LUMINARY MICRO™

LM3S2950 微控制器

数据手册

DS-LM3S2950-1972

法律声明和商标信息

本文档提供了LUMINARY MICRO产品的相关信息。对于本文档的任何知识产权，不能以许可、明示、暗示、被禁止的或其它的任何形式授权。除了LUMINARY MICRO产品的销售条款所提供的信息外，LUMINARY MICRO不承担任何责任。LUMINARY MICRO声明有关LUMINARY MICRO产品销售和/或使用的任何明示或暗示的保证，包括关于对特殊用途的适用性、商销性或任何专利、版权或其它知识产权侵权行为的责任或保证。 LUMINARY MICRO的产品不用于医疗、救治或维持生命的应用。

Luminary Micro可以在没有通知的情况下随时更改说明规范和产品描述。在订购您的产品之前，联系您当地的Luminary Micro销售办公室或分销商来获得最新的说明规范。

设计者不必理会标记了“保留”或“未定义”的任何特性或说明的存在性或特殊性。Luminary Micro 保留这些特性以用于将来定义，关于将来对这些特性的更改所引起的冲突或不兼容，Luminary Micro 不具有任何责任。

Luminary Micro公司
108 Wild Basin, Suite 350
Austin, TX 78746
主要联系电话：+1-512-279-8800
传真：+1-512-279-8879
http://www.luminarymicro.com

LUMINARY MICRO™

目录

插图清单

表格清单

寄存器列表

关于本文档

该数据手册提供 LM3S2950 微控制器的参考信息，描述围绕 ARM® Cortex™-M3 内核设计的片上系统（SoC）器件的功能模块。

读者

本手册的受众是系统软件开发人员、硬件设计人员和应用设计人员。

关于本手册

本文档分成多个章节，每个章节与主要的特性相对应。

相关文档

以下文档作为该数据手册的参考文档，可从CD或Luminary Micro网站www.luminarymicro.com中获得：

- *ARM® Cortex™-M3 技术参考手册*
- *ARM® CoreSight 技术参考手册*
- *ARM® v7-M 结构应用层参考手册*

本数据手册也可参考下列相关文档：

- *IEEE 标准 1149.1-测试访问端口和边界扫描结构*

这个文献列表的实时性根据发布的日期来判断。包括应用笔记和白皮书在内的其它文献请登陆Luminary Micro网站核对。

文档约定

本文档使用的约定如表 1 在 20页所示。

表 **1.** 文档约定

表示法	含义
通用寄存器的表示法	
寄存器	APB寄存器用大写的粗体表示。例如，**PBORCTL**是上电和掉电复位控制寄存器。如果一个寄存器名称包含一个小写的n，它就代表了多个寄存器。例如，**SRCRn**代表了下面3个软件复位控制寄存器的任何一个或全部： **SRCR0, SRCR1** 和**SRCR2**。
位	寄存器的一个位。
位域	2个或更多连续和相关联的位。
偏移量 0x*nnn*	寄存器地址的一个十六进制增量，增量是相对“存储器映射” 在 39页中指定的模块基址而言的。
寄存器*N*	为了方便引用，在整篇文档中，寄存器被顺序编号。寄存器编号对软件没有意义。
保留	标注*保留*的寄存器位保留下来供将来使用。在大多数情况下，保留位被设置成0；但是，用户软件不应当依赖保留位的值。为了使软件兼容未来的器件，保留位的值应当在读-修改-写过程中保留下来。
yy:xx	寄存器位的范围从xx到yy（xx和yy包括在内）。例如，31:15表示相应寄存器的位15到31。
寄存器 位/域 类型	寄存器位框图中的这个值指明了在控制器上运行的软件是否能改变位域的值。
RC	软件可以读取这个域。位/域在被读取之后由硬件清零。
RO	软件可以读取这个域。始终写入芯片复位值。

表示法	含义
R/W	软件可以读或写这个域。
R/W1C	软件可以读或写这个域。向W1C位写入0不影响寄存器中的位值。写入1清零寄存器中位的值；剩余的位保持变。 这个寄存器类型主要用来清零中断状态位，执行读操作来提供中断状态，写入读取的值只清除在读寄存器时被报告的中断。
W1C	软件可以写这个域。向W1C位写入0不影响寄存器中的位值。写入1清零寄存器中位的值；剩余的位保持变。读寄存器返回的数据没有意义。 这个寄存器通常用来清零中断寄存器中相应的位。
WO	只有软件的写操作才有效；读寄存器返回的数据没有意义。
寄存器 位/域 复位值	寄存器位框图中的这个值是任何复位后的位/域值，但特别注明的除外。
0	芯片复位时位被清零。
1	芯片复位时位被置1。
-	不确定。
管脚/信号表示法	
[ ]	管脚备用功能；管脚默认用作不带方括号的信号。
管脚	参考封装上的物理连接。
信号	参考管脚的电气信号编码。
使一个信号有效	将信号的值从逻辑假状态变为逻辑真状态。对于高电平有效的信号，有效的信号值为1（高）；对于低电平有效的信号，有效的信号值为0（低）。有效极性（高或低）由信号名称来定义（见下面的SIGNAL和SIGNAL）。
使一个信号无效	将信号的值从逻辑真状态变为逻辑假状态。
SIGNAL	信号名称采用大写字母，使用Courier字体。信号名称带有上横线表示该信号低有效。使SIGNAL有效就将其驱动为低电平；使SIGNAL无效就将其驱动为高电平。
SIGNAL	信号名称采用大写字母，使用Courier字体。高有效的信号没有上横线。使SIGNAL有效就将其驱动为高电平；使 SIGNAL无效就将其驱动为低电平。
数值	
X	大写的X表示几个值都是可取的，此处的X可以是任何合法的样式。例如，二进制值0X00可以是0100或0000，十六进制值0xX可以是0x0或0x1，等等。
0x	十六进制值带有前缀0x。例如，0x00FF是十六进制值FF。 寄存器表中的所有其它数假设为二进制。在概念信息中，二进制数用一个b后缀表示，例如，1011b，写十进制数无需前缀或后缀。

1 结构概述

Luminary Micro公司 Stellaris®所提供一系列的微控制器是首款基于ARM® Cortex™-M3的控制器，它们为对成本尤其敏感的嵌入式微控制器应用方案带来了高性能的32位运算能力。 这些具备领先技术的芯片使用户能够以传统的8位和16位器件的价位来享受32位的性能，而且所有型号都是以小占位面积的封装形式提供。

该Stellaris® 系列芯片能够提供高效的性能、广泛的集成功能以及按照要求定位的选择，适用于各种关注成本并明确要求具有的过程控制以及连接能力的应用方案。 该Stellaris® LM3S1000 系列使用更大的片上存储器、增强型电源管理和扩展I/O以及控制功能来扩展Stellaris®家族。 该Stellaris® LM3S2000 系列是针对控制器局域网（CAN）应用方案而设计的一组芯片，它在Stellaris系列芯片的基础上扩展了Bosch CAN网络技术——短距离工业网络里的黄金标准。 该Stellaris® LM3S2000 系列芯片还标志着先进的Cortex-M3内核和CAN能力的首次结合运用。 该Stellaris® LM3S6000 系列芯片结合了10/100以太网媒体访问控制（MAC）以及物理层（PHY），标志着ARM Cortex-M3 MCU已经具备集成连接能力，还是唯一集成了10/100以太网MAC和PHY物理层的ARM架构MCU。 该Stellaris® LM3S8000 系列结合了 Bosch 控制器局域网技术和 10/100 以太网媒体访问控制(MAC)以及物理 (PHY) 层。

该LM3S2950 微控制器是针对工业应用方案而设计的，包括远程监控、电子贩售机、测试和测量设备、网络设备和交换机、工厂自动化、HVAC和建筑控制、游戏设备、运动控制、医疗器械、以及火警安防。

至于那些对功耗有特别要求的应用方案，LM3S2950微控制器还具有一个电池备用的休眠模块，从而有效的使LM3S2950芯片在未被激活的时候进入低功耗状态。 一个上电/掉电序列发生器、连续的时间计数器（RTC）、一对匹配寄存器、一个到系统总线的APB接口以及专用的非易失性存储器、休眠模块等功能组件使LM3S2950微控制器极其适合用在电池的应用中。

除此之外，该LM3S2950微控制器的优势还在于能够方便的运用多种ARM的开发工具和片上系统（SoC）的底层IP应用方案，以及广大的用户群体。 另外，该微控制器使用了兼容ARM的Thumb®指令集的Thumb2指令集来减少存储容量的需求，并以此达到降低成本的目的。 最后，LM3S2950微控制器与Stellaris®系列的所有成员是代码兼容的，这为用户提供了灵活性，能够适应各种精确的需求。

为了能够帮助用户产品快速的上市，Luminary Micro公司提供了一整套的解决方案，包括评估和开发用的板卡、白皮书和应用笔记 、方便使用的外设驱动程序库、以及强劲的支持、销售和分销网络。

1.1 产品特性

该LM3S2950微控制器包括下列的产品特性：

- 32位RISC性能
  - 采用为小封装应用方案而优化的 32位ARM® Cortex™-M3 v7M架构。
  - 提供系统时钟、包括一个简单的24位写清零、递减、自装载计数器，同时具有灵活的控制机制
  - 仅采用与Thumb®兼容的Thumb-2指令集以获取更高的代码密度
  - 工作频率为50-MHz
  - 硬件除法和单周期乘法
  - 集成嵌套向量中断控制器（NVIC），使中断的处理更为简捷

- 36 中断具有8个优先等级
- 带存储器保护单元（MPU），提供特权模式来保护操作系统的功能
- 非对齐式数据访问，使数据能够更为有效的安置到存储器中
- 精确的位操作（bit-banding），不仅最大限度的利用了存储器空间而且还改良了对外设的控制

■ 内部存储器

- 256 KB单周期Flash
  - 可由用户管理 对flash块的保护，以2KB为单位
  - 可由用户管理对flash的编程
  - 可由用户定义和管理的flash保护块
- 64 KB单周期访问的SRAM

■ 通用定时器

- 4个通用定时器模块(GPTM)，每个提供2个16-位定时器。 每个 GPTM 可被独立配置进行操作:
  - 作为一个32位定时器
  - 作为一个32位的实时时钟（RTC）来捕获事件
  - 用于脉宽调解器（PWM）
- 32位定时器模式
  - 可编程单次触发定时器
  - 可编程周期定时器
  - 当接入32.768-KHz外部时钟输入时可作为实时时钟使用
  - 在调试期间，当控制器发出CPU暂停标志时，在周期和单次触发模式中用户可以使能中止。
- 16位定时器模式
  - 通用定时器功能，并带一个8位的预分频器
  - 可编程单次触发定时器
  - 可编程周期定时器
  - 在调试的时候，当控制器发出CPU暂停标志时，用户可设定暂停周期或者单次模式下的计数
- 16位输入捕获模式
  - 提供输入边沿计数捕获功能

    - 提供输入边沿时间捕获功能
  - 16位PWM模式
    - 简单的PWM模式，对PWM信号输出的取反可由软件编程决定
- 兼容ARM FiRM的看门狗定时器
  - 32位向下计数器，带可编程的装载寄存器
  - 带使能功能的独立看门狗时钟
  - 带中断屏蔽功能的可编程中断产生逻辑
  - 软件跑飞时可锁定寄存器以提供保护
  - 带使能/禁能的复位产生逻辑
  - 在调试的时候，当控制器发出CPU暂停标志时，用户可以设定暂停定时器的周期
- CAN
  - 支持CAN协议版本2.0 part A/B
  - 传输位速率可达1Mb/s
  - 32个消息对象，每个都带有独立的标识符屏蔽
  - 可屏蔽的中断
  - 可禁止TTCAN的自动重发模式
  - 可编程设定的自循环自检操作
- 同步串行接口（SSI）
  - 两个SSI模块，每个都具有以下的特点：
  - 主机或者从机方式运作
  - 可编程控制的时钟位速率和预分频
  - 独立的发送和接收FIFO，8X16位宽的深度
  - 可编程控制的接口，可与Freescale的SPI接口，MICROWIRE或者TI器件的同步串行接口相连
  - 可编程决定数据帧大小，范围为4到16位
  - 内部循环自检模式可用于诊断/调试
- UART
  - 3个完全可编程的16C550-type UART，支持IrDA
  - 带有独立的16x8发送（TX）以及16x12接收（RX）FIFO，可减轻CPU中断服务的负担

- 可编程的波特率产生器，并带有分频器
- 可编程设置FIFO长度，包括1字节深度的操作，以提供传统的双缓冲接口。
- FIFO 触发水平可设为1/8, 1/4, 1/2, 3/4 和 7/8
- 标准异步通信位：开始位、停止位、奇偶位
- 无效起始位检测
- 行中止的产生和检测

- 模拟比较器
  - 3个独立集成的模拟比较器
  - 可以把输出配置为：驱动输出管脚或者产生中断
  - 比较两个外部管脚输入或者将外部管脚输入与内部可编程参考电压相比较
- I^2C
  - 在标准模式下主机和从机接收和发送操作的速度可达100Kbps，在快速模式下可达400Kbps
  - 中断的产生
  - 主机带有仲裁和时钟同步功能、支持多个主机、以及7位寻址模式
- PWM
  - 3个 PWM信号发生模块，每个模块都带有1个16位的计数器、2个比较器, 1个PWM信号发生器、以及一个死区发生器
  - 1个16位的计数器
    - 运行在递减或递增/递减模式
    - 输出频率由一个16位的装载值控制
    - 可同步更新装载值
    - 当计数器的值到达零或者装载值的时候生成输出信号
  - 2个 PWM比较器
    - 比较器值的更新可以同步
    - 在匹配的时候产生输出信号
  - PWM信号发生器
    - 根据计数器和PWM比较器的输出信号来产生PWM输出信号
    - 可产生两个独立的PWM信号
  - 死区发生器

        - 产生2个带有可编程死区延时的PWM信号，适合驱动半H桥（half-H bridge）
        - 可以被旁路，不修改输入PWM信号
    - 灵活的输出控制模块，每个PWM信号都具有PWM输出使能
        - 每个PWM信号都具有PWM输出使能
        - 每个PWM信号都可以选择将输出反相（极性控制）
        - 每个PWM信号都可以选择进行故障处理
        - PWM发生器模块的定时器同步
        - PWM发生器模块的定时器/比较器更新同步
        - PWM发生器模块中断状态被汇总
- QEI
    - 硬件位置积分器追踪编码器的位置
    - 使用内置的定时器进行速率捕获
    - 在出现索引脉冲、速度定时器时间到、方向改变以及检测到正交错误时产生中断
- GPIO
    - 高达10-60个GPIO，具体数目取决于配置
    - 输入/输出可承受5V
    - 中断产生可编程为边沿触发或电平检测
    - 在读和写操作中通过地址线进行位屏蔽
    - GPIO端口配置的可编程控制
        - 弱上拉或下拉电阻
        - 2mA、4mA和8mA端口驱动
        - 8-mA驱动的斜率控制
        - 开漏使能
        - 数字输入使能
- 功率
    - 片内低压差（LDO）稳压器，具有可编程的输出电压，用户可调节的范围为2.25V到2.75V
    - 休眠模块处理3.3V通电/断电序列，并控制内核的数字逻辑和模拟电路
    - 控制器的低功耗模式：睡眠模式和深度睡眠模式
    - 外设的低功耗模式：软件控制单个外设的关断

  - LDO带有检测不可调整电压和自动复位的功能，可由用户控制使能
  - 3.3V电源掉电检测，可通过中断或复位来报告
- 灵活的复位源
  - 上电复位
  - 复位管脚有效
  - 掉电（BOR）检测器向系统发出电源下降的警报
  - 软件复位
  - 看门狗定时器复位
  - 内部低压差（LDO）稳压器输出变为不可调整
- 其他特性
  - 6个复位源
  - 可编程的时钟源控制
  - 可对单个外设的时钟进行选通以节省功耗
  - 遵循IEEE 1149.1-1990标准的测试访问端口（TAP）控制器
  - 通过JTAG和串行线接口进行调试访问
  - 完整的JTAG边界扫描
- 工业范围内遵循RoHS标准的100脚LQFP封装

1.2 目标应用

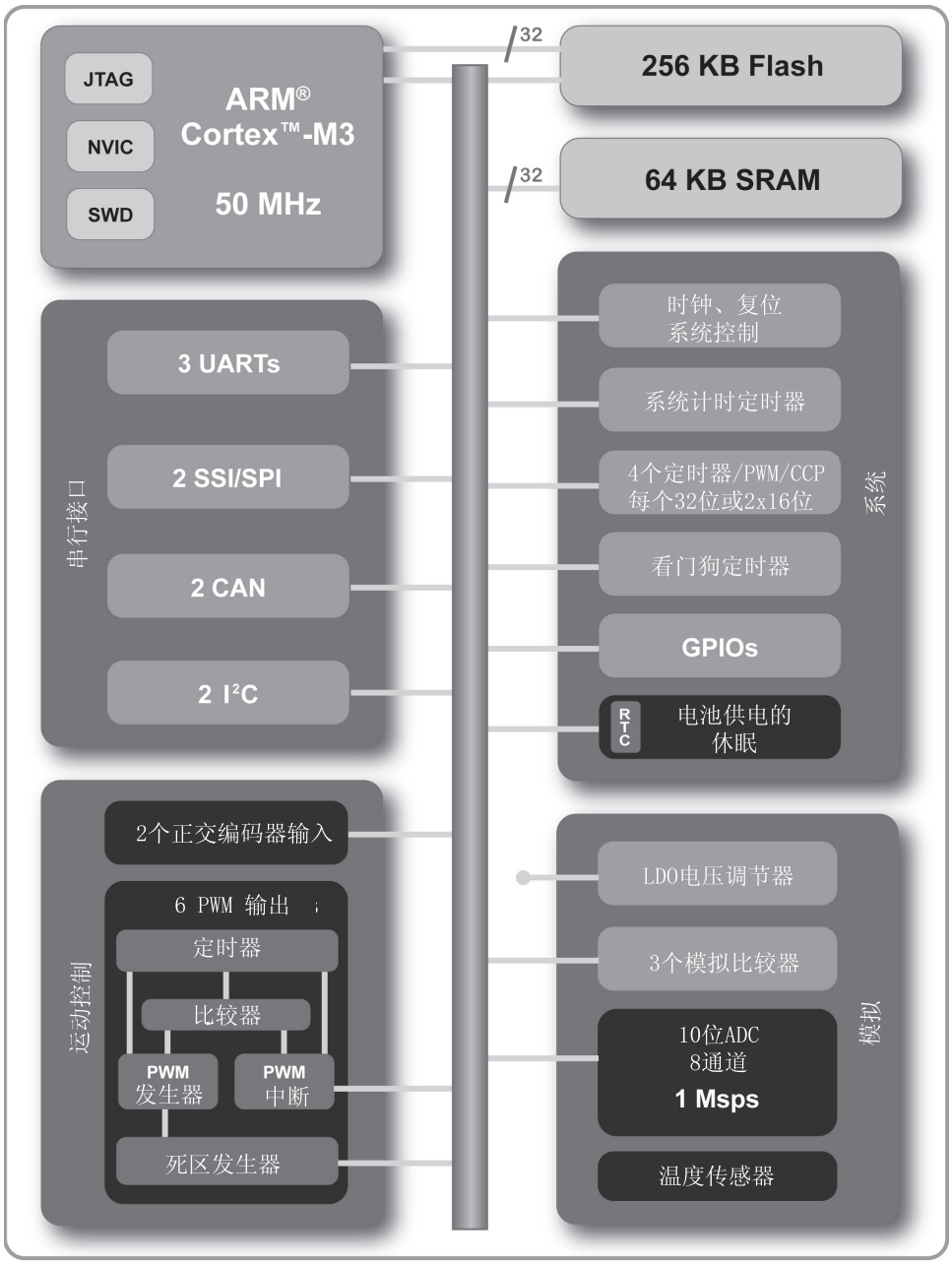
- 远程监控
- 电子销售点机器（POS）
- 测试和测量仪器
- 网络应用和交换机
- 工厂自动化
- HVAC和楼宇控制
- 游戏设备
- 运动控制
- 医疗器械
- 火警和安防用具
- 功耗和能源

- 运输业

1.3 高级方框图

图 1-1 在 28页表示在 Stellaris® 2000 系列器件中的全部特性；不是所有特性在LM3S2950微控制器中都可以使用。

图 **1-1. Stellaris® 2000** 系列高级方框图



1.4 功能概述

下面的小节给出了LM3S2950微控制器特性的概述。 括号中的页码编号指出该特性将在哪一章中详细描述。 订购和支持信息可在“订购和联系信息” 在 502页中找到。

1.4.1 ARM Cortex™-M3

1.4.1.1 处理器内核（见34页）

Stellaris®产品系列的所有成员（包括LM3S2950微控制器）都是围绕ARM Cortex™-M3处理器内核来设计的。 ARM Cortex-M3处理器为满足小存储要求解决方案、简化管脚数以及低功耗三方面要求的高性能、低成本平平台提供一个内核，与此同时，它还提供了出色的计算性能和优越的系统系统中断响应能力。

“ARM Cortex-M3处理器内核” 在 34页提供ARM内核的概述，内核的详细描述请见*ARM® Cortex™-M3 技术参考手册*。

1.4.1.2 系统定时器（SysTick）

Cortex-M3集成了一个系统定时器，SysTick。 SysTick给一个简单的24位写清零、递减、计数到零时重装的计数器提供灵活的控制机制。 该计数器有几种使用方法，例如：

- 用作一个RTOS时钟节拍定时器，以编程设定的速率（例如，100Hz）启动，调用一个SysTick程序。
- 用作一个使用系统时钟的高速报警定时器。
- 用作一个速率可变的报警或信号定时器——它的工作时间取决于使用的参考时钟和计数器的动态范围。
- 用作一个简单的计数器。 软件可以使用这个定时器来测量完成操作的时间或操作用掉的时间。
- 一个根据未到达/到达的时间（missing/meeting durations）来控制的内部时钟。 作为动态时钟管理控制循环的一部分，控制和状态寄存器中的COUNTFLAG位域可以用来决定某项操作是否在设定的时间内完成。

1.4.1.3 嵌套向量中断控制器（NVIC）

LM3S2950 控制器包含ARM Cortex-M3内核中的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。 所有的异常都是在处理模式中处理的。 在出现异常时，处理器的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，这样便可以高效地进入中断。 处理器还支持末尾连锁，这使处理器无需保存和恢复状态便可执行两个连续的中断。 软件可在7个异常（系统处理程序）和36个中断上设置8个优先级。

“中断” 在 41页提供对NVIC控制器和中断映射的概述。 异常和中断的详情请参阅*ARM® Cortex™-M3 技术参考手册*。

1.4.2 电机控制外设

为了加强电机控制，LM3S2950控制器包含脉宽调制（PWM）输出和正交编码器接口（QEI）。

1.4.2.1 PWM

脉宽调制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对模拟信号进行编码。 典型的应用包括开关电源和电机控制。

在LM3S2950上，PWM 运动控制功能可通过以下实现：

- 使用PWM管脚的特定、灵活的运动控制硬件
- 使用CCP管脚的通用定时器的运动控制特性

***PWM**管脚（见**400**页）*

该LM3S2950 PWM 模块由3个PWM 发生器模块和控制模块组成。 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、一个死区发生器和一个中断。 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信号发生模块都产生两个PWM信号,它们可以是独立的信号，或者是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。

***CCP**管脚（见**196**页）*

通用定时器模块的CCP（捕获比较PWM）管脚可由软件编程来支持简单的PWM模式，在该模式中PWM信号的输出反相可由软件编程控制。

1.4.2.2 QEI（见435页）

正交编码器，又称为2通道增量式编码器，用于将线性位移转换为脉冲信号。 通过监控脉冲的数目和两个信号的相对相位，用户可以跟踪旋转位置、旋转方向和速度。 另外，第三个通道称为索引信号，可用于对位置计数器进行复位。

带有索引脉冲的正交编码器（QEI）模块对由正交编码器转轮所产生的编码进行解码，从而计算位置对时间的积分，并确定旋转的方向。 另外，该接口还能捕获编码器转轮的运行速率。

1.4.3 模拟外设

为了支持模拟信号，LM3S2950 微控制器提供 3个模拟比较器。

1.4.3.1 模拟比较器（见388页）

模拟比较器是比较两个模拟电压，并提供一个逻辑输出来表示比较结果的外设。

该LM3S2950 微控制器提供3个独立集成的模拟比较器，这些模拟比较器可配置为驱动输出或产生中断。

比较器可以把测试电压与下列任何一种电压进行比较：

- 单个外部参考电压
- 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信号来通知应用程序，使其启动捕获采样序列。

1.4.4 串行通信外设

该LM3S2950 控制器支持异步和同步串行通信：

- 3个完全可编程的16C550型UART
- 2个SSI模块
- 1个I^2C模块

- 2 个CAN单元

1.4.4.1 UART（见249页）

通用异步收发器（UART）是一个用于RS232C串行通信的集成电路，它带有一个发送器（并行到串行的转换器）和一个接收器（串行到并行的转换器），它们各自独立计时。

该LM3S2950 控制器包括 3个完全可编程的 16C550-type UART，支持高达 460.8 Kbps的数据传输速率。 （尽管16C550-type UART在功能上与16C550 UART类似，但它们的寄存器不兼容。）另外，每个 UART都能支持 IrDA功能。

提供独立的16x8发送（TX）和16x12接收（RX） FIFO，可减轻CPU中断服务的负担。 UART能够根据RX、TX、调制解调器的状态和错误条件产生独立可屏蔽的中断 。 如果任何中断发生并且未被屏蔽，那么模块将提供一个组合的中断。

1.4.4.2 SSI（见287页）

同步串行接口（SSI）是一个4线双向的通信接口。

该LM3S2950 控制器包括 2个SSI模块，提供器件与外围设备之间的同步串行通信功能，模块可以配置成使用Freescale SPI、MICROWIRE或TI同步串行接口的帧格式。 数据帧的大小也是可以配置的，范围为4～16位，包括4和16。

每个SSI模块对从外围器件接收到的数据执行串并转换，对发送到外围设备的数据执行并串转换。TX和RX的通路都有内部FIFO负责缓冲，可单独存储8个16位的值 。

每个SSI模块可以配置用作主设备或从设备。作为从机设备的时候，还可以通过配置将SSI模块的输出禁能，从而使一个主设备可以与多个从设备相连。

每个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，SSI模块输入的时钟信号将通过它们来生成SSI输出的串行时钟信号。 位速率根据输入时钟产生，最大位速率取决于连接的外设。

1.4.4.3 I^2C (见322页)

内部集成电路（I^2C）总线通过一个两线设计（串行数据线SDA和串行时钟线SCL）来提供双向数据传输。

I^2C总线可与诸如串行存储器（RAM和ROM）、网络设备、LCD、音频发生器等外部I^2C器件相连。在产品开发和生产过程中，I^2C总线还能够用作系统测试和诊断用途。

该LM3S2950 控制器包括1个 I^2C 模块 ，提供了与I^2C总线上其它IC器件进行通信的能力。 I^2C总线支持能够发送和接收（读和写）数据的器件。

I^2C总线上的器件能够被指定为主机或从机。 这个I^2C模块支持作为主机或从机来发送和接收数据，也支持既用作主机又用作从机的同步操作。 I^2C共有4种工作模式：主发送、主接收、从发送和从接收。

Stellaris® I^2C模块可以以两种速度运行：标准速度（100 Kbps）和快速（400 Kbps）。

I^2C主机和从机都能够产生中断。 I^2C主机在发送或接收操作完成（或由于出现错误而中止）时产生中断。 I^2C从机在主机已向其发送数据或请求数据时产生中断。

1.4.4.4 CAN网络（见353页）

控制局域网（CAN）是一种连接电子控制单元（ECU）的多播共享串行总线标准。CAN总线专门为了承受电磁噪声较强的环境而设计，可以使用差分平衡线，如RS-485或更结实的双绞线。最初，它被设计成用在汽车行业中，但是现在已被广泛应用到许多嵌入式控制应用中（例如，工业和医疗等）。当网络长度不足40米时传输的位速率可高达1Mb/s。网络传输距离越长，位速率就越小（例如，500m时的位速率为125 Kb/s）。

发送器向所有CAN结点发送一条消息（广播）。每个节点根据接收到的标识符来判断是否处理该消息。标识符还决定了总线访问竞争中消息享有的优先级。每个CAN消息都能够传输0到8个字节的用户信息。该LM3S2950 包括 2个CAN 单元。

1.4.5 系统外设

1.4.5.1 可编程的**GPIO**（见**152**页）

通用输入/输出（GPIO）管脚为各种连接方式带来了灵活性。

Stellaris®GPIO模块由8个物理GPIO块组成，每个块对应一个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）并支持高达 10-60个可编程的输入/输出管脚。可用的GPIO数目取决于正在使用的外设（有关每个GPIO管脚可用的信号见“信号表” 在 452页）。

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

1.4.5.2 **4**个可编程的定时器（见**191**页）

可编程的定时器可用来计数或定时驱动定时器输入管脚的外部事件。

Stellaris®通用定时器模块（GPTM）包含4个GPTM块。每个GPTM 模块提供2个16位的定时器/计数器，可配置作为定时器或事件计数器独立操作，或配置为一个32位的定时器或一个32位的实时时钟（RTC）。

当配置为32位模式时，定时器可作为实时时钟（RTC），单次触发定时器或周期定时器运行。当配置成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位预分频器来扩展精度。16位的定时器也可以被配置为事件捕获或者脉宽调制（PWM）功能。

1.4.5.3 看门狗定时器（见**226**页）

看门狗定时器在到达超时值时可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。

Stellaris®看门狗定时器模块包括一个32位的递减计数器、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。

看门狗定时器可被配置成在第一次超时时产生到控制器的中断，并在第二次超时时产生复位信号。看门狗定时器配置完毕后，可以写入锁定寄存器以防止定时器配置被意外改动。

1.4.6 存储器外设

该LM3S2950 控制器提供单周期SRAM和单周期Flash存储器。

1.4.6.1 **SRAM**（见**128**页）

该LM3S2950静态随机存取存储器（SRAM）控制器支持64 KB SRAM。Stellaris®器件的内部SRAM位于地址0x0000.0000的器件存储器映射中。为了减少读－修改－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-band的处理器中，存储器映射的特定区域（SRAM和外设空间）能够使用地址别名，在单个的原子操作中访问各个位。

1.4.6.2 **Flash**存储器（见**129**页）

该LM3S2950 Flash 控制器支持 256 KB的Flash存储器。Flash由一组可独立擦除的1KB区块构成。擦除一个区块将会使整个区块的内容都变为1。这些区块配对组成的2KB区块可以分别受到保护。区块可以被标记为只读和只执行，以提供不同级别的代码保护。只读的区块不能被擦除或者编程，从而保护区块的内容不被修改。只执行的区块不能被擦除或者编程，而且只能通过控制器指令获取机制来读取，这可以保护区块的内容不被控制器或者调试器读取。

1.4.7 其他特性

1.4.7.1 存储器映射（见**39**页）

存储器映射列出了指令和数据在存储器中所处的位置。 LM3S2950 控制器的存储器映射可以在“存储器映射” 在 39页中找到。 寄存器地址相对于存储器映射中所示的模块基址，以十六进制递增的方式给出。

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**43**页）

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数据寄存器可以用来测试组合印刷电路板的互连性，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端口以低成本来提供高级的可测试性和芯片级访问。

JTAG端口由5个标准的管脚组成： TRST，TCK，TMS，TDI和TDO。 数据通过TDI管脚串行地发送到控制器，再由控制器通过TDO管脚输出。 这些数据的解析取决于TAP控制器的当前状态。 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*。

Luminary Micro JTAG控制器是与内置到Cortex-M3内核的ARM JTAG控制器 一同工作的。 通过复用这两个JTAG控制器的TDO输出来实现。 ARM JTAG指令选择ARM的TDO输出，而Luminary Micro JTAG指令选择Luminary Micro的TDO输出。 而复用器将由Luminary Micro JTAG控制器来控制，它可以对ARM、Luminary Micro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**53**页）

系统控制决定了器件的所有操作。 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，还处理复位的检测和报告。

1.4.7.4 休眠模块（见**111**页）

休眠模块提供了一种逻辑，将主处理器和外设电源暂时关闭，在外部事件或基于时间的事件发生时唤醒。 休眠模块包括上电顺序逻辑、一个带有一对匹配寄存器的实时时钟、低电池电压检测电路和向处理器提交中断的机制。 它还包含了64X32位字的非易失性存储器，用于在休眠过程中保存状态。

1.4.8 硬件细节

有关管脚和封装的详细信息可在下一节中找到：

- “管脚图” 在 451页
- “信号表” 在 452页
- “工作特性” 在 466页
- “电气特性” 在 467页
- “封装信息” 在 478页

2 ARM Cortex-M3处理器内核

ARM Cortex-M3处理器为高性能、低成本的平台提供一个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，它还提供出色的计算性能和优越的系统中断响应能力。特性包括：

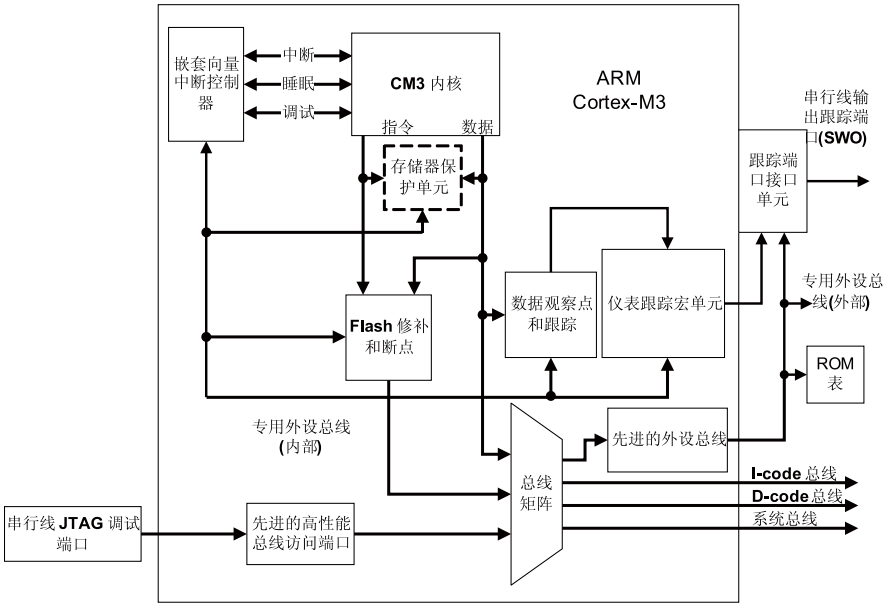
- 紧凑的内核
- Thumb-2指令集，在通常与8位和16位设备相关的存储容量中，特别是在微控制器级应用的几千字节存储量中，提供ARM内核所期望的高性能。
- 高速的应用通过Harvard结构执行，以独立的指令和数据总线为特征。
- 优越的中断处理能力，通过执行寄存器操作来实现，这些寄存器操作在处理硬件中断时使用。
- 存储器保护单元（MPU）为复杂的应用提供特权操作模式。
- 从 ARM7™ 处理器系列中移植过来，以获得更好的性能和电源效率。
- 功能齐全的调试解决方案：
  - 串行线JTAG调试端口 （SWJ-DP）
  - Flash 修补和断点（FPB）单元，用于实现断点操作
  - 数据观察点和触发（DWT）单元，用于执行观察点、触发源和系统性能分析
  - 仪表跟踪宏单元（ITM），用于支持printf 型调试
  - 跟踪端口接口单元（TPIU）用作跟踪端口分析仪的桥接

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、楼宇自动化和步进电机提供了高性能的32位运算能力。

有关ARM Cortex-M3处理器内核的更多信息，请见*ARM® Cortex™-M3 技术参考手册*。有关SWJ-DP的信息，请见*ARM® CoreSight 技术参考手册*。

2.1 方框图

图 **2-1. CPU**方框图



2.2 功能描述

重要: *ARM® Cortex™-M3 技术参考手册* 描述了ARM Cortex-M3的全部特性。但是，这些特性根据具体实现的不同而不同。本节描述了 Stellaris® 的实现方案。

Luminary Micro已实现了在 图 2-1 在 35页 中所示的ARM Cortex-M3内核。正如 *ARM® Cortex™-M3 技术参考手册* 所述，在Cortex-M3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。以下小节将会对各个组件进行详细介绍。

2.2.1 串行线和JTAG调试

Luminary Micro使用与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* 的第12章“调试端口”并不适用于Stellaris®器件。

SWJ-DP接口将SWD和JTAG调试端口组合到一个模块中。详见 *CoreSight™ 设计套件技术参考指南* 中对SWJ-DP的描述。

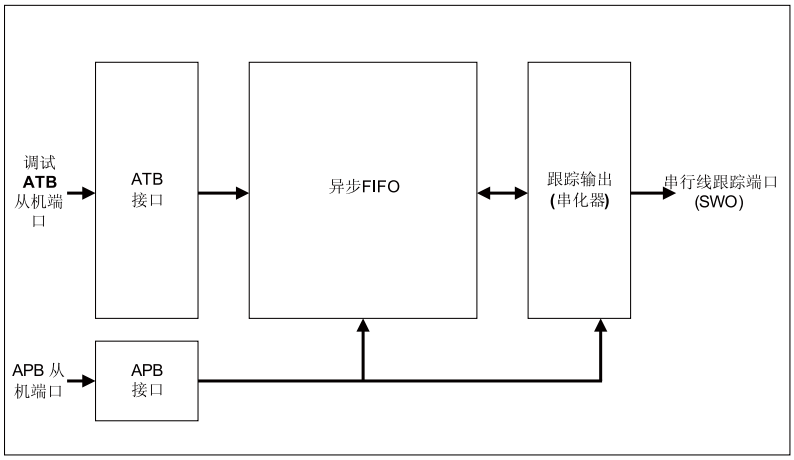
2.2.2 嵌入式跟踪宏单元 （ETM）

ETM没有在 Stellaris® 器件中执行。这就意味着 *ARM® Cortex™-M3技术参考手册* 的第15章和第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跟踪数据以及片外跟踪端口分析仪之间的桥接器。Stellaris® 器件已实现了图 2-2 在 36页 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* 中描述的ROM表。

2.2.5 存储器保护单元（MPU）

LM3S2950 控制器含有存储器保护单元(MPU)，并支持标准ARMv7的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）

嵌套向量中断控制器（NVIC）：

- 提供低-等待延时异常和中断处理
- 控制电源管理
- 执行系统控制寄存器

NVIC支持多达240个可动态配置优先级的中断，每个中断具有多达256个优先级。NVIC和处理器内核接口紧密耦合，这使能了低等待延时中断的处理和迟到达（late arriving)中断的有效处理。 NVIC保留了堆栈（嵌套）中断的内容来使能中断的尾部链接（tail-chaining）。

你只可以完全访问特权模式的NVIC，但如果你使能配置控制寄存器，你就可以在用户模式中挂起中断（见ARM® Cortex™-M3 技术参考手册）。任何其它的用户模式访问都会引起总线错误。

所有NVIC寄存器可使用字节、半字和字来访问，除非特别说明。

所有NVIC寄存器和系统调试寄存器都是小端配置，不管处理器的端点状态如何。

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。LM3S2950微控制器支持36中断，带8个优先级。

2.2.6.2 系统定时器 (SysTick)

Cortex-M3 包含一个集成的系统定时器 SysTick。SysTick 提供了一种简单的、24位写清零（clear-on-write)、递减的、到零重装（wrap-on-zero)的计数器，该计数器带有灵活的控制机制。该计数器可以在几种不同的场合中使用，例如：

- RTOS节拍定时器以一个可编程的速率（例如100Hz）运行并调用SysTick 程序。
- 使用系统时钟的高速报警定时器。
- 速率可变的报警或信号定时器—时间延迟的范围由使用的参考时钟和计数器的动态范围决定。
- 简单的计数器。软件可使用该计数器来测量完成时间和使用时间。
- 基于错过/满足持续时间的内部时钟源控制。控制和状态寄存器中的COUNTFLAG位字段可用来确定操作是否作为动态时钟管理控制循环的一部分，在设定的持续时间内完成。

功能描述

定时器包括以下3个寄存器：

- 控制和状态计数器用来配置其时钟、使能计数器、使能SysTick中断以及确定计数器状态。
- 计数器的重装值，用来提供计数器的重装值（wrap value）。
- 计数器的当前值。

第4个寄存器，SysTick 校验值寄存器，不在 Stellaris® 器件中执行。

当使能时，定时器从重装值开始往下计数一直到0，然后在下一时钟沿重新载入 SysTick 重装值寄存器中的值，接着又在下一时钟开始递减计数。向重装值寄存器写入0会在下次重装时禁止计数器。当计数器到达0时，COUNTFLAG 状态位置位。COUNTFLAG 位在读操作时清零。

对当前值寄存器进行写操作会将寄存器和COUNTFLAG状态位清零。写操作并不会引发 SysTick 异常逻辑。在读取的时候，当前值是指寄存器被访问时的值。

如果内核处于调试状态（中止），那么计数值将不会递减。定时器是根据参考时钟来计时的。参考时钟可以是内核时钟或外部时钟源。

***SysTick** 控制和状态寄存器*

使用 SysTick 控制和状态寄存器来使能 SysTick 特性。复位是 0x0000.0000。

位/字段	名称	类型	复位	描述
31:17	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
16	COUNTFLAG	R/W	0	如果上次读取时计数器计数值为0，则返回1。通过应用进行读操作时清零。如果调试器使用DAP读取，那么只要AHB-AP控制寄存器中的 MasterType位被设为0，该位就会在只读操作时清零。否则，COUNTFLAG位不会因为调试器的读操作而改变。
15:3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	CLKSOURCE	R/W	0	0 = 外部参考时钟（Stellaris微控制器没有执行。） 1 = 内核时钟。 如果没有提供参考时钟，那么CLKSOURCE会保持为1，并且提供和内核时钟一样的时间。内核时钟必须至少是参考时钟的2.5倍。如果不是这样，那么计数值将不可预知。

位/字段	名称	类型	复位	描述
1	TICKINT	R/W	0	1 = 计数到0会挂起SysTick处理程序。 0 = 计数到0不会挂起SysTick处理程序。软件可使用 COUNTFLAG 来确定是否计数到0。
0	ENABLE	R/W	0	1 = 计数器在多次触发（multi-shot）方式下操作。也就是说，计数器装载重装值，然后递减计数。当计数到0时，它将COUNTFLAG置位，并且可以根据TICKINT的值来选择是否将SysTick处理程序挂起。然后又重新装载重装值并开始计数。 0 = 计数器被禁能。

SysTick 重装值寄存器

SysTick 重装值寄存器用于指定当计数器计数到达0时装入当前值寄存器的起始值。它可以是1到0x00FF.FFFF之间的任意值。起始值也可以是0，但是由于SysTick中断和COUNTFLAG在计数从1到0时都会被激活，所以没什么作用。

因此，作为多次触发（multi-shot）定时器，它每N+1个时钟脉冲就会触发，此处N是1到0x00FF.FFFF之间的任意值。因此，如果要求每100个时钟脉冲产生一个节拍（tick）中断，那么必须向RELOAD写入99。如果在每个节拍中断时写入新值，那么它就被当作单次触发，这样就必须写入实际的递减值。例如，如果要求在400个时钟脉冲后产生一个节拍中断，那么必须向RELOAD写入400。

位/字段	名称	类型	复位	描述
31:24	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
23:0	RELOAD	W1C	-	当计数器到达0时装入 SysTick 当前值寄存器的值。

SysTick 当前值寄存器

使用 SysTick 当前值寄存器来查找该寄存器的当前值。

位/字段	名称	类型	复位	描述
31:24	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
23:0	CURRENT	W1C	-	寄存器被访问时的当前值。没有提供读-修改-写保护，所以在更改时要特别注意。 该寄存器是写清零。向该寄存器写入任意值都会将寄存器清零。清零该寄存器也会将SysTick控制和状态寄存器的COUNTFLAG位清零。

SysTick 校验值寄存器

不含SysTick校验值寄存器。

3 存储器映射

LM3S2950控制器的存储器映射在表 3-1 在 39页中给出。

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一一对应。同见 *ARM® Cortex™-M3 技术参考手册*中的第4章“存储器映射”。

重要: 在表 3-1 在 39页中，没有列出的地址被保留。

表 **3-1.** 存储器映射[a]

起始	结束	描述	有关寄存器的详细内容，请见页面。
存储器			
0x0000.0000	0x0003.FFFF	片内Flash[b]	132
0x2000.0000	0x2000.FFFF	片内带 Bit-banded特性的 SRAM[c]	132
0x2010.0000	0x21FF.FFFF	保留的不带bit-banded特性的SRAM空间	-
0x2200.0000	0x23FF.FFFF	0x2000.0000到0x200F.FFFF 的bit-band别名。	128
0x2400.0000	0x3FFF.FFFF	保留的不带bit-banded特性的SRAM空间	-
FiRM 外设			
0x4000.0000	0x4000.0FFF	看门狗定时器	228
0x4000.4000	0x4000.4FFF	GPIO 端口 A	158
0x4000.5000	0x4000.5FFF	GPIO 端口 B	158
0x4000.6000	0x4000.6FFF	GPIO 端口 C	158
0x4000.7000	0x4000.7FFF	GPIO 端口 D	158
0x4000.8000	0x4000.8FFF	SSI0	297
0x4000.9000	0x4000.9FFF	SSI1	297
0x4000.C000	0x4000.CFFF	UART0	255
0x4000.D000	0x4000.DFFF	UART1	255
0x4000.E000	0x4000.EFFF	UART2	255
外设			
0x4002.0000	0x4002.07FF	I2C 主机 0	333
0x4002.0800	0x4002.0FFF	I2C 从机 0	344
0x4002.4000	0x4002.4FFF	GPIO 端口 E	158
0x4002.5000	0x4002.5FFF	GPIO 端口 F	158
0x4002.6000	0x4002.6FFF	GPIO 端口 G	158
0x4002.7000	0x4002.7FFF	GPIO 端口 H	158
0x4002.8000	0x4002.8FFF	PWM	406
0x4002.C000	0x4002.CFFF	QEI0	438
0x4003.0000	0x4003.0FFF	定时器 0	201
0x4003.1000	0x4003.1FFF	定时器 1	201
0x4003.2000	0x4003.2FFF	定时器 2	201
0x4003.3000	0x4003.3FFF	定时器 3	201
0x4003.C000	0x4003.CFFF	模拟比较器	388
0x4004.0000	0x4004.0FFF	CAN0 控制器	363

起始	结束	描述	有关寄存器的详细内容，请见页面。
0x4004.1000	0x4004.1FFF	CAN1 控制器	363
0x400F.C000	0x400F.CFFF	睡眠模块	115
0x400F.D000	0x400F.DFFF	Flash 控制	132
0x400F.E000	0x400F.EFFF	系统控制	59
0x4200.0000	0x43FF.FFFF	0x4000.0000到0x400F.FFFF的bit-band 别名。	-
专用的外设总线			
0xE000.0000	0xE000.0FFF	仪表跟踪宏单元（ITM）	*ARM® Cortex™-M3 技术参考手册*
0xE000.1000	0xE000.1FFF	数据观察点和跟踪（DWT）	
0xE000.2000	0xE000.2FFF	Flash 修补和断点（FPB）	
0xE000.3000	0xE000.DFFF	保留	
0xE000.E000	0xE000.EFFF	嵌套向量中断控制器（NVIC）	
0xE000.F000	0xE003.FFFF	保留	
0xE004.0000	0xE004.0FFF	跟踪端口的接口单元（TPIU）	
0xE004.1000	0xE004.1FFF	保留	-
0xE004.2000	0xE00F.FFFF	保留	-
0xE010.0000	0xFFFF.FFFF	保留用于厂商外设	-

a. 在对所有被保留的空间执行读或写操作时将返回一个总线故障。

b. 不可使用的Flash将在这个范围内出现总线错误。

c. 不可使用的 SRAM 将在这个范围内出现总线错误。

4 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。所有异常都在处理器模式中处理。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），这使处理器无需保存和恢复状态便可执行连续的(back-to-back)中断。

表 4-1 在 41页列出了所有的异常。软件可在7个异常（系统处理程序）以及36个中断上设置8个优先级（在 表 4-2 在 42页中列出）。

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。中断是通过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* 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)在内部看作是优先级4，仅次于复位、NMI以及硬件故障。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的优先级更高。

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

注意: 在 表 4-2 在 42页中没有列出的中断为保留。

表 **4-1.** 异常类型

异常类型	位置	优先级[a]	描述
-	0	-	复位时，栈顶从向量表的第一个入口加载。
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（因此它被称为激活（中断）的基础级别）。这是异步的。
不可屏蔽的中断(NMI)	2	-2	不可停止，也不会被任何异常抢占，复位除外。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可配置的故障处理程序被禁能而无法激活时，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失配，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。
调试监控器	12	可调整	调试监控器（当没有暂停时）。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。

异常类型	位置	优先级[a]	描述
中断	16及其以上	可调整	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 在 42页 列出了LM3S2950 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	SSI0
8	I2C0
9	PWM 故障
10	PWM发生器 0
11	PWM发生器 1
12	PWM发生器 2
13	QEI0
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	模拟比较器 1
27	模拟比较器 2
28	系统控制
29	Flash 控制
30	GPIO 端口 F
31	GPIO 端口 G
32	GPIO 端口 H
33	UART2
34	SSI1
35	定时器3 A
36	定时器3 B
39	CAN0
40	CAN1
43	睡眠模块

5 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）可用来测试组合印刷电路板的互连和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，以及调试。

JTAG端口由5个标准的管脚组成：TRST, TCK, TMS, TDI和TDO。数据通过TDI串行发送至控制器，然后通过TDO从控制器串行输出。该数据的解析取决于TAP控制器的当前状态。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结构*。

Luminary Micro 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Luminary Micro JTAG指令则选择Luminary Micro TDO输出管脚。多路复用器由 Luminary Micro JTAG 控制器控制，它可以对 ARM、Luminary Micro和未执行的JTAG指令进行综合编程。

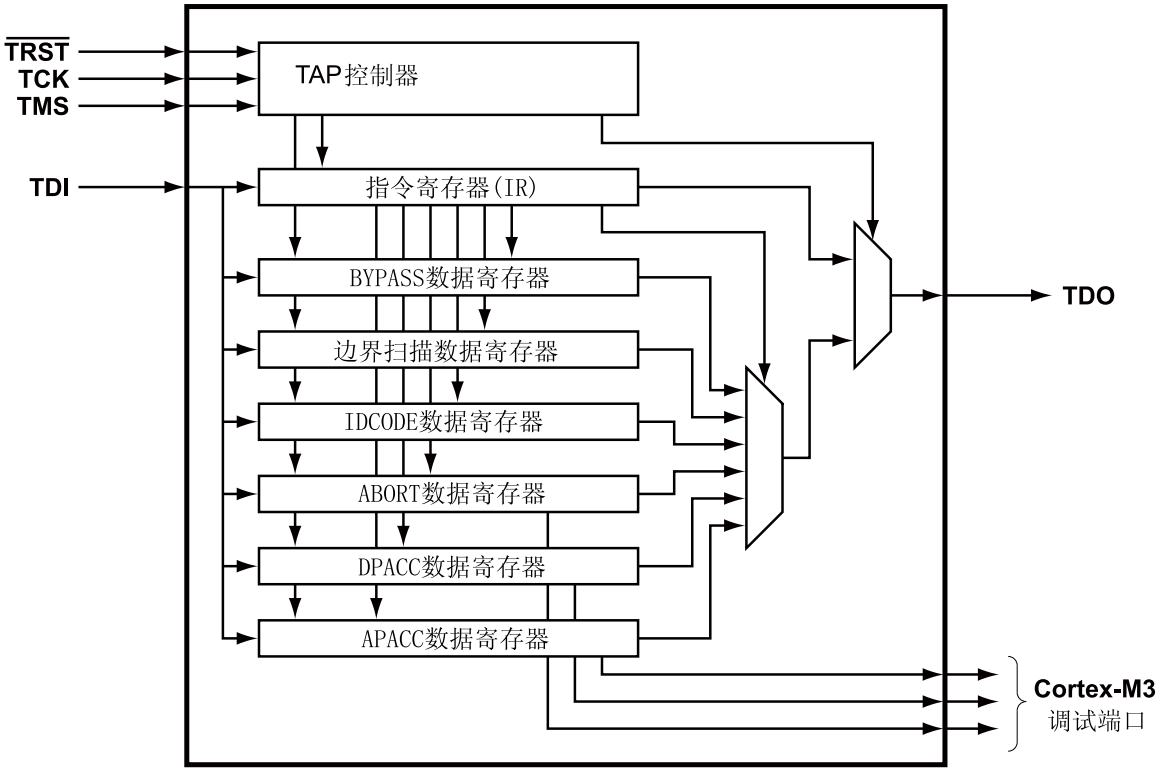
JTAG模块具有以下特性：

- IEEE 1149.1-1990 可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™-M3 技术参考手册*。

5.1 方框图

图 **5-1. JTAG** 模块方框图



5.2 功能描述

JTAG模块的高级概念图如 图 5-1 在 44页所示。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 TRST, TCK 和 TMS输入管脚控制。TAP控制器的当前状态取决于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，像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 在 49页）。

要了解JTAG的时序图，请参考“JTAG和边界扫描” 在 474页。

5.2.1 JTAG 接口管脚

JTAG接口包括5个标准的管脚：TRST, TCK, TMS, TDI 和 TDO。这些管脚以及与其相关联的复位状态在 表 5-1 在 45页中给出。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
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 驱动器	高阻

5.2.1.1 测试复位输入 (TRST)

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。当 TRST 有效时，TAP控制器复位到Test-Logic-Reset状态，并且在TRST有效期间一直保持这种状态。当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，TRST 管脚上的内部上拉电阻在复位后使能。 GPIO端口B变为上拉电阻设置时，应该确保PB7/TRST上的内部上拉电阻保持使能，否则可能会丢失JTAG通信信息。

5.2.1.2 测试时钟输入（TCK）

TCK 管脚是JTAG模块的时钟。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，它确保采用菊链（daisy-chain）方式的多个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通过一个自由运行的时钟（额定占空比为50%）来驱动。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。这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 管脚在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* 期望 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。当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。因此，可将该序列当成一种复位机制来使用，其效果与TRST信号有效相似。可以在 图 5-2 在 46页中完整地看到JTAG测试访问端口状态机。

默认情况下，TMS管脚上的内部上拉电阻在复位后使能。GPIO端口C变为上拉电阻设置时，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通信信息。

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。GPIO端口C变为上拉电阻设置时，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通信信息。

