 的数字功能。有关用于确定与要配置的 IO 相对应的 PINCM 寄存器索引的信息，请参阅器件特定数据表。

8.2.1 外设功能 (PF) 分配

在 BOOTRST 之后为 IO 设置初始的 IOMUX 配置时，应用软件可以通过向 PF 字段写入相应的外设选择值，同时设置与目标引脚对应的 PINCMx 寄存器中的 PC 和 INENA 位，从支持的选项中选择要连接到 IO 的数字外设。必须先设置给定外设的 IOMUX 配置，然后再初始化连接到 IO 的外设以运行。

为数字 IO 配置外设功能后，要在运行时更改该 IO 的外设功能选择，应按以下过程操作：

1. 禁用当前连接的外设功能
2. 将相应 PINCMx 寄存器中的 PC 位（输出连接位）和 INENA（输入连接位）清零
3. 将 0x0 写入 PINCMx 中的 PF 字段以清除数据路径中的逻辑
4. 通过将外设功能 ID 写入 PF 寄存器来选择新的外设功能
5. 设置 PINCMx 寄存器中的 PC 和 INENA 位以连接新选择的外设
6. 启用新选择的外设以运行

在运行时，如果需要，INENA 位可用于屏蔽从 IO 到外设的输入。清除 INENA 后，无论 IO 的外部状态如何，通过 IO 都会看到连接的外设功能为逻辑低电平 (0)。如果 IO 支持从 SHUTDOWN 模式唤醒，则 INENA 位还控制 IO 状态到 SHUTDOWN 模式唤醒逻辑的传播。

如果将外设分配给 IO，但外设本身处于禁用状态，则最后一个有效输出条件（输出逻辑电平和高阻态）会锁存在 IOMUX 输出逻辑中。启用外设后，IOMUX 将释放锁存状态，使（现在已启用的）外设的输出状态传播到 IO。PMCU 通过 IORET 信号向 IOMUX 指示外设何时进入禁用状态，该信号通过逻辑“或”与 PC 信号结合以控制输出状态锁存器。当进入 STOP 或 STANDBY 模式时，此机制可以保存电源域 1 (PD1) 外设的最后一个有效输出状态，因为 PD1 外设在进入 STOP/STANDBY 模式时始终被暂时禁用，在退出 STOP/STANDBY 模式时被重新启用。

当未选择外设功能 (PF==0) 时，输出锁存器进入复位状态，导致输出 NMOS 和 PMOS 被禁用（除了任何启用的上拉/下拉电阻之外，使 IO 引脚保持在高阻态）。请注意，上拉/下拉电阻始终不会由连接的外设或外设复用逻辑进行控制。这些电阻仅由 IOMUX 控制位进行控制（请参阅上拉/下拉电阻）。

8.2.2 逻辑高电平转换到高阻态

IOMUX 支持将连接的外设的输出高电平信号转换为 IO 引脚上的高阻态输出状态。此功能对于开漏数字输入/输出应用特别有用。启用此功能后，IO 引脚状态与外设输出的函数关系如表 8-2 所示。

表 8-2. 逻辑高电平转换到高阻态的真值表

连接的外设输出	IO 引脚状态 (Z1 = 0x0)	IO 引脚状态 (Z1 = 0x1)
逻辑低电平 (0)	输出低电平	输出低电平
逻辑高电平 (1)	输出高电平	高阻抗 (Hi-Z)

要在数字 IO 上启用逻辑高电平到高阻态的转换，请设置相应 PINCMx 寄存器中的 Z1 位。

请注意，对于可耐受 5V 电压的开漏 IO 引脚，Z1 控制不起作用，因为不存在高侧驱动器。在这些引脚上，从外设到 IO 引脚的逻辑高电平输出始终会导致高阻态。

8.2.3 逻辑反相

IOMUX 支持数字输入/输出路径的逻辑反相。对于 UART 功能或 SPI 片选功能需要相反极性的情况，逻辑反相非常实用。

要对数字 IO 启用逻辑反相，请设置相应 PINCMx 寄存器中的 INV 位。要禁用逻辑反相，请清除相应的位。默认情况下禁用逻辑反相。

当对容差为 5V 的开漏 IO 启用逻辑反相时，输出逻辑低电平状态的已连接外设将导致 IO 引脚进入高阻态。当外设应用逻辑高电平状态时，IO 引脚将进入输出低电平状态。

8.2.4 SHUTDOWN 模式唤醒逻辑

在 SHUTDOWN 模式下会禁用器件的整个稳压内核电源，器件只能从支持唤醒并具有唤醒配置的 IO、从 NRST 或从调试连接进行唤醒。用于退出 SHUTDOWN 模式的 IO 唤醒机制由 IOMUX 管理并基于电平。5V 容限开漏 IO、高驱动 IO 和某些标准驱动 IO 包括额外的唤醒逻辑，这个唤醒逻辑可用于在电平匹配时将器件从 SHUTDOWN 工作模式唤醒。

要配置支持唤醒的 IO 以便从 SHUTDOWN 模式唤醒，请执行以下操作：

1. 设置 INENA 位以使输入状态从 IO 传播到唤醒逻辑。
2. 通过设置或清除目标引脚对应的 PINCMx 寄存器中的 WCOMP 位来选择用于唤醒的比较电平。
3. 通过设置目标引脚对应的 PINCMx 寄存器中的 WUEN 位来启用唤醒。

经过前面的配置后，可以通过 SYSCTL 中的相应命令进入 SHUTDOWN 模式。如果器件上的引脚包含由 IOMUX 控制的数字 IO，则在器件进入 SHUTDOWN 模式时，这些引脚会保持当前状态。虽然在进入 SHUTDOWN 模式时会锁存数字 IO 状态，但 IOMUX 配置寄存器（所有 PINCMx 寄存器）会在稳压内核电源关闭时丢失其内容。

进入 SHUTDOWN 模式后，任何具有唤醒配置的引脚上的电平匹配都会触发从 SHUTDOWN 模式退出的序列。当器件退出 SHUTDOWN 模式时，会发生 BOR 级别的复位，但数字 IO 的状态在复位后仍然锁存，保持进入 SHUTDOWN 模式时的 IO 状态。此状态将一直保持到在 SYSCTL 中释放 IO。在 BOR 之后，SYSCTL 会捕获到复位的原因是退出 SHUTDOWN 模式，这样软件可以识别此情况并采取相应的操作来重新配置器件。

如果配置了多个引脚可从 SHUTDOWN 模式唤醒，则应用软件可以轮询在退出 SHUTDOWN 模式之前启用了唤醒的所有 IO 中的 WAKESTATE 位，从而确定是哪个配置了唤醒的 IO 产生了该唤醒。

应用软件必须按照以下过程来恢复从 SHUTDOWN 模式退出时的 IO 状态：

1. 必要时按照如下方式检查哪个 IO 触发了从 SHUTDOWN 模式唤醒：
   a. 重新配置与要测试唤醒状态的 IO 对应的 PINCMx 寄存器，并设置外设连接 (PC) 位（PC 位用于门控 WAKESTATE 指示）。
   b. 测试与待测试 IO 对应的 PINCMx 寄存器中的 WAKESTATE 位，确定该特定 IO 是否根据先前配置的 WCOMP 和 WUEN 配置接收到了 WAKE 状态。
2. 将任何其余的 IOMUX PINCM 寄存器重新配置为正确的状态。
3. 重新配置通过 IOMUX 连接到引脚的外设并启用它们。
4. 释放 SYSCTL 中的 SHUTDOWN IO 锁定。
5. 清除 PINCMx 寄存器中的 WUEN 位以复位 WAKESTATE 状态。

备注

从 SHUTDOWN 模式唤醒后，如果 WUEN 位未清零且 SYSCTL 中的关断释放位未设置，则重新进入 SHUTDOWN 模式会产生立即唤醒事件，因为未从先前的唤醒事件中清除 WAKESTATE 状态。

8.2.5 上拉/下拉电阻

大多数数字 IO 类型都提供了可编程的上拉/下拉电阻，这些电阻分别连接到 VDD/VSS。由于采用开漏配置，可耐受 5V 电压的开漏数字 IO 不提供上拉电阻。

要启用数字 IO 上的上拉或下拉电阻，请分别设置相应 PINCMx 寄存器中的 PIPU 或 PIPD 位。要禁用上拉或下拉电阻，请清除相应的位。

上拉/下拉电阻可以随时启用，其配置与外设函数配置无关。可以在更改所选外设函数时启用上拉/下拉电阻。

8.2.6 驱动强度控制

高驱动和高速数字 IO 类型具有可编程驱动强度（低驱动和高驱动）。默认驱动强度为低驱动。应用软件可以通过设置 PINCMx 寄存器中对应于目标数字 IO 的 DRV 位来请求高驱动。驱动强度控制不适用于标准驱动和开漏 IO 类型。

驱动强度控制完全独立于所选的外设功能 (PF)，可随时通过应用软件进行更改。

有关给定 IO 在每种驱动模式下的驱动性能的详细电气规格，请参阅器件特定数据表中的数字 IO 参数。

8.2.7 迟滞和逻辑电平控制

5V 容限开漏数字 IO 提供迟滞和逻辑电平控制，以支持在输入模式下使用标准 CMOS 逻辑（迟滞已启用，CMOS 逻辑电平）和 TTL 逻辑（迟滞已禁用，TTL 逻辑电平）运行。

5V 容限开漏数字 IO 的默认模式是 TLL 模式（PINCMx 寄存器中的 HYSTEN 位清零）。要在启用迟滞的 CMOS 模式下使用 5V 容限开漏数字 IO，请设置与目标 IO 对应的 PINCMx 寄存器中的 HYSTEN 位。

TTL 模式（左）和 CMOS 模式（右）之间的输入逻辑电平差异如图 8-2 所示。

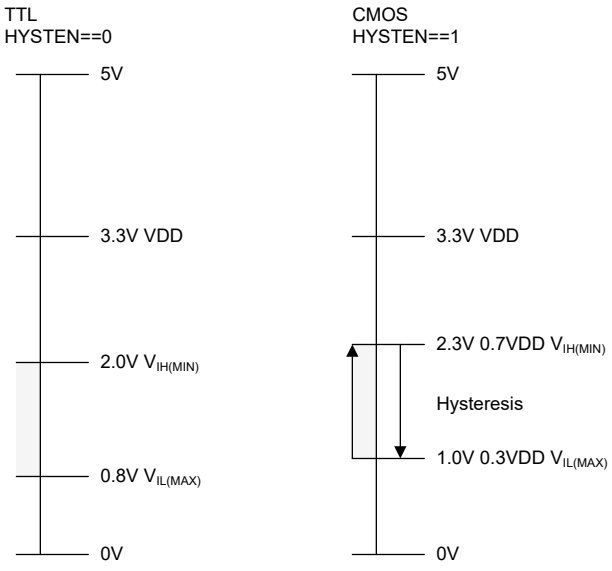


图 8-2. 输入逻辑电平 - 5V 容限开漏数字 IO

8.3 IOMUX (PINCMx) 寄存器格式

8.3.1 PINCM（偏移 = 4h）[复位 = X]

引脚控制管理寄存器

图 8-3. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED			DRV	HYSTEN	INENA	PIPU	PIPD
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
保留		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	6	5	4	3	2	1	0
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

表 8-3. PINCM 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	WCOMP	R/W	0h	唤醒比较值位 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能位 0h = 唤醒禁用。 1h = 唤醒启用
26	INV	R/W	0h	数据反转选择 0h = 数据反转禁用。 1h = 数据反转启用
25	HIZ1	R/W	0h	在该位启用时，高输出值将对输出进行三态处理 0h = 开漏禁用。 1h = 开漏启用。
24-21	RESERVED	R/W	0h	
20	DRV	R/W	0h	驱动强度控制选择，仅用于 HS IOCELL 0h = 选择 0 的驱动设置 1h = 选择 1 的驱动设置
19	HYSTEN	R/W	0h	迟滞使能控制选择 0h = 迟滞禁用。 1h = 迟滞启用
18	INENA	R/W	0h	输入使能控制选择 0h = 输入使能禁用。 1h = 输入使能启用。
17	PIPU	R/W	0h	上拉控制选择 0h = 上拉禁用。 1h = 上拉启用
16	PIPD	R/W	0h	下拉控制选择 0h = 下拉禁用。 1h = 下拉启用
15-14	RESERVED	R/W	0h	

表 8-3. PINCM 字段说明 (continued)

位	字段	类型	复位	说明
13	WAKESTAT	R	0h	这将 IOPAD WAKEUP 信号作为状态位。 0h = 唤醒源并非来自此 IOCELL 1h = 唤醒源来自此 IOCELL
12-8	保留	R/W	0h	
7	PC	R/W	0h	外设“已连接” 0h = 外设的输出（及其输出使能）不会传播到 IOCELL 1h = 数据流的输出锁存器将为“透明”
6	RESERVED	R/W	0h	
5-0	PF	R/W	0h	P 通道功能选择位 0h = 保留为未连接 3Fh = 一个可连接至此引脚的按功能编码。

8.4 IOMUX 寄存器

表 8-4 列出了 IOMUX 寄存器的存储器映射寄存器。表 8-4 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 8-4. IOMUX 寄存器

偏移	缩写	寄存器名称	组	部分
4h	PINCM	SECCFG 区域中的引脚控制管理寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 8-5 显示了适用于此部分中访问类型的代码。

表 8-5. IOMUX 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
复位或默认值		
-n		复位后的值或默认值

8.4.1 PINCM（偏移 = 4h）[复位 = X]

图 8-4 展示了 PINCM，表 8-6 中对此进行了介绍。

返回到汇总表。

引脚控制管理寄存器

图 8-4. PINCM

31	30	29	28	27	26	25	24
RESERVED			WCOMP	WUEN	INV	HIZ1	RESERVED
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
		RESERVED	DRV	HYSTEN	INENA	PIPU	PIPD
		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
保留		WAKESTAT	RESERVED				
R/W-0h		R-0h	R/W-0h				
7	6	5	4	3	2	1	0
PC	RESERVED	PF					
R/W-0h	R/W-0h	R/W-0h					

表 8-6. PINCM 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	WCOMP	R/W	0h	唤醒比较值位 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能位 0h = 唤醒禁用。 1h = 唤醒启用
26	INV	R/W	0h	数据反转选择 0h = 数据反转禁用。 1h = 数据反转启用
25	HIZ1	R/W	0h	在该位启用时，高输出值将对输出进行三态处理 0h = 开漏禁用。 1h = 开漏启用。
24	保留	R/W	0h	
21	保留	R/W	0h	
20	DRV	R/W	0h	驱动强度控制选择，仅用于 HS IOCELL 0h = 选择 0 的驱动设置 1h = 选择 1 的驱动设置
19	HYSTEN	R/W	0h	迟滞使能控制选择 0h = 迟滞禁用。 1h = 迟滞启用
18	INENA	R/W	0h	输入使能控制选择 0h = 输入使能禁用。 1h = 输入使能启用。
17	PIPU	R/W	0h	上拉控制选择 0h = 上拉禁用。 1h = 上拉启用
16	PIPD	R/W	0h	下拉控制选择 0h = 下拉禁用。 1h = 下拉启用

表 8-6. PINCM 字段说明 (continued)

位	字段	类型	复位	说明
15-14	RESERVED	R/W	0h	
13	WAKESTAT	R	0h	这将 IOPAD WAKEUP 信号作为状态位。 0h = 唤醒源并非来自此 IOCELL 1h = 唤醒源来自此 IOCELL
12-8	保留	R/W	0h	
7	PC	R/W	0h	外设“已连接” 0h = 外设的输出（及其输出使能）不会传播到 IOCELL 1h = 数据流的输出锁存器将为“透明”
6	RESERVED	R/W	0h	
5-0	PF	R/W	0h	外设功能选择位 0h = 保留为未连接 3Fh = 一个可连接至此引脚的按功能编码。

This page intentionally left blank.

章节 9
GPIO

GPIO 外设为用户提供了一种通过器件引脚写入数据和读取数据的方法。它还提供了一种在器件处于低功耗状态时检测唤醒事件的方法。本章介绍了 GPIO 外设的运行情况。

9.1 GPIO 概述

GPIO 用于从器件引脚读入数字数据以及将数字数据发出到器件引脚。

GPIO 模块的特性包括：

- 从 CPU 访问 MMR 的零等待状态
- 无需在软件中使用“读取、修改、写入”结构，即可设置/清零/切换多个位
- 无需在软件中使用“读取、修改、写入”结构，即可直接写入到各个 GPIO 输出位 (DOUT)
- 无需在软件中使用屏蔽，即可直接读取各个 GPIO 输入位 (DIN) 的比较结果
- DOUT 可供 DMA 用于在指定引脚上生成预定义的输出序列
- “快速唤醒”功能支持通过任意 GPIO 端口从 STOP 和 STANDBY 模式进行低功耗唤醒
- 用户控制的输入滤波（可按 IO 配置）
- 通过事件发布者和事件订阅者与设备事件结构互连（仅限 GPIOA 实例）

图 9-1 展示了 GPIO 外设的方框图。

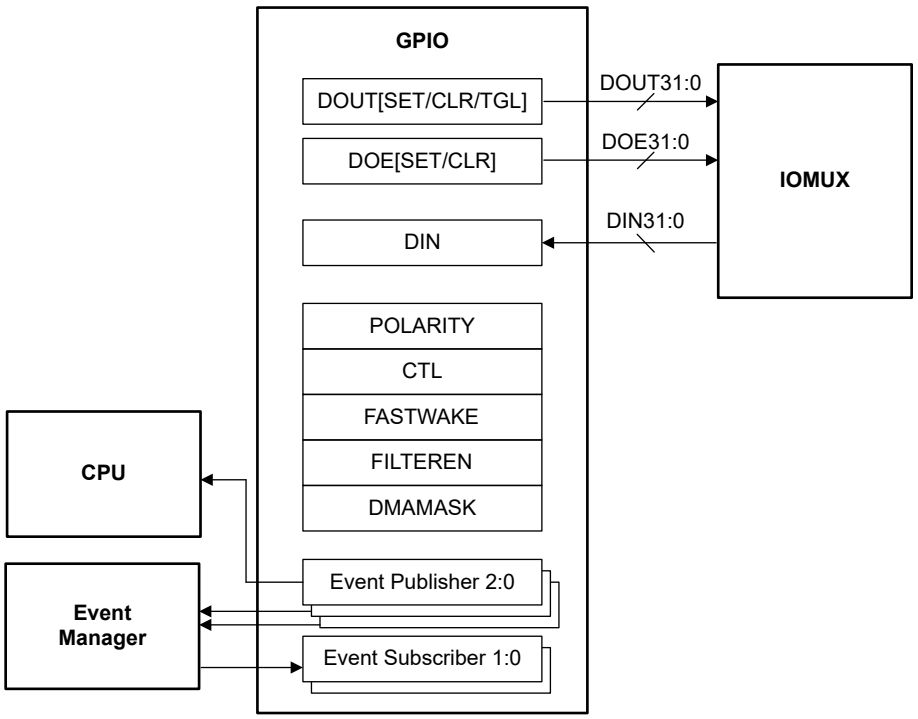


图 9-1. GPIO 方框图

备注

MSPM0 平台的 GPIO 模块不管理完整的数字 IO 功能（例如，上拉、下拉或其他功能多路复用）。有关完整的数字 IO 控制详细信息，请参阅章节 8。与任何其他外设类似，GPIO 的输入和输出（具有输出使能功能）允许 GPIO 与 IOMUX 连接以连接到 IO 引脚。

9.2 GPIO 操作

可以通过用户软件对 GPIO 外设进行配置。以下各节讨论了 GPIO 的设置和操作。

9.2.1 GPIO 端口

MSPM0 平台中的一个 GPIO 外设实例支持多达 32 个数据输入/输出 (DIO) 位。对于具有超过 32 个 GPIO 的器件，GPIO 外设的多个实例用于对所有器件引脚寻址。GPIO 端口和位名称直接映射到器件数据表的*引脚配置和功能*部分中与每个器件引脚关联的信号名称。

表 9-1. GPIO 端口和器件引脚映射

GPIO 端口和位名称	器件引脚信号名称
GPIO 端口 A 位 0 (DIO0)	PA0
GPIO 端口 A 位 1 (DIO1)	PA1
GPIO 端口 B 位 0 (DIO0)	PB0
GPIO 端口 B 位 1 (DIO1)	PB1
GPIO 端口 x 位 y (DIOy)	Pxy

9.2.2 GPIO 读取/写入接口

GPIO 外设具有多项功能和专用寄存器，无需在软件中执行“读取、修改、写入”结构即可进行高级位操作。下面概述了这些功能。

读取接口

DIN31_0 寄存器是给定 GPIO 端口的数据输入寄存器。读取 DIN31_0 寄存器中的位对应于读取相关器件引脚信号上的信号电压电平。GPIO 外设为 DIN 的所有位提供 single-bit 字节读取地址。这些寄存器名为 DIN31_28、DIN27_24、…、DIN3_0，本质上是 DIN31_0 数据输入寄存器中所有位的别名寄存器。读取 DIN31_28 寄存器将授予您对 DIN31:28 的字节级访问权限，这让您可以在不屏蔽的情况下执行直接读取比较。

写入接口

在许多情况下，根据用户系统为器件引脚信号写入特定值很有用。寄存器 DOUT31_0 是给定 GPIO 端口的物理数据输出寄存器。当通过 DOE31_0 寄存器启用输出时，设置 DOUT31_0 寄存器中的位，会设置相应的器件引脚信号。相反，将 DOUT31_0 寄存器中的位清零会清除相应的器件引脚信号。

GPIO 外设还为 DOUT 的所有位提供 single-bit 字节写入地址。这些寄存器名为 DOUT31_28、DOUT27_24、…、DOUT3_0，本质上是 DOUT31_0 数据输出寄存器中所有位的别名寄存器。写入 DOUT31_28 寄存器将授予您对 DOUT31:28 的字节级访问权限，这让您可以直接写入这些单独的位，而无需使用读取、修改、写入结构。

设置/清零/切换

与上述用例类似，在应用中设置、清零或切换器件引脚或引脚集合时也很有用。MSPM0 平台让您可以使用 DOUTSET31_0、DOUTCLR31_0 和 DOUTTGL31_0 寄存器来在一个写入周期内执行这些功能。通过设置这些寄存器中的位，可以在写入“0”无效时实现相应的功能。此功能让您可以每个 GPIO 端口使用单个命令，直接设置、清零和切换 DOUT31_0 寄存器中的任何位。

与设置和清零数据输出位的方式相同，也可以通过一些寄存器来实现与数据输出使能寄存器 DOE31_0 相同的功能。通过向 DOESET31_0 和 DOECLR31_0 寄存器中的位写入“1”，可以在写入“0”无效时实现相应的功能。此功能让您可以每个 GPIO 端口使用单个命令，直接设置或清零 DOE31_0 寄存器中的任何位。

9.2.3 GPIO 输入干扰滤波和同步

GPIO 模块以 ULPCLK（PD0 总线时钟）速率评估输入引脚的状态，在将 GPIO 状态传递给输入干扰滤波器之前，通过 2 级同步器将引脚状态同步到 ULPCLK。

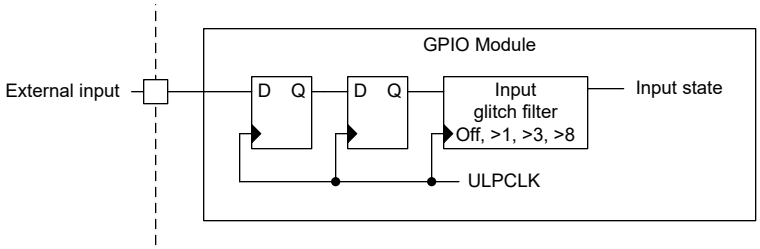


图 9-2. GPIO 输入同步器

提供可编程输入干扰滤波器，用于抑制数字输入引脚上的噪声。干扰滤波器以 ULPCLK 速率运行。可实现四级用户指定的输入滤波：

- 无滤波采样输入（经可靠检测后得到的最小脉冲宽度为一个由引脚状态与 ULPCLK 同步引起的 ULPCLK 周期+在为 STANDBY0/1、STOP1/2 和 SLEEP2 模式启用快速唤醒的情况下，从异步请求边沿到第一个 32MHz MCLK 边沿的延迟时间）
- 滤除不大于 1 个 ULPCLK 周期的同步输入
- 滤除不大于 3 个 ULPCLK 周期的同步输入
- 滤除不大于 8 个 ULPCLK 周期的同步输入

此功能使用户能够在需要滤除输入引脚上的快速切换的情况下，在硬件中轻松实现输入滤波。通过 FILTEREN31_16 和 FILTEREN15_0 寄存器中的位字段，用户可以配置相应 GPIO 位所需的滤波级别。

由于同步中的 1 个不确定性 ULPCLK 周期，在某些情况下可以传递相同脉冲长度的输入脉冲，而在其他情况下可以进行滤波。

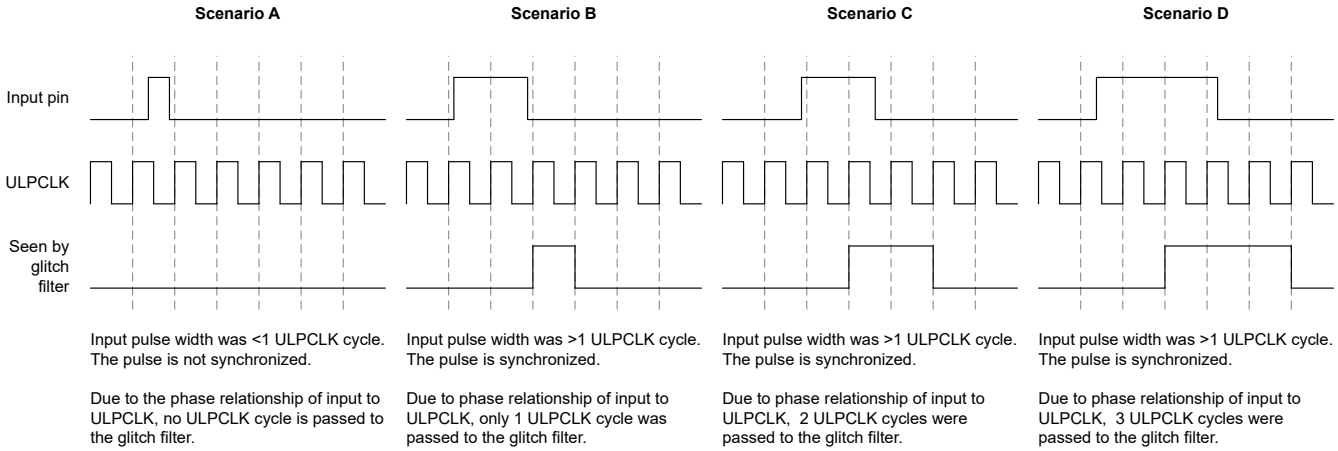


图 9-3. GPIO 输入同步器和干扰滤波场景

- 在场景 A 中，输入脉冲小于一个 ULPCLK 周期。小于一个 ULPCLK 周期的脉冲可能无法捕获。为了确保始终捕获 GPIO 输入，GPIO 输入脉冲宽度必须大于 ULPCLK 周期。
- 在场景 B 中，输入脉冲的长度接近两个 ULPCLK 周期，但由于上升沿发生在 ULPCLK 边沿之后，GPIO 同步器仅将输入引脚视为在 1 个 ULPCLK 周期内一直处于高电平。禁用干扰滤波器时，干扰滤波器不会滤除这种场景，但干扰滤波器值 >1 时将导致此脉冲被滤除。相反，在场景 C 中，相同的输入脉冲宽度会导致输入引脚在两个 ULPCLK 周期内被视为高电平，因为上升沿发生在 ULPCLK 边沿之前。在这种情况下，当指定干扰滤波器值 >1 时，不会滤除此场景。
- 在场景 D 中，三个 ULPCLK 周期被传递到干扰滤波器。在这种情况下，当指定干扰滤波器值 >1 时，不会滤除该场景，但当干扰滤波器值 >3 或 >8 时则会滤除该场景。

备注

当启用快速唤醒模式（在输入引脚活动时异步请求 SYSOSC）时，ULPCLK 将从关闭（如 STANDBY1 中的情况）或 32kHz（如 STANDBY0 中的情况）切换到 32MHz，导致输入同步逻辑和干扰滤波器在一段时间延迟后以 32MHz 运行。有关异步快速时钟请求唤醒时间，请参阅器件特定数据表，并在使用快速唤醒时将此时间预算到任何最小脉冲宽度计算中。

9.2.4 GPIO 快速唤醒

MSPM0 GPIO 外设中的“快速唤醒”功能使 GPIO 模块能够保持低功耗状态，并在不需要高速时钟的情况下检测器件引脚上的中断事件。这使得该器件能够支持通过任一 GPIO 引脚从低功耗模式（例如停止和待机模式）快速唤醒。

可以使用 FASTWAKE 寄存器来逐位启用“快速唤醒”功能。通过设置 FASTWAKE 寄存器中的位，可启用相应的器件引脚信号，以支持快速唤醒功能。CTL 寄存器包含一个名为 FASTWAKEONLY 的位字段，该位字段支持对“快速唤醒”功能进行全局控制。通过设置 FASTWAKEONLY 位，可启用相应 GPIO 端口中的所有位，以支持快速唤醒功能。

备注

请勿在 GPIO 中启用“快速唤醒”的同时阻止 SYSCTL 中的异步快速时钟请求。启用“快速唤醒”后，GPIO 预期与 SYSCTL 握手以获得快速时钟。如果 SYSCTL 忽略该请求，则在 SYSCTL 完成异步快速时钟请求握手之前，GPIO 将不会收到时钟。

9.2.5 GPIO DMA 接口

GPIO 外设支持对 DOUT31_0 寄存器进行 DMA 写入访问。此功能允许用户在指定的器件引脚上生成预定义的输出序列。某些应用需要预加载的 GPIO 引脚更改序列，并且 MSPM0 平台允许 DMA 运行该序列，以便 CPU 能够保持睡眠状态并节省能源。

DMAMASK 寄存器用于指示允许 DMA 修改哪些 GPIO 位。通过设置 DMAMASK 寄存器中的位，便可以通过 DMA 修改相应的 DOUT 位。

备注

无论 DMAMASK 值如何，CPU 都可以向任何 DOUT31_0 位写入数据。

在 DMA 和 CPU 同时尝试访问和修改 DOUT31_0 寄存器的情况下，用户有责任管理 DMA 和 CPU 总线事务，这些事务以要修改的同一位为目标。

- 如果已设置 DMAMASK 位，则将优先由 DMA 修改相应的 DOUT 位。
- 如果 DMAMASK 位已清零，则将优先由 CPU 修改相应的 DOUT 位。

9.2.6 事件发布者和订阅者

有三个独立的事件发布者可用于 GPIOx 外设：

1. 第一个事件发布者 (CPU_INT)
   - 用于生成 CPU 中断
   - 在软件读取 IIDX 寄存器或写入相应的 ICLR 寄存器位时，会清除中断 (RIS) 标志。
   - 可通过 POLARITY 寄存器为每个 GPIO 位单独指定一个到 CPU 的事件：
     - 0：禁用
     - 1：上升事件
     - 2：下降事件
     - 3：上升或下降事件
2. 第二个事件发布者 (GEN_EVENT0)，仅在 GPIOA 上可用
   - 使用与 CPU_INT 相同的 POLARITY 寄存器定义
   - 应用于 GPIO 位 15 直至 0 (DIO15:0)

3. 第三个事件发布者 (GEN_EVENT1)，仅在 GPIOA 上可用
   - 使用与 CPU_INT 相同的 POLARITY 寄存器定义
   - 应用于 GPIO 位 31 直至 16 (DIO31:16)

有 **2 个事件订阅者**（仅在 GPIOA 外设上可用）：

1. 第一个事件订阅者 (FSUB_0)
   - 可以指示特定引脚在发生事件时更改状态
   - 订阅者事件只能使单个位执行操作
   - 应用于 GPIO 位 15 直至 0 (DIO15:0)
   - SUB0CFG 寄存器用于启用 FSUB_0 事件并定义特定 GPIO 引脚的输出策略
2. 第二个事件订阅者 (FSUB_1)
   - 可以指示特定引脚在发生事件时更改状态
   - 订阅者事件只能使单个位执行操作
   - 应用于 GPIO 位 31 直至 16 (DIO31:16)
   - SUB1CFG 寄存器用于启用 FSUB_1 事件并定义特定 GPIO 引脚的输出策略

9.3 GPIO 寄存器

表 9-2 列出了 GPIO 寄存器的存储器映射寄存器。表 9-2 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 9-2. GPIO 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者端口 0		转到
404h	FSUB_1	订阅者端口 1		转到
444h	FPUB_0	发布者端口 0		转到
448h	FPUB_1	发布者端口 1		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1010h	CLKOVR	时钟覆盖		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT 0	转到
1058h	IMASK	中断屏蔽	GEN_EVENT 0	转到
1060h	RIS	原始中断状态	GEN_EVENT 0	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT 0	转到
1070h	ISET	中断设置	GEN_EVENT 0	转到
1078h	ICLR	中断清除	GEN_EVENT 0	转到
1080h	IIDX	中断索引	Gen_EVENT1	转到
1088h	IMASK	中断屏蔽	Gen_EVENT1	转到
1090h	RIS	原始中断状态	Gen_EVENT1	转到
1098h	MIS	已屏蔽中断状态	Gen_EVENT1	转到
10A0h	ISET	中断设置	Gen_EVENT1	转到
10A8h	ICLR	中断清除	Gen_EVENT1	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1200h	DOUT3_0	数据输出 3 到 0		转到
1204h	DOUT7_4	数据输出 7 到 4		转到
1208h	DOUT11_8	数据输出 11 到 8		转到
120Ch	DOUT15_12	数据输出 15 到 12		转到
1210h	DOUT19_16	数据输出 19 到 16		转到
1214h	DOUT23_20	数据输出 23 到 20		转到
1218h	DOUT27_24	数据输出 27 到 24		转到

表 9-2. GPIO 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
121Ch	DOUT31_28	数据输出 31 到 28		转到
1280h	DOUT31_0	数据输出 31 到 0		转到
1290h	DOUTSET31_0	数据输出设置 31 到 0		转到
12A0h	DOUTCLR31_0	数据输出清除 31 到 0		转到
12B0h	DOUTTGL31_0	数据输出切换 31 到 0		转到
12C0h	DOE31_0	数据输出启用 31 到 0		转到
12D0h	DOESET31_0	数据输出启用设置 31 到 0		转到
12E0h	DOECLR31_0	数据输出启用清除 31 到 0		转到
1300h	DIN3_0	数据输入 3 到 0		转到
1304h	DIN7_4	数据输入 7 到 4		转到
1308h	DIN11_8	数据输入 11 到 8		转到
130Ch	DIN15_12	数据输入 15 到 12		转到
1310h	DIN19_16	数据输入 19 到 16		转到
1314h	DIN23_20	数据输入 23 到 20		转到
1318h	DIN27_24	数据输入 27 到 24		转到
131Ch	DIN31_28	数据输入 31 到 28		转到
1380h	DIN31_0	数据输入 31 到 0		转到
1390h	POLARITY15_0	极性 15 到 0		转到
13A0h	POLARITY31_16	极性 31 到 16		转到
1400h	CTL	FAST WAKE GLOBAL EN		转到
1404h	FASTWAKE	FAST WAKE ENABLE		转到
1500h	SUB0CFG	订阅者 0 配置		转到
1508h	FILTEREN15_0	滤波器使能 15 到 0		转到
150Ch	FILTEREN31_16	滤波器使能 31 到 16		转到
1510h	DMAMASK	DMA 写入屏蔽		转到
1520h	SUB1CFG	订阅者 1 配置		转到

复杂的位访问类型经过编码可适应小型表单元。表 9-3 显示了适用于此部分中访问类型的代码。

表 9-3. GPIO 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

9.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 9-4 展示了 FSUB_0，表 9-4 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 9-4. FSUB_0

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

表 9-4. FSUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

9.3.2 FSUB_1（偏移 = 404h）[复位 = 00000000h]

图 9-5 展示了 FSUB_1，表 9-5 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 9-5. FSUB_1

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

表 9-5. FSUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

9.3.3 FPUB_0（偏移 = 444h）[复位 = 00000000h]

图 9-6 展示了 FPUB_0，表 9-6 中对此进行了介绍。

返回到汇总表。

发布者端口

图 9-6. FPUB_0

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

表 9-6. FPUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

9.3.4 FPUB_1（偏移 = 448h）[复位 = 00000000h]

图 9-7 展示了 FPUB_1，表 9-7 中对此进行了介绍。

返回到汇总表。

发布者端口

图 9-7. FPUB_1

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

表 9-7. FPUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

9.3.5 PWREN（偏移 = 800h）[复位 = 00000000h]

图 9-8 展示了 PWREN，表 9-8 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 9-8. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-							K-0h

表 9-8. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

9.3.6 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 9-9 展示了 RSTCTL，表 9-9 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 9-9. RSTCTL

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

表 9-9. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

9.3.7 STAT（偏移 = 814h）[复位 = 00000000h]

图 9-10 展示了 STAT，表 9-10 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 9-10. STAT

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

表 9-10. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

9.3.8 CLKOVR (偏移 = 1010h) [复位 = 00000000h]

图 9-11 展示了 CLKOVR，表 9-11 中对此进行了介绍。

返回到汇总表。

该寄存器覆盖了外设向系统发出的功能时钟请求

图 9-11. CLKOVR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED						RUN_STOP	OVERRIDE
R/W-0h						R/W-0h	R/W-0h

表 9-11. CLKOVR 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	RUN_STOP	R/W	0h	如果启用了 OVERRIDE，该寄存器用于手动控制外设向系统发出的时钟请求 0h = 运行/取消门控功能时钟 1h = 停止/门控功能时钟
0	OVERRIDE	R/W	0h	解锁 RUN_STOP 的功能以覆盖自动外设时钟请求 0h = 禁用覆盖 1h = 启用覆盖

9.3.9 PDBGCTL（偏移 = 1018h）[复位 = 00000001h]

图 9-12 展示了 PDBGCTL，表 9-12 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 9-12. PDBGCTL

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

表 9-12. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

9.3.10 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 9-13 展示了 IIDX，表 9-13 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 9-13. IIDX

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

表 9-13. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 没有位置位意味着没有挂起的中断请求 1h = DIO0 中断 2h = DIO1 中断 3h = DIO2 中断 4h = DIO3 中断 5h = DIO4 中断 6h = DIO5 中断 7h = DIO6 中断 8h = DIO7 中断 9h =DIO8 中断 Ah = DIO9 中断 BH = DIO10 中断 Ch = DIO11 中断 DH = DIO12 中断 EH = DIO13 中断 FH = DIO14 中断 10h = DIO15 中断 11h = DIO16 中断 12h = DIO17 中断 13h = DIO18 中断 14h = DIO19 中断 15h = DIO20 中断 16h = DIO21 中断 17h = DIO22 中断 18h = DIO23 中断 19h = DIO24 中断 1Ah = DIO25 中断 1Bh = DIO26 中断 1Ch = DIO27 中断 1Dh = DIO28 中断 1Eh = DIO29 中断 1Fh = DIO30 中断 20h = DIO31 中断

9.3.11 IMASK（偏移 = 1028h）[复位= 00000000h]

图 9-14 展示了 IMASK，表 9-14 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 9-14. IMASK

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-14. IMASK 字段说明

位	字段	类型	复位	说明
31	DIO31	R/W	0h	DIO31 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
30	DIO30	R/W	0h	DIO30 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
29	DIO29	R/W	0h	DIO29 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
28	DIO28	R/W	0h	DIO28 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
27	DIO27	R/W	0h	DIO27 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
26	DIO26	R/W	0h	DIO26 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
25	DIO25	R/W	0h	DIO25 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
24	DIO24	R/W	0h	DIO24 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
23	DIO23	R/W	0h	DIO23 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

表 9-14. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	R/W	0h	DIO22 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
21	DIO21	R/W	0h	DIO21 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
20	DIO20	R/W	0h	DIO20 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
19	DIO19	R/W	0h	DIO19 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
18	DIO18	R/W	0h	DIO18 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
17	DIO17	R/W	0h	DIO17 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
16	DIO16	R/W	0h	DIO16 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
15	DIO15	R/W	0h	DIO15 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
14	DIO14	R/W	0h	DIO14 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
13	DIO13	R/W	0h	DIO13 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
12	DIO12	R/W	0h	DIO12 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
11	DIO11	R/W	0h	DIO11 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
10	DIO10	R/W	0h	DIO10 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
9	DIO9	R/W	0h	DIO9 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
8	DIO8	R/W	0h	DIO8 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
7	DIO7	R/W	0h	DIO7 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
6	DIO6	R/W	0h	DIO6 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

表 9-14. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
5	DIO5	R/W	0h	DIO5 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
4	DIO4	R/W	0h	DIO4 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
3	DIO3	R/W	0h	DIO3 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
2	DIO2	R/W	0h	DIO2 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
1	DIO1	R/W	0h	DIO1 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
0	DIO0	R/W	0h	DIO0 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

9.3.12 RIS（偏移 = 1030h）[复位 = 00000000h]

图 9-15 展示了 RIS，表 9-15 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 9-15. RIS

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 9-15. RIS 字段说明

位	字段	类型	复位	说明
31	DIO31	R	0h	DIO31 事件 0h = 未发生 DIO31 事件 1h = 已发生 DIO31 事件
30	DIO30	R	0h	DIO30 事件 0h = 未发生 DIO30 事件 1h = 已发生 DIO30 事件
29	DIO29	R	0h	DIO29 事件 0h = 未发生 DIO29 事件 1h = 已发生 DIO29 事件
28	DIO28	R	0h	DIO28 事件 0h = 未发生 DIO28 事件 1h = 已发生 DIO28 事件
27	DIO27	R	0h	DIO27 事件 0h = 未发生 DIO27 事件 1h = 已发生 DIO27 事件
26	DIO26	R	0h	DIO26 事件 0h = 未发生 DIO26 事件 1h = 已发生 DIO26 事件
25	DIO25	R	0h	DIO25 事件 0h = 未发生 DIO25 事件 1h = 已发生 DIO25 事件
24	DIO24	R	0h	DIO24 事件 0h = 未发生 DIO24 事件 1h = 已发生 DIO24 事件
23	DIO23	R	0h	DIO23 事件 0h = 未发生 DIO23 事件 1h = 已发生 DIO23 事件

表 9-15. RIS 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	R	0h	DIO22 事件 0h = 未发生 DIO22 事件 1h = 已发生 DIO22 事件
21	DIO21	R	0h	DIO21 事件 0h = 未发生 DIO21 事件 1h = 已发生 DIO21 事件
20	DIO20	R	0h	DIO20 事件 0h = 未发生 DIO20 事件 1h = 已发生 DIO20 事件
19	DIO19	R	0h	DIO19 事件 0h = 未发生 DIO19 事件 1h = 已发生 DIO19 事件
18	DIO18	R	0h	DIO18 事件 0h = 未发生 DIO18 事件 1h = 已发生 DIO18 事件
17	DIO17	R	0h	DIO17 事件 0h = 未发生 DIO17 事件 1h = 已发生 DIO17 事件
16	DIO16	R	0h	DIO16 事件 0h = 未发生 DIO16 事件 1h = 已发生 DIO16 事件
15	DIO15	R	0h	DIO15 事件 0h = 未发生 DIO15 事件 1h = 已发生 DIO15 事件
14	DIO14	R	0h	DIO14 事件 0h = 未发生 DIO14 事件 1h = 已发生 DIO14 事件
13	DIO13	R	0h	DIO13 事件 0h = 未发生 DIO13 事件 1h = 已发生 DIO13 事件
12	DIO12	R	0h	DIO12 事件 0h = 未发生 DIO12 事件 1h = 已发生 DIO12 事件
11	DIO11	R	0h	DIO11 事件 0h = 未发生 DIO11 事件 1h = 已发生 DIO11 事件
10	DIO10	R	0h	DIO10 事件 0h = 未发生 DIO10 事件 1h = 已发生 DIO10 事件
9	DIO9	R	0h	DIO9 事件 0h = 未发生 DIO9 事件 1h = 已发生 DIO9 事件
8	DIO8	R	0h	DIO8 事件 0h = 未发生 DIO8 事件 1h = 已发生 DIO8 事件
7	DIO7	R	0h	DIO7 事件 0h = 未发生 DIO7 事件 1h = 已发生 DIO7 事件
6	DIO6	R	0h	DIO6 事件 0h = 未发生 DIO6 事件 1h = 已发生 DIO6 事件

表 9-15. RIS 字段说明 (continued)

位	字段	类型	复位	说明
5	DIO5	R	0h	DIO5 事件 0h = 未发生 DIO5 事件 1h = 已发生 DIO5 事件
4	DIO4	R	0h	DIO4 事件 0h = 未发生 DIO4 事件 1h = 已发生 DIO4 事件
3	DIO3	R	0h	DIO3 事件 0h = 未发生 DIO3 事件 1h = 已发生 DIO3 事件
2	DIO2	R	0h	DIO2 事件 0h = 未发生 DIO2 事件 1h = 已发生 DIO2 事件
1	DIO1	R	0h	DIO1 事件 0h = 未发生 DIO1 事件 1h = 已发生 DIO1 事件
0	DIO0	R	0h	DIO0 事件 0h = 未发生 DIO0 事件 1h = 已发生 DIO0 事件

9.3.13 MIS（偏移 = 1038h）[复位 = 00000000h]

图 9-16 展示了 MIS，表 9-16 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 9-16. MIS

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 9-16. MIS 字段说明

位	字段	类型	复位	说明
31	DIO31	R	0h	DIO31 事件 0h = 未发生 DIO31 事件 1h = 已发生 DIO31 事件
30	DIO30	R	0h	DIO30 事件 0h = 未发生 DIO30 事件 1h = 已发生 DIO30 事件
29	DIO29	R	0h	DIO29 事件 0h = 未发生 DIO29 事件 1h = 已发生 DIO29 事件
28	DIO28	R	0h	DIO28 事件 0h = 未发生 DIO28 事件 1h = 已发生 DIO28 事件
27	DIO27	R	0h	DIO27 事件 0h = 未发生 DIO27 事件 1h = 已发生 DIO27 事件
26	DIO26	R	0h	DIO26 事件 0h = 未发生 DIO26 事件 1h = 已发生 DIO26 事件
25	DIO25	R	0h	DIO25 事件 0h = 未发生 DIO25 事件 1h = 已发生 DIO25 事件
24	DIO24	R	0h	DIO24 事件 0h = 未发生 DIO24 事件 1h = 已发生 DIO24 事件
23	DIO23	R	0h	DIO23 事件 0h = 未发生 DIO23 事件 1h = 已发生 DIO23 事件
22	DIO22	R	0h	DIO22 事件 0h = 未发生 DIO22 事件 1h = 已发生 DIO22 事件

表 9-16. MIS 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	R	0h	DIO21 事件 0h = 未发生 DIO21 事件 1h = 已发生 DIO21 事件
20	DIO20	R	0h	DIO20 事件 0h = 未发生 DIO20 事件 1h = 已发生 DIO20 事件
19	DIO19	R	0h	DIO19 事件 0h = 未发生 DIO19 事件 1h = 已发生 DIO19 事件
18	DIO18	R	0h	DIO18 事件 0h = 未发生 DIO18 事件 1h = 已发生 DIO18 事件
17	DIO17	R	0h	DIO17 事件 0h = 未发生 DIO17 事件 1h = 已发生 DIO17 事件
16	DIO16	R	0h	DIO16 事件 0h = 未发生 DIO16 事件 1h = 已发生 DIO16 事件
15	DIO15	R	0h	DIO15 事件 0h = 未发生 DIO15 事件 1h = 已发生 DIO15 事件
14	DIO14	R	0h	DIO14 事件 0h = 未发生 DIO14 事件 1h = 已发生 DIO14 事件
13	DIO13	R	0h	DIO13 事件 0h = 未发生 DIO13 事件 1h = 已发生 DIO13 事件
12	DIO12	R	0h	DIO12 事件 0h = 未发生 DIO12 事件 1h = 已发生 DIO12 事件
11	DIO11	R	0h	DIO11 事件 0h = 未发生 DIO11 事件 1h = 已发生 DIO11 事件
10	DIO10	R	0h	DIO10 事件 0h = 未发生 DIO10 事件 1h = 已发生 DIO10 事件
9	DIO9	R	0h	DIO9 事件 0h = 未发生 DIO9 事件 1h = 已发生 DIO9 事件
8	DIO8	R	0h	DIO8 事件 0h = 未发生 DIO8 事件 1h = 已发生 DIO8 事件
7	DIO7	R	0h	DIO7 事件 0h = 未发生 DIO7 事件 1h = 已发生 DIO7 事件
6	DIO6	R	0h	DIO6 事件 0h = 未发生 DIO6 事件 1h = 已发生 DIO6 事件
5	DIO5	R	0h	DIO5 事件 0h = 未发生 DIO5 事件 1h = 已发生 DIO5 事件

表 9-16. MIS 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	R	0h	DIO4 事件 0h = 未发生 DIO4 事件 1h = 已发生 DIO4 事件
3	DIO3	R	0h	DIO3 事件 0h = 未发生 DIO3 事件 1h = 已发生 DIO3 事件
2	DIO2	R	0h	DIO2 事件 0h = 未发生 DIO2 事件 1h = 已发生 DIO2 事件
1	DIO1	R	0h	DIO1 事件 0h = 未发生 DIO1 事件 1h = 已发生 DIO1 事件
0	DIO0	R	0h	DIO0 事件 0h = 未发生 DIO0 事件 1h = 已发生 DIO0 事件

9.3.14 ISET（偏移 = 1040h）[复位 = 00000000h]

图 9-17 展示了 ISET，表 9-17 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 9-17. ISET

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-17. ISET 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	DIO31 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO31
30	DIO30	W	0h	DIO30 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO30
29	DIO29	W	0h	DIO29 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO29
28	DIO28	W	0h	DIO28 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO28
27	DIO27	W	0h	DIO27 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO27
26	DIO26	W	0h	DIO26 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO26
25	DIO25	W	0h	DIO25 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO25
24	DIO24	W	0h	DIO24 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO24
23	DIO23	W	0h	DIO23 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO23

表 9-17. ISET 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	W	0h	DIO22 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO22
21	DIO21	W	0h	DIO21 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO21
20	DIO20	W	0h	DIO20 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO20
19	DIO19	W	0h	DIO19 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO19
18	DIO18	W	0h	DIO18 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO18
17	DIO17	W	0h	DIO17 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO17
16	DIO16	W	0h	DIO16 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO16
15	DIO15	W	0h	DIO15 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO15
14	DIO14	W	0h	DIO14 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO14
13	DIO13	W	0h	DIO13 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO13
12	DIO12	W	0h	DIO12 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO12
11	DIO11	W	0h	DIO11 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO11
10	DIO10	W	0h	DIO10 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO10
9	DIO9	W	0h	DIO9 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO9
8	DIO8	W	0h	DIO8 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO8
7	DIO7	W	0h	DIO7 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO7
6	DIO6	W	0h	DIO6 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO6

表 9-17. ISET 字段说明 (continued)

位	字段	类型	复位	说明
5	DIO5	W	0h	DIO5 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO5
4	DIO4	W	0h	DIO4 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO4
3	DIO3	W	0h	DIO3 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO3
2	DIO2	W	0h	DIO2 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO2
1	DIO1	W	0h	DIO1 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO1
0	DIO0	W	0h	DIO0 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO0

9.3.15 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 9-18 展示了 ICLR，表 9-18 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 9-18. ICLR

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-18. ICLR 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	DIO31 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO31
30	DIO30	W	0h	DIO30 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO30
29	DIO29	W	0h	DIO29 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO29
28	DIO28	W	0h	DIO28 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO28
27	DIO27	W	0h	DIO27 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO27
26	DIO26	W	0h	DIO26 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO26
25	DIO25	W	0h	DIO25 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO25
24	DIO24	W	0h	DIO24 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO24
23	DIO23	W	0h	DIO23 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO23
22	DIO22	W	0h	DIO22 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO22

表 9-18. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	W	0h	DIO21 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO21
20	DIO20	W	0h	DIO20 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO20
19	DIO19	W	0h	DIO19 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO19
18	DIO18	W	0h	DIO18 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO18
17	DIO17	W	0h	DIO17 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO17
16	DIO16	W	0h	DIO16 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO16
15	DIO15	W	0h	DIO15 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO15
14	DIO14	W	0h	DIO14 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO14
13	DIO13	W	0h	DIO13 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO13
12	DIO12	W	0h	DIO12 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO12
11	DIO11	W	0h	DIO11 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO11
10	DIO10	W	0h	DIO10 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO10
9	DIO9	W	0h	DIO9 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO9
8	DIO8	W	0h	DIO8 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO8
7	DIO7	W	0h	DIO7 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO7
6	DIO6	W	0h	DIO6 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO6
5	DIO5	W	0h	DIO5 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO5

表 9-18. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	W	0h	DIO4 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO4
3	DIO3	W	0h	DIO3 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO3
2	DIO2	W	0h	DIO2 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO2
1	DIO1	W	0h	DIO1 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO1
0	DIO0	W	0h	DIO0 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO0

9.3.16 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 9-19 展示了 IIDX，表 9-19 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 9-19. IIDX

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

表 9-19. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 没有位置位意味着没有挂起的中断请求 1h = DIO0 中断 2h = DIO1 中断 3h = DIO2 中断 4h = DIO3 中断 5h = DIO4 中断 6h = DIO5 中断 7h = DIO6 中断 8h = DIO7 中断 9h =DIO8 中断 Ah = DIO9 中断 BH = DIO10 中断 Ch = DIO11 中断 DH = DIO12 中断 EH = DIO13 中断 FH = DIO14 中断 10h = DIO15 中断

9.3.17 IMASK（偏移 = 1058h）[复位= 00000000h]

图 9-20 展示了 IMASK，表 9-20 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 9-20. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-20. IMASK 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	DIO15	R/W	0h	DIO15 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
14	DIO14	R/W	0h	DIO14 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
13	DIO13	R/W	0h	DIO13 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
12	DIO12	R/W	0h	DIO12 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
11	DIO11	R/W	0h	DIO11 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
10	DIO10	R/W	0h	DIO10 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
9	DIO9	R/W	0h	DIO9 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
8	DIO8	R/W	0h	DIO8 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
7	DIO7	R/W	0h	DIO7 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

表 9-20. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
6	DIO6	R/W	0h	DIO6 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
5	DIO5	R/W	0h	DIO5 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
4	DIO4	R/W	0h	DIO4 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
3	DIO3	R/W	0h	DIO3 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
2	DIO2	R/W	0h	DIO2 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
1	DIO1	R/W	0h	DIO1 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
0	DIO0	R/W	0h	DIO0 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

9.3.18 RIS（偏移 = 1060h）[复位 = 00000000h]

图 9-21 展示了 RIS，表 9-21 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 9-21. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 9-21. RIS 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15	DIO15	R	0h	DIO15 事件 0h = 未发生 DIO15 事件 1h = 已发生 DIO15 事件
14	DIO14	R	0h	DIO14 事件 0h = 未发生 DIO14 事件 1h = 已发生 DIO14 事件
13	DIO13	R	0h	DIO13 事件 0h = 未发生 DIO13 事件 1h = 已发生 DIO13 事件
12	DIO12	R	0h	DIO12 事件 0h = 未发生 DIO12 事件 1h = 已发生 DIO12 事件
11	DIO11	R	0h	DIO11 事件 0h = 未发生 DIO11 事件 1h = 已发生 DIO11 事件
10	DIO10	R	0h	DIO10 事件 0h = 未发生 DIO10 事件 1h = 已发生 DIO10 事件
9	DIO9	R	0h	DIO9 事件 0h = 未发生 DIO9 事件 1h = 已发生 DIO9 事件
8	DIO8	R	0h	DIO8 事件 0h = 未发生 DIO8 事件 1h = 已发生 DIO8 事件
7	DIO7	R	0h	DIO7 事件 0h = 未发生 DIO7 事件 1h = 已发生 DIO7 事件

表 9-21. RIS 字段说明 (continued)

位	字段	类型	复位	说明
6	DIO6	R	0h	DIO6 事件 0h = 未发生 DIO6 事件 1h = 已发生 DIO6 事件
5	DIO5	R	0h	DIO5 事件 0h = 未发生 DIO5 事件 1h = 已发生 DIO5 事件
4	DIO4	R	0h	DIO4 事件 0h = 未发生 DIO4 事件 1h = 已发生 DIO4 事件
3	DIO3	R	0h	DIO3 事件 0h = 未发生 DIO3 事件 1h = 已发生 DIO3 事件
2	DIO2	R	0h	DIO2 事件 0h = 未发生 DIO2 事件 1h = 已发生 DIO2 事件
1	DIO1	R	0h	DIO1 事件 0h = 未发生 DIO1 事件 1h = 已发生 DIO1 事件
0	DIO0	R	0h	DIO0 事件 0h = 未发生 DIO0 事件 1h = 已发生 DIO0 事件

9.3.19 MIS (偏移 = 1068h) [复位 = 00000000h]

图 9-22 展示了 MIS，表 9-22 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 9-22. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 9-22. MIS 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15	DIO15	R	0h	DIO15 事件 0h = 未发生 DIO15 事件 1h = 已发生 DIO15 事件
14	DIO14	R	0h	DIO14 事件 0h = 未发生 DIO14 事件 1h = 已发生 DIO14 事件
13	DIO13	R	0h	DIO13 事件 0h = 未发生 DIO13 事件 1h = 已发生 DIO13 事件
12	DIO12	R	0h	DIO12 事件 0h = 未发生 DIO12 事件 1h = 已发生 DIO12 事件
11	DIO11	R	0h	DIO11 事件 0h = 未发生 DIO11 事件 1h = 已发生 DIO11 事件
10	DIO10	R	0h	DIO10 事件 0h = 未发生 DIO10 事件 1h = 已发生 DIO10 事件
9	DIO9	R	0h	DIO9 事件 0h = 未发生 DIO9 事件 1h = 已发生 DIO9 事件
8	DIO8	R	0h	DIO8 事件 0h = 未发生 DIO8 事件 1h = 已发生 DIO8 事件
7	DIO7	R	0h	DIO7 事件 0h = 未发生 DIO7 事件 1h = 已发生 DIO7 事件

表 9-22. MIS 字段说明 (continued)

位	字段	类型	复位	说明
6	DIO6	R	0h	DIO6 事件 0h = 未发生 DIO6 事件 1h = 已发生 DIO6 事件
5	DIO5	R	0h	DIO5 事件 0h = 未发生 DIO5 事件 1h = 已发生 DIO5 事件
4	DIO4	R	0h	DIO4 事件 0h = 未发生 DIO4 事件 1h = 已发生 DIO4 事件
3	DIO3	R	0h	DIO3 事件 0h = 未发生 DIO3 事件 1h = 已发生 DIO3 事件
2	DIO2	R	0h	DIO2 事件 0h = 未发生 DIO2 事件 1h = 已发生 DIO2 事件
1	DIO1	R	0h	DIO1 事件 0h = 未发生 DIO1 事件 1h = 已发生 DIO1 事件
0	DIO0	R	0h	DIO0 事件 0h = 未发生 DIO0 事件 1h = 已发生 DIO0 事件

9.3.20 ISET（偏移 = 1070h）[复位 = 00000000h]

图 9-23 展示了 ISET，表 9-23 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 9-23. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-23. ISET 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	W	0h	
15	DIO15	W	0h	DIO15 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO15
14	DIO14	W	0h	DIO14 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO14
13	DIO13	W	0h	DIO13 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO13
12	DIO12	W	0h	DIO12 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO12
11	DIO11	W	0h	DIO11 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO11
10	DIO10	W	0h	DIO10 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO10
9	DIO9	W	0h	DIO9 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO9
8	DIO8	W	0h	DIO8 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO8
7	DIO7	W	0h	DIO7 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO7

表 9-23. ISET 字段说明 (continued)

位	字段	类型	复位	说明
6	DIO6	W	0h	DIO6 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO6
5	DIO5	W	0h	DIO5 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO5
4	DIO4	W	0h	DIO4 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO4
3	DIO3	W	0h	DIO3 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO3
2	DIO2	W	0h	DIO2 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO2
1	DIO1	W	0h	DIO1 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO1
0	DIO0	W	0h	DIO0 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO0

9.3.21 ICLR (偏移 = 1078h) [复位 = 00000000h]

图 9-24 展示了 ICLR，表 9-24 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 9-24. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-24. ICLR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	W	0h	
15	DIO15	W	0h	DIO15 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO15
14	DIO14	W	0h	DIO14 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO14
13	DIO13	W	0h	DIO13 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO13
12	DIO12	W	0h	DIO12 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO12
11	DIO11	W	0h	DIO11 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO11
10	DIO10	W	0h	DIO10 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO10
9	DIO9	W	0h	DIO9 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO9
8	DIO8	W	0h	DIO8 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO8
7	DIO7	W	0h	DIO7 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO7

表 9-24. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
6	DIO6	W	0h	DIO6 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO6
5	DIO5	W	0h	DIO5 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO5
4	DIO4	W	0h	DIO4 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO4
3	DIO3	W	0h	DIO3 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO3
2	DIO2	W	0h	DIO2 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO2
1	DIO1	W	0h	DIO1 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO1
0	DIO0	W	0h	DIO0 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO0

9.3.22 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 9-25 展示了 IIDX，表 9-25 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 9-25. IIDX

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

表 9-25. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 没有位置位意味着没有挂起的中断请求 1h = DIO0 中断 2h = DIO1 中断 3h = DIO2 中断 4h = DIO3 中断 5h = DIO4 中断 6h = DIO5 中断 7h = DIO6 中断 8h = DIO7 中断 9h =DIO8 中断 Ah = DIO9 中断 BH = DIO10 中断 Ch = DIO11 中断 DH = DIO12 中断 EH = DIO13 中断 FH = DIO14 中断 10h = DIO15 中断

9.3.23 IMASK（偏移 = 1088h）[复位= 00000000h]

图 9-26 展示了 IMASK，表 9-26 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 9-26. IMASK

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 9-26. IMASK 字段说明

位	字段	类型	复位	说明
31	DIO31	R/W	0h	DIO31 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
30	DIO30	R/W	0h	DIO30 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
29	DIO29	R/W	0h	DIO29 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
28	DIO28	R/W	0h	DIO28 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
27	DIO27	R/W	0h	DIO27 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
26	DIO26	R/W	0h	DIO26 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
25	DIO25	R/W	0h	DIO25 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
24	DIO24	R/W	0h	DIO24 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
23	DIO23	R/W	0h	DIO23 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽

表 9-26. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	R/W	0h	DIO22 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
21	DIO21	R/W	0h	DIO21 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
20	DIO20	R/W	0h	DIO20 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
19	DIO19	R/W	0h	DIO19 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
18	DIO18	R/W	0h	DIO18 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
17	DIO17	R/W	0h	DIO17 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
16	DIO16	R/W	0h	DIO16 事件屏蔽 0h = 事件被屏蔽 1h = 事件未被屏蔽
15-0	保留	R/W	0h	

9.3.24 RIS（偏移 = 1090h）[复位 = 00000000h]

图 9-27 展示了 RIS，表 9-27 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 9-27. RIS

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

表 9-27. RIS 字段说明

位	字段	类型	复位	说明
31	DIO31	R	0h	DIO31 事件 0h = 未发生 DIO31 事件 1h = 已发生 DIO31 事件
30	DIO30	R	0h	DIO30 事件 0h = 未发生 DIO30 事件 1h = 已发生 DIO30 事件
29	DIO29	R	0h	DIO29 事件 0h = 未发生 DIO29 事件 1h = 已发生 DIO29 事件
28	DIO28	R	0h	DIO28 事件 0h = 未发生 DIO28 事件 1h = 已发生 DIO28 事件
27	DIO27	R	0h	DIO27 事件 0h = 未发生 DIO27 事件 1h = 已发生 DIO27 事件
26	DIO26	R	0h	DIO26 事件 0h = 未发生 DIO26 事件 1h = 已发生 DIO26 事件
25	DIO25	R	0h	DIO25 事件 0h = 未发生 DIO25 事件 1h = 已发生 DIO25 事件
24	DIO24	R	0h	DIO24 事件 0h = 未发生 DIO24 事件 1h = 已发生 DIO24 事件
23	DIO23	R	0h	DIO23 事件 0h = 未发生 DIO23 事件 1h = 已发生 DIO23 事件

表 9-27. RIS 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	R	0h	DIO22 事件 0h = 未发生 DIO22 事件 1h = 已发生 DIO22 事件
21	DIO21	R	0h	DIO21 事件 0h = 未发生 DIO21 事件 1h = 已发生 DIO21 事件
20	DIO20	R	0h	DIO20 事件 0h = 未发生 DIO20 事件 1h = 已发生 DIO20 事件
19	DIO19	R	0h	DIO19 事件 0h = 未发生 DIO19 事件 1h = 已发生 DIO19 事件
18	DIO18	R	0h	DIO18 事件 0h = 未发生 DIO18 事件 1h = 已发生 DIO18 事件
17	DIO17	R	0h	DIO17 事件 0h = 未发生 DIO17 事件 1h = 已发生 DIO17 事件
16	DIO16	R	0h	DIO16 事件 0h = 未发生 DIO16 事件 1h = 已发生 DIO16 事件
15-0	RESERVED	R	0h	

9.3.25 MIS (偏移 = 1098h) [复位 = 00000000h]

图 9-28 展示了 MIS，表 9-28 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 9-28. MIS

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

表 9-28. MIS 字段说明

位	字段	类型	复位	说明
31	DIO31	R	0h	DIO31 事件 0h = 未发生 DIO31 事件 1h = 已发生 DIO31 事件
30	DIO30	R	0h	DIO30 事件 0h = 未发生 DIO30 事件 1h = 已发生 DIO30 事件
29	DIO29	R	0h	DIO29 事件 0h = 未发生 DIO29 事件 1h = 已发生 DIO29 事件
28	DIO28	R	0h	DIO28 事件 0h = 未发生 DIO28 事件 1h = 已发生 DIO28 事件
27	DIO27	R	0h	DIO27 事件 0h = 未发生 DIO27 事件 1h = 已发生 DIO27 事件
26	DIO26	R	0h	DIO26 事件 0h = 未发生 DIO26 事件 1h = 已发生 DIO26 事件
25	DIO25	R	0h	DIO25 事件 0h = 未发生 DIO25 事件 1h = 已发生 DIO25 事件
24	DIO24	R	0h	DIO24 事件 0h = 未发生 DIO24 事件 1h = 已发生 DIO24 事件
23	DIO23	R	0h	DIO23 事件 0h = 未发生 DIO23 事件 1h = 已发生 DIO23 事件
22	DIO22	R	0h	DIO22 事件 0h = 未发生 DIO22 事件 1h = 已发生 DIO22 事件

表 9-28. MIS 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	R	0h	DIO21 事件 0h = 未发生 DIO21 事件 1h = 已发生 DIO21 事件
20	DIO20	R	0h	DIO20 事件 0h = 未发生 DIO20 事件 1h = 已发生 DIO20 事件
19	DIO19	R	0h	DIO19 事件 0h = 未发生 DIO19 事件 1h = 已发生 DIO19 事件
18	DIO18	R	0h	DIO18 事件 0h = 未发生 DIO18 事件 1h = 已发生 DIO18 事件
17	DIO17	R	0h	DIO17 事件 0h = 未发生 DIO17 事件 1h = 已发生 DIO17 事件
16	DIO16	R	0h	DIO16 事件 0h = 未发生 DIO16 事件 1h = 已发生 DIO16 事件
15-0	RESERVED	R	0h	

9.3.26 ISET（偏移 = 10A0h）[复位 = 00000000h]

图 9-29 展示了 ISET，表 9-29 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 9-29. ISET

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED							
W-0h							

表 9-29. ISET 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	DIO31 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO31
30	DIO30	W	0h	DIO30 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO30
29	DIO29	W	0h	DIO29 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO29
28	DIO28	W	0h	DIO28 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO28
27	DIO27	W	0h	DIO27 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO27
26	DIO26	W	0h	DIO26 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO26
25	DIO25	W	0h	DIO25 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO25
24	DIO24	W	0h	DIO24 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO24
23	DIO23	W	0h	DIO23 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO23

表 9-29. ISET 字段说明 (continued)

位	字段	类型	复位	说明
22	DIO22	W	0h	DIO22 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO22
21	DIO21	W	0h	DIO21 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO21
20	DIO20	W	0h	DIO20 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO20
19	DIO19	W	0h	DIO19 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO19
18	DIO18	W	0h	DIO18 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO18
17	DIO17	W	0h	DIO17 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO17
16	DIO16	W	0h	DIO16 事件 0h = 无效 1h = 设置 RIS 寄存器中的 DIO16
15-0	保留	W	0h	

9.3.27 ICLR (偏移 = 10A8h) [复位 = 00000000h]

图 9-30 展示了 ICLR，表 9-30 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 9-30. ICLR

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED							
W-0h							

表 9-30. ICLR 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	DIO31 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO31
30	DIO30	W	0h	DIO30 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO30
29	DIO29	W	0h	DIO29 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO29
28	DIO28	W	0h	DIO28 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO28
27	DIO27	W	0h	DIO27 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO27
26	DIO26	W	0h	DIO26 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO26
25	DIO25	W	0h	DIO25 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO25
24	DIO24	W	0h	DIO24 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO24
23	DIO23	W	0h	DIO23 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO23
22	DIO22	W	0h	DIO22 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO22

表 9-30. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	W	0h	DIO21 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO21
20	DIO20	W	0h	DIO20 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO20
19	DIO19	W	0h	DIO19 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO19
18	DIO18	W	0h	DIO18 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO18
17	DIO17	W	0h	DIO17 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO17
16	DIO16	W	0h	DIO16 事件 0h = 无效 1h = 清除 RIS 寄存器中的 DIO16
15-0	保留	W	0h	

9.3.28 EVT_MODE（偏移 = 10E0h）[复位 = 00000029h]

图 9-31 展示了 EVT_MODE，表 9-31 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 9-31. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED		EVT2_CFG		EVT1_CFG		INT0_CFG	
R/W-		R-2h		R-2h		R-1h	

表 9-31. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	EVT2_CFG	R	2h	none.GEN_EVENT1 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	EVT1_CFG	R	2h	none.GEN_EVENT0 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	none.CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

9.3.29 DESC（偏移 = 10FCh）[复位 = 16110000h]

图 9-32 展示了 DESC，表 9-32 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 9-32. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-1611h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-				R-				R-				R-			

表 9-32. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	1611h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

9.3.30 DOUT3_0（偏移 = 1200h）[复位 = 00000000h]

图 9-33 展示了 DOUT3_0，表 9-33 中对此进行了介绍。

返回到汇总表。

配置为 DIO3 至 DIO0 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 3 至 0 进行字节访问。

图 9-33. DOUT3_0

31	30	29	28	27	26	25	24
RESERVED							DIO3
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO2
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO1
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO0
W-0h							W-0h

表 9-33. DOUT3_0 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO3	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO3 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO2	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO2 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO1	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO1 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO0	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO0 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.31 DOUT7_4（偏移 = 1204h）[复位 = 00000000h]

图 9-34 展示了 DOUT7_4，表 9-34 中对此进行了介绍。

返回到汇总表。

配置为 DIO7 至 DIO4 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 7 至 4 进行字节访问。

图 9-34. DOUT7_4

31	30	29	28	27	26	25	24
RESERVED							DIO7
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO6
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO5
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO4
W-0h							W-0h

表 9-34. DOUT7_4 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO7	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO7 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO6	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO6 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO5	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO5 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO4	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO4 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.32 DOUT11_8（偏移 = 1208h）[复位 = 00000000h]

图 9-35 展示了 DOUT11_8，表 9-35 中对此进行了介绍。

返回到汇总表。

配置为 DIO11 至 DIO8 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 11 至 8 进行字节访问。

图 9-35. DOUT11_8

31	30	29	28	27	26	25	24
RESERVED							DIO11
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO10
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO9
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO8
W-0h							W-0h

表 9-35. DOUT11_8 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO11	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO11 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO10	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO10 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO9	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO9 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO8	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO8 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.33 DOUT15_12（偏移 = 120Ch）[复位 = 00000000h]

图 9-36 展示了 DOUT15_12，表 9-36 中对此进行了介绍。

返回到汇总表。

配置为 DIO15 至 DIO12 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 15 至 12 进行字节访问。

图 9-36. DOUT15_12

31	30	29	28	27	26	25	24
RESERVED							DIO15
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO14
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO13
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO12
W-0h							W-0h

表 9-36. DOUT15_12 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO15	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO15 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO14	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO14 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO13	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO13 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO12	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO12 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.34 DOUT19_16（偏移 = 1210h）[复位 = 00000000h]

图 9-37 展示了 DOUT19_16，表 9-37 中对此进行了介绍。

返回到汇总表。

配置为 DIO19 至 DIO16 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 19 至 16 进行字节访问。

图 9-37. DOUT19_16

31	30	29	28	27	26	25	24
RESERVED							DIO19
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO18
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO17
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO16
W-0h							W-0h

表 9-37. DOUT19_16 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO19	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO19 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO18	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO18 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO17	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO17 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO16	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO16 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.35 DOUT23_20（偏移 = 1214h）[复位 = 00000000h]

图 9-38 展示了 DOUT23_20，表 9-38 中对此进行了介绍。

返回到汇总表。

配置为 DIO23 至 DIO20 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 23 至 20 进行字节访问。

图 9-38. DOUT23_20

31	30	29	28	27	26	25	24
RESERVED							DIO23
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO22
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO21
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO20
W-0h							W-0h

表 9-38. DOUT23_20 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO23	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO23 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO22	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO22 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO21	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO21 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO20	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO20 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.36 DOUT27_24（偏移 = 1218h）[复位 = 00000000h]

图 9-39 展示了 DOUT27_24，表 9-39 中对此进行了介绍。

返回到汇总表。

配置为 DIO27 至 DIO24 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 27 至 24 进行字节访问。

图 9-39. DOUT27_24

31	30	29	28	27	26	25	24
RESERVED							DIO27
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO26
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO25
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO24
W-0h							W-0h

表 9-39. DOUT27_24 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO27	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO27 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO26	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO26 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO25	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO25 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO24	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO24 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.37 DOUT31_28（偏移 = 121Ch）[复位 = 00000000h]

图 9-40 展示了 DOUT31_28，表 9-40 中对此进行了介绍。

返回到汇总表。

配置为 DIO31 至 DIO28 的引脚的数据输出。这是一个别名寄存器，用于对 DOUT31_0 寄存器中的位 31 至 28 进行字节访问。

图 9-40. DOUT31_28

31	30	29	28	27	26	25	24
RESERVED							DIO31
W-0h							W-0h
23	22	21	20	19	18	17	16
RESERVED							DIO30
W-0h							W-0h
15	14	13	12	11	10	9	8
保留							DIO29
W-0h							W-0h
7	6	5	4	3	2	1	0
ESERVED							DIO28
W-0h							W-0h

表 9-40. DOUT31_28 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	W	0h	
24	DIO31	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO31 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23-17	RESERVED	W	0h	
16	DIO30	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO30 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15-9	保留	W	0h	
8	DIO29	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO29 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7-1	RESERVED	W	0h	
0	DIO28	W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO28 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.38 DOUT31_0（偏移 = 1280h）[复位 = 00000000h]

图 9-41 展示了 DOUT31_0，表 9-41 中对此进行了介绍。

返回到汇总表。

配置为 DIO31 至 DIO0 的引脚的数据输出。

图 9-41. DOUT31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-41. DOUT31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO31 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
30	DIO30	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO30 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
29	DIO29	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO29 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
28	DIO28	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO28 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
27	DIO27	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO27 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
26	DIO26	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO26 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
25	DIO25	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO25 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

表 9-41. DOUT31_0 字段说明 (continued)

位	字段	类型	复位	说明
24	DIO24	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO24 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
23	DIO23	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO23 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
22	DIO22	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO22 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
21	DIO21	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO21 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
20	DIO20	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO20 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
19	DIO19	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO19 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
18	DIO18	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO18 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
17	DIO17	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO17 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
16	DIO16	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO16 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
15	DIO15	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO15 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
14	DIO14	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO14 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
13	DIO13	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO13 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
12	DIO12	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO12 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
11	DIO11	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO11 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

表 9-41. DOUT31_0 字段说明 (continued)

位	字段	类型	复位	说明
10	DIO10	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO10 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
9	DIO9	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO9 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
8	DIO8	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO8 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
7	DIO7	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO7 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
6	DIO6	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO6 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
5	DIO5	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO5 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
4	DIO4	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO4 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
3	DIO3	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO3 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
2	DIO2	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO2 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
1	DIO1	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO1 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1
0	DIO0	R/W	0h	当通过 DOE31_0 寄存器启用输出时，该位设置配置为 DIO0 的引脚的值。 0h = 输出设定为 0 1h = 输出设定为 1

9.3.39 DOUTSET31_0（偏移 = 1290h）[复位 = 00000000h]

图 9-42 展示了 DOUTSET31_0，表 9-42 中对此进行了介绍。

返回到汇总表。

向该寄存器中的位位置写入 1 会设置 DOUT31_0 寄存器中的相应位。

图 9-42. DOUTSET31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-42. DOUTSET31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO31 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO31
30	DIO30	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO30 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO30
29	DIO29	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO29 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO29
28	DIO28	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO28 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO28
27	DIO27	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO27 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO27
26	DIO26	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO26 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO26
25	DIO25	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO25 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO25

表 9-42. DOUTSET31_0 字段说明 (continued)

位	字段	类型	复位	说明
24	DIO24	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO24 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO24
23	DIO23	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO23 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO23
22	DIO22	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO22 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO22
21	DIO21	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO21 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO21
20	DIO20	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO20 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO20
19	DIO19	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO19 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO19
18	DIO18	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO18 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO18
17	DIO17	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO17 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO17
16	DIO16	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO16 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO16
15	DIO15	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO15 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO15
14	DIO14	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO14 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO14
13	DIO13	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO13 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO13
12	DIO12	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO12 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO12
11	DIO11	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO11 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO11

表 9-42. DOUTSET31_0 字段说明 (continued)

位	字段	类型	复位	说明
10	DIO10	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO10 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO10
9	DIO9	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO9 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO9
8	DIO8	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO8 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO8
7	DIO7	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO7 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO7
6	DIO6	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO6 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO6
5	DIO5	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO5 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO5
4	DIO4	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO4 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO4
3	DIO3	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO3 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO3
2	DIO2	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO2 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO2
1	DIO1	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO1 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO1
0	DIO0	W	0h	向该位写入 1 会设置 DOUT31_0 寄存器中的 DIO0 位。写入 0 无效。 0h = 无效 1h = 设置 DOUT31_0 中的 DIO0

9.3.40 DOUTCLR31_0（偏移 = 12A0h）[复位 = 00000000h]

图 9-43 展示了 DOUTCLR31_0，表 9-43 中对此进行了介绍。

返回到汇总表。

向该寄存器中的位位置写入 1 会将 DOUT31_0 寄存器中的相应位清零。

图 9-43. DOUTCLR31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-43. DOUTCLR31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	向该位写入 1 会将 DOUT31_0 寄存器中的 DIO31 位清零。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO31
30	DIO30	W	0h	向该位写入 1 会将 DOUT31_0 寄存器中的 DIO30 位清零。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO30
29	DIO29	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO29 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO29
28	DIO28	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO28 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO28
27	DIO27	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO27 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO27
26	DIO26	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO26 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO26
25	DIO25	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO25 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO25

表 9-43. DOUTCLR31_0 字段说明 (continued)

位	字段	类型	复位	说明
24	DIO24	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO24 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO24
23	DIO23	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO23 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO23
22	DIO22	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO22 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO22
21	DIO21	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO21 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO21
20	DIO20	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO20 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO20
19	DIO19	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO19 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO19
18	DIO18	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO18 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO18
17	DIO17	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO17 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO17
16	DIO16	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO16 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO16
15	DIO15	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO15 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO15
14	DIO14	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO14 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO14
13	DIO13	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO13 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO13
12	DIO12	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO12 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO12
11	DIO11	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO11 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO11

表 9-43. DOUTCLR31_0 字段说明 (continued)

位	字段	类型	复位	说明
10	DIO10	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO10 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO10
9	DIO9	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO9 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO9
8	DIO8	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO8 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO8
7	DIO7	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO7 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO7
6	DIO6	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO6 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO6
5	DIO5	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO5 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO5
4	DIO4	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO4 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO4
3	DIO3	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO3 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO3
2	DIO2	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO2 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO2
1	DIO1	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO1 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO1
0	DIO0	W	0h	向该位写入 1 会清除 DOUT31_0 寄存器中的 DIO0 位。写入 0 无效。 0h = 无效 1h = 清除 DOUT31_0 中的 DIO0

9.3.41 DOUTTGL31_0（偏移 = 12B0h）[复位 = 00000000h]

图 9-44 展示了 DOUTTGL31_0，表 9-44 中对此进行了介绍。

返回到汇总表。

向该寄存器中的位位置写入 1 将反转相应的 DIO 输出。

图 9-44. DOUTTGL31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-44. DOUTTGL31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	该位用于切换 DIO31 输出。 0h = 无效 1h = 切换输出
30	DIO30	W	0h	该位用于切换 DIO30 输出。 0h = 无效 1h = 切换输出
29	DIO29	W	0h	该位用于切换 DIO29 输出。 0h = 无效 1h = 切换输出
28	DIO28	W	0h	该位用于切换 DIO28 输出。 0h = 无效 1h = 切换输出
27	DIO27	W	0h	该位用于切换 DIO27 输出。 0h = 无效 1h = 切换输出
26	DIO26	W	0h	该位用于切换 DIO26 输出。 0h = 无效 1h = 切换输出
25	DIO25	W	0h	该位用于切换 DIO25 输出。 0h = 无效 1h = 切换输出
24	DIO24	W	0h	该位用于切换 DIO24 输出。 0h = 无效 1h = 切换输出
23	DIO23	W	0h	该位用于切换 DIO23 输出。 0h = 无效 1h = 切换输出
22	DIO22	W	0h	该位用于切换 DIO22 输出。 0h = 无效 1h = 切换输出

表 9-44. DOUTTGL31_0 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	W	0h	该位用于切换 DIO21 输出。 0h = 无效 1h = 切换输出
20	DIO20	W	0h	该位用于切换 DIO20 输出。 0h = 无效 1h = 切换输出
19	DIO19	W	0h	该位用于切换 DIO19 输出。 0h = 无效 1h = 切换输出
18	DIO18	W	0h	该位用于切换 DIO18 输出。 0h = 无效 1h = 切换输出
17	DIO17	W	0h	该位用于切换 DIO17 输出。 0h = 无效 1h = 切换输出
16	DIO16	W	0h	该位用于切换 DIO16 输出。 0h = 无效 1h = 切换输出
15	DIO15	W	0h	该位用于切换 DIO15 输出。 0h = 无效 1h = 切换输出
14	DIO14	W	0h	该位用于切换 DIO14 输出。 0h = 无效 1h = 切换输出
13	DIO13	W	0h	该位用于切换 DIO13 输出。 0h = 无效 1h = 切换输出
12	DIO12	W	0h	该位用于切换 DIO12 输出。 0h = 无效 1h = 切换输出
11	DIO11	W	0h	该位用于切换 DIO11 输出。 0h = 无效 1h = 切换输出
10	DIO10	W	0h	该位用于切换 DIO10 输出。 0h = 无效 1h = 切换输出
9	DIO9	W	0h	该位用于切换 DIO9 输出。 0h = 无效 1h = 切换输出
8	DIO8	W	0h	该位用于切换 DIO8 输出。 0h = 无效 1h = 切换输出
7	DIO7	W	0h	该位用于切换 DIO7 输出。 0h = 无效 1h = 切换输出
6	DIO6	W	0h	该位用于切换 DIO6 输出。 0h = 无效 1h = 切换输出
5	DIO5	W	0h	该位用于切换 DIO5 输出。 0h = 无效 1h = 切换输出

表 9-44. DOUTTGL31_0 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	W	0h	该位用于切换 DIO4 输出。 0h = 无效 1h = 切换输出
3	DIO3	W	0h	该位用于切换 DIO3 输出。 0h = 无效 1h = 切换输出
2	DIO2	W	0h	该位用于切换 DIO2 输出。 0h = 无效 1h = 切换输出
1	DIO1	W	0h	该位用于切换 DIO1 输出。 0h = 无效 1h = 切换输出
0	DIO0	W	0h	该位用于切换 DIO0 输出。 0h = 无效 1h = 切换输出

9.3.42 DOE31_0（偏移 = 12C0h）[复位 = 00000000h]

图 9-45 展示了 DOE31_0，表 9-45 中对此进行了介绍。

返回到汇总表。

该寄存器用于启用 DIO31 至 DIO0 的数据输出。

图 9-45. DOE31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-45. DOE31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	R/W	0h	启用 DIO31 的数据输出。 0h = 输出禁用 1h = 输出启用
30	DIO30	R/W	0h	启用 DIO30 的数据输出。 0h = 输出禁用 1h = 输出启用
29	DIO29	R/W	0h	启用 DIO29 的数据输出。 0h = 输出禁用 1h = 输出启用
28	DIO28	R/W	0h	启用 DIO28 的数据输出。 0h = 输出禁用 1h = 输出启用
27	DIO27	R/W	0h	启用 DIO27 的数据输出。 0h = 输出禁用 1h = 输出启用
26	DIO26	R/W	0h	启用 DIO26 的数据输出。 0h = 输出禁用 1h = 输出启用
25	DIO25	R/W	0h	启用 DIO25 的数据输出。 0h = 输出禁用 1h = 输出启用
24	DIO24	R/W	0h	启用 DIO24 的数据输出。 0h = 输出禁用 1h = 输出启用
23	DIO23	R/W	0h	启用 DIO23 的数据输出。 0h = 输出禁用 1h = 输出启用
22	DIO22	R/W	0h	启用 DIO22 的数据输出。 0h = 输出禁用 1h = 输出启用

表 9-45. DOE31_0 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	R/W	0h	启用 DIO21 的数据输出。 0h = 输出禁用 1h = 输出启用
20	DIO20	R/W	0h	启用 DIO20 的数据输出。 0h = 输出禁用 1h = 输出启用
19	DIO19	R/W	0h	启用 DIO19 的数据输出。 0h = 输出禁用 1h = 输出启用
18	DIO18	R/W	0h	启用 DIO18 的数据输出。 0h = 输出禁用 1h = 输出启用
17	DIO17	R/W	0h	启用 DIO17 的数据输出。 0h = 输出禁用 1h = 输出启用
16	DIO16	R/W	0h	启用 DIO16 的数据输出。 0h = 输出禁用 1h = 输出启用
15	DIO15	R/W	0h	启用 DIO15 的数据输出。 0h = 输出禁用 1h = 输出启用
14	DIO14	R/W	0h	启用 DIO14 的数据输出。 0h = 输出禁用 1h = 输出启用
13	DIO13	R/W	0h	启用 DIO13 的数据输出。 0h = 输出禁用 1h = 输出启用
12	DIO12	R/W	0h	启用 DIO12 的数据输出。 0h = 输出禁用 1h = 输出启用
11	DIO11	R/W	0h	启用 DIO11 的数据输出。 0h = 输出禁用 1h = 输出启用
10	DIO10	R/W	0h	启用 DIO10 的数据输出。 0h = 输出禁用 1h = 输出启用
9	DIO9	R/W	0h	启用 DIO9 的数据输出。 0h = 输出禁用 1h = 输出启用
8	DIO8	R/W	0h	启用 DIO8 的数据输出。 0h = 输出禁用 1h = 输出启用
7	DIO7	R/W	0h	启用 DIO7 的数据输出。 0h = 输出禁用 1h = 输出启用
6	DIO6	R/W	0h	启用 DIO6 的数据输出。 0h = 输出禁用 1h = 输出启用
5	DIO5	R/W	0h	启用 DIO5 的数据输出。 0h = 输出禁用 1h = 输出启用

表 9-45. DOE31_0 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	R/W	0h	启用 DIO4 的数据输出。 0h = 输出禁用 1h = 输出启用
3	DIO3	R/W	0h	启用 DIO3 的数据输出。 0h = 输出禁用 1h = 输出启用
2	DIO2	R/W	0h	启用 DIO2 的数据输出。 0h = 输出禁用 1h = 输出启用
1	DIO1	R/W	0h	启用 DIO1 的数据输出。 0h = 输出禁用 1h = 输出启用
0	DIO0	R/W	0h	启用 DIO0 的数据输出。 0h = 输出禁用 1h = 输出启用

9.3.43 DOESET31_0（偏移 = 12D0h）[复位 = 00000000h]

图 9-46 展示了 DOESET31_0，表 9-46 中对此进行了介绍。

返回到汇总表。

向该寄存器中的位位置写入 1 会设置 DOE31_0 寄存器中的相应位。

图 9-46. DOESET31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-46. DOESET31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO31 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO31
30	DIO30	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO30 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO30
29	DIO29	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO29 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO29
28	DIO28	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO28 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO28
27	DIO27	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO27 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO27
26	DIO26	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO26 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO26
25	DIO25	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO25 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO25
24	DIO24	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO24 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO24
23	DIO23	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO23 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO23
22	DIO22	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO22 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO22

表 9-46. DOESET31_0 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO21 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO21
20	DIO20	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO20 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO20
19	DIO19	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO19 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO19
18	DIO18	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO18 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO18
17	DIO17	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO17 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO17
16	DIO16	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO16 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO16
15	DIO15	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO15 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO15
14	DIO14	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO14 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO14
13	DIO13	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO13 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO13
12	DIO12	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO12 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO12
11	DIO11	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO11 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO11
10	DIO10	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO10 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO10
9	DIO9	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO9 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO9
8	DIO8	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO8 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO8
7	DIO7	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO7 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO7
6	DIO6	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO6 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO6
5	DIO5	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO5 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO5

表 9-46. DOESET31_0 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO4 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO4
3	DIO3	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO3 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO3
2	DIO2	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO2 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO2
1	DIO1	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO1 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO1
0	DIO0	W	0h	向该位写入 1 会设置 DOE31_0 寄存器中的 DIO0 位。写入 0 无效。 0h = 无效 1h = 设置 DOE31_0 中的 DIO0

9.3.44 DOECLR31_0（偏移 = 12E0h）[复位 = 00000000h]

图 9-47 展示了 DOECLR31_0，表 9-47 中对此进行了介绍。

返回到汇总表。

向该寄存器中的位位置写入 1 会清除 DOE31_0 寄存器中的相应位。

图 9-47. DOECLR31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 9-47. DOECLR31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO31 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO31
30	DIO30	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO30 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO30
29	DIO29	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO29 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO29
28	DIO28	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO28 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO28
27	DIO27	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO27 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO27
26	DIO26	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO26 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO26
25	DIO25	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO25 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO25
24	DIO24	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO24 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO24
23	DIO23	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO23 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO23
22	DIO22	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO22 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO22

表 9-47. DOECLR31_0 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO21 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO21
20	DIO20	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO20 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO20
19	DIO19	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO19 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO19
18	DIO18	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO18 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO18
17	DIO17	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO17 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO17
16	DIO16	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO16 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO16
15	DIO15	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO15 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO15
14	DIO14	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO14 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO14
13	DIO13	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO13 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO13
12	DIO12	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO12 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO12
11	DIO11	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO11 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO11
10	DIO10	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO10 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO10
9	DIO9	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO9 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO9
8	DIO8	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO8 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO8
7	DIO7	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO7 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO7
6	DIO6	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO6 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO6
5	DIO5	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO5 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO5

表 9-47. DOECLR31_0 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO4 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO4
3	DIO3	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO3 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO3
2	DIO2	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO2 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO2
1	DIO1	W	0h	向该位写入 1 会清除 DOE31_0 寄存器中的 DIO1 位。写入 0 无效。 0h = 无效 1h = 清除 DOE31_0 中的 DIO1
0	DIO0	W	0h	向该位写入 1 将 DOE31_0 寄存器中的 DIO0 位清零。写入 0 无效。 0h = 无效 1h = 将 DOE31_0 中的 DIO0 清零

9.3.45 DIN3_0（偏移 = 1300h）[复位 = 00000000h]

图 9-48 展示了 DIN3_0，表 9-48 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO3 至 DIO0 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 3 至 0 进行字节访问。

图 9-48. DIN3_0

31	30	29	28	27	26	25	24
RESERVED							DIO3
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO2
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO1
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO0
R-0h							R-0h

表 9-48. DIN3_0 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO3	R	0h	该位读取 DIO3 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO2	R	0h	该位读取 DIO2 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO1	R	0h	该位读取 DIO1 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO0	R	0h	该位读取 DIO0 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.46 DIN7_4 (偏移 = 1304h) [复位 = 00000000h]

图 9-49 展示了 DIN7_4，表 9-49 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO7 至 DIO4 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 7 至 4 进行字节访问。

图 9-49. DIN7_4

31	30	29	28	27	26	25	24
RESERVED							DIO7
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO6
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO5
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO4
R-0h							R-0h

表 9-49. DIN7_4 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO7	R	0h	该位读取 DIO7 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO6	R	0h	该位读取 DIO6 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO5	R	0h	该位读取 DIO5 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO4	R	0h	该位读取 DIO4 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.47 DIN11_8（偏移 = 1308h）[复位 = 00000000h]

图 9-50 展示了 DIN11_8，表 9-50 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO11 至 DIO8 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 11 至 8 进行字节访问。

图 9-50. DIN11_8

31	30	29	28	27	26	25	24
RESERVED							DIO11
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO10
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO9
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO8
R-0h							R-0h

表 9-50. DIN11_8 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO11	R	0h	该位读取 DIO11 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO10	R	0h	该位读取 DIO10 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO9	R	0h	该位读取 DIO9 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO8	R	0h	该位读取 DIO8 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.48 DIN15_12（偏移 = 130Ch）[复位 = 00000000h]

图 9-51 展示了 DIN15_12，表 9-51 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO15 至 DIO12 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 15 至 12 进行字节访问。

图 9-51. DIN15_12

31	30	29	28	27	26	25	24
RESERVED							DIO15
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO14
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO13
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO12
R-0h							R-0h

表 9-51. DIN15_12 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO15	R	0h	该位读取 DIO15 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO14	R	0h	该位读取 DIO14 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO13	R	0h	该位读取 DIO13 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO12	R	0h	该位读取 DIO12 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.49 DIN19_16（偏移 = 1310h）[复位 = 00000000h]

图 9-52 展示了 DIN19_16，表 9-52 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO19 至 DIO16 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 19 至 16 进行字节访问。

图 9-52. DIN19_16

31	30	29	28	27	26	25	24
RESERVED							DIO19
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO18
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO17
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO16
R-0h							R-0h

表 9-52. DIN19_16 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO19	R	0h	该位读取 DIO19 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO18	R	0h	该位读取 DIO18 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO17	R	0h	该位读取 DIO17 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO16	R	0h	该位读取 DIO16 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.50 DIN23_20（偏移 = 1314h）[复位 = 00000000h]

图 9-53 展示了 DIN23_20，表 9-53 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO23 至 DIO20 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 23 至 20 进行字节访问。

图 9-53. DIN23_20

31	30	29	28	27	26	25	24
RESERVED							DIO23
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO22
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO21
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO20
R-0h							R-0h

表 9-53. DIN23_20 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO23	R	0h	该位读取 DIO23 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO22	R	0h	该位读取 DIO22 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO21	R	0h	该位读取 DIO21 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO20	R	0h	该位读取 DIO20 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.51 DIN27_24（偏移 = 1318h）[复位 = 00000000h]

图 9-54 展示了 DIN27_24，表 9-54 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO27 至 DIO24 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 27 至 24 进行字节访问。

图 9-54. DIN27_24

31	30	29	28	27	26	25	24
RESERVED							DIO27
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO26
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO25
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO24
R-0h							R-0h

表 9-54. DIN27_24 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO27	R	0h	该位读取 DIO27 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO26	R	0h	该位读取 DIO26 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO25	R	0h	该位读取 DIO25 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO24	R	0h	该位读取 DIO24 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.52 DIN31_28（偏移 = 131Ch）[复位 = 00000000h]

图 9-55 展示了 DIN31_28，表 9-55 中对此进行了介绍。

返回到汇总表。

来自配置为 DIO31 至 DIO28 的引脚的数据输入。这是一个别名寄存器，用于对 DIN31_0 寄存器中的位 31 至 28 进行字节访问。

图 9-55. DIN31_28

31	30	29	28	27	26	25	24
RESERVED							DIO31
R-0h							R-0h
23	22	21	20	19	18	17	16
RESERVED							DIO30
R-0h							R-0h
15	14	13	12	11	10	9	8
保留							DIO29
R-0h							R-0h
7	6	5	4	3	2	1	0
ESERVED							DIO28
R-0h							R-0h

表 9-55. DIN31_28 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R	0h	
24	DIO31	R	0h	该位读取 DIO31 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23-17	RESERVED	R	0h	
16	DIO30	R	0h	该位读取 DIO30 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15-9	保留	R	0h	
8	DIO29	R	0h	该位读取 DIO29 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7-1	RESERVED	R	0h	
0	DIO28	R	0h	该位读取 DIO28 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.53 DIN31_0（偏移 = 1380h）[复位 = 00000000h]

图 9-56 展示了 DIN31_0，表 9-56 中对此进行了介绍。

返回到汇总表。

配置为 DIO31 至 DIO0 的引脚的数据输入值。

图 9-56. DIN31_0

31	30	29	28	27	26	25	24
DIO31	DIO30	DIO29	DIO28	DIO27	DIO26	DIO25	DIO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
DIO23	DIO22	DIO21	DIO20	DIO19	DIO18	DIO17	DIO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
DIO15	DIO14	DIO13	DIO12	DIO11	DIO10	DIO9	DIO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 9-56. DIN31_0 字段说明

位	字段	类型	复位	说明
31	DIO31	R	0h	该位读取 DIO31 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
30	DIO30	R	0h	该位读取 DIO30 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
29	DIO29	R	0h	该位读取 DIO29 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
28	DIO28	R	0h	该位读取 DIO28 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
27	DIO27	R	0h	该位读取 DIO27 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
26	DIO26	R	0h	该位读取 DIO26 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
25	DIO25	R	0h	该位读取 DIO25 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
24	DIO24	R	0h	该位读取 DIO24 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
23	DIO23	R	0h	该位读取 DIO23 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
22	DIO22	R	0h	该位读取 DIO22 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

表 9-56. DIN31_0 字段说明 (continued)

位	字段	类型	复位	说明
21	DIO21	R	0h	该位读取 DIO21 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
20	DIO20	R	0h	该位读取 DIO20 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
19	DIO19	R	0h	该位读取 DIO19 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
18	DIO18	R	0h	该位读取 DIO18 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
17	DIO17	R	0h	该位读取 DIO17 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
16	DIO16	R	0h	该位读取 DIO16 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
15	DIO15	R	0h	该位读取 DIO15 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
14	DIO14	R	0h	该位读取 DIO14 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
13	DIO13	R	0h	该位读取 DIO13 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
12	DIO12	R	0h	该位读取 DIO12 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
11	DIO11	R	0h	该位读取 DIO11 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
10	DIO10	R	0h	该位读取 DIO10 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
9	DIO9	R	0h	该位读取 DIO9 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
8	DIO8	R	0h	该位读取 DIO8 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
7	DIO7	R	0h	该位读取 DIO7 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
6	DIO6	R	0h	该位读取 DIO6 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
5	DIO5	R	0h	该位读取 DIO5 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

表 9-56. DIN31_0 字段说明 (continued)

位	字段	类型	复位	说明
4	DIO4	R	0h	该位读取 DIO4 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
3	DIO3	R	0h	该位读取 DIO3 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
2	DIO2	R	0h	该位读取 DIO2 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
1	DIO1	R	0h	该位读取 DIO1 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1
0	DIO0	R	0h	该位读取 DIO0 的数据输入值。 0h = 输入值为 0 1h = 输入值为 1

9.3.54 POLARITY15_0（偏移 = 1390h）[复位 = 00000000h]

图 9-57 展示了 POLARITY15_0，表 9-57 中对此进行了介绍。

返回到汇总表。

该寄存器用于启用和配置 DIO15 至 DIO0 上输入边沿检测的极性。当输入事件与配置的极性匹配时，RIS 寄存器中相应的 DIO 位将被置位。

图 9-57. POLARITY15_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIO15		DIO14		DIO13		DIO12		DIO11		DIO10		DIO9		DIO8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO7		DIO6		DIO5		DIO4		DIO3		DIO2		DIO1		DIO0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

表 9-57. POLARITY15_0 字段说明

位	字段	类型	复位	说明
31-30	DIO15	R/W	0h	启用和配置 DIO15 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
29-28	DIO14	R/W	0h	启用和配置 DIO14 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
27-26	DIO13	R/W	0h	启用和配置 DIO13 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
25-24	DIO12	R/W	0h	启用和配置 DIO12 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
23-22	DIO11	R/W	0h	启用和配置 DIO11 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
21-20	DIO10	R/W	0h	启用和配置 DIO10 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
19-18	DIO9	R/W	0h	启用和配置 DIO9 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿

表 9-57. POLARITY15_0 字段说明 (continued)

位	字段	类型	复位	说明
17-16	DIO8	R/W	0h	启用和配置 DIO8 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
15-14	DIO7	R/W	0h	启用和配置 DIO7 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
13-12	DIO6	R/W	0h	启用和配置 DIO6 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
11-10	DIO5	R/W	0h	启用和配置 DIO5 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
9-8	DIO4	R/W	0h	启用和配置 DIO4 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
7-6	DIO3	R/W	0h	启用和配置 DIO3 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
5-4	DIO2	R/W	0h	启用和配置 DIO2 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
3-2	DIO1	R/W	0h	启用和配置 DIO1 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
1-0	DIO0	R/W	0h	启用和配置 DIO0 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿

9.3.55 POLARITY31_16（偏移 = 13A0h）[复位 = 00000000h]

图 9-58 展示了 POLARITY31_16，表 9-58 中对此进行了介绍。

返回到汇总表。

该寄存器用于启用和配置 DIO31 至 DIO16 上输入边沿检测的极性。当输入事件与配置的极性匹配时，RIS 寄存器中相应的 DIO 位将被置位。

图 9-58. POLARITY31_16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIO31		DIO30		DIO29		DIO28		DIO27		DIO26		DIO25		DIO24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO23		DIO22		DIO21		DIO20		DIO19		DIO18		DIO17		DIO16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

表 9-58. POLARITY31_16 字段说明

位	字段	类型	复位	说明
31-30	DIO31	R/W	0h	启用和配置 DIO31 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
29-28	DIO30	R/W	0h	启用和配置 DIO30 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
27-26	DIO29	R/W	0h	启用和配置 DIO29 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
25-24	DIO28	R/W	0h	启用和配置 DIO28 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
23-22	DIO27	R/W	0h	启用和配置 DIO27 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
21-20	DIO26	R/W	0h	启用和配置 DIO26 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
19-18	DIO25	R/W	0h	启用和配置 DIO25 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿

表 9-58. POLARITY31_16 字段说明 (continued)

位	字段	类型	复位	说明
17-16	DIO24	R/W	0h	启用和配置 DIO24 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
15-14	DIO23	R/W	0h	启用和配置 DIO23 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
13-12	DIO22	R/W	0h	启用和配置 DIO22 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
11-10	DIO21	R/W	0h	启用和配置 DIO21 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
9-8	DIO20	R/W	0h	启用和配置 DIO20 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
7-6	DIO19	R/W	0h	启用和配置 DIO19 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
5-4	DIO18	R/W	0h	启用和配置 DIO18 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
3-2	DIO17	R/W	0h	启用和配置 DIO17 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿
1-0	DIO16	R/W	0h	启用和配置 DIO16 的边沿检测极性。 0h = 边沿检测禁用 1h = 检测输入事件的上升沿 2h = 检测输入事件的下降沿 3h = 检测输入事件的上升沿和下降沿

9.3.56 CTL（偏移 = 1400h）[复位= 00000000h]

图 9-59 展示了 CTL，表 9-59 中对此进行了介绍。

返回到汇总表。

GPIO 控制寄存器

图 9-59. CTL

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

表 9-59. CTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	FASTWAKEONLY	R/W	0h	FASTWAKEONLY 用于全局控制快速唤醒 0h = 快速唤醒的全局控制未启用，每位快速唤醒功能取决于 FASTWAKE.DIN 1h = 快速唤醒的全局控制启用

9.3.57 FASTWAKE（偏移 = 1404h）[复位 = 00000000h]

图 9-60 展示了 FASTWAKE，表 9-60 中对此进行了介绍。

返回到汇总表。

这是针对位切片的每位快速唤醒使能，允许 GPIO 模块保持低功耗状态，并且无需输入同步器或滤波器的高速时钟

图 9-60. FASTWAKE

31	30	29	28	27	26	25	24
DIN31	DIN30	DIN29	DIN28	DIN27	DIN26	DIN25	DIN24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DIN23	DIN22	DIN21	DIN20	DIN19	DIN18	DIN17	DIN16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DIN15	DIN14	DIN13	DIN12	DIN11	DIN10	DIN9	DIN8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DIN7	DIN6	DIN5	DIN4	DIN3	DIN2	DIN1	DIN0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-60. FASTWAKE 字段说明

位	字段	类型	复位	说明
31	DIN31	R/W	0h	启用 DIN31 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
30	DIN30	R/W	0h	启用 DIN30 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
29	DIN29	R/W	0h	启用 DIN29 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
28	DIN28	R/W	0h	启用 DIN29 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
27	DIN27	R/W	0h	启用 DIN27 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
26	DIN26	R/W	0h	启用 DIN26 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
25	DIN25	R/W	0h	启用 DIN25 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
24	DIN24	R/W	0h	启用 DIN24 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
23	DIN23	R/W	0h	启用 DIN23 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用

表 9-60. FASTWAKE 字段说明 (continued)

位	字段	类型	复位	说明
22	DIN22	R/W	0h	启用 DIN22 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
21	DIN21	R/W	0h	启用 DIN21 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
20	DIN20	R/W	0h	启用 DIN20 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
19	DIN19	R/W	0h	启用 DIN19 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
18	DIN18	R/W	0h	启用 DIN18 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
17	DIN17	R/W	0h	启用 DIN17 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
16	DIN16	R/W	0h	启用 DIN16 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
15	DIN15	R/W	0h	启用 DIN15 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
14	DIN14	R/W	0h	启用 DIN14 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
13	DIN13	R/W	0h	启用 DIN13 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
12	DIN12	R/W	0h	启用 DIN12 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
11	DIN11	R/W	0h	启用 DIN11 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
10	DIN10	R/W	0h	启用 DIN10 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
9	DIN9	R/W	0h	启用 DIN9 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
8	DIN8	R/W	0h	启用 DIN8 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
7	DIN7	R/W	0h	启用 DIN7 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
6	DIN6	R/W	0h	启用 DIN6 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用

表 9-60. FASTWAKE 字段说明 (continued)

位	字段	类型	复位	说明
5	DIN5	R/W	0h	启用 DIN5 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
4	DIN4	R/W	0h	启用 DIN4 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
3	DIN3	R/W	0h	启用 DIN3 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
2	DIN2	R/W	0h	启用 DIN2 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
1	DIN1	R/W	0h	启用 DIN1 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用
0	DIN0	R/W	0h	启用 DIN0 的快速唤醒功能 0h = 快速唤醒功能禁用 1h = 快速唤醒功能启用

9.3.58 SUB0CFG（偏移 = 1500h）[复位 = 00000000h]

图 9-61 展示了 SUB0CFG，表 9-61 中对此进行了介绍。

返回到汇总表。

该寄存器用于启用订阅者 0 事件并定义所选 DIO 0-15 引脚上的输出策略。

图 9-61. SUB0CFG

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED				INDEX			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
保留						OUTPOLICY	
R/W-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/W-0h

表 9-61. SUB0CFG 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	0h	
19-16	INDEX	R/W	0h	指示订阅者操作所针对的低 16 位中的特定位 0h = 订阅者操作所针对的特定位为位 0 Fh = 订阅者操作所针对的特定位为位 15
15-10	保留	R/W	0h	
9-8	OUTPOLICY	R/W	0h	这些位配置订阅者 0 事件的输出策略。 0h = 所选 DIO 引脚被置位 1h = 所选 DIO 引脚被清零 2h = 所选 DIO 引脚被切换
7-1	RESERVED	R/W	0h	
0	ENABLE	R/W	0h	该位用于启用订阅者 0 事件。 0h = 订阅者 0 事件禁用 1h = 订阅者 0 事件启用

9.3.59 FILTEREN15_0（偏移 = 1508h）[复位 = 00000000h]

图 9-62 展示了 FILTEREN15_0，表 9-62 中对此进行了介绍。

返回到汇总表。

DIN0-DIN15 数字干扰滤波器的可编程计数器长度

图 9-62. FILTEREN15_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN15		DIN14		DIN13		DIN12		DIN11		DIN10		DIN9		DIN8	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN7		DIN6		DIN5		DIN4		DIN3		DIN2		DIN1		DIN0	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

表 9-62. FILTEREN15_0 字段说明

位	字段	类型	复位	说明
31-30	DIN15	R/W	0h	DIN15 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
29-28	DIN14	R/W	0h	DIN14 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
27-26	DIN13	R/W	0h	DIN13 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
25-24	DIN12	R/W	0h	DIN12 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
23-22	DIN11	R/W	0h	DIN11 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
21-20	DIN10	R/W	0h	DIN10 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
19-18	DIN9	R/W	0h	DIN9 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样

表 9-62. FILTEREN15_0 字段说明 (continued)

位	字段	类型	复位	说明
17-16	DIN8	R/W	0h	DIN8 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
15-14	DIN7	R/W	0h	DIN7 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
13-12	DIN6	R/W	0h	DIN6 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
11-10	DIN5	R/W	0h	DIN5 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
9-8	DIN4	R/W	0h	DIN4 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
7-6	DIN3	R/W	0h	DIN3 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
5-4	DIN2	R/W	0h	DIN2 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
3-2	DIN1	R/W	0h	DIN1 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
1-0	DIN0	R/W	0h	DIN0 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样

9.3.60 FILTEREN31_16（偏移 = 150Ch）[复位 = 00000000h]

图 9-63 展示了 FILTEREN31_16，表 9-63 中对此进行了介绍。

返回到汇总表。

DIN16-DIN31 数字干扰滤波器的可编程计数器长度

图 9-63. FILTEREN31_16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN31		DIN30		DIN29		DIN28		DIN27		DIN26		DIN25		DIN24	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN23		DIN22		DIN21		DIN20		DIN19		DIN18		DIN17		DIN16	
R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h		R/W-0h	

表 9-63. FILTEREN31_16 字段说明

位	字段	类型	复位	说明
31-30	DIN31	R/W	0h	DIN31 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
29-28	DIN30	R/W	0h	DIN30 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
27-26	DIN29	R/W	0h	DIN29 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
25-24	DIN28	R/W	0h	DIN28 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
23-22	DIN27	R/W	0h	DIN27 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
21-20	DIN26	R/W	0h	DIN26 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
19-18	DIN25	R/W	0h	DIN25 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样

表 9-63. FILTEREN31_16 字段说明 (continued)

位	字段	类型	复位	说明
17-16	DIN24	R/W	0h	DIN24 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
15-14	DIN23	R/W	0h	DIN23 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
13-12	DIN22	R/W	0h	DIN22 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
11-10	DIN21	R/W	0h	DIN21 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
9-8	DIN20	R/W	0h	DIN20 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
7-6	DIN19	R/W	0h	DIN19 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
5-4	DIN18	R/W	0h	DIN18 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
3-2	DIN17	R/W	0h	DIN17 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样
1-0	DIN16	R/W	0h	DIN16 数字干扰滤波器的可编程计数器长度 0h = 除了 CDC 同步采样之外没有额外的滤波器 1h = 1 ULPCLK 最小采样 2h = 3 ULPCLK 最小采样 3h = 8 ULPCLK 最小采样

9.3.61 DMAMASK（偏移 = 1510h）[复位 = 00000000h]

图 9-64 展示了 DMAMASK，表 9-64 中对此进行了介绍。

返回到汇总表。

DMA MASK，用于指示允许 DMA 修改哪些位通道。

图 9-64. DMAMASK

31	30	29	28	27	26	25	24
DOUT31	DOUT30	DOUT29	DOUT28	DOUT27	DOUT26	DOUT25	DOUT24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
DOUT23	DOUT22	DOUT21	DOUT20	DOUT19	DOUT18	DOUT17	DOUT16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DOUT15	DOUT14	DOUT13	DOUT12	DOUT11	DOUT10	DOUT9	DOUT8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DOUT7	DOUT6	DOUT5	DOUT4	DOUT3	DOUT2	DOUT1	DOUT0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 9-64. DMAMASK 字段说明

位	字段	类型	复位	说明
31	DOUT31	R/W	0h	允许 DMA 修改 DOUT31 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
30	DOUT30	R/W	0h	允许 DMA 修改 DOUT30 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
29	DOUT29	R/W	0h	允许 DMA 修改 DOUT29 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
28	DOUT28	R/W	0h	允许 DMA 修改 DOUT28 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
27	DOUT27	R/W	0h	允许 DMA 修改 DOUT27 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
26	DOUT26	R/W	0h	允许 DMA 修改 DOUT26 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
25	DOUT25	R/W	0h	允许 DMA 修改 DOUT25 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
24	DOUT24	R/W	0h	允许 DMA 修改 DOUT24 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
23	DOUT23	R/W	0h	允许 DMA 修改 DOUT23 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
22	DOUT22	R/W	0h	允许 DMA 修改 DOUT22 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道

表 9-64. DMAMASK 字段说明 (continued)

位	字段	类型	复位	说明
21	DOUT21	R/W	0h	允许 DMA 修改 DOUT21 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
20	DOUT20	R/W	0h	允许 DMA 修改 DOUT20 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
19	DOUT19	R/W	0h	允许 DMA 修改 DOUT19 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
18	DOUT18	R/W	0h	允许 DMA 修改 DOUT18 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
17	DOUT17	R/W	0h	允许 DMA 修改 DOUT17 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
16	DOUT16	R/W	0h	允许 DMA 修改 DOUT16 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
15	DOUT15	R/W	0h	允许 DMA 修改 DOUT15 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
14	DOUT14	R/W	0h	允许 DMA 修改 DOUT14 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
13	DOUT13	R/W	0h	允许 DMA 修改 DOUT13 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
12	DOUT12	R/W	0h	允许 DMA 修改 DOUT12 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
11	DOUT11	R/W	0h	允许 DMA 修改 DOUT11 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
10	DOUT10	R/W	0h	允许 DMA 修改 DOUT10 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
9	DOUT9	R/W	0h	允许 DMA 修改 DOUT9 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
8	DOUT8	R/W	0h	允许 DMA 修改 DOUT8 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
7	DOUT7	R/W	0h	允许 DMA 修改 DOUT7 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
6	DOUT6	R/W	0h	允许 DMA 修改 DOUT6 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
5	DOUT5	R/W	0h	允许 DMA 修改 DOUT5 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道

表 9-64. DMAMASK 字段说明 (continued)

位	字段	类型	复位	说明
4	DOUT4	R/W	0h	允许 DMA 修改 DOUT4 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
3	DOUT3	R/W	0h	允许 DMA 修改 DOUT3 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
2	DOUT2	R/W	0h	允许 DMA 修改 DOUT2 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
1	DOUT1	R/W	0h	允许 DMA 修改 DOUT1 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道
0	DOUT0	R/W	0h	允许 DMA 修改 DOUT0 0h = 不允许 DMA 修改此位通道 1h = 允许 DMA 修改此位通道

9.3.62 SUB1CFG（偏移 = 1520h）[复位 = 00000000h]

图 9-65 展示了 SUB1CFG，表 9-65 中对此进行了介绍。

返回到汇总表。

该寄存器用于启用订阅者 1 事件并定义所选 DIO 16-31 引脚上的输出策略。

图 9-65. SUB1CFG

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED				INDEX			
R/W-0h				R/W-0h			
15	14	13	12	11	10	9	8
保留						OUTPOLICY	
R/W-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/W-0h

表 9-65. SUB1CFG 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	0h	
19-16	INDEX	R/W	0h	指示订阅者操作所针对的高 16 位中的特定位 0h = 订阅者操作所针对的特定位为位 16 Fh = 订阅者操作所针对的特定位为位 31
15-10	保留	R/W	0h	
9-8	OUTPOLICY	R/W	0h	这些位配置订阅者 1 事件的输出策略。 0h = 所选 DIO 引脚被置位 1h = 所选 DIO 引脚被清零 2h = 所选 DIO 引脚被切换
7-1	RESERVED	R/W	0h	
0	ENABLE	R/W	0h	该位用于启用订阅者 1 事件。 0h = 订阅者 1 事件禁用 1h = 订阅者 1 事件启用

章节 10
ADC

ADC 是一款高性能逐次逼近型寄存器 (SAR) 模数转换器。本章介绍 ADC 外设的特性和运行情况。

10.1 ADC 概述

ADC 支持测量模拟信号，并以最少的 CPU 干预将其转换为数字表示形式。

此 ADC 支持快速的 12 位、10 位和 8 位模数转换，具有 12 位 SAR 内核、采样和转换模式控制功能和多达 12 个独立的转换和控制缓冲器。此 ADC 允许在无需 CPU 干预的情况下，转换和存储多达 12 个独立的模数转换器 (ADC) 样本。

ADC 模块特性包括：

- 12 位分辨率下的转换率为 4Msps
- 集成硬件过采样，平均处理多达 128 个样本
- 满量程 ADC 工作电压范围
- 由软件或定时器控制的可编程采样周期的采样保持
- 两个采样触发源：软件触发和事件触发
- 可通过软件选择 1.4V 或 2.5V 片上基准电压
- 可配置 ADC 基准源：VDD、内部基准 (VREF) 或外部基准（VREF+ 和 VREF-）
- 多达 16 个可单独配置的外部输入通道
- 具有用于温度检测、电源监控和模拟信号链的内部转换通道（请参阅器件特定的数据表，了解可用性和通道映射）
- 可配置 ADC 时钟源
- 单通道、重复单通道、序列（自动扫描）和重复序列（重复自动扫描）转换模式
- 12 个转换结果存储寄存器 (MEMRES0:11)
- 使用窗口比较器对来自转换结果寄存器的输入信号进行低功耗监控
- DMA 支持在传输完成时生成中断事件
- 在 RUN、SLEEP 和 STOP 模式下运行
- 可以在除 SHUTDOWN 之外的任何工作模式下触发
- 针对低功耗工作模式的自动电源、基准和时钟控制
- 支持并行的两个 ADC 同时和同步运行（采样和转换）
- CDAC 修整值半自动校准

图 10-1 展示了 ADC 外设的功能方框图。

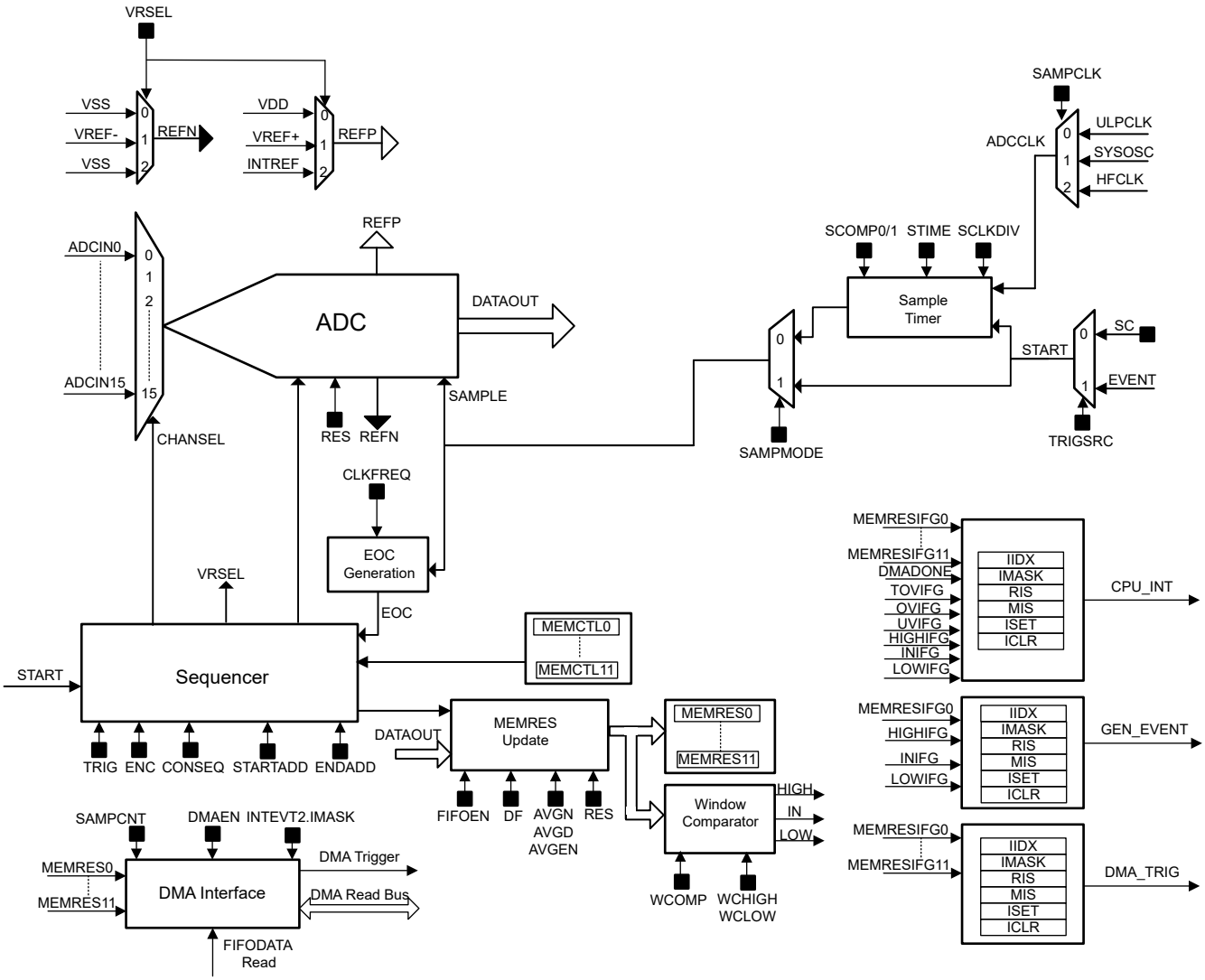


图 10-1. ADC 方框图

备注

采样时钟 (SAMPCLK) 以 ULPCLK、SYSOSC 或 HFCLK 为时钟源，转换时钟 (CONVCLK) 以 ADC IP 内的 80Mhz 振荡器为时钟源。

10.2 ADC 操作

通过用户软件配置 ADC。以下各节介绍了 ADC 的设置和运行。

备注

ADC 结果寄存器 (MEMRES) 在器件上的两个地址区域之间存在别名：

- 主区域，CPU 或 DMA 可通过该区域以 ULPCLK 速率访问所有 ADC 寄存器
- 别名区域，CPU 或 DMA 可通过该区域以 MCLK 速率访问 ADC MEMRES 寄存器以快速读取 ADC 结果

在以相同频率运行 MCLK 和 ULPCLK 的器件上，通过别名区域访问 ADC MEMRES 寄存器没有性能优势。在 MCLK 大于 ULPCLK 的设备配置上，建议使用别名区域。但是，应用软件可以在所有器件上使用别名区进行 MEMRES 访问，以保持软件实现的一致性。

10.2.1 ADC 内核

ADC 内核将模拟输入转换为数字表示形式。该内核利用两个电压电平（V_{R+} 和 V_{R-}）来定义转换的上限和下限。当输入信号等于或高于 V_{R+} 时，数字输出 (N_{ADC}) 为满量程，当输入信号等于或低于 V_{R-} 时，数字输出为零。输入通道和正基准电平 (V_{R+}) 在转换控制存储器中进行了定义。

下面的方程式 19 显示了 n 位分辨率模式下 ADC 结果 N_{ADC} 的转换公式。

$$N_{ADC} = \left(2^n - 1\right) \times \frac{(V_{in} + 0.5LSB) - V_{R-}}{V_{R+} - V_{R-}} \quad \text{其中，} \quad LSB = \frac{V_{R+} - V_{R-}}{2^n} \tag{19}$$

备注

该 ADC 支持的 V_{R-} 为 0V。所有后续公式和部分都将反映此固有属性。

假设此 ADC 中的 V_{R-} 为 0V，N_{ADC} 的公式如下：

$$N_{ADC} = \left(2^n - 1\right) \times \frac{(V_{in} + 0.5LSB)}{V_{R+}} \quad \text{其中，} \quad LSB = \frac{V_{R+}}{2^n} \tag{20}$$

下面的方程式 21 描述了 ADC 输出饱和时的输入电压：

$$V_{in} = V_{R+} - 1.5LSB \tag{21}$$

备注

ADC 在 STANDBY 或 SHUTDOWN 模式下无法正常工作。

10.2.2 电压基准选项

可以通过 MEMCTL 寄存器中的 VRSEL 位来配置 ADC 电压基准 (V_{R+})。可以选择不同的基准源以在不同的通道上进行转换。有三个选项可用于为 ADC 提供基准电压：

1. 通过 VREF+和 VREF- 引脚为 ADC 提供外部基准
2. MCU 电源电压 (VDD)
3. 可配置的 1.4 V 和 2.5 V 内部基准电压 (VREF)

为 ADC 提供外部基准时，通过适当的去耦电路将 VREF+ 引脚连接到基准源，并将 VREF- 引脚接地。

内部基准 (VREF) 是由包括 ADC 外设在内的多个模拟外设共享的电压基准模块。VREF+ 引脚上提供此内部基准，并建议在器件引脚上连接一个具有指定值的去耦电容。请参阅器件特定的数据表以了解建议的电容值。为 ADC 提供内部基准作为基准时，对 ADC 采样率没有限制，因此可以全速运行。

10.2.3 通用分辨率模式

ADC 支持在 12 位（默认）、10 位和 8 位分辨率模式下运行。分辨率模式通过 CTL2 寄存器中的 RES 位进行配置。

- 当选择 12 位模式时，转换阶段总共需要 14 个转换时钟周期
- 当选择 10 位模式时，转换阶段总共需要 12 个转换时钟周期
- 当选择 8 位模式时，转换阶段总共需要 9 个转换时钟周期

转换时钟 (CONVCLK) 源自 ADC IP 内的 80MHz 振荡器。转换窗口基于分辨率模式和 ADCCLK 的频率。有关更多详细信息，请参阅图 10-2 and 图 10-3。

10.2.4 硬件均值计算

该 ADC 在硬件中实现数字样本均值计算（硬件均值计算），无需软件或 CPU 干预即可有效地提高 ADC 的有效分辨率。硬件均值计算功能通过 CTL1 寄存器中的 AVGN 和 AVGD 位进行配置。

- AVGN 定义为当前 MEMCTLx 累积的转换次数
- AVGD 定义累加值使用位移所除以的值

备注

MEMRES 结果寄存器最长为 16 位。如果没有适当地移位，结果将被截断。

表 10-1. 可用的硬件均值计算设置

位字段值	AVGN 设置（累积样本数）	AVGD 设置（右移位数）
0x0	0	0
0x1	2	1
0x2	4	2
0x3	8	3
0x4	16	4
0x5	32	5
0x6	64	6
0x7	128	7

均值计算配置是全局配置，适用于启用均值计算功能的任何通道。每个通道无法定义不同的均值计算配置。通过 MEMCTL 寄存器的 AVGEN 位可以为每个通道启用均值计算功能。当已启用均值计算的通道收到样本触发器时，会自动连续执行所需的转换次数，并且最终的均值会存储在 MEMRES 寄存器或 FIFODAT 中。

备注

在使用硬件均值计算功能时，必须将数据格式选择为无符号二进制。

10.2.5 ADC 时钟

ADC 外设时钟 (ADCCLK) 由节 2.4 提供，用于采样时钟 (SAMPCLK)。SYSOSC、HFCLK 和 ULPCLK 是可用于 ADCCLK 的时钟源，可支持高达 48MHz 的频率。请参阅特定器件数据表，了解支持的 ADCCLK 频率。ULPCLK 是所有外设的总线时钟，对于确定性采样开始和同步采样非常有用。当需要一个非常精确、低抖动的采样周期时，使用 HFCLK 作为 ADCCLK 的时钟源非常有用。.通过对 CLKCFG 寄存器中的 SAMPCLK 位进行编程，可以选择 ADC 时钟源。转换时钟源自专用的 80MHz 本地振荡器，此振荡器可实现高达 4Msps 的高速 12 位转换。

SYSOSC 需要激活才能使 ADC 正常运行。如果 SYSOSC 未运行且 ADC 被触发，ADC 将在转换期间自动请求 SYSCTL 来启用 SYSOSC 并将其设置为基频。如果 SYSOSC 已启用，它将保持相同的频率。唯一的例外是处于 STOP1 工作模式，在该模式下，当触发 ADC 时，SYSOSC 将变为基频。

为了提供一种方法来确保电源模式之间以可预测的采样率运行，可以设置 CCONRUN 和 CCONSTOP 位向 ADC 发出信号：预计在器件分别处于 RUN 和 STOP 模式时，SYSOSC 将已经打开。当这些位置位时，ADC 不会等待 SYSCTL 的 ACK 以确保 SYSOSC 在开始采样之前正在运行。此功能使用户能够在不要求确定性采样时序的应用中灵活地省电。有关如何正确使用 CCONRUN 和 CCONSTOP 控制位的示例，请参阅节 10.2.6。

用户必须根据预期的 ADCCLK 频率将 CLKFREQ 寄存器中的 FRANGE 位配置为适当的设置。ADC 使用 CLKFREQ.FRANGE 值和由 CTL2.RES 位配置的分辨率模式来确定转换在向转换结束 (EOC) 发出信号之前需要多少个 ADCCLK 时钟周期。有关如何正确配置 CLKFREQ 寄存器以及它如何影响 EOC 信号生成的更多详细信息，请参阅下面的表 10-2。

表 10-2. CLKFREQ 寄存器配置

CLKFREQ.FRANGE 值	ADCCLK 频率范围 (MHz)	转换时长（时钟周期）		
		12 位	10 位	8 位
0	>1 至 4	1	1	1
1	>4 至 8	2	2	1

表 10-2. CLKFREQ 寄存器配置 (continued)

CLKFREQ.FRANGE 值	ADCCLK 频率范围 (MHz)	转换时长（时钟周期）		
		12 位	10 位	8 位
2	>8 至 16	3	3	2
3	>16 至 20	4	4	3
4	>20 至 24	5	4	3
5	>24 至 32	6	6	4
6	>32 至 40	8	7	5
7	>40 至 48	9	8	6

10.2.6 常见的 ADC 用例

从工作模式和时钟的角度来看，有许多 ADC 用例，但其中大多数适合以下项目之一：

- 在 RUN 或 SLEEP 模式下触发
  - 如果触发 ADC 以启动转换（软件或事件），并且器件处于 RUN0/RUN1/SLEEP0/SLEEP1 模式（SYSOSC 已在任何频率下运行），则：
    - 在此模式下，采样时钟可以是 ULPCLK、HFCLK 或 SYSOSC
    - 转换将在不改变 SYSOSC 频率的情况下运行
    - 允许使用 4MHz、16MHz、24MHz 或 32MHz SYSOSC 频率
  - 如果触发 ADC 以启动转换（软件或事件），并且器件处于 RUN2 或 SLEEP2 模式（SYSOSC 被禁用，MCLK = LFCLK = 32kHz），则：
    - 在此模式下，采样时钟可以是 ULPCLK 或 SYSOSC
    - SYSCTL 将 ADC CLK REQ 解读为异步快速时钟请求，在 32MHz 时启用 SYSOSC，并强制 MCLK 或 ULPCLK 为 32MHz，直到 ADC 使该请求无效
    - 在此用例中 CCONRUN 必须清零
    - 在此用例中 CCONSTOP 必须清零
- 在 STOP 模式下触发
  - 在此模式下，采样时钟可以是 ULPCLK 或 SYSOSC
  - 如果触发 ADC 以启动转换（事件），并且器件处于 STOP0 模式（SYSOSC 以任意频率运行，ULPCLK=4MHz），则：
    - 转换将在不改变 SYSOSC 频率的情况下运行
    - 允许使用 4MHz、16MHz、24MHz 或 32MHz SYSOSC 频率
  - 如果触发 ADC 以启动转换（事件），并且器件处于 STOP1 模式（SYSOSC 转换为 4MHz），则：
    - 当接收到 ADC CLK REQ 时，SYSCTL 将强制 SYSOSC 为 BASE（与运行模式一致，因为设置了 USE4MHZSTOP），而 SYSCTL 将在 ADC CLK REQ 被移除后将 SYSOSC 释放回 4MHz
    - CCONRUN 必须清零
    - CCONSTOP 必须清零
  - 如果触发 ADC 以启动转换（事件），并且器件处于 STOP2 模式（SYSOSC 禁用），则：
    - 触发事件以 32kHz 的频率通过事件结构传播，ADC 接收触发并使 ADC CLK REQ (CPCLK REQ) 对 SYSCTL 有效，SYSCTL 接收 ADC CLK REQ 作为异步快速时钟请求，暂停 STOP 模式，以 32MHz 的频率启用 SYSOSC，并强制 MCLK 或 ULPCLK 为 32MHz，直到 ADC 使 ADC CLK REQ 无效
    - CCONRUN 必须清零
    - CCONSTOP 必须清零
- 在 STANDBY 模式下触发
  - 在此模式下，采样时钟可以是 ULPCLK 或 SYSOSC
  - 如果触发 ADC 以启动转换（事件），并且器件处于 STANDBY0 模式（SYSOSC 被禁用但 ULPCLK 正在运行），则：

- 触发事件以 32kHz 的频率通过事件结构传播，ADC 将接收触发并使 ADC CLK REQ (CPCLK REQ) 对 SYSCTL 有效，SYSCTL 将 ADC CLK REQ 解读为异步快速时钟请求，暂停 STANDBY 模式，以 32MHz 的频率启用 SYSOSC，并强制 MCLK/ULPCLK 为 32MHz，直到 ADC 使 ADC CLK REQ 无效
- CCONRUN 必须清零
- CCONSTOP 必须清零

– 如果触发 ADC 以启动转换（TIMG0 或 TIMG1 事件），并且器件处于 STANDBY1（在 STOPCLKSTBY 置位的情况下选通 ULPCLK），则：

- TIMG0 或 TIMG1 事件本身将触发异步快速时钟请求以暂停 STANDBY 模式，以 32MHz 的频率启动 SYSOSC，并强制 MCLK 或 ULPCLK 为 32MHz；然后，TIMG0 或 TIMG1 事件有 32 个 SYSOSC 周期来继续完成整个事件结构，ADC 则有 32 个 SYSOSC 周期来捕获计时器事件并使 ADC CLK REQ 有效来保持 SYSOSC 启用以运行转换
- 当 ADC 使 ADC CLK REQ 无效时，ULPCLK 将运行 32 个额外周期使任何 ADC 事件（DMA 请求或 IRQ）传播，之后 SYSCTL 将通过 STOPCLKSTBY (STANDBY1) 恢复 STANDBY 模式
- CCONRUN 必须清零
- CCONSTOP 必须清零

10.2.7 断电行为

PWREN 寄存器中的 ENABLE 位可以启用或禁用 ADC 外设。为了省电，应在不使用 ADC 时将其禁用。CTL0 寄存器中的 PWRDN 位可以在 AUTO 和 MANUAL 之间选择 ADC 断电策略。此策略在 ADC 以 RUN、SLEEP 和 STOP MCU 功耗模式运行时生效。

应根据所需的最大 ADC 采样率和不同 MCU 功耗模式下的运行需求配置 PWRDN。在 STOP 模式下运行期间，ADC 硬件不会将断电策略强制设置为 AUTO。无论器件功耗模式如何，此策略都遵循用户设置。

PWRDN 的复位值为 0，其默认行为是在转换结束时以及不需要立即将下一个采样信号设置为有效状态时使 ADC 外设自动断电。当 PWRDN 位设置为“1”时，选择的是手动断电行为。在该设置中，ADC 在转换结束时不会断电，而是保持启用状态。这意味着 ADC 外设只能通过 PWREN 寄存器断电。

备注

有关 ADC 唤醒和启用时间的规格，请参阅器件特定的数据表。

10.2.8 采样触发源和采样模式

采样触发器

可以通过 CTL1 寄存器中的 TRIGSRC 位选择两个采样触发源：一个是软件触发器，另一个是事件触发器。

当选择软件触发器作为触发源时，应用软件可以设置 CTL1 寄存器中的 SC 位来启动采样阶段。当选择事件触发器作为触发源时，事件管理器中所选事件的上升沿将启动采样阶段。事件始终为边沿触发。

采样模式

有两种可用的采样模式：AUTO 和 MANUAL，可以通过 CTL1 寄存器中的 SAMPMODE 位进行选择。

10.2.8.1 自动采样模式

在自动模式下，采样信号的生成与采样时钟 (SAMPCLK) 同步，并且可使用内部采样计时器来设定，以确定采样窗口的持续时间。采样计时器为 10 位宽，并且有两个采样时间比较寄存器 (SCOMPx) 可用于考虑测量信号所使用的各种源阻抗。可以使用 MEMCTL 寄存器中的 STIME 位来选择这两个 SCOMP 寄存器中的一个。

从触发采样到采样周期开始有 2-3 个周期的延迟。通过将 ULPCLK 设置为 ADCCLK 的源，可以绕过该延迟。这种同步旁路特性对于确定性采样和两个 ADC 外设的同步运行非常有用。

图 10-2 显示了当 ADC 配置为自动采样模式时的 ADC 采样和转换时序图。

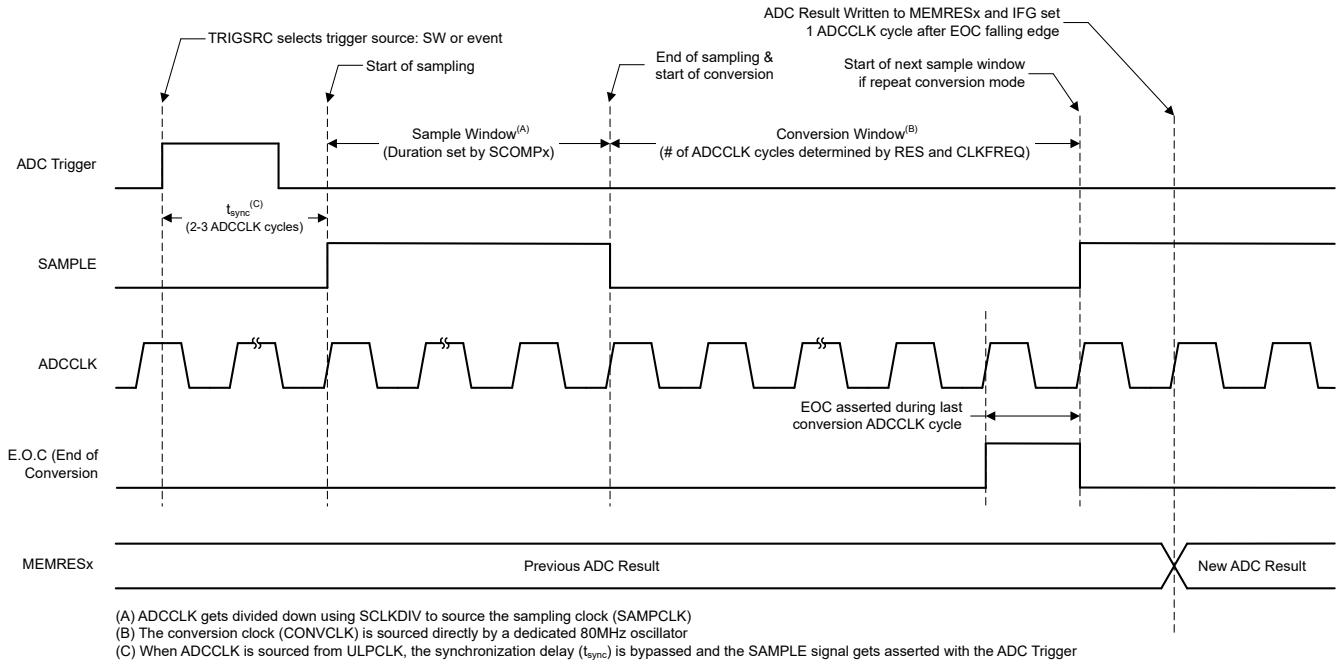


图 10-2. 自动采样模式 ADC 采样和转换时序图

备注

当 PWRDN 的复位值设置为“0”时（此时具有自动断电的默认行为），需要在每个采样窗口中考虑 ADC 唤醒/启用时间。有关 ADC 唤醒/启用时间的规范，请参阅器件特定的数据表。例如，如果最大 ADC 唤醒/启用时间为 5uS，则意味着 SCOMPx 设置的持续时间应 > (5uS + 采样窗口持续时间)。

10.2.8.2 手动采样模式

在手动模式下，当设置 SC 位时将生成采样信号，该信号可以与采样时钟异步。采样窗口的持续时间由软件通过将 SC 位保持在高电平来控制。由于事件始终是边沿触发的，因此，任何转换模式都不支持通过事件触发的手动模式。仅单通道单次转换模式支持手动采样模式的软件触发，而其他三种转换模式均不支持。

从采样窗口结束到转换窗口开始，具有 2-3 个周期的同步延迟。

图 10-3 显示了当 ADC 配置为手动采样模式时的 ADC 采样和转换时序图：

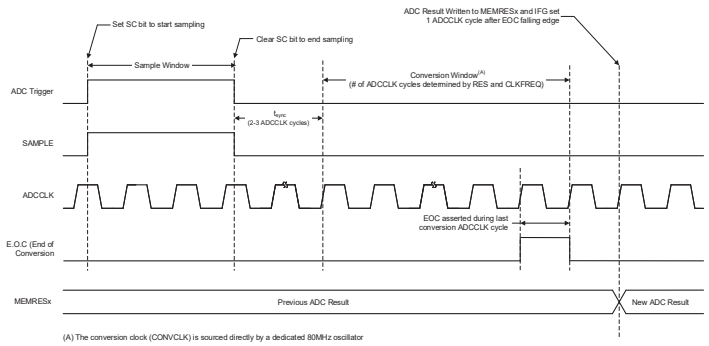


图 10-3. 手动采样模式 ADC 采样和转换时序图

备注

1. 在手动采样模式下，CLKCFG 寄存器中的 CCONRUN 位必须设置为 1。
2. 当 PWRDN 的复位值设置为“0”时（此时具有自动断电的默认行为），需要考虑在采样窗口之前留出 ADC 唤醒/启用时间。这意味着，在设置 PWREN 寄存器中的 ENABLE 位以启用 ADC 之后，应用软件应设置延迟以等待 ADC 唤醒/启用时间，然后再开始采样。有关 ADC 唤醒/启用时间的规范，请参阅器件特定的数据表。
3. 一旦 SC 位由软件设置，SC 位将在采样持续时间后由硬件自动清除，采样持续时间是模拟信号被采样的持续时间。如果软件尝试在采样持续时间 + 2~3 个周期同步延迟之前设置 SC 位，则 TOVIFG 标志将不会被设置。

10.2.9 采样周期

使用 CLKCFG 寄存器中的 SAMPCLK 位在 SYSCTL 模块中可以选择采样时钟源。运行 ADC 所需的采样周期可通过内部时钟分频器和/或采样计时器产生，这种情况适用于 AUTO 采样模式。使用 CTL0 寄存器中的 SCLKDIV 位可以配置内部时钟分频器，分频选项为 1、2、4、8、16、24、32 和 48。

采样周期的持续时间可以编程为由 SCOMP0 和 SCOMP1 采样计时器寄存器设置的两个用户定义值之一。SCOMPx 中的值可以定义采样时钟数来设置采样窗口，从而配置采样周期。默认的 SCOMPx 采样计时器值会转换为 1 个周期宽的采样脉冲，使采样周期可以完全基于采样时钟和 SCLKDIV。通常，可以利用三个旋钮来控制采样周期：SCOMPx、SCLKDIV 和采样时钟源。

当使用 PWRDN=0 选择 AUTO 断电模式时，会在采样信号生效后的一个采样时钟周期内产生 ADC 外设的模块使能信号。除了温度传感器、VREF、OPA 等其他模拟模块的 ADC 功耗时间或稳定时间需求外，用户还应在采样窗口计算中考虑这一点。

10.2.10 转换模式

ADC 提供四种转换模式：

1. 单通道单次转换
   - 可使用 MEMCTL 选择通道
   - 所选通道仅采样和转换一次
   - 启用硬件均值计算后，执行多次转换
2. 单通道重复转换
   - 可使用 MEMCTL 选择通道
   - 所选通道将被重复采样和转换，直到 ENC 被软件清零
     - 如果设置了 TRIG 位，那么需要一次触发来进行下一个转换
   - 启用硬件均值计算后，执行多次转换
3. 序列通道单次转换
   - 可使用 STARTADD、ENDADD 和 MEMCTL 寄存器形成通道组
   - 组中的每个通道仅采样和转换一次
   - 启用硬件均值计算后，序列期间会在一个通道上执行多次转换
   - 即使 ENC 在序列中被清零，也将完成该序列
4. 序列通道重复转换
   - 可使用 STARTADD、ENDADD 和 MEMCTL 寄存器形成通道组
   - 通道组将被重复采样和转换，直到 ENC 被软件清零
     - 如果设置了 TRIG 位，那么需要一次触发来进行下一个转换
   - 如果 ENC 被清零，则在最后一次转换结束时停止操作
   - 启用均值计算后，序列期间会在一个通道上执行多次转换

以下步骤概述了为所需转换模式配置 ADC 的建议过程：

1. 使用 CTL1 寄存器中的 CONSEQ 位选择所需的 ADC 转换模式
2. 使用 CTL2 寄存器中的 STARTADD 位，为一种序列模式选择用于单次转换或用作第一个 MEMCTL 的 MEMCTLx
3. 选定一种序列模式时，使用 CTL2 寄存器中的 ENDADD 位来选择用于该序列最后一次转换的 MEMCTLx
4. 使用 CHANSEL 位将 ADC 输入通道分配给相应的 MEMCTLx 寄存器
   - 对于序列模式，您必须为配置序列中的每个 MEMCTLx 分配一个 ADC 输入通道
5. 使用 CTL1 寄存器中的 TRIGSRC 位选择硬件或软件触发
6. 使用 CTL1 寄存器中的 SAMPMODE 位选择自动或手动采样模式
   - 如果使用自动模式，则在 SCOMPx 寄存器中设置所需的采样计时器值，并使用 MEMCTLx 寄存器中的 STIME 位选择适当的采样计时器源（SCOMP0 或 SCOMP1）
7. 如果使用单通道重复转换或序列转换模式，则对每个 MEMCTLx 寄存器中的 TRIG 位进行编程，以指示是否需要触发来发展到序列中的下一个 MEMCTL
8. 设置 CTL1 寄存器中的 ENC 位以启用 ADC 转换
9. 下表矩阵描述了基于所选触发和采样模式的 ADC 配置和用法的下一步：

表 10-3. 触发和采样模式 ADC 用法矩阵

	触发模式	
	软件触发	事件触发

表 10-3. 触发和采样模式 ADC 用法矩阵 (continued)

采样模式	自动	• 设置 SC 位以开始采样相位（持续时间由采样计时器决定） • 一旦采样相位结束，转换便会开始 • 在单通道单次转换中，当转换结束时，ENC 将清零 • 一旦捕获到触发，SC 位就会自动清零 • 对于重复和序列模式，如果在 MEMCTL 中 TRIG 被置位，则需要将 SC 位置位才能继续下一次转换	• 硬件触发开始采样相位（持续时间由采样计时器决定） • 一旦采样相位结束，转换便会开始 • 在单通道单次转换中，当转换结束时，ENC 将清零 • 对于重复和序列模式，ADC 等待硬件触发或根据 TRIG 设置自动开始下一次转换
	手动	• 设置 SC 位以开始采样相位（SC 位不会自动复位） • 将 SC 位清零以结束采样相位并开始转换 • 在单通道单次转换中，当转换结束时，ENC 位将清零 • 此配置不支持重复/序列转换模式和硬件均值计算	此配置不支持 ADC 操作

10. ADC 结果存储在相关 MEMCTL 的 MEMRES 寄存器中（例如，MEMCTL0 结果存储在 MEMRES0 中）。
    - 对于重复转换模式，在每次相关的 MEMCTL 转换后，MEMRES 中的结果都会更新
11. 对于重复转换模式，将 ENC 位清零以停止 ADC 操作

10.2.11 数据格式

ADC 支持两种数据格式 – 无符号二进制和二进制补码有符号二进制。无符号二进制结果右对齐存储在 MEMRES 寄存器或 FIFO 中。有符号二进制结果左对齐存储在 MEMRES 寄存器或 FIFO 中。

表 10-4. ADC 数据格式

数据格式	分辨率	结果范围（十进制）	结果范围（十六进制）
无符号	8 位	0 至 255	0000h 至 00FFh
	10 位	0 至 1023	0000h 至 03FFh
	12 位	0 至 4095	0000h 0FFFh
有符号	8 位	-128 至 127	8000h 至 7F00h
	10 位	-512 至 511	8000h 至 7FC0h
	12 位	-2048 至 2047	8000h 至 7FF0h

10.2.12 高级特性

以下各节介绍了 ADC 外设提供的其他特性和优势，以及如何在应用中利用这些特性和优势。

10.2.12.1 同时采样

电流和电压检测等一些应用需要同时在多个模拟信号中进行测量。在这些情况下，需要在单个 MCU 上使用多个 ADC 来执行同步采样。MSPM0xx 平台中具有多个 ADC 外设的任何器件都支持同步采样。

对于这个用例，应该使用 ULPCLK 作为 ADCCLK 的源，因为它也是 PD0 中所有外设的时钟。这意味着它已经与总线时钟同步，所以可以确保采样开始的确定性时序。除了使用 ULPCLK 作为两个 ADC 外设的采样时钟源外，用户还应选择相同的采样触发器，并在两个 ADC 上使用相同的值对时钟预分频器和/或 SCOMP 进行编程。

10.2.12.2 窗口比较器

ADC 中有一个可用的窗口比较器单元，可用于检查输入信号是否处于软件设置的预定义阈值范围内。进入 MEMRES 或 FIFO 的 ADC 结果将根据窗口比较器的阈值进行检查。

根据比较结果可以产生 3 个中断条件：

1. LOWIFG – 转换结果低于下限阈值 (WCLOW)
2. HIGHIFG – 转换结果高于上限阈值 (WCHIGH)
3. INIFG – 转换结果介于或等于下限阈值和上限阈值

窗口比较器下限阈值和上限阈值是适用于所有通道的全局阈值，并且可以根据需要使用 MEMCTL 寄存器中的 WINCOMP 位为每个通道启用窗口比较特性。

当 ADC 结果数据格式 (CTL2.DF) 或分辨率 (CTL2.RES) 配置发生更改时，窗口比较器阈值不会由硬件复位，而是按原样保留。在更改数据格式和/或分辨率配置后，软件应用程序应该会根据需要重新配置阈值。

10.2.12.3 DMA 和 FIFO 操作

ADC 具有一个用于与 DMA 进行通信的专用接口。该接口可使用 DMA 自动将 ADC 结果存储到存储器，从而有助于减轻 CPU 的工作负载。图 10-4 展示了构成此接口的信号：

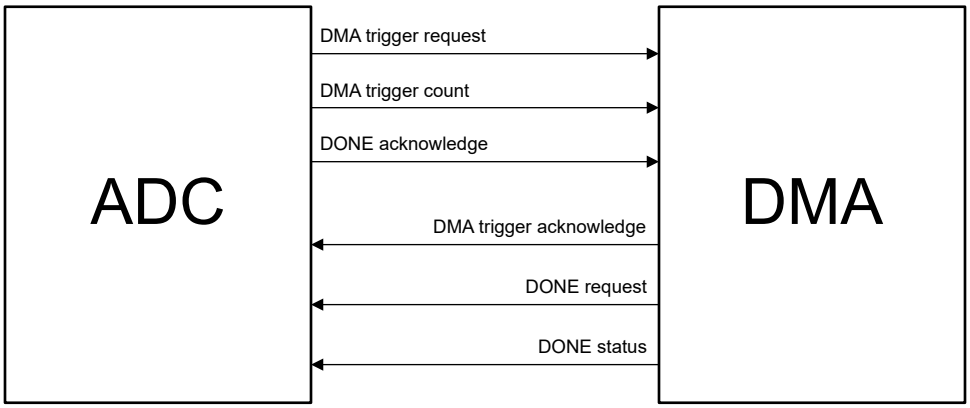


图 10-4. 内部 ADC-DMA 接口

备注

“DMA 触发计数”信号指示 DMA 在一次触发请求时可以传输的样本数。ADC 使用“DONE 状态”信号生成 DMA DONE 中断，该信号指示编程块大小的 DMA 数据传输是否完成。

CTL2 寄存器中的 DMAEN 位用于使 DMA 能够进行 ADC 数据传输。当 DMA 发出“DONE 状态”信号时，ADC 硬件会将 DMAEN 位清零。预计软件将使用 DMAEN 重新启用 DMA，以使 ADC 生成下一个 DMA 触发。

ADC 还包含一个可选的先入先出缓冲区，提供了一种方法来存储 ADC 结果供将来使用，例如由 DMA 传输到存储器。无论是启用还是禁用 FIFO，CPU 和 DMA 均可用于从 ADC 移动数据。第三个事件发布者的 RIS 寄存器中的存储器结果标志用作 FIFO 阈值，可以取消屏蔽以生成 DMA 触发。

以下各节介绍了在各种转换模式下以及在启用或禁用 FIFO 的情况下使用 ADC+DMA/CPU 的详细信息

非 FIFO 模式 (FIFOEN=0) 下的 ADC-DMA/CPU 操作

非 FIFO 模式 (FIFOEN=0) 下的 ADC-DMA/CPU 操作

- 单次转换和重复单次转换
  - 配置 STARTADD 位以选择所需的 MEMCTLx 寄存器
    - MEMCTLx 与 MEMRESx **相关**
    - MEMRESx 与 MEMRESIFGx **相关**
  - 配置 MEMCTL CHANSEL 位以选择所需的 ADC 通道
  - MEMRESx 中提供转换数据
  - 可以设置 MEMRESIFGx 以生成 CPU 中断或 DMA 触发
  - 必须通过软件将 SAMPCNT **编程为 1** 以执行 DMA 操作
  - 当 ADC 在 CPU 或 DMA 读取前一个样本之前更新 MEMRESx 时，会设置转换溢出标志 OVIFG

  - 当 CPU 或 DMA 在下一个转换结果可用前读取 MEMRESx 寄存器时，会设置转换下溢标志 UVIFG
- 序列转换和重复序列转换
  - 配置 STARTADD 位以选择序列中的第一个 MEMCTL
  - 配置 ENDADD 位以选择序列中的最后一个 MEMCTL
    - MEMCTLx 与 MEMRESx **相关**
    - MEMRESx 与 MEMRESIFGx **相关**
  - 配置每个 MEMCTLx CHANSEL 位以选择所需的 ADC 通道
  - MEMRESx 中提供转换数据
  - 可以设置 MEMRESIFGx 以生成 CPU 中断或 DMA 触发
  - 必须根据软件针对 DMA 操作进行的阈值设置，通过软件将 SAMPCNT 编程为**一个合适的值**
  - 当 ADC 在 CPU 或 DMA 读取前一个样本之前更新 MEMRESx 时，会设置转换溢出标志 OVIFG
  - 当 CPU 或 DMA 在下一个转换结果可用前读取 MEMRESx 寄存器时，会设置转换下溢标志 UVIFG

备注

对于基于 DMA 的操作，由于 DMA 源不回滚，MEMCTL 起始地址应小于单次序列转换的结束地址。重复序列转换模式不支持基于 DMA 的数据传输，因为 DMA 不支持循环寻址模式。

FIFO 模式 (FIFOEN=1) 下的 ADC-DMA/CPU 操作

- 单次转换和重复单次转换
  - 配置 STARTADD 位以选择所需的 MEMCTLx 寄存器
    - MEMCTLx 与 MEMRESx **不相关**
    - MEMRESx 与 MEMRESIFGx **相关**
  - 配置 MEMCTL CHANSEL 位以选择所需的 ADC 通道
  - 转换数据按顺序加载到 MEMRES0/1/2/…/N（FIFO 结构）
  - CPU 或 DMA 必须从专用的 FIFODAT 寄存器（而不直接从 MEMRES 寄存器）读取 ADC 样本
    - FIFO 中的数据始终压缩为两个样本，并在 CPU 或 DMA 读取 FIFODAT 时作为 32 位数据提供
  - MEMRESIFGx 可用作阈值条件以生成 CPU 中断或 DMA 触发
    - 为了充分利用 FIFO，可以使用最后一个 MEMRESIFG
  - 必须根据针对 DMA 操作进行的阈值设置，通过软件将 SAMPCNT 编程为**一个合适的值**
  - 当 ADC 在 CPU 或 DMA 读取前一个样本之前更新 MEMRESx 时，会设置转换溢出标志 OVIFG
  - 当 CPU 或 DMA 在 MEMRESx 寄存器中的转换结果可用之前读取 FIFODAT 寄存器时，会设置转换下溢标志 UVIFG。

备注

对于基于 CPU 或 DMA 的操作，不建议采用启用了 FIFO 的单次转换模式。这将导致下溢情况，并且必须在软件中丢弃不需要的 16 位数据。

- 序列转换和重复序列转换
  - 配置 STARTADD 位以选择序列中的第一个 MEMCTL
  - 配置 ENDADD 位以选择序列中的最后一个 MEMCTL
    - MEMCTLx 与 MEMRESx **不相关**
    - MEMRESx 与 MEMRESIFGx **相关**
  - 配置每个 MEMCTLx CHANSEL 位以选择所需的 ADC 通道
  - 转换数据按顺序加载到 MEMRES0/1/2/…/N（FIFO 结构）
  - CPU 或 DMA 必须从专用的 FIFODAT 寄存器（而不直接从 MEMRES 寄存器）读取 ADC 样本
    - FIFO 中的数据始终压缩为两个样本，并在 CPU 或 DMA 读取 FIFODAT 时作为 32 位数据提供
  - MEMRESIFGx 可用作阈值条件以生成 CPU 中断或 DMA 触发
    - 为了充分利用 FIFO，可以使用最后一个 MEMRESIFG
  - 必须根据针对 DMA 操作进行的阈值设置，通过软件将 SAMPCNT 编程为**一个合适的值**
  - 当 ADC 在 CPU 或 DMA 读取前一个样本之前更新 MEMRESx 时，会设置转换溢出标志 OVIFG

- 当 CPU 或 DMA 在 MEMRESx 寄存器中的转换结果可用之前读取 FIFODAT 寄存器时，会设置转换下溢标志 UVIFG

备注

- CPU 或 DMA 读取后，不会自动清除 FIFODAT 寄存器中的数据。新的转换数据会覆盖 FIFODAT 寄存器中的先前数据。
- 为确保同步读取存储 16 位样本的 32 位 FIFO 中的字节，可以使用特定的 DMA 触发器。尤其是，选择 MEMRES1、MEMRES3、MEMRES5、MEMRES7、MEMRES9 和 MEMRES11 将使 FIFO 中的字节读取与相应的 MEMRESx 字节同步。
- 如果 ADC 在重复序列模式或正常重复模式期间被禁用，则值得注意的是在 ADC 完全停止之前可能会发生额外的转换。

表 10-5. ADC-DMA/CPU 操作摘要矩阵

转换模式	FIFO 禁用 (FIFOEN=0) 未压缩样本 直接从 MEMRESx 寄存器读取		FIFO 启用 (FIFOEN=1) 始终压缩样本 仅从 FIFODAT 寄存器读取	
	CPU 读取/写入	DMA 读取/写入	CPU 读取/写入	DMA 读取/写入
单次	支持	支持 SAMPCNT=1 16 位样本	不建议 将设置下溢标志 应忽略不需要的 16 位	不建议 将设置下溢标志 应忽略不需要的 16 位
重复单次	支持	支持 SAMPCNT=1 16 位样本	支持 MEMRESIFG=CPU 中断 32 位 FIFODAT 读取	支持 MEMRESIFG=DMA 触发 SAMPCNT=32 位样本
序列	支持	支持 SAMPCNT=16 位样本 STARTADD<ENDADD	支持 MEMRESIFG=CPU 中断 32 位 FIFODAT 读取	支持 MEMRESIFG=DMA 触发 SAMPCNT=32 位样本
重复序列	支持	不支持	支持 MEMRESIFG=CPU 中断 32 位 FIFODAT 读取	支持 MEMRESIFG=DMA 触发 SAMPCNT=32 位样本

10.2.12.4 模拟外设互连

MCU 的 MSPM0 平台提供了一组丰富的高性能模拟外设，它们可以彼此交互以执行各种模拟信号链功能。以下各项介绍了 ADC 如何与其他板载模拟外设进行交互：

具有内部基准模块 (VREF) 的 ADC

ADC 具有专用的使能请求以及与内部电压基准模块的就绪接口。当采样触发时，VREF 使能变为有效，同时为 ADC 操作选择内部基准缓冲器。在 ADC 状态寄存器 (REFBUFRDY) 中捕获 VREF 的就绪响应。

VREF 的就绪响应不会选通 ADC 中的采样相位。ADC 采样窗口在采样触发后启动，并且需要在采样周期中酌情考虑内部基准缓冲器的稳定时间。软件可以使用软件使能位启用 VREF 模块，这样在 ADC 开始对输入通道采样时，该模块已经稳定。在这种情况下，采样时间可以更短，并且不必考虑 VREF 的启用时间。

具有 OPA 的 ADC

当选择 OPA 外设作为 ADC 输入通道且 MEMCTL 寄存器的 BCSEN 位置位时，ADC 会向其提供烧毁电流源 (BCS) 使能信号。该信号允许 ADC 外设与 OPA 外设协同工作，以监控传感器运行状况并检测故障。在硬件取平均值操作期间，BCS 使能信号在所有转换中保持高电平。当 ADC 辅助 OPA 斩波被激活并且硬件取平均值功能被启用时，ADC 接收一个斩波请求信号并向 OPA 外设提供一个斩波状态控制信号。在硬件取平均值操作期间，斩波状态控制信号在每次转换结束时切换。当 ADC 辅助 OPA 斩波被激活时，平均采样数量应该被设定为一个偶数值。

具有温度传感器的 ADC

选择温度感测通道后，ADC 会生成温度传感器模块使能信号。由于温度传感器没有就绪响应，因此应在采样周期内考虑温度传感器的稳定时间。

10.2.13 状态寄存器

ADC 状态寄存器 (STATUS) 包含两个位：REFBUFRDY 和 BUSY。

- 当 ADC 在使能请求生效后从内部基准缓冲器 (VREF/REFBUF) 接收到就绪信号时，便会设置 REFBUFRDY 位
- BUSY 等于“1”表示 ADC 正忙于执行采样或转换操作
  - 对于**单通道单次转换**，它表示已收到触发信号并且正在进行采样或转换。转换完成后将清除 BUSY 位
  - 对于**重复单次转换**，它表示重复单次操作已开始但尚未结束。当 ENC 被写入“0”且最后一次转换完成时将清除 BUSY 位
  - 对于**通道转换序列**，它表示通道转换序列已开始。在该序列结束时将清除 BUSY 位
  - 对于**重复通道转换序列**，它表示重复序列正在进行中。当 ENC 被写入“0”且最后一次转换完成时将清除 BUSY 位

10.2.14 ADC 事件

ADC 外设包含三个事件发布者和一个事件订阅者。

一个事件发布者 (CPU_INT) 通过静态事件路由管理到 CPU 子系统的 ADC 中断请求 (IRQ)。第二个事件发布者 (GEN_EVENT) 可用于通过通用事件路由通道向订阅者发布 ADC 事件。第三个事件发布者 (DMA_TRIG) 可用作 ADC 到 DMA 的触发器，用以通过 DMA 事件路由将 ADC 事件直接发送到 DMA。

事件订阅者 (FSUB_0) 可用于订阅通过通用事件路由通道发布到事件结构的事件。

表 10-6 中总结了 ADC 事件。

表 10-6. ADC 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	ADC	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 ADC 到 CPU 的中断路由
通用发布者事件	发布者	ADC	通用事件通道	通用路由 (FPUB_0)	GEN_EVENT 寄存器，FPUB_0 寄存器	从 ADC 触发通用事件通道
DMA 触发事件	发布者	ADC	DMA	DMA 路由	DMA_TRIG 寄存器	修复了从 ADC 到 DMA 的触发器路由
通用订阅者事件	订阅者	其他外设	ADC	通用路由 (FSUB_0)	FSUB_0	通用事件通道的 ADC 订阅

10.2.14.1 CPU 中断事件发布者 (CPU_INT)

ADC 外设提供了许多中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 10-7 中列出了来自 ADC 的 CPU 中断事件。

表 10-7. ADC CPU 中断事件条件 (CPU_INT)

索引 (IIDX)	名称	说明
0x0	NO_INTR	未置位 (IIDX.STAT = 0) 意味着没有挂起的中断请求
0x1	OVIFG	当 ADC 在 CPU 或 DMA 读取前一个样本之前更新 MEMRESx 时，设置转换溢出中断标志
0x2	TOVIFG	如果 ADC 在前一个采样+转换仍在进行中时接收到新的采样触发器，系统会设置序列转换时间溢出中断标志
0x3	HIGHIFG	当 MEMRESx 结果寄存器高于窗口比较器的 WCHIGH 阈值时，高阈值比较中断标志被置位

表 10-7. ADC CPU 中断事件条件 (CPU_INT) (continued)

索引 (IIDX)	名称	说明
0x4	LOWIFG	当 MEMRESx 结果寄存器低于窗口比较器的 WCLOW 阈值时，低阈值比较中断标志被置位
0x5	INIFG	当 MEMRESx 结果寄存器在窗口比较器的 WCLOW 和 WCHIGH 范围内时，范围内比较器中断标志被置位
0x6	DMADONE	当编程块大小的 DMA 数据传输完成时，设置 DMA 完成中断标志
0x7	UVIFG	当 CPU 或 DMA 在下一个转换结果可用前读取 MEMRESx 寄存器时，设置转换下溢中断标志 UVIFG
0x9 至 0x20	MEMRESIFG[0 至 24] (1)	当 MEMRESx 加载新的转换结果时，系统将设置存储器寄存器中断标志

(1) 请参阅器件特定的数据表，了解您的器件支持多少个转换结果存储寄存器 (MEMRES)。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。在软件读取 IIDX 寄存器或写入相应的 ICLR 寄存器位时，会清除中断 (RIS) 标志。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

10.2.14.2 通用事件发布者 (GEN_EVENT)

ADC 外设提供 4 个中断源，其中一个中断源可配置为将事件发布到通用事件路由通道。表 10-8 列出了这些中断源。

表 10-8. ADC 通用事件发布者条件 (GEN_EVENT)

索引	名称	说明
0x0	NO_INTR	没有位置位意味着没有挂起的中断请求
0x3	HIGHIFG	当 MEMRESx 结果寄存器高于窗口比较器的 WCHIGH 阈值时，高阈值比较中断标志被置位
0x4	LOWIFG	当 MEMRESx 结果寄存器低于窗口比较器的 WCLOW 阈值时，低阈值比较中断标志被置位
0x5	INIFG	当 MEMRESx 结果寄存器在窗口比较器的 WCLOW 和 WCHIGH 范围内时，范围内比较器中断标志被置位
0x9	MEMRESIFG0	当 MEMRES0 被载入一个新的转换结果时，存储器寄存器中断标志被置位

通用事件发布者配置通过 GEN_EVENT 事件管理寄存器进行管理。根据通过事件结构接收到的来自用户模块的确认 (ACK) 信号清除中断 (RIS) 标志。有关为通用事件发布者配置事件寄存器的指导，请参阅节 7.2.5。

必须通过将目标通用通道 ID 写入 ADC 中的 **FPUB_0** 寄存器来选择将 GEN_EVENT 发布到的通用事件通道。有关配置通用事件路由的指导，请参阅节 7.1.3.3。

如果应用中未使用该发布者，则 FPUB_0 寄存器可以保持断开状态（设置为零），并且不应通过 ADC GEN_EVENT 寄存器集中的 MIS 寄存器解除屏蔽任何事件。

10.2.14.3 DMA 触发事件发布者 (DMA_TRIG)

ADC 模块提供了许多可配置为 DMA 触发源的中断源。为了降低中断优先级，表 10-9 中列出了来自 ADC 的 DMA 触发事件。当 ADC 需要 DMA 通道时，应在 DMA_TRIG 的 IMASK 寄存器中取消屏蔽 DMA 触发，并根据需要配置 DMA 以支持 ADC 操作。

表 10-9. ADC DMA 触发事件条件 (DMA_TRIG)

索引 (IIDX)	名称	说明
0x0	NO_INTR	未置位 (IIDX.STAT = 0) 意味着没有挂起的中断请求
0x9 至 0x20	MEMRESIFG[0 至 24] (1)	当 MEMRESx 加载新的转换结果时，系统将设置存储器寄存器中断标志

(1) 查看器件特定数据表，了解您的器件支持多少个转换结果存储寄存器 (MEMRES)。

通过 DMA_TRIG 事件管理寄存器来管理 DMA 触发事件配置。根据来自 DMA 的 ACK 清除中断 (RIS) 标志。有关为 DMA 触发配置事件寄存器的指导，请参阅节 7.2.5。

10.2.14.4 通用事件订阅者 (FSUB_0)

ADC 外设支持从其他外设接收通过通用通道路由的事件。有关通用事件路由的工作方式，请参阅节 7.1.3.3 和节 7.2.3。

一旦确定要使用的通道并且已知所连接外设的发布者端口和订阅者端口后，请使用以下步骤建立事件连接。此示例中将配置一个由 GPIO 触发的 ADC 应用，使用 GPIO 端口 A 将事件发布到通用通道 1，并由 ADC0 订阅通用通道 1 作为转换开始触发器。

1. 配置 GPIO 端口 A 的 GEN_EVENT 寄存器，根据相应的事件（例如 DIN 上升事件）设置事件请求。
2. 将 0x1 存储到 GPIO 端口 A 的 FPUB_0 寄存器中，以便将 GEN_EVENT 寄存器选择的 GPIO 事件发布到通用路由通道 1。通道 1 不能正在被另一个外设使用。
3. 将 0x1 存储到 ADC0 的 FSUB_0 寄存器中，以便 ADC0 侦听计时器发布到通道 1 的事件。
4. 根据节 10.2.8 中的配置说明，将 ADC0 配置为从订阅者端口进行触发。
5. 配置并启用相应的 GPIO 引脚以监控输入电压事件

10.3 ADC12 寄存器

表 10-10 列出了 ADC12 寄存器的存储器映射寄存器。表 10-10 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 10-10. ADC12 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者配置寄存器。		转到
444h	FPUB_1	发布者配置寄存器。		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
808h	CLKCFG	ADC 时钟配置寄存器		转到
814h	STAT	状态寄存器		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT	转到
1058h	IMASK	中断屏蔽	GEN_EVENT	转到
1060h	RIS	原始中断状态	GEN_EVENT	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT	转到
1070h	ISET	中断设置	GEN_EVENT	转到
1078h	ICLR	中断清除	GEN_EVENT	转到
1080h	IIDX	中断索引	DMA_TRIG	转到
1088h	IMASK	中断屏蔽扩展	DMA_TRIG	转到
1090h	RIS	原始中断状态扩展	DMA_TRIG	转到
1098h	MIS	屏蔽中断状态扩展	DMA_TRIG	转到
10A0h	ISET	中断设置扩展	DMA_TRIG	转到
10A8h	ICLR	中断清除扩展	DMA_TRIG	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	CTL0	控制寄存器 0		转到
1104h	CTL1	控制寄存器 1		转到
1108h	CTL2	控制寄存器 2		转到
110Ch	CTL3	控制寄存器 3		转到
1110h	CLKFREQ	采样时钟频率范围寄存器		转到
1114h	SCOMP0	采样时间比较 0 寄存器		转到
1118h	SCOMP1	采样时间比较 1 寄存器		转到
111Ch	REFCFG	基准缓冲区配置寄存器		转到
1148h	WCLOW	窗口比较器低阈值寄存器		转到
1150h	WCHIGH	窗口比较器高阈值寄存器		转到
1160h	FIFODATA	FIFO 数据寄存器		转到
1170h	ASCRES	ASC 结果寄存器		转到
1180h + 公式	MEMCTL[y]	转换存储器控制寄存器		转到

表 10-10. ADC12 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
1280h + 公式	MEMRES[y]	存储器结果寄存器		转到
1340h	状态	状态寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 10-11 展示了适用于此部分中访问类型的代码。

表 10-11. ADC12 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
RH	R H	读取 由硬件置位或清除
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值
寄存器数组变量		
i、j、k、l、m、n		当这些变量用于寄存器名称、偏移或地址时，它们指的是寄存器数组的值，其中寄存器是一组重复寄存器的一部分。寄存器组构成分层结构，数组用公式表示。
y		当该变量用于寄存器名称、偏移或地址时，它指的是寄存器数组的值。

10.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 10-5 展示了 FSUB_0，表 10-12 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 10-5. FSUB_0

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

表 10-12. FSUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 255。

10.3.2 FPUB_1（偏移 = 444h）[复位 = 00000000h]

图 10-6 展示了 FPUB_1，表 10-13 中对此进行了介绍。

返回到汇总表。

发布者端口

图 10-6. FPUB_1

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

表 10-13. FPUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 255。

10.3.3 PWREN（偏移 = 800h）[复位 = 00000000h]

图 10-7 展示了 PWREN，表 10-14 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 10-7. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 10-14. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

10.3.4 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 10-8 展示了 RSTCTL，表 10-15 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 10-8. RSTCTL

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

表 10-15. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

10.3.5 CLKCFG（偏移 = 808h）[复位 = 00000000h]

图 10-9 展示了 CLKCFG，表 10-16 中对此进行了介绍。

返回到汇总表。

ADC 时钟配置

图 10-9. CLKCFG

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
保留		CCONSTOP	CCONRUN	保留		SAMPCLK	
R/W-0h		R/W-0h	R/W-0h	R/W-0h		R/W-0h	

表 10-16. CLKCFG 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 A9h = 允许对该寄存器进行写入访问的 KEY
23-6	保留	R/W	0h	
5	CCONSTOP	R/W	0h	CCONSTOP：当器件处于 STOP 模式时强制 SYSOSC 以基础频率运行，此时器件可用作 ADC 采样或转换时钟源。 0h = 在 STOP 模式期间，ADC 转换时钟源不会持续保持打开状态。 1h = 在 STOP 模式期间，ADC 转换时钟源持续保持打开状态。
4	CCONRUN	R/W	0h	CCONRUN：当器件处于 RUN 模式时强制 SYSOSC 以基础频率运行，此时器件可用作 ADC 采样或转换时钟源。 0h = 在 RUN 模式期间，ADC 转换时钟源不会持续保持打开状态。 1h = 在 RUN 模式期间，ADC 转换时钟源持续保持打开状态。
3-2	RESERVED	R/W	0h	
1-0	SAMPCLK	R/W	0h	ADC 采样时钟源选择。 0h = ULPCLK 是 ADC 采样时钟源。 1h = SYSOSC 是 ADC 采样时钟源。 2h = HFCLK 时钟是 ADC 采样时钟源。 注意：HFCLK 可能并非在所有器件上都可用。

10.3.6 STAT（偏移 = 814h）[复位 = 00000000h]

图 10-10 展示了 STAT，表 10-17 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 10-10. STAT

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

表 10-17. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

10.3.7 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 10-11 展示了 IIDX，表 10-18 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。0x0 表示无事件挂起。中断 1 是最高优先级，2 是次高优先级，4、8、⋯2^{31} 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 10-11. IIDX

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

表 10-18. IIDX 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R	0h	
9-0	STAT	R	0h	中断索引状态 00h = 没有位置位意味着没有挂起的中断请求 01h = MEMRESx 溢出中断 02h = 序列转换时间溢出中断 03h = 高阈值比较中断 04h = 低阈值比较中断 05h = 主序列范围比较器中断 6h = DMA 完成中断，在 DMA 传输完成时生成 07h = MEMRESx 下溢中断 9h = MEMRES0 数据载入中断 Ah = MEMRES1 数据载入中断 Bh = MEMRES2 数据载入中断 Ch = MEMRES3 数据载入中断 Dh = MEMRES4 数据载入中断 Eh = MEMRES5 数据载入中断 Fh = MEMRES6 数据载入中断 10h = MEMRES7 数据载入中断 11h = MEMRES8 数据载入中断 12h = MEMRES9 数据载入中断 13h = MEMRES10 数据载入中断 14h = MEMRES11 数据载入中断 15h = MEMRES12 数据载入中断 16h = MEMRES13 数据载入中断 17h = MEMRES14 数据载入中断 18h = MEMRES15 数据载入中断 19h = MEMRES16 数据载入中断 1Ah = MEMRES17 数据载入中断 1Bh = MEMRES18 数据载入中断 1Ch = MEMRES19 数据载入中断 1Dh = MEMRES20 数据载入中断 1Eh = MEMRES21 数据载入中断 1Fh = MEMRES22 数据载入中断 20h = MEMRES23 数据载入中断

10.3.8 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 10-12 展示了 IMASK，表 10-19 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 10-12. IMASK

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
ASCDONE	UVIFG	DMADONE	INIFG	LOWIFG	HIGHIFG	TOVIFG	OVIFG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 10-19. IMASK 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R/W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R/W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R/W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R/W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-19. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R/W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES19 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R/W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES18 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R/W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES17 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R/W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES16 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R/W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES15 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R/W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES14 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R/W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES13 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R/W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES12 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-19. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R/W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES11 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R/W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES10 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R/W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES9 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R/W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES8 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R/W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES7 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R/W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES6 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R/W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES5 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R/W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES4 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-19. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R/W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R/W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R/W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R/W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7	ASCDONE	R/W	0h	ASC 屏蔽完成原始中断标志 0h = 中断未挂起。 1h = 中断挂起。
6	UVIFG	R/W	0h	针对 MEMRESx 下溢的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
5	DMADONE	R/W	0h	DMADONE 的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
4	INIFG	R/W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R/W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

表 10-19. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
2	HIGHIFG	R/W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1	TOVIFG	R/W	0h	序列转换超时溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
0	OVIFG	R/W	0h	针对 MEMRESx 溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

10.3.9 RIS（偏移 = 1030h）[复位 = 00000000h]

图 10-13 展示了 RIS，表 10-20 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 10-13. RIS

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
ASCDONE	UVIFG	DMADONE	INIFG	LOWIFG	HIGHIFG	TOVIFG	OVIFG
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 10-20. RIS 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES23 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES22 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES21 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES20 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-20. RIS 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES19 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES18 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES17 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES16 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES15 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES14 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES13 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES12 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-20. RIS 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-20. RIS 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7	ASCDONE	R	0h	ASC 完成的原始中断标志 0h = 中断未挂起。 1h = 中断挂起。
6	UVIFG	R	0h	针对 MEMRESx 下溢的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
5	DMADONE	R	0h	DMADONE 的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
4	INIFG	R	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

表 10-20. RIS 字段说明 (continued)

位	字段	类型	复位	说明
2	HIGHIFG	R	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1	TOVIFG	R	0h	序列转换触发溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
0	OVIFG	R	0h	针对 MEMRESx 溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

10.3.10 MIS (偏移 = 1038h) [复位 = 00000000h]

图 10-14 展示了 MIS，表 10-21 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 10-14. MIS

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
ASCDONE	UVIFG	DMADONE	INIFG	LOWIFG	HIGHIFG	TOVIFG	OVIFG
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 10-21. MIS 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES23 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES22 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES21 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES20 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-21. MIS 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES19 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES18 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES17 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES16 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES15 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES14 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES13 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES12 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-21. MIS 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-21. MIS 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7	ASCDONE	R	0h	ASC 完成的屏蔽中断状态 0h = 中断未挂起。 1h = 中断挂起。
6	UVIFG	R	0h	针对 MEMRESx 下溢的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
5	DMADONE	R	0h	DMADONE 的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
4	INIFG	R	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

表 10-21. MIS 字段说明 (continued)

位	字段	类型	复位	说明
2	HIGHIFG	R	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1	TOVIFG	R	0h	序列转换超时溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
0	OVIFG	R	0h	针对 MEMRESx 溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

10.3.11 ISET（偏移 = 1040h）[复位 = 00000000h]

图 10-15 展示了 ISET，表 10-22 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 10-15. ISET

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
ASCDONE	UVIFG	DMADONE	INIFG	LOWIFG	HIGHIFG	TOVIFG	OVIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 10-22. ISET 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-22. ISET 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES19 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES18 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES17 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES16 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES15 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES14 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES13 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES12 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-22. ISET 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-22. ISET 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7	ASCDONE	W	0h	在 RIS 中设置 ASC 完成标志 0h = 中断未挂起。 1h = 中断挂起。
6	UVIFG	W	0h	针对 MEMRESx 下溢的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
5	DMADONE	W	0h	DMADONE 的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
4	INIFG	W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

表 10-22. ISET 字段说明 (continued)

位	字段	类型	复位	说明
2	HIGHIFG	W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1	TOVIFG	W	0h	序列转换超时溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
0	OVIFG	W	0h	针对 MEMRESx 溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

10.3.12 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 10-16 展示了 ICLR，表 10-23 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 10-16. ICLR

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
ASCDONE	UVIFG	DMADONE	INIFG	LOWIFG	HIGHIFG	TOVIFG	OVIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 10-23. ICLR 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-23. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES19 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES18 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES17 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES16 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES15 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES14 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES13 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES12 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-23. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-23. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7	ASCDONE	W	0h	清除 RIS 中的 ASC 完成标志 0h = 中断未挂起。 1h = 中断挂起。
6	UVIFG	W	0h	针对 MEMRESx 下溢的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
5	DMADONE	W	0h	DMADONE 的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
4	INIFG	W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

表 10-23. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
2	HIGHIFG	W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1	TOVIFG	W	0h	序列转换超时溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
0	OVIFG	W	0h	针对 MEMRESx 溢出的原始中断标志。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。

10.3.13 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 10-17 展示了 IIDX，表 10-24 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。0x0 表示无事件挂起。中断 1 是最高优先级，2 是次高优先级，4、8、⋯2^{31} 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 10-17. IIDX

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

表 10-24. IIDX 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R	0h	
9-0	STAT	R	0h	中断索引状态 00h = 没有位置位意味着没有挂起的中断请求 03h = 高阈值比较中断 04h = 低阈值比较中断 05h = 主序列范围比较器中断 9h = MEMRES0 数据载入中断

10.3.14 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 10-18 展示了 IMASK，表 10-25 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 10-18. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							MEMRESIFG0
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
保留			INIFG	LOWIFG	HIGHIFG	保留	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	

表 10-25. IMASK 字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	0h	
8	MEMRESIFG0	R/W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-5	RESERVED	R/W	0h	
4	INIFG	R/W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R/W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
2	HIGHIFG	R/W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1-0	保留	R/W	0h	

10.3.15 RIS（偏移 = 1060h）[复位 = 00000000h]

图 10-19 展示了 RIS，表 10-26 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 10-19. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							MEMRESIFG0
R-0h							R-0h
7	6	5	4	3	2	1	0
保留			INIFG	LOWIFG	HIGHIFG	保留	
R-0h			R-0h	R-0h	R-0h	R-0h	

表 10-26. RIS 字段说明

位	字段	类型	复位	说明
31-9	保留	R	0h	
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-5	RESERVED	R	0h	
4	INIFG	R	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
2	HIGHIFG	R	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1-0	RESERVED	R	0h	

10.3.16 MIS (偏移 = 1068h) [复位 = 00000000h]

图 10-20 展示了 MIS，表 10-27 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 10-20. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							MEMRESIFG0
R-0h							R-0h
7	6	5	4	3	2	1	0
保留			INIFG	LOWIFG	HIGHIFG	保留	
R-0h			R-0h	R-0h	R-0h	R-0h	

表 10-27. MIS 字段说明

位	字段	类型	复位	说明
31-9	保留	R	0h	
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-5	RESERVED	R	0h	
4	INIFG	R	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	R	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
2	HIGHIFG	R	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1-0	RESERVED	R	0h	

10.3.17 ISET (偏移 = 1070h) [复位 = 00000000h]

图 10-21 展示了 ISET，表 10-28 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 10-21. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留							MEMRESIFG0
W-0h							W-0h
7	6	5	4	3	2	1	0
保留			INIFG	LOWIFG	HIGHIFG	保留	
W-0h			W-0h	W-0h	W-0h	W-0h	

表 10-28. ISET 字段说明

位	字段	类型	复位	说明
31-9	保留	W	0h	
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-5	保留	W	0h	
4	INIFG	W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
2	HIGHIFG	W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1-0	RESERVED	W	0h	

10.3.18 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 10-22 展示了 ICLR，表 10-29 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 10-22. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留							MEMRESIFG0
W-0h							W-0h
7	6	5	4	3	2	1	0
保留			INIFG	LOWIFG	HIGHIFG	保留	
W-0h			W-0h	W-0h	W-0h	W-0h	

表 10-29. ICLR 字段说明

位	字段	类型	复位	说明
31-9	保留	W	0h	
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-5	保留	W	0h	
4	INIFG	W	0h	屏蔽 MIS_EX 寄存器中的 INIFG。 0h = 中断未挂起。 1h = 中断挂起。
3	LOWIFG	W	0h	MEMRESx 结果寄存器的原始中断标志低于 窗口比较器的 WCLOWx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
2	HIGHIFG	W	0h	MEMRESx 结果寄存器的原始中断标志高于 窗口比较器的 WCHIGHx 阈值。 读取 IIDX 时会将位设置为 0，或者当 ICLR_EX 中的相位应设置为 1 时， 会将位设置为 0。 0h = 中断未挂起。 1h = 中断挂起。
1-0	RESERVED	W	0h	

10.3.19 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 10-23 展示了 IIDX，表 10-30 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。0x0 表示无事件挂起。中断 1 是最高优先级，2 是次高优先级，4、8、⋯2^{31} 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 10-23. IIDX

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

表 10-30. IIDX 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R	0h	
9-0	STAT	R	0h	中断索引状态 00h = 没有位置位意味着没有挂起的中断请求 9h = MEMRES0 数据载入中断 Ah = MEMRES1 数据载入中断 Bh = MEMRES2 数据载入中断 Ch = MEMRES3 数据载入中断 Dh = MEMRES4 数据载入中断 Eh = MEMRES5 数据载入中断 Fh = MEMRES6 数据载入中断 10h = MEMRES7 数据载入中断 11h = MEMRES8 数据载入中断 12h = MEMRES9 数据载入中断 13h = MEMRES10 数据载入中断 14h = MEMRES11 数据载入中断 15h = MEMRES12 数据载入中断 16h = MEMRES13 数据载入中断 17h = MEMRES14 数据载入中断 18h = MEMRES15 数据载入中断 19h = MEMRES16 数据载入中断 1Ah = MEMRES17 数据载入中断 1Bh = MEMRES18 数据载入中断 1Ch = MEMRES19 数据载入中断 1Dh = MEMRES20 数据载入中断 1Eh = MEMRES21 数据载入中断 1Fh = MEMRES22 数据载入中断 20h = MEMRES23 数据载入中断

10.3.20 IMASK（偏移 = 1088h）[复位= 00000000h]

图 10-24 展示了 IMASK，表 10-31 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 10-24. IMASK

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 10-31. IMASK 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R/W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R/W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R/W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R/W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-31. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R/W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES19 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R/W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES18 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R/W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES17 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R/W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES16 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R/W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES15 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R/W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES14 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R/W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES13 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R/W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES12 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-31. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R/W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R/W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R/W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R/W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R/W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R/W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R/W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R/W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-31. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R/W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R/W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R/W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R/W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-0	保留	R/W	0h	

10.3.21 RIS（偏移 = 1090h）[复位 = 00000000h]

图 10-25 展示了 RIS，表 10-32 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 10-25. RIS

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

表 10-32. RIS 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-32. RIS 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES19 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES18 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES17 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES16 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES15 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES14 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES13 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES12 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-32. RIS 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-32. RIS 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-0	保留	R	0h	

10.3.22 MIS (偏移 = 1098h) [复位 = 00000000h]

图 10-26 展示了 MIS，表 10-33 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态扩展。这是 IMASK 和 RIS 寄存器的与运算。

图 10-26. MIS

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED							
R-0h							

表 10-33. MIS 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	R	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	R	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	R	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	R	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-33. MIS 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	R	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES19 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	R	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES18 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	R	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES17 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	R	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES16 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	R	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES15 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	R	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES14 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	R	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES13 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	R	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES12 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-33. MIS 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	R	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES11 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	R	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES10 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	R	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES9 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	R	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES8 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	R	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES7 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	R	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES6 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	R	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES5 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	R	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES4 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-33. MIS 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	R	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	R	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	R	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	R	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-0	保留	R	0h	

10.3.23 ISET（偏移 = 10A0h）[复位 = 00000000h]

图 10-27 展示了 ISET，表 10-34 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 10-27. ISET

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
ESERVED							
W-0h							

表 10-34. ISET 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-34. ISET 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES19 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES18 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES17 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES16 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES15 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES14 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES13 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES12 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-34. ISET 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-34. ISET 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-0	保留	W	0h	

10.3.24 ICLR (偏移 = 10A8h) [复位 = 00000000h]

图 10-28 展示了 ICLR，表 10-35 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 10-28. ICLR

31	30	29	28	27	26	25	24
MEMRESIFG23	MEMRESIFG22	MEMRESIFG21	MEMRESIFG20	MEMRESIFG19	MEMRESIFG18	MEMRESIFG17	MEMRESIFG16
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
MEMRESIFG15	MEMRESIFG14	MEMRESIFG13	MEMRESIFG12	MEMRESIFG11	MEMRESIFG10	MEMRESIFG9	MEMRESIFG8
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
MEMRESIFG7	MEMRESIFG6	MEMRESIFG5	MEMRESIFG4	MEMRESIFG3	MEMRESIFG2	MEMRESIFG1	MEMRESIFG0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
ESERVED							
W-0h							

表 10-35. ICLR 字段说明

位	字段	类型	复位	说明
31	MEMRESIFG23	W	0h	MEMRES23 的原始中断状态。 当 MEMRES23 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES23 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
30	MEMRESIFG22	W	0h	MEMRES22 的原始中断状态。 当 MEMRES22 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES22 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
29	MEMRESIFG21	W	0h	MEMRES21 的原始中断状态。 当 MEMRES21 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES21 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
28	MEMRESIFG20	W	0h	MEMRES20 的原始中断状态。 当 MEMRES20 被载入一个新的转换结果时，该位被设置为 1。 读取 MEMRES20 寄存器将清除该位，或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-35. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
27	MEMRESIFG19	W	0h	MEMRES19 的原始中断状态。 当 MEMRES19 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES19 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
26	MEMRESIFG18	W	0h	MEMRES18 的原始中断状态。 当 MEMRES18 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES18 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
25	MEMRESIFG17	W	0h	MEMRES17 的原始中断状态。 当 MEMRES17 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES17 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
24	MEMRESIFG16	W	0h	MEMRES16 的原始中断状态。 当 MEMRES16 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES16 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
23	MEMRESIFG15	W	0h	MEMRES15 的原始中断状态。 当 MEMRES15 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES15 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
22	MEMRESIFG14	W	0h	MEMRES14 的原始中断状态。 当 MEMRES14 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES14 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
21	MEMRESIFG13	W	0h	MEMRES13 的原始中断状态。 当 MEMRES13 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES13 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
20	MEMRESIFG12	W	0h	MEMRES12 的原始中断状态。 当 MEMRES12 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES12 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-35. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
19	MEMRESIFG11	W	0h	MEMRES11 的原始中断状态。 当 MEMRES11 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES11 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
18	MEMRESIFG10	W	0h	MEMRES10 的原始中断状态。 当 MEMRES10 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES10 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
17	MEMRESIFG9	W	0h	MEMRES9 的原始中断状态。 当 MEMRES9 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES9 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
16	MEMRESIFG8	W	0h	MEMRES8 的原始中断状态。 当 MEMRES8 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES8 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
15	MEMRESIFG7	W	0h	MEMRES7 的原始中断状态。 当 MEMRES7 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES7 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
14	MEMRESIFG6	W	0h	MEMRES6 的原始中断状态。 当 MEMRES6 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES6 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
13	MEMRESIFG5	W	0h	MEMRES5 的原始中断状态。 当 MEMRES5 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES5 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
12	MEMRESIFG4	W	0h	MEMRES4 的原始中断状态。 当 MEMRES4 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES4 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。

表 10-35. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
11	MEMRESIFG3	W	0h	MEMRES3 的原始中断状态。 当 MEMRES3 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES3 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
10	MEMRESIFG2	W	0h	MEMRES2 的原始中断状态。 当 MEMRES2 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES2 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
9	MEMRESIFG1	W	0h	MEMRES1 的原始中断状态。 当 MEMRES1 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES1 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
8	MEMRESIFG0	W	0h	MEMRES0 的原始中断状态。 当 MEMRES0 被载入一个新的转换结果时， 该位被设置为 1。 读取 MEMRES0 寄存器将清除该位， 或当 ICLR 中的相应位设置为 1 时将清除该位 0h = 没有新数据已准备就绪。 1h = 新数据已准备好被读取。
7-0	保留	W	0h	

10.3.25 EVT_MODE（偏移 = 10E0h）[复位 = 00000009h]

图 10-29 展示了 EVT_MODE，表 10-36 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线路

图 10-29. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED							
R-							
15	14	13	12	11	10	9	8
RESERVED							
R-							
7	6	5	4	3	2	1	0
RESERVED				EVT1_CFG		INT0_CFG	
R-				R-2h		R-1h	

表 10-36. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3-2	EVT1_CFG	R	2h	GEN_EVENT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

10.3.26 DESC（偏移 = 10FCh）[复位 = 26110010h]

图 10-30 展示了 DESC，表 10-37 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 10-30. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-2611h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-0h				R-0h				R-1h				R-0h			

表 10-37. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	2611h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数
7-4	MAJREV	R	1h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

10.3.27 CTL0（偏移 = 1100h）[复位 = 00000000h]

图 10-31 展示了 CTL0，表 10-38 中对此进行了介绍。

返回到汇总表。

控制寄存器 0

图 10-31. CTL0

31	30	29	28	27	26	25	24
保留					SCLKDIV		
R/W-0h					R/W-0h		
23	22	21	20	19	18	17	16
保留							PWRDN
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
保留							ENC
R/W-0h							RH/W-0h

表 10-38. CTL0 字段说明

位	字段	类型	复位	说明
31-27	保留	R/W	0h	
26-24	SCLKDIV	R/W	0h	采样时钟分频器 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 4 分频 3h = 对时钟源进行 8 分频 4h = 对时钟源进行 16 分频 5h = 对时钟源进行 24 分频 6h = 对时钟源进行 32 分频 7h = 对时钟源进行 48 分频
23-17	保留	R/W	0h	
16	PWRDN	R/W	0h	断电策略 0h = 如果没有挂起的触发器，则 ADC 在转换完成时断电 1h = 只要通过软件启用 ADC，ADC 就保持通电状态。
15-1	RESERVED	R/W	0h	
0	ENC	RH/W	0h	启用转换 0h = 禁用转换。ENC 从 ON 变为 OFF 将中止 MEMCTLx 边界上的单个或重复序列。当前转换将完成，结果将存储在相应的 MEMRESx 中。 1h = 启用转换。ADC 序列发生器等待有效触发（软件或硬件）。

10.3.28 CTL1（偏移 = 1104h）[复位 = 00000000h]

图 10-32 展示了 CTL1，表 10-39 中对此进行了介绍。

返回到汇总表。

控制寄存器 1

图 10-32. CTL1

31	30	29	28	27	26	25	24
保留	AVGD			保留	AVGN		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
23	22	21	20	19	18	17	16
保留			SAMPMODE	保留		CONSEQ	
R/W-0h			R/W-0h	R/W-0h		R/W-0h	
15	14	13	12	11	10	9	8
保留							SC
R/W-0h							RH/W-0h
7	6	5	4	3	2	1	0
保留							TRIGSRC
R/W-0h							R/W-0h

表 10-39. CTL1 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30-28	AVGD	R/W	0h	硬件平均器分母。累加值要除以的数字（这是一个移位）。注意，结果寄存器最长为 16 位，因此如果没有适当地移位，结果将被截断。 0h (R/W) = 不移位 1h (R/W) = 移动 1 位 2h (R/W) = 移动 2 位 3h (R/W) = 移动 3 位 4h (R/W) = 移动 4 位 5h (R/W) = 移动 5 位 6h (R/W) = 移动 6 位 7h (R/W) = 移动 7 位
27	保留	R/W	0h	
26-24	AVGN	R/W	0h	硬件平均器分子。为当前 MEMCTLx 选择要累加的转换数，然后将其除以 AVGD。结果将存储在 MEMRESx 中。 0h (R/W) = 禁用平均器 1h (R/W) = 在存储到 MEMRESx 寄存器中之前求 2 次转换的平均值 2h (R/W) = 在存储到 MEMRESx 寄存器中之前求 4 次转换的平均值 3h (R/W) = 在存储到 MEMRESx 寄存器中之前求 8 次转换的平均值 4h (R/W) = 在存储到 MEMRESx 寄存器中之前求 16 次转换的平均值 5h (R/W) = 在存储在 MEMRESx 寄存器中之前求 32 次转换的平均值 6h (R/W) = 在存储在 MEMRESx 寄存器中之前求 64 次转换的平均值 7h (R/W) = 在存储在 MEMRESx 寄存器中之前求 128 次转换的平均值
23-21	保留	R/W	0h	
20	SAMPMODE	R/W	0h	采样模式。该位选择采样信号源。 当选择 TRIGSRC 作为硬件事件触发器时，MANUAL 选项无效。 0h = 采样计时器高相位用作采样信号 1h = 软件触发器用作采样信号
19-18	保留	R/W	0h	

表 10-39. CTL1 字段说明 (continued)

位	字段	类型	复位	说明
17-16	CONSEQ	R/W	0h	转换序列模式 0h = STARTADD 指向的 MEMCTLx 中的 ADC 通道将被转换一次 1h = STARTADD 和 ENDADD 指向的 ADC 通道序列将被转换一次 2h = STARTADD 指向的 MEMCTLx 中的 ADC 通道将被重复转换 3h = STARTADD 和 ENDADD 指向的 ADC 通道序列将被重复转换
15-9	保留	R/W	0h	
8	SC	RH/W	0h	转换开始 0h = 当 SAMPMODE 设置为 MANUAL 时，清除该位将结束采样阶段，转换阶段将开始。 当 SAMPMODE 设为 AUTO 时，写入 0 不起作用。 1h = 当 SAMPMODE 设置为 MANUAL 时，设置该位将启动采样阶段。只要设置该位，采样阶段就会持续。 当 SAMPMODE 设置为 AUTO 时，设置该位将触发基于计时器的采样时间。
7-1	RESERVED	R/W	0h	
0	TRIGSRC	R/W	0h	采样触发源 0h = 软件触发 1h = 硬件事件触发

10.3.29 CTL2（偏移 = 1108h）[复位 = 00000000h]

图 10-33 展示了 CTL2，表 10-40 中对此进行了介绍。

返回到汇总表。

控制寄存器 2

图 10-33. CTL2

31	30	29	28	27	26	25	24
保留			ENDADD				
R/W-0h			R/W-0h				
23	22	21	20	19	18	17	16
保留			STARTADD				
R/W-0h			R/W-0h				
15	14	13	12	11	10	9	8
SAMPCNT					FIFOEN	保留	DMAEN
R/W-0h					R/W-0h	R/W-0h	RH/W-0h
7	6	5	4	3	2	1	0
保留					RES		DF
R/W-0h					R/W-0h		R/W-0h

表 10-40. CTL2 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28-24	ENDADD	R/W	0h	序列结束地址。这些位选择哪个 MEMCTLx 是序列模式的最后一个。ENDADD 的值为 0x00 至 0x17，对应于 MEMRES0 至 MEMRES23。 00h = MEMCTL0 被选为序列的结束地址。 01h = MEMCTL1 被选为序列的结束地址。 02h = MEMCTL2 被选为序列的结束地址。 03h = MEMCTL3 被选为序列的结束地址。 04h = MEMCTL4 被选为序列的结束地址。 05h = MEMCTL5 被选为序列的结束地址。 06h = MEMCTL6 被选为序列的结束地址。 07h = MEMCTL7 被选为序列的结束地址。 08h = MEMCTL8 被选为序列的结束地址。 09h = MEMCTL9 被选为序列的结束地址。 0Ah = MEMCTL10 被选为序列的结束地址。 0Bh = MEMCTL11 被选为序列的结束地址。 0Ch = MEMCTL12 被选为序列的结束地址。 0Dh = MEMCTL13 被选为序列的结束地址。 0Eh = MEMCTL14 被选为序列的结束地址。 0Fh = MEMCTL15 被选为序列的结束地址。 10h = MEMCTL16 被选为序列的结束地址。 11h = MEMCTL17 被选为序列的结束地址。 12h = MEMCTL18 被选为序列的结束地址。 13h = MEMCTL19 被选为序列的结束地址。 14h = MEMCTL20 被选为序列的结束地址。 15h = MEMCTL21 被选为序列的结束地址。 16h = MEMCTL22 被选为序列的结束地址。 17h = MEMCTL23 被选为序列的结束地址。
23-21	保留	R/W	0h	

表 10-40. CTL2 字段说明 (continued)

位	字段	类型	复位	说明
20-16	STARTADD	R/W	0h	序列发生器起始地址。这些位选择哪个 MEMCTLx 用于单次转换或作为序列模式的第一个 MEMCTL。 STARTADD 的值为 0x00 至 0x17，对应于 MEMRES0 至 MEMRES23。 00h = MEMCTL0 被选为序列的起始地址或用于单次转换。 01h = MEMCTL1 被选为序列的起始地址或用于单次转换。 02h = MEMCTL2 被选为序列的起始地址或用于单次转换。 03h = MEMCTL3 被选为序列的起始地址或用于单次转换。 04h = MEMCTL4 被选为序列的起始地址或用于单次转换。 05h = MEMCTL5 被选为序列的起始地址或用于单次转换。 06h = MEMCTL6 被选为序列的起始地址或用于单次转换。 07h = MEMCTL7 被选为序列的起始地址或用于单次转换。 08h = MEMCTL8 被选为序列的起始地址或用于单次转换。 09h = MEMCTL9 被选为序列的起始地址或用于单次转换。 0Ah = MEMCTL10 被选为序列的起始地址或用于单次转换。 0Bh = MEMCTL11 被选为序列的起始地址或用于单次转换。 0Ch = MEMCTL12 被选为序列的起始地址或用于单次转换。 0Dh = MEMCTL13 被选为序列的起始地址或用于单次转换。 0Eh = MEMCTL14 被选为序列的起始地址或用于单次转换。 0Fh = MEMCTL15 被选为序列的起始地址或用于单次转换。 10h = MEMCTL16 被选为序列的起始地址或用于单次转换。 11h = MEMCTL17 被选为序列的起始地址或用于单次转换。 12h = MEMCTL18 被选为序列的起始地址或用于单次转换。 13h = MEMCTL19 被选为序列的起始地址或用于单次转换。 14h = MEMCTL20 被选为序列的起始地址或用于单次转换。 15h = MEMCTL21 被选为序列的起始地址或用于单次转换。 16h = MEMCTL22 被选为序列的起始地址或用于单次转换。 17h = MEMCTL23 被选为序列的起始地址或用于单次转换。
15-11	SAMPCNT	R/W	0h	DMA 触发器上要传输的 ADC 转换样本数 0h = 最小值 18h = 最大值
10	FIFOEN	R/W	0h	启用基于 FIFO 的操作 0h = 禁用 1h = 启用
9	保留	R/W	0h	
8	DMAEN	RH/W	0h	为数据传输启用 DMA 触发。 注意：DMAEN 位在数据传输结束时根据 DMA 完成信号被硬件清除。软件必须为 ADC 重新启用 DMAEN 位以生成 DMA 触发。 0h (R/W) = 未启用 DMA 触发 1h (R/W) = 启用 DMA 触发
7-3	RESERVED	R/W	0h	
2-1	RES	R/W	0h	分辨率。这些位定义 ADC 转换结果的分辨率。 注意：值 3 默认为 12 位分辨率。 0h = 12 位分辨率 1h = 10 位分辨率 2h = 8 位分辨率
0	DF	R/W	0h	数据读回格式。数据始终以二进制无符号格式存储。 0h = 数字结果以二进制无符号格式读取。 1h = 数字结果以有符号二进制格式读取。（二进制补码），左对齐。

10.3.30 CTL3（偏移 = 110Ch）[复位 = 00000000h]

图 10-34 展示了 CTL3，表 10-41 中对此进行了介绍。

返回到汇总表。

控制寄存器 3。该寄存器用于配置 ADC 进行临时单次转换。

图 10-34. CTL3

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留		ASCVRSEL		保留			ASCSTIME
R/W-0h		R/W-0h		R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
保留				ASCCHSEL			
R/W-0h				R/W-0h			

表 10-41. CTL3 字段说明

位	字段	类型	复位	说明
31-14	保留	R/W	0h	
13-12	ASCVRSEL	R/W	0h	为 ASC 操作选择电压基准。选择外部基准选项时，VEREFM 必须连接到电路板地。 注意：写入值 0x3 默认为 INTREF。 0h = VDDA 基准。 1h = EXTREF 引脚基准。 2h = 内部基准。
11-9	保留	R/W	0h	
8	ASCSTIME	R/W	0h	ASC 采样时间比较值选择。这用于在 SCOMP0 和 SCOMP1 寄存器之间进行选择以进行 ASC 操作。 0h = 选择 SCOMP0 1h = 选择 SCOMP1
7-5	RESERVED	R/W	0h	

表 10-41. CTL3 字段说明 (continued)

位	字段	类型	复位	说明
4-0	ASCCHSEL	R/W	0h	ASC 通道选择 00h = 选择通道 0 01h = 选择通道 1 02h = 选择通道 2 03h = 选择通道 3 04h = 选择通道 4 05h = 选择通道 5 06h = 选择通道 6 07h = 选择通道 7 08h = 选择通道 8 09h = 选择通道 9 0Ah = 选择通道 10 0Bh = 选择通道 11 0Ch = 选择通道 12 0Dh = 选择通道 13 0Eh = 选择通道 14 0Fh = 选择通道 15 10h = 选择通道 16 11h = 选择通道 17 12h = 选择通道 18 13h = 选择通道 19 14h = 选择通道 20 15h = 选择通道 21 16h = 选择通道 22 17h = 选择通道 23 18h = 选择通道 24 19h = 选择通道 25 1Ah = 选择通道 26 1Bh = 选择通道 27 1Ch = 选择通道 28 1Dh = 选择通道 29 1Eh = 选择通道 30 1Fh = 选择通道 31

10.3.31 CLKFREQ (偏移 = 1110h) [复位 = 00000000h]

图 10-35 展示了 CLKFREQ，表 10-42 中对此进行了介绍。

返回到汇总表。

采样时钟频率范围寄存器。

图 10-35. CLKFREQ

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

表 10-42. CLKFREQ 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	FRANGE	R/W	0h	频率范围。 0h = 1MHz 至 4MHz 1h = >4MHz 至 8MHz 2h = >8MHz 至 16MHz 3h = >16MHz 至 20MHz 4h = >20MHz 至 24MHz 5h = >24MHz 至 32MHz 6h = >32MHz 至 40MHz 7h = >40MHz 至 48MHz

10.3.32 SCOMP0（偏移 = 1114h）[复位 = 00000000h]

图 10-36 展示了 SCOMP0，表 10-43 中对此进行了介绍。

返回到汇总表。

采样时间比较 0 寄存器。指定采样时间，以 ADC 采样时钟周期数表示。CTL0.ENC 必须为 0 才能对该寄存器进行写入操作。

图 10-36. SCOMP0

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

表 10-43. SCOMP0 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	
9-0	值	R/W	0h	指定采样时钟数。 当 VAL = 0 或 1 时，采样时钟数 = 采样时钟分频值。 当 VAL > 1 时，采样时钟数 = VAL x 采样时钟分频值。 注意：采样时钟分频值不是写入 SCLKDIV 的值，而是实际分频值（SCLKDIV = 2 表示分频值为 4）。 示例：VAL = 4、SCLKDIV = 3 表示 32 个采样时钟周期。

10.3.33 SCOMP1（偏移 = 1118h）[复位 = 00000000h]

图 10-37 展示了 SCOMP1，表 10-44 中对此进行了介绍。

返回到汇总表。

采样时间比较 1 寄存器。指定采样时间，以 ADC 采样时钟周期数表示。CTL0.ENC 必须为 0 才能对该寄存器进行写入操作。

图 10-37. SCOMP1

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

表 10-44. SCOMP1 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	
9-0	值	R/W	0h	指定采样时钟数。 当 VAL = 0 或 1 时，采样时钟数 = 采样时钟分频值。 当 VAL > 1 时，采样时钟数 = VAL x 采样时钟分频值。 注意：采样时钟分频值不是写入 SCLKDIV 的值，而是实际分频值（SCLKDIV = 2 表示分频值为 4）。 示例：VAL = 4、SCLKDIV = 3 表示 32 个采样时钟周期。

10.3.34 REFCFG（偏移 = 111Ch）[复位 = 00000000h]

图 10-38 展示了 REFCFG，表 10-45 中对此进行了介绍。

返回到汇总表。

基准缓冲区配置寄存器

图 10-38. REFCFG

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
保留			IBPROG		保留	REFVSEL	REFEN
R/W-0h			R/W-0h		R/W-0h	R/W-0h	R/W-0h

表 10-45. REFCFG 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	0h	
4-3	IBPROG	R/W	0h	配置基准缓冲器偏置电流输出值 0h = 1uA 1h = 0.5uA 2h = 2uA 3h = 0.67uA
2	RESERVED	R/W	0h	
1	REFVSEL	R/W	0h	配置基准缓冲器输出电压 0h = 基准缓冲器生成 2.5V 输出 1h = 基准缓冲器生成 1.4V 输出
0	REFEN	R/W	0h	启用基准缓冲器 0h = 禁用 1h = 启用

10.3.35 WCLOW（偏移 = 1148h）[复位 = 00000000h]

图 10-39 展示了 WCLOW，表 10-46 中对此进行了介绍。

返回到汇总表。

窗口比较器低阈值寄存器。
用于写入和读取 WCLOW 的数据格式取决于 CTL2 寄存器中 DF 位的值。
CTL0.ENC 必须为 0 才能对该寄存器进行写入操作。
注意：更改 ADC 数据格式或分辨率不会复位 WCLOW。

图 10-39. WCLOW

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

表 10-46. WCLOW 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	数据	R/W	0h	如果 DF = 0，则必须使用无符号二进制格式。 基于分辨率的值必须与左侧的 MSB 右对齐。 对于 10 位和 8 位分辨率，未使用的位必须为 0。 如果 DF = 1，则必须使用二进制补码格式。 基于分辨率的值必须与右侧的 LSB 左对齐。 对于 10 位和 8 位分辨率，未使用的位必须为 0。

10.3.36 WCHIGH（偏移 = 1150h）[复位 = 00000000h]

图 10-40 展示了 WCHIGH，表 10-47 中对此进行了介绍。

返回到汇总表。

窗口比较器高阈值寄存器。
用于写入和读取 WCHIGH 的数据格式取决于 CTL2 寄存器中 DF 位的值。
CTL0.ENC 必须为 0 才能对该寄存器进行写入操作。
注意：更改 ADC 数据格式或分辨率不会复位 WCHIGH。

图 10-40. WCHIGH

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

表 10-47. WCHIGH 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	数据	R/W	0h	如果 DF = 0，则必须使用无符号二进制格式。 阈值必须与左侧的 MSB 右对齐。 对于 10 位和 8 位分辨率，未使用的位必须为 0。 如果 DF = 1，则必须使用二进制补码格式。 基于分辨率的值必须与右侧的 LSB 左对齐。 对于 10 位和 8 位分辨率，未使用的位必须为 0。

10.3.37 FIFODATA (偏移 = 1160h) [复位 = 00000000h]

图 10-41 展示了 FIFODATA，表 10-48 中对此进行了介绍。

返回到汇总表。

FIFO 数据寄存器。这是一个用于从 FIFO 中进行读取的虚拟寄存器。

图 10-41. FIFODATA

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

表 10-48. FIFODATA 字段说明

位	字段	类型	复位	说明
31-0	数据	R	0h	读取该数据字段将返回 FIFO 中的 ADC 采样值。

10.3.38 ASCRES（偏移 = 1170h）[复位 = 00000000h]

图 10-42 展示了 ASCRES，表 10-49 中对此进行了介绍。

返回到汇总表。

ASC 结果寄存器

图 10-42. ASCRES

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

表 10-49. ASCRES 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15-0	数据	R	0h	ADC 临时单次转换的结果。 如果 DF = 0，则使用无符号二进制格式： 转换结果右对齐。在 10 位和 8 位模式中，未使用的 MSB 位强制为 0。 如果 DF = 1，则使用二进制补码格式： 转换结果左对齐。在 10 位和 8 位模式中，未使用的 LSB 位强制为 0。 该数据以右对齐格式被存储并在回读期间被转换为左对齐的二进制补码格式。

10.3.39 MEMCTL[y]（偏移 = 1180h + 公式）[复位 = 00000000h]

图 10-43 展示了 MEMCTL[y]，表 10-50 中对此进行了介绍。

返回到汇总表。

转换存储器控制寄存器。
CTL0.ENC 必须为 0 才能对该寄存器进行写入操作。

偏移 = 1180h + (y * 4h)；其中 y = 0h 至 17h

图 10-43. MEMCTL[y]

31	30	29	28	27	26	25	24
保留			WINCOMP	保留			TRIG
R/W-0h			R/W-0h	R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
保留			BCSEN	保留			AVGEN
R/W-0h			R/W-0h	R/W-0h			R/W-0h
15	14	13	12	11	10	9	8
保留			STIME	保留		VRSEL	
R/W-0h			R/W-0h	R/W-0h		R/W-0h	
7	6	5	4	3	2	1	0
保留			CHANSEL				
R/W-0h			R/W-0h				

表 10-50. MEMCTL[y] 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	WINCOMP	R/W	0h	启用窗口比较器。 0h = 禁用 1h = 启用
27-25	保留	R/W	0h	
24	TRIG	R/W	0h	触发策略。指示是否需要触发器来步进到序列中的下一个 MEMCTL，或者在重复单通道转换的情况下执行下一个转换。 0h = 下一个转换是自动的 1h = 下一个转换需要触发器
23-21	保留	R/W	0h	
20	BCSEN	R/W	0h	启用烧毁电流源。 0h = 禁用 1h = 启用
19-17	保留	R/W	0h	
16	AVGEN	R/W	0h	启用硬件均值计算。 0h (R/W) = 禁用均值计算。 1h = 启用均值计算。
15-13	RESERVED	R/W	0h	
12	STIME	R/W	0h	在 SCOMP0 和 SCOMP1 之间选择采样计时器周期的来源。 0h = 选择 SCOMP0 1h = 选择 SCOMP1
11-10	RESERVED	R/W	0h	

表 10-50. MEMCTL[y] 字段说明 (continued)

位	字段	类型	复位	说明
9-8	VRSEL	R/W	0h	电压基准选择。选择外部基准选项时，VEREFM 必须连接到电路板地。 注意：写入值 0x3 默认为 INTREF。 0h = VDDA 基准 1h = 来自引脚的外部基准 2h = 内部基准
7-5	RESERVED	R/W	0h	
4-0	CHANSEL	R/W	0h	输入通道选择。 00h = 选择通道 0 01h = 选择通道 1 02h = 选择通道 2 03h = 选择通道 3 04h = 选择通道 4 05h = 选择通道 5 06h = 选择通道 6 07h = 选择通道 7 08h = 选择通道 8 09h = 选择通道 9 0Ah = 选择通道 10 0Bh = 选择通道 11 0Ch = 选择通道 12 0Dh = 选择通道 13 0Eh = 选择通道 14 0Fh = 选择通道 15 10h = 选择通道 16 11h = 选择通道 17 12h = 选择通道 18 13h = 选择通道 19 14h = 选择通道 20 15h = 选择通道 21 16h = 选择通道 22 17h = 选择通道 23 18h = 选择通道 24 19h = 选择通道 25 1Ah = 选择通道 26 1Bh = 选择通道 27 1Ch = 选择通道 28 1Dh = 选择通道 29 1Eh = 选择通道 30 1Fh = 选择通道 31

10.3.40 MEMRES[y]（偏移 = 1280h + 公式）[复位 = 00000000h]

图 10-44 展示了 MEMRES[y]，表 10-51 中对此进行了介绍。

返回到汇总表。

存储器结果寄存器

偏移 = 1280h + (y * 4h)；其中 y = 0h 至 17h

图 10-44. MEMRES[y]

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

表 10-51. MEMRES[y] 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15-0	数据	R	0h	MEMRES 结果寄存器。 如果 DF = 0，则使用无符号二进制格式： 转换结果右对齐。在 10 位和 8 位模式中，未使用的 MSB 位强制为 0。 如果 DF = 1，则使用二进制补码格式： 转换结果左对齐。在 10 位和 8 位模式中，未使用的 LSB 位强制为 0。 该数据以右对齐格式被存储并在回读期间被转换为左对齐的二进制补码格式。

10.3.41 STATUS (偏移 = 1340h) [复位 = 00000000h]

图 10-45 展示了 STATUS，表 10-52 中对此进行了介绍。

返回到汇总表。

状态寄存器

图 10-45. STATUS

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

表 10-52. STATUS 字段说明

位	字段	类型	复位	说明
31-3	保留	R	0h	
2	ASCACT	R	0h	ASC 运行 0h = 空闲或完成 1h = ASC 运行
1	REFBUFRDY	R	0h	表示基准缓冲器已加电并就绪。 0h = 未就绪 1h = 就绪
0	忙	R	0h	忙。该位表示正在进行有源 ADC 采样或转换操作。 0h = 没有正在进行的 ADC 采样或转换。 1h = 正在进行 ADC 采样或转换。

This page intentionally left blank.

