LM3S617 微控制器

数据手册

Rev.00

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 15 楼 F1

网址：http://www.zlgmcu.com

技术支持

如果您对文档有所疑问，您可以在办公时间（星期一至星期五上午 8:30~11:50；下午 1:30~5:30；星期六上午 8:30~11:50）拨打技术支持电话或 E-mail 联系。

网　　址：　www.zlgmcu.com

联系电话：　+86 (020)　22644358　22644359　22644360　22644361

E-mail：　zlgmcu.support@zlgmcu.com

销售与服务网络

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 15 楼 F1　邮编：510630

电话：(020)38730916　38730917　38730976　38730977

传真：(020)38730925

网址：http://www.zlgmcu.com

广州专卖店

地址：广州市天河区新赛格电子城 203-204 室

电话：(020)87578634　87569914

传真：(020)87578842

南京周立功

地址：南京市珠江路 280 号珠江大厦 2006 室

电话：(025)83613221　83613271　83603500

传真：(025)83613271

北京周立功

地址：北京市海淀区知春路 113 号银网中心 715 室（中发电子市场斜对面）

电话：(010)62536178　62536179　82628073

传真：(010)82614433

重庆周立功

地址：重庆市石桥铺科园一路二号大西洋国际大厦（赛格电子市场）1115 室

电话：(023)68796438　68796439

传真：(023)68796439

杭州周立功

地址：杭州市登云路 428 号浙江时代电子市场 205 号

电话：(0571)88009205　88009932　88009933

传真：(0571)88009204

成都周立功

地址：成都市一环路南一段 57 号金城大厦 612 室

电话：(028)85399320　85437446

传真：(028)85439505

深圳周立功

地址：深圳市深南中路 2070 号电子科技大厦 A 座 24 楼 2403 室

电话：(0755)83781768　83781788　83782922

传真：(0755)83793285

武汉周立功

地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室（华中电脑数码市场）

电话：(027)87168497　87168297　87168397

传真：(027)87163755

上海周立功

地址：上海市北京东路 668 号科技京城东座 7E 室

电话：(021)53083452　53083453　53083496

传真：(021)53083491

西安办事处

地址：西安市长安北路 54 号太平洋大厦 1201 室

电话：(029)87881296　83063000　85399492

传真：(029)87880865

目录

第1章 结构概述

Luminary Micro Stellaris®系列微控制器是首款基于 ARM® Cortex™-M3 的控制器，它将高性能的 32 位计算引入到对价格敏感的嵌入式微控制器应用中。这些堪称先锋的器件，价格与 8 位和 16 位器件相同，却能为用户提供 32 位器件的性能，并且所有器件都是以小型封装的形式提供。

Stellaris 系列的 LM3S617 微控制器拥有 ARM 微控制器所具有的众多优点，如拥有广泛使用的开发工具，片上系统（SoC）的底层结构 IP 的应用，以及众多的用户群体。此外，控制器还采用了 ARM 可兼容 Thumb®的 Thumb-2 指令集来降低内存的需求量，进而降低成本。

Luminary Micro 提供了一套完整的解决方案以便快速打入市场，其中包括了用户开发板、白皮书和应用手册，以及强大的支持、销售和分销商网络。

1.1 产品特性

LM3S617 微控制器包含以下的产品特性：

- 32 位 RISC 性能
  - 采用为小型嵌入式应用方案而优化的 32 位 ARM® Cortex™ M3 v7M 结构
  - 可兼容 Thumb®的 Thumb-2 专用指令集处理器内核，可提高代码密度
  - 50-MHz 工作频率
  - 硬件除法和单周期乘法
  - 集成了嵌套向量中断控制器（NVIC）以提供明确的中断处理
  - 26 个中断，带 8 个优先级
  - 存储器保护单元（MPU）为受保护的操作系统功能提供了一种特权模式
  - 非对齐式的数据访问，使数据可以有效地压入内存
  - 极细微的位元处理操作(atomic bit-banding)可最大限度地使用内存，并且提供精简（streamlined）的外设控制
- 内部存储器
  - 32KB 单周期 Flash
    - 用户管理的 Flash 块保护，以 2KB 块大小为基础
    - 用户管理的 Flash 数据编程
    - 用户定义和管理的 Flash 保护块
  - 8KB 单周期 SRAM
- 通用定时器
  - 3 个定时器，每个都可配置为一个 32 位定时器、两个 16 位定时器或启动一个 ADC 事件
  - 32 位定时器模式：
    - 可编程的单次触发(one-shot)定时器

- 可编程的周期定时器
- 使用外部 32.768-KHz 时钟作为输入时的实时时钟
- 在周期或者单次触发的定时器模式下进行调试期间，当控制器使 CPU 的暂停(Halt)标志有效时，暂停操作(stalling)可由用户来使能
- ADC 事件触发器

— 16 位定时器模式

- 带有 8 位预分频器的通用定时器功能
- 可编程的单次触发定时器
- 可编程的周期定时器
- 调试期间，当控制器使 CPU 的暂停(Halt)标志有效时，暂停操作(stalling)可由用户来使能
- ADC 事件触发器

— 16 位输入捕获模式

- 输入边沿计数捕获
- 输入边沿时间捕获

— 16 位 PWM 模式

- 简单 PWM 模式，可通过软件编程来控制 PWM 信号的输出反相（inversion）

- 遵循 ARM FiRM 规范的看门狗定时器
  — 带有可编程装载寄存器的 32 位递减计数器（down counter）
  — 带有使能的独立看门狗时钟
  — 带有中断屏蔽的可编程中断产生逻辑
  — 提供锁定寄存器保护，以防止软件跑飞（runaway）的情况
  — 带有使能/禁能的复位产生逻辑
  — 在调试期间，当控制器使 CPU 的暂停（Halt）标志有效时的暂停操作（stalling）可由用户来使能
- 同步串行接口(SSI)
  — 主机或从机操作
  — 可编程的时钟位速率和预分频
  — 独立的发送和接收 FIFO，16 位宽、8 单元深
  — Freescale SPI、MICROWIRE 或 Texas 仪器同步串行接口的可编程接口操作
  — 4 到 16 位的可编程数据帧大小
  — 用于诊断/调试测试的内部回送（loopback）测试模式
- UART
  — 2 个完全可编程的 16C550-型 UART
  — 独立的 16×8 发送(Tx)和 16×12 接收(Rx) FIFO，减少 CPU 中断服务的负载

- 带小数分频器的可编程波特率发生器
- 可编程的 FIFO 长度，包含提供常用的双缓冲接口的 1 字节深的操作
- FIFO 触发深度为 1/8, 1/4, 1/2, 3/4 和 7/8
- 用于起始、停止和奇偶校验的标准异步通信位
- 错误-起始-位检测
- 线中断（line-break）的产生和检测

- ADC
  - 单端输入和差分输入配置
  - 用作单端输入时，含 6 个 10 位通道（输入）
  - 采样率：500,000 次采样/秒
  - 灵活、可配置的模数转换
  - 4 个可编程的采样转换序列，1 到 8 个入口（entry），每个序列均带有相应的转换结果 FIFO
  - 每个序列由软件或内部事件（定时器、模拟比较器、PWM 或 GPIO）触发
- 模拟比较器
  - 1 个独立集成的模拟比较器
  - 输出可以配置为驱动输出管脚、产生中断或启动一个 ADC 采样序列（sample sequence）
  - 可将外部管脚输入与外部管脚输入或内部可编程参考电压进行比较
- PWM
  - 3 个 PWM 发生器模块，每个模块含有 1 个 16 位计数器、2 个比较器、1 个 PWM 发生器和 1 个死区（dead-band）发生器
  - 1 个 16 位计数器
    - 在递减（Down）或递增/递减（Up/Down）计数模式下运行
    - 输出频率由 16 位装载值控制
    - 装载值的更新可以同步进行
    - 在计数值为 0 或为装载值时产生输出信号
  - 2 个比较器
    - 比较器值的更新可以同步进行
    - 在值相等时产生输出信号
  - PWM 发生器
    - 输出 PWM 信号根据计数器和比较器输出信号引发的操作来构造。
    - 产生 2 个独立的 PWM 信号
  - 死区（Dead-band）发生器
    - 产生 2 个带有可编程死区延迟的 PWM 信号，适于驱动半 H 桥（half-H bridge）

- 可以被旁路，不对输入 PWM 信号进行修改

— 灵活的输出控制块，可以对每个 PWM 信号进行输出使能

- 可以对每个 PWM 信号进行输出使能
- 可以选择将每个 PWM 输出信号进行反相（极性控制）
- 可以选择对每个 PWM 信号进行故障处理
- PWM 发生器模块中含有定时器同步机制
- PWM 发生器模块中含有定时器/比较器更新的同步机制
- PWM 发生器模块汇总了所有的中断状态

— 可以启动一个 ADC 采样序列

- GPIO
  - — 1～30 个 GPIO，视具体配置而定
  - — 输入/输出端口可承受 5V 电压
  - — 中断触发可以编程为边沿触发或电平触发
  - — 在读和写操作中通过地址线进行位屏蔽
  - — 可以启动一个 ADC 采样序列
  - — GPIO 端口配置的可编程控制
    - 弱上拉或下拉电阻
    - 2-mA, 4-mA 和 8-mA 端口驱动
    - 8-mA 驱动的斜率控制
    - 开漏使能
    - 数字输入使能
- 电源
  - — 片内低压差线性稳压器（LDO），具有可编程输出电压，用户可调节的范围为 2.25V～2.75V
  - — 控制器上的低功耗选项：睡眠和深度睡眠模式
  - — 外设的低功耗选项：软件控制单个外设的关断
  - — LDO 未调整电压检测和自动复位由用户控制使能
  - — 带 3.3V 电源掉电检测，可通过中断或者复位报告该情况
  - — 片内温度传感器
- 灵活的复位源
  - — 上电复位(POR)
  - — 复位脚生效
  - — 掉电（BOR）检测器可向系统报警电源电压下降的情况
  - — 软件复位
  - — 看门狗定时器复位
  - — 内部低压差线性稳压器（LDO）输出变成未调节

- 其它特性
  - 6 个复位源
  - 可编程的时钟源控制
  - 可选通单个外设时钟以节省功耗
  - 符合 IEEE 1149.1-1990 标准的测试访问端口（TAP）控制器
  - 通过 JTAG 和串行线接口的调试访问
  - 完整的 JTAG 边界扫描
- 在工业范围内符合 RoHS 标准的 48 脚 LQFP 封装

1.2 目标应用

- 工厂自动化和控制
- 工业控制的电源设备
- 楼宇和家庭自动化
- 步进电动机
- 无刷直流电动机
- 交流感应电动机

1.3 高级方框图

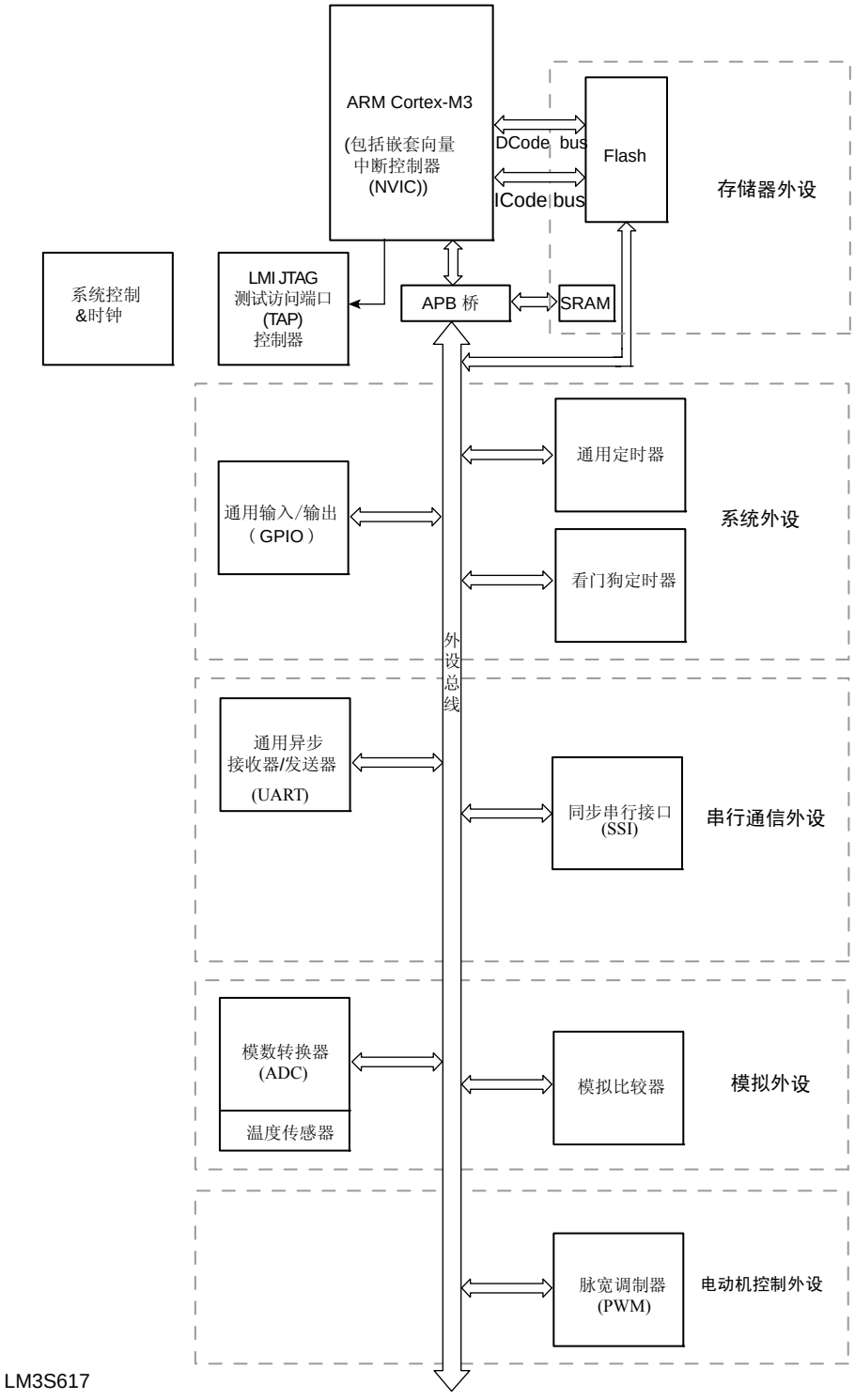


图 1.1 Stellaris®高级方框图

1.4 功能概述

以下小节概述了 LM3S617 微控制器的特性。圆括号中标注的章节编号表示该特性将在该章节中详细描述。订购和支持信息可在“订购和联系信息”中找到。

1.4.1 ARM Cortex™-M3

1.4.1.1 处理器内核（见 第 2 章）

Stellaris 系列产品的所有成员(包括 LM3S617 微控制器)都是围绕着 ARM Cotex™-M3 处理器内核来设计的。ARM Cortex-M3 处理器为高性能、低成本的平台提供了一个能够满足小存储要求解决方案（minimal memory implementation）、简化管脚数、以及低功耗三方面要求的内核，与此同时，它还提供了出色的计算性能和优越的系统中断响应能力。

本文的 第 2 章 “ARM Cortex-M3 处理器内核”将对ARM内核进行概述；“ARM® Cortex™-M3 技术参考手册”对该内核进行了详细介绍。

1.4.1.2 嵌套向量中断控制器（NVIC）

LM3S617 控制器包含 ARM Cortex-M3 内核上的 ARM 嵌套向量中断控制器（NVIC）。NVIC 和 Cortex-M3 将区分所有异常的优先等级并对其进行处理。所有异常都是在处理器模式中处理的。在出现异常时，处理器状态会自动存储到堆栈，并且在中断服务程序(ISR)结束时自动从堆栈中恢复。取出向量和保存状态是同时进行的，这样便可以高效的进入中断。处理器还支持末尾连锁（tail-chaining），使得处理器无需保存和恢复状态便可执行两个连续的背对背(back-to-back)中断。软件可对 7 个异常（系统处理程序）和 26 个中断设置 8 个优先级。

第 4 章 “中断”提供NVIC控制器和中断映射的概述。异常和中断在“ARM® Cortex™-M3 技术参考手册”中详述。

1.4.2 电机控制外设

为了增强电机控制，LM3S617 控制器包含脉宽调制（PWM）输出。

1.4.2.1 PWM（“16 位 PWM 模式”）

脉宽调制（PWM）是一种对模拟信号电平进行数字编码的强大技术。使用高分辨率计数器来产生方波，并且通过调整方波的占空比来编码模拟信号。典型的应用包括开关电源（switching power）和电机控制。

在 LM3S617 上，PWM 运动控制功能可通过专门的灵活的运动控制硬件（PWM 管脚）或通用定时器的运动控制特性（使用 CCP 管脚）来实现。

PWM 管脚（见 第 15 章）

LM3S617 的 PWM 模块由 3 个 PWM 发生器模块和一个控制模块组成。每个 PWM 发生器模块包含 1 个定时器（16 位递减或递增/递减计数器）、2 个比较器、1 个 PWM 信号发生器、1 个死区发生器，以及一个中断/ADC-触发选择器。控制模块决定了 PWM 信号的极性，以及将哪个信号传递给管脚。

每个 PWM 发生器模块产生 2 个 PWM 信号，它们可以是独立的信号，也可以是一对插入了死区延迟的互补（complementary）信号。PWM 发生器模块的输出在传递到器件管脚之前由输出控制模块进行管理。

CCP管脚（“16 位PWM模式”）

通用定时器模块的 CCP（捕获比较 PWM）管脚可以通过软件进行编程，以支持简单的 PWM 模式。在该模式下，PWM 信号的输出反相可以通过软件进行编程。

1.4.3 模拟外设

为了处理模拟信号，LM3S617 控制器提供 1 个模数转换器（ADC）和 1 个模拟比较器。

1.4.3.1 ADC（见 第 11 章）

模数转换器（ADC）外设用于将连续的模拟电压转换成离散的数字量。

群星（Stellaris）ADC 模块的转换分辨率为 10 位，含 6 个输入通道和 1 个内部温度传感器。4 个带缓冲的采样序列无需使用控制器，就可以对多达 6 个模拟输入源进行快速采样。每个采样序列均可以灵活的编程，其输入源、触发事件、中断的发生和序列优先级都是可配置的。

1.4.3.2 模拟比较器（见 第 14 章）

模拟比较器外设比较两个模拟电压，并提供一个逻辑输出来表示比较结果。

LM3S617 控制器提供 1 个独立的集成模拟比较器，模拟比较器可以被配置成驱动输出或产生中断/ADC 事件。

比较器可将测试电压与下面的其中一种电压相比较：

- 单个的外部参考电压
- 一个共用的外部参考电压
- 一个共用的内部参考电压

比较器可以向器件管脚提供输出信号，以代替板上的模拟比较器；比较器还可以通过中断或 ADC 触发信号通知应用程序，让它开始捕获一个采样序列。产生中断和 ADC 触发的逻辑是独立的。即，例如中断可以在上升沿产生，而 ADC 可以在下降沿触发。

1.4.4 串行通信外设

LM3S617 控制器支持异步和同步的串行通信，它含有 2 个完全可编程的 16C550-型 UART 和 SSI 串行通信。

1.4.4.1 UART（见 第 12 章）

通用异步接收器/发送器（UART）是一个用于 RS-232C 串行通信的集成电路，它带有一个发送器（并行到串行转换器）和一个接收器（串行到并行转换器），每个都是独立计时。

LM3S617 控制器包含 2 个完全可编程的 16C550-型 UART，它所支持的数据传输速率高达 460.8Kbps。（虽然 16C550-型 UART 与 16C550 UART 的功能类似，但寄存器不相互兼容）。

独立的 16×8 发送(Tx)和 16×12 接收(Rx) FIFO 减少 CPU 中断服务负载。UART 通过 RX、TX、modem 状态和错误状态可单独产生屏蔽中断。在任意中断发出且均未被屏蔽时，模块产生的是一个组合（single combined）中断。

1.4.4.2 SSI（见 第 13 章）

同步串行接口（SSI）是一个 4 线双向的通信接口。

群星（Stellaris）SSI 模块为同步串行通信功能提供外设器件，并可配置成使用 Freescale SPI、MICROWIRE 或 TI 同步串行接口帧格式。数据帧的大小也可配置，可以设置成 4 到 16 位，包括 4 位和 16 位。

从外设器件中接收到数据时 SSI 模块执行串行到并行转换，在数据发送到外设器件时 SSI 模块执行并行到串行转换。TX 和 RX 是通过内部 FIFO 来进行缓冲的，该 FIFO 最多可以单独存储 8 个 16 位的值。

SSI 模块可被配置为主机或从机设备。作为从机设备时，还可以通过配置把 SSI 模块的输出禁能，从而使主机可以与多个从机设备相连。

SSI 模块还包含一个可编程的位速率时钟分频器和预分频器来生成串行输出时钟。位速率是根据输入时钟来产生的，而所连接的外设则决定了位速率的最大值。

1.4.5 系统外设

1.4.5.1 可编程的GPIO（见 第 8 章）

通用输入输出(GPIO)管脚为各种连接方式带来了灵活性。

群星（Stellaris）GPIO模块由 5 个物理GPIO模块组成，每个对应一个独立的GPIO端口。GPIO模块遵循FiRM规范(遵循ARM实时微控制器底层IP规范)并支持 1 到 30 个可编程的输入/输出管脚。可用的GPIO数视正在使用的外设而定（有关每个GPIO脚可用的信号见表 17.4）。

GPIO 模块的特点是所有管脚都可以被编程为以边沿或者电平触发的方式来产生中断；可以通过编程来控制 GPIO 端口的配置；还可以在读写操作的时候通过地址线进行位屏蔽。

1.4.5.2 3 个可编程的定时器（见 第 9 章）

可编程的定时器可以对驱动定时器输入管脚的外部事件进行计数或者定时。

群星（Stellaris）通用定时器模块（GPTM）包含 3 个 GPTM 模块。每个 GPTM 模块都提供了 2 个 16 位的定时器/计数器，可将它们配置为两个独立运作的定时器/事件计数器，还可以将它们配置成 1 个 32 位的定时器或 1 个 32 的位实时时钟（RTC）。此外，定时器还可以用来触发模数转换（ADC）。

当被配置为 32 位模式的时候，定时器可作为单次触发定时器、周期定时器或实时时钟(RTC)运行。当被配置为 16 位模式的时候，定时器可作为单次触发定时器或周期定时器运行，并可通过使用一个 8 位预分频器来扩展其精度。也可以将其配置成 16 位定时器，以便于捕获事件或产生脉宽调制(PWM)信号。

1.4.5.3 看门狗定时器（见 第 10 章）

看门狗定时器在到达超时值（time-out）时可能会产生不可屏蔽中断(NMI)或复位。当系统由于软件错误而无法响应或外部器件不是以期望的方式响应时，使用看门狗定时器可重新获得控制权。

群星（Stellaris）看门狗定时器模块包括一个 32 位的递减（down）计数器、一个可编程的装载寄存器、中断产生逻辑和一个锁定寄存器。

看门狗定时器可以配置为在首次超时(time-out)时产生中断，并在再次超时的时候产生复位信号。一旦配置了看门狗定时器，就可以写锁定寄存器来防止定时器配置被意外更改。

1.4.6 存储器外设

群星（Stellaris）控制器提供 SRAM 和 Flash 存储器。

1.4.6.1 SRAM（见 7.2.1）

LM3S617 静态随机存取存储器（SRAM）控制器支持 8KB SRAM。群星（Stellaris）器件的内部 SRAM 位于地址 0x2000 0000 的器件存储器映射中。为了减少执行读-修改-写(RWM)操作的时间，ARM 在新的 Cortex-M3 处理器中引入了 bit-banding 技术。在使能 bit-banding 的处理器中，在进行单次或原子（atomic）操作时，存储器映射中的特定区域（SRAM 和外设空间）可以使用地址别名来访问各个位。

1.4.6.2 Flash（见 7.2.2）

LM3S617 Flash 控制器支持 32KB 的 Flash 存储器。Flash 是由一组可独立擦除的 1KB 区块所构成的。对一个区块进行擦除将使该区块的全部内容复位为 1。这些模块配对后便组成了一组可分别进行保护的 2KB 区块。区块可被标记为只读或只执行(execute-only)，以提供不同级别的代码保护。只读区块不能进行擦除或者编程，以保护区块的内容免受更改。只执行区块不能进行擦除或者编程，而且只能通过控制器取指的机制来读取它的内容，这可以保护区块的内容，使其不被控制器或调试器读取。

1.4.7 其它特性

1.4.7.1 存储器映射（见 第 3 章）

存储器映射列出了指令和数据在存储器中的位置。LM3S617 控制器的存储器映射可在“表 3.1”中找到。寄存器地址将采用十六进制递增的形式给出，并与存储器映射中模块的基址相对应。

有关存储器映射的详细信息请见“ARM® Cortex™-M3 技术参考手册”。

1.4.7.2 JTAG TAP控制器（见 第 5 章）

联合测试行动组（JTAG）端口提供了一个标准的串行接口来控制测试访问端口(TAP)和相关的测试逻辑。TAP、JTAG 指令寄存器和 JTAG 数据寄存器可用于测试互连的集成印刷电路板，获取组件的制造信息，并且在正常操作中观察和/或控制控制器的输入和输出。JTAG 端口以低的成本提供高级的可测试性和芯片级访问。

JTAG 端口由 5 个标准管脚组成：$\overline{\text{TRST}}$、TCK、TMS、TDI 和 TDO。数据通过 TDI 管脚被串行的发送到控制器，再由控制器通过 TDO 管脚输出。该数据的解析（interpretation）视 TAP 控制器的当前状态而定。有关 JTAG 端口和 TAP 控制器操作的详细信息，请参考“IEEE 标准 1149.1－测试访问端口和边界扫描结构”。

LMI JTAG 控制器是与内置到 Cortex-M3 内核的 ARM JTAG 控制器一起工作的。这可以通过多路复用这两个 JTAG 控制器的 TDO 输出管脚来实现。ARM 的 JTAG 指令将选择 ARM 的 TDO 输出，而 LMI 的 JTAG 指令将选择 LMI 的 TDO 输出。而复用器将由 LMI JTAG 控制器来控制，它可以对 ARM、LMI 和未执行的 JTAG 指令进行综合编程。

1.4.7.3 系统控制和时钟（见 第 6 章）

系统控制决定了器件的整体运作。它不但提供了有关器件的信息，对器件和单个外设的时钟进行控制，还负责处理复位的检测和报告。

1.4.8 硬件详述

有关管脚和封装的详细内容可在下面几章中找到：

- 第 16 章“管脚图”
- 第 17 章“信号表”
- 第 18 章“工作特性”
- 第 19 章“电气特性”
- 第 20 章“封装信息”

1.5 系统方框图

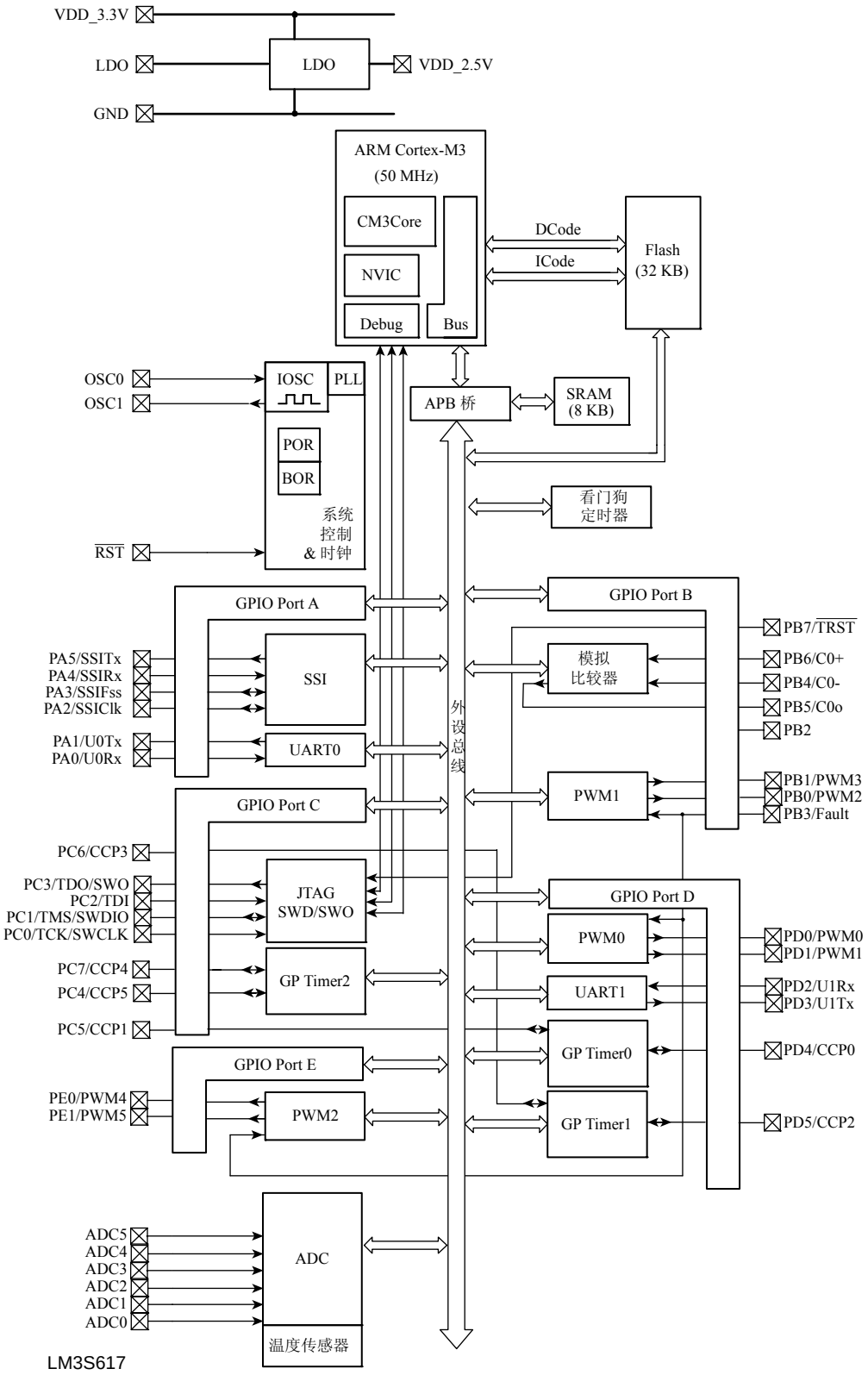


图 1.2 LM3S617 控制器系统级方框图

第2章 ARM Cortex-M3 处理器内核

ARM Cortex-M3 处理器为高性能、低成本的平台提供了一个能够满足小存储要求解决方案（minimal memory implementation）、简化管脚数、以及低功耗三方面要求的内核，与此同时，它还提供了出色的计算性能和优越的系统中断响应能力。其特性如下：

- 紧凑的内核
- Thumb-2 指令集，在与 8 位和 16 位器件相关的存储容量中，特别是在微控制器级应用的几千字节存储量中，提供了 ARM 内核所期望的高性能。
- 优越的中断处理能力，通过执行寄存器操作来实现，这些寄存器操作在处理硬件中断时使用。
- 存储器保护单元（MPU），为复杂的应用提供特权操作模式
- 功能齐全的调试解决方案，包括：
  - 串行线 JTAG 调试端口（SWJ-DP）
  - Flash 修补和断点（FPB）单元，用于实现断点操作
  - 数据观察点和触发单元（DWT），用于执行观察点、触发源和系统性能分析等操作。
  - 仪表跟踪宏单元（ITM），用于支持 Printf 型调试
  - 跟踪端口的接口单元（TPIU），用作跟踪端口分析仪（TPA）的桥接

群星（stellaris）系列微控制器基于 Cortex-M3 内核，为注重成本的嵌入式微控制器应用，如工厂自动化与控制、工业控制电源设备以及楼宇和家庭自动化提供了高性能的 32 位运算能力。

要详细了解 ARM Cortex-M3 处理器内核，请参考“ARM® Cortex™-M3 技术参考手册”。要了解 SWJ-DP 的信息，请参考“CoreSight™设计套件技术参考指南”。

2.1 结构图

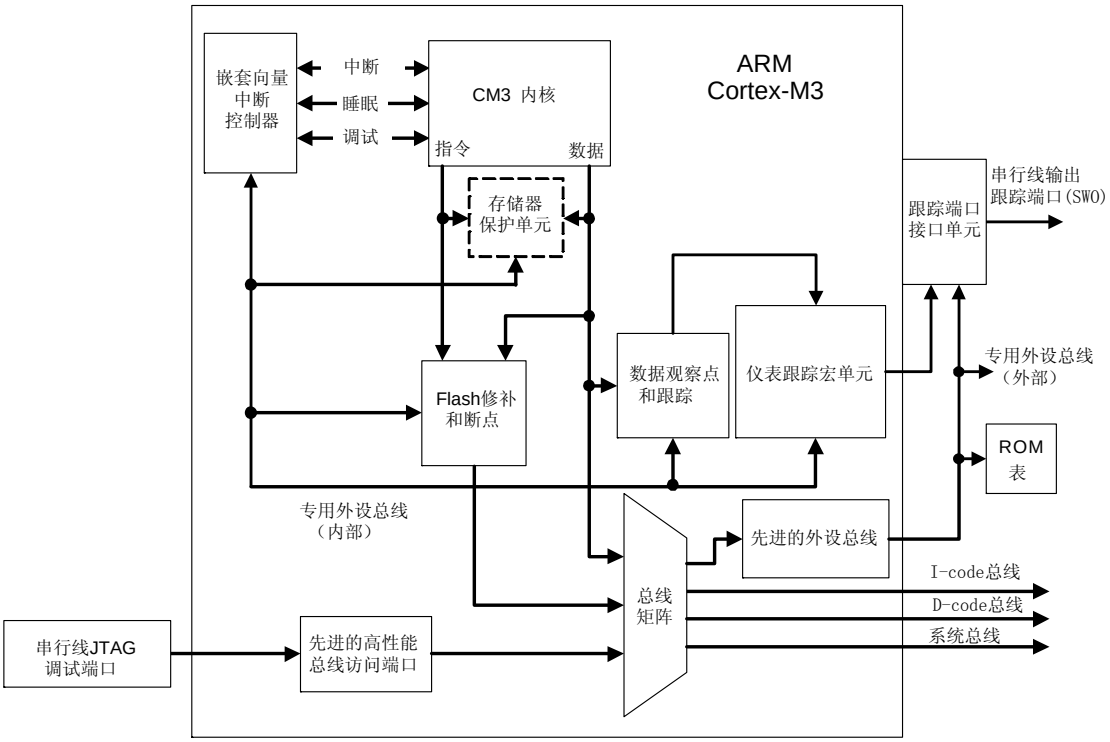


图 2.1 CPU 结构图

2.2 功能描述

要点：“ARM® Cortex™-M3 技术参考手册”详细描述了 ARM Cortex-M3 的全部特性。但是，这些特性根据具体的实现的不同而不同。本节主要描述了群星（stellaris）系列的实现方案。

Luminary Micro已经使用了如 图 2.1 所示的ARM Cortex-M3 内核。正如“ARM® Cortex™-M3 技术参考手册”所说，在它们的实现方案中，SW/JTAG-DP、ETM、TPIU、ROM表、MPU和嵌套向量中断控制器（NVIC）这几个Cortex-M3 组件是灵活可变的。以下小节将会对这各个组件进行详细介绍。

2.2.1 串行线和 JTAG 调试

Luminary Micro 已经使用了与 ARM CoreSight™兼容的串行线 JTAG 调试端口（SWJ-DP）接口来替代 ARM SW-DP 和 JTAG-DP。这就意味着“ARM® Cortex™-M3 技术参考手册”中的第十二章“调试端口”并不适用于群星（stellaris）器件。

SWJ-DP 接口将 SWD 和 JTAG 调试端口集成到一个模块。详见“CoreSight™设计套件技术参考指南”中对 SWJ-DP 的描述。

2.2.2 嵌入式跟踪宏单元（ETM）

群星（stellaris）器件中没有使用 ETM。这就意味着“ARM® Cortex™-M3 技术参考手册”中的第 15 章和 16 章的内容都可忽略。

2.2.3 跟踪端口的接口单元（TPIU）

TPIU充当来自ITM的Cortex-M3 跟踪数据和片外跟踪端口分析仪之间的桥接器。群星（stellaris）器件已经使用了如 图 2.2 所示的TPIU。这与“ARM® Cortex™-M3 技术参考手册”中描述的无ETM版本类似，但是SWJ-DP仅为TPIU提供SWV输出。

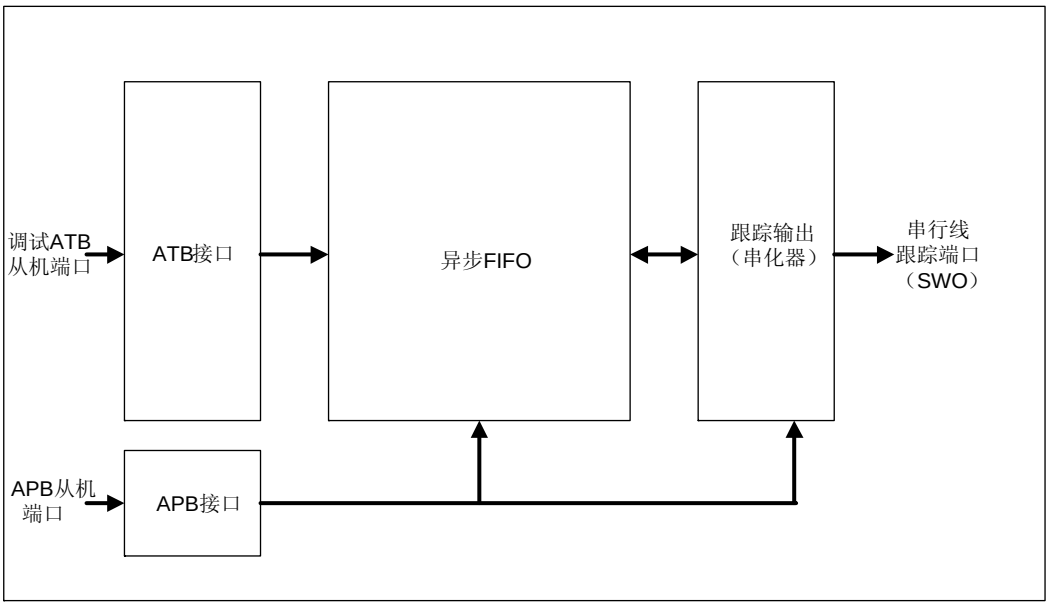


图 2.2 TPIU 方框图

2.2.4 ROM 表

默认情况下使用的是“ARM® Cortex™-M3 技术参考手册”中介绍的 ROM 表。

2.2.5 存储器保护单元（MPU）

LM3S617 控制器包含存储器保护单元（MPU），MPU 支持标准的 ARMv7 受保护的存储器系统架构（PMSA）模型。MPU 全力支持保护区域、重叠保护区域、访问许可，以及将存储器属性输出给系统。

2.2.6 嵌套向量中断控制器（NVIC）

2.2.6.1 中断

“ARM® Cortex™-M3 技术参考手册”描述了中断和中断优先级的最大数量。LM3S617 微控制器支持 26 种中断，8 个优先级。

2.2.6.2 SysTick 校验值寄存器

不使用 SysTick 校验值寄存器。

第3章 存储器映射

LM3S617 的存储器映射如 表 3.1 所示。在本手册中，寄存器地址将采用十六进制递增的形式给出，并与存储器映射表中模块的基址相对应。存储器映射的相关内容也可以参考“ARM Cortex-M3 技术参考手册”的第 4 章“存储器映射”。

表 3.1 存储器映射

起始地址	结束地址	描述
存储器		
0x00000000	0x00007FFF	片内 Flash
0x00008000	0x1FFFFFFF	保留 [a]
0x20000000	0x20001FFF	片内带 bit-banding 特性的 SRAM
0x20002000	0x200FFFFF	保留 [a]
0x22000000	0x2203FFFF	0x20000000～0x20001FFF 的 bit-band 别名
0x22040000	0x23FFFFFF	保留 [a]
FiRM 外设		
0x40000000	0x40000FFF	看门狗定时器
0x40001000	0x40003FFF	留给其它 3 个看门狗定时器（遵循 FiRM 规范）
0x40004000	0x40004FFF	GPIO 端口 A
0x40005000	0x40005FFF	GPIO 端口 B
0x40006000	0x40006FFF	GPIO 端口 C
0x40007000	0x40007FFF	GPIO 端口 D
0x40008000	0x40008FFF	SSI
0x40009000	0x4000BFFF	留给其它 3 个 SSI（遵循 FiRM 规范）[a]
0x4000C000	0x4000CFFF	UART0
0x4000D000	0x4000DFFF	UART1
0x4000E000	0x4000FFFF	留给其它 2 个 UART（遵循 FiRM 规范）[a]
0x40010000	0x4001FFFF	留给将来的 FiRM 外设 [a]
外设		
0x40020000	0x40023FFF	保留 [a]
0x40024000	0x40024FFF	GPIO 端口 E
0x40025000	0x40027FFF	保留 [a]
0x40028000	0x40028FFF	PWM
0x40029000	0x4002BFFF	保留 [a]
0x4002C000	0x4002FFFF	保留 [a]
0x40030000	0x40030FFF	定时器 0
0x40031000	0x40031FFF	定时器 1
0x40032000	0x40032FFF	定时器 2
0x40033000	0x40037FFF	保留 [a]
0x40038000	0x40038FFF	ADC
0x40039000	0x4003BFFF	保留 [a]
0x4003C000	0x4003CFFF	模拟比较器
0x4003D000	0x400FCFFF	保留 [a]
0x400FD000	0x400FDFFF	Flash 控制
0x400FE000	0x400FFFFF	系统控制
0x40100000	0x41FFFFFF	保留 [a]
0x42000000	0x43FFFFFF	0x40000000～0x400FFFFF 的 bit-band 别名
0x44000000	0xDFFFFFFF	保留 [a]

续上表

起始地址	结束地址	描述
专用外设总线		
0xE0000000	0xE0000FFF	仪表跟踪宏单元（ITM）
0xE0001000	0xE0001FFF	数据观察点和跟踪（DWT）
0xE0002000	0xE0002FFF	Flash 修补和断点（FPB）
0xE0003000	0xE000DFFF	保留 [a]
0xE000E000	0xE000EFFF	嵌套向量中断控制器（NVIC）
0xE000F000	0xE003FFFF	保留 [a]
0xE0040000	0xE0040FFF	跟踪端口的接口单元（TPIU）
0xE0041000	0xE0041FFF	保留 [a]
0xE0042000	0xE00FFFFF	保留 [a]
0xE0100000	0xFFFFFFFF	留给厂商外设 [a]

a 对所有保留空间执行读写操作时都将返回总线故障。

第4章 中断

ARM Cortex-M3 处理器和嵌套向量中断控制器(NVIC)将区分所有异常的优先等级并对其进行处理。所有异常将于处理器模式中处理。在出现异常时，处理器的状态将被自动存储到堆栈中，并在中断服务程序（ISR）结束时自动从堆栈中恢复。取出向量和保存状态是同时进行的，这样便提高了进入中断的效率。处理器还支持末尾连锁（tail-chaining），使得处理器无需保存和恢复状态便可执行两个连续的背对背（back-to-back）中断。

表 4.1 列出了所有的异常。软件可在 7 个异常（系统处理程序）以及 26 个中断上设置 8 个优先级（在 表 4.2 中列出）。系统处理程序的优先级是通过NVIC系统处理程序优先级寄存器来设置的。而中断则是通过NVIC中断设置使能寄存器来使能的，并且由NVIC中断优先级寄存器来区分其优先等级。你还可以把优先级划分为占先优先级（Pre-emption priorities）和次要优先级(subpriorites)两组。所有的中断寄存器在“ARM® Cortex™-M3 技术参考手册”的第 8 章“嵌套向量中断控制器”中描述。

用户可设置的最高优先级(0)是仅次于复位，NMI 以及硬件故障的第四优先级。**注意**：0 是所有可调整优先级的默认优先级。

如果你将两个或更多的中断指定为相同的优先级，那么它们的硬件优先级（位置编号越高优先级越低）就决定了处理器激活这些中断的顺序。例如，如果 GPIO 端口 A 和 GPIO 端口 B 都为优先级 1，那么 GPIO 端口 A 优先级更高。

有关异常和中断的更多信息请见“ARM® Cortex™-M3 技术参考手册”的第 5 章“异常”和第 8 章“嵌套向量中断控制器”。

表 4.1 异常类型

异常类型	位置	优先级 [a]	描述
-	0	-	复位时载入向量表的第一项作为栈顶地址。
复位	1	-3（最高）	在上电和热复位时调用。在执行第一条指令时，优先级将降为最低（也就是所谓的激活（中断）的基础级别）。这是异步的。
不可屏蔽中断（NMI）	2	-2	不可停止，也不会被复位之外的任何异常抢占。这是异步的。 NMI 仅可由软件通过 NVIC 中断控制状态寄存器来产生。
硬故障	3	-1	当故障由于优先级或者是可配置的故障处理程序被禁能的原因而无法激活时，所有类型的故障都会以硬故障的方式激活。这是同步的。
存储器管理	4	可调整	MPU 不匹配，包括访问冲突(access violation)和不匹配。这是同步的。这种异常的优先级可被改变。
总线故障	5	可调整	预取指故障、存储器访问故障和其它地址/存储器相关的故障。当为精确的总线故障时是同步的，为不精确的总线故障时是异步的。 你可以使能或禁能这种故障。
使用故障	6	可调整	使用故障，例如执行未定义的指令或试图进行非法的状态转变。这是同步的。
-	7-10	-	保留。
SVCall	11	可调整	使用 SVC 指令的系统服务调用。这是同步的。

续上表

异常类型	位置	优先级 [a]	描述
调试监控器	12	可调整	调试监控器（当没有暂停(Halt)时）。这是同步的，但仅在使能时有效。如果它的优先级比当前激活的处理程序的优先级更低，那么调试监控器不能激活。
-	13	-	保留。
PendSV	14	可调整	系统服务的可挂起(pendable)请求。这是异步的且仅通过软件挂起。
SysTick	15	可调整	系统节拍定时器已启动(fired)。这是异步的。
Interrupts	≥16	可调整	中断在ARM Cortex-M3 内核之外发出且通过NVIC返回（区分优先级）。这些都是异步的。表 4.2 列出了LM3S617 控制器上的中断。

a. 0 是所有可调整优先级的默认优先级。

表 4.2 中断

中断（在中断寄存器中的位）	描述
0	GPIO 端口 A
1	GPIO 端口 B
2	GPIO 端口 C
3	GPIO 端口 D
4	GPIO 端口 E
5	UART0
6	UART1
7	SSI
8	保留
9	PWM 故障
10	PWM 发生器 0
11	PWM 发生器 1
12	PWM 发生器 2
13	保留
14	ADC 序列 0
15	ADC 序列 1
16	ADC 序列 2
17	ADC 序列 3
18	看门狗定时器
19	定时器 0 a
20	定时器 0 b
21	定时器 1 a
22	定时器 1 b
23	定时器 2 a
24	定时器 2 b
25	模拟比较器 0
26 - 27	保留
28	系统控制

续上表

中断（在中断寄存器中的位）	描述
29	Flash 控制
30 – 31	保留

第5章 JTAG 接口

联合测试行动组（JTAG）端口是一个 IEEE 标准，它定义了数字集成电路的测试访问端口和边界扫描架构，并为控制相关的测试逻辑提供了一个标准的串行接口。TAP、指令寄存器（IR）和数据寄存器（DR）可以用来测试集成印刷电路板的互连，以及获取组件的制造信息。JTAG 端口还提供了一种访问和控制“可测试性设计（design-for-test）”特性的方法，如观察与控制 I/O 管脚、扫描测试和调试。

JTAG 端口由 5 个标准的管脚组成：$\overline{\text{TRST}}$、TCK、TMS、TDI 和 TDO。数据通过 TDI 串行发送至控制器，然后通过 TDO 管脚从控制器串行输出。该数据的解析视 TAP 控制器的当前状态而定。要详细了解 JTAG 端口和 TAP 控制器的操作，请参考“IEEE 标准 1149.1——测试访问端口和边界扫描架构”。

LMI JTAG 控制器是与植入 Cortex-M3 内核内的 ARM JTAG 控制器一起工作的。这可以通过多路复用这两个 JTAG 控制器的 TDO 输出管脚来实现。ARM JTAG 指令选择 ARM TDO 输出管脚，而 LMI JTAG 指令则选择 LMI TDO 输出管脚。多路复用器将由 LMI JTAG 控制器控制，它可以对 ARM、LMI 和未执行的 JTAG 指令进行综合编程。

JTAG 模块含以下特性：

- IEEE 1149.1-1990 兼容的测试访问端口（TAP）控制器
- 4 位指令寄存器（IR）链，用于存储 JTAG 指令
- IEEE 标准指令：
  - BYPASS 指令
  - IDCODE 指令
  - SAMPLE/PRELOAD 指令
  - EXTEST 指令
  - INTEST 指令
- ARM 附加指令：
  - APACC 指令
  - DPACC 指令
  - ABORT 指令
- 集成的 ARM 串行线调试（SWD）

要详细了解有关 ARM JTAG 控制器的信息，请参考“ARM® Cortex™技术参考手册”。

5.1 结构图

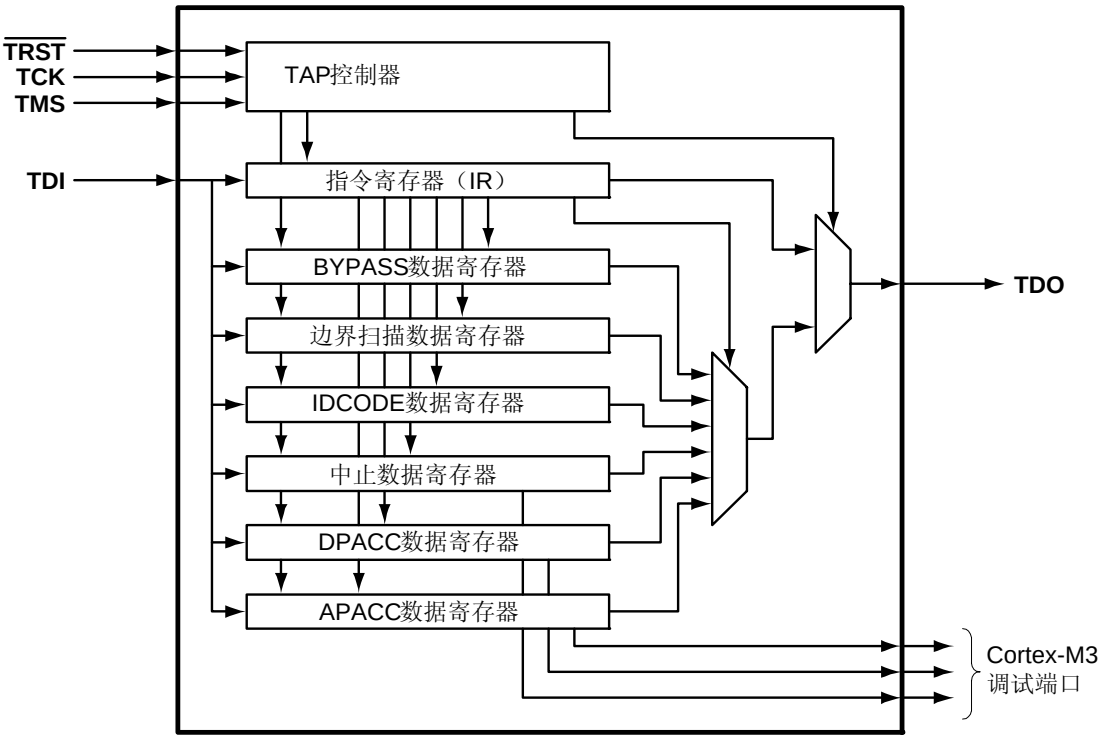


图 5.1 JTAG 模块结构图

5.2 功能描述

JTAG模块的高级概念图如 图 5.1 所示。JTAG模块由测试访问端口（TAP）控制器和带有并行更新寄存器的串行移位链组成。TAP控制器是一个简单的状态机，它由 $\overline{\text{TRST}}$ 、TCK和TMS输入管脚控制。TAP控制器的当前状态取决于 $\overline{\text{TRST}}$ 的当前值以及TMS管脚在TCK信号上升沿所捕获的值的序列。TAP控制器决定了串行移位链何时捕获新数据，何时将数据从TDI移向TDO，以及何时更新并行加载寄存器。TAP控制器的当前状态还决定了正在访问的是指令寄存器（IR）链还是其中一个数据寄存器（DR）链。

带有并行加载寄存器的串行移位链是由一个指令寄存器（IR）链和多个数据寄存器（DR）链组成的。当前被加载到并行加载寄存器的指令决定了在 TAP 控制器排序过程中，哪一个 DR 链将被捕获、移位或更新。

某些指令，象EXTEST和INTEST，会对当前位于DR链中的数据进行操作，但不会捕获、移动或更新任何链。为了确保TDI和TDO之间的串行通道一直连接，未被执行的指令将会译码成BYPASS指令（要得到已执行指令的一览表，请参考 表 5.2）。

要了解JTAG的时序图，请参考“JTAG和边界扫描”。

5.2.1 JTAG 接口的管脚

JTAG接口由 5 个标准的管脚： $\overline{\text{TRST}}$ 、TCK、TMS、TDI和TDO组成。这些管脚以及与其相关联的复位状态如 表 5.1 所示。下面接着详细介绍各个管脚。

表 5.1 JTAG 端口管脚复位状态

管脚名称	数据方向	内部上拉	内部下拉	驱动强度	驱动值
$\overline{\text{TRST}}$	输入	使能	禁能	N/A	N/A
TCK	输入	使能	禁能	N/A	N/A
TMS	输入	使能	禁能	N/A	N/A
TDI	输入	使能	禁能	N/A	N/A
TDO	输出	使能	禁能	2-mA 驱动	高阻抗

5.2.1.1 测试复位输入（$\overline{\text{TRST}}$）

$\overline{\text{TRST}}$ 管脚是一个低电平有效的异步输入信号，可以用来对 JTAG TAP 控制器和相关的 JTAG 电路进行初始化和复位。**$\overline{\text{TRST}}$** 生效时，TAP 控制器复位到 Test-Logic-Reset 状态，并且在 **$\overline{\text{TRST}}$** 生效期间一直保持这种状态。TAP 控制器进入 Test-Logic-Reset 状态时，JTAG 指令寄存器（IR）复位为默认的 IDCODE 指令。

默认情况下，**$\overline{\text{TRST}}$** 管脚上的内部上拉电阻在复位后使能。在修改 GPIO 端口 B（PB 口）的上拉电阻的设置时应该确保 PB7/**$\overline{\text{TRST}}$** 上的内部上拉电阻保持使能，否则可能会丢失 JTAG 通信。

5.2.1.2 测试时钟输入（TCK）

TCK 管脚是 JTAG 模块的时钟输入。通过该时钟输入，测试逻辑可以独立于其他系统时钟而单独运行。此外，它确保多个组成链环（daisy-chain）的 JTAG TAP 控制器可以在组件之间同步传送串行测试数据。正常工作期间，TCK 由自由振荡的时钟（额定占空比为 50%）驱动。必要时，可以将 TCK 保持为 0 或 1，并持续多个周期。当 TCK 保持为 0 或 1 时，TAP 控制器的状态不发生改变，且 JTAG 指令和数据寄存器中的数据不会丢失。

默认情况下，TCK 管脚上的内部上拉电阻在复位后使能。因而确保在管脚没有被外部源驱动时不会进行计时。只要 TCK 管脚连续被外部源驱动，我们就可以通过关闭内部上拉和下拉电阻来节省内部功耗。

5.2.1.3 测试模式选择（TMS）

TMS 管脚选择 JTAG TAP 控制器的下一个状态。TMS 在 TCK 的上升沿被采样。根据当前的 TAP 状态以及 TMS 的采样值进入下一个状态。因为 TMS 管脚在 TCK 的上升沿被采样，所以 IEEE 标准 1149.1 希望 TMS 的值在 TCK 的下降沿发生变化。

将 TMS 设为高电平并维持 5 个连续的 TCK 周期将驱使 TAP 控制器状态机进入 Test-Logic-Reset(测试逻辑复位)状态。TAP控制器进入Test-Logic-Reset状态时，JTAG指令寄存器（IR）复位到默认的指令IDCODE。因此，可将该时序当成一种复位机制来使用，

其效果与将 $\overline{\text{TRST}}$ 信号变有效相似。通过 图 5.2，我们可以完整地了解JTAG测试访问端口的状态机。

默认情况下，TMS 管脚上的内部上拉电阻在复位后使能。在改变 GPIO 端口 C（PC 口）的上拉电阻的设置时，应该确保 PC1/TMS 上的内部上拉电阻保持使能；否则可能会丢失 JTAG 通信。

5.2.1.4 测试数据输入（TDI）

TDI 管脚将一串串行信息传送给 IR 链和 DR 链。TDI 在 TCK 的上升沿被采样，并根据当前的 TAP 状态以及当前指令，将该数据传送给合适的移位寄存器链。因为 TDI 管脚在 TCK 的上升沿被采样，所以 IEEE 标准 1149.1 期望 TDI 的值在 TCK 的下降沿发生变化。

默认情况下，TDI 管脚上的内部上拉电阻在复位后使能。在改变 GPIO 端口 C（PC 口）的上拉电阻的设置时应该确保 PC2/TDI 上的内部上拉电阻保持使能；否则可能会丢失 JTAG 通信。

5.2.1.5 测试数据输出（TDO）

TDO 管脚将一串串行信息从 IR 链或 DR 链输出。TDO 的值取决于当前的 TAP 状态、当前指令、以及正在访问的链。为了在不使用 JTAG 端口时节省功耗，若当前并无移出数据的活动，TDO 管脚将被置为未激活的驱动状态。因为在链环配置中，TDO 可以与另一个控制器的 TDI 管脚相连。所以 IEEE 标准 1149.1 期望 TDO 的值在 TCK 的下降沿发生变化。

默认情况下，TDO 管脚上的内部上拉电阻在复位后使能。这样便确保了在不使用 JTAG 端口时，该管脚的逻辑电平能够保持恒定。如果高阻输出值在某些 TAP 控制状态过程中是满足要求的，那么可以通过关闭内部上拉和内部下拉电阻来节省内部功耗。

5.2.2 JTAG TAP 控制器

JTAG TAP控制器状态机如 图 5.2 所示。TAP控制器状态机在上电复位（POR）或 $\overline{\textbf{TRST}}$ 信号生效时复位到Test-Logic-Reset状态。在正确的时序信号被发送到TMS管脚时，JTAG模块可以移入新指令或移入数据，或者在扩展的测试序列期间变空闲。要详细了解有关TAP控制器功能的信息和发生在每个状态的操作，请参考“IEEE标准 1149.1”。

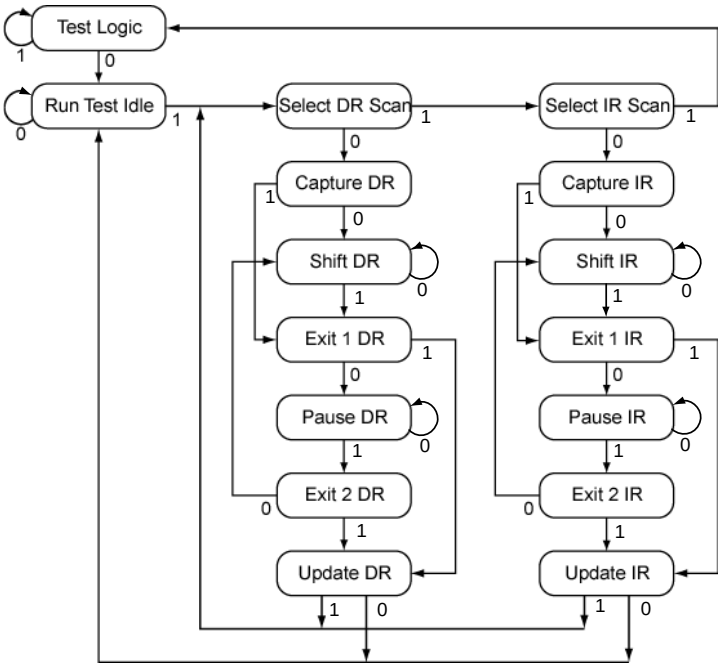


图 5.2 测试访问端口状态机

5.2.3 移位寄存器

移位寄存器由一个串行移位寄存器链和一个并行加载寄存器组成。串行移位寄存器链在TAP控制器的CAPTURE（捕获）状态下对特定的信息进行采样。并允许在TAP控制器的SHIFT（移位）状态下将这些信息从TDO管脚移出。在已采样的数据通过TDO管脚移出链的同时，新的采样数据也正在通过TDI管脚移入串行移位寄存器。在TAP控制器的UPDATE状态期间，这些新的数据将被存放到并行加载寄存器中。“移位寄存器”小节将对每个移位寄存器进行详细的介绍。

5.2.4 操作时的注意事项（operational consideration）

在使用 JTAG 模块时有某些特定的操作方法。因为 JTAG 管脚可被编程为 GPIO，所以必须考虑这些管脚的板卡配置（board configuration）和复位条件。此外，由于 JTAG 模块已经集成了 ARM 串行线调试，所以在两个操作模式之间进行切换时，必须对切换方式进行说明。

5.2.4.1 GPIO 功能

当控制器通过 **POR** 或 **$\overline{\text{RST}}$** 复位时，JTAG 端口管脚将被设为它们默认的 JTAG 配置。默认的配置包括使能上拉电阻（将 GPIOPUR 中与 PB7 和 PC[3:0]对应的位设为 1）和使能 JTAG 管脚上交替的硬件功能（将 GPIOAFSEL 中与 PB7 和 PC[3:0]对应的位设为 1）。

复位后，软件只需向 GPIOAFSEL 寄存器中与 PB7 和 PC[3:0]对应的位写入 0 便可以将这 5 个管脚配置成 GPIO。如果用户在调试或板级测试时不需要 JTAG 端口，这样做便可以腾出 5 个 GPIO 端口，供设计使用。

注：如果 JTAG 管脚在设计中用作 GPIO，那么 PB7 和 PB2 不能同时接外部下拉电阻。如果这两个管脚在复位过程都被拉至低电平，那么控制器会出现不可预测的行为。一旦这种情况发生，应移除其中一个下拉电阻，或者把两个下拉电阻都移除，并且使用 **$\overline{\text{RST}}$** 复位或关机后重新上电。

此外，可以建立一个软件程序来阻止调试器与群星系列微控制器相连。如果加载到 Flash 的程序代码立即将 JTAG 管脚变成它们的 GPIO 功能，那么在 JTAG 管脚功能切换前调试器将没有足够的时间去连接和停止控制器。这会将调试器锁在器件之外。而通过一个使用外部触发器来恢复 JTAG 功能的软件程序就可以避免这种情况发生。

5.2.4.2 ARM 串行线调试（SWD）

为了无缝的集成 ARM 串行线调试(SWD)的功能，串行线调试器必须能够与 Cortex-M3 内核相连，而无需执行或者了解 JTAG 的运行情况。这可以通过一个 SWD 前导码(preamble)来实现，这个 SWD 前导码（preamble）在 SWD 对话（session）开始前发布。

用来使能 SWJ-DP 模块的前导码在 TAP 控制器处于 Test-Logic-Reset(测试逻辑复位)状态时启用。此时，前导码将依次把 TAP 控制器设置为下列状态：Run Test Idle，Select DR，Select IR，Capture IR，Exit1 IR，Update IR，Run Test Idle，Select DR，Select IR，Capture IR，Exit1 IR，Update IR，Run Test Idle，Select DR，Select IR 和 Test-Logic-Reset 状态。

若让 TAP 状态机的 JTAG TAP 指令寄存器（IR）载入序列单步执行两次但并不移入新的指令，将使能 SWD 接口并且禁能 JTAG 接口。要了解更多有关该操作和 SWD 接口方面的信息，请参考“ARM® Cortex™技术参考手册”和“ARM® CoreSight 技术参考指南”。

因为上述序列(sequence)是一系列有效的、可发出的 JTAG 操作，所以 ARM JTAG TAP 控制器并没有完全兼容 IEEE 标准 1149.1。而这就是 ARM JTAG TAP 控制器唯一与规范不完全兼容的地方。由于在 TAP 控制器的正常操作过程中该序列（sequence）出现的可能性很低，所以它应该不会影响 JTAG 接口的正常执行。

5.3 初始化和配置

上电复位或外部复位后（$\overline{\text{RST}}$），JTAG 管脚自动配置成 JTAG 通信。不需要用户定义初始化或进行配置。但是，如果用户应用要将 JTAG 的这些管脚改为 GPIO 功能，那么这些管脚必须先被重新配置成 JTAG 功能，然后才能恢复 JTAG 通信。这可以通过使用 GPIOAFSEL 寄存器来使能 5 个 JTAG 管脚（PB7 和 PC[3:0]）的交替功能来实现。

5.4 寄存器描述

JTAG TAP 控制器或移位寄存器链中无可访问 APB 总线的寄存器。JTAG 控制器中的所有寄存器都是通过 TAP 控制器以串行方式来访问的。寄存器可以分为指令寄存器和数据寄存器两大类。

5.4.1 指令寄存器（IR）

JTAG-TAP指令寄存器（IR）是一个 4 位串行扫描链，带有一个在JTAG TDI与TDO管脚之间相连的并行加载寄存器。当TAP控制器被置为正确状态时，数据位便能够移位进入指令寄存器了。这些位一旦移入链并且被更新后，就会被解析成当前指令。有关指令寄存器的位的译码如 表 5.2 所示。下表还对每条指令进行了详细解释，并给出了与其相关的数据寄存器。

表 5.2 JTAG 指令寄存器命令

IR[3:0]	指令	描述
0000	EXTEST	将被 SAMPLE/PRELOAD 指令预加载到边界扫描链的值驱动到引脚上
0001	INTEST	将被 SAMPLE/PRELOAD 指令预加载到边界扫描链的值驱动到控制器
0010	SAMPLE/PRELOAD	捕获当前的 I/O 值，并在新预加载数据移入时将采样值移出边界扫描链
1000	ABORT	将数据移入“ARM 调试端口中止（Debug Port Abort）寄存器”
1010	DPACC	将数据移入和移出“ARM DP 访问（access）寄存器”
1011	APACC	将数据移入和移出“ARM AC 访问（access）寄存器”
1110	IDCODE	将由 IEEE 标准 1149.1 定义的生产信息加载到 IDCODE 链然后将它移出
1111	BYPASS	通过一个移位寄存器链将 TDI 与 TDO 相连
其他	保留	默认为 BYPASS 指令，以确保 TDI 总是与 TDO 相连。

5.4.1.1 EXTEST 指令

EXTEST 指令并没有相关的数据寄存器链。EXTEST 指令将使用被 SAMPLE/PRELOAD 指令加载到边界扫描数据寄存器的数据。当 EXTEST 指令出现在指令寄存器中时，将采用与输出和输出使能相关联的边界扫描数据寄存器中的预加载数据来驱动 GPIO 引脚（pad），而不是采用来自内核的信号来驱动。这样便可以展开测试，将已知的值驱动到控制器外，并以此验证连通性。

5.4.1.2 INTEST 指令

INTEST 指令不含相关的数据寄存器链。INTEST 指令将使用被 SAMPLE/PRELOAD 指令加载到边界扫描数据寄存器的数据。当 INTEST 指令出现在指令寄存器中时，将使用与输入相关联的边界扫描数据寄存器中的预加载数据来驱动进入内核的信号，而不是使用来自 GPIO 引脚（pad）的信号来驱动。这样便可以将已知的值驱动到控制器内，从而可以进行测试。特别需要注意的一点是，尽管 $\overline{\text{RST}}$ 输入管脚在边界扫描数据寄存器链上，但它只可以供观察。

5.4.1.3 SAMPLE/PRELOAD 指令

通过 SAMPLE/PRELOAD 指令，可以将边界扫描数据寄存器连接到 TDI 和 TDO 之间。该指令采样管脚的当前状态以供观察，并预加载新的测试数据。每个 GPIO 管脚都含一个相关的输入、输出、以及输出使能信号。在该指令执行期间，当 TAP 控制器进入 Capture DR 状态时，将捕获每个 GPIO 管脚的输入、输出和输出使能信号。在 TAP 控制器处于 Shift DR 状态时这些采样值将以串行的方式移出 TDO，并且可用于在各种测试中进行观察或比较。

在这些输入、输出和输出使能信号的采样值被移出边界扫描数据寄存器的同时，新的数据也正通过TDI管脚移入边界扫描数据寄存器。一旦新的数据移入边界扫描数据寄存器，在TAP控制器进入Update DR状态时就会把这些数据保存到并行加载寄存器中。并行加载寄存器的这种更新能够将数据预加载到与每个输入、输出和输出使能相关的边界扫描数据寄存器中。这些被预加载的数据可以和EXTEST和INTEST指令一起使用，以便将数据驱动到控制器内或控制器外。详见“边界扫描数据寄存器”。

5.4.1.4 ABORT 指令

通过ABORT指令，可以将ABORT数据寄存器链连接到TDI和TDO之间。该指令提供对ARM调试访问端口（DAP）的ABORT寄存器的读和写操作。将合适的数据移入数据寄存

器可以清除各种错误位或针对先前所作出的请求发出DAP中止信号。详见“ABORT数据寄存器”。

5.4.1.5 DPACC 指令

通过DPACC指令，可以将DPACC数据寄存器链连接到TDI和TDO之间。该指令提供对ARM调试访问端口（DAP）的DPACC寄存器的读和写操作。将合适的数据移入数据寄存器，并从该寄存器中读取输出数据，这样便可以对ARM调试和状态寄存器进行读和写操作，详见“DPACC数据寄存器”。

5.4.1.6 APACC 指令

通过APACC指令，可以将APACC数据寄存器链连接到TDI和TDO之间。该指令提供对ARM调试访问端口（DAP）的APACC寄存器的读和写操作。将合适的数据移入数据寄存器，并从该寄存器中读取输出数据，这样便可以通过调试端口对内部组件和总线进行读和写操作。详见“APACC数据寄存器”。

5.4.1.7 IDCODE 指令

通过IDCODE指令，可以将IDCODE数据寄存器链连接到TDI和TDO之间。该指令提供生产厂商的信息、器件型号和ARM内核的版本信息。测试设备和调试器可以使用这些信息来自动配置它们的输入和输出数据流。在上电复位（POR）、$\overline{\text{TRST}}$ 信号变有效，或进入Test-Logic-Reset状态时，IDCODE是默认加载到JTAG指令寄存器的一个指令。详见 5.4.2.1 节“IDCODE数据寄存器”。

5.4.1.8 BYPASS 指令

通过BYPASS指令，可以将BYPASS数据寄存器链连接到TDI和TDO之间。该指令用来在TDI和TDO端口之间创建一条长度最短的串行路径。BYPASS数据寄存器是一个 1 位移位寄存器。通过BYPASS指令加载特定测试中不需要的组件，可以将这些组件在JTAG扫描链中旁路掉，由此可以提高测试效率。详见 5.4.2.2 节“BYPASS数据寄存器”。

5.4.2 数据寄存器

JTAG 模块包含 6 个数据寄存器。它们是：IDCODE 寄存器、BYPASS 寄存器、Boundary Scan 寄存器、APACC 寄存器和 ABORT 串行数据寄存器链。下面的小节将对每个数据寄存器进行介绍。

5.4.2.1 IDCODE 数据寄存器

由IEEE标准 1149.1 定义的 32 位IDCODE数据寄存器的格式如 图 5.3 所示。该标准要求每个与JTAG兼容的设备都将IDCODE指令或者BYPASS指令当作默认指令来执行。IDCODE数据寄存器的LSB(最低位)被定义成 1，以将它和LSB为 0 的BYPASS指令区分开来。这样，自动配置测试工具就可以决定哪个指令才是默认指令了。

JTAG 最普遍还是应用在生产厂商测试组件以及开发和调试程序等场合下。为了便于使用自动配置调试工具，IDCODE 指令输出一个 0x1BA00477 值。该值表示 ARM Cortex-M3 的第一版处理器。这样，调试就可以自动进行配置以便在调试过程中能够正确的和 Cortex-M3 一起工作。

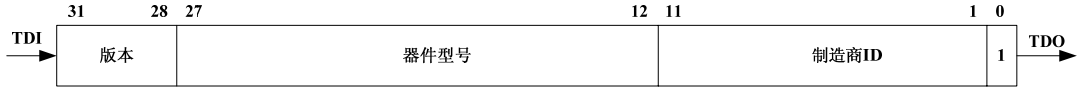


图 5.3 IDCODE 寄存器的格式

5.4.2.2 BYPASS 数据寄存器

由IEEE标准 1149.1 定义的 1 位BYPASS数据寄存器的格式如 图 5.4 所示。该标准要求每个与JTAG兼容的器件将BYPASS指令或者IDCODE指令当作默认指令来执行。BYPASS数据寄存器的LSB被定义成 0，以便将它与LSB为 1 的IDCODE指令区分开来。这样，自动配置测试工具就可以决定那条指令才是默认指令了。

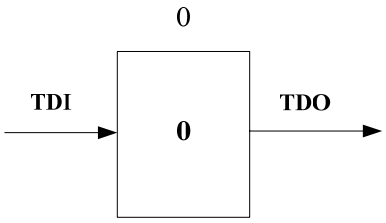


图 5.4 BYPASS 寄存器的格式

5.4.2.3 边界扫描数据寄存器

边界扫描数据寄存器的格式如 图 5.5 所示。边界扫描（Boundary Scan）数据寄存器包含每个GPIO管脚，这些管脚的次序为JTAG端口管脚排列的逆时针方向。每个GPIO管脚含 3 个与之相关的数字信号，这些信号都包括在链中。这三个信号分别为输入、输出和输出使能，而且它们就按照前述的这种顺序排列，如下图所示。除了GPIO管脚外，控制器的复位管脚 $\overline{\text{RST}}$ 也包含在这个链中。因为该复位管脚总是输入，所有数据寄存器链仅包含输入信号。

当通过 SAMPLE/PRELOAD 指令访问边界扫描数据寄存器时，每个数字引脚（pad）的输入、输出以及输出使能信号将被采样，然后采样值会被移出待校验的链。这些值的采样发生在 TAP 控制器的 Capture DR 状态下的 TCK 上升沿。当被采样的数据正从 TAP 控制器的 Shift DR 状态下的边界扫描链移出时，新的数据便可以被预先加载到链中，以便和 EXTEST 和 INTEST 指令一起使用。这些指令要不通过 EXTEST 指令将数据强制移出控制器，要不就通过 INTEST 指令将数据强制移入控制器。

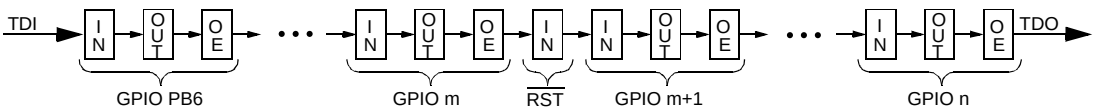


图 5.5 边界扫描寄存器的格式

要详细了解每个GPIO端口的输入、输出和输出使能位，请参考群星（stellaris）系列的边界扫描描述语言（BSDL）文件，这些文件都可以从 www.luminarymicro.com网站上下载。

5.4.2.4 APACC 数据寄存器

由 ARM 定义的 35 位 APACC 数据寄存器的格式如“ARM® Cortex-M3 技术参考手册”

中所述。

5.4.2.5 DPACC 数据寄存器

由 ARM 定义的 35 位 DPACC 数据寄存器的格式如“ARM® Cortex-M3 技术参考手册”中所述。

5.4.2.6 ABORT 数据寄存器

由 ARM 定义的 35 位 ABORT 数据寄存器的格式如“ARM® Cortex-M3 技术参考手册”中所述。

第6章 系统控制

系统控制决定了器件的全部操作。它提供有关器件的信息，控制器件和各个外设的时钟，并处理复位检测和报告。

6.1 功能描述

系统控制模块提供以下功能：

- 器件标识
- 局部控制，例如复位、功率与时钟控制
- 系统控制（运行、睡眠和深度睡眠模式）

6.1.1 器件标识

7 个只读寄存器提供了有关微控制器的信息，这些信息包括版本、元件型号、SRAM 大小、Flash 大小和其它特性。详见“**DID0**、**DID1**、**DC0-DC4** 寄存器”。

6.1.2 复位控制

本小节论述复位过程中硬件方面的功能和复位序列之后的系统软件请求。

6.1.2.1 复位源

LM3S617 控制器有 6 个复位源：

1. 外部复位输入管脚（$\overline{\text{RST}}$）生效
2. 上电复位（POR）
3. 内部掉电（BOR）监测器
4. 软件启动的复位（利用软件复位寄存器）
5. 看门狗定时器复位条件违犯
6. 内部低压差线性（LDO）稳压器输出

复位之后，**复位原因（RESC）**寄存器中的对应位置位。该寄存器中的位具有“粘着特性（sticky）”，在通过多个复位序列之后仍能保持其状态，外部复位除外。外部复位之后，**RESC** 寄存器中的其它所有位清零。

注：主振荡器供外部复位和上电复位使用，内部振荡器供内部复位和时钟验证电路等内部处理使用。

6.1.2.2 $\overline{\text{RST}}$ 管脚有效

外部复位管脚（$\overline{\text{RST}}$）可用于复位微控制器。该复位信号将内核及所有外设复位，JTAG TAP 控制器（见“JTAG 接口”）除外。外部复位顺序如下：

1. 外部复位管脚（$\overline{\text{RST}}$）生效，然后失效。
2. $\overline{\text{RST}}$ 失效之后，晶体主振荡器必须经过一段时间才能稳定下来，内部主振荡器计数器会对这段时间（15-30ms）进行计时。在此期间，控制器其余部分的内部复位保持有效。

3. 内部复位释放，微控制器读取和加载初始堆栈指针、初始程序计数器、以及由程序计数器指定的第 1 条指令，然后开始执行。

外部复位时序如 图 19.8 所示。

6.1.2.3 上电复位（POR）

上电复位（POR）电路检测电源电压是否上升，并在检测到电压上升时产生片内复位脉冲。为使用片内电路，$\overline{RST}$ 输入需连接一个上拉电阻（1K~10KΩ）。

在片内上电复位脉冲结束时，器件必须在指定的工作参数范围内操作。指定的工作参数包括电源电压、频率、温度等等。如果在POR就要结束时还没有满足工作条件，那么群星（Stellaris）控制器将不能正确工作。此时，必须使用外部电路将复位时间延长。外部电路如 图 6.1 所示，与 $\overline{RST}$ 输入相连。

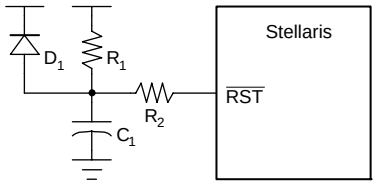


图 6.1 延长复位时间的外部电路

R_1 和 C_1 定义了上电延迟时间。R_2 减轻了 $\overline{RST}$ 输入的泄漏。C1 通过二极管在电源关断时快速放电。

上电复位顺序如下：

1. 控制器等待后来的外部复位（ $\overline{RST}$ ）或等待内部 POR 变为无效。
2. 复位失效后，晶体主振荡器必须经过一段时间才能稳定下来，内部主振荡器计数器会对这段时间（15-30ms）进行计时。在此期间，控制器其余部分的内部复位保持有效。
3. 内部复位释放，控制器读取和加载初始堆栈指针、初始程序计数器、以及由程序计数器指定的第 1 条指令，然后开始执行。

内部POR只在控制器最初上电时生效，上电复位时序如 图 19.9 所示。

6.1.2.4 掉电复位（BOR）

由于输入电压下降而导致内部掉电检测器生效，这一特性可用来复位控制器。这种复位特性在最初是被禁止的，但是可以通过软件来使能。

系统提供了一个掉电检测电路，该电路在 V_{DD} 低于 V_{BTH} 时触发，其目的是为了防止逻辑电路和外设在低于 V_{DD} 电压或非 LDO 电压下工作时产生不正确操作。在检测到掉电条件时，系统会产生控制器中断或系统复位。BOR 电路有一个数字滤波器，避免进行与噪音相关的检测。我们可以选择将这种掉电复位特性使能/禁能。

掉电复位利用**上电和掉电复位控制（PBORCTL）**寄存器进行控制。**PBORCTL** 寄存器的 BORIOR 位必须置位，以便在出现掉电时触发一次复位。掉电复位的顺序如下：

1. 当 V_{DD} 低于 V_{BTH} 时，设置内部 BOR 条件。
2. 如果 **PBORCTL** 寄存器的 BORWT 位置位，那么在一段时间之后可以重新对 BOR 条件（时间由 BORTIM 指定）进行采样，以此来确定原来的条件是否由噪音引起。

如果再一次不满足 BOR 条件，则不产生任何动作。

3. 如果 BOR 条件存在，则内部复位有效。
4. 内部复位释放，控制器读取和加载初始堆栈指针、初始程序计数器、以及由程序计数器指定的第 1 条指令，然后开始执行。
5. 内部 $\overline{\text{BOR}}$ 信号在 500μs 之后释放，以防止另一个 BOR 条件在软件可能调查最初掉电的原因之前置位。

内部掉电复位时序如 图 19.10 所示。

6.1.2.5 软件复位

每个外设都能通过软件来复位。LM3S617 有 3 个寄存器是用来控制软件复位功能（详见**SRCRn**寄存器）的。如果寄存器中与外设对应的位置位，则对应外设将复位。复位寄存器的编码与外设和片内功能的时钟门控的编码是一致的（见 6.1.5 节“系统控制”）。向位通道写入 1 会将对应单元复位。**注**：用于指定单元全部时钟的所有复位信号在软件启动复位后生效。

整个系统也能通过软件来复位。将 Cortex-M3 应用中断和复位控制寄存器中的 SYSRESETREQ 位置位，可将包括内核在内的整个系统复位。软件启动的复位顺序如下：

1. 通过对 ARM Cortex-M3 应用中断和复位控制寄存器中的 SYSRESETREQ 位执行写操作，可启动软件复位。
2. 内部复位生效。
3. 内部复位释放，控制器读取和加载初始堆栈指针、初始程序计数器，取出由程序计数器指定的第 1 条指令，然后开始执行。

软件启动的系统复位的时序如 图 19.11 所示。

6.1.2.6 看门狗定时器复位

看门狗定时器模块的功能是防止系统挂起（hang）。看门狗定时器可配置为在其第一次超时（time out）时向控制器产生中断，第二次超时（time out）时产生复位信号。