5.2.1.5 测试数据输出（TDO）

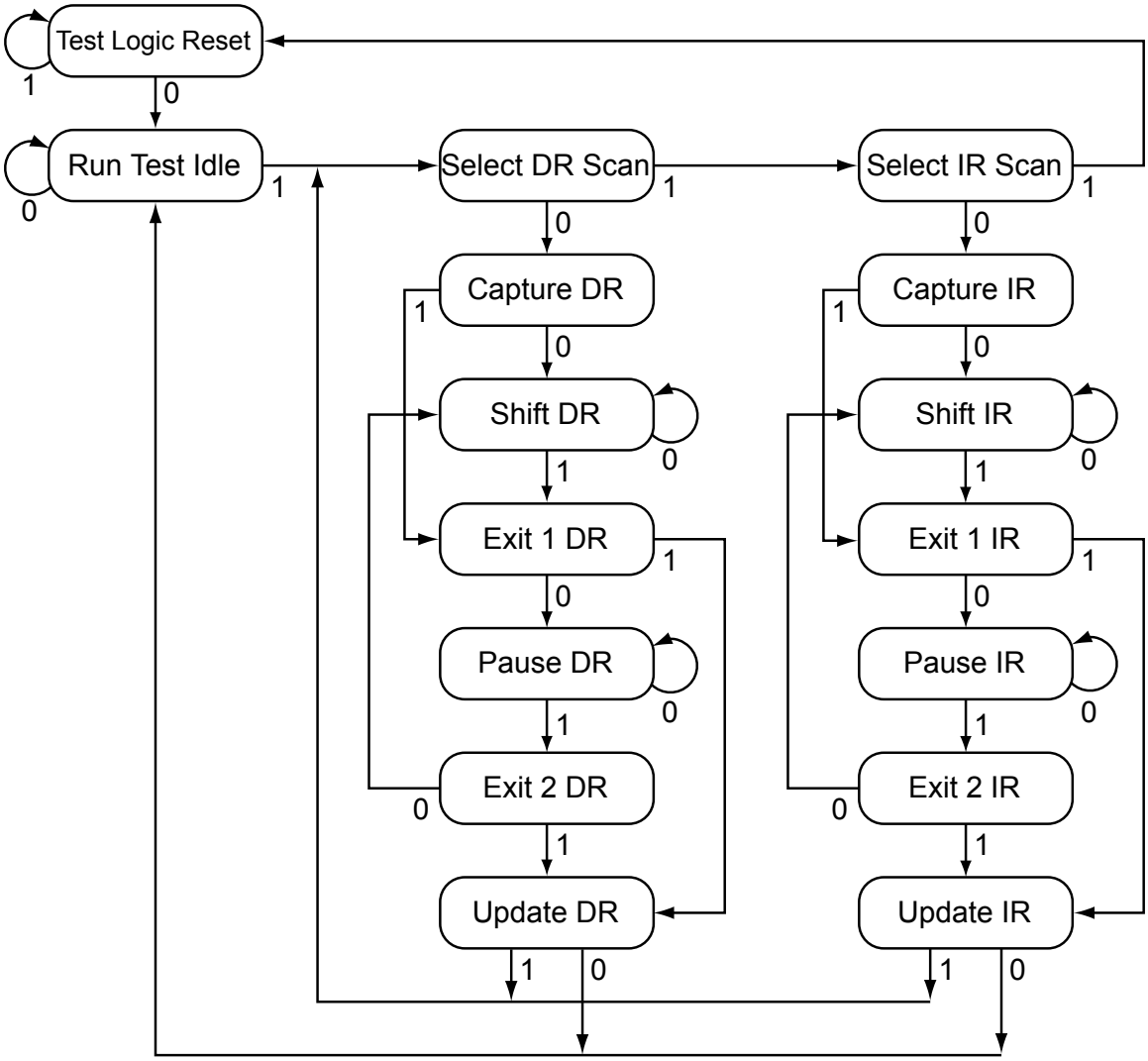
TDO管脚将一串串行信息从IR链或DR链输出。TDO的值取决于当前的TAP状态、当前指令、以及正在访问的数据链中的数据。为了在不使用JTAG端口时节省功耗，若当前并没有移出数据的活动，TDO管脚将被放置于停止的驱动状态中。因为在菊链配置中，TDO可以与另一个控制器的TDI管脚相连，所以*IEEE 标准1149.1*期望TDO的值在TCK的下降沿发生变化。

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。如果在某些TAP控制状态中能够接受高阻输出值，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 在 46页 所示。TAP控制器的状态机在发出上电复位（POR）或TRST信号时复位到Test-Logic-Reset状态。在TMS管脚上发出正确的序列，可以使JTAG模块移入新指令、移入数据、或在扩展测试序列期间变为空闲。要详细了解有关TAP控制器的功能以及每个状态发生的操作的信息，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

移位寄存器由串行移位寄存器链和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。当采样的数据通过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。每个移位寄存器将在 “寄存器描述” 在 49页 中详述。

5.2.4 操作时的注意事项（**Operational Considerations**）

在使用JTAG模块时需注意某些操作事项。因为JTAG管脚可被编程为GPIO，所以必须考虑这些管脚的电路板配置和复位条件。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RST复位时，JTAG/SWD 端口管脚将被设为管脚默认的JTAG/SWD 配置。默认的配置包括使能数字功能（将 **GPIODEN** 设为1），使能上拉电阻（将 **GPIOPUR** 设为1）和使能 PB7 和 PC[3:0] JTAG/SWD 管脚上交替的硬件功能（将**GPIOAFSEL**设为1）。

在复位后，软件只需向**GPIOAFSEL**寄存器中与PB7 和PC[3:0]对应的位写入0便可以将这些管脚配置成GPIO。如果用户不需要 JTAG/SWD 端口进行调试或板极测试，那么这提供了5个以上的GPIO以便在设计中使用。

小心 **–** 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**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**Stellaris**®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中止控制器。这会将调试器锁在元件外。通过使用一个基于外部或软件的触发器来恢复**JTAG**功能的软件程序就可以避免这种情况发生。

确认（commit）控制寄存器提供了保护层以防止对重要硬件外设的意外编程。写 **GPIO** 可选功能选择 **(GPIOAFSEL)** 寄存器（见 168页）的保护位不确认存储，除非 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 177页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 178页）的相应位已被设为1。

恢复“锁定的”器件

如果软件配置任意一个JTAG/SWD 管脚为 GPIO，并且失去与调试器进行通信的能力，则有一个调试序列可用来恢复器件。当保持器件在复位期间大量擦除Flash存储器时，执行总共10个 JTAG到SWD 和 SWD到JTAG的切换序列。恢复器件的序列为：

1. 发出和保持 RST 信号。
2. 执行 JTAG到SWD 的切换序列。
3. 执行 SWD到JTAG的切换序列。
4. 执行 JTAG到SWD 的切换序列。
5. 执行 SWD到JTAG的切换序列。
6. 执行 JTAG到SWD 的切换序列。
7. 执行 SWD到JTAG的切换序列。

8. 执行 JTAG到SWD 的切换序列。
9. 执行 SWD到JTAG的切换序列。
10. 执行 JTAG到SWD 的切换序列。
11. 执行 SWD到JTAG的切换序列。
12. 释放 RST 信号。

JTAG到SWD 和 SWD到JTAG 的切换序列在 “ARM 串行线调试（SWD）” 在 48页中描述。当执行切换序列以用于恢复器件的调试功能时，应该仅执行切换序列的步骤1和步骤2。

5.2.4.2 ARM 串行线调试（SWD）

为了无缝的地结合ARM串行线调试（SWD）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SWD接口的前导码在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, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR 和 Test Logic Reset states。

若通过 TAP 状态机的序列执行步进，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™-M3 技术参考手册* 和*ARM® CoreSight 技术参考手册*。

因为上述的序列是一系列有效的、可发出的JTAG操作，所以ARM JTAG TAP控制器不能完全兼容 *IEEE 标准1149.1*。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。

JTAG到SWD 切换

要将调试访问端口（DAP）的操作模式从JTAG切换为SWD模式，那么外部调试硬件必须发送一个切换序列到器件。 切换为SWD模式的16-位切换序列被定义为 b1110011110011110，先发送LSB位。当先发送LSB位时，这也可以表示为 16'hE79E。 完整的切换序列应包括在TCK/SWCLK 和 TMS/SWDIO信号上的下列操作：

1. 发送至少 50 TCK/SWCLK周期，TMS/SWDIO设为1。这确保了 JTAG和SWD都在它们的复位/空闲状态下。
2. 发送 16-位 JTAG到SWD 切换序列, 16'hE79E。
3. 发送 至少 50 TCK/SWCLK周期，TMS/SWDIO设为1。这确保了在发送切换序列之前，如果 SWJ-DP 已经在 SWD 模式，那么 SWD 进入线复位状态。

SWD到JTAG 切换

要将调试访问端口（DAP）的操作模式从SWD切换为JTAG模式，那么外部调试硬件必须发送一个切换序列到器件。切换为 JTAG 模式的16-位切换序列被定义为 b1110011110011110，先发送LSB位。若先发送MSB位则可表示为 16'hE73C。完整的切换序列应包括在TCK/SWCLK 和 TMS/SWDIO信号上的下列操作：

1. 发送至少 50 TCK/SWCLK周期，TMS/SWDIO设为1。这确保了 JTAG和SWD都在它们的复位/空闲状态下。

2. 发送16-位 SWD到JTAG 切换序列, 16'hE73C。

3. 发送至少5 TCK/SWCLK周期，TMS/SWDIO设为1。这确保了在发送切换序列之前，如果SWJ-DP已经在 JTAG 模式，那么 JTAG 进入测试逻辑复位状态。

5.3 初始化和配置

在上电复位或外部复位（RST）后，JTAG管脚将被自动配置以进行JTAG通信。不需要用户定义的初始化或配置。但是，如果用户的应用程序将这些管脚变成GPIO功能，那么在可以恢复JTAG通信前必须将这些管脚重新配置成JTAG功能。这可以通过使用**GPIOAFSEL**寄存器来使能5个JTAG管脚(PB7 和 PC[3:0])的备用功能来实现。

5.4 寄存器描述

JTAG TAP控制器或移位寄存器链中没有可访问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 和 TDO管脚之间相连的并行加载寄存器。当TAP控制器被放置在正确的状态下时，数据位便可移入指令寄存器。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 在 49页所示。下面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	将由SAMPLE/PRELOAD 指令预加载到边界扫描链的值驱动到引脚(pad)上。
0001	INTEST	将由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，并在新的预加载数据移入时将采样的值移出边界扫描链。
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 访问寄存器”。
1011	APACC	将数据移入和移出“ARM AC访问寄存器”。
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。
其它	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 指令使用被SAMPLE/PRELOAD指令预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。这样便可以开发测试程序来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 指令使用被SAMPLE/PRELOAD指令预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链连接到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，或将数据驱动到输出点。详细内容请见 “边界扫描数据寄存器” 在 51页。

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。将适当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。详细内容请见 “ABORT 数据寄存器” 在 52页。

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。将适当的数据移入数据寄存器，并从该寄存器中读取输出数据，便可以对ARM调试和状态寄存器进行读和写操作。详细内容请见 “DPACC 数据寄存器” 在 52页。

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，便可以通过调试端口对内部组件和总线进行读和写操作。详细内容请见 “APACC 数据寄存器” 在 51页。

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。在发出上电复位（POR）信号、发出TRST信号，或在进入Test-Logic-Reset状态时，IDCODE是默认加载到JTAG指令寄存器的一个指令。详细内容请见 “IDCODE 数据寄存器” 在 50页。

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1个位的移位寄存器。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。详细内容请见 “BYPASS 数据寄存器” 在 51页。

5.4.2 数据寄存器

JTAG模块含有6个数据寄存器。它们包括：IDCODE、BYPASS、边界扫描、APACC、DPACC和ABORT串行数据寄存器链。下面的小节会对每个数据寄存器进行介绍。

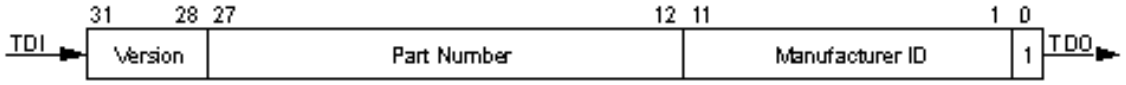
5.4.2.1 IDCODE 数据寄存器

由*IEEE 标准1149.1*定义的32-位IDCODE数据寄存器的格式如图 5-3 在 51页所示。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。这使得自动配置测试工具可以决定哪个指令是默认指令。

JTAG端口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 0x3BA00477的值。该值表示了 ARM Cortex-M3 第一个版本的处理器。这样，调试器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。

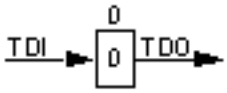
图 **5-3. IDCODE** 寄存器格式



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 在 51页所示。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。这使得自动配置测试工具可以决定哪个指令是默认指令。

图 **5-4. BYPASS** 寄存器格式

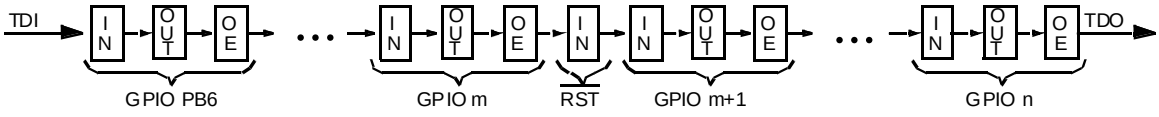


5.4.2.3 边界扫描数据寄存器

边界扫描数据寄存器的格式如图 5-5 在 51页所示。每个GPIO管脚都包含在边界扫描数据寄存器中，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，而且它们按下图中的这种顺序排列。除了GPIO管脚外，控制器的复位管脚RST也包含在这个链中。因为该复位管脚总是输入，所以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。这些指令通过EXTEST指令将数据强制移出控制器，或者通过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Stellaris®系列边界扫描描述语言（BSDL）文件，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数据寄存器的格式在*ARM® Cortex™-M3 技术参考手册*中描述。

5.4.2.5 DPACC 数据寄存器

由ARM定义的35-位 DPACC数据寄存器的格式在 *ARM® Cortex™-M3 技术参考手册*中描述。

5.4.2.6 ABORT 数据寄存器

由ARM定义的35-位 ABORT数据寄存器的格式在 *ARM® Cortex™-M3 技术参考手册*中描述。

6 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，见“器件标识” 在 53页
- 局部控制，例如复位(见“复位控制” 在 53页), 电源(见“功率控制” 在 55页)和时钟控制(见“时钟控制” 在 55页)
- 系统控制（运行、睡眠和深度睡眠模式），见“系统控制” 在 57页

6.1.1 器件标识

7个只读寄存器给软件提供有关微控制器的信息，包括版本、元件型号、SRAM大小、Flash大小和其它特性。 见**DID0**、**DID1**和**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要求。

6.1.2.1 CMOD0和CMOD1测试模式控制管脚

定义了 CMOD0和CMOD1两个管脚，生产过程中Luminary Micro用它们来测试器件。 这两个管脚没有最终用户功能，不能被使用。 CMOD管脚应该连接到地。

6.1.2.2 复位源

控制器有5个复位源：

1. 外部复位输入管脚（RST）有效，见“$\overline{\text{RST}}$管脚有效” 在 53页。
2. 上电复位（POR），见“上电复位（POR）” 在 54页。
3. 内部掉电（BOR）检测器，见“掉电复位（BOR）” 在 54页。
4. 软件启动的复位（利用软件复位寄存器），见“软件复位” 在 55页。
5. 违反看门狗定时器复位条件，见“看门狗定时器复位” 在 55页。

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）”，在经过多个复位序列之后仍能保持其状态，内部POR复位除外。内部POR复位之后，**RESC**寄存器中除POR指示器对应位之外的其它位都被清零。

6.1.2.3 $\overline{\text{RST}}$管脚有效

外部复位管脚（RST）可将微控制器复位。 该复位信号使内核及所有外设复位，JTAG TAP控制器除外（见 “JTAG 接口” 在 43页）。 外部复位序列如下：

1. 外部复位管脚（RST）有效，然后失效。

2. 内部复位释放，内核从存储器加载初始堆栈指针、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。 为了同步，从RST无效到复位序列的启动需要几个时钟周期的时间。

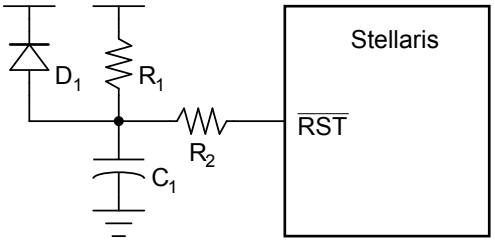
外部复位时序如图 22-10 在 476页所示。

6.1.2.4 上电复位（POR）

上电复位（POR）电路对电源电压（V_{DD}）进行监测。 当电源上升到阈值（V_{TH}）时，POR电路产生一个到内部逻辑的复位信号。 如果应用只使用POR电路，则RST输入需要通过一个上拉电阻（1K～10K Ω）连接到电源（V_{DD}）。

在片内上电复位脉冲结束时，器件必须正工作在指定的工作参数范围内。 为了保证正确工作，器件的3.3V电源必须在它经过2.0V的10ms 之内到达3.0V。 如果应用要求使用一个外部复位来使器件保持在复位状态的时间长于内部POR，RST输入可以和图 6-1 在 54页所示的电路一起使用。

图 **6-1.** 延长复位时间的外部电路



R_1和C_1定义了上电延时。 R_2电阻缓解RST输入的任何泄漏。 C_1在电源关断时通过二极管（D_1）快速放电。

上电复位序列如下：

1. 控制器等待后来的外部复位（RST）或内部POR变为无效。
2. 内部复位释放，内核从存储器加载初始堆栈指针、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有效。 上电复位时序如图 22-11 在 476页所示。

注意： 上电复位也复位JTAG控制器。 外部复位不能复位JTAG控制器。

6.1.2.5 掉电复位（BOR）

当输入电压下降导致内部掉电检测器有效时，能将控制器复位。 该复位特性最初是禁止的，可通过软件使能。

系统提供的掉电检测电路在电源（V_{DD}）降至低于掉电阈值电压（V_{BTH}）时触发。 如果检测到掉电条件，系统可产生控制器中断或系统复位。

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，以便出现掉电条件时触发一次复位。

掉电复位等效于外部RST输入的一次有效，复位在恢复到合适的V_{DD}电平之前一直保持有效。 在复位中断处理程序中可以检查**RESC**寄存器来确定掉电条件是否是复位的原因，这就允许软件确定需要恢复何种操作。

内部掉电复位时序如图 22-12 在 477页所示。

6.1.2.6 软件复位

软件可复位特定的外设或产生整个系统的复位。

外设可以单独由软件通过控制每个外设复位信号的3个寄存器（见**SRCRn**寄存器）来复位。 如果寄存器中与外设对应的位置位，则相应的外设被复位。 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“系统控制”在57页）。 注：用于指定单元所有时钟的所有复位信号在软件启动复位时有效。

整个系统可以由软件通过置位Cortex-M3应用中断和复位控制寄存器的SYSRESETREQ位来复位，这可以将包括内核在内的整个系统复位。 软件启动的系统复位序列如下：

1. 软件系统复位通过写ARM Cortex-M3应用中断和复位控制寄存器的SYSRESETREQ位来启动。
2. 内部复位有效。
3. 内部复位释放，控制器从存储器加载初始堆栈指针、初始程序计数器以及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时序如图 22-13 在 477页所示。

6.1.2.7 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成在第一次溢出（time out）时向控制器产生中断，在第二次溢出（time out）时产生复位信号。

在第一次溢出事件之后，将看门狗定时器装载（**WDTLOAD**）寄存器的值重装入32位计数器，定时器从该值继续递减计数。 如果在第一次溢出中断清除之前定时器再次递减计数到零，并且复位信号已使能，则看门狗定时器将其复位信号发送到系统。 看门狗定时器复位序列如下：

1. 看门狗定时器第二次溢出时不需要被服务。
2. 内部复位有效。
3. 内部复位释放，控制器从存储器加载初始堆栈指针、初始程序计数器以及程序计数器指定的第1条指令，然后开始执行。

看门狗复位时序如图 22-14 在 477页所示。

6.1.3 功率控制

Stellaris®微控制器提供一个集成的LDO稳压器，它可以被 用来给控制器的大部份内部逻辑提供电源。 LDO稳压器给软件提供了一种调整稳定值（regulated value）的机制，在2.25V～2.75V（2.25V和2.75V包含在内）或2.5 V ± 10%的范围内对电压进行小增量（VSTEP）调节。 调节操作通过改变**LDO**功率控制（**LDOPCTL**）寄存器的VADJ域来实现。

注意: LDO的使用是可选的。 内部逻辑可以由片内LDO或外部稳压器供电。 如果LDO被使用，则LDO输出管脚连接到印刷电路板的VDD25管脚。 在印刷电路板上LDO需要使用去耦电容。 如果外部稳压器被使用，强烈建议外部稳压器只给控制器供电，不给印刷电路板上的其它器件提供电源。

6.1.4 时钟控制

系统控制决定了该部分的时钟控制。

6.1.4.1 基础时钟源

器件有4个时钟源可供使用：

- 内部振荡器（**IOSC**）：内部振荡器是片内时钟源。它不需要使用任何外部元件。内部振荡器的频率是12 MHz ± 30%。不依赖精确时钟源的应用可以使用这个时钟源来降低系统成本。内部振荡器是器件在POR过程中和POR之后使用的时钟源 如果需要主振荡器，软件必须在复位后使能主振荡器，并在改变时钟基准前使主振荡器稳定下来。

- 主振荡器：主振荡器用下面的其中一种方法来提供一个频率精确的时钟源：OSC0输入管脚连接一个外部单端时钟源或在OSC0输入和OSC1输出管脚之间连接一个外部晶体。允许的晶体值取决于主振荡器是否用作PLL的时钟参考源。如果主振荡器用作PLL的时钟参考源，那么支持的晶体频率范围为3.579545 MHz～8.192 MHz （3.579545 MHz和8.192 MHz包含在内）。如果没有使用PLL，则支持的晶体频率在1 MHz和8.192 MHz之间。单端时钟源的范围从DC到器件的指定速率之间。支持的晶体在**RCC**寄存器的XTAL位中列出(见 68页)。

- 内部**30kHz**的振荡器：内部30kHz振荡器与内部振荡器类似，它提供30 kHz ± 30%的工作频率。它主要用在深度睡眠的节电模式中。这个节电模式从减少的内部切换中获益，它还允许主振荡器掉电。

- 外部实时振荡器：外部实时振荡器提供一个低频率、精确的时钟基准。它的目的是给系统提供一个实时时钟源。实时振荡器是休眠模块的一部分（“休眠模块”在 111页），它也提供了一个精确的深度睡眠或休眠模式节电源。

内部系统时钟（sysclk）来自 4个时钟源以及内部PLL输出和4分频的内部振荡器 （3 MHz ± 30%）。PLL时钟基准的频率必须在3.579545 MHz 到 8.192 MHz 的范围之内（3.579545MHz、8.192MHz 和 16.384MHz 包括在内）。

运行模式时钟配置（**RCC**）和运行模式时钟配置**2**（**RCC2**）寄存器提供了系统时钟的控制。**RCC2**寄存器用来扩充域，提供了**RCC**寄存器包含之外的 其它编码。使用时，**RCC2**寄存器中域的值被**RCC**寄存器相应域的逻辑使用。特别地，**RCC2**提供了更多分类的时钟配置选项。

6.1.4.2 主振荡器（**MOSC**）的晶体配置

主振荡器支持使用选择的晶体值。如果PLL使用主振荡器作为参考时钟，那么晶体支持的范围从 3.579545 到 8.192 MHz，否则，晶体支持的范围从 1 到 8.192 MHz。

RCC寄存器中的XTAL位 (见 68页) 描述了可用的晶体选择和默认的编程值。

软件用晶体值来配置**RCC**寄存器的XTAL 域。如果PLL在设计中被使用，XTAL域的值就在内部转化成PLL设置。

6.1.4.3 **PLL**频率配置

PLL上电复位过程中默认是禁能的，如果需要，PLL可以以后通过软件将其使能。软件配置PLL输入参考时钟源，指定输出分频值来设置系统时钟频率，并使能PLL驱动输出。

如果主振荡器给PLL提供了时钟基准，软件可以从**XTAL**到 **PLL** 转换（**PLLCFG**）寄存器中使用硬件提供的用来编程PLL的转换（见 72页）。内部转换提供的转换在目标PLL VCO频率的± 1%以内。

68页的晶体值域 (XTAL) 描述了可用的晶体选择和**PLLCFG**寄存器的默认设置。晶体值被写入运行模式时钟配置**(RCC)**寄存器的XTAL域。只要XTAL域的值改变，新设置就被转换，内部PLL设置被更新。

6.1.4.4 **PLL** 模式

PLL有2种工作模式：正常模式和掉电模式。

- 正常模式：PLL倍频输入时钟基准并驱动输出。

- 掉电模式：大部分PLL内部电路被禁止，PLL不驱动输出。

使用**RCC/RCC2**寄存器的域来编程PLL模式（见 68页和 73页）。

6.1.4.5 PLL操作

如果PLL配置发生变化，则PLL输出频率不稳定，直到它重新会聚（重新锁定）到新的设置 。 配置改变和重新锁定之间的时间是T_{READY} (见表 22-6 在 470页)。 在这段时间内，PLL不用作时钟源。

PLL通过下面的其中一种方法来改变：

- 更改为**RCC**寄存器的XTAL值——相同值的写入不会引起重新锁定。
- 使PLL从掉电模式变为正常模式。

定义了一个寄存器来测量T_{READY}的值。 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8.192MHz时大约为600µs）。 。 此时将提供硬件确保PLL不用作系统时钟，直到上述其中一个变化之后满足T_{READY}条件。 用户需要确保在**RCC/RCC2**寄存器切换成使用PLL之前必须有一个稳定的时钟源（类似于主振荡器）。

6.1.5 系统控制

为了节省功耗，**RCGCn**、**SCGCn**和 **DCGCn**寄存器分别控制着控制器在运行、睡眠和深度睡眠模式时系统中每个外设或模块的时钟门控逻辑。

在运行模式下，控制器执行代码。 在睡眠模式中，有效外设的时钟频率不变，但处理器不再需要时钟，所以也不再执行代码。 在深度睡眠模式中，除了正在停止的处理器时钟之外，有效外设的时钟频率可以改变（由运行模式的时钟配置决定）。 中断可使器件从其中一种睡眠模式返回到运行模式；睡眠模式在代码请求时进入。 下面对每种模式进行更详细地描述。

定义了4种级别地器件操作：

- 运行模式。 运行模式提供了处理器和**RCGCn**寄存器当前使能的所有外设的正常操作。 系统时钟可以是包括PLL在内的任何可用的时钟源。
- 睡眠模式。 睡眠模式通过Cortex-M3内核执行一条WFI（等待中断）指令进入。 系统中任何适当配置的中断事件将使处理器回到运行模式。 更详细的信息请见*ARM® Cortex™-M3 技术参考手册*的系统控制NVIC一节。

  在睡眠模式中，Cortex-M3处理器内核和存储器子系统不使用时钟。 当自动时钟门控使能时（见**RCC**寄存器），**SCGCn**寄存器使能的外设时钟被使用；而当自动时钟门控禁止时，**RCGCn**寄存器使能的外设时钟被使用。 睡眠模式下的系统时钟，其时钟源和频率与运行模式下的相同。
- 深度睡眠模式。 深度睡眠模式通过先写ARM Cortex-M3 NVIC系统控制寄存器的深度睡眠使能位、再执行一条WFI指令进入。 系统中任何适当配置的中断事件将使处理器回到运行模式。 更详细的信息请见*ARM® Cortex™-M3 技术参考手册*的系统控制NVIC一节。

  Cortex-M3处理器内核和存储器子系统不使用时钟。 当自动时钟门控使能时（见**RCC** 寄存器），**DCGCn**寄存器使能的外设时钟被使用；而当自动时钟门控禁止时， **RCGCn** 寄存器使能的外设时钟被使用。 系统的时钟源默认是主振荡器或**DSLPCLKCFG**寄存器指定的内部振荡器，前提是其中一个时钟源要使能。 当使用**DSLPCLKCFG**寄存器时，内部振荡器上电，如有必要，主振荡器可以掉电。 如果PLL在执行WFI指令时运行，则硬件将使PLL掉电，并将有效**RCC/RCC2**寄存器的SYSDIV域分别替换成/16或/64。 当出现深度睡眠退出事件时，硬件会把在深度睡眠阶段被停止的时钟使能之前，将系统时钟恢复为开始进入深度睡眠模式时采用的时钟源和频率。
- 休眠模式。 这种模式下，器件主要功能部件的电源关断，只有休眠模块的电路有效。 需要一个外部唤醒事件或RTC事件来使器件回到运行模式。 Cortex-M3处理器和休眠模块之外的外设看见一个正常的“上电”序列，处理器启动运行代码。 通过检查休眠模块寄存器可以确定已经从休眠模式重新启动。

6.2 初始化和配置

PLL的配置可通过直接对**RCC/RCC2** 寄存器执行写操作来实现。如果 **RCC2**寄存器正在使用，则USERCC2位必须置位，适当的 **RCC2**位/域被使用。 成功改变基于PLL的系统时钟所需的步骤如下：

1. 通过置位**RCC**寄存器的BYPASS位，清零寄存器的USESYS位来旁路PLL和系统时钟分频器。 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. 选择晶体值（XTAL）和振荡器源（OSCSRC），并清零<P2><I5>RCC</I5>的PWRDN 位和OEN位</P2><P6>**RCC/RCC2**的<I7>PWRDN</I7>位<P6>。 XTAL域的设置操作将自动获得所选晶体的有效的PLL配置数据，清零PWRDN位将给PLL及其输出供电并将它们使能。
3. 在**RCC/RCC2** 中选择所需的系统分频器（SYSDIV），置位**RCC**的USESYS位。 SYSDIV域决定了微控制器的系统频率。
4. 通过查询原始中断状态（**RIS**）寄存器的 PLLLRIS位来等待PLL锁定。
5. 通过清零**RCC/RCC2**中的 BYPASS位来使能PLL的使用。

6.3 寄存器映射

表 6-1 在 58页列出了按功能分组的系统控制寄存器。 列出的偏移量是相对于0x400F.E000的系统控制基址的寄存器地址的十六进制增量。

注意: 未使用的系统控制寄存器空间保留用于未来使用或供给Luminary Micro公司内部使用。 软件不应修改任何保留的存储器地址。

表 **6-1.** 系统控制 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	DID0	RO	-	器件标识0	60
0x004	DID1	RO	-	器件标识1	76
0x008	DC0	RO	0x00FF.007F	器件功能0	78
0x010	DC1	RO	0x0310.30DF	器件功能1	79
0x014	DC2	RO	0x070F.1137	器件功能2	81
0x018	DC3	RO	0x3F00.FFFF	器件功能3	83
0x01C	DC4	RO	0x0000.00FF	器件功能4	85
0x030	PBORCTL	R/W	0x0000.7FFD	掉电复位控制	62
0x034	LDOPCTL	R/W	0x0000.0000	LDO功率控制	63
0x040	SRCR0	RW	0x00000000	软件复位控制0	107
0x044	SRCR1	RW	0x00000000	软件复位控制1	108
0x048	SRCR2	RW	0x00000000	软件复位控制2	110
0x050	RIS	RO	0x0000.0000	原始中断状态	64
0x054	IMC	R/W	0x0000.0000	中断屏蔽控制	65
0x058	MISC	R/W1C	0x0000.0000	屏蔽后中断的状态和清零	66

偏移量	名称	类型	复位	描述	见页面
0x05C	RESC	R/W	-	复位原因	67
0x060	RCC	R/W	0x07AE.3AD1	运行模式时钟配置	68
0x064	PLLCFG	RO	-	XTAL到PLL转换	72
0x070	RCC2	R/W	0x0780.2800	运行模式时钟配置2	73
0x100	RCGC0	RW	0x00000040	运行模式时钟选通控制寄存器0	86
0x104	RCGC1	RW	0x00000000	运行模式时钟选通控制寄存器1	92
0x108	RCGC2	RW	0x00000000	运行模式时钟选通控制寄存器2	101
0x110	SCGC0	RW	0x00000040	睡眠模式时钟选通控制寄存器0	88
0x114	SCGC1	RW	0x00000000	睡眠模式时钟选通控制寄存器1	95
0x118	SCGC2	RW	0x00000000	睡眠模式时钟选通控制寄存器2	103
0x120	DCGC0	RW	0x00000040	深度睡眠模式时钟选通控制寄存器0	90
0x124	DCGC1	RW	0x00000000	深度睡眠模式时钟选通控制寄存器1	98
0x128	DCGC2	RW	0x00000000	深度睡眠模式时钟选通控制寄存器2	105
0x144	DSLPCLKCFG	R/W	0x0780.0000	深度睡眠时钟配置	75

6.4 寄存器描述

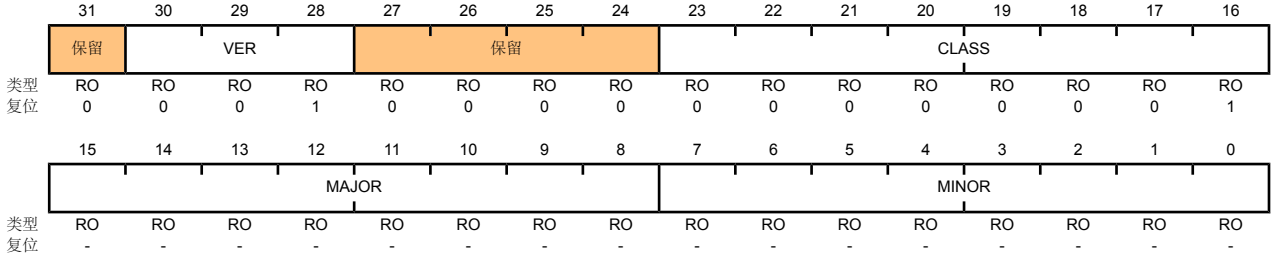
所有给出的地址都是相对于0x400F.E000的系统控制基址而言的。

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
类型 RO, 复位 -



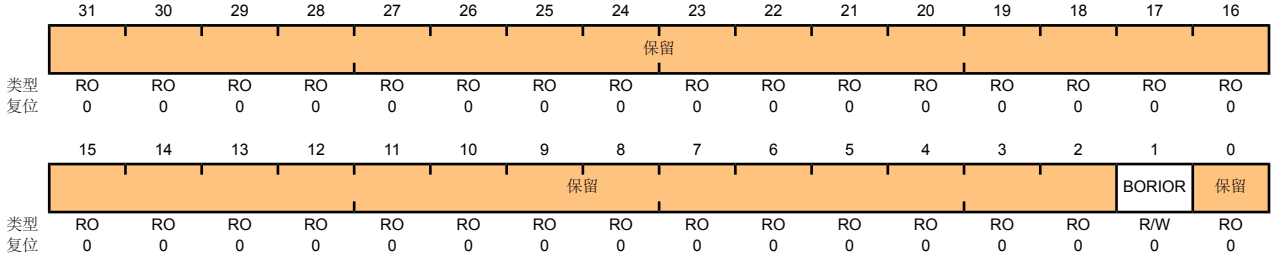
位/域	名称	类型	复位	描述
31	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
30:28	VER	RO	0x1	DID0版本 该域定义了**DID0**寄存器格式的版本。 版本号是数字。 VER域的值编码如下: 值 描述 0x1 Stellaris® Fury-class器件的第一版**DID0**寄存器格式。
27:24	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
23:16	CLASS	RO	0x1	器件分类 CLASS域的值用来识别内部设计，根据内部设计，所有器件的全部掩膜组在特定的产品线中产生。 CLASS域的值跟随新的产品线、fab过程中的变化（例如，重新映射或收缩）而变化，或者，在MAJOR或MINOR域需要和之前的器件区分开来时改变。 CLASS域的值编码如下（所有其它的编码保留）: 值 描述 0x0 Stellaris® Sandstorm-class器件。 0x1 Stellaris® Fury-class器件。
15:8	MAJOR	RO	-	主版本 这个域指定了器件的主版本号。 主版本反映了设计的基本层的改变。主版本编号在器件型号中以字母表示（A代表第一版，B代表第二版，依此类推）。 该域编码如下: 值 描述 0x0 版本A（原始器件） 0x1 版本B（第一个基础层版本） 0x2 版本C（第二个基础层版本） 依此类推。

位/域	名称	类型	复位	描述
7:0	MINOR	RO	-	次版本 这个字段指定了器件的次版本号。次版本反映了设计的金属层的改变。MAJOR域改变时MINOR域的值重新设置。这个域的值是数字，编码如下： 值 描述 0x0 原始器件，或者一个主版本更新。 0x1 第一个金属层改变。 0x2 第二个金属层改变。 依此类推。

寄存器 2: 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。

掉电复位控制 (PBORCTL)
偏移量 0x030
类型 R/W, 复位 0x0000.7FFD



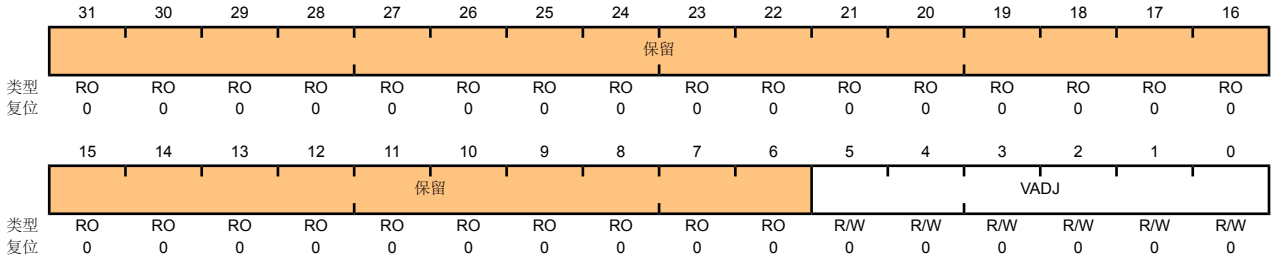
位/域	名称	类型	复位	描述
31:2	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORIOR	R/W	0	BOR中断或复位 该位控制如何将BOR事件发送给控制器。如果该位为1，发出复位信号。否则发出中断。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 3: LDO功率控制（LDOPCTL），偏移量 0x034

该寄存器的VADJ域用来调整片内输出电压（V_{OUT}）。

LDO功率控制 (LDOPCTL)

偏移量 0x034
类型 R/W, 复位 0x0000.0000



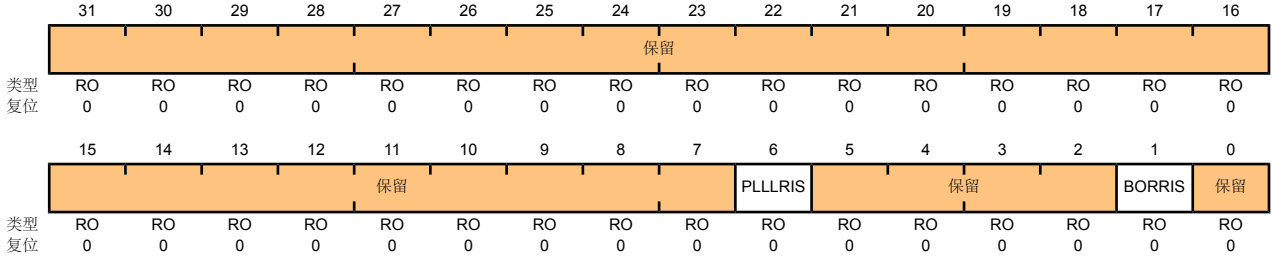
位/域	名称	类型	复位	描述
31:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5:0	VADJ	R/W	0x0	LDO输出电压 这个域设置片内输出电压。 VADJ域的编程值在下面提供。

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

寄存器 4: 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。 寄存器的位由硬件置位和清零。

原始中断状态 (RIS)
偏移量 0x050
类型 RO, 复位 0x0000.0000

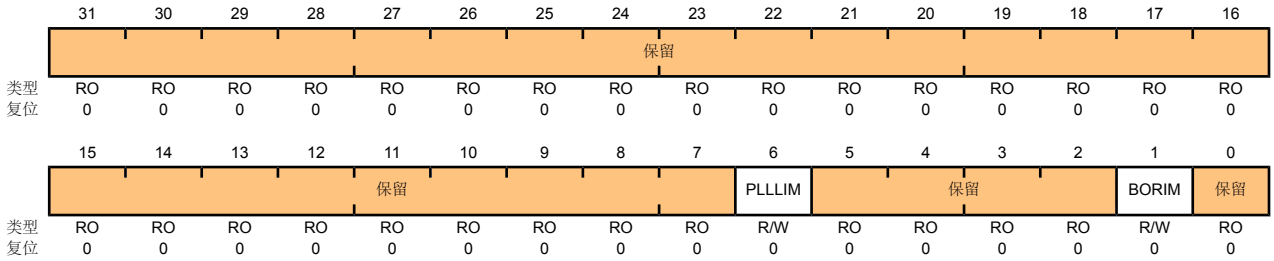


位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLRIS	RO	0	PLL锁定原始中断状态 该位在PLL T_{READY}定时器有效时置位。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORRIS	RO	0	掉电复位原始中断状态 该位是所有掉电条件的原始中断状态。 如果该位置位，则掉电条件目前有效。 这是一个来自掉电检测电路的未注册信号（unregistered signal）。 如果**IMC**寄存器的`BORIM`位被置位， **PBORCTL**寄存器的`BORIOR`位被清零，则中断被报告。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 5: 中断屏蔽控制（IMC），偏移量 0x054

系统使用该寄存器来控制中断屏蔽。

中断屏蔽控制 (IMC)
偏移量 0x054
类型 R/W, 复位 0x0000.0000



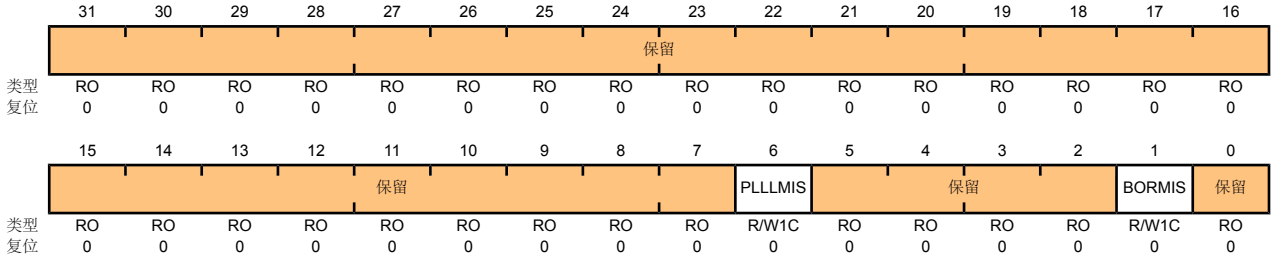
位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLIM	R/W	0	PLL锁定的中断屏蔽 该位指定限流检测是否引发一个控制器中断。 当该位置位时，如果**RIS**的PLLLRIS位置位，则产生中断；否则不产生中断。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORIM	R/W	0	掉电复位的中断屏蔽 该位指定掉电条件是否引发一个控制器中断。 当该位置位时，如果BORRIS置位，则产生中断；否则不产生中断。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 **6:** 屏蔽后中断的状态和清零（**MISC**），偏移量 **0x058**

系统使用该寄存器来控制RIS和IMC相与的结果，以便向控制器产生中断。 寄存器的所有位都是R/W1C 类型，因此这个动作也会清零**RIS** 寄存器中相应的原始中断位（见 64页）。

屏蔽后中断的状态和清零 (MISC)

偏移量 0x058
类型 R/W1C, 复位 0x0000.0000



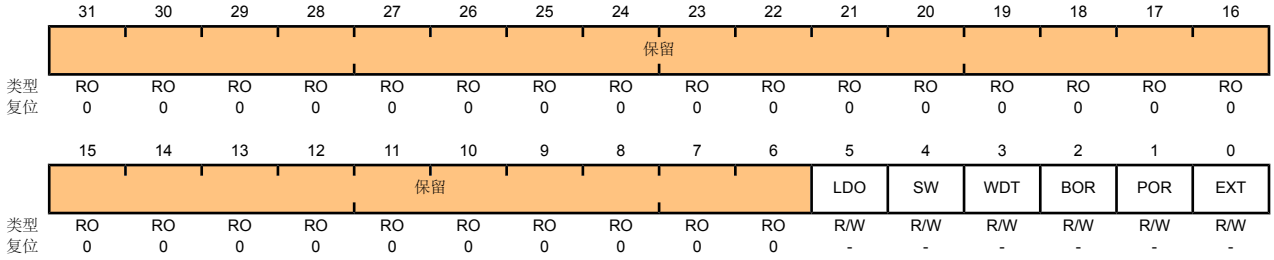
位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLMIS	R/W1C	0	PLL锁定屏蔽后的中断状态 该位在PLL T_{READY}定时器有效时置位。 通过向该位写入1来清除中断。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORMIS	R/W1C	0	BOR屏蔽后的中断状态 BORMIS只是BORRIS和屏蔽值BORIM相与的结果。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 7: 复位原因（RESC），偏移量 0x05C

这个寄存器在复位后根据复位原因来设置。这个寄存器中的位具有“粘着特性（sticky）”，在经过多个复位序列后仍保持状态，但当复位源是外部复位时例外。外部复位之后，**RESC** 寄存器的所有其它位全部清零。

复位原因 (RESC)

偏移量 0x05C
类型 R/W, 复位 -



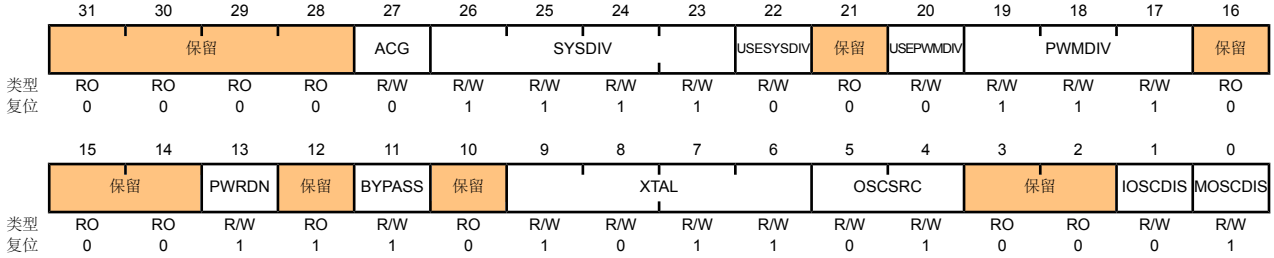
位/域	名称	类型	复位	描述
31:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	LDO	R/W	-	LDO复位 该位置位时表明复位事件是LDO电路不可调引起的。
4	SW	R/W	-	软件复位 该位置位时表明复位事件是软件复位。。
3	WDT	R/W	-	看门狗定时器复位 该位置位时表明复位事件是看门狗复位。
2	BOR	R/W	-	掉电复位 该位置位时表明复位事件是掉电复位。
1	POR	R/W	-	上电复位 该位置位时表明复位事件是上电复位。
0	EXT	R/W	-	外部复位 该位置位时表明复位事件是一个外部复位（RST有效）。

寄存器 **8:** 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)

偏移量 0x060
类型 R/W, 复位 0x07AE.3AD1



位/域	名称	类型	复位	描述
31:28	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
27	ACG	R/W	0	自动时钟门控 该位指定在控制器进入睡眠或深度睡眠模式时系统是否分别使用睡眠模式时钟门控控制（**SCGCn**）寄存器和深度睡眠模式时钟门控控制（**DCGCn**）寄存器。 如果该位置位，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。 在运行模式中，总是使用**RCGCn**寄存器来控制时钟。 这样就可以在控制器处于睡眠模式并且不需要使用外设时降低功耗。

位/域	名称	类型	复位	描述
26:23	SYSDIV	R/W	0xF	系统时钟分频值 指定使用哪个分频值来从PLL输出产生系统时钟。 PLL VCO的频率是400 MHz。 值 分频值（BYPASS=1） 频率（BYPASS=0） 0x0 保留 保留 0x1 /2 保留 0x2 /3 保留 0x3 /4 50 MHz 0x4 /5 40 MHz 0x5 /6 33.33 MHz 0x6 /7 28.57 MHz 0x7 /8 25 MHz 0x8 /9 22.22 MHz 0x9 /10 20 MHz 0xA /11 18.18 MHz 0xB /12 16.67 MHz 0xC /13 15.38 MHz 0xD /14 14.29 MHz 0xE /15 13.33 MHz 0xF /16 12.5 MHz（默认） 当读运行模式时钟配置（**RCC**）寄存器（见 68页）时，如果请求的分频值更小并且正在使用PLL，则SYSDIV的值是 MINSYSDIV。 这个更小的值允许分频非PLL时钟源。
22	USESYSDIV	R/W	0	使能系统时钟分频器 使用系统时钟分频器作为系统的时钟源。 当选择PLL作为时钟源时，将强制使用系统时钟分频器。
21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
20	USEPWMDIV	R/W	0	使能PWM时钟分频器 使用PWM时钟分频器作为PWM的时钟源。

位/域	名称	类型	复位	描述
19:17	PWMDIV	R/W	0x7	PWM单元的时钟分频值 该域指定了用来向下预分频系统时钟的二进制分频值，预分频后的系统时钟作为PWM模块的时序参考。 只能对该时钟进行2的n次幂分频并且上升沿是同步的，与系统时钟之间没有相移。 值 分频值 0x0 /2 0x1 /4 0x2 /8 0x3 /16 0x4 /32 0x5 /64 0x6 /64 0x7 /64 （默认）
16:14	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
13	PWRDN	R/W	1	PLL 掉电 该位与PLL PWRDN输入相关联。 1的复位值使PLL掉电。
12	保留	RO	1	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11	BYPASS	R/W	1	PLL旁路 选择是从PLL输出还是OSC时钟源获得系统时钟。 如果该位置位，则从OSC时钟源获得系统时钟。 否则，驱动系统的时钟是经系统分频器分频的PLL输出时钟。
10	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

位/域	名称	类型	复位	描述
9:6	XTAL	R/W	0xB	晶体值 该域指定了与主振荡器相关的晶体的值。 该域的编码在下面提供。

值	不使用PLL时的晶体频率（MHz）	使用PLL时的晶体频率（MHz）
0x0	1.000	保留
0x1	1.8432	保留
0x2	2.000	保留
0x3	2.4576	保留
0x4	3.579545 MHz	
0x5	3.6864 MHz	
0x6	4 MHz	
0x7	4.096 MHz	
0x8	4.9152 MHz	
0x9	5 MHz	
0xA	5.12 MHz	
0xB	6 MHz（复位值）	
0xC	6.144 MHz	
0xD	7.3728 MHz	
0xE	8 MHz	
0xF	8.192 MHz	

5:4	OSCSRC	R/W	0x1	振荡源 从OSC的4个输入源中选择。 值为： 值 输入源 0x0 主振荡器（默认） 0x1 内部振荡器（默认） 0x2 内部振荡器/ 4（如果用作PLL的输入，则这是必须的） 0x3 保留
3:2	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	IOSCDIS	R/W	0	内部振荡器禁能 0：内部振荡器（IOSC）被使能。 1：内部振荡器被禁能。
0	MOSCDIS	R/W	1	主振荡器禁能 0：主振荡器被使能。 1：主振荡器被禁能（默认）。

寄存器 9: XTAL到PLL转换（PLLCFG），偏移量 0x064

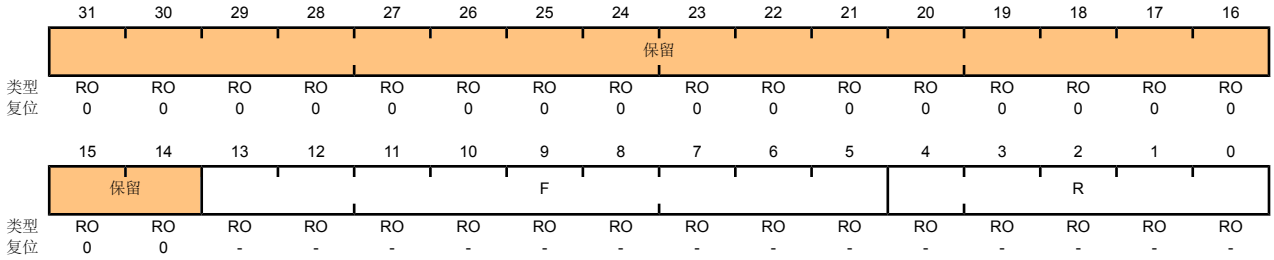
该寄存器提供将外部晶振频率转换为适当的PLL设置的方法。它在复位序列的执行过程中被初始化，并且，一旦运行模式时钟配置（**RCC**）寄存器的 XTAL域的值改变，寄存器的值就随之更新（见 68页）。

PLL的频率用**PLLCFG**域的值计算得到，如下所示：

```
PLLFreq = OSCFreq * F / (R + 1)
```

XTAL到PLL转换 (PLLCFG)

偏移量 0x064
类型 RO, 复位 -



位/域	名称	类型	复位	描述
31:14	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
13:5	F	RO	-	PLL F的值 这个域指定提供给PLL F输入的值。
4:0	R	RO	-	PLL R的值 这个域指定提供给PLL R输入的值。

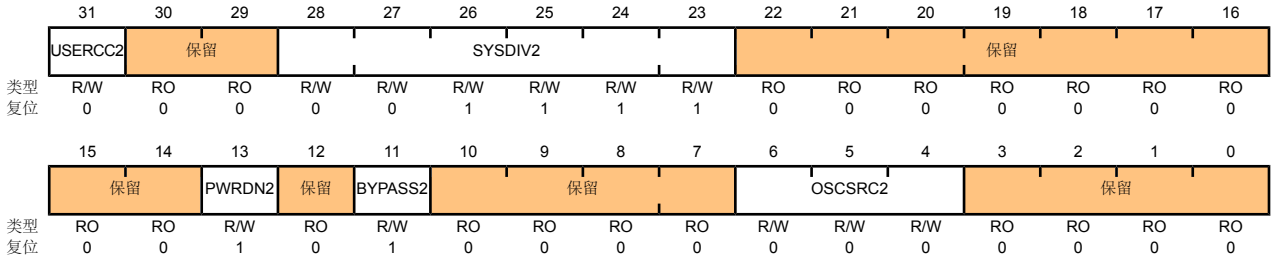
寄存器 10: 运行模式时钟配置2（RCC2），偏移量 0x070

当USERCC2位被置位时，这个寄存器可以取代**RCC**类似的寄存器域。 这就允许RCC2被用来扩展功能，同时也提供了一个向后兼容先前器件的方法。 **RCC2**寄存器中的域和它们在**RCC**寄存器中占用相同的位位置，LSB对齐。

SYSDIV2域更宽，因此可能附加有更大的分频值。 这样就可以获得更低的系统时钟频率，进一步降低深度睡眠的功耗。

运行模式时钟配置2 (RCC2)