章节 11
COMP

比较器模块 (COMP) 是具有通用比较器功能的模拟电压比较器。本章介绍比较器模块的运行情况。

11.1 比较器概述

比较器模块可用于电源电压监控和外部模拟信号监控。

比较器的特性包括：

- 快速和超低功耗工作模式
- 反向和非反向引脚输入复用器
- 反相和同相端子短路和交换功能
- 用于比较器输出并可通过软件选择的模拟滤波器
- 可编程迟滞
- 来自内部电压基准 VDD 或来自外部基准引脚的基准电压
- 可配置基准电压发生器，来自具有两个输入代码的集成式 8 位 DAC
- 输出连接到事件系统
- 窗口比较器模式
- 支持低功耗运行的中断驱动测量系统

图 11-1 展示了比较器方框图。

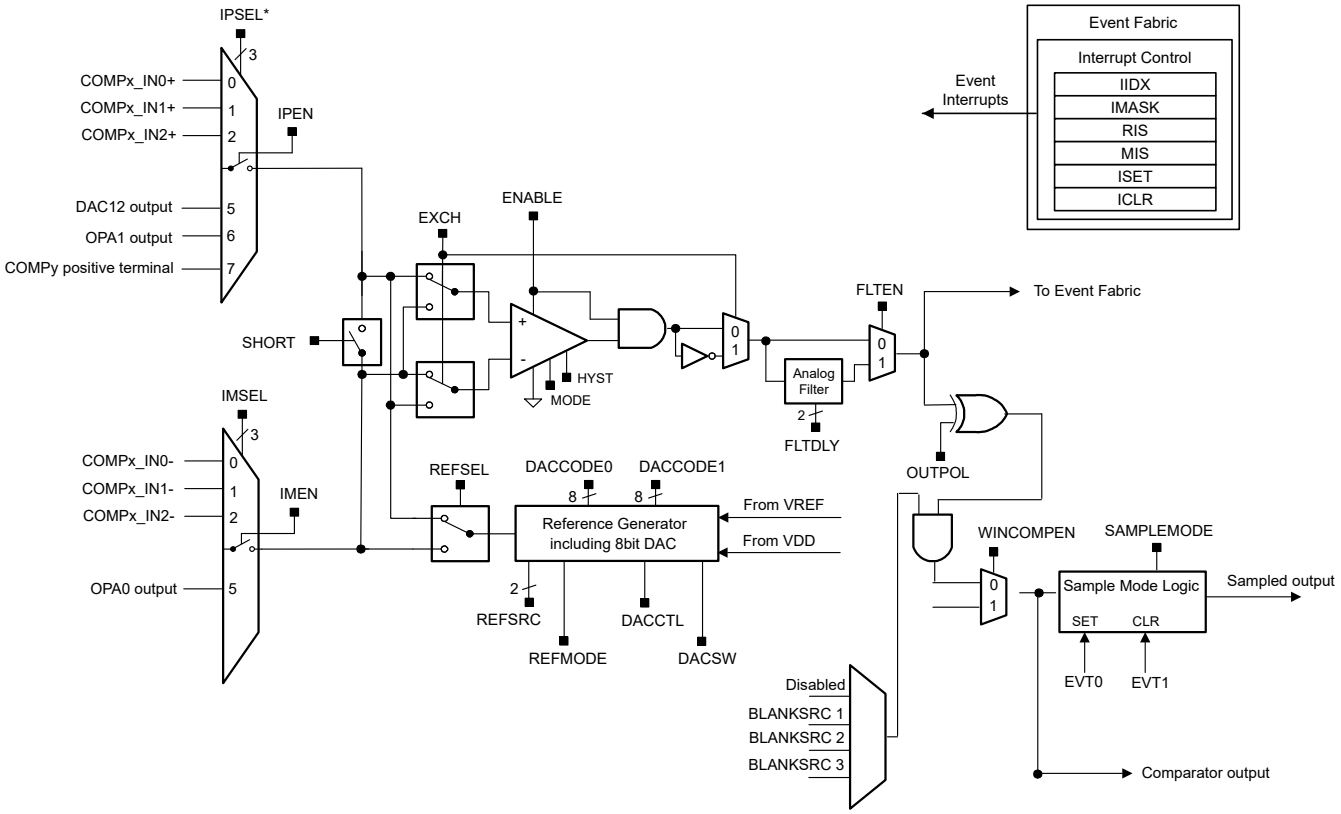


图 11-1. 比较器块图

11.2 比较器运行

比较器模块由用户软件配置。以下各节将讨论比较器的设置和运行。

11.2.1 比较器配置

比较器会比较正 (+) 和负 (–) 输入端的模拟电压。如果 + 端子比 - 端子更加正向，则比较器输出为高电平。比较器输出的极性可以通过 COMPx.CTL1 寄存器中的 OUTPOL 位进行配置。可以使用 COMPx.CTL1 寄存器中的 ENABLE 位打开或关闭比较器。在不使用该比较器时，应该将其关闭以减小电流消耗。当比较器关闭时，OUT 在 OUTPOL 位设置为 0 时为低电平，而 OUT 在 OUTPOL 位设置为 1 时为高电平。

可以使用 COMPx.CTL1 寄存器中的 MODE 位将比较器配置为快速或超低功耗模式。MODE 位的默认值为 0，可将比较器配置为快速模式。当 MODE 位设置为 1 时，比较器配置为超低功耗模式。在快速模式下，比较器消耗的电流较高，但响应时间较短。在超低功耗模式下，比较器消耗非常低的电流，但响应时间较慢。为了优化应用的电流消耗，应选择满足比较器速度要求的电源模式 (有关比较器传播延迟，请参阅器件特定数据表) 。

比较器的时钟控制由系统控制器 (SYSCTL) 管理，SYSCTL 知道比较器模块是否被启用，还知道它是处于超低功耗模式还是快速模式。用户需要确保为不同的比较器工作模式选择正确的总线时钟：

- 对于超低功耗模式比较器，总线时钟可以是 LFCLK 或任何高速时钟。
- 对于快速模式比较器，总线时钟不能是 LFCLK，如果在快速模式下使能比较器并且总线时钟是 LFCLK，系统将在 SYSCTL 中产生时钟错误中断。

11.2.2 比较器通道选择

正极端子上的比较器通道通过 IPSEL 位选择，并通过寄存器 COMPx.CTL0 中的 IPEN 位启用。类似地，负端子上的比较器通道通过 IMSEL 位选择并通过 IMEN 位启用。IPSEL 和 IMSEL 位可用于从器件引脚或内部模拟模块选择比较器通道输入。正负端子上比较器输入通道多路复用器中的开关通过先断后合布置实现，以便在通道选择发生变化时更大限度地减少串扰。

IPSEL 和 IMSEL 位允许：

- 将外部信号连接到比较器的正负端子
- 将内部模拟信号 (如 DAC 或 OPA) 连接到比较器的正负端子
- 在窗口比较模式中，将一个比较器正极端子信号连接到另一个比较器正极端子

COMPx.CTL1 寄存器中的 EXCH 位控制输入多路复用器，交换比较器正负端子的输入信号。此外，交换比较器端子后，来自比较器的输出信号也会反相。这就使得用户可以确定或者补偿比较器输入的偏移电压。

备注

比较器输入连接

有关输入通道选择配置，请参阅器件数据表 COMP 部分。

当比较器打开时，其输入端子应该连接到一个信号、电源或者接地。否则，悬空电平会产生意外的中断并增加流耗。

比较器配置

比较器运行期间不得更改其配置，但寄存器 COMPx.CTL0 中 IPSEL、IMSEL 和 EXCH 设置发生的更改除外。要更改比较器的配置，必须首先禁用它，根据需要重新配置，然后重新启用。

11.2.3 比较器输出

比较器输出可从 COMPx.STAT 寄存器的 OUT 位读取。输出也在器件引脚上发出。为了使比较器输出与器件引脚成功连接，需要软件在 IOMUX 模块中配置相应的器件引脚。比较器输出也进入事件接口以生成 CPU 中断事件和通用发布者事件，用于与其他模块进行内部连接。

11.2.4 输出滤波器

比较器的输出可以使用内部的滤波器，也可以不使用。输出滤波只能在比较器处于快速模式 (COMPx.CTL1.MODE = 0) 时启用。如果设置了 COMPx.CTL1 寄存器中的 FLTEN 位，输出将通过片上模拟滤波器进行滤波。可以使用 COMPx.CTL1 寄存器中的 FLTDLY 位通过四个不同的步骤调整滤波器的延迟（请参阅表 11-1）。

表 11-1. 典型输出滤波器延迟设置

COMPx.CTL1 中的 FLTDLY 位	典型滤波器延迟
0	70ns
1	500ns
2	1200ns
3	2700ns

滤波器延迟设置是对特定于器件的数据表中概述的传播延迟的补充。

如果输入端子上的电压差很小，则比较器输出会振荡（请参阅图 11-2）。内部和外部的寄生效应以及信号线、电源线和系统其他器件之上及之间产生的交叉耦合会导致这种行为。比较器输出振荡会降低比较结果的精度和分辨率。选择输出滤波器可以减少与比较器振荡相关的误差。

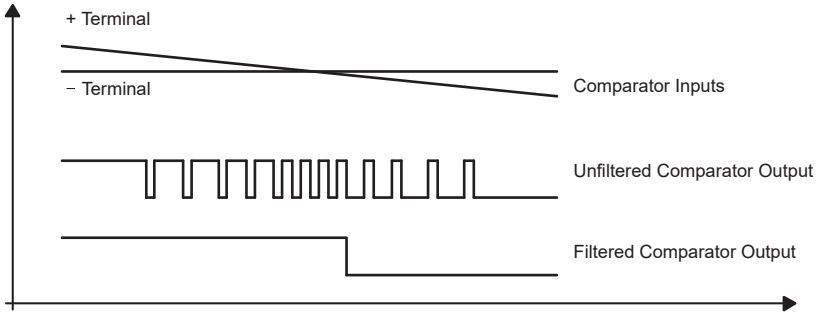


图 11-2. 比较器输出端的模拟滤波器响应

备注

仅当比较器在快速模式下运行时，输出滤波器才对其起作用。

11.2.5 采样输出模式

采样模式是比较器的一种操作方法，它比较离散时间间隔或样本的输入信号，生成仅在采样点处变化的输出信号。CTL2 寄存器中的 SAMPMODE 位启用或禁用采样模式。

为了定义采样窗口，计时器会生成 EVT0 和 EVT1 两个事件。通过使用这些事件，采样窗口可以与噪声较小的相位对齐。EVT0 设置采样窗口，而 EVT1 将其清除。

在采样模式下，比较器的输出仅在采样窗口为高电平时被捕获，其他时间不捕获输出。捕获的输出用于中断和事件生成。

通过使用采样模式，比较器可以降低输入信号中噪声的影响，从而实现更加准确可靠的比较。使用 EVT0 和 EVT1 定义采样窗口可以更灵活地将窗口与输入信号噪声最小的相位对齐，从而进一步提高比较的准确性。

11.2.6 消隐模式

由于噪声或其他干扰，输入信号可能会在阈值电压附近波动，从而导致输出快速且不稳定地切换。这可能导致误触发。比较器消隐模式在输入信号超过阈值电压的时间和输出改变状态的时间之间引入了短暂的延迟。

在该延迟期间，比较器忽略输入信号的任何进一步变化并保持其当前输出状态，这主要用于某些用例，例如电机驱动、电源控制等中的计时器指示的 PWM 波形。

寄存器 CTL2 中的控制位 BLANKSRC 可用于配置禁用或者哪个源应该在消隐模式下工作。请参阅器件的数据表，了解支持的特定消隐源。

备注

计时器用作比较器消隐模式的来源，在计时器输出和比较器输出之间通常会有一个 TIMCLK 周期延迟。

11.2.7 基准电压发生器

图 11-3 展示了比较器基准电压发生器的方框图。比较器基准电压发生器包含一个 8 位 DAC 以及一些配置选项。COMPx.CTL2 寄存器中的 REFSRC 位用于选择比较器的基准源。

- 当 REFSRC = 0 时，将禁用基准电压发生器，不能将基准电压发生器用于运行比较器。
- 当 REFSRC = 1 时，将选择模拟电源 VDDA 作为 DAC 的基准输入，而 DAC 输出将用作比较器的基准电压。
- 当 REFSRC = 2 时，将选择内部基准模块输出的 VREF 作为 DAC 的基准输入，而 DAC 输出将用作比较器的基准电压。
- 当 REFSRC = 3 时，内部基准模块输出的 VREF 将直接用作比较器的基准电压，并且 DAC 关闭。

可以使用 COMPx.CTL2 寄存器的 REFSEL 位将基准电压发生器的输出施加到比较器的正极端子或负极端子上。如果比较器的两个输入端子都使用外部信号，请关闭内部基准电压发生器，以便降低电流消耗。

如果使用 COMPx.CTL2 寄存器的 REFSEL 位将基准电压发生器的输出施加到比较器端子上，并且还使用 COMPx.CTL0 寄存器中的 IPSEL/IPEN 或 IMSEL/IMEN 位在同一端子上选择比较器通道（来自器件引脚或来自内部模拟模块），那么选择的比较器通道具有更高优先级。

DAC8 输出也通过 OPAx 连接到内部模拟 OPA 模块（请参阅图 12-1）。

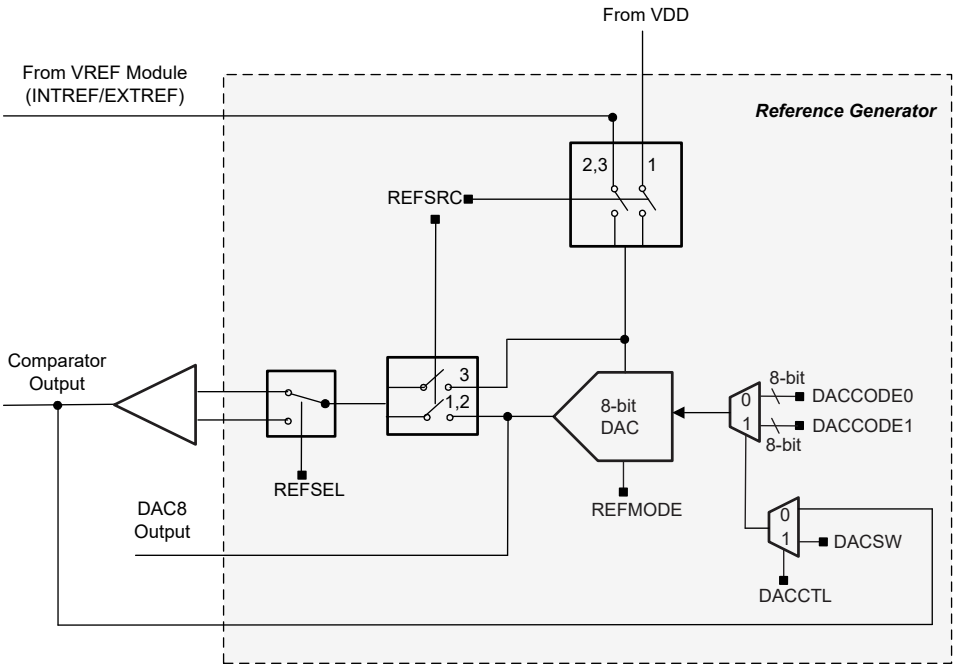


图 11-3. 基准电压发生器方框图

集成 8 位 DAC

可以通过 COMPx.CTL3 寄存器中的 DACCODE0 或 DACCODE1 位提供 8 位 DAC 输入代码。COMPx.CTL2 寄存器中的 DACCTL 位决定了是比较器输出值还是 COMPx.CTL2 寄存器中的软件控制位 DACSW 将选择 DACCODE0 或 DACCODE1 位作为 DAC 的输入。

- 如果 DACCTL 为 0，则比较器输出值将选择 DACCODE0 或 DACCODE1。如果比较器输出值为 0，DACCODE0 将作为 DAC 的输入。如果比较器输出值为 1，DACCODE1 将作为 DAC 的输入。

- 如果 DACCTL 为 1，则软件编程的 DACSW 位值将选择 DACCODE0 或 DACCODE1。如果 DACSW 为 0，DACCODE0 将作为 DAC 的输入。如果 DACSW 为 1，DACCODE1 将作为 DAC 的输入。借助这种配置，无需使用外部元件即可为比较器生成所需的迟滞电平。
- COMPx.CTL2 寄存器中的 REFMODE 位决定了比较器请求在快速模式还是超低功耗模式下运行内部 VREF，并确定 8 位 DAC 的工作模式。如果 REFMODE 位为 0，则会请求内部 VREF 在快速模式下运行，并且比较器中的 8 位 DAC 也会配置为快速模式。如果 REFMODE 位为 1，则会请求内部 VREF 在超低功耗模式下运行，并且比较器中的 8 位 DAC 也配置为超低功耗模式。在快速模式下运行可提供更高的精度，但会消耗更高的电流，而在超低功耗模式下运行则会消耗更低的电流，但基准电压精度会降低。请参阅器件特定的数据表中的比较器电气规格，了解详细信息。

11.2.8 窗口比较器模式

窗口比较器模式的目的是在由上下限阈值定义的指定阈值范围内监控输入信号。

如图 11-4 所示，COMP0 和 COMP1 可以组合在一起来创建一个窗口比较器。输入信号连接到连接在一起的比较器的正极端子，上下限阈值电压连接到比较器的负极端子。

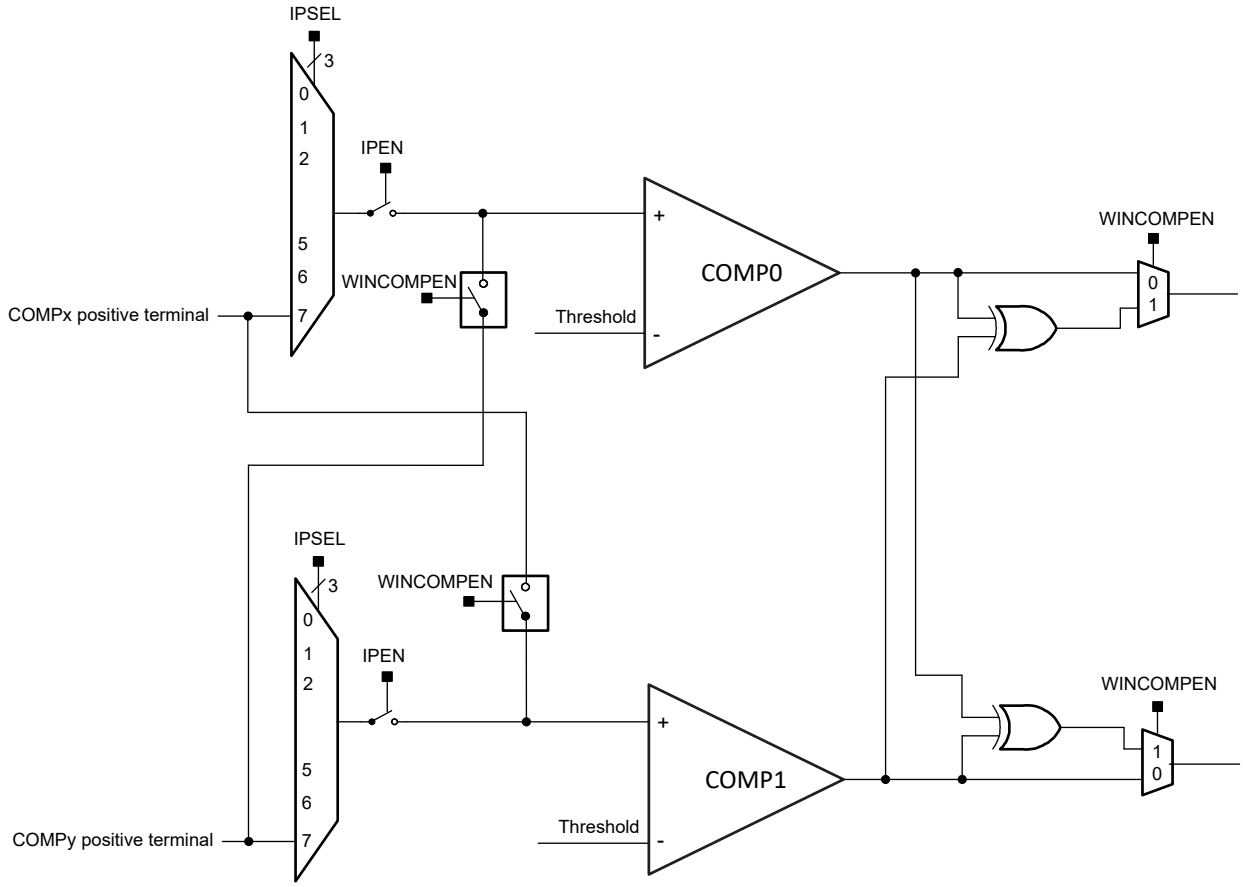


图 11-4. 窗口比较器模式

可以通过设置 COMPx.CTL1 寄存器中的 WINCOMPEN 位来启用窗口比较器模式。下面的图 11-5 展示了一个窗口模式配置示例：

- 将 COMP0.CTL0 寄存器中的 IPSEL 位配置为相应的输入信号引脚。
- 将 COMP0.CTL1 寄存器中的 WINCOMPEN 位设置为 1。
- 清除 COMP1.CTL1 寄存器中的 WINCOMPEN 位。
- 将 COMP1.CTL0 寄存器中的 IPSEL 位配置为 0x07，选择 COMP0 正极端子作为输入。

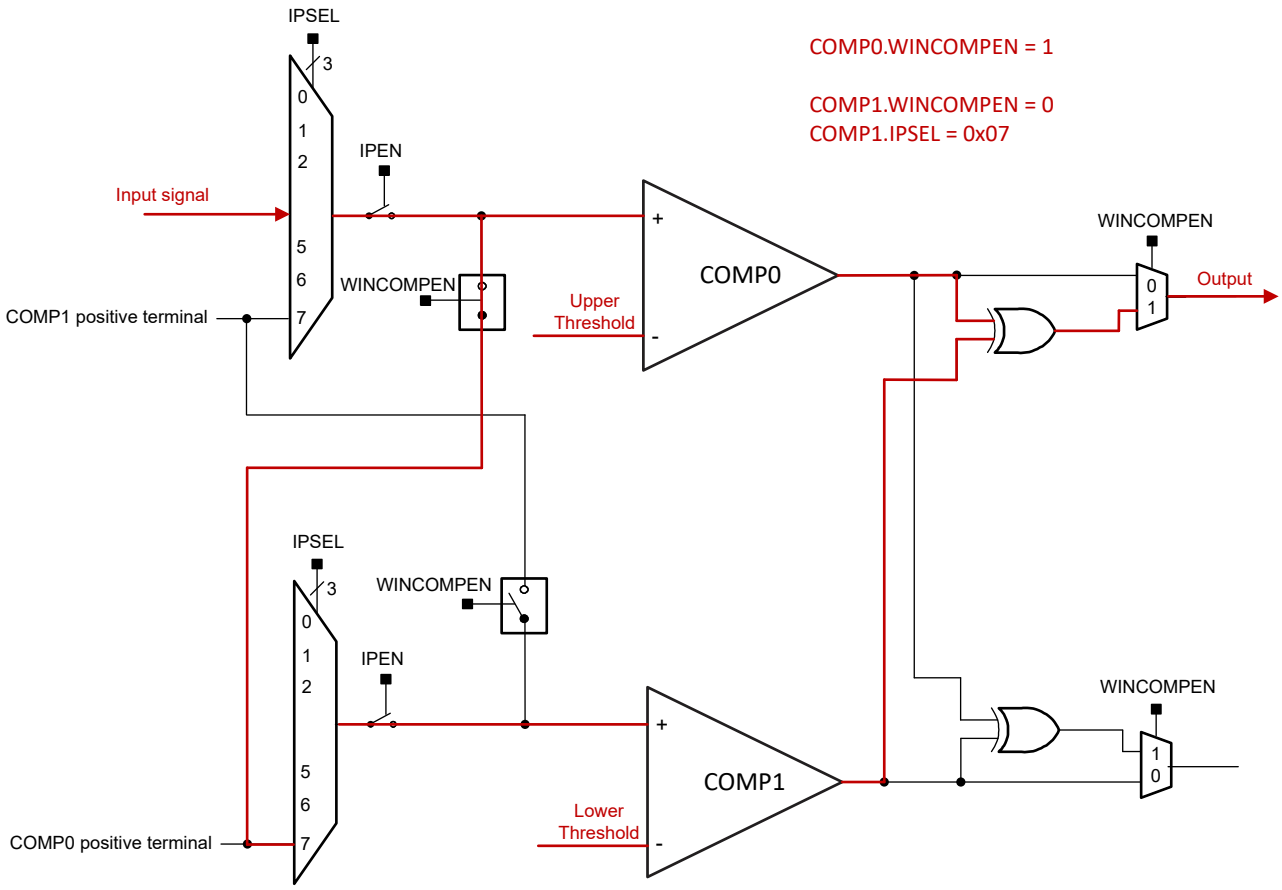


图 11-5. 窗口比较器模式配置

11.2.9 比较器滞后

该比较器支持针对快速和超低功耗模式的可编程迟滞电压，以避免在输入信号出现噪声时发生杂散输出转换。当 Compx.CTL1 寄存器中的 HYST 位为 0 时，比较器不会产生迟滞。比较器为 HYST 位值 1、2 和 3 分别生成 10mV、20mV 和 30mV 典型迟滞电压。

还可以使用内部 8 位 DAC 来实现迟滞。参考信号发生器 8 位 DAC 的输入可以通过 COMPx.CTL3 寄存器中的 DACCODE0 和 DACCODE1 两个值来提供。用户可以将 COMPx.CTL2 寄存器中的 DACCTL 位配置为 1，比较器输出值将在 DACCODE0 和 DACCODE1 值之间选择 DAC 输入。借助这种配置，无需使用外部元件即可为比较器生成所需的迟滞电平。有关此配置的更多信息，请参阅集成 8 位 DAC 部分。

11.2.10 输入短路开关

SHORT 位会使比较器输入短路，从而为比较器提供一个简单的采样保持电路（请参阅图 11-6）

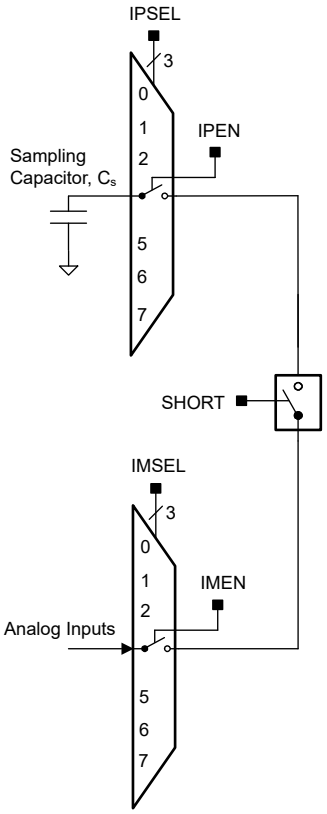


图 11-6. 比较器短路开关

所需的采样时间与采样电容 (CS) 的大小、与短路开关串联的输入开关的电阻 (RI)，以及外部信号源的电阻 (RS) 成比例。采样电容 CS 应大于 100pF。为采样电容 CS 充电的时间常数 Tau 可以使用方程式 22 来计算得出。

$$T_{au} = (R_I + R_S)\text{x}C_S \tag{22}$$

根据所需的精度，可使用 3Tau 至 10Tau 作为采样时间。使用 3Tau 时，采样电容可被充电至输入信号电压电平的 95%，使用 5Tau 时，采样电容可被充电至输入信号电压电平的 99% 以上，而用 10Tau 时，被采样的电压可以充分满足 12 位的精度要求。

11.2.11 中断和事件支持

比较器模块包含两个事件发布者和两个事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理对 CPU 子系统的比较器中断请求 (IRQ)。第二个事件发布者 (GEN_EVENT) 用于通过通用路由设置通用事件发布者。

两个事件订阅者用于定义采样输出模式的起点和终点。

表 11-2 总结了比较器事件。

表 11-2. 比较器事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	比较器	CPU 子系统	静态路由	CPU_INT 寄存器	来自比较器的固定中断路由 CPU 的中断信号
通用发布者事件	发布者	比较器	其他外设	通用路由	GEN_EVENT FPUB_1 寄存器	从比较器到其他外设的可配置路由

表 11-2. 比较器事件 (continued)

事件	类型	源	目标	路由	配置	功能
通用订阅者事件	订阅者	其他外设	比较器	通用路由	GEN_EVENT FSUB_0 寄存器	从其他外设到比较器开始采样窗口的可配置路由
通用订阅者事件	订阅者	其他外设	比较器	通用路由	GEN_EVENT FSUB_1 寄存器	从其他外设到比较器停止采样窗口的可配置路由

11.2.11.1 CPU 中断事件发布者 (CPU_INT)

比较器模块提供 3 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，比较器的 CPU 中断事件为：

表 11-3. 比较器 CPU 中断事件条件 (CPU_INT)

IIDX STAT	名称	说明
0x01	COMPIFG	在比较器输出的上升沿或下降沿设置中断标志 COMPIFG 和 COMPINVIFG，具体由 IES 位选择。当 IES 位为 0 时，比较器输出的上升沿设置 COMPIFG，下降沿设置 COMPINVIFG。当 IES 位为 1 时，比较器输出的下降沿设置 COMPIFG，上升沿设置 COMPINVIFG。
0x02	COMPINVIFG	在比较器输出的上升沿或下降沿设置中断标志 COMPIFG 和 COMPINVIFG，具体由 IES 位选择。当 IES 位为 0 时，比较器输出的上升沿设置 COMPIFG，下降沿设置 COMPINVIFG。当 IES 位为 1 时，比较器输出的下降沿设置 COMPIFG，上升沿设置 COMPINVIFG。
0x03	OUTRDYIFG	比较器输出就绪中断。在比较器输出有效时设置该位。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

11.2.11.2 通用事件发布者 (GEN_EVENT)

通用路由是这样的一种路由，其中发布事件的比较器外设配置为使用多个可用的通用路由通道之一来将事件发布到另一个实体（如果是分离器路由，则为多个实体）。在这里，实体可以是另一个外设、通用 DMA 触发事件或通用 CPU 事件。

GEN_EVENT 寄存器用于选择用于发布事件的外设条件（表 11-4）。FPUB_1 是发布者端口寄存器，用于配置使用哪个通用路由通道来广播事件。另一个外设、DMA 或 CPU 可以通过将其订阅者端口配置为侦听发布外设所连同一通用路由通道来订阅此事件。

例如，通过使用通用事件通道，可以通过将比较器 FPUB_1 和 TIM FSUB_x 连接到同一个通用事件通道，直接从比较器事件触发计时器捕捉。有关通用事件路由的工作方式，请参阅节 7.1.3.3 和节 7.2.3。

表 11-4. 比较器通用事件条件 (GEN_EVENT)

IIDX STAT	名称	说明
0x01	COMPIFG	在比较器输出的上升沿或下降沿设置中断标志 COMPIFG 和 COMPINVIFG，具体由 IES 位选择。当 IES 位为 0 时，比较器输出的上升沿设置 COMPIFG，下降沿设置 COMPINVIFG。当 IES 位为 1 时，比较器输出的下降沿设置 COMPIFG，上升沿设置 COMPINVIFG。
0x02	COMPINVIFG	在比较器输出的上升沿或下降沿设置中断标志 COMPIFG 和 COMPINVIFG，具体由 IES 位选择。当 IES 位为 0 时，比较器输出的上升沿设置 COMPIFG，下降沿设置 COMPINVIFG。当 IES 位为 1 时，比较器输出的下降沿设置 COMPIFG，上升沿设置 COMPINVIFG。
0x03	OUTRDYIFG	比较器输出就绪中断。在比较器输出有效时设置该位。

有关配置事件寄存器的指导，请参阅节 7.2.5。

11.2.11.3 通用事件订阅者

当比较器用于采样输出模式时，两个事件订阅者用于建立采样开始和停止时间。

- FSUB_0 选择发送采样窗口开始事件的器件级别的通用事件路由。
- FSUB_1 选择发送采样窗口停止事件的器件级别的通用事件路由。

由于采样窗口是由通用事件路由驱动的，因此任何具有通用发布者功能的外设模块都可以设置比较器采样窗口，包括计时器和 GPIO。

11.3 COMP 寄存器

表 11-5 列出了 COMP 寄存器的存储器映射寄存器。表 11-5 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 11-5. COMP 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者端口 0		转到
404h	FSUB_1	订阅者端口 1		转到
444h	FPUB_1	发布者端口 1		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
808h	CLKCFG	外设时钟配置寄存器		转到
814h	STAT	状态寄存器		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT	转到
1058h	IMASK	中断屏蔽	GEN_EVENT	转到
1060h	RIS	原始中断状态	GEN_EVENT	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT	转到
1070h	ISET	中断设置	GEN_EVENT	转到
1078h	ICLR	中断清除	GEN_EVENT	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	CTL0	控制 0		转到
1104h	CTL1	控制 1		转到
1108h	CTL2	控制 2		转到
110Ch	CTL3	控制 3		转到
1120h	STAT	状态		转到

复杂的位访问类型经过编码可适应小型表单元。表 11-6 展示了适用于此部分中访问类型的代码。

表 11-6. COMP 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

11.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 11-7 展示了 FSUB_0，表 11-7 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 11-7. FSUB_0

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

表 11-7. FSUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

11.3.2 FSUB_1（偏移 = 404h）[复位 = 00000000h]

图 11-8 展示了 FSUB_1，表 11-8 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 11-8. FSUB_1

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

表 11-8. FSUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

11.3.3 FPUB_1（偏移 = 444h）[复位 = 00000000h]

图 11-9 展示了 FPUB_1，表 11-9 中对此进行了介绍。

返回到汇总表。

发布者端口

图 11-9. FPUB_1

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

表 11-9. FPUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

11.3.4 PWREN（偏移 = 800h）[复位 = 00000000h]

图 11-10 展示了 PWREN，表 11-10 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 11-10. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-							K-0h

表 11-10. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

11.3.5 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 11-11 展示了 RSTCTL，表 11-11 中对此进行了介绍。

返回到汇总表。

用于控制复位置为有效和无效的寄存器

图 11-11. RSTCTL

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

表 11-11. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

11.3.6 CLKCFG（偏移 = 808h）[复位 = 00000000h]

图 11-12 展示了 CLKCFG，表 11-12 中对此进行了介绍。

返回到汇总表。

外设时钟配置寄存器

图 11-12. CLKCFG

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							BLOCKASYNC
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 11-12. CLKCFG 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许状态更改的 KEY -- 0xA9 A9h = 用于写入该寄存器以使写入生效的密钥值。
23-9	保留	R/W	0h	
8	BLOCKASYNC	R/W	0h	已阻止异步时钟请求启动 SYSOSC 或强制总线时钟为 32MHz 0h = 禁用 COMP 请求 SYSOSC 1h = 启用 COMP 请求 SYSOSC
7-0	保留	R/W	0h	

11.3.7 STAT（偏移 = 814h）[复位 = 00000000h]

图 11-13 展示了 STAT，表 11-13 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 11-13. STAT

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

表 11-13. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示，自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自上次 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设尚未复位 1h = 自从上次该位清零以来，外设已复位
15-0	RESERVED	R	0h	

11.3.8 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 11-14 展示了 IIDX，表 11-14 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 11-14. IIDX

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

表 11-14. IIDX 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 2h = 比较器输出中断 3h = 比较器输出反相中断 4h = 比较器输出就绪中断

11.3.9 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 11-15 展示了 IMASK，表 11-15 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 11-15. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				OUTRDYIFG	COMPINVIFG	COMPIFG	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 11-15. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	OUTRDYIFG	R/W	0h	屏蔽 OUTRDYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
2	COMPINVIFG	R/W	0h	屏蔽 COMPINVIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
1	COMPIFG	R/W	0h	屏蔽 COMPIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
0	RESERVED	R/W	0h	

11.3.10 RIS（偏移 = 1030h）[复位 = 00000000h]

图 11-16 展示了 RIS，表 11-16 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 11-16. RIS

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

表 11-16. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	OUTRDYIFG	R	0h	比较器输出就绪中断标志的原始中断状态。在比较器输出有效时设置该位。 0h = 无中断挂起 1h = 中断挂起
2	COMPINVIFG	R	0h	比较器输出反相中断标志的原始中断状态。IES 位定义设置该位的比较器输出的转换。 0h = 无中断挂起 1h = 中断挂起
1	COMPIFG	R	0h	比较器输出中断标志的原始中断状态。IES 位定义设置该位的比较器输出的转换。 0h = 无中断挂起 1h = 中断挂起
0	RESERVED	R	0h	

11.3.11 MIS（偏移 = 1038h）[复位 = 00000000h]

图 11-17 展示了 MIS，表 11-17 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 11-17. MIS

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

表 11-17. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	OUTRDYIFG	R	0h	OUTRDYIFG 的屏蔽中断状态 0h = OUTRDYIFG 不请求中断服务例程 1h = OUTRDYIFG 请求一个中断服务例程
2	COMPINVIFG	R	0h	COMPINVIFG 的屏蔽中断状态 0h = COMPINVIFG 不请求中断服务例程 1h = COMPINVIFG 请求一个中断服务例程
1	COMPIFG	R	0h	COMPIFG 的屏蔽中断状态 0h = COMPIFG 不请求中断服务例程 1h = COMPIFG 请求一个中断服务例程
0	RESERVED	R	0h	

11.3.12 ISET（偏移 = 1040h）[复位 = 00000000h]

图 11-18 展示了 ISET，表 11-18 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 11-18. ISET

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

表 11-18. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	OUTRDYIFG	W	0h	设置 RIS 寄存器中的 OUTRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 OUTRDYIFG 的 RIS 位被设置
2	COMPINVIFG	W	0h	设置 RIS 寄存器中的 COMPINVIFG 0h = 写入 0 不产生影响 1h = 对应于 COMPINVIFG 的 RIS 位被设置
1	COMPIFG	W	0h	设置 RIS 寄存器中的 COMPIFG 0h = 写入 0 不产生影响 1h = 对应于 COMPIFG 的 RIS 位被设置
0	RESERVED	W	0h	

11.3.13 ICLR (偏移 = 1048h) [复位 = 00000000h]

图 11-19 展示了 ICLR，表 11-19 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 11-19. ICLR

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

表 11-19. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	OUTRDYIFG	W	0h	RIS 寄存器中的 OUTRDYIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 OUTRDYIFG 的 RIS 位被清零
2	COMPINVIFG	W	0h	RIS 寄存器中的 COMPINVIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 COMPINVIFG 的 RIS 位被清零
1	COMPIFG	W	0h	RIS 寄存器中的 COMPIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 COMPIFG 的 RIS 位被清零
0	RESERVED	W	0h	

11.3.14 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 11-20 展示了 IIDX，表 11-20 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。除了使用 IIDX 外，用户还可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 11-20. IIDX

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

表 11-20. IIDX 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 2h = 比较器输出中断 3h = 比较器输出反相中断 4h = 比较器输出就绪中断

11.3.15 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 11-21 展示了 IMASK，表 11-21 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 11-21. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				OUTRDYIFG	COMPINVIFG	COMPIFG	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 11-21. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	OUTRDYIFG	R/W	0h	屏蔽 OUTRDYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
2	COMPINVIFG	R/W	0h	屏蔽 COMPINVIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
1	COMPIFG	R/W	0h	屏蔽 COMPIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
0	RESERVED	R/W	0h	

11.3.16 RIS（偏移 = 1060h）[复位 = 00000000h]

图 11-22 展示了 RIS，表 11-22 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 11-22. RIS

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

表 11-22. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	OUTRDYIFG	R	0h	比较器输出就绪中断标志的原始中断状态。在比较器输出有效时设置该位。 0h = 无中断挂起 1h = 中断挂起
2	COMPINVIFG	R	0h	比较器输出反相中断标志的原始中断状态。IES 位定义设置该位的比较器输出的转换。 0h = 无中断挂起 1h = 中断挂起
1	COMPIFG	R	0h	比较器输出中断标志的原始中断状态。IES 位定义设置该位的比较器输出的转换。 0h = 无中断挂起 1h = 中断挂起
0	RESERVED	R	0h	

11.3.17 MIS (偏移 = 1068h) [复位 = 00000000h]

图 11-23 展示了 MIS，表 11-23 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 11-23. MIS

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

表 11-23. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	OUTRDYIFG	R	0h	OUTRDYIFG 的屏蔽中断状态 0h = OUTRDYIFG 不请求中断服务例程 1h = OUTRDYIFG 请求一个中断服务例程
2	COMPINVIFG	R	0h	COMPINVIFG 的屏蔽中断状态 0h = COMPINVIFG 不请求中断服务例程 1h = COMPINVIFG 请求一个中断服务例程
1	COMPIFG	R	0h	COMPIFG 的屏蔽中断状态 0h = COMPIFG 不请求中断服务例程 1h = COMPIFG 请求一个中断服务例程
0	RESERVED	R	0h	

11.3.18 ISET（偏移 = 1070h）[复位 = 00000000h]

图 11-24 展示了 ISET，表 11-24 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 11-24. ISET

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

表 11-24. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	OUTRDYIFG	W	0h	设置 RIS 寄存器中的 OUTRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 OUTRDYIFG 的 RIS 位被设置
2	COMPINVIFG	W	0h	设置 RIS 寄存器中的 COMPINVIFG 0h = 写入 0 不产生影响 1h = 对应于 COMPINVIFG 的 RIS 位被设置
1	COMPIFG	W	0h	设置 RIS 寄存器中的 COMPIFG 0h = 写入 0 不产生影响 1h = 对应于 COMPIFG 的 RIS 位被设置
0	RESERVED	W	0h	

11.3.19 ICLR (偏移 = 1078h) [复位 = 00000000h]

图 11-25 展示了 ICLR，表 11-25 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 11-25. ICLR

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

表 11-25. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	OUTRDYIFG	W	0h	RIS 寄存器中的 OUTRDYIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 OUTRDYIFG 的 RIS 位被清零
2	COMPINVIFG	W	0h	RIS 寄存器中的 COMPINVIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 COMPINVIFG 的 RIS 位被清零
1	COMPIFG	W	0h	RIS 寄存器中的 COMPIFG 清零 0h = 写入 0 不产生影响 1h = 对应于 COMPIFG 的 RIS 位被清零
0	RESERVED	W	0h	

11.3.20 EVT_MODE（偏移 = 10E0h）[复位 = 00000009h]

图 11-26 展示了 EVT_MODE，表 11-26 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清零 RIS）或硬件模式（硬件清零 RIS）下是否禁用每条线

图 11-26. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				EVT1_CFG		INT0_CFG	
R/W-0h				R-2h		R-1h	

表 11-26. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	EVT1_CFG	R	2h	GEN_EVENT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 由软件处理的事件。软件必须清除关联的 RIS 标志。 2h = 由硬件处理的事件。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 由软件处理的事件。软件必须清除关联的 RIS 标志。 2h = 由硬件处理的事件。硬件（另一个模块）会自动清除关联的 RIS 标志。

11.3.21 DESC（偏移 = 10FCh）[复位 = 06110000h]

图 11-27 展示了 DESC，表 11-27 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 11-27. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-611h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-0h				R-				R-0h				R-0h			

表 11-27. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	611h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本
3-0	MINREV	R	0h	IP 的次要版本

11.3.22 CTL0（偏移 = 1100h）[复位 = 00000000h]

图 11-28 展示了 CTL0，表 11-28 中对此进行了介绍。

返回到汇总表。

控制 0 寄存器。

图 11-28. CTL0

31	30	29	28	27	26	25	24
IMEN	RESERVED						
R/W-0h	R/W-0h						
23	22	21	20	19	18	17	16
RESERVED					IMSEL		
R/W-0h					R/W-0h		
15	14	13	12	11	10	9	8
IPEN	RESERVED						
R/W-0h	R/W-0h						
7	6	5	4	3	2	1	0
RESERVED					IPSEL		
R/W-0h					R/W-0h		

表 11-28. CTL0 字段说明

位	字段	类型	复位	说明
31	IMEN	R/W	0h	比较器负极端子的通道输入使能。 0h = 所选的负极端子模拟输入通道被禁用 1h = 所选的负极端子模拟输入通道被启用
30-19	RESERVED	R/W	0h	
18-16	IMSEL	R/W	0h	如果 IMEN 被设置为 1，则表示为比较器的负极端子选择的通道输入。 0h = 选择了通道 0 1h = 选择了通道 1 2h = 选择了通道 2 3h = 选择了通道 3 4h = 选择了通道 4 5h = 选择了通道 5 6h = 选择了通道 6 7h = 选择了通道 7
15	IPEN	R/W	0h	比较器正极端子的通道输入使能。 0h = 所选的正极端子模拟输入通道被禁用 1h = 所选的正极端子模拟输入通道被启用
14-3	RESERVED	R/W	0h	
2-0	IPSEL	R/W	0h	如果 IPEN 被设置为 1，则表示为比较器的正极端子选择的通道输入。 0h = 选择了通道 0 1h = 选择了通道 1 2h = 选择了通道 2 3h = 选择了通道 3 4h = 选择了通道 4 5h = 选择了通道 5 6h = 选择了通道 6 7h = 选择了通道 7

11.3.23 CTL1（偏移 = 1104h）[复位 = 00000000h]

图 11-29 展示了 CTL1，表 11-29 中对此进行了介绍。

返回到汇总表。

控制 1 寄存器。

图 11-29. CTL1

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED			WINCOMPEN	RESERVED	FLTDLY		FLTEN
R/W-0h			R/W-0h	R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
OUTPOL	HYST		IES	SHORT	EXCH	MODE	ENABLE
R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 11-29. CTL1 字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	WINCOMPEN	R/W	0h	该位启用比较器的窗口比较器操作。 0h = 窗口比较器被禁用 1h = 窗口比较器被启用
11	RESERVED	R/W	0h	
10-9	FLTDLY	R/W	0h	这些位选择比较器输出滤波器延迟。请参阅特定于器件的数据表，了解不同滤波器延迟设置的比较器传播延迟具体值。 0h = 70ns 的典型滤波器延迟 1h = 500ns 的典型滤波器延迟 2h = 1200ns 的典型滤波器延迟 3h = 2700ns 的典型滤波器延迟
8	FLTEN	R/W	0h	该位启用比较器输出端的模拟滤波器。 0h = 比较器输出滤波器被禁用 1h = 比较器输出滤波器被启用
7	OUTPOL	R/W	0h	该位选择比较器输出极性。 0h = 比较器输出为同相 1h = 比较器输出为反相
6-5	滞后 (HYST)	R/W	0h	这些位选择比较器的迟滞设置。 0h = 无迟滞 1h = 低迟滞，典型值为 10mV 2h = 中迟滞，典型值为 20mV 3h = 高迟滞，典型值为 30mV
4	IES	R/W	0h	该位选择 COMPIFG 和 COMPINVIFG 的中断边沿。 0h = 上升沿设置 COMPIFG，下降沿设置 COMPINVIFG 1h = 下降沿设置 COMPIFG，上升沿设置 COMPINVIFG
3	SHORT	R/W	0h	该位短接比较器的正负输入端子。 0h = 不短接比较器正负输入端子 1h = 短接比较器正负输入端子

表 11-29. CTL1 字段说明 (continued)

位	字段	类型	复位	说明
2	EXCH	R/W	0h	该位交换比较器输入并使比较器输出反相。 0h = 比较器输入未交换，输出未反相 1h = 比较器输入已交换，输出已反相
1	MODE	R/W	0h	该位选择比较器工作模式。 0h = 比较器处于快速模式 1h = 比较器处于超低功耗模式
0	ENABLE	R/W	0h	该位可以打开比较器。比较器在关闭时不耗电。 0h = 比较器关闭 1h = 比较器开启

11.3.24 CTL2（偏移 = 1108h）[复位 = 00000000h]

图 11-30 展示了 CTL2，表 11-30 中对此进行了介绍。

返回到汇总表。

控制 2 寄存器。

图 11-30. CTL2

31	30	29	28	27	26	25	24
RESERVED							SAMPMODE
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED						DACSW	DACCTL
R/W-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED					BLANKSRC		
R/W-0h					R/W-0h		
7	6	5	4	3	2	1	0
REFSEL	RESERVED	REFSRC			RESERVED		REFMODE
R/W-0h	R/W-0h	R/W-0h			R/W-0h		R/W-0h

表 11-30. CTL2 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R/W	0h	
24	SAMPMODE	R/W	0h	启用比较器的采样模式。 0h = 采样模式被禁用 1h = 采样模式被启用
23-18	RESERVED	R/W	0h	
17	DACSW	R/W	0h	当 DACCTL 位为 1 时，该位在 DACCODE0 和 DACCODE1 之间选择 8 位 DAC。 0h = 选择 DACCODE0 作为 8 位 DAC 1h = 选择 DACCODE1 作为 8 位 DAC
16	DACCTL	R/W	0h	该位决定是比较器输出还是 DACSW 位控制在 DACCODE0 和 DACCODE1 之间的选择。 0h = 比较器输出控制在 DACCODE0 和 DACCODE1 之间的选择 1h = DACSW 位控制在 DACCODE0 和 DACCODE1 之间的选择
15-11	RESERVED	R/W	0h	
10-8	BLANKSRC	R/W	0h	这些位选择比较器的消隐源。 0h = 消隐源被禁用 1h = 选择消隐源 1 2h = 选择消隐源 2 3h = 选择消隐源 3 4h = 选择消隐源 4 5h = 选择消隐源 5 6h = 选择消隐源 6
7	REFSEL	R/W	0h	该位选择所选的基准电压是施加到比较器的正极端子还是负极端子上。 0h = 如果 EXCH 位为 0，则所选基准施加到正极端子上。 如果 EXCH 位为 1，则所选基准施加到负极端子上。 1h = 如果 EXCH 位为 0，则所选基准施加到负极端子上。 如果 EXCH 位为 1，则所选基准施加到正极端子上。
6	RESERVED	R/W	0h	

表 11-30. CTL2 字段说明 (continued)

位	字段	类型	复位	说明
5-3	REFSRC	R/W	0h	这些位选择比较器的基准源。 0h = 基准电压发生器被禁用（本地基准缓冲器和 DAC）。 1h = 选择 VDDA 作为 DAC 的基准源，DAC 输出用作比较器的基准。 2h = 选择 VREF 作为 DAC 的基准，DAC 输出用作比较器的基准。 3h = 在内部 VREF 被缓冲并连接到外部 VREF 引脚的器件中，VREF 用作比较器基准。DAC 关闭。 5h = VDDA 用作比较器基准。 6h = 选择内部基准作为 DAC 的基准源，DAC 输出作为比较器基准。 7h = 内部 VREF 用作比较器的源。并非所有器件都有此选项。
2-1	RESERVED	R/W	0h	
0	REFMODE	R/W	0h	该位请求在快速模式（静态）或低功耗模式（采样）下进行 ULP_REF 带隙运行。比较器模块内部的本地基准缓冲器和 8 位 DAC 也进行相应的配置。 快速模式运行可提供更高的精度，但会消耗更高的电流。低功耗运行会消耗较低的电流，但基准电压精度会降低。仅当 REFLVL > 0 时，比较器才从 ULP_REF 请求基准电压。 0h = ULP_REF 带隙以及比较器内部的本地基准缓冲器和 8 位 DAC 在静态模式下运行。 1h = ULP_REF 带隙以及比较器内部的本地基准缓冲器和 8 位 DAC 在采样模式下运行。

11.3.25 CTL3（偏移 = 110Ch）[复位 = 00000000h]

图 11-31 展示了 CTL3，表 11-31 中对此进行了介绍。

返回到汇总表。

控制 3 寄存器。

图 11-31. CTL3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								DACCODE1								RESERVED								DACCODE0							
R/W-0h								R/W-0h								R/W-0h								R/W-0h							

表 11-31. CTL3 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23-16	DACCODE1	R/W	0h	这是第二个 8 位 DAC 代码。当 DAC 代码为 0x0 时，DAC 输出将为选择的基准电压 x 1/256V。当 DAC 代码为 0xFF 时，DAC 输出将为选择的基准电压 x 255/256V。 0h = 最小 DAC 代码值 FFh = 最小 DAC 代码值
15-8	保留	R/W	0h	
7-0	DACCODE0	R/W	0h	这是第一个 8 位 DAC 代码。当 DAC 代码为 0x0 时，DAC 输出将为选择的基准电压 x 1/256V。当 DAC 代码为 0xFF 时，DAC 输出将为选择的基准电压 x 255/256V。 0h = 最小 DAC 代码值 FFh = 最小 DAC 代码值

11.3.26 STAT（偏移 = 1120h）[复位 = 00000000h]

图 11-32 展示了 STAT，表 11-32 中对此进行了介绍。

返回到汇总表。

状态寄存器。

图 11-32. STAT

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

表 11-32. STAT 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	OUT	R	0h	该位反映比较器输出的值。写入该位对比较器输出没有影响。 0h = 比较器输出为低电平 1h = 比较器输出为高电平

This page intentionally left blank.

OPA 是一款具有可编程增益级的零漂移斩波稳定型运算放大器。本章介绍 OPA 外设的特性和运行情况。

12.1 OPA 概述

集成到 MSPM0xx MCU 平台中的 OPA 外设旨在提供一种在系统中缓冲或放大模拟信号而无需使用分立式运算放大器的方法。

OPA 基于经过出厂修整的高性能放大器核心，并配备可编程增益级反馈环路、可配置输入多路复用器以及用于监控传感器运行状况的烧毁电流源。

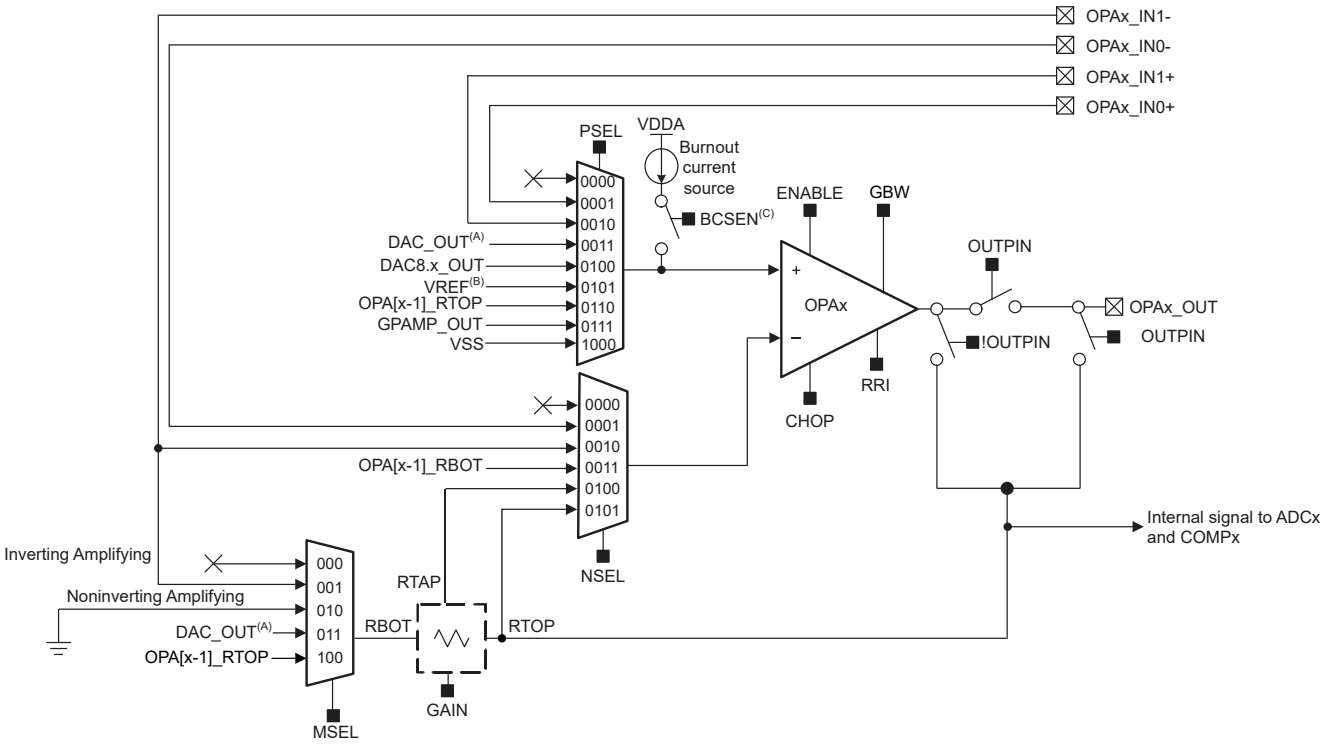
OPA 的特性包括：

- 可通过软件选择零漂移斩波稳定性
- 工厂修整以消除失调误差
- 轨到轨输入和输出
- 支持在满量程电源电压范围内运行
- 双电源模式提高了灵活性
- 高达 32 倍的可编程增益放大器 (PGA)
- 可配置的放大器模式包括通用模式、缓冲模式、反相/同相 PGA 模式、差分放大器模式和级联放大器模式
- 丰富的外部/内部运算放大器端子输入选项
- 双差分输入对可将多个传感器连接到单个 OPA
- 集成烧毁电流源 (BCS) 以监测传感器运行状况
- 在 RUN、SLEEP 和 STOP 模式下运行

备注

大多数器件集成了多个 OPA 外设。如果器件上出现多个 OPA，则它们的运行方式相同。本章中出现的 OPAx 和 OPA[x-1] 等命名规则用于描述 OPA 外设的不同实例。

图 12-1 展示了 OPA 外设的功能方框图。



(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-1. OPA 方框图

12.2 OPA 运行

OPA 由应用软件配置。以下各节介绍了 OPA 外设的设置和运行。

12.2.1 模拟内核

OPA 集成了一个高性能斩波稳定低功耗轨到轨输入/输出运算放大器。它可以实时重新配置以满足特定应用的性能要求。下表描述了所有可配置的运算放大器模拟内核特性和参数：

表 12-1. OPA 可配置参数

参数	配置选项
轨到轨输入 (RRI)	启用/禁用
增益带宽积 (GBW)	6MHz (STD 模式) / 1MHz (LP 模式)
斩波 (CHOP)	禁用/标准斩波/ADC 辅助斩波

在某些基本信号调节应用中，不需要轨到轨输入，因此可以禁用以降低总体功耗。可通过清零 CFGBASE 寄存器中的 RRI 位来禁用轨到轨输入。同样的概念也适用于需要低于 1MHz GBW 的通用应用的增益带宽积。通过清除 CFGBASE 寄存器中的 GBW 位，可以将 OPA 配置为低功耗工作模式 (LP 模式) 。

在一些模拟检测应用中，需要高精度和低温漂。对于这些用例，可启用斩波以显著提高失调电压和温漂性能。有关如何正确配置和利用 OPA 的斩波功能的更多详细信息，请参阅节 12.2.6。

12.2.2 上电行为

PWREN 寄存器中的 ENABLE 位可以将 OPA 外设连接到总线和时钟系统。CTL 寄存器中的 ENABLE 位可以激活 OPA 模拟和数字内核。STAT 寄存器中有一个在 OPA 准备就绪时便会设置的就绪位 (RDY)。请参阅器件特定的数据表，了解 OPA 外设的启用时间。

备注

在写入 CTL.ENABLE 或任何其他 OPA 存储器映射寄存器 (MMR) 之前，必须设置 PWREN.ENABLE。

12.2.3 输入

OPA 使用输入多路复用器 P-MUX 和 N-MUX 来为放大器的同相和反相端子提供一系列输入通道。这些输入可通过 PSEL 和 NSEL 控制位进行配置。

下面表 12-2 概括了输入通道映射。有关可用 OPAx 输入的更详细描述，请参阅特定于器件的数据表。

表 12-2. 通用 OPAx 输入通道

PSEL	OPAx 同相通道	NSEL	OPAx 反相输入通道
0x0	开路	0x0	开路
0x1	OPAx_IN0+	0x1	OPAx_IN0-
0x2	OPAx_IN1+	0x2	OPAx_IN1-
0x3	DAC_OUT	0x3	OPA[x-1]_RBOT
0x4	DAC8.x_OUT	0x4	RTAP
0x5	VREF	0x5	RTOP
0x6	OPA[x-1]_RTOP		
0x7	GPAMP 输出		
0x8	VSS		

备注

OPAx 和 OPA[x-1] 指的是器件中有 2 个 OPA 外设的实例。在这些情况下，外设是配对的，输入/输出是共享的，如表 12-2 所示。DAC8.x 是指配对的 COMP 基准 DAC，可路由到 OPAx 的同相输入。

12.2.4 输出

OPA 输出连接到一个互补开关网络，以允许放大器输出为纯内部信号或允许通过器件引脚访问该信号。如果 OUTPIN = 0，则该输出在内部连接到可编程增益级，并且能够路由到其他模拟外设。如果 OUTPIN = 1，则该输出会进入器件引脚，同时仍保持与可编程增益级和其他模拟外设的连接。通过访问器件引脚上的放大器输出可以使用外部滤波电路。

12.2.5 时钟要求

OPA 要求 SYSOSC 有效以实现正常运行，从而支持斩波和轨到轨输入等功能。未进行任何配置以让 OPA 从 SYSCTL 自动请求 SYSOSC，因此用户必须确保 SYSOSC 被启用以使用 OPA。输入时钟的最低要求取决于下表所示的 OPA 配置。

表 12-3.

RRI	支持的 SYSOSC 频率
0	4/16/24/32MHz
1	32MHz

备注

应用软件需要确保将 SYSOSC 编程为适合轨到轨输入的频率。

12.2.6 斩波

OPA 外设实现了斩波稳定，以减少失调电压、漂移和 1/f 噪声。对于需要外部滤波器的标准斩波模式，CFG.CHOP 可被设定为 0x1。请注意，OPA 会根据用于选择 OPA PGA 增益的 CFG.GAIN 自动调节斩波频率，但在使用标准斩波时，也需要使用外部滤波器，如表 12-4 所示。

表 12-4. OPA 的标准斩波频率

增益	反相/同相增益	斩波频率 (Hz)	电阻器 (kΩ)	电容器 (nF)
0x0	1x	125k	1	10
0x1	-1x / 2x	62.5k	1	30
0x2	-3x / 4x	31.25k	1	90
0x3	-7x / 8x	15.6k	1	270
0x4	-15x / 16x	7.8k	1	810
0x5	-31x / 32x	3.9k	1	2400

12.2.7 OPA 放大器模式

OPA 使用一个连接到可编程增益级电阻梯的输入多路复用器 M-MUX，允许用户针对各种模拟信号链配置对 OPA 进行编程。这些配置包括单位增益缓冲器、反相/同相放大、PGA、级联放大，等等。

在下面的表 12-5 概括了 M-MUX 的输入通道映射。请参阅器件特定的数据表，了解可用的 M-MUX 输入的详细模式描述。

表 12-5. 通用 OPA M-MUX 通道映射

MSEL	OPAx M-MUX 输入通道
0x0	开路
0x1	OPAx_IN1-
0x2	GND

表 12-5. 通用 OPA M-MUX 通道映射 (continued)

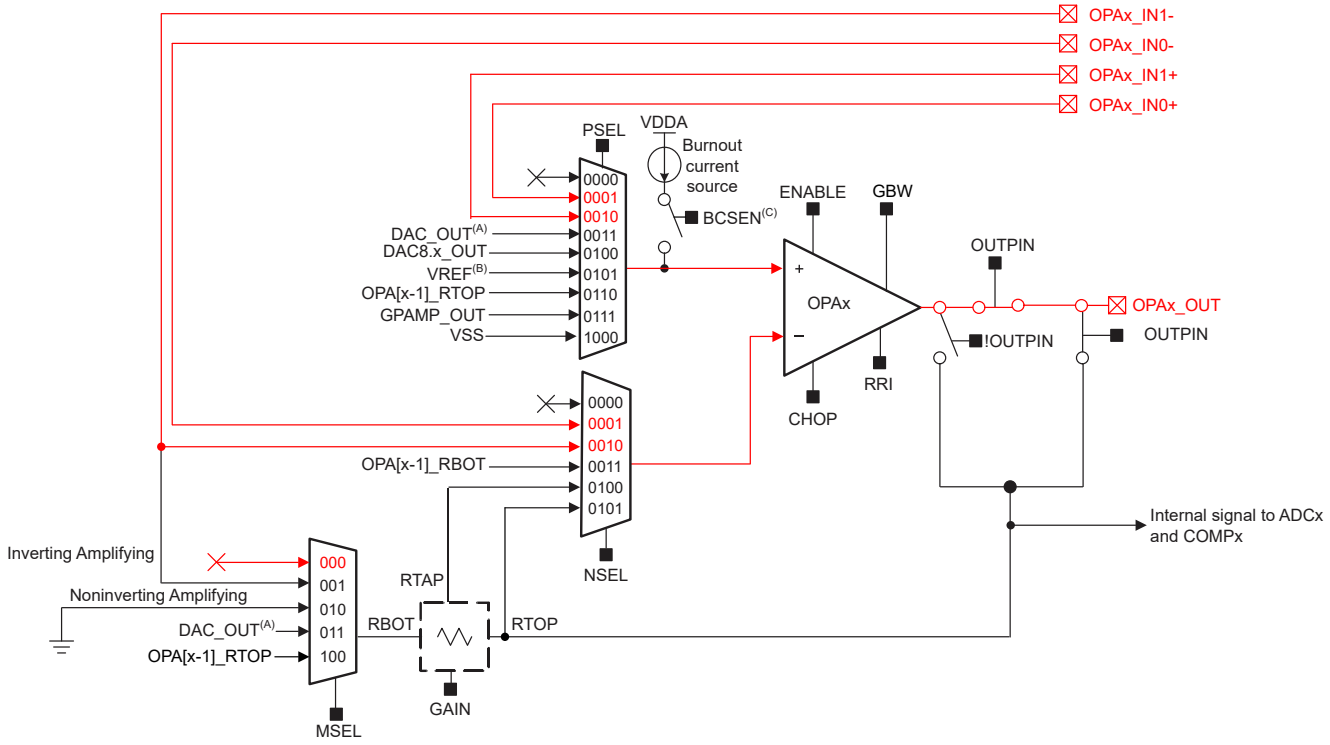
MSEL	OPAx M-MUX 输入通道
0x3	DAC_OUT
0x4	OPA[x-1]_RTOP

以下各节举例说明了如何针对各种放大器模式配置 P-MUX、N-MUX 和 M-MUX。

12.2.7.1 通用模式

OPA 内部多路复用器支持 GP 模式，此模式允许从器件引脚访问所有输入和输出端子。通过将此模式与外部组件配合使用，放大器可以实现分立式运算放大器可实现的任何模拟信号链电路。双路差分输入 IN0+/- 和 IN1+/- 允许将两个不同的输入网络连接到同一个 OPA 外设。

图 12-2 显示了 GP 模式下的 OPA 方框图。



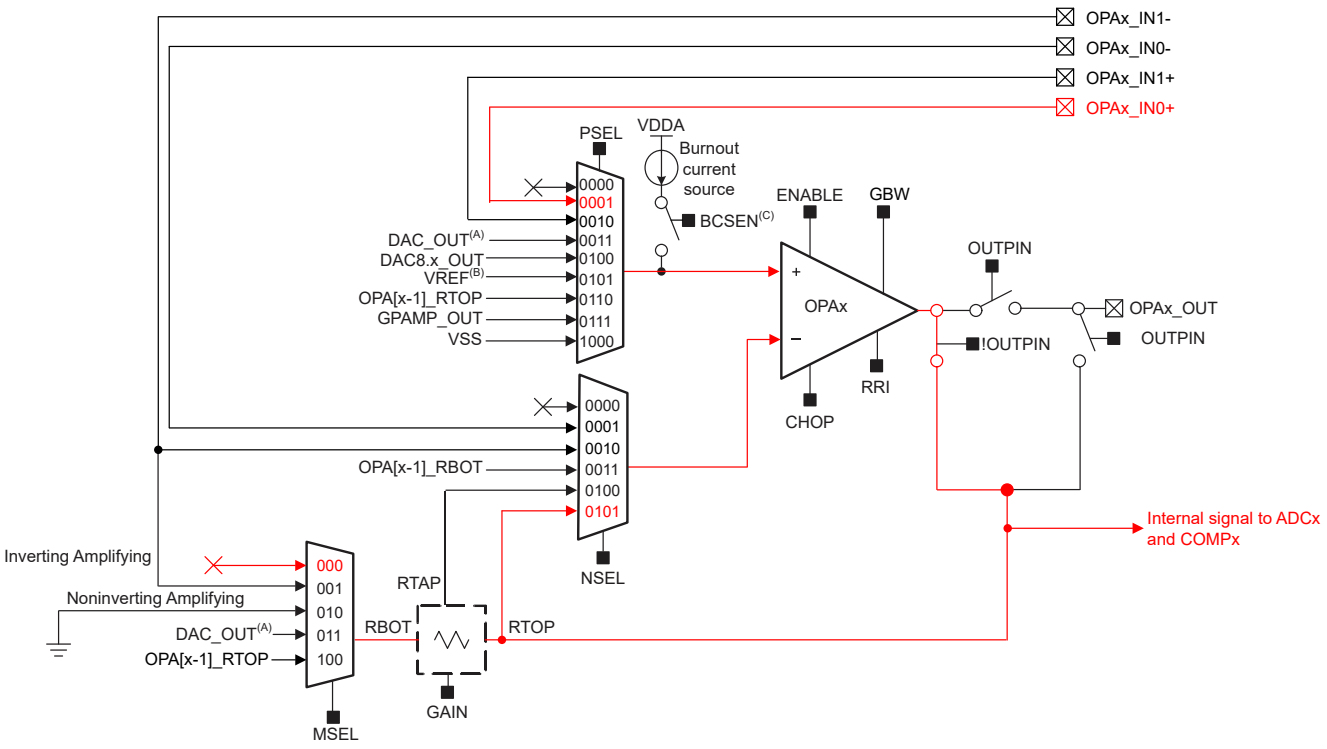
(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-2. 通用 (GP) 模式下的 OPA

12.2.7.2 缓冲模式

可将内部反馈环编程为单位增益配置，以支持 OPA 缓冲模式。同相输入可以来自各种信号，例如外部 OPAx_INx 或板载 DAC 和 VREF 外设。OPA 可以通过设置 OUTPIN = 0x1 输出到外部引脚，或者通过设置 OUTPIN = 0x0 在内部路由到板载模拟外设，例如 ADC、COMP 和/或配对 OPA。

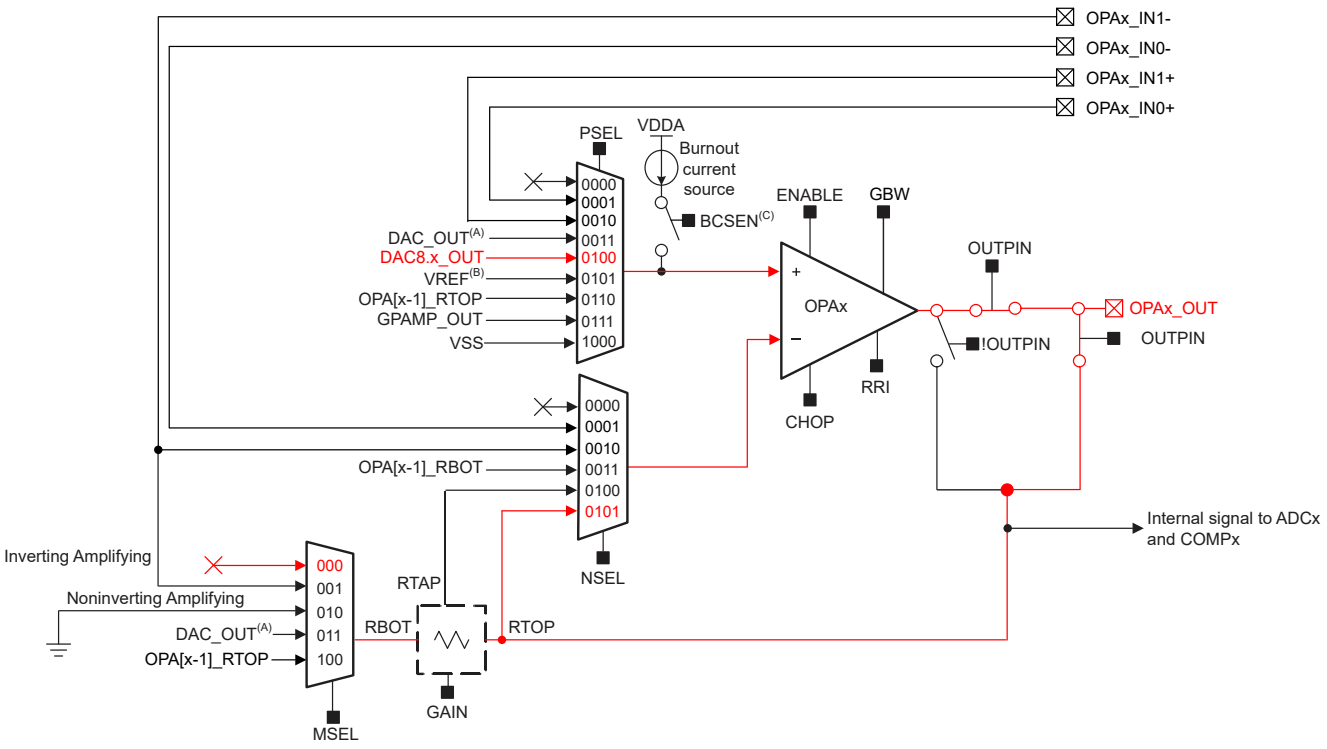
图 12-3 展示了缓冲模式下的 OPA 方框图，其中外部 OPAx_IN0+ 作为同相输入和输出在内部路由到 ADC。



(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-3. ADC 缓冲器配置中的 OPA

图 12-4 展示了缓冲模式下的 OPA 方框图，其中 DAC8.x_OUT 作为同相输入，输出路由到器件引脚。该模式使用户能够将 OPA 外设用作比较器板载基准 DAC 的输出缓冲器。



(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-4. DAC 输出缓冲器配置中的 OPA

12.2.7.3 OPA PGA 模式

OPA 在反馈环路中集成了一个可编程增益级以将 OPA 配置为 PGA（可编程增益放大器）。增益级是反馈电阻梯，支持高达 32 倍放大。表 12-6 列出了 OPA PGA 模式增益设置。

表 12-6. PGA 模式和增益设置

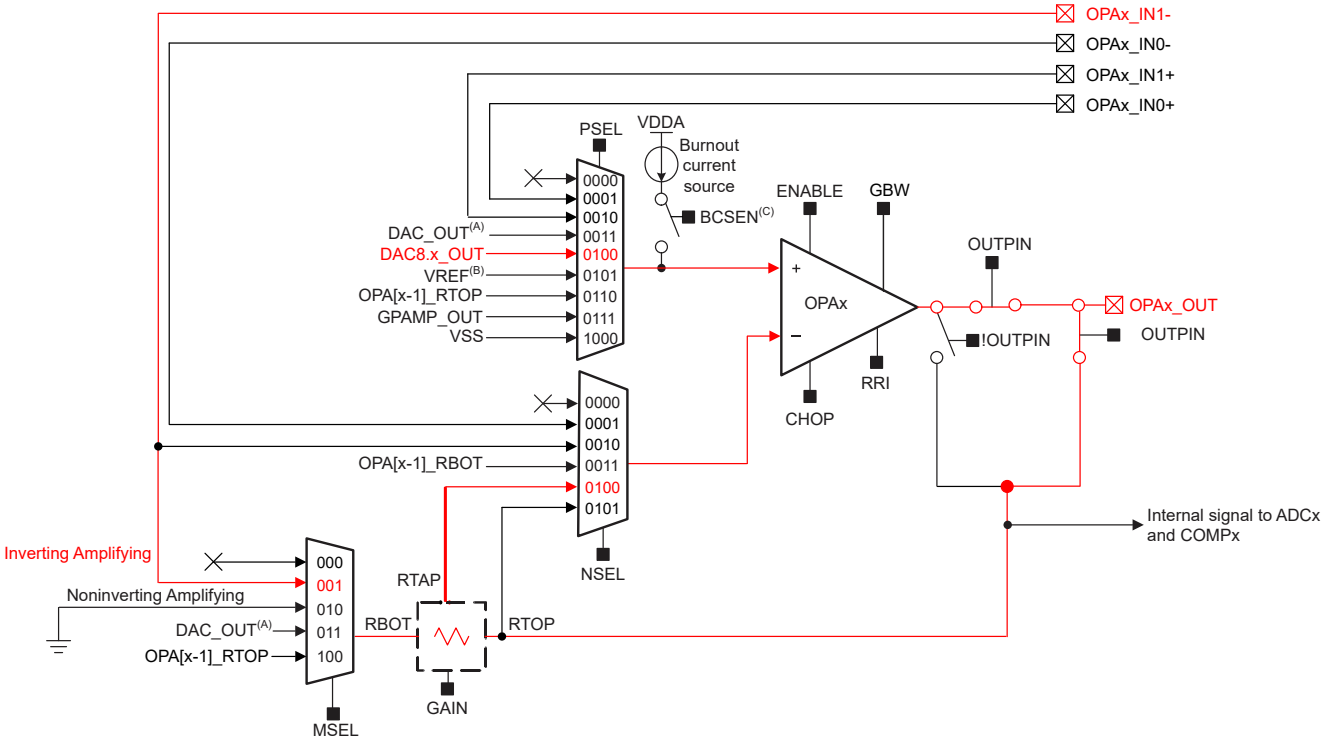
PGA 模式配置	增益	来自 PGA 的增益值（内部增益）
反相 PGA 模式	0x0	无效
	0x1	-1x
	0x2	-3x
	0x3	-7x
	0x4	-15x
	0x5	-31x
同相 PGA 模式	0x0	不有效
	0x1	2x
	0x2	4x
	0x3	8x
	0x4	16x
	0x5	32x

可编程增益级与 P-MUX、N-MUX 和 M-MUX 配合使用，支持各种内部放大器配置，后续部分将详细介绍这些配置。

12.2.7.3.1 反相 PGA 模式

在此模式下，反相输入来自 PGA 电阻梯上的外部 OPAx_IN1-。MSEL 位选择 OPAx_IN1- 作为源 (MSEL = 0x1)，NSEL 位必须选择 RTAP 作为放大器的反相输入 (NSEL = 0x4)。由于 OPA 是单电源放大器，因此需要对同相输入进行偏置，以确保输出保持在线性范围内。同相输入可以来自外部 OPAx_INX+ 输入或 DAC 输入。

图 12-5 展示了 OPA 反相 PGA 模式的一个示例，其中以 8 位基准 DAC 作为偏置消除。



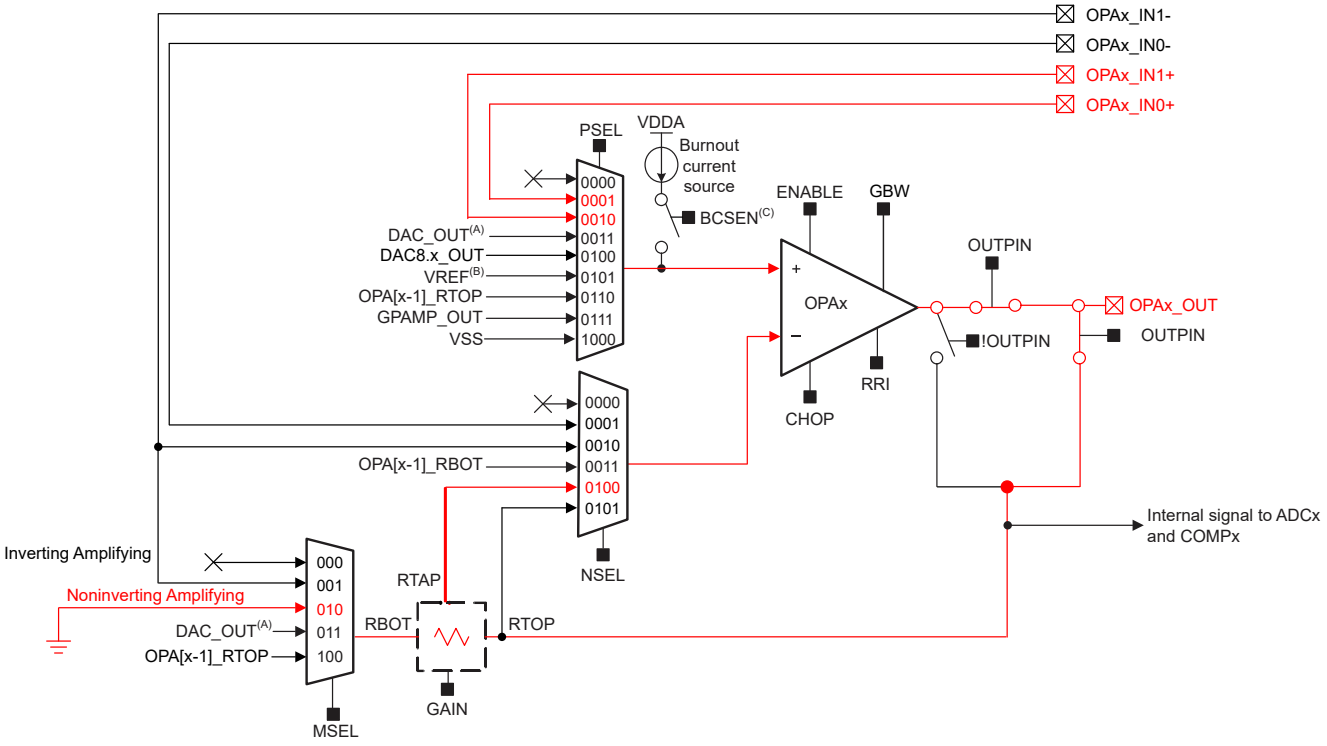
(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-5. 采用反相 PGA 配置的 OPA

12.2.7.3.2 同相 PGA 模式

在此模式下，同相输入来自外部 OPAx_INx+ 输入（PSEL = 0x1 或 0x2）。NSEL 位必须选择 RTAP 作为放大器的反相输入 (NSEL = 0x4)，并从 MSEL 位中选择一个来自接地或 12 位缓冲 DAC (DAC_OUT) 的偏置。

图 12-6 展示了同相 PGA 模式下的 OPA 示例。



(A) Available on select devices. See the device-specific data sheet for availability.
(B) From VREF module
(C) From ADC MMR

图 12-6. 采用同相 PGA 配置的 OPA

12.2.7.4 差分放大器模式

当一个器件上有两个或多个 OPA 时，可以将两个 OPA 组合成一个差分放大器。图 12-7 中的 Vdiff 公式为差分放大器的输出公式。两个 OPA 的内部连接配置，输出由 ADC 采样，如图 12-8 所示。

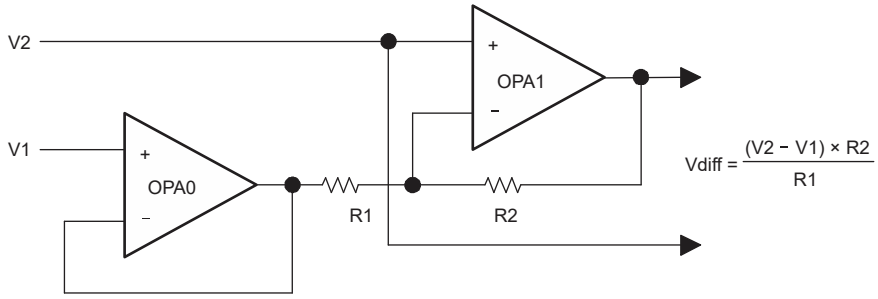


图 12-7. 双 OPA 差分放大器的方框图和公式

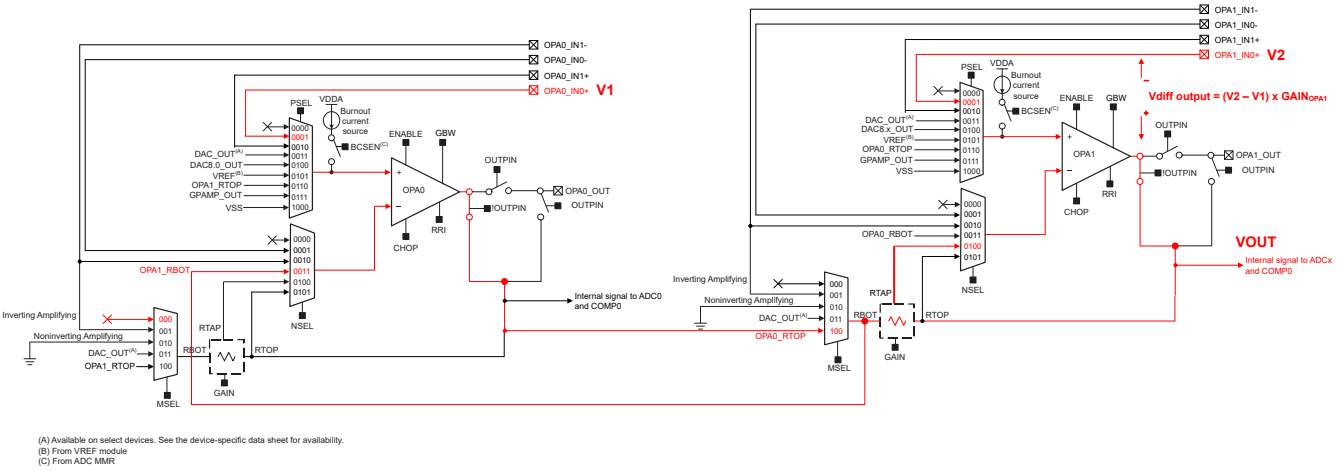


图 12-8. 差分放大器配置中的两个 OPA

12.2.7.5 级联放大器模式

当一个器件上有两个或更多个 OPA 时，可以将两个 OPA 组合成一个多级或级联放大器。使用可编程输入多路复用器可以实现反相和同相多级放大器的所有组合。V_{out} 公式（图 12-9）为同相到同相级联放大器的输出公式。两个 OPA 的内部连接配置如图 12-10 所示。

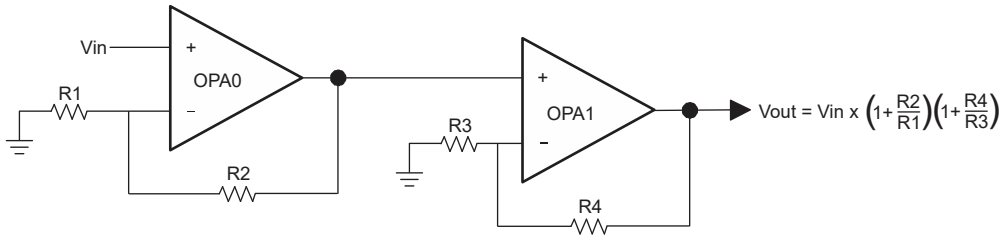


图 12-9. 双 OPA 同相到同相级联放大器的方框图和公式

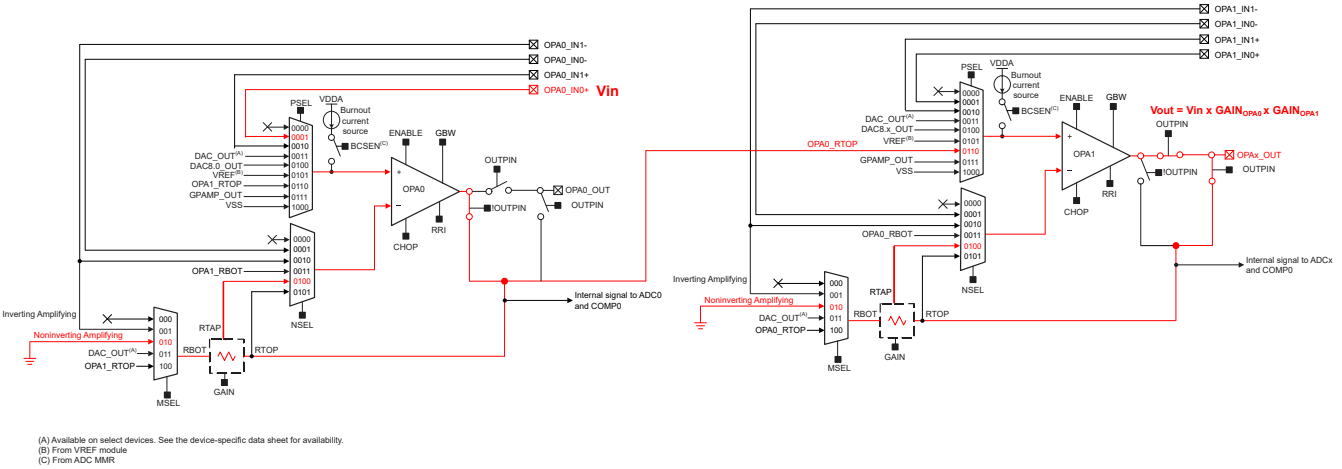


图 12-10. 同相到同相级联放大器配置中的两个 OPA

12.2.8 选择 OPA 配置

OPA 外设允许客户动态更改 OPA 增益，而无需使用寄存器 CFG.GAIN 控制位来禁用 OPA 功能。当使用控制位 CFG.GAIN 在不同的增益值之间进行选择时，用户必须确保额外的 CFG 控制位保持不变，并且 CFG.GAIN 控制位不被清除为 0x0。

为确保 OPA 正常运行，除了由 CFG 寄存器控制的 CFG.GAIN 位之外的配置设置只能在 OPA 被禁用时更改。这意味着为了在应用程序中更改 OPA 输入或模式，用户必须执行以下操作：

- 清除 CTL.ENABLE 位
- 更新 CFG.MSEL、CFG.NSEL 或 CFG.PSEL 位
- 设置 CTL.ENABLE 位

备注

在执行该过程时，应用程序必须考虑 OPA 启用时间。有关详细信息，请参阅特定于器件的数据表中的 OPA 规格。

备注

如果将 CFG.GAIN 位的值更改为 0x0 或从 0x0 更改为其他值，那么也必须遵循此过程。

12.2.9 烧毁电流源

OPA 有烧毁电流源 (BCS)，可在 ADC 请求时使用。当 ADC 被选为 ADC 输入通道且 ADC MEMCTL 寄存器中的 BCSEN 位置位时，ADC 将向 OPA 外设提供烧毁电流源使能信号。启用后，烧毁电流源将向 OPA 的正输入注入电流。然后，应使用 ADC 测量传感器或电阻惠斯通电桥两端的电压，输入信号通过配置为缓冲模式的 OPA（或通常使用的任何 OPA 配置）。然后，启用烧毁电流源的 ADC 读数可以检测传感器是开路（即断线导致全范围信号）还是短路 (0V)。用户可以定期启用烧毁电流源，以检查传感器是否正常运行。测量期间应将其关闭。

12.3 OA 寄存器

表 12-7 列出了 OA 寄存器的存储器映射寄存器。表 12-7 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 12-7. OA 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1010h	CLKOVR	时钟覆盖		转到
101Ch	PWRCTL	功率控制		转到
1100h	CTL	控制寄存器		转到
1108h	CFG	配置寄存器		转到
1118h	STAT	状态寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 12-8 显示了适用于此部分中访问类型的代码。

表 12-8. OA 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

12.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 12-11 展示了 PWREN，表 12-9 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 12-11. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 12-9. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

12.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 12-12 展示了 RSTCTL，表 12-10 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 12-12. RSTCTL

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

表 12-10. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

12.3.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 12-13 展示了 STAT，表 12-11 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 12-13. STAT

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

表 12-11. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

12.3.4 CLKOVR（偏移 = 1010h）[复位 = 00000000h]

图 12-14 展示了 CLKOVR，表 12-12 中对此进行了介绍。

返回到汇总表。

该寄存器覆盖了外设向系统发出的功能时钟请求

图 12-14. CLKOVR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED						RUN_STOP	OVERRIDE
R/W-0h						R/W-0h	R/W-0h

表 12-12. CLKOVR 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	RUN_STOP	R/W	0h	如果启用了 OVERRIDE，该寄存器用于手动控制外设向系统发出的时钟请求 0h = 运行/取消门控功能时钟 1h = 停止/门控功能时钟
0	OVERRIDE	R/W	0h	解锁 RUN_STOP 的功能以覆盖自动外设时钟请求 0h = 禁用覆盖 1h = 启用覆盖

12.3.5 PWRCTL（偏移 = 101Ch）[复位 = 00000001h]

图 12-15 展示了 PWRCTL，表 12-13 中对此进行了介绍。

返回到汇总表。

该寄存器控制外设处于空闲状态时是否被禁用。

图 12-15. PWRCTL

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

表 12-13. PWRCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	AUTO_OFF	R/W	1h	设置后，外设将删除其本地 IP“启用请求”，以便在系统中没有其他实体请求启用该外设时将其禁用。 0h = 禁用自动断电功能 1h = 启用自动断电功能

12.3.6 CTL（偏移 = 1100h）[复位 = X]

图 12-16 展示了 CTL，表 12-14 中对此进行了介绍。

返回到汇总表。

控制寄存器

图 12-16. CTL

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

表 12-14. CTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	ENABLE	R/W	0h	OAxn 使能。 0h (R/W) = OAxn 关闭 1h (R/W) = OAxn 开启

12.3.7 CFG（偏移 = 1108h）[复位 = 00000000h]

图 12-17 展示了 CFG，表 12-15 中对此进行了介绍。

返回到汇总表。

配置寄存器

图 12-17. CFG

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
增益			MSEL			NSEL	
R/W-0h			R/W-0h			R/W-0h	
7	6	5	4	3	2	1	0
NSEL	PSEL				OUTPIN	CHOP	
R/W-0h	R/W-0h				R/W-0h	R/W-0h	

表 12-15. CFG 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-13	增益	R/W	0h	增益设置。有关枚举信息，请参阅 TRM。 0h = 最小增益值 7h = 最大增益值。
12-10	MSEL	R/W	0h	MSEL 多路复用器选择。 有关可用的确切通道，请参阅器件特定数据表。 0h (R/W) = 无连接 1h (R/W) = 外部引脚 OAn-1 2h (R/W) = VSS 3h (R/W) = DAC12 输出 4h (R/W) = OA[n-1]Rtop
9-7	NSEL	R/W	0h	负 OA 输入选择。 有关可用的确切通道，请参阅器件特定数据表。 0h (R/W) = 无连接 1h (R/W) = 外部引脚 OAn-0 2h (R/W) = 外部引脚 OAn-1 3h (R/W) = OA[n+1]Rbot 4h (R/W) = OA[n]Rtap 5h (R/W) = OA[n]Rtop 6h (R/W) = 备用输入
6-3	PSEL	R/W	0h	正 OA 输入选择。 有关可用的确切通道，请参阅器件特定数据表。 0h (R/W) = 无连接 1h (R/W) = 外部引脚 OA+0 2h (R/W) = 外部引脚 OAn+1 3h (R/W) = DAC12OUT 4h (R/W) = DAC8OUT 5h (R/W) = VREF 通道 6h (R/W) = OA[n-1]Rtop 7h (R/W) = GPAMP_OUT_INT 输入 8h = 内部接地连接

表 12-15. CFG 字段说明 (continued)

位	字段	类型	复位	说明
2	OUTPIN	R/W	0h	启用输出引脚 0h (R/W) = 禁用输出引脚 1h (R/W) = 启用输出引脚
1-0	CHOP	R/W	0h	斩波使能。 0h (R/W) = 斩波禁用。 1h (R/W) = 标准斩波启用。 2h (R/W) = 进入平均模式后的斩波。需要在平均模式下将输出端连接到 ADC。

12.3.8 STAT（偏移 = 1118h）[复位 = X]

图 12-18 展示了 STAT，表 12-16 中对此进行了介绍。

返回到汇总表。

状态寄存器

图 12-18. STAT

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

表 12-16. STAT 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	RDY	R	0h	OA 就绪状态。 0h (R) = OAxn 未就绪。 1h (R) = OAxn 已就绪。

This page intentionally left blank.

章节 13
GPAMP

GPAMP 是一款具有轨到轨输入和输出的斩波稳定型通用运算放大器。本章介绍了 GPAMP 外设的特性和运行情况。

13.1 GPAMP 概述

集成到 MSPM0xx MCU 平台中的 GPAMP 外设旨在提供一种在系统中缓冲或放大模拟信号而无需使用分立式运算放大器的方法。

GPAMP 基于经过出厂修整的放大器核心，并配备可配置的反相输入多路复用器以实现各种模拟信号链配置。

GPAMP 的特性包括：

- 软件可选斩波稳定
- 轨到轨输入和输出
- 支持在满量程电源电压范围内运行
- 可编程内部单位增益反馈环路
- 在 RUN、SLEEP 和 STOP 模式下运行

图 13-1 展示了 GPAMP 外设的功能方框图。

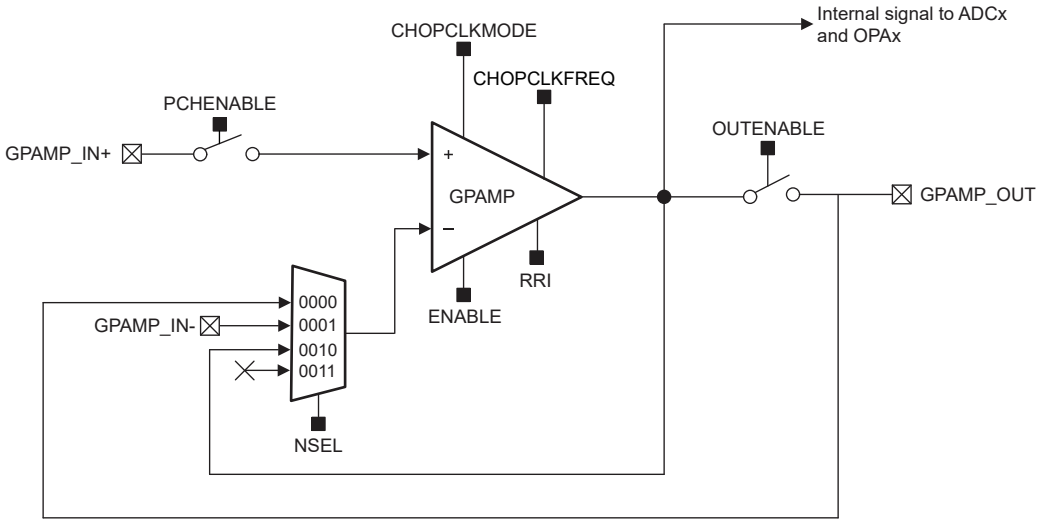


图 13-1. GPAMP 方框图

13.2 GPAMP 操作

GPAMP 外设由用户软件使用 PMUOPAMP 寄存器进行配置，其中 PMUOPAMP 寄存器位于存储器映射寄存器的 PMCU 区域中。对于位字段定义，您可以参阅本章末尾或 PMCU 的寄存器部分。以下各节讨论了 GPAMP 的设置和操作。

13.2.1 模拟内核

GPAMP 集成了斩波稳定型低功耗轨到轨输入/输出运算放大器。它可以实时重新配置以满足各种应用的性能要求。下表描述了所有可配置的放大器模拟内核特性和参数：

表 13-1. GPAMP 可配置参数

参数	配置选项
轨到轨输入 (RRI)	模式 0 /模式 1 /模式 2
斩波 (CHOPCLKMODE)	标准斩波/禁用

在某些基本信号调节应用中，不需要轨到轨输入，因此可以禁用以降低总体功耗。可通过将 RRI 位字段编程为 0x0（模式 0）来禁用轨到轨输入。有关各种 RRI 模式下支持的共模输入范围，请参阅器件特定数据表中的 GPAMP 规格。

在一些模拟检测应用中，需要高精度和低温漂。对于这些用例，可启用斩波以显著提高失调电压和温漂性能。有关如何正确配置和利用 GPAMP 的斩波功能的更多详细信息，请参阅节 13.2.6。

13.2.2 上电行为

PMUOPAMP 寄存器中的 ENABLE 位可以激活 GPAMP 模拟和数字内核。GPAMP 要求 PMU 中的 VBOOST 电路在使用前保持稳定。请参阅节 2.2.6，详细了解什么是 VBOOST 电路，以及为何需要使用它来实现 GPAMP 的正常运行。请参阅器件特定的数据表，了解 GPAMP 外设的启用时间。

13.2.3 输入

GPAMP 使用输入多路复用器 N-MUX 来为放大器的反相端子提供一系列输入通道。这些输入可通过 NSEL 控制位进行配置。放大器的同相端子通过 PCHENABLE 控制位启用，并允许从器件引脚上的 GPAMP_IN+ 信号驱动输入。

下面表 13-2 描述了输入通道映射。

表 13-2. GPAMP 输入通道

PCHENABLE	GPAMP 同相输入	NSEL	GPAMP 反相输入通道
0x0	开路	0x0	GPAMP 输出焊盘 (GPAMP_OUT)
0x1	GPAMP_IN+	0x1	GPAMP_IN-
		0x2	GPAMP 内部放大器输出
		0x3	开路

13.2.4 输出

GPAMP 输出连接到一个开关，以允许放大器输出为纯内部信号或允许通过器件引脚访问该信号。如果 OUTENABLE = 1，则放大器输出会进入器件引脚，同时仍保持与反相输入多路复用器和其他模拟外设的连接。如果 OUTENABLE = 0，则放大器输出会与器件引脚断开，但仍在内部连接到反相输入多路复用器和其他板载模拟外设。通过访问器件引脚上的放大器输出可以使用外部滤波电路。请参阅 GPAMP 方框图，了解 GPAMP 输出电路的详细原理图。

13.2.5 GPAMP 放大器模式

GPAMP 使用 N-MUX 和内部单位增益反馈环路，允许用户针对各种模拟信号链配置对 GPAMP 进行编程。这些配置包括通用模式、ADC 缓冲器和单位增益模式。

以下各节举例说明了如何针对这些不同的放大器模式配置 GPAMP。

13.2.5.1 通用模式

GPAMP 支持通用 (GP) 模式，允许从器件引脚访问所有输入和输出端子。要实现此 GP 模式，NSEL、PCHENABLE 和 OUTENABLE 必须全部设置为 0x1。使用此模式需要外部组件和适当的反馈环路来构建所需的电路。

图 13-2 显示了 GP 模式下的 GPAMP 方框图。

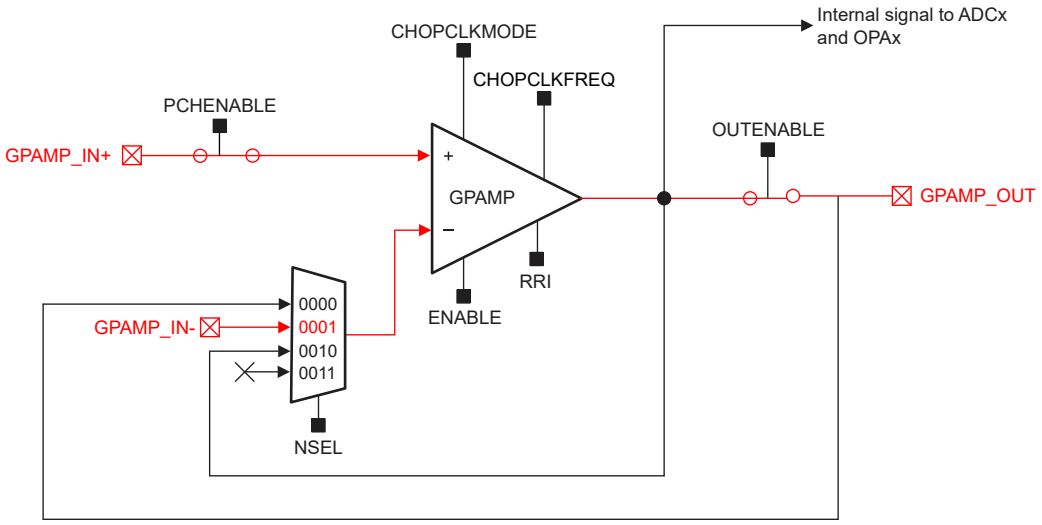


图 13-2. 通用 (GP) 模式下的 GPAMP

13.2.5.2 ADC 缓冲模式

GPAMP 可用于在内部将模拟信号缓冲到板载 ADC 中。要实现此 ADC 缓冲模式，NSEL 必须设置为 0x2，PCHENABLE 必须设置为 0x1，OUTENABLE 必须设置为 0x0。

图 13-3 显示了 ADC 缓冲模式下的 GPAMP 方框图，其中 GPAMP_IN+ 作为同相输入和输出在内部路由到 ADC。

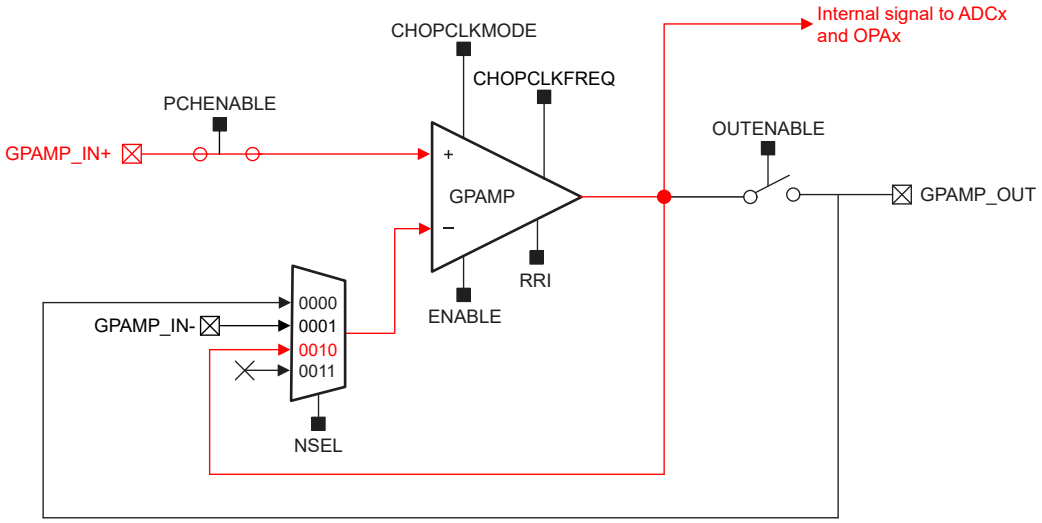


图 13-3. 采用 ADC 缓冲器配置的 GPAMP

13.2.5.3 单位增益模式

GPAMP 的内部反馈环路可用于实现单位增益放大器以将模拟信号缓冲到系统中的其他器件。为了实现这种单位增益模式，必须将 NSEL 设置为 0x0，必须将 PCHENABLE 设置为 0x1，必须将 OUTENABLE 设置为 0x1。

图 13-4 所示为单位增益模式下的 GPAMP 的方框图。

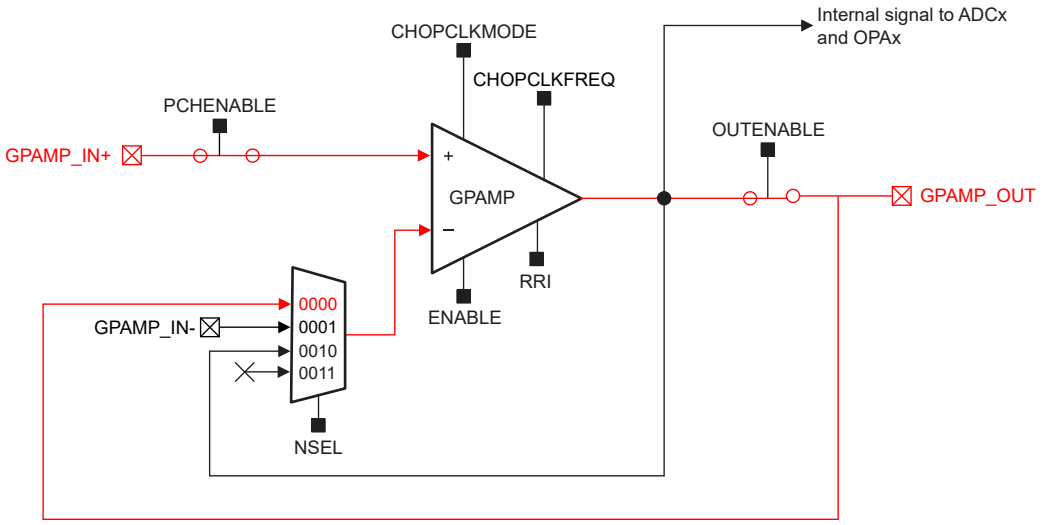


图 13-4. 单位增益配置中的 GPAMP

13.2.6 斩波

GPAMP 外设实现了斩波稳定，以减少失调电压、漂移和 1/f 噪声。对于需要外部滤波器的标准斩波模式，CHOPCLKMODE 可被设定为 0x1。有关建议值，请参阅具体器件的数据表。由 CHOPCLKFREQ 设置的斩波频率需要随着增益的增加而降低。有关给定增益所需的斩波频率的信息，请参阅表 13-3。

表 13-3. GPAMP 的标准斩波频率

CHOPCLKFREQ	支持的外部增益配置	斩波频率 (Hz)
0x0	-1x / 2x	16k
0x1	-3x / 4x	8k
0x2	-7x / 8x	4k
0x3	-15x / 16x	2k

13.3 GPAMP 寄存器

GPAMP 控制寄存器 (PMUOPAMP) 位于 SYSCTL 寄存器中。有关详细信息，请参阅 PMUOPAMP 寄存器。

This page intentionally left blank.

章节 14
DAC

DAC 模块是一款 12 位电压输出数模转换器 (DAC)。本章介绍了 DAC 模块的运行情况。

14.1 DAC 简介

DAC 模块是一个 12 位电压输出 DAC。DAC 可配置为 8 位或 12 位分辨率设置，并可与 DMA 控制器结合使用。

DAC 的特性包括：

- 8 位或 12 位电压输出分辨率
- 直接二进制或二进制补码数据格式
- 用于生成预定义采样率的内部采样时间发生器
- 事件结构中用于 D/A 转换的两个硬件触发
- 用于数据流速控制的 4x12 位内部 FIFO
- 在没有 CPU 干预的情况下使用 DMA 控制器进行操作
- 可选择的电压基准选项
- 输出内部连接到 OPA、ADC 和 COMP 等模拟模块
- 针对偏移误差校正的自校准选项

图 14-1 展示了 DAC 模块的方框图。

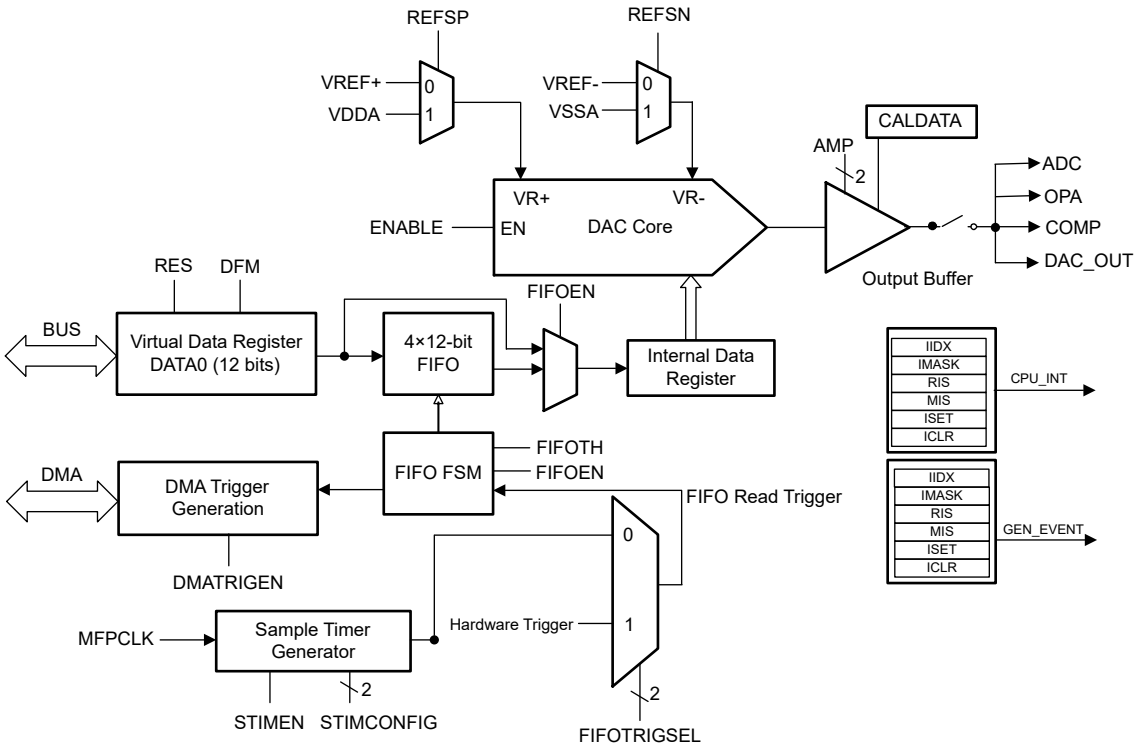


图 14-1. DAC 方框图

14.2 DAC 运行

DAC 模块可由用户软件配置。以下各节讨论了 DAC 的设置和操作。

14.2.1 DAC 内核

可以通过 CTL0.RES 位将 DAC 配置为以 8 位或 12 位的分辨率运行。CTL0.DFM 位可用于选择直接二进制或二进制补码的数据格式。使用直接二进制数据格式时，输出电压公式如表 14-1 所述。

表 14-1. DAC 满标量程

分辨率	CTL0.RES	输出电压公式
12 位	1	$V_{out} = V_{ref} \times (DATA\_VALUE/4096)$
8 位	0	$V_{out} = V_{ref} \times (DATA\_VALUE/256)$

- 在 8 位分辨率设置中，DATA0 寄存器中 DATA_VALUE 的最大可用值为 255 (0xFF)。
- 在 12 位分辨率设置中，DATA0 寄存器中 DATA_VALUE 的最大可用值为 4095 (0xFFF)。

DAC 可通过设置 CTL0.ENABLE 位来启用。当 DAC 被启用时，DAC 内核和输出缓冲器开始稳定，并生成一次模块就绪中断条件 (MODRDYIFG)，表示 DAC 随时可用。

备注

在应用程序中，必须在配置所有控制寄存器之后再启用 DAC。如果在 DAC 运行时需要更改配置，则必须先禁用 DAC，将新配置编程到控制寄存器后，再重新启用 DAC。DAC 运行期间，控制寄存器的任何变化都会导致不可预测的结果。

14.2.2 DAC 输出

可以通过设置 CTL1.OPS 位将 DAC 输出连接到内部模拟模块 OPA、ADC 与 COMP 或 DAC_OUT 引脚。CTL1.OPS 位的复位值为 0。存在以下情况：

- 如果 CTL1.OPS 位被置位，DAC 输出会连接至 OPA、ADC、COMP 和 DAC_OUT。
- 如果 CTL1.OPS 位未置位，则没有 DAC 输出。引脚 DAC_OUT 可以连接到 ADC、OPA 或 COMP。

备注

当引脚 DAC_OUT 配置为 ADC、OPA 或 COMP 的输入时，CTL1.OPS 位必须设置为 0。

14.2.3 DAC 电压基准

CTL1.REFSP 和 CTL1.REFSN 位从以下选项中选择 DAC 的电压基准。

- 模拟电源 (VDDA) (请参阅 MSPM0Gxx PMU 方框图) 。
- 可在器件的 VREF+ 和 VREF- 引脚上提供的外部基准电压。
- VREF 模块输出的内部基准电压。

14.2.4 DAC 输出缓冲器

DAC 的电压输出缓冲器可以通过设置 CTL1.AMPEN 位来配置为启用或禁用。禁用时，可以使用 CTL1.AMPHIZ 位将输出放大器设置为接地或高阻抗。

14.2.5 DAC 数据格式

DAC 支持直接二进制和二进制补码数据格式。使用直接二进制数据格式时，满量程输出值在 12 位分辨率设置时为 0xFFF，在 8 位分辨率设置时为 0xFF (请参阅图 14-2) 。

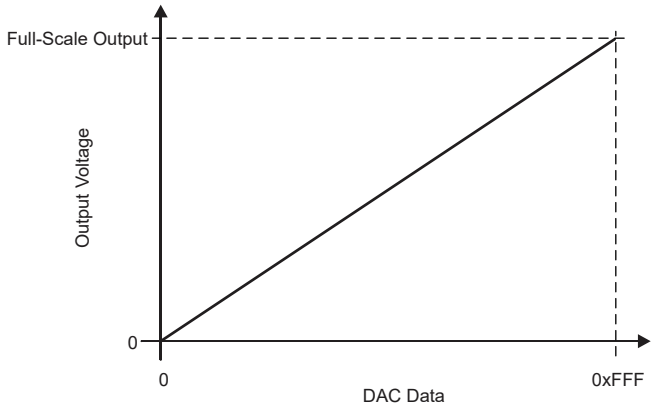


图 14-2. 12 位直接二进制模式下输出电压与 DAC 数据的关系

使用二进制补码数据格式时，范围会发生偏移，使得 12 位分辨率设置时 DAC 数据值 0x800 和 8 位分辨率设置时 DAC 数据值 0x80 会产生零输出电压，0x0 是中标度输出电压，12 位分辨率设置时 0x7FF 和 8 位分辨率设置时 0x7F 是满标度电压输出（请参阅图 14-3）。

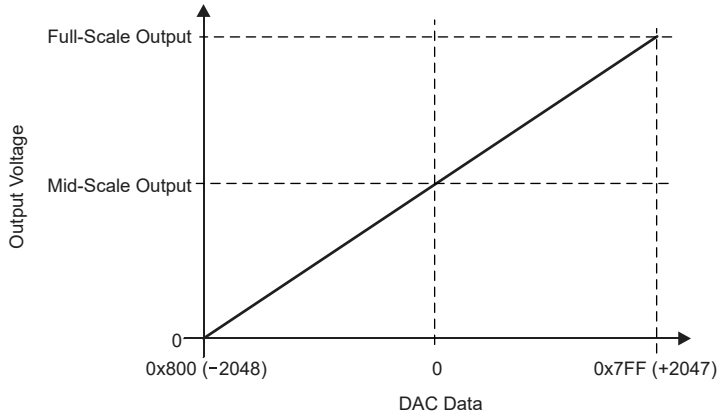


图 14-3. 在 12 位二进制补码模式时输出电压与 DAC 数据的关系

14.2.6 采样时间发生器

DAC 模块中提供了一个采样时间发生器。它可用于为某些预定义的采样率生成 DAC 转换触发。采样时间发生器默认为禁用状态，可通过设置 CTL3.STIMEN 位来启用。启用采样时间发生器时，它会自动请求 MFPCLK (4MHz) 并根据编程的触发速率生成触发。支持的预定义采样率为 500sps、1Ksps、2Ksps、4Ksps、8Ksps、16Ksps、100Ksps、200Ksps、500Ksps 和 1Msps。这些触发速率通过设置位 CTL3.STIMCONFIG 进行编程。表 14-2 中给出了预定义的采样率

表 14-2. 预定义的采样率

CTL3.STIMCONFIG	采样率
0	500sps
1	1Ksps
2	2Ksps
3	4Ksps
4	8Ksps
5	16Ksps
6	100Ksps
7	200Ksps

表 14-2. 预定义的采样率 (continued)

CTL3.STIMCONFIG	采样率
8	500Ksps
9	1Msps

备注

启用采样时间发生器时，总线时钟必须等于 MFPCLK，否则 FIFO 中的样本可能会丢失。

14.2.7 DAC FIFO 结构

DAC 模块具有一个 4x12 位的 FIFO。默认情况下，禁用 FIFO 操作，可以通过设置 CTL2.FIFOEN 位来启用该操作。当 CPU 或 DMA 控制器将数据写入内存映射的数据寄存器 DATA0 时，数据将被写入到写指针指向的位置处的 FIFO。当 FIFO 读取触发有效时，数据将从 FIFO 中读取并加载到内部 DAC 数据寄存器中。写入 FIFO 的数据可以是二进制或二进制补码格式。

14.2.7.1 将数据从 FIFO 加载到内部 DAC 数据寄存器

CTL2.FIFOTRIGSEL 寄存器用于选择将数据从 FIFO 加载到内部 DAC 数据寄存器的触发源。触发源有三种：

- 可以选择采样时间发生器输出作为触发源。
- 硬件触发器，侦听通过寄存器 FSUB_0 中配置的 GEN_EVENT 发布的事件。

对于硬件触发源，DAC 模块可以通过事件结构订阅事件。将 CTR2.FIFOTRIGSEL 设置为 1 是为了订阅 FSUB_0 上的事件。

当使用采样时间发生器时，MFPCLK 用于生成触发信号并将数据从 FIFO 传输到内部 DAC 数据寄存器。在这种情况下，DAC 采样率是可预测的，并且与 ULPCLK 无关。

当不使用采样时间发生器时，ULPCLK 用于在外部事件触发时将数据从 FIFO 传输到内部 DAC 数据寄存器。为了实现可预测的 DAC 采样率，应用软件必须管理 ULPCLK 频率，使其具有确定性。在这种情况下，如果在 DAC 运行期间 ULPCLK 频率发生任何变化，则采样率会受到影响。

只要选定的触发源被置为有效，读取指针指向的 FIFO 中的数据就会被读取并加载到内部 DAC 数据寄存器中进行转换。

14.2.8 通过 DMA 控制器控制 DAC 运行

DMA 控制器可以将数据从 SRAM 移入 DAC。对 CTL2.DMATRIGEN 和 CTL2.FIFOEN 位进行编程可以启用通过 DMA 控制器控制 DAC 运行。当 DAC 中同时启用了 DMA 触发机制与 FIFO 时，FIFO 硬件状态机会评估 FIFO 中可用的空位置并生成 DMA 触发信号。FIFO 阈值深度可通过 CTL2.FIFOTH 位进行选择。可用的 FIFO 深度设置为空、1/4、1/2 和 3/4。

在运行期间，FIFO 中的空位置由 FIFO 硬件状态机持续评估，并与所选的 FIFO 阈值深度进行比较。当空位置的数量与预设的 FIFO 阈值相匹配时，DAC 会生成一个 DMA 触发。

DAC 至 DMA 控制器接口如图 14-4 所示，后续部分会进行相应说明。

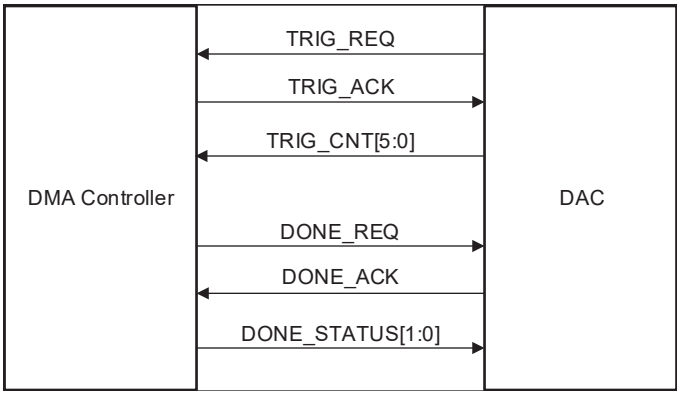


图 14-4. 连接 DMA 控制器的 DAC 接口

14.2.8.1 DMA 触发接口

DAC 和 DMA 之间有一个定义的协议：

1. DAC 向 DMA 生成触发 TRIG_REQ 以及触发计数 TRIG_CNT。TRIG_CNT 是一个 6 位信号，指示 DAC FIFO 中可供 DMA 填充的空位置数量。TRIG_CNT 的值与 FIFO 元素大小一致。
2. 当 TRIG_REQ 置为有效时，DMA 将触发确认信号 TRIG_ACK 发送回 DAC。

触发事件和 TRIG_ACK 通过事件结构进行路由，并遵循标准的 4 次请求、确认握手协议。

14.2.8.2 DMA 状态接口

当 DAC 触发被 DMA 捕获并提供了触发确认后，DAC 和 DMA 都处于触发状态。在这种状态下，DAC 不能生成新的 DMA 触发。当 DMA 开始为 DAC 提供服务时，DMA 可以根据触发计数指示的实际值或小于触发计数的值进行数据传输（也可能没有传输）。这是不确定的，具体取决于 DMA 带宽和通道优先级等因素。

当 DMA 执行一定数量的传输时，它会向 DAC 断言 DMA 完成信号 DONE_REQ，同时携带 DMA 状态侧带信号 DONE_STATUS。DAC 模块提供 DMA 完成确认 DONE_ACK 作为响应。DMA 完成事件和 DONE_ACK 通过事件结构进行路由，并遵循标准的 4 次请求、确认握手协议。

14.2.8.3 DMA 触发生成方案

DMA 状态信号 DONE_STATUS 值 0 表示 DMA 有更多数据元素要传输到 DAC。当 DAC 观察到 DONE_STATUS 为 0 时，它会对 FIFO 中的空位置数量执行新的评估，如果与编程的 FIFO 阈值级别匹配，它会生成下一个 TRIG_REQ 和 TRIG_CNT。然后重复上面提到的整个操作序列。

当 DMA 传输完全部数据时，它会将 DMA 完成信号 DONE_REQ 以及具有非零值的 DMA 状态信号 DONE_STATUS 置为有效。当 DAC 捕获非零值的 DMA 状态信号时，它会设置 DMADONE 中断标志。该中断标志可用于向拥有 DAC 的 CPU 生成中断，以便采取适当的操作。CTL2.DMATRIGEN 位应由软件清零以停止进一步的 DMA 触发。该中断标志可用于向拥有 DAC 的 CPU 生成中断，以便采取适当的操作。

以下部分说明了在启用 FIFO 的情况下使用 DAC 和 DMA 的详细信息

DAC-DMA 在 FIFO 模式下运行 (CTL2.FIFOEN=1) 并启用采样时间发生器

- 配置时钟，应启用 MFPCLK，因为将使用采样时间发生器
- 配置 DMA 源地址和目标地址。目标地址为 FIFO 数据寄存器 DATA0
- 选择传输模式并在字节（8 位）、短字（16 位）、字（32 位）或长字（64 位）之间选择 DMA 传输大小
- 设置 DAC 分辨率（8 位或 12 位）
- 通过支持 ¼、½ 和 ¾ 的 CTL2.FIFOTH 位配置 FIFO 触发阈值
- 配置并启用 DAC 和 DMA 中断
- 使用 CTL2.FIFOTRIGSEL 将采样时间发生器配置为将数据加载到 FIFO 数据寄存器的触发源。确保正确设置 MCLK 和 ULPCLK 的时钟配置，以避免 FIFO 欠运转问题和 DAC 输出失真

  - 确保 ULPCLK 等于 MFPCLK 以保持时钟域之间的同步
  - 当 MCLK 来源为 HFCLK 或 SYSPLL，并且其频率等于或大于 32MHz 时，将 FIFO 阈值设置调整为 ½ 或 ¾
- 启用 DMA 触发

14.2.9 采用 CPU 的 DAC 操作

CPU 可以直接将数据加载到 DAC 中进行转换，具体情况如下：

- **固定直流电压输出且 FIFO 禁用：**CPU 将数据写入数据寄存器 DATA0。该数据将传递到内部 DAC 数据寄存器进行转换。
- **固定直流电压输出且 FIFO 启用：**CPU 将数据写入数据寄存器 DATA0，该寄存器从 FIFO 中读取并在 FIFO 读取触发有效时加载到内部 DAC 数据寄存器中。
- **固定交流电压输出且 FIFO 启用：**CPU 将数据写入数据寄存器 DATA0，该寄存器从 FIFO 中读取并在 FIFO 读取触发有效时加载到内部 DAC 数据寄存器中。

备注

1. 如果需要固定交流电压输出，则必须启用 FIFO。
2. 对于 8 位 DAC 分辨率，CPU 必须通过字节 0 将数据写入数据寄存器 DATA0；对于 12 位 DAC 分辨率，CPU 必须通过低半字（字节 0 和 1）将数据写入数据寄存器 DATA0。
3. 当 CPU 将数据载入到 DAC 中时，DMA 触发生成机制必须保持禁用。

对于 FIFO 读取触发，可以使用采样时间发生器或可用于事件结构的外部事件。

14.2.9.1 DAC 与 CPU 交互的中断条件

当 FIFO 处于启用状态并使用 CPU 将数据载入到 DAC 中时，DAC 中会产生某些特定于 FIFO 的中断条件，具体是 FIFO 空、FIFO 满、FIFO 1/4 空、FIFO 1/2 空和 FIFO 3/4 空。所有这些中断标志的复位值为 0。当 DAC 模块与 FIFO 一起启用时，内部逻辑评估 FIFO 状态并相应地更新特定于 FIFO 的中断标志。这些中断条件可由软件适当启用，这样，就可以通知 CPU 在 FIFO 中有可用于将数据载入 DAC 的空位置。

14.2.10 数据寄存器格式

DAC 支持可编程分辨率设置和数据格式。本节给出的图以 DATA0 寄存器为例，描述了 CPU 或 DMA 控制器将以何种形式将数据写入 DATA0 寄存器进行一次数据采样。

CPU 始终只写入 DATA0 寄存器。如果 FIFO 被禁用，CPU 只能写入字节 0 来获得 8 位 DAC 分辨率或低半字（字节 0 和 1）来获得 12 位 DAC 分辨率。DATA0 寄存器的上半字不会被写入。

当 FIFO 被启动时，CPU 最多可以向 DATA0 寄存器写入 4 个数据样本来获得 8 位或 12 位分辨率。

图 14-5 显示了 DATA0 寄存器中 8 位二进制或二进制补码数据的格式。

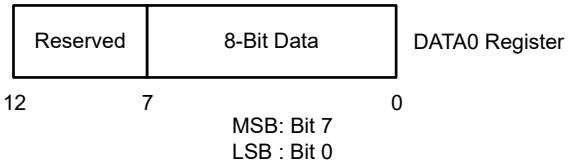


图 14-5. 8 位数据

14.2.11 DAC 输出放大器失调电压校准

DAC 输出放大器的失调电压可以是正的，也可以是负的。当失调电压为负时，输出放大器试图驱动负电压，但不能成功。输出电压保持在零，直到 DAC 的数字输入产生足够的正输出电压来克服负失调电压，从而形成图 14-6 中的传输函数。

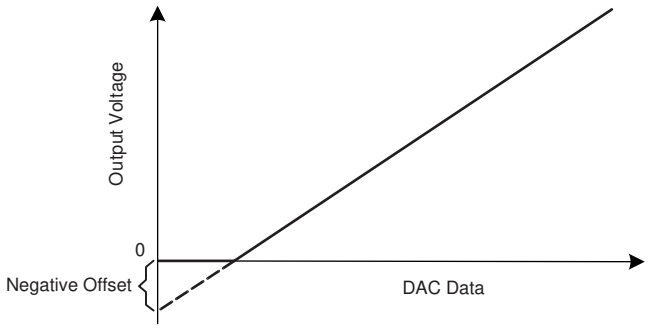


图 14-6. 负失调电压

当输出放大器有一个正失调电压时，一个值为 0 的数字输入不会导致输出电压为零。DAC 输出电压在 DAC 数据达到最大代码之前就达到最大输出电平（请参阅图 14-7）。

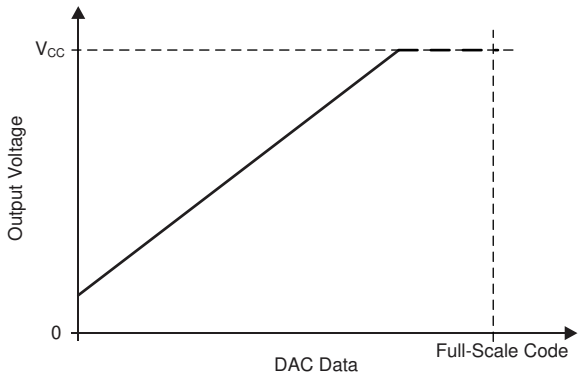


图 14-7. 正失调电压

DAC 可以自动校准输出放大器的失调电压。应用程序可以执行 DAC 输出缓冲区自动校准，从而达到电气规格中给出的偏移量误差。通过 CALCTL.CALON 位来启动偏移量误差校准。在校准期间，DAC 输出为高阻抗状态。校准完成后，CALCTL.CALON 位自动复位。校准前，软件必须配置 CTL1.AMP 位。

为了获得最佳的校准结果，在校准期间应尽量减少器件引脚和 CPU 中的活动。校准数据被加载到 CALDATA 寄存器中；当通过 CALCTL.CALON 位启动 DAC 自校准时，CALDATA 寄存器在校准过程中将持续更新。只有在 CALCTL.CALON 位被清零后才能读取 CALDATA 寄存器，否则可能读取到不正确的值。DAC 校准数据为二进制补码格式。只使用了低字节，高字节对偏移量校准无影响。

备注

在使用 DAC 时，如果更改了 VREF 模块中的基准电压电平，请重新校准 DAC 输出缓冲器。

14.2.12 中断和事件支持

DAC 模块包含三个事件发布者和两个事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理到 CPU 子系统的 DAC 中断请求 (IRQ)。第二个事件发布者 (GEN_EVENT) 可通过通用路由设置通用事件发布者。直接 DMA 触发可用作 DAC 到 DMA 的触发，通过 DMA 事件路由将 DAC 事件发送到 DMA。

事件订阅者 (FSUB_0) 可用于订阅通过通用事件路由通道发布到事件结构的事件。

表 14-1 中总结了 DAC 事件。

表 14-508. DAC 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	DAC	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 DAC 到 CPU 的中断路由
通用发布者事件	发布者	DAC	其他外设	通用路由 (FPUB_0)	GEN_EVENT 和 FPUB_0 寄存器	从 DAC 触发通用事件通道
DMA 触发事件	发布者	DAC	DMA	DMA 路由	DMA_TRIG 和 FPUB_1 寄存器	修复了从 ADC 到 DMA 的触发路由
通用订阅者事件	订阅者	其他外设	DAC	通用路由 (FSUB_0)	FSUB_0	通用事件通道的 DAC 订阅

14.2.12.1 CPU 中断事件发布者 (CPU_INT)

DAC 模块提供多个不同的中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，DAC 的 CPU 中断事件为：

表 14-509. DAC 中断事件条件 (CPU_INT)

索引 (IIDX)	名称	说明
0x0	NO_INTR	未置位 (IIDX.STAT = 0) 意味着没有挂起的中断请求
0x2	MODRDYIFG	DAC 输出在使用 CTL0 寄存器中的 ENABLE 开启后稳定
0x9	FIFOFULLIFG	FIFO 已满时，设置 FIFO 已满中断标志
0xA	FIFO1B4IFG	FIFO 四分之一位置为空时，设置 FIFO 四分之一空中断标志。
0xB	FIFO1B2IFG	FIFO 一半位置为空时，设置 FIFO 半空中断标志。
0xC	FIFO3B4IFG	FIFO 中所有位置都为空时，设置 FIFO 四分之三空中断标志。
0xD	FIFOEMPTYIFG	FIFO 中所有数据都被移出时，设置 FIFO 空中断标志。
0xE	FIFOURUNIFG	FIFO 为空且 FIFO 读取触发有效时，设置 FIFO 欠运转标志。
0xF	DMADONEIFG	当编程块大小的 DMA 数据传输完成时，设置 DMA 完成中断标志

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。在软件读取 IIDX 寄存器或写入相应的 ICLR 寄存器位时，会清除中断 (RIS) 标志。

有关为 CPU 中断配置事件寄存器的指导，请参阅 *节 7.2.5* 部分。

14.2.12.2 通用事件发布者 (GEN_EVENT)

DAC 模块提供如下中断源，其中一个中断源可配置为将事件发布到通用事件路由通道。

表 14-510. DAC 通用事件发布者条件 (GEN_EVENT)

索引 (IIDX)	名称	说明
0x0	NO_INTR	未置位 (IIDX.STAT = 0) 意味着没有挂起的中断请求
0x2	MODRDYIFG	DAC 被启用且 DAC 输出稳定时的模块就绪中断
0x9	FIFOFULLIFG	FIFO 已满时，设置 FIFO 已满中断标志
0xA	FIFO1B4IFG	FIFO 四分之一位置为空时，设置 FIFO 四分之一空中断标志。
0xB	FIFO1B2IFG	FIFO 一半位置为空时，设置 FIFO 半空中断标志。
0xC	FIFO3B4IFG	FIFO 中所有位置都为空时，设置 FIFO 四分之三空中断标志。
0xD	FIFOEMPTYIFG	FIFO 中所有数据都被移出时，设置 FIFO 空中断标志。
0xE	FIFOURUNIFG	FIFO 为空且 FIFO 读取触发有效时，设置 FIFO 欠运转标志。

表 14-510. DAC 通用事件发布者条件 (GEN_EVENT) (continued)

索引 (IIDX)	名称	说明
0xF	DMADONEIFG	当编程块大小的 DMA 数据传输完成时，设置 DMA 完成中断标志

通用事件发布者配置通过 GEN_EVENT 事件管理寄存器进行管理。根据通过事件结构接收到的来自用户模块的确认 (ACK) 信号清除中断 (RIS) 标志。有关为通用事件发布者配置事件寄存器的指导，请参阅 [使用事件寄存器]。必须通过将目标通用通道 ID 写入 DAC 中的 **FPUB_0** 寄存器来选择将 GEN_EVENT 发布到的通用事件通道。有关配置通用事件路由的指导，请参阅 [通用事件路由]。如果应用中未使用该发布者，则 FPUB_0 寄存器可以保持断开状态（设置为零），并且不应通过 ADC GEN_EVENT 寄存器集中的 MIS 寄存器解除屏蔽任何事件。

14.2.12.3 DMA 触发事件发布者

DAC 模块提供了可配置为 DMA 触发源的不同中断源。为了降低中断优先级，下表中列出了来自 DAC 的 DMA 触发事件。当 DAC 需要 DMA 通道时，应在 IMASK 寄存器中取消屏蔽 DMA 触发。

备注

来自 DAC 的 DMA 触发属于硬件的直接触发，不属于可配置事件管理寄存器的 DMA_TRIG 组。

要配置 DMA 控制器从 DAC 触发，请参阅节 14.2.8。

表 14-6. DAC DMA 触发事件发布者条件

索引 (IIDX)	名称	说明
0x0	NO_INTR	未置位 (IIDX.STAT = 0) 意味着没有挂起的中断请求
0x2	MODRDYIFG	DAC 被启用且 DAC 输出稳定时的模块就绪中断
0x9	FIFOFULLIFG	FIFO 已满时，设置 FIFO 已满中断标志
0xA	FIFO1B4IFG	FIFO 四分之一位置为空时，设置 FIFO 四分之一空中断标志。
0xB	FIFO1B2IFG	FIFO 一半位置为空时，设置 FIFO 半空中断标志。
0xC	FIFO3B4IFG	FIFO 中所有位置都为空时，设置 FIFO 四分之三空中断标志。
0xD	FIFOEMPTYIFG	FIFO 中所有数据都被移出时，设置 FIFO 空中断标志。
0xE	FIFOURUNIFG	FIFO 为空且 FIFO 读取触发有效时，设置 FIFO 欠运转标志。
0xF	DMADONEIFG	当编程块大小的 DMA 数据传输完成时，设置 DMA 完成中断标志

14.2.12.4 通用事件订阅者 (FSUB_0)

DAC 模块可以通过从其他外设接收通过通用通道路由的事件来触发 FIFO 传输。当接收到该事件时，FIFO 中的数据会加载到内部 DAC 数据寄存器中。有关通用事件路由工作方式的详细信息，请参阅通用事件路由和外设间事件。确定要使用的通道并且已知所连接外设的发布者端口和订阅者端口后，请使用以下步骤建立事件连接。

此示例中将配置一个由 GPIO 触发的 DAC FIFO 传输，使用 GPIO 端口 A 将事件发布到通用通道 1，并由 DAC 订阅通用通道 1 作为转换开始触发器。

1. 配置 GPIO 端口 A 的 GEN_EVENT 寄存器，根据相应的事件（例如 DIN 上升事件）设置事件请求。
2. 将 0x1 存储到 GPIO 端口 A 的 FPUB_0 寄存器中，以便将 GEN_EVENT 寄存器选择的 GPIO 事件发布到通用路由通道 1。通道 1 不能正在被另一个外设使用。
3. 将 0x1 存储到 DAC 的 FSUB_0 寄存器中，以便 DAC 侦听计时器发布到通道 1 的事件。
4. 根据将数据从 FIFO 加载到内部 DAC 数据寄存器中的配置说明，将 DAC 配置为从订阅者端口进行触发。
5. 配置并启用相应的 GPIO 引脚以监控输入电压事件。

14.3 DAC12 寄存器

表 14-7 列出了 DAC12 寄存器的存储器映射寄存器。表 14-7 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 14-7. DAC12 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者端口 0		转到
444h	FPUB_1	发布者端口 1		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT	转到
1058h	IMASK	中断屏蔽	GEN_EVENT	转到
1060h	RIS	原始中断状态	GEN_EVENT	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT	转到
1070h	ISET	中断设置	GEN_EVENT	转到
1078h	ICLR	中断清除	GEN_EVENT	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	CTL0	控制 0		转到
1110h	CTL1	控制 1		转到
1120h	CTL2	控制 2		转到
1130h	CTL3	控制 3		转到
1140h	CALCTL	校准控制		转到
1160h	CALDATA	校准数据		转到
1200h	DATA0	数据 0		转到

复杂的位访问类型经过编码可适应小型表单元。表 14-8 显示了适用于此部分中访问类型的代码。

表 14-8. DAC12 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
RH	R H	读取 由硬件置位或清除
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		

表 14-8. DAC12 访问类型代码 (continued)

访问类型	代码	说明
-n		复位后的值或默认值

14.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 14-8 展示了 FSUB_0，表 14-9 中对此进行了介绍。

返回到汇总表。

订阅者端口 0

图 14-8. FSUB_0

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

表 14-9. FSUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 others = 已连接至 channel_ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

14.3.2 FPUB_1（偏移 = 444h）[复位 = 00000000h]

图 14-9 展示了 FPUB_1，表 14-10 中对此进行了介绍。

返回到汇总表。

发布者端口 1

图 14-9. FPUB_1

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

表 14-10. FPUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 others = 已连接至 channel_ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

14.3.3 PWREN（偏移 = 800h）[复位 = 00000000h]

图 14-10 展示了 PWREN，表 14-11 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 14-10. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-							K-0h

表 14-11. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

14.3.4 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 14-11 展示了 RSTCTL，表 14-12 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 14-11. RSTCTL

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

表 14-12. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

14.3.5 STAT（偏移 = 814h）[复位 = 00000000h]

图 14-12 展示了 STAT，表 14-13 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 14-12. STAT

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

表 14-13. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

14.3.6 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 14-13 展示了 IIDX，表 14-14 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。或者，用户可以使用其他显示所有已发生中断的寄存器来实现自己的优先级方案。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 14-13. IIDX

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

表 14-14. IIDX 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 2h = 模块就绪中断 9h = FIFO 已满中断 Ah = FIFO 四分之一空中断 Bh = FIFO 半空中断 Ch = FIFO 四分之三空中断 Dh = FIFO 空中断 Eh = FIFO 欠运转中断 Fh = DMA 完成中断

14.3.7 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 14-14 展示了 IMASK，表 14-15 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 14-14. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R/W-0h						R/W-0h	R/W-0h

表 14-15. IMASK 字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14	DMADONEIFG	R/W	0h	屏蔽 DMADONEIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
13	FIFOURUNIFG	R/W	0h	屏蔽 FIFOURUNIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
12	FIFOEMPTYIFG	R/W	0h	屏蔽 FIFOEMPTYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
11	FIFO3B4IFG	R/W	0h	屏蔽 FIFO3B4IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
10	FIFO1B2IFG	R/W	0h	屏蔽 FIFO1B2IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
9	FIFO1B4IFG	R/W	0h	屏蔽 FIFO1B4IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
8	FIFOFULLIFG	R/W	0h	屏蔽 FIFOFULLIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
7-2	RESERVED	R/W	0h	
1	MODRDYIFG	R/W	0h	屏蔽 MODRDYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置

表 14-15. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	R/W	0h	

14.3.8 RIS（偏移 = 1030h）[复位 = 00001E00h]

图 14-15 展示了 RIS，表 14-16 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 14-15. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R-0h	R-0h	R-0h	R-1h	R-1h	R-1h	R-1h	R-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R-0h						R-0h	R-0h

表 14-16. RIS 字段说明

位	字段	类型	复位	说明
31-15	保留	R	0h	
14	DMADONEIFG	R	0h	DMADONEIFG 的原始中断状态 0h = 未发生 DMA 完成情况 1h = 已发生 DMA 完成情况
13	FIFOURUNIFG	R	0h	FIFOURUNIFG 的原始中断状态 0h = 未发生 FIFO 欠运转情况 1h = 已发生 FIFO 欠运转情况
12	FIFOEMPTYIFG	R	1h	FIFOEMPTYIFG 的原始中断状态 0h = 未发生 FIFO 空情况 1h = 已发生 FIFO 空情况
11	FIFO3B4IFG	R	1h	FIFO3B4IFG 的原始中断状态 0h = 未发生 FIFO 四分之三空情况 1h = 已发生 FIFO 四分之三空情况
10	FIFO1B2IFG	R	1h	FIFO1B2IFG 的原始中断状态 0h = 未发生 FIFO 半空情况 1h = 已发生 FIFO 半空情况
9	FIFO1B4IFG	R	1h	FIFO1B4IFG 的原始中断状态 0h = 未发生 FIFO 四分之一空情况 1h = 已发生 FIFO 四分之一空情况
8	FIFOFULLIFG	R	0h	FIFOFULLIFG 的原始中断状态 0h = 未发生 FIFO 已满情况 1h = 已发生 FIFO 已满情况
7-2	RESERVED	R	0h	
1	MODRDYIFG	R	0h	MODRDYIFG 的原始中断状态 0h = 未发生 DAC 模块就绪事件 1h = 已发生 DAC 模块就绪事件

表 14-16. RIS 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	R	0h	

14.3.9 MIS（偏移 = 1038h）[复位 = 00000000h]

图 14-16 展示了 MIS，表 14-17 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 14-16. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R-0h						R-0h	R-0h

表 14-17. MIS 字段说明

位	字段	类型	复位	说明
31-15	保留	R	0h	
14	DMADONEIFG	R	0h	DMADONEIFG 的屏蔽中断状态 0h = DMADONEIFG 不请求中断服务例程 1h = DMADONEIFG 请求一个中断服务例程
13	FIFOURUNIFG	R	0h	FIFOURUNIFG 的屏蔽中断状态 0h = FIFOURUNIFG 不请求中断服务例程 1h = FIFOURUNIFG 请求一个中断服务例程
12	FIFOEMPTYIFG	R	0h	FIFOEMPTYIFG 的屏蔽中断状态 0h = FIFOEMPTYIFG 不请求中断服务例程 1h = FIFOEMPTYIFG 请求一个中断服务例程
11	FIFO3B4IFG	R	0h	FIFO3B4IFG 的屏蔽中断状态 0h = FIFO3B4IFG 不请求中断服务例程 1h = FIFO3B4IFG 请求一个中断服务例程
10	FIFO1B2IFG	R	0h	FIFO1B2IFG 的屏蔽中断状态 0h = FIFO1B2IFG 不请求中断服务例程 1h = FIFO1B2IFG 请求一个中断服务例程
9	FIFO1B4IFG	R	0h	FIFO1B4IFG 的屏蔽中断状态 0h = FIFO1B4IFG 不请求中断服务例程 1h = FIFO1B4IFG 请求一个中断服务例程
8	FIFOFULLIFG	R	0h	FIFOFULLIFG 的屏蔽中断状态 0h = FIFOFULLIFG 不请求中断服务例程 1h = FIFOFULLIFG 请求一个中断服务例程
7-2	RESERVED	R	0h	
1	MODRDYIFG	R	0h	MODRDYIFG 的屏蔽中断状态 0h = MODRDYIFG 不请求中断服务例程 1h = MODRDYIFG 请求一个中断服务例程
0	RESERVED	R	0h	

14.3.10 ISET (偏移 = 1040h) [复位 = 00000000h]

图 14-17 展示了 ISET，表 14-18 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 14-17. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	保留
W-0h						W-0h	W-0h

表 14-18. ISET 字段说明

位	字段	类型	复位	说明
31-15	保留	W	0h	
14	DMADONEIFG	W	0h	设置 RIS 寄存器中的 DMADONEIFG 0h = 写入 0 不产生影响 1h = 对应于 DMADONEIFG 的 RIS 位被设置
13	FIFOURUNIFG	W	0h	设置 RIS 寄存器中的 FIFOURUNIFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFOURUNIFG 的 RIS 位会被设置
12	FIFOEMPTYIFG	W	0h	设置 RIS 寄存器中的 FIFOEMPTYIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOEMPTYIFG 的 RIS 位被设置
11	FIFO3B4IFG	W	0h	设置 RIS 寄存器中的 FIFO3B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO3B4IFG 的 RIS 位会被设置
10	FIFO1B2IFG	W	0h	设置 RIS 寄存器中的 FIFO1B2IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B2IFG 的 RIS 位会被设置
9	FIFO1B4IFG	W	0h	设置 RIS 寄存器中的 FIFO1B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B4IFG 的 RIS 位会被设置
8	FIFOFULLIFG	W	0h	设置 RIS 寄存器中的 FIFOFULLIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOFULLIFG 的 RIS 位被设置
7-2	RESERVED	W	0h	
1	MODRDYIFG	W	0h	设置 RIS 寄存器中的 MODRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 MODRDYIFG 的 RIS 位被设置

表 14-18. ISET 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	W	0h	

14.3.11 ICLR (偏移 = 1048h) [复位 = 00000000h]

图 14-18 展示了 ICLR，表 14-19 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 14-18. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	保留
W-0h						W-0h	W-0h

表 14-19. ICLR 字段说明

位	字段	类型	复位	说明
31-15	保留	W	0h	
14	DMADONEIFG	W	0h	清除 RIS 寄存器中的 DMADONEIFG 0h = 写入 0 不产生影响 1h = 对应于 DMADONEIFG 的 RIS 位被清零
13	FIFOURUNIFG	W	0h	清除 RIS 寄存器中的 FIFOURUNIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOURUNIFG 的 RIS 位被清零
12	FIFOEMPTYIFG	W	0h	清除 RIS 寄存器中的 FIFOEMPTYIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOEMPTYIFG 的 RIS 位被清零
11	FIFO3B4IFG	W	0h	清除 RIS 寄存器中的 FIFO3B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO3B4IFG 的 RIS 位被清零
10	FIFO1B2IFG	W	0h	清除 RIS 寄存器中的 FIFO1B2IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B2IFG 的 RIS 位被清零
9	FIFO1B4IFG	W	0h	清除 RIS 寄存器中的 FIFO1B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B4IFG 的 RIS 位被清零
8	FIFOFULLIFG	W	0h	清除 RIS 寄存器中的 FIFOFULLIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOFULLIFG 的 RIS 位被清零
7-2	RESERVED	W	0h	
1	MODRDYIFG	W	0h	清除 RIS 寄存器中的 MODRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 MODRDYIFG 的 RIS 位被清零
0	RESERVED	W	0h	

14.3.12 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 14-19 展示了 IIDX，表 14-20 中对此进行了介绍。

返回到汇总表。

中断索引寄存器。该只读寄存器提供最高优先级的挂起中断的中断索引。它还指示是否没有中断挂起。优先级顺序是固定的：索引越小，优先级越高。或者，用户可以使用其他显示所有已发生中断的寄存器来实现自己的优先级方案。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（而不是从调试接口）读取后，必须使用第二高优先级中断更新该寄存器，或该寄存器必须指示没有挂起的中断。仅指示通过 IMASK 选择的中断。

图 14-19. IIDX

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

表 14-20. IIDX 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3-0	STAT	R	0h	中断索引状态 0h = 无挂起中断 2h = 模块就绪中断 9h = FIFO 已满中断 Ah = FIFO 四分之一空中断 Bh = FIFO 半空中断 Ch = FIFO 四分之三空中断 Dh = FIFO 空中断 Eh = FIFO 欠运转中断 Fh = DMA 完成中断

14.3.13 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 14-20 展示了 IMASK，表 14-21 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 14-20. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R/W-0h						R/W-0h	R/W-0h

表 14-21. IMASK 字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14	DMADONEIFG	R/W	0h	屏蔽 DMADONEIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
13	FIFOURUNIFG	R/W	0h	屏蔽 FIFOURUNIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
12	FIFOEMPTYIFG	R/W	0h	屏蔽 FIFOEMPTYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
11	FIFO3B4IFG	R/W	0h	屏蔽 FIFO3B4IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
10	FIFO1B2IFG	R/W	0h	屏蔽 FIFO1B2IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
9	FIFO1B4IFG	R/W	0h	屏蔽 FIFO1B4IFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
8	FIFOFULLIFG	R/W	0h	屏蔽 FIFOFULLIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置
7-2	RESERVED	R/W	0h	
1	MODRDYIFG	R/W	0h	屏蔽 MODRDYIFG 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且 MIS 中的相应位将被设置

表 14-21. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	R/W	0h	

14.3.14 RIS（偏移 = 1060h）[复位 = 00001E00h]

图 14-21 展示了 RIS，表 14-22 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 14-21. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R-0h	R-0h	R-0h	R-1h	R-1h	R-1h	R-1h	R-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R-0h						R-0h	R-0h

表 14-22. RIS 字段说明

位	字段	类型	复位	说明
31-15	保留	R	0h	
14	DMADONEIFG	R	0h	DMADONEIFG 的原始中断状态 0h = 未发生 DMA 完成情况 1h = 已发生 DMA 完成情况
13	FIFOURUNIFG	R	0h	FIFOURUNIFG 的原始中断状态 0h = 未发生 FIFO 欠运转情况 1h = 已发生 FIFO 欠运转情况
12	FIFOEMPTYIFG	R	1h	FIFOEMPTYIFG 的原始中断状态 0h = 未发生 FIFO 空情况 1h = 已发生 FIFO 空情况
11	FIFO3B4IFG	R	1h	FIFO3B4IFG 的原始中断状态 0h = 未发生 FIFO 四分之三空情况 1h = 已发生 FIFO 四分之三空情况
10	FIFO1B2IFG	R	1h	FIFO1B2IFG 的原始中断状态 0h = 未发生 FIFO 半空情况 1h = 已发生 FIFO 半空情况
9	FIFO1B4IFG	R	1h	FIFO1B4IFG 的原始中断状态 0h = 未发生 FIFO 四分之一空情况 1h = 已发生 FIFO 四分之一空情况
8	FIFOFULLIFG	R	0h	FIFOFULLIFG 的原始中断状态 0h = 未发生 FIFO 已满情况 1h = 已发生 FIFO 已满情况
7-2	RESERVED	R	0h	
1	MODRDYIFG	R	0h	MODRDYIFG 的原始中断状态 0h = 未发生 DAC 模块就绪事件 1h = 已发生 DAC 模块就绪事件

表 14-22. RIS 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	R	0h	

14.3.15 MIS (偏移 = 1068h) [复位 = 00000000h]

图 14-22 展示了 MIS，表 14-23 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 14-22. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	RESERVED
R-0h						R-0h	R-0h

表 14-23. MIS 字段说明

位	字段	类型	复位	说明
31-15	保留	R	0h	
14	DMADONEIFG	R	0h	DMADONEIFG 的屏蔽中断状态 0h = DMADONEIFG 不请求中断服务例程 1h = DMADONEIFG 请求一个中断服务例程
13	FIFOURUNIFG	R	0h	FIFOURUNIFG 的屏蔽中断状态 0h = FIFOURUNIFG 不请求中断服务例程 1h = FIFOURUNIFG 请求一个中断服务例程
12	FIFOEMPTYIFG	R	0h	FIFOEMPTYIFG 的屏蔽中断状态 0h = FIFOEMPTYIFG 不请求中断服务例程 1h = FIFOEMPTYIFG 请求一个中断服务例程
11	FIFO3B4IFG	R	0h	FIFO3B4IFG 的屏蔽中断状态 0h = FIFO3B4IFG 不请求中断服务例程 1h = FIFO3B4IFG 请求一个中断服务例程
10	FIFO1B2IFG	R	0h	FIFO1B2IFG 的屏蔽中断状态 0h = FIFO1B2IFG 不请求中断服务例程 1h = FIFO1B2IFG 请求一个中断服务例程
9	FIFO1B4IFG	R	0h	FIFO1B4IFG 的屏蔽中断状态 0h = FIFO1B4IFG 不请求中断服务例程 1h = FIFO1B4IFG 请求一个中断服务例程
8	FIFOFULLIFG	R	0h	FIFOFULLIFG 的屏蔽中断状态 0h = FIFOFULLIFG 不请求中断服务例程 1h = FIFOFULLIFG 请求一个中断服务例程
7-2	RESERVED	R	0h	
1	MODRDYIFG	R	0h	MODRDYIFG 的屏蔽中断状态 0h = MODRDYIFG 不请求中断服务例程 1h = MODRDYIFG 请求一个中断服务例程
0	RESERVED	R	0h	

14.3.16 ISET (偏移 = 1070h) [复位 = 00000000h]

图 14-23 展示了 ISET，表 14-24 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 14-23. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	保留
W-0h						W-0h	W-0h

表 14-24. ISET 字段说明

位	字段	类型	复位	说明
31-15	保留	W	0h	
14	DMADONEIFG	W	0h	设置 RIS 寄存器中的 DMADONEIFG 0h = 写入 0 不产生影响 1h = 对应于 DMADONEIFG 的 RIS 位被设置
13	FIFOURUNIFG	W	0h	设置 RIS 寄存器中的 FIFOURUNIFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFOURUNIFG 的 RIS 位会被设置
12	FIFOEMPTYIFG	W	0h	设置 RIS 寄存器中的 FIFOEMPTYIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOEMPTYIFG 的 RIS 位被设置
11	FIFO3B4IFG	W	0h	设置 RIS 寄存器中的 FIFO3B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO3B4IFG 的 RIS 位会被设置
10	FIFO1B2IFG	W	0h	设置 RIS 寄存器中的 FIFO1B2IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B2IFG 的 RIS 位会被设置
9	FIFO1B4IFG	W	0h	设置 RIS 寄存器中的 FIFO1B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B4IFG 的 RIS 位会被设置
8	FIFOFULLIFG	W	0h	设置 RIS 寄存器中的 FIFOFULLIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOFULLIFG 的 RIS 位被设置
7-2	RESERVED	W	0h	
1	MODRDYIFG	W	0h	设置 RIS 寄存器中的 MODRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 MODRDYIFG 的 RIS 位被设置

表 14-24. ISET 字段说明 (continued)

位	字段	类型	复位	说明
0	RESERVED	W	0h	

14.3.17 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 14-24 展示了 ICLR，表 14-25 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 14-24. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留	DMADONEIFG	FIFOURUNIFG	FIFOEMPTYIFG	FIFO3B4IFG	FIFO1B2IFG	FIFO1B4IFG	FIFOFULLIFG
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
RESERVED						MODRDYIFG	保留
W-0h						W-0h	W-0h

表 14-25. ICLR 字段说明

位	字段	类型	复位	说明
31-15	保留	W	0h	
14	DMADONEIFG	W	0h	清除 RIS 寄存器中的 DMADONEIFG 0h = 写入 0 不产生影响 1h = 对应于 DMADONEIFG 的 RIS 位被清零
13	FIFOURUNIFG	W	0h	清除 RIS 寄存器中的 FIFOURUNIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOURUNIFG 的 RIS 位被清零
12	FIFOEMPTYIFG	W	0h	清除 RIS 寄存器中的 FIFOEMPTYIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOEMPTYIFG 的 RIS 位被清零
11	FIFO3B4IFG	W	0h	清除 RIS 寄存器中的 FIFO3B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO3B4IFG 的 RIS 位被清零
10	FIFO1B2IFG	W	0h	清除 RIS 寄存器中的 FIFO1B2IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B2IFG 的 RIS 位被清零
9	FIFO1B4IFG	W	0h	清除 RIS 寄存器中的 FIFO1B4IFG 0h = 写入 0 不会产生影响 1h = 对应于 FIFO1B4IFG 的 RIS 位被清零
8	FIFOFULLIFG	W	0h	清除 RIS 寄存器中的 FIFOFULLIFG 0h = 写入 0 不产生影响 1h = 对应于 FIFOFULLIFG 的 RIS 位被清零
7-2	RESERVED	W	0h	
1	MODRDYIFG	W	0h	清除 RIS 寄存器中的 MODRDYIFG 0h = 写入 0 不产生影响 1h = 对应于 MODRDYIFG 的 RIS 位被清零
0	RESERVED	W	0h	

14.3.18 EVT_MODE（偏移 = 10E0h）[复位 = 00000009h]

图 14-25 展示了 EVT_MODE，表 14-26 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 14-25. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				EVT1_CFG		INT0_CFG	
R/W-0h				R-2h		R-1h	

表 14-26. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	EVT1_CFG	R	2h	none.GEN_EVENT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 由软件处理的事件。软件必须清除关联的 RIS 标志。 2h = 由硬件处理的事件。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	none.CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 由软件处理的事件。软件必须清除关联的 RIS 标志。 2h = 由硬件处理的事件。硬件（另一个模块）会自动清除关联的 RIS 标志。

14.3.19 DESC（偏移 = 10FCh）[复位 = 03110000h]

图 14-26 展示了 DESC，表 14-27 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 14-26. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-311h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-0h				R-				R-0h				R-0h			

表 14-27. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	311h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本
3-0	MINREV	R	0h	IP 的次要版本

14.3.20 CTL0（偏移 = 1100h）[复位 = 00000000h]

图 14-27 展示了 CTL0，表 14-28 中对此进行了介绍。

返回到汇总表。

控制 0 寄存器。

图 14-27. CTL0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							DFM
R/W-0h							R/W-0h
15	14	13	12	11	10	9	8
保留							RES
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/W-0h

表 14-28. CTL0 字段说明

位	字段	类型	复位	说明
31-17	保留	R/W	0h	
16	DFM	R/W	0h	该位定义 DAC 的输入数据的格式。 0h = 直接二进制 1h = 二进制补码
15-9	保留	R/W	0h	
8	RES	R/W	0h	这些位定义 DAC 输出电压的分辨率。 0h = 8 位分辨率 1h = 12 位分辨率
7-1	RESERVED	R/W	0h	
0	ENABLE	R/W	0h	该位启用 DAC 模块。 0h = DAC 已禁用 1h = DAC 已启用

14.3.21 CTL1（偏移 = 1110h）[复位 = 00000000h]

图 14-28 展示了 CTL1，表 14-29 中对此进行了介绍。

返回到汇总表。

控制 1 寄存器。

图 14-28. CTL1

31	30	29	28	27	26	25	24
RESERVED							OPS
R/W-0h							R/W-0h
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留						REFSN	REFSP
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						AMPHIZ	AMPEN
R/W-0h						R/W-0h	R/W-0h

表 14-29. CTL1 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R/W	0h	
24	OPS	R/W	0h	这些位选择器件引脚上的 DAC 输出。 0h = 无连接。两个 DAC 输出开关均打开。 1h = OUT0 输出已选择
23-10	RESERVED	R/W	0h	
9	REFSN	R/W	0h	该位选择 DAC 基准电压源和输入。 0h = VEREFN 引脚为 VR- 1h = 模拟电源 (VSSA) 为 VR-
8	REFSP	R/W	0h	该位选择 DAC 基准电压源和输入。 0h = 模拟电源 (VDDA) 为 VR+ 1h = VEREFP 引脚为 VR+
7-2	RESERVED	R/W	0h	
1	AMPHIZ	R/W	0h	AMPHIZ - 放大器输出值 0：放大器输出为高阻抗 1：放大器输出被下拉至地 0h = 禁用时 HiZ 1h = 禁用时 dacout 下拉
0	AMPEN	R/W	0h	AMP_EN - 输出放大器启用或禁用 0：禁用 1：启用 0h = 禁用 1h = 启用

14.3.22 CTL2（偏移 = 1120h）[复位 = 00000000h]

图 14-29 展示了 CTL2，表 14-30 中对此进行了介绍。

返回到汇总表。

控制 2 寄存器。

图 14-29. CTL2

31	30	29	28	27	26	25	24
RESERVED							DMATRIGEN
R/W-0h							RH/W-0h
23	22	21	20	19	18	17	16
RESERVED						FIFOTRIGSEL	
R/W-0h						R/W-0h	
15	14	13	12	11	10	9	8
保留						FIFOTH	
R/W-0h						R/W-0h	
7	6	5	4	3	2	1	0
RESERVED							FIFOEN
R/W-0h							R/W-0h

表 14-30. CTL2 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R/W	0h	
24	DMATRIGEN	RH/W	0h	该位会启用 DMA 触发生成机制。当设置了该位和 FIFOEN 位时，将根据由 FIFOTH 设置限定的空 FIFO 位置生成 DMA 触发。该位应由软件清零来停止进一步的 DMA 触发。 0h = 禁用 DMA 触发生成机制 1h = 启用 DMA 触发生成机制
23-18	RESERVED	R/W	0h	
17-16	FIFOTRIGSEL	R/W	0h	这些位选择 FIFO 读取触发源。当选择的 FIFO 读取触发有效时，数据将从 FIFO（如读取指针所指）中移动到内部 DAC 数据寄存器中。 0h = 采样时间发生器输出 1h = 来自事件结构的硬件触发 0 2h = 保留 - 未实现 3h = 保留 - 未实现
15-10	保留	R/W	0h	
9-8	FIFOTH	R/W	0h	这些位可确定 FIFO 阈值。在基于 DMA 的操作中，当 FIFO 中的空位置数量与所选的 FIFO 阈值级别匹配时，DAC 会生成新的 DMA 触发。而在基于 CPU 的操作中，不关注 FIFO 阈值位，FIFO 水平通过 RIS 寄存器中相应的位直接指示。 0h = FIFO 四分之一位置为空 1h = FIFO 一半位置为空 2h = FIFO 四分之三位置为空 3h = 预留值默认情况下，与 FIFOTH 设置为 0 时相同效果（FIFO 四分之一位置为空）。
7-1	RESERVED	R/W	0h	
0	FIFOEN	R/W	0h	该位使能 FIFO 和 FIFO 硬件控制状态机。 0h = FIFO 已禁用 1h = FIFO 已启用

14.3.23 CTL3（偏移 = 1130h）[复位 = 00000000h]

图 14-30 展示了 CTL3，表 14-31 中对此进行了介绍。

返回到汇总表。

控制 3 寄存器。

图 14-30. CTL3

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留				STIMCONFIG			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
RESERVED							STIMEN
R/W-0h							R/W-0h

表 14-31. CTL3 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11-8	STIMCONFIG	R/W	0h	这些位用于配置采样时间发生器的触发速率。STIMCONFIG 的取值 10-15 被保留，并且默认情况下与取值 0 (500SPS) 具有相同的效果。 0h = 触发速率为 500sps（时钟分频值为 4000） 1h = 触发速率为 1ksps（时钟分频值为 2000） 2h = 触发速率为 2ksps（时钟分频值为 1000） 3h = 触发速率为 4ksps（时钟分频值为 500） 4h = 触发速率为 8ksps（时钟分频值为 250） 5h = 触发速率为 16ksps（时钟分频值为 125） 6h = 触发速率为 100ksps（时钟分频值为 20） 7h = 触发速率为 200ksps（时钟分频值为 10） 8h = 触发速率为 500ksps（时钟分频值为 4） 9h = 触发速率为 1Msps（时钟分频值为 2）
7-1	RESERVED	R/W	0h	
0	STIMEN	R/W	0h	该位会启用采样时间发生器。 0h = 采样时间发生器被禁用 1h = 采样时间发生器被启用

14.3.24 CALCTL（偏移 = 1140h）[复位 = 00000000h]

图 14-31 展示了 CALCTL，表 14-32 中对此进行了介绍。

返回到汇总表。

校准控制寄存器。

图 14-31. CALCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						CALSEL	CALON
R/W-0h						RH/W-0h	RH/W-0h

表 14-32. CALCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	CALSEL	RH/W	0h	该位用于选择出厂调整或自校准。 0h (R/W) = 出厂调整：当启用校准时，使用出厂调整校准值。 1h (R/W) = 自校准：当启用校准时，使用自校准值。
0	CALON	RH/W	0h	当设置该位时，将启动 DAC 偏移量误差校准序列，并在偏移量误差校准完成后自动复位。 0h = 偏移量误差校准未激活 1h = 启动偏移量误差校准或偏移量误差校准已在进行中

14.3.25 CALDATA（偏移 = 1160h）[复位 = 00000000h]

图 14-32 展示了 CALDATA，表 14-33 中对此进行了介绍。

返回到汇总表。

这是偏移量误差校准数据寄存器。

图 14-32. CALDATA

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

表 14-33. CALDATA 字段说明

位	字段	类型	复位	说明
31-7	RESERVED	R	0h	
6-0	数据	RH	0h	DAC 偏移量误差校准数据。DAC 偏移量误差校准数据采用二进制补码格式表示，取值范围在 -64 到 +63。 该位为只读位，反映了校准数据。对该寄存器进行写操作不会产生影响，不会改变校准值。

14.3.26 DATA0（偏移 = 1200h）[复位 = 00000000h]

图 14-33 展示了 DATA0，表 14-34 中对此进行了介绍。

返回到汇总表。

Data 0 寄存器。该寄存器可以通过 8 位或 12 位的数字输入数据进行写入。

图 14-33. DATA0

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

表 14-34. DATA0 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11-0	DATA_VALUE	R/W	0h	这是为数模转换而写入的数据。

章节 15
VREF

VREF 模块包含一个可配置的电压基准缓冲器，让用户能够为板载模拟外设提供一个稳定的内部基准。该模块还支持为需要更高精度的应用提供外部基准。本章介绍 VREF 模块的特性和运行。

15.1 VREF 概述

MSPM0Gxx 系列的 VREF 模块是一个可供各种板载模拟外设使用的共享电压基准模块。VREF 允许用户选择使用内部生成的基准电压还是使用 MCU 外部提供的基准电压。

VREF 模块的特性包括：

- 用户可选择 1.4V 和 2.5V 内部基准电压
- 支持在 VREF+/- 器件引脚上接收外部基准电压
- VREF+ 器件引脚上提供内部基准电压输出
- 采样保持模式支持在 STANDBY 工作模式下运行 VREF
- 内部基准支持全速运行 ADC

图 15-1 展示了 VREF 模块的方框图。

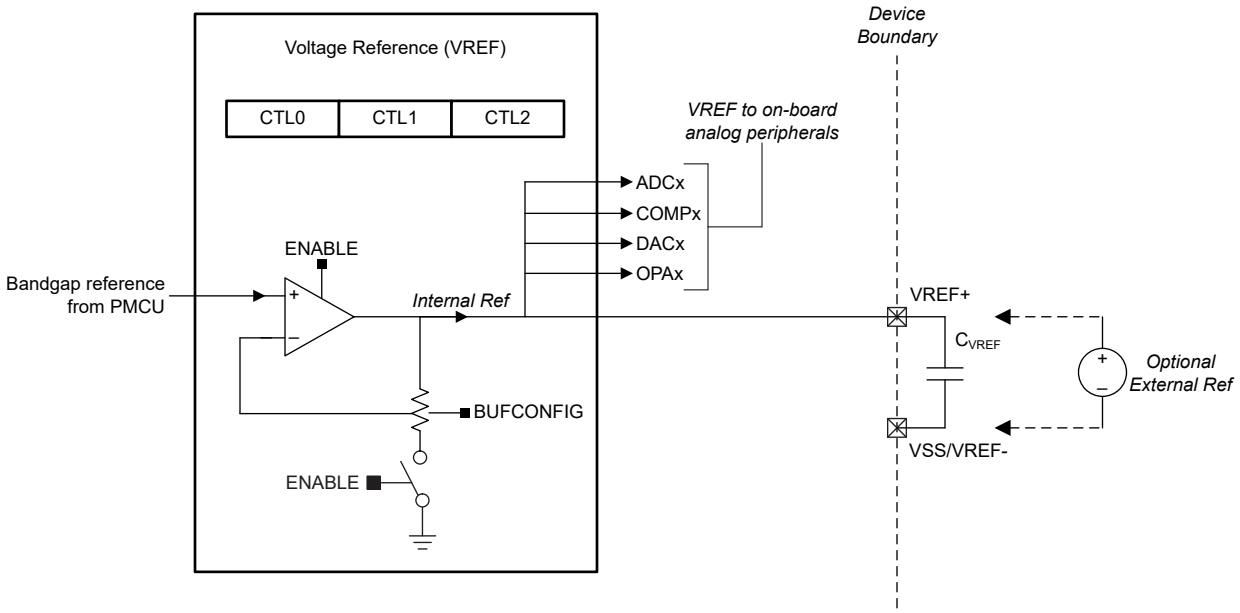


图 15-1. VREF 方框图

15.2 VREF 运行

VREF 模块可由用户软件配置。以下各节将讨论 VREF 的设置和运行。

15.2.1 内部基准生成

要使用 VREF 生成内部电压基准，用户必须首先使用 PWREN 寄存器中的 ENABLE 控制位来启用模块的电源，然后使用 CTL0 寄存器中的 ENABLE 控制位来启用基准缓冲器。VREF 模块根据来自 PMU 的工厂校准带隙生成电压基准。带隙基准通过同相放大器进行缓冲，以生成两个内部基准电压（1.4V 或 2.5V）之一。通过 CTL0 中的 BUFCONFIG 控制位，一次只能选择一个电压。

备注

VREF 生成的内部基准需要一个外部去耦电容器 (C_{VREF}) 才能正常运行。有关 C_{VREF} 的值和放置要求的更多信息，请参阅特定于器件的数据表。

启用并稳定后，内部基准可用作板载模拟外设 的精确稳定电压基准。有关模拟外设如何利用此基准电压的更多信息，请参阅节 15.2.3。

VREF 在 CTL1 寄存器中提供了一个 READY 指示位。首次启用 VREF 时，READY 位将保持清零，直到 VREF 启动并稳定，之后 READY 位由硬件置位。如果 VREF 被禁用，READY 位将由硬件清零。如果稍后重新启用 VREF，则 READY 位将不会置位，应用软件必须管理 VREF 启动时间。

15.2.2 外部基准输入

对于精度是关键要求的情况，可通过 VREF+ 和 VREF- 器件引脚将外部基准引入 MCU。一次只能选择一种类型的基准（内部或外部）。为 MCU 提供外部基准时，建议在基准引脚上使用基于电压源的值连接一个去耦电容器。

备注

在将外部基准应用于器件引脚之前，请务必通过将 CTL0 寄存器中的 ENABLE 控制位清零来禁用 VREF 基准缓冲器。

15.2.3 模拟外设接口

VREF 至 ADC

ADC 外设可以利用内部可配置基准或使用 ADC MEMCTL 寄存器中 VRSEL 控制位的外部基准。

当使用内部基准作为 ADC 电压基准 (V_{R+}) 时，用户必须确保在触发 ADC 转换之前 VREF 基准缓冲器已经稳定。有关 VREF 稳定时间，请参阅器件专用数据表；有关 ADC 外设可用的所有基准选项的更多详细信息，请参阅节 10.2.2。

VREF 至 COMP

比较器外设可以通过 CTL2 寄存器中的 REFSRC 控制位来利用内部可配置基准或外部基准。

有关比较器外设可用的所有基准选项的更多详细信息，请参阅节 11.2.7。

VREF 至 DAC

DAC 外设可以通过 DAC CTL1 寄存器中的 REFSP 控制位利用内部可配置基准或外部基准。

有关 DAC 外设可用的所有基准选项的更多详细信息，请参阅 DAC 参考章节的占位符。

VREF 至 OPA

OPA 外设可利用内部可配置基准或外部基准，使用 CFG 寄存器中的 PSEL 控制位在放大器的同相输入端子上设置电压偏置。如果系统中的其他器件需要使用与 MCU 相同的基准，OPA 可用于单位增益缓冲器配置，以将 VREF 内部基准电压驱动至 OPA_OUT 引脚，该引脚可用作 PCB 上其他器件的基准。

有关可用于放大器同相放大器输入端子的所有通道的更多详细信息，请参阅节 12.2.3。

15.3 VREF 寄存器

表 15-1 列出了 VREF 寄存器的存储器映射寄存器。表 15-1 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 15-1. VREF 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1000h	CLKDIV	时钟分频器		转到
1008h	CLKSEL	时钟选择		转到
1100h	CTL0	控制 0		转到
1104h	CTL1	控制 1		转到
1108h	CTL2	控制 2		转到

复杂的位访问类型经过编码可适应小型表单元。表 15-2 展示了适用于此部分中访问类型的代码。

表 15-2. VREF 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

15.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 15-2 展示了 PWREN，表 15-3 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 15-2. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-							K-0h

表 15-3. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

15.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 15-3 展示了 RSTCTL，表 15-4 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 15-3. RSTCTL

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

表 15-4. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

15.3.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 15-4 展示了 STAT，表 15-5 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 15-4. STAT

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

表 15-5. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

15.3.4 CLKDIV（偏移 = 1000h）[复位 = 00000000h]

图 15-5 展示了 CLKDIV，表 15-6 中对此进行了介绍。

返回到汇总表。

时钟分频器

图 15-5. CLKDIV

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

表 15-6. CLKDIV 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择要在采样保持逻辑中使用的模块时钟的分频比

15.3.5 CLKSEL（偏移 = 1008h）[复位 = 00000000h]

图 15-6 展示了 CLKSEL，表 15-7 中对此进行了介绍。

返回到汇总表。

时钟选择

图 15-6. CLKSEL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				BUSCLK_SEL	MFCLK_SEL	LFCLK_SEL	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 15-7. CLKSEL 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	BUSCLK_SEL	R/W	0h	如果启用，则选择 BUSCLK 作为时钟源
2	MFCLK_SEL	R/W	0h	如果启用，则选择 MFCLK 作为时钟源
1	LFCLK_SEL	R/W	0h	如果启用，则选择 LFCLK 作为时钟源
0	RESERVED	R/W	0h	

15.3.6 CTL0（偏移 = 1100h）[复位 = 00000000h]

图 15-7 展示了 CTL0，表 15-8 中对此进行了介绍。

返回到汇总表。

控制 0 寄存器。

图 15-7. CTL0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							SHMODE
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
BUFCONFIG	保留					COMP_VREF_ENABLE	ENABLE
R/W-0h	R/W-0h					R/W-0h	R/W-0h

表 15-8. CTL0 字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	0h	
8	SHMODE	R/W	0h	该位启用采样保持模式 0h = 采样保持模式禁用 1h = 采样保持模式启用
7	BUFCONFIG	R/W	0h	这些位配置输出缓冲器。 0h = 输出 2p5v：将输出缓冲器配置为 2.5V 1h = 输出 1p4v：将输出缓冲器配置为 1.4V
6-2	RESERVED	R/W	0h	
1	COMP_VREF_ENABLE	R/W	0h	比较器 Vref 启用 0h = COMP VREF 已禁用 1h = COMP VREF 已启用
0	ENABLE	R/W	0h	该位启用 VREF 模块。 0h = VREF 已禁用 1h = VREF 已启用

15.3.7 CTL1（偏移 = 1104h）[复位 = 00000000h]

图 15-8 展示了 CTL1，表 15-9 中对此进行了介绍。

返回到汇总表。

控制 1 寄存器。

图 15-8. CTL1

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

表 15-9. CTL1 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
0	就绪	R	0h	这些位定义 VREF 的状态 0h = VREF 输出未就绪 1h = VREF 输出已就绪

15.3.8 CTL2（偏移 = 1108h）[复位 = 00000000h]

图 15-9 展示了 CTL2，表 15-10 中对此进行了介绍。

返回到汇总表。

控制 2 寄存器。

图 15-9. CTL2

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

表 15-10. CTL2 字段说明

位	字段	类型	复位	说明
31-16	HCYCLE	R/W	0h	保持周期计数 当 VREF 在采样保持模式下处于待机状态以节省功耗时，模块时钟在保持相位的总周期数。 有关采样保持时间的建议值，请参阅数据表的 VREF 部分。 0h = 最小保持周期 FFFFh = 最大保持周期
15-0	SHCYCLE	R/W	0h	采样保持周期计数 当 VREF 在采样保持模式下处于待机状态以节省功耗时，模块时钟在采样保持相位的总周期数。 该字段应大于 HCYCLE 字段。该字段与 HCYCLE 的差值即是采样相位的周期数。 有关采样保持时间的建议值，请参阅数据表的 VREF 部分。 0h = 最小采样保持周期计数 FFFFh = 最大采样保持周期计数

通用异步接收器/发送器 (UART) 模块为串行通信协议提供了一个接口。

16.1 UART 概述

16.1.1 外设的用途

此接口可用于在 MSPM0 器件和其他器件之间传输数据，支持异步串行通信协议，例如 LIN（本地互连网络）、ISO7816（智能卡协议）、IrDA（红外数据协会）、硬件流控制 (CTS/RTS) 和多处理器通信。

16.1.2 特性

UART 控制器包含以下特性：

- 完全可编程串行接口
  - 5、6、7 或 8 个数据位
  - 偶校验、奇校验、固定校验或无奇偶校验位生成与检测
  - 可产生 1 或 2 个停止位
  - 最低有效位或最高有效位数据最先传送和接收
  - 断线检测
  - 输入信号上的干扰滤波器
  - 可编程波特率生成，过采样率为 16、8 或 3 倍
- 独立的发送和接收 4 深度 FIFO 可减少 CPU 中断服务负载
- 支持 DMA 数据传输
- 提供标准的基于 FIFO 深度的中断以及发送结束中断
- 在包括停止和待机模式在内的所有低功耗模式下均有效
- 在低功耗模式下运行时，支持通过异步快速时钟请求在检测到起始位时唤醒 SYSOSC（在 FCL 模式下使用 SYSOSC 时，支持高达 19200B 的速率）（精度 1%）
- 支持环回模式运行
- 支持硬件流控制
- 支持 9 位多点配置
- 支持的协议：
  - 本地互连网络 (LIN) 支持
  - DALI
  - IrDA
  - ISO7816 Smart Card
  - RS485
  - 曼彻斯特编码
  - 空闲线多处理器

备注

这是一个总体概述，下面的表 16-1 列出了 UART 扩展模块与 UART 主模块之间的差异。有关 UART 扩展模块和 UART 主模块的特定 UART 配置，请参阅器件数据表。

表 16-1. UART 扩展模块和主模块特性(1)

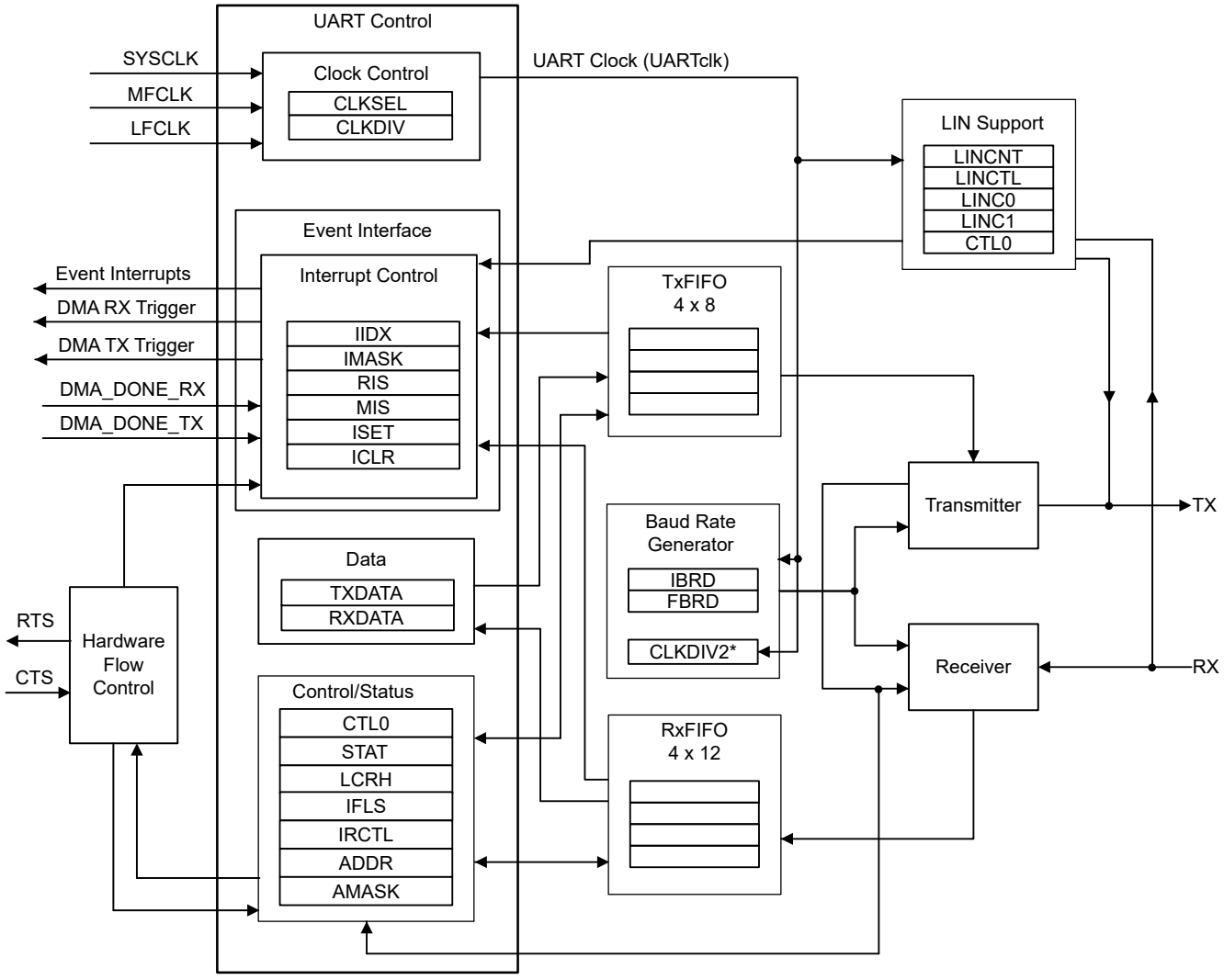
特性	UART 扩展模块	UART 主模块
硬件流控制	是	是
过采样选项	3、8、16	3、8、16
独立的发送 FIFO 和接收 FIFO	是	是
在所有低功耗模式下均有效	是(2)	是(2)
通过开始位唤醒	是(3)	是(3)
数字干扰滤波器	是	否
模拟干扰滤波器	是	是
9 位多点配置	是	是
空闲线多处理器	是	是
RS-485 的数字隔离器	是	是

表 16-1. UART 扩展模块和主模块特性(1) (continued)

特性	UART 扩展模块	UART 主模块
支持 LIN 模式	是	-
支持 DALI	是	-
支持 IrDA	是	-
支持 ISO7816 Smart Card	是	-
支持曼彻斯特编码	是	-

(1) 有关 UART 扩展模块和 UART 主模块及其电源域的器件特定配置，请参阅器件特定数据表。
(2) UART 可在包括停止和待机在内的所有低功耗模式下有效，除非 UART 实例处于电源域 1 (PD1) 中。
(3) 仅适用于电源域 0 (PD0) 中的 UART 实例。

16.1.3 功能方框图



*CLKDIV2 is for IrDA mode only

图 16-1. UART 功能方框图

16.2 UART 运行

本节介绍了 UART 外设的运行情况。

16.2.1 时钟控制

UART 内部功能时钟被选择并按 IP 的功能时钟分频。

- 使用 UARTx.CLKSEL 寄存器来选择 UART 功能时钟的源。
  - BUSSCLK：当前总线时钟被选为 UART 的源。当前总线时钟取决于电源域。如果 UART 实例位于电源域 1 (PD1) 中，请参阅 MCLK，如果 UART 实例位于电源域 0 (PD0) 中，请参阅 ULPCLK。
  - MFCLK：选择 MFCLK 作为 UART 的时钟源，请参阅 MFCLK。
  - LFCLK：选择 LFCLK 作为 UART 的源，请参阅 LFCLK
- 使用 UARTx.CLKDIV 寄存器选择 UART 功能时钟的分频比，选项为按 1 至 8 分频。对于 UART Extend，有一个 CLKDIV2 寄存器来进一步分频 UART 功能时钟以支持 IrDA 模式。使用 IrDA 模式时，必须将 CLKDIV2.RATIO 设置为 1h 才能进行正确的 IrDA 计时。

所选的时钟源始终可用，频率取决于电源模式。有关更多信息，请参阅时钟模块 (CKM) 部分。通过置位 ENABLE 位启用 UART 模块后，该模块即准备好开始接收和发送数据。

16.2.2 信号说明

UART 通信需要两个引脚：接收数据 (RX) 和发送数据 (TX)：

- RX（接收数据）：RX 是串行数据输入。需要对接收信号使用干扰滤波器和过采样技术来确保传入数据的准确性。
- TX（发送数据）：TX 是串行数据输出。当启用发送器且无需发送数据时，TX 引脚将保持高电平。在 ISO7816 智能卡协议等一些双向协议中，该引脚也用于接收数据。

在硬件流控制模式下，还使用以下引脚：

- CTS（允许发送）：被外部信号驱动为高电平时，该信号在当前传输结束时阻止数据传输。
- RTS（请求发送）：为低电平时，该信号表示 UART 已准备好接收数据。

16.2.3 通用架构和协议

UART 发送和接收字符时的比特率与另一器件不同步。每个字符的时序取决于所选的波特率。传输和接收功能使用相同的波特率。

通常，仅当 UART 被禁用（UARTx.CTL0 寄存器中的 ENABLE 位清零）后，才应当对控制寄存器进行编程。如果 UART 正在运行并在发送或接收操作期间被禁用，则当前事务会在 UART 停止之前完成。可以在不禁用 UART 的情况下修改波特率除数寄存器（UARTx.IBRD 和 UARTx.FBRD）。

16.2.3.1 发送/接收逻辑

发送逻辑单元从发送 FIFO 取出数据后执行并-串转换。控制逻辑输出串行位流时，最先输出起始位，然后按照编程配置依次输出若干数据位（LSB 在前）、奇偶校验位和停止位。

接收逻辑单元在检测到有效的起始脉冲后，对接收到的串行位码流执行串-并转换。此外还会对溢出错误、奇偶校验错误、帧错误和线路中断错误进行检测，并将检测到的状态附加到被写入接收 FIFO 的数据中，这便是使用 12 位接收 FIFO 的原因。

UARTx.CTL0.ENABLE 位用于启用和禁用 UART 模块，UARTx.CTL0.TXE 和 RXE 位用于启用发送和接收模式，UARTx.LCRH.WLEN 位用于配置在一个帧中发送或接收的数据位数，UARTx.LCRH.PEN 用于启用奇偶校验，UARTx.LCRH.STP2 用于发送两个停止位。

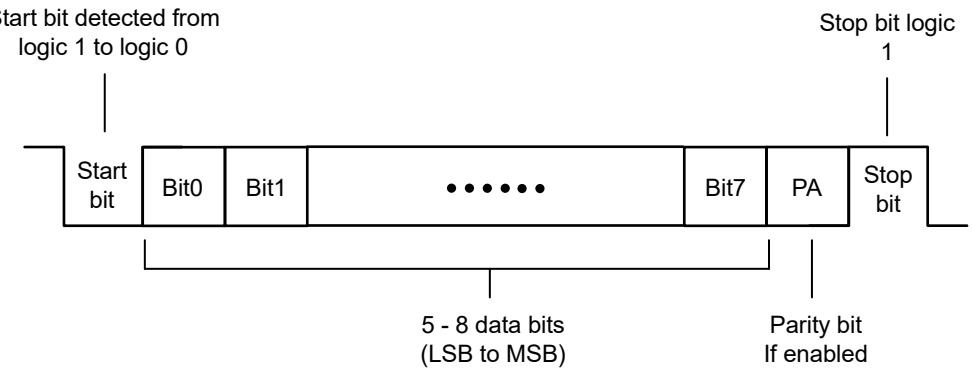


图 16-2. UART 字符帧

16.2.3.2 位采样

默认情况下，UARTx.CTL0.HSE 设置为 0 并选择 16 个过采样，接收位的长度预计为 16 个 UART 时钟 UARTclk 周期，并在第 8 个 UARTclk 周期进行采样。

将 UARTx.CTL0.HSE 位设置为 1 可选择 8 个过采样，其中接收位的时钟长度应为 8 个 UART 时钟 UARTclk 周期，并在第 4 个 UARTclk 周期进行采样。

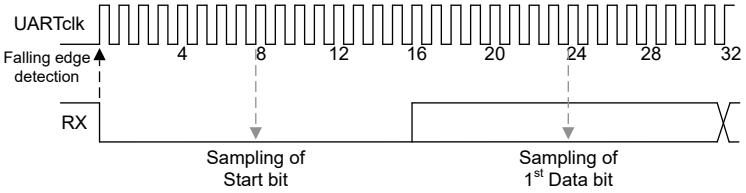
将 UARTx.CTL0.HSE 位设置为 2 可选择 3 个过采样，接收位应具有 3 个 UART 时钟 UARTclk 周期的长度，并在第 2 个 UARTclk 周期进行采样。

上述情景假设 IBRD = 1 且 FBRD = 0。

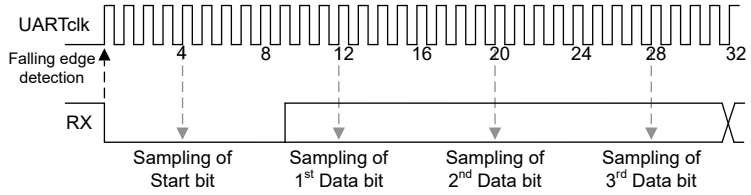
具体取决于应用：

- 选择 3 或 8 倍过采样可通过 UARTclk/8 或 UARTclk/3 实现更高的速度。在这种情况下，接收器对时钟偏差的容差会降低。
- 选择 16 倍过采样以增加接收器对时钟偏差的容差。最大速度限制为 UARTclk/16。

16x oversampling mode (HSE = 0)



8x oversampling mode (HSE = 1)



3x oversampling mode (HSE = 2)

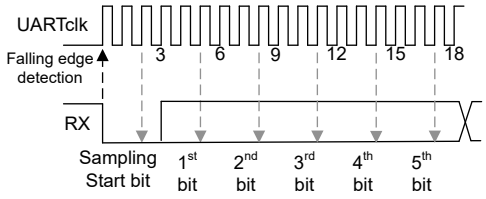


图 16-3. UART 过采样模式

16.2.3.3 波特率生成

波特率分频系数是由 16 位整数部分和 6 位小数部分组成的 22 位二进制数。波特率发生器使用这两个值形成的数字来确定位采样周期。分数波特率分频器可让 UART 非常准确地生成所有标准波特率。