发生第一次溢出事件之后，将**看门狗定时器装载（WDTLOAD）**寄存器的值重新装入 32 位计数器，然后定时器从该值继续递减计数。如果在第一次超时中断清零之前定时器再次递减到零，并且复位信号已使能，则看门狗定时器会将其复位信号发送到系统。看门狗定时器复位顺序如下：

1. 看门狗定时器第二次超时，无需对其服务。
2. 内部复位生效
3. 内部复位释放，控制器读取和加载初始堆栈指针、初始程序计数器、并取出由程序计数器指定的第 1 条指令，然后开始执行。

看门狗复位时序见 图 19.12 所示。

6.1.2.7 低压差线性稳压器（Linear Drop-Out）

当低压差线性（LDO）稳压器输出不可调整时，将产生复位。该特性最初是禁止的，可通过软件来使能。利用 **LDO 功率控制（LDOPCTL）**寄存器可对 LDO 进行控制。LDO 复位序列如下：

1. LDO 不可调整，**LDOARST** 寄存器的 LDOARST 位置位。
2. 内部复位生效
3. 内部复位释放，控制器读取和加载初始堆栈指针、初始程序计数器、并取出由程序计数器指定的第 1 条指令，然后开始执行。

LDO复位时序如 图 19.13 所示。

6.1.3 功率控制

LDO 稳压器允许对片内输出电压（V_{OUT}）进行调整。输出电压可以在 2.25～2.75V 范围内，增量为 50mV。调整操作通过 **LDO 功率控制（LDOPCTL）**寄存器的 VADJ 位来实现。

6.1.4 时钟控制

系统控制决定了该器件的时钟计时和控制。

6.1.4.1 基础时钟源

LM3S617 器件可以使用以下两个基础时钟源：

- 主振荡器，由外部晶体或单端时钟源来驱动。由晶体驱动时，主振荡器源指定为运行在 1-8MHz。但当晶体用作 PLL 源时，它必须在 3.579545～8.192MHz 之间以满足 PLL 要求。由单端时钟源驱动时，频率范围是从 DC 到器件的指定速率。
- 内部振荡器，它是片内自由运行的时钟。内部振荡器指定的运行速率为 12MHz ±50%，它能用来给系统提供时钟，但必须满足频率范围的容差（tolerance）。

内部系统时钟可由上述两个参考源中的任一个驱动，如果 PLL 输入连接到满足其 AC 要求的时钟源，则也可由内部 PLL 来驱动。

几乎所有的时钟控制都是通过**运行-模式时钟配置（RCC）**寄存器来提供的。

图 6.2 显示了主时钟树逻辑。外设模块由系统时钟信号驱动，可以设置为使能/禁止。ADC时钟信号被自动分频成 14～18MHz，以供ADC正确操作使用。PWM时钟信号通过系统时钟分频得到，使得PWM电路的频率范围更宽。

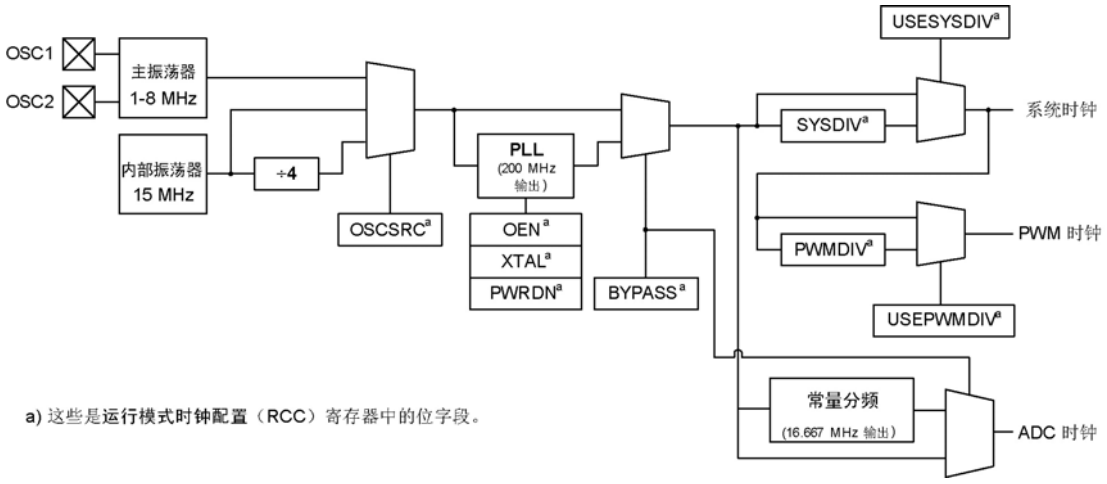


图 6.2 主时钟树

6.1.4.2 PLL 频率配置

用户不对 PLL 频率直接进行控制，但要求使用的外部晶体要与内部 PLL-晶体表匹配。该表针对所选的晶体产生最适合的 PLL 参数。尽管频率在±1%内，但并不是所有的晶体都能使 PLL 在精确的 200MHz 下工作。PLL 查找结果保存在 **XTAL 到 PLL 转换（PLLCTL）**寄存器中。

表 6.4 描述了可用的晶体选项和PLLCTL寄存器的默认设置。将晶体编号写入**运行-模式时钟配置（RCC）**寄存器中的XTAL字段。任何时候当XTAL变化时，执行内部表的读操作可获得正确的值。表 6.4 描述了可用的晶体选项和默认设置。

6.1.4.3 PLL 模式

PLL 有两种操作模式：正常和掉电。

- 正常模式：PLL 将输入时钟参考倍频并驱动输出。
- 掉电模式：大部分 PLL 内部电路被禁能，PLL 不驱动输出。

使用**RCC**寄存器可以对PLL模式进行设置，如 表 6.4 所示。

6.1.4.4 PLL 操作

如果 PLL 配置发生变化，PLL 输出在一段时间（PLL T_{READY}=0.5ms）内将不稳定，在此期间，PLL 不可用作时钟参考。

PLL 可通过以下方法改变：

- 更改为 **RCC** 寄存器中的 XTAL 值——写入相同值不会引起重锁定（relock）操作。
- 将 PLL 模式从掉电改为正常。

T_{READY} 的值由计数器来测量。计数器的时钟由主振荡器提供。考虑到主振荡器的范围，递减计数器的初值设置为 0x1200（即外部振荡器时钟为 8.192-MHz 时大约为 600μs）。此时将提供硬件以确保 PLL 不用作系统时钟，直到发生上述其中一个变化之后 PLL 才满足 T_{READY} 条件。用户需确保在 **RCC** 寄存器转换为使用 PLL 之前，必须有稳定的时钟源（与主振荡器一样）。

6.1.4.5 时钟验证定时器（clock verification timer）

LM3S617 有 3 个相同的时钟验证电路，可通过软件来使能。该电路使用定时器通过一个慢速时钟来检验快速时钟：

- 主振荡器检验 PLL
- 主振荡器检验内部振荡器
- 内部振荡器除以 64 检验主振荡器

如果验证定时器功能使能并检测到失败，则主时钟树立即切换到工作时钟，并向控制器产生中断。然后软件能够决定采取的行动的过程。在没有写 **CLKVCLR** 寄存器、没有外部复位或 POR 复位时，失败指示和时钟切换不会清零。时钟验证定时器由 **RCC** 寄存器中的 PLLVER、IOSCVER 和 MOSCVER 位控制。

6.1.5 系统控制

为了节省功耗，当控制器处于运行、睡眠、和深度睡眠模式时，请分别使用 **RCGCn**、**SCGCn** 和 **DCGCn** 寄存器来控制系统中各个外设或模块的时钟门控逻辑。**DC1**、**DC2** 和 **DC4** 寄存器作为 **RCGCn**、**SCGCn** 和 **DCGCn** 寄存器的写屏蔽。

在运行模式中，控制器积极地执行代码。在睡眠模式中，器件的时钟不变，但控制器不再执行代码（并且也不再需要时钟）。在深度睡眠模式中，器件的时钟可以改变（由运行模式的时钟配置决定），并且控制器不再执行代码（也不需要时钟）。中断可使器件从其中一种睡眠模式返回到运行模式；在代码的请求下可以进入睡眠模式。在本节中将对每种模式进行详细描述。

6.1.5.1 运行模式

运行模式下，处理器和所有当前被 **RCGCn** 寄存器使能的外设均可以正常运行。系统时钟可以由包括 PLL 在内的所有可用时钟源提供。

6.1.5.2 睡眠模式

睡眠模式下，Cortex-M3 处理器内核和存储器子系统都不使用时钟。外设仅在相应的时钟门控在 **SCGCn** 寄存器中使能且 Auto Clock Gating（见 RCC 寄存器）使能时，或者在相应的时钟门控在 **RCGCn** 寄存器中使能且 Auto Clock Gating 被禁能时，才使用时钟。睡眠模式下，系统时钟源和频率均与运行模式下相同。

6.1.5.3 深度睡眠模式

深度睡眠模式下，Cortex-M3 处理器内核和存储器子系统都不使用时钟。外设仅在相应的时钟门控在 **DCGCn** 寄存器中使能且 Auto Clock Gating（见 RCC 寄存器）使能时，或者在相应的时钟门控在 **RCGCn** 寄存器中使能且 Auto Clock Gating 被禁能时，才使用时钟。在睡眠模式下，系统时钟源默认为主振荡器。但如果 **DSLPCLKCFG** 寄存器中的 IOSC 位被置位，那么系统时钟源也可以是内部振荡器。在使用 **DSLPCLKCFG** 寄存器时，如有必要，可以让内部振荡器上电，同时断开主振荡器。如果 PLL 在执行 WFI 指令时工作，硬件将会让主振荡器断电，并将激活的 RCC 寄存器中的 SYSDIV 字段分别变为/16 或/64。当发生深度睡眠退出事件时，在使能深度睡眠期间被停止的时钟前，硬件先将系统时钟的时钟源和频率变回到开始进入深度睡眠模式时的值。

6.2 初始化和配置

PLL 的配置可通过直接向**运行-模式时钟配置（RSC）**寄存器执行写操作来实现。成功改变基于 PLL 的系统时钟所需的步骤如下：

1. 通过将 **RCC** 寄存器的 BYPASS 位置位以及将 USESYS 位清零，PLL 和系统时钟分频器旁路。该操作将系统配置为选择“原始的（raw）”时钟源（使用主振荡器或内部振荡器），并在系统时钟切换为 PLL 之前允许新的 PLL 配置生效。
2. 选择晶体的值（XTAL）和振荡源（OSCSRC），并将 **RCC** 中的 PWRDN 和 OE 位清零。XTAL 值的设置操作将自动获得所选晶体的有效的 PLL 配置数据，PWRDN 和 OE 位的清零操作将给 PLL 及其输出供电并将它们使能。
3. 选择所需的系统分频器（SYSDIV）并置位 **RCC** 的 USESYS 位。SYSDIV 字段决定了微控制器的系统频率。
4. 通过查询原始(raw)中断状态（RIS）寄存器的 PLLLRIS 位来等待 PLL 锁定。如果 PLL 没有锁定，则配置无效。
5. 通过将 RCC 的 BYPASS 位清零来使能对 PLL 的使用。

要点：如果在 PLL 锁定之前将 BYPASS 位清零，器件可能变为不可用。

6.3 寄存器映射

表 6.1 列出的是系统控制寄存器，它们按照功能分组。表中所列偏移量都是寄存器地址相对于系统控制基址 0x400FE000 的 16 进制增量。

表 6.1 系统控制的寄存器映射

偏移量	名称	复位	类型	描述
器件标识和功能				
0x000	DID0	-	RO	器件标识 0
0x004	DID1	-	RO	器件标识 1
0x008	DC0	0x001F000F	RO	器件功能 0
0x010	DC1	0x001132BF	RO	器件功能 1
0x014	DC2	0x01070013	RO	器件功能 2
0x018	DC3	0x3F3F01FF	RO	器件功能 3
0x01C	DC4	0x0000001F	RO	器件功能 4
局部控制				
0x030	PBORCTL	0x00007FFD	R/W	上电和掉电复位控制
0x034	LDOPCTL	0x00000000	R/W	LDO 功率控制
0x040	SRCR0	0x00000000	R/W	软件复位控制 0
0x044	SRCR1	0x00000000	R/W	软件复位控制 1
0x048	SRCR2	0x00000000	R/W	软件复位控制 2
0x050	RIS	0x00000000	RO	原始中断状态
0x054	IMC	0x00000000	R/W	中断屏蔽控制
0x058	MISC	0x00000000	R/W1C	屏蔽中断状态并清零
0x05C	RESC	-	R/W	复位原因

续上表

偏移量	名称	复位	类型	描述
0x060	RCC	0x078E3AC0	R/W	运行-模式时钟配置
0x064	PLLCFG	-	RO	XTAL 到 PLL 转换
系统控制				
0x100	RCGC0	0x00000001	R/W	运行-模式时钟的门控控制 0
0x104	RCGC1	0x00000000	R/W	运行-模式时钟的门控控制 1
0x108	RCGC2	0x00000000	R/W	运行-模式时钟的门控控制 2
0x110	SCGC0	0x00000001	R/W	睡眠-模式时钟的门控控制 0
0x114	SCGC1	0x00000000	R/W	睡眠-模式时钟的门控控制 1
0x118	SCGC2	0x00000000	R/W	睡眠-模式时钟的门控控制 2
0x120	DCGC0	0x00000001	R/W	深度-睡眠-模式时钟的门控控制 0
0x124	DCGC1	0x00000000	R/W	深度-睡眠-模式时钟的门控控制 1
0x128	DCGC2	0x00000000	R/W	深度-睡眠-模式时钟的门控控制 2
0x144	DSLPCLKCFG	0x78000000	R/W	深度-睡眠时钟配置
0x150	CLKVCLR	0x00000000	R/W	时钟验证清零
0x160	LDOARST	0x00000000	R/W	允许不可调整的 LDO 来将元件复位

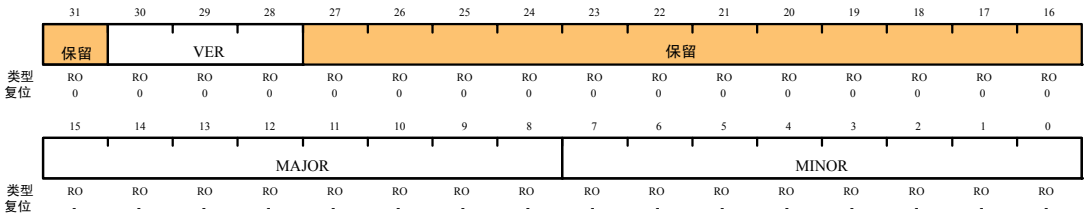
6.4 寄存器描述

下文将按地址偏移的数字顺列出系统控制寄存器，并对它们进行描述。

寄存器 1：器件标识 0（DID0），偏移量 0x000

该寄存器用来识别器件的版本。

器件标识 0 (DID0)
偏移量 0x000



位/字段	名称	类型	复位	描述
31	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。
30:28	VER	RO	0	该字段定义了 **DID0** 寄存器格式的版本： 0＝群星（Stellaris）微控制器的寄存器版本
27:16	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。
15:8	MAJOR	RO	-	该字段指定了器件的主要修订号。主要修订号在元件型号中以字母指示（A 为第 1 版，B 为第二版，依此类推）。 该字段编码如下： 0：修订 A（原始器件） 1：修订 B（首次修订）….依此类推

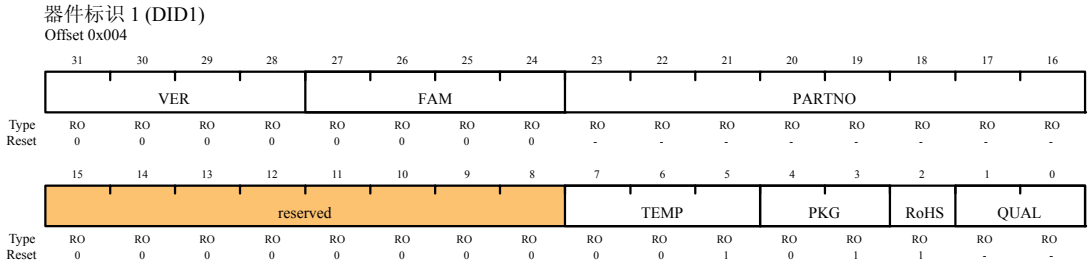
续上表

位/字段	名称	类型	复位	描述
7:0	MINOR	RO	-	该字段指定了器件的次要修订号，以数字表示。该字段编码如下： 0：无改动，最近一次的更新为主要修订 1：从上一次主要修订更新后的一次互连改动（interconnect change）。 2：从上一次主要修订更新后的两次互连改动，等等。

寄存器 2：器件标识 1（DID1），偏移量 0x004

该寄存器用来识别器件的系列、元件型号、温度范围和封装类型。

注：位图中表示的一些值是器件所特有的。下表表示的值针对的是用户购买的器件。



位/字段	名称	类型	复位	描述
31:28	VER	RO	0x0	该字段定义了 **DID1** 寄存器格式的版本： 0 为 Stellaris 微控制器的寄存器版本
27:24	FAM	RO	0x0	系列 该字段提供 Luminary Micro 产品组合（product portfolio）中的器件的系列标识。 0x0 表示是 Stellaris 系列微控制器。
23:16	PART NO	RO	0x28	元件型号 该字段提供系列内的器件的元件型号 0x28 表示是 LM3S617 微控制器。
15:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。
7:5	TEMP	RO	见表	温度范围 该字段指定器件的温度等级。 该字段编码如下： **TEMP** / **描述** 000 / 商业温度范围（0℃～70℃） 001 / 工业温度范围（-40℃～85℃） 010-111 / 保留
4:3	PKG	RO	0x1	该区域指定了封装类型。值为 1 时表示是 48 脚 LQFP 封装。
2	RoHS	RO	1	符合 RoHS 标准 该位为 1 时表示符合 RoHS 标准

续上表

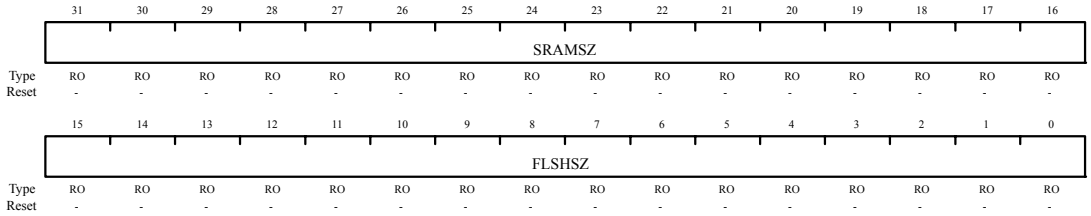
位/字段	名称	类型	复位	描述
1:0	QUAL	RO	见表	该字段指定了器件的认证情况（qualification status），编码如下： QUAL 描述：00 工程样片（未经认证）；01 试制生产（未经认证）；10 完全通过认证；11 保留

寄存器 3：器件功能 0（DC0），偏移量 0x008

该寄存器根据元件来预先定义并能用来验证其特性。

注：位图中表示的一些值是器件所特有的。下表表示的值针对的是用户购买的器件。

器件功能寄存器 0 (DC0)
Offset 0x004

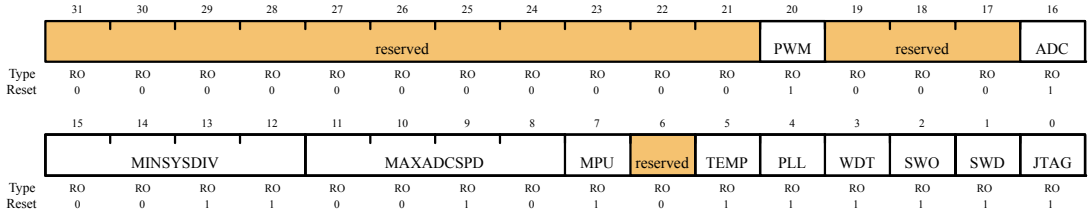


位/字段	名称	类型	复位	描述
31:16	SRAMSZ	RO	0x001F	表示片内 SRAM 的大小。值为 0x001F 时表示 SRAM 为 8KB。
15:0	FLSHSZ	RO	0x000F	表示片内 Flash 的大小。值为 0x000F 时表示 Flash 为 32KB。

寄存器 4：器件功能 1（DC1），偏移量 0x010

该寄存器根据元件来预先定义并能用来验证其特性。

器件功能 1 (DC1)
Offset 0x010



位/字段	名称	类型	复位	描述
31:21	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。
20	PWM[a]	RO	1	该位为 1 时表示含有 PWM 模块
19:17	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
16	ADC[a]	RO	1	该位为 1 时表示含有 ADC 模块

续上表

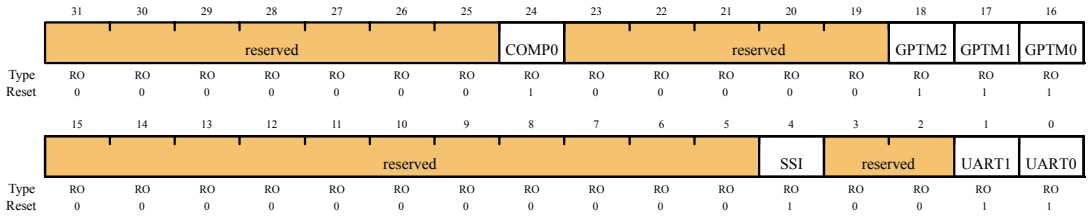
位/字段	名称	类型	复位	描述
15:12	MINSYSDIV	RO	0x03	该字段的复位值与硬件相关。值为 0x03 表示 50MHz 的 CPU 时钟，PLL 分频器的值为 4。要想了解如何使用 SYSDIV 位来改变系统时钟除数，请参考“**RCC** 寄存器”。
11:8	MAXADCSPD[a]	RO	0x2	该字段指示 ADC 采样数据的最大速率。该位为 0x2 时表示 500，000 采样/秒，即每秒采样 50 万次。
7	MPU	RO	1	该位指示 Cortex-M3 中的存储器保护单元(MPU)是否可用。该位为 0 时表示 MPU 不可用，为 1 时表示可用。有关 MPU 的信息详见“ARM® CortexTM-M3 技术参考手册”。
6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5	TEMP	RO	1	该位确定是否存在内部温度传感器
4	PLL	RO	1	该位为 1 时表示器件中存在已实现的 PLL。
3	WDT[a]	RO	1	该位为 1 时表示器件含有看门狗定时器。
2	SWO[a]	RO	1	该位为 1 时表示含有 ARM 串行线输出（SWO）跟踪端口功能。
1	SWD[a]	RO	1	该位为 1 时表示含有 ARM 串行线调试（SWD）功能。
0	JTAG[a]	RO	1	该位为 1 时表示含有 JTAG 端口。

a 这些位屏蔽了**运行-模式时钟门控控制 0（RCGC0）**寄存器、**睡眠-模式时钟门控控制 0（SCGC0）**寄存器，以及**深度-睡眠-模式时钟门控控制 0（DCGC0）**寄存器。没有标注的位作为 0 传送。ADCSP 被调节成 DC1 指定的最大值。

寄存器 5：器件功能 2（DC2），偏移量 0x014

该寄存器由元件进行预先定义并可用来验证其特性。

器件功能2（DC2）寄存器
偏移量 0x014



位/字段	名称	类型	复位	描述
31: 25	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
24	COMP0	RO	1	该位为 1 表示存在模拟比较器 0
23:19	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
18	GPTM2	RO	1	该位为 1 表示存在通用定时器模块 2
17	GPTM1	RO	1	该位为 1 表示存在通用定时器模块 1
16	GPTM0	RO	1	该位为 1 表示存在通用定时器模块 0
15: 5	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
4	SSI	RO	1	该位为 1 表示存在 SSI 模块
3:2	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变

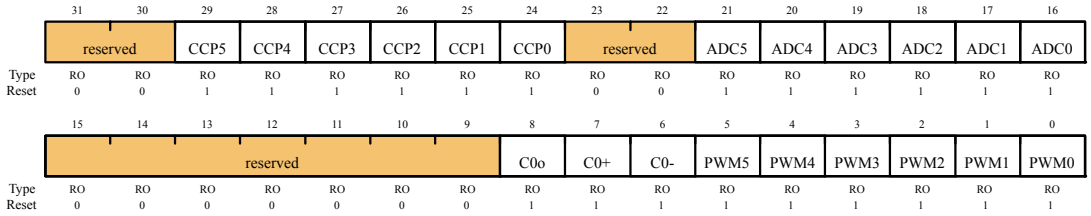
续上表

位/字段	名称	类型	复位	描述
1	UART1	RO	1	该位为 1 表示存在 UART1 模块
0	UART0	RO	1	该位为 1 表示存在 UART0 模块

寄存器 6：器件功能 3（DC3），偏移量 0x018

该寄存器根据元件来预先定义并能用来验证其特性。

器件功能 3 (DC3)
Offset 0x018

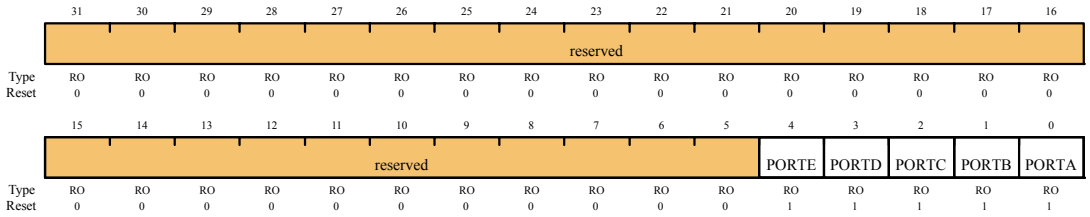


位/字段	名称	类型	复位	描述
31:30	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
29	CCP5	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 5
28	CCP4	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 4
27	CCP3	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 3
26	CCP2	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 2
25	CCP1	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 1
24	CCP0	RO	1	该位为 1 表示存在捕获/比较/PWM 管脚 0
23:22	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
21	ADC5	RO	1	该位为 1 表示存在 ADC5 管脚
20	ADC4	RO	1	该位为 1 表示存在 ADC4 管脚
19	ADC3	RO	1	该位为 1 表示存在 ADC3 管脚
18	ADC2	RO	1	该位为 1 表示存在 ADC2 管脚
17	ADC1	RO	1	该位为 1 表示存在 ADC1 管脚
16	ADC0	RO	1	该位为 1 表示存在 ADC0 管脚
15:9	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
8	C0o	RO	1	该位为 1 表示存在 C0o 管脚
7	C0+	RO	1	该位为 1 表示存在 C0+管脚
6	C0-	RO	1	该位为 1 表示存在 C0-管脚
5	PWM5	RO	1	该位为 1 表示存在 PWM5 管脚
4	PWM4	RO	1	该位为 1 表示存在 PWM4 管脚
3	PWM3	RO	1	该位为 1 表示存在 PWM3 管脚
2	PWM2	RO	1	该位为 1 表示存在 PWM2 管脚
1	PWM1	RO	1	该位为 1 表示存在 PWM1 管脚
0	PWM0	RO	1	该位为 1 表示存在 PWM0 管脚

寄存器 7：器件功能 4（DC4），偏移量 0x01C

该寄存器根据元件来预先定义并能用来验证其特性。

器件功能4（DC4）寄存器
偏移量 0x01C

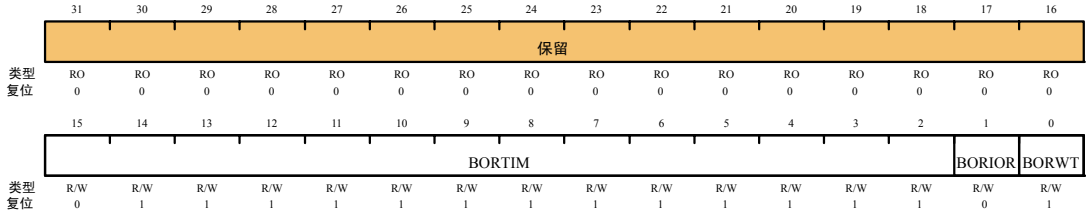


位/字段	名称	类型	复位	描述
31:5	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
4	PORTE	RO	1	该位为 1 表示存在 GPIO 端口 E
3	PORTD	RO	1	该位为 1 表示存在 GPIO 端口 D
2	PORTC	RO	1	该位为 1 表示存在 GPIO 端口 C
1	PORTB	RO	1	该位为 1 表示存在 GPIO 端口 B
0	PORTA	RO	1	该位为 1 表示存在 GPIO 端口 A

寄存器 8：上电和掉电复位控制（PBORCTL），偏移量 0x030

该寄存器用来控制初始上电复位之后的复位条件。

上电和掉电复位控制 (PBORCTL)
偏移量 0x030



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
15:2	BORTIM	R/W	0x1FFF	如果 BORWT 位置位，则该字段确定在 BOR 输出被重新采样之前延迟的内部振荡器时钟的个数。 该字段的宽度由 500µs 的 t_{BOR} 宽度和 15MHz±50%的内部振荡器(OSC)频率来控制。如果是+50%，则计数值必须超过 10,000。
1	BORIOR	R/W	0	BOR 中断或复位 该位控制如何将 BOR 事件发送给控制器。如果该位为 1 则发出复位信号，否则发出中断。

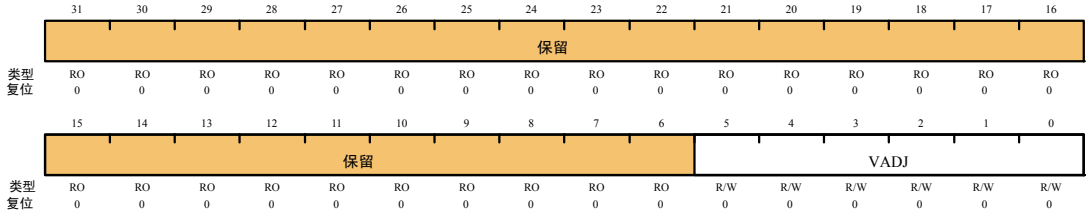
续上表

位/字段	名称	类型	复位	描述
0	BORWT	R/W	1	BOR 等待并检查噪音 该位指定了对掉电信号生效的响应。如果 BORWT 置 1，则控制器在重新采样 BOR 输出之前等待 BORTIM 个 IOSC 周期，并且如果重新采样有效，则发出 BOR 条件中断或复位。如果 BOR 重新采样无效，则初始有效的原因可能是噪音，因此要抑制中断或复位。如果 BORWT 为 0，则 BOR 有效不会对输出重新采样，并立即报告任何的条件（如果使能）。

寄存器 9：LDO 功率控制（LDOPCTL），偏移量 0x034

该寄存器的 VADJ 字段用来调整片内输出电压（V_{OUT}）。

LDO功率控制 (LDOPCTL)
偏移量 0x034



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5:0	VADJ	R/W	0x0	该字段指定了片内输出电压。VADJ字段的设置如下 表 6.2 所示。

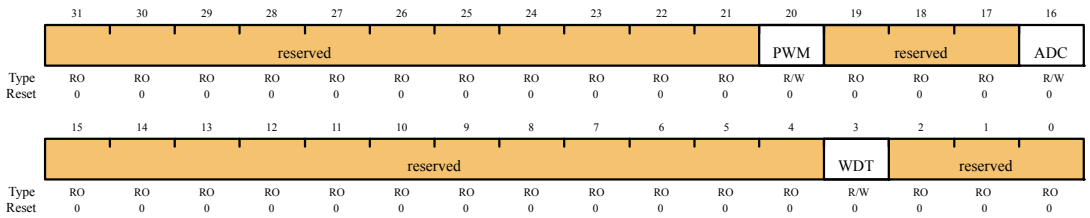
表 6.2 VADJ～V_{OUT}

VADJ	V_{OUT}(V)	VADJ	V_{OUT}(V)	VADJ	V_{OUT}(V)
0x1B	2.75	0x1F	2.55	0x03	2.35
0x1C	2.70	0x00	2.50	0x04	2.30
0x1D	2.65	0x01	2.45	0x05	2.25
0x1E	2.60	0x02	2.40	0x06~0x3F	保留

寄存器 10：软件复位控制 0（SRCR0），偏移量 0x040

写入该寄存器的值被**器件功能 1（DC1）**寄存器中的位屏蔽。

软件复位控制0（SRCR0）寄存器
偏移量 0x040

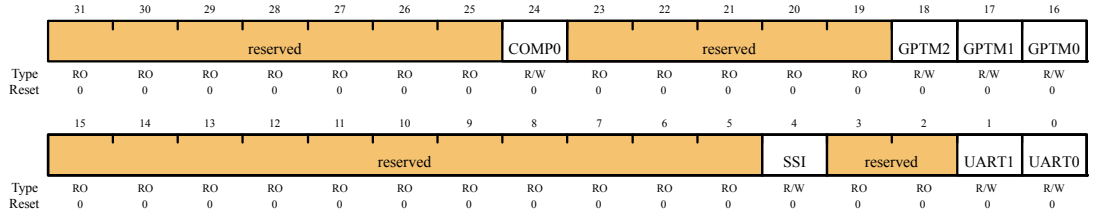


位/字段	名称	类型	复位	描述
31:21	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
20	PWM	R/W	0	PWM 单元的复位控制
19:17	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
16	ADC	R/W	0	ADC 单元的复位控制
15:4	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
3	WDT	R/W	0	看门狗单元的复位控制
2:0	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变

寄存器 11：软件复位控制 1（SRCR1），偏移量 0x044

写入该寄存器的值被**器件功能 2（DC2）**寄存器中的位屏蔽。

软件复位控制1(SRCR1)寄存器
偏移量 0x044

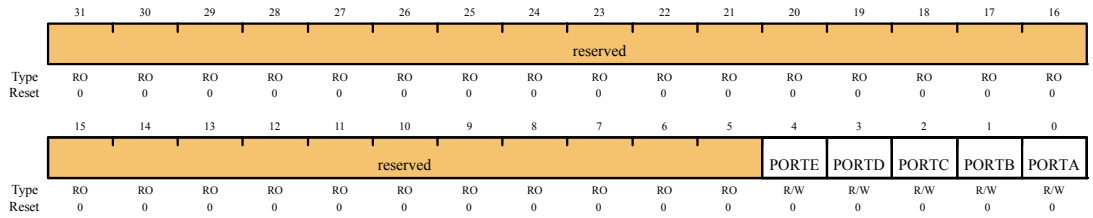


位/字段	名称	类型	复位	描述
31: 25	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
24	COMP0	R/W	0	模拟比较器 0 的复位控制
23:19	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
18	GPTM2	R/W	0	通用定时器模块 2 的复位控制
17	GPTM1	R/W	0	通用定时器模块 1 的复位控制
16	GPTM0	R/W	0	通用定时器模块 0 的复位控制
15:5	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
4	SSI	R/W	0	SSI 模块的复位控制
3:2	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
1	UART1	R/W	0	UART1 模块的复位控制
0	UART0	R/W	0	UART0 模块的复位控制

寄存器 12：软件复位控制 2（SRCR2），偏移量 0x048

写入该寄存器的值被**器件功能 4（DC4）**寄存器中的位屏蔽。

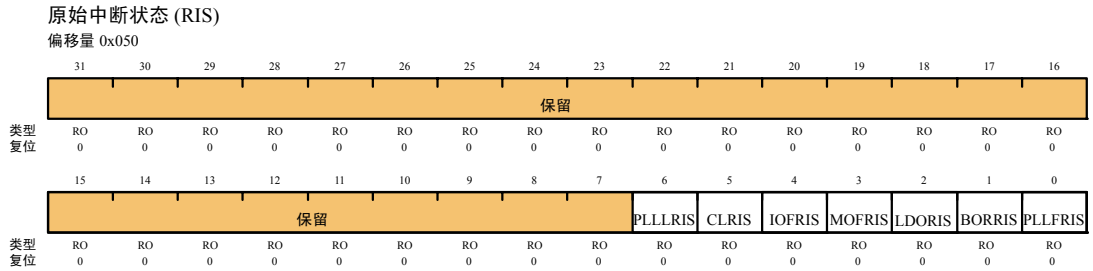
软件复位控制2（SRCR2）寄存器
偏移量 0x048



位/字段	名称	类型	复位	描述
31:5	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
4	PORTE	R/W	0	GPIO 端口 E 的复位控制
3	PORTD	R/W	0	GPIO 端口 D 的复位控制
2	PORTC	R/W	0	GPIO 端口 C 的复位控制
1	PORTB	R/W	0	GPIO 端口 B 的复位控制
0	PORTA	R/W	0	GPIO 端口 A 的复位控制

寄存器 13：原始中断状态（RIS），偏移量 0x050

系统使用该寄存器来对原始中断进行控制。寄存器中的位由硬件置位和清零。

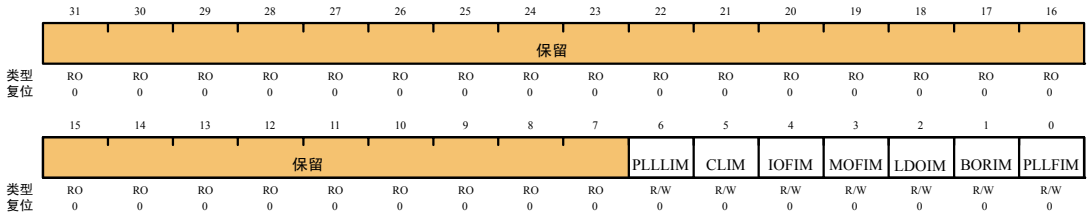


位/字段	名称	类型	复位	描述
31:7	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
6	PLLLRIS	RO	0	PLL 锁定的原始中断状态 该位在 PLL T_{READY} 定时器有效时置位。
5	CLRIS	RO	0	电流限制（current limit）的原始中断状态 该位在 LDO 的 CLE 输出有效时置位。
4	IOFRIS	RO	0	内部振荡器故障的原始中断状态 该位在检测到内部振荡器故障时置位。
3	MOFRIS	RO	0	主振荡器故障的原始中断状态 该位在检测到主振荡器故障时置位
2	LDORIS	RO	0	LDO 功率不可调整的原始中断状态 该位在 LDO 电压不可调整时置位
1	BORRIS	RO	0	掉电复位的原始中断状态 该位表示所有掉电情况的原始中断状态。如果该位置位，则检测到掉电条件。如果 **IMC** 寄存器中的 BORIM 位置位以及 **PBORCTL** 寄存器中的 BORIOR 位清零，则报告中断。
0	PLLFRIS	RO	0	PLL 故障的原始中断状态 该位在检测到 PLL 故障（停止振荡）时置位。

寄存器 14：中断屏蔽控制（IMC），偏移量 0x054

该寄存器主要被系统用来控制中断屏蔽。

中断屏蔽控制 (IMC)
偏移量 0x054

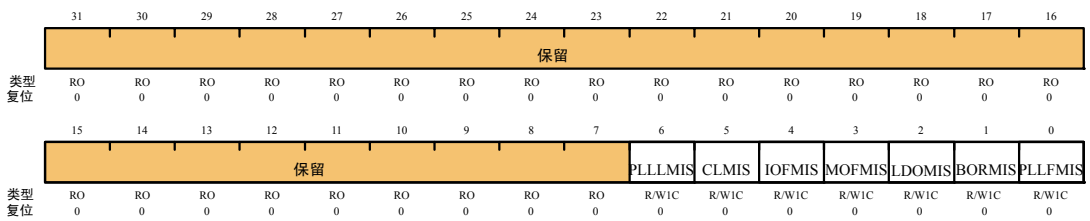


位	名称	类型	复位	描述
31:7	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
6	PLLLIM	R/W	0	PLL 锁定的中断屏蔽 该位确定在检测到 PLL 锁定时是否引起控制器中断。如果该位置位，则在 **RIS** 的 PLLLRIS 位置位时将产生中断，否则不产生中断。
5	CLIM	R/W	0	电流限制的中断屏蔽 该位确定在检测到电流限制时是否引起控制器中断。如果该位置位，则在 CLRIS 置位时将产生中断，否则不产生中断。
4	IOFIM	R/W	0	内部振荡器故障的中断屏蔽 该位确定在检测到内部振荡器故障时是否引起控制器中断。如果该位置位，则在 IOFRIS 置位时将产生中断，否则不产生中断。
3	MOFIM	R/W	0	主振荡器故障的中断屏蔽 该位确定在检测到主振荡器故障时是否引起控制器中断。如果该位置位，则在 MOFRIS 置位时将产生中断，否则不产生中断。
2	LDOIM	R/W	0	LDO 功率不可调整的中断屏蔽 该位确定在检测到 LDO 功率不可调整的情况时是否引起控制器中断。如果该位置位，则在 LDORIS 置位时将产生中断，否则不产生中断。
1	BORIM	R/W	0	掉电复位的中断屏蔽 该位确定在检测到掉电状态时是否引起控制器中断。如果该位置位，则在 BORRIS 置位时将产生中断，否则不产生中断。
0	PLLFIM	R/W	0	PLL 故障的中断屏蔽 该位确定在检测到 PLL 故障时是否引起控制器中断。如果该位置位，则在 PLLFRIS 置位时将产生中断，否则不产生中断。

寄存器 15：屏蔽后的中断状态和清零（MISC），偏移量 0x058

该寄存器主要被系统用来控制 RIS 和 IMC 相与的结果，以决定是否向控制器产生中断。寄存器的所有位都是 R/W1C，这个动作也将 **RIS** 寄存器中对应的原始中断位清零。

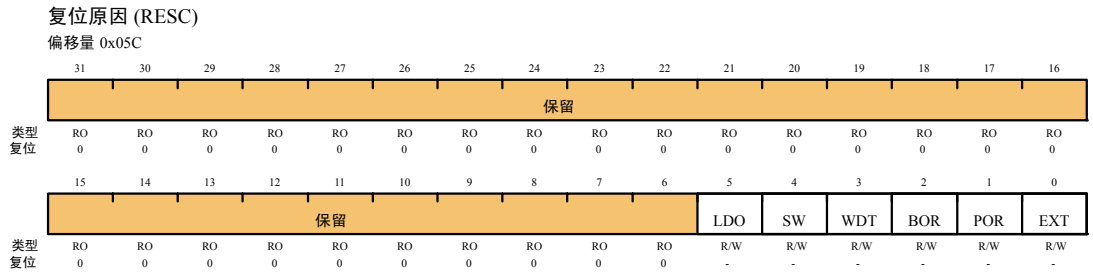
屏蔽后的中断状态和清零 (MISC)
偏移量 0x058



位/字段	名称	类型	复位	描述
31:7	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
6	PLLLMIS	R/W1C	0	PLL 锁定屏蔽后的中断状态 PLL 的 T_{READY} 定时器有效时该位置位。向该位写 1 可将中断清零。
5	CLMIS	R/W1C	0	电流限制屏蔽后的中断状态 LDO 的 CLE 输出有效时该位置位。向该位写 1 可将中断清零。
4	IOFMIS	R/W1C	0	内部振荡器故障屏蔽后的中断状态 在检测到内部振荡器故障时该位置位。向该位写 1 可将中断清零。
3	MOFMIS	R/W1C	0	主振荡器故障屏蔽后的中断状态 在检测到主振荡器故障时该位置位。向该位写 1 可将中断清零。
2	LDOMIS	R/W1C	0	LDO 功率不可调整屏蔽后的中断状态 在检测到 LDO 功率不可调整时该位置位。向该位写 1 可将中断清零。
1	BORMIS	R/W1C	0	掉电复位屏蔽后的中断状态 该位表示任一掉电条件屏蔽后的中断状态。如果该位置位，则表示检测到掉电条件。如果 **IMC** 寄存器的 BORIM 位置位，并且 **PBORCTL** 寄存器的 BORIOR 位清零，则报告中断。
0	PLLFMIS	R/W1C	0	PLL 故障屏蔽后的中断状态 在检测到 PLL 故障时该位置位。向该位写 1 可将中断清零。

寄存器 16：复位原因（RESC），偏移量 0x05C

该寄存器表明了复位事件的原因。复位值由复位原因来决定。如果是外部复位（EXT 置位），则其它所有复位位清零。但如果是由其它原因引起的，则剩余的位是“粘着的（sticky）”，可以通过软件来查看所有的复位原因。

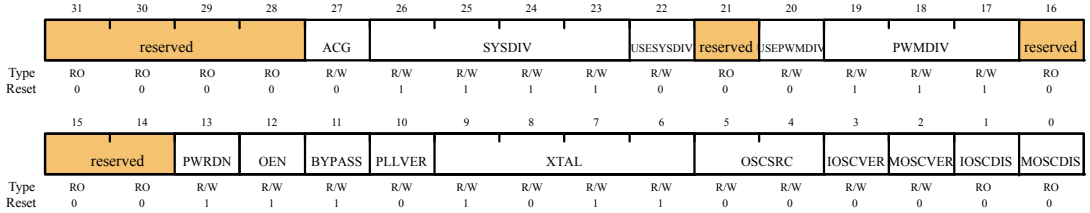


位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5	LDO	R/W	-	该位为 1 时，复位事件是由 LDO 功率不可调整引起的。
4	SW	R/W	-	该位为 1 时，复位事件是由软件复位引起的
3	WDT	R/W	-	该位为 1 时，复位事件是由看门狗复位引起的
2	BOR	R/W	-	该位为 1 时，复位事件是由掉电复位引起的
1	POR	R/W	-	该位为 1 时，复位事件是由上电复位引起的
0	EXT	R/W	-	该位为 1 时，复位事件是由外部复位（ $\overline{\text{RST}}$ ）引起的。

寄存器 17：运行-模式时钟配置（RCC），偏移量 0x060

该寄存器定义为提供时钟源控制和频率。

运行-模式时钟配置(RCC)寄存器
偏移量 0x060



位/字段	名称	类型	复位	描述
31:28	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
27	ACG	R/W	0	自动时钟门控 该位决定在控制器进入睡眠或深度睡眠模式时，系统是使用**睡眠模式时钟门控控制（SCGCn）**寄存器还是**深度睡眠模式时钟门控控制（DCGCn）**寄存器。如果置位，则当控制器处于睡眠模式时使用 **SCGCn** 或 **DCGCn** 寄存器来控制分配给外设的时钟。否则，当控制器进入睡眠模式时使用**运行模式时钟门控控制（RCGCn）**寄存器。**RCGCn** 寄存器始终用来控制运行模式下的时钟。这样，在控制器处于睡眠模式并且不需要使用外设时可以降低外设的功耗。
26:23	SYSDIV	R/W	0xF	系统时钟除数 这几个位确定使用哪个除数来从 PLL 输出（200MHz）上产生系统时钟。

二进制值	除数（BYPASS=1）	频率 (BYPASS=0)
0000	保留	保留
0001	/2	保留
0010	/3	保留
0011	/4	50MHz
0100	/5	40MHz
0101	/6	33.33MHz
0110	/7	28.57MHz
0111	/8	25MHz
1000	/9	22.22MHz
1001	/10	20MHz
1010	/11	18.18MHz
1011	/12	16.67MHz
1100	/13	15.38 MHz
1101	/14	14.29MHz
1110	/15	13.33MHz
1111	/16	12.5MHz（缺省）

如果请求的分频值更小并且正在使用 PLL，则读取**运行模式时钟配置（RCC）**寄存器时，SYSDIV 的值为 MINSYSDIV 的值。这个更小的值允许将非 PLL 的时钟源分频。

续上表

<table>
<tr><th>位/字段</th><th>名称</th><th>类型</th><th>复位</th><th>描述</th></tr>
<tr><td>22</td><td>USESYSDIV</td><td>R/W</td><td>0</td><td>使用系统时钟分频器来产生系统时钟。当选择 PLL 作为时钟源时，将强制使用系统时钟分频器。</td></tr>
<tr><td>21</td><td>保留</td><td>RO</td><td>0</td><td>保留位返回一个不确定的值，并且应该永不改变</td></tr>
<tr><td>20</td><td>USEPWMDIV</td><td>R/W</td><td>0</td><td>将 PWM 时钟分频器用作 PWM 时钟源</td></tr>
<tr><td>19:17</td><td>PWMDIV</td><td>R/W</td><td>0x7</td><td>PWM 单元时钟除数
该字段确定用来预分频系统时钟的二进制除数，分频后的结果被用作 PWM 模块的基准时钟。该时钟只能进行 2^n 分频，并且其上升沿是同步的，并且不存在 PCLK/HCLK 移相。
<table>
<tr><td>值</td><td>除数</td></tr>
<tr><td>000</td><td>/2</td></tr>
<tr><td>001</td><td>/4</td></tr>
<tr><td>010</td><td>/8</td></tr>
<tr><td>011</td><td>/16</td></tr>
<tr><td>100</td><td>/32</td></tr>
<tr><td>101</td><td>/64</td></tr>
<tr><td>110</td><td>/64</td></tr>
<tr><td>111</td><td>/64（缺省）</td></tr>
</table></td></tr>
<tr><td>16:14</td><td>保留</td><td>RO</td><td>0</td><td>保留位返回一个不确定的值，并且应该永不改变</td></tr>
<tr><td>13</td><td>PWRDN</td><td>R/W</td><td>1</td><td>PLL 掉电
该位与PLL PWRDN输入相关联。复位值为 1 时将PLL掉电。PLL模式控制见 表 6.3。</td></tr>
<tr><td>12</td><td>OEN</td><td>R/W</td><td>1</td><td>PLL 输出使能
该位确定 PLL 输出分频器是否使能。如果清零，分频器向输出发送 PLL 时钟。否则，PLL 时钟不在 PLL 模块外振荡。
注：PWRDN 和 OEN 必须都清零才能运行 PLL。</td></tr>
<tr><td>11</td><td>BYPASS</td><td>R/W</td><td>1</td><td>PLL 旁路
选择是从 PLL 输出还是 OSC 时钟源中获取系统时钟。如果该位置位，则从 OSC 时钟源获得系统时钟，否则选择被系统分频器分频后的 PLL 输出作为系统时钟。
注：为了正确工作，ADC 模块的时钟必须由 PLL 提供，或者直接由 14MHz～18MHz 的时钟源提供。</td></tr>
<tr><td>10</td><td>PLLVER</td><td>R/W</td><td>0</td><td>PLL 验证
该位控制 PLL 验证定时器的功能。如果该位置位，验证定时器使能，并在 PLL 不起作用时产生中断。否则，不使能验证定时器。</td></tr>
<tr><td>9:6</td><td>XTAL</td><td>R/W</td><td>0xB</td><td>该字段确定与主振荡器相关的晶体的值。其编码如 表 6.4 所示。</td></tr>
</table>

续上表

<table>
<tr><th>位/字段</th><th>名称</th><th>类型</th><th>复位</th><th>描述</th></tr>
<tr><td colspan="5">与振荡器相关的位</td></tr>
<tr><td>5:4</td><td>OSCSRC</td><td>R/W</td><td>0x0</td><td>从 OSC 的 4 个输入源中选择。值如下：
<table>
<tr><th>值</th><th>输入源</th></tr>
<tr><td>00</td><td>主振荡器（缺省）</td></tr>
<tr><td>01</td><td>内部振荡器</td></tr>
<tr><td>10</td><td>内部振荡器/4（如果用作 PLL 的输入，则这是必须的）</td></tr>
<tr><td>11</td><td>保留</td></tr>
</table></td></tr>
<tr><td>3</td><td>IOSCVER</td><td>R/W</td><td>0</td><td>该位控制内部振荡器验证定时器的功能。如果该位置位，则验证定时器使能，并在定时器不起作用时产生中断。否则，验证定时器不使能。</td></tr>
<tr><td>2</td><td>MOSCVER</td><td>R/W</td><td>0</td><td>该位控制主振荡器验证定时器的功能。如果该位置位，则验证定时器使能，并在定时器不起作用时产生中断。否则，验证定时器不使能。</td></tr>
<tr><td>1</td><td>IOSCDIS</td><td>R/W</td><td>0</td><td>内部振荡器禁能
0：内部振荡器被使能
1：内部振荡器被禁能</td></tr>
<tr><td>0</td><td>MOSCDIS</td><td>R/W</td><td>0</td><td>主振荡器禁能
0：主振荡器被使能
1：主振荡器被禁能</td></tr>
</table>

表 6.3 PLL 模式控制

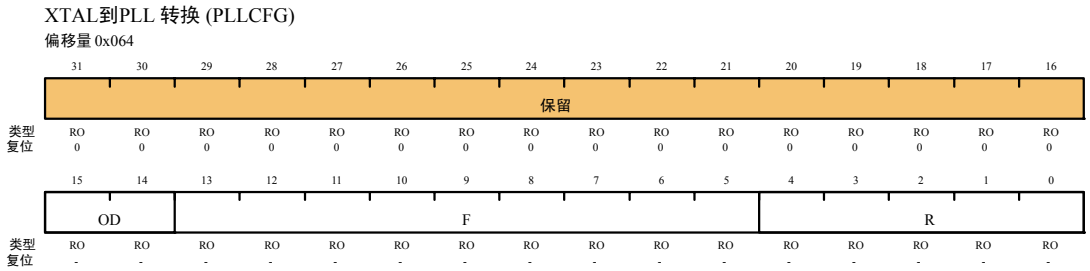
PWRDN	OEN	模式
1	X	掉电模式
0	0	正常模式

表 6.4 默认的晶体字段值和 PLL 设置

晶体编号（XTAL 二进制值）	晶体频率（MHz）
0000-0011	保留
0100	3.579545MHz
0101	3.6864MHz
0110	4 MHz
0111	4.096 MHz
1000	4.9152MHz
1001	5 MHz
1010	5.12 MHz
1011	6 MHz（复位值）
1100	6.144 MHz
1101	7.3728 MHz
1110	8 MHz
1111	8.192 MHz

寄存器 18：XTAL 到 PLL 转换（PLLCFG），偏移量 0x064

该寄存器用于将外部晶振频率转换为相应的 PLL 设置。该寄存器在复位序列期间进行初始化并在**运行-模式时钟配置（RCC）**寄存器的 XTAL 字段发生变化时进行更新。



位	名称	类型	复位	描述
31:16	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
15:14	OD	RO	-	该字段指定了供 PLL 的 OD 输入使用的值。
13:5	F	RO	-	该字段指定了供 PLL 的 F 输入使用的值。
4:0	R	RO	-	该字段指定了供 PLL 的 R 输入使用的值。

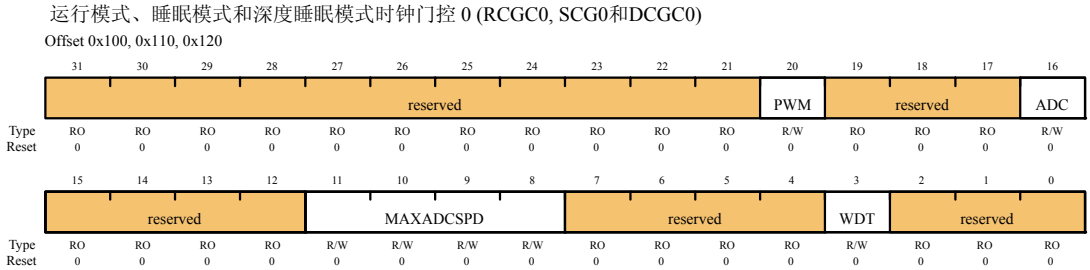
寄存器 19：运行-模式时钟门控控制 0（RCGC0），偏移量 0x100

寄存器 20：睡眠-模式时钟门控控制 0（SCGC0），偏移量 0x110

寄存器 21：深度-睡眠-模式时钟门控控制 0（DCGC0），偏移量 0x120

这几个寄存器用来控制时钟门控逻辑。每个位控制一个给定接口、功能、或单元的时钟使能。如果置位，则对应的单元接收时钟并运行。否则，对应的单元不使用时钟并禁止（节能）。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。除非特别说明，否则这些位的复位状态都为 0（不使用时钟），即所有功能单元都禁止。应用所需的端口需通过软件来使能。注：这些寄存器除了含有对接口、功能、或单元进行控制的位以外，还可能含有其它位，这样可保证与其它系列以及将来的部件实现合理的代码兼容。

RCGC0 是运行操作，**SCGC0** 是睡眠操作，**DCGC0** 是深度睡眠操作的时钟配置寄存器。将**运行-模式时钟配置（RCC）**寄存器的 ACG 位置位时，系统使用睡眠模式。



位	名称	类型	复位	描述
31:21	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
20	PWM	R/W	0	该位控制 PWM 模块的时钟门控。若置位，则 PWM 接收时钟并运行。否则，该单元不使用时钟并且不运行。[a]
19:17	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变

续上表

位	名称	类型	复位	描述
16	ADC	R/W	1	该位控制 ADC 模块的时钟门控。若置位，则 ADC 接收时钟并运行。否则，WDT 不使用时钟并且不运行。[a]
15:12	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
11:8	MAXADCSPD	R/W	0x0	该字段设置 ADC 采样数据的速率。你可以通过设置 MAXADCSPD 位来设置采样速率，但设置的采样速率不能高于最大速率。 值 / 采样速率 0x0 / 125K 采样/秒 0x1 / 250K 采样/秒 0x2 / 500K 采样/秒
7:4	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
3	WDT	R/W	0	该位控制 WDT 模块的时钟门控。若置位，则 WDT 接收时钟并运行。否则，WDT 不使用时钟并且不运行。[a]
2	SWO	R/W	0	该位控制 SWO 模块的时钟门控。若置位，则 SWO 接收时钟并运行。否则，SWO 不使用时钟并且不运行。[a]
1	SWD	R/W	0	该位控制 SWD 模块的时钟门控。若置位，则 SWD 接收时钟并运行。否则，WDT 不使用时钟并且不运行。[a]
0	JTAG	R/W	1	该位控制 JTAG 模块的时钟门控。该位的复位状态是 1。复位时，则 JTAG 接收时钟并运行。否则，WDT 不使用时钟并且不运行。[a]

a. 如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。

寄存器 22：运行-模式时钟门控控制 1（RCGC1），偏移量 0x104

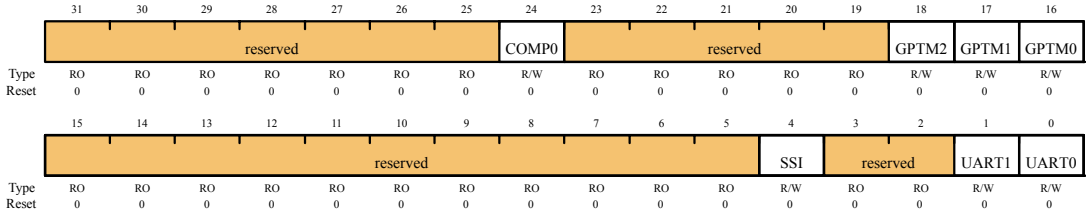
寄存器 23：睡眠-模式时钟门控控制 1（SCGC1），偏移量 0x114

寄存器 24：深度-睡眠-模式时钟门控控制 1（DCGC1），偏移量 0x124

这几个寄存器用来控制时钟门控逻辑。每个位控制一个给定接口、功能、或单元的时钟使能。如果置位，则对应的单元接收时钟并运行。否则，对应的单元不使用时钟并禁止（节能）。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。除非特别说明，否则这些位的复位状态都为 0（不使用时钟），即所有功能单元都禁止。应用所需的端口需通过软件来使能。**注**：这些寄存器除了含有对接口、功能、或单元进行控制的位以外，还可能含有其它位，这样可保证与其它系列以及将来的部件实现合理的代码兼容。

RCGC1 是运行操作，**SCGC1** 是睡眠操作，**DCGC1** 是深度睡眠操作的时钟配置寄存器。将**运行-模式时钟配置（RCC）**寄存器的 ACG 位置位时，系统使用睡眠模式。

运行-模式、睡眠－模式和深度-睡眠-模式时钟门控1(RCGC1、SCGC1和DCGC1)寄存器
偏移量 0x104，0x0114和0x124



位/字段	名称	类型	复位	描述
31: 25	保留	RO	0	保留位，返回不确定的值，并且应永不改变。
24	COMP0	R/W	0	该位控制比较器 0 模块的时钟选通。如果该位置位，比较器 0 接收时钟并运行。否则，不使用时钟并禁止。[a]
23:19	保留	R/W	0	保留位，返回不确定的值，并且应永不改变。
18	GPTM2	R/W	0	该位控制定时器 2 模块的时钟选通。如果该位置位，定时器 2 接收时钟并运行。否则，不使用时钟并禁止。[a]
17	GPTM1	R/W	0	该位控制定时器 1 模块的时钟选通。如果该位置位，定时器 1 接收时钟并运行。否则，不使用时钟并禁止。
16	GPTM0	R/W	0	该位控制定时器 0 模块的时钟选通。如果该位置位，定时器 0 接收时钟并运行。否则，不使用时钟并禁止。
15: 5	保留	RO	0	保留位，返回不确定的值，并且应永不改变。
4	SSI	R/W	0	该位控制 SSI 模块的时钟选通。如果该位置位，SSI 接收时钟并运行。否则，不使用时钟并禁止。[a]
3:2	保留	RO	0	保留位，返回不确定的值，并且应永不改变。
1	UART1	R/W	0	该位控制 UART1 模块的时钟选通。如果该位置位，UART1 接收时钟并运行。否则，不使用时钟并禁止。[a]
0	UART0	R/W	0	该位控制 UART0 模块的时钟选通。如果该位置位，UART0 接收时钟并运行。否则，不使用时钟并禁止。[a]

a. 如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。

寄存器 25：运行-模式时钟门控控制 2（RCGC2），偏移量 0x108

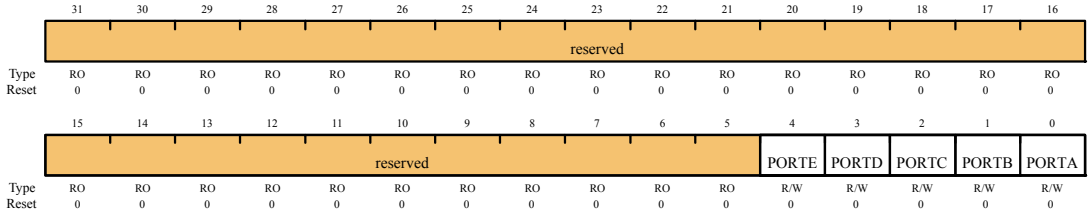
寄存器 26：睡眠-模式时钟门控控制 2（SCGC2），偏移量 0x118

寄存器 27：深度-睡眠-模式时钟门控控制 2（DCGC2），偏移量 0x128

这几个寄存器用来控制时钟门控逻辑。每个位控制一个给定接口、功能、或单元的时钟使能。如果置位，则对应的单元接收时钟并运行。否则，对应的单元不使用时钟并禁止（节能）。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。除非特别说明，否则这些位的复位状态都为 0（不使用时钟），即所有功能单元都禁止。应用所需的端口需通过软件来使能。**注**：这些寄存器除了含有对接口、功能、或单元进行控制的位以外，还可能含有其它位，这样可保证与其它系列以及将来的部件实现合理的代码兼容。

RCGC2 是运行操作，**SCGC2** 是睡眠操作，**DCGC2** 是深度睡眠操作的时钟配置寄存器。将**运行-模式时钟配置（RCC）**寄存器的 ACG 位置位时，系统使用睡眠模式。

运行模式、睡眠模式和深度睡眠模式时钟门控 2 (RCGC2, SCGC2和DCGC2)
Offset 0x108, 0x118, and 0x128



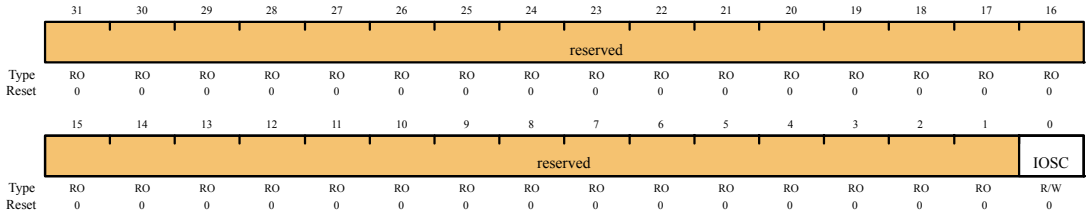
位	名称	类型	复位	描述
31:5	保留	RO	0	保留位，返回不确定的值，并且应永不改变
4	PORTE	R/W	0	该位控制 GPIO 端口 E 模块的时钟选通。如果该位置位，对应单元接收时钟并运行。否则，不使用时钟并禁止。[a]
3	PORTD	R/W	0	该位控制 GPIO 端口 D 模块的时钟选通。如果该位置位，对应单元接收时钟并运行。否则，不使用时钟并禁止。[a]
2	PORTC	R/W	0	该位控制 GPIO 端口 C 模块的时钟选通。如果该位置位，对应单元接收时钟并运行。否则，不使用时钟并禁止。[a]
1	PORTB	R/W	0	该位控制 GPIO 端口 B 模块的时钟选通。如果该位置位，对应单元接收时钟并运行。否则，不使用时钟并禁止。[a]
0	PORTA	R/W	0	该位控制 GPIO 端口 A 模块的时钟选通。如果该位置位，对应单元接收时钟并运行。否则，不使用时钟并禁止。[a]

a. 如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。

寄存器 28：深度睡眠时钟配置（DSLPCLKCFG）寄存器，偏移量 0x144

该寄存器用于在进入深度睡眠模式时自动从主振荡器切换成内部振荡器。系统时钟源默认情况下是主振荡器。在该寄存器置位时，内部振荡器开始上电，主振荡器开始断电。当发生深度-睡眠退出事件时，硬件会将系统时钟变回到刚进入深度-睡眠模式时的时钟源和频率。

深度-睡眠时钟配置(DSLPCLKCFG)寄存器
偏移量 0x144

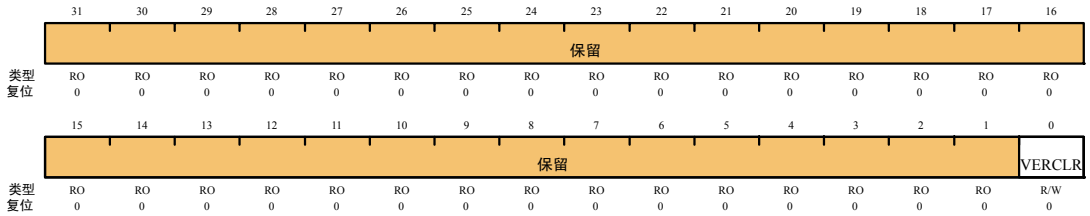


位	名称	类型	复位	描述
31:1	保留	RO	0	保留位，返回不确定的值，并且应永不改变
0	IOSC	R/W	0	该位允许在运行深度-睡眠模式时使主振荡器无效。置位时，该位在深度-睡眠模式期间会将内部振荡器强制变为时钟源。否则，主振荡器将始终作为默认的系统时钟源。

寄存器 29：时钟验证清零（CLKVCLR），偏移量 0x150

该寄存器为使用软件实现时钟验证电路的清零操作提供一种方法。因为时钟验证电路强制一个已知的性能优良的时钟来对处理进行控制，这样控制器有机会来解决问题并清除验证故障。寄存器将所有时钟验证故障清除。为实现这一操作，VERCLR 位必须置位，然后通过软件清零，该位不能自己清零。

时钟验证清零 (CLKVCLR)
偏移量 0x150

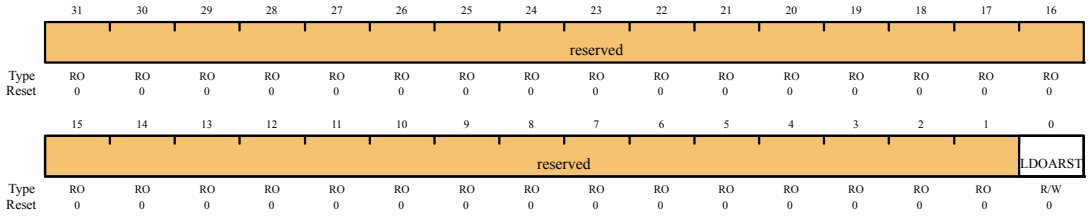


位	名称	类型	复位	描述
31:1	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
0	VERCLR	R/W	0	清除时钟验证故障

寄存器 30：允许不可调整的 LDO 复位部件（LDOARST），偏移量 0x160

该寄存器在电压变为不可调整时提供一种使用 LDO 将部件复位的方法。根据 LDO 脉动的设计容差，当 LDO 变为不可调整时，使用该寄存器来选择是否自动将部件复位。

允许未调整LDO复位器件（LDOARST）寄存器
偏移量 0x160



位	名称	类型	复位	描述
31:1	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
0	LDOARST	R/W	0	该位置位时允许不可调整的 LDO 输出将部件复位

第7章 内部存储器

LM3S617 微控制器带有 8KB 具有“bit-banded”功能的 SRAM 和 32KB 的 Flash 存储器。Flash 控制器提供了一个友好的用户接口，使得 Flash 编程成为一项简单的任务。在 Flash 存储器中可以 2-KB 块大小为基础使用 Flash 保护。

7.1 方框图

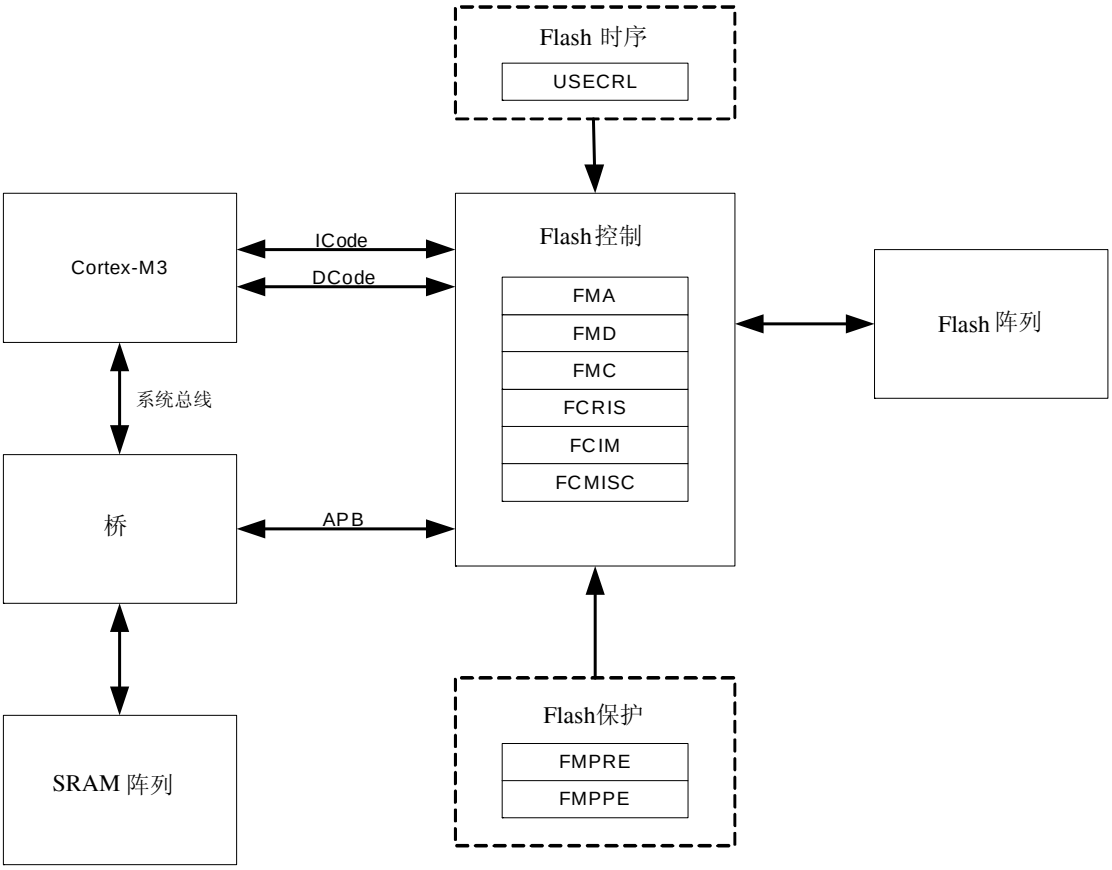


图 7.1 Flash 方框图

7.2 功能描述

这一节描述了两个存储器的功能。

7.2.1 SRAM 存储器

Stellaris 器件的内部 SRAM 位于器件存储器映射地址 0x20000000 处。为了减少执行读－改－写（RMW）操作花费的时间，ARM 在新的 Cortex-M3 处理器中引入了 bit-banding 技术。在使能 bit-banding 的处理器中，当对存储器映射中的特定区域（SRAM 和外设空间）进行单次和原子（atomic）操作时，可以使用地址别名来访问各个位。

bit-band 别名可通过以下等式计算得到：

bit-band 别名 = bit-band 基址 + (字节偏移量 * 32) + (位编号 * 4)

例如，如果要修改地址 0x20001000 的位 3，则 bit-band 别名可通过下列等式计算得到：

0x22000000 + (0x1000 * 32) + (3 * 4) = 0x2202000C

此时别名地址的结果为 0x2202000C，对该地址执行读/写操作的指令仅允许直接访问地址 0x20001000 处字节的位 3。

有关 bit-banding 的详细信息，请参考“ARM® Cortex™-M3 技术参考手册”的第 4 章“存储器映射”。

7.2.2 Flash 存储器

Flash 是由一组可独立擦除的 1KB 区块所构成的。对一个区块进行擦除将使该区块的全部内容复位为 1。这些模块配对后便组成了一组可分别进行保护的 2KB 区块。区块可被标记为只读或只执行(execute-only)，以提供不同级别的代码保护。只读区块不能进行擦除或者编程，以保护区块的内容免受更改。只执行区块不能进行擦除或者编程，而且控制器只能采用取指机制来读取其内容，这可以保护区块的内容，避免它被控制器或调试器读取。

7.2.2.1 Flash 存储器时序

Flash 的时序是由 Flash 控制器自动处理的。但如此便需要知道系统的时钟速率以便对内部的信号进行精确地计时。为了完成这种计时，必须向 Flash 控制器提供每微秒所占用的时钟周期数目。而将此信息通过 **USec 重载**（**USECRL**）寄存器传达到 Flash 控制器以保持其更新状态的工作则由软件来负责实现。

复位时，有一个值会被载入到 **USECRL** 寄存器中，该值会对 Flash 的时序进行配置，使得它能够在所选的晶振频率下工作。如果软件想改变系统操作频率，那么在试图对 Flash 进行任何修改之前必须将新的操作频率载入到 **USECRL** 中。例如，如果器件此时的工作频率为 20MHz，那么必须在 **USECRL** 寄存器中写入值 0x13。

7.2.2.2 Flash 存储器保护

在两个 32 位宽的寄存器中，以 2KB Flash 块为基础向用户提供两种形式的 Flash 保护。每种形式的保护策略均由**FMPPE** 和**FMPRE** 寄存器的各个位来控制(每个块有一个策略)。

- **Flash 存储器保护编程使能（FMPPE）**：如果置位，则可以对模块进行编程（写）或擦除。如果清零，则不可以改变模块。
- **Flash 存储器保护读使能（FMPRE）**：如果置位，则通过软件或调试器执行或者读取模块。如果清零，则只可以执行模块。存储器模块的内容禁止作为数据来访问，并且也不能经过 DCode 总线。

可以将这些策略进行组合，如 表 7.1 所示。

表 7.1 Flash 保护策略组合

FMPPE	FMPRE	保护
0	0	只执行保护。只可以执行模块而不能对其进行写或擦除。使用该模式来保护代码。
1	0	可以写、擦除或执行模块，但不能进行读取。这种组合不常用。
0	1	只读保护。可以读或执行模块，但不能对其进行写或擦除。在允许任何读或执行访问时，使用该模式来锁定模块以防对其进行更多的修改。
1	1	无保护。可对模块进行写入、擦除、执行或读取操作。

试图对受到擦写保护的模块进行编程或者擦除的访问操作是被禁止的。可选择产生控

制器中断（通过置位 **FIM** 寄存器中的 AMASK 位）来向软件开发者警报在开发和调试阶段中出现的错误软件操作。

试图对受到读保护的模块进行读取的访问操作是被禁止的。此类访问所返回的数据将全部为 0。可选择产生控制器中断来向软件开发者警报在开发和调试阶段中出现的错误软件操作。

在 **FMPRE** 和 **FMPPE** 寄存器的出厂设置中，所有已经实现的存储器组所对应的位的值为 1。这实现了一种带有开放式访问和可编程特性的策略。寄存器的位可通过写特定的寄存器位来改变。而这种改变不是永久性的，除非寄存器已获确认(已保存)，那时，位的改变才是永久性的。如果位在从 1 变为 0 时没有被确认，可以通过执行上电复位序列来恢复该位。

7.2.2.3 Flash 存储器编程

写 Flash 存储器要求在 SRAM 之外执行写入的代码以避免破坏或打断总线时序。Flash 页面的擦除是以页面为单位（每页 1KB 大小）来进行的，也可以通过完全擦除整个 Flash 来完成。

所有的擦除和编程操作都是通过使用**Flash存储器地址(FMA)、Flash存储器数据(FMD)和Flash存储器控制(FMC)**寄存器来执行的。例子见 7.3 小节。

7.3 初始化和配置

本小节给出的是一个使用 flash 控制器对 flash 存储器的内容进行各种操作的实例。

7.3.1 改变 Flash 保护位

如在 7.2.2.2 中所述，在保护位生效之前必须确认对这些位的改变。改变和确认一个位的顺序如下：

1. 写 **Flash 存储器保护读使能(FMPRE)寄存器**和 **Flash 存储器保护编程使能(FMPPE)** 寄存器，改变需要修改的位（intended bit）。在该状态下可通过软件对这些更改的举动进行测试。
2. 如果需要确认 **FMPPE** 寄存器，那么把 **Flash 存储器地址(FMA)**寄存器的位 0 设为 1；否则，当位 0 被设为 0 时会确认 **FMPRE** 寄存器。
3. 写 **Flash 存储器控制(FMC)**寄存器，将 COMT 位置位。该操作启动一个写序列并确认改变。

7.3.2 Flash 编程

Stellaris 器件为 Flash 编程提供了一个友好的用户接口。通过 3 个寄存器 **FMA**、**FMD** 和 **FMC** 来处理所有擦除/编程操作。

按照下列次序来对 Flash 进行编程：

1. 把源数据写入 **FMD** 寄存器。
2. 把目标地址写入 **FMA** 寄存器。
3. 把 flash 写入密钥（flash write key）写入 **FMC** 寄存器,并将 WRITE 位置位。（写入 0xA4420001）。

4. 查询 **FMC** 寄存器直至 WRITE 位被清零。

按照下列顺序执行 1-KB 页的擦除：

1. 将页地址写入 **FMA** 寄存器
2. 将 Flash 写入密钥（flash write key）写入 **FMC** 寄存器，并将 ERASE 位置位（写入 0xA4420002）。
3. 查询 **FMC** 寄存器直至 ERASE 位被清零。

按照下列顺序要执行 Flash 的完全擦除：

1. 将 Flash 写入密钥（flash write key）写入 **FMC** 寄存器，并将 MERASE 位置位（写入 0xA4420004）。
2. 查询 **FMC** 寄存器直至 MERASE 位被清零。

7.4 寄存器映射

表 7.2 列出的是Flash存储器和控制寄存器。表中所列偏移量是寄存器地址相对Flash控制基址 0x400FD000 的十六进制增量；但**FMPRE**和**FMPPE**除外，与**FMPRE**和**FMPPE**相对应的系统控制基址为 0x400FE000。

表 7.2 Flash 寄存器映射

偏移量	名称	复位	类型	描述
0x130 [a]	FMPRE	0xFFFF	R/W0	Flash 存储器读保护
0x134 [a]	FMPPE	0xFFFF	R/W0	Flash 存储器编程保护
0x140 [a]	USECRL	0x31	R/W	USec 重装
0x000	FMA	0x00000000	R/W	Flash 存储器地址
0x004	FMD	0x00000000	R/W	Flash 存储器数据
0x008	FMC	0x00000000	R/W	Flash 存储器控制
0x00C	FCRIS	0x00000000	RO	Flash 控制器原始中断状态
0x010	FCIM	0x00000000	R/W	Flash 控制器中断屏蔽
0x014	FCMISC	0x00000000	R/W1C	Flash 控制器屏蔽后的中断状态和中断状态的清除

a. 对应的系统控制基址为 0x400FE000。

7.5 寄存器描述

下文将按地址偏移量的数字顺序列出 Flash 存储器寄存器，并对它们进行描述。

寄存器 1：Flash 存储器保护读使能(FMPRE)，偏移量 0x130

寄存器 2：Flash 存储器保护编程使能(FMPPE)，偏移量 0x134

注：偏移量是相对于 0x400FE000 的系统控制基址而言。

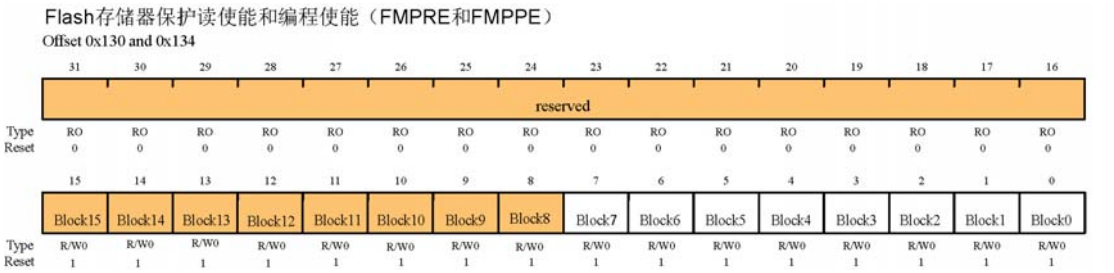
这些寄存器所保存的是每个 2KB flash 块的只读（**FMPRE**）和只执行（**FMPPE**）保护位。该寄存器会在上电复位序列期间载入。

在 **FMPRE** 和 **FMPPE** 寄存器的出厂设置中，所有已经实现的存储器组所对应的位的值为 1。这实现了一种带有开放式访问和可编程特性的策略。寄存器的位可通过写入特定