偏移量 0x070
类型 R/W, 复位 0x0780.2800



位/域	名称	类型	复位	描述
31	USERCC2	R/W	0	使用RCC2 该位置位时替换**RCC**寄存器中的域。
30:29	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
28:23	SYSDIV2	R/W	0x0F	系统时钟分频值 指定使用哪个分频值来从PLL输出产生系统时钟。 PLL VCO的频率是400 MHz。 这个域比**RCC**寄存器的SYSDIV域更宽，可以提供更多的分频值。 这就允许系统时钟在深度睡眠模式中能在低很多的频率下运行。 例如，**RCC**寄存器的SYSDIV域的编码为 1111时提供的分频值是/16，而**RCC2**寄存器的SYSDIV2域的编码为111111时提供的分频值是/64。
22:14	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
13	PWRDN2	R/W	1	掉电PLL 该位置位时使PLL掉电。
12	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11	BYPASS2	R/W	1	旁路PLL 该位置位时旁路时钟源的PLL。
10:7	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

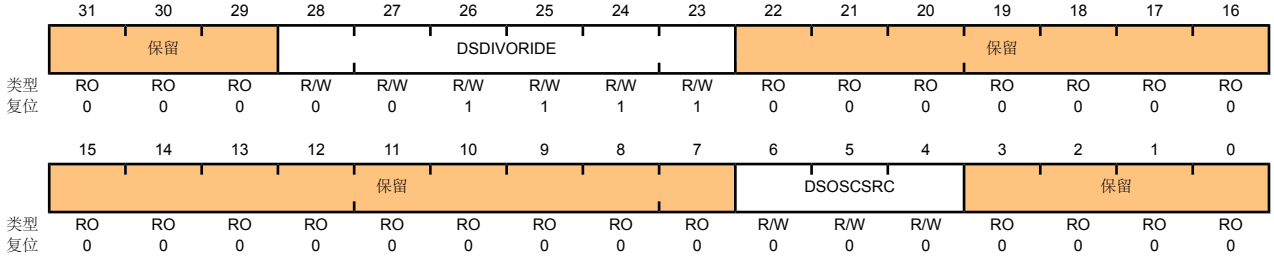
位/域	名称	类型	复位	描述
6:4	OSCSRC2	R/W	0x0	系统时钟源 值 描述 0x0 主振荡器(MOSC) 0x1 内部振荡器(IOSC) 0x2 内部振荡器 / 4 0x3 30 kHz的内部振荡器 0x7 32 kHz的外部振荡器
3:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 11: 深度睡眠时钟配置（DSLPCLKCFG），偏移量 0x144

这个寄存器为深度睡眠模式的硬件控制提供了配置信息。

深度睡眠时钟配置 (DSLPCLKCFG)

偏移量 0x144
类型 R/W, 复位 0x0780.0000

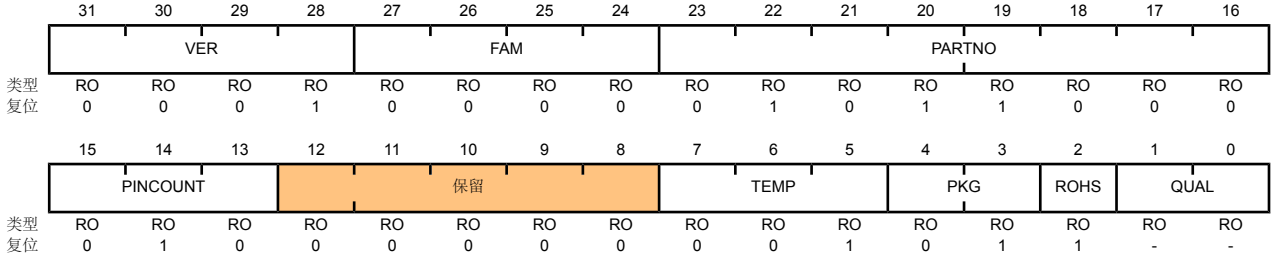


位/域	名称	类型	复位	描述
31:29	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
28:23	DSDIVORIDE	R/W	0x0F	分频器域替换 当PLL运行时出现深度睡眠，6位的系统分频器域替换。
22:7	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6:4	DSOSCSRC	R/W	0x0	时钟源 该位置位时强制IOSC在深度睡眠模式中用作时钟源。 值 名称 描述 0x0 NOORIDE 不替换振荡器时钟源。 0x1 IOSC 使用内部12MHz的振荡器作为时钟源 0x3 30kHz 使用30kHz的内部振荡器 0x7 32kHz 使用32kHz的外部振荡器
3:0	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 12: 器件标识1 （DID1）， 偏移量 0x004

该寄存器用来标识器件所属系列、型号、温度范围，管脚数，和 封装类型。

器件标识1 (DID1)
偏移量 0x004
类型 RO, 复位 -



位/域	名称	类型	复位	描述
31:28	VER	RO	0x1	DID1版 该域定义了**DID1** 寄存器格式的版本。版本号是数字的。 该VER位域的值编码如下(所有其它编码保留）： 值 描述值 0x1 **DID1** 寄存器格式的第一版，包括Stellaris系列Fury-Class器件。
27:24	FAM	RO	0x0	系列 这个位域提供Luminary Micro产品阵容内器件的系列标识符。该值编码如下（所有其它编码都保留）： 值 描述值 0x0 Stellaris系列微控制器，即所有带外部型号的器件都以LM3S开头。
23:16	PARTNO	RO	0x58	器件型号 这个位域提供了系列内器件的型号。该值编码如下（所有其它编码保留）： 值 描述值 0x58 LM3S2950
15:13	PINCOUNT	RO	0x2	封装管脚数 这个位域规定了器件封装的管脚数。该值编码如下（所有其它编码保留）： 值 描述值 0x2 100-脚封装
12:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

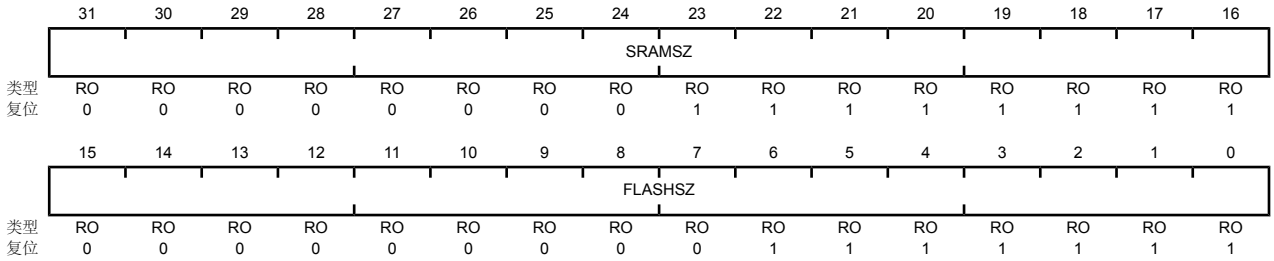
位/域	名称	类型	复位	描述
7:5	TEMP	RO	0x1	温度范围 这个位域规定了器件的温度范围。该值编码如下（所有其它编码保留）： 值 描述值 0x1 工业温度范围(-40°C～85°C)
4:3	PKG	RO	0x1	封装类型 这个位域规定了封装类型。该值编码如下（所有其它编码保留）： 值 描述值 0x1 LQFP封装
2	ROHS	RO	1	RoHS兼容性 这个位决定器件是否符合RoHS标准。A1表示器件符合RoHS标准。
1:0	QUAL	RO	-	认证情况 这个位域决定了器件的认证情况。该值编码如下（所有其它编码保留）： 值 描述值 0x0 工程样片(未经认证) 0x1 试制生产(未经认证) 0x2 完全通过认证

寄存器 **13:** 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。

器件功能0 (DC0)

偏移量 0x008
类型 RO, 复位 0x00FF.007F

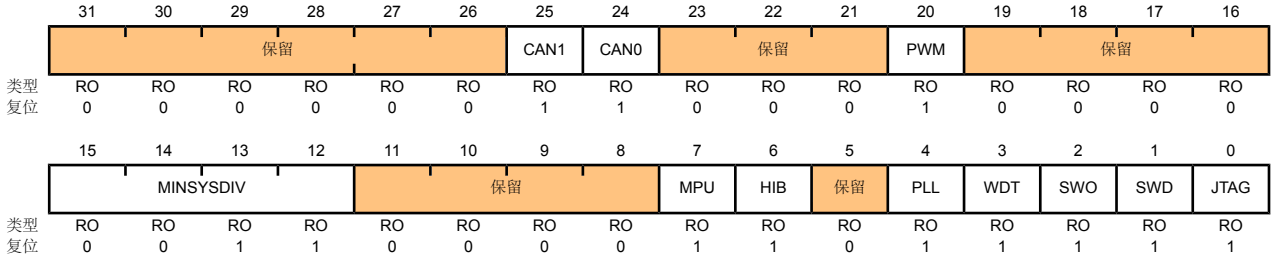


位/域	名称	类型	复位	描述
31:16	SRAMSZ	RO	0x00FF	SRAM大小 示片内SRAM存储器的大小。 值 描述值 0x00FF 64KB SRAM
15:0	FLASHSZ	RO	0x007F	Flash大小 表示片内Flash存储器的大小。 值 描述值 0x007F 256KB Flash

寄存器 14: 器件功能1 （DC1）， 偏移量 0x010

该寄存器提供系统中可用的一系列特性。Stellaris系列使用该寄存器格式来表明在特定器件中可使用下面的系列特性: CAN, PWM, ADC, 看门狗,休眠模块和调试功能。该寄存器也指示最大的时钟频率和最大的ADC采样率。该寄存器与**RCGC0**, **SCGC0**, 和**DCGC0** 时钟控制寄存器以及**SRCR0** 软件复位控制寄存器一致。

器件功能1 (DC1)
偏移量 0x010
类型 RO, 复位 0x0310.30DF



位/域	名称	类型	复位	描述
31:26	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
25	CAN1	RO	1	CAN模块1存在 置位时表示存在CAN单元1。
24	CAN0	RO	1	CAN模块0存在 置位时，表示存在CAN单元0。
23:21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
20	PWM	RO	1	PWM模块存在 置位时，表示存在PWM模块。
19:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
15:12	MINSYSDIV	RO	0x3	系统时钟分频器 系统时钟的最小4位分频值。复位值与硬件无关。参考**RCC** 寄存器以便了解如何使用SYSDIV位来改变系统时钟除数。 值 描述值 0x3 指定一个PLL四分频的50-MHz CPU时钟。
11:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	MPU	RO	1	MPU存在标志 置位时表示存在Cortex-M3存储器保护单元（MPU）模块。有关MPU的详细内容请参考Cortex-M3技术参考手册。

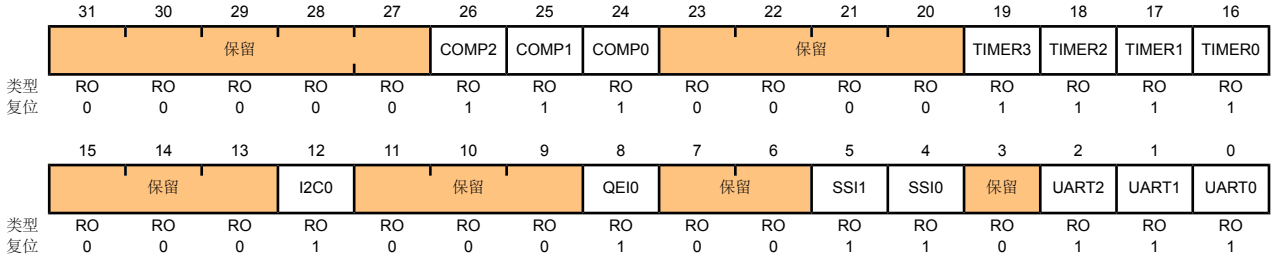
位/域	名称	类型	复位	描述
6	HIB	RO	1	休眠模块的存在标志 置位时表示存在休眠模块。
5	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
4	PLL	RO	1	PLL存在标志。 置位时表示存在片上锁相环（PLL）。
3	WDT	RO	1	看门狗定时器存在标志。 置位时表示存在看门狗定时器。
2	SWO	RO	1	SWO跟踪端口存在标志 置位时表示存在串行线输出（SWO）跟踪端口。
1	SWD	RO	1	SWD存在标志 置位时表示存在串行线调试器（SWD）。
0	JTAG	RO	1	JTAG存在标志 置位时表示存在JTAG调试器接口。

寄存器 **15:** 器件功能**2** （**DC2**）， 偏移量 **0x014**

该寄存器提供系统中可使用的一系列特性。Stellaris系列使用该寄存器格式来表明在特定器件中可使用下面的系列特性： 模拟比较器通用定时器, I2C, QEI, SSI 和 UART。 该寄存器的格式与 **RCGC1**, **SCGC1**, and **DCGC1** 时钟控制寄存器和 **SRCR1** 软件复位控制寄存器一致。

器件功能2 (DC2)

偏移量 0x014
类型 RO, 复位 0x070F.1137



位/域	名称	类型	复位	描述
31:27	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
26	COMP2	RO	1	模拟比较器2存在标志 置位时表示存在模拟比较器2。
25	COMP1	RO	1	模拟比较器1存在标志 置位时表示存在模拟比较器1。
24	COMP0	RO	1	模拟比较器0存在标志 置位时表示存在模拟比较器0。
23:20	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
19	TIMER3	RO	1	定时器3存在标志 置位时表示存在通用定时器模块3。
18	TIMER2	RO	1	定时器2存在标志 置位时表示存在通用定时器模块2。
17	TIMER1	RO	1	定时器1存在标志 置位时，表示存在通用定时器模块1。
16	TIMER0	RO	1	定时器0存在标志 置位时表示存在通用定时器模块0。
15:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	I2C0	RO	1	I2C模块0存在标志 置位时表示存在I2C模块0。

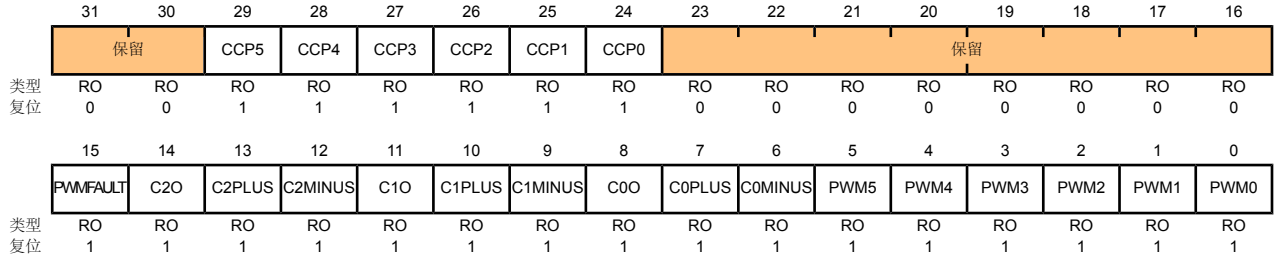
位/域	名称	类型	复位	描述
11:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
8	QEI0	RO	1	QEI0存在标志 置位时表示存在QEI模块0。
7:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
5	SSI1	RO	1	SSI1存在标志 置位时表示存在SSI模块1。
4	SSI0	RO	1	SSI0存在标志 置位时表示存在SSI模块0。
3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
2	UART2	RO	1	UART2存在标志 置位时表示存在UART模块2。
1	UART1	RO	1	UART1存在标志 置位时表示存在UART模块1。
0	UART0	RO	1	UART0存在标志 置位时表示存在UART模块0。

寄存器 16: 器件功能3 （DC3）， 偏移量 0x018

该寄存器提供一系列在系统中可以使用的特性。Stellaris系列使用该寄存器格式来表明在特定器件中可使用下面的特性： 模拟比较器I/O, CCP I/O, ADC I/O 和 PWM I/O。

器件功能3 (DC3)

偏移量 0x018
类型 RO, 复位 0x3F00.FFFF



位/域	名称	类型	复位	描述
31:30	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
29	CCP5	RO	1	CCP5管脚存在标志 置位时表示存在捕获/比较/PWM管脚5。
28	CCP4	RO	1	CCP4管脚存在标志 置位时表示存在捕获/比较/PWM管脚4。
27	CCP3	RO	1	CCP3管脚存在标志。 置位时表示存在捕获/比较/PWM管脚3。
26	CCP2	RO	1	CCP2管脚存在标志。 置位时表示存在捕获/比较/PWM管脚2。
25	CCP1	RO	1	CCP1管脚存在标志 置位时表示存在捕获/比较/PWM管脚1。
24	CCP0	RO	1	CCP0管脚存在标志。 置位时表示存在捕获/比较/PWM管脚0。
23:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	PWMFAULT	RO	1	PWM故障管脚存在标志。 置位时表示存在PWM故障管脚。
14	C2O	RO	1	C2o管脚存在标志。 置位时表示存在模拟比较器2输出管脚。
13	C2PLUS	RO	1	C2+管脚存在标志 置位时表示存在模拟比较器2（+）输入管脚。

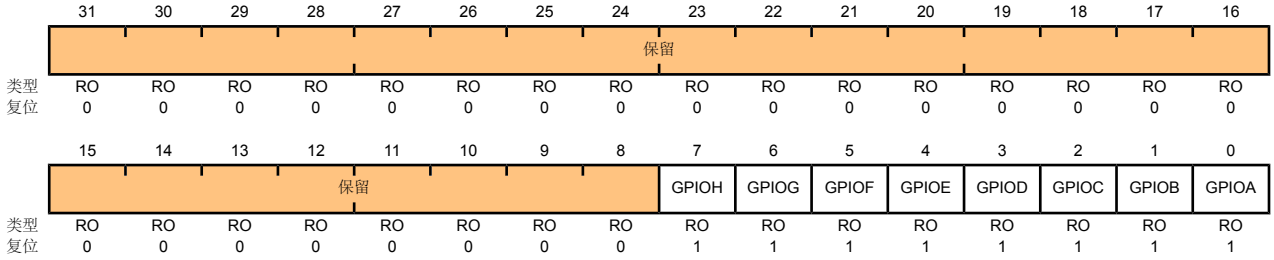
位/域	名称	类型	复位	描述
12	C2MINUS	RO	1	C2-管脚存在标志 置位时表示存在模拟比较器2（-）输入管脚。
11	C1O	RO	1	C1o管脚存在标志 置位时表示存在模拟比较器1输出管脚。
10	C1PLUS	RO	1	C1+管脚存在标志 置位时表示存在模拟比较器1（+）输入管脚。
9	C1MINUS	RO	1	C1-管脚存在标志。 置位时表示存在模拟比较器1（-）输入管脚。
8	C0O	RO	1	C0o管脚存在标志。 置位时表示存在模拟比较器0输出管脚。
7	C0PLUS	RO	1	C0+管脚存在标志 置位时表示存在模拟比较器0（+）输入管脚。
6	C0MINUS	RO	1	C0-管脚存在标志 置位时表示存在模拟比较器0（-）输入管脚。
5	PWM5	RO	1	PWM5管脚存在标志 置位时表示存在PWM管脚5。
4	PWM4	RO	1	PWM4管脚存在标志 置位时表示存在PWM管脚4。
3	PWM3	RO	1	PWM3管脚存在标志 置位时表示存在PWM管脚3。
2	PWM2	RO	1	PWM2管脚存在标志 置位时表示存在PWM管脚2。
1	PWM1	RO	1	PWM1管脚存在标志 置位时表示存在PWM管脚1。
0	PWM0	RO	1	PWM0管脚存在标志 置位时表示存在PWM管脚0

寄存器 17: 器件功能4 （DC4）， 偏移量 0x01C

该寄存器提供在系统中可以使用的一系列特性。Stellaris系列使用该寄存器格式来表明在特定器件中可使用下面的特性： 以太网MAC和PHY, GPIO 以及 CCP I/O。该寄存器的格式与 **RCGC2**, **SCGC2** 和 **DCGC2** 时钟控制寄存器以及 **SRCR2** 软件复位控制寄存器一致。

器件功能4 (DC4)

偏移量 0x01C
类型 RO, 复位 0x0000.00FF



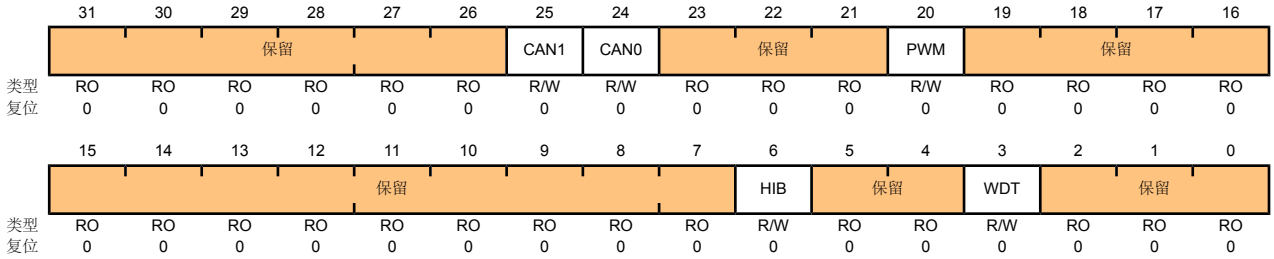
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7	GPIOH	RO	1	GPIO端口H存在标志 置位时表示存在GPIO端口H。
6	GPIOG	RO	1	GPIO端口G存在标志 置位时表示存在GPIO端口G。
5	GPIOF	RO	1	GPIO端口F存在标志 置位时表示存在GPIO端口F
4	GPIOE	RO	1	GPIO端口E存在标志 置位时表示存在GPIO端口E。
3	GPIOD	RO	1	GPIO端口D存在标志 置位时表示存在GPIO端口D。
2	GPIOC	RO	1	GPIO端口C存在标志 置位时表示存在GPIO端口C。
1	GPIOB	RO	1	GPIO端口B存在标志 置位时表示存在GPIO端口B。
0	GPIOA	RO	1	GPIO端口A存在标志 置位时表示存在GPIO端口A。

寄存器 18: 运行模式时钟选通控制寄存器0 （RCGC0）， 偏移量 0x100

该寄存器用来控制时钟选通逻辑。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** 是运行操作的时钟配置寄存器，**SCGC0** 是睡眠操作的时钟配置寄存器，**DCGC0** 是深度睡眠操作的时钟配置寄存器。运行模式时钟配置**(RCC)**寄存器的ACG位置位时，便是系统使用睡眠模式。

运行模式时钟选通控制寄存器0 (RCGC0)

偏移量 0x100
类型 RW, 复位 0x00000040



位/域	名称	类型	复位	描述
31:26	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
25	CAN1	R/W	0	CAN1 时钟选通控制 这个位控制CAN单元1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
24	CAN0	R/W	0	CAN0时钟选通控制 这个位控制CAN单元0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
23:21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
20	PWM	R/W	0	PWM时钟选通控制 这个位控制PWM模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
19:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	HIB	R/W	0	HIB时钟选通控制 这个位控制睡眠模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
5:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

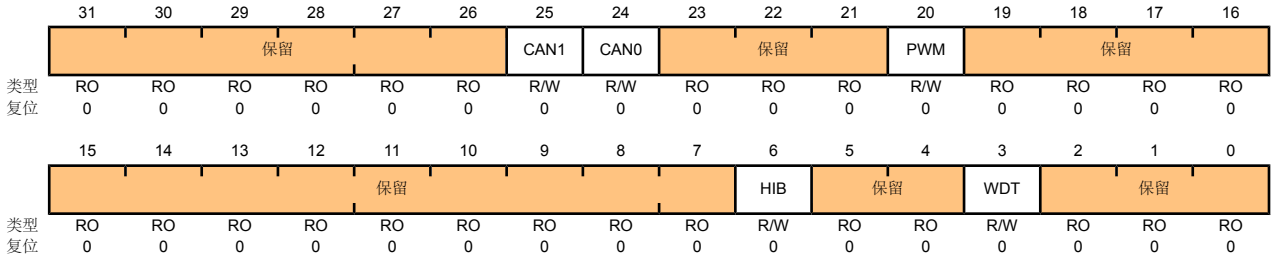
位/域	名称	类型	复位	描述
3	WDT	R/W	0	WDT时钟选通控制 这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

寄存器 19: 睡眠模式时钟选通控制寄存器0 （SCGC0）， 偏移量 0x110

该寄存器用来控制时钟选通逻辑。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** 是运行操作的时钟配置寄存器，**SCGC0** 是睡眠操作的时钟配置寄存器，**DCGC0** 是深度睡眠操作的时钟配置寄存器。当 运行模式时钟配置**(RCC)** 寄存器中的ACG位置位时，表示系统使用睡眠模式。

睡眠模式时钟选通控制寄存器0 (SCGC0)

偏移量 0x110
类型 RW, 复位 0x00000040



位/域	名称	类型	复位	描述
31:26	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
25	CAN1	R/W	0	CAN1时钟选通控制 这个位控制CAN功能单元1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行
24	CAN0	R/W	0	CAN0时钟选通控制 这个位控制CAN功能单元0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
23:21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
20	PWM	R/W	0	PWM时钟选通控制 这个位控制PWM模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
19:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	HIB	R/W	0	HIB时钟选通控制 这个位控制休眠模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
5:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

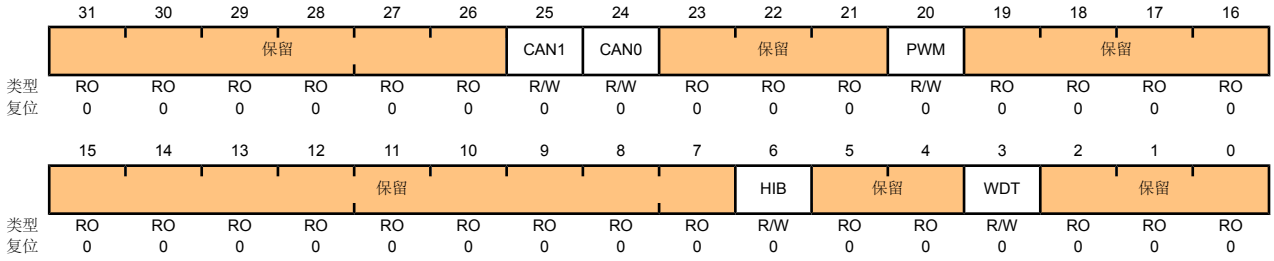
位/域	名称	类型	复位	描述
3	WDT	R/W	0	WDT时钟选通控制 WDT时钟选通控制。这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将产生总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

寄存器 **20:** 深度睡眠模式时钟选通控制寄存器**0** （**DCGC0**）， 偏移量 **0x120**

该寄存器用来控制时钟选通逻辑。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** 是运行操作的时钟配置寄存器，**SCGC0** 是睡眠操作的时钟配置寄存器，**DCGC0** 是深度睡眠操作的时钟配置寄存器。当运行模式时钟配置**(RCC)**寄存器中的ACG位置位时表示系统使用睡眠模式。

深度睡眠模式时钟选通控制寄存器0 (DCGC0)

偏移量 0x120
类型 RW, 复位 0x00000040



位/域	名称	类型	复位	描述
31:26	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
25	CAN1	R/W	0	CAN1时钟选通控制 这个位控制CAN功能单元1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
24	CAN0	R/W	0	CAN0时钟选通控制 这个位控制CAN功能单元0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
23:21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
20	PWM	R/W	0	PWM时钟选通控制 这个位控制PWM模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
19:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	HIB	R/W	0	HIB时钟选通控制 这个位控制睡眠模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。
5:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

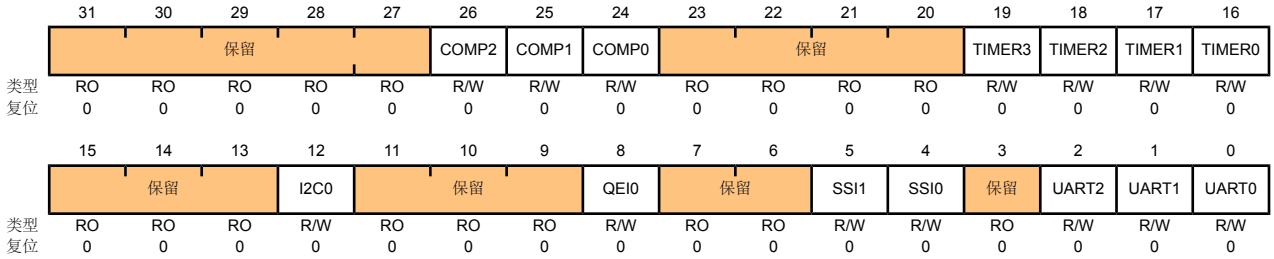
位/域	名称	类型	复位	描述
3	WDT	R/W	0	WDT时钟选通控制 这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

寄存器 21: 运行模式时钟选通控制寄存器1 （RCGC1）， 偏移量 0x104

该寄存器控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC1**是睡眠操作的时钟配置寄存器，**DCGC1**是深度睡眠操作的时钟配置寄存器。当运行模式时钟配置**(RCC)**寄存器的ACG位置位时，表示系统使用睡眠模式。

运行模式时钟选通控制寄存器1 (RCGC1)

偏移量 0x104
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:27	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
26	COMP2	R/W	0	模拟比较器2时钟选通 这个位控制模拟比较器2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
25	COMP1	R/W	0	模拟比较器1时钟选通 这个位控制模拟比较器1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
24	COMP0	R/W	0	模拟比较器0时钟选通 这个位控制模拟比较器0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
23:20	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
19	TIMER3	R/W	0	定时器3时钟选通控制 这个位控制通用定时器模块3的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
18	TIMER2	R/W	0	定时器2时钟选通控制 这个位控制通用定时器模块2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。

位/域	名称	类型	复位	描述
17	TIMER1	R/W	0	定时器1时钟选通控制 这个位控制通用定时器模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
16	TIMER0	R/W	0	定时器0时钟选通控制 这个位控制通用定时器模块0时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
15:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	I2C0	R/W	0	I2C0时钟选通控制 这个位控制I2C模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
11:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
8	QEI0	R/W	0	QEI0时钟选通控制 这个位控制QEI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
7:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
5	SSI1	R/W	0	SSI1时钟选通控制 这个位控制SSI模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
4	SSI0	R/W	0	SSI0时钟选通控制 这个位控制SSI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
2	UART2	R/W	0	UART2时钟选通控制 这个位控制UART模块2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
1	UART1	R/W	0	UART1时钟选通控制 这个位控制UART模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。

位/域	名称	类型	复位	描述
0	UART0	R/W	0	UART0时钟选通控制 这个位控制UART模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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: 睡眠模式时钟选通控制寄存器1 （SCGC1）， 偏移量 0x114

该寄存器用来控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC1**是睡眠操作的时钟配置寄存器，**DCGC1**是深度睡眠操作的时钟配置寄存器。当运行模式时钟配置**(RCC)**寄存器的ACG位置位时，表示系统使用睡眠模式。

睡眠模式时钟选通控制寄存器1 (SCGC1)

偏移量 0x114
类型 RW, 复位 0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	保留					COMP2	COMP1	COMP0	保留				TIMER3	TIMER2	TIMER1	TIMER0
类型	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	保留			I2C0	保留			QEI0	保留		SSI1	SSI0	保留	UART2	UART1	UART0
类型	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位/域	名称	类型	复位	描述
31:27	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
26	COMP2	R/W	0	模拟比较器2时钟选通 这个位控制模拟比较器2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
25	COMP1	R/W	0	模拟比较器1时钟选通 这个位控制模拟比较器1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
24	COMP0	R/W	0	模拟比较器0时钟选通 这个位控制模拟比较器0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
23:20	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
19	TIMER3	R/W	0	定时器3时钟选通控制 这个位控制通用定时器模块3的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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。
18	TIMER2	R/W	0	定时器2时钟选通控制 该位控制着通用定时器模块2的时钟选通。如果该位置位，则功能单元接收时钟并运行。否则，功能单元不使用时钟并禁能。如果功能单元不使用时钟，则对功能单元执行读或写操作将产生总线错误。

位/域	名称	类型	复位	描述
17	TIMER1	R/W	0	定时器1时钟选通控制 这个位控制通用定时器模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
16	TIMER0	R/W	0	定时器0时钟选通控制 这个位控制通用定时器模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
15:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	I2C0	R/W	0	I2C0时钟选通控制 这个位控制I2C模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
11:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
8	QEI0	R/W	0	QEI0时钟选通控制 这个位控制QEI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
7:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
5	SSI1	R/W	0	SSI1时钟选通控制 这个位控制SSI模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
4	SSI0	R/W	0	SSI0时钟选通控制 这个位控制SSI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
2	UART2	R/W	0	UART2时钟选通控制 这个位控制UART模块2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
1	UART1	R/W	0	UART1时钟选通控制 这个位控制UART模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

位/域	名称	类型	复位	描述
0	UART0	R/W	0	UART0时钟选通控制 这个位控制UART模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

寄存器 **23:** 深度睡眠模式时钟选通控制寄存器**1** （**DCGC1**）， 偏移量 **0x124**

该寄存器用来控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC1**是睡眠操作的时钟配置寄存器，**DCGC1**是深度睡眠操作的时钟配置寄存器。当 运行模式时钟配置**(RCC)**寄存器的ACG位置位时，表示系统使用睡眠模式。

深度睡眠模式时钟选通控制寄存器1 (DCGC1)

偏移量 0x124
类型 RW, 复位 0x00000000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
域	保留					COMP2	COMP1	COMP0	保留				TIMER3	TIMER2	TIMER1	TIMER0
类型	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
域	保留			I2C0	保留			QEI0	保留		SSI1	SSI0	保留	UART2	UART1	UART0
类型	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位/域	名称	类型	复位	描述
31:27	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
26	COMP2	R/W	0	模拟比较器2时钟选通 这个位控制模拟比较器2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
25	COMP1	R/W	0	模拟比较器1时钟选通 这个位控制模拟比较器1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
24	COMP0	R/W	0	模拟比较器0时钟选通 这个位控制模拟比较器0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
23:20	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
19	TIMER3	R/W	0	定时器3时钟选通控制 这个位控制通用定时器模块3的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
18	TIMER2	R/W	0	定时器2时钟选通控制 这个位控制通用定时器模块2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

位/域	名称	类型	复位	描述
17	TIMER1	R/W	0	定时器1时钟选通控制 这个位控制通用定时器模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
16	TIMER0	R/W	0	定时器0时钟选通控制 这个位控制通用定时器模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
15:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	I2C0	R/W	0	I2C0时钟选通控制 这个位控制I2C模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
11:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
8	QEI0	R/W	0	QEI0时钟选通控制 这个位控制QEI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
7:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
5	SSI1	R/W	0	SSI1时钟选通控制 这个位控制SSI模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
4	SSI0	R/W	0	SSI0时钟选通控制 这个位控制SSI模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
2	UART2	R/W	0	UART2时钟选通控制 这个位控制UART模块2的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
1	UART1	R/W	0	UART1时钟选通控制 这个位控制UART模块1的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

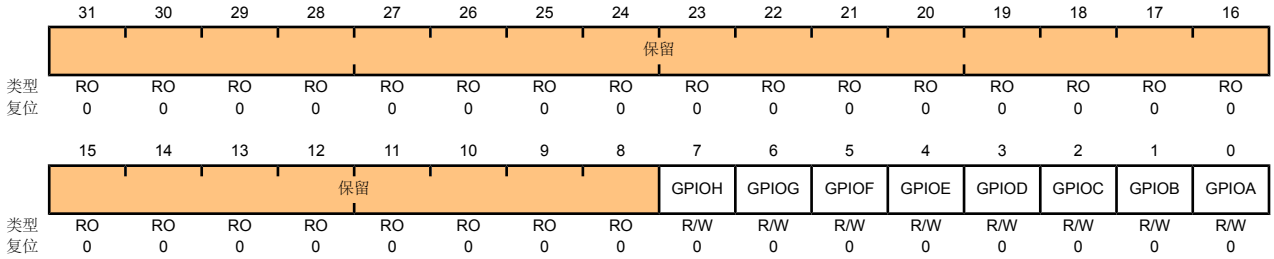
位/域	名称	类型	复位	描述
0	UART0	R/W	0	UART0时钟选通控制 这个位控制UART模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。

寄存器 24: 运行模式时钟选通控制寄存器2 （RCGC2）， 偏移量 0x108

该寄存器用来控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC2**是睡眠操作的时钟配置寄存器，**DCGC2**是深度睡眠操作的时钟配置寄存器。当 运行模式时钟配置 **(RCC)** 寄存器的ACG位置位时，表示系统使用睡眠模式。

运行模式时钟选通控制寄存器2 (RCGC2)

偏移量 0x108
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7	GPIOH	R/W	0	端口H时钟选通控制 这个位控制端口H的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
6	GPIOG	R/W	0	端口G时钟选通控制 这个位控制端口G的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
5	GPIOF	R/W	0	端口F时钟选通控制 这个位控制端口F的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
4	GPIOE	R/W	0	端口E时钟选通控制 这个位控制端口E的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
3	GPIOD	R/W	0	端口D时钟选通控制 这个位控制端口D的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

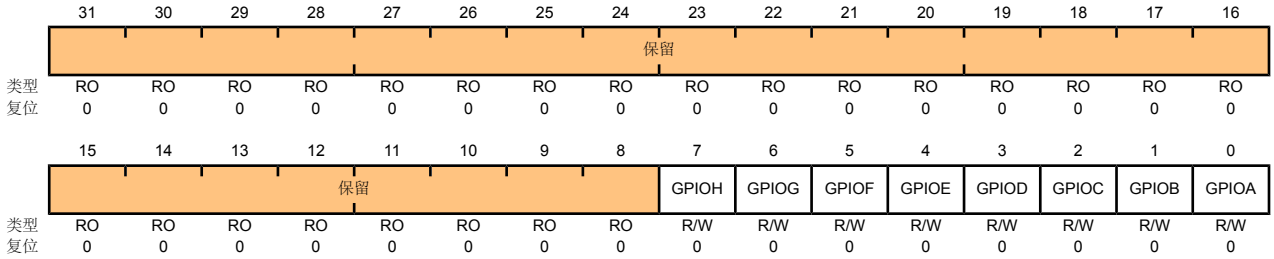
位/域	名称	类型	复位	描述
2	GPIOC	R/W	0	端口C时钟选通控制 这个位控制端口C的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
1	GPIOB	R/W	0	端口B时钟选通控制 这个位控制端口B的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
0	GPIOA	R/W	0	端口A时钟选通控制 这个位控制端口A的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。

寄存器 25: 睡眠模式时钟选通控制寄存器2 （SCGC2）， 偏移量 0x118

该寄存器用来控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC2**是睡眠操作的时钟配置寄存器，**DCGC2**是深度睡眠操作的时钟配置寄存器。当运行模式时钟配置**(RCC)**寄存器的ACG位置位时，表示系统使用睡眠模式。

睡眠模式时钟选通控制寄存器2 (SCGC2)

偏移量 0x118
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7	GPIOH	R/W	0	端口H时钟选通控制 这个位控制端口H的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
6	GPIOG	R/W	0	端口G时钟选通控制 这个位控制端口G的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
5	GPIOF	R/W	0	端口F时钟选通控制 这个位控制端口F的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
4	GPIOE	R/W	0	端口E时钟选通控制 这个位控制端口E的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
3	GPIOD	R/W	0	端口D时钟选通控制 这个位控制端口D的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

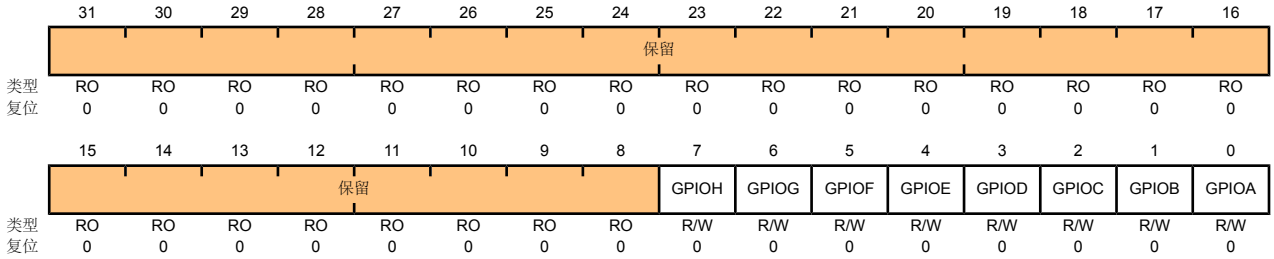
位/域	名称	类型	复位	描述
2	GPIOC	R/W	0	端口C时钟选通控制 这个位控制端口C的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
1	GPIOB	R/W	0	端口B时钟选通控制 这个位控制端口B的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
0	GPIOA	R/W	0	端口A时钟选通控制 这个位控制端口A的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误

寄存器 26: 深度睡眠模式时钟选通控制寄存器2 （DCGC2）， 偏移量 0x128

该寄存器用来控制时钟选通逻辑。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**是运行操作的时钟配置寄存器，**SCGC2**是睡眠操作的时钟配置寄存器，**DCGC2**是深度睡眠操作的时钟配置寄存器。当 运行模式时钟配置**(RCC)**寄存器的ACG位置位时，表示系统使用睡眠模式。

深度睡眠模式时钟选通控制寄存器2 (DCGC2)

偏移量 0x128
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7	GPIOH	R/W	0	端口H时钟选通控制 这个位控制端口H的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
6	GPIOG	R/W	0	端口G时钟选通控制 这个位控制端口G的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
5	GPIOF	R/W	0	端口F时钟选通控制 这个位控制端口F的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
4	GPIOE	R/W	0	端口E时钟选通控制 这个位控制端口E的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。
3	GPIOD	R/W	0	端口D时钟选通控制 这个位控制端口D的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。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错误。

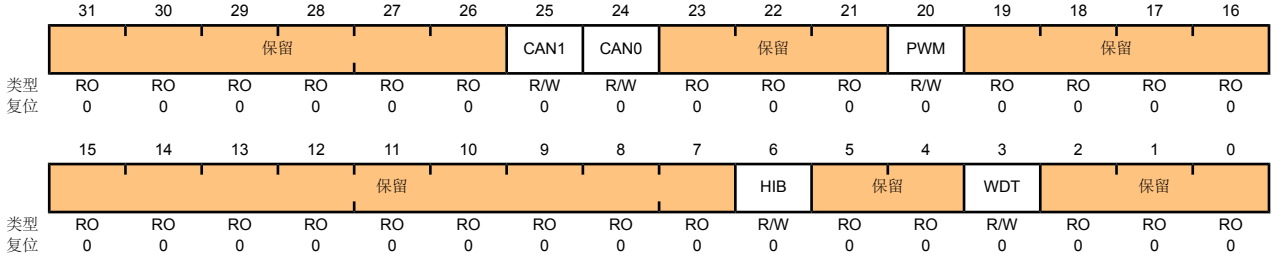
位/域	名称	类型	复位	描述
2	GPIOC	R/W	0	端口C时钟选通控制 这个位控制端口C的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
1	GPIOB	R/W	0	端口B时钟选通控制 这个位控制端口B的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
0	GPIOA	R/W	0	端口A时钟选通控制 这个位控制端口A的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。

寄存器 27: 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)

偏移量 0x040
类型 RW, 复位 0x00000000



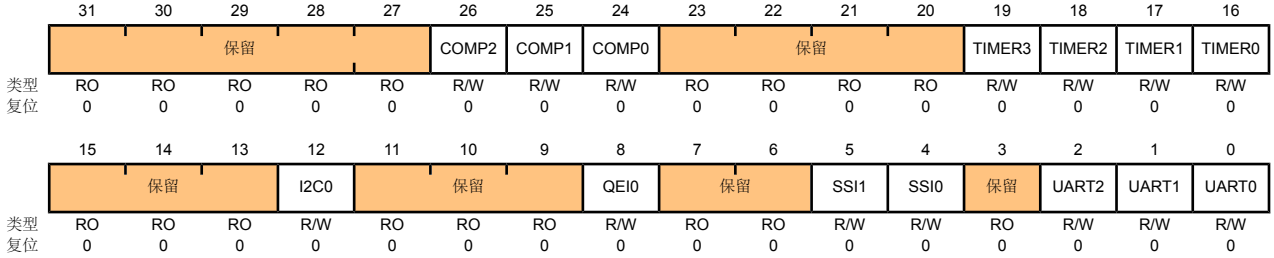
位/域	名称	类型	复位	描述
31:26	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
25	CAN1	R/W	0	CAN1复位控制 CAN功能单元1的复位控制。
24	CAN0	R/W	0	CAN0复位控制 CAN功能单元0的复位控制。
23:21	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
20	PWM	R/W	0	PWM复位控制 PWM模块的复位控制。
19:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	HIB	R/W	0	HIB复位控制 休眠模块的复位控制。
5:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
3	WDT	R/W	0	WDT复位控制 看门狗单位的复位控制。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

寄存器 28: 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)

偏移量 0x044
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:27	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
26	COMP2	R/W	0	模拟比较器2复位控制 模拟比较器2的复位控制。
25	COMP1	R/W	0	模拟比较器1复位控制 模拟比较器1的复位控制。
24	COMP0	R/W	0	模拟比较器0复位控制 模拟比较器0的复位控制。
23:20	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
19	TIMER3	R/W	0	定时器3复位控制。 通用定时器模块3的复位控制。
18	TIMER2	R/W	0	定时器2复位控制 通用定时器模块2的复位控制。
17	TIMER1	R/W	0	定时器1复位控制 通用定时器模块1的复位控制。
16	TIMER0	R/W	0	定时器0复位控制。 通用定时器模块0的复位控制。
15:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	I2C0	R/W	0	I2C0复位控制 I2C功能单元0的复位控制。
11:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

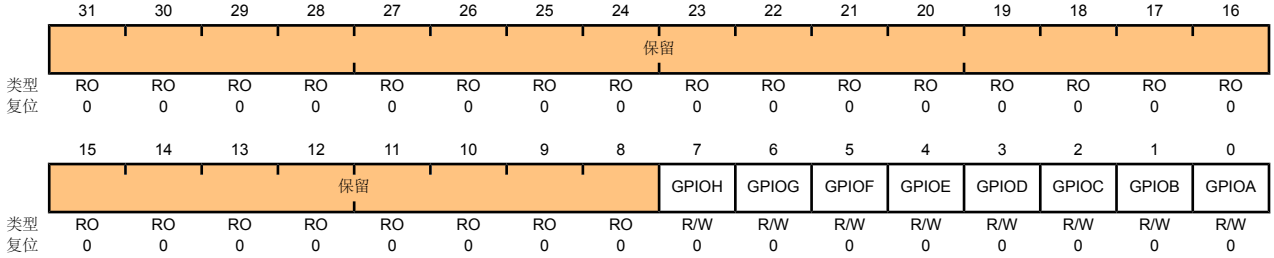
位/域	名称	类型	复位	描述
8	QEI0	R/W	0	QEI0复位控制 QEI功能单元0的复位控制。
7:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
5	SSI1	R/W	0	SSI1复位控制 SSI功能单元1的复位控制。
4	SSI0	R/W	0	SSI0复位控制 SSI功能单元0的复位控制。
3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
2	UART2	R/W	0	UART2复位控制 UART功能单元2的复位控制。
1	UART1	R/W	0	UART1复位控制 UART功能单元1的复位控制。
0	UART0	R/W	0	UART0复位控制 UART功能单元0的复位控制。

寄存器 **29:** 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)

偏移量 0x048
类型 RW, 复位 0x00000000



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7	GPIOH	R/W	0	端口H复位控制 GPIO端口H的复位控制。
6	GPIOG	R/W	0	端口G复位控制 GPIO端口G的复位控制
5	GPIOF	R/W	0	端口F复位控制 GPIO端口F的复位控制。
4	GPIOE	R/W	0	端口E复位控制 GPIO端口E的复位控制。
3	GPIOD	R/W	0	端口D复位控制 GPIO端口D的复位控制。
2	GPIOC	R/W	0	端口C复位控制 GPIO端口C的复位控制。
1	GPIOB	R/W	0	端口B复位控制 GPIO端口B的复位控制。
0	GPIOA	R/W	0	端口A复位控制 GPIO端口A的复位控制。

7 休眠模块

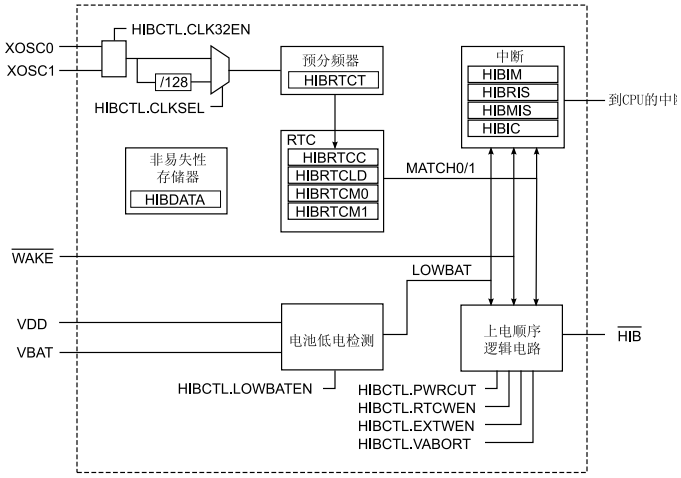
休眠模块管理微控制器其它功能模块电源的解除和恢复，提供一种降低功耗的方法。 当处理器和外设空闲时，电源可以完全解除，只维持休眠模块的供电。 电源可以因为一个外部信号而恢复，也可以利用内置实时时钟（RTC），在某个时刻被恢复。 休眠模块可单独通过电池或辅助电源提供电源。

休眠模块具有以下特性：

- 到个别外部稳压器的电源切换逻辑
- 用作外部信号唤醒的专门管脚
- 低电池检测、发出信号和中断发生
- 32位实时计数器（RTC）
- 2个32位的RTC匹配寄存器，用作定时唤醒和中断产生
- 时钟源来自一个32.768kHz的外部振荡器或一个4.194304MHz的晶体
- RTC预分频器调整，对时钟速率进行良好地调节
- 64个32位字的非易失性存储器
- 可编程的RTC匹配、外部唤醒和低电池电压事件的中断

7.1 方框图

图 **7-1.** 休眠模块方框图



7.2 功能描述

休眠模块用一个使能信号（HIB）来控制处理器的电源，使能信号通知外部稳压器停止运行。 休眠模块电源动态确定。 休眠模块的电源是较大的主电压源（VDD）或电池/辅助电压源（VBAT）。 表决电路指示较大的电源且内部电源切换选择相关的电压源。 休眠模块也具有一个独立的时钟源来保

持实时时钟（RTC）。 一旦进入休眠，模块就通知外部稳压器返回到外部管脚（WAKE）有效或内部RTC达到某个特定值时的电压。 休眠模块也可以检测到电池电压何时过低，也可以选择在电池电压过低时阻止进入休眠。

电源从上电降到使代码执行的这段时间定义成稳压器导通时间（规定最大为$t_{HIB_TO_VDD}$）加上正常的芯片POR时间（见"休眠模块" 在 472页）。

7.2.1 寄存器访问时序

由于休眠模块有一个独立的时钟域，因此，某些寄存器被写入时必须间隔一个访问之间的时间差（timing gap）。 延时时间为$t_{HIB_REG_WRITE}$，所以，软件必须保证，在连续写某些休眠寄存器之间，或者在后面紧跟着的是读这些寄存器的写休眠寄存器的操作之间，插入一个$t_{HIB_REG_WRITE}$的延时。 对连续读休眠模块的时序没有任何限制。 关于哪些寄存器要遵守这种时序限制的详细情况请参考"寄存器描述" 在 115页。

7.2.2 时钟源

即使RTC特性不被使用，休眠模块的时钟也必须由外部时钟源提供。 外部振荡器或时钟可以用来给休眠模块提供时钟。 使用晶体时，一个4.194304MHz的晶体被连接到 XOSC0 和XOSC1管脚之间。 这个时钟信号在内部进行128分频来产生32.768-kHz的时钟基准。 为了使用一个更精确的时钟源，可以将一个32.768kHz的振荡器连接到XOSC0管脚。

时钟源通过置位**HIBCTL**寄存器的CLK32EN位来使能。 时钟源的类型可以这样选择：将CLKSEL位设置为0来选择一个 4.194304MHz的时钟源；将CLKSEL位设置为1来选择一个32.768kHz的时钟源。 如果CLKSEL位被设为0，则输入时钟被128分频来产生一个 32.768kHz 的时钟源。 如果晶体被用作时钟源，那么在置位CLK32EN位之后和执行任何其它休眠模块寄存器的访问之前，软件都必须留出一个t_{XOSC_SETTLE}的延时。 延时保证了晶体的上电和稳定。 如果振荡器被用作时钟源，则不需要延时。

7.2.3 电池管理

休眠模块可以由电池或辅助电源单独供电。 模块可以监测电池的电压电平并检测到电压何时变得过低。 当电池电压过低时产生中断。 模块也可以配置成 在电池电压过低时不进入休眠模式。

注意：休眠模块消耗拥有更高电压的电源（VBAT 或 VDD）的电流。 因此，重要的是将电路设计成确保VDD的电压更高、VBAT处于额定条件下，否则，即使VDD可用，休眠模块也只消耗电池的电流。

休眠模块也可以通过置位**HIBCTL**寄存器的LOWBATEN位配置成对低电池电压条件进行检测。 在这种配置中，电池电压过低时**HIBRIS**寄存器的LOWBAT位将被置位。 如果VABORT位也置位，那么在检测到低电池电压时模块就被阻止进入休眠模式。 模块也可以配置成在低电池电压条件出现时产生中断（见"中断和状态" 在 113页）。

7.2.4 实时时钟

休眠模块包含一个32位的计数器，该计数器利用合适的时钟源和配置，每秒递增一次（见 "时钟源" 在 112页）。 32.768-kHz的时钟信号被馈送到一个预分频器寄存器，该寄存器对32.768kHz的时钟周期进行递减计数来获得每秒一次的RTC时钟速率。 通过使用预分频器调整寄存器，可以调节速率来对时钟源的不精确进行补偿。 这个寄存器有一个指定值0x7FFF，每64秒只有1秒钟使用这个寄存器，以此分频输入时钟。 这就允许软件通过调节预分频调整寄存器，使寄存器从0x7FFF开始向上增加或向下递减来对时钟速率进行良好的校准。为了降低RTC的速率，预分频器调整应当从0x7FFF向上调节；而为了加快RTC的速率，预分频器调整应当从0x7FFF向下调节。

休眠模块包含2个32位的匹配寄存器，它们的值被用来与RTC计数器的值进行比较。 匹配寄存器可以用来将处理器从休眠模式唤醒，或者，在处理器不处于休眠模式时用来向处理器产生中断。

必须利用**HIBCTL**寄存器的RTCEN位使能RTC。 RTC的值可以随时通过写**HIBRTCLD**寄存器来设置。 预分频器调整可以通过读和写**HIBRTCT**寄存器来调节。 预分频器每隔64秒使用这个寄存器一次来

对时钟速率进行调节。 2个匹配寄存器可以通过写**HIBRTCM0**和 **HIBRTCM1**寄存器来设置。 通过使用中断寄存器，RTC可以配置成产生中断（见“中断和状态” 在 113页）。

7.2.5 非易失性存储器

休眠模块包含64个32位字的存储器，其内容在休眠过程中保持不变。在休眠过程中这个存储器由电池或辅助电源供电。 处理器软件可以在休眠之前将状态信息保存到存储器中，然后在唤醒时将状态恢复。 非易失性存储器可以通过**HIBDATA**寄存器来访问。