16 位整数通过 UART 整数波特率分频器 UARTx.IBRD 寄存器进行加载，而 6 位小数则通过 UART 分数波特率分频器 UARTx.FBRD 寄存器进行加载。

波特率分频可以使用以下公式计算：

BRD = UART 时钟 / (过采样 x 波特率) (23)

UART 时钟是 UART 时钟控制逻辑的时钟输出，由 CLKSEL 和 CLKDIV 配置。过采样由 UARTx.CTL0 寄存器的 HSE 位进行选择，可以是 16、8 或 3。

- UARTx.IBRD = INT(BRD)，BRD 的整数部分
- UARTx.FBRD = BRD % 64，小数部分

BRD 的整数部分被加载到 UARTx.IBRD 寄存器中。必须将 6 位小数加载到 UARTx.FBRD 寄存器中。

备注

当 IBRD = 0 时，FBRD 被忽略，UART 不会传输任何数据。同样，当 IBRD = 65535 (即 0xFFFF) 时，FBRD 不得大于零。如果超过此值，则会导致发送或接收中止。

下面的示例显示了一种简单的方法来计算 19200 位/秒波特率下的 IBRD.DIVINT 和 FBRD.DIVFRAC：

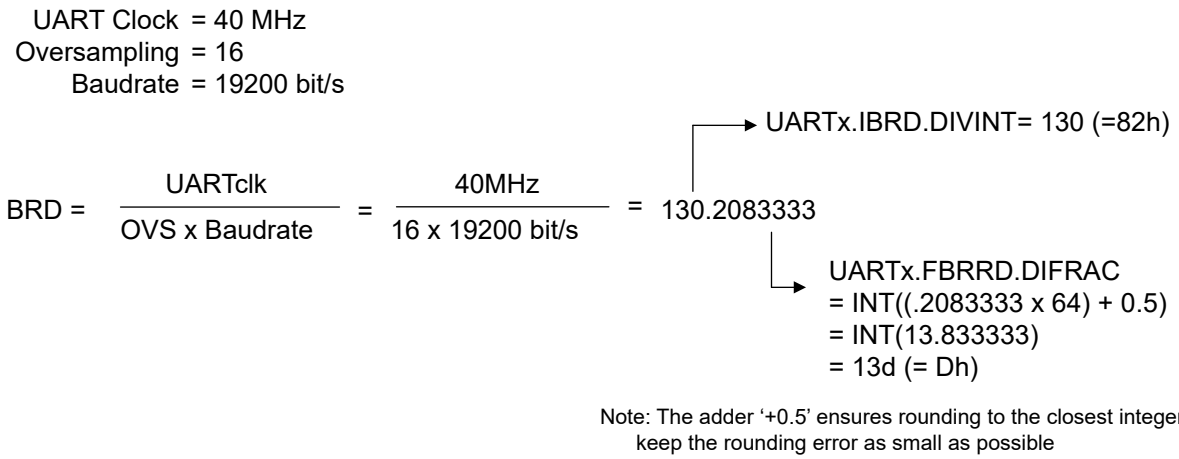


图 16-4. 波特率配置

当更新波特率分频（UARTx.IBRD 或 UARTx.IFRD）时，也必须写入 UART.LCRH 寄存器，所以波特率分频的任何改变都必须在写入 LCRH 寄存器之后，更改才会生效。UART.IBRD 和 UART.FBRD 寄存器的内容在当前字符的发送或接收完成后才会更新。

16.2.3.4 数据传输

接收或发送的数据存储在两个 FIFO 中，但是接收 FIFO 的每个字符都有额外的四位用于指示状态。

发送数据：

当需要进行发送时，先将数据写入发送 FIFO。如果启用 UART，则会根据 UARTx.LCRH 寄存器中指示的参数开始发送数据帧。UART 模块会持续发送数据，直到发送 FIFO 中没有可发数据为止。数据一经写入发送 FIFO（即，如果 FIFO 不为空），UARTx.STAT 寄存器中的 BUSY 位即会生效，并在数据发送期间一直保持有效。仅当发送 FIFO 为空，且最后一个字符（包括停止位）已发送到移位寄存器时，BUSY 位才为负。即使无法再启用 UART，UART 也可以指示它正忙。生成 BREAK 信号期间，也会设置 BUSY。

接收数据：

当接收器空闲（RX 信号持续为 1）且数据输入变为低电平（已接收到开始位）时，接收计数器开始运行，并根据 UARTx.CTL0 寄存器中 HSE 位的过采样设置，在不同的周期对数据进行采样。如果 RX 信号在基于过采样设置的特定数量的周期后仍为低电平，则开始位有效并被识别。检测到有效的开始位后，将根据数据字符的编程长度对连续的数据位进行采样。之后将捕获并校验奇偶校验位（如果使能了奇偶校验）。数据长度和奇偶校验都在 UART.LCRH 寄存器中定义。节 16.2.3.2 中介绍了过采样。

最后，如果 RXD 信号为高电平，则确认有效停止位，否则发生成帧错误。若成功接收到一帧数据，则数据和与之相关的错误标志都将保存到接收 FIFO 中。

16.2.3.5 错误和状态

对于接收到的数据，数据字节和 4 位状态（中断、帧、奇偶校验和溢出）会推到 12 位宽的接收 FIFO 上。可以通过读取 UARTx.RXDATA 寄存器来检索错误和状态，如表 16-2 所示。

表 16-2. UART 错误和条件

错误条件(1)	位字段	说明
组帧错误	FRMERR	当一个低电平停止位被监测到时发生一个组帧错误。当使用两个停止位时，这两个位都会被检查是否有组帧错误。检测到成帧错误时，设置 FRMERR 位。
奇偶校验错误	PARERR	奇偶校验错误也就是字符中 1s 的数量和奇偶校验位的值不匹配。当一个地址位包含于字符时，它同时也被包含进奇偶校验计算中。检测到奇偶校验错误时，设置 PARERR 位。
接收溢出	OVRERR	如果将某个字符加载到 RXDATA/FIFO 中，而尚未读取其前一个字符，则会发生溢出错误。发生溢出错误时，设置 OVRERR 位。

表 16-2. UART 错误和条件 (continued)

错误条件(1)	位字段	说明
中断状态	BRKERR	当所有接收到的数据、奇偶校验和停止位均为 0 时，会检测到中断。检测到中断条件时，设置 BRKERR 位。如果设置了中断使能 IMASK.BRKERR 位，则中断条件也可以设置中断标志 RXINT。

(1) 启用 LIN 模式时不设置成帧错误和中断条件，此模式用于为 LIN 发送同步帧信号。中断检测不适用于 IRDA、曼彻斯特和 DALI 模式。

也可以通过读取 UARTx.STAT 寄存器来检查 UART 模块标志状态，如表 16-3 所示。

表 16-3. UART 标志状态

位字段	说明
BUSY	一旦发送 FIFO 不为空时（不管 UART 是否使能）该位都会置位。在 IDLE_Line 模式下，BUSY 信号在空闲时间生成期间也保持设置状态。
RXFE	当接收 FIFO 为空时，设置该位。若 FIFO 被禁用（FEN = 0），则表明接收保持寄存器已满。若 FIFO 被使能（FEN = 1），则表明接收 FIFO 已满。
RXFF	当接收 FIFO 已满时，设置该位。如果 FIFO 处于禁用状态（FEN 为 0），则接收保持寄存器为空。如果 FIFO 处于启用状态（FEN 为 1），则接收 FIFO 为空。
TXFE	当发送 FIFO 为空时，设置该位。如果 FIFO 处于禁用状态（FEN 为 0），则发送保持寄存器为满。如果 FIFO 处于启用状态（FEN 为 1），则发送 FIFO 为满。
TXFF	当发送 FIFO 已满时，设置该位。若 FIFO 被禁用（FEN = 0），则表明发送保持寄存器为空。若 FIFO 被使能（FEN = 1），则表明发送 FIFO 为空。
CTS	当 CTS 信号有效（低电平）时设置该位，当 CTS 信号无效（高电平）时将该位清零。
IDLE（闲置）	在空闲线多处理器模式下检测到 IDLE 模式。IDLE 位用作每个字符块的地址标签。在空闲线多处理器格式中，当接收到的字符是地址时设置该位。

16.2.3.6 本地互连网络 (LIN) 支持

本部分仅与支持 LIN 模式的 UART 扩展接口相关。有关 UART 扩展接口和 UART 主接口的器件特定配置，请参阅器件数据表。

为了支持本地互连网络 (LIN) 协议，在 UART 模块中实现了以下硬件增强功能：

- 由 UART 时钟计时的 16 位加法计数器 (LINCNT)。
- 计数器溢出时的中断功能 (CPU_INT.IMASK.LINOVF)。
- 具有两种可配置模式的 16 位捕捉寄存器 (LINC0)
  - 在 RXD 下降沿捕捉 LINCNT 值。捕捉时的中断能力。
  - 比较 LINCNT 与匹配时的中断能力。
- 可以配置 16 位捕捉寄存器 (LINC1)：
  - 在 RXD 上升沿捕捉 LINCNT 值。捕捉时的中断能力。

LIN 发送

可以通过设置 UARTx.LCRH 寄存器中的 BRK 位来发送中断信号。需要在将数据写入 FIFO 或发送数据寄存器 TXDATA 之前设置该位。

要生成 LIN 响应器信号（如唤醒信号），可通过寄存器 UARTx.CTL0 中的 TXD_OUT 和 TXD_CTL_EN 位将 TX 引脚配置为由软件控制。通过将 TXD_CTL_EN 位设置为 1，如果禁用 UART 发送（CTL0.TXE 清零），TX 输出引脚可以由 TXD_OUT 位控制。

LIN 接收

LIN 命令器在每帧开始时发出一个中断域和一个同步域。必须添加硬件，这样 LIN 响应器软件驱动程序才能合理地检测中断同步并测量必要的时序参数，以调整波特率或确定错误。

对于 LIN 接收，需要中断域检测和比较模式。要配置这些功能，请执行以下操作：

1. 将 LIN 计数器初始化为 0 (UARTx.LINCNT = 0)

2. 启用计数器比较匹配模式 (UARTx.LINCTL.LINC0_MATCH = 1)
3. 使用与 9.5 x T_{bit} 相对应的计数器值加载 UARTx.LINC0（计数器捕捉 0 寄存器）（有关此时序的详细信息，请参阅 LIN 规范）。
4. 启用 LINC0 匹配中断 (CPU_INT.IMASK.LINC0 = 1)
5. 设置 LIN 计数控制 (UARTx.LINCTL)：
   - 当 RXD 上的信号为低电平时启用计数 (LINCTL.CNTRXLOW = 1)
   - 在 RXD 下降沿启用 LIN 计数器清零 (LINCTL.ZERONE = 1)
   - 启用 LIN 计数器 (LINCTL.CTRENA = 1)

   可选：
   - 用户可以对 UART TX 信号中断启用上升沿 (CPU_INT.IMASK.RXPE = 1)，当触发 RXPE 中断时，软件可以直接读取 LINCTR 以查看中断域时序。
   - 用户可以启用 LIN 计数器溢出中断 (CPU_INT.IMASK.LINOVF = 1)，以检测中断域过长并溢出 16 位计数器。超时可通过以下公式计算得出：$t_{Timeout} = 2^{16}$/UART 时钟。

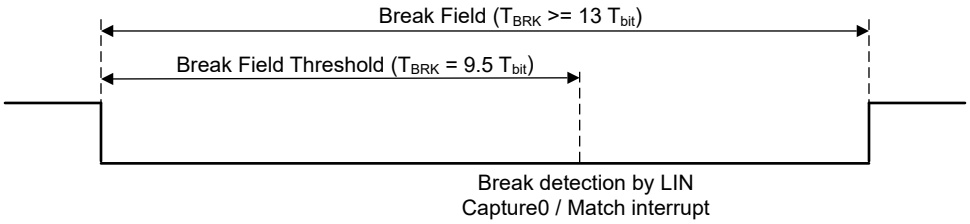


图 16-5. LIN 中断域检测

需要进行同步域验证，以确保 LIN 报头部分的准确性并计算命令器波特率。同步域由字节域内的数据 0x55 构成（请参阅图 16-6）。软件检测到有效中断后，可以设置计数器来测量同步域。使用 LIN 计数器中的两个捕捉寄存器，以便软件中断较少。图 16-6 展示了同步字节格式。LINCTR 应在起始位下降沿设置为 0 并持续计数。LINC0 捕捉或 RX 下降沿中断在 RX 线路的下降沿触发。在中断处理期间，软件可以使用 LINC0 和 LINC1 寄存器中的值来测量各个位本身的单独高-低电平时间，以确保所有时序都有效。

以下流程描述了一个可行的 LIN 同步域验证过程：

1. 在检测到有效的中断域后，将 LIN 计数器初始化为 0 (UARTx.LINCNT = 0)。
2. 在 RX 下降沿启用中断 (CPU_INT.IMASK.RXNE = 1)
3. 设置 LIN 计数控制 (LINCTL)：
   - 在 RX 上升沿启用 LIN 计数器捕捉 (LINCTL.LINC1CAP = 1)
   - 在 RX 下降沿启用 LIN 计数器捕捉 (LINCTL.LINC0CAP = 1)
   - 在 RX 下降沿启用 LIN 计数器清零 (LINCTL.ZERONE = 1)
   - 启用 LIN 计数器 (LINCTL.CTRENA = 1)

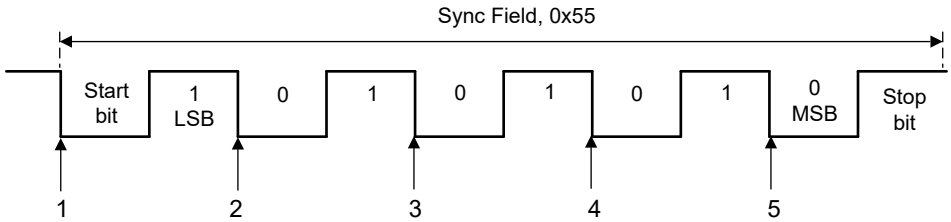


图 16-6. LIN 同步域检测

同步域在 RX 线路的每个下降沿的操作如下所示：

1. LIN 计数器设置为 0，并在 RX 下降沿开始计数。(LINCTL.ZERONE = 1)
2. RX 下降沿中断触发 (RXNE)：
   - 读取捕捉寄存器 LINC0（下降沿）和 LINC1（上升沿）的值

- 验证位时间

3. RX 下降沿中断触发 (RXNE) :
   - 读取捕捉寄存器 LINC0 (下降沿) 和 LINC1 (上升沿) 的值
   - 验证位时间
4. RX 下降沿中断触发 (RXNE) :
   - 读取捕捉寄存器 LINC0 (下降沿) 和 LINC1 (上升沿) 的值
   - 验证位时间
5. RX 下降沿中断触发 (RXNE) :
   - 读取捕捉寄存器 LINC0 (下降沿) 和 LINC1 (上升沿) 的值
   - 验证位时间
   - 计算要设置的正确波特率。软件必须在同步域之后 PID 字段的起始位之前设置波特率。

每次发生中断时，必须读取捕捉寄存器，并且位时间需要由应用软件验证。如果出现位时间验证错误，必须中止同步域分析过程，并且应用软件必须切换回中断检测。

如果在同步域检测期间出现诸如中断命令器通信之类的错误，则可以通过启用 LIN 计数器溢出 (IMASK.LINOVF = 1) 来生成超时中断。当计数器溢出时，中断处理程序可以中止同步域分析并切换回中断检测。计数器溢出中断发生的时间可通过以下公式计算得出：$t_{Timeout} = 2^{16}$/UART 时钟。

备注

同步域自动存储在 RX FIFO 中，可能会误读为 PID。因此，应在接收到同步域之后并在接收到 PID 之前刷新 RX FIFO。

16.2.3.6.1 LIN 响应者传输延迟

用于启动响应者线路上的传输的中断 RXINT 始终发生在指挥官停止位的中点。根据 BUSCLK 和波特率，指挥官的 STOP 位和响应者的 START 位之间可能没有足够的响应空间，并且数据传输可能在 STOP 位结束之前开始。

为了克服该问题，可以添加一半 STOP 位周期的延迟作为响应空间时间，以确保在响应者数据传输开始之前，STOP 和 START 位之间有足够的延迟时间。

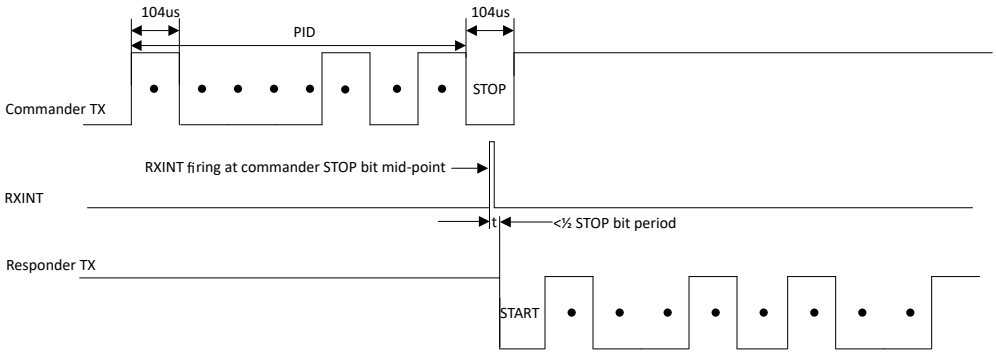


图 16-7. LIN 响应者传输延迟

16.2.3.7 流控

可通过硬件完成流量控制，以下各节介绍了实现方法。

在 UART 模式 (CTL0.MODE 设置为 0) 下，通过将 RTS 输出连接到接收器件上的 CTS 输入，并将接收器件上的 RTS 输出连接到 CTS 输入来完成两个器件之间的硬件流量控制。RTS 输出信号为低电平有效，CTS 输入对于发送请求预计会有一个低电平信号，如图 16-8 所示。

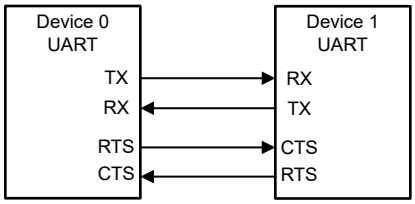


图 16-8. 流控

CTS 输入控制发送器，器件 0 和器件 1 发送器只有在其 CTS 输入为低电平有效时才能发送数据。启用 RTS 流量控制后，RTS 输出信号指示接收 FIFO 的状态。例如，在达到器件 0 的预编程 RX FIFO 水线位之前，器件 1 的 CTS 保持低电平有效，表明器件 0 的接收 FIFO 没有空间来存储额外的字符。

UART.CTL0 寄存器中的 CTSEN 和 RTSEN 位指定流量控制模式，如表 16-4 所示。

表 16-4. 流量控制启用

CTSEN	RTSEN	说明
1	1	启用 RTS 与 CTS 流量控制
1	0	仅启用 CTS 流量控制
0	1	仅启用 RTS 流量控制
0	0	禁用 RTS 和 CTS 流量控制

当 RTSEN 设置为 1 时，将忽略 CTL0.RTS 位的值，RTS 输出信号由硬件触发电平生成，如下所述。当 RTSEN 位清零时，RTS 信号输出由 CTL0.RTS 位控制以进行软件控制。

RTS 流量控制：

RTS 流量控制逻辑链接到可编程接收 FIFO 水线位，可以使用 UARTx.IFLS 寄存器对其进行配置。启用 RTS 流量控制后，RTS 将有效（低电平），直到接收 FIFO 填充至水线位。当达到接收 FIFO 水线位时，RTS 信号无效（高电平），表明没有更多空间来接收更多数据。预计数据传输将在发送当前字符后停止。当从接收 FIFO 中读出数据时，RTS 信号会重新有效（低电平），以便填充至水线位以下。如果禁用了 RTS 流量控制并且 UART 仍然启用，则会接收到数据，直到接收 FIFO 填满，或者不再有数据发送到接收 FIFO。

因为将最后接收到的字符放入 FIFO 而达到 FIFO 水线位时，RTS 信号无效。这意味着在背对背传输时，发送者可能已经开始了另一个字符传输。因此，在这种情况下，应将水线位设置为低一级，以确保可以接收所有数据并将其放入 FIFO。

CTS 流量控制：

如果启用了 CTS 流量控制，则发送器在发送下一个字节之前会检查 CTS 信号。如果 CTS 信号有效（低电平），则会发送字节，否则不会发送。当 CTS 有效（低电平）且发送 FIFO 不为空时，将继续发送数据。如果发送 FIFO 为空且 CTS 信号有效（低电平），则不发送数据。如果 CTS 信号无效（高电平）且启用了 CTS 流量控制，则完成当前字符传输后才会停止传输。如果禁用了 CTS 流量控制并启用了 UART，则将继续发送数据，直到发送 FIFO 为空。

16.2.3.8 空闲线多处理器

当设置 CTL0.MODE 寄存器位中的 IDLELINE 时，将选中空闲线多处理器格式。数据块会在发送或接收线路上以一段空闲时间隔开（图 16-9）。在一个字符的一个或两个停止位后，当接收到 10 个或更多的连续数据块（标志）时，一条空闲接收线路会被检测到。在接收到一条空闲线后，在下一次开始边沿被检测到前波特率发生器被切断。当检测到空闲线时，会设置 UARTx.STAT 中的 IDLE 位。在空闲线模式下，UART 接收器在无奇偶校验模式下运行，并且 UART 字长 (UARTx.LCRH.WLEN) 必须设置为 8 位。

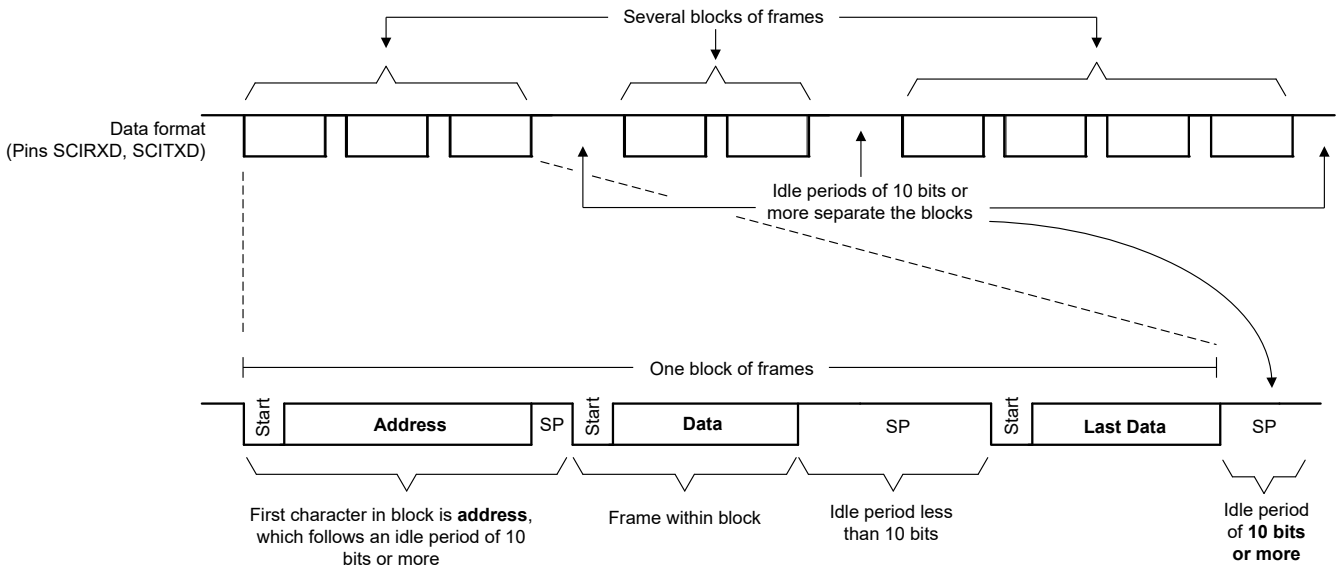


图 16-9. 空闲线多处理器

一段空闲时段后接收到的第一个字符为一个地址字符。UARTx.STAT 寄存器中的 IDLE 位用作每个字符块的地址标记。在空闲线多处理器模式下，当接收的一个字符是一个地址时该位就会被置 1。

如果接收到地址字符、则将其与 ADDR 寄存器进行比较、并应用 AMASK。如果接收到的字符匹配，地址字符和所有后续接收到的字符都会传输到 RXDATA 缓冲区，并产生中断/标志，直到接收到下一个没有匹配的地址。当地址匹配时，UARTx.STAT 寄存器中的 IDLE 位会自动清零；否则，将设置 IDLE 位，直到地址匹配。

当设置 UARTx.LCRH 寄存器中的 SENDIDLE 位时，UART 会在总线上插入一个 11 位的空闲周期，同时正在进行的传输会首先完成。下一个传输将会延迟，直到这个空闲周期结束。然后，下一个传输可以从地址字符开始。

以下程序通过发送一个空闲帧来表示一个地址字符及随后其关联的数据：

1. 设置 SENDIDLE，然后将地址字符写入 TXDATA。TXDATA 必须准备好接收新数据（TXINT 中断必须为 1）。这可以产生一个地址字符之后的 11 位空闲周期。当地址字符已经传输（所有位都从移位寄存器中发出）时，SENDIDLE 会自动复位。
2. 将所需的数据字符写入 TXDATA。TXDATA 必须准备好接收新数据（TXINT 中断必须为 1）。写入 TXDATA 中的数据会传输到移位寄存器，并会在移位寄存器准备好接收新数据后立即开始发送。

空闲周期不能超过地址和数据传输之间或者数据和数据传输的时间。否则，传输的数据将被误解为一个地址。

IDLELINE 模式下的 BUSY 位：

- TX：在生成 IDLELINE 信号以及发送地址和数据字节时，会设置 BUSY
- RX：在接收数据期间，会设置 BUSY 信号，直到接收到前 10 个空闲位。

16.2.3.9 9 位 UART 模式

将 UARTx.CTL0 寄存器中的 MODE 位设置为 ADDR9BIT，可启用 9 位模式。此功能在 UART 的多分支配置中非常有用，在这种配置下，连接到多个外设的单个控制器可以通过其地址或地址集以及地址字节的限定符与特定外设进行通信。在 9 位 UART 模式下，奇偶校验使能/模式位被忽略，并且 UART 字长 (UARTx.LCRH.WLEN) 必须设置为 8 位。

接收交易：

在 9 位模式下，外设在奇偶校验位的位置检查地址限定符。如果该位置位，则将接收到的字节与 UARTx.ADDR 寄存器中预编程的地址进行比较：

- 如果地址匹配，则会生成 9 位模式地址匹配中断 (ADDR_MATCH)；如果已启用，则会接收到更多数据。
- 如果地址不匹配，则丢弃该地址字节以及后续的所有数据字节。。

可以在 UART.ADDR 寄存器中预定义该地址，以匹配接收到的字节。通过使用 UART.AMASK 寄存器中的地址屏蔽，匹配范围可扩大为地址集。默认情况下，UART.AMASK 为 0xFF，表示仅匹配指定的地址

传输事务：

9 位 UART 模式下的所有发送事务都被解释为：

- 如果第 9 位被置位，则为地址字节
- 如果第 9 位被清零，则为数据字节

在 9 位模式下，第 9 位可由软件控制。LCRH 寄存器的 EPS 位设置反映了传输事务的第 9 位。为了指示地址字节，软件必须在字节传输前将 EPS 位置位。对于数据字节传输，在字节传输之前必须将 EPS 位清零。对于一个完整的传输事务，地址字节必须作为单字节传输进行发送，且 EPS 位被置位，随后是数据字节突发且 EPS 位被清零。

第 9 位处理：

表 16-5. 第 9 位处理

PEN	EPS	第 9 位（已传输或验证）
0	X	未传输或验证
1	0	0 (= Data)
1	1	1 (= Address)

16.2.3.10 RS485 支持

RS485 是串行通信系统中使用的一项标准。这项标准适用于长距离通信和有电噪声的环境。可以将多个接收器连接到这样的网络。这些特性使得 RS-485 在工业控制系统和类似应用中非常有用。

通过 RS485 方向信号可以控制外部 RS485 PHY。在此模式下会对方向信号使用 RTS I/O。一旦开始数据发送，该信号就会自动设置为高电平。如果是背靠背发送数据，将在字节之间保持该设置。如果正在接收数据，则应延迟新的发送，直至接收到该数据且方向信号已设置为发送。

图 16-10 所示的数据交换序列如下：

- 等待正在进行的接收完成。
- 激活在 RTS 引脚上进行发送的方向信号
- 发送数据（一个或多个字节）
- 等待正在进行的接收完成。
- 停用在 RTS 引脚上进行发送的方向信号

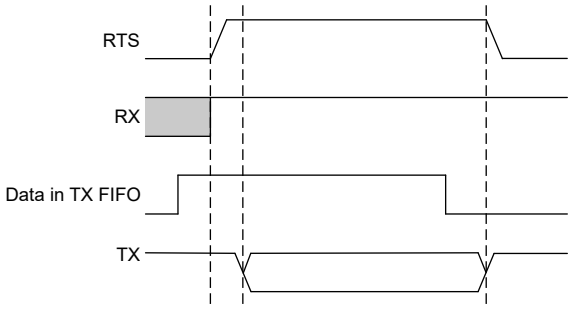


图 16-10. RS485 数据交换

LCRH 寄存器的两个位字段用于定义外部驱动器方向控制的设置和保持时间：

- EXTDIR_SETUP 位定义了由信号用于控制 RS485 外部驱动器的 UART 时钟节拍数将在发送 START 位之前设置。产生的设置时间将介于 EXDIR_SETUP 值和 EXDIR_SETUP + 一个波特率周期之间
- EXTDIR_HOLD 位定义了由信号用于控制 RS485 外部驱动器的 UART 时钟节拍数将在 STOP 位开始后复位。（如果启用了 2 个 STOP 位，则是在第二个 STOP 位开始后复位。）

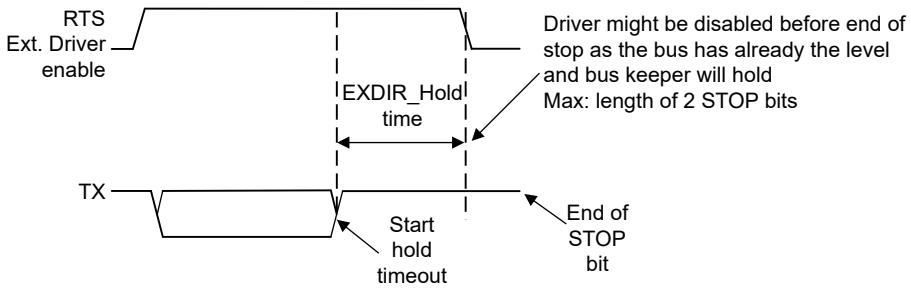


图 16-11. RS485 外部控制

16.2.3.11 DALI 协议

DALI 表示数字可寻址照明接口 (Digital Addressable Lighting Interface)。它是一个国际标准 (IEC 62386) 照明控制系统，为所有电子控制装置（光源）和电子控制器件（照明控制器）提供单一接口。

UART 模块支持底层 DALI 协议发送和接收用于前向帧和后向帧的位流。任何前向帧和后向帧序列之间的时序都需要由软件处理和检查。

传输前向帧或后向帧：

传输前向帧时，用户需要确保在第一个字节被移出之前将第二个字节写入缓冲区。然后，硬件将发送这两个字节，而不在两者之间插入停止位。否则，将发送停止位，像后向帧那样处理数据。

接收数据：

处于 DALI 模式的 UART 模块将检查起始位之后的第 9 位，以检测前向帧或后向帧。如果该位确实有相位变化（= 无停止位），则会检测到前向帧，并且：

- 地址比较在软件或硬件中完成，具体取决于 MASK
- 地址和数据始终传输到 RX 缓冲区中

否则会检测到一个后向帧并将数据传输到 RX 缓冲区中，而不设置 ADDR_MATCH 中断。

AMASK 寄存器可用作多播操作期间使用的组分配，通过 MSB 来指示器件是否为 DALI 组的一部分。要启用处于 DALI 模式的 UART 以响应所有地址，需要将 AMASK 寄存器清零。

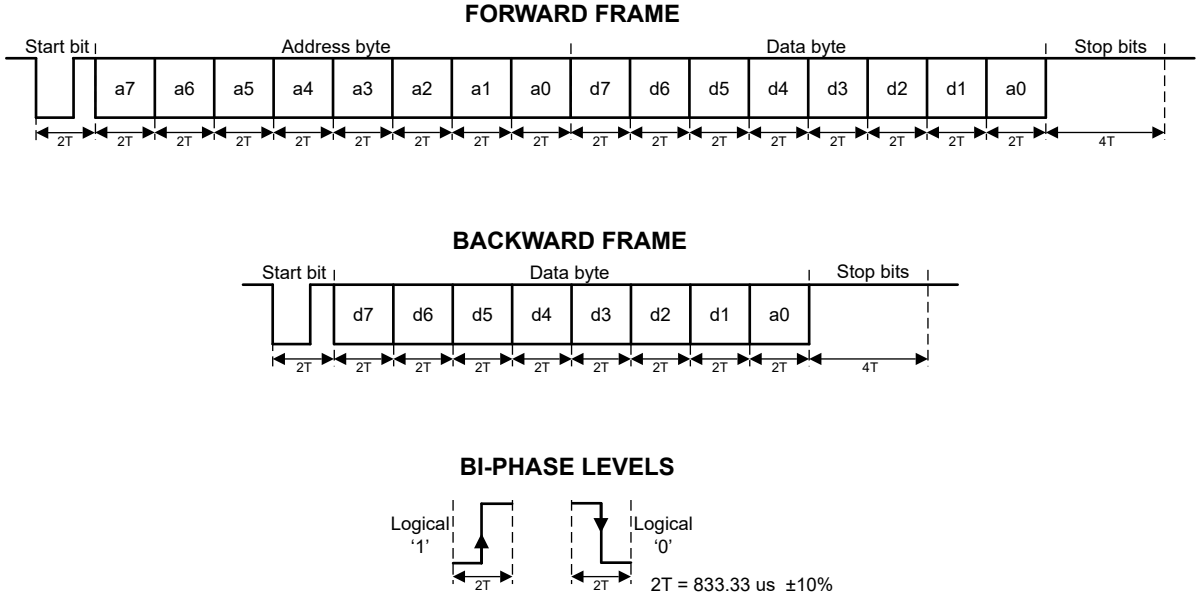


图 16-12. DALI 协议

根据标准 IEC 62386-102，时间 T 的限制应为：334 µ s < T < 500 µ s。DALI 模式需要启用曼彻斯特编码。使用 DALI 模式（CTL0.MODE 设置为 DALI）时，CTL0 和 LCRH 寄存器必须设置为：

- 8 位字长（WLEN 位）
- 无奇偶校验/2 停止位
- 启用曼彻斯特编码
- 波特率配置设置为匹配 2T = 833.33us
- DALI 模式需要启用 FIFO

16.2.3.12 曼彻斯特编码和解码

UART 提供接收和发送曼彻斯特编码数据的选项。该功能由 CTL0 寄存器中的 MENC 位启用，以生成符合 IEEE 802.3 标准的波形。借助 GPIO 控制模块中的反转功能，可以反转输出信号以生成符合 G. E. Thomas 标准的波形。

输出信号是通过将数据与 UART 时钟信号进行异或运算而生成的。UART 时钟的速度需要是波特率的两倍。因此，对于数据发送，每个数据位的开头和中间都有一个边沿。对于接收信号，检测到位中间的边沿以解码 RX 数据。

16.2.3.13 IrDA 编码和解码

当设置 UARTx.IRCTL 寄存器中的 IREN 位时，IrDA 编码器和解码器会启用，并为 IrDA 通信提供硬件位整形。IrDA 编码/解码只能用于 UART 模式（UARTx.CTL0.MODE 为 0）

IrDA 编码

在来自 UART 的发送位流中编码器为每个 0 位发送一个脉冲（请见图 16-13）。脉冲持续时间由 IRTXPL 位决定，该位用来指定由 IRTXCLK 选中的半时钟周期数。

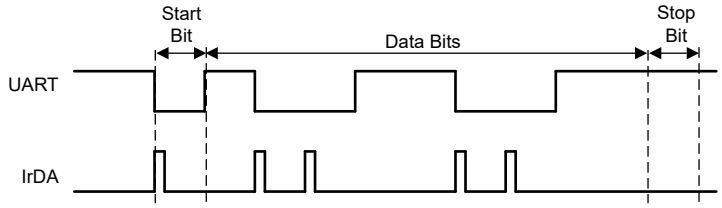


图 16-13. IrDA 协议

（将在下一个版本中添加更多信息）

IrDA 解码

当 IRRXPL = 0 时，解码器检测到高电平脉冲。否则就监测到低脉冲。

可编程数字滤波器级可通过设置 UARTx.GFCTL.DGFSEL > 0 来启用。当设置 IRCTL.IREN 时，还应设置数字干扰滤波器，以便仅传递长于编程滤波器长度的脉冲并丢弃较短的脉冲。（有关如何设置滤波器的信息，请参阅“干扰抑制”一章。）

16.2.3.14 ISO7816 智能卡支持

UART 为与 ISO7816 智能卡通信提供一些基本的支持。当配置 UARTx.CTL0 寄存器中智能卡对应的 MODE 位 (0x4) 时，TXD 信号用作位时钟，而 RXD 信号用作连接到智能卡的半双工通信线路。UART 不支持其他智能卡信号。

在 ISO7816 模式下，UART 时钟的时钟速率必须在 1MHz 至 5MHz 范围内。使用 ISO7816 模式时，UARTx.LCRH 寄存器必须设置为：

- 8 位字长（WLEN 位配置为 0x3）
- 偶校验（PEN 设置和 EPS 设置）
- 无固定校验（SPS 清除）

在 ISO7816 模式下，UART 会自动使用 2 个停止位；因此，LCRH 寄存器 STP2 位会被忽略。

如果传输期间检测到奇偶校验错误，RXD 将在第二个停止位期间被拉至低电平。在这种情况下，UART 将中止传输，清空传输 FIFO 并丢弃其中包含的所有数据。此外，它会生成一个奇偶校验错误中断，允许软件检测问题并启动受影响数据的重新传输，因为 UART 在这种情况下不支持自动重新传输。在出现奇偶校验错误时，UART 不支持自动重传。如果在发送过程中检测到奇偶校验错误，会中止所有后续发送操作，必须由软件来处理受影响字节或报文的重新发送。

在智能卡模式下，如果出现奇偶校验错误，接收器会将线路驱动为低电平，并且奇偶校验中断标志生效。发送器根据该位的值进行响应。

16.2.3.15 地址检测

UARTx.ADDR 寄存器用于设置应与接收地址字节匹配的具体地址。该寄存器与 UARTx.AMASK 配合使用，从而与接收到的地址字节匹配。仅考虑 AMASK 设置为“1”的位。因此，对于完整地址，AMASK 寄存器设置为 0xFF。此功能用于 DALI、UART 9 位或空闲线路模式。

表 16-6. 地址检测

条件	DALI 模式	空闲线路模式	9 位模式
地址匹配	地址和数据被移动到 RXDATA	地址和数据被移动到 RXDATA	地址和数据被移动到 RXDATA
地址不匹配	地址和数据将被丢弃	地址和数据将被丢弃	地址和数据将被丢弃

16.2.3.16 FIFO 操作

UART 有两个 FIFO，深度为 4 个条目，一个 FIFO 用于发送，另一个 FIFO 用于接收。可通过 UART 数据 (TXDATA/RXDATA) 寄存器访问 FIFO。RXDATA 寄存器的读取操作返回一个 12 位值，该值由 8 个数据位和 4 个错误标志组成。对 TXDATA 的写入操作将 8 位数据放入发送 FIFO 中。

复位后，两个 FIFO 默认都是禁用的，其表现如同 1 字节深的保持寄存器。可通过设置 UARTx.CTL0 中的 FEN 位来启用 FIFO。可通过 UARTx.STAT 寄存器和中断事件来监控 FIFO 状态。

而对空、满和溢出条件的监控则是由硬件来完成的

UARTx.STAT 寄存器包含空和满标志（TXFE、TXFF、RXFE 和 RXFF 位），而 CPU_INT.RIS 寄存器通过 OVRERR 位指示接收 FIFO 的超限状态。发送 FIFO 溢出没有指示器。如果写入操作溢出发送 FIFO，则该操作将丢失。如果 FIFO 被禁用，将根据 1 字节深的保持寄存器的状态设置空和满标志。当接收到的数据多于 FIFO 能够捕获的数据时，最早的数据将被接收到的数据覆盖。

通过 UARTx.IFLS 寄存器控制 FIFO 生成中断的触发点。两个 FIFO 可分别配置为不同的触发深度。发送 FIFO 的可用配置包括 3/4、1/2、1/4 和空，用于接收 FIFO 1/4、1/2、3/4 和满。例如，如果为接收 FIFO 选择了 3/4 选项，则 UART 在接收了 3 个数据字节之后会生成接收中断。复位结束后，两个 FIFO 均配置为在 1/2 标记处触发中断。在以下情况下，FIFO 完整性是不确定的：

- 在启动 UART 发送中断后 (LCRH.BRK = 1)
- 如果软件在传输过程中禁用 UART，并且 FIFO 中包含数据，则重新启用它。

16.2.3.17 回送操作

通过设置 UART.CTL0 寄存器中的 LBE 位，可以将 UART 置于内部环回模式，以开展诊断或调试工作。在环回模式下，UART 具有以下行为：

- 在 TX 输出上发送的数据在 RX 输入上接收
- 在 TX 输出上传输的数据不会传播到 TX IO 引脚
- 在 RX IO 引脚上接收到的数据被忽略

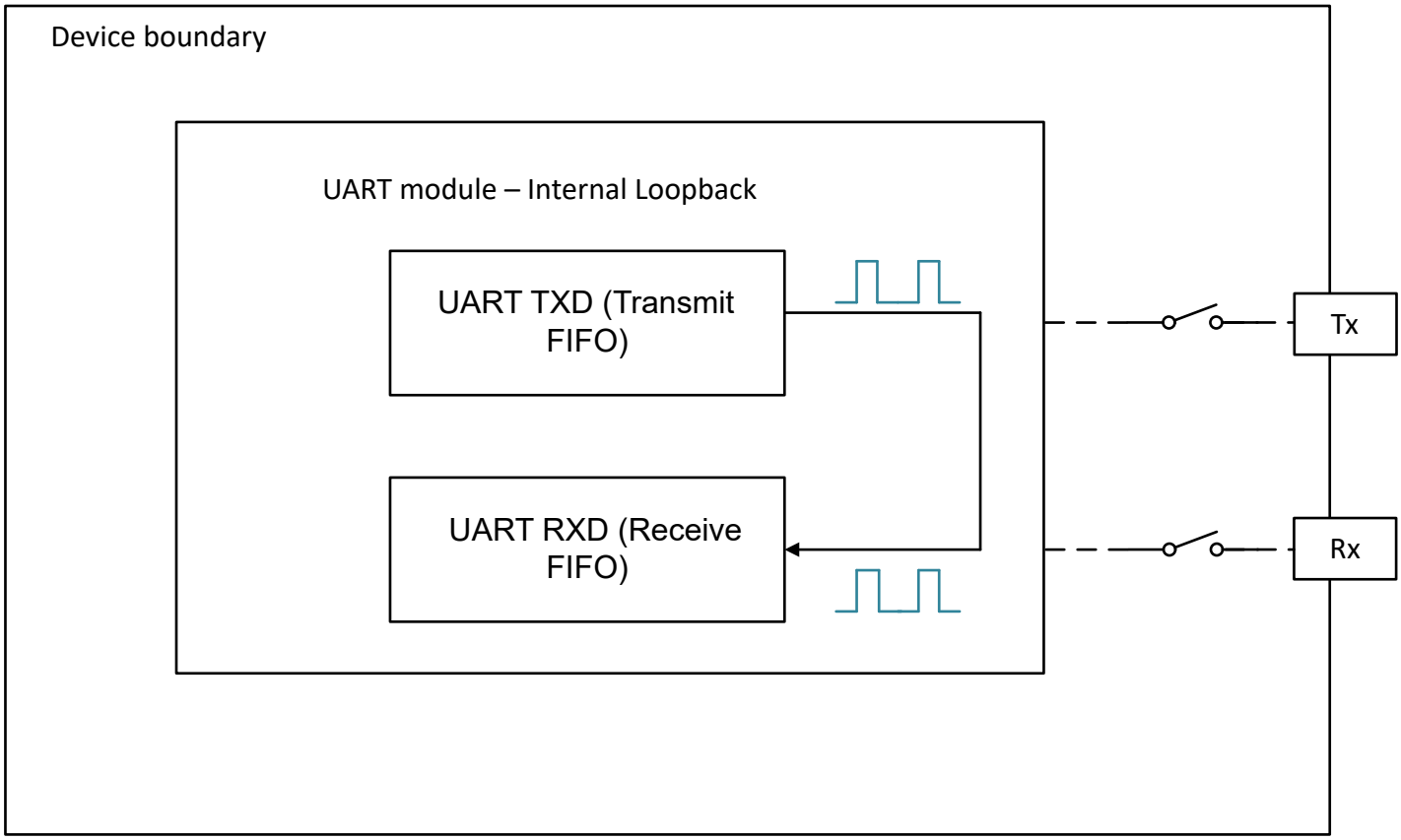


图 16-14. UART 环回模式

16.2.3.18 干扰抑制

数字滤波器

数字滤波器基于 UART 功能时钟。可以对 UARTx.GFCTL 寄存器中的 DGFSEL 位进行编程，以便为 RX 线路提供干扰抑制并确保信号值正确。干扰抑制值以功能时钟为单位。当干扰抑制为非零时，所有信号都将在内部延迟。例如，如果 DGFSEL 设置为 0x5，则在计算预期的事务时间时，应加上 5 个时钟。DGFSEL 需要配置为干扰抑制脉冲宽度小于正常数据脉冲的 1/3，以避免对正常脉冲进行意外滤波。

模拟滤波器

RX 线路上的模拟干扰抑制基于模拟干扰滤波器，并可通过 UARTx.GFCTL 寄存器中的 AGFSEL 位进行选择。有关可选择的干扰滤波器值，请参阅数据表。模拟干扰滤波器通过 AGFEN 使能，如果不设置，输入信号将在不经过滤波的情况下传递到 UART 模块。

16.2.4 低功耗运行

（将在下一个版本中添加更多信息）

16.2.5 复位注意事项

软件复位注意事项

可以通过同时设置 RSTCTL 寄存器中的 RESETASSERT 和 KEY 位来执行软件复位。正在进行的传输将立即终止，并可能使软件处于未定义状态。因此，在请求复位之前，应终止正在进行的传输。

硬件复位注意事项

硬件复位也会初始化 IO 配置。此过程会将 IO 设置为高阻抗状态，并且数据线可能会悬空。如果对于应用或 UART 接口上连接的器件至关重要，可能需要外部上拉或下拉电阻。

16.2.6 初始化

在 UART 设置或配置更改之前，应将 ENABLE 位清零，以避免在更新过程中或之后首次接收或发送数据时出现不可预知的行为。

要启用和初始化 UART，请执行以下步骤：

1. 使用 IOMUX 寄存器配置 RX 和 TX 引脚功能。
2. 使用 UARTx.RSTCTL 寄存器复位外设
3. 使用 UARTx.PWREN 寄存器启用 UART 外设的电源
4. 使用 UART.CLKSEL 和 UART.CLKDIV 寄存器选择 UART 功能时钟源和分频选项。
5. 通过将 UART.CTL0.ENABLE 位清零来禁用 UART。
6. 使用节 16.2.3.3 中的波特率公式来计算 UARTx.IBRD 和 UARTx.FBRD 寄存器。
7. 将 BRD 的整数部分写入 UART.IBRD 寄存器。
8. 将 BRD 的小数部分写入 UART.FBRD 寄存器。
9. 将所需的过采样和 FIFO 配置写入 UART.CTL0 寄存器
10. 将所需的串行参数写入 UART.LCRH 寄存器。
11. 通过设置 UART.CTL0.ENABLE 位来启用 UART。

16.2.7 中断和事件支持

UART 模块包含三个事件发布者而没有事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理对 CPU 子系统的 UART 中断请求 (IRQ)。第二个和第三个事件发布者（DMA_TRIG_RX、DMA_TRIG_TX）用于通过 DMA 事件路由设置 DMA 的触发信号。

表 16-7 中总结了 UART 事件。

表 16-7. UART 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	UART	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 UART 到 CPU 的中断路由
DMA 触发	发布者	UART	DMA	DMA 事件路由	DMA_TRIG_RX 寄存器	修复了从 UART 到 DMA 的中断路由
DMA 触发	发布者	UART	DMA	DMA 事件路由	DMA_TRIG_TX 寄存器	修复了从 UART 到 DMA 的中断路由

16.2.7.1 CPU 中断事件发布者 (CPU_INT)

UART 模块提供 18 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，UART 的 CPU 中断事件为：

表 16-8. UART CPU 中断事件条件 (CPU_INT)

IIDX STAT	名称	说明
0x01	RTOUT	UART 接收超时中断，当接收 FIFO 不为空且在 UARTx.IFLS.RXTOSEL 位指定的时间没有接收到更多数据时，此中断有效。下面提供了更多信息。
0x02	FRMERR	UART 帧错误中断，有关更多信息，请参阅节 16.2.3.5。
0x03	PARERR	UART 奇偶校验错误中断，有关更多信息，请参阅节 16.2.3.5。
0x04	BRKERR	UART 中断错误中断，有关更多信息，请参阅节 16.2.3.5。
0x05	OVRERR	UART 接收超限错误中断，有关更多信息，请参阅节 16.2.3.5。
0x06	RXNE	RX 中断的下降沿，当 RX 线上有下降沿时，触发该中断。
0x07	RXPE	RX 中断的上升沿，当 RX 线上有上升沿时，触发该中断。
0x08	LINC0	LIN Capture 0 匹配中断，该中断在 LIN 计数器中达到定义的 Capture 0 值时触发。
0x09	LINC1	LIN Capture 1 匹配中断，该中断在 LIN 计数器中达到定义的 Capture 1 值时触发。

表 16-8. UART CPU 中断事件条件 (CPU_INT) (continued)

IIDX STAT	名称	说明
0x0A	LINOVF	LIN 计数器溢出中断，该中断在 16 位 LIN 计数器溢出时触发。
0x0B	RXINT	UART 接收中断。下面提供了更多信息。
0x0C	TXINT	UART 发送中断。下面提供了更多信息。
0x0D	EOT	UART 发送结束中断，表示所有发送数据和状态的最后一位已离开串行器，TX FIFO 中没有任何其他数据。
0x0E	ADDR_MATCH	地址匹配中断，用于协议，通过地址表示发生了地址匹配。
0x0F	CTS	UART 清除发送中断，指示 CTS 信号状态。
0x10	DMA_DONE_RX	如果 RX DMA 通道发送 DONE 信号，则设置该中断。
0x11	DMA_DONE_TX	如果 TX DMA 通道发送 DONE 信号，则设置该中断。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

当接收 FIFO 不为空且在 IFLS.RXTOSEL 位指定的时间没有接收到更多数据时，此接收超时中断有效。当 FIFO 通过读取所有数据（或通过读取保持寄存器）变为空，从 IIDX 读取中断索引或当 ICLR 寄存器中的相应位写入 1 时，将清除接收超时中断。

发生以下事件之一时，接收中断（RXINT，0x0B）将更改状态：

- 如果启用 FIFO 且接收 FIFO 到达编程的触发电平，则设置 RXINT 位。通过从接收 FIFO 读取数据直至其低于触发电平，从 IIDX 读取中断索引或向 ICLR 中的 RXINT 位写入 1 来清除接收中断。
- 如果 FIFO 禁用（深度为一个位置）并且接收到数据从而填充该位置，则设置 RXINT 位。通过对接收 FIFO 执行单次读取，从 IIDX 读取中断索引或向 ICLR 中的 RXINT 位写入 1 来清除接收中断。

发生以下事件之一时，发送中断（TXINT，0x0C）将更改状态：

- 如果 FIFO 启用且发送 FIFO 超过编程的触发电平，则 TXINT 位置位。发送的数据量超过某一水平时将会触发发送中断信号，因此 FIFO 载入的数据量必须超过既定触发水平，否则不会再产生发送中断信号。通过向发送 FIFO 写入数据直至其高于触发电平，通过从 IIDX 读取中断索引或通过向 ICLR 中的 TXINT 位写入 1 来清除发送中断。
- 如果 FIFO 禁用（深度为一个位置）并且发送器单个位置中没有数据，则 TXINT 位置位。通过对发送 FIFO 执行单个写入操作，从 IIDX 读取中断索引或向 ICLR 中的 TXINT 位写入 1 可以将其清除。

16.2.7.2 DMA 触发发布者（DMA_TRIG_RX、DMA_TRIG_TX）

DMA_TRIG_RX 和 DMA_TRIG_TX 寄存器用于设置 DMA 的触发信号。这可以通过灵活的方式进行设置，使用表 16-9 和表 16-10 中的触发条件，来触发 DMA 执行接收或发送事件。

DMA_TRIG_RX 用于触发 DMA 以执行接收数据传输，DMA_TRIG_TX 用于触发 DMA 以执行发送数据传输。

表 16-9. UART DMA 触发条件 (DMA_TRIG_RX)

IIDX STAT	名称	说明
0x01	RTOUT	UART 接收超时中断，当接收 FIFO 不为空且在 UARTx.IFLS.RXTOSEL 位指定的时间没有接收到更多数据时，此中断有效。下面提供了更多信息。
0x0B	RXINT	UART 接收中断。下面提供了更多信息。

当接收 FIFO 不为空且在 IFLS.RXTOSEL 位指定的时间没有接收到更多数据时，此接收超时中断有效。当 FIFO 通过读取所有数据（或通过读取保持寄存器）变为空，从 IIDX 读取中断索引或当 ICLR 寄存器中的相应位写入 1 时，将清除接收超时中断。

发生以下事件之一时，接收中断（RXINT，0x0B）将更改状态：

- 如果启用 FIFO 且接收 FIFO 到达编程的触发电平，则设置 RXINT 位。通过从接收 FIFO 读取数据直至其低于触发电平，从 IIDX 读取中断索引或向 ICLR 中的 RXINT 位写入 1 来清除接收中断。

- 如果 FIFO 禁用（深度为一个位置）并且接收到数据从而填充该位置，则设置 RXINT 位。通过对接收 FIFO 执行单次读取，从 IIDX 读取中断索引或向 ICLR 中的 RXINT 位写入 1 来清除接收中断。

表 16-10. UART DMA 触发条件 (DMA_TRIG_TX)

IIDX STAT	名称	说明
0x0C	TXINT	UART 发送中断。下面提供了更多信息。

发生以下事件之一时，发送中断（TXINT，0x0C）将更改状态：

- 如果 FIFO 启用且发送 FIFO 超过编程的触发电平，则 TXINT 位置位。发送的数据量超过某一水平时将会触发发送中断信号，因此 FIFO 载入的数据量必须超过既定触发水平，否则不会再产生发送中断信号。通过向发送 FIFO 写入数据直至其高于触发电平，通过从 IIDX 读取中断索引或通过向 ICLR 中的 TXINT 位写入 1 来清除发送中断。
- 如果 FIFO 禁用（深度为一个位置）并且发送器单个位置中没有数据，则 TXINT 位置位。通过对发送 FIFO 执行单个写入操作，从 IIDX 读取中断索引或向 ICLR 中的 TXINT 位写入 1 可以将其清除。

通过 DMA_TRIG_RX 和 DMA_TRIG_TX 事件管理寄存器来管理 DMA 触发事件配置。有关配置事件寄存器的指导，请参阅节 7.2.5；有关 DMA 触发事件的工作原理，请参阅节 7.1.3.2。

16.2.8 仿真模式

器件处于调试模式时的模块行为由 PDBGCTL 寄存器中的 FREE 和 SOFT 位控制。

当器件处于调试模式并设置为停机模式时，可配置以下行为。

表 16-11. 调试模式外设行为

PDBGCTL.FREE	PDBGCTL.SOFT	功能
1	x	模块继续运行
0	0	模块立即停止
0	1	模块在下一次传输完成后停止

16.3 UART 寄存器

表 16-12 列出了 UART 寄存器的存储器映射寄存器。表 16-12 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 16-12. UART 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
808h	CLKCFG	外设时钟配置寄存器		转到
814h	STAT	状态寄存器		转到
1000h	CLKDIV	时钟分频器		转到
1008h	CLKSEL	超低功耗外设的时钟选择		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	DMA_TRIG_RX	转到
1058h	IMASK	中断屏蔽	DMA_TRIG_RX	转到
1060h	RIS	原始中断状态	DMA_TRIG_RX	转到
1068h	MIS	已屏蔽中断状态	DMA_TRIG_RX	转到
1070h	ISET	中断设置	DMA_TRIG_RX	转到
1078h	ICLR	中断清除	DMA_TRIG_RX	转到
1080h	IIDX	中断索引	DMA_TRIG_TX	转到
1088h	IMASK	中断屏蔽	DMA_TRIG_TX	转到
1090h	RIS	原始中断状态	DMA_TRIG_TX	转到
1098h	MIS	已屏蔽中断状态	DMA_TRIG_TX	转到
10A0h	ISET	中断设置	DMA_TRIG_TX	转到
10A8h	ICLR	中断清除	DMA_TRIG_TX	转到
10E0h	EVT_MODE	事件模式		转到
10E4h	INTCTL	中断控制寄存器		转到
1100h	CTL0	UART 控制寄存器 0		转到
1104h	LCRH	UART 线路控制寄存器		转到
1108h	STAT	UART 状态寄存器		转到
110Ch	IFLS	UART 中断 FIFO 级别选择寄存器		转到
1110h	IBRD	UART 整数波特率除数寄存器		转到
1114h	FBRD	UART 分数波特率除数寄存器		转到

表 16-12. UART 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
1118h	GFCTL	干扰滤波器控制		转到
1120h	TXDATA	UART 发送数据寄存器		转到
1124h	RXDATA	UART 接收数据寄存器		转到
1130h	LINCNT	UART LIN 模式计数器寄存器		转到
1134h	LINCTL	UART LIN 模式控制寄存器		转到
1138h	LINC0	UART LIN 模式捕获 0 寄存器		转到
113Ch	LINC1	UART LIN 模式捕获 1 寄存器		转到
1140h	IRCTL	eUSCI_Ax IrDA 控制字寄存器		转到
1148h	AMASK	自地址屏蔽寄存器		转到
114Ch	ADDR	自地址寄存器		转到
1160h	CLKDIV2	时钟分频器		转到

复杂的位访问类型经过编码可适应小型表单元。表 16-13 展示了适用于此部分中访问类型的代码。

表 16-13. UART 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

16.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 16-15 展示了 PWREN，表 16-14 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 16-15. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 16-14. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

16.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 16-16 展示了 RSTCTL，表 16-15 中对此进行了介绍。

返回到汇总表。

用于控制复位置为有效和无效的寄存器

图 16-16. RSTCTL

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

表 16-15. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	使外设复位有效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位置为有效

16.3.3 CLKCFG（偏移 = 808h）[复位 = 00000000h]

图 16-17 展示了 CLKCFG，表 16-16 中对此进行了介绍。

返回到汇总表。

外设时钟配置寄存器

图 16-17. CLKCFG

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							BLOCKASYNC
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 16-16. CLKCFG 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许状态更改的 KEY -- 0xA9 A9h = 0xA9
23-9	保留	R/W	0h	
8	BLOCKASYNC	R/W	0h	阻止异步时钟请求启动 SYSOSC 或强制总线时钟为 32MHz 0h = 0 1h = 1
7-0	保留	R/W	0h	

16.3.4 STAT（偏移 = 814h）[复位 = 00000000h]

图 16-18 展示了 STAT，表 16-17 中对此进行了介绍。

返回到汇总表。

复位状态寄存器

图 16-18. STAT

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

表 16-17. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

16.3.5 CLKDIV（偏移 = 1000h）[复位 = 00000000h]

图 16-19 展示了 CLKDIV，表 16-18 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块专用分频比

图 16-19. CLKDIV

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

表 16-18. CLKDIV 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 3 分频 3h = 对时钟源进行 4 分频 4h = 对时钟源进行 5 分频 5h = 对时钟源进行 6 分频 6h = 对时钟源进行 7 分频 7h = 对时钟源进行 8 分频

16.3.6 CLKSEL（偏移 = 1008h）[复位 = 00000000h]

图 16-20 展示了 CLKSEL，表 16-19 中对此进行了介绍。

返回到汇总表。

外设时钟源选择

图 16-20. CLKSEL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				BUSCLK_SEL	MFCLK_SEL	LFCLK_SEL	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 16-19. CLKSEL 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	BUSCLK_SEL	R/W	0h	如果启用，选择 BUS CLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
2	MFCLK_SEL	R/W	0h	如果启用，选择 MFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
1	LFCLK_SEL	R/W	0h	如果启用，选择 LFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
0	RESERVED	R/W	0h	

16.3.7 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 16-21 展示了 PDBGCTL，表 16-20 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 16-21. PDBGCTL

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

表 16-20. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设将阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

16.3.8 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 16-22 展示了 IIDX，表 16-21 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 16-22. IIDX

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

表 16-21. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	UART 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 RIS 和 MIS 寄存器中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 01h = UART 接收超时中断；中断标志： RT；中断优先级：最高 02h = UART 组帧错误中断；中断标志： FE 03h = UART 奇偶校验错误中断；中断标志： PE 04h = UART 中断错误中断；中断标志： BE 05h = UART 接收溢出错误中断；中断标志： OE 06h = UARTxRXD 负边沿中断；中断标志： RXNE 07h = UARTxRXD 正边沿中断；中断标志： RXPE 08h = LIN 捕获 0/匹配中断；中断标志： LINC0 09h = LIN 捕获 1 中断；中断标志： LINC1 0Ah = LIN 硬件计数器上溢中断；中断标志： LINOVF 0Bh = UART 接收中断；中断标志： RX 0Ch = UART 发送中断；中断标志： TX 0Dh = UART 发送结束中断（发送串行器为空）；中断标志： EOT 0Eh = 9 位模式地址匹配中断；中断标志： MODE_9B FH = UART 允许发送调制解调器中断；中断标志： CTS 10h = RX 上 DMA 完成 11h = TX 上 DMA 完成 12h = 噪声错误事件

16.3.9 IMASK（偏移 = 1028h）[复位= 00000000h]

图 16-23 展示了 IMASK，表 16-22 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 16-23. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED						NERR	DMA_DONE_TX
R/W-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
DMA_DONE_RX	CTS	ADDR_MATCH	EOT	TXINT	RXINT	LINOVF	LINC1
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
LINC0	RXPE	RXNE	OVRERR	BRKERR	PARERR	FRMERR	RTOUT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 16-22. IMASK 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R/W	0h	
17	NERR	R/W	0h	三重投票时的噪声误差。当 3 个多数表决样本不相等时置为有效 0h = 清除中断屏蔽 1h = 设置中断屏蔽
16	DMA_DONE_TX	R/W	0h	启用 TX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 设置中断屏蔽
15	DMA_DONE_RX	R/W	0h	启用 RX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 设置中断屏蔽
14	CTS	R/W	0h	启用 UART 允许发送调制解调器中断。 0h = 中断已禁用 1h = 设置中断屏蔽
13	ADDR_MATCH	R/W	0h	启用地址匹配中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
12	EOT	R/W	0h	启用 UART 发送结束中断。表示所有发送数据和标志的最后一位已离开串行器，TX FIFO 或缓冲器中没有任何其他数据。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
11	TXINT	R/W	0h	启用 UART 发送中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
10	RXINT	R/W	0h	启用 UART 接收中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 16-22. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
9	LINOVF	R/W	0h	启用 LIN 硬件计数器上溢中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	LINC1	R/W	0h	启用 LIN 捕获 1 中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	LINC0	R/W	0h	启用 LIN 捕获 0/匹配中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	RXPE	R/W	0h	启用 UARTxRXD 正边沿中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
5	RXNE	R/W	0h	启用 UARTxRXD 负边沿中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	OVRERR	R/W	0h	启用 UART 接收溢出错误中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	BRKERR	R/W	0h	启用 UART 中断错误中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	PARERR	R/W	0h	启用 UART 奇偶校验错误中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	FRMERR	R/W	0h	启用 UART 组帧错误中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RTOUT	R/W	0h	启用 UARTOUT 接收超时中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

16.3.10 RIS（偏移 = 1030h）[复位 = 00000000h]

图 16-24 展示了 RIS，表 16-23 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 16-24. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED						NERR	DMA_DONE_TX
R-						R-0h	R-0h
15	14	13	12	11	10	9	8
DMA_DONE_RX	CTS	ADDR_MATCH	EOT	TXINT	RXINT	LINOVF	LINC1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
LINC0	RXPE	RXNE	OVRERR	BRKERR	PARERR	FRMERR	RTOUT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 16-23. RIS 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R	0h	
17	NERR	R	0h	三重投票时的噪声误差。当 3 个多数表决样本不相等时置为有效 0h = 未发生中断 1h = 已发生中断
16	DMA_DONE_TX	R	0h	TX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 已发生中断
15	DMA_DONE_RX	R	0h	RX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 已发生中断
14	CTS	R	0h	UART 允许发送调制解调器中断。 0h = 中断已禁用 1h = 已发生中断
13	ADDR_MATCH	R	0h	地址匹配中断。 0h = 未发生中断 1h = 已发生中断
12	EOT	R	0h	UART 发送结束中断。表示所有发送数据和标志的最后一位已离开串行器，TX FIFO 或缓冲器中没有任何其他数据。 0h = 未发生中断 1h = 已发生中断
11	TXINT	R	0h	UART 发送中断。 0h = 未发生中断 1h = 已发生中断
10	RXINT	R	0h	UART 接收中断。 0h = 未发生中断 1h = 已发生中断

表 16-23. RIS 字段说明 (continued)

位	字段	类型	复位	说明
9	LINOVF	R	0h	LIN 硬件计数器上溢中断。 0h = 未发生中断 1h = 已发生中断
8	LINC1	R	0h	LIN 捕获 1 中断。 0h = 未发生中断 1h = 已发生中断
7	LINC0	R	0h	LIN 捕获 0/匹配中断。 0h = 未发生中断 1h = 已发生中断
6	RXPE	R	0h	UARTxRXD 正边沿中断。 0h = 未发生中断 1h = 已发生中断
5	RXNE	R	0h	UARTxRXD 负边沿中断。 0h = 未发生中断 1h = 已发生中断
4	OVRERR	R	0h	UART 接收溢出错误中断。 0h = 未发生中断 1h = 已发生中断
3	BRKERR	R	0h	UART 中断错误中断。 0h = 未发生中断 1h = 已发生中断
2	PARERR	R	0h	UART 奇偶校验错误中断。 0h = 未发生中断 1h = 已发生中断
1	FRMERR	R	0h	UART 组帧错误中断。 0h = 未发生中断 1h = 已发生中断
0	RTOUT	R	0h	UARTOUT 接收超时中断。 0h = 未发生中断 1h = 已发生中断

16.3.11 MIS (偏移 = 1038h) [复位 = 00000000h]

图 16-25 展示了 MIS，表 16-24 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 16-25. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED						NERR	DMA_DONE_TX
R-0h						R-0h	R-0h
15	14	13	12	11	10	9	8
DMA_DONE_RX	CTS	ADDR_MATCH	EOT	TXINT	RXINT	LINOVF	LINC1
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
LINC0	RXPE	RXNE	OVRERR	BRKERR	PARERR	FRMERR	RTOUT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 16-24. MIS 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R	0h	
17	NERR	R	0h	三重投票时的噪声误差。当 3 个多数表决样本不相等时置为有效 0h = 未发生中断 1h = 已发生中断
16	DMA_DONE_TX	R	0h	已屏蔽 TX 事件通道上 DMA 完成中断 0h = 未发生中断 1h = 已发生中断
15	DMA_DONE_RX	R	0h	已屏蔽 RX 事件通道上 DMA 完成中断 0h = 未发生中断 1h = 已发生中断
14	CTS	R	0h	已屏蔽 UART 允许发送调制解调器中断。 0h = 未发生中断 1h = 已发生中断
13	ADDR_MATCH	R	0h	已屏蔽地址匹配中断。 0h = 未发生中断 1h = 已发生中断
12	EOT	R	0h	UART 发送结束中断。表示所有发送数据和标志的最后一位已离开串行器，TX FIFO 或缓冲器中没有任何其他数据。 0h = 未发生中断 1h = 已发生中断
11	TXINT	R	0h	已屏蔽 UART 发送中断。 0h = 未发生中断 1h = 已发生中断
10	RXINT	R	0h	已屏蔽 UART 接收中断。 0h = 未发生中断 1h = 已发生中断

表 16-24. MIS 字段说明 (continued)

位	字段	类型	复位	说明
9	LINOVF	R	0h	已屏蔽 LIN 硬件计数器上溢中断。 0h = 未发生中断 1h = 已发生中断
8	LINC1	R	0h	已屏蔽 LIN 捕获 1 中断。 0h = 未发生中断 1h = 已发生中断
7	LINC0	R	0h	已屏蔽 LIN 捕获 0/匹配中断。 0h = 未发生中断 1h = 已发生中断
6	RXPE	R	0h	已屏蔽 UARTxRXD 正边沿中断。 0h = 未发生中断 1h = 已发生中断
5	RXNE	R	0h	已屏蔽 UARTxRXD 负边沿中断。 0h = 未发生中断 1h = 已发生中断
4	OVRERR	R	0h	已屏蔽 UART 接收溢出错误中断。 0h = 未发生中断 1h = 已发生中断
3	BRKERR	R	0h	已屏蔽 UART 中断错误中断。 0h = 未发生中断 1h = 已发生中断
2	PARERR	R	0h	已屏蔽 UART 奇偶校验错误中断。 0h = 未发生中断 1h = 已发生中断
1	FRMERR	R	0h	已屏蔽 UART 组帧错误中断。 0h = 未发生中断 1h = 已发生中断
0	RTOUT	R	0h	已屏蔽 UARTOUT 接收超时中断。 0h = 未发生中断 1h = 已发生中断

16.3.12 ISET (偏移 = 1040h) [复位 = 00000000h]

图 16-26 展示了 ISET，表 16-25 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 16-26. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED						NERR	DMA_DONE_TX
W-0h						W-0h	W-0h
15	14	13	12	11	10	9	8
DMA_DONE_RX	CTS	ADDR_MATCH	EOT	TXINT	RXINT	LINOVF	LINC1
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
LINC0	RXPE	RXNE	OVRERR	BRKERR	PARERR	FRMERR	RTOUT
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 16-25. ISET 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	W	0h	
17	NERR	W	0h	三重投票时的噪声误差。当 3 个多数表决样本不相等时置为有效 0h = 写入该位无效 1h = 设置中断
16	DMA_DONE_TX	W	0h	设置 TX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 设置中断
15	DMA_DONE_RX	W	0h	设置 RX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 设置中断
14	CTS	W	0h	设置 UART 允许发送调制解调器中断。 0h = 写入 0 无效 1h = 设置中断
13	ADDR_MATCH	W	0h	设置地址匹配中断。 0h = 写入 0 无效 1h = 设置中断
12	EOT	W	0h	设置 UART 发送结束中断。表示所有发送数据和标志的最后一位已离开串行器，TX FIFO 或缓冲器中没有任何其他数据。 0h = 写入 0 无效 1h = 设置中断
11	TXINT	W	0h	设置 UART 发送中断。 0h = 写入 0 无效 1h = 设置中断
10	RXINT	W	0h	设置 UART 接收中断。 0h = 写入 0 无效 1h = 设置中断

表 16-25. ISET 字段说明 (continued)

位	字段	类型	复位	说明
9	LINOVF	W	0h	设置 LIN 硬件计数器上溢中断。 0h = 写入 0 无效 1h = 设置中断
8	LINC1	W	0h	设置 LIN 捕获 1 中断。 0h = 写入 0 无效 1h = 设置中断
7	LINC0	W	0h	设置 LIN 捕获 0/匹配中断。 0h = 写入 0 无效 1h = 设置中断
6	RXPE	W	0h	设置 UARTxRXD 正边沿中断。 0h = 写入 0 无效 1h = 设置中断
5	RXNE	W	0h	设置 UARTxRXD 负边沿中断。 0h = 写入 0 无效 1h = 设置中断
4	OVRERR	W	0h	设置 UART 接收溢出错误中断。 0h = 写入 0 无效 1h = 设置中断
3	BRKERR	W	0h	设置 UART 中断错误中断。 0h = 写入 0 无效 1h = 设置中断
2	PARERR	W	0h	设置 UART 奇偶校验错误中断。 0h = 写入 0 无效 1h = 设置中断
1	FRMERR	W	0h	设置 UART 组帧错误中断。 0h = 写入 0 无效 1h = 设置中断
0	RTOUT	W	0h	设置 UARTOUT 接收超时中断。 0h = 写入 0 无效 1h = 设置中断

16.3.13 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 16-27 展示了 ICLR，表 16-26 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 16-27. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED						NERR	DMA_DONE_TX
W-0h						W-0h	W-0h
15	14	13	12	11	10	9	8
DMA_DONE_RX	CTS	ADDR_MATCH	EOT	TXINT	RXINT	LINOVF	LINC1
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
LINC0	RXPE	RXNE	OVRERR	BRKERR	PARERR	FRMERR	RTOUT
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 16-26. ICLR 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	W	0h	
17	NERR	W	0h	三重投票时的噪声误差。当 3 个多数表决样本不相等时置为有效 0h = 写入 0 无效 1h = 清除中断
16	DMA_DONE_TX	W	0h	清除 TX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 清除中断
15	DMA_DONE_RX	W	0h	清除 RX 事件通道上 DMA 完成中断 0h = 中断已禁用 1h = 清除中断
14	CTS	W	0h	清除 UART 允许发送调制解调器中断。 0h = 写入 0 无效 1h = 清除中断
13	ADDR_MATCH	W	0h	清除地址匹配中断。 0h = 写入 0 无效 1h = 清除中断
12	EOT	W	0h	清除 UART 发送结束中断。表示所有发送数据和标志的最后一位已离开串行器，TX FIFO 或缓冲器中没有任何其他数据。 0h = 写入 0 无效 1h = 清除中断
11	TXINT	W	0h	清除 UART 发送中断。 0h = 写入 0 无效 1h = 清除中断
10	RXINT	W	0h	清除 UART 接收中断。 0h = 写入 0 无效 1h = 清除中断

表 16-26. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
9	LINOVF	W	0h	清除 LIN 硬件计数器上溢中断。 0h = 写入 0 无效 1h = 清除中断
8	LINC1	W	0h	清除 LIN 捕获 1 中断。 0h = 写入 0 无效 1h = 清除中断
7	LINC0	W	0h	清除 LIN 捕获 0/匹配中断。 0h = 写入 0 无效 1h = 清除中断
6	RXPE	W	0h	清除 UARTxRXD 正边沿中断。 0h = 写入 0 无效 1h = 清除中断
5	RXNE	W	0h	清除 UARTxRXD 负边沿中断。 0h = 写入 0 无效 1h = 清除中断
4	OVRERR	W	0h	清除 UART 接收溢出错误中断。 0h = 写入 0 无效 1h = 清除中断
3	BRKERR	W	0h	清除 UART 中断错误中断。 0h = 写入 0 无效 1h = 清除中断
2	PARERR	W	0h	清除 UART 奇偶校验错误中断。 0h = 写入 0 无效 1h = 清除中断
1	FRMERR	W	0h	清除 UART 组帧错误中断。 0h = 写入 0 无效 1h = 清除中断
0	RTOUT	W	0h	清除 UARTOUT 接收超时中断。 0h = 写入 0 无效 1h = 清除中断

16.3.14 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 16-28 展示了 IIDX，表 16-27 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 16-28. IIDX

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

表 16-27. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	UART 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 UARTRIS 和 UARTMISC 中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 01h = UART 接收超时中断；中断标志： RT；中断优先级：最高 0Bh = UART 接收中断；中断标志： RX

16.3.15 IMASK（偏移 = 1058h）[复位= 00000000h]

图 16-29 展示了 IMASK，表 16-28 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 16-29. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留					RXINT	RESERVED	
R/W-0h					R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
ESERVED							RTOUT
R/W-0h							R/W-0h

表 16-28. IMASK 字段说明

位	字段	类型	复位	说明
31-11	保留	R/W	0h	
10	RXINT	R/W	0h	启用 UART 接收中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9-1	保留	R/W	0h	
0	RTOUT	R/W	0h	启用 UARTOUT 接收超时中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

16.3.16 RIS（偏移 = 1060h）[复位 = 00000000h]

图 16-30 展示了 RIS，表 16-29 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 16-30. RIS

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

表 16-29. RIS 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	R	0h	
10	RXINT	R	0h	UART 接收中断。 0h = 未发生中断 1h = 已发生中断
9-1	RESERVED	R	0h	
0	RTOUT	R	0h	UARTOUT 接收超时中断。 0h = 未发生中断 1h = 已发生中断

16.3.17 MIS（偏移 = 1068h）[复位 = 00000000h]

图 16-31 展示了 MIS，表 16-30 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 16-31. MIS

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

表 16-30. MIS 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	R	0h	
10	RXINT	R	0h	已屏蔽 UART 接收中断。 0h = 未发生中断 1h = 已发生中断
9-1	RESERVED	R	0h	
0	RTOUT	R	0h	已屏蔽 UARTOUT 接收超时中断。 0h = 未发生中断 1h = 已发生中断

16.3.18 ISET (偏移 = 1070h) [复位 = 00000000h]

图 16-32 展示了 ISET，表 16-31 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 16-32. ISET

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

表 16-31. ISET 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	W	0h	
10	RXINT	W	0h	设置 UART 接收中断。 0h = 写入 0 无效 1h = 设置中断
9-1	RESERVED	W	0h	
0	RTOUT	W	0h	设置 UARTOUT 接收超时中断。 0h = 写入 0 无效 1h = 设置中断

16.3.19 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 16-33 展示了 ICLR，表 16-32 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 16-33. ICLR

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

表 16-32. ICLR 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	W	0h	
10	RXINT	W	0h	清除 UART 接收中断。 0h = 写入 0 无效 1h = 清除中断
9-1	RESERVED	W	0h	
0	RTOUT	W	0h	清除 UARTOUT 接收超时中断。 0h = 写入 0 无效 1h = 清除中断

16.3.20 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 16-34 展示了 IIDX，表 16-33 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 16-34. IIDX

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

表 16-33. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	UART 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 UARTRIS 和 UARTMISC 中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 0Ch = UART 发送中断；中断标志： TX

16.3.21 IMASK（偏移 = 1088h）[复位= 00000000h]

图 16-35 展示了 IMASK，表 16-34 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 16-35. IMASK

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

表 16-34. IMASK 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11	TXINT	R/W	0h	启用 UART 发送中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
10-0	RESERVED	R/W	0h	

16.3.22 RIS（偏移 = 1090h）[复位 = 00000000h]

图 16-36 展示了 RIS，表 16-35 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 16-36. RIS

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

表 16-35. RIS 字段说明

位	字段	类型	复位	说明
31-12	RESERVED	R	0h	
11	TXINT	R	0h	UART 发送中断。 0h = 未发生中断 1h = 已发生中断
10-0	RESERVED	R	0h	

16.3.23 MIS（偏移 = 1098h）[复位 = 00000000h]

图 16-37 展示了 MIS，表 16-36 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 16-37. MIS

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

表 16-36. MIS 字段说明

位	字段	类型	复位	说明
31-12	RESERVED	R	0h	
11	TXINT	R	0h	已屏蔽 UART 发送中断。 0h = 未发生中断 1h = 已发生中断
10-0	RESERVED	R	0h	

16.3.24 ISET（偏移 = 10A0h）[复位 = 00000000h]

图 16-38 展示了 ISET，表 16-37 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 16-38. ISET

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

表 16-37. ISET 字段说明

位	字段	类型	复位	说明
31-12	RESERVED	W	0h	
11	TXINT	W	0h	设置 UART 发送中断。 0h = 写入 0 无效 1h = 设置中断
10-0	RESERVED	W	0h	

16.3.25 ICLR（偏移 = 10A8h）[复位 = 00000000h]

图 16-39 展示了 ICLR，表 16-38 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 16-39. ICLR

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

表 16-38. ICLR 字段说明

位	字段	类型	复位	说明
31-12	RESERVED	W	0h	
11	TXINT	W	0h	清除 UART 发送中断。 0h = 写入 0 无效 1h = 清除中断
10-0	RESERVED	W	0h	

16.3.26 EVT_MODE（偏移 = 10E0h）[复位 = 00000029h]

图 16-40 展示了 EVT_MODE，表 16-39 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 16-40. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
ESERVED		INT2_CFG		INT1_CFG		INT0_CFG	
R/W-		R-2h		R-2h		R-1h	

表 16-39. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	INT2_CFG	R	2h	none.DMA_TRIG_TX 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	INT1_CFG	R	2h	none.DMA_TRIG_RX 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	none.CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

16.3.27 INTCTL（偏移 = 10E4h）[复位 = 00000000h]

图 16-41 展示了 INTCTL，表 16-40 中对此进行了介绍。

返回到汇总表。

中断控制寄存器

图 16-41. INTCTL

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

表 16-40. INTCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	INTEVAL	W	0h	向该字段写入 1 会重新评估中断源。 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。

16.3.28 CTL0（偏移 = 1100h）[复位 = 00000038h]

图 16-42 展示了 CTL0，表 16-41 中对此进行了介绍。

返回到汇总表。

UART 控制寄存器
CTL0 寄存器是控制寄存器。除了启用发送位 (TXE) 和启用接收位 (RXE) 处于置位状态外，其他所有位都在复位后清零。要使能 UART 模块，必须将 UARTEN 置位。假如软件准备对 UART 模块的配置进行更改，则必须先将 UARTEN 清零，之后才能写入更改的配置内容。假如在接收或发送期间禁用 UART，则当前数据会话结束后，UART 模块才会停止运行。注：启用 UART 后不应更改 CTL0 寄存器，否则会产生无法预料的结果。建议按照以下顺序更改 CTL0 寄存器。
1.禁用 UART 模块；
2.等待当前数据会话（传输的字符）结束；
3.清除 UART 控制寄存器 CTL0 中的 FEN 位以清空发送 FIFO。
4.修改控制寄存器的内容；
5.重新使能 UART 模块。

图 16-42. CTL0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED				MSBFIRST	MAJVOTE	FEN	HSE
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
HSE	CTSEN	RTSEN	RTS	RESERVED	MODE		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
MENC	TXD_OUT	TXD_OUT_EN	TXE	RXE	LBE	RESERVED	ENABLE
R/W-0h	R/W-0h	R/W-1h	R/W-1h	R/W-1h	R/W-0h	R/W-0h	R/W-0h

表 16-41. CTL0 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	0h	
19	MSBFIRST	R/W	0h	最高有效位在前 该位对协议字节的发送和接收方式都有影响。 说明：用户需要将协议与该位的正确值进行匹配以首先发送 MSB 或 LSB。硬件引擎将完全根据该位发送字节。 0h = 在协议数据包中首先发送最低有效位 1h = 在协议数据包中首先发送最高有效位
18	MAJVOTE	R/W	0h	启用多数投票 启用多数表决后，使用三个中心位确定接收到的采样值。如果出现错误（即所有 3 位不相同），则会检测到噪声错误，并设置 RIS.NERR 位和寄存器 RXDATA.NERR。 16 倍过采样：使用位 7、8、9 8 倍过采样：使用位 3、4、5 禁用：使用单个采样值（中心值） 0h = 禁用多数表决 1h = 启用多数表决
17	FEN	R/W	0h	UART 启用 FIFO 0h = 禁用 FIFO（字符模式）。FIFO 变成 1 字节深的保持寄存器。 1h = 启用发送和接收 FIFO 缓冲器（FIFO 模式）。

表 16-41. CTL0 字段说明 (continued)

位	字段	类型	复位	描述
16-15	HSE	R/W	0h	高速位过采样使能 **注意：**位过采样会影响 UART 波特率配置。在 ISO7816 智能卡模式（设置 SMART 位）下，该位的状态对时钟生成没有影响。 0h = 16 倍过采样。 1h = 8 倍过采样。 2h = 3 倍过采样。启用 3 倍过采样后，不支持 IrDA、Manchester 和 DALI。
14	CTSEN	R/W	0h	启用允许发送 0h = 禁用 CTS 硬件流控制。 1h = 启用 CTS 硬件流控制。仅当 UARTxCTS 信号有效时才发送数据。
13	RTSEN	R/W	0h	启用硬件控制的“请求发送” 0h = 禁用 RTS 硬件流控制。 1h = 启用 RTS 硬件流控制。仅当接收 FIFO 具有可用条目时才请求数据（通过使 UARTxRTS 有效）。
12	RTS	R/W	0h	请求发送 如果设置 RTSEN，则硬件逻辑将使用 FIFO 填充级别或 TXDATA 缓冲器来控制 RTS 输出信号。 如果清除 RTSEN，则 RTS 输出由 RTS 位控制。该位是 UART“请求发送”的补码，即 RTS 调制解调器状态输出。 0h = 信号非 RTS 1h = 信号是 RTS
11	RESERVED	R/W	0h	
10-8	MODE	R/W	0h	设置使用的通信模式和协议。 （未定义的设置使用默认设置：0） 0h = 正常运行 1h = RS485 模式： UART 需要处于空闲状态并在 EXTDIR_HOLD 设置时间内接收数据。EXTDIR_SETUP 定义在发送之前将 RTS 线设置为高电平的时间。当缓冲器为空时，RTS 线再次设置为低电平。只要 UART 正在接收数据，发送就会延迟。 2h = UART 在空闲线路模式下运行 3h = UART 在 9 位地址模式下运行 4h = ISO7816 智能卡支持。使用 ISO7816 模式时，应用程序必须确保其在 UARTLCRH 中设置 8 位字长（WLEN 设置为 3h）和偶校验（PEN 设置为 1，EPS 设置为 1，SPS 设置为 0）。 在此模式下会忽略 UARTLCRH 中 STP2 位的值并强制采用 2 个停止位。 5h = DALI 模式：
7	MENC	R/W	0h	曼彻斯特编码使能 0h = 禁用曼彻斯特编码 1h = 启用曼彻斯特编码
6	TXD_OUT	R/W	0h	TXD 引脚控制。当 TXD_OUT_EN = 1 且 TXE = 0 时控制 TXD 引脚。 0h = TXD 引脚为低电平 1h = TXD 引脚为高电平
5	TXD_OUT_EN	R/W	1h	TXD 引脚控制使能。当 UART 的发送部分被禁用 (TXE = 0) 时，TXD 引脚可由 TXD_OUT 位控制。 0h = TXD 引脚不能由 TXD_OUT 控制 1h = TXD 引脚可由 TXD_OUT 控制
4	TXE	R/W	1h	UART 发送使能。如果在发送过程中禁用 UART，则会在发送完当前字符后再停止。**注：**要启用发送，必须设置 UARTEN 位。 0h = 禁用 UART 的发送部分。启用后，UART 的 UARTxTXD 引脚可由 TXD_CTL 位控制。 1h = 启用 UART 的发送部分。
3	RXE	R/W	1h	UART 接收使能。如果在接收过程中禁用 UART，则会在接收完当前字符后再停止。**注：**要启用接收，必须设置 UARTEN 位。 0h = 禁用 UART 的接收部分。 1h = 启用 UART 的接收部分。

表 16-41. CTL0 字段说明 (continued)

位	字段	类型	复位	描述
2	LBE	R/W	0h	UART 环回使能 0h = 正常运行。 1h = UARTxTX 路径在内部通过 UARTxRX 路径馈送。
1	RESERVED	R/W	0h	
0	ENABLE	R/W	0h	UART 模块使能。假如在收发过程中禁用 UART，那么仍然会完成当前数据会话后再停止 UART 模块。 如果未设置 ENABLE 位，仍然可以访问和更新所有寄存器。建议设置并更改 UART 工作模式，同时清除 ENABLE 位，以免在设置或更新过程中出现不可预知的行为。 如果禁用，UART 模块不会发送或接收任何数据，并且逻辑将保持在复位状态。 0h = 禁用模块 1h = 启用模块

16.3.29 LCRH（偏移 = 1104h）[复位 = 00000000h]

图 16-43 展示了 LCRH，表 16-42 中对此进行了介绍。

返回到汇总表。

UART 线路控制寄存器。LCRH 寄存器是线路控制寄存器。数据长度、奇偶校验位、停止位等串行通讯参数都是通过本寄存器设置的。更新波特率除数（UARTIBRD 或 UARTIFRD）时，还必须写入 LCRH 寄存器。波特率除数寄存器的写入选通与 LCRH 寄存器相关联。

图 16-43. LCRH

31	30	29	28	27	26	25	24
RESERVED						EXTDIR_HOLD	
R/W-0h						R/W-0h	
23	22	21	20	19	18	17	16
EXTDIR_HOLD			EXTDIR_SETUP				
R/W-0h			R/W-0h				
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
SENDIDLE	SPS	WLEN		STP2	EPS	PEN	BRK
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 16-42. LCRH 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R/W	0h	
25-21	EXTDIR_HOLD	R/W	0h	定义由信号用于控制 RS485 外部驱动器的 UARTclk 时钟节拍数将在 STOP 位开始后复位。（如果启用了 2 个 STOP 位，则是在第二个 STOP 位开始后复位。） 0h = 最小值 1Fh = 尽可能高的值
20-16	EXTDIR_SETUP	R/W	0h	定义由信号用于控制 RS485 外部驱动器的 UARTclk 时钟节拍数将在发送 START 位之前设置 0h = 最小值 1Fh = 尽可能高的值
15-8	保留	R/W	0h	
7	SENDIDLE	R/W	0h	UART 发送空闲模式。设置该位后，TX 线上将发送 11 位时间的 SENDIDLE 周期。之后该位由硬件清除。 0h = 禁用发送空闲模式 1h = 启用发送空闲模式
6	SPS	R/W	0h	UART 粘着奇偶校验选择 粘着奇偶校验选择 (SPS) 位用于在发送或接收数据时将奇偶校验设置为永久“1”或永久“0”，其目的通常是指示包的第一个字节或者是标记地址字节（例如在多点 RS-485 网络中）。 设置 UARTLCRH 的 PEN、EPS 和 SPS 位后，会发送奇偶校验位，且校验结果为 0。 设置 PEN 和 SPS 位并清除 EPS 后，会发送奇偶校验位，且校验结果为 1。 0h = 禁用粘着奇偶校验 1h = 启用粘着奇偶校验

表 16-42. LCRH 字段说明 (continued)

位	字段	类型	复位	说明
5-4	WLEN	R/W	0h	UART 字长。这些位指示帧中发送或接收的数据位数，如下所示： 0h = 5 位（默认值） 1h = 6 位 2h = 7 位 3h = 8 位
3	STP2	R/W	0h	UART 两个停止位选择。在 7816 智能卡模式（设置 UARTCTL 寄存器中的 SMART 位）下，强制采用 2 个停止位。 0h = 在一帧结束时发送一个停止位。 1h = 在一帧结束时发送两个停止位。接收逻辑检查是否接收到两个停止位，如果其中一个停止位无效，则提供帧错误。
2	EPS	R/W	0h	UART 偶校验选择。当 PEN 位禁用奇偶检验时，该位无效。对于 9 位 UART 模式发送，该位控制地址字节和数据字节指示（第 9 位）。0 = 传输的字节为数据字节。1 = 传输的字节为地址字节。 0h = 执行奇校验，检查是否存在奇数个 1。 1h = 在发送和接收期间执行偶校验生成和检查，检查数据位和奇偶校验位中是否存在偶数个 1。
1	PEN	R/W	0h	UART 奇偶校验使能 0h = 禁用奇偶校验，不向数据帧添加奇偶校验位。 1h = 启用奇偶校验检查和生成。
0	BRK	R/W	0h	UART 发送中断（用于 LIN 协议） 0h = 正常使用。 1h = 当前字符发送完毕后，UARTxTXD 信号持续输出低电平。为了正确执行中止命令，软件必须令该位处于置位状态，并且持续至少 2 帧（字符周期）。

16.3.30 STAT（偏移 = 1108h）[复位 = 00000144h]

图 16-44 展示了 STAT，表 16-43 中对此进行了介绍。

返回到汇总表。

UART 状态寄存器

图 16-44. STAT

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED							
R-							
15	14	13	12	11	10	9	8
保留						IDLE（闲置）	CTS
R-						R-0h	R-1h
7	6	5	4	3	2	1	0
TXFF	TXFE	RESERVED		RXFF	RXFE	RESERVED	BUSY
R-0h	R-1h	R-		R-0h	R-1h	R-	R-0h

表 16-43. STAT 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R	0h	
9	IDLE（闲置）	R	0h	空闲线路多处理器模式下检测到空闲模式。 IDLE 位用作每个字符块的地址标签。在空闲线多处理器模式下，当接收的一个字符是一个地址时该位就会被置 1。 0h = 在最后一个接收到的字符之前未检测到空闲（在空闲线路多处理器模式下）。 1h = 在最后一个接收到的字符之前检测到空闲（在空闲线路多处理器模式下）。
8	CTS	R	1h	允许发送 0h = CTS 信号无效（高电平）。 1h = CTS 信号有效（低电平）。
7	TXFF	R	0h	UART 发送 FIFO 已满。该位的含义取决于 CTL0 寄存器中 FEN 位的状态。 0h = 发送器未满。 1h = 如果 FIFO 处于禁用状态（FEN 为 0），则发送保持寄存器已满。如果 FIFO 处于启用状态（FEN 为 1），则发送 FIFO 为满。
6	TXFE	R	1h	UART 发送 FIFO 为空。该位的含义取决于 CTL0 寄存器中 FEN 位的状态。 0h = 发送器有数据要发送。 1h = 如果 FIFO 处于禁用状态（FEN 为 0），则发送保持寄存器为空。若 FIFO 被启用（FEN = 1），则表明发送 FIFO 为空。
5-4	RESERVED	R	0h	
3	RXFF	R	0h	UART 接收 FIFO 已满。该位的含义取决于 CTL0 寄存器中 FEN 位的状态。 0h = 接收器可以接收数据。 1h = 如果 FIFO 处于禁用状态（FEN 为 0），则接收保持寄存器已满。若 FIFO 被启用（FEN = 1），则表明接收 FIFO 已满。

表 16-43. STAT 字段说明 (continued)

位	字段	类型	复位	说明
2	RXFE	R	1h	UART 接收 FIFO 为空。该位的含义取决于 CTL0 寄存器中 FEN 位的状态。 0h = 接收器不为空。 1h = 如果 FIFO 处于禁用状态（FEN 为 0），则接收保持寄存器为空。如果 FIFO 处于启用状态（FEN 为 1），则接收 FIFO 为空。
1	RESERVED	R	0h	
0	忙	R	0h	UART 忙 一旦发送 FIFO 或 TXDATA 寄存器变为非空状态（无论是否启用 UART），或者如果当前正在接收数据（在检测到起始边沿后，直到移位寄存器接收到包括所有停止位在内的完整字节），就会设置该位。 在 IDLE_Line 模式下，忙信号在空闲时间生成期间也保持设置状态。 0h = UART 未处于忙状态。 1h = UART 正忙于发送数据。该位会保持设置状态，直至包括所有停止位在内的全部字节都从移位寄存器发出或接收到移位寄存器中为止。

16.3.31 IFLS（偏移 = 110Ch）[复位 = 00000022h]

图 16-45 展示了 IFLS，表 16-44 中对此进行了介绍。

返回到汇总表。

IFLS 寄存器是中断 FIFO 级别选择寄存器。该寄存器可用于定义触发 TX、RX 和超时中断标志的电平。中断是在 FIFO 深度从不满足触发条件到满足触发条件的跳变沿产生的。简单来说，中断是在 FIFO 深度越过触发门限时产生的。例如，如果接收触发电平设置为中途标志，则会在接收 FIFO 中填充了两个或多个字符时触发中断。复位后，TXIFLSEL 和 RXIFLSEL 位域均默认设置为 1/2 触发深度。

图 16-45. IFLS

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留				RXTOSEL			
R/W-0h				R/W-0h			
7	6	5	4	3	2	1	0
ESERVED	RXIFLSEL			RESERVED	TXIFLSEL		
R/W-0h	R/W-2h			R/W-0h	R/W-2h		

表 16-44. IFLS 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11-8	RXTOSEL	R/W	0h	UART 接收中断超时选择。如果在设置的位时间内没有接收到额外字符的起始边沿，即使未达到 FIFO 级别，也会设置 RX 中断。值为 0 将禁用此功能。 0h = 最小值 Fh = 尽可能高的值
7	RESERVED	R/W	0h	
6-4	RXIFLSEL	R/W	2h	UART 接收中断 FIFO 级别选择。接收中断的触发点如下： 注意： 在 ULP 域中，触发电平用于： 0：LVL_1_4 4：LVL_FULL 对于未定义的设置，使用默认配置。 0h = RX FIFO >= 1/4 满 注意：对于 ULP 域 1h = RX FIFO >= 1/4 满 2h = RX FIFO >= 1/2 满（默认值） 3h = RX FIFO >= 3/4 满 4h = RX FIFO 已满 注意： 对于 ULP 域 5h = RX FIFO 已满 7h = RX FIFO >= 1 个可用条目 注意：尤其对于 DMA 触发而言是必需的设置
3	RESERVED	R/W	0h	

表 16-44. IFLS 字段说明 (continued)

位	字段	类型	复位	说明
2-0	TXIFLSEL	R/W	2h	UART 发送中断 FIFO 级别选择。发送中断的触发点如下： 注意：对于未定义的设置，使用默认配置。 1h = TX FIFO <= 3/4 空 2h = TX FIFO <= 1/2 空 (默认值) 3h = TX FIFO <= 1/4 空 5h = TX FIFO 为空 7h = TX FIFO >= 1 个可用条目 注意：尤其对于 DMA 触发而言是必需的设置

16.3.32 IBRD（偏移 = 1110h）[复位 = 00000000h]

图 16-46 展示了 IBRD，表 16-45 中对此进行了介绍。

返回到汇总表。

更改 IBRD 寄存器时，直到发送/接收当前字符结束后，新的值才会生效。对波特率分频系数进行任何修改后，必须写一次 UARTLCRH 寄存器。有关配置详细信息，请参阅“波特率生成”一章。

图 16-46. IBRD

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

表 16-45. IBRD 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	DIVINT	R/W	0h	整数波特率除数 0h = 最小值 FFFFh = 尽可能高的值

16.3.33 FBRD（偏移 = 1114h）[复位 = 00000000h]

图 16-47 展示了 FBRD，表 16-46 中对此进行了介绍。

返回到汇总表。

UART 分数波特率除数寄存器。FBRD 寄存器是波特率除数值的小数部分。复位时本寄存器将清零。更改 FBRD 寄存器时，直到发送/接收当前字符结束后，新的值才会生效。对波特率分频系数进行任何修改后，必须写一次 UARTLCRH 寄存器。有关配置详细信息，请参阅“波特率生成”一章。

图 16-47. FBRD

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

表 16-46. FBRD 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-0	DIVFRAC	R/W	0h	分数波特率除数 0h = 最小值 3Fh = 尽可能高的值

16.3.34 GFCTL（偏移 = 1118h）[复位 = X]

图 16-48 展示了 GFCTL，表 16-47 中对此进行了介绍。

返回到汇总表。

该寄存器控制 RX 输入端的干扰滤波器。

图 16-48. GFCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留				CHAIN	AGFSEL		AGFEN
R/W-0h				R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
ESERVED		DGFSEL					
R/W-0h		R/W-0h					

表 16-47. GFCTL 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11	CHAIN	R/W	0h	模拟和数字噪声滤波器链接使能。 0 禁用：为 0 时，禁用链接，只有数字滤波器输出可供 IP 逻辑用于进行采样 1 启用：为 1 时，会将模拟和数字干扰滤波器链接在一起，并将该组合的输出提供给 IP 逻辑进行采样 0h = 禁用 1h = 启用
10-9	AGFSEL	R/W	0h	模拟干扰抑制脉宽 该字段可控制 RX 线上用于进行模拟干扰抑制的脉宽选择。 请参阅器件数据表以了解确切值。 0h = 过滤掉长度短于 5ns 的脉冲。 1h = 过滤掉长度短于 10ns 的脉冲。 2h = 过滤掉长度短于 25ns 的脉冲。 3h = 过滤掉长度短于 50ns 的脉冲。
8	AGFEN	R/W	0h	模拟干扰抑制使能 0h = 禁用模拟干扰滤波器 1h = 启用模拟干扰滤波器
7-6	RESERVED	R/W	0h	
5-0	DGFSEL	R/W	0h	干扰抑制脉宽 该字段可控制 RX 线上用于进行干扰抑制的脉宽选择。 该字段中编程的值指定在 RX 线上抑制干扰的最长功能时钟周期数。 在 IRDA 模式下： 接收的最小脉冲长度由下式给出： t(MIN) = (DGFSEL) / f(IRTXCLK) 0h = 旁路 GF 3Fh = 尽可能高的值

16.3.35 TXDATA（偏移 = 1120h）[复位 = 00000000h]

图 16-49 展示了 TXDATA，表 16-48 中对此进行了介绍。

返回到汇总表。

UART 发送数据寄存器。该寄存器是发送数据寄存器（FIFO 的接口）。当发送数据时，若 FIFO 已使能，则对本寄存器写入的数据将推入发送 FIFO。如果发送 FIFO 被禁用，则数据仅保存在发送保持寄存器（即发送 FIFO 的最底部单元）中。对本寄存器执行写操作即会启动 UART 的发送。

图 16-49. TXDATA

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

表 16-48. TXDATA 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	数据	R/W	0h	发送或接收的数据。要通过 UART 发送的数据会写入该字段。读此位域时，将返回 UART 收到的数据。 0h = 最小值 FFh = 尽可能高的值

16.3.36 RXDATA（偏移 = 1124h）[复位 = 00000000h]

图 16-50 展示了 RXDATA，表 16-49 中对此进行了介绍。

返回到汇总表。

UART 接收数据寄存器。该寄存器是数据接收寄存器（FIFO 的接口）。当接收数据时，若 FIFO 已使能，则收到的数据字节以及 4 个状态位（线中止错误、帧错误、奇偶校验错误、溢出错误）将推入 12 位宽的接收 FIFO 中。如果接收 FIFO 被禁用，则数据字节和状态位保存在接收保持寄存器（即接收 FIFO 的最底部单元）中。对本寄存器的读操作即可获取数据字节。

图 16-50. RXDATA

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留			NERR	OVRERR	BRKERR	PARERR	FRMERR
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
数据							
R-0h							

表 16-49. RXDATA 字段说明

位	字段	类型	复位	说明
31 - 13	RESERVED	R	0h	
12	NERR	R	0h	噪声错误。 向该位写入无效。向 UART EVENT ICLR 寄存器中的 NERR 位写入 1 即可清除该标志。 0h = 未发生噪声错误 1h = 在多数表决期间发生噪声错误
11	OVRERR	R	0h	UART 接收溢出错误。向该位写入无效。向 UART EVENT ICLR 寄存器中的 OVRERR 位写入 1 即可清除该标志。在发生接收 FIFO 上溢的情况下，FIFO 的内容将保持有效，因为当 FIFO 已满时不会再写入任何数据。只会覆盖移位寄存器的内容。此时，CPU 必须读取数据以便将 FIFO 清空。 0h = 没有数据因接收溢出而丢失。 1h = 接收到新数据但无法存储，因为没有读取先前的数据（导致数据丢失）。
10	BRKERR	R	0h	UART 中断错误。向该位写入无效。向 UART EVENT ICLR 寄存器中的 BRKERR 位写入 1 即可清除该标志。该错误与 FIFO 顶部的字符相关。发生中止时，只有一个 0 字符被加载到 FIFO。仅在接收数据输入变为 1（标记状态）且接收到下一个有效的起始位后，才启用下一字符。 0h = 未发生中断情况 1h = 检测到中断情况，表示接收数据输入保持低电平的时间过长，超过一个完整字的发送时间（包含起始位、数据位、奇偶校验位、停止位）。
9	PARERR	R	0h	UART 奇偶校验错误。向该位写入无效。向 UART EVENT ICLR 寄存器中的 PARERR 位写入 1 即可清除该标志。 0h = 未发生奇偶校验错误 1h = 接收到的数据字符的奇偶性与 UARTLCRH 寄存器的位 2 和位 7 所定义的奇偶性不匹配。

表 16-49. RXDATA 字段说明 (continued)

位	字段	类型	复位	说明
8	FRMERR	R	0h	UART 组帧错误。向该位写入无效。向 UART EVENT ICLR 寄存器中的 FRMERR 位写入 1 即可清除该标志。该错误与 FIFO 顶部的字符相关。 0h = 未发生组帧错误 1h = 接收到的字符没有有效的停止位序列，即一个或两个停止位，具体取决于 UARTLCRH.STP2 设置 (有效的停止位为 1) 。
7-0	数据	R	0h	接收的数据。读此位域时，将返回 UART 收到的数据。 0h = 最小值 FFh = 尽可能高的值

16.3.37 LINCNT（偏移 = 1130h）[复位 = 00000000h]

图 16-51 展示了 LINCNT，表 16-50 中对此进行了介绍。

返回到汇总表。

UART LIN 模式计数器寄存器

图 16-51. LINCNT

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

表 16-50. LINCNT 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	值	R/W	0h	由 UART 的功能时钟计时的 16 位向上计数器。 0h = 最小值 FFFFh = 尽可能高的值

16.3.38 LINCTL（偏移 = 1134h）[复位 = 00000000h]

图 16-52 展示了 LINCTL，表 16-51 中对此进行了介绍。

返回到汇总表。

UART LIN 模式控制寄存器

图 16-52. LINCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED	LINC0_MATCH	LINC1CAP	LINC0CAP	RESERVED	CNTRXLOW	ZERONE	CTRENA
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 16-51. LINCTL 字段说明

位	字段	类型	复位	说明
31-7	保留	R/W	0h	
6	LINC0_MATCH	R/W	0h	计数器比较匹配模式。该位设置为 1 时，与 LINC0 寄存器的计数器比较匹配会在启用后触发 LINC0 中断。 0h = 禁用计数器比较匹配模式（启用捕获模式） 1h = 启用计数器比较匹配（禁用捕获模式）
5	LINC1CAP	R/W	0h	RXD 正边沿上的捕获计数器。启用后，该计数器值会在每个 RXD 正边沿捕获到 LINC1 寄存器中。启用后会触发 LINC1 中断。 0h = 禁用 UARTxRXD 正边沿上的捕获计数器 1h = 启用 UARTxRXD 正边沿上的捕获计数器
4	LINC0CAP	R/W	0h	RXD 负边沿上的捕获计数器。启用后，该计数器值会在每个 RXD 负边沿捕获到 LINC0 寄存器中。启用后会触发 LINC0 中断。 0h = 禁用 UARTxRXD 负边沿上的捕获计数器 1h = 启用 UARTxRXD 负边沿上的捕获计数器
3	RESERVED	R/W	0h	
2	CNTRXLOW	R/W	0h	当 RXD 上的信号为低电平时计数。当启用计数器并且 RXD 上的信号为低电平时，计数器递增。 0h = 禁止当 UARTxRXD 上的信号为低电平时计数 1h = 允许当 UARTxRXD 上的信号为低电平时计数
1	ZERONE	R/W	0h	在 RXD 负边沿上为零。启用后，在 RXD 负边沿上，该计数器设置为 0 并开始计数 0h = 禁止在 RXD 负边沿上为零 1h = 允许在 RXD 负边沿上为零
0	CTRENA	R/W	0h	LIN 计数器使能。LIN 计数器仅在启用后计数。 0h = 禁用计数器 1h = 启用计数器

16.3.39 LINC0（偏移 = 1138h）[复位 = 00000000h]

图 16-53 展示了 LINC0，表 16-52 中对此进行了介绍。

返回到汇总表。

UART LIN 模式捕获 0 寄存器

图 16-53. LINC0

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

表 16-52. LINC0 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	数据	R/W	0h	16 位捕获/比较寄存器 启用捕获 (LINC0CAP = 1) 后，在 RXD 下降沿捕获当前的 LINCTR 值并生成 LINC0 中断。 如果启用比较模式 (LINC0_MATCH = 1) 后，计数器匹配可生成 LINC0 中断。 0h = 最小值 FFFFh = 尽可能高的值

16.3.40 LINC1（偏移 = 113Ch）[复位 = 00000000h]

图 16-54 展示了 LINC1，表 16-53 中对此进行了介绍。

返回到汇总表。

UART LIN 模式捕获 1 寄存器

图 16-54. LINC1

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

表 16-53. LINC1 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	数据	R/W	0h	16 位捕获/比较寄存器 启用捕获 (LINC1CAP = 1) 后，在 RXD 上升沿捕获当前的 LINCTR 值并生成 LINC1 中断 0h = 最小值 FFFFh = 尽可能高的值

16.3.41 IRCTL（偏移 = 1140h）[复位 = X]

图 16-55 展示了 IRCTL，表 16-54 中对此进行了介绍。

返回到汇总表。

IrDA 控制寄存器

图 16-55. IRCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留						IRRXPL	RESERVED
R/W-0h						R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
IRTXPL						IRTXCLK	IREN
R/W-0h						R/W-0h	R/W-0h

表 16-54. IRCTL 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	
9	IRRXPL	R/W	0h	IrDA 接收输入 UCAxRXD 极性 0h = 高电平：当看到光脉冲时 IrDA 收发器发出高脉冲 1h = 低电平： 当一个轻脉冲出现时，IrDA 收发器传递了一个低脉冲
8	保留	R/W	0h	
7-2	IRTXPL	R/W	0h	发送脉冲长度。 脉冲长度 t(PULSE) = (IRTXPLx + 1) / [2 * f(IRTXCLK)] （IRTXCLK = UART 的功能时钟） 0h = 最小值 3Fh = 尽可能高的值
1	IRTXCLK	R/W	0h	IrDA 发送脉冲时钟选择 0h (R/W) = IrDA 编码数据基于功能时钟。 1h (R/W) = IrDA 编码数据基于波特率时钟。 当选择 8 倍过采样时，IRTXPL 周期应小于 8。 当选择 16 倍过采样时，IRTXPL 周期应小于 16。
0	IREN	R/W	0h	IrDA 编码器/解码器使能 0h (R/W) = 禁用 IrDA 编码器/解码器 1h (R/W) = 启用 IrDA 编码器/解码器

16.3.42 AMASK（偏移 = 1148h）[复位 = 000000FFh]

图 16-56 展示了 AMASK，表 16-55 中对此进行了介绍。

返回到汇总表。

自地址屏蔽寄存器。AMASK 寄存器用于启用 9 位或空闲线路模式的地址屏蔽。屏蔽地址位，从而创建一组将与接收到的地址字节相匹配的地址。
用于 DALI、UART 9 位或空闲线路模式。

图 16-56. AMASK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESERVED																								值							
R/W-0h																								R/W-FFh							

表 16-55. AMASK 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	值	R/W	FFh	9 位模式的自地址屏蔽。该字段包含的地址屏蔽可用于创建一组应匹配的地址。MSK 位字段中的 0 位将 UARTxADDR 寄存器 ADDR 位字段中的相应位配置为“无关”位。MSK 位字段中的 1 位将 UARTxADDR 寄存器 ADDR 位字段中的相应位配置为“必须匹配”位。 0h = 最小值 FFh = 尽可能高的值

16.3.43 ADDR（偏移 = 114Ch）[复位 = 00000000h]

图 16-57 展示了 ADDR，表 16-56 中对此进行了介绍。

返回到汇总表。

自地址寄存器。ADDR 寄存器用于写入在地址屏蔽 (AMASK) 设置为 FFh 时应与接收字节匹配的具体地址。该寄存器与 AMASK 配合使用，从而与接收到的地址字节匹配。
用于 DALI、UART 9 位或空闲线路模式。

图 16-57. ADDR

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

表 16-56. ADDR 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	值	R/W	0h	9 位模式的自地址。该字段包含当 UARTxAMASK 为 FFh 时应匹配的地址。 0h = 最小值 FFh = 尽可能高的值

16.3.44 CLKDIV2（偏移 = 1160h）[复位 = 00000000h]

图 16-58 展示了 CLKDIV2，表 16-57 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块专用分频比。
（仅适用于扩展的 UART）

图 16-58. CLKDIV2

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

表 16-57. CLKDIV2 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 3 分频 3h = 对时钟源进行 4 分频 4h = 对时钟源进行 5 分频 5h = 对时钟源进行 6 分频 6h = 对时钟源进行 7 分频 7h = 对时钟源进行 8 分频

This page intentionally left blank.

章节 17
SPI

串行外设接口 (SPI) 模块的一个标准化串行接口可在 MSPM0 器件与具有 SPI 接口的其他外部器件之间传输数据。

17.1 SPI 概述

SPI 模块的一个标准化串行接口可使用 SPI 协议在 MSPM0 器件与其他外部器件（例如传感器、存储器、ADC 或 DAC）之间传输数据。

17.1.1 外设的用途

SPI 模块充当控制器或外设接口，用于与外设和其他控制器进行同步串行通信。发送和接收路径由内部独立的 FIFO 存储器进行缓冲，允许多达 4 个 16 位宽度的入口。还有一个 DMA 接口用于与发送和接收 FIFO 进行数据交换。

17.1.2 特性

SPI 模块具有以下特性：

- 可配置为控制器或外设
- 可编程的时钟位速率以及预分频器；
- 独立的发送 (TX) 和接收 (RX) 先入先出 (FIFO) 缓冲区
- 可编程数据帧大小从 4 位到 16 位（控制器模式）
- 可编程数据帧大小从 7 位到 16 位（外设模式）
- 支持 PACKEN 功能，允许将 2 个 16 位 FIFO 条目打包为一个 32 位值以提高 CPU 性能。不过，并非所有器件都支持打包。请查看器件特定数据表。
- 发送和接收 FIFO 中断、超限和超时中断以及 DMA 完成
- 可编程 SPI 模式支持 Motorola SPI、MICROWIRE 或德州仪器 (TI) 格式
- 发送和接收路径均使用 CTL1.PTEN 和 CTL1.PREN 位支持 single-bit 奇偶校验
- 直接存储器存取控制器接口 (DMA)：
  - 相互独立的发送通道和接收通道
  - 发送完成中断

17.1.3 功能方框图

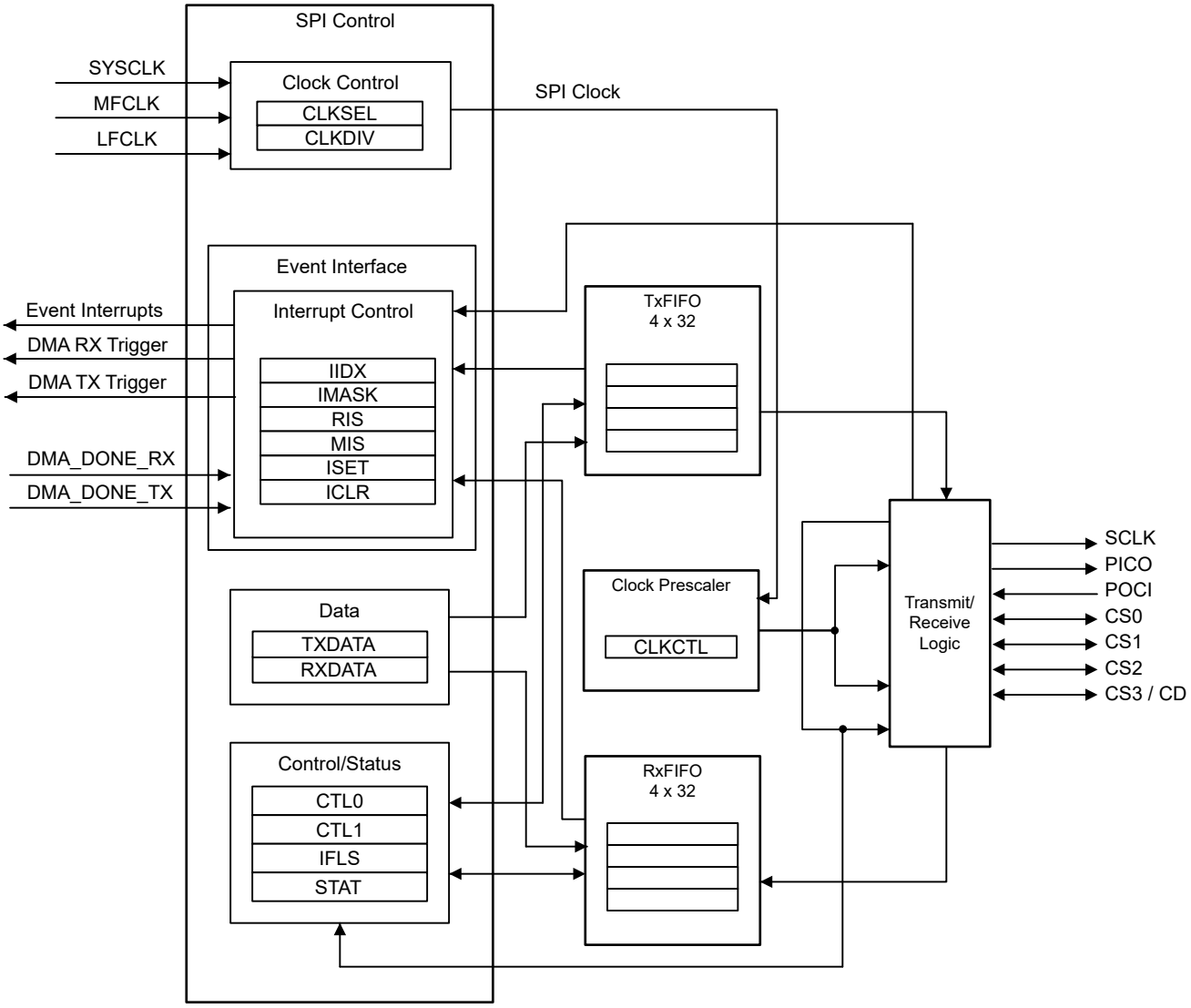


图 17-1. SPI 功能方框图

17.1.4 外部连接和信号说明

图 17-2 和表 17-1 概述了 SPI 模块不同工作模式下的引脚功能。

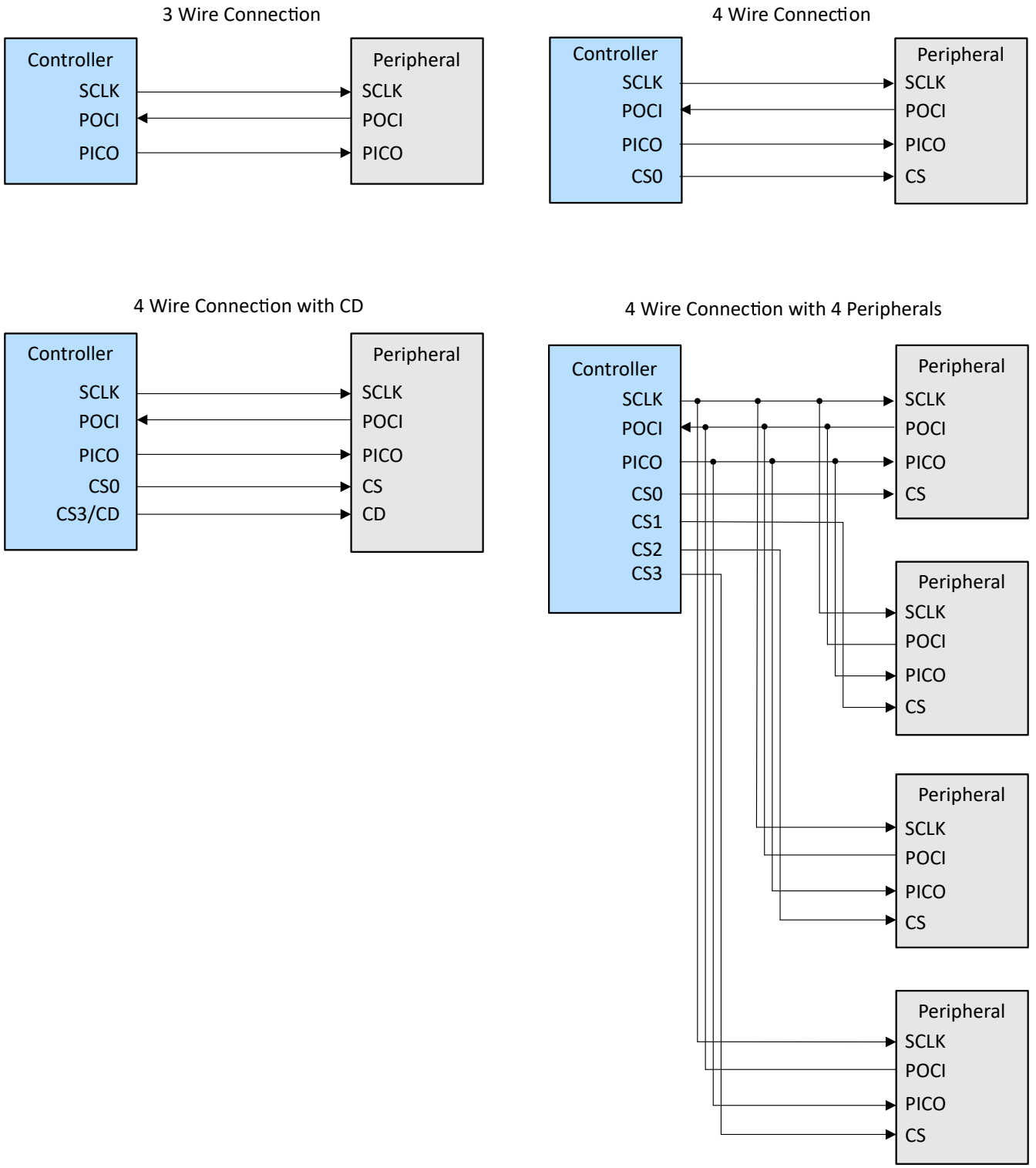


图 17-2. 针对不同 SPI 配置的外部连接

表 17-1. 引脚功能概述

标准 SPI	特性
SCLK	**SPI 时钟** 控制器模式：SCLK 是输出 外设模式：SCLK 是输入
PICO	**控制器输出，外设输入** 控制器模式：PICO 是数据输出线路 外设模式：PICO 是数据输入线路
POCI	**控制器输入，外设输出** 控制器模式：POCI 是数据输入线路 外设模式：POCI 是数据输出线路
CS0	片选信号 0，用于 4 引脚模式
CS1	片选信号 1，用于 4 引脚模式
CS2	片选信号 2，用于 4 引脚模式
CS3/CD	片选信号 3 或命令，用于 4 引脚模式

17.2 SPI 运行

17.2.1 时钟控制

选择 SPI 内部功能时钟并将其从源自该模块的时钟中分频。

- 使用 SPIx.CLKSEL 寄存器来选择 SPI 功能时钟的源。
  - BUSSCLK：当前总线时钟被选为 SPI 的源。当前总线时钟取决于电源域。如果 SPI 实例位于电源域 1 (PD1) 中，请参阅 MCLK；如果 SPI 实例位于电源域 0 (PD0) 中，请参阅 ULPCLK。
  - MFCLK：选择 MFCLK 作为 SPI 的时钟源，请参阅 MFCLK。
  - LFCLK：选择 LFCLK 作为 SPI 的源，请参阅 LFCLK
- 使用 SPIx.CLKDIV 寄存器来选择 SPI 功能时钟的分频比。选项包括 1 分频至 8 分频。

在使用 SPIx.PWREN 寄存器中的 ENABLE 位配置使用之前，必须启用 SPI 模块（请参阅外设电源使能）。当设置 SPI 或者应更改配置时，ENABLE 位应清零以避免更新期间或者之后的首次数据接收或发送时出现不可预知的行为。

控制器和外设模式支持的最大 SPI 频率取决于器件时钟选项和 IO 选项。更多相关信息，请参阅特定器件数据表规格。

17.2.2 通用架构

17.2.2.1 芯片选择和命令处理

17.2.2.1.1 片选控制

可以通过将 CTL1.MS 设置为 1 来将 SPI 配置为控制器模式，通过清除 CTL1.MS 位来将 SPI 配置为外设模式。

CTL0.CSSEL 位选择通过最多 4 个 CS 信号寻址哪个连接的外设。这些位由控制器或目标/外设模式中的 SPI 模块控制。在传输过程中控制所选信号。

芯片选择信号需要由控制器在四线制模式下提供，并且可以通过配置 PINCM.CSx.INV 寄存器来反转芯片选择极性。

在外设模式下，时钟由控制器提供并由外设用于捕获数据。外设可选择在 3 线或 4 线模式下运行。4 线制模式仅在 CS 激活时接受数据传输。

设置 CTL0.CSCLR 位后，发送/接收移位寄存器计数器会在 CS 进入非运行状态时自动清零。使用 Motorola 4 线或 National Microwire 模式时，请遵循以下约束条件：

- 在 CS 引脚重新置为有效之前，CS 禁用周期必须长于 2 个功能时钟周期
- CS 超前时间（CS 运行至第一个位时钟沿）必须至少为 2 个 SPI 功能时钟周期

在时钟线路上出现干扰时或在初始化期间，遵循这些限制有助于外设在控制器上再次同步。该位仅在外设模式下相关。

- CTL0.CSCLR = 0：当 CS 信号禁用外设时，发送/接收位计数器状态将保持不变。
- CTL0.CSCLR = 1：当 CS 信号禁用外设时，发送/接收位计数器将清零。

备注

CSCLR 功能要求 CS 禁用脉冲长于 2 个 SPI 功能时钟周期，以正确检测和清除 SPI 中的位计数器。CS 超前时间（CS 运行至第一个位时钟沿）也需要至少 2 个 SPI 功能时钟周期。

17.2.2.1.2 命令数据控制

当使用 Motorola 帧格式时，CDMODE 位可以设置为使用 CS3/CD 线路作为信号来区分命令和数据信息。这通常用于 LCD 或数据存储设备。

- CD 电平低：命令函数
- CD 电平高：数据函数

CTL1.CDMODE 可以写入 1-14 的值以指定字节数，对于 SPI 发送的给定字节数，CD 线将变为低电平，从下一个要传输的值开始。传输字节数后，CD 将自动变为高电平。如果 0xF 的值被置位，C/D 将一直保持低电平；如被设置为 0，则在发送当前字符后立即将 CD 线路设置为高电平。

此选项仅适用于控制器模式。CTL1.CDENABLE 只能在禁用 SPI 模块时更新，CTL1.CDMODE 可在不同数据包之间更新。将在 CDENABLE 或 SPI ENABLE 设置为禁用时复位计数器。在 CTL1.CDMODE 中设置新值之前，应检查 FIFO 的状态为空，SPI 应处于 Idle 模式。

当写入一个新值到 CTL1.CDMODE 时，内部计数器将被复位，新值将被用于计数。如果计数器确实倒数到 0 并且应该发送另一个命令包，则需要首先再次设置 CDMODE，否则下一个数据以 CD 引脚信令数据模式作为数据发送。

17.2.2.2 数据格式

控制位 CTL1.MSB 定义了数据输入和输出的方向，最高有效位 (MSB) 或最低有效位 (LSB) 在前。如果启用奇偶校验，则奇偶校验位始终作为最后一个位接收。

使用控制寄存器位 CTL0.DSS 时，每次传输的位长度将定义为 4-16 位（对于控制器模式）和 7-16 位（对于外设模式）。

写入 TX 缓冲区寄存器将触发传输。数据写入至少需要提供传输的位数。例如，如果仅将一个字节写入 TX 缓冲区但传输的长度大于 8，则丢失的位将用 0 填充。在接收路径上，接收到 CTL0.DSS 寄存器中定义的位数后，接收到的数据将移至 RXFIFO 或 RX 缓冲区。

访问 RX 和 TX 缓冲区时，位数应至少能够覆盖一次传输。

- 4-8 位：字节访问（外设模式：7-8 位）
- 9-16 位：16 位访问

时钟极性 (CTL0.SPO) 用于控制不传输数据时的时钟极性，仅在 Motorola SPI 帧模式下使用。

- 0h = 不传输数据时，外设在 CLKOUT 引脚上生成稳态低电平值。
- 1h = 不传输数据时，外设在 CLKOUT 引脚上生成稳态高电平值。

时钟相位 (CTL0.SPH) 位选择捕获数据的时钟沿，并能使时钟沿更改状态；由于可以允许或不允许在第一个数据捕获沿之前进行时钟转换，所以对第一个传输位的影响最大。请参阅 Motorola SPI 帧模式部分以查看示意图。

- 0h = 在第一次时钟沿转换时捕获数据。
- 1h = 在第二次时钟沿转换时捕获数据。

可以将 SPI 配置为在外设模式下工作，CTL1.MS 位 = 0。在外设模式下，时钟由控制器提供，并可用于 CLK 引脚上的外设（需要针对输入进行配置）。不使用时钟选择和分频器控制位。CS 输入信号用于在 4 线制模式下选择/启用外设的数据接收路径。

SPI 可配置为用作控制器，CTL1.MS 位 = 1。在控制器模式下，需要通过使用时钟选择位选择可用的时钟源来生成时钟。还需要根据所选协议控制 CS 信号。

设置 CTL1.PEN 位时，最后一位将用作奇偶校验，以评估先前位的完整性。CTL1.PES 位选择奇偶校验模式为偶数或奇数。检测到故障时，设置中断标志 RIS.PER 以将数据标记为无效。奇偶校验特性旨在提高通信稳健性。

17.2.2.3 延迟的数据采样

如果由于运行时条件和以下输入数据采样级，输入数据到达 POCI 引脚时有一些延迟，则会在采样时钟沿对之前的数据进行采样。为了补偿这种情况，可通过 CLKCTL.DSAMPLE 位设置延迟采样。延迟采样仅在控制器模式下可用。通过设置控制寄存器位 CLKCTL.DSAMPLE，可以按 SPI 输入时钟步长调整延迟。允许的最大延迟不应超过一个数据帧的长度。

17.2.2.4 时钟生成

SPI 包含一个可编程比特率时钟分频器和预分频器来生成串行输出时钟 (SCLK)。

支持的比特率最高为输入时钟除以 2。输入时钟选择取决于特定器件，请参阅器件数据表和时钟控制部分。

SPI 功能可以与以下选定的输入中的任何一个一起使用：SYSCLK、MFCLK 和 LFCLK

“SPI 时钟”是根据 CLKDIV 寄存器选择的比特率执行时钟分频后的输出。SPI 时钟 = 选择的输入时钟/(1 + CLKDIV)

SPI 采样时钟 (SCLK) 是将 SPI 时钟除以预分频器值后的输出。SCLK = SPI 时钟/((1 + SCR)*2)

如果 CLKDIV 设置系数二 (*2)，则输入时钟必须至少比 SPI 时钟快 2 倍。

17.2.2.5 FIFO 操作

发送 FIFO

TX FIFO 是一个 32 位宽、4 位置深的先入先出存储器缓冲区。CPU 通过写入 SPI 数据寄存器 TXDATA.DATA 将数据写入 FIFO，数据存储在 FIFO 中，直到被传输逻辑读出。

当配置为控制器或外设时，并行数据在串行转换之前写入 TX FIFO，并分别通过 PICO 或 POCI 引脚传输到连接的外设或控制器。

在外设模式下，每次控制器启动事务时，SPI 都会发送数据。如果 TX FIFO 为空且控制器启动传输，则外设发送写入发送 FIFO 的最新值。用户或软件负责根据需要向 FIFO 提供有效数据。SPI 可配置为在 FIFO 为空时生成中断或 DMA 请求。发送 FIFO 有一个 TXFIFO_UNF 中断用于指示 FIFO 下溢情况。

接收 FIFO

RX FIFO 是一个 32 位宽、4 位置深的先入先出存储器缓冲区。从串行接口接收到的数据在由 CPU 或 DMA 读出之前一直存储在缓冲区中，CPU 或 DMA 通过读取 SPIx.RXDATA 寄存器来访问读取 FIFO。

当配置为控制器或外设时，通过 POCI 或 PICO 引脚接收串行数据。由于每次访问都会更新 FIFO 的访问指针，因此需要通过单次传输来访问数据。

借助位于 STAT 寄存器（TFE、TNF、RFE、RNF）中的 FIFO 填充电平触发信号，FIFO 缓冲区允许应用在一个缓冲区中连续流式传输串行数据，同时应用移动或处理来自另一个缓冲区的数据。如果 FIFO 已满且新数据在未读数据的情况下写入 FIFO，则设置 RXFIFO 溢出事件。接收 FIFO 将生成一个 RXFULL 中断来指示 FIFO 已满情况。

17.2.2.6 环回模式

通过设置 CTL1 寄存器中的 LBM 位，可以将 SPI 模块置于内部环回模式，以开展诊断或调试工作。在环回模式下，来自 TX FIFO 的数据可以串行传输到 RX FIFO 中。可以读取 RX FIFO 中的数据以检查是否进行了正确的传输。当模块被设置为内部环回模式时，IO 的外部切换无效。

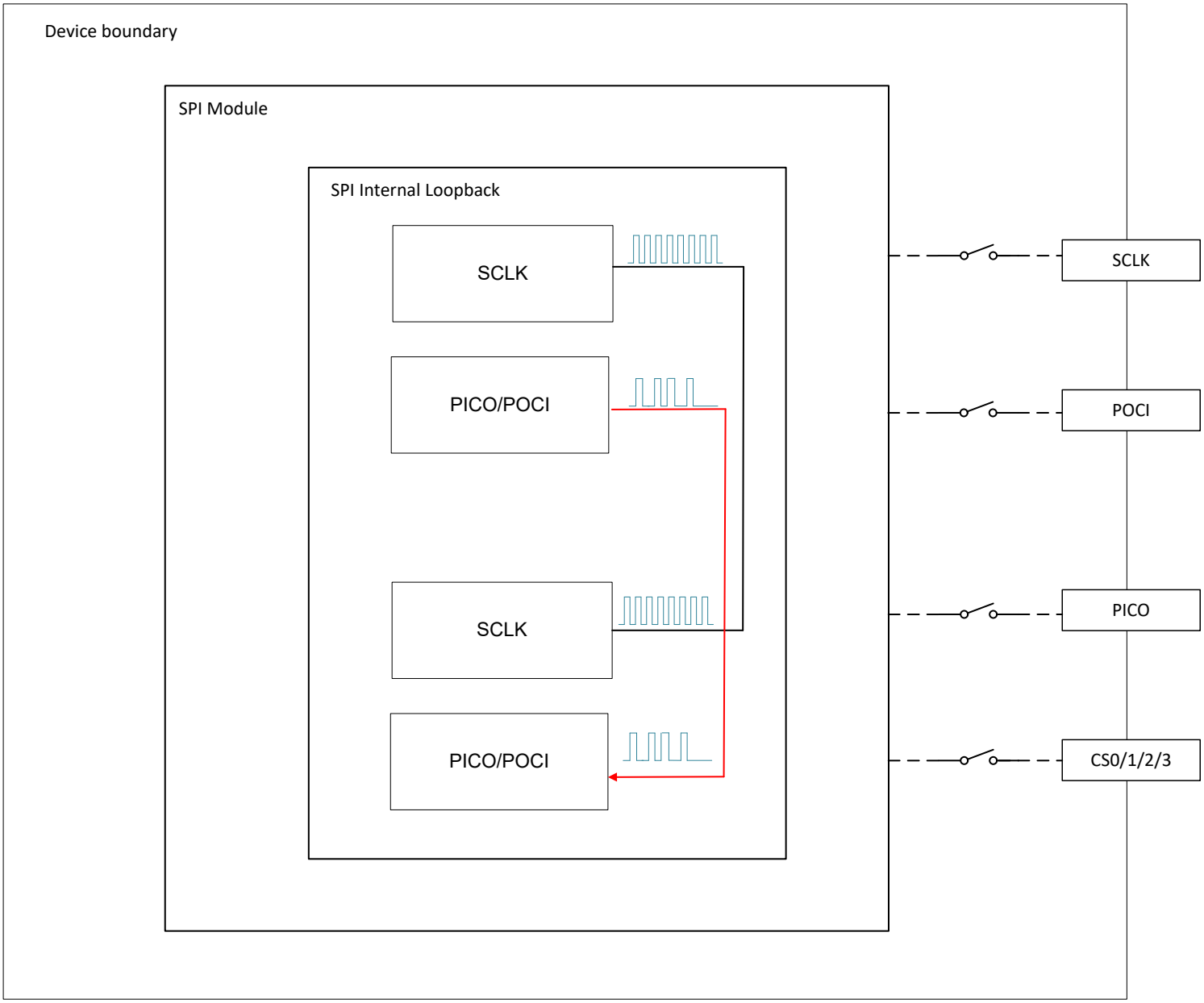


图 17-3. 内部环回模式

17.2.2.7 DMA 操作

SPI 为 DMA 控制器提供了一个接口，该接口具有独立的发送通道和接收通道。SPI 的 DMA 操作通过 SPI 事件和 DMA 寄存器启用。启用 DMA 操作后，当相关的 FIFO 可以传输数据时，SPI 在接收通道或发送通道上发出 DMA 请求。

对于接收通道，只要接收 FIFO 中的数据量达到或超过使用 IFL 寄存器中的 RXIFLSEL 位配置的 FIFO 触发电平，或者已经触发接收超时，就会发出传输请求。在这种情况下，将发送到目前为止接收到的数据量。

对于发送通道，只要发送 FIFO 中包含的字符少于使用 IFL 寄存器中的 TXIFLSEL 位配置的 FIFO 触发电平，就会发出传输请求。DMA 传输请求由 DMA 控制器自动处理，具体取决于 DMA 通道的配置方式。

DMA 传输可以在 SPI 传输的数据宽度和 8/16 位的总线访问宽度之间进行配置和对齐，以有效利用总线。对于接收和发送，触发和传输是独立的。

有关中断和事件的更多信息，请参阅节 17.2.6.2 部分。

17.2.2.8 重复传输模式

使用 CTL1.REPEATTX 位后，最后一个字符的发送将按照寄存器位的定义重复。CTL1.REPEATTX 位的值为 0 将禁用此模式，这一功能仅在控制器模式下可用。传输将通过将数据写入 TX 缓冲器开始，然后会使用给定的值重复发送数据。该行为与将数据写入 TX 缓冲器的行为相同，并与此处的值定义的次数相同。这可用于清理传输，或从外设中提取特定数量的数据。

当使用 REPEATTX 时，它需要与 FIFO 中的数据对齐。所以应该使用下面显示的顺序：

- 等待并检查直到 FIFO 为空
- 设置 REPEATTX
- 写入到 TXDATA/FIFO
- 等待收到请求的数据

17.2.2.9 低功率模式

SPI 模块位于电源域 1 (PD1) 中，因此仅在运行和睡眠模式下处于活动状态。如果 SPI 模块由应用软件启用，则进入停止或待机低功耗模式会强制 SPI 模块在器件处于停止或待机模式时被暂时禁用。

17.2.3 协议说明

可以使用 CTL0.FRF 寄存器选择协议格式模式。支持的选项包括 Motorola 3 wire、Motorola 4 wire、Texas Instruments Synchronous 和 MICROWIRE。

17.2.3.1 Motorola SPI 帧格式

Motorola SPI 接口是一个 4 线接口，其中 CS 信号充当外设选择。在 3 线模式下，不需要 CS 信号，并且模块的行为就像始终选中一样。Motorola SPI 格式的主要特点是：可以通过 SPIx.CTL0 控制寄存器中的 SPO 和 SPH 位，对 SCLK 信号的非活动状态和相位进行编程。

SPO 时钟极性位

如果 CTL0.SPO 时钟极性控制位清零，则当不传输数据时，该位会在 SCLK 引脚上产生一个稳态低电平值。如果设置了 CTL0.SPO 位，则当不传输数据时，该位会在 SCLK 引脚上放置一个稳态高电平值。

SPH 相位控制位

CTL0.SPH 相位控制位用来选择捕捉数据的时钟边沿，并允许边沿改变状态。该位的状态对发送的第一个位影响最大，即允许或不允许在第一个数据捕捉边沿之前进行时钟转换。如果 CTL0.SPH 相位控制位清零，则在第一个时钟边沿转换时捕捉数据。如果 SPH 位为高，则在第二个时钟边沿转换时捕获数据。

SPO = 0 和 SPH = 0 时的 Motorola SPI 帧格式

图 17-4 显示了 SPO = 0 和 SPH = 0 时 Motorola SPI 格式的信号序列。

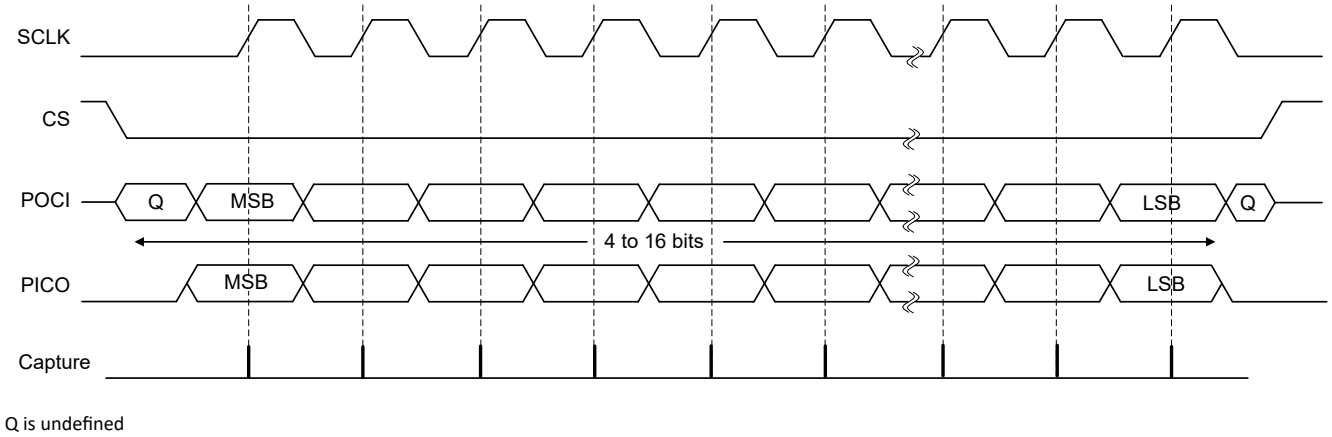


图 17-4. SPO = 0 和 SPH = 0 时的 Motorola SPI 格式

在此配置中，空闲期间会发生以下情况：

- SCLK 被强制为低电平
- CS 被强制为高电平
- 发送数据线路 PICO 被强制为高电平
- 当 SPI 配置为控制器时，它启用 SCLK 引脚
- 当 SPI 配置为外设时，它禁用 SCLK 引脚

如果启用了 SPI 并且 TX FIFO 中存在有效数据，则 CS 控制器信号在发送开始时被驱动为低电平，这会导致使外设数据能够进入到控制器的 POCI 输入线路上。此时将启用控制器 PICO 输出引脚。

半个 SCLK 周期后，有效的控制器数据会传输到 PICO 引脚。一旦同时设置了控制器和外设数据，SCLK 控制器时钟引脚就会在额外的半个 SCLK 周期后变为高电平。现在，将在 SCLK 信号的上升沿捕捉数据，并在下降沿传播数据。

对于单字传输，在传输数据字的所有位之后，CS 线路会在最后一个位被捕捉后的一个 SCLK 周期内返回其 IDLE 高电平状态。对于连续的背靠背传输，CS 信号必须在每个数据字传输之间产生高脉冲，因为当 SPH 位清零时，外设选择引脚会冻结其串行外设寄存器中的数据，并且不允许更改数据。控制器器件必须在每次数据传输之间拉高外围器件的 CS 引脚，以启用串行外设数据写入。当连续传输完成时，CS 引脚将在最后一个位被捕捉后的一个 SCLK 周期内返回到其 IDLE 状态。

SPO = 0 和 SPH = 1 时的 Motorola SPI 帧格式

图 17-5 显示了 SPO = 0 和 SPH = 1 时 Motorola SPI 格式的信号序列。

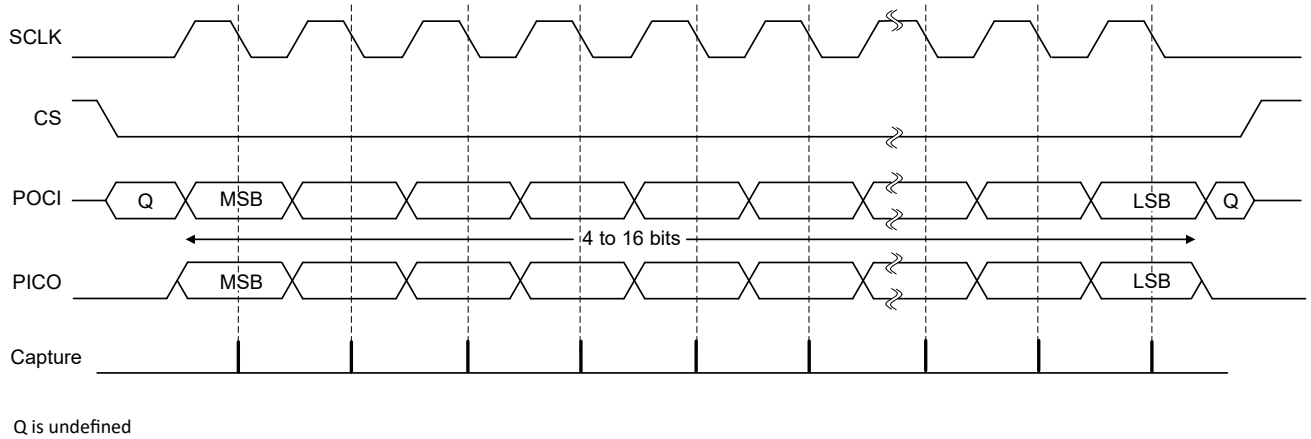


图 17-5. SPO = 0 和 SPH = 1 时的 Motorola SPI 帧格式

如果启用了 SPI 并且 TX FIFO 中存在有效数据，则 CS 控制器信号在开始发送时变为低电平。此时将启用控制器 PICO 输出。在额外的半个 SCLK 周期之后，控制器和外设有效数据都将能够进入到各自的发送线路上。同时，利用上升沿转换启用 SCLK。然后，在 SCLK 信号的下降沿捕捉数据，并在上升沿传播数据。

对于单字传输，在传输所有位之后，CS 线路会在最后一个位被捕捉后的一个 SCLK 周期内返回其 IDLE 高电平状态。对于连续的背靠背传输，CS 引脚在连续数据字之间保持低电平，并像单字传输一样终止。

SPO = 1 和 SPH = 0 时的 Motorola SPI 帧格式

图 17-6 显示了 SPO = 1 和 SPH = 0 时 Motorola SPI 格式的信号序列。

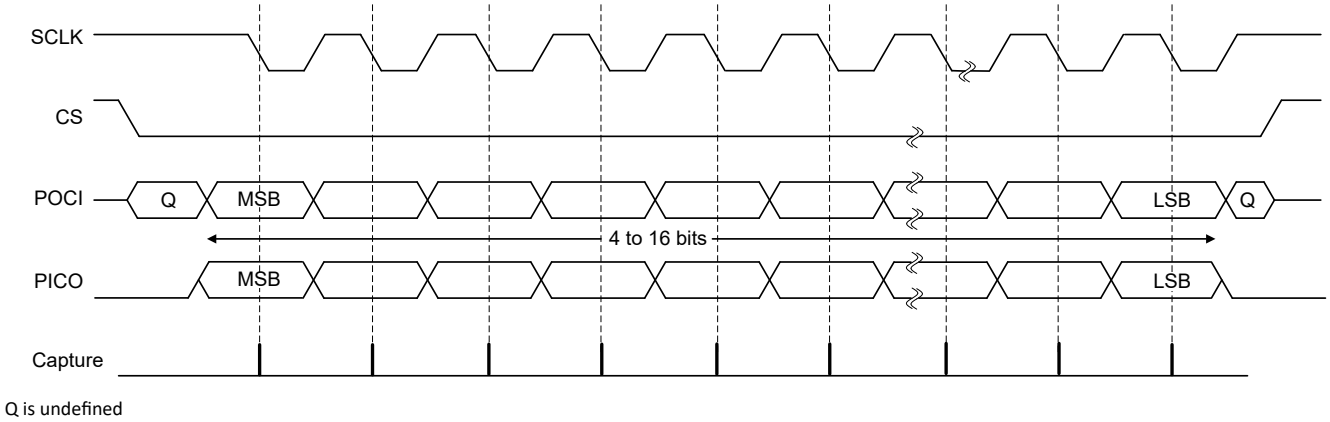


图 17-6. SPO = 1 和 SPH = 0 时的 Motorola SPI 帧格式

在此配置中，空闲期间会发生以下情况：

- SCLK 被强制为高电平
- CS 被强制为高电平
- 发送数据线路 PICO 被任意强制为高电平
- 当 SPI 配置为控制器时，它启用 SCLK 引脚
- 当 SPI 配置为外设时，它禁用 SCLK 引脚

如果启用了 SPI 并且 TX FIFO 中存在有效数据，则 SPI CS 控制器信号在开始发送时变为低电平，并立即将外设数据传输到控制器的 POCI 线路上。此时将启用控制器 PICO 输出引脚。

半个 SCLK 周期后，有效的控制器数据会传输到 PICO 线路上。当控制器和外设数据都已设置时，SCLK 控制器时钟引脚在一个额外的半个 SCLK 周期后变为低电平。然后，在 SCLK 信号的下降沿捕捉数据，并在上升沿传播数据。

对于单字传输，在传输数据字的所有位之后，CS 线路会在最后一个位被捕捉后的一个 SCLK 周期内返回其 IDLE 高电平状态。对于连续的背靠背传输，CS 信号必须在每个数据字传输之间产生高脉冲，因为当 SPH 位清零时，外设选择引脚会冻结其串行外设寄存器中的数据，并防止更改数据。控制器器件必须在每次数据传输之间拉高外围器件的 CS 引脚，以启用串行外设数据写入。当连续传输完成时，CS 引脚将在最后一个位被捕捉后的一个 SCLK 周期内返回到其 IDLE 状态。

SPO = 1 和 SPH = 1 时的 Motorola SPI 帧格式

图 17-7 显示了 SPO = 1 和 SPH = 1 时 Motorola SPI 格式的信号序列。

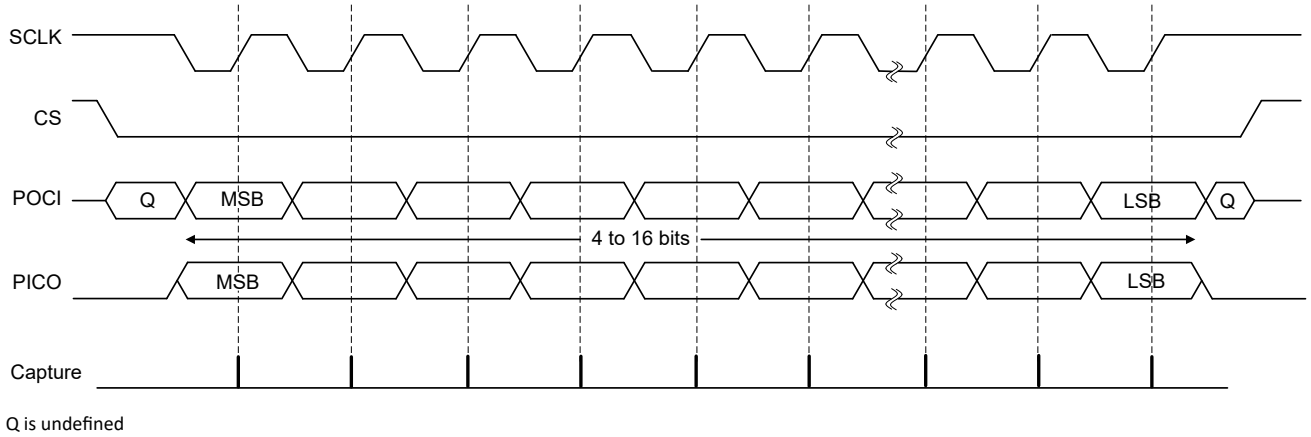


图 17-7. SPO = 1 和 SPH = 1 时的 Motorola SPI 帧格式

在此配置中，空闲期间会发生以下情况：

- SCLK 被强制为高电平
- CS 被强制为高电平
- 发送数据线路 PICO 被任意强制为高电平
- 当 SPI 配置为控制器时，它启用 SCLK 引脚
- 当 SPI 配置为外设时，它禁用 SCLK 引脚

如果启用了 SPI 并且 TX FIFO 中存在有效数据，则通过 CS 控制器信号变为低电平来指示开始发送。此时将启用控制器 PICO 输出引脚。在额外的半个 SCLK 周期之后，控制器和外设数据都将能够进入到其各自的发送线路上。同时，利用下降沿转换启用 SCLK。然后，在 SCLK 信号的上升沿捕捉数据，并在下降沿传播数据。

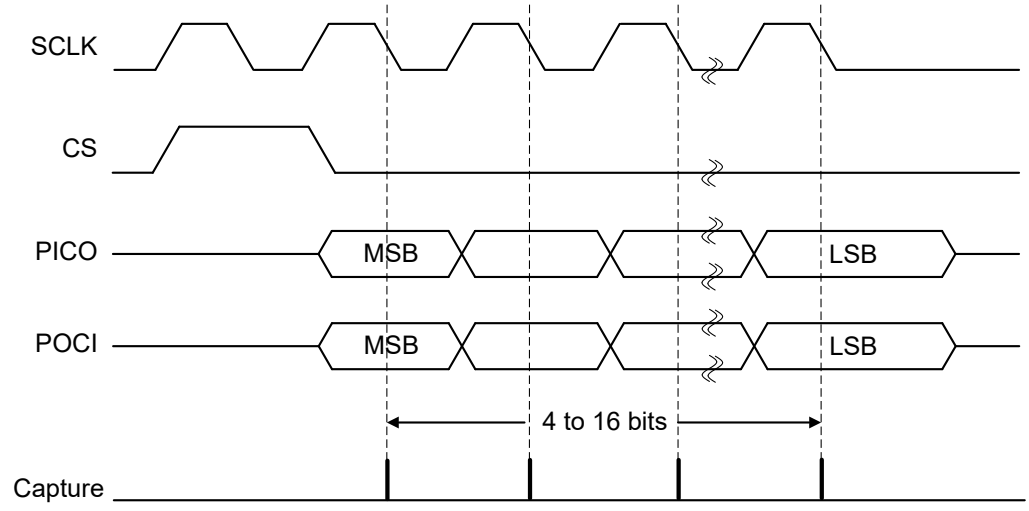
对于单字传输，在传输所有位之后，CS 线路会在最后一个位被捕捉后的一个 SCLK 周期内返回其 IDLE 高电平状态。对于连续的背靠背传输，CS 引脚保持在其低电平有效状态，直到最后一个字的最后一个位被捕捉，然后返回其 IDLE 状态。对于连续的背靠背传输，CS 引脚在连续数据字之间保持低电平，并像单字传输一样终止。

串行时钟 (SCLK) 在 SPI 空闲时保持非活动状态，并且 SCLK 仅在正在发送或接收数据期间以编程频率进行转换。如果 RX FIFO 在超时期间之后仍包含数据，则会发生接收超时，而 SCLK 的 IDLE 状态会提供相应的超时指示。

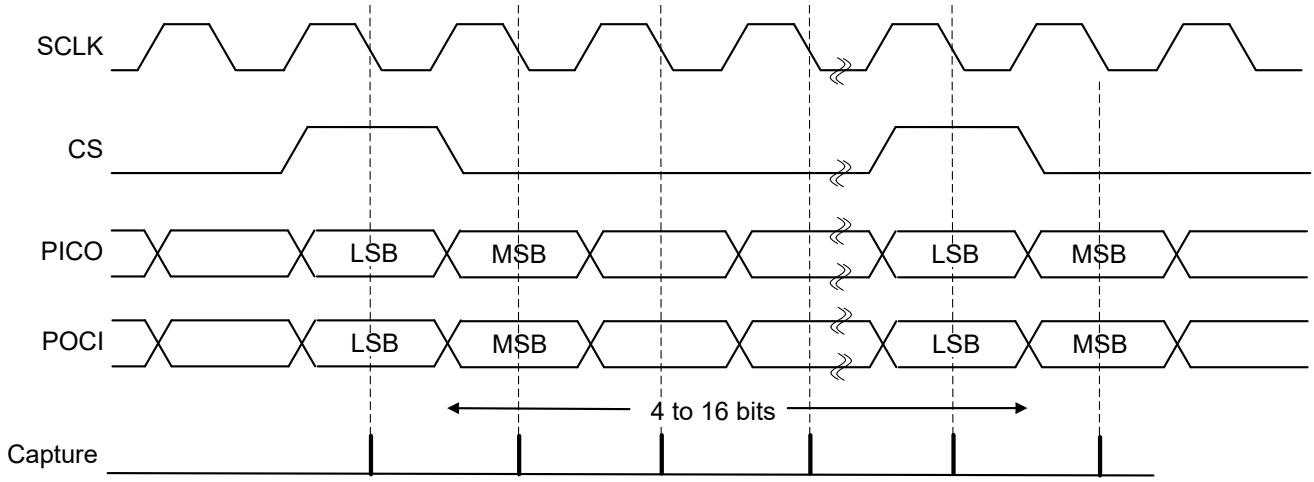
17.2.3.2 TI 同步串行接口帧格式

SPI 外设与德州仪器 (TI) 同步串行帧格式兼容。

图 17-8 所示为单个帧和连续发送的帧的 TI 同步串行帧格式。



Single Transmission Signal Sequences



Continuous Transmission Signal Sequences

图 17-8. TI 同步串行帧格式

只要 SPI 为空闲状态，便会强制 SCLK 和 CS 为低电平，并会将发送数据线路 PICO 置于三态。当 TX FIFO 的底部条目包含数据时，CS 会变为高电平脉冲并持续一个 SCLK 周期。发送的值也从 TX FIFO 传输到发送逻辑的串行移位寄存器。在 SCLK 的下一个上升沿，4 至 16 位数据帧的 MSB 在 PICO 引脚上移出。同样，接收到的数据的 MSB 也通过片外串行外设移到 POCI 引脚上。然后，SPI 和片外串行外设均在 SCLK 的每个下降沿将数据位逐个记录到相应的串行移位器中。在最低有效位 (LSB) 被锁存之后 SCLK 的第一个上升沿，接收到的数据从串行移位器传输到 RX FIFO。

串行时钟 (SCLK) 在 SPI 空闲时保持非活动状态，并且 SCLK 仅在正在发送或接收数据期间以编程频率进行转换。如果 RX FIFO 在超时期间之后仍包含数据，则会发生接收超时，而 SCLK 的 IDLE 状态会提供相应的超时指示。

17.2.4 复位注意事项

软件复位注意事项

可以通过同时设置 RSTCTL 寄存器中的 RESETASSERT 和 KEY 位来执行软件复位。正在进行的传输将立即终止，并可能使软件处于未定义状态。因此，在请求复位之前，应终止正在进行的传输。

硬件复位注意事项

硬件复位也会初始化 IO 配置。此过程会将 IO 设置为高阻抗状态，并且数据线可能会悬空。如果对于应用或 SPI 接口上连接的器件至关重要，可能需要外部上拉或下拉电阻。

17.2.5 初始化

要启用和初始化 SPI，需要执行以下步骤：

1. 使用 SPI 信号多路复用到的相应 GPIO 引脚配置 IOMUX

备注

可使用上拉电阻来避免 SPI 引脚上出现不必要的切换，该切换会使外设进入错误状态。此外，如果通过 CTL0 寄存器的 SPO 位将 SCLK 信号编程为稳态高电平，那么软件还必须将与 SCLK 信号相对应的 GPIO 端口引脚配置为上拉。

对于每种帧格式，应按照如下步骤配置 SPI：

1. 在进行任何配置更改之前，确保 CTL1 寄存器中的 ENABLE 位已清零。
2. 通过写入 CLKSEL 和 CLKDIV 寄存器来选择和配置时钟预分频因子。
3. 选择 SPI 是控制器还是外设：
   - 对于控制器操作，设置 CTL1 寄存器中的 MS 位。
   - 对于外设模式，将 CTL1 寄存器中的 MS 位清零。
4. 通过写入 CLKCTL 寄存器来配置时钟分频器。
5. 请注意，在切换 SPI 协议格式时，需要进行 SPI 软件复位（请参阅第 15.3.4 节）。
6. 根据所需的协议、数据宽度和其他特殊配置，配置 CTL0 和 CTL1 寄存器。
7. 可选择配置 DMA
8. 通过设置 CTL1 寄存器中的 ENABLE 位来启用 SPI。

17.2.6 中断和事件支持

SPI 模块包含三个事件发布者而没有事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理到 CPU 子系统的 SPI 中断请求 (IRQ)。第二个和第三个事件发布者（DMA_TRIG_RX、DMA_TRIG_TX）用于通过 DMA 事件路由设置 DMA 的触发信号。

表 17-2 中总结了 SPI 事件。

表 17-2. SPI 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	SPI	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 SPI 到 CPU 的中断路由
DMA 触发	发布者	SPI	DMA	DMA 事件路由	DMA_TRIG_RX 寄存器	从 SPI RX 到 DMA 的固定中断路由
DMA 触发	发布者	SPI	DMA	DMA 事件路由	DMA_TRIG_TX 寄存器	从 SPI TX 到 DMA 的固定中断路由

17.2.6.1 CPU 中断事件发布者 (CPU_INT)

SPI 模块提供 9 个中断源，这些中断源可以生成 CPU 中断事件。表 17-3 按照优先级从高到低的顺序列出了来自 SPI 的 CPU 中断事件。

表 17-3. SPI CPU_INT 触发条件

IIDX STAT	名称	说明
0x01	RXFIFO_OVF	RXFIFO 溢出事件。如果检测到 RX FIFO 溢出，则设置此中断。

表 17-3. SPI CPU_INT 触发条件 (continued)

IIDX STAT	名称	说明
0x02	PER	奇偶校验错误事件。如果检测到奇偶校验错误，则会设置该位。
0x03	RTOUT	外设接收超时事件。当处于外设模式且未在 CTL1.RXTIMEOUT 选定数量的功能时钟周期内接收数据时。
0x04	RX	接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则设置此中断。
0x05	TX	发送 FIFO 事件。如果已达到选定的发送 FIFO 级别，则设置此中断。
0x06	TXEMPTY	发送 FIFO 空中断。如果发送 FIFO 中的所有数据都已移出，则会设置该位。
0x07	IDLE（闲置）	SPI 空闲。SPI 已完成传输并更改为空闲模式。当 STAT.BUSY 位变为低电平时，设置该位。
0x08	DMA_DONE1_RX	如果 RX DMA 通道发送 DONE 信号，则会设置此中断。
0x09	DMA_DONE1_TX	如果 TX DMA 通道发送 DONE 信号，则会设置此中断。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关配置事件寄存器的指导，请参阅节 7.2.5。

17.2.6.2 DMA 触发发布者（DMA_TRIG_RX、DMA_TRIG_TX）

DMA_TRIG_RX 和 DMA_TRIG_TX 寄存器用于设置 DMA 的触发信号。这可以通过灵活的方式进行设置，使用表 17-4 和表 17-5 中的触发条件，来触发 DMA 执行接收或发送事件。

DMA_TRIG_RX 用于触发 DMA 以执行接收数据传输，DMA_TRIG_TX 用于触发 DMA 以执行发送数据传输。

表 17-4. SPI DMA_TRIG_RX DMA 触发条件

IIDX STAT	名称	说明
0x03	RTOUT	外设接收超时事件。当处于外设模式且未在 CTL1.RXTIMEOUT 选定数量的功能时钟周期内接收数据时。
0x04	RX	接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则设置此中断。

表 17-5. SPI DMA_TRIG_TX DMA 触发条件

IIDX STAT	名称	说明
0x05	TX	发送 FIFO 事件。如果已达到选定的发送 FIFO 级别，则设置此中断。

通过 DMA_TRIG_RX 和 DMA_TRIG_TX 事件管理寄存器来管理 DMA 触发事件配置。有关配置事件寄存器的指导，请参阅节 7.2.5；有关 DMA 触发事件的工作原理，请参阅节 7.1.3.2。

17.2.7 仿真模式

器件处于调试模式时的模块行为由 PDBGCTL 寄存器中的 FREE 和 SOFT 位控制。

当器件处于调试模式并设置为停机模式时，可配置以下行为。

表 17-6. 调试模式外设行为

PDBGCTL.FREE	PDBGCTL.SOFT	功能
1	**x**	模块继续运行
0	**0**	模块立即停止
0	**1**	模块在下一次传输完成后停止

17.3 SPI 寄存器

表 17-7 列出了 SPI 寄存器的存储器映射寄存器。表 17-7 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 17-7. SPI 寄存器

偏移	缩写	寄存器名称	组	部分
4h	SCLK	SCLK		转到
8h	MOSI	MOSI		转到
Ch	MISO	MISO		转到
18h	CS0	SPI 芯片选择 0		转到
1Ch	CS1_MISO1	SPI 芯片选择 1		转到
20h	CS2_MISO2	SPI 芯片选择 2		转到
24h	CS3_CD_MISO3	SPI 芯片选择 3		转到
204h	SCLK	SCLK 的 FUPDATE 版本		转到
208h	MOSI	MOSI 的 FUPDATE 版本		转到
20Ch	MISO	MISO 的 FUPDATE 版本		转到
218h	CS0	CS0 的 FUPDATE 版本		转到
21Ch	CS1_MISO1	CS1 的 FUPDATE 版本		转到
220h	CS2_MISO2	CS2 的 FUPDATE 版本		转到
224h	CS3_CD_MISO3	CS3 的 FUPDATE 版本		转到
480h	CPU_CONNECT_0	CPU 连接		转到
504h	DMA_MAP_RX	DMA 映射		转到
505h	DMA_TRIG_RX	DMA 触发		转到
506h	DMA_ENTRY_RX	DMA 条目		转到
508h	DMA_MAP_TX	DMA 映射		转到
509h	DMA_TRIG_TX	DMA 触发		转到
50Ah	DMA_ENTRY_TX	DMA 条目		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
808h	CLKCFG	外设时钟配置寄存器		转到
814h	STAT	状态寄存器		转到
1000h	CLKDIV	时钟分频器		转到
1004h	CLKSEL	超低功耗外设的时钟选择		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引寄存器	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引寄存器	DMA_TRIG_RX	转到
1058h	IMASK	中断屏蔽	DMA_TRIG_RX	转到
1060h	RIS	原始中断状态	DMA_TRIG_RX	转到

表 17-7. SPI 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
1068h	MIS	已屏蔽中断状态	DMA_TRIG_RX	转到
1070h	ISET	中断设置	DMA_TRIG_RX	转到
1078h	ICLR	中断清除	DMA_TRIG_RX	转到
1080h	IIDX	中断索引寄存器	DMA_TRIG_TX	转到
1088h	IMASK	中断屏蔽	DMA_TRIG_TX	转到
1090h	RIS	原始中断状态	DMA_TRIG_TX	转到
1098h	MIS	已屏蔽中断状态	DMA_TRIG_TX	转到
10A0h	ISET	中断设置	DMA_TRIG_TX	转到
10A8h	ICLR	中断清除	DMA_TRIG_TX	转到
10E0h	EVT_MODE	事件模式		转到
10E4h	INTCTL	中断控制寄存器		转到
1100h	CTL0	SPI 控制寄存器 0		转到
1104h	CTL1	SPI 控制寄存器 1		转到
1108h	CLKCTL	时钟预分频器和分频器寄存器。		转到
110Ch	IFLS	中断 FIFO 级别选择寄存器		转到
1110h	STAT	状态寄存器		转到
1130h	RXDATA	RXDATA 寄存器		转到
1140h	TXDATA	TXDATA 寄存器		转到
1E00h	TEST0	测试 0 寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 17-8 展示了适用于此部分中访问类型的代码。

表 17-8. SPI 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
R-0	R -0	读取 返回 0
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

17.3.1 SCLK (偏移 = 4h) [复位 = 00000000h]

图 17-9 展示了 SCLK，表 17-9 中对此进行了介绍。

返回到汇总表。

SCLK 信号控制器：时钟输出外设：时钟输入

图 17-9. SCLK

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-9. SCLK 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-9. SCLK 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.2 MOSI（偏移 = 8h）[复位 = 00000000h]

图 17-10 展示了 MOSI，表 17-10 中对此进行了介绍。

返回到汇总表。

MOSI 信号控制器：数据输出外设：数据输入

图 17-10. MOSI

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-10. MOSI 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-10. MOSI 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.3 MISO (偏移 = Ch) [复位 = 00000000h]

图 17-11 展示了 MISO，表 17-11 中对此进行了介绍。

返回到汇总表。

MISO 信号控制器：数据输入外设：数据输出

图 17-11. MISO

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-11. MISO 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-11. MISO 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.4 CS0（偏移 = 18h）[复位 = 00000000h]

图 17-12 展示了 CS0，表 17-12 中对此进行了介绍。

返回到汇总表。

SPI 芯片选择 0：控制器：输出外设：输入

图 17-12. CS0

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-12. CS0 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-12. CS0 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.5 CS1_MISO1（偏移 = 1Ch）[复位 = 00000000h]

图 17-13 展示了 CS1_MISO1，表 17-13 中对此进行了介绍。

返回到汇总表。

SPI 芯片选择 1/MISO1 控制器：输出/输入外设：输出

图 17-13. CS1_MISO1

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-13. CS1_MISO1 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-13. CS1_MISO1 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.6 CS2_MISO2（偏移 = 20h）[复位 = 00000000h]

图 17-14 展示了 CS2_MISO2，表 17-14 中对此进行了介绍。

返回到汇总表。

SPI 芯片选择 2/MISO2 控制器：输出/输入外设：输出

图 17-14. CS2_MISO2

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-14. CS2_MISO2 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-14. CS2_MISO2 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.7 CS3_CD_MISO3（偏移 = 24h）[复位 = 00000000h]

图 17-15 展示了 CS3_CD_MISO3，表 17-15 中对此进行了介绍。

返回到汇总表。

SPI 芯片选择 3/命令数据/MISO3 控制器：输出/输出/输入外设：输出

图 17-15. CS3_CD_MISO3

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 17-15. CS3_CD_MISO3 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 17-15. CS3_CD_MISO3 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

17.3.8 SCLK (偏移 = 204h) [复位 = 00000000h]

图 17-16 展示了 SCLK , 表 17-16 中对此进行了介绍。

返回到汇总表。

SCLK 的 FUPDATE 版本

图 17-16. SCLK

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

表 17-16. SCLK 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.9 MOSI（偏移 = 208h）[复位 = 00000000h]

图 17-17 展示了 MOSI，表 17-17 中对此进行了介绍。

返回到汇总表。

MOSI 的 FUPDATE 版本

图 17-17. MOSI

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

表 17-17. MOSI 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.10 MISO (偏移 = 20Ch) [复位 = 00000000h]

图 17-18 展示了 MISO，表 17-18 中对此进行了介绍。

返回到汇总表。

MISO 的 FUPDATE 版本

图 17-18. MISO

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

表 17-18. MISO 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.11 CS0（偏移 = 218h）[复位 = 00000000h]

图 17-19 展示了 CS0，表 17-19 中对此进行了介绍。

返回到汇总表。

CS0 的 FUPDATE 版本

图 17-19. CS0

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

表 17-19. CS0 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.12 CS1_MISO1（偏移 = 21Ch）[复位 = 00000000h]

图 17-20 展示了 CS1_MISO1，表 17-20 中对此进行了介绍。

返回到汇总表。

CS1_MISO1 的 FUPDATE 版本

图 17-20. CS1_MISO1

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

表 17-20. CS1_MISO1 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.13 CS2_MISO2（偏移 = 220h）[复位 = 00000000h]

图 17-21 展示了 CS2_MISO2，表 17-21 中对此进行了介绍。

返回到汇总表。

CS2_MISO2 的 FUPDATE 版本

图 17-21. CS2_MISO2

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

表 17-21. CS2_MISO2 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.14 CS3_CD_MISO3（偏移 = 224h）[复位 = 00000000h]

图 17-22 展示了 CS3_CD_MISO3，表 17-22 中对此进行了介绍。

返回到汇总表。

CS3_CD_MISO3 的 FUPDATE 版本

图 17-22. CS3_CD_MISO3

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

表 17-22. CS3_CD_MISO3 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

17.3.15 CPU_CONNECT_0（偏移 = 480h）[复位 = 00000000h]

图 17-23 展示了 CPU_CONNECT_0，表 17-23 中对此进行了介绍。

返回到汇总表。

将外设发布者端口直接连接到应用处理器

图 17-23. CPU_CONNECT_0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						CPUSS0_CONN	RESERVED
R/W-0h						R/W-0h	R/W-0h

表 17-23. CPU_CONNECT_0 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	CPUSS0_CONN	R/W	0h	CPUSS0 连接位。 0h = CPU 未连接。 1h = CPU 已连接。
0	RESERVED	R/W	0h	

17.3.16 DMA_MAP_RX (偏移 = 504h) [复位 = 00h]

图 17-24 展示了 DMA_MAP_RX，表 17-24 中对此进行了介绍。

返回到汇总表。

DMA 中与此外设触发对应的触发端口 ID

图 17-24. DMA_MAP_RX

7	6	5	4	3	2	1	0
RESERVED	TRIG_ID						
R-0h	R-0h						

表 17-24. DMA_MAP_RX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R	0h	
6-0	TRIG_ID	R	0h	DMA 中与此外设触发对应的触发端口 ID 0h = 未选择触发器 1h = 已选择触发器 1 7Fh = 已选择触发器 127

17.3.17 DMA_TRIG_RX（偏移 = 505h）[复位 = 00h]

图 17-25 展示了 DMA_TRIG_RX，表 17-25 中对此进行了介绍。

返回到汇总表。

此外设触发的触发控制和状态寄存器

图 17-25. DMA_TRIG_RX

7	6	5	4	3	2	1	0
RESERVED	THRHLD			TRIGLOST	STATECLR	状态	
R/W-0h	R-0h			R-0h	R-0/W-0h	R-0h	

表 17-25. DMA_TRIG_RX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R/W	0h	
6-4	THRHLD	R	0h	DMA 接受触发请求的阈值。 0h = 可能的最低阈值 6h = 可能的最高阈值
3	TRIGLOST	R	0h	当触发端口处于 TRIGGER_PEND 或 TRIGGERED 时，只要接收到触发请求，就会设置粘滞标志。通过写入 STATECLR 来清除。 0h = 触发器未丢失 1h = 触发器已丢失
2	STATECLR	R-0/W	0h	清除触发状态。向该寄存器写入 1 会清除该端口上任何挂起的 DMA 触发器并将端口转换为未触发状态。 0h = 写入 0 不起作用 1h = 清除 DMA 触发器
1-0	状态	R	0h	返回 DMA tx 触发端口的当前状态 0h = 通道未触发 1h = 通道触发挂起 2h = 通道已触发

17.3.18 DMA_ENTRY_RX（偏移 = 506h）[复位 = 0FFFh]

图 17-26 展示了 DMA_ENTRY_RX，表 17-26 中对此进行了介绍。

返回到汇总表。

描述符连接到外设 DMA 触发器

图 17-26. DMA_ENTRY_RX

15	14	13	12	11	10	9	8
保留				ENTRY_ID			
R/W-				R/W-FFFh			
7	6	5	4	3	2	1	0
ENTRY_ID							
R/W-FFFh							

表 17-26. DMA_ENTRY_RX 字段说明

位	字段	类型	复位	说明
15-12	保留	R/W	0h	
11-0	ENTRY_ID	R/W	FFFh	此触发器被路由到的 DMA 描述符的 ID。这样可以确保另一个 DMA 通道不能监听或影响负责处理此外设数据的 DMA 通道。 0h = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 Fh = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 10h = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFEh = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFFh = 触发未启用

17.3.19 DMA_MAP_TX（偏移 = 508h）[复位 = 00h]

图 17-27 展示了 DMA_MAP_TX，表 17-27 中对此进行了介绍。

返回到汇总表。

DMA 中与此外设触发对应的触发端口 ID

图 17-27. DMA_MAP_TX

7	6	5	4	3	2	1	0
RESERVED	TRIG_ID						
R-0h	R-0h						

表 17-27. DMA_MAP_TX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R	0h	
6-0	TRIG_ID	R	0h	DMA 中与此外设触发对应的触发端口 ID 0h = 未选择触发器 1h = 已选择触发器 1 7Fh = 已选择触发器 127

17.3.20 DMA_TRIG_TX (偏移 = 509h) [复位 = 00h]

图 17-28 展示了 DMA_TRIG_TX，表 17-28 中对此进行了介绍。

返回到汇总表。

此外设触发的触发控制和状态寄存器

图 17-28. DMA_TRIG_TX

7	6	5	4	3	2	1	0
RESERVED	THRHLD			TRIGLOST	STATECLR	状态	
R/W-0h	R-0h			R-0h	R-0/W-0h	R-0h	

表 17-28. DMA_TRIG_TX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R/W	0h	
6-4	THRHLD	R	0h	DMA 接受触发请求的阈值。 0h = 可能的最低阈值 6h = 可能的最高阈值
3	TRIGLOST	R	0h	当触发端口处于 TRIGGER_PEND 或 TRIGGERED 时，只要接收到触发请求，就会设置粘滞标志。通过写入 STATECLR 来清除。 0h = 触发器未丢失 1h = 触发器已丢失
2	STATECLR	R-0/W	0h	清除触发状态。向该寄存器写入 1 会清除该端口上任何挂起的 DMA 触发器并将端口转换为未触发状态。 0h = 写入 0 不起作用 1h = 清除 DMA 触发器
1-0	状态	R	0h	返回 DMA tx 触发端口的当前状态 0h = 通道未触发 1h = 通道触发挂起 2h = 通道已触发

17.3.21 DMA_ENTRY_TX（偏移 = 50Ah）[复位 = 0FFFh]

图 17-29 展示了 DMA_ENTRY_TX，表 17-29 中对此进行了介绍。

返回到汇总表。

描述符连接到外设 DMA 触发器

图 17-29. DMA_ENTRY_TX

15	14	13	12	11	10	9	8
保留				ENTRY_ID			
R/W-				R/W-FFFh			
7	6	5	4	3	2	1	0
ENTRY_ID							
R/W-FFFh							

表 17-29. DMA_ENTRY_TX 字段说明

位	字段	类型	复位	说明
15-12	保留	R/W	0h	
11-0	ENTRY_ID	R/W	FFFh	此触发器被路由到的 DMA 描述符的 ID。这样可以确保另一个 DMA 通道不能监听或影响负责处理此外设数据的 DMA 通道。 0h = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 Fh = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 10h = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFEh = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFFh = 触发未启用

17.3.22 PWREN（偏移 = 800h）[复位 = 00000000h]

图 17-30 展示了 PWREN，表 17-30 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 17-30. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 17-30. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

17.3.23 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 17-31 展示了 RSTCTL，表 17-31 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 17-31. RSTCTL

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

表 17-31. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

17.3.24 CLKCFG（偏移 = 808h）[复位 = 00000000h]

图 17-32 展示了 CLKCFG，表 17-32 中对此进行了介绍。

返回到汇总表。

外设时钟配置寄存器

图 17-32. CLKCFG

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							BLOCKASYNC
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 17-32. CLKCFG 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许状态更改的 KEY - 0xA9 A9h = 允许更改 GPRCM 字段的密钥值
23-9	保留	R/W	0h	
8	BLOCKASYNC	R/W	0h	阻止异步时钟请求启动 SYSOSC 或强制总线时钟为 32MHz 0h = 不阻止异步时钟请求 1h = 阻止异步时钟请求
7-0	保留	R/W	0h	

17.3.25 STAT（偏移 = 814h）[复位 = 00000000h]

图 17-33 展示了 STAT，表 17-33 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 17-33. STAT

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

表 17-33. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

17.3.26 CLKDIV（偏移 = 1000h）[复位 = 00000000h]

图 17-34 展示了 CLKDIV，表 17-34 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块专用分频比

图 17-34. CLKDIV

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

表 17-34. CLKDIV 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 3 分频 3h = 对时钟源进行 4 分频 4h = 对时钟源进行 5 分频 5h = 对时钟源进行 6 分频 6h = 对时钟源进行 7 分频 7h = 对时钟源进行 8 分频

17.3.27 CLKSEL（偏移 = 1004h）[复位 = 00000000h]

图 17-35 展示了 CLKSEL，表 17-35 中对此进行了介绍。

返回到汇总表。

外设时钟源选择

图 17-35. CLKSEL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED				SYSCLK_SEL	MFCLK_SEL	LFCLK_SEL	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 17-35. CLKSEL 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	SYSCLK_SEL	R/W	0h	如果启用，选择 SYSCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
2	MFCLK_SEL	R/W	0h	如果启用，选择 MFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
1	LFCLK_SEL	R/W	0h	如果启用，选择 LFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
0	RESERVED	R/W	0h	

17.3.28 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 17-36 展示了 PDBGCTL，表 17-36 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 17-36. PDBGCTL

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

表 17-36. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设将阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

17.3.29 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 17-37 展示了 IIDX，表 17-37 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 17-37. IIDX

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

表 17-37. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 1h = RX FIFO 上溢事件/中断挂起 2h = 发送奇偶校验事件/中断挂起 3h = SPI 接收超时中断 4h = 接收事件/中断挂起 5h = 发送事件/中断挂起 6h = 发送缓冲器空事件/中断挂起 7h = 发送结束事件/中断挂起 8h = 接收 DMA 完成事件/中断挂起 9h = 发送 DMA 完成事件/中断挂起 Ah = TX FIFO 下溢中断 Bh = RX FIFO 已满中断

17.3.30 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 17-38 展示了 IMASK，表 17-38 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 17-38. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留					RXFULL	TXFIFO_UNF	DMA_DONE_TX
R/W-0h					R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
DMA_DONE_RX	IDLE（闲置）	TXEMPTY	TX	RX	RTOUT	PER	RXFIFO_OVF
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 17-38. IMASK 字段说明

位	字段	类型	复位	说明
31-11	保留	R/W	0h	
10	RXFULL	R/W	0h	RX FIFO 已满中断屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9	TXFIFO_UNF	R/W	0h	TX FIFO 下溢中断屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	DMA_DONE_TX	R/W	0h	TX 事件 DMA 完成 1 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	DMA_DONE_RX	R/W	0h	RX 事件 DMA 完成 1 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	IDLE（闲置）	R/W	0h	SPI 空闲事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
5	TXEMPTY	R/W	0h	发送 FIFO 空事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	TX	R/W	0h	发送 FIFO 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	RX	R/W	0h	接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则会设置该中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 17-38. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
2	RTOUT	R/W	0h	启用 SPI 接收超时事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	PER	R/W	0h	奇偶校验错误事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RXFIFO_OVF	R/W	0h	RXFIFO 上溢事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

17.3.31 RIS（偏移 = 1030h）[复位 = 00000000h]

图 17-39 展示了 RIS，表 17-39 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 17-39. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留					RXFULL	TXFIFO_UNF	DMA_DONE_TX
R-0h					R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMA_DONE_RX	IDLE（闲置）	TXEMPTY	TX	RX	RTOUT	PER	RXFIFO_OVF
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 17-39. RIS 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	R	0h	
10	RXFULL	R	0h	RX FIFO 已满中断 0h = 未发生中断 1h = 已发生中断
9	TXFIFO_UNF	R	0h	TX FIFO 下溢中断 0h = 未发生中断 1h = 已发生中断
8	DMA_DONE_TX	R	0h	TX 的 DMA 完成 1 事件。如果 TX DMA 通道发送 DONE 信号，则会设置此中断。这允许在映射的外设内部处理 DMA 事件。 0h = 未发生中断 1h = 已发生中断
7	DMA_DONE_RX	R	0h	RX 的 DMA 完成 1 事件。如果 RX DMA 通道发送 DONE 信号，则会设置此中断。这允许在映射的外设内部处理 DMA 事件。 0h = 未发生中断 1h = 已发生中断
6	IDLE（闲置）	R	0h	SPI 已完成传输并更改为空闲模式。当 BUSY 变为低电平时会设置该位。 0h = 未发生中断 1h = 已发生中断
5	TXEMPTY	R	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移到移位寄存器，则会设置该中断。 0h = 未发生中断 1h = 已发生中断
4	TX	R	0h	发送 FIFO 事件。如果已达到选定的发送 FIFO 级别，则会设置该中断。 0h = 未发生中断 1h = 已发生中断

表 17-39. RIS 字段说明 (continued)

位	字段	类型	复位	说明
3	RX	R	0h	接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则会设置该中断 0h = 未发生中断 1h = 已发生中断
2	RTOUT	R	0h	SPI 接收超时事件。 0h = 未发生中断 1h = 已发生中断
1	PER	R	0h	奇偶校验错误事件：如果检测到奇偶校验错误，则会设置该位 0h = 未发生中断 1h = 已发生中断
0	RXFIFO_OVF	R	0h	RXFIFO 溢出事件。如果检测到 RX FIFO 溢出，则设置此中断。 0h = 未发生中断 1h = 已发生中断

17.3.32 MIS (偏移 = 1038h) [复位 = 00000000h]

图 17-40 展示了 MIS，表 17-40 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 17-40. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留					RXFULL	TXFIFO_UNF	DMA_DONE_TX
R-0h					R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
DMA_DONE_RX	IDLE (闲置)	TXEMPTY	TX	RX	RTOUT	PER	RXFIFO_OVF
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 17-40. MIS 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	R	0h	
10	RXFULL	R	0h	RX FIFO 已满中断 0h = 未发生中断 1h = 已发生中断
9	TXFIFO_UNF	R	0h	TX FIFO 下溢中断 0h = 未发生中断 1h = 已发生中断
8	DMA_DONE_TX	R	0h	已屏蔽 TX 的 DMA 完成 1 事件。 0h = 未发生中断 1h = 已发生中断
7	DMA_DONE_RX	R	0h	已屏蔽 RX 的 DMA 完成 1 事件。 0h = 未发生中断 1h = 已发生中断
6	IDLE (闲置)	R	0h	已屏蔽 SPI 空闲模式事件。 0h = 未发生中断 1h = 已发生中断
5	TXEMPTY	R	0h	已屏蔽发送 FIFO 空事件。 0h = 未发生中断 1h = 已发生中断
4	TX	R	0h	已屏蔽发送 FIFO 事件。如果已达到选定的发送 FIFO 级别，则会设置该中断。 0h = 未发生中断 1h = 已发生中断
3	RX	R	0h	已屏蔽接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则会设置该中断 0h = 未发生中断 1h = 已发生中断

表 17-40. MIS 字段说明 (continued)

位	字段	类型	复位	说明
2	RTOUT	R	0h	已屏蔽 SPI 接收超时中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	PER	R	0h	已屏蔽奇偶校验错误事件：如果检测到奇偶校验错误，则会设置该位 0h = 未发生中断 1h = 已发生中断
0	RXFIFO_OVF	R	0h	已屏蔽 RXFIFO 上溢事件。如果检测到 RX FIFO 溢出，则设置此中断。 0h = 未发生中断 1h = 已发生中断

17.3.33 ISET (偏移 = 1040h) [复位 = 00000000h]

图 17-41 展示了 ISET，表 17-41 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 17-41. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留					RXFULL	TXFIFO_UNF	DMA_DONE_TX
W-0h					W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMA_DONE_RX	IDLE (闲置)	TXEMPTY	TX	RX	RTOUT	PER	RXFIFO_OVF
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 17-41. ISET 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	W	0h	
10	RXFULL	W	0h	设置 RX FIFO 已满事件 0h = 写入无效 1h = 设置中断
9	TXFIFO_UNF	W	0h	设置 TX FIFO 下溢事件 0h = 写入无效 1h = 设置中断
8	DMA_DONE_TX	W	0h	为 TX 设置 DMA 完成 1 事件。 0h = 写入 0 无效 1h = 设置中断
7	DMA_DONE_RX	W	0h	为 RX 设置 DMA 完成 1 事件。 0h = 写入 0 无效 1h = 设置中断
6	IDLE (闲置)	W	0h	设置 SPI 空闲模式事件。 0h = 写入 0 无效 1h = 设置中断
5	TXEMPTY	W	0h	设置发送 FIFO 空事件。 0h = 写入 0 无效 1h = 设置中断
4	TX	W	0h	设置发送 FIFO 事件。 0h = 写入 0 无效 1h = 设置中断
3	RX	W	0h	设置接收 FIFO 事件。 0h = 写入 0 无效 1h = 设置中断

表 17-41. ISET 字段说明 (continued)

位	字段	类型	复位	说明
2	RTOUT	W	0h	设置 SPI 接收超时事件。 0h = 写入 0 无效 1h = 设置中断屏蔽
1	PER	W	0h	设置奇偶校验错误事件。 0h = 写入 0 无效 1h = 设置中断
0	RXFIFO_OVF	W	0h	设置 RXFIFO 上溢事件。 0h = 写入 0 无效 1h = 设置中断

17.3.34 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 17-42 展示了 ICLR，表 17-42 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 17-42. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
保留					RXFULL	TXFIFO_UNF	DMA_DONE_TX
W-0h					W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
DMA_DONE_RX	IDLE（闲置）	TXEMPTY	TX	RX	RTOUT	PER	RXFIFO_OVF
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 17-42. ICLR 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	W	0h	
10	RXFULL	W	0h	清除 RX FIFO 下溢事件 0h = 写入无效 1h = 清除中断
9	TXFIFO_UNF	W	0h	清除 TXFIFO 下溢事件 0h = 写入无效 1h = 清除中断
8	DMA_DONE_TX	W	0h	清除 TX 的 DMA 完成 1 事件。 0h = 写入 0 无效 1h = 清除中断
7	DMA_DONE_RX	W	0h	清除 RX 的 DMA 完成 1 事件。 0h = 写入 0 无效 1h = 清除中断
6	IDLE（闲置）	W	0h	清除 SPI 空闲模式事件。 0h = 写入 0 无效 1h = 清除中断
5	TXEMPTY	W	0h	清除发送 FIFO 空事件。 0h = 写入 0 无效 1h = 清除中断
4	TX	W	0h	清除发送 FIFO 事件。 0h = 写入 0 无效 1h = 清除中断
3	RX	W	0h	清除接收 FIFO 事件。 0h = 写入 0 无效 1h = 清除中断
2	RTOUT	W	0h	清除 SPI 接收超时事件。 0h = 写入 0 无效 1h = 设置中断屏蔽

表 17-42. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
1	PER	W	0h	清除奇偶校验错误事件。 0h = 写入 0 无效 1h = 清除中断
0	RXFIFO_OVF	W	0h	清除 RXFIFO 上溢事件。 0h = 写入 0 无效 1h = 清除中断

17.3.35 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 17-43 展示了 IIDX，表 17-43 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 17-43. IIDX

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

表 17-43. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 3h = SPI 接收超时中断 4h = 接收事件/中断挂起

17.3.36 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 17-44 展示了 IMASK，表 17-44 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 17-44. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED				RX	RTOUT	RESERVED	
R/W-0h				R/W-0h	R/W-0h	R/W-0h	

表 17-44. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	RX	R/W	0h	接收 FIFO 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	RTOUT	R/W	0h	SPI 接收超时事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1-0	保留	R/W	0h	

17.3.37 RIS（偏移 = 1060h）[复位 = 00000000h]

图 17-45 展示了 RIS，表 17-45 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 17-45. RIS

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

表 17-45. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	RX	R	0h	接收 FIFO 事件。如果已达到选定的接收 FIFO 级别，则会设置该中断 0h = 未发生中断 1h = 已发生中断
2	RTOUT	R	0h	SPI 接收超时事件。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1-0	RESERVED	R	0h	

17.3.38 MIS (偏移 = 1068h) [复位 = 00000000h]

图 17-46 展示了 MIS，表 17-46 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 17-46. MIS

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

表 17-46. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	RX	R	0h	接收 FIFO 事件屏蔽。 0h = 未发生中断 1h = 已发生中断
2	RTOUT	R	0h	SPI 接收超时事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1-0	RESERVED	R	0h	

17.3.39 ISET (偏移 = 1070h) [复位 = 00000000h]

图 17-47 展示了 ISET，表 17-47 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 17-47. ISET

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

表 17-47. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	RX	W	0h	设置接收 FIFO 事件。 0h = 写入 0 无效 1h = 设置中断
2	RTOUT	W	0h	设置 SPI 接收超时事件。 0h = 写入 0 无效 1h = 设置中断屏蔽
1-0	RESERVED	W	0h	

17.3.40 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 17-48 展示了 ICLR，表 17-48 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 17-48. ICLR

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

表 17-48. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	RX	W	0h	清除接收 FIFO 事件。 0h = 写入 0 无效 1h = 清除中断
2	RTOUT	W	0h	清除 SPI 接收超时事件。 0h = 写入 0 无效 1h = 设置中断屏蔽
1-0	RESERVED	W	0h	

17.3.41 IIDX (偏移 = 1080h) [复位 = 00000000h]

图 17-49 展示了 IIDX，表 17-49 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断 (最高优先级) 由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU (不是从调试接口) 读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 17-49. IIDX

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

表 17-49. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 5h = 发送事件/中断挂起

17.3.42 IMASK（偏移 = 1088h）[复位= 00000000h]

图 17-50 展示了 IMASK，表 17-50 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 17-50. IMASK

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

表 17-50. IMASK 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	0h	
4	TX	R/W	0h	发送 FIFO 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3-0	RESERVED	R/W	0h	

17.3.43 RIS（偏移 = 1090h）[复位 = 00000000h]

图 17-51 展示了 RIS，表 17-51 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 17-51. RIS

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

表 17-51. RIS 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	
4	TX	R	0h	发送 FIFO 事件：读取操作返回发送 FIFO 中断的当前屏蔽。写入 1 时会设置发送 FIFO 中断的屏蔽，这意味着中断状态将反映在 MIS.TXMIS 中。写入 0 会清除屏蔽，这意味着 MIS.TXMIS 不会反映中断。 0h = 未发生中断 1h = 已发生中断
3-0	RESERVED	R	0h	

17.3.44 MIS (偏移 = 1098h) [复位 = 00000000h]

图 17-52 展示了 MIS，表 17-52 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 17-52. MIS

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

表 17-52. MIS 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	
4	TX	R	0h	已屏蔽发送 FIFO 事件 0h = 未发生中断 1h = 已发生中断
3-0	RESERVED	R	0h	

17.3.45 ISET（偏移 = 10A0h）[复位 = 00000000h]

图 17-53 展示了 ISET，表 17-53 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 17-53. ISET

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

表 17-53. ISET 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	W	0h	
4	TX	W	0h	设置发送 FIFO 事件。 0h = 写入 0 无效 1h = 设置中断
3-0	RESERVED	W	0h	

17.3.46 ICLR（偏移 = 10A8h）[复位 = 00000000h]

图 17-54 展示了 ICLR，表 17-54 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 17-54. ICLR

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

表 17-54. ICLR 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	W	0h	
4	TX	W	0h	清除发送 FIFO 事件。 0h = 写入 0 无效 1h = 清除中断
3-0	RESERVED	W	0h	

17.3.47 EVT_MODE（偏移 = 10E0h）[复位 = 00000029h]

图 17-55 展示了 EVT_MODE，表 17-55 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线路

图 17-55. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
ESERVED		INT2_CFG		INT1_CFG		INT0_CFG	
R/W-		R-2h		R-2h		R-1h	

表 17-55. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	INT2_CFG	R	2h	none.INT_EVENT2 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	INT1_CFG	R	2h	对应于 none.INT_EVENT1 的事件的事件线路模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	对应于 none.INT_EVENT0 的事件的事件线路模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

17.3.48 INTCTL（偏移 = 10E4h）[复位 = 00000000h]

图 17-56 展示了 INTCTL，表 17-56 中对此进行了介绍。

返回到汇总表。

中断控制寄存器

图 17-56. INTCTL

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

表 17-56. INTCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	INTEVAL	W	0h	向该字段写入 1 会重新评估中断源。 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。

17.3.49 CTL0（偏移 = 1100h）[复位 = 00000000h]

图 17-57 展示了 CTL0，表 17-57 中对此进行了介绍。

返回到汇总表。

SPI 控制寄存器 0

图 17-57. CTL0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	CSCLR	CSSEL		RESERVED		SPH	SPO
R/W-0h	R/W-0h	R/W-0h		R/W-0h		R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
PACKEN	FRF		DSS				
R/W-0h	R/W-0h		R/W-0h				

表 17-57. CTL0 字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14	CSCLR	R/W	0h	CS 无效时清除移位寄存器计数器 该位仅在外设模式 (CTL1.CP=0) 中适用。 0h = 禁止在 CS 变为禁用状态时自动清除移位寄存器。 1h = 允许在 CS 变为禁用状态时自动清除移位寄存器。
13-12	CSSEL	R/W	0h	选择要在数据传输时控制的 CS 线路 该位同时适用于控制器/目标模式 0h (R/W) = CS 线路选择：0 1h (R/W) = CS 线路选择： 1 2h (R/W) = CS 线路选择： 2 3h (R/W) = CS 线路选择： 3
11-10	RESERVED	R/W	0h	
9	SPH	R/W	0h	CLKOUT 相位（仅适用于 Motorola SPI 帧格式） 该位选择捕获数据的时钟沿并使其能够更改状态。 由于可以允许或不允许在第一个数据捕获沿之前进行时钟转换，所以对第一个发送的位的影响最大。 0h = 在第一次时钟沿转换时捕获数据。 1h = 在第二次时钟沿转换时捕获数据。
8	SPO	R/W	0h	CLKOUT 极性（仅适用于 Motorola SPI 帧格式） 0h = SPI 在 CLKOUT 上产生稳态低电平值 1h = SPI 在 CLKOUT 上产生稳态高电平值
7	PACKEN	R/W	0h	打包使能。 为 1 时，在 IP 内部启用打包功能 为 0 时，在 IP 内部禁用打包功能 0h = 禁用打包功能 1h = 启用打包功能
6-5	FRF	R/W	0h	帧格式选择 0h = Motorola SPI 帧格式（3 线制模式） 1h = Motorola SPI 帧格式（4 线制模式） 2h = TI 同步串行帧格式 3h = National Microwire 帧格式

表 17-57. CTL0 字段说明 (continued)

位	字段	类型	复位	说明
4-0	DSS	R/W	0h	数据大小选择。 值 0 - 2 是保留值，不得使用。 3h = 4_BIT：4 位数据 SPI 仅允许不超过 16 位的值 3h (R/W) = 数据大小选择位：4 4h (R/W) = 数据大小选择位： 5 5h (R/W) = 数据大小选择位： 6 6h (R/W) = 数据大小选择位： 7 7h (R/W) = 数据大小选择位： 8 8h (R/W) = 数据大小选择位： 9 9h (R/W) = 数据大小选择位： 10 Ah (R/W) = 数据大小选择位： 11 Bh (R/W) = 数据大小选择位： 12 Ch (R/W) = 数据大小选择位： 13 Dh (R/W) = 数据大小选择位： 14 Eh (R/W) = 数据大小选择位： 15 Fh (R/W) = 数据大小选择位： 16

17.3.50 CTL1（偏移 = 1104h）[复位 = 00000004h]

图 17-58 展示了 CTL1，表 17-58 中对此进行了介绍。

返回到汇总表。

SPI 控制寄存器 1

图 17-58. CTL1

31	30	29	28	27	26	25	24
RESERVED				RXTIMEOUT			
R/W-0h				R/W-0h			
23	22	21	20	19	18	17	16
REPEATTX							
R/W-0h							
15	14	13	12	11	10	9	8
CDMODE				CDENABLE	RESERVED		PTEN
R/W-0h				R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
RESERVED	PES	PREN	MSB	POD	CP	LBM	ENABLE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-0h

表 17-58. CTL1 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29-24	RXTIMEOUT	R/W	0h	接收超时（仅适用于外设模式）。定义在设置接收超时标志 RTOUT 之前或之后的时钟周期数。在控制器模式配置中通过用于时钟选择和分频器的控制寄存器来计算该时间。值为 0 将禁用此功能。 0h = 最小值 3Fh = 尽可能高的值
23-16	REPEATTX	R/W	0h	重复上一次传输的计数器。0：禁用重复上一次传输。x：按给定的次数重复上一次传输。传输将通过将数据写入 Tx 缓冲区开始，发送数据的过程将按给定的值重复，因此数据将总共传输 X+1 次。该行为与将数据写入 Tx 缓冲区的行为相同，并与此处的值定义的次数相同。这可用于清理传输，或由外设提取特定数量的数据。 0h = 最小值 FFh = 尽可能高的值
15-12	CDMODE	R/W	0h	命令/数据模式值 当 CTL1.CDENABLE 为 1 时，CS3 线路用作 C/D 信号来区分命令（C/D 低电平）和数据（C/D 高电平）信息。 将一个值写入 CTL1.CDMODE 位后，C/D (CS3) 线路将在 SPI 发送给定数量的字节期间变为低电平（从下一个要发送的值开始），此后 C/D 线路将自动变为高电平 0：手动模式，C/D 信号为高电平。 1-14：在发送此字节数量期间，C/D 为低电平，此后该字段设置为 0 并且 C/D 变为高电平。任何时候读取该字段都会返回剩余的命令字节数。 15：手动模式，C/D 信号为低电平。 0h = 手动模式：数据 0h = 最小值 Fh = 手动模式： 命令
11	CDENABLE	R/W	0h	命令/数据模式使能 0h = CS3 用于芯片选择 1h = CS3 用作 CD 信号
10-9	保留	R/W	0h	

表 17-58. CTL1 字段说明 (continued)

位	字段	类型	复位	说明
8	PTEN	R/W	0h	奇偶校验发送使能 如果启用，则会对控制器和外设模式都执行奇偶校验发送。 0h = 禁用奇偶校验发送 1h = 启用奇偶校验发送
7	RESERVED	R/W	0h	
6	PES	R/W	0h	偶校验选择 0h = 奇校验模式 1h = 偶校验模式
5	PREN	R/W	0h	奇偶校验接收使能 如果启用，则会对控制器和外设模式都执行奇偶校验接收检查 在奇偶校验不匹配的情况下，将设置奇偶校验错误标志 RIS.PER。 0h = 禁用奇偶校验接收功能 1h = 启用奇偶校验接收功能
4	MSB	R/W	0h	MSB 优先选择。控制移位寄存器接收和发送的方向。 0h = LSB 在前 1h = MSB 在前
3	POD	R/W	0h	外设模式：禁用数据输出 该位仅在外设模式下适用。在多外设系统拓扑中，SPI 控制器可以向所有外设广播消息，而只有一个外设驱动线路。 POD 可由 SPI 外设用于禁用线路上的驱动数据。 0h = SPI 可在外设模式下驱动 MISO 输出。 1h = SPI 无法在外设模式下驱动 MISO 输出。
2	CP	R/W	1h	控制器或外设模式选择。仅当禁用 SPI (CTL1.ENABLE=0) 时才能修改该位。 0h = 选择外设模式 1h = 选择控制器模式
1	LBM	R/W	0h	环回模式 0h = 禁用环回模式 1h = 启用环回模式
0	ENABLE	R/W	0h	SPI 使能 0h = 禁用模块功能 1h = 启用模块功能

17.3.51 CLKCTL（偏移 = 1108h）[复位 = 00000000h]

图 17-59 展示了 CLKCTL，表 17-59 中对此进行了介绍。

返回到汇总表。

时钟预分频器和分频器寄存器。该寄存器包含时钟预分频器和分频器设置。

图 17-59. CLKCTL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSAMPLE				RESERVED											
R/W-0h				R/W-0h											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESERVED						SCR									
R/W-0h						R/W-0h									

表 17-59. CLKCTL 字段说明

位	字段	类型	复位	说明
31-28	DSAMPLE	R/W	0h	延迟采样值。 在控制器模式下，输入引脚上的数据将被内部功能时钟定义的时钟周期延迟采样，从而放宽输入数据的设置时间。在系统中，如果电路板延迟和外部外设延迟超过控制器输入设置时间，该设置将非常有用。请参阅数据表了解控制器输入设置时间的值，并评估满足系统要求的 DSAMPLE 值。 注意：过高的 DSAMPLE 值可能会导致 HOLD 时间违规，必须在计算中考虑这一点。 0h = 最小值 Fh = 尽可能高的值
27-10	保留	R/W	0h	
9-0	SCR	R/W	0h	串行时钟分频器：这用于生成 SPI 的发送和接收比特率。SPI 比特率为 (SPI 的功能时钟频率)/((SCR+1)*2)。SCR 是 0-1023 之间的值。 0h = 最小值 3FFh = 尽可能高的值

17.3.52 IFLS（偏移 = 110Ch）[复位 = 00000012h]

图 17-60 展示了 IFLS，表 17-60 中对此进行了介绍。

返回到汇总表。

IFLS 寄存器是中断 FIFO 级别选择寄存器。该寄存器可用于定义触发 TX、RX 和超时中断标志的电平。中断是在 FIFO 深度从不满足触发条件到满足触发条件的跳变沿产生的。简单来说，中断是在 FIFO 深度越过触发门限时产生的。例如，如果接收触发电平设置为中途标志，则会在接收 FIFO 中填充了两个或多个字符时触发中断。复位后，TXIFLSEL 和 RXIFLSEL 位域均默认设置为 1/2 触发深度。

图 17-60. IFLS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED															
R/W-0h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESERVED										RXIFLSEL			TXIFLSEL		
R/W-0h										R/W-2h			R/W-2h		

表 17-60. IFLS 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-3	RXIFLSEL	R/W	2h	SPI 接收中断 FIFO 级别选择。接收中断的触发点如下： 0h = 保留 1h = RX FIFO >= 1/4 满 2h = RX FIFO >= 1/2 满（默认值） 3h = RX FIFO >= 3/4 满 4h = 保留 5h = RX FIFO 已满 6h = 保留 7h = 当 RX FIFO 包含 >= 1 帧时触发
2-0	TXIFLSEL	R/W	2h	SPI 发送中断 FIFO 级别选择。发送中断的触发点如下： 0h = 保留 1h = TX FIFO <= 3/4 空 2h = TX FIFO <= 1/2 空（默认值） 3h = TX FIFO <= 1/4 空 4h = 保留 5h = TX FIFO 为空 6h = 保留 7h = 当 TX FIFO 具有 >= 1 个可用帧时触发

17.3.53 STAT（偏移 = 1110h）[复位 = 0000000Fh]

图 17-61 展示了 STAT，表 17-61 中对此进行了介绍。

返回到汇总表。

状态寄存器

图 17-61. STAT

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED							
R-							
15	14	13	12	11	10	9	8
RESERVED							
R-							
7	6	5	4	3	2	1	0
ESERVED			BUSY	RNF	RFE	TNF	TFE
R-			R-0h	R-1h	R-1h	R-1h	R-1h

表 17-61. STAT 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	
4	忙	R	0h	忙状态 0h = SPI 处于空闲模式。 1h = SPI 当前正在发送和/或接收数据，或者发送 FIFO 不为空。
3	RNF	R	1h	接收 FIFO 未满 0h = 接收 FIFO 已满。 1h = 接收 FIFO 未满。
2	RFE	R	1h	接收 FIFO 为空。 0h = 接收 FIFO 不为空。 1h = 接收 FIFO 为空。
1	TNF	R	1h	发送 FIFO 未满 0h = 发送 FIFO 已满。 1h = 发送 FIFO 未满。
0	TFE	R	1h	发送 FIFO 为空。 0h = 发送 FIFO 不为空。 1h = 发送 FIFO 为空。

17.3.54 RXDATA（偏移 = 1130h）[复位 = 00000000h]

图 17-62 展示了 RXDATA，表 17-62 中对此进行了介绍。

返回到汇总表。

RXDATA 寄存器
读取该寄存器将返回 FIFO 的值。如果 FIFO 为空，则返回最后读取的值。
写入无效并被忽略。
当 PACKEN=1 时，FIFO 的两个条目作为 32 位值返回。当 PACKEN=0 时，FIFO 的一个条目作为 16 位值返回。

图 17-62. RXDATA

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

表 17-62. RXDATA 字段说明

位	字段	类型	复位	说明
31-0	数据	R	0h	接收的数据 当 PACKEN=1 时，FIFO 的两个条目作为 32 位值返回。当 PACKEN=0 时，FIFO 的一个条目作为 16 位值返回。 当接收逻辑从传入数据帧中删除数据值时，会将这些数据值放入接收 FIFO 内部由当前 FIFO 写入指针指向的条目中。 接收到的小于 16 位的数据在接收缓冲器中自动右对齐。 0h = 最小值 FFFFFFFFh = 尽可能高的值

17.3.55 TXDATA（偏移 = 1140h）[复位 = 00000000h]

图 17-63 展示了 TXDATA，表 17-63 中对此进行了介绍。

返回到汇总表。

TXDATA 寄存器
写入操作将数据放入 TX FIFO 中。读取该寄存器将返回最后写入的值。
当 PACKEN=0 时，只会将写入寄存器的数据的低 16 位传输到一个 16 位宽的 TX FIFO 条目
当 PACKEN=1 时，32 位写入数据的高 16 位和低 16 位都会传输到两个 16 位宽的 TX FIFO 条目

图 17-63. TXDATA

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

表 17-63. TXDATA 字段说明

位	字段	类型	复位	说明
31-0	数据	R/W	0h	发送数据 读取时将返回最后写入的值。如果对该字段的最后一次写入是 32 位写入 (PACKEN=1)，则将返回 32 位，如果最后一次写入是 16 位写入 (PACKEN=0)，则将返回这些 16 位。 写入时将根据 PACKEN 的值写入一个或两个 FIFO 条目。发送逻辑从发送 FIFO 中一次移出一个数据值。这个数据值加载到发送串行移位器中，然后以编程的比特率串行移出到 TXD 输出引脚。 当所选的数据大小小于 16 位时，用户必须将写入发送 FIFO 的数据右对齐。发送逻辑会忽略未使用的位。 0h = 最小值 FFFFFFFFh = 尽可能高的值

17.3.56 TEST0（偏移 = 1E00h）[复位 = 00000000h]

图 17-64 展示了 TEST0，表 17-64 中对此进行了介绍。

返回到汇总表。

测试 0 寄存器。

图 17-64. TEST0

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

表 17-64. TEST0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	DTB_MUX_SEL	R/W	0h	该位字段用于选择 DTB 多路复用器数字输出信号。 0h = 禁用 DTB 多路复用器 1h = 选择测试组 1 2h = 选择测试组 2 3h = 选择测试组 3 4h =选择测试组 4 5h = 选择测试组 5 6h = 选择测试组 6 7h = 选择测试组 7 8h = 选择测试组 8 9h = 选择测试组 9

This page intentionally left blank.