的寄存器位来改变。但是，该寄存器为 R/W0；用户只能将保护位从 1 变为 0（而不能将该位从 0 变为 1）。

但这种改变并不是永久性的，除非寄存器已获确认（已保存），那时，位的改变才是永久性的。如果位在从 1 变为 0 时没有确认，可以通过执行上电复位序列来恢复该位。

要了解更多信息，请见 7.2.2.2 节 “Flash存储器保护”。

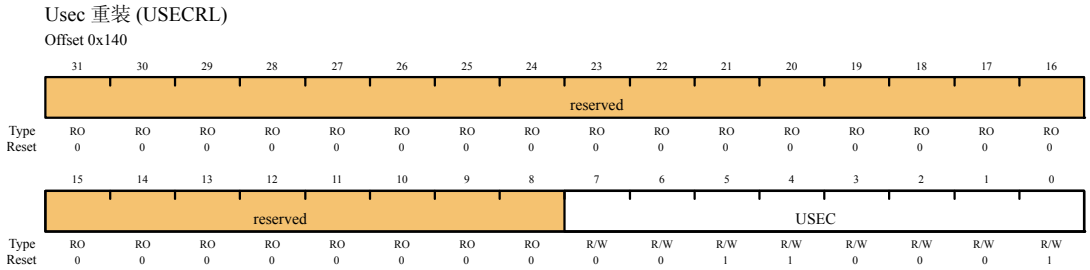


位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
15:0	模块 15-模块 0	R/W0	1	可以对 2KB Flash块进行写或擦除操作（**FMPRE**寄存器），或者对它进行执行或读取操作（**FMPPE**寄存器）。可以将保护的策略进行组合，如 表 7.1 所示。

寄存器 3：USec 重装（USECRL），偏移量 0x140

注：偏移量所对应的系统控制基址为 0x400FE000。

该寄存器所提供的是为 Flash 控制器产生 1μs 节拍分频器的重装值的一种方法。对于高电平写入脉冲能够生效的持续时间，内部 flash 对最大和最小值具有特殊要求。它要求无论何时对 Flash 进行擦除或者编程操作，该寄存器都能够包含工作频率的值（MHz-1）。如果 Flash 擦除/编程操作的时钟条件发生改变，那么用户也需要改变该值。



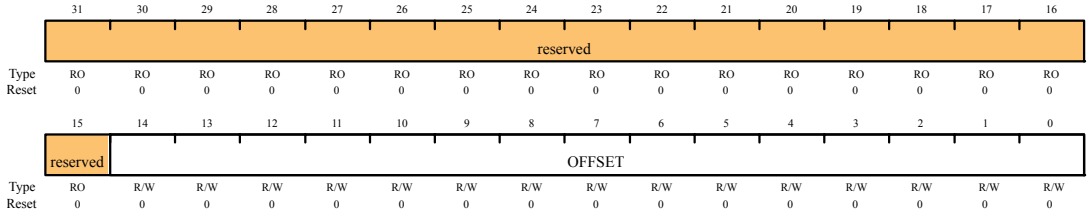
位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	USEC	R/W	0x31	当 Flash 被擦除或编程时，控制器时钟的频率为 MHz-1。 无论何时对 Flash 存储器进行擦除或者编程操作，都应把 USEC 的值相应的设为 0x31(49 MHz)。

寄存器 4：Flash 存储器地址（FMA），偏移量 0x000

在写操作过程中，该寄存器带有一个以 4 个字节-为单位对齐的地址并指定在哪里写入数据。在擦除操作过程中，该寄存器带有一个以 1KB-为单位对齐的地址并指定哪一页被擦除。注意必须要符合（地址）对齐的要求，否则操作的结果将不可预知。

Flash 存储器地址 (FMA)

Offset 0x000



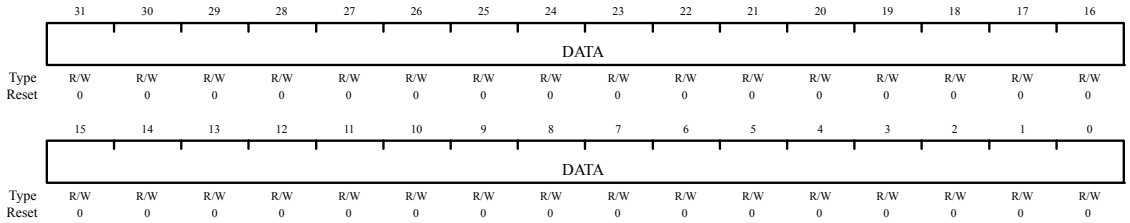
位/字段	名称	类型	复位	描述
31: 15	保留	RO	0x0	保留位返回一个不确定的值，并且应永不改变。
14:0	OFFSET	R/W	0x0	Flash 中执行操作处的地址偏移量。

寄存器 5：Flash 存储器数据（FMD），偏移量 0x004

该寄存器所存放的是编程周期中被写入的数据或在读周期中被读取的数据。需要注意的是，对于只执行模块的读取访问来说，该寄存器的内容是未定义的。在擦除的周期中并不使用该寄存器。

Flash 存储器数据 (FMD)

偏移量 0x004



位/字段	名称	类型	复位	描述
31:0	DATA	R/W	0x0	写操作的数据值。

寄存器 6：Flash 存储器控制（FMC），偏移量 0x008

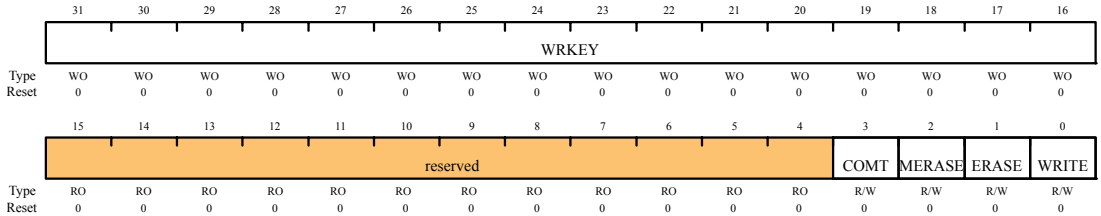
当该寄存器被写入的时候，Flash 控制器将对 **Flash 存储器地址(FMA)**中指定的位置发起适当周期数目的访问操作。如果访问是写入访问，那么将会写入 **Flash 存储器数据(FMD)** 寄存器中保存的数据。

这是写入的最后一个寄存器，并将启动存储器操作。在该寄存器的低位字节中有 4 个控制位，当这些位被置位时将启动存储器操作。这些寄存器中经常使用的位为 ERASE 和 WRITE 位。

写入多个控制位是一种编程错误，而且这种操作的结果是无法预知的。

Flash 存储器控制 (FMC)

偏移量 0x008

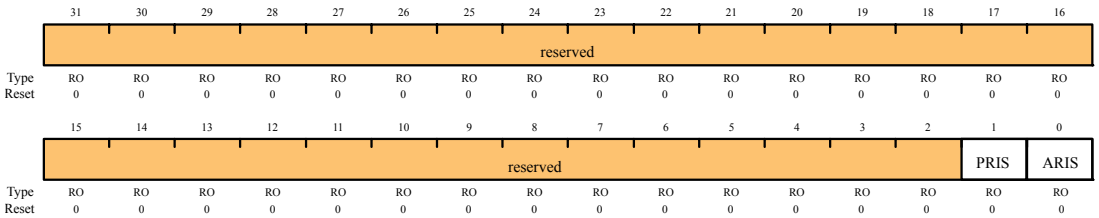


位/字段	名称	类型	复位	描述
31:16	WRKEY	WO	0x0	该字段包含了一个 write key（写入密钥），最大限度地缩小了意外 Flash 写入操作的范围。必须向该字段写入 0xA442 的值以便发出写入操作。若写入 **FMC** 寄存器的值不带 WRKEY，那么该值将被忽略掉。读取该字段将返回 0。
15:4	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
3	COMT	R/W	0	向非易失性存储器确认（写）寄存器的值。写 0 对该位的状态无影响。 如果执行读操作，则提供先前所确认的访问状态。如果先前的确认访问完成，则返回 0；否则，如果确认访问没有完成，则返回 1。 该操作所需的时间最多可达 50μs。
2	MERASE	R/W	0	海量（mass）擦除 Flash 存储器。 如果该位置位，则器件的 Flash 主存储器的内容将被全部擦除。写 0 对该位的状态没有影响。 如果执行读操作，则提供先前完全擦除访问的状态。如果先前完全擦除的访问已完成，则返回 0；否则，如果先前的访问没有完成，则返回 1。 该操作所需的时间最多可达 250ms。
1	ERASE	R/W	0	擦除 Flash 存储器的页。 如果该位置位，则擦除 **FMA** 内容所指定的 Flash 主存储器页。写 0 对该位的状态没有影响。 如果执行读操作，则提供先前擦除访问的状态。如果先前的擦除访问已完成，则返回 0；否则，如果先前的访问还没有完成，则返回 1。 该操作所需的时间最多可达 25ms。
0	WRITE	R/W	0	写一个字到 Flash 存储器。 如果该位置位，则 **FMD** 中存储的数据被写到 **FMA** 内容所指定的位置。写 0 对该位的状态没有影响。 如果执行读操作，则提供先前写更新的状态。如果先前的写访问已完成，则返回 0；否则，如果写访问还没有完成，则返回 1。 该操作所需的时间最多可达 50μs。

寄存器 7：Flash 控制器原始中断状态（FCRIS），偏移量 0x00C

该寄存器指示 Flash 控制器的中断状况。而一个中断信号只有在其相应的 **FCIM** 寄存器位被置位时才能发出。

Flash 控制器原始中断状态 (FCRIS)
偏移量0x00C

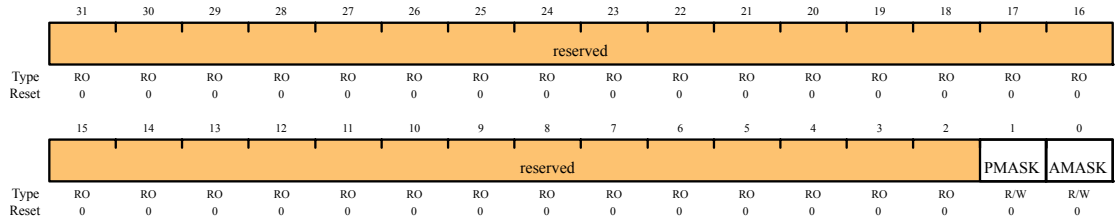


位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
1	PRIS	RO	0	编程的原始中断状态。 该位表示编程周期的当前状态。如果置位，则编程周期完成；如果清零，则编程周期还没有完成。编程周期是指通过 **Flash 存储器控制(FMC)**寄存器位产生的写或擦除操作。
0	ARIS	RO	0	访问的原始中断状态。 该位指示 Flash 是否被不正确地访问。如果置位，则表示程序试图访问 Flash 的操作与 **Flash 存储器保护读使能(FMPRE)**和 **Flash 存储器保护编程使能(FMPPE)**寄存器中设置的策略相反。否则，没有试图错误访问 Flash 的情况出现。

寄存器 8：Flash 控制器中断屏蔽（FCIM），偏移量 0x010

该寄存器控制 Flash 控制器是否向控制器产生中断。

Flash 控制器中断屏蔽 (FCIM)
偏移量 0x010

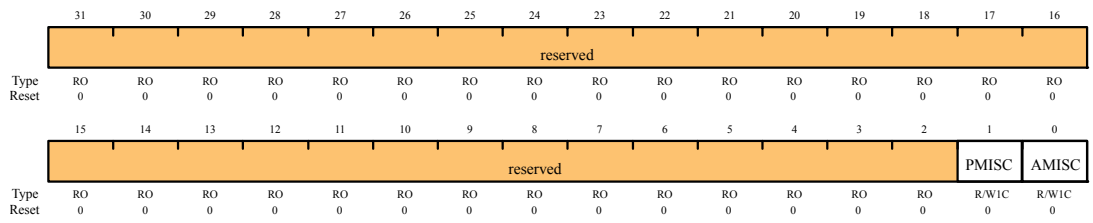


位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
1	PMASK	R/W	0	编程中断屏蔽。 该位控制把编程原始中断状态报告到控制器的过程。如果置位，则向控制器提交编程所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。
0	AMASK	R/W	0	访问中断屏蔽。 该位控制把访问原始中断状态报告到控制器的过程。如果置位，则向控制器提交访问所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。

寄存器 9：Flash 控制器屏蔽后的中断状态和清除(FCMISC)，偏移量 0x014

该寄存器提供了两个功能。首先，它可以指示出哪个或者哪几个中断源正在发出中断信号，由此报告出中断产生的原因。其次，还可以将它作为清除中断报告的一种方法来使用。

Flash 控制器屏蔽后的中断状态和清除 (FCMISC)
偏移量 0x014



位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
1	PMISC	R/W1C	0	屏蔽后编程中断的状态和清除。 该位指示是否由于编程周期结束且未被屏蔽而发出中断信号。该位通过写 1 来清零。当清零 PMISC 位时，**FCRIS** 寄存器中的 PRIS 位也被清零。
0	AMISC	R/W1C	0	屏蔽后访问中断的状态和清除。 该位指示是否由于试图进行错误地访问且未被屏蔽而发出中断信号。该位通过写 1 来清零。当清零 AMISC 位时，**FCRIS** 寄存器中的 ARIS 位也被清零。

第8章 通用输入/输出（GPIO）

GPIO 模块由 5 个物理 GPIO 块组成，一块对应一个 GPIO 端口（PA，PB，PC，PD 和 PE 口）。GPIO 模块遵循 FiRM 规范，并且支持 1～30 个可编程的输入/输出管脚，具体取决于正在使用的外设。GPIO 模块包含以下特性：

- 可编程控制 GPIO 中断
  - 屏蔽中断发生
  - 边沿触发（上升沿，下降沿，上升/下降沿）
  - （高或低）电平触发
- 输入/输出端口可承受 5V 电压
- 在读和写操作中通过地址线进行位屏蔽
- 可编程控制 GPIO 引脚（pad）配置：
  - 弱上拉或下拉电阻
  - 2-mA、4-mA 和 8-mA 的引脚（pad）驱动
  - 8-mA 驱动的斜率控制
  - 开漏使能
  - 数字输入使能

8.1 方框图

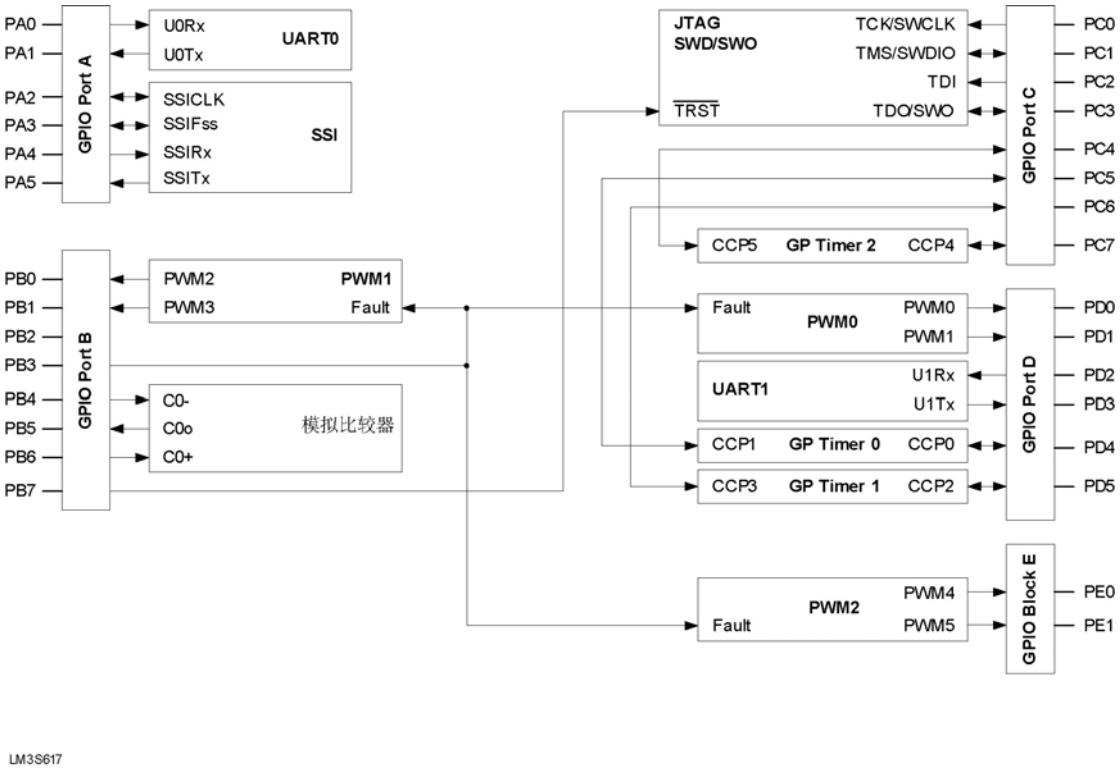


图 8.1 GPIO 模块结构图

8.2 功能描述

要点：除了 5 个 JTAG 管脚（PB7 和 PC[3:0]）之外，所有 GPIO 管脚默认下都是输入管脚（**GPIODIR**=0 且 **GPIOAFSEL**=0）。JTAG 管脚默认为 JTAG 功能（**GPIOAFSEL**＝1）。通过上电复位（**POR**）或外部复位（$\overline{\text{RST}}$）可以让这两组管脚都回到其默认状态。

每个GPIO端口都是同一物理块中（见 图 8.2）的独立硬件的实例化（hardware instantiation）。LM3S617 微控制器含 5 个端口和 5 个 图 8.2 中的物理GPIO块。

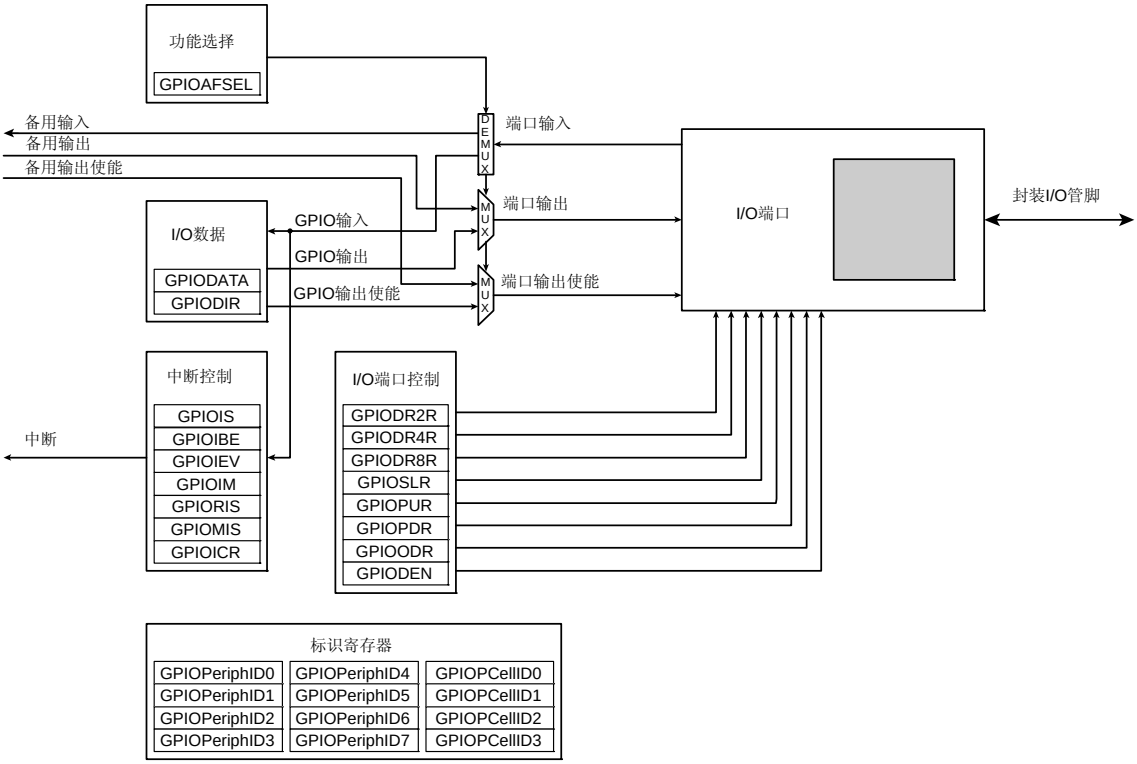


图 8.2 GPIO 端口结构图

8.2.1 数据寄存器操作

为了提高软件的效率，通过将地址总线的位[9:2]用作屏蔽位，GPIO 端口允许对 **GPIO 数据（GPIODATA）**寄存器中的各个位进行修改。这样，软件驱动程序仅使用一条指令就可以对各个 GPIO 管脚进行修改，而不会影响其他管脚的状态。这点与通过执行读-修改-写操作来置位或清零单独的 GPIO 管脚的“典型”做法不同。为了提供这种特性，**GPIODATA** 寄存器包含了存储器映射中 256 个单元。

在写操作过程中，如果与数据位相关联的地址位被设为 1，那么 **GPIODATA** 寄存器的值将发生变化。如果被清零，那么 GPIODATA 的值将保持不变。

例如，将 0xEB写入地址GPIODATA+0x098 处，结果如 图 8.3 所示。图中，u表示没有被写操作改变的数据。

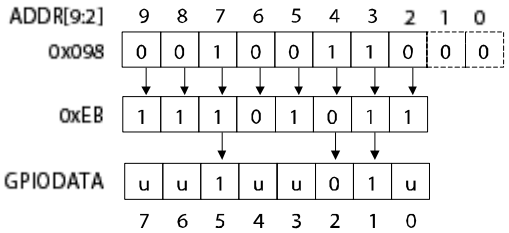


图 8.3 “GPIODATA 写”实例

在读操作过程中，如果与数据位相关联的地址位被设为 1，那么读取该值。如果与数据位相关联的地址位被设为 0，那么不管它的实际值是什么，都将该值读作 0。例如，读取地址GPIODATA+0x0C4 处的值，结果如 图 8.4 所示。

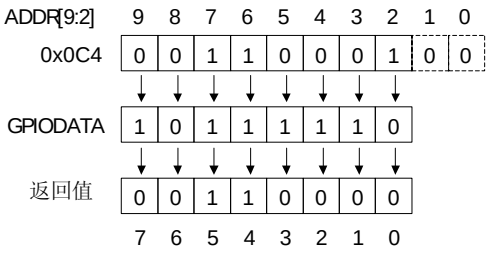


图 8.4 “GPIODATA 读”实例

8.2.2 数据方向

通过 **GPIO 方向（GPIODIR）**寄存器，将各个位配置成输入或输出。

8.2.3 中断操作

每个 GPIO 端口的中断能力都由 7 个一组的寄存器控制。通过这些寄存器可以选择中断源、中断极性以及边沿属性。当一个或多于一个 GPIO 输入产生中断时，只将一个中断输出发送到供所有 GPIO 端口使用的中断控制器。对于边沿触发中断，为了使能其他中断，软件必须清除该中断。对于电平触发中断，假设外部源保持电平不发生变化，以便中断能被控制器识别。

使用 3 个寄存器来对产生中断的边沿或触发信号进行定义：

- **GPIO 中断检测（GPIOIS）**寄存器
- **GPIO 中断双边沿（GPIOIBE）**寄存器
- **GPIO 中断事件（GPIOIEV）**寄存器

通过 **GPIO 中断屏蔽（GPIOM）**寄存器可以使能或禁能中断。当产生中断条件时，可以在 **GPIO 原始（raw）中断状态（GPIORIS）**寄存器和 **GPIO 屏蔽后中断状态（GPIOMIS）**寄存器两个地方观察到中断信号的状态。顾名思义，**GPIOMIS** 寄存器仅显示允许被传送到控制器的中断条件。而 **GPIORIS** 寄存器则表示 GPIO 管脚满足中断条件，但是没有必要发送到控制器。

PB4 除了可以提供 GPIO 功能外，还可以用作 ADC 的外部触发器。如果将 PB4 配置成一个不可屏蔽中断管脚（**GPIOIM** 设为 1），那么 PB 将会产生中断，并且会将外部触发信号发送给 ADC。如果 **ADC 事件多路复用器选择（ADCEMUX）**寄存器被配置成使用外部触发器，那么将启动 ADC 转换。

如果不使用其他 PB 管脚来产生中断，那么 ARM 集成的嵌套向量中断控制器（NVIC）中断设置使能（SETNA）寄存器可以将 PB 中断禁能，且可以使用 ADC 中断来读回被转换的数据。否则，PB 中断处理器必须在 B4 上忽略和清除中断，并且要等待 ADC 中断，或者必需在 SETNA 寄存器中禁能 ADC 中断。此外，PB 中断处理器还会轮询 ADC 寄存器直至转换结束。

写 1 到 **GPIO 中断清除（GPIOICR）**寄存器可以清除中断。

在对中断进行编程时，应该屏蔽中断（将 **GPIOIM** 设为 0）。如果相应的位被使能，那么向中断控制寄存器（**GPIOIS**，**GPIOIBE** 或 **GPIOIEV**）写入任意值都有可能产生伪中断。

8.2.4 模式控制

GPIO 管脚可以由硬件或软件控制。当通过 **GPIO 备用（Alternate）功能选择（GPIOAFSEL）**寄存器将硬件控制使能时，管脚状态将由它备用的（Alternate）功能（即外设）控制。软件控制相当于 GPIO 模式，在该模式下，**GPIODATA** 寄存器用来读/写相应的管脚。

8.2.5 引脚（pad）配置

引脚配置寄存器使软件能够根据应用的要求来进行 GPIO 引脚配置。引脚配置寄存器包括：**GPIODR2R**、**GPIODR4R**、**GPIODR8R**、**GPIOODR**、**GPIOPUR**、**GPIOPDR**、**GPIOSLR** 和 **GPIODEN** 寄存器。

8.2.6 标识（identification）

复位时配置的标识寄存器允许软件将模块当作 GPIO 块进行检测和识别。标识寄存器包括 **GPIOPeriphID0～GPIOPeriphID7** 寄存器和 **GPIOPCellID0－GPIOPCellID3** 寄存器。

8.3 初始化和配置

为了使用 GPIO，外设时钟必须通过置位 RCGC2 寄存器的 PORTA、PORTB、PORTC、PORTD 和 PORTE 来使能。

复位后，所有GPIO管脚（5 个JTAG管脚除外）都将默认为通用输入模式（**GPIODIR**和**GPIOAFSEL**都设为 0）。表 8.1 列出了GPIO端口的所有可能的配置以及为实现这些配置所要求的控制寄存器的设置。表 8.2 给出了为GPIO端口的管脚 2 配置上升沿中断的方法。

表 8.1 GPIO 端口的配置实例

配置	寄存器位的值 [a]									
	GPIOAFSEL	GPIODIR	GPIOODR	GPIODEN	GPIOPUR	GPIOPDR	GPIODR2R	GPIODR4R	GPIODR8R	GPIOSLR
数字输入（GPIO）	0	0	0	1	?	?	X	X	X	X
数字输出（GPIO）	0	1	0	1	?	?	?	?	?	?
开漏输入（GPIO）	0	0	1	1	X	X	X	X	X	X
开漏输出（GPIO）	0	1	1	1	X	X	?	?	?	?
数字输入（定时器 CCP）	1	X	0	1	?	?	X	X	X	X
数字输出（PWM）	1	X	0	1	?	?	?	?	?	?
数字输出（定时器 PWM）	1	X	0	1	?	?	?	?	?	?
数字输入/输出（SSI）	1	X	0	1	?	?	?	?	?	?
数字输入/输出（UART）	1	X	0	1	?	?	?	?	?	?
模拟输入（比较器）	0	0	0	0	0	0	X	X	X	X
数字输出（比较器）	1	X	0	1	?	?	?	?	?	?

a. X＝忽略（无关位）；

？＝可以是 0 或 1，具体取决于配置。

表 8.2 GPIO 中断配置实例

寄存器	期望的中断事件触发	管脚 2 各位的值[a]							
		7	6	5	4	3	2	1	0
GPIOIS	0＝边沿 1＝电平	X	X	X	X	X	0	X	X
GPIOIBE	0＝单边沿 1＝双边沿	X	X	X	X	X	0	X	X
GPIOIEV	0＝低电平，或负边沿； 1＝高电平，或正边沿	X	X	X	X	X	1	X	X
GPIOIM	0＝屏蔽 1＝不屏蔽	0	0	0	0	0	1	0	0

a. X＝忽略（无关位）

8.4 寄存器映射

表 8.2 列出了GPIO寄存器。所列偏移量是寄存器相对于GPIO端口基址的 16 进制增量。GPIO端口的基址如下：

- GPIO 端口 A（PA）：0x40004000
- GPIO 端口 B（PB）：0x40005000
- GPIO 端口 C（PC）：0x40006000
- GPIO 端口 D（PD）：0x40007000
- GPIO 端口 E（PE）：0x40024000

要点：本章的GPIO寄存器在每个GPIO块中都是相同的，但是根据块的不同，8 个位可能并不是全部与GPIO端口相连，（见 图 8.1）。在那些情况下，向未连接的位进行写操作是没有作用的，且读取这些未连接的位时返回的也是无用的数据。

表 8.3 GPIO 寄存器映射

偏移	名称	复位	类型	描述
0x000	GPIODATA	0x00000000	R/W	数据
0x400	GPIODIR	0x00000000	R/W	数据方向
0x404	GPIOIS	0x00000000	R/W	中断检测（sense）
0x408	GPIOIBE	0x00000000	R/W	中断双边沿
0x40C	GPIOIEV	0x00000000	R/W	中断事件
0x410	GPIOIM	0x00000000	R/W	中断屏蔽使能
0x414	GPIORIS	0x00000000	RO	原始（raw）中断状态
0x418	GPIOMIS	0x00000000	RO	屏蔽后（masked）的中断状态
0x41C	GPIOICR	0x00000000	W1C	中断清除
0x420	GPIOAFSEL	见注释[a]	R/W	备用（Alternate）功能选择

续上表

偏移	名称	复位	类型	描述
0x500	GPIODR2R	0x000000FF	R/W	2-mA 驱动选择
0x504	GPIODR4R	0x00000000	R/W	4-mA 驱动选择
0x508	GPIODR8R	0x00000000	R/W	8-mA 驱动选择
0x50C	GPIOODR	0x00000000	R/W	开漏选择
0x510	GPIOPUR	0x000000FF	R/W	上拉选择
0x514	GPIOPDR	0x00000000	R/W	下拉选择
0x518	GPIOSLR	0x00000000	R/W	斜率控制选择
0x51C	GPIODEN	0x000000FF	R/W	数字输入使能
0xFD0	GPIOPeriphID4	0x00000000	RO	外设标识 4
0xFD4	GPIOPeriphID5	0x00000000	RO	外设标识 5
0xFD8	GPIOPeriphID6	0x00000000	RO	外设标识 6
0xFDC	GPIOPeriphID7	0x00000000	RO	外设标识 7
0xFE0	GPIOPeriphID0	0x00000061	RO	外设标识 0
0xFE4	GPIOPeriphID1	0x00000000	RO	外设标识 1
0xFE8	GPIOPeriphID2	0x00000018	RO	外设标识 2
0xFEC	GPIOPeriphID3	0x00000001	RO	外设标识 3
0xFF0	GPIOPCellID0	0x0000000D	RO	GPIO PrimeCelle 标识 0
0xFF4	GPIOPCellID1	0x000000F0	RO	GPIO PrimeCell 标识 1
0xFF8	GPIOPCellID2	0x00000005	RO	GPIO PrimeCell 标识 2
0xFFC	GPIOPCellID3	0x000000B1	RO	GPIO PrimeCell 标识 3

a. 对于除 5 个 JTAG 管脚（PB7 和 PC[3:0]）之外的所有 GPIO 管脚，**GPIOAFSEL** 寄存器默认的复位值都是 0x00000000。那 5 个管脚默认为 JTAG 功能。正因为这样，对于 GPIO 端口 B（PB），**GPIOAFSEL** 默认的复位值是 0x00000080，而对于端口 C（PC），**GPIOAFSEL** 默认的复位值是 0x0000000F。

8.5 寄存器描述

下文将按地址偏移量的数字顺序列出 GPIO 寄存器，并对它们进行描述。

寄存器 1：GPIO 数据（GPIODATA）寄存器，偏移量：0x000

GPIODATA 寄存器是数据寄存器。在软件控制模式中，如果通过 **GPIO 方向（GPIODIR）**寄存器将各个管脚配置成输出，那么写入 **GPIODATA** 寄存器的值将被发送到 GPIO 端口管脚。

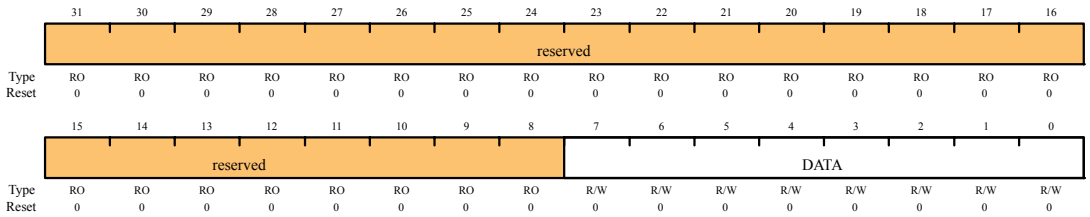
为了对 **GPIODATA** 寄存器执行写操作，由地址总线位[9:2]产生的相关屏蔽位必须为“1”。否则，该位的值不会被写操作改变。

同样，从该寄存器读取的值由也会根据从访问数据寄存器的地址处获取的屏蔽位[9:2]

的情况来决定。如果地址屏蔽位为 1，那么读取 **GPIODATA** 中相应位的值；如果地址屏蔽位为 0，那么不管 **GPIODATA** 中相应位的值是什么，都会将它们读作 0。

如果各自的管脚被配置成输出，读取 **GPIODATA** 将返回最后写入的位值；或者当这些管脚被配置成输入时，将返回相应的输入管脚上的值。所有的位都被复位清零。

GPIO数据（GPIODATA）寄存器
偏移量 0x000

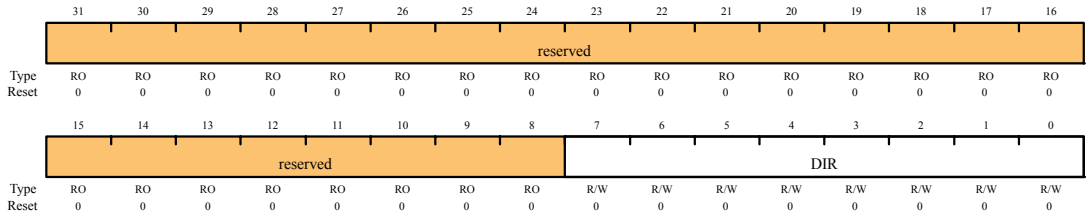


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DATA	R/W	0	GPIO 数据 该寄存器被虚拟地映射到地址空间的 256 个单元中。为便于通过单独的驱动器读写这些寄存器，从这些寄存器读取的值和写入这些寄存器的值都被 8 条地址线ipaddr[9:2]屏蔽。读取该寄存器将返回其当前状态。写入该寄存器仅会影响那些没有被ipaddr[9:2]屏蔽的位和被配置成输出的位。关于读写操作的实例，请见“数据寄存器操作”。

寄存器 2：GPIO 方向（GPIODIR）寄存器，偏移量：0x400

GPIODIR 寄存器是数据方向寄存器。**GPIODIR** 寄存器中设为 1 的位将相应的管脚配置成输出，而 **GPIODIR** 寄存器中设为 0 的位将相应的管脚配置成输入。所有位在复位时都会被清零，即默认情况下所有 GPIO 管脚都是输入。

GPIO 方向 (GPIODIR)寄存器
偏移量 0x400



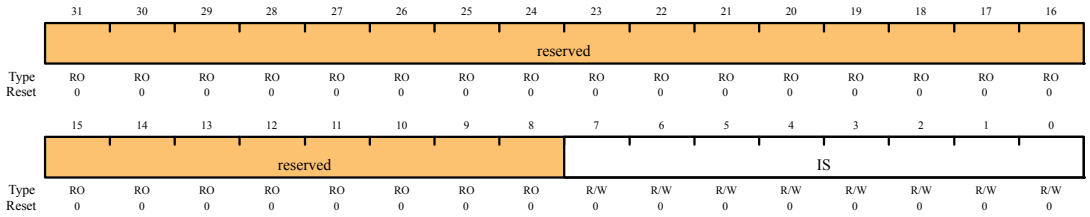
位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DIR	R/W	0x00	GPIO 数据方向 0：管脚为输入 1：管脚为输出

寄存器 3：GPIO 中断检测（sense）（GPIOIS）寄存器，偏移量：0x404

GPIOIS 寄存器是中断检测（sense）寄存器。**GPIOIS** 寄存器中设为 1 的位将相应的管脚配置成检测电平，而 **GPIOIS** 寄存器中设为 0 的位将相应的管脚配置成检测边沿。所

有位在复位时都会被清零。

GPIO中断检测（GPIOIS）寄存器
偏移量 0x404

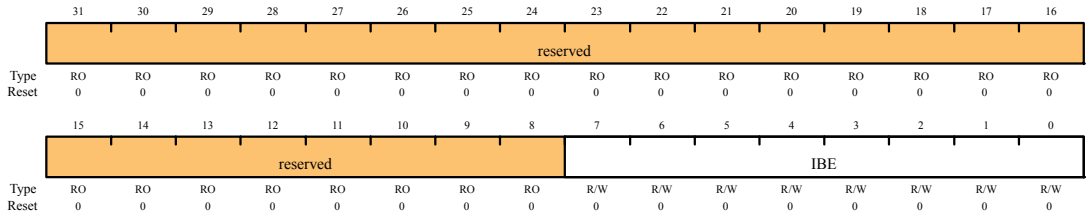


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	IS	R/W	0x00	GPIO 中断检测 0：检测的是相关管脚的边沿（边沿触发） 1：检测的是相关管脚的电平（电平触发）

寄存器 4：GPIO 中断双边沿（GPIOIBE）寄存器，偏移量：0x408

GPIOIBE 寄存器是中断双边沿寄存器。当 **GPIO 中断检测（GPIOIS）**寄存器中相应的位被设置成检测边沿时，**GPIOIBE** 中被设为“1”的位将相应的管脚配置成检测上升沿和下降沿，而不用考虑 **GPIO 中断事件（GPIOIEV）**寄存器的相应位。将位清零会将该管脚配置成由 **GPIOIEV** 控制。所有位在复位时都会被清零。

GPIO中断双边沿（GPIOIBE）寄存器
偏移量 0x408

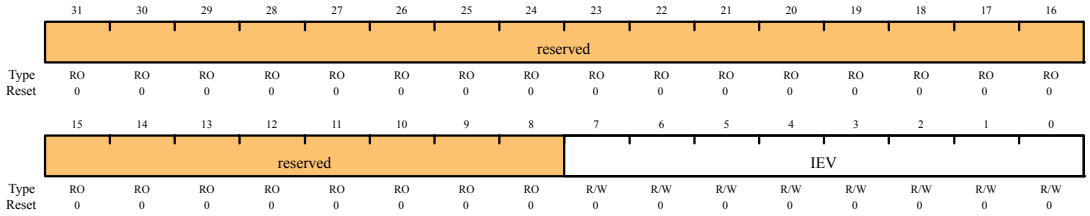


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	IBE	R/W	0x00	GPIO 中断双边沿 0：由 **GPIO 中断事件（GPIOIEV）**寄存器控制是否产生中断 1：相应管脚的上升沿和下降沿都会触发中断 **注**：单边沿由 **GPIOIEV** 中相应的位来决定

寄存器 5：GPIO 中断事件（GPIOIEV）寄存器，偏移量：0x40C

GPIOIEV 寄存器是中断事件寄存器。**GPIOIEV** 中设为“1”的位将相应的管脚配置成检测上升沿或高电平，具体取决于 **GPIO 中断检测（GPIOIS）**寄存器中相应位的值。如果位被清零，会将管脚配置成检测下降沿或低电平，具体取决于 **GPIOIS** 中相应位的值。所有位在复位时都会被清零。

GPIO中断事件（GPIOIEV）寄存器
偏移量 0x40C

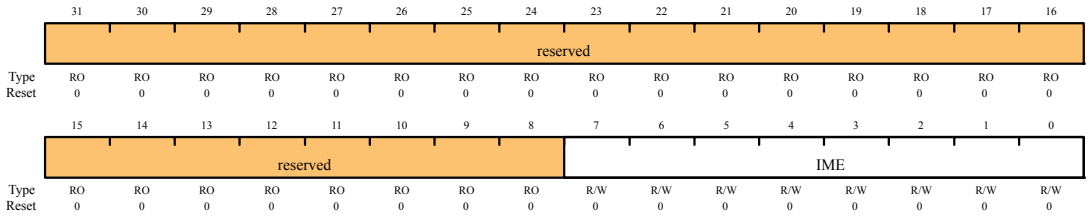


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	IEV	R/W	0x00	GPIO 中断事件 0：相应管脚上的下降沿或低电平触发中断 1：相应管脚的上升沿或高电平触发中断

寄存器 6：GPIO 中断屏蔽（GPIOIM）寄存器，偏移量：0x410

GPIOIM 寄存器是中断屏蔽寄存器。把 **GPIOIM** 中的位设为“1”将会允许其对应的管脚的独立中断以及联合的 GPIOINTR 线路触发。而将位清零将会禁止该管脚上的中断触发。所有位在复位时都会被清零。

GPIO中断屏蔽（GPIOIM）寄存器
偏移量 0x410

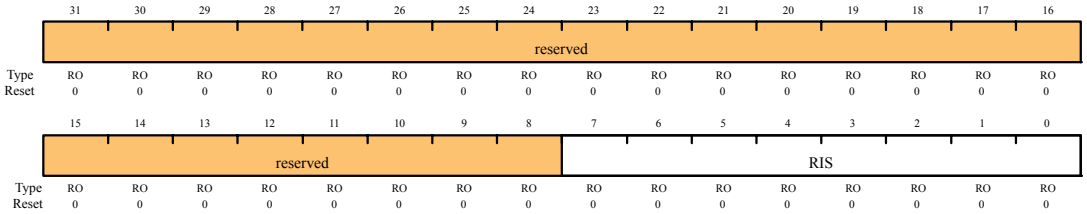


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	IME	R/W	0x00	GPIO 中断屏蔽使能 0：相应管脚的中断被屏蔽 1：相应管脚的中断未被屏蔽

寄存器 7：GPIO 原始（Raw）中断状态（GPIORIS）寄存器，偏移量：0x414

GPIORIS 寄存器是原始（Raw）中断状态寄存器。**GPIORIS** 中被读作“1”的位反映了检测到的中断触发条件的状态（原始的，屏蔽前的），表示在 **GPIO 中断屏蔽（GPIOIM）**寄存器最终允许这些位触发前所有的中断条件都已经满足。**GPIORIS** 中被读作“0”的位表示相应的输入管脚没有发起过中断。所有位在复位时都会被清零。

GPIO原始（Raw）中断状态（GPIORIS）寄存器
偏移量 0x414



位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	RIS	RO	0x00	GPIO 中断原始（raw）状态 反映在管脚上检测到的中断触发条件的状态（原始的，屏蔽前的） 0：没有满足相应管脚的中断条件 1：相应管脚的中断满足条件

寄存器 8：GPIO 屏蔽后的中断状态（GPIOMIS）寄存器，偏移量：0x418

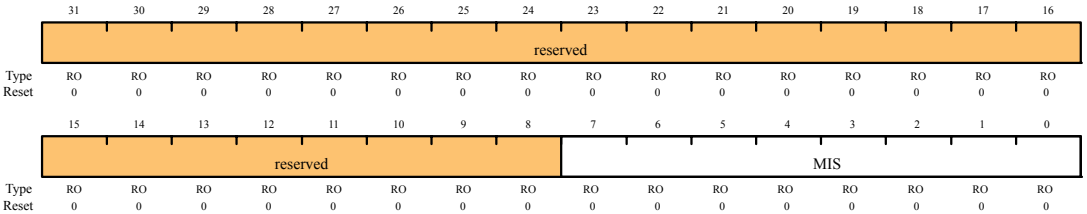
GPIOMIS 寄存器是屏蔽后的中断状态寄存器。**GPIOMIS** 中被读作“1”的位反映了触发中断的输入线路的状态。被读作“0”的位表示没有产生中断，或者中断被屏蔽。

此外，为了提供 GPIO 功能，PB4 也可以用作 ADC 的外部触发器。如果 PB4 被配置成一个不可屏蔽中断管脚（**GPIOIM** 设为 1），那么不仅仅 PB 端口会产生中断，而且还会发送一个外部触发信号给 ADC。如果 **ADC 事件多路复用器选择（ADCEMUX）**寄存器被配置成使用外部触发，那么将启动一次 ADC 转换。

如果没有其他 PB 端口可以用来产生中断，那么 ARM 集成的嵌套向量中断控制器（NVIC）中断设置使能（SETNA）寄存器可以禁能 PB 端口的中断，并且可以使用 ADC 中断来读回被转换的数据。否则，PB 端口中断处理器必须忽略和清除 B4 上的中断，并等待 ADC 中断，或者必需在 SETNA 寄存器中禁能 ADC 中断。此外，PB 端口中断处理器会轮询 ADC 寄存器单元直至转换结束。

GPIOMIS 是屏蔽后中断的状态。

GPIO屏蔽后的中断状态（GPIOMIS）寄存器
偏移量 0x418

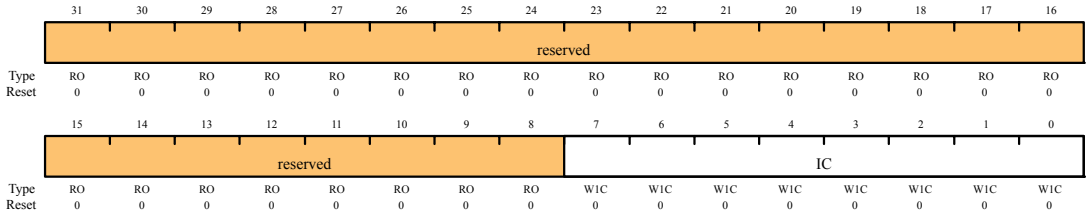


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	MIS	RO	0x00	GPIO 屏蔽后的中断状态 相应管脚上中断屏蔽后的值 0：相应的 GPIO 线路的中断未被激活 1：相应的 GPIO 线路发出中断

寄存器 9：GPIO 中断清除（GPIOCIR）寄存器，偏移量：0x41C

GPIOICR 寄存器是中断清除寄存器，对该寄存器中的位写“1”会将相应的中断边沿检测逻辑寄存器清零。写“0”没什么影响。

GPIO中断清除（GPIOICR）寄存器
偏移量 0x41C



位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	IC	W1C	0x00	GPIO 中断清除 0：相应的中断未受影响 1：相应的中断被清除

寄存器 10：GPIO 备用功能选择（GPIOAFSEL）寄存器，偏移量：0x420

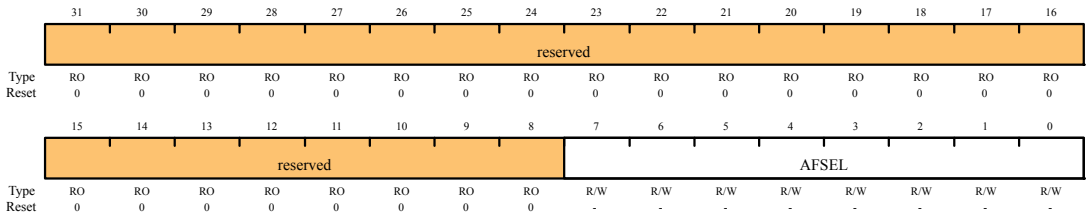
GPIOAFSEL 寄存器是模式控制选择寄存器。向该寄存器中的任意位写“1”表示选择该 GPIO 线路所对应的硬件控制（功能）。由于所有的位都在复位时都会清零，因此在默认的情况下，并无 GPIO 线被设为硬件控制（功能）。

注意：除了 5 个 JTAG 管脚（PB7 和 PC[3:0]）之外，所有 GPIO 管脚默认下都是输入管脚（GPIODIR=0 且 GPIOAFSEL=0）。JTAG 管脚在默认情况下为 JTAG 功能（GPIOAFSEL=1）。通过上电复位（POR）或外部复位（$\overline{\text{RST}}$）可以让这两组管脚都回到其默认状态。

如果 JTAG 管脚在设计中用作 GPIO，那么 PB7 和 PC2 不能同时接外部下拉电阻。如果这两个管脚在复位过程都被拉至低电平，那么控制器会出现不可预测的行为。一旦这种情况发生，应移除其中一个下拉电阻，或者把两个下拉电阻都移除，并且使用 $\overline{\text{RST}}$ 复位或关机后重新上电。

此外，可以建立一个软件程序来阻止调试器与群星系列微控制器相连。如果加载到 Flash 的程序代码立即将 JTAG 管脚变成它们的 GPIO 功能，那么在 JTAG 管脚功能切换前调试器将没有足够的时间去连接和停止控制器。这会将调试器锁在元件外。而通过一个使用外部触发器来恢复 JTAG 功能的软件程序就可以避免这种情况发生。

GPIO备用（Alternate）功能选择（GPIOAFSEL）寄存器
偏移量 0x420

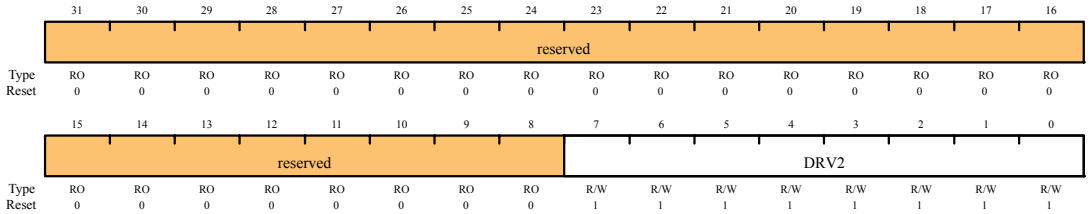


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	AFSEL	R/W	见注释	GPIO 备用功能选择 0：软件控制相应的 GPIO 线（GPIO 模式） 1：硬件控制相应的 GPIO 线（可选的硬件功能） **注**：对于除 5 个 JTAG 管脚（PB7 和 PC[3:0]）之外的所有 GPIO 管脚，**GPIOAFSEL** 寄存器的默认复位值是 0x00。那 5 个 JTAG 管脚默认为 JTAG 功能。因此对于 GPIO 端口 B（PB），**GPIOAFSEL** 的默认复位值为 0x80，而对于 GPIO 端口 C（PC），**GPIOAFSEL** 的默认复位值为 0x0F。

寄存器 11：GPIO 2-mA 驱动选择（GPIODR2R）寄存器，偏移量：0x500

GPIODR2R 寄存器是 2-mA 驱动控制寄存器。它允许对端口的每个 GPIO 信号进行单独配置，而不会影响其他引脚（pad）。当对 GPIO 信号的 DRV2 位进行写入时，**GPIODR4R** 寄存器中相应的 DRV4 位和 **GPIODR8R** 寄存器中的 DRV8 位都自动被硬件清零。

GPIO 2-mA驱动选择（GPIODR2R）寄存器
偏移量 0x500

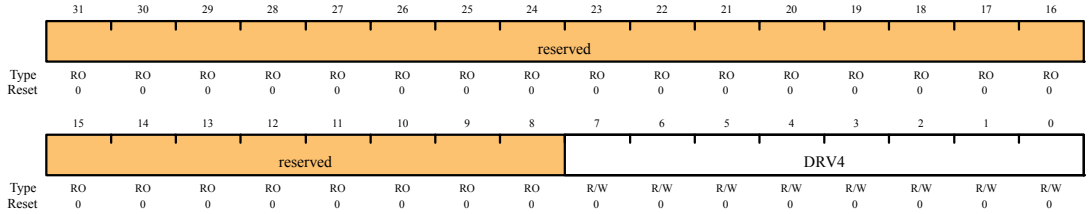


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DRV2	R/W	0xFF	输出端口 2-mA 驱动使能 向 **GPIODR4[n]**或者 **GPIODR8[n]**写入 1，都会使相应的 2-mA 使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

寄存器 12：GPIO 4-mA 驱动选择（GPIODR4R）寄存器，偏移量：0x504

GPIODR4R 寄存器是 4-mA 驱动控制寄存器。它允许对端口的每个 GPIO 信号进行单独配置，而不会影响其他引脚（pad）。当对 GPIO 信号的 DRV4 位进行写入时，**GPIODR2R** 寄存器中相应的 DRV2 位和 **GPIODR8R** 寄存器中的 DRV8 位都自动被硬件清零。

GPIO 4-mA驱动选择（GPIODR4R）寄存器
偏移量 0x504

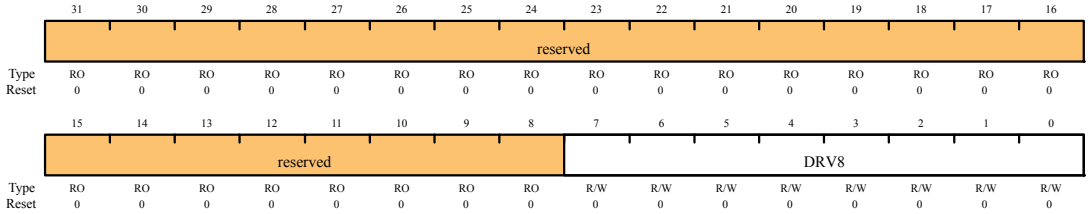


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DRV4	R/W	0x00	输出端口 4-mA 驱动使能 向 **GPIODR2[n]**或者 **GPIODR8[n]**写入 1，都会使相应的 4-mA 使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

寄存器 13：GPIO 8-mA 驱动选择（GPIODR8R）寄存器，偏移量：0x508

GPIODR8R 寄存器是 8-mA 驱动控制寄存器。它允许对端口的每个 GPIO 信号进行单独配置，而不会影响其他引脚（pad）。当对 GPIO 信号的 DRV8 位进行写入时，**GPIODR2R** 寄存器中相应的 DRV2 位和 **GPIODR4R** 寄存器中的 DRV4 位都自动被硬件清零。

GPIO 8-mA驱动选择（GPIODR8R）寄存器
偏移量 0x508

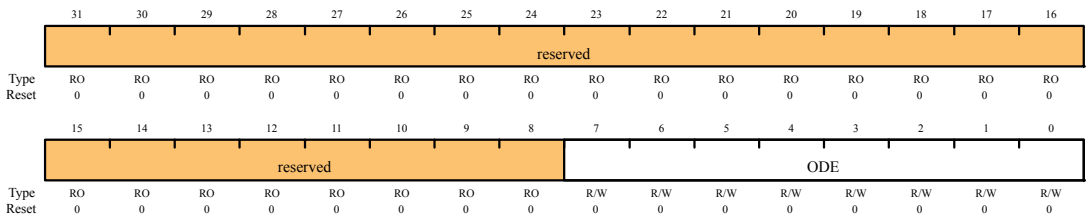


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DRV8	R/W	0x00	输出端口 8-mA 驱动使能 向 **GPIODR2[n]**或者 **GPIODR4[n]**写入 1，都会使相应的 8-mA 使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

寄存器 14：GPIO 开漏选择（GPIOODR）寄存器，偏移量：0x50C

GPIOODR 寄存器是开漏控制寄存器。置位该寄存器中的位会使能相应 GPIO 端口的开漏配置（功能）。当开漏模式使能时，还应该将 **GPIO 数字输入使能（GPIOEN）**寄存器中相应的位置位。为了达到所需的上升和下降时间，可以将驱动强度寄存器（**GPIODR2R**，**GPIODR4R**，**GPIODR8R** 和 **GPIOSLR**）中的相应位置位。如果 **GPIODIR** 寄存器中相应的位被设为 0，那么 GPIO 将充当开漏输入；如果设为 1，那么将充当开漏输出。

GPIO开漏选择（GPIOODR）寄存器
偏移量 0x50C

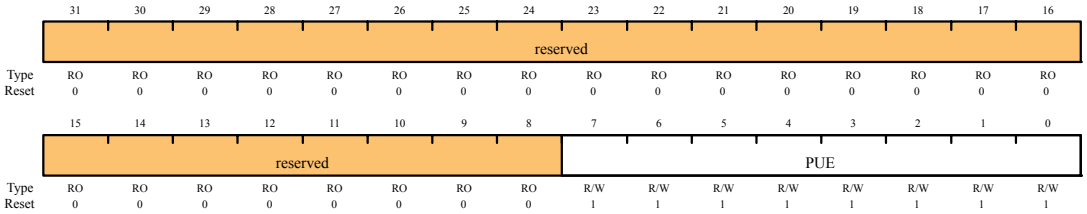


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	ODE	R/W	0x00	输出端口开漏使能 1：开漏配置被禁能 0：开漏配置被使能

寄存器 15：GPIO 上拉选择（GPIOPUR）寄存器，偏移量：0x510

GPIOPUR 寄存器是上拉控制寄存器。当其中的位被设为 1 时，它会使能相应的 GPIO 信号上的弱上拉电阻。置位 **GPIOPUR** 中的位会使 **GPIO 下拉选择（GPIOPDR）**寄存器中相应的位自动清零。

GPIO上拉选择（GPIOPUR）寄存器
偏移量 0x510

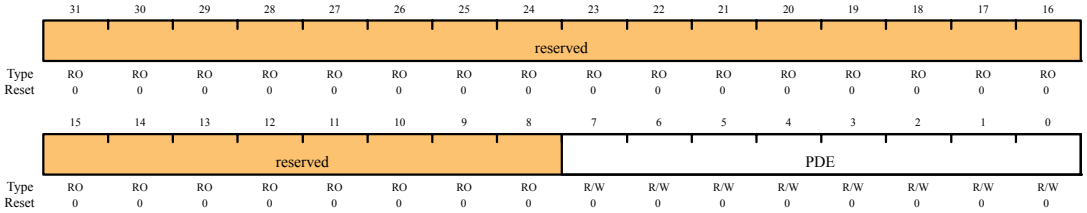


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PUE	R/W	0xFF	端口弱上拉使能 向 **GPIOPDR[n]**写 1 会清零相应的 **GPIOPUR[n]**使能位。这种改变在写操作之后的第二个时钟周期有效

寄存器 16：GPIO 下拉选择（GPIOPDR）寄存器，偏移量：0x514

GPIOPDR 寄存器是下拉控制寄存器。当其中的位被设为 1 时，它会使能相应的 GPIO 信号上的弱下拉电阻。置位 **GPIOPDR** 中的位会使 **GPIO 上拉选择（GPIOPUR）**寄存器中相应的位自动清零。

GPIO下拉选择（GPIOPDR）寄存器
偏移量 0x514

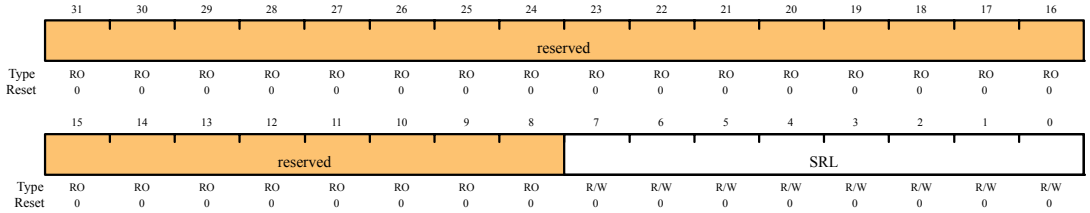


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PDE	R/W	0x00	端口弱下拉使能 向 **GPIOPUR[n]**写 1 会清零相应的 **GPIOPDR[n]**使能位。这种改变在写操作之后的第二个时钟周期有效

寄存器 17：GPIO 斜率控制选择（GPIOSLR）寄存器，偏移量：0x518

GPIOSLR 寄存器是斜率控制寄存器。斜率控制只在通过 **GPIO 8-mA 驱动选择（GPIODR8R）**寄存器采用 8-mA 驱动强度选项时才有效。

GPIO斜率控制选择（GPIOSLR）寄存器
偏移量 0x518

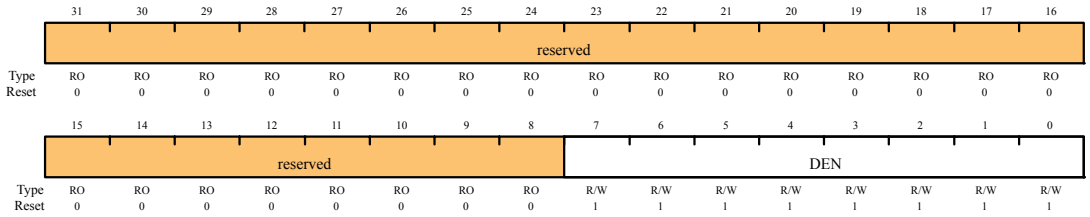


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	SRL	R/W	0	斜率限制使能（仅为 8-mA 驱动） 0：斜率控制被禁能 1：斜率控制被使能

寄存器 18：GPIO 数字输入使能（GPIODEN）寄存器，偏移量：0x51C

GPIODEN 寄存器是数字输入使能寄存器。默认下，复位时所有 GPIO 信号都被配置成数字输入。只有当 GPIO 管脚被配置成模拟比较器的其中一个模拟输入信号时管脚才不会被配置成数字输入。

GPIO数字输入使能（GPIODEN）寄存器
偏移量 0x51C

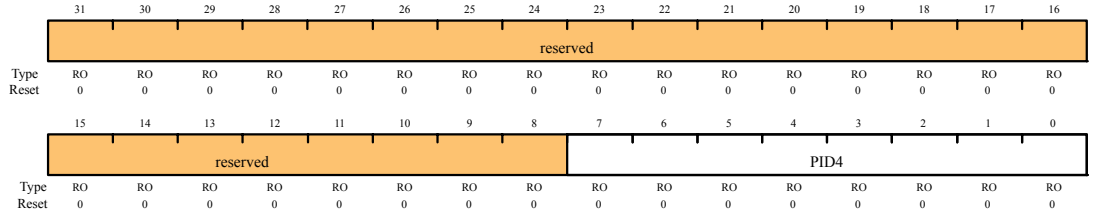


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DEN	R/W	0xFF	数字输入使能 0：数字输入被禁能 1：数字输入被使能

寄存器 19：GPIO 外设标识 4（GPIOPeriphID4）寄存器，偏移量：0xFD0

GPIOPeriphID4，**GPIOPeriphID5**，**GPIOPeriphID6** 和 **GPIOPeriphID7** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识4（GPIOPeriphID4）寄存器
偏移量 0xFD0

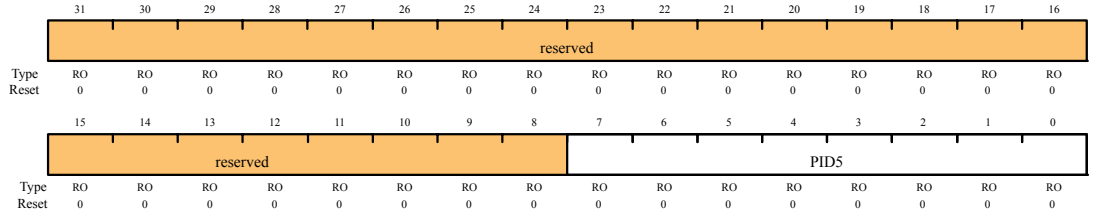


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID4	RO	0x00	GPIO 外设 ID 寄存器[7:0]

寄存器 20：GPIO 外设标识 5（GPIOPeriphID5）寄存器，偏移量：0xFD4

GPIOPeriphID4，**GPIOPeriphID5**，**GPIOPeriphID6** 和 **GPIOPeriphID7** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识5（GPIOPeriphID5）寄存器
偏移量 0xFD4

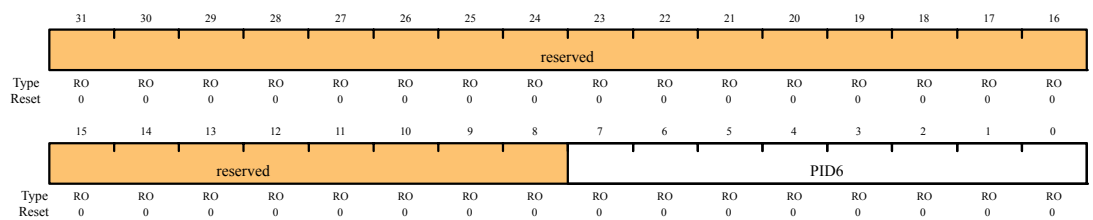


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID5	RO	0x00	GPIO 外设 ID 寄存器[15:8]

寄存器 21：GPIO 外设标识 6（GPIOPeriphID6）寄存器，偏移量：0xFD8

GPIOPeriphID4，**GPIOPeriphID5**，**GPIOPeriphID6** 和 **GPIOPeriphID7** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识6（GPIOPeriphID6）寄存器
偏移量 0xFD8

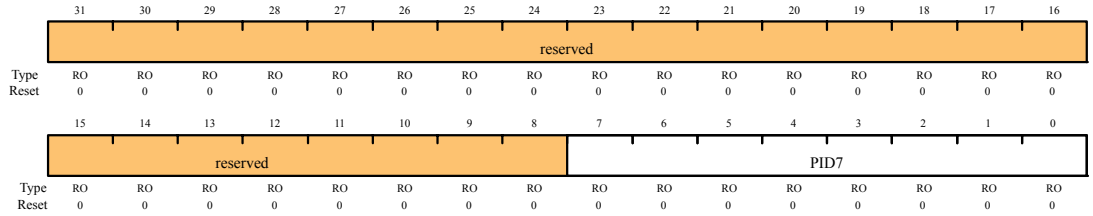


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID6	RO	0x00	GPIO 外设 ID 寄存器[23:16]

寄存器 22：GPIO 外设标识 7（GPIOPeriphID7）寄存器，偏移量：0xFDC

GPIOPeriphID4，**GPIOPeriphID5**，**GPIOPeriphID6** 和 **GPIOPeriphID7** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识7（GPIOPeriphID7）寄存器
偏移量 0xFDC

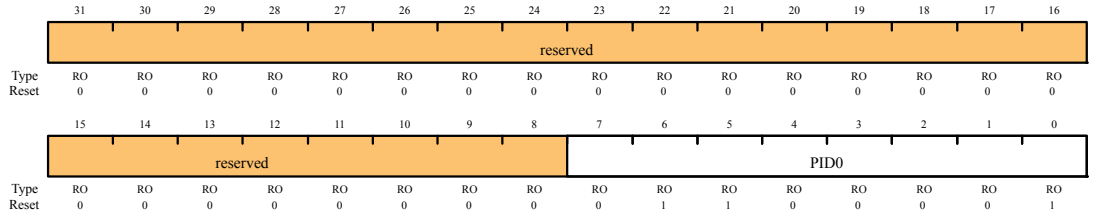


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID7	RO	0x00	GPIO 外设 ID 寄存器[31:24]

寄存器 23：GPIO 外设标识 0（GPIOPeriphID0）寄存器，偏移量：0xFE0

GPIOPeriphID0，**GPIOPeriphID1**，**GPIOPeriphID2** 和 **GPIOPeriphID3** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识0（GPIOPeriphID0）寄存器
偏移量 0xFE0

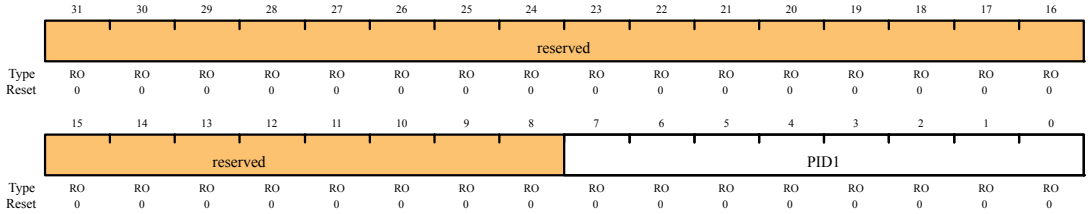


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID0	RO	0x61	GPIO 外设 ID 寄存器[7:0] 可被软件用来标识该外设是否存在

寄存器 24：GPIO 外设标识 1（GPIOPeriphID1）寄存器，偏移量：0xFE4

GPIOPeriphID0，**GPIOPeriphID1**，**GPIOPeriphID2** 和 **GPIOPeriphID3** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识1（GPIOPeriphID1）寄存器
偏移量 0xFE4

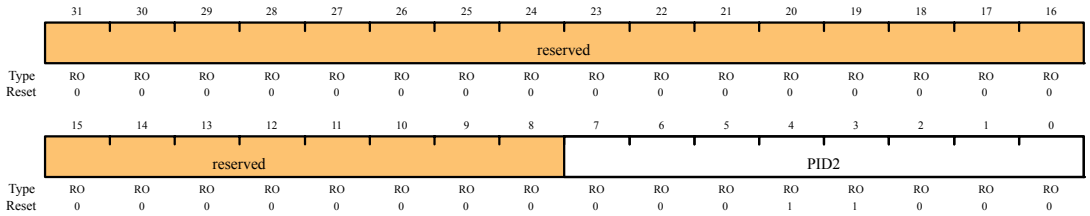


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID1	RO	0x00	GPIO 外设 ID 寄存器[15:8] 可被软件用来标识该外设是否存在

寄存器 25：GPIO 外设标识 2（GPIOPeriphID2）寄存器，偏移量：0xFE8

GPIOPeriphID0，**GPIOPeriphID1**，**GPIOPeriphID2** 和 **GPIOPeriphID3** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识2（GPIOPeriphID2）寄存器
偏移量 0xFE8

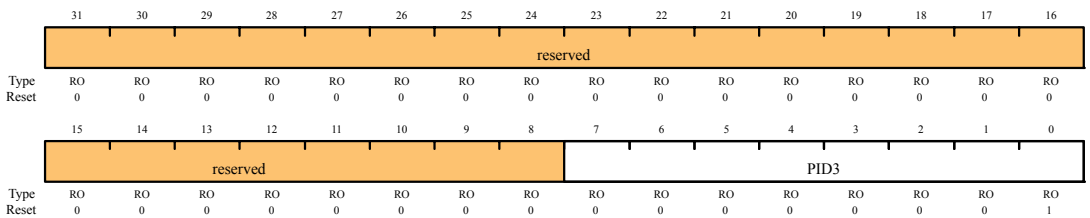


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID2	RO	0x18	GPIO 外设 ID 寄存器[23:16] 可被软件用来标识该外设是否存在

寄存器 26：GPIO 外设标识 3（GPIOPeriphID3）寄存器，偏移量：0xFEC

GPIOPeriphID0，**GPIOPeriphID1**，**GPIOPeriphID2** 和 **GPIOPeriphID3** 寄存器在概念上可以看作一个 32 位寄存器；每个寄存器包含 32 位寄存器的 8 个位，被软件用来标识外设。

GPIO外设标识3（GPIOPeriphID3）寄存器
偏移量 0xFEC

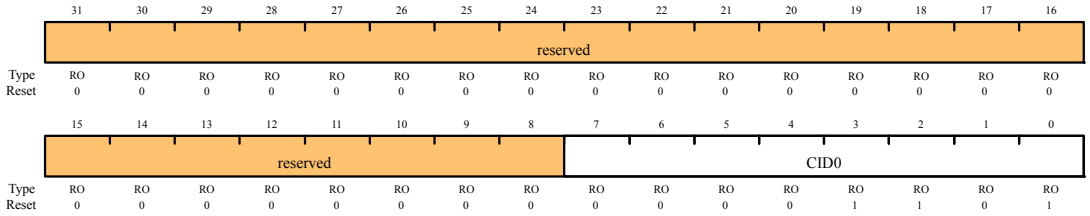


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID3	RO	0x01	GPIO 外设 ID 寄存器[31:24] 可被软件用来标识该外设是否存在

寄存器 27：GPIO PrimeCell 标识 0（GPIOPCellID0）寄存器，偏移量：0xFF0

GPIOPCellID0，**GPIOPCellID1**，**GPIOPCellID2** 和 **GPIOPCellID3** 寄存器都是 8 位寄存器，在概念上可以将它们看作一个 32 位寄存器。该寄存器将作为一个标准的交叉外设（cross-peripheral）标识系统来使用。

GPIO PrimeCell标识0（GPIOPCellID0）寄存器
偏移量 0xFF0

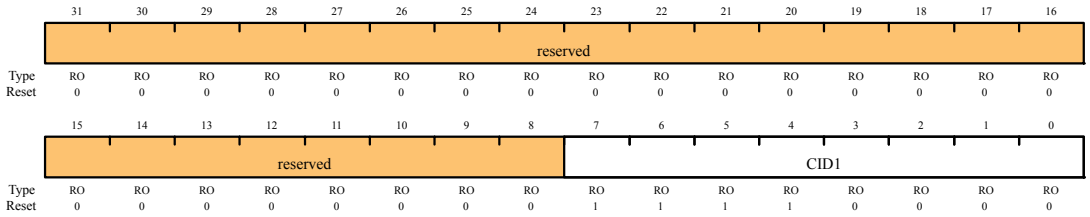


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID0	RO	0x0D	GPIO PrimeCeII ID 寄存器[7:0] 向软件提供一个标准的交叉外设标识系统

寄存器 28：GPIO PrimeCell 标识 1（GPIOPCellID1）寄存器，偏移量：0xFF4

GPIOPCellID0，**GPIOPCellID1**，**GPIOPCellID2** 和 **GPIOPCellID3** 寄存器都是 8 位寄存器，在概念上可以将它们看作一个 32 位寄存器。该寄存器将作为一个标准的交叉外设（cross-peripheral）标识系统来使用。

GPIO PrimeCell标识1（GPIOPCellID1）寄存器
偏移量 0xFF4



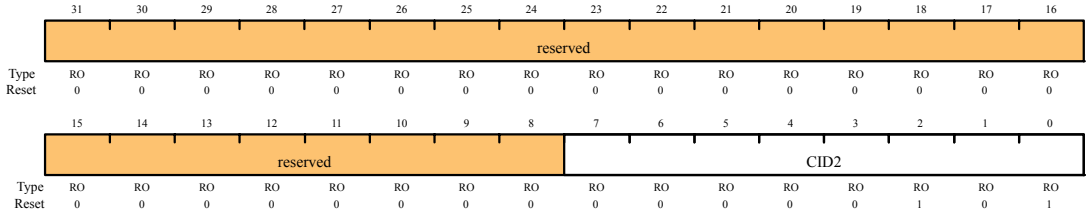
位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID1	RO	0xF0	GPIO PrimeCeII ID 寄存器[15:8] 向软件提供一个标准的交叉外设标识系统

寄存器 29：GPIO PrimeCell 标识 2（GPIOPCellID2）寄存器，偏移量：0xFF8

GPIOPCellID0，**GPIOPCellID1**，**GPIOPCellID2** 和 **GPIOPCellID3** 寄存器都是 8 位寄存器，在概念上可以将它们看作一个 32 位寄存器。该寄存器将作为一个标准的交叉外设

（cross-peripheral）标识系统来使用。

GPIO PrimeCell标识2（GPIOPCellID2）寄存器
偏移量 0xFF8

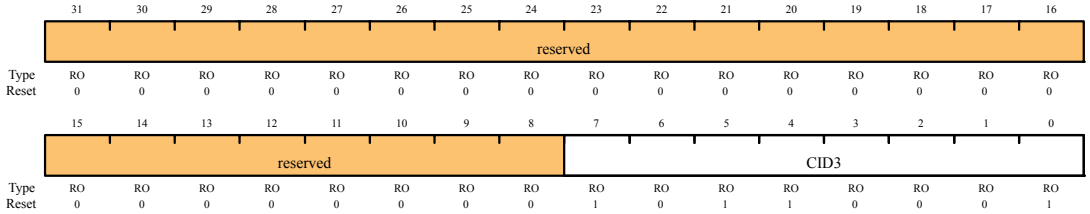


位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID2	RO	0x05	GPIO PrimeCeII ID 寄存器[23:16] 向软件提供一个标准的交叉外设标识系统

寄存器 30：GPIO PrimeCell 标识 3（GPIOPCellID3）寄存器，偏移量：0xFFC

GPIOPCellID0，**GPIOPCellID1**，**GPIOPCellID2** 和 **GPIOPCellID3** 寄存器都是 8 位寄存器，在概念上可以将它们看作一个 32 位寄存器。该寄存器将作为一个标准的交叉外设（cross-peripheral）标识系统来使用。

GPIO PrimeCeII标识3（GPIOPCeIIID3）寄存器
偏移量 0xFFC



位	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID3	RO	0xB1	GPIO PrimeCeII ID 寄存器[31:24] 向软件提供一个标准的交叉外设标识系统

第9章 通用定时器

可编程定时器可对驱动定时器输入管脚的外部事件进行计数或定时。LM3S617 控制器的通用定时器模块（GPTM）包含 3 个 GPTM 模块（Timer0、Timer1 和 Timer2）。每个 GPTM 模块含有两个 16 位的定时/计数器（称作 TimerA 和 TimerB），它们可配置为独立的定时器或事件计数器，或配置为一个 32 位定时器或一个 32 位实时时钟（RTC）。定时器也可用于触发模数转换（ADC）。通用定时器的所有触发信号在到达 ADC 模块之前要执行“或”操作，这样只有一个定时器能用于触发 ADC 事件。

GPTM 模块支持以下模式：

- 32 位定时器模式：
  - — 可编程单次触发（one-shot）定时器
  - — 可编程周期（periodic）定时器
  - — 实时时钟，使用 32.768KHz 输入时钟
  - — 事件的停止可由软件来控制（RTC 模式除外）
- 16 位定时器模式：
  - — 带 8 位预分频器的通用定时器功能
  - — 可编程单次触发（one-shot）定时器
  - — 可编程周期（periodic）定时器
  - — 事件的停止可由软件来控制
- 16 位输入捕获模式：
  - — 输入边沿计数捕获
  - — 输入边沿定时捕获
- 16 位 PWM 模式：
  - — 简单的 PWM 模式，可通过软件实现 PWM 信号的输出反相。

9.1 模块框图

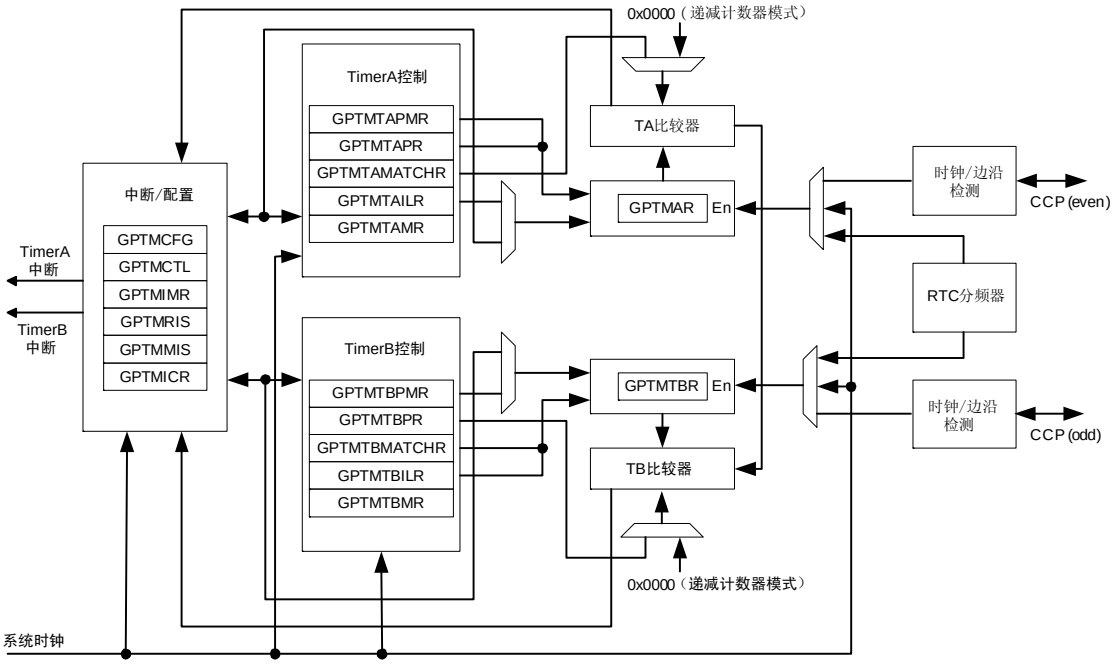


图 9.1 GPTM 模块框图

9.2 功能描述

每个 GPTM 模块的主要元件包括两个自由运行的递增/递减计数器（称作 TimerA 和 TimerB）、两个 16 位匹配寄存器、两个预分频器匹配寄存器、两个 16 位装载/初始化寄存器和它们相关的控制功能。GPTM 的准确功能可由软件来控制，并通过寄存器接口进行配置。

在通过软件对 GPTM 进行配置时需用到 **GPTM 配置(GPTMCFG)**寄存器、**GPTM TimerA 模式(GPTMTAMR)**寄存器、和 **GPTM TimerB 模式(GPTMTBMR)**寄存器。当 GPTM 模块处于其中一种 32 位模式时，该定时器只能作为 32 位定时器使用。但如果配置为 16 位模式，则 GPTM 的两个 16 位定时器可配置为 16 位模式的任意组合。

9.2.1 GPTM 复位条件

GPTM 模块复位后处于未激活状态，所有控制寄存器均被清零，同时进入默认状态。计数器 TimerA 和 TimerB 连同与它们对应的装载寄存器：**GPTM TimerA 间隔装载(GPTMTAILR)**寄存器和 **GPTM TimerB 间隔装载(GPTMTBILR)**寄存器一起初始化为 0xFFFF。预分频计数器：**GPTM TimerA 预分频(GPTMTAPR)**寄存器和 **GPTM TimerB 预分频 (GPTMTBPR)**寄存器初始化为 0x00。

9.2.2 32 位定时器操作模式

注：编号为奇数和偶数的 CCP 管脚都可用于 16 位模式，而只有编号为偶数的 CCP 管脚可用于 32 位模式。

本小节将介绍 GPTM 的 3 种 32 位定时器模式（单次触发、周期、RTC），并对其配置进行描述。

通过向 **GPTM 配置(GPTMCFG)**寄存器写入 0（单次触发/周期 32 位定时器模式）或 1（RTC 模式），可将 GPTM 模块配置为 32 位模式。在 32 位模式中，需将某些 GPTM 寄存器连在一起形成伪 32 位寄存器，进行连接的寄存器包括：

- **GPTM TimerA 间隔装载(GPTMTAILR)**寄存器[15:0]
- **GPTM TimerB 间隔装载(GPTMTBILR)**寄存器[15:0]
- **GPTM TimerA(GPTMTAR)**寄存器[15:0]
- **GPTM TimerB(GPTMTBR)**寄存器[15:0]

在 32 位模式中，GPTM 把对 **GPTMTAILR** 的 32 位写访问转换为对 **GPTMTAILR** 和 **GPTMTBILR** 的写访问。