7.2.6 功率控制

休眠模块通过HIB管脚的使用来控制到处理器的电源，该管脚与外部稳压器的使能信号相连，给微控制器提供3.3V和/或2.5V的电压。 当休眠模块使得HIB信号有效时，外部稳压器停止工作，不再给微控制器供电。 休眠模块保持VBAT电源的供电，它可以是一个电池或一个辅助电源。 微控制器通过置位**HIBCTL**寄存器的HIBREQ位来启动休眠模式。 在启动休眠模式之前，必须将唤醒条件配置成外部WAKE管脚或RTC匹配。

通过置位**HIBCTL** 寄存器的PINWEN位，休眠模块配置成由外部WAKE管脚唤醒。 休眠模块通过置位RTCWEN位配置成由RTC匹配唤醒。 这两个位中的其中一个位或全部两个位可以在进入休眠之前被置位。 WAKE 管脚包括一个弱内部上拉。 注意HIB和WAKE管脚使用休眠模块的内部电源作为逻辑1参考。

当休眠模式唤醒时，微控制器将“看到”一个正常的上电复位。 通过检查原始中断状态寄存器（见“中断和状态” 在 113页）和查找非易失性存储器中的状态数据（见 “非易失性存储器” 在 113页），微控制器可以检测到上电是由于休眠的唤醒造成的。

当HIB信号无效时，使能外部稳压器，外部稳压器必须在$t_{HIB_TO_VDD}$时间内到达操作电压。

7.2.7 中断和状态

当以下条件出现时，休眠模块可以产生中断：

- WAKE管脚有效
- RTC匹配
- 检测到低电池电压

所有的中断相或后发送到中断控制器，因此在休眠模块只能在给定的时间向控制器产生一个中断请求。 软件中断处理程序可以通过读**HIBMIS**寄存器服务几个中断事件。 软件也可以通过读 **HIBRIS** 寄存器（该寄存器显示了所有的挂起事件）来随时读取休眠模块的状态。 上电时可以利用这个寄存器来判断唤醒条件是否正在挂起，这是向软件指示休眠唤醒的出现。

可以触发中断的事件通过置位**HIBIM**寄存器中相应的位来配置。 挂起的中断可以通过写**HIBIC**寄存器中相应的位来清除。

7.3 初始化和配置

休眠模块可以配置成几种不同组合。 下面各小节给出了不同情况下推荐的编程序列。 下面的例子中，假设使用的是32.768kHz的振荡器，因此，**HIBCTL**寄存器的位2（CLKSEL）总是显示被设置为1。 如果改为使用4.194304MHz的晶体，那么CLKSEL位保持清零。 由于休眠模块运行在32 kHz的频率下，与系统其它部分不同步，所以，在写某些寄存器之后，软件必须留有一个$t_{HIB_REG_WRITE}$的延时（见“寄存器访问时序” 在 112页）。 需要延时的寄存器在表 7-1 在 115页中标有一个脚注。

7.3.1 初始化

即使RTC将不被使用，时钟源也必须先使能。 如果使用的是4.194304MHz的晶体，则执行以下步骤：

1. 向偏移量为0x10的**HIBCTL**寄存器写入0x40来使能晶体和选择128分频的输入通路。
2. 在执行休眠模块相关的任何其它操作之前，等待一段 t_{XOSC_SETTLE}的时间，以便晶体上电和稳定。

如果使用的是32.678kHz的振荡器，则执行以下步骤：

1. 向偏移量为0x10的**HIBCTL**寄存器写入0x44来使能振荡器输入。
2. 不需要延时。

以上操作只需要在整个系统第一次初始化时执行。 如果处理器由于一个休眠唤醒而被加电，那么休眠模块就已经上电，这时就无需执行以上步骤。 软件可以通过检查**HIBCTL**寄存器的CLK32EN位检测到休眠模块和时钟已经被供电。

7.3.2 RTC匹配功能（未休眠）

需要执行以下步骤来使用休眠模块的RTC匹配功能：

1. 向偏移量为0x004或0x008的一个**HIBRTCMn**寄存器写入要求的RTC匹配值。
2. 将要求的RTC装载值写入偏移量为0x00C的**HIBRTCLD** 寄存器。
3. 将偏移量为0x014的**HIBIM**寄存器中的RTCALT0和RTCALT1位（位1:0）置位。
4. 向偏移量为0x010的**HIBCTL**寄存器写入0x0000.0041来使能RTC开始计数。

7.3.3 RTC 匹配/休眠唤醒

需要执行以下步骤来使用休眠模块的RTC匹配和唤醒功能：

1. 写所需的RTC匹配值到**HIBRTCMn**寄存器，偏移量为0x004或0x008。
2. 将要求的RTC装载值写入偏移量为0x00C的**HIBRTCLD** 寄存器。
3. 将在电源切断过程中要保留的任何数据写入偏移量为0x030-0x12C的**HIBDATA**寄存器中。
4. 通过写0x0000.004F到偏移量为0x010的**HIBCTL** 寄存器来设置RTC匹配唤醒和启动休眠序列。

7.3.4 外部休眠唤醒

在外部WAKE管脚作为微控制器唤醒源的情况下，需要执行以下步骤来使用休眠模块：

1. 将在电源切断过程中要保留的任何数据写入偏移量为0x030-0x12C的**HIBDATA**寄存器中。
2. 通过向偏移量为0x010的**HIBCTL**寄存器写入0x0000.0056来使能外部唤醒和启动休眠序列。

7.3.5 RTC/外部休眠唤醒

1. 写所需的RTC匹配值到**HIBRTCMn**寄存器，偏移量为0x004或0x008。
2. 将要求的RTC装载值写入偏移量为0x00C的**HIBRTCLD** 寄存器。

3. 将在电源切断过程中要保留的任何数据写入偏移量为0x030-0x12C的**HIBDATA**寄存器中。
4. 通过向偏移量为0x010的**HIBCTL**寄存器写入0x0000.005F来设置RTC匹配/外部唤醒和启动休眠序列。

7.4 寄存器映射

表 7-1 在 115页列出了休眠寄存器。 给出的所有地址都是相对0x400F.C000的休眠模块基址而言的。

注意: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**和**HIBDATA**都在休眠模块时钟域并在写访问之间需要 $t_{HIB_REG_WRITE}$的延时。 见“寄存器访问时序” 在 112页。

表 **7-1.** 休眠模块 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	HIBRTCC	RO	0x0000.0000	休眠RTC计数器	116
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	休眠RTC匹配0	117
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	休眠RTC匹配1	118
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	休眠RTC装载	119
0x010	HIBCTL	R/W	0x0000.0000	休眠控制	120
0x014	HIBIM	R/W	0x0000.0000	休眠中断屏蔽	122
0x018	HIBRIS	RO	0x0000.0000	休眠原始中断状态	123
0x01C	HIBMIS	RO	0x0000.0000	休眠可屏蔽中断的状态	124
0x020	HIBIC	R/W1C	0x0000.0000	休眠中断清除	125
0x024	HIBRTCT	R/W	0x0000.7FFF	休眠RTC调整	126
0x030-0x12C	HIBDATA	R/W	0x0000.0000	休眠数据	127

7.5 寄存器描述

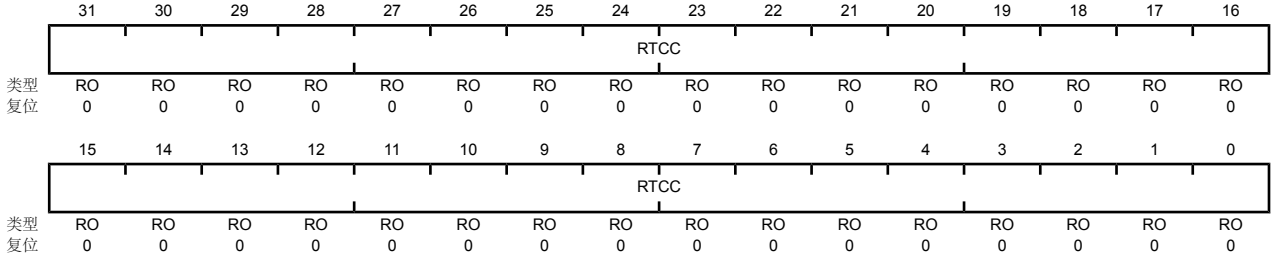
本节的剩余部分按照地址偏移量的数字顺序列出和描述了休眠模块寄存器。

寄存器 1: 休眠RTC计数器（HIBRTCC），偏移量 0x000

该寄存器是RTC计数器的当前32位值。

休眠RTC计数器 (HIBRTCC)

偏移量 0x000
类型 RO, 复位 0x0000.0000



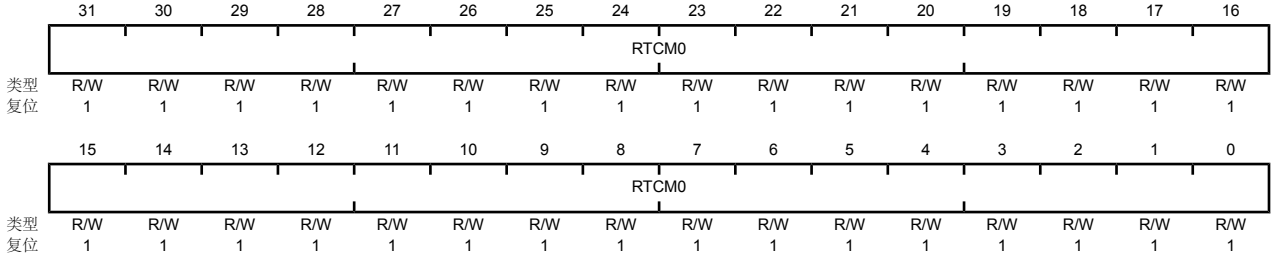
位/域	名称	类型	复位	描述
31:0	RTCC	RO	0x0000.0000	RTC计数器 读操作返回32位的计数器值。 这个寄存器只可读。 使用**HIBRTCLD**寄存器来改变本寄存器的值。

寄存器 2: 休眠RTC匹配0（HIBRTCM0），偏移量 0x004

这个寄存器是RTC计数器的32位匹配0寄存器。

休眠RTC匹配0 (HIBRTCM0)

偏移量 0x004
类型 R/W, 复位 0xFFFF.FFFF



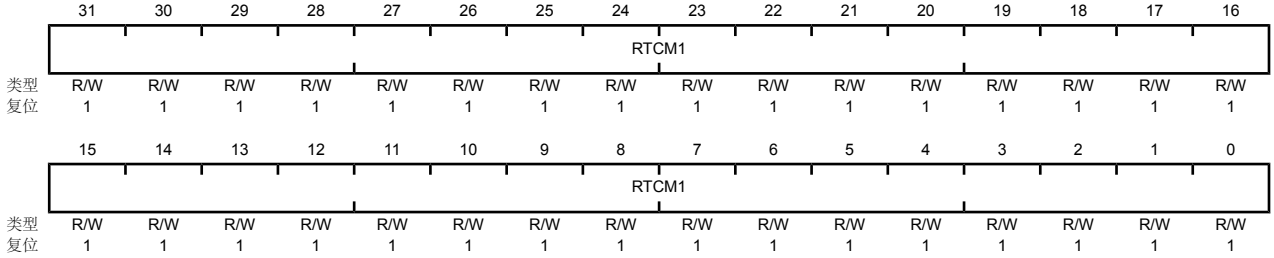
位/域	名称	类型	复位	描述
31:0	RTCM0	R/W	0xFFFF.FFFF	RTC匹配0 写时将值装入RTC匹配寄存器。 读时返回当前的匹配值。

寄存器 3: 休眠RTC匹配1（HIBRTCM1），偏移量 0x008

这个寄存器是RTC计数器的32位匹配1寄存器。

休眠RTC匹配1 (HIBRTCM1)

偏移量 0x008
类型 R/W, 复位 0xFFFF.FFFF



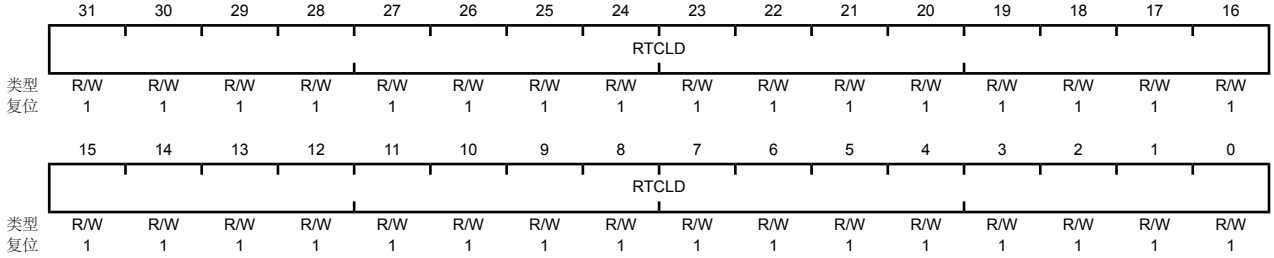
位/域	名称	类型	复位	描述
31:0	RTCM1	R/W	0xFFFF.FFFF	RTC匹配1 写时将值装入RTC匹配寄存器。 读时返回当前的匹配值。

寄存器 4: 休眠RTC装载（HIBRTCLD），偏移量 0x00C

这个寄存器的内容是装入RTC计数器的32位值。

休眠RTC装载 (HIBRTCLD)

偏移量 0x00C
类型 R/W, 复位 0xFFFF.FFFF

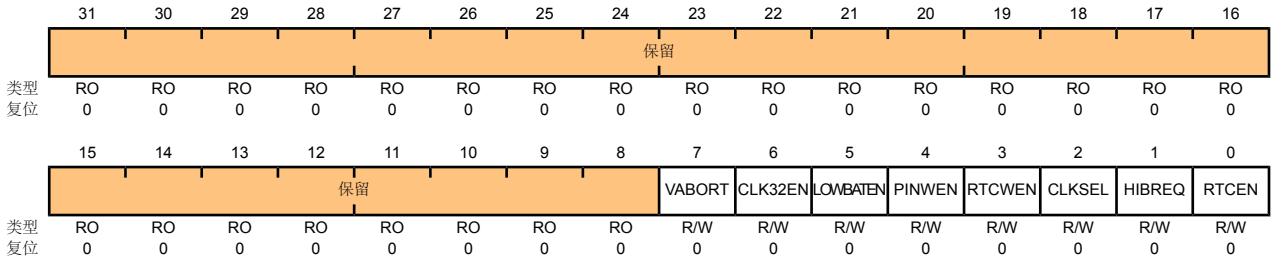


位/域	名称	类型	复位	描述
31:0	RTCLD	R/W	0xFFFF.FFFF	RTC装载 写载入当前值到RTC计数器(RTCC)。 读时返回32位的装载值。

寄存器 5: 休眠控制（HIBCTL），偏移量 0x010

这个寄存器是休眠模块的控制寄存器。

休眠控制 (HIBCTL)
偏移量 0x010
类型 R/W, 复位 0x0000.0000



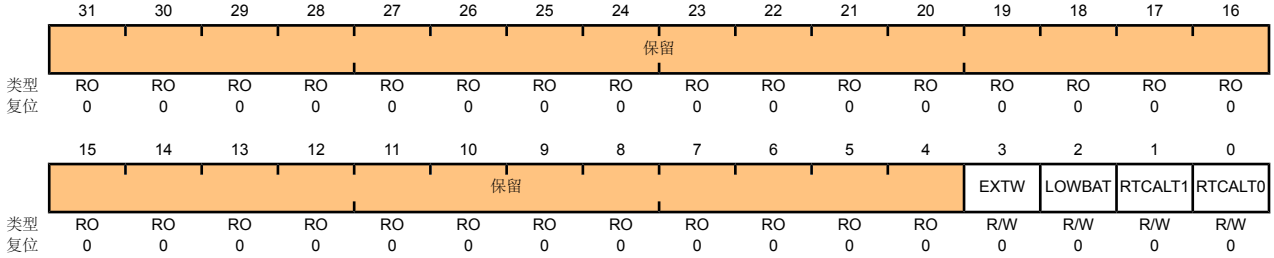
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	VABORT	R/W	0	电源切断终止使能 0：在低电池电压报警过程中切断电源。 1：终止切断电源。
6	CLK32EN	R/W	0	32kHz振荡器使能 0：禁能 1：使能 要使用休眠模块，这个位就必须使能。 如果使用的是晶体，则在置位该位后软件应该等待20ms，以便晶体上电和稳定。
5	LOWBATEN	R/W	0	低电池电压监测使能 0：禁能 1：使能 该位置位时使能低电池电压检测。
4	PINWEN	R/W	0	外部WAKE管脚使能 0：禁能 1：使能 当该位置位时，WAKE管脚将对器件进行重新供电。
3	RTCWEN	R/W	0	RTC唤醒使能 0：禁能 1：使能 该位置位时，根据RTC计数器值与对应的匹配寄存器0或1的匹配，RTC匹配事件（RTCM0或 RTCM1）将重新对器件进行供电。
2	CLKSEL	R/W	0	休眠模块时钟选择 0：使用128分频的输出。 选择4MHz的晶体时使用该值。 1：使用原始的输出。 选择32kHz的振荡器时使用该值。

位/域	名称	类型	复位	描述
1	HIBREQ	R/W	0	休眠请求 0：禁能 1：休眠启动 唤醒事件后，该位由软件清零。
0	RTCEN	R/W	0	RTC定时器使能 0：禁能 1：使能

寄存器 6: 休眠中断屏蔽（HIBIM），偏移量 0x014

这个寄存器是休眠模块中断源的中断屏蔽寄存器。

休眠中断屏蔽 (HIBIM)
偏移量 0x014
类型 R/W, 复位 0x0000.0000

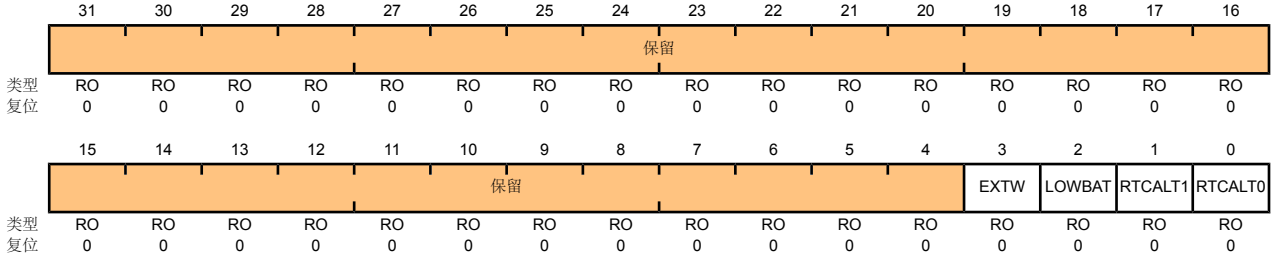


位/域	名称	类型	复位	描述
31:4	保留	RO	0x000.0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	EXTW	R/W	0	外部唤醒中断屏蔽 0：屏蔽 1：不屏蔽
2	LOWBAT	R/W	0	低电池电压中断屏蔽 0：屏蔽 1：不屏蔽
1	RTCALT1	R/W	0	RTC 报警1中断屏蔽 0：屏蔽 1：不屏蔽
0	RTCALT0	R/W	0	RTC报警0中断屏蔽 0：屏蔽 1：不屏蔽

寄存器 7: 休眠原始中断状态（HIBRIS），偏移量 0x018

这个寄存器是休眠模块中断源的原始中断状态。

休眠原始中断状态 (HIBRIS)
偏移量 0x018
类型 RO, 复位 0x0000.0000

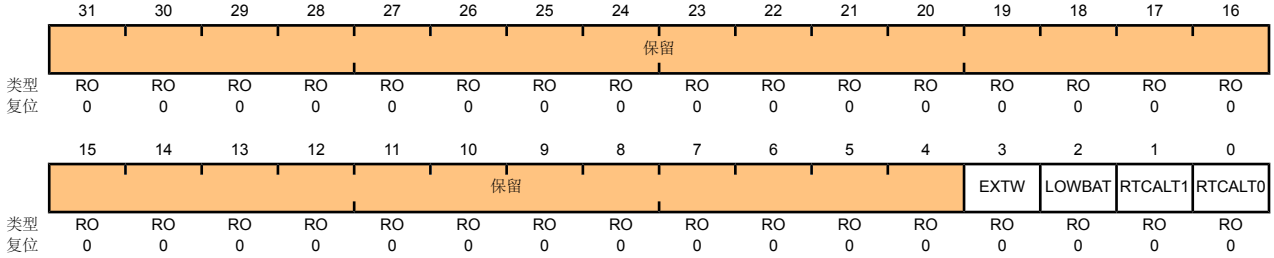


位/域	名称	类型	复位	描述
31:4	保留	RO	0x000.0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	EXTW	RO	0	外部唤醒原始中断状态
2	LOWBAT	RO	0	低电池电压原始中断状态
1	RTCALT1	RO	0	RTC报警1原始中断状态
0	RTCALT0	RO	0	RTC报警0原始中断状态

寄存器 8: 休眠可屏蔽中断的状态（HIBMIS），偏移量 0x01C

这个寄存器是休眠模块中断源的可屏蔽中断的状态。

休眠可屏蔽中断的状态 (HIBMIS)
偏移量 0x01C
类型 RO, 复位 0x0000.0000



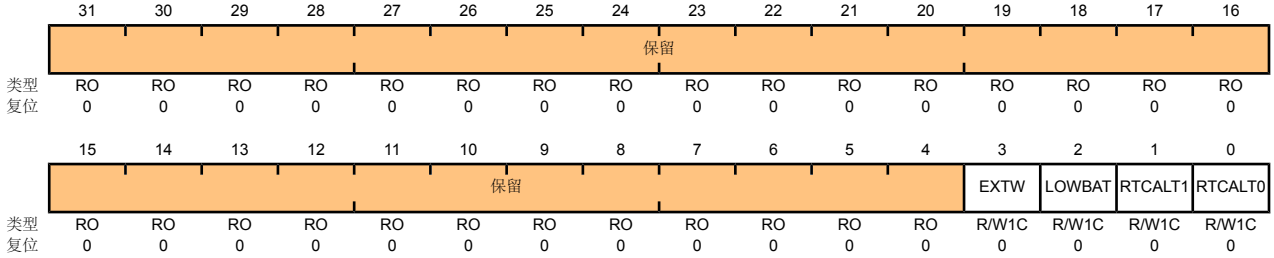
位/域	名称	类型	复位	描述
31:4	保留	RO	0x000.0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	EXTW	RO	0	外部唤醒可屏蔽中断状态
2	LOWBAT	RO	0	低电池电压可屏蔽中断状态
1	RTCALT1	RO	0	RTC报警1可屏蔽中断状态
0	RTCALT0	RO	0	RTC报警0可屏蔽中断状态

寄存器 9: 休眠中断清除（HIBIC），偏移量 0x020

这个寄存器是休眠模块中断源的中断写1清除寄存器。

休眠中断清除 (HIBIC)

偏移量 0x020
类型 R/W1C, 复位 0x0000.0000



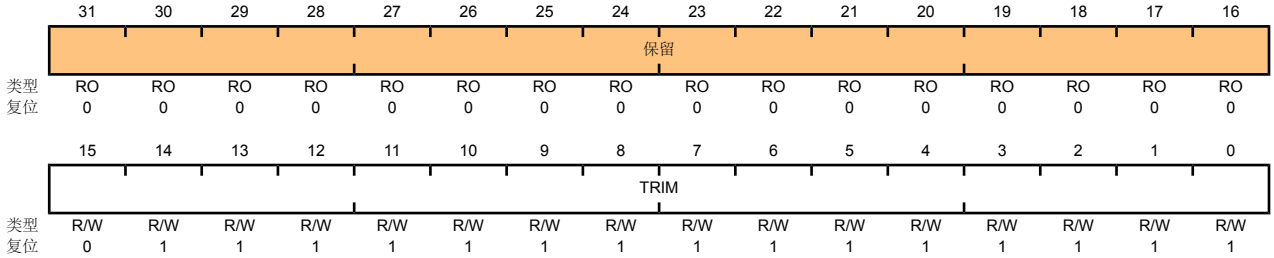
位/域	名称	类型	复位	描述
31:4	保留	RO	0x000.0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	EXTW	R/W1C	0	外部唤醒可屏蔽中断清除 读时返回一个不确定的值。
2	LOWBAT	R/W1C	0	低电池电压可屏蔽中断清除 读时返回一个不确定的值。
1	RTCALT1	R/W1C	0	RTC报警1可屏蔽中断清除 读时返回一个不确定的值。
0	RTCALT0	R/W1C	0	RTC报警0可屏蔽中断清除 读时返回一个不确定的值。

寄存器 **10:** 休眠**RTC**调整（**HIBRTCT**），偏移量 **0x024**

这个寄存器包含用来调整RTC时钟预分频器的值。 它代表的是计算的下溢值，该值被用在调整周期中。 表示成0x7FFF ± N个时钟周期。

休眠RTC调整 (HIBRTCT)

偏移量 0x024
类型 R/W, 复位 0x0000.7FFF



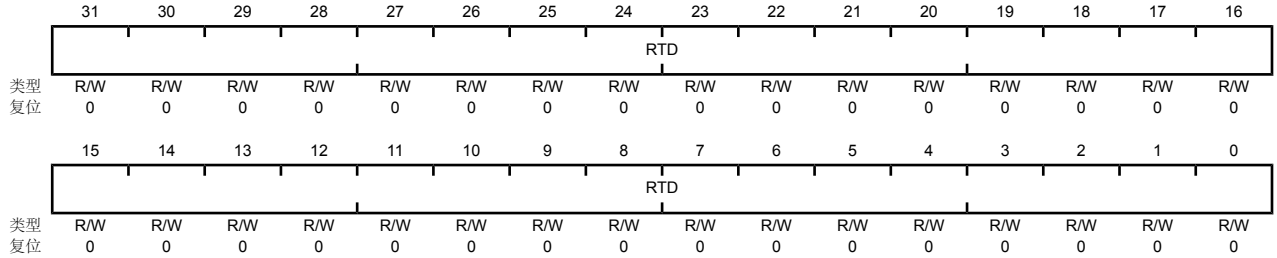
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TRIM	R/W	0x7FFF	RTC调整值 这个值每隔64秒装入到RTC预分频器中。 它被用来调整RTC的速度，补偿时钟源的漂移和不精确。 由软件通过上调或下调默认值0x7FFF来进行补偿。

寄存器 11: 休眠数据（HIBDATA），偏移量 0x030- 0x12C

这个地址空间用作一个 64x32位的存储器（256个字节）。 为了保存任何非易失性状态数据，该地址空间可以由系统处理器来装载，这部分空间在电源切断过程中不会断电。

休眠数据 (HIBDATA)

偏移量 0x030- 0x12C
类型 R/W, 复位 0x0000.0000



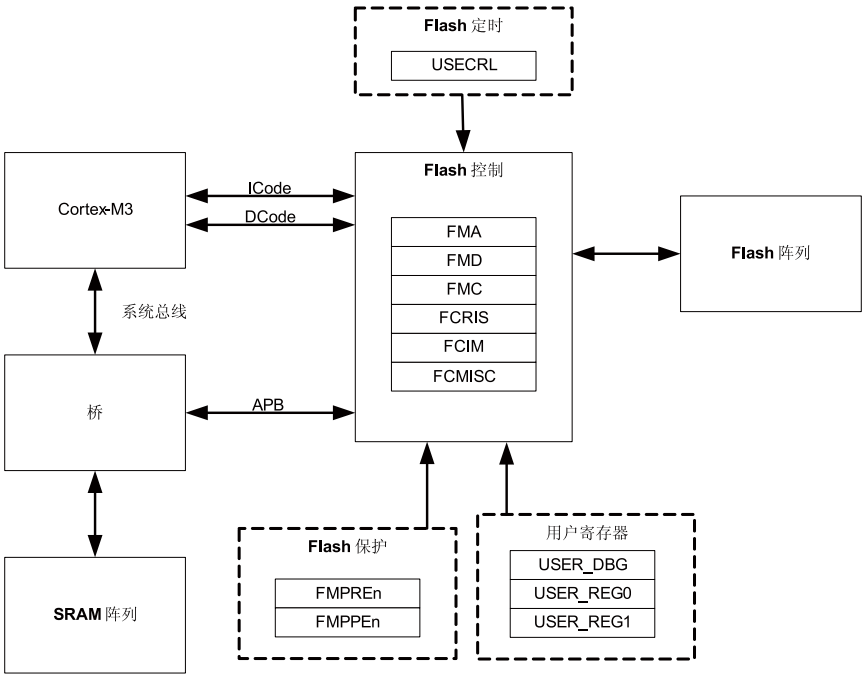
位/域	名称	类型	复位	描述
31:0	RTD	R/W	0x0000.0000	休眠模块非易失性寄存器[63:0]

8 内部存储器

LM3S2950微控制器带有64 KB的bit-banded SRAM和256 KB的Flash存储器。Flash控制器提供了一个友好的用户接口，使Flash编程成为一项简单的任务。在Flash存储器中可应用Flash保护，以2-KB块大小为单位。

8.1 方框图

图 **8-1. Flash**方框图



8.2 功能描述

这一节描述了Flash存储器和SRAM存储器的功能。

8.2.1 SRAM存储器

Stellaris®器件的内部SRAM位于器件存储器映射的地址0x2000.0000。为了减少读－修改－写（RMW）操作的时间，ARM在Cortex-M3处理器中引入了*bit-banding*技术。在bit-banding使能的处理器中，存储器映射的特定区域（SRAM和外设空间）能够使用地址别名，在单个原子操作中访问各个位。

使用下面的公式来计算bit-band别名：

```
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0x2000.1000的位3，则bit-band别名计算如下：