TEXAS INSTRUMENTS

I²C 模块提供了一个标准化串行接口，用于在 MSP 器件和其他外部 I²C 器件（例如传感器、存储器或 DAC）之间传输数据。

18.1 I²C 概述

I²C 模块的一个标准化串行接口可在 MSP 器件与其他外部 I²C 器件（例如传感器、存储器或 DAC）之间传输数据。

18.1.1 外设的用途

I²C 外设通过由数据 (SDA) 和时钟 (SCL) 线路组成的两线制串行总线提供双向数据传输。I²C 总线广泛用于连接电池管理 IC、传感器、其他 MCU 等器件。该 I²C 外设能够与总线上的其他 I²C 器件之间进行发送和接收。

18.1.2 特性

控制器包含 I²C 模块，其特性如下：

- I²C 总线上的器件可指定为控制器或具有 7 位寻址的目标器件。
- 支持四种 I²C 模式
  - 控制器发送
  - 控制器接收
  - 目标发送
  - 目标接收
- 支持的传输速度：
  - 标准模式 (SM)，比特率高达 100kbps
  - 快速模式 (FM)，比特率高达 400kbps
  - 超快速模式 (FM+)，比特率高达 1Mbps
- 用于接收和发送的独立 8 字节 FIFO
- 双目标地址能力
- 故障抑制
- 独立控制器和目标中断生成
- 具有仲裁、时钟同步和多控制器支持的控制器运行
- 对 SMBus 和 PMBus 的硬件支持
  - 时钟低电平超时检测和中断
  - 快速命令功能
- 为 DMA 提供硬件支持，具有独立的发送通道和接收通道

18.1.3 功能方框图

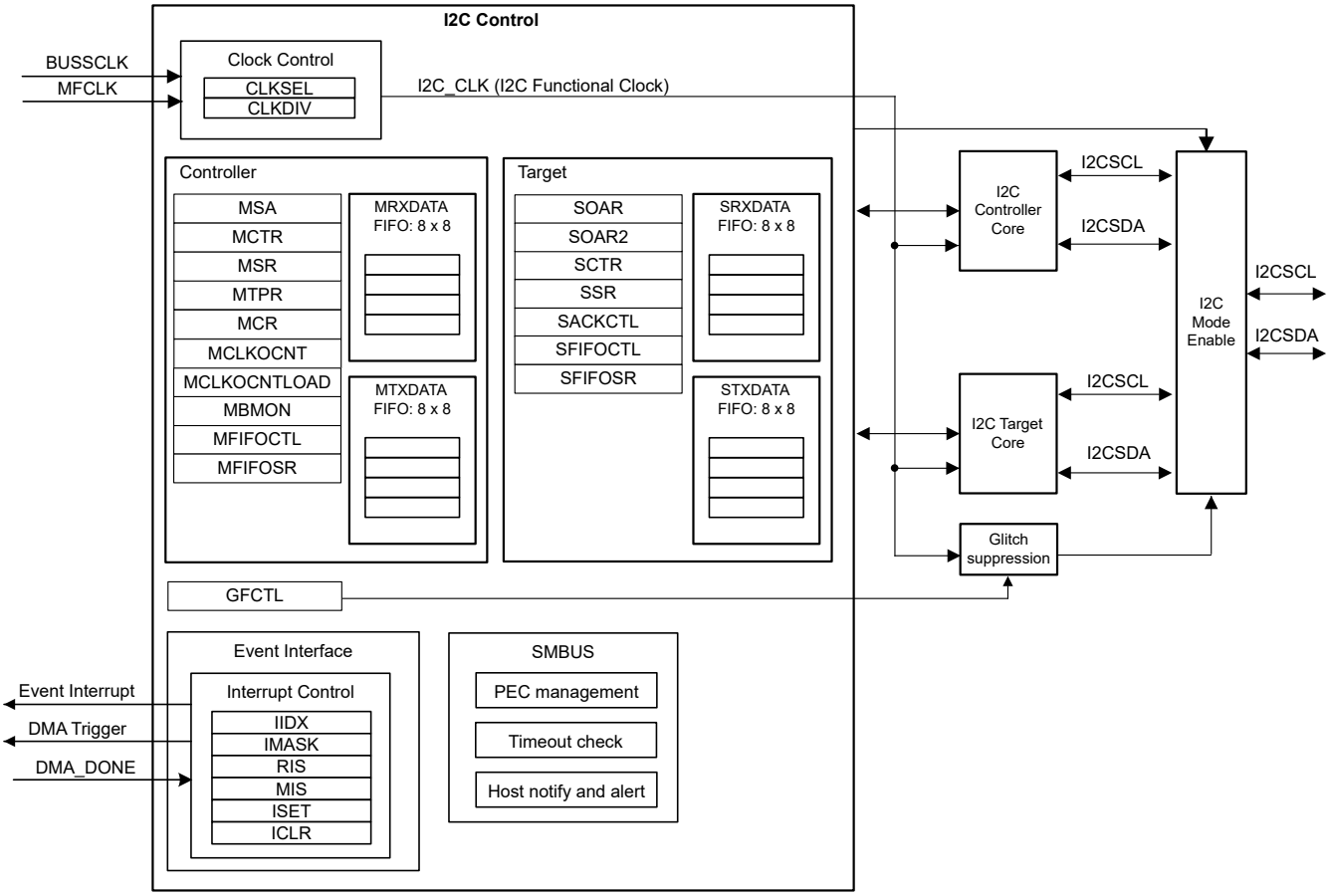


图 18-1. I2C 功能方框图

18.1.4 环境和外部连接

I²C 模式支持任何与 I²C 兼容的目标器件或控制器。图 18-2 展示了一个 I²C 总线的示例。每个 I²C 外设实例均由控制器和目标函数组成，其中可以使用 2 个独立的用户定义地址来寻找目标器件地址。进行数据传输时，一个连接到 I²C 总线的器件可视为控制器或目标器件。控制器启动数据传输并生成时钟信号 SCL。任何由控制器寻址的器件均视为目标器件。

I²C 数据是用串行数据 (SDA) 引脚和串行时钟 (SCL) 引脚进行通信的。SDA 和 SCL 均是开漏双向引脚，必须使用上拉电阻连接到正电源电压。

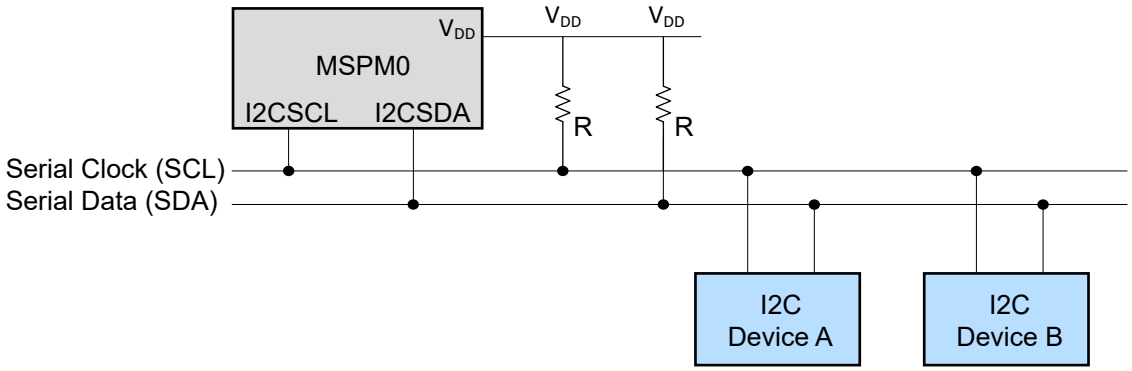


图 18-2. 总线连接图

18.2 I²C 操作

本节介绍了 I²C 外设的操作。

18.2.1 时钟控制

18.2.1.1 时钟选择和 I²C 速度

可以使用 I²C 控制器计时器周期 (I2CMTPR) 寄存器中的值来选择标准、快速和快速+ 模式，从而得到最高的 SCL 频率如下：

- 标准模式下为 100kbps
- 快速模式下为 400kbps
- 快速+ 模式下为 1Mbps

I²C 频率 I2C_FREQ 由 I2C_CLK 频率以及位字段 TPR、SCL_LP 和 SCL_HP 确定，其中：

- I2C_CLK 是 I²C 模块的功能时钟频率。请注意，I²C 内部功能时钟首先从源时钟分频：
  - 使用 I2Cx.CLKSEL 寄存器选择 I²C 功能时钟的时钟源
    - BUSSCLK：当前总线时钟被选为 I²C 的时钟源。当前总线时钟取决于电源域。如果 I²C 实例位于电源域 1 (PD1) 中，请参阅 MCLK，如果 I²C 实例位于电源域 0 (PD0) 中，请参阅 ULPCLK。
    - MFCLK：选择 MFCLK 作为 I²C 的时钟源，请参阅 MFCLK。
  - 使用 I2Cx.CLKDIV 寄存器选择 I²C 功能时钟的分频比，选项为 1 分频至 8 分频。
- SCL_LP 为 SCL 的低相位（必须固定为 6）
- SCL_HP 为 SCL 的高相位（必须固定为 4）
- TPR 是 I2Cx.MTPR 寄存器中的 TPR 位的编程值。通过替换以下公式中的已知变量即可获得 TPR 值

I²C 频率的计算公式如下：

$$I2C\_FREQ = I2C\_CLK / ((1+TPR) * (SCL\_LP + SCL\_HP)) \quad (24)$$

例如，如果 I²C 功能时钟频率为 32MHz，目标 SCL 频率为 400kHz：

I2C_CLK = 32MHz

I2C_FREQ = 400kHz

SCL_LP = 6，SCL_HP = 4

TPR = (I2C_CLK / (I2C_FREQ * (4 + 6))) - 1

TPR = 7 (0x07)

表 18-1. 典型时钟配置的控制器时钟设置示例

功能时钟	TPR 位 标准模式 100kHz SCL	TPR 位 快速模式 400kHz SCL	TPR 位 快速+ 模式 1000kHz SCL
4 MHz	0x03	-	-
8MHz	0x07	0x01	-
20MHz	0x13	0x04	0x01
32MHz	0x1F	0x07	0x02(1)
40MHz	0x27	0x09	0x03

(1) 对于 32MHz 功能时钟，TPR = 0x01 会产生 1.6MHz SCL 频率，而 TPR = 0x02 会产生 1.067MHz SCL 频率。

I²C 功能时钟必须大于或等于 SCL 频率的 20 倍，即 I2C_CLK ≥ 20 × I2C_FREQ。

以特定的 I2C 时钟速度运行时，需要以下最低功能时钟频率：

- 当 I²C 速度为 0kHz 至 100kHz 时，I2C_CLK ≥ 2MHz

- 当 I²C 速度为 100kHz 至 400kHz 时，I2C_CLK ≥ 8MHz
- 当 I²C 速度为 400kHz 至 1MHz 时，I2C_CLK ≥ 20MHz

18.2.1.2 时钟启动

所选的时钟源始终可用，频率取决于电源模式。有关更多信息，请参阅 PMU/时钟部分。通过置位 I2C.PWREN.ENABLE 位启用 I²C 模块后，该模块即准备好开始接收和发送数据。

18.2.2 信号说明

I²C 总线包含时钟信号和数据信号。时钟信号可以在内部（在控制器运行期间）或外部（在目标运行期间）产生。

表 18-2. I²C 信号说明

器件引脚	功能
I2Cx_SCL	I²C 时钟信号
I2Cx_SDA	I²C 数据信号

18.2.3 通用架构

18.2.3.1 I²C 总线功能概览

I²C 总线只使用两个信号：SDA 和 SCL。SDA 是双向串行数据线，SCL 是双向串行时钟线。当这两条线路都处于高电平状态并且没有传输正在进行时，则认为总线处于空闲状态。

I²C 总线上每个事务的长度为 9 位，其中包括 8 个数据位和 1 个确认位。传输定义为一个有效的开始和停止条件之间的时间，如图 18-3 中所述。每次传输的字节数不受限制；但是，每个数据字节后面必须紧跟一个确认位，并且数据必须以 MSB 优先的形式传输。当接收器无法完整接收另一个字节时，它可以将时钟线 SCL 保持为低电平，并强制发送器进入等待状态。这个过程通常称为时钟扩展。当接收器释放了时钟线 SCL 的时候，数据传输得以继续进行。

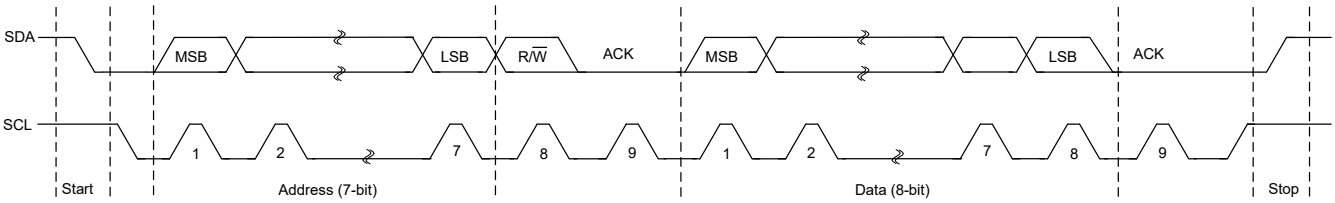


图 18-3. 模块数据传输

在时钟的高电平期间，SDA 线上的数据必须保持稳定，只有 SCL 为低电平时，数据线才能改变（请参阅图 18-4），否则会生成开始或停止条件。

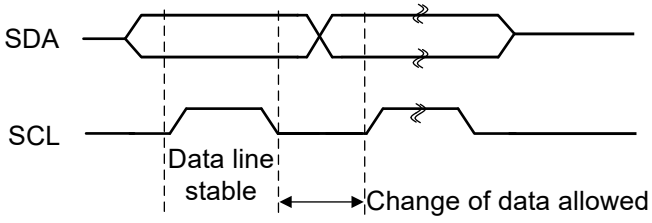


图 18-4. 数据更改周期

18.2.3.2 START 和 STOP 条件

I²C 总线协议定义了两种状态，以便开始和结束数据传输：START 和 STOP。当 SCL 为高电平时 SDA 线路上从高电平到低电平的跳变被定义为 START 条件，而当 SCL 为高电平时 SDA 线路上从低电平到高电平的跳变被定义为 STOP 条件。START 和 STOP 条件始终由控制器产生。总线在 START 条件之后被视为忙状态，在 STOP 条件之后被视为空闲(free)状态。

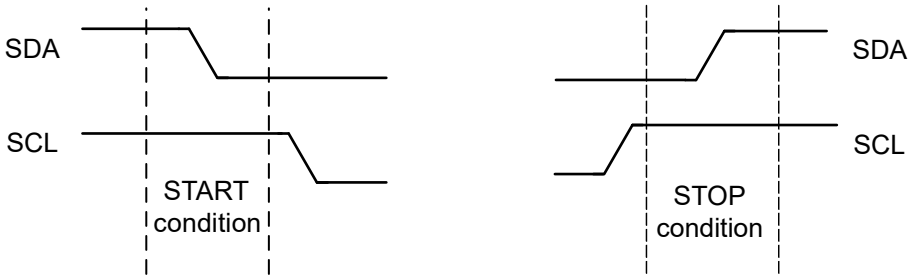


图 18-5. START 和 STOP 条件

STOP 位决定事务是在数据周期结束时停止还是继续直至重复的 START 条件。

为了生成单个事务，应向 I²C 控制器目标地址 I2Cx.MSA.SADDR 寄存器写入所需的地址，将 I2Cx.MSA.DIR 位设置为 1 启动接收操作，设置为 0 启动发送操作，并向控制寄存器 (I2Cx.MCTR) 写入 ACK = X（0 或 1）、STOP = 1、START = 1 和 RUN = 1 来执行操作并在结束时产生 STOP 条件。当操作完成（或由于错误而中止）时，设置中断标志。I²C 模块以控制器接收器模式运行时，通常会设置 ACK 位（在没有错误的情况下），从而使 I²C 总线控制器在每个字节接收完之后自动发送一个确认。当 I²C 总线控制器无需接收从目标发送器发送的更多数据时，必须清除该位。更多有关 I²C 控制器模式配置的信息，请参阅节 18.2.4.1

在目标模式下运行时，CPU_INT.RIS 寄存器中的 START 和 STOP 位可以指示是否检测到总线上的启动和停止条件，并且可以配置 CPU_INT.IMASK 以允许将 START 和 STOP 提升为控制器中断（启用中断后）。更多有关 I²C 目标模式配置的信息，请参阅节 18.2.4.2

总线状态标志：

- 在总线上检测到 START 条件时会设置 BUSBSY 位，在总线上检测到 STOP 条件时会按顺序清除该位。一旦 SCL 和 SDA 均拉至高电平，设置一个时钟超时 I2Cx.MSR.CLKTO 后，也会清除该位。
- 在 START/RESTART 后还会设置 BUSY 位，并且在传输 I2Cx.MCTR.MBLEN 个字节的数据后清除该位。在数据被 NACK 或检测到 STOP 后，也会清除该位。
- 当控制器 I²C 状态机处于 IDLE 状态（表示没有正在进行的传输）时会设置 IDLE 位。

18.2.3.3 带有 7 位地址的数据格式

数据传输遵循图 18-6 所示的格式。在开始条件之后，发送一个目标地址。该地址长度为 7 位，后跟第 8 位，即数据方向位（该位仅作为控制器模式，I2Cx.MSA 寄存器中的 DIR 位）。如果 I2Cx.MSA.DIR 位为 0，则表示发送操作（发送）；如果设置为 1，则表示请求接收数据（接收）。数据传输始终由控制器生成的停止条件终止；但是，控制器可在不先生成停止条件的情况下，通过生成重复的开始条件，并对另一个目标器件寻址，来启动与总线上另一个器件的通信，请参阅重复开始部分。因此，在一次传输过程中可能会存在各种不同组合的接收/发送格式。第九位是确认位，如节 18.2.3.4 所述。

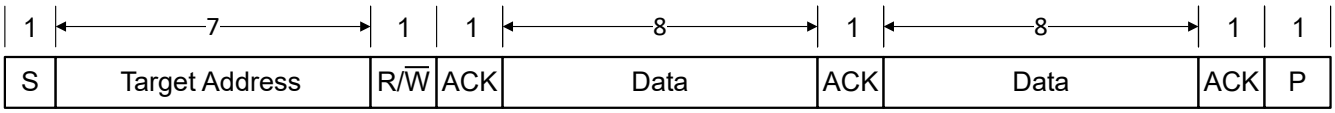


图 18-6. 带有 7 位地址的数据格式

可以通过 I2Cx.SCTR.GENCALL 位来启用 I²C 模块，以响应 I²C 总线上的通用广播。该通用广播由 0x00 地址标识，R/W 位设置为 0。可以通过 CPU_INT.IMASK.GENCALL 位来启用通用广播中断。

18.2.3.4 应答

所有总线事务都带有所需的确认时钟周期，该时钟周期由控制器产生。在确认周期中，发送器（可以是控制器或目标）会释放 SDA 线。为了响应传输，接收器必须在应答时钟周期过程中拉低 SDA。确认周期必须符合数据有效性要求。

当目标接收器没有确认目标地址时，SDA 必须由目标方保持为高电平，这样控制器可以产生一个停止条件并中止当前传输，或者产生一个重复的启动条件来启动一个新的传输。如果控制器器件在传输期间充当接收器，则它负责确认目标进行的每次传输。因为控制器控制传输中的字节数，所以它会通过不对最后一个数据字节产生确认来向目标发送器发出数据结束的信号。然后目标发送器必须释放 SDA 以允许控制器生成停止或重复的启动条件。

目标可以手动或自动生成 ACK/NACK。当 I2Cx.SACKCTL.ACKOEN=0 时，目标将发送自动 ACK。请注意，由于 FIFO，器件将自动接收并确认所有字节，直到 RX FIFO 已满。设置 SACKCTL.ACKOEN =1 会启用手动 ACK。当不需要自动 FIFO 接收时，可以使用手动 ACK 覆盖来评估每个接收到的字节或减慢通信速度。手动 ACK 覆盖操作启用后，在最后一个数据位之后，I²C 目标模块的时钟被拉低，直到该 SACKCTL.ACKOVAL 被写入指示的响应。新数据的接收由 SRXDONE 中断标志指示。

如果控制器在发送数据时接收到 NACK，则 RIS 寄存器中的 NACK 和 MTXDONE 位将置位。如果 FIFO 中仍有数据，TXEMPTY 位将不会置位，从而通知软件可能需要清空 TX FIFO。

18.2.3.5 重复开始

控制器可以通过发出一个重复的 START 条件，在不事先停止传输的情况下改变 SDA 上的数据流方向。这种条件称为 RESTART，请参阅图 18-7 中的数据格式。发出 RESTART 后，会再次以 R/W 位指定的新数据方向将目标地址再次发送出去。

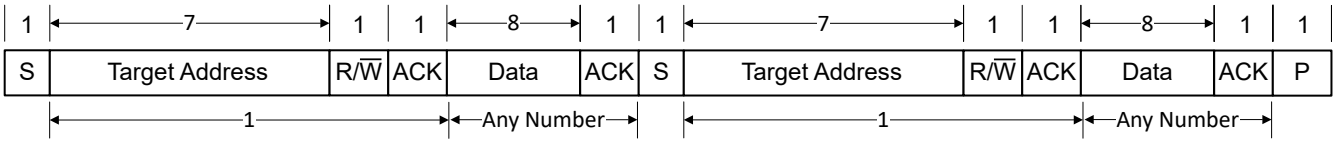


图 18-7. 重复启动条件下的数据格式

控制器发送操作的重复启动序列如下所述：

1. 当器件处于空闲状态时，控制器将目标地址写入 I2Cx.MSA 寄存器，并将 DIR 位配置为所需的传输类型。
2. 数据将写入 I2Cx.MTXDATA 寄存器。
3. 当 MSR 寄存器中的 BUSY 位为 0 时，需要设置 I2Cx.MCTR 寄存器中的 BURSTRUN 和 START 位才能启动传输。
4. 等待 I2Cx.MSR 寄存器中的 BUSY 位变为 0。
5. 控制器不会产生 STOP 条件，而是将另一个目标地址写入 I2Cx.MSA 寄存器，然后通过写操作再次设置 BURSTRUN 和 START 位以启动重复的 START 操作。

控制器接收操作的重复启动序列与上述序列类似：

1. 当器件处于空闲状态时，控制器将目标地址写入 I2Cx.MSA 寄存器，并将 DIR 位配置为所需的传输类型。
2. 当 I2Cx.MSR 寄存器中的 BUSY 位为 0 时，需要设置 I2Cx.MCTR 寄存器中的 BURSTRUN 和 START 位才能启动传输。
3. 控制器从 I2Cx.MRXDATA 寄存器读取数据。
4. 等待 I2Cx.MSR 寄存器中的 BUSY 位变为 0。
5. 控制器不会产生 STOP 条件，而是将另一个目标地址写入 I2Cx.MSA 寄存器，然后通过写操作再次设置 BURSTRUN 和 START 位以启动重复的 START 操作。

18.2.3.6 SCL 时钟低电平超时

I²C 目标可以将时钟周期性地拉低以减慢通信，从而扩展事务。I²C 模块有一个 12 位的可编程计数器，它可以跟踪时钟被拉低了多长时间。计数值的高 8 位可通过 I2CTIMEOUT_CTL 寄存器进行软件编程。I2CTIMEOUT_CTL.TCNTLA 寄存器中编程的 CNTL 值必须大于 0x01 才能启用超时功能。请注意，低电平超时配置只需在初始化期间（而非运行状态期间）设置。

应用程序可以对计数器的 8 个最高有效位进行编程，以反映事务中可接受的累积低周期。每个计数等于 (1 + TPR) × 12 个功能时钟 FCLK 的超时周期，其中 TPR 是可编程计时器周期。超时计数器 A 在 SCL 保持低电平的整个时

间内连续进行计数。当 SCL 为高电平时，超时计数器 A 重新加载 I2CTIMEOUT_CTL.TCNTLA 寄存器位中的值，并在 SCL 的下降沿从该值开始向下计数。

即使 SCL 在总线上保持低电平，为超时计数器生成的 BUSSCLK 时钟也能继续运行，与编程的 I²C 速度无关。

I²C 时钟低电平超时周期计算方法如下：

- 累计时钟低电平周期 = 超时计数器 * 一个超时周期
- 一个超时周期 = BUSSCLK 周期 * (1 + TPR) * 12
- 超时计数器 = I2CTIMEOUT_CTL.TCNTLA 寄存器（该寄存器包含一个 12 位计数器值的高 8 位，低 4 位设置为 0）

例如，如果 BUSSCLK 时钟为 20MHz 并且 I² C 模块以 100kHz 的速度运行，TPR 将等于 19。请参阅节 18.2.1.1，了解 TPR 的计算方法。一个超时周期等于 (1 / 20MHz) × (1 + 19) × 12，即 12µs。将 I2CTIMEOUT_CTL.TCNTLA 寄存器编程为 0xDA 将得到值 0xDA0，因为低 4 位设置为 0x0。换言之，也就是 3488 个时钟周期，即在 100kHz 频率下，累计时钟低电平周期为 3488 × 12µs，即 41.856ms。

当到达时钟超时期限时，I²C 控制器原始中断状态 (RIS) 寄存器中的 TIMEOUTA 位被设置，以便让控制器开始纠正措施，解决远程目标状态问题。此外，还会设置 I²C 控制器状态 MSR 寄存器中的 BUSBSY 位。在 I²C 变为空闲状态或 I²C 控制器复位期间，该位清零。软件可以通过 I²C 控制器总线监视 MBMON 寄存器中的 SDA 和 SCL 位读取 SDA 和 SCL 信号的原始状态，从而帮助确定远程目标的状态。

发生超时情况时，应用软件必须选择如何尝试恢复总线。如果在传输（接收或传输）结束之前检测到超时，软件应在初始化下一次传输之前刷新 FIFO。SMBus 和 PMBus 实现需要时钟低电平超时。

备注

控制器时钟低电平超时计数器寄存器 I2CTIMEOUT_CNT.TCNTA 在 SCL 持续保持低电平的整个时间内进行计数。当 SCL 为高电平时，计数器重新加载 I2CTIMEOUT_CTL.TCNTLA 寄存器中的值，并在 SCL 的下降沿从该值开始向下计数。

18.2.3.7 时钟扩展

在控制器模式下，如果总线上没有任何目标支持时钟延展，则可以禁用时钟延展，从而使控制器达到总线上的最大速度。否则，可通过将时钟保持在低电平的目标或 I²C 模块内的时钟状态检测延迟来减慢时钟。

为了确保符合 I²C 规范，需要启用时钟延展。当设置已满 RX FIFO 或空的 TX FIFO 时，时钟延展被激活。通过配置 I2Cx.MCR 寄存器中的 CLKSTRETCH 位，可以启用或禁用时钟延展支持。

在目标模式下，时钟延展始终启用，并由 I²C 目标状态寄存器 I2Cx.SSR 的 TREQ 和 RREQ 位发出信号。

- 设置 TREQ 位后，它会指示 I²C 控制器已经作为目标发送器被寻址，并且正在使用时钟延展来延迟控制器，直到数据被写入 STXDATA FIFO（目标 TX FIFO 为空）。
- 设置 RREQ 位后，它会指示 I²C 控制器有来自 I2C 控制器的未处理接收数据，并且正在使用时钟延展来延迟控制器，直到从 SRXDATA FIFO（目标 RX FIFO 已满）读取数据。

备注

目标模式下的时钟延展可与异步快速时钟请求一起使用，以便支持在检测到 I2C 起始位时使器件进入暂停低功耗模式状态，从而使 I2C 模块支持低功耗模式下的 100kHz（标准模式）或 400kHz（快速模式）操作，此时总线时钟速度低于相应模式所需的最小过采样速度。有关最低频率要求，请参阅时钟选择和 I²C 速度部分。如果时钟延展与异步快速时钟请求一起使用，则在 I²C 空闲时，器件可以在 STOP 或 STANDBY 模式下等待，在看到 I²C 总线边沿时，快速时钟请求将请求基频下的 SYSOSC，总线时钟将切换到基频下的 SYSOSC，允许 I²C 模块处理总线事务并在产生中断时唤醒处理器。

18.2.3.8 双地址

I²C 目标接口支持目标器件的双地址功能。一个额外的可编程 I²C 目标自身地址寄存器 I2CX。提供了 SOAR，如果启用，则可以匹配。当双地址功能禁用 (I2Cx.SOAR2.OAR2EN=0) 时，如果地址与 I2Cx.SOAR 寄存器中的 OAR 字段相匹配，则 I²C 目标在总线上提供 ACK。在双地址模式 (I2Cx.SOAR2.OAR2EN=1) 下，如果

I2Cx.SOAR 寄存器中的 OAR 字段或者 I2Cx.SOAR2 寄存器中的 OAR2 字段匹配，则 I²C 目标在总线上提供 ACK。

I2Cx.SSR 寄存器中的 OAR2SEL 位指示经过 ACK 的地址是否为备用地址。当该位清零时，表示主地址匹配或没有 OAR2 地址匹配。

18.2.3.9 仲裁

只有在总线空闲时，控制器才可以启动传输在启动条件的最短保持时间内，两个或两个以上的控制器可以生成一个启动条件。在这些情况下，当 SCL 为高电平时，SDA 线上产生仲裁机制（请参阅图 18-8）。产生逻辑高电平的第一个控制器发送器被产生逻辑低电平的另一个控制器控制，而丢失的控制器释放总线，直到总线再次空闲。

仲裁可以在多个位上发生。仲裁的第一个阶段是比较地址位，如果两个控制器都试图寻址同一个器件，仲裁将继续比较数据位

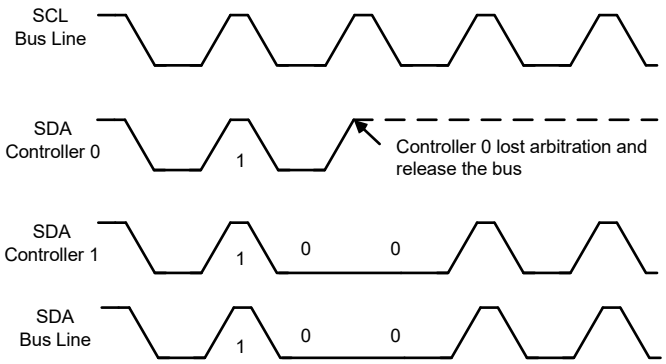


图 18-8. 两个控制器发送器之间的仲裁过程

当检测到仲裁丢失时，I2Cx.MSR.ARBLST 标志置位。在总线上检测到下一个停止条件时，该标志将由硬件复位。此外，会设置 CPU_INT.RIS 寄存器中的 ARBLOST 标志。

如果在 I²C 控制器启动传输时仲裁丢失，应用程序应执行以下步骤来正确处理仲裁丢失：

- 清空 TX FIFO
- TXEMPTY 位通过 IMASK 和 ICLR 寄存器清除和屏蔽 TX Empty 中断。
- 一旦总线空闲，TXFIFO 就可以被填满并启用，TXEMPTY 位就可以被解除屏蔽，并启动新事务。

18.2.3.10 多控制器模式

在多控制器系统中运行时，必须设置 I2Cx.MCR.MMST 位。

在仲裁过程中，必须同步来自不同控制器的时钟。在 SCL 上第一个产生低电平周期的器件会驳回其他器件，以便迫使其他器件起始其本地低电平周期。之后 SCL 被低电平周期最长的器件保持为低电平。其他器件必须首先等待 SCL 被释放，然后才能开始其高电平周期

在多控制器配置中，仲裁期间的时钟同步（图 18-9）处于启用状态，一旦检测到 SCL 线路为高电平，SCL 高电平时间就会计数。如果未启用，则只要 I²C 控制器将 SCL 线路设置为高电平，高电平时间就会计数，而该控制器允许 I²C 达到 I2Cx.MTPR 寄存器所指定速度的最高速度。

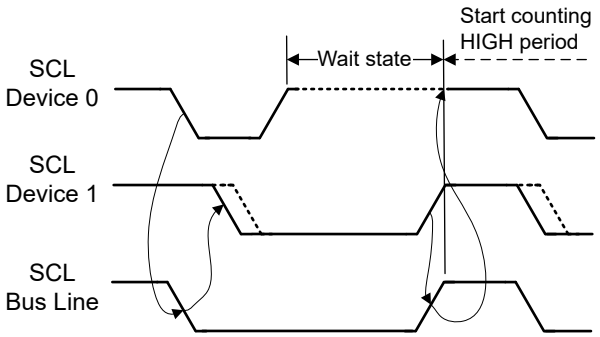


图 18-9. 时钟同步

18.2.3.11 干扰抑制

I²C 模块支持 SCL 和 SDA 线路上的干扰抑制，以满足 I²C 规范中规定的 50ns 干扰抑制要求。

模拟干扰滤波器

默认情况下，模拟干扰滤波器会启用并配置为抑制脉冲宽度高达 50ns 的尖峰。I²C 规范建议抑制小于 50ns 的噪声尖峰。用户可以通过清除 I2Cx.GFCTL.AGFEN 位来禁用此滤波器，也可以使用 I2Cx.GFCTL.AGFEN 来配置抑制脉冲宽度。模拟干扰滤波器只能用于在低功耗模式下唤醒 I²C。

数字干扰滤波器

可以对 I2Cx.GFCTL 寄存器中的 DGFSEL 位进行编程，以便为 SCL 和 SDA 线路提供干扰抑制确保信号值正确。干扰抑制值以 I²C 功能时钟为单位。当干扰抑制为非零时，所有信号都将在内部延迟。例如，如果 DGFSEL 设置为 0x7，则在计算预期事务时间时应加上 31 个时钟；更多信息请参阅寄存器描述。数字干扰滤波器不能用于将 I2C 从低功耗模式唤醒。

表 18-3. 干扰抑制滤波器

	模拟干扰滤波器	数字干扰滤波器
默认值	默认启用且宽度为 50ns	默认绕过
所抑制尖峰的脉冲宽度	除电流限制以外的 5ns、10ns、25ns、50ns	可编程的 I²C 时钟周期 1、2、3、4、8、16、31
优势	无需时钟即可使用	• 可编程长度以提供额外的滤波 • 稳定的滤波长度
限制	随温度、电压和工艺而变化	当没有足够的时钟时，在低功耗唤醒模式下不工作。 仅在 I²C 数据包开始 3 个时钟周期后启用

备注

只有控制器和目标的 I² 模块未启用时，才应修改寄存器 I2Cx.GFCTL 内的干扰滤波器配置。

18.2.3.12 FIFO 操作

接收数据寄存器 I2Cx.MRXDATA（用于控制器模式）和 I2Cx.SRXDATA（用于目标模式）是用户可访问的，包含要从 RX FIFO 栈读取的当前字符。最后接收到的来自接收移位寄存器的字符将推送到 FIFO 堆栈的末尾。

发送数据寄存器 I2Cx.MTXDATA（用于控制器模式）和 I2Cx.STXDATA（用于目标模式）是用户可访问的，并保存最后写入 TX FIFO 的数据。TX FIFO 包含等待移入发送移位寄存器并在 SDA 上发送的数据。

FIFO 可用于控制器和目标器件的接收和发送。每个 FIFO 条目的宽度为 8 位，并且应以字节模式访问。每个 FIFO 都有一个可编程的阈值点（在控制器模式下，通过 I2Cx.MFIFOCTL 寄存器中的 RXTRIG 和 TXTRIG 位配置；在目标模式下，通过 I2Cx.SFIFOCTL 寄存器中的 RXTRIG 和 TXTRIG 位配置），可以指示何时应生成 FIFO 服务中断。此外，可以在控制器和目标器件的中断屏蔽 (MASK) 寄存器中启用 FIFO 接收满中断和发送空中断。

通过将 I2Cx.FIFOCTL 寄存器中的 TXFLUSH 或 RXFLUSH 位设置为 1，可以清除 FIFO 的内容。当 I²C 复位时，也需要清除 FIFO 的内容。FIFO 清除只能在 I²C 处于 IDLE 模式时执行。触发清除之前，应禁用 FIFO 中断，并且清除完成后需要检查中断标志。

18.2.3.12.1 在目标模式下刷新过时的 Tx 数据

在大多数用例中，用户不希望在下一帧中传输前一帧中 I2C 外设 Tx FIFO 的剩余数据。该器件为软件提供了一种机制来选择是否刷新过时的数据。

状态位 SSR.STALE_TXFIFO 表示 I2C 外设 TX FIFO 中的数据是否过时。控制位 SCTR.TXWAIT_STALE_TXFIFO 用于启用对目标逻辑的修改空指示 - 当 Tx FIFO 为空或 Tx FIFO 中存在过时数据时，向 I2C 外设 FSM 指示空。控制位 SCTR.TXEMPTY_ON_TREQ 允许 RIS.STXEMPTY 中断用于指示 TREQ 条件，即 SCL 正在延展以等待来自 I2C 外设的传输数据。

建议的序列

1. 设置 SCTR.TXWAIT_STALE_TXFIFO 后，在停止、重启或超时时，I2C 外设 FSM 会得到空指示，即使 I2C 外设 Tx FIFO 中存在过时数据也是如此。
2. I2C 模块不会立即生成 TX FIFO 空中断/DMA 请求。相反，会等待 I2C 总线上的控制器尝试从 I2C 外设读取数据，此时 I2C 外设时钟延长。
3. 设置 SCTR.TXEMPTY_ON_TREQ 后，I2C 外设向 CPU 发出时钟延展 (TREQ) 中断。
4. ISR 中的应用软件会检查是否存在 SSR.STALE_TXFIFO 标志，然后使用 SFIFOCTL.TXFLUSH 刷新 TX FIFO，这也会清除 SSR.STALE_TXFIFO 的状态。

18.2.3.13 环回模式

通过设置 I²C 控制器配置 I2Cx.MCR 寄存器中的 LPBK 位，可以将 I²C 模块置于内部环回模式，以便开展诊断或调试工作。在环回模式下，来自 I²C 的控制器部分的 SDA 和 SCL 信号连接到 I²C 模块的目标部分的 SDA 和 SCL 信号，以便在无需连接 I/O 的情况下对器件进行内部测试。

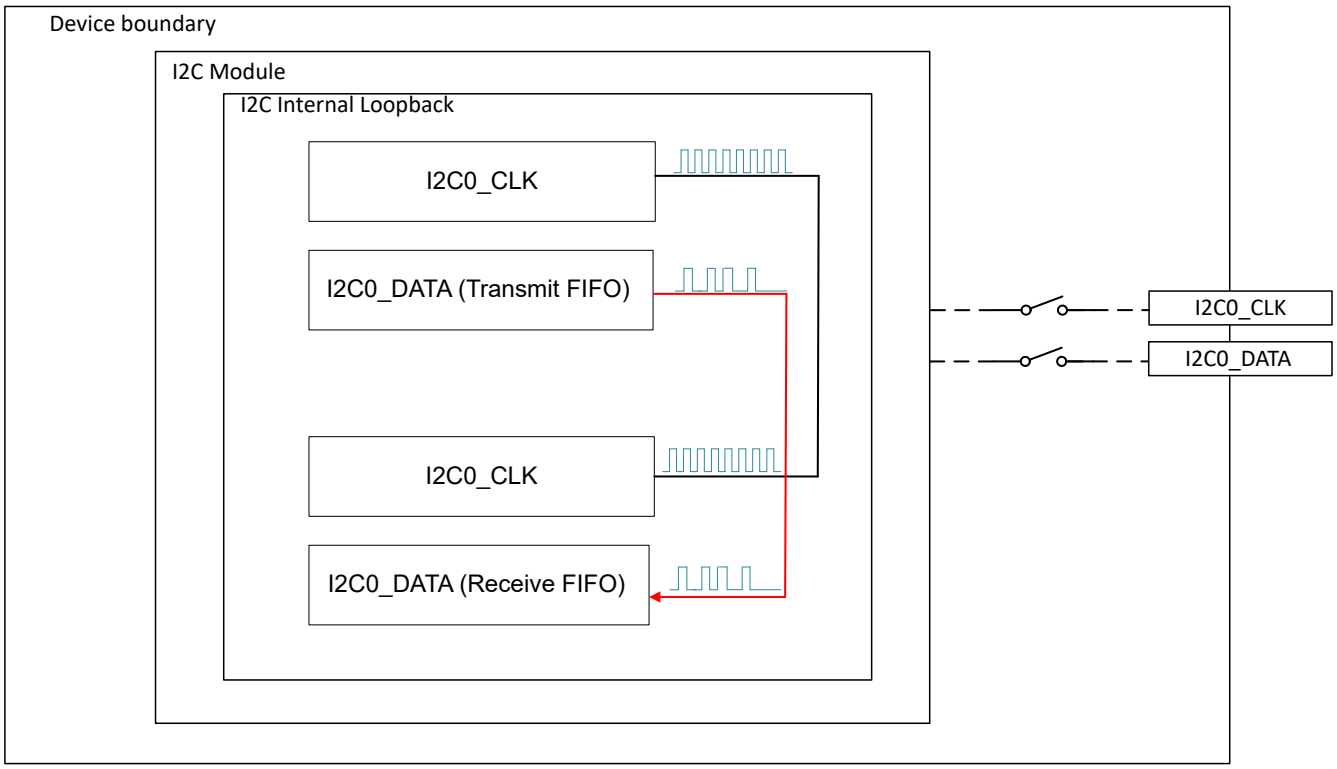


图 18-10. I2C 内部环回模式

18.2.3.14 突发模式

为控制器模块提供了突发模式，该模式允许使用 DMA 或软件处理 FIFO 中数据的数据传输序列。要启用突发模式，将控制器控制寄存器 I2Cx.MCTR 中的 MBLEN 位设置为大于 1 的值，这会设置突发传输的字节数。该值会自动写入 I²C 控制器状态寄存器 I2Cx.MSR 中的 MBCNT 位，以便在突发传输期间用作向下计数器。

写入 I²C FIFO 的字节将被传输到 RX FIFO 或 TX FIFO，具体取决于是执行发送还是接收。如果在突发期间数据被否定确认 (NACK) 且 I2Cx.MCTR 寄存器中的停止位被设置，则传输终止。如果停止位没有被置位，软件必须在 NACK 中断有效时发出停止或重复的启动条件。对于 NACK，I2Cx.MSR 寄存器中的 MBCNT 位可用于确定在突发终止之前传输的数据量。如果在传输过程中该地址被否定确认 (NACK)，则发出停止条件。

18.2.3.15 DMA 操作

I²C 提供了一个接口，以连接带有两个通道的 DMA 控制器。通过 I²C 事件寄存器和 DMA 外设寄存器启用 I²C 的 DMA 操作。启用 DMA 功能后，当相关的 FIFO 可以发送或接收数据时，I²C 会在所选通道上发出 DMA 请求。

有关 I²C 事件和 DMA 的更多信息，请参阅中断和事件支持和 DMA 触发发布者部分。

备注

每个 DMA 通道在同一时间只能由 IMASK 寄存器启用一个事件源。DMA 事务描述符需要与所选触发器匹配，并为控制器或目标以及 RX 或 TX 配置进行正确设置。仅当没有 I²C 传输正在进行且之前触发的 DMA 传输已完成时，才应更改 DMA 触发。如果无法确保这一点，则应首先禁用 I²C 和 DMA 通道。

18.2.3.16 低功耗操作

I²C 支持在低功耗模式下运行。

控制器模式支持的功耗模式：

- 速度为 100kHz 的控制器模式可在运行、睡眠、停止功耗模式下运行
- 速度为 400kHz 的控制器模式可在运行、睡眠功耗模式下运行
- 速度为 1MHz 的控制器模式可在运行、睡眠功耗模式下运行

在低功耗模式下，I²C 接口在检测到启动条件后，会自动请求在 CLKSEL 控制寄存器中选择的时钟。检测到停止条件后，会释放时钟请求。

目标模式支持的功耗模式和唤醒模式：

- 速度为 100kHz 的目标模式可在运行、睡眠、停止功耗模式下运行
- 速度为 100kHz 的目标模式无法在待机模式下运行，因为待机模式下的最高总线时钟为 32kHz，要支持 100kHz 的速度，需要最低 400kHz 的时钟。但是，在待机模式下，I2C 目标仍然可以从起始位唤醒并执行异步快速时钟请求，以临时获得 32MHz 时钟来接收数据，直到 FIFO 或地址匹配中断唤醒 CPU。
- 运行、睡眠功耗模式下速度为 400kHz 的目标模式
- 运行、睡眠功耗模式下速度为 1MHz 的目标模式

18.2.4 协议说明

18.2.4.1 I²C 控制器模式

如功能方框图部分所示，I²C 外设具有一组特定的控制器寄存器，用于配置模块配置为 I²C 目标模式时的操作。

18.2.4.1.1 控制器配置

I²C 控制器控制寄存器 I2Cx.MCTR 和 I²C 控制器目标地址寄存器 I2Cx.MSA 用于控制控制器发送和接收模式。可以修改以下设置来控制不同的事务。

- 长度表示事务的字节数，由 I2Cx.MCTR.MBLEN 位配置
- 方向（发送或接收）由 I2Cx.MSA.DIR 位配置
- ACK 生成由 I2Cx.MCTR.MBLEN 位配置
- STOP 条件生成由 I2Cx.MCTR.STOP 位配置

- START 或重复 START 条件生成由 I2Cx.MCTR.START 位配置
- RUN 启用控制器事务，由 I2Cx.MCTR.BURSTRUN 位配置

控制器发送数据事务

表 18-4. 从空闲模式开始发送

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	0	x	0 或 1	1	1	START+ADDR+R/W+ DATA*n+(STOP)	STOP 的发送取决于 STOP 位

表 18-5. 当最后一次发送完成而没有停止时，继续发送

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	0	x	0 或 1	0	1	DATA*n+ (ACK/NACK)+ (STOP)	STOP 的发送取决于 STOP 位

如果有来自目标的 NACK 响应，控制器将自动发送出停止以完成发送。控制器将无法在 ADDR 或 DATA NACK 之后发送 RESTART。

控制器接收数据事务

表 18-6. 从空闲模式开始接收

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	1	0 或 1	0 或 1	1	1	START+ADDR +R/ W+DATA *n +(ACK/NACK) +(STOP)	最后数据的 ACK 或 NACK 取决于 ACK 位；额外的 STOP 发送取决于 STOP 位

表 18-7. 当最后一次接收完成而没有 STOP 或 NACK 时，继续接收

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	1	0 或 1	0 或 1	0	1	DATA*n+ (ACK/NACK) + (STOP)	最后数据后跟 ACK 或 NACK 取决于 ACK 位；额外的 STOP 发送取决于 STOP 位

如果最后的事务以 NACK 结束，则不允许此配置，因为 NACK 只能后跟 STOP 或 RESTART。ACK 和 STOP 位不应同时设置为 1，因为需要先通知目标释放总线，然后才能发送出 STOP。

控制器重复开始事务

如果最后一次发送或接收完成而没有停止，则可以生成重复开始以启动新事务

表 18-8. 重复开始发送

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	0	0 或 1	0 或 1	1	1	Restart+ADDR +R/ W+DATA*n +(STOP)	额外的 STOP 发送取决于 STOP 位

如果有来自目标的 NACK 响应，控制器将自动发送出停止以完成发送。控制器将无法在 ADDR 或 DATA NACK 之后发送 RESTART。

表 18-9. 重复开始接收

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	1	0 或 1	0 或 1	1	1	Restart+ADDR +R/ W+DATA*n +(ACK/NACK) +(STOP)	最后数据后跟 ACK 或 NACK 取决于 ACK 位；额外的 STOP 发送取决于 STOP 位

ACK 和 STOP 位不应同时设置为 1，因为需要先通知目标释放总线，然后才能发送出 STOP。

控制器发送仅 STOP 事务

表 18-10. 发送仅 STOP

Length	方向	ACK	停止	启动	运行	Format	说明
n (n>0)	X	X	1	0	1	停止	STOP 命令

只有在前一个事务成功完成后才允许发送，如果控制器当前处于接收模式，则不能在没有 NACK 的情况下向目标发送 STOP

控制器快速命令事务

快速命令只能在事务开始时发送，而不能跟随其他事务（不停止）或重复开始。

表 18-11. 控制器快速命令

Length	方向	ACK	停止	启动	运行	Format	说明
0	0/1	X	1	1	1	START+ADDR +R/ W+STOP	快速命令

18.2.4.1.2 控制器模式操作

I²C 控制器初始化

1. 使用 IOMUX 寄存器配置 SDA 和 SCL 引脚功能并选择作为输入。
2. 使用 I2Cx.RSTCTL 寄存器复位外设。
3. 使用 I2Cx.PWREN 寄存器启用外设的电源。
4. 使用 CLKCTL 和 CLKDIV 寄存器选择和配置 I²C 时钟。
5. 通过向 I2Cx.MTPR 寄存器中的 TPR 位写入正确的值来设置所需的 SCL 时钟速度。有关如何计算 TPR 值的更多信息，请参阅时钟控制部分。例如，使用 20MHz I²C 时钟以实现 100kbps SCL 时钟时，TPR 值将为 19 (0x13)，因此我们可以向 I2Cx.MTPR 寄存器写入值 0x13。
6. 通过写入 I2Cx.MSA 寄存器，为下一个操作指定目标地址和模式（发送或接收）。例如，如果目标地址为 0x3B (MSA.SADDR[7:1] = 0x3B)，并且我们希望发送 (MSA.DIR[0] = 0x0) 数据，那么我们可以向 I2Cx.MSA 寄存器写入值 0x76。
7. 如果控制器正在发送数据，用户可以通过将所需数据写入 I2Cx.MTXDATA 寄存器，将要发送的数据（字节）放入数据寄存器中。
8. 通过写入 I2Cx.MCTR 寄存器来配置控制器发送或接收模式。有关如何为不同模式配置 I2Cx.MCTR 寄存器的更多信息，请参阅上面的控制器配置部分。
9. 使用 CPU_INT.IMASK 寄存器启用所需的中断和/或 DMA 事件。

I²C 控制器状态

用户可以通过读取 I2Cx.MSR 寄存器来检查 I²C 控制器的当前状态。

表 18-12. 控制器状态寄存器

位域	说明
BUSY	I2C 控制器忙。在事务正在进行期间，设置 BUSY 位。
ERR	I2C 错误。该错误可能源于目标地址未得到确认或者发送数据未得到确认。

表 18-12. 控制器状态寄存器 (continued)

位域	说明
ADRACK	确认地址。如果发送的地址未得到确认，则设置该位。
DATACK	确认数据。如果发送的数据未得到确认，则设置该位。
ARBLST	仲裁丢失。如果控制器丢失了仲裁，则设置该位。
IDLE	I2C 总线空闲。
BUSBSY	I2C 总线忙。该位根据 START 和 STOP 条件发生变化，将其设置为总线忙。
CLKTO	时钟超时错误。如果发生时钟超时错误，则设置该位。
MBCNT	I2C 控制器事务计数。此字段包含事务的当前倒计时值。

I²C 控制器接收器模式

为了让控制器脱离空闲模式开始接收数据，用户需要设置 I2Cx.MCTR 寄存器中的 START 位来生成开始条件。然后，一旦控制器检测到总线空闲，就会自动发送 START 条件，后跟目标地址。应遵循以下所有过程。

I2Cx.MSA.DIR 设置为 1 以启用接收模式，I2Cx.MCTR.START 设置为生成开始条件，可以对 I2Cx.MBLEN 进行编程以指示接收操作的字节数 (n)。I2Cx.MCTR.ACK 和 I2Cx.MCTR.STOP 位可以根据用户配置设置或清零。设置 I2Cx.MCTR.BURSTRUN 以开始操作。数据包格式为 START+ADDR+R+DATA*n +(ACK/NACK) + (STOP)。最后数据的 ACK/NACK 取决于 ACK 位，额外的 STOP 发送取决于 STOP 位。

接收到最后一个字节后，设置 CPU_INT.IIDX 寄存器中的 MRXDONE (0x01) 中断，以指示控制器接收事务已完成。用户可以使用 CPU_INT.IIDX 寄存器中的 MRXFIFOTRG (0x03) 中断从接收 FIFO 中读取数据。当控制器 RX FIFO 包含的内容 >= 定义的字节时将触发该中断，触发电平可以通过使用 I2Cx.MFIFOCTL 寄存器中的 RXTRIG 位来定义。控制器接收器模式的流程图如图 18-11 所示。

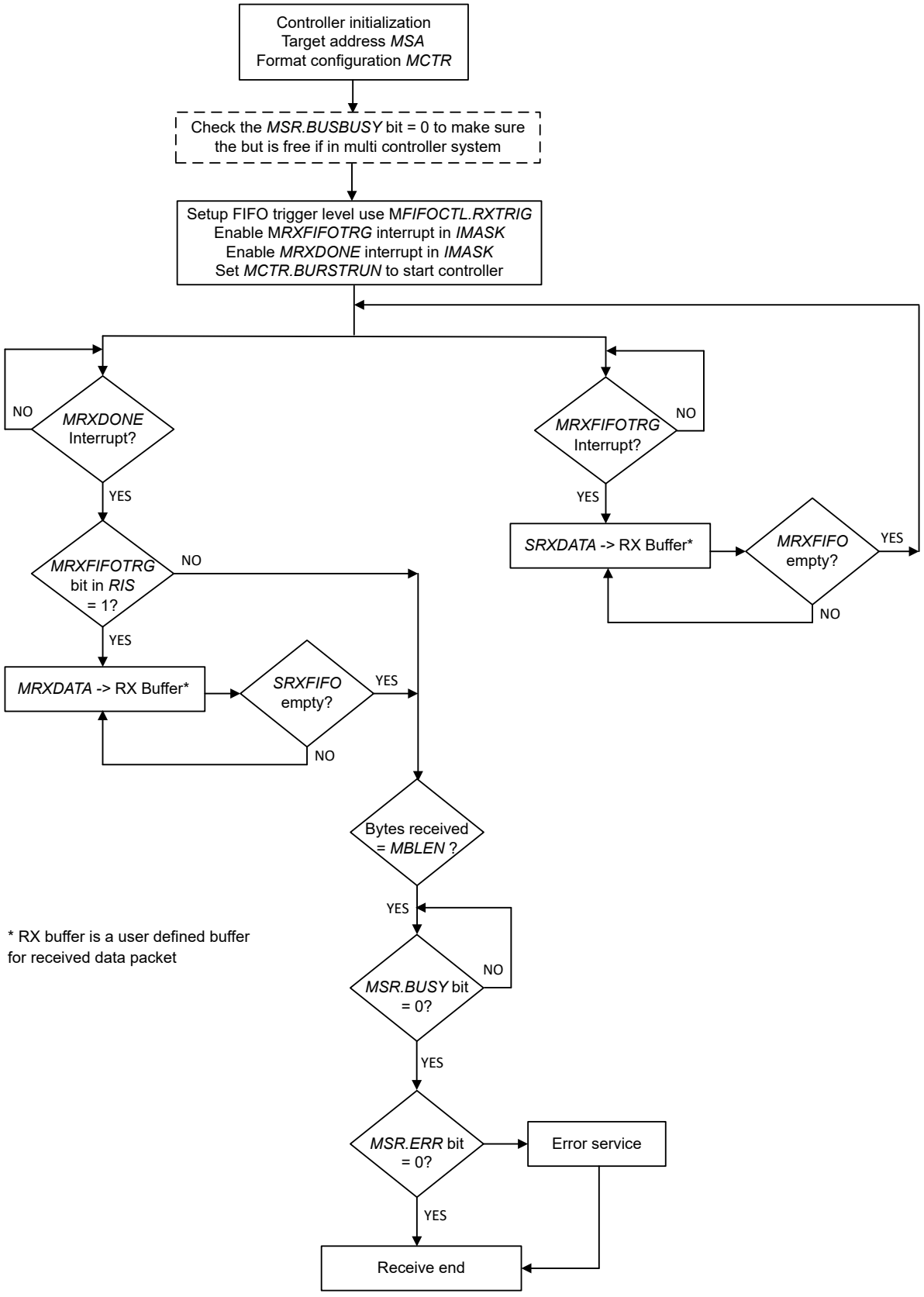


图 18-11. 控制器接收器模式

I²C 控制器发送器模式

为了让控制器脱离空闲模式开始发送数据，用户需要设置 I2Cx.MCTR 寄存器中的 START 位来生成开始条件。然后，一旦控制器检测到总线空闲，就会自动发送 START 条件，后跟目标地址。如果在目标地址发送过程中没有丢失仲裁，则将发送写入 MTXFIFO 的数据。应遵循以下所有过程。

I2Cx.MSA.DIR 清零以启用发送模式，I2Cx.MCTR.START 设置为生成开始条件，可以对 I2Cx.MBLEN 进行编程以指示发送操作的字节数 (n)。I2Cx.MCTR.STOP 位可根据用户配置设置或清零。设置 I2Cx.MCTR.BURSTRUN 以开始操作。数据包格式为 START+ADDR+W+DATA*n + (STOP)，STOP 的发送取决于 STOP 位。

发送完最后一个字节后，设置 CPU_INT.IIDX 寄存器中的 MTXDONE (0x02) 中断，以指示控制器发送事务已完成。用户还可以设置 CPU_INT.IIDX 寄存器中的 MTXEMPTY (0x06) 中断，以查看 MTXFIFO 是否为空且已准备好加载更多数据。如果发送 FIFO 中的所有数据都已移出，则将触发此中断。控制器接收器模式的流程图如图 18-12 所示。

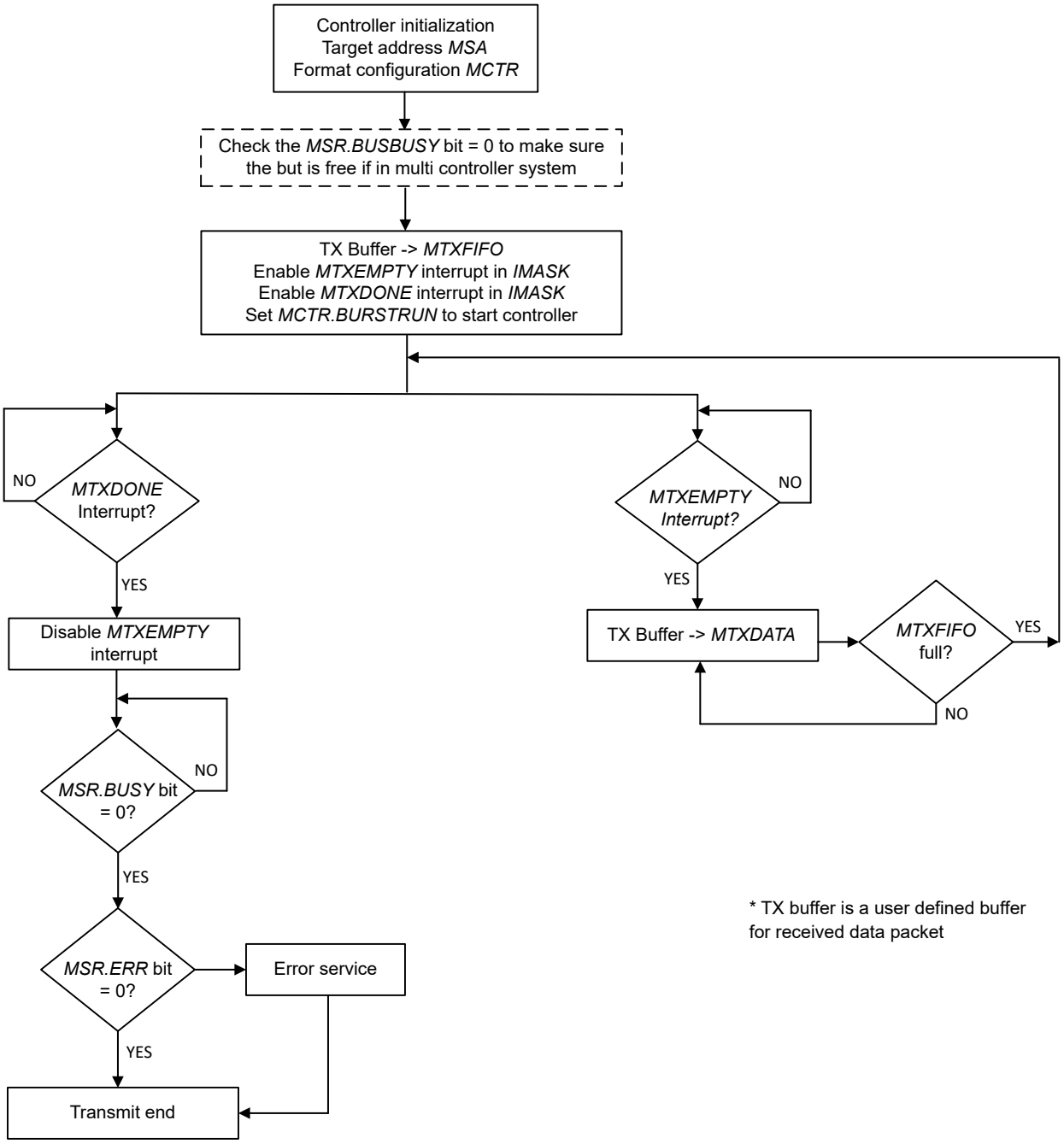


图 18-12. 控制器发送器模式

18.2.4.1.3 TX 为空时读取

通常，在任何 I2C 事务中，都需要先写入目标以设置寄存器编号，然后执行重复的启动 + 读取以返回数据值。但是，通过使用 RD_ON_EMPTY 标志，设置一次 I2C 控制器即可执行两个 I2C 事务。用户可以同时设置写入和读取，其限制是“写入”阶段发送的数据不能超过 TX FIFO 中可容纳的数据。这是一种尽可能降低中断处理要求的优化手段。大致设置如下：

- 使用 RECEIVE = 1 将 MSA 设置为目标地址（RD_ON_TXEMPTY 必须设置 I2C_MSR_DIR_RECEIVE）
- 设置 MCTR 来启动事务，在本用例中写入：
  - MBLEN= READ 阶段的大小
  - RD_ON_TXEMPTY

- ACK_DISABLE（我的用例在读取后将 NACK+STOP）
- STOP_ENABLE
- START_ENABLE
- BURST_ENABLE

此设置的预期行为是控制器将执行 START，发送所有 TXFIFO 数据，当 TXFIFO 为空时，自动执行一次重复的 START，读取 MBLEN 字节，然后 STOP。

18.2.4.2 I²C 目标模式

18.2.4.2.1 目标模式运行

I²C 目标初始化

1. 使用 IOMUX 寄存器配置 SDA 和 SCL 引脚功能并选择作为输入。
2. 使用 I2Cx.RSTCTL 寄存器复位外设。
3. 使用 I2Cx.PWREN 寄存器启用外设的电源。
4. 使用 CLKCTL 和 CLKDIV 寄存器选择和配置 I²C 时钟。
5. 通过将 7 位地址写入 I2Cx.SOAR 寄存器来配置至少一个目标地址。可以使用 I2Cx.SOAR2 寄存器启用和配置额外的目标地址。
6. 使用 CPU_INT.IMASK 寄存器启用所需的中断和/或 DMA 事件。
7. 可以通过设置 I2Cx.SCTR 寄存器中的 GENCALL 位来启用通用广播响应。
8. 通过设置 I2Cx.SCTR 寄存器中的 ACTIVE 位来启用 I²C 目标模式。

I²C 目标状态

用户可以通过读取 I2Cx.SSR 寄存器来检查 I²C 目标的当前状态。

表 18-13. 目标状态寄存器

位域	说明
RREQ	如果 I2C 控制器具有来自 I2C 控制器的未处理的接收数据，并且正在使用时钟拉伸技术来延迟控制器直到从 SRXDATA FIFO 读取数据（目标 RX FIFO 已满），则会设置该位。
TREQ	如果 I2C 控制器已被寻址为目标发送器，并且正在使用时钟拉伸技术来延迟控制器直到数据被写入 STXDATA FIFO（目标 TX FIFO 为空），则会设置该位。
OAR2SEL	如果 SOAR2.OAR2 地址匹配并被目标确认 (ACK)，则会设置该位。
QCMDST	快速命令状态值。如果最后一个事务是正常事务或没有发生事务，则该位为 0。如果最后一个事务是快速命令事务，则会设置该位。
QCMDRW	快速命令读取/写入状态。仅当设置了 QCMDST 位时，该位才有意义。如果快速命令是写入，则该位为 0。如果快速命令是读取，则会设置该位。

I²C 目标接收器模式

当控制器发送的目标地址与其自己的地址相匹配并且接收到已清除的 R/W 位时，将进入目标接收器模式。在目标接收器模式下，会使用控制器器件生成的时钟脉冲将 SDA 上接收的串行数据位移入。目标器件不会产生时钟，但是当一个字节接收完毕后需要 CPU 的干预时，它可以保持 SCL 为低电平。

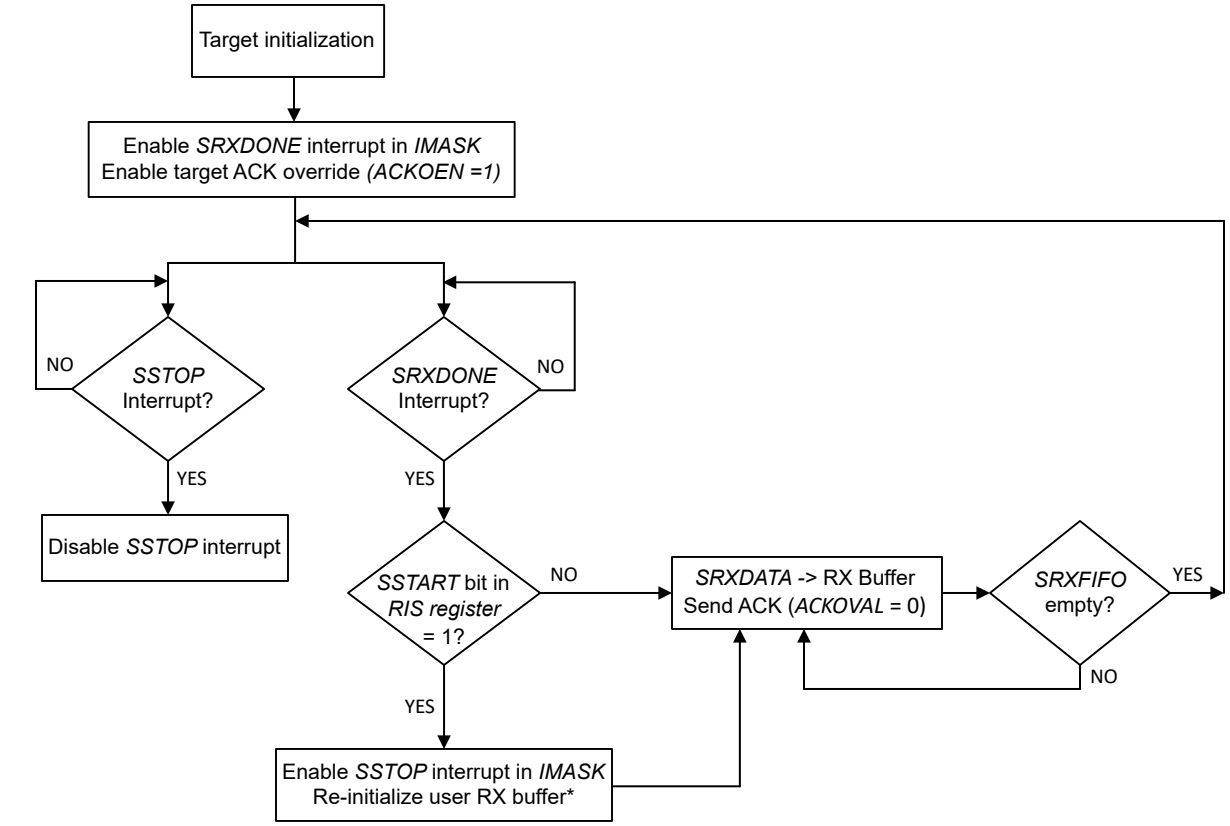
在收到一个数据字节后，会设置 CPU_INT.IIDX 寄存器中的 SRXDONE (0x11) 中断以指示已接收到一个字节。I²C 模块会自动确认接收到的数据，用户也可以选择手动配置 I2Cx.SACKCTL 寄存器，以便在接收到每个字节后发送确认。

当控制器产生启动条件时，设置 CPU_INT.IIDX 寄存器中的 SSTART (0x17) 中断。当控制器产生停止条件时，设置 CPU_INT.IIDX 寄存器中的 SSTOP (0x18) 中断。

用户也可以设置使用 CPU_INT.IIDX 寄存器中的 SRXFIFOTRG (0x13) 中断从接收 FIFO 中读取数据。当接收 FIFO 包含的内容 >= 定义的字节时将触发该中断，触发电平可以通过使用 I2Cx.SFIFOCTL 寄存器中的 RXTRIG 位来定义。

如果目标希望减慢通信速度以评估每个字节的接收情况，则可以使用 SRXDONE 方法，而 SRXFIFOTRG 方法可用于尽可能提高吞吐量并避免时钟拉伸。

使用 SRXDONE 和 SRXFIFOTRG 中断来读取接收数据的流程图分别如图 18-13 和图 18-14 所示。



* RX buffer is a user defined buffer for received data packet

图 18-13. 使用 SRXDONE 和 ACK 覆盖的目标接收器模式

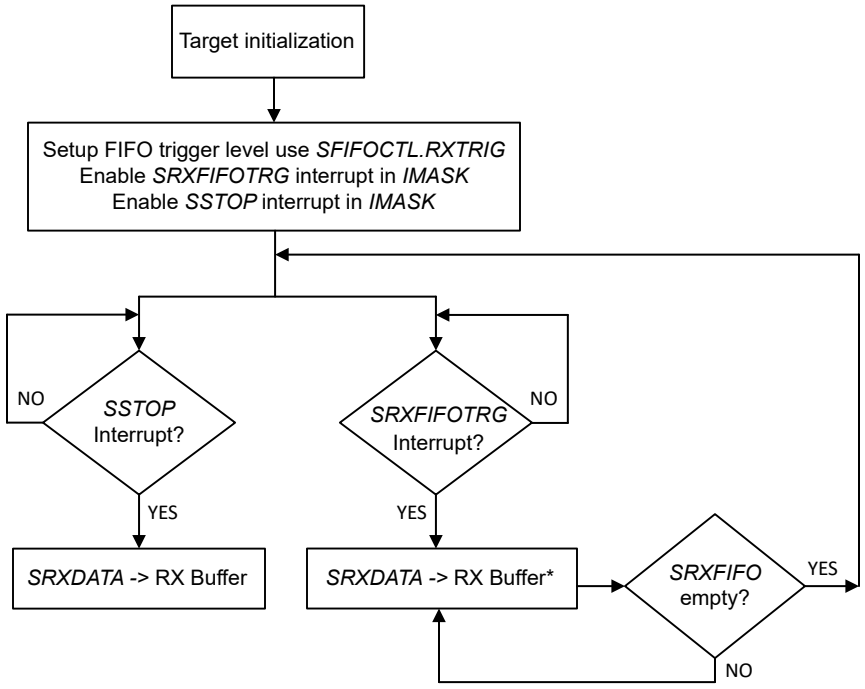


图 18-14. 使用 SRXFIFOTRG 和自动 ACK 的目标接收器模式

I²C 目标发送器模式

当控制器发送的目标地址与其自身设置了 R/W 位的地址相同时，将进入目标发送器模式。目标发送器使用控制器器件生成的时钟脉冲将 SDA 上的串行数据移出。目标器件不会产生时钟，但是当一个字节发送完毕后需要 CPU 的干预时，它可以保持 SCL 为低电平。

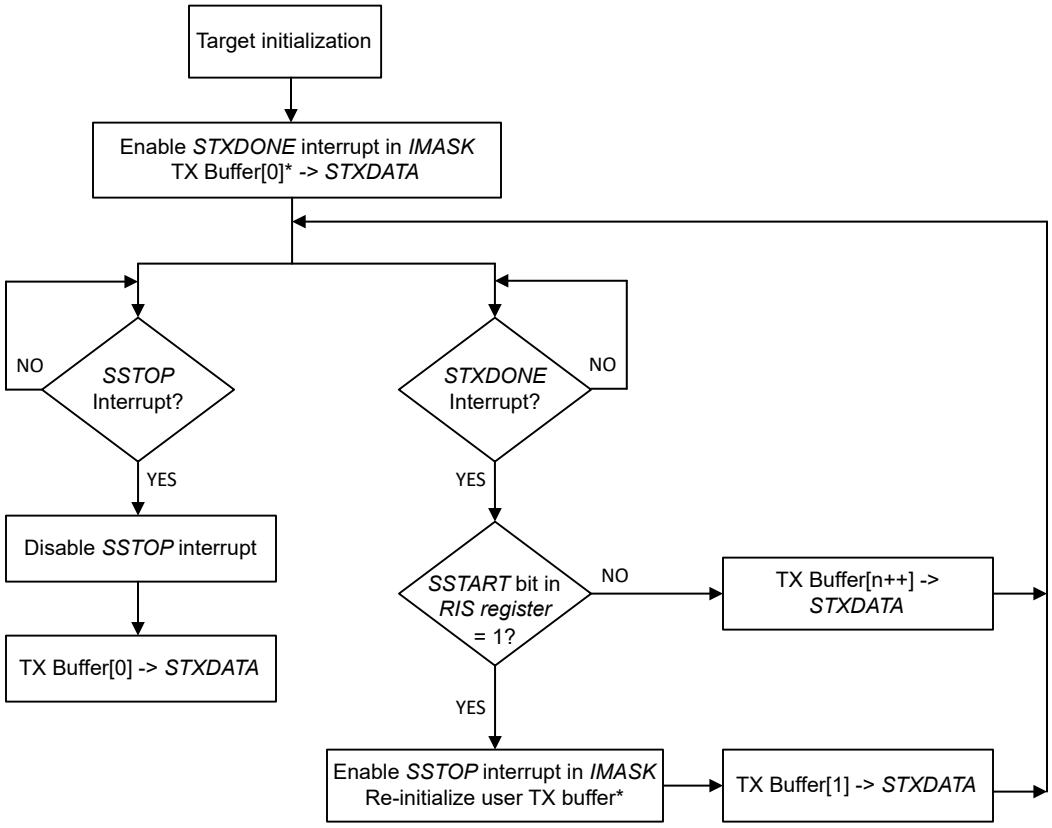
在发送一个数据字节后，会设置 CPU_INT.IIDX 寄存器中的 STXDONE (0x12) 中断，指示已发送一个字节。

当控制器产生启动条件时，设置 CPU_INT.IIDX 寄存器中的 SSTART (0x17) 中断。当控制器产生停止条件时，设置 CPU_INT.IIDX 寄存器中的 SSTOP (0x18) 中断。

用户也可以设置 CPU_INT.IIDX 寄存器中的 STXFIFOTRG (0x14) 中断，将数据加载到发送 FIFO 中。当发送 FIFO 包含的内容 <= 定义的字节时将触发该中断，触发电平可以通过使用 I2Cx.SFIFOCTL 寄存器中的 TXTRIG 位来定义。

如果目标希望减慢通信速度以评估每个字节的接收情况，则可以使用 STXDONE 方法，而 STXFIFOTRG 方法可用于尽可能提高吞吐量并避免时钟拉伸。

使用 STXDONE 和 STXFIFOTRG 中断来发送数据的流程图如图 18-15 和图 18-16 所示。



* TX buffer is a user defined buffer
for transmit data packet

图 18-15. 使用 STXDONE 的目标发送器模式

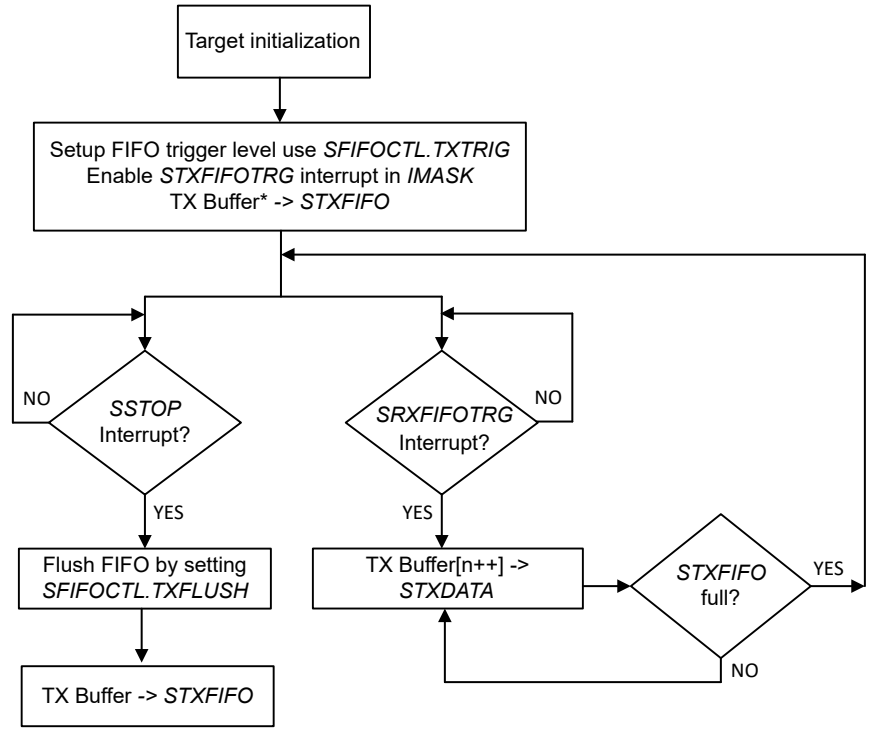


图 18-16. 使用 STXFIFOTRG 的目标发送器模式

18.2.5 复位注意事项

软件复位注意事项

可以通过同时设置 I2Cx.RSTCTL 寄存器中的 RESETASSERT 位和 KEY 位来执行软件复位。我们建议仅在终止事务后进行复位。

硬件复位注意事项

硬件复位也会初始化 IO 配置。此过程会将 IO 设置为高阻抗状态，并通过 I²C 的外部上拉电阻将线路上拉至高电平。

表 18-14 展示了禁用控制器或目标后状态位的行为。

表 18-14. 禁用控制器后的状态位

寄存器	位	禁用控制器后的行为	禁用目标后的行为	启用控制器/目标后的行为
I2Cx.MSR	Busy	复位状态	不用考虑	发送启动条件时进行更新
	ERR	复位状态	不用考虑	检测到下一个事件时进行更新
	ADRACK	复位状态	不用考虑	检测到下一个事件时进行更新
	DATACK	复位状态	不用考虑	检测到下一个事件时进行更新
	ARBLST	复位状态	不用考虑	检测到下一个事件时进行更新
	IDLE（闲置）	复位状态	不用考虑	检测到下一个事件时进行更新
	BUSBSY	复位状态	不用考虑	在总线上检测到下一次启动（或 SDA/SCL 为低电平）时进行更新
	CLKTO	复位状态	不用考虑	检测到下一个事件时进行更新
	MBCNT	复位状态	不用考虑	检测到下一个事件时进行更新
I2Cx.MCLKOCNT	CLKCNT	复位状态	不用考虑	当 SCL 为高电平时通过控制器使能进行更新

表 18-14. 禁用控制器后的状态位 (continued)

寄存器	位	禁用控制器后的行为	禁用目标后的行为	启用控制器/目标后的行为
I2Cx.MBMON	SCL	复位状态	不用考虑	通过控制器使能进行更新
	SDA	复位状态	不用考虑	通过控制器使能进行更新

表 18-15. 禁用目标后的状态位

寄存器	位	禁用控制器后的行为	禁用目标后的行为	启用控制器/目标后的行为
I2Cx.SSR	RREQ	不用考虑	复位状态	检测到下一个事件时进行更新
	TREQ	不用考虑	复位状态	检测到下一个事件时进行更新
	OAR2SEL	不用考虑	复位状态	检测到下一个事件时进行更新
	QCMDST	不用考虑	复位状态	检测到下一个事件时进行更新
	QCMDRW	不用考虑	复位状态	检测到下一个事件时进行更新
I2Cx.SFIFOSR	RXFIFOCNT	不用考虑	未更改	在 FIFO 访问时进行更新
	TXFIFOCNT	不用考虑	未更改	在 FIFO 访问时进行更新

18.2.6 初始化

有关控制器初始化，请参阅节 18.2.4.1.2；有关目标初始化，请参阅节 18.2.4.2.1。

备注

当 I2C 数据传输正在进行时，应用软件不得修改 I2C 模块的配置寄存器（包括时钟配置、模式配置和超时计数）。

18.2.7 中断和事件支持

I²C 模块包含三个事件发布者而没有事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理对 CPU 子系统的 I²C 中断请求 (IRQ)。第二个和第三个事件发布者（DMA_TRIG1、DMA_TRIG0）用于通过 DMA 事件路由设置 DMA 的触发信号。

表 18-16 中总结了 I²C 事件。

表 18-16. I2C 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	I²C	CPU 子系统	静态路由	CPU_INT 寄存器	来自 I²C 的固定中断路由 CPU 的中断信号
DMA 触发	发布者	I²C	DMA	DMA 事件路由	DMA_TRIG1 寄存器	来自 I²C 的固定中断路由 连接至 DMA
DMA 触发	发布者	I²C	DMA	DMA 事件路由	DMA_TRIG0 寄存器	来自 I²C 的固定中断路由 连接至 DMA

18.2.7.1 CPU 中断事件发布者 (CPU_INT)

I²C 模块提供 24 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，I²C 的 CPU 中断事件为：

表 18-17. 控制器的 I²C CPU 中断事件条件 (CPU_INT)

IIDX STAT	名称	说明
0x01	MRXDONE	控制器接收传输完成中断
0x02	MTXDONE	控制器发送传输事务完成中断

表 18-17. 控制器的 I²C CPU 中断事件条件 (CPU_INT) (continued)

IIDX STAT	名称	说明
0x03	MRXFIFOTRG	控制器接收 FIFO 触发信号。当 RX FIFO 包含 >= 定义的字节时触发
0x04	MTXFIFOTRG	控制器发送 FIFO 触发信号。当发送 FIFO 包含 <= 定义的字节时触发
0x05	MRXFIFOFULL	控制器 RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。
0x06	MTXEMPTY	控制器发送 FIFO 为空中断。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。
0x07	MCLKTO	控制器时钟超时中断
0x08	MnACK	地址/数据 NACK 中断
0x09	MSTART	控制器 START 检测中断
0x0A	MSTOP	控制器 STOP 检测中断
0x0B	MARBLOST	控制器仲裁丢失中断
0x0C	MDMA_DONE_TX	控制器 DMA TX 完成信号（有关详细信息，请参阅下一节）
0x0D	MDMA_DONE_RX	控制器 DMA RX 完成信号（有关详细信息，请参阅下一节）

表 18-18. 目标的 I²C CPU 中断事件条件 (CPU_INT)

IIDX STAT	名称	说明
0x11	SRXDONE	目标接收事务完成中断
0x12	STXDONE	目标传输事务完成中断
0x13	SRXFIFOTRG	目标器件接收 FIFO 触发信号。它会在接收 FIFO 包含 >= 定义的字节时触发
0x14	STXFIFOTRG	目标器件发送 FIFO 触发信号。它会在发送 FIFO 包含 <= 定义的字节时触发
0x15	SXFIFOFULL	目标 RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。
0x16	STXEMPTY	目标发送 FIFO 为空中断。如果目标器件发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。
0x17	SSTART	目标 START 检测中断
0x18	SSTOP	目标 STOP 检测中断
0x19	SGENCALL	常规调用中断
0x1A	SDMA_DONE_TX	目标器件 DMA TX 完成信号（有关详细信息，请参阅下一节）
0x1B	SDMA_DONE_RX	目标器件 DMA RX 完成信号（有关详细信息，请参阅下一节）

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

18.2.7.2 DMA 触发发布者（DMA_TRIG1、DMA_TRIG0）

DMA_TRIG1 和 DMA_TRIG0 寄存器用于设置 DMA 的触发信号。这可以通过灵活的方式设置，以触发控制器或目标器件的 DMA，并使用以下四个触发条件接收或发送事件：

表 18-19. I²C DMA 触发条件（DMA_TRIG1 和 DMA_TRIG0）

IIDX STAT	名称	说明
0x01	MRXFIFOTRG	控制器接收 FIFO 触发信号。当 RX FIFO 包含 >= 定义的字节时触发
0x02	MTXFIFOTRG	控制器发送 FIFO 触发信号。当发送 FIFO 包含 <= 定义的字节时触发
0x03	SRXFIFOTRG	目标器件接收 FIFO 触发信号。当 RX FIFO 包含 >= 定义的字节时触发
0x04	STXFIFOTRG	目标器件发送 FIFO 触发信号。当发送 FIFO 包含 <= 定义的字节时触发

通过 DMA_TRIG1 和 DMA_TRIG0 事件管理寄存器来管理 DMA 触发事件配置。有关配置事件寄存器的指导，请参阅节 7.2.5；有关 DMA 触发事件的工作原理，请参阅节 7.1.3.2。DMA_TRIG1 和 DMA_TRIG0 是两个事件管理寄存器，对应于两个 DMA 通道。

如图 18-17 所示，每个 DMA 通道都可以由表 18-19 中列出的任何条件触发，并且可以生成控制器 DMA 完成信号或目标器件 DMA 完成信号。

例如，用户可以使用 MTXFIFOTRG 配置 DMA_TRIG1 触发器，使用 SRXFIFOTRG 配置 DMA_TRIG0 触发器。当通道 1 DMA 状态更改为完成时，将设置 MDMA_DONE_TX 和 MDMA_DONE_RX 中断，当通道 2 DMA 状态更改为完成时，将设置 SDMA_DONE_TX 和 SDMA_DONE_RX 中断。

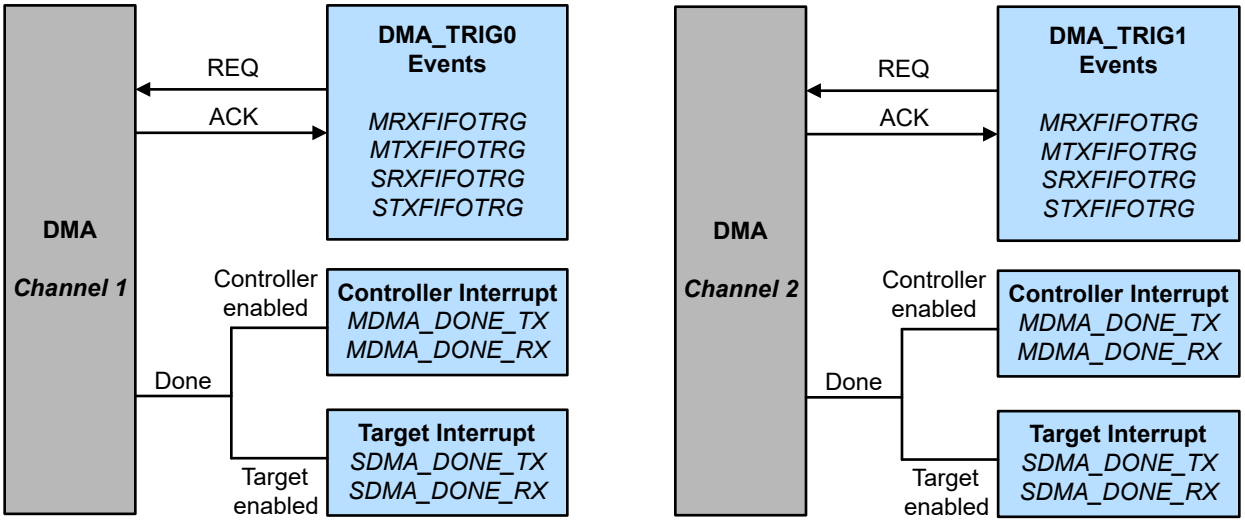


图 18-17. I²C DMA 触发和状态

18.2.8 仿真模式

器件处于调试模式时的模块行为由 PDBGCTL 寄存器中的 FREE 和 SOFT 位控制。

当器件处于调试模式并设置为停机模式时，可配置以下行为。

表 18-20. 调试模式外设行为

PDBGCTL.FREE	PDBGCTL.SOFT	功能
1	x	模块继续运行
0	0	模块立即停止
0	1	模块在下一次传输完成后停止

18.3 I2C 寄存器

表 18-21 列出了 I2C 寄存器的存储器映射寄存器。表 18-21 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 18-21. I2C 寄存器

偏移	缩写	寄存器名称	组	部分
4h	SCL	SCL 引脚配置		转到
8h	SDA	SDA 引脚配置		转到
204h	SCL	SCL 的 FUPDATE 版本		转到
208h	SDA	SDA 的 FUPDATE 版本		转到
480h	CPU_CONNECT_0	CPU 连接		转到
504h	DMA_MAP_TX	DMA 映射		转到
505h	DMA_TRIG_TX	DMA 触发器		转到
506h	DMA_ENTRY_TX	DMA 条目		转到
508h	DMA_MAP_RX	DMA 映射		转到
509h	DMA_TRIG_RX	DMA 触发器		转到
50Ah	DMA_ENTRY_RX	DMA 条目		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
808h	CLKCFG	外设时钟配置寄存器		转到
814h	STAT	状态寄存器		转到
1000h	CLKDIV	时钟分频器		转到
1004h	CLKSEL	超低功耗外设的时钟选择		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	DMA_TRIG1	转到
1058h	IMASK	中断屏蔽	DMA_TRIG1	转到
1060h	RIS	原始中断状态	DMA_TRIG1	转到
1068h	MIS	已屏蔽中断状态	DMA_TRIG1	转到
1070h	ISET	中断设置	DMA_TRIG1	转到
1078h	ICLR	中断清除	DMA_TRIG1	转到
1080h	IIDX	中断索引	DMA_TRIG0	转到
1088h	IMASK	中断屏蔽	DMA_TRIG0	转到
1090h	RIS	原始中断状态	DMA_TRIG0	转到
1098h	MIS	已屏蔽中断状态	DMA_TRIG0	转到
10A0h	ISET	中断设置	DMA_TRIG0	转到
10A8h	ICLR	中断清除	DMA_TRIG0	转到
10E0h	EVT_MODE	事件模式		转到
10E4h	INTCTL	中断控制寄存器		转到
10FCh	DESC	模块说明		转到
1200h	GFCTL	I2C 干扰滤波器控制		转到

表 18-21. I2C 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
1204h	TIMEOUT_CTL	I2C 超时计数控制寄存器		转到
1208h	TIMEOUT_CNT	I2C 超时计数寄存器		转到
1210h	MSA	I2C 控制器目标地址寄存器		转到
1214h	MCTR	I2C 控制器控制寄存器		转到
1218h	MSR	I2C 控制器状态寄存器		转到
121Ch	MRXDATA	I2C 控制器 RX 数据		转到
1220h	MTXDATA	I2C 控制器 TX 数据		转到
1224h	MTPR	I2C 控制器计时器周期		转到
1228h	MCR	I2C 控制器配置		转到
1234h	MBMON	I2C 控制器总线监控器		转到
1238h	MFIFOCTL	I2C 控制器 FIFO 控制		转到
123Ch	MFIFOSR	I2C 控制器 FIFO 状态寄存器		转到
1240h	CONTROLLER_I2CPECCTL	I2C 控制器 PEC 控制寄存器		转到
1244h	CONTROLLER_PECSR	I2C 控制器 PEC 状态寄存器		转到
1250h	SOAR	I2C 目标自身地址		转到
1254h	SOAR2	I2C 目标自身地址 2		转到
1258h	SCTR	I2C 目标控制寄存器		转到
125Ch	SSR	I2C 目标状态寄存器		转到
1260h	SRXDATA	I2C 目标 RX 数据		转到
1264h	STXDATA	I2C 目标 TX 数据		转到
1268h	SACKCTL	I2C 目标 ACK 控制		转到
126Ch	SFIFOCTL	I2C 目标 FIFO 控制		转到
1270h	SFIFOSR	I2C 目标 FIFO 状态寄存器		转到
1274h	TARGET_PECCTL	I2C 目标 PEC 控制寄存器		转到
1278h	TARGET_PECSR	I2C 目标 PEC 状态寄存器		转到
1E00h	TEST0	测试 0 寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 18-22 展示了适用于此部分中访问类型的代码。

表 18-22. I2C 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
R-0	R -0	读取 返回 0
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

18.3.1 SCL（偏移 = 4h）[复位 = 00000000h]

图 18-18 展示了 SCL，表 18-23 中对此进行了介绍。

返回到汇总表。

I2C SCL 引脚 PINCM 配置

图 18-18. SCL

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 18-23. SCL 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 18-23. SCL 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

18.3.2 SDA（偏移 = 8h）[复位 = 00000000h]

图 18-19 展示了 SDA，表 18-24 中对此进行了介绍。

返回到汇总表。

I2C SDA 引脚 PINCM 配置

图 18-19. SDA

31	30	29	28	27	26	25	24
RESERVED	GFLT	SLEW	WCOMP	WUEN	INV	HIGHZ1	HIGHZ0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED	DRV			HYSTEN	INENA	PIPU	PIPD
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
GSTATE		RESERVED					
R/W-0h		R/W-0h					
7	6	5	4	3	2	1	0
PSTATE		RESERVED					
R/W-0h		R/W-0h					

表 18-24. SDA 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	GFLT	R/W	0h	干扰滤波器使能 0h = 无内部干扰滤波器 1h = 使用内部干扰滤波器
29	SLEW	R/W	0h	被保留压摆率控制 0h = 无压摆率控制 1h = 使用压摆率控制
28	WCOMP	R/W	0h	唤醒比较值 0h = 在匹配为 0 时唤醒 1h = 在匹配为 1 时唤醒
27	WUEN	R/W	0h	唤醒使能 0h = 唤醒未启用 1h = 唤醒被启用
26	INV	R/W	0h	相对于外设/GPIO 的反相数字输入/输出 0h = 输入和输出非反相 1h = 输入和输出反相
25	HIGHZ1	R/W	0h	高阻态而非高电平输出 0h = 引脚可被驱动为高电平 1h = 引脚为三态而不是被驱动为高电平
24	HIGHZ0	R/W	0h	高阻态而非低电平输出 0h = 引脚可被驱动为低电平 1h = 引脚为三态而不是被驱动为低电平
23	RESERVED	R/W	0h	

表 18-24. SDA 字段说明 (continued)

位	字段	类型	复位	说明
22-20	DRV	R/W	0h	驱动强度选项 0h = 最低驱动强度 1h = 驱动强度 2/8 2h = 驱动强度 3/8 3h = 驱动强度 4/8 4h = 驱动强度 5/8 5h = 驱动强度 6/8 6h = 驱动强度 7/8 7h = 最高驱动强度
19	HYSTEN	R/W	0h	迟滞使能 0h = 无迟滞 1h = 迟滞打开
18	INENA	R/W	0h	输入使能 0h = 连接内核的输入 0 1h = 连接内核的输入 IO 焊盘值
17	PIPU	R/W	0h	上拉使能 0h = 无上拉 1h = 上拉
16	PIPD	R/W	0h	下拉使能 0h = 无下拉 1h = 下拉
15-14	GSTATE	R/W	0h	GPIO 通道状态 0h = G 通道处于未分配状态 1h = G 通道处于切换状态 2h = G 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = G 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
13-8	RESERVED	R/W	0h	
7-6	PSTATE	R/W	0h	外设模拟通道状态 0h = P 通道处于未分配状态 1h = P 通道处于切换状态 2h = P 通道处于连接状态且未锁定（即，允许更改 F 字段，而不返回未分配状态） 3h = P 通道处于连接状态并且锁定（即，在 G 和 P 通道都变为未分配状态之前，不允许将 F 字段更改为不同的非零值）
5-0	保留	R/W	0h	

18.3.3 SCL（偏移 = 204h）[复位 = 00000000h]

图 18-20 展示了 SCL，表 18-25 中对此进行了介绍。

返回到汇总表。

SCL 的 FUPDATE 版本

图 18-20. SCL

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

表 18-25. SCL 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

18.3.4 SDA（偏移 = 208h）[复位 = 00000000h]

图 18-21 展示了 SDA，表 18-26 中对此进行了介绍。

返回到汇总表。

SDA 的 FUPDATE 版本

图 18-21. SDA

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

表 18-26. SDA 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	W	0h	
27-2	IOADDR	W	0h	IO 地址。这是与引脚多路复用子区域的“完全写入”子区域中模块 IP 实例特定 IO 信号的 SOC 地址 [27:2] 相对应的地址。
1	LOCK	W	0h	设置锁定位 0h = 写入此值无效 1h = 设置通道锁定位
0	GSEL	W	0h	GPIO 通道选择 0：为 F 更新 1 选择 P 通道：为 F 更新选择 G 通道 0h = 为 F 更新选择 P 通道 1h = 为 F 更新选择 G 通道

18.3.5 CPU_CONNECT_0（偏移 = 480h）[复位 = 00000000h]

图 18-22 展示了 CPU_CONNECT_0，表 18-27 中对此进行了介绍。

返回到汇总表。

将外设发布者端口直接连接到应用处理器

图 18-22. CPU_CONNECT_0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						CPUSS0_CONN	RESERVED
R/W-0h						R/W-0h	R/W-0h

表 18-27. CPU_CONNECT_0 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	CPUSS0_CONN	R/W	0h	CPUSS0 连接位。 0h = CPU 未连接。 1h = CPU 已连接。
0	RESERVED	R/W	0h	

18.3.6 DMA_MAP_TX（偏移 = 504h）[复位 = 00h]

图 18-23 展示了 DMA_MAP_TX，表 18-28 中对此进行了介绍。

返回到汇总表。

DMA 中与此外设触发对应的触发端口 ID

图 18-23. DMA_MAP_TX

7	6	5	4	3	2	1	0
RESERVED	TRIG_ID						
R-0h	R-0h						

表 18-28. DMA_MAP_TX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R	0h	
6-0	TRIG_ID	R	0h	DMA 中与此外设触发对应的触发端口 ID 0h = 未选择触发器 1h = 已选择触发器 1 7Fh = 已选择触发器 127

18.3.7 DMA_TRIG_TX (偏移 = 505h) [复位 = 00h]

图 18-24 展示了 DMA_TRIG_TX，表 18-29 中对此进行了介绍。

返回到汇总表。

此外设触发的触发控制和状态寄存器

图 18-24. DMA_TRIG_TX

7	6	5	4	3	2	1	0
RESERVED	THRHLD			TRIGLOST	STATECLR	状态	
R/W-0h	R-0h			R-0h	R-0/W-0h	R-0h	

表 18-29. DMA_TRIG_TX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R/W	0h	
6-4	THRHLD	R	0h	DMA 接受触发请求的阈值。 0h = 可能的最低阈值 6h = 可能的最高阈值
3	TRIGLOST	R	0h	当触发端口处于 TRIGGER_PEND 或 TRIGGERED 时，只要接收到触发请求，就会设置粘滞标志。通过写入 STATECLR 来清除。 0h = 触发器未丢失 1h = 触发器已丢失
2	STATECLR	R-0/W	0h	清除触发状态。向该寄存器写入 1 会清除该端口上任何挂起的 DMA 触发器并将端口转换为未触发状态。 0h = 写入 0 不起作用 1h = 清除 DMA 触发器
1-0	状态	R	0h	返回 DMA tx 触发端口的当前状态 0h = 通道未触发 1h = 通道触发挂起 2h = 通道已触发

18.3.8 DMA_ENTRY_TX (偏移 = 506h) [复位 = 0FFFh]

图 18-25 展示了 DMA_ENTRY_TX，表 18-30 中对此进行了介绍。

返回到汇总表。

描述符连接到外设 DMA 触发器

图 18-25. DMA_ENTRY_TX

15	14	13	12	11	10	9	8
保留				ENTRY_ID			
R/W-				R/W-FFFh			
7	6	5	4	3	2	1	0
ENTRY_ID							
R/W-FFFh							

表 18-30. DMA_ENTRY_TX 字段说明

位	字段	类型	复位	说明
15-12	保留	R/W	0h	
11-0	ENTRY_ID	R/W	FFFh	此触发器被路由到的 DMA 描述符的 ID。这样可以确保另一个 DMA 通道不能监听或影响负责处理此外设数据的 DMA 通道。 0h = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 Fh = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 10h = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFEh = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFFh = 触发未启用

18.3.9 DMA_MAP_RX (偏移 = 508h) [复位 = 00h]

图 18-26 展示了 DMA_MAP_RX，表 18-31 中对此进行了介绍。

返回到汇总表。

DMA 中与此外设触发对应的触发端口 ID

图 18-26. DMA_MAP_RX

7	6	5	4	3	2	1	0
RESERVED	TRIG_ID						
R-0h	R-0h						

表 18-31. DMA_MAP_RX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R	0h	
6-0	TRIG_ID	R	0h	DMA 中与此外设触发对应的触发端口 ID 0h = 未选择触发器 1h = 已选择触发器 1 7Fh = 已选择触发器 127

18.3.10 DMA_TRIG_RX（偏移 = 509h）[复位 = 00h]

图 18-27 展示了 DMA_TRIG_RX，表 18-32 中对此进行了介绍。

返回到汇总表。

此外设触发的触发控制和状态寄存器

图 18-27. DMA_TRIG_RX

7	6	5	4	3	2	1	0
RESERVED	THRHLD			TRIGLOST	STATECLR	状态	
R/W-0h	R-0h			R-0h	R-0/W-0h	R-0h	

表 18-32. DMA_TRIG_RX 字段说明

位	字段	类型	复位	说明
7	RESERVED	R/W	0h	
6-4	THRHLD	R	0h	DMA 接受触发请求的阈值。 0h = 可能的最低阈值 6h = 可能的最高阈值
3	TRIGLOST	R	0h	当触发端口处于 TRIGGER_PEND 或 TRIGGERED 时，只要接收到触发请求，就会设置粘滞标志。通过写入 STATECLR 来清除。 0h = 触发器未丢失 1h = 触发器已丢失
2	STATECLR	R-0/W	0h	清除触发状态。向该寄存器写入 1 会清除该端口上任何挂起的 DMA 触发器并将端口转换为未触发状态。 0h = 写入 0 不起作用 1h = 清除 DMA 触发器
1-0	状态	R	0h	返回 DMA tx 触发端口的当前状态 0h = 通道未触发 1h = 通道触发挂起 2h = 通道已触发

18.3.11 DMA_ENTRY_RX（偏移 = 50Ah）[复位 = 0FFFh]

图 18-28 展示了 DMA_ENTRY_RX，表 18-33 中对此进行了介绍。

返回到汇总表。

描述符连接到外设 DMA 触发器

图 18-28. DMA_ENTRY_RX

15	14	13	12	11	10	9	8
保留				ENTRY_ID			
R/W-				R/W-FFFh			
7	6	5	4	3	2	1	0
ENTRY_ID							
R/W-FFFh							

表 18-33. DMA_ENTRY_RX 字段说明

位	字段	类型	复位	说明
15-12	保留	R/W	0h	
11-0	ENTRY_ID	R/W	FFFh	此触发器被路由到的 DMA 描述符的 ID。这样可以确保另一个 DMA 通道不能监听或影响负责处理此外设数据的 DMA 通道。 0h = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 Fh = DCLB 索引 i=0-15。这只能与专用的 DCLB 一起使用。 10h = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFEh = RACE 存储器中索引 i=16-4094 处的 DMA 条目。只有在系统中启用了无限 DMA 时，才能使用此字段。 FFFh = 触发未启用

18.3.12 PWREN（偏移 = 800h）[复位 = 00000000h]

图 18-29 展示了 PWREN，表 18-34 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 18-29. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 18-34. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

18.3.13 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 18-30 展示了 RSTCTL，表 18-35 中对此进行了介绍。

返回到汇总表。

用于控制复位置为有效和无效的寄存器

图 18-30. RSTCTL

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

表 18-35. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	使外设复位有效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位置为有效

18.3.14 CLKCFG（偏移 = 808h）[复位 = 00000000h]

图 18-31 展示了 CLKCFG，表 18-36 中对此进行了介绍。

返回到汇总表。

外设时钟配置寄存器

图 18-31. CLKCFG

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							BLOCKASYNC
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 18-36. CLKCFG 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许状态更改的 KEY - 0xA9 A9h = 允许更改 GPRCM 字段的密钥值
23-9	保留	R/W	0h	
8	BLOCKASYNC	R/W	0h	阻止异步时钟请求启动 SYSOSC 或强制总线时钟为 32MHz 0h = 不阻止异步时钟请求 1h = 阻止异步时钟请求
7-0	保留	R/W	0h	

18.3.15 STAT（偏移 = 814h）[复位 = 00000000h]

图 18-32 展示了 STAT，表 18-37 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 18-32. STAT

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

表 18-37. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

18.3.16 CLKDIV（偏移 = 1000h）[复位 = 00000000h]

图 18-33 展示了 CLKDIV，表 18-38 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块专用分频比

图 18-33. CLKDIV

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

表 18-38. CLKDIV 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 3 分频 3h = 对时钟源进行 4 分频 4h = 对时钟源进行 5 分频 5h = 对时钟源进行 6 分频 6h = 对时钟源进行 7 分频 7h = 对时钟源进行 8 分频

18.3.17 CLKSEL（偏移 = 1004h）[复位 = 00000000h]

图 18-34 展示了 CLKSEL，表 18-39 中对此进行了介绍。

返回到汇总表。

时钟源选择。

图 18-34. CLKSEL

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

表 18-39. CLKSEL 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	BUSCLK_SEL	R/W	0h	如果启用，选择 BUSCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
2	MFCLK_SEL	R/W	0h	如果启用，选择 MFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
1-0	保留	R/W	0h	

18.3.18 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 18-35 展示了 PDBGCTL，表 18-40 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 18-35. PDBGCTL

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

表 18-40. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设将阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

18.3.19 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 18-36 展示了 IIDX，表 18-41 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。

图 18-36. IIDX

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

表 18-41. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	I2C 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 RIS 和 MISC 中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 01h = 已接收控制器数据 02h = 已发送控制器数据 03h = 控制器接收 FIFO 触发电平 04h = 控制器发送 FIFO 触发电平 5h = RX FIFO 满事件/中断挂起 6h = 发送 FIFO/缓冲器空事件/中断挂起 08h = 地址/数据 NACK 09h = 启动事件 0Ah = 停止事件 0Bh = 仲裁失败 Ch = 通道 TX 上 DMA 完成 Dh = 通道 RX 上 DMA 完成 Eh = 控制器 PEC 接收错误事件 Fh = 超时 A 事件 10h = 超时 B 事件 11h = 目标数据事件 12h = 目标数据事件 13h = 目标接收 FIFO 触发电平 14h = 目标发送 FIFO 触发电平 15h = RX FIFO 已满事件/中断挂起 16h = 发送 FIFO/缓冲器空事件/中断挂起 17h = 启动事件 18h = 停止事件 19h = 常规调用事件 1Ah = 通道 TX 上 DMA 完成 1Bh = 通道 RX 上 DMA 完成 1Ch = 目标 PEC 接收错误事件 1Dh = 目标 TX FIFO 下溢 1Eh = 目标 RX FIFO 上溢事件 1Fh = 目标仲裁失败事件 20h = 中断上溢事件

18.3.20 IMASK（偏移 = 1028h）[复位= 00000000h]

图 18-37 展示了 IMASK，表 18-42 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 18-37. IMASK

31	30	29	28	27	26	25	24
INTR_OVFL	SARBLOST	SRX_OVFL	STX_UNFL	SPEC_RX_ERR	SDMA_DONE_RX	SDMA_DONE_TX	SGENCALL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
SSTOP	SSTART	STXEMPTY	SRXFIFOFULL	STXFIFOTRG	SRXFIFOTRG	STXDONE	SRXDONE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
TIMEOUTB	TIMEOUTA	MPEC_RX_ERR	MDMA_DONE_RX	MDMA_DONE_TX	MARBLOST	MSTOP	MSTART
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
MnACK	RESERVED	MTXEMPTY	MRXFIFOFULL	MTXFIFOTRG	MRXFIFOTRG	MTXDONE	MRXDONE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 18-42. IMASK 字段说明

位	字段	类型	复位	说明
31	INTR_OVFL	R/W	0h	中断上溢中断屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
30	SARBLOST	R/W	0h	目标仲裁失败 0h = 清除设置的中断屏蔽 1h = 设置中断屏蔽
29	SRX_OVFL	R/W	0h	目标 RX FIFO 上溢 0h = 清除中断屏蔽 1h = 设置中断屏蔽
28	STX_UNFL	R/W	0h	目标 TX FIFO 下溢 0h = 清除中断屏蔽 1h = 设置中断屏蔽
27	SPEC_RX_ERR	R/W	0h	目标 RX PEC 错误中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
26	SDMA_DONE_RX	R/W	0h	事件通道 RX 上目标 DMA 完成 0h = 清除中断屏蔽 1h = 设置中断屏蔽
25	SDMA_DONE_TX	R/W	0h	事件通道 TX 上目标 DMA 完成 0h = 清除中断屏蔽 1h = 设置中断屏蔽
24	SGENCALL	R/W	0h	通用广播中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
23	SSTOP	R/W	0h	停止条件中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 18-42. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
22	SSTART	R/W	0h	启动条件中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
21	STXEMPTY	R/W	0h	目标发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
20	SRXFIFOFULL	R/W	0h	RXFIFO 满事件。如果目标 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
19	STXFIFOTRG	R/W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
18	SRXFIFOTRG	R/W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
17	STXDONE	R/W	0h	目标发送事务完成中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
16	SRXDONE	R/W	0h	目标接收数据中断 表示已接收到一个字节的信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
15	TIMEOUTB	R/W	0h	超时 B 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
14	TIMEOUTA	R/W	0h	超时 A 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
13	MPEC_RX_ERR	R/W	0h	控制器 RX PEC 错误中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
12	MDMA_DONE_RX	R/W	0h	事件通道 RX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
11	MDMA_DONE_TX	R/W	0h	事件通道 TX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
10	MARBLOST	R/W	0h	仲裁失败中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9	MSTOP	R/W	0h	STOP 检测中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	MSTART	R/W	0h	START 检测中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	MnACK	R/W	0h	地址/数据 NACK 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	RESERVED	R/W	0h	

表 18-42. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
5	MTXEMPTY	R/W	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	MRXFIFOFULL	R/W	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	MTXFIFOTRG	R/W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	MRXFIFOTRG	R/W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXDONE	R/W	0h	控制器发送事务完成中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXDONE	R/W	0h	控制器接收事务完成中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.21 RIS（偏移 = 1030h）[复位 = 00000000h]

图 18-38 展示了 RIS，表 18-43 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 18-38. RIS

31	30	29	28	27	26	25	24
INTR_OVFL	SARBLOST	SRX_OVFL	STX_UNFL	SPEC_RX_ERR	SDMA_DONE_RX	SDMA_DONE_TX	SGENCALL
R-	R-0 h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
SSTOP	SSTART	STXEMPTY	SRXFIFOFULL	STXFIFOTRG	SRXFIFOTRG	STXDONE	SRXDONE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
TIMEOUTB	TIMEOUTA	MPEC_RX_ERR	MDMA_DONE_RX	MDMA_DONE_TX	MARBLOST	MSTOP	MSTART
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
MnACK	RESERVED	MTXEMPTY	MRXFIFOFULL	MTXFIFOTRG	MRXFIFOTRG	MTXDONE	MRXDONE
R-0h	R-	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 18-43. RIS 字段说明

位	字段	类型	复位	说明
31	INTR_OVFL	R	0h	中断上溢中断 当 SSTART 或 SSTOP 中断上溢发生两次而未得到处理时设置此位 0h = 未发生中断 1h = 已发生中断
30	SARBLOST	R	0h	目标仲裁失败 0h = 未发生中断 1h = 已发生中断
29	SRX_OVFL	R	0h	目标 RX FIFO 上溢 0h = 未发生中断 1h = 已发生中断
28	STX_UNFL	R	0h	目标 TX FIFO 下溢 0h = 未发生中断 1h = 已发生中断
27	SPEC_RX_ERR	R	0h	目标 RX PEC 错误中断 0h = 未发生中断 1h = 已发生中断
26	SDMA_DONE_RX	R	0h	事件通道 RX 上 DMA 完成 0h = 清除中断 1h = 设置中断
25	SDMA_DONE_TX	R	0h	事件通道 TX 上 DMA 完成 0h = 清除中断 1h = 设置中断
24	SGENCALL	R	0h	通用广播中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 18-43. RIS 字段说明 (continued)

位	字段	类型	复位	说明
23	SSTOP	R	0h	停止条件中断 0h = 清除中断 1h = 设置中断
22	SSTART	R	0h	启动条件中断 0h = 清除中断 1h = 设置中断
21	STXEMPTY	R	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
20	SRXFIFOFULL	R	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
19	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
18	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
17	STXDONE	R	0h	目标发送事务完成中断 0h = 未发生中断 1h = 设置中断屏蔽
16	SRXDONE	R	0h	目标接收数据中断 表示已接收到一个字节的信号 0h = 未发生中断 1h = 设置中断屏蔽
15	TIMEOUTB	R	0h	超时 B 中断 0h = 未发生中断 1h = 已发生中断
14	TIMEOUTA	R	0h	超时 A 中断 0h = 未发生中断 1h = 设置中断屏蔽
13	MPEC_RX_ERR	R	0h	控制器 RX PEC 错误中断 0h = 未发生中断 1h = 已发生中断
12	MDMA_DONE_RX	R	0h	事件通道 RX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
11	MDMA_DONE_TX	R	0h	事件通道 TX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
10	MARBLOST	R	0h	仲裁失败中断 0h = 未发生中断 1h = 设置中断屏蔽
9	MSTOP	R	0h	STOP 检测中断 0h = 未发生中断 1h = 设置中断屏蔽
8	MSTART	R	0h	START 检测中断 0h = 未发生中断 1h = 设置中断屏蔽
7	MnACK	R	0h	地址/数据 NACK 中断 0h = 未发生中断 1h = 设置中断屏蔽
6	RESERVED	R	0h	

表 18-43. RIS 字段说明 (continued)

位	字段	类型	复位	说明
5	MTXEMPTY	R	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
4	MRXFIFOFULL	R	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
3	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXDONE	R	0h	控制器发送事务完成中断 0h = 未发生中断 1h = 设置中断屏蔽
0	MRXDONE	R	0h	控制器接收事务完成中断 0h = 未发生中断 1h = 设置中断屏蔽

18.3.22 MIS (偏移 = 1038h) [复位 = 00000000h]

图 18-39 展示了 MIS，表 18-44 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 18-39. MIS

31	30	29	28	27	26	25	24
INTR_OVFL	SARBLOST	SRX_OVFL	STX_UNFL	SPEC_RX_ERR	SDMA_DONE_RX	SDMA_DONE_TX	SGENCALL
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
SSTOP	SSTART	STXEMPTY	SRXFIFOFULL	STXFIFOTRG	SRXFIFOTRG	STXDONE	SRXDONE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
TIMEOUTB	TIMEOUTA	MPEC_RX_ERR	MDMA_DONE_RX	MDMA_DONE_TX	MARBLOST	MSTOP	MSTART
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
MnACK	RESERVED	MTXEMPTY	MRXFIFOFULL	MTXFIFOTRG	MRXFIFOTRG	MTXDONE	MRXDONE
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 18-44. MIS 字段说明

位	字段	类型	复位	说明
31	INTR_OVFL	R	0h	中断上溢 0h = 未发生中断 1h = 已发生中断
30	SARBLOST	R	0h	目标仲裁失败 0h = 清除中断屏蔽 1h = 设置中断屏蔽
29	SRX_OVFL	R	0h	目标 RX FIFO 上溢 0h = 清除中断屏蔽 1h = 设置中断屏蔽
28	STX_UNFL	R	0h	目标 TX FIFO 下溢 0h = 清除中断屏蔽 1h = 设置中断屏蔽
27	SPEC_RX_ERR	R	0h	目标 RX PEC 错误中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
26	SDMA_DONE_RX	R	0h	事件通道 RX 上 DMA 完成 0h = 清除 MIS 1h =设置 MIS
25	SDMA_DONE_TX	R	0h	事件通道 TX 上 DMA 完成 0h = 清除 MIS 1h = 设置 MIS
24	SGENCALL	R	0h	通用广播中断 0h = 未发生中断 1h = 设置中断屏蔽
23	SSTOP	R	0h	目标 STOP 检测中断 0h = 清除 MIS 1h = 设置 MIS

表 18-44. MIS 字段说明 (continued)

位	字段	类型	复位	说明
22	SSTART	R	0h	目标 START 检测中断 0h = 清除 MIS 1h = 设置 MIS
21	STXEMPTY	R	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
20	SRXFIFOFULL	R	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
19	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
18	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
17	STXDONE	R	0h	目标发送事务完成中断 0h = 未发生中断 1h = 设置中断屏蔽
16	SRXDONE	R	0h	目标接收数据中断 表示已接收到一个字节的信号 0h = 未发生中断 1h = 设置中断屏蔽
15	TIMEOUTB	R	0h	超时 B 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
14	TIMEOUTA	R	0h	超时 A 中断 0h = 未发生中断 1h = 设置中断屏蔽
13	MPEC_RX_ERR	R	0h	控制器 RX PEC 错误中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
12	MDMA_DONE_RX	R	0h	事件通道 RX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
11	MDMA_DONE_TX	R	0h	事件通道 TX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
10	MARBLOST	R	0h	仲裁失败中断 0h = 未发生中断 1h = 设置中断屏蔽
9	MSTOP	R	0h	STOP 检测中断 0h = 未发生中断 1h = 设置中断屏蔽
8	MSTART	R	0h	START 检测中断 0h = 未发生中断 1h = 设置中断屏蔽
7	MnACK	R	0h	地址/数据 NACK 中断 0h = 未发生中断 1h = 设置中断屏蔽
6	RESERVED	R	0h	

表 18-44. MIS 字段说明 (continued)

位	字段	类型	复位	说明
5	MTXEMPTY	R	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
4	MRXFIFOFULL	R	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 未发生中断 1h = 设置中断屏蔽
3	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXDONE	R	0h	控制器发送事务完成中断 0h = 未发生中断 1h = 设置中断屏蔽
0	MRXDONE	R	0h	控制器接收数据中断 0h = 未发生中断 1h = 设置中断屏蔽

18.3.23 ISET (偏移 = 1040h) [复位 = 00000000h]

图 18-40 展示了 ISET，表 18-45 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 18-40. ISET

31	30	29	28	27	26	25	24
INTR_OVFL	SARBLOST	SRX_OVFL	STX_UNFL	SPEC_RX_ERR	SDMA_DONE_RX	SDMA_DONE_TX	SGENCALL
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
SSTOP	SSTART	STXEMPTY	SRXFIFOFULL	STXFIFOTRG	SRXFIFOTRG	STXDONE	SRXDONE
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
TIMEOUTB	TIMEOUTA	MPEC_RX_ERR	MDMA_DONE_RX	MDMA_DONE_TX	MARBLOST	MSTOP	MSTART
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
MnACK	RESERVED	MTXEMPTY	MRXFIFOFULL	MTXFIFOTRG	MRXFIFOTRG	MTXDONE	MRXDONE
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 18-45. ISET 字段说明

位	字段	类型	复位	说明
31	INTR_OVFL	W	0h	中断上溢 0h = 无效 1h = 设置中断
30	SARBLOST	W	0h	目标仲裁失败 0h = 写入 0 无效 1h = 设置中断
29	SRX_OVFL	W	0h	目标 RX FIFO 上溢 0h = 写入 0 无效 1h = 设置中断
28	STX_UNFL	W	0h	目标 TX FIFO 下溢 0h = 写入 0 无效 1h = 设置中断
27	SPEC_RX_ERR	W	0h	目标 RX PEC 错误中断 0h = 写入 0 无效 1h = 设置中断
26	SDMA_DONE_RX	W	0h	事件通道 RX 上 DMA 完成 0h = 写入 0 无效 1h = 设置中断
25	SDMA_DONE_TX	W	0h	事件通道 TX 上 DMA 完成 0h = 写入 0 无效 1h = 设置中断
24	SGENCALL	W	0h	通用广播中断 0h = 写入 0 无效 1h = 设置中断屏蔽
23	SSTOP	W	0h	停止条件中断 0h = 写入 0 无效 1h = 设置中断

表 18-45. ISET 字段说明 (continued)

位	字段	类型	复位	说明
22	SSTART	W	0h	启动条件中断 0h = 写入 0 无效 1h = 设置中断
21	STXEMPTY	W	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 写入 0 无效 1h = 设置中断屏蔽
20	SRXFIFOFULL	W	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
19	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
18	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
17	STXDONE	W	0h	目标发送事务完成中断 0h = 写入 0 无效 1h = 设置中断屏蔽
16	SRXDONE	W	0h	目标接收数据中断 表示已接收到一个字节的信号 0h = 写入 0 无效 1h = 设置中断屏蔽
15	TIMEOUTB	W	0h	超时 B 中断 0h = 写入 0 无效 1h = 设置中断
14	TIMEOUTA	W	0h	超时 A 中断 0h = 写入 0 无效 1h = 设置中断屏蔽
13	MPEC_RX_ERR	W	0h	控制器 RX PEC 错误中断 0h = 写入 0 无效 1h = 设置中断
12	MDMA_DONE_RX	W	0h	事件通道 RX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
11	MDMA_DONE_TX	W	0h	事件通道 TX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
10	MARBLOST	W	0h	仲裁失败中断 0h = 写入 0 无效 1h = 设置中断屏蔽
9	MSTOP	W	0h	STOP 检测中断 0h = 写入 0 无效 1h = 设置中断屏蔽
8	MSTART	W	0h	START 检测中断 0h = 写入 0 无效 1h = 设置中断屏蔽
7	MnACK	W	0h	地址/数据 NACK 中断 0h = 写入 0 无效 1h = 设置中断屏蔽
6	RESERVED	W	0h	

表 18-45. ISET 字段说明 (continued)

位	字段	类型	复位	说明
5	MTXEMPTY	W	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 写入 0 无效 1h = 设置中断屏蔽
4	MRXFIFOFULL	W	0h	RXFIFO 满事件。 0h = 写入 0 无效 1h = 设置中断屏蔽
3	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXDONE	W	0h	控制器发送事务完成中断 0h = 写入 0 无效 1h = 设置中断屏蔽
0	MRXDONE	W	0h	控制器接收数据中断 表示已接收到一个字节的信号 0h = 写入 0 无效 1h = 设置中断屏蔽

18.3.24 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 18-41 展示了 ICLR，表 18-46 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 18-41. ICLR

31	30	29	28	27	26	25	24
INTR_OVFL	SARBLOST	SRX_OVFL	STX_UNFL	SPEC_RX_ERR	SDMA_DONE_RX	SDMA_DONE_TX	SGENCALL
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
SSTOP	SSTART	STXEMPTY	SRXFIFOFULL	STXFIFOTRG	SRXFIFOTRG	STXDONE	SRXDONE
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
15	14	13	12	11	10	9	8
TIMEOUTB	TIMEOUTA	MPEC_RX_ERR	MDMA_DONE_RX	MDMA_DONE_TX	MARBLOST	MSTOP	MSTART
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
MnACK	RESERVED	MTXEMPTY	MRXFIFOFULL	MTXFIFOTRG	MRXFIFOTRG	MTXDONE	MRXDONE
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 18-46. ICLR 字段说明

位	字段	类型	复位	说明
31	INTR_OVFL	W	0h	中断上溢 0h = 无效 1h = 清除中断
30	SARBLOST	W	0h	目标仲裁失败 0h = 写入 0 无效 1h = 清除中断
29	SRX_OVFL	W	0h	目标 RX FIFO 上溢 0h = 写入 0 无效 1h = 清除中断
28	STX_UNFL	W	0h	目标 TX FIFO 下溢 0h = 写入 0 无效 1h = 清除中断
27	SPEC_RX_ERR	W	0h	目标 RX PEC 错误中断 0h = 写入 0 无效 1h = 清除中断
26	SDMA_DONE_RX	W	0h	事件通道 RX 上 DMA 完成 0h = 写入 0 无效 1h = 清除中断
25	SDMA_DONE_TX	W	0h	事件通道 TX 上 DMA 完成 0h = 写入 0 无效 1h = 清除中断
24	SGENCALL	W	0h	通用广播中断 0h = 写入 0 无效 1h = 设置中断屏蔽
23	SSTOP	W	0h	目标 STOP 检测中断 0h = 写入 0 无效 1h = 清除中断

表 18-46. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
22	SSTART	W	0h	目标 START 检测中断 0h = 写入 0 无效 1h = 清除中断
21	STXEMPTY	W	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 写入 0 无效 1h = 设置中断屏蔽
20	SRXFIFOFULL	W	0h	RXFIFO 满事件。如果 RX FIFO 已满，则设置此中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
19	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
18	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
17	STXDONE	W	0h	目标发送事务完成中断 0h = 写入 0 无效 1h = 设置中断屏蔽
16	SRXDONE	W	0h	目标接收数据中断 表示已接收到一个字节的信号 0h = 写入 0 无效 1h = 设置中断屏蔽
15	TIMEOUTB	W	0h	超时 B 中断 0h = 写入 0 无效 1h = 清除中断
14	TIMEOUTA	W	0h	超时 A 中断 0h = 写入 0 无效 1h = 设置中断屏蔽
13	MPEC_RX_ERR	W	0h	控制器 RX PEC 错误中断 0h = 写入 0 无效 1h = 清除中断
12	MDMA_DONE_RX	W	0h	事件通道 RX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
11	MDMA_DONE_TX	W	0h	事件通道 TX 上 DMA 完成 0h = 中断已禁用 1h = 设置中断屏蔽
10	MARBLOST	W	0h	仲裁失败中断 0h = 写入 0 无效 1h = 设置中断屏蔽
9	MSTOP	W	0h	STOP 检测中断 0h = 写入 0 无效 1h = 设置中断屏蔽
8	MSTART	W	0h	START 检测中断 0h = 写入 0 无效 1h = 设置中断屏蔽
7	MnACK	W	0h	地址/数据 NACK 中断 0h = 写入 0 无效 1h = 设置中断屏蔽
6	RESERVED	W	0h	

表 18-46. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
5	MTXEMPTY	W	0h	发送 FIFO 空中断屏蔽。如果发送 FIFO 中的所有数据都已移出并且发送进入空闲模式，则设置此中断。 0h = 写入 0 无效 1h = 设置中断屏蔽
4	MRXFIFOFULL	W	0h	RXFIFO 满事件。 0h = 写入 0 无效 1h = 设置中断屏蔽
3	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXDONE	W	0h	控制器发送事务完成中断 0h = 写入 0 无效 1h = 设置中断屏蔽
0	MRXDONE	W	0h	控制器接收数据中断 表示已接收到一个字节的信号 0h = 写入 0 无效 1h = 设置中断屏蔽

18.3.25 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 18-42 展示了 IIDX，表 18-47 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。

图 18-42. IIDX

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

表 18-47. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	I2C 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 RIS 和 MISC 中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 01h = 控制器接收 FIFO 触发电平 02h = 控制器发送 FIFO 触发电平 03h = 目标接收 FIFO 触发电平 04h = 目标发送 FIFO 触发电平

18.3.26 IMASK（偏移 = 1058h）[复位= 00000000h]

图 18-43 展示了 IMASK，表 18-48 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 18-43. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 18-48. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	STXFIFOTRG	R/W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R/W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R/W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R/W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.27 RIS（偏移 = 1060h）[复位 = 00000000h]

图 18-44 展示了 RIS，表 18-49 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 18-44. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R-0h				R-0h	R-0h	R-0h	R-0h

表 18-49. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.28 MIS (偏移 = 1068h) [复位 = 00000000h]

图 18-45 展示了 MIS，表 18-50 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 18-45. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R-0h				R-0h	R-0h	R-0h	R-0h

表 18-50. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.29 ISET (偏移 = 1070h) [复位 = 00000000h]

图 18-46 展示了 ISET，表 18-51 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 18-46. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
W-0h				W-0h	W-0h	W-0h	W-0h

表 18-51. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.30 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 18-47 展示了 ICLR，表 18-52 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 18-47. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
W-0h				W-0h	W-0h	W-0h	W-0h

表 18-52. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.31 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 18-48 展示了 IIDX，表 18-53 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。

图 18-48. IIDX

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

表 18-53. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	I2C 模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 RIS 和 MISC 中的相应中断标志。15h-1Fh = 保留 00h = 无中断挂起 01h = 控制器接收 FIFO 触发电平 02h = 控制器发送 FIFO 触发电平 03h = 目标接收 FIFO 触发电平 04h = 目标发送 FIFO 触发电平

18.3.32 IMASK（偏移 = 1088h）[复位= 00000000h]

图 18-49 展示了 IMASK，表 18-54 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 18-49. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 18-54. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	STXFIFOTRG	R/W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R/W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R/W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R/W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.33 RIS（偏移 = 1090h）[复位 = 00000000h]

图 18-50 展示了 RIS，表 18-55 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 18-50. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R-0h				R-0h	R-0h	R-0h	R-0h

表 18-55. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.34 MIS（偏移 = 1098h）[复位 = 00000000h]

图 18-51 展示了 MIS，表 18-56 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 18-51. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
R-0h				R-0h	R-0h	R-0h	R-0h

表 18-56. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	STXFIFOTRG	R	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	R	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	R	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	R	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.35 ISET（偏移 = 10A0h）[复位 = 00000000h]

图 18-52 展示了 ISET，表 18-57 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 18-52. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
W-0h				W-0h	W-0h	W-0h	W-0h

表 18-57. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.36 ICLR（偏移 = 10A8h）[复位 = 00000000h]

图 18-53 展示了 ICLR，表 18-58 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 18-53. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				STXFIFOTRG	SRXFIFOTRG	MTXFIFOTRG	MRXFIFOTRG
W-0h				W-0h	W-0h	W-0h	W-0h

表 18-58. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	STXFIFOTRG	W	0h	目标发送 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SRXFIFOTRG	W	0h	目标接收 FIFO 触发信号 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	MTXFIFOTRG	W	0h	控制器发送 FIFO 触发信号 当发送 FIFO 包含 <= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	MRXFIFOTRG	W	0h	控制器接收 FIFO 触发信号 当 RX FIFO 包含 >= 定义的字节时触发 0h = 清除中断屏蔽 1h = 设置中断屏蔽

18.3.37 EVT_MODE（偏移 = 10E0h）[复位 = 00000029h]

图 18-54 展示了 EVT_MODE，表 18-59 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 18-54. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED		EVT2_CFG		INT1_CFG		INT0_CFG	
R/W-		R-2h		R-2h		R-1h	

表 18-59. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	EVT2_CFG	R	2h	none.INT_EVENT2 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	INT1_CFG	R	2h	none.INT_EVENT1 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	none.INT_EVENT0 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

18.3.38 INTCTL（偏移 = 10E4h）[复位 = 00000000h]

图 18-55 展示了 INTCTL，表 18-60 中对此进行了介绍。

返回到汇总表。

中断控制寄存器

图 18-55. INTCTL

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

表 18-60. INTCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	INTEVAL	W	0h	向该字段写入 1 会重新评估中断源。 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。

18.3.39 DESC（偏移 = 10FCh）[复位 = 15110010h]

图 18-56 展示了 DESC，表 18-61 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 18-56. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-1511h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-0h				R-0h				R-1h				R-0h			

表 18-61. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	1511h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数 0h = 最小值 Fh = 尽可能高的值
7-4	MAJREV	R	1h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

18.3.40 GFCTL（偏移 = 1200h）[复位 = 00000F00h]

图 18-57 展示了 GFCTL，表 18-62 中对此进行了介绍。

返回到汇总表。

该寄存器控制 SCL 和 SDA 线上的干扰滤波器

图 18-57. GFCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
保留				CHAIN	AGFSEL		AGFEN
R/W-				R/W-1h	R/W-3h		R/W-1h
7	6	5	4	3	2	1	0
ESERVED					DGFSEL		
R/W-					R/W-0h		

表 18-62. GFCTL 字段说明

位	字段	类型	复位	说明
31-12	保留	R/W	0h	
11	CHAIN	R/W	1h	模拟和数字噪声滤波器链接使能。 0h = 为 0 时，禁用链接，只有数字滤波器输出可用于 IP 逻辑以进行过采样 1h = 为 1 时，模拟和数字干扰滤波器将链接起来，组合的输出可用于 IP 逻辑以进行过采样
10-9	AGFSEL	R/W	3h	模拟干扰抑制脉宽 该字段可控制 SCL 和 SDA 线上用于进行模拟干扰抑制的脉宽选择。请参阅器件数据表以了解确切值。 （仅适用于 ULP I2C） 0h = 过滤掉长度短于 5ns 的脉冲。 1h = 过滤掉长度短于 10ns 的脉冲。 2h = 过滤掉长度短于 25ns 的脉冲。 3h = 过滤掉长度短于 50ns 的脉冲。
8	AGFEN	R/W	1h	模拟干扰抑制使能 0h = 禁用模拟干扰滤波器 1h = 启用模拟干扰滤波器
7-3	RESERVED	R/W	0h	
2-0	DGFSEL	R/W	0h	干扰抑制脉宽 该字段可控制 SCL 和 SDA 线上用于进行干扰抑制的脉宽选择。以下值是功能时钟方面的干扰抑制值。 （仅适用于内核域） 0h = 旁路 1h = 1 个时钟 2h = 2 个时钟 3h = 3 个时钟 4h = 4 个时钟 5h = 8 个时钟 6h = 16 个时钟 7h = 31 个时钟

18.3.41 TIMEOUT_CTL（偏移 = 1204h）[复位 = 00020002h]

图 18-58 展示了 TIMEOUT_CTL，表 18-63 中对此进行了介绍。

返回到汇总表。

该寄存器包含超时计数器 A 和 B 的控制

图 18-58. TIMEOUT_CTL

31	30	29	28	27	26	25	24
TCNTBEN	RESERVED						
R/W-0h	R/W-0h						
23	22	21	20	19	18	17	16
TCNTLB							
R/W-2h							
15	14	13	12	11	10	9	8
TCNTAEN	RESERVED						
R/W-0h	R/W-0h						
7	6	5	4	3	2	1	0
TCNTLA							
R/W-2h							

表 18-63. TIMEOUT_CTL 字段说明

位	字段	类型	复位	说明
31	TCNTBEN	R/W	0h	超时计数器 B 使能 0h = 禁用超时计数器 B 1h = 启用超时计数器 B
30-24	保留	R/W	0h	
23-16	TCNTLB	R/W	2h	超时计数 B 加载：计数器 B 用于 SCL 高电平检测。该字段包含用于超时 B 计数的 12 位预加载值的高 8 位。注：CNTLB 的值必须大于 1h。 每个计数等于 1* 时钟周期。例如，使用 10MHz 功能时钟时，一个超时周期将等于 1*100ns。 0h = 尽可能小的值 FFh = 尽可能高的值
15	TCNTAEN	R/W	0h	超时计数器 A 使能 0h = 禁用超时计数器 B 1h = 启用超时计数器 B
14-8	保留	R/W	0h	
7-0	TCNTLA	R/W	2h	超时计数器 A 加载值 计数器 A 用于 SCL 低电平检测。该字段包含用于超时 A 计数的 12 位预加载值的高 8 位。注：CNTLA 的值必须大于 1h。 每个计数等于功能时钟超时周期的 520 倍。例如，使用 8MHz 功能时钟和 100KHz 工作 I2C 时钟时，一个超时周期将等于 (1 / 8MHz) * 520，即 65us。 0h = 最小值 FFh = 尽可能高的值

18.3.42 TIMEOUT_CNT（偏移 = 1208h）[复位 = 00020002h]

图 18-59 展示了 TIMEOUT_CNT，表 18-64 中对此进行了介绍。

返回到汇总表。

该寄存器包含计数器 A 和 B 的 12 位当前计数器值的高 8 位。计数器的低 4 位非用户可见，始终为 0h。

图 18-59. TIMEOUT_CNT

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

表 18-64. TIMEOUT_CNT 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23-16	TCNTB	R	2h	超时计数 B 当前计数：该字段包含超时计数器 B 的 12 位当前计数器的高 8 位 0h = 最小值 FFh = 尽可能高的值
15-8	保留	R	0h	
7-0	TCNTA	R	2h	超时计数 A 当前计数：该字段包含超时计数器 A 的 12 位当前计数器的高 8 位 0h = 最小值 FFh = 尽可能高的值

18.3.43 MSA (偏移 = 1210h) [复位 = 00000000h]

图 18-60 展示了 MSA，表 18-65 中对此进行了介绍。

返回到汇总表。

I2C 控制器目标地址寄存器

图 18-60. MSA

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
MMODE	RESERVED				SADDR		
R/W-0h	R/W-0h				R/W-0h		
7	6	5	4	3	2	1	0
SADDR							DIR
R/W-0h							R/W-0h

表 18-65. MSA 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	MMODE	R/W	0h	该位选择要在控制器模式下使用的寻址模式 为 0 时，使用 7 位寻址。 为 1 时，使用 10 位寻址。 0h = 7 位寻址模式 1h = 10 位寻址模式
14-11	保留	R/W	0h	
10-1	SADDR	R/W	0h	I2C 目标地址。该字段指定目标地址的 A9 到 A0 位。 在由 MSA.MODE 位选择的 7 位寻址模式中，前 3 位是无关位 0h = 最小值 3FFh = 尽可能高的值
0	DIR	R/W	0h	接收/发送 R/S 位指定下一个控制器操作是接收 (高电平) 还是发送 (低电平)。 0h = 发送 1h = 接收 0h = 控制器处于发送模式。 1h = 控制器处于接收模式。

18.3.44 MCTR（偏移 = 1214h）[复位 = 00000000h]

图 18-61 展示了 MCTR，表 18-66 中对此进行了介绍。

返回到汇总表。

该控制寄存器用于配置 I2C 控制器操作。START 位用来产生开始或重复开始信号。STOP 位决定周期是在数据周期结束时停止，还是继续运行下一个传输周期（可能是重复 START）。为生成单个发送周期，I2C 控制器目标地址（MSA）寄存器中写入所需的地址，R/S 位清零，此寄存器中写入 ACK = X（0 或 1）、STOP = 1、START = 1、RUN = 1，从而执行操作并停止。当操作完成（或由于错误而中止）时，字节事务完成中断将激活，并可从 MRXDATA 寄存器中读取数据。当 I2C 模块在控制器接收器模式下运行时，设置 ACK 位会使 I2C 总线控制器在每个字节后自动发送确认。当 I2C 总线控制器无需接收从目标发送器发送的更多数据时，必须清除该位。

图 18-61. MCTR

31	30	29	28	27	26	25	24
RESERVED				MBLEN			
R/W-0h				R/W-0h			
23	22	21	20	19	18	17	16
MBLEN							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED		RD_ON_TXEMPTY	MACKOEN	ACK	停止	启动	BURSTRUN
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 18-66. MCTR 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	R/W	0h	
27-16	MBLEN	R/W	0h	I2C 事务长度 该字段包含已编程的事务字节长度。 0h = 最小值 FFFh = 尽可能高的值
15-6	RESERVED	R/W	0h	
5	RD_ON_TXEMPTY	R/W	0h	TX 为空时读取 0h = 无特殊行为 1h = 为 1 时，控制器将发送来自 TX FIFO 的所有字节，然后继续执行编程的突发运行读取。 如果 DIR 在 MSA 中未设置为读取，则会忽略该位。必须在 MCTR 中设置启动条件以实现正确的 I2C 协议。在发送 TX FIFO 中的字节之前，控制器将首先发送启动条件，即 I2C 地址的 R/W 位设置为写入。当 TX FIFO 为空时，I2C 事务将按照 MTCR 和 MSA 中的编程继续进行，而不发送停止条件。 旨在用于执行简单的基于 I2C 命令的读取转换，该转换将在启动后完成，无需获取中断来进行总线回转。
4	MACKOEN	R/W	0h	控制器 ACK 覆盖使能 0h = 无特殊行为 1h = 为 1 并且控制器正在接收数据并已接收到 MBLEN 中指示的字节数时，状态机将生成一个 rxdone 中断，并在 ACK 的开头等待 FW 指示应发送 ACK 还是 NACK。 选择 ACK 或 NACK 的方法是写入 MCTR 寄存器并相应地设置 ACK。此时还可以写入该寄存器中的其他字段以继续运行事务。如果发送 NACK，状态机将自动发送停止。

表 18-66. MCTR 字段说明 (continued)

位	字段	类型	复位	说明
3	ACK	R/W	0h	数据确认使能。 软件需要配置该位来发送 ACK 或 NACK。 请参阅“MCTR 字段说明”表中的字段说明。 0h = 控制器不自动确认事务最后一个接收的数据字节。 1h = 控制器自动确认事务最后一个接收的数据字节。
2	停止	R/W	0h	生成停止条件 0h = 控制器不生成停止条件。 1h = 控制器生成停止条件。请参阅“MCTR 字段说明”表中的字段说明。
1	启动	R/W	0h	生成启动条件 0h = 控制器不生成启动条件。 1h = 控制器生成启动条件或重复启动条件。请参阅“MCTR 字段说明”表中的字段说明。
0	BURSTRUN	R/W	0h	I2C 控制器使能 和启动事务 0h = 在标准模式下，此编码表示控制器无法发送或接收数据。 1h = 控制器能够发送或接收数据。请参阅“MCTR 字段说明”表中的字段说明。

18.3.45 MSR (偏移 = 1218h) [复位 = 00000020h]

图 18-62 展示了 MSR，表 18-67 中对此进行了介绍。

返回到汇总表。

状态寄存器指示 I2C 总线控制器的状态。

图 18-62. MSR

31	30	29	28	27	26	25	24
RESERVED				MBCNT			
R-				R-0h			
23	22	21	20	19	18	17	16
MBCNT							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-							
7	6	5	4	3	2	1	0
ESERVED	BUSBSY	IDLE (闲置)	ARBLST	DATACK	ADRACK	ERR	BUSY
R-	R-0h	R-1h	R-0h	R-0h	R-0h	R-0h	R-0h

表 18-67. MSR 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	R	0h	
27-16	MBCNT	R	0h	I2C 控制器事务计数 该字段包含事务的当前倒计时值。 0h = 最小值 FFFh = 尽可能高的值
15-7	RESERVED	R	0h	
6	BUSBSY	R	0h	I2C 总线忙 控制器状态机将等待清除该位后再启动事务。在多控制器环境中首次启用控制器时，FW 应在将 ACTIVE 设置为高电平后等待一个 I2C 时钟周期，然后再写入 MTCR 寄存器以启动事务，这样如果 SCL 变为低电平，便会触发 BUSBSY。 0h = I2C 总线空闲。 1h = 在启动时或 SCL 变为低电平时会设置该状态位。在停止时或发生 SCL 高电平总线忙超时并且 SCL 和 SDA 都为高电平时，会清除该位。当 ACTIVE 位为低电平时，会清除该状态。 请注意，控制器状态机将等待清除该位后再启动 I2C 事务。在多控制器环境中首次启用控制器时，FW 应在将 ACTIVE 设置为高电平后等待一个 I2C 时钟周期，然后再写入 MTCR 寄存器以启动事务，这样如果 SCL 变为低电平，便会触发 BUSBSY。
5	IDLE (闲置)	R	1h	I2C 空闲 0h = I2C 控制器非空闲。 1h = I2C 控制器空闲。
4	ARBLST	R	0h	仲裁失败 0h = I2C 控制器赢得仲裁。 1h = I2C 控制器仲裁失败。
3	DATACK	R	0h	确认数据 0h = 发送的数据已确认。 1h = 发送的数据未确认。
2	ADRACK	R	0h	确认地址 0h = 发送的地址已确认。 1h = 发送的地址未确认。

表 18-67. MSR 字段说明 (continued)

位	字段	类型	复位	说明
1	ERR	R	0h	错误 该错误可能源于目标地址未得到确认或者发送数据未得到确认。 0h = 在最后一个操作中未检测到错误。 1h = 在最后一个操作中发生了错误。
0	忙	R	0h	I2C 控制器 FSM 忙 在正在进行的事务中会设置 BUSY 位，因此会在发送/接收 MBLEN 中设置的数据量时（包括根据当前事务的需要生成启动、重新启动、地址和停止信号时）设置该位。 0h = 控制器空闲。 1h = 控制器忙。

18.3.46 MRXDATA（偏移 = 121Ch）[复位 = 00000000h]

图 18-63 展示了 MRXDATA，表 18-68 中对此进行了介绍。

返回到汇总表。

I2C 控制器 RX FIFO 读取数据字节
该字段包含此时在 RX FIFO 栈中读取的当前字节。
如果接收 FIFO 被禁用，则数据字节和状态位保存在接收保持寄存器（即接收 FIFO 的最底部单元）中。对本寄存器的读操作即可获取数据字节。

图 18-63. MRXDATA

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

表 18-68. MRXDATA 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	值	R	0h	接收的数据。 该字段包含最后接收的数据。 0h = 最小值 FFh = 尽可能高的值

18.3.47 MTXDATA（偏移 = 1220h）[复位 = 00000000h]

图 18-64 展示了 MTXDATA，表 18-69 中对此进行了介绍。

返回到汇总表。

I2C 控制器发送数据寄存器。
该寄存器是发送数据寄存器（FIFO 的接口）。当发送数据时，若 FIFO 已使能，则对本寄存器写入的数据将推入发送 FIFO。如果发送 FIFO 被禁用，则数据仅保存在发送保持寄存器（即发送 FIFO 的最底部单元）中。

图 18-64. MTXDATA

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

表 18-69. MTXDATA 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	值	R/W	0h	发送数据 该字节包含下一个事务期间要传输的数据。 0h = 最小值 FFh = 尽可能高的值

18.3.48 MTPR（偏移 = 1224h）[复位 = 00000001h]

图 18-65 展示了 MTPR，表 18-70 中对此进行了介绍。

返回到汇总表。

对该寄存器进行编程，以便设置 SCL 时钟的计时器周期，并将 SCL 时钟指定为标准模式。

图 18-65. MTPR

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

表 18-70. MTPR 字段说明

位	字段	类型	复位	说明
31-7	保留	R/W	0h	
6-0	TPR	R/W	1h	计时器周期 该字段用在 SCL_PERIOD 的计算公式中： SCL_PERIOD = (1 + TPR) × (SCL_LP + SCL_HP) × INT_CLK_PRD 其中： SCL_PRD 是 SCL 线周期（I2C 时钟）。 TPR 是计时器周期寄存器值（范围为 1 到 127）。 SCL_LP 是 SCL 低电平周期（固定为 6）。 SCL_HP 是 SCL 高电平周期（固定为 4）。 CLK_PRD 是以 ns 为单位的功能时钟周期。 0h = 最小值 7Fh = 尽可能高的值

18.3.49 MCR (偏移 = 1228h) [复位 = 00000000h]

图 18-66 展示了 MCR，表 18-71 中对此进行了介绍。

返回到汇总表。

控制器配置寄存器

图 18-66. MCR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							LPBK
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
ESERVED					CLKSTRETCH	MMST	ACTIVE
R/W-0h					R/W-0h	R/W-0h	R/W-0h

表 18-71. MCR 字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	0h	
8	LPBK	R/W	0h	I2C 环回 0h = 正常运行。 1h = 控制器处于测试模式环回配置中。
7-3	RESERVED	R/W	0h	
2	CLKSTRETCH	R/W	0h	时钟延展。该位控制对 I2C 总线时钟延展的支持。 0h = 禁用时钟延展检测。 如果总线上的任何目标都不支持时钟延展，则可以禁用此功能，以便能够达到总线上的最大速度。 1h = 启用时钟延展检测。 启用时钟延展可确保符合 I2C 标准，但可能会因时钟延展而限制速度。
1	MMST	R/W	0h	多控制器模式。在多控制器模式下，一旦检测到 SCL 线为高电平，SCL 高电平时间便会开始计时。如果未启用此功能，一旦 I2C 控制器将 SCL 线设置为高电平，高电平时间便会开始计时。 0h = 禁用多控制器模式。 1h = 启用多控制器模式。
0	有效	R/W	0h	器件激活。设置该位后，在通过写入 0 或进行复位而清除该位之前，不得再次设置该位，否则可能发生传输失败。 0h = 禁止 I2C 控制器运行。 1h = 允许 I2C 控制器运行。

18.3.50 MBMON（偏移 = 1234h）[复位 = 00000003h]

图 18-67 展示了 MBMON，表 18-72 中对此进行了介绍。

返回到汇总表。

该寄存器用来确定 SCL 和 SDA 的信号状态。

图 18-67. MBMON

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

表 18-72. MBMON 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1	SDA	R	1h	I2C SDA 状态 0h = I2CSDA 信号为低电平。 1h = I2CSDA 信号为高电平。注意：在复位期间和复位之后，SDA 引脚处于 GPIO 输入模式，内部拉电阻未启用。为了让 I2C 正确运行，用户应在开始任何 I2C 操作之前设置好外部上拉电阻。
0	SCL	R	1h	I2C SCL 状态 0h = I2CSCL 信号为低电平。 1h = I2CSCL 信号为高电平。注意：在复位期间和复位之后，SCL 引脚处于 GPIO 输入模式，内部拉电阻未启用。为了让 I2C 正确运行，用户应在开始任何 I2C 操作之前设置好外部上拉电阻。

18.3.51 MFIFOCTL（偏移 = 1238h）[复位 = 00000000h]

图 18-68 展示了 MFIFOCTL，表 18-73 中对此进行了介绍。

返回到汇总表。

I2C 控制器 FIFO 控制

图 18-68. MFIFOCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RXFLUSH	RESERVED					RXTRIG	
R/W-0h	R/W-0h					R/W-0h	
7	6	5	4	3	2	1	0
TXFLUSH	RESERVED					TXTRIG	
R/W-0h	R/W-0h					R/W-0h	

表 18-73. MFIFOCTL 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	RXFLUSH	R/W	0h	RX FIFO 清空 设置该位将清空 RX FIFO。 在清除该位以停止清空之前，应校验 RXFIFOCNT 为 0 以指示清空已完成。 0h = 不清空 FIFO 1h = 清空 FIFO
14-11	保留	R/W	0h	
10-8	RXTRIG	R/W	0h	RX FIFO 触发信号 指示 RX FIFO 中哪个填充级别将生成触发信号。 注意：将 RXTRIG 编程为 0x0 无效，因为没有数据从 RX FIFO 中传出。 0h = 当 RX FIFO 包含 >= 1 字节时触发 1h = 当 RX FIFO 包含 >= 2 字节时触发 2h = 当 RX FIFO 包含 >= 3 字节时触发 3h = 当 RX FIFO 包含 >= 4 字节时触发 4h = 当 RX FIFO 包含 >= 5 字节时触发 5h = 当 RX FIFO 包含 >= 6 字节时触发 6h = 当 RX FIFO 包含 >= 7 字节时触发 7h = 当 RX FIFO 包含 >= 8 字节时触发
7	TXFLUSH	R/W	0h	TX FIFO 清空 设置该位将清空 TX FIFO。 在清除该位以停止清空之前，应校验 TXFIFOCNT 为 8 以指示清空已完成。 0h = 不清空 FIFO 1h = 清空 FIFO
6-3	RESERVED	R/W	0h	

表 18-73. MFIFOCTL 字段说明 (continued)

位	字段	类型	复位	说明
2-0	TXTRIG	R/W	0h	TX FIFO 触发信号 指示 TX FIFO 中哪个填充级别将生成触发信号。 0h = 当 TX FIFO 为空时触发。 1h = 当 TX FIFO 包含 ≤ 1 字节时触发 2h = 当 TX FIFO 包含 ≤ 2 字节时触发 3h = 当 TX FIFO 包含 ≤ 3 字节时触发 4h = 当 TX FIFO 包含 ≤ 4 字节时触发 5h = 当 TX FIFO 包含 ≤ 5 字节时触发 6h = 当 TX FIFO 包含 ≤ 6 字节时触发 7h = 当 TX FIFO 包含 ≤ 7 字节时触发

18.3.52 MFIFOSR（偏移 = 123Ch）[复位 = 00000800h]

图 18-69 展示了 MFIFOSR，表 18-74 中对此进行了介绍。

返回到汇总表。

I2C 控制器 FIFO 状态寄存器
注意：仅当 BUSY 为 0 时才应读取该寄存器

图 18-69. MFIFOSR

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

表 18-74. MFIFOSR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15	TXFLUSH	R	0h	TX FIFO 清空 设置该位时，TX FIFO 的清空操作将激活。清除控制寄存器中的 TXFLUSH 位将停止。 0h = FIFO 清空未激活 1h = FIFO 清空激活
14-12	RESERVED	R	0h	
11-8	TXFIFOCNT	R	8h	可放入 TX FIFO 的字节数 0h = 最小值 8h = 尽可能高的值
7	RXFLUSH	R	0h	RX FIFO 清空 设置该位时，RX FIFO 的清空操作将激活。清除控制寄存器中的 RXFLUSH 位将停止。 0h = FIFO 清空未激活 1h = FIFO 清空激活
6-4	RESERVED	R	0h	
3-0	RXFIFOCNT	R	0h	可从 RX FIFO 读取的字节数 0h = 最小值 8h = 尽可能高的值

18.3.53 CONTROLLER_I2CPECCTL（偏移 = 1240h）[复位 = 00000000h]

图 18-70 展示了 CONTROLLER_I2CPECCTL，表 18-75 中对此进行了介绍。

返回到汇总表。

I2C 控制器 PEC 控制寄存器

图 18-70. CONTROLLER_I2CPECCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留			PECEN	RESERVED			PECCNT
R/W-0h			R/W-0h	R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
PECCNT							
R/W-0h							

表 18-75. CONTROLLER_I2CPECCTL 字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	PECEN	R/W	0h	PEC 使能 该位启用 SMB 数据包错误检查 (PEC)。启用后，对于除 Start、Stop、Ack 和 Nack 外的所有位都会计算 PEC。当状态机处于空闲状态时（该状态出现在停止之后或发生超时时），PEC LSFR 和字节计数器设置为 0。在发送或接收 PEC 字节后，计数器也设置为 0。请注意，NACK 会在导致 PEC 错误的 PEC 字节之后自动发送。 PEC 多项式为 x^8 + x^2 + x^1 + 1。 0h = 在控制器模式下禁用 PEC 1h = 在控制器模式下启用 PEC
11-9	保留	R/W	0h	
8-0	PECCNT	R/W	0h	PEC 计数 该字段为非零值时，将进行 I2C 字节数量的计数（请注意，虽然会在 I2C 地址计算 PEC，但不会在字节内进行计数）。当字节计数 = PECCNT 且状态机正在发送时，LSFR 的内容将加载到移位寄存器中，而不是加载从 TX FIFO 接收到的字节。当状态机正在接收时，在接收到该字节的最后一位后，会检查 LSFR，如果它不为零，则会生成 PEC RX 错误中断。必须填充 I2C 数据包以包括用于发送和接收的 PEC 字节。在发送模式下，FIFO 必须加载一个虚拟 PEC 字节。在接收模式下，PEC 字节将传递到 RX FIFO。 在正常的控制器用例中，FW 会设置 PECEN=1 和 PECCNT=SMB 数据包长度（不包括目标地址字节，但包括 PEC 字节）。然后，FW 会配置 DMA 以允许数据包独立完成，并写入 MCTR 以启动事务。 请注意，当字节计数 = PEC CNT 时，字节计数将复位为 0，并且在单个 I2C 事务中可以自动进行多次 PEC 计算。 请注意，对 Controller_I2CPECCTL 寄存器的任何写入操作都将清除控制器状态机中的当前 PEC 字节计数。 0h = 最小值 1FFh = 最大值

18.3.54 CONTROLLER_PECSR（偏移 = 1244h）[复位 = 00000000h]

图 18-71 展示了 CONTROLLER_PECSR，表 18-76 中对此进行了介绍。

返回到汇总表。

控制器 PEC 状态寄存器

图 18-71. CONTROLLER_PECSR

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

表 18-76. CONTROLLER_PECSR 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R	0h	
17	PECSTS_ERROR	R	0h	该状态位指示在最后一次停止前发生的事务中是否发生 PEC 检查错误。锁定在停止位置。 0h = 指示在最后一次停止前发生的事务中没有出现 PEC 检查错误 1h = 指示在最后一次停止前发生的事务中发生了 PEC 检查错误
16	PECSTS_CHECK	R	0h	该状态位指示在最后一次停止前发生的事务中是否检查了 PEC。锁定在停止位置。 0h = 指示在最后一次停止前发生的事务中没有检查 PEC 1h = 指示在最后一次停止前发生的事务中检查了 PEC
15-9	保留	R	0h	
8-0	PECBYTECNT	R	0h	PEC 字节计数 这是控制器状态机的当前 PEC 字节计数。 0h = 最小值 1FFh = 最大值

18.3.55 SOAR（偏移 = 1250h）[复位 = 00004000h]

图 18-72 展示了 SOAR，表 18-77 中对此进行了介绍。

返回到汇总表。

该寄存器包含七个用于标识 I2C 总线上 I2C 器件的地址位。

图 18-72. SOAR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
SMODE	OAREN	RESERVED				OAR	
R/W-0h	R/W-1h	R/W-0h				R/W-0h	
7	6	5	4	3	2	1	0
OAR							
R/W-0h							

表 18-77. SOAR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	SMODE	R/W	0h	该位选择要在目标模式中使用的寻址模式。 为 0 时，使用 7 位寻址。 为 1 时，使用 10 位寻址。 0h = 启用 7 位寻址 1h = 启用 10 位寻址
14	OAREN	R/W	1h	I2C 目标自身地址使能 0h = 禁用 OAR 地址 1h = 启用 OAR 地址
13-10	保留	R/W	0h	
9-0	OAR	R/W	0h	I2C 目标自身地址：该字段指定目标地址的 A9 到 A0 位。 在由 I2CSOAR.MODE 位选择的 7 位寻址模式中，前 3 位是无关位 0h = 最小值 3FFh = 尽可能高的值

18.3.56 SOAR2（偏移 = 1254h）[复位 = 00000000h]

图 18-73 展示了 SOAR2，表 18-78 中对此进行了介绍。

返回到汇总表。

该寄存器包含七个用于标识 I2C 总线上 I2C 器件备用地址的地址位。

图 18-73. SOAR2

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	OAR2_MASK						
R/W-0h	R/W-0h						
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
OAR2EN	OAR2						
R/W-0h	R/W-0h						

表 18-78. SOAR2 字段说明

位	字段	类型	复位	说明
31-23	保留	R/W	0h	
22-16	OAR2_MASK	R/W	0h	I2C 目标自身地址 2 屏蔽：该字段指定目标地址的 A6 到 A0 位。SOAR2.OAR2_MASK 字段内值为“1”的位将使相应的传入地址位默认匹配，不用考虑 SOAR2.OAR2 中的值，即相应的 SOAR2.OAR2 位是无关位。 0h = 最小值 7Fh = 最大值
15-8	保留	R/W	0h	
7	OAR2EN	R/W	0h	I2C 目标自身地址 2 使能 0h = 禁用备用地址。 1h = 允许在 OAR2 字段中使用备用地址。
6-0	OAR2	R/W	0h	I2C 目标自身地址 2 该字段指定备用的 OAR2 地址。 0h = 最小值 7Fh = 尽可能高的值

18.3.57 SCTR (偏移 = 1258h) [复位 = 00000404h]

图 18-74 展示了 SCTR，表 18-79 中对此进行了介绍。

返回到汇总表。

I2C 目标控制寄存器

图 18-74. SCTR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留					SWUEN	EN_DEFDEVADR	EN_ALRESPADR
R/W-0h					R/W-1h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
EN_DEFHOSTADR	RXFULL_ON_RREQ	TXWAIT_STALE_TXFIFO	TXTRIG_TXMODE	TXEMPTY_ON_TREQ	SCLKSTRETCH	GENCALL	ACTIVE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h	R/W-0h

表 18-79. SCTR 字段说明

位	字段	类型	复位	说明
31-11	保留	R/W	0h	
10	SWUEN	R/W	1h	目标唤醒使能 0h = 为 0 时，不允许目标在 START 检测时进行时钟延展 1h = 为 1 时，允许目标在 START 检测时进行时钟延展并等待更快的时钟可供使用。 这样可以在低功耗模式用例中为 I2C 提供纯净的唤醒支持
9	EN_DEFDEVADR	R/W	0h	启用默认器件地址 0h = 当该位为 0 时，不匹配默认器件地址。注意：如果在 SOAR/SOAR2 内部进行了相应编程，则仍然可能匹配。 1h = 当该位为 1 时，目标地址匹配逻辑会始终匹配默认器件地址 7’h110_0001。
8	EN_ALRESPADR	R/W	0h	启用警报响应地址 0h = 当该位为 0 时，不匹配警报响应地址。注意：如果在 SOAR/SOAR2 内部进行了相应编程，则仍然可能匹配 1h = 当该位为 1 时，目标地址匹配逻辑会始终匹配警报响应地址 7’h000_1100。
7	EN_DEFHOSTADR	R/W	0h	启用默认主机地址 0h = 当该位为 0 时，不匹配默认主机地址 注意：如果在 SOAR/SOAR2 内部进行了相应编程，则仍然可能匹配 1h = 当该位为 1 时，目标地址匹配逻辑会始终匹配默认主机地址 7’h000_1000。
6	RXFULL_ON_RREQ	R/W	0h	在 SSR 中所示的 RREQ 条件下产生的 RX 已满中断 0h = 为 0 时，仅当目标 RX FIFO 已满时才会设置 RIS:SRXFULL。这种情况下允许使用 SRXFULL 中断来指示 I2C 总线正在进行时钟延展，并且 FW 必须读取 RX FIFO 或对当前的 RX 字节进行 ACK/NACK。 1h = 为 1 时，在目标状态机处于 RX_WAIT 或 RX_ACK_WAIT 状态的情况下将设置 RIS:SRXFULL；由于 RX FIFO 已满或 ACKOEN 已设置且状态机正在等待 FW 对当前字节进行 ACK/NACK，而导致 I2C 事务进行时钟延展时，便会出现这种情况。

表 18-79. SCTR 字段说明 (continued)

位	字段	类型	复位	说明
5	TXWAIT_STALE_TXFIFO	R/W	0h	TX FIFO 中存在过时数据时的 TX 传输等待。 这可防止在下一个 I2C 数据包上自动发送 TX FIFO 中剩余的过时字节。注意：这应该与 TXEMPTY_ON_TREQ 设置一起使用，防止目标状态机在 FIFO 数据过时时等待 TX FIFO 数据而不发出中断通知。 0h = 为 0 时，发送到目标状态机的 TX FIFO 空信号指示 TX FIFO 为空。 1h = 为 1 时，发送到目标状态机的 TX FIFO 空信号指示 TX FIFO 为空或 TX FIFO 数据过时。当目标状态机离开 SSR 寄存器中定义的 TXMODE 时，如果 TX FIFO 中有数据，则 TX FIFO 数据将被确定为过时。如果 TX FIFO 中有剩余字节时发生停止或超时，可能会出现这种情况。
4	TXTRIG_TXMODE	R/W	0h	目标 FSM 处于 TX 模式时的 TX 触发信号 0h = 无特殊行为 1h = 为 1 时，在目标 TX FIFO 达到触发电平并且目标状态机处于 SSR 寄存器中定义的 TXMODE 的情况下将设置 RIS:TXFIFOTRG。清除后，在目标 TX FIFO 处于或者高于触发电平的情况下将设置 RIS:TXFIFOTRG。 此设置可用于延缓 TX DMA 直至事务开始。 这样可以在 I2C 空闲时配置 DMA，但让它等到事务开始加载目标 TX FIFO，这样它就可以从可能随时间变化的存储器缓冲区进行加载。
3	TXEMPTY_ON_TREQ	R/W	0h	TREQ 时的 TX 空中断 0h = 为 0 时，仅当目标 TX FIFO 为空的情况下才会设置 RIS:STXEMPTY。 这样可以使用 STXEMPTY 中断来指示 I2C 总线正在进行时钟延展并且需要目标 TX 数据。 1h = 为 1 时，在目标状态机处于 TX_WAIT 状态的情况下将设置 RIS:STXEMPTY；当 TX FIFO 为空且 I2C 事务处于时钟延展状态而等待 FIFO 接收数据时，便会出现这种情况。
2	SCLKSTRETCH	R/W	1h	目标时钟延展使能 0h = 禁用目标时钟延展 1h = 启用目标时钟延展
1	GENCALL	R/W	0h	通用广播响应使能 仅当 UCSWRST = 1 时修改。 0b = 不响应通用广播 1b = 响应通用广播 0h = 不响应通用广播 1h = 响应通用广播
0	有效	R/W	0h	器件激活。设置该位可启用目标功能。 0h = 禁止 I2C 目标运行。 1h = 允许 I2C 目标运行。

18.3.58 SSR（偏移 = 125Ch）[复位 = 00000000h]

图 18-75 展示了 SSR，表 18-80 中对此进行了介绍。

返回到汇总表。

写入时，该寄存器是控制寄存器；读取时，该寄存器是状态寄存器。

图 18-75. SSR

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED					ADDRMATCH		
R-0h					R-0h		
15	14	13	12	11	10	9	8
ADDRMATCH							STALE_TXFIFO
R-0h							R-0h
7	6	5	4	3	2	1	0
TXMODE	BUSBSY	QCMDRW	QCMDST	OAR2SEL	RXMODE	TREQ	RREQ
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 18-80. SSR 字段说明

位	字段	类型	复位	说明
31-19	RESERVED	R	0h	
18-9	ADDRMATCH	R	0h	指示发生目标地址匹配的地址 0h = 最小值 3FFh = 最大值
8	STALE_TXFIFO	R	0h	TX FIFO 过时 0h = TX FIFO 未过时 1h = TX FIFO 已过时 在前一个 I2C 事务期间 TX FIFO 未清空时，便会发生这种情况。
7	TXMODE	R	0h	目标 FSM 处于 TX 模式 0h = 总线方向设置为读取时，目标状态机未处于 TX_DATA、TX_WAIT、TX_ACK 或 ADDR_ACK 状态。 1h = 总线方向设置为读取时，目标状态机处于 TX_DATA、TX_WAIT、TX_ACK 或 ADDR_ACK 状态。
6	BUSBSY	R	0h	I2C 总线忙 0h = I2C 总线不忙 1h = I2C 总线忙 在超时时清除。
5	QCMDRW	R	0h	快速命令读取/写入 仅当设置了 QCMDST 位时，该位才有意义。 值说明： 0：快速指令是写入 1：快速命令是读取 0h = 快速命令是写入 1h = 快速命令是读取
4	QCMDST	R	0h	快速命令状态 值说明： 0：上一个通信传输是正常通信传输，或者通信传输没有发生。 1：最后一个事务是快速命令事务 0h = 最后一个事务是正常事务或没有发生事务。 1h = 最后一个事务是快速指令事务。

表 18-80. SSR 字段说明 (continued)

位	字段	类型	复位	说明
3	OAR2SEL	R	0h	OAR2 地址匹配 每次比较地址之后会重新评估该位。 0h = OAR2 地址不匹配，或匹配处于旧模式。 1h = OAR2 地址匹配并得到目标确认。
2	RXMODE	R	0h	目标 FSM 处于 RX 模式 0h = 总线方向设置为写入时，目标状态机未处于 RX_DATA、RX_ACK、RX_WAIT、RX_ACK_WAIT 或 ADDR_ACK 状态。 1h = 总线方向设置为写入时，目标状态机处于 RX_DATA、RX_ACK、RX_WAIT、RX_ACK_WAIT 或 ADDR_ACK 状态。
1	TREQ	R	0h	发送请求 0h = 无未处理的发送请求。 1h = I2C 控制器已被寻址为目标发送器，并且正在使用时钟延展技术来延迟控制器直到数据被写入 STXDATA FIFO（目标 TX FIFO 为空）。
0	RREQ	R	0h	接收请求 0h = 无未处理的接收数据。 1h = I2C 控制器具有来自 I2C 控制器的未处理的接收数据，并且正在使用时钟延展技术来延迟控制器直到从 SRXDATA FIFO 读取数据（目标 RX FIFO 已满）。

18.3.59 SRXDATA（偏移 = 1260h）[复位 = 00000000h]

图 18-76 展示了 SRXDATA，表 18-81 中对此进行了介绍。

返回到汇总表。

I2C 目标 RX FIFO 读取数据字节
该字段包含此时在 RX FIFO 栈中读取的当前字节。
如果接收 FIFO 被禁用，则数据字节和状态位保存在接收保持寄存器（即接收 FIFO 的最底部单元）中。对本寄存器的读操作即可获取数据字节。

图 18-76. SRXDATA

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

表 18-81. SRXDATA 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	值	R	0h	接收的数据。 该字段包含最后接收的数据。 0h = 最小值 FFh = 尽可能高的值

18.3.60 STXDATA（偏移 = 1264h）[复位 = 00000000h]

图 18-77 展示了 STXDATA，表 18-82 中对此进行了介绍。

返回到汇总表。

I2C 目标发送数据寄存器。
该寄存器是发送数据寄存器（FIFO 的接口）。当发送数据时，若 FIFO 已使能，则对本寄存器写入的数据将推入发送 FIFO。如果发送 FIFO 被禁用，则数据仅保存在发送保持寄存器（即发送 FIFO 的最底部单元）中。

图 18-77. STXDATA

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

表 18-82. STXDATA 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	值	R/W	0h	发送数据 该字节包含下一个事务期间要传输的数据。 0h = 最小值 FFh = 尽可能高的值

18.3.61 SACKCTL（偏移 = 1268h）[复位 = 00000000h]

图 18-78 展示了 SACKCTL，表 18-83 中对此进行了介绍。

返回到汇总表。

该寄存器使 I2C 目标能够对无效数据或命令进行否定确认 (NACK)，或对有效数据和命令进行确认 (ACK)。最后一个数据位之后会将 I2C 时钟拉至低电平，直到写入该寄存器。

图 18-78. SACKCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED			ACKOEN_ON_PECDONE	ACKOEN_ON_PECNEXT	ACKOEN_ON_START	ACKOVAL	ACKOEN
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 18-83. SACKCTL 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	0h	
4	ACKOEN_ON_PECDONE	R/W	0h	设置该位后将在接收到的 PEC 字节的 ACK/NACK 之后自动启用目标 ACKOEN 字段。 0h = 无特殊行为 1h = 设置该位后将在接收到的 PEC 字节的 ACK/NACK 之后自动启用目标 ACKOEN 字段。
3	ACKOEN_ON_PECNEXT	R/W	0h	设置该位后将在 PEC 字节之前接收到的字节的 ACK/NACK 之后自动启用目标 ACKOEN 字段。 请注意，设置 ACKOEN 后，状态机不会自动对 PEC 字节进行 ACK/NACK，FW 必须通过写入 Target_SACKCTL 来执行该功能。 0h = 无特殊行为 1h = 设置该位后将在 PEC 字节之前接收到的字节的 ACK/NACK 之后自动启用目标 ACKOEN 字段。 请注意，设置 ACKOEN 后，状态机不会自动对 PEC 字节进行 ACK/NACK，FW 必须通过写入 Target_SACKCTL 来执行该功能。
2	ACKOEN_ON_START	R/W	0h	设置该位后将在一个启动条件之后自动启用目标 ACKOEN 字段。 0h = 无特殊行为 1h = 设置该位后将在一个启动条件之后自动启用目标 ACKOEN 字段。
1	ACKOVAL	R/W	0h	I2C 目标 ACK 覆盖值 注意：对于通用广播，如果设置为 NACK，则会忽略该位，目标继续接收数据。 0h = 发送 ACK 以指示有效数据或命令。 1h = 发送 NACK 以指示无效数据或命令。
0	ACKOEN	R/W	0h	I2C 目标 ACK 覆盖使能 0h = 不提供响应。 1h = 根据写入 ACKOVAL 位的值发送 ACK 或 NACK。

18.3.62 SFIFOCTL（偏移 = 126Ch）[复位 = 00000000h]

图 18-79 展示了 SFIFOCTL，表 18-84 中对此进行了介绍。

返回到汇总表。

I2C 目标 FIFO 控制

图 18-79. SFIFOCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RXFLUSH	RESERVED					RXTRIG	
R/W-0h	R/W-0h					R/W-0h	
7	6	5	4	3	2	1	0
TXFLUSH	RESERVED					TXTRIG	
R/W-0h	R/W-0h					R/W-0h	

表 18-84. SFIFOCTL 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	RXFLUSH	R/W	0h	RX FIFO 清空 设置该位将清空 RX FIFO。 在清除该位以停止清空之前，应校验 RXFIFOCNT 为 0 以指示清空已完成。 0h = 不清空 FIFO 1h = 清空 FIFO
14-11	保留	R/W	0h	
10-8	RXTRIG	R/W	0h	RX FIFO 触发信号 指示 RX FIFO 中哪个填充级别将生成触发信号。 注意：将 RXTRIG 编程为 0x0 无效，因为没有数据从 RX FIFO 中传出。 4h = 当 RX FIFO 包含 >= 5 字节时触发 5h = 当 RX FIFO 包含 >= 6 字节时触发 6h = 当 RX FIFO 包含 >= 7 字节时触发 7h = 当 RX FIFO 包含 >= 8 字节时触发
7	TXFLUSH	R/W	0h	TX FIFO 清空 设置该位将清空 TX FIFO。 在清除该位以停止清空之前，应校验 TXFIFOCNT 为 8 以指示清空已完成。 0h = 不清空 FIFO 1h = 清空 FIFO
6-3	RESERVED	R/W	0h	
2-0	TXTRIG	R/W	0h	TX FIFO 触发信号 指示 TX FIFO 中哪个填充级别将生成触发信号。 4h = 当 TX FIFO 包含 ≤ 4 字节时触发 5h = 当 TX FIFO 包含 ≤ 5 字节时触发 6h = 当 TX FIFO 包含 ≤ 6 字节时触发 7h = 当 TX FIFO 包含 ≤ 7 字节时触发

18.3.63 SFIFOSR (偏移 = 1270h) [复位 = 00000800h]

图 18-80 展示了 SFIFOSR，表 18-85 中对此进行了介绍。

返回到汇总表。

I2C 目标 FIFO 状态寄存器
注意：仅当 BUSY 为 0 时才应读取该寄存器

图 18-80. SFIFOSR

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

表 18-85. SFIFOSR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15	TXFLUSH	R	0h	TX FIFO 清空 设置该位时，TX FIFO 的清空操作将激活。清除控制寄存器中的 TXFLUSH 位将停止。 0h = FIFO 清空未激活 1h = FIFO 清空激活
14-12	RESERVED	R	0h	
11-8	TXFIFOCNT	R	8h	可放入 TX FIFO 的字节数 0h = 最小值 8h = 尽可能高的值
7	RXFLUSH	R	0h	RX FIFO 清空 设置该位时，RX FIFO 的清空操作将激活。清除控制寄存器中的 RXFLUSH 位将停止。 0h = FIFO 清空未激活 1h = FIFO 清空激活
6-4	RESERVED	R	0h	
3-0	RXFIFOCNT	R	0h	可从 RX FIFO 读取的字节数 0h = 最小值 8h = 尽可能高的值

18.3.64 TARGET_PECCTL (偏移 = 1274h) [复位 = 00000000h]

图 18-81 展示了 TARGET_PECCTL，表 18-86 中对此进行了介绍。

返回到汇总表。

I2C 目标 PEC 控制寄存器

图 18-81. TARGET_PECCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留			PECEN	RESERVED			PECCNT
R/W-0h			R/W-0h	R/W-0h			R/W-0h
7	6	5	4	3	2	1	0
PECCNT							
R/W-0h							

表 18-86. TARGET_PECCTL 字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	PECEN	R/W	0h	PEC 使能 该位启用 SMB 数据包错误检查 (PEC)。启用后，对于除 Start、Stop、Ack 和 Nack 外的所有位都会计算 PEC。当状态机处于空闲状态时 (该状态出现在停止之后或发生超时时) ，PEC LSFR 和字节计数器设置为 0。在发送或接收 PEC 字节后，计数器也设置为 0。请注意，NACK 会在导致 PEC 错误的 PEC 字节之后自动发送。 PEC 多项式为 x^8 + x^2 + x^1 + 1。 0h = 禁用 PEC 发送和检查 1h = 启用 PEC 发送和检查
11-9	保留	R/W	0h	
8-0	PECCNT	R/W	0h	该字段为非零值时，将对 I2C 数据字节数进行计数。当字节计数 = PECCNT 且状态机正在发送时，LSFR 的内容将加载到移位寄存器中，而不是加载从 TX FIFO 接收到的字节。当状态机正在接收时，在接收到该字节的最后一位后，会检查 LSFR，如果它不为零，则会生成 PEC RX 错误中断。必须填充 I2C 数据包以包括用于发送和接收的 PEC 字节。在发送模式下，FIFO 必须加载一个虚拟 PEC 字节。在接收模式下，PEC 字节将传递到 RX FIFO。 在正常的目标用例中，FW 会设置 PECEN=1 和 PECCNT=0 并使用 ACKOEN 直到知道剩余的 SMB 数据包长度。然后，FW 会将 PECCNT 设置为剩余的数据包长度 (包括 PEC 字节) 。然后，FW 会配置 DMA，以便允许数据包独立完成并退出 NoAck 模式。 请注意，当字节计数 = PEC CNT 时，字节计数将复位为 0，并且在单个 I2C 事务中可以自动进行多次 PEC 计算 0h = 最小值 1FFh = 最大值

18.3.65 TARGET_PECSR (偏移 = 1278h) [复位 = 00000000h]

图 18-82 展示了 TARGET_PECSR，表 18-87 中对此进行了介绍。

返回到汇总表。

目标 PEC 状态寄存器

图 18-82. TARGET_PECSR

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

表 18-87. TARGET_PECSR 字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R	0h	
17	PECSTS_ERROR	R	0h	该状态位指示在最后一次停止前发生的事务中是否发生 PEC 检查错误。锁定在停止位置。 0h = 指示在最后一次停止前发生的事务中没有出现 PEC 检查错误 1h = 指示在最后一次停止前发生的事务中发生了 PEC 检查错误
16	PECSTS_CHECK	R	0h	该状态位指示在最后一次停止前发生的事务中是否检查了 PEC。锁定在停止位置。 0h = 指示在最后一次停止前发生的事务中没有检查 PEC 1h = 指示在最后一次停止前发生的事务中检查了 PEC
15-9	保留	R	0h	
8-0	PECBYTECNT	R	0h	这是目标状态机的当前 PEC 字节计数。 0h = 最小值 1FFh = 最大值

18.3.66 TEST0（偏移 = 1E00h）[复位 = 00000000h]

图 18-83 展示了 TEST0，表 18-88 中对此进行了介绍。

返回到汇总表。

测试 0 寄存器。

图 18-83. TEST0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
TRIM_SEL	SDA_TRIM_EN	SCL_TRIM_EN	DTB_2nd_Level_MUX_SEL		DTB_MUX_SEL		
R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h		

表 18-88. TEST0 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7	TRIM_SEL	R/W	0h	选择要修整的分支 0h = 选择在下降沿上修整 1h = 选择在上升沿上修整
6	SDA_TRIM_EN	R/W	0h	针对 SDA 启用模拟干扰滤波器修整 0h = 针对 SDA 禁用修整 1h = 针对 SDA 启用修整
5	SCL_TRIM_EN	R/W	0h	针对 SCL 启用模拟干扰滤波器修整 0h = 针对 SCL 禁用修整 1h = 针对 SCL 启用修整
4-3	DTB_2nd_Level_MUX_SEL	R/W	0h	该位字段用于选择 DTB 二级多路复用器数字输出信号。32B->8B 0h = 选择测试组 0 1h = 选择测试组 1 2h = 选择测试组 2 3h = 选择测试组 3
2-0	DTB_MUX_SEL	R/W	0h	该位字段用于选择 DTB 多路复用器数字输出信号。 0h = 禁用 DTB MUX 1h = 选择测试组 1 2h = 选择测试组 2 3h =选择测试组 3 4h = 选择测试组 4 5h = 选择测试组 5 6h = 选择测试组 6 7h = 选择测试组 7

This page intentionally left blank.