这样，写操作最终的字顺序为：PTMTBILR[15:0]: GPTMTAILR[15:0]。同样，对 **GPTMTAR** 的读操作返回的值为：GPTMTBR[15:0]:GPTMTAR[15:0]。

9.2.2.1 32 位单次触发/周期定时器模式

在 32 位单次触发和周期定时器模式中，TimerA 和 TimerB 寄存器连在一起被配置为 32 位递减计数器。然后根据写入 **GPTM TimerA 模式(GPTMTAMR)**寄存器的 TAMR 字段的值可确定选择的是单次触发模式还是周期模式，此时不需要写 **GPTM TimerB 模式(GPTMTBMR)**寄存器。

当软件对 **GPTM 控制(GPTMCTL)**寄存器的 TAEN 位执行写操作时，定时器从其预加载的值开始递减计数。当到达 0x00000000 状态时，定时器会在下一个周期从相连的 **GPTMTAILR** 中重新装载它的初值。如果配置为单次触发模式，则定时器停止计数并将 **GPTMCTL** 寄存器的 TAEN 位清零。如果配置为周期定时器，则继续计数。

除了重装计数值，GPTM 还在到达 0x00000000 状态时产生中断并输出触发信号。GPTM 将 **GPTM 原始中断状态(GPTMRIS)**寄存器中的 TATORIS 位置位，并保持该值直到向 **GPTM 中断清零(GPTMICR)**寄存器执行写操作将其清零。如果 **GPTM 中断屏蔽(GPTIMR)** 寄存器的超时(time-out)中断使能，则 GPTM 还将 **GPTM 屏蔽后的中断状态(GPTMMIS)** 寄存器的 TATOMIS 位置位。

输出触发信号是一个单时钟周期的脉冲，它在计数器刚好到达 0x00000000 状态时生效，在紧接着的下一个周期失效。通过将 **GPTMCTL** 中的 TAOTE 位置位可将输出触发使能。输出触发信号可以触发 SoC 级别的事件，例如 ADC 转换。

如果软件在计数器运行过程中重装 **GPTMTAILR** 寄存器，则计数器在下一个时钟周期装载新值并从新值继续计数。

如果 **GPTMCTL** 寄存器的 TASTALL 位有效，则定时器停止（freeze）计数直到该信号失效。

9.2.2.2 32 位实时时钟定时器模式

在实时时钟（RTC）模式中，TimerA 和 TimerB 寄存器连在一起被配置为 32 位递增计数器。在首次选择 RTC 模式时，计数器装载的值为 0x00000001。后面装载的值全都必须通过控制器写入 **GPTM TimerA 匹配(GPTMTAMATCHR)**寄存器。

在 RTC 模式中，CCP0、CCP2、CCP4 管脚上的输入时钟为 32.768KHz。然后，将时钟信号分频为 1Hz，并通过 32 位计数器的输入端。

在软件写 **GPTMCTL** 中的 TAEN 位时，计数器从其预装载的值 0x00000001 开始递增

计数。在当前计数值与 **GPTMTAMATCHR** 中的预装载值匹配时，计数器返回到 0x00000000 并继续计数，直到出现硬件复位或被软件禁能（将 TAEN 位清零），计数停止。当计数值与预装载值匹配时，GPTM 让 **GPTMRIS** 中的 RTCRIS 位有效。如果 **GPTIMR** 中的 RTC 中断使能，则 GPTM 也将 **GPTMISR** 寄存器中的 RTCMIS 位置位并产生控制器中断。通过写 **GPTMICR** 的 RTCCINT 位可将状态标志清零。

如果 **GPTMCTL** 寄存器的 TASTALL 和/或 TBSTALL 位置位，那么定时器在 **GPTMCTL** 的 RTCEN 位置位时不会停止。

9.2.3 16 位定时器操作模式

通过向 **GPTM 配置(GPTMCFG)**寄存器写入 0x04，可将 GPTM 配置为全局 16 位模式。本小节将描述每一个 GPTM 16 位操作模式。TimerA 和 TimerB 的模式相同，因此我们只介绍一次，并用字母 **n** 来表示这两个定时器的寄存器。

9.2.3.1 16 位单次触发/周期定时器模式

在 16 位单次触发/周期定时器模式中，定时器被配置为带可选的 8 位预分频器的 16 位递减计数器，预分频器可有效地将定时器的计数范围扩大到 24 位。选择单次触发模式还是周期模式由写入 **GPTMTnMR** 寄存器的 TnMR 字段的值决定。可选预分频器中的值被加载到 **GPTM Timern 预分频(GPTMTnPR)**寄存器中。

在对 **GPTMCTL** 寄存器的 TnEN 位执行写操作时，定时器从其预装载的值开始递减计数。一旦到达 0x0000 状态，定时器便在下一个周期到来时将 **GPTMTnILR** 和 **GPTMTnPR** 的值重新载入。如果配置为单次触发模式，则定时器停止计数并将 **GPTMCTL** 寄存器的 TnEN 位清零。如果配置为周期模式，则定时器继续计数。

在到达 0x0000 状态时，定时器除了重装计数值，还产生中断并输出触发信号。GPTM 将 **GPTMRIS** 寄存器的 TnTORIS 位置位，并保持该值直到执行 **GPTMICR** 寄存器写操作将该位清零。如果 **GPTMIMR** 的超时中断使能，则 GPTM 还将 **GPTMMIS** 寄存器的 TnTOMIS 位置位并产生控制器中断。

输出触发信号是一个单时钟周期的脉冲，在计数器刚好到达 0x0000 状态时生效，并在紧接着的下一个周期失效。它通过对 **GPTMCTL** 寄存器中的 TnOTE 位置位来使能，并且可以触发 SoC 级事件，例如 ADC 转换。

如果软件在计数器正在运行时重装 **GPTMTAILR** 寄存器，则计数器将在下一个时钟周期装载新值，并从新值继续计数。

如果 **GPTMCTL** 寄存器的 TnSTALL 位使能，则定时器停止（freeze）计数，直到该信号失效后再继续计数。

下面的例子显示了在使用预分频器时 16 位自由运行的定时器的各种配置。所有的值都是以时钟频率 50MHz（或时钟周期 T_C=20ns）作为标准进行计算。

表 9.1 带预分器配置的 16 位定时器

预分频	# 时钟 (T_C)[a]	最大时间	单位
00000000	1	1.3107	ms
00000001	2	2.6214	ms
00000010	3	3.9321	ms
……..	…		
11111100	254	332.9229	ms
11111110	255	334.2336	ms
11111111	256	335.5443	ms

a T_C 为时钟周期

9.2.3.2 16 位输入边沿计数模式

在边沿计数模式中，定时器被配置为能够捕获 3 种事件类型的递减计数器，这 3 种事件类型为上升沿、下降沿、或双边沿。为了把定时器设置为边沿计数模式，**GPTMTnMR** 寄存器的 TnCMR 位必须设为 0。定时器计数时所采用的边沿类型由 **GPTMCTL** 寄存器的 TnEVENT 字段决定。在初始化过程中，需对 **GPTM Timern 匹配(GPTMTnMATCHR)**寄存器进行配置，以便 **GPTMTnILR** 寄存器和 **GPTMTnMATCHR** 寄存器之间的差值等于必须计算的边沿事件的数目。

当软件写 **GPTM 控制(GPTMCTL)**寄存器的 TnEN 位时，定时器使能并用于事件捕获。CCP 管脚上每输入一个事件，计数器的值就减 1，直到事件计数的值与 **GPTMTnMACHR** 的值匹配。这时，GPTM 让 **GPTMRIS** 寄存器的 CnMRIS 位有效（如果中断没有屏蔽，则也要让 CnMMIS 位有效）。然后计数器使用 **GPTMTnILR** 中的值执行重装操作，并且由于 GPTM 自动将 **GPTMCTL** 寄存器的 TnEN 位清零，因此计数器停止计数。一旦事件计数值满足要求，接下来的所有事件都将被忽略，直到通过软件重新将 TnEN 使能。

图 9.2 显示了输入边沿计数模式的工作情况。在这种情况下，定时器的初值设置为 **GPTMTnILR**=0x000A，匹配值**GPTMTnMATCHR**=0x0006，这样，需计数 4 个边沿事件。计数器配置为检测输入信号的双边沿。

注：在当前计数值与 **GPTMnMR** 寄存器中的值匹配之后定时器自动将 TnEN 位清零，因此最后两个边沿没有计算在内。

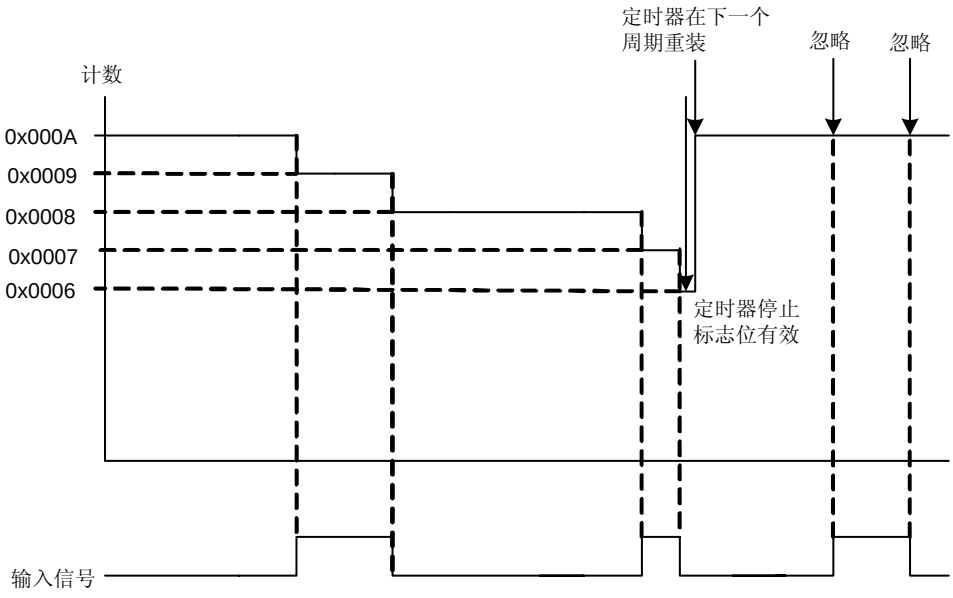


图 9.2 16 位输入边沿计数模式实例

9.2.3.3 16 位输入边沿定时模式

在边沿定时模式中，定时器被配置为自由运行的递减计数器，其初始值从 **GPTMTnILR** 寄存器中加载（复位时初始化为 0xFFFF）。该模式允许在上升沿或下降沿捕获事件。通过置位 **GPTMTnMR** 寄存器的 TnMR 位可将定时器置于边沿定时模式，而定时器捕获时采用的事件类型由 **GPTMCTL** 寄存器的 TnEVENT 字段来决定。

在软件写 **GPTMCTL** 寄存器的 TnEN 位时，定时器使能并用于事件捕获。在检测到所选的输入事件时，从 **GPTMTnR** 寄存器中捕获 **Tn** 计数器的当前值，且该值可通过控制器来读取。然后 GPTM 让 CnERIS 位有效（如果中断没有被屏蔽，则也让 CnEMIS 位有效）。

在捕获到事件之后，定时器继续计数直到 TnEN 位清零。当定时器到达 0x0000 状态时，将 **GPTMnILR** 寄存器中的值重新载入定时器。

图 9.3 显示了输入边沿定时模式的工作原理。在图中，假定定时器的初值为默认值 0xFFFF，定时器配置为捕获上升沿事件。

每当检测到上升沿事件时，当前计数值便装载到 **GPTMTnR** 寄存器中，且该值一直保持在寄存器中直到检测到下一个上升沿（在此上升沿处，新的计数值装载到 **GPTMTnR** 中）。

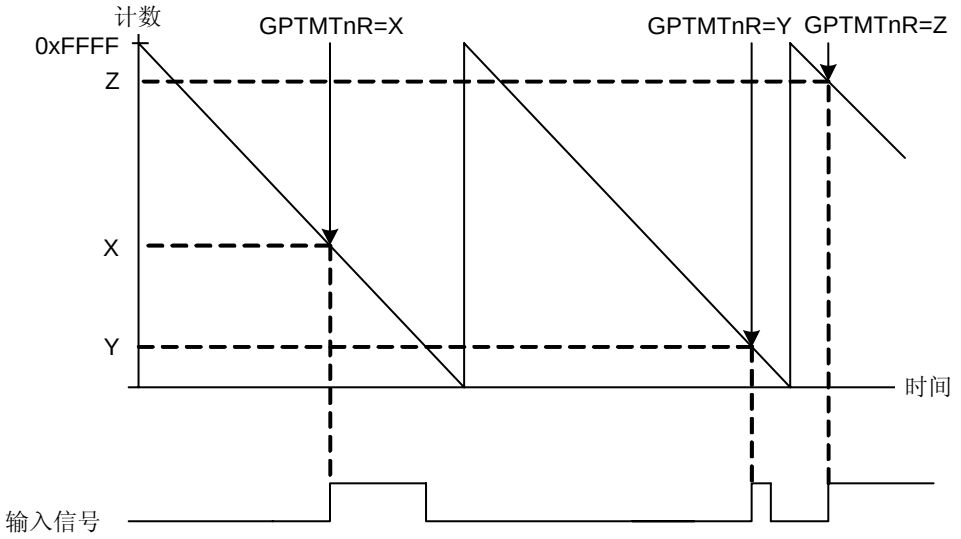


图 9.3 16 位输入边沿定时模式实例

9.2.3.4 16 位 PWM 模式

GPTM 支持简单的 PWM 生成模式。在 PWM 模式中，定时器配置为递减计数器，初值由 **GPTMTnILR** 定义。通过将 **GPTMTnMR** 寄存器的 TnAMS 位置 1、TnCMR 位置 0、TnMR 字段设置为 0x02 来使能 PWM 模式。

PWM 模式通过使用 **GPTM Timern 分频(GPTMTnPR)**寄存器和 **GPTM Timern 匹配(GPTMTnPMR)**寄存器来利用 8 位预分频器。这有效地将定时器的计数范围扩大到 24 位。

在软件写 **GPTMCTL** 寄存器的 TnEN 位时，计数器开始递减计数，直到到达 0x0000 状态。在下一个计数周期，计数器将 **GPTMTnILR** 寄存器中的值重新载入，作为它的初值（如果使用了预分频器，则还要重新装载 **GPTMTnPR** 中的值），并继续计数直到计数器因软件将 **GPTMCTL** 寄存器的 TnEN 位清零而被禁止。在 PWM 模式中，不产生中断或状态位。

当计数器的值与 **GPTMTnILR** 寄存器的值（计数器的初始状态）相等时，输出 PWM 信号生效，当计数器的值与 **GPTM Timern 匹配寄存器(GPTMTnMATCHR)**的值相等时，输出 PWM 信号失效。通过将 **GPTMCTL** 寄存器的 TnPWML 位置位，软件可实现将输出 PWM 信号反相的功能。

图 9.4显示了在输入时钟为 50MHz以及TnPWML为 0 的情况下，如何产生周期为 1ms、占空比为 66％的输出PWM（TnPWML=1 时，占空比为 33％）。在这个例子中，初值 **GPTMTnILR**=0xC350，匹配值**GPTMnMR**=0x411A。

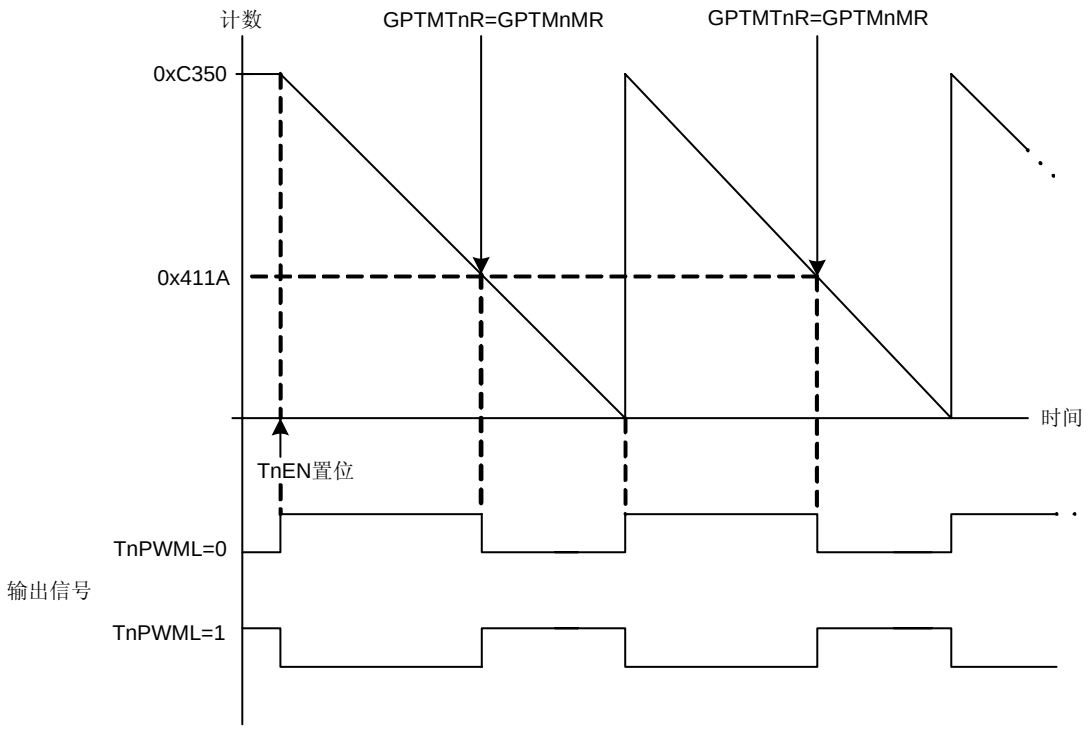


图 9.4 16 位 PWM 模式实例

9.3 初始化和配置

在使用通用定时器时，外设时钟必须使能，该操作通过将 **RCGC1** 寄存器中的 GPTM0、GPTM1 和 GPTM2 位置位来实现。

针对每种支持的定时器模式，本节提供了模块的初始化以及配置示例。

9.3.1 32 位单次触发/周期定时器模式

将 GPTM 配置为 32 位单次触发和周期模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止(将 **GPTMCTL** 寄存器的 TAEN 位清零)。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x0。
3. 设置 **GPTM TimerA 模式(GPTMTAMR)**寄存器的 TAMR 字段：
   a 写入 0x1 设为单次触发模式
   b 写入 0x2 设为周期模式
4. 将初值装入 **GPTM TimerA 间隔装载(GPTMTAILR)**寄存器。
5. 如果需要中断，将 **GPTM 中断屏蔽(GPTMIMR)**寄存器的 TATOIM 位置位。
6. 置位 **GPTMCTL** 寄存器的 TAEN 位来使能定时器并开始计数。
7. 查询 **GPTMRIS** 寄存器的 TATORIS 位或等待中断的产生（如果使能）。在这两种情况下，通过向 **GPTM 中断清零(GPTMICR)**寄存器的 TATOCINT 位写 1 将状态标志清零。

在单次触发模式中，定时器在步骤 7 之后停止计数。需重复上述步骤才能将定时器重新使能。而周期模式下的定时器在超时之后不会停止计数。

9.3.2 32 位实时时钟(RTC)模式

在使用 RTC 模式时，定时器在其 CCP0、CCP2 或 CCP4 管脚上必须有一个 32.768KHz 输入信号。在使能 RTC 功能时，需遵循以下步骤：

1. 确保定时器在发生任何变化之前先禁止（TAEN 位清零）。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x1。
3. 向 **GPTM TimerA 匹配(GPTMTAMATCHR)**寄存器写入所需的匹配值。
4. 根据需要将 **GPTM 控制(GPTMCTL)**寄存器的 RTCEN 位置位/清零。
5. 如果需要中断，将 **GPTM 中断屏蔽(GPTMIMR)**寄存器的 RTCIM 位置位。
6. 置位 **GPTMCTL** 寄存器的 TAEN 位来使能定时器并开始计数。

当定时器的计数值等于 **GPTMTAMATCHR** 寄存器中的值时，计数器重新加载 0x00000000 并开始计数。如果中断使能，不必将其清除。

9.3.3 16 位单次触发/周期定时器模式

将定时器配置为 16 位单次触发和周期模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（TnEN 位清零）。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x4。
3. 设置 **GPTM Timer 模式(GPTMTnMR)**寄存器的 TnMR 字段：
   a 写入 0x1 设为单次触发模式
   b 写入 0x2 设为周期模式
4. 如果使用预分频器，则将预分频值写入 **GPTM Timer 预分频(GPTMTnPR)**寄存器。
5. 将初值装入 **GPTM Timer 间隔装载(GPTMTnILR)**寄存器。
6. 如果需要中断，将 **GPTM 中断屏蔽(GPTMIMR)**寄存器的 TnTOIM 位置位。
7. 置位 **GPTM 控制(GPTMCTL)**寄存器的 TnEN 位来使能定时器并开始计数。
8. 查询 **GPTMRIS** 寄存器的 TnTORIS 位或等待中断的产生（如果使能）。在这两种情况下，通过向 **GPTM 中断清零(GPTMICR)**寄存器的 TnTOCINT 位写 1 将状态标志清零。

在单次触发模式中，定时器在步骤 8 之后停止计数。需重复上述步骤才能将定时器重新使能。而周期模式下的定时器在超时之后不会停止计数。

9.3.4 16 位输入边沿计数模式

将定时器配置为输入边沿计数模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（TnEN 位清零）。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x4。
3. 在 **GPTM Timer 模式(GPTMTnMR)**寄存器中，分别向 TnCMR 和 TnMR 字段写入 0x0 和 0x3。
4. 通过对 **GPTM 控制(GPTMCTL)**寄存器的 TnEVENT 字段进行写操作来配置定时器捕获操作的事件类型。

5. 将定时器初值装入 GPTM 定时器**间隔装载(GPTMTnILR)**寄存器。
6. 将所需的事件数装入 GPTM 定时器**匹配(GPTMTnMATCHR)**寄存器。
7. 如果需要中断，将 **GPTM 中断屏蔽(GPTMIMR)**寄存器的 CnMIM 位置位。
8. 置位 **GPTM 控制(GPTMCTL)**寄存器的 TnEN 位来使能定时器并开始等待边沿事件。
9. 查询 **GPTMRIS** 寄存器的 CnMRIS 位或等待中断的产生（如果使能）。在这两种情况下，通过向 **GPTM 中断清零(GPTMICR)**寄存器的 CnMCINT 位写 1 将状态标志清零。

在输入边沿计数模式中，定时器在检测到所需的边沿事件数之后停止。需确保 TnEN 位清零并重复步骤 4-9 才能重新使能定时器。

9.3.5 16 位输入边沿定时模式

将定时器配置为输入边沿定时模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（将 TnEN 位清零）。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x4。
3. 在 **GPTM Timer 模式(GPTMTnMR)**寄存器中，分别向 TnCMR 和 TnMR 字段写入 0x1 和 0x3。
4. 通过写 **GPTM 控制(GPTMCTL)**寄存器的 TnEVENT 字段来配置定时器捕获操作的事件类型。
5. 将定时器初值装入 GPTM 定时器**间隔装载(GPTMTnILR)**寄存器。
6. 如果需要中断，将 **GPTM 中断屏蔽(GPTMIMR)**寄存器的 CnEIM 位置位。
7. 置位 **GPTM 控制(GPTMCTL)**寄存器的 TnEN 位来使能定时器并开始计数。
8. 查询 **GPTMRIS** 寄存器的 CnERIS 位或等待中断的产生（如果使能）。在这两种情况下，通过向 **GPTM 中断清零(GPTMICR)**寄存器的 CnECINT 位写 1 将状态标志清零。事件发生的时间可通过读 **GPTM Timern(GPTMTnR)**寄存器来获得。

在输入边沿定时模式中，定时器在检测到边沿事件之后继续运行。而通过写 **GPTMTnILR** 寄存器可在任何时候改变定时器间隔。此改变在写操作的下一个周期生效。

9.3.6 16 位 PWM 模式

将定时器配置为 PWM 模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（将 TnEN 位清零）。
2. 向 **GPTM 配置(GPTMCFG)**寄存器写入 0x4。
3. 在 **GPTM Timer 模式(GPTMTnMR)**寄存器中，TnAMS 位设置为 0x1，TnCMR 位设置为 0x0，TnMR 字段设置为 0x2。
4. 在 **GPTM 控制(GPTMCTL)**寄存器的 TnEVENT 字段中，配置 PWM 信号的输出状态（是否需要反相）。
5. 将定时器初值装入 **GPTM Timern 间隔装载(GPTMTnILR)**寄存器。
6. 将所需的值装入 **GPTM Timern 匹配(GPTMTMATCHR)**寄存器。
7. 如果正在使用预分频器，则配置 **GPTM Timern 预分频(GPTMTnPR)**寄存器和

GPTM Timern 预分频匹配(GPTMTnPMR)寄存器。

8. 置位 **GPTM 控制(GPTMCTL)**寄存器的 TnEN 位来使能定时器并开始输出 PWM 信号。

在 PWM 模式中，定时器在产生 PWM 信号之后继续运行。通过写 **GPTMTnILR** 寄存器可在任何时候对 PWM 周期进行调整，此改变在写操作的下一个周期生效。

9.4 寄存器映射

表 9.2 列出了GPTM寄存器。偏移量相对于定时器的基址，并在寄存器地址上采用十六进制递增的方式列出。

定时器的基址：

- 定时器 0：0x40030000
- 定时器 1：0x40031000
- 定时器 2：0x40032000

表 9.2 GPTM 寄存器映射

偏移量	名称	复位	类型	描述
0x000	GPTMCFG	0x00000000	R/W	配置
0x004	GPTMTAMR	0x00000000	R/W	TimerA 模式
0x008	GPTMTBMR	0x00000000	R/W	TimerB 模式
0x00C	GPTMCTL	0x00000000	R/W	控制
0x018	GPTMIMR	0x00000000	R/W	中断屏蔽
0x01C	GPTMRIS	0x00000000	RO	中断状态
0x020	GPTMMIS	0x00000000	RO	屏蔽后的中断状态
0x024	GPTMICR	0x00000000	W1C	中断清零
0x028	GPTMTAILR	0x0000FFFF[a] 0xFFFFFFFF	R/W	TimerA 间隔装载
0x02C	GPTMTBILR	0x0000FFFF	R/W	TimerB 间隔装载
0x030	GPTMTAMATCHR	0x0000FFFF[a] 0xFFFFFFFF	R/W	TimerA 匹配
0x034	GPTMTBMATCHR	0x0000FFFF	R/W	TimerB 匹配
0x038	GPTMTAPR	0x00000000	R/W	TimerA 预分频
0x03C	GPTMTBPR	0x00000000	R/W	TimerB 预分频
0x040	GPTMTAPMR	0x00000000	R/W	TimerA 预分频匹配
0x044	GPTMTBPMR	0x00000000	R/W	TimerB 预分频匹配
0x048	GPTMTAR	0x0000FFFF[a] 0xFFFFFFFF	RO	TimerA
0x04C	GPTMTBR	0x0000FFFF	RO	TimerB

a 在 16 位模式中，**GPTMTAILR**、**GPTMTAMATCHR**、和 **GPTMTAR** 寄存器的默认复位值为 0x0000FFFF，在 32 位模式中为 0xFFFFFFFF。

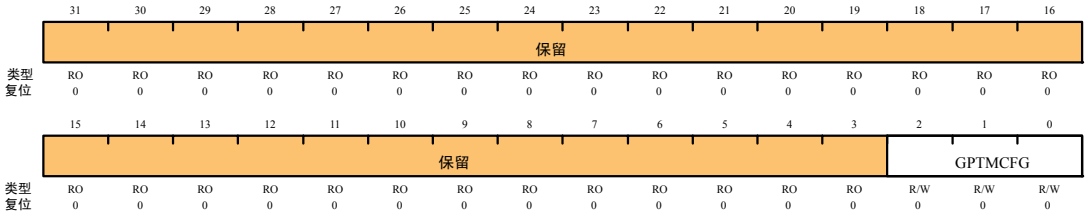
9.5 寄存器描述

下面将按地址偏移的数字顺序列举并描述 GPTM 寄存器。

寄存器 1：GPTM 配置(GPTMCFG)，偏移量 0x000

该寄存器对 GPTM 模块的全局操作进行配置。写入该寄存器的值决定了 GPTM 是 32 位还是 16 位模式。

GPTM 配置 (GPTMCFG)
偏移量 0x000

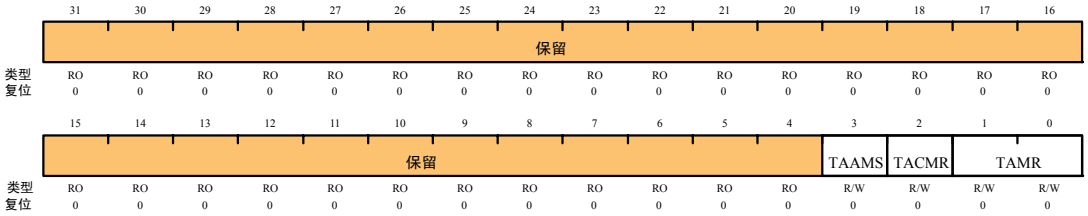


位/字段	名称	类型	复位	描述
31:3	保留	RO	0	保留位，返回不确定的值，并且应永不改变
2:0	GPTMCFG	R/W	0	GPTM 配置 0x0：32 位定时器配置 0x1：32 位实时时钟（RTC）计数器配置 0x2：保留 0x3：保留 0x4-0x7：16 位定时器配置，功能由 **GPTMTAMR** 和 **GPTMTBMR** 的位 1:0 控制。

寄存器 2：GPTM TimeA 模式(GPTMTAMR)，偏移量 0x004

该寄存器根据 **GPTMCFG** 寄存器中所选的配置来进一步配置 GPTM。当 GPTM 为 16 位 PWM 模式中时，TAAMS 位设为 0x1，TACMR 位设为 0x0，TAMR 字段设为 0x2。

GPTM TimerA 模式 (GPTMTAMR)
偏移量 0x004

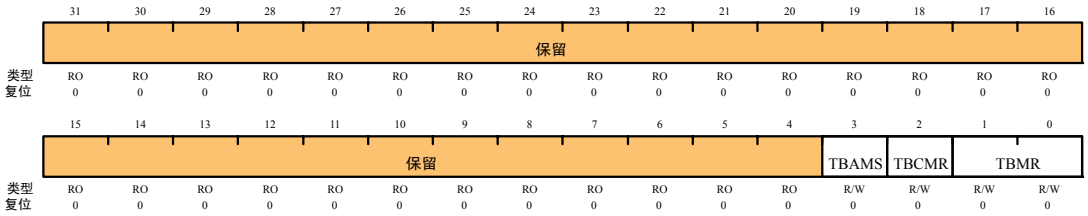


位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	TAAMS	R/W	0	GPTM TimerA 可选的模式选择 0：捕获模式使能 1：PWM 模式使能 注：为了使能 PWM 模式，用户还要将 TACMR 位清零并将 TAMR 字段设置为 0x2。
2	TACMR	R/W	0	GPTM TimerA 捕获模式 0：边沿计数模式 1：边沿定时模式
1:0	TAMR	R/W	0	GPTM TimerA 模式 0x0：保留 0x1：单次触发定时器模式 0x2：周期定时器模式 0x3：捕获模式 定时器模式基于 **GPTMCFG** 寄存器的位 2:0 所定义的定时器配置（16 或 32 位）。 在 16 位定时器配置中，TAMR 控制 TimerA 的 16 位定时器模式。 在 32 位定时器配置中，用该寄存器来控制模式，**GPTMTBMR** 的内容被忽略。

寄存器 3：GPTM TimeB 模式(GPTMTBMR)，偏移量 0x008

该寄存器根据 **GPTMCFG** 寄存器中所选的配置来进一步配置 GPTM。当 GPTM 为 16 位 PWM 模式时，TBAMS 位设为 0x1，TBCMR 位设为 0x0，TBMR 字段设为 0x2。

GPTM TimerB 模式 (GPTMTBMR)
偏移量 0x008



位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	TBAMS	R/W	0	GPTM TimerB 可选的模式选择 0：捕获模式使能 1：PWM 模式使能 注：为了使能 PWM 模式，用户还要将 TBCMR 位清零并将 TBMR 字段设置为 0x2。
2	TBCMR	R/W	0	GPTM TimerB 捕获模式 0：边沿计数模式 1：边沿定时模式

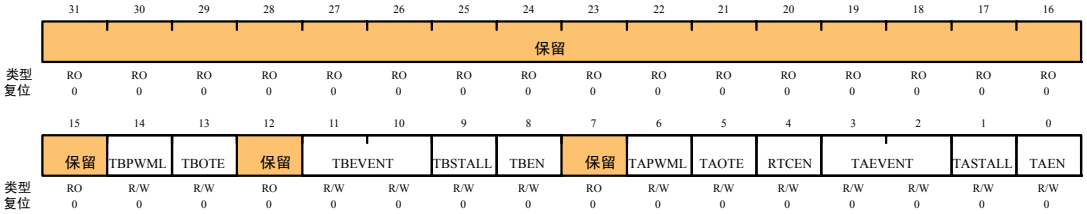
续上表

位/字段	名称	类型	复位	描述
1:0	TBMR	R/W	0	GPTM TimerB 模式 0x0：保留 0x1：单次触发定时器模式 0x2：周期定时器模式 0x3：捕获模式 定时器模式基于 **GPTMCFG** 寄存器的位 2:0 所定义的定时器配置。 在 16 位定时器配置中，TBMR 控制 TimerB 的 16 位定时器模式。 在 32 位定时器配置中，该寄存器的内容被忽略，并使用 **GPTMTAMR** 寄存器。

寄存器 4：GPTM 控制(GPTMCTL)，偏移量 0x00C

该寄存器与 **GPTMCFG** 和 **GPTMTnMR** 寄存器一起使用，对定时器配置进行微小调整，并使能其它诸如定时器停止和输出触发信号等特性。输出触发器可以用来启动 ADC 模块的传输。

GPTM 控制 (GPTMCTL)
偏移量 0x00C



位/字段	名称	类型	复位	描述
31:15	保留	RO	0	保留位，返回不确定的值，并且应永不改变
14	TBPWML	R/W	0	GPTM TimerB 的 PWM 输出电平 0：输出不改变 1：输出反相
13	TBOTE	R/W	0	GPTM TimerB 的输出触发使能 0：TimerB 输出触发禁止 1：TimerB 输出触发使能
12	保留	RO	0	保留位，返回不确定的值，并且应永不改变
11:10	TBEVENT	R/W	0	GPTM TimerB 事件模式 00：上升沿 01：下降沿 10：保留 11：双边沿

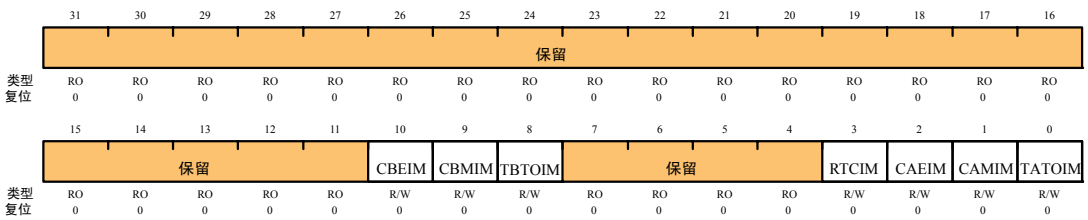
续上表

位/字段	名称	类型	复位	描述
9	TBSTALL	R/W	0	GPTM TimerB 停止使能 0：禁止 TimerB 停止 1：使能 TimerB 停止
8	TBEN	R/W	0	GPTM TimerB 使能 0：TimerB 禁止 1：TimerB 使能并开始计数，或根据 **GPTMCFG** 寄存器使能捕获逻辑。
7	保留	RO	0	保留位，返回不确定的值，并且应永不改变
6	TAPWML	R/W	0	GPTM TimerA 的 PWM 输出电平 0：输出未改变 1：输出反相
5	TAOTE	R/W	0	GPTM TimerA 的输出触发使能 0：TimerA 输出触发禁止 1：TimerA 输出触发使能
4	RTCEN	R/W	0	GPTM RTC 使能 0：RTC 计数禁止 1：RTC 计数使能
3:2	TAEVENT	R/W	0	GPTM TimerA 事件模式 00：上升沿 01：下降沿 10：保留 11：双边沿
1	TASTALL	R/W	0	GPTM TimerA 停止使能 0：禁止 TimerA 停止 1：使能 TimerA 停止
0	TAEN	R/W	0	GPTM TimerA 使能 0：TimerA 禁止 1：TimerA 使能并开始计数，或根据 **GPTMCFG** 寄存器使能捕获逻辑。

寄存器 5：GPTM 中断屏蔽(GPTMIMR)，偏移量 0x018

该寄存器允许软件使能/禁止 GPTM 控制器级中断。写入 1 使能中断，写入 0 禁止中断。

GPTM 中断屏蔽 (GPTMIMR)
偏移量 0x018

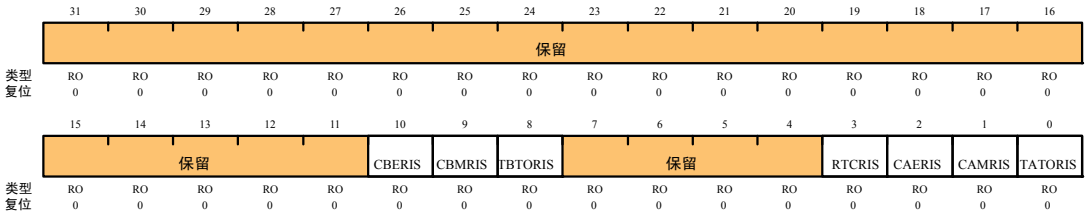


位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位，返回不确定的值，并且应永不改变
10	CBEIM	R/W	0	GPTM CaptureB 的事件中断屏蔽 0：中断禁止 1：中断使能
9	CBMIM	R/W	0	GPTM CaptureB 的匹配中断屏蔽 0：中断禁止 1：中断使能
8	TBTOIM	R/W	0	GPTM TimerB 的超时中断屏蔽 0：中断禁止 1：中断使能
7:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	RTCIM	R/W	0	GPTM RTC 的中断屏蔽 0：中断禁止 1：中断使能
2	CAEIM	R/W	0	GPTM CaptureA 的事件中断屏蔽 0：中断禁止 1：中断使能
1	CAMIM	R/W	0	GPTM CaptureA 的匹配中断屏蔽 0：中断禁止 1：中断使能
0	TATOIM	R/W	0	GPTM TimerA 的超时中断屏蔽 0：中断禁止 1：中断使能

寄存器 6：GPTM 原始中断状态(GPTMRIS)，偏移量 0x01C

该寄存器显示了 GPTM 内部中断信号的状态。不管是否在 **GPTMIMR** 寄存器中将中断屏蔽，**GPTMRIS** 中的位都会置位。向 **GPTMICR** 的某一位写 1 可将 **GPTMRIS** 寄存器的对应位清零。

GPTM 原始中断状态 (GPTMRIS)
偏移量 0x01C



位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位，返回不确定的值，并且应永不改变
10	CBERIS	RO	0	GPTM CaptureB 的事件原始中断 该位表示屏蔽之前 CaptureB 的事件中断状态
9	CBMRIS	RO	0	GPTM CaptureB 的匹配原始中断 该位表示屏蔽之前 CaptureB 的匹配中断状态

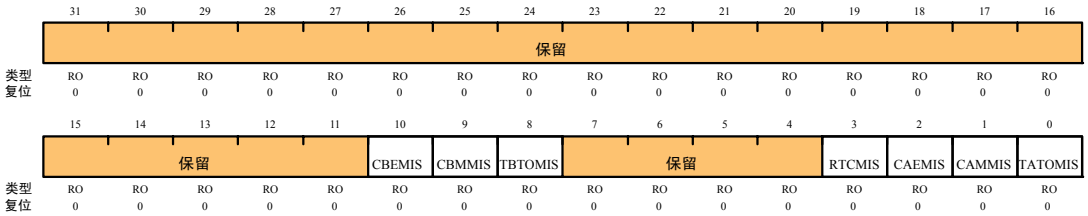
续上表

位/字段	名称	类型	复位	描述
8	TBTORIS	RO	0	GPTM TimerB 的超时原始中断 该位表示屏蔽之前 TimerB 的超时中断状态
7:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	RTCRIS	RO	0	GPTM 的 RTC 原始中断 该位表示屏蔽之前的 RTC 事件中断状态
2	CAERIS	RO	0	GPTM CaptureA 的事件原始中断 该位表示屏蔽之前 CaptureA 的事件中断状态
1	CAMRIS	RO	0	GPTM CaptureA 的匹配原始中断 该位表示屏蔽之前 CaptureA 的匹配中断状态
0	TATORIS	RO	0	GPTM TimerA 的超时原始中断 该位表示屏蔽之前 TimerA 的超时中断状态

寄存器 7：GPTM 屏蔽后的中断状态(GPTMMIS)，偏移量 0x020

该寄存器显示了 GPTM 控制器级中断的状态。如果没有在 **GPTMIMR** 寄存器中将中断屏蔽，并在此时出现一个使中断有效的事件，那么该寄存器中相应的位将会置位。通过向 **GPTMICR** 的对应位写 1 可将所有位清零。

GPTM 屏蔽后的中断状态 (GPTMMIS)
偏移量 0x020



位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位，返回不确定的值，并且应永不改变
10	CBEMIS	RO	0	GPTM CaptureB 的事件屏蔽后中断 该位表示屏蔽之后 CaptureB 的事件中断状态
9	CBMMIS	RO	0	GPTM CaptureB 的匹配屏蔽后中断 该位表示屏蔽之后 CaptureB 的匹配中断状态
8	TBTOMIS	RO	0	GPTM TimerB 的超时屏蔽后中断 该位表示屏蔽之后 TimerB 的超时中断状态
7:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	RTCMIS	RO	0	GPTM 的 RTC 屏蔽后中断 该位表示屏蔽之后的 RTC 事件中断状态
2	CAEMIS	RO	0	GPTM CaptureA 的事件屏蔽后中断 该位表示屏蔽之后 CaptureA 的事件中断状态
1	CAMMIS	RO	0	GPTM CaptureA 的匹配屏蔽后中断 该位表示屏蔽之后 CaptureA 的匹配中断状态

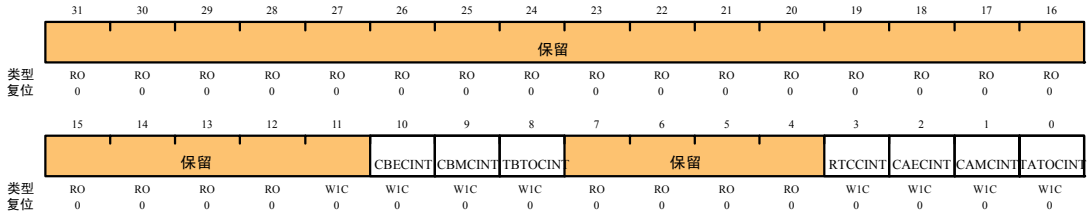
续上表

位/字段	名称	类型	复位	描述
0	TATOMIS	RO	0	GPTM TimerA 的超时屏蔽后中断 该位表示屏蔽之后 TimerA 的超时中断状态

寄存器 8：GPTM 中断清零(GPTMICR)，偏移量 0x024

该寄存器用来将 **GPTMRIS** 和 **GPTMMIS** 寄存器中的状态位清零。只要向 **GPTMICR** 的某一位写 1，便可将 **GPTMRIS** 和 **GPTMMIS** 寄存器中的对应位清零。

GPTM 中断清零 (GPTMICR)
偏移量 0x024

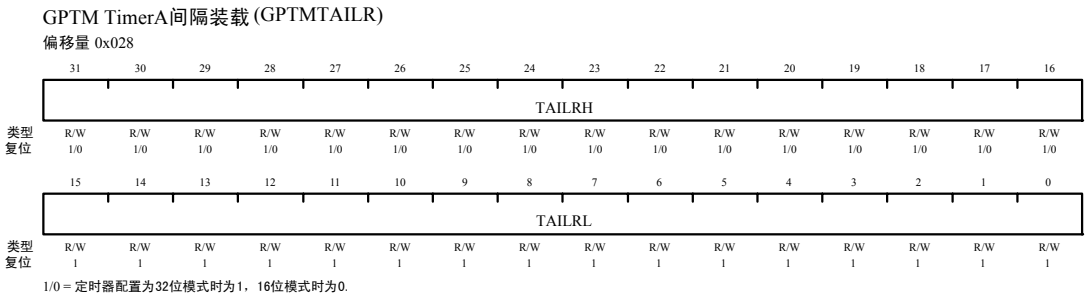


位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位，返回不确定的值，并且应永不改变
10	CBECINT	W1C	0	GPTM CaptureB 的事件中断清零 0：中断不受影响 1：中断清零
9	CBMCINT	W1C	0	GPTM CaptureB 的匹配中断清零 0：中断不受影响 1：中断清零
8	TBTOCINT	W1C	0	GPTM TimerB 的超时中断清零 0：中断不受影响 1：中断清零
7:4	保留	RO	0	保留位，返回不确定的值，并且应永不改变
3	RTCCINT	W1C	0	GPTM RTC 中断清零 0：中断不受影响 1：中断清零
2	CAECINT	W1C	0	GPTM CaptureA 的事件中断清零 0：中断不受影响 1：中断清零
1	CAMCINT	W1C	0	GPTM CaptureA 的匹配原始中断 这是屏蔽后 CaptureA 的匹配中断状态
0	TATOCINT	W1C	0	GPTM TimerA 的超时原始中断 0：中断不受影响 1：中断清零

寄存器 9：GPTM TimerA 间隔装载(GPTMTAILR)，偏移量 0x028

该寄存器用来将起始计数值装入定时器。当 GPTM 配置为其中一种 32 位模式时，

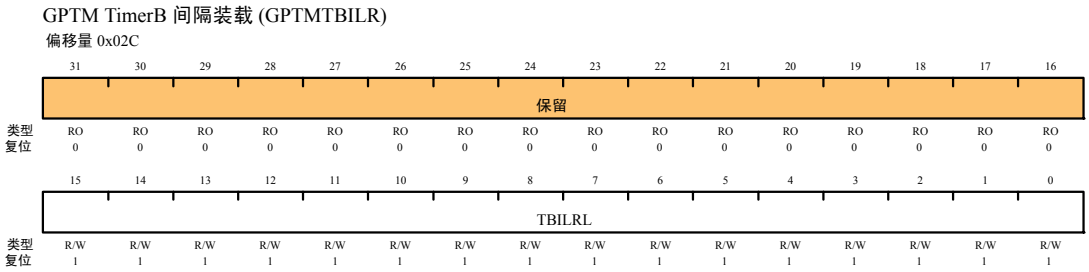
GPTMTAILR 作为 32 位寄存器使用（高 16 位对应于 **GPTM TimerB 间隔装载(GPTMTBILR)**寄存器的值）。在 16 位模式中，该寄存器的高 16 位读作 0，不影响 **GPTMTBILR** 寄存器的状态。



位/字段	名称	类型	复位	描述
31:16	TAILRH	R/W	0xFFFF (32 位模式) 0x0000 (16 位模式)	GPTM TimerA 间隔装载寄存器的高半字 当通过 **GPTMCFG** 寄存器配置为 32 位模式时，**GPTM TimerB 间隔装载(GPTMTBILR)**寄存器通过写操作来装载该值，读操作时返回 **GPTMTBILR** 的当前值。 在 16 位模式中，该字段在读操作时返回 0，不影响 **GPTMTBILR** 寄存器的状态。
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA 间隔装载寄存器的低半字 在 16 位和 32 位模式中，对该字段执行写操作将装载 TimerA 的计数器，执行读操作将返回 **GPTMTAILR** 的当前值。

寄存器 10：GPTM TimerB 间隔装载(GPTMTBILR)，偏移量 0x02C

该寄存器用来将起始计数值装入 TimerB。当 GPTM 配置为 32 位模式时，对 **GPTMTBILR** 执行读操作将返回 TimerB 的当前值，写操作被忽略。



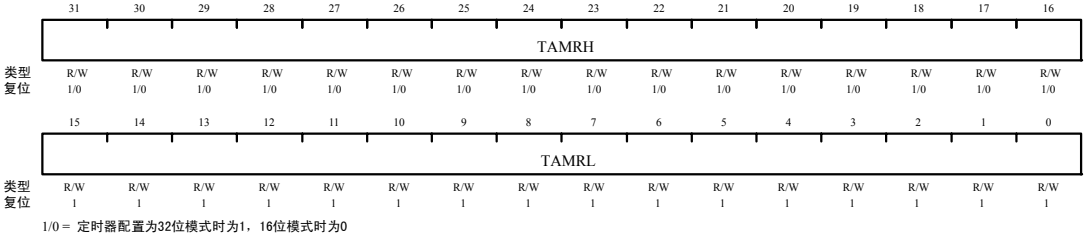
位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且应永不改变
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB 间隔装载寄存器 当 GPTM 没有被配置为 32 位定时器时，对该字段执行写操作将更新 **GPTMTBILR** 的值。在 32 位模式中，写操作被忽略，读操作返回 **GPTMTBILR** 的当前值。

寄存器 11：GPTM TimerA 匹配(GPTMTAMATCH)，偏移量 0x030

该寄存器用于 32 位实时时钟模式、16 位 PWM 和输入边沿计数模式。

GPTM TimerA 匹配 (GPTMTAMATCHR)

偏移量0x030



1/0 = 定时器配置为32位模式时为1，16位模式时为0

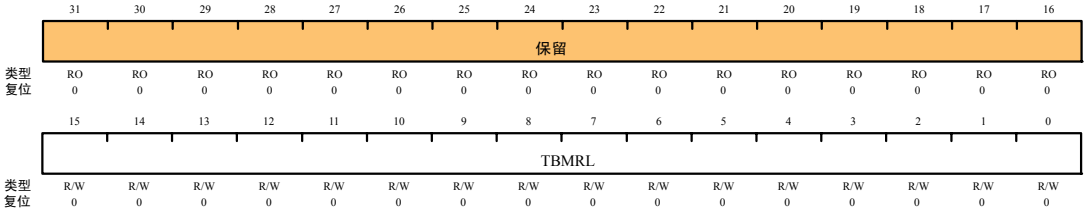
位/字段	名称	类型	复位	描述
31:16	TAMRH	R/W	0xFFFF (32 位模式) 0x0000 (16 位模式)	GPTM TimerA 匹配寄存器的高半字 当通过 **GPTMCFG** 寄存器配置为 32 位实时时钟（RTC）模式时，该值与 **GPTMTAR** 的高半字进行比较，来确定匹配事件。 在 16 位模式中，对该字段的读操作返回 0，不影响 **GPTMTBMATCHR** 寄存器的状态。
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA 匹配寄存器的低半字 当通过 **GPTMCFG** 寄存器配置为 32 位实时时钟（RTC）模式时，该值与 **GPTMTAR** 的低半字进行比较，来确定匹配事件。 当配置为 PWM 模式时，该值与 **GPTMTAILR** 一起，确定输出 PWM 信号的占空比。 当配置为边沿计数模式时，该值与 **GPTMTAILR** 一起，确定需计数多少边沿事件。总的边沿事件数等于 **GPTMTAILR** 的值与该值的差。

寄存器 12：GPTM TimerB 匹配(GPTMTBMATCHR)，偏移量 0x034

该寄存器用于 32 位实时时钟模式、16 位 PWM 和输入边沿计数模式。

GPTM TimerB 匹配 (GPTMTBMATCHR)

偏移量 0x034

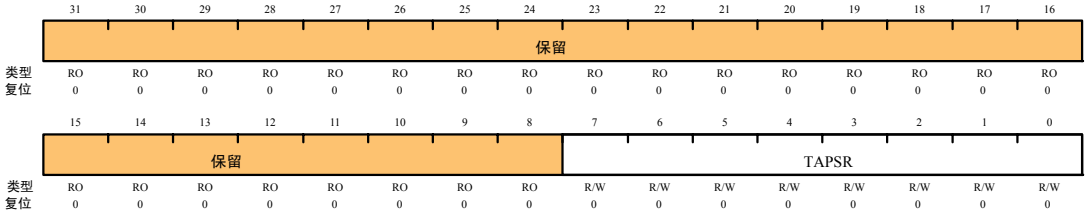


位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且应永不改变
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB 匹配寄存器的低半字 当配置为 PWM 模式时，该值与 **GPTMTBILR** 一起，确定输出 PWM 信号的占空比。 当配置为边沿计数模式时，该值与 **GPTMTBILR** 一起，确定需计数多少边沿事件数。总的边沿事件数等于 **GPTMTBILR** 与该值的差。

寄存器 13：GPTM TimerA 预分频(GPTMTAPR)，偏移量 0x038

该寄存器允许软件扩充 16 位定时器的范围。

GPTM TimerA 预分频 (GPTMTAPR)
偏移量 0x038

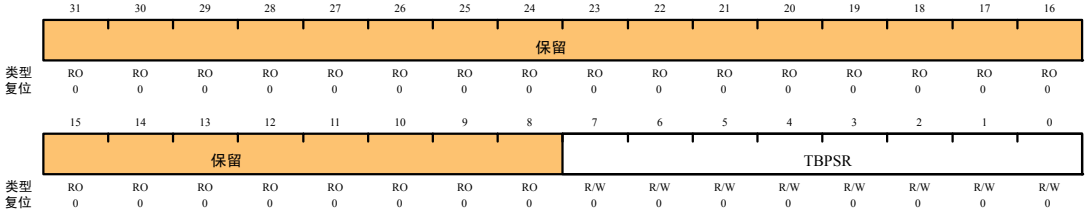


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且应永不改变
7:0	TAPSR	R/W	0	GPTM TimerA 预分频 寄存器通过写操作载入该值，执行读操作将返回寄存器的当前值。 详细信息以及实例请参考 表 9.1。

寄存器 14：GPTM TimerB 预分频(GPTMTBPR)，偏移量 0x03C

该寄存器允许软件扩充 16 位定时器的范围。

GPTM TimerB 预分频 (GPTMTBPR)
偏移量 0x03C

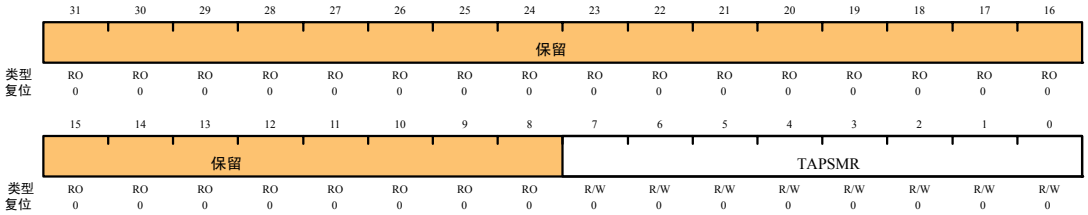


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且应永不改变
7:0	TBPSR	R/W	0	GPTM TimerB 预分频 寄存器通过写操作载入该值，执行读操作将返回寄存器的当前值。 详细信息以及实例请参考 表 9.1。

寄存器 15：GPTM TimerA 预分频匹配(GPTMTAPMR)，偏移量 0x040

该寄存器有效地将 **GPTMTAMATCHR** 的范围扩充到 24 位。

GPTM TimerA 预分频匹配 (GPTMTAPMR)
偏移量 0x040

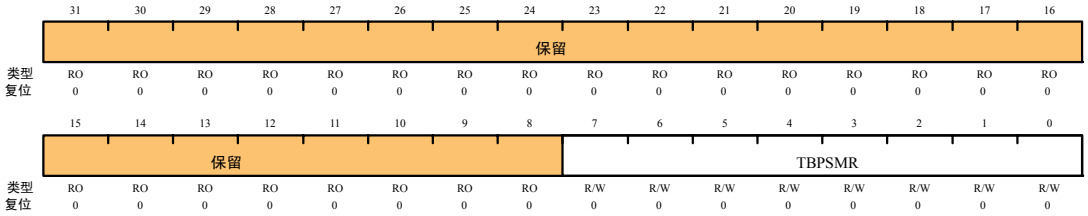


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且应永不改变
7:0	TAPSMR	R/W	0	GPTM TimerA 预分频匹配 该值与 **GPTMTAMATCHR** 一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

寄存器 16：GPTM TimerB 预分频匹配(GPTMTBPMR)，偏移量 0x044

该寄存器有效地将 **GPTMTBMATCHR** 的范围扩充到 24 位。

GPTM TimerB 预分频匹配 (GPTMTBPMR)
偏移量 0x044

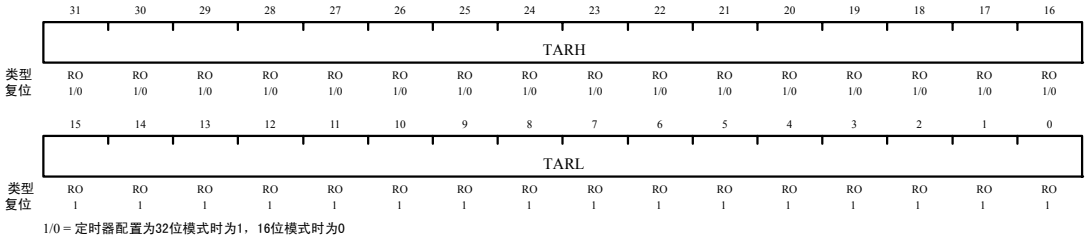


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且应永不改变
7:0	TBPSMR	R/W	0	GPTM TimerB 预分频匹配 该值与 **GPTMTAMBTCHR** 一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

寄存器 17：GPTM TimerA (GPTMTAR)，偏移量 0x048

该寄存器显示了除输入边沿计数模式之外的所有情况下 TimerA 计数器的当前值。在输入边沿计数模式中，该寄存器包含上一次边沿事件发生的时间。

GPTM TimerA (GPTMTAR)
偏移量 0x048



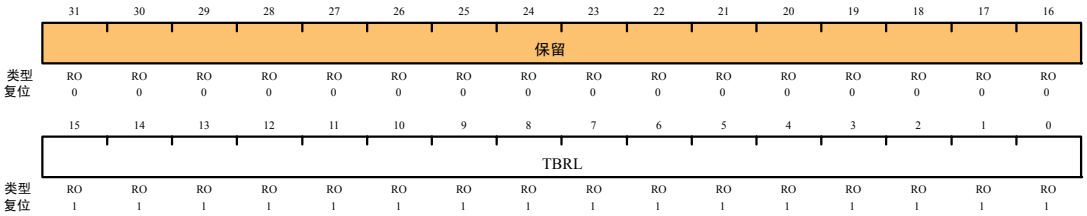
1/0 = 定时器配置为32位模式时为1，16位模式时为0

位/字段	名称	类型	复位	描述
31:16	TARH	RO	0xFFFF (32 位模式) 0x0000 (16 位模式)	GPTM TimerA 寄存器的高半字 如果将 **GPTMCFG** 配置为 32 位模式，则对该字段执行读操作将获得 TimerB 的值。如果配置为 16 位模式，则读操作返回 0。
15:0	TARL	RO	0xFFFF	GPTM TimerA 寄存器的低半字 读取该字段将返回 **GPTM TimerA 计数寄存器**的当前值，但输入边沿计数模式除外。在该模式中，读操作返回上一次边沿事件的时间戳（timestamp）。

寄存器 18：GPTM TimerB (GPTMTBR)，偏移量 0x04C

该寄存器显示了除输入边沿计数模式之外的所有情况下 TimerB 计数器的当前值。在输入边沿计数模式中，该寄存器包含上一次边沿事件发生的时间。

GPTM TimerB (GPTMTBR)
偏移量 0x04C



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且应永不改变
15:0	TBRL	RO	0xFFFF	GPTM TimerB 读取该字段将返回 **GPTM TimerB 计数寄存器**的当前值，但输入边沿计数模式除外。在该模式中，读操作返回上一次边沿事件的时间戳（timestamp）。

第10章 看门狗定时器

看门狗定时器在到达超时值时可能会产生不可屏蔽的中断（NMI）或复位。当系统由于软件错误而无法响应或外部器件不是以期望的方式响应时，使用看门狗定时器可重新获得控制权。

Stellaris 看门狗定时器模块包括 32 位递减（down）计数器、可编程的装载寄存器、中断产生逻辑、锁定寄存器以及用户使能的中止。

可将看门狗定时器配置为在首次超时(time-out)时产生中断，并在再次超时的时候产生复位信号。一旦配置了看门狗定时器，就可以写锁定寄存器来防止定时器配置被意外更改。

10.1 方框图

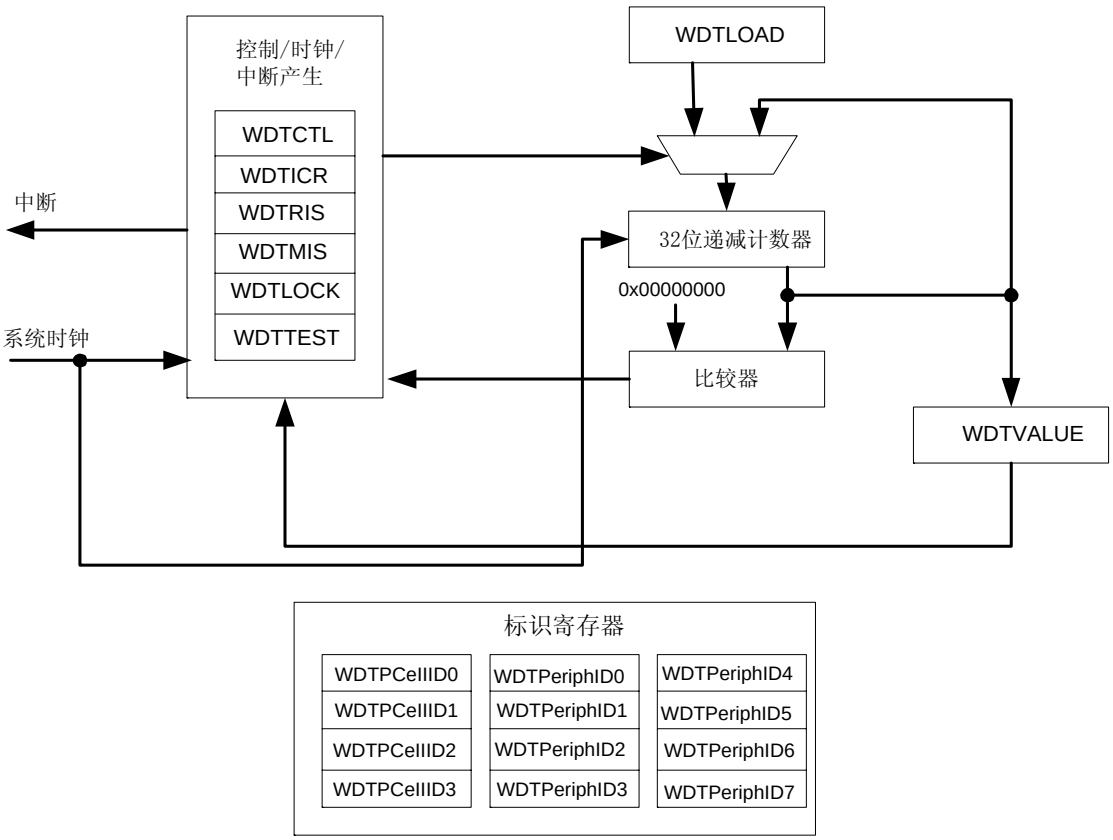


图 10.1 WDT 模块方框图

10.2 功能描述

看门狗定时器模块包括一个 32 位递减（down）计数器、可编程的装载寄存器、中断产生逻辑和锁定寄存器。一旦配置了看门狗定时器，就可以写**看门狗定时器锁定(WDTLOCK)**寄存器,以防止软件意外地改写定时器配置。

当 32 位计数器在使能后到达 0 状态时，看门狗定时器模块产生第一个超时信号；使能计数器的同时还使能看门狗定时器中断。在发生了第一个超时事件后，用**看门狗定时器装载(WDTLOAD)**寄存器的值重装 32 位计数器，并且定时器从该值恢复递减计数。

在清除第一个超时中断之前，如果定时器的值再次递减为 0，且复位信号已使能（通过 WatchdogResetEnable 功能），则看门狗定时器向系统提交其复位信号。如果中断在 32 位计数器到达其第二次超时之前被清零，则把 **WDTLOAD** 寄存器中的值载入 32 位计数器，并且从该值开始重新计数。

如果在看门狗定时器计数器正在计数时把新的值写入 **WDTLOAD**，则计数器将装入新的值并继续计数。

写入 **WDTLOAD** **并不会**不清除已经激活的中断。必须通过写**看门狗中断清除(WDTICR)**寄存器来清除中断。

可根据需要使能或禁能看门狗模块中断和产生复位。当中断被重新使能的时候，会被预先载入到 32 位计数器中的是载入寄存器的值，而不是其最后的状态值。

10.3 初始化和配置

要使用 WDT，WDT 的外设时钟就必须通过置位 RCGC0 寄存器的 WDT 位使能。按照以下顺序（sequence）配置看门狗定时器：

1. 把所需的定时器值载入 **WDTLOAD** 寄存器
2. 如果看门狗被配置为触发系统复位，则置位 **WDTCTL** 寄存器中的 RESEN 位。
3. 将 **WDTCTL** 寄存器中的 INTEN 位置位来使能看门狗并锁定控制寄存器。

如果软件需要锁定所有的看门狗寄存器，则写任意值到 **WDTLOCK** 寄存器便可以完全锁定看门狗定时器模块。若需要解锁看门狗定时器，则需写入 0x1ACCE551。

10.4 寄存器映射

表 10.1 列出了看门狗寄存器。所列偏移量是寄存器相对于看门狗定时器基址 0x40000000 的十六进制增量。

表 10.1 WDT 寄存器映射

偏移量	名称	复位	类型	描述
0x000	WDTLOAD	0xFFFFFFFF	R/W	装载
0x004	WDTVALUE	0xFFFFFFFF	RO	当前值
0x008	WDTCTL	0x00000000	R/W	控制
0x00C	WDTICR	-	WO	中断清除
0x010	WDTRIS	0x00000000	RO	原始中断状态
0x014	WDTMIS	0x00000000	RO	屏蔽中断状态
0x418	WDTTEST	0x00000000	R/W	看门狗中止使能
0xC00	WDTLOCK	0x00000000	R/W	锁定
0xFD0	WDTPeriphID4	0x00000000	RO	外设标识 4
0xFD4	WDTPeriphID5	0x00000000	RO	外设标识 5
0xFD8	WDTPeriphID6	0x00000000	RO	外设标识 6
0xFDC	WDTPeriphID7	0x00000000	RO	外设标识 7
0xFE0	WDTPeriphID0	0x00000005	RO	外设标识 0

续上表

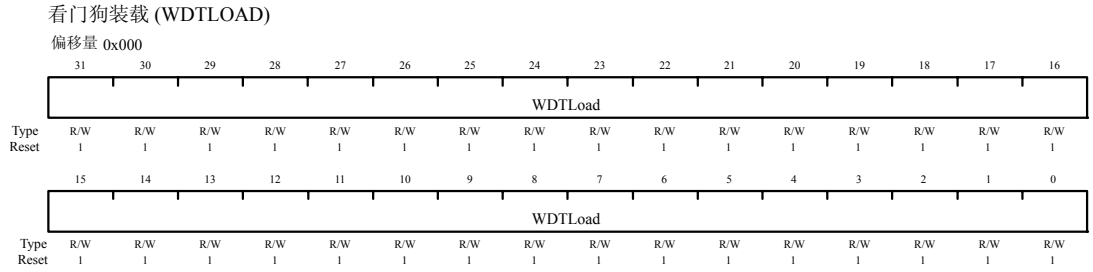
偏移量	名称	复位	类型	描述
0xFE4	WDTPeriphID1	0x00000018	RO	外设标识 1
0xFE8	WDTPeriphID2	0x00000018	RO	外设标识 2
0xFEC	WDTPeriphID3	0x00000001	RO	外设标识 3
0xFF0	WDTPCellID0	0x0000000D	RO	PrimeCell 标识 0
0xFF4	WDTPCellID1	0x000000F0	RO	PrimeCell 标识 1
0xFF8	WDTPCellID2	0x00000005	RO	PrimeCell 标识 2
0xFFC	WDTPCellID3	0x000000B1	RO	PrimeCell 标识 3

10.5 寄存器描述

下文将按地址偏移量的数字顺序列出 WDT 寄存器，并对它们进行描述。

寄存器 1：看门狗装载(WDTLOAD)，偏移量 0x000

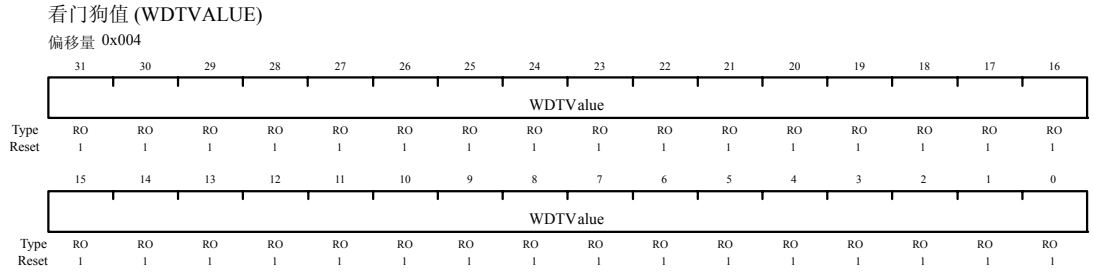
该寄存器存放的是供 32-位计数器使用的 32-位间隔值。该寄存器被写入时，这个值将被立即装载，而且计数器会从新的值重新开始递减计数。如果用 0x00000000 装载 **WDTLOAD** 寄存器，则立即产生中断。



位/字段	名称	类型	复位	描述
31:0	WDTLoad	R/W	0xFFFFFFFF	看门狗装载值。

寄存器 2：看门狗值(WDTVALUE)，偏移量 0x004

该寄存器包含了定时器的当前计数值。

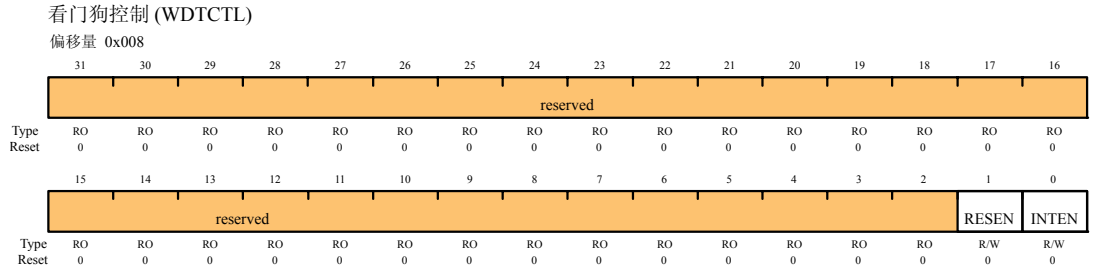


位/字段	名称	类型	复位	描述
31:0	WDTValue	RO	0xFFFFFFFF	看门狗值。 32 位递减计数器的当前值。

寄存器 3：看门狗控制(WDTCTL)，偏移量 0x008

该寄存器是看门狗控制寄存器。可以将看门狗定时器配置为产生复位信号（在第二次超时时）或者是在超时的时候产生中断。

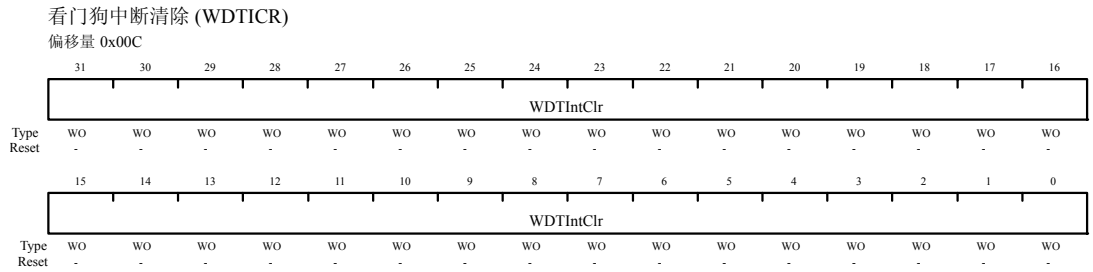
在看门狗中断已经被使能的情况下，之后写入控制寄存器的所有值都将被忽略。而硬件复位是重新使能写操作的唯一方法。



位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
1	RESEN	R/W	0	看门狗复位使能。 0：禁能。 1：使能看门狗模块复位输出。
0	INTEN	R/W	0	看门狗中断使能。 0：中断事件禁能（一旦该位被置位，则只能通过硬件复位来清零该位）。 1：中断事件使能。一旦被使能，之后所有的写操作都会被忽略。

寄存器 4：看门狗中断清除(WDTICR)，偏移量 0x00C

该寄存器是中断清除寄存器。向该寄存器写任意值将清除看门狗中断，并且将 **WDTLOAD** 寄存器所保存的计数值重新载入到 32 位计数器中。（该寄存器的）读取值或者复位后的值无法确定。

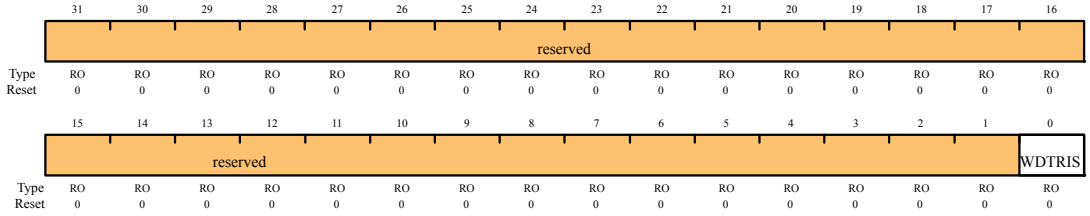


位/字段	名称	类型	复位	描述
31:0	WDTIntClr	WO	-	清除看门狗中断。

寄存器 5：看门狗原始中断状态(WDTRIS)，偏移量 0x010

该寄存器是原始中断状态寄存器。如果控制器中断被屏蔽，则通过该寄存器可监控看门狗中断事件。

看门狗原始中断状态 (WDTRIS)
偏移量 0x010

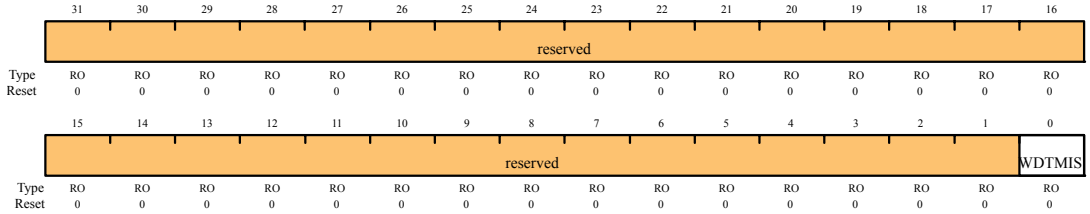


位/字段	名称	类型	复位	描述
31:1	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
0	WDTRIS	RO	0	看门狗原始中断状态。 给出 **WDTINTR** 的原始中断状态（在屏蔽之前）。

寄存器 6：看门狗屏蔽后中断状态(WDTMIS)，偏移量 0x014

该寄存器是经过屏蔽后的中断状态寄存器。该寄存器的值是将原始中断位和看门狗中断使能位进行逻辑与运算(AND)的结果。

看门狗已屏蔽的中断状态 (WDTMIS)
偏移量 0x014

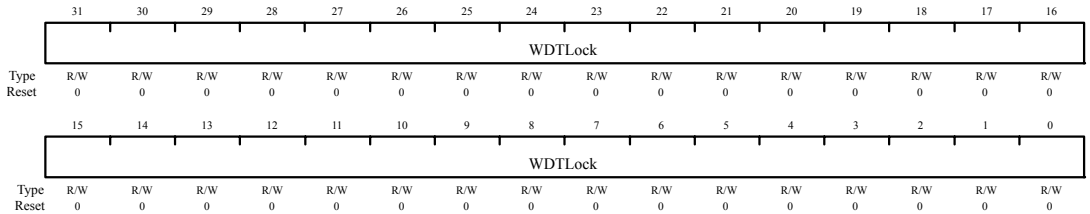


位/字段	名称	类型	复位	描述
31:1	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
0	WDTMIS	RO	0	看门狗屏蔽后的中断状态。 给出 **WDTINTR** 中断在屏蔽后的中断状态。

寄存器 7：看门狗锁定(WDTLOCK)，偏移量 0xC00

把 0x1ACCE551 写入 **WDTLOCK** 寄存器可以使能对其它所有寄存器的写访问。把任意值写入到 **WDTLOCK** 寄存器可重新使能锁定的状态。读 **WDTLOCK** 寄存器时返回的是锁定的状态而不是被写入的 32 位值。因此，当写入访问被禁止时，读取 **WDTLOCK** 寄存器将返回 0x00000001(这是在已锁定的情况下；否则，返回的值为 0x00000000(未锁定))。

看门狗锁定 (WDTLOCK)
偏移量 0xC00

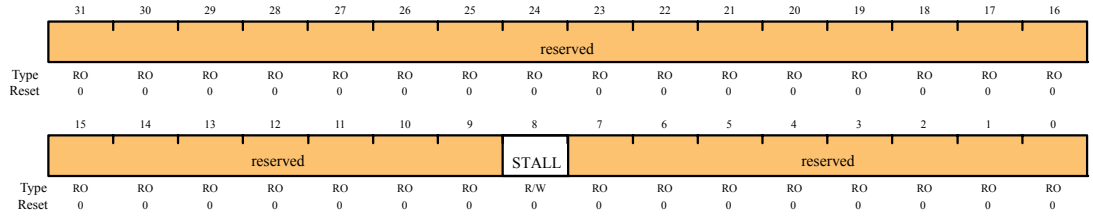


位/字段	名称	类型	复位	描述
31:0	WDTLock	R/W	0x0000	看门狗锁定。 写入值 0x1ACCE551 解锁看门狗寄存器来执行写访问。写入其它值则重新应用锁定，以防止更新任何寄存器。 读该寄存器返回下面的值： 已锁定的：0x00000001 未锁定的：0x00000000

寄存器 8：看门狗测试(WDTTEST)，偏移量 0x418

进行调试期间，当微控制器使 CPU 的暂停（Halt）标志有效时的暂停操作（stalling）可由用户通过该寄存器来控制使能

看门狗测试 (WDTTEST)
偏移量 0x418

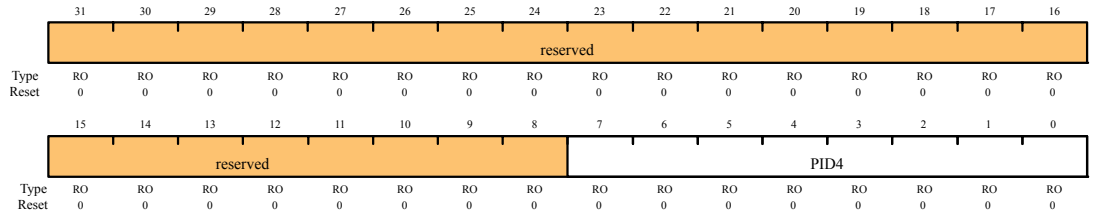


位/字段	名称	类型	复位	描述
31:9	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
8	STALL	R/W	0	看门狗中止使能。 当设为 1 时，如果调试器使 Stellaris 微控制器停止，则看门狗定时器也会停止计数。而一旦微控制器重新启动，则看门狗定时器也会恢复计数。
7:0	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。

寄存器 9：看门狗外设标识 4 (WDTPeriphID4)，偏移量 0xFD0

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 4 (WDTPeriphID4)
偏移量 0xFD0

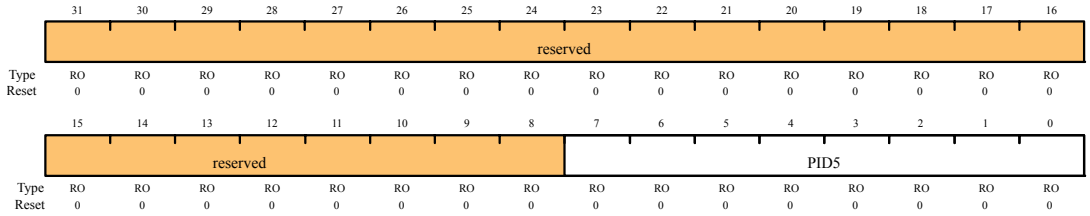


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID4	RO	0x00	WDT 外设 ID 寄存器[7:0]。

寄存器 10：看门狗外设标识 5 (WDTPeriphID5)，偏移量 0xFD4

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 5 (WDTPeriphID5)
偏移量 0xFD4

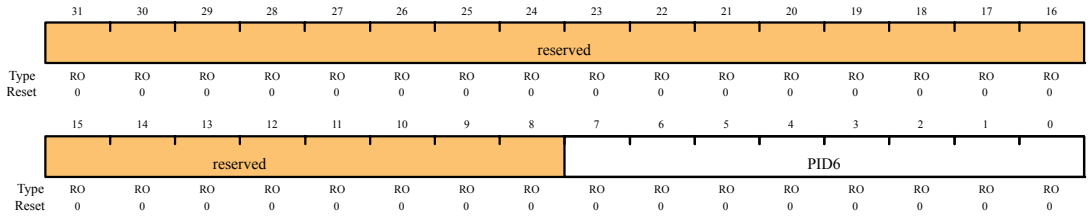


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID5	RO	0x00	WDT 外设 ID 寄存器[15:8]。

寄存器 11：看门狗外设标识 6(WDTPeriphID6)，偏移量 0xFD8

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 6 (WDTPeriphID6)
偏移量 0xFD8

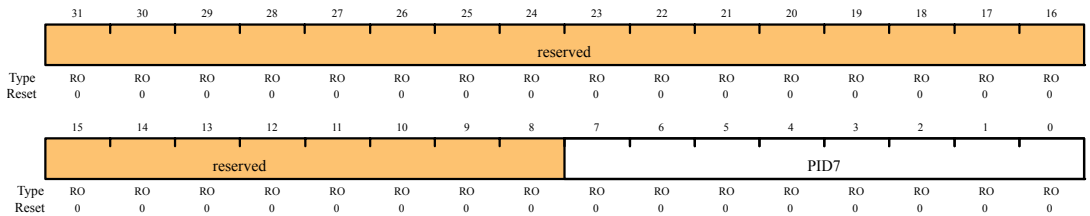


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID5	RO	0x00	WDT 外设 ID 寄存器[23:16]。