```
0x2200.0000 + (0x1000 * 32) + (3 * 4) = 0x2202.000C
```

通过计算得出的别名地址，对地址0x2202.000C执行读/写操作的指令仅允许直接访问地址0x2000.1000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”。

8.2.2 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。每个32位的字可以被编程为将当前为1的位变为0。这些区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，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机制来读取它的内容，这可以保护区块的内容不被控制器或调试器读取。

有关不使用调节接口用来将代码下载到器件的Flash存储器中的预编程Flash驻留（flash-resident）实用程序见“串行Flash加载程序” 在 480页。

8.2.2.1 Flash存储器时序

Flash的时序是由Flash控制器自动处理的。但是，如此便需要得知系统的时钟速率以便对内部的信号进行精确的计时。为了完成这种计时，必须向Flash控制器提供每微秒的时钟周期数。由软件负责通过**USec**重装（**USECRL**）寄存器用此信息来使Flash控制器保持更新。

复位时，用来配置flash时序的值将被加载到**USECRL**寄存器中，使flash能和器件的最大时钟速率协同工作。如果软件改变系统工作频率，那么在尝试对Flash进行任何修改之前必须将新的操作频率减去1（MHz）装载到**USECRL**中。例如，如果器件正工作在20MHz的频率下，那么必须向**USECRL**寄存器写入值0x13（20-1）。

8.2.2.2 Flash存储器保护

在4对 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。由**FMPPEn**和**FMPREn**寄存器的各个位来控制每种形式的保护策略（每个块一个策略）。

- **Flash**存储器保护编程使能（**FMPPEn**）：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（**FMPREn**）：如果置位，则通过软件或调试器执行或读模块。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，也不能通过DCode总线。

这些策略可以进行组合，如表 8-1 在 129页所示。

表 **8-1. Flash**保护策略组合

FMPPEn	FMPREn	保护
0	0	只执行保护。模块只能被执行，不能被写或擦除。这种模式用来保护代码。
1	0	模块可以被写、擦除或执行，不能被读取。这种组合不可能被使用。
0	1	只读保护。模块可以被读或执行，但不能被写或擦除。这种模块用来锁定模块防止对其进行进一步的修改，但允许对其执行任意的读或执行访问。
1	1	无保护。模块可以被读、擦除、执行或读取。

禁止对受到PE保护的模块尝试编程或擦除访问。可选择产生控制器中断（通过置位**FIM**寄存器的AMASK位）来向软件开发者报警在开发和调试阶段中出现的错误软件操作。

禁止对受到RE保护的模块尝试执行读访问。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报警在开发和调试阶段中出现的错误软件操作。

在**FMPREn**和**FMPPEn**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。而这种改变不是永久性的，除非寄存器已确认（已保存），在那时，位的改变才是永久性的。如果位在从1变为0时没有确认，则可以通过执行上电复位序列来恢复该位。有关编程这些位的详情在“非易失性寄存器编程” 在 130页中讨论。

8.3 Flash存储器的初始化和配置

8.3.1 Flash编程

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

8.3.1.1 执行以下步骤来编程一个**32**位的字：

1. 写源数据到**FMD**寄存器。
2. 写目标地址到**FMA**寄存器。
3. 将Flash写密钥（flash write key）写入**FMC**寄存器,并置位WRITE位（写入0xA442.0001）。
4. 查询**FMC**寄存器，直至WRITE位被清零。

8.3.1.2 执行以下步骤来擦除**1KB**的页：

1. 将页地址写入**FMA**寄存器。
2. 将Flash写密钥写入**FMC**寄存器，并置位ERASE位（写入 0xA442.0002）。
3. 查询**FMC**寄存器，直至ERASE位被清零。

8.3.1.3 执行以下步骤来完成**Flash**的整体擦除：

1. 将Flash写密钥写入**FMC**寄存器，并置位MERASE位（写入0xA442.0004）。
2. 查询**FMC**寄存器，直至MERASE位被清零。

8.3.2 非易失性寄存器编程

本节讨论如何更新驻留在Flash存储器自身内部的寄存器。这些寄存器位于主Flash阵列的一个独立空间，不受擦除或整体擦除操作的影响。它们通过使用**FMC**寄存器的COMT位激活一个写操作来更新。写入**USER_DBG**寄存器的数据在“确认”之前必须被装入**FMD**寄存器。所有其它的寄存器都是可读/写的，它们在确认到到非易失性存储器之前可以对操作进行尝试。

重要: 这些寄存器的位只能由用户使其从1变为0，用户无法将它们擦除，使它们变回1。

另外，**USER_REG0**、**USER_REG1**和**USER_DBG**使用各自的位31（NW）来指示它们可以供用户写入。这3个寄存器只能写入一次，而Flash保护寄存器可以被写入多次。当**FMC**寄存器的COMT位写入值0xA442.0008时，表 8-2 在 130页提供了每个寄存器确认所需的FMA地址以及要写入的源数据。在写COMT位之后，用户可以查询**FMC** 寄存器来等待确认操作的结束。

表 **8-2. Flash**驻留寄存器[a]

被确认的寄存器	**FMA**值	数据源
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0008	FMPRE3
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1

被确认的寄存器	**FMA**值	数据源
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_DBG	0x7510.0000	FMD

a. 哪个FMPREn和FMPPEn寄存器可用取决于特定的Stellaris®器件的Flash大小。

8.4 寄存器映射

表 8-3 在 131页列出了Flash存储器和控制寄存器。列出的偏移量是寄存器地址的十六进制增量。**FMA**、**FMD**、**FMC**、**FCRIS**、**FCIM**和**FCMISC**寄存器的偏移量是相对 0x400F.D000的Flash控制基址而言的。**FMPREn**、**FMPPEn**、**USECRL**、**USER_DBG**和**USER_REGn** 寄存器的偏移量是相对0x400F.E000的系统控制基址而言的。

表 **8-3. Flash** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
Flash控制偏移量					
0x000	FMA	R/W	0x0000.0000	Flash存储器地址	133
0x004	FMD	R/W	0x0000.0000	Flash存储器数据	134
0x008	FMC	R/W	0x0000.0000	Flash存储器控制	135
0x00C	FCRIS	RO	0x0000.0000	Flash控制器原始中断状态	137
0x010	FCIM	R/W	0x0000.0000	Flash控制器中断屏蔽	138
0x014	FCMISC	R/W1C	0x0000.0000	Flash控制器可屏蔽中断的状态和清除	139
系统控制偏移量					
0x130 and(cn) 0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash存储器保护读使能0	141
0x134 and(cn) 0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash存储器保护编程使能0	142
0x140	USECRL	R/W	0x31	USec重装	140
0x1D0	USER_DBG	RW	0xFFFF.FFFE	用户调试	143
0x1E0	USER_REG0	RW	0xFFFF.FFFF	用户寄存器0	144
0x1E4	USER_REG1	RW	0xFFFF.FFFF	用户寄存器1	145
0x204	FMPRE1	RW	0xFFFF.FFFF	Flash存储器保护读使能1	146
0x208	FMPRE2	RW	0xFFFF.FFFF	Flash存储器保护读使能2	147
0x20C	FMPRE3	RW	0xFFFF.FFFF	Flash存储器保护读使能3	148
0x404	FMPPE1	RW	0xFFFF.FFFF	Flash存储器保护编程使能1	149
0x408	FMPPE2	RW	0xFFFF.FFFF	Flash存储器保护编程使能2	150
0x40C	FMPPE3	RW	0xFFFF.FFFF	Flash存储器保护编程使能3	151

8.5 Flash寄存器描述（Flash控制偏移量）

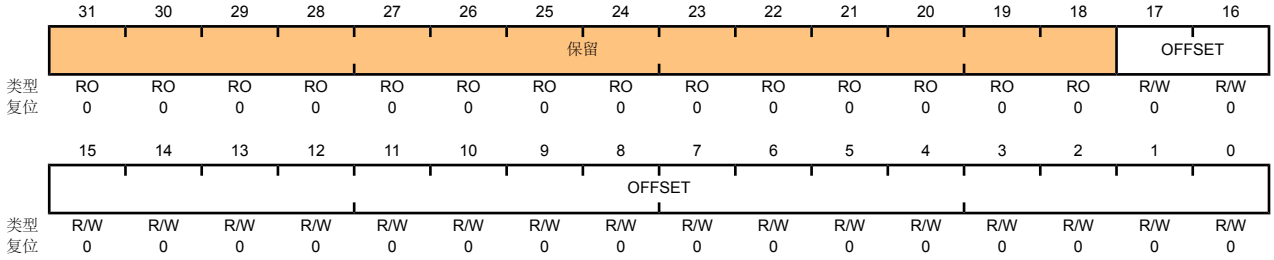
本节的剩余部分按照地址偏移的数字顺序排列和描述Flash存储器寄存器。本节的寄存器与0x400F.D000 的Flash控制基址相关。

寄存器 1: Flash存储器地址（FMA），偏移量 0x000

在写操作过程中，该寄存器含有一个4字节对齐的地址并指定在哪里写入数据。在擦除操作过程中，该寄存器含有一个1KB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)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



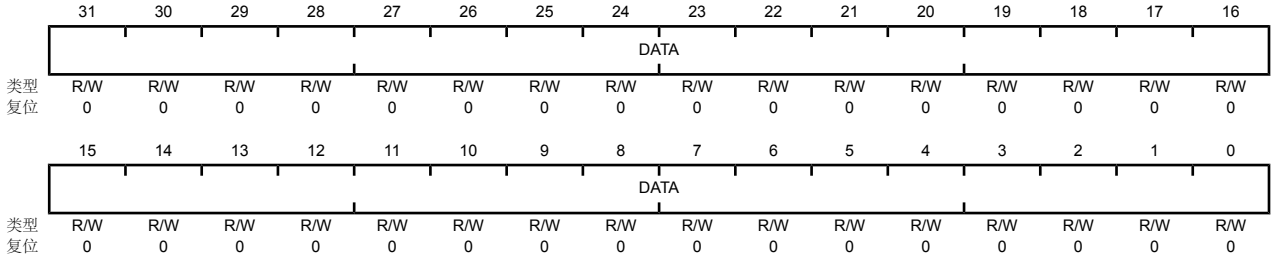
位/域	名称	类型	复位	描述
31:18	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
17:0	OFFSET	R/W	0x0	地址偏移量 执行操作的flash地址偏移量, 非易失性寄存器除外 (有关该域的值的详细内容见 “非易失性寄存器编程” 在 130页)。

寄存器 2: Flash存储器数据（FMD），偏移量 0x004

该寄存器包含的是编程周期中被写入的数据或读周期中被读取的数据。需要注意的是，并未定义只执行模块读取访问时该寄存器的内容。在擦除过程中不使用该寄存器。

Flash存储器数据 (FMD)

偏移量 0x004
类型 R/W, 复位 0x0000.0000



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

寄存器 3: Flash存储器控制（FMC），偏移量 0x008

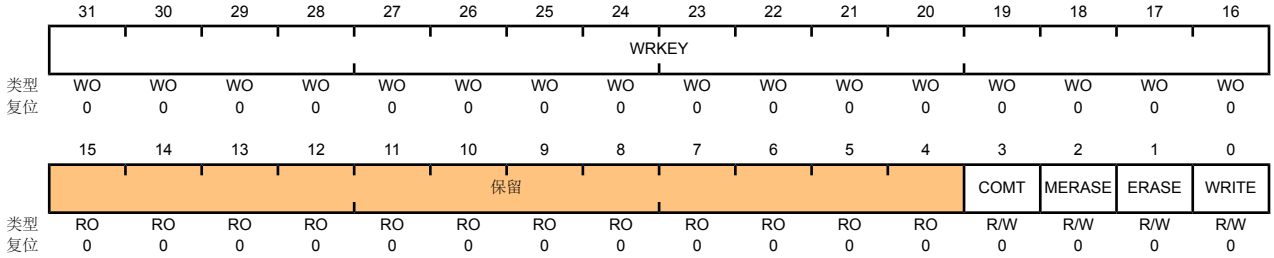
当该寄存器被写入时，Flash控制器将对**Flash**存储器地址（**FMA**）指定的位置启动适当周期数目的访问操作（见133页）。如果是写访问，**Flash**存储器数据（**FMD**）寄存器包含的数据（见134页）被写入。

这是最后被写入的寄存器，将启动存储器的操作。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。这些寄存器中最常使用的位是ERASE和WRITE。

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
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	WRKEY	WO	0x0	Flash写密钥 该域包含一个写密钥，用来最大限度地减少意外写Flash的事件。必须向这个域写入值0xA442来触发写操作。如果写入**FMC**寄存器的值不包含WRKEY的值，那么该值将被忽略。读取该字段将返回0。
15:4	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	COMT	R/W	0	确认寄存器值 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	整体擦除Flash存储器 如果该位被置位，器件的Flash主存储器的内部全部被擦除。写0对该位的状态无影响。 如果该位被读取，则提供先前整体擦除访问的状态。如果前面的整体擦除访问结束，则返回0；否则，如果前面的整体擦除访问没有结束，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 如果该位被置位，**FMA**内容指定的Flash主存储器的页被擦除。写0对该位的状态无影响。 如果该位被读取，则提供先前的擦除访问的状态。如果前面的擦除访问结束，则返回0；否则，如果前面的擦除访问没有结束，返回1。 该操作所需的时间最多可达25ms。

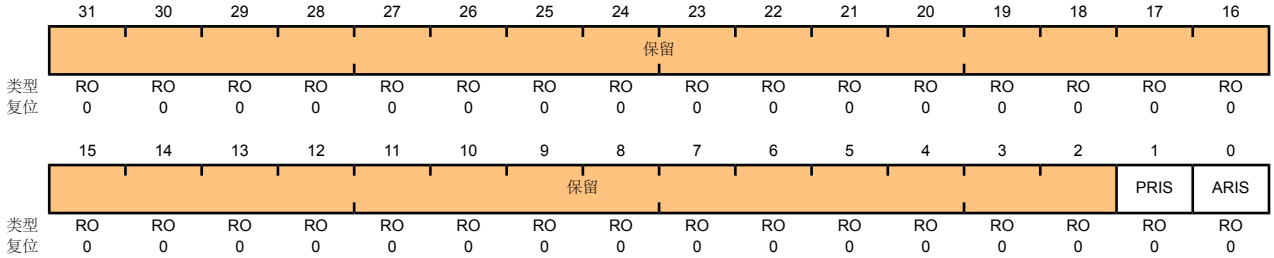
位/域	名称	类型	复位	描述
0	WRITE	R/W	0	写一个字到Flash存储器 如果该位被置位，**FMD**存放的数据写入到**FMA**的内容所指定的位置。写0对该位的状态无影响。 如果该位被读取，则提供前面写更新的状态。如果前面的写访问结束，则返回0；否则，如果写访问没有结束，返回1。 该操作所需的时间最多可达50 µs。

寄存器 4: Flash控制器原始中断状态（FCRIS），偏移量 0x00C

这个寄存器指示Flash控制有一个中断条件。而一个中断信号只有在其相应的**FCIM**寄存器位被置位时才发出。

Flash控制器原始中断状态 (FCRIS)

偏移量 0x00C
类型 RO, 复位 0x0000.0000



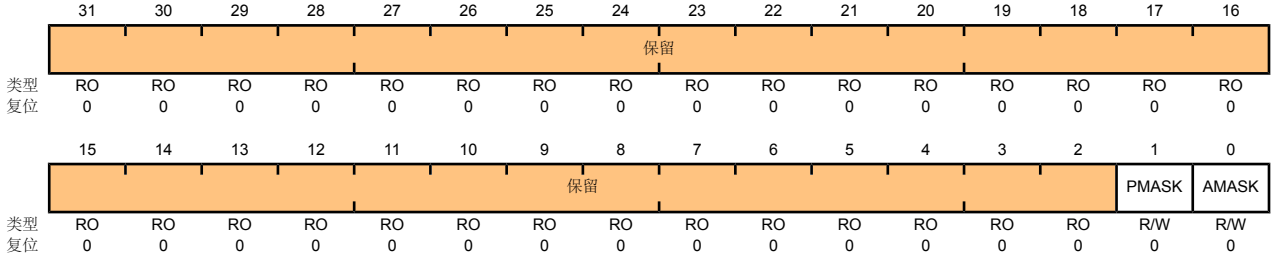
位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PRIS	RO	0	编程的原始中断状态 该位表示编程周期的当前状态。如果置位，则编程周期完成；如果清零，则编程周期还没有完成。编程周期是指通过**Flash**存储器控制（**FMC**）寄存器位产生的写或擦除操作（见 135页）。
0	ARIS	RO	0	访问的原始中断状态 该位指示Flash是否被不正确地访问。如果置位，则表示程序试图访问Flash的操作与**Flash** 存储器保护读使能（**FMPREn**）和 **Flash**存储器保护编程使能（**FMPPEn**）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 5: 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
类型 R/W, 复位 0x0000.0000



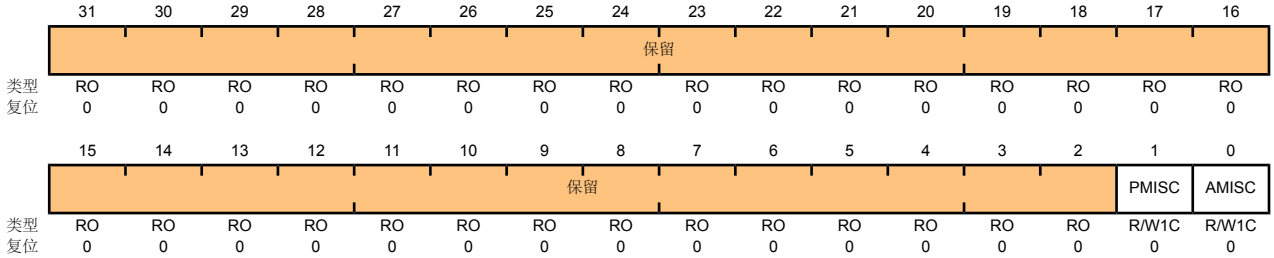
位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PMASK	R/W	0	编程中断屏蔽 该位控制着编程原始中断状态向控制器的报告。如果置位，则向控制器提交编程所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。
0	AMASK	R/W	0	访问中断屏蔽。 该位控制着访问原始中断向控制器的报告。如果置位，则向控制器提交访问所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。

寄存器 6: Flash控制器可屏蔽中断的状态和清除（FCMISC），偏移量 0x014

该寄存器提供了两个功能。首先，它可以通过指示哪个中断源或哪些中断源正在发出中断信号来报告中断产生的原因。其次，它还可以作为清除中断报告的方法。

Flash控制器可屏蔽中断的状态和清除 (FCMISC)

偏移量 0x014
类型 R/W1C, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PMISC	R/W1C	0	编程可屏蔽中断的状态和清除 该位指示中断是否由于编程周期结束而产生以及中断是否不屏蔽。该位通过写入1来清零。当PMISC位被清零时，**FCRIS**寄存器的 PRIS 位（见 137页）也被清零。
0	AMISC	R/W1C	0	访问可屏蔽中断的状态和清除 该位指示中断是否由于试图执行错误的访问而产生以及中断是否不屏蔽。该位通过写入1来清零。当AMISC位清零时，**FCRIS**寄存器的 ARIS 位也被清零。

8.6 Flash寄存器描述（系统控制偏移量）

本节的剩余部分按照地址偏移的数字顺序排列和描述Flash存储器寄存器。本节中的寄存器与0x400F.E000的系统控制基址相关。

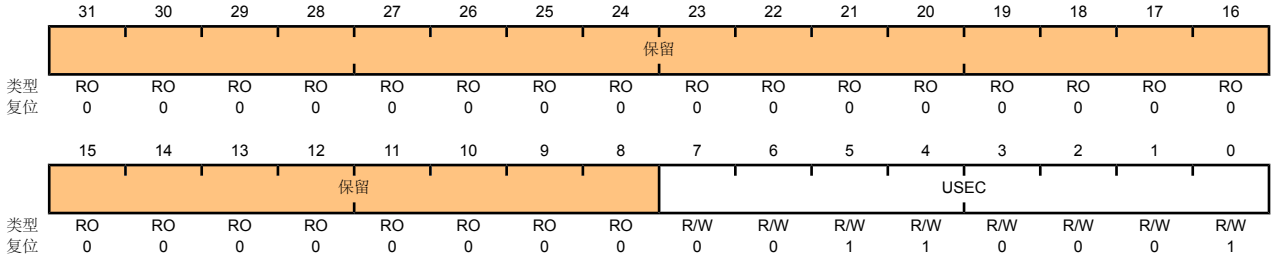
寄存器 7: USec重装（USECRL），偏移量 0x140

注意: 偏移量是相对于基址而言的0x400F.E000

这个寄存器作为一个方法提供给Flash控制器，用来创建一个1µs时钟节拍分频器的重装值。内部Flash对可以被应用的高电压写脉冲的时间长度有特定的最大值和最小值要求。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，那么也要求用户改变该值。

USec重装 (USECRL)

偏移量 0x140
类型 R/W, 复位 0x31



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	USEC	R/W	0x31	微秒重装值 当Flash正在被擦除或编程时，控制器时钟的频率为MHz-1。 当Flash正在被擦除或编程时，USEC应该被设置成 0x31 (50 MHz) 。

寄存器 8: Flash存储器保护读使能0 （FMPRE0）， 偏移量 0x130 and(cn) 0x200

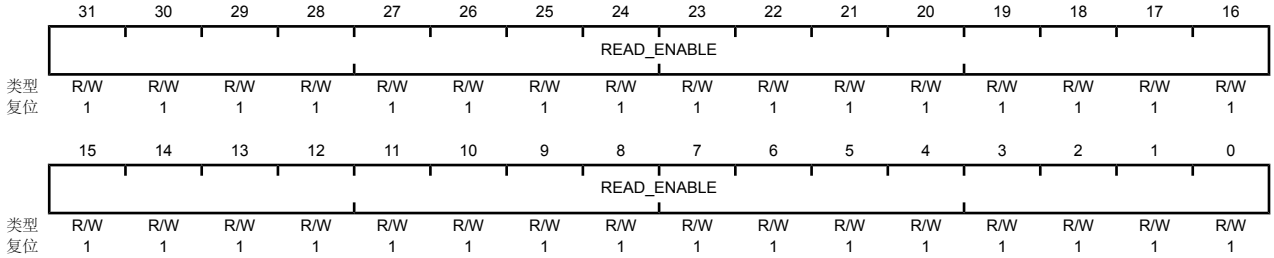
注意: 这个寄存器采用别名，用来向后兼容。

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

这个寄存器存放的是每个2-KB Flash区块的只读保护位 (**FMPPEn** 保存只执行位)。该寄存器在上电复位期间加载。 对所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是，这个寄存器属于R/W0；用户只可以将保护位从1变为0（不能从0变为1）。 这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见“Flash存储器保护”小节。

Flash存储器保护读使能0 (FMPRE0)

偏移量 0x130 and(cn) 0x200
类型 R/W, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash读使能

允许对2-KB flash区块执行命令或进行读操作。可以结合如表 “Flash保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

寄存器 9: Flash存储器保护编程使能0 （FMPPE0）， 偏移量 0x134 and(cn) 0x400

注意: 这个寄存器采用别名，用来向后兼容。

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是2-KB Flash区块的只执行保护位(**FMPREn** 保存只执行位)。该寄存器在上电复位期间加载。 对于所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是，这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。 这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash存储器保护"小节。

Flash存储器保护编程使能0 (FMPPE0)

偏移量 0x134 and(cn) 0x400
类型 R/W, 复位 0xFFFF.FFFF

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位/域	名称	类型	复位	描述
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash编程使能 将2KB flash区块配置为只执行。结合 “Flash保护策略组合”所示的策略。 值 描述值 0xFFFFFFFF 使能256KB flash。

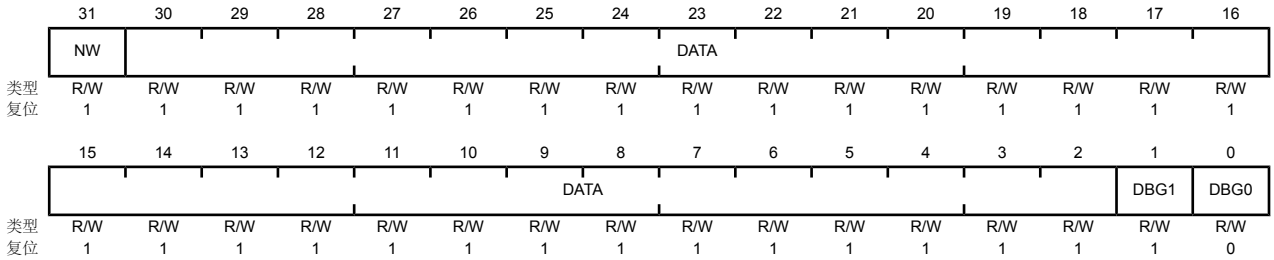
寄存器 10: 用户调试 （USER_DBG）， 偏移量 0x1D0

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器通过提供一种“写一次”机制来禁止外部调试器访问器件和27位用户定义的数据。DBG0 位(位0)在出厂时被设置为0，DBG1 位(位1)被设为1，这些设置将使能外部调试器。 通过改变DBG1 位为0，从器件的下一个上电周期开始可以永久地禁止外部调试器访问器件。NOTWRITTEN 位(位31)表示寄存器可以被写，并且通过硬件控制来确保该寄存器只能写一次。

用户调试 (USER_DBG)

偏移量 0x1D0
类型 RW, 复位 0xFFFF.FFFE



位/域	名称	类型	复位	描述
31	NW	R/W	1	用户调试没有被写 表示这个32位双字没有被写。
30:2	DATA	R/W	0x1FFFFFFF	用户数据 包含用户数据值。该位域初始化后为全1，并且只能写一次。
1	DBG1	R/W	1	调试控制1 该DBG1 位必须为1且DBG0 必须为0使得可以实现调试。
0	DBG0	R/W	0	调试控制0 该DBG1 位必须为1且DBG0 必须为0使得可以实现调试。

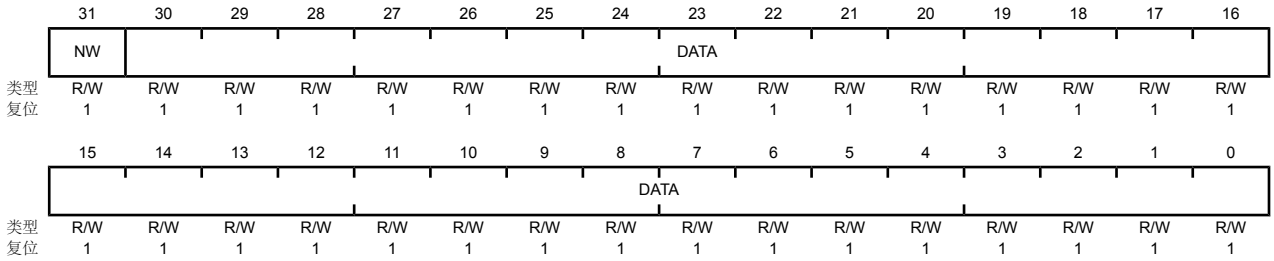
寄存器 11: 用户寄存器0 （USER_REG0）， 偏移量 0x1E0

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器包含31位用户定义的数据，这些数据是非易失的，且只能写一次。位31表示该寄存器可以被写并且通过硬件控制来确保该寄存器只能写一次。寄存器的这种“写一次”特性在保存像通信地址这类各器件所独有的静态信息时非常有用，否则便需要外部EEPROM或其它非易失性器件。

用户寄存器0 (USER_REG0)

偏移量 0x1E0
类型 RW, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31	NW	R/W	1	不被写 表示该32位的双字还没有被写。
30:0	DATA	R/W	0x7FFFFFFF	用户数据 包含用户数据值。这个位域初始化后为全1，并且只能写一次。

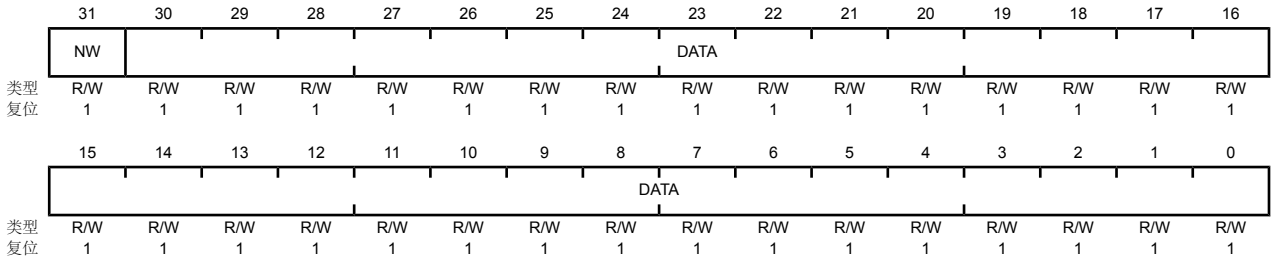
寄存器 12: 用户寄存器1 （USER_REG1）， 偏移量 0x1E4

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器包含31位用户定义的数据，这些数据是非易失的，且只能写一次。位31表示寄存器可以被写，并且通过硬件控制来确保该寄存器只能写一次。该寄存器的这种“写一次”特性在保存像通信地址这类各器件所独有的静态信息时非常有用，否则就需要外部EEPROM或其它非易失性器件了。

用户寄存器1 (USER_REG1)

偏移量 0x1E4
类型 RW, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31	NW	R/W	1	没有被写。 表示这个32位双字还没有被写。
30:0	DATA	R/W	0x7FFFFFFF	用户数据 包含用户数据值。这个位域初始化后为全1，并且只能写一次。

寄存器 13: Flash存储器保护读使能1 （FMPRE1）， 偏移量 0x204

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只读保护位 (**FMPPEn** 保存只执行位)。该寄存器在上电复位期间加载。 对于所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设置为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash存储器保护" 小节。

Flash存储器保护读使能1 (FMPRE1)

偏移量 0x204
类型 RW, 复位 0xFFFF.FFFF

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	READ_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash读使能

允许对2-KB flash区块执行命令或读操作。可以结合如表“Flash 保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

寄存器 14: Flash存储器保护读使能2 （FMPRE2）， 偏移量 0x208

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只读保护位 (**FMPPEn** 保存只执行位)。该寄存器在上电复位期间加载。 对所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设置为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash 存储器保护" 小节。

Flash存储器保护读使能2 (FMPRE2)

偏移量 0x208
类型 RW, 复位 0xFFFF.FFFF

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	READ_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash读使能

允许对2-KB flash区块执行命令或读操作。可以结合如表 “Flash 保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

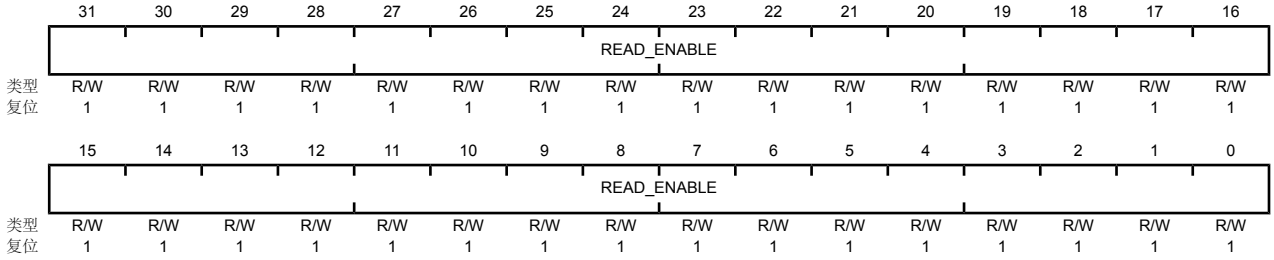
寄存器 15: Flash存储器保护读使能3 （FMPRE3）， 偏移量 0x20C

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只读保护位(**FMPPEn** 保存只执行位)。该寄存器在上电复位序列期间加载。 对于所有只执行的存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。 这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash存储器保护"小节。

Flash存储器保护读使能3 (FMPRE3)

偏移量 0x20C
类型 RW, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash读使能

允许对2-KB Flash区块执行命令或读操作。可以结合如表 “Flash保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

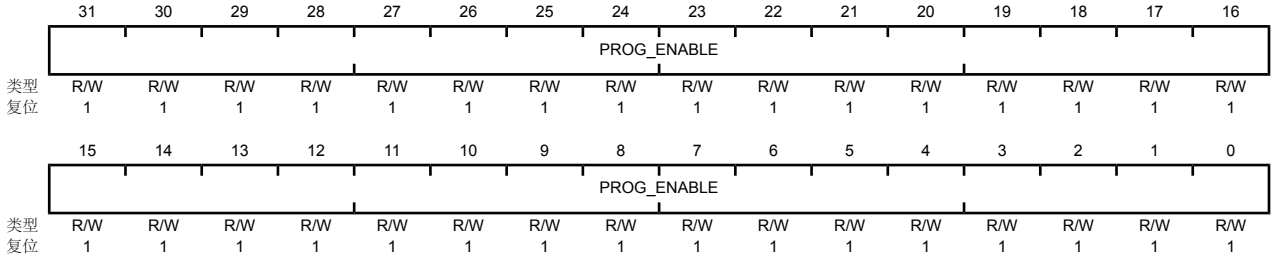
寄存器 16: Flash存储器保护编程使能1 （FMPPE1）， 偏移量 0x404

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只执行保护位 (**FMPREn** 保存只执行位)。该寄存器在上电复位期间加载。 对所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设置为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash存储器保护"小节。

Flash存储器保护编程使能1 (FMPPE1)

偏移量 0x404
类型 RW, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash编程使能

将2-KB Flash区块配置为只执行。可以结合如表“Flash保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

寄存器 17: Flash存储器保护编程使能2 （FMPPE2）， 偏移量 0x408

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只执行保护位(**FMPREn**保存只执行位)。该寄存器在上电复位期间加载。 对于所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存在出厂时都被设置为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见"Flash存储器保护"小节。

Flash存储器保护编程使能2 (FMPPE2)

偏移量 0x408
类型 RW, 复位 0xFFFF.FFFF

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
类型	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
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位/域	名称	类型	复位	描述
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash编程使能

将2-KB Flash区块配置为只执行。可以结合如表“Flash 保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

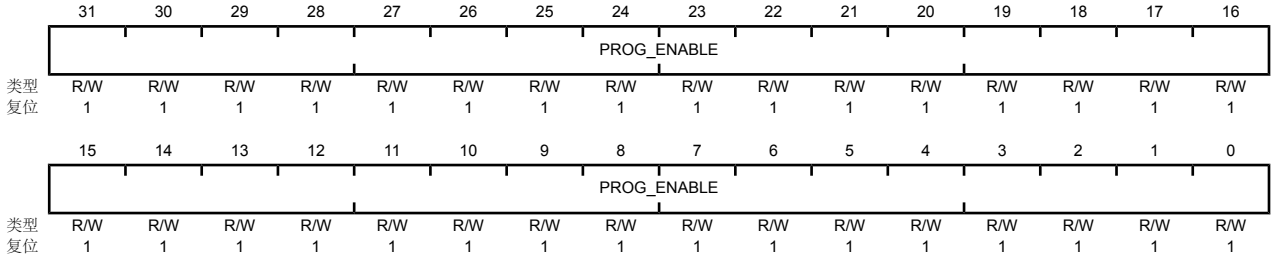
寄存器 18: Flash存储器保护编程使能3 （FMPPE3）， 偏移量 0x40C

注意: 偏移量是相对 0x400FE000的系统控制基址而言的。

该寄存器存放的是每个2-KB Flash区块的只执行保护位(**FMPREn** 保存只执行位)。该寄存器在上电复位期间加载。 对于所有执行存储体来说，**FMPREn** 和 **FMPPEn** 寄存器在出厂时都被设置为1。这样就可以得到一种开放式访问和编程的策略。我们可以通过写特定的寄存器位来改变该寄存器的位。但是这个寄存器属于R/W0；用户只能将保护位从1变为0（不能从0变为1）。 这种改变不是永久的，直至寄存器被提交（保存），此时位的改变是永久性的。如果某个位从1变为0且没有提交，其内容可以通过执行上电复位序列进行恢复。 详见 "Flash存储器保护" 小节。

Flash存储器保护编程使能3 (FMPPE3)

偏移量 0x40C
类型 RW, 复位 0xFFFF.FFFF



位/域	名称	类型	复位	描述
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash编程使能

将2KB Flash区块配置为只执行。可以结合如表 “Flash保护策略组合”所示的策略。

值	描述值
0xFFFFFFFF	使能256KB flash。

9 通用输入/输出端口（GPIO）

GPIO模块由 8个物理GPIO 模块组成，每个对应一个独立的GPIO端口（端口A, 端口B, 端口C, 端口D, 端口E, 端口F, 端口G, 和端口H）。GPIO模块遵循FiRM规范，并且支持10-60 个可编程的输入/输出管脚，具体取决于正在使用的外设。

GPIO模块具有以下的特性：

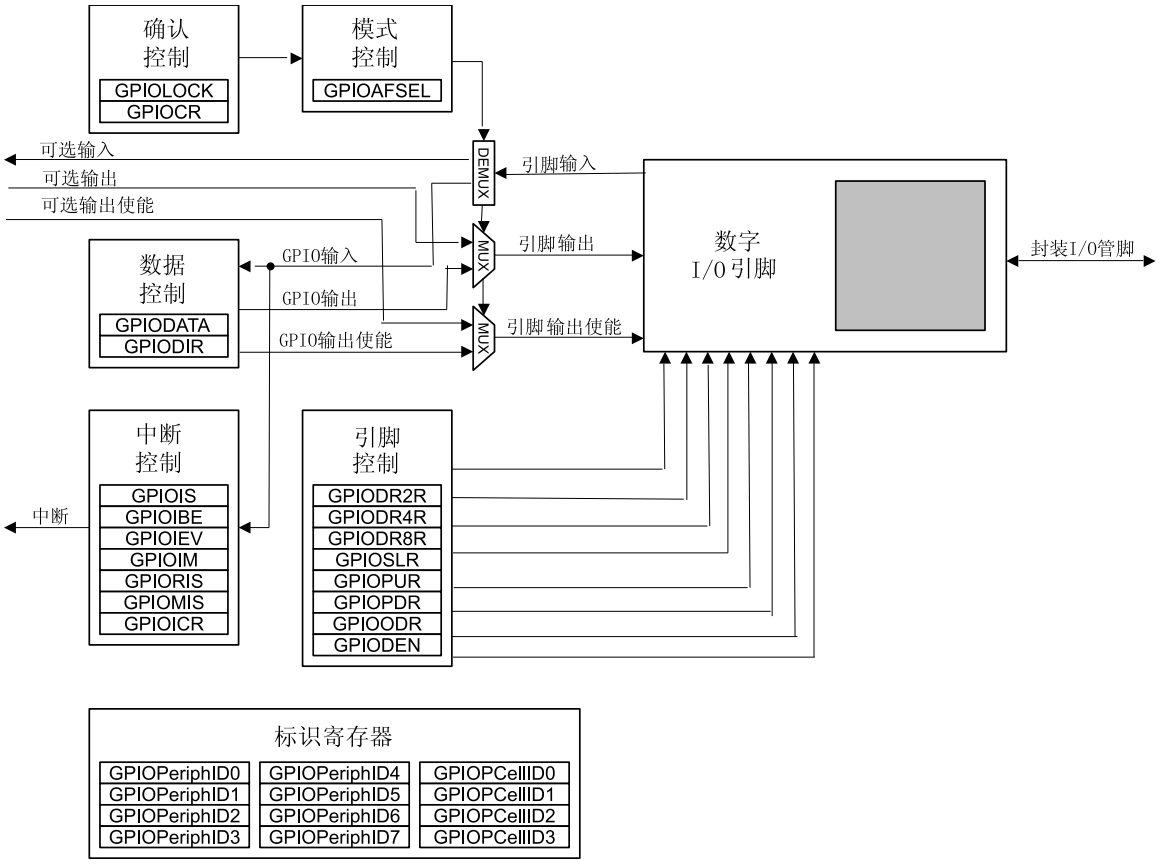
- 可编程控制GPIO中断
  - 屏蔽中断发生
  - 边沿触发（上升沿，下降沿，上升、下降沿）
  - (高或低)电平触发
- 输入/输出可承受5V电压
- 在读和写操作中通过地址线进行位屏蔽
- 可编程控制GPIO引脚（pad）配置
  - 弱上拉或下拉电阻
  - 2-mA, 4-mA 和 8-mA 引脚驱动
  - 8-mA驱动的斜率控制
  - 开漏使能
  - 数字输入使能

9.1 功能描述

重要: 除了5个JTAG/SWD管脚（PB7 和PC[3:0]）之外，所有GPIO管脚默认都是三态管脚（**GPIOAFSEL=0, GPIODEN=0, GPIOPDR=0,** 且 **GPIOPUR**=0）。JTAG/SWD 管脚默认为 JTAG/SWD 功能（**GPIOAFSEL**=1, **GPIODEN=1** 且 **GPIOPUR**=1）。通过上电复位（POR）或外部复位（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(见图 9-1 在 153页)中的独立硬件的实例化（hardware instantiation）。LM3S2950 微控制器含有 8个 端口以及 8个 的这些物理GPIO模块。

图 **9-1. GPIO**端口方框图



9.1.1 数据控制

数据控制寄存器允许软件配置GPIO的操作模式。当数据寄存器捕获输入的数据或驱动数据从引脚输出时，数据方向寄存器将GPIO配置为输入或输出。

9.1.1.1 数据方向操作

GPIO 方向 **(GPIODIR)** 寄存器（见 160页）用来将每个独立的管脚配置为输入或输出。当数据方向位设为0时，GPIO配置为输入，并且对应的数据寄存器位将捕获和存储GPIO端口上的值。当数据方向位设为1时，GPIO配置为输出，并且对应的数据寄存器位将在GPIO端口上输出。

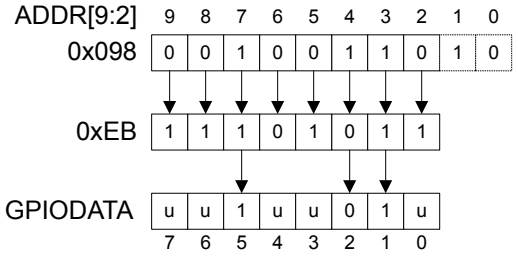
9.1.1.2 数据寄存器操作

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的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，那么该寄存器的值将保持不变。

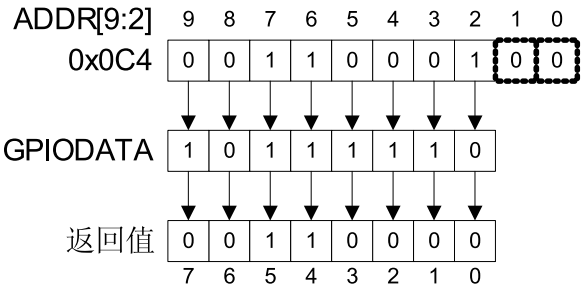
例如，将0xEB的值写入地址GPIODATA+0x098处，结果如图 9-2 在 154页所示，其中，u表示没有被写操作改变的数据。

图 **9-2. 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 9-3 在 154页所示。

图 **9-3. GPIODATA** 读实例



9.1.2 中断控制

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（见 161页）
- **GPIO** 中断双边沿 **(GPIOIBE)** 寄存器（见 162页）
- **GPIO** 中断事件 **(GPIOIEV)**寄存器（见 163页）

通过**GPIO** 中断屏蔽 **(GPIOIM)**寄存器可以使能或禁能中断（见 164页）。

当产生中断条件时，可以在 **GPIO** 原始中断状态 **(GPIORIS)** 和 **GPIO** 屏蔽后的中断状态 **(GPIOMIS)** 寄存器中观察到中断信号的状态（见 165页和 166页）。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。**GPIORIS**寄存器则表示GPIO管脚满足中断条件，但是不一定发送到控制器。

写1到**GPIO** 中断清零 **(GPIOICR)** 寄存器可以清除中断（见 167页）。

在对下列中断控制寄存器进行编程时，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。

9.1.3 模式控制

GPIO 管脚可以由硬件或软件控制。当通过**GPIO** 备用功能选择 **(GPIOAFSEL)** 寄存器（见 168页）将硬件控制使能时，管脚状态将由它备用的功能（即外设）控制。软件控制相当于GPIO模式，在该模式下，使用**GPIODATA**寄存器来读/写相应的管脚。

9.1.4 确认（**commit**）控制

确认（commit）控制寄存器提供了保护层以防止对重要硬件外设的意外编程。写 **GPIO** 可选功能选择 **(GPIOAFSEL)** 寄存器（见 168页）的保护位不确认存储，除非 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 177页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 178页）的相应位已被设为1。

9.1.5 引脚（**Pad**）控制

引脚控制寄存器使软件能够根据应用的要求来配置GPIO引脚。引脚控制寄存器包括 **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR** 和 **GPIODEN**寄存器。

9.1.6 标识

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** 寄存器以及 **GPIOPCellID0**-**GPIOPCellID3** 寄存器。

9.2 初始化和配置

为了使用GPIO，必须通过置位**RCGC2**寄存器中相应的GPIO端口位字段（`GPIOn`）将外设时钟使能。

复位时，所有GPIO 管脚（除了5个 JTAG 管脚外）都配置为非驱动（三态）：**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0 且 **GPIOPUR**=0。表 9-1 在 155页 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 9-2 在 156页 给出了为GPIO端口的管脚2配置上升沿中断的方法。

表 **9-1. GPIO** 端口配置实例

配置	**GPIO** 寄存器位的值[a]									
	AFSEL	**DIR**	**ODR**	**DEN**	**PUR**	**PDR**	**DR2R**	**DR4R**	**DR8R**	**SLR**
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	?	?	?	?
开漏输入/输出 (I^2C)	1	X	1	1	X	X	?	?	?	?
数字输入 (定时器 CCP)	1	X	0	1	?	?	X	X	X	X
数字输入 (QEI)	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 **9-2. GPIO** 中断配置实例

寄存器	期望的中断事件触发	管脚**2**位值[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）

9.3 寄存器映射

表 9-3 在 157页列出GPIO 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GPIO端口对应的基址如下：

- GPIO 端口A:0x4000.4000
- GPIO 端口B:0x4000.5000
- GPIO 端口C:0x4000.6000
- GPIO 端口D:0x4000.7000
- GPIO 端口E:0x4002.4000
- GPIO 端口F:0x4002.5000
- GPIO 端口G:0x4002.6000
- GPIO 端口H:0x4002.7000

重要: 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。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。

注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**，**GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD功能。正因为这样，对于GPIO端口B，该寄存器默认的复位值是0x0000.0080，而对于端口C，其默认的复位值是0x0000.000F。

对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOCR**寄存器默认的寄存器类型都是RO。这5个管脚是当前仅受**GPIOCR**寄存器保护的GPIO。正因为这样，对于GPIO 端口B7 和 GPIO 端口C[3:0]的寄存器类型是R/W。

对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOCR**寄存器默认的复位值都是0x0000.00FF。为了确保JTAG端口不被意外地编程为GPIO，这5个管脚默

认为不可确认（non-commitable）。正因为这样，对于GPIO端口B，**GPIOCR** 默认的复位值是0x0000.007F，而对于端口C，**GPIOCR**默认的复位值是0x0000.00F0。

表 **9-3. GPIO** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	GPIODATA	R/W	0x0000.0000	GPIO 数据	159
0x400	GPIODIR	R/W	0x0000.0000	GPIO 方向	160
0x404	GPIOIS	R/W	0x0000.0000	GPIO 中断检测	161
0x408	GPIOIBE	R/W	0x0000.0000	GPIO 中断双边沿	162
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO 中断事件	163
0x410	GPIOIM	R/W	0x0000.0000	GPIO 中断屏蔽	164
0x414	GPIORIS	RO	0x0000.0000	GPIO 原始中断状态	165
0x418	GPIOMIS	RO	0x0000.0000	GPIO 屏蔽后的中断状态	166
0x41C	GPIOICR	W1C	0x0000.0000	GPIO 中断清除	167
0x420	GPIOAFSEL	R/W	-	GPIO 备用功能选择	168
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA 驱动选择	169
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA 驱动选择	170
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA 驱动选择	171
0x50C	GPIOODR	R/W	0x0000.0000	GPIO 开漏选择	172
0x510	GPIOPUR	R/W	-	GPIO 上拉选择	173
0x514	GPIOPDR	R/W	0x0000.0000	GPIO 下拉选择	174
0x518	GPIOSLR	R/W	0x0000.0000	GPIO 斜率控制选择	175
0x51C	GPIODEN	R/W	-	GPIO 数字使能	176
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO 锁定	177
0x524	GPIOCR	-	-	GPIO 确认	178
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO 外设标识4	179
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO 外设标识 5	180
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO 外设标识 6	181
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO 外设标识7	182
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO 外设标识0	183