CAN-FD

模块化控制器局域网 (MCAN) 外设支持通过传统 CAN 和 CAN-FD 协议进行通信。

19.1 MCAN 概述

控制器局域网 (CAN) 是一种串行通信协议，用于有效地为具有高可靠性的分布式实时控制提供支持。CAN 具有较高的抗电气干扰能力，并且能够检测各种类型的错误。在 CAN 中，许多较短的信息会广播到整个网络，从而在系统的每个节点中提供数据一致性。

MCAN 模块支持传统 CAN 和 CAN FD (具有灵活数据速率的 CAN) 协议。CAN FD 特性可实现更高的吞吐量和增加每个数据帧的有效负载。传统 CAN 和 CAN FD 器件可以在同一网络上共存而不会发生任何冲突，前提是传统 CAN 器件使用部分网络收发器，其中该收发器可以检测和忽略 CAN FD，而不产生总线错误。MCAN 模块符合 ISO 11898-1:2015 标准。

备注

CAN FD 功能并非在所有器件上都可用。有关更多信息，请参阅特定于器件的数据表。

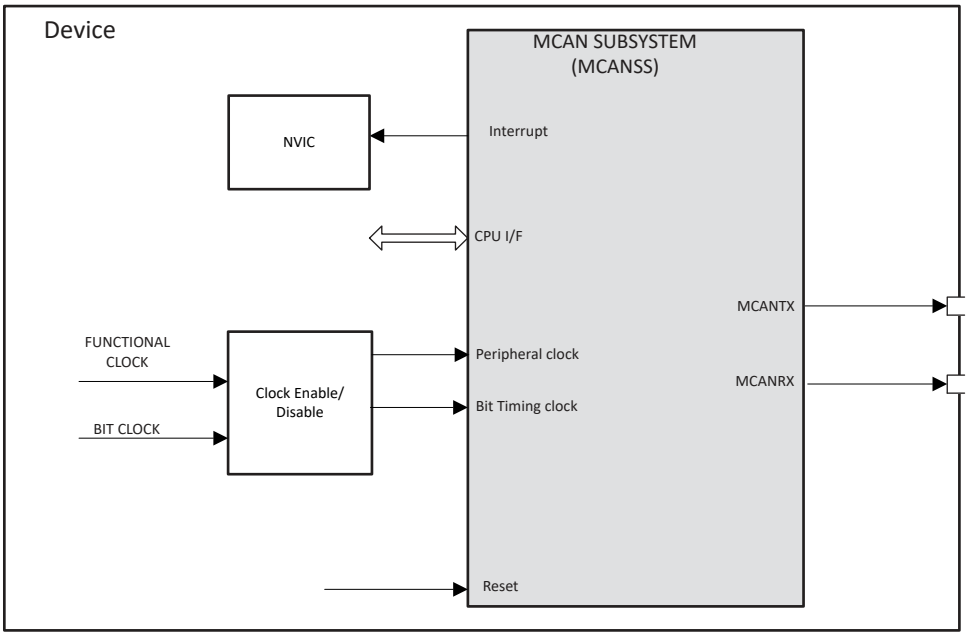


图 19-1. MCAN 方框图

19.1.1 MCAN 特性

MCAN 模块可实现以下特性：

- 符合 CAN 协议 2.0A、B 和 ISO 11898-1:2015 标准
- 完全支持 CAN FD（最多 64 个数据字节）
- 支持 AUTOSAR 和 SAE J1939
- 多达 32 个专用发送缓冲器
- 可配置的发送 FIFO，最多 32 个元素
- 可配置的发送队列，最多 32 个元素
- 可配置的发送事件 FIFO，最多 32 个元素
- 多达 14 个专用接收缓冲器
- 两个可配置的接收 FIFO，每个 FIFO 最多 14 个元素，具有 1kB 的消息 RAM
- 多达 128 个滤波器元素
- 用于自检的环回模式
- 可屏蔽中断（两条可配置的中断线路、可纠正的 ECC、计数器溢出和时钟停止或唤醒）
- 不可屏蔽中断（不可纠正的 ECC）
- 两个时钟域（CAN 时钟和主机时钟）
- 针对消息 RAM 的 ECC 检查
- 支持时钟停止和唤醒
- 时间戳计数器
- 1Mbps 标称比特率、8Mbps 数据比特率

不支持的特性：

- 主机总线防火墙
- 时钟校准
- 通过 CAN 进行调试

19.2 MCAN 环境

CAN 网络物理层由两线制差分总线（通常是双绞线）组成，并提供高水平抗干扰性能。需要使用一个外部 CAN 收发器 IC 来访问总线。

图 19-2 展示了典型的 MCAN 接线。表 19-1 介绍了 MCAN 模块的外部信号。

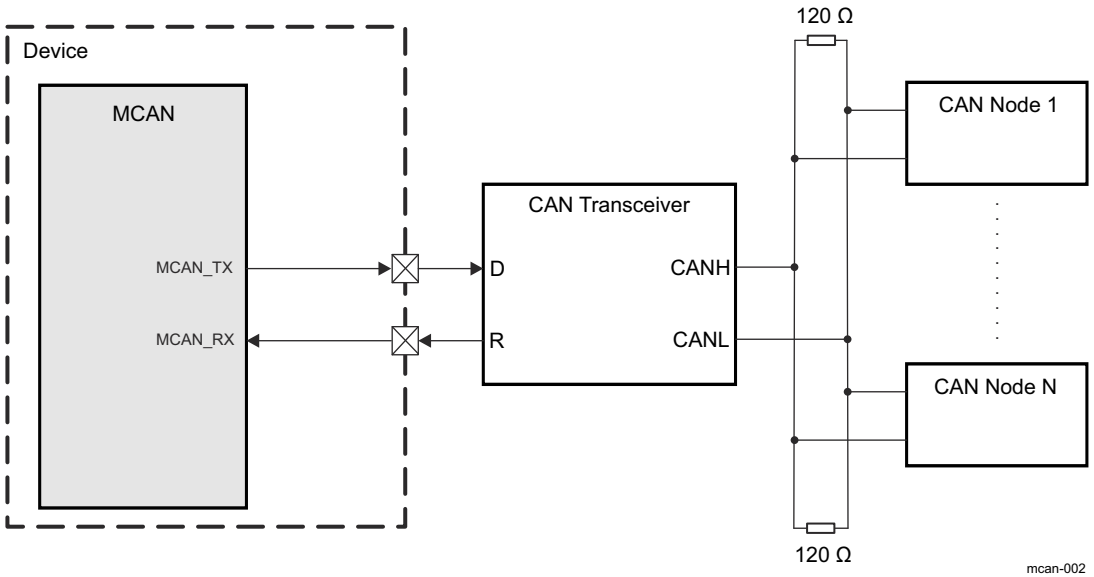


图 19-2. MCAN 典型总线布线

表 19-1. MCAN I/O 说明

模块信号	I/O	说明	复位时的值
MCAN_RX	输入	来自外部 CAN 收发器的串行数据输入。	HiZ
MCAN_TX	输出	连接到外部 CAN 收发器的串行数据输出。	HiZ

备注

请参阅器件数据表中的*端子配置和功能*部分以及*通用输入/输出 (GPIO)* 一章来配置要连接到器件引脚的该外设。

19.3 CAN 网络基础知识

网络基础知识包括：

- CAN 总线是使用非归零 (NRZ) 编码的 2 线制差分总线，有两种状态：
  - 隐性状态（逻辑 1）
  - 显性状态（逻辑 0）
- 当总线空闲时，任何节点都可以发起到任何其他节点的传输。当两个或多个节点 (ECU) 尝试同时发送时，无损仲裁技术会按优先级顺序发送消息，从而不会丢失任何消息。
- 消息传输采用多播形式。发送的数据消息基于标识符，而非基于地址。
- 消息的内容由在整个网络中唯一的标识符（例如 RPM、温度、位置、压力等）进行标记。
- 网络上的所有节点都会接收该消息，并且每个节点都会对标识符执行验收测试。如果消息是相关的，则对其进行处理；否则消息会被忽略。
- 唯一标识符还决定了消息的优先级（标识符的数值越低，优先级越高）。
- 数据使用消息帧进行发送和接收，消息帧由以下基本字段组成：
  - 仲裁字段
  - 控制字段
  - 数据字段（传统 CAN 最多 8 个字节，CAN FD 最多 64 个字节）
  - CRC 字段
  - 确认字段

有关更多信息，请参阅 *ISO 11898-1:2015：CAN 数据链路层和物理信令*。

MCAN 上电和时钟序列：

1. 从可用的时钟源中选择 CANCLK：HFCLK（HFXT 或 HFCLK_IN）或 SYSPLL (SYSPLLCLK1)。
2. 启用相应的时钟源。
3. 等待时钟源稳定（SYSCTL_CLKSTATUS[*GOOD] 位 = 1）。
4. 通过设置 MCAN 的 PWREN 来启用 MCAN 电源。
5. 等待 MCAN 就绪 (SYSCTL_SYSSTATUS[MCAN0READY] = 1)。

19.4 MCAN 功能说明

MCAN 模块根据 ISO 11898-1:2015 执行 CAN 协议通信。比特率可以编程为最大 8Mbit/s 的值。需要额外的收发器硬件来实现与物理层（CAN 总线）的连接。

对于 CAN 网络上的通信，可以配置单独的消息帧。消息帧和标识符掩码存储在消息 RAM 中。

在消息处理器中实施关于消息处理的所有功能。

可以通过模块接口直接访问 MCAN 模块的寄存器组。这些寄存器用于控制和配置 CAN 内核和消息处理程序，以及访问消息 RAM。

图 19-3 展示了 MCAN 模块方框图，后跟 MCAN 模块的各个块的说明。

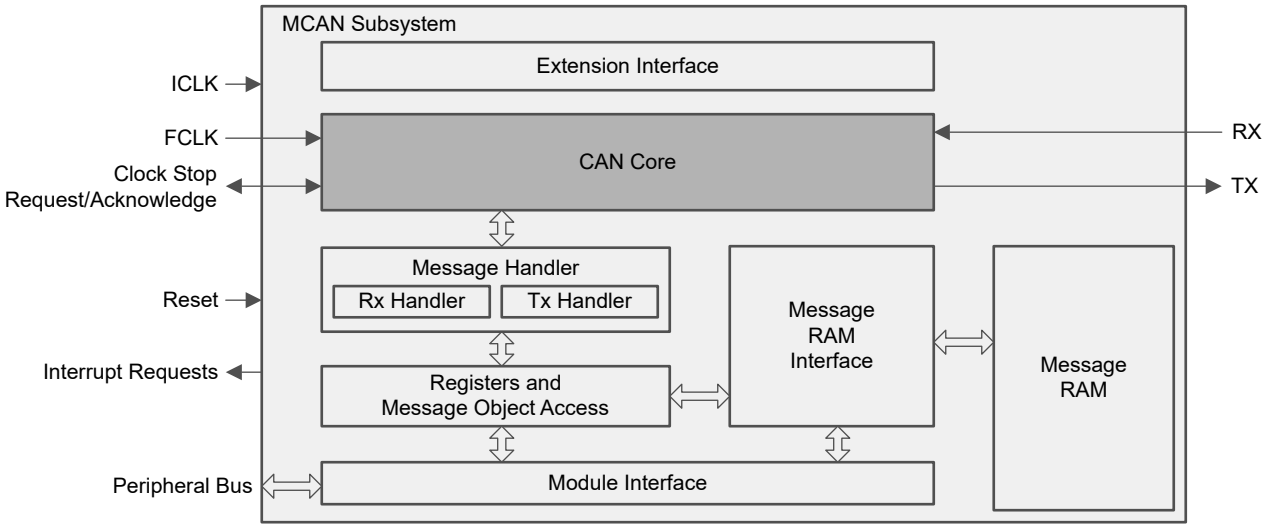


图 19-3. MCAN 方框图

- **CAN 内核**：CAN 内核由 CAN 协议控制器和 Rx/Tx 移位寄存器组成。该内核处理所有 ISO 11898-1:2015 协议功能并支持 11 位和 29 位标识符。
- **消息处理程序**：消息处理程序（Rx 处理程序和 Tx 处理程序）是一个状态机，用于控制单端口消息 RAM 和 CAN 内核的 Rx/Tx 移位寄存器之间的数据传输。它还按照控制寄存器中的编程处理接受过滤和中断生成。
- **消息 RAM**：消息 RAM 的主要用途是存储 Rx/Tx 消息、Tx 事件元素和消息 ID 过滤器元素（有关更多信息，请参阅节 19.4.19）。
- **消息 RAM 接口**：启用消息 RAM 和 MCAN 模块中其他块之间的连接。
- **寄存器和消息对象访问**：通过对消息对象的间接访问来确保数据一致性。在正常操作期间，所有软件和 DMA 对消息 RAM 的访问都是通过接口寄存器完成的。接口寄存器与消息 RAM 具有相同的字长。
- **模块接口**：MCAN 模块寄存器由用户软件通过 32 位外设总线接口进行访问。
- **计时**：向 MCAN 模块提供两个时钟：外设同步时钟（接口时钟 - MCAN_ICLK）和外设异步时钟（功能时钟 - MCAN_FCLK）。
- **扩展接口**：所有选定的内部状态和控制信号都路由到该接口（配置更改启用位 (MCAN_CCCR.CCE) 和中断寄存器位 (MCAN_IR) 的指示信号除外）。

19.4.1 时钟设置

可以通过 HFXT 或 SYSPLL 为 MCAN 的外设异步时钟 (MCAN_FCLK) 计时。必须通过 SYSCTL 寄存器来完成该时钟配置。*请参阅时钟 一节下的 CANCLK（CAN-FD 功能时钟）*。

19.4.2 模块时钟要求

为 MCAN 模块提供了两个时钟：

- 主机时钟：外设同步时钟 (MCAN_ICLK) 作为通用模块时钟源，以及
- CAN 时钟：为 CAN 内核提供外设异步时钟 (MCAN_FCLK)，用于生成 CAN 位时序。

在 MCAN 模块内部署了同步机制，可确保两个时钟域之间的数据传输安全。从主机时钟域到 CAN 时钟域的信号之间、相反方向的信号之间以及到主机时钟域的复位信号与到 CAN 时钟域的复位信号之间存在同步。

备注

MCAN_ICLK 必须始终高于或等于 MCAN_FCLK，才能实现稳定的 MCAN 模块功能：$f_{ICLK} \geq f_{FCLK}$

CAN-FD 支持更高的运行速度，因此比传统 CAN 具有更严格的时序要求。为了提高性能，TI 建议使用最低的 N 分频器值来维持系统的工作 PLL REF_CLK。较低的 N 分频器值会增加 PLL 的环路带宽，从而提高 CAN-FD 的时序裕度。

CAN-FD 支持更高的运行速度，因此比传统 CAN 具有更严格的时序要求。为了提高性能，TI 建议使用最低的 N 分频器值来维持系统的工作 PLL REF_CLK。较低的 N 分频器值会增加 PLL 的环路带宽，从而提高 CAN-FD 的时序裕度。可以通过 HFXT 或 SYSPLL 为外设异步时钟 (MCAN_FCLK) 计时。必须通过 SYSCTL 寄存器实现相关配置。有关更多信息，请参阅时钟模块

19.4.3 中断请求

MCAN 模块产生中断请求。它通过主机 CPU 进行配置。挂起模式可防止中断请求传播到主机 CPU。MCAN 内核有 2 条中断线路和 30 个内部中断源。每个源都可以配置为驱动两条中断线路之一。中断是“高电平”中断。MCAN 内核提供两个中断请求（MCANSS_INT0 和 MCANSS_INT1）。

有关更多信息，请参阅以下寄存器：

- 中断寄存器 (MCAN_IR)
- 中断启用 (MCAN_IE)
- 中断线路选择 (MCAN_ILS)
- 中断线路启用 (MCAN_ILE)

MCAN 模块支持外部时间戳计数器。外部时间戳计数器在翻转时会产生中断（请参阅节 19.4.12.1）。

有关更多信息，请参阅以下寄存器：

- 中断清除影子寄存器 (MCANSS_ICS)
- 中断原始状态寄存器 (MCANSS_IRS)
- 中断启用清除影子寄存器 (MCANSS_IECS)
- 中断启用寄存器 (MCANSS_IE)
- 中断启用状态寄存器 (MCANSS_IES)
- 中断结束寄存器 (MCANSS_EOI)
- 外部时间戳预分频器寄存器 (MCANSS_EXT_TS_PRESCALER)
- 外部时间戳未处理中断计数器寄存器 (MCANSS_EXT_TS_UNSERVICED_INTR_CNTR)

要清除 IRQ_INT0、IRQ_INT1 和 TS_WAKE 中断，请对 MCANSS_EOI 寄存器中所述的相应中断号的 EOI 位字段进行写入。

为中断（外部时间戳、中断 0/1）提供服务后，在 MCANSS_EOI 寄存器的相应位中写入“1”以清除中断。如果发生 ECC 中断，清除 ECC 中断源后，应用软件还必须向 EOI 寄存器 (MCANERR_SEC_EOI.EOI_WR/ CANERR_DED_EOI.EOI_WR) 写入“1”。有关更多信息，请参阅节 19.4.14.2。

IRQ 序列为：

1. 通过设置 IMASK 中的相应位来启用中断源之一。
2. 设置 MCANSS_IE 以启用 IRQ，为 line0 或 line1 设置 MCANSS_ILS。

3. 等待中断源被触发。
4. 中断被触发，应用程序跳转到 IRQ 服务例程 (ISR)。
5. 在 ISR 中，通过读取 IIDX 检查 IRQ 是否由预期源触发。读取 IIDX 会清除 IRQ 源，因此无需写入 ICLR 来清除 IRQ。

   由于 MCAN 的设计，需要在 ISR 中执行步骤 (6) 来清除 IRQ：
6. 向 MCAN_IR 写入 1 以清除中断。如果源是 IRQ_INT0、IRQ_INT1 或 TS_WAKE，还需要清除 MCANSS_EOI，否则将无法识别下一个 IRQ。

19.4.4 操作模式

下面的章节对这些工作模式进行了讨论。

19.4.4.1 正常运行

一旦 MCAN 模块初始化且 MCAN_CCCR.INIT 位重置为零，MCAN 模块就会将自身与 CAN 总线同步，从而为通信做好准备。通过接受过滤后，接收到的消息（包括消息标识符 (ID) 和数据长度代码 (DLC)）将存储到专用 Rx 缓冲区或 Rx FIFO 0/Rx FIFO 1 中。

对于要发送的消息，可以初始化或更新专用 Tx 缓冲区以及 Tx FIFO 或 Tx 队列。

备注

不支持接收到远程帧时自动传输。

19.4.4.2 CAN 传统

CAN 支持使用标准（11 位）或（29 位）标识符传输最多 8 字节的数据。

19.4.4.3 CAN FD 操作

CAN FD 标准允许发送扩展帧，单个帧中最多可发送 64 个数据字节，并且帧数据阶段的比特率更高，最高可达 8Mbps。CAN FD 标准引入了从一个比特率切换到另一个比特率的功能。扩展数据长度 (EDL)（如图 19-4 所示，表 19-2 对此进行了介绍）将数据长度设置为最多 8 或 64 个数据字节。比特率切换 (BRS) 指示是否启用两种比特率（数据阶段以与仲裁阶段不同的比特率发送）。

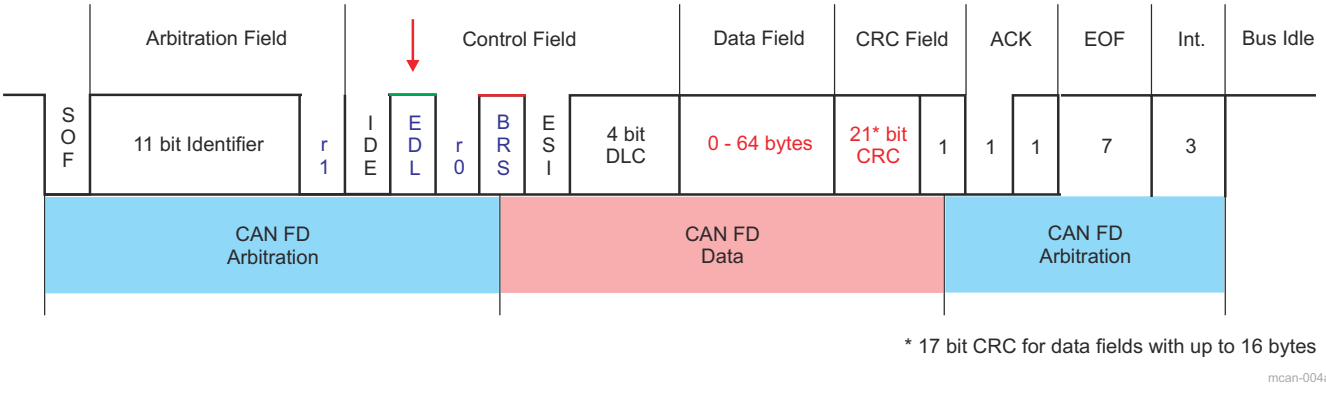


图 19-4. CAN FD 帧

表 19-2. CAN FD 帧说明

位	说明
SOF	帧起始
IDE	标识符扩展（针对 29 位扩展 ID）
FDF	灵活数据格式
BRS	比特率切换
ESI	错误状态指示器
DLC	数据长度码
CRC	循环冗余校验

CAN FD 帧传输有两种变体：

- 无比特率切换的 CAN FD 帧传输
- 控制字段、数据字段和 CRC 字段的发送比特率高于帧的开头和结尾的 CAN FD 帧发送

在 CAN 帧中，FDF = 隐性（逻辑 1）表示 CAN FD 帧，FDF = 显性（逻辑 0）表示传统 CAN 帧。在 CAN FD 帧中，FDF 之后的两个位（res 和 BRS）决定是否切换该 CAN FD 帧内的比特率。CAN FD 比特率切换由 res = 显性和 BRS = 隐性表示。请注意，res = 隐性的编码被保留用于未来扩展协议。如果 MCAN 模块接收到 FDF = 隐性且 res = 隐性的帧，则 MCAN 通过设置 MCAN_PSR.PXE 位来发出协议异常事件信号。当启用协议异常处理 (MCAN_CCCR.PXHD = 0) 时，这会使操作状态在下一个采样点从接收器 (MCAN_PSR.ACT = 10) 更改为集成 (MCAN_PSR.ACT = 00)。如果禁用协议异常处理 (MCAN_CCCR.PXHD = 1)，那么 MCAN 将隐性位视为错误并使用错误帧进行响应。

可以通过对 MCAN_CCCR.FDOE 位进行编程来启用 CAN FD 操作。如果 MCAN_CCCR.FDOE = 1，则启用 CAN FD 帧的传输和接收。对于传统 CAN 帧，始终可以执行传输和接收操作。可以使用相应 Tx 缓冲区元素中的 FDF 位来配置是发送 CAN FD 帧还是发送传统 CAN 帧。

当 MCAN_CCCR.FDOE = 0 时，接收到的帧被解释为传统 CAN 帧，这会导致在接收 CAN FD 帧时传输错误帧。当禁用 CAN FD 操作时，即使设置了 Tx 缓冲区元素的 FDF 位，也不会发送 CAN FD 帧。MCAN_CCCR.FDOE 和 MCAN_CCCR.BRSE 位只能在同时设置了 MCAN_CCCR.INIT 和 MCAN_CCCR.CCE 位时进行更改。当 MCAN_CCCR.FDOE = 0 时，FDF 和 BRS 位的设置被忽略，帧以传统 CAN 格式发送。

当 MCAN_CCCR.FDOE = 1 且 MCAN_CCCR.BRSE = 0 时，仅评估 Tx 缓冲区元素的 FDF 位。当 MCAN_CCCR.FDOE = 1 且 MCAN_CCCR.BRSE = 1 时，启用具有比特率切换的 CAN FD 帧传输。所有设置了位 FDF 和 BRS 的 Tx 缓冲区元素均以具有比特率切换的 CAN FD 格式发送。

仅在以下条件下建议在 CAN 操作期间更改模式：

- CAN FD 数据阶段的故障率明显高于 CAN FD 仲裁阶段的故障率。在这种情况下，应禁用传输的 CAN FD 比特率切换选项。
- 在系统启动期间，所有节点都发送传统 CAN 消息，直到证实节点能够以 CAN FD 格式进行通信。如果确实如此，则所有节点都会切换到 CAN FD 操作。
- CAN 局部联网中的唤醒消息必须以传统 CAN 格式发送。
- 下线编程，以防并非所有节点都支持 CAN FD。非 CAN FD 节点将保持静音模式，直到编程完成。然后，所有节点都切换回传统 CAN 通信。

采用 CAN FD 格式的 DLC 的编码与标准 CAN 格式不同。DLC 代码 0 至 8 与标准 CAN 中的编码相同（0 至 8 个数据字节），代码 9 至 15（在标准 CAN 中均编码 8 字节的数据字段）根据表 19-3 进行编码。

表 19-3. CAN FD 中的 DLC 编码

DLC	9	10	11	12	13	14	15
数据字节数。	12	16	20	24	32	48	64

对于 CAN FD 帧，如果 BRS（比特率切换）位为隐性位，则在该位之后在帧内切换位时序。在 CAN FD 仲裁阶段，在 BRS 位之前，按照标称位时序和预分频器寄存器 (MCAN_NBTP) 的配置使用标称 CAN 位时序（请参阅图 19-5）。在接下来的 CAN FD 数据阶段，按照数据位时序和预分频器寄存器 (MCAN_DBTP) 的配置使用数据阶段位时序。位时序在 CRC 定界符处或在检测到错误时（以先发生者为准）从数据阶段时序向回切换。

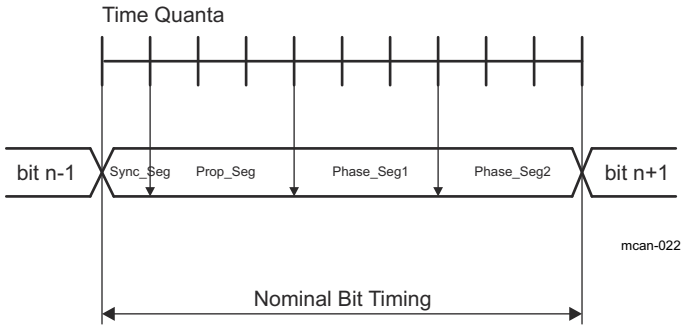


图 19-5. CAN 位时序

最大可配置数据阶段位时序取决于 CAN 时钟频率 (MCAN_FCLK)。示例：当 MCAN_FCLK = 20MHz 且最短可配置位时间为 $4t_q$（时间量子）时，数据阶段的比特率为 5Mbit/s。

在两种数据帧格式（CAN FD 和具有比特率切换的 CAN FD）中，错误状态指示符 (ESI) 位的值取决于发送开始时监视的发送器错误状态（请参阅 MCAN_PSR.RESI 位）。如果发送器具有错误被动标志，则 ESI 位以隐性方式发送；否则 ESI 位以显性方式发送。

19.4.5 软件初始化

当 MCAN_CCCR.init 位设置为 1 时，开始进行软件初始化。当在消息 RAM 中检测到未更正的位错误时，这可以通过软件或硬件复位来完成，或者通过进入 Bus_Off 状态来完成。当 MCAN_CCCR.INIT 位被置位时，消息传输停止并且输出 TX 引脚的状态为隐性（高电平）。错误管理逻辑 (EML) 的计数器保持不变。设置 MCAN_CCCR.INIT 位不会更改任何配置寄存器。重置 MCAN_CCCR.init 位将完成软件初始化。等待出现 11 个连续隐性位序列（指示 Bus_Idle 状态）后，消息传输开始。

仅当 MCAN_CCCR.INIT 和 MCAN_CCCR.CCE 位都被置位时，才能启用对 MCAN 配置寄存器的访问（写保护）。

仅当 MCAN_CCCR.INIT = 1 时，才能设置/重置 MCAN_CCCR.CCE 位。当 MCAN_CCCR.INIT 位被重置时，MCAN_CCCR.CCE 位会自动重置。

当设置 MCAN_CCCR.CCE 位时，以下寄存器被重置：

- MCAN_HPMS - 高优先级消息状态
- MCAN_RXF0S - Rx FIFO 0 状态
- MCAN_RFX1S - Rx FIFO 1 状态
- MCAN_TXFQS - Tx FIFO/队列状态
- MCAN_TXBRP - Tx 缓冲区请求待处理
- MCAN_TXBTO - 发生 Tx 缓冲区传输
- MCAN_TXBCF - Tx 缓冲区取消完成
- MCAN_TXEFS - Tx 事件 FIFO 状态

当设置 MCAN_CCCR.CCE 位时，超时计数器值 MCAN_TOCV.TOC 字段被预设为 MCAN_TOCC.TOP 字段配置的值。

此外，当 MCAN_CCCR.CCE = 1 时，Tx 处理程序和 Rx 处理程序保持空闲状态。

以下寄存器仅在 MCAN_CCCR.CCE = 0 时可写

- MCAN_TXBAR - Tx 缓冲区添加请求
- MCAN_TXBCR - Tx 缓冲区取消请求

当 MCAN_CCCR.INIT = 1 且 MCAN_CCCR.CCE = 1 时，MCAN_CCCR.TEST 和 MCAN_CCCR.MON 位只能通过主机 CPU 设置。这两个位随时可以重置。仅当 MCAN_CCCR.INIT = 1 且 MCAN_CCCR.CCE = 1 时，才能设置/重置 MCAN_CCCR.DAR 位。

表 19-4 展示了配置 MCAN 模块的步骤。

表 19-4. 配置 MCAN 模块的步骤

步骤	操作	说明	伪代码
1	初始化 MCAN_CCCR	设置 MCAN_CCCR.INIT 位 并检查该位是否已设置	INIT = 1; If INIT ≠ 1, wait until set
2	解锁受保护的寄存器	设置 MCAN_CCCR.CCE 位	CCE = 1;
3	配置 CAN 模式	将 MCAN_CCCR.FDOE 位设置为 CAN FD	FDOE = 1 for CAN FD FDOE = 0 for Classic CAN
4	配置比特率切换	设置 MCAN_CCCR.BRSE 位	BRSE = 1 for bit rate switching BRSE = 0 for no bit rate switching
5	设置标称位时序(1)	设置 MCAN_NBTP 寄存器	

表 19-4. 配置 MCAN 模块的步骤 (continued)

步骤	操作	说明	伪代码
6	锁定受保护的寄存器	清除 MCAN_CCCR.CCE 位	CCE = 0;
7	使 MCAN 模块恢复正常运行	清除 MCAN_CCCR.INIT 位 并检查该位是否已清除	INIT = 0; If INIT ≠ 0, wait until cleared

(1) 请参阅 *MCAN_REGS 寄存器* 部分中的 MCAN_NBTP 寄存器，了解如何对 CAN 位时序进行编程。

19.4.6 发送器延迟补偿

19.4.6.1 说明

当只有一个 CAN FD 节点正在发送且所有其他节点都是接收器时，总线长度没有影响。当使用 TX 引脚发送时，MCAN 模块使用 RX 引脚从其 CAN 收发器接收传输的数据。接收到的数据被延迟。如果发送器延迟大于 TSEG1（采样点之前的时间段），则检测到位错误。

MCAN 模块提供延迟补偿机制来补偿发送器延迟。该补偿机制能够在 CAN FD 数据阶段以更高的比特率进行传输，而与特定 CAN 收发器的延迟无关。如果没有发送器延迟补偿，数据阶段的比特率将受到发送器延迟的限制。

该机制支持数据位时间短于发送器延迟的配置（ISO 11898-1:2015 中进行了详细介绍）。通过将 MCAN_DBTP.TDC 位设置为 1 来启用发送器延迟补偿。

将延迟的传输数据与辅助采样点 (SSP) 处接收的数据进行比较，以检查发送节点数据阶段期间的位错误。如果检测到位错误，发送器将在下一个常规采样点对该位错误做出反应。在仲裁阶段，延迟补偿始终处于禁用状态。

在 SSP 处将接收到的位与发送的位进行比较。SSP 位置定义为从 MCAN 发送输出 TX 引脚通过收发器到接收输入 RX 引脚的测量延迟加上由 MCAN_TDCR.TDCO 字段配置的发送器延迟补偿偏移之和（请参阅图 19-6）。发送器延迟补偿偏移用于调整接收位内 SSP 的位置（例如：数据阶段中位时间的一半）。SSP 的位置向下舍入到下一个 mtq 整数。

可以通过读取 MCAN_PSR.TDCV 字段来检查实际发送器延迟补偿值。该字段在设置 MCAN_CCCR_INIT 位时被清除，并在设置 MCAN_DBTP.TDC 位时在每次传输 CAN FD 帧时更新。

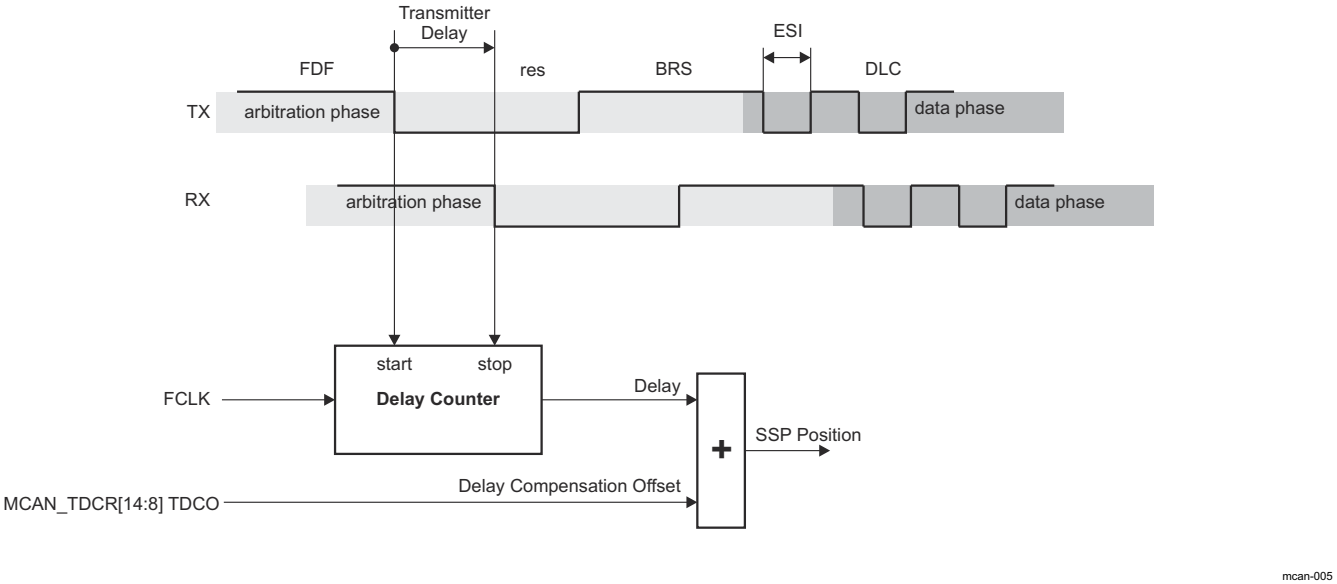


图 19-6. 发送器延迟测量

19.4.6.2 发送器延迟补偿测量

当启用发送器延迟补偿（通过编程 MCAN_DBTP.TDC = 1）时，会在每个发送的 CAN FD 帧内的 FDF 位至位 r0 的下降沿开始进行测量。当在发送器的接收输入 RX 引脚上看到该边沿时，测量停止。该测量的分辨率为 1mtq（请参阅图 19-6）。mtq（最小时间量子）大小等于 CAN 时钟周期 (MCAN_FCLK)。

可以通过对 MCAN_TDCR.TDCF 字段进行编程来启用发送器延迟补偿过滤器窗口。该过滤器功能定义了 SSP 位置的最小值，以避免接收到的 FDF 位内的显性干扰在接收到的 res 位的下降沿之前结束延迟补偿测量（从而导致过早采用 SSP 位置）的情况。对于发送器延迟测量，会忽略导致过早 SSP 位置的 RX 引脚上的显性边沿。当 SSP 位置至少为 MCAN_TDCR.TDCF 字段且 RX 引脚为低电平时，测量停止。

必须考虑以下边界条件：

- 测得的 TX 引脚到 RX 引脚的延迟和配置的发送器延迟补偿偏移（MCAN_TDCR.TDCO 字段）之和必须小于数据阶段的六位时间。
- 测得的 TX 引脚到 RX 引脚的延迟和配置的发送器延迟补偿偏移 (MCAN_TDCR.TDCO) 字段的总和必须小于或等于 127mtq。如果该总和超过 127mtq，则为发送器延迟补偿使用最大值 127mtq。
- 数据阶段在 CRC 定界符的采样点处结束，从而停止 SSP 处的接收位检查。

19.4.7 受限运行模式

在受限运行模式下，CAN 节点能够接收数据和远程帧并对有效帧进行确认，但节点不会发送数据帧、远程帧、活动错误帧或过载帧。如果出现错误情况或过载情况，则节点不会发送显性位；相反，节点等待出现总线空闲条件，以将自身重新与 CAN 通信同步。当 CAN 错误记录 (MCAN_ECR.CEL) 处于活动状态时，接收和发送错误计数器（MCAN_ECR.REC 和 MCAN_ECR.TEC）会被冻结。主机 CPU 可以通过设置 MCAN_CCCR.ASM 位将 MCAN 模块设置为受限运行模式。仅当 MCAN_CCCR.CCE 和 MCAN_CCCR.INIT 位均设置为 1 时，主机 CPU 才能随时设置该位。

当 Tx 处理程序无法及时从消息 RAM 读取数据时，会自动进入受限运行模式。要离开受限运行模式，主机 CPU 必须重置 MCAN_CCCR.ASM 位。该模式可用于需要适应不同 CAN 比特率的应用。在这种情况下，应用会测试不同的比特率，并在节点接收到有效帧后退出受限运行模式。

备注

受限运行模式不得与环回模式组合使用。

19.4.8 总线监控模式

可以通过将 MCAN_CCCR.MON 位设置为 1 来进入总线监控模式。在该模式（请参阅 ISO 11898-1:2015 *总线监控* 部分）下，MCAN 模块能够接收有效数据和远程帧，但无法开始传输。MCAN 模块仅在 CAN 总线上发送隐性位。如果 MCAN 模块需要发送显性位（确认位、过载标志、活动错误标志），则该位会在内部重新路由，以便 MCAN 模块监视该显性位，但 CAN 总线可能仍处于隐性状态。在总线监控模式下，MCAN_TXBRP 寄存器保持在复位状态。总线监控模式可用于分析 CAN 总线上的流量，而不会因显性位的传输而影响总线。图 19-7 显示了总线监控模式下 TX 和 RX 信号到 MCAN 模块的连接。

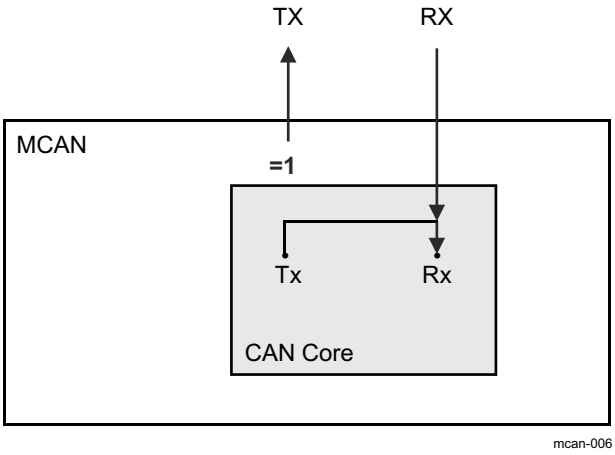


图 19-7. 总线监控模式下的信号连接

19.4.9 禁用自动重新传输 (DAR) 模式

根据 CAN 规范（请参阅 ISO11898-1:2015 *恢复管理* 部分），MCAN 模块提供了自动重新传输仲裁失败或在传输过程中受到错误干扰的帧的方法。默认情况下，启用自动重新传输（请参阅 MCAN_CCCR.DAR 位）。

19.4.9.1 DAR 模式下的帧传输

在 DAR 模式下，所有传输在 CAN 总线上启动后都会自动取消。在成功传输后，如果传输在取消点尚未开始、因仲裁丢失而中止或在帧传输期间发生错误，则 Tx 缓冲区的 Tx 请求挂起 (MCAN_TXBRP[xx]) TRPx 位将复位。

传输成功：

- 设置相应的 Tx 缓冲区传输已发生 (MCAN_TXBTO.TOx) TOx 位
- 不设置相应的 Tx 缓冲区取消已完成 (MCAN_TXBCF.CFx) CFx 位

传输成功但被取消：

- 设置相应的 Tx 缓冲区传输已发生 (MCAN_TXBTO.TOx) TOx 位
- 设置相应的 Tx 缓冲区取消已完成 (MCAN_TXBCF.CFx) CFx 位

仲裁丢失或帧传输受到干扰：

- 不设置相应的 Tx 缓冲区传输已发生 (MCAN_TXBTO.TOx) TOx 位
- 设置相应的 Tx 缓冲区取消已完成 (MCAN_TXBCF.CFx) CFx 位

在成功传输帧的情况下，如果启用了 Tx 事件的存储，则写入事件类型 ET = 10 的 Tx 事件 FIFO 元素（尽管取消，仍进行传输）。

19.4.10 时钟停止模式

可通过对以下两个位之一进行编程来进入时钟停止模式：

- MCAN_IP 中的时钟停止请求位 (MCAN_CCCR.CSR)
- MCAN_WRAPPER 中的停止请求位 (MCANSS_CLKCTL.STOPREQ)

只要 MCAN_WRAPPER 位 (MCANSS_CLKCTL.STOPREQ) 有效，MCAN_IP 中的寄存器位 (MCAN_CCCR.CSR) 读数就为 1。

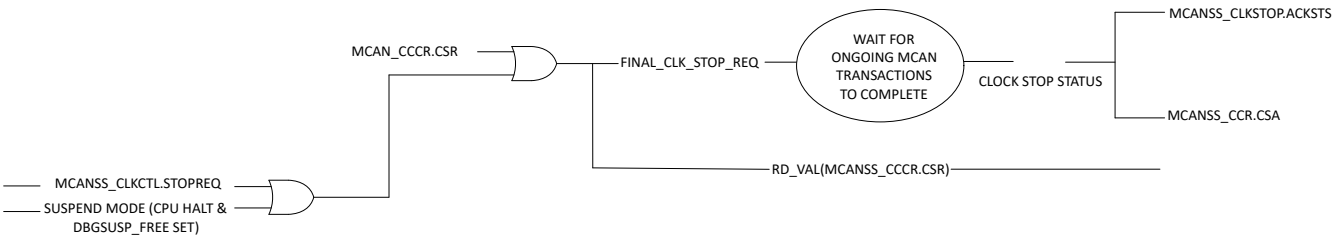


图 19-8. 时钟停止请求

所有挂起的传输请求完成后，MCAN 模块将等待，直到检测到总线空闲状态，然后将 MCAN_CCCR.INIT 设置为 1 以防止进一步的 CAN 传输。然后，MCAN 模块通过将时钟停止确认位 MCAN_CCCR.CSA 设置为 1（MCANSS_CLKSTS.CLKSTOP_ACKSTS 也反映相同的值）来确认 MCAN 已为断电做好准备。在该状态下，在时钟关闭之前，可以进一步访问寄存器。但是，对 MCAN_CCCR.INIT 位的写访问无效。现在可以使用 MCANSS_CLKEN.CLK_REQEN 关闭模块时钟输入 MCAN_ICLK 和 MCAN_FCLK。

要退出断电模式，应用程序必须先开启模块时钟，然后再清除时钟停止请求位（确保 MCAN_CCCR.CSR 和 MCANSS_CLKCTL.STOPREQ 都被清除）。MCAN 通过清除 MCANSS_CLKSTS.CLKSTOP_ACKSTS 和 MCAN_CCCR.CSA 位来确认时钟请求删除。应用程序现在可以通过清除 MCAN_CCCR.INIT 位来重新启动 CAN 通信。

通过 MCANSS_CTRL.AUTOWAKEUP 和 MCANSS_CTRL.WAKEUPREQEN 位支持自动从断电模式唤醒（由于 MCAN Rx 上的活动）（有关更多信息，请参阅节 19.4.10.2）。

图 19-9 展示了时钟停止和自动唤醒。

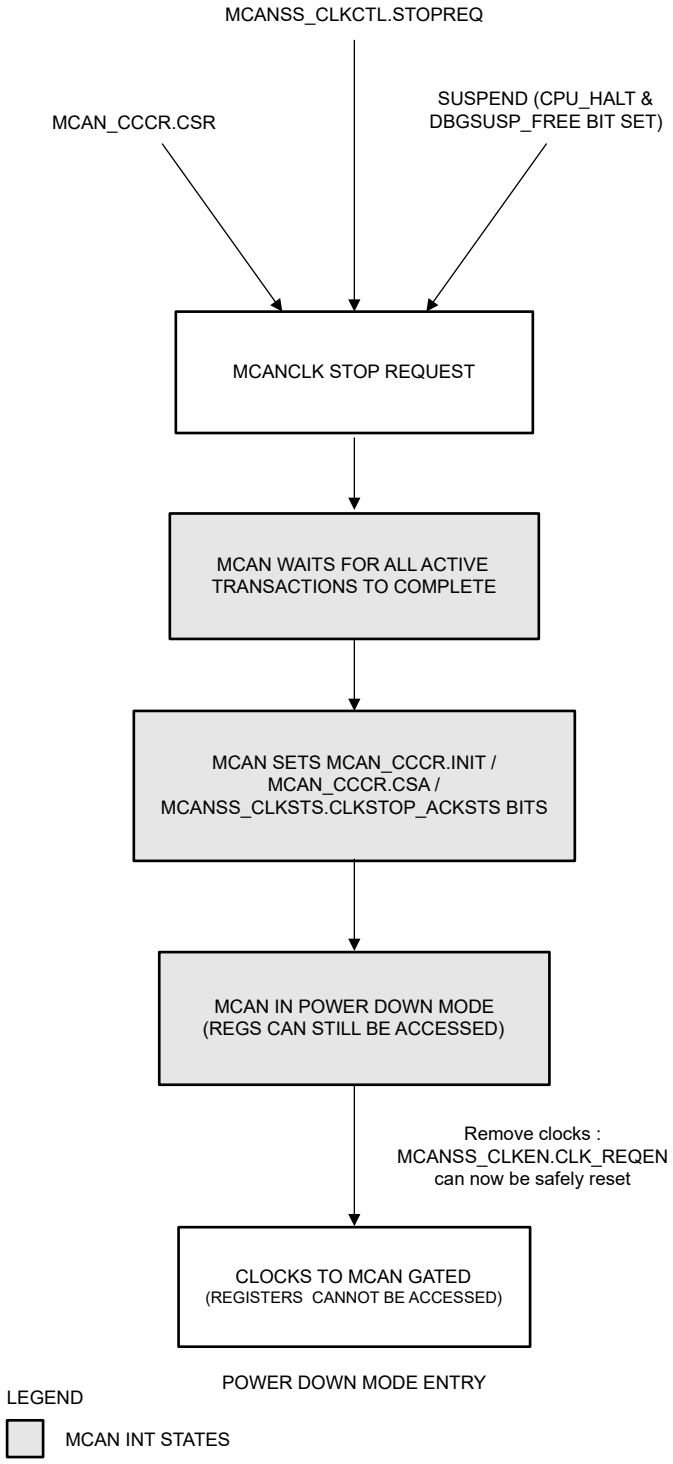


图 19-9. 断电进入

19.4.10.1 挂起模式

MCAN 模块支持两种挂起模式：

- 立即
- 正常

在正常挂起模式（请参阅 MCANSS_CTRL.DBGSUSP_FREE 位）下，当挂起请求有效时，会向 MCAN 内核发出时钟停止请求。当所有待处理的 Tx 消息均已得到处理且检测到空闲线路时，MCAN 内核会以时钟停止确认进行响应。此时会设置 MCAN_CCCR.INIT 位，MCAN 内核保持空闲状态。可以通过读取 MCAN_CCCR.INIT 位来验证挂起状态。

可以通过将 MCANSS_CTRL.AUTOWAKEUP 和 MCANSS_CTRL.WAKEUPREQEN 位设置为 1 来启用自动唤醒功能（有关更多信息，请参阅节 19.4.10.2）。当挂起请求被移除时，如果没有外部时钟停止请求处于活动状态，则会对 MCAN_CCCR.INIT 位执行读-修改-写操作以清除该位。

在挂起模式期间，会禁用自动清除功能。以下寄存器字段具有自动清除功能：

- MCAN_ECR.CEL
- MCAN_PSR.LEC
- MCAN_PSR.DLEC
- MCAN_PSR.RESI
- MCAN_PSR.RBRS
- MCAN_PSR.RFDF
- MCAN_PSR.PXE

19.4.10.2 唤醒请求

发出时钟停止请求会使 MCAN 模块进入断电模式（睡眠模式）。在从 IDLE 过渡到 ACTIVE 期间，如果启用 MCANSS_CTRL.AUTOWAKEUP 和 MCANSS_CTRL.WAKEUPREQEN 位，则在 MCAN 内核响应删除时钟停止请求并删除时钟停止确认后，会发出“读取-修改-写入”来清除 MCAN_CCCR.INIT 位，MCAN 内核会恢复运行。请注意，硬件删除时钟停止请求后，不会收到第一个帧（唤醒帧）。这是因为此操作是在发出时钟停止后发生的；IP 中没有运行活动时钟。因此，在删除时钟停止请求之后，必须重新发送启用时钟的唤醒帧。

如果设置了 MCANSS_CTRL.WAKEUPREQEN 位，则 MCAN 模块在发生以下唤醒事件时提供唤醒请求 ：

- 接收 RX 引脚为显性引脚（逻辑 0）

当发生以下任一情况时，唤醒请求无效：

- 时钟停止请求被删除并且时钟停止确认无效
- 向 MCAN 模块应用了复位

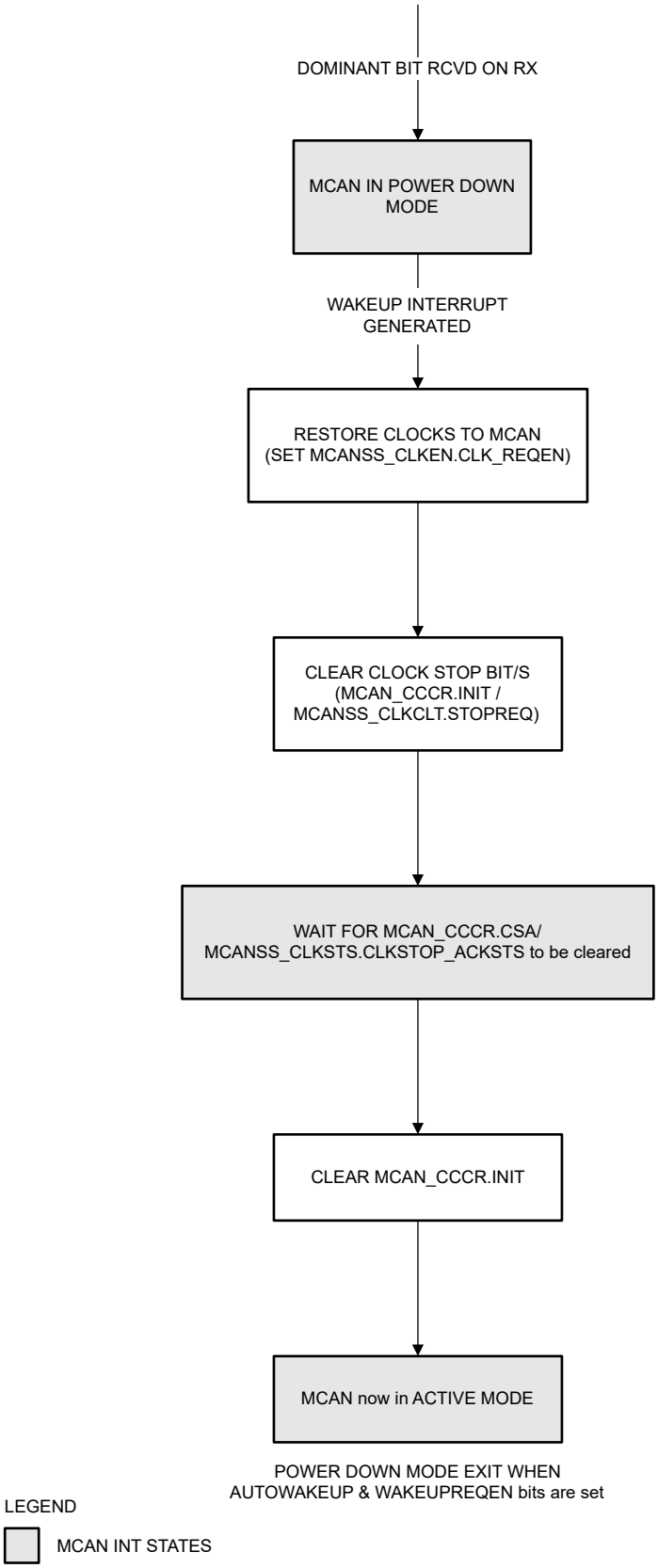


图 19-10. 因自动唤醒而从断电状态退出

请在发出唤醒请求时注意建议的用户说明。

- 如果在发出时钟停止请求 (CSR) 时 CAN 总线上正在接收一条消息，那么该消息会在接收结束时创建唤醒中断，从而使 MCAN 模块不进入时钟停止模式。为了避免不必要的唤醒，用户应在设置请求之前检查协议状态寄存器 (PSR) 中的节点活动位 (ACT)，以确保处于空闲状态。
- 用户还应避免创建仅通过停止时钟确认 (CSA) 退出的软件轮询循环，因为软件检查和唤醒中断清除 CSA 的时序可能会导致无限软件循环。
- 如果 Rx 引脚上的噪声过大，则可能会导致频繁唤醒。为了避免这种情况，建议用户等待指定的时间以查看是否接收到唤醒，如果没有，则使时钟停止状态生效。
- 如果在时钟停止请求有效时使 MCAN 复位，则唤醒逻辑可能会被意外禁用。因此，用户应避免在 CSR 有效时使 MCAN 软复位生效。

19.4.11 测试模式

通过将测试模式使能 MCAN_CCCR.TEST 位设置为 1 来启用 MCAN_TEST 寄存器写入权限。MCAN_TEST 寄存器允许配置测试模式和测试功能。

发送 (TX) 引脚有四个不同的输出功能，可通过对 MCAN_TEST.TX 字段进行编程来选择这些功能。默认功能为串行数据输出。该引脚也可由恒定显性或隐性值驱动。也可以驱动采样点信号来监控位时序。

接收 (RX) 引脚的实际值可从 MCAN_TEST.RX 位进行监控。这两个功能都可用于检查物理层。由于 CAN 时钟 (MCANx_FCLK) 和主机时钟 (MCANx_ICLK) 域之间的同步机制，在写入 MCAN_TEST.TX 字段直到在输出 TX 引脚上看到新配置可能要经历几个主机时钟周期的延迟。这也适用于通过 MCAN_TEST.RX 位读取输入 RX 引脚的情况。

备注

测试模式只能用于自检。TX 引脚的软件控制会干扰所有 CAN 协议功能。不建议对应用使用测试模式。

19.4.11.1 外部环回模式

通过将 MCAN_TEST.LBCK 编程为 1，可以将 MCAN 模块设置为外部环回模式。在环回模式下，MCAN 将其自己发送的消息视为接收的消息，并将其存储（如果它们通过接受过滤）到 Rx 缓冲区或 Rx FIFO 中，如下图所示。该图显示了外部环回模式下 TX 和 RX 引脚与 MCAN 模块的连接。

该模式用于硬件自检。为了独立于外部刺激，MCAN 模块在环回模式下忽略确认错误（在数据/远程帧的确认时隙中采样的隐性位）。在该模式下，MCAN 模块执行从 Tx 输出到 Rx 输入的内部反馈。RX 输入引脚的实际值被 MCAN 模块忽略。发送的消息在 TX 引脚上受到监测。

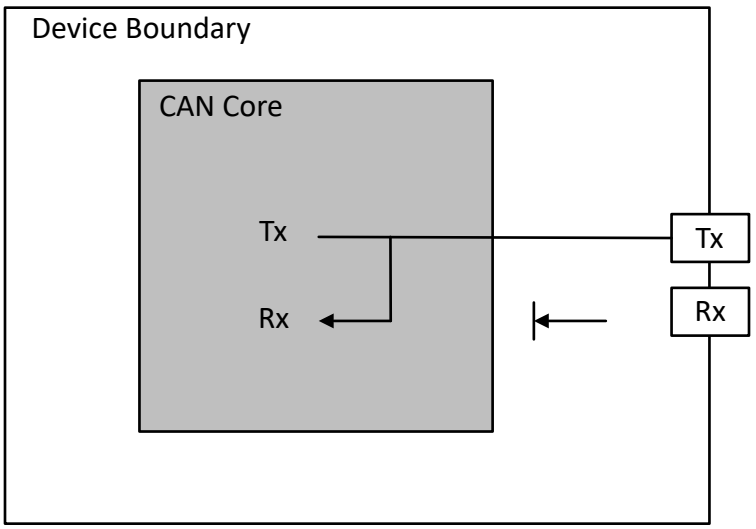


图 19-11. 外部环回模式

19.4.11.2 内部环回模式

通过将 MCAN_TEST.LBCK 和 MCAN_CCCR.MON 位编程为 1，可以将 MCAN 模块设置为内部环回模式。内部环回模式用于热自检。热自检允许在不影响连接到 TX 和 RX 引脚的正在运行的 CAN 系统的情况下测试 MCAN 模块。在该模式下，RX 引脚与 MCAN 模块断开，TX 引脚保持隐性。图 19-12 展示了内部环回模式下 TX 和 RX 引脚与 MCAN 模块的连接。

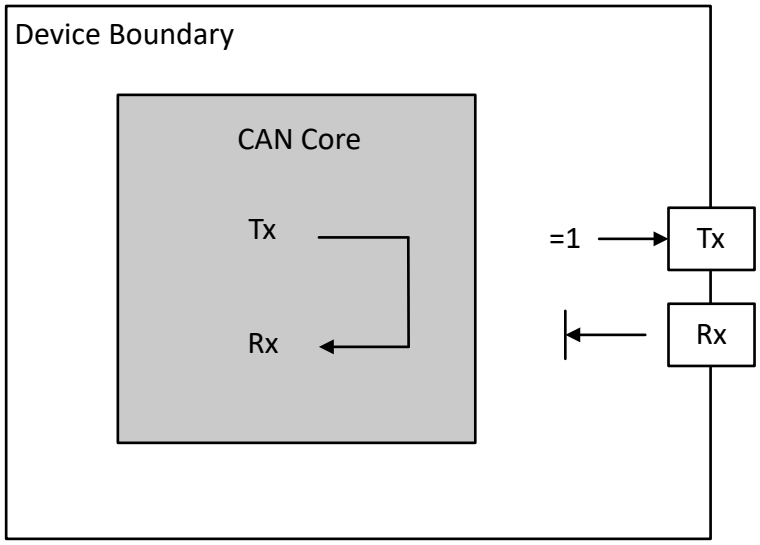


图 19-12. 内部环回模式

19.4.12 时间戳生成

MCAN 模块集成了一个用于时间戳生成的 16 位绕回计数器。时间戳计数器预分频器 MCAN_TSCC.TCP 字段可配置为以 CAN 位时间 (1-16) 的倍数对计数器进行计时。可以通过 MCAN_TSCV.TSC 字段对计数器进行读取。对 MCAN_TSCV 寄存器的写入访问会将计数器重置为零。当时间戳计数器绕回时，设置中断 MCAN_IR.TSW 标志。在帧接收/传输开始时，会捕获计数器值并将其存储到 Rx 缓冲区/Rx FIFO (RXTS[15:0]) 或 Tx 事件 FIFO (TXTS[15:0]) 元素的时间戳部分中。有关更多信息，请参阅节 19.4.19。

19.4.12.1 外部时间戳计数器

对于 CAN FD 运行模式，MCAN 内核需要一个外部时间戳计数器。外部生成的 16 位矢量可取代集成的 16 位 CAN 位时间计数器（内部时间戳计数器）来生成接收和发送时间戳。通过对 MCAN_TSCC.TSS 字段进行编程，可以使用外部 16 位时间戳计数器。

外部时间戳计数器使用接口时钟 (MCANx_ICLK) 作为参考时钟。MCAN 内核接受 16 位时间戳。24 位预分频器提供一个针对时间戳的可编程分辨率（请参阅 MCANSS_EXT_TS_PRESCALER.PRESCALER 位字段）。可通过 MCANSS_CTRL.EXT_TS_CNTR_EN 位启用或禁用外部时间戳计数器。禁用后，计数器被复位回零。启用后，计数器会不断递增。当时间戳回滚时，会生成 MCAN_IRQ_TS 中断。

当时间戳回滚时，设置 MCANSS_IRS 寄存器。通过写入 MCANSS_IESS 寄存器进行置位或写入 MCANSS_IECS 寄存器进行清零，MCANSS_IE 寄存器可能会受到影响。MCANSS_IESS 寄存器是映射到与 MCANSS_IE 寄存器相同的地址的影子寄存器。电平中断反映了所设置的 MCANSS_IRS 和 MCANSS_IE。MCANSS_IES 寄存器反映了电平中断。当发生回滚事件时，中断计数器会递增。写入 MCANSS_ICS 寄存器以清除 MCANSS_IRS 寄存器也会使中断计数器递减。如果中断计数器不为零，则写入 MCANSS_EOI 寄存器会发出另一个脉冲。

软件可以通过写入中断集影子寄存器 (MCANSS_ISS) 来人工模拟回滚事件。MCANSS_ISS 寄存器是映射到与 MCANSS_IRS 寄存器相同的地址的影子寄存器。

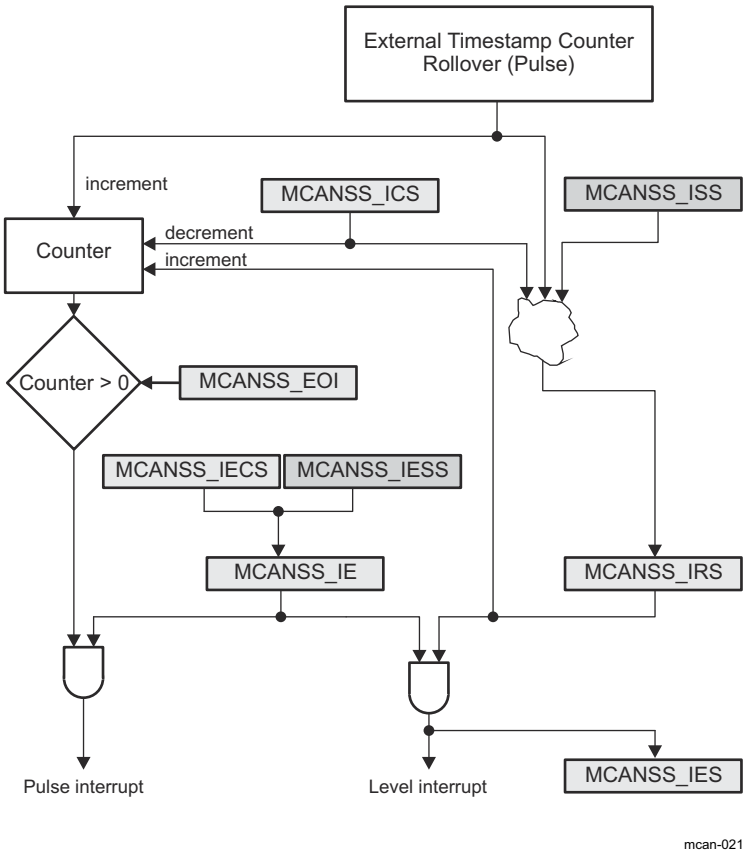


图 19-13. 外部时间戳计数器中断

19.4.13 超时计数器

MCAN 模块集成了一个 16 位超时计数器。超时计数器用于为 Rx FIFO 0、Rx FIFO 1 和 Tx 事件 FIFO 消息 RAM 元素指示超时状态。可以使用 MCAN_TOCC 寄存器配置超时计数器，使用 MCAN_TOCC.ETOC 位启用该计数器。超时计数器作为向下计数器运行，并使用由 MCAN_TSCC.TCP 字段编程的相同预分频器作为时间戳计数器。实际计数器值可以通过 MCAN_TOCV.TOC 字段进行监控。超时计数器只能在 MCAN_CCCR.INIT = 0 时启动，在 MCAN_CCCR.INIT = 1 时停止（例如当 MCAN 进入 Bus_Off 状态时）。可以通过 MCAN_TOCC.TOS 字段来选择运行模式。选择连续模式时，计数器在 MCAN_CCCR.INIT = 0 时启动，对 MCAN_TOCV 寄存器进行写入会将计数器预设为 MCAN_TOCC.TOP 字段配置的值，并继续向下计数。

如果超时计数器由 FIFO 之一控制，则空 FIFO 会将计数器预设为 MCAN_TOCC.TOP 字段配置的值。存储第一个 FIFO 元素时，开始向下计数。对 MCAN_TOCV 寄存器进行写入无效。当计数器达到零时，设置中断 MCAN_IR.TOO 标志。

在连续模式下，计数器立即以 MCAN_TOCC.TOP 字段配置的值重新启动。

备注

超时计数器的时钟信号来自 CAN 内核采样点信号。因此，超时计数器递减的时间点可能会因 CAN 内核的同步/重新同步机制而变化。如果使用 CAN FD 中的波特率切换功能，则超时计数器在仲裁和数据字段中的计时方式不同。

19.4.14 安全

消息存储器封装在可提供 SECDED 奇偶校验功能的 ECC 包装器中。ECC 包装器由 ECC 聚合器进行控制。

19.4.14.1 ECC 包装器

ECC 包装器为消息存储器内容提供单比特错误更正 (SEC) 和双比特错误检测 (DED) 奇偶校验。ECC 包装器具有用于错误通知的边带信号。ECC 包装器实现了错误注入测试模式。

错误校正是使用延迟写回完成的。当检测到错误时，会在 FIFO 队列中记录该错误，等待访问间隔以写入数据并刷新存储器。如果事务在延迟写回完成之前将新数据写入受损条目，则写回将被丢弃。

19.4.14.2 ECC 聚合器

本节介绍了 ECC 聚合器模块的功能详细信息。

19.4.14.2.1 ECC 聚合器概述

ECC 聚合器模块支持以下一般功能：

- 提供一种机制来控制和监视 MCAN 模块中的 ECC RAM。
- 提供对所有 ECC 相关寄存器的软件访问。
- 支持 ECC single-bit/double-bit 错误的软件可读状态以及相关信息，例如错误的 RAM 地址和数据位。
- 将 ECC RAM 的电平挂起状态聚合到主机 CPU 的单个中断。

不支持以下功能：

- 统计数据，例如跟踪 single-bit 和 double-bit 错误的数量。如果需要，这些操作可由软件进行处理。

19.4.14.2.2 ECC 聚合器寄存器

ECC 聚合器模块中有三组寄存器：

- **全局寄存器：**聚合器修订版本寄存器 (MCANERR_REV)、ECC 向量寄存器 (MCANERR_VECTOR)、其他状态寄存器 (MCANERR_STAT)、ECC 控制寄存器 (MCANERR_CTRL) 和 ECC 包装器修订版本寄存器 (MCANERR_WRAP_REV)。
- **控制和状态寄存器：**ECC 错误控制寄存器（MCANERR_ERR_CTRL1 和 MCANERR_ERR_CTRL2）和 ECC 错误状态寄存器（MCANERR_ERR_STAT1、MCANERR_ERR_STAT2 和 MCANERR_ERR_STAT3）。
- **中断寄存器：**作为标准中断模块的一部分的中断状态、中断使能设置、中断使能清除和 EOI（中断结束）寄存器。有关更多信息，请参阅以下寄存器：
  - MCANERR_SEC_EOI
  - MCANERR_SEC_STATUS
  - MCANERR_SEC_ENABLE_SET
  - MCANERR_SEC_ENABLE_CLR
  - MCANERR_DED_EOI
  - MCANERR_DED_STATUS
  - MCANERR_DED_ENABLE_SET
  - MCANERR_DED_ENABLE_CLR

19.4.14.3 读取 ECC 控制和状态寄存器

可以通过向 ECC 向量寄存器写入一条“读取消息”来触发对 ECC 控制和状态寄存器的读取，如下所示：

- 软件将值 (ECC RAM ID) 写入 MCANERR_VECTOR.ECC_VECTOR 字段，以选择用于控制或状态的 ECC RAM。
- 软件将 1 写入 MCANERR_VECTOR.RD_SVBUS 位以触发读取。
- 软件将读取地址写入 MCANERR_VECTOR.RD_SVBUS_ADDRESS 字段。
- 然后软件轮询 MCANERR_VECTOR.RD_SVBUS_DONE 位以检查该位是否为 1。该位表示读取操作已完成。
- 软件从 ECC 控制或状态寄存器读取数据。下一个时钟周期 (MCAN_ICLK) 返回读取的数据。

19.4.14.4 ECC 中断

ECC 聚合器模块将 ECC RAM 的电平挂起状态聚合到一个基于 EOI 握手的中断中，发送到主机 CPU。软件应遵循所述的顺序：

- 软件通过写入 MCANERR_SEC_ENABLE_SET/MCANERR_DED_ENABLE_SET 寄存器来启用 ECC RAM 的中断。
- 软件在 MCANERR_VECTOR.ECC_VECTOR 中写入 ECC RAM ID。
- 软件写入 MCANERR_VECTOR.RD_SVBUS 位来触发读取。
- 软件将 MCANERR_ERR_STAT1 寄存器地址写入 MCANERR_VECTOR.RD_SVBUS_ADDRESS 字段。如果软件需要读取 MCANERR_ERR_STAT2 寄存器，则需要再次在 rMCANERR_VECTOR 寄存器中加载“读取消息”。
- 软件轮询 MCANERR_VECTOR.RD_SVBUS_DONE 位。设置此位时，将执行 MCANERR_ERR_STAT1/MCANERR_ERR_STAT2 寄存器的读取。
- 在中断被处理后，软件通过写入 MCANERR_ERR_STAT1.CLR_ECC_SEC 或 MCANERR_ERR_STAT1.CLR_ECC_DED 位来清除中断状态，具体取决于 ECC 错误的类型。
- 软件轮询 MCANERR_ERR_STAT1 寄存器以验证状态位是否已被清除。
- 软件写入 MCANERR_SEC_EOI/MCANERR_DED_EOI 寄存器以清除中断。
- 清除 ECC 中断源后，应用软件还必须向 MCANERR_SEC_EOI.EOI_WR /MCANERR_DED_EOI.EOI_WR 位写入 1。

19.4.15 Tx 处理

Tx 处理程序用于处理 Tx 请求。该处理程序控制发送消息从专用 Tx 缓冲区、Tx FIFO 和 Tx 队列到 CAN 内核、Tx 事件 FIFO 的传输以及 Put 和 Get 索引操作。MCAN 模块支持多达 32 个 Tx 缓冲区。这些 Tx 缓冲区可配置为专用 Tx 缓冲区、Tx FIFO 或 Tx 队列，以及专用 Tx 缓冲区/Tx FIFO 或专用 Tx 缓冲区/Tx 队列的组合。对于每个 Tx 缓冲区元素，可以配置传统 CAN 或 CAN FD 传输模式。下表显示了消息传输的可能配置。

表 19-5. 消息传输配置

MCAN_CCCR 寄存器		缓冲区元素		帧传输
BRSE	**FDOE**	**FDF**	**BRS**	
将被忽略	0	将被忽略	将被忽略	传统 CAN
0	1	0	将被忽略	传统 CAN
0	1	1	将被忽略	无比特率切换功能的 CAN FD
1	1	0	将被忽略	传统 CAN
1	1	1	0	无比特率切换功能的 CAN FD
1	1	1	1	有比特率切换功能的 CAN FD

当 Tx 缓冲区请求待处理 (MCAN_TXBRP) 寄存器更新时，或者当传输开始时，Tx 处理程序开始扫描以检查优先级最高的待处理 Tx 请求。具有最低消息 ID 的 Tx 缓冲区具有最高优先级。

备注

AUTOSAR 需要至少三个 Tx 队列缓冲区以及发送取消支持。

19.4.15.1 传输暂停

传输暂停功能适用于 CAN 消息 ID 是特定的且无法轻易更改的 CAN 网络。这些消息 ID 可能具有比其他定义的消息 ID 更高的优先级，而在特定的应用中其相对优先级应该是相反的。这允许出现以下情况：一个 ECU 发送 CAN 消息突发，导致另一条 ECU CAN 消息延迟（暂停）。

可以通过 MCAN_CCCR.TXP 位来启用传输暂停功能。默认情况下禁用该位 (MCAN_CCCR.TXP = 0)。每次成功发送消息后，在下一次发送开始之前会暂停两个 CAN 位时间。这使网络中的其他 CAN 节点能够发送消息，即使其消息 ID 具有较低的优先级也可以发送。

19.4.15.2 专用 Tx 缓冲区

专用 Tx 缓冲区用于在主机 CPU 完全控制下的消息传输。有两个选项。

- 每个专用 Tx 缓冲区都配置有特定的消息 ID
- 两个或多个专用 Tx 缓冲区配置有相同的消息 ID。在这种情况下，首先发送缓冲区编号最低的 Tx 缓冲区

数据部分更新后，通过添加请求来请求传输。这是使用 MCAN_TXBAR[x]ARn 位（其中 x = 0 至 31）完成的。请求的消息在内部与来自可选 Tx FIFO 或 Tx 队列的消息进行仲裁，在外部与 CAN 总线上的消息进行仲裁，并根据其消息 ID 发送出去。

下表显示了 Tx 缓冲区、Tx FIFO 和 Tx 队列元素大小。专用 Tx 缓冲区在消息 RAM 中分配元素大小为 32 位的字。消息 RAM 中专用 Tx 缓冲区的起始地址是通过将从 0 到 31 的发送缓冲区索引 (MCAN_TXFQS.TFQP) × 元素大小添加到 Tx 缓冲区起始地址 MCAN_TXBC.TBSA 字段来计算的。

表 19-6. Tx 缓冲区、Tx FIFO、Tx 队列元素大小

MCAN_TXESC.TBDS	数据字段 [字节]	元素大小 [RAM 字]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

19.4.15.3 Tx FIFO

可以通过设置位 MCAN_TXBC.TFQM = 0 来配置 Tx FIFO 模式。存储在 Tx FIFO 中的消息从 Get 索引 MCAN_TXFQS.TFGI 字段引用的消息开始发送。每次传输后，Get 索引都会递增，直到 Tx FIFO 为空。Tx FIFO 空闲级别 MCAN_TXFQS.TFFL 字段指示可用的空闲 Tx FIFO 元素的数量。Tx FIFO 允许

从不同的 Tx 缓冲区发送具有相同消息 ID 的消息，其发送顺序为这些消息写入 Tx FIFO 的顺序。

新的发送消息必须从 Put 索引 MCAN_TXFQS.TFQP 字段引用的 Tx 缓冲区开始写入 Tx FIFO。每次添加请求 (MCAN_TXBAR[x] ARn = 1) 之后，Put 索引会递增至下一个空闲 Tx FIFO 元素。当 Put 索引达到 Get 索引 (MCAN_TXFQS.TFQP = MCAN_TXFQS.TFGI) 时，通过位 MCAN_TXFQS.TFQF = 1 来指示 Tx FIFO 已满状态。在这种情况下，在发送下一条消息并且 Get 索引已递增之前，不应再将任何消息写入 Tx FIFO。

请求的 Tx 缓冲区数量不应超过 Tx FIFO 空闲级别 MCAN_TXFQS.TFFL 字段指示的空闲 Tx 缓冲区数量。

如果取消由 Get 索引引用的 Tx 缓冲区的传输请求，则 Get 索引将递增到具有待处理传输请求的下一个 Tx 缓冲区，并且会重新计算 Tx FIFO 空闲级别 MCAN_TXFQS.TFFL 字段。如果将发送取消应用于任何其他 Tx 缓冲区，则 Get 索引和 FIFO 空闲级别保持不变。

Tx FIFO 元素在消息 RAM 中分配元素大小为 32 位的字。下一个可用（空闲）Tx FIFO 缓冲区的起始地址是通过将 Tx FIFO/队列 Put 索引 MCAN_TXFQS.TFQP（从 0 到 31）× 元素大小添加到 Tx 缓冲区起始地址 MCAN_TXBC.TBSA 字段来计算的。

19.4.15.4 Tx 队列

可以通过设置位 MCAN_TXBC.TFQM = 1 来配置 Tx 队列模式。存储在 Tx 队列中的消息从优先级最高的消息（最低消息 ID）开始发送。如果两个或多个队列缓冲区配置了相同的消息 ID，则首先传输缓冲区编号最小的队列缓冲区。

新的发送消息必须从 Put 索引 MCAN_TXFQS.TFQP 字段引用的 Tx 缓冲区开始写入 Tx FIFO。每个添加请求都会将 Put 索引循环递增到下一个空闲 Tx 缓冲区。如果出现 Tx 队列已满情况 (MCAN_TXFQS.TFQF = 1)，则 Put 索引无效，并且在至少一条请求的消息已发送出去或挂起的传输请求已取消之前，不应再将任何消息写入 Tx 队列。

应用程序可以使用 MCAN_TXBRP 寄存器来代替 Put 索引，并且可以将消息放置到任何 Tx 缓冲区中，而无需等待传输请求。

Tx 队列缓冲区在消息 RAM 中分配元素大小为 32 位的字。下一个可用（空闲）Tx 队列缓冲区的起始地址是通过将 Tx FIFO/队列 Put 索引 MCAN_TXFQS.TFQP（从 0 到 31）× 元素大小添加到 Tx 缓冲区起始地址 MCAN_TXBC.TBSA 字段来计算的。

19.4.15.5 混合专用 Tx 缓冲区/Tx FIFO

对于该组合，消息 RAM 中的 Tx 缓冲区段分为两部分：

- 专用 Tx 缓冲区：专用 Tx 缓冲区的数量由 MCAN_TXBC.NDTB 字段配置。
- Tx FIFO：分配给 Tx FIFO 的 Tx 缓冲区数量由 MCAN_TXBC.TFQS 字段配置。如果 MCAN_TXBC.TFQS 字段为空（零），则仅使用专用 Tx 缓冲区。

Tx 优先级划分：

- 扫描专用 Tx 缓冲区和最旧的挂起 Tx FIFO 缓冲区（由 MCAN_TXFQS.TFGI 字段引用）
- 具有最低消息 ID 的缓冲区获得最高优先级并接下来发送该缓冲区
- 下图显示了混合专用 Tx 缓冲区/Tx FIFO 示例。

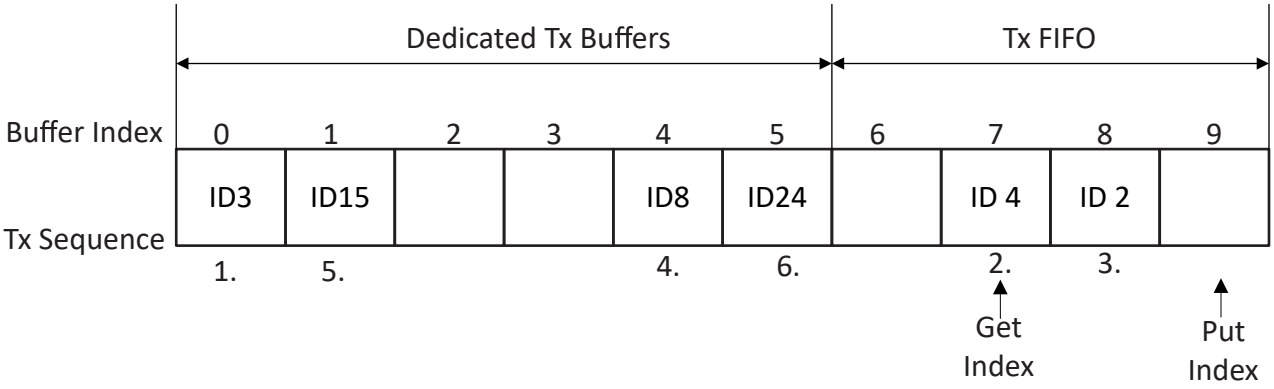


图 19-14. 混合专用 Tx 缓冲区/Tx FIFO（示例）

19.4.15.6 混合专用 Tx 缓冲区/Tx 队列

对于该组合，消息 RAM 中的 Tx 缓冲区段分为两部分：

- 专用 Tx 缓冲区：专用 Tx 缓冲区的数量由 MCAN_TXBC.NDTB 字段配置。
- Tx 队列：分配给 Tx 队列的 Tx 缓冲区数量由 MCAN_TXBC.TFQS 字段配置，如果 MCAN_TXBC.TFQS 字段为空（零），则仅使用专用 Tx 缓冲区。

Tx 优先级划分：

- 扫描所有具有激活传输请求的 Tx 缓冲区
- 具有最低消息 ID 的 Tx 缓冲区获得最高优先级并接下来发送该缓冲区
- 下图显示了混合专用 Tx 缓冲区/Tx 队列示例。

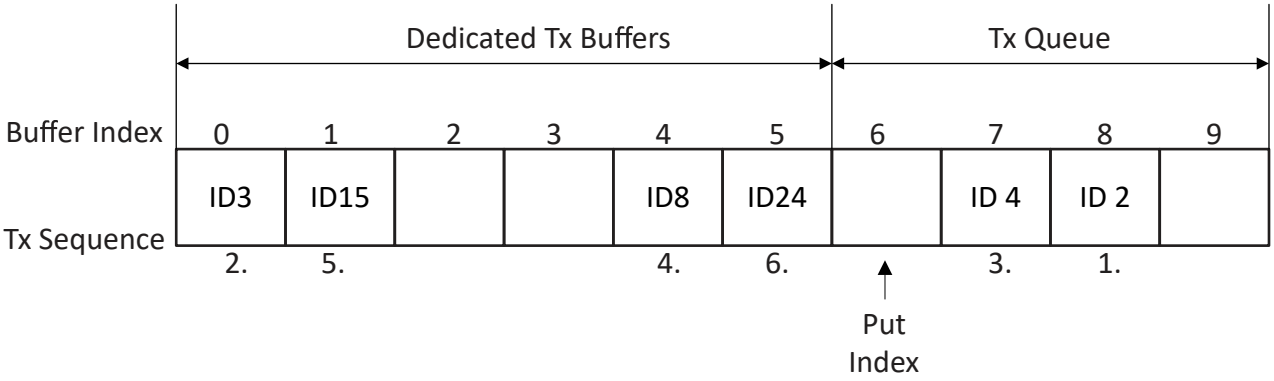


图 19-15. 混合专用 Tx 缓冲区/Tx FIFO（示例）

19.4.15.7 发送取消

该功能特别适用于网关和基于 AUTOSAR 的应用。主机 CPU 可以通过设置位 MCAN_TXBCR[n] CRn = 1（其中 n = 0 - 31）来取消来自专用 Tx 缓冲区或 Tx 队列缓冲区的请求传输。对应的位位置 n 相当于 Tx 缓冲区的编号。

发送消除不适用于 Tx FIFO 操作。

可以通过设置 MCAN_TXBCF 寄存器的相应位 (MCAN_TXBCF[n] CFn = 1) 来指示取消成功。

如果来自 Tx 缓冲区的传输已经在进行中并且已请求取消传输，则只要传输正在进行，相应的 MCAN_TXBRP[n] TRPn 位就会保持设置状态。如果传输成功，则会设置相应的 MCAN_TXBTO[n] TOn 和 MCAN_TXBCF[n] CFn 位。如果传输不成功，则仅相应的位 MCAN_TXBCF[n] CFn = 1。

备注

如果恰好在挂起的传输要开始之前取消该传输，则会出现一个短时间窗口，在该时间窗口中，即使该节点中还有另一条消息也处于挂起状态，也不会启动传输。这可以使另一个节点能够传输优先级比该节点中的第二条消息低的消息。

19.4.15.8 Tx 事件处理

为了支持 Tx 事件处理，消息 RAM 实现了 Tx 事件 FIFO 段。最多可配置 32 个 Tx 事件 FIFO 元素。请参阅 Tx 事件 FIFO 元素一章。在 CAN 总线上传输消息后，消息 ID 和时间戳存储在 Tx 事件 FIFO 元素中。为了将 Tx 事件链接到 Tx 事件 FIFO 元素，发送的 Tx 缓冲区中的消息标记被复制到 Tx 事件 FIFO 元素中。

Tx 事件 FIFO 已满状态由 MCAN_IR.TEFF 位进行指示。在这种情况下，在至少有一个元素被读出且 Tx 事件 FIFO Get 索引已递增 (MCAN_TXEFS.EFGI) 之前，不会再将任何元素写入 Tx 事件 FIFO。如果在 Tx 事件 FIFO 已满时发生 Tx 事件，则该事件会被拒绝，并且会设置中断标志 MCAN_IR.TEFL 位。

可以配置 Tx 事件 FIFO 水线，以避免 Tx 事件 FIFO 溢出。当 Tx 事件 FIFO 填充级别达到由 MCAN_TXEFC.EFWM 字段配置的 Tx 事件 FIFO 水线时，设置中断标志 MCAN_IR.TEFW。当从 Tx 事件 FIFO 中进行读取时，必须将两倍的 Tx 事件 FIFO Get 索引 MCAN_TXEFS.EFGI 字段添加到 Tx 事件 FIFO 起始地址 MCAN_TXEFC.EFSA 字段。

19.4.15.9 FIFO 确认处理

可以通过对相应的 FIFO 确认索引（请参阅 MCAN_RXF0A、MCAN_RXF1A 和 MCAN_TXEFA）进行写入来控制两个 Rx FIFO（Rx FIFO 0 或 Rx FIFO 1）和 Tx 事件 FIFO 的 Get 索引。对 FIFO 确认索引进行写入会将 FIFO Get 索引设置为 FIFO 确认索引加一，从而更新 FIFO 填充级别。

有两个用例：

- 已从 FIFO 读取单个元素：Get 索引值被写入 FIFO 确认索引。
- 已从 FIFO 读取元素序列：Get 索引值（最后读取的元素的索引）在该读取序列末尾写入 FIFO 确认索引。

主机 CPU 可以自由访问消息 RAM。以任意顺序读取 FIFO 元素时必须特别小心（不考虑 Get 索引）。当从两个 Rx FIFO 之一读取高优先级消息时，这非常有用。在这种情况下，不应写入 FIFO 的确认索引，因为这会将 Get 索引设置到错误的位置，并且还会更改 FIFO 的填充级别。在这种情况下，一些较旧的 FIFO 元素将会丢失。

备注

应用程序必须确保将有效值写入 FIFO 确认索引。MCAN 模块不会检查是否存在错误的值。

19.4.16 RX 处理

Rx 处理程序控制以下操作：

- 接受过滤
- 将接收到的消息传输到 Rx 缓冲区或两个 Rx FIFO 之一（Rx FIFO 0 或 Rx FIFO 1）
- Rx FIFO Put 和 Get 索引操作

19.4.16.1 接受过滤

MCAN 模块采用两组接受过滤器 - 一组用于标准标识符，一组用于扩展标识符。这些过滤器可以分配给 Rx 缓冲区或两个 Rx FIFO 之一。

过滤器元素的主要特性为：

- 每个过滤器元素可配置为：
  - 范围过滤器（从 - 至）
  - 针对特定 ID 的过滤器（用于一个或两个专用 ID）
  - 传统位掩码过滤器
- 可以单独启用/禁用每个过滤器元素
- 每个过滤器元素均可配置为接受或拒绝过滤
- 按顺序检查过滤器，在第一个匹配的过滤器元素处或到达过滤器列表末尾时停止执行（接受过滤过程）

相关的配置寄存器为：

- 全局过滤器配置 (MCAN_GFC) 寄存器
- 标准 ID 过滤器配置 (MCAN_SIDFC) 寄存器
- 扩展 ID 过滤器配置 (MCAN_XIDFC) 寄存器
- 扩展 ID 和屏蔽 (MCAN_XIDAM) 寄存器

根据过滤器元素配置（请参阅节 19.4.19 中的 SFEC/EFEC），如果过滤器匹配，则会执行以下操作之一：

- 接收到的帧存储在 FIFO 0 或 FIFO 1 中
- 接收到的帧存储在 Rx 缓冲区中
- 接收到的帧存储在 Rx 缓冲区中，并执行过滤器事件引脚上的脉冲生成。这是高电平单 MCAN_ICLK 脉冲。
- 接收到的帧被拒绝
- 设置高优先级消息中断标志 MCAN_IR.HPM
- 设置高优先级消息中断标志 MCAN_IR.HPM 并将接收到的帧存储在 FIFO 0 或 FIFO 1 中

当接收到完整的消息 ID 时，接受过滤开始。接受过滤在第一个匹配的已启用过滤器元素处或到达过滤器列表末尾时停止。如果过滤器元素匹配 - Rx 处理程序开始将接收到的消息数据以 32 位的形式写入到匹配的 Rx 缓冲区或 Rx FIFO 中。如果出现错误情况（例如：CRC 错误），该消息将被拒绝，并对受影响的 Rx 缓冲区或 Rx FIFO 产生以下影响：

- Rx 缓冲区：不设置匹配 Rx 缓冲区的新数据标志 (MCAN_NDAT1/MCAN_NDAT2)，但使用接收到的数据（部分）覆盖 Rx 缓冲区（有关错误类型，请分别参阅 MCAN_PSR.LEC 和 MCAN_PSR.DLEC 字段）。
- Rx FIFO：不更新匹配 Rx FIFO 的输入索引，但使用接收到的数据（部分）覆盖相关的 Rx FIFO 元素（有关错误类型，请分别参阅 MCAN_PSR.LEC 和 MCAN_PSR.DLEC 字段）。如果匹配 Rx FIFO 配置为在覆盖模式下运行，则必须考虑节 19.4.17.2 中介绍的边界条件。

备注

当接受的消息被写入两个 Rx FIFO 之一或 Rx 缓冲区时，未修改的接收标识符将独立于所使用的过滤器进行存储。接受过滤器过程的结果很大程度上取决于配置的过滤器元素的顺序。

19.4.16.1.1 范围过滤器

每个过滤器元素均可配置为作为范围过滤器运行（标准过滤器类型 SFT = 00/扩展过滤器类型 EFT = 00）。该过滤器匹配 ID 在 SFID1 至 SFID2 (SFID2 ≥ SFID1) 范围内或 EFID1 至 EFID2 (EFID2 ≥ EFID1) 范围内的所有已接收消息帧。有关更多信息，请参阅节 19.4.19.5 和节 19.4.19.6。

扩展帧的范围过滤有两个选项：

- 扩展过滤器类型 EFT = 00：扩展 ID 与掩码 (MCAN_XIDAM) 用于范围过滤。在应用范围过滤器之前，接收到的帧的消息 ID 与扩展 ID 与掩码 (MCAN_XIDAM) 进行与运算。
- 扩展过滤器类型 EFT = 11：扩展 ID 与掩码 (MCAN_XIDAM) 不用于范围过滤。

19.4.16.1.2 针对特定 ID 的过滤器

每个过滤器元素可配置为过滤一个或两个专用消息 ID（标准过滤器类型 SFT = 01/扩展过滤器类型 EFT = 01）。要仅过滤一个特定消息 ID，必须将过滤器元素配置为 SFID1 = SFID2 和 EFID1 = EFID2。有关更多信息，请参阅节 19.4.19.5 和节 19.4.19.6。

19.4.16.1.3 传统位掩码过滤器

传统位掩码过滤可以过滤消息 ID 组（标准过滤器类型 SFT = 10/扩展过滤器类型 EFT =10）。这是通过屏蔽接收消息 ID 的 single bit 来完成的。在这种情况下，SFID1/EFID1 元素用作消息 ID 过滤器，而 SFID2/EFID2 元素用作过滤器掩码。

过滤器掩码 (SFID2/EFID2) 处的 0 位屏蔽配置的消息 ID 过滤器 (SFID1/EFID1) 的相应位位置，该位位置处接收到的消息 ID 的值与接受过滤无关。仅接收到的消息 ID 中相应掩码位为 1 的那些位与接受过滤相关。

有以下两种有趣的情况：

- 所有掩码位均为 1：仅当接收到的消息 ID 与配置的消息 ID 过滤器相同时才会发生匹配。
- 所有屏蔽位都为 0：所有消息 ID 都匹配。

19.4.16.1.4 标准消息 ID 过滤

图 19-16 展示了标准消息 ID (11 位 ID) 过滤流。节 19.4.19.5 介绍了标准消息 ID 过滤器元素。

接收到的帧的远程传输请求 (RTR) 和扩展标识符 (XTD) 位与配置的过滤器元素列表进行比较。这是由以下寄存器控制的：

- 全局过滤器配置 (MCAN_GFC) 寄存器
- 标准 ID 过滤器配置 (MCAN_SIDFC) 寄存器

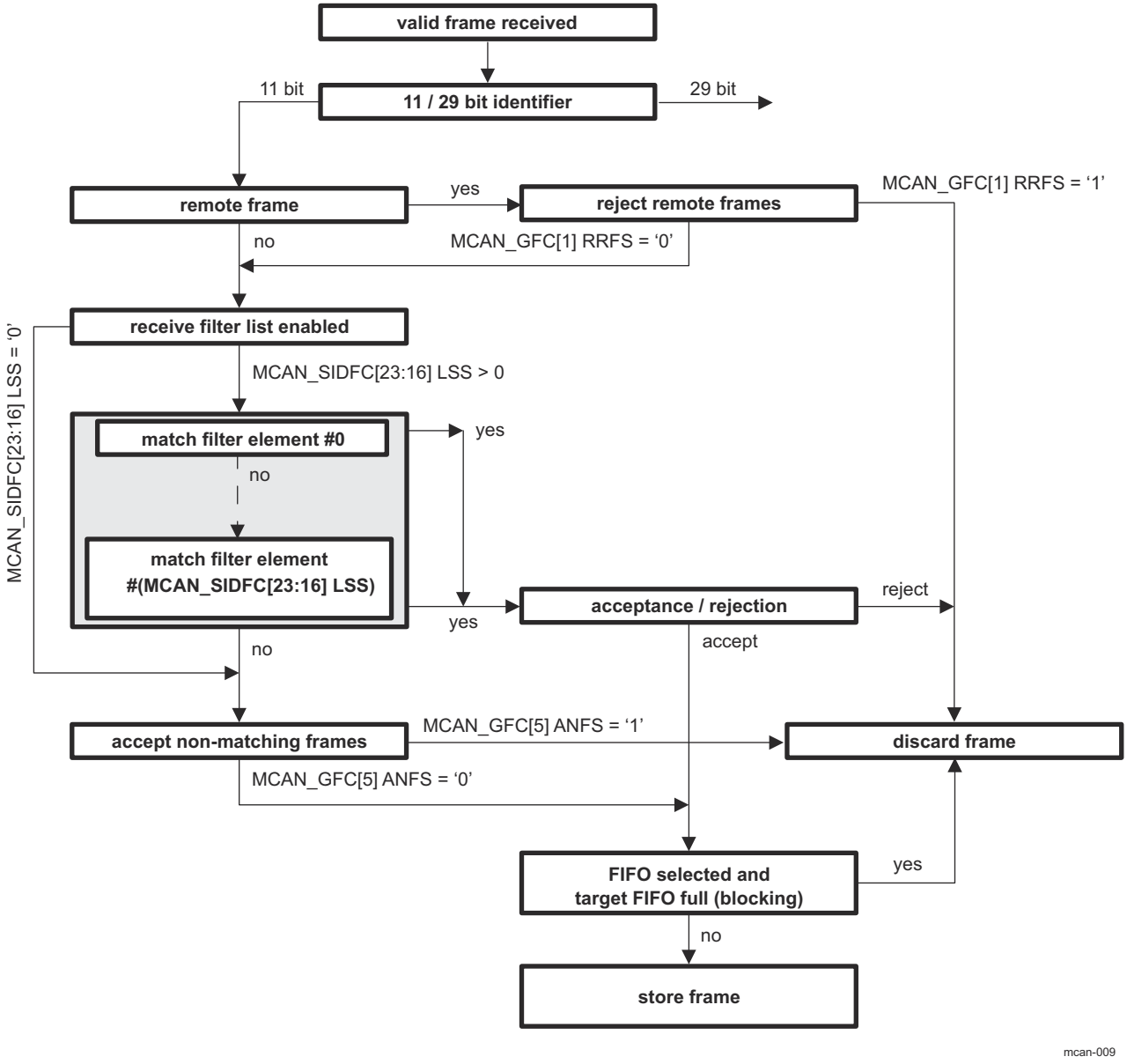


图 19-16. 标准消息 ID 过滤器路径

备注

在 MCAN_GFC[1]RRFS 和 MCAN_GFC[5]ANFS 中，[1] 和 [5] 分别表示 MCAN_GFC 寄存器中 RRFS 和 ANFS 位的位位置。

19.4.16.1.5 扩展消息 ID 过滤

图 19-17 展示了扩展消息 ID (29 位 ID) 过滤流。节 19.4.19.6 介绍了扩展消息 ID 过滤器元素。

接收到的帧的远程传输请求 (RTR) 和扩展标识符 (XTD) 位与配置的过滤器元素列表进行比较。这是由以下寄存器控制的：

- 全局过滤器配置 (MCAN_GFC) 寄存器
- 扩展 ID 过滤器配置 (MCAN_XIDFC) 寄存器

请注意，在执行过滤器列表之前，接收到的标识符与扩展 ID 与掩码 (MCAN_XIDAM) 进行与运算。

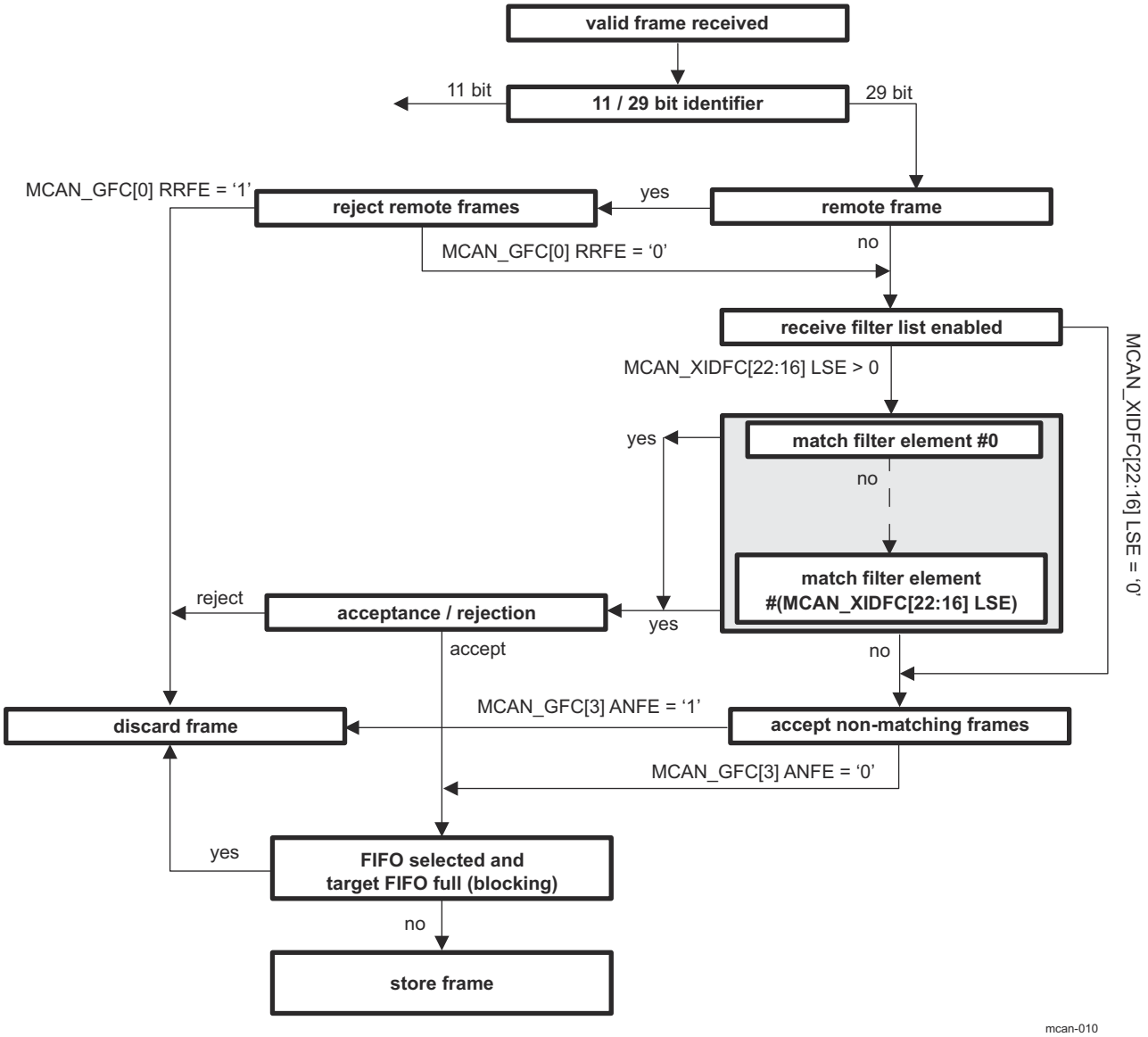


图 19-17. 扩展消息 ID 过滤器路径

在 MCAN_GFC[0]RRFS 和 MCAN_GFC[3]ANFS 中，[0] 和 [3] 分别表示 MCAN_GFC 寄存器中 RRFS 和 ANFS 位的位位置。

19.4.17 Rx FIFO

Rx FIFO（Rx FIFO 0 和 Rx FIFO 1）的配置可以通过 MCAN_RXF0C 和 MCAN_RXF1C 寄存器来完成。每个 Rx FIFO 可配置为存储最多 64 条接收到的消息。

在接受过滤后，传递的已接收消息将被传输到 Rx FIFO。节 19.4.16.1 介绍了 Rx FIFO 0 和 Rx FIFO 1 可用的过滤机制。节 19.4.19.2 介绍了 Rx FIFO 元素。

Rx FIFO 水线可用于防止 Rx FIFO 溢出。如果 Rx FIFO 填充级别达到由 MCAN_RXFnC[30:24] FnWM 字段（其中 n = 0 或 1）配置的 Rx FIFO 水线，则会设置中断标志 MCAN_IR.RF0W/MCAN_IR.RF1W。

当 Rx FIFO Put 索引达到 Rx FIFO Get 索引 (MCAN_RXFnS[21:16] FnPI = MCAN_RXFnS[13:8] FnGI) 时，MCAN_RXFnS[24] FnF 状态位会指示 Rx FIFO 已满状态，并且会设置中断标志 MCAN_IR.RF0F/MCAN_IR.RF1F。图 19-18 展示了 Rx FIFO 状态。FIFO 填充级别显示在 MCAN_RXFnS[6:0] FnFL 字段中（元素数量存储在 Rx FIFO 中）。

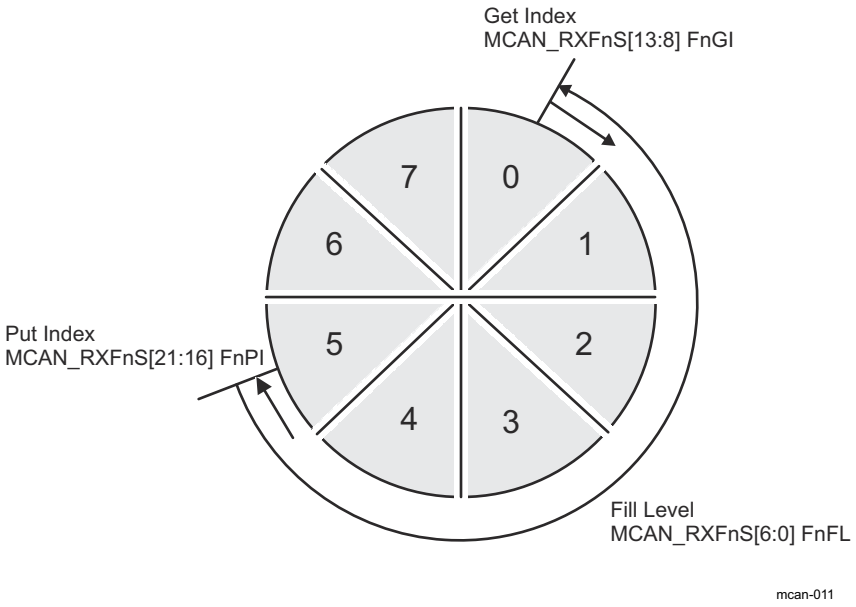


图 19-18. Rx FIFO 状态

从 Rx FIFO 中进行读取（Rx FIFO Get 索引 - MCAN_RXFnS[13:8] FnGI）时，必须配置消息 RAM 中的 Rx FIFO 起始地址（MCAN_RXFnC[15:2]FnSA 字段）。表 19-7 展示了不同 Rx 缓冲区/Rx FIFO 数据字段大小的 Rx 缓冲区/Rx FIFO 元素大小，可通过 MCAN_RXESC 寄存器对其进行配置。

表 19-7. Rx 缓冲区/Rx FIFO 元素大小

MCAN_RXESC 寄存器 RBDS/F0DS/F1DS 位	数据字段 [字节]	FIFO 元素大小 [RAM 字]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

19.4.17.1 Rx FIFO 阻塞模式

Rx FIFO 阻塞模式是 Rx FIFO 的默认工作模式，可以通过 MCAN_RXFnC[31] FnOM = 0 进行配置。

如果达到 Rx FIFO 已满状态 (MCAN_RXFnS[21:16] FnPI = MCAN_RXFnS[13:8] FnGI)，则不会再将任何消息写入相应的 Rx FIFO，直到至少一条消息被读出并且 Rx FIFO Get 索引已递增。Rx FIFO 已满状态由 MCAN_RXFnS[24] FnF = 1 进行指示，并且会设置中断标志 MCAN_IR.RF0F/MCAN_IR.RF1F。

如果在相应的 Rx FIFO 已满时接收到消息，则该消息被拒绝，消息丢失状态由 MCAN_RXFnS[25] RFnL = 1 进行指示，并且会设置中断标志 MCAN_IR.RF0L/MCAN_IR.RF1L。

19.4.17.2 Rx FIFO 覆盖模式

Rx FIFO 覆盖模式可通过 MCAN_RXFnC[31] FnOM = 1 进行配置。当达到由 MCAN_RXFnS[24] FnF = 1 指示的 Rx FIFO 已满状态 (MCAN_RXFnS[21:16] FnPI = MCAN_RXFnS[13:8] FnGI) 时，FIFO 的下一条已接受消息会覆盖最旧的 FIFO 消息。Put 索引/Get 索引都会递增 1。

在覆盖模式下，如果指示 Rx FIFO 已满状态，则至少从 Get 索引 + 1 处开始读取 Rx FIFO 元素。这是因为当主机 CPU 从消息 RAM 中读取 (Get 索引) 时，接收到的消息被写入消息 RAM (Put 索引)。在这种情况下，可能会从相应的 Rx FIFO 元素中读取不一致的数据。可以通过在从 Rx FIFO 中进行读取时向 Get 索引添加一个偏移来解决该问题。该偏移取决于主机 CPU 访问 Rx FIFO 的速度。图 19-19 展示了读取 Rx FIFO 时相对于 Get 索引的偏移 2。在这种情况下，存储在元素 1 和元素 2 中的两条消息会丢失。

从 Rx FIFO 进行读取后，必须将最后读取的元素的编号写入 Rx FIFO 确认索引 MCAN_RXFnA[5:0] FnAI。这会使 Get 索引递增至该元素编号。如果 Put 索引尚未递增至该 Rx FIFO 元素，则 Rx FIFO 已满状态被重置 (MCAN_RXFnS[24] FnF = 0)。

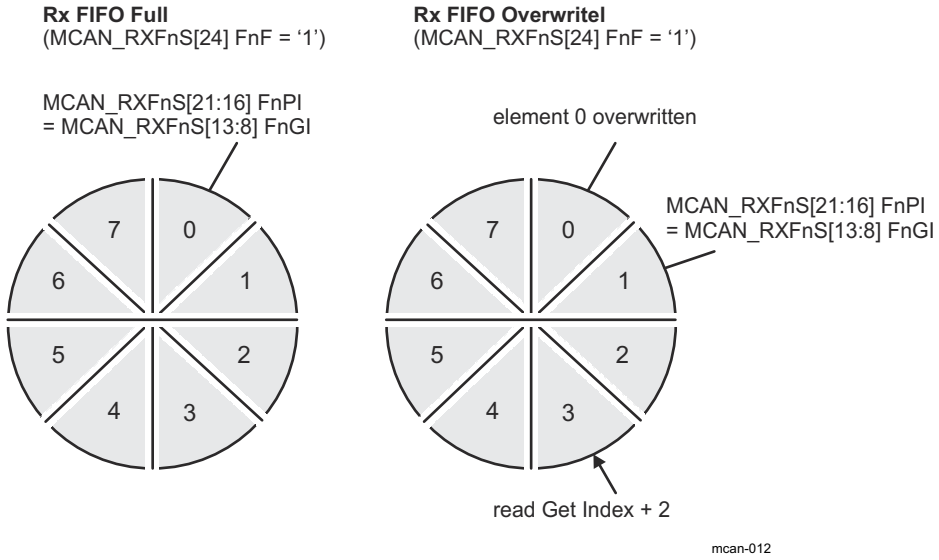


图 19-19. Rx FIFO 溢出处理

19.4.18 专用 Rx 缓冲区

MCAN 支持多达 64 个专用 Rx 缓冲区。消息 RAM 中 Rx 缓冲区部分的起始地址通过 MCAN_RXBC.RBSA 字段进行配置。要在 Rx 缓冲区中存储标准或扩展消息 ID，必须配置 SFEC/EFEC = 111 且 SFID2/EFID2[10:9] = 00 的过滤器元素（请参阅节 19.4.19.5 和节 19.4.19.6）。

过滤器元素接受接收到的消息后，该消息将存储到过滤器元素引用的消息 RAM 中的 Rx 缓冲区中（格式与 Rx FIFO 元素的格式相同）。此外，还会设置标志 MCAN_IR.DRX（存储在专用 Rx 缓冲区中的消息）。

表 19-8 显示了 Rx 缓冲区的过滤器配置示例。

表 19-8. Rx 缓冲区的过滤器配置示例

过滤器元素	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID 消息 1	00	00 0000
1	ID 消息 2	00	00 0001
2	ID 消息 3	00	00 0010

将匹配的接收消息的最后一个字写入消息 RAM 后，会设置寄存器 MCAN_NDAT1/MCAN_NDAT2 中相应的新数据标志。只要设置了新数据标志，相应的 Rx 缓冲区就会被锁定，以防止通过接收到的匹配帧进行更新。必须由主机 CPU 通过向相应位位置写入 1 来重置新数据标志。

当设置 Rx 缓冲区的新数据标志时，引用该特定 Rx 缓冲区的消息 ID 过滤器元素将不匹配，从而会继续进行接受过滤。遵循消息 ID 过滤器元素可能会使接收到的消息被存储到另一个 Rx 缓冲区或 Rx FIFO 中，或者消息可能被拒绝，具体取决于过滤器配置。

19.4.18.1 Rx 缓冲区处理

Rx 缓冲区处理包含以下步骤：

- 重置中断标志 MCAN_IR.DRX
- 读取新数据寄存器
- 从消息 RAM 中读取消息
- 重置已处理消息的新数据标志

19.4.19 消息 RAM

MCAN 模块有一个消息 RAM。消息 RAM 的主要用途是存储：

- 收到的消息
- 传输消息
- Tx 事件元素
- 消息 ID 滤波器元素

19.4.19.1 消息 RAM 配置

MCAN 模块可以具有不同的消息 RAM 大小。此处介绍了一个将 MCAN 模块配置为 1KB 大小、32 位宽度的示例。

消息 RAM 可以包含消息 RAM 配置中列出的每个段。没有必要配置每个段（消息 RAM 中的某个段可以是 0），并且对于各个段的顺序没有限制。对于奇偶校验或 ECC，必须在每个字中添加相应的位数。当 MCAN 模块对消息 RAM 进行寻址时，该模块会对 32 位字进行寻址。起始地址是可配置的，它们是 32 位字地址。

可以针对以下各项配置元素大小：

- RX FIFO 0，通过 MCAN_RXESC.F0DS 字段进行配置
- RX FIFO 1，通过 MCAN_RXESC.F1DS 字段进行配置
- Rx 缓冲区，通过 MCAN_RXESC.RBDS 字段进行配置
- Tx 缓冲区，通过 MCAN_TXESC.TBDS 字段进行配置

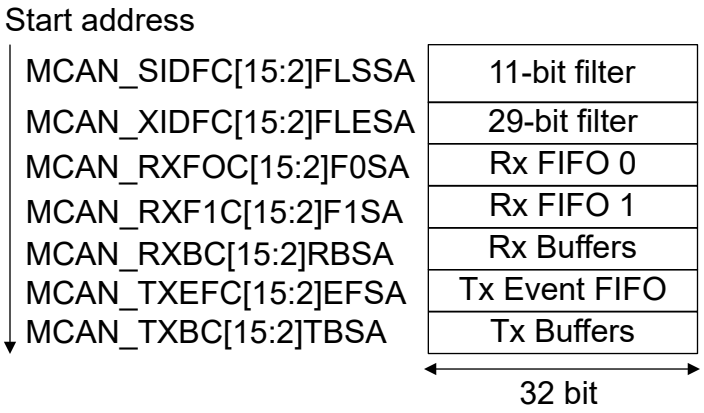


图 19-20. 消息 RAM 配置

主机 CPU 在消息 RAM 中配置以下信息：

- 各个存储器段的起始地址
- 每个段中的元素数量
- 某些段中元素的大小

备注

MCAN 模块不会检查消息 RAM 配置中是否存在错误。必须仔细完成不同段的起始地址和每个段的元素数量配置。这可以防止数据伪造或丢失。

消息传输配置 表提供了基于 Rx FIFO 的 1kB RAM 大小的每个段可容纳的最大元素数量的示例。

表 19-9. 消息传输配置

帧大小，DLC 代码（字节）	元素大小（字）	最大元素数
8	4	64
12	5	51
16	6	43
20	7	37
24	8	32
32	10	26
48	14	18
64	18	14

19.4.19.2 Rx 缓冲区和 FIFO 元素

消息 RAM 中最多可配置 64 个 Rx 缓冲区和两个 Rx FIFO。每个 Rx FIFO 段可配置为存储最多 64 条接收到的消息。可以通过 MCAN_RXESC 寄存器配置元素大小，以存储具有最多 64 字节数据字段的 CAN FD 消息。

图 19-21 展示了 Rx 缓冲区/Rx FIFO 元素结构。表 19-10 展示了 Rx 缓冲区/Rx FIFO 元素字段说明。

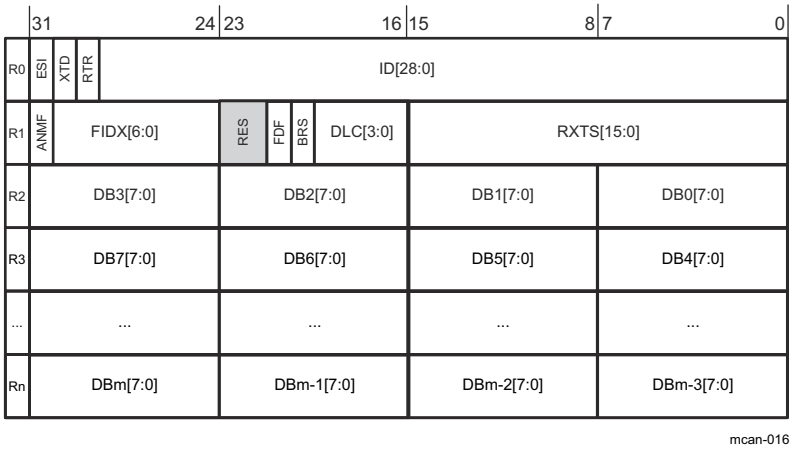


图 19-21. Rx 缓冲区/Rx FIFO 元素结构

表 19-10. Rx 缓冲区/Rx FIFO 元素字段说明

字	位	字段名称	说明
R0	31	ESI	错误状态指示器 • 0x0：发送节点是错误主动节点 • 0x1：发送节点是错误被动节点
	30	XTD	扩展标识符 向主机 CPU 发出信号，指示接收到的帧具有标准标识符还是扩展标识符。 • 0x0：11 位标准标识符 • 0x1：29 位扩展标识符
	29	RTR	远程传输请求 向主机 CPU 发出信号，指示接收到的帧是数据帧还是远程帧。 • 0x0：接收到的帧是数据帧 • 0x1：接收到的帧是远程帧 **注意：**没有采用 CAN FD 格式的远程帧。如果接收到 CAN FD 帧 (FDF = 1)，则 RTR 位反映保留的 r1 位 (RES[23]) 的状态。在 CAN FD 帧 (FDF=1) 中，显性 RRS（远程请求替代）位取代了 RTR（远程传输请求）位。
	28:0	ID[28:0]	标识符 标准或扩展标识符，具体取决于 XTD 位。标准标识符存储在 ID[28:18] 中。

表 19-10. Rx 缓冲区/Rx FIFO 元素字段说明 (continued)

字	位	字段名称	说明
R1	31	ANMF	接受的不匹配帧 可以使用 MCAN_GFC.ANFS 和 MCAN_GFC.ANFE 字段来启用非匹配帧接受。 • 0x0：接收到的帧与过滤器索引 FIDX 字段相匹配 • 0x1：接收到的帧与任何 Rx 过滤器元素都不匹配
	30:24	FIDX[6:0]	过滤器索引 0x0-0x7F (0-127)：匹配 Rx 接受过滤器元素的索引（如果 ANMF = 1，则无效）。 范围为 0 至 MCAN_SIDFC.LSS - 1 或 MCAN_XIDFC.LSE - 1。
	23:22	RES	保留
	21	FDF	FD 格式 • 0x0：标准帧格式 • 0x1：CAN FD 帧格式（新的 DLC 编码和 CRC）
	20	BRS	比特率切换 • 0x0：在无比特率切换的情况下接收帧 • 0x1：在有比特率切换的情况下接收帧
	19:16	DLC[3:0]	数据长度码 • 0x0-0x8 (0-8)：CAN + CAN FD：接收到的帧具有 0-8 个数据字节 • 0x9-0xF (9-15)：CAN：接收到的帧具有 8 个数据字节 • 0x9-0xF (9-15)：CAN FD：接收到的帧具有 12/16/20/24/32/48/64 个数据字节
	15:0	RXTS[15:0]	Rx 时间戳 在帧接收开始时捕获时间戳计数器值。分辨率取决于时间戳计数器预分频器 MCAN_TSCC.TCP 的配置。
R2	31:24	DB3[7:0]	数据字节 3
	23:16	DB2[7:0]	数据字节 2
	15:8	DB1[7:0]	数据字节 1
	7:0	DB0[7:0]	数据字节 0
R3	31:24	DB7[7:0]	数据字节 7
	23:16	DB6[7:0]	数据字节 6
	15:8	DB5[7:0]	数据字节 5
	7:0	DB4[7:0]	数据字节 4
...	...	...	...
Rn	31:24	DBm[7:0]	数据字节 m
	23:16	DBm-1[7:0]	数据字节 m-1
	15:8	DBm-2[7:0]	数据字节 m-2
	7:0	DBm-3[7:0]	数据字节 m-3

注意：根据元素大小 (MCAN_RXESC) 的配置，使用二至十六个 32 位字 (Rn = 3-17) 来存储一条 CAN 消息的数据字段。

19.4.19.3 Tx 缓冲区元素

Tx 缓冲区部分可配置为保存专用 Tx 缓冲区以及 Tx FIFO/Tx 队列。如果 Tx 缓冲区部分由专用 Tx 缓冲区和 Tx FIFO/Tx 队列共享，则专用 Tx 缓冲区从 Tx 缓冲区部分的开头开始，后跟分配给 Tx FIFO 或 Tx 队列的缓冲区。Tx 处理程序通过 MCAN_TXBC.TFQS 和 MCAN_TXBC.NDTB 字段区分专用 Tx 缓冲区和 Tx FIFO/Tx 队列。可以通过 MCAN_TXESC 寄存器配置元素大小，以存储具有最多 64 字节数据字段的 CAN FD 消息。

图 19-22 展示了 Tx 缓冲区元素结构。表 19-11 展示了 Tx 缓冲区元素字段说明。

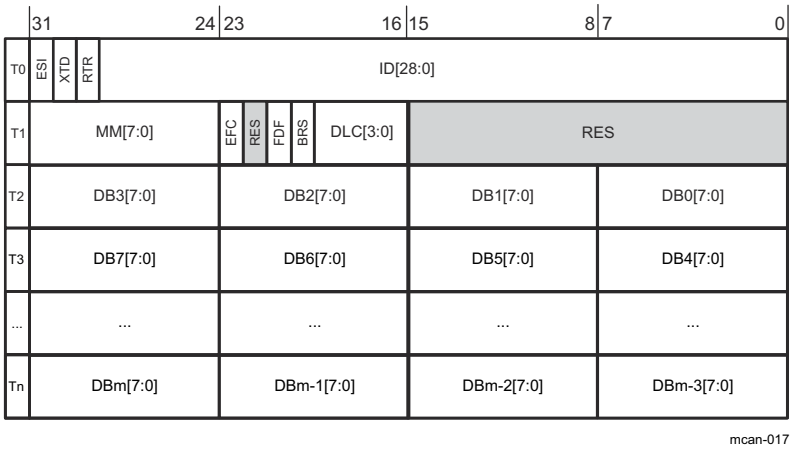


图 19-22. Tx 缓冲区元素结构

表 19-11. Tx 缓冲区元素字段说明

字	位	字段名称	说明
T0	31	ESI	错误状态指示器 • 0x0：CAN FD 格式的 ESI 位仅取决于错误被动标志 • 0x1：CAN FD 格式的 ESI 位隐性发送 **注意：**发送缓冲区的 ESI 位与错误被动标志进行或运算，以决定发送的 CAN FD 帧中 ESI 位的值。根据 CAN FD 协议规范的要求，错误主动节点可以选择性地发送隐性 ESI 位，但错误被动节点始终发送隐性 ESI 位。
	30	XTD	扩展标识符 • 0x0：11 位标准标识符 • 0x1：29 位扩展标识符
	29	RTR	远程传输请求 • 0x0：发送数据帧 • 0x1：发送远程帧 **注意：**当 RTR = 1 时，即使 MCAN_CCCR.FDOE 位启用 CAN FD 格式的传输，MCAN 模块也会根据 ISO11898-1:2015 发送远程帧。
	28:0	ID[28:0]	标识符 标准或扩展标识符，具体取决于 XTD 位。必须将标准标识符写入 ID[28:18]。

表 19-11. Tx 缓冲区元素字段说明 (continued)

字	位	字段名称	说明
T1	31:24	MM[7:0]	消息标记 在 Tx 缓冲区配置期间由主机 CPU 写入。复制到 Tx 事件 FIFO 元素中以标识 Tx 消息状态（另请参阅表 19-12 中的 MM[7:0] 字段）。
	23	EFC	事件 FIFO 控制 • 0x0：不存储 Tx 事件 • 0x1：存储 Tx 事件
	22	RES	保留
	21	FDF	FD 格式 • 0x0：以传统 CAN 格式发送的帧 • 0x1：以 CAN FD 格式发送的帧
	20	BRS	比特率切换 • 0x0：在无比特率切换的情况下发送 CAN FD 帧 • 0x1：在有比特率切换的情况下发送 CAN FD 帧 **注意：**仅当使用 MCAN_CCCR.FDOE 位启用 CAN FD 操作时，才会评估 ESI、FDF 和 BRS 位。仅当 MCAN_CCCR.BRSE = 1 时才会评估 BRS 位。
	19:16	DLC[3:0]	数据长度码 • 0x0-0x8 (0-8)：CAN + CAN FD：发送帧具有 0-8 个数据字节 • 0x9-0xF (9-15)：CAN：发送帧有 8 个数据字节 • 0x9-0xF (9-15)：CAN FD：发送帧具有 12/16/20/24/32/48/64 个数据字节
	15:0	RES	保留
T2	31:24	DB3[7:0]	数据字节 3
	23:16	DB2[7:0]	数据字节 2
	15:8	DB1[7:0]	数据字节 1
	7:0	DB0[7:0]	数据字节 0
T3	31:24	DB7[7:0]	数据字节 7
	23:16	DB6[7:0]	数据字节 6
	15:8	DB5[7:0]	数据字节 5
	7:0	DB4[7:0]	数据字节 4
...	...	...	...
Tn	31:24	DBm[7:0]	数据字节 m
	23:16	DBm-1[7:0]	数据字节 m-1
	15:8	DBm-2[7:0]	数据字节 m-2
	7:0	DBm-3[7:0]	数据字节 m-3

备注

根据元素大小 (MCAN_TXESC) 的配置，使用二至十六个 32 位字 (Tn = 3-17) 来存储一条 CAN 消息的数据字段。

19.4.19.4 Tx 事件 FIFO 元素

每个元素存储有关已发送消息的信息。通过读取 Tx 事件 FIFO，主机 CPU 按照消息发送的顺序获取该信息。可以从 MCAN_TXEFS 寄存器获取有关 Tx 事件 FIFO 的状态信息。

图 19-23 展示了 Tx 事件 FIFO 元素结构。表 19-12 展示了 Tx 事件 FIFO 元素字段说明。

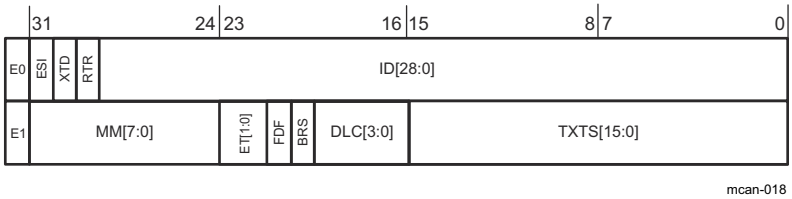


图 19-23. TX 事件 FIFO 元素结构

表 19-12. Tx 事件 FIFO 元素字段说明

字	位	字段名称	说明
E0	31	ESI	错误状态指示器 • 0x0：发送节点是错误主动节点 • 0x1：发送节点是错误被动节点
	30	XTD	扩展标识符 • 0x0：11 位标准标识符 • 0x1：29 位扩展标识符
	29	RTR	远程传输请求 • 0x0：发送数据帧 • 0x1：发送远程帧
	28:0	ID[28:0]	标识符 标准或扩展标识符，具体取决于 XTD 位。必须将标准标识符写入 ID[28:18]。

表 19-12. Tx 事件 FIFO 元素字段说明 (continued)

字	位	字段名称	说明
E1	31:24	MM[7:0]	消息标记 从 Tx 缓冲区复制到 Tx 事件 FIFO 元素以标识 Tx 消息状态（另请参阅表 19-11 中的 MM[7:0] 字段）。
	23:22	ET[1:0]	活动类型 • 0x0：保留 • 0x1：Tx 事件 • 0x2：尽管取消，仍进行传输（始终为 DAR 模式下的传输设置） • 0x3：保留
	21	FDF	FD 格式 • 0x0：标准帧格式 • 0x1：CAN FD 帧格式（新的 DLC 编码和 CRC）
	20	BRS	比特率切换 • 0x0：在无比特率切换的情况下发送帧 • 0x1：在有比特率切换的情况下发送帧
	19:16	DLC[3:0]	数据长度码 • 0x0-0x8 (0-8)：CAN + CAN FD：发送的帧具有 0-8 个数据字节 • 0x9-0xF (9-15)：CAN：发送的帧具有 8 个数据字节 • 0x9-0xF (9-15)：CAN FD：发送的帧具有 12/16/20/24/32/48/64 个数据字节
	15:0	TXTS[15:0]	Tx 时间戳 在帧发送开始时捕获时间戳计数器值。分辨率取决于时间戳计数器预分频器 MCAN_TSCC.TCP 字段的配置。

19.4.19.5 标准消息 ID 过滤器元素

最多可以为 11 位标准 ID 配置 128 个过滤器元素。当访问标准消息 ID 过滤器元素时，元素地址是过滤器列表标准起始地址 MCAN_SIDFC.FLSSA 字段加上过滤器元素的索引 (0-127)。

图 19-24 展示了标准消息 ID 过滤器元素结构。表 19-13 展示了标准消息 ID 过滤器元素字段说明。

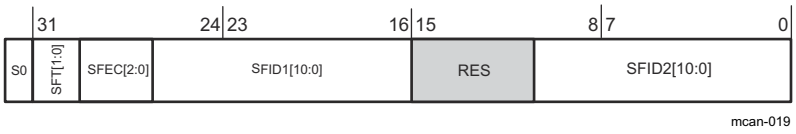


图 19-24. 标准消息 ID 过滤器元素结构

表 19-13. 标准消息 ID 过滤器元素字段说明

字	位	字段名称	说明
S0	31:30	SFT[1:0]	标准过滤器类型 • 0x0：从 SFID1 到 SFID2 的范围过滤器 (SFID2 ≥ SFID1) • 0x1：用于 SFID1 或 SFID2 的双 ID 过滤器 • 0x2：传统过滤器：SFID1 = 过滤器；SFID2 = 掩码 • 0x3：禁用过滤器元素 **注意：**当 SFT = 11 时，过滤器元素被禁用，并且接受过滤继续执行 (与 SFEC = 000 时的行为相同)
	29:27	SFEC[2:0]	标准过滤器元素配置 所有启用的过滤器元素都用于标准帧的接受过滤。接受过滤在第一个匹配的已启用过滤器元素处或到达过滤器列表末尾时停止。如果 SFEC = 100、101 或 110，则匹配设置中断标志 MCAN_IR.HPM，如果启用，则会生成中断。在这种情况下，MCAN_HPM 寄存器将更新为优先级匹配的状态。 • 0x0：禁用过滤器元素 • 0x1：如果过滤器匹配，则存储在 Rx FIFO 0 中 • 0x2：如果过滤器匹配，则存储在 Rx FIFO 1 中 • 0x3：如果过滤器匹配，则拒绝 ID • 0x4：如果过滤器匹配，则设置优先级 • 0x5：如果过滤器匹配，则设置优先级并存储在 FIFO 0 中 • 0x6：如果过滤器匹配，则设置优先级并存储在 FIFO 1 中 • 0x7：存储到 Rx 缓冲区中，忽略 SFT[1:0] 的配置
	26:16	SFID1[10:0]	标准过滤器 ID 1 当过滤 Rx 缓冲区时，该字段定义要存储的标准消息的 ID。接收到的标识符必须完全匹配，不使用掩码机制。
	15:11	RES	保留
	10:0	SFID2[10:0]	标准过滤器 ID 2 根据 SFEC 的配置，该位字段具有不同的含义： • SFEC = 001 - 110：标准 ID 过滤器元素的第二个 ID • SFEC = 111：用于 Rx 缓冲区过滤器
		SFID2[10:9]	该字段决定是将接收到的消息存储到 Rx 缓冲区中还是将其视为调试消息序列的消息 A、B 或 C。 • 0x0：将消息存储到 Rx 缓冲区中 • 0x1：调试消息 A • 0x2：调试消息 B • 0x3：调试消息 C **注意：**不支持调试功能。
		SFID2[8:6]	该字段用于控制扩展接口上的过滤器事件引脚 。如果过滤器匹配，则相应位位置上的 1 可以在相关过滤器事件引脚处生成一个持续时间为一个 MCAN_ICKL 周期的脉冲。 **注意：**仅支持两个过滤器事件引脚。
		SFID2[5:0]	该字段定义 Rx 缓冲区起始地址 MCAN_RXBC.RBSA 字段的偏移量，用于存储匹配消息。

19.4.19.6 扩展消息 ID 过滤器元素

最多可以为 29 位扩展 ID 配置 64 个过滤器元素。当访问扩展消息 ID 过滤器元素时，元素地址是过滤器列表扩展起始地址 MCAN_XIDFC.FLESA 字段加上过滤器元素索引 (0-63) 的两倍。

图 19-25 展示了扩展消息 ID 过滤器元素结构。表 19-14 展示了扩展消息 ID 过滤器元素字段说明。

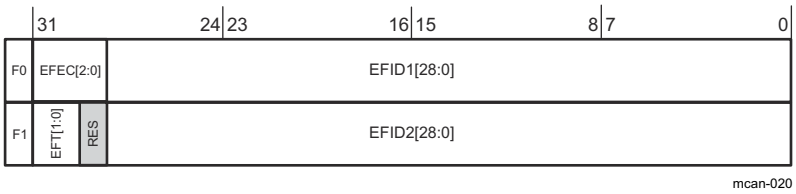


图 19-25. 扩展消息 ID 过滤器元素结构

表 19-14. 扩展消息 ID 过滤器元素字段说明

字	位	字段名称	说明
F0	31:29	EFEC[2:0]	扩展过滤器元素配置 所有启用的过滤器元素都用于扩展帧的接受过滤。接受过滤在第一个匹配的已启用过滤器元素处或到达过滤器列表末尾时停止。如果 EFEC = 100、101 或 110，则匹配设置中断标志 MCAN_IR.HPM，如果启用，则会生成中断。在这种情况下，MCAN_HPM 寄存器将更新为优先级匹配的状态。 • 0x0：禁用过滤器元素 • 0x1：如果过滤器匹配，则存储在 Rx FIFO 0 中 • 0x2：如果过滤器匹配，则存储在 Rx FIFO 1 中 • 0x3：如果过滤器匹配，则拒绝 ID • 0x4：如果过滤器匹配，则设置优先级 • 0x5：如果过滤器匹配，则设置优先级并存储在 FIFO 0 中 • 0x6：如果过滤器匹配，则设置优先级并存储在 FIFO 1 中 • 0x7：存储到 Rx 缓冲区中或存储为调试消息，忽略 EFT[1:0] 的配置
	28:0	EFID1[28:0]	扩展过滤器 ID 1 扩展 ID 过滤器元素的第一个 ID。 当过滤 Rx 缓冲区时，该字段定义要存储的扩展消息的 ID。接收到的标识符必须完全匹配，仅使用 XIDAM 掩码机制（请参阅节 19.4.16.1.5）。

表 19-14. 扩展消息 ID 过滤器元素字段说明 (continued)

字	位	字段名称	说明
F1	31:30	EFT[1:0]	扩展过滤器类型 • 0x0：从 EFID1 到 EFID2 的范围过滤器 (EFID2 ≥ EFID1) • 0x1：用于 EFID1 或 EFID2 的双 ID 过滤器 • 0x2：传统过滤器：EFID1 = 过滤器，EFID2 = 掩码 • 0x3：从 EFID1 到 EFID2 的范围过滤器 (EFID2 ≥ EFID1)，未应用 XIDAM 掩码
	29	RES	保留
	28:0	EFID2[28:0]	扩展过滤器 ID 2 根据 EFEC 的配置，该位字段具有不同的含义： • EFEC = 001 - 110：扩展 ID 过滤器元素的第二个 ID • EFEC = 111：用于 Rx 缓冲区过滤器
		EFID2[10:9]	该字段决定是将接收到的消息存储到 Rx 缓冲区中还是将其视为调试消息序列的消息 A、B 或 C。 • 0x0：将消息存储到 Rx 缓冲区中 • 0x1：调试消息 A • 0x2：调试消息 B • 0x3：调试消息 C **注意：**不支持调试功能。
		EFID2[8:6]	该字段用于控制扩展接口上的过滤器事件引脚 。如果过滤器匹配，则相应位位置上的 1 可以在相关过滤器事件引脚处生成一个持续时间为一个 MCAN_ICKL 周期的脉冲。 **注意：**仅支持两个过滤器事件引脚。
		EFID2[5:0]	该字段定义 Rx 缓冲区起始地址 MCAN_RXBC.RBSA 字段的偏移量，用于存储匹配消息。

19.5 MCAN 集成

图 19-26 展示了器件中的 MCAN 模块集成。

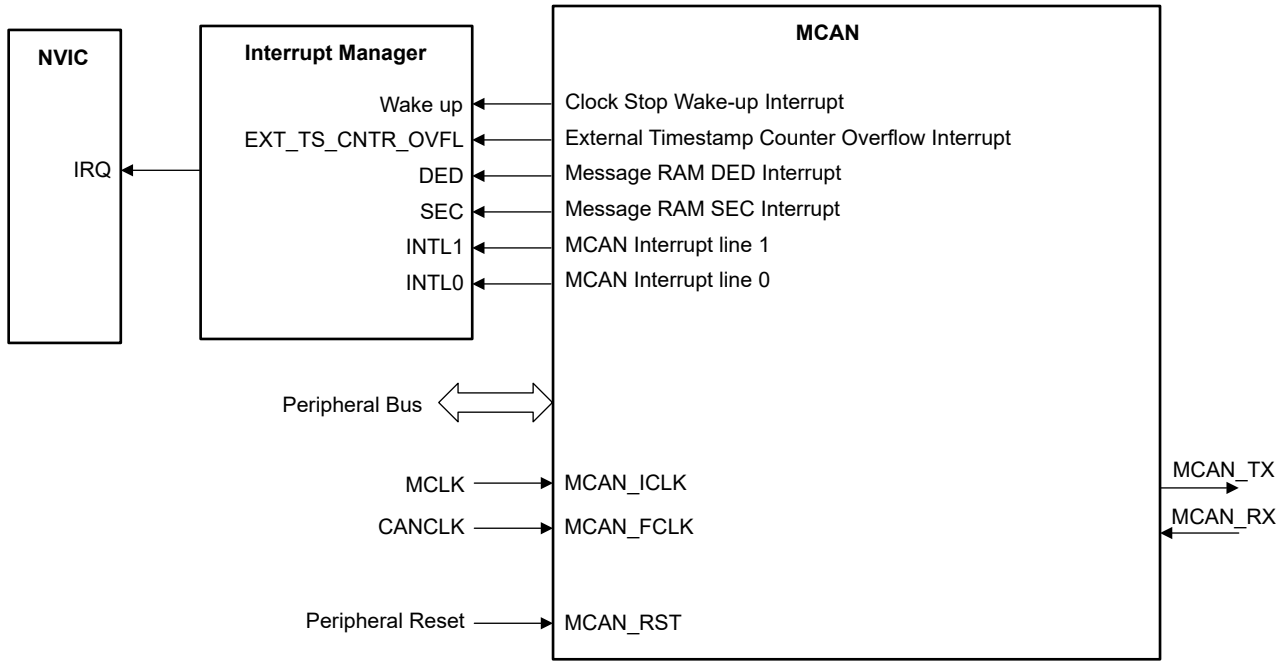


图 19-26. MCAN 集成

表 19-15 总结了器件中的 MCAN 模块集成。

表 19-15. MCAN 时钟和复位

目标信号名称	源信号名称	说明
时钟		
MCAN_ICLK	MCLK	MCAN 模块的接口时钟
MCAN_FCLK	CANCLK	MCAN 的位时序时钟
复位		
MCAN_RST	外设复位	连接到 MCAN 模块的异步复位信号

19.6 中断和事件支持

MCAN 模块包含一个事件发布者 (CPU_INT)，用于通过静态事件路由管理对 CPU 子系统的 MCAN 中断请求 (IRQ)。

表 19-16 中总结了 MCAN 事件。

表 19-16. MCAN 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	MCAN	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 MCAN 到 CPU 的中断路由

19.6.1 CPU 中断事件发布者 (CPU_INT)

MCAN 模块提供不同的中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 19-17 显示了来自 MCAN 的 CPU 中断事件。

表 19-17. MCAN 中断事件条件 (CPU_INT)

索引 (IIDX)	名称	说明
0x1	MCAN_IRQ_INT0	MCAN 中断 0

表 19-17. MCAN 中断事件条件 (CPU_INT) (continued)

索引 (IIDX)	名称	说明
0x2	MCAN_IRQ_INT1	MCAN 中断 1
0x3	MCAN_IRQ_ECC	MCAN ECC SEC（单错校正）中断
0x4	MCAN_IRQ_ECC_UNCORR	MCAN ECC 不可纠正/DED（双错检测）中断
0x5	MCANSS_IRQ_TS_CNTR_OVFL	MCAN 时间戳计数器溢出中断
0x6	MCANSS_IRQ_TS_WAKE	MCAN 时钟停止唤醒中断

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。在软件读取 IIDX 寄存器或写入相应的 ICLR 寄存器位时，会清除中断 (RIS) 标志。

有关为 CPU 中断配置事件寄存器的指导，请参阅 *节 7.2.5*。

19.7 MCAN 寄存器

本节介绍模块化控制器局域网 (MCAN) 模块寄存器。

19.7.1 MCAN 寄存器

表 19-18 列出了 MCAN 寄存器的存储器映射寄存器。表 19-18 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 19-18. MCAN 寄存器

偏移	缩写	寄存器名称	组	部分
6800h	PWREN	电源使能		转到
6804h	RSTCTL	复位控制		转到
6814h	STAT	状态寄存器		转到
7000h	MCAN_CREL	MCAN 内核版本寄存器		转到
7004h	MCAN_ENDN	MCAN 字节序寄存器		转到
700Ch	MCAN_DBTP	MCAN 数据位时序和预分频器寄存器		转到
7010h	MCAN_TEST	MCAN 测试寄存器		转到
7014h	MCAN_RWD	MCAN RAM 看门狗		转到
7018h	MCAN_CCCR	MCAN CC 控制寄存器		转到
701Ch	MCAN_NBTP	MCAN 标称位时序和预分频器寄存器		转到
7020h	MCAN_TSCC	MCAN 时间戳计数器配置		转到
7024h	MCAN_TSCV	MCAN 时间戳计数器值		转到
7028h	MCAN_TOCC	MCAN 超时计数器配置		转到
702Ch	MCAN_TOCV	MCAN 超时计数器值		转到
7040h	MCAN_ECR	MCAN 错误计数器寄存器		转到
7044h	MCAN_PSR	MCAN 协议状态寄存器		转到
7048h	MCAN_TDCR	MCAN 发送器延迟补偿寄存器		转到
7050h	MCAN_IR	MCAN 中断寄存器		转到
7054h	MCAN_IE	MCAN 中断使能		转到
7058h	MCAN_ILS	MCAN 中断线选择		转到
705Ch	MCAN_ILE	MCAN 中断线使能		转到
7080h	MCAN_GFC	MCAN 全局过滤器配置		转到
7084h	MCAN_SIDFC	MCAN 标准 ID 过滤器配置		转到
7088h	MCAN_XIDFC	MCAN 扩展 ID 过滤器配置		转到
7090h	MCAN_XIDAM	MCAN 扩展 ID 和掩码		转到
7094h	MCAN_HPMS	MCAN 高优先级消息状态		转到
7098h	MCAN_NDAT1	MCAN 新数据 1		转到
709Ch	MCAN_NDAT2	MCAN 新数据 2		转到
70A0h	MCAN_RXF0C	MCAN Rx FIFO 0 配置		转到
70A4h	MCAN_RXF0S	MCAN Rx FIFO 0 状态		转到
70A8h	MCAN_RXF0A	MCAN Rx FIFO 0 确认		转到
70ACh	MCAN_RXBC	MCAN Rx 缓冲区配置		转到
70B0h	MCAN_RXF1C	MCAN Rx FIFO 1 配置		转到
70B4h	MCAN_RXF1S	MCAN Rx FIFO 1 状态		转到
70B8h	MCAN_RXF1A	MCAN Rx FIFO 1 确认		转到
70BCh	MCAN_RXESC	MCAN Rx 缓冲区/FIFO 元素大小配置		转到
70C0h	MCAN_TXBC	MCAN Tx 缓冲区配置		转到
70C4h	MCAN_TXFQS	MCAN Tx FIFO/队列状态		转到
70C8h	MCAN_TXESC	MCAN Tx 缓冲区元素大小配置		转到

表 19-18. MCAN 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
70CCh	MCAN_TXBRP	MCAN Tx 缓冲区请求待处理		转到
70D0h	MCAN_TXBAR	MCAN Tx 缓冲区添加请求		转到
70D4h	MCAN_TXBCR	MCAN Tx 缓冲区取消请求		转到
70D8h	MCAN_TXBTO	发生 MCAN Tx 缓冲区传输		转到
70DCh	MCAN_TXBCF	MCAN Tx 缓冲区取消完成		转到
70E0h	MCAN_TXBTIE	MCAN Tx 缓冲区传输中断使能		转到
70E4h	MCAN_TXBCIE	MCAN Tx 缓冲区取消完成中断使能		转到
70F0h	MCAN_TXEFC	MCAN Tx 事件 FIFO 配置		转到
70F4h	MCAN_TXEFS	MCAN Tx 事件 FIFO 状态		转到
70F8h	MCAN_TXEFA	MCAN Tx 事件 FIFO 确认		转到
7200h	MCANSS_PID	MCAN 子系统修订版本寄存器		转到
7204h	MCANSS_CTRL	MCAN 子系统控制寄存器		转到
7208h	MCANSS_STAT	MCAN 子系统状态寄存器		转到
720Ch	MCANSS_ICS	MCAN 子系统中断清除影子寄存器		转到
7210h	MCANSS_IRS	MCAN 子系统中断原始状态寄存器		转到
7214h	MCANSS_IECS	MCAN 子系统中断使能清除影子寄存器		转到
7218h	MCANSS_IE	MCAN 子系统中断使能寄存器		转到
721Ch	MCANSS_IES	MCAN 子系统中断使能状态		转到
7220h	MCANSS_EOI	MCAN 子系统中断结束		转到
7224h	MCANSS_EXT_TS_PRESCALER	MCAN 子系统外部时间戳预分频器 0		转到
7228h	MCANSS_EXT_TS_UNSERVICED_INTR_CNTR	MCAN 子系统外部时间戳未处理中断计数器		转到
7400h	MCANERR_REV	MCAN 错误聚合器修订版本寄存器		转到
7408h	MCANERR_VECTOR	MCAN ECC 向量寄存器		转到
740Ch	MCANERR_STAT	MCAN 错误其他状态		转到
7410h	MCANERR_WRAP_REV	MCAN ECC 包装器修订版本寄存器		转到
7414h	MCANERR_CTRL	MCAN ECC 控制		转到
7418h	MCANERR_ERR_CTRL1	MCAN ECC 错误控制 1 寄存器		转到
741Ch	MCANERR_ERR_CTRL2	MCAN ECC 错误控制 2 寄存器		转到
7420h	MCANERR_ERR_STAT1	MCAN ECC 错误状态 1 寄存器		转到
7424h	MCANERR_ERR_STAT2	MCAN ECC 错误状态 2 寄存器		转到
7428h	MCANERR_ERR_STAT3	MCAN ECC 错误状态 3 寄存器		转到
743Ch	MCANERR_SEC_EOI	MCAN 单错校正中断结束寄存器		转到
7440h	MCANERR_SEC_STATUS	MCAN 单错校正中断状态寄存器		转到
7480h	MCANERR_SEC_ENABLE_SET	MCAN 单错校正中断使能设置寄存器		转到
74C0h	MCANERR_SEC_ENABLE_CLR	MCAN 单错校正中断使能清除寄存器		转到
753Ch	MCANERR_DED_EOI	MCAN 双错检测中断结束寄存器		转到
7540h	MCANERR_DED_STATUS	MCAN 双错检测中断状态寄存器		转到
7580h	MCANERR_DED_ENABLE_SET	MCAN 双错检测中断使能设置寄存器		转到
75C0h	MCANERR_DED_ENABLE_CLR	MCAN 双错检测中断使能清除寄存器		转到

表 19-18. MCAN 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
7600h	MCANERR_AGGR_ENABLE_SET	MCAN 错误聚合器使能设置寄存器		转到
7604h	MCANERR_AGGR_ENABLE_CLR	MCAN 错误聚合器使能清除寄存器		转到
7608h	MCANERR_AGGR_STATUS_SET	MCAN 错误聚合器状态设置寄存器		转到
760Ch	MCANERR_AGGR_STATUS_CLR	MCAN 错误聚合器状态清除寄存器		转到
7820h	IIDX	中断索引寄存器	CPU_INT	转到
7828h	IMASK	中断屏蔽	CPU_INT	转到
7830h	RIS	原始中断状态	CPU_INT	转到
7838h	MIS	已屏蔽中断状态	CPU_INT	转到
7840h	ISET	中断设置	CPU_INT	转到
7848h	ICLR	中断清除	CPU_INT	转到
78E0h	EVT_MODE	事件模式		转到
78FCh	DESC	模块说明		转到
7900h	MCANSS_CLKEN	MCAN 模块时钟使能		转到
7904h	MCANSS_CLKDIV	时钟分频器		转到
7908h	MCANSS_CLKCTL	MCAN-SS 时钟停止控制寄存器		转到
790Ch	MCANSS_CLKSTS	MCANSS 时钟停止状态寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 19-19 展示了适用于此部分中访问类型的代码。

表 19-19. MCAN 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
R-0	R -0	读取 返回 0
RC	R C	读取 以清除
RS	R S	读取 以设置
写入类型		
W	W	写入
W1C	W 1C	写入 1 以进行清除
W1S	W 1S	写入 1 以进行设置
W1SQ	W 1S Q	写入 1 以设置 符合。必须满足一个条件才能执行此操作。
WD	W D	写入 递减。将指定的位字段按写入的量递减。
WI	W I	写入 递增。将指定的位字段按写入的量递增。
WK	W K	写入 受密钥保护的写入
WQ	W Q	写入 符合。必须满足一个条件才能执行此操作。

表 19-19. MCAN 访问类型代码 (continued)

访问类型	代码	说明
复位或默认值		
-n		复位后的值或默认值

19.7.1.1 PWREN（偏移 = 6800h）[复位 = 00000000h]

图 19-27 展示了 PWREN，表 19-20 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 19-27. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 19-20. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

19.7.1.2 RSTCTL（偏移 = 6804h）[复位 = 00000000h]

图 19-28 展示了 RSTCTL，表 19-21 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 19-28. RSTCTL

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

表 19-21. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

19.7.1.3 STAT（偏移 = 6814h）[复位 = 00000000h]

图 19-29 展示了 STAT，表 19-22 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 19-29. STAT

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

表 19-22. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

19.7.1.4 MCAN_CREL（偏移 = 7000h）[复位 = 32380608h]

图 19-30 展示了 MCAN_CREL，表 19-23 中对此进行了介绍。

返回到汇总表。

MCAN 内核版本寄存器

图 19-30. MCAN_CREL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL				STEP				SUBSTEP				YEAR			
R-3h				R-2h				R-3h				R-8h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON								DAY							
R-6h								R-8h							

表 19-23. MCAN_CREL 字段说明

位	字段	类型	复位	说明
31-28	REL	R	3h	内核版本。一位，BCD 编码。
27-24	STEP	R	2h	内核版本的步骤。一位，BCD 编码。
23-20	SUBSTEP	R	3h	内核版本的子步骤。一位，BCD 编码。
19-16	YEAR	R	8h	时间戳年。一位，BCD 编码。
15-8	MON	R	6h	时间戳月。两位，BCD 编码。
7-0	DAY	R	8h	时间戳日。两位，BCD 编码。

19.7.1.5 MCAN_ENDN（偏移 = 7004h）[复位 = 87654321h]

图 19-31 展示了 MCAN_ENDN，表 19-24 中对此进行了介绍。

返回到汇总表。

MCAN 字节序寄存器

图 19-31. MCAN_ENDN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV																															
R-87654321h																															

表 19-24. MCAN_ENDN 字段说明

位	字段	类型	复位	说明
31-0	ETV	R	87654321h	字节序测试值。通过读取该寄存器中保持的常量值，软件可以确定主机 CPU 的字节序。

19.7.1.6 MCAN_DBTP（偏移 = 700Ch）[复位 = 00000A33h]

图 19-32 展示了 MCAN_DBTP，表 19-25 中对此进行了介绍。

返回到汇总表。

仅当设置 CCCR.CCE 和 CCCR.INIT 位时，该寄存器才可写。可在 4 到 49 个时间量子的范围内对 CAN 位时间进行编程。可在 1 到 32 个 m_can_cclk 周期的范围内对 CAN 时间量子进行编程。tq = (DBRP + 1) mtq。
DTSEG1 是 Prop_Seg 和 Phase_Seg1 的总和。DTSEG2 为 Phase_Seg2。
因此，位时间的长度为 (编程值) (DTSEG1 + DTSEG2 + 3) tq 或 (功能值) (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq。
信息处理时间 (IPT) 为零，这意味着下一位的数据在采样点之后的第一个时钟沿可用。

图 19-32. MCAN_DBTP

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
TDC	RESERVED		DBRP				
R/WQ-0h	R/W-0h		R/WQ-0h				
15	14	13	12	11	10	9	8
保留			DTSEG1				
R/W-0h			R/WQ-Ah				
7	6	5	4	3	2	1	0
DTSEG2				DSJW			
R/WQ-3h				R/WQ-3h			

表 19-25. MCAN_DBTP 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23	TDC	R/WQ	0h	发送器延迟补偿 0 发送器延迟补偿已禁用 1 发送器延迟补偿已启用
22-21	RESERVED	R/W	0h	
20-16	DBRP	R/WQ	0h	数据比特率预分频器。该值是通过对振荡器频率分频获得的，用于产生位时间份额。位时间由这个量子的倍数构成。比特率预分频器的有效值为 0 至 31。硬件在实际应用中对该值的解释要比此处编程的值大 1。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
15-13	RESERVED	R/W	0h	
12-8	DTSEG1	R/WQ	Ah	采样点之前的数据时间段。有效值为 0 至 31。硬件对该值的实际解释是使用的值比编程的值大 1。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
7-4	DTSEG2	R/WQ	3h	采样点之后的数据时间段。有效值为 0 至 15。硬件对该值的实际解释是使用的值比编程的值大 1。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
3-0	DSJW	R/WQ	3h	数据重新同步跳转宽度。有效值为 0 至 15。硬件在实际应用中对该值的解释要比此处编程的值大 1。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。

19.7.1.7 MCAN_TEST（偏移 = 7010h）[复位 = X]

图 19-33 展示了 MCAN_TEST，表 19-26 中对此进行了介绍。

返回到汇总表。

必须通过将位 CCCR.TEST 设置为“1”来启用对测试寄存器的写入访问。当位 CCCR.TEST 复位时，所有测试寄存器功能均设置为其复位值。
环回模式和内部 CAN TX 引脚的软件控制是硬件测试模式。TX 的
编程？“00”可能会干扰 CAN 总线上的消息传输。

图 19-33. MCAN_TEST

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RX	TX		LBCK	保留			
R-X	R/WQ-0h		R/WQ-0h	R/W-0h			

表 19-26. MCAN_TEST 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7	RX	R	X	接收引脚。监测 CAN 接收引脚的实际值。 0h = 显性：CAN 总线为显性状态（CAN RX 引脚 = “0”） 1h = 隐性：CAN 总线为隐性状态（CAN RX 引脚 = “1”）
6-5	TX	R/WQ	0h	控制发送引脚 00 CAN TX 引脚由 CAN 内核控制，在 CAN 位时间结束时更新 01 可在 CAN TX 引脚处监测采样点 10 CAN TX 引脚处为显性（“0”）电平 11 CAN TX 引脚处为隐性（“1”） 仅当 CCCR.CCE = “1”且 CCCR.INIT = “1”时，才能进行限定写入。
4	LBCK	R/WQ	0h	环回模式。仅当 CCCR.CCE = “1”且 CCCR.INIT = “1”时，才能进行限定写入。 0h = 禁用：复位值，禁用环回模式 1h = 启用： 启用环回模式
3-0	RESERVED	R/W	0h	

19.7.1.8 MCAN_RWD（偏移 = 7014h）[复位 = 00000000h]

图 19-34 展示了 MCAN_RWD，表 19-27 中对此进行了介绍。

返回到汇总表。

MCAN RAM 看门狗

图 19-34. MCAN_RWD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																WDV								WDC							
R/W-0h																R-0h								R/WQ-0h							

表 19-27. MCAN_RWD 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-8	WDV	R	0h	看门狗值。实际消息 RAM 看门狗计数器值。 RAM 看门狗监测消息 RAM 的 READY 输出。通过 MCAN 的通用主接口访问消息 RAM 时，消息 RAM 看门狗计数器将以 WDC 字段配置的值开始计数。当消息 RAM 通过激活其 READY 输出发出成功完成信号时，计数器将重新加载 WDC。如果在计数器倒计时至零之前消息 RAM 没有响应，则计数器停止并设置中断标志 MCAN_IR.WDI。 RAM 看门狗计数器由主机（系统）时钟计时。
7-0	WDC	R/WQ	0h	看门狗配置。消息 RAM 看门狗计数器的起始值。当复位值为“00”时，计数器处于禁用状态。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。

19.7.1.9 MCAN_CCCR（偏移 = 7018h）[复位 = 00000001h]

图 19-35 展示了 MCAN_CCCR，表 19-28 中对此进行了介绍。

返回到汇总表。

MCAN CC 控制寄存器

图 19-35. MCAN_CCCR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
NISO	TXP	EFBI	PXHD	RESERVED		BRSE	FDOE
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/W-0h		R/WQ-0h	R/WQ-0h
7	6	5	4	3	2	1	0
测试	DAR	MON	CSR	CSA	ASM	CCE	INIT
R/W1SQ-0h	R/WQ-0h	R/W1SQ-0h	R/W-0h	R-0h	R/W1SQ-0h	R/WQ-0h	R/W-1h

表 19-28. MCAN_CCCR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	NISO	R/WQ	0h	非 ISO 操作。如果设置该位，则 MCAN 使用 Bosch CAN FD 规范 V1.0 指定的 CAN FD 帧格式。 0 CAN FD 帧格式符合 ISO 11898-1:2015 1 CAN FD 帧格式符合 Bosch CAN FD 规范 V1.0
14	TXP	R/WQ	0h	传输暂停。如果设置该位，则 MCAN 在成功传输一帧后，会暂停两个 CAN 位时间，然后再开始下一次传输。 0 禁用传输暂停 1 启用传输暂停 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
13	EFBI	R/WQ	0h	总线集成期间的边沿滤波 0 禁用边沿滤波 1 需要两个连续的显性 tq 以检测用于硬同步的边沿 仅当 CCCR.CCE = “1” 且 CCCR.INIT= “1” 时，才能进行限定写入。
12	PXHD	R/WQ	0h	协议异常处理禁用 0 启用协议异常处理 1 禁用协议异常处理 注意：禁用协议异常处理后，MCAN 将在检测到协议异常情况时传输一个错误帧。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
11-10	RESERVED	R/W	0h	
9	BRSE	R/WQ	0h	比特率开关启用 0 禁用比特率传输开关 1 启用比特率传输开关 注意：禁用 CAN FD 运行后，FDOE = “0” ，不评估 BRSE。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

表 19-28. MCAN_CCCR 字段说明 (continued)

位	字段	类型	复位	说明
8	FDOE	R/WQ	0h	灵活数据速率运行启用 0 禁用 FD 运行 1 启用 FD 运行 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
7	测试	R/W1SQ	0h	测试模式启用 0 正常运行，寄存器 TEST 保留复位值 1 测试模式，启用对寄存器 TEST 的写入访问 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能将限定写入设置为 1。
6	DAR	R/WQ	0h	禁用自动重传 0 启用未成功传输消息时自动重传 1 禁用自动重传 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
5	MON	R/W1SQ	0h	总线监控模式。仅当 CCE 和 INIT 均设置为 “1” 时，位 MON 才能由 SW 设置。该位可随时由 SW 复位。 0 禁用总线监控模式 1 启用总线监控模式 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能将限定写入设置为 1。
4	CSR	R/W	0h	时钟停止请求 0 未请求时钟停止 1 已请求时钟停止。当请求时钟停止时，在所有待处理的传输请求完成且 CAN 总线达到空闲状态后，将首先设置 INIT，然后设置 CSA。
3	CSA	R	0h	时钟停止确认 0 未确认时钟停止 1 可通过停止主机和 CAN 时钟在断电时设置 MCAN
2	ASM	R/W1SQ	0h	受限运行模式。仅当 CCE 和 INIT 均设置为 “1” 时，位 ASM 才能由 SW 设置。该位可随时由 SW 复位。 0 正常 CAN 运行 1 受限运行模式激活 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能将限定写入设置为 1。
1	CCE	R/WQ	0h	配置更改启用 0 CPU 不具有对受保护配置寄存器的写入访问权限 1 CPU 具有对受保护配置寄存器的写入访问权限（当 CCCR.INIT = “1” 时） 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
0	INIT	R/W	1h	初始化 0 正常运行 1 初始化已启动 注意：由于两个时钟域之间的同步机制，在读回写入 INIT 的值之前可能会有延迟。因此，将 INIT 设置为新值之前，程序员必须通过读取 INIT 来确保已接受先前写入 INIT 的值。

19.7.1.10 MCAN_NBTP（偏移 = 701Ch）[复位 = 06000A03h]

图 19-36 展示了 MCAN_NBTP，表 19-29 中对此进行了介绍。

返回到汇总表。

仅当设置 CCCR.CCE 和 CCCR.INIT 位时，该寄存器才可写。可在 4 到 385 个时间量子的范围内对 CAN 位时间进行编程。可在 1 到 512 个 m_can_cclk 周期的范围内对 CAN 时间量子进行编程。tq = (NBRP + 1) mtq。
NTSEG1 是 Prop_Seg 和 Phase_Seg1 的总和。NTSEG2 为 Phase_Seg2。
因此，位时间的长度为 (编程值) (NTSEG1 + NTSEG2 + 3) tq 或 (功能值) (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq。
信息处理时间 (IPT) 为零，这意味着下一位的数据在采样点之后的第一个时钟沿可用。
注意：当 CAN 时钟为 8MHz 时，复位值 0x06000A03 将 MCAN 配置为 500kBit/s 的比特率。

图 19-36. MCAN_NBTP

31	30	29	28	27	26	25	24
NSJW							NBRP
R/WQ-3h							R/WQ-0h
23	22	21	20	19	18	17	16
NBRP							
R/WQ-0h							
15	14	13	12	11	10	9	8
NTSEG1							
R/WQ-Ah							
7	6	5	4	3	2	1	0
RESERVED	NTSEG2						
R/W-0h	R/WQ-3h						

表 19-29. MCAN_NBTP 字段说明

位	字段	类型	复位	说明
31-25	NSJW	R/WQ	3h	标称（重新）同步跳转宽度。有效值为 0 至 127。硬件在实际应用中对该值的解释要比此处编程的值大 1。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
24-16	NBRP	R/WQ	0h	标称比特率预分频器。该值是通过对振荡器频率分频获得的，用于产生位时间份额。位时间由这个量子的倍数构成。比特率预分频器的有效值为 0 至 511。硬件在实际应用中对该值的解释要比此处编程的值大 1。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
15-8	NTSEG1	R/WQ	Ah	采样点之前的标称时间段。有效值为 1 至 255。硬件对该值的实际解释是使用的值比编程的值大 1。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
7	RESERVED	R/W	0h	
6-0	NTSEG2	R/WQ	3h	采样点之后的标称时间段。有效值为 1 至 127。硬件对该值的实际解释是使用的值比编程的值大 1。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

19.7.1.11 MCAN_TSCC（偏移 = 7020h）[复位 = 00000000h]

图 19-37 展示了 MCAN_TSCC，表 19-30 中对此进行了介绍。

返回到汇总表。

MCAN 时间戳计数器配置

图 19-37. MCAN_TSCC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED												TCP			
R/W-0h												R/WQ-0h			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED														TSS	
R/W-0h														R/WQ-0h	

表 19-30. MCAN_TSCC 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	0h	
19-16	TCP	R/WQ	0h	时间戳计数器预分频器。以 CAN 位时间的倍数配置时间戳和超时计数器时间单位。有效值为 0 至 15。硬件在实际应用中对该值的解释要比此处编程的值大 1。 注意：对于 CAN FD，需要一个外部计数器来生成时间戳（TSS = “10”）。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
15-2	保留	R/W	0h	
1-0	TSS	R/WQ	0h	时间戳选择 00 时间戳计数器值始终为 0x0000 01 时间戳计数器值根据 TCP 递增 10 使用外部时间戳计数器值 11 与 “00” 相同 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

19.7.1.12 MCAN_TSCV（偏移 = 7024h）[复位 = 00000000h]

图 19-38 展示了 MCAN_TSCV，表 19-31 中对此进行了介绍。

返回到汇总表。

MCAN 时间戳计数器值

图 19-38. MCAN_TSCV

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

表 19-31. MCAN_TSCV 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	TSC	R/W	0h	时间戳计数器。帧开始时捕获内部/外部时间戳计数器值（Rx 和 Tx）。当 TSCC.TSS = “01” 时，时间戳计数器以 CAN 位时间的倍数 (1...16) 递增，具体取决于 TSCC.TCP 的配置。绕回会设置中断标志 IR.TSW。写入访问将计数器复位为零。当 TSCC.TSS = “10” 时，TSC 反映外部时间戳计数器值，而写入访问没有影响。 注意：“绕回”是指时间戳计数器值从非零变为零， 这并不是由对 MCAN_TSCV 的写入访问引起。

19.7.1.13 MCAN_TOCC（偏移 = 7028h）[复位 = FFFF0000h]

图 19-39 展示了 MCAN_TOCC，表 19-32 中对此进行了介绍。

返回到汇总表。

MCAN 超时计数器配置

图 19-39. MCAN_TOCC

31	30	29	28	27	26	25	24
TOP							
R/WQ-FFFFh							
23	22	21	20	19	18	17	16
TOP							
R/WQ-FFFFh							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED					TOS		ETOC
R/W-0h					R/WQ-0h		R/WQ-0h

表 19-32. MCAN_TOCC 字段说明

位	字段	类型	复位	说明
31-16	TOP	R/WQ	FFFFh	超时周期。超时计数器（向下计数器）的起始值。配置超时周期。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
15-3	RESERVED	R/W	0h	
2-1	TOS	R/WQ	0h	超时选择。在连续模式下运行时，写入 TOCV 会将计数器预设为 TOCC.TOP 配置的值，并继续向下计数。当超时计数器由其中一个 FIFO 控制时，空 FIFO 会将计数器预设为 TOCC.TOP 配置的值。存储第一个 FIFO 元素时，开始向下计数。 00 连续运行 01 由 Tx 事件 FIFO 控制的超时 10 由 Rx FIFO 0 控制的超时 11 由 Rx FIFO 1 控制的超时 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
0	ETOC	R/WQ	0h	启用超时计数器 0 禁用超时计数器 1 启用超时计数器 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

19.7.1.14 MCAN_TOCV (偏移 = 702Ch) [复位 = 0000FFFFh]

图 19-40 展示了 MCAN_TOCV，表 19-33 中对此进行了介绍。

返回到汇总表。

MCAN 超时计数器值

图 19-40. MCAN_TOCV

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

表 19-33. MCAN_TOCV 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	TOC	R/W	FFFFh	超时计数器。超时计数器以 CAN 位时间的倍数 (1...16) 递减，具体取决于 TSCC.TCP 的配置。当递减至零时，设置中断标志 IR.TOO，超时计数器停止。通过 TOCC.TOS 配置启动和复位/重启条件。

19.7.1.15 MCAN_ECR (偏移 = 7040h) [复位 = 00000000h]

图 19-41 展示了 MCAN_ECR，表 19-34 中对此进行了介绍。

返回到汇总表。

MCAN 错误计数器寄存器

图 19-41. MCAN_ECR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED								CEL							
R-0h								RC-0h							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC							TEC							
R-0h	R-0h							R-0h							

表 19-34. MCAN_ECR 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R	0h	
23-16	CEL	RC	0h	CAN 错误记录。每当 CAN 协议错误导致发送错误计数器或接收错误计数器递增时，计数器都会递增。这通过对 CEL 的读取访问来复位。计数器在 0xFF 处停止；TEC 或 REC 的下一个增量设置中断标志 IR.ELO。 注意：设置 CCCR.ASM 后，CAN 协议控制器不会在检测到 CAN 协议错误时使 TEC 和 REC 递增，但 CEL 仍会递增。
15	RP	R	0h	接收错误被动 0 接收错误计数器低于 128 的错误被动级别 1 接收错误计数器达到 128 的错误被动级别
14-8	REC	R	0h	接收错误计数器。接收错误计数器的实际状态，值介于 0 和 127 之间。 注意：设置 CCCR.ASM 后，CAN 协议控制器不会在检测到 CAN 协议错误时使 TEC 和 REC 递增，但 CEL 仍会递增。
7-0	TEC	R	0h	发送错误计数器。发送错误计数器的实际状态，值介于 0 和 255 之间。 注意：设置 CCCR.ASM 后，CAN 协议控制器不会在检测到 CAN 协议错误时使 TEC 和 REC 递增，但 CEL 仍会递增。

19.7.1.16 MCAN_PSR (偏移 = 7044h) [复位 = 00000707h]

图 19-42 展示了 MCAN_PSR，表 19-35 中对此进行了介绍。

返回到汇总表。

MCAN 协议状态寄存器

图 19-42. MCAN_PSR

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED	TDCV						
R-0h	R-0h						
15	14	13	12	11	10	9	8
保留	PXE	RFDF	RBRS	RESI	DLEC		
R-0h	RC-0h	RC-0h	RC-0h	RC-0h	RS-7h		
7	6	5	4	3	2	1	0
BO	EW	EP	ACT		LEC		
R-0h	R-0h	R-0h	R-0h		RS-7h		

表 19-35. MCAN_PSR 字段说明

位	字段	类型	复位	说明
31-23	RESERVED	R	0h	
22-16	TDCV	R	0h	发送器延迟补偿值。二次采样点的位置，由从内部 CAN TX 信号到内部 CAN RX 信号测得的延迟与 TDCR.TDCO 之和定义。SSP 位置是数据段传输位的起点和二次采样点之间的 mtq 数。有效值为 0 至 127 mtq。
15	保留	R	0h	
14	PXE	RC	0h	协议异常事件 0 自上次读取访问以来未发生协议异常事件 1 发生了协议异常事件
13	RFDF	RC	0h	收到 CAN FD 消息。该位的设置与接受过滤无关。 0 由于该位由 CPU 复位，因此未接收到 CAN FD 消息 1 已接收到设置了 FDF 标志的 CAN FD 格式消息
12	RBRS	RC	0h	最后接收到的 CAN FD 消息的 BRS 标志。该位与 RFDF 一起设置，与接受过滤无关。 0 最后接收到的 CAN FD 消息未设置 BRS 标志 1 最后接收到的 CAN FD 消息已设置 BRS 标志
11	RESI	RC	0h	最后接收到的 CAN FD 消息的 ESI 标志。该位与 RFDF 一起设置，与接受过滤无关。 0 最后接收到的 CAN FD 消息未设置 ESI 标志 1 最后接收到的 CAN FD 消息已设置 ESI 标志
10-8	DLEC	RS	7h	数据段最后一个错误代码。在设置了 BRS 标志的 CAN FD 格式帧的数据段发生的最后一个错误的类型。编码方式与 LEC 相同。当设置了 BRS 标志的 CAN FD 格式帧已无误地传输 (接收或发送) 时，该字段将清零。
7	BO	R	0h	Bus_Off 状态 0 M_CAN 不处于 Bus_Off 状态 1 M_CAN 处于 Bus_Off 状态
6	EW	R	0h	警告状态 0 两个错误计数器均低于 Error_Warning 限值 96 1 至少有一个错误计数器达到 Error_Warning 限值 96

表 19-35. MCAN_PSR 字段说明 (continued)

位	字段	类型	复位	说明
5	EP	R	0h	错误被动 0 M_CAN 处于 Error_Active 状态。它通常参与总线通信，并在检测到错误时发送活动错误标志 1 M_CAN 处于 Error_Passive 状态
4-3	ACT	R	0h	节点活动。监测模块的 CAN 通信状态。 00 同步 - 节点在 CAN 通信上同步 01 空闲 - 节点既不是接收器也不是发送器 10 接收器 - 节点作为接收器运行 11 发送器 - 节点作为发送器运行 注意：一个协议异常事件将 ACT 设置为“00”。
2-0	LEC	RS	7h	最后一个错误代码。LEC 指示 CAN 总线上最后发生的错误的类型。当无误地传输（接收或发送）消息后，该字段将自动清为“0”。 0 无错误：由于 LEC 已通过成功接收或发送而复位，因此未发生错误。 1 填充错误：一个序列中有超过 5 个相同极性的位出现在接收报文中，这是接收报文所不能允许的。 2 格式错误：接收到的帧的固定格式部分的格式错误。 3 AckError：MCAN 发送的消息未被另一节点确认。 4 Bit1Error：在消息（仲裁字段除外）传输期间，器件想要发送一个隐性电平（逻辑值为“1”的位），但监测到的总线值为显性。 5 Bit0Error：在消息（或确认位、活动错误标志或过载标志）传输期间，器件想要发送一个显性电平（数据或标识符位逻辑值为“0”），但监测到的总线值为隐性。在 Bus_Off 恢复期间，每次监测到含 11 个隐性位的序列时都会设置此状态。这使得 CPU 能够监测 Bus_Off 恢复序列的进程（表明总线并未卡在显性状态或持续受到干扰）。 6 CRCError：接收到的消息的 CRC 校验和不正确。传入消息的 CRC 与根据接收到的数据计算得出的 CRC 不匹配。 7 NoChange：对协议状态寄存器的任何读取访问都会将 LEC 重新初始化为“7”。当 LEC 显示值“7”时，表示自上次 CPU 对协议状态寄存器的读取访问以来，未检测到 CAN 总线事件。 注意：当一个 CAN FD 格式的帧到达设置了 BRS 标志的数据段时，下一个 CAN 事件（错误或有效帧）将显示在 DLEC 而不是 LEC 中。CAN FD CRC 序列的固定填充位中的错误将显示为“格式错误”，而不是“填充错误”。注意：设置或复位 CCCR.INIT 并不能缩短 Bus_Off 恢复序列（请参阅 ISO 11898-1:2015）。如果器件进入 Bus_Off 状态，它将自行设置 CCCR.INIT，从而停止所有总线活动。一旦 CCCR.INIT 被 CPU 清零，器件将等待 129 次出现总线空闲（129 * 11 个连续隐性位），然后再恢复正常运行。Bus_Off 恢复序列结束时，错误管理计数器将会复位。在 CCCR.INIT 复位后的等待期间，每次监测到含 11 个隐性位的序列时，都会向 PSR.LEC 写入一个 Bit0Error 代码，使 CPU 不但可以轻易地检测 CAN 总线是卡在显性状态还是持续受到干扰，还可以对 Bus_Off 恢复序列进行监测。ECR.REC 用于对这些序列进行计数。

19.7.1.17 MCAN_TDCR（偏移 = 7048h）[复位 = 00000000h]

图 19-43 展示了 MCAN_TDCR，表 19-36 中对此进行了介绍。

返回到汇总表。

MCAN 发送器延迟补偿寄存器

图 19-43. MCAN_TDCR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	TDCO						
R/W-0h	R/WQ-0h						
7	6	5	4	3	2	1	0
RESERVED	TDCF						
R/W-0h	R/WQ-0h						

表 19-36. MCAN_TDCR 字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14-8	TDCO	R/WQ	0h	发送器延迟补偿偏移。偏移值，定义从内部 CAN TX 信号到内部 CAN RX 信号测得的延迟与二次采样点之间的距离。有效值为 0 至 127 mtq。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
7	RESERVED	R/W	0h	
6-0	TDCF	R/WQ	0h	发送器延迟补偿滤波器窗口长度。定义 SSP 位置的最小值，测量发送器延迟时，将忽略内部 CAN RX 信号中会导致较早 SSP 位置的显性边沿。当 TDCF 配置为大于 TDCO 的值时，会启用该功能。有效值为 0 至 127 mtq。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。

19.7.1.18 MCAN_IR (偏移 = 7050h) [复位 = 80000000h]

图 19-44 展示了 MCAN_IR，表 19-37 中对此进行了介绍。

返回到汇总表。

当检测到列出的其中一个条件时设置这些标志 (边沿敏感)。这些标志保持设置状态，直到主机将其清除。通过向相应的位位置写入“1”来清除标志。写入“0”不起作用。硬复位将清除寄存器。IE 的配置可控制是否生成中断。ILS 的配置可控制在哪条中断线上发出中断信号。

图 19-44. MCAN_IR

31	30	29	28	27	26	25	24
RESERVED		ARA	PED	PEA	WDI	BO	EW
R/W-2h		R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
23	22	21	20	19	18	17	16
EP	ELO	BEU	RESERVED	DRX	TOO	MRAF	TSW
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
15	14	13	12	11	10	9	8
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
7	6	5	4	3	2	1	0
RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

表 19-37. MCAN_IR 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	2h	
29	ARA	R/W1C	0h	访问保留地址 0 未发生访问保留地址的情况 1 已发生访问保留地址的情况
28	PED	R/W1C	0h	数据段中的协议错误 (使用了数据位时间) 0 数据段中无协议错误 1 检测到数据段中存在协议错误 (PSR.DLEC ? 0,7)
27	PEA	R/W1C	0h	仲裁段中的协议错误 (使用了标称位时间) 0 仲裁段中无协议错误 1 检测到仲裁段中存在协议错误 (PSR.LEC ? 0,7)
26	WDI	R/W1C	0h	看门狗中断 0 未发生消息 RAM 看门狗事件 1 由于缺少 READY 而发生消息 RAM 看门狗事件
25	BO	R/W1C	0h	Bus_Off 状态 0 Bus_Off 状态不变 1 Bus_Off 状态已更改
24	EW	R/W1C	0h	警告状态 0 Error_Warning 状态不变 1 Error_Warning 状态已更改
23	EP	R/W1C	0h	错误被动 0 Error_Passive 状态不变 1 Error_Passive 状态已更改
22	ELO	R/W1C	0h	错误记录溢出 0 CAN 错误记录计数器未溢出 1 CAN 错误记录计数器发生溢出

表 19-37. MCAN_IR 字段说明 (continued)

位	字段	类型	复位	说明
21	BEU	R/W1C	0h	位错误未校正。检测到消息 RAM 位错误，未校正。当附加到消息 RAM 的 ECC 聚合器检测到双位错误时，设置该位。一个未校正的消息 RAM 位错误将 CCCR.INIT 设置为“1”。这样做是为了避免传输损坏的数据。 0 从消息 RAM 读取时未检测到位错误 1 检测到位错误，未校正 (例如奇偶校验逻辑)
20	RESERVED	R/W	0h	
19	DRX	R/W1C	0h	存储到专用 Rx 缓冲区的消息。每当接收到的消息已存储到专用 Rx 缓冲区中时，就会设置该标志。 0 未更新 Rx 缓冲区 1 已将至少一条接收到的消息存储在 Rx 缓冲区中
18	TOO	R/W1C	0h	发生超时 0 无超时 1 达到超时
17	MRAF	R/W1C	0h	消息 RAM 访问失败。当 Rx 处理程序出现以下情况时会设置该标志： - 在接收到后续消息的仲裁字段之前，尚未完成接受过滤或对已接受消息的存储。在这种情况下，接受过滤或消息存储将中止，Rx 处理程序开始处理后续消息。 - 无法将消息写入消息 RAM。在这种情况下，消息存储将中止。 在这两种情况下，均不会更新 FIFO Put 索引，或者不设置专用 Rx 缓冲区的“新数据”标志，当下一条消息存储到该位置时，将覆盖部分存储的消息。 当 Tx 处理程序无法及时从消息 RAM 中读取消息时，也会设置该标志。在这种情况下，消息传输将中止。如果 Tx 处理程序访问失败，MCAN 将切换到受限运行模式。要离开受限运行模式，主机 CPU 必须复位 CCCR.ASM。 0 未发生消息 RAM 访问失败的情况 1 发生消息 RAM 访问失败的情况
16	TSW	R/W1C	0h	时间戳绕回 0 无时间戳计数器绕回 1 时间戳计数器绕回
15	TEFL	R/W1C	0h	Tx 事件 FIFO 元素丢失 0 Tx 事件 FIFO 元素未丢失 1 Tx 事件 FIFO 元素丢失，也会在尝试写入大小为零的 Tx 事件 FIFO 后设置
14	TEFF	R/W1C	0h	Tx 事件 FIFO 已满 0 Tx 事件 FIFO 未满 1 Tx 事件 FIFO 已满
13	TEFW	R/W1C	0h	Tx 事件 FIFO 达到水线 0 Tx 事件 FIFO 填充级别低于水线 1 Tx 事件 FIFO 填充级别达到水线
12	TEFN	R/W1C	0h	Tx 事件 FIFO 新条目 0 Tx 事件 FIFO 不变 1 Tx 处理程序已写入 Tx 事件 FIFO 元素
11	TFE	R/W1C	0h	Tx FIFO 为空 0 Tx FIFO 非空 1 Tx FIFO 为空
10	TCF	R/W1C	0h	完成传输取消 0 未完成传输取消 1 已完成传输取消
9	TC	R/W1C	0h	完成传输 0 未完成传输 1 已完成传输
8	HPM	R/W1C	0h	高优先级消息 0 未接收到高优先级消息 1 已接收到高优先级消息

表 19-37. MCAN_IR 字段说明 (continued)

位	字段	类型	复位	说明
7	RF1L	R/W1C	0h	Rx FIFO 1 消息丢失 0 Rx FIFO 1 消息未丢失 1 Rx FIFO 1 消息丢失，也会在尝试写入大小为零的 Rx FIFO 1 后设置
6	RF1F	R/W1C	0h	Rx FIFO 1 已满 0 Rx FIFO 1 未满 1 Rx FIFO 1 已满
5	RF1W	R/W1C	0h	Rx FIFO 1 达到水线 0 Rx FIFO 1 填充级别低于水线 1 Rx FIFO 1 填充级别达到水线
4	RF1N	R/W1C	0h	Rx FIFO 1 新消息 0 无新消息写入 Rx FIFO 1 1 有新消息写入 Rx FIFO 1
3	RF0L	R/W1C	0h	Rx FIFO 0 消息丢失 0 Rx FIFO 0 消息未丢失 1 Rx FIFO 0 消息丢失，也会在尝试写入大小为零的 Rx FIFO 0 后设置
2	RF0F	R/W1C	0h	Rx FIFO 0 已满 0 Rx FIFO 0 未满 1 Rx FIFO 0 已满
1	RF0W	R/W1C	0h	Rx FIFO 0 达到水线 0 Rx FIFO 0 填充级别低于水线 1 Rx FIFO 0 填充级别达到水线
0	RF0N	R/W1C	0h	Rx FIFO 0 新消息 0 无新消息写入 Rx FIFO 0 1 有新消息写入 Rx FIFO 0

19.7.1.19 MCAN_IE（偏移 = 7054h）[复位 = 00000000h]

图 19-45 展示了 MCAN_IE，表 19-38 中对此进行了介绍。

返回到汇总表。

MCAN 中断使能

图 19-45. MCAN_IE

31	30	29	28	27	26	25	24
RESERVED		ARAE	PEDE	PEAE	WDIE	BOE	EWE
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 19-38. MCAN_IE 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29	ARAE	R/W	0h	访问保留地址使能
28	PEDE	R/W	0h	数据段中的协议错误使能
27	PEAE	R/W	0h	仲裁段中的协议错误使能
26	WDIE	R/W	0h	看门狗中断使能
25	BOE	R/W	0h	Bus_Off 状态使能
24	EWE	R/W	0h	警告状态使能
23	EPE	R/W	0h	错误被动使能
22	ELOE	R/W	0h	错误记录溢出使能
21	BEUE	R/W	0h	位错误未校正使能
20	BECE	R/W	0h	位错误已纠正使能 通过 MCAN_ERROR_REGS 提供为已纠正位错误保留的单独中断线。建议用户使用这些寄存器并将该位清除为“0”。
19	DRXE	R/W	0h	存储到专用 Rx 缓冲区的消息使能
18	TOOE	R/W	0h	发生超时使能
17	MRAFE	R/W	0h	消息 RAM 访问失败使能
16	TSWE	R/W	0h	时间戳绕回使能
15	TEFLE	R/W	0h	Tx 事件 FIFO 元素丢失使能
14	TEFFE	R/W	0h	Tx 事件 FIFO 已满使能
13	TEFWE	R/W	0h	Tx 事件 FIFO 达到水线使能
12	TEFNE	R/W	0h	Tx 事件 FIFO 新条目使能
11	TFEE	R/W	0h	Tx FIFO 为空使能
10	TCFE	R/W	0h	完成传输取消使能
9	TCE	R/W	0h	完成传输使能
8	HPME	R/W	0h	高优先级消息使能

表 19-38. MCAN_IE 字段说明 (continued)

位	字段	类型	复位	说明
7	RF1LE	R/W	0h	Rx FIFO 1 消息丢失使能
6	RF1FE	R/W	0h	Rx FIFO 1 已满使能
5	RF1WE	R/W	0h	Rx FIFO 1 达到水线使能
4	RF1NE	R/W	0h	Rx FIFO 1 新消息使能
3	RF0LE	R/W	0h	Rx FIFO 0 消息丢失使能
2	RF0FE	R/W	0h	Rx FIFO 0 已满使能
1	RF0WE	R/W	0h	Rx FIFO 0 达到水线使能
0	RF0NE	R/W	0h	Rx FIFO 0 新消息使能

19.7.1.20 MCAN_ILS（偏移 = 7058h）[复位 = 00000000h]

图 19-46 展示了 MCAN_ILS，表 19-39 中对此进行了介绍。

返回到汇总表。

中断线选择寄存器将中断寄存器中一个由特定中断标志生成的中断分配给两条模块中断线之一。为了生成中断，必须通过 ILE.EINT0 和 ILE.EINT1 启用相应的中断线。

图 19-46. MCAN_ILS

31	30	29	28	27	26	25	24
RESERVED		ARAL	PEDL	PEAL	WDIL	BOL	EWL
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
EPL	ELOL	BEUL	BECL	DRXL	工具	MRAFL	TSWL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 19-39. MCAN_ILS 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29	ARAL	R/W	0h	访问保留地址行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
28	PEDL	R/W	0h	数据段中的协议错误行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
27	PEAL	R/W	0h	仲裁段中的协议错误行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
26	WDIL	R/W	0h	看门狗中断行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
25	BOL	R/W	0h	Bus_Off 状态行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
24	EWL	R/W	0h	警告状态行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
23	EPL	R/W	0h	错误被动行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
22	ELOL	R/W	0h	错误记录溢出行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
21	BEUL	R/W	0h	位错误未校正行 0 中断源分配给中断线 0 1 中断源分配给中断线 1

表 19-39. MCAN_ILS 字段说明 (continued)

位	字段	类型	复位	说明
20	BECL	R/W	0h	位错误已纠正行 通过 MCAN_ERROR_REGS 提供为已纠正位错误保留的单独中断线。建议用户使用这些寄存器并将 MCAN_IE.BECE 位清除为“0”（禁用），从而将该位降级为不适用。
19	DRXL	R/W	0h	存储到专用 Rx 缓冲区的消息行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
18	工具	R/W	0h	发生超时行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
17	MRAFL	R/W	0h	消息 RAM 访问失败行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
16	TSWL	R/W	0h	时间戳绕回行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
15	TEFLL	R/W	0h	Tx 事件 FIFO 元素丢失行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
14	TEFFL	R/W	0h	Tx 事件 FIFO 已满行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
13	TEFWL	R/W	0h	Tx 事件 FIFO 达到水线行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
12	TEFNL	R/W	0h	Tx 事件 FIFO 新条目行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
11	TFEL	R/W	0h	Tx FIFO 为空行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
10	TCFL	R/W	0h	完成传输取消行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
9	TCL	R/W	0h	完成传输行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
8	HPML	R/W	0h	高优先级消息行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
7	RF1LL	R/W	0h	Rx FIFO 1 消息丢失行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
6	RF1FL	R/W	0h	Rx FIFO 1 已满行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
5	RF1WL	R/W	0h	Rx FIFO 1 达到水线行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
4	RF1NL	R/W	0h	Rx FIFO 1 新消息行 0 中断源分配给中断线 0 1 中断源分配给中断线 1

表 19-39. MCAN_ILS 字段说明 (continued)

位	字段	类型	复位	说明
3	RF0LL	R/W	0h	Rx FIFO 0 消息丢失行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
2	RF0FL	R/W	0h	Rx FIFO 0 已满行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
1	RF0WL	R/W	0h	Rx FIFO 0 达到水线行 0 中断源分配给中断线 0 1 中断源分配给中断线 1
0	RF0NL	R/W	0h	Rx FIFO 0 新消息行 0 中断源分配给中断线 0 1 中断源分配给中断线 1

19.7.1.21 MCAN_ILE（偏移 = 705Ch）[复位 = 00000000h]

图 19-47 展示了 MCAN_ILE，表 19-40 中对此进行了介绍。

返回到汇总表。

MCAN 中断线使能

图 19-47. MCAN_ILE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						EINT1	EINT0
R/W-0h						R/W-0h	R/W-0h

表 19-40. MCAN_ILE 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	EINT1	R/W	0h	启用中断线 1 0 中断线 1 已禁用 1 中断线 1 已启用
0	EINT0	R/W	0h	启用中断线 0 0 中断线 0 已禁用 1 中断线 0 已启用

19.7.1.22 MCAN_GFC（偏移 = 7080h）[复位 = 00000000h]

图 19-48 展示了 MCAN_GFC，表 19-41 中对此进行了介绍。

返回到汇总表。

MCAN 全局过滤器配置

图 19-48. MCAN_GFC

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		ANFS		ANFE		RRFS	RRFE
R/W-0h		R/WQ-0h		R/WQ-0h		R/WQ-0h	R/WQ-0h

表 19-41. MCAN_GFC 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	ANFS	R/WQ	0h	接受不匹配的标准帧。定义如何处理接收到的具有 11 位 ID 且与过滤器列表中任何元素均不匹配的消息。 00 在 Rx FIFO 0 中接受 01 在 Rx FIFO 1 中接受 10 拒绝 11 拒绝 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
3-2	ANFE	R/WQ	0h	接受不匹配的扩展帧。定义如何处理接收到的具有 29 位 ID 且与过滤器列表中任何元素均不匹配的消息。 00 在 Rx FIFO 0 中接受 01 在 Rx FIFO 1 中接受 10 拒绝 11 拒绝 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
1	RRFS	R/WQ	0h	拒绝远程标准帧 0 过滤具有 11 位标准 ID 的远程帧 1 拒绝所有具有 11 位标准 ID 的远程帧 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
0	RRFE	R/WQ	0h	拒绝远程扩展帧 0 过滤具有 29 位扩展 ID 的远程帧 1 拒绝所有具有 29 位扩展 ID 的远程帧 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

19.7.1.23 MCAN_SIDFC（偏移 = 7084h）[复位 = 00000000h]

图 19-49 展示了 MCAN_SIDFC，表 19-42 中对此进行了介绍。

返回到汇总表。

MCAN 标准 ID 过滤器配置

图 19-49. MCAN_SIDFC

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
LSS							
R/WQ-0h							
15	14	13	12	11	10	9	8
FLSSA							
R/WQ-0h							
7	6	5	4	3	2	1	0
FLSSA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-42. MCAN_SIDFC 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23-16	LSS	R/WQ	0h	列表大小标准 0 无标准消息 ID 过滤器 1-128 标准消息 ID 过滤器元素数量 >128 大于 128 的值视为 128
15-2	FLSSA	R/WQ	0h	过滤器列表标准起始地址。标准消息 ID 过滤器列表的起始地址（32 位字地址）。
1-0	保留	R/W	0h	

19.7.1.24 MCAN_XIDFC（偏移 = 7088h）[复位 = 00000000h]

图 19-50 展示了 MCAN_XIDFC，表 19-43 中对此进行了介绍。

返回到汇总表。

MCAN 扩展 ID 过滤器配置

图 19-50. MCAN_XIDFC

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED	LSE						
R/W-0h	R/WQ-0h						
15	14	13	12	11	10	9	8
FLESA							
R/WQ-0h							
7	6	5	4	3	2	1	0
FLESA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-43. MCAN_XIDFC 字段说明

位	字段	类型	复位	说明
31-23	保留	R/W	0h	
22-16	LSE	R/WQ	0h	过滤器列表扩展起始地址。扩展消息 ID 过滤器列表的起始地址（32 位字地址）。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
15-2	FLESA	R/WQ	0h	列表大小扩展 0 无扩展消息 ID 过滤器 1-64 扩展消息 ID 过滤器元素数量 >64 大于 64 的值视为 64 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
1-0	保留	R/W	0h	

19.7.1.25 MCAN_XIDAM（偏移 = 7090h）[复位 = 1FFFFFFFh]

图 19-51 展示了 MCAN_XIDAM，表 19-44 中对此进行了介绍。

返回到汇总表。

MCAN 扩展 ID 和掩码

图 19-51. MCAN_XIDAM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED			EIDM																												
R/W-0h			R/WQ-1FFFFFFFh																												

表 19-44. MCAN_XIDAM 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28-0	EIDM	R/WQ	1FFFFFFFh	扩展 ID 掩码。对于扩展帧的接受过滤，扩展 ID 和掩码与接收帧的消息 ID 进行“与”运算。用于屏蔽 SAE J1939 中的 29 位 ID。当所有位的复位值均设置为 1 时，掩码不处于活动状态。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。

19.7.1.26 MCAN_HPMS（偏移 = 7094h）[复位 = 00000000h]

图 19-52 展示了 MCAN_HPMS，表 19-45 中对此进行了介绍。

返回到汇总表。

每当消息 ID 过滤器元素配置为生成优先级事件匹配时，该寄存器都会更新。这可用于监测传入的高优先级消息的状态，并实现对这些消息的快速访问。

图 19-52. MCAN_HPMS

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

表 19-45. MCAN_HPMS 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15	FLST	R	0h	过滤器列表。表示匹配的过滤器元素的过滤器列表。 0 标准过滤器列表 1 扩展过滤器列表
14-8	FIDX	R	0h	过滤器索引。匹配的过滤器元素的索引。范围为 0 至 SIDFC.LSS - 1 或 XIDFC.LSE - 1。
7-6	MSI	R	0h	消息存储指示器 00 未选择 FIFO 01 FIFO 消息丢失 10 消息存储在 FIFO 0 中 11 消息存储在 FIFO 1 中
5-0	BIDX	R	0h	缓冲区索引。存储消息的 Rx FIFO 元素的索引。仅当 MSI(1) = “1” 时有效。

19.7.1.27 MCAN_NDAT1（偏移 = 7098h）[复位 = 00000000h]

图 19-53 展示了 MCAN_NDAT1，表 19-46 中对此进行了介绍。

返回到汇总表。

MCAN 新数据 1

图 19-53. MCAN_NDAT1

31	30	29	28	27	26	25	24
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
23	22	21	20	19	18	17	16
ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
15	14	13	12	11	10	9	8
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
7	6	5	4	3	2	1	0
ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

表 19-46. MCAN_NDAT1 字段说明

位	字段	类型	复位	说明
31	ND31	R/W1C	0h	新数据 RX 缓冲区 31 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
30	ND30	R/W1C	0h	新数据 RX 缓冲区 30 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
29	ND29	R/W1C	0h	新数据 RX 缓冲区 29 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
28	ND28	R/W1C	0h	新数据 RX 缓冲区 28 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
27	ND27	R/W1C	0h	新数据 RX 缓冲区 27 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
26	ND26	R/W1C	0h	新数据 RX 缓冲区 26 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
25	ND25	R/W1C	0h	新数据 RX 缓冲区 25 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
24	ND24	R/W1C	0h	新数据 RX 缓冲区 24 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
23	ND23	R/W1C	0h	新数据 RX 缓冲区 23 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
22	ND22	R/W1C	0h	新数据 RX 缓冲区 22 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

表 19-46. MCAN_NDAT1 字段说明 (continued)

位	字段	类型	复位	说明
21	ND21	R/W1C	0h	新数据 RX 缓冲区 21 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
20	ND20	R/W1C	0h	新数据 RX 缓冲区 20 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
19	ND19	R/W1C	0h	新数据 RX 缓冲区 19 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
18	ND18	R/W1C	0h	新数据 RX 缓冲区 18 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
17	ND17	R/W1C	0h	新数据 RX 缓冲区 17 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
16	ND16	R/W1C	0h	新数据 RX 缓冲区 16 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
15	ND15	R/W1C	0h	新数据 RX 缓冲区 15 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
14	ND14	R/W1C	0h	新数据 RX 缓冲区 14 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
13	ND13	R/W1C	0h	新数据 RX 缓冲区 13 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
12	ND12	R/W1C	0h	新数据 RX 缓冲区 12 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
11	ND11	R/W1C	0h	新数据 RX 缓冲区 11 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
10	ND10	R/W1C	0h	新数据 RX 缓冲区 10 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
9	ND9	R/W1C	0h	新数据 RX 缓冲区 9 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
8	ND8	R/W1C	0h	新数据 RX 缓冲区 8 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
7	ND7	R/W1C	0h	新数据 RX 缓冲区 7 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
6	ND6	R/W1C	0h	新数据 RX 缓冲区 6 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
5	ND5	R/W1C	0h	新数据 RX 缓冲区 5 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

表 19-46. MCAN_NDAT1 字段说明 (continued)

位	字段	类型	复位	说明
4	ND4	R/W1C	0h	新数据 RX 缓冲区 4 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
3	ND3	R/W1C	0h	新数据 RX 缓冲区 3 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
2	ND2	R/W1C	0h	新数据 RX 缓冲区 2 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
1	ND1	R/W1C	0h	新数据 RX 缓冲区 1 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
0	ND0	R/W1C	0h	新数据 RX 缓冲区 0 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

19.7.1.28 MCAN_NDAT2（偏移 = 709Ch）[复位 = 00000000h]

图 19-54 展示了 MCAN_NDAT2，表 19-47 中对此进行了介绍。

返回到汇总表。

MCAN 新数据 2

图 19-54. MCAN_NDAT2

31	30	29	28	27	26	25	24
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
23	22	21	20	19	18	17	16
ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
15	14	13	12	11	10	9	8
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h
7	6	5	4	3	2	1	0
ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h	R/W1C-0h

表 19-47. MCAN_NDAT2 字段说明

位	字段	类型	复位	说明
31	ND63	R/W1C	0h	新数据 RX 缓冲区 63 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
30	ND62	R/W1C	0h	新数据 RX 缓冲区 62 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
29	ND61	R/W1C	0h	新数据 RX 缓冲区 61 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
28	ND60	R/W1C	0h	新数据 RX 缓冲区 60 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
27	ND59	R/W1C	0h	新数据 RX 缓冲区 59 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
26	ND58	R/W1C	0h	新数据 RX 缓冲区 58 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
25	ND57	R/W1C	0h	新数据 RX 缓冲区 57 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
24	ND56	R/W1C	0h	新数据 RX 缓冲区 56 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
23	ND55	R/W1C	0h	新数据 RX 缓冲区 55 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
22	ND54	R/W1C	0h	新数据 RX 缓冲区 54 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

表 19-47. MCAN_NDAT2 字段说明 (continued)

位	字段	类型	复位	说明
21	ND53	R/W1C	0h	新数据 RX 缓冲区 53 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
20	ND52	R/W1C	0h	新数据 RX 缓冲区 52 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
19	ND51	R/W1C	0h	新数据 RX 缓冲区 51 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
18	ND50	R/W1C	0h	新数据 RX 缓冲区 50 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
17	ND49	R/W1C	0h	新数据 RX 缓冲区 49 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
16	ND48	R/W1C	0h	新数据 RX 缓冲区 48 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
15	ND47	R/W1C	0h	新数据 RX 缓冲区 47 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
14	ND46	R/W1C	0h	新数据 RX 缓冲区 46 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
13	ND45	R/W1C	0h	新数据 RX 缓冲区 45 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
12	ND44	R/W1C	0h	新数据 RX 缓冲区 44 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
11	ND43	R/W1C	0h	新数据 RX 缓冲区 43 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
10	ND42	R/W1C	0h	新数据 RX 缓冲区 42 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
9	ND41	R/W1C	0h	新数据 RX 缓冲区 41 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
8	ND40	R/W1C	0h	新数据 RX 缓冲区 40 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
7	ND39	R/W1C	0h	新数据 RX 缓冲区 39 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
6	ND38	R/W1C	0h	新数据 RX 缓冲区 38 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
5	ND37	R/W1C	0h	新数据 RX 缓冲区 37 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

表 19-47. MCAN_NDAT2 字段说明 (continued)

位	字段	类型	复位	说明
4	ND36	R/W1C	0h	新数据 RX 缓冲区 36 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
3	ND35	R/W1C	0h	新数据 RX 缓冲区 35 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
2	ND34	R/W1C	0h	新数据 RX 缓冲区 34 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
1	ND33	R/W1C	0h	新数据 RX 缓冲区 33 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区
0	ND32	R/W1C	0h	新数据 RX 缓冲区 32 0 Rx 缓冲区未更新 1 新消息更新了 Rx 缓冲区

19.7.1.29 MCAN_RXF0C（偏移 = 70A0h）[复位 = 00000000h]

图 19-55 展示了 MCAN_RXF0C，表 19-48 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 0 配置

图 19-55. MCAN_RXF0C

31	30	29	28	27	26	25	24
F0OM	F0WM						
R/WQ-0h	R/WQ-0h						
23	22	21	20	19	18	17	16
RESERVED	F0S						
R/W-0h	R/WQ-0h						
15	14	13	12	11	10	9	8
F0SA							
R/WQ-0h							
7	6	5	4	3	2	1	0
F0SA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-48. MCAN_RXF0C 字段说明

位	字段	类型	复位	说明
31	F0OM	R/WQ	0h	FIFO 0 运行模式。FIFO 0 可在阻塞模式或覆盖模式下运行。 0 FIFO 0 阻塞模式 1 FIFO 0 覆盖模式 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
30-24	F0WM	R/WQ	0h	Rx FIFO 0 水线 0 禁用水线中断 1-64 Rx FIFO 0 水线中断 (IR.RF0W) 级别 >64 禁用水线中断 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
23	RESERVED	R/W	0h	
22-16	F0S	R/WQ	0h	Rx FIFO 0 大小。Rx FIFO 0 元素的索引范围为 0 至 F0S - 1。 0 无 Rx FIFO 0 1-64 Rx FIFO 0 元素数量 >64 大于 64 的值视为 64 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
15-2	F0SA	R/WQ	0h	Rx FIFO 0 起始地址。消息 RAM 中 Rx FIFO 0 的起始地址（32 位字地址）。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
1-0	保留	R/W	0h	

19.7.1.30 MCAN_RXF0S（偏移 = 70A4h）[复位 = 00000000h]

图 19-56 展示了 MCAN_RXF0S，表 19-49 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 0 状态

图 19-56. MCAN_RXF0S

31	30	29	28	27	26	25	24
RESERVED						RF0L	F0F
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED		F0PI					
R-0h		R-0h					
15	14	13	12	11	10	9	8
保留		F0GI					
R-0h		R-0h					
7	6	5	4	3	2	1	0
RESERVED	F0FL						
R-0h	R-0h						

表 19-49. MCAN_RXF0S 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	RF0L	R	0h	Rx FIFO 0 消息丢失。该位是中断标志 IR.RF0L 的副本。当 IR.RF0L 复位时，该位也会复位。 0 Rx FIFO 0 消息未丢失 1 Rx FIFO 0 消息丢失，也会在尝试写入大小为零的 Rx FIFO 0 后设置 注意：当 RXF0C.F0OM =“1”时，覆盖最早的消息将不会设置该标志。
24	F0F	R	0h	Rx FIFO 0 已满 0 Rx FIFO 0 未满 1 Rx FIFO 0 已满
23-22	RESERVED	R	0h	
21-16	F0PI	R	0h	Rx FIFO 0 Put 索引。Rx FIFO 0 写索引指针，范围为 0 至 63。
15-14	保留	R	0h	
13-8	F0GI	R	0h	Rx FIFO 0 Get 索引。Rx FIFO 0 读索引指针，范围为 0 至 63。
7	RESERVED	R	0h	
6-0	F0FL	R	0h	Rx FIFO 0 填充级别。存储在 Rx FIFO 0 中的元素数量，范围为 0 至 64。

19.7.1.31 MCAN_RXF0A（偏移 = 70A8h）[复位 = 00000000h]

图 19-57 展示了 MCAN_RXF0A，表 19-50 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 0 确认

图 19-57. MCAN_RXF0A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										F0AI					
R/W-0h																										R/W-0h					

表 19-50. MCAN_RXF0A 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-0	F0AI	R/W	0h	Rx FIFO 0 确认索引。主机从 Rx FIFO 0 读取一条消息或一系列消息后，必须将从 Rx FIFO 0 读取的最后一个元素的缓冲区索引写入 F0AI。这会将 Rx FIFO 0 Get 索引 RXF0S.F0GI 设置为 F0AI + 1，并更新 FIFO 0 填充级别 RXF0S.F0FL。

19.7.1.32 MCAN_RXBC（偏移 = 70ACh）[复位 = 00000000h]

图 19-58 展示了 MCAN_RXBC，表 19-51 中对此进行了介绍。

返回到汇总表。

MCAN Rx 缓冲区配置

图 19-58. MCAN_RXBC

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

表 19-51. MCAN_RXBC 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-2	RBSA	R/WQ	0h	Rx 缓冲区起始地址。配置消息 RAM 中 Rx 缓冲区部分的起始地址（32 位字地址）。 +I466
1-0	保留	R/W	0h	

19.7.1.33 MCAN_RXF1C（偏移 = 70B0h）[复位 = 00000000h]

图 19-59 展示了 MCAN_RXF1C，表 19-52 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 1 配置

图 19-59. MCAN_RXF1C

31	30	29	28	27	26	25	24
F1OM	F1WM						
R/WQ-0h	R/WQ-0h						
23	22	21	20	19	18	17	16
RESERVED	F1S						
R/W-0h	R/WQ-0h						
15	14	13	12	11	10	9	8
F1SA							
R/WQ-0h							
7	6	5	4	3	2	1	0
F1SA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-52. MCAN_RXF1C 字段说明

位	字段	类型	复位	说明
31	F1OM	R/WQ	0h	FIFO 1 运行模式。FIFO 1 可在阻塞模式或覆盖模式下运行。 0 FIFO 1 阻塞模式 1 FIFO 1 覆盖模式 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
30-24	F1WM	R/WQ	0h	Rx FIFO 1 水线 0 禁用水线中断 1-64 Rx FIFO 1 水线中断 (IR.RF1W) 级别 >64 禁用水线中断 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
23	RESERVED	R/W	0h	
22-16	F1S	R/WQ	0h	Rx FIFO 1 大小。Rx FIFO 1 元素的索引范围为 0 至 F1S - 1。 0 无 Rx FIFO 1 1-64 Rx FIFO 1 元素数量 >64 大于 64 的值视为 64 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。
15-2	F1SA	R/WQ	0h	Rx FIFO 1 起始地址。消息 RAM 中 Rx FIFO 1 的起始地址（32 位字地址）。
1-0	保留	R/W	0h	

19.7.1.34 MCAN_RXF1S（偏移 = 70B4h）[复位 = 00000000h]

图 19-60 展示了 MCAN_RXF1S，表 19-53 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 1 状态

图 19-60. MCAN_RXF1S

31	30	29	28	27	26	25	24
DMS		RESERVED				RF1L	F1F
R-0h		R-0h				R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED		F1PI					
R-0h		R-0h					
15	14	13	12	11	10	9	8
保留		F1GI					
R-0h		R-0h					
7	6	5	4	3	2	1	0
RESERVED	F1FL						
R-0h	R-0h						

表 19-53. MCAN_RXF1S 字段说明

位	字段	类型	复位	说明
31-30	DMS	R	0h	调试消息状态 00 空闲状态，等待接收调试消息 01 收到调试消息 A 10 收到调试消息 A、B 11 收到调试消息 A、B、C
29-26	保留	R	0h	
25	RF1L	R	0h	Rx FIFO 1 消息丢失。该位是中断标志 IR.RF1L 的副本。当 IR.RF1L 复位时，该位也会复位。 0 Rx FIFO 1 消息未丢失 1 Rx FIFO 1 消息丢失，也会在尝试写入大小为零的 Rx FIFO 1 后设置 注意：当 RXF1C.F1OM = “1” 时，覆盖最早的消息将不会设置该标志。
24	F1F	R	0h	Rx FIFO 1 已满 0 Rx FIFO 1 未满 1 Rx FIFO 1 已满
23-22	RESERVED	R	0h	
21-16	F1PI	R	0h	Rx FIFO 1 Put 索引。Rx FIFO 1 写索引指针，范围为 0 至 63。
15-14	保留	R	0h	
13-8	F1GI	R	0h	Rx FIFO 1 Get 索引。Rx FIFO 1 读索引指针，范围为 0 至 63。
7	RESERVED	R	0h	
6-0	F1FL	R	0h	Rx FIFO 1 填充级别。存储在 Rx FIFO 1 中的元素数量，范围为 0 至 64。

19.7.1.35 MCAN_RXF1A（偏移 = 70B8h）[复位 = 00000000h]

图 19-61 展示了 MCAN_RXF1A，表 19-54 中对此进行了介绍。

返回到汇总表。

MCAN Rx FIFO 1 确认

图 19-61. MCAN_RXF1A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																										F1AI					
R/W-0h																										R/W-0h					

表 19-54. MCAN_RXF1A 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-0	F1AI	R/W	0h	Rx FIFO 1 确认索引。主机从 Rx FIFO 1 读取一条消息或一系列消息后，必须将从 Rx FIFO 1 读取的最后一个元素的缓冲区索引写入 F1AI。这会将 Rx FIFO 1 Get 索引 RXF1S.F1GI 设置为 F1AI + 1，并更新 FIFO 1 填充级别 RXF1S.F1FL。

19.7.1.36 MCAN_RXESC（偏移 = 70BCh）[复位 = 00000000h]

图 19-62 展示了 MCAN_RXESC，表 19-55 中对此进行了介绍。

返回到汇总表。

配置属于 Rx 缓冲区/Rx FIFO 元素的数据字节数。>8 字节的数据字段大小仅用于 CAN FD 运行。

图 19-62. MCAN_RXESC

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留						RBDS	
R/W-0h						R/WQ-0h	
7	6	5	4	3	2	1	0
RESERVED	F1DS			RESERVED	F0DS		
R/W-0h	R/WQ-0h			R/W-0h	R/WQ-0h		

表 19-55. MCAN_RXESC 字段说明

位	字段	类型	复位	说明
31-11	保留	R/W	0h	
10-8	RBDS	R/WQ	0h	Rx 缓冲区数据字段大小 000 8 字节数据字段 001 12 字节数据字段 010 16 字节数据字段 011 20 字节数据字段 100 24 字节数据字段 101 32 字节数据字段 110 48 字节数据字段 111 64 字节数据字段 注意：如果接受的 CAN 帧的数据字段大小超过为匹配的 Rx 缓冲区或 Rx FIFO 配置的数据字段大小，则仅将 RXESC 配置的字节数存储到 Rx 缓冲区或 Rx FIFO 元素。帧数据字段的其余部分将被忽略。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
7	RESERVED	R/W	0h	
6-4	F1DS	R/WQ	0h	Rx FIFO 1 数据字段大小 000 8 字节数据字段 001 12 字节数据字段 010 16 字节数据字段 011 20 字节数据字段 100 24 字节数据字段 101 32 字节数据字段 110 48 字节数据字段 111 64 字节数据字段 注意：如果接受的 CAN 帧的数据字段大小超过为匹配的 Rx 缓冲区或 Rx FIFO 配置的数据字段大小，则仅将 RXESC 配置的字节数存储到 Rx 缓冲区或 Rx FIFO 元素。帧数据字段的其余部分将被忽略。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
3	RESERVED	R/W	0h	

表 19-55. MCAN_RXESC 字段说明 (continued)

位	字段	类型	复位	说明
2-0	F0DS	R/WQ	0h	Rx FIFO 0 数据字段大小 000 8 字节数据字段 001 12 字节数据字段 010 16 字节数据字段 011 20 字节数据字段 100 24 字节数据字段 101 32 字节数据字段 110 48 字节数据字段 111 64 字节数据字段 注意：如果接受的 CAN 帧的数据字段大小超过为匹配的 Rx 缓冲区或 Rx FIFO 配置的数据字段大小，则仅将 RXESC 配置的字节数存储到 Rx 缓冲区或 Rx FIFO 元素。帧数据字段的其余部分将被忽略。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。

19.7.1.37 MCAN_TXBC（偏移 = 70C0h）[复位 = 00000000h]

图 19-63 展示了 MCAN_TXBC，表 19-56 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区配置

图 19-63. MCAN_TXBC

31	30	29	28	27	26	25	24
RESERVED	TFQM	TFQS					
R/W-0h	R/WQ-0h	R/WQ-0h					
23	22	21	20	19	18	17	16
RESERVED		NDTB					
R/W-0h		R/WQ-0h					
15	14	13	12	11	10	9	8
TBSA							
R/WQ-0h							
7	6	5	4	3	2	1	0
TBSA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-56. MCAN_TXBC 字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30	TFQM	R/WQ	0h	Tx FIFO/队列模式 0 Tx FIFO 运行 1 Tx 队列运行 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
29-24	TFQS	R/WQ	0h	发送 FIFO/队列大小 0 无 Tx FIFO/队列 1-32 用于 Tx FIFO/队列的 Tx 缓冲区数量 >32 大于 32 的值视为 32 注意：请注意，TFQS 和 NDTB 之和不得大于 32。不会检查错误配置。消息 RAM 中的 Tx 缓冲区部分从专用的 Tx 缓冲区开始。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
23-22	保留	R/W	0h	
21-16	NDTB	R/WQ	0h	专用发送缓冲区数量 0 无专用 Tx 缓冲区 1-32 专用 Tx 缓冲区数量 >32 大于 32 的值视为 32 注意：请注意，TFQS 和 NDTB 之和不得大于 32。不会检查错误配置。消息 RAM 中的 Tx 缓冲区部分从专用的 Tx 缓冲区开始。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
15-2	TBSA	R/WQ	0h	Tx 缓冲区起始地址。消息 RAM 中 Tx 缓冲区部分的起始地址（32 位字地址）。 仅当 CCCR.CCE = “1” 且 CCCR.INIT = “1” 时，才能进行限定写入。
1-0	保留	R/W	0h	

19.7.1.38 MCAN_TXFQS（偏移 = 70C4h）[复位 = 00000000h]

图 19-64 展示了 MCAN_TXFQS，表 19-57 中对此进行了介绍。

返回到汇总表。

Tx FIFO/队列状态与寄存器 TXBRP 中列出的待处理 Tx 请求相关。因此，由于正在运行 Tx 扫描（TXBRP 尚未更新），添加/取消请求的效果可能会延迟。

图 19-64. MCAN_TXFQS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED		TFQF	TFQP				
R-0h		R-0h	R-0h				
15	14	13	12	11	10	9	8
保留			TFGI				
R-0h			R-0h				
7	6	5	4	3	2	1	0
RESERVED		TFFL					
R-0h		R-0h					

表 19-57. MCAN_TXFQS 字段说明

位	字段	类型	复位	说明
31-22	保留	R	0h	
21	TFQF	R	0h	Tx FIFO/队列已满 0 Tx FIFO/队列未满 1 Tx FIFO/队列已满
20-16	TFQP	R	0h	Tx FIFO/队列 Put 索引。Tx FIFO/队列写索引指针，范围为 0 至 31。 注意：在专用 Tx 缓冲区与 Tx FIFO 或 Tx 队列组合的混合配置情况下，Put 和 Get 索引将指示从第一个专用 Tx 缓冲区开始的 Tx 缓冲区数量。示例：对于 12 个专用 Tx 缓冲区和含 20 个缓冲区的 Tx FIFO 的配置，Put 索引 (15) 指向 Tx FIFO 的第四个缓冲区。
15-13	保留	R	0h	
12-8	TFGI	R	0h	Tx FIFO Get 索引。Tx FIFO 读索引指针，范围为 0 至 31。配置 Tx 队列运行（TXBC.TFQM =“1”）时读取为零。 注意：在专用 Tx 缓冲区与 Tx FIFO 或 Tx 队列组合的混合配置情况下，Put 和 Get 索引将指示从第一个专用 Tx 缓冲区开始的 Tx 缓冲区数量。示例：对于 12 个专用 Tx 缓冲区和含 20 个缓冲区的 Tx FIFO 的配置，Put 索引 (15) 指向 Tx FIFO 的第四个缓冲区。
7-6	RESERVED	R	0h	
5-0	TFFL	R	0h	Tx FIFO 空闲级别。从 TFGI 开始的连续空闲 Tx FIFO 元素数量，范围为 0 至 32。配置 Tx 队列运行（TXBC.TFQM =“1”）时读取为零。

19.7.1.39 MCAN_TXESC（偏移 = 70C8h）[复位 = 00000000h]

图 19-65 展示了 MCAN_TXESC，表 19-58 中对此进行了介绍。

返回到汇总表。

配置属于 Tx 缓冲区元素的数据字节数。>8 字节的数据字段大小仅用于 CAN FD 运行。

图 19-65. MCAN_TXESC

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

表 19-58. MCAN_TXESC 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	TBDS	R/WQ	0h	Tx 缓冲区数据字段大小 000 8 字节数据字段 001 12 字节数据字段 010 16 字节数据字段 011 20 字节数据字段 100 24 字节数据字段 101 32 字节数据字段 110 48 字节数据字段 111 64 字节数据字段 注意：如果 Tx 缓冲区元素的数据长度代码 (DLC) 配置为一个高于 Tx 缓冲区数据字段大小 TXESC.TBDS 的值，则未由 Tx 缓冲区定义的字节将作为“0xCC”（填充字节）传输。 仅当 CCCR.CCE =“1”且 CCCR.INIT =“1”时，才能进行限定写入。

19.7.1.40 MCAN_TXBRP（偏移 = 70CCh）[复位 = 00000000h]

图 19-66 展示了 MCAN_TXBRP，表 19-59 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区请求待处理

图 19-66. MCAN_TXBRP

31	30	29	28	27	26	25	24
TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 19-59. MCAN_TXBRP 字段说明

位	字段	类型	复位	说明
31	TRP31	R	0h	传输请求待处理 31。请参阅位 0 的说明。
30	TRP30	R	0h	传输请求待处理 30。请参阅位 0 的说明。
29	TRP29	R	0h	传输请求待处理 29。请参阅位 0 的说明。
28	TRP28	R	0h	传输请求待处理 28。请参阅位 0 的说明。
27	TRP27	R	0h	传输请求待处理 27。请参阅位 0 的说明。
26	TRP26	R	0h	传输请求待处理 26。请参阅位 0 的说明。
25	TRP25	R	0h	传输请求待处理 25。请参阅位 0 的说明。
24	TRP24	R	0h	传输请求待处理 24。请参阅位 0 的说明。
23	TRP23	R	0h	传输请求待处理 23。请参阅位 0 的说明。
22	TRP22	R	0h	传输请求待处理 22。请参阅位 0 的说明。
21	TRP21	R	0h	传输请求待处理 21。请参阅位 0 的说明。
20	TRP20	R	0h	传输请求待处理 20。请参阅位 0 的说明。
19	TRP19	R	0h	传输请求待处理 19。请参阅位 0 的说明。
18	TRP18	R	0h	传输请求待处理 18。请参阅位 0 的说明。
17	TRP17	R	0h	传输请求待处理 17。请参阅位 0 的说明。
16	TRP16	R	0h	传输请求待处理 16。请参阅位 0 的说明。
15	TRP15	R	0h	传输请求待处理 15。请参阅位 0 的说明。
14	TRP14	R	0h	传输请求待处理 14。请参阅位 0 的说明。
13	TRP13	R	0h	传输请求待处理 13。请参阅位 0 的说明。
12	TRP12	R	0h	传输请求待处理 12。请参阅位 0 的说明。
11	TRP11	R	0h	传输请求待处理 11。请参阅位 0 的说明。
10	TRP10	R	0h	传输请求待处理 10。请参阅位 0 的说明。
9	TRP9	R	0h	传输请求待处理 9。请参阅位 0 的说明。
8	TRP8	R	0h	传输请求待处理 8。请参阅位 0 的说明。

表 19-59. MCAN_TXBRP 字段说明 (continued)

位	字段	类型	复位	说明
7	TRP7	R	0h	传输请求待处理 7。请参阅位 0 的说明。
6	TRP6	R	0h	传输请求待处理 6。请参阅位 0 的说明。
5	TRP5	R	0h	传输请求待处理 5。请参阅位 0 的说明。
4	TRP4	R	0h	传输请求待处理 4。请参阅位 0 的说明。
3	TRP3	R	0h	传输请求待处理 3。请参阅位 0 的说明。
2	TRP2	R	0h	传输请求待处理 2。请参阅位 0 的说明。
1	TRP1	R	0h	传输请求待处理 1。请参阅位 0 的说明。
0	TRP0	R	0h	传输请求待处理 0。 每个 Tx 缓冲区都有自己的传输请求待处理位。这些位通过寄存器 TXBAR 进行设置。在一个请求的传输完成后或已通过寄存器 TXBCR 取消后，这些位会复位。 只为那些通过 TXBC 配置的 Tx 缓冲区设置 TXBRP 位。设置 TXBRP 位后，开始进行一次 Tx 扫描，检查具有最高优先级的待处理 Tx 请求（具有最低消息 ID 的 Tx 缓冲区）。 取消请求会复位寄存器 TXBRP 的相应传输请求待处理位。如果在请求取消时已启动传输，则无论传输是否成功，都将在传输结束时执行此操作。复位相应的 TXBRP 位后，将直接复位取消请求位。 发出取消请求后，在以下情况下，将通过 TXBCF 发出完成取消的信号 - 传输成功后（连同相应的 TXBTO 位） - 取消时尚未开始传输 - 传输因仲裁丢失而中止 - 在帧传输期间发生错误 在 DAR 模式下，如果传输不成功，所有传输都会自动取消。对于所有不成功的传输，将设置相应的 TXBCF 位。 0 无传输请求待处理 1 传输请求待处理 注意：在这个特定的 Tx 扫描期间，不考虑 Tx 扫描进行过程中设置的 TXBRP 位。如果请求取消此类 Tx 缓冲区，则立即取消该添加请求，并复位相应的 TXBRP 位。

19.7.1.41 MCAN_TXBAR（偏移 = 70D0h）[复位 = 00000000h]

图 19-67 展示了 MCAN_TXBAR，表 19-60 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区添加请求

图 19-67. MCAN_TXBAR

31	30	29	28	27	26	25	24
AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
23	22	21	20	19	18	17	16
AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
15	14	13	12	11	10	9	8
AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
7	6	5	4	3	2	1	0
AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h

表 19-60. MCAN_TXBAR 字段说明

位	字段	类型	复位	说明
31	AR31	R/WQ	0h	添加请求 31。请参阅位 0 的说明。
30	AR30	R/WQ	0h	添加请求 30。请参阅位 0 的说明。
29	AR29	R/WQ	0h	添加请求 29。请参阅位 0 的说明。
28	AR28	R/WQ	0h	添加请求 28。请参阅位 0 的说明。
27	AR27	R/WQ	0h	添加请求 27。请参阅位 0 的说明。
26	AR26	R/WQ	0h	添加请求 26。请参阅位 0 的说明。
25	AR25	R/WQ	0h	添加请求 25。请参阅位 0 的说明。
24	AR24	R/WQ	0h	添加请求 24。请参阅位 0 的说明。
23	AR23	R/WQ	0h	添加请求 23。请参阅位 0 的说明。
22	AR22	R/WQ	0h	添加请求 22。请参阅位 0 的说明。
21	AR21	R/WQ	0h	添加请求 21。请参阅位 0 的说明。
20	AR20	R/WQ	0h	添加请求 20。请参阅位 0 的说明。
19	AR19	R/WQ	0h	添加请求 19。请参阅位 0 的说明。
18	AR18	R/WQ	0h	添加请求 18。请参阅位 0 的说明。
17	AR17	R/WQ	0h	添加请求 17。请参阅位 0 的说明。
16	AR16	R/WQ	0h	添加请求 16。请参阅位 0 的说明。
15	AR15	R/WQ	0h	添加请求 15。请参阅位 0 的说明。
14	AR14	R/WQ	0h	添加请求 14。请参阅位 0 的说明。
13	AR13	R/WQ	0h	添加请求 13。请参阅位 0 的说明。
12	AR12	R/WQ	0h	添加请求 12。请参阅位 0 的说明。
11	AR11	R/WQ	0h	添加请求 11。请参阅位 0 的说明。
10	AR10	R/WQ	0h	添加请求 10。请参阅位 0 的说明。
9	AR9	R/WQ	0h	添加请求 9。请参阅位 0 的说明。
8	AR8	R/WQ	0h	添加请求 8。请参阅位 0 的说明。

表 19-60. MCAN_TXBAR 字段说明 (continued)

位	字段	类型	复位	说明
7	AR7	R/WQ	0h	添加请求 7。请参阅位 0 的说明。
6	AR6	R/WQ	0h	添加请求 6。请参阅位 0 的说明。
5	AR5	R/WQ	0h	添加请求 5。请参阅位 0 的说明。
4	AR4	R/WQ	0h	添加请求 4。请参阅位 0 的说明。
3	AR3	R/WQ	0h	添加请求 3。请参阅位 0 的说明。
2	AR2	R/WQ	0h	添加请求 2。请参阅位 0 的说明。
1	AR1	R/WQ	0h	添加请求 1。请参阅位 0 的说明。
0	AR0	R/WQ	0h	添加请求 0。 每个 Tx 缓冲区都有自己的添加请求位。写入“1”将设置相应的添加请求位；写入“0”则没有影响。这使得主机能够通过一次写入 TXBAR 为多个 Tx 缓冲区设置传输请求。只为那些通过 TXBC 配置的 Tx 缓冲区设置 TXBAR 位。当没有运行 Tx 扫描时，这些位会立即复位，否则这些位会保持设置状态，直到 Tx 扫描过程完成。 0 未添加传输请求 1 已添加请求传输 注意：如果对具有待处理传输请求（已设置相应的 TXBRP 位）的 Tx 缓冲区应用了一个添加请求，则忽略该添加请求。 仅当 CCCR.CCE =“0”时，才能进行限定写入。

19.7.1.42 MCAN_TXBCR（偏移 = 70D4h）[复位 = 00000000h]

图 19-68 展示了 MCAN_TXBCR，表 19-61 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区取消请求

图 19-68. MCAN_TXBCR

31	30	29	28	27	26	25	24
CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
23	22	21	20	19	18	17	16
CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
15	14	13	12	11	10	9	8
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h
7	6	5	4	3	2	1	0
CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h	R/WQ-0h

表 19-61. MCAN_TXBCR 字段说明

位	字段	类型	复位	说明
31	CR31	R/WQ	0h	取消请求 31。请参阅位 0 的说明。
30	CR30	R/WQ	0h	取消请求 30。请参阅位 0 的说明。
29	CR29	R/WQ	0h	取消请求 29。请参阅位 0 的说明。
28	CR28	R/WQ	0h	取消请求 28。请参阅位 0 的说明。
27	CR27	R/WQ	0h	取消请求 27。请参阅位 0 的说明。
26	CR26	R/WQ	0h	取消请求 26。请参阅位 0 的说明。
25	CR25	R/WQ	0h	取消请求 25。请参阅位 0 的说明。
24	CR24	R/WQ	0h	取消请求 24。请参阅位 0 的说明。
23	CR23	R/WQ	0h	取消请求 23。请参阅位 0 的说明。
22	CR22	R/WQ	0h	取消请求 22。请参阅位 0 的说明。
21	CR21	R/WQ	0h	取消请求 21。请参阅位 0 的说明。
20	CR20	R/WQ	0h	取消请求 20。请参阅位 0 的说明。
19	CR19	R/WQ	0h	取消请求 19。请参阅位 0 的说明。
18	CR18	R/WQ	0h	取消请求 18。请参阅位 0 的说明。
17	CR17	R/WQ	0h	取消请求 17。请参阅位 0 的说明。
16	CR16	R/WQ	0h	取消请求 16。请参阅位 0 的说明。
15	CR15	R/WQ	0h	取消请求 15。请参阅位 0 的说明。
14	CR14	R/WQ	0h	取消请求 14。请参阅位 0 的说明。
13	CR13	R/WQ	0h	取消请求 13。请参阅位 0 的说明。
12	CR12	R/WQ	0h	取消请求 12。请参阅位 0 的说明。
11	CR11	R/WQ	0h	取消请求 11。请参阅位 0 的说明。
10	CR10	R/WQ	0h	取消请求 10。请参阅位 0 的说明。
9	CR9	R/WQ	0h	取消请求 9。请参阅位 0 的说明。
8	CR8	R/WQ	0h	取消请求 8。请参阅位 0 的说明。

表 19-61. MCAN_TXBCR 字段说明 (continued)