寄存器 12：看门狗外设标识 7 (WDTPeriphID7)，偏移量 0xFDC

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 7 (WDTPeriphID7)
偏移量 0xFDC

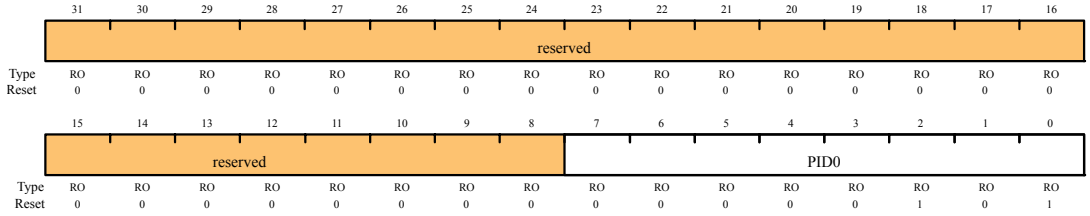


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID7	RO	0x00	WDT 外设 ID 寄存器[31:24]。

寄存器 13：看门狗外设标识 0 (WDTPeriphID0)，偏移量 0xFE0

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 0 (WDTPeriphID0)
偏移量 0xFE0

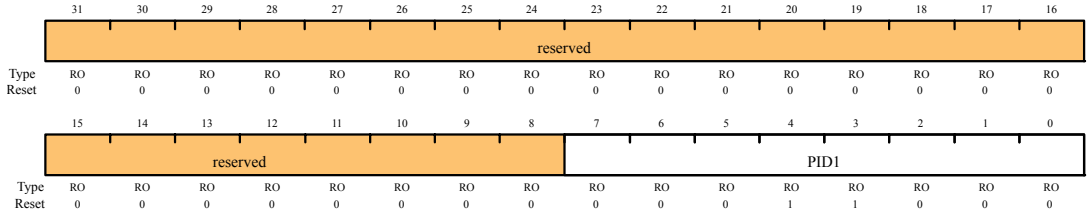


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID0	RO	0x05	看门狗外设 ID 寄存器[7:0]。

寄存器 14：看门狗外设标识 1(WDTPeriphID1)，偏移量 0xFE4

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 1 (WDTPeriphID1)
偏移量 0xFE4

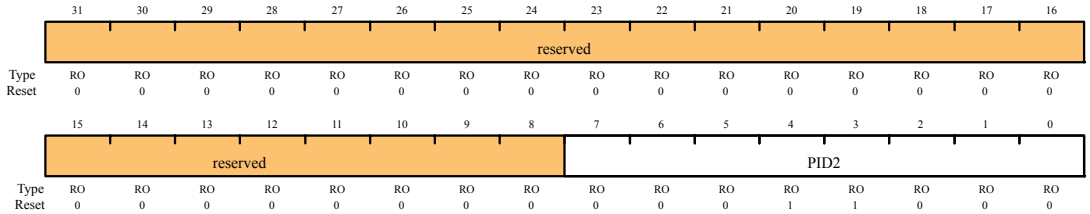


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID1	RO	0x18	看门狗外设 ID 寄存器[15:8]。

寄存器 15：看门狗外设标识 2 (WDTPeriphID2)，偏移量 0xFE8

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 2 (WDTPeriphID2)
偏移量 0xFE8

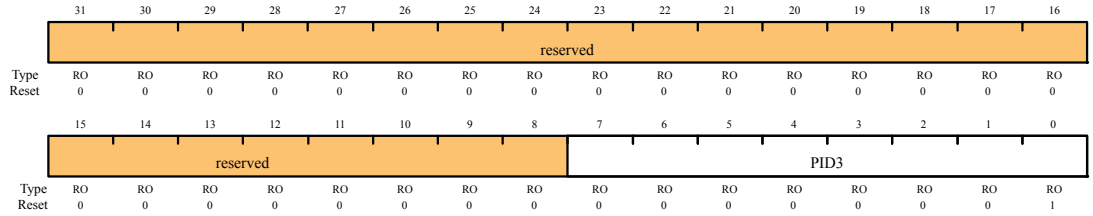


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID2	RO	0x18	看门狗外设 ID 寄存器[23:16]。

寄存器 16：看门狗外设标识 3 (WDTPeriphID3)，偏移量 0xFEC

WDTPeriphIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗外设标识 3 (WDTPeriphID3)
偏移量 0xFEC

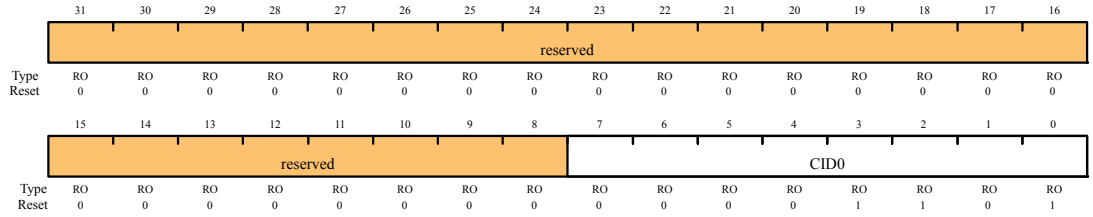


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	PID3	RO	0x01	看门狗外设 ID 寄存器[31:24]。

寄存器 17：看门狗 PrimeCell 标识 0 (WDTPCellID0)，偏移量 0xFF0

WDTPCellIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗PrimeCell标识 0 (WDTPCellID0)
偏移量 0xFF0

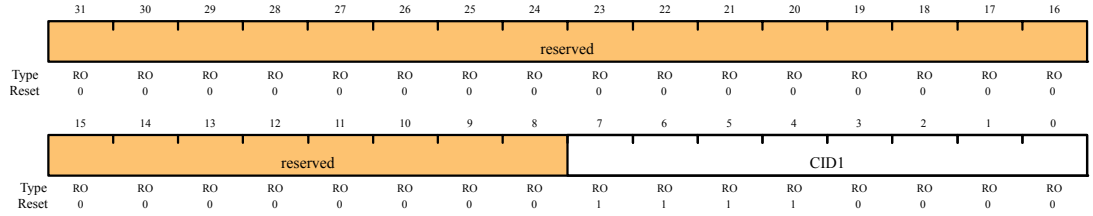


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	CID0	RO	0x0D	看门狗 PrimeCell ID 寄存器[7:0]。

寄存器 18：看门狗 PrimeCell 标识 1 (WDTPCellID1)，偏移量 0xFF4

WDTPCellIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗PrimeCell标识 1 (WDTPCellID1)
偏移量 0xFF4

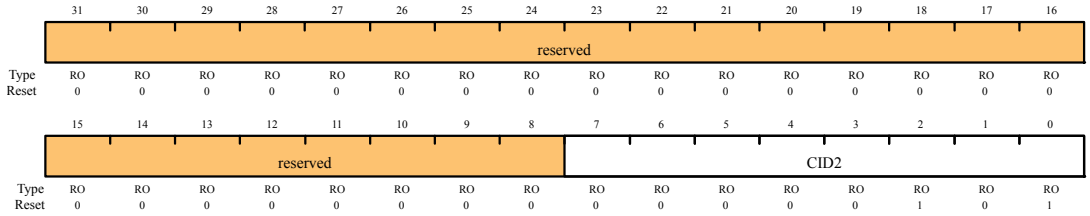


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	CID1	RO	0xF0	看门狗 PrimeCell ID 寄存器[15:8]。

寄存器 19：看门狗 PrimeCell 标识 2 (WDTPCellID2)，偏移量 0xFF8

WDTPCellIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗PrimeCell标识 2 (WDTPCellID2)
偏移量 0xFF8

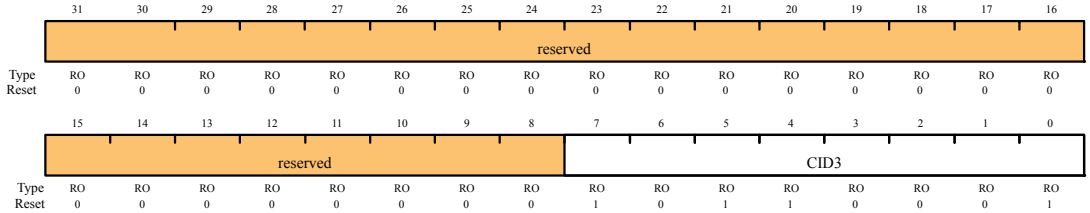


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	CID2	RO	0x05	看门狗 PrimeCell ID 寄存器[23:16]。

寄存器 20：看门狗 PrimeCell 标识 3 (WDTPCellID3)，偏移量 0xFFC

WDTPCellIDn 寄存器为硬编码（hard-coded），寄存器内的字段决定了复位值。

看门狗PrimeCell标识 3 (WDTPCellID3)
偏移量 0xFFC



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7:0	CID3	RO	0xB1	看门狗 PrimeCell ID 寄存器[31:24]。

第11章 模数转换器（ADC）

模数转换器（ADC）外设用于将连续的模拟电压转换成离散的数字量。

群星（Stellaris）ADC 模块的转换分辨率为 10 位，含 6 个输入通道和一个内部温度传感器。ADC 模块包含一个可编程的序列发生器，无需控制器参与，允许对多个模拟输入源进行采样。每个采样序列均可以灵活的编程，其输入源、触发事件、中断的发生和序列优先级都是可配置的。

群星（Stellaris）ADC 含以下特性：

- 6 个 10 位模拟输入通道
- 单端和差分（differential）输入配置
- 内部温度传感器
- 采样率：500,000 采样/秒（每秒采样 500,000 次）
- 4 个可编程的采样转换序列，1 到 8 个入口，每个序列均带有相应的转换结果 FIFO
- 灵活的触发控制
  - 控制器（软件）
  - 定时器
  - 模拟比较器
  - PWM
  - GPIO
- 硬件可对多达 64 个采样进行平均计算，从而实现提高精度的目的

11.1 流程图

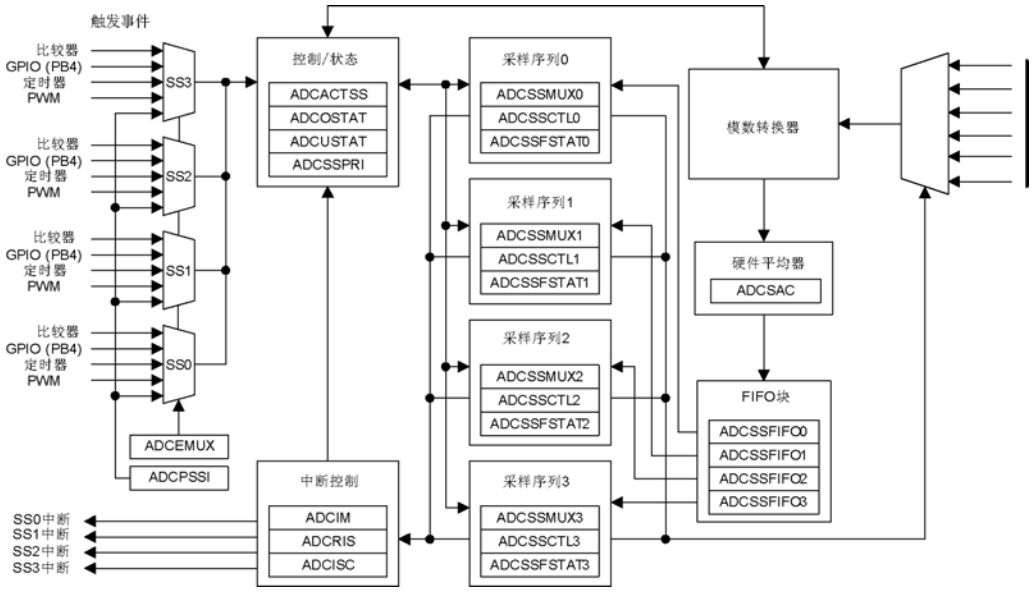


图 11.1 ADC 模块流程图

11.2 功能描述

传统的 ADC 模块大多采用单次采样或双采样的方法收集采样数据，而 Stellaris ADC 模块却不同，它采用的是一种基于序列（sequence-based）的可编程方法。采样序列均为一系列程序化的连续（背对背）采样，使得 ADC 可以从多个输入源中收集数据，而无需控制器对它进行重新配置或处理。对采样序列内的采样进行编程，包括对某些参数进行编程，如输入源和输入模式（差分输入还是单端输入），采样结束时的中断产生机制，以及指示序列最后一个采样的指示符（indicator）。

11.2.1 采样序列发生器（sample sequencer）

采样控制与数据捕获由“采样序列发生器”进行处理。所有序列发生器的实现方法都相同，不同的只是各自可以捕获的采样数目和FIFO深度。表 11.1 给出了每个序列发生器可以捕获的最大采样数以及其相应的FIFO深度。在本实现方案中，每个FIFO入口均为 32 位（1 个字），低 10 位包含的是转换结果。

表 11.1 序列发生器的采样数和 FIFO 深度

序列发生器	采样数	FIFO 深度
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

对于一个给定的采样序列，每个采样均通过 **ADC 采样序列输入多路复用器选择（ADCSSMUXn）**寄存器的 4 个位和 **ADC 采样序列控制（ADCSSCTLn）**寄存器的 4 个位（半字节）进行定义，此处，“n”对应的是序列编号。**ADCSSMUXn** 的半字节用于选择输入管脚，**ADCSSCTLn** 半字节是采样控制位，这些控制位分别与参数（如温度传感器的选择、中断使能、序列末端和差分输入模式）对应。采样序列发生器虽然可以通过置位 **ADC 活动采样序列发生器（ADCACTSS）**寄存器的相应位进行使能，但也可以在使能前进行配置。

在对采样序列进行配置时，允许同一序列内的相同输入管脚有多种用法。在 **ADCSSCTLn** 寄存器中，任意采样组合的 Interrupt Enable（IE）位均可以置位，使得中断在必要的时候可以在每个采样序列结束后产生中断。同样，END 位也可以在采样序列的任何断点处（point）置位。例如，如果使用的是序列发生器 0，那么 END 位可以在第五个采样的半字节处置位，使得序列发生器 0 可以在第五个采样之后结束采样序列。

结束采样序列后，可以从 **ADC 采样序列结果 FIFO（ADCSSFIFOn）**寄存器中读取结果数据。所有的 FIFO 均为简单的环形缓冲器，该缓冲器通过读取 ADCSSFIFOn 地址来“获取（pop）”结果数据。为了进行软件调试，FIFO 头指针和尾指针的位置，连同 FULL 和 EMPTY 状态标志都可以在 **ADC 采样序列 FIFO 状态（ADCSSFSTATn）**寄存器中找到。上溢和下溢条件也可以通过 **ADCOSTAT** 和 **ADCUSTAT** 寄存器进行监控。

11.2.2 模块控制

在采样序列发生器的外面，控制逻辑的剩余部分负责如中断产生、序列优先级设置以及触发配置等任务。

大多数的 ADC 控制逻辑都是在 14-18MHz 的 ADC 时钟速率下运行。当选择了系统 XTAL 时，内部 ADC 分频器通过硬件自动配置。所有的 Stellaris 器件，其自动时钟分频器的配置均以 16.667MHz 操作频率为目标。

11.2.2.1 中断

采样序列发生器虽然会对引起中断的事件进行检测，但是不必控制中断是否真正发送到中断控制器。ADC 模块的中断信号由 **ADC 中断屏蔽（ADCIM）**寄存器的 MASK 位控制。中断状态可以在两个位置观察到，一个是 **ADC 原始中断状态（ADCRIS）**寄存器，它显示的是采样序列发生器的中断信号的原始状态；另外一个是 **ADC 中断状态和清除（ADCISC）**寄存器，它显示的是 **ADCRIS** 寄存器的 INR 位与 **ADCIM** 寄存器的 MASK 位进行逻辑与所得到的结果。中断通过写 1 到 **ADCISC** 中对应的 IN 位进行清除。

11.2.2.2 优先级设置

在同时出现多个采样事件（触发）时，**ADC 采样序列发生器优先级（ADCSSPRI）**寄存器的值将为这些事件设置优先级，安排它们的处理顺序。优先级值的有效范围是 0 到 3，其中 0 代表优先级最高，而 3 代表优先级最低。如果多个活动采样序列发生器单元的优先级都相同，它们通常不会提供一致（consistent）的结果，所以软件必须确保活动采样序列发生器的优先级是唯一的。

11.2.2.3 采样事件

每个采样序列发生器的采样触发（sample triggering）在 **ADC 事件多路复用器选择（ADCEMUX）**寄存器中定义。外部的外设触发源随着 Stellaris 系列器件的变化而改变，但是所有器件均复用“控制器”和“一直（always）”触发器。软件通过置位 **ADC 处理器采样序列启动（ADCPSSI）**寄存器的 CH 位来启动采样。

在使用“一直（always）”触发器时必须非常小心。序列的优先级如果太高，可能会忽略其他低优先级序列。

11.2.3 硬件采样平均电路

通过使用硬件平均电路，可以获得更高的精度。但这却是以牺牲吞吐量（throughput）为代价。我们可以将多达 64 个采样累加起来，然后取它们的平均值，这样在序列发生器 FIFO 内就只得到一个数据入口。根据进行平均计算的采样数，吞吐量会按比例相应地减少。例如，如果我们将平均电路配置为对 16 个采样进行平均计算，那么吞吐量会以 16 倍数递减。

默认情况下，平均电路是断开的，并且转换器的所有数据都会传送到序列发生器 FIFO。平均硬件电路由 **ADC 采样平均控制(ADCSAC)**寄存器控制。平均电路只有一个，并且不管是单端输入还是差分输入，所有输入通道均接收相同采样数量的平均值。

11.2.4 模数转换器

转换器本身会为所选模拟输入产生 10 位输出值。通过某些特定的模拟端口，输入的失真可以降到最低。

11.2.5 测试模式

ADC 模块的测试模式允许在 ADC 模块的数字部分内执行回送（loopback）操作。这在调试软件中非常有用，因为无需提供真实的模拟激励信号（analog stimulus）。**ADC 测试**

模式回送（ADCTMLB）寄存器中对该测试模式进行了描述。

11.2.6 内部温度传感器

内部温度传感器提供模拟温度读取操作和参考电压。输出终端 SENSO 的电压通过以下等式计算得到：

SENSO ＝ 2.7 －（（T+55）／ 75）

图 11.2 用图形的方式显示了上式中各参数之间的关系。

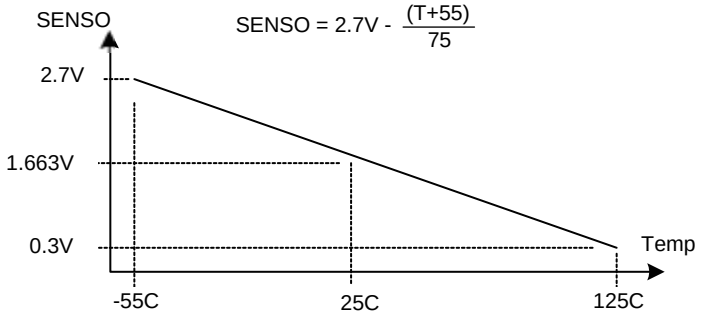


图 11.2 内部温度传感器特性

11.3 初始化和配置

为了使用 ADC 模块，必须使能 PLL，同时使用所支持的晶振频率（见 **RCC** 寄存器的描述）。使用不支持的频率可能会导致 ADC 模块的的运作发生错误。

11.3.1 模块初始化

ADC 模块的初始化过程很简单，只需几个步骤。主要的步骤包括使能 ADC 时钟和配置采样序列发生器的优先级（如有必要）。

ADC 初始化的顺序如下：

1. 通过写 0x00010000 到系统控制模块中的 **RCGC1** 寄存器将 ADC 时钟使能
2. 一经应用要求，必须在 **ADCSSPRI** 寄存器中对采样序列发生器的优先级进行配置。默认的配置是采样序列发生器 0 优先级最高，采样序列发生器 3 优先级最低。

11.3.2 采样序列发生器的配置

采样序列发生器的配置要稍微比模块初始化的过程复杂，因为每个采样序列是完全可编程的。

每个采样序列发生器按照以下步骤进行配置：

1. 保证采样序列发生器被禁能，这可以通过写 0 到 **ADCACTSS** 寄存器中对应的 ASEN 位来实现。采样序列发生器无需使能就可编程。如果在配置过程发生触发事件，那么在编程过程中禁能序列发生器就可以预防发生错误的执行操作。
2. 为 **ADCEMUX** 寄存器中的采样序列发生器配置触发事件。
3. 在 **ADCSSMUXn** 寄存器中为采样序列中的每个采样配置相应的输入源。
4. 在 **ADCSSCTLn** 寄存器中为采样序列中的每个采样配置采样控制位。在对最后半个字节进行编程时，确保 END 位已置位。不过置位 END 位失败可能会引发不可

预测的行为。

5. 如果要使用中断，必须写 1 到 **ADCIM** 寄存器中相应的 MASK 位。
6. 通过写 1 到 **ADCACTSS** 寄存器中相应的 ASEN 位将采样序列发生器逻辑使能。

11.4 寄存器映射

表 11.2 是ADC寄存器的汇总。表中的偏移量是寄存器地址相对于ADC基址0x40038000的十六进制增量。

表 11.2 ADC 寄存器映射

偏移量	名称	复位	类型	描述
0x000	ADCACTSS	0x00000000	R/W	活动采样序列发生器
0x004	ADCRIS	0x00000000	RO	原始中断状态和清除
0x008	ADCIM	0x00000000	R/W	中断屏蔽
0x00C	ADCISC	0x00000000	R/W1C	中断状态和清除
0x010	ADCOSTAT	0x00000000	R/W1C	上溢状态
0x014	ADCEMUX	0x00000000	R/W	事件多路复用器选择
0x018	ADCUSTAT	0x00000000	R/W1C	下溢状态
0x020	ADCSSPRI	0x00003210	R/W	采样序列发生器优先级
0x028	ADCPSSI	-	WO	处理器采样序列启动
0x030	ADCSAC	0x00000000	R/W	采样平均控制
0x040	ADCSSMUX0	0x00000000	R/W	采样序列输入多路复用器选择 0
0x044	ADCSSCTL0	0x00000000	R/W	采样序列控制 0
0x048	ADCSSFIFO0	0x00000000	RO	采样序列结果 FIFO 0
0x04C	ADCSSFSTAT0	0x00000100	RO	采样序列 FIFO 0 状态
0x060	ADCSSMUX1	0x00000000	R/W	采样序列输入多路复用器选择 1
0x064	ADCSSCTL1	0x00000000	R/W	采样序列控制 1
0x068	ADCSSFIFO1	0x00000000	RO	采样序列结果 FIFO 1
0x06C	ADCSSFSTAT1	0x00000100	RO	采样序列 FIFO 1 状态
0x080	ADCSSMUX2	0x00000000	R/W	采样序列输入多路复用器选择 2
0x084	ADCSSCTL2	0x00000000	R/W	采样序列控制 2
0x088	ADCSSFIFO2	0x00000000	RO	采样序列结果 FIFO 2
0x08C	ADCSSFSTAT2	0x00000100	RO	采样序列 FIFO 2 状态
0x0A0	ADCSSMUX3	0x00000000	R/W	采样序列输入多路复用器选择 3
0x064	ADCSSCTL3	0x00000002	R/W	采样序列控制 3
0x0A8	ADCSSFIFO3	0x00000000	RO	采样序列结果 FIFO 3
0x0AC	ADCSSFSTAT3	0x00000100	RO	采样序列 FIFO 3 状态

续上表

偏移量	名称	复位	类型	描述
0x100	ADCTMLB	0x00000000	R/W	测试模式回送（loopback）

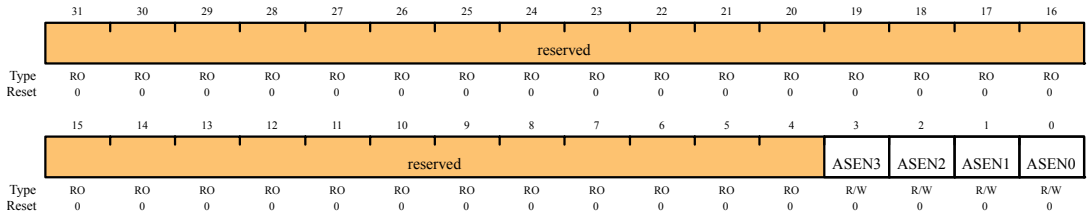
11.5 寄存器描述

下文将按照地址偏移量的数字顺序列出 ADC 寄存器，并对这些寄存器进行描述。

寄存器 1：ADC 活动采样序列发生器（ADCACTSS），偏移量 0x000

该寄存器控制采样序列发生器是否激活。每个采样序列发生器都可以单独地使能和禁能。

ADC活动取样序列发生器（ADCACTSS）寄存器
偏移量 0x000

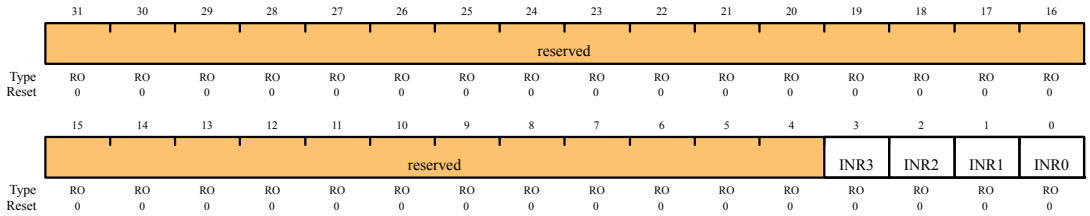


位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	ASEN3	R/W	0	确定采样序列发生器 3 是否使能。如果置位，序列发生器 3 采样序列逻辑则激活。否则将不激活
2	ASEN2	R/W	0	确定采样序列发生器 2 是否使能。如果置位，序列发生器 2 采样序列逻辑则激活。否则将不激活
1	ASEN1	R/W	0	确定采样序列发生器 1 是否使能。如果置位，序列发生器 1 采样序列逻辑则激活。否则将不激活
0	ASEN0	R/W	0	确定采样序列发生器 0 是否使能。如果置位，序列发生器 0 采样序列逻辑则激活。否则将不激活

寄存器 2：ADC 原始中断状态（ADCRIS），偏移量 0x004

该寄存器显示了每个采样序列发生器的原始中断信号的状态。软件可以通过直接轮询这些位来查找中断条件，而无需产生控制器中断。

ADC原始中断状态（ADCRIS）寄存器
偏移量 0x004

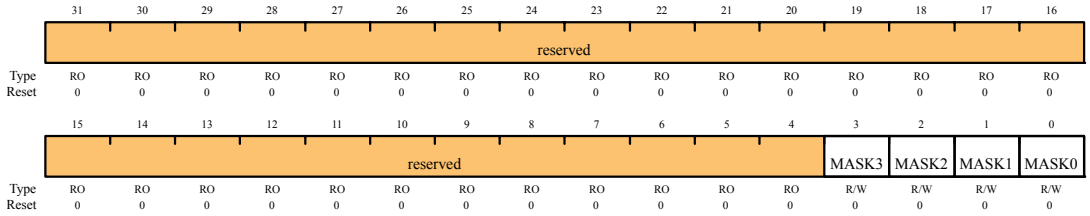


位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	INR3	RO	0	当某个采样其对应的 **ADCSSCTAL3** 中的 IE 位完成转换后，该位被硬件置位。该位通过写 1 到 **ADCISC** 的 IN3 位被清零
2	INR2	RO	0	当某个采样其对应的 **ADCSSCTAL2** 中的 IE 位完成转换后，该位被硬件置位。该位通过写 1 到 **ADCISC** 的 IN2 位被清零
1	INR1	RO	0	当某个采样其对应的 **ADCSSCTAL1** 中的 IE 位完成转换后，该位被硬件置位。该位通过写 1 到 **ADCISC** 的 IN1 位被清零
0	INR0	RO	0	当某个采样其对应的 **ADCSSCTAL0** 中的 IE 位完成转换后，该位被硬件置位。该位通过写 1 到 **ADCISC** 的 IN0 位被清零

寄存器 3：ADC 中断屏蔽（ADCIM），偏移量 0x008

该寄存器决定是否将采样序列发生器的原始中断信号提交给控制器中断。每个采样序列发生器的原始中断信号均可以单独地被屏蔽。

ADC中断屏蔽（ADCIM）寄存器
偏移量 0x008

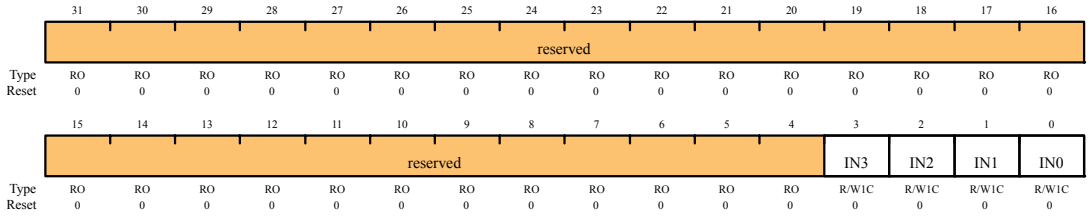


位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	MASK3	R/W	0	确定是否将采样序列发生器 3 的原始中断信号（**ADCRIS** 寄存器中的 INR3 位）提交给控制器中断。如果置位，原始中断信号将被提交给控制器中断。否则，不提交
2	MASK2	R/W	0	确定是否将采样序列发生器 2 的原始中断信号（**ADCRIS** 寄存器中的 INR2 位）提交给控制器中断。如果置位，原始中断信号将被提交给控制器中断。否则，不提交
1	MASK1	R/W	0	确定是否将采样序列发生器 1 的原始中断信号（**ADCRIS** 寄存器中的 INR1 位）提交给控制器中断。如果置位，原始中断信号将被提交给控制器中断。否则，不提交
0	MASK0	R/W	0	确定是否将采样序列发生器 0 的原始中断信号（**ADCRIS** 寄存器中的 INR0 位）提交给控制器中断。如果置位，原始中断信号将被提交给控制器中断。否则，不提交

寄存器 4：ADC 中断状态和清除（ADCISC），偏移量 0x00C

该寄存器不仅提供了一种清除中断条件的机制，还可以用于显示由采样序列发生器产生的控制器中断的状态。读取该寄存器时，每个位的值均表示各自的 INR 和 MASK 位进行逻辑与（logic AND）的结果。向相应的位写 1 可以将该中断清除。如果软件是在轮询 **ADCRIS**，而不是产生中断，那么即使 IN 位没有置位，INR 位仍可以通过 **ADCISC** 寄存器清零。

ADC中断状态和清除（ADCISC）寄存器
偏移量 0x00C

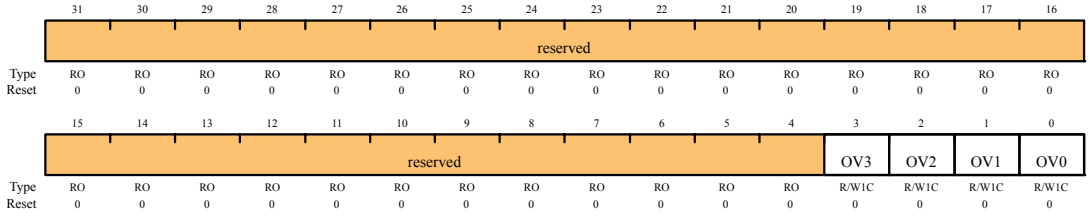


位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	IN3	R/W1C	0	当 MASK3 和 INR3 都为 1 时，该位被硬件置位，向控制器提供一个基于电平（level-based）的中断。该中断可以通过写 1 到 IN3 位来清除，同时 INR3 位也将清零。
2	IN2	R/W1C	0	当 MASK2 和 INR2 都为 1 时，该位被硬件置位，向控制器提供一个基于电平（level-based）的中断。该中断可以通过写 1 到 IN2 位来清除，同时 INR2 位也将清零。
1	IN1	R/W1C	0	当 MASK1 和 INR1 都为 1 时，该位被硬件置位，向控制器提供一个基于电平（level-based）的中断。该中断可以通过写 1 到 IN1 位来清除，同时 INR1 位也将清零
0	IN0	R/W1C	0	当 MASK0 和 INR0 都为 1 时，该位被硬件置位，向控制器提供一个基于电平（level-based）的中断。该中断可以通过写 1 到 IN0 位来清除，同时 INR0 位也将清零。

寄存器 5：ADC 上溢状态（ADCOSTAT），偏移量 0x010

该寄存器用来指示采样序列发生器 FIFO 中的上溢条件。上溢条件一经软件处理后，就可以通过写 1 到相应的位被清除。

ADC上溢状态（ADCOSTAT）寄存器
偏移量 0x010



位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	OV3	R/W1C	0	该位确定采样序列发生器 3 的 FIFO 是否出现上溢条件（FIFO 满且要求写操作）。当探测到溢出条件时，取消最近的写操作，且该位被硬件置位以指示发生丢弃数据。该位通过写 1 清零。
2	OV2	R/W1C	0	该位确定采样序列发生器 2 的 FIFO 是否出现上溢条件（FIFO 满且要求写操作）。当探测到溢出条件时，取消最近的写操作，且该位被硬件置位以指示发生丢弃数据。该位通过写 1 清零。

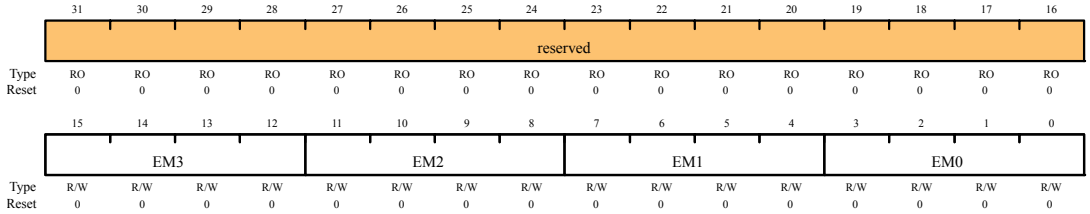
续上表

位	名称	类型	复位	描述
1	OV1	R/W1C	0	该位确定采样序列发生器 1 的 FIFO 是否出现上溢条件（FIFO 满且要求写操作）。当探测到溢出条件时，取消最近的写操作，且该位被硬件置位以指示发生丢弃数据。该位通过写 1 清零。
0	OV0	R/W1C	0	该位确定采样序列发生器 0 的 FIFO 是否出现上溢条件（FIFO 满且要求写操作）。当探测到溢出条件时，取消最近的写操作，且该位被硬件置位以指示发生丢弃数据。该位通过写 1 清零。

寄存器 6：ADC 事件多路复用器选择（ADCEMUX），偏移量 0x014

ADCEMUX 为每个采样序列发生器选择用来启动采样的事件（触发器）。每个采样序列发生器均可以使用唯一的触发源进行配置。

ADC事件多路复用器选择（ADCEMUX）寄存器
偏移量 0x014

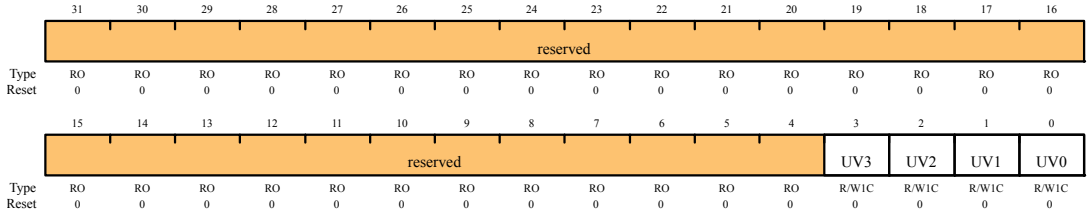


位	名称	类型	复位	描述
31:16	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
15:12	EM3	R/W	0	该位用于选择采样序列发生器 3 的触发源。该位的有效配置如下： EM 二进制值 / 事件 0000 控制器（缺省） 0001 模拟比较器 0 0010 保留 0011 保留 0100 外部（GPIO PB4） 0101 定时器 0110 PWM0 0111 PWM1 1000 PWM2 1001-1110 保留 1111 一直（Always）（连续采样）
11:8	EM2	R/W	0	该位用于选择采样序列发生器 2 的触发源。编码与 EM3 相同
7:4	EM1	R/W	0	该位用于选择采样序列发生器 1 的触发源。编码与 EM3 相同
3:0	EM0	R/W	0	该位用于选择采样序列发生器 0 的触发源。编码与 EM3 相同

寄存器 7：ADC 下溢状态（ADCUSTAT），偏移量 0x018

该寄存器用来指示采样序列发生器 FIFO 中的下溢条件。下溢条件一经软件处理后，就可以通过写 1 到相应的位被清除。

ADC下溢状态（ADCUSTAT）寄存器
偏移量 0x018

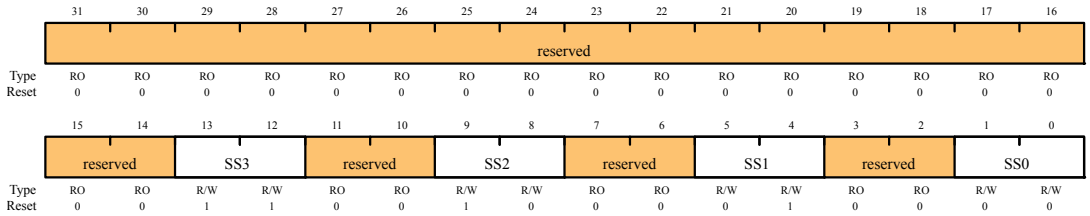


位	名称	类型	复位	描述
31:4	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
3	UV3	R/W1C	0	该位确定采样序列发生器 3 的 FIFO 是否出现下溢条件（FIFO 空且要求读操作）。这个有问题的读操作不会移动 FIFO 指针，且返回 0。该位通过写 1 清零。
2	UV2	R/W1C	0	该位确定采样序列发生器 2 的 FIFO 是否出现下溢条件（FIFO 空且要求读操作）。这个有问题的读操作不会移动 FIFO 指针，且返回 0。该位通过写 1 清零。
1	UV1	R/W1C	0	该位确定采样序列发生器 1 的 FIFO 是否出现下溢条件（FIFO 空且要求读操作）。这个有问题的读操作不会移动 FIFO 指针，且返回 0。该位通过写 1 清零。
0	UV0	R/W1C	0	该位确定采样序列发生器 0 的 FIFO 是否出现下溢条件（FIFO 空且要求读操作）。这个有问题的读操作不会移动 FIFO 指针，且返回 0。该位通过写 1 清零。

寄存器 8：ADC 采样序列发生器优先级（ADCSSPRI），偏移量 0x020

该寄存器为每个采样序列发生器设置优先级。复位后，序列发生器 0 优先级最高，采样序列发生器 3 优先级最低。在配置序列优先级时，每个序列的优先级均必须只有一个，否则 ADC 行为往往不一致（inconsistent）。

ADC取样序列发生器优先级（ADCSSPRI）寄存器
偏移量 0x020

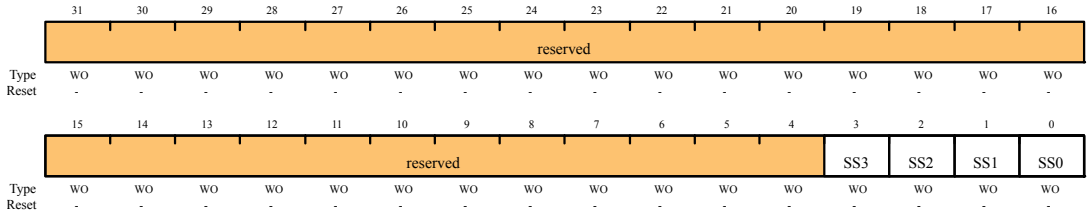


位	名称	类型	复位	描述
31:14	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
13:12	SS3	R/W	0x3	SS3 位包含确定采样序列发生器 3 的优先级编码的二进制编码值。优先级编码 0 优先级最高，3 最低。分配给序列发生器的优先级必须被唯一地映射。如果存在两个或两个以上的位相等，那么 ADC 的行为将不一致（consistent）。
11:10	保留	R/O	0	保留位返回一个不确定值，并且应该永不改变
9:8	SS2	R/W	0x2	SS2 位包含确定采样序列发生器 2 的优先级编码的二进制编码值。
7:6	保留	R/O	0	保留位返回一个不确定值，并且应该永不改变
5:4	SS1	R/W	0x1	SS1 位包含确定采样序列发生器 1 的优先级编码的二进制编码值。
3:2	保留	RO	0	保留位返回一个不确定值，并且应该永不改变
1:0	SS0	R/W	0x0	SS0 位包含确定采样序列发生器 0 的优先级编码的二进制编码值。

寄存器 9：ADC 处理器采样序列发生器启动（ADCPSSI），偏移量 0x028

该寄存器为应用软件提供了一种机制，以便在采样序列发生器中启动采样。采样序列可以单独启动，也可以组合启动。当多个序列同时触发时，**ADCSSPRI** 中的优先级编码对执行顺序进行检测。

ADC处理器样本序列启动（ADCPSSI）寄存器
偏移量 0x028



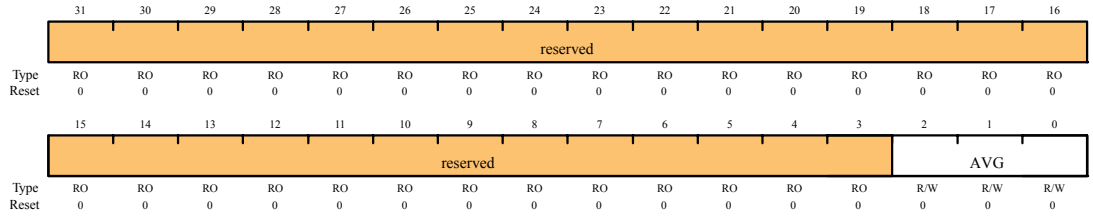
位	名称	类型	复位	描述
31:4	保留	WO	-	只有软件执行的写操作才有效；读取该寄存器将返回无意义的数据
3	SS3	WO	-	只有软件执行的写操作才有效；读取该寄存器将返回无意义的数据。在被软件置位时，采样序列发生器 3 触发采样，假定序列发生器在 **ADCACTSS** 寄存器中被使能。
2	SS2	WO	-	只有软件执行的写操作才有效；读取该寄存器将返回无意义的数据。在被软件置位时，采样序列发生器 2 触发采样，假定序列发生器在 **ADCACTSS** 寄存器中被使能。
1	SS1	WO	-	只有软件执行的写操作才有效；读取该寄存器将返回无意义的数据。在被软件置位时，采样序列发生器 1 触发采样，假定序列发生器在 **ADCACTSS** 寄存器中被使能。
0	SS0	WO	-	只有软件执行的写操作才有效；读取该寄存器将返回无意义的数据。在被软件置位时，采样序列发生器 0 触发采样，假定序列发生器在 **ADCACTSS** 寄存器中被使能。

寄存器 10：ADC 采样平均控制(ADCSAC)寄存器，偏移量 0x030

该寄存器用来控制硬件进行平均计算而最终得到转换结果的采样个数。最终的转换结果是以指定的 ADC 速率通过 2^{AVG} 个连续的 ADC 采样进行平均计算而得到的。如果 AVG 等于 0，那么采样将直接通过，而无需进行平均计算。如果 AVG 等于 6，那么在对 64 个

连续的 ADC 采样进行平均计算后，会在序列发生器 FIFO 中得到一个结果。如果 AVG 等于 7 会返回一个不可预测的结果。

ADC采样平均控制(ADCSAC)寄存器
偏移量 0x030



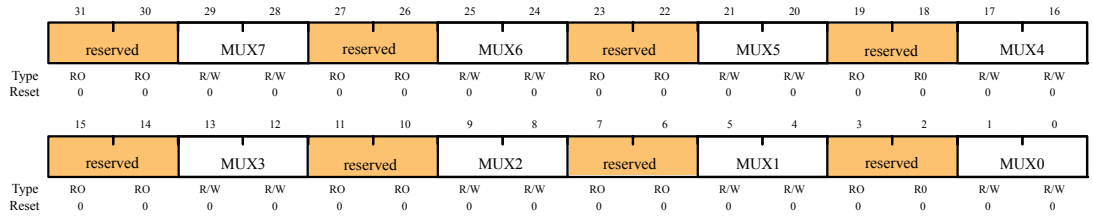
位/字段	名称	类型	复位	描述
31:3	保留	RO	0	保留位，返回一个不确定的值，并且应该永不改变
2:0	AVG	R/W	0	确定平均计算的 ADC 采样个数。AVG 字段可以是 0 到 6 之间的任意值。当它等于 7 时，其结果将不可预测。

寄存器 11：ADC 采样序列输入多路复用器选择 0（ADCSSMUX0），偏移量 0x040

该寄存器为采样序列发生器 0 所执行序列的每个采样进行模拟输入配置。

该寄存器为 32 位宽，并且包含 8 个可能采样的信息。

ADC取样序列输入多路复用器选择0（ADCSSMUX0）寄存器
偏移量 0x040



位	名称	类型	复位	描述
31:30	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
29:28	MUX7	R/W	0	MUX7 位在采样序列发生器 0 所执行序列的第 8 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。此处设置的值指示对应的管脚，例如，1 代表输入为 ADC1。
27:26	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
25:24	MUX6	R/W	0	MUX6 位在采样序列发生器 0 所执行序列的第 7 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
23:22	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
21:20	MUX5	R/W	0	MUX5 位在采样序列发生器 0 所执行序列的第 6 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
19:18	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
17:16	MUX4	R/W	0	MUX4 位在采样序列发生器 0 所执行序列的第 5 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
15:14	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变

续上表

位	名称	类型	复位	描述
13:12	MUX3	R/W	0	MUX3 位在采样序列发生器 0 所执行序列的第 4 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
11:10	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
9:8	MUX2	R/W	0	MUX2 位在采样序列发生器 0 所执行序列的第 3 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
7:6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5:4	MUX1	R/W	0	MUX1 位在采样序列发生器 0 所执行序列的第 2 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。
3:2	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
1:0	MUX0	R/W	0	MUX0 位在采样序列发生器 0 所执行序列的第 1 个采样中使用。它决定在进行模数转换时采样哪个模拟输入。

寄存器 12：ADC 采样序列控制 0（ADCSSCTL0），偏移量 0x044

该寄存器包含采样序列发生器 0 所执行序列的每个采样的配置信息。在对采样序列进行配置时，END 位必须在某些断点（point）处置位，而不管它是在第一个采样之后，还是在最后一个采样后，或者是在任意中间的采样之后。

该寄存器为 32 位宽，并且包含 8 个可能采样的信息。

ADC取样序列控制0（ADCSSCTL0）寄存器
偏移量 0x044

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	名称	类型	复位	描述
31	TS7	R/W	0	TS7 位使用在采样序列的第 8 个采样中，并且可以用来指定采样的输入源。如果置位，则读取温度传感器。否则，读取由 **ADCAMUX** 寄存器指定的输入管脚。
30	IE7	R/W	0	IE7 位使用在采样序列的第 8 个采样中，并且可以用来确定原始中断信号（INR0 位）是否在采样值转换结束时生效。如果 ADCIM 寄存器中的 MASK0 位置位，那么中断被提交给一个控制器级别（controller-level）的中断。当 IE7 位置位时，原始中断发出，否则不发出。允许一个序列内多个采样产生中断。
29	END7	R/W	0	END7 位指示序列的最后一个采样。可以在任意的采样位置结束序列。当一个采样紧跟在包含固定 END 的采样之后被定义，那么它不会请求转换，即使该组的字段（field）为非零。这就要求软件在序列内的某处写 END 位。（序列中仅含一个采样的采样序列发生器 3 被硬连线，以便将 END0 位置位）。 该位置位表示该采样是序列的最后一个采样。

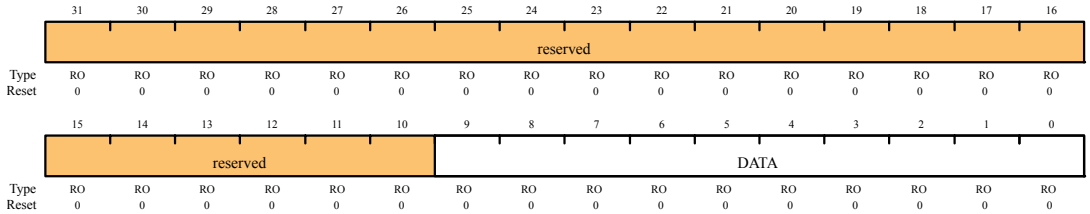
续上表

位	名称	类型	复位	描述
28	D7	R/W	0	D7 位表示模拟输入将被差分采样。相应的 **ADCSSMUXx** 半字节必须成对地配置，这样，成对的输入即为“2i 和 2i+1”。温度传感器不含差分选项。置位时，模拟输入被差分采样。
27	TS6	R/W	0	定义和 TS7 一样，但是使用在第 7 个采样中
26	IE6	R/W	0	定义和 IE7 一样，但是使用在第 7 个采样中
25	END6	R/W	0	定义和 END7 一样，但是使用在第 7 个采样中
24	D6	R/W	0	定义和 D7 一样，但是使用在第 7 个采样中
23	TS5	R/W	0	定义和 TS7 一样，但是使用在第 6 个采样中
22	IE5	R/W	0	定义和 IE7 一样，但是使用在第 6 个采样中
21	END5	R/W	0	定义和 END7 一样，但是使用在第 6 个采样中
20	D5	R/W	0	定义和 D7 一样，但是使用在第 6 个采样中
19	TS4	R/W	0	定义和 TS7 一样，但是使用在第 5 个采样中
18	IE4	R/W	0	定义和 IE7 一样，但是使用在第 5 个采样中
17	END4	R/W	0	定义和 END7 一样，但是使用在第 5 个采样中
16	D4	R/W	0	定义和 D7 一样，但是使用在第 5 个采样中
15	TS3	R/W	0	定义和 TS7 一样，但是使用在第 4 个采样中
14	IE3	R/W	0	定义和 IE7 一样，但是使用在第 4 个采样中
13	END3	R/W	0	定义和 END7 一样，但是使用在第 4 个采样中
12	D3	R/W	0	定义和 D7 一样，但是使用在第 4 个采样中
11	TS2	R/W	0	定义和 TS7 一样，但是使用在第 3 个采样中
10	IE2	R/W	0	定义和 IE7 一样，但是使用在第 3 个采样中
9	END2	R/W	0	定义和 END7 一样，但是使用在第 3 个采样中
8	D2	R/W	0	定义和 D7 一样，但是使用在第 3 个采样中
7	TS1	R/W	0	定义和 TS7 一样，但是使用在第 2 个采样中
6	IE1	R/W	0	定义和 IE7 一样，但是使用在第 2 个采样中
5	END1	R/W	0	定义和 END7 一样，但是使用在第 2 个采样中
4	D1	R/W	0	定义和 D7 一样，但是使用在第 2 个采样中
3	TS0	R/W	0	定义和 TS7 一样，但是使用在第 1 个采样中
2	IE0	R/W	0	定义和 IE7 一样，但是使用在第 1 个采样中
1	END0	R/W	0	定义和 END7 一样，但是使用在第 1 个采样中
0	D0	R/W	0	定义和 D7 一样，但是使用在第 1 个采样中

寄存器 13：ADC 采样序列结果 FIFO 0（ADCSSFIFO0），偏移量 0x048

该寄存器包含的是由采样序列发生器 0 收集的采样的转换结果。读取该寄存器时，将按照采样 0、采样 1，…的顺序返回转换结果数据，直至 FIFO 为空。如果软件没有对 FIFO 进行正确的处理，那么上溢和下溢条件均被记录在 **ADCOSTAT** 和 **ADCUSTAT** 寄存器中。

ADC取样序列结果FIFO 0（ADCSSFIFO0）寄存器
偏移量 0x048

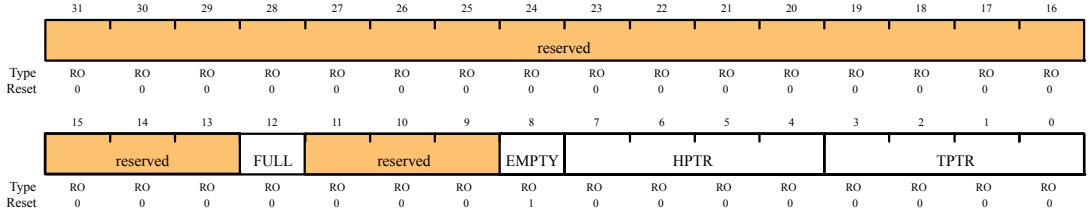


位	名称	类型	复位	描述
31:10	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
9:0	DATA	RO	0	转换结果数据

寄存器 14：ADC 采样序列 FIFO 0 状态（ADCSSFSTAT0），偏移量 0x04C

该寄存器为采样序列发生器 FIFO 0 提供了一个窗口，包括满/空状态信息以及头指针和尾指针的位置。复位值 0x100 表示 FIFO 为空。

ADC取样序列FIFO 0状态（ADCSSFSTAT0）寄存器
偏移量 0x04C



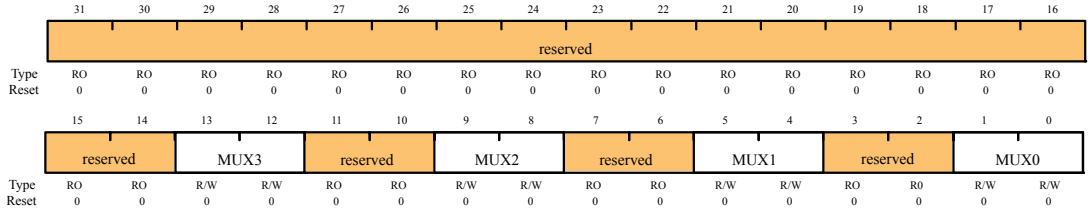
位	名称	类型	复位	描述
31:13	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
12	FULL	RO	0	置位时，表示 FIFO 此时刚满
11:9	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
8	EMPTY	RO	1	置位时，表示 FIFO 此时刚为空
7:4	HPTR	RO	0	HPTR 位包含当前 FIFO 的“头”指针索引（index），即下一个要执行写操作的入口（entry）
3:0	TPTR	RO	0	TPTR 位包含当前 FIFO 的“尾”指针索引（index），即下一个要读取的入口（entry）

寄存器 15：ADC 采样序列输入多路复用器选择 1（ADCSSMUX1），偏移量 0x060

该寄存器为序列发生器 1 所执行序列的每个采样定义模拟输入配置。

该寄存器为 16 位宽，并且包含 4 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSMUX1** 各个位的定义均与 **ADCSSMUX0** 寄存器相同，只不过 **ADCSSMUX1** 是用于采样序列发生器 1。

ADC取样序列输入多路复用器选择1（ADCSSMUX1）寄存器
偏移量 0x060

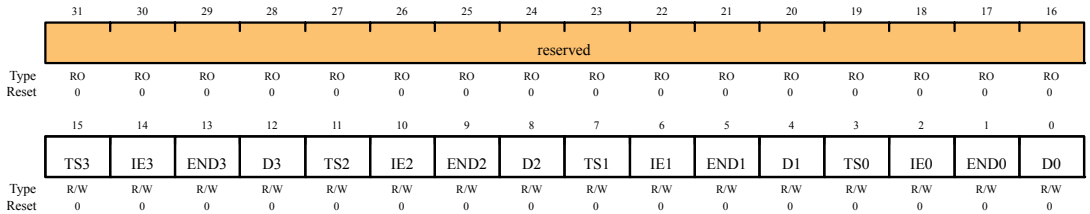


寄存器 16：ADC 采样序列控制 1（ADCSSCTL1），偏移量 0x064

该寄存器包含序列发生器 1 所执行序列的每个采样的配置信息。在对采样序列进行配置时，END 位必须在某些断点（point）处置位，而不管它是在第一个采样之后，还是在最后一个采样后，或者是在任意中间的采样后。

该寄存器为 16 位宽，并且包含 4 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSCTL1** 各个位的定义均与 **ADCSSCTL0** 寄存器相同，只不过 **ADCSSCTL1** 是用于采样序列发生器 1。

ADC取样序列控制1（ADCSSCTL1）寄存器
偏移量 0x064



寄存器 17：ADC 采样序列结果 FIFO 1（ADCSSFIFO1），偏移量 0x068

该寄存器包含的是由采样序列发生器 1 收集的采样的转换结果。读取该寄存器时，将按照采样 0、采样 1，…的顺序返回转换结果数据，直至 FIFO 为空。如果软件没有对 FIFO 进行正确的处理，那么上溢和下溢条件均被记录在 **ADCOSTAT** 和 **ADCUSTAT** 寄存器中。

ADCSSFIFO1 的各个位以及定义均与 **ADCSSFIFO0** 相同，只不过 **ADCSSFIFO1** 是用于 FIFO 1。

寄存器 18：ADC 采样序列 FIFO 1 状态（ADCSSFSTAT1），偏移量 0x06C

该寄存器为采样序列发生器 FIFO 1 提供了一个窗口，包括满/空状态信息以及头指针和尾指针的位置。复位值 0x100 表示 FIFO 为空。

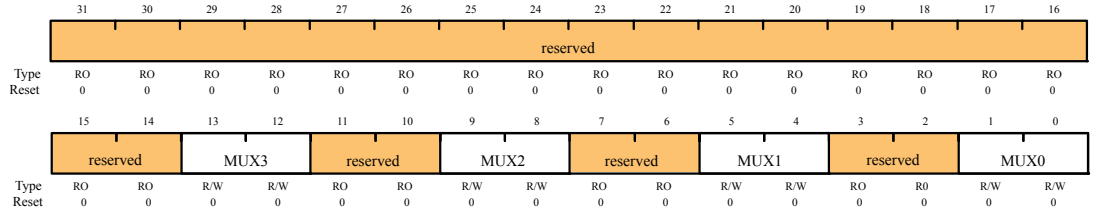
ADCSSFSTAT1 的各个位以及定义均与 **ADCSSFSTAT0** 相同，只不过 **ADCSSFSTAT1** 是用于 FIFO 1。

寄存器 19：ADC 采样序列输入多路复用器选择 2（ADCSSMUX2），偏移量 0x080

该寄存器为序列发生器 2 所执行序列的每个采样定义模拟输入配置。

该寄存器为 16 位宽，并且包含 4 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSMUX2** 各个位的定义均与 **ADCSSMUX0** 寄存器相同，只不过 **ADCSSMUX2** 是用于采样序列发生器 2。

ADC取样序列输入多路复用器选择2（ADCSSMUX2）寄存器
偏移量 0x080



寄存器 20：ADC 采样序列控制 2（ADCSSCTL2），偏移量 0x084

该寄存器包含采样序列发生器 2 所执行序列的每个采样的配置信息。在对采样序列进行配置时，END 位必须在某些断点（point）处置位，而不管它是在第一个采样之后，还是在最后一个采样后，或者是在任意中间的采样后。

该寄存器为 16 位宽，并且包含 4 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSCTL2** 各个位的定义均与 **ADCSSCTL0** 寄存器相同，只不过 **ADCSSCTL2** 是用于采样序列发生器 2。

ADC取样序列控制2（ADCSSCTL2）寄存器
偏移量 0x084

寄存器 21：ADC 采样序列结果 FIFO 2（ADCSSFIFO2），偏移量 0x088

该寄存器包含的是由采样序列发生器 2 收集的采样的转换结果。读取该寄存器时，将按照采样 0、采样 1，…的顺序返回转换结果数据，直至 FIFO 为空。如果软件没有对 FIFO 进行正确的处理，那么上溢和下溢条件均被记录在 **ADCOSTAT** 和 **ADCUSTAT** 寄存器中。

ADCSSFIFO2 的各个位以及定义均与 **ADCSSFIFO0** 相同，只不过 **ADCSSFIFO2** 是用于 FIFO 2。

寄存器 22：ADC 采样序列 FIFO 2 状态（ADCSSFSTAT2），偏移量 0x08C

该寄存器为采样序列发生器 FIFO 2 提供了一个窗口，包括满/空状态信息以及头指针和尾指针的位置。复位值 0x100 表示 FIFO 为空。

ADCSSFSTAT2 的各个位以及定义均与 **ADCSSFSTAT0** 相同，只不过 **ADCSSFSTAT2** 是用于 FIFO 2。

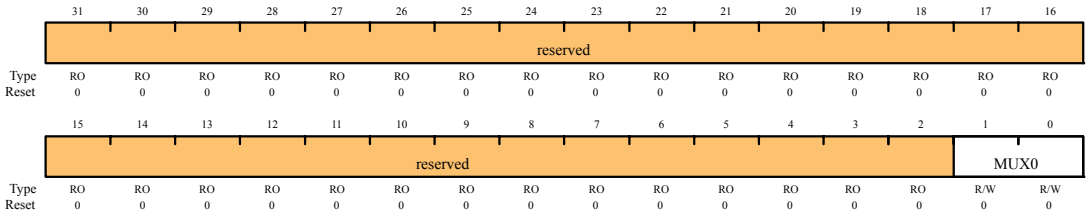
寄存器 23：ADC 采样序列输入多路复用器选择 3（ADCSSMUX3），偏移量 0x0A0

该寄存器为序列发生器 3 所执行序列的每个采样定义模拟输入配置。

该寄存器为 4 位宽，并且包含 1 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSMUX3** 各个位的定义均与 **ADCSSMUX0** 寄存器相同，只不过 **ADCSSMUX3** 是用

于采样序列发生器 3。

ADC取样序列输入多路复用器选择3（ADCSSMUX3）寄存器
偏移量 0x0A0

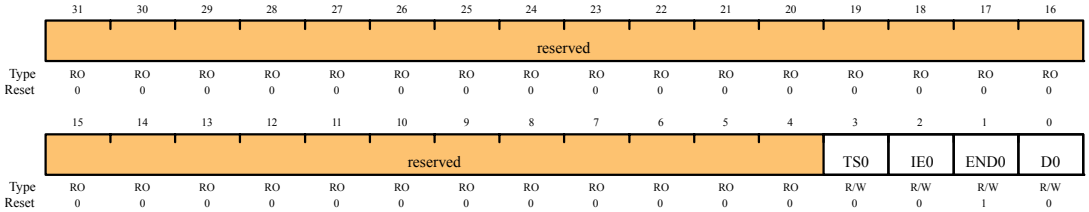


寄存器 24：ADC 采样序列控制 3（ADCSSCTL3），偏移量 0x0A4

该寄存器包含序列发生器 3 所执行序列的每个采样的配置信息。在对采样序列进行配置时，END 位必须在某些断点（point）处置位，而不管它是在第一个采样之后，还是在最后一个采样后，或者是在任意中间的采样后。

该寄存器为 4 位宽，并且包含 1 个可能采样的信息。该寄存器的各个位如下图所示。**ADCSSCTL3** 各个位的定义均与 **ADCSSCTL0** 寄存器相同，只不过 **ADCSSCTL3** 是用于采样序列发生器 3。

ADC取样序列控制3（ADCSSCTL3）寄存器
偏移量 0x064



寄存器 25：ADC 采样序列结果 FIFO 3（ADCSSFIFO3），偏移量 0x0A8

该寄存器包含的是由采样序列发生器 3 收集的采样的转换结果。读取该寄存器时，将按照采样 0、采样 1，…的顺序返回转换结果数据，直至 FIFO 为空。如果软件没有对 FIFO 进行正确的处理，那么上溢和下溢条件均被记录在 **ADCOSTAT** 和 **ADCUSTAT** 寄存器中。

ADCSSFIFO3 的各个位以及定义均与 **ADCSSFIFO0** 相同，只不过 **ADCSSFIFO3** 是用于 FIFO 3。

寄存器 26：ADC 采样序列 FIFO 3 状态（ADCSSFSTAT3），偏移量 0x08C

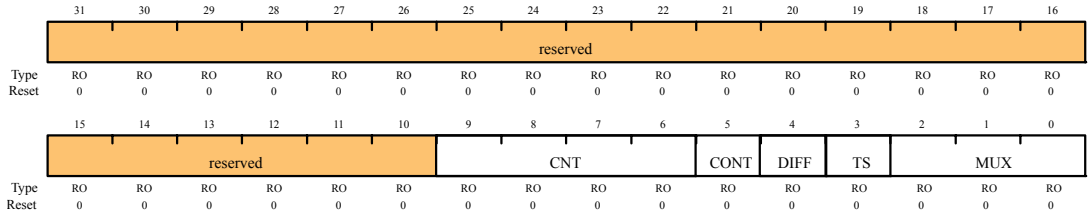
该寄存器为采样序列发生器 FIFO 3 提供了一个窗口，包括满/空状态信息以及头指针和尾指针的位置。复位值 0x100 指示 FIFO 为空。

ADCSSFSTAT3 的各个位以及定义均与 **ADCSSFSTAT0** 相同，只不过 **ADCSSFSTAT3** 是用于 FIFO 3。

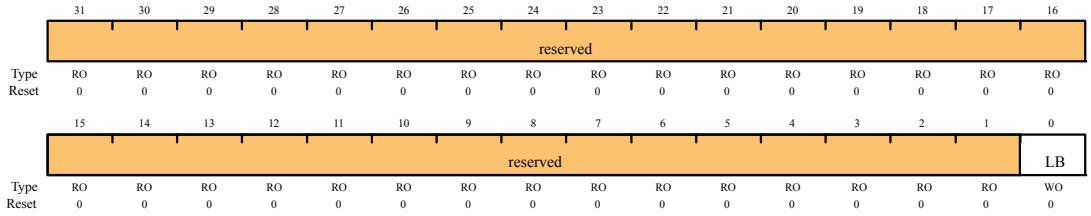
寄存器 27：ADC 测试模式回送（ADCTMLB），偏移量 0x100

该寄存器在 ADC 的数字逻辑内提供回送（loopback）操作，这在调试软件中非常有用，因为无需提供真实的模拟激励信号源（analog stimulus）。该测试模式通过写 0x00000001 到该寄存器来进入。当在回送模式下从 FIFO 读取数据时，返回的是该寄存器的只读部分。

ADC测试模式回送（ADCTMLB）寄存器：读
偏移量 0x100



ADC测试模式回送（ADCTMLB）寄存器：写
偏移量 0x100



位	名称	类型	复位	描述
只读寄存器				
31:10	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。
9:6	CNT	RO	0	连续采样计数器，初始化为 0，并且对每个采样进行计数。这有助于为接收数据提供一个唯一值。
5	CONT	RO	0	置位时，表示这将是一个连续采样。例如，如果两个序列发生器即将背对背（back-to-back）运行，那么就表示控制器将以全速保持连续采样。
4	DIFF	RO	0	置位时，表示这将是一个差分采样
3	TS	RO	0	置位时，表示这将是一个温度传感器采样
2:0	MUX	RO	0	表明哪个模拟输入被采样
只写寄存器				
31:1	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
0	LB	WO	0	置位时，在数字块中强制执行回送，以提供有关输入的信息和唯一的编号。 10 位回送数据的定义见“只读寄存器的位 9:0”。

第12章 通用异步收发器（UART）

通用异步收发器（UART）提供了完全可编程的能力，同时具备 16C550 类型串行接口的特性。LM3S617 控制器配有 2 个 UART 模块。

每个 UART 含以下特性：

- 独立的发送 FIFO 和接收 FIFO
- FIFO 长度可编程，包括提供传统双缓冲接口的 1 字节深的操作
- FIFO 触发深度（trigger level）为：1/8、1/4、1/2、3/4 和 7/8
- 可编程的波特率发生器，允许速率高达 460.8Kbps
- 标准的异步通信位：起始位、停止位和奇偶校验位（parity）
- 检测错误的起始位
- 线中止（Line-break）的产生和检测
- 完全可编程的串行接口特性：
  - 5、6、7 或 8 个数据位
  - 偶校验、奇校验、粘着或无奇偶校验位的产生/检测
  - 产生 1 或 2 个停止位

12.1 方框图

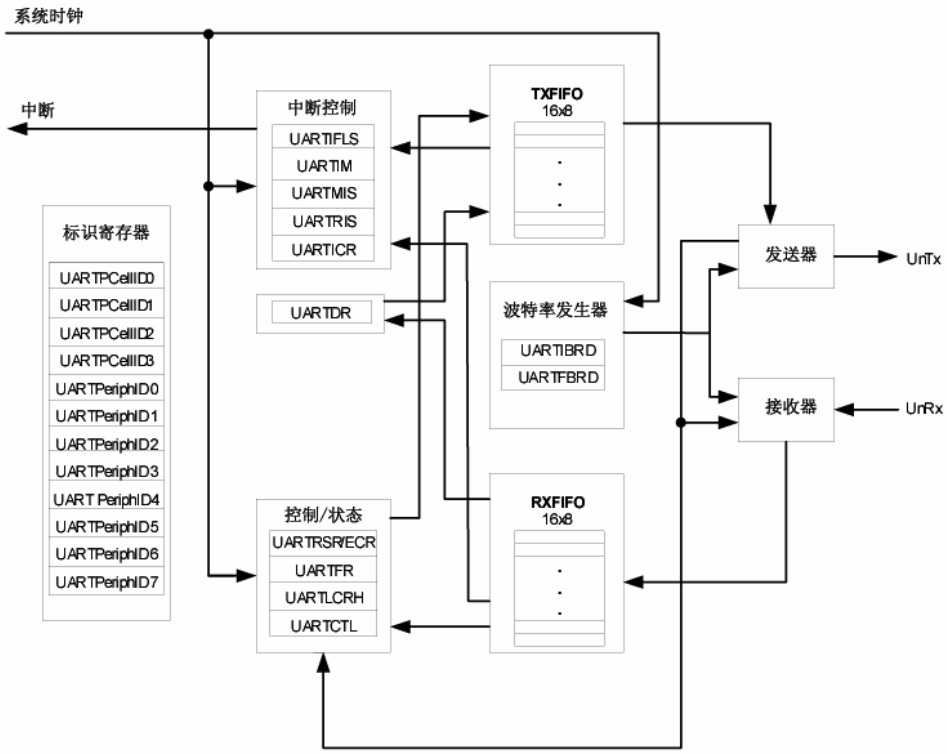


图 12.1 UART 模块的方框图

12.2 功能描述

群星（Stellaris）UART 可执行“并－串”和“串－并”转换功能。其功能与 16C550 UART 类似，但两者的寄存器不兼容。

用户可通过 **UART 控制（UARTCTL）**寄存器的 TXE 和 RXE 位将 UART 配置成发送和/或接收。复位完成后，发送和接收都是使能的。在对任一控制寄存器编程之前，必须将 UART 禁能，这可以通过将 **UARTCTL** 寄存器的 UARTEN 位清零来实现。如果 UART 在 TX 或 RX 操作过程中被禁能，则当前的处理会在 UART 停止前完成。

12.2.1 发送/接收逻辑

发送逻辑对从发送FIFO读取的数据执行“并－串”转换。控制逻辑会以起始位为开始输出串行位流，然后根据控制寄存器中已编程的配置，紧接着输出数据位（最低位先输出）、奇偶校验位和停止位。详见 图 12.2。

在检测到一个有效的起始位脉冲后，接收逻辑会对接收到的位流执行“串－并”转换。此外还会对溢出错误、奇偶校验错误、帧错误和线中止（line-break）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。

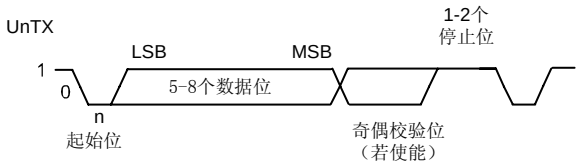


图 12.2 UART 字符帧

12.2.2 波特率的产生

波特率除数（divisor）是一个 22 位数，它由 16 位整数和 6 位小数组成。波特率发生器使用这两个值组成的数字来决定位周期。通过小数波特率除法器，UART 可以产生所有标准的波特率。

16 位整数通过 **UART 整数波特率除数（UARTIBRD）**寄存器进行加载，而 6 位小数则通过 **UART 小数波特率除数（UARTFBRD）**寄存器进行加载。波特率除数（BRD）和系统时钟具有以下关系（*BRDI* 是 BRD 的整数部分，*BRDF* 是 BRD 的小数部分，它被一个小数位隔开）：

BRD = BRDI + BRDF = SysClk / (16 * 波特率)

以下等式是 6 位小数（被加载到 **UARTFBRD** 寄存器的 DIVFRAC 字段）的计算方法。即，将波特率除数的小数部分乘以 64 再加 0.5 以便将误差四舍五入。

UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)

UART 产生一个 16 倍于波特率（称作 Baud16）的内部波特率参考时钟。该参考时钟经过 16 分频后便可产生发送时钟，它还可以在接收操作过程中用作错误检测。

高字节（UARTLCRH）寄存器、**UARTIBRD 和 UARTFBRD 寄存器**连同 **UART 线控制**一起共同形成一个内部 30 位寄存器。该内部寄存器仅在对 **UARTLCRH** 进行写操作时才会更新，所以为了使修改波特率除数生效，后面必须紧跟一个写 **UARTLCRH** 寄存器

操作。

有四种序列可以用来更新波特率寄存器：

- 写入 **UARTIBRD**、然后写入 **UARTFBRD**，最后写入 **UARTLCRH**
- 写入 **UARTFBRD**、然后写入 **UARTIBRD**，最后写入 **UARTLCRH**
- 写入 **UARTIBRD**、然后写入 **UARTLCRH**
- 写入 **UARTFBRD**、然后写入 **UARTLCRH**

12.2.3 数据发送

尽管接收 FIFO 每个字符还包含 4 个额外的状态信息位，但是接收或发送的数据都存放在 2 个 16 位 FIFO 中。发送时，数据被写入发送 FIFO。如果 UART 被使能，它会让数据帧按照 **UARTLCRH** 寄存器中设置的参数开始发送。然后就一直发送数据，直至发送 FIFO 中没有数据剩下为止。当数据被写入 FIFO 后（即，如果 FIFO 未空），**UART 标志（UARTFR）**寄存器中的 BUSY 位就会生效，并且在发送数据期间一直保持有效。BUSY 位仅在发送 FIFO 为空，且最后一个字符、包括停止位在内也已经从移位寄存器中发出时，才会变无效。即使 UART 不再使能，它也可以指示出 UART 是否处于忙(busy)状态。

在接收器空闲（U0Rx或U1Rx连续为 1）且数据输入变为“低电平”（收到起始位）时，接收计数器开始运行，并且数据在Baud16 的第八个周期被采样（详见“发送/接收逻辑”）。

如果 U0Rx 或 U1Rx 在 Baud16 的第八个周期仍然为低电平，那么起始位有效，否则检测到的起始位是错误的并将该起始位忽略。可以在 **UART 接收状态（UARTRSR）**寄存器中观察起始位的错误。如果起始位有效，根据设置的数据字符长度，每逢 Baud16 的的第 16 个周期（即一个位周期之后）就会对连续的数据位进行采样。如果奇偶校验模式使能，那么接着检测奇偶校验位。数据长度和奇偶校验都在 **UARTLCRH** 寄存器中定义。

最后，如果 U0Rx 或 U1Rx 为高电平，那么有效的停止位被确认，否则发生帧错误。当接收到一个完整的字时，数据会被存放到接收 FIFO 中，而与该字相关的错误位也包括在内。

12.2.4 FIFO 操作

UART 含 2 个 16 位入口的 FIFO；一个用于发送，另一个用于接收。两个 FIFO 都通过 **UART 数据（UARTDR）**寄存器进行访问。UARTDR 的写操作会把 8 位的数据放入发送 FIFO，而 **UARTDR** 寄存器的读操作返回的却是一个 12 位的值，该值由 8 个数据位和 4 个错误标志组成。

复位完成后，两个 FIFO 都被禁能，并充当 1 字节深的保存（holding）寄存器。通过置位 **UARTLCRH** 的 FEN 位可以使 FIFO 使能。

用户可以通过 **UART 标志（UARTFR）**寄存器和 **UART 接收状态（UARTRSR）**寄存器来监控 FIFO 状态。而对空、满和溢出条件的监控则是由硬件来完成的。**UARTFR** 寄存器包含了空和满的标志（TXFE、TXFF、RXFE 和 RXFF 位），而 **UARTRSR** 寄存器则通过 OE 位指示出溢出的状态。

促使 FIFO 产生中断的触发点是通过 **UART 中断的 FIFO 深度选择（UARTIFLS）**寄存器来控制的。可将两个 FIFO 的中断分别配置为不同的触发深度。可供选择的配置包括 1/8、1/4、1/2、3/4 和 7/8。例如，如果接收 FIFO 选择 1/4，那么 UART 在接收了 4 个数据字节之后产生接收中断。在复位完成后，两个 FIFO 都被配置为以 1/2 的深度来触发中断。

12.2.5 中断

在观察到以下情况时，UART 会产生中断：

- 溢出(overrun)错误
- 中止错误（Break Error）
- 奇偶校验（partiy error）错误
- 帧错误
- 接收超时（timeout）
- 发送（在满足 **UARTIFLS** 寄存器中 TXIFLSEL 位所定义的条件时）
- 接收（在满足 **UARTIFLS** 寄存器中 RXIFLSEL 位所定义的条件时）

由于所有中断事件在发送到中断控制器前会一起进行或（OR）操作，所以任意时刻 UART 只能向中断产生一个中断请求。通过读取 **UART 屏蔽后的中断状态（UARTMIS）**寄存器，软件可以在一个中断服务程序中处理多个中断事件。

通过将 **UART 中断屏蔽(UARTIM)**寄存器中对应的 IM 位设置为 1，可以定义能够触发控制器级别中断的中断事件。假如不使用中断，原始的中断状态也是始终可见的，通过 **UART 原始中断状态(UARTRIS)**寄存器便可查询到该状态。

只需把 **UART 中断清除(UARTICR)**寄存器中相应的位置位，便可以清除中断（**UARTMIS** 和 **UARTRIS** 寄存器的中断）。

12.2.6 回送（loopback）操作

UART 可以进入一个内部回送模式，用于诊断或调试。这是通过置位 **UARTCTL** 寄存器中的 LBE 位来实现的。在回送模式下，U0Tx 上发送的数据将被 U0Rx 输入端接收，U1Tx 上发送的数据将被 U1Rx 输入端接收。

12.3 初始化和配置

要使用 UART，必须把 **RCGC1** 寄存器中的 UART0 或 UART1 位置位，来使能外设时钟。

本小节讨论了使用 UART 模块所需的步骤。例如，假定系统时钟为 20MHz，且所需的 UART 配置为：

- 115200 波特率
- 8 位数据长度
- 一个停止位
- 无奇偶校验
- FIFO 禁能
- 无中断

因为对**UARTIBRD**和**UARTFBRD**寄存器的写操作必须先于**UARTLCRH**寄存器，所以在对UART进行编程时，首先要考虑的是波特率除数（BRD）。而BRD可以通过“波特率的产生”中描述的等式计算得到：

BRD = 20,000,000 / (16 * 115,200) = 10.8507

即 **UARTIBRD** 寄存器的 DIVINT 字段应该设为 10。载入到 **UARTFBRD** 寄存器的值是通过以下等式算出来的：

UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54

如此便得到了 BRD 的值，接着要按照以下顺序将 UART 配置写入模块：

1. 将 **UARTCTL** 寄存器中的 UARTEN 位清零，以便将 UART 禁能
2. 将 BRD 的整数部分写入 **UARTIBRD** 寄存器
3. 将 BRD 的小数部分写入 **UARTFBRD** 寄存器
4. 将所需的串行参数写入 **UARTLCRH** 寄存器（这种情况下为 0x00000060）。
5. 将 **UARTCTL** 寄存器中的 UARTEN 位置位，以便将 UART 使能。

12.4 寄存器映射

表 12.1 列出了UART寄存器。所列的偏移量是 16 进制的，并按照寄存器地址递增，与UART对应的基址如下：

- UART0：0x4000C000
- UART1：0x4000D000

注：在重新对任意控制寄存器进行编程以前，必须将 UART 禁能（见 **UARTCTL** 寄存器中对 UARTEN 位的说明）。如果在 TX 或 RX 操作过程中 UART 被禁能，那么当前的处理将在 UART 停止前完成。

表 12.1 UART 寄存器映射

偏移量	名称	复位	类型	描述
0x000	UARTDR	0x00000000	R/W	数据
0x004	UARTRSR UARTECR	0x00000000	R/W	接收状态（读）；错误清除（写）
0x018	UARTFR	0x00000090	RO	标志寄存器（只读）
0x024	UARTIBRD	0x00000000	R/W	波特率除数的整数部分
0x028	UARTFBRD	0x00000000	R/W	波特率除数的小数部分
0x02C	UARTLCRH	0x00000000	R/W	线控制寄存器，高字节
0x030	UARTCTL	0x00000300	R/W	控制寄存器
0x034	UARTIFLS	0x00000012	R/W	中断的 FIFO 深度（level）选择
0x038	UARTIM	0x00000000	R/W	中断屏蔽
0x03C	UARTRIS	0x0000000F	RO	原始（raw）中断状态
0x040	UARTMIS	0x00000000	RO	屏蔽后的中断状态
0x044	UARTICR	0x00000000	W1C	中断清除
0xFD0	UARTPeriphID4	0x00000000	RO	外设标识 4
0xFD4	UARTPeriphID5	0x00000000	RO	外设标识 5
0xFD8	UARTPeriphID6	0x00000000	RO	外设标识 6

续上表

偏移量	名称	复位	类型	描述
0xFDC	UARTPeriphID7	0x00000000	RO	外设标识 7
0xFE0	UARTPeriphID0	0x00000011	RO	外设标识 0
0xFE4	UARTPeriphID1	0x00000000	RO	外设标识 1
0xFE8	UARTPeriphID2	0x00000018	RO	外设标识 2
0xFEC	UARTPeriphID3	0x00000001	RO	外设标识 3
0xFF0	UARTPCellID0	0x0000000D	RO	PrimeCell 标识 0
0xFF4	UARTPCellID1	0x000000F0	RO	PrimeCell 标识 1
0xFF8	UARTPCellID2	0x00000005	RO	PrimeCell 标识 2
0xFFC	UARTPCellID3	0x000000B1	RO	PrimeCell 标识 3

12.5 寄存器描述

本小节按照地址偏移量的数字顺序列出并描述了 UART 寄存器。

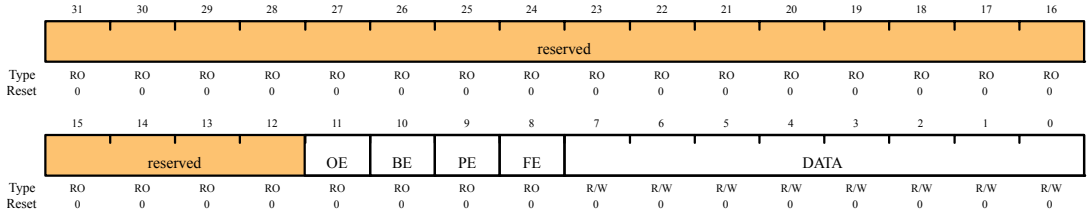
寄存器 1：UART 数据（UARTDR）寄存器，偏移量：0x000

该寄存器是数据寄存器（FIFO 的接口）。

当 FIFO 使能时，写入该单元中的数据被移入发送 FIFO。如果 FIFO 被禁能，数据将存放在发送器保存寄存器（发送 FIFO 底部的字）中。对该寄存器进行写操作会开启一个 UART 发送操作。