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO 外设标识1	184
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO 外设标识2	185
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO 外设标识3	186
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell 标识0	187
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell 标识1	188
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell 标识2	189

偏移量	名称	类型	复位	描述	见页面
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell 标识3	190

9.4 寄存器描述

本小节将按地址偏移量的数字顺序逐个列出GPIO寄存器，并对各个寄存器进行描述。

寄存器 1: GPIO 数据（GPIODATA）， 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（见 160页）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。

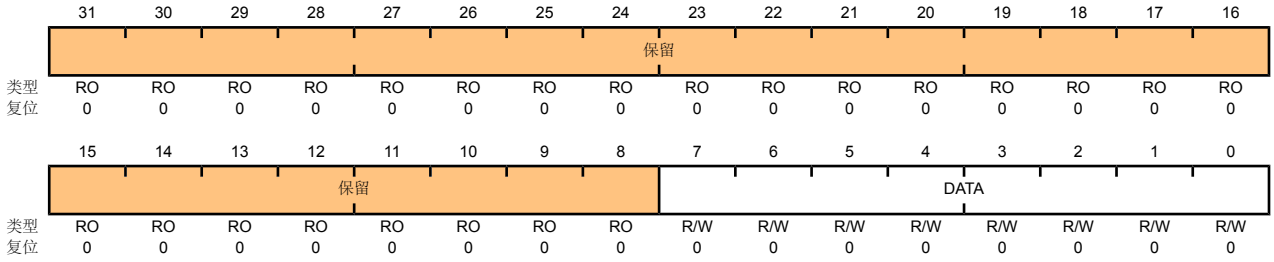
为了对**GPIODATA**寄存器执行写操作，由地址总线位[9:2]产生的相关屏蔽位必须为高电平。否则，该位的值不会被写操作改变。

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，那么读取**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。所有位在复位时都被清零。

GPIO 数据 (GPIODATA)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



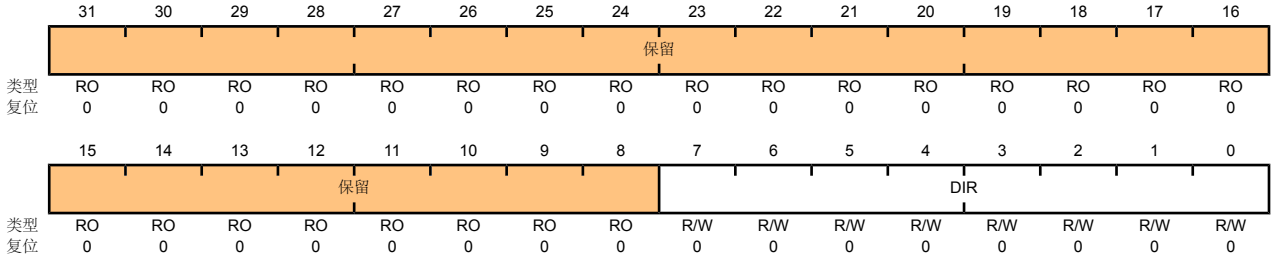
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	R/W	0x00	GPIO 数据 该寄存器被虚拟地映射到地址空间的256个单元中。为便于通过单独的驱动器读写这些寄存器，从这些寄存器读取的值和写入这些寄存器的值都被8条地址线`ipaddr[9:2]`屏蔽。读取该寄存器将返回其当前状态。写该寄存器仅影响那些没有被`ipaddr[9:2]`屏蔽的位和被配置成输出的位。关于读写操作的实例，请见“数据寄存器操作” 在 153页。

寄存器 2: GPIO 方向（GPIODIR），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)

偏移量 0x400
类型 R/W, 复位 0x0000.0000



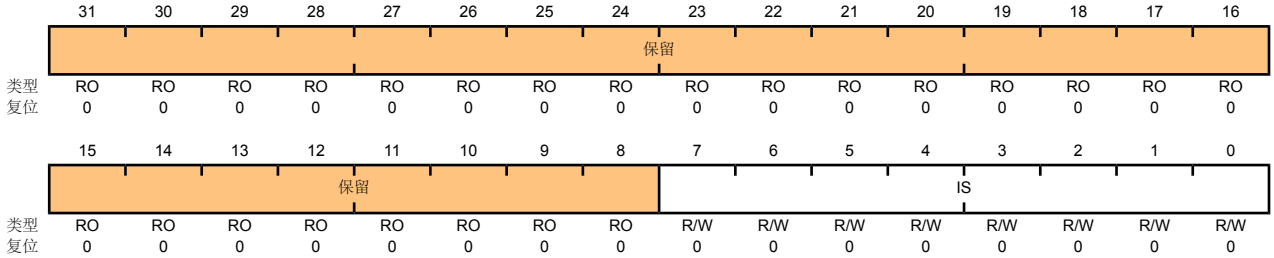
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DIR	R/W	0x00	GPIO 数据方向 `DIR` 值如下定义： 值 描述 0 管脚为输入。 1 管脚为输出。

寄存器 3: GPIO 中断检测 （GPIOIS），偏移量 0x404

GPIOIS 寄存器是中断检测寄存器。**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。所有位在复位时都被清零。

GPIO 中断检测 (GPIOIS)

偏移量 0x404
类型 R/W, 复位 0x0000.0000



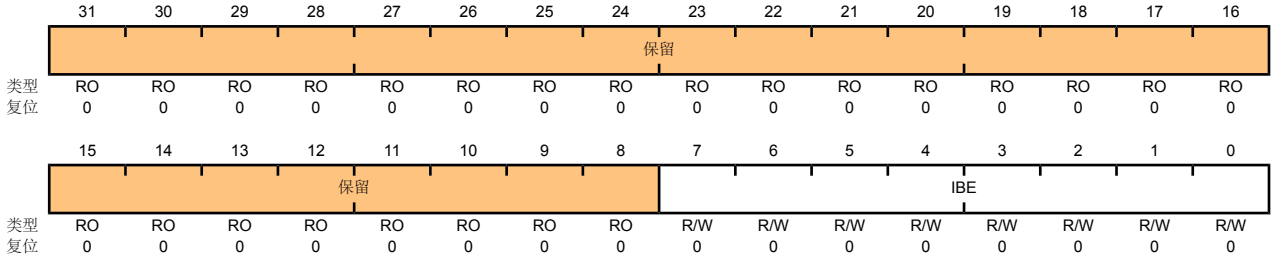
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IS	R/W	0x00	GPIO 中断检测 IS 值如下定义： 值 描述 0 检测到相应管脚上的边沿（边沿检测）。 1 检测到相应管脚上的电平（电平检测）。

寄存器 4: GPIO 中断双边沿（GPIOIBE），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（见 161页）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（见 163页）的相应位。将位清零会将该管脚配置成由**GPIOIEV** 控制。所有位在复位时都被清零。

GPIO 中断双边沿 (GPIOIBE)

偏移量 0x408
类型 R/W, 复位 0x0000.0000



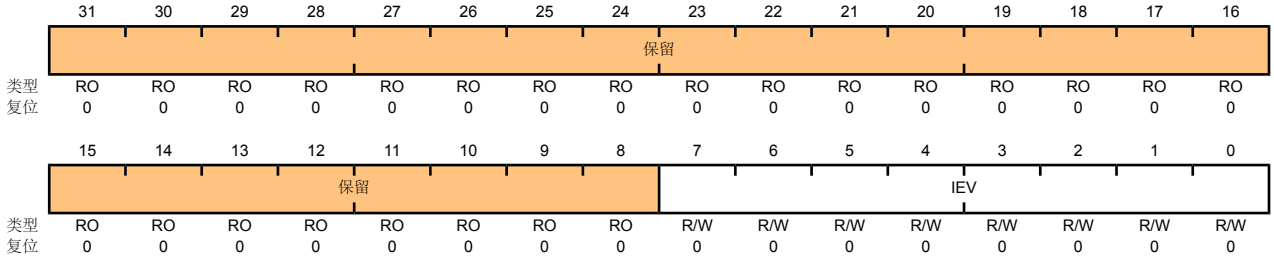
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IBE	R/W	0x00	GPIO 中断双边沿 IBE 值如下定义： 值 描述 0 通过**GPIO** 中断事件 **(GPIOIEV)** 寄存器控制中断发生 (见 163页)。 1 相应管脚上的双边沿触发中断。 注意: 单边沿由 **GPIOIEV** 的相应位来确定。

寄存器 5: GPIO 中断事件（GPIOIEV），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（见 161页）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。所有位在复位时都被清零。

GPIO 中断事件 (GPIOIEV)

偏移量 0x40C
类型 R/W, 复位 0x0000.0000



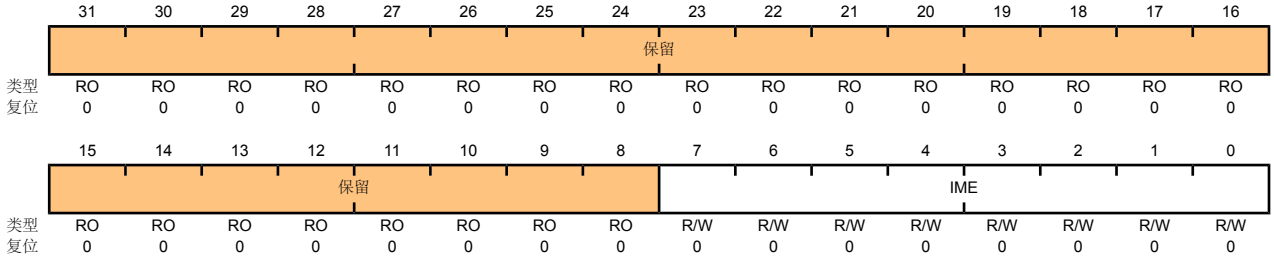
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IEV	R/W	0x00	GPIO 中断事件 IEV 值如下定义： 值 描述 0 相应管脚的下降沿或低电平触发中断。 1 相应管脚的上升沿或高电平触发中断。

寄存器 6: GPIO 中断屏蔽（GPIOIM），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触发它们各自的中断和联合的GPIOINTR线路。将位清零将会禁止该管脚上的中断触发。所有位在复位时都被清零。

GPIO 中断屏蔽 (GPIOIM)

偏移量 0x410
类型 R/W, 复位 0x0000.0000



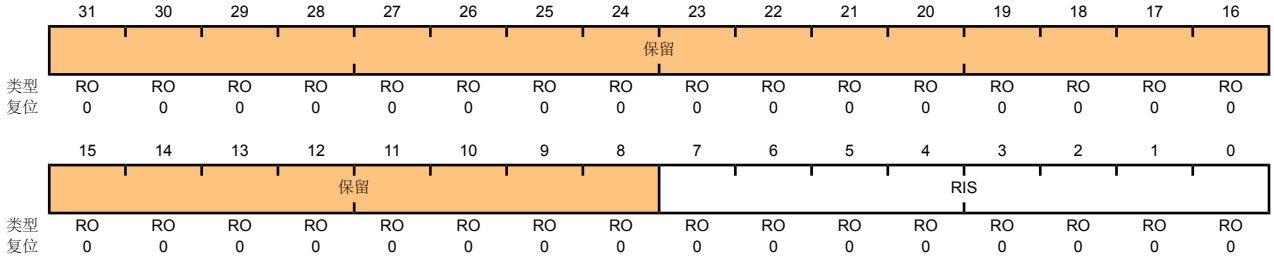
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IME	R/W	0x00	GPIO 中断屏蔽使能 IME 值如下定义： 值 描述 0 相应管脚的中断被屏蔽。 1 相应管脚的中断未被屏蔽。

寄存器 7: GPIO 原始中断状态（GPIORIS），偏移量 0x414

GPIORIS 寄存器是原始中断状态寄存器。**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（见 164页）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。所有位在复位时都被清零。

GPIO 原始中断状态 (GPIORIS)

偏移量 0x414
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	RIS	RO	0x00	GPIO 中断原始状态 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）。 RIS 值如下定义： 值 描述 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。

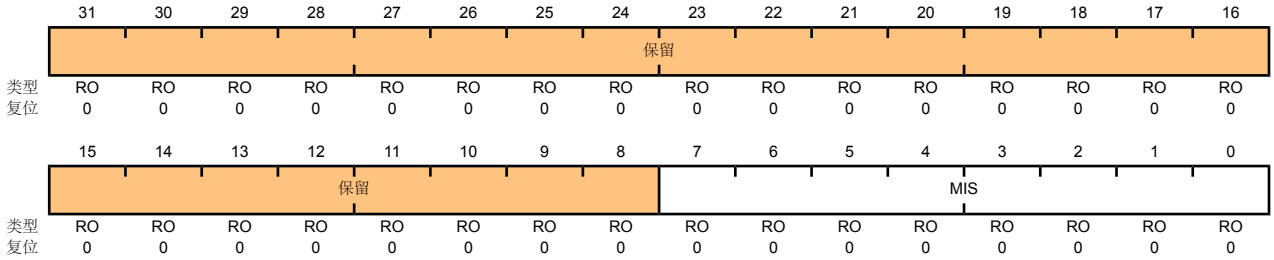
寄存器 8: GPIO 屏蔽后的中断状态（GPIOMIS），偏移量 0x418

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。

GPIOMIS 是屏蔽后中断的状态。

GPIO 屏蔽后的中断状态 (GPIOMIS)

偏移量 0x418
类型 RO, 复位 0x0000.0000



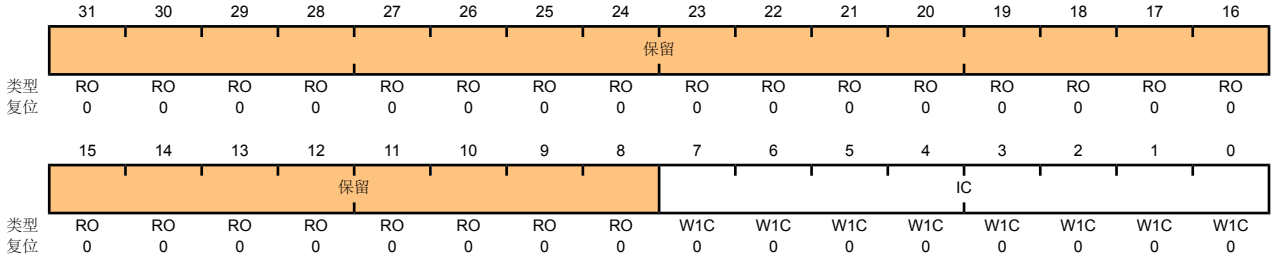
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	MIS	RO	0x00	GPIO 屏蔽后的中断状态 相应管脚上中断已屏蔽的值。 MIS 值如下定义： 值 描述 0 相应的GPIO线的中断未被激活。 1 相应的GPIO线发出中断。

寄存器 9: GPIO 中断清除（GPIOICR），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)

偏移量 0x41C
类型 W1C, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IC	W1C	0x00	GPIO 中断清除 IC 值如下定义： 值 描述 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），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）。

确认（commit）控制寄存器提供了保护层以防止对重要硬件外设的意外编程。写 **GPIO** 可选功能选择 **(GPIOAFSEL)** 寄存器（见 168页）的保护位不确认存储，除非 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 177页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 178页）的相应位已被设为1。

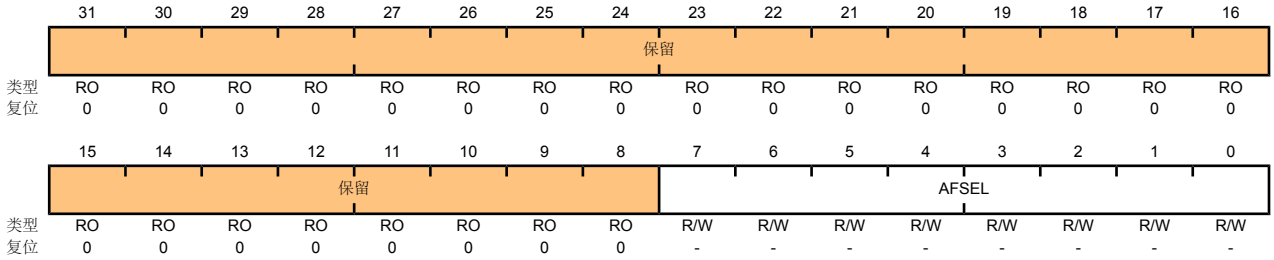
重要: 除了5个JTAG/SWD管脚（PB7 和PC[3:0]）之外，所有GPIO管脚默认都是三态管脚（**GPIOAFSEL=0, GPIODEN=0, GPIOPDR=0,** 且 **GPIOPUR**=0）。JTAG/SWD 管脚默认为 JTAG/SWD 功能（**GPIOAFSEL**=1, **GPIODEN=1** 且 **GPIOPUR**=1）。通过上电复位（POR）或外部复位（RST），可以让这两组管脚都回到其默认状态。

小心 **–** 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**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**Stellaris®**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中止控制器。这会将调试器锁在元件外。通过使用一个基于外部或软件的触发器来恢复**JTAG**功能的软件程序就可以避免这种情况发生。

GPIO 备用功能选择 (GPIOAFSEL)

偏移量 0x420
类型 R/W, 复位 -



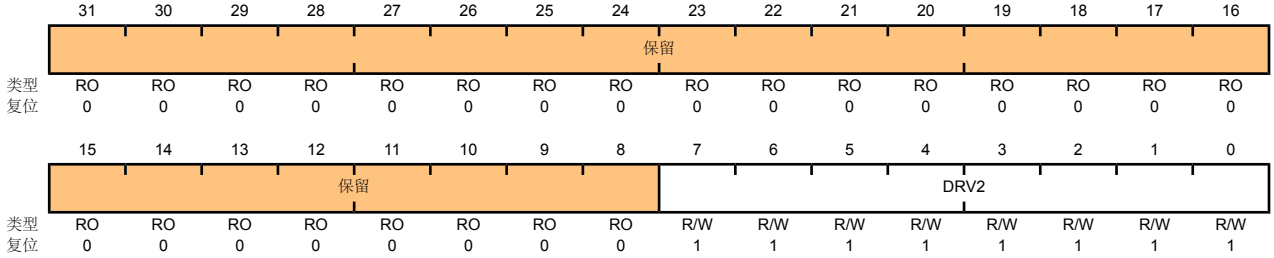
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	AFSEL	R/W	-	GPIO 备用功能选择 AFSEL 值如下定义： 值 描述 0 软件控制相应的GPIO线（GPIO模式）。 1 硬件控制相应的GPIO线（备用的硬件功能）。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**, **GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD 功能。正因为这样，对于GPIO端口B，该寄存器默认的复位值是0x0000.0080，而对于端口C，其默认的复位值是0x0000.000F。

寄存器 11: GPIO 2-mA 驱动选择（GPIODR2R），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)

偏移量 0x500
类型 R/W, 复位 0x0000.00FF



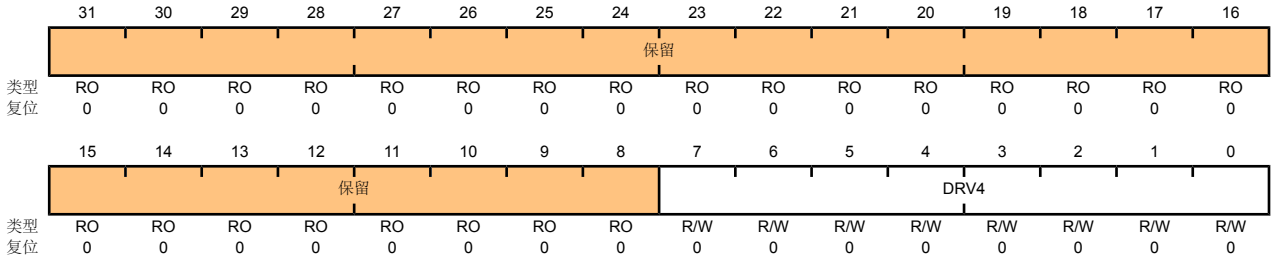
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV2	R/W	0xFF	输出端口 2-mA 驱动使能 向**GPIODR4[n]** 或 **GPIODR8[n]**写入1，都会使相应的2-mA使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

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

The **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)

偏移量 0x504
类型 R/W, 复位 0x0000.0000



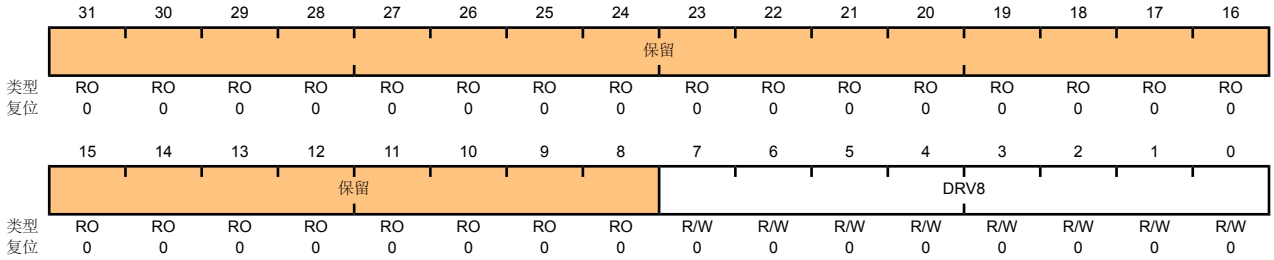
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV4	R/W	0x00	输出端口 4-mA 驱动使能 向**GPIODR2[n]** 或 **GPIODR8[n]**写入1，都会使相应的4-mA使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

寄存器 13: GPIO 8-mA 驱动选择（GPIODR8R），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)

偏移量 0x508
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV8	R/W	0x00	输出端口 8-mA 驱动使能 向**GPIODR2[n]** 或 **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。

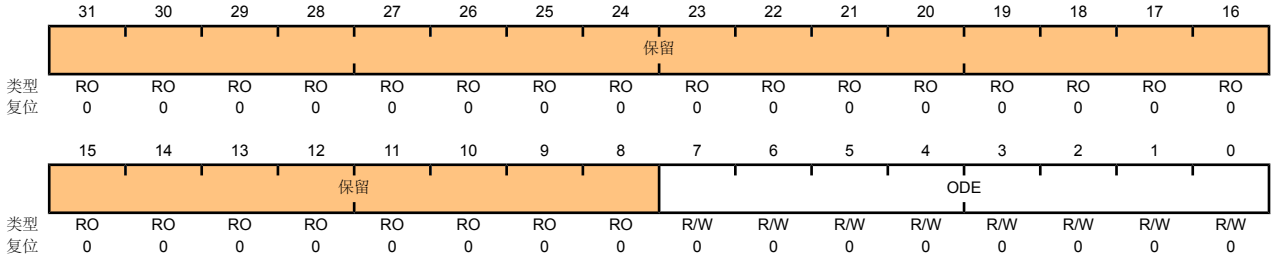
寄存器 14: GPIO 开漏选择（GPIOODR），偏移量 0x50C

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 **(GPIODEN)**寄存器（见 176页）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。

当使用 I^2C 模块时，PB2和PB3的**GPIO** 备用功能选择 **(GPIOAFSEL)** 寄存器位应设为1（见“初始化和配置” 在 155页中的实例）。

GPIO 开漏选择 (GPIOODR)

偏移量 0x50C
类型 R/W, 复位 0x0000.0000



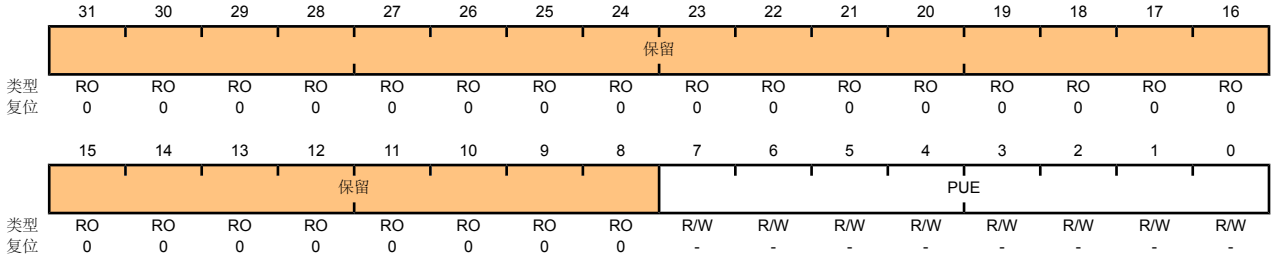
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	ODE	R/W	0x00	输出端口开漏使能 ODE 值如下定义： 值 描述 0 开漏配置被禁能。 1 开漏配置被使能。

寄存器 15: GPIO 上拉选择（GPIOPUR），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（见 174页）中相应的位自动清零。

GPIO 上拉选择 (GPIOPUR)

偏移量 0x510
类型 R/W, 复位 -



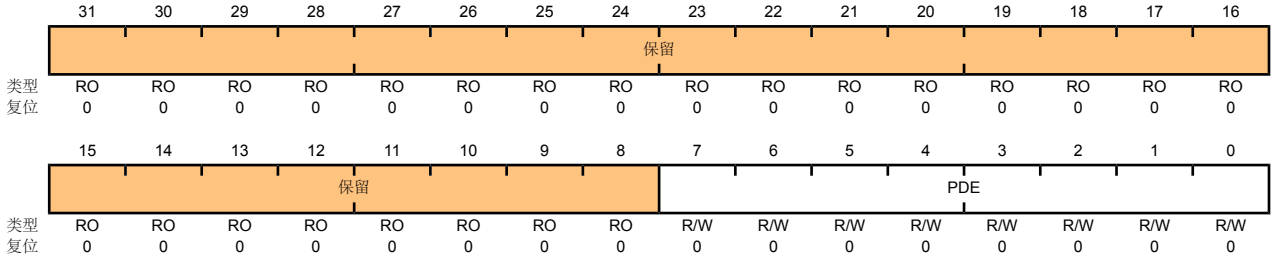
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PUE	R/W	-	端口弱上拉使能 向**GPIOPDR[n]**写1会清零相应的**GPIOPUR[n]**使能位。这种修改会在写操作之后的第二个时钟周期生效。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**, **GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD功能。正因为这样，对于GPIO端口B，该寄存器默认的复位值是0x0000.0080，而对于端口C，其默认的复位值是0x0000.000F。

寄存器 16: GPIO 下拉选择（GPIOPDR），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信号的弱下拉电阻。置位**GPIOPDR**中的位会使**GPIO** 上拉选择 **(GPIOPUR)**寄存器（见 173页）中相应的位自动清零。

GPIO 下拉选择 (GPIOPDR)

偏移量 0x514
类型 R/W, 复位 0x0000.0000



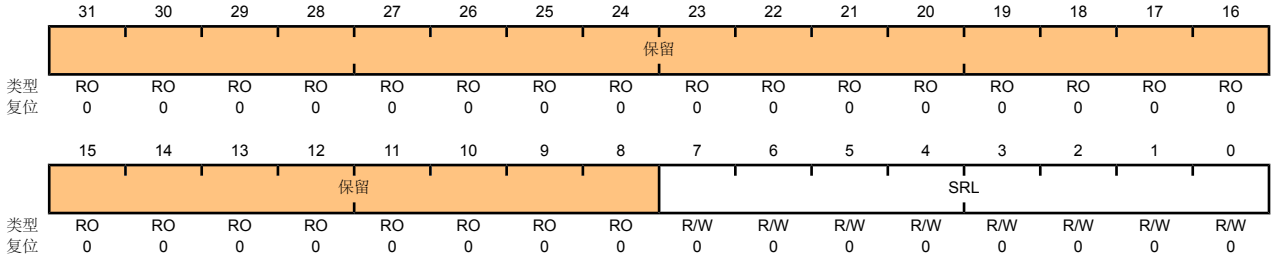
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PDE	R/W	0x00	端口弱下拉使能 向**GPIOPUR[n]**写1会清零相应的**GPIOPDR[n]**使能位。这种修改会在写操作之后的第二个时钟周期生效。

寄存器 17: GPIO 斜率控制选择（GPIOSLR），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（见 171页）采用8-mA驱动强度选项时才有效。

GPIO 斜率控制选择 (GPIOSLR)

偏移量 0x518
类型 R/W, 复位 0x0000.0000



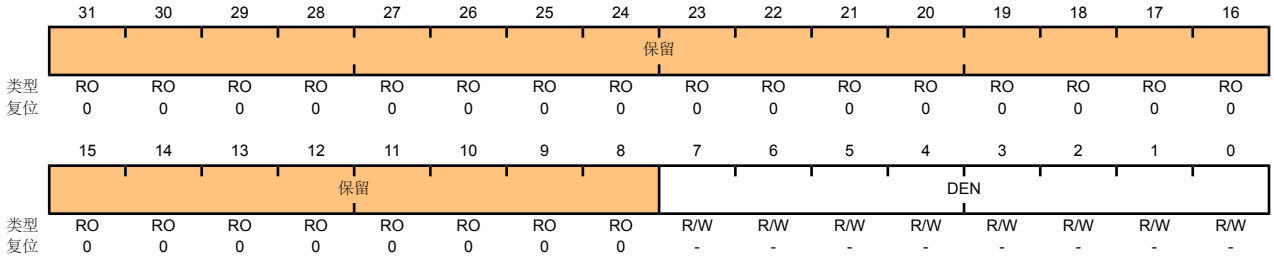
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	SRL	R/W	0x00	斜率限制使能（仅为8-mA驱动） SRL 值如下定义： 值 描述 0 斜率控制被禁能。 1 斜率控制被使能。

寄存器 18: GPIO 数字使能（GPIODEN），偏移量 0x51C

GPIODEN寄存器是数字使能寄存器。默认情况下，除了用作JTAG/SWD功能的的GPIO信号外，所有其它的GPIO信号都被配置为非驱动（三态）。它们的数字功能被禁能；它们不驱动管脚上的逻辑值且它们不允许管脚电压进入GPIO接收器。为了使用管脚的数字功能（GPIO或可选功能），相应的GPIODEN位必须置位。

GPIO 数字使能 (GPIODEN)

偏移量 0x51C
类型 R/W, 复位 -



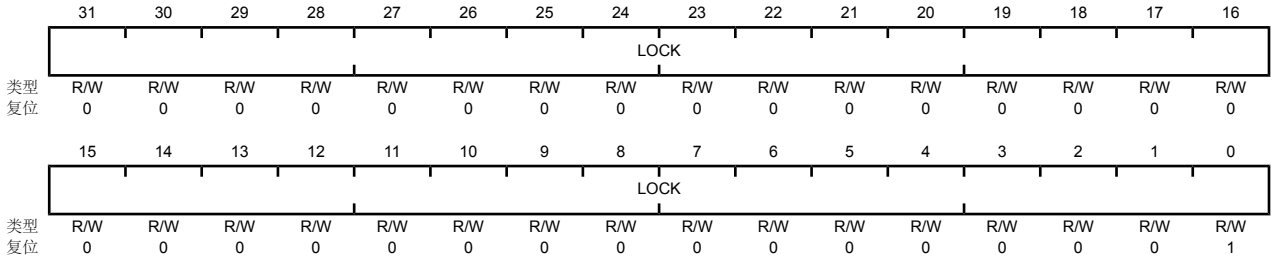
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DEN	R/W	-	数字使能 DEN 值如下定义： 值 描述 0 数字功能被禁能。 1 数字功能被使能。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**, **GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD 功能。正因为这样，对于GPIO端口B，该寄存器默认的复位值是0x0000.0080，而对于端口C，其默认的复位值是0x0000.000F。

寄存器 19: GPIO 锁定（GPIOLOCK），偏移量 0x520

GPIOLOCK 寄存器使能对 **GPIOCR** 寄存器（见 178页）的写访问。写 0x1ACCE551 到 **GPIOLOCK** 寄存器将解锁**GPIOCR** 寄存器。写其它任意值到 **GPIOLOCK** 寄存器重新使能锁定的状态。读取 **GPIOLOCK** 寄存器返回锁定状态，而不是先前写入的32-位值。因此，当写访问被禁能或锁定时，读 **GPIOLOCK** 寄存器返回 0x00000001。当写访问被使能或解锁时，读 **GPIOLOCK** 寄存器返回 0x00000000。

GPIO 锁定 (GPIOLOCK)

偏移量 0x520
类型 R/W, 复位 0x0000.0001



位/域	名称	类型	复位	描述
31:0	LOCK	R/W	0x0000.0001	GPIO 锁定

写 0x1ACCE551的值解锁 **GPIO** 确认 **(GPIOCR)** 寄存器以用于写访问。写其它任何值重新应用锁定，防止任何寄存器更新。读该寄存器返回下列的值:

值	描述
0x0000.0001	锁定
0x0000.0000	未锁定

寄存器 20: GPIO 确认（GPIOCR），偏移量 0x524

GPIOCR 寄存器是确认寄存器。当执行写 **GPIOAFSEL** 寄存器时，**GPIOCR** 寄存器的值确定了 **GPIOAFSEL** 寄存器中的哪些位将被确认。如果在 **GPIOCR** 寄存器中的一个位为0，那么写入 **GPIOAFSEL**寄存器中相应位的数据将不被确认并保留其原来的值。如果 **GPIOCR** 寄存器中的位为1，那么写入**GPIOAFSEL** 寄存器中相应位的数据将被确认到寄存器并反映新的值。

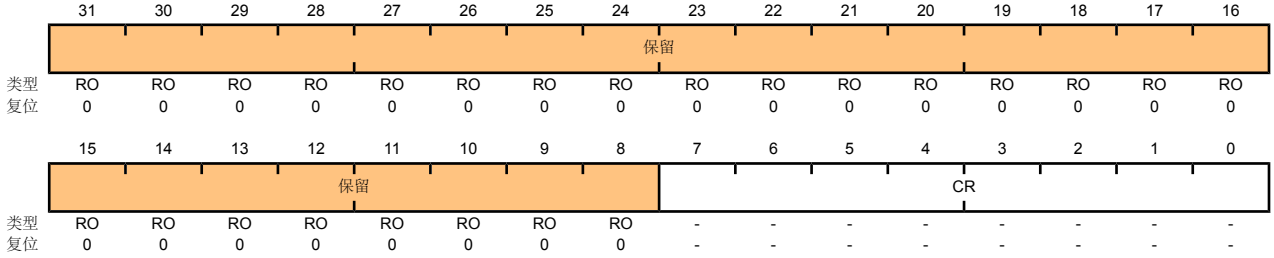
GPIOCR 寄存器的内容只有在 **GPIOLOCK** 寄存器解锁时才能被修改。如果**GPIOLOCK**寄存器被锁定，那么写 GPIOCR 寄存器将被忽略。

重要: 该寄存器用于防止意外地设置 **GPIOAFSEL** 寄存器，该寄存器控制到JTAG/SWD 调试硬件的连通性。通过将 **GPIOCR** 寄存器中对应的PB7和PC[3:0]位初始化为0，JTAG/SWD 调试端口只能通过对**GPIOLOCK**, **GPIOCR**和**GPIOAFSEL**寄存器的一系列写操作来转换为GPIO。

因为这种保护当前只在PB7 和 PC[3:0]的 JTAG/SWD 管脚上执行，所以 **GPIOCR** 寄存器中所有其它的值都不能被写入0x0。这些位硬连接（hardwired）为 0x1，确保总是可以确认新的值到其它管脚的**GPIOAFSEL**寄存器位。

GPIO 确认 (GPIOCR)

偏移量 0x524
类型 -, 复位 -



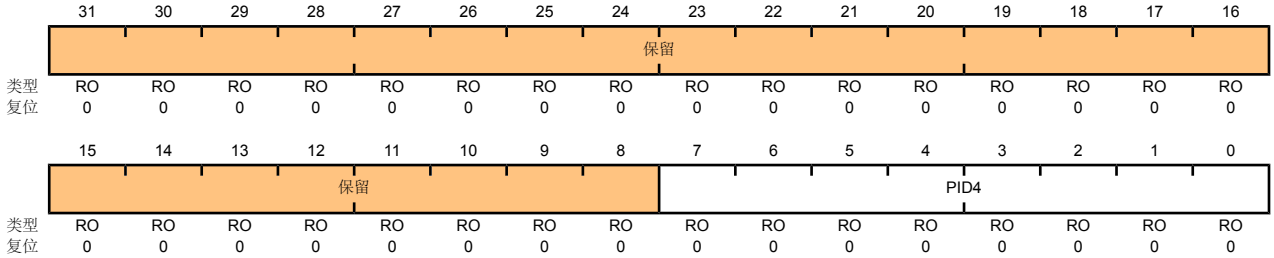
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CR	-	-	GPIO 确认（commit） 在按位（bit-wise）基础上，任何设置的位允许相应的GPIOAFSEL位被设为其备用功能。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOCR**寄存器默认的寄存器类型都是RO。这5个管脚是当前仅受**GPIOCR**寄存器保护的GPIO。正因为这样，对于GPIO 端口B7 和 GPIO 端口C[3:0]的寄存器类型是R/W。 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOCR**寄存器默认的复位值都是0x0000.00FF。为了确保JTAG端口不被意外地编程为GPIO，这5个管脚默认为不可确认（non-commitable）。正因为这样，对于GPIO端口B，**GPIOCR** 默认的复位值是0x0000.007F，而对于端口C，**GPIOCR**默认的复位值是0x0000.00F0。

寄存器 21: GPIO 外设标识4（GPIOPeriphID4），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)

偏移量 0xFD0
类型 RO, 复位 0x0000.0000



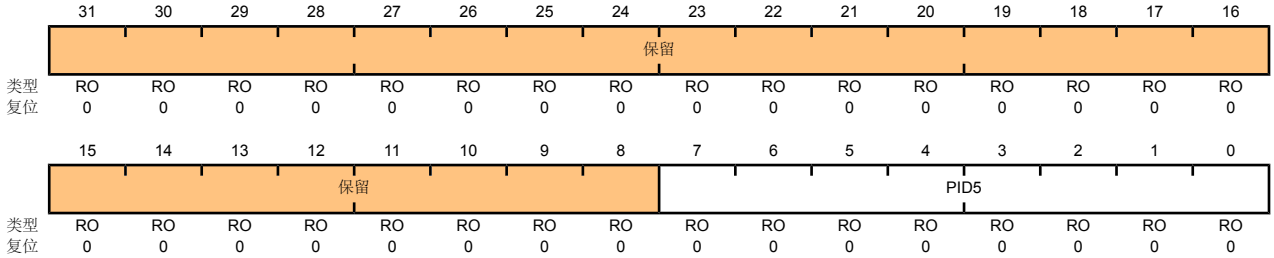
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	GPIO 外设ID寄存器 [7:0]

寄存器 22: GPIO 外设标识 5 （GPIOPeriphID5），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)

偏移量 0xFD4
类型 RO, 复位 0x0000.0000



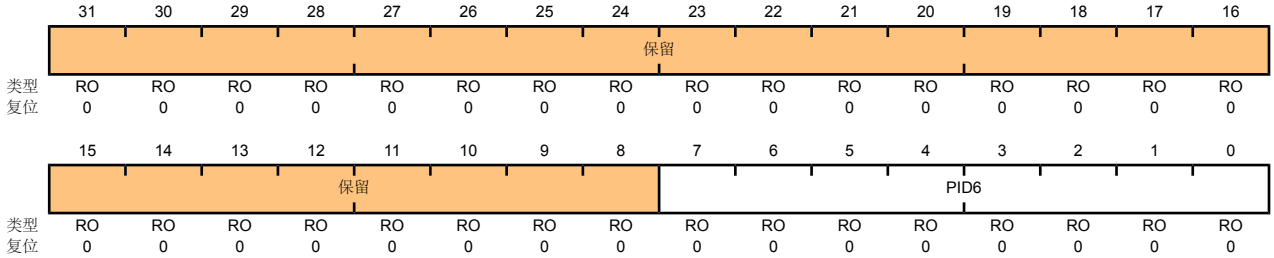
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	GPIO 外设ID寄存器 [15:8]

寄存器 23: GPIO 外设标识 6 （GPIOPeriphID6），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)

偏移量 0xFD8
类型 RO, 复位 0x0000.0000



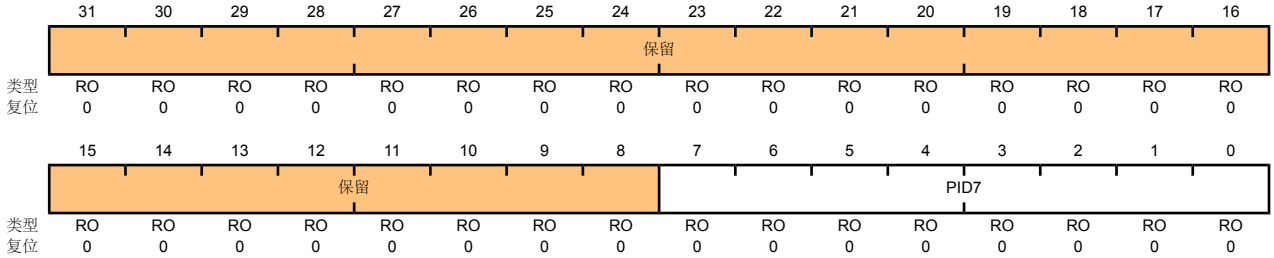
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	GPIO 外设ID寄存器 [23:16]

寄存器 24: GPIO 外设标识7（GPIOPeriphID7），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)

偏移量 0xFDC
类型 RO, 复位 0x0000.0000



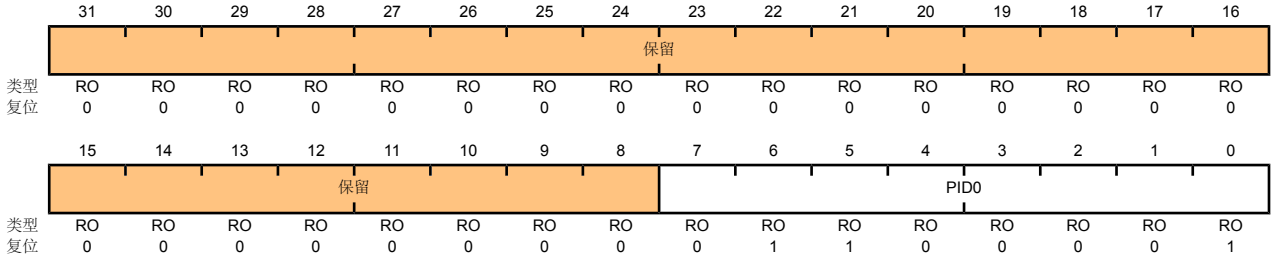
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	GPIO 外设ID寄存器 [31:24]

寄存器 25: GPIO 外设标识0 （GPIOPeriphID0），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)

偏移量 0xFE0
类型 RO, 复位 0x0000.0061



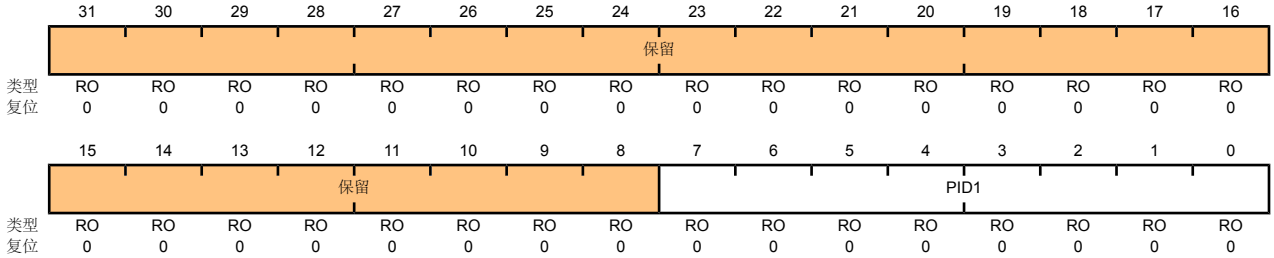
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x61	GPIO 外设ID寄存器 [7:0] 可被软件用来标识该外设是否存在。

寄存器 26: GPIO 外设标识1 （GPIOPeriphID1），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)

偏移量 0xFE4
类型 RO, 复位 0x0000.0000



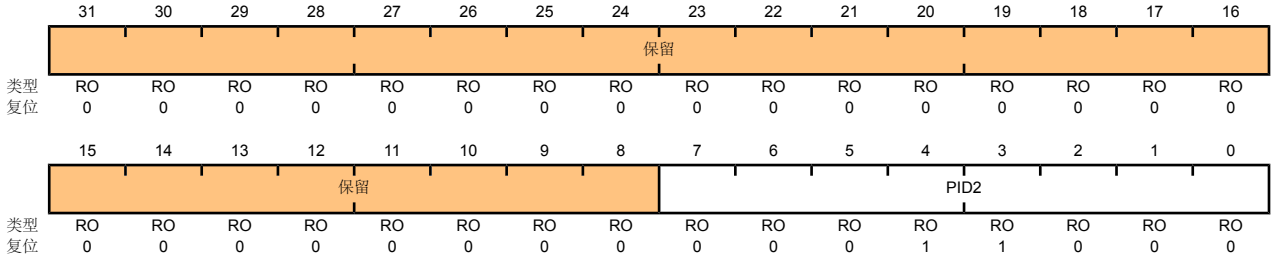
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	GPIO 外设ID寄存器 [15:8] 可被软件用来标识该外设是否存在。

寄存器 27: GPIO 外设标识2 （GPIOPeriphID2），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)

偏移量 0xFE8
类型 RO, 复位 0x0000.0018



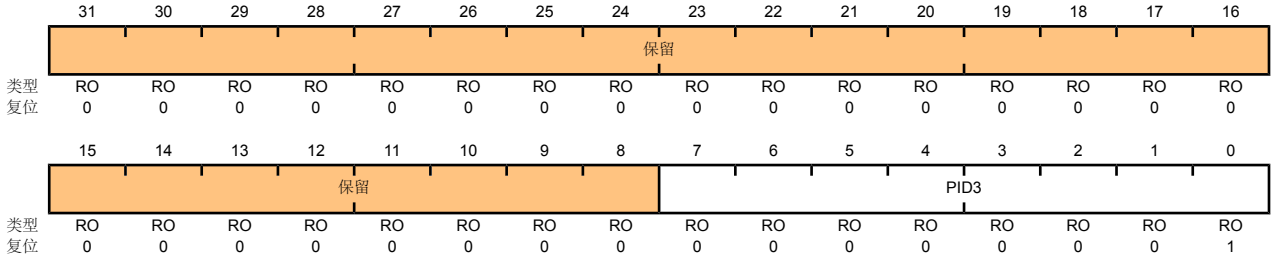
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	GPIO 外设ID寄存器 [23:16] 可被软件用来标识该外设是否存在。

寄存器 28: GPIO 外设标识3 （GPIOPeriphID3），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)

偏移量 0xFEC
类型 RO, 复位 0x0000.0001



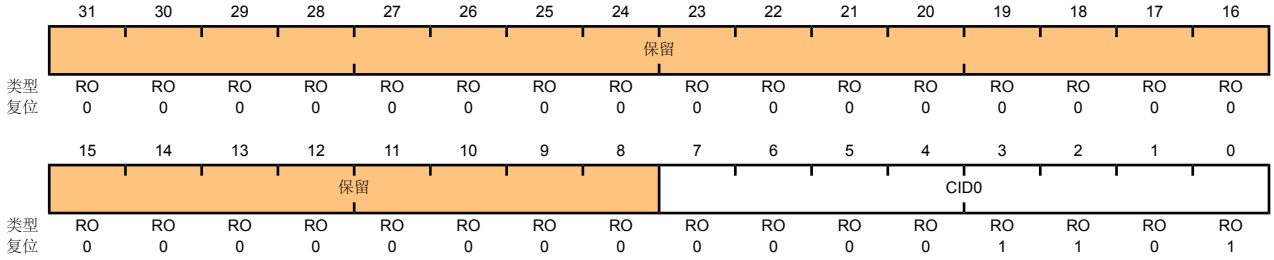
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	GPIO 外设ID寄存器 [31:24] 可被软件用来标识该外设是否存在。

寄存器 29: GPIO PrimeCell 标识0（GPIOPCellID0），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。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)

偏移量 0xFF0
类型 RO, 复位 0x0000.000D



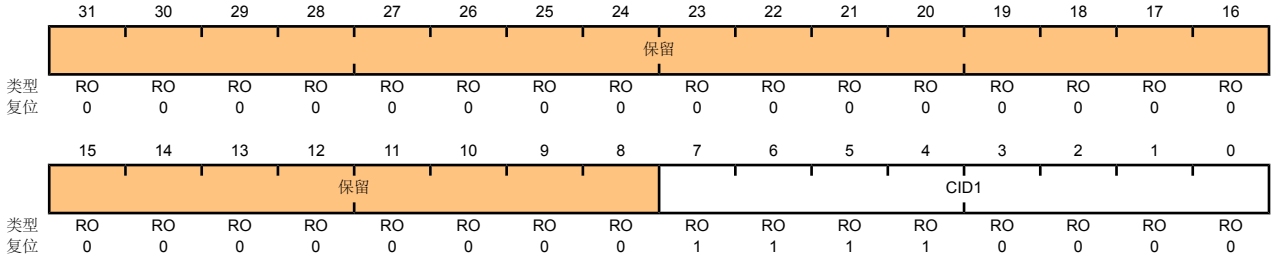
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	GPIO PrimeCell ID寄存器 [7:0] 向软件提供一个标准的交叉外设标识系统。

寄存器 30: GPIO PrimeCell 标识1 （GPIOPCellID1），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。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)

偏移量 0xFF4
类型 RO, 复位 0x0000.00F0



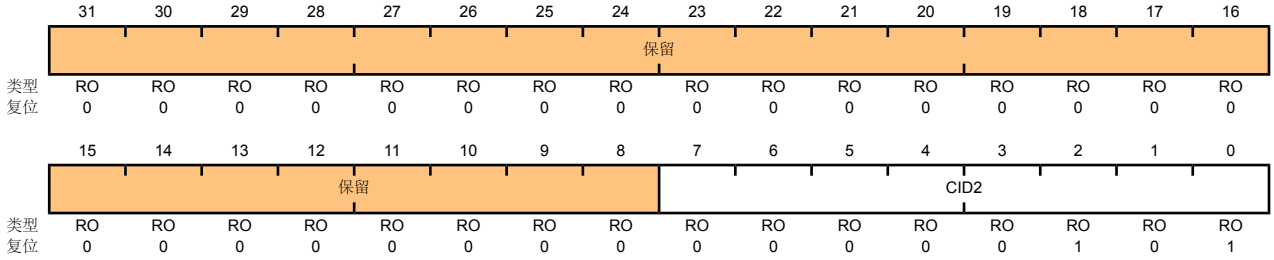
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	GPIO PrimeCell ID寄存器 [15:8] 向软件提供一个标准的交叉外设标识系统。

寄存器 31: GPIO PrimeCell 标识2（GPIOPCellID2），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。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)

偏移量 0xFF8
类型 RO, 复位 0x0000.0005



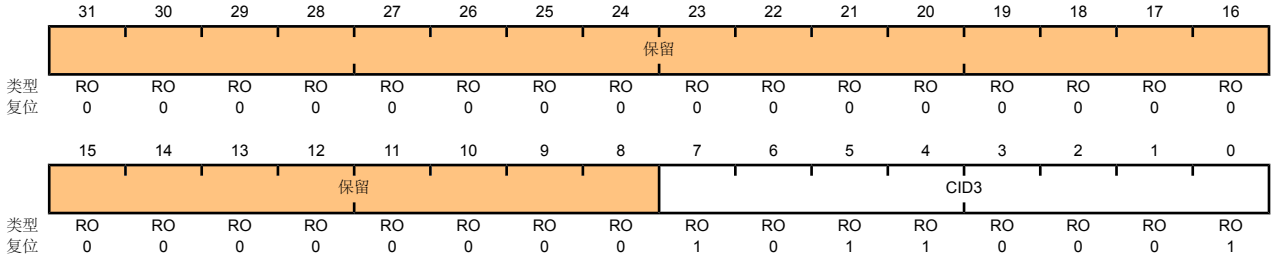
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	GPIO PrimeCell ID寄存器 [23:16] 向软件提供一个标准的交叉外设标识系统。

寄存器 32: GPIO PrimeCell 标识3（GPIOPCellID3），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。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3 (GPIOPCellID3)

偏移量 0xFFC
类型 RO, 复位 0x0000.00B1



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID3	RO	0xB1	GPIO PrimeCell ID寄存器 [31:24] 向软件提供一个标准的交叉外设标识系统。

10 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。Stellaris® 通用定时器模块 (GPTM) 包含 4个 GPTM 模块 (定时器0, 定时器1, 定时器 2和定时器 3)。每个GPTM模块包含两个16位的定时器/计数器（称作TimerA和TimerB），用户可以将它们配置成独立运行的定时器 或事件计数器，或将它们配置成1个32位定时器 或一个32位实时时钟 (RTC)。

注意: 定时器2是一个内部定时器，只能用来产生内部中断。

通用定时器模块是 Stellaris® 微控制器的一个定时资源。其它定时器资源还包括系统定时器 (SysTick) (见 “系统定时器 (SysTick)” 在 37页) 和PWM模块中的PWM定时器 (见 “PWM定时器” 在 400页)。

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
  - 可编程周期定时器
  - 使用32.768-KHz 输入时钟的实时时钟
  - 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模块 (仅单次触发模式和周期模式)
  - 可编程单次触发（one-shot）定时器
  - 可编程周期定时器
  - 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。

10.1 结构图

注意: 在图 10-1 在 192页中，可使用的特定 CCP 管脚由 Stellaris® 器件来决定。可使用的CCP见表 10-1 在 192页。

图 **10-1. GPTM**模块的结构图

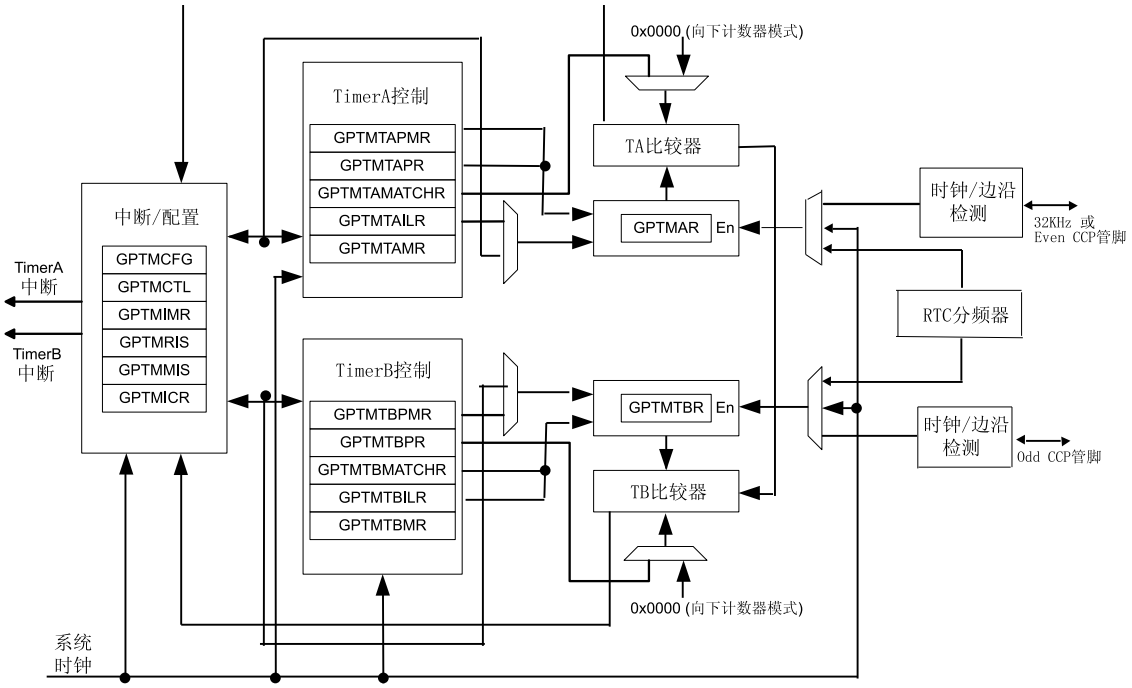


表 **10-1.** 可使用的**CCP**管脚

定时器	**16-**位向上**/**向下计数器	偶数**CCP**管脚	奇数**CCP**管脚
定时器0	TimerA	CCP0	-
	TimerB	-	CCP1
定时器1	TimerA	CCP2	-
	TimerB	-	CCP3
定时器2	TimerA	CCP4	-
	TimerB	-	CCP5
定时器3	TimerA	-	-
	TimerB	-	-

10.2 功能描述

每个GPTM模块的主要元件包括两个自由运行的先递增后递减计数器（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 (见 202页)、**GPTM TimerA** 模式 **(GPTMTAMR)** 寄存器 (见 203页)和 **GPTM TimerB** 模式 **(GPTMTBMR)** 寄存器 (见 205页)。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。

10.2.1 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（见 216页） 和 **GPTM TimerB** 间隔装载 （**GPTMTBILR**）寄存器（见 217页）一起初始化为0xFFFF。

预分频计数器：**GPTM TimerA** 预分频**(GPTMTAPR)** 寄存器 (见 220页) 和**GPTM TimerB** 预分频**(GPTMTBPR)** 寄存器 (见 221页)初始化为0x00。

10.2.2 32-位定时器工作模式

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/32位周期定时器模式）或1（RTC模式），可将GPTM模块配置为32位模式。在两种配置中，都需将某些GPTM寄存器连在一起形成伪32位寄存器。这些寄存器包括：

- **GPTM TimerA** 间隔装载（**GPTMTAILR**） 寄存器 [15:0]，见 216页
- **GPTM TimerB** 间隔装载（**GPTMTBILR**） 寄存器 [15:0]，见 217页
- **GPTM TimerA**（**GPTMTAR**） 寄存器 [15:0]，见 224页
- **GPTM TimerB** （**GPTMTBR**） 寄存器 [15:0]，见 225页

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：

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

同样，对**GPTMTAR**的读操作返回的值为：

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

10.2.2.1 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 (见 203页)的 TAMR 位域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 (见 207页)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0x0000.0000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，它将继续计数。

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 (见 212页)中的TATORIS位置位，并保持该值直到向**GPTM** 中断清零 **(GPTMICR)** 寄存器 (见 214页)执行写操作将其清零。如果**GPTM** 中断屏蔽 **(GPTIMR)** 寄存器 (见 210页)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 (见 213页)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。

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。

10.2.2.2 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0x0000.0001。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 (见 218页) 。

在RTC模式中，要求CCP0, CCP2或CCP4管脚上的输入时钟为32.768KHz。然后将时钟信号分频为1 Hz，将其传送给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0x0000.0001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0x0000.0000并继续计数，直到出现硬件复位或被软件禁能（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，那么GPTM 也会将**GPTMISR**中的 RTCMIS位置位并产生一个控制器中断。 通过写 **GPTMICR**的RTCCINT位可将状态标志清零。

如果**GPTMCTL**寄存器中的 TASTALL位和/或 TBSTALL 位置位，那么定时器在**GPTMCTL**的RTCEN位置位时不会停止计数。

10.2.3 16-位定时器的工作模式

通过向**GPTM**配置 **(GPTMCFG)** 寄存器 (见 202页)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的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。

10.2.3.1 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**寄存器中TnMR位域的值来决定。可选预分频器中的值被加载到**GPTM Timern** 预分频 **(GPTMTnPR)**寄存器中。

在软件对**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。如果配置为周期定时器，它将继续计数。

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**GPTIMR**的超时中断使能，则GPTM还将 **GPTMISR**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，并且可以触发启动转换（SoC）事件。

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**寄存器中的 TnSTALL 位被使能，那么定时器停止（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。所有值都是以时钟频率 50-MHz （时钟周期Tc=20 ns）作为标准进行计算。

表 **10-2.** 带预分频器配置的**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

预分频	#时钟 (T c)[a]	最大时间	单位
11111111	256	335.5443	mS

a. Tc表示时钟周期。

10.2.3.2 16-位输入边沿计数模式

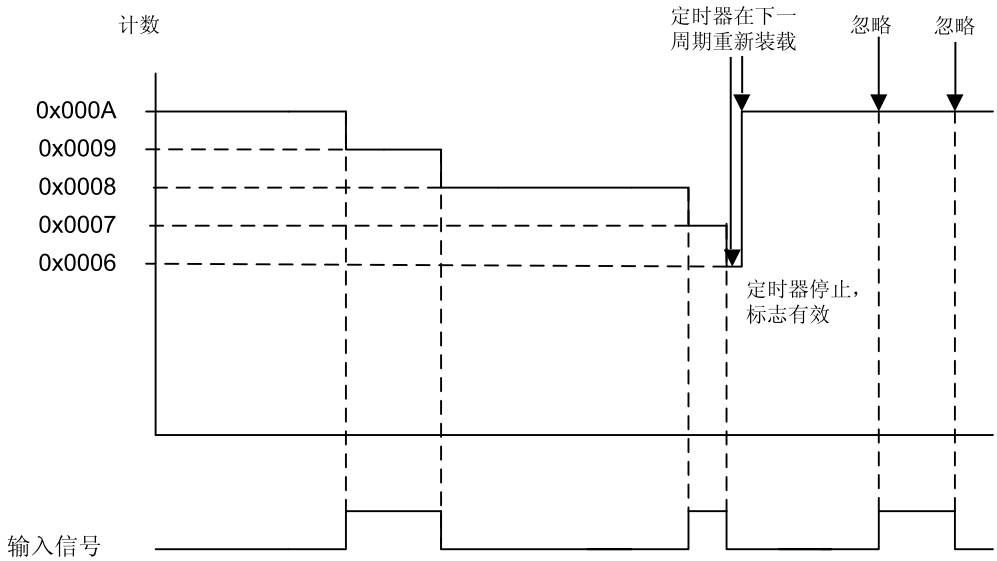
在边沿计数模式中，定时器被配置为能够捕获3种事件类型的递减计数器，这3种事件类型为上升沿、下降沿、或上升/下降沿。为了把定时器设置为边沿计数模式，**GPTMTnMR**寄存器的TnCMR 位必须设为0。定时器计数时所采用的边沿类型由**GPTMCTL**寄存器的TnEVENT位域决定。在初始化过程中，需对**GPTM Timern** 匹配 **(GPTMTnMATCHR)**寄存器进行配置，以便**GPTMTnILR**寄存器和**GPTMTnMATCHR**寄存器之间的差值等于必须计算的边沿事件的数目。

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**GPTMTnMATCHR**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 10-2 在 195页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**GPTMnILR**=0x000A，匹配值**GPTMnMATCHR**=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上升/下降沿。

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 **10-2. 16**位输入边沿计数模式实例



10.2.3.3 16-位输入边沿定时模式

注意: 16位输入边沿计时模式中不含预分频器。

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TnCMR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**GPTMCnTL**寄存器的TnEVENT位域来决定。

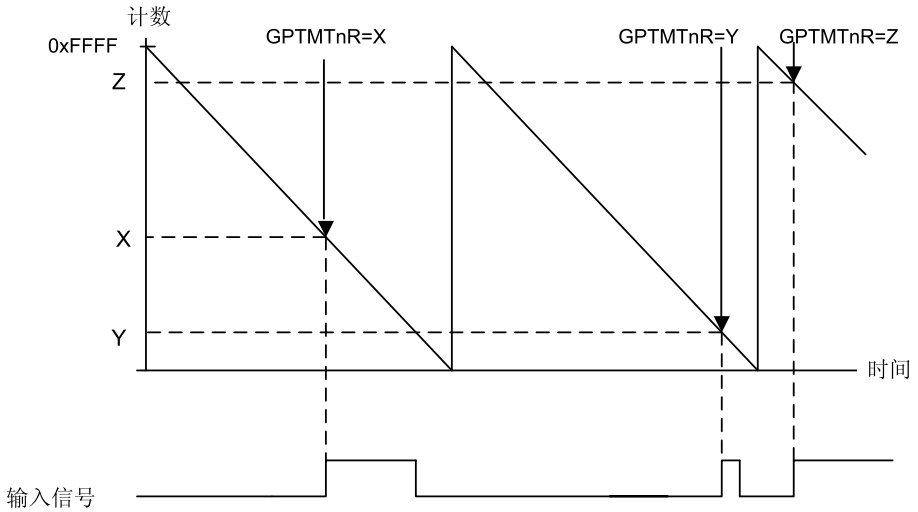
在软件写**GPTMCTL**寄存器的TnEN 位时，定时器使能并用于事件捕获。在检测到所选的输入事件时，从**GPTMTnR**寄存器中捕获**Tn**计数器的当前值，且该值可通过控制器来读取。 然后GPTM让CnERIS位有效（如果中断没有被屏蔽，则也让CnEMIS位有效）。

在捕获到事件之后，定时器不会停止计数。它会继续计数，直至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10-3在196页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 **10-3. 16**位输入边沿定时模式实例



10.2.3.4 16-位**PWM**模式

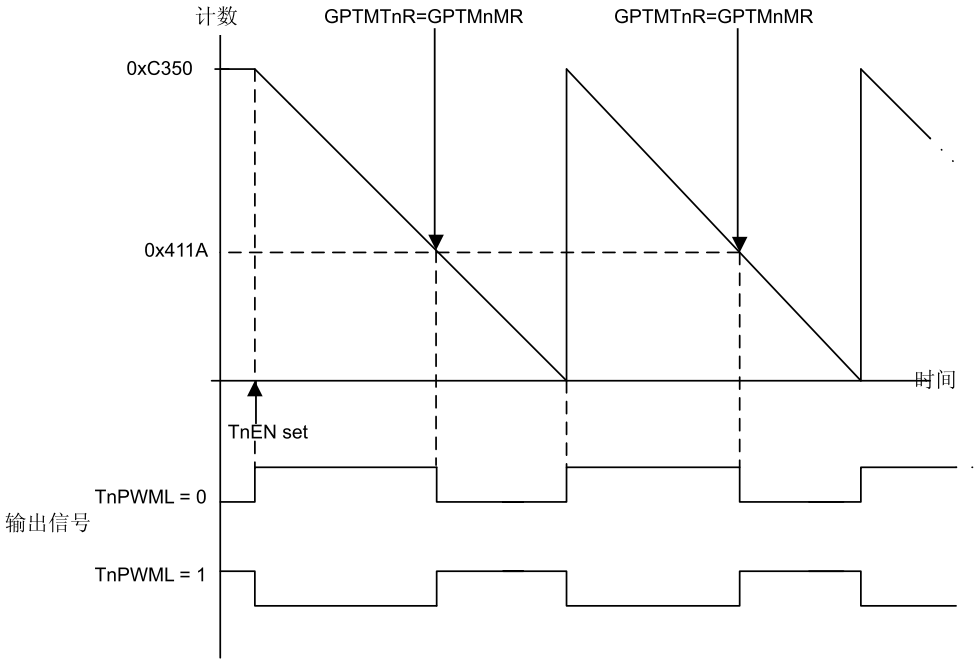
GPTM支持简单的PWM生成模式。在PWM模式中，定时器配置为递减计数器，初值由**GPTMTnILR**定义。通过将**GPTMTnMR**寄存器的TnAMS位置为0x1、TnCMR位置为0x0、TnMR位域置为0x2来使能PWM模式。

在软件写**GPTMCTL**寄存器的 TnEN 位时，计数器开始递减计数，直到计数值到达0x0000。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 **(GPTMnMATCHR)**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 10-4 在 197页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，初值在这个例子中，初值**GPTMnIRL**=0xC350，匹配值**GPTMnMR**=0x411A。

图 **10-4. 16-**位**PWM**模式实例



10.3 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TIMER0、 TIMER1、TIMER2和TIMER3 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。