位	字段	类型	复位	说明
7	CR7	R/WQ	0h	取消请求 7。请参阅位 0 的说明。
6	CR6	R/WQ	0h	取消请求 6。请参阅位 0 的说明。
5	CR5	R/WQ	0h	取消请求 5。请参阅位 0 的说明。
4	CR4	R/WQ	0h	取消请求 4。请参阅位 0 的说明。
3	CR3	R/WQ	0h	取消请求 3。请参阅位 0 的说明。
2	CR2	R/WQ	0h	取消请求 2。请参阅位 0 的说明。
1	CR1	R/WQ	0h	取消请求 1。请参阅位 0 的说明。
0	CR0	R/WQ	0h	取消请求 0。 每个 Tx 缓冲区都有自己的取消请求位。写入“1”将设置相应的取消请求位；写入“0”则没有影响。这使得主机能够通过一次写入 TXBCR 为多个 Tx 缓冲区设置取消请求。只为那些通过 TXBC 配置的 Tx 缓冲区设置 TXBCR 位。这些位保持设置状态，直到 TXBRP 的相应位复位。 0 无取消待处理 1 取消待处理 仅当 CCCR.CCE =“0”时，才能进行限定写入

19.7.1.43 MCAN_TXBTO（偏移 = 70D8h）[复位 = 00000000h]

图 19-69 展示了 MCAN_TXBTO，表 19-62 中对此进行了介绍。

返回到汇总表。

发生 MCAN Tx 缓冲区传输

图 19-69. MCAN_TXBTO

31	30	29	28	27	26	25	24
TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 19-62. MCAN_TXBTO 字段说明

位	字段	类型	复位	说明
31	TO31	R	0h	发生传输 31。请参阅位 0 的说明。
30	TO30	R	0h	发生传输 30。请参阅位 0 的说明。
29	TO29	R	0h	发生传输 29。请参阅位 0 的说明。
28	TO28	R	0h	发生传输 28。请参阅位 0 的说明。
27	TO27	R	0h	发生传输 27。请参阅位 0 的说明。
26	TO26	R	0h	发生传输 26。请参阅位 0 的说明。
25	TO25	R	0h	发生传输 25。请参阅位 0 的说明。
24	TO24	R	0h	发生传输 24。请参阅位 0 的说明。
23	TO23	R	0h	发生传输 23。请参阅位 0 的说明。
22	TO22	R	0h	发生传输 22。请参阅位 0 的说明。
21	TO21	R	0h	发生传输 21。请参阅位 0 的说明。
20	TO20	R	0h	发生传输 20。请参阅位 0 的说明。
19	TO19	R	0h	发生传输 19。请参阅位 0 的说明。
18	TO18	R	0h	发生传输 18。请参阅位 0 的说明。
17	TO17	R	0h	发生传输 17。请参阅位 0 的说明。
16	TO16	R	0h	发生传输 16。请参阅位 0 的说明。
15	TO15	R	0h	发生传输 15。请参阅位 0 的说明。
14	TO14	R	0h	发生传输 14。请参阅位 0 的说明。
13	TO13	R	0h	发生传输 13。请参阅位 0 的说明。
12	TO12	R	0h	发生传输 12。请参阅位 0 的说明。
11	TO11	R	0h	发生传输 11。请参阅位 0 的说明。
10	TO10	R	0h	发生传输 10。请参阅位 0 的说明。
9	TO9	R	0h	发生传输 9。请参阅位 0 的说明。
8	TO8	R	0h	发生传输 8。请参阅位 0 的说明。

表 19-62. MCAN_TXBTO 字段说明 (continued)

位	字段	类型	复位	说明
7	TO7	R	0h	发生传输 7。请参阅位 0 的说明。
6	TO6	R	0h	发生传输 6。请参阅位 0 的说明。
5	TO5	R	0h	发生传输 5。请参阅位 0 的说明。
4	TO4	R	0h	发生传输 4。请参阅位 0 的说明。
3	TO3	R	0h	发生传输 3。请参阅位 0 的说明。
2	TO2	R	0h	发生传输 2。请参阅位 0 的说明。
1	TO1	R	0h	发生传输 1。请参阅位 0 的说明。
0	TO0	R	0h	发生传输 0。 每个 Tx 缓冲区都有自己的发生传输位。当成功传输后相应的 TXBRP 位清零时，设置这些位。当通过向寄存器 TXBAR 的相应位写入“1”来请求新的传输时，会复位这些位。 0 未发生传输 1 发生传输

19.7.1.44 MCAN_TXBCF（偏移 = 70DCh）[复位 = 00000000h]

图 19-70 展示了 MCAN_TXBCF，表 19-63 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区取消完成

图 19-70. MCAN_TXBCF

31	30	29	28	27	26	25	24
CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
15	14	13	12	11	10	9	8
CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 19-63. MCAN_TXBCF 字段说明

位	字段	类型	复位	说明
31	CF31	R	0h	取消完成 31。请参阅位 0 的说明。
30	CF30	R	0h	取消完成 30。请参阅位 0 的说明。
29	CF29	R	0h	取消完成 29。请参阅位 0 的说明。
28	CF28	R	0h	取消完成 28。请参阅位 0 的说明。
27	CF27	R	0h	取消完成 27。请参阅位 0 的说明。
26	CF26	R	0h	取消完成 26。请参阅位 0 的说明。
25	CF25	R	0h	取消完成 25。请参阅位 0 的说明。
24	CF24	R	0h	取消完成 24。请参阅位 0 的说明。
23	CF23	R	0h	取消完成 23。请参阅位 0 的说明。
22	CF22	R	0h	取消完成 22。请参阅位 0 的说明。
21	CF21	R	0h	取消完成 21。请参阅位 0 的说明。
20	CF20	R	0h	取消完成 20。请参阅位 0 的说明。
19	CF19	R	0h	取消完成 19。请参阅位 0 的说明。
18	CF18	R	0h	取消完成 18。请参阅位 0 的说明。
17	CF17	R	0h	取消完成 17。请参阅位 0 的说明。
16	CF16	R	0h	取消完成 16。请参阅位 0 的说明。
15	CF15	R	0h	取消完成 15。请参阅位 0 的说明。
14	CF14	R	0h	取消完成 14。请参阅位 0 的说明。
13	CF13	R	0h	取消完成 13。请参阅位 0 的说明。
12	CF12	R	0h	取消完成 12。请参阅位 0 的说明。
11	CF11	R	0h	取消完成 11。请参阅位 0 的说明。
10	CF10	R	0h	取消完成 10。请参阅位 0 的说明。
9	CF9	R	0h	取消完成 9。请参阅位 0 的说明。
8	CF8	R	0h	取消完成 8。请参阅位 0 的说明。

表 19-63. MCAN_TXBCF 字段说明 (continued)

位	字段	类型	复位	说明
7	CF7	R	0h	取消完成 7。请参阅位 0 的说明。
6	CF6	R	0h	取消完成 6。请参阅位 0 的说明。
5	CF5	R	0h	取消完成 5。请参阅位 0 的说明。
4	CF4	R	0h	取消完成 4。请参阅位 0 的说明。
3	CF3	R	0h	取消完成 3。请参阅位 0 的说明。
2	CF2	R	0h	取消完成 2。请参阅位 0 的说明。
1	CF1	R	0h	取消完成 1。请参阅位 0 的说明。
0	CF0	R	0h	取消完成 0。 每个 Tx 缓冲区都有自己的取消完成位。当通过 TXBCR 请求取消后，相应的 TXBRP 位清零时，会复位这些位。如果在取消时未设置相应的 TXBRP 位，则立即设置 CF。当通过向寄存器 TXBAR 的相应位写入“1”来请求新的传输时，会复位这些位。 0 无发送缓冲区取消 1 发送缓冲区取消完成

19.7.1.45 MCAN_TXBTIE（偏移 = 70E0h）[复位 = 00000000h]

图 19-71 展示了 MCAN_TXBTIE，表 19-64 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区传输中断使能

图 19-71. MCAN_TXBTIE

31	30	29	28	27	26	25	24
TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 19-64. MCAN_TXBTIE 字段说明

位	字段	类型	复位	说明
31	TIE31	R/W	0h	传输中断使能 31。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
30	TIE30	R/W	0h	传输中断使能 30。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
29	TIE29	R/W	0h	传输中断使能 29。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
28	TIE28	R/W	0h	传输中断使能 28。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
27	TIE27	R/W	0h	传输中断使能 27。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
26	TIE26	R/W	0h	传输中断使能 26。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
25	TIE25	R/W	0h	传输中断使能 25。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
24	TIE24	R/W	0h	传输中断使能 24。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
23	TIE23	R/W	0h	传输中断使能 23。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
22	TIE22	R/W	0h	传输中断使能 22。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用

表 19-64. MCAN_TXBTIE 字段说明 (continued)

位	字段	类型	复位	说明
21	TIE21	R/W	0h	传输中断使能 21。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
20	TIE20	R/W	0h	传输中断使能 20。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
19	TIE19	R/W	0h	传输中断使能 19。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
18	TIE18	R/W	0h	传输中断使能 18。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
17	TIE17	R/W	0h	传输中断使能 17。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
16	TIE16	R/W	0h	传输中断使能 16。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
15	TIE15	R/W	0h	传输中断使能 15。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
14	TIE14	R/W	0h	传输中断使能 14。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
13	TIE13	R/W	0h	传输中断使能 13。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
12	TIE12	R/W	0h	传输中断使能 12。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
11	TIE11	R/W	0h	传输中断使能 11。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
10	TIE10	R/W	0h	传输中断使能 10。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
9	TIE9	R/W	0h	传输中断使能 9。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
8	TIE8	R/W	0h	传输中断使能 8。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
7	TIE7	R/W	0h	传输中断使能 7。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
6	TIE6	R/W	0h	传输中断使能 6。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
5	TIE5	R/W	0h	传输中断使能 5。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用

表 19-64. MCAN_TXBTIE 字段说明 (continued)

位	字段	类型	复位	说明
4	TIE4	R/W	0h	传输中断使能 4。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
3	TIE3	R/W	0h	传输中断使能 3。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
2	TIE2	R/W	0h	传输中断使能 2。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
1	TIE1	R/W	0h	传输中断使能 1。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用
0	TIE0	R/W	0h	传输中断使能 0。每个 Tx 缓冲区都有自己的传输中断使能位。 0 传输中断禁用 1 传输中断启用

19.7.1.46 MCAN_TXBCIE（偏移 = 70E4h）[复位 = 00000000h]

图 19-72 展示了 MCAN_TXBCIE，表 19-65 中对此进行了介绍。

返回到汇总表。

MCAN Tx 缓冲区取消完成中断使能

图 19-72. MCAN_TXBCIE

31	30	29	28	27	26	25	24
CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 19-65. MCAN_TXBCIE 字段说明

位	字段	类型	复位	说明
31	CFIE31	R/W	0h	取消完成中断使能 31。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
30	CFIE30	R/W	0h	取消完成中断使能 30。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
29	CFIE29	R/W	0h	取消完成中断使能 29。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
28	CFIE28	R/W	0h	取消完成中断使能 28。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
27	CFIE27	R/W	0h	取消完成中断使能 27。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
26	CFIE26	R/W	0h	取消完成中断使能 26。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
25	CFIE25	R/W	0h	取消完成中断使能 25。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用

表 19-65. MCAN_TXBCIE 字段说明 (continued)

位	字段	类型	复位	说明
24	CFIE24	R/W	0h	取消完成中断使能 24。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
23	CFIE23	R/W	0h	取消完成中断使能 23。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
22	CFIE22	R/W	0h	取消完成中断使能 22。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
21	CFIE21	R/W	0h	取消完成中断使能 21。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
20	CFIE20	R/W	0h	取消完成中断使能 20。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
19	CFIE19	R/W	0h	取消完成中断使能 19。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
18	CFIE18	R/W	0h	取消完成中断使能 18。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
17	CFIE17	R/W	0h	取消完成中断使能 17。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
16	CFIE16	R/W	0h	取消完成中断使能 16。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
15	CFIE15	R/W	0h	取消完成中断使能 15。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
14	CFIE14	R/W	0h	取消完成中断使能 14。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
13	CFIE13	R/W	0h	取消完成中断使能 13。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
12	CFIE12	R/W	0h	取消完成中断使能 12。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
11	CFIE11	R/W	0h	取消完成中断使能 11。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用

表 19-65. MCAN_TXBCIE 字段说明 (continued)

位	字段	类型	复位	说明
10	CFIE10	R/W	0h	取消完成中断使能 10。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
9	CFIE9	R/W	0h	取消完成中断使能 9。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
8	CFIE8	R/W	0h	取消完成中断使能 8。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
7	CFIE7	R/W	0h	取消完成中断使能 7。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
6	CFIE6	R/W	0h	取消完成中断使能 6。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
5	CFIE5	R/W	0h	取消完成中断使能 5。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
4	CFIE4	R/W	0h	取消完成中断使能 4。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
3	CFIE3	R/W	0h	取消完成中断使能 3。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
2	CFIE2	R/W	0h	取消完成中断使能 2。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
1	CFIE1	R/W	0h	取消完成中断使能 1。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用
0	CFIE0	R/W	0h	取消完成中断使能 0。每个 Tx 缓冲区都有自己的消除完成中断使能位。 0 取消完成中断禁用 1 取消完成中断启用

19.7.1.47 MCAN_TXEFC（偏移 = 70F0h）[复位 = 00000000h]

图 19-73 展示了 MCAN_TXEFC，表 19-66 中对此进行了介绍。

返回到汇总表。

MCAN Tx 事件 FIFO 配置

图 19-73. MCAN_TXEFC

31	30	29	28	27	26	25	24
RESERVED		EFWM					
R/W-0h		R/WQ-0h					
23	22	21	20	19	18	17	16
RESERVED		EFS					
R/W-0h		R/WQ-0h					
15	14	13	12	11	10	9	8
EFSA							
R/WQ-0h							
7	6	5	4	3	2	1	0
EFSA						RESERVED	
R/WQ-0h						R/W-0h	

表 19-66. MCAN_TXEFC 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29-24	EFWM	R/WQ	0h	事件 FIFO 水线 0 禁用水线中断 1-32 Tx 事件 FIFO 水线中断 (IR.TEFW) 级别 >32 禁用水线中断
23-22	保留	R/W	0h	
21-16	EFS	R/WQ	0h	事件 FIFO 大小。Tx 事件 FIFO 元素的索引范围为 0 至 EFS - 1。 0 禁用 Tx 事件 FIFO 1-32 Tx 事件 FIFO 元素数量 >32 大于 32 的值视为 32
15-2	EFSA	R/WQ	0h	事件 FIFO 起始地址。消息 RAM 中 Tx 事件 FIFO 的起始地址（32 位字地址）。
1-0	保留	R/W	0h	

19.7.1.48 MCAN_TXEFS（偏移 = 70F4h）[复位 = 00000000h]

图 19-74 展示了 MCAN_TXEFS，表 19-67 中对此进行了介绍。

返回到汇总表。

MCAN Tx 事件 FIFO 状态

图 19-74. MCAN_TXEFS

31	30	29	28	27	26	25	24
RESERVED						TEFL	EFF
R-0h						R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED			EFPI				
R-0h			R-0h				
15	14	13	12	11	10	9	8
保留			EFGI				
R-0h			R-0h				
7	6	5	4	3	2	1	0
RESERVED		EFFL					
R-0h		R-0h					

表 19-67. MCAN_TXEFS 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R	0h	
25	TEFL	R	0h	Tx 事件 FIFO 元素丢失。该位是中断标志 IR.TEFL 的副本。当 IR.TEFL 复位时，该位也会复位。 0 Tx 事件 FIFO 元素未丢失 1 Tx 事件 FIFO 元素丢失，也会在尝试写入大小为零的 Tx 事件 FIFO 后设置。
24	EFF	R	0h	事件 FIFO 已满 0 Tx 事件 FIFO 未满 1 Tx 事件 FIFO 已满
23-21	RESERVED	R	0h	
20-16	EFPI	R	0h	事件 FIFO Put 索引。Tx 事件 FIFO 写索引指针，范围为 0 至 31。
15-13	保留	R	0h	
12-8	EFGI	R	0h	事件 FIFO Get 索引。Tx 事件 FIFO 读索引指针，范围为 0 至 31。
7-6	RESERVED	R	0h	
5-0	EFFL	R	0h	事件 FIFO 填充级别。存储在 Tx 事件 FIFO 中的元素数量，范围为 0 至 32。

19.7.1.49 MCAN_TXEFA（偏移 = 70F8h）[复位 = 00000000h]

图 19-75 展示了 MCAN_TXEFA，表 19-68 中对此进行了介绍。

返回到汇总表。

MCAN Tx 事件 FIFO 确认

图 19-75. MCAN_TXEFA

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

表 19-68. MCAN_TXEFA 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	0h	
4-0	EFAI	R/W	0h	事件 FIFO 确认索引。主机从 Tx 事件 FIFO 中读取一个元素或一系列元素后，必须将从 Tx 事件 FIFO 中读取的最后一个元素的索引写入 EFAI。这会将 Tx 事件 FIFO Get 索引 TXEFS.EFGI 设置为 EFAI + 1，并更新事件 FIFO 填充级别 TXEFS.EFFL。

19.7.1.50 MCANSS_PID（偏移 = 7200h）[复位 = 68E04901h]

图 19-76 展示了 MCANSS_PID，表 19-69 中对此进行了介绍。

返回到汇总表。

MCAN 子系统修订版本寄存器

图 19-76. MCANSS_PID

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCHEME				MODULE_ID											
R-1h				R-8E0h											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					MAJOR					MINOR					
					R-1h					R-1h					

表 19-69. MCANSS_PID 字段说明

位	字段	类型	复位	说明
31-30	SCHEME	R	1h	PID 寄存器方案
27-16	MODULE_ID	R	8E0h	模块识别号
10-8	MAJOR	R	1h	MCAN 子系统的主要修订版本
5-0	MINOR	R	1h	MCAN 子系统的次要修订版本

19.7.1.51 MCANSS_CTRL（偏移 = 7204h）[复位 = 00000008h]

图 19-77 展示了 MCANSS_CTRL，表 19-70 中对此进行了介绍。

返回到汇总表。

MCAN 子系统控制寄存器

图 19-77. MCANSS_CTRL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED	EXT_TS_CNTR_EN	AUTOWAKEUP	WAKEUPREQEN	DBGSUSP_FREE	RESERVED		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-0h		

表 19-70. MCANSS_CTRL 字段说明

位	字段	类型	复位	说明
31-7	保留	R/W	0h	
6	EXT_TS_CNTR_EN	R/W	0h	外部时间戳计数器使能。 0 禁用外部时间戳计数器 1 启用外部时间戳计数器
5	AUTOWAKEUP	R/W	0h	自动唤醒使能。使 MCANSS 能够根据启用的唤醒请求自动将 MCAN CCCR.INIT 位清零，从而完全唤醒 MCAN。 0 禁用自动写入 CCCR.INIT 1 启用自动写入 CCCR.INIT
4	WAKEUPREQEN	R/W	0h	唤醒请求使能。使 MCANSS 能够在 CAN RXD 活动时唤醒。 0 禁用唤醒请求 1 启用唤醒请求
3	DBGSUSP_FREE	R/W	1h	调试挂起自由位。启用调试挂起。 0 禁用调试挂起 1 启用调试挂起
2-0	RESERVED	R/W	0h	

19.7.1.52 MCANSS_STAT（偏移 = 7208h）[复位 = X]

图 19-78 展示了 MCANSS_STAT，表 19-71 中对此进行了介绍。

返回到汇总表。

MCAN 子系统状态寄存器

图 19-78. MCANSS_STAT

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					ENABLE_FDOE	MEM_INIT_DONE	复位
R-0h					R-X	R-0h	R-0h

表 19-71. MCANSS_STAT 字段说明

位	字段	类型	复位	说明
31-3	保留	R	0h	
2	ENABLE_FDOE	R	X	启用 FD（灵活数据速率）配置 反映了 mcanss_enable_fdoe 配置端口的值， -h = mcanss_enable_fdoe。
1	MEM_INIT_DONE	R	0h	存储器初始化完成。 0 正在初始化消息 RAM 1 已初始化消息 RAM 以供使用
0	复位	R	0h	软复位状态。 0 未处于复位状态 1 正在复位

19.7.1.53 MCANSS_ICS（偏移 = 720Ch）[复位 = 00000000h]

图 19-79 展示了 MCANSS_ICS，表 19-72 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断清除影子寄存器

图 19-79. MCANSS_ICS

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EXT_TS_CNTR_OVFL
R/W-0h							R-0/W1C-0h

表 19-72. MCANSS_ICS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EXT_TS_CNTR_OVFL	R-0/W1C	0h	外部时间戳计数器溢出中断状态清除。读取时始终返回 0。 0 写入“0”不起作用 1 写入“1”会清除 MCANSS_IRS.EXT_TS_CNTR_OVFL 位

19.7.1.54 MCANSS_IRS（偏移 = 7210h）[复位 = 00000000h]

图 19-80 展示了 MCANSS_IRS，表 19-73 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断原始状态寄存器

图 19-80. MCANSS_IRS

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EXT_TS_CNTR_OVFL
R/W-0h							R/W1S-0h

表 19-73. MCANSS_IRS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EXT_TS_CNTR_OVFL	R/W1S	0h	外部时间戳计数器溢出中断状态。该位由 HW 或 SW 写入“1”进行设置。要清除，请使用 MCANSS_ICS.EXT_TS_CNTR_OVFL 位。 0 外部时间戳计数器没有溢出 1 外部时间戳计数器溢出 当 HW 或 SW 将该位设置为“1”时，MCANSS_EXT_TS_UNSERVICED_INTR_CNTR.EXT_TS_INTR_CNTR 位字段将递增 1。

19.7.1.55 MCANSS_IECS（偏移 = 7214h）[复位 = 00000000h]

图 19-81 展示了 MCANSS_IECS，表 19-74 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断使能清除影子寄存器

图 19-81. MCANSS_IECS

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EXT_TS_CNTR_OVFL
R/W-0h							R-0/W1C-0h

表 19-74. MCANSS_IECS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EXT_TS_CNTR_OVFL	R-0/W1C	0h	外部时间戳计数器溢出中断使能清除。读取时始终返回 0。 0 写入“0”不起作用 1 写入“1”会清除 MCANSS_IES.EXT_TS_CNTR_OVFL 位

19.7.1.56 MCANSS_IE（偏移 = 7218h）[复位 = 00000000h]

图 19-82 展示了 MCANSS_IE，表 19-75 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断使能寄存器

图 19-82. MCANSS_IE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EXT_TS_CNTR_OVFL
R/W-0h							R/W1S-0h

表 19-75. MCANSS_IE 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EXT_TS_CNTR_OVFL	R/W1S	0h	外部时间戳计数器溢出中断使能。写入“0”不起作用。写入“1”会设置 MCANSS_IES.EXT_TS_CNTR_OVFL 位。

19.7.1.57 MCANSS_IES（偏移 = 721Ch）[复位 = 00000000h]

图 19-83 展示了 MCANSS_IES，表 19-76 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断使能状态

图 19-83. MCANSS_IES

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

表 19-76. MCANSS_IES 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	EXT_TS_CNTR_OVFL	R	0h	外部时间戳计数器溢出中断使能状态。要进行设置，请使用 CANSS_IE.EXT_TS_CNTR_OVFL 位。要清除，请使用 MCANSS_IECS.EXT_TS_CNTR_OVFL 位。 0 外部时间戳计数器溢出中断未启用 1 外部时间戳计数器溢出中断已启用

19.7.1.58 MCANSS_EOI (偏移 = 7220h) [复位 = 00000000h]

图 19-84 展示了 MCANSS_EOI，表 19-77 中对此进行了介绍。

返回到汇总表。

MCAN 子系统中断结束

图 19-84. MCANSS_EOI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								EOI							
R/W-0h																								R-0/W1S-0h							

表 19-77. MCANSS_EOI 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	EOI	R-0/W1S	0h	中断结束。写入该寄存器将清除关联的中断。如果未处理的中断计数器 > 1，则会生成另一个中断。 0x00 外部 TS 中断已清除 0x01 MCAN(0) 中断已清除 0x02 MCAN(1) 中断已清除 忽略其他写入。

19.7.1.59 MCANSS_EXT_TS_PRESCALER（偏移 = 7224h）[复位 = 00000000h]

图 19-85 展示了 MCANSS_EXT_TS_PRESCALER，表 19-78 中对此进行了介绍。

返回到汇总表。

MCAN 子系统外部时间戳预分频器 0

图 19-85. MCANSS_EXT_TS_PRESCALER

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								预分频器																							
R/W-0h								R/W-0h																							

表 19-78. MCANSS_EXT_TS_PRESCALER 字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23-0	预分频器	R/W	0h	外部时间戳预分频器重载值。外部时间戳计数速率是主机（系统）时钟速率除以该值，除非该值为 0。该位字段中的零值将与 0x000001 值的作用相同。

19.7.1.60 MCANSS_EXT_TS_UNSERVICED_INTR_CNTR（偏移 = 7228h）[复位 = 00000000h]

图 19-86 展示了 MCANSS_EXT_TS_UNSERVICED_INTR_CNTR，表 19-79 中对此进行了介绍。

返回到汇总表。

MCAN 子系统外部时间戳未处理中断计数器

图 19-86. MCANSS_EXT_TS_UNSERVICED_INTR_CNTR

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

表 19-79. MCANSS_EXT_TS_UNSERVICED_INTR_CNTR 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	
4-0	EXT_TS_INTR_CNTR	R	0h	外部时间戳计数器未处理翻转中断。如果该值大于 1，则 MCANSS_EOI 将“1”写入位 0 将发出另一个中断。 该位字段的状态受 MCANSS_IRS.EXT_TS_CNTR_OVFL 位字段的影响。

19.7.1.61 MCANERR_REV (偏移 = 7400h) [复位 = 66A0EA00h]

图 19-87 展示了 MCANERR_REV，表 19-80 中对此进行了介绍。

返回到汇总表。

MCAN 错误聚合器修订版本寄存器

图 19-87. MCANERR_REV

31	30	29	28	27	26	25	24
SCHEME				MODULE_ID			
R-1h				R-6A0h			
23	22	21	20	19	18	17	16
MODULE_ID							
R-6A0h							
15	14	13	12	11	10	9	8
					REVMAJ		
					R-2h		
7	6	5	4	3	2	1	0
		REVMIN					
		R-0h					

表 19-80. MCANERR_REV 字段说明

位	字段	类型	复位	说明
31-30	SCHEME	R	1h	PID 寄存器方案
27-16	MODULE_ID	R	6A0h	模块识别号
10-8	REVMAJ	R	2h	错误聚合器的主要修订版本
5-0	REVMIN	R	0h	错误聚合器的次要修订版本

19.7.1.62 MCANERR_VECTOR (偏移 = 7408h) [复位 = 00000000h]

图 19-88 展示了 MCANERR_VECTOR，表 19-81 中对此进行了介绍。

返回到汇总表。

每个错误检测和校正 (EDC) 控制器都有一组与其关联的错误寄存器 (偏移 0x10 - 0x3B) 。可通过内部串行总线 (SVBUS) 访问这些寄存器。要通过 ECC 聚合器访问它们，所需的控制器 ID 必须与 RD_SVBUS 触发和 RD_SVBUS_ADDRESS 位字段一起写入 ECC_VECTOR 字段。这将启动串行读取，并通过设置 RD_SVBUS_DONE 位来完成。此时，可通过 CPU 对相应偏移地址的正常读取来读取已寻址寄存器。

图 19-88. MCANERR_VECTOR

31	30	29	28	27	26	25	24
RESERVED							RD_SVBUS_DONE
R/W-0h							R-0h
23	22	21	20	19	18	17	16
RD_SVBUS_ADDRESS							
R/W-0h							
15	14	13	12	11	10	9	8
RD_SVBUS	RESERVED				ECC_VECTOR		
R-0/W1S-0h	R/W-0h				R/W-0h		
7	6	5	4	3	2	1	0
ECC_VECTOR							
R/W-0h							

表 19-81. MCANERR_VECTOR 字段说明

位	字段	类型	复位	说明
31-25	RESERVED	R/W	0h	
24	RD_SVBUS_DONE	R	0h	读取完成标志
23-16	RD_SVBUS_ADDRESS	R/W	0h	读取地址偏移
15	RD_SVBUS	R-0/W1S	0h	读取触发
14-11	保留	R/W	0h	
10-0	ECC_VECTOR	R/W	0h	ECC RAM ID。每个错误检测和校正 (EDC) 控制器都有一组与其关联的错误寄存器 (偏移 0x10 - 0x3B) 。可通过内部串行总线 (SVBUS) 访问这些寄存器。要通过 ECC 聚合器访问它们，所需的控制器 ID 必须与 RD_SVBUS 触发和 RD_SVBUS_ADDRESS 位字段一起写入 ECC_VECTOR 字段。这将启动串行读取，并通过设置 RD_SVBUS_DONE 位来完成。此时，可通过 CPU 对相应偏移地址的正常读取来读取已寻址寄存器。 0x000 已选择消息 RAM ECC 控制器 其他保留 (不使用) 通过 SVBUS (偏移 0x10 - 0x3B) 的后续写入将延迟完成。为避免冲突，请在写入后对该范围内的一个寄存器执行读回。

19.7.1.63 MCANERR_STAT（偏移 = 740Ch）[复位 = 00000002h]

图 19-89 展示了 MCANERR_STAT，表 19-82 中对此进行了介绍。

返回到汇总表。

MCAN 错误其他状态

图 19-89. MCANERR_STAT

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

表 19-82. MCANERR_STAT 字段说明

位	字段	类型	复位	说明
31-11	RESERVED	R	0h	
10-0	NUM_RAMS	R	2h	RAM 数量。由聚合器提供服务的 ECC RAM 数量。

19.7.1.64 MCANERR_WRAP_REV（偏移 = 7410h）[复位 = 66A46A02h]

图 19-90 展示了 MCANERR_WRAP_REV，表 19-83 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-90. MCANERR_WRAP_REV

31	30	29	28	27	26	25	24
SCHEME				MODULE_ID			
R-1h				R-6A4h			
23	22	21	20	19	18	17	16
MODULE_ID							
R-6A4h							
15	14	13	12	11	10	9	8
					REVMAJ		
					R-2h		
7	6	5	4	3	2	1	0
		REVMIN					
		R-2h					

表 19-83. MCANERR_WRAP_REV 字段说明

位	字段	类型	复位	说明
31-30	SCHEME	R	1h	PID 寄存器方案
27-16	MODULE_ID	R	6A4h	模块识别号
10-8	REVMAJ	R	2h	错误聚合器的主要修订版本
5-0	REVMIN	R	2h	错误聚合器的次要修订版本

19.7.1.65 MCANERR_CTRL（偏移 = 7414h）[复位 = 00000187h]

图 19-91 展示了 MCANERR_CTRL，表 19-84 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-91. MCANERR_CTRL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							CHECK_SVBUS_TIMEOUT
R/W-0h							R/W-1h
7	6	5	4	3	2	1	0
RESERVED	ERROR_ONCE	FORCE_N_ROW	FORCE_DED	FORCE_SEC	ENABLE_RMW	ECC_CHECK	ECC_ENABLE
R/W-1h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-1h	R/W-1h	R/W-1h

表 19-84. MCANERR_CTRL 字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	0h	
8	CHECK_SVBUS_TIMEOUT	R/W	1h	启用串行 VBUS 超时机制
7	RESERVED	R/W	1h	
6	ERROR_ONCE	R/W	0h	如果设置了该位，FORCE_SEC/FORCE_DED 将只向指定行注入一次错误。一旦发生写回，FORCE_SEC 位将清零。如果未启用写回，则将在数据校正时读取之后的周期内清除该错误。对于双位错误，将在双位错误之后的周期内将 FORCE_DED 位清零。任何后续读取都不会强制出现错误。
5	FORCE_N_ROW	R/W	0h	无论 MCANERR_ERR_CTRL1.ECC_ROW 设置如何，都会在下次读取 RAM 时启用一位/双位错误。对于直写模式，这适用于写入和读取。
4	FORCE_DED	R/W	0h	强制出现双位错误。如果 ERROR_ONCE 置为有效，则清除错误之后的周期。对于直写模式，这适用于写入和读取。应在设置该位之前配置 MCANERR_ERR_CTRL1 和 MCANERR_ERR_CTRL2。
3	FORCE_SEC	R/W	0h	强制出现一位错误。如果 ERROR_ONCE 置为有效，则在写回时或错误之后的周期内清零。对于直写模式，这适用于写入和读取。应在设置该位之前配置 MCANERR_ERR_CTRL1 和 MCANERR_ERR_CTRL2。
2	ENABLE_RMW	R/W	1h	对部分字写入启用读取-修改-写入
1	ECC_CHECK	R/W	1h	启用 ECC 检查。如果 ECC_ENABLE 和 ECC_CHECK 均为“0”，则完全绕过 ECC。
0	ECC_ENABLE	R/W	1h	启用 ECC 生成

19.7.1.66 MCANERR_ERR_CTRL1（偏移 = 7418h）[复位 = 00000000h]

图 19-92 展示了 MCANERR_ERR_CTRL1，表 19-85 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-92. MCANERR_ERR_CTRL1

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

表 19-85. MCANERR_ERR_CTRL1 字段说明

位	字段	类型	复位	说明
31-0	ECC_ROW	R/W	0h	需要应用 FORCE_SEC 或 FORCE_DED 的行地址。如果设置了 FORCE_N_ROW，则忽略该位。

19.7.1.67 MCANERR_ERR_CTRL2（偏移 = 741Ch）[复位 = 00000000h]

图 19-93 展示了 MCANERR_ERR_CTRL2，表 19-86 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-93. MCANERR_ERR_CTRL2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC_BIT2																ECC_BIT1															
R/W-0h																R/W-0h															

表 19-86. MCANERR_ERR_CTRL2 字段说明

位	字段	类型	复位	说明
31-16	ECC_BIT2	R/W	0h	设置 FORCE_DED 时需要翻转的第二列/数据位
15-0	ECC_BIT1	R/W	0h	设置 FORCE_SEC 或 FORCE_DED 时需要翻转的列/数据位

19.7.1.68 MCANERR_ERR_STAT1（偏移 = 7420h）[复位 = 00000000h]

图 19-94 展示了 MCANERR_ERR_STAT1，表 19-87 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-94. MCANERR_ERR_STAT1

31	30	29	28	27	26	25	24
ECC_BIT1							
R-0h							
23	22	21	20	19	18	17	16
ECC_BIT1							
R-0h							
15	14	13	12	11	10	9	8
CLR_CTRL_REG_ERROR	RESERVED		CLR_ECC_OTHER	CLR_ECC_DED		CLR_ECC_SEC	
R/W1S-0h	R/W-0h		R/W1C-0h	R/WD-0h		R/WD-0h	
7	6	5	4	3	2	1	0
CTRL_REG_ERROR	RESERVED		ECC_OTHER	ECC_DED		ECC_SEC	
R/W1S-0h	R/W-0h		R/W1S-0h	R/WI-0h		R/WI-0h	

表 19-87. MCANERR_ERR_STAT1 字段说明

位	字段	类型	复位	说明
31-16	ECC_BIT1	R	0h	ECC 错误位位置。指示发生 SEC 错误时 RAM 数据中出错的位位置。仅在发生 SEC 错误时有效。 0 位 0 出错 1 位 1 出错 2 位 2 出错 3 位 3 出错 ... 31 位 31 出错 >32 无效
15	CLR_CTRL_REG_ERROR	R/W1S	0h	写入“1”会将 CTRL_REG_ERROR 位清零
14-13	RESERVED	R/W	0h	
12	CLR_ECC_OTHER	R/W1C	0h	写入“1”会将 ECC_OTHER 位清零。
11-10	CLR_ECC_DED	R/WD	0h	将 ECC_DED 清零。向该位字段写入一个非零值时，ECC_DED 位字段将按所提供的值递减。
9-8	CLR_ECC_SEC	R/WD	0h	将 ECC_SEC 清零。向该位字段写入一个非零值时，ECC_SEC 位字段将按所提供的值递减。
7	CTRL_REG_ERROR	R/W1S	0h	控制寄存器错误。控制寄存器中的位字段处于不明确状态。这意味着冗余寄存器检测到了一种并非所有值都相同的状态，并已默认为复位状态。S/W 需要将这些寄存器重写为一种已知状态。写入 1 将设置该中断标志。
6-5	RESERVED	R/W	0h	
4	ECC_OTHER	R/W1S	0h	写回时 SEC 错误状态 0 写回挂起时无 SEC 错误 1 指示在写回仍处于挂起状态时发生了连续的一位错误

表 19-87. MCANERR_ERR_STAT1 字段说明 (continued)

位	字段	类型	复位	说明
3-2	ECC_DED	R/WI	0h	检测到双位错误状态。一个 2 位饱和计数器，指示自上次清除以来发生的 DED 错误数量。 0 未检测到双位错误 1 检测到一个双位错误 2 检测到两个双位错误 3 检测到三个双位错误 向该位字段写入一个非零值时，该位字段将按所提供的值递增。
1-0	ECC_SEC	R/WI	0h	已纠正一位错误状态。一个 2 位饱和计数器，指示自上次清除以来发生的 SEC 错误数量。 0 未检测到一位错误 1 检测到一个一位错误并已纠正 2 检测到两个一位错误并已纠正 3 检测到三个一位错误并已纠正 向该位字段写入一个非零值时，该位字段将按所提供的值递增。

19.7.1.69 MCANERR_ERR_STAT2 (偏移 = 7424h) [复位 = 00000000h]

图 19-95 展示了 MCANERR_ERR_STAT2，表 19-88 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-95. MCANERR_ERR_STAT2

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

表 19-88. MCANERR_ERR_STAT2 字段说明

位	字段	类型	复位	说明
31-0	ECC_ROW	R	0h	指示发生了一位或双位错误的行地址。该值等于地址偏移量/4。

19.7.1.70 MCANERR_ERR_STAT3 (偏移 = 7428h) [复位 = 00000000h]

图 19-96 展示了 MCANERR_ERR_STAT3，表 19-89 中对此进行了介绍。

返回到汇总表。

通过内部串行总线经由 ECC 聚合器访问该寄存器。要进行访问，首先必须在 MCAN ECC 向量寄存器中遵循相应的过程。

图 19-96. MCANERR_ERR_STAT3

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留						CLR_SVBUS_TIMEOUT	RESERVED
R/W-0h						R-0/W1C-0h	R/W-0h
7	6	5	4	3	2	1	0
RESERVED						SVBUS_TIMEOUT	WB_PEND
R/W-0h						R-0/W1S-0h	R-0h

表 19-89. MCANERR_ERR_STAT3 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	
9	CLR_SVBUS_TIMEOUT	R-0/W1C	0h	写入 1 可清除串行 VBUS 超时标志
8-2	RESERVED	R/W	0h	
1	SVBUS_TIMEOUT	R-0/W1S	0h	串行 VBUS 超时标志。写入 1 以进行设置。
0	WB_PEND	R	0h	延迟写回挂起状态 0 无写回挂起 1 一个 ECC 数据校正写回挂起

19.7.1.71 MCANERR_SEC_EOI (偏移 = 743Ch) [复位 = 00000000h]

图 19-97 展示了 MCANERR_SEC_EOI，表 19-90 中对此进行了介绍。

返回到汇总表。

MCAN 单错校正中断结束寄存器

图 19-97. MCANERR_SEC_EOI

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EOI_WR
R/W-0h							R-0/W1S-0h

表 19-90. MCANERR_SEC_EOI 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EOI_WR	R-0/W1S	0h	写入该寄存器表示软件已确认挂起的中断，可将下一个中断发送到主机。 请注意，对 MCANERR_ERR_STAT1.CLR_ECC_SEC 的写入通过 SVBUS 进行并延迟完成。为避免额外的中断，在写入该位字段之前，先读回 MCANERR_ERR_STAT1 寄存器。

19.7.1.72 MCANERR_SEC_STATUS（偏移 = 7440h）[复位 = 00000000h]

图 19-98 展示了 MCANERR_SEC_STATUS，表 19-91 中对此进行了介绍。

返回到汇总表。

MCAN 单错校正中断状态寄存器

图 19-98. MCANERR_SEC_STATUS

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

表 19-91. MCANERR_SEC_STATUS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_PEND	R-0	0h	消息 RAM SEC 中断挂起 0 无 SEC 中断挂起 1 SEC 中断挂起

19.7.1.73 MCANERR_SEC_ENABLE_SET（偏移 = 7480h）[复位 = 00000000h]

图 19-99 展示了 MCANERR_SEC_ENABLE_SET，表 19-92 中对此进行了介绍。

返回到汇总表。

MCAN 单错校正中断使能设置寄存器

图 19-99. MCANERR_SEC_ENABLE_SET

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							MSGMEM_ENABLE_SET
R/W-0h							R/W1S-0h

表 19-92. MCANERR_SEC_ENABLE_SET 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_ENABLE_SET	R/W1S	0h	消息 RAM SEC 中断挂起使能设置。向该位写入 1 将启用消息 RAM SEC 错误中断。写入 0 不起作用。读取时将返回相应使能位的当前值。

19.7.1.74 MCANERR_SEC_ENABLE_CLR（偏移 = 74C0h）[复位 = 00000000h]

图 19-100 展示了 MCANERR_SEC_ENABLE_CLR，表 19-93 中对此进行了介绍。

返回到汇总表。

MCAN 单错校正中断使能清除寄存器

图 19-100. MCANERR_SEC_ENABLE_CLR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							MSGMEM_ENABLE_CLR
R/W-0h							R/W1C-0h

表 19-93. MCANERR_SEC_ENABLE_CLR 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_ENABLE_CLR	R/W1C	0h	消息 RAM SEC 中断挂起使能清除。向该位写入 1 将禁用消息 RAM SEC 错误中断。写入 0 不起作用。读取时将返回相应使能位的当前值。

19.7.1.75 MCANERR_DED_EOI (偏移 = 753Ch) [复位 = 00000000h]

图 19-101 展示了 MCANERR_DED_EOI，表 19-94 中对此进行了介绍。

返回到汇总表。

MCAN 双错检测中断结束寄存器

图 19-101. MCANERR_DED_EOI

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							EOI_WR
R/W-0h							R-0/W1S-0h

表 19-94. MCANERR_DED_EOI 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	EOI_WR	R-0/W1S	0h	写入该寄存器表示软件已确认挂起的中断，可将下一个中断发送到主机。 请注意，对 MCANERR_ERR_STAT1.CLR_ECC_DED 的写入通过 SVBUS 进行并延迟完成。为避免额外的中断，在写入该位字段之前，先读回 MCANERR_ERR_STAT1 寄存器。

19.7.1.76 MCANERR_DED_STATUS（偏移 = 7540h）[复位 = 00000000h]

图 19-102 展示了 MCANERR_DED_STATUS，表 19-95 中对此进行了介绍。

返回到汇总表。

MCAN 双错检测中断状态寄存器

图 19-102. MCANERR_DED_STATUS

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

表 19-95. MCANERR_DED_STATUS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_PEND	R-0	0h	消息 RAM DED 中断挂起 0 无 DED 中断挂起 1 DED 中断挂起

19.7.1.77 MCANERR_DED_ENABLE_SET（偏移 = 7580h）[复位 = 00000000h]

图 19-103 展示了 MCANERR_DED_ENABLE_SET，表 19-96 中对此进行了介绍。

返回到汇总表。

MCAN 双错检测中断使能设置寄存器

图 19-103. MCANERR_DED_ENABLE_SET

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							MSGMEM_ENABLE_SET
R/W-0h							R/W1S-0h

表 19-96. MCANERR_DED_ENABLE_SET 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_ENABLE_SET	R/W1S	0h	消息 RAM DED 中断挂起使能设置。向该位写入 1 将启用消息 RAM DED 错误中断。写入 0 不起作用。读取时将返回相应使能位的当前值。

19.7.1.78 MCANERR_DED_ENABLE_CLR (偏移 = 75C0h) [复位 = 00000000h]

图 19-104 展示了 MCANERR_DED_ENABLE_CLR，表 19-97 中对此进行了介绍。

返回到汇总表。

MCAN 双错检测中断使能清除寄存器

图 19-104. MCANERR_DED_ENABLE_CLR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							MSGMEM_ENABLE_CLR
R/W-0h							R/W1C-0h

表 19-97. MCANERR_DED_ENABLE_CLR 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	MSGMEM_ENABLE_CLR	R/W1C	0h	消息 RAM DED 中断挂起使能清除。向该位写入 1 将禁用消息 RAM DED 错误中断。写入 0 不起作用。读取时将返回相应使能位的当前值。

19.7.1.79 MCANERR_AGGR_ENABLE_SET (偏移 = 7600h) [复位 = 00000000h]

图 19-105 展示了 MCANERR_AGGR_ENABLE_SET，表 19-98 中对此进行了介绍。

返回到汇总表。

MCAN 错误聚合器使能设置寄存器

图 19-105. MCANERR_AGGR_ENABLE_SET

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						ENABLE_TIMEOUT_SET	ENABLE_PARITY_SET
R/W-0h						R/W1S-0h	R/W1S-0h

表 19-98. MCANERR_AGGR_ENABLE_SET 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	ENABLE_TIMEOUT_SET	R/W1S	0h	写入 1 可启用超时错误。读取时将返回相应使能位的当前值。
0	ENABLE_PARITY_SET	R/W1S	0h	写入 1 可启用奇偶校验错误。读取时将返回相应使能位的当前值。

19.7.1.80 MCANERR_AGGR_ENABLE_CLR (偏移 = 7604h) [复位 = 00000000h]

图 19-106 展示了 MCANERR_AGGR_ENABLE_CLR，表 19-99 中对此进行了介绍。

返回到汇总表。

MCAN 错误聚合器使能清除寄存器

图 19-106. MCANERR_AGGR_ENABLE_CLR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						ENABLE_TIMEOUT_CLR	ENABLE_PARITY_CLR
R/W-0h						R/W1C-0h	R/W1C-0h

表 19-99. MCANERR_AGGR_ENABLE_CLR 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	ENABLE_TIMEOUT_CLR	R/W1C	0h	写入 1 可禁用超时错误。读取时将返回相应使能位的当前值。
0	ENABLE_PARITY_CLR	R/W1C	0h	写入 1 可禁用奇偶校验错误。读取时将返回相应使能位的当前值。

19.7.1.81 MCANERR_AGGR_STATUS_SET (偏移 = 7608h) [复位 = 00000000h]

图 19-107 展示了 MCANERR_AGGR_STATUS_SET，表 19-100 中对此进行了介绍。

返回到汇总表。

MCAN 错误聚合器状态设置寄存器

图 19-107. MCANERR_AGGR_STATUS_SET

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				SVBUS_TIMEOUT		AGGR_PARITY_ERR	
R/W-0h				R/WI-0h		R/WI-0h	

表 19-100. MCANERR_AGGR_STATUS_SET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	SVBUS_TIMEOUT	R/WI	0h	聚合器串行 VBUS 超时错误状态 2 位饱和计数器，指示自上次清除以来发生的 SVBUS 超时错误数量。 0 未发生超时错误 1 发生了一个超时错误 2 发生了两个超时错误 3 发生了三个超时错误 向该位字段写入一个非零值时，该位字段将按所提供的值递增。
1-0	AGGR_PARITY_ERR	R/WI	0h	聚合器奇偶校验错误状态 2 位饱和计数器，指示自上次清除以来发生的奇偶校验错误数量。 0 未发生奇偶校验错误 1 发生了一个奇偶校验错误 2 发生了两个奇偶校验错误 3 发生了三个奇偶校验错误 向该位字段写入一个非零值时，该位字段将按所提供的值递增。

19.7.1.82 MCANERR_AGGR_STATUS_CLR（偏移 = 760Ch）[复位 = 00000000h]

图 19-108 展示了 MCANERR_AGGR_STATUS_CLR，表 19-101 中对此进行了介绍。

返回到汇总表。

MCAN 错误聚合器状态清除寄存器

图 19-108. MCANERR_AGGR_STATUS_CLR

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				SVBUS_TIMEOUT		AGGR_PARITY_ERR	
R/W-0h				R/WD-0h		R/WD-0h	

表 19-101. MCANERR_AGGR_STATUS_CLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	SVBUS_TIMEOUT	R/WD	0h	聚合器串行 VBUS 超时错误状态 2 位饱和计数器，指示自上次清除以来发生的 SVBUS 超时错误数量。 0 未发生超时错误 1 发生了一个超时错误 2 发生了两个超时错误 3 发生了三个超时错误 向该位字段写入一个非零值时，该位字段将按所提供的值递减。
1-0	AGGR_PARITY_ERR	R/WD	0h	聚合器奇偶校验错误状态 2 位饱和计数器，指示自上次清除以来发生的奇偶校验错误数量。 0 未发生奇偶校验错误 1 发生了一个奇偶校验错误 2 发生了两个奇偶校验错误 3 发生了三个奇偶校验错误 向该位字段写入一个非零值时，该位字段将按所提供的值递减。

19.7.1.83 IIDX (偏移 = 7820h) [复位 = 00000000h]

图 19-109 展示了 IIDX，表 19-102 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断 (最高优先级) 由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU (不是从调试接口) 读取后，必须使用第二高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 19-109. IIDX

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

表 19-102. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起。 1h = MCAN 中断线 0 中断挂起。 2h = MCAN 中断线 1 中断挂起。 3h = 消息 RAM SEC (单错校正) 中断挂起。 4h = 消息 RAM DED (双错检测) 中断挂起。 5h = 外部时间戳计数器溢出中断挂起。 6h = 时钟停止唤醒中断挂起。

19.7.1.84 IMASK（偏移 = 7828h）[复位 = 00000000h]

图 19-110 展示了 IMASK，表 19-103 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 19-110. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		WAKEUP	EXT_TS_CNTR_OVFL	DED	SEC	INTL1	INTL0
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 19-103. IMASK 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5	WAKEUP	R/W	0h	时钟停止唤醒中断屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	EXT_TS_CNTR_OVFL	R/W	0h	外部时间戳计数器溢出中断屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	DED	R/W	0h	消息 RAM DED 中断屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	SEC	R/W	0h	消息 RAM SEC 中断屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	INTL1	R/W	0h	MCAN 中断线 1 屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	INTL0	R/W	0h	MCAN 中断线 0 屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

19.7.1.85 RIS（偏移 = 7830h）[复位 = 00000000h]

图 19-111 展示了 RIS，表 19-104 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 19-111. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		WAKEUP	EXT_TS_CNTR_OVFL	DED	SEC	INTL1	INTL0
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 19-104. RIS 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	WAKEUP	R	0h	时钟停止唤醒中断。 0h = 未发生中断 1h = 已发生中断
4	EXT_TS_CNTR_OVFL	R	0h	外部时间戳计数器溢出中断。 0h = 未发生中断 1h = 已发生中断
3	DED	R	0h	消息 RAM DED 中断。 0h = 未发生中断 1h = 已发生中断
2	SEC	R	0h	消息 RAM SEC 中断。 0h = 未发生中断 1h = 已发生中断
1	INTL1	R	0h	MCAN 中断线 1。 0h = 未发生中断 1h = 已发生中断
0	INTL0	R	0h	MCAN 中断线 0。 0h = 未发生中断 1h = 已发生中断

19.7.1.86 MIS（偏移 = 7838h）[复位 = 00000000h]

图 19-112 展示了 MIS，表 19-105 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 19-112. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		WAKEUP	EXT_TS_CNTR_OVFL	DED	SEC	INTL1	INTL0
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 19-105. MIS 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	WAKEUP	R	0h	已屏蔽时钟停止唤醒中断。 0h = 未发生中断 1h = 已发生中断
4	EXT_TS_CNTR_OVFL	R	0h	已屏蔽外部时间戳计数器溢出中断。 0h = 未发生中断 1h = 已发生中断
3	DED	R	0h	已屏蔽消息 RAM DED 中断。 0h = 未发生中断 1h = 已发生中断
2	SEC	R	0h	已屏蔽消息 RAM SEC 中断。 0h = 未发生中断 1h = 已发生中断
1	INTL1	R	0h	已屏蔽 MCAN 中断线 1。 0h = 未发生中断 1h = 已发生中断
0	INTL0	R	0h	已屏蔽 MCAN 中断线 0。 0h = 未发生中断 1h = 已发生中断

19.7.1.87 ISET（偏移 = 7840h）[复位 = 00000000h]

图 19-113 展示了 ISET，表 19-106 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 19-113. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		WAKEUP	EXT_TS_CNTR_OVFL	DED	SEC	INTL1	INTL0
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 19-106. ISET 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	WAKEUP	W	0h	设置时钟停止唤醒中断。 0h = 写入 0 无效 1h = 设置中断
4	EXT_TS_CNTR_OVFL	W	0h	设置外部时间戳计数器溢出中断。 0h = 写入 0 无效 1h = 设置中断
3	DED	W	0h	设置消息 RAM DED 中断。 0h = 写入 0 无效 1h = 设置中断
2	SEC	W	0h	设置消息 RAM SEC 中断。 0h = 写入 0 无效 1h = 设置中断
1	INTL1	W	0h	设置 MCAN 中断线 1。 0h = 写入 0 无效 1h = 设置中断
0	INTL0	W	0h	设置 MCAN 中断线 0。 0h = 写入 0 不产生影响 1h = 设置中断

19.7.1.88 ICLR（偏移 = 7848h）[复位 = 00000000h]

图 19-114 展示了 ICLR，表 19-107 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 19-114. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		WAKEUP	EXT_TS_CNTR_OVFL	DED	SEC	INTL1	INTL0
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 19-107. ICLR 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	WAKEUP	W	0h	清除时钟停止唤醒中断。 0h = 写入 0 无效 1h = 清除中断
4	EXT_TS_CNTR_OVFL	W	0h	清除外部时间戳计数器溢出中断。 0h = 写入 0 无效 1h = 清除中断
3	DED	W	0h	清除消息 RAM DED 中断。 0h = 写入 0 无效 1h = 清除中断
2	SEC	W	0h	清除消息 RAM SEC 中断。 0h = 写入 0 无效 1h = 清除中断
1	INTL1	W	0h	清除 MCAN 中断线 1。 0h = 写入 0 无效 1h = 清除中断
0	INTL0	W	0h	清除 MCAN 中断线 0。 0h = 写入 0 不产生影响 1h = 清除中断

19.7.1.89 EVT_MODE（偏移 = 78E0h）[复位 = 00000001h]

图 19-115 展示了 EVT_MODE，表 19-108 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 19-115. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
ESERVED						INT0_CFG	
R/W-						R-1h	

表 19-108. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1-0	INT0_CFG	R	1h	[IPSTANDARD.CPU_INT] 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

19.7.1.90 DESC（偏移 = 78FCh）[复位 = 09400000h]

图 19-116 展示了 DESC，表 19-109 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 19-116. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-940h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				RESERVED				MAJREV				MINREV			
R-0h				R-				R-0h				R-0h			

表 19-109. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	940h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	0h	模块*实例*的功能集 0h = 启用了 CAN-FD 模式的 MCAN 模块 <<内部说明：这是一种 IP 内的纸张旋转型号。这如何映射到 SYS_MCAN_ENABLE_FD 选择值？>> 1h = 禁用了 CAN-FD 模式的 MCAN 模块 <<内部说明：这是一种 IP 内的纸张旋转型号。这如何映射到 SYS_MCAN_ENABLE_FD 选择值？>>
11-8	RESERVED	R	0h	
7-4	MAJREV	R	0h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

19.7.1.91 MCANSS_CLKEN（偏移 = 7900h）[复位 = 00000000h]

图 19-117 展示了 MCANSS_CLKEN，表 19-110 中对此进行了介绍。

返回到汇总表。

MCAN 模块时钟（用于访问 MCAN 模块 MMR 的功能时钟和 Vbusp）使能寄存器
<内部说明> 只要通过纸张旋转配置启用了 IP，该特定于 IP 的 MMR 本身就不具有时钟选通功能

图 19-117. MCANSS_CLKEN

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

表 19-110. MCANSS_CLKEN 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	CLK_REQEN		0h	MCAN 功能时钟和 MCAN/MCANSS MMR 时钟请求使能位 0h = 未请求 MCAN 模块功能时钟和 Vbusp。 这些时钟被选通至 MCAN 模块。 1h = 设置该位会请求 MCAN 模块功能时钟和 Vbusp。 这些时钟不选通至 MCAN 模块。

19.7.1.92 MCANSS_CLKDIV（偏移 = 7904h）[复位 = 00000000h]

图 19-118 展示了 MCANSS_CLKDIV，表 19-111 中对此进行了介绍。

返回到汇总表。

一旦可用，就需要转到管理界面

图 19-118. MCANSS_CLKDIV

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

表 19-111. MCANSS_CLKDIV 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1-0	RATIO	R/W	0h	时钟分频比规范。能够配置时钟分频设置，以便 MCAN 功能时钟输入到 MCAN-SS。 0h (R/W) = 对输入时钟进行 1 分频 1h (R/W) = 对输入时钟进行 2 分频 2h (R/W) = 对输入时钟进行 4 分频 3h (R/W) = 对输入时钟进行 1 分频

19.7.1.93 MCANSS_CLKCTL（偏移 = 7908h）[复位 = 00000000h]

图 19-119 展示了 MCANSS_CLKCTL，表 19-112 中对此进行了介绍。

返回到汇总表。

MCANSS 时钟停止控制 MMR。
<内部说明> 包装器 MMR（包括这个 MMR）的总线时钟不由该寄存器选通。

图 19-119. MCANSS_CLKCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							WKUP_GLTFLT_EN
R/W-0h							R/W-0h
7	6	5	4	3	2	1	0
RESERVED			WAKEUP_INT_EN	RESERVED			STOPREQ
R/W-0h			R/W-0h	R/W-0h			R/W-0h

表 19-112. MCANSS_CLKCTL 字段说明

位	字段	类型	复位	说明
31-9	RESERVED	R/W	0h	
8	WKUP_GLTFLT_EN	R/W	0h	设置该位会启用 MCAN RXD 输入端的干扰滤波器，从而唤醒 MCAN 控制器以退出时钟选通。 0h = 当 MCAN 处于时钟停止模式时，禁用 RXD 输入端的干扰滤波器使能（等待 RXD 输入端的事件以进行时钟停止唤醒）。 1h = 当 MCAN 处于时钟停止模式时，启用 RXD 输入端的干扰滤波器使能（等待 RXD 输入端的事件以进行时钟停止唤醒）。
7-5	RESERVED	R/W	0h	
4	WAKEUP_INT_EN	R/W	0h	该位可控制 MCAN IP 时钟停止唤醒中断的启用或禁用（在启用了 MCANSS_CTRL.WAKEUPREQEN 唤醒请求以在 CAN RXD 活动时唤醒 MCAN IP 的情况下） 0h = 禁用 MCAN IP 时钟停止唤醒中断 1h = 启用 MCAN IP 时钟停止唤醒中断
3-1	保留	R/W	0h	
0	STOPREQ	R/W	0h	该位用于启用/禁用 MCAN 时钟（主机时钟和功能时钟）选通请求。 注意：HW 可在 CAN RX 活动时通过时钟停止唤醒来复位该位。请参阅规范，了解更多详细信息。 0h = 禁用 MCAN-SS 时钟停止请求 1h = 启用 MCAN-SS 时钟停止请求

19.7.1.94 MCANSS_CLKSTS（偏移 = 790Ch）[复位 = 00000000h]

图 19-120 展示了 MCANSS_CLKSTS，表 19-113 中对此进行了介绍。

返回到汇总表。

MCANSS 时钟停止状态寄存器，用于指示时钟停止机制的状态

图 19-120. MCANSS_CLKSTS

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

表 19-113. MCANSS_CLKSTS 字段说明

位	字段	类型	复位	说明
31-9	保留	R	0h	
8	CCLKDONE	R	0h	该位指示来自 GPRCM 的 MCAN 控制器时钟请求的状态。 0h = MCAN 控制器时钟不可用于 MCAN IP。 1h = MCAN 控制器时钟已启用，可用于 MCAN IP。
7-5	RESERVED	R	0h	
4	STOPREQ_HW_OVR	R	0h	MCANSS 时钟停止 HW 覆盖状态位。 当通过 CAN RX 活动触发时钟停止唤醒事件时，该位指示 HW 何时将 MCANSS_CLKCTL.STOPREQ 位清零。 0h = MCANSS_CLKCTL.STOPREQ 位尚未被 HW 清零。 1h = MCANSS_CLKCTL.STOPREQ 位已被 HW 清零。
3-1	RESERVED	R	0h	
0	CLKSTOP_ACKSTS	R	0h	MCAN IP 的时钟停止确认状态 0h = 未确认时钟停止。 1h = MCAN IP 已确认时钟停止；可通过同时停止 CAN 主机时钟和功能时钟对 MCAN-SS 进行时钟选通。

章节 20
CRC

循环冗余校验 (CRC) 加速器根据 CRC16-CCITT 多项式或 CRC32-ISO3309 多项式为给定数据序列生成签名。

20.1 CRC 概述

CRC 加速器为给定的数据序列生成 CRC 签名。支持 CRC16-CCITT 和 CRC32-ISO3309 CRC 函数。当使用一个固定种子值初始化 CRC 时，相同的输入数据序列会产生相同的 CRC 签名。通常，对于给定的 CRC 函数，不同的输入数据序列会产生不同的签名。

CRC 加速器的主要特性包括：

- 支持 CRC16-CCITT 和 CRC32-ISO3309
- 对每个数据输入的新 CRC 输出进行快速单周期计算（无等待状态）
- 支持输入/输出位反转
- 支持小端字节序或大端字节序模式
- 为 CRCIN 输入字节、半字或字
- 512 字 CRCIN_IDX 输入字段，其中所有地址都映射到 CRCIN，支持使用标准 C 风格 memcpy() 例程将数据加载到 CRC 模块中，数据长度高达 2KB

20.1.1 CRC16-CCITT

对于 CRC16-CCITT，CRC 签名是根据 16 位 CCITT 标准中给出的多项式生成的，如下面的等式中所示。

$$f(x)=x^{16}+x^{12}+x^{5}+1 \quad (25)$$

CRC16-CCITT 摘要大小为 16 位（半字）。

20.1.2 CRC32-ISO3309

对于 CRC32-ISO3309，CRC 签名是根据 ISO3309 以太网标准中给出的多项式生成的，如下面的等式所示。

$$f(x)=x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1 \quad (26)$$

CRC32-ISO3309 的摘要大小为 32 位。

20.2 CRC 运行

通过以下方式初始化 CRC 生成器：在 CRCCTRL 寄存器中配置所需的运行模式，然后将种子值写入 CRCSEED 寄存器。在种子加载到 CRCSEED 寄存器后，CRCOUT 寄存器将反映加载到 CRCSEED 的 SEED 值。

备注

如果在种子值被写入 CRCSEED 之前字节序模式被配置为大端字节序，那么当种子值被载入 CRC 模块时，被写入 CRCSEED 的种子值的字节顺序被交换。

初始化后，可以通过使用字节（8 位）、半字（16 位）或字（32 位）写入 CRCIN 寄存器来将数据输入到 CRC 生成器中。CPU 或 DMA 可用于将输入数据移动到 CRC 加速器。

备注

字节写入不需要字对齐；系统将以相同的方式解释对 CRCIN 中任何字节位置的字节写入（将 8 个写入位添加到所计算的 CRC）。半字写入也不需要字对齐、，但必须半字对齐。例如，半字可以写入 CRCIN 的 BIT0-BIT15 或 BIT16-BIT32，但不能写入 BIT8-BIT23。

当使用 CRC 生成器验证数据集时，要包括在 CRC 计算中的所有数据必须按照计算原始 CRC 签名时使用的相同顺序写入 CRCIN 寄存器。当使用 CRC 生成器创建一个新的签名以供将来验证时，请确保在执行验证时以相同的方式和相同的设置加载数据。

可以通过读取 CRCOUT 寄存器随时读取当前 CRC 输出。

20.2.1 CRC 生成器实现

CRC 生成器是用一组 XOR 树实现的。通过写入 CRCIN 寄存器向 CRC 加速器提供一组 8、16 或 32 位后，将对整组输入位进行计算。当新数据写入 CRCIN 时，CRC 生成器在单个周期内更新 CRC 输出。无需总线等待状态即可将数据背对背加载到 CRC 生成器中。

20.2.2 配置

CRC 加速器支持多项式选择、位反转选择和字节顺序（字节序）选择。本节介绍了这些配置方面。

在通过 PWREN 寄存器使用之前必须启用 CRC 加速器（请参阅外设电源使能）。

CRC 加速器处于电源域 1 (PD1) 中，因此只能在 RUN 或 SLEEP 模式下激活。如果 CRC 加速器被配置为由应用软件启用，并且器件进入 STOP 或 STANDBY 模式，SYSCTL 将强制 CRC 进入禁用状态，直到器件存在 STOP 或 STANDBY 模式。当 CRC 在 STOP 或 STANDBY 模式下强制进入禁用状态时，所有 CRC 寄存器内容都被保留。

CRC 模块只能从 PD1 总线时钟 (MCLK) 运行。

20.2.2.1 多项式选择

使用 CRCCTRL 寄存器中的 POLYSIZE 位来选择所需的多项式（CRC16-CCITT 或 CRC32-ISO3309）。必须在加载种子值和任何数据值之前选择要使用的多项式。

当 CRC 发生器配置为 16 位摘要 (CRC16-CCITT) 时，满足以下条件：

- CRCSEED 寄存器的上半字（16 位）被忽略，仅使用下半字
- CRCOUT 寄存器的上半字（16 位）回读为零，仅下半字有效

当 CRC 发生器配置为 CRC32-ISO3309 时，CRCSEED 和 CRCOUT 的所有 32 位都有效。

20.2.2.2 位顺序

在主框架计算机时代定义了各种 CRC 标准。当时，BIT0 被视为 MSB。在现代计算中，BIT0 通常是 LSB。

Arm Cortex-M0+ CPU 将 BIT0 视为 LSB，这在现代 CPU 和 MCU 中很常见。这有时会引起混淆，因为在某些情况下 BIT0 被视为 LSB，在其他情况下被视为 MSB。因此，CRC 加速器提供了一个位顺序反转功能来支持这两种惯例。

通过设置 CRCCTRL 寄存器的 BITREVERSE 位可以启用位顺序反转，从而为输入和输出数据提供以下行为：

- **输入数据：**在传递给用于 CRC 计算的 CRC 发生器之前，写入 CRCIN 寄存器的每个输入字节的位顺序相反
- **输出数据：**从 CRCOUT 寄存器读取时，16 位或 32 位 CRC 结果的位顺序相反

备注

如果输入数据必须以位反转的方式提供，但输出必须以非反转的方式读取，则可以在将数据加载到 CRCIN 之前设置 BITREVERSE 位，然后在所有数据写入 CRCIN 之后、但在从 CRCOUT 读取生成的签名之前清零。同样，可以在 BITREVERSE 位清零的情况下将输入数据加载到 CRCIN（非反转），并在读取输出之前翻转输出（通过在读取 CRCOUT 之前设置 BITREVERSE）。

20.2.2.3 字节交换

寄存器 CRCCTRL 中的 OUTPUT_BYTESWAP 位可用于启用或禁用 CRC 输出字节交换。该位会控制在读取 CRCOUT 寄存器时输出是否按字节交换。如果 CRCOUT 作为半字进行访问，并且 OUTPUT_BYTESWAP 设置为 1，则交换并返回 16 位访问中的两个字节。使用 B0、B1、B2 和 B3 识别字节 0、字节 1、字节 2、字节 3。B1 作为 B0 返回，B0 作为 B1 返回。如果将 CRCOUT 作为一个字进行访问，并且 OUTPUT_BYTESWAP 设置为 1，则交换 32 位读取中的四个字节。B3 作为 B0 返回，B2 作为 B1 返回，B1 作为 B2 返回，B0 作为 B3 返回。

请注意，如果 CRC POLYSIZE 为 16 位且在启用 OUTPUT_BYTESWAP 的情况下执行 32 位 CRCOUT 读取，则输出为：MSB LSB 0x0 0x0 B0 B1。如果 CRC POLYSIZE 为 16 位且在禁用 OUTPUT_BYTESWAP 的情况下执行 32 位 CRCOUT 读取，则输出为：MSB LSB 0x0 0x0 B1 B0。

20.2.2.4 字节顺序

处理半字或字输入数据时，输入字节顺序可配置为小端字节序或大端字节序。默认配置为小端字节序。要在使用半字或字输入时反转字节顺序，请设置 CRCCTRL 寄存器中的 INPUT_ENDIANNESS 位。

反转字节序将导致半字和字写入的以下转换：

表 20-1. CRCIN 字节顺序转换

字节序	写入 CRCIN 的数据	应用于 CRC 逻辑的数据
0（小端）	0xAABB	0xAABB
1（大端）	0xAABB	0xBBAA
0（小端）	0xAABBCCDD	0xAABBCCDD
1（大端）	0xAABBCCDD	0xDDCCBBAA

备注

如果在种子值写入 CRCSEED 之前设置 ENDIANNESS 位，则写入 CRCSEED 的种子值的字节顺序在加载到 CRC 中时也会反转，并且在写入 CRCSEED 寄存器后读取 CRCOUT 寄存器时，将反向读取种子值。

20.2.2.5 CRC C 库兼容性

为了简化由软件将数据加载到 CRC 的过程，CRC 加速器提供了一个 512 字 (2KB) 的 CRCIN_IDX 区域。在 CRCIN_IDX 区域内，对任何字的写入被重新映射为对 CRCIN 寄存器的写入，在功能上相当于对 CRCIN 寄存器的写入。这种机制允许使用标准 C 库 *memcpy()* 例程将数据从 SRAM 或闪存复制到 CRC 中，前提是源数据小于 2KB（2,048 字节）。

20.3 CRC 寄存器

表 20-2 列出了 CRC 寄存器的存储器映射寄存器。表 20-2 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 20-2. CRC 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1004h	CLKSEL	时钟选择		转到
10FCh	DESC	模块说明		转到
1100h	CRCCTRL	CRC 控制寄存器		转到
1104h	CRCSEED	CRC 种子寄存器		转到
1108h	CRCIN	CRC 输入数据寄存器		转到
110Ch	CRCOUT	CRC 输出结果寄存器		转到
1800h + 公式	CRCIN_IDX[y]	CRC 输入数据数组寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 20-3 显示了适用于此部分中访问类型的代码。

表 20-3. CRC 访问类型代码

访问类型	代码	说明
读取类型		
H	高电平	由硬件置位或清除
R	R	读取
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值
寄存器数组变量		
i、j、k、l、m、n		当这些变量用于寄存器名称、偏移或地址时，它们指的是寄存器数组的值，其中寄存器是一组重复寄存器的一部分。寄存器组构成分层结构，数组用公式表示。
y		当该变量用于寄存器名称、偏移或地址时，它指的是寄存器数组的值。

20.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 20-1 展示了 PWREN，表 20-4 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 20-1. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 20-4. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

20.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 20-2 展示了 RSTCTL，表 20-5 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 20-2. RSTCTL

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						RESETSTKYCLR	RESETASSERT
R-0h						WK-0h	WK-0h

表 20-5. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	R	0h	保留
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

20.3.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 20-3 展示了 STAT，表 20-6 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 20-3. STAT

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

表 20-6. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	保留
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	保留

20.3.4 CLKSEL（偏移 = 1004h）[复位 = 00000001h]

图 20-4 展示了 CLKSEL，表 20-7 中对此进行了介绍。

返回到汇总表。

时钟源选择

图 20-4. CLKSEL

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

表 20-7. CLKSEL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	MCLK_SEL	R	1h	如果启用，选择主时钟 (MCLK) 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源

20.3.5 DESC（偏移 = 10FCh）[复位 = 20117010h]

图 20-5 展示了 DESC，表 20-8 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 20-5. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-2011h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-7h				R-0h				R-1h				R-0h			

表 20-8. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	2011h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	7h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数 0h = 最小值 Fh = 尽可能高的值
7-4	MAJREV	R	1h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

20.3.6 CRCCTRL（偏移 = 1100h）[复位 = X]

图 20-6 展示了 CRCCTRL，表 20-9 中对此进行了介绍。

返回到汇总表。

CRC 控制寄存器。CRC 的配置控制。

图 20-6. CRCCTRL

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			OUTPUT_BYTESWAP	RESERVED	INPUT_ENDIANNESS	BITREVERSE	POLYSIZE
R-0h			R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h

表 20-9. CRCCTRL 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	保留
4	OUTPUT_BYTESWAP	R/W	0h	CRC 输出字节交换使能。该位会控制在读取 CRCOUT 寄存器时输出是否按字节交换。 如果 CRCOUT 作为半字进行访问，并且 OUTPUT_BYTESWAP 设置为 1，则交换并返回 16 位访问中的两个字节。 B1 作为 B0 返回 B0 作为 B1 返回 如果将 CRCOUT 作为一个字进行访问，并且 OUTPUT_BYTESWAP 设置为 1，则交换 32 位读取中的四个字节。 B3 作为 B0 返回 B2 作为 B1 返回 B1 作为 B2 返回 B0 作为 B3 返回 请注意，如果 CRC POLYSIZE 为 16 位且在启用 OUTPUT_BYTESWAP 的情况下执行 32 位 CRCOUT 读取， 则输出为： MSB LSB 0x0 0x0 B0 B1 如果 CRC POLYSIZE 为 16 位且在禁用 OUTPUT_BYTESWAP 的情况下执行 32 位 CRCOUT 读取， 则输出为： MSB LSB 0x0 0x0 B1 B0 0h = 输出字节交换被禁用 1h = 输出字节交换被启用。
3	RESERVED	R	0h	保留
2	INPUT_ENDIANNESS	R/W	0h	CRC 字节序。该位指示输入数据的一个字或半字内的字节顺序。 0h = LSB 是最低存储器地址，首先被处理。 1h = LSB 是最高存储器地址，最后被处理。

表 20-9. CRCCTRL 字段说明 (continued)

位	字段	类型	复位	说明
1	BITREVERSE	R/W	0h	CRC 位输入和输出反转。该位指示用于 CRC 计算的每个输入字节的位顺序在传递给生成器之前被反转，并且计算的 CRC 的位顺序在从 CRC_RESULT 读取时被反转。 0h = 位顺序不反转。 1h = 位顺序反转。
0	POLYSIZE	R/W	0h	该位指示生成器执行哪个 CRC 计算。 0h = 执行 CRC-32 ISO-3309 计算 1h = 执行 CRC-16 CCITT

20.3.7 CRCSEED (偏移 = 1104h) [复位 = 00000000h]

图 20-7 展示了 CRCSEED，表 20-10 中对此进行了介绍。

返回到汇总表。

CRC 种子寄存器。写入该寄存器的数据用于使用此 SEED 值初始化 CRC 结果。请注意，在 16 位模式下，仅使用该值的低 16 位。写入该寄存器后，CRC 输出结果寄存器将反映该值。

图 20-7. CRCSEED

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

表 20-10. CRCSEED 字段说明

位	字段	类型	复位	说明
31-0	SEED	W	0h	种子数据 00000000h = 最小值 FFFFFFFFh = 最大值

20.3.8 CRCIN（偏移 = 1108h）[复位 = 00000000h]

图 20-8 展示了 CRCIN，表 20-11 中对此进行了介绍。

返回到汇总表。

CRC 输入数据寄存器。写入该寄存器的数据与当前 CRC 结果一起用于计算下一个 CRC 结果。这是在单个时钟周期内完成的，不需要等待状态。该寄存器能够以字节、半字或字传输进行写入，正确的位数将用于下一个 CRC 结果。
该寄存器还映射到从 0xTDB_X000 开始到 0xTDB_XFFF 结束的一系列寄存器，以允许使用 memcpy 而不是 DMA 进行不超出该寄存器存储器范围边界的 CRC 计算。

图 20-8. CRCIN

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

表 20-11. CRCIN 字段说明

位	字段	类型	复位	说明
31-0	数据	W	0h	输入数据 00000000h = 最小值 FFFFFFFFh = 最大值

20.3.9 CRCOUT（偏移 = 110Ch）[复位 = 00000000h]

图 20-9 展示了 CRCOUT，表 20-12 中对此进行了介绍。

返回到汇总表。

CRC 输出结果寄存器。该寄存器存储当前 CRC 计算的结果。请注意，当配置为 16 位模式时，高位将读回 0。请注意，如果在 CRC 控制寄存器中设置了输出反转，它将被应用。

图 20-9. CRCOUT

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

表 20-12. CRCOUT 字段说明

位	字段	类型	复位	说明
31-0	结果	R	0h	结果 00000000h = 最小值 FFFFFFFFh = 最大值

20.3.10 CRCIN_IDX[y]（偏移 = 1800h + 公式）[复位 = 00000000h]

图 20-10 展示了 CRCIN_IDX[y]，表 20-13 中对此进行了介绍。

返回到汇总表。

该寄存器被双映射到 CRCIN，旨在允许使用 C memcpy 例程运行。

偏移 = 1800h + (y * 4h)；其中 y = 0h 至 1FFh

图 20-10. CRCIN_IDX[y]

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

表 20-13. CRCIN_IDX[y] 字段说明

位	字段	类型	复位	说明
31-0	数据	HW	0h	输入数据 00000000h = 最小值 FFFFFFFFh = 最大值

章节 21
AES

AES 加速器模块会加速硬件中基于 FIPS PUB 197 高级加密标准 (AES) 的加密和解密操作。

21.1 AES 概述

AES 加速器模块根据高级加密标准 (AES) 在硬件中使用 128 位或 256 位密钥对 128 位数据块进行加密和解密。AES 是 FIPS PUB 197 中指定的对称密钥块加密算法。

AES 加速器的特性包括：

- AES 128 位块加密和解密
- DMA 触发器支持自动执行 NIST SP 800-38 中定义的 ECB、CBC、OFB 和 CFB 块加密模式
- 通过加密预先计算的（nonce || 计数器）块以及使用生成的密钥流加速纯文本异或，支持对 CTR 加密模式进行加速
- 支持对 CBC-MAC 标签计算（具有零初始化矢量的 CBC DMA 模式）进行加速
- 动态密钥扩展进行加密和解密
- 用于解密的离线密钥生成
- 用于存储所有密钥长度的初始密钥的影子寄存器
- 8 位字节或 32 位字访问，以提供关键数据、输入数据和输出数据
- AES 就绪中断
- 在运行和睡眠模式下受支持（请参阅器件技术参考手册的*工作模式*部分）

AES 加速器硬件由 128 位状态存储器和相关的输入/输出寄存器、AES 加密/解密内核和控制逻辑、256 位 AES 密钥存储器和相关的输入寄存器组成。图 21-1 展示了 AES 硬件。

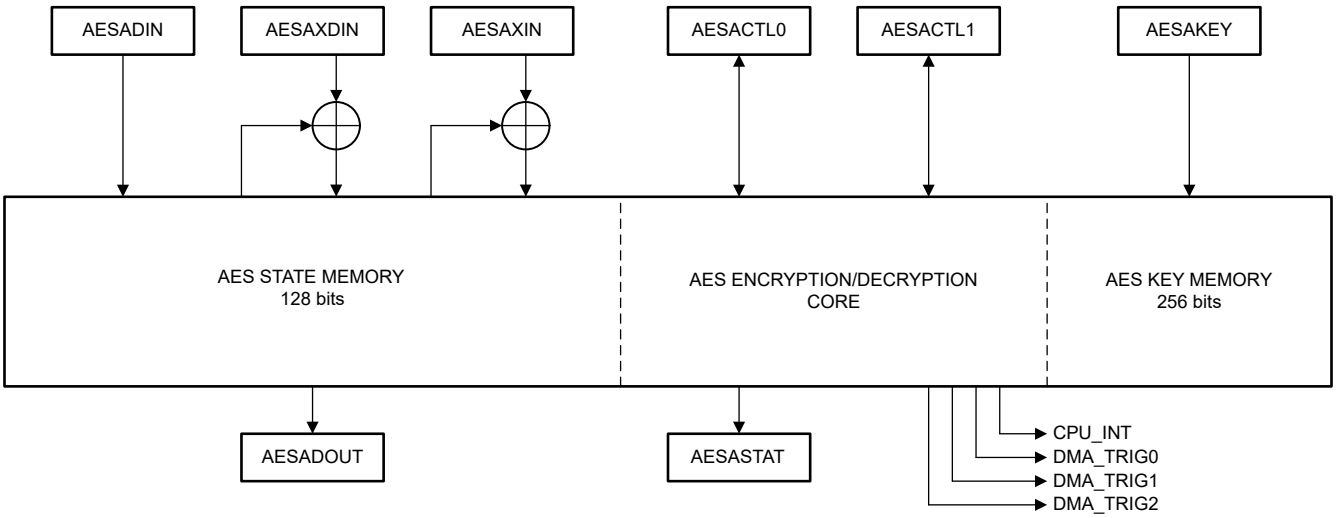


图 21-1. AES 加速器框图

21.1.1 AES 性能

AES 加速器可对 128 位块进行快速加密和解密。表 21-1 中以块加密和块解密（使用预先生成的解密密钥）的周期和执行时间形式给出了 AES 加速器性能。

表 21-1. AES 硬件加速器关键性能指标

AES 密钥长度	加密 (OPx==0x0)			解密 (OPx==0x3)		
	周期数	时间 (32MHz)	时间 (80MHz)	周期数	时间 (32MHz)	时间 (80MHz)
128 位	168	5.25 µs	2.10 µs	168	5.25 µs	2.10 µs
256 位	234	7.31 µs	2.93 µs	234	7.31 µs	2.93 µs

21.2 AES 运行

AES 加速器使用用户软件进行配置，有两种通用操作模式（使用 AESACTL0 寄存器进行配置）：

- **单块模式**

  - 在软件控制下一次加密或解密一个 128 位数据块
  - AES 硬件不直接生成 DMA 触发；应用软件写入和读取所有数据，或配置 DMA 以使用备用触发加载数据
  - 该模式还可用于预生成分组密码模式下自动解密所需的最后一轮密钥
- **自动分组密码模式**
  - 可以使用 ECB、CBC、CFB 或 OFB 分组密码模式操作以自动方式加密或解密多个 128 位数据块
  - 生成 DMA 触发，在 2 个或 3 个 DMA 通道（取决于所选的分组密码模式）的帮助下，自动将指定数量的块的数据移入和移出加速器

无论是单块模式还是分组密码模式，密钥长度和操作类型均由 AESACTL0 寄存器中的 KL 和 OP 字段指定。AES 软件复位控制（AESACTL0 寄存器中的 SWRST 位）不会复位 KLx 和 OPx 字段。更改 KLx 或 OPx 字段时，会重置密钥存储器。

- KLx 字段选择要使用的 AES 密钥大小（128 位或 256 位）。
- OPx 字段选择要执行的 AES 操作（加密、解密、生成最后一轮解密密钥、使用预先生成的最后一轮解密密钥进行解密）。

KLx 和 OPX 组合决定了每个块操作的周期计数，如表 21-2 所示。

表 21-2. OPx（操作）和 KLx（密钥长度）配置的 AES 操作概述

AESOPx	AESKLx	操作	时钟周期
00	00	AES128 加密	168
	10	AES256 加密	234
01	00	AES128 解密	215
	10	AES256 解密	292
10	00	执行 AES128 加密密钥调度（以生成最后一个轮密钥）	53
	10	执行 AES256 加密密钥调度（以生成最后一个轮密钥）	68
11	00	执行 AES128 解密（使用最后一个轮密钥）	168
	10	执行 AES256 解密（使用最后一个轮密钥）	234

在内部，AES 算法的运算是在称为 STATE 的二维字节数组上执行的。STATE 由四行字节组成，每行包含四个字节，与执行 AES128 还是 AES256 无关。输入被分配给 STATE 数组，如图 21-2 中所示，in[0] 是写入 AES 加速器输入寄存器（AESADIN、AESAXDIN 和 AESXIN）之一的第一个数据字节。然后对 STATE 数组进行加密或解密操作，之后可以从输出中读取其最终值，out[0] 是从 AES 加速器数据输出寄存器 (AESADOUT) 读取的第一个数据字节。

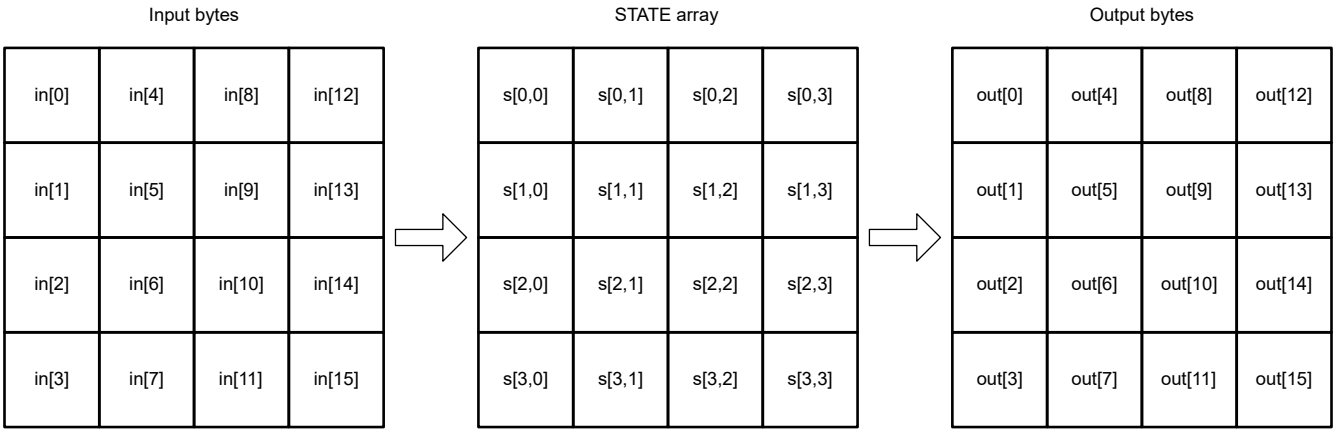


图 21-2. AES 状态数组输入和输出

如果要执行加密，则初始状态称为明文。如果要执行解密，则初始状态称为密文。

21.2.1 AES 寄存器访问规则

必须先启用 AES 加速器，然后再通过 PWREN 寄存器配置使用（请参阅外设电源使能）。AES 加速器位于电源域 1 (PD1) 中，因此仅在运行和睡眠模式下可用。如果应用软件启用了 AES 加速器，则进入停止或待机低功耗模式将强制在器件处于停止或待机模式时禁用 AES 加速器，并且 AES 加速器寄存器内容（包括任何密钥或状态内容）将丢失。

用于读取和写入密钥和数据的数据大小

该模块允许对所有密钥和数据寄存器（AESAKEY、AESADIN、AESAXDIN、AESAXIN 和 AESADOUT）进行 32 位字或 8 位字节访问。在读取或写入其中一个寄存器时，不得混合使用字和字节访问。但是，它是能够通过使用字节访问来写入一个寄存器，另一个寄存器可用字访问写入。

一般访问限制

当 AES 加速器繁忙时（在 AESASTAT 寄存器中设置 AESBUSY 位）：

- AESADOUT 始终读取为零。
- DOUTCNTx 计数器、DOUTRD 标志和 DINWR 标志将复位。
- 任何更改 OPx、KLx、DINWR 或 KEYWR 的尝试都将被忽略。
- 写入 AESAKEY、AESADIN、AESAXDIN 或 AESAXIN 会中止当前操作，复位整个模块（IMASK 寄存器、OPx 和 KLx 除外），并设置 AES 错误标志 ERRFG。

AESADIN、AESAXDIN、AESAXIN 和 AESAKEY 是只写寄存器，始终读取为零。

将数据写入 AESADIN、AESAXDIN 或 AESAXIN 会影响相应输出数据的内容；例如，写入 [0] 会更改输出 [0]，写入 [1] 会更改输出 [1]，以此类推。但是，交错操作是可能的；例如，先读取 [0]，然后写入 [0]，然后继续读取 [1]，然后写入 [1]，以此类推。此交错操作必须是对 in[x] 和 out[x] 的字节或字访问。

启用分组密码模式时的访问限制

启用自动分组密码模式（在 AESACTL0 中设置 CMEN）并正在处理密码分组操作 (BLKCNTx > 0) 时，将忽略对以下位的写入，与 BUSY 位状态无关：CMEN、CMx、KLx、OPX 和 BLKCNTx。如果 BUSY = 1，则写入 AESAKEY 将中止密码分组模式运行，并且复位整个模块（IMASK 寄存器、OPx 和 KLx 除外）。

21.2.2 加载密钥

可以通过写入 AESAKEY 寄存器或设置 KEYWR 位来加载密钥。根据所选的密钥长度（AESCTL0 寄存器中的 KLx），必须加载不同数量的位：

- 如果 KLx = 00，则必须使用 16 字节写入或 4 字写入 AESAKEY 来加载 128 位密钥
- 如果 KLx = 10，则必须使用 32 字节写入或 8 字写入 AESAKEY 来加载 256 位密钥

更改 KLx 位后，密钥存储器将复位。

如果之前加载了密钥而没有更改 OPx 位，则 KEYWR 标志将在对 AESAKEY 的第一次写访问中清除。

如果在未写入新密钥的情况下触发操作，则使用最后一个密钥。在写入数据之前，必须始终写入密钥。

AES 算法不仅在 STATE 上运行，而且在密钥上运行。为了避免需要为每次操作重新加载密钥，加入一个密钥缓冲区，以便用于轮次密钥生成的密钥扩展操作不会删除 AESAKEY 中加载的密码密钥。

21.2.3 正在加载数据

可以通过 16 字节或 4 字写入 AESADIN、AESAXDIN 或 AESAXIN 来加载状态。写入状态时不要混合字节和字模式。允许使用相同的字节或字数据格式写入 AESADIN、AESAXDIN 和 AESAXIN 的组合。当写入状态的第 16 个字节或第 4 个字时，在 AESASTAT 寄存器中设置 DINWR。

写入 AESADIN 时，状态的相应字节或字将被覆盖。如果使用 AESADIN 写入状态的最后一个字节或字，则会自动开始加密或解密。

写入 AESAXDIN 时，相应的字节或字与状态的当前字节或字进行“异或”运算。如果使用 AESAXDIN 写入状态的最后一个字节或字，则会自动开始加密或解密。

写入 AESAXIN 与写入 AESAXIN 的行为相同：相应的字节或字与状态的当前字节或字进行“异或”运算；但是，使用 AESAXIN 写入状态的最后一个字节或字不会启动加密或解密。

备注

当使用 AES 加速器协助实现 CTR 密码模式时，AESAXIN 寄存器可用于加速具有加密密钥流（随机数 || 计数器）的明文的“异或”运算。在 CTR 密码模式场景中，在（随机数 || 计数器）值加密完成后，可以将明文分组加载到 AESAXIN，因为加密的（随机数 || 计数器）值将保留在 STATE 中。将数据加载到 AESAXIN 后，可以从 AESADOUT 读取（明文 （随机数 || 计数器）输出。

21.2.4 读取数据

当 AESASTAT 中的 BUSY 位被清除时，可以读取 STATE。使用 16 字节读取或 4 字读取从 AESADOUT 读取 STATE。当读取所有 16 个字节时，AESASTAT 中的 DOUTRD 标志指示读取完成。

21.2.5 触发加密或解密

如果 STATE 完全写入 AESADIN 或 AESAXDIN 寄存器，则会触发加密或解密操作。或者，如果 CMEN 被清零，则可以设置 AESASTAT 寄存器中的 DINWR 位来触发一个操作。

21.2.6 单块操作

单块操作包括加密、解密和解密密钥生成。

21.2.6.1 加密

要执行加密，请进行以下操作：

1. 设置 OPx = 0x0 以选择加密。请注意，更改 OPx 位会清除 OPx 标志，并必须在下一步骤中载入一个新密钥。
2. 按照 节 21.2.2 中所述加载密钥。
3. 按照 节 21.2.3 中所述加载 STATE（数据）。载入数据后，AES 模块便会开始加密过程。
4. 等待操作完成（等待 BUSY 标志清除）。
5. 加密完成后，可以从 AESADOUT 读取结果，如 节 21.2.4 所述。
6. 要使用步骤 2 中载入的同一密钥来加密其他数据，请在从 AESADOUT 中读取之前数据的操作结果之后将新数据写入 AESADIN。当写入额外的 16 数据字节时，该模块会通过使用步骤 2 中载入的密钥自动启动加密。

备注

将 AESASTAT 寄存器中的 DINWR 位置位将触发另一个加密，并且 AES 加速器会使用之前加密中的输出数据作为输入数据来启动加密。这在具有反馈的特定块加密模式中很有用。

图 21-3 中给出了加密过程（显示了 128 位密钥和 10 轮加密）。

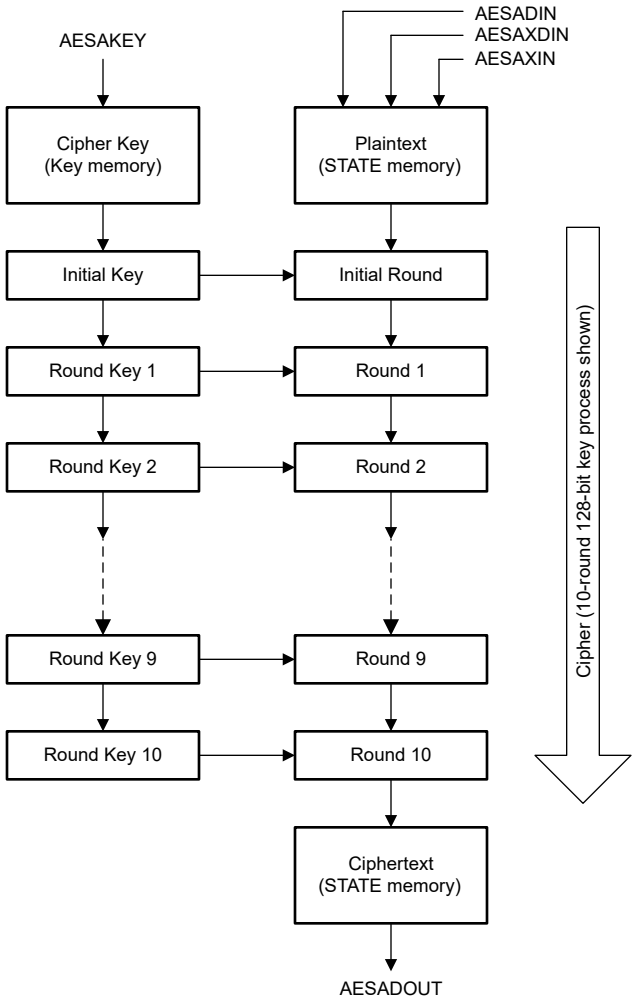


图 21-3. AES 加密（OPx=0x0，128 位密钥）

21.2.6.2 解密

要执行解密，请进行以下操作：

1. 设置 OPx = 0x1 以选择使用用于加密的同一密钥来解密，或者如果解密所需的第一轮密钥（用于加密的最后一轮密钥）已经预先生成并将在步骤 2 中载入，则设置 OPx = 0x3。更改 OPx 位会清除 KEYWR 标志，并必须在步骤 2 中载入一个新密钥。
2. 按照 节 21.2.2 中所述载入密钥。
3. 按照 节 21.2.3 中所述载入 STATE（数据）。载入数据后，AES 模块便会开始解密。
4. 等待操作完成（等待 BUSY 标志清除）。
5. 解密完成后，可以从 AESADOUT 读取结果，如 节 21.2.4 所述。
6. 如果需用用步骤 2 中载入的相同的密钥来解密额外数据，可在从 AESADOUT 中读取之前数据的操作结果之后把新数据写入 AESADIN。当写入一个额外 16 数据字节时，该模块会通过使用步骤 2 中载入的密钥来自动起始解密。

图 21-4 中给出了解密过程以及解密轮次密钥生成（显示了 128 位密钥和 10 轮解密）。

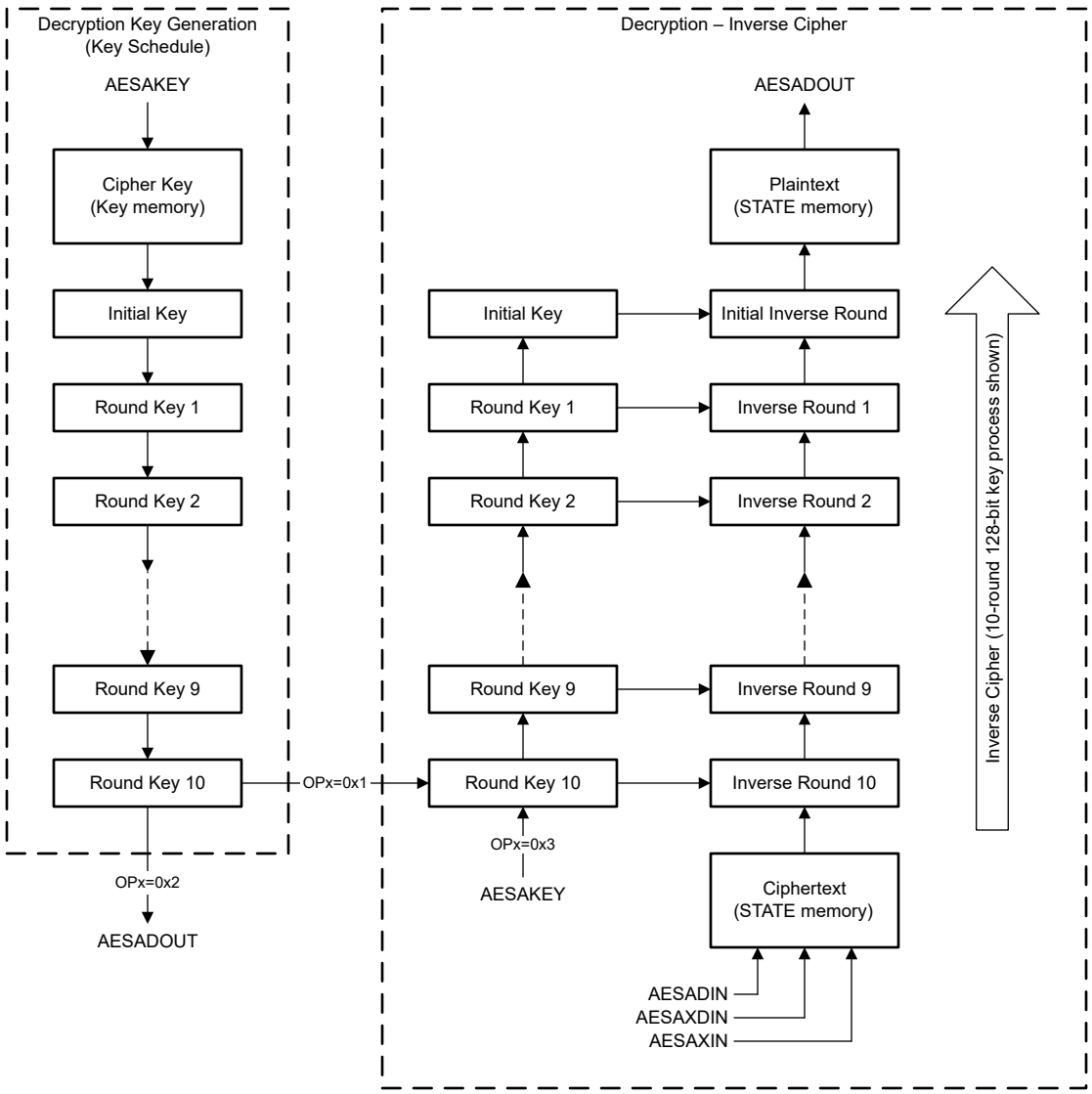


图 21-4. AES 解密（OPx = 0x1，128 位密钥）

21.2.6.2.1 重新生成解密密钥

当使用 OPx=0x1 进行解密时，用于加密的原始密码密钥会加载到 AESAKEY 中。解密需要执行密钥调度才能获取用于最后一轮加密的轮次密钥。然后，解密逻辑将最终加密轮次密钥用作首轮解密密钥。如果 OPx=0x1，在解密开始之前，将在加载到 AESAKEY 中的密钥材料上执行密钥调度以获得最后一轮密钥。生成用于加密的最后一轮密钥后，解密过程将自动开始。

虽然 OPx=0x1 对于单个块情况很方便（因为可以在没有特殊处理的情况下加载原始加密密钥），但对处理多个块不是很有效，因为需要针对每个块执行密钥调度，才能获取开始解密过程所需的最后一轮密钥。为了减少解密所需的周期计数，可以使用 OPx=0x2 重新生成最后一轮加密密钥。然后，通过加载重新生成的密钥材料并运行 OPx=0x3 解密操作，可以使用重新生成的轮次密钥立即开始任何解密操作，该操作会假定密钥材料是来自最后一轮加密的轮次密钥。

要执行密钥调度并获取用于加密的最后一轮密钥（用于解密的首轮密钥），而不实际运行解密操作，请执行以下步骤：

1. 将 OPx 设置为 0x2，选择解密密钥的生成。
2. 将用于加密的密码密钥加载到 AESAKEY 中，如节 21.2.2 所述。密钥材料加载完成后即开始生成。

3. 在 AES 模块忙于执行密钥调度时，设置 AESBUSY 位。完成 128 位密码密钥的密钥生成过程需要 53 个时钟周期。完成后，设置 AESRDY 中断，128 位结果可从 AESDOUT 读取。在读取 AESADOUT 或者写入 AESAKEY 或 AESADIN 时，AESRDY 被清除。

生成解密首轮密钥后，可将其加载到 AESAKEY 中并在 OPx=3 时用于解密。

21.2.7 分组密码模式操作

分组密码模式用于将 AES 作为底层分组密码来实现电码本 (ECB)、密码分组链接 (CBC)、输出反馈 (OFB) 和密码反馈 (CFB)。这些模式通过 2 或 3 个 DMA 触发来利用 DMA，有助于轻松、快速实现 128 位以上的加密或解密功能。

21.2.7.1 电码本 (ECB) 模式

电码本 (ECB) 密码是最简单的分组密码模式。明文数据拆分为多个 128 位分组，每个分组都独立于任何其他分组进行加密和解密。ECB 密码如图 21-5 所示。请注意，您可以在不了解其他明文分组或密文分组的情况下单独加密或解密每个 128 位数据分组。

虽然 ECB 易于理解和实施，但它有一个主要缺点：相同的 128 位明文分组始终加密到相同的密文分组中，使得密文模式可被发现。

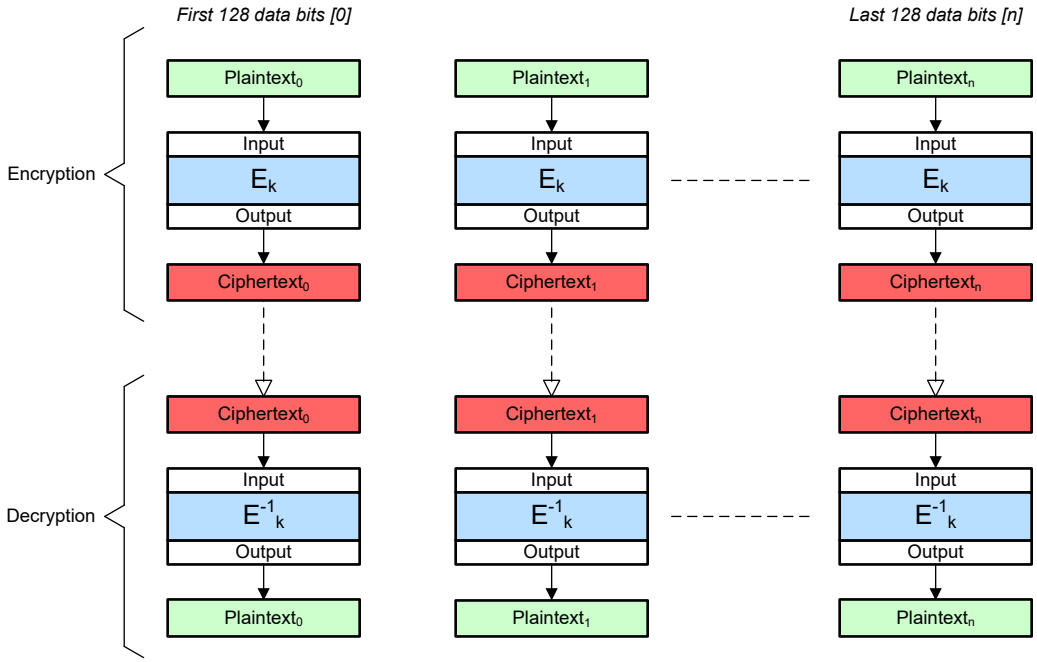


图 21-5. ECB 密码

AES 加速器支持通过结合使用 DMA 与 AES 加速器，在 ECB 分组密码模式下对 128 位以上数据进行自动加密和解密。ECB 模式将两个 DMA 通道（本节中称为 DMA_A 和 DMA_B）分别用于 AES 加速器的输出和输入。表 21-3 中给出了使用 DMA 通道在 ECB 模式下进行加密和解密操作的方法。

表 21-3. ECB 分组密码 DMA 行为

AES 运行			AES DMA 触发		
CMEN	CMx	OPx	DMA_A (DMA_TRIG0)	DMA_B (DMA_TRIG1)	DMA_C (DMA_TRIG2)
0x1	0x0 (ECB)	0x0（加密）	用于从 AESADOUT 读取密文	用于将明文写入 AESADIN（当 128 位写入 AESADIN 时，也会触发分组加密的开始）	未使用
		0x1/0x2（解密）	用于从 AESADOUT 读取明文	用于将密文写入 AESADIN（当 128 位写入 AESADIN 时，也会触发分组解密的开始）	未使用

21.2.7.1.1 ECB 加密

通过使用 2 个 DMA 通道（称作 DMA_A 和 DMA_B），可以在 ECB 模式下将 *N* 个明文块加密为 *N* 个密文块，而无需 CPU 交互。要实现 ECB 加密，请按照下列步骤操作：

1. 使用 ECB 将 AESACTL0 寄存器配置为分组密码加密模式：
   a. 设置 CMEN 以启用分组密码模式
   b. 将 CMx 设置为 ECB 模式
   c. 将 OPx 设置为 0x0（加密模式）
2. 按节 21.2.2 中所述加载密钥
3. 配置 DMA_A 通道以保存密文：
   a. 将 DMA 通道触发选择设置为 AES0 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储密文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N**4
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG0 的 IMASK 寄存器中的 DMA0
4. 配置 DMA_B 通道以加载明文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为存储明文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESADIN
   d. 将 DMA 通道传输大小设置为 *N**4
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

   1
5. 为 DMA 控制器中的 DMA_A 通道配置并启用 DMA 中断
6. 通过将块计数 *N* 写入 AESACTL1 寄存器中的 BLKCNTx 来开始加密
7. 等待表示操作完成的 DMA_A 通道中断。密文输出将存储在步骤 3c 中配置的位置。

21.2.7.1.2 ECB 解密

通过使用 2 个 DMA 通道（称作 DMA_A 和 DMA_B），可以在 ECB 模式下将 *N* 个密文块解密为 *N* 个明文块，而无需 CPU 交互。要实现 ECB 解密，请按照下列步骤操作：

1. 配置 AESACTL0 寄存器以预生成解密密钥（最后一轮密钥）：
   a. 清除 CMEN 以禁用分组密码模式
   b. 将 OPx 设置为 0x2（解密密钥生成）
   c. 如节 21.2.2 中所述，将密钥写入 AESAKEY 寄存器
   d. 等待 AESSTAT 寄存器中的 BUSY 状态清除，这表示密钥生成已经完成
2. 使用 ECB 将 AESACTL0 寄存器配置为分组密码解密模式：
   a. 设置 CMEN 以启用分组密码模式
   b. 将 CMx 设置为 ECB 模式
   c. 将 OPx 设置为 0x3（解密模式）

3. 设置 AESSTAT 中的 AESKEYWR 位（以使用步骤 1 中预生成的密钥）
4. 配置 DMA_A 通道以保存明文：
   a. 将 DMA 通道触发选择设置为 AES0 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储明文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N**4
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA0

      0
5. 配置 DMA_B 通道以加载密文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为存储密文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESADIN
   d. 将 DMA 通道传输大小设置为 *N**4
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

      1
6. 为 DMA 控制器中的 DMA_A 通道配置并启用 DMA 中断
7. 通过在 AESACTL1 寄存器中将块计数 *N* 写入 BLKCNTx 来开始解密
8. 等待表示操作完成的 DMA_A 通道中断。明文输出将存储在步骤 3c 中配置的位置。

21.2.7.2 密码分组链接 (CBC) 模式

密码分组链接 (CBC) 密码模式建立在 ECB 密码模式的基础上，使每个块的密文输出不仅取决于明文和密码密钥 *k*，而且还取决于前一个块的密文。CBC 密码如图 21-6 所示。与 ECB 模式一样，明文数据分成 128 位块。与 ECB 模式不同，在 CBC 模式中，每个新的明文块与之前的密文块进行逐位异或运算，以创建 AES 块密码的输入。

在 CBC 模式下，必须提供一个不可预测的初始化矢量 (IV)。初始化矢量与第一个明文块进行异或运算，因为没有“前一个”密文块与第一个明文块进行异或运算。

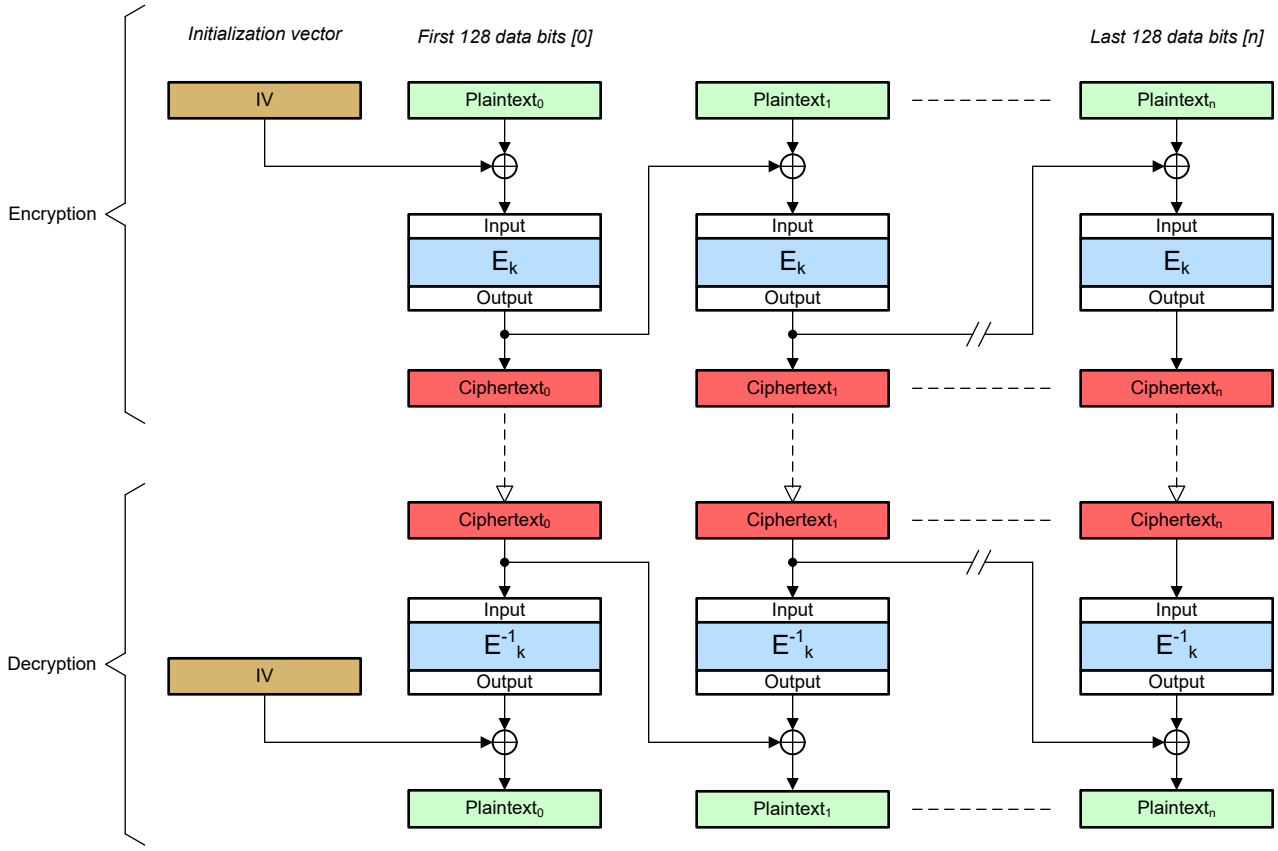


图 21-6. CBC 密码

AES 加速器支持通过结合使用 DMA 与 AES 加速器，在 CBC 分组密码模式下对 128 位以上数据进行自动加密和解密。在加密时，CBC 模式将两个 DMA 通道（本节中称为 DMA_A 和 DMA_B）分别用于 AES 加速器的输出和输入。在解密时，CBC 模式将三个 DMA 通道（DMA_A、DMA_B 和 DMA_C）用于 AES 加速器的输入、输出和输入。表 21-4 中给出了使用 DMA 通道在 CBC 模式下进行加密和解密操作的方法。

表 21-4. CBC 分组密码 DMA 行为

AES 运行			AES DMA 触发		
CMEN	**CMx**	**OPx**	**DMA_A (DMA_TRIG0)**	**DMA_B (DMA_TRIG1)**	**DMA_C (DMA_TRIG2)**
0x1	0x1 (CBC)	0x0（加密）	用于从 AESADOUT 读取密文	用于将明文写入 AESAXIN（当将 128 位写入 AESADIN 时，也会触发分组加密的开始）	未使用
		0x1/0x2（解密）	用于将初始化矢量和之前的密文块写入 AESAXIN	用于从 AESADOUT 读取明文	用于将下一个密文写入 AESADIN（当将 128 位写入 AESADIN 时，也会触发分组解密的开始）

21.2.7.2.1 CBC 加密

通过使用 2 个 DMA 通道（称作 DMA_A 和 DMA_B），可以在 CBC 模式下将 *N* 个明文块加密为 *N* 个密文块，而无需 CPU 交互。要实现 CBC 加密，请按照下列步骤操作：

1. 使用 CBC 将 AESACTL0 寄存器配置为分组密码加密模式：
   a. 复位 AES 模块以清除内部状态存储器（设置 AESACTL0 寄存器中的 SWRST）
   b. 设置 CMEN 以启用分组密码模式
   c. 将 CMx 设置为 CBC 模式
   d. 将 OPx 设置为 0x0（加密模式）
2. 按节 21.2.2 中所述加载密钥

3. 将初始化矢量 (IV) 载入 AESAXIN
   a. 加载到 AESAXIN 不会启动任何加密
   b. 必须复位 STATE（上述步骤 1a），以便加载时 IV 与零进行“异或”运算
4. 配置 DMA_A 通道以保存密文：
   a. 将 DMA 通道触发选择设置为 AES0 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储密文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA0

      0
5. 配置 DMA_B 通道以加载明文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为存储明文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESAXDIN
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

      1
6. 为 DMA 控制器中的 DMA_A 通道配置并启用 DMA 中断
7. 通过将块计数 *N* 写入 AESACTL1 寄存器中的 BLKCNTx 来开始加密
8. 等待表示操作完成的 DMA_A 通道中断。密文输出将存储在步骤 4c 中配置的位置。

21.2.7.2.2 CBC 解密

通过使用 3 个 DMA 通道（称作 DMA_A、DMA_B 和 DMA_C），可以在 CBC 模式下将 *N* 个密文块解密为 *N* 个明文块，而无需 CPU 交互。要实现 CBC 解密，请按照下列步骤操作：

1. 配置 AESACTL0 寄存器来预生成解密密钥（最后一轮密钥）：
   a. 清除 CMEN 以禁用分组密码模式
   b. 将 OPx 设置为 0x2（解密密钥生成）
   c. 如节 21.2.2 中所述，将密钥写入 AESAKEY 寄存器
   d. 等待 AESSTAT 寄存器中的 BUSY 状态清除，这表示密钥生成已经完成
2. 使用 CBC 将 AESACTL0 寄存器配置为分组密码解密模式：
   a. 设置 CMEN 来启用分组密码模式 (0x1)
   b. 将 CMx 设置为 0x1（CBC 模式）
   c. 将 OPx 设置为 0x3（解密模式）
3. 设置 AESSTAT 中的 AESKEYWR 位（来使用步骤 1 中预生成的密钥）
4. 配置 DMA_A 通道来加载初始化矢量和密文：
   a. 在存储器中设置密文缓冲区，并在密文前加上初始化矢量 (IV)
   b. 将 DMA 通道触发选择设置为 AES0 触发
   c. 将 DMA 通道源地址设置为在密文前存储初始化矢量的位置（例如，SRAM）
   d. 将 DMA 通道目标地址设置为 AESAXIN
   e. 将 DMA 通道传输大小设置为 *N**4 个字
   f. 将 DMA 通道模式设置为单字或单字节传输模式
   g. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA0
5. 配置 DMA_B 通道来保存明文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储明文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N**4 个字

e. 将 DMA 通道模式设置为单字或单字节传输模式
f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

6. 配置 DMA_C 通道来加载密文：
   a. 将 DMA 通道触发选择设置为 AES2 触发
   b. 将 DMA 通道源地址设置为在初始化矢量之后存储密文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESADIN
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA2
7. 为 DMA 控制器中的 DMA_A 和 DMA_B 通道配置并启用 DMA 中断
8. 通过在 AESACTL1 寄存器中将块计数 *N* 写入 BLKCNTx 来开始解密
9. 等待表示操作完成的 DMA_B 通道中断。明文输出将存储在步骤 5c 中配置的位置。

21.2.7.3 输出反馈 (OFB) 模式

输出反馈模式利用初始化矢量 (IV) 通过使用密码密钥对 IV 重复加密来生成密钥流。输出密文是通过对明文与初始化矢量的加密和重新加密版本进行“异或”运算获得的。OFB 密码如图 21-7 所示。

在 OFB 模式下，初始化矢量必须为随机数（一次性数字）。为了防止丧失机密性，每个 IV 只能与给定密钥一起使用一次，并且任何传入密码 E_k 以匹配给定秘钥 *k* 的值都不得用作具有相同密钥 *k* 的初始化矢量。

备注

由于 OFB 是一种流密码，因此，在执行 OFB 加密或 OFB 解密时，在正向（加密）模式下使用 AES 分组函数。解密时，只需重新生成密钥流，就可以与密文进行“异或”运算，从而得到明文。

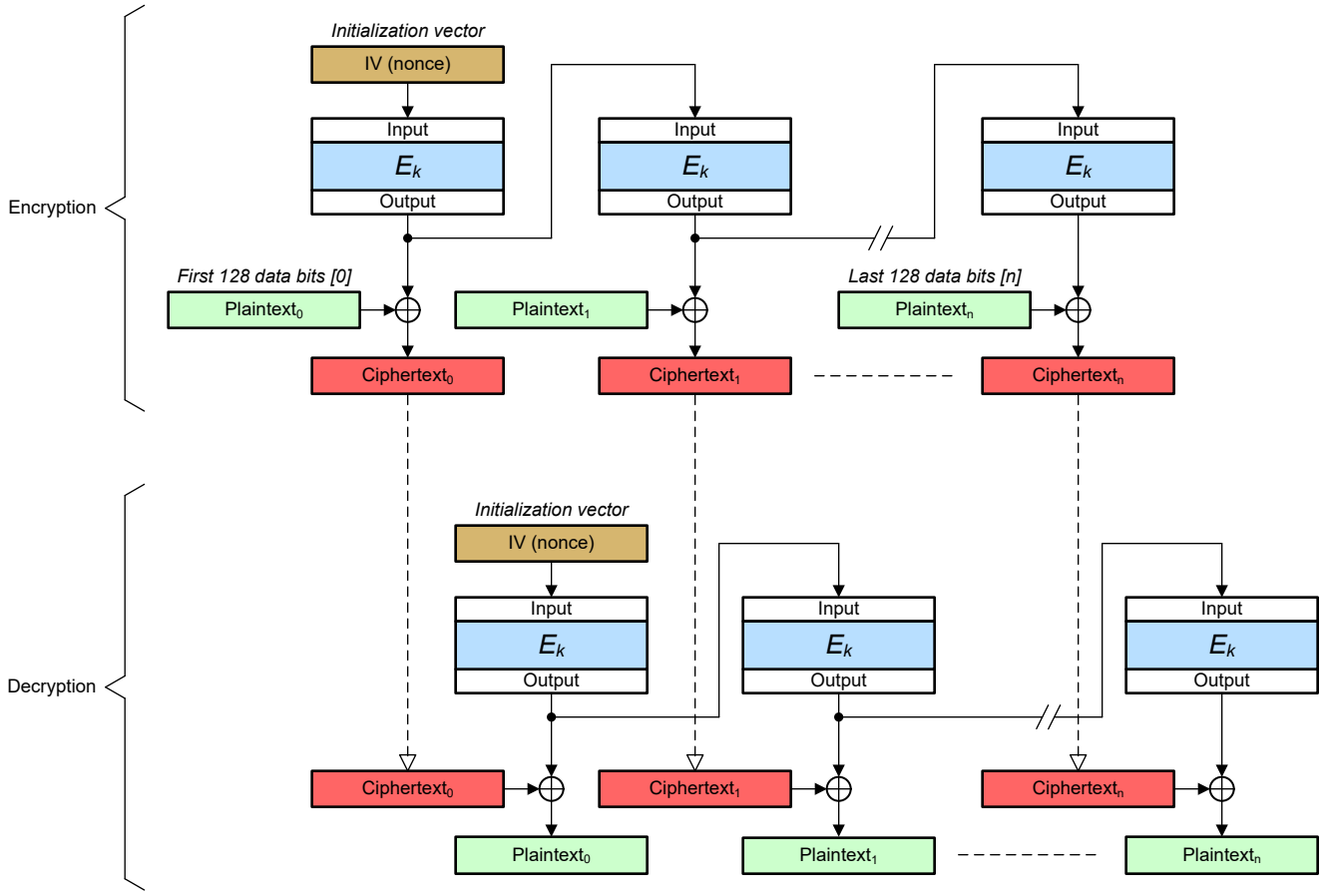


图 21-7. OFB 密码

AES 加速器支持通过结合使用 DMA 与 AES 加速器，在 OFB 分组密码模式下对 128 位以上数据进行自动加密和解密。OFB 模式将三个 DMA 通道（本节中称为 DMA_A、DMA_B 和 DMA_C）分别用于 AES 加速器的输入、输出和输入。在表 21-5 中给出了使用 DMA 通道在 OFB 模式下进行加密和解密操作的方法。

表 21-5. OFB 密码 DMA 行为

AES 运行			AES DMA 触发		
CMEN	**CMx**	**OPx**	**DMA_A (DMA_TRIG0)**	**DMA_B (DMA_TRIG1)**	**DMA_C (DMA_TRIG2DMA_TRIG)**
0x1	0x2 (OFB)	0x0（加密）	用于将当前块的明文写入 AESAXIN	用于从 AESADOUT 读取密文	用于将当前块的明文写入 AESAXDIN（这会触发下一次加密）
		0x1（解密）	用于将当前块的密文写入 AESAXIN	用于从 AESADOUT 读取明文	用于将当前块的密文写入 AESAXDIN（这会触发下一次解密）

21.2.7.3.1 OFB 加密

通过使用 3 个 DMA 通道（称作 DMA_A、DMA_B 和 DMA_C），可以在 OFB 模式下将 *N* 个明文块加密为 *N* 个密文块，而无需 CPU 交互。要实现 OFB 加密，请按照下列步骤操作：

1. 使用 OFB 将 AESACTL0 寄存器配置为分组密码加密模式：
   a. 复位 AES 模块以清除内部状态存储器（设置 AESACTL0 寄存器中的 SWRST）
   b. 设置 CMEN 以启用分组密码模式
   c. 将 CMx 设置为 OFB 模式 (0x2)
   d. 将 OPx 设置为 0x0（加密模式）
2. 按节 21.2.2 中所述加载密钥
3. 将初始化矢量 (IV) 载入 AESAXIN
   a. 加载到 AESAXIN 不会启动任何加密
   b. 必须复位 STATE（上述步骤 1a），以便加载时 IV 与零进行“异或”运算
4. 配置 DMA_A 通道以加载明文：
   a. 将 DMA 通道触发选择设置为 AES0 触发
   b. 将 DMA 通道源地址设置为存储明文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESAXIN
   d. 将 DMA 通道传输大小设置为 *N*×4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA0

      0
5. 配置 DMA_B 通道以读取密文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储密文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N*×4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

      1
6. 配置 DMA_C 通道以加载明文：
   a. 将 DMA 通道触发选择设置为 AES2 触发
   b. 将 DMA 通道源地址设置为存储明文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESAXDIN
   d. 将 DMA 通道传输大小设置为 *N*×4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
7. 为 DMA 控制器中的 DMA_B 通道配置并启用 DMA 中断

8. 通过将块计数 *N* 写入 AESACTL1 寄存器中的 BLKCNTx，并设置 AESASTAT 寄存器中的 DINWR 位来开始加密
9. 等待表示操作完成的 DMA_B 通道中断。密文输出将存储在步骤 5c 中配置的位置。

21.2.7.3.2 OFB 解密

通过使用 3 个 DMA 通道（称作 DMA_A、DMA_B 和 DMA_C），可以在 OFB 模式下将 *N* 个密文块解密为 *N* 个密文块，而无需 CPU 交互。要实现 OFB 解密，请按照下列步骤操作：

1. 使用 OFB 将 AESACTL0 寄存器配置为分组密码解密模式：
   a. 复位 AES 模块以清除内部状态存储器（在 AESCTL0 寄存器中设置 SWRST）
   b. 设置 CMEN 以启用分组密码模式
   c. 将 CMx 设置为 OFB 模式 (0x2)
   d. 将 OPx 设置为解密模式 (0x1)
2. 按节 21.2.2 中所述加载密钥
3. 将初始化矢量 IV 载入 AESAXIN
   a. 加载到 AESAXIN 不会启动任何解密
   b. 必须复位 STATE（上述步骤 1a），以便加载时 IV 与零进行“异或”运算
4. 配置 DMA_A 通道以加载密文：
   a. 将 DMA 通道触发选择设置为 AES0 触发
   b. 将 DMA 通道源地址设置为存储密文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESAXIN
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
5. 配置 DMA_B 通道以读取明文：
   a. 将 DMA 通道触发选择设置为 AES1 触发
   b. 将 DMA 通道源地址设置为 AESADOUT
   c. 将 DMA 通道目标地址设置为要存储明文的位置（例如，SRAM）
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
   f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

      1
6. 配置 DMA_C 通道以加载密文：
   a. 将 DMA 通道触发选择设置为 AES2 触发
   b. 将 DMA 通道源地址设置为存储密文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESAXDIN
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
7. 为 DMA 控制器中的 DMA_B 通道配置并启用 DMA 中断
8. 通过在 AESACTL1 寄存器中将块计数 *N* 写入 BLKCNTx，并在 AESASTAT 寄存器中设置 DINWR 位来开始解密
9. 等待表示操作完成的 DMA_B 通道中断。明文输出将存储在步骤 5c 中配置的位置。

21.2.7.4 密码反馈 (CFB) 模式

密码反馈 (CFB) 模式与输出反馈 (OFB) 模式类似，主要区别在于用于生成密钥流的分组密码 ***E*** 的输入块是从之前的密文块中获取的（在与明文进行异或运算之后），而不是在与明文进行异或运算之前获取的（例如在 OFB 模式下）。因此，密钥流取决于明文，而 OFB 则不是。CFB 密码如图 21-8 所示。

与 OFB 类似，CFB 需要一个初始化矢量 (IV)。在 CFB 模式下，初始化矢量 (IV) 必须不可预测。

备注

由于 CFB 是一种流密码，因此，在执行 CFB 加密或 CFB 解密时，使用 AES 块功能在正向（加密）模式下进行操作。

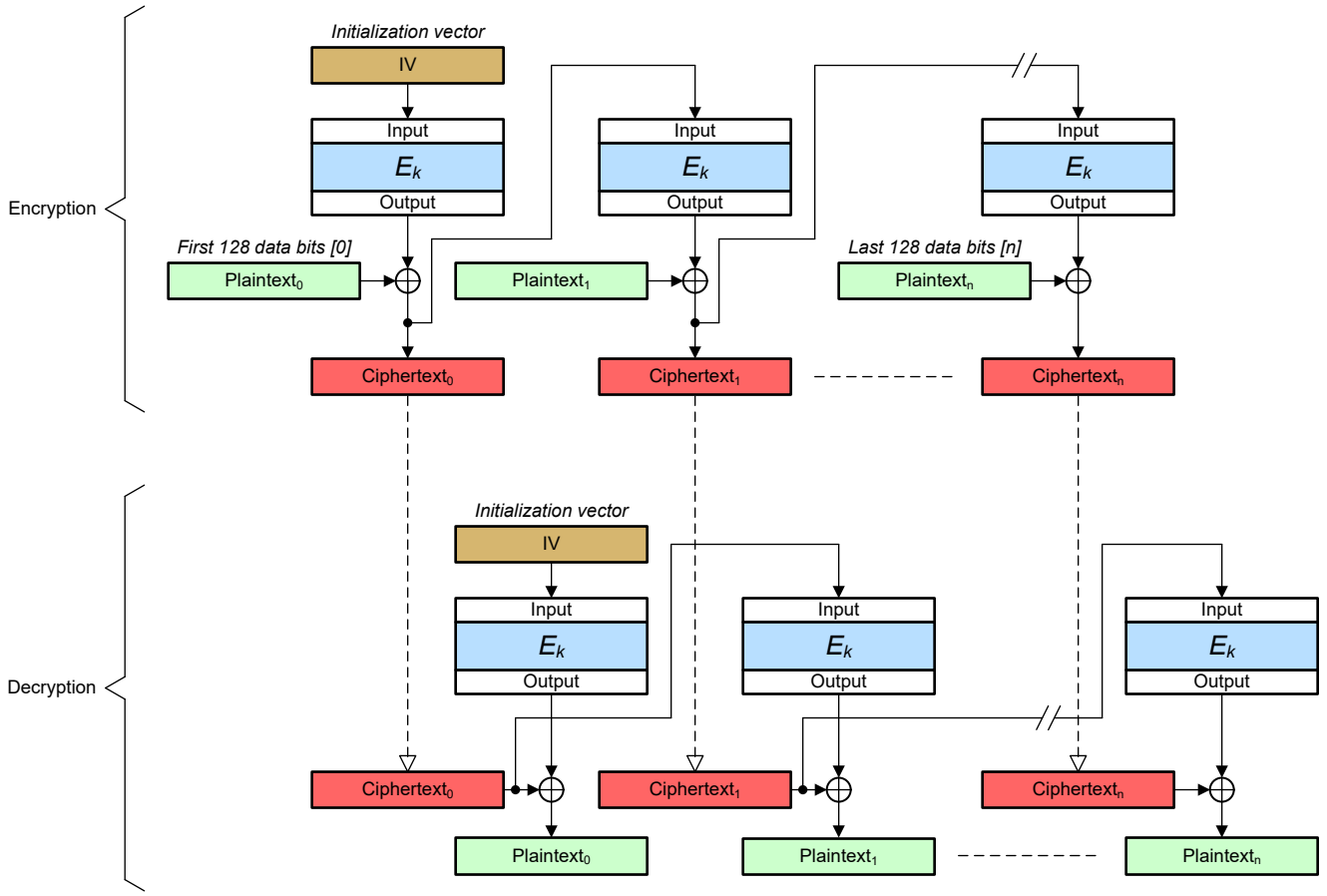


图 21-8. CFB 密码

AES 加速器支持通过结合使用 DMA 与 AES 加速器，在 CFB 分组密码模式下对 128 位以上数据进行自动加密和解密。CFB 模式将三个 DMA 通道（本节中称为 DMA_A、DMA_B 和 DMA_C）分别用于 AES 加速器的输入、输出和输入。表 21-6 中给出了使用 DMA 通道在 CFB 模式下进行加密和解密操作的方法。

表 21-6. CFB 密码 DMA 行为

AES 运行			AES DMA 触发		
CMEN	**CMx**	**OPx**	**DMA_A (DMA_TRIG0)**	**DMA_B (DMA_TRIG1)**	**DMA_C (DMA_TRIG2)**
0x1	0x3 (CFB)	0x0（加密）	用于将当前块的明文写入 AESAXIN	用于从 AESADOUT 读取密文（这也会触发下一次解密）	未使用
		0x1（解密）	用于将当前块的密文写入 AESAXIN	用于从 AESADOUT 读取明文	用于将当前块的密文写入 AESADIN（这也会触发下一次解密）

21.2.7.4.1 CFB 加密

通过使用两个 DMA 通道（称作 DMA_A 和 DMA_B），可以在 CFB 模式下将 *N* 个明文块加密为 *N* 个密文块，而无需 CPU 交互。要实现 CFB 加密，请按照下列步骤操作：

1. 使用 OFB 将 AESACTL0 寄存器配置为分组密码加密模式：
   a. 复位 AES 模块以清除内部状态存储器（设置 AESACTL0 寄存器中的 SWRST）

b. 设置 CMEN 以启用分组密码模式
c. 将 CMx 设置为 CFB 模式 (0x3)
d. 将 OPx 设置为 0x0（加密模式）
2. 按节 21.2.2 中所述加载密钥
3. 将初始化矢量 (IV) 载入 AESAXIN
a. 加载到 AESAXIN 不会启动任何加密
b. 必须复位 STATE（上述步骤 1a），以便加载时 IV 与零进行“异或”运算
4. 配置 DMA_A 通道以加载明文：
a. 将 DMA 通道触发选择设置为 AES0 触发
b. 将 DMA 通道源地址设置为存储明文的位置（例如，SRAM）
c. 将 DMA 通道目标地址设置为 AESAXIN
d. 将 DMA 通道传输大小设置为 *N**4 个字
e. 将 DMA 通道模式设置为单字或单字节传输模式
5. 配置 DMA_B 通道以读取密文：
a. 将 DMA 通道触发选择设置为 AES1 触发
b. 将 DMA 通道源地址设置为 AESADOUT
c. 将 DMA 通道目标地址设置为要存储密文的位置（例如，SRAM）
d. 将 DMA 通道传输大小设置为 *N**4 个字
e. 将 DMA 通道模式设置为单字或单字节传输模式
f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

1
6. 为 DMA 控制器中的 DMA_B 通道配置并启用 DMA 中断
7. 通过将块计数 *N* 写入 AESACTL1 寄存器中的 BLKCNTx，并设置 AESASTAT 寄存器中的 DINWR 位来开始加密
8. 等待表示操作完成的 DMA_B 通道中断。密文输出将存储在步骤 5c 中配置的位置。

21.2.7.4.2 CFB 解密

通过使用 3 个 DMA 通道（称为 DMA_A、DMA_B 和 DMA_C），在没有 CPU 交互的情况下实现将 *N* 个密文块解密为 *N* 个密文块的 CFB 模式解密。要实现 CFB 解密，请按照下列步骤操作：

1. 使用 OFB 将 AESACTL0 寄存器配置为分组密码解密模式：
a. 复位 AES 模块以清除内部状态存储器（在 AESCTL0 寄存器中设置 SWRST）
b. 设置 CMEN 以启用分组密码模式
c. 将 CMx 设置为 CFB 模式 (0x3)
d. 将 OPx 设置为解密模式 (0x1)
2. 按节 21.2.2 中所述加载密钥
3. 将初始化矢量 IV 载入 AESAXIN
a. 加载到 AESAXIN 不会启动任何解密
b. 必须复位 STATE（上述步骤 1a），以便加载时 IV 与零进行“异或”运算
4. 配置 DMA_A 通道以加载密文：
a. 将 DMA 通道触发选择设置为 AES0 触发
b. 将 DMA 通道源地址设置为存储密文的位置（例如，SRAM）
c. 将 DMA 通道目标地址设置为 AESAXIN
d. 将 DMA 通道传输大小设置为 *N**4 个字
e. 将 DMA 通道模式设置为单字或单字节传输模式
5. 配置 DMA_B 通道以读取明文：
a. 将 DMA 通道触发选择设置为 AES1 触发
b. 将 DMA 通道源地址设置为 AESADOUT
c. 将 DMA 通道目标地址设置为要存储明文的位置（例如，SRAM）
d. 将 DMA 通道传输大小设置为 *N**4 个字

e. 将 DMA 通道模式设置为单字或单字节传输模式
f. 在 AES 事件寄存器中，取消屏蔽 DMA_TRIG 的 IMASK 寄存器中的 DMA1

1

6. 配置 DMA_C 通道以加载密文：
   a. 将 DMA 通道触发选择设置为 AES2 触发
   b. 将 DMA 通道源地址设置为存储密文的位置（例如，SRAM）
   c. 将 DMA 通道目标地址设置为 AESADIN
   d. 将 DMA 通道传输大小设置为 *N**4 个字
   e. 将 DMA 通道模式设置为单字或单字节传输模式
7. 为 DMA 控制器中的 DMA_B 通道配置并启用 DMA 中断
8. 通过在 AESACTL1 寄存器中将块计数 *N* 写入 BLKCNTx，并在 AESASTAT 寄存器中设置 DINWR 位来开始解密
9. 等待表示操作完成的 DMA_B 通道中断。明文输出将存储在步骤 5c 中配置的位置。

21.2.7.5 计数器模式 (CTR)

计数器模式利用一次性随机数 (nonce) 和计数整数来生成密钥流，方法是将计数器附加到随机数，并使用密码密钥对组合的随机数 || 计数器值进行加密。

随机数只能与给定的密钥 *k* 一起使用一次。计数器的值可以从任何值开始，并在每个 128 位数据块中递增。

通过使用密码密钥 *k* 对每个 128 位数据块的随机数 || 计数器值进行加密，可以得到密钥流。然后，通过对每个数据块的明文与加密的随机数 || 计数器值进行“异或”运算，可以获得输出的密文。CTR 密码模式如图 21-9 所示。

备注

由于 CTR 模式是一种流密码，因此，在执行 CTR 加密或 CTR 解密时，在正向（加密）模式下使用 AES 分组函数。解密时，只需重新生成密钥流，就可以与密文进行“异或”运算，从而得到明文。

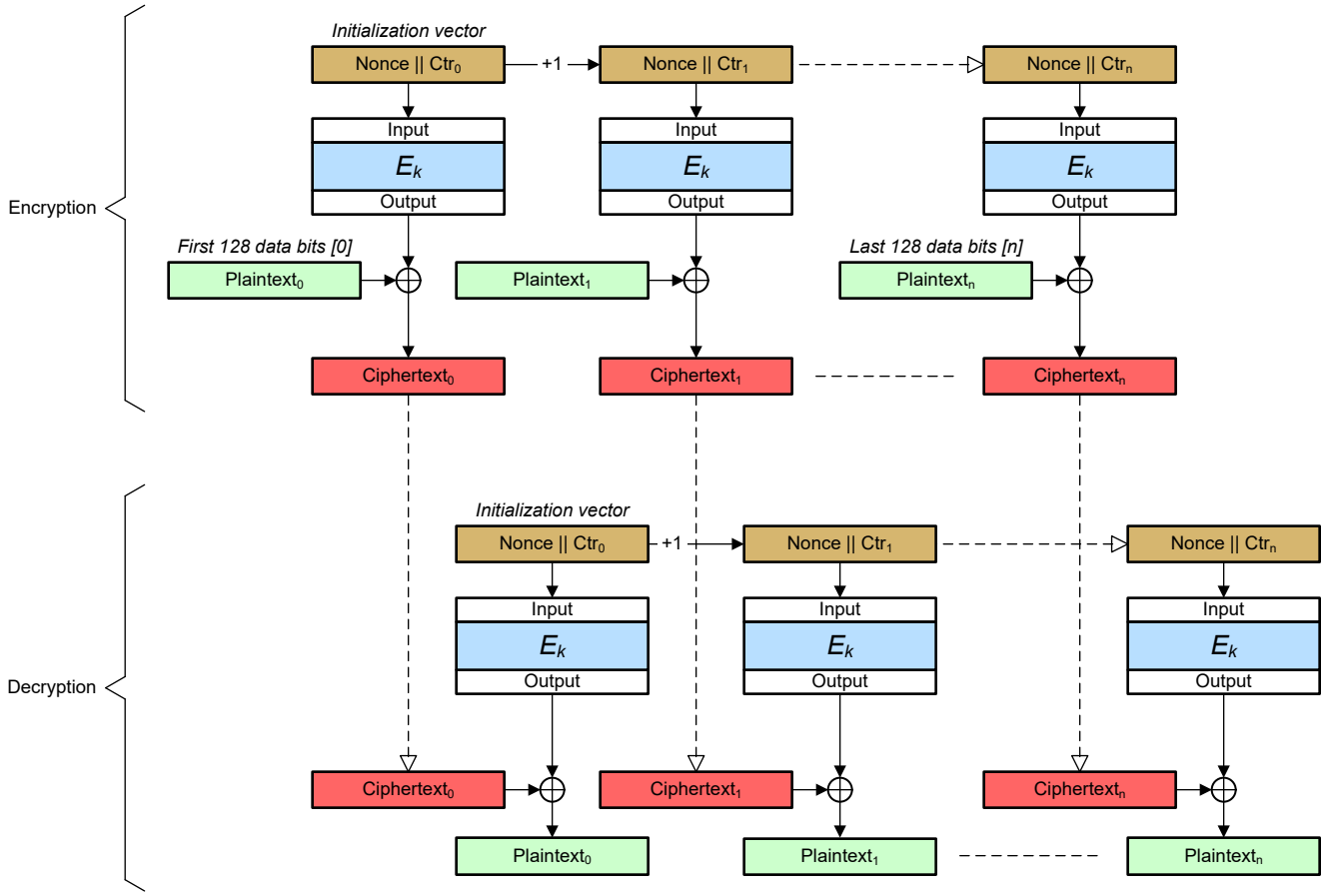


图 21-9. CTR 密码

可以使用 AES 加速器对 CTR 加密模式进行加速，方法是通过密码密钥 *k* 加速对每个随机数 || 计数器值进行加密，并加速明文（在加密的情况下）或密文（在解密的情况下）与密钥流的“异或”操作。在实现 CTR 密码时，不支持 DMA 触发生成。

21.2.7.5.1 CTR 加密

CTR 模式下，将 *N* 个明文块加密为 *N* 个密文块的过程如下所示：

1. 将 AESACTL0 寄存器配置为单块加密模式：
   a. 清除 CMEN 以禁用分组密码模式（如果已启用）
   b. 将 OPx 设置为 0x0（加密模式）
2. 按节 21.2.2 中所述加载密钥
3. 根据节 21.2.3 所述，将第一个（随机数 || 计数器）值加载到 AESADIN 中。
4. 等待加密过程完成（等待 BUSY 标志清除）。
5. 加密后的（随机数 || 计数器）值现在位于 STATE 中。第一个密文块可以通过将第一个明文块与 STATE 中的密钥流进行“异或”运算来获得。
   a. 将第一个明文块加载到 AESAXIN 中（对明文与密钥流进行“异或”运算，但不开始任何加密操作）。
6. 根据节 21.2.4 所述，从 AESADOUT 中读取第一个密文块。
7. 对于其他数据块，重复步骤 3-6，每次增加计数器的值。

21.2.7.5.2 CTR 解密

CTR 模式下，将 *N* 个密文块解密为 *N* 个明文块的过程与加密相同（参阅节 21.2.7.5.1），只是将密文替换为明文，反之亦然。请注意，在 CTR 模式下只使用正向密码；加密和解密操作是相同的，可以使用相同的软件实现，只需在解密时将密文替换为明文。

21.2.8 AES 事件

AES 模块包含四个事件发布者而没有事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理针对 CPU 子系统的 AES 中断请求 (IRQ)。第二、第三和第四个事件发布者（DMA_TRIG0、DMA_TRIG1 和 DMA_TRIG2）可通过 DMA 事件路由将 AES 事件发布到 DMA。

表 21-7 中总结了 AES 事件。

表 21-7. AES 事件

事件	类型	来源	目标	路由	配置	功能
CPU 中断事件	发布者	AES	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 RTC 到 CPU 的中断路由
DMA 触发事件 0	发布者	AES	DMA	DMA 路由	DMA_TRIG 0 寄存器	针对分组密码模式的 DMA 触发 0
DMA 触发事件 1	发布者	AES	DMA	DMA 路由	DMA_TRIG 1 寄存器	针对分组密码模式的 DMA 触发 1
DMA 触发事件 2	发布者	AES	DMA	DMA 路由	DMA_TRIG 2 寄存器	针对分组密码模式的 DMA 触发 2

通常，CPU 中断事件将 AES 操作的完成情况传达给 CPU，DMA 触发则使用 DMA 和 AES 加速器一起实现分组密码模式，包含 ECB、CBC、OFB 和 CFB。

当在 AESACTL0 中设置 CMEN 位时，AES 模块将触发 DMA 触发事件 0、DMA 触发事件 1 和 DMA 触发事件 2，从而与 DMA 一起执行不同分组密码模式的操作。

例如，当使用 ECB 加密模式且 AESCMEN 设置为 0x1 时，DMA 触发事件 0 和 AES 触发 1 将分别触发四次，分别用于 DMA 字访问来读取 AESADOUT 和将下一个数据填充到 AESADIN 中。AES 模块为每个字生成一个触发，因此 DMA 必须配置为单通道重复模式。有关如何配置 DMA 来支持 ECB、CBC、OFB 和 CFB 分组密码模式的详细信息，请参阅分组密码模式部分。

21.2.8.1 CPU 中断事件发布者 (CPU_EVENT)

AES 模块提供 4 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 21-8 中列出了来自 AES 的 CPU 中断事件。

表 21-8. AES CPU 中断事件条件 (CPU_EVENT)

索引 (IIDX)	名称	说明
0	AESRDY	表示 AES 模块已经完成了选定的操作，并且结果可从 AESADOUT 中读取（如果适用）。
1	DMA0	不用于正常运行。
2	DMA1	不用于正常运行。
3	DMA2	不用于正常运行。

CPU 中断事件配置通过事件管理寄存器进行管理。有关为 CPU 中断配置这些寄存器的指导，请参阅节 7.2.5。

21.2.8.2 DMA 触发事件发布者 (DMA_TRIG0)

AES 模块提供四个触发源，这些触发源可以配置为 DMA 触发 0 的源。为了降低中断优先级，表 21-9 中列出了来自 AES 的 DMA0 触发事件。当 AES 需要 DMA0 通道进行分组密码操作时，应在 DMA_TRIG0 的 IMASK 寄存器中取消屏蔽 DMA0 触发，并根据需要配置 DMA 以支持 AES 操作。

表 21-9. AES DMA 触发 0 事件条件 (DMA_TRIG0)

索引 (IIDX)	名称	说明
0	AESRDY	不用于正常运行。
1	DMA0	DMA0 触发指示
2	DMA1	不用于正常运行。
3	DMA2	不用于正常运行。

通过 DMA_TRIG0 事件管理寄存器来管理 DMA 触发 0 事件配置。有关为 DMA 触发配置事件寄存器的指导，请参阅节 7.2.5。

21.2.8.3 DMA 触发事件发布者 (DMA_TRIG1)

AES 模块提供四个触发源，这些触发源可以配置为 DMA 触发 1 的源。为了降低中断优先级，表 21-10 中列出了来自 AES 的 DMA1 触发事件。当 AES 需要 DMA1 通道进行分组密码操作时，应在 DMA_TRIG1 的 IMASK 寄存器中取消屏蔽 DMA1 触发，并根据需要配置 DMA 以支持 AES 操作。

表 21-10. AES DMA 触发 1 事件条件 (DMA_TRIG1)

索引 (IIDX)	名称	说明
0	AESRDY	不用于正常运行。
1	DMA0	不用于正常运行。
2	DMA1	DMA1 触发指示
3	DMA2	不用于正常运行。

通过 DMA_TRIG1 事件管理寄存器来管理 DMA 触发 1 事件配置。有关为 DMA 触发配置事件寄存器的指导，请参阅节 7.2.5。

21.2.8.4 DMA 触发事件发布者 (DMA_TRIG2)

AES 模块提供四个触发源，这些触发源可以配置为 DMA 触发 2 的源。为了降低中断优先级，表 21-11 中列出了来自 AES 的 DMA2 触发事件。当 AES 需要 DMA2 通道进行分组密码操作时，应在 DMA_TRIG2 的 IMASK 寄存器中取消屏蔽 DMA2 触发，并根据需要配置 DMA 以支持 AES 操作。

表 21-11. AES DMA 触发 2 事件条件 (DMA_TRIG2)

索引 (IIDX)	名称	说明
0	AESRDY	不用于正常运行。
1	DMA0	不用于正常运行。
2	DMA1	不用于正常运行。
3	DMA2	DMA2 触发指示

通过 DMA_TRIG2 事件管理寄存器来管理 DMA 触发 2 事件配置。有关为 DMA 触发配置事件寄存器的指导，请参阅节 7.2.5。

21.3 AES 寄存器

表 21-12 列出了 AES 寄存器的存储器映射寄存器。表 21-12 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 21-12. AES 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引寄存器	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引寄存器	DMA_TRIG0	转到
1058h	IMASK	中断屏蔽	DMA_TRIG0	转到
1060h	RIS	原始中断状态	DMA_TRIG0	转到
1068h	MIS	已屏蔽中断状态	DMA_TRIG0	转到
1070h	ISET	中断设置	DMA_TRIG0	转到
1078h	ICLR	中断清除	DMA_TRIG0	转到
1080h	IIDX	中断索引寄存器	DMA_TRIG1	转到
1088h	IMASK	中断屏蔽	DMA_TRIG1	转到
1090h	RIS	原始中断状态	DMA_TRIG1	转到
1098h	MIS	已屏蔽中断状态	DMA_TRIG1	转到
10A0h	ISET	中断设置	DMA_TRIG1	转到
10A8h	ICLR	中断清除	DMA_TRIG1	转到
10B0h	IIDX	中断索引寄存器	DMA_TRIG2	转到
10B8h	IMASK	中断屏蔽	DMA_TRIG2	转到
10C0h	RIS	原始中断状态	DMA_TRIG2	转到
10C8h	MIS	已屏蔽中断状态	DMA_TRIG2	转到
10D0h	ISET	中断设置	DMA_TRIG2	转到
10D8h	ICLR	中断清除	DMA_TRIG2	转到
10E0h	EVT_MODE	事件模式		转到
1100h	AESACTL0	AES 加速控制寄存器 0		转到
1104h	AESACTL1	AES 加速控制寄存器 1		转到
1108h	AESASTAT	AES 加速器状态寄存器		转到
110Ch	AESAKEY	AES 加速器密钥寄存器		转到
1110h	AESADIN	AES 加速器数据输入寄存器		转到
1114h	AESADOUT	AES 加速器数据输出寄存器		转到
1118h	AESAXDIN	AES 加速器“异或”运算数据输入寄存器		转到
111Ch	AESAXIN	AES 加速器“异或”运算数据输入寄存器 (无触发)		转到

复杂的位访问类型经过编码可适应小型表单元。表 21-13 显示了适用于此部分中访问类型的代码。

表 21-13. AES 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

21.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 21-10 展示了 PWREN，表 21-14 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 21-10. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 21-14. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

21.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 21-11 展示了 RSTCTL，表 21-15 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 21-11. RSTCTL

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

表 21-15. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

21.3.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 21-12 展示了 STAT，表 21-16 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 21-12. STAT

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

表 21-16. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

21.3.4 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 21-13 展示了 PDBGCTL，表 21-17 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 21-13. PDBGCTL

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

表 21-17. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设将阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

21.3.5 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 21-14 展示了 IIDX，表 21-18 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 21-14. IIDX

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

表 21-18. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 1h = AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取

21.3.6 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 21-15 展示了 IMASK，表 21-19 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 21-15. IMASK

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

表 21-19. IMASK 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	AESRDY	R/W	0h	AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

21.3.7 RIS（偏移 = 1030h）[复位 = 00000000h]

图 21-16 展示了 RIS，表 21-20 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 21-16. RIS

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

表 21-20. RIS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	AESRDY	R	0h	AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取。 0h = 未发生中断 1h = 已发生中断

21.3.8 MIS (偏移 = 1038h) [复位 = 00000000h]

图 21-17 展示了 MIS，表 21-21 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 21-17. MIS

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

表 21-21. MIS 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	AESRDY	R	0h	AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取。 0h = 未发生中断 1h = 已发生中断

21.3.9 ISET（偏移 = 1040h）[复位 = 00000000h]

图 21-18 展示了 ISET，表 21-22 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 21-18. ISET

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

表 21-22. ISET 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	
0	AESRDY	W	0h	AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取。 0h = 写入 0 不产生影响 1h = 设置中断

21.3.10 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 21-19 展示了 ICLR，表 21-23 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 21-19. ICLR

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

表 21-23. ICLR 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	
0	AESRDY	W	0h	AES 就绪中断：表示所选的 AES 操作完成且结果可从 AESADOUT 中读取。 0h = 写入 0 不产生影响 1h = 清除中断

21.3.11 IIDX（偏移 = 1050h）[复位 = 00000000h]

图 21-20 展示了 IIDX，表 21-24 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 21-20. IIDX

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

表 21-24. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 2h = AES 触发 0 DMA

21.3.12 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 21-21 展示了 IMASK，表 21-25 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 21-21. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED						DMA0	RESERVED
R/W-0h						R/W-0h	R/W-0h

表 21-25. IMASK 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	DMA0	R/W	0h	DMA0 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RESERVED	R/W	0h	

21.3.13 RIS（偏移 = 1060h）[复位 = 00000000h]

图 21-22 展示了 RIS，表 21-26 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 21-22. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						DMA0	RESERVED
R-0h						R-0h	R-0h

表 21-26. RIS 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1	DMA0	R	0h	DMA0 事件 0h = 未发生中断 1h = 已发生中断
0	RESERVED	R	0h	

21.3.14 MIS (偏移 = 1068h) [复位 = 00000000h]

图 21-23 展示了 MIS，表 21-27 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 21-23. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						DMA0	RESERVED
R-0h						R-0h	R-0h

表 21-27. MIS 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1	DMA0	R	0h	DMA0 事件 0h = 未发生中断 1h = 已发生中断
0	RESERVED	R	0h	

21.3.15 ISET (偏移 = 1070h) [复位 = 00000000h]

图 21-24 展示了 ISET，表 21-28 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 21-24. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED						DMA0	保留
W-0h						W-0h	W-0h

表 21-28. ISET 字段说明

位	字段	类型	复位	说明
31-2	保留	W	0h	
1	DMA0	W	0h	DMA0 0h = 写入 0 无效 1h = 设置中断
0	RESERVED	W	0h	

21.3.16 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 21-25 展示了 ICLR，表 21-29 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 21-25. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED						DMA0	保留
W-0h						W-0h	W-0h

表 21-29. ICLR 字段说明

位	字段	类型	复位	说明
31-2	保留	W	0h	
1	DMA0	W	0h	DMA0 事件 0h = 写入 0 无效 1h = 清除中断
0	RESERVED	W	0h	

21.3.17 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 21-26 展示了 IIDX，表 21-30 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 21-26. IIDX

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

表 21-30. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 3h = AES 触发 1 DMA

21.3.18 IMASK（偏移 = 1088h）[复位 = 00000000h]

图 21-27 展示了 IMASK，表 21-31 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 21-27. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED					DMA1	RESERVED	
R/W-0h					R/W-0h	R/W-0h	

表 21-31. IMASK 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2	DMA1	R/W	0h	DMA1 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1-0	保留	R/W	0h	

21.3.19 RIS（偏移 = 1090h）[复位 = 00000000h]

图 21-28 展示了 RIS，表 21-32 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 21-28. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					DMA1	RESERVED	
R-0h					R-0h	R-0h	

表 21-32. RIS 字段说明

位	字段	类型	复位	说明
31-3	保留	R	0h	
2	DMA1	R	0h	DMA1 事件 0h = 未发生中断 1h = 已发生中断
1-0	RESERVED	R	0h	

21.3.20 MIS (偏移 = 1098h) [复位 = 00000000h]

图 21-29 展示了 MIS，表 21-33 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 21-29. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					DMA1	RESERVED	
R-0h					R-0h	R-0h	

表 21-33. MIS 字段说明

位	字段	类型	复位	说明
31-3	保留	R	0h	
2	DMA1	R	0h	DMA1 事件 0h = 未发生中断 1h = 已发生中断
1-0	RESERVED	R	0h	

21.3.21 ISET (偏移 = 10A0h) [复位 = 00000000h]

图 21-30 展示了 ISET，表 21-34 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 21-30. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED					DMA1	保留	
W-0h					W-0h	W-0h	

表 21-34. ISET 字段说明

位	字段	类型	复位	说明
31-3	保留	W	0h	
2	DMA1	W	0h	DMA1 事件 0h = 写入 0 无效 1h = 设置中断
1-0	RESERVED	W	0h	

21.3.22 ICLR (偏移 = 10A8h) [复位 = 00000000h]

图 21-31 展示了 ICLR，表 21-35 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 21-31. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED					DMA1	保留	
W-0h					W-0h	W-0h	

表 21-35. ICLR 字段说明

位	字段	类型	复位	说明
31-3	保留	W	0h	
2	DMA1	W	0h	DMA1 事件 0h = 写入 0 无效 1h = 清除中断
1-0	RESERVED	W	0h	

21.3.23 IIDX（偏移 = 10B0h）[复位 = 00000000h]

图 21-32 展示了 IIDX，表 21-36 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 21-32. IIDX

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

表 21-36. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 4h = AES 触发 2 DMA

21.3.24 IMASK（偏移 = 10B8h）[复位 = 00000000h]

图 21-33 展示了 IMASK，表 21-37 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 21-33. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA2	RESERVED		
R/W-0h				R/W-0h	R/W-0h		

表 21-37. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	DMA2	R/W	0h	DMA2 事件屏蔽。 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2-0	RESERVED	R/W	0h	

21.3.25 RIS（偏移 = 10C0h）[复位 = 00000000h]

图 21-34 展示了 RIS，表 21-38 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 21-34. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA2	RESERVED		
R-0h				R-0h	R-0h		

表 21-38. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	DMA2	R	0h	DMA2 事件 0h = 未发生中断 1h = 已发生中断
2-0	RESERVED	R	0h	

21.3.26 MIS (偏移 = 10C8h) [复位 = 00000000h]

图 21-35 展示了 MIS，表 21-39 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 21-35. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA2	RESERVED		
R-0h				R-0h	R-0h		

表 21-39. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	DMA2	R	0h	DMA2 事件 0h = 未发生中断 1h = 已发生中断
2-0	RESERVED	R	0h	

21.3.27 ISET (偏移 = 10D0h) [复位 = 00000000h]

图 21-36 展示了 ISET，表 21-40 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 21-36. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA2	保留		
W-0h				W-0h	W-0h		

表 21-40. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	DMA2	W	0h	DMA2 事件 0h = 写入 0 无效 1h = 设置中断
2-0	保留	W	0h	

21.3.28 ICLR (偏移 = 10D8h) [复位 = 00000000h]

图 21-37 展示了 ICLR，表 21-41 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 21-37. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED				DMA2	保留		
W-0h				W-0h	W-0h		

表 21-41. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	DMA2	W	0h	DMA2 事件 0h = 写入 0 无效 1h = 清除中断
2-0	保留	W	0h	

21.3.29 EVT_MODE（偏移 = 10E0h）[复位 = 000000A9h]

图 21-38 展示了 EVT_MODE，表 21-42 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 21-38. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
EVT3_CFG		EVT2_CFG		EVT1_CFG		INT0_CFG	
R-2h		R-2h		R-2h		R-1h	

表 21-42. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-6	EVT3_CFG	R	2h	none.DMA_TRIG2 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
5-4	EVT2_CFG	R	2h	none.DMA_TRIG1 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	EVT1_CFG	R	2h	none.DMA_TRIG0 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	INT0_CFG	R	1h	none.CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

21.3.30 AESACTL0（偏移 = 1100h）[复位 = 00000000h]

在图 21-39 中展示出了 AESACTL0 并在表 21-43 中对其进行了说明。

返回到汇总表。

AES 加速控制寄存器 0

图 21-39. AESACTL0

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
CMEN	RESERVED			ERRFG	RESERVED		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
SWRST	CMx		RESERVED	KLx		OPx	
R/W-0h	R/W-0h		R/W-0h	R/W-0h		R/W-0h	

表 21-43. AESACTL0 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	CMEN	R/W	0h	AESCMEN 和 DMA 一起支持 ECB、CBC、OFB 和 CFB 加密模式。当 AESCMEN = 1 和 AESBLKCNTx > 0 时，写入将被忽略。0 = 不生成 DMA 触发。1 = 使能 DMA 加密模式支持运行，并生成相应的 DMA 触发。 0h = 不生成 DMA 触发。 1h = 使能 DMA 加密模式支持运行，并生成相应的 DMA 触发。
14-12	RESERVED	R/W	0h	
11	ERRFG	R/W	0h	AES 错误标志。在一个 AES 正常运行期间，AESAKEY 或 AESADIN 将被写入。该位必须由软件清除。0b = 无错；1b = 出错 0h (R/W) = 无错 1h (R/W) = 出错
10-8	RESERVED	R/W	0h	
7	SWRST	R/W	0h	AES 软件复位。除了 AESRDYIE，AESKLx 和 AESOPx 位，即使忙线，也要立即复位整个 AES 加速器模块。它还会清零（内部）状态存储器。AESSWRST 位被自动复位并始终读取为 0。0b = 无复位；1b = 复位 AES 加速器模块 0h = 无复位。 1h = 复位 AES 加速器模块。
6-5	CMx	R/W	0h	AES 的密码方式选择。当 AESCMEN 设置为 0 时，忽略这些位。当 AESCMEN = 1 和 AESBLKCNTx > 0 时，写入将被忽略。00b = ECB；01b = CBC；10b = OFB；11b = CFB 0h = ECB 1h = CBC 2h = OFB 3h = CFB
4	保留	R/W	0h	

表 21-43. AESACTL0 字段说明 (continued)

位	字段	类型	复位	说明
3-2	KLx	R/W	0h	AES 密钥长度。 这些位定义了执行的 1 AES 标准。不是通过令 AESSWRST = 1 来复位 AESKLx 位的。 当 AESCMEN 设置为 1 且 AESBLKCNTx 大于 0 时，写操作会被忽略。 0h = 密钥大小为 128 位。 2h = 密钥大小为 256 位。
1-0	OPx	R/W	0h	AES 运行。不是通过令 AESSWRST = 1 来复位 AESOPx 位的。 当 AESCMEN = 1 和 AESBLKCNTx > 0 时，写入将被忽略。00b = 加密。01b = 解密。提供的密钥与用于加密的密钥是相同的。10b = 生成解密所需的首轮密钥。11b = 解密。提供的密钥解密所需的首轮密钥。 0h (R/W) = 加密 1h (R/W) = 解密。提供的密钥与用于加密的密钥是相同的。 2h (R/W) = 生成解密所需的首轮密钥。 3h (R/W) = 解密。提供的密钥解密所需的首轮密钥。

21.3.31 AESACTL1（偏移 = 1104h）[复位 = 00000000h]

在图 21-40 中展示出了 AESACTL1 并在表 21-44 中对其进行了说明。

返回到汇总表。

AES 加速控制寄存器 1

图 21-40. AESACTL1

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

表 21-44. AESACTL1 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	BLKCNTx	R/W	0h	密码块计数器。启用分组密码模式时，需要加密或解密的块数 (AESCMEN = 1)。如果 AESCMEN 设置为 0，则忽略。随着加密或解密的完成，块计数器会递减。当 AESCMEN = 1 和 AESBLKCNTx > 0 时，写入将被忽略。 00h = 最小值 FFh = 尽可能高的值

21.3.32 AESASTAT（偏移 = 1108h）[复位 = 00000000h]

在图 21-41 中展示出了 AESASTAT 并在表 21-45 中对其进行了说明。

返回到汇总表。

AES 加速器状态寄存器。

图 21-41. AESASTAT

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
DOUTCNTx				DINCNTx			
R-0h				R-0h			
7	6	5	4	3	2	1	0
KEYCNTx				DOUTRD	DINWR	KEYWR	BUSY
R-0h				R-0h	R/W-0h	R/W-0h	R-0h

表 21-45. AESASTAT 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R	0h	
15-12	DOUTCNTx	R	0h	从 AESADOUT 读取的字节。AESDOUTRD 被复位时复位。如果 AESDOUTCNTx = 0 且 AESDOUTRD = 0，则就无字节被读取。如果 AESDOUTCNTx = 0 且 AESDOUTRD 的 = 1，则所有字节均被读取。 0h = 尽可能小的值 Fh = 尽可能高的值
11-8	DINCNTx	R	0h	写入 AESADIN、AESAXDIN 或 AESAXIN 的字节。AESDINWR 被复位时复位。如果 AESDINCNTx = 0 且 AESDINWR = 0，则无字节被写入。如果 AESDINCNTx = 0 且 AESDINWR = 1，所有字节均被写入。 0h = 尽可能小的值 Fh = 尽可能高的值
7-4	KEYCNTx	R	0h	AESKLx 设置为 00 时写入 AESAKEY 的字节，以及 AESKLx 设置为 b10 时写入 AESAKEY 的半字。AESKEYWR 被复位时复位。如果 AESKEYCNTx = 0 且 AESKEYWR = 0，无字节被写入。如果 AESKEYCNTx = 0 且 AESKEYWR = 1，所有字节均被写入。 0h = 尽可能小的值 Fh = 尽可能高的值
3	DOUTRD	R	0h	从 AESADOUT 中读取的所有 16 字节数据。当 AES 加速器忙线和输出数据被再次读取，AESSWRST，一个错误状态，更改 AESOPx 和 AESKLx 时，AESDOUTRD 被 PUC 复位。 0 = 未读取所有字节；1 = 已读取所有字节 0h = 未读取所有字节 1h = 已读取所有字节

表 21-45. AESASTAT 字段说明 (continued)

位	字段	类型	复位	说明
2	DINWR	R/W	0h	所有的 16 字节数据都被写入 AESADIN，AESAXDIN 或 AESAXIN。由软件改变该状态后，也会复位 AESDINCNTx 位。当 AES 加速器忙线，AESSWRST，一个错误状态更改 AESOPx 和 AESKLx，开始（覆写）写入数据时，AESDOUTRD 被 PUC 复位。由于 AESOPx 或 AESKLx 被更改时，它被复位，所以它可以再次被软件置位来指示当前的数据仍然有效。0 = 未写入所有字节；1 = 已写入所有字节 0h = 未写入所有字节 1h = 已写入所有字节
1	KEYWR	R/W	0h	写入 AESAKEY 的所有字节。如果 AESCMEN = 1，该位可以有软件修改但一定不能被软件复位 (1→0)。由软件改变该状态后，也会复位 AESKEYCNTx 位。AESSWRST，一个错误状态更改 AESOPx 和 AESKLx，开始（覆写）写入新密钥时，AESDOUTRD 会被 PUC 复位。当 AESOPx 发生变化时，该密钥会被复位。所以，可以通过软件再次设置该密钥，表示加载的密钥仍然有效。 0h = 未写入所有字节 1h = 写入所有字节
0	忙	R	0h	AES 加速器模块忙线；加密，解密，密钥生成。0 = 不忙；1 = 忙 0h = 不忙 1h = 忙

21.3.33 AESAKEY（偏移 = 110Ch）[复位 = 00000000h]

AESAKEY 在图 21-42 中展示出并在表 21-46 中对其进行了说明。

返回到汇总表。

AES 加速器密钥寄存器

图 21-42. AESAKEY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY3x								KEY2x								KEY1x								KEY0x							
W-0h								W-0h								W-0h								W-0h							

表 21-46. AESAKEY 字段说明

位	字段	类型	复位	说明
31-24	KEY3x	W	0h	当 AESAKEY 被作为字写入时，AES 密钥 n+3 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。密钥由软件或通过设置 AESSWRST = 1 复位。 00h = 尽可能小的值 FFh = 尽可能高的值
23-16	KEY2x	W	0h	当 AESAKEY 被作为字写入时，AES 密钥 n+2 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。密钥由软件或通过设置 AESSWRST = 1 复位。 00h = 尽可能小的值 FFh = 尽可能高的值
15-8	KEY1x	W	0h	当 AESAKEY 被作为字写入时，AES 密钥 n+1 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。密钥由软件或通过设置 AESSWRST = 1 复位。 00h = 尽可能小的值 FFh = 尽可能高的值
7-0	KEY0x	W	0h	当 AESAKEY 被作为字写入时，AES 密钥为 n 字节。当 AESAKEY 被作为字节写入时，AES 下一个密钥为字节。不能将字访问和字节访问混淆。一直读取为。密钥由软件或通过设置 AESSWRST = 1 复位。 00h = 尽可能小的值 FFh = 尽可能高的值

21.3.34 AESADIN（偏移 = 1110h）[复位 = 00000000h]

AESADIN 在图 21-43 中展示出并在表 21-47 中对其进行了说明。

返回到汇总表。

AES 加速器数据输入寄存器

图 21-43. AESADIN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN3x								DIN2x								DIN1x								DIN0x							
W-0h								W-0h								W-0h								W-0h							

表 21-47. AESADIN 字段说明

位	字段	类型	复位	说明
31-24	DIN3x	W	0h	当 AESADIN 作为字写入时，AES 数据低于 n+3 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
23-16	DIN2x	W	0h	当 AESADIN 作为字写入时，AES 数据低于 n+2 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
15-8	DIN1x	W	0h	当 AESADIN 作为字写入时，AES 数据低于 n+1 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
7-0	DIN0x	W	0h	当 AESADIN 被作为字写入时，AES 数据低于 n 字节。当 AESADIN 被作为字节写入时，AES 下一个数据低于字节。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值

21.3.35 AESADOUT（偏移 = 1114h）[复位 = 00000000h]

AESADOUT 在图 21-44 中展示出并在表 21-48 中对其进行了说明。

返回到汇总表。

AES 加速器数据输出寄存器

图 21-44. AESADOUT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT3x								DOUT2x								DOUT1x								DOUT0x							
R-0h								R-0h								R-0h								R-0h							

表 21-48. AESADOUT 字段说明

位	字段	类型	复位	说明
31-24	DOUT3x	R	0h	当 AESADOUT 作为字被读取时，AES 数据高于 n+3 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。 00h = 尽可能小的值 FFh = 尽可能高的值
23-16	DOUT2x	R	0h	当 AESADOUT 作为字被读取时，AES 数据高于 n+2 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。 00h = 尽可能小的值 FFh = 尽可能高的值
15-8	DOUT1x	R	0h	当 AESADOUT 作为字被读取时，AES 数据高于 n+1 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。 00h = 尽可能小的值 FFh = 尽可能高的值
7-0	DOUT0x	R	0h	当 AESADOUT 被作为字读取时，AES 数据高于 n 字节。当 AESADOUT 被作为字节读取时，AES 下一个数据输出字节。不能将字访问和字节访问混淆。 00h = 尽可能小的值 FFh = 尽可能高的值

21.3.36 AESAXDIN（偏移 = 1118h）[复位 = 00000000h]

AESAXDIN 在图 21-45 中展示出并在表 21-49 中对其进行了说明。

返回到汇总表。

AES 加速器“异或”运算数据输入寄存器

图 21-45. AESAXDIN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDIN3x								XDIN2x								XDIN1x								XDIN0x							
W-0h								W-0h								W-0h								W-0h							

表 21-49. AESAXDIN 字段说明

位	字段	类型	复位	说明
31-24	XDIN3x	W	0h	当 AESAXDIN 作为字写入时，AES 数据低于 n+3 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
23-16	XDIN2x	W	0h	当 AESAXDIN 作为字写入时，AES 数据低于 n+2 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
15-8	XDIN1x	W	0h	当 AESAXDIN 作为字写入时，AES 数据低于 n+1 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
7-0	XDIN0x	W	0h	当 AESAXDIN 被作为字写入时，AES 数据低于 n 字节。当 AESAXDIN 被作为字节写入时，AES 下一个数据输入字节。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值

21.3.37 AESAXIN（偏移 = 111Ch）[复位 = 00000000h]

AESAXIN 在图 21-46 中展示出并在表 21-50 中对其进行了说明。

返回到汇总表。

AES 加速器“异或”运算数据输入寄存器（无触发）

图 21-46. AESAXIN

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XIN3x								XIN2x								XIN1x								XIN0x							
W-0h								W-0h								W-0h								W-0h							

表 21-50. AESAXIN 字段说明

位	字段	类型	复位	说明
31-24	XIN3x	W	0h	当 AESAXIN 作为字写入时，AES 数据低于 n+3 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
23-16	XIN2x	W	0h	当 AESAXIN 作为字写入时，AES 数据低于 n+2 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
15-8	XIN1x	W	0h	当 AESAXIN 作为字写入时，AES 数据低于 n+1 字节。这些位不能用于字节访问。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值
7-0	XIN0x	W	0h	当 AESAXIN 被作为字写入时，AES 数据低于 n 字节。当 AESAXIN 被作为字节写入时，AES 下一个数据输入字节。不能将字访问和字节访问混淆。一直读取为。 00h = 尽可能小的值 FFh = 尽可能高的值

真随机数生成器 (TRNG) 块是一个可用于生成随机位序列的熵源。

22.1 TRNG 概述

真随机数生成器 (TRNG) 模块通过内部模拟熵源和数字调节/抽取块安全地生成随机 32 位数字。TRNG 可用作熵源，在实现确定性随机比特生成器 (DRBG) 时派生真随机种子值。TRNG 适用于构建确定性随机比特生成器 (DRBG) 系统，该系统可以通过用于加密随机数生成器的 NIST SP800-22 统计测试套件。

TRNG 模块的主要特性包括：

- 32 位真随机数输出
- 可在整个器件电源电压范围（电压和温度）内正常工作
- 基于 ΔΣ 调制器的模拟熵源
- 调节块
- 可配置抽取块
- 集成启动和持续运行状况测试，符合 NIST SP800-22 标准
- 专用内部 LDO 稳压器，用于抵御电源操控攻击

22.2 TRNG 运行

TRNG 包含两个主要组件：模拟块（包含 LDO 稳压器和熵源）和数字块（包含状态机、启动逻辑、调节、抽取、运行状况测试和寄存器接口）。图 22-1 展示了 TRNG 块。

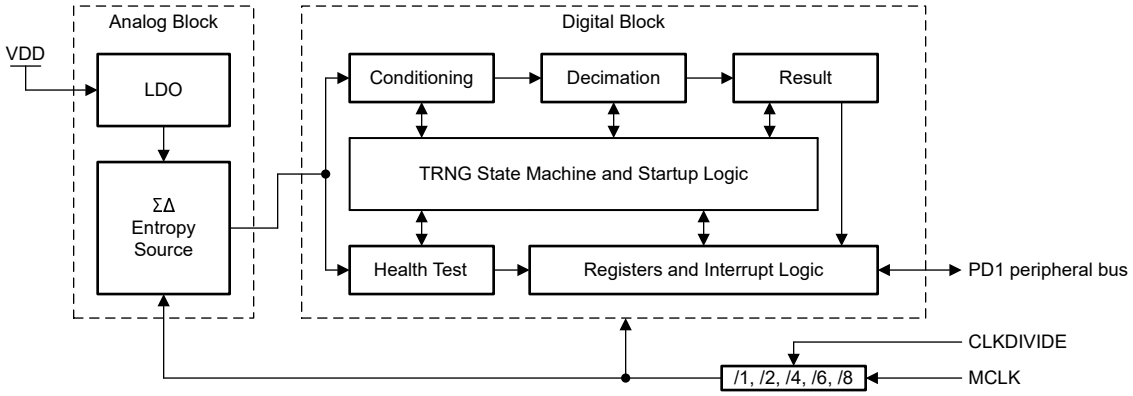


图 22-1. TRNG 方框图

22.2.1 TRNG 生成数据路径

随机数生成数据路径从模拟块开始。模拟块包含一个专用的低压降稳压器，用于向熵源供电，从而提高了 MCU 电源对电源操控攻击的抵御能力。Σ-Δ 熵源本身通过约翰逊-奈奎斯特噪音的 Δ-Σ 调制来衍生熵。

在操作期间，熵源输出被数字化，并且数字输出被发送到数字调节块以生成随机比特流。条件块实现流密码方案。在调节块之后，提供抽取块以通过累积可配置数量的样本的熵来提高熵。抽取块的输出最终在结果字保持寄存器中被捕获，以供应用软件读取。

备注

抽取块的目的是通过合并来自多个样本的熵来增加 TRNG 的总熵。抽取块将累积来自调节块的位，并且每 *n* 个样本抽取 1 个样本，其中 *n* 是抽取值。抽取由 *n* 个条件位的逐位异或函数实现。抽取整数 *n* 可设置为介于 1 和 8 之间。

22.2.2 时钟配置和输出速率

TRNG 功能时钟源自 MCLK。TRNG 需要一个在指定频率范围内的功能时钟才能正常运行。请参阅器件特定的数据表，了解给定器件上允许的 TRNG 频率范围。TRNG 中包含一个时钟分频器，用于导出供 TRNG 使用的频率。通过 TRNG 中 CLKDIVIDE 寄存器的 RATIO 字段指定时钟分频器。必须在启用 TRNG 之后（通过 PWREN 寄存器）且在 TRNG 状态离开 OFF 状态之前设置此字段。以超出器件数据表中指定范围的频率运行 TRNG 可能会导致意外行为。

TRNG 模块的输出速率取决于 TRNG 功能时钟频率和选定的抽取率。当抽取率设置为 DECIM_RATE = 0x0（1 倍抽取或无抽取）时，TRNG 功能时钟至少需要 32 个周期来捕获 32 个随机位。当抽取率设置为 DECIM_RATE = 0x3（4 倍抽取率）时，需要 128 个周期来捕获 32 个随机位。在抽取率设置为 4 倍抽取的 10MHz 时钟速率下，捕获 32 个随机数据位所需的时间为 12.8 µs。方程式 27 给出了根据 TRNG 功能时钟频率和所选抽取率计算生成 32 个随机位所需时间的公式。

$$t_{GENERATE}=(32*(DECIM\_RATE+1) / f_{TRNG} \quad (27)$$

备注

为了获得通过 NIST SP800-22 统计测试套件所需的最小熵，需要 4 倍或更大的抽取率 (DECIM_RATE=0x3)。TI 建议在加密应用中使用 TRNG 时使用 4 倍或更大的抽取率。

备注

更改 DECIM_RATE 字段时，必须将 NORM_FUNC 命令重新发送到 TRNG 才能使新速率生效（通过将 0x3 写入 CTL 寄存器中的 CMD 字段）。

当捕获的 32 位随机数据在 DATA_CAPTURE 寄存器中可用时，TRNG 使 IRQ_CAPTURED_RDY 中断功能生效，以便向处理器指示数据已准备就绪。一旦处理器读取数据，就开始捕获接下来的 32 位随机数据。

22.2.3 低功耗模式下的行为

TRNG 在 RUN 和 SLEEP 模式下可用。在任何其他工作模式下均不可用，并且当器件转换到低于 SLEEP 的任何工作模式时，所有 TRNG 配置设置和数据都会丢失。使用低于 SLEEP 的低功耗工作模式时，必须重新配置 TRNG，以便在将器件从 STOP、STANDBY 或 SHUTDOWN 模式唤醒时根据需要使用。

22.2.4 健康检测

提供了三种健康检测机制：数字块加电自检、模拟块加电自检和运行时自检。当 TRNG 处于 NORM_FUNC 状态时，通过向 TRNG 状态机发送相应的 CMD 来执行数字和模拟上电自检。当 TRNG 处于 NORM_FUNC 状态时，始终执行运行时自检。

22.2.4.1 数字块启动自检

TRNG 模块上电时，应用软件会运行数字启动运行状况测试。该内置自检通过在完整的数字块中运行预定义的数字样本序列，并检查预期输出，来验证数字块的正常运行。测试序列包括八个测试。每个测试需要 1024 个 TRNG 时钟周期才能完成。因为，每次测试都会将 1024 个样本输入到数字块。

会在 TEST_RESULTS 寄存器中的 DIG_TEST 位中报告数字启动自检的结果。每个测试都以独立的结果位报告其状态。每个测试通过时，硬件会设置 DIG_TEST 中相应的位。数字启动自检完成后，如果 TEST_RESULTS 寄存器的 DIG_TEST 字段的所有八个位都被设置，则表示通过。如果 DIG_TEST 字段中的任意位没有设置，则表示测试失败。如果任何数字启动运行状况测试失败，TRNG 将不可用。

备注

执行时，数字自检会改变抽取率。这是自检功能的一部分。

备注

数字自检作为测试序列的一部分，向数据路径注入确定性值。在完成数字块启动自检后，在正常 (NORM_FUNC) 工作模式下，从 DATA_CAPTURE 寄存器中读取的第一个数据是来自数字自检的确定性值，而不是真正的随机数。**运行数字启动自检后，始终丢弃第一个 DATA_CAPTURE 值。**

22.2.4.2 模拟块启动自检

TRNG 模块上电时，应用软件会运行模拟启动运行状况测试。此测试通过捕获 4,096 个连续模拟样本，并验证样本是否通过运行状况测试来验证模拟块是否正常运行。

如果运行状况测试失败，则运行状况失败 (IRQ_HEALTH_FAIL) 和命令完成 (IRQ_CMD_DONE) 中断均置为有效，并且 TRNG 状态机进入 ERROR 状态。还会在 TEST_RESULTS 寄存器中的 ANA_TEST 位中报告模拟启动自检的结果。如果自检通过，则设置 ANA_TEST 位。如果自检失败，则 ANA_TEST 位清零。

如果自检失败，则表明在自检过程中检测到模拟块中存在严重的熵损失。这意味着熵源不能在其当前状态下使用。该自检返回误报的概率（报告发生故障但模拟块正常运行的概率）在统计上非常低，但不是零。这意味着即使 TRNG 模拟块正常运行，在某些情况下也有很小的可能性指示故障。为了确保测试失败是合法的，并且没有熵损失，可以重复模拟启动运行状况测试来验证失败。如果出现故障，建议执行以下步骤：

1. 清除 IRQ_HEALTH_FAIL 中断
2. 关闭 TRNG 电源
3. 再次运行模拟启动运行状况测试
4. 根据测试结果执行以下相应操作：
   a. 如果测试通过，则可以使用 TRNG
   b. 如果第二次测试失败，请转到第 1 步并尝试第三次运行测试
   c. 如果第三次测试失败，则会出现灾难性的熵损失，不应使用 TRNG

22.2.4.3 运行时健康检测

在 TRNG 正常运行期间，通过健康检测逻辑发送熵源的原始数据，以确保持续存在足够的熵。这些测试旨在以统计方式检查每个样本是否至少有 0.3 位熵。TRNG 可执行重复计数和自适应比例连续自检。

22.2.4.3.1 重复计数测试

重复计数测试可快速检测导致熵源长时间保持在单个输出值的故障。如果熵源为 135 个连续样本输出相同的位值，重复计数测试将失败。

22.2.4.3.2 自适应比例测试

可在 1024 个样本的窗口中进行自适应比例测试，检测导致样本数量不成比例的故障为同一位值和/或位模式。如果违反表 22-1 中的任意条件，则自适应比例测试失败。例如，如果 1024 个样本的窗口中超过 912 个样本的位值相同，则测试失败。

表 22-1. 自适应比例测试范围

位模式	位模式出现次数限制
1	$112 < n < 912$
10	$211 < n < 341$
001	$97 < n < 192$
1011	$11 < n < 104$

表 22-1 中用于自适应比例测试的位模式旨在有效检测 TRNG 中的熵灾难性损失，同时尽可能减少误报故障检测的发生。

22.2.4.3.3 处理运行时运行状况测试失败

如果运行时运行状况测试确定存在的熵不足，则会断言运行状况失败 (IRQ_HEALTH_FAIL) 中断，并且 TRNG 状态机将进入 ERROR 状态。由于 TRNG 模块的随机性，即使 TRNG 正常工作，在某些情况下也可能指示运行状况故障。要确定是否存在真正的熵损失，建议执行以下过程：

1. 清除 IRQ_HEALTH_FAIL 中断
2. 关闭 TRNG 电源
3. 再次将 TRNG 开机至正常模式
   a. 如果未再次断言运行状况故障，则可以使用 TRNG
   b. 如果运行状况测试第二次失败，请转到步骤 1 并尝试第三次运行测试
   c. 如果运行状况测试第三次失败，则会出现灾难性的熵损失，不应使用 TRNG

TRNG 会提供诊断信息以指示哪些测试失败。要确定哪些运行时测试失败，请分别检查 STAT 寄存器中重复测试和自适应比例测试的 REP_FAIL 位和 ADAP_FAIL 位。

22.2.5 配置

TRNG 模块由应用软件配置和使用，可通过 PD1 仅 CPU 外设总线进行访问。DMA 无法访问 TRNG。

TRNG 包含一个状态机，用于管理 TRNG 的当前工作模式。应用软件可以通过命令配置 TRNG 来更改当前状态。TRNG 还会向 CPU 发送中断来传达状态。

备注

在将 CMD 写入 TRNG 之前，写入 TRNG 配置寄存器会导致意外的命令失败中断请求 (IRQ_CMD_FAIL) 有效。在写入 TRNG 配置寄存器时，屏蔽 TRNG 命令失败中断，直到 CMD 写入 TRNG。在写入 CMD 之后以及启用 IRQ_CMD_FAIL 中断之前，清除 IRQ_CMD_FAIL 中断状态。

22.2.5.1 TRNG 状态机

TRNG 包含一个状态机，用于管理 TRNG 的当前工作状态。通过向 TRNG 发送命令以更改其状态，配置 TRNG 来进行使用。TRNG 具有七种状态，如表 22-2 所示。

表 22-2. TRNG 状态

代码	状态名称	说明
0x0	关闭	模拟块已断电，数字块已禁用
0x1	PWRUP_ES	TRNG 正在为熵源加电，并将转换为 NORM_FUNC。
0x2	PWRDOWN_ES	TRNG 正在为熵源断电，并将转换为 OFF。
0x3	NORM_FUNC	TRNG 正在正常运行并生成随机位。
0x7	TEST_DIG	TRNG 正在运行数字开机自检。
0xB	TEST_ANA	TRNG 正在运行模拟开机自检。
0xA	ERROR	操作因错误情况而停止。模拟器件保持通电状态，但运行状况和调节逻辑已停止。

当前 TRNG 状态可由应用软件确定。要检查当前状态，请读取 TRNG 的 STAT 寄存器中的 FSM_STATE 字段。

22.2.5.1.1 更改 TRNG 状态

应用软件可以通过向 CTL 寄存器中的 CMD 字段写入新的命令值来设置 TRNG 的工作状态。虽然 TRNG 状态机有 7 个状态，但这 7 个状态中只有 4 个是由用户使用 CMD (OFF、TEST_DIG、TEST_ANA 和 NORM_FUNC) 控制进入的。其他 3 种状态 (PWRUP_ES、PWRDOWN_ES 和 ERR) 仅在模拟块上下电或出现错误情况时由硬件进入。表 22-3 中给出了 TRNG 的运行状态更改命令。

表 22-3. TRNG 操作模式命令

CMD	模式	说明
0x0	PWROFF	TRNG 模拟块断电，TRNG 数字块被时钟门控。
0x1	TEST_DIG	执行 TRNG 数字开机自检，结果在 TEST_RESULTS 寄存器中报告。
0x2	TEST_ANA	执行 TRNG 模拟上电自检，结果在 TEST_RESULTS 寄存器中报告。
0x3	NORM_FUNC	TRNG 被置于正常工作模式，在该模式中，将收集、调节、抽取来自熵源的样本并放置在 DATA_CAPTURE 寄存器中。连续统计运行状况测试在此模式下运行。

从给定状态开始，只能进行某些状态更改。例如，从 OFF 仅支持切换到 NORM_FUNC 的 CMD。支持从 NORM_FUNC 切换到 TEST_DIG、TEST_ANA 和 OFF 的 CMD。仅支持从 ERR 状态切换至 OFF。图 22-2 中给出了 TRNG 的状态流程。

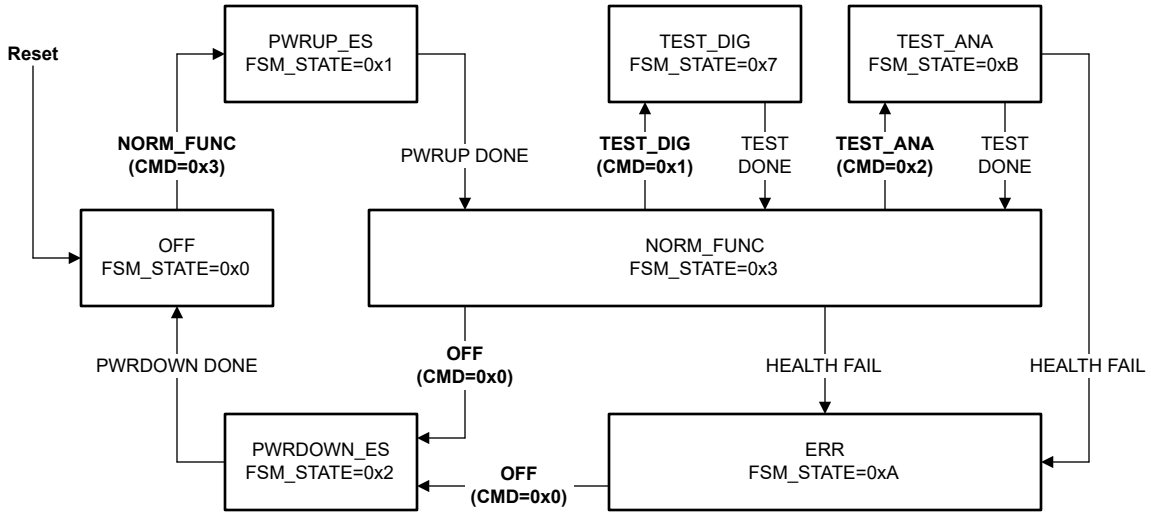


图 22-2. TRNG 状态流程

只有在之前发出的命令运行完成后，才能将新命令写入 CMD 字段。如果在无效时间写入命令，则该命令会被拒绝，并且命令失败中断将置为有效。

22.2.5.2 使用 TRNG

请按照以下过程启动 TRNG：

1. 通过在 TRNG PWREN 寄存器中设置 ENABLE 位以及 KEY 来启用 TRNG。
2. 配置 TRNG 时钟分频器，确保 TRNG 功能时钟处于允许的范围内（典型值为 10MHz，有关更多详细信息，请参阅器件数据表)。通过将所需值编程到 CLKDIVIDE 寄存器的 RATIO 字段，可配置时钟分频器。例如，如果 MCLK 为 80MHz，则 RATIO 字段应设置为 0x7（8 分频），以便为 TRNG 模块提供 10MHz 功能时钟。
3. 验证 TRNG 中断是否被禁用（将中断屏蔽位清零以屏蔽中断）。
4. 通过将 NORM_FUNC 命令 (0x3) 写入 TRNG 的 CTL 寄存器中的 CMD 字段，将 TRNG 从默认 OFF 状态移至 NORM_FUNC 状态。等待设置 IRQ_CMD_DONE 中断标志，表示 CMD 完成。
5. 运行数字块启动自检例程，确保 TRNG 数字块正常工作：
   a. 通过将 TEST_DIG 命令 (0x1) 写入 CTL 寄存器中的 CMD 字段，将 TRNG 从 NORM_FUNC 状态移到 TEST_DIG 状态。等待设置 IRQ_CMD_DONE 中断标志，表示数字自检已经完成。
   b. 通过确保设置 TEST_RESULTS 寄存器中的 DIG_TEST 字段 (DIG_TEST=0xFF)，检查所有 8 个数字测试已通过。
   c. 数字测试完成后，TRNG 将自动返回到 NORM_FUNC 状态。
6. 运行模拟块启动自检例程，确保 TRNG 模拟块正常工作：
   a. 通过将 TEST_ANA 命令 (0x2) 写入 CTL 寄存器中的 CMD 字段，将 TRNG 从 NORM_FUNC 状态移到 TEST_ANA 状态。等待设置 IRQ_CMD_DONE 中断标志，表示模拟块自检已经完成。
   b. 通过验证已设置 TEST_RESULTS 寄存器中的 ANA_TEST 位，检查模拟测试获得通过。
   c. 模拟测试完成后，如果测试通过，TRNG 将自动返回到 NORM_FUNC 状态。如果测试失败，TRNG 将进入 ERR 状态，必须先进入 OFF 状态，然后才能尝试再次使用它。
7. 运行启动自检后，将 TRNG 配置为正常运行：
   a. 通过设置相应的 ICLR 位来清除 IRQ_CAPTURED_RDY_IRQ 状态，这是因为在之前执行自检期间可能已经设置了此位。
   b. 通过将新的抽取率编程到 CTL 寄存器的 DECIM_RATE 字段中，然后再次发送 NORM_FUNC 命令（通过将 0x3 写入 CTL 寄存器中的 CMD 字段），将抽取率设置为所需值。建议抽取率为 4 (DECIM_RATE=0x3) 或更高。
   c. 可以通过设置 IMASK 寄存器中的 IRQ_HEALTH_FAIL 位来启用运行状况失败中断。
   d. 可以通过设置 IMASK 寄存器中的 IRQ_CAPTURED_RDY 位来启用数据捕获中断。

8. 等待第一个 IRQ_CAPTURED_RDY IRQ，并读取 DATA_CAPTURE 寄存器。该值（运行启动自检后从 DATA_CAPTURE 读取的第一个值）不是真随机值，必须在从 DATA_CAPTURE 寄存器收集真随机数据之前读取和丢弃该值。
9. 当 IRQ_CAPTURED_RDY IRQ 再次有效时，随机位可在 DATA_CAPTURE 寄存器中读出。
10. 如果 IRQ_HEALTH_FAIL IRQ 有效，则具备低熵条件，并且 TRNG 将自动切换到 ERR 状态以停止操作。要退出 ERR 状态，请清除 IRQ_HEALTH_FAIL 中断。然后，通过向 CTL 寄存器中的 CMD 字段发送 OFF 命令 (0x0)，将 TRNG 转换为 OFF 状态。等待设置 IRQ_CMD_DONE 中断标志，然后返回到步骤 2，再次为 TRNG 上电，测试是否有足够的熵可用。

22.2.5.3 TRNG 事件

TRNG 模块包含一个事件发布者，不包含事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理到 CPU 子系统的 TRNG 中断请求 (IRQ)。

表 22-4 列出了 TRNG 事件。

表 22-4. TRNG 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	TRNG	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 TRNG 到 CPU 的中断路由

22.2.5.3.1 CPU 中断事件发布者 (CPU_INT)

TRNG 模块提供 4 个中断源，这些中断源可配置为产生 CPU 中断事件。表 22-5 列出了 TRNG 的中断条件。

表 22-5. TRNG CPU 中断条件 (CPU_INT)

索引 (IIDX)	名称	说明
1	IRQ_HEALTH_FAIL	表示运行状况测试失败。
2	IRQ_CMD_FAIL	表示已发布 CMD 故障。
3	IRQ_CMD_DONE	表示已发布 CMD 已完成。
4	IRQ_CAPTURED_RDY	表示该处理器可以读取包含随机位的新的 32 位数据字。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置这些寄存器的指导，请参阅节 7.2.5。

22.3 TRNG 寄存器

表 22-6 列出了 TRNG 寄存器的存储器映射寄存器。表 22-6 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 22-6. TRNG 寄存器

偏移	缩写	寄存器名称	组	部分
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
10FCh	DESC	模块说明		转到
1100h	CTL	控制命令和抽取率		转到
1104h	STAT	通知运行状况测试结果和上次发出的命令的状态寄存器		转到
1108h	DATA_CAPTURE	RNG 数据的捕获字缓冲器		转到
110Ch	TEST_RESULTS	来自 TEST_ANA 和 TEST_DIG 的测试结果		转到
1110h	CLKDIVIDE	时钟分频器		转到

复杂的位访问类型经过编码可适应小型表单元。表 22-7 显示了适用于此部分中访问类型的代码。

表 22-7. TRNG 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

22.3.1 PWREN（偏移 = 800h）[复位 = 00000000h]

图 22-3 展示了 PWREN，表 22-8 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 22-3. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 22-8. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

22.3.2 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 22-4 展示了 RSTCTL，表 22-9 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 22-4. RSTCTL

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

表 22-9. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

22.3.3 STAT（偏移 = 814h）[复位 = 00000000h]

图 22-5 展示了 STAT，表 22-10 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 22-5. STAT

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

表 22-10. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

22.3.4 IIDX（偏移 = 1020h）[复位 = X]

图 22-6 展示了 IIDX，表 22-11 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。0h 表示无事件挂起。中断 1 是最高优先级，2 是次高优先级，4、8、⋯2^{31} 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0h。

图 22-6. IIDX

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

表 22-11. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	X	
7-0	STAT	R	0h	中断索引状态 0h = 没有设置任何位意味着没有挂起的中断请求 1h = 表示运行状况测试失败。TRNG 在中断被清除前处于错误状态。 2h = 表示刚刚发出的命令被拒绝而未在执行。 3h = 表示当前命令/模式已完成。根据模式，这可能具有不同的含义： OFF --> 电源已关闭 PWRUP_DIG --> 数字上电测试已完成 PWRUP_ANA --> 模拟上电测试已完成 NORM_FUNC --> 无 IRQ，因为模式会无限期运行，直到发出新命令 4h = 表示捕获的字缓冲区可随时复制到内存

22.3.5 IMASK（偏移 = 1028h）[复位 = X]

图 22-7 展示了 IMASK，表 22-12 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置某个位，则设置相应的中断屏蔽。

图 22-7. IMASK

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
保留							
R/W-X							
15	14	13	12	11	10	9	8
保留							
R/W-X							
7	6	5	4	3	2	1	0
RESERVED				IRQ_CAPTURED_RDY	IRQ_CMD_DONE	IRQ_CMD_FAIL	IRQ_HEALTH_FAIL
R/W-X				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 22-12. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	X	
3	IRQ_CAPTURED_RDY	R/W	0h	IRQ_CAPTURED_RDY 的屏蔽。向 CPU 指示捕获的字可随时读取。 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且将设置 MIS 中的相应位
2	IRQ_CMD_DONE	R/W	0h	IRQ_CMD_DONE 的屏蔽。指示命令已完成 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并将设置 MIS 中的相应位
1	IRQ_CMD_FAIL	R/W	0h	IRQ_CMD_FAIL 的屏蔽中断源。表示刚刚发出的命令/模式已被拒绝。 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且将设置 MIS 中的相应位
0	IRQ_HEALTH_FAIL	R/W	0h	IRQ_HEALTH_FAIL 的屏蔽。表示运行状况测试失败。 0h = 中断被屏蔽掉 1h = 中断将请求一个中断服务例程，并且将设置 MIS 中的相应位

22.3.6 RIS（偏移 = 1030h）[复位 = X]

图 22-8 展示了 RIS，表 22-13 中对此进行了介绍。

返回到汇总表。

原始中断状态反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 22-8. RIS

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留							
R-X							
7	6	5	4	3	2	1	0
RESERVED				IRQ_CAPTURED_RDY	IRQ_CMD_DONE	IRQ_CMD_FAIL	IRQ_HEALTH_FAIL
R-X				R-0h	R-0h	R-0h	R-0h

表 22-13. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	X	
3	IRQ_CAPTURED_RDY	R	0h	向 CPU 指示捕获的字可随时读取。读取 IIDX 将清除该中断。 0h = IRQ_CAPTURED_READY 未发生 1h = IRQ_CAPTURED_READY 已发生
2	IRQ_CMD_DONE	R	0h	IRQ_CMD_DONE 的原始中断源。表示发出的命令/模式已完成。 0h = IRQ_CMD_DONE 未发生 1h = IRQ_CMD_DONE 已发生
1	IRQ_CMD_FAIL	R	0h	IRQ_CMD_FAIL 的屏蔽中断源。表示刚刚发出的命令/模式已被拒绝。 0h = IRQ_CMD_FAIL 未发生 1h = IRQ_CMD_FAIL 已发生
0	IRQ_HEALTH_FAIL	R	0h	向 CPU 指示任何运行状况测试均已失败。读取 IIDX 将清除该中断。 0h = IRQ_CAPTURED_READY 未发生 1h = IRQ_CAPTURED_READY 已发生

22.3.7 MIS (偏移 = 1038h) [复位 = X]

图 22-9 展示了 MIS，表 22-14 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 22-9. MIS

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
保留							
R-X							
15	14	13	12	11	10	9	8
保留							
R-X							
7	6	5	4	3	2	1	0
RESERVED				IRQ_CAPTURED_RDY	IRQ_CMD_DONE	IRQ_CMD_FAIL	IRQ_HEALTH_FAIL
R-X				R-0h	R-0h	R-0h	R-0h

表 22-14. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	X	
3	IRQ_CAPTURED_RDY	R	0h	CAPTURED_READY 的屏蔽中断结果。向 CPU 指示捕获的字可随时读取。读取 IIDX 将清除该中断。 0h = IRQ_CAPTURED_READY 未请求中断服务例程 1h = IRQ_CAPTURED_READY 请求中断服务例程
2	IRQ_CMD_DONE	R	0h	IRQ_CMD_DONE 的屏蔽中断源。表示发出的命令/模式已完成。 0h = IRQ_CAPTURED_READY 未请求中断服务例程 1h = IRQ_CMD_DONE 请求中断服务例程
1	IRQ_CMD_FAIL	R	0h	IRQ_CMD_FAIL 的屏蔽中断源。表示刚刚发出的命令/模式已被拒绝。 0h = IRQ_CMD_FAIL 未请求中断服务例程 1h = IRQ_CMD_FAIL 请求中断服务例程
0	IRQ_HEALTH_FAIL	R	0h	HEALTH_FAIL 的屏蔽中断结果。向 CPU 指示最新 1024 位窗口的任何运行状况测试均失败。 0h = IRQ_CAPTURED_READY 未请求中断服务例程 1h = IRQ_CAPTURED_READY 请求中断服务例程

22.3.8 ISET（偏移 = 1040h）[复位 = 00000000h]

图 22-10 展示了 ISET，表 22-15 中对此进行了介绍。

返回到汇总表。

ISET 允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 22-10. ISET

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

表 22-15. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	IRQ_CAPTURED_RDY	W	0h	向 CPU 指示捕获的字可随时读取。读取 IIDX 或 DATA_CAPTURE 寄存器将清除此中断。 0h = 写入 0 不会产生影响 1h = 设置对应于 CAPTURED_READY 的 RIS 位
2	IRQ_CMD_DONE	W	0h	写入以开启 CMD_DONE IRQ。表示最后发出的 TRNG 命令已完成。 0h = 写入 0 不起作用。 1h = 设置与 CMD_DONE 对应的 RIS 位
1	IRQ_CMD_FAIL	W	0h	IRQ_CMD_FAIL 的屏蔽中断源。表示刚刚发出的命令/模式已被拒绝。 0h = 写入 0 不起作用。 1h = 设置与 CMD_FAIL 对应的 RIS 位
0	IRQ_HEALTH_FAIL	W	0h	向 CPU 指示任何运行状况测试均已失败。读取 IIDX 或 DATA_CAPTURE 寄存器将清除此中断。 0h = 写入 0 不会产生影响 1h = 设置对应于 HEALTH_FAIL 的 RIS 位

22.3.9 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 22-11 展示了 ICLR，表 22-16 中对此进行了介绍。

返回到汇总表。

写入 1 以清除相应的中断。

图 22-11. ICLR

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

表 22-16. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	IRQ_CAPTURED_RDY	W	0h	向 CPU 指示捕获的字可随时读取。读取 IIDX 或 DATA_CAPTURE 寄存器将清除此中断。 0h = 写入 0 不会产生影响 1h = 将与 CAPTURED_READY 对应的 RIS 位清零
2	IRQ_CMD_DONE	W	0h	写入以关闭 CMD_DONE IRQ。表示最后发出的 TRNG 命令已完成。 0h = 写入 0 不起作用。 1h = 将与 CMD_DONE 对应的 RIS 位清零
1	IRQ_CMD_FAIL	W	0h	IRQ_CMD_FAIL 的屏蔽中断源。表示刚刚发出的命令/模式已被拒绝。 0h = 写入 0 不起作用。 1h = 将与 CMD_FAIL 对应的 RIS 位清零
0	IRQ_HEALTH_FAIL	W	0h	向 CPU 指示任何运行状况测试均已失败。读取 IIDX 或 DATA_CAPTURE 寄存器将清除此中断。 0h = 写入 0 不会产生影响 1h = 将与 CAPTURED_READY 对应的 RIS 位清零

22.3.10 DESC（偏移 = 10FCh）[复位 = 05110000h]

图 22-12 展示了 DESC，表 22-17 中对此进行了介绍。

返回到汇总表。

该寄存器用于为需要在 MCLK 和另一个特殊时钟源之间进行选择的任何特殊模块指定时钟源选择。该寄存器预计不会出现在大多数标准 SVT 外设上，但可能出现在 ADC 等模块上。该寄存器不存在于 VDDCOREULP 域外设上并将保留，读数为 0。

图 22-12. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-511h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-0h				R-0h				R-0h				R-0h			

表 22-17. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	511h	模块标识符 - 创建了一个内部 TI 页面以请求唯一模块 ID
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数
7-4	MAJREV	R	0h	IP 的主要版本
3-0	MINREV	R	0h	IP 的次要版本

22.3.11 CTL（偏移 = 1100h）[复位 = X]

图 22-13 展示了 CTL，表 22-18 中对此进行了介绍。

返回到汇总表。

影响 TRNG 系统的各种参数

图 22-13. CTL

31	30	29	28	27	26	25	24
保留							
R/W-X							
23	22	21	20	19	18	17	16
RESERVED			PWRUP_PSTART_CFG		PWRUP_PCHRG_CFG		PWRUP_CLKDIV
R/W-X			R/W-2h		R/W-2h		R/W-0h
15	14	13	12	11	10	9	8
保留					DECIM_RATE		
R/W-X					R/W-0h		
7	6	5	4	3	2	1	0
RESERVED						CMD	
R/W-X						R/W-0h	

表 22-18. CTL 字段说明

位	字段	类型	复位	说明
31-21	保留	R/W	X	
20-19	PWRUP_PSTART_CFG	R/W	2h	配置脉冲启动序列长度 b00 = 已禁用 b01 = 在 10us 时上升，在 50us 时下降 b10 = 在 10us 时上升，在 70us 时下降（默认值） b11 = 在 10us 时上升，在 90us 时下降
18-17	PWRUP_PCHRG_CFG	R/W	2h	配置 PCHARGE 序列长度 b00 = 已禁用 b01 = 20us PCHARGE b10 = 30us PCHARGE（默认） b11 = 40us PCHARGE
16	PWRUP_CLKDIV	R/W	0h	当为“1”时，上电序列将需要两倍的时间（即时钟频率减半）
15-11	RESERVED	R/W	X	
10-8	DECIM_RATE	R/W	0h	设置抽取率。抽取因子为 n 0x0 = 抽取因子为 1（无抽取） 0x1 = 抽取因子为 2（每隔一个样片跳过） … 0x7 = 抽取因子为 8（每 8 个样片取样一次）
7-2	RESERVED	R/W	X	

表 22-18. CTL 字段说明 (continued)

位	字段	类型	复位	说明
1-0	CMD	R/W	0h	通过命令设置 TRNG 模式。直到完成前一条命令后，模式才会更新，如 IRQ_CMD_DONE 所示。 00 --> OFF 01 --> PWRUP_DIG 10 --> PWRUP_ANA 11 --> NORM_FUNC 0h = 关闭模拟源的电源并为数字接口计时 1h = 启动数字元件的上电测试序列。这将验证数字元件是否正常工作。IRQ_CMD_DONE 表示测试已完成。此测试的结果在 TEST_RESULTS 寄存器中的位 0:6 中 2h = 启动模拟 TRNG 的上电测试序列。这验证了模拟元件是否正在生成具有足够熵的序列。IRQ_CMD_DONE 表示测试已完成。此测试的结果位于 TEST_RESULTS 寄存器的位 7 中 3h = TRNG 的正常工作模式。所有元件均已开启。

22.3.12 STAT（偏移 = 1104h）[复位 = X]

图 22-14 展示了 STAT，表 22-19 中对此进行了介绍。

返回到汇总表。

通知运行状况测试结果、上次发出的命令和当前 FSM 状态的状态寄存器

图 22-14. STAT

31	30	29	28	27	26	25	24
保留							
R-X							
23	22	21	20	19	18	17	16
RESERVED				FSM_STATE			
R-X				R-0h			
15	14	13	12	11	10	9	8
保留						ISSUED_CMD	
R-X						R-0h	
7	6	5	4	3	2	1	0
RESERVED						REP_FAIL	ADAP_FAIL
R-X						R-0h	R-0h

表 22-19. STAT 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R	X	
19-16	FSM_STATE	R	0h	前端 FSM 的当前状态（在时钟域交叉之后）。 需要 2 次读取，因为读取其内容时可能存在亚稳态 状态： 0000：OFF 0001：PWRUP_ES 0011：NORM_FUNC 0111：TEST_DIG 1011：TEST_ANA 1010：ERROR 0010：PWRDOWN_ES
15-10	保留	R	X	
9-8	ISSUED_CMD	R	0h	指示向 TRNG 接口发出的最后一条接受的命令。 写入有效命令后，该寄存器将更新，并且该命令将继续执行，直至设置 CMD_DONE_IRQ。 CMD_DONE_IRQ 指示状态处于 PWROFF、NORM_FUNC 或 ERROR。这些状态将接受新命令。 00 --> OFF 01 --> PWRUP_DIG 10 --> PWRUP_ANA 11 --> NORM_FUNC
7-2	RESERVED	R	X	
1	REP_FAIL	R	0h	表示重复计数器测试导致了最近的故障。这样，运行状况计数数字很可能不适用于完整的 1024 位窗口。
0	ADAP_FAIL	R	0h	表示自适应比例测试（1、2、3 或 4 位计数器）失败，因为在最后的 1024 位窗口中计数的样片过多或过少。

22.3.13 DATA_CAPTURE（偏移 = 1108h）[复位 = 00000000h]

图 22-15 展示了 DATA_CAPTURE，表 22-20 中对此进行了介绍。

返回到汇总表。

从抽取块捕获的数据

图 22-15. DATA_CAPTURE

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

表 22-20. DATA_CAPTURE 字段说明

位	字段	类型	复位	说明
31-0	BUFFER	R	0h	从抽取块捕获的数据

22.3.14 TEST_RESULTS（偏移 = 110Ch）[复位 = X]

图 22-16 展示了 TEST_RESULTS，表 22-21 中对此进行了介绍。

返回到汇总表。

包括一个描述哪个启动测试失败的位。在数字启动验证检查中，前 8 位检查数字元件的状况，第 9 位检查熵源是否正在生成具有足够熵的样片。当命令未完成时，该寄存器将读取所有 '0s。每个测试均为低电平有效，“0”表示测试失败，“1”表示测试通过。

图 22-16. TEST_RESULTS

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

表 22-21. TEST_RESULTS 字段说明

位	字段	类型	复位	说明
31-9	保留	R	X	
8	ANA_TEST	R	0h	从启用的熵源运行完 4096 个样片，并验证所有运行状况测试均未失败，表示模拟元件生成了足够的熵
7-0	DIG_TEST	R	0h	位 0 表示第一次抽取率测试和运行状况测试（验证调节、抽取和捕获的缓冲区）是否失败，位 1 表示第二次抽取测试和运行状况测试是否失败 位 0 - decim_test0 (decim = 0x0) 位 1 - decim_test1 (decim = 0x1) ...