对于接收到的数据来说，如果 FIFO 被使能，数据字节和 4 个状态位（间隔、帧、奇偶校验和溢出）被移入 12 位宽的接收 FIFO。如果 FIFO 被禁能，那么数据字节和状态位将存放在接收保存寄存器（接收 FIFO 底部的字）中。读取该寄存器可以重新得到接收的数据。

UART数据（UARTDR）寄存器
偏移量：0x000



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
11	OE	RO	0	UART 溢出错误 1＝ 当 FIFO 满时接收到新的数据，导致数据丢失 0＝ 没有出现因为 FIFO 溢出而导致数据丢失的情况

续上表

位/字段	名称	类型	复位	描述
10	BE	RO	0	UART 中止错误（break error） 在检测到中止（break）条件时该位被设为 1，表示接收数据输入端在长于一个完整字的传输时间（定义为起始位、数据位、奇偶校验位和停止位）内一直保持低电平。 在 FIFO 模式下，该错误与 FIFO 顶部的字符有关。在发生中止（break）时，只有一个 0 字符被加载到 FIFO。下一字符仅在接收数据输入端变为 1（标志（marking）状态）且接收到下一有效的起始位时才使能。
9	PE	RO	0	UART 奇偶校验错误 当接收到的数据字符与 **UARTLCRH** 寄存器中位 2 和位 7 所定义的奇偶不匹配时，该位被设为 1。 在 FIFO 模式下，该错误与 FIFO 顶部的字符有关。
8	FE	RO	0	UART 帧错误 在接收的字符未含有效的停止位（有效的停止位为 1）时，该位被设为 1。
7:0	DATA	R/W	0	被写时，数据由 UART 发送。读取时，数据由 UART 接收

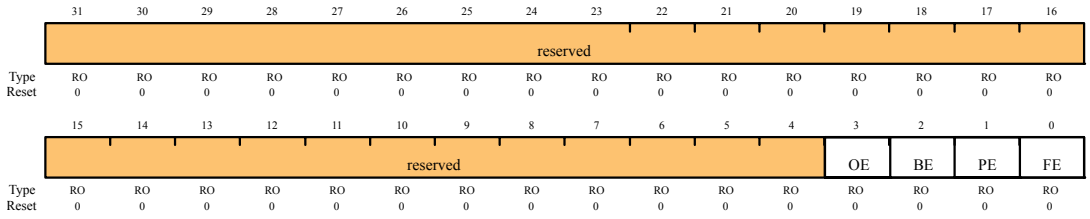
寄存器 2：UART 接收状态/错误清除（UARTRSR/UARTECR）寄存器，偏移量：0x004

UARTRSR 和 UARTECR 寄存器分别为接收状态寄存器和错误清除寄存器。

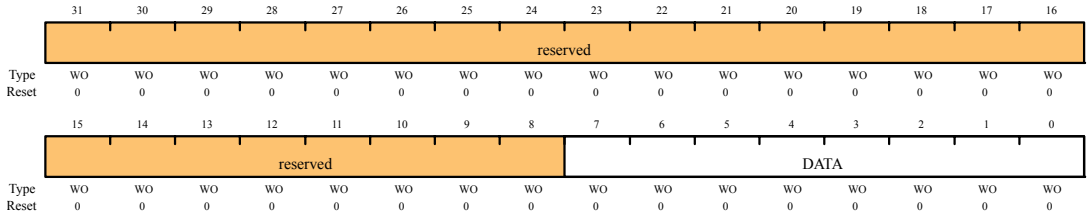
接收状态除了可以从 **UARTDR** 寄存器中读取之外，还可以从 **UARTRSR** 寄存器中读取。如果从 **UARTRSR** 读取状态，与入口相对应的状态信息将在读取 **UARTRSR** 前先从 **UARTDR** 读取。当溢出的情况发生时，溢出的状态信息将被立即置位。

将任意值写入 **UARTECR** 寄存器都会将帧、奇偶校验、中止和溢出错误清除。所有位在复位后都会被清零。

UART接收状态（UARTRSR）寄存器：读
偏移量：0x004



UART错误清除（UARTECR）寄存器：写
偏移量：0x004

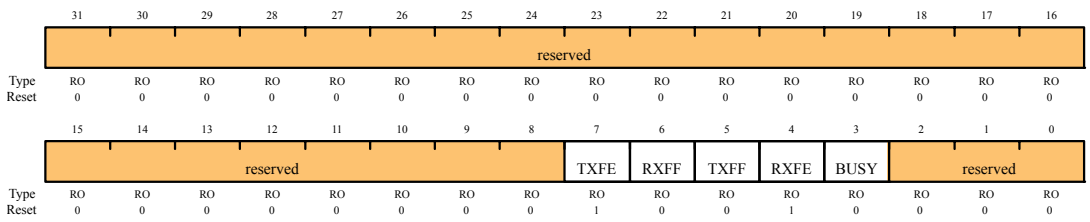


位/字段	名称	类型	复位	描述
只读接收状态（UARTRSR）寄存器				
31:4	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。**UARTRSR** 寄存器不能被写
3	OE	RO	0	UART 溢出错误 当 FIFO 满且接收到新的数据时该位被设为 1。对 **UARTECR** 进行写操作会将该位清零。 由于在 FIFO 满时不再有数据写入，所以 FIFO 内容将保持有效，只有移位寄存器的内容会被覆盖。此时，CPU 必须读取数据以便将 FIFO 清空。
2	BE	RO	0	UART 中止错误（break error） 在检测到中止的情况时该位会被设为 1，表示接收数据输入在长于一个完整字的传输时间（定义为起始位、数据位、奇偶校验位和停止位）内一直保持低电平。 对 **UARTECR** 进行写操作会将该位清零。 在 FIFO 模式下，该错误与 FIFO 顶部的字符有关。在发生中止时，只有一个 0 字符被加载到 FIFO。下一字符仅在接收数据输入变为 1（标志（marking）状态）且接收到下一个有效的起始位时才使能。
1	PE	RO	0	UART 奇偶校验错误 当接收到的数据字符的奇偶校验值与 **UARTLCRH** 寄存器中位 2 和位 7 所定义的奇偶校验值不符时，该位被设为 1。 对 **UARTECR** 进行写操作会将该位清零。
0	FE	RO	0	UART 帧错误 当接收到的字符不含有效的停止位（有效的停止位为 1）时，该位被设为 1。 对 **UARTECR** 进行写操作会将该位清零。 在 FIFO 模式下，该错误与 FIFO 顶部的字符有关。
只写错误清除（UARTECR）寄存器				
31:8	保留	WO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	DATA	WO	0	将任意数据写入该寄存器会将帧、奇偶校验、中止和溢出标志清零

寄存器 3：UART 标志（UARTFR）寄存器，偏移量：0x018

UARTFR 寄存器是标志寄存器。复位后，TXFF、RXFF 和 BUSY 位都为 0，而 TXFE 和 RXFE 位为 1。

UART标志（UARTFR）寄存器
偏移量：0x018

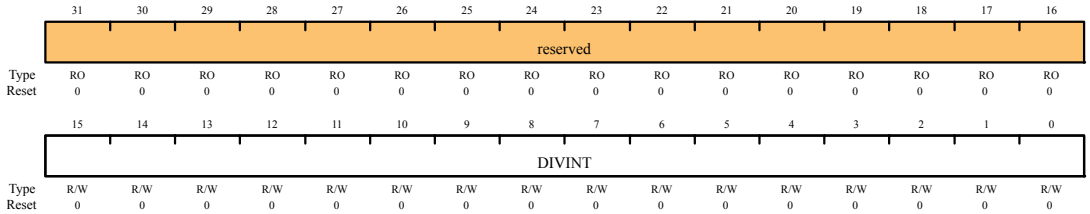


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7	TXFE	RO	1	UART 发送 FIFO 空 该位的具体意思取决于 **UARTLCRH** 寄存器中 FEN 位的状态。 如果 FIFO 被禁能（FEN 为 0），那么该位在发送保存寄存器为空时置位 如果 FIFO 使能（FEN 为 1），那么该位在发送 FIFO 为空时置位
6	RXFF	RO	0	UART 接收 FIFO 满 该位的具体意思取决于 **UARTLCRH** 寄存器中 FEN 位的状态。 如果 FIFO 被禁能，那么该位在接收保存寄存器满时置位。 如果 FIFO 使能，那么该位在接收 FIFO 满时置位。
5	TXFF	RO	0	UART 发送 FIFO 满 该位的具体意思取决于 **UARTLCRH** 寄存器中 FEN 位的状态。 如果 FIFO 被禁能，那么该位在发送保存寄存器满时置位。 如果 FIFO 使能，那么该位在发送 FIFO 满时置位。
4	RXFE	RO	1	UART 接收 FIFO 空 该位的具体意思取决于 **UARTLCRH** 寄存器中 FEN 位的状态。 如果 FIFO 被禁能，那么该位在接收保存寄存器为空时置位。 如果 FIFO 使能，那么该位在接收 FIFO 为空时置位。
3	BUSY	RO	0	UART 忙 该位为 1 时，UART 忙于发送数据。该位保持置位，直至移位寄存器将包括所有停止位在内的全部字节发送。 一旦发送 FIFO 不为空时（不管 UART 是否使能）该位都会置位。
2:0	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变

寄存器 4：UART 整数波特率除数（UARTIBRD）寄存器，偏移量：0x024

UARTIBRD寄存器是波特率除数的整数部分。它的所有位在复位后都会被清零。可实现的最小比率为 1（当**UARTIBRD**＝0 时），在此情况中， **UARTFBRD**寄存器将被忽略。在修改**UARTIBRD**寄存器时，直到发送/接收当前字符结束之后，新的值才会生效。对波特率除数的任意修改，其后都要紧跟一个写入**UARTLCRH**寄存器的操作。要了解配置的详情，请参考“波特率的产生”。

UART整数波特率除数（UARTIBRD）寄存器
偏移量：0x024

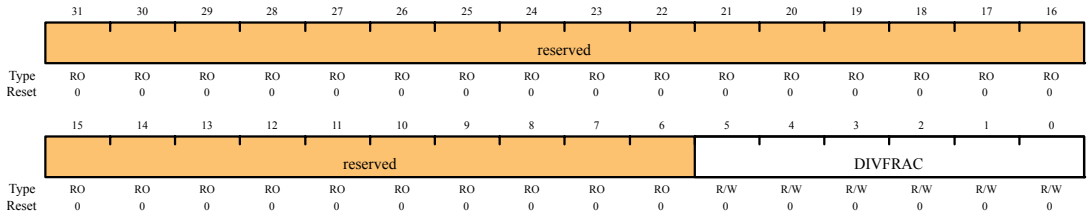


位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
15:0	DIVINT	R/W	0x0000	整数波特率除数

寄存器 5：UART 小数波特率除数（UARTIBRD）寄存器，偏移量：0x028

UARTFBRD寄存器是波特率除数的小数部分。它的所有位在复位后都会被清零。在修改**UARTFBRD**寄存器时，直到发送/接收当前字符结束后，新的值才会生效。对波特率除数的任意修改，其后都要紧跟一个写入**UARTLCRH**寄存器的操作。要了解配置的详情，请参考“波特率的产生”。

UART小数波特率除数（UARTFBRD）寄存器
偏移量：0x028



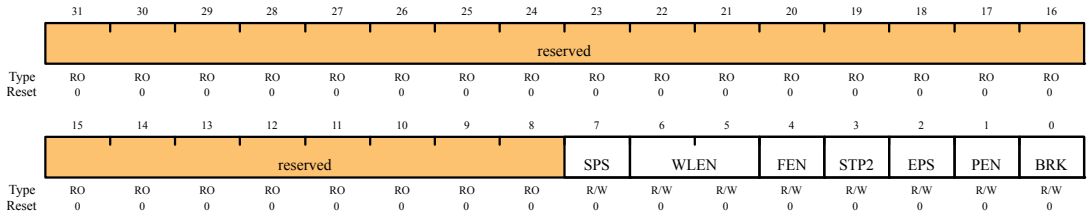
位/字段	名称	类型	复位	描述
31: 6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5:0	DIVFRAC	R/W	0x00	小数波特率除数

寄存器 6：UART 线控制（UARTLCRH）寄存器，偏移量：0x02C

UARTLCRH 寄存器是线控制寄存器。串行参数，例如数据长度、奇偶校验位和停止位的选择都是在该寄存器中完成的。

在更新波特率除数（**UARTIBRD** 和/或 **UARTIFRD**）时，还必须写 **UARTLCRH** 寄存器。波特率除数寄存器的写入选通（strobe）信号与 **UARTLCRH** 寄存器相连。

UART线控制（UARTLCRH）寄存器
偏移量：0x02C



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7	SPS	R/W	0	UART 保留（stick）奇偶校验选择 在 **UARTLCRH** 的位 1、位 2 和位 7 置位时，发送奇偶校验位，且检测结果为 0。在位 1 和位 7 都置位且位 2 清零时，发送奇偶校验位，且检测结果为 1。 该位清零时，保留奇偶校验被禁能

续上表

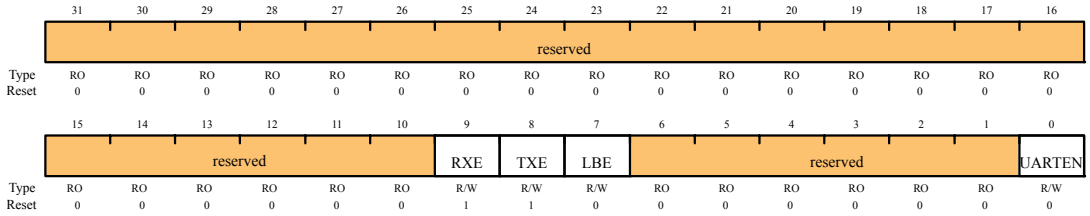
位/字段	名称	类型	复位	描述
6:5	WLEN	R/W	0	UART 字长 该位表示在发送或接收时一帧中所含的数据位数，如下： 0x3：8 位 0x2：7 位 0x1：6 位 0x0：5 位（默认）
4	FEN	R/W	0	UART 使能 FIFO 如果该位设为 1，那么发送和接收 FIFO 缓冲器都会被使能（FIFO 模式） 若被清零，那么所有 FIFO 都会被禁能（字符模式）。且 FIFO 都会变成 1 字节深的保存寄存器。
3	STP2	R/W	0	UART 双停止位选择 如果该位设为 1，在帧的末尾将发送两个停止位。接收逻辑不会检测正在接收的 2 个停止位。
2	EPS	R/W	0	UART 偶校验（even parity）选择 如果该位设为 1，那么偶校验的产生和检测都在发送和接收过程中进行，检测数据位加奇偶校验位“1”的位数是否为偶数 清零时，执行奇校验，检查“1”的位数是否为奇数。
1	PEN	R/W	0	UART 奇偶校验使能 如果该位设为 1，那么奇偶校验及其产生都使能；否则，奇偶校验被禁能，且数据帧中不会增加奇偶校验位。
0	BRK	R/W	0	UART 发送中止（break） 如果该位设为 1，在完成当前字符的发送后，UnTX 输出端上会连续的输出低电平。在正确执行中止（break）命令时，软件必须将该位置位，并且持续至少 2 个帧（字符周期）。在正常使用时，该位必须清零。

寄存器 7：UART 控制（UARTCTL）寄存器，偏移量：0x030

UARTCTL 寄存器是控制寄存器。所有位都在复位后清零，“发送使能（TXE）”和“接收使能（RXE）”位除外，它们都被设为 1。

为了使能 UART 模块，UARTEN 位必须置位。如果软件要求修改模块的配置，那么在对配置的改动被写入前 UARTEN 位必须清零。如果 UART 在发送或接收操作过程中被禁能，那么当前的处理将在 UART 停止前完成。

UART控制（UARTCTL）寄存器
偏移量：0x030



位/字段	名称	类型	复位	描述
31:10	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
9	RXE	R/W	1	UART 接收使能 如果该位置位，那么 UART 的接收部分被使能。如果 UART 在接收中途被禁能，它会在停止前处理完当前字符。
8	TXE	R/W	1	UART 发送使能 如果该位置位，那么 UART 的发送部分被使能。如果 UART 在发送中途被禁能，它会在停止前处理完当前字符。
7	LBE	R/W	0	UART 回送使能 如果该位置位，那么 UnRX 输入端将连接到 UnTX。
6:1	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
0	UARTEN	R/W	0	UART 使能 如果该位置位，那么 UART 被使能。如果 UART 在发送或接收中途被禁能，它会在停止前处理完当前字符。

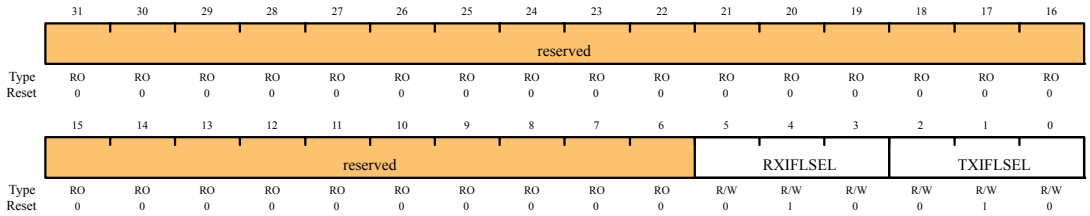
寄存器 8：UART 中断的 FIFO 深度（level）选择（UARTIFLS）寄存器，偏移量：0x034

UARTIFLS 寄存器是中断的 FIFO 深度（level）选择寄存器。你可以使用该寄存器来定义 **UARTRIS** 寄存器中 TXRIS 和 RXRIS 位触发（中断）时的 FIFO 深度。

中断触发的依据是，当 FIFO 的载入的数据量(深度)超过某一水平时触发，而不是当载入的数据量(深度)达到某一水平的时候触发。也就是说，FIFO 所装的数据量(深度)超过规定触发的水平时，就产生中断。例如，如果接收触发的水平被设为 1/2，那么中断将在模块接收第 9 个字符时触发。

复位完成后，TXIFLSEL 和 RXIFLSEL 位都会被配置，因此 FIFO 将以 1/2 作为触发深度触发中断。

UART中断的FIFO深度选择（UARTIFLS）寄存器
偏移量：0x034



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
5:3	RXIFLSEL	R/W	0x2	UART 接收中断 FIFO 深度选择 接收中断的触发点如下： 000：RX FIFO 的数据量≥全满时的 1/8 001：RX FIFO 的数据量≥全满时的 1/4 010：RX FIFO 的数据量≥全满时的 1/2（默认） 011：RX FIFO 的数据量≥全满时的 3/4 100：RX FIFO 的数据量≥全满时的 7/8 101-111：保留

续上表

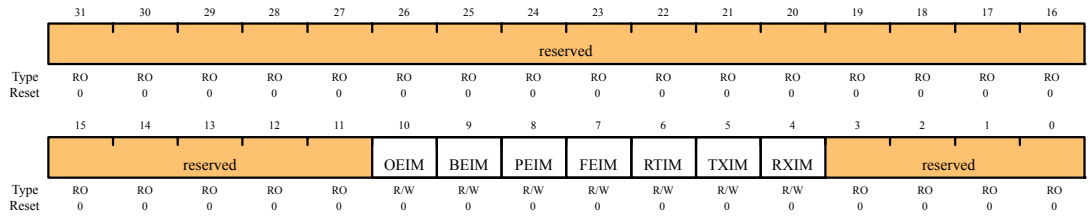
位/字段	名称	类型	复位	描述
2:0	TXIFLSEL	R/W	0x2	UART 发送中断 FIFO 填充度选择 发送中断的触发点如下： 000：RX FIFO 的数据量≤全满时的 1/8 001：RX FIFO 的数据量≤全满时的 1/4 010：RX FIFO 的数据量≤全满时的 1/2（默认） 011：RX FIFO 的数据量≤全满时的 3/4 100：RX FIFO 的数据量≤全满时的 7/8 101-111：保留

寄存器 9：UART 中断屏蔽寄存器（UARTIM）寄存器，偏移量：0x038

UARTIM 寄存器是中断屏蔽设置/清除寄存器。

读取时，该寄存器会给出当前对相关中断的屏蔽值。向某个位写 1 时，将允许与该位相对应的原始中断信号发送到中断控制器。向某个位写 0 时，则禁止将与该位相对应的原始中断信号发送到中断控制器。

UART中断屏蔽（UARTIM）寄存器
偏移量：0x038



位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
10	OEIM	R/W	0	UART 溢出错误中断屏蔽 读取时，返回 OEIM 中断的当前屏蔽值。 该位置位时，可以将 OEIM 中断发送到中断控制器
9	BEIM	R/W	0	UART 中止（break）错误中断屏蔽 读取时，返回 BEIM 中断的当前屏蔽值 该位置位时，可以将 BEIM 中断发送到中断控制器
8	PEIM	R/W	0	UART 奇偶校验错误中断屏蔽 读取时，返回 PEIM 中断的当前屏蔽值 该位置位时，可以将 PEIM 中断发送到中断控制器
7	FEIM	R/W	0	UART 帧错误中断屏蔽 读取时，返回 FEIM 中断的当前屏蔽值 该位置位时，可以将 FEIM 中断发送到中断控制器
6	RTIM	R/W	0	UART 接收超时中断屏蔽 读取时，返回 RTIM 中断的当前屏蔽值 该位置位时，可以将 RTIM 中断发送到中断控制器

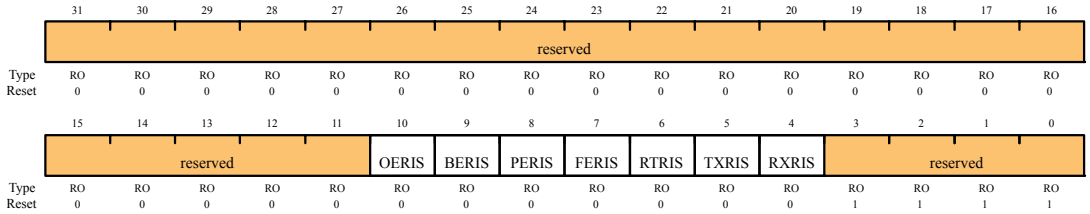
续上表

位/字段	名称	类型	复位	描述
5	TXIM	R/W	0	UART 发送中断屏蔽 读取时，返回 TXIM 中断的当前屏蔽值 该位置位时，可以将 TXIM 中断发送到中断控制器
4	RXIM	R/W	0	UART 接收中断屏蔽 读取时，返回 RXIM 中断的当前屏蔽值 该位置位时，可以将 RXIM 中断发送到中断控制器
3:0	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变。

寄存器 10：UART 原始中断状态（UARTRIS）寄存器，偏移量：0x03C

UARTRIS 寄存器是原始中断状态寄存器。读取时，该寄存器显示了相应中断的当前原始状态值。对该寄存器进行写操作没有什么用处。

UART原始中断状态（UARTRIS）寄存器
偏移量：0x03C

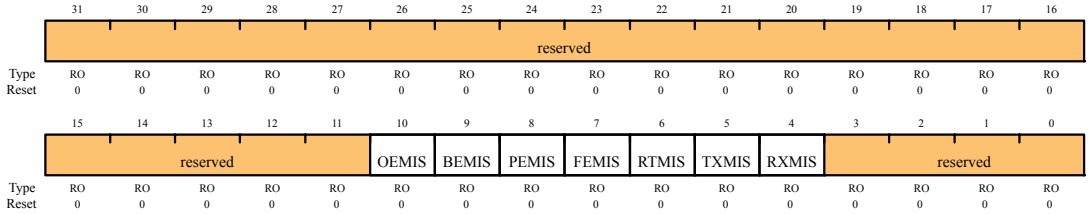


位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
10	OERIS	RO	0	UART 溢出错误的原始中断状态 显示溢出错误中断的原始（屏蔽前）中断状态
9	BERIS	RO	0	UART 中止（break）错误的原始中断状态 显示中止（break）错误中断的原始（屏蔽前）中断状态
8	PERIS	RO	0	UART 奇偶校验错误的原始中断状态 显示奇偶校验错误中断的原始（屏蔽前）中断状态
7	FERIS	RO	0	UART 帧错误的原始中断状态 显示帧错误中断的原始（屏蔽前）中断状态
6	RTRIS	RO	0	UART 接收超时的原始中断状态 显示接收超时中断的原始（屏蔽前）中断状态
5	TXRIS	RO	0	UART 发送的原始中断状态 显示发送中断的原始（屏蔽前）中断状态
4	RXRIS	RO	0	UART 接收的原始中断状态 显示接收中断的原始（屏蔽前）中断状态
3:0	保留	RO	0xF	保留位只读，并且其复位值为 0xF。

寄存器 11：UART 屏蔽后的中断状态（UARTMIS）寄存器，偏移量：0x040

UARTMIS 寄存器是屏蔽后的中断状态寄存器。读取时，该寄存器显示了相应中断当前的屏蔽状态值。对该寄存器进行写操作没有什么用处。

UART屏蔽后的中断状态（UARTMIS）寄存器
偏移量：0x040

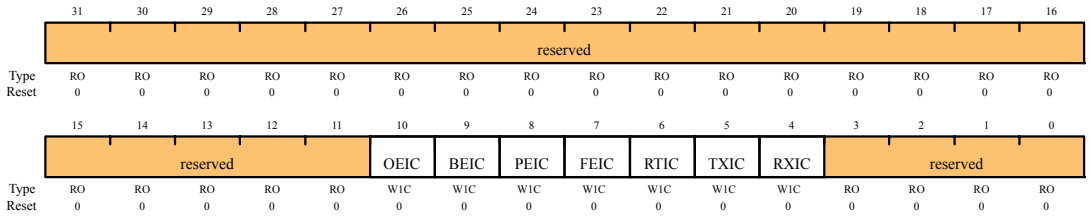


位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
10	OEMIS	RO	0	UART 溢出错误屏蔽后的中断状态 显示溢出错误中断屏蔽后的中断状态
9	BEMIS	RO	0	UART 中止（break）错误屏蔽后的中断状态 显示中止（break）错误中断屏蔽后的中断状态
8	PEMIS	RO	0	UART 奇偶校验错误屏蔽后的中断状态 显示奇偶校验错误中断屏蔽后的中断状态
7	FEMIS	RO	0	UART 帧错误屏蔽后的中断状态 显示帧错误中断屏蔽后的中断状态
6	RTMIS	RO	0	UART 接收超时屏蔽后的中断状态 显示接收超时中断屏蔽后的中断状态
5	TXMIS	RO	0	UART 发送屏蔽后的中断状态 显示发送中断屏蔽后的中断状态
4	RXMIS	RO	0	UART 接收屏蔽后的中断状态 显示接收中断屏蔽后的中断状态
3:0	保留	RO	0	保留位返回一个不确定值，并且应该永不改变。

寄存器 12：UART 中断清除（UARTICR）寄存器，偏移量：0x044

UARTICR 寄存器是中断清除寄存器。在写入 1 时，相应的中断（原始中断和已屏蔽中断，如果使能）被清除。写入 0 没什么用处。

UART中断清除（UARTICR）寄存器
偏移量：0x044



位/字段	名称	类型	复位	描述
31:11	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
10	OEIC	W1C	0	UART 溢出错误中断清除 0：对中断不起作用 1：清除中断

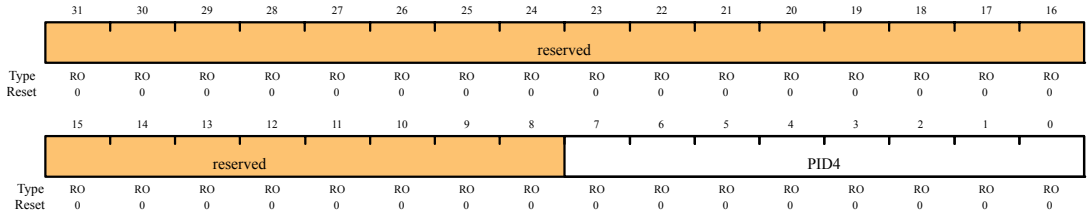
续上表

位/字段	名称	类型	复位	描述
9	BEIC	W1C	0	UART 中止（break）错误中断清除 0：对中断不起作用 1：清除中断
8	PEIC	W1C	0	UART 奇偶校验错误中断清除 0：对中断不起作用 1：清除中断
7	FEIC	W1C	0	UART 帧错误中断清除 0：对中断不起作用 1：清除中断
6	RTIC	W1C	0	UART 接收超时中断清除 0：对中断不起作用 1：清除中断
5	TXIC	W1C	0	UART 发送中断清除 0：对中断不起作用 1：清除中断
4	RXIC	W1C	0	UART 接收中断清除 0：对中断不起作用 1：清除中断
3:0	保留	RO	0	保留位返回一个不确定值，并且应该永不改变。

寄存器 13：UART 外设标识 4（UARTPeriphID4）寄存器，偏移量：0xFD0

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识4（UARTPeriphID4）寄存器
偏移量：0xFD0

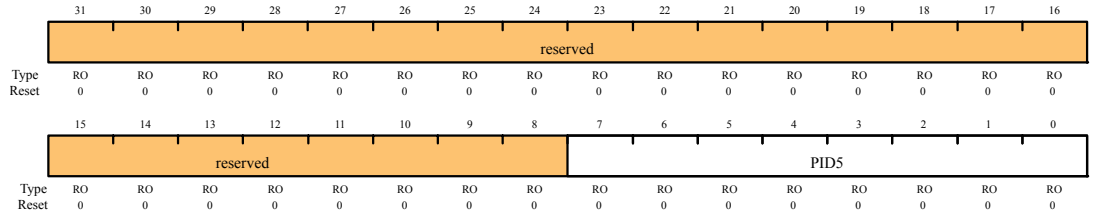


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID4	RO	0x00	UART 外设 ID 寄存器[7:0]

寄存器 14：UART 外设标识 5（UARTPeriphID5）寄存器，偏移量：0xFD4

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识5（UARTPeiriphID5）寄存器
偏移量：0xFD4

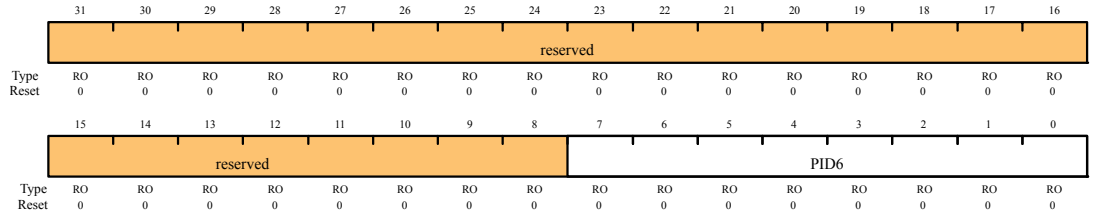


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID5	RO	0x00	UART 外设 ID 寄存器[15:8]

寄存器 15：UART 外设标识 6（UARTPeriphID6）寄存器，偏移量：0xFD8

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识6（UARTPeriphID6）寄存器
偏移量：0xFD8

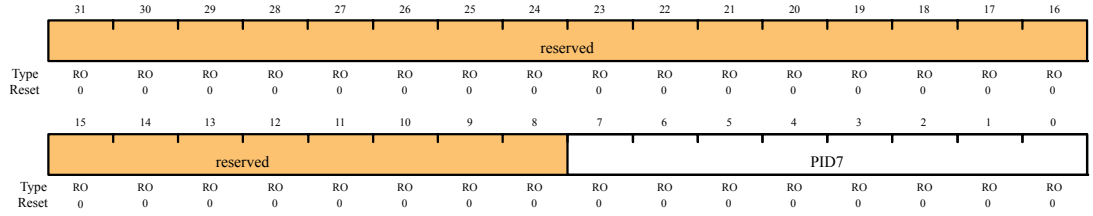


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID6	RO	0x00	UART 外设 ID 寄存器[23:16]

寄存器 16：UART 外设标识 7（UARTPeriphID7）寄存器，偏移量：0xFDC

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识7（UARTPeriphID7）寄存器
偏移量：0xFDC

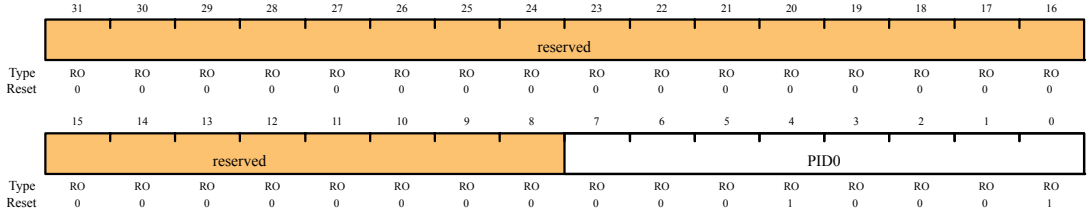


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID7	RO	0x00	UART 外设 ID 寄存器[31:24]

寄存器 17：UART 外设标识 0（UARTPeriphID0）寄存器，偏移量：0xFE0

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识0（UARTPeriphID0）寄存器
偏移量：0xFE0

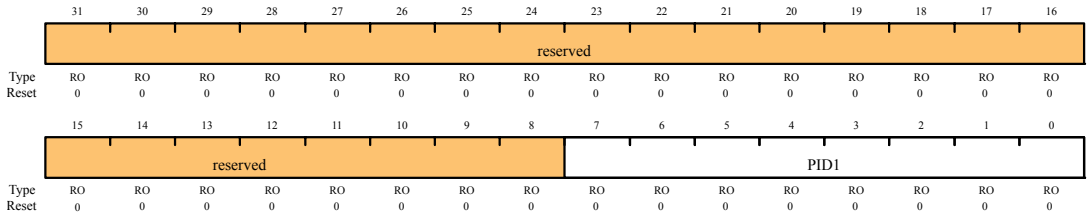


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID0	RO	0x11	UART 外设 ID 寄存器[7:0] 可以被软件用来识别该外设是否存在

寄存器 18：UART 外设标识 1（UARTPeriphID1）寄存器，偏移量：0xFE4

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识1（UARTPeriphID1）寄存器
偏移量：0xFE4

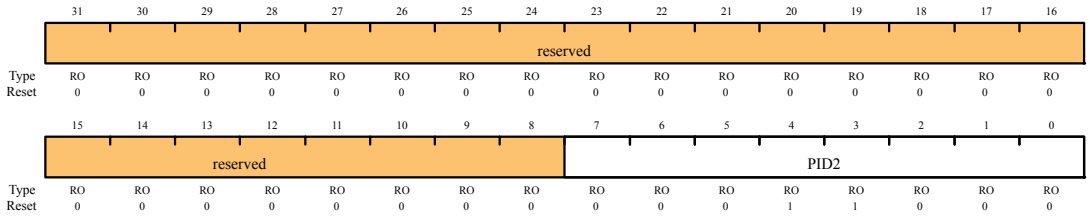


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID1	RO	0x00	UART 外设 ID 寄存器[15:8] 可以被软件用来识别该外设是否存在

寄存器 19：UART 外设标识 2（UARTPeriphID2）寄存器，偏移量：0xFE8

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识2（UARTPeriphID2）寄存器
偏移量：0xFE8

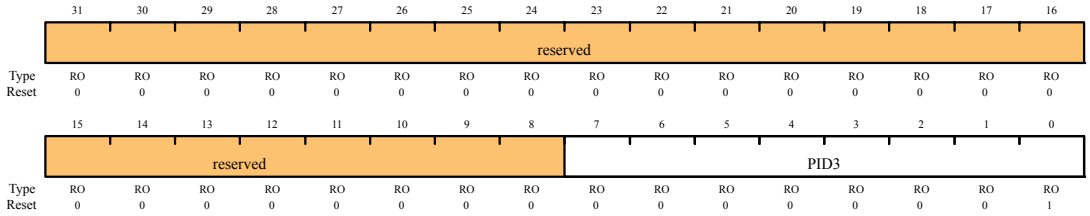


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID2	RO	0x18	UART 外设 ID 寄存器[23:16] 可以被软件用来识别该外设是否存在

寄存器 20：UART 外设标识 3（UARTPeriphID3）寄存器，偏移量：0xFEC

UARTPeriphIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART外设标识3（UARTPeriphID3）寄存器
偏移量：0xFEC

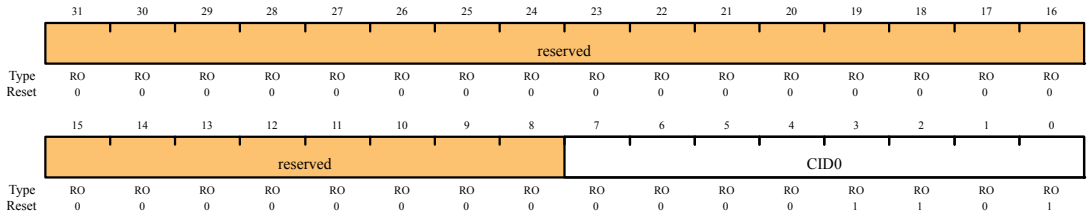


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	PID3	RO	0x11	UART 外设 ID 寄存器[31:24] 可以被软件用来识别该外设是否存在

寄存器 21：UART PrimeCell 标识 0（UARTPCellID0）寄存器，偏移量：0xFF0

UARTPCellIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART PrimeCell 标识0（UARTPCellID0）寄存器
偏移量：0xFF0

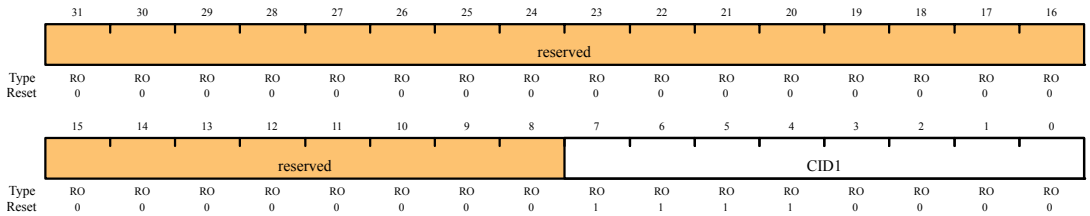


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID0	RO	0x0D	UART PrimeCell ID 寄存器[7:0] 提供软件一个标准的跨外设标识系统

寄存器 22：UART PrimeCell 标识 1（UARTPCellID1）寄存器，偏移量：0xFF4

UARTPCellIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART PrimeCell 标识1（UARTPCellID1）寄存器
偏移量：0xFF4

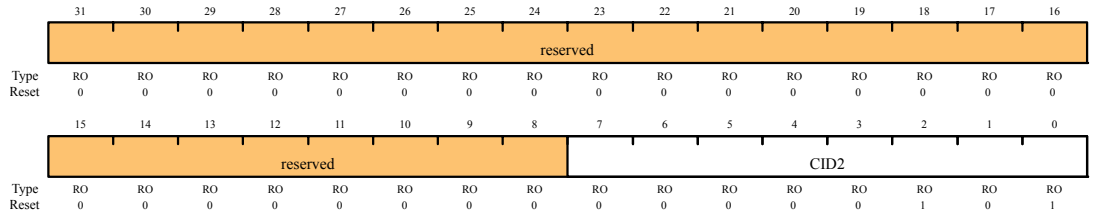


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID1	RO	0xF0	UART PrimeCeII ID 寄存器[15:8] 提供软件一个标准的跨外设标识系统

寄存器 23：UART PrimeCell 标识 2（UARTPCellID2）寄存器，偏移量：0xFF8

UARTPCellIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART PrimeCeII 标识2（UARTPCeIIID2）寄存器
偏移量：0xFF8

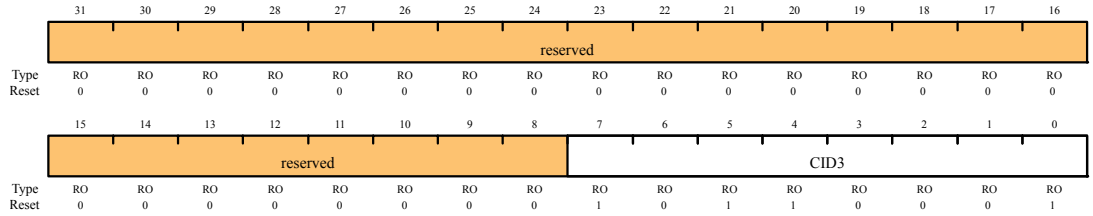


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID2	RO	0x05	UART PrimeCeII ID 寄存器[23:16] 提供软件一个标准的跨外设标识系统

寄存器 24：UART PrimeCell 标识 3（UARTPCellID3）寄存器，偏移量：0xFFC

UARTPCellIDn 寄存器是硬编码的寄存器；该寄存器中的字段决定了其复位后的值。

UART PrimeCeII标识3（UARTPCeIIID3）寄存器
偏移量：0xFFC



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位返回一个不确定的值，并且应该永不改变
7:0	CID3	RO	0xB1	UART PrimeCeII ID 寄存器[31:24] 提供软件一个标准的跨外设标识系统

第13章 同步串行接口（SSI）

Stellaris 同步串行接口是与具有 Freescale SPI、MICROWIRE、或 Texas Instruments 同步串行接口的外设器件进行同步串行通信的主机或从机接口。

Stellaris SSI 具有以下特性：

- 主机或从机操作
- 时钟位速率和预分频可编程
- 独立的发送和接收 FIFO，16 位宽，8 个单元深
- Freescale SPI、MICROWIRE、或 Texas Instruments 同步串行接口的操作可编程。
- 数据帧大小可编程，范围为 4～16 位
- 内部回送测试（loopback test）模式，可进行诊断/调试测试

13.1 模块框图

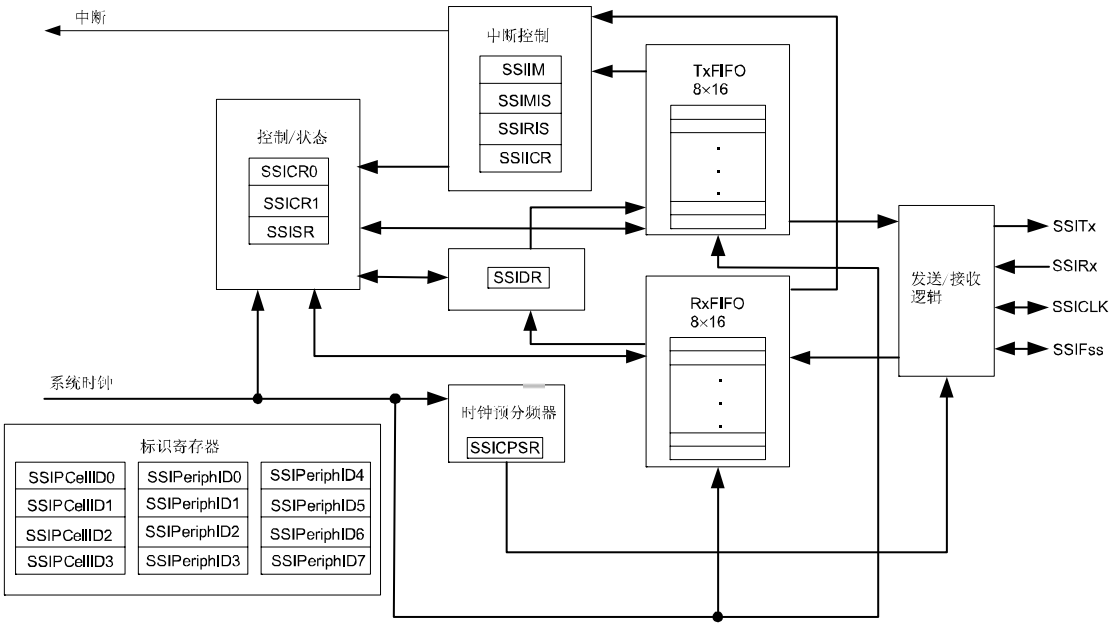


图 13.1 SSI 模块框图

13.2 功能描述

SSI 对从外设器件接收到的数据执行串行到并行转换。CPU 访问数据、控制和状态信息。发送和接收路径利用内部 FIFO 存储单元进行缓冲，该 FIFO 可在发送和接收模式下独立存储多达 8 个 16 位值。

13.2.1 位速率的产生

SSI 包含一个可编程的位速率时钟分频器和预分频器来生成串行输出时钟。尽管最大位速率由外设器件决定，但仍然支持 2MHz 甚至更高的位速率。

串行位速率通过对 50MHz 的输入时钟进行分频来获得。首先，使用范围在 2～254 的偶数分频值 CPSDVSR 对输入时钟进行分频，CPSDVSR 的值在 **SSI 时钟预分频(SSICPSR)** 寄存器中设置。然后再使用 1～256 的其中一个值（即 1+SCR）对时钟进一步分频，此处的 SCR 在 **SSI 控制 0(SSICR0)**寄存器中设置。

输出时钟 SSICLK 的频率由下式来定义：

FSSICLK=FSysCLK/(CPSDVR*(1+SCR))

注：虽然 SSICLK 发送时钟在理论上可达到 25MHz，但模块未必能在该速率下工作。对于发送操作，系统时钟至少是 SSICLK 的两倍。对于接收操作，系统时钟至少是 SSICLK 的 12 倍。

SSI的时序参数请参考 第 19 章。

13.2.2 FIFO 操作

13.2.2.1 发送 FIFO

通用发送 FIFO 是一个 16 位宽、8 单元深、先进先出的存储缓冲区。CPU 通过写 **SSI 数据(SSIDR)**寄存器来将数据写入发送 FIFO，数据在由发送逻辑读出之前一直保存在发送 FIFO 中。

当 SSI 配置为主机或从机时，并行数据在进行串行转换并通过 SSITx 管脚分别发送到相关的从机或主机之前先写入发送 FIFO。

13.2.2.2 接收 FIFO

通用接收 FIFO 是一个 16 位宽、8 单元深、先进先出的存储缓冲区。从串行接口接收到的数据在由 CPU 读出之前一直保存在缓冲区中，CPU 通过读 **SSIDR** 寄存器来访问读 FIFO。

当 SSI 配置为主机或从机时，从 SSIRx 管脚接收到的串行数据在分别并行加载到相关的从机或主机接收 FIFO 之前先进行记录（registered）。

13.2.3 中断

SSI 可在出现下列情况时产生中断：

- 发送 FIFO 服务
- 接收 FIFO 服务
- 接收 FIFO 超时
- 接收 FIFO 溢出

所有中断事件在发送到中断控制器之前要先执行“或”操作，因此，在任何给定的时刻 SSI 只能向控制器发送一个中断请求。在 4 个可单独屏蔽的中断中，每个都可以通过设置 **SSI 中断屏蔽(SSIIM)**寄存器中适当的位来屏蔽。将适当的屏蔽位置 1 可使能中断。

SSI 提供单独的输出和组合的中断输出，这样，允许使用全局中断服务程序或组合的器件驱动程序来处理中断。发送和接收动态数据流的中断与状态中断是分开的，因此，可以根据 FIFO 的触发深度（trigger level）对数据执行读和写操作。各个中断源的状态可从 **SSI 原**

始中断状态(SSIRIS)和 SSI 屏蔽后的中断状态(SSIMIS)寄存器中读取。

13.2.4 帧格式

根据所设置的数据大小，每个数据帧的长度均在 4～16 位之间，并且从最高有效位（MSB）开始发送。此处，有 3 种基本的帧类型可供选择：

- Texas Instruments 同步串行
- Freescale SPI
- MICROWIRE

对于上述 3 种帧格式，串行时钟(SSICLK)在 SSI 空闲时保持不活动状态，只有当数据的发送或接收处于活动状态时，SSICLK 才在设置好的频率下工作。利用 SSICLK 的空闲状态可提供接收超时指示。如果一个超时周期之后接收 FIFO 仍含有数据，则产生超时指示。

对于 Freescale SPI 和 MICROWIRE 这两种帧格式，串行帧（SSIFss）管脚为低电平有效，并在整个帧的传输过程中保持有效（被下拉）。

而对于 Texas Instruments 同步串行帧格式，在发送每个帧之前，SSIFss 管脚会发出一个以上升沿开始并持续一个时钟周期的脉冲。在这种帧格式中，SSI 和片外从器件在 SSICLK 的上升沿驱动各自的输出数据，并在下降沿锁存另一个器件的数据。

不同于其它两种全双工传输的帧格式，在半双工下工作的 MICROWIRE 格式使用特殊的主-从消息技术。在该模式中，当帧开始传输时向片外从机发送 8 位控制消息。在发送过程中，SSI 不会接收到任何输入数据。在消息发送完毕之后，片外从机对消息进行译码，并在 8 位控制消息的最后一位发送完成之后等待一个串行时钟周期，之后以请求的数据来响应。返回的数据，其长度在 4～16 位之间，这样，无论在何处，总的帧长度都在 13～25 位之间。

13.2.4.1 Texas Instruments 同步串行帧格式

图 13.2 显示了一次传输的Texas Instruments同步串行帧格式。

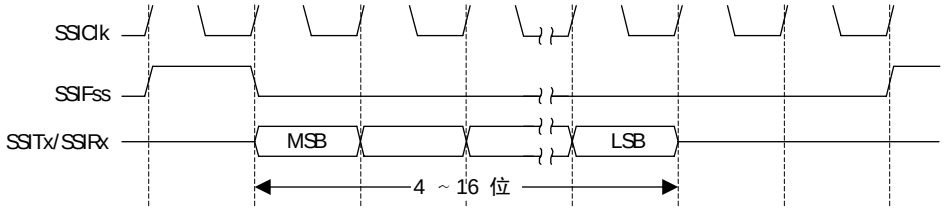


图 13.2 TI 同步串行帧格式（单次传输）

在该模式中，任何时候当 SSI 空闲时，SSICLK 和 SSIFss 被强制为低电平，发送数据线 SSITx 为三态。一旦发送 FIFO 的底部入口包含数据，SSIFss 就会变为高电平并持续一个 SSICLK 周期。要发送的值也从发送 FIFO 传输到发送逻辑的串行移位寄存器中。在 SSICLK 的下一个上升沿，4～16 位数据帧的 MSB 从 SSITx 管脚移出。同样，接收到的数据的 MSB 也通过片外串行从器件移到 SSIRx 管脚上。

然后，SSI 和片外从器件在 SSICLK 的每一个下降沿时刻将数据位逐个移入各自的串行移位器中。在锁存了 LSB 之后的第一个 SSICLK 上升沿上，接收数据从串行移位器传输到接收 FIFO。

图 13.3 显示了背对背（back-to-back）传输时的Texas Instruments同步串行帧格式。

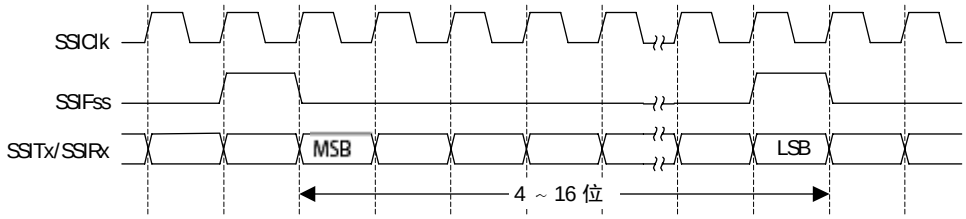


图 13.3 TI 同步串行帧格式（连续传输）

13.2.4.2 Freescale SPI 帧格式

Freescale SPI 接口是一个 4 线接口，其中 SSIFss 信号用作从机选择。Freescale SPI 格式的主要特性为：SSICLK 信号的不活动状态和相位均通过 **SSISCR0** 控制寄存器中的 SPO 和 SPH 位来设置。

SPO 时钟极性位

当 SPO 时钟极性控制位为低时，它在 SSICLK 管脚上产生稳定的低电平值。如果 SPO 位为高，则在没有进行数据传输的情况下，它在 SSICLK 管脚上产生一个稳定的高电平值。

SPH 相位控制位

SPH 相位控制位用来选择捕获数据的时钟边沿并允许边沿改变状态。SPH 在第一个传输位上的影响最大，因为它可以在第一个数据捕获边沿之前允许或不允许一次时钟转换。当 SPH 相位控制位为低时，在第一个时钟边沿转换时捕获数据。如果 SPH 位为高，则在第二个时钟边沿转换时捕获数据。

13.2.4.3 SPO=0 和 SPH=0 时的 Freescale SPI 帧格式

SPO=0 和SPH=0 时，Freescale SPI帧格式的单次和连续传输信号序列如 图 13.4 和 图 13.5 所示。

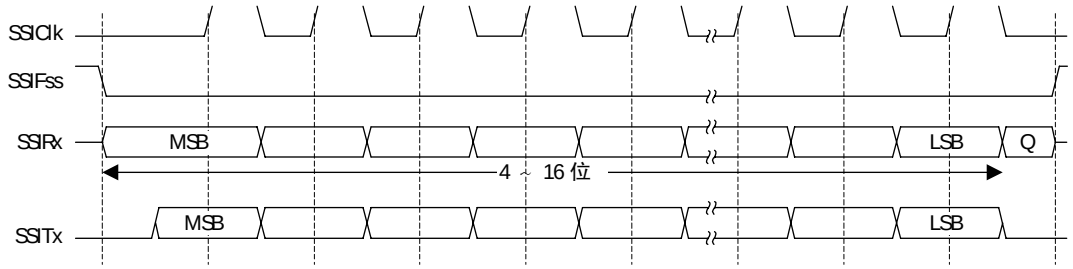


图 13.4 SPO=0 和 SPH=0 时的 Freescale SPI 帧格式（单次传输）

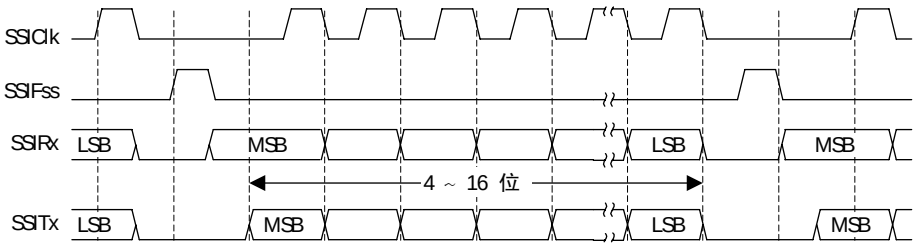


图 13.5 SPO=0 和 SPH=0 时的 Freescale SPI 帧格式（连续传输）

在上述配置中，当 SSI 处于空闲周期时：

- SSICLK 被强制为低电平
- SSIFss 被强制为高电平
- 发送数据线 SSITx 被强制为低电平
- 当 SSI 配置为主机时，使能 SSICLK 端口
- 当 SSI 配置为从机时，禁止 SSICLK 端口

如果 SSI 使能并且在发送 FIFO 中含有有效的数据，则通过将 SSIFss 主机信号驱动为低电平表示发送操作开始。这使得从机数据能够放在主机的 SSIRx 输入线上。主机 SSITx 输出端口使能。

在半个 SSICLK 周期之后，有效的主机数据传输到 SSITx 管脚。既然主机和从机数据都已设置好，则在下半个 SSICLK 周期之后，SSICLK 主机时钟管脚变为高电平。

这时，数据在 SSICLK 信号的上升沿被捕获，在 SSICLK 的下降沿进行传输。

如果传输一个字，则在数据字的所有位都已传输完之后，SSIFss 线在捕获到最后一个位之后的一个 SSICLK 周期返回到其空闲的高电平状态。

而在连续的背对背传输中，SSIFss 信号必须在每次数据字的传输之间保持高电平。因为当 SPH 位为逻辑 0 时，从机选择管脚将冻结串行外设寄存器中的数据，使其不能修改。因此，主器件必须在每次数据传输之间将从器件的 SSIFss 管脚拉高，以便使能串行外设的数据写操作。当连续传输完成时，SSIFss 线将在捕获到最后一位之后的一个 SSICLK 周期返回到其空闲的高电平状态。

13.2.4.4 SPO=0 和 SPH=1 时的 Freescale SPI 帧格式

SPO=0 和SPH=1 时，Freescale SPI帧格式的传输信号序列如 图 13.6 所示，该图涵盖了单次和连续传输这两种情况。

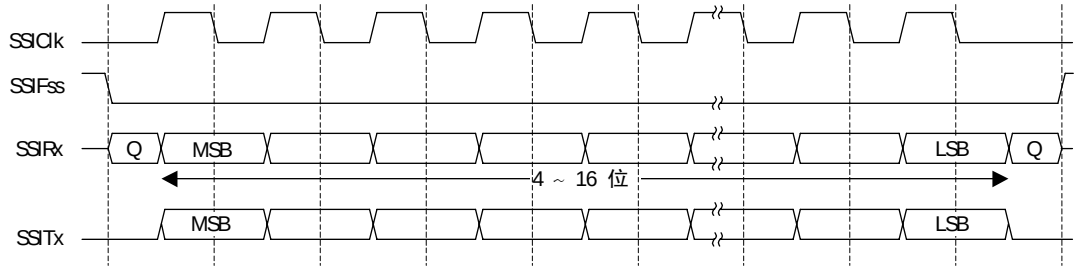


图 13.6 SPO=0 和 SPH=1 时的 Freescale SPI 帧格式

在该配置中，当 SSI 处于空闲周期时：

- SSICLK 被强制为低电平
- SSIFss 被强制为高电平
- 发送数据线 SSITx 被强制为低电平
- 当 SSI 配置为主机时，使能 SSICLK 端口
- 当 SSI 配置为从机时，禁止 SSICLK 端口

如果 SSI 使能并且在发送 FIFO 中含有有效的数据，则通过将 SSIFss 主机信号驱动为低

电平来表示发送操作开始。主机 SSITx 输出使能。在后半个 SSICLK 周期之后，主机和从机有效数据能够放在各自的传输线上。同时，利用一个上升沿跳变将 SSICLK 使能。

这时，数据在 SSICLK 信号的下降沿被捕获，在 SSICLK 的上升沿进行传输。

如果传输一个字，则在所有位传输完之后，SSIFss 线在捕获到最后一个位之后的一个 SSICLK 周期返回到其空闲的高电平状态。

如果是背对背（back-to-back）传输，则 SSIFss 管脚在连续的数据字之间保持为低电平，连续传输的结束情况与单字传输的相同。

13.2.4.5 SPO=1 和 SPH=0 时的 Freescale SPI 帧格式

SPO=1 和SPH=0 时，Freescale SPI帧格式的单次和连续传输信号序列如 图 13.7 和 图 13.8 所示。

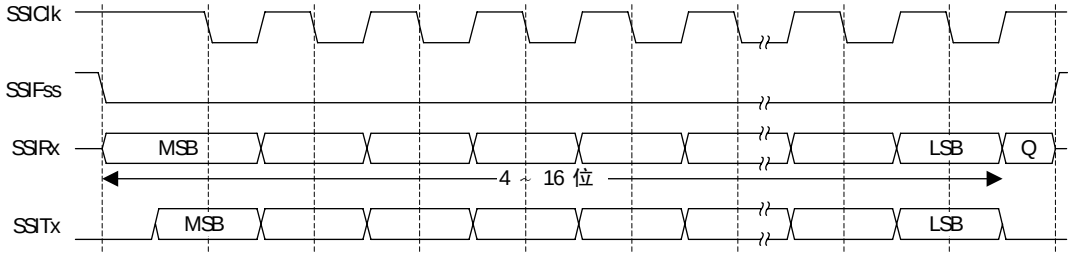


图 13.7 SPO=1 和 SPH=0 时的 Freescale SPI 帧格式（单次传输）

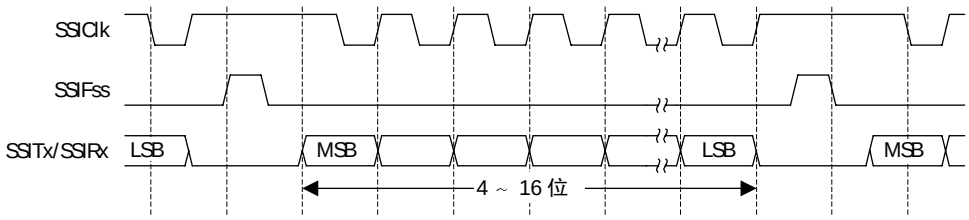


图 13.8 SPO=1 和 SPH=0 时的 Freescale SPI 帧格式（连续传输）

在该配置中，当 SSI 处于空闲周期时：

- SSICLK 被强制为高电平
- SSIFss 被强制为高电平
- 发送数据线 SSITx 被强制为低电平
- SSI 配置为主机时，使能 SSICLK 端口
- SSI 配置为从机时，禁止 SSICLK 端口

如果 SSI 使能并且在发送 FIFO 中含有有效的数据，则通过将 SSIFss 主机信号驱动为低电平来表示传输操作开始，这可使从机数据立即传输到主机的 SSIRx 线上。主机 SSITx 输出端口使能。

半个周期之后，有效的主机数据传输到 SSITx 线上。既然主机和从机的有效数据都已设置好，则在下半个 SSICLK 周期之后，SSICLK 主机时钟管脚变为低电平。这表示数据在 SSICLK 的下降沿被捕获，在 SSICLK 信号的上升沿进行传输。

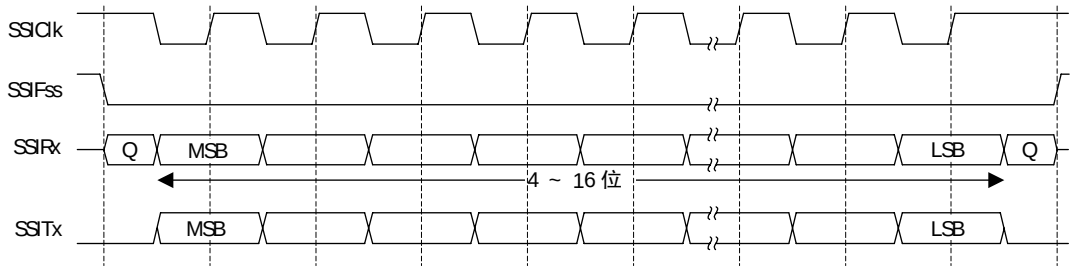
如果传输一个字，则在数据字的所有位传输完之后，SSIFss 线在最后一个位传输完之后

的一个 SSICLK 周期返回到其空闲的高电平状态。

而在连续的背对背（back-to-back）传输中，SSIFss 信号必须在每次数据字的传输之间保持高电平。因为当 SPH 位为逻辑 0 时，从机选择管脚将冻结串行外设寄存器中的数据，使其不能修改。因此，每次数据传输之间，主器件必须将从器件的 SSIFss 管脚拉高，以便使能串行外设的数据写操作。在连续传输完成时，SSIFss 管脚在捕获到最后一个位之后的一个 SSICLK 周期返回其空闲状态。

13.2.4.6 SPO=1 和 SPH=1 时的 Freescale SPI 帧格式

SPO=1 和SPH=1 时，Freescale SPI帧格式的传输信号序列如 图 13.9 所示。该图涵盖了单次和连续传输两种情况。



注：图中的 Q 表示未定义

图 13.9 SPO=1 和 SPH=1 时的 Freescale SPI 帧格式

在该配置中，当 SSI 处于空闲周期时：

- SSICLK 被强制为高电平
- SSIFss 被强制为高电平
- 发送数据线 SSIFss 被强制为低电平
- 当 SSI 配置为主机时，使能 SSICLK 端口
- 当 SSI 配置为从机时，禁止 SSICLK 端口

如果 SSI 使能并且在发送 FIFO 中含有有效的数据，则通过将 SSIFss 主机信号驱动为低电平来表示发送操作开始。主机 SSITx 输出端口使能。在下半个 SSICLK 周期之后，主机和从机数据都能够放在各自的传输线上。同时，利用 SSICLK 的下降沿跳变将 SSICLK 使能。然后，数据在 SSICLK 的上升沿被捕获，在 SSICLK 信号的下降沿进行传输。

在所有位传输完之后，如果是单个字传输，则在最后一个位传输完之后的一个 SSICLK 周期中，SSIFss 线返回到其空闲的高电平状态。

而对于连续的背对背(back-to-back)传输，$SSIF_{SS}$ 管脚保持其有效的低电平状态，直至最后一个字的最后一位捕获完成，再返回其上述的空闲状态。

对于连续的背对背(back-to-back)传输，SSIFss 管脚在连续的数据字之间保持为低电平，连续传输的结束情况与单个字传输相同。

13.2.4.7 MICROWIRE 帧格式

图 13.10 显示了单次传输的MICROWIRE帧格式，而 图 13.11 为该格式的背对背（back-to-back）传输情况。

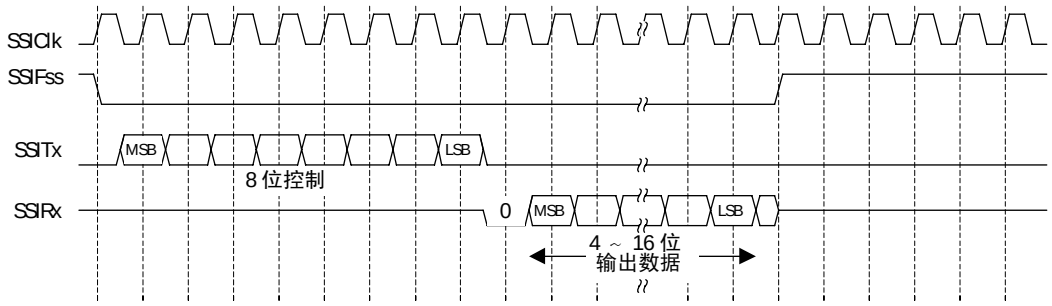


图 13.10 MICROWIRE 帧格式（单次传输）

MICROWIRE 格式与 SPI 格式非常类似，不同的是其采用的是使用主-从消息传递技术的半双工模式而并非全双工模式。每次串行传输都由 SSI 向片外从器件发送 8 位控制字开始。在此传输过程中，SSI 不会接收到输入的数据。在消息发送完毕之后，片外从机对消息进行译码，并在 SSI 将 8 位控制消息的最后一位发送完成之后等待一个串行时钟周期，之后从机以请求的数据来响应。返回的数据在长度上为 4～16 位，使得任何地方的总的帧长度都为 13～25 位。

在该配置中，当 SSI 处于空闲状态时：

- SSICLK 被强制为低电平
- SSIFss 被强制为高电平
- 发送数据线 SSITx 被强制为低电平。

通过向发送 FIFO 写入一个控制字节可以触发一次传输。在 SSIFss 的下降沿，发送 FIFO 底部入口包含的值被传输到发送逻辑的串行移位寄存器中，而 8 位控制帧的 MSB 被移出到 SSITx 管脚上。在该控制帧的传输期间 SSIFss 保持低电平，SSIRx 管脚保持三态。

片外串行从器件在每个 SSICLK 的上升沿处将每个控制位锁存到其串行移位器中。在将最后一位锁存之后，从器件在一个时钟周期的等待状态期间对控制字节进行译码，并且从机通过将数据发送回 SSI 来响应。每个数据位在 SSICLK 的下降沿时刻被驱动到 SSIRx 线上。SSI 在 SSICLK 的上升沿时依次将每个位锁存。在帧传输结束时，对于单次传输，SSIFss 信号在最后一位已锁存到接收串行移位器之后的一个时钟周期被拉为高电平，这使得数据传输到接收 FIFO 中。

注：在接收移位器将 LSB 锁存之后的 SSICLK 的下降沿上或在 SSIFss 管脚变为高电平时，片外从器件能够将接收线置为三态。

对于连续传输，数据传输的开始与结束与单次传输相同。但 SSIFss 线持续有效（保持低电平），并且数据传输以背对背(back-to-back)方式产生。在从当前帧接收到数据的最低有效位（LSB）之后紧跟着下一帧的控制字节。在当前帧的 LSB 锁存到 SSI 之后，所接收到的每个值在 SSICLK 的下降沿时刻从接收移位器中进行传输。

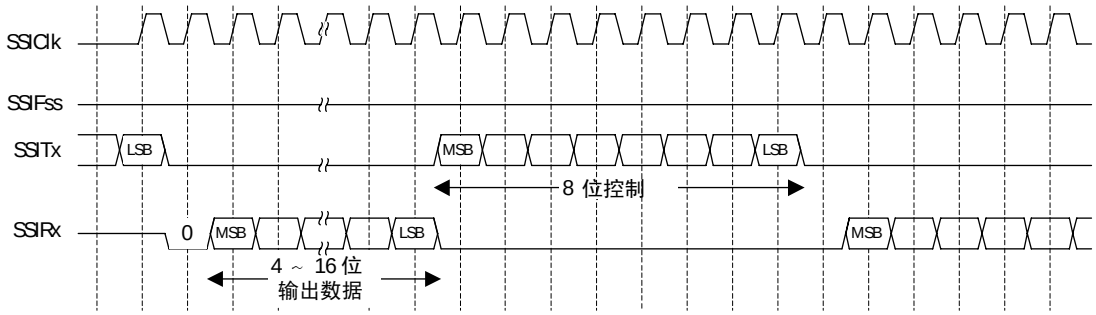


图 13.11 MICROWIRE 帧格式（连续传输）

在 MICROWIRE 模式中，当 SSIFss 变为低电平之后，SSI 从机在 SSICLK 的上升沿时刻对接收数据的第一个位进行采样。用来驱动自由运行的 SSICLK 的主机必须确保 SSIFss 信号相对于 SSICLK 的上升沿具有足够的建立时间和保持时间裕量（setup and hold margins）。

图 13.12 阐明了建立和保持时间要求。相对于SSI从机对接收数据的第一位进行采样时所在的SSICLK上升沿，SSIFss的建立时间至少必须是SSI操作时钟（SSICLK）的周期的两倍。相对于该边沿之前的SSICLK上升沿，SSIFss至少必须具有一个SSICLK周期的保持时间。

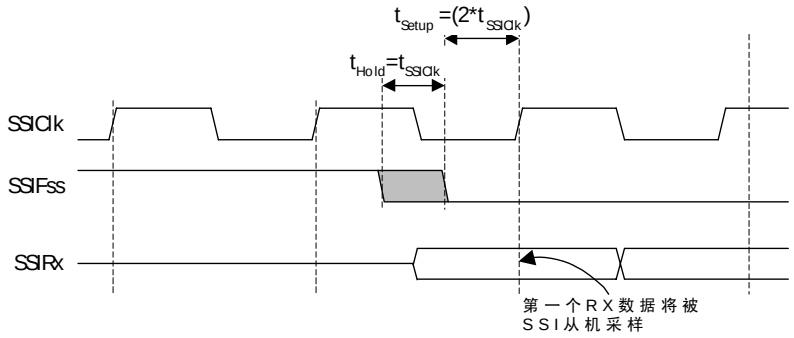


图 13.12 MICROWIRE 帧格式，SSIFss 输入建立和保持时间要求

13.3 初始化和配置

在使用 SSI 时，必须通过置位 **RCGC1** 寄存器中的 SSI 位来使能 SSI 外设时钟。针对不同的帧格式，SSI 可通过以下步骤进行配置：

1. 确保在对任何配置进行更改之前先将 **SSICR1** 寄存器中的 SSE 位禁止。
2. 选择 SSI 为主机或从机：
   a 作为主机时，将 **SSICR1** 寄存器设置为 0x00000000
   b 作为从机时（输出使能），将 **SSICR1** 寄存器设置为 0x00000004
   c 作为从机时（输出禁止），将 **SSICR1** 寄存器设置为 0x0000000C
3. 通过写 **SSICPSR** 寄存器来配置时钟预分频除数
4. 写 **SSICR0** 寄存器，实现以下配置：
   — 串行时钟率（SCR）
   — 如果使用 Freescale SPI 模式，则配置所需的时钟相位/极性（SPH 和 SPO）
   — 协议模式：Freescale SPI、TI SSF、MICROWIRE（FRF）

— 数据大小（DSS）

5. 通过置位 **SSICR1** 寄存器的 SSE 位来使能 SSI

举例：假定 SSI 的配置如下：

- 主机操作
- Freescale SPI 模式（SPO=1，SPH=1）
- 1Mbps 位速率
- 8 个数据位

如果系统时钟为 20MHz，则位速率的计算如下：

FSSICLK=FSysCLK/(CPSDVR*(1+SCR))＇1x10^6＝20x10^6/(CPSDVR*(1+SCR))

在此情况下，如果 CPSDVR=2，则 SCR 必须为 9。

具体的配置序列如下：

1. 确保 **SSICR1** 寄存器的 SSE 位禁止
2. 向 **SSICR1** 寄存器写入 0x00000000
3. 向 **SSICPSR** 寄存器写入 0x00000002
4. 向 **SSICR0** 寄存器写入 0x000009C7
5. 将 **SSICR1** 寄存器的 SSE 位置 1 来使能 SSI

13.4 寄存器映射

表 13.1 列出了SSI寄存器。偏移量相对于SSI的基址 0x40008000，并在寄存器地址上采用十六进制递增的方式列出。

注：在对任何控制寄存器重新编程之前，必须将 SSI 禁止（见 **SSICR1** 寄存器的 SSE 位）。

表 13.1 SSI 寄存器映射

偏移	名称	复位	类型	描述
0x000	SSICR0	0x00000000	RW	控制 0
0x004	SSICR1	0x00000000	RW	控制 1
0x008	SSIDR	0x00000000	RW	数据
0x00C	SSISR	0x00000003	RO	状态
0x010	SSICPSR	0x00000000	RW	时钟预分频
0x014	SSIIM	0x00000000	RW	中断屏蔽
0x018	SSIRIS	0x00000008	RO	原始中断状态
0x01C	SSIMIS	0x00000000	RO	屏蔽后的中断状态
0x020	SSIICR	0x00000000	WIC	中断清零
0xFD0	SSIPeriphID4	0x00000000	RO	外设标识 4
0xFD4	SSIPeriphID5	0x00000000	RO	外设标识 5
0xFD8	SSIPeriphID6	0x00000000	RO	外设标识 6

续上表

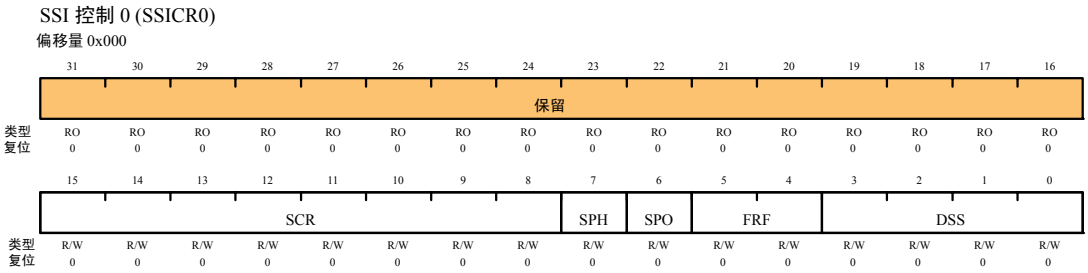
偏移	名称	复位	类型	描述
0xFDC	SSIPeriphID7	0x00000000	RO	外设标识 7
0xFE0	SSIPeriphID0	0x00000022	RO	外设标识 0
0xFE4	SSIPeriphID1	0x00000000	RO	外设标识 1
0xFE8	SSIPeriphID2	0x00000018	RO	外设标识 2
0xFEC	SSIPeriphID3	0x00000001	RO	外设标识 3
0xFF0	SSIPCellID0	0x0000000D	RO	PrimeCell 标识 0
0xFF4	SSIPCellID1	0x000000F0	RO	PrimeCell 标识 1
0xFF8	SSIPCellID2	0x00000005	RO	PrimeCell 标识 2
0xFFC	SSIPCellID3	0x000000B1	RO	PrimeCell 标识 3

13.5 寄存器描述

本章余下的内容将按地址偏移量的数字顺序列举并描述 SSI 寄存器。

寄存器 1：SSI 控制 0（SSICR0），偏移量 0x000

SSICR0 为控制寄存器 0，其位域用来控制 SSI 模块内的各种功能。诸如协议模式、时钟速率和数字大小等功能都在该寄存器中配置。



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
15:8	SCR	R/W	0	SSI 串行时钟速率 SCR 的值用来产生 SSI 的发送和接收位速率。该位速率为： $BR=F_{SSICLK}/(CPSDVR*(1+SCR))$ 此处，CPSDVR 为 2-254 之间的一个偶数值，在 **SSICPSR** 寄存器中设置，SCR 为 0-255 之间的一个值。
7	SPH	R/W	0	SSI 串行时钟相位 该位只适用于 Freescale SPI 格式。 SPH 控制位用来选择捕获数据的时钟边沿并允许边沿改变状态。SPH 在第一个传输位上的影响最大，因为它能够在第一个数据捕获边沿之前允许或不允许一次时钟跳变。 当 SPH 位为 0 时，数据在第一个时钟边沿转换时捕获，如果 SPH 为 1，则数据在第二个时钟边沿转换时捕获。

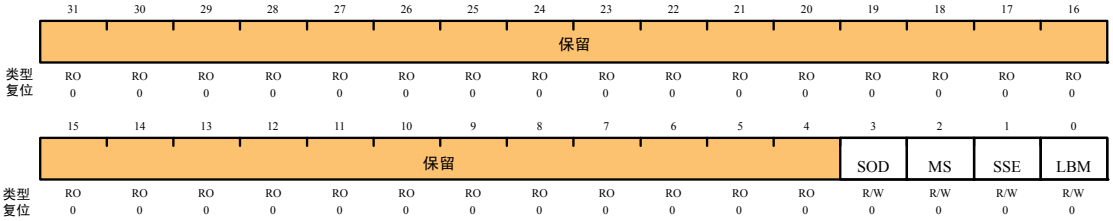
续上表

位/字段	名称	类型	复位	描述
6	SPO	R/W	0	SSI 串行时钟极性 该位只适用于 Freescale SPI 格式。 当 SPO 为 0 时，它在 SSICLK 管脚上产生稳定的低电平，如果 SPO 为 1，则在没有传输数据时在 SSICLK 管脚上产生稳定的高电平。
5:4	FRF	R/W	0	SSI 帧格式选择 FRF 的值定义如下： FRF 帧格式 00 Freescale SPI 帧格式 01 Texas Instruments 同步串行帧格式 10 MICROWIRE 帧格式 11 保留
3:0	DSS	R/W	0	SSI 数据大小选择 DSS 的值定义如下： DSS 数据大小 0000- 0010 保留 0011 4 位数据 0100 5 位数据 0101 6 位数据 0110 7 位数据 0111 8 位数据 1000 9 位数据 1001 10 位数据 1010 11 位数据 1011 12 位数据 1100 13 位数据 1101 14 位数据 1110 15 位数据 1111 16 位数据

寄存器 2：SSI 控制 1（SSICR1），偏移量 0x004

SSICR1 为控制寄存器 1，其位域用来控制 SSI 模块内的各种功能。主机和从机模式功能就由该寄存器控制。

SSI 控制1 (SSCR1)
偏移量 0x004



位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
3	SOD	R/W	0	SSI 从机模式输出禁止 该位只在从机模式（MS=1）中使用。在多从机系统中，SSI 主机可以向系统中的所有从机广播一则消息，同时能保证只有一个从机将数据驱动到串行输出线上。在这样的系统中，多个从机的 TXD 线可以连在一起。在这样的系统中操作时，需对 SOD 位进行配置以使 SSI 模式不驱动 SSITx 管脚。 0：SSI 能够在从机输出模式中驱动 SSITx 输出。 1：SSI 不可以在从机模式中驱动 SSITx 输出。
2	MS	R/W	0	SSI 主/从选择 该位选择主机或从机模式并且只有当 SSI 禁止（SSE=0）时才能修改。 0：器件配置为主机 1：器件配置为从机
1	SSE	R/W	0	SSI 同步串行端口使能 将该位置位可使能 SSI 操作 0：SSI 操作禁止 1：SSI 操作使能 注：在对任何控制器重新编程之前必须先把该位设置为 0。
0	LBM	R/W	0	SSI 还回（loopback）模式 将该位置位可使能回送（loopback）测试模式 0：正常的串行端口操作使能 1：发送串行移位寄存器的输出与接收串行移位寄存器的输入在内部相连。

寄存器 3：SSI 数据（SSIDR），偏移量 0x008

SSIDR 为 16 位宽的数据寄存器。对其进行读操作也就是对接收 FIFO 的入口（由当前 FIFO 读指针来指向）进行访问。当 SSI 接收逻辑从输入的数据帧中将数据转移出来后，将它们放入接收 FIFO 的入口（由当前 FIFO 写指针来指向）。

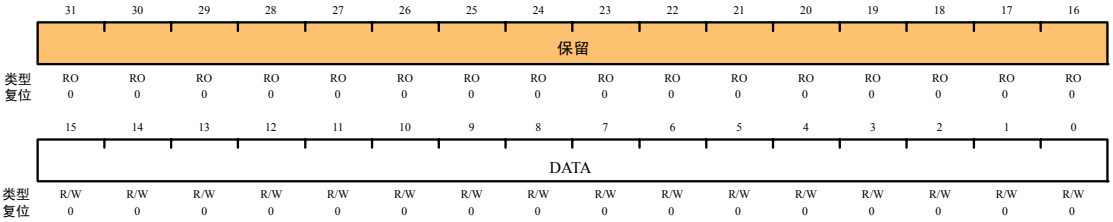
对 **SSIDR** 进行写操作也就是将数据写入发送 FIFO 的入口（由写指针来指向）。发送逻辑从发送 FIFO 中一次移出一个数据值。这个数据值被加载到发送串行移位器中，然后以设置好的位速率串行移出到 SSITx 管脚。

当所选的数据大小小于 16 位时，用户必须正确调整写入发送 FIFO 的数据。发送逻辑忽略未使用的位。小于 16 位的接收数据在接收缓冲区中自动调整。

当 SSI 设置为 MICROWIRE 帧格式时，发送数据的默认大小为 8 位（忽略最高有效字

节），接收数据的大小由程序员控制。即使当 **SSICR1** 寄存器的 SSE 位设置为 0 时，发送 FIFO 和接收 FIFO 也不会被清零。这样，可在使能 SSI 之前使用软件来填充发送 FIFO。