10.3.1 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. 写0x0到 **GPTM** 配置寄存器 **(GPTMCFG)** 。
3. 设置**GPTM TimerA** 模式寄存器 **(GPTMTAMR)**的 TAMR 位域:
   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** 间隔装载寄存器 （**GPTMTAILR**）。
5. 如果需要中断，将**GPTM** 中断屏蔽寄存器 **(GPTMIMR)**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** 中断清零寄存器 **(GPTMICR)**的TATOCINT位写1将状态标志清零。

在单次触发模式中，定时器在step 7 在 198页之后停止计数 。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。

10.3.2 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. 写0x1到**GPTM** 配置寄存器 **(GPTMCFG)**。
3. 向**GPTM TimerA** 匹配寄存器 **(GPTMTAMATCHR)**写入所需的匹配值。
4. 根据需要将**GPTM** 控制寄存器 **(GPTMCTL)**的RTCEN位置位/清零。
5. 如果需要中断，将**GPTM** 中断屏蔽寄存器 **(GPTMIMR)**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0x0000.0000并开始计数。如果中断使能，不必将其清除。

10.3.3 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. 写0x4到 **GPTM** 配置寄存器 **(GPTMCFG)** 。
3. 设置**GPTM** 定时器模式 **(GPTMTnMR)**寄存器的TnMR位域:
   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** 定时器**n** 预分频寄存器 **(GPTMTnPR)**。
5. 将初值装入**GPTM** 定时器间隔装载寄存器 **(GPTMTnILR)**。
6. 如果需要中断，将**GPTM** 中断屏蔽寄存器 **(GPTMIMR)**的TnTOIM位置位。
7. 置位**GPTM** 控制寄存器 **(GPTMCTL)**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** 中断清零寄存器 **(GPTMICR)**的TnTOCINT位写1将状态标志清零。

在单次触发模式中，定时器在step 8 在 198页之后停止计数 。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。

10.3.4 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. 写0x4到 **GPTM** 配置 **(GPTMCFG)** 寄存器。
3. 在 **GPTM** 定时器模式**(GPTMTnMR)** 寄存器中，写0x0到 TnCMR位域，写0x3到 TnMR 位域。
4. 通过对**GPTM** 控制 **(GPTMCTL)**寄存器的TnEVENT位域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**n** 间隔装载 **(GPTMTnILR)**寄存器。
6. 将所需的事件数装入**GPTM** 定时器**n** 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**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 step 4 在 199页到step 9 在 199页才能重新使能定时器。

10.3.5 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（TnEN位清零）。
2. 写0x4到 **GPTM** 配置 **(GPTMCFG)** 寄存器。
3. 在**GPTM** 定时器模式 **(GPTMTnMR)** 寄存器中，写0x1到 TnCMR 位域，写0x3到 TnMR 位域。
4. 通过写**GPTM** 控制 **(GPTMCTL)**寄存器的TnEVENT位域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**n** 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写1将状态标志清零。事件发生的时间可通过读 **GPTM** 定时器**n** **(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，但通过写**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。

10.3.6 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（TnEN位清零）。

2. 写0x4到 **GPTM** 配置 **(GPTMCFG)** 寄存器。
3. 在**GPTM** 定时器模式 **(GPTMTnMR)** 寄存器中，将 TnAMS 位置为 0x1，将TnCMR 位置为 0x0，将 TnMR 位域置为 0x2。
4. 在**GPTM** 控制 **(GPTMCTL)**寄存器的TnEVENT位域中，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** 定时器**n** 间隔装载 **(GPTMTnILR)**寄存器。
6. 将所需的值装入**GPTM** 定时器**n** 匹配 **(GPTMTnMATCHR)**寄存器。
7. 如果使用预分频器，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定时模式中，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。

10.4 寄存器映射

表 10-3 在 200页列出了GPTM寄存器。所列偏移量都是寄存器相对于定时器基址的16进制增量：

- Timer0: 0x4003.0000
- Timer1: 0x4003.1000
- Timer2: 0x4003.2000
- Timer3: 0x4003.3000

表 **10-3.** 定时器 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	GPTMCFG	R/W	0x0000.0000	GPTM 配置	202
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA 模式	203
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB 模式	205
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM 控制	207
0x018	GPTMIMR	R/W	0x0000.0000	GPTM 中断屏蔽	210
0x01C	GPTMRIS	RO	0x0000.0000	GPTM 原始中断状态	212
0x020	GPTMMIS	RO	0x0000.0000	GPTM屏蔽后的中断状态	213
0x024	GPTMICR	W1C	0x0000.0000	GPTM中断清零	214
0x028	GPTMTAILR	R/W	0x0000.FFFF（16位模式） 和 0xFFFF.FFFF（32位模式）	GPTM TimerA间隔装载	216
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB间隔装载	217

偏移量	名称	类型	复位	描述	见页面
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF（16位模式）和0xFFFF.FFFF（32位模式）	GPTM TimerA匹配	218
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB匹配	219
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA预分频	220
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB预分频	221
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA预分频匹配	222
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB预分频匹配	223
0x048	GPTMTAR	RO	0x0000.FFFF（16位模式）和0xFFFF.FFFF（32位模式）	GPTM TimerA	224
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	225

10.5 寄存器描述

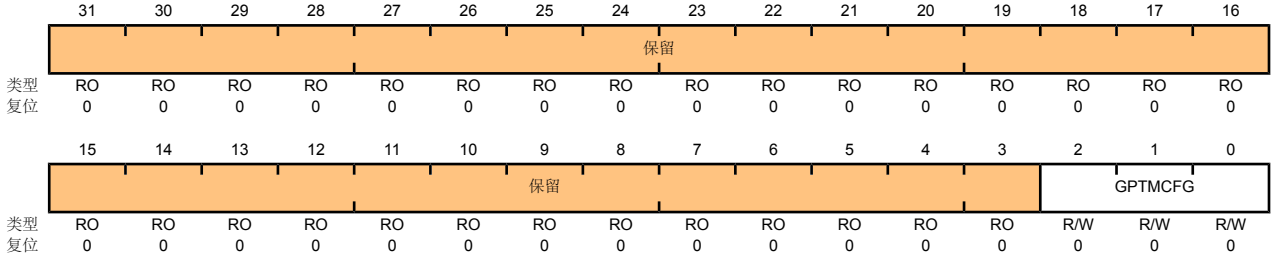
下文将按地址偏移量的数字顺序列出GPTM寄存器，并对它们进行描述。

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
类型 R/W, 复位 0x0000.0000



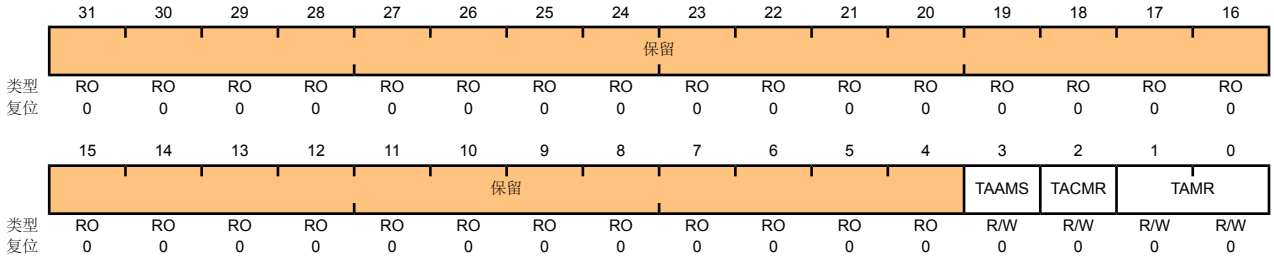
位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2:0	GPTMCFG	R/W	0x0	GPTM 配置 GPTMCFG 值如下定义： 值 描述 0x0 32-位定时器配置。 0x1 32-位实时时钟(RTC) 计数器配置。 0x2 保留。 0x3 保留。 0x4-0x7 16-位定时器配置，功能由**GPTMTAMR** 和 **GPTMTBMR**的位1:0来控制。

寄存器 2: GPTM TimerA 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。当在16-位PWM模式时，设置TAAMS位为0x1, TACMR位为0x0, 以及TAMR 域为0x2。

GPTM TimerA 模式 (GPTMTAMR)

偏移量 0x004
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TAAMS	R/W	0	GPTM TimerA 备用模式选择 TAAMS 值如下定义： 值 描述 0 捕获模式被使能。 1 PWM 模式被使能。 注意: 要使能PWM模式，必须清零 TACMR 位并将 TAMR 位域置为 0x2。
2	TACMR	R/W	0	GPTM TimerA 捕获模式 TACMR 值如下定义： 值 描述 0 边沿-计数模式。 1 边沿-定时模式。

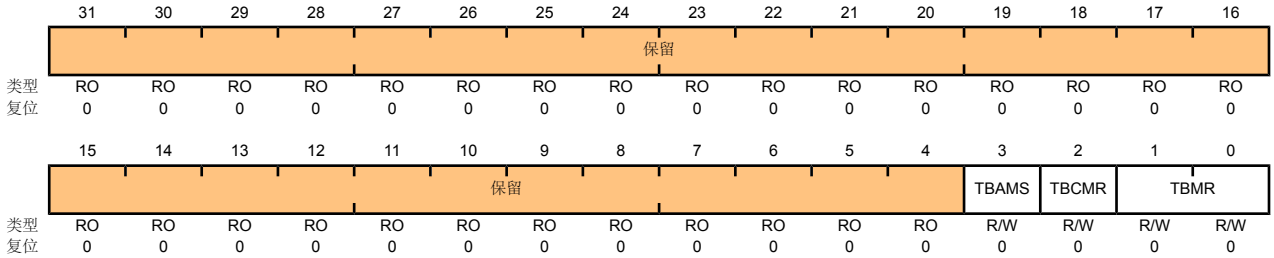
位/域	名称	类型	复位	描述
1:0	TAMR	R/W	0x0	GPTM TimerA 模式 TAMR值如下定义: 值 描述 0x0 保留。 0x1 单次触发定时器模式。 0x2 周期定时器模式。 0x3 捕获模式。 定时器模式基于**GPTMCFG**寄存器（16-或32-位）2:0位定义的定时器配置。 在16位定时器配置中，TAMR 控制TimerA的16位定时器模式。 在32位定时器配置中，该寄存器用来控制模式并且**GPTMTBMR**的内容被忽略。

寄存器 3: GPTM TimerB 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。当为16-位 PWM 模式时，将TBAMS位置为 0x1，将TBCMR位置为 0x0，将 TBMR域置为 0x2。

GPTM TimerB 模式 (GPTMTBMR)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TBAMS	R/W	0	GPTM TimerB 备用模式选择 TBAMS 值如下定义： 值 描述 0 捕获模式被使能。 1 PWM 模式被使能。 注意: 要使能 PWM 模式，你必须清零 TBCMR 位，并将 TBMR 位域置为 0x2。
2	TBCMR	R/W	0	GPTM TimerB 捕获模式 TBCMR 值如下定义： 值 描述 0 边沿-计数模式。 1 边沿-定时模式。

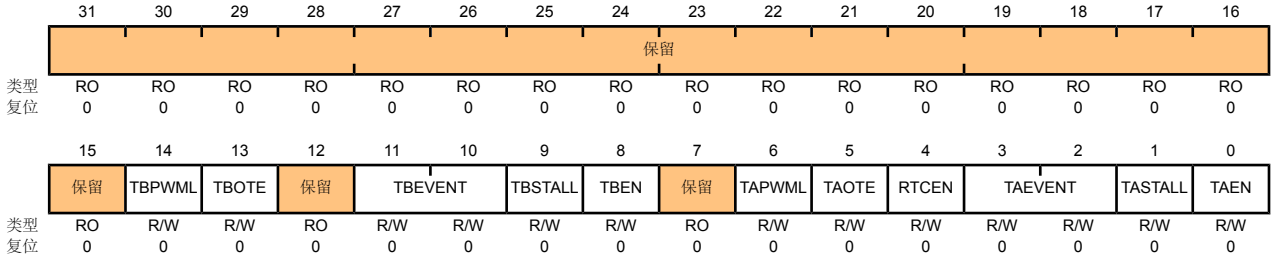
位/域	名称	类型	复位	描述
1:0	TBMR	R/W	0x0	GPTM TimerB 模式 TBMR 值如下定义： 值 描述 0x0 保留。 0x1 单次触发定时器模式。 0x2 周期定时器模式。 0x3 捕获模式。 定时器模式基于**GPTMCFG**寄存器的 2:0位定义的定时器配置。 在16-位定时器配置中，这些位用来控制TimerB的16位定时器模式。 在32-位定时器配置中，这个寄存器的内容被忽略，并且使用了**GPTMTAMR**。

寄存器 4: GPTM 控制（GPTMCTL），偏移量 0x00C

该寄存器与**GPTMCFG** 和 **GMTMTnMR**寄存器一起使用，对定时器配置进行微小调整，并使能其它特性诸如定时器停止。

GPTM 控制 (GPTMCTL)

偏移量 0x00C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:15	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
14	TBPWML	R/W	0	GPTM TimerB PWM 输出电平 TBPWML 值如下定义： 值 描述 0 输出不受影响。 1 输出反相。
13	TBOTE	R/W	0	GPTM TimerB 输出触发使能标志 TB0TE 值如下定义： 值 描述 0 输出TimerB 触发器被禁能。 1 输出 TimerB 触发器被使能。
12	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11:10	TBEVENT	R/W	0x0	GPTM TimerB 事件模式 TBEVENT 值如下定义： 值 描述 0x0 上升沿。 0x1 下降沿。 0x2 保留 0x3 双边沿。

位/域	名称	类型	复位	描述
9	TBSTALL	R/W	0	GPTM TimerB 停止使能标志 TBSTALL 值如下定义： 值 描述 0 禁止TimerB 停止。 1 使能TimerB 停止。
8	TBEN	R/W	0	GPTM TimerB使能标志 TBEN 值如下定义： 值 描述 0 TimerB 禁止。 1 根据**GPTMCFG**寄存器，TimerB使能并开始计数或使能捕获逻辑。
7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	TAPWML	R/W	0	GPTM TimerA的PWM输出电平 TAPWML 值如下定义： 值 描述 0 输出不受影响。 1 输出反相。
5	TAOTE	R/W	0	GPTM TimerA 的输出触发使能标志 TAOTE 值如下定义： 值 描述 0 输出 TimerA 触发器被禁能。 1 输出 TimerA 触发器被使能。
4	RTCEN	R/W	0	GPTM RTC 使能标志 RTCEN 值如下定义： 值 描述 0 RTC 计数被禁能。 1 RTC 计数被使能。

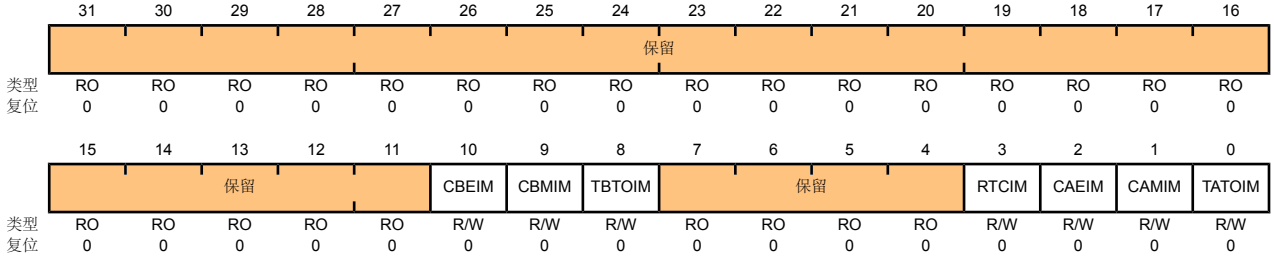
位/域	名称	类型	复位	描述
3:2	TAEVENT	R/W	0x0	GPTM TimerA 事件模式 TAEVENT 值如下定义： 值 描述 0x0 上升沿。 0x1 下降沿。 0x2 保留 0x3 双边沿。
1	TASTALL	R/W	0	GPTM TimerA 停止使能标志 TASTALL 值如下定义： 值 描述 0 禁止TimerA 停止。 1 使能TimerA 停止。
0	TAEN	R/W	0	GPTM TimerA使能标志 TAEN 值如下定义： 值 描述 0 禁止TimerA。 1 根据**GPTMCFG**寄存器，TimerA使能并开始计数或使能捕获逻辑。

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
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBEIM	R/W	0	GPTM CaptureB事件的中断屏蔽标志 CBEIM 值如下定义： 值 描述 0 中断禁止。 1 中断使能。
9	CBMIM	R/W	0	GPTM CaptureB匹配的中断屏蔽标志 CBMIM 值如下定义： 值 描述 0 中断禁止。 1 中断使能。
8	TBTOIM	R/W	0	GPTM TimerB超时的中断屏蔽标志 TBTOIM 值如下定义： 值 描述 0 中断禁止。 1 中断使能。
7:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	RTCIM	R/W	0	GPTM RTC的中断屏蔽标志 RTCIM 值如下定义： 值 描述 0 中断禁止。 1 中断使能。

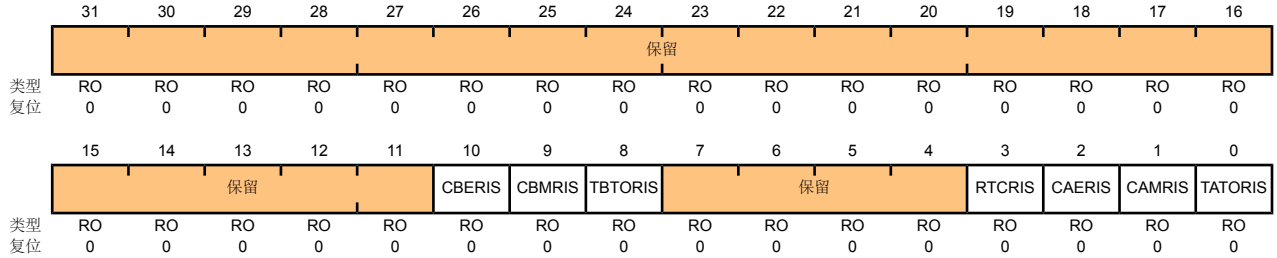
位/域	名称	类型	复位	描述
2	CAEIM	R/W	0	GPTM CaptureA事件的中断屏蔽标志 CAEIM 值如下定义: 值 描述 0 中断禁止。 1 中断使能。
1	CAMIM	R/W	0	GPTM CaptureA匹配的中断屏蔽标志 CAMIM 值如下定义: 值 描述 0 中断禁止。 1 中断使能。
0	TATOIM	R/W	0	GPTM TimerA超时的中断屏蔽标志 TATOIM 值如下定义: 值 描述 0 中断禁止。 1 中断使能。

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
类型 RO, 复位 0x0000.0000



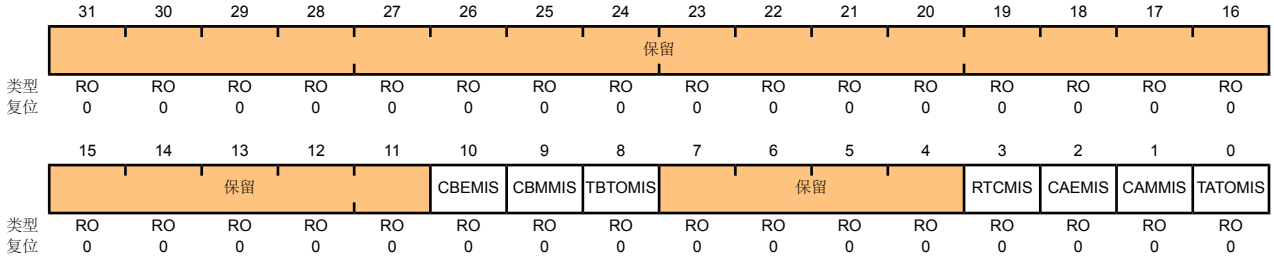
位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBERIS	RO	0	GPTM CaptureB事件的原始中断标志 该位表示屏蔽之前CaptureB事件的中断状态。
9	CBMRIS	RO	0	GPTM CaptureB匹配的原始中断标志 该位表示屏蔽之前CaptureB匹配的中断状态。
8	TBTORIS	RO	0	GPTM TimerB超时的原始中断标志 该位表示屏蔽之前TimerB超时的中断状态。
7:4	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	RTCRIS	RO	0	GPTM RTC的原始中断标志 该位表示屏蔽之前RTC事件的中断状态。
2	CAERIS	RO	0	GPTM CaptureA事件的原始中断标志 该位表示屏蔽之前CaptureA事件的中断状态。
1	CAMRIS	RO	0	GPTM CaptureA匹配的原始中断标志 该位表示屏蔽之前CaptureA匹配的中断状态。
0	TATORIS	RO	0	GPTM TimerA超时的原始中断标志 该位表示屏蔽之前TimerA超时的中断状态。

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
类型 RO, 复位 0x0000.0000



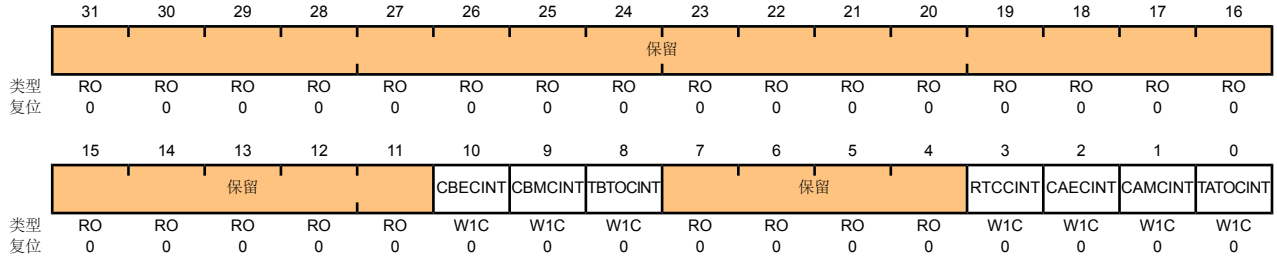
位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBEMIS	RO	0	GPTM CaptureB事件屏蔽后的中断标志 该位表示屏蔽之后CaptureB事件的中断状态。
9	CBMMIS	RO	0	GPTM CaptureB匹配屏蔽后的中断标志 该位表示屏蔽之后CaptureB匹配的中断状态。
8	TBTOMIS	RO	0	GPTM TimerB超时屏蔽后的中断标志 该位表示屏蔽之后TimerB超时的中断状态。
7:4	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	RTCMIS	RO	0	GPTM RTC屏蔽后的中断标志 该位表示屏蔽之后RTC事件的中断状态。
2	CAEMIS	RO	0	GPTM CaptureA事件屏蔽后的中断标志 该位表示屏蔽之后CaptureA事件的中断状态。
1	CAMMIS	RO	0	GPTM CaptureA匹配屏蔽后的中断标志 该位表示屏蔽之后CaptureA匹配的中断状态。
0	TATOMIS	RO	0	GPTM TimerA超时屏蔽后的中断标志 该位表示屏蔽之后TimerA超时的中断状态。

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
类型 W1C, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBECINT	W1C	0	GPTM CaptureB事件的中断清零标志 CBECINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。
9	CBMCINT	W1C	0	GPTM CaptureB匹配的中断清零标志 CBMCINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。
8	TBTOCINT	W1C	0	GPTM TimerB超时的中断清零标志 TBTOCINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。
7:4	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

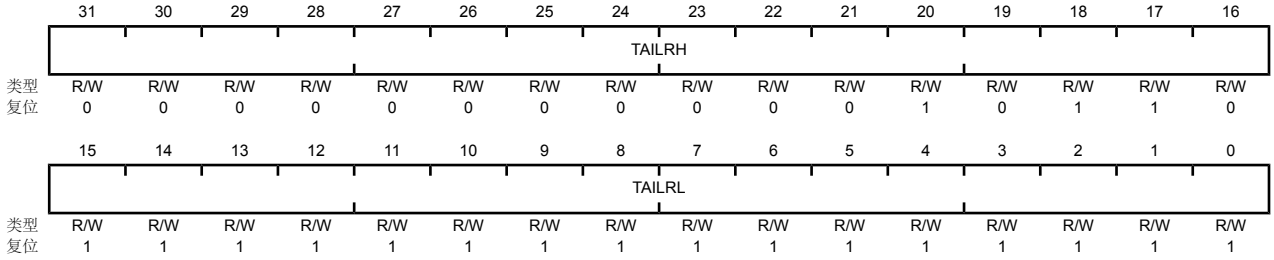
位/域	名称	类型	复位	描述
3	RTCCINT	W1C	0	GPTM RTC的中断清零标志 RTCCINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。
2	CAECINT	W1C	0	GPTM CaptureA事件的中断清零标志 CAECINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。
1	CAMCINT	W1C	0	GPTM CaptureA匹配的原始中断标志 该位表示屏蔽之后CaptureA匹配的中断状态。
0	TATOCINT	W1C	0	GPTM TimerA超时的原始中断标志 TATOCINT 值如下定义： 值 描述 0 中断不受影响。 1 中断被清除。

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。

GPTM TimerA间隔装载 (GPTMTAILR)

偏移量 0x028
类型 R/W, 复位 0x0000.FFFF（16位模式） 和 0xFFFF.FFFF（32位模式）



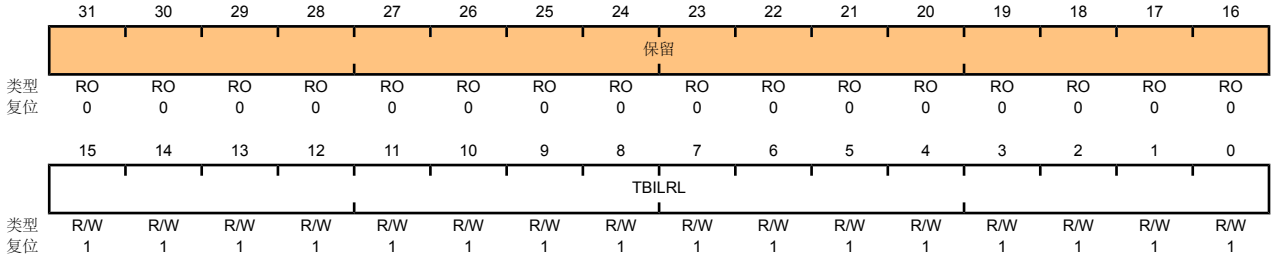
位/域	名称	类型	复位	描述
31:16	TAILRH	R/W	0xFFFF （32位模式） 0x0000 （16位模式）	GPTM TimerA间隔装载寄存器的高字节 当通过**GPTMCFG**寄存器配置为32位模式时，**GPTM TimerB**间隔装载（**GPTMTBILR**）寄存器通过写操作来装载该值。读操作时返回**GPTMTBILR**的当前值。 在16位模式中，该位域在读操作时返回0，不影响**GPTMTBILR**寄存器的状态。
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA间隔装载寄存器的低字节 在16位和32位模式中，对该位域执行写操作将装载TimerA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）寄存器，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，**GPTMTBILR**返回TimerB的当前值，写操作被忽略。

GPTM TimerB间隔装载 (GPTMTBILR)

偏移量 0x02C
类型 R/W, 复位 0x0000.FFFF



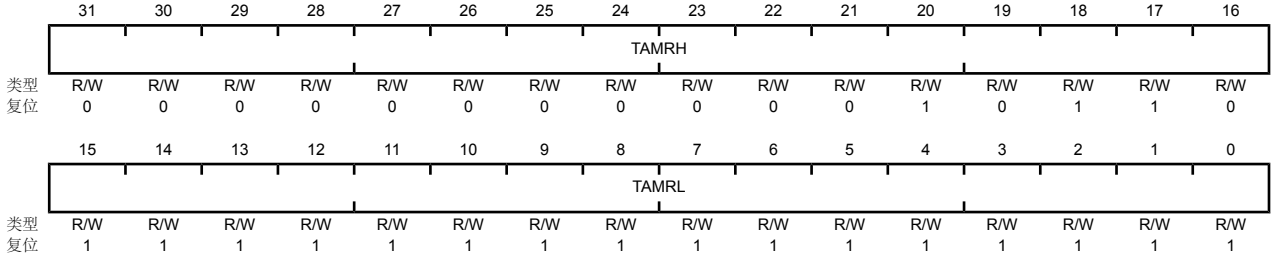
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB间隔装载寄存器 当GPTM没有被配置为32位定时器时，对该位域执行写操作将更新**GPTMTBILR**的值。在32位模式中，写操作被忽略，读操作返回**GPTMTBILR**的当前值。

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

该寄存器用于32位实时时钟模式和16位PWM和输入边沿计数模式。

GPTM TimerA匹配 (GPTMTAMATCHR)

偏移量 0x030
类型 R/W, 复位 0x0000.FFFF（16位模式） 和 0xFFFF.FFFF（32位模式）



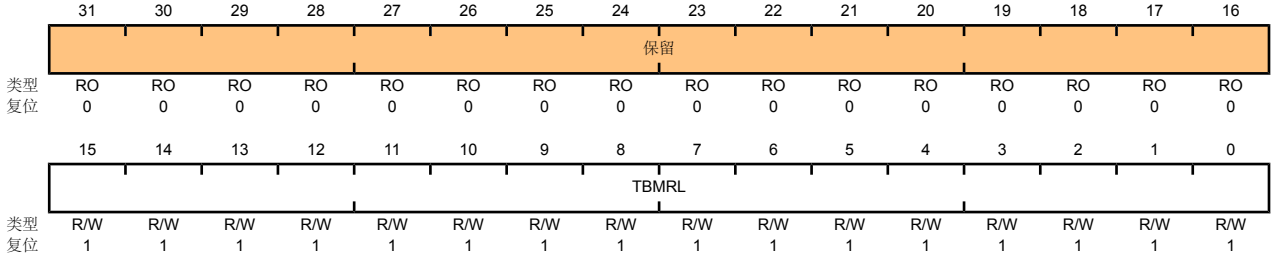
位/域	名称	类型	复位	描述
31:16	TAMRH	R/W	0xFFFF （32位模式） 0x0000 （16位模式）	GPTM TimerA匹配寄存器的高字节 当通过**GPTMCFG**寄存器配置为32位实时时钟（RTC）模式时，该值与**GPTMTAR**的高半字进行比较，来确定匹配事件。 在16位模式中，对该位域的读操作返回0，不影响**GPTMTBMATCHR**寄存器的状态。
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA匹配寄存器的低字节 当通过**GPTMCFG**寄存器配置为32位实时时钟（RTC）模式时，该值与**GPTMTAR**的低半字进行比较，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位实时时钟模式和16位PWM和输入边沿计数模式。

GPTM TimerB匹配 (GPTMTBMATCHR)

偏移量 0x034
类型 R/W, 复位 0x0000.FFFF



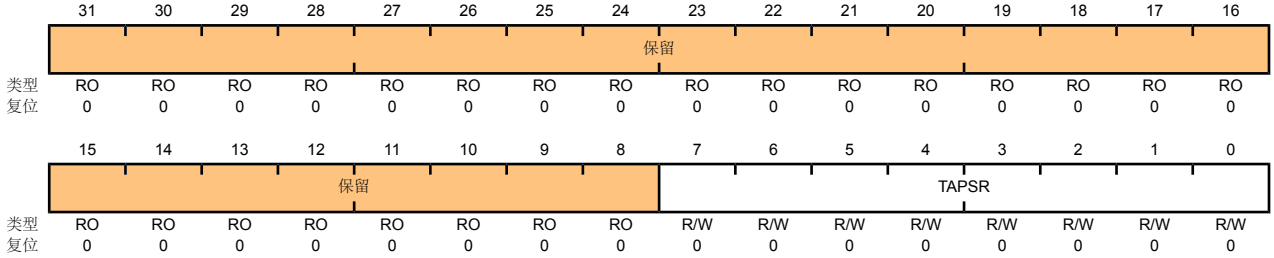
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB匹配寄存器的低字节 配置为PWM模式时，该值与**GPTMTBILR**一起，确定输出PWM信号的占空比。 配置为边沿计数模式时，该值与**GPTMTBILR**一起，确定需计数多少边沿事件数。总的边沿事件数等于**GPTMTBILR**与该值的差。

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

当工作在单次触发或周期模式时，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
类型 R/W, 复位 0x0000.0000



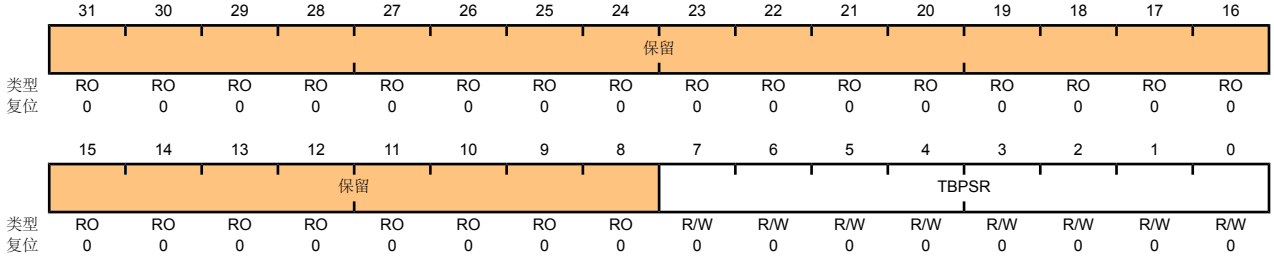
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TAPSR	R/W	0x00	GPTM TimerA 预分频 寄存器通过写操作载入该值。执行读操作将返回寄存器的当前值。 详细信息和实例请参考 表 10-2 在 194页 。

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

当工作在单次触发或周期模式时，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
类型 R/W, 复位 0x0000.0000



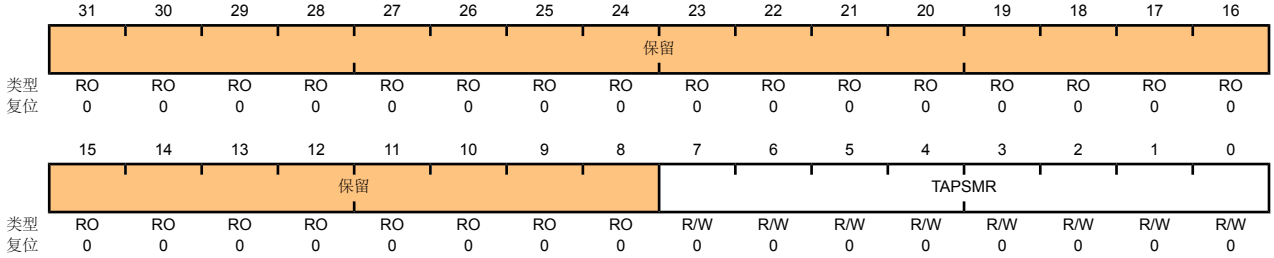
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TBPSR	R/W	0x00	GPTM TimerB预分频 寄存器通过写操作载入该值。执行读操作将返回寄存器的当前值。 详细信息和实例请参考 表 10-2 在 194页 。

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

当工作在16位单次触发或周期模式时，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
类型 R/W, 复位 0x0000.0000



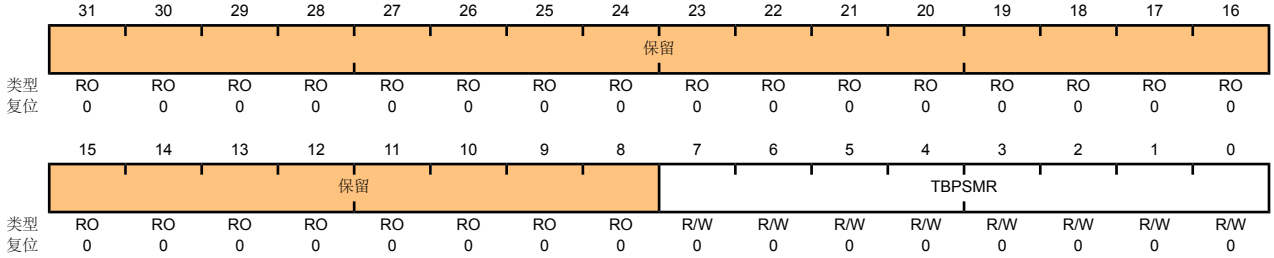
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TAPSMR	R/W	0x00	GPTM TimerA预分频匹配 该值与**GPTMTAMATCHR**一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

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

当工作在16位单次触发或周期模式时，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
类型 R/W, 复位 0x0000.0000



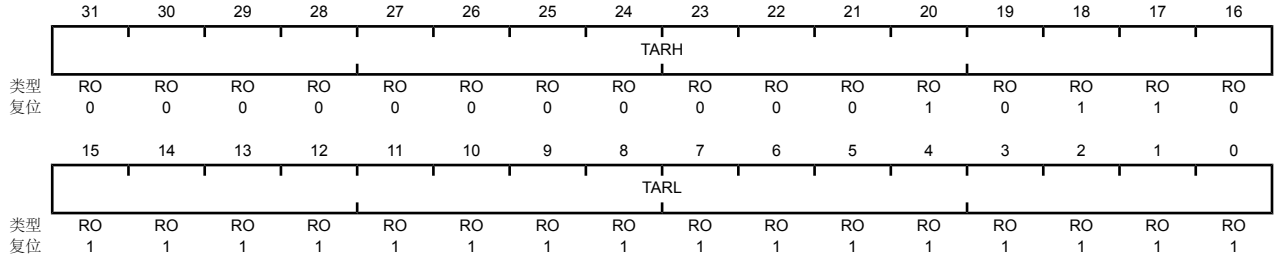
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TBPSMR	R/W	0x00	GPTM TimerB预分频匹配 该值与**GPTMTBMATCHR**一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

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

该寄存器显示了TimerA计数器的当前值，输入边沿计数模式的情况除外。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
类型 RO, 复位 0x0000.FFFF （16位模式） 和 0xFFFF.FFFF（32位模式）



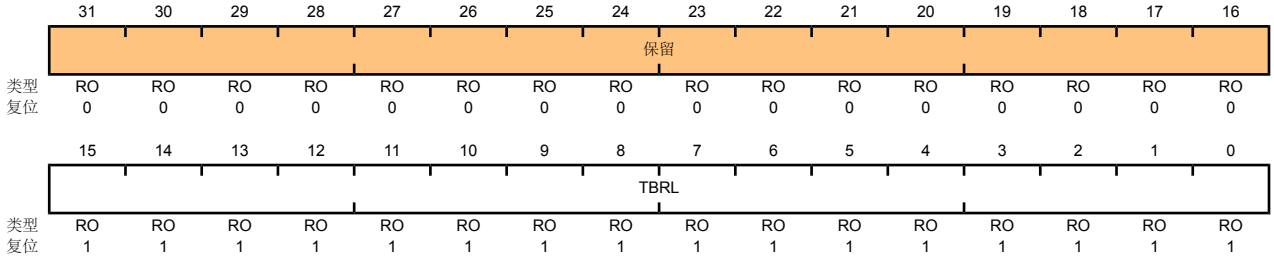
位/域	名称	类型	复位	描述
31:16	TARH	RO	0xFFFF （32位模式） 0x0000 （16位模式）	GPTM TimerA寄存器高字节 如果**GPTMCFG**配置为32位模式，则对该位域执行读操作将获得TimerB的值。如果**GPTMCFG**配置为16位模式，则读操作返回0。
15:0	TARL	RO	0xFFFF	GPTM TimerA寄存器高字节 读取该位域将返回**GPTM TimerA** 计数寄存器的当前值，但输入边沿计数模式除外，在该模式中，读操作返回上一次边沿事件的时间戳（timestamp）。

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

该寄存器显示了TimerB计数器的当前值，输入边沿计数模式的情况除外。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
类型 RO, 复位 0x0000.FFFF



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TBRL	RO	0xFFFF	GPTM TimerB 读取该位域将返回**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）。

11 看门狗定时器

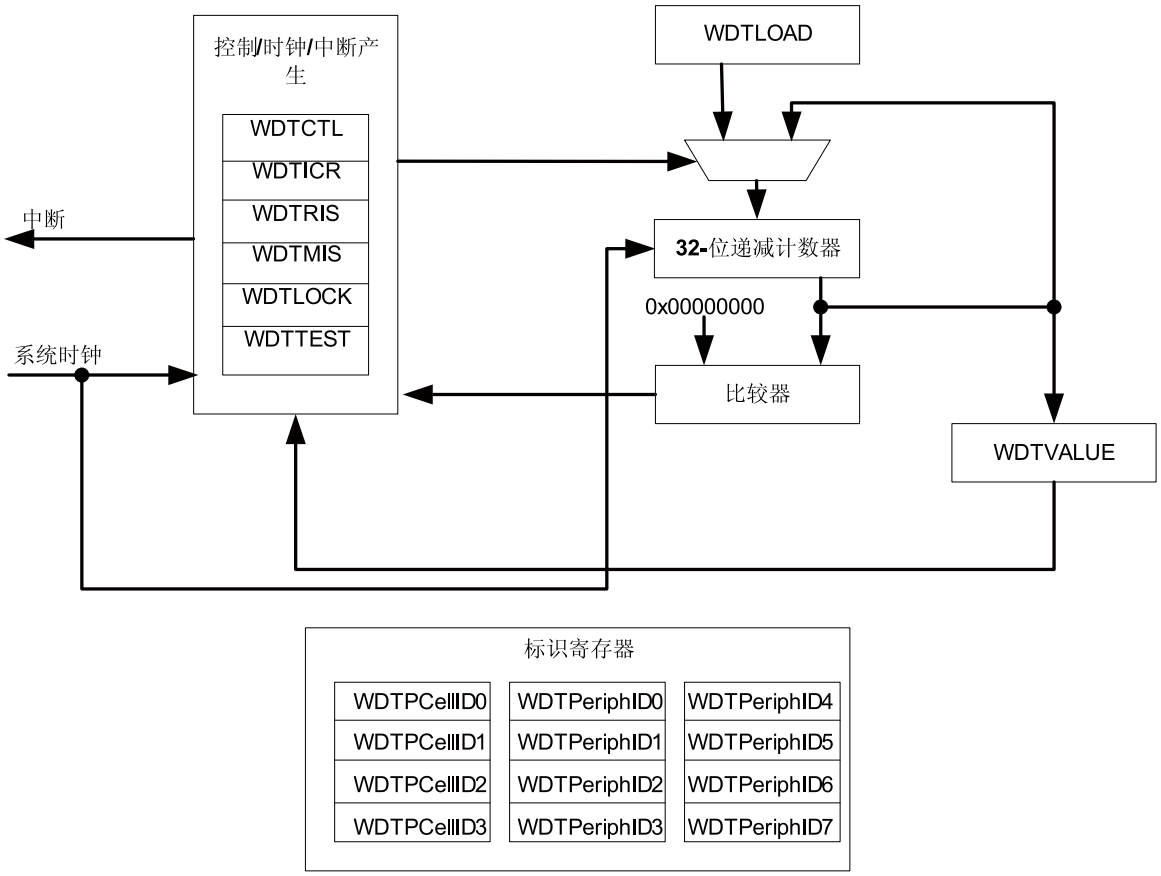
看门狗定时器在到达超时值时可能会产生不可屏蔽的中断（NMIs）或复位。当系统由于软件错误而无法响应或外部器件不是以期望的方式响应时，使用看门狗定时器可重新获得控制权。

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

看门狗定时器可配置为在首次超时(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，就可以通过写锁定寄存器来防止定时器配置被意外更改。

11.1 结构图

图 **11-1. WDT**模块的结构图



11.2 功能描述

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。一旦配置了看门狗定时器，看门狗定时器锁定 **(WDTLOCK)**寄存器被写，以防止定时器配置通过软件意外更改。

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（通过看门狗复位使能功能），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零 **(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。

11.3 初始化和配置

要使用WDT，必须把**RCGC0**寄存器的WDT位置位以使能其外部时钟。按照下列顺序（sequence）配置看门狗定时器：

1. 将所需的定时器值装入 **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0x1ACC.E551。

11.4 寄存器映射

表 11-1 在 227页列出了看门狗定时器。所列偏移量是寄存器相对于看门狗定时器基址0x4000.0000的十六进制增量。

表 **11-1.** 看门狗定时器 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	WDTLOAD	R/W	0xFFFF.FFFF	看门狗装载	229
0x004	WDTVALUE	RO	0xFFFF.FFFF	看门狗值	230
0x008	WDTCTL	R/W	0x0000.0000	看门狗控制	231
0x00C	WDTICR	WO	-	看门狗中断清零	232
0x010	WDTRIS	RO	0x0000.0000	看门狗原始中断状态	233
0x014	WDTMIS	RO	0x0000.0000	看门狗屏蔽后的中断状态	234
0x418	WDTTEST	R/W	0x0000.0000	看门狗测试	235
0xC00	WDTLOCK	R/W	0x0000.0000	看门狗锁定	236
0xFD0	WDTPeriphID4	RO	0x0000.0000	看门狗外设标识 4	237
0xFD4	WDTPeriphID5	RO	0x0000.0000	看门狗外设标识 5	238
0xFD8	WDTPeriphID6	RO	0x0000.0000	看门狗外设标识 6	239
0xFDC	WDTPeriphID7	RO	0x0000.0000	看门狗外设标识 7	240
0xFE0	WDTPeriphID0	RO	0x0000.0005	看门狗外设标识0	241
0xFE4	WDTPeriphID1	RO	0x0000.0018	看门狗外设标识1	242
0xFE8	WDTPeriphID2	RO	0x0000.0018	看门狗外设标识2	243

偏移量	名称	类型	复位	描述	见页面
0xFEC	WDTPeriphID3	RO	0x0000.0001	看门狗外设标识3	244
0xFF0	WDTPCellID0	RO	0x0000.000D	看门狗PrimeCell标识 0	245
0xFF4	WDTPCellID1	RO	0x0000.00F0	看门狗PrimeCell标识1	246
0xFF8	WDTPCellID2	RO	0x0000.0005	看门狗PrimeCell标识2	247
0xFFC	WDTPCellID3	RO	0x0000.00B1	看门狗PrimeCell标识3	248

11.5 寄存器描述

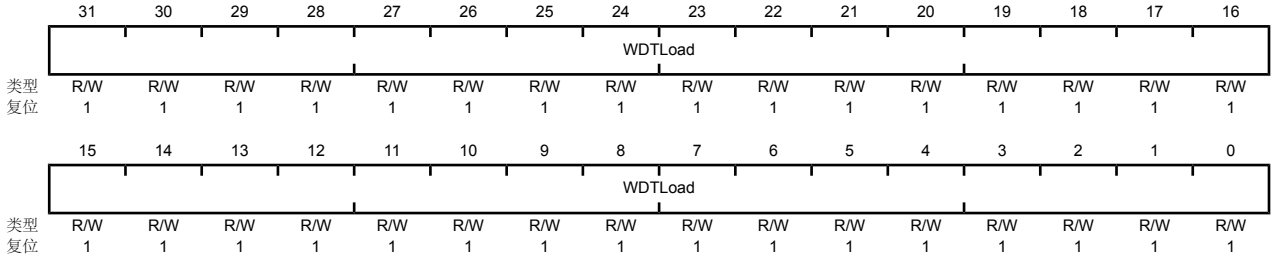
下文将按地址偏移量的数字顺序列出WDT寄存器，并对它们进行描述。

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。在对该寄存器执行写操作时，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。如果将0x0000.0000装载进**WDTLOAD**寄存器，则会立即产生中断。

看门狗装载 (WDTLOAD)

偏移量 0x000
类型 R/W, 复位 0xFFFF.FFFF



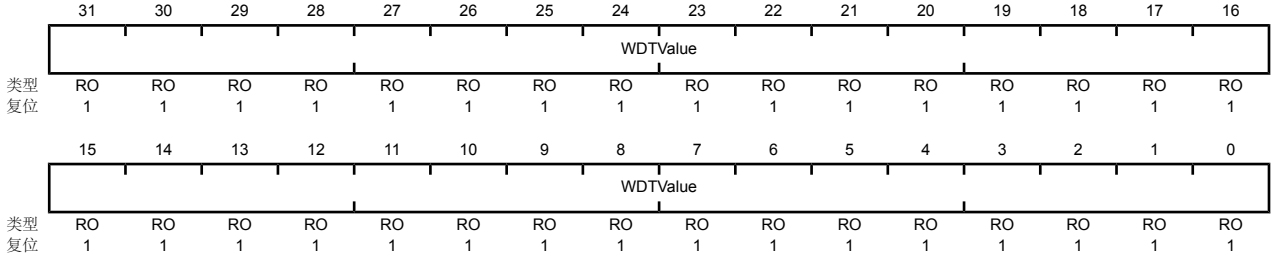
位/域	名称	类型	复位	描述
31:0	WDTLoad	R/W	0xFFFF.FFFF	看门狗装载值

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。

看门狗值 (WDTVALUE)

偏移量 0x004
类型 RO, 复位 0xFFFF.FFFF



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

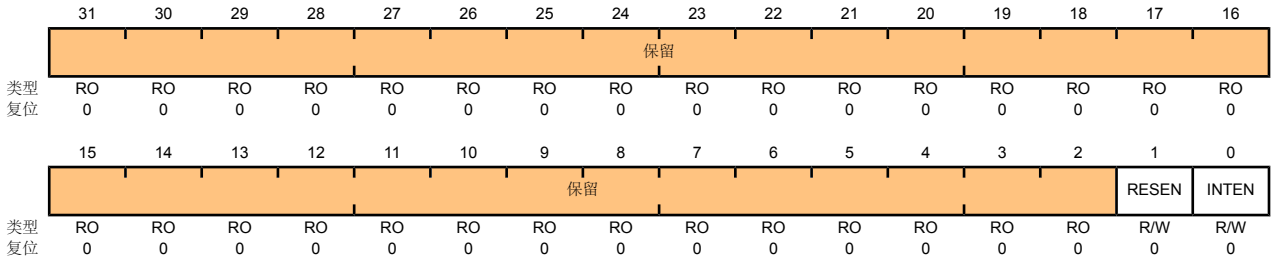
寄存器 3: 看门狗控制（WDTCTL），偏移量 0x008

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。

看门狗控制 (WDTCTL)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



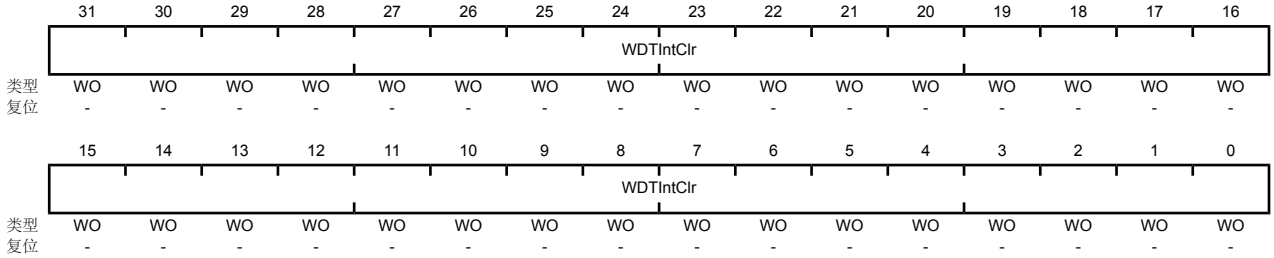
位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	RESEN	R/W	0	看门狗复位使能标志 RESEN值如下定义： 值 描述 0 禁能 1 使能看门狗模块复位输出。
0	INTEN	R/W	0	看门狗中断使能标志 INTEN值如下定义： 值 描述 0 中断事件被禁能（一旦该位被置位，则它仅可通过硬件复位来清除）。 1 中断事件被使能。一旦被使能，之后所有的写操作都会被忽略。

寄存器 **4:** 看门狗中断清零（**WDTICR**），偏移量 **0x00C**

该寄存器是中断清零寄存器。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。

看门狗中断清零 (WDTICR)

偏移量 0x00C
类型 WO, 复位 -



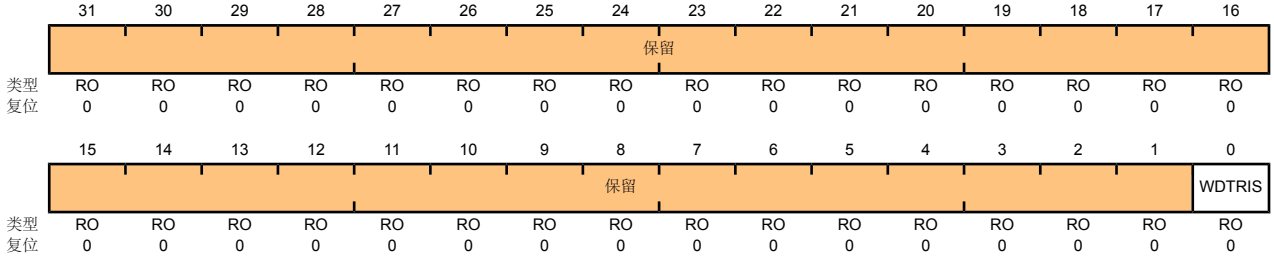
位/域	名称	类型	复位	描述
31:0	WDTIntClr	WO	-	清除看门狗中断。

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
类型 RO, 复位 0x0000.0000



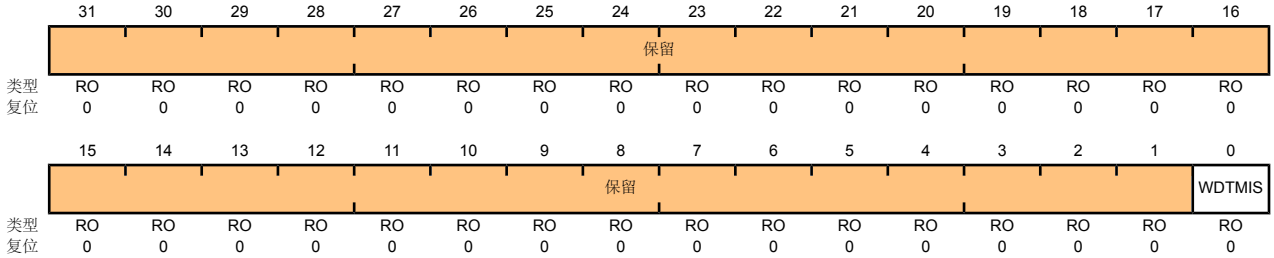
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	WDTRIS	RO	0	看门狗原始中断状态 给出**WDTINTR**的原始中断状态（在屏蔽之前）。.

寄存器 6: 看门狗屏蔽后的中断状态（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。

看门狗屏蔽后的中断状态 (WDTMIS)

偏移量 0x014
类型 RO, 复位 0x0000.0000



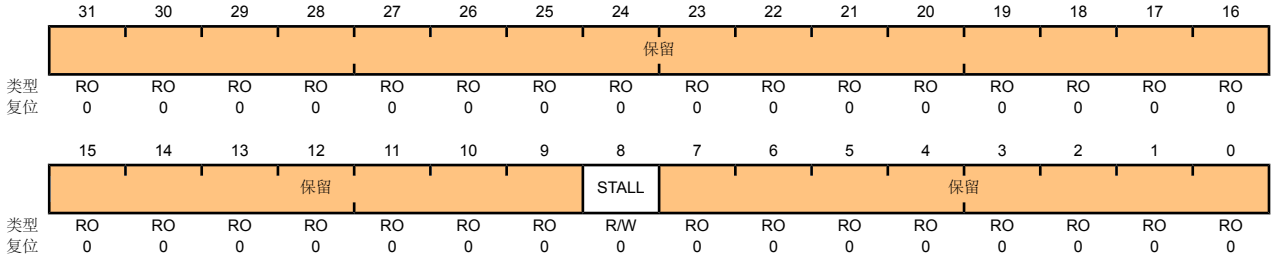
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	WDTMIS	RO	0	看门狗屏蔽后的中断状态 给出**WDTINTR**中断在屏蔽后的中断状态。

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
类型 R/W, 复位 0x0000.0000



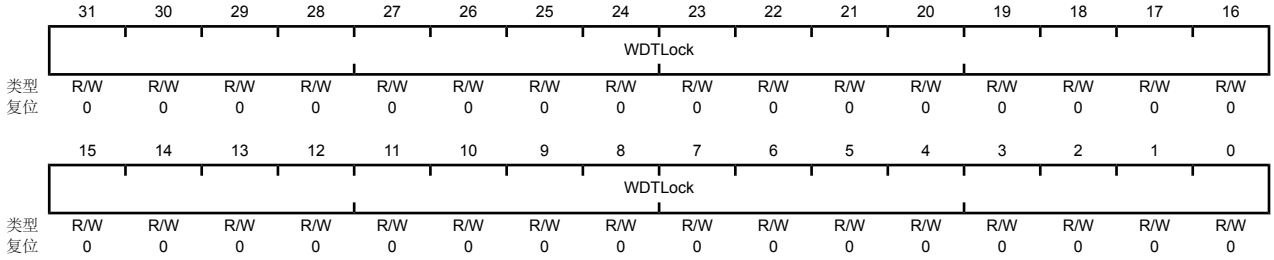
位/域	名称	类型	复位	描述
31:9	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
8	STALL	R/W	0	看门狗停止使能标志 当设为1时，如果调试器使 Stellaris®微控制器停止，则看门狗定时器也会停止计数。而一旦微控制器重新启动，则看门狗定时器也会恢复计数。
7:0	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 **8:** 看门狗锁定（**WDTLOCK**），偏移量 **0xC00**

向**WDTLOCK**寄存器写入 0x1ACC.E551允许对其它所有寄存器执行写操作。在**WDTLOCK**寄存器中写入其它任意值，寄存器再次不能对所有其它寄存器执行写操作。读取**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0x0000.0001(这是在已锁定的情况下；否则，返回的值为0x0000.0000(未锁定))。

看门狗锁定 (WDTLOCK)

偏移量 0xC00
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:0	WDTLock	R/W	0x0000	看门狗锁定

写入值0x1ACC.E551 允许看门狗寄存器执行写操作。写入其它值则重新进行锁定，防止对任意寄存器进行更新。

读取该寄存器时返回以下值：

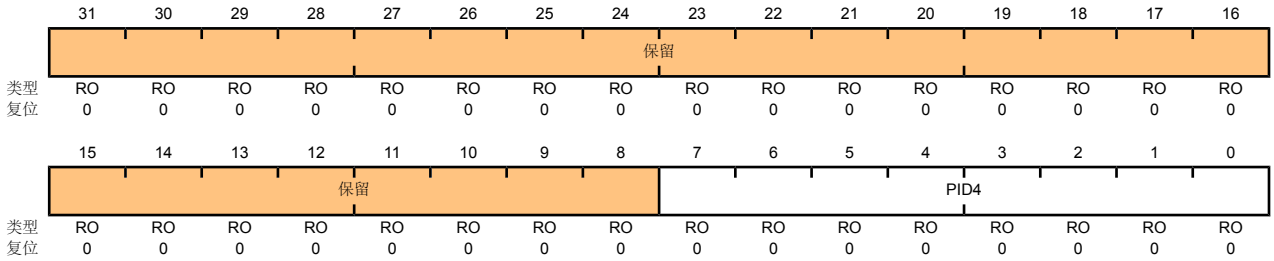
值	描述
0x0000.0001	锁定
0x0000.0000	未锁定

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），寄存器内的位域决定了复位值。

看门狗外设标识 4 (WDTPeriphID4)

偏移量 0xFD0
类型 RO, 复位 0x0000.0000



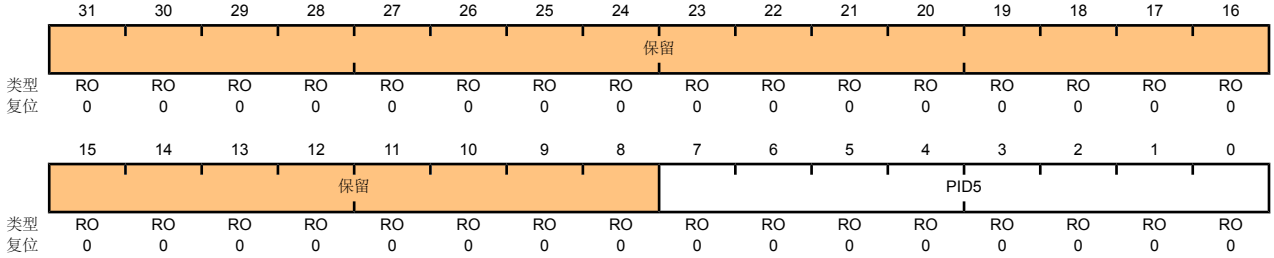
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	WDT 外设ID寄存器[7:0]

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），寄存器内的位域决定了复位值。

看门狗外设标识 5 (WDTPeriphID5)

偏移量 0xFD4
类型 RO, 复位 0x0000.0000



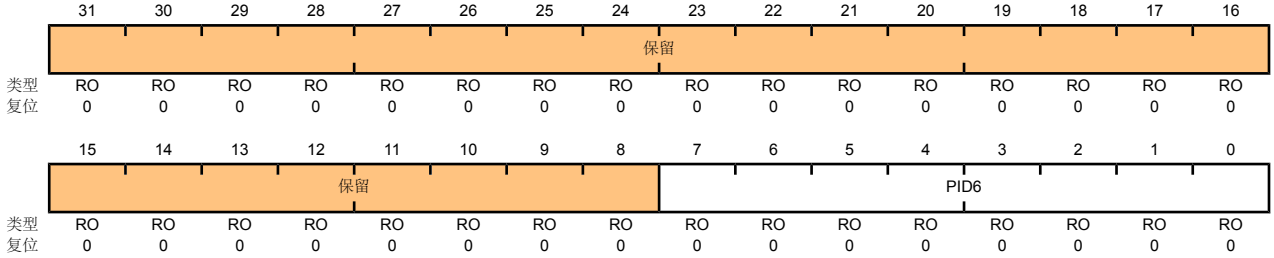
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	WDT外设ID寄存器[15:8]

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），寄存器内的位域决定了复位值。

看门狗外设标识 6 (WDTPeriphID6)

偏移量 0xFD8
类型 RO, 复位 0x0000.0000



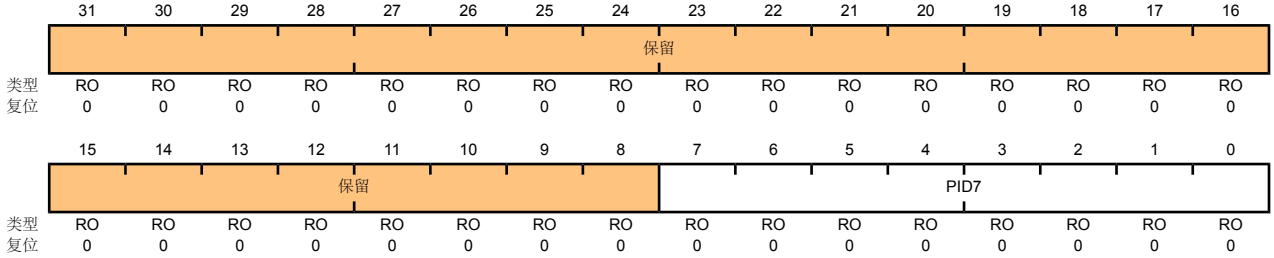
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	WDT外设ID寄存器[23:16]

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），寄存器内的位域决定了复位值。

看门狗外设标识 7 (WDTPeriphID7)

偏移量 0xFDC
类型 RO, 复位 0x0000.0000



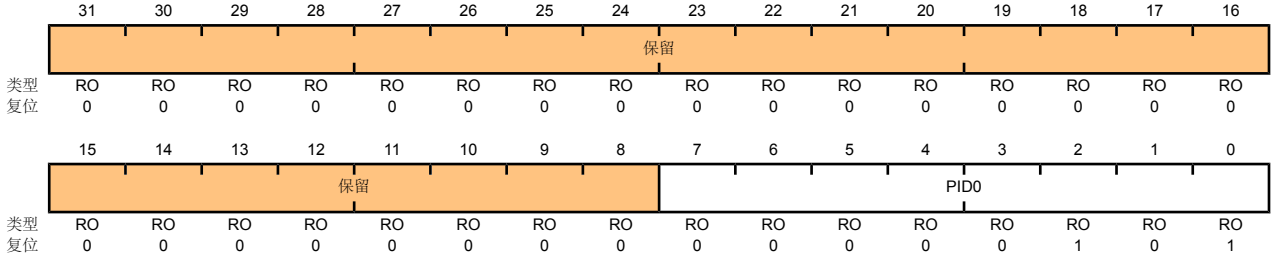
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	WDT外设ID寄存器[31:24]

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），寄存器内的位域决定了复位值。

看门狗外设标识0 (WDTPeriphID0)

偏移量 0xFE0
类型 RO, 复位 0x0000.0005



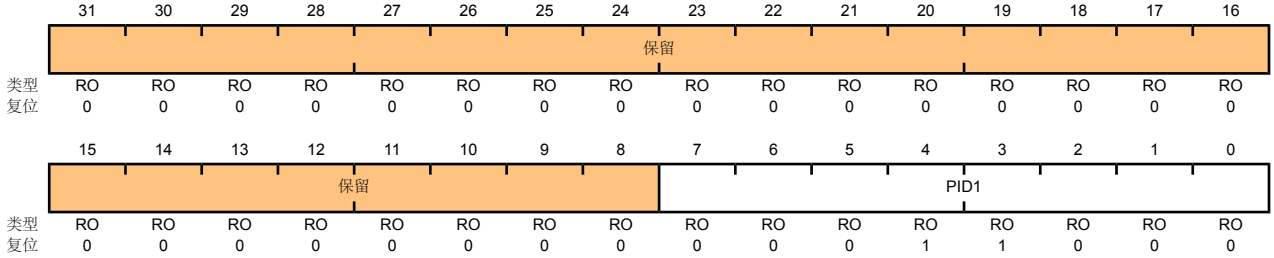