22.3.15 CLKDIVIDE (偏移 = 1110h) [复位 = 00000000h]

图 22-17 展示了 CLKDIVIDE，表 22-22 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块特定的分频比，它只支持 TRNG 的偶数分频。

图 22-17. CLKDIVIDE

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

表 22-22. CLKDIVIDE 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 3h = 对时钟源进行 4 分频 5h = 对时钟源进行 6 分频 7h = 对时钟源进行 8 分频

章节 23

计时器 (TIMx)

计时器模块 (TIMx) 是一个带有多个比较/捕获块的计时器计数模块。根据器件，提供两种类型的计时器：通用计时器 (TIMA) 和高级控制计时器 (TIMA)。这两个计时器包含很多可用于各种功能的常见特性，例如测量输入信号边沿和周期（捕获模式）或生成输出波形（比较模式输出），例如 PWM。请参阅器件特定数据表，以确定有哪些可用的计时器和计时器实例。

23.1 TIMx 概述

计时器模块 (TIMx) 是一个带有多个比较/捕获块的计时器计数模块。根据器件，提供两种类型的计时器：通用计时器 (TIMA) 和高级控制计时器 (TIMA)。这两个计时器都使用通用计时器架构，可在具有常见功能的计时器实例之间轻松迁移。这更大限度地减少了为基于计时器的应用编写额外软件的需求，并允许在 TIMx 实例之间轻松移植和维护。

备注

请参阅节 23.1.3，以确定 TIMA 和 TIMA 实例之间可用的常见功能。

在这一部分：

- “TIMx”表示在 TIMA 和 TIMA 上可用的一个常见功能。
- “TIMA”表示仅在 TIMA 上可用的一个功能。
- “TIMG”表示仅在 TIMG 上可用的一个功能。

23.1.1 TIMG 概述

TIMG 模块包含由可编程预分频器驱动的 16 位和 32 位自动重新加载计数器，以及两个用于多个捕获/比较、PWM 输出和间隔计时的捕获/比较 (CC) 块。TIMG 还具有广泛的事件生成功能，包括针对各种用例的计数器溢出、重新加载和捕获/比较操作。

23.1.1.1 TIMG 特性

TIMG 的具体特性包括：

- 具有重复重新加载模式的 16 位或 32 位向上、向下或向上/向下计数器
- 用于对计数器时钟频率进行分频的 8 位可编程预分频器
- 最多两个独立通道，用于：
  - 输出比较
  - 输入捕捉
  - PWM 输出（边沿对齐和中心对齐）
  - 单次触发模式
- 用于加载寄存器的影子寄存器模式
- 用于 CC 寄存器的影子比较模式
- 支持用于定位和移动检测的正交编码器接口 (QEI) 和 3 输入霍尔传感器模式
- 支持同一电源域中不同 TIMx 实例之间的同步和交叉触发
- 支持中断/DMA 触发生成以及跨外设（例如 ADC、DAC 等）触发功能

23.1.1.2 功能方框图

图 23-1 展示了 TIMG 方框图。

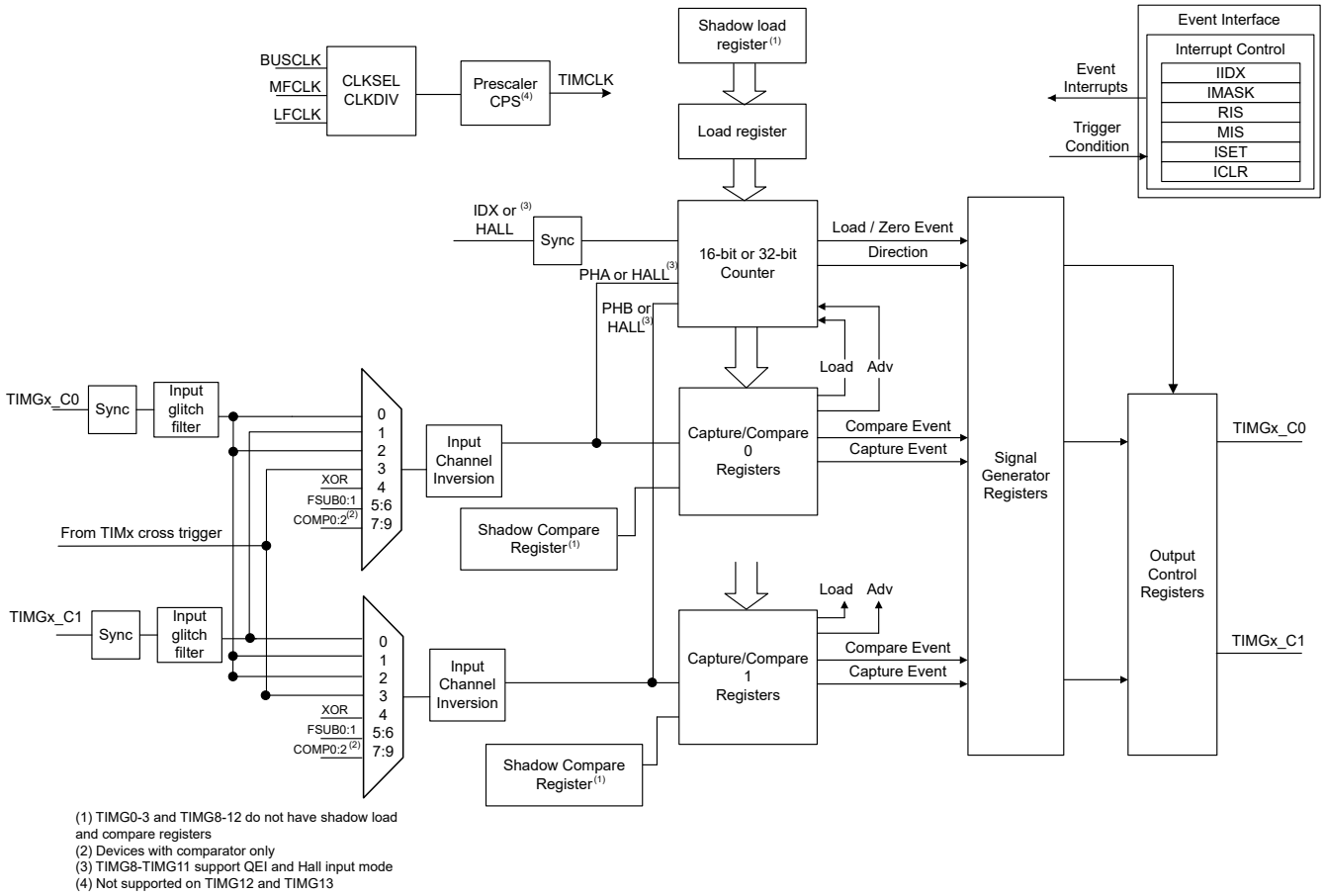


图 23-1. TIMG 功能方框图

23.1.2 TIMA 概述

TIMA 模块包含一个由可编程预分频器驱动的 16 位自动重新加载计数器，以及最多四个用于多次捕获/比较、具有死区插入的 PWM 输出和间隔计时的捕获/比较 (CC) 块。TIMA 具有广泛的事件生成功能，可生成不同的计数器事件（例如溢出事件、重新加载事件以及来自每个捕获/比较寄存器的事件）。它还具有处理由内部或外部电路生成的故障信号以指示系统中故障的硬件设计。

23.1.2.1 TIMA 特性

每个 TIMA 实例的具体特性包括：

- 具有重复重新加载模式的 16 位向上、向下或向上/向下计数器
- 可选和可配置的时钟源
- 用于对计数器时钟频率进行分频的 8 位可编程预分频器
- 重复计数器，仅在计数器的给定周期数之后生成中断或事件
- 最多四个独立通道，用于：
  - 输出比较
  - 输入捕捉
  - PWM 输出（边沿对齐和中心对齐）
  - 单次触发模式
- 用于加载和 CC 寄存器的影子寄存器
- 具有可编程死区插入功能的互补 PWM 输出
- 非对称 PWM 输出
- 故障处理机制，确保在遇到故障状况时，输出信号处于用户定义的安全状态
- 支持同一电源域中不同 TIMx 实例之间的同步和交叉触发

- 支持中断触发生成以及跨外设（例如 ADC 或 DAC）触发功能
- 两个用于内部事件的额外捕捉/比较通道

23.1.2.2 功能方框图

图 23-2 展示了 TIMA 方框图。

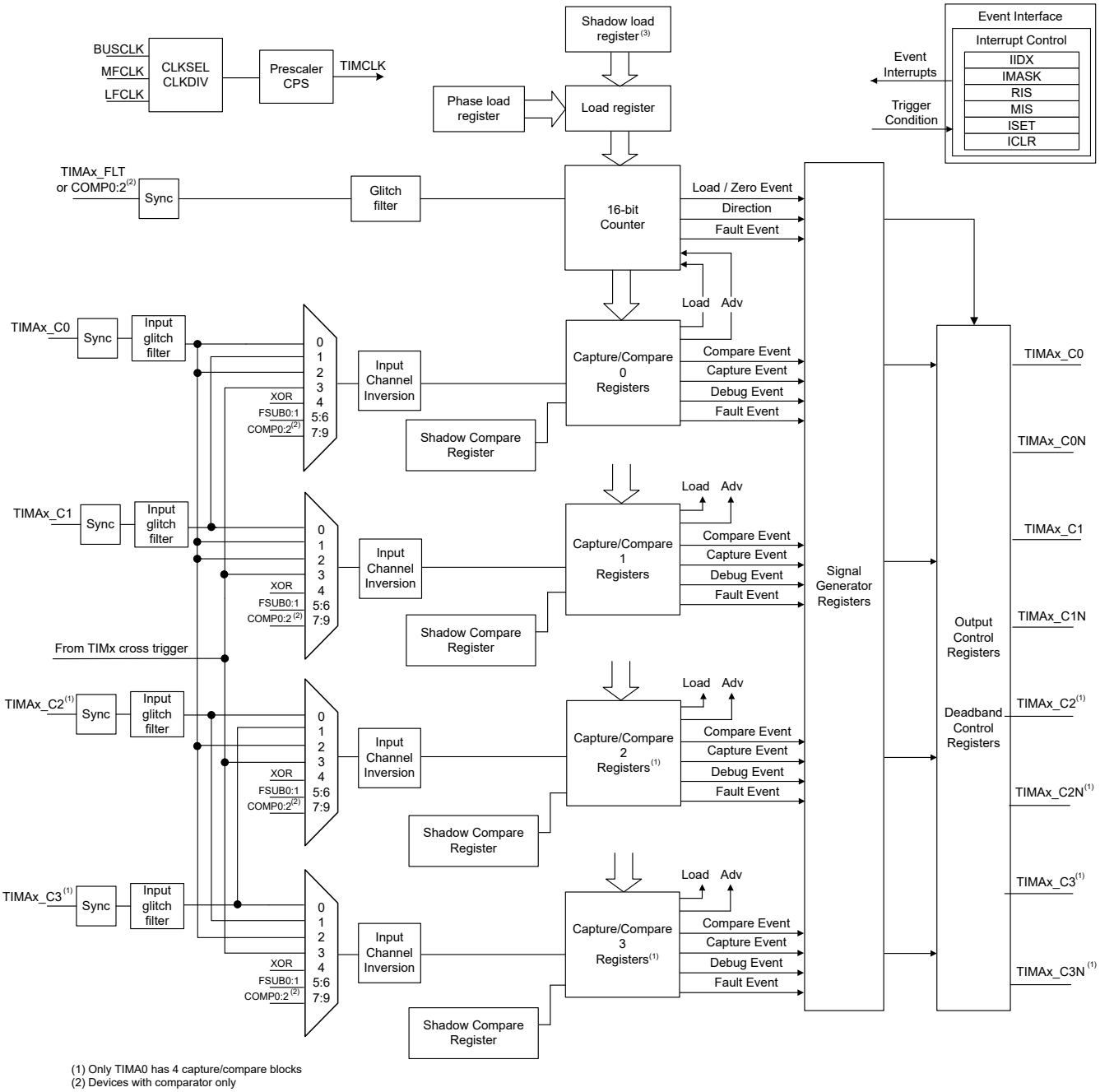


图 23-2. TIMA 方框图

23.1.3 TIMx 实例配置

表 23-1 展示了 TIMA 和 TIMG 实例的 TIMx 实例配置。

表 23-1. TIMx 实例配置

实例	电源域	计数器分辨率	预分频器	重复计数器	CCP 通道（外部/内部）	外部 PWM 通道	相负载	影子负载	影子 CC	Deadband	故障处理程序	QEI/霍尔输入模式
TIMG0	PD0	16 位	8 位	-	2	2	-	-	-	-	-	-
TIMG1	PD0	16 位	8 位	-	2	2	-	-	-	-	-	-
TIMG2	PD0	16 位	8 位	-	2	2	-	-	-	-	-	-
TIMG3	PD0	16 位	8 位	-	2	2	-	-	-	-	-	-
TIMG4	PD0	16 位	8 位	-	2	2	-	是	是	-	-	-
TIMG5	PD0	16 位	8 位	-	2	2	-	是	是	-	-	-
TIMG6	PD1	16 位	8 位	-	2	2	-	是	是	-	-	-
TIMG7	PD1	16 位	8 位	-	2	2	-	是	是	-	-	-
TIMG8	PD0	16 位	8 位	-	2	2	-	-	-	-	-	是
TIMG9	PD0	16 位	8 位	-	2	2	-	-	-	-	-	是
TIMG10	PD1	16 位	8 位	-	2	2	-	-	-	-	-	是
TIMG11	PD1	16 位	8 位	-	2	2	-	-	-	-	-	是
TIMG12	PD1	32 位	-	-	2	2	-	-	是	-	-	-
TIMG13	PD0	32 位	-	-	2	2	-	-	是	-	-	-
TIMG14	PD1	16 位	8 位	-	4	4	-	-	-	-	-	-
TIMA0	PD1	16 位	8 位	是	4/2	8	是	是	是	是	是	-
TIMA1	PD1	16 位	8 位	是	2/2	4	是	是	是	是	是	-

备注

在 TIMA 实例中，外部 PWM 通道是互补 PWM 输出信号对，具有相对于 CC 块实例的死区生成功能，例如 TIMA0_C0 和 TIMA0_C0N。为了生成独立的 PWM 输出，必须使用独立的同相通道，例如 TIMA0_C0 和 TIMA0_C1。

备注

内部 CC 通道 4 和 5 (CC_45) 可用于内部比较事件，仅在 TIMA 中可用。

备注

查看特定于器件的数据表，以便检查器件上提供了哪些 TIMx 实例。

23.2 TIMx 操作

TIMx 模块可由用户软件配置。以下各节将讨论 TIMx 的设置和运行。

备注

计时器模块中的寄存器数组可将用于不同捕获/比较端口的具有相同位字段的寄存器进行分组。例如，TIMx.CC_01[0/1] 是一个寄存器数组，其中包含 CCP0 和 CCP1 的捕获/比较寄存器。访问 TIMx.CC_01[0] 可读取 CC0 捕获/比较值，而访问 TIMx.CC_01[1] 可读取 CC1 捕获/比较值。

备注

TIMx 通过事件寄存器数组支持事件订阅者和事件发布者。有关更多信息，请参阅事件章节。

23.2.1 计时器计数器

所有 TIMx 实例都具有 16 位计数器块，但 TIMG12 和 TIMG13 除外，它们具有 32 位计数器块。计时器计数器寄存器 (TIMx.CTR) 可根据运行模式进行向下、向上-向下或向上计数。它还可以通过软件读取或写入。每次计数在 TIMCLK 信号的每个上升沿或外部信号的两个边沿发生。

启用 TIMx 计数器

计数器由预分频器输出 TIMCLK 计时。计数器使能位 TIMx.CTRCTL.EN 可以通过三种方式启用：

- 在软件中手动设置
- 该位可由软件进行设置以启用计数器，并会根据 TIMx.CTRCTL.CVAE 设置来设置启用后计数器值 (CVAE)。
- 在满足加载条件 (LCOND) 或零条件 (ZCOND) 后，将启用后计数器值 (CVAE) 分别更改为加载值或零值。

备注

可以在 TIMx 运行时对计数器寄存器进行写入，但应避免该操作，因为用户写入可能与加载、零或前进事件发生冲突。根据预分频比，应用程序无法预测写入的时序，从而影响正确的计时器周期。

23.2.1.1 时钟源选择和预分频器

TIMx 时钟 (TIMCLK) 可通过内部时钟或外部信号触发来推进时钟。

23.2.1.1.1 内部时钟和预分频器

通过设置 TIMx.CLKSEL 寄存器，TIMx 时钟 (TIMCLK) 能够以 BUSCLK、MFCLK 和 LFCLK 为时钟源。也可以通过设置 TIMx.CLKDIV 寄存器和预分频器（如果存在）来按比率进行分频。所选源时钟始终可用，频率取决于电源模式。更多信息，请参阅时钟模块 (CKM) 部分。

TIMCLK 可以来自以下来源：

- BUSCLK：当前总线时钟被选为 TIMx 的源。当前总线时钟取决于电源域。
  - 如果 TIMx 实例位于电源域 1 (PD1) 中，请参阅 MCLK。
  - 如果 TIMx 实例位于电源域 0 (PD0) 中，请参阅 ULPCLK。
- MFCLK：选择 MFCLK 作为 TIMx 的源，请参阅 MFCLK。
- LFCLK：选择 LFCLK 作为 TIMx 的源，请参阅 LFCLK。

选定的时钟源可以直接传递给 TIMx 或由 8 位预分频器分频。预分频器设置可通过寄存器 TIMx.cps.PCNT 位进行配置。所选 TIMCLK 源按值 (PCNT+1) 分频。PCNT 值 0 将 TIMCLK 按 1 分频，从而有效地绕过分频器。大于 0 的 PCNT 值会对 TIMCLK 源分频，以生成更慢的时钟。

TIMx 还具有用于禁用计时器时钟的软件机制。将 TIMx.CCLKCTL.CLKEN 设置为 0 以将计时器置于空闲状态。

TIMCLK 配置

要配置时钟源、分频器和预分频器，请进行以下操作：

1. 使用 CLKSEL 寄存器选择 TIMCLK 时钟源（BUSCLK、MFCLK 或 LFCLK）。
2. 可选择使用 CLKDIV.RATIO 对 TIMCLK 进行分频。
3. 在带有预分频器的 TIMx 实例中，可选择使用 CPS.PCNT 设置预分频器。
4. 通过设置 CCLKCTL.CLKEN = 1 来启用 TIMCLK。

TIMCLK 的频率由方程式 28 确定。

$$f_{TIMCLK} = \frac{f_{CLK\_SOURCE}}{((CLKDIV.RATIO + 1)*(CPS.PCNT + 1))} \quad (28)$$

23.2.1.1.2 外部信号触发

计数器还可以通过使用计时器输入引脚上的外部信号或通过系统中其他外设产生的事件触发而前进（递增或递减）。这可以通过使用高级条件设置 (TIMx.CCCTL_xy[0/1].ACOND) 来配置，以指定创建前进事件的内容。要指定哪些事件使计数器前进，请使用 TIMx.CTRCTL.CAC 设置。

计数器可以使用内部时钟 TIMCLK、计时器外部输入的不同边沿或来自其他外设的内部触发事件而前进。

23.2.1.2 重复计数器（仅限 TIMA）

在仅 TIMA 中，重复计数器 (RC) 是一个 8 位计数器，它提供抑制不必要事件和生成真实事件的机制，以实现更佳的中断生成。具体来说，如果计时器正在生成针对已知周期数重复发生的事件（例如周期性 PWM 输出波形），则重复计数器可以抑制加载事件、比较事件和归零事件。这可防止在每个计时器周期内产生过多和不必要的中断。

当计时器计数器 (TIMA.CTR) 前进时，一旦计数器重新加载 (TIMA.CTR = 0)，重复计数器 (TIMA.RC) 就会前进。用户可以通过设置 TIMA.RCLD 寄存器，设置在生成中断和事件之前重新加载的计时器计数器的数量。当 TIMA.RC = TIMA.RCLD 时，重复计数器复位回零，并且在中断和事件状态寄存器中发生重复计数器归零事件 (REPC)。

备注

如果计数器配置为在调试或故障条件下停止计数，则重复计数器也应停止。有关更多详细信息，请参阅节 23.2.6 和节 23.2.10。

此外，当 TIMA.RC 不等于零时，重复计数器提供抑制归零事件、加载事件和比较事件生成的功能。

- 通过设置 TIMA.CTRCTL.SLZERCNEZ 寄存器位来抑制归零事件和负载事件
- 通过设置 TIMA.CCCTL_xy[0/1].SCERCNEZ 位来抑制比较事件（请参阅表 23-8）

表 23-2 显示了相对于计时器计数器和重复计数器负载值的重复计数器行为。

表 23-2. 重复计数器行为

TIMA.CTR 前进 (+1)	计数器值	TIMA.RC = TIMA.RCLD	重复计数器行为	抑制负载和归零事件 (SLZERCNEZ = 1)	抑制比较事件 (SCERCNEZ = 1)
否	-	-	不前进	是	是
是	TIMA.CTR ≠ 0	-	不前进	是	是
是	TIMA.CTR = 0	否	前进 (+1)	是	是
是	TIMA.CTR = 0	是	TIMA.RC = 0	否	否

重复计数器示例

如图 23-3 所示，TIMA.CTR 配置为向下计数模式，一旦 TIMA.CTR = 0，就会生成归零事件。要抑制中断生成，直至发生 4 次计时器重新加载，请设置 TIMA.RCLD = 4 和 TIMA.CTRCTL.SLZERCNEZ = 1，以抑制归零事件和加载事件，直到 RC = 0（一旦 TIMA.RC = TIMA.RCLD，就会发生）。

备注

使用重复计数器不会影响输出信号生成。无论发送到信号发生器单元的重复计数器值是多少，都会生成所有事件。

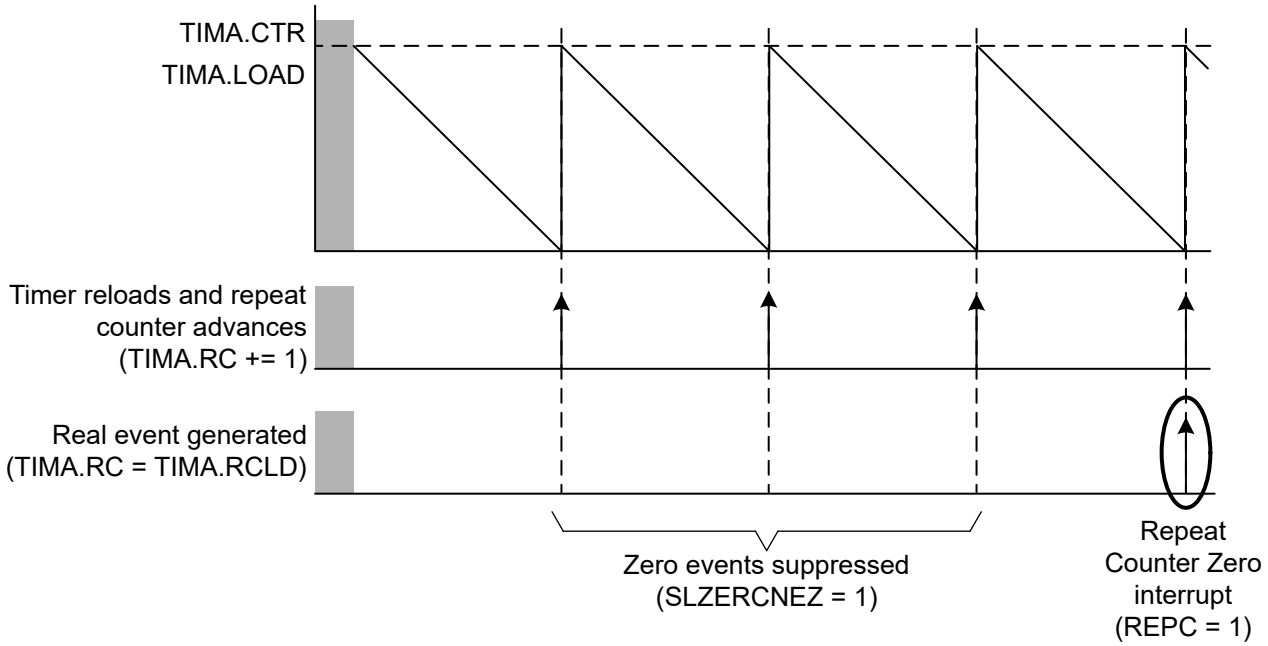


图 23-3. 在向下计数模式下，当 TIMx.RCLD = 4 时，重复计数器抑制事件

23.2.2 计数模式控制

当器件未复位时，TIMx 被禁用。向 TIMx.CTRCTL.EN 位写入 1 会启用计数器。如果 TIMx.CTRCTL.REPEAT=0（不自动重新加载）且计数器值为零，则该位自动清零。

TIMx 启用时有三种计数模式：向下、向上/向下和向上。工作模式由 TIMx.CTRCTL.CM 位选择（如表 23-3 所示）。启用计数器后，计时器将在启用后的计数值 (TIMx.CTRCTL.CVAE) 设置后开始计数。

表 23-3. TIMx 计数模式

CM 位	模式	启用后的计数值 (CVAE)	说明
0	向下计数	CVAE = 0（加载值）	TIMx 从 TIMx.LOAD 值向下计数到零
1	向上/向下	CVAE = 0（加载值）	TIMx 从 TIMx.LOAD 值向下计数到零，然后向上计数回到 TIMx.LOAD
		CVAE = 2（零值）	TIMx 从零向上计数到 TIMx.LOAD，然后向下计数回到零
2	向上	CVAE = 2（零值）	TIMx 从零向上计数到 TIMx.LOAD 值

备注

TIMx.CTRCTL.CVAE = 1 对启用后的计数值没有影响。

23.2.2.1 单次触发和周期模式

图 23-4 展示了 TIMx 在单次触发模式和周期模式下的工作情况。

单次触发模式

当 TIMx.CTRCTL.REPEAT 位设置为 0 时，TIMx 在以下情况下不会前进：

- TIMx.CTR 值在向下或向上/向下计数模式下达到 0

- TIMx.CTR 值在向上计数模式下达到 TIMx.LOAD

周期性模式（计数器重新加载）

当 TIMx.CTRCTL.REPEAT 设置为 1h 时，一旦发生归零事件，TIMx 将自动重复。在以下情况下会发生这种情况：

- TIMx.CTR 在向下或向上/向下计数模式下达到 0
  - 在向下计数模式下，归零事件之后，在下一个前进条件下是加载事件
  - 在向上/向下计数模式下，归零事件之后跟随一个前进事件 (+1)
- TIMx.CTR 在向上计数模式下达到 TIMx.LOAD
  - 加载事件之后，在下一个前进条件下是归零事件

TIMx.CTRCTL.REPEAT 也可以设置为 3h，以便 TIMx 仅在不处于调试条件时重复。如果存在调试条件，TIMx 将计数到归零事件，并且仅在移除调试条件后重复。

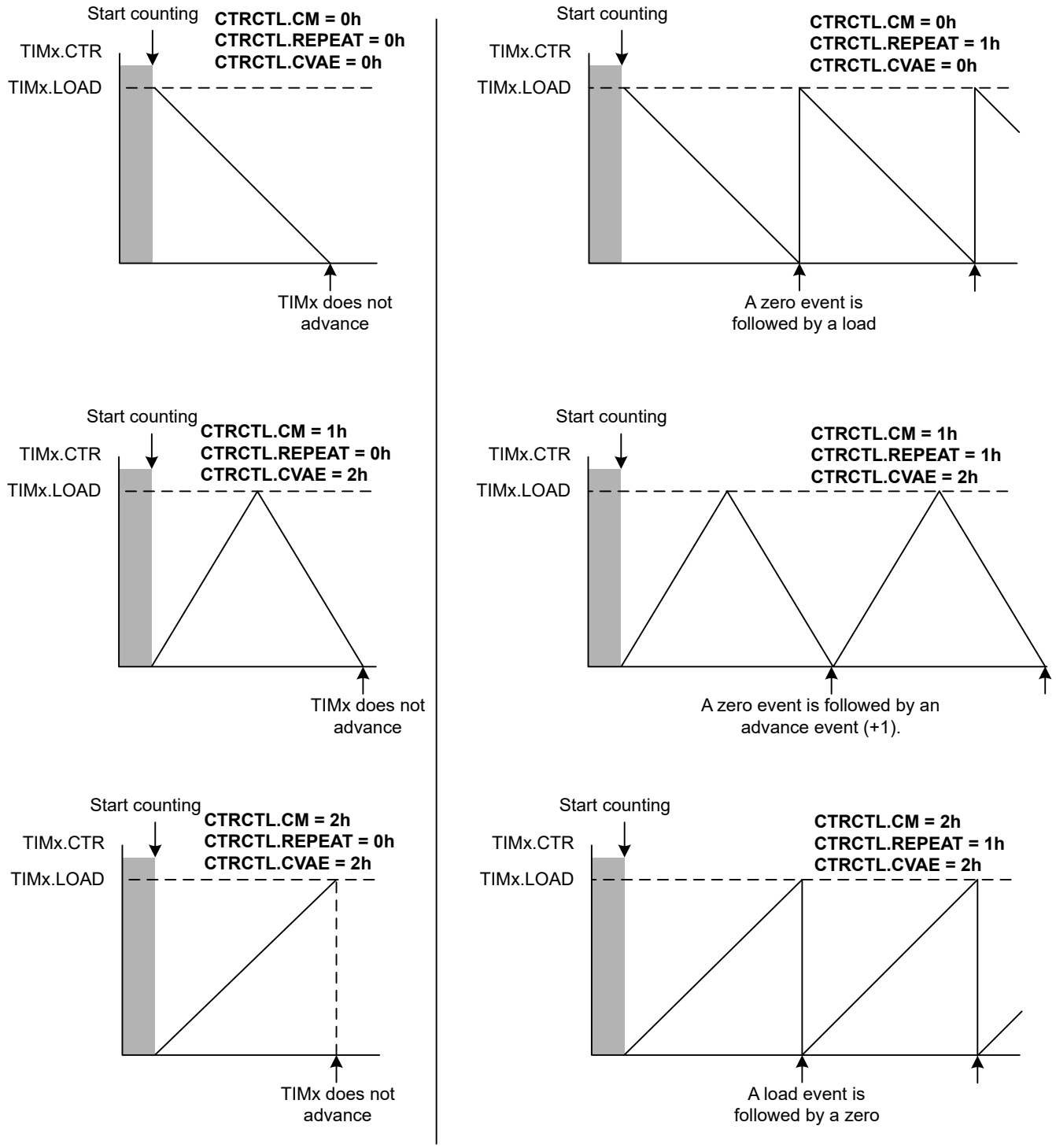


图 23-4. 单次触发行为和周期模式行为

23.2.2.2 向下计数模式

在向下计数模式 (CM = 0) 下，TIMx 从 TIMx.LOAD (CVAE = 0) 中定义的值向下计数到零。当 TIMx.CTR 值等于零，且 TIMx.CTRCTL.REPEAT 设置为 1 时，TIMx.LOAD 值加载到 TIMx.CTR，计时器重复向下计数模式，如图 23-5 中所示。

当 TIMx 计数为零时，将生成归零事件。当 TIMx 从零计数到 TIMx.LOAD 值时，将生成加载事件。图 23-6 展示了事件生成周期。

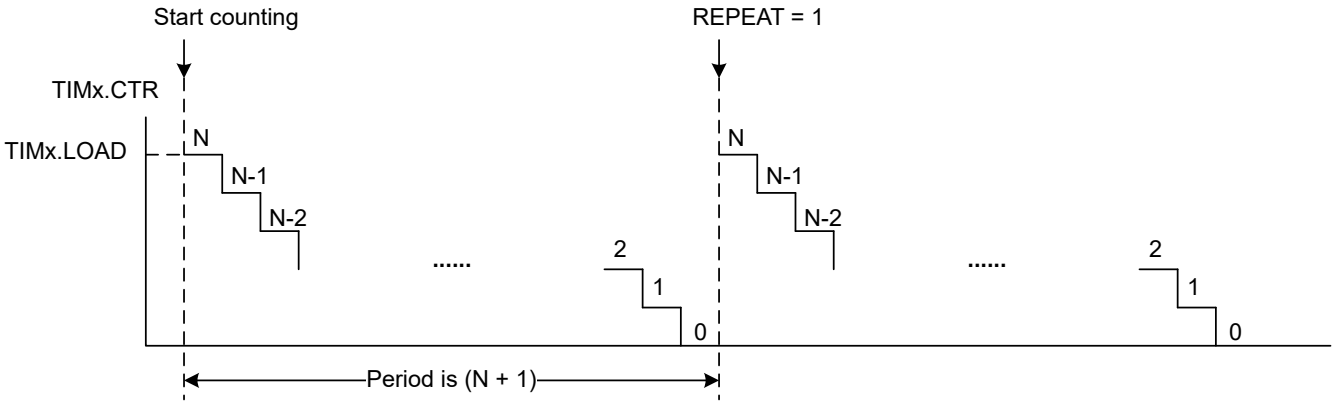


图 23-5. 向下计数模式，CVAE = 0

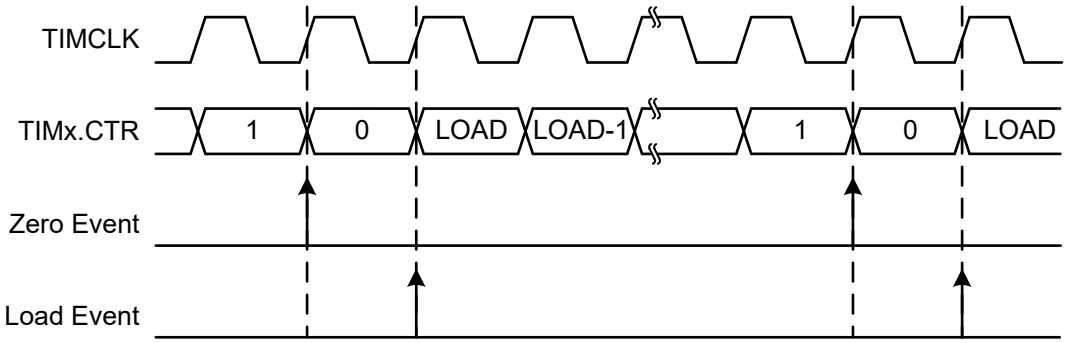


图 23-6. 向下计数模式事件生成

23.2.2.3 向上/向下计数模式

根据 TIMx.CTRCTL.CVAE 值，向上/向下计数模式可进行向下-向上或向上-向下计数。TIMx.CTRCTL.CVAE 位指定计数器的初始化条件。

表 23-4. 启用初始化条件后的计数器值

TIMx.CTRCTL.CVAE 值	启用后计数器值
0x0	加载值
0x1	没有变化
0x2	零

向下-向上计数

当 TIMx.CTRCTL.CVAE = 0 时，TIMx.CTR 设置为 TIMx.LOAD 寄存器值，TIMx 向下计数。当达到零时，会生成一个归零事件，TIMx 会重新向上计数至 TIMx.LOAD 值。当达到 TIMx.LOAD 值时，会生成一个加载事件。

图 23-7 展示了当 TIMx.CTRCTL.CVAE = 0 时的 TIMx 向下-向上计数。

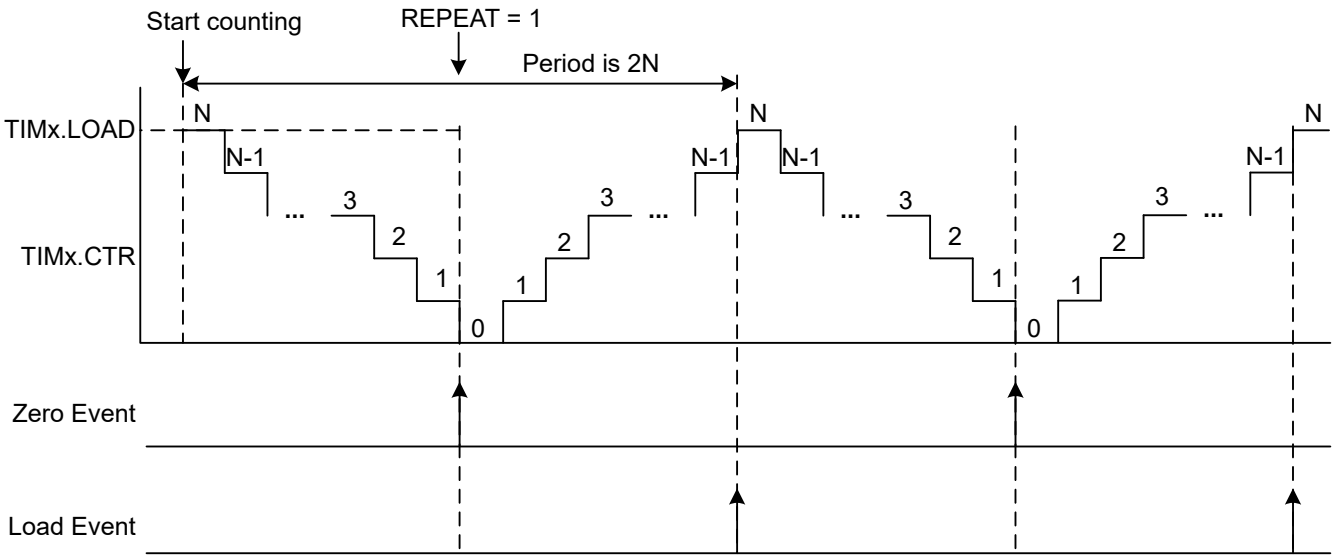


图 23-7. 向下-向上计数模式和事件生成，CVAE = 0

向上-向下计数

当 TIMx.CTRCTL.CVAE = 2 时，TIMx.CTR 设置为零，TIMx 向上计数。当达到 TIMx.LOAD 时，会生成一个加载事件，TIMx 会重新向下计数至零。当达到零时，会生成一个归零事件。

图 23-8 展示了当 TIMxCTRCTL.CVAE = 2 时的 TIMx 向上-向下计数。

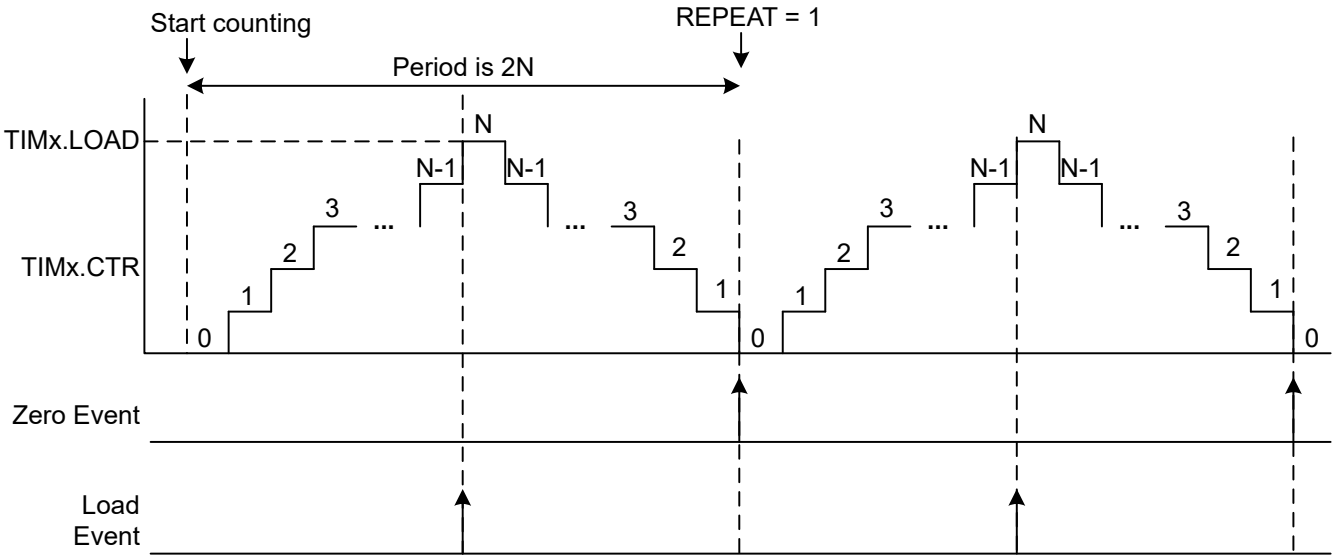


图 23-8. 向上-向下计数模式和事件生成，CVAE = 2

23.2.2.4 向上计数模式

在向上计数模式下，TIMx 从零向上计数到 TIMx.load 中定义的值。当 TIMx.CTR 值等于 TIMx.LOAD 且 TIMx.CTRCTL.REPEAT 不等于 0 时，零被加载到 TIMx.CTR 中，计时器重复向上计数模式，如图 23-5 所示。

当 TIMx 计数到 TIMx.LOAD 时，会生成一个加载事件。当 TIMx 从 TIMx.LOAD 计数到零值时，将生成归零事件。图 23-6 展示了事件生成周期。

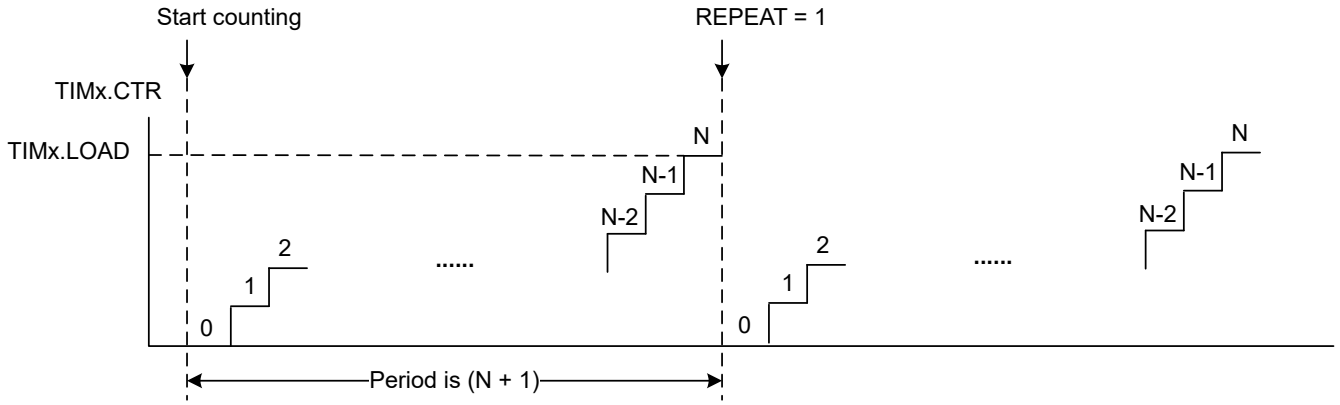


图 23-9. 向上计数模式，CVAE = 2

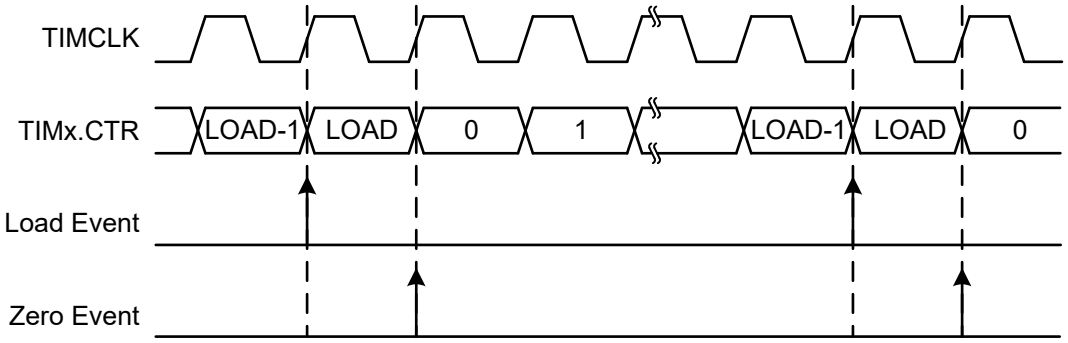


图 23-10. 向上计数模式事件生成

23.2.2.5 相位加载（仅限 TIMA）

（仅在 TIMA）相位加载寄存器 TIMA.PL 使 TIMA.CTR 能够在向上/向下计数模式下从零或 TIMA.LOAD 以外的值进行计数。相位加载用于生成非对称的中心对齐 PWM 输出信号，并在不同计时器实例之间控制相移。

当 TIMA.PL 为非零时，通过设置 TIMA.CTRCTL.PLEN = 1 来启用相位加载，并在 TIMA.CTTRIG.TRIG = 1 时触发相位加载。当在 TIMA.CTRCTL.CVAE = 0 时触发相位加载后，计时器从 TIMA.PL 值向下计数。当在 TIMA.CTRCTL.CVAE = 0 时触发相位加载后，TIMx 从 TIMA.PL 值向上计数。

TIMA.PL 在计时器启动时被锁存，并且每当计数器达到先前锁存的 TIMA.PL 值时，TIMA.PL 就会同步。图 23-11 展示了当计时器向上-向下计数且相位加载值变为新值时的相位加载寄存器工作原理。

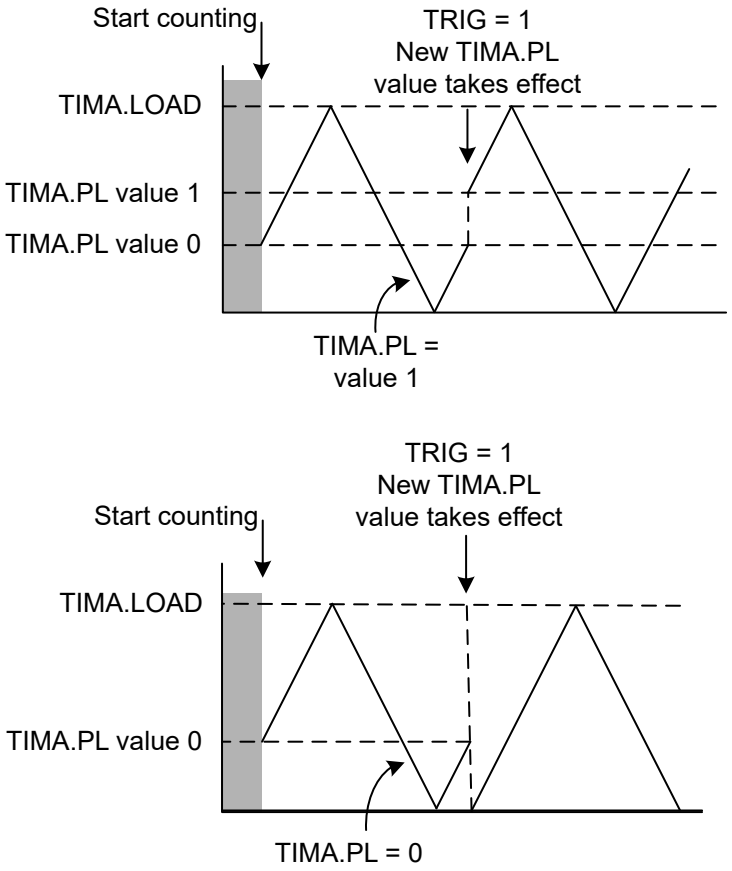


图 23-11. 向上-向下模式下的相位加载寄存器同步

23.2.3 捕获/比较模块

捕获/比较 (CC) 块用于捕获事件或比较事件。TIMG 具有多达 2 个相同的捕获/比较块，TIMA 具有多达 4 个相同的捕获/比较块，用于支持外部或内部信号。此外，TIMA 还为仅来自内部信号的比较事件提供了 2 个额外的 CC 块（CC4 和 CC5）。这些 TIMx 捕获/比较块中的任何一个都可用于捕获输入信号的时序或生成时间间隔。

23.2.3.1 捕获模式

当 TIMx.CCCTL_xy[0/1].COC 位被设定为 1 时，会选择捕获模式。捕获模式用于生成捕获事件并记录时间间隔，这对于速度计算或时间测量很有用。

用于配置捕获模式的主要寄存器：

- **TIMx.LOAD**：在发生指定执行“加载”的任何操作时，此寄存器的内容都会复制到计数器 (TIMx.CTR)。该值还用于与计数器值进行比较，以便生成可用于中断、触发或信号生成操作的“加载事件”。
- **TIMx.CC_xy[0/1]**：该寄存器可用作捕获寄存器来获取或记录事件的下一个计数器值，或用作当前计数器的比较寄存器来创建事件。
- **TIMx.CCCTL_xy[0/1]**：该寄存器可控制各个 CC（捕获/比较）块的操作。例如，它可以配置上升沿还是下降沿会生成一个捕获条件。
- **TIMx.CTRCTL**：此寄存器用于控制不同条件下的计数器操作。
- **TIMx.IFCTL_xy[0/1]**：该寄存器控制相关 CC 模块的输入滤波（FE、FP、CPV）、选择 (ISEL) 和反转 (INV)。

23.2.3.1.1 输入选择和反转

TIMx.IFCTL 寄存器用于为捕获/比较块选择输入源、滤波和最终反转选项。

图 23-12 展示了具有两个 CC 通道的 TIMx 捕获块的方框图。

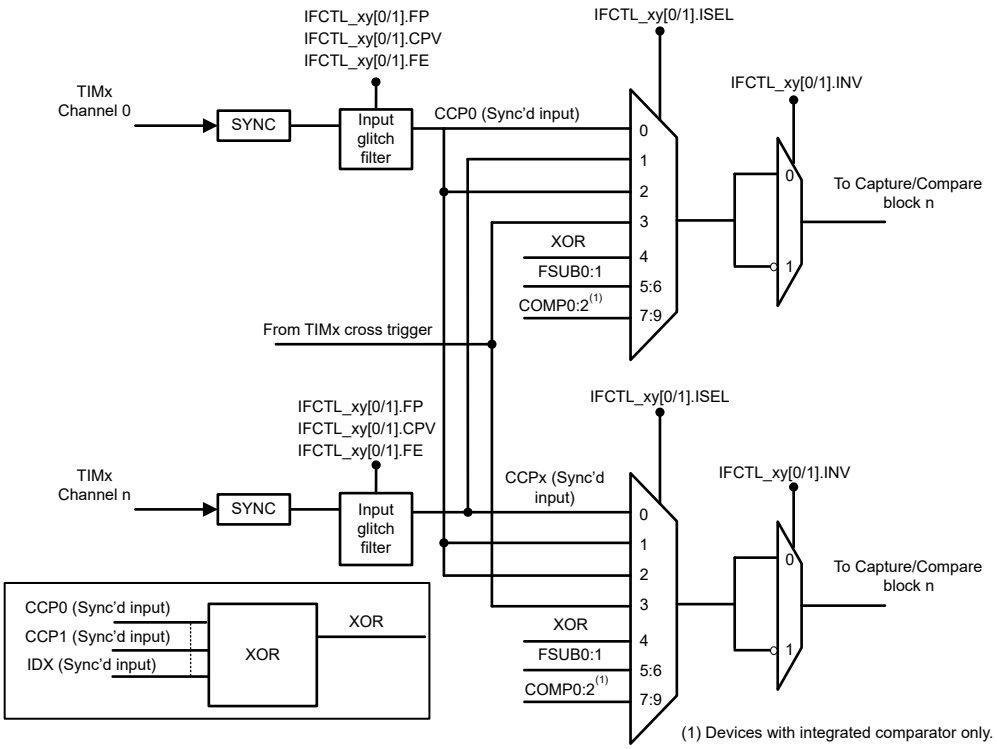


图 23-12. TIMx 捕获方框图

CCP 输入边沿同步

当使用捕获/比较引脚 (CCP) 作为输入时，请配置 PINCMx 用于 TIMx CCP 输入（如 TIMG0_C0）。

CCP 输入信号始终通过同步器，并且输入状态（高电平或低电平）必须大于一个 TIMCLK 时钟周期，以便同步器检测到边沿。CCP 输入边沿检测至少需要一个 TIMCLK 周期来同步边沿输入。第一个 TIMCLK 周期中的时序是不确定的，因为无法预测第一个 TIMCLK 周期中的边沿检测。

当捕获条件发生时，需要一个额外的 TIMCLK 周期来生成捕获事件。

输入滤波

可以通过设置 TIMx 来启用输入干扰滤波器。IFCTL_01[0/1].FE 位。滤波器周期通过设置 TIMx 进行配置。IFCTL_01[0/1].FP 位。

由 TIMx.IFCTL_xy[0/1].CPV 位选择的连续周期或多数表决格式用于选择 CCP 输入信号的标准。

- **连续周期** - CCP 输入信号必须在定义的 FP 计时器时钟数内处于指定电平，以便处理 CCP 输入。
- **多数表决** - 滤波器在滤波器周期内忽略一个相反逻辑的时钟。例如，在输入的 FP 样本数量中，最多 1 个样本可能具有相反的逻辑值（干扰）而不影响输出。

图 23-13 中所示的示例展示了连续周期和多数表决格式之间的差异，其中实现了一个数字滤波器以捕获 3 个 TIMCLK 周期的 CCP 输入信号。

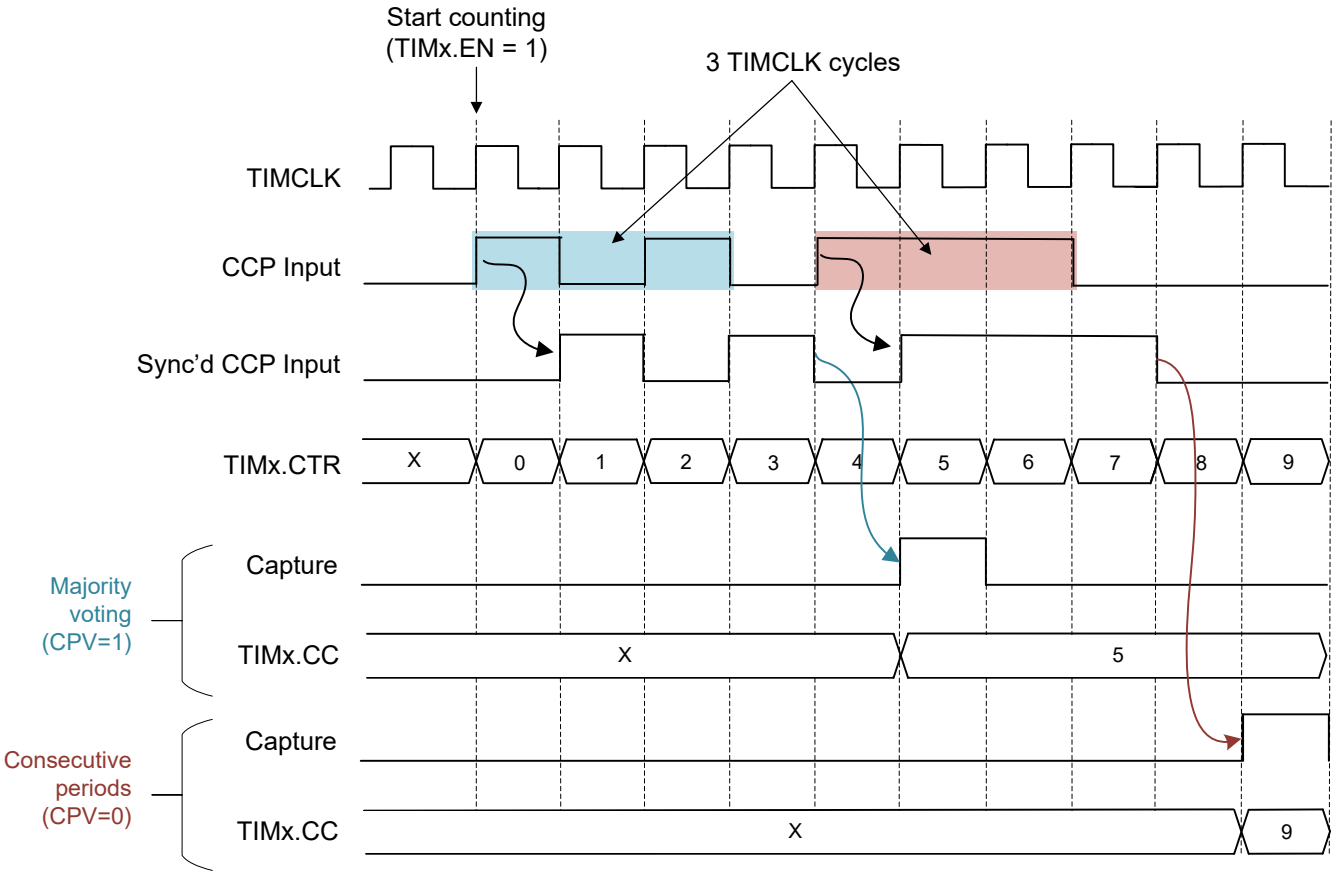


图 23-13. 使用 FP = 0（3 个 TIMCLK 周期）的输入干扰滤波的连续周期和多数表决

输入选择

输入选择位 TIMx.IFCTL_xy[0/1].ISEL 选择滤波器输入的输入源作为相应的 CCP 输入、CCP0 用于 CC 块上交叉触发、外部触发、异或（用在霍尔输入模式中）、事件订阅者或比较器输入。

23.2.3.1.2 用例

在捕获模式下可实现多种不同的用例，后续章节中将对此进行讨论。

23.2.3.1.2.1 边沿时间捕获

边沿时间捕获功能测量从捕获操作开始到信号边沿的时间（以 TIMCLK 周期为单位）。计数器在 TIMx 启用时加载，并在每个 TIMCLK 周期进行计数，直到检测到 CCP 边沿，这会触发计时器值捕获并生成捕获事件。捕获边沿时间等于计数器的起始值与 TIMx.CC_xy[0/1] 寄存器中的捕获值之间的差值。

边沿时间捕获配置

1. 设置 TIMx.LOAD 值。
2. 在 CTRCTL 寄存器中，为以下各项设置所需的计数器控制设置：
   a. 计数模式 (CM) 和启用后计数器值 (CVAE)（请参阅节 23.2.2）
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
3. 针对捕获模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
4. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
5. 对于相应的 CC 块控制寄存器 (CCCTL_01[0/1])，将 CCOND 进行相应的设置，从而根据输入信号条件（上升沿和/或下降沿）捕获事件。此外，根据所使用的计数模式设置 ZCOND 或 LCOND。

6. 按照节 23.2.3.1.1 中所述，配置 TIMx.IFCTL_xy[0/1] 寄存器中的输入捕获设置。
7. 通过设置 EN = 1 或等待从输入边沿发生捕获事件来启用计数器。

使用向上计数模式进行上升沿捕获的示例

在从零开始的向上计数模式（CM = 2，CVAE = 2）下，通过将 ZCOND 设置为 1，TIMx 可配置为生成一个零脉冲，并根据配置的捕获事件 (CCOND) 启动计数器。

图 23-14 展示了在向上计数模式下上升沿或正边沿时间捕获的预期内部时序。

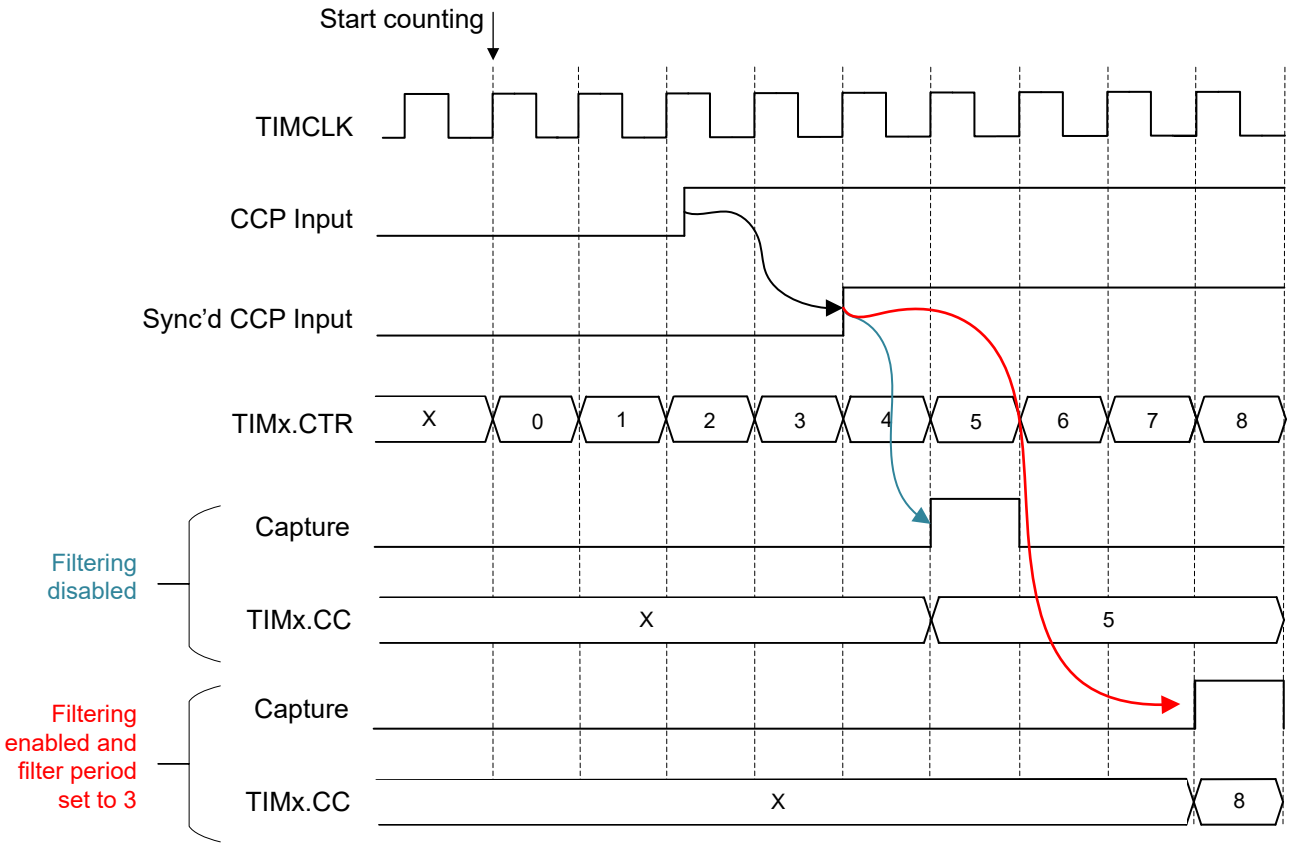


图 23-14. CVAE = 2 向上计数模式下的边沿时间捕获模式

23.2.3.1.2.2 周期捕获

周期捕获测量 TIMCLK 周期内输入 CCP 上的信号周期。在 CCP 输入的每个正（或负）边沿，TIMx.CTR 值都被捕获到 TIMx.CC 寄存器中以生成捕获事件。周期捕获时间等于生成的计数器起始值与捕获事件之间的差值，或周期性输入信号的重复捕获事件之间的时间。

周期捕获模式

1. 设置 TIMx.LOAD 值。
2. 在 CTRCTL 寄存器中，为以下各项设置所需的计数器控制设置：
   a. 计数模式 (CM) 和启用后计数器值 (CVAE)（请参阅节 23.2.2）
   b. 前进 (CAC)，用以指定控件在什么条件下将计数器向前推进
   c. 重复或单次触发模式 (REPEAT)
3. 针对捕获模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
4. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
5. 对于相应的 CC 块控制寄存器 (CCCTL_01[0/1])，
   a. 根据输入信号条件（上升沿和/或下降沿）将 CCOND 设置为相应的设置以捕获事件

b. 根据所使用的计数模式，设置 ZCOND 或 LCOND
6. 按照节 23.2.3.1.1 中所述，配置 TIMx.IFCTL_xy[0/1] 寄存器中的输入捕获设置。
7. 通过设置 EN = 1 或等待从输入边沿发生捕获事件来启用计数器。

使用向上计数模式进行上升沿周期捕获的示例

在从零开始的向上计数模式下（CM = 2，CVAE = 2），通过设置 CCOND = 1h，TIMx 通道 0 可配置为从上升沿输入生成捕获事件。启用计数器后，当检测到上升沿输入时，计数器将捕获 TIMx.CC 中的计数器值。在捕获事件之后，将 TIMx.LOAD 值设置回 0 以复位周期 CCP 输入信号的计时器计数器。

图 23-14 展示了使用两个上升沿时向上计数模式下周期捕获的预期内部时序。

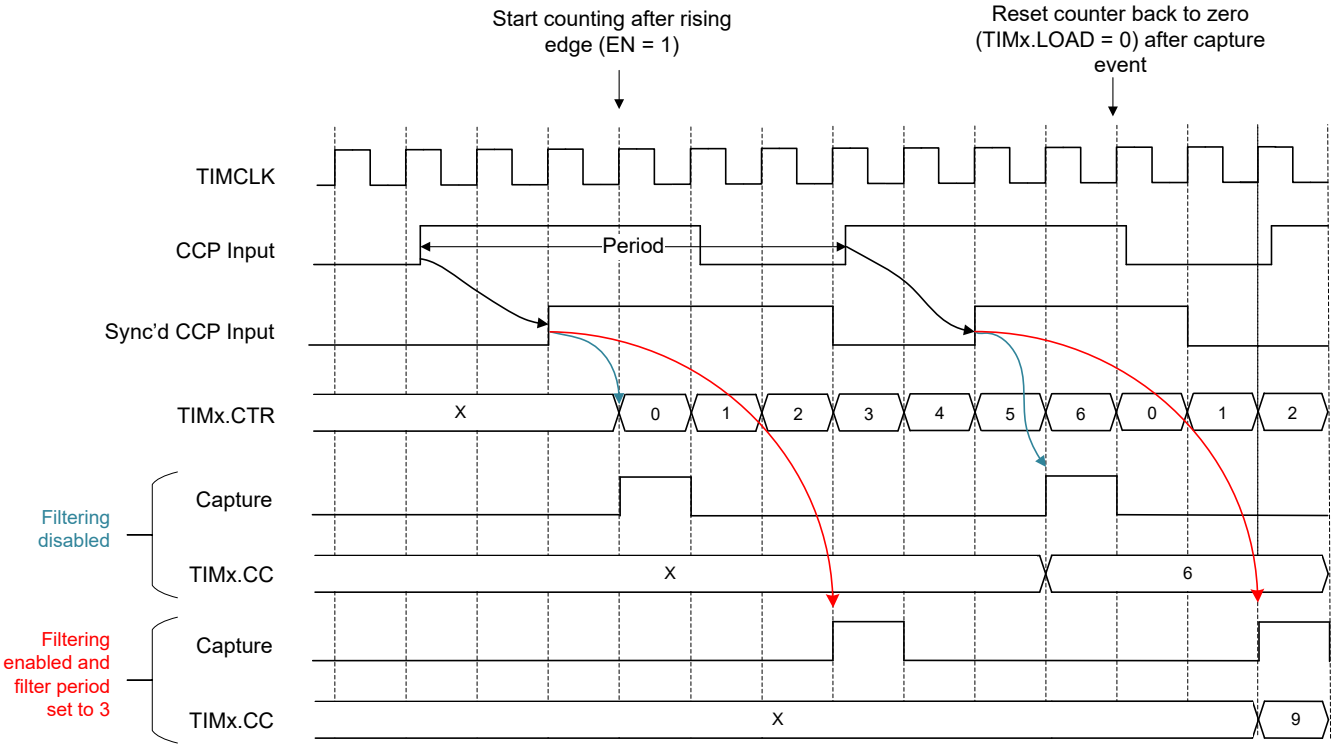


图 23-15. 向上计数模式下的周期捕获模式，CVAE = 2

23.2.3.1.2.3 脉宽捕捉

脉宽捕捉可以测量 CCP 上信号的高电平时间。高电平时间是从 CCP 输入的上升沿到下降沿的 TIMCLK 周期数，对于测量 PWM 输入信号的占空比等应用非常有用。计数器在正边沿被加载，并在负边沿被捕获（生成捕获事件）。

脉宽捕获模式

1. 设置 TIMx.LOAD 值。
2. 在 CTRCTL 寄存器中，为以下各项设置所需的计数器控制设置：
   a. 计数模式 (CM) 和启用后计数器值 (CVAE)（请参阅节 23.2.2）
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
3. 针对捕获模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
4. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
5. 对于相应的 CC 块控制寄存器 (CCCTL_01[0/1])，将 CCOND 进行相应的设置，从而根据输入信号条件（上升沿和/或下降沿）捕获事件。此外，根据所使用的计数模式设置 ZCOND 或 LCOND。
6. 按照节 23.2.3.1.1 中所述，配置 TIMx.IFCTL_xy[0/1] 寄存器中的输入捕获设置。

7. 通过设置 EN = 1 或等待从输入边沿发生捕获事件来启用计数器。

使用向上计数模式进行脉冲宽度捕获的示例

在从零 (CM = 2, CVAE = 2) 开始的向上计数模式下，TIMx 通道 0 可配置为生成一个零脉冲，并通过将 ZCOND 设置为 1 来根据配置的捕获事件 (CCOND) 启动计数器。要启动计数器，可以通过设置 LCOND = 1 从 CCP 上升沿输入触发加载条件。

图 23-14 展示了使用上升沿和下降沿在向上计数模式下进行脉冲宽度捕获的预期内部时序。

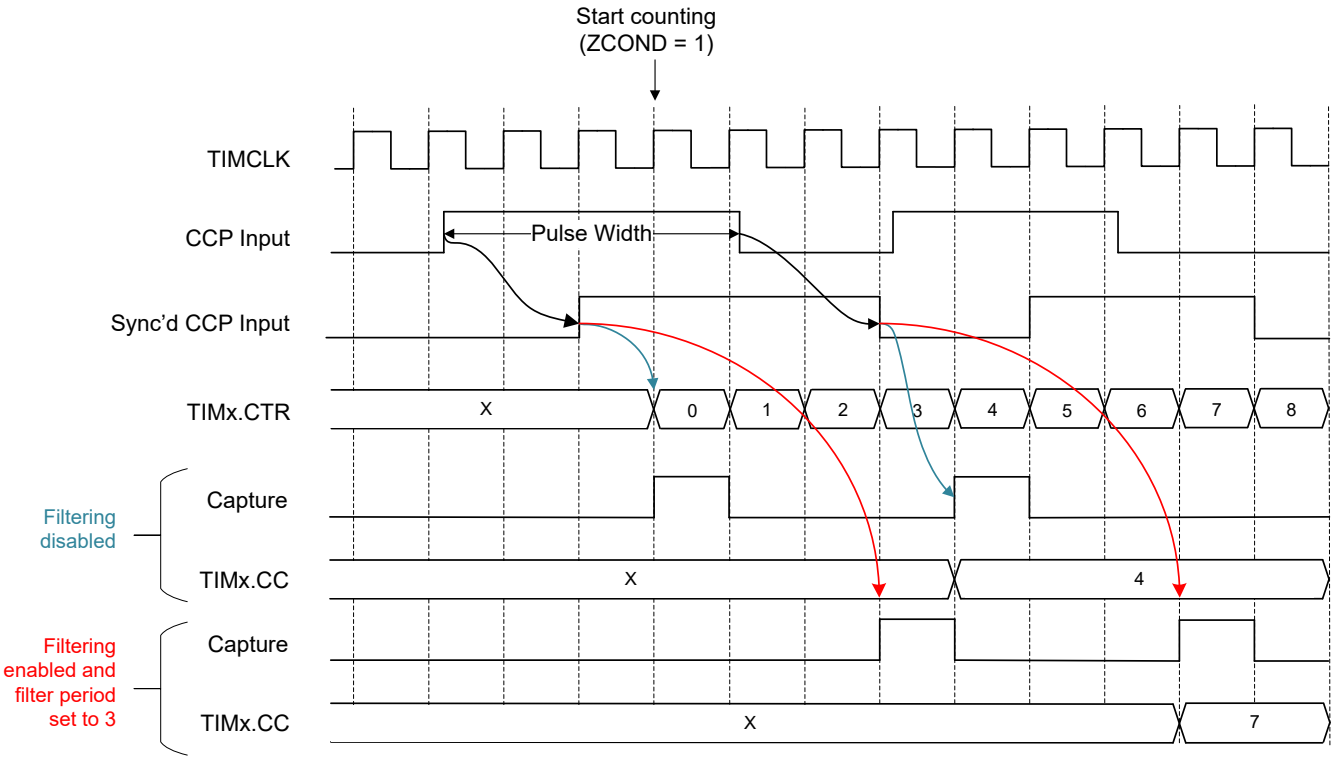


图 23-16. 脉宽捕获模式

23.2.3.1.2.4 组合的脉宽和周期时间

使用两个捕获寄存器可以同时捕获单个输入波形的脉宽和周期。输入信号可以从外部连接到 CCP 通道 0，IFCTL_01[1] 寄存器可以配置为在内部将输入连接到 CCP 通道 1，从而使捕获寄存器 0 (TIMx.CC0) 来捕获脉宽，捕获寄存器 1 (TIMx .CC1) 来捕获周期。组合脉宽和周期捕获的预期内部时序如图 23-17 所示。

脉宽和周期捕获配置

1. 设置 TIMx.LOAD 值。
2. 在 CTRCTL 寄存器中，为以下各项设置所需的计数器控制设置：
   a. 计数模式 (CM) 和启用后计数器值 (CVAE) (请参阅节 23.2.2)
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
3. 为每个 CC 通道的捕获模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
4. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为每个 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
5. 对于相应的 CC 块控制寄存器 (CCCTL_01[0/1])，
   a. 根据输入信号条件 (上升沿和/或下降沿) 将 CCOND 设置为相应的设置以捕获事件
   b. 根据所使用的计数模式，设置 ZCOND 或 LCOND。
6. 按照节 23.2.3.1.1 中所述，配置 TIMx.IFCTL_xy[0/1] 寄存器中的输入捕获设置。

7. 通过设置 EN = 1 或等待从输入边沿发生捕获事件来启用计数器。

使用脉宽和周期时间捕获的示例

在向上计数模式下，TIMx 可配置为生成一个零脉冲，并通过将 ZCOND 设置为 1 来根据配置的捕获事件 (CCOND) 启动计数器。

图 23-14 展示了使用两个 CC 块在向上计数模式下进行脉宽和周期捕获的预期内部时序。

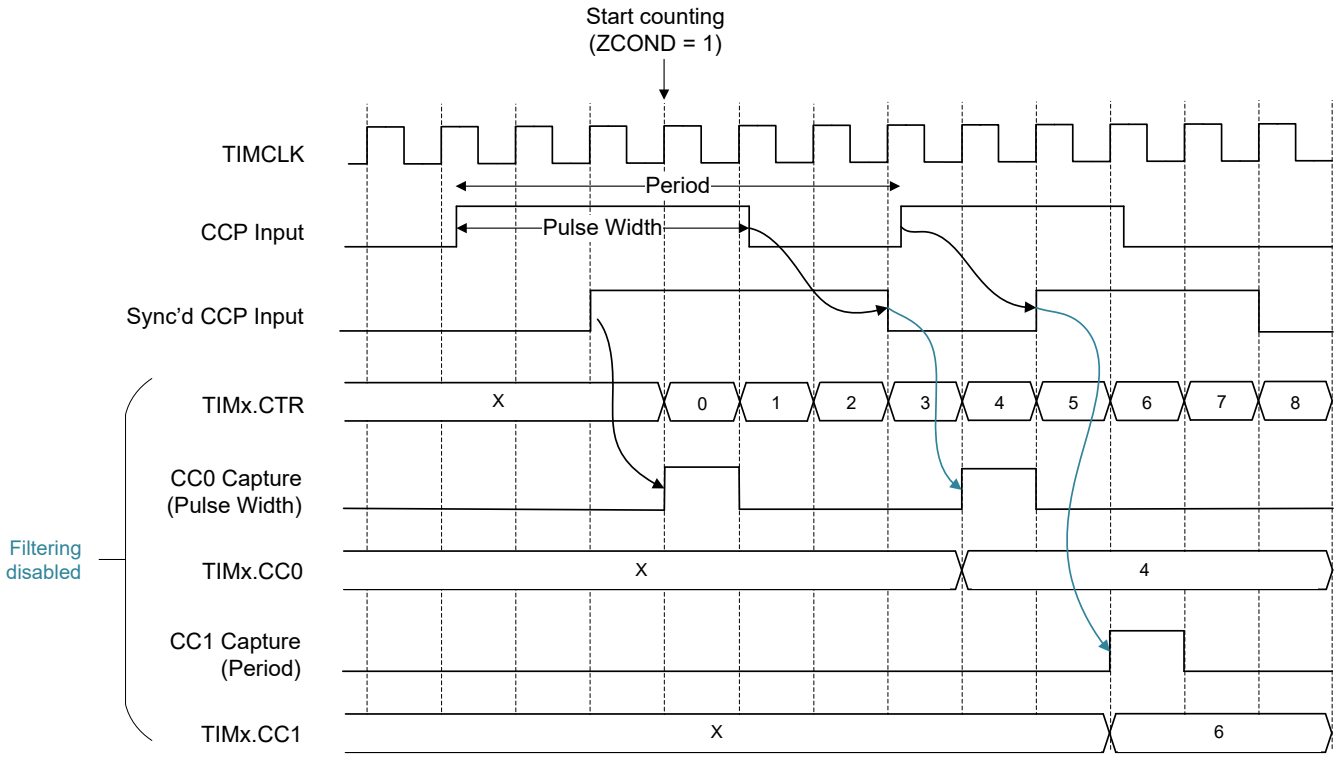


图 23-17. 组合的脉宽和周期捕获

23.2.3.1.3 QEI 模式 (仅限支持 QEI 的 TIMG)

在支持 QEI 的 TIMGx 实例中，正交编码器接口 (QEI) 模式提供了一个连接正交编码器输出的接口。它可以对正交编码的数据进行解码，以提供线性或旋转运动的相对定位和移动的相关信息。

QEI 由两个灰度编码的正交输入信号 PHA 和 PHB 以及一个索引输入信号 IDX 组成。所有输入信号都进入单个计数器的 CCP 输入，使 PHA 和 PHB 映射到 CCP0 和 CCP1，而 IDX 作为单独的输入引入。一个错误检测机制可以报告错误转换，以避免不正确的信号解码。

备注

有关支持 QEI/霍尔输入模式的 TIMG 实例，请参阅节 23.1.3 和特定于器件的数据表。

23.2.3.1.3.1 具有 2 信号的 QEI

QEI 用于解码来自光学位置编码器的信号。光学位置编码器通常输出 2 个信号 (PHA/PHB)，这些信号通常用于测量具有旋转轴的物理部件 (例如三相电机) 的旋转或线性移动。该接口提供的测量是递增的；也就是说，当发生移动时，该接口能捕获到与上一个位置的相对变化。

在 QEI 模式下运行时，计数器会累积增量更新，从而根据初始位置推导出当前位置和累积的变化。初始位置可由 IDX 输入信号 (3 信号 QEI 模式) 或其他软件方法 (软件直接设置初始位置) 确定。通过定义捕获条件，捕获和比较寄存器可用于存储位置值。

当发生方向改变 (DC) 时，RIS 寄存器中会产生一个 DC 中断。

图 23-18 展示了 QEI 接口中用于通过两个 CCP 输入信号（PHA 和 PHB）检测旋转方向的状态机。

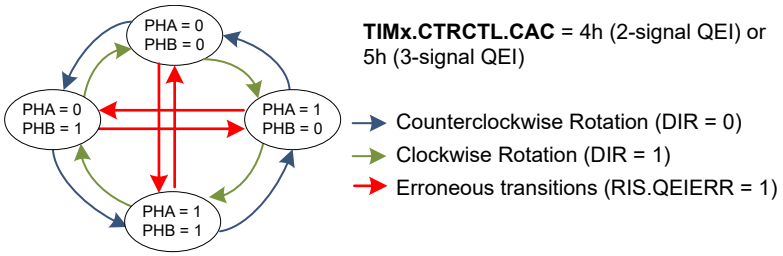


图 23-18. 用于 2 信号和 3 信号 QEI 模式的状态机

QEI 2 信号模式配置

1. 为 TIMGx_C0 (PHA) 和 TIMGx_C1 (PHB) 配置 PINCMx。
2. 对于 CCP 通道 0 和 1（PHA 和 PHB）的捕获模式，设置 TIMG.CCCTL_01[0].COC = 1 和 TIMG.CCCTL_01[1].COC = 1。
3. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为每个 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
4. 将 TIMx.LOAD 设置为编码器分辨率，例如 4000。
5. 在 CTRCTL 寄存器中，将 CAC、CZC 和 CLC 位设置为 4h。
6. 如果需要，请按照节 23.2.3.1.1 所述配置输入捕获设置。
7. 通过设置 EN = 1 使计数器开始计数。

使用具有 2 个输入信号的 QEI 模式的示例

PHA/PHB 的行为遵循表 23-5 和图 23-19。

表 23-5. PHA/PhB 状态表和计数器操作

当前状态（PHA、PHB）	下一个状态（PHA、PHB）	方向 (DIR)	计数器操作
00	10	1（上）	+1（如果 TIMx.CTR = LOAD，则 CTR = 0）
10	11		
11	01		
01	00		
00	01	0（下）	-1（如果 TIMx.CTR = 0，则 CTR = TIMx.LOAD）
01	11		
11	10		
10	00		

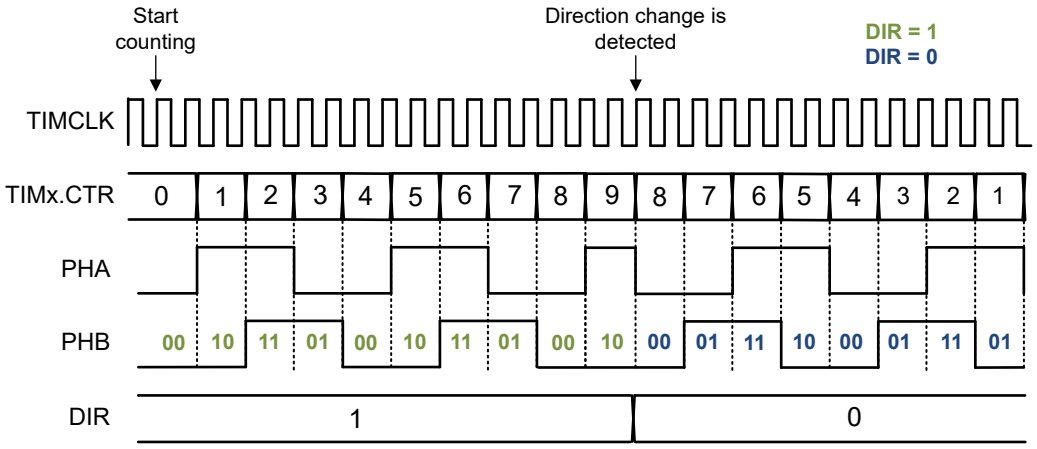


图 23-19. 2 信号 QEI 运行

23.2.3.1.3.2 具有索引输入的 QEI

3 信号 QEI 模式与具有额外索引输入信号 IDX 的 2 信号模式类似。通常，每旋转一次 IDX 输入产生一次脉冲，可用于重置计数器。这可用于两个应用之一：

- 每个移动周期都会生成一个 IDX 脉冲。在这种情况下，累积位置代表移动周期的比例。
- IDX 脉冲是在非循环移动中的特定位置生成的。

QEI 3 信号模式配置

1. 为 TIMGx_C0 (PHA)、TIMGx_C1 (PHB) 和 TIMGx_IDX (IDX) 配置 PINCMx。
2. 对于 CCP 通道 0 和 1（PHA 和 PHB）的捕获模式，设置 TIMG.CCCTL_01[0].COC = 1 和 TIMG.CCCTL_01[1].COC = 1。
3. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为每个 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
4. 将 TIMx.LOAD 值设置为编码器分辨率，例如 4000。
5. 在 CTRCTL 寄存器中，将 CAC、CZC 和 CLC 位设置为 5h。
6. 如果需要，请按照节 23.2.3.1.1 所述配置输入捕获设置。
7. 通过设置 EN = 1 使计数器开始计数。

当检测到上升沿时，会对 IDX 输入进行采样。IDX 信号会根据方向来影响计数器值，如表 23-6 所示。

表 23-6. IDX 输入值与计数器值的关系

方向 (DIR)	IDX	计数器操作
1	上升	零（TIMx.CTR 设置为零）
0	上升	负载（TIMx.CTR 设置为 TIMx.LOAD）

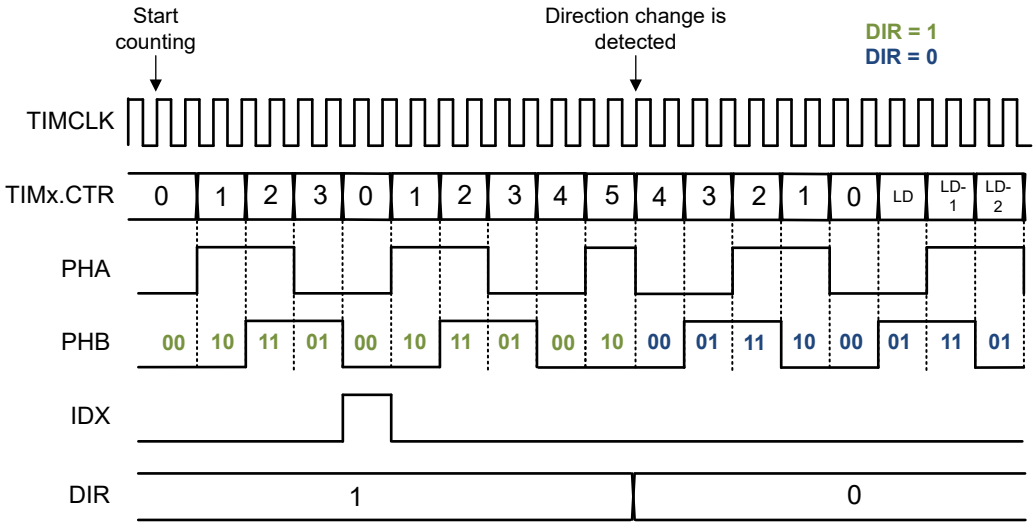


图 23-20. 具有索引输入操作的 2 信号 QEI

23.2.3.1.3.3 QEI 错误检测

QEI 模块可以检测错误事务或状态错误，如图 23-18 所示。会生成 QEIERR 中断，并且计数器或方向信号在错误状态下不会改变。

备注

如果 TIMCLK 周期长于 PHA 或 PHB 信号的周期，则可能发生 QEIERR。

23.2.3.1.4 霍尔输入模式 (仅限支持 QEI 的 TIMG)

在支持 QEI 的 TIMGx 实例中，三个数字霍尔信号可以输入到 CCP 通道 0 (CCP0)、CCP 通道 1 (CCP1) 和 IDX，用于三相霍尔传感器电机应用的位置控制。霍尔信号用于检测电机控制应用中的实时电机位置，并可用于速度计算测量、位置控制或电机失速状态。

备注

有关支持 QEI/霍尔输入模式的 TIMG 实例，请参阅节 23.1.3 和特定于器件的数据表。

表 23-7 展示了霍尔信号 A (U)、B (V) 和 C (W) 到 TIMG 捕获/比较输入信号的信号映射。

表 23-7. 霍尔输入和 TIMx 输入信号映射

霍尔输入信号	TIMx 输入
霍尔 A/霍尔 U	CCP0
霍尔 B/霍尔 V	CCP1
霍尔 C/霍尔 W	IDX

备注

霍尔输入信号应该是来自霍尔传感器 IC 的数字输入，具有连接至 VCC 的上拉电阻器。

如图 23-21 所示，输入捕获模块提供同步 CCP0、CCP1 和 IDX 信号的 3 输入异或，来创建频率发生器 (FG) 信号。当 IFCTL_xy[0/1].ISEL 设置为 4h 时选择异或输出信号。有关输入捕获方框图中的异或选项，请参阅图 23-21。

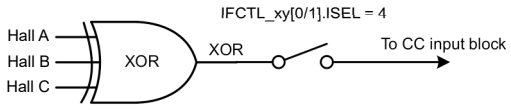


图 23-21. 频率发生器 (FG) 信号的霍尔 3 输入异或，连接到 CC 输入块

异或输出信号传播到 CC 块，周期或脉冲宽度捕获可用于计算与 TIMx.CC 寄存器中计算的周期或脉冲宽度相关的线性电机转速。请参阅节 23.2.3.1.2.2 和节 23.2.3.1.2.3，了解如何根据经过异或运算的输入信号计算周期和脉冲宽度捕获。

图 23-22 展示了可用于转速计算的 CC 块输入信号。

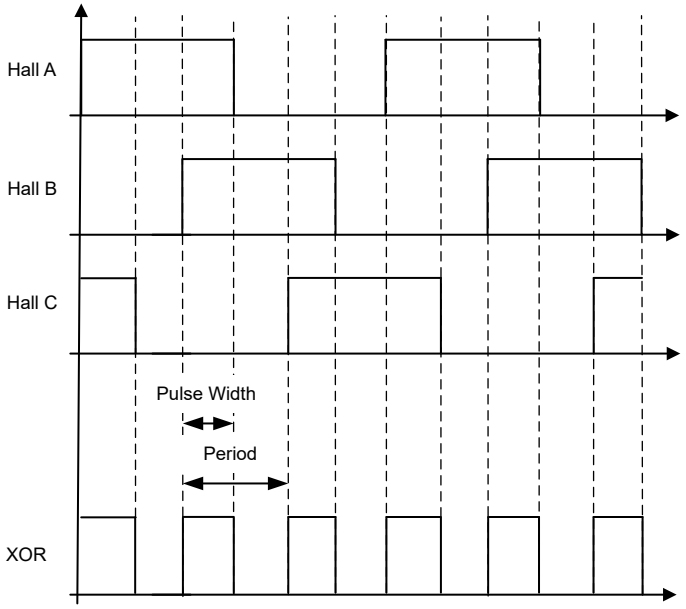


图 23-22. 与脉冲宽度或周期捕获一起用于转速计算的霍尔 3 输入异或信号

霍尔输入模式配置

1. 为 TIMGx_C0 (HALLA)、TIMGx_C1 (HALLB) 和 TIMGx_IDX (HALLC) 配置 PINCMx。
2. 对于 CCP 通道 0 和 1（HALLA 和 HALLB）的捕获模式，设置 TIMG.CCCTL_01[0].COC = 1 和 TIMG.CCCTL_01[1].COC = 1。
3. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为每个 CC 块的输入。例如，如果 TIMx 通道 0 是输入，则设置 CCPD.C0CCP0 = 0。
4. 设置 TIMx.LOAD 值。
5. 设置 TIMG.IFCTL_xy[0/1].ISEL = 4h 以选择霍尔信号的异或选项。
6. 通过设置 EN = 1 使计数器开始计数。

23.2.3.2 比较模式

当 TIMx.CCCTL_xy[0/1].COC = 0 时，选择比较模式。比较模式用于在特定时间间隔生成事件或 PWM 输出信号。

可以根据 CC 操作控制寄存器 CCACT_xy[0/1] 的配置生成多种类型的比较模式事件。当 TIMx.CTR 向上计数 (CCU) 或向下计数 (CCD) 到 TIMx.CC_xy[0/1] 中的值时，CC 通道会发生比较事件。此外，CC 通道的辅助比较递增 (CC2U) 或比较递减 (CC2D) 事件可以根据同一计时器实例中的 TIMx.CC_xy[0/1] 值从另一个 CC 块的 CCU 或 CCD 事件生成。比较事件可用于在内部生成时序基准，或使用活动、非活动或切换操作行为通过特定的配置文件生成 PWM 输出。

备注

向上/向下计数模式下的辅助比较事件可用于生成具有两倍 PWM 频率（但 PWM 分辨率减半）的 PWM 信号。

仅在 TIMA 中，额外的内部第 5 个和第 6 个 CC 块 (TIMA.CC_45[0/1]) 可用于辅助比较事件，同时继续使用具有专用输出引脚的 CC 通道来生成外部 PWM 信号。

表 23-8 展示了可以生成的比较模式事件的类型和生成事件的条件。

表 23-8. 比较模式事件

事件	名称	事件条件
CCDn（n = CC 通道）	捕获/比较递减事件	当计时器向下计数时，TIMx.CTR = TIMx.CC_xy[0/1]
CCUn（n = CC 通道）	捕获/比较递增事件	当计时器向上计数时，TIMx.CTR = TIMx.CC_xy[0/1]
CC2Dn（n = CC 通道）	次级捕获/比较递减事件	当 CC2SELD 配置为 CCDn 事件源时，会发生该事件
CC2Un（n = CC 通道）	次级捕获/比较递增事件	当 CC2SELU 配置为 CCUn 事件源时，会发生该事件

备注

查看特定于器件的数据表以检查器件上每个 TIMx 实例中有多少个 CC 通道可用。

23.2.3.2.1 边沿计数

除了生成事件或 PWM 输出之外，比较模式还可用于输入信号边沿计数，以便确定何时检测到多个边沿。在边沿计数操作中，一个 CCP 输入边沿可根据 ACOND 条件使计数器前进。计数器寄存器通过起始值初始化，检测到的 CCP 输入边沿的数量可以随时根据计数模式配置递增或递减。用户可以通过配置 CCOND 值来对上升沿、下降沿或两个边沿进行计数。

边沿计数配置

1. 对于比较模式，设置 TIMx.CCCTL_xy[0/1].COC = 0。
2. 可以选择将相应的 TIMx.CC_xy[0/1] 设置为比较值，以便在计数器达到该值时生成比较中断。
3. 在 CTRCTL 寄存器中，为以下各项设置所需的计数器控制设置：
   a. 计数模式 (CM) 和启用后计数器值 (CVAE)（请参阅节 23.2.2）
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
4. 将 ACOND 设置为根据输入边沿极性使计数器前进的设置。
5. 如果需要，请按照节 23.2.3.1.1 所述配置输入捕获设置。
6. 通过设置 EN = 1 来启用计数器。

向上计数模式下使用边沿计数操作示例

图 23-23 展示了从零开始计数的向上计数模式下（CM = 2，CVAE = 2），使计数器递增的上升沿计数操作的预期内部时序。

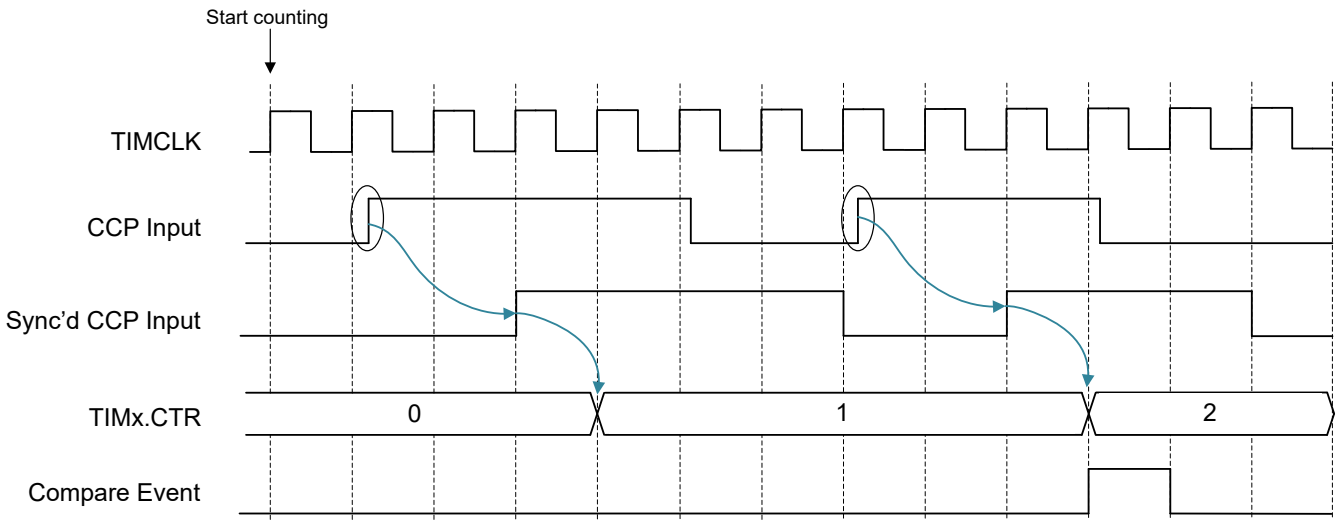


图 23-23. 生成比较事件的边沿计数操作 (TIMx.CC = 2)

23.2.4 影子加载和影子比较

一些计时器模块具有影子加载和影子比较寄存器功能，使用户可以灵活地保持加载和 CC 值的更新，直到某个事件发生。这在时序关键型应用中非常有用，在这些应用中，PWM 控制信号需要使用正确的时序进行更新，例如占空比更新。有关 TIMx 模块的具体配置，请参阅计时器功能。

备注

有关支持影子加载和影子比较的 TIMx 实例，请参阅节 23.1.3 和特定于器件的数据表。

23.2.4.1 影子负载

利用影子加载功能，可以保持加载值的更新，直到发生归零事件。要启用影子加载，请设置 TIMx.GCTL.SHDWLDEN 位。

如果 TIMx 模块具有影子加载功能，则存在一个用于加载值的内部影子寄存器 (TIMx.LOAD)。该影子寄存器将在发生归零事件时更新加载值，如图 23-24 所示。

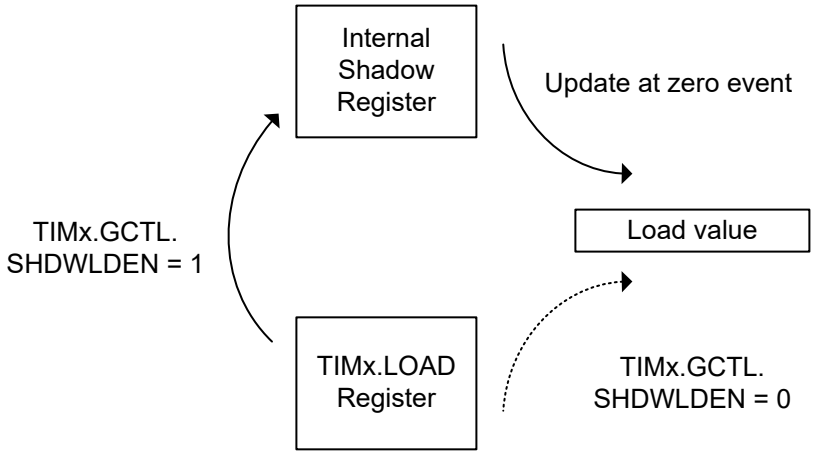


图 23-24. 影子加载更新加载值

当 TIMx.GCTL.SHDWLDEN = 1 时，对于所有计数模式，加载值都会在发生归零事件时更新。请参阅下面的计数模式操作以确定是否需要影子加载：

- 在向下计数模式下，由于 TIMx.LOAD 值在发生归零事件时更新，因此在这些模式下不需要影子加载。

- 在向上/向下计数模式下，TIMx.LOAD 与计数器值进行比较以确定是否达到峰值以及何时开始向下计数。影子加载是必要的，可确保 TIMx 在发生归零事件之前计数到加载值，否则加载值可能会立即更新并导致不正确的时序。
- 在向上计数模式下，计时器计数到 TIMx.LOAD。影子加载是必要的，可确保 TIMx 在发生归零事件之前计数到加载值，否则加载值可能会立即更新并导致不正确的时序。

图 23-25 展示了影子加载和影子比较如何在向上/向下计数模式下针对 TIMx.LOAD 和 TIMx.CC 值在发生归零事件时生效的示例。

23.2.4.2 影子比较

当启用影子比较来更新捕获/比较寄存器 (TIMx.CC) 时，写入相应比较寄存器的值将首先存储到影子比较寄存器中，然后在发生通过设置 TIMx.CCCTL_xy[0/1].CCUPD 位配置的不同事件时传输到比较寄存器。

此外，捕获/比较操作寄存器 (TIMx.CCACT) 能够在发生通过设置 TIMx.CCCTL_xy[0/1].CCACTUPD 位配置的不同事件时更新操作。

表 23-9 展示了用于配置在发生不同事件时影子比较和操作何时发生的设置。

表 23-9. 影子比较和操作更新行为

位字段	值	说明/注释
CCUPD /CCACTUPD	0	写入 TIMx.CC 寄存器的值立即生效。
	1	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在发生归零事件后的 TIMCLK 周期内传输到 TIMx.CC 寄存器（TIMx.CTR 值等于 0）。
	2	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在发生比较（递减）事件后的 TIMCLK 周期内传输到 TIMx.CC 寄存器（TIMx.CTR 值等于 TIMx.CC）
	3	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在发生比较（递增）事件后的 TIMCLK 周期内传输到 TIMx.CC 寄存器（TIMx.CTR 值等于 TIMx.CC）。
	4	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在发生归零事件或加载事件后的 TIMCLK 周期内传输到 TIMx.CC 寄存器（TIMx.CTR 值等于 0 或 TIMx.CTR 等于 TIMx.LOAD）。 **注意：此更新机制仅用于向上/向下计数模式。**
	5	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在发生归零事件且重复计数等于零后的 TIMCLK 周期内传输到 TIMx.CC 寄存器（TIMx.CTR 值等于 0 且 TIMx.RC 等于 0）。
	6	写入 TIMx.CC 寄存器的值存储在影子比较寄存器中，并在触发脉冲后的 TIMCLK 周期内传输到 TIMx.CC 寄存器。 请参阅节 23.2.7

图 23-25 展示了影子加载和影子比较如何在向上/向下计数模式下针对 TIMx.LOAD 和 TIMx.CC 值在发生归零事件时生效的示例。

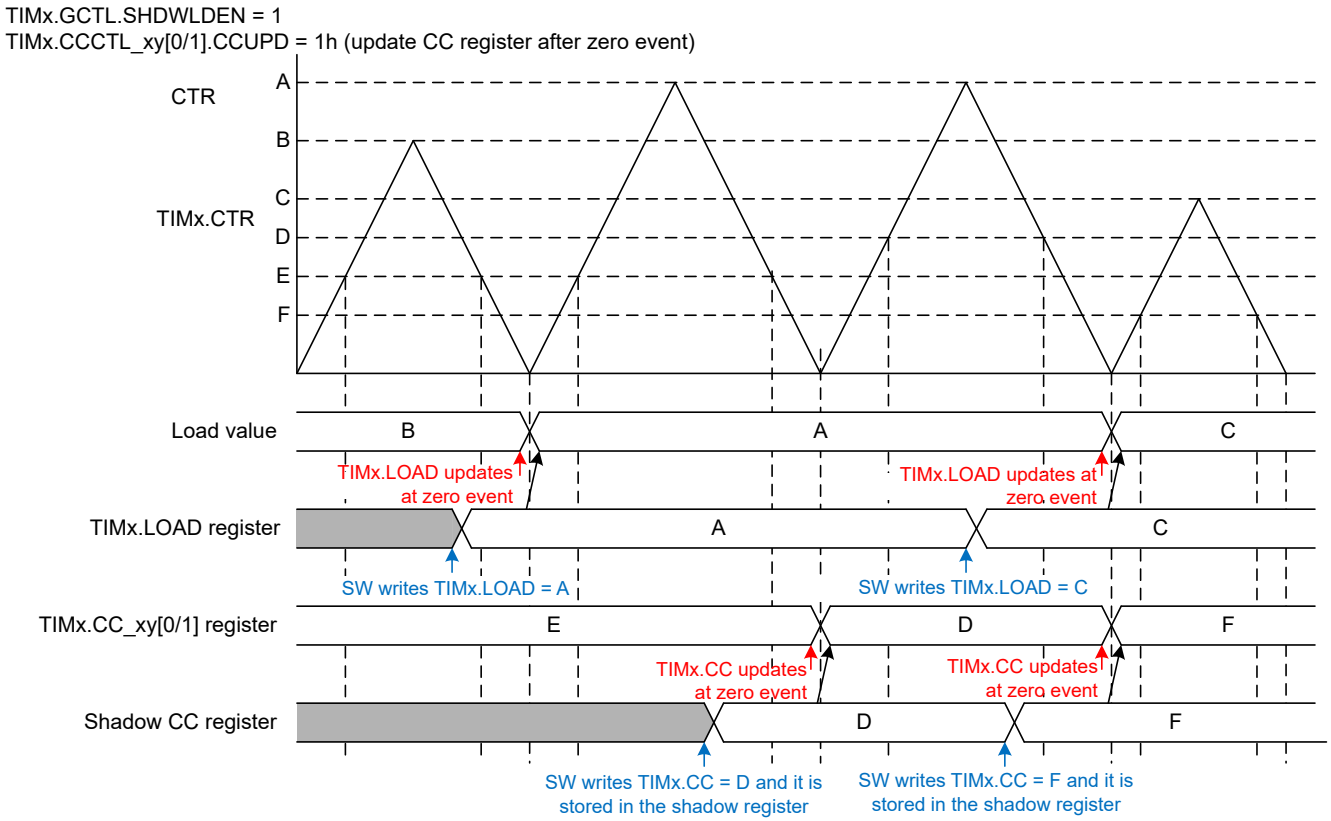


图 23-25. 在递增/递减模式下，影子加载和影子比较在发生归零事件时生效

23.2.5 输出发生器

输出信号生成单元可与计数器和捕获/比较模块一起使用，以生成所需的脉宽调制 (PWM) 输出波形、事件信号、同步捕获输入或计数器方向。许多输出波形是由计数器事件（加载、归零、计数器方向）和捕获/比较块（比较匹配）生成的。

TIMA 和 TIMG 在输出生成信号单元上有很多共同的特性。此外，TIMA 还具有高级输出生成特性，例如互补输出信号、死区插入和故障生成。

图 23-27 展示了 TIMG 输出方框图。

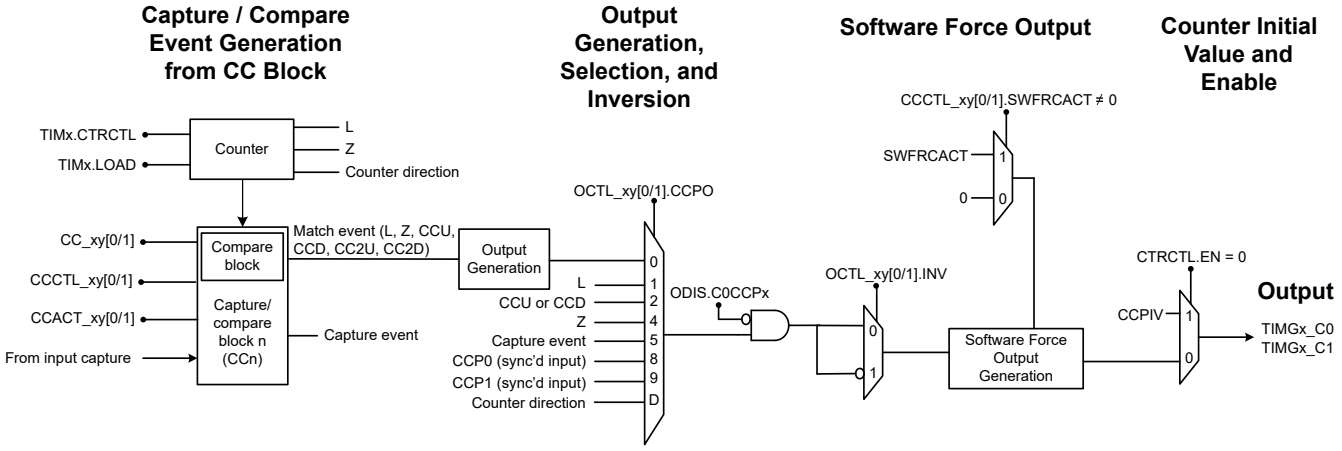


图 23-26. TIMG 的输出连接

图 23-27 展示了 TIMA 输出方框图。

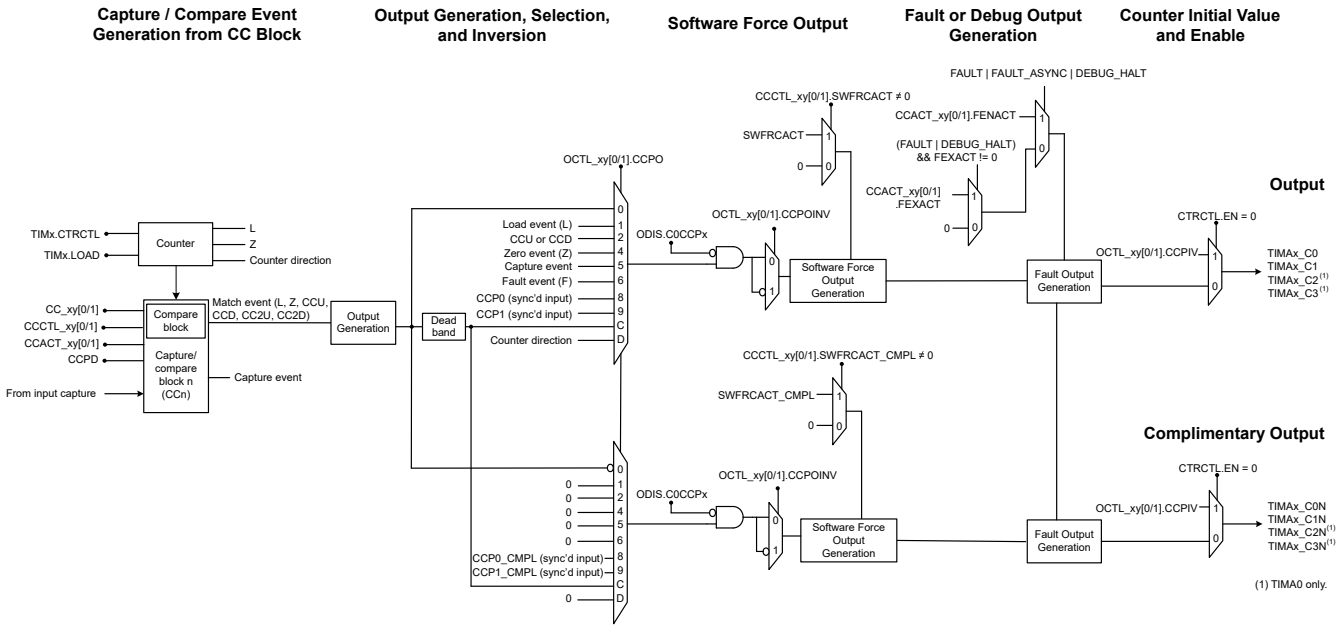


图 23-27. TIMA 的输出连接

信号发生器操作

表 23-10 展示了输出发生器能够执行的信号发生器操作类型。信号发生器操作在 CCACT_xy[0/1] 寄存器中配置为归零、加载和比较事件。有关比较事件的类型，请参阅表 23-8。

表 23-10. 来自比较事件的信号发生器操作

值	操作
0h	事件被禁用并选择优先级较低的事件（如果生效）
1h	CCP 输出值设置为高电平
2h	CCP 输出值设置为低电平
3h	CCP 输出值被切换

用于生成输出信号的主要寄存器为：

- **LOAD**：在发生指定执行“加载”的任何操作时，此寄存器的内容都会复制到计数器 (TIMx.CTR)。该值还用于与计数器值进行比较，以便生成可用于中断、触发或信号发生器操作的“加载事件”。
- **CCPD**：此寄存器将 CCP 引脚的方向配置为输入或输出。
- **CC_xy[0/1]**：此寄存器用作与当前计数器进行比较的值，以创建匹配事件。
- **CTRCTL**：此寄存器用于控制不同条件下的计数器操作。
- **CCCTL_xy[0/1]**：此寄存器用于控制相应 CC 寄存器和计数器的操作。
- **OCTL_xy[0/1]**：此寄存器用于控制计数器的捕获/比较部分的输出。还能够选择驱动源以及初始条件值和最终反转选项。
- **CCACT_xy[0/1]**：此寄存器根据在计数器块、捕获和比较块以及调试事件中创建的事件来控制捕获/比较部分的信号发生器的操作。
- **ODIS**：该寄存器禁用 OCTL.CCPO 选择的输出信号（在条件反转之前），以允许软件在配置或关断期间将 CCP 输出保持为低电平。

这些是用于配置比较模式以生成 PWM 信号的主要寄存器。

23.2.5.1 配置

在 TIMx 器件中配置输出信号生成分为五个阶段：

- 计数器和 CC 块事件生成
- 输出生成、选择和反转
- 软件强制输出
- 故障输出生成（仅限 TIMA）
- 计数器初始值和使能

计数器和 CC 块事件生成

计数器块包含计数器，可根据所使用的计数模式产生加载事件 (L)、归零事件 (Z) 和计数方向。

CC 块包含 CC 寄存器，可生成两种输出信号：比较匹配事件和捕获事件。有关可生成的比较事件，请参阅表 23-8。

输出生成、选择和反转

TIMx.CCACT 寄存器根据计数模式和计数器比较操作来指定 CCP 输出的波形生成。

TIMx.OCTL_xy[0/1].CCPO 可控制输出生成单元、具有死区的输出生成单元（仅限 TIMA）、计数器事件、比较事件、捕获事件、故障事件或信号输入的 CCP 输出选择。输出禁用寄存器 (ODIS) 可以选择禁用 CCP 输出，以便视情况在配置或关断期间将 CCP 输出保持在低电平。TIMx.OCTL_xy[0/1].INV 控制最终反转选项。

备注

对于 TIMAx CC2 和 CC3 实例，同步输入的多路复用器选择绑定到 0。请勿使用 TIMAx.OCTL_23[0/1].CCPO = 8 或 9。

（仅在 TIMA 器件上）CCP 互补输出通道可由输出生成单元生成（在信号名称中用“N”表示）。例如，TIMA0 通道 2 (TIMA0_C2) 也可以产生互补输出 (TIMA0_C2N)。CCPO 和 INV 位还控制互补输出的选择和反转选项。

具有死区插入功能的互补输出是基于逆变器且采用半桥拓扑的应用的常见用例。有关更多信息，请参阅节 23.2.5.2.4。

软件强制输出

通过将 CCCTL_xy[0/1].SWFRACT 设置为非零值，可在软件中覆盖信号发生器的输出。（仅限 TIMA 器件）通过将 CCCTL_xy[0/1].SWFRACT_CMPL 设置为非零值，可以覆盖信号发生器的互补输出。

有关更多信息，请参阅节 23.2.5.3。

故障/调试输出生成（仅限 TIMA）

在 TIMA 器件上，如果存在系统故障 (FAULT)、退出时的故障条件 (FEXACT)、进入时的故障条件 (FENACT)、异步故障 (FAULT_ASYNC) 或调试器已停止 (DEBUG_HALT)，则在软件强制输出块之后，CCP 输出可以被覆盖。

有关更多信息，请参阅节 23.2.6 和节 23.2.10。

计数器比较初始值和使能

要在计数器被禁用时为 CCP 输出指定初始值，请将 OCTL_xy[0/1].CCPIV 的低值设定为 0，或将高值设定为 1。这对于在启用计数器之前需要将 CCP 输出置于默认状态的应用非常有用。

要启用计数器，请将 TIMx.CTRCTL.EN 设置为 1。

23.2.5.2 用例

使用输出发生器可以实现多种不同的用例，后续章节中将对此进行讨论。

23.2.5.2.1 边沿对齐的 PWM

要生成边沿对齐 PWM，可以将 TIMx 配置为递增或向下计数模式，TIMCLK 周期中的 PWM 周期为 TIMx.LOAD + 1。波形使用加载、归零和比较事件将 CCPx 输出驱动为高电平或低电平，具体取决于比较/捕获块和计数器的配置设置。

边沿对齐 PWM 配置

要使用计数器的比较匹配事件生成中心对齐 PWM，请执行以下操作：

1. 在 TIMx.CTRCTL 寄存器中，为以下各项配置所需的计数器控制设置：
   a. 向上计数 (CM = 2) 或向下计数模式 (CM = 0) 和使能后的计数器值 (CVAE) (请参阅节 23.2.2)
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
2. 设置 TIMx.LOAD 值以配置 PWM 周期。
3. 设置 TIMx.CC_xy[0/1] 值以配置占空比。
4. 针对比较模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
5. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为 CC 块的输出。例如，如果 TIMx 通道 0 是输出，则设置 CCPD.C0CCP0 = 1。
6. 在 TIMx.CCACT_xy[0/1] 中，设置比较事件、归零事件、加载事件、软件强制操作或故障事件的 CCP 输出操作设置 (仅限 TIMA) 。
7. 在 TIMx.OCTL_xy[0/1] 中，设置 CCPO = 0 以选择信号发生器输出。
8. 通过针对相应计数器 n 将 ODIS.C0CCPn 设置为 1 来启用相应的 CCP 输出。
9. 使用 CCPOINV 位配置信号的极性，并配置 CCPIV 以指定禁用时的 CCP 输出状态。
10. 通过设置 TIMx.CTRCTL.EN = 1 来启用计数器。

在向下计数模式下使用边沿对齐 PWM 的示例

图 23-28 展示了向下计数模式下的典型 2 通道边沿对齐 PWM 生成，其中具有以下边沿对齐 PWM 输出波形：

- CCP0 输出产生：
  - 从 TIMx.LOAD 到 TIMx.CC0 值的高脉冲宽度 (LACT = 1h)
  - 从 TIMx.CC0 值到零的低脉冲宽度 (CDACT = 2h)
- CCP1 输出产生：
  - 从 TIMx.LOAD 到 TIMx.CC1 值的高脉冲宽度 (LACT = 1h)
  - 从 TIMx.CC1 值到零的低脉冲宽度 (CDACT = 2h)

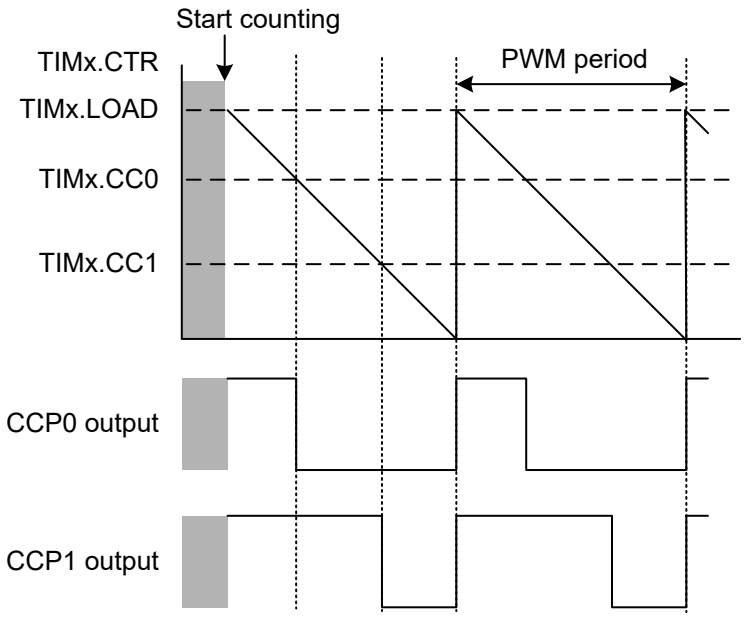


图 23-28. 向下计数模式下的边沿对齐 PWM 信号

在向上计数模式下使用边沿对齐 PWM 的示例

图 23-28 展示了向上计数模式下典型的 2 通道边沿对齐 PWM 生成，其中具有以下边沿对齐 PWM 输出波形：

- CCP0 输出产生：

  - 从零到 TIMx.CC0 值的高脉冲宽度 (ZACT = 1h)
  - 从 TIMx.CC0 值到 TIMx.LOAD 的低脉冲宽度 (CUACT = 2h)
- CCP1 输出产生：
  - 从零到 TIMx.CC1 值的高脉冲宽度 (ZACT = 1h)
  - 从 TIMx.CC1 值到 TIMx.LOAD 的低脉冲宽度 (CUACT = 2h)

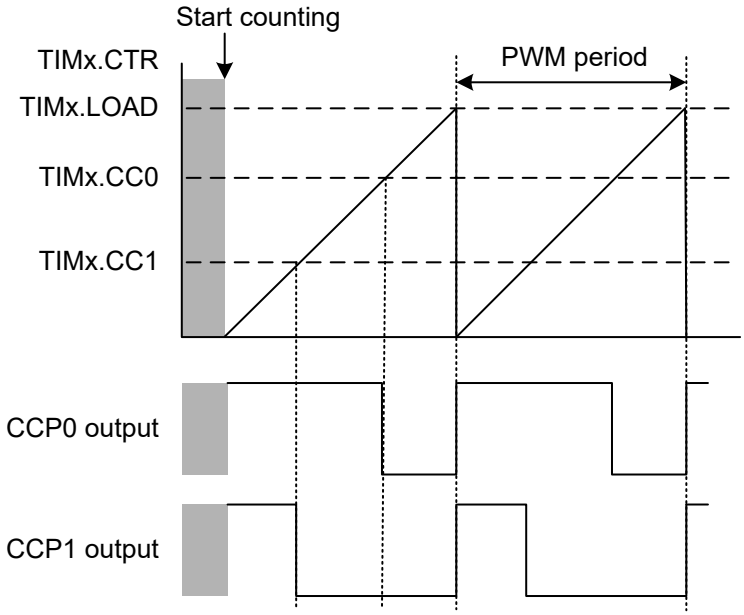


图 23-29. 向上计数模式下的边沿对齐 PWM 信号

23.2.5.2.2 中心对齐 PWM

为了生成中心对齐的 PWM，TIMx 配置为向上/向下计数模式，TIMx.LOAD 值包含半个周期。波形使用递增比较事件和递减比较事件将 CCPx 输出驱动为高电平或低电平，具体取决于比较/捕获块和计数器的配置设置。

在 TIMCLK 周期中，PWM 周期为 (2 * TIMx.LOAD)，占空比为 1 - (TIMx.CC_xy[0/1]/TIMx.LOAD)。

中心对齐 PWM 配置

要使用计数器的比较匹配事件生成中心对齐 PWM，请执行以下操作：

1. 在 TIMx.CTRCTL 寄存器中，为以下各项配置所需的计数器控制设置：
   a. 向上/向下计数模式 (CM = 1) 和启用后计数器值 (CVAE) (请参阅节 23.2.2)
   b. 清零 (CZC)、前进 (CAC) 和加载控制 (CLC)，用于指定控制计数器清零、前进或加载的条件
   c. 重复或单次触发模式 (REPEAT)
2. 设置 TIMx.LOAD 值以配置 PWM 周期。
3. 设置 TIMx.CC_xy[0/1] 值以配置占空比。
4. 针对比较模式设置 TIMx.CCCTL_xy[0/1].COC = 1。
5. 通过设置 CCPD 寄存器中的相应位，将 CCP 配置为 CC 块的输出。例如，如果 TIMx 通道 0 是输出，则设置 CCPD.C0CCP0 = 1。
6. 在 TIMx.CCACT_xy[0/1] 中，设置比较事件、归零事件、加载事件、软件强制操作或故障事件的 CCP 输出操作设置 (仅限 TIMA) 。
7. 在 TIMx.OCTL_xy[0/1] 中，设置 CCPO = 0 以选择信号发生器输出。
8. 通过针对相应计数器 n 将 ODIS.C0CCPn 设置为 1 来启用相应的 CCP 输出。
9. 使用 CCPOINV 位配置信号的极性，并配置 CCPIV 以指定禁用时的 CCP 输出状态。
10. 通过设置 TIMx.CTRCTL.EN = 1 来启用计数器。

在向上/向下计数模式下使用中心对齐 PWM 的示例

使用向上/向下计数模式的典型 2 通道中心对齐 PWM 生成如图 23-30 所示，其中具有以下中心对齐 PWM 输出波形：

- CCP0 输出产生：
  - 从 TIMx.CC0 比较递增事件到 TIMx.CC0 比较递减事件的高脉冲宽度 (CUACT = 1h)
  - 从 TIMx.CC0 比较递减事件到 TIMx.CC0 比较递增事件的低脉冲宽度 (CDACT = 2h)
- CCP1 输出产生：
  - 从 TIMx.CC0 比较递增事件到 TIMx.CC0 比较递减事件的高脉冲宽度 (CUACT = 1h)
  - 从 TIMx.CC0 比较递减事件到 TIMx.CC0 比较递增事件的低脉冲宽度 (CDACT = 2h)

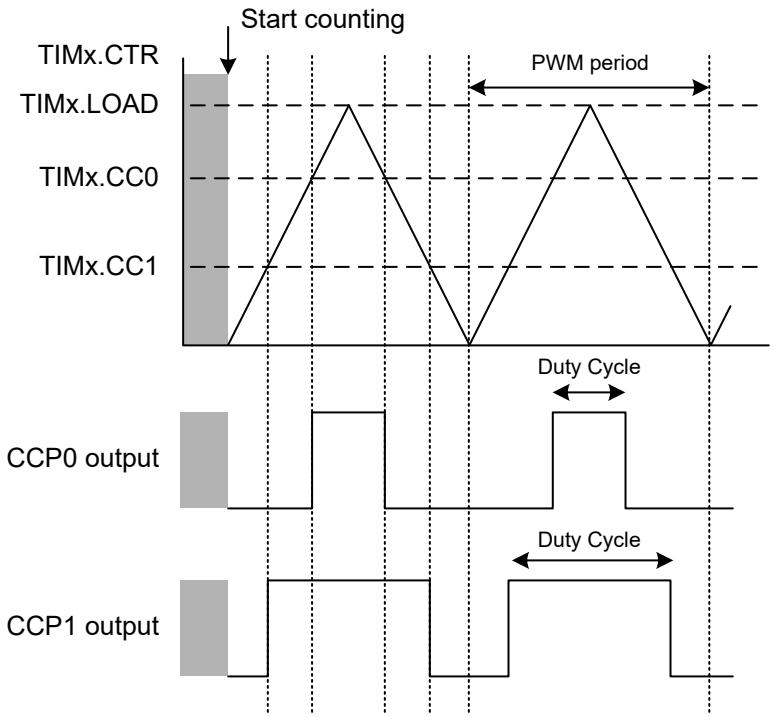


图 23-30. 中心对齐 PWM

23.2.5.2.3 非对称 PWM（仅限 TIMA）

仅在 TIMA 中，可以通过生成两个具有受控相移的同步中心对齐 PWM 信号来生成非对称 PWM。要生成非对称 PWM 信号，应按照节 23.2.5.5 中所述使用相位加载特性。

非对称 PWM 配置

要使用计数器的比较匹配事件生成非对称 PWM，请执行以下操作：

1. 使用交叉触发器同步 TIMA0 和 TIMA1，如节 23.2.7 所述。
2. 使用 TIMA0 和 TIMA1 配置两个中心对齐 PWM，如节 23.2.5.2.2 所述。TIMA0 和 TIMA1 应该具有相同的负载值 (TIMA.LOAD) 和比较值 (TIMA.CC_xy[0/1]) 来生成相同的 PWM 频率和占空比。
3. 通过配置相位加载值 TIMA.PL 为 TIMA0 或 TIMA1 添加相移值，如节 23.2.2.5 所述。
4. 通过设置 TIMA.CTRCTL.EN = 1 来启用计数器。

图 23-31 显示了使用 TIMA0 和 TIMA1 的 CCP 通道 0 的非对称 PWM 配置示例。

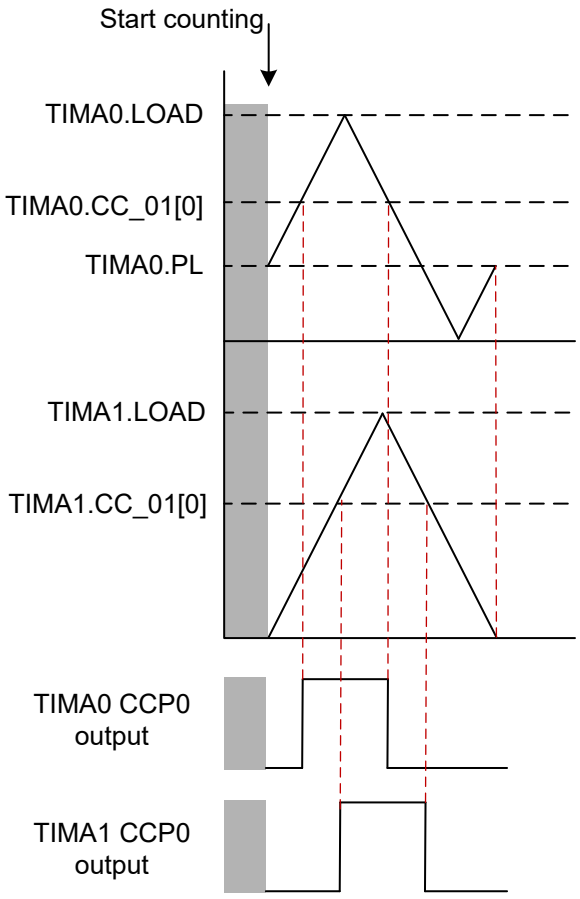


图 23-31. TIMA0 和 TIMA1 的 CCP 通道 0 具有相位加载的非对称 PWM 配置

23.2.5.2.4 具有死区插入的互补 PWM（仅限 TIMA）

TIMA 提供了从信号 PWM 基准信号生成具有死区插入（互补 PWM 信号中的非重叠转换）的互补 PWM 输出的选项。死区对于采用半桥控制的应用非常有用，可以避免击穿情况，例如基于电机驱动器或逆变器的应用。

TIMA 在互补 CCP 输出通道（例如 TIMA0_C2 和 TIMA0_C2N）上提供该可选功能，用于 TIMA0 CCP 输出通道 2 上的基准 PWM 信号。

使用死区模式和时序信息对死区控制寄存器 (TIMA.DBCTL) 进行编程。死区模式为模式 0 或模式 1，可以通过 M1_ENABLE 位进行选择，可以通过 RISEDELAY 和 FALLDELAY 位字段选择控制 TIMCLK 周期内死区宽度的时序信息。有关死区模式和死区宽度设置的配置和两者之间的关系，请参阅表 23-11。

表 23-11. DBCTL 寄存器中的死区模式和延迟时序配置

死区模式	位域	说明	计数模式
模式 0	M1_ENABLE = 0	RISEDELAY 和 FALLDELAY 应用于输出发生器信号的上升沿和下降沿，以生成 CCP 和 CCP 互补信号	不限
模式 1	M1_ENABLE = 1	• CCP 信号与输出发生器信号相同 • RISEDELAY 和 FALLDELAY 应用于 CCP 互补信号	仅限向上/向下计数模式

死区时序公式和示例

方程式 29 和方程式 30 显示了根据 TIMCLK 频率和死区时序配置 RISEDELAY 和 FALLDELAY 的公式。

$$RISEDELAY = f_{TIMCLK} \times t_{dead\_rise} \quad (29)$$

$$FALLDELAY = f_{TIMCLK} \times t_{dead\_fall} \quad (30)$$

例如，如果使用 80MHz 的 TIMCLK 频率时需要 400ns 的死区，并且模式 1 与中心对齐 PWM 一起使用以在每个 PWM 周期生成相等的死区，则 RISEDELAY = FALLDELAY = (80MHz) * (400ns) = 32。

具有死区配置的互补 PWM

1. 为 TIMA 中的任何 CCP 输出通道配置边沿对齐 PWM (节 23.2.5.2.1) 或中心对齐 PWM (节 23.2.5.2.2) 的 PWM 输出。
2. 使用指定的死区模式 (M1_ENABLE) 和死区宽度 RISEDELAY 和 FALLDELAY 配置 TIMA.DBCTL，具体取决于死区模式。
3. 在 TIMx.OCTL_xy[0/1] 中，设置 CCPO = 0xC 以选择具有死区输出的信号发生器。
4. 通过设置 TIMx.CTRCTL.EN = 1 来启用计数器。

示例 1 - 在向下计数模式下使用边沿对齐 PWM、具有死区的互补 PWM 输出

对于边沿对齐 PWM，模式 0 只能用于死区插入模式。有关如何使用向下计数模式、TIMA 输出通道 0 和边沿对齐 PWM 插入可配置死区的信息，请参阅图 23-32。对于中心对齐 PWM，有两种模式，如图 23-33 所示。

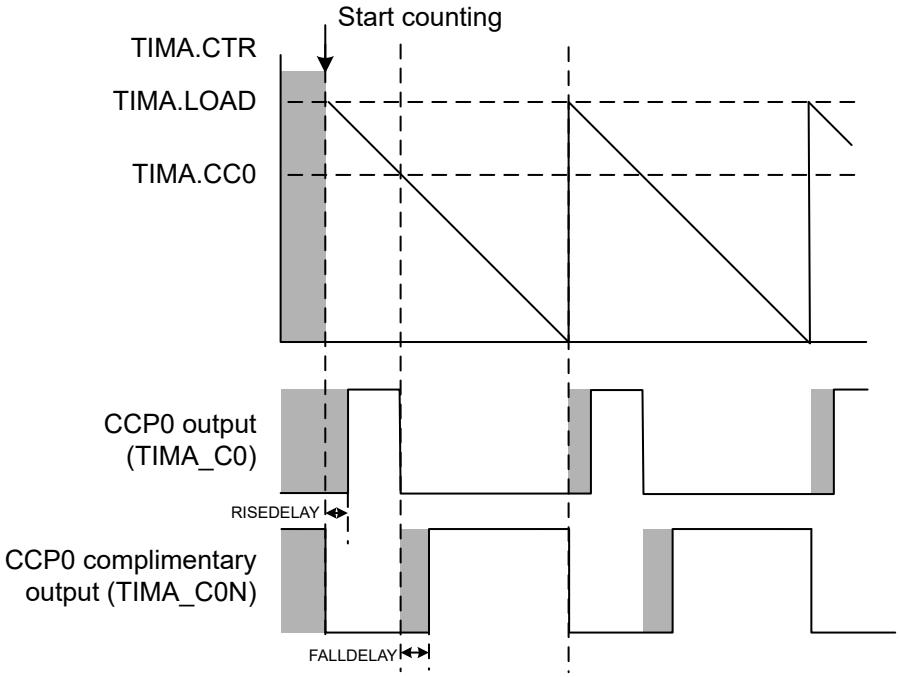


图 23-32. 使用 TIMA CC 输出通道 0 (M1_ENABLE = 0) 在向下计数模式下针对边沿对齐 PWM 插入死区

示例 2 - 使用中心对齐 PWM、具有死区的互补 PWM 输出

对于中心对齐 PWM，模式 0 或模式 1 可用于死区插入模式。有关如何针对模式 0 和模式 1 使用向上/向下计数模式、TIMA 输出通道 0 和中心对齐 PWM 插入可配置死区的信息，请参阅图 23-33。

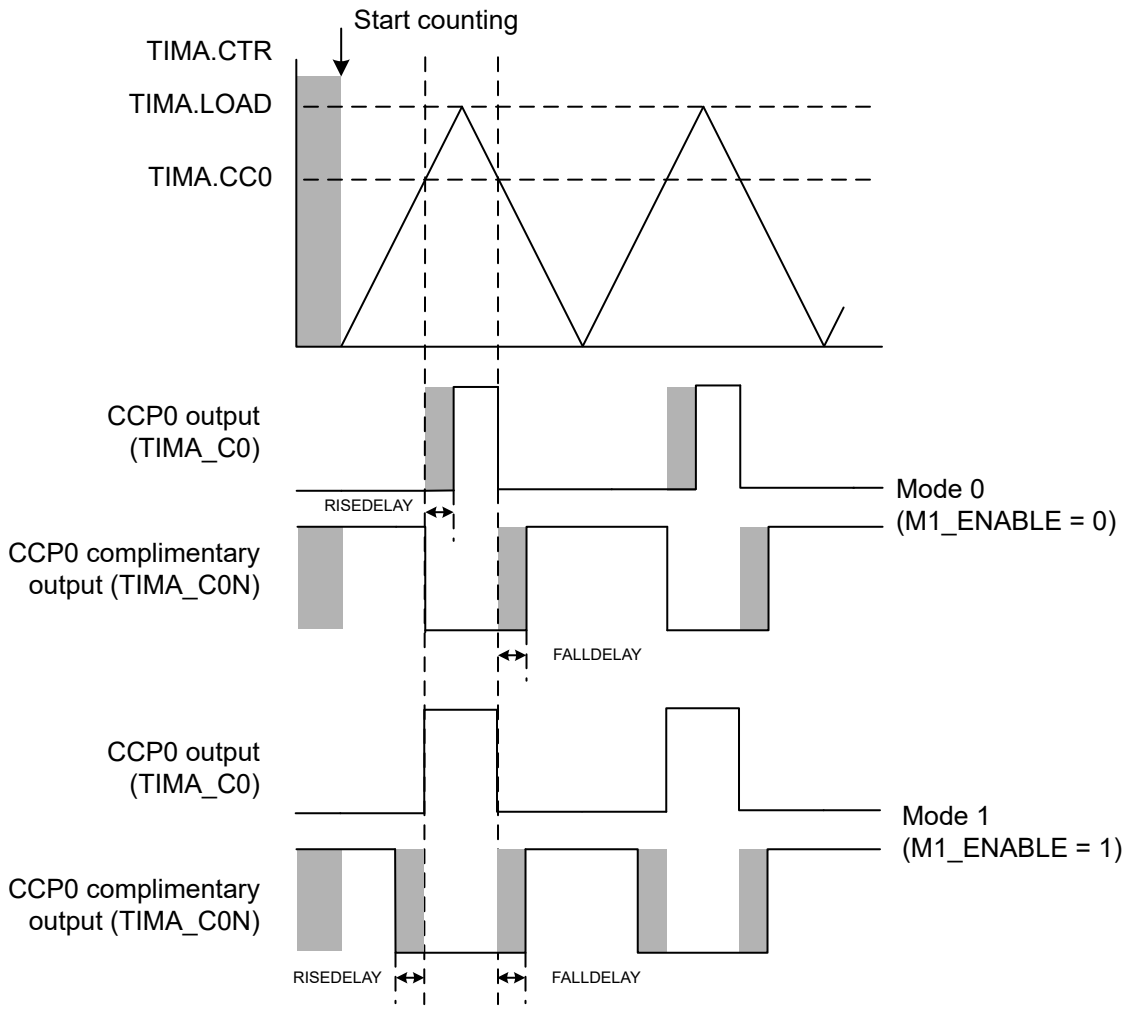


图 23-33. 中心对齐 PWM 的死区插入（模式 0 和模式 1）

23.2.5.3 强制输出

可以通过软件直接将每个输出通道信号强制设置为高电平或低电平，而与比较寄存器和计数器之间的任何比较无关。

通过设置 TIMx.CCACT_xy[0/1] 寄存器中的 SWFRCACT 位，可以将 CCP 通道的输出强制设置为高电平或低电平。

此外，仅在 TIMA 中，还可以通过设置 TIMx.CCACT_xy[0/1] 寄存器中的 SWFRCACT_CMPL 位将互补输出通道强制设置为高电平或低电平。

表 23-12 展示了软件强制输出操作配置选项。

表 23-12. 强制输出操作配置

位字段	值	说明/注释
SWFRCACT/SWFRCACT_CMPL	0	无强制输出。输出直接来自信号生成块。
	1	强制输出高电平
	2	强制输出低电平

23.2.6 故障处理程序（仅限 TIMA）

仅在 TIMA 中，存在可用于控制 PWM 信号生成的内部和外部故障输入。这些输入的预期用途是作为内部或外部电路指示系统中故障的一种机制。这使硬件能够快速响应外部故障，同时可选地发出中断信号以进行软件校正并使输出信号处于安全状态。

务必考虑系统中故障的以下基本属性，例如：

- 故障输入选择（来自外部 IC 的故障信号、内部信号等）
- 故障情况的持续时间
- 计数器如何对故障情况的进入和退出做出反应
- 输出信号如何对故障情况的进入和退出做出反应

可以使用 TIMCLK 同步检测故障情况，也可以异步检测故障情况。同步故障具有可配置的干扰滤波器，并且可以生成锁存故障事件。异步故障无法被锁存，并且不会生成故障事件。CCP 输出可根据进入和退出条件配置为任一类型的故障。

故障处理程序逻辑图分为三个部分：异步故障、同步故障和故障输出生成。

图 23-34 展示了异步故障处理程序逻辑连接。

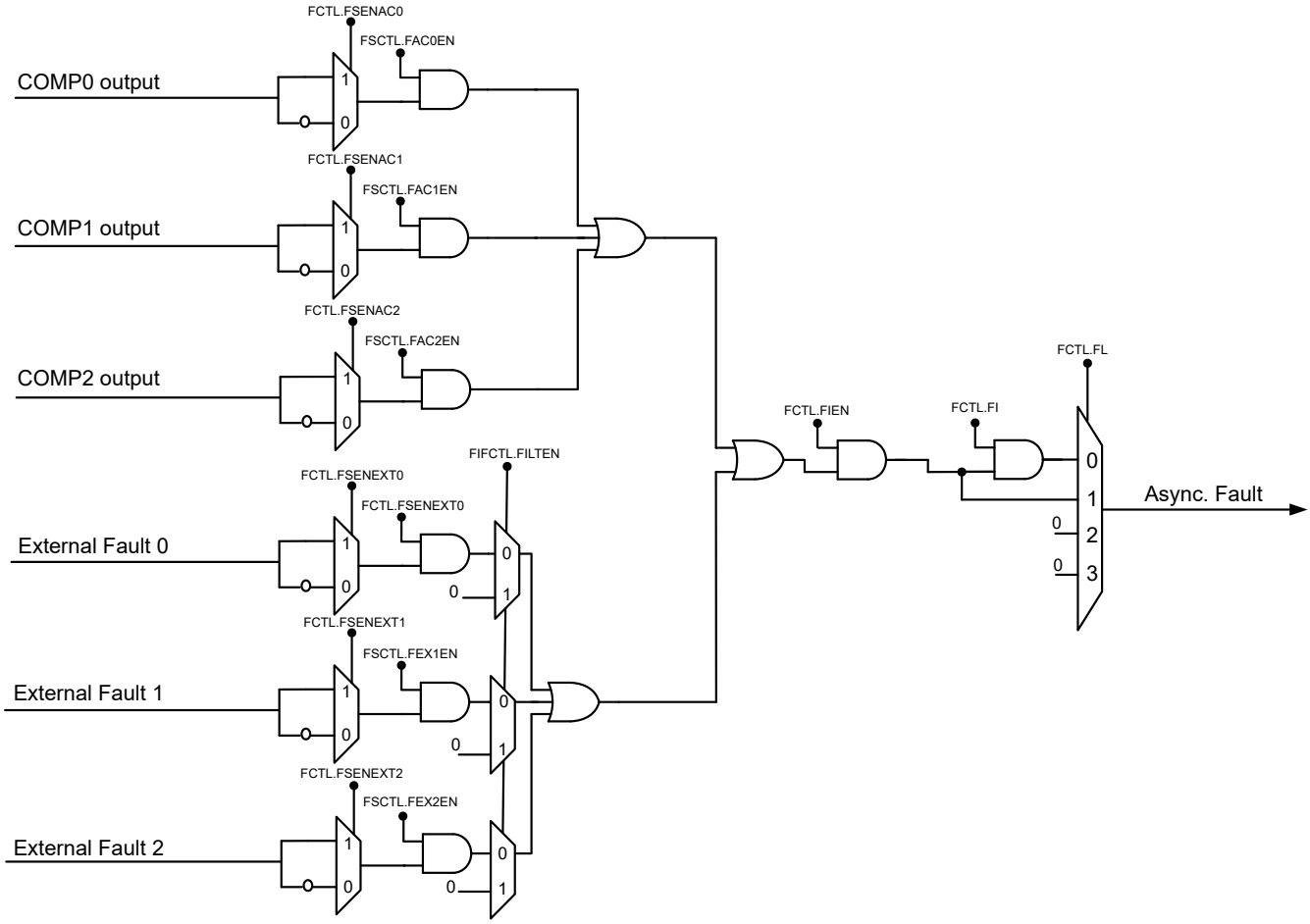


图 23-34. 异步故障处理程序连接

图 23-35 展示了同步故障处理程序逻辑连接。

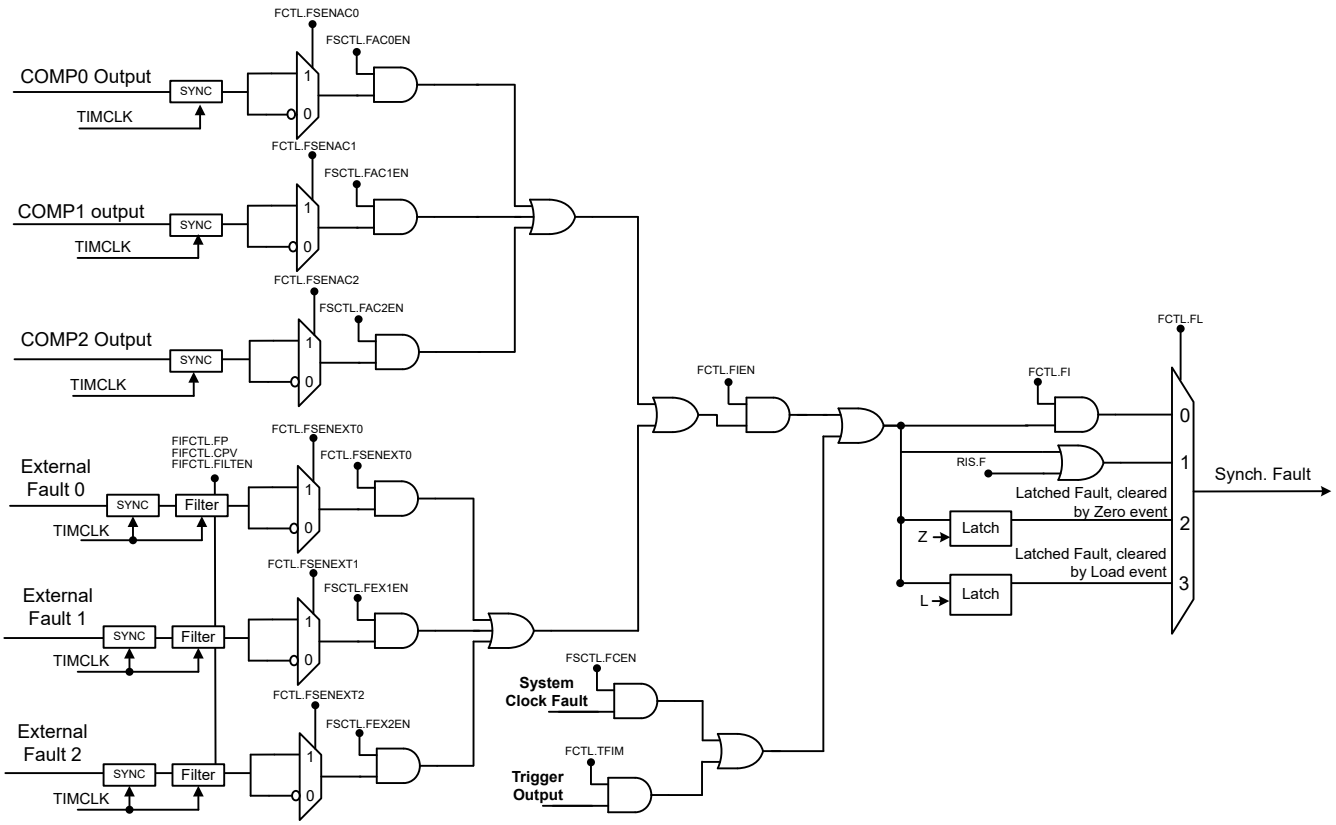


图 23-35. 同步故障处理程序连接

图 23-36 展示了故障输出生成逻辑连接。

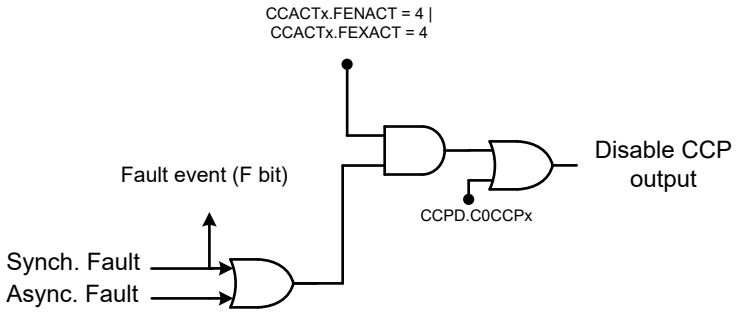


图 23-36. 故障输出生成连接

用于配置故障处理程序的主要寄存器包括：

- **TIMA.FCTL**：该寄存器控制故障输入、故障检测和错误处理行为。
- **TIMA.FSCTL**：该寄存器控制故障源的选择和使能。
- **TIMA.FIFCTL**：该寄存器控制故障输入的输入滤波（FILTEN、FP、CPV）。
- **TIMA.CCACT_xy[0/1]**：该寄存器根据故障事件控制捕获比较部分的信号发生器的操作。

23.2.6.1 故障输入调节

比较器和外部故障引脚故障源输入会经过两个 TIMCLK 同步级，可以使用故障输入滤波器 (TIMA.FIFCTL) 寄存器通过干扰滤波器对故障输入进行滤波。

可以通过设置 TIMA.FIFCTL.FILTEN 位来启用故障输入干扰滤波器。可以通过设置 TIMA.FIFCTL.FP 位来配置滤波器周期。

由 TIMA.FIFCTL.CPV 位选择的连续周期或多数表决格式用于选择 CCP 输入信号的标准。

- **连续周期** - 对于要处理的故障输入，故障输入信号必须在指定数量的 FP 计时器时钟内处于指定的电平。
- **多数表决** - 滤波器在滤波器周期内忽略一个相反逻辑的时钟。例如，在故障输入的 FP 样本数量中，最多 1 个样本可能具有相反的逻辑值（干扰）而不影响输出。

图 23-13 中所示的示例展示了连续周期和多数表决格式之间的差异，其中实现了一个数字滤波器以捕获 3 个 TIMCLK 周期的故障输入。

23.2.6.2 故障输入源

故障控制 (FCTL) 和故障源控制 (FSCTL) 寄存器用于选择极性并启用各种故障输入源，如表 23-13 所示。

要为故障检测启用最终输入，请设置 TIMA.FTCL.FIEN = 1。

有四种类型的故障输入源可用于同步或异步故障检测：

比较器 (COMP) 输出

当 COMP 用于检测过流或过压事件时，比较器输出对于故障检测非常有用。要启用比较器输出以进行故障检测，请设置 TIMA.FSCTL.FACxEN 位，并使用 TIMA.FCTL.FSENACx 位配置极性以检测故障（对于 COMP 实例，x = 0、1 或 2）。

外部故障管脚

许多 IC 器件都包含故障检测引脚（即 nFAULT），当系统中存在故障情况时，MCU 可以检测到该引脚。每个 TIMA 模块均连接 3 个故障外部信号引脚 (TIMA_FLTx)，其中 x = 0、1 或 2。每个信号引脚均可通过设置 TIMA.FSCTL.FEXxEN 位来启用，触发故障条件的该信号的极性可通过使用 TIMA.FCTL.FSENEXTx 位进行配置（其中对于每个 TIMA_FLTx 引脚，x = 0、1 或 2）。

系统时钟故障

任何系统时钟故障均可用于将 PWM 输出触发至高阻状态。这可以通过设置 TIMA.FSCTL.FCEN 位来启用。

备注

当发生 SYSCLK 故障时，会产生器件复位。当器件处于复位状态时，各种 TIMA 故障进入和退出选项均无效。

触发条件

触发器可以配置为生成可检测的故障条件。这对于执行诊断或从事件结构中的其他外设创建故障依赖性非常有用。有关触发器配置，请参阅节 23.2.7。可以通过设置 TIMA.FSCTL.TFIM 位来启用故障输入掩码。

表 23-13. 故障输入源和配置

信号名称	输入源	故障类型	极性位	使能位
COMP0_OUT	COMP0 输出	同步或异步	FSENAC0	FAC0EN
COMP1_OUT	COMP1 输出		FSENAC1	FAC1EN
COMP2_OUT	COMP2 输出		FSENAC2	FAC2EN
TIMA_FLT0	外部故障 0		FSENEXT0	FSENEXT0
TIMA_FLT1	外部故障 1		FSENEXT1	FSENEXT1
TIMA_FLT2	外部故障 2		FSENEXT2	FSENEXT2
SYSCLK	时钟源	同步	-	FCEN
TRIG	触发输出		-	TFIM

23.2.6.3 故障条件下的计数器行为

有两种设置可用于指定故障条件下的计数器行为：TIMA.CTRCTL.FB（故障行为期间）和 TIMA.CTRCTL.FRB（故障恢复行为）。在故障处理程序行为期间，应继续启用计数器 (TIMA.CTRCTL.EN = 1)。

表 23-14 和图 23-37 介绍了故障条件的计数器行为。

表 23-14. TIMA.CTRCTL 寄存器故障条件下的计数器行为

位字段				计数器行为
FB	FRB	CVAE	REPEAT	
0	X	X	0	忽略故障模式。计数器在故障期间继续计数，在达到零时停止。
			1/3	忽略故障模式。计数器在故障期间继续计数并重复。
1	0	X	0/1/3	立即对故障模式作出反应。计数器立即停止计数，并在整个故障模式期间停止计数。退出故障模式后，计数器会从中断的位置继续计数。
	1	0	X	立即对故障模式作出反应。计数器立即停止计数，并在整个调试模式期间停止计数。退出故障模式后，计数器会从 LOAD 值重新启动（重新启动向下计数）。
		1		立即对故障模式作出反应。计数器立即停止计数，并在整个调试模式期间停止计数。退出调试模式后，计数器会从故障进入暂停的位置重新启动。
		2		立即对故障模式作出反应。计数器立即停止计数，并在整个故障模式期间停止计数。退出故障模式后，它会从 0 值重新启动（重新启动向上/向下计数）。

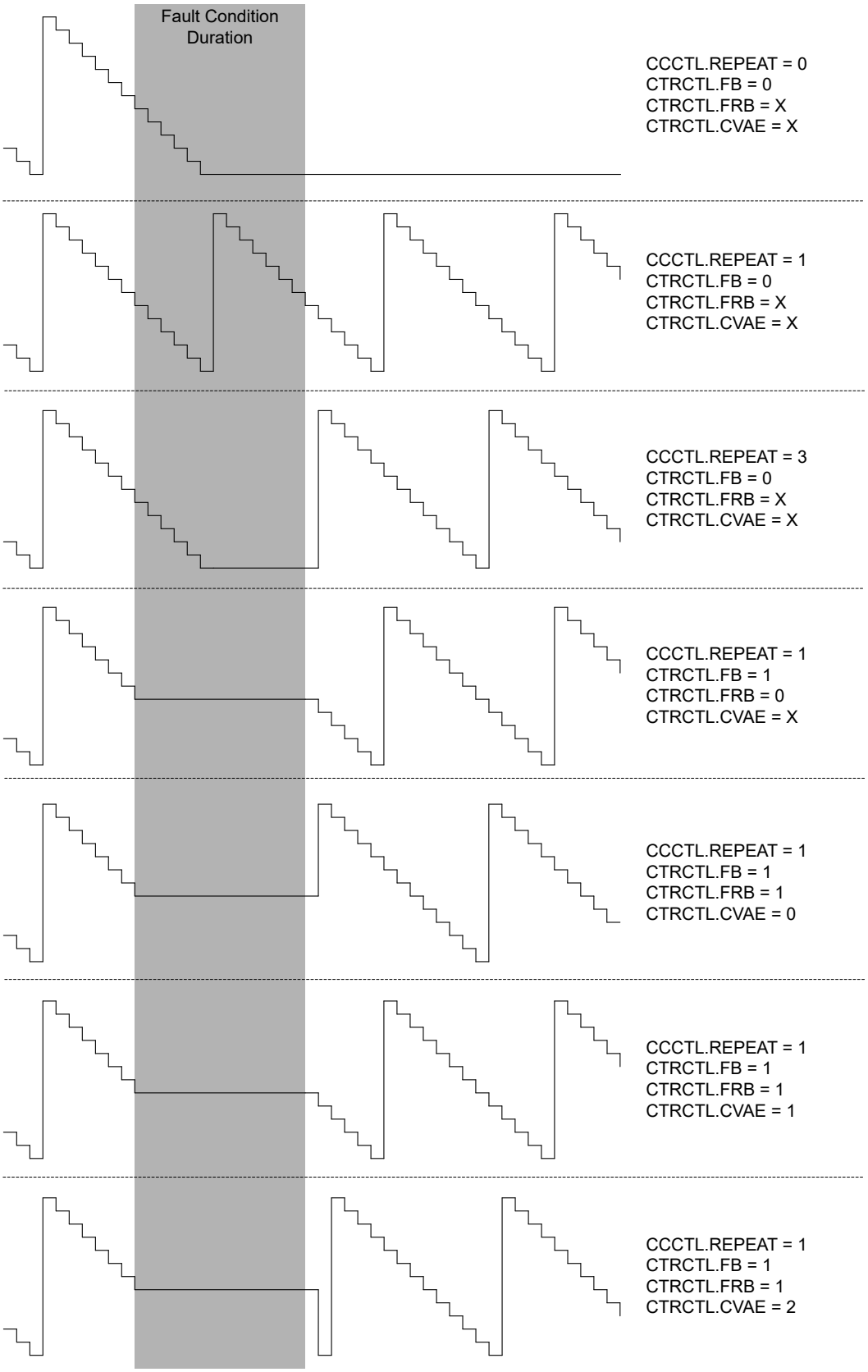


图 23-37. 使用 TIMx.CTRCTL 寄存器在故障条件下的计数器行为

23.2.6.4 故障条件下的输出行为

故障条件下 CCP 输出通道行为有两种设置，即 TIMA.CCACT_01[0/1].FEXACT（故障退出行为）和 TIMA.CCACT_01[0/1].FENACT（故障进入行为）。表 23-15 展示了故障条件的输出行为。

表 23-15. 使用 TIMA.CCACT 寄存器时故障条件下的 CCP 输出行为

位字段		输出行为
FEXACT	FENACT	
0		CCP 输出值不受故障事件的影响
1		CCP 输出值设置为高电平
2		CCP 输出值设置为低电平
3		CCP 输出值会切换
4		CCP 输出为三态（高阻态）

备注

故障进入和退出行为与启用的计数器无关。当处于故障模式时，它们将始终更新输出。在故障模式下通过软件写入启用计数器将生成一个加载事件，该事件将在 RIS 中被捕获，但计数器不会继续进行。当处于故障模式时，加载事件不会影响输出。故障事件具有绝对优先权，在退出故障模式之前，任何事件都无法更新输出。

23.2.7 通过交叉触发同步

通过将多个计时器连接在一起来使用主/次计时器配置时，交叉触发特性可以指示同一电源域或不同电源域中使用事件结构的多个计时器模块同时开始计数。

可以使用软件启用交叉触发，比较来自其他计时器实例的事件、零或加载事件或通用订阅者事件。某些应用可能需要多个计数器块，这些计数器块可在同一电源域（例如 TIMA0 和 TIMA1）或不同电源域（例如 TIMA0 和 TIMG0）上同时启动。

该配置使用来自主计时器模块的交叉触发作为次级计时器的输入触发条件。计时器交叉触发本质上是控制 TIMx.CTRCTL 寄存器中 EN 位的硬件与软件条件的组合逻辑。

从主计时器输出的交叉触发连接到其他次级计时器模块的外部触发输入。如图 23-38 所示，TIMGx 是主计时器，TIMAx 是将在配置示例中交叉触发的次级计时器。

备注

对于为计时器实例启用的电源域和交叉触发选择源，请参阅器件专用数据表。

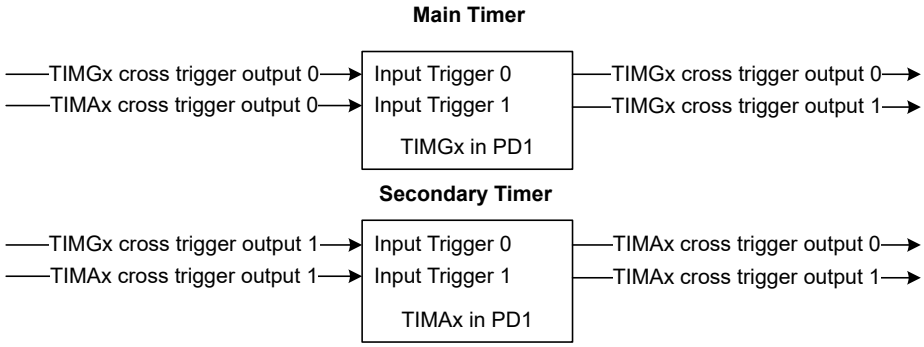


图 23-38. 电源域 1 中主计时器 (TIMGx) 和次级计时器 (TIMAx) 的交叉触发连接

23.2.7.1 主计时器交叉触发器配置

以下步骤用于配置主计时器交叉触发器（在本例中为 TIMGx）：

1. 配置主计时器（用于触发其他次级计时器）来提供所需的功能，例如配置 PWM 输出生成或使用比较模式来触发其他外设。有关如何配置 PWM 生成，请参阅节 23.2.5。
2. 选择需要生成的交叉触发器输出。例如，在图 23-38 中，TIMGx 交叉触发器 1 可用于触发 TIMAx，TIMGx 交叉触发器 0 可用于触发自身。
3. 通过将 TIMx.CTTRIGCTL.CTEN 位设置为 1 来启用交叉触发器输出功能。
4. 选择如何触发这些已连接计时器的启动：可以是来自用户端口、归零事件、加载事件或比较事件的软件触发器或硬件触发器。
   a. 对于软件事件触发器，设置 TIMx.CTTRIG.TRIG 位。
   b. 对于硬件触发事件，使用 TIMx.CTTRIGCTL.EVTCTTRIGSEL 选择触发源，并通过设置 TIMx.CTTRIGCTL.EVTCTEN 启用硬件触发。

图 23-39 展示了连接逻辑和寄存器。

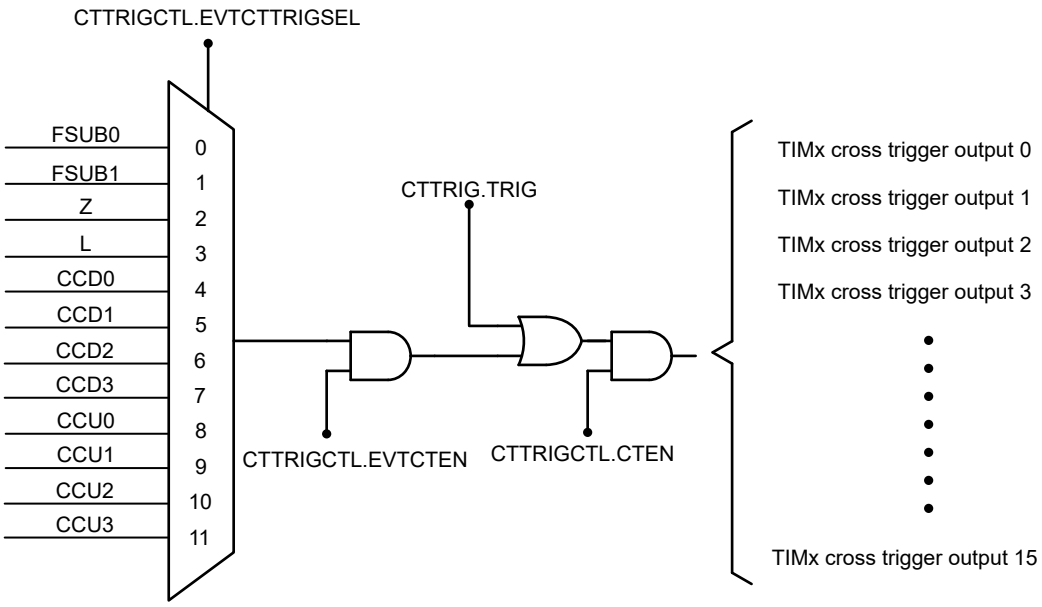


图 23-39. 主计时器交叉触发器输出配置

23.2.7.2 次级计时器交叉触发器配置

以下步骤用于配置次级计时器交叉触发器（在本例中为 TIMAx）：

1. 配置次级计时器（由主计时器触发）来提供该计时器所需的功能，例如配置 PWM 输出生成或使用比较模式来触发其他外设。有关如何配置 PWM 生成，请参阅节 23.2.5。
2. 根据特定于器件的数据表选择要使用的输入触发器。使用图 23-38 中的示例连接，TIMAx 必须由 TIMGx 触发，并且 TIMGx 的交叉触发器输出 1 连接到 TIMAx 的输入触发器 0。因此，应通过将 TIMA.TSEL.ETSEL 位设置为 0 来选择 TIMAx 的输入触发器 0。
3. 通过将 TIMA.TSEL.TE 位设置为 1 来启用输入触发器功能。
4. 将 TIMAx.IFCTL_01[0].ISEL 和 TIMA.IFCTL_01[1].ISEL 位设置为 3，以便选择触发器作为输入源。
   a. 对于中心对齐的 PWM，将 TIMA.CCCTL_01[0].ZCOND 和 TIMA.CCCTL_01[1].ZCOND 位设置为 1，以便针对归零事件使用触发器生效边沿。
   b. 对于边沿对齐的 PWM，将 TIMA.CCCTL_01[0].LCOND 和 TIMA.CCCTL_01[1].LCOND 位设置为 1，以便针对加载事件使用触发器生效边沿。
5. 在满足 LCOND 或 ZCOND 条件时会设置 TIMx.CTRCTL.EN 位，并且计数器值分别更改为加载值或零值。

由于主计时器 TIMGx 还必须触发自身，因此请完成 TIMAx 先前的配置步骤，以便触发 TIMGx 自身。

图 23-40 展示了逻辑连接。

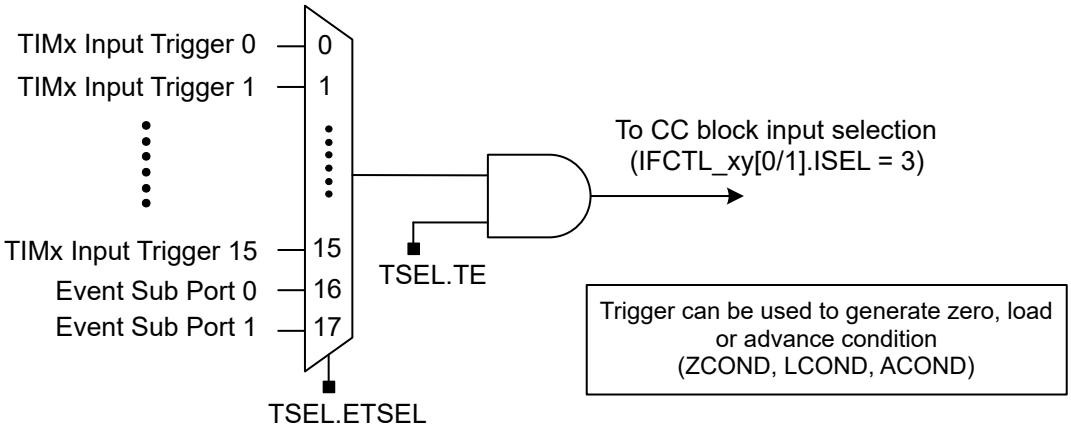


图 23-40. 次级计时器交叉触发器输入配置

备注

请参阅特定于器件的数据表中的“TIMx 交叉触发器映射”，了解使用 ETSEL 位启用的交叉触发器映射。例如，如果器件的计时器实例都使用触发器输入 0 (TRIG0) 来交叉触发其他计时器，则只有 TRIG0 可用于交叉触发其他实例。

23.2.8 低功耗运行

有关可用时钟源的低功耗模式和行为的详细信息，请参阅节 2.1.1。

电源域 PD0 中的计时器模块可以处于活动状态并配置为在除关断模式之外的所有电源模式下继续计数。有关每种低功耗模式下可用的时钟源，请参阅节 2.1.1。用户需要配置合适的时钟，以便在低功耗模式下为计时器提供时钟源。

电源域 PD1 中的计时器模块只能在运行和睡眠模式下处于活动状态。当系统进入停止或待机模式时，计时器模块将被强制进入禁用状态，并在系统返回至运行或睡眠模式时恢复。

23.2.9 中断和事件支持

TIMx 模块包含三个事件发布者和两个事件订阅者。

- 一个事件发布者 (CPU_INT) 通过静态事件路由来管理对 CPU 子系统的 TIMx 中断请求 (IRQ)。
- 第二个和第三个事件（GEN_EVENT0 和 GEN_EVENT1）用于通过通用路由设置通用事件发布者和订阅者。

表 23-16 中总结了 TIMx 事件。

表 23-16. TIMx 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断	发布者	TIMx	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 TIMx 到 CPU 的中断路由
通用发布者事件	发布者	TIMx	其他外设	通用路由	GEN_EVENT0 和 FPUB_0 寄存器	从 TIMx 到其他外设的可配置中断路由
通用发布者事件	发布者	TIMx	其他外设	通用路由	GEN_EVENT 1 和 FPUB_1 寄存器	从 TIMx 到其他外设的可配置中断路由
通用订阅者事件	订阅者	其他外设	TIMx	通用路由	FSUB_0	从其他外设到 TIMx 的可配置中断路由
通用订阅者事件	订阅者	其他外设	TIMx	通用路由	FSUB_1	从其他外设到 TIMx 的可配置中断路由

23.2.9.1 CPU 中断事件发布者 (CPU_INT)

TIMx 模块提供 18 个中断源（取决于特定的 TIMx 模块功能），这些中断源可配置为产生 CPU 中断事件。表 23-17 按照降序排列的中断优先级列出了来自 TIMx 的 CPU 中断事件。

表 23-17. TIMx CPU 中断事件条件 (CPU_INT)

IIDX STAT	名称	说明	计时器模块
0x01	Z	归零事件中断。存在归零事件时设置该中断。	TIMx
0x02	L	加载事件中断。存在加载事件时设置该中断。	TIMx
0x05	CCD0	捕获或比较 0 递减事件。当 CC0 处发生向下比较匹配事件时，设置此中断。	TIMx
0x06	CCD1	捕获或比较 1 递减事件。当 CC1 处发生向下比较匹配事件时，设置此中断。	TIMx
0x07	CCD2	捕获或比较 2 递减事件。当 CC2 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA0。	TIMx
0x08	CCD3	捕获或比较 3 递减事件。当 CC3 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA0。	TIMx
0x09	CCU0	捕获或比较 0 递增事件。当 CC0 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0A	CCU1	捕获或比较 1 递增事件。当 CC1 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0B	CCU2	捕获或比较 2 递增事件。当 CC2 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0C	CCU3	捕获或比较 3 递增事件。当 CC3 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0D	CCD4	捕获或比较 4 递减事件。当 CC4 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x0E	CCD5	捕获或比较 5 递减事件。当 CC5 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x0F	CCU4	捕获或比较 4 递增事件。当 CC4 处发生向上比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x10	CCU5	捕获或比较 5 递增事件。当 CC5 处发生向上比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x19	F	故障事件中断。存在故障条件事件时，设置此中断。请参阅节 23.2.6。该中断仅适用于具有故障处理程序功能的 TIMA 模块。	TIMA
0x1A	TOV	触发溢出中断。如果在相关触发通道处于运行状态时产生触发事件，则设置此中断。	TIMx
0x1B	REPC	重复计数器归零中断。如果重复计数器值从非零值转变为零，该位控制中断的产生。该中断仅适用于具有重复计数器功能的 TIMA 模块。	TIMA
0x1C	DC	QEI 模式下使用的方向改变中断。此中断仅适用于具有 QEI 功能的 TIMG 模块。	TIMG
0X1D	QEIERR	QEI 模式下使用的方向改变中断。此中断仅适用于具有 QEI 功能的 TIMG 模块。	TIMG

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

23.2.9.2 通用事件发布者和订阅者（GEN_EVENT0 和 GEN_EVENT1）

通用路由是这样的一种路由，其中发布事件的比较器外设配置为使用多个可用的通用路由通道之一来将事件发布到另一个实体（如果是分离器路由，则为多个实体）。在这里，实体可以是另一个外设、通用 DMA 触发事件或通用 CPU 事件。

GEN_EVENT0 和 GEN_EVENT1 寄存器用于选择用于发布或订阅事件的外设条件（表 23-18）。FPUB_0 和 FPUB_1 是发布者端口寄存器，用于配置使用哪个通用路由通道来广播事件。FSUB_0 和 FSUB_1 是订阅者端口寄存器，用于配置使用哪个通用路由通道来订阅事件。其他外设、DMA 或 CPU 可以通过将其订阅者端口配置为侦听发布外设所连同一通用路由通道来订阅此事件。

例如，通过使用通用事件通道，可以通过将 TIMx FPUB_x 和 ADC FSUB_0 连接到同一通用事件通道，直接从 TIMx 事件启动 ADC 转换。有关通用事件路由的工作方式，请参阅节 7.1.3.3 和节 7.2.3。

表 23-18. TIMx 通用事件条件（GEN_EVENT0 和 GEN_EVENT1）

IIDX STAT	名称	说明	计时器模块
0x01	Z	归零事件中断。存在归零事件时设置该中断。	TIMx
0x02	L	加载事件中断。存在加载事件时设置该中断。	TIMx
0x05	CCD0	捕获或比较 0 递减事件。当 CC0 处发生向下比较匹配事件时，设置此中断。	TIMx
0x06	CCD1	捕获或比较 1 递减事件。当 CC1 处发生向下比较匹配事件时，设置此中断。	TIMx
0x07	CCD2	捕获或比较 2 递减事件。当 CC2 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA0。	TIMx
0x08	CCD3	捕获或比较 3 递减事件。当 CC3 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA0。	TIMx
0x09	CCU0	捕获或比较 0 递增事件。当 CC0 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0A	CCU1	捕获或比较 1 递增事件。当 CC1 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0B	CCU2	捕获或比较 2 递增事件。当 CC2 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0C	CCU3	捕获或比较 3 递增事件。当 CC3 处发生向上比较匹配事件时，设置此中断。	TIMx
0x0D	CCD4	捕获或比较 4 递减事件。当 CC4 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x0E	CCD5	捕获或比较 5 递减事件。当 CC5 处发生向下比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x0F	CCU4	捕获或比较 4 递增事件。当 CC4 处发生向上比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x10	CCU5	捕获或比较 5 递增事件。当 CC5 处发生向上比较匹配事件时，设置此中断。该中断仅适用于 TIMA 模块。	TIMA
0x19	F	故障事件中断。存在故障条件事件时，设置此中断。 请参阅节 23.2.6 该中断仅适用于具有故障处理程序功能的 TIMA 模块。	TIMA
0x1A	TOV	触发溢出中断。如果在相关触发通道处于运行状态时产生触发事件，则设置此中断。	TIMx
0x1B	REPC	重复计数器归零中断。如果重复计数器值从非零值转变为零，该位控制中断的产生。该中断仅适用于具有重复计数器功能的 TIMA 模块。	TIMA
0x1C	DC	QEI 模式下使用的方向改变中断。此中断仅适用于具有 QEI 功能的 TIMG 模块。	TIMG
0x1D	QEIERR	QEI 模式下使用的方向改变中断。此中断仅适用于具有 QEI 功能的 TIMG 模块。	TIMG

有关配置事件寄存器的指导，请参阅节 7.2.5。

23.2.9.3 通用订阅者事件示例

要使用比较器输出事件在 TIMx 模块中生成捕获事件，请将比较器用作事件发布者 (FPUB_1)，将计时器用作事件订阅者（FSUB_0 或 FSUB_1）。

比较器配置（发布者）：

1. 设置 GEN_EVENT0 IMASK 寄存器中的 COMPIFG 位来屏蔽比较器输出中断。请参阅节 11.2.11.2。
2. 在比较器模块中配置 FPUB_1 寄存器以连接到事件通道 y。
3. 配置和启用比较器。

TIMx 配置（订阅者）：

1. 通过配置 TIMx.TSEL.ETSEL = 0x10（对于 FSUB 端口 0）或 0x11（对于 FSUB 端口 1），将事件订阅者端口配置为触发源。通过将 TIMx.TSEL.TE 位设置为 1 来启用输入触发器功能。
2. 将 TIMx.IFCTL_xy[0/1].ISEL 位设置为 3 以选择交叉触发器作为 TIMx 模块的输入源。请参阅图 23-12。
3. 使用 TIMx.CCCTL_xy[0/1] 寄存器中的 LCOND、ZCOND 和 CCOND 位来配置使用交叉触发器生成加载、归零、前进或捕获事件。请参阅节 23.2.7。
4. 配置 TIMx 模块中的 FSUB_0 或 FSUB_1 寄存器以连接到事件通道 y。通道 y 不能正在被另一个外设使用。
5. 配置并启用 TIMx。

备注

还可以使用 TIMx.IFCTL_xy[0/1].ISEL = 7h、8h 或 9h 将比较器输出 (COMP0:2) 直接配置为 TIMx CC 模块的输入。这是一条延迟较低的路径，对于逐周期过流限制等应用很有用。请参阅图 23-12。

23.2.10 调试处理程序（仅限 TIMA）

仅在 TIMA 中，CPU 暂停调试模式下的计时器输出和计数器行为也可以通过软件使用 PDBGCTL 和 CCACT_xy[0/1] 寄存器进行配置。

23.3 计时器 (TIMx) 寄存器

表 23-19 列出了计时器 (TIMx) 寄存器的存储器映射寄存器。表 23-19 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 23-19. 计时器 (TIMX) 寄存器

偏移	缩写	寄存器名称	组	部分
400h	FSUB_0	订阅者端口 0		转到
404h	FSUB_1	订阅者端口 1		转到
444h	FPUB_0	发布者端口 0		转到
448h	FPUB_1	发布者端口 1		转到
800h	PWREN	电源使能		转到
804h	RSTCTL	复位控制		转到
814h	STAT	状态寄存器		转到
1000h	CLKDIV	时钟分频器		转到
1008h	CLKSEL	超低功耗外设的时钟选择		转到
1018h	PDBGCTL	外设调试控制		转到
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
1050h	IIDX	中断索引	GEN_EVENT 0	转到
1058h	IMASK	中断屏蔽	GEN_EVENT 0	转到
1060h	RIS	原始中断状态	GEN_EVENT 0	转到
1068h	MIS	已屏蔽中断状态	GEN_EVENT 0	转到
1070h	ISET	中断设置	GEN_EVENT 0	转到
1078h	ICLR	中断清除	GEN_EVENT 0	转到
1080h	IIDX	中断索引	Gen_EVENT1	转到
1088h	IMASK	中断屏蔽	Gen_EVENT1	转到
1090h	RIS	原始中断状态	Gen_EVENT1	转到
1098h	MIS	已屏蔽中断状态	Gen_EVENT1	转到
10A0h	ISET	中断设置	Gen_EVENT1	转到
10A8h	ICLR	中断清除	Gen_EVENT1	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	CCPD	CCP 方向		转到
1104h	ODIS	输出禁用		转到
1108h	CCLKCTL	计数器时钟控制寄存器		转到
110Ch	CPS	时钟预分频寄存器		转到
1110h	CPSV	时钟预分频计数状态寄存器		转到
1114h	CTTRIGCTL	计时器交叉触发器控制寄存器		转到
111Ch	CTTRIG	计时器交叉触发器寄存器		转到

表 23-19. 计时器 (TIMX) 寄存器 (continued)

偏移	缩写	寄存器名称	组	部分
1120h	FSCTL	故障源控制		转到
1124h	GCTL	全局控制寄存器		转到
1800h	计数器	计数器寄存器		转到
1804h	CTRCTL	计数器控制寄存器		转到
1808h	LOAD	加载寄存器		转到
1810h + 公式	CC_01[y]	捕获或比较寄存器 0/1		转到
1818h + 公式	CC_23[y]	捕获或比较寄存器 0/1		转到
1820h + 公式	CC_45[y]	CC_45 寄存器是可用于与当前 CTR 进行比较以创建事件 CC4U、CC4D、CC5U 和 CC5D 的寄存器。		转到
1830h + 公式	CCCTL_01[y]	捕获或比较控制寄存器		转到
1838h + 公式	CCCTL_23[y]	捕获或比较控制寄存器 0/1		转到
1840h + 公式	CCCTL_45[y]	捕获或比较控制寄存器 2/3		转到
1850h + 公式	OCTL_01[y]	CCP 输出控制寄存器 4/5		转到
1858h + 公式	OCTL_23[y]	CCP 输出控制寄存器 0/1		转到
1870h + 公式	CCACT_01[y]	捕获或比较操作寄存器 2/3		转到
1878h + 公式	CCACT_23[y]	捕获或比较操作寄存器 0/1		转到
1880h + 公式	IFCTL_01[y]	输入滤波器控制寄存器 0/1		转到
1888h + 公式	IFCTL_23[y]	输入滤波器控制寄存器 2/3		转到
18A0h	PL	相位加载寄存器		转到
18A4h	DBCTL	死区插入控制寄存器		转到
18B0h	TSEL	触发选择寄存器		转到
18B4h	RC	重复计数器寄存器		转到
18B8h	RCLD	重复计数器加载寄存器		转到
18BCh	QDIR	QEI 计数方向寄存器		转到
18D0h	FCTL	故障控制寄存器		转到
18D4h	FIFCTL	故障输入滤波器控制寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 23-20 展示了适用于此部分中访问类型的代码。

表 23-20. 计时器 (TIMx) 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入

表 23-20. 计时器 (TIMx) 访问类型代码 (continued)

访问类型	代码	说明
复位或默认值		
-n		复位后的值或默认值
寄存器数组变量		
i、j、k、l、m、n		当这些变量用于寄存器名称、偏移或地址时，它们指的是寄存器数组的值，其中寄存器是一组重复寄存器的一部分。寄存器组构成分层结构，数组用公式表示。
y		当该变量用于寄存器名称、偏移或地址时，它指的是寄存器数组的值。

23.3.1 FSUB_0（偏移 = 400h）[复位 = 00000000h]

图 23-41 展示了 FSUB_0，表 23-21 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 23-41. FSUB_0

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

表 23-21. FSUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

23.3.2 FSUB_1（偏移 = 404h）[复位 = 00000000h]

图 23-42 展示了 FSUB_1，表 23-22 中对此进行了介绍。

返回到汇总表。

订阅者端口

图 23-42. FSUB_1

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

表 23-22. FSUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

23.3.3 FPUB_0（偏移 = 444h）[复位 = 00000000h]

图 23-43 展示了 FPUB_0，表 23-23 中对此进行了介绍。

返回到汇总表。

发布者端口

图 23-43. FPUB_0

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

表 23-23. FPUB_0 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

23.3.4 FPUB_1（偏移 = 448h）[复位 = 00000000h]

图 23-44 展示了 FPUB_1，表 23-24 中对此进行了介绍。

返回到汇总表。

发布者端口

图 23-44. FPUB_1

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

表 23-24. FPUB_1 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

23.3.5 PWREN（偏移 = 800h）[复位 = 00000000h]

图 23-45 展示了 PWREN，表 23-25 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 23-45. PWREN

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							K-0h

表 23-25. PWREN 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

23.3.6 RSTCTL（偏移 = 804h）[复位 = 00000000h]

图 23-46 展示了 RSTCTL，表 23-26 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 23-46. RSTCTL

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

表 23-26. RSTCTL 字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 STAT 寄存器中的 RESETSTKY 位 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 清除复位粘滞位
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

23.3.7 STAT（偏移 = 814h）[复位 = 00000000h]

图 23-47 展示了 STAT，表 23-27 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 23-47. STAT

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

表 23-27. STAT 字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 清除该位以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次清除该位以来，外设尚未复位 1h = 自从上次清除该位以来，外设已复位
15-0	RESERVED	R	0h	

23.3.8 CLKDIV（偏移 = 1000h）[复位 = 00000000h]

图 23-48 展示了 CLKDIV，表 23-28 中对此进行了介绍。

返回到汇总表。

该寄存器用于指定功能时钟的模块专用分频比

图 23-48. CLKDIV

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

表 23-28. CLKDIV 字段说明

位	字段	类型	复位	说明
31-3	RESERVED	R/W	0h	
2-0	RATIO	R/W	0h	选择模块时钟的分频比 0h = 不对时钟源进行分频 1h = 对时钟源进行 2 分频 2h = 对时钟源进行 3 分频 3h = 对时钟源进行 4 分频 4h = 对时钟源进行 5 分频 5h = 对时钟源进行 6 分频 6h = 对时钟源进行 7 分频 7h = 对时钟源进行 8 分频

23.3.9 CLKSEL（偏移 = 1008h）[复位 = 00000000h]

图 23-49 展示了 CLKSEL，表 23-29 中对此进行了介绍。

返回到汇总表。

时钟源选择寄存器

图 23-49. CLKSEL

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED				BUSCLK_SEL	MFCLK_SEL	LFCLK_SEL	RESERVED
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 23-29. CLKSEL 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	BUSCLK_SEL	R/W	0h	如果启用，选择 BUSCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
2	MFCLK_SEL	R/W	0h	如果启用，选择 MFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
1	LFCLK_SEL	R/W	0h	如果启用，选择 LFCLK 作为时钟源 0h = 不选择此时钟作为时钟源 1h = 选择此时钟作为时钟源
0	RESERVED	R/W	0h	

23.3.10 PDBGCTL（偏移 = 1018h）[复位 = 00000003h]

图 23-50 展示了 PDBGCTL，表 23-30 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 23-50. PDBGCTL

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

表 23-30. PDBGCTL 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	SOFT	R/W	1h	软停止边界控制。此功能仅在 FREE 设置为“STOP”时可用 0h = 外设将立即停止，即使系统重新启动后产生的状态将导致损坏的情况下也是如此 1h = 外设阻止调试冻结，直至其达到可以恢复而不会损坏的边界
0	免费	R/W	1h	自由运行控制 0h = 当“内核停止”输入变为有效时，外设功能冻结；当该输入变为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

23.3.11 IIDX（偏移 = 1020h）[复位 = 00000000h]

图 23-51 展示了 IIDX，表 23-31 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 23-51. IIDX

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

表 23-31. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 01h = 中断源：归零事件 (Z) 02h = 中断源： 加载事件 (L) 05h = 中断源： 捕获或比较递减事件 (CCD0) 06h = 中断源：捕获或比较递减事件 (CCD1) 07h = 中断源：捕获或比较递减事件 (CCD2) 08h = 中断源：捕获或比较递减事件 (CCD3) 09h = 中断源：捕获或比较递增事件 (CCU0) 0Ah = 中断源：捕获或比较递增事件 (CCU1) 0Bh = 中断源：捕获或比较递增事件 (CCU2) 0Ch = 中断源：捕获或比较递增事件 (CCU3) 0Dh = 中断源：比较递减事件 (CCD4) 0Eh = 中断源：比较递减事件 (CCD5) 0Fh = 中断源：比较递减事件 (CCU4) 10h = 中断源：比较递减事件 (CCU5) 19h = 中断源：故障事件产生了中断。(F) 1Ah = 中断源： 触发溢出 (TOV) 1Bh = 中断源： 重复计数器归零 (REPC) 1Ch = 中断源： 方向更改 (DC) 1Dh = 中断源：QEI 错误状态转换错误 (QEIERR)

23.3.12 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 23-52 展示了 IMASK，表 23-32 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 23-52. IMASK

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R/W-			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-		R/W-0h	R/W-0h

表 23-32. IMASK 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	QEIERR	R/W	0h	QEIERR 事件屏蔽 0h = 禁用事件 1h = 启用事件
27	DC	R/W	0h	方向更改事件屏蔽 0h = 禁用事件 1h = 启用事件
26	REPC	R/W	0h	重复计数器归零事件屏蔽 0h = 禁用事件 1h = 启用事件
25	TOV	R/W	0h	触发溢出事件屏蔽 0h = 禁用事件 1h = 启用事件
24	F	R/W	0h	故障事件屏蔽 0h = 禁用事件 1h = 启用事件
23-16	保留	R/W	0h	
15	CCU5	R/W	0h	比较递增事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽
14	CCU4	R/W	0h	比较递增事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
13	CCD5	R/W	0h	比较 DN 事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 23-32. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R/W	0h	比较 DN 事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
11	CCU3	R/W	0h	捕获或比较递增事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
10	CCU2	R/W	0h	捕获或比较递增事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9	CCU1	R/W	0h	捕获或比较递增事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	CCU0	R/W	0h	捕获或比较递增事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	CCD3	R/W	0h	捕获或比较递减事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	CCD2	R/W	0h	捕获或比较递减事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
5	CCD1	R/W	0h	捕获或比较递减事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	CCD0	R/W	0h	捕获或比较递减事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3-2	RESERVED	R/W	0h	
1	L	R/W	0h	加载事件屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	Z	R/W	0h	归零事件屏蔽 0h = 禁用事件 1h = 启用事件

23.3.13 RIS（偏移 = 1030h）[复位 = 00000000h]

图 23-53 展示了 RIS，表 23-33 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 23-53. RIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-33. RIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR，当编码器接口上发生错误的状态转换时设置。 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCU4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCD5 0h = 事件已被清除 1h = 事件已被设置

表 23-33. RIS 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R	0h	比较递减事件生成了一个中断 CCD4 0h = 事件已被清除 1h = 事件已被设置
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.14 MIS (偏移 = 1038h) [复位 = 00000000h]

图 23-54 展示了 MIS，表 23-34 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 23-54. MIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-34. MIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
12	CCD4	R	0h	比较递减事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置

表 23-34. MIS 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.15 ISET (偏移 = 1040h) [复位 = 00000000h]

图 23-55 展示了 ISET，表 23-35 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 23-55. ISET

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-35. ISET 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
27	直流	W	0h	方向更改事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
26	REPC	W	0h	重复计数器归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
25	TOV	W	0h	触发溢出事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
24	F	W	0h	故障事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
14	CCU4	W	0h	比较递增事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
13	CCD5	W	0h	比较递减事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

表 23-35. ISET 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	W	0h	比较递减事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
11	CCU3	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
10	CCU2	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
9	CCU1	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
8	CCU0	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
7	CCD3	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
6	CCD2	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
5	CCD1	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
4	CCD0	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
0	Z	W	0h	归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

23.3.16 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 23-56 展示了 ICLR，表 23-36 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 23-56. ICLR

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-36. ICLR 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
27	直流	W	0h	方向更改事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
26	REPC	W	0h	重复计数器归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
25	TOV	W	0h	触发溢出事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
24	F	W	0h	故障事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
14	CCU4	W	0h	比较递增事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除
13	CCD5	W	0h	比较递减事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
12	CCD4	W	0h	比较递减事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除

表 23-36. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
10	CCU2	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
9	CCU1	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
8	CCU0	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
7	CCD3	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
6	CCD2	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
5	CCD1	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
4	CCD0	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
0	Z	W	0h	归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除

23.3.17 IIDX (偏移 = 1050h) [复位 = 00000000h]

图 23-57 展示了 IIDX，表 23-37 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断 (最高优先级) 由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU (不是从调试接口) 读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 23-57. IIDX

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

表 23-37. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 01h = 中断源：归零事件 (Z) 02h = 中断源： 加载事件 (L) 05h = 中断源： 捕获或比较递减事件 (CCD0) 06h = 中断源：捕获或比较递减事件 (CCD1) 07h = 中断源：捕获或比较递减事件 (CCD2) 08h = 中断源：捕获或比较递减事件 (CCD3) 09h = 中断源：捕获或比较递增事件 (CCU0) 0Ah = 中断源：捕获或比较递增事件 (CCU1) 0Bh = 中断源：捕获或比较递增事件 (CCU2) 0Ch = 中断源：捕获或比较递增事件 (CCU3) 0Dh = 中断源：比较递减事件 (CCD4) 0Eh = 中断源：比较递减事件 (CCD5) 0Fh = 中断源：比较递减事件 (CCU4) 10h = 中断源：比较递减事件 (CCU5) 19h = 中断源：故障事件产生了中断。(F) 1Ah = 中断源： 触发溢出 (TOV) 1Bh = 中断源： 重复计数器归零 (REPC) 1Ch = 中断源： 方向更改 (DC) 1Dh = 中断源：QEI 错误状态转换错误 (QEIERR)

23.3.18 IMASK（偏移 = 1058h）[复位 = 00000000h]

图 23-58 展示了 IMASK，表 23-38 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 23-58. IMASK

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R/W-			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-		R/W-0h	R/W-0h

表 23-38. IMASK 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	QEIERR	R/W	0h	QEIERR 事件屏蔽 0h = 禁用事件 1h = 启用事件
27	DC	R/W	0h	方向更改事件屏蔽 0h = 禁用事件 1h = 启用事件
26	REPC	R/W	0h	重复计数器归零事件屏蔽 0h = 禁用事件 1h = 启用事件
25	TOV	R/W	0h	触发溢出事件屏蔽 0h = 禁用事件 1h = 启用事件
24	F	R/W	0h	故障事件屏蔽 0h = 禁用事件 1h = 启用事件
23-16	保留	R/W	0h	
15	CCU5	R/W	0h	比较递增事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽
14	CCU4	R/W	0h	比较递增事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
13	CCD5	R/W	0h	比较 DN 事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 23-38. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R/W	0h	比较 DN 事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
11	CCU3	R/W	0h	捕获或比较递增事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
10	CCU2	R/W	0h	捕获或比较递增事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9	CCU1	R/W	0h	捕获或比较递增事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	CCU0	R/W	0h	捕获或比较递增事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	CCD3	R/W	0h	捕获或比较递减事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	CCD2	R/W	0h	捕获或比较递减事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
5	CCD1	R/W	0h	捕获或比较递减事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	CCD0	R/W	0h	捕获或比较递减事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3-2	RESERVED	R/W	0h	
1	L	R/W	0h	加载事件屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	Z	R/W	0h	归零事件屏蔽 0h = 禁用事件 1h = 启用事件

23.3.19 RIS（偏移 = 1060h）[复位 = 00000000h]

图 23-59 展示了 RIS，表 23-39 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 23-59. RIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-39. RIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR，当编码器接口上发生错误的状态转换时设置。 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCU4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCD5 0h = 事件已被清除 1h = 事件已被设置

表 23-39. RIS 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R	0h	比较递减事件生成了一个中断 CCD4 0h = 事件已被清除 1h = 事件已被设置
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.20 MIS (偏移 = 1068h) [复位 = 00000000h]

图 23-60 展示了 MIS，表 23-40 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 23-60. MIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-40. MIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
12	CCD4	R	0h	比较递减事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置

表 23-40. MIS 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.21 ISET (偏移 = 1070h) [复位 = 00000000h]

图 23-61 展示了 ISET，表 23-41 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 23-61. ISET

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-41. ISET 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
27	直流	W	0h	方向更改事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
26	REPC	W	0h	重复计数器归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
25	TOV	W	0h	触发溢出事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
24	F	W	0h	故障事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
14	CCU4	W	0h	比较递增事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
13	CCD5	W	0h	比较递减事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

表 23-41. ISET 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	W	0h	比较递减事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
11	CCU3	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
10	CCU2	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
9	CCU1	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
8	CCU0	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
7	CCD3	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
6	CCD2	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
5	CCD1	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
4	CCD0	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
0	Z	W	0h	归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

23.3.22 ICLR（偏移 = 1078h）[复位 = 00000000h]

图 23-62 展示了 ICLR，表 23-42 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 23-62. ICLR

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-42. ICLR 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
27	直流	W	0h	方向更改事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
26	REPC	W	0h	重复计数器归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
25	TOV	W	0h	触发溢出事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
24	F	W	0h	故障事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
14	CCU4	W	0h	比较递增事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除
13	CCD5	W	0h	比较递减事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
12	CCD4	W	0h	比较递减事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除

表 23-42. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
10	CCU2	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
9	CCU1	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
8	CCU0	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
7	CCD3	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
6	CCD2	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
5	CCD1	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
4	CCD0	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
0	Z	W	0h	归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除

23.3.23 IIDX（偏移 = 1080h）[复位 = 00000000h]

图 23-63 展示了 IIDX，表 23-43 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 23-63. IIDX

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

表 23-43. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 01h = 中断源：归零事件 (Z) 02h = 中断源： 加载事件 (L) 05h = 中断源： 捕获或比较递减事件 (CCD0) 06h = 中断源：捕获或比较递减事件 (CCD1) 07h = 中断源：捕获或比较递减事件 (CCD2) 08h = 中断源：捕获或比较递减事件 (CCD3) 09h = 中断源：捕获或比较递增事件 (CCU0) 0Ah = 中断源：捕获或比较递增事件 (CCU1) 0Bh = 中断源：捕获或比较递增事件 (CCU2) 0Ch = 中断源：捕获或比较递增事件 (CCU3) 0Dh = 中断源：比较递减事件 (CCD4) 0Eh = 中断源：比较递减事件 (CCD5) 0Fh = 中断源：比较递减事件 (CCU4) 10h = 中断源：比较递减事件 (CCU5) 19h = 中断源：故障事件产生了中断。(F) 1Ah = 中断源： 触发溢出 (TOV) 1Bh = 中断源： 重复计数器归零 (REPC) 1Ch = 中断源： 方向更改 (DC) 1Dh = 中断源：QEI 错误状态转换错误 (QEIERR)

23.3.24 IMASK（偏移 = 1088h）[复位= 00000000h]

图 23-64 展示了 IMASK，表 23-44 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 23-64. IMASK

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R/W-			R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-		R/W-0h	R/W-0h

表 23-44. IMASK 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R/W	0h	
28	QEIERR	R/W	0h	QEIERR 事件屏蔽 0h = 禁用事件 1h = 启用事件
27	DC	R/W	0h	方向更改事件屏蔽 0h = 禁用事件 1h = 启用事件
26	REPC	R/W	0h	重复计数器归零事件屏蔽 0h = 禁用事件 1h = 启用事件
25	TOV	R/W	0h	触发溢出事件屏蔽 0h = 禁用事件 1h = 启用事件
24	F	R/W	0h	故障事件屏蔽 0h = 禁用事件 1h = 启用事件
23-16	保留	R/W	0h	
15	CCU5	R/W	0h	比较递增事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽
14	CCU4	R/W	0h	比较递增事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
13	CCD5	R/W	0h	比较 DN 事件屏蔽 CCP5 0h = 清除中断屏蔽 1h = 设置中断屏蔽

表 23-44. IMASK 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R/W	0h	比较 DN 事件屏蔽 CCP4 0h = 清除中断屏蔽 1h = 设置中断屏蔽
11	CCU3	R/W	0h	捕获或比较递增事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
10	CCU2	R/W	0h	捕获或比较递增事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
9	CCU1	R/W	0h	捕获或比较递增事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
8	CCU0	R/W	0h	捕获或比较递增事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
7	CCD3	R/W	0h	捕获或比较递减事件屏蔽 CCP3 0h = 清除中断屏蔽 1h = 设置中断屏蔽
6	CCD2	R/W	0h	捕获或比较递减事件屏蔽 CCP2 0h = 清除中断屏蔽 1h = 设置中断屏蔽
5	CCD1	R/W	0h	捕获或比较递减事件屏蔽 CCP1 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	CCD0	R/W	0h	捕获或比较递减事件屏蔽 CCP0 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3-2	RESERVED	R/W	0h	
1	L	R/W	0h	加载事件屏蔽 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	Z	R/W	0h	归零事件屏蔽 0h = 禁用事件 1h = 启用事件

23.3.25 RIS（偏移 = 1090h）[复位 = 00000000h]

图 23-65 展示了 RIS，表 23-45 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 23-65. RIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-45. RIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR，当编码器接口上发生错误的状态转换时设置。 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCU4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCD5 0h = 事件已被清除 1h = 事件已被设置

表 23-45. RIS 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	R	0h	比较递减事件生成了一个中断 CCD4 0h = 事件已被清除 1h = 事件已被设置
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.26 MIS (偏移 = 1098h) [复位 = 00000000h]

图 23-66 展示了 MIS，表 23-46 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 23-66. MIS

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
R-0h	R-0h	R-0h	R-0h	R-0h		R-0h	R-0h

表 23-46. MIS 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	R	0h	
28	QEIERR	R	0h	QEIERR 0h = 事件已被清除 1h = 事件已被设置
27	直流	R	0h	方向更改 0h = 事件已被清除 1h = 事件已被设置
26	REPC	R	0h	重复计数器归零 0h = 事件已被清除 1h = 事件已被设置
25	TOV	R	0h	触发溢出 0h = 事件已被清除 1h = 事件已被设置
24	F	R	0h	故障 0h = 事件已被清除 1h = 事件已被设置
23-16	RESERVED	R	0h	
15	CCU5	R	0h	比较递增事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
14	CCU4	R	0h	比较递增事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置
13	CCD5	R	0h	比较递减事件生成了一个中断 CCP5 0h = 事件已被清除 1h = 事件已被设置
12	CCD4	R	0h	比较递减事件生成了一个中断 CCP4 0h = 事件已被清除 1h = 事件已被设置

表 23-46. MIS 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	R	0h	捕获或比较递增事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
10	CCU2	R	0h	捕获或比较递增事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
9	CCU1	R	0h	捕获或比较递增事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
8	CCU0	R	0h	捕获或比较递增事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
7	CCD3	R	0h	捕获或比较递减事件生成了一个中断 CCP3 0h = 事件已被清除 1h = 事件已被设置
6	CCD2	R	0h	捕获或比较递减事件生成了一个中断 CCP2 0h = 事件已被清除 1h = 事件已被设置
5	CCD1	R	0h	捕获或比较递减事件生成了一个中断 CCP1 0h = 事件已被清除 1h = 事件已被设置
4	CCD0	R	0h	捕获或比较递减事件生成了一个中断 CCP0 0h = 事件已被清除 1h = 事件已被设置
3-2	RESERVED	R	0h	
1	L	R	0h	加载事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置
0	Z	R	0h	归零事件生成了一个中断。 0h = 事件已被清除 1h = 事件已被设置

23.3.27 ISET (偏移 = 10A0h) [复位 = 00000000h]

图 23-67 展示了 ISET，表 23-47 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断 (在诊断和安全检查中很有用) 。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 23-67. ISET

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-47. ISET 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
27	直流	W	0h	方向更改事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
26	REPC	W	0h	重复计数器归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
25	TOV	W	0h	触发溢出事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
24	F	W	0h	故障事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
14	CCU4	W	0h	比较递增事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
13	CCD5	W	0h	比较递减事件 5 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

表 23-47. ISET 字段说明 (continued)