SSI 数据 (SSIDR)
偏移量 0x008

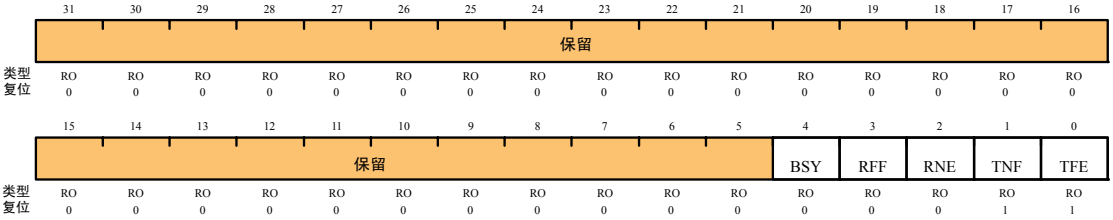


位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
15:0	DATA	R/W	0	SSI 接收/发送数据 对该字段的读操作即是读接收 FIFO，写操作即是写发送 FIFO。 当 SSI 设置为数据大小小于 16 位时，软件必须正确地调整数据。发送逻辑将忽略顶部未使用的位。接收逻辑自动调整数据。

寄存器 4：SSI 状态（SSISR），偏移量 0x00C

SSISR 是一个状态寄存器，其位域用来表示 FIFO 的填充状态以及 SSI 忙状态。

SSI 状态 (SSISR)
偏移量 0x00C



位/字段	名称	类型	复位	描述
31:5	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
4	BSY	RO	0	SSI 忙状态位 0：SSI 空闲 1：SSI 当前正在发送和/或接收帧，或者发送 FIFO 不为空。
3	RFF	RO	0	SSI 接收 FIFO 满 0：接收 FIFO 未满 1：接收 FIFO 满
2	RNE	RO	0	SSI 接收 FIFO 不为空 0：接收 FIFO 为空 1：接收 FIFO 不为空
1	TNF	RO	1	SSI 发送 FIFO 不为满 0：发送 FIFO 满 1：发送 FIFO 未满

续上表

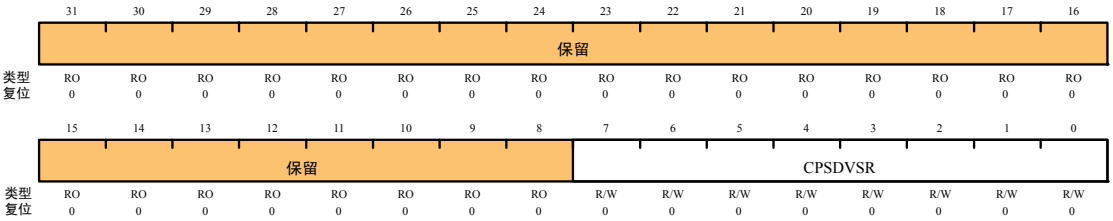
位/字段	名称	类型	复位	描述
0	TFE	RO	1	SSI 发送 FIFO 为空 0：发送 FIFO 不为空 1：发送 FIFO 为空

寄存器 5：SSI 时钟预分频（SSICPSR），偏移量 0x010

SSICPSR 为时钟预分频寄存器。它指定了分频因子，在进一步使用系统时钟之前必须根据该分频因子对系统时钟进行分频。

写入该寄存器的值必须是 2-254 之间的一个偶数。所设的值的最低有效位硬编码为 0。如果向该寄存器写入奇数，则该寄存器读操作返回的值中，最低有效位为 0。

SSI 时钟预分频 (SSICPSR)
偏移量 0x010



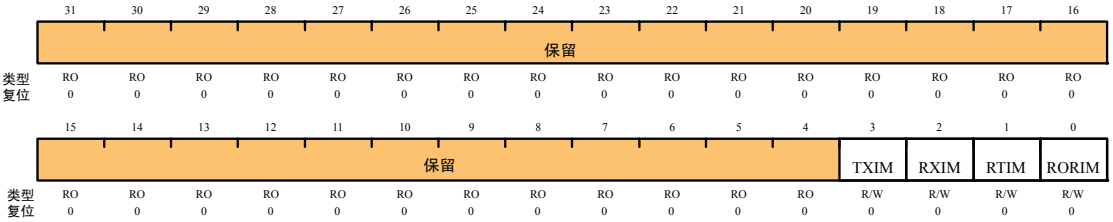
位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	CPSDVSR	R/W	0	SSI 时钟预分频因子 该值必须为 2-254 之间的一个偶数，具体取值由 SSICLK 的频率决定。执行读操作时，LSB 的返回值始终为 0。

寄存器 6：SSI 中断屏蔽（SSIIM），偏移量 0x014

SSIIM 为中断屏蔽置位或清零寄存器。它是一个读/写寄存器，复位时所有位都清零。

对该寄存器执行读操作将获得相关中断屏蔽的当前值。向寄存器的特定位写 1 将设置屏蔽，使得中断能够读出，写 0 可清除对应的中断屏蔽。

SSI 中断屏蔽 (SSIIM)
偏移量 0x014



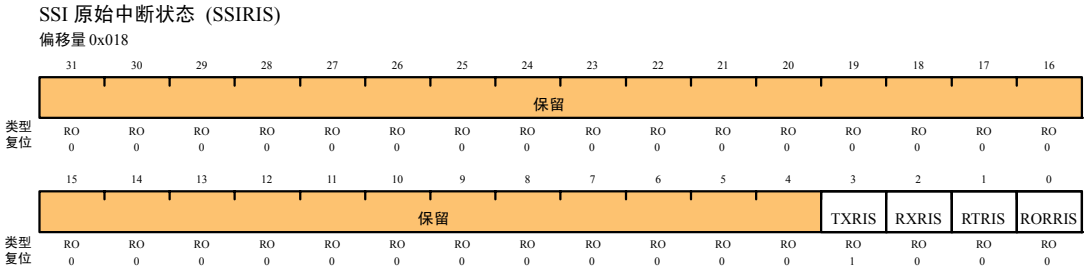
位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
3	TXIM	R/W	0	SSI 发送 FIFO 中断屏蔽 0：TX FIFO 半空或少于半空条件中断被屏蔽 1：TX FIFO 半空或少于半空条件没有被屏蔽

续上表

位/字段	名称	类型	复位	描述
2	RXIM	R/W	0	SSI 接收 FIFO 中断屏蔽 0：RX FIFO 半满或少于半满条件中断被屏蔽 1：RX FIFO 半满或少于半满条件中断没有被屏蔽
1	RTIM	R/W	0	SSI 接收超时中断屏蔽 0：RX FIFO 超时中断被屏蔽 1：RX FIFO 超时中断没有被屏蔽
0	RORIM	R/W	0	SSI 接收溢出中断屏蔽 0：RX FIFO 溢出中断被屏蔽 1：RX FIFO 溢出中断没有被屏蔽

寄存器 7：SSI 原始中断状态（SSIRIS），偏移量 0x018

SSIRIS 为原始中断状态寄存器。对其执行读操作可获得对应中断在屏蔽之前的当前原始中断状态值。写操作无效。



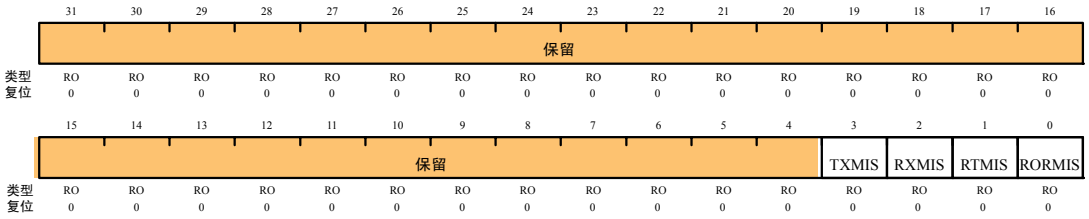
位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
3	TXRIS	RO	1	SSI 发送 FIFO 原始中断状态 该位置位表示发送 FIFO 为半空或多于半空。
2	RXRIS	RO	0	SSI 接收 FIFO 原始中断状态 该位置位表示接收 FIFO 为半空或多于半空。
1	RTRIS	RO	0	SSI 接收超时原始中断状态 该位置位表示发生接收超时
0	RORRIS	RO	0	SSI 接收溢出原始中断状态 该位置位表示接收 FIFO 溢出

寄存器 8：SSI 屏蔽后的中断状态（SSIMIS），偏移量 0x01C

SSIMIS 为屏蔽后的中断状态寄存器。对其执行读操作将获得对应中断在屏蔽后的当前状态值。写操作无效。

SSI 屏蔽后的中断状态 (SSIMIS)

偏移量 0x01C



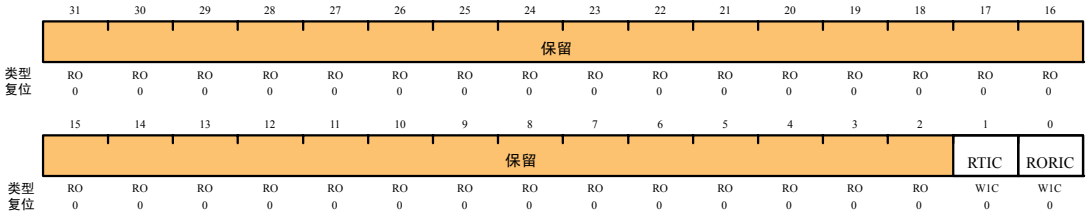
位/字段	名称	类型	复位	描述
31:4	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
3	TXMIS	RO	0	SSI 发送 FIFO 屏蔽后的中断状态 该位置位表示发送 FIFO 的状态为半空或多于半空。
2	RXMIS	RO	0	SSI 接收 FIFO 屏蔽后的中断状态 该位置位表示接收 FIFO 的状态为半空或多于半空。
1	RTMIS	RO	0	SSI 接收超时屏蔽后的中断状态 该位置位表示发生接收超时
0	RORMIS	RO	0	SSI 接收溢出屏蔽后的中断状态 该位置位表示接收 FIFO 溢出

寄存器 9：SSI 中断清零（SSIICR），偏移量 0x020

SSIICR 为中断清零寄存器。向该寄存器写 1 可将对应中断清零。写 0 无效。

SSI 中断清零 (SSIICR)

偏移量 0x020

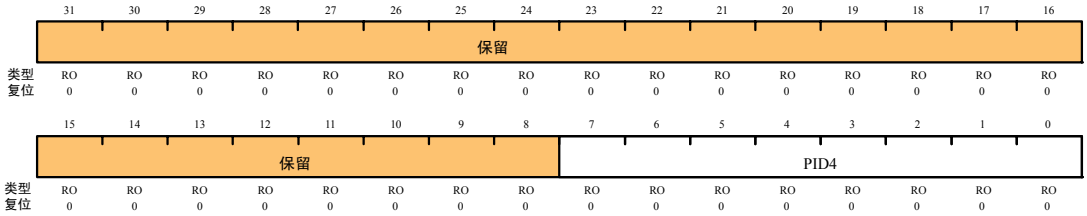


位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
1	RTIC	W1C	0	SSI 接收超时中断清零 0：对中断无影响 1：将中断清零
0	RORIC	W1C	0	SSI 接收溢出中断清零 0：对中断无影响 1：将中断清零

寄存器 10：SSI 外设标识 4（SSIPeriphID4），偏移量 0xFD0

SSIPeriphID4 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识4 (SSIPeriphID4)
偏移量 0xFD0

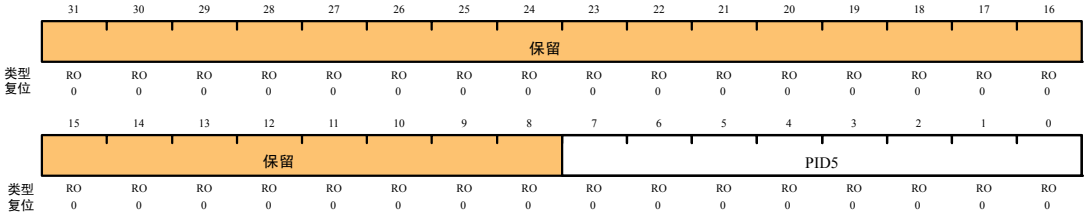


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID4	RO	0x00	SSI 外设 ID 寄存器[7:0]

寄存器 11：SSI 外设标识 5（SSIPeriphID5），偏移量 0xFD4

SSIPeriphID5 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 5 (SSIPeriphID5)
偏移量 0xFD4

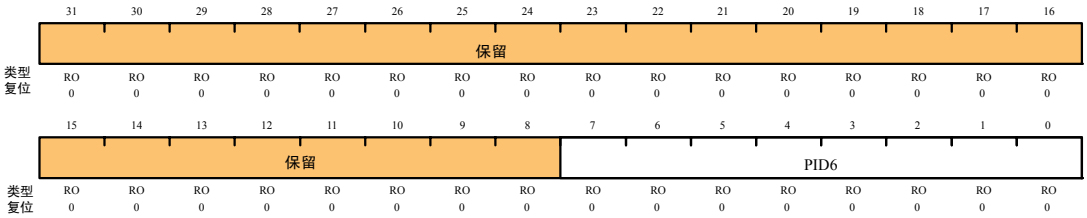


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID5	RO	0x00	SSI 外设 ID 寄存器[15:8]

寄存器 12：SSI 外设标识 6（SSIPeriphID6），偏移量 0xFD8

SSIPeriphID6 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 6 (SSIPeriphID6)
偏移量 0xFD8

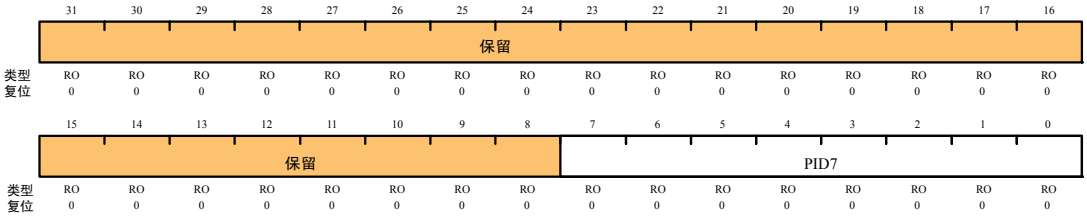


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID6	RO	0x00	SSI 外设 ID 寄存器[23:16]

寄存器 13：SSI 外设标识 7（SSIPeriphID7），偏移量 0xFDC

SSIPeriphID7 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 7 (SSIPeriphID7)
偏移量 0xFDC

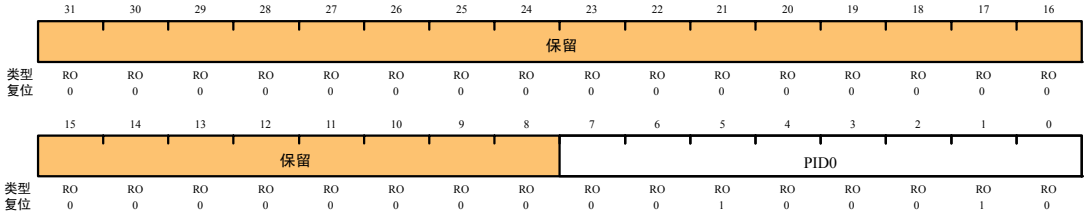


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID7	RO	0x00	SSI 外设 ID 寄存器[31:24]

寄存器 14：SSI 外设标识 0（SSIPeriphID0），偏移量 0xFE0

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 0 (SSIPeriphID0)
偏移量 0xFEO

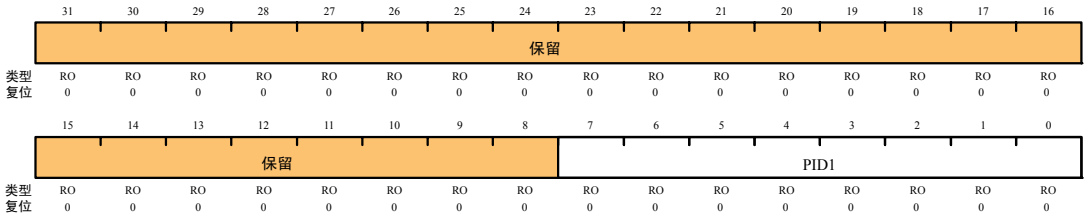


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID0	RO	0x22	SSI 外设 ID 寄存器[7:0] 软件可以使用该字段来判断这个外设是否存在。

寄存器 15：SSI 外设标识 1（SSIPeriphID1），偏移量 0xFE4

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 1 (SSIPeriphID1)
偏移量 0xFE4

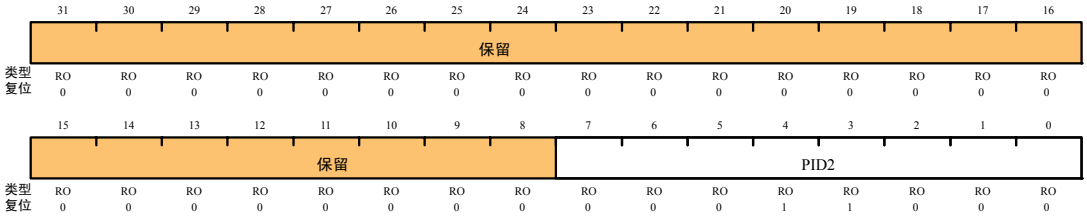


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID1	RO	0x00	SSI 外设 ID 寄存器[15:8] 软件可以使用该字段来判断这个外设是否存在。

寄存器 16：SSI 外设标识 2（SSIPeriphID2），偏移量 0xFE8

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识2 (SSIPeriphID2)
偏移量 0xFE8

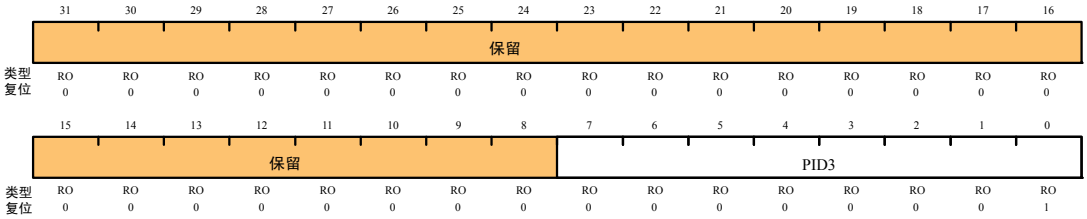


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID2	RO	0x18	SSI 外设 ID 寄存器[23:16] 软件可以使用该字段来判断这个外设是否存在。

寄存器 17：SSI 外设标识 3（SSIPeriphID3），偏移量 0xFEC

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI 外设标识 3 (SSIPeriphID3)
偏移量 0xFEC

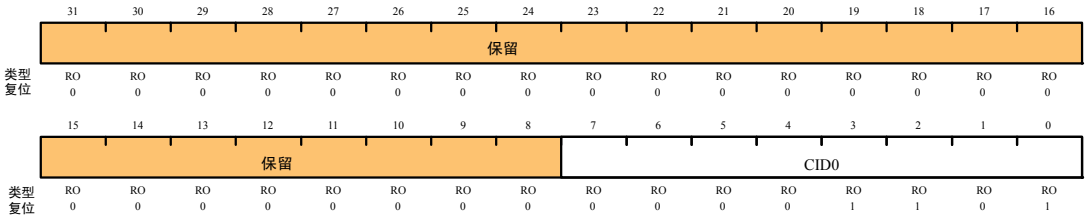


位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	PID3	RO	0x01	SSI 外设 ID 寄存器[31:24] 软件可以使用该字段来判断这个外设是否存在。

寄存器 18：SSI PrimeCell 标识 0（SSIPCellID0），偏移量 0xFF0

SSIPCellIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

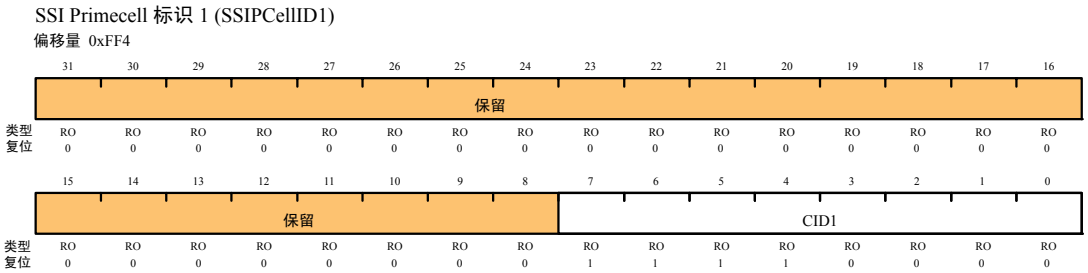
SSI PrimeCell标识 0 (SSIPCellID0)
偏移量 0xFF0



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	CID0	RO	0x0D	SSI PrimeCell ID 寄存器[7:0] 为软件提供一个标准的交叉外设识别系统

寄存器 19：SSI PrimeCell 标识 1（SSIPCellID1），偏移量 0xFF4

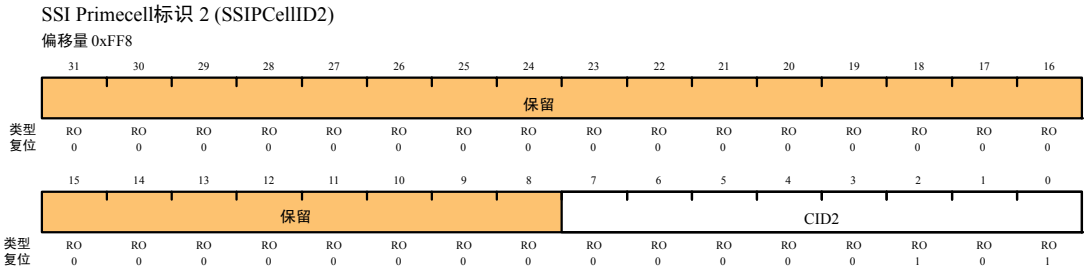
SSIPCellIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	CID1	RO	0xF0	SSI PrimeCell ID 寄存器[15:8] 为软件提供一个标准的交叉外设识别系统

寄存器 20：SSI PrimeCell 标识 2（SSIPCellID2），偏移量 0xFF8

SSIPCellIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。



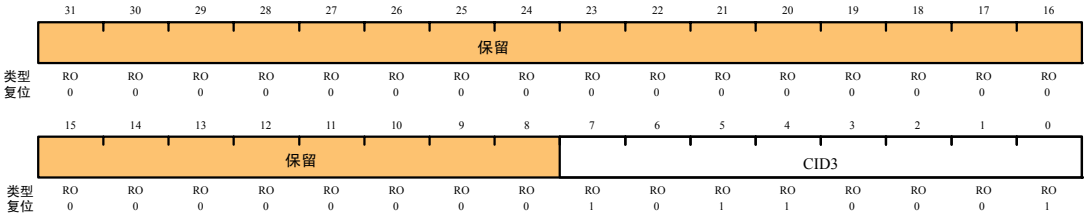
位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	CID2	RO	0x05	SSI PrimeCell ID 寄存器[23:16] 为软件提供一个标准的交叉外设识别系统

寄存器 21：SSI PrimeCell 标识 3（SSIPCellID3），偏移量 0xFFC

SSIPCellIDn 为硬编码的寄存器，该寄存器中的字段决定了复位值。

SSI Primecell 标识 3 (SSIPCellID3)

偏移量0xFFC



位/字段	名称	类型	复位	描述
31:8	保留	RO	0	保留位，返回不确定的值，并且不应该改变。
7:0	CID3	RO	0xB1	SSI PrimeCell ID 寄存器[31:24] 为软件提供一个标准的交叉外设识别系统

第14章 模拟比较器

模拟比较器是一种外设，它能够比较两个模拟电压的大小，并通过自身提供的逻辑输出端将比较结果以信号的形式输出。

LM3S617 控制器提供 1 个独立的集成模拟比较器，可配置模拟比较器来驱动输出、产生中断或 ADC 事件。

比较器可将测试电压与下面的其中一种电压相比较：

- 独立的外部参考电压
- 一个共用的外部参考电压
- 共用的内部参考电压

比较器可以向器件管脚提供输出，以替换板上的模拟比较器，或通过比较器中断或触发 ADC 来通知应用以使得它开始捕获一个采样序列。中断产生和 ADC 触发逻辑是相互独立的。这就意味着，例如，中断可以在上升沿产生而 ADC 在下降沿触发。

14.1 方框图

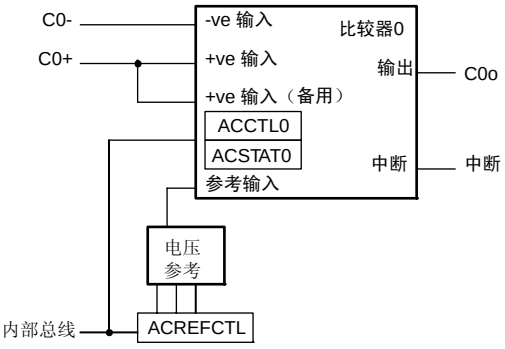


图 14.1 模拟比较器模块方框图

14.2 功能描述

要点：建议禁止模拟输入引脚的数字输入使能（GPIO 模块的 GPIODEN 位），以防止从 I/O 端口中汲取过量的电流。

比较器通过比较 VIN-和 VIN+输入来产生输出 VOUT。

如 图 14.2 所示，VIN-的输入源为一个外部输入电压。而VIN+的输入源除了外部输入电压之外，还可以是比较器 0 的+ve输入或内部参考电压。

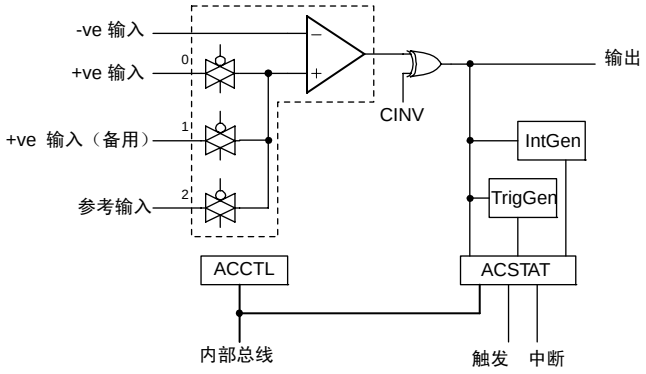


图 14.2 比较器单元的结构

比较器是通过两个状态/控制寄存器(**ACCTL**和**ACSTAT**)来配置的。而内部参考电压则是通过一个控制寄存器(**ACREFCTL**)来配置的。至于中断的状态和控制要通过三个寄存器(**ACMIS**、**ACRIS**和**ACINTEN**)来配置。比较器的操作模式请参考 表 14.1。

通常，会在内部使用比较器输出来产生控制器中断。但该输出也可以用来驱动外部管脚或产生一次模数转换器（ADC）的触发。

要点：在使用模拟比较器之前必须置位某些寄存器位。比较器输入和输出管脚的正确端口配置在 表 8.1 中描述。

表 14.1 比较器 0 操作模式

ACCNTL0	比较器 0			
ASRCP	**VIN-**	**VIN+**	**输出**	**中断**
00	C0-	C0+	C0o	是
01	C0-	C0+	C0o	是
10	C0-	Vref	C0o	是
11	C0-	保留	C0o	是

14.2.1 内部参考编程

内部参考的结构如 图 14.3 所示。该结构通过一个配置寄存器(**ACREFCTL**)来控制。而 表 14.2 中列出的是用于获得(develop)特定的内部参考值的编程选项，以便将外部电压与内部产生的特定电压进行比较。

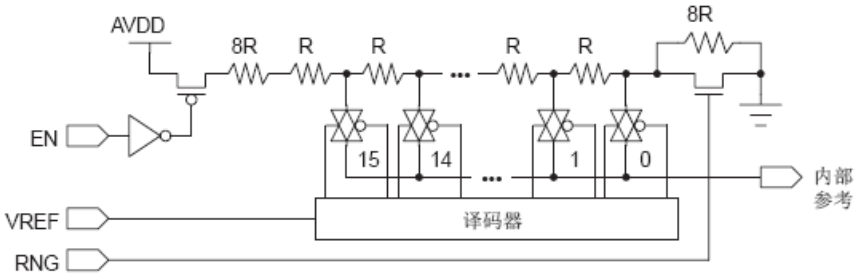


图 14.3 比较器内部参考的结构

表 14.2 内部参考电压和 ACREFCTL 字段值

ACREFCTL 寄存器		基于 VREF 字段值的输出参考电压
EN 位值	**RNG 位值**	
EN=0	RNG=X	无论 VREF 为任何值,输出参考电压都为 0；然而，建议使用 RNG=1 且 VREF=0 来获得最小噪声的参考地。
EN=1	RNG=0	阶梯电阻的总阻值为 32R。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ V_{REF} =0.825＋0.103・VREF 在该模式中内部参考电压的范围是 0.825－2.37V。
	RNG=1	阶梯电阻的总阻值为 24R。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ V_{REF} =0.1375・VREF 在该模式中内部参考电压的范围是 0.0－2.0625V。

14.3 初始化和配置

下面的例子展示了应如何配置模拟比较器才能从内部寄存器中读回其输出值。

1. 向系统控制模块中的 **RCGC1** 寄存器写入 0x00100000 的值来使能模拟比较器 0 的时钟。
2. 在 GPIO 模块中，使能与 C0- 相关的 GPIO 端口/管脚并将其用作 GPIO 输入。
3. 向 **ACREFCTL** 寄存器写入 0x0000030C 的值，从而将内部电压参考配置为 1.65V。
4. 向 **ACCTL0** 寄存器写入 0x0000040C 的值，从而将比较器 0 配置为使用内部电压作为参考、且不在 C0O 管脚上输出任何值的模式
5. 延时一段时间。
6. 读取 **ACSTAT0** 寄存器的 OVAL 值，便可获得比较器的输出值。

改变 C0-上输入信号的电平以观察 OVAL 值的变化。

14.4 寄存器映射

表 14.3 列出了比较器寄存器。所列的偏移量相对于 0x4003C000 的模拟比较器基址，在寄存器地址上采用十六进制递增的方式来列举。

表 14.3 模拟比较器寄存器映射

偏移量	名称	复位	类型	描述
0x00	ACMIS	0x00000000	RO	中断状态
0x04	ACRIS	0x00000000	RO	原始中断状态
0x08	ACINTEN	0x00000000	R/W	中断使能
0x10	ACREFCTL	0x00000000	R/W	参考电压控制
0x20	ACSTAT0	0x00000000	RO	比较器 0 状态
0x24	ACCTL0	0x00000000	R/W	比较器 0 控制

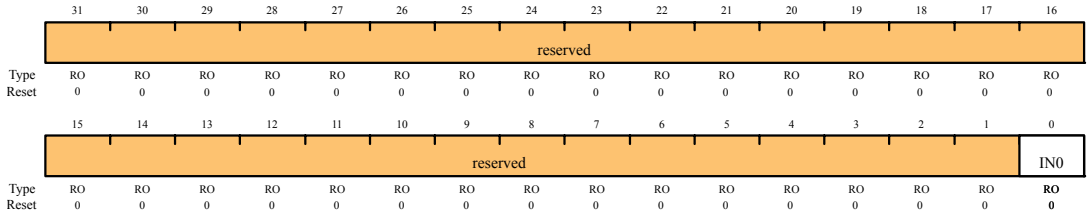
14.5 寄存器描述

本节内容将按地址偏移量的数字顺序来列举并且描述模拟比较器寄存器。

寄存器 1：模拟比较器屏蔽后的中断状态(ACMIS)，偏移量 0x00

该寄存器汇总了比较器（经过屏蔽后的）中断状态。

模拟比较器屏蔽后的中断状态 (ACMIS)
Offset 0x000

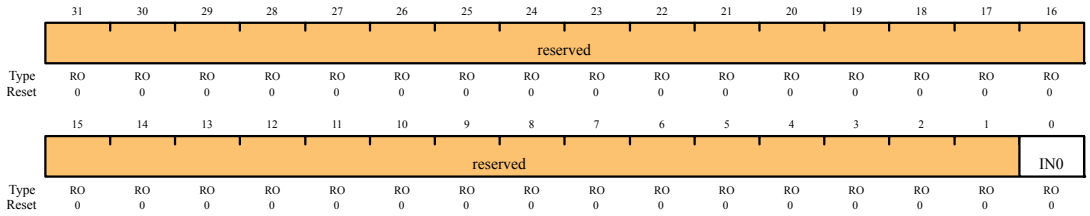


位/字段	名称	类型	复位	描述
31: 1	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
0	IN0	RO	0	比较器 0 屏蔽后的中断状态。 给出该中断屏蔽后的中断状态。

寄存器 2：模拟比较器原始中断状态(ACRIS)，偏移量 0x04

该寄存器汇总了比较器的（原始）中断状态。

模拟比较器原始中断状态 (ACRIS)
Offset 0x04

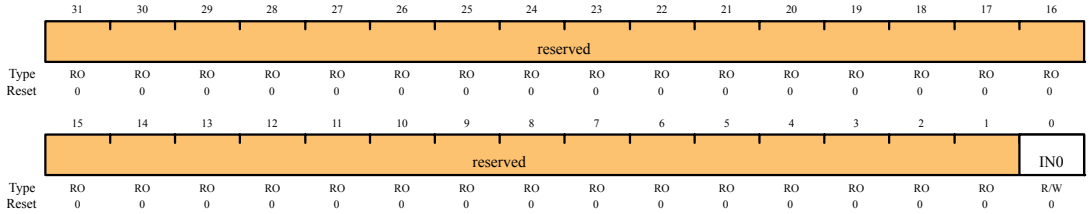


位/字段	名称	类型	复位	描述
31: 1	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
0	IN0	RO	0	当该位置位时，表示中断已通过比较器 0 产生。

寄存器 3：模拟比较器中断使能(ACINTEN)，偏移量 0x08

该寄存器为比较器提供中断使能。

模拟比较器中断使能 (ACINTEN)
Offset 0x08

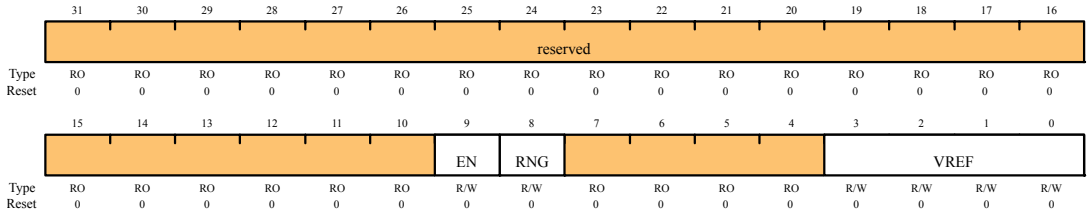


位/字段	名称	类型	复位	描述
31: 1	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
0	IN0	R/W	0	当该位置位时，使能比较器 0 输出的控制器中断。

寄存器 4：模拟比较器参考电压控制(ACREFCTL)，偏移量 0x10

该寄存器指示阶梯电阻 (resistor ladder)是否已上电，该电阻的范围和抽头(tap)。

模拟比较器参考电压控制 (ACREFCTL)
偏移量 0x010



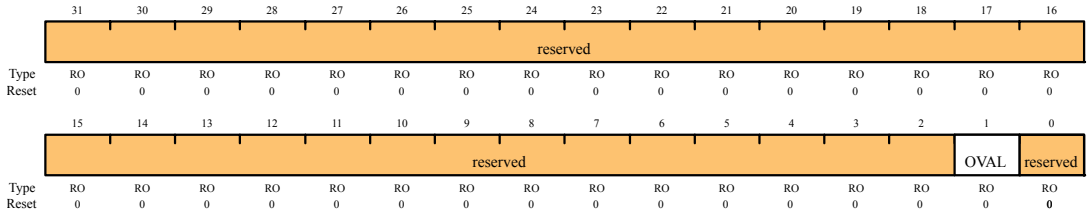
位/字段	名称	类型	复位	描述
31:10	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
9	EN	R/W	0	EN 位指示阶梯电阻(resistor ladder)是否已上电。如果该位为 0，则阶梯电阻未上电。如果该位为 1，则阶梯电阻被连接到模拟 V_{DD}。 该位复位为 0 使得在未使用和未编程的情况下内部参考消耗的功率总量最小。
8	RNG	R/W	0	RNG 位指示阶梯电阻的范围。如果该位为 0，则阶梯电阻的总电阻为 32R。如果该位为 1，则阶梯电阻的总电阻为 24R。
7:4	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
3:0	VREF	R/W	0	VREF字段指示的是通过模拟复用器的阶梯电阻的抽头。每个抽头(tap)所对应的电压是可用于比较的内部参考电压。有关输出参考电压的一些例子请见 表 14.2。

寄存器 5：模拟比较器状态 0 (ACSTAT0)，偏移量 0x20

这个寄存器能指示出各自对应的比较器的当前输出值。

模拟比较器状态 0 (ACSTAT0)

偏移量 0x020



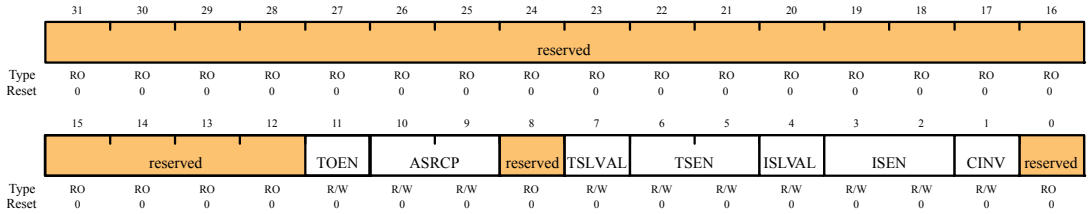
位/字段	名称	类型	复位	描述
31:2	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
1	OVAL	RO	0	OVAL 位能指示出比较器的当前输出值。
0	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。

寄存器 6：模拟比较器控制 0 (ACCTL0)，偏移量 0x24

这个寄存器能配置各自对应的比较器的输入和输出。

模拟比较器控制0（ACCTL0）

Offset 0x024



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
11	TOEN	R/W	0	TOEN 位使 ADC 事件传送到 ADC。如果该位为 0，则事件被删除，不发送到 ADC。如果该位为 1，则事件被传送到 ADC。
10:9	ASRCP	R/W	0	ASRCP 字段指定了到比较器 VIN+端口的输入电压源。该字段的编码如下： **ASRCP 功能** 00 管脚值 01 C0+的管脚值 10 内部电压参考 11 保留
8:5	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。
7	TSLVAL	R/W	0	TSLVAL 位指定电平检测模式下产生 ADC 事件的输入的感应值。如果该位为 0，则比较器输出为低时产生 ADC 事件。否则，ADC 事件在比较器输出为高时产生。

续上表

位/字段	名称	类型	复位	描述
6:5	TSEN	R/W	0	TSEN 字段指定产生 ADC 事件的比较器输出的检测方式。检测条件如下： TSEN 功能 00 检测电平，见“TSLVAL”。 01 下降沿 10 上升沿 11 上升沿/下降沿
4	ISLVAL	R/W	0	ISLVAL 位指定了在电平检测模式中能够使中断产生的输入电平的感应值。如果该位为 0，则比较器输出为低时产生中断。否则，在比较器输出为高时产生中断。
3:2	ISEN	R/W	0	ISEN 字段指定了产生中断的比较器输出的检测方式。检测条件如下： **ISEN 功能** 00 电平检测，见 ISLVAL 01 下降沿 10 上升沿 11 上升沿或下降沿
1	CINV	R/W	0	CINV 位有条件地翻转(invert)比较器的输出。如果该位为 0，那么比较器的输出不变。但如果该位为 1，那么在交由硬件处理之前会将比较器的输出电平翻转。
0	保留	RO	0	保留位返回一个不确定的值，并且应永不改变。

第15章 脉宽调制器（PWM）

脉宽调制（PWM）是对模拟信号电平进行数字化编码的一项功能强大的技术。在脉宽调制中使用高分辨率计数器来产生方波，并调整方波的占空比以对模拟信号进行编码。PWM 的典型应用包括开关电源和电机控制。

LM3S617 的 PWM 模块由 3 个 PWM 发生器模块和一个控制模块组成。每个 PWM 发生器模块包含一个定时器（16 位递减或先增后减计数器）、两个比较器、一个 PWM 信号发生器、一个死区发生器、和一个中断/ADC-触发选择器。控制模块决定了 PWM 信号的极性，以及将哪个信号传递到管脚。

每个 PWM 发生器模块产生两个 PWM 信号，这两个信号可以是独立的（不同于基于相同定时器，具有相同频率的信号），或一对插入了死区延迟的互补信号。PWM 发生器模块的输出在传递到器件管脚之前由输出控制模块管理。

LM3S617 的 PWM 模块具有极大的灵活性。它可以产生简单的 PWM 信号，例如供简单的充电泵使用的信号；也可以产生带死区延迟的成对 PWM 信号，例如供半 H 桥驱动器使用的信号。它也可以产生 3 相反相器桥需要的全部 6 路门控制信号。

15.1 框图

图 15.1 提供了一个Stellaris PWM模块的框图。LM3S617 控制器包含 3 个发生器模块（PWM0、PWM1 和PWM2），它产生 6 个独立的PWM信号，或 3 对带死区延迟的PWM信号。

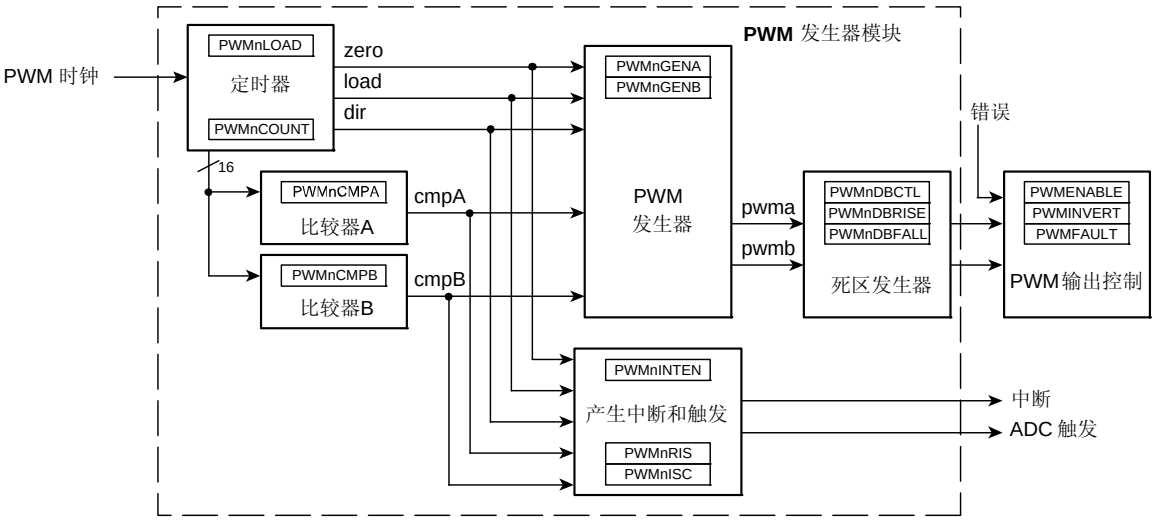


图 15.1 PWM 模块框图

15.2 功能描述

15.2.1 PWM 定时器

每个 PWM 发生器中的定时器都有两种工作模式：递减计数模式或先增后减计数模式。在递减计数模式中，定时器从装载值开始计数，在到达零时返回到装载值并继续递减计数。在先增后减计数模式中，定时器从 0 开始向上计数到装载值，再从装载值递减到零，然后再递增到装载值，以此类推。通常，递减计数模式用于产生左对齐或右对齐的 PWM 信号，而先增后减计数模式则用于产生中心对齐的 PWM 信号。

定时器会输出 3 个在产生 PWM 信号的过程中要用到的信号，它们分别为：方向信号、零脉冲信号和装载脉冲信号。方向信号在递减模式中始终保持低电平，但在先增后减模式中则在低电平和高电平之间轮流转换；零脉冲信号在计数器变为 0 时发出一个宽度为单个时钟周期的高电平脉冲；而装载脉冲会在计数值等于装载值时发出一个宽度为单个时钟周期的高电平脉冲。需要注意的是：在递减计数模式中，在零脉冲信号之后会紧跟着一个装载脉冲信号。

15.2.2 PWM 比较器

每个 PWM 发生器中有两个比较器，用来监控计数器的值。当比较器的值与计数器的值相等时，比较器会输出一个宽度为单个时钟周期的高电平脉冲。在先增后减计数模式中，比较器在递增和递减计数时都要进行比较，因此，它们需由计数器的方向信号来限定。方向信号的限定脉冲可在产生 PWM 信号的过程中使用。如果比较器的匹配值大于计数器的装载值，则该比较器永远不输出高电平脉冲。

图 15.2 显示了计数器处于递减计数模式时的行为以及这些脉冲的关系。图 15.3 显示了计数器处于先增后减计数模式时的行为以及这些脉冲的关系。

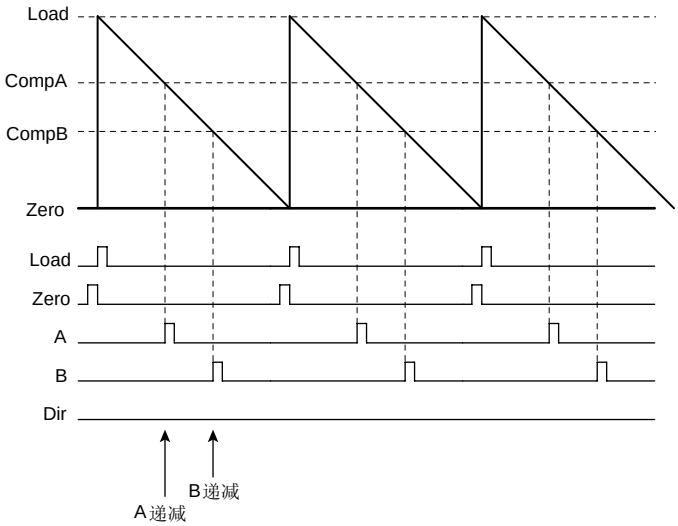


图 15.2 PWM 递减计数模式

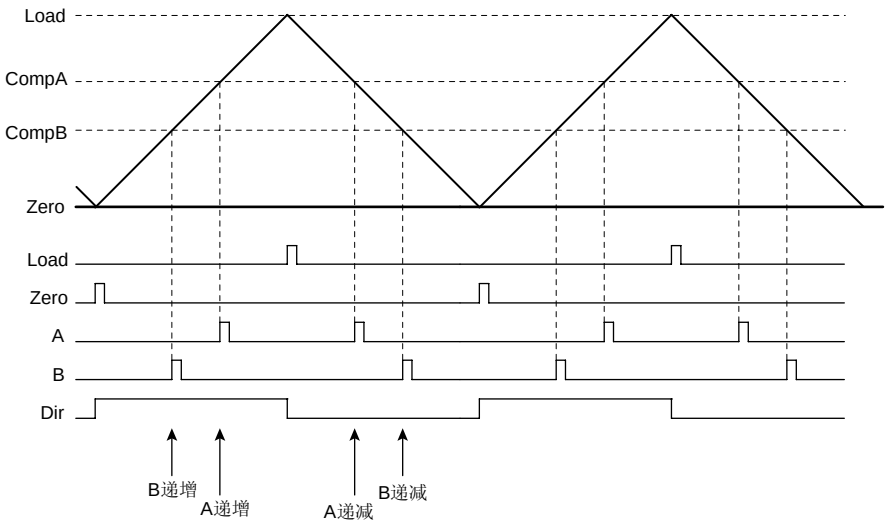


图 15.3 PWM 先增后减计数模式

15.2.3 PWM 信号发生器

PWM 发生器获得这些脉冲（由方向信号来限定），并产生两个 PWM 信号。在递减计数模式中，能够影响 PWM 信号的事件有 4 个：归零、装载、匹配 A 递减、匹配 B 递减。在先增后减计数模式中，能够影响 PWM 信号的事件有 6 个：归零、装载、匹配 A 递减、匹配 A 递增、匹配 B 递减、匹配 B 递增。当匹配 A 和匹配 B 事件与零或装载事件相符时将被忽略。如果匹配 A 和匹配 B 相同，则第一个信号 PWMA 只根据匹配 A 事件产生。第二个信号 PWMB，只根据匹配 B 事件产生。

各个事件在PWM输出信号上的影响都是可编程的：可以保留（忽略该事件），可以翻转，可以驱动为低电平、或驱动为高电平。这些动作可用来在不同位置产生具有不同占空比的一对PWM信号，并且这对信号可以重叠或不重叠。图 15.4 显示了使用先增后减计数模式产生的一对中心对齐的、具有不同占空比的重叠PWM信号。

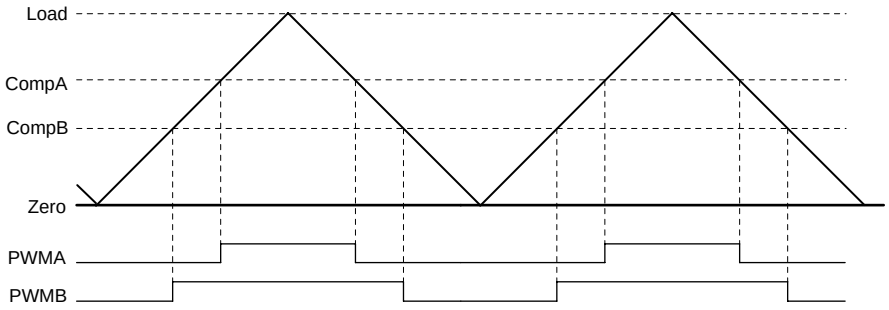


图 15.4 在先增后减计数模式中产生 PWM 信号

在该例中，第一次将 PWM 发生器设置为在出现匹配 A 递增事件时驱动为高电平，出现匹配 A 递减事件时驱动为低电平，并忽略其它 4 个事件。第二次将发生器设置为在出现匹配 B 递增事件时驱动为高电平，出现匹配 B 递减事件时驱动为低电平，并忽略其它 4 个事件。改变比较器 A 的值可改变 PWMA 信号的占空比，改变比较器 B 的值可改变 PWMB 信号的占空比。

15.2.4 死区发生器

PWM 发生器产生的两个 PWM 信号会传递到死区发生器。如果死区功能被禁止，则 PWM 信号只简单地通过该模块，并不进行修改。如果死区功能被使能，则丢弃第二个 PWM 信号，并在第一个 PWM 信号基础上产生两个 PWM 信号。第一个输出 PWM 信号为带上升沿延迟的输入信号，延迟时间可编程。第二个输出 PWM 信号为输入信号的反相信号，在输入信号的下降沿和这个新信号的上升沿之间增加了可编程的延迟时间。

因此，这是一对高电平有效的信号，其中总有一路为高电平，但进行切换时的可编程时间量除外，在该段时间内，两个信号都为低电平。因此，这两个信号适合于驱动半H桥，它们所带有的死区延迟可防止短路电流破坏功率型电子器件（power electronics）。图 15.5 显示了死区发生器对输入PWM信号的影响。

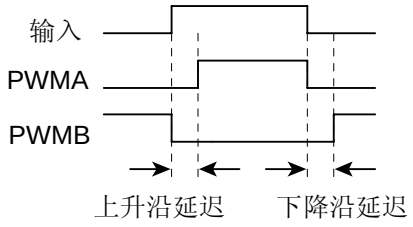


图 15.5 PWM 死区发生器

15.2.5 中断/ADC-触发选择器

PWM 发生器还获取相同的 4 个（或 6 个）计数器事件，并使用它们来产生中断或 ADC 触发。这些事件中的任何一个或一组都可以被选择为中断源，当所选的事件中的任一个发生时，中断便会产生。

另外，也可选择与中断源相同的一个事件、不同的一个事件、或者是相同的一组事件、不同的一组事件作为 ADC 触发源。当所选事件中的任一个发生时，产生 ADC 触发脉冲。通过对事件进行选择，可以使中断或者 ADC 触发脉冲在 PWM 信号中的特定位置产生。注：中断和 ADC 触发都是基于原始事件的，不考虑死区发生器在 PWM 信号边沿上产生的延迟。

15.2.6 同步方法

PWM 模块有一个全局复位功能，能够将 PWM 发生器中的任一个或所有计数器进行同步复位。如果将多个 PWM 发生器配置为使用相同的计数器装载值，则使用全局复位能够保证这些计数器也具有相同的计数值（这表示 PWM 发生器必须在同步前进行配置）。这样，在产生两个以上的 PWM 信号时，这些信号边沿之间的关系是已知的，因为计数器始终具有相同的值。

PWM 发生器的计数器装载值和比较器匹配值可采用两种方法进行更新。第一种是立即更新模式，在此方法中，一旦计数器计数到零就立即使用新值。因为在更新时要等待计数器计数到零，所以更新便是一个经过定义而且确定会发生的行为，这样就可以避免出现过短或过长的 PWM 输出脉冲。

另一种方法是同步更新。在此方法中，直到全局同步更新信号生效时才使用新值；此时，一旦计数器到达零就会立即使用新值。这种方法允许将多个 PWM 发生器中的多个项目同步更新，而不会在更新过程中出现意外的影响。所有逻辑在根据新值运行之前都在原来的值上运行。装载值和比较器匹配值的更新模式可以在每个 PWM 发生器模块中单独进行配置。如果 PWM 发生器模块中的定时器被同步，则在跨接这些模块时只能使用同步更

新机制才有意义，虽然这对于这种机制能够正常工作来说并不是必需的。

15.2.7 故障状态

影响 PWM 模块的外部状态有两个：故障管脚的信号输入以及由调试器引起的控制器停止。这两种状态可采用两种机制来处理：强制输出信号进入停止状态和/或停止 PWM 定时器。

每个输出信号都有一个故障位。如果将故障位置位，则故障输入信号将使对应的输出信号进入停止状态。如果停止状态是信号能够长期停留的安全状态，则这样做可避免输出信号在故障状态下以危险的方式驱动外部电路。故障状态也可产生一次控制器中断。

每个 PWM 发生器还可配置为在暂停状态中停止计数。用户可选择让计数器一直运行直到到达零时停止，或到达零时继续计数和重装。暂停状态不产生控制器中断。

15.2.8 输出控制模块

每个 PWM 发生器模块产生两个原始 PWM 信号，输出控制模块会在 PWM 信号到达管脚之前对其最后的状态进行控制。而且仅对一个寄存器进行操作，便能够对实际传递到管脚的 PWM 信号作出修改。例如，只需对寄存器进行一次写操作，便能修改 PWM 信号来完成无刷直流电机的换相（而无需通过修改反馈控制回路来修改各个 PWM 发生器）。同样，故障控制也能够禁止任意一路 PWM 信号。最后，还能够对任意一路 PWM 信号执行反相操作，使得默认高电平有效的信号变为低电平有效。

15.3 初始化和配置

下面是对 PWM 发生器 0 进行初始化的示例。要求频率为 25KHz，PWM0 管脚上的占空比为 25%，PWM1 管脚上的占空比为 75%。该例假定系统时钟为 20MHz。

1. 向系统控制模块中的 **RCG0** 寄存器写入 0x00100000 来使能 PWM 时钟。
2. 在 GPIO 模块中，使用 **GPIOAFSEL** 寄存器来使能适当的管脚的备用功能。
3. 将系统控制模块中的**运行-模式时钟配置（RCC）**寄存器配置为使用 PWM 分频（USEPWMDIV）并将分频器（PWMDIV）设置为 2 分频（000）。
4. 将 PWM 发生器配置为递减计数模式，并立即更新参数。
   - 向 **PWM0CTL** 寄存器写入 0x00000000。
   - 向 **PWM0GENA** 寄存器写入 0x0000008C
   - 向 **PWM0GENB** 寄存器写入 0x0000080C
5. 设置周期。对于 25KHz 频率，周期为 1/25,000，即 40μs。PWM 时钟源为 10MHz，通过对系统时钟进行 2 分频获得。要从 10MHz 的 PWM 时钟源中获得 25KHz 频率，这意味着一个周期为 400 个 PWM 时钟周期。然后便可以使用这个值来设置 **PWM0LOAD** 寄存器。在递减计数模式中，须将 **PWM0LOAD** 寄存器的 LOAD 字段设置为要求的周期数减 1。
   - 向 **PWMOLOAD** 寄存器写入 0x0000018F。
6. 将 PWM0 管脚的脉冲宽度设置为 25%占空比
   - 向 **PWM0CMPA** 寄存器写入 0x0000012B
7. 将 PWM1 管脚的脉冲宽度设置为 75%占空比

— 向 **PWM0CMPB** 寄存器写入 0x00000063

8. 启动 PWM 发生器 0 中的定时器

— 向 PWM0CTL 寄存器写入 0x00000001

9. 使能 PWM 输出

— 向 **PWMENABLE** 寄存器写入 0x00000003

15.4 寄存器映射

表 15.1 列出了PWM的寄存器。寄存器地址相对于PWM基址 0x4002800 的偏移量并以十六进制递增的方式列出。

表 15.1 PWM 寄存器映射

偏移量	名称	复位	类型	描述
PWM 模块控制				
0x000	PWMCTL	0x00000000	R/W	PWM 模块的主控制
0x004	PWMSYNC	0x00000000	R/W	PWM 发生器的计数器同步
0x008	PWMENABLE	0x00000000	R/W	PWM 输出管脚的主机使能
0x00C	PWMINVERT	0x00000000	R/W	PWM 输出管脚的反相控制
0x010	PWMFAULT	0x00000000	R/W	PWM 输出管脚的故障处理
0x014	PWMINTEN	0x00000000	R/W	中断使能
0x018	PWMRIS	0x00000000	RO	原始中断状态
0x01C	PWMISC	0x00000000	R/W1C	中断状态和清零
0x020	PWMSTATUS	0x00000000	RO	故障输入信号的值
PWM 发生器 0				
0x040	PWM0CTL	0x00000000	R/W	PWM0 发生器模块的主控制
0x044	PWM0INTEN	0x00000000	R/W	中断和触发使能
0x048	PWM0RIS	0x00000000	RO	原始中断状态
0x04C	PWM0ISC	0x00000000	R/W1C	中断状态和清零
0x050	PWM0LOAD	0x00000000	R/W	计数器的装载值
0x054	PWM0COUNT	0x00000000	RO	计数器的当前值
0x058	PWM0CMPA	0x00000000	R/W	比较器 A 的值
0x05C	PWM0CMPB	0x00000000	R/W	比较器 B 的值
0x060	PWM0GENA	0x00000000	R/W	控制 PWM 发生器 A
0x064	PWM0GENB	0x00000000	R/W	控制 PWM 发生器 B
0x068	PWM0DBCTL	0x00000000	R/W	控制死区发生器
0x06C	PWM0DBRISE	0x00000000	R/W	死区上升沿延迟计数
0x070	PWM0DBFALL	0x00000000	R/W	死区下降沿延迟计数
PWM 发生器 1				
0x080	PWM1CTL	0x00000000	R/W	PWM1 发生器模块的主控制
0x084	PWM1INTEN	0x00000000	R/W	中断和触发使能
0x088	PWM1RIS	0x00000000	RO	原始中断状态
0x08C	PWM1ISC	0x00000000	R/W1C	中断状态和清零

续上表

偏移量	名称	复位	类型	描述
PWM 发生器 1				
0x090	PWM1LOAD	0x00000000	R/W	计数器的装载值
0x094	PWM1COUNT	0x00000000	RO	计数器的当前值
0x098	PWM1CMPA	0x00000000	R/W	比较器 A 的值
0x09C	PWM1CMPB	0x00000000	R/W	比较器 B 的值
0x0A0	PWM1GENA	0x00000000	R/W	控制 PWM 发生器 A
0x0A4	PWM1GENB	0x00000000	R/W	控制 PWM 发生器 B
0x0A8	PWM1DBCTL	0x00000000	R/W	控制死区发生器
0x0AC	PWM1DBRISE	0x00000000	R/W	死区上升沿延迟计数
0x0B0	PWM1DBFALL	0x00000000	R/W	死区下降沿延迟计数
PWM 发生器 2				
0x0C0	PWM2CTL	0x00000000	R/W	PWM2 发生器模块的主控制
0x0C4	PWM2INTEN	0x00000000	R/W	中断和触发使能
0x0C8	PWM2RIS	0x00000000	RO	原始中断状态
0x0CC	PWM2ISC	0x00000000	R/W1C	中断状态和清零
0x0D0	PWM2LOAD	0x00000000	R/W	计数器的装载值
0x0D4	PWM2COUNT	0x00000000	RO	计数器的当前值
0x0D8	PWM2CMPA	0x00000000	R/W	比较器 A 的值
0x0DC	PWM2CMPB	0x00000000	R/W	比较器 B 的值
0x0E0	PWM2GENA	0x00000000	R/W	控制 PWM 发生器 A
0x0E4	PWM2GENB	0x00000000	R/W	控制 PWM 发生器 B
0x0E8	PWM2DBCTL	0x00000000	R/W	控制死区发生器
0x0EC	PWM2DBRISE	0x00000000	R/W	死区上升沿延迟计数
0x0F0	PWM2DBFALL	0x00000000	R/W	死区下降沿延迟计数

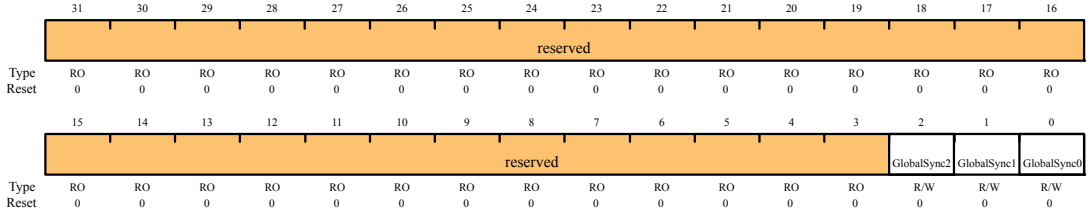
15.5 寄存器描述

下面将按地址偏移量的数字顺序列出了 PWM 的寄存器并对其进行描述。

寄存器 1：PWM 主控制（PWMCTL），偏移量 0x000

该寄存器为 PWM 发生模块提供主控制。

PWM 主控制 (PWMCTL)
Offset 0x000

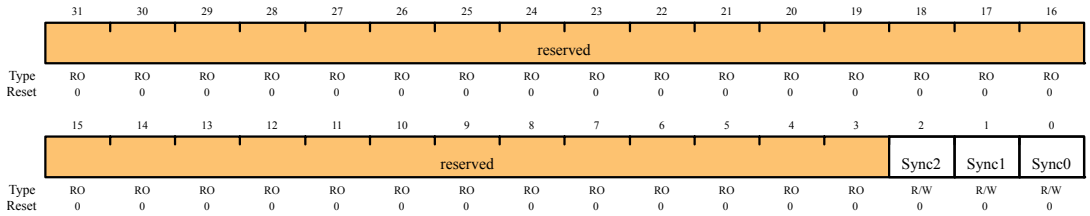


位/字段	名称	类型	复位	描述
31:3	保留	RO	0	保留位，返回不确定的值，并且不应该改变
2	GlobalSync2	R/W	0	与 GlobalSync0 相同，但针对的是 PWM 发生器 2
1	GlobalSync1	R/W	0	与 GlobalSync0 相同，但针对的是 PWM 发生器 1
0	GlobalSync0	R/W	0	将该位置位会使 PWM 发生器 0 中的加载或比较寄存器在对应的计数器下一次变为 0 时进行更新。当更新完成时该位自动清零，它不能由软件来清零。

寄存器 2：PWM 时基同步（PWMSYNC），偏移量 0x004

该寄存器提供了一种让 PWM 发生模块中的计数器同步化的方法。向该寄存器的非保留位写入 1 可使指定的计数器复位为零，如果 PWM 模块的数目超过 1 个的话，那么向多个非保留位写入 1 会使对应的多个计数器同时复位。复位之后，寄存器中的位将自动清零，若读取这些位所返回的结果为零，则表示同步已完成。

PWM 时基同步(PWMSYNC)
Offset 0x004

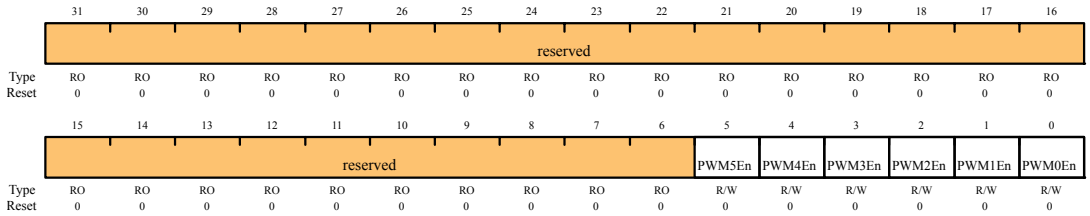


位/字段	名称	类型	复位	描述
31:3	保留	RO	0	保留位，返回不确定的值，并且不应该改变
2	Sync2	R/W	0	对 PWM 发生器 2 的计数器进行复位
1	Sync1	R/W	0	对 PWM 发生器 1 的计数器进行复位
0	Sync0	R/W	0	对 PWM 发生器 0 的计数器进行复位

寄存器 3：PWM 输出使能（PWMENABLE），偏移量 0x008

该寄存器可以控制是否将某个已产生的 PWM 信号输出到器件管脚。禁止 PWM 输出后，PWM 信号的产生过程可在不将 PWM 信号传递到管脚的情况下继续进行（例如，当时基同步时）。当寄存器中的位被置位时，可将对应的 PWM 信号传递到由 **PWMINVERT** 寄存器控制的输出级。没有置位时，PWM 信号会被 0 代替，也传递到输出级。

PWM 输出使能 (PWMENABLE)
Offset 0x008



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	PWM5En	R/W	0	该位置位时，允许所产生的 PWM5 信号传递到器件管脚上
4	PWM4En	R/W	0	该位置位时，允许所产生的 PWM4 信号传递到器件管脚上

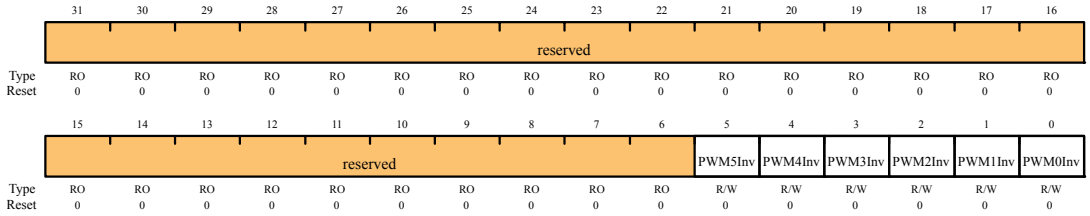
续上表

位/字段	名称	类型	复位	描述
3	PWM3En	R/W	0	该位置位时，允许所产生的 PWM3 信号传递到器件管脚上
2	PWM2En	R/W	0	该位置位时，允许所产生的 PWM2 信号传递到器件管脚上
1	PWM1En	R/W	0	该位置位时，允许所产生的 PWM1 信号传递到器件管脚上
0	PWM0En	R/W	0	该位置位时，允许所产生的 PWM0 信号传递到器件管脚上

寄存器 4：PWM 输出反相（PWMINVERT），偏移量 0x00C

该寄存器可以控制器件管脚上的 PWM 信号的极性。由死区模块产生的 PWM 信号为高电平有效，可选择通过该寄存器来变为低电平有效。被禁止的 PWM 通道的输出也会通过输出反相器（假如这样配置的话），这样，不工作的通道也能保持准确的极性。

PWM 输出反相 (PWMINVERT)
Offset 0x00C



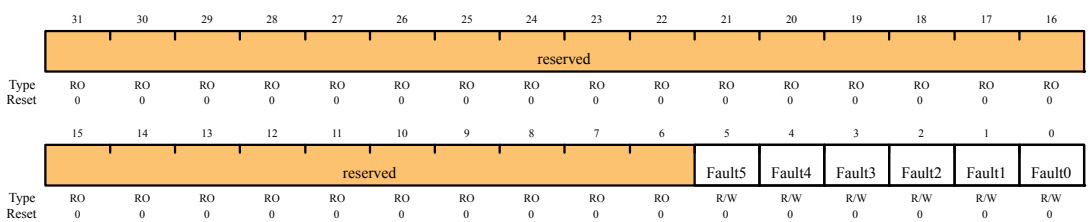
位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	PWM5Inv	R/W	0	该位置位时，将已生成的 PWM5 信号反相
4	PWM4Inv	R/W	0	该位置位时，将已生成的 PWM4 信号反相
3	PWM3Inv	R/W	0	该位置位时，将已生成的 PWM3 信号反相
2	PWM2Inv	R/W	0	该位置位时，将已生成的 PWM2 信号反相
1	PWM1Inv	R/W	0	该位置位时，将已生成的 PWM1 信号反相
0	PWM0Inv	R/W	0	该位置位时，将已生成的 PWM0 信号反相

寄存器 5：PWM 输出故障（PWMFAULT），偏移量 0x10

该寄存器用来在发生故障状态时，控制 PWM 输出的行为。故障输入和调试事件都看作是故障状态。在故障状态下，每个 PWM 信号可以采用的方式是直接通过，不进行修改，又或者是被驱动为低电平。对于配置为通过式的输出，对应 PWM 发生器所处理的调试事件还决定了是否继续产生 PWM 信号。

故障状态控制在输出反相器之前进行，因此，如果将通道配置为反相，则在故障状态时驱动为低电平的 PWM 信号将被反相（即，该管脚在存在故障状态时驱动为高电平）。

PWM 输出故障 (PWMFAULT)
Offset 0x010

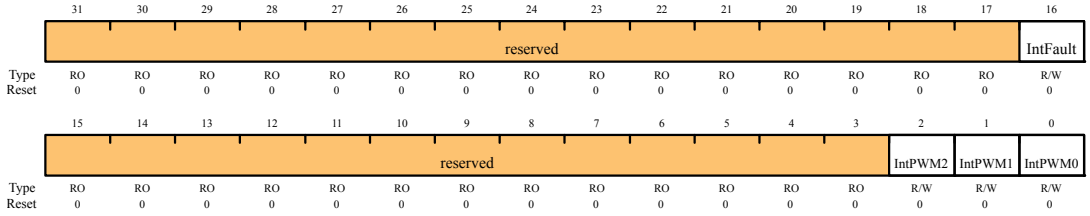


位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	Fault5	R/W	0	该位置位时，输出信号 PWM5 在故障状态下驱动为低电平
4	Fault4	R/W	0	该位置位时，输出信号 PWM4 在故障状态下驱动为低电平
3	Fault3	R/W	0	该位置位时，输出信号 PWM3 在故障状态下驱动为低电平
2	Fault2	R/W	0	该位置位时，输出信号 PWM2 在故障状态下驱动为低电平
1	Fault1	R/W	0	该位置位时，输出信号 PWM1 在故障状态下驱动为低电平
0	Fault0	R/W	0	该位置位时，输出信号 PWM0 在故障状态下驱动为低电平

寄存器 6：PWM 中断使能（PWMINTEN），偏移量 0x014

该寄存器控制 PWM 模块的全局中断产生功能。能够引起中断的事件包括故障输入和来自 PWM 发生器的各个中断。

PWM 中断使能 (PWMINTEN)
Offset 0x014

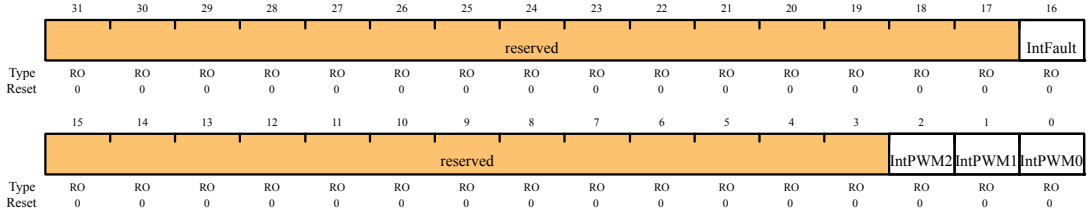


位/字段	名称	类型	复位	描述
31:17	保留	RO	0	保留位，返回不确定的值，并且不应该改变
16	IntFault	R/W	0	如果该位为 1，则在在出现故障输入时中断产生。
15:3	保留	RO	0	保留位，返回不确定的值，并且不应该改变
2	IntPWM2	R/W	0	如果该位为 1，则在 PWM 发生器 2 模块发出中断时中断产生。
1	IntPWM1	R/W	0	如果该位为 1，则在 PWM 发生器 1 模块发出中断时中断产生。
0	IntPWM0	R/W	0	如果该位为 1，则在 PWM 发生器 0 模块发出中断时中断产生。

寄存器 7：PWM 原始中断状态（PWMRIS），偏移量 0x018

该寄存器提供已发出的中断源的当前设置状态，而不管它们是否会引起一次有效的控制器中断。故障中断在检测时锁存，它必须通过 **PWM 中断状态和清零（PWMISC）**寄存器来清零。PWM 发生器中断只简单地反映 PWM 发生器的状态，它们是通过 PWM 发生器模块中的中断状态寄存器来清零的。寄存器中被设为 1 的位表示活动的事件，0 位表示正被查询的事件处于停止状态。

PWM 原始中断状态 (PWMRIS)
Offset 0x018

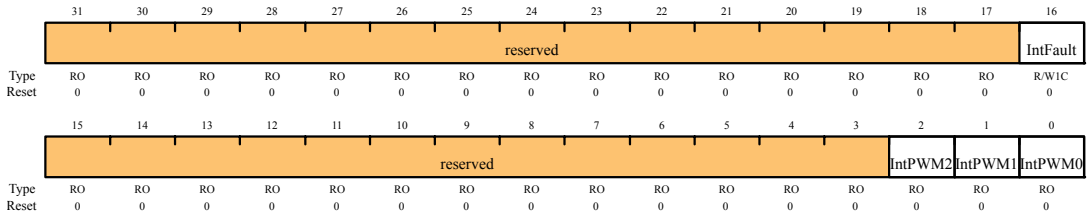


位/字段	名称	类型	复位	描述
31:17	保留	RO	0	保留位，返回不确定的值，并且不应该改变
16	IntFault	R/W	0	表示已发出故障输入
15: 3	保留	RO	0	保留位，返回不确定的值，并且不应该改变
2	IntPWM2	RO	0	表示 PWM 发生器 2 模块发出中断
1	IntPWM1	RO	0	表示 PWM 发生器 1 模块发出中断
0	IntPWM0	RO	0	表示 PWM 发生器 0 模块发出中断

寄存器 8：PWM 中断状态和清零（PWMISC），偏移量 0x01C

该寄存器汇总了 PWM 发生器模块的中断状态。寄存器中的位为 1 表示发生器模块的中断正有效。至于中断产生的原因，必须通过查询各个中断状态寄存器才能确定，而且通过使用这些寄存器可将中断清零。对于故障中断，向该位写 1 即可将锁存的中断状态清零。

PWM 中断状态和清零(PWMISC)
Offset 0x01C

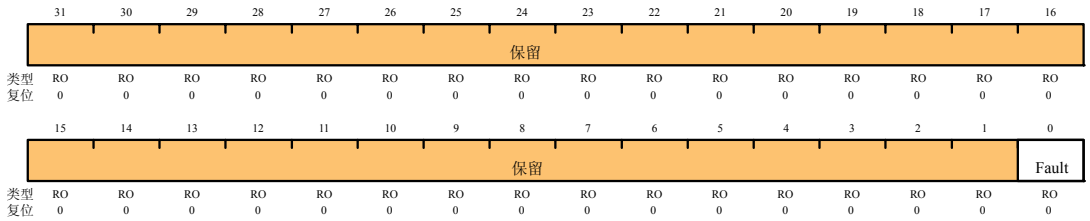


位/字段	名称	类型	复位	描述
31:17	保留	RO	0	保留位，返回不确定的值，并且不应该改变
16	IntFault	R/W1C	0	表示故障输入是否发出了一个中断
15: 3	保留	RO	0	保留位，返回不确定的值，并且不应该改变
2	IntPWM2	RO	0	表示 PWM 发生器 2 模块是否发出了一个中断
1	IntPWM1	RO	0	表示 PWM 发生器 1 模块是否发出了一个中断
0	IntPWM0	RO	0	表示 PWM 发生器 0 模块是否发出了一个中断

寄存器 9：PWM 状态（PWMSTATUS），偏移量 0x020

该寄存器提供故障输入信号的状态。

PWM 状态 (PWMSTATUS)
偏移量 0x020



位/字段	名称	类型	复位	描述
31:1	保留	RO	0	保留位，返回不确定的值，并且不应该改变
0	Fault	R/W	0	该位置位表示故障输入已经发出。

寄存器 10：PWM0 控制（PWM0CTL），偏移量 0x040

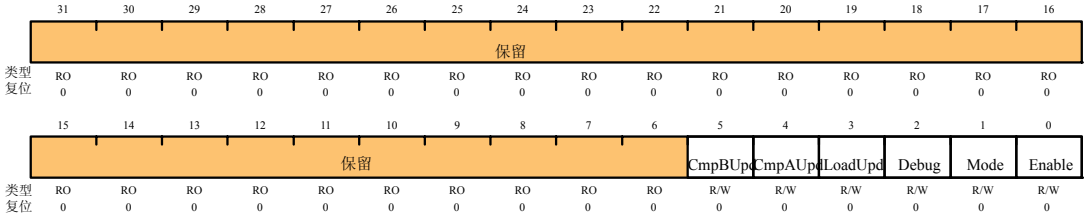
寄存器 11：PWM1 控制（PWM1CTL），偏移量 0x080

寄存器 12：PWM2 控制（PWM2CTL），偏移量 0x0C0

上述寄存器用来对 PWM 信号发生模块进行配置（**PWM0CTL** 控制 PWM 发生器 0 模块，以此类推）。寄存器更新模式、调试模式、计数模式、以及模块使能模式都是通过这些寄存器来控制的。PWM 模块可以产生两个独立的 PWM 信号（来自同一个计数器），或一对添加了死区延迟的 PWM 信号。

PWM0 模块产生 PWM0 和 PWM1 输出，PWM1 模块产生 PWM2 和 PWM3 输出，PWM2 模块产生 PWM4 和 PWM5 输出。

PWMn 控制 (PWMnCTL)



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	CmpBUpd	R/W	0	与 CmpAUpd 相同，只是用于比较器 B 寄存器。
4	CmpAUpd	R/W	0	比较器 A 寄存器的更新模式。如果该位为 0，则对寄存器的更新在计数器下一次到达 0 时反映给比较器。如果为 1，则将寄存器更新进行延迟，直到已通过 **PWM 主控制（PWMCTL）**寄存器请求了同步更新之后，且计数器下一次到达 0 时才反映给比较器。
3	LoadUpd	R/W	0	装载寄存器的更新模式。如果该位为 0，则对寄存器的更新在计数器下一次到达 0 时反映给计数器。如果为 1，则将寄存器更新进行延迟，直到已通过 **PWM 主控制（PWMCTL）**寄存器请求了同步更新之后，且计数器下一次到达 0 时才反映给计数器。
2	Debug	R/W	0	计数器在调试模式中的行为。如果该位为 0，则计数器在下一次到达 0 时停止运行，直至不再处于调试模式时，才再次继续运行。如果为 1，则计数器一直运行。
1	Mode	R/W	0	计数器的模式。如果该位为 0，则计数器从装载值开始递减计数到 0，然后回到装载值（递减计数模式）。如果为 1，则计数器先从 0 开始递增计数到装载值，然后再递减到 0，并一直重复该过程（先增后减计数模式）。
0	Enable	R/W	0	PWM 发生模块的主机使能。如果该位为 0，则整个模块禁止，并且断开时钟信号。如果为 1，则模块使能并提供 PWM 信号。

寄存器 13：PWM0 中断/触发使能（PWM0INTEN），偏移量 0x044

寄存器 14：PWM1 中断/触发使能（PWM1INTEN），偏移量 0x084

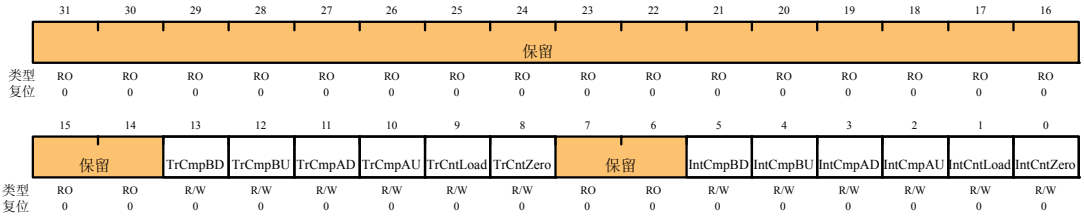
寄存器 15：PWM2 中断/触发使能（PWM2INTEN），偏移量 0x0C4

上述寄存器控制 PWM 发生器的中断和 ADC 触发产生功能(**PWM0INTEN** 控制 PWM 发生器 0 模块，以此类推)。能够引起中断或 ADC 触发的事件包括：

- 计数器等于装载寄存器
- 计数器等于零
- 递增计数时，计数器等于比较器 A 寄存器
- 递减计数时，计数器等于比较器 A 寄存器
- 递增计数时，计数器等于比较器 B 寄存器
- 递减计数时，计数器等于比较器 B 寄存器

尽管引起 ADC 触发的实际事件是无法确定的，但上述事件的任何组合都可以产生中断或 ADC 触发。

PWMn 中断/触发使能(PWMnINTEN)



位/字段	名称	类型	复位	描述
31:14	保留	RO	0	保留位，返回不确定的值，并且不应该改变
13	TrCmpBD	R/W	0	如果该位为 1，则当计数器与比较器 B 的值相等并且计数器正递减计数时，输出触发脉冲。
12	TrCmpBU	R/W	0	如果该位为 1，则当计数器与比较器 B 的值相等并且计数器正递增计数时，输出触发脉冲。
11	TrCmpAD	R/W	0	如果该位为 1，则当计数器与比较器 A 的值相等并且计数器正递减计数时，输出触发脉冲。
10	TrCmpAU	R/W	0	如果该位为 1，则当计数器与比较器 A 的值相等并且计数器正递增计数时，输出触发脉冲。
9	TrCntLoad	R/W	0	如果该位为 1，则当计数器与 **PWMnLOAD** 寄存器的值相等时，输出触发脉冲。
8	TrCntZero	R/W	0	如果该位为 1，则当计数器为零时输出触发脉冲。
7:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	IntCmpBD	R/W	0	如果该位为 1，则当计数器与比较器 B 的值相等并且计数器正递减计数时，产生中断。
4	IntCmpBU	R/W	0	如果该位为 1，则当计数器与比较器 B 的值相等并且计数器正递增计数时，产生中断。