位	字段	类型	复位	说明
12	CCD4	W	0h	比较递减事件 4 设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
11	CCU3	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
10	CCU2	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
9	CCU1	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
8	CCU0	W	0h	捕获或比较递增事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
7	CCD3	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
6	CCD2	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
5	CCD1	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
4	CCD0	W	0h	捕获或比较递减事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置
0	Z	W	0h	归零事件设置 0h = 写入 0 不产生影响。 1h = 事件已被设置

23.3.28 ICLR（偏移 = 10A8h）[复位 = 00000000h]

图 23-68 展示了 ICLR，表 23-48 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 23-68. ICLR

31	30	29	28	27	26	25	24
RESERVED			QEIERR	直流	REPC	TOV	F
W-0h			W-0h	W-0h	W-0h	W-0h	W-0h
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
CCU5	CCU4	CCD5	CCD4	CCU3	CCU2	CCU1	CCU0
W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h	W-0h
7	6	5	4	3	2	1	0
CCD3	CCD2	CCD1	CCD0	RESERVED		L	Z
W-0h	W-0h	W-0h	W-0h	W-0h		W-0h	W-0h

表 23-48. ICLR 字段说明

位	字段	类型	复位	说明
31-29	RESERVED	W	0h	
28	QEIERR	W	0h	QEIERR 事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
27	直流	W	0h	方向更改事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
26	REPC	W	0h	重复计数器归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
25	TOV	W	0h	触发溢出事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
24	F	W	0h	故障事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
23-16	RESERVED	W	0h	
15	CCU5	W	0h	比较递增事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
14	CCU4	W	0h	比较递增事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除
13	CCD5	W	0h	比较递减事件 5 清除 0h = 写入 0 不产生影响。 1h = 事件清除
12	CCD4	W	0h	比较递减事件 4 清除 0h = 写入 0 不产生影响。 1h = 事件清除

表 23-48. ICLR 字段说明 (continued)

位	字段	类型	复位	说明
11	CCU3	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
10	CCU2	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
9	CCU1	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
8	CCU0	W	0h	捕获或比较递增事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
7	CCD3	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
6	CCD2	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
5	CCD1	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
4	CCD0	W	0h	捕获或比较递减事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
3-2	RESERVED	W	0h	
1	L	W	0h	加载事件清除 0h = 写入 0 不产生影响。 1h = 事件清除
0	Z	W	0h	归零事件清除 0h = 写入 0 不产生影响。 1h = 事件清除

23.3.29 EVT_MODE（偏移 = 10E0h）[复位 = 00000029h]

图 23-69 展示了 EVT_MODE，表 23-49 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线路

图 23-69. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		EVT2_CFG		EVT1_CFG		EVT0_CFG	
R/W-0h		R-2h		R-2h		R-1h	

表 23-49. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5-4	EVT2_CFG	R	2h	对应于 GEN_EVENT1 的事件的事件线路模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
3-2	EVT1_CFG	R	2h	对应于 GEN_EVENT0 的事件的事件线路模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	EVT0_CFG	R	1h	CPU_INT 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线路处于软件模式。软件必须清除 RIS。 2h = 中断或事件线路处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

23.3.30 DESC（偏移 = 10FCh）[复位 = 11110000h]

图 23-70 展示了 DESC，表 23-50 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 23-70. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-1111h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-				R-				R-				R-			

表 23-50. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	1111h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数 0h = 最小值 Fh = 尽可能高的值
7-4	MAJREV	R	0h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

23.3.31 CCPD (偏移 = 1100h) [复位 = 00000000h]

图 23-71 展示了 CCPD，表 23-51 中对此进行了介绍。

返回到汇总表。

CCP 方向。控制 CCP 是用作输入还是输出。

图 23-71. CCPD

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED				C0CCP3	C0CCP2	C0CCP1	C0CCP0
R/W-				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 23-51. CCPD 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	C0CCP3	R/W	0h	CCP3 方向 0h = 输入 1h = 输出
2	C0CCP2	R/W	0h	CCP2 方向 0h = 输入 1h = 输出
1	C0CCP1	R/W	0h	CCP1 方向 0h = 输入 1h = 输出
0	C0CCP0	R/W	0h	CCP0 方向 0h = 输入 1h = 输出

23.3.32 ODIS（偏移 = 1104h）[复位 = 00000000h]

图 23-72 展示了 ODIS，表 23-52 中对此进行了介绍。

返回到汇总表。

ODIS 寄存器输出反相，然后与 OCTL 寄存器 CCPO 字段选择的输出信号进行“与”运算（在条件反相之前），从而使软件能够在配置或关闭期间将 CCP 输出保持在低电平。

图 23-72. ODIS

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED				C0CCP3	C0CCP2	C0CCP1	C0CCP0
R/W-				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 23-52. ODIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	C0CCP3	R/W	0h	计数器 CCP3 禁用屏蔽 定义计数器 n 的 CCP3 是否被强制设置为低电平 0h = OCTL 寄存器 CCPO 字段选择的输出函数被提供给输出反转块。 1h = CCP 输出被强制设置为低电平。
2	C0CCP2	R/W	0h	计数器 CCP2 禁用屏蔽 定义计数器 n 的 CCP2 是否被强制设置为低电平 0h = OCTL 寄存器 CCPO 字段选择的输出函数被提供给输出反转块。 1h = CCP 输出被强制设置为低电平。
1	C0CCP1	R/W	0h	计数器 CCP1 禁用屏蔽 定义计数器 n 的 CCP0 是否被强制设置为低电平 0h = OCTL 寄存器 CCPO 字段选择的输出函数被提供给输出反转块。 1h = CCP 输出被强制设置为低电平。
0	C0CCP0	R/W	0h	计数器 CCP0 禁用屏蔽 定义计数器 n 的 CCP0 是否被强制设置为低电平 0h = OCTL 寄存器 CCPO 字段选择的输出函数被提供给输出反转块。 1h = CCP 输出被强制设置为低电平。

23.3.33 CCLKCTL（偏移 = 1108h）[复位 = 00000000h]

图 23-73 展示了 CCLKCTL，表 23-53 中对此进行了介绍。

返回到汇总表。

CCLKCTL 寄存器提供了一种 SW 机制，用于在预计不使用模块
但电源域处于运行状态时门控 TIMER 时钟。
这会有效地将 IP 置于 IDLE 状态

图 23-73. CCLKCTL

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

表 23-53. CCLKCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	CLKEN	R/W	0h	时钟使能 禁用模块的时钟门控。SW 必须将该值显式编程为 0 以门控时钟。 0h = 时钟被禁用。 1h = 时钟被启用

23.3.34 CPS（偏移 = 110Ch）[复位 = 00000000h]

图 23-74 展示了 CPS，表 23-54 中对此进行了介绍。

返回到汇总表。

CPS 寄存器提供时钟预分频器的值。

图 23-74. CPS

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

表 23-54. CPS 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	PCNT	R/W	0h	预分频计数 该字段指定预分频计数值。所选 TIMCLK 源按值 (PCNT+1) 分频。 PCNT 值 0 将 TIMCLK 按 1 分频，从而有效地绕过分频器。 大于 0 的 PCNT 值将 TIMCLK 源分频，从而生成更慢的时钟 0h = 最小值 FFh = 最大值

23.3.35 CPSV（偏移 = 1110h）[复位 = 00000000h]

图 23-75 展示了 CPSV，表 23-55 中对此进行了介绍。

返回到汇总表。

CPSV 寄存器提供读取当前时钟预分频计数值的功能。

图 23-75. CPSV

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

表 23-55. CPSV 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	CPSVAL	R	0h	当前预分频计数值 0h = 最小值 FFh = 最大值

23.3.36 CTTRIGCTL（偏移 = 1114h）[复位 = 00000000h]

图 23-76 展示了 CTTRIGCTL，表 23-56 中对此进行了介绍。

返回到汇总表。

交叉计时器触发控制寄存器
该寄存器用于控制同一电源域中不同计时器实例的使能和故障的交叉触发器连接。有关详细信息，请参阅“计时器模块交叉触发器（输入/输出）”和“故障交叉触发”部分。

图 23-76. CTTRIGCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED				EVTCTTRIGSEL			
R/W-				R/W-0h			
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED						EVTCTEN	CTEN
R/W-						R/W-0h	R/W-0h

表 23-56. CTTRIGCTL 字段说明

位	字段	类型	复位	描述
31-20	RESERVED	R/W	0h	
19-16	EVTCTTRIGSEL	R/W	0h	用于选择应用于输入交叉触发器的订阅者端口。 0h = 使用 FSUB0 作为交叉触发源。 1h = 使用 FSUB1 作为交叉触发源。 2h = 使用归零事件作为交叉触发源。 3h = 使用加载事件作为交叉触发源。 4h = 使用 CCD0 事件作为交叉触发源。 5h = 使用 CCD1 事件作为交叉触发源。 6h = 使用 CCD2 事件作为交叉触发源。 7h = 使用 CCD3 事件作为交叉触发源。 8h = 使用 CCU0 事件作为交叉触发源。 9h = 使用 CCU1 事件作为交叉触发源。 Ah = 使用 CCU2 事件作为交叉触发源。 Bh = 使用 CCU3 事件作为交叉触发源。
15-2	保留	R/W	0h	
1	EVTCTEN	R/W	0h	启用计时器模块的输入触发器条件作为交叉触发器的条件。 0h = 交叉触发器生成被禁用。 1h = 交叉触发器生成被启用
0	CTEN	R/W	0h	计时器交叉触发器使能。该字段用于指定 SW 或 HW 逻辑是否可以在系统中生成计时器交叉触发器事件。 这些交叉触发器连接到 SOC 电源域中其他计时器 IP 的相应计时器触发器输入。 计时器交叉触发器本质上是控制 CTRCTL 寄存器中 EN 位的 HW 与 SW 条件的组合逻辑。 0h = 交叉触发器生成被禁用。 1h = 交叉触发器生成被启用

23.3.37 CTTRIG（偏移 = 111Ch）[复位 = 00000000h]

图 23-77 展示了 CTTRIG，表 23-57 中对此进行了介绍。

返回到汇总表。

交叉计时器触发器寄存器
该寄存器用于触发使用 CTTRIGCTL 和 CTTRIGMSK 寄存器连接和启用的计时器实例。

图 23-77. CTTRIG

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

表 23-57. CTTRIG 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	
0	TRIG	W	0h	生成交叉触发器 该位在编程时将生成所有交叉触发器启用的计时器实例（包括当前计时器实例）的同步触发条件。 0h = 交叉触发器生成被禁用 1h = 生成交叉触发器脉冲

23.3.38 FSCTL（偏移 = 1120h）[复位= 00000000h]

图 23-78 展示了 FSCTL，表 23-58 中对此进行了介绍。

返回到汇总表。

FSCTL 寄存器控制故障源的选择和使能。有 5 个输入故障源（通过同步路径处理或异步路径处理）。

图 23-78. FSCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED	FEX2EN	FEX1EN	FEX0EN	FAC2EN	FAC1EN	FAC0EN	FCEN
R/W-	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 23-58. FSCTL 字段说明

位	字段	类型	复位	说明
31-7	保留	R/W	0h	
6	FEX2EN	R/W	0h	该字段控制故障是否由外部故障引脚 2 引起。 0h = 禁用 1h = 启用
5	FEX1EN	R/W	0h	该字段控制故障是否由外部故障引脚 1 引起。 0h = 禁用 1h = 启用
4	FEX0EN	R/W	0h	该字段控制故障是否由外部故障引脚 0 引起。 0h = 禁用 1h = 启用
3	FAC2EN	R/W	0h	该字段控制故障是否由 COMP2 输出引起。 0h = 禁用 1h = 启用
2	FAC1EN	R/W	0h	该字段控制故障是否由 COMP1 输出引起。 0h = 禁用 1h = 启用
1	FAC0EN	R/W	0h	该字段控制故障信号是否由 COMP0 输出引起。 0h = 禁用 1h = 启用
0	FCEN	R/W	0h	该字段控制故障是否由系统时钟故障引起。 0h = 禁用 1h = 启用

23.3.39 GCTL（偏移 = 1124h）[复位 = 00000001h]

图 23-79 展示了 GCTL，表 23-59 中对此进行了介绍。

返回到汇总表。

全局控制寄存器

图 23-79. GCTL

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

表 23-59. GCTL 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	SHDWLDEN	R/W	1h	启用缓冲寄存器和寄存器字段的影子到活动加载。 0h = 禁用 1h = 启用

23.3.40 CTR (偏移 = 1800h) [复位= 00000000h]

图 23-80 展示了 CTR，表 23-60 中对此进行了介绍。

返回到汇总表。

这是 TIMER 计数器寄存器。
这可由 SW 进行设置。不过，如果软件试图在计数器运行时
设置某个值，则写入将不可预测。

图 23-80. 计数器

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

表 23-60. CTR 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	CCTR	R/W	0h	当前计数器值 0h = 最小值 00FFFFFFh = 最大值

23.3.41 CTRCTL（偏移 = 1804h）[复位 = 0000FF80h]

图 23-81 展示了 CTRCTL，表 23-61 中对此进行了介绍。

返回到汇总表。

该寄存器提供对计数器操作的控制。
配置可以更改，也可以在单次写入中设置 EN 位。
无需先更改配置，然后执行额外的
写入来设置 EN 位。

图 23-81. CTRCTL

31	30	29	28	27	26	25	24
RESERVED		CVAE		RESERVED			PLEN
R/W-0h		R/W-0h		R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
SLZERCNEZ	RESERVED			FRB	FB	DRB	RESERVED
R/W-0h	R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
CZC				CAC		CLC	
R/W-7h				R/W-7h		R/W-7h	
7	6	5	4	3	2	1	0
CLC	RESERVED	CM		REPEAT			EN
R/W-7h	R/W-0h	R/W-0h		R/W-0h			R/W-0h

表 23-61. CTRCTL 字段说明

位	字段	类型	复位	说明
31-30	RESERVED	R/W	0h	
29-28	CVAE	R/W	0h	启用后计数器值。该字段指定通过写入 CTRCTL 寄存器使 EN 位从 0 变为 1 时计数器的初始化条件。请注意，外部事件也可能使 EN 位激活。 0h = 计数器被设置为 LOAD 寄存器值 1h = 计数器值保持为当前值，该值可能已由软件初始化 2h = 计数器被设置为零
27-25	保留	R/W	0h	
24	PLEN	R/W	0h	相负载使能。该位允许计时器具有相负载功能。 0h = 禁用 1h = 启用
23	SLZERCNEZ	R/W	0h	如果重复计数器不等于零，则抑制加载和归零事件。 当重复计数器 (RC) 值不为 0 时， 该位抑制计数器产生 Z（零） 和 L（加载）事件。 0h = 禁用。Z 和 L 事件总是在它们的条件生成时从计数器生成。 1h = 启用。Z 和 L 事件在它们的条件生成且 RC 寄存器值为 0 时从计数器生成。
22-20	RESERVED	R/W	0h	
19	FRB	R/W	0h	故障恢复行为。该位指定器件在故障条件释放/退出后执行的操作。 0h = 恢复计数 1h = 执行 CVAE 字段指定的操作。
18	FB	R/W	0h	故障行为。该位指定计数器在故障模式期间是继续运行还是暂停。 REPEAT 下有一个单独的控制位表示是否在下一次计数器 == 0 时暂停计数 0h = 继续计数 1h = 暂停计数

表 23-61. CTRCTL 字段说明 (continued)

位	字段	类型	复位	说明
17	DRB	R/W	0h	调试恢复行为。该位指定器件在调试模式释放/退出后执行的操作。 0h = 恢复计数 1h = 执行 CVAE 字段指定的操作。
16	RESERVED	R/W	0h	
15-13	CZC	R/W	7h	计数器零控制。该字段指定由什么来控制关于将计数器值归零的计数器操作。 基于 CCPC 参数值 存在编码 1-3。基于 HQEI 参数值 存在位 4-5。任何未提供的编码 均被记录为保留。 0h = CCCTL_0 ZCOND 1h = CCCTL_1 ZCOND 2h = CCCTL_2 ZCOND 当有 4 个通道时存在该值。 3h = CCCTL_3 ZCOND 当有 4 个通道时存在该值。 4h = 由 2 输入 QEI 模式控制 当 TIMER 支持 QEI 功能时存在该值。 5h = 由 3 输入 QEI 模式控制 。当 TIMER 支持 QEI 功能时存在该值。
12-10	CAC	R/W	7h	计数器计数控制。该字段指定由什么来控制关于计数器值进展（递增或递减）的计数器操作。 基于 CCPC 参数值 存在编码 1-3。基于 HQEI 参数值 存在位 4-5。任何未提供的编码 均被记录为保留。 0h = CCCTL_0 ACOND 1h = CCCTL_1 ACOND 2h = CCCTL_2 ACOND 当有 4 个通道时存在该值。 3h = CCCTL_3 ACOND 当有 4 个通道时存在该值。 4h = 由 2 输入 QEI 模式控制 当 TIMER 支持 QEI 功能时存在该值。 5h = 由 3 输入 QEI 模式控制 。当 TIMER 支持 QEI 功能时存在该值。
9-7	CLC	R/W	7h	计数器加载控制。该字段指定由什么来控制关于将计数器设置为 LD 寄存器值的计数器操作。 基于 CCPC 参数值 存在编码 1-3。基于 HQEI 参数值 存在位 4-5。任何未提供的编码 均被记录为保留。 0h = CCCTL_0 LCOND 1h = CCCTL_1 LCOND 2h = CCCTL_2 LCOND 当有 4 个通道时存在该值。 3h = CCCTL_3 LCOND 当有 4 个通道时存在该值。 4h = 由 2 输入 QEI 模式控制。 当 TIMER 支持 QEI 功能时存在该值。 5h = 由 3 输入 QEI 模式控制。 当 TIMER 支持 QEI 功能时存在该值。
6	RESERVED	R/W	0h	
5-4	CM	R/W	0h	计数模式 0h = 递减 1h = 递增/递减 2h = 计数器递增计数。

表 23-61. CTRCTL 字段说明 (continued)

位	字段	类型	复位	说明
3-1	REPEAT	R/W	0h	重复。重复位控制计数器是否在归零事件或调试或故障条件退出后继续计数。对于向下计数，发生归零事件后会在下次满足计数条件时执行加载。对于向上/向下计数，发生归零事件后会执行计数 (+1)。编码 3 的目的是，如果调试条件生效，则延迟生成加载脉冲，直到调试条件结束。这允许计数器在暂停计数之前达到零。 0h = 在归零事件后不自动计数。 1h = 在归零事件后继续计数。 2h = 保留 3h = 如果调试模式未生效，或在调试模式释放后，在归零事件后继续计数。 4h = 保留
0	EN	R/W	0h	计数器启用。该位允许计时器计时。如果 REPEAT=0 (不自动重新加载) 且计数器值为零，则该位自动清零。CPU 写入：设置 EN 位的寄存器写入，根据 CVAE 值设置计数器值。硬件：在满足 LCOND 或 ZCOND 条件时也会设置该位，并且计数器值分别更改为加载值或零值。 0h = 禁用 1h = 启用

23.3.42 LOAD（偏移 = 1808h）[复位 = 00000000h]

图 23-82 展示了 LOAD，表 23-62 中对此进行了介绍。

返回到汇总表。

在指定执行“加载”的任何操作中，LOAD 寄存器的内容被复制到 CTR 中。LOAD 用于与 CTR 进行比较，以便生成可用于中断、触发或信号发生器操作的“加载事件”。

图 23-82. LOAD

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

表 23-62. LOAD 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	LD	R/W	0h	加载值 0h = 最小值 00FFFFFFh = 最大值

23.3.43 CC_01[y] (偏移 = 1810h + 公式) [复位 = 00000000h]

图 23-83 展示了 CC_01[y]，表 23-63 中对此进行了介绍。

返回到汇总表。

CC_01 寄存器可用作捕获寄存器来捕获事件上的下一个 CTR 值，或用作比较寄存器来与当前 CTR 进行比较以创建事件。它不能同时作为这两者运行。每个计数器有两个捕获/比较硬件切片，因此每个计时器有两个 CC_01 寄存器。在发生捕获事件时，会加载 CTR 的下一个值，以便 CTR 和 CC_01 (捕获的) 在从捕获操作创建中断或触发事件的周期内相等。

偏移 = 1810h + (y * 4h)；其中 y = 0h 至 1h

图 23-83. CC_01[y]

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

表 23-63. CC_01[y] 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	CCVAL	R/W	0h	捕获或比较值 0h = 最小值 FFFFh = 最大值

23.3.44 CC_23[y] (偏移 = 1818h + 公式) [复位 = 00000000h]

图 23-84 展示了 CC_23[y]，表 23-64 中对此进行了介绍。

返回到汇总表。

CC_23 寄存器可用作捕获寄存器来捕获事件上的下一个 CTR 值，或用作比较寄存器来与当前 CTR 进行比较以创建事件。它不能同时作为这两者运行。每个计数器有两个捕获/比较硬件切片，因此每个计时器有两个 CC_01 寄存器。在发生捕获事件时，会加载 CTR 的下一个值，以便 CTR 和 CC_01 (捕获的) 在从捕获操作创建中断或触发事件的周期内相等。

偏移 = 1818h + (y * 4h)；其中 y = 0h 至 1h

图 23-84. CC_23[y]

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

表 23-64. CC_23[y] 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	CCVAL	R/W	0h	捕获或比较值 0h = 最小值 FFFFh = 最大值

23.3.45 CC_45[y] (偏移 = 1820h + 公式) [复位 = 00000000h]

图 23-85 展示了 CC_45[y] , 表 23-65 中对此进行了介绍。

返回到汇总表。

CC_45 寄存器是可用于与当前 CTR 进行比较以创建事件 CC4U、CC4D、CC5U 和 CC5D 的寄存器。

偏移 = 1820h + (y * 4h) ; 其中 y = 0h 至 1h

图 23-85. CC_45[y]

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

表 23-65. CC_45[y] 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	CCVAL	R/W	0h	捕获或比较值 0h = 最小值 FFFFh = 最大值

23.3.46 CCCTL_01[y] (偏移 = 1830h + 公式) [复位 = 00000000h]

图 23-86 展示了 CCCTL_01[y]，表 23-66 中对此进行了介绍。

返回到汇总表。

CCCTL_01 寄存器控制相应 CC 寄存器和计数器的操作。

偏移 = 1830h + (y * 4h)；其中 y = 0h 至 1h

图 23-86. CCCTL_01[y]

31	30	29	28	27	26	25	24
CC2SELD			CCACTUPD			SCERCNEZ	CC2SELU
R/W-0h			R/W-0h			R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
CC2SELU		RESERVED	CCUPD			COC	RESERVED
R/W-0h		R/W-0h	R/W-0h			R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED	ZCOND			RESERVED	LCOND		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
RESERVED	ACOND			RESERVED	CCOND		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		

表 23-66. CCCTL_01[y] 字段说明

位	字段	类型	复位	说明
31-29	CC2SELD	R/W	0h	选择源第二个 CCD 事件。 0h = 从 CC0 中选择 CCD。 1h = 从 CC1 中选择 CCD。 2h = 从 CC2 中选择 CCD。 3h = 从 CC3 中选择 CCD。 4h = 从 CC4 中选择 CCD。 5h = 从 CC5 中选择 CCD。

表 23-66. CCCTL_01[y] 字段说明 (continued)

位	字段	类型	复位	说明
28-26	CCACTUPD	R/W	0h	CCACT 影子寄存器更新方法 该字段控制如何执行对 CCACT 影子寄存器的更新 0h = 写入 CCACT 寄存器的值立即生效。 1h = 在归零事件 (CTR=0) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并 在 CTR 等于 0 后的 TIMCLK 周期中传输到 CCACTx_y。 2h = 在 CCD 事件 (CTR=CC_xy) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值后的 TIMCLK 周期 中传输到 CCACTx_y。 3h = 在 CCU 事件 (CTR=CC_xy) 之后 对 CCACTx_y 寄存器的 写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值 后的 TIMCLK 周期中传输到 CCACTx_y。 4h = 在归零事件 (CTR=0) 或加载事件 (CTR = LOAD) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 或 CTR 后的 TIMCLK 周期中传输到 CCACTx_y。等于 LDn。 请注意，该更新机制 被定义为仅在使用 向上/向下计数的配置中 使用。该模式不适用于递减计数 配置。 5h = 在重复计数也为零 (RC=0) 的归零事件 (CTR=0) 之后。 对 CCACTx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 0 后以及 RC 等于 0 时在 TIMCLK 周期中传输到 CCACTx_y。 6h = 在 TRIG 脉冲上，存储在 CCACT_xy 影子寄存器中的值被加载到 CCACT_xy 寄存器中。
25	SCERCNEZ	R/W	0h	如果重复计数器不等于零，则抑制比较事件 当重复计数器 (RC) 值不为 0 时，该位抑制来自 计数器的比较 (CCD、CCU 和 RC) 事件的生成。 0h = CCD、CCU 和 RC 事件总是在 它们的条件生成时 从计数器生成。 1h = CCD、CCU 和 RC 事件在它们的 条件生成 且 RC 寄存器值为 0 时从 计数器生成。

表 23-66. CCCTL_01[y] 字段说明 (continued)

位	字段	类型	复位	说明
24-22	CC2SELU	R/W	0h	选择源第二个 CCU 事件。 0h = 从 CC0 中选择 CCU。 1h = 从 CC1 中选择 CCU。 2h = 从 CC2 中选择 CCU。 3h = 从 CC3 中选择 CCU。 4h = 从 CC4 中选择 CCU。 5h = 从 CC5 中选择 CCU。
21	保留	R/W	0h	
20-18	CCUPD	R/W	0h	捕获和比较更新方法 该字段控制如何执行对影子 捕获和比较寄存器的更新 （在比较模式下运行时，COC=0）。 0h = 对 CCx_y 寄存器的写入直接写入寄存器并立即生效。 1h = 在归零事件 (CTR=0) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 0 后的 TIMCLK 周期中传输到 CCx_y。 2h = 在 CCD 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 CCx_y 寄存器 值后的 TIMCLK 周期中 传输到 CCx_y。 3h = 在 CCU 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值后的 TIMCLK 周期中传输到 CCx_y。 4h = 在归零事件 (CTR=0) 或加载事件 (CTR=LOAD) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 或 CTR 后的 TIMCLK 周期中传输到 ECCx_y。等于 LD。 请注意，该更新机制 被定义为仅在使用 向上/向下计数的配置中 使用。该模式不适用于递减计数 配置。 5h = 在重复计数也为零 (RC=0) 的归零事件 (CTR=0) 之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 后 以及 RC 等于 0 时 在 TIMCLK 周期中 传输到 CCx_y。 6h = 在 TRIG 脉冲之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中并传输到 CCx_y

表 23-66. CCCTL_01[y] 字段说明 (continued)

位	字段	类型	复位	说明
17	COC	R/W	0h	捕获或比较。 指定相应的 CC 寄存器是用作捕获寄存器还是比较寄存器（不能同时用作两者）。 0h = 比较 1h = 捕获
16-15	RESERVED	R/W	0h	
14-12	ZCOND	R/W	0h	零条件。 该字段指定生成零脉冲的条件。 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）
11	RESERVED	R/W	0h	
10-8	LCOND	R/W	0h	加载条件。 指定生成加载脉冲的条件。 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）
7	RESERVED	R/W	0h	
6-4	ACOND	R/W	0h	移动条件。 指定生成移动脉冲的条件。 0h = 每个 TIMCLK 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿） 5h = CCP 高或触发器生效（电平）
3	RESERVED	R/W	0h	
2-0	CCOND	R/W	0h	捕获条件。 指定生成捕获脉冲的条件。 0h = 无（从不捕获） 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）

23.3.47 CCCTL_23[y] (偏移 = 1838h + 公式) [复位 = 00000000h]

图 23-87 展示了 CCCTL_23[y]，表 23-67 中对此进行了介绍。

返回到汇总表。

CCCTL 寄存器控制相应 CC 寄存器和计数器的操作。

偏移 = 1838h + (y * 4h)；其中 y = 0h 至 1h

图 23-87. CCCTL_23[y]

31	30	29	28	27	26	25	24
CC2SELD			CCACTUPD			SCERCNEZ	CC2SELU
R/W-0h			R/W-0h			R/W-0h	R/W-0h
23	22	21	20	19	18	17	16
CC2SELU		RESERVED	CCUPD			COC	RESERVED
R/W-0h		R/W-0h	R/W-0h			R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
RESERVED	ZCOND			RESERVED	LCOND		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
RESERVED	ACOND			RESERVED	CCOND		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		

表 23-67. CCCTL_23[y] 字段说明

位	字段	类型	复位	说明
31-29	CC2SELD	R/W	0h	选择源第二个 CCD 事件。 0h = 从 CC0 中选择 CCD。 1h = 从 CC1 中选择 CCD。 2h = 从 CC2 中选择 CCD。 3h = 从 CC3 中选择 CCD。 4h = 从 CC4 中选择 CCD。 5h = 从 CC5 中选择 CCD。

表 23-67. CCCTL_23[y] 字段说明 (continued)

位	字段	类型	复位	说明
28-26	CCACTUPD	R/W	0h	CCACT 影子寄存器更新方法 该字段控制如何执行对 CCCACT 影子寄存器的更新 0h = 写入 CCACTx_y 寄存器的值立即生效。 1h = 在归零事件 (CTR=0) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并 在 CTR 等于 0 后的 TIMCLK 周期中传输到 CCACTx_y。 2h = 在 CCD 事件 (CTR=CC_xy) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值后的 TIMCLK 周期 中传输到 CCACTx_y。 3h = 在 CCU 事件 (CTR=cc_xy) 之后 对 CCACTx_y 寄存器的 写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值 后的 TIMCLK 周期中 传输到 CCACTx_y。 4h = 在归零事件 (CTR=0) 或加载事件 (CTR = LOAD) 之后 对 CCACTx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 或 CTR 后的 TIMCLK 周期中传输到 CCACTx_y。等于 LDn。 请注意，该更新机制 被定义为仅在使用 向上/向下计数的配置中 使用。该模式不适用于递减计数 配置。 5h = 在重复计数也为零 (RC=0) 的归零事件 (CTR=0) 之后。 对 CCACTx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 0 后以及 RC 等于 0 时在 TIMCLK 周期中传输到 CCACTx_y。 6h = 在 TRIG 脉冲上，存储在 CCACTx_y 影子寄存器中的值被加载到 CCACTx_y 活动寄存器中。
25	SCERCNEZ	R/W	0h	如果重复计数器不等于零，则抑制比较事件 当重复计数器 (RCn) 值不为 0 时，该位抑制来自计数器的 比较（CCD、CCU 和 RC）事件的生成。 0h = CCD、CCU 和 RC 事件总是在 它们的条件生成时 从计数器生成。 1h = CCD、CCU 和 RC 事件在它们的 条件生成 且 RC 寄存器值为 0 时从 计数器生成。

表 23-67. CCCTL_23[y] 字段说明 (continued)

位	字段	类型	复位	说明
24-22	CC2SELU	R/W	0h	选择源第二个 CCU 事件。 0h = 从 CC0 中选择 CCU。 1h = 从 CC1 中选择 CCU。 2h = 从 CC2 中选择 CCU。 3h = 从 CC3 中选择 CCU。 4h = 从 CC4 中选择 CCU。 5h = 从 CC5 中选择 CCU。
21	保留	R/W	0h	
20-18	CCUPD	R/W	0h	捕获和比较更新方法 该字段控制如何执行对影子 捕获和比较寄存器的更新 （在比较模式下运行时，COC=0）。 0h = 对 CCx_y 寄存器的写入直接写入寄存器并立即生效。 1h = 在归零事件 (CTR=0) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 0 后的 TIMCLK 周期中传输到 CCx_y。 2h = 在 CCD 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 CCx_y 寄存器 值后的 TIMCLK 周期中 传输到 CCx_y。 3h = 在 CCU 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值后的 TIMCLK 周期中传输到 CCx_y。 4h = 在零或加载事件之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 或 CTR 后 的 TIMCLK 周期中传输到 CCx_y。等于 LDn。 请注意，该更新机制 被定义为仅在使用 向上/向下计数的配置中 使用。该模式不适用于递减计数 配置。 5h = 在重复计数也为零 (RC=0) 的归零事件 (CTR=0) 之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 后 以及 RC 等于 0 时 在 TIMCLK 周期中 传输到 CCx_y。 6h = 在 TRIG 脉冲之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中并传输到 CCx_y #xD; 0。

表 23-67. CCCTL_23[y] 字段说明 (continued)

位	字段	类型	复位	说明
17	COC	R/W	0h	捕获或比较。 指定相应的 CC 寄存器是用作捕获寄存器还是比较寄存器（不能同时用作两者）。 0h = 比较 1h = 捕获
16-15	RESERVED	R/W	0h	
14-12	ZCOND	R/W	0h	零条件。 该字段指定生成零脉冲的条件。4h-Fh = 保留 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）
11	RESERVED	R/W	0h	
10-8	LCOND	R/W	0h	加载条件。 指定生成加载脉冲的条件。4h-Fh = 保留 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）
7	RESERVED	R/W	0h	
6-4	ACOND	R/W	0h	移动条件。 指定生成移动脉冲的条件。6h-Fh = 保留 0h = 每个 TIMCLK 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿） 5h = CCP 高或触发器生效（电平）
3	RESERVED	R/W	0h	
2-0	CCOND	R/W	0h	捕获条件。 指定生成捕获脉冲的条件。4h-Fh = 保留 0h = 无（从不捕获） 1h = CCP 的上升沿或触发器生效边沿 2h = CCP 的下降沿或触发器失效边沿 3h = CCP 边沿或触发器变化边沿（生效/失效边沿）

23.3.48 CCCTL_45[y] (偏移 = 1840h + 公式) [复位 = 00000000h]

图 23-88 展示了 CCCTL_45[y]，表 23-68 中对此进行了介绍。

返回到汇总表。

CCCTL 寄存器控制相应 CC 寄存器和计数器的操作。

偏移 = 1840h + (y * 4h)；其中 y = 0h 至 1h

图 23-88. CCCTL_45[y]

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

表 23-68. CCCTL_45[y] 字段说明

位	字段	类型	复位	说明
31-26	RESERVED	R/W	0h	
25	SCERCNEZ	R/W	0h	如果重复计数器不等于零，则抑制比较事件 当重复计数器 (RC) 值不为 0 时，该位抑制来自 计数器的比较 (CCD、CCU 和 RC) 事件的生成。 0h = CCD、CCU 和 RC 事件总是在 它们的条件生成时 从计数器生成。 1h = CCD、CCU 和 RC 事件在它们的 条件生成 且 RC 寄存器值为 0 时从 计数器生成。
24-21	RESERVED	R/W	0h	

表 23-68. CCCTL_45[y] 字段说明 (continued)

位	字段	类型	复位	说明
20-18	CCUPD	R/W	0h	捕获和比较更新方法 该字段控制如何执行对影子 捕获和比较寄存器的更新 （在比较模式下运行时，COC=0）。 0h = 对 CCx_y 寄存器的写入直接写入寄存器并立即生效。 1h = 在归零事件 (CTR=0) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 后的 TIMCLK 周期中传输到 ECCx_y。 2h = 在 CCD 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中， 并在 CTR 等于 CCx_y 寄存器 值后的 TIMCLK 周期中 传输到 CCx_y。 3h = 在 CCU 事件 (CTR=CC_xy) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 CCx_y 寄存器值后的 TIMCLK 周期中传输到 CCx_y。 4h = 在归零事件 (CTR=0) 或加载事件 (CTR=LOAD) 之后 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 或 CTR 后的 TIMCLK 周期中传输到 CCx_y。等于 LD。 请注意，该更新机制 被定义为仅在使用 向上/向下计数的配置中 使用。该模式不适用于递减计数 配置。 5h = 在重复计数也为零 (RC=0) 的归零事件 (CTR=0) 之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中，并在 CTR 等于 0 后 以及 RC 等于 0 时 在 TIMCLK 周期中 传输到 CCx_y。 6h = 在 TRIG 脉冲之后。 对 CCx_y 寄存器的写入存储在 影子寄存器中并传输到 CCx_y #xD; 0。
17-0	保留	R/W	0h	

23.3.49 OCTL_01[y] (偏移 = 1850h + 公式) [复位 = 00000000h]

图 23-89 展示了 OCTL_01[y]，表 23-69 中对此进行了介绍。

返回到汇总表。

OCTL_01 寄存器控制计数器捕获/比较切片的 CCP 输出。还能够选择驱动源以及初始条件值和最终反转选项。

偏移 = 1850h + (y * 4h)；其中 y = 0h 至 1h

图 23-89. OCTL_01[y]

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		CCPIV	CCPOINV	CCPO			
R/W-0h		R/W-0h	R/W-0h	R/W-0h			

表 23-69. OCTL_01[y] 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5	CCPIV	R/W	0h	CCP 初始值。该位指定在计数器被禁用 (CTRCTL.EN == 0) 时为信号发生器状态设置的逻辑值。 0h = 低电平 1h = 高电平
4	CCPOINV	R/W	0h	CCP 输出反相。CCPO 选择的输出被有条件地反相。 0h = 不反相 1h = 反相
3-0	CCPO	R/W	0h	CCP 输出源 0h = 信号发生器值 (例如 PWM、触发的 PWM) 1h = 加载事件 2h = CCU 事件或 CCD 事件 4h = 归零事件 5h = 捕获事件 6h = 故障条件 8h = 将第一个捕获和比较寄存器的 CCP 镜像到其他捕获比较块 9h = 其他捕获比较块中第二个捕获和比较寄存器的镜像 CCP Ch = 插入死区后的信号发生器输出 Dh = 计数器方向

23.3.50 OCTL_23[y] (偏移 = 1858h + 公式) [复位 = 00000000h]

图 23-90 展示了 OCTL_23[y]，表 23-70 中对此进行了介绍。

返回到汇总表。

OCTL 寄存器控制计数器捕获/比较切片的 CCP 输出。还能够选择驱动源以及初始条件值和最终反转选项。

偏移 = 1858h + (y * 4h)；其中 y = 0h 至 1h

图 23-90. OCTL_23[y]

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		CCPIV	CCPOINV	CCPO			
R/W-0h		R/W-0h	R/W-0h	R/W-0h			

表 23-70. OCTL_23[y] 字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5	CCPIV	R/W	0h	CCP 初始值。该位指定在计数器被禁用 (CTRCTL.EN == 0) 时为信号发生器状态设置的逻辑值。 0h = 低电平 1h = 高电平
4	CCPOINV	R/W	0h	CCP 输出反相。CCPO 选择的输出被有条件地反相。 0h = 不反相 1h = 反相
3-0	CCPO	R/W	0h	CCP 输出源 0h = 信号发生器值 (例如 PWM、触发的 PWM) 1h = 加载条件 2h = CCU 事件或 CCD 事件 4h = 归零事件 5h = 捕获事件 6h = 故障条件 8h = 其他捕获比较块中第一个捕获和比较寄存器的镜像 CCP 9h = 其他捕获比较块中第二个捕获和比较寄存器的镜像 CCP。 Ch = 死区插入输出 Dh = 计数器方向

23.3.51 CCACT_01[y] (偏移 = 1870h + 公式) [复位 = 00000000h]

图 23-91 展示了 CCACT_01[y]，表 23-71 中对此进行了介绍。

返回到汇总表。

CCACT_01 寄存器根据在计数器块、捕捉和比较块以及调试事件中创建的事件来控制捕捉/比较切片的信号发生器的操作。

偏移 = 1870h + (y * 4h)；其中 y = 0h 至 1h

图 23-91. CCACT_01[y]

31	30	29	28	27	26	25	24
SWFRCACT_CMPL		SWFRCACT		FEXACT			FENACT
R/W-0h		R/W-0h		R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
FENACT		RESERVED					CC2UACT
R/W-0h		R/W-0h					R/W-0h
15	14	13	12	11	10	9	8
CC2UACT	RESERVED	CC2DACT		RESERVED	CUACT		RESERVED
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
CDACT		RESERVED	LACT		RESERVED	ZACT	
R/W-0h		R/W-0h	R/W-0h		R/W-0h	R/W-0h	

表 23-71. CCACT_01[y] 字段说明

位	字段	类型	复位	说明
31-30	SWFRCACT_CMPL	R/W	0h	软件强制输出时的 CCP 互补输出操作 该字段说明软件强制的结果操作。 该操作有一个影子寄存器，它会在特定条件下被更新。 因此该寄存器不能立即生效。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 互补输出值被设置为高电平 2h = CCP 互补输出值被设置为低电平
29-28	SWFRCACT	R/W	0h	软件强制输出时的 CCP 输出操作 该字段说明软件强制的结果操作。 该操作有一个影子寄存器，它会在特定条件下被更新。 因此该寄存器不能立即生效。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平
27-25	FEXACT	R/W	0h	故障退出时的 CCP 输出操作 该字段说明信号发生器在退出故障条件时的结果操作。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换 4h = CCP 输出值被设置为三态

表 23-71. CCACT_01[y] 字段说明 (continued)

位	字段	类型	复位	说明
24-22	FENACT	R/W	0h	故障进入时的 CCP 输出操作 该字段说明信号发生器在检测到故障时的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换 4h = CCP 输出值被设置为三态
21-17	保留	R/W	0h	
16-15	CC2UACT	R/W	0h	发生 CC2U 事件时的 CCP 输出操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
14	保留	R/W	0h	
13-12	CC2DACT	R/W	0h	发生 CC2D 事件时的 CCP 输出操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
11	RESERVED	R/W	0h	
10-9	CUACT	R/W	0h	比较（递增）时的 CCP 输出操作。该字段说明信号发生器在向上计数时检测到比较事件后的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
8	保留	R/W	0h	
7-6	CDACT	R/W	0h	比较（递减）时的 CCP 输出操作。该字段说明信号发生器在向下计数时检测到比较事件后的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
5	RESERVED	R/W	0h	
4-3	LACT	R/W	0h	加载时的 CCP 输出操作。指定 CCP 输出因加载事件而发生的变化。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
2	RESERVED	R/W	0h	
1-0	ZACT	R/W	0h	零时的 CCP 输出操作。指定 CCP 输出因归零事件而发生的变化。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换

23.3.52 CCACT_23[y] (偏移 = 1878h + 公式) [复位 = 00000000h]

图 23-92 展示了 CCACT_23[y]，表 23-72 中对此进行了介绍。

返回到汇总表。

CCACT 寄存器根据在计数器块、捕捉和比较块以及调试事件中创建的事件来控制捕捉/比较切片的信号发生器的操作。

偏移 = 1878h + (y * 4h)；其中 y = 0h 至 1h

图 23-92. CCACT_23[y]

31	30	29	28	27	26	25	24
SWFRCACT_CMPL		SWFRCACT		FEXACT			FENACT
R/W-0h		R/W-0h		R/W-0h			R/W-0h
23	22	21	20	19	18	17	16
FENACT		RESERVED					CC2UACT
R/W-0h		R/W-0h					R/W-0h
15	14	13	12	11	10	9	8
CC2UACT	RESERVED	CC2DACT		RESERVED	CUACT		RESERVED
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h		R/W-0h
7	6	5	4	3	2	1	0
CDACT		RESERVED	LACT		RESERVED	ZACT	
R/W-0h		R/W-0h	R/W-0h		R/W-0h	R/W-0h	

表 23-72. CCACT_23[y] 字段说明

位	字段	类型	复位	说明
31-30	SWFRCACT_CMPL	R/W	0h	软件强制输出时的 CCP 互补输出操作 该字段说明软件强制的结果操作。 该操作有一个影子寄存器，它会在特定条件下被更新。 因此该寄存器不能立即生效。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 互补输出值被设置为高电平 2h = CCP 互补输出值被设置为低电平
29-28	SWFRCACT	R/W	0h	软件强制输出时的 CCP 输出操作 该字段说明软件强制的结果操作。 该操作有一个影子寄存器，它会在特定条件下被更新。 因此该寄存器不能立即生效。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平
27-25	FEXACT	R/W	0h	故障退出时的 CCP 输出操作 该字段说明信号发生器在退出故障条件时的结果操作。 0h = 该事件被禁用并选择优先级较低的事件 (如果生效)。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换 4h = CCP 输出值被设置为三态

表 23-72. CCACT_23[y] 字段说明 (continued)

位	字段	类型	复位	说明
24-22	FENACT	R/W	0h	故障进入时的 CCP 输出操作 该字段说明信号发生器在检测到故障时的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换 4h = CCP 输出值被设置为三态
21-17	保留	R/W	0h	
16-15	CC2UACT	R/W	0h	发生 CC2U 事件时的 CCP 输出操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
14	保留	R/W	0h	
13-12	CC2DACT	R/W	0h	发生 CC2D 事件时的 CCP 输出操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
11	RESERVED	R/W	0h	
10-9	CUACT	R/W	0h	比较（递增）时的 CCP 输出操作。该字段说明信号发生器在向上计数时检测到比较事件后的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
8	保留	R/W	0h	
7-6	CDACT	R/W	0h	比较（递减）时的 CCP 输出操作。该字段说明信号发生器在向下计数时检测到比较事件后的结果操作。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
5	RESERVED	R/W	0h	
4-3	LACT	R/W	0h	加载时的 CCP 输出操作。指定 CCP 输出因加载事件而发生的变化。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换
2	RESERVED	R/W	0h	
1-0	ZACT	R/W	0h	零时的 CCP 输出操作。指定 CCP 输出因归零事件而发生的变化。 0h = 该事件被禁用并选择优先级较低的事件（如果生效）。CCP 输出值不受事件的影响。 1h = CCP 输出值被设置为高电平 2h = CCP 输出值被设置为低电平 3h = CCP 输出值被切换

23.3.53 IFCTL_01[y] (偏移 = 1880h + 公式) [复位 = 00000000h]

图 23-93 展示了 IFCTL_01[y]，表 23-73 中对此进行了介绍。

返回到汇总表。

IFCTL_01 寄存器控制相关捕获/比较切片的输入选择和反相。

偏移 = 1880h + (y * 4h)；其中 y = 0h 至 1h

图 23-93. IFCTL_01[y]

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED			FE	CPV	RESERVED	FP	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
INV	RESERVED			ISEL			
R/W-0h	R/W-0h			R/W-0h			

表 23-73. IFCTL_01[y] 字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	FE	R/W	0h	滤波器使能 该位控制输入是被输入 滤波器滤波还是旁路到边沿 检测。 0h = 旁路。 1h = 滤波。
11	CPV	R/W	0h	连续周期/表决选择 该位控制输入滤波器是使用 更严格的连续周期计数还是多数 表决。 0h = 连续周期 在将输入传递到滤波器输出之前，输入必须在 FP 定义的周期内处于 特定的逻辑电平。 1h = 表决 滤波器在滤波器周期内 忽略一个相反逻辑的 时钟。 即在输入的 FP 样本中， 最多 1 个样本可能 具有相反的逻辑值 (干扰) 而不影响输出。
10	RESERVED	R/W	0h	
9-8	FP	R/W	0h	滤波器周期。该字段指定输入滤波器的 采样周期。即输入在滤波期间在 FP 个 计时器时钟上被采样。 0h = 分频因子为 3 1h = 分频因子为 5 2h = 分频因子为 8

表 23-73. IFCTL_01[y] 字段说明 (continued)

位	字段	类型	复位	说明
7	INV	R/W	0h	输入反相。该位控制所选输入是否反相。 0h = 同相 1h = 反相
6-4	保留	R/W	0h	
3-0	ISEL	R/W	0h	输入选择 (CCP0)。该字段选择滤波器输入的输入源。4h-7h = 保留 0h = 相应捕获比较单元的 CCP 1h = 捕获比较单元的输入对 CCPX。对于 CCP0，输入对为 CCP1；对于 CCP1，输入对为 CCP0。 2h = 计数器的 CCP0 3h = 触发器 4h = 作为输入源的 CCP 输入异或 (用在霍尔输入模式中) 。 5h = 作为输入源的订阅者 0 事件。 6h = 作为输入源的订阅者 1 事件。 7h = 比较器 0 输出。 8h = 比较器 1 输出。 9h = 比较器 2 输出。

23.3.54 IFCTL_23[y] (偏移 = 1888h + 公式) [复位 = 00000000h]

图 23-94 展示了 IFCTL_23[y]，表 23-74 中对此进行了介绍。

返回到汇总表。

IFCTL 寄存器控制相关捕获/比较切片的输入选择和反相。

偏移 = 1888h + (y * 4h)；其中 y = 0h 至 1h

图 23-94. IFCTL_23[y]

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED			FE	CPV	RESERVED	FP	
R/W-0h			R/W-0h	R/W-0h	R/W-0h	R/W-0h	
7	6	5	4	3	2	1	0
INV	RESERVED			ISEL			
R/W-0h	R/W-0h			R/W-0h			

表 23-74. IFCTL_23[y] 字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	FE	R/W	0h	滤波器使能 该位控制输入是被输入 滤波器滤波还是旁路到边沿 检测。 0h = 旁路。 1h = 滤波。
11	CPV	R/W	0h	连续周期/表决选择 该位控制输入滤波器是使用 更严格的连续周期计数还是多数 表决。 0h = 连续周期 在将输入传递到滤波器输出之前，输入必须在 FP 定义的周期内处于 特定的逻辑电平。 1h = 表决 滤波器在滤波器周期内 忽略一个相反逻辑的 时钟。 即在输入的 FP 样本中， 最多 1 个样本可能 具有相反的逻辑值 (干扰) 而不影响输出。
10	RESERVED	R/W	0h	
9-8	FP	R/W	0h	滤波器周期。该字段指定输入滤波器的 采样周期。即输入在滤波期间在 FP 个 计时器时钟上被采样。 0h = 分频因子为 3 1h = 分频因子为 5 2h = 分频因子为 8

表 23-74. IFCTL_23[y] 字段说明 (continued)

位	字段	类型	复位	说明
7	INV	R/W	0h	输入反相。该位控制所选输入是否反相。 0h = 同相 1h = 反相
6-4	保留	R/W	0h	
3-0	ISEL	R/W	0h	输入选择 (CCP0)。该字段选择滤波器输入的输入源。4h-7h = 保留 0h = 相应捕获比较单元的 CCP 1h = 捕获比较单元的输入对 CCPX。对于 CCP0，输入对为 CCP1；对于 CCP1，输入对为 CCP0。 2h = 计数器的 CCP0 3h = 触发器 4h = 作为输入源的 CCP 输入异或 (用在霍尔输入模式中) 。 5h = 作为输入源的订阅者 0 事件。 6h = 作为输入源的订阅者 1 事件。 7h = 比较器 0 输出。 8h = 比较器 1 输出。 9h = 比较器 2 输出。

23.3.55 PL (偏移 = 18A0h) [复位 = 00000000h]

图 23-95 展示了 PL，表 23-75 中对此进行了介绍。

返回到汇总表。

这是相位加载寄存器。

图 23-95. PL

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

表 23-75. PL 字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15-0	PHASE	R/W	0h	相位加载值 0h = 最小值 00FFFFFFh = 最大值

23.3.56 DBCTL（偏移 = 18A4h）[复位= 00000000h]

图 23-96 展示了 DBCTL，表 23-76 中对此进行了介绍。

返回到汇总表。

DBCTL 寄存器控制脉宽调制输出的死区插入。

图 23-96. DBCTL

31	30	29	28	27	26	25	24
RESERVED				FALLDELAY			
R/W-				R/W-0h			
23	22	21	20	19	18	17	16
FALLDELAY							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED			M1_ENABLE	RISEDELAY			
R/W-			R/W-	R/W-0h			
7	6	5	4	3	2	1	0
RISEDELAY							
R/W-0h							

表 23-76. DBCTL 字段说明

位	字段	类型	复位	说明
31-28	RESERVED	R/W	0h	
27-16	FALLDELAY	R/W	0h	下降延迟 CCP 信号的下降沿和 CCP 互补信号的上升沿之间插入的 TIMCLK 周期数 0h = 最小值 FFFh = 最大值
15-13	RESERVED	R/W	0h	
12	M1_ENABLE	R/W	0h	死区模式 1 启用。 0h = 禁用 1h = 启用
11-0	RISEDELAY	R/W	0h	上升延迟 CCP 信号的下降沿和 CCP 互补信号的上升沿之间插入的 TIMCLK 周期数 0h = 最小值 FFFh = 最大值

23.3.57 TSEL（偏移 = 18B0h）[复位 = 00000000h]

图 23-97 展示了 TSEL，表 23-77 中对此进行了介绍。

返回到汇总表。

TSEL 寄存器控制输入触发器启用和触发源的选择。触发源由其他外设通过其各自的发布者端口（由计时器的订阅者端口订阅）生成。

图 23-97. TSEL

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

表 23-77. TSEL 字段说明

位	字段	类型	复位	说明
31-10	RESERVED	R/W	0h	
9	TE	R/W	0h	触发器启用。 这选择是否为该计数器启用触发器 0x0 = 不使用触发器 0x1 = 按照 ETSEL 字段的选择使用触发器 0h = 不使用触发器。 1h = 按照 IE、ITSEL 和 ETSEL 字段的选择使用触发器。
8-5	RESERVED	R/W	0h	
4-0	ETSEL	R/W	0h	外部触发器选择。 这将选择在输入滤波器选择触发器时使用哪个系统事件。 触发器 0-15 用于连接由其他计时器模块生成的触发器。有关与计时器触发源相关的详细信息，请参阅 SoC 数据表。 触发器 16 和 17 连接到事件管理器订阅者端口。 事件线路 18-31 保留供将来使用。 0h = TRIGx =来自 TIM x 的外部触发器输入。 1h = TRIGx =来自 TIM x 的外部触发器输入。 2h = TRIGx =来自 TIM x 的外部触发器输入。 3h = TRIGx = 来自 TIM x 的外部触发器输入。 4h = TRIGx = 来自 TIM x 的外部触发器输入。 5h = TRIGx = 来自 TIM x 的外部触发器输入。 6h = TRIGx = 来自 TIM x 的外部触发器输入。 7h = TRIGx = 来自 TIM x 的外部触发器输入。 8h = TRIGx = 来自 TIM x 的外部触发器输入。 9h = TRIGx = 来自 TIM x 的外部触发器输入。 Ah = TRIGx = 来自 TIM x 的外部触发器输入。 Bh = TRIGx = 来自 TIM x 的外部触发器输入。 Ch = TRIGx = 来自 TIM x 的外部触发器输入。 Dh = TRIGx = 来自 TIM x 的外部触发器输入。 Eh = TRIGx = 来自 TIM x 的外部触发器输入。 Fh = TRIGx = 来自 TIM x 的外部触发器输入。 10h = TRIG_SUBx = 来自订阅者端口 x 的外部触发器输入。 11h = TRIG_SUBx = 来自订阅者端口 x 的外部触发器输入。

23.3.58 RC（偏移 = 18B4h）[复位 = 00000000h]

图 23-98 展示了 RC，表 23-78 中对此进行了介绍。

返回到汇总表。

重复计数器用于减少中断开销。重复计数器提供了抑制不必要中断的机制；对于程序周期数，将每个事件类型生成的中断数减少为 1。具体来说，重复计时器可以抑制加载、比较（递增/递减、正常/影子）和归零事件。

图 23-98. RC

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

表 23-78. RC 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	RC	R	0h	重复计数器值 0h = 最小值 FFh = 最大值

23.3.59 RCLD（偏移 = 18B8h）[复位 = 00000000h]

图 23-99 展示了 RCLD，表 23-79 中对此进行了介绍。

返回到汇总表。

当计数器加载输入有效时，加载寄存器值被传输到计数器。

图 23-99. RCLD

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

表 23-79. RCLD 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-0	RCLD	R/W	0h	重复计数器加载值 该字段提供在重复计数器值 等于 0 之后的加载事件中 加载到重复计数器中的值。 0h = 最小值 FFh = 最大值

23.3.60 QDIR（偏移 = 18BCh）[复位 = 00000000h]

图 23-100 展示了 QDIR，表 23-80 中对此进行了介绍。

返回到汇总表。

QDIR 寄存器提供计数方向，供在 QEI 中操作计数器时使用。

图 23-100. QDIR

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

表 23-80. QDIR 字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	DIR	R	0h	计数方向 0h = 向下（B 相在 A 相前） 1h = 向上（A 相在 B 相前）

23.3.61 FCTL（偏移 = 18D0h）[复位= 00000000h]

图 23-101 展示了 FCTL，表 23-81 中对此进行了介绍。

返回到汇总表。

FCTL 寄存器控制故障输入、故障检测和错误处理行为。

图 23-101. FCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED		FSENEXT2	FSENEXT1	FSENEXT0	FSENAC2	FSENAC1	FSENAC0
R/W-		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h
7	6	5	4	3	2	1	0
TFIM	RESERVED		FL		FI	RESERVED	FIEN
R/W-0h	R/W-		R/W-0h		R/W-0h	R/W-	R/W-0h

表 23-81. FCTL 字段说明

位	字段	类型	复位	说明
31-14	保留	R/W	0h	
13	FSENEXT2	R/W	0h	指定是否将外部故障 pin2 高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
12	FSENEXT1	R/W	0h	指定是否将外部故障 pin1 高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
11	FSENEXT0	R/W	0h	指定是否将外部故障 pin0 高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
10	FSENAC2	R/W	0h	指定是否将 COMP2 输出高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
9	FSENAC1	R/W	0h	指定是否将 COMP1 输出高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
8	FSENAC0	R/W	0h	指定是否将 COMP0 输出高电平/低电平视为故障条件。 0h = 故障输入为低电平有效。 1h = 故障输入为高电平有效。
7	TFIM	R/W	0h	触发器故障输入屏蔽 指定所选触发器是否作为故障输入参与。如果启用并且触发器有效，则触发器被视为故障。 0h = 所选触发器不参与故障条件生成 1h = 所选触发器参与故障条件生成
6-5	RESERVED	R/W	0h	

表 23-81. FCTL 字段说明 (continued)

位	字段	类型	复位	说明
4-3	FL	R/W	0h	故障锁存模式 指定是否锁存故障条件并配置锁存器清除条件。 0h = 整体故障条件不取决于 RIS 中的 F 位 1h = 整体故障条件取决于 RIS 中的 F 位 2h = 锁存故障条件。如果故障输入为 0，则在发生归零事件时清除故障条件。 3h = 锁存故障条件。如果故障输入为 0，则在发生加载事件时清除故障条件。
2	FI	R/W	0h	故障输入 指定整体故障条件是否取决于检测到的故障引脚。 0h = 整体故障条件不取决于检测到的输入。 1h = 总体故障条件取决于检测到的输入。
1	RESERVED	R/W	0h	
0	FIEN	R/W	0h	故障输入启用 该位启用故障检测输入。 0h = 禁用故障输入 1h = 启用故障输入

23.3.62 FIFCTL（偏移 = 18D4h）[复位 = 00000000h]

图 23-102 展示了 FIFCTL，表 23-82 中对此进行了介绍。

返回到汇总表。

FIFCTL 寄存器控制故障输入的滤波。

图 23-102. FIFCTL

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED			FILTEN	CPV	RESERVED	FP	
R/W-			R/W-0h	R/W-0h	R/W-	R/W-0h	

表 23-82. FIFCTL 字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R/W	0h	
4	FILTEN	R/W	0h	滤波器启用 该位控制输入是由输入滤波器滤波还是旁路以直接进入可选的预分频滤波器，然后进入边缘检测。 0h = 旁路 1h = 滤波。
3	CPV	R/W	0h	连续周期/表决选择 该位控制输入滤波器是使用更严格的连续周期计数还是多数表决。 0h = 连续周期。 在将输入传递到滤波器输出之前，输入必须在 FP 定义的周期内处于特定的逻辑电平。 1h = 表决。 滤波器在滤波器周期内忽略一个相反逻辑的时钟。 即在输入的 FP 样本中，最多 1 个样本可能具有相反的逻辑值（干扰）而不影响输出。
2	RESERVED	R/W	0h	
1-0	FP	R/W	0h	滤波器周期 该字段指定输入滤波器的采样周期。即输入在滤波期间在 FP 个计时器时钟上被采样。 0h = 滤波器周期 3 1h = 滤波器周期 5 2h = 滤波器周期 8

This page intentionally left blank.

章节 24
RTC

实时时钟 (RTC) 模块提供了带日历模式的时钟计数器，灵活的可编程报警，偏移校准以及温度补偿配置。

24.1 概述

实时时钟 (RTC) 模块为应用提供计时，具有采用可选二进制或二进制编码小数 (BCD) 格式的秒、分钟、小时、星期、日期和年份计数器，可提供各种可编程的中断报警。

RTC 的主要特性包括：

- 实时时钟和日历模式提供秒、分钟、小时、星期、日期和年份信息
- 可选二进制或二进制编码小数 (BCD) 格式
- 闰年修正（1901 年至 2099 年有效）
- 两个基于分钟、小时、星期和日期的可定制日历报警中断
- 在每分钟、每小时、午夜或中午唤醒的间隔报警中断
- 以 4096Hz、2048Hz、1024Hz、512Hz、256Hz 或 128Hz 唤醒的周期性中断
- 以 64Hz、32Hz、16Hz、8Hz、4Hz、2Hz、1Hz 和 0.5Hz 唤醒的周期性中断
- 通过 STOPCLKSTBY 将中断功能降至 STANDBY 模式
- 晶体偏移误差和晶体温度漂移校准（总计高达 ±240ppm）
- RTC 时钟输出到引脚以进行校准

24.2 基本操作

RTC 利用两个预分频器块（RT0PS、RT1PS）分别从 32.768kHz RTCCLK 源生成 128Hz 和 1Hz 时钟。每个预分频器块都支持在 4096Hz 到 0.5Hz 的各个速率下生成周期性中断报警。RT1PS 的 1Hz 时钟输出会对计数器块进行计时，该计数器块可跟踪秒、分钟、小时和星期。计数器块还在每分钟、每小时以及午夜或中午产生一次间隔报警中断。该计数器的午夜输出为日历块，提供日期、月份和年份计数，附带闰年校正。提供的两个独立的日历报警中断用于在特定时间唤醒应用程序。

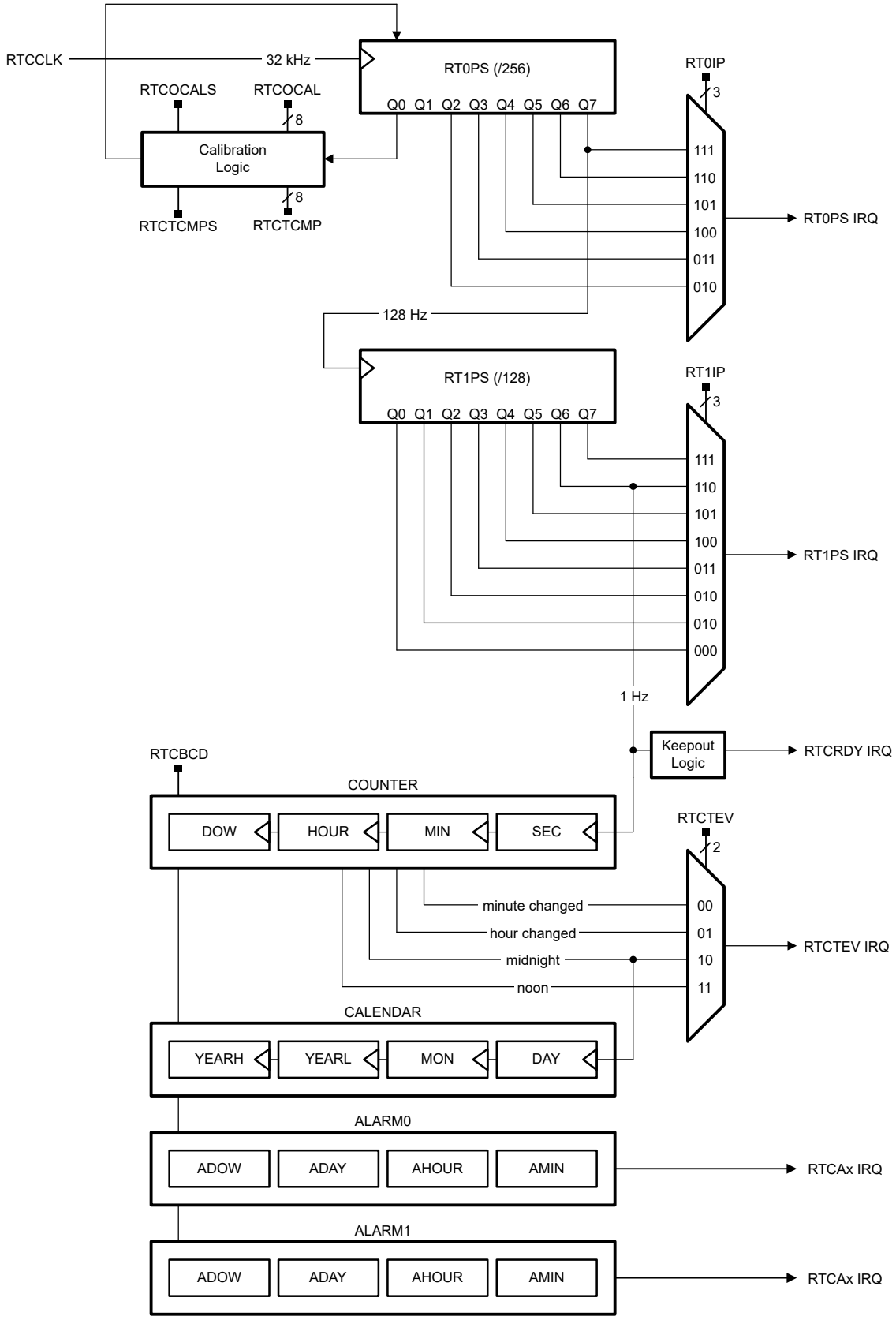


图 24-1. RTC 方框图

24.3 配置

RTC 通过 RTC 外设寄存器进行配置。在配置使用之前，必须通过设置 PWREN 寄存器中的 ENABLE 位（请参阅外设电源使能）以及匹配的 PWREN KEY 值来启用 RTC。

应用软件可以通过设置 RTC 的 RSTCTL 寄存器中的 RESETASSERT 位以及匹配的 RSTCTL KEY 值来随时复位 RTC 状态。在 RTC 复位后，RTC 的 STAT 寄存器中的 RESETSTKY 位将被设置。可以通过在 RTC 的 RSTCTL 寄存器中设置 RESETSTKYCLR 位来清除此标志。因此，软件可以确定自上次将粘性复位位 (RESETSTKY) 清零以来，是否已复位 RTC。许多 RTC 寄存器没有初始条件。在使用之前，这些寄存器必须通过应用软件进行配置。

要启动 RTC 计数器，必须通过应用软件来设置 CLKCTL 寄存器中的 MODCLKEN 位。在 RTC 复位后，MODCLKEN 位最初被清零。

启用和配置后，RTC 在除 SHUTDOWN 之外的所有电源模式下运行。RTC 不会被 CPURST、SYSRST 或 NRST 引脚触发的 BOOTRST 复位（器件复位级别详情，请参阅复位控制部分）。因此，即使通过软件调用引导加载程序 (BSL) 或在外部 NRST 引脚上保持低于 1s，RTC 也会继续运行。

24.3.1 计时

RTC 直接由 RTCCLK 提供时钟，而 RTCCLK 既可以使用内部 LFOSC 作为时钟源，也可以使用外部低频晶体振荡器 (LFXT) 作为时钟源。LFOSC 和 LFXT 都需要启动时间。在启用 RTC 模块的电源之前，确保已配置应用中用作 LFCLK 时钟源的振荡器，并且 LFCLK 工作正常。

器件会自动配置两个预分频器（RT0PS 和 RT1PS）来为 RTC 提供一个 1 秒的时钟间隔。为了确保 RTC 正常运行，LFCLK 必须以 32kHz 的时钟频率运行。RT0PS 分频器直接以 LFCLK 为源，频率为 32kHz。RT0PS 分频器块的输出为 32kHz/256 或 128Hz，并用作 RT1PS 分频器块的输入。RT1PS 输出为 128Hz/128 或 1Hz，用于为实时时钟计数器寄存器提供所需的 1 秒时间间隔。

将 RTC CLKCTL 寄存器中的 MODCLKEN 位清零会暂停向 RTC 中的实时计数器和预分频计数器（RT0PS 和 RT1PS）传播 LFCLK。它对 LFCLK 本身没有任何影响。

24.3.2 读取和写入 RTC 外设寄存器

外设总线时钟 (ULPCLK) 与 RTC 时钟源 (RTCCLK) 异步。因此，在读取和写入某些 RTC 外设寄存器时必须特别小心。

计数器/日历寄存器

RTC 计数器/日历寄存器每秒更新一次。为了避免在更新时读取任一计数器/日历寄存器（因为这样会导致读取无效时间），提供了一个避让窗口。避让窗口大约在计数器更新前 128/32768 秒。在避让窗口期内，STA 寄存器中的只读 RTCRDY 位被复位，在避让窗口期之外则被置位。在 STA 寄存器中的 RTCRDY 位被复位时，对计数器/日历寄存器的任何读取都可能无效，并且时间读取应被忽略。

RTCRDY 中断机制也可被用于安全读取 RTC 计数器/日历寄存器。如果启用了 RTCRDY 中断，基于 RTCRDY 位的上升沿产生中断，从而设置 RTCRDY 中断标志。这时，应用就会有将近 1 秒整的时间来安全读取任何或所有 RTC 寄存器。为了确保安全读取，可以在中断服务例程中再次读取 STA 寄存器中的 RTCRDY 位，并且可以禁用其他中断。该同步过程可阻止在计数器/日历转换期间读取时间值。当中断得到响应后，RTCRDY 中断标志被自动复位，也可以由软件将其复位。

RTC 计数器/日历寄存器可随时写入。写入日历寄存器需要 2 到 3 个 RTCCLK 周期才能生效。如果对计数器/日历寄存器执行背对背写操作，那么在 2-3 个 RTCCLK 周期内，该寄存器可能会被设置为一个未定义的值。因此，需要避免对日历寄存器进行背对背写入。请注意，由于同步，如果在写入日历寄存器之后立即执行读取操作，则回读值始终是实际计数器值。由于同步新写入的值需要 2-3 个 RTCCLK 周期，因此该值可能与写入的值不同。

下列寄存器受上述限制的影响：SEC、MIN、HOUR、DAY、MON、YEAR。

控制寄存器

可随时读取 RTC 控制寄存器 (CTL)。只有当 TEV (间隔计时器) 中断被禁用并设置了 RTCRDY 中断后，才应写入 CTL 寄存器。

以下寄存器受上述限制的影响：CTL。

警报寄存器

可随时读取 RTC 报警配置寄存器。当相应报警中断被禁用并设置了 RTCRDY 中断后，必须完成对报警寄存器的写入。

下列寄存器受上述限制的影响：A1MIN、A1HOUR、A1DAY、A2MIN、A2HOUR、A2DAY。

偏移校正和温度补偿寄存器

只有当设置了 STA 寄存器中的 RTCTCRDY 位时，才能对偏移校正寄存器 (CAL) 和温度补偿寄存器 (TCMP) 进行写入。RTCTCRDY 是一个只读位，在硬件准备引入新的校正或补偿值时设置该位。RTCTCRDY 被清除时应用的写入无效。STA 寄存器中提供了 RTCTCOK 状态位。如果写入 CAL 或 TCMP 成功，则将设置 RTCTCOK，否则它将被复位。在尝试下一次写操作之前，RTCTCOK 将保持其状态。如果写入不成功，当设置 RTCTCRDY 时，软件需要重新尝试写入。

下列寄存器受上述限制的影响：TCMP、CAL。

预分频器控制寄存器

可以随时读取预分频器间隔控制寄存器。只有当 PS0 和 PS1 中断被禁用时，才应该对预分频器控制寄存器执行写操作。

下列寄存器受上述限制的影响：PSCTL。

24.3.3 二进制与 BCD

RTC 以可选择的二进制或二进制编码小数 (BCD) 格式提供秒、分钟、小时、星期几、日期、月份和年份。当设置 RTC 的 CTL 寄存器中的 RTCBCD 位时，选择 BCD 格式。当选择 BCD 格式时，日历寄存器中只有 BCD 位有效。必须在时间被设置前选择格式。

24.3.4 闰年处理

日历包括闰年算法，所有能被四整除的年份均为闰年。该算法将从 1901 年计算到 2099 年。

24.3.5 日历报警配置

RTC 提供了一个灵活的报警系统，其中有两个用户可编程的报警 (A0 和 A1) 配置，它们可以根据报警寄存器中的分钟、小时、星期和日期设置进行编程。

每个报警寄存器包含一个报警使能 (AE) 位，该位可被用于启用各自的报警寄存器。通过设置多种报警寄存器的 AE 位，可以产生多种报警事件。

- **示例 1：**用户希望将报警时间设定为每小时的一刻钟，如 00:15:00、01:15:00、02:15:00 等等。通过将 AxMIN 设置为 15，可以实现这一目标。通过设置 AxMIN 的 AE 位并将其他所有的报警寄存器的 AE 位清零，可以启用该报警。报警启用后，当时间计数从 00:14:59 转变为 00:15:00、01:14:59 转变为 01:15:00、02:14:59 转变为 02:15:00 等等时，设置 RTCAx 中断标志。
- **示例 2：**用户希望将报警时间设定为每天的 04:00:00。通过将 AxHOUR 设置为 4，可以实现这一目标。通过设置 AxHOUR 的 AE 位并将其他所有的报警寄存器的 AE 位清零，可以启用该报警。启用报警后，当时间计数从 03:59:59 转变为 04:00:00 时，设置 RTCAx 中断标志。
- **示例 3：**用户希望将报警时间设定为 06:30:00。AxHOUR 将被设置为 6，AxMIN 将被设置为 30。通过设置 AxHOUR 和 AxMIN 的 AE 位，可以启用该报警。启用报警后，当时间计数从 06:29:59 转变为 06:30:00 时，设置 RTCAx 中断标志。这样，每天 06:30:00 时就会发生报警事件。

- **示例 4：**用户希望将报警时间设定为每周二的 06:30:00。AxDAY 寄存器中的 AxDOW 将被设置为 2，AxHOUR 将被设置为 6，AxMIN 将被设置为 30。通过设置 AxDOW、AxHOUR 和 AxMIN 的 AE 位，可以启用该报警。启用报警后，当时间计数从 06:29:59 转变为 06:30:00 且 DOW 从 1 转变为 2 时，设置 RTCAx 中断标志。
- **示例 5：**用户希望将报警时间设定为每月 5 日的 06:30:00。AxDAY 将被设置为 5，AxHOUR 将被设置为 6，AxMIN 将被设置为 30。通过设置 AxDAY、AxHOUR 和 AxMIN 的 AE 位，可以启用该报警。启用报警后，当时间计数从 06:29:59 转变为 06:30:00 且 RTCDAY 等于 5 时，设置 RTCAx 中断标志。

备注

无效报警设置

硬件不会检查无效报警设置。用户需要进行有效的报警设置。

备注

无效时间和日期

写入无效的日期和/或时间信息，或超出日历和报警寄存器中规定的合法范围的数据值，可能会出现不可预测的行为。

备注

设置报警

在设置初始报警之前，应将所有的报警寄存器清零，包括 AE 位。

为了防止潜在的错误报警发生，在向 RTC 时间寄存器写入初始值或新的时间值之前，应通过清除 RTCAx 中断使能、RTCAx 中断标志和 AE 位来禁用报警功能。

24.3.6 间隔报警配置

除了两个日历报警之外，还提供间隔报警，并且可以配置为在发生以下事件时生成间隔报警事件：

- 分钟变化
- 小时变化
- 每天午夜
- 每天中午

通过在 RTC 的 CTL 寄存器中对 RTCTEVTX 字段进行编程来选择间隔。间隔报警产生 RTCTEV 中断。

24.3.7 定期报警配置

针对应用提供了两种定期报警，可将其配置为生成两个周期性时基。

RT0PS 定期报警可配置为以下列任一速率生成周期性中断：4096Hz、2048Hz、1024Hz、512Hz、256Hz 或 128Hz。RT0PS 周期由 PSCTL 寄存器中的 RT0IP 位设置。RT0PS 事件为 RT0PS 中断提供源。

RT1PS 定期报警可配置为以下列任一速率生成周期性中断：64Hz、32Hz、16Hz、8Hz、4Hz、2Hz、1Hz 或 0.5Hz。RT1PS 周期由 PSCTL 寄存器中的 RT1IP 位设置。RT1PS 事件为 RT1PS 中断提供源。

备注

在相应预分频器运行期间改变定期报警配置的设置（RT0IP、RT1IP）可能会引发设置相应的中断标志。为了避免这种情况，如果在预分频器运行时改变 RT0IP 或 RT1IP，RT0PS 和 RT1PS 中断应该被屏蔽。

24.3.8 Calibration

RTC 模块提供校准机制，用来减少晶体偏移误差和晶体温度漂移导致的计时误差。晶体偏移误差可通过 RTC_OUT 输出测量晶体频率并调整 RTC 中的校准寄存器来校正。晶体温度漂移可通过以下方法进行校正：使用软件测量系统温度，根据测量的温度计算校正因数（以 ppm 为单位），并将校正因数应用于 RTC。

可单独应用高达 ±240ppm 的偏移校正或温度补偿校正，但同时使用偏移校正和温度补偿时，允许的总校正为 ±240ppm。这意味着偏移误差和温度补偿带符号相加不应超过最大值 ±240ppm，否则，硬件将忽略超过 ±240ppm 的值。

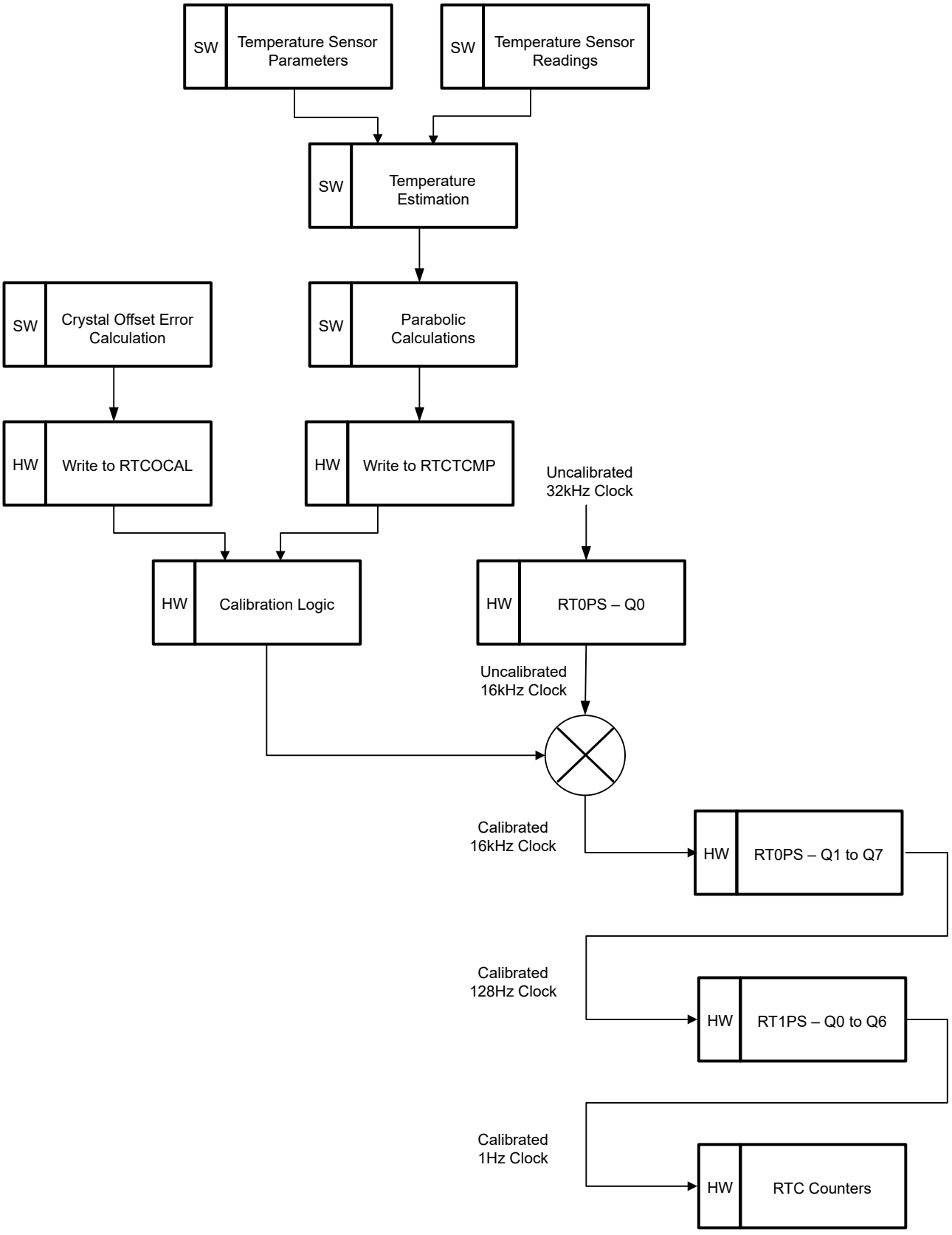


图 24-2. RTC 校准

24.3.8.1 晶体偏移误差

RTC 模块可以针对晶体制造公差或偏移误差进行校准，以获得更高的计时精度。高达 ±240ppm 的晶体频率误差可以在 60 秒内顺利校准。

CAL 寄存器用于调整频率。校准值写入 CAL 寄存器中的 RTCOCALX 字段，该字段中的每个 LSB 代表大约 ±1ppm 的校正（符号基于 CAL 寄存器中的 RTCOCALS 位）。设置 RTCOCALS 时，选择向上校准，RTCOCALX 中的每个 LSB 代表 +1ppm 调整。RTOCALS 被清零时，选择向下校准，RTCOCALX 中的每个 LSB 代表 -1ppm 调整。RTCOCALX 字段为 8 位。软件可以向该寄存器写入高达 256ppm 的值，但可以纠正的最大频率误差仅为 240ppm。软件负责将合法值写入 RTCOCALX。当 RTC 未启用（MODCLKEN 清零）或 RTCOCALX 为零时，偏移误差校准处于非活动状态。写入 RTOCAL 寄存器会将温度补偿重置为零。

备注

CAL 寄存器只能通过半字（16 位）或字（32 位）宽的操作进行访问，以确保符号位与校准值一起设置。

为了校准 RTC 频率，在引脚上提供 RTC_OUT 输出信号。CAL 寄存器中的 RTCCALFX 字段可用于选择引脚输出信号的频率。当 RTCCALFX = 0x0 时，RTC_OUT 引脚上没有信号输出。RTCCALFX 的其他设置选择三个频率选项之一进行输出：512Hz、256Hz 或 1Hz。可以测量 RTC_OUT 输出，并将测量结果应用于 RTCOCALS 和 RTCOCALX 位，以有效减少 RTCCLK 的初始偏移。

24.3.8.1.1 偏移量误差校正机制

在 RTC 中，偏移量误差校准在 60 秒内进行。为了实现大约 ±1ppm 的校正，调整 16kHz 时钟（RT0PS 的 Q0 输出）以增加或减少 1 个时钟脉冲。该校准每 15 秒发生一次，直至可编程的 ppm 误差被补偿。

RTC_OUT 引脚上所有三种可能的输出频率 512Hz、256Hz 和 1Hz 都受校准设置的影响。RT0PS – Q2 至 Q7 触发的 RT0PS 中断基于校准时钟。RT1PS 中断 (RT1PS) 和 RTC 时钟时间事件中断 (RTCTEV) 也基于校准时钟。

以下可作为设置 RTCOCALS 和 RTCOCALX 的指南，给定 $f_{RTCCLK} = f_{RTCCLK, meas}$ x 分频器因子（由 RTCCALFX 设置）：

慢速晶体 (f_{RTCCLK} < 32768Hz)

- 设置 RTCOCALS = 1
- 将 RTCOCALX 设置为舍入 (60 x 16384 x (1-f_{RTCCLK}/32768))

正如增加校准的示例，当测量频率为 511.9658Hz 时，相对于基准频率 512Hz，频率误差就约低了 67ppm。为了补偿这 67ppm 的频率误差，RTCOCALS 必须被置位，并且 RTCOCALx 应该被设置为大约 (60 x 16384 x (1-511.9658 x 64/32768)) = 66。

快速晶体 (f_{ACLK} ≥ 32768Hz)

- 设置 RTCOCALS = 0
- 将 RTCOCALX 设置为舍入 (60 x 16384 x (1-f_{RTCCLK}/32768))

正如降低校准的示例，当测量频率 512.0241Hz 时，相对于基准频率 512Hz，频率误差就约高了 47ppm。为了补偿这 47ppm 的频率误差，RTCOCALS 必须被置位，并且 RTCOCALx 应该被设置为大约 (60 x 16384 x (1-512.0241 x 64/32768)) = 46。

24.3.8.2 晶体温度误差

晶体的输出频率由于温度漂移而可能发生显著变化。可以使用软硬件混合的方法来实现 RTC 的温度补偿。RTC 支持高达 ±240ppm 的温度补偿。

应用软件可以利用芯片上的温度传感器，在所需的时间间隔内（例如每几秒或几分钟）测量器件的温度，从而近似计算电路的环境温度。然后，软件可用于执行抛物线计算来确定相应的频率误差（以 ppm 为单位）。可以将该频率误差写入 TCMP 寄存器来进行温度补偿。

RTCTCMPX 字段包含 8 位，可提供高达 ±240ppm 的频率校正。根据 TCMP 寄存器中的 RTCTCMPS 位，LSB 表示 ±1ppm。当设置 RTCTCMPS 时，RTCTCMPX 中的每个阶跃代表一个 +1ppm 调整（向上校准）。当 RTCTCMPS 位被清零时，RTCTCMPX 中的每个阶跃代表一个 -1ppm 调整（向下校准）。

24.3.8.2.1 温度漂移校正机制

将温度补偿值写入 RTCTCMPX 后，它将与偏移误差校准值相加，得到的值将从下一个校准周期开始被计入。正在进行的校准周期将不受写入 TCMP 寄存器的值影响。

在任何时候读取 TCMPX 都会返回累积校正值（偏移 + 温度），这是 RTCOCALX 和 RTCTCMPX 的有符号加法，以及加法结果的更新符号位 (RTCTCMPS)。请注意，写入 RTCOCAL 寄存器会将温度补偿值复位为零。

例如，如果 RTCOCALX 为 +150ppm，并向 RTCTCMP 写入 +200ppm，则计入下一个校准周期的有效值将为 +240ppm（饱和）。如果 RTCOCALX 为 +150ppm，并向 RTCTCMP 写入 +50ppm，则计入下一个校准周期的有效值为 +200ppm（补偿余量保持在 40ppm）。

为了实现有效的温度补偿，软件负责：

1. 根据工作条件、晶体性能和目标精度的要求，经常测量温度
2. 计算温度引起的频率误差
3. 将温度补偿值写入 RTCTCMPX，同时不超过 ±240ppm 的最大组合校正限值

写入 TCMP 寄存器进行温度补偿需要 60 秒（1 分钟）才能在下一个校准周期中生效。因此，如果必须以高于每分钟一次的频率（例如，每 5 秒一次）测量温度，则需要对一分钟内的误差求平均值，并每分钟更新一次 TCMP 寄存器。

备注

TCMP 寄存器只能通过半字（16 位）或字（32 位）宽的操作进行访问，以确保符号位与校准值一起设置。

24.3.9 RTC 事件

RTC 模块包含两个事件发布者而没有事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理针对 CPU 子系统的 RTC 中断请求 (IRQ)。第二个事件发布者 (GEN_EVENT) 可用于通过通用事件路由通道向订阅者发布 RTC 事件。

表 24-1 中总结了 RTC 事件。

表 24-1. RTC 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	RTC	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 RTC 到 CPU 的中断路由
通用事件	发布者	RTC	通用事件通道	通用路由 (FPUB_0)	GEN_EVENT 寄存器，FPUB_0 寄存器	从 RTC 触发通用事件通道

24.3.9.1 CPU 中断事件发布者 (CPU_INT)

RTC 模块提供 6 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 24-2 中列出了来自 RTC 的 CPU 中断事件。

表 24-2. RTC CPU 中断事件条件 (CPU_INT)

索引 (IIDX)	名称	说明
0	RTCRDY	表示 RTC 计数器/日历寄存器可被安全读取大约一秒钟
1	RTCTEV	间隔中断，可配置为每分钟一次、每小时一次、午夜或中午
2	RTCA1	日历报警 1 中断

表 24-2. RTC CPU 中断事件条件 (CPU_INT) (continued)

索引 (IIDX)	名称	说明
3	RTCA2	日历报警 2 中断
4	RTC0PS	预分频器 0 定期报警中断
5	RTC1PS	预分频器 1 定期报警中断

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

24.3.9.2 通用事件发布者 (GEN_EVENT)

RTC 模块提供六个中断源，其中一个中断源可配置为将事件发布到通用事件路由通道，如表 24-3 所示。

表 24-3. RTC 通用事件发布者条件 (GEN_EVENT)

索引	名称	说明
0	RTCRDY	表示 RTC 计数器/日历寄存器可被安全读取大约一秒钟
1	RTCTEV	间隔中断，可配置为每分钟一次、每小时一次、午夜或中午
2	RTCA1	日历报警 1 中断
3	RTCA2	日历报警 2 中断
4	RTC0PS	预分频器 0 定期报警中断
5	RTC1PS	预分频器 1 定期报警中断

通用事件发布者配置通过 GEN_EVENT 事件管理寄存器进行管理。有关为通用事件发布者配置事件寄存器的指导，请参阅节 7.2.5。

必须通过将目标通用通道 ID 写入 RTC 中的 **FPUB_0** 寄存器来选择将通用事件发布到的通用事件通道。有关配置通用事件路由的指导，请参阅节 7.1.3.3。

如果应用中未使用该发布者，则 FPUB_0 寄存器可以保持断开状态（设置为零），并且不应通过 RTC GEN_EVENT 寄存器集中的 MIS 寄存器解除屏蔽任何事件。

24.4 RTC 寄存器

表 24-4 列出了 RTC 寄存器的存储器映射寄存器。表 24-4 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 24-4. RTC 寄存器

偏移	缩写	寄存器名称	部分
444h	FPUB_0	发布者端口 0	FPUB_0 寄存器 (偏移 = 444h) [复位 = 00000000h]
800h	PWREN	电源使能	PWREN 寄存器 (偏移 = 800h) [复位 = 00000000h]
804h	RSTCTL	复位控制	RSTCTL 寄存器 (偏移 = 804h) [复位 = 00000000h]
808h	CLKCFG	外设时钟配置寄存器	CLKCFG 寄存器 (偏移 = 808h) [复位 = 00000000h]
814h	STAT	状态寄存器	STAT 寄存器 (偏移 = 814h) [复位 = 00000000h]
1004h	CLKSEL	超低功耗外设的时钟选择	CLKSEL 寄存器 (偏移 = 1004h) [复位 = 00000002h]
1020h	IIDX	中断索引寄存器	IIDX 寄存器 (偏移 = 1020h) [复位 = 00000000h]
1028h	IMASK	中断屏蔽	IMASK 寄存器 (偏移 = 1028h) [复位= 00000000h]
1030h	RIS	原始中断状态	RIS 寄存器 (偏移 = 1030h) [复位 = 00000000h]
1038h	MIS	屏蔽中断状态	MIS 寄存器 (偏移 = 1038h) [复位 = 00000000h]
1040h	ISET	中断设置	ISET 寄存器 (偏移 = 1040h) [复位 = 00000000h]
1048h	ICLR	中断清除	ICLR 寄存器 (偏移 = 1048h) [复位 = 00000000h]
1050h	IIDX	中断索引寄存器	IIDX 寄存器 (偏移 = 1050h) [复位 = 00000000h]
1058h	IMASK	中断屏蔽	IMASK 寄存器 (偏移 = 1058h) [复位 = 00000000h]
1060h	RIS	原始中断状态	RIS 寄存器 (偏移 = 1060h) [复位 = 00000000h]
1068h	MIS	屏蔽中断状态	MIS 寄存器 (偏移 = 1068h) [复位 = 00000000h]
1070h	ISET	中断设置	ISET 寄存器 (偏移 = 1070h) [复位 = 00000000h]
1078h	ICLR	中断清除	ICLR 寄存器 (偏移 = 1078h) [复位 = 00000000h]
10E0h	EVT_MODE	事件模式	EVT_MODE 寄存器 (偏移 = 10E0h) [复位 = 00000009h]
10FCh	DESC	RTC 描述符寄存器	DESC 寄存器 (偏移 = 10FCh) [复位 = 09118010h]
1100h	CLKCTL	RTC 时钟控制寄存器	CLKCTL 寄存器 (偏移 = 1100h) [复位 = 00000000h]
1104h	DBGCTL	RTC 模块调试控制寄存器	DBGCTL 寄存器 (偏移 = 1104h) [复位 = 00000000h]
1108h	CTL	RTC 控制寄存器	CTL 寄存器 (偏移 = 1108h) [复位 = 00000000h]

表 24-4. RTC 寄存器 (continued)

偏移	缩写	寄存器名称	部分
110Ch	STA	RTC 状态寄存器	STA 寄存器 (偏移 = 110Ch) [复位 = 00000000h]
1110h	CAL	RTC 时钟偏移校准寄存器	CAL 寄存器 (偏移 = 1110h) [复位 = 00000000h]
1114h	TCMP	RTC 温度补偿寄存器	TCMP 寄存器 (偏移 = 1114h) [复位 = 00000000h]
1118h	SEC	RTC 秒寄存器 - 二进制/BCD 格式的日历模式	SEC 寄存器 (偏移 = 1118h) [复位 = X]
111Ch	最小值	RTC 分钟寄存器 - 二进制/BCD 格式的日历模式	MIN 寄存器 (偏移 = 111Ch) [复位 = X]
1120h	HOUR	RTC 小时寄存器 - 二进制/BCD 格式的日历模式	HOUR 寄存器 (偏移 = 1120h) [复位 = X]
1124h	DAY	RTC 星期/日期寄存器 - 二进制/BCD 格式的日历模式	DAY 寄存器 (偏移 = 1124h) [复位 = X]
1128h	MON	RTC 月寄存器 - 二进制/BCD 格式的日历模式	MON 寄存器 (偏移 = 1128h) [复位 = X]
112Ch	YEAR	RTC 年寄存器 - 二进制/BCD 格式的日历模式	YEAR 寄存器 (偏移 = 112Ch) [复位 = X]
1130h	A1MIN	RTC 分钟报警寄存器 - 二进制/BCD 格式的日历模式	A1MIN 寄存器 (偏移 = 1130h) [复位 = 00000000h]
1134h	A1HOUR	RTC 小时报警寄存器 - 二进制/BCD 格式的日历模式	A1HOUR 寄存器 (偏移 = 1134h) [复位 = 00000000h]
1138h	A1DAY	RTC 报警星期/日期寄存器 - 二进制/BCD 格式的日历模式	A1DAY 寄存器 (偏移 = 1138h) [复位 = 00000000h]
113Ch	A2MIN	RTC 分钟报警寄存器 - 二进制/BCD 格式的日历模式	A2MIN 寄存器 (偏移 = 113Ch) [复位 = 00000000h]
1140h	A2HOUR	RTC 小时报警寄存器 - 二进制/BCD 格式的日历模式	A2HOUR 寄存器 (偏移 = 1140h) [复位= 00000000h]
1144h	A2DAY	RTC 报警星期/日期寄存器 - 二进制/BCD 格式的日历模式	A2DAY 寄存器 (偏移 = 1144h) [复位 = 00000000h]
1148h	PSCTL	RTC 预分频计时器 0/1 控制寄存器	PSCTL 寄存器 (偏移 = 1148h) [复位 = 00000008h]

复杂的位访问类型经过编码可适应小型表单元。表 24-5 显示了适用于此部分中访问类型的代码。

表 24-5. RTC 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
R-0	R -0	读取 返回 0
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

24.4.1 FPUB_0 寄存器（偏移 = 444h）[复位 = 00000000h]

图 24-3 展示了 FPUB_0，表 24-6 中对此进行了介绍。

返回到汇总表。

发布者端口

图 24-3. FPUB_0 寄存器

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

表 24-6. FPUB_0 寄存器字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-0	CHANID	R/W	0h	0 = 已断开连接。 1-15 = 已连接至通道 ID = CHANID。 0h = 值 0 指定事件未连接 Fh = 请参阅您的器件数据表，因为实际允许的最大值可能小于 15。

24.4.2 PWREN 寄存器（偏移 = 800h）[复位 = 00000000h]

图 24-4 展示了 PWREN，表 24-7 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 24-4. PWREN 寄存器

31	30	29	28	27	26	25	24
主要							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-0h							R/WK-0h

表 24-7. PWREN 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	R-0/W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	R/WK	0h	启用电源 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

24.4.3 RSTCTL 寄存器（偏移 = 804h）[复位 = 00000000h]

图 24-5 展示了 RSTCTL，表 24-8 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 24-5. RSTCTL 寄存器

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

表 24-8. RSTCTL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	将 STAT 寄存器中的 RESETSTKY 位清零 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

24.4.4 CLKCFG 寄存器（偏移 = 808h）[复位 = 00000000h]

图 24-6 展示了 CLKCFG，表 24-9 中对此进行了介绍。

返回到汇总表。

外设时钟配置寄存器

图 24-6. CLKCFG 寄存器

31	30	29	28	27	26	25	24
主要							
R-0/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留							BLOCKASYNC
R/W-0h							R/WK-0h
7	6	5	4	3	2	1	0
RESERVED							
R/W-0h							

表 24-9. CLKCFG 寄存器字段说明

位	字段	类型	复位	说明
31-24	KEY	R-0/W	0h	允许状态更改的 KEY -- 0xA9 A9h (W) = 允许更改 GPRCM 字段的密钥值
23-9	保留	R/W	0h	
8	BLOCKASYNC	R/WK	0h	阻止异步时钟请求启动 SYSOSC 或强制总线时钟为 32MHz 必须将 KEY 设置为 A9h 才能写入该位。 0h = 不阻止异步时钟请求 1h = 阻止异步时钟请求
7-0	保留	R/W	0h	

24.4.5 STAT 寄存器（偏移 = 814h）[复位 = 00000000h]

图 24-7 展示了 STAT，表 24-10 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 24-7. STAT 寄存器

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

表 24-10. STAT 寄存器字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

24.4.6 CLKSEL 寄存器（偏移 = 1004h）[复位 = 00000002h]

图 24-8 展示了 CLKSEL，表 24-11 中对此进行了介绍。

返回到汇总表。

ULP 外设的时钟源选择

图 24-8. CLKSEL 寄存器

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

表 24-11. CLKSEL 寄存器字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1	LFCLK_SEL	R	1h	如果启用，选择 LFCLK 作为时钟源 0h = 禁用 LFCLK 作为时钟源 1h = 启用 LFCLK 作为时钟源
0	RESERVED	R	0h	

24.4.7 IIDX 寄存器（偏移 = 1020h）[复位 = 00000000h]

图 24-9 展示了 IIDX，表 24-12 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 24-9. IIDX 寄存器

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

表 24-12. IIDX 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 1h = RTC 就绪中断；中断标志：RTCRDY 2h = 时间事件中断；中断标志： RTCTEV 3h = 报警 1 中断；中断标志：RTCA1 4h = 报警 2 中断；中断标志：RTCA2 5h = 预分频器 0 中断；中断标志：RT0PS 6h = 预分频器 1 中断；中断标志：RT1PS

24.4.8 IMASK 寄存器（偏移 = 1028h）[复位= 00000000h]

图 24-10 展示了 IMASK，表 24-13 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 24-10. IMASK 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 24-13. IMASK 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5	RT1PS	R/W	0h	预分频器 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	RT0PS	R/W	0h	预分频器 0 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	RTCA2	R/W	0h	启用报警 2 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	RTCA1	R/W	0h	启用报警 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	RTCTEV	R/W	0h	启用时间事件中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RTCRDY	R/W	0h	启用 RTC 就绪中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

24.4.9 RIS 寄存器（偏移 = 1030h）[复位 = 00000000h]

图 24-11 展示了 RIS，表 24-14 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 24-11. RIS 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 24-14. RIS 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	RT1PS	R	0h	原始预分频器 1 中断状态 0h = 未发生中断 1h = 已发生中断
4	RT0PS	R	0h	原始预分频器 0 中断状态 0h = 未发生中断 1h = 已发生中断
3	RTCA2	R	0h	原始报警 2 中断状态 0h = 未发生中断 1h = 已发生中断
2	RTCA1	R	0h	原始报警 1 中断状态 0h = 未发生中断 1h = 已发生中断
1	RTCTEV	R	0h	原始时间事件中断状态 0h = 未发生中断 1h = 已发生中断
0	RTCRDY	R	0h	原始 RTC 就绪中断状态 0h = 未发生中断 1h = 已发生中断

24.4.10 MIS 寄存器（偏移 = 1038h）[复位 = 00000000h]

图 24-12 展示了 MIS，表 24-15 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的逻辑与。

图 24-12. MIS 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 24-15. MIS 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	RT1PS	R	0h	已屏蔽预分频器 1 中断状态 0h = 未发生中断 1h = 已发生中断
4	RT0PS	R	0h	已屏蔽预分频器 0 中断状态 0h = 未发生中断 1h = 已发生中断
3	RTCA2	R	0h	已屏蔽报警 2 中断状态 0h = 未发生中断 1h = 已发生中断
2	RTCA1	R	0h	已屏蔽报警 1 中断状态 0h = 未发生中断 1h = 已发生中断
1	RTCTEV	R	0h	已屏蔽时间事件中断状态 0h = 未发生中断 1h = 已发生中断
0	RTCRDY	R	0h	已屏蔽 RTC 就绪中断状态 0h = 未发生中断 1h = 已发生中断

24.4.11 ISET 寄存器（偏移 = 1040h）[复位 = 00000000h]

图 24-13 展示了 ISET，表 24-16 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 24-13. ISET 寄存器

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 24-16. ISET 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	RT1PS	W	0h	设置预分频器 1 中断 0h = 写入 0 无效 1h = 设置中断
4	RT0PS	W	0h	设置预分频器 0 中断 0h = 写入 0 无效 1h = 设置中断
3	RTCA2	W	0h	设置报警 2 中断 0h = 写入 0 无效 1h = 设置中断
2	RTCA1	W	0h	设置报警 1 中断 0h = 写入 0 无效 1h = 设置中断
1	RTCTEV	W	0h	设置时间事件中断 0h = 写入 0 无效 1h = 设置中断
0	RTCRDY	W	0h	设置 RTC 就绪中断 0h = 写入 0 无效 1h = 设置中断

24.4.12 ICLR 寄存器（偏移 = 1048h）[复位 = 00000000h]

图 24-14 展示了 ICLR，表 24-17 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 24-14. ICLR 寄存器

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 24-17. ICLR 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	RT1PS	W	0h	清除预分频器 1 中断 0h = 写入 0 无效 1h = 清除中断
4	RT0PS	W	0h	清除预分频器 0 中断 0h = 写入 0 无效 1h = 清除中断
3	RTCA2	W	0h	清除报警 2 中断 0h = 写入 0 无效 1h = 清除中断
2	RTCA1	W	0h	清除报警 1 中断 0h = 写入 0 无效 1h = 清除中断
1	RTCTEV	W	0h	清除时间事件中断 0h = 写入 0 无效 1h = 清除中断
0	RTCRDY	W	0h	清除 RTC 就绪中断 0h = 写入 0 无效 1h = 清除中断

24.4.13 IIDX 寄存器（偏移 = 1050h）[复位 = 00000000h]

图 24-15 展示了 IIDX，表 24-18 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。值 0x00 表示没有事件挂起。中断 1 是最高优先级，IIDX 是第二高优先级、4、8、…IIDX^31 是最低优先级。也就是说，设置为 1 的最低位位置表示最高优先级的挂起中断。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断（最高优先级）由硬件自动清除，同时 [RIS] 和 [MIS] 中相应的中断标志也会被清除。从 CPU（不是从调试接口）读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则应显示 0x0。

图 24-15. IIDX 寄存器

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

表 24-18. IIDX 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 00h = 无中断挂起 1h = RTC 就绪中断；中断标志：RTCRDY 2h = 时间事件中断；中断标志：RTCTEV 3h = 报警 1 中断；中断标志：RTCA1 4h = 报警 2 中断；中断标志：RTCA2 5h = 预分频器 0 中断；中断标志：RT0PS 6h = 预分频器 1 中断；中断标志：RT1PS

24.4.14 IMASK 寄存器（偏移 = 1058h）[复位 = 00000000h]

图 24-16 展示了 IMASK，表 24-19 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 24-16. IMASK 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R/W-0h		R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 24-19. IMASK 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R/W	0h	
5	RT1PS	R/W	0h	预分频器 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	RT0PS	R/W	0h	预分频器 0 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	RTCA2	R/W	0h	启用报警 2 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	RTCA1	R/W	0h	启用报警 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	RTCTEV	R/W	0h	启用时间事件中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RTCRDY	R/W	0h	启用 RTC 就绪中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽

24.4.15 RIS 寄存器（偏移 = 1060h）[复位 = 00000000h]

图 24-17 展示了 RIS，表 24-20 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 24-17. RIS 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 24-20. RIS 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	RT1PS	R	0h	原始预分频器 1 中断状态 0h = 未发生中断 1h = 已发生中断
4	RT0PS	R	0h	原始预分频器 0 中断状态 0h = 未发生中断 1h = 已发生中断
3	RTCA2	R	0h	原始报警 2 中断状态 0h = 未发生中断 1h = 已发生中断
2	RTCA1	R	0h	原始报警 1 中断状态 0h = 未发生中断 1h = 已发生中断
1	RTCTEV	R	0h	原始时间事件中断状态 0h = 未发生中断 1h = 已发生中断
0	RTCRDY	R	0h	原始 RTC 就绪中断状态 0h = 未发生中断 1h = 已发生中断

24.4.16 MIS 寄存器（偏移 = 1068h）[复位 = 00000000h]

图 24-18 展示了 MIS，表 24-21 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的逻辑与。

图 24-18. MIS 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

表 24-21. MIS 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	R	0h	
5	RT1PS	R	0h	已屏蔽预分频器 1 中断状态 0h = 未发生中断 1h = 已发生中断
4	RT0PS	R	0h	已屏蔽预分频器 0 中断状态 0h = 未发生中断 1h = 已发生中断
3	RTCA2	R	0h	已屏蔽报警 2 中断状态 0h = 未发生中断 1h = 已发生中断
2	RTCA1	R	0h	已屏蔽报警 1 中断状态 0h = 未发生中断 1h = 已发生中断
1	RTCTEV	R	0h	已屏蔽时间事件中断状态 0h = 未发生中断 1h = 已发生中断
0	RTCRDY	R	0h	已屏蔽 RTC 就绪中断状态 0h = 未发生中断 1h = 已发生中断

24.4.17 ISET 寄存器（偏移 = 1070h）[复位 = 00000000h]

图 24-19 展示了 ISET，表 24-22 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 24-19. ISET 寄存器

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 24-22. ISET 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	RT1PS	W	0h	设置预分频器 1 中断 0h = 写入 0 无效 1h = 设置中断
4	RT0PS	W	0h	设置预分频器 0 中断 0h = 写入 0 无效 1h = 设置中断
3	RTCA2	W	0h	设置报警 2 中断 0h = 写入 0 无效 1h = 设置中断
2	RTCA1	W	0h	设置报警 1 中断 0h = 写入 0 无效 1h = 设置中断
1	RTCTEV	W	0h	设置时间事件中断 0h = 写入 0 无效 1h = 设置中断
0	RTCRDY	W	0h	设置 RTC 就绪中断 0h = 写入 0 无效 1h = 设置中断

24.4.18 ICLR 寄存器（偏移 = 1078h）[复位 = 00000000h]

图 24-20 展示了 ICLR，表 24-23 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 24-20. ICLR 寄存器

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
RESERVED		RT1PS	RT0PS	RTCA2	RTCA1	RTCTEV	RTCRDY
W-0h		W-0h	W-0h	W-0h	W-0h	W-0h	W-0h

表 24-23. ICLR 寄存器字段说明

位	字段	类型	复位	说明
31-6	RESERVED	W	0h	
5	RT1PS	W	0h	清除预分频器 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
4	RT0PS	W	0h	清除预分频器 0 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
3	RTCA2	W	0h	清除报警 2 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
2	RTCA1	W	0h	清除报警 1 中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
1	RTCTEV	W	0h	清除时间事件中断 0h = 清除中断屏蔽 1h = 设置中断屏蔽
0	RTCRDY	W	0h	清除 RTC 就绪中断 0h = 清除中断屏蔽 1h = 清除中断

24.4.19 EVT_MODE 寄存器（偏移 = 10E0h）[复位 = 00000009h]

图 24-21 展示了 EVT_MODE，表 24-24 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 24-21. EVT_MODE 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED				EVT1_CFG		EVT0_CFG	
R/W-				R-2h		R-1h	

表 24-24. EVT_MODE 寄存器字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3-2	EVT1_CFG	R	2h	事件线模式 1 选择 0h = 中断或事件线禁用。 1h = 中断或事件线处于软件模式。软件 ISR 清除关联的 RIS 标志。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。
1-0	EVT0_CFG	R	1h	事件线模式 0 选择 0h = 中断或事件线禁用。 1h = 中断或事件线处于软件模式。软件 ISR 清除关联的 RIS 标志。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

24.4.20 DESC 寄存器（偏移 = 10FCh）[复位 = 09118010h]

图 24-22 展示了 DESC，表 24-25 中对此进行了介绍。

返回到汇总表。

RTC 描述符寄存器

图 24-22. DESC 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-911h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-8h				R-0h				R-1h				R-0h			

表 24-25. DESC 寄存器字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	911h	模块标识符。此 ID 对于每个模块都是唯一的。0x0911 = RTC 模块的模块 ID 0000h = 最小值 FFFFh = 最大值
15-12	FEATUREVER	R	8h	此模块的功能集。如果存在差异，则区分实际实例化模块的复杂性。 0h = 最小值 Fh = 最大值
11-8	INSTNUM	R	0h	实例化版本。描述了访问的模块实例。 0h = 如果只有一个实例，则为默认值，例如对于 SSIM
7-4	MAJREV	R	1h	主要版本。此数字包含模块修订版，由模块开发人员递增。n = 主要版本（请参阅器件特定的数据表） 0h = 最小值 Fh = 最大值
3-0	MINREV	R	0h	次要版本。此数字包含模块修订版，由模块开发人员递增。n = 模块的次要修订版（请参阅器件特定的数据表） 0h = 最小值 Fh = 最大值

24.4.21 CLKCTL 寄存器（偏移 = 1100h）[复位 = 00000000h]

图 24-23 展示了 CLKCTL，表 24-26 中对此进行了介绍。

返回到汇总表。

RTC 时钟控制寄存器

图 24-23. CLKCTL 寄存器

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

表 24-26. CLKCTL 寄存器字段说明

位	字段	类型	复位	说明
31	MODCLKEN	R/W	0h	该位启用将 32kHz 时钟提供给 RTC。它不会为 32kHz 晶体振荡器加电，这需要在时钟系统模块中完成。 0h = 32kHz 时钟不提供给 RTC。 1h = 32kHz 时钟提供给 RTC。
30-0	RESERVED	R/W	0h	

24.4.22 DBGCTL 寄存器（偏移 = 1104h）[复位 = 00000000h]

图 24-24 展示了 DBGCTL，表 24-27 中对此进行了介绍。

返回到汇总表。

RTC 模块调试控制寄存器

图 24-24. DBGCTL 寄存器

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

表 24-27. DBGCTL 寄存器字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1	DBGINT	R/W	0h	调试中断使能。 0h = 如果 CPU 处于调试状态，将不再捕获模块的中断。这意味着不更新 RTCRIS、RTCMISC 和 RTCIIDX 寄存器。 1h = 在调试模式下启用中断。向中断控制器发送中断请求信号。如果软件需要这些标志（轮询模式），则需要将 DGBINT 位设置为 1。
0	DBGRUN	R/W	0h	调试运行。 0h = 如果 CPU 处于调试状态，计数器将暂停。 1h = 忽略 CPU 的调试状态，继续正常运行。

24.4.23 CTL 寄存器（偏移 = 1108h）[复位 = 00000000h]

图 24-25 展示了 CTL，表 24-28 中对此进行了介绍。

返回到汇总表。

RTC 控制寄存器

图 24-25. CTL 寄存器

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

表 24-28. CTL 寄存器字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7	RTCBCD	R/W	0h	实时时钟 BCD 选择。选择 BCD 来为实时时钟计数。 0h = 选择了二进制代码 1h = 二进制编码小数 (BCD) 代码
6-2	RESERVED	R/W	0h	
1-0	RTCTEVTX	R/W	0h	实时时钟时间事件。 0h = 更改了分钟。 1h = 更改了小时。 2h = 每天午夜 (00:00)。 3h = 每天中午 (12:00)。

24.4.24 STA 寄存器（偏移 = 110Ch）[复位 = 00000000h]

图 24-26 展示了 STA，表 24-29 中对此进行了介绍。

返回到汇总表。

RTC 状态寄存器

图 24-26. STA 寄存器

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

表 24-29. STA 寄存器字段说明

位	字段	类型	复位	说明
31-3	保留	R	0h	
2	RTCTCOK	R	0h	实时时钟的温度补偿写入 OK。这只是一个指示写入 RTCTCMPx 是否成功的位。 0h = 写入 RTCTCMPx 不成功 1h = 写入 RTCTCMPx 成功
1	RTCTCRDY	R	0h	实时时钟的温度补偿就绪。这只是一个指示何时可以写入 RTCTCMPx 的位。在 RTCTCRDY 被复位时应避免写入 RTCTCMPx。 0h = 实时时钟温度补偿未就绪 1h = 实时时钟温度补偿就绪
0	RTCRDY	R	0h	实时时钟就绪。该位指示何时读取实时时钟的时间值才是安全的。 0h = 转换中的 RTC 时间值 1h = 可安全读取 RTC 时间值

24.4.25 CAL 寄存器（偏移 = 1110h）[复位 = 00000000h]

图 24-27 展示了 CAL，表 24-30 中对此进行了介绍。

返回到汇总表。

RTC 时钟偏移校准寄存器

图 24-27. CAL 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED						RTCCALFX	
R/W-0h						R/W-0h	
15	14	13	12	11	10	9	8
RTCOCALS	RESERVED						
R/W-0h	R/W-0h						
7	6	5	4	3	2	1	0
RTCOCALX							
R/W-0h							

表 24-30. CAL 寄存器字段说明

位	字段	类型	复位	说明
31-18	RESERVED	R/W	0h	
17-16	RTCCALFX	R/W	0h	实时时钟校准频率。选择输出到 RTC_OUT 引脚的频率来校准测量。须为外设模块的功能配置相应的端口。 0h = 无频率输出到 RTC_OUT 引脚 1h = 512Hz 2h = 256Hz 3h = 1Hz
15	RTCOCALS	R/W	0h	实时时钟呃偏移误差校正标志。该位决定了偏移误差校正标志。 0h = 向下校准。频率下调。 1h = 向上校准。频率上调。
14-8	保留	R/W	0h	
7-0	RTCOCALX	R/W	0h	实时时钟呃偏移误差校正。每个 LSB 代表大概 +1ppm (RTCOCALXS = 1) 或 -1ppm (RTCOCALXS = 0) 的频率调整。最大有效校准值是 +/-240ppm。写入的高于 +/-240ppm 超标值将被硬件忽略。 0h = 最小有效校准值。 FFh = 最大有效校准值是 +/-240ppm。写入的高于 +/-240ppm 超标值将被硬件忽略。

24.4.26 TCMP 寄存器（偏移 = 1114h）[复位 = 00000000h]

图 24-28 展示了 TCMP，表 24-31 中对此进行了介绍。

返回到汇总表。

RTC 温度补偿寄存器

图 24-28. TCMP 寄存器

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

表 24-31. TCMP 寄存器字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	RTCTCMPS	R/W	0h	实时时钟的温度补偿标志。该位决定了温度补偿的标志。 0h = 向下校准。频率下调。 1h = 向上校准。频率上调。
14-8	保留	R/W	0h	
7-0	RTCTCMPX	R/W	0h	实时时钟的温度补偿。写入该寄存器的值用于 RTC 的温度补偿。每个 LSB 代表约为 +1ppm (RTCTCMPS = 1) 或 -1ppm (RTCTCMPS = 0) 的频率调整。最大有效校准值是 +/-240ppm。写入的高于 +/-240ppm 超标值会被硬件忽略。在任何时候从 RTCTCMP 寄存器读取都会返回累积值，即 RTCOCALx 和 RTCTCMPX 值的有符号相加，以及相加结果的更新符号位 (RTCTCMPS)。 00h = 最小值 FFh = 最大值

24.4.27 SEC 寄存器（偏移 = 1118h）[复位 = X]

图 24-29 展示了 SEC，表 24-32 中对此进行了介绍。

返回到汇总表。

RTC 秒寄存器 - 二进制/BCD 格式的日历模式

图 24-29. SEC 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	SECHIGHBCD			SECLOWBCD			
R/W-0h	R/W-X			R/W-X			
7	6	5	4	3	2	1	0
RESERVED		SECBIN					
R/W-0h		R/W-X					

表 24-32. SEC 寄存器字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14-12	SECHIGHBCD	R/W	X	秒 BCD – 高位（0 至 5）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 5h = 最大值
11-8	SECLOWBCD	R/W	X	秒 BCD – 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7-6	RESERVED	R/W	0h	
5-0	SECBIN	R/W	X	秒二级制（0 至 59）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 3Bh = 最大值

24.4.28 MIN 寄存器（偏移 = 111Ch）[复位 = X]

图 24-30 展示了 MIN，表 24-33 中对此进行了介绍。

返回到汇总表。

RTC 分钟寄存器 - 二进制/BCD 格式的日历模式

图 24-30. MIN 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留	MINHIGHBCD			MINLOWBCD			
R/W-0h	R/W-X			R/W-X			
7	6	5	4	3	2	1	0
RESERVED		MINBIN					
R/W-0h		R/W-X					

表 24-33. MIN 寄存器字段说明

位	字段	类型	复位	说明
31-15	保留	R/W	0h	
14-12	MINHIGHBCD	R/W	X	分钟 BCD - 高位（0 至 5）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 5h = 最大值
11-8	MINLOWBCD	R/W	X	分钟 BCD – 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7-6	RESERVED	R/W	0h	
5-0	MINBIN	R/W	X	分钟二进制（0 至 59）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 3Bh = 最大值

24.4.29 HOUR 寄存器（偏移 = 1120h）[复位 = X]

图 24-31 展示了 HOUR，表 24-34 中对此进行了介绍。

返回到汇总表。

RTC 小时寄存器 - 二进制/BCD 格式的日历模式

图 24-31. HOUR 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留		HOURHIGHBCD		HOURLOWBCD			
R/W-0h		R/W-X		R/W-X			
7	6	5	4	3	2	1	0
RESERVED			HOURBIN				
R/W-0h			R/W-X				

表 24-34. HOUR 寄存器字段说明

位	字段	类型	复位	说明
31-14	保留	R/W	0h	
13-12	HOURHIGHBCD	R/W	X	小时 BCD – 高位（0 至 2）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值。 2h = 最大值。
11-8	HOURLOWBCD	R/W	X	小时 BCD – 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值。 9h = 最大值。
7-5	RESERVED	R/W	0h	
4-0	HOURBIN	R/W	X	小时二进制（0 至 23）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值。 17h = 最大值。

24.4.30 DAY 寄存器（偏移 = 1124h）[复位 = X]

图 24-32 展示了 DAY，表 24-35 中对此进行了介绍。

返回到汇总表。

RTC 星期/日期寄存器 - 二进制/BCD 格式的日历模式

图 24-32. DAY 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED		DOMHIGHBCD		DOMLOWBCD			
R/W-0h		R/W-X		R/W-X			
15	14	13	12	11	10	9	8
保留			DOMBIN				
R/W-0h			R/W-X				
7	6	5	4	3	2	1	0
RESERVED						DOW	
R/W-0h						R/W-X	

表 24-35. DAY 寄存器字段说明

位	字段	类型	复位	说明
31-22	保留	R/W	0h	
21-20	DOMHIGHBCD	R/W	X	日期 BCD – 高位（0 至 3）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 3h = 最大值
19-16	DOMLOWBCD	R/W	X	日期 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
15-13	RESERVED	R/W	0h	
12-8	DOMBIN	R/W	X	日期二进制（1 至 28、29、30、31）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 1Fh = 最大值
7-3	RESERVED	R/W	0h	
2-0	DOW	R/W	X	星期（0 至 6）。如果 RTCBCD=1 或 RTCBCD=0，这些位有效。 0h = 最小值 6h = 最大值

24.4.31 MON 寄存器（偏移 = 1128h）[复位 = X]

图 24-33 展示了 MON，表 24-36 中对此进行了介绍。

返回到汇总表。

RTC 月寄存器 - 二进制/BCD 格式的日历模式

图 24-33. MON 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
保留			MONHIGHBCD	MONLOWBCD			
R/W-0h			R/W-X	R/W-X			
7	6	5	4	3	2	1	0
RESERVED				MONBIN			
R/W-0h				R/W-X			

表 24-36. MON 寄存器字段说明

位	字段	类型	复位	说明
31 - 13	保留	R/W	0h	
12	MONHIGHBCD	R/W	X	月 BCD - 高位（0 或 1）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 1h = 最大值
11-8	MONLOWBCD	R/W	X	月 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7-4	RESERVED	R/W	0h	
3-0	MONBIN	R/W	X	月二进制（1 至 12）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 Ch = 最大值

24.4.32 YEAR 寄存器（偏移 = 112Ch）[复位 = X]

图 24-34 展示了 YEAR，表 24-37 中对此进行了介绍。

返回到汇总表。

RTC 年寄存器 - 二进制/BCD 格式的日历模式

图 24-34. YEAR 寄存器

31	30	29	28	27	26	25	24
RESERVED	CENTHIGHBCD			CENTLOWBCD			
R/W-0h	R/W-X			R/W-X			
23	22	21	20	19	18	17	16
DECADEBCD				YEARLOWESTBCD			
R/W-X				R/W-X			
15	14	13	12	11	10	9	8
保留				YEARHIGHBIN			
R/W-0h				R/W-X			
7	6	5	4	3	2	1	0
YEARLOWBIN							
R/W-X							

表 24-37. YEAR 寄存器字段说明

位	字段	类型	复位	说明
31	保留	R/W	0h	
30-28	CENTHIGHBCD	R/W	X	世纪 BCD – 高位（0 至 4）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 4h = 最大值
27-24	CENTLOWBCD	R/W	X	世纪 BCD – 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
23-20	DECADEBCD	R/W	X	十进制 BCD（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
19-16	YEARLOWESTBCD	R/W	X	年 BCD – 最低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
15-12	保留	R/W	0h	
11-8	YEARHIGHBIN	R/W	X	年二进制 – 高字节。年的有效值是 0 到 4095。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 Fh = 最大值
7-0	YEARLOWBIN	R/W	X	年二进制 – 低字节。年的有效值是 0 到 4095。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 FFh = 最大值

24.4.33 A1MIN 寄存器（偏移 = 1130h）[复位 = 00000000h]

图 24-35 展示了 A1MIN，表 24-38 中对此进行了介绍。

返回到汇总表。

RTC 分钟报警寄存器 - 二进制/BCD 格式的日历模式

图 24-35. A1MIN 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
AMINAEBCD	AMINHIGHBCD			AMINLOWBCD			
R/W-0h	R/W-0h			R/W-0h			
7	6	5	4	3	2	1	0
AMINAEBIN	RESERVED	AMINBIN					
R/W-0h	R/W-0h	R/W-0h					

表 24-38. A1MIN 寄存器字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	AMINAEBCD	R/W	0h	报警分钟 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14-12	AMINHIGHBCD	R/W	0h	报警分钟 BCD - 高位（0 至 5）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 5h = 最大值
11-8	AMINLOWBCD	R/W	0h	报警分钟 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7	AMINAEBIN	R/W	0h	报警分钟二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
6	RESERVED	R/W	0h	
5-0	AMINBIN	R/W	0h	报警分钟二进制（0 至 59）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 3Bh = 最大值

24.4.34 A1HOUR 寄存器（偏移 = 1134h）[复位 = 00000000h]

图 24-36 展示了 A1HOUR，表 24-39 中对此进行了介绍。

返回到汇总表。

RTC 小时报警寄存器 - 二进制/BCD 格式的日历模式

图 24-36. A1HOUR 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
AHOURAEBCD	RESERVED	AHOURHIGHBCD		AHOURLOWBCD			
R/W-0h	R/W-0h	R/W-0h		R/W-0h			
7	6	5	4	3	2	1	0
AHOURAEBIN	RESERVED			AHOURBIN			
R/W-0h	R/W-0h			R/W-0h			

表 24-39. A1HOUR 寄存器字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	AHOURAEBCD	R/W	0h	报警小时 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14	保留	R/W	0h	
13-12	AHOURHIGHBCD	R/W	0h	报警小时 BCD – 高位（0 至 2）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 2h = 最大值
11-8	AHOURLOWBCD	R/W	0h	报警小时 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7	AHOURAEBIN	R/W	0h	报警小时二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
6-5	RESERVED	R/W	0h	
4-0	AHOURBIN	R/W	0h	报警小时二进制（0 至 23）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 17h = 最大值

24.4.35 A1DAY 寄存器（偏移 = 1138h）[复位 = 00000000h]

图 24-37 展示了 A1DAY，表 24-40 中对此进行了介绍。

返回到汇总表。

RTC 报警星期/日期寄存器 - 二进制/BCD 格式的日历模式

图 24-37. A1DAY 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
ADOMAEBCD	RESERVED	ADOMHIGHBCD		ADOMLOWBCD			
R/W-0h	R/W-0h	R/W-0h		R/W-0h			
15	14	13	12	11	10	9	8
ADOMAEBIN	RESERVED		ADOMBIN				
R/W-0h	R/W-0h		R/W-0h				
7	6	5	4	3	2	1	0
ADOWAE	RESERVED				ADOW		
R/W-0h	R/W-0h				R/W-0h		

表 24-40. A1DAY 寄存器字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23	ADOMAEBCD	R/W	0h	报警日期 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
22	RESERVED	R/W	0h	
21-20	ADOMHIGHBCD	R/W	0h	报警日期 BCD – 高位（0 至 3）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 3h = 最大值
19-16	ADOMLOWBCD	R/W	0h	报警日期 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
15	ADOMAEBIN	R/W	0h	报警日期二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14-13	RESERVED	R/W	0h	
12-8	ADOMBIN	R/W	0h	报警日期二进制（1 至 28、29、30、31）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 1Fh = 最大值
7	ADOWAE	R/W	0h	报警星期使能。如果 RTCBCD=1 或 RTCBCD=0，此位有效。 0h = 禁用报警 1h = 启用报警
6-3	RESERVED	R/W	0h	

表 24-40. A1DAY 寄存器字段说明 (continued)

位	字段	类型	复位	说明
2-0	ADOW	R/W	0h	报警星期（0 至 6）。如果 RTCBCD=1 或 RTCBCD=0，这些位有效。 0h = 最小值 6h = 最大值

24.4.36 A2MIN 寄存器（偏移 = 113Ch）[复位 = 00000000h]

图 24-38 展示了 A2MIN，表 24-41 中对此进行了介绍。

返回到汇总表。

RTC 分钟报警寄存器 - 二进制/BCD 格式的日历模式

图 24-38. A2MIN 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
AMINAEBCD	AMINHIGHBCD			AMINLOWBCD			
R/W-0h	R/W-0h			R/W-0h			
7	6	5	4	3	2	1	0
AMINAEBIN	RESERVED	AMINBIN					
R/W-0h	R/W-0h	R/W-0h					

表 24-41. A2MIN 寄存器字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	AMINAEBCD	R/W	0h	报警分钟 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14-12	AMINHIGHBCD	R/W	0h	报警分钟 BCD - 高位（0 至 5）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 5h = 最大值
11-8	AMINLOWBCD	R/W	0h	报警分钟 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7	AMINAEBIN	R/W	0h	报警分钟二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
6	RESERVED	R/W	0h	
5-0	AMINBIN	R/W	0h	报警分钟二进制（0 至 59）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 3Bh = 最大值

24.4.37 A2HOUR 寄存器 (偏移 = 1140h) [复位= 00000000h]

图 24-39 展示了 A2HOUR，表 24-42 中对此进行了介绍。

返回到汇总表。

RTC 小时报警寄存器 - 二进制/BCD 格式的日历模式

图 24-39. A2HOUR 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
AHOURAEBCD	RESERVED	AHOURHIGHBCD		AHOURLOWBCD			
R/W-0h	R/W-0h	R/W-0h		R/W-0h			
7	6	5	4	3	2	1	0
AHOURAEBIN	RESERVED		AHOURBIN				
R/W-0h	R/W-0h		R/W-0h				

表 24-42. A2HOUR 寄存器字段说明

位	字段	类型	复位	说明
31-16	RESERVED	R/W	0h	
15	AHOURAEBCD	R/W	0h	报警小时 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14	保留	R/W	0h	
13-12	AHOURHIGHBCD	R/W	0h	报警小时 BCD – 高位 (0 至 2) 。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 2h = 最大值
11-8	AHOURLOWBCD	R/W	0h	报警小时 BCD - 低位 (0 至 9) 。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
7	AHOURAEBIN	R/W	0h	报警小时二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
6-5	RESERVED	R/W	0h	
4-0	AHOURBIN	R/W	0h	报警小时二进制 (0 至 23) 。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 17h = 最大值

24.4.38 A2DAY 寄存器（偏移 = 1144h）[复位 = 00000000h]

图 24-40 展示了 A2DAY，表 24-43 中对此进行了介绍。

返回到汇总表。

RTC 报警星期/日期寄存器 - 二进制/BCD 格式的日历模式

图 24-40. A2DAY 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
ADOMAEBCD	RESERVED	ADOMHIGHBCD		ADOMLOWBCD			
R/W-0h	R/W-0h	R/W-0h		R/W-0h			
15	14	13	12	11	10	9	8
ADOMAEBIN	RESERVED		ADOMBIN				
R/W-0h	R/W-0h		R/W-0h				
7	6	5	4	3	2	1	0
ADOWAE	RESERVED				ADOW		
R/W-0h	R/W-0h				R/W-0h		

表 24-43. A2DAY 寄存器字段说明

位	字段	类型	复位	说明
31-24	RESERVED	R/W	0h	
23	ADOMAEBCD	R/W	0h	报警日期 BCD 使能。如果 RTCBCD=0，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
22	RESERVED	R/W	0h	
21-20	ADOMHIGHBCD	R/W	0h	报警日期 BCD – 高位（0 至 3）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 3h = 最大值
19-16	ADOMLOWBCD	R/W	0h	报警日期 BCD - 低位（0 至 9）。如果 RTCBCD=0，将忽略对这些位的写入，而读取会提供值 0。 0h = 最小值 9h = 最大值
15	ADOMAEBIN	R/W	0h	报警日期二进制使能。如果 RTCBCD=1，则该位始终为 0。写入该位将被忽略。 0h = 禁用报警 1h = 启用报警
14-13	RESERVED	R/W	0h	
12-8	ADOMBIN	R/W	0h	报警日期二进制（1 至 28、29、30、31）。如果 RTCBCD=1，将忽略对这些位的写入，而读取会提供值 0。 00h = 最小值 1Fh = 最大值
7	ADOWAE	R/W	0h	报警星期使能。如果 RTCBCD=1 或 RTCBCD=0，此位有效。 0h = 禁用报警 1h = 启用报警
6-3	RESERVED	R/W	0h	

表 24-43. A2DAY 寄存器字段说明 (continued)

位	字段	类型	复位	说明
2-0	ADOW	R/W	0h	报警星期（0 至 6）。如果 RTCBCD=1 或 RTCBCD=0，这些位有效。 0h = 最小值 6h = 最大值

24.4.39 PSCTL 寄存器（偏移 = 1148h）[复位 = 00000008h]

图 24-41 展示了 PSCTL，表 24-44 中对此进行了介绍。

返回到汇总表。

RTC 预分频计时器 0/1 控制寄存器

图 24-41. PSCTL 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED			RT1IP			RESERVED	
R/W-0h			R/W-0h			R/W-0h	
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
RESERVED			RT0IP			RESERVED	
R/W-0h			R/W-2h			R/W-0h	

表 24-44. PSCTL 寄存器字段说明

位	字段	类型	复位	说明
31-21	保留	R/W	0h	
20-18	RT1IP	R/W	0h	预分频计时器 1 中断间隔 0h = 2 分频 - 15.6 毫秒间隔 1h = 4 分频 - 31.2 毫秒间隔 2h = 8 分频 - 62.5 毫秒间隔 3h = 16 分频 - 125 毫秒间隔 4h = 32 分频 - 250 毫秒间隔 5h = 64 - 500 毫秒间隔 6h = 128 分频 - 1 秒间隔 7h = 256 分频 - 2 秒间隔
17-5	RESERVED	R/W	0h	
4-2	RT0IP	R/W	2h	预分频计时器 0 中断间隔 2h = 8 分频 - 244 微秒间隔 3h = 16 分频 - 488 微秒间隔 4h = 32 分频 - 976 微秒间隔 5h = 64 分频 - 1.95 毫秒间隔 6h = 128 分频 - 3.90 毫秒间隔 7h = 256 分频- 7.81 毫秒间隔
1-0	保留	R/W	0h	

窗口看门狗计时器 (WWDT) 可以监控代码执行。如果应用软件在所编程的开放时间窗口内未成功复位窗口看门狗，窗口看门狗会产生复位。

25.1 WWDT 概述

窗口看门狗计时器 (WWDT) 的主要功能是在器件因软件或系统意外延迟而导致无法正确运行时发起复位。WWDT 可以设置一个预定义的时间窗口，在这个时间窗口中，应用软件必须重新开始计时器，表明应用程序正在正常执行。如果应用软件未能在指定窗口内重新开始计时器，WWDT 将向 SYSCTL 发出一个 WWDT 违例信号以产生复位。

如果应用中不需要看门狗功能，则 WWDT 也可以配置为基本的系统间隔计时器，能够为 CPU 生成周期性可屏蔽中断。

WWDT 的主要特性包括：

- 一个具有闭合和开放窗口的 25 位计数器
- 使用可编程时钟分频器从 LFCLK（固定 32kHz 时钟路径）驱动计数器
- 八个可选的看门狗计时器周期
- 在低功耗模式下运行时可选择自动暂停计数器
- 支持标准窗口看门狗模式或间隔计时器（非看门狗）模式

器件可以有 1 个或 2 个 WWDT 实例。一个 WWDT0 违例生成一个 BOOTRST，这个 BOOTRST 会复位外设和 CPU 状态，也会使引导配置例程 (BCR) 运行。一个 WWDT1 违例生成一个 SYSRST，这个 SYSRST 会复位外设和 CPU 状态，但不会触发 BCR 的执行。因此，WWDT1 非常适合从因为软件执行而导致的执行停顿中恢复，而 WWDT0 则非常适合捕获较大的问题，例如损坏的修整值，但代价是复位时间更长。

25.1.1 看门狗模式

在看门狗模式下，WWDT 配置为进行计数，长达指定的 WWDT 周期。WWDT 计数器必须以所配置的 WWDT 周期开放窗口重启计时，否则 WWDT 将向 SYSCTL 发出 WWDT 违例，并产生复位。

如图 25-1 所示，窗口看门狗计时器支持使用可选的封闭窗口来检测“太晚”响应和“太早”响应。WWDT 周期包含封闭窗口期和开放窗口期。封闭窗口期先开始，然后是开放窗口期。WWDT 只能在开放窗口期内重新启动。在封闭窗口期内尝试重新启动 WWDT 会导致违例。在封闭窗口期后，如果在开放窗口期结束之前未重新启动 WWDT，则 WWDT 周期将会过期，还会产生违例。

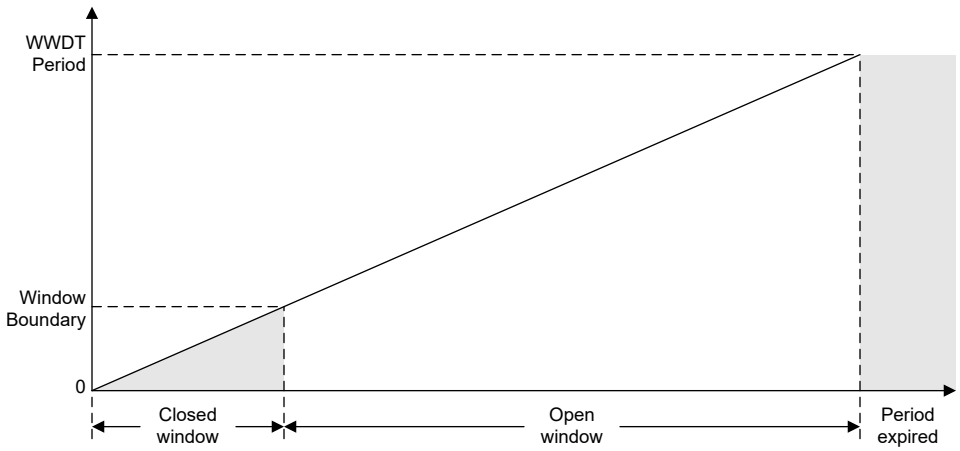


图 25-1. WWDT 功能

如果不需要封闭窗口功能，可以将其禁用（设置为 0%），从而提供传统的看门狗计时器功能，该功能可以在 WWDT 周期到期之前的任何时间复位 WWDT。

25.1.2 间隔定时器模式

当不使用看门狗功能时，WWDT 可用于在间隔计时器模式下对 CPU 生成周期性中断。如果在间隔计时器模式下使用，当 WWDT 周期到期时或向 WWDT 控制寄存器应用了不正确的密码时，会生成 WWDT 中断。

25.2 WWDT 运行

必须先启用 WWDT，然后才能通过 PWREN 寄存器配置为使用（请参阅外设电源使能）。

WWDT 通过 WWDTCTL0 和 WWDTCTL1 寄存器进行配置。寄存器受密码保护。任何寄存器访问（读或写）都必须是 32 位访问。写访问还必须在最高有效字节中包含相应的密码（对于 WWDTCTL0 为 0xC9，对于 WWDTCTL1 为 0xBE）。如果尝试在没有正确密码的情况下写入寄存器，或者尝试使用 32 位访问以外的访问进行写入，则会向 SYSCTL 生成 WWDT 违例。密码字节始终读取为 0x00。

WWDT 在 SYSRST 之后被禁用和清除。WWDTCTL0 寄存器用于设置 WWDT 的静态配置，包括时钟分频器、计时器周期、两个关闭窗口百分比、计时器模式（WWDT 或间隔）以及休眠停止状态。第一次写入（具有匹配密钥）到 WWDTCTL0 寄存器将启用 WWDT。一旦启用了 WWDT，WWDTCTL0 寄存器便会受写保护。启用 WWDT 后，任何写入 WWDTCTL0 寄存器的尝试都会向 SYSCTL 生成 WWDT 违例。WWDTSTAT 寄存器中的 RUN 位指示 WWDT 正在运行。

图 25-2 展示了 WWDT 功能方框图。

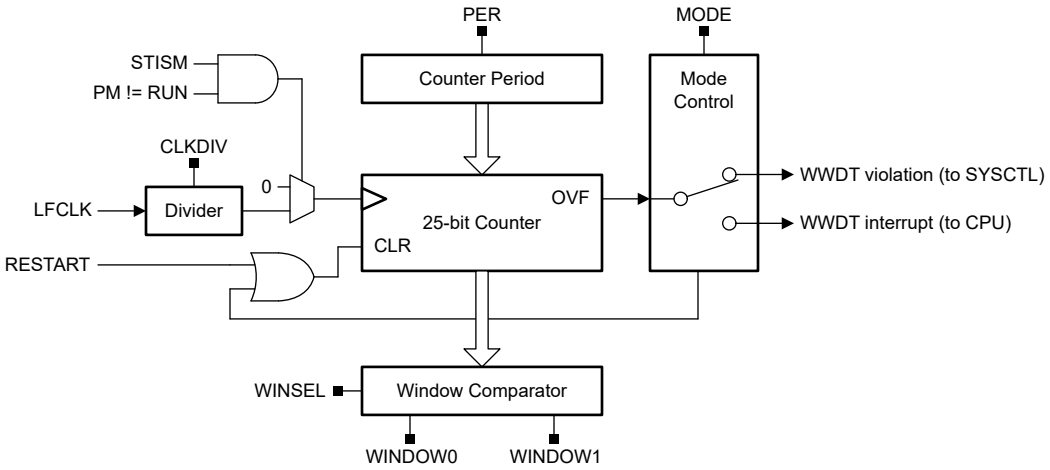


图 25-2. WWDT 方框图

25.2.1 模式选择

WWDT 工作模式（看门狗模式或间隔计时器模式）可通过 WWDTCTL0 寄存器中的 MODE 位来选择。默认模式是看门狗模式（MODE 清零）。设置 MODE 位可将 WWDT 配置为间隔模式。

当 WWDT 处于看门狗模式时，必须通过向 WWDTCNTRST 寄存器写入 RESTART 值 (0x000000A7)，从而在开放窗口期内重新启动 WWDT 计数器。在复位或重新启动后，WWDT 计数器将会从零重新开始。如果未能在开放窗口期内重新启动 WWDT，或者尝试在关闭窗口期内重新启动 WWDT 计数器，将会向 SYSCTL 生成 WWDT 违例。如果向 WWDTCNTRST 寄存器写入除 RESTART 值以外的任何值，也会导致生成 WWDT 违例。

当 WWDT 处于间隔模式时，计时器充当间隔计时器，并按照 WWDT 周期指定的方式向 CPU 生成 WWDT 中断。一旦在间隔模式下启用 WWDT，就会在计时器到期后使 WWDT 间隔计时器中断有效。无需在间隔计时器模式下重新启动 WWDT。

25.2.2 时钟配置

WWDT 从 32kHz 低频时钟 (LFCLK) 运行。时钟分频器支持使用 WWDTCTL0 寄存器中的 CLKDIV 字段，将输入时钟从 /1（不分频）分频为 /8（8 分频）。默认 CLKDIV 设置为 0x03（32kHz 进行 4 分频，即 8kHz）。

通过从 LFCLK 运行，只要不将这些时钟配置为从 LFCLK 运行，WWDT 时基就与主时钟 (MCLK) 和 CPU 时钟 (CPUCLK) 时基无关。虽然时基可能被视为独立的并源自一个单独的振荡器源，但在提供 WWDT 之前，LFCLK 边沿会与 MCLK 同步，以实现从应用软件对存储器映射寄存器的简单访问。图 25-3 中提供了时钟方案的简化视

图。在图 25-3 中，内部 LFOSC 配置为设置 LFCLK 速率，内部 SYSOSC 设置 MCLK/CPUCLK 速率。为了简化视图，图中不包括时钟选择多路复用器和分频器；节 2.3.3 中提供了完整的时钟树。

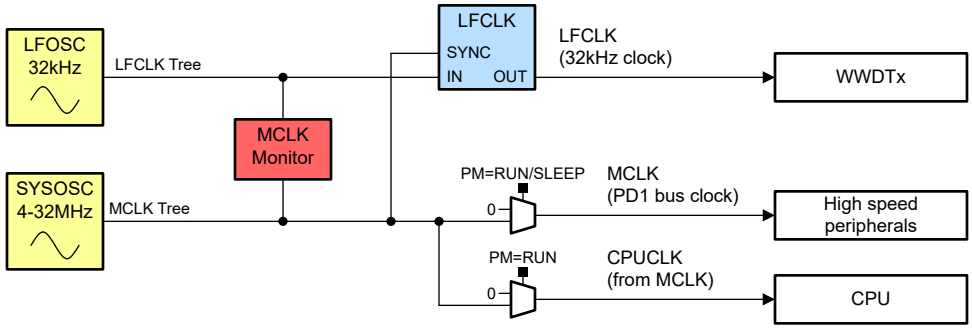


图 25-3. WWDT 简化时钟源树

如果 MCLK 失败并且 LFCLK 与 MCLK 的同步丢失，则可以通过启用连续 MCLK 监测器来检测到该故障。当 MCLK 监测器被启用时，MCLK 的丢失始终是一个灾难性故障，它会在 12 个 LFCLK 周期内生成一个 BOOTRST。因此，MCLK 树的丢失和相应的同步丢失不会阻止生成 BOOTRST。

期间选择

WWDT 有一个 25 位计数器，最初会在 SYSRST 之后停止。WWDT 周期（总时间间隔）由 WWDTCTL0 寄存器中的 PER 字段来设置。总 WWDT 周期的计算方法如下：

$$T_{WWDT} = (CLKDIV + 1) * PER_{COUNT} / 32768Hz \quad (31)$$

对于总计时器计数 PER_{COUNT}，选择 8 个可能的周期计数值之一，表 25-1 给出了相应的编码。

表 25-1. WWDT 周期总计时器计数

PER	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
PER_{COUNT}	2^{25}	2^{21}	2^{18}	2^{15}	2^{12}	2^{10}	2^{8}	2^{6}

周期选择 (PER) 和时钟分频器 (CLKDIV) 的组合可实现宽范围的 WWDT 周期，范围从 1.95ms 到 136.53 分钟。表 25-2 给出了给定 PER 和 CLKDIV 组合的所有可能的 WWDT 周期。

表 25-2. WWDT 周期时序选项

CLKDIV	PER							
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
	min	*min*	*s*	*s*	*s*	*ms*	*ms*	*ms*
0x0 (/1)	17.07	1.07	8.00	1.00	0.13	31.25	7.81	1.95
0x1 (/2)	34.13	2.13	16.00	2.00	0.25	62.50	15.63	3.91
0x2 (/3)	51.20	3.20	24.00	3.00	0.38	93.75	23.44	5.86
0x3 (/4)	68.27	4.27	32.00	4.00	0.50	125.00	32.25	7.81
0x4 (/5)	85.33	5.33	40.00	5.00	0.63	156.25	39.06	9.77
0x5 (/6)	102.40	6.40	48.00	6.00	0.75	187.50	46.88	11.72
0x6 (/7)	119.47	7.47	56.00	7.00	0.88	218.75	54.69	13.67
0x7 (/8)	136.53	8.53	64.00	8.00	1.00	250.00	62.50	15.63

同步延迟

当启动或重新启动 WWDT 计数器时，在 WWDT 计数器从零开始计数之前，可能会发生一个 32kHz 时钟周期 (30.5 µs) 的最大同步延迟。表 25-2 中给出的周期不包括此同步延迟。

关闭窗口选择

通过设置 WWDTCTL0 寄存器中的 WINDOW0 和 WINDOW1 字段，可以配置两个关闭窗口期。WWDTCTL1 寄存器中的 WINSEL 位决定运行窗口（WINDOW0 或 WINDOW1）。任一窗口都可以设置为 8 种可能的窗口设置之一。

表 25-3. WWDT 窗口选项

窗口	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
关闭窗口	0%	12.5%	18.75%	25%	50%	75%	81.25%	87.5%

将 WINDOWx 值设置为 0x0（0% 关闭，100% 打开）等效于禁用 WWDT 的窗口功能。在此配置中，可以在 WWDT 期间的任意时刻重新启动 WWDT。

启用 WWDT 后，可以更改活跃窗口选择。当向 WWDTCNTRST 寄存器写入数据以重新启动 WWDT 时，必须至少在四个 32kHz 时钟周期（约为 122 µs）内更改关闭窗口选择 (WINSEL)。

25.2.3 低功耗模式行为

WWDT 计数器可配置为在器件处于低功耗模式（CPU 被禁用）时继续计数，或者在器件处于低功耗模式时继续运行。

WWDTCTL0 寄存器中的 STISM 位控制着 WWDT 计数器在睡眠模式下是否停止计数。默认情况下，STISM 位清零，表示 WWDT 在低功耗模式下继续计数。要使 WWDT 在低功耗模式下停止计数，请在加载 WWDTCTL0 配置时设置 STISM 位以启动 WWDT。在这种情况下，当退出低功耗模式并且 CPU 恢复运行时，WWDT 计数器会以它在进入低功耗模式之前保持的相同值恢复计数。

25.2.4 调试行为

当 CPU 暂停以供调试子系统调试时，可将 WWDT 配置为停止计数或继续计数。默认情况下，当 CPU 暂停以供调试且器件处于调试状态时，WWDT 停止计数。要在 CPU 停止以供调试时允许 WWDT 继续自由运行，请设置 PDBGCTL 寄存器中的 FREE 位。

25.2.5 WWDT 事件

WWDT 模块包含一个事件发布者，不包含事件订阅者。一个事件发布者 (CPU_INT) 通过静态事件路由来管理针对 CPU 子系统的 WWDT 中断请求 (IRQ)。

表 25-4 列出了 WWDT 事件。

表 25-4. WWDT 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	WWDT	CPU 子系统	静态路由	CPU_INT 寄存器	从 WWDT 到 CPU 的固定中断路由

25.2.5.1 CPU 中断事件发布者 (CPU_INT)

WWDT 模块提供 1 个中断源，这些中断源可配置为产生 CPU 中断事件。表 25-5 中给出了 WWDT 中断条件。

表 25-5. WWDT CPU 中断条件 (CPU_INT)

索引 (IIDX)	名称	说明
0	INTTIM	表示 WWDT 间隔计时器周期已过期

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

25.3 WWDT 寄存器

表 25-6 列出了 WWDT 寄存器的存储器映射寄存器。表 25-6 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 25-6. WWDT 寄存器

偏移	缩写	寄存器名称	部分
800h	PWREN	电源使能	节 25.3.1
804h	RSTCTL	复位控制	节 25.3.2
814h	STAT	状态寄存器	节 25.3.3
1018h	PDBGCTL	外设调试控制	节 25.3.4
1020h	IIDX	中断索引	节 25.3.5
1028h	IMASK	中断屏蔽	节 25.3.6
1030h	RIS	原始中断状态	节 25.3.7
1038h	MIS	屏蔽中断状态	节 25.3.8
1040h	ISET	中断设置	节 25.3.9
1048h	ICLR	中断清除	节 25.3.10
10E0h	EVT_MODE	事件模式	节 25.3.11
10FCh	DESC	模块说明	节 25.3.12
1100h	WWDTCTL0	窗口看门狗计时器控制寄存器 0	节 25.3.13
1104h	WWDTCTL1	窗口看门狗计时器控制寄存器 0	节 25.3.14
1108h	WWDTCNTRST	窗口看门狗计时器计数器复位寄存器	节 25.3.15
110Ch	WWDTSTAT	窗口看门狗计时器状态寄存器	节 25.3.16

复杂的位访问类型经过编码可适应小型表单元。表 25-7 显示了适用于此部分中访问类型的代码。

表 25-7. WWDT 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
写入类型		
K	K	受密钥保护的写入
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

25.3.1 PWREN 寄存器（偏移 = 800h）[复位 = 00000000h]

图 25-4 展示了 PWREN，表 25-8 中对此进行了介绍。

返回到汇总表。

用于控制电源状态的寄存器

图 25-4. PWREN 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
RESERVED							ENABLE
R/W-							K-0h

表 25-8. PWREN 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许电源状态更改的 KEY 26h = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	ENABLE	K	0h	启用电源 注意：对于安全器件，电源一旦启用就无法禁用。 必须将 KEY 设置为 26h 才能写入该位。 0h = 禁用电源 1h = 启用电源

25.3.2 RSTCTL 寄存器（偏移 = 804h）[复位 = 00000000h]

图 25-5 展示了 RSTCTL，表 25-9 中对此进行了介绍。

返回到汇总表。

用于控制复位有效和无效的寄存器

图 25-5. RSTCTL 寄存器

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

表 25-9. RSTCTL 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	解锁密钥 B1h = 允许对该寄存器进行写入访问的 KEY
23-2	RESERVED	W	0h	
1	RESETSTKYCLR	WK	0h	清除 RESETSTKY 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 不产生影响 1h = 将复位粘滞位清零
0	RESETASSERT	WK	0h	外设复位生效 注意：对于安全器件，无法通过软件进行看门狗复位。 必须将 KEY 设置为 B1h 才能写入该位。 0h = 写入 0 无效 1h = 复位生效

25.3.3 STAT 寄存器（偏移 = 814h）[复位 = 00000000h]

图 25-6 展示了 STAT，表 25-10 中对此进行了介绍。

返回到汇总表。

外设启用和复位状态

图 25-6. STAT 寄存器

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

表 25-10. STAT 寄存器字段说明

位	字段	类型	复位	说明
31-17	RESERVED	R	0h	
16	RESETSTKY	R	0h	该位指示自 RSTCTL 寄存器中的 RESETSTKYCLR 将该位清零以来，外设是否复位 0h = 自 RSTCTL 寄存器中的 RESETSTKYCLR 上次将该位清零以来，外设尚未复位 1h = 自从上次将该位清零以来，外设已复位
15-0	RESERVED	R	0h	

25.3.4 PDBGCTL 寄存器（偏移 = 1018h）[复位 = 00000000h]

图 25-7 展示了 PDBGCTL，表 25-11 中对此进行了介绍。

返回到汇总表。

软件开发人员可以使用该寄存器来控制外设相对于“内核停止”输入的行为

图 25-7. PDBGCTL 寄存器

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

表 25-11. PDBGCTL 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	免费	R/W	0h	自由运行控制 0h = 当“内核停止”输入被置为有效时，外设功能冻结；当它被置为无效时，外设功能恢复。 1h = 外设忽略“内核停止”输入的状态

25.3.5 IIDX 寄存器（偏移 = 1020h）[复位 = 00000000h]

图 25-8 展示了 IIDX，表 25-12 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。

图 25-8. IIDX 寄存器

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

表 25-12. IIDX 寄存器字段说明

位	字段	类型	复位	说明
31-5	RESERVED	R	0h	
4-0	STAT	R	0h	模块中断矢量值。该寄存器提供了最高优先级中断索引。读取操作会清除 RIS 和 MISC 中的相应中断标志。 0h = 无中断挂起 1h = 间隔计时器中断；中断标志： INTTIM；中断优先级：最高

25.3.6 IMASK 寄存器（偏移 = 1028h）[复位= 00000000h]

图 25-9 展示了 IMASK，表 25-13 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 25-9. IMASK 寄存器

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

表 25-13. IMASK 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R/W	0h	
0	INTTIM	R/W	0h	间隔计时器中断。 0h = 清除中断屏蔽 1h = 设置中断屏蔽

25.3.7 RIS 寄存器（偏移 = 1030h）[复位 = 00000000h]

图 25-10 展示了 RIS，表 25-14 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 25-10. RIS 寄存器

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

表 25-14. RIS 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	INTTIM	R	0h	间隔计时器中断。 0h = 未发生中断 1h = 已发生中断

25.3.8 MIS 寄存器（偏移 = 1038h）[复位 = 00000000h]

图 25-11 展示了 MIS，表 25-15 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的逻辑与。

图 25-11. MIS 寄存器

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

表 25-15. MIS 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	INTTIM	R	0h	间隔计时器中断。 0h = 未发生中断 1h = 已发生中断

25.3.9 ISET 寄存器（偏移 = 1040h）[复位 = 00000000h]

图 25-12 展示了 ISET，表 25-16 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 25-12. ISET 寄存器

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

表 25-16. ISET 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	
0	INTTIM	W	0h	间隔计时器中断。 0h = 写入 0 不产生影响 1h = 设置中断

25.3.10 ICLR 寄存器（偏移 = 1048h）[复位 = 00000000h]

图 25-13 展示了 ICLR，表 25-17 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 25-13. ICLR 寄存器

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

表 25-17. ICLR 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	W	0h	
0	INTTIM	W	0h	间隔计时器中断。 0h = 写入 0 不产生影响 1h = 清除中断

25.3.11 EVT_MODE 寄存器（偏移 = 10E0h）[复位 = 00000001h]

图 25-14 展示了 EVT_MODE，表 25-18 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 25-14. EVT_MODE 寄存器

31	30	29	28	27	26	25	24
RESERVED							
R/W-							
23	22	21	20	19	18	17	16
RESERVED							
R/W-							
15	14	13	12	11	10	9	8
RESERVED							
R/W-							
7	6	5	4	3	2	1	0
ESERVED						INT0_CFG	
R/W-						R-1h	

表 25-18. EVT_MODE 寄存器字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R/W	0h	
1-0	INT0_CFG	R	1h	none.INT_EVENT[0] 对应事件的事件线模式选择 0h = 中断或事件线被禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

25.3.12 DESC 寄存器（偏移 = 10FCh）[复位 = 1F117010h]

图 25-15 展示了 DESC，表 25-19 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 25-15. DESC 寄存器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-1F11h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-7h				R-0h				R-1h				R-0h			

表 25-19. DESC 寄存器字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	1F11h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。 0h = 最小值 FFFFh = 尽可能高的值
15-12	FEATUREVER	R	7h	模块 *实例* 的功能集 0h = 最小值 Fh = 尽可能高的值
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数 0h = 最小值 Fh = 尽可能高的值
7-4	MAJREV	R	1h	IP 的主要版本 0h = 最小值 Fh = 尽可能高的值
3-0	MINREV	R	0h	IP 的次要版本 0h = 最小值 Fh = 尽可能高的值

25.3.13 WWDTCTL0 寄存器（偏移 = 1100h）[复位 = 00000043h]

图 25-16 展示了 WWDTCTL0，表 25-20 中对此进行了介绍。

返回到汇总表。

窗口看门狗计时器控制 0 寄存器
注意：系统复位后，对该寄存器的写入会启用。第一次成功写入（密钥匹配）将启用看门狗。启用看门狗后，对该寄存器的所有后续写入都会激活向 ESM 发送 WWDT 错误信号。

图 25-16. WWDTCTL0 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED						STISM	MODE
R/W-0h						R/W-0h	R/W-0h
15	14	13	12	11	10	9	8
保留	WINDOW1			RESERVED	WINDOW0		
R/W-0h	R/W-0h			R/W-0h	R/W-0h		
7	6	5	4	3	2	1	0
ESERVED	PER			RESERVED	CLKDIV		
R/W-0h	R/W-4h			R/W-0h	R/W-3h		

表 25-20. WWDTCTL0 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许对该寄存器进行写入访问的 KEY。 使用不正确的密钥写入该寄存器将激活向 ESM 发送 WWDT 错误信号。 读为 0。 C9h (W) = 允许对该寄存器进行写入访问的 KEY。
23-18	RESERVED	R/W	0h	
17	STISM	R/W	0h	在睡眠模式下停止。 该位的功能要求 POLICY.HWCEN = 0。如果 POLICY.HWCEN = 1，则 WWDT 会在睡眠期间复位并需要重新配置。 注意：由于不支持睡眠模式，该位对全局窗口看门狗没有影响。 0h = WWDT 在睡眠模式下继续工作。 1h = WWDT 在睡眠模式下停止，并在唤醒后从停止处恢复。
16	MODE	R/W	0h	窗口看门狗计时器模式 0h = 窗口看门狗计时器模式。发生以下情况时，WWDT 将向 ESM 生成一个错误信号：- 计时器过期（超时）- 在活跃窗口关闭期间复位 WWDT - 关键字违例 1h = 间隔计时器模式。 WWDT 用作间隔计时器。它在超时时生成中断。
15	保留	R/W	0h	

表 25-20. WWDTCTL0 寄存器字段说明 (continued)

位	字段	类型	复位	说明
14-12	WINDOW1	R/W	0h	以计时器间隔百分比表示的关闭窗口期。WWDTCTL1.WINSEL 确定活跃窗口设置（WWDTCTL0.WINDOW0 或 WWDTCTL0.WINDOW1）。 0h = 0%（无关闭窗口） 1h = 总计时器周期的 12.50% 为关闭窗口 2h = 总计时器周期的 18.75% 为关闭窗口 3h = 总计时器周期的 25% 为关闭窗口 4h = 总计时器周期的 50% 为关闭窗口 5h = 总计时器周期的 75% 为关闭窗口 6h = 总计时器周期的 81.25% 为关闭窗口 7h = 总计时器周期的 87.50% 为关闭窗口
11	RESERVED	R/W	0h	
10-8	WINDOW0	R/W	0h	以计时器间隔百分比表示的关闭窗口期。WWDTCTL1.WINSEL 确定活跃窗口设置（WWDTCTL0.WINDOW0 或 WWDTCTL0.WINDOW1）。 0h = 0%（无关闭窗口） 1h = 总计时器周期的 12.50% 为关闭窗口 2h = 总计时器周期的 18.75% 为关闭窗口 3h = 总计时器周期的 25% 为关闭窗口 4h = 总计时器周期的 50% 为关闭窗口 5h = 总计时器周期的 75% 为关闭窗口 6h = 总计时器周期的 81.25% 为关闭窗口 7h = 总计时器周期的 87.50% 为关闭窗口
7	RESERVED	R/W	0h	
6-4	PER	R/W	4h	WWDT 的计时器周期。这些位选择总看门狗计时器计数。 0h = 总计时器计数为 2^{25} 1h = 总计时器计数为 2^{21} 2h = 总计时器计数为 2^{18} 3h = 总计时器计数为 2^{15} 4h = 总计时器计数为 2^{12}（默认值） 5h = 总计时器计数为 2^{10} 6h = 总计时器计数为 2^{8} 7h = 总计时器计数为 2^{6}
3	RESERVED	R/W	0h	
2-0	CLKDIV	R/W	3h	模块时钟分频器，对时钟源进行 CLKDIV+1 分频。 分频器值可以为 1 到 8。 时钟分频器当前为 4 位。位 4 没有任何影响并且应该始终写入 0。 0h = 最小值 7h = 最大值

25.3.14 WWDTCTL1 寄存器（偏移 = 1104h）[复位 = 00000000h]

图 25-17 展示了 WWDTCTL1，表 25-21 中对此进行了介绍。

返回到汇总表。

窗口看门狗计时器控制 1 寄存器

图 25-17. WWDTCTL1 寄存器

31	30	29	28	27	26	25	24
主要							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED							WINSEL
R/W-0h							R/W-0h

表 25-21. WWDTCTL1 寄存器字段说明

位	字段	类型	复位	说明
31-24	主要	W	0h	允许对该寄存器进行写入访问的 KEY。 使用不正确的密钥写入该寄存器将激活向 ESM 发送 WWDT 错误信号。 读为 0。 BEh (W) = 允许对该寄存器进行写入访问的 KEY
23-1	RESERVED	R/W	0h	
0	WINSEL	R/W	0h	关闭窗口选择 0h = 在窗口模式中，WDDTCTL0 的字段 WINDOW0 定义关闭窗口大小。 1h = 在窗口模式中，WDDTCTL0 的 WINDOW1 字段定义关闭窗口大小。

25.3.15 WWDTCNTRST 寄存器（偏移 = 1108h）[复位 = 00000000h]

图 25-18 展示了 WWDTCNTRST，表 25-22 中对此进行了介绍。

返回到汇总表。

窗口看门狗计时器计数器重启寄存器

图 25-18. WWDTCNTRST 寄存器

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

表 25-22. WWDTCNTRST 寄存器字段说明

位	字段	类型	复位	说明
31-0	RESTART	R/W	0h	窗口看门狗计时器计数器重启向该寄存器写入 00A7h 会重启 WWDT 计数器。 写入任何其他值会导致向 ESM 生成一个错误。 读为 0。 0h = 最小值 FFFFFFFFh = 最大值

25.3.16 WWDTSTAT 寄存器（偏移 = 110Ch）[复位 = 00000000h]

图 25-19 展示了 WWDTSTAT，表 25-23 中对此进行了介绍。

返回到汇总表。

窗口看门狗计时器状态寄存器
对该寄存器的写入操作无效。

图 25-19. WWDTSTAT 寄存器

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

表 25-23. WWDTSTAT 寄存器字段说明

位	字段	类型	复位	说明
31-1	RESERVED	R	0h	
0	运行	R	0h	看门狗运行状态标志。 0h = 看门狗计数器已停止。 1h = 看门狗正在运行。

This page intentionally left blank.

章节 26
调试

调试子系统 (DEBUGSS) 在所有 MSPM0 器件中实现。通过串行线调试 (SWD) 接口将外部调试探针连接到器件系统，DEBUGSS 可在开发期间全面调试处理器上运行的应用软件。

26.1 概述

调试子系统 (DEBUGSS) 将串行线调试(SWD) 两线制物理接口连接到器件内的多个调试功能。MSPM0 器件支持调试处理器执行情况、器件状态和电源状态（通过 EnergyTrace 技术）。DEBUGSS 还提供一个邮箱系统，可通过 SWD 与软件进行通信。

调试子系统提供的主要特性包括：

- 双线制（SWDIO、SWCLK）调试接口，与 TI 及第三方调试探针均兼容
  - 分别用于 SWDIO 和 SWCLK 的片上上拉和下拉电阻，默认启用
  - 支持禁用 SWD 功能，以便将 SWD 引脚用作通用输入/输出引脚
  - 能够在发生有效的 SWD 活动时将器件从 SHUTDOWN 模式唤醒
- 处理器调试
  - 运行、暂停和单步调试支持
  - 4 个硬件断点 (BPU)
  - 2 个硬件观察点 (DWT)
  - 通过 Arm 微跟踪缓冲器 (MTB) 对多达 4 个分支进行指令跟踪
  - 无限的软件断点
- 可通过软件配置处理器调试期间的外设行为
  - 能够通过调试暂停的方式自由运行某些外设
  - 能够在调试暂停时暂停某些外设
  - 能够向 PMCU 请求复位和模式更改
- 通过 EnergyTrace 技术监控 CPU 状态
- 使用邮箱 (DSSM) 在 SWD 接口和引导 ROM（以及应用软件）之间传递数据和控制信号
- 支持各种安全特性，包括 SWD 锁定和密码身份验证调试

26.1.1 调试互连

图 26-1 显示了 DEBUGSS 架构。

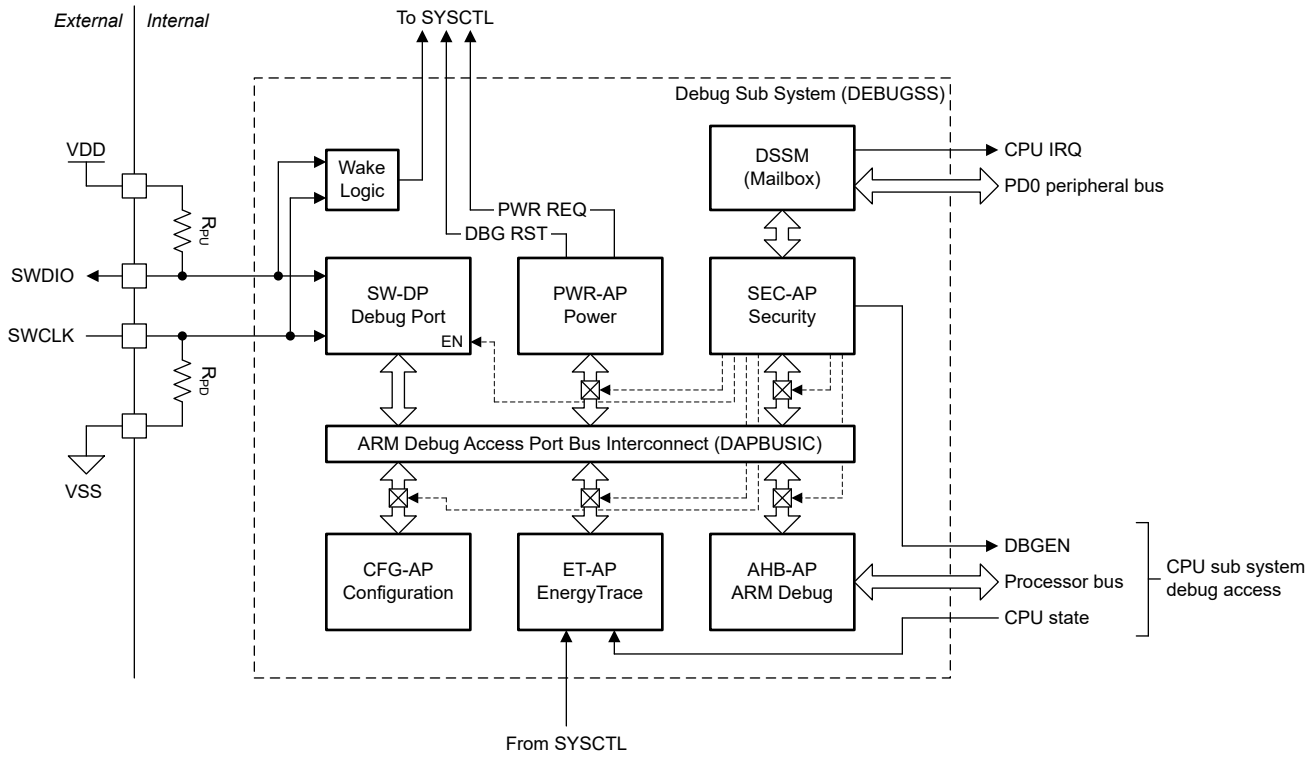


图 26-1. 调试子系统方框图

SWD 物理接口与 ARM 串行线调试端口 (SW-DP) 进行交互，以便在启用 SW-DP 时获得对调试访问端口总线互连 (DAPBUSIC) 的访问权限。TI 器件发货时启用了 SW-DP，允许 SWD 访问器件以进行开发和生产编程，但可以通过引导配置策略将 SW-DP 配置为永久禁用（请参阅节 26.4）。

DAPBUSIC 使调试探针能够访问一个或多个调试访问端口。为了使调试探针能够与访问端口通信，器件引导配置策略不得禁用 SW-DP 调试端口，也不得通过引导配置策略禁用目标访问端口。节 26.1.3 中介绍了可用的访问端口。

SWD 和 SW-DP 还包含发送到 PMCU 模块的信号，以支持调试生成的复位和工作模式更改（请参阅节 26.3）。

26.1.2 物理接口

支持通过一个符合 Arm 串行线调试 (SWD) 标准的接口来实现器件的调试连接。SWD 接口需要两个连接：

- 用于向器件发送数据以及从器件接收数据的双向数据线 (SWDIO)
- 由连接到器件的调试探针进行驱动的单向时钟线 (SWCLK)

SWD 接口使用器件的标准逻辑电平进行 SWD 通信。请参阅器件特定的数据表，了解给定电源电压 (VDD) 下的输入和输出逻辑电平。DEBUGSS 支持高达 10MHz 的 SWCLK 频率。

在 SWD 运行期间，SWDIO 线可由目标器件或调试探针驱动为高电平或低电平。由于任一器件都可以驱动该线路，因此当共享 SWDIO 线的所有权在器件和调试探针之间切换时，未驱动的时隙将作为 SWD 协议的一部分插入。SWDIO 线上的上拉电阻和 SWCLK 线上的下拉电阻的主要用途是在未连接调试探针时将 SWD 引脚置于已知状态。Arm 建议最小电阻为 100kΩ。内部上拉/下拉电阻满足此要求，因此 SWD 接口的正确运行不需要外部电阻。

上电复位 (POR) 后，MSPM0 器件将 SWD 引脚配置为 SWD 模式，在 SWDIO 线上启用内部上拉电，并在 SWCLK 线上启用内部下拉电阻。如果器件配置没有永久禁用所有 SWD 访问，则在引导过程中会启用 SWD 接口，并且可以将调试探针连接到 DEBUGSS。

如果软件将器件配置为进入 SHUTDOWN 模式，然后在激活 SWCLK 的情况下将调试探针连接到 SWD 引脚，则唤醒逻辑将触发从 SHUTDOWN 模式退出，并会产生 BOR。然后可以在 BOR 完成后建立与 DEBUGSS 的调试连接。

完成调试探针的物理连接后，必须从调试探针向目标器件发送配置序列，以便启动与 SW-DP 的有效 SWD 连接。无效序列不会将器件从 SHUTDOWN 模式唤醒。一旦应用了序列并建立了 SWD 连接，便可与启用的调试访问点进行通信，并通过将 DEBUGSS PWRUPIFG 中断设置为有效状态来提醒应用程序代码。当调试探针断开并且 SWD 连接中断时，PWRDWNIFG 中断设置为有效状态。

应用软件可以禁用 SYSCTL 中的 SWD 接口，从而释放 IO 以用于通用 IO 功能。请查看节 2.4.1.4，了解在 SYSCTL 中如何将 SWD 引脚用于 SWD 以外的功能。一旦软件禁用了 SWD 功能，除非触发 POR，否则不能重新启用该功能。POR 将自动重新启用 SWD 功能，并在启用上拉/下拉电阻的情况下将 SWD 引脚置于 SWD 模式。如果器件包含在启动时禁用 SWD 引脚的软件，为了重新获得对该器件的调试访问权限，必须在 POR 期间使用 NRST 引脚将器件保持在复位状态。这将阻止应用软件启动，并允许调试探针访问器件，此时可以通过调试探针从集成开发环境向器件发送批量擦除 DSSM 命令，来删除正在禁用 SWD 引脚的应用软件。

备注

BOR、BOOTRST 和 SYSRST 电平确实会复位 IOMUX 逻辑，这将重新启用 SWDIO/SWCLK 引脚上的上拉/下拉电阻。但是，在下一次 POR 之前，SWD 功能将保持禁用状态。由于器件始终在启用 SWDIO 上拉和 SWCLK 下拉电阻的情况下上电，因此在启动后将 SWD 引脚用于除 SWD 以外的其他功能时，硬件设计必须考虑到这一点。复位后，应用软件可能会禁用 IOMUX 中的上拉/下拉电阻器以释放 SWD 引脚用于其他目的。

26.1.3 调试访问端口

表 26-1 中列出了 DEBUGSS 中的调试访问端口。

表 26-1. DEBUGSS 访问端口列表

APSEL	AP	端口描述	用途
0x0	AHB-AP	MCPUSS 调试访问端口	处理器和外设的调试
0x1	CFG-AP	配置访问端口	访问器件类型信息
0x2	SEC-AP	安全访问端口	访问调试邮箱 (DSSM)
0x3	ET-AP	EnergyTrace™ 技术访问端口	读取 EnergyTrace 技术的电源状态数据来进行功率感知调试
0x4	PWR-AP	电源访问端口	配置器件电源状态（与 PMCU/SYSCTL 连接）

AHB-AP、PWR-AP 和 ET-AP 提供完整的器件调试功能（处理器调试、外设和存储器总线访问、电源状态控制和处理器状态）。有关更多信息，请参阅节 26.2。

CFG-AP 为调试探针提供器件信息，以便调试探针能够识别器件特性，包括器件型号和器件修订版本。

SEC-AP 提供对邮箱的访问，以便通过 SWD 与器件上运行的软件进行通信。有关更多信息，请参阅节 26.5。

26.2 调试特性

DEBUGSS 支持处理器调试、处理器跟踪、外设调试和能量状态调试。

26.2.1 处理器调试

Arm Cortex-M0+ 处理器支持多种特性，可在开发过程中简化应用软件的调试。MSPM0 MCU 支持的主要特性包括：

- 能够通过发出 HALT 信号、配置的调试事件（例如硬故障进入或复位）或 BKPT 指令（对于软件断点）来暂停处理器
- 能够单步执行指令（启用或不启用外设中断）
- 能够全面运行指令（启用或不启用外设中断）
- 能够在暂停时读取和写入 CPU 寄存器
- 能够通过 Cortex-M0+ 系统控制空间 (SCS) 读取异常信息
- 支持 4 个硬件断点
- 支持 2 个硬件观察点
- 支持访问器件存储器映射

26.2.1.1 断点单元 (BPU)

断点单元 (BPU) 提供 4 个比较器，当指令提取地址与编程到相应 BPU 比较器中的地址相匹配时，这些比较器可用于生成调试事件。

对于数据读取或数据写入访问，在地址匹配时，BPU 不会生成调试事件。

对于从 CODE 区域（0x0000.0000 到 0x1FFF.FFFF）获取的半字（16 位）指令和字（32 位）指令，可实现地址匹配。

如果一个调试方案需要不止四个断点，软件断点可与使用 BKPT 指令的硬件断点一起使用。如果需要在 SRAM 区域中调试代码，则硬件断点不可用，而软件断点必须由调试探针插入。

```
// Example of a breakpoint function in C (TI Arm CLANG compiler)
__BKPT(0);
```

26.2.1.2 数据观察点和跟踪单元 (DWT)

数据观察点和跟踪单元 (DWT) 提供 2 个比较器，它们都支持在数据地址匹配（观察点事件）或指令地址匹配（PC 观察点事件）时生成事件。

DWT 比较器支持地址屏蔽，当处理器尝试访问指定地址范围内的地址时，可以生成事件。

26.2.1.3 处理器跟踪 (MTB)

MSPM0G 器件支持基本指令执行跟踪，以便获取导致处理器特定状态的执行序列上下文。处理器跟踪引擎基于 Arm CoreSight MTB-M0+ 微跟踪缓冲器。

当处理器的程序计数器 (PC) 由于分支指令或异常而以非顺序方式变化时，MTB 可以捕获 PC 状态。MTB 不捕获加载和存储活动。MTB 在检测到非顺序执行时，会捕获更改并将其存储在一个小的缓冲存储器中（如表 26-2 所述），稍后应用软件或调试探针可以将其读出。

表 26-2. MTB 缓冲存储器

起始地址	终止地址	Length
0x4040.3000	0x4040.3020	32B（4 个跟踪数据包）

跟踪数据包数据

对于每次跟踪捕获，MTB 将源地址（分支的起始地址）和目标地址（分支的终止地址）存储到缓冲存储器中。因此，每个跟踪捕获数据包使用两个 32 位字。由于指令为半字对齐，地址的 LSB 不是必需的，因此可用于将有关状态的额外上下文存储到跟踪数据包中。

- 已记录的源地址的 LSB 由 MTB 进行修改，并被称为“A”位。“A”位根据分支时的处理器原子状态设置为 0 或 1。
  - 当捕获到“A”位为“0”时，分支源自于代码流中的指令。
  - 当捕获到“A”位为“1”时，分支源自于正在调试的异常或 PC 更新。
- 已记录的目标地址的 LSB 会被修改，并被称为“S”位。“S”位根据开始跟踪后捕获是否第一次发生而设置为 0 或 1。
  - 当捕获到“S”位为“1”时，数据包是开始跟踪后的第一个数据包。
  - 当捕获到“S”位为“0”时，数据包不是开始跟踪后的第一个数据包。

在返回异常的情况下，两个跟踪数据包会存储在缓冲存储器中：

1. 第一个数据包的“A”位设置为“0”并存储以下值：
   a. 源地址是导致返回异常的指令（BX 或 POP）的地址。
   b. 目标地址是 EXC_RETURN 值。
2. 第二个数据包的“A”位设置为“1”并存储以下值：
   a. 源地址是 EXC_RETURN 值。
   b. 目标地址是异常之后再次开始执行的指令的地址。

26.2.2 外设调试

除了处理器调试之外，还可以使用 DEBUGSS 从处理器的角度访问器件存储器映射。因此，可以使用连接的调试探针来读取和写入存储器映射的外设寄存器、系统 SRAM 以及闪存。

某些外设支持高级调试配置选项。这些选项由应用软件（或可选的调试探针）通过设置/清除给定外设的存储器映射中的各种调试控制位来进行配置。通常，特定外设的调试行为在每个外设的 PDBGCTL 寄存器中指定。许多外设提供以下选项：在暂停处理器以进行调试时暂停外设功能时钟，从而使外设与处理器一起暂停（默认配置）；或者即使暂停处理器以进行调试，仍然让外设运行。

例如，WWDT 外设支持 PDBGCTL 寄存器中的 FREE 位。设置 WWDT 的 PDBGCTL 中的 FREE 位，即使暂停处理器以进行调试，也会使 WWDT 计数器运行。

26.2.3 EnergyTrace 技术

MSPM0 器件中的 DEBUGSS 支持 EnergyTrace 技术。EnergyTrace 技术支持对运行应用代码的 MCU 器件进行功耗性能评测。这在开发必须针对低功耗运行进行优化的应用时非常有用。

德州仪器 (TI) 的开发工具（包括 MSPM0 LaunchPad 开发工具）支持随着时间的推移，通过 EnergyTrace 电荷计数对目标 MSPM0 进行硬件电能测量。这种机制使开发人员能够基于具有宽动态范围的实际电流测量结果，获取应用的电量使用情况。

为了给支持 EnergyTrace 技术的硬件开发工具进行的电能测量提供背景信息，MSPM0 MCU 还启用 EnergyTrace+。EnergyTrace+ 是 DEBUGSS 的一部分，可让调试探针在器件运行时记录处理器的状态（RUN、SLEEP）和当前程序计数器值。然后，这些状态信息会被电能测量值覆盖，以确定高电平电流的原因是处理器正在运行，还是器件上的一些其他活动。

TI 的 Code Composer Studio 集成开发环境为使用 MSPM0 器件进行 EnergyTrace 电能测量和 EnergyTrace+ 处理器状态记录提供开箱即用支持。

26.3 低功耗模式下的行为

在除 SHUTDOWN 外的所有工作模式中，DEBUGSS 均支持通过 SWD 保持调试连接。

在 RUN 模式和 SLEEP 模式下，可访问器件存储器和外设，而且，调试探针可主动连接到 AHB-AP 访问端口以连接处理器。在 STOP 和 STANDBY 模式下，可以与 DEBUGSS 建立和/或保持调试连接，但不能与 CPU 调试访问端口建立和/或保持连接。

在 SHUTDOWN 模式下，当调试逻辑通过器件 VCORE 断电时，任何有效的调试连接都会终止。虽然在器件处于 SHUTDOWN 模式时无法实现与 DEBUGSS 的调试连接，但调试探针可能会通过尝试与 SWD 引脚通信使器件退出 SHUTDOWN 模式。即使器件处于 SHUTDOWN 状态，器件也会检测尝试的 SWD 通信。如果检测到活动，则启动退出 SHUTDOWN，器件将通过 BOR 状态转换，之后可通过 SWD 与 DEBUGSS 建立调试连接。

表 26-3 中提供了不同工作模式下支持的 DEBUGSS 功能。

表 26-3. 不同工作模式下支持的 DEBUGSS 功能

功能	运行	睡眠	停止	待机	关断	NRST HOLD
处理器调试	是	是	否	否	否	否
存储器映射访问	是	是	否	否	否	否
通过 SW-DP 调试状态	是	是	是	是	否	是
保持调试状态	是	是	是	是	否	否
从 SWD 唤醒	-	-	-	-	是	-

26.4 限制调试访问

调试子系统支持多种方法来限制通过 SWD 接口访问器件。调试访问策略由 NONMAIN 闪存区域中指定的用户配置决定。

表 26-4 中列出了 3 种访问控制级别。默认情况下，从 TI 发货的产品在器件完全打开的情况下处于“启用调试”状态。不建议在生产环境中使用此状态。对于生产环境，TI 建议将调试配置更改为受密码保护或禁用。

表 26-4. 调试访问控制

DEBUGSS 函数	调试配置		
	启用调试（默认）	使用密码启用调试	禁用调试
SW-DP（调试端口）	EN	EN	DIS
CFG-AP	EN	EN	DIS
SEC-AP	EN	EN	DIS
ET-AP	EN	EN w/ PW	DIS
AHB-AP（CPU 调试）	EN	EN w/ PW	DIS

使用密码启用调试时，调试探针必须将调试访问命令与用户指定的调试访问密码一起提供给 DEBUGSS 邮箱，并且必须发出 BOOTRST。

禁用调试时，将在引导过程中禁用 SW-DP，并在引导期间忽略之前发送到邮箱的任何命令。引导后，将忽略任何连接到 SW-DP 的尝试。

通过将 NONMAIN 闪存区域配置为禁用调试访问，同时将 NONMAIN 闪存区域配置为受静态写保护（锁定），可以永久锁定对器件的调试访问。锁定 NONMAIN 配置可提高安全性，防止引导加载程序 (BSL) 和应用程序代码更改调试安全策略。

26.5 邮箱 (DSSM)

调试子系统邮箱 (DSSM) 使调试探针能够通过 SWD 接口将消息传递到目标器件，以及使目标器件能够将数据返回到调试探针。

DSSM 支持以下功能：

- 在启动期间向器件发送命令，包括对调试探针进行身份验证以执行受密码保护的调试、批量擦除和恢复出厂设置操作
- 当不存在其他通信接口时，与目标器件上运行的应用软件进行通信

为 TX 数据（调试探针到目标器件）和 RX 数据（目标器件到调试探针）提供了两个 32 位字数据缓冲区。这些数据缓冲区在 DEBUGSS 中实现为 32 位存储器映射寄存器。此外，还提供了 TXCTL 和 RXCTL 寄存器，用于启用流控制和指示邮箱的状态。

表 26-5. DSSM 寄存器功能

DSSM 寄存器	说明	调试探针	目标器件	操作
TX_DATA	数据缓冲区	RW	R	TXCTL.TRANSMIT 在调试探针进行写入时设置，并在目标器件进行读取时清零；TXIFG 也在调试探针进行写入时设置
TXCTL	流控制和状态	RW	R	无
RX_DATA	数据缓冲区	R	RW	RXCTL.RECEIVE 在目标器件进行写入时设置，并在调试探针进行读取时清零；RXIFG 也在目标器件进行写入时设置
RXCTL	流控制和状态	R	RW	无

TXCTL 和 RXCTL 寄存器分别在 BIT0 位置提供 TRANSMIT 和 RECEIVE 标志。当调试探针向 TX_DATA 缓冲区寄存器写入数据时，将设置 TXCTL 寄存器中的 TRANSMIT 位。然后，TRANSMIT 标志将保持为设置状态，直到目标器件读取 TX_DATA 或发生 POR。当目标器件向 RX_DATA 缓冲区寄存器写入数据时，将设置 RXCTL 寄存器中的 RECEIVE 标志。然后，RECEIVE 标志将保持为设置状态，直到调试探针从 RX_DATA 读取数据。

目标器件上运行的软件无法写入 TX_DATA，并且，除了通过读取 TX_DATA 之外，目标软件也无法清除 TRANSMIT 标志。TXCTL 寄存器的高 31 位包含通用标志位，如果需要，调试探针可以设置或清除这些标志位以实现协议。只有调试探针才能写入 TXCTL 中的 TRANSMIT_FLAGS 字段。

类似地，只有目标器件软件才可以写入 RX_DATA 和 RXCTL。调试探针无法写入 RX_DATA，它只能通过读取 RX_DATA 来清除 RXCTL 中的 RECEIVE 标志。RXCTL 的 BIT1 到 BIT7 (0xFE) 包含 RECEIVE_FLAGS 字段。如果需要，目标器件上的软件可以设置或清除 RECEIVE_FLAGS 字段中的位以实现协议。这些标志可以由调试探针读取，但不能由调试探针修改。

有关启动配置例程在器件启动配置期间支持的 DSSM 命令的完整列表，请参阅节 1.4。

26.5.1 DSSM 事件

DSSM 包含一个事件发布者，没有事件订阅者。一个事件发布者 (CPU_INT) 通过一个静态事件路由来管理到 CPU 子系统的 DSSM 中断请求 (IRQ)。

表 26-6 中总结了 DSSM 事件。

表 26-6. DSSM 事件

事件	类型	源	目标	路由	配置	功能
CPU 中断事件	发布者	DEBUGSS	CPU 子系统	静态路由	CPU_INT 寄存器	修复了从 DEBUGSS 到 CPU 的中断路由

26.5.1.1 CPU 中断事件 (CPU_INT)

DSSM 提供 4 个中断源，这些中断源可配置为产生 CPU 中断事件。为了降低中断优先级，表 26-7 中列出了来自 DSSM 的 CPU 中断事件。

表 26-7. DSSM CPU 中断事件条件 (CPU_INT)

索引 (IIDX)	名称	说明
0	TXIFG	表示 DSSM 中的 TX_DATA 缓冲器已接收数据。
1	RXIFG	指示读取了 DSSM 中 RX_DATA 缓冲区中的数据。
2	PWRUPIFG	指示由于一个调试探针连接到器件而启动了 DEBUGSS。
3	PWRDWNIFG	指示由于调试探针断开与器件的连接而停止了 DEBUGSS。

CPU 中断事件配置通过 CPU_INT 事件管理寄存器进行管理。有关为 CPU 中断配置事件寄存器的指导，请参阅节 7.2.5。

26.5.2 DEBUGSS 寄存器

表 26-8 列出了 DEBUGSS 寄存器的存储器映射寄存器。表 26-8 中未列出的所有寄存器偏移地址都应视为保留的位置，并且不应修改寄存器内容。

表 26-8. DEBUGSS 寄存器

偏移	缩写	寄存器名称	组	部分
1020h	IIDX	中断索引	CPU_INT	转到
1028h	IMASK	中断屏蔽	CPU_INT	转到
1030h	RIS	原始中断状态	CPU_INT	转到
1038h	MIS	已屏蔽中断状态	CPU_INT	转到
1040h	ISET	中断设置	CPU_INT	转到
1048h	ICLR	中断清除	CPU_INT	转到
10E0h	EVT_MODE	事件模式		转到
10FCh	DESC	模块说明		转到
1100h	TXD	发送数据寄存器		转到
1104h	TXCTL	发送控制寄存器		转到
1108h	RXD	接收数据寄存器		转到
110Ch	RXCTL	接收控制寄存器		转到
1200h	SPECIAL_AUTH	特殊使能授权寄存器		转到
1210h	APP_AUTH	应用 CPU0 授权寄存器		转到

复杂的位访问类型经过编码可适应小型表单元。表 26-9 显示了适用于此部分中访问类型的代码。

表 26-9. DEBUGSS 访问类型代码

访问类型	代码	说明
读取类型		
R	R	读取
R-0	R -0	读取 返回 0
写入类型		
W	W	写入
WK	W K	写入 受密钥保护的写入
复位或默认值		
-n		复位后的值或默认值

26.5.2.1 IIDX (偏移 = 1020h) [复位 = 00000000h]

图 26-2 展示了 IIDX，表 26-10 中对此进行了介绍。

返回到汇总表。

该寄存器提供了具有最高优先级的中断索引。0xFF 表示没有事件挂起。中断 0x0 是最高优先级，0x1 是次要优先级，而 0xFE 是最低优先级。优先级顺序是固定的。但是，用户可以使用其他寄存器来实现自己的优先级方案，这些寄存器显示了已经发生的中断的完整集合。
每次读取时，仅指示一个中断。读取时，当前中断 (最高优先级) 由硬件自动清除，同时 RIS 和 MIS 中相应的中断标志也会被清除。从 CPU (不是从调试接口) 读取后，必须使用下一个最高优先级中断更新该寄存器，如果没有中断挂起，则显示 0xFF。

图 26-2. IIDX

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

表 26-10. IIDX 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R	0h	
7-0	STAT	R	0h	中断索引状态 0h = 无挂起中断请求 1h = TX 中断 2h = RX 中断 3h = 上电中断。调试会话已启动。 4h = 上电中断。调试会话已启动。

26.5.2.2 IMASK（偏移 = 1028h）[复位 = 00000000h]

图 26-3 展示了 IMASK，表 26-11 中对此进行了介绍。

返回到汇总表。

中断屏蔽。如果设置了某个位，相应的中断会被取消屏蔽。取消屏蔽中断会导致原始中断显示在 IIDX 以及 MIS 中。

图 26-3. IMASK

31	30	29	28	27	26	25	24
RESERVED							
R/W-0h							
23	22	21	20	19	18	17	16
RESERVED							
R/W-0h							
15	14	13	12	11	10	9	8
RESERVED							
R/W-0h							
7	6	5	4	3	2	1	0
ESERVED				PWRDWNIFG	PWRUPIFG	RXIFG	TXIFG
R/W-0h				R/W-0h	R/W-0h	R/W-0h	R/W-0h

表 26-11. IMASK 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R/W	0h	
3	PWRDWNIFG	R/W	0h	屏蔽 MIS 寄存器中的 PWRDWNIFG 0h = 中断会被屏蔽掉 1h = 中断将请求一个中断处理例程，并且 MIS 中的相应位将被置位
2	PWRUPIFG	R/W	0h	屏蔽 MIS 寄存器中的 PWRUPIFG 0h = 中断会被屏蔽掉 1h = 中断将请求一个中断处理例程，并且 MIS 中的相应位将被置位
1	RXIFG	R/W	0h	屏蔽 MIS 寄存器中的 RXIFG 0h = 中断会被屏蔽掉 1h = 中断将请求一个中断处理例程，并且 MIS 中的相应位将被置位
0	TXIFG	R/W	0h	屏蔽 MIS 寄存器中的 TXIFG 0h = 中断会被屏蔽掉 1h = 中断将请求一个中断处理例程，并且 MIS 中的相应位将被置位

26.5.2.3 RIS（偏移 = 1030h）[复位 = 00000000h]

图 26-4 展示了 RIS，表 26-12 中对此进行了介绍。

返回到汇总表。

原始中断状态。反映所有挂起的中断，而不管屏蔽与否。RIS 寄存器允许用户实施轮询方案。即使相应的 IMASK 位未启用，也可以通过向 ICLR 寄存器位写入 1 来清除该寄存器中设置的标志。

图 26-4. RIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				PWRDWNIFG	PWRUPIFG	RXIFG	TXIFG
R-0h				R-0h	R-0h	R-0h	R-0h

表 26-12. RIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	PWRDWNIFG	R	0h	PWRDWNIFG 的原始中断状态 0h = 未发生 PWRUPIFG 1h = 已发生 PWRUPIFG
2	PWRUPIFG	R	0h	PWRUPIFG 的原始中断状态 0h = 未发生 PWRUPIFG 1h = 已发生 PWRUPIFG
1	RXIFG	R	0h	RXIFG 的原始中断状态 0h = 未发生 RXIFG 1h = 已发生 RXIFG
0	TXIFG	R	0h	TXIFG 的原始中断状态 0h = 未发生 TXIFG 1h = 已发生 TXIFG

26.5.2.4 MIS（偏移 = 1038h）[复位 = 00000000h]

图 26-5 展示了 MIS，表 26-13 中对此进行了介绍。

返回到汇总表。

屏蔽中断状态。这是 IMASK 和 RIS 寄存器的与运算。

图 26-5. MIS

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
ESERVED				PWRDWNIFG	PWRUPIFG	RXIFG	TXIFG
R-0h				R-0h	R-0h	R-0h	R-0h

表 26-13. MIS 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	PWRDWNIFG	R	0h	PWRDWNIFG 的屏蔽中断状态 0h = PWRUPIFG 没有请求中断服务例程 1h = PWRUPIFG 请求一个中断服务例程
2	PWRUPIFG	R	0h	PWRUPIFG 的屏蔽中断状态 0h = PWRUPIFG 没有请求一个中断处理例程 1h = PWRUPIFG 请求一个中断处理例程
1	RXIFG	R	0h	RXIFG 的屏蔽中断状态 0h = RXIFG 没有请求一个中断处理例程 1h = RXIFG 请求一个中断处理例程
0	TXIFG	R	0h	TXIFG 的屏蔽中断状态 0h = TXIFG 没有请求一个中断处理例程 1h = TXIFG 请求一个中断处理例程

26.5.2.5 ISET（偏移 = 1040h）[复位 = 00000000h]

图 26-6 展示了 ISET，表 26-14 中对此进行了介绍。

返回到汇总表。

中断设置。允许通过软件设置中断（在诊断和安全检查中很有用）。向 ISET 中的某个位写入 1 将设置事件，因此相关的 RIS 位也会置位。如果通过屏蔽启用了中断，那么也会设置相应的 MIS 位。

图 26-6. ISET

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				PWRDWNIFG	PWRUPIFG	RXIFG	TXIFG
W-0h				W-0h	W-0h	W-0h	W-0h

表 26-14. ISET 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	PWRDWNIFG	W	0h	设置 RIS 寄存器中的 PWRDWNIFG 0h = 写入 0 不会产生影响 1h = 对应于 PWRUPIFG 的 RIS 位会被置位
2	PWRUPIFG	W	0h	设置 RIS 寄存器中的 PWRUPIFG 0h = 写入 0 不会产生影响 1h = 对应于 PWRUPIFG 的 RIS 位会被置位
1	RXIFG	W	0h	设置 RIS 寄存器中的 RXIFG 0h = 写入 0 不会产生影响 1h = 对应于 RXIFG 的 RIS 位会被置位
0	TXIFG	W	0h	设置 RIS 寄存器中的 TXIFG 0h = 写入 0 不会产生影响 1h = 对应于 TXIFG 的 RIS 位会被置位

26.5.2.6 ICLR（偏移 = 1048h）[复位 = 00000000h]

图 26-7 展示了 ICLR，表 26-15 中对此进行了介绍。

返回到汇总表。

中断清除。写入 1 以清除相应的中断。

图 26-7. ICLR

31	30	29	28	27	26	25	24
RESERVED							
W-0h							
23	22	21	20	19	18	17	16
RESERVED							
W-0h							
15	14	13	12	11	10	9	8
RESERVED							
W-0h							
7	6	5	4	3	2	1	0
ESERVED				PWRDWNIFG	PWRUPIFG	RXIFG	TXIFG
W-0h				W-0h	W-0h	W-0h	W-0h

表 26-15. ICLR 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	W	0h	
3	PWRDWNIFG	W	0h	清除 RIS 寄存器中的 PWRDWNIFG 0h = 写入 0 不会产生影响 1h = 对应于 PWRUPIFG 的 RIS 位被清零
2	PWRUPIFG	W	0h	清除 RIS 寄存器中的 PWRUPIFG 0h = 写入 0 不会产生影响 1h = 对应于 PWRUPIFG 的 RIS 位被清零
1	RXIFG	W	0h	清除 RIS 寄存器中的 RXIFG 0h = 写入 0 不会产生影响 1h = 对应于 RXIFG 的 RIS 位被清零
0	TXIFG	W	0h	清除 RIS 寄存器中的 TXIFG 0h = 写入 0 不会产生影响 1h = 对应于 TXIFG 的 RIS 位被清零

26.5.2.7 EVT_MODE（偏移 = 10E0h）[复位 = 00000001h]

图 26-8 展示了 EVT_MODE，表 26-16 中对此进行了介绍。

返回到汇总表。

事件模式寄存器。它用于选择在软件模式（软件清除 RIS）或硬件模式（硬件清除 RIS）下是否禁用每条线

图 26-8. EVT_MODE

31	30	29	28	27	26	25	24
RESERVED							
R-							
23	22	21	20	19	18	17	16
RESERVED							
R-							
15	14	13	12	11	10	9	8
RESERVED							
R-							
7	6	5	4	3	2	1	0
ESERVED						INT0_CFG	
R-						R-1h	

表 26-16. EVT_MODE 字段说明

位	字段	类型	复位	说明
31-2	RESERVED	R	0h	
1-0	INT0_CFG	R	1h	外设事件的事件线模式选择 0h = 中断或事件线禁用。 1h = 中断或事件线处于软件模式。软件必须清除 RIS。 2h = 中断或事件线处于硬件模式。硬件（另一个模块）会自动清除关联的 RIS 标志。

26.5.2.8 DESC（偏移 = 10FCh）[复位 = 03400000h]

图 26-9 展示了 DESC，表 26-17 中对此进行了介绍。

返回到汇总表。

该寄存器标识外设及其确切版本。

图 26-9. DESC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODULEID															
R-340h															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEATUREVER				INSTNUM				MAJREV				MINREV			
R-0h				R-0h				R-0h				R-0h			

表 26-17. DESC 字段说明

位	字段	类型	复位	说明
31-16	MODULEID	R	340h	模块标识包含唯一的外设标识号。所有平台模块的分配都保存在中央数据库中，可确保唯一性。
15-12	FEATUREVER	R	0h	模块 *实例* 的功能集
11-8	INSTNUM	R	0h	器件中的实例编号。对于具有多个实例的模块，这将是 RTL 的参数
7-4	MAJREV	R	0h	IP 的主要版本
3-0	MINREV	R	0h	IP 的次要版本

26.5.2.9 TXD (偏移 = 1100h) [复位 = 00000000h]

图 26-10 展示了 TXD，表 26-18 中对此进行了介绍。

返回到汇总表。

该寄存器用于从外部调试工具到 DSSM 模块的数据传输。该寄存器由调试工具写入并由 CPU 读取。

图 26-10. TXD

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

表 26-18. TXD 字段说明

位	字段	类型	复位	说明
31-0	TX_DATA	R	0h	包含由外部调试工具写入 SEC-AP TXDATA 寄存器的数据

26.5.2.10 TXCTL（偏移 = 1104h）[复位 = 00000000h]

图 26-11 展示了 TXCTL，表 26-19 中对此进行了介绍。

返回到汇总表。

发送控制寄存器

图 26-11. TXCTL

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

表 26-19. TXCTL 字段说明

位	字段	类型	复位	说明
31-1	TRANSMIT_FLAGS	R	0h	可由外部调试工具设置的通用 TX 标志。功能由 SW 定义。
0	发送	R	0h	指示 DSSM.TXD 中的数据请求，在通过调试 AP 写入 DSSM.TXD 时置 1。 通过 SW 读取 DSSM.TXD 寄存器时将清除 TX 字段。该工具可以通过读取该字段来检查 TXD 是否为空。 0h = TXD 为空 1h = TXD 已满

26.5.2.11 RXD（偏移 = 1108h）[复位 = 00000000h]

图 26-12 展示了 RXD，表 26-20 中对此进行了介绍。

返回到汇总表。

接收数据寄存器。该寄存器包含由 CPU 写入的数据。
该数据由外部调试工具读取。

图 26-12. RXD

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

表 26-20. RXD 字段说明

位	字段	类型	复位	说明
31-0	RX_DATA	R/W	0h	包含由 SM/OW 写入的数据。

26.5.2.12 RXCTL（偏移 = 110Ch）[复位 = 00000000h]

图 26-13 展示了 RXCTL，表 26-21 中对此进行了介绍。

返回到汇总表。

接收控制寄存器

图 26-13. RXCTL

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

表 26-21. RXCTL 字段说明

位	字段	类型	复位	说明
31-8	RESERVED	R/W	0h	
7-1	RECEIVE_FLAGS	R/W	0h	可由 SW 设置并由外部调试工具读取的通用 RX 标志。功能由 SW 定义。
0	RECEIVE	R	0h	指示 SW 对 DSSM.RXD 寄存器的写入操作。 通过 SWD 访问端口读取 DSSM.RXD 寄存器时将清除 RX 字段。 0h = RXD 为空 1h = RXD 已满

26.5.2.13 SPECIAL_AUTH（偏移 = 1200h）[复位 = 00000013h]

图 26-14 展示了 SPECIAL_AUTH，表 26-22 中对此进行了介绍。

返回到汇总表。

该寄存器用于控制 ET-AP、DFT-TAP、SWD、CFG-AP 和 SEC-AP。

图 26-14. SPECIAL_AUTH

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	PWRAPEN	AHBAPEN	CFGAPEN	ETAPEN	DFTAPEN	SWDPORTEN	SECAPEN
R-0h	R-0h	R-0h	R-1h	R-0h	R-0h	R-1h	R-1h

表 26-22. SPECIAL_AUTH 字段说明

位	字段	类型	复位	说明
31-7	RESERVED	R	0h	
6	PWRAPEN	R	0h	高电平有效输入。当该位为有效状态（且 SWD 访问也被允许）时，调试工具可以访问 PWR-AP 来获取 CPU 的电源和复位状态。当该位为无效状态时，DAPBUS 防火墙将隔离 AP 并阻止访问。 0h = 禁用 PWR-AP 1h = 启用 PWR-AP
5	AHBAPEN	R	0h	通过 AHB-AP DAP 总线隔离来禁用或启用对 M0+ 内核的调试访问。 0h = 禁用 AHB-AP 1h = 启用 AHB-AP
4	CFGAPEN	R	1h	高电平有效输入。当该位为有效状态（且 SWD 访问也被允许）时，调试工具可以通过 Config-AP 来读取器件配置信息。当该位为无效状态时，DAPBUS 防火墙将隔离 AP 并阻止访问 Config-AP。 0h = 禁用 CFG-AP 1h = 启用 CFG-AP
3	ETAPEN	R	0h	高电平有效输入。当该位为有效状态（且 SWD 访问也被允许）时，调试工具可以访问位于 DebugSS Lite 外部的 ET-AP。当该位为无效状态时，DAPBUS 防火墙将隔离 AP 并阻止访问。 0h = 禁用 ET+ -AP 1h = 启用 ET+ -AP
2	DFTAPEN	R	0h	高电平有效输入。当该位为有效状态（且 SWD 访问也被允许）时，调试工具可以访问位于 DebugSS Lite 外部的 DFT-AP。当该位为无效状态时，DAPBUS 防火墙将隔离 AP 并阻止访问。 0h = 禁用 DFT-TAP 1h = 启用 DFT-TAP
1	SWDPORTEN	R	1h	当该位为有效状态时，SW-DP 功能正常。 当该位为无效状态时，SW-DP 会有效地禁用所有外部调试访问。 0h = 禁用 SWD 端口 1h = 启用 SWD 端口

表 26-22. SPECIAL_AUTH 字段说明 (continued)

位	字段	类型	复位	说明
0	SECAPEN	R	1h	高电平有效输入。当该位为有效状态 (且 SWD 访问也被允许) 时，调试工具可以通过 Security-AP 来与安全控制逻辑通信。当该位为无效状态时，DAPBUS 防火墙将隔离 AP 并阻止访问 Security-AP。 0h = 禁用 SEC-AP 1h = 启用 SEC-AP

26.5.2.14 APP_AUTH（偏移 = 1210h）[复位 = 00000000h]

图 26-15 展示了 APP_AUTH，表 26-23 中对此进行了介绍。

返回到汇总表。

该寄存器用于控制应用 CPU0 的 DBGEN、NIDEN、SPIDEN 和 SPNIDEN。
DBGEN 和 NIDEN 由 DSW 根据有效和调试 IPF ID 进一步进行处理。

图 26-15. APP_AUTH

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
保留							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				SPNIDEN	SPIDEN	NIDEN	DBGEN
R-0h				R-0h	R-0h	R-0h	R-0h

表 26-23. APP_AUTH 字段说明

位	字段	类型	复位	说明
31-4	RESERVED	R	0h	
3	SPNIDEN	R	0h	安全非侵入式调试启用。 0h = 禁用侵入式调试 1h = 启用侵入式调试
2	SPIDEN	R	0h	控制是否启用安全侵入式调试。 0h = 禁用侵入式调试 1h = 启用侵入式调试
1	NIDEN	R	0h	控制是否启用非侵入式调试。 0h = 禁用非侵入式调试 1h = 启用非侵入式调试
0	DBGEN	R	0h	控制是否启用侵入式调试。 0h = 禁用侵入式调试 1h = 启用侵入式调试

注：以前版本的页码可能与当前版本的页码不同

日期	修订版本	说明
June 2023	*	初始发行版

This page intentionally left blank.

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265