续上表

位/字段	名称	类型	复位	描述
3	IntCmpAD	R/W	0	如果该位为 1，则当计数器与比较器 A 的值相等并且计数器正递减计数时，产生中断。
2	IntCmpAU	R/W	0	如果该位为 1，则当计数器与比较器 A 的值相等并且计数器正递增计数时，产生中断。
1	IntCntLoad	R/W	0	如果该位为 1，则当计数器与 **PWMnLOAD** 寄存器的值相等时，产生中断。
0	IntCntZero	R/W	0	如果该位为 1，则当计数器为零时产生中断。

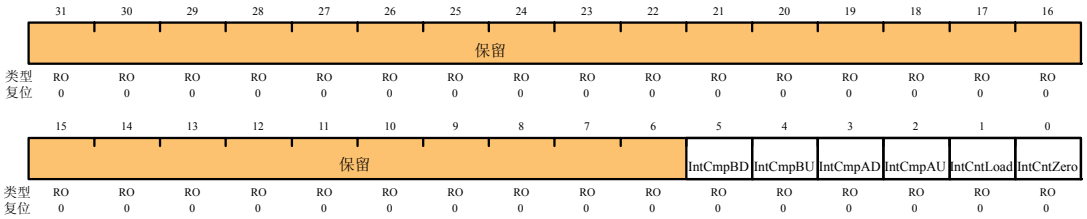
寄存器 16：PWM0 原始中断状态（PWM0RIS），偏移量 0x048

寄存器 17：PWM1 原始中断状态（PWM1RIS），偏移量 0x088

寄存器 18：PWM2 原始中断状态（PWM2RIS），偏移量 0x0C8

上述寄存器提供已发出的中断源的当前状态，不管它们是否引起了有效的控制器中断（PWM0RIS 控制 PWM 发生器 0 模块，以此类推）。寄存器中的位为 1 表示已发生锁存事件。为 0 表示正被查询的事件没有发生。

PWMn 原始中断状态 (PWMnRIS)



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	IntCmpBD	RO	0	表示在递减计数时计数器已与比较器 B 的值相等。
4	IntCmpBU	RO	0	表示在递增计数时计数器已与比较器 B 的值相等。
3	IntCmpAD	RO	0	表示在递减计数时计数器已与比较器 A 的值相等。
2	IntCmpAU	RO	0	表示在递增计数时计数器已与比较器 A 的值相等。
1	IntCntLoad	RO	0	表示计数器已与 **PWMnLOAD** 寄存器的值相等。
0	IntCntZero	RO	0	表示计数器已等于 0。

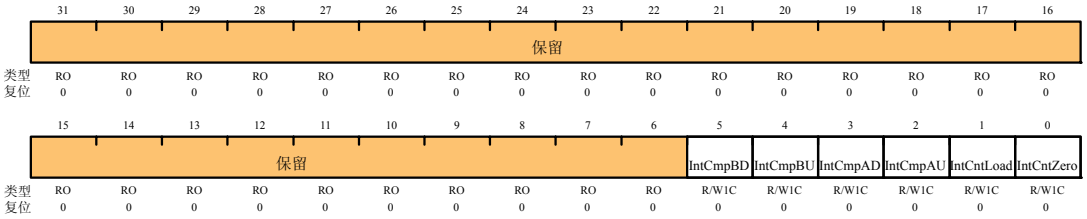
寄存器 19：PWM0 中断状态和清零（PWM0ISC），偏移量 0x04C

寄存器 20：PWM1 中断状态和清零（PWM1ISC），偏移量 0x08C

寄存器 21：PWM2 中断状态和清零（PWM2ISC），偏移量 0x0CC

上述寄存器提供已发出给控制器的中断源的当前设置状态（**PWM0ISC** 控制 PWM 发生器 0 模块，以此类推）。寄存器的位为 1 表示已出现锁存的事件。为 0 表示正被查询的事件没有发生。它们都是 R/W1C，即向某个位写 1 将使对应的中断原因清零。

PWMn 中断状态 (PWMnISC)



位/字段	名称	类型	复位	描述
31:6	保留	RO	0	保留位，返回不确定的值，并且不应该改变
5	IntCmpBD	R/W1C	0	表示在递减计数时计数器已与比较器 B 的值相等。
4	IntCmpBU	R/W1C	0	表示在递增计数时计数器已与比较器 B 的值相等。
3	IntCmpAD	R/W1C	0	表示在递减计数时计数器已与比较器 A 的值相等。
2	IntCmpAU	R/W1C	0	表示在递增计数时计数器已与比较器 A 的值相等。
1	IntCntLoad	R/W1C	0	表示计数器已与 **PWMnLOAD** 寄存器的值相等。
0	IntCntZero	R/W1C	0	表示计数器已等于 0。

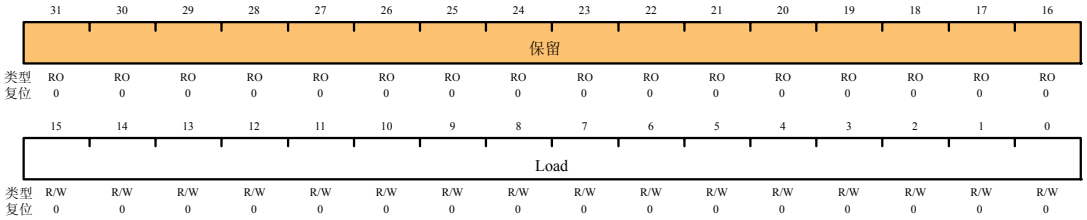
寄存器 22：PWM0 装载（PWM0LOAD），偏移量 0x050

寄存器 23：PWM1 装载（PWM1LOAD），偏移量 0x090

寄存器 24：PWM2 装载（PWM2LOAD），偏移量 0x0D0

上述寄存器包含 PWM 计数器的装载值（**PWM0LOAD** 控制 PWM 发生器 0 模块，以此类推）。根据计数模式，该值可在计数器到达零之后加载到计数器中，或在计数器递减到零之后，作为递增计数的界限。如果装载值更新模式为立即模式，则在下一次计数器到达零时使用该值。如果为同步模式，则在通过 **PWM 主控制（PWMCTL）**寄存器请求了同步更新之后，下一次计数器到达零时使用该值。如果在实际更新装载值之前重写该寄存器，则之前的值会丢失，尽管它还没有被采用。

PWMn 装载 (PWMnLOAD)



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变
15:0	Load	R/W	0	计数器的装载值

寄存器 25：PWM0 计数器（PWM0COUNT），偏移量 0x054

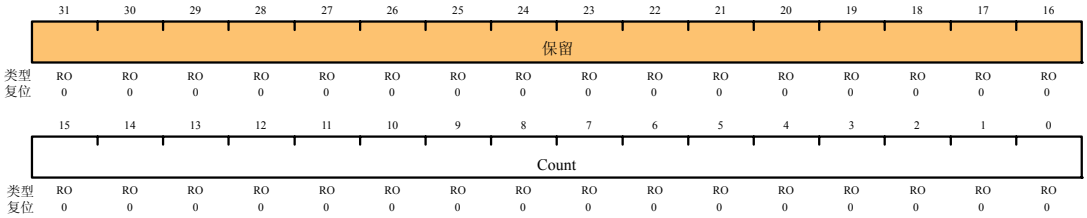
寄存器 26：PWM1 计数器（PWM1COUNT），偏移量 0x094

寄存器 27：PWM2 计数器（PWM2COUNT），偏移量 0x0D4

上述寄存器包含 PWM 计数器的当前值（**PWM0COUNT** 控制 PWM 发生器 0 模块，

以此类推）。当该值与装载寄存器的值相等时，产生一个脉冲，该脉冲能够驱动 PWM 信号的产生（通过 **PWMnGENA/PWMnGENB** 寄存器），驱动中断或 ADC 触发（通过 **PWMnINTEN** 寄存器）。当该值为零时，将产生具有相同功能的脉冲。

PWMn 计数器 (PWMnCOUNT)



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变
15:0	Count	R/W	0	计数器的当前值

寄存器 28：PWM0 比较 A（PWM0CMPA），偏移量 0x058

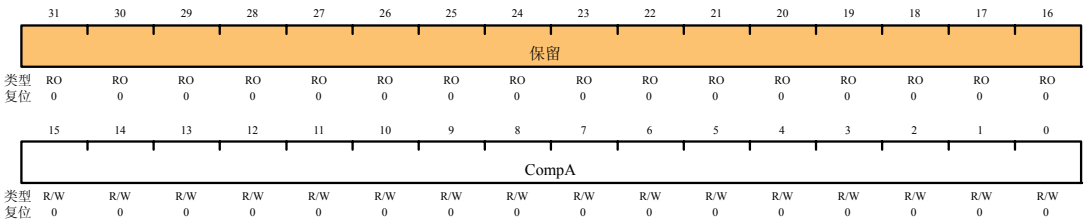
寄存器 29：PWM1 比较 A（PWM1CMPA），偏移量 0x098

寄存器 30：PWM2 比较 A（PWM2CMPA），偏移量 0x0D8

上述寄存器包含与计数器进行比较的值（**PWM0CMPA** 控制 PWM 发生器 0 模块，以此类推）。当该值与计数器的值相等时，输出一个脉冲。该脉冲能够驱动 PWM 信号的产生（通过 **PWMnGENA/PWMnGENB** 寄存器），驱动中断或 ADC 触发（通过 **PWMnINTEN** 寄存器）。如果该寄存器的值大于 **PWMnLOAD** 寄存器的值，则始终不输出脉冲。

对于比较器 A，如果更新模式为立即模式（根据 **PWMnCTL** 寄存器的 CmpAUpd 位），则在计数器下一次到达零时使用该寄存器的 16 位 CompA。如果为同步更新，则在通过 **PWM 主控制（PWMCTL）**寄存器请求了同步更新之后，且等到计数器下一次到达零时使用该值。如果在进行实际更新之前重写该寄存器，则之前的值会丢失，尽管它还没有被采用。

PWMn 比较 A (PWMnCMPA)



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变
15:0	CompA	R/W	0	与计数器进行比较的值

寄存器 31：PWM0 比较 B（PWM0CMPB），偏移量 0x05C

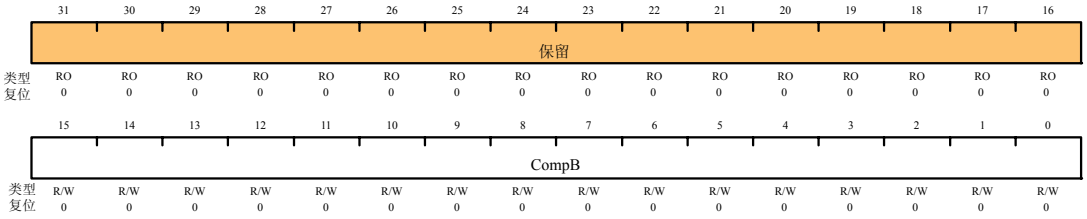
寄存器 32：PWM1 比较 B（PWM1CMPB），偏移量 0x09C

寄存器 33：PWM2 比较 B（PWM2CMPB），偏移量 0x0DC

上述寄存器包含与计数器进行比较的值（**PWM0CMPB** 控制 PWM 发生器 0 模块，以此类推）。当该值与计数器的值相等时，输出一个脉冲。该脉冲能够驱动 PWM 信号的产生（通过 **PWMnGENA/PWMnGENB** 寄存器），驱动中断或 ADC 触发（通过 **PWMnINTEN** 寄存器）。如果该寄存器的值大于 **PWMnLOAD** 寄存器的值，则始终不输出脉冲。

对于比较器 B，如果更新模式为立即模式（根据 **PWMnCTL** 寄存器的 CmpBUpd 位），则在计数器下一次到达零时使用该寄存器的 16 位 CompB。如果为同步更新，则在通过 **PWM 主控制（PWMCTL）**寄存器请求了同步更新之后，且等到计数器下一次到达零时使用该值。如果在进行实际更新之前重写该寄存器，则之前的值从会丢失，尽管它还没有被采用。

PWMn 比较 B (PWMnCMPB)



位/字段	名称	类型	复位	描述
31:16	保留	RO	0	保留位，返回不确定的值，并且不应该改变
15:0	CompB	R/W	0	与计数器进行比较的值

寄存器 34：PWM0 发生器 A 控制（PWM0GENA），偏移量 0x060

寄存器 35：PWM1 发生器 A 控制（PWM1GENA），偏移量 0x0A0

寄存器 36：PWM2 发生器 A 控制（PWM2GENA），偏移量 0x0E0

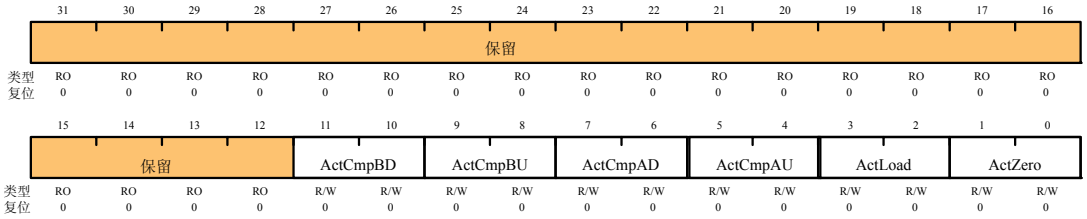
上述寄存器根据来自计数器的装载输出脉冲和零输出脉冲以及来自比较器的比较 A 脉冲和比较 B 脉冲来控制 PWMnA 信号的产生（**PWM0GENA** 控制 PWM 发生器 0 模块，以此类推）。当计数器在递减计数模式中运行时，只会出现其中的 4 个事件。当在先增后减模式中运行时，6 个事件都会出现。这些事件在确定 PWM 信号产生的位置及信号的占空比时具有极大的灵活性。

PWM0GENA 寄存器控制 PWM0A 信号的产生；**PWM1GENA** 寄存器控制 PWM1A 信号的产生；**PWM2GENA** 寄存器控制 PWM2A 信号的产生。

表 15.2 中定义了各个事件对输出信号的影响，寄存器中的每个字段都可以是该表所定义的某个值。

如果零或装载事件与比较 A 或比较 B 事件同时发生，那么将会对零或装载事件作出响应，而比较 A 或比较 B 将被忽略。如果比较 A 事件与比较 B 事件同时发生，那么会对比较 A 事件作出响应，而比较 B 事件将被忽略。

PWMn 发生器A控制 (PWMnGENA)



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位，返回不确定的值，并且不应该改变
11:10	ActCmpBD	R/W	0	在递减计数时，当计数器的值与比较器 B 相等时采取的行动
9:8	ActCmpBU	R/W	0	在递增计数时，当计数器的值与比较器 B 相等时采取的动作。只有当 **PWMnCTL** 寄存器的 Mode 位设置为 1 时才发生。
7:6	ActCmpAD	R/W	0	在递减计数时，当计数器的值与比较器 A 相等时采取的动作
5:4	ActCmpAU	R/W	0	在递增计数时，当计数器的值与比较器 A 相等时采取的动作。只有当 **PWMnCTL** 寄存器的 Mode 位设置为 1 时才发生。
3:2	ActLoad	R/W	0	当计数器的值与装载值相等时采取的动作。
1:0	ActZero	R/W	0	当计数器的值为零时采取的动作。

表 15.2 PWM 发生器的动作编码

值	描述
00	不采取任何动作
01	将输出信号反相
10	将输出信号设置为 0
11	将输出信号设置为 1

寄存器 37：PWM0 发生器 B 控制（PWM0GENB），偏移量 0x064

寄存器 38：PWM1 发生器 B 控制（PWM1GENB），偏移量 0x0A4

寄存器 39：PWM2 发生器 B 控制（PWM2GENB），偏移量 0x0E4

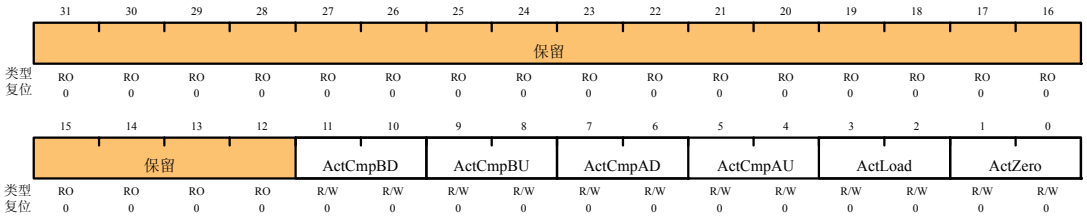
上述寄存器根据来自计数器的装载输出脉冲和零输出脉冲以及来自比较器的比较 A 脉冲和比较 B 脉冲来控制 PWMnB 信号的产生（**PWM0GENB** 控制 PWM 发生器 0 模块，以此类推）。当计数器在递减计数模式中运行时，只出现其中的 4 个事件。当在先增后减模式中运行时，6 个事件都会出现。这些事件在确定 PWM 信号产生的位置及信号的占空比时具有极大的灵活性。

PWM0GENB 寄存器控制 PWM0B 信号的产生；**PWM1GENB** 寄存器控制 PWM1B 信号的产生。**PWM2GENB** 寄存器控制 PWM2B 信号的产生。

表 15.2 中定义了各个事件对输出信号的影响，寄存器中的每个字段都可以是该表定义的某个值。

如果零或装载事件与比较 A 或比较 B 事件同时发生，那么会对零或装载事件作出响应，而比较 A 或比较 B 事件将被忽略。如果比较 A 事件与比较 B 事件同时发生，那么会对比较 A 事件作出响应，而比较 B 事件将被忽略。

PWMn 发生器 B 控制 (PWMnGENB)



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位，返回不确定的值，并且不应该改变
11:10	ActCmpBD	R/W	0	在递减计数时，当计数器的值与比较器 B 相等时采取的动作
9:8	ActCmpBU	R/W	0	在递增计数时，当计数器的值与比较器 B 相等时采取的动作。只有当 **PWMnCTL** 寄存器的 Mode 位设置为 1 时才发生。
7:6	ActCmpAD	R/W	0	在递减计数时，当计数器的值与比较器 A 相等时采取的动作
5:4	ActCmpAU	R/W	0	在递增计数时，当计数器的值与比较器 A 相等时采取的动作。只有当 **PWMnCTL** 寄存器的 Mode 位设置为 1 时才发生。
3:2	ActLoad	R/W	0	当计数器的值与装载值相等时采取的动作。
1:0	ActZero	R/W	0	当计数器的值为零时采取的动作。

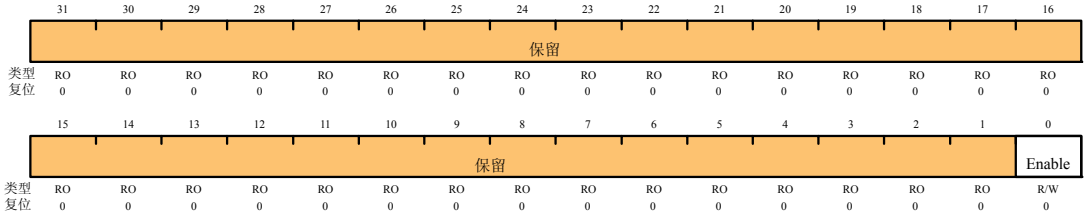
寄存器 40：PWM0 死区控制（PWM0DBCTL），偏移量 0x068

寄存器 41：PWM1 死区控制（PWM1DBCTL），偏移量 0x0A8

寄存器 42：PWM2 死区控制（PWM2DBCTL），偏移量 0x0E8

PWM0DBCTL 寄存器用来控制死区发生器。死区发生器根据信号 PWM0A 和 PWM0B 产生信号 PWM0 和 PWM1。当死区功能被禁止时，PWM0A 直接通过死区模块成为信号 PWM0，而 PWM0B 直接通过死区模块成为信号 PWM1。当死区功能被使能时，PWM0B 信号会被忽略掉，而延迟 PWM0A 信号的上升沿来产生 PWM0，延迟时间由 **PWM0DBRISE** 寄存器的值确定，同时延迟 PWM0A 信号的下降沿来产生 PWM1 信号，延迟时间由 **PWM0DBFALL** 寄存器的值确定。同样，PWM2 和 PWM3 由 PWM1A 和 PWM1B 信号来产生。PWM4 和 PWM5 由 PWM2A 和 PWM2B 信号来产生。

PWMn 死区控制 (PWMnDBCTL)



位/字段	名称	类型	复位	描述
31:1	保留	RO	0	保留位，返回不确定的值，并且不应该改变
0	Enable	R/W	0	该位置位时，死区发生器会将死区插入到输出信号中；为零时，只是简单地通过死区模块传递 PWM 信号。

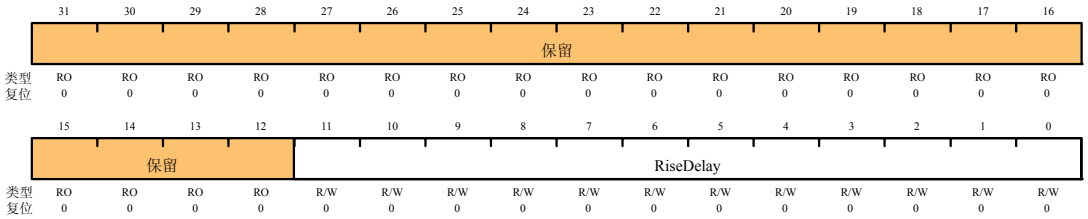
寄存器 43：PWM0 死区上升沿延迟（PWM0DBRISE），偏移量 0x06C

寄存器 44：PWM1 死区上升沿延迟（PWM1DBRISE），偏移量 0x0AC

寄存器 45：PWM2 死区上升沿延迟（PWM2DBRISE），偏移量 0x0EC

PWM0DBRISE 寄存器所包含的是生成信号 PWM0 时对信号 PWM0A 的上升沿进行延迟的时钟周期数。如果通过 **PWMnDBCTL** 寄存器将死区发生器禁止，则寄存器 **PWM0DBRISE** 将被忽略。如果该寄存器的值大于 PWM 输入信号的高电平宽度，则上升沿延迟会占用信号的整个高电平时间，从而导致在输出上没有高电平。因此，必须注意要确保输入信号的高电平时间始终大于上升沿延迟。同样，PWM2 由带上升沿延迟的 PWM1A 产生。PWM4 由带上升沿延迟的 PWM2A 产生。

PWMn 死区上升沿延迟 (PWMnDBRISE)



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位，返回不确定的值，并且不应该改变
11:0	RiseDelay	R/W	0	将上升沿进行延迟的时钟数

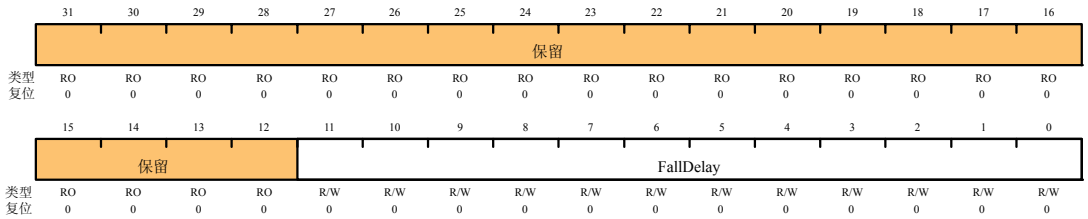
寄存器 46：PWM0 死区下降沿延迟（PWM0DBFALL），偏移量 0x070

寄存器 47：PWM1 死区下降沿延迟（PWM1DBFALL），偏移量 0x0B0

寄存器 48：PWM2 死区下降沿延迟（PWM2DBFALL），偏移量 0x0F0

PWM0DBFALL 寄存器所包含的是生成信号 PWM1 时对信号 PWM0A 的下降沿进行延迟的时钟周期数。如果死区发生器被禁止，则忽略该寄存器。如果该寄存器的值大于 PWM 输入信号的低电平宽度，则下降沿延迟会占用信号的整个低电平时间，从而导致在输出上没有低电平。因此，必须注意要确保输入信号的低电平时间始终大于下降沿延迟。同样，PWM3 由带下降沿延迟的 PWM1A 产生。PWM5 由带下降沿延迟的 PWM2A 产生。

PWMn 死区下降沿延迟 (PWMnDBFALL)



位/字段	名称	类型	复位	描述
31:12	保留	RO	0	保留位，返回不确定的值，并且不应该改变
11:0	FallDelay	R/W	0	将下降沿进行延迟的时钟数

第16章 管脚图

图 16.1 所示为管脚图和管脚与信号名称对应图。

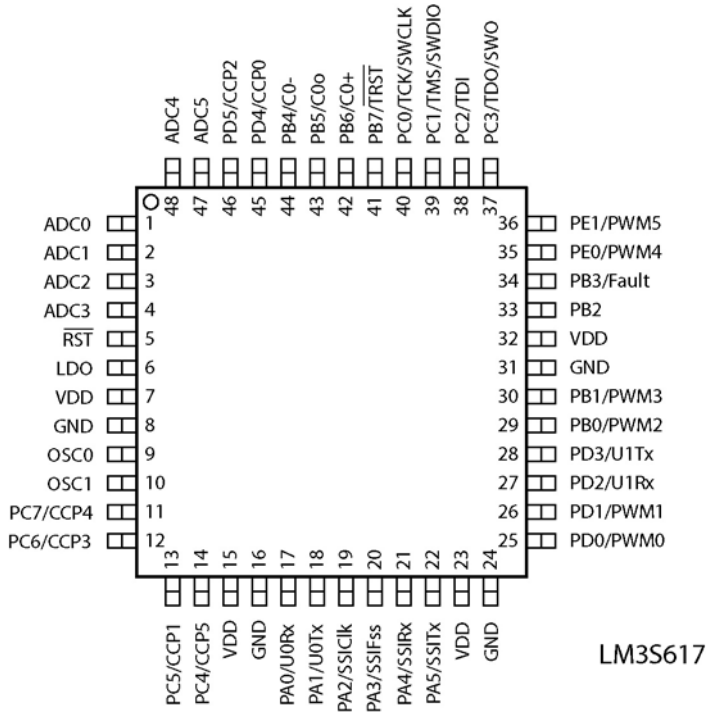


图 16.1 管脚连接图

第17章 信号表

下表列出了每个管脚可用的信号。通过软件使用 **GPIOAFSEL** 寄存器来使能管脚功能。

要点：在默认情况下所有复用管脚都为 GPIO 管脚，只有 5 个 JTAG 管脚(PB7 和 PC[3:0])除外，这 5 个管脚默认用作 JTAG 功能。

表 17.1 所示为管脚到信号名称的映射，包括信号的功能特性。表 17.2 按照信号名称的字母顺序列出信号。表 17.3 按照功能将信号分组，GPIO管脚除外。表 17.4 列出GPIO管脚和它们的备用功能。

表 17.1 按管脚编号排列的信号

管脚编号	信号名称	管脚类型	缓冲区类型	描述
1	ADC0	I	模拟	模数比较器输入 0。
2	ADC1	I	模拟	模数转换器输入 1。
3	ADC2	I	模拟	模数转换器输入 2。
4	ADC3	I	模拟	模数转换器输入 3。
5	$\overline{\text{RST}}$	I	TTL	系统复位输入。
6	LDO	-	电源	线性稳压器输出电压。在该管脚和 GND 之间需要一个 1uF 或更大的外部电容。
7	VDD	-	电源	逻辑和 I/O 管脚的正电源
8	GND	-	电源	逻辑和 I/O 管脚的地参考
9	OSC0	I	模拟	振荡器晶体输入或外部时钟参考输入。
10	OSC1	O	模拟	振荡器晶体输出。
11	PC7	I/O	TTL	GPIO 端口 C 位 7。
	CCP4	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 4。
12	PC6	I/O	TTL	GPIO 端口 C 位 6。
	CCP3	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 3。
13	PC5	I/O	TTL	GPIO 端口 C 位 5。
	CCP1	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 1。
14	PC4	I/O	TTL	GPIO 端口 C 位 4。
	CCP5	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 5。
15	VDD	-	电源	逻辑和 I/O 管脚的正电源。
16	GND	-	电源	逻辑和 I/O 管脚的地参考。
17	PA0	I/O	TTL	GPIO 端口 A 位 0。
	U0Rx	I	TTL	UART0 接收数据输入。
18	PA1	I/O	TTL	GPIO 端口 A 位 1。
	U0Tx	O	TTL	UART0 发送数据输出。
19	PA2	I/O	TTL	GPIO 端口 A 位 2。
	SSIClk	I/O	TTL	SSI 时钟基准（在从模式下时为输入，在主模式下时为输出）。

续上表

管脚编号	信号名称	管脚类型	缓冲区类型	描述
20	PA3	I/O	TTL	GPIO 端口 A 位 3。
	SSIFss	I/O	TTL	SSI 帧使能（对于 SSI 从器件是输入，对于 SSI 主器件是输出）。
21	PA4	I/O	TTL	GPIO 端口 A 位 4。
	SSIRx	I	TTL	SSI 接收数据输入。
22	PA5	I/O	TTL	GPIO 端口 A 位 5。
	SSITx	O	TTL	SSI 发送数据输出。
23	VDD	-	电源	逻辑和 I/O 管脚的正电源。
24	GND	-	电源	逻辑和 I/O 管脚的地参考。
25	PD0	I/O	TTL	GPIO 端口 D 位 0。
	PWM0	O	TTL	脉宽调制器通道 0 输出。
26	PD1	I/O	TTL	GPIO 端口 D 位 1。
	PWM1	O	TTL	脉宽调制器通道 1 输出。
27	PD2	I/O	TTL	GPIO 端口 D 位 2。
	U1Rx	I	TTL	UART1 接收数据输入。
28	PD3	I/O	TTL	GPIO 端口 D 位 3。
	U1Tx	O	TTL	UART1 发送数据输出。
29	PB0	I/O	TTL	GPIO 端口 B 位 0。
	PWM2	O	TTL	脉宽调制器通道 2 输出。
30	PB1	I/O	TTL	GPIO 端口 B 位 1。
	PWM3	O	TTL	脉宽调制器通道 3 输出。
31	GND	-	电源	逻辑和 I/O 管脚的地参考。
32	VDD	-	电源	逻辑和 I/O 管脚的正电源。
33	PB2	I/O	TTL	GPIO 端口 B 位 2。
34	PB3	I/O	TTL	GPIO 端口 B 位 3。
	FAULT	I	TTL	电动机控制 0 故障。
35	PE0	I/O	TTL	GPIO 端口 E 位 0。
	PWM4	O	TTL	脉宽调制器通道 4 输出。
36	PE1	I/O	TTL	GPIO 端口 E 位 1。
	PWM5	O	TTL	脉宽调制器通道 5 输出。
37	PC3	I/O	TTL	GPIO 端口 C 位 3。
	TDO	O	TTL	JTAG 扫描测试数据输出。
	SWO	O	TTL	串行线输出。
38	PC2	I/O	TTL	GPIO 端口 C 位 2。
	TDI	I	TTL	JTAG 扫描测试数据输入。
39	PC1	I/O	TTL	GPIO 端口 C 位 1
	TMS	I	TTL	JTAG 扫描测试模式选择输入。
	SWDIO	I/O	TTL	串行线调试输入/输出。

续上表

管脚编号	信号名称	管脚类型	缓冲区类型	描述
40	PC0	I/O	TTL	GPIO 端口 C 位 0。
	TCK	I	TTL	JTAG 扫描测试时钟参考输入。
	SWCLK	I	TTL	串行线时钟参考输入。
41	PB7	I/O	TTL	GPIO 端口 B 位 7。
	$\overline{\text{TRST}}$	I	TTL	JTAG 扫描测试复位输入。
42	PB6	I/O	TTL	GPIO 端口 B 位 6。
	C0+	I	模拟	模拟比较器 0 正相参考输入。
43	PB5	I/O	TTL	GPIO 端口 B 位 5。
	C0o	O	TTL	模拟比较器 0 输出
44	PB4	I/O	TTL	GPIO 端口 B 位 4。
	C0-	I	模拟	模拟比较器 0 反相参考输入。
45	PD4	I/O	TTL	GPIO 端口 D 位 4。
	CCP0	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 0。
46	PD5	I/O	TTL	GPIO 端口 D 位 5。
	CCP2	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 2。
47	ADC5	I	模拟	模数转换器输入 5。
48	ADC4	I	模拟	模数转换器输入 4。

表 17.2 按信号名称排列的信号

信号名称	管脚编号	管脚类型	缓冲区类型	描述
ADC0	1	I	模拟	模数转换器输入 0。
ADC1	2	I	模拟	模数转换器输入 1。
ADC2	3	I	模拟	模数转换器输入 2。
ADC3	4	I	模拟	模数转换器输入 3。
ADC4	48	I	模拟	模数转换器输入 4。
ADC5	47	I	模拟	模数转换器输入 5。
C0+	42	I	模拟	模拟比较器 0 正相参考输入。
C0-	44	I	模拟	模拟比较器 0 反相参考输入。
C0o	43	O	TTL	模拟比较器 0 输出。
CCP0	45	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 0。
CCP1	13	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 1。
CCP2	46	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 2。
CCP3	12	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 3。
CCP4	11	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 4。
CCP5	14	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 5。
Fault	34	I	TTL	PWM 故障检测输入。
GND	8	-	电源	逻辑和 I/O 管脚的地参考。
GND	16	-	电源	逻辑和 I/O 管脚的地参考。
GND	24	-	电源	逻辑和 I/O 管脚的地参考。
GND	31	-	电源	逻辑和 I/O 管脚的地参考。

续上表

信号名称	管脚编号	管脚类型	缓冲区类型	描述
LDO	6	-	电源	线性稳压器输出电压。在该管脚和 GND 之间需要一个 1uF 或更大的外部电容。
OSC0	9	I	模拟	振荡器晶体输入或外部时钟参考输入。
OSC1	10	O	模拟	振荡器晶体输出。
PA0	17	I/O	TTL	GPIO 端口 A 位 0。
PA1	18	I/O	TTL	GPIO 端口 A 位 1。
PA2	19	I/O	TTL	GPIO 端口 A 位 2。
PA3	20	I/O	TTL	GPIO 端口 A 位 3。
PA4	21	I/O	TTL	GPIO 端口 A 位 4。
PA5	22	I/O	TTL	GPIO 端口 A 位 5。
PB0	29	I/O	TTL	GPIO 端口 B 位 0。
PB1	30	I/O	TTL	GPIO 端口 B 位 1。
PB2	33	I/O	TTL	GPIO 端口 B 位 2。
PB3	34	I/O	TTL	GPIO 端口 B 位 3。
PB4	44	I/O	TTL	GPIO 端口 B 位 4。
PB5	43	I/O	TTL	GPIO 端口 B 位 5。
PB6	42	I/O	TTL	GPIO 端口 B 位 6。
PB7	41	I/O	TTL	GPIO 端口 B 位 7。
PC0	40	I/O	TTL	GPIO 端口 C 位 0。
PC1	39	I/O	TTL	GPIO 端口 C 位 1。
PC2	38	I/O	TTL	GPIO 端口 C 位 2。
PC3	37	I/O	TTL	GPIO 端口 C 位 3。
PC4	14	I/O	TTL	GPIO 端口 C 位 4。
PC5	13	I/O	TTL	GPIO 端口 C 位 5。
PC6	12	I/O	TTL	GPIO 端口 C 位 6。
PC7	11	I/O	TTL	GPIO 端口 C 位 7。
PD0	25	I/O	TTL	GPIO 端口 D 位 0。
PD1	26	I/O	TTL	GPIO 端口 D 位 1。
PD2	27	I/O	TTL	GPIO 端口 D 位 2。
PD3	28	I/O	TTL	GPIO 端口 D 位 3。
PD4	45	I/O	TTL	GPIO 端口 D 位 4。
PD5	46	I/O	TTL	GPIO 端口 D 位 5。
PE0	35	I/O	TTL	GPIO 端口 E 位 0。
PE1	36	I/O	TTL	GPIO 端口 E 位 1。
PWM0	25	O	TTL	脉宽调制器通道 0 输出。
PWM1	26	O	TTL	脉宽调制器通道 1 输出。
PWM2	29	O	TTL	脉宽调制器通道 2 输出。
PWM3	30	O	TTL	脉宽调制器通道 3 输出。
PWM4	35	O	TTL	脉宽调制器通道 4 输出。
PWM5	36	O	TTL	脉宽调制器通道 5 输出。

续上表

信号名称	管脚编号	管脚类型	缓冲区类型	描述
$\overline{RST}$	5	I	TTL	系统复位输入。
SSIClk	19	I/O	TTL	SSI 时钟参考（在从模式中为输入，主模式中为输出）
SSIFss	20	I/O	TTL	SSI 帧使能（对于 SSI 从器件是输入，对于 SSI 主器件是输出）。
SSIRx	21	I	TTL	SSI 接收数据输入。
SSITx	22	O	TTL	SSI 发送数据输出。
SWCLK	40	I	TTL	串行线时钟参考输入。
SWDIO	39	I/O	TTL	串行线调试输入/输出。
SWO	37	O	TTL	串行线输出。
TCK	40	I	TTL	JTAG 扫描测试时钟参考输入。
TDI	38	I	TTL	JTAG 扫描测试数据输入。
TDO	37	O	TTL	JTAG 扫描测试数据输出。
TMS	39	I	TTL	JTAG 扫描测试模式选择管脚。
$\overline{TRST}$	41	I	TTL	JTAG 扫描测试复位输入。
U0Rx	17	I	TTL	UART0 接收数据输入。
U0Tx	18	O	TTL	UART0 发送数据输出。
U1Rx	27	I	TTL	UART1 接收数据输入。
U1Tx	28	O	TTL	UART1 发送数据输出。
VDD	7	-	电源	逻辑和 I/O 管脚的正电源。
VDD	15	-	电源	逻辑和 I/O 管脚的正电源。
VDD	23	-	电源	逻辑和 I/O 管脚的正电源。
VDD	32	-	电源	逻辑和 I/O 管脚的正电源。

表 17.3 按功能分组的信号，GPIO 管脚除外

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
ADC	ADC0	1	I	模拟	模数转换器输入 0。
	ADC1	2	I	模拟	模数转换器输入 1。
	ADC2	3	I	模拟	模数转换器输入 2。
	ADC3	4	I	模拟	模数转换器输入 3。
	ADC4	48	I	模拟	模数转换器输入 4。
	ADC5	47	I	模拟	模数转换器输入 5。
模拟比较器	C0+	42	I	模拟	模拟转换器 0 正相参考输入。
	C0-	44	I	模拟	模拟比较器 0 反相参考输入。
	C0o	43	O	TTL	模拟比较器 0 输出。

续上表

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
通用定时器	CCP0	45	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 0
	CCP1	13	I/O	TTL	Timer0 捕获输入、比较输出或 PWM 输出通道 1
	CCP2	46	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 2
	CCP3	12	I/O	TTL	Timer1 捕获输入、比较输出或 PWM 输出通道 3
	CCP4	11	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 4
	CCP5	14	I/O	TTL	Timer2 捕获输入、比较输出或 PWM 输出通道 5
JTAG/SWD/ SWO	SWCLK	40	I	TTL	串行线时钟参考输入
	SWDIO	39	I/O	TTL	串行线调试输入/输出
	SWO	37	O	TTL	串行线输出
	TCK	40	I	TTL	JTAG 扫描测试时钟参考输入
	TDI	38	I	TTL	JTAG 扫描测试数据输入
	TDO	37	O	TTL	JTAG 扫描测试数据输出
	TMS	39	I	TTL	JTAG 扫描测试模式选择输入
	$\overline{\text{TRST}}$	41	I	TTL	JTAG 扫描测试复位输入
电源	GND	8	-	电源	逻辑和 I/O 管脚的地参考
	GND	16	-	电源	逻辑和 I/O 管脚的地参考
	GND	24	-	电源	逻辑和 I/O 管脚的地参考
	GND	31	-	电源	逻辑和 I/O 管脚的地参考
	LDO	6	-	电源	线性稳压器输出电压。在该管脚和 GND 之间需要一个 1uF 或更大的外部电容。
	VDD	7	-	电源	逻辑和 I/O 管脚的正电源
	VDD	15	-	电源	逻辑和 I/O 管脚的正电源
	VDD	23	-	电源	逻辑和 I/O 管脚的正电源
	VDD	32	-	电源	逻辑和 I/O 管脚的正电源
PWM	PWM0	25	O	TTL	脉宽调制器通道 0 输出。
	PWM1	26	O	TTL	脉宽调制器通道 1 输出。
	PWM2	29	O	TTL	脉宽调制器通道 2 输出。
	PWM3	30	O	TTL	脉宽调制器通道 3 输出。
	PWM4	35	O	TTL	脉宽调制器通道 4 输出。
	PWM5	36	O	TTL	脉宽调制器通道 5 输出。

续上表

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
SSI	SSIClk	19	I/O	TTL	SSI 时钟参考（该管脚在从模式中用作输入，在主模式中用作输出）
	SSIFss	20	I/O	TTL	SSI 帧使能（对于 SSI 从器件该管脚为输入，对于 SSI 主器件该管脚为输出）
	SSIRx	21	I	TTL	SSI 接收数据输入
	SSITx	22	O	TTL	SSI 发送数据输出
系统控制和时钟	OSC0	9	I	模拟	振荡器晶体的输入或外部时钟参考输入
	OSC1	10	O	模拟	振荡器晶体的输出
	$\overline{\text{RST}}$	5	I	TTL	系统复位输入
UART	U0Rx	17	I	TTL	UART0 接收数据输入
	U0Tx	18	O	TTL	UART0 发送数据输出
	U1Rx	27	I	TTL	UART1 接收数据输入
	U1Tx	28	O	TTL	UART1 发送数据输出。

表 17.4 GPIO 管脚和可选的功能

GPIO 管脚	管脚编号	复用功能	复用功能
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIClk	
PA3	20	SSIFss	
PA4	21	SSIRx	
PA5	22	SSITx	
PB0	29	PWM2	
PB1	30	PWM3	
PB2	33		
PB3	34	FAULT	
PB4	44	C0-	
PB5	43	C0o	
PB6	42	C0+	
PB7	41	$\overline{\text{TRST}}$	
PC0	40	TCK	SWCLK
PC1	39	TMS	SWDIO
PC2	38	TDI	
PC3	37	TDO	SWO
PC4	14	CCP5	
PC5	13	CCP1	
PC6	12	CCP3	

续上表

GPIO 管脚	管脚编号	复用功能	复用功能
PC7	11	CCP4	
PD0	25	PWM0	
PD1	26	PWM1	
PD2	27	U1Rx	
PD3	28	U1Tx	
PD4	45	CCP0	
PD5	46	CCP2	
PD6	47	ADC5	
PD7	48	ADC4	
PE0	35	PWM4	
PE1	36	PWM5	
PE2	4	ADC3	
PE3	3	ADC2	
PE4	2	ADC1	
PE5	1	ADC0	

第18章 工作特性

表 18.1 温度特性

特性	符号	值	单位
工作温度范围 [a]	T_A	工业级温度：-40～+85	℃

a. 最大的存储温度为 150℃。

表 18.2 热特性(Thermal Characteristics)

特性	符号	值	单位
热电阻（结点到环境） [a]	θ_{JA}	76	℃/W
平均结点温度 [b]	T_J	$T_A + (P_{AVG} \cdot \theta_{JA})$	℃
最大结点温度	T_{JMAX}	未定(pending) [c]	℃

a. 结点到环境热电阻 θ_{JA} 的值由封装模拟器（package simulator）确定。

b. 功率损耗是温度的函数。

c. 特性未定。

第19章 电气特性

19.1 直流（DC）特性

19.1.1 最大额定值

最大额定值是指器件能够承受的极限值，超过最大额定值可能会造成器件永久损坏。

注：不能保证器件在最大额定值下能正确工作。

表 19.1 最大额定值

特性 [a]	符号	值	单位
电源电压范围（V_{DD}）	V_{DD}	0.0 ～ +3.6	V
输入电压	V_{IN}	-0.3 ～5.5	V
管脚上的最大电流，充当 GPIO 功能的管脚除外	I	100	mA
GPIO 管脚的最大电流	I	100	mA

a. 所测电压都相对于 GND

要点：该器件含有保护电路，可以避免高静电压或电磁场对输入信号造成损害；但建议仍要采取一些常规措施，以避免在这个高阻抗电路上施加的电压高于最大额定电压。如果将未用的输入与相应的逻辑电压电平（例如，GND 或 V_{DD}）相连，可以提高工作的稳定性。

19.1.2 建议的直流工作条件

表 19.2 建议的直流工作条件

参数	参数名称	最小值	额定值	最大值	单位
V_{DD}	电源电压	3.0	3.3	3.6	V
V_{IH}	高电平输入电压	2.0	-	5.0	V
V_{IL}	低电平输入电压	-0.3	-	1.3	V
V_{SIH}	肖特基（Schottky）输入的高电平输入电压	0.8* V_{DD}	-	V_{DD}	V
V_{SIL}	肖特基（Schottky）输入的低电平输入电压	0	-	0.2* V_{DD}	V
V_{OH}	高电平输出电压	2.4	-	-	V
V_{OL}	低电平输出电压	-	-	0.4	V
I_{OH}	高电平源电流(source current)，V_{OH}=2.4V				
	2-mA 驱动	2.0	-	-	mA
	4-mA 驱动	4.0	-	-	mA
	8-mA 驱动	8.0	-	-	mA
I_{OL}	低电平灌电流（sink current），V_{OL}=0.4V				
	2-mA 驱动	2.0	-	-	mA
	4-mA 驱动	4.0	-	-	mA
	8-mA 驱动	8.0	-	-	mA

19.1.3 片内线性压差（Linear Drop-Out）稳压器特性

表 19.3 LDO 稳压器特性

参数	参数名称	最小值	额定值	最大值	单位
V_{LDOOUT}	可编程内部（逻辑）电源输出值	2.25	-	2.75	V
	输出电压的精度	-	2%	-	%
t_{PON}	上电时间	-	-	100	μs
t_{ON}	导通时间（time on）	-	-	200	μs
t_{OFF}	关断时间（time off）	-	-	100	μs
V_{STEP}	每一步的增量电压	-	50	-	mV
C_{LDO}	用于内部电源的外部滤波器的电容规格	-	1	-	μF

19.1.4 功率规范

表 19.4 所示的功率测量结果是使用SRAM的内核处理器在以下规范下测得的：

- V_{DD}=3.3V
- LDO=2.5
- 温度＝25℃
- 系统时钟＝50MHz（带 PLL）
- 无活动的（active）外设，代码 while(1){}从 SRAM 执行

表 19.4 功率规范

参数	参数名称	最小值	额定值	最大值	单位
I_{DD_RUN}	运行模式	-	70[a]	未定[a]	mA
I_{DD_SLEEP}	睡眠模式	-	未定[a]	未定[a]	μA
$I_{DD\_DEEPSLEEP}$	深度睡眠模式	-	未定[a]	未定[a]	μA

a. 特性未定

19.1.5 Flash 存储器的特性

表 19.5 Flash 存储器的特性

参数	参数名称	最小值	额定值	最大值	单位
PE_{CYC}	在故障前已保证的编程/擦除周期数[a]	10,000	-	-	周期
T_{RET}	平均工作温度为 85°C 下的数据保持时间	10	-	-	年
T_{PROG}	字编程时间	20	-	-	μs
T_{ERASE}	页擦除时间	20	-	-	ms
T_{ME}	批量（mass）擦除时间	200	-	-	ms

a. 一个编程/擦除周期被定义成位从 1→0→1 的转换

19.2 交流（AC）特性

19.2.1 加载条件

除非特定说明，否则下列条件对所有时间测量来说都是正确的。时序值都是在 4-mA 驱动强度下测量得到的。

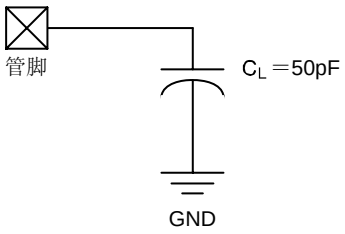


图 19.1 加载条件

19.2.2 时钟

表 19.6 锁相环（PLL）特性

参数	参数名称	最小值	额定值	最大值	单位
$f_{REF\_CRYSTAL}$	晶体参考频率 [a]	3.579545	-	8.192	MHz
f_{REF_EXT}	外部时钟参考频率 [a]	3.579545	-	8.192	MHz
f_{PLL}	PLL 频率 [b]	-	200	-	MHz
T_{READY}	PLL 锁定时间	-	-	0.5	ms

a. 精确值由**运行模式时钟配置（RCC）**寄存器的 XTAL 字段设置的晶体值决定。

b. PLL 频率由基于 **RCC** 寄存器的 XTAL 字段的硬件自动算出。

表 19.7 时钟特性

参数	参数名称	最小值	额定值	最大值	单位
f_{IOSC}	内部振荡器频率	7	15	22	MHz
f_{MOSC}	主振荡器频率	1	-	8	MHz
t_{MOSC_PER}	主振荡器周期	125	-	1000	ns
$f_{REF\_CRYSTAL\_BYPASS}$	使用主振荡器的晶体参考频率(PLL 处于旁路（BYPASS）模式)[a]	1	-	8	MHz
$f_{REF\_EXT\_BYPASS}$	外部时钟参考频率(PLL 处于旁路（BYPASS）模式)[a]	0	-	50	MHz
f_{SYSTEM_CLOCK}	系统时钟	0	-	50	MHz

a) 为了正确工作，ADC 时钟必须由 PLL 提供，或者直接由 14MHz～18MHz 的时钟源提供。

19.2.3 模数转换器

表 19.8 ADC 特性

参数	参数名称	最小值	额定值	最大值	单位
V_{ADCIN}	最大单端、满量程的模拟输入电压	-	-	3.0	V
	最大单端、满量程的模拟输入电压	-	-	0	V
	最大微分、满量程的模拟输入电压	-	-	1.5	V
	最大微分、满量程模拟输入电压	-	-	－1.5	V
C_{ADCIN}	等效输入电容	-	1	-	pF
N	分辨率	-	10	-	bits
f_{ADC}	ADC 内部时钟频率				MHz
$t_{ADCCONV}$	转换时间	-	-	16	t_{ADC} 周期 [a]
$f_{ADCCONV}$	转换率				k 采样/秒
INL	积分非线性	-	-	±1	LSB
DNL	微分非线性	-	-	±1	LSB
OFF	偏移量	-	-	＋2	LSB
GAIN	增益	-	-	±2	LSB

a) $t_{ADC}＝1/f_{ADCclock}$

19.2.4 模拟比较器

表 19.9 模拟比较器特性

参数	参数名称	最小值	额定值	最大值	单位
V_{OS}	输入偏移电压	-	±10	±25	mV
V_{CM}	输入共模电压范围	0	-	V_{DD}-1.5	V
C_{MRR}	共模抑制比（rejection ratio）	50	-	-	dB
T_{RT}	响应时间	-	-	1	μs
T_{MC}	比较器模式变为输出有效的时间	-	-	10	μs

表 19.10 模拟比较器的电压参考特性

参数	参数名称	最小值	额定值	最大值	单位
R_{HR}	高量程（high range）分辨率	-	$V_{DD}/32$	-	LSB
R_{LR}	低量程（low range）分辨率	-	$V_{DD}/24$	-	LSB
A_{HR}	高量程（high range）绝对精度	-	-	±1/2	LSB
A_{LR}	低量程（low range）绝对精度	-	-	±1/4	LSB

19.2.5 同步串行接口（SSI）

表 19.11 SSI 特性

参数编号	参数	参数名称	最小值	额定值	最大值	单位
S1	t_{CLK_PER}	SSIClk 周期时间	2	-	65024	系统时钟
S2	t_{CLK_HIGH}	SSIClk 高电平时间	-	1/2	-	t_{CLK_PER}
S3	t_{CLK_LOW}	SSIClk 低电平时间	-	1/2	-	t_{CLK_PER}
S4	t_{CLKRF}	SSIClk 上升/下降时间	-	7.4	26	ns
S5	t_{DMD}	主机数据的有效延迟时间	0	-	20	ns
S6	t_{DMS}	主机数据的建立时间	20	-	-	ns
S7	t_{DMH}	主机数据的保持时间	40	-	-	ns
S8	t_{DSS}	从机数据的建立时间	20	-	-	ns
S9	t_{DSH}	从机数据的保持时间	40	-	-	ns

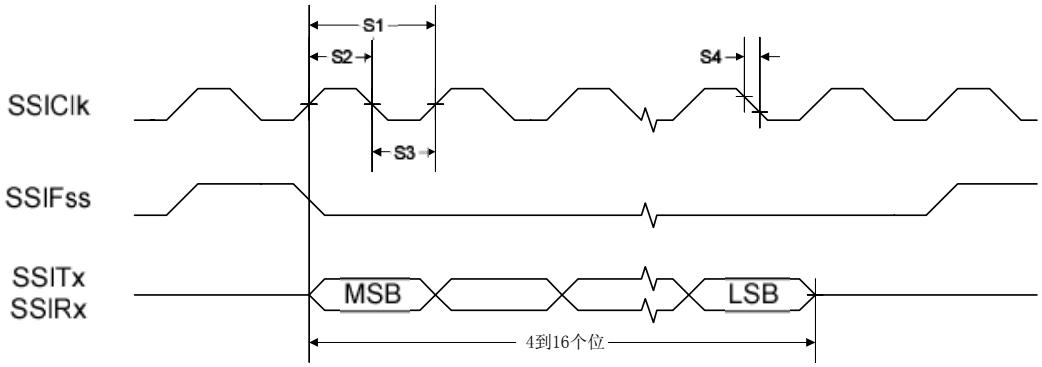


图 19.2 TI 帧格式（FRF＝01）的 SSI 时序，单次传输的时间测量

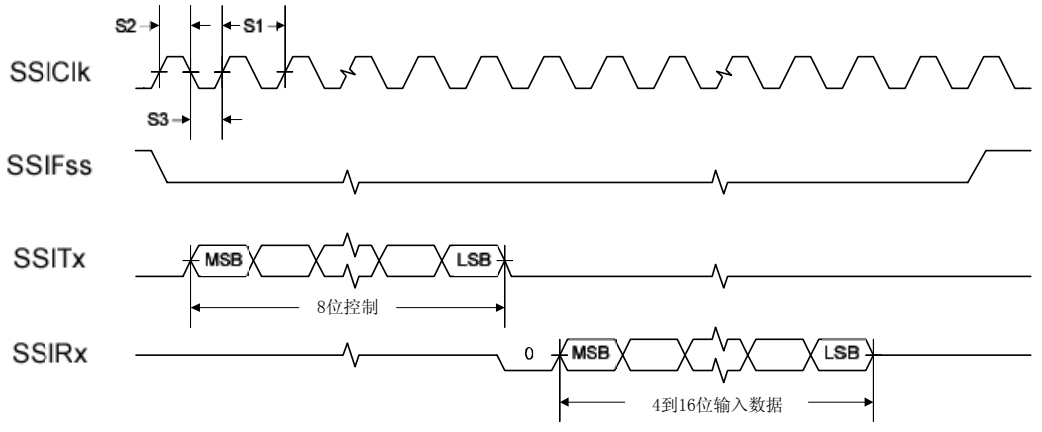


图 19.3 MICROWIRE 帧格式的 SSI 时序（FRF＝10），单次传输

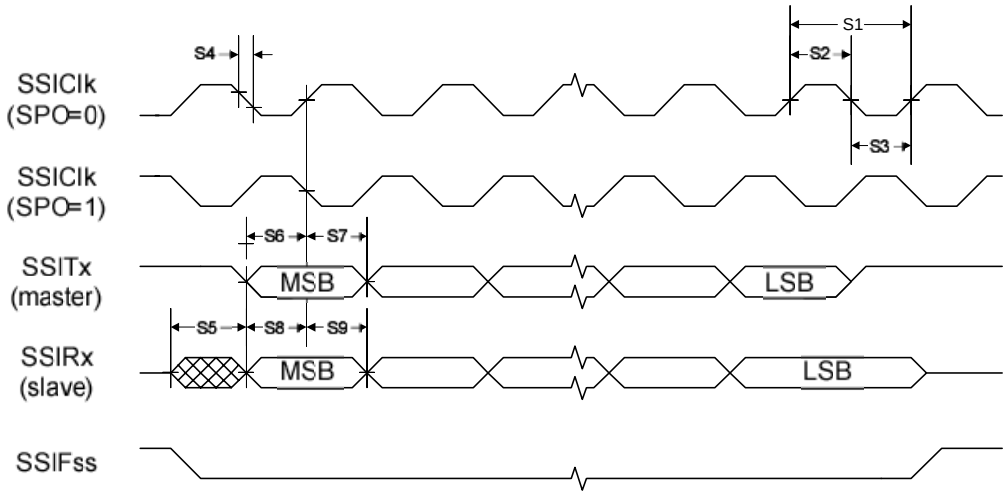


图 19.4 SPI 帧格式的 SSI 时序（FRF＝00），SPH＝1

19.2.6 JTAG 和边界扫描

表 19.12 JTAG 特性

参数编号	参数	参数名称	最小值	额定值	最大值	单位
J1	f_{TCK}	TCK 工作时钟频率	0	-	10	MHz
J2	t_{TCK}	TCK 工作时钟周期	100	-	-	ns
J3	t_{TCK_LOW}	TCK 时钟低电平时间	-	½ t_{TCK}	-	ns
J4	t_{TCK_HIGH}	TCK 时钟高电平时间	-	½ t_{TCK}	-	ns
J5	t_{TCK_R}	TCK 上升时间	0	-	10	ns
J6	t_{TCK_F}	TCK 下降时间	0	-	10	ns
J7	t_{TMS_SU}	到 TCK 上升沿为止，TMS 信号的建立时间	20	-	-	ns
J8	t_{TMS_HLD}	从 TCK 上升沿开始计算，TMS 信号的保持时间	20	-	-	ns
J9	t_{TDI_SU}	到 TCK 上升沿为止，TDI 信号的建立时间	25	-	-	ns
J10	t_{TDI_HLD}	从 TCK 上升沿开始计算，TDI 信号的保持时间	25	-	-	ns
J11 t_{TDO_ZDV}	从 TCK 下降沿开始计算，直到 TDO 从高阻状态变为数据有效之间的时间	2-mA 驱动	-	23	35	ns
		4-mA 驱动		15	26	ns
		8-mA 驱动		14	25	ns
		带斜率控制的 8-mA 驱动		18	29	ns

续上表

参数编号	参数	参数名称	最小值	额定值	最大值	单位
J12 t_{TDO_DV}	从 TCK 下降沿开始计算，直到 TDO 从当前数据有效转到下一个有效数据之间的时间	2-mA 驱动	-	21	35	ns
		4-mA 驱动		14	25	ns
		8-mA 驱动		13	24	ns
		带有斜率控制的 8-mA 驱动		18	28	ns
J13 t_{TDO_DVZ}	从 TCK 下降沿开始计算，直到 TDO 从数据有效变为高阻状态之间的时间	2-mA 驱动	-	9	11	ns
		4-mA 驱动		7	9	ns
		8-mA 驱动		6	8	ns
		带有斜率控制的 8-mA 驱动		7	9	ns
J14	t_{TRST}	$\overline{TRST}$ 有效的时间	100	-	-	ns
J15	t_{TRST_SU}	$\overline{TRST}$ 建立到 TCK 上升的时间	10	-	-	ns

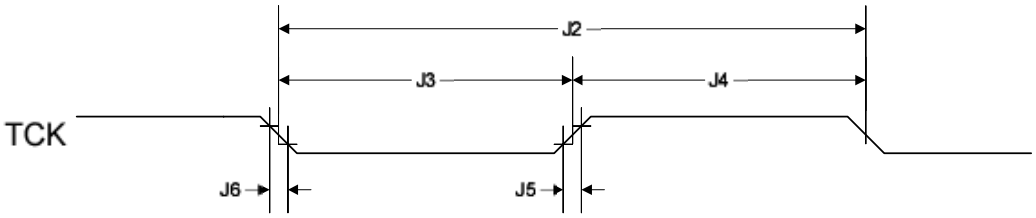


图 19.5 JTAG 测试时钟的输入时序

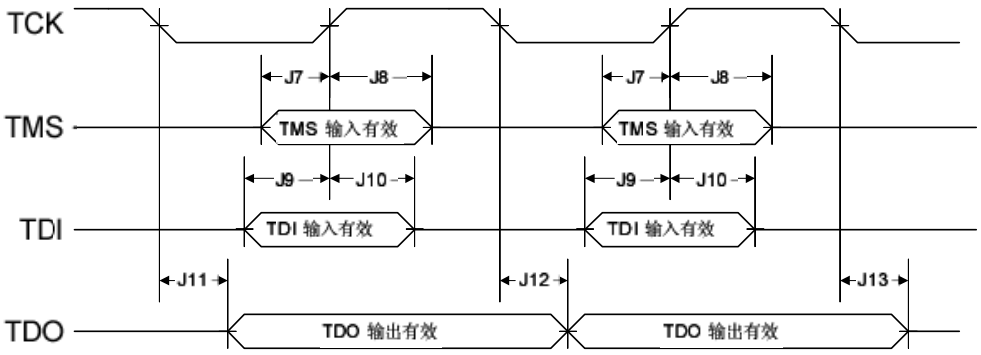


图 19.6 JTAG 测试访问端口（TAP）的时序

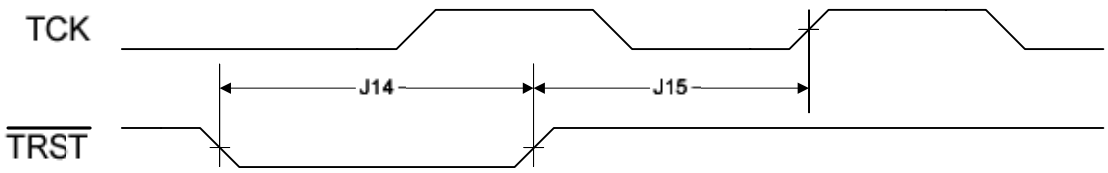


图 19.7 JTAG $\overline{TRST}$ 时序

19.2.7 通用 I/O 口

表 19.13 GPIO 特性 [a]

参数	参数名称	条件	最小值	额定值	最大值	单位
t_{GPIOR}	GPO 上升时间 (从 20% V_{DD} 到 80%V_{DD})	2-mA 驱动	-	17	26	ns
		4-mA 驱动		9	13	ns
		8-mA 驱动		6	9	ns
		带有斜率控制的 8-mA 驱动		10	12	ns
t_{GPIOF}	GPO 下降时间 (从 80%V_{DD} 到 20%V_{DD})	2-mA 驱动	-	17	25	ns
		4-mA 驱动		8	12	ns
		8-mA 驱动		6	10	ns
		带有斜率控制的 8-mA 驱动		11	13	ns

a 所有 GPIO 最大都可承受 5V

19.2.8 复位

表 19.14 复位特性

参数编号	参数	参数名称	最小值	额定值	最大值	单位
R1	V_{TH}	复位阈值（threshold）电压	-	2.0	-	V
R2	V_{BTH}	掉电（brown-out）阈值电压	2.85	2.9	2.95	V
R3	T_{POR}	上电复位超时	-	10	-	ms
R4	T_{BOR}	掉电（brown-out）超时	-	500	-	μs
R5	T_{IRPOR}	上电复位（POR）之后内部复位的超时	15	-	30	ms
R6	T_{IRBOR}	掉电复位（BOR）之后内部复位的超时 [a]	2.5	-	20	μs
R7	T_{IRHWR}	硬件复位（$\overline{RST}$ 管脚）之后内部复位的超时	15	-	30	ms
R8	T_{IRSWR}	软件初始化系统复位之后内部复位的超时 [a]	2.5	-	20	μs
R9	T_{IRWDR}	看门狗复位之后内部复位的超时 [a]	2.5	-	20	μs
R10	T_{IRLDOR}	LDO 复位之后内部复位的超时 [a]	2.5	-	20	μs
R11	$T_{VDDRISE}$	电源电压(V_{DD})上升(0V-3.3V)时间			100	ms

a $20*t_{MOSC\_PER}$

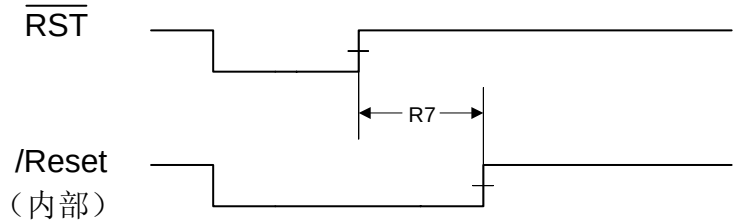


图 19.8 外部复位时序

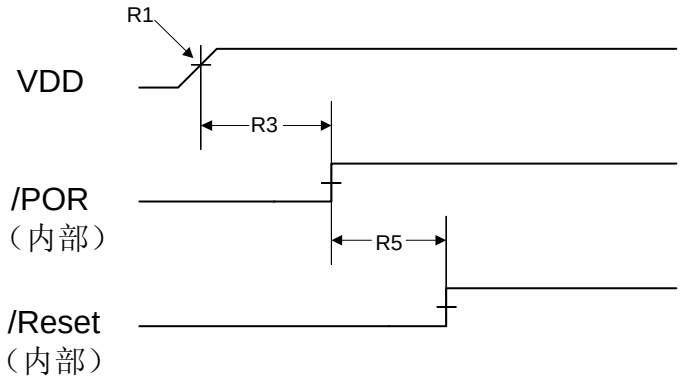


图 19.9 上电复位时序

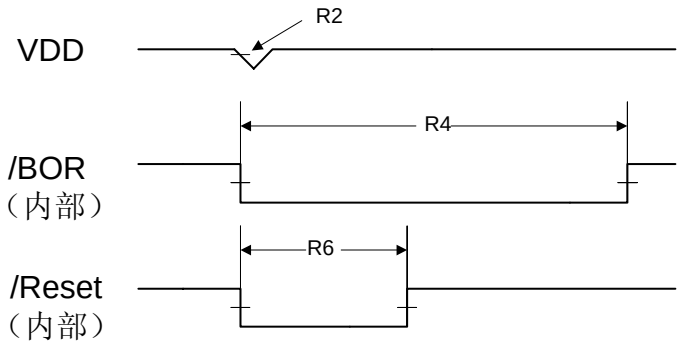


图 19.10 掉电（brown-out）复位时序

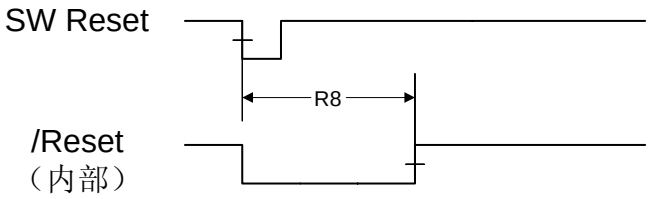


图 19.11 软件复位时序

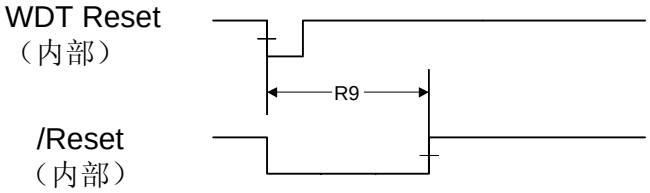


图 19.12 看门狗复位时序

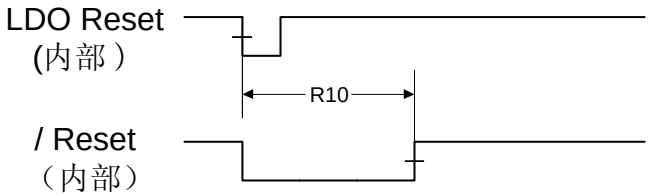
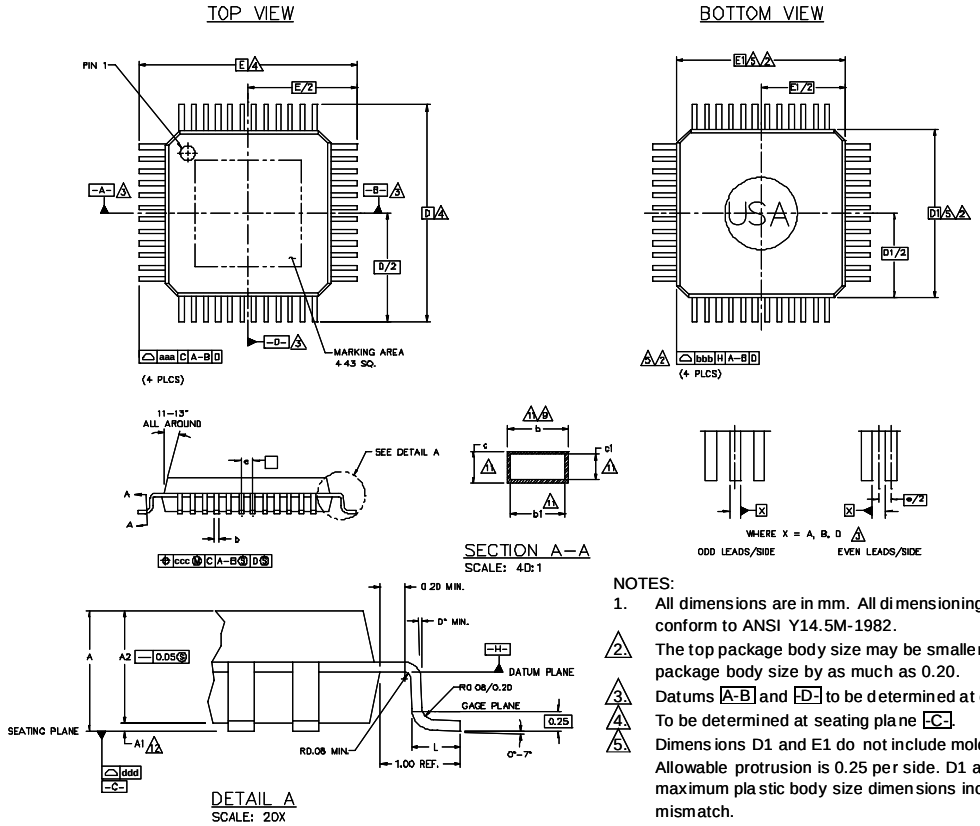


图 19.13 LDO 复位时序

第20章 封装信息



SYMBOL	LEAD COUNT; FOOT PRINT			NOTE
	48, 2.0 FP			
	MIN	NOM	MAX	
A	===	===	1.60	
A1	0.05	===	0.15	
A2	1.35	1.40	1.45	
D	9.00 BSC			
D1	7.00 BSC			
E	9.00 BSC			
E1	7.00 BSC			
L	0.45	0.80	0.75	
e	0.50 BSC			
b	0.17	0.22	0.27	
b1	0.17	0.20	0.23	
c	0.09	===	0.20	
c1	0.09	===	0.16	
	Tolerances of form and position			
aaa	0.20			
bbb	0.20			
ccc	0.08			
ddd	0.08			

NOTES:

1. All dimensions are in mm. All dimensioning and tolerancing conform to ANSI Y14.5M-1982.
2. The top package body size may be smaller than the bottom package body size by as much as 0.20.
3. Datums A-B and -D- to be determined at datum plane -H-.
4. To be determined at seating plane -C-.
5. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 per side. D1 and E1 are maximum plastic body size dimensions including mold mismatch.
6. Surface finish of the package is #24-27 Charmille (1.6-2.3 mR_0) Pin 1 and ejector pin may be less than 0.1 mR_0.
7. Dambar removal protrusion does not exceed 0.08. Intrusion does not exceed 0.03.
8. Burr does not exceed 0.08 in any direction.
9. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall not cause the lead width to exceed the maximum b dimension by more than 0.08. Dambar cannot be located on the lower radius or the foot. Minimum space between protrusion and adjacent lead is 0.07 for 0.40 and 0.50 pitch package.
10. Corner radius of plastic body does not exceed 0.20.
11. These dimensions apply to the flat section of the lead between 0.10 and 0.25 from the lead tip.
12. A1 is defined as the distance from the seating plane to the lowest point of the package body.
13. Finish of leads is tin plated.
14. All specifications and dimensions are subjected to IPAC'S manufacturing process flow and materials.
15. The packages described in the drawing conform to JEDEC M5-026A. Where discrepancies between the JEDEC and IPAC documents exist, this drawing will take the precedence.

图 20.1 48 脚 LQFP 封装

附录A 串行 Flash 加载程序

Stellaris 串行 flash 加载程序可在无需使用调试接口的情况下将代码下载到器件的 flash 存储器中。该串行加载程序使用一个简单的信息包（packet）接口来提供与器件的同步通信。它使用晶振而不使能 PLL。因此，其速度由使用的晶体决定。它可使用两个串行接口：UART0 和 SSI 接口。为简单起见，这两种串行接口的数据格式和通信协议都相同。

A.1 接口

一旦通过其中一个串行接口确立了与 flash 加载程序的通信，该接口将一直处于使用状态，直到 flash 加载程序被复位或者新代码获得控制权。例如，一旦用户启动了使用 SSI 端口的通信，通过 UART 与 flash 加载程序的通信就会被禁止，直到将器件复位。

A.1.1 UART

通用异步收发器（UART）的通信使用一个固定的串行格式：8 个数据位、无奇偶校验位和 1 个停止位。通信所使用的波特率由 flash 加载程序自动检测，它可以是主机和器件支持的任何有效的波特率。自动检测序列要求波特率不能超过运行该加载程序的电路板所具有的晶振频率的 1/32。这实际上与 stellaris 器件上的任何 UART 的最大波特率硬件限制是相同的。

为了确定波特率，串行 flash 加载程序需确定其自身的晶振频率与波特率之间的关系。知道了这两者的关系，flash 加载程序就可以将其 UART 配置为与主机的波特率相同。因为它的自动波特率检测功能允许主机使用任何有效的波特率来与器件进行通信。

在执行自动同步功能时需依靠主机向 flash 加载程序发送两个都为 0x55 的字节。这样可向 flash 加载程序产生一系列脉冲，这些脉冲可用于计算对 UART 进行编程以与主机的波特率相同时所需的比率。主机在发送了用于同步的格式之后，它将试图从 UART 中读回一个数据字节。Flash 加载程序返回 0xCC 来表示波特率检测成功。在至少经过了传输两个字节所需的两倍时间之后如果仍没有接收到该字节，则主机将重新发送两个字节的 0x55，并再次等待 0xCC 字节，直到 flash 加载程序作出已正确接收到该同步格式的应答。例如，等待从 flash 加载程序返回数据所需的时间至少应该为 2*(20(b/sync)/波特率(b/s))。当波特率为 115200 时，该时间为 2*(20/115200)，即 0.35ns。

A.1.2 SSI

同步串行接口（SSI）的端口也使用一个固定的串行格式进行通信，它所使用的帧被定义为 Motorola 格式，即 SPH 设为 1，SPO 设为 1。有关该传输协议的详细描述请见 SSI 章的 SSI 格式部分。与 UART 类似，该接口对硬件也有一定的要求，以限制 SSI 时钟能够运行的最大速率。它允许的 SSI 最大时钟为运行 flash 加载程序的电路板所具有的晶振频率的 1/12。主器件作为通信的主机，因此，时钟直接由主机提供，运行 flash 加载程序的器件上的 SSI 不需要确定时钟。

A.2 信息包处理

所有通信，除了 UART 自动波特率以外，均通过已定义的信息包来完成，这些信息包由器件作出应答（ACK）或非应答（NAK）。接收和发送信息包时使用的格式是相同的，

包括对信息包的成功接收和没有成功接收作出应答时所使用的方法也是相同的。

A.2.1 信息包格式

所有从器件中发送和接收的信息包均使用下面的字节包装格式：

```
struct
{
unsigned char ucSize;
unsigned char ucCheckSum;
unsigned char Data[];
};
```

ucSize：接收到的第一个字节，它含有的是包括数据长度（size）和校验和字节在内的总的传输长度。

unChecksum：这是只对数据缓冲区中的字节计算而得的校验和。该算法为：Data[0]+Data[1]+…+Data[ucSize-3]。

Data：这是器件需要的原始数据，它在某种格式的命令接口中进行格式化。该数据缓冲区应向器件发送或从器件中接收 ucSize-2 个数据字节。

A.2.2 信息包发送

实际的信息包字节可以单独发送也可以一起发送。发送操作只有一个限制，即引起 flash 存储器访问的命令应限制下载的容量（size）以避免在 flash 编程过程中丢失字节。该限制在后面与 flash 相作用的命令中将会进一步讨论。

一旦主机将信息包进行了正确的格式化后，它就可以通过 UART 或 SSI 接口发送出去。然后，主机应查询 UART 或 SSI 接口，来等待从器件中返回的第一个非零数据。该非零字节将是来自器件的 ACK（0xCC）或 NAK（0x33）应答，表示信息包已成功接收（ACK）还是没有成功接收（NAK）。该应答并不表示位于信息包数据部分的已发送命令的实际内容是有效的，而只表示信息包已成功接收。

A.2.3 信息包接收

Flash 加载程序以接收信息包的相同的格式来发送一个数据包。该加载程序在发送第一个实际的数据字节之前可以先传输几个零。第一个非零字节为信息包的长度，后面跟着的是校验和字节，最后跟着的才是数据本身。在从 flash 加载程序中发送了第一个非零字节之后，数据之间就没有了间隔。一旦与 flash 加载程序进行通信的器件接收了所有字节，它就必须对信息包作出 ACK 或 NAK 应答以表示发送是否成功。如果该器件向 flash 加载程序发送的是 NAK 应答，则后面它要重新发送失败的命令并再次请求数据。如有必要，主机在向 flash 加载程序发送 ACK/NAK 信号之前可先发送零。flash 加载程序只将第一个非零数据看作是有效的响应，因此，为了从 flash 加载程序中接收数据或向其发送数据，上述的填充零操作对于 SSI 接口来说是必需的。

A.3 命令

下面的内容定义了能够发送给 flash 加载程序的命令列表。数据的第一个字节应始终为其中一个已定义的命令，后面跟着的是由被发送命令确定的数据或参数。

A.3.1 COMMAND_PING（0x20）

COMMAND_PING 命令用于简单地接收命令并将全局状态设置为成功。信息包的格式如下：

```
Byte[0] = 0x03;
Byte[1] = checksum(Byte[2]);
Byte[2] = COMMAND_PING;
```

Ping 命令具有 3 个字节，COMMAND_PING 的值为 0x20，一个字节的校验和即为该字节本身，因此，Byte[1]也为 0x20。ping 命令没有实际的返回状态，因此，接收到的 ACK 应答可解释为成功地对 flash 加载程序执行了 ping 操作。

A.3.2 COMMAND_GET_STATUS（0x23）

COMMAND_GET_STATUS 命令返回的是上一次发送的命令的状态。该命令通常在每个命令之后发送，以确保前一个命令发送成功，或者如果失败，则可确保对失败作出正确响应。首先，该命令需要信息包数据中的一个字节，然后是对带有一个字节数据（该数据中包含一个状态代码）的信息包进行读操作。最后一步是对接收到的数据作出 ACK 或 NAK 应答，以便 flash 加载程序能够知道数据已读取。

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.3.3 COMMAND_DOWNLOAD（0x21）

将 COMMAND_DOWNLOAD 命令发送给 flash 加载程序可指示在何处存储数据以及在 COMMAND_SEND_DATA 命令之后应发送多少个字节。该命令由两个均先传输 MSB 的 32 位值组成。第一个 32 位值是对数据进行编程的起始地址，而第二个 32 位值是即将被发送的数据的字节数。该命令还可对即将进行编程的整个区域触发一次擦除操作。因此，该命令所花的时间比其它命令要长。这也导致要花更长的时间接收电路板的 ACK/NAK 应答。该命令后应跟着 COMMAND_GET_STATUS，以确保编程地址和编程容量对于运行 flash 加载程序的器件来说是有效的。

发送该命令的包格式如下：

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_DOWNLOAD
Byte[3] = Program Address [31:24]
Byte[4] = Program Address [23:16]
Byte[5] = Program Address [15:8]
Byte[6] = Program Address [7:0]
Byte[7] = Program Size [31:24]
Byte[8] = Program Size [23:16]
Byte[9] = Program Size [15:8]
```

Byte[10] = Program Size [7:0]

A.3.4 COMMAND_SEND_DATA（0x24）

COMMAND_SEND_DATA 命令的后面只能跟着 COMMAND_DOWNLOAD 命令，或者如果需要更多数据，也可跟着另一个 COMMAND_SEND_DATA 命令。连续的数据发送命令将自动使地址加 1 并从之前的位置继续编程。调用者应对数据传输进行限制，即把信息包的数据限制为最多 8 个字节，以便成功地对 flash 进行编程，并且不会造成串口输入缓冲区的溢出。一旦已成功接收到由 COMMAND_DOWNLOAD 命令指示的字节数，该命令就会结束编程。每次调用该功能时，后面应跟着一个 COMMAND_GET_STATUS，以保证将数据成功地编程到 flash 中。如果 flash 加载程序向该命令发送一个 NAK 应答，则 flash 加载程序不会将当前地址加 1，以便重新发送之前的数据。

Byte[0] = 11

Byte[1] = checksum(Bytes[2:10])

Byte[2] = COMMAND_SEND_DATA

Byte[3] = Data[0]

Byte[4] = Data[1]

Byte[5] = Data[2]

Byte[6] = Data[3]

Byte[7] = Data[4]

Byte[8] = Data[5]

Byte[9] = Data[6]

Byte[10] = Data[7]

A.3.5 COMMAND_RUN（0x22）

COMMAND_RUN 命令通知 flash 加载程序从作为该命令中的参数进行传递的地址处执行。该命令由一个 32 位值组成，该值被解释为即将执行时的地址。它以 MSB 在前的方式发送，并且在给定的地址处实际执行代码之前，flash 加载程序会向主机返回一个 ACK 信号作为应答。这使得主机能够知道命令已成功接收，并且现在运行代码。

Byte[0] = 7

Byte[1] = checksum(Bytes[2:6])

Byte[2] = COMMAND_RUN

Byte[3] = Execute Address[31:24]

Byte[4] = Execute Address[23:16]

Byte[5] = Execute Address[15:8]

Byte[6] = Execute Address[7:0]

A.3.6 COMMAND_RESET(0x25)

COMMAND_RESET 命令将运行 flash 加载程序的器件复位。这在下载一个将 flash 加载程序覆盖掉的新映象（imgae）以及希望从一个完全复位的状态启动时是非常有用的。该命令不同于 COMMAND_RUN 命令，它允许由硬件读取初始堆栈指针以及为新代码设立初

始堆栈指针。如果出现严重错误以及主机希望重新启动与 flash 加载程序的通信，则该命令也可用于将 flash 加载程序复位。

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

在对运行 flash 加载程序的器件实际执行软件复位之前，Flash 加载程序会向主机返回一个 ACK 信号作为应答。这使得主机能够知道命令已成功接收，器件将被复位。

附录B 版本信息

修订版本	修订日期	描述
Rev .00	2007 年 4 月 1 日	原始版本