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x05	看门狗标识ID寄存器[7:0]

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），寄存器内的位域决定了复位值。

看门狗外设标识1 (WDTPeriphID1)

偏移量 0xFE4
类型 RO, 复位 0x0000.0018



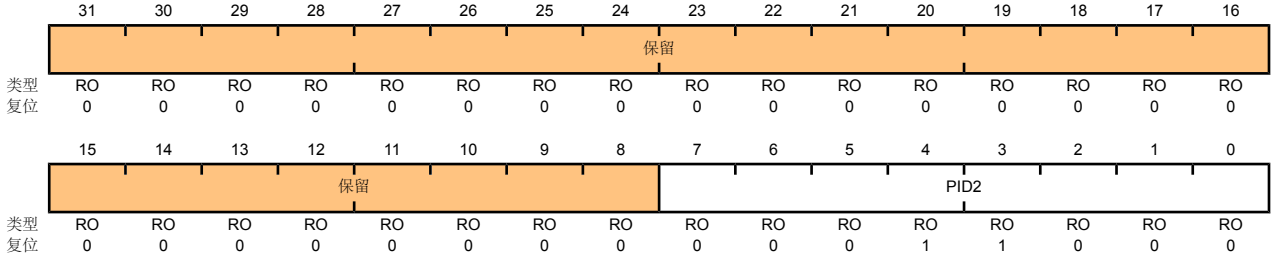
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x18	看门狗外设ID寄存器[15:8]

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），寄存器内的位域决定了复位值。

看门狗外设标识2 (WDTPeriphID2)

偏移量 0xFE8
类型 RO, 复位 0x0000.0018



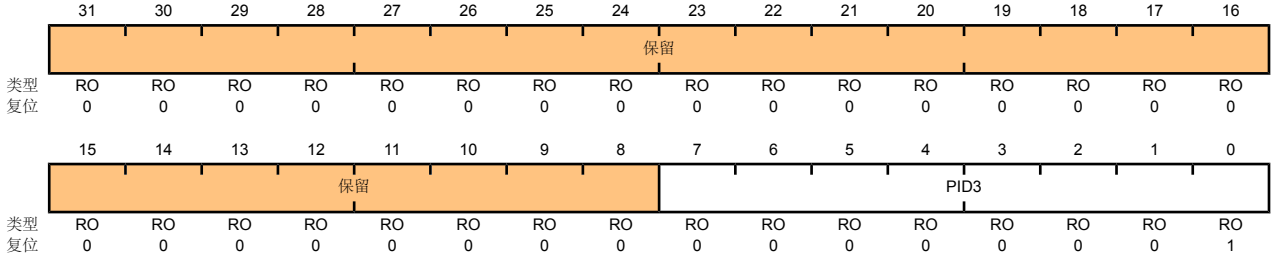
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	看门狗外设ID寄存器[23:16]

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），寄存器内的位域决定了复位值。

看门狗外设标识3 (WDTPeriphID3)

偏移量 0xFEC
类型 RO, 复位 0x0000.0001



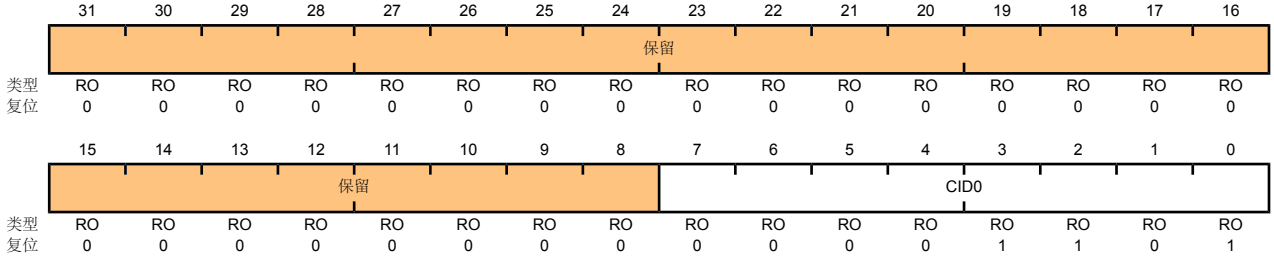
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	看门狗外设ID寄存器[31:24]

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），寄存器内的位域决定了复位值。

看门狗PrimeCell标识 0 (WDTPCellID0)

偏移量 0xFF0
类型 RO, 复位 0x0000.000D



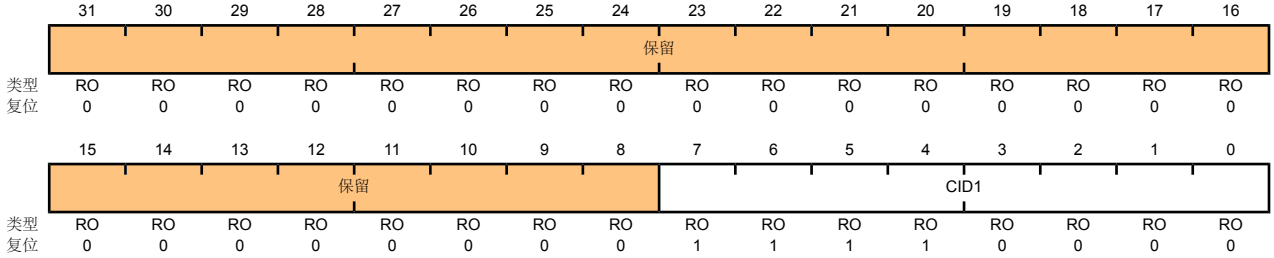
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	看门狗PrimeCell ID寄存器[7:0]

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），寄存器内的位域决定了复位值。

看门狗PrimeCell标识1 (WDTPCellID1)

偏移量 0xFF4
类型 RO, 复位 0x0000.00F0



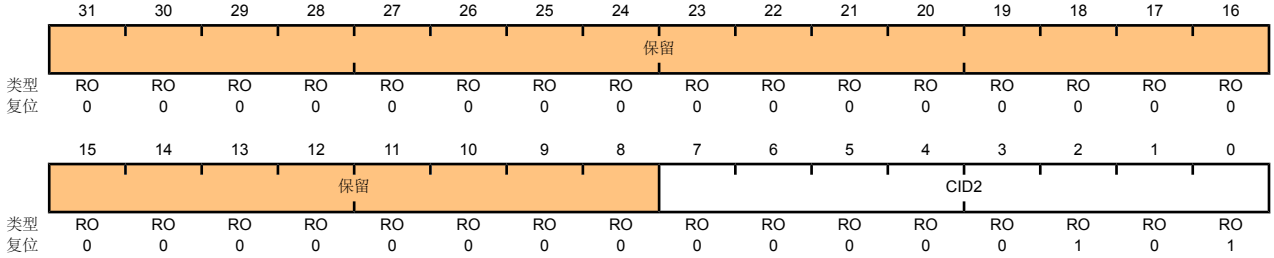
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	看门狗PrimeCell ID寄存器[15:8]

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），寄存器内的位域决定了复位值。

看门狗PrimeCell标识2 (WDTPCellID2)

偏移量 0xFF8
类型 RO, 复位 0x0000.0005

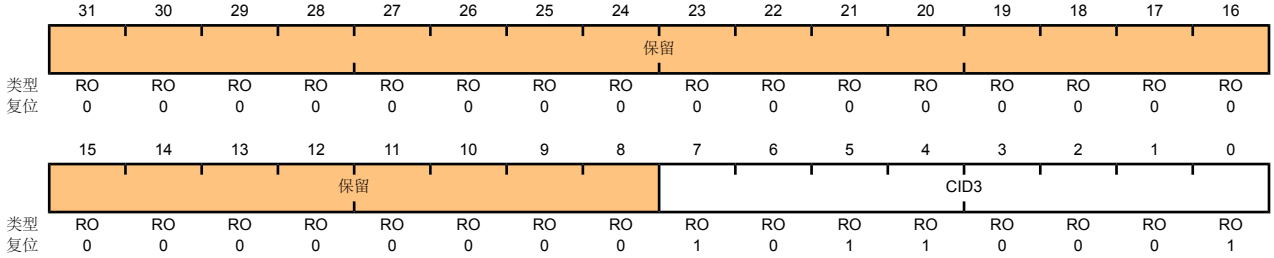


位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	看门狗PrimeCell ID寄存器[23:16]

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），寄存器内的位域决定了复位值。

看门狗PrimeCell标识3 (WDTPCellID3)
偏移量 0xFFC
类型 RO, 复位 0x0000.00B1



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID3	RO	0xB1	看门狗PrimeCell ID寄存器[31:24]

12 通用异步收发器 (UART)

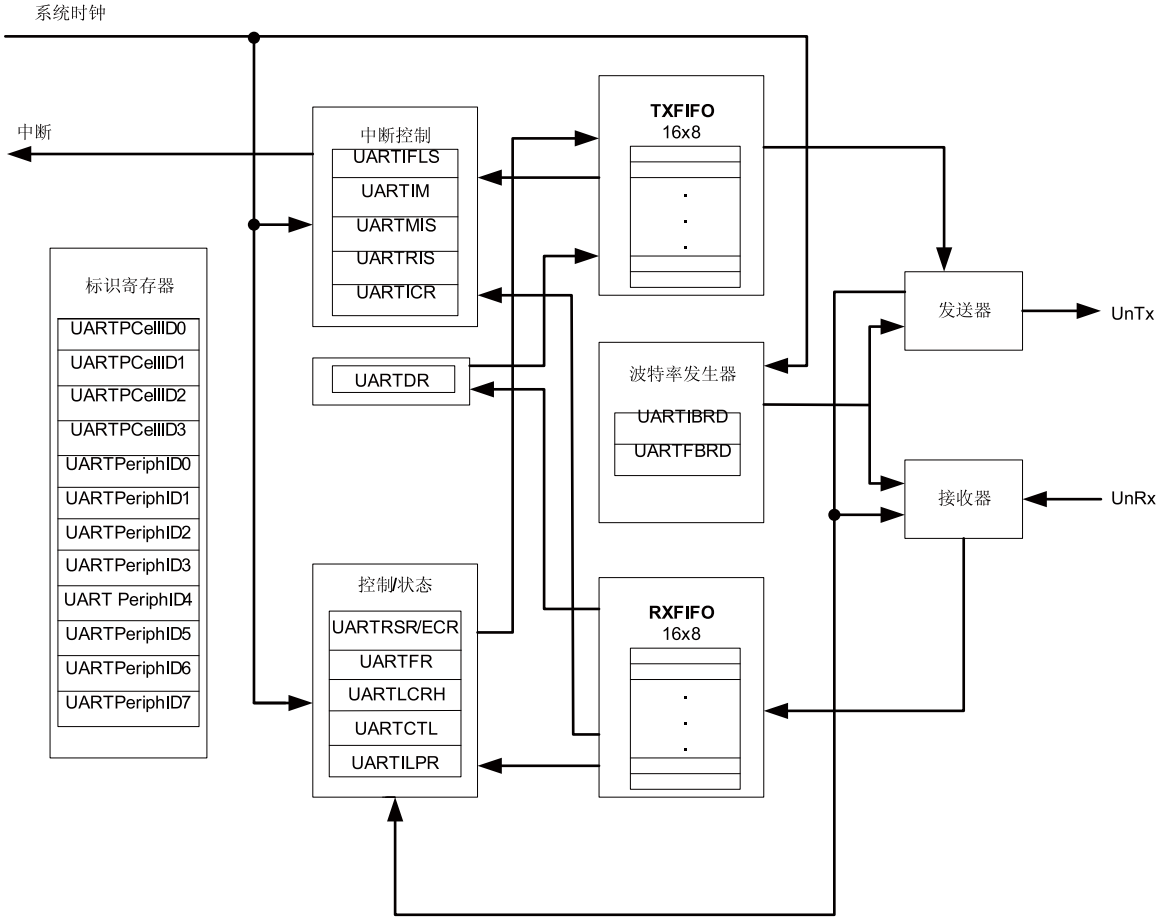
Stellaris® 通用异步收发器 (UART) 具有完全可编程、16C550型串行接口的特性。LM3S2950控制器带有 3个 UART模块s。

每个 UART 具有以下特性：

- 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触发深度可为：1/8、 1/4、 1/2、 3/4或 7/8
- 可编程的波特率发生器允许速率高达3.125 Mbps
- 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
- IrDA 串行红外 (SIR) 编码器/解码器具有以下特性：
  - 用户可以根据需要对 IrDA串行红外 (SIR) 或 UART 输入/输出端进行编程
  - IrDA SIR 编码器/解码器功能模块在半双工时其数据速率可高达115.2 Kbps
  - 位持续时间（bit duration）为3/16（正常） 和1.41-2.23 µs (低功耗)
  - 可编程的内部时钟发生器，允许对参考时钟进行1到256分频以得到低功耗模式的位持续时间。

12.1 结构图

图 **12-1. UART**模块的结构图



12.2 功能描述

每个 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 (见 266页)的TXE位和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。

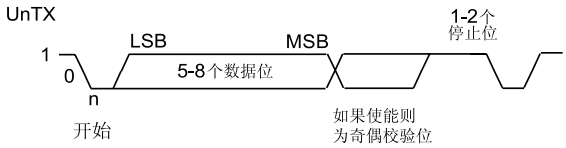
UART外设还包含一个串行红外（SIR）编码器/解码器模块，这个模块可以通过与一个红外收发器连接来实现IrDA SIR物理层规范。通过UARTCTL寄存器对SIR功能进行编程。

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 251页 。

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 (见 262页) 进行加载，而6-位小数则通过**UART**小数波特率除数 **(UARTFBRD)** 寄存器 (见 263页)进行加载。波特率除数（BRD）和系统时钟具有以下关系 (其中 ***BRDI***是BRD的整数部分， ***BRDF*** 是小数部分，之间用一个小数位隔开):

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

以下等式是6位小数 (被加载到**UARTFBRD**寄存器中的 DIVFRAC 位域) 的计算方法，即——波特率除数的小数部分乘以64，然后为了消除四舍五入误差再加上0.5。

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

UART产生一个16倍于波特率的内部波特率参考时钟（称作Baud16）。这个参考时钟经过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。

UARTIBRD和 **UARTFBRD** 寄存器连同**UART**线控,高字节 **(UARTLCRH)** 寄存器 (见 264页)一起共同组成一个内部30位寄存器。这个内部寄存器仅在对**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-字节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 (见 260页)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（UnRx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” 在 250页）。

如果UnRx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 (见 258页)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UnRx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。

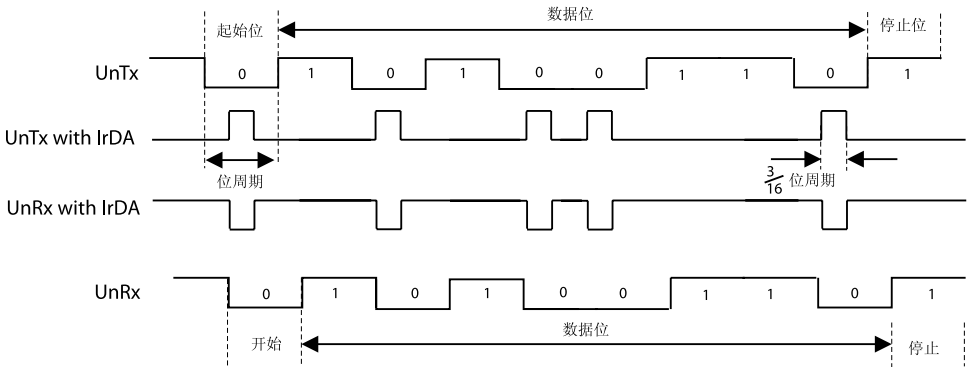
12.2.4 串行红外 **(SIR)**协议

UART还包含一个IrDA串行红外（SIR）编码器/解码器模块。IrDA SIR模块的功能是在异步UART数据流和半双工串行SIR接口之间进行转换。片上不会执行任何模拟处理操作。SIR模块的任务就是要给UART提供一个数字编码输出和一个解码输入。UART信号管脚可以与一个红外收发器连接以实现IrDA SIR 物理层链接。SIR模块具有两种工作模式：

- 在正常的IrDA模式中， 输出管脚上的逻辑0电平被当作3/16所选波特率位周期的高脉冲发送，而逻辑1电平被当作静态低信号发送。这些电平控制红外发送器的驱动器，为每个0发送光脉冲。在接收端，接收到的光脉冲给接收器的光敏晶体管基极加电，将其输出拉至低电平。并将UART输入管脚变为低电平。
- 在低功耗IrDA模式中，通过改变**UARTCR**中的相应位，可以将发射的红外脉冲的宽度设置为内部产生的 IrLPBaud16信号周期的3倍 (1.63 µs，假定额定频率为1.8432 MHz) 。

图 12-3 在 252页给出了含有IrDA调制和不含IrDA调制时的UART发送和接收信号

图 **12-3. IrDA**数据调制



在正常IrDA模式和低功耗IrDA模式下：

- 在发送过程中，UART数据位是编码的基础；
- 在接收过程中，译码位被传输到UART接收逻辑电路。

IrDA SIR 物理层指定了一个半双工通信链接，发送和接收之间的延迟最小为10ms。这个延迟可以由软件产生，因为UART不会自动提供。之所以需要这个延迟，是因为红外接收器电子设备可能会出现偏移，有时从相邻的发送器LED耦合而产生的光强甚至会将它变饱和。这个延迟称做等待时间，或接收器建立时间。

12.2.5 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(见 256页)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**(264页)中的FEN位可以使FIFO使能。

FIFO的状态可以通过 **UART**标志 **(UARTFR)**寄存器 (见 260页) 和 **UART**接收状态**(UARTRSR)** 寄存器进行监控。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0E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 (见 268页)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。

12.2.6 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）
- 奇偶校验错误（Parity Error）
- 帧错误
- 接收超时
- 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(见 272页)，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 (见 269页) 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 (见 271页)便可查询到该状态。

只需把**UART**中断清除**(UARTICR)** 寄存器 (见 273页)中相应的位置位，便可以清除中断（**UARTMIS**和**UARTRIS**寄存器的中断）。

当接收FIFO不为空时接收超时中断有效，超过32-位周期不接收更多的数据。当FIFO通过读所有数据（或通过读保存寄存器）变为空时，或当写1到**UARTICR**寄存器的相应位时，接收超时中断被清除。

12.2.7 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(见 266页)中的LBE位来实现的。在回送模式下，UnTx上发送的数据将被UnRx输入端接收。

12.2.8 IrDA SIR模块

IrDA SIR模块包含一个IrDA串行红外 (SIR) 协议编码/解码器。使能时， SIR模块将 UnTx 和 UnRx 管脚用于SIR协议，这两个管脚应该与IR收发器连接。

SIR模块可以接收和发送，但是只能以半双工方式进行红外通信，所以它不能同时接收和发送。发送必须在可以接收数据前停止。IrDA SIR物理层规定发送和接收之间的最小延迟为10ms。

12.3 初始化和配置

要使用UART，必须使能外设时钟，这可以通过将**RCGC1**寄存器中的UART0、 UART1或 UART2 位置位来实现。

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：

- 波特率115200
- 数据长度8位
- 1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" 在 251页”中描述的等式计算得到：

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

即**UARTIBRD** 寄存器(见 262页) 的DIVINT位域应该设为10。加载到**UARTFBRD** 寄存器(见 263页) 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0x0000.0060）。
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 在 255页列出了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: 0x4000.C000

- UART1: 0x4000.D000
- UART2: 0x4000.E000

注意: 在重新对任意控制寄存器进行编程以前，必须将UART禁能（见 266页中**UARTCTL** 寄存器对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	类型	复位	描述	见页面
0x000	UARTDR	R/W	0x0000.0000	UART数据	256
0x004	UARTRSR/ UARTECR	R/W	0x0000.0000	UART接收状态/错误清除	258
0x018	UARTFR	RO	0x0000.0090	UART标志	260
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA 低功耗寄存器	261
0x024	UARTIBRD	R/W	0x0000.0000	UART整数波特率除数	262
0x028	UARTFBRD	R/W	0x0000.0000	UART小数波特率除数	263
0x02C	UARTLCRH	R/W	0x0000.0000	UART线控	264
0x030	UARTCTL	R/W	0x0000.0300	UART控制	266
0x034	UARTIFLS	R/W	0x0000.0012	UART中断的FIFO深度选择	268
0x038	UARTIM	R/W	0x0000.0000	UART中断屏蔽	269
0x03C	UARTRIS	RO	0x0000.000F	UART原始中断状态	271
0x040	UARTMIS	RO	0x0000.0000	UART屏蔽后的中断状态	272
0x044	UARTICR	W1C	0x0000.0000	UART中断清除	273
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART外设标识4	275
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART外设标识5	276
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART外设标识6	277
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART外设标识7	278
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART外设标识0	279
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART外设标识1	280
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART外设标识2	281
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART外设标识3	282
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell 标识0	283
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell标识1	284
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell标识2	285
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell标识3	286

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），偏移量 0x000

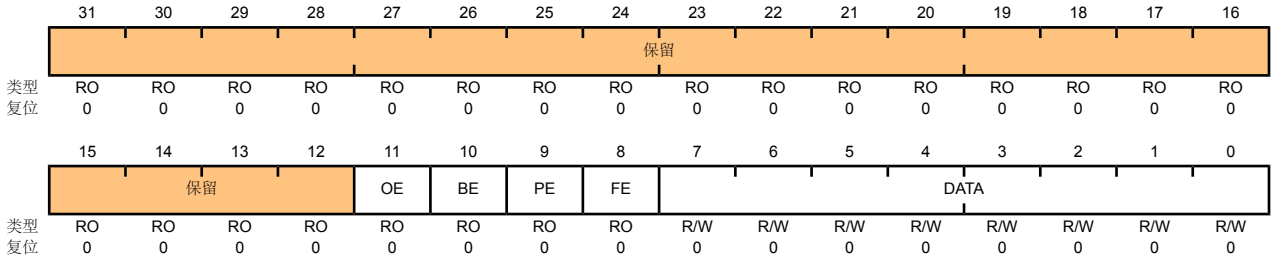
该寄存器是数据寄存器（FIFO的接口）。

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)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:12	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11	OE	RO	0	UART溢出错误标志 0E值如下定义： 值 描述 0 没有数据由于FIFO溢出而丢失。 1 当FIFO满时接收到新数据，导致数据丢失。
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。

位/域	名称	类型	复位	描述
7:0	DATA	R/W	0	Data发送或接收 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**），偏移量 **0x004**

UARTRSR/UARTECR 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。如果从该寄存器读取状态，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。

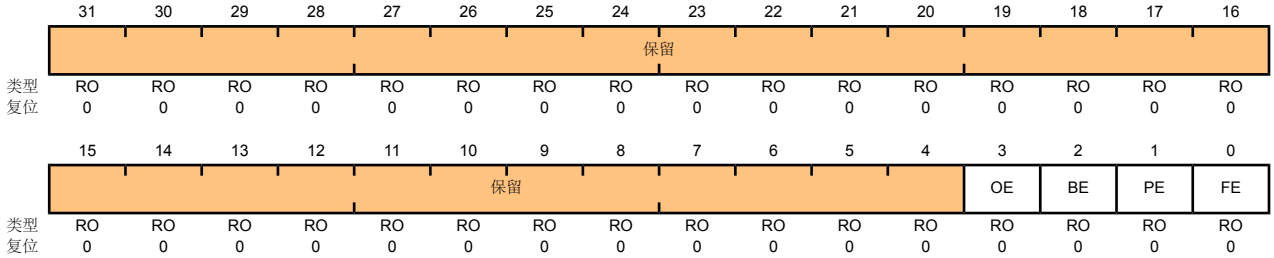
UARTRSR寄存器不能被写。

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。

只读接收状态（**UARTRSR**） 寄存器

UART接收状态/错误清除 (UARTRSR/ UARTECR)

偏移量 0x004
类型 RO, 复位 0x0000.0000



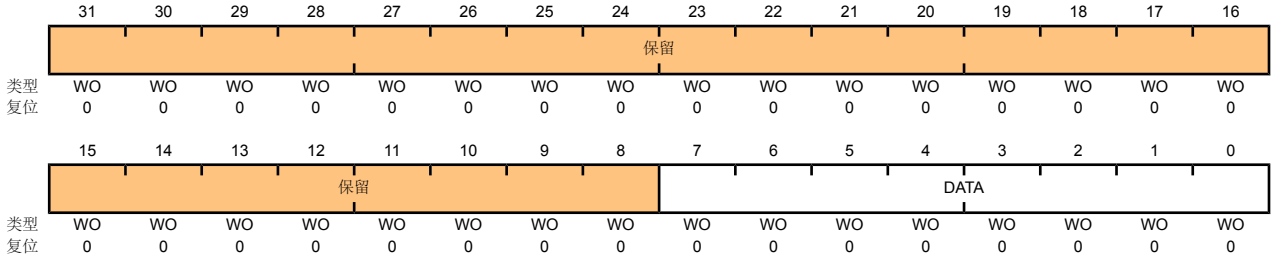
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	OE	RO	0	UART溢出错误标志 当FIFO满且接收到新的数据时该位被设为1。对**UARTECR**进行写操作会将该位清零。 由于在FIFO满时不再有数据写入，所以FIFO内容将保持有效，只有移位寄存器的内容会被覆盖。此时，CPU必须读取数据以便将FIFO清空。
2	BE	RO	0	UART中止错误（Break Error） 在检测到中止的情况时该位会被设为1，表示接收数据输入在长于一个完整字的传输时间（定义为起始位、数据位、奇偶校验位和停止位）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（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。 对**UARTECR**进行写操作会将该位清零。

位/域	名称	类型	复位	描述
0	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。 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

UART接收状态/错误清除 (UARTRSR/ UARTECR)

偏移量 0x004
类型 WO, 复位 0x0000.0000



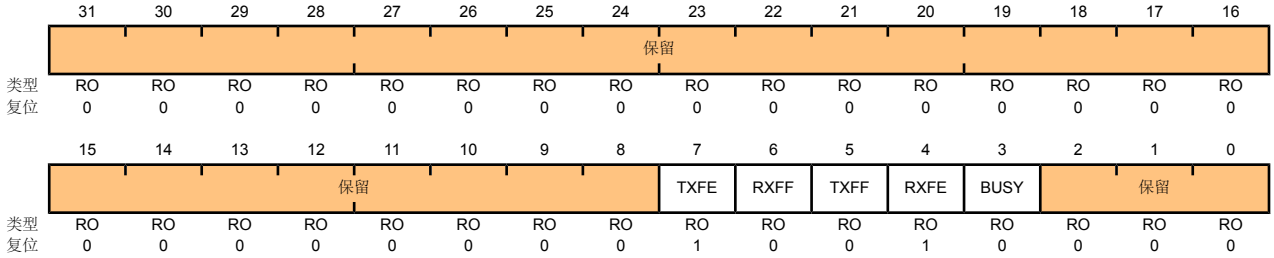
位/域	名称	类型	复位	描述
31:8	保留	WO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	WO	0	错误清除标志 将任意数据写入该寄存器会将帧、奇偶校验、中止和溢出标志清零。

寄存器 **3: UART**标志（**UARTFR**），偏移量 **0x018**

UARTFR寄存器是标志寄存器。复位后，TXFF、RXFF和BUSY位均为0，TXFE和RXFE位均为1。

UART标志 (UARTFR)

偏移量 0x018
类型 RO, 复位 0x0000.0090



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
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忙标志 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	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 4: UART IrDA 低功耗寄存器（UARTILPR），偏移量 0x020

UARTILPR 寄存器是一个8位读/写寄存器，它存放了低功耗计数器分频值，这个值用来产生 IrLPBaud16信号，这可以通过分频系统时钟(SysClk)来实现。复位时，所有位都变为0。

根据写入**UARTILPR**中的低功耗分频值对UARTCLK信号进行分频，因此产生 IrLPBaud16 内部信号。低功耗分频值根据以下等式计算等到：

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

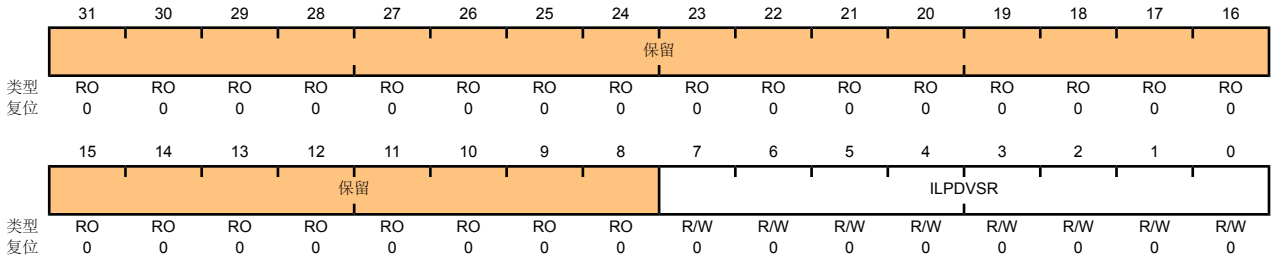
此处， $F_{IrLPBaud16}$ 额定值为 1.8432 MHz。

当使用低功耗模式时，IrLPBaud16 是指用来产生SIR脉冲的内部信号。你必须旋转分频值，因此 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz，这样低功耗脉冲的持续时间就可以为1.41–2.11 µs（IrLPBaud16周期的3倍）。IrLPBaud16的最小频率可以保证会丢弃小于IrLPBaud16一个周期的脉冲，而把大于1.4µs的脉冲当作有效脉冲接收。

注意： 0是非法值。如果编程为0，将不会产生 IrLPBaud16 脉冲。

UART IrDA 低功耗寄存器 (UARTILPR)

偏移量 0x020
类型 R/W, 复位 0x0000.0000



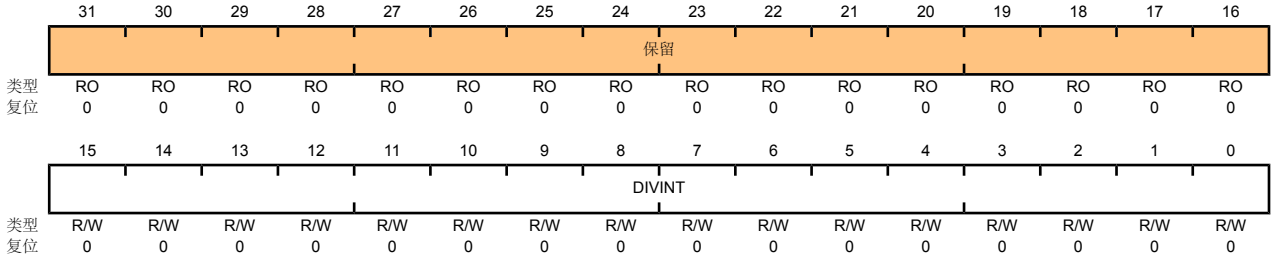
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读–修改–写操作过程中应当保持不变。
7:0	ILPDVSR	R/W	0x00	IrDA低功耗除数 这是一个8-位低功耗除数。

寄存器 **5: UART**整数波特率除数（**UARTIBRD**），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。可实现的最小分频比为1（当**UARTIBRD=0**时），在此情况下，**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” 在 251页。

UART整数波特率除数 (UARTIBRD)

偏移量 0x024
类型 R/W, 复位 0x0000.0000



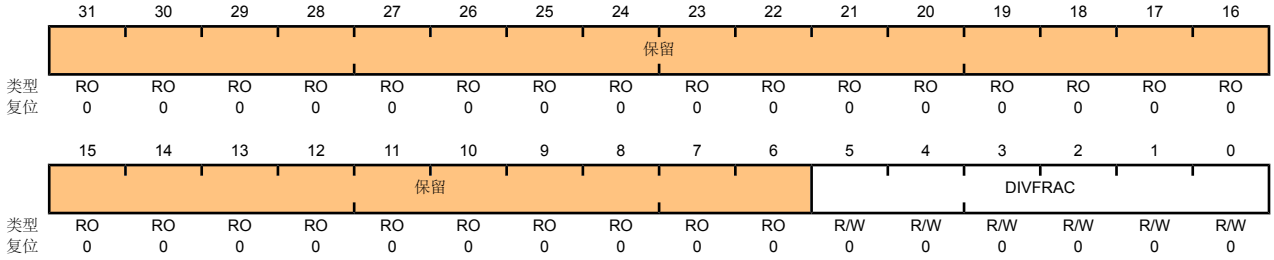
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	DIVINT	R/W	0x0000	整数波特率除数

寄存器 6: UART小数波特率除数（UARTFBRD），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” 在 251页。

UART小数波特率除数 (UARTFBRD)

偏移量 0x028
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5:0	DIVFRAC	R/W	0x000	小数波特率除数

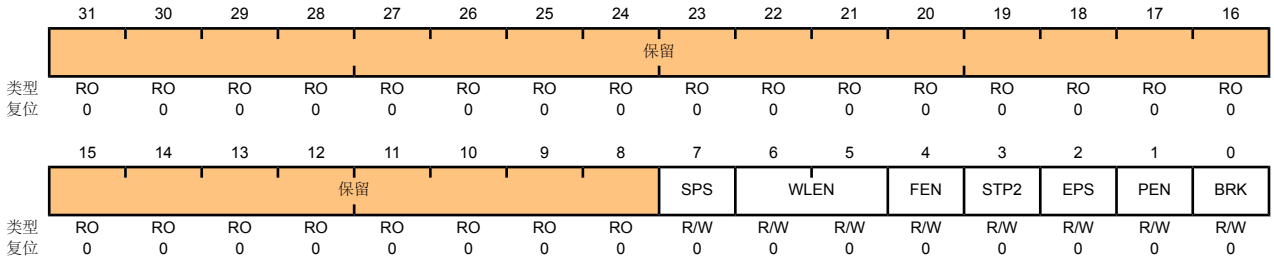
寄存器 **7: UART**线控（**UARTLCRH**），偏移量 **0x02C**

UARTLCRH寄存器是线控寄存器。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 (UARTLCRH)

偏移量 0x02C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	SPS	R/W	0	UART粘着奇偶校验选择 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，粘着奇偶校验被禁能。
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： 值 描述 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。

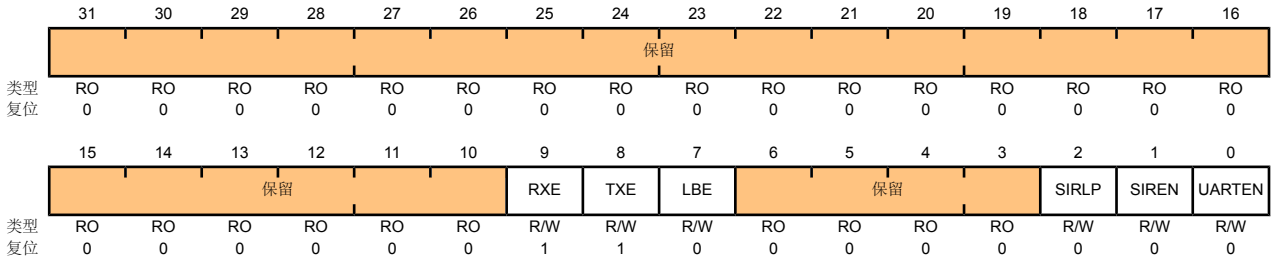
位/域	名称	类型	复位	描述
2	EPS	R/W	0	UART偶校验（even parity）选择 如果该位设为1，那么偶校验的产生和检测都在发送和接收过程中进行，检测数据位加奇偶校验位“1”的位数是否为偶数。 清零时，执行奇校验，检查“1”的位数是否为奇数。 当奇偶检验被PEN位禁止时，该位无效。
1	PEN	R/W	0	UART奇偶校验使能 如果该位设为1，那么奇偶校验及其产生都使能；否则，奇偶校验被禁能，且数据帧中不会增加奇偶校验位。
0	BRK	R/W	0	UART发送中止（break） 如果该位设为1，在完成当前字符的发送后，UnTX输出端上会连续的输出低电平。在正确执行中止（break）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 8: UART控制（UARTCTL），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）和接受使能（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)
偏移量 0x030
类型 R/W, 复位 0x0000.0300



位/域	名称	类型	复位	描述
31:10	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
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。 注意: 要使能接收，还必须将UARTEN位置位。
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。 注意: 要使能发送，还必须将UARTEN 位置位。
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。
6:3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	SIRLP	R/W	0	UART SIR低功耗模式 该位用来选择IrDA编码模式。如果该位被设为0，低电平位将被当作宽度为3/16位周期的高电平有效的脉冲发送。如果该位被设为1，低电平位将被当作脉宽等于3倍 `IrLPBaud16`输入信号周期的脉冲发送, 而不管所选位速率如何。该位置位时消耗的功率更少，但却可能会缩短发送距离。详见 261页。
1	SIREN	R/W	0	UART SIR使能 如果该位被设为1，IrDA SIR模块将被使能，并且UART将使用SIR协议来发送何接收数据。

位/域	名称	类型	复位	描述
0	UARTEN	R/W	0	UART使能 如果该位被设为1，那么UART将被使能。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 9: UART中断的FIFO深度选择 （UARTIFLS），偏移量 0x034

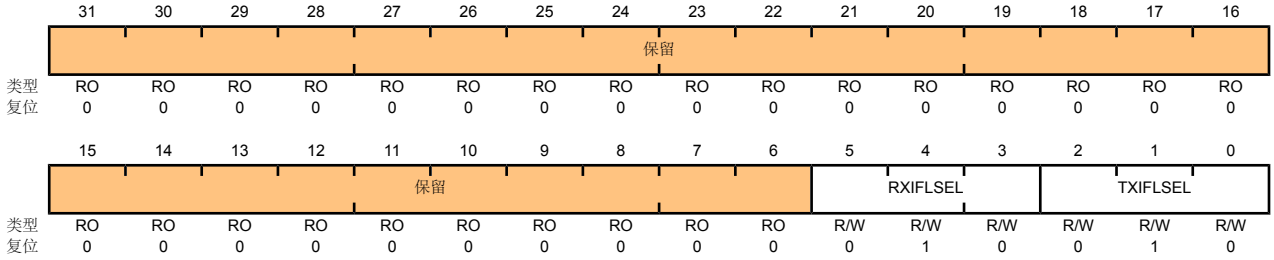
UARTIFLS寄存器是中断的FIFO深度（level）选择寄存器。你可以使用该寄存器来定义**UARTRIS**寄存器中TXRIS和RXRIS位触发（中断）时的FIFO深度。

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)

偏移量 0x034
类型 R/W, 复位 0x0000.0012



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5:3	RXIFLSEL	R/W	0x2	UART接收中断的FIFO深度选择 接收中断的触发点如下: 值 描述 0x0 RX FIFO ≥ 1/8 满 0x1 RX FIFO ≥ ¼ 满 0x2 RX FIFO ≥ ½ 满（默认） 0x3 RX FIFO ≥ ¾ 满 0x4 RX FIFO ≥ 7/8 满 0x5-0x7 保留
2:0	TXIFLSEL	R/W	0x2	UART发送中断的FIFO深度选择 发送中断的触发点如下: 值 描述 0x0 TX FIFO ≤ 1/8 满 0x1 TX FIFO ≤ ¼ 满 0x2 TX FIFO ≤ ½ 满（默认） 0x3 TX FIFO ≤ ¾ 满 0x4 TX FIFO ≤ 7/8 满 0x5-0x7 保留

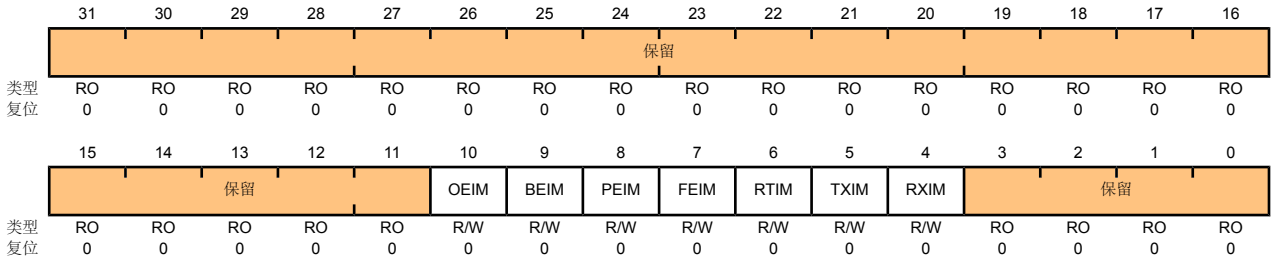
寄存器 10: UART中断屏蔽（UARTIM），偏移量 0x038

UARTIM寄存器是中断屏蔽设置/清除寄存器。

读取时，该寄存器会给出相关中断屏蔽位的当前值。向某个位写1时，将允许与该位相对应的原始中断信号传递到中断控制器。向某个位写0时，则禁止将与该位相对应的原始中断信号传递到中断控制器。

UART中断屏蔽 (UARTIM)

偏移量 0x038
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OEIM	R/W	0	UART溢出错误中断屏蔽位 读取时，返回OEIM中断屏蔽位的当前值。 该位置位时，可以将OEIM中断传递到中断控制器。
9	BEIM	R/W	0	UART中止（break）错误中断屏蔽位 读取时，返回BEIM中断屏蔽位的当前值。 该位置位时，可以将BEIM中断传递到中断控制器。
8	PEIM	R/W	0	UART奇偶校验错误中断屏蔽位 读取时，返回PEIM中断屏蔽位的的当前值。 该位置位时，可以将PEIM中断传递到中断控制器。
7	FEIM	R/W	0	UART帧错误中断屏蔽位 读取时，返回FEIM中断屏蔽位的当前值。 该位置位时，可以将FEIM中断传递到中断控制器。
6	RTIM	R/W	0	UART接收超时中断屏蔽位 读取时，返回RTIM中断屏蔽位的当前值。 该位置位时，可以将RTIM中断传递到中断控制器。
5	TXIM	R/W	0	UART发送中断屏蔽位 读取时，返回TXIM中断屏蔽位的当前值。 该位置位时，可以将TXIM中断传递到中断控制器。

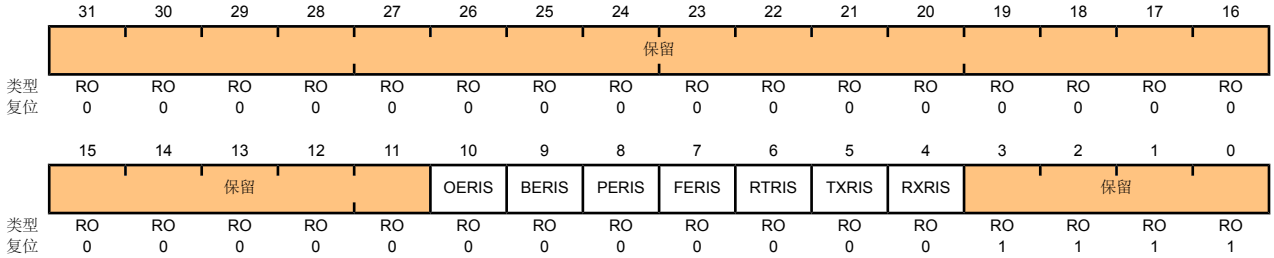
位/域	名称	类型	复位	描述
4	RXIM	R/W	0	UART接收中断屏蔽位 读取时，返回RXIM中断屏蔽位的当前值。 该位置位时，可以将RXIM中断传递到中断控制器。
3:0	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 11: UART原始中断状态（UARTRIS），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)

偏移量 0x03C
类型 RO, 复位 0x0000.000F



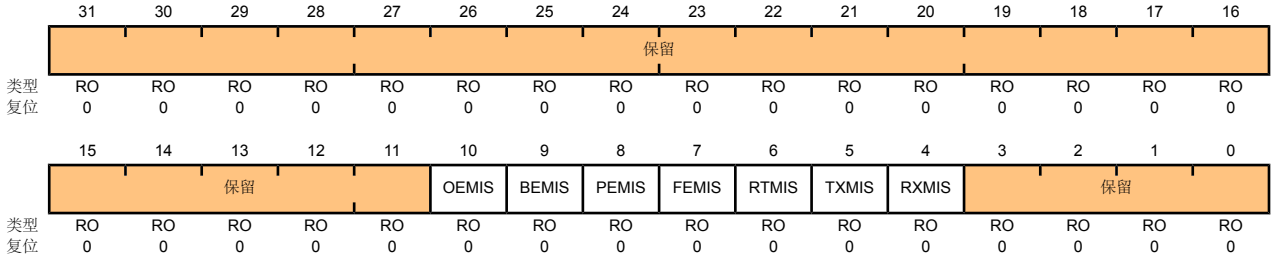
位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OERIS	RO	0	UART溢出错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
9	BERIS	RO	0	UART中止（break）错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
8	PERIS	RO	0	UART奇偶校验错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
7	FERIS	RO	0	UART帧错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
6	RTRIS	RO	0	UART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 显示接收中断的原始（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	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 **12: UART**屏蔽后的中断状态（**UARTMIS**），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)

偏移量 0x040
类型 RO, 复位 0x0000.0000



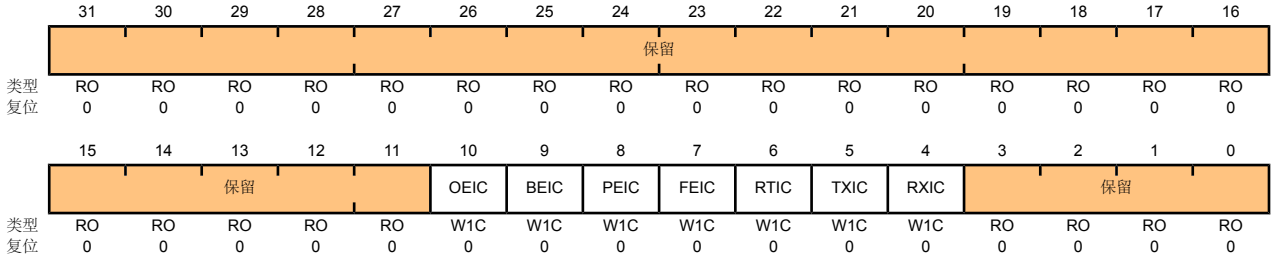
位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OEMIS	RO	0	UART溢出错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
9	BEMIS	RO	0	UART 中止（break）错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
8	PEMIS	RO	0	UART奇偶校验错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
7	FEMIS	RO	0	UART帧错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
6	RTMIS	RO	0	UART接收超时屏蔽后的中断状态 显示接收屏蔽后的中断状态。
5	TXMIS	RO	0	UART发送屏蔽后的中断状态 显示接收屏蔽后的中断状态。
4	RXMIS	RO	0	UART接收屏蔽后的中断状态 显示接收屏蔽后的中断状态。
3:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 13: UART中断清除 （UARTICR），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)

偏移量 0x044
类型 W1C, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OEIC	W1C	0	溢出错误中断清除 OEIC值如下定义： 值 描述 0 中断没有影响。 1 清除中断。
9	BEIC	W1C	0	中止（break）错误中断清除 BEIC值如下定义： 值 描述 0 中断没有影响。 1 清除中断。
8	PEIC	W1C	0	奇偶校验错误中断清除 PEIC值如下定义： 值 描述 0 中断没有影响。 1 清除中断。
7	FEIC	W1C	0	帧错误中断清除 FEIC值如下定义： 值 描述 0 中断没有影响。 1 清除中断。

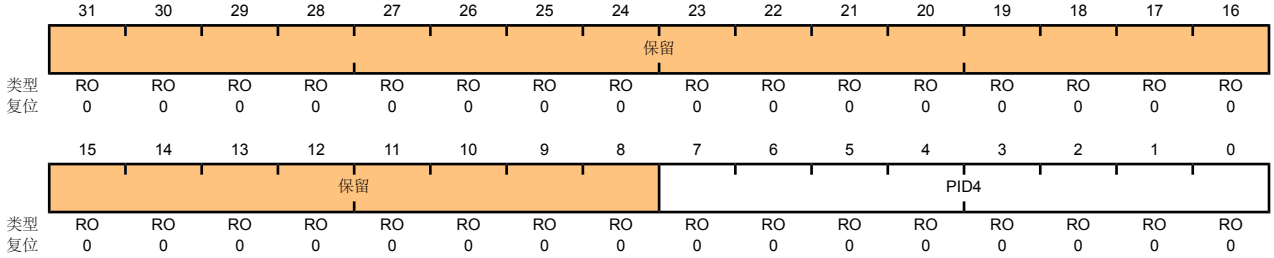
位/域	名称	类型	复位	描述
6	RTIC	W1C	0	接收超时中断清除 RTIC值如下定义: 值 描述 0 中断没有影响。 1 清除中断。
5	TXIC	W1C	0	发送中断清除 TXIC值如下定义: 值 描述 0 中断没有影响。 1 清除中断。
4	RXIC	W1C	0	接收中断清除 RXIC值如下定义: 值 描述 0 中断没有影响。 1 清除中断。
3:0	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 14: UART外设标识4 （UARTPeriphID4），偏移量 0xFD0

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识4 (UARTPeriphID4)

偏移量 0xFD0
类型 RO, 复位 0x0000.0000



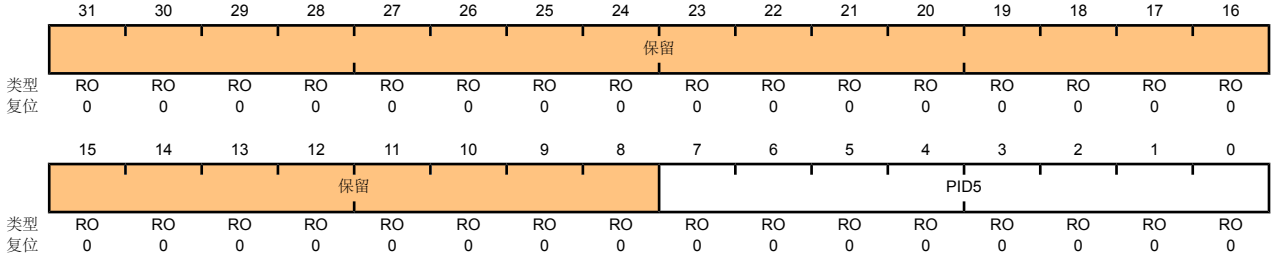
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x0000	UART外设ID寄存器[7:0] 可被软件用来标识该外设的存在与否。

寄存器 **15: UART**外设标识**5**（**UARTPeriphID5**），偏移量 **0xFD4**

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识5 (UARTPeriphID5)

偏移量 0xFD4
类型 RO, 复位 0x0000.0000



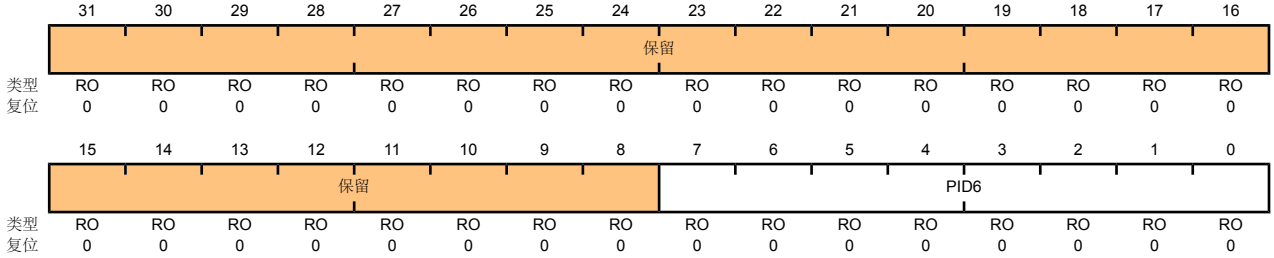
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x0000	UART外设ID寄存器[15:8] 可被软件用来标识该外设的存在与否。

寄存器 16: UART外设标识6（UARTPeriphID6），偏移量 0xFD8

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识6 (UARTPeriphID6)

偏移量 0xFD8
类型 RO, 复位 0x0000.0000



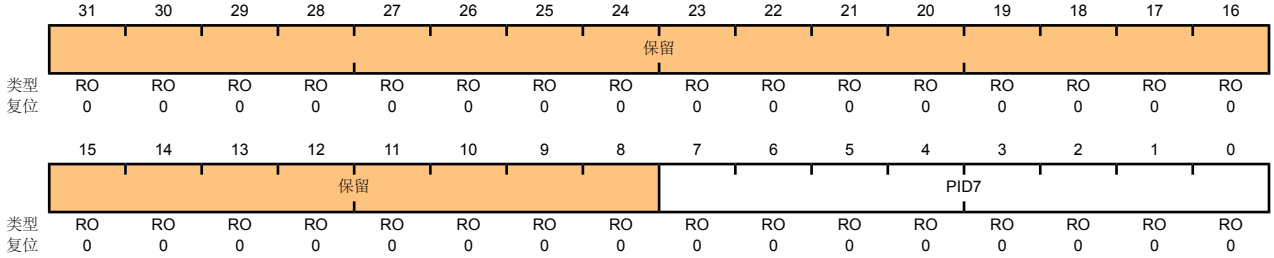
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x0000	UART外设ID寄存器[23:16] 可被软件用来标识该外设的存在与否。

寄存器 **17: UART**外设标识**7**（**UARTPeriphID7**），偏移量 **0xFDC**

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识7 (UARTPeriphID7)

偏移量 0xFDC
类型 RO, 复位 0x0000.0000



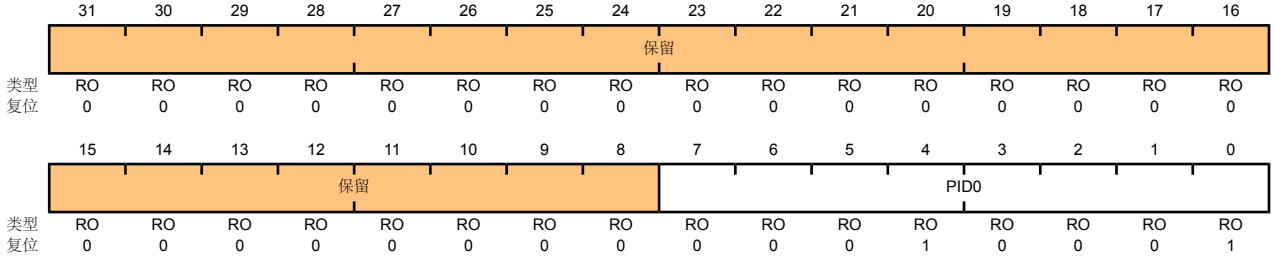
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x0000	UART外设ID寄存器[31:24] 可被软件用来标识该外设的存在与否。

寄存器 18: UART外设标识0（UARTPeriphID0），偏移量 0xFE0

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识0 (UARTPeriphID0)

偏移量 0xFE0
类型 RO, 复位 0x0000.0011



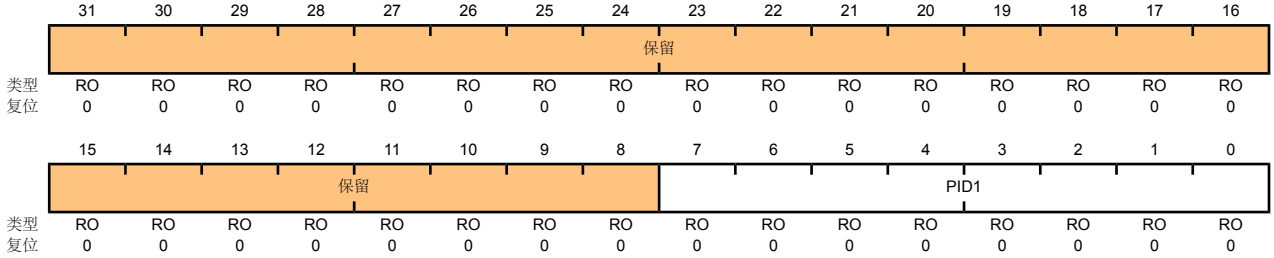
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x11	UART外设ID寄存器[7:0] 可被软件用来标识该外设的存在与否。

寄存器 **19: UART**外设标识**1** （**UARTPeriphID1**），偏移量 **0xFE4**

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识1 (UARTPeriphID1)

偏移量 0xFE4
类型 RO, 复位 0x0000.0000



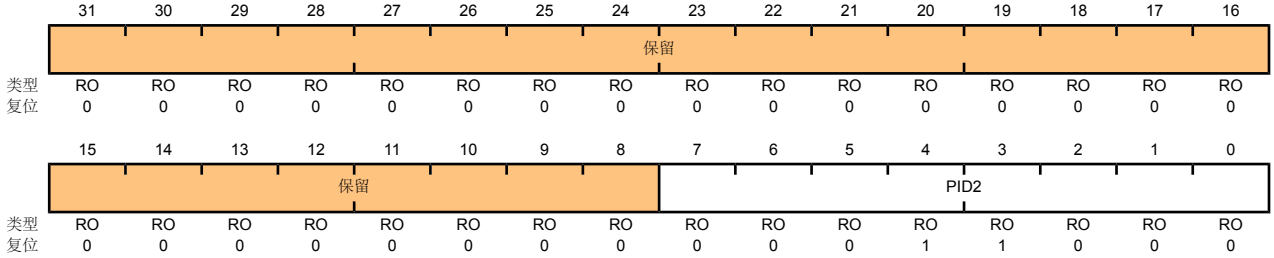
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	UART外设ID寄存器[15:8] 可被软件用来标识该外设的存在与否。

寄存器 20: UART外设标识2（UARTPeriphID2），偏移量 0xFE8

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识2 (UARTPeriphID2)

偏移量 0xFE8
类型 RO, 复位 0x0000.0018



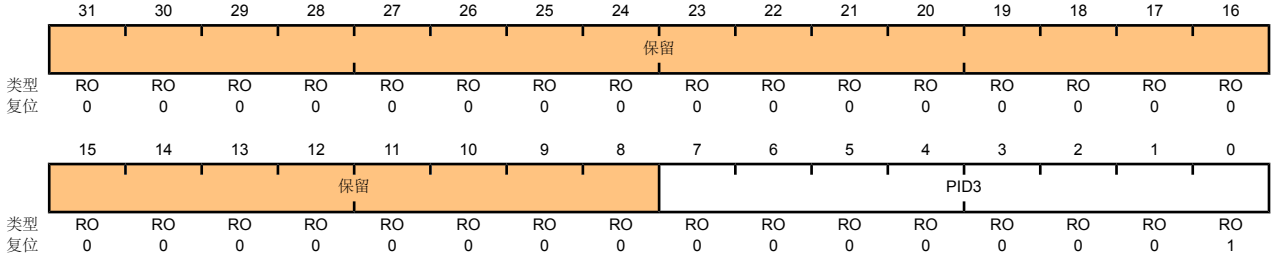
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	UART外设ID寄存器[23:16] 可被软件用来标识该外设的存在与否。

寄存器 **21: UART**外设标识**3** （**UARTPeriphID3**），偏移量 **0xFEC**

UARTPeriphIDn是硬编码的寄存器，其中的位域决定了其复位后的值。

UART外设标识3 (UARTPeriphID3)

偏移量 0xFEC
类型 RO, 复位 0x0000.0001



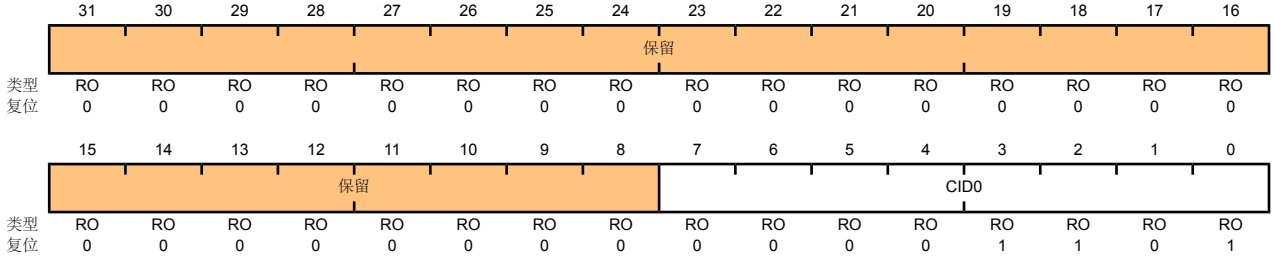
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	UART外设ID寄存器[31:24] 可被软件用来标识该外设的存在与否。

寄存器 22: UART PrimeCell 标识0 （UARTPCellID0），偏移量 0xFF0

UARTPCellIDn 寄存器是硬编码的寄存器，其中的位域决定了其复位后的值。

UART PrimeCell 标识0 (UARTPCellID0)

偏移量 0xFF0
类型 RO, 复位 0x0000.000D



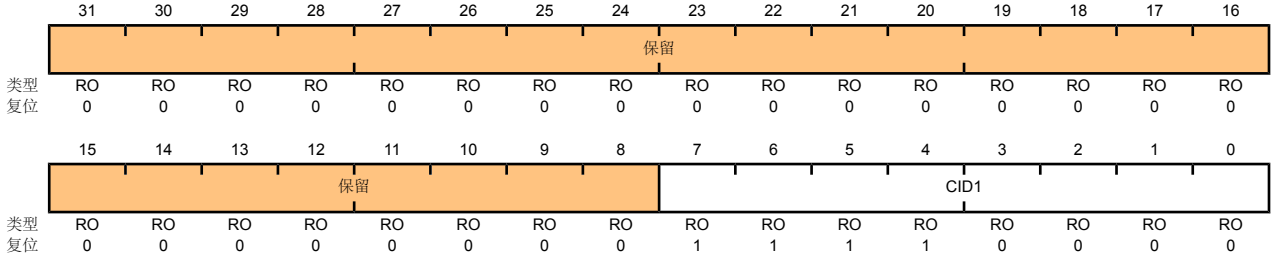
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	UART PrimeCell ID 寄存器[7:0] 提供软件一个标准的跨外设标识系统。

寄存器 **23: UART PrimeCell**标识**1**（**UARTPCellID1**），偏移量 **0xFF4**

UARTPCellIDn 寄存器是硬编码的寄存器，其中的位域决定了其复位后的值。

UART PrimeCell标识1 (UARTPCellID1)

偏移量 0xFF4
类型 RO, 复位 0x0000.00F0



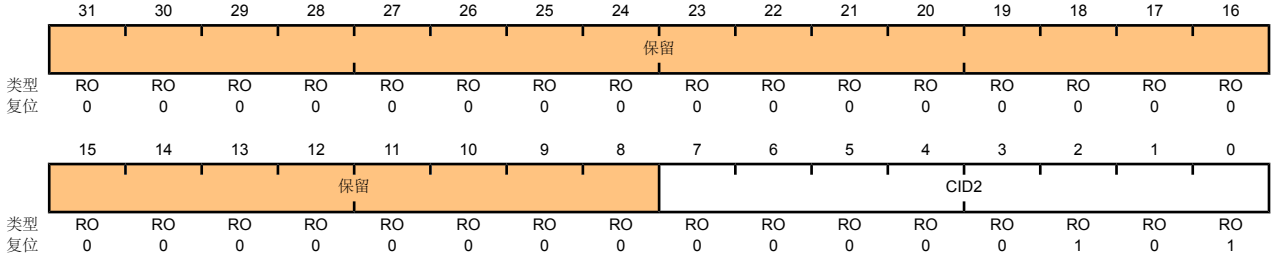
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	UART PrimeCell ID寄存器[15:8] 提供软件一个标准的跨外设标识系统。

寄存器 24: UART PrimeCell标识2（UARTPCellID2），偏移量 0xFF8

UARTPCellIDn 寄存器是硬编码的寄存器，其中的位域决定了其复位后的值。

UART PrimeCell标识2 (UARTPCellID2)

偏移量 0xFF8
类型 RO, 复位 0x0000.0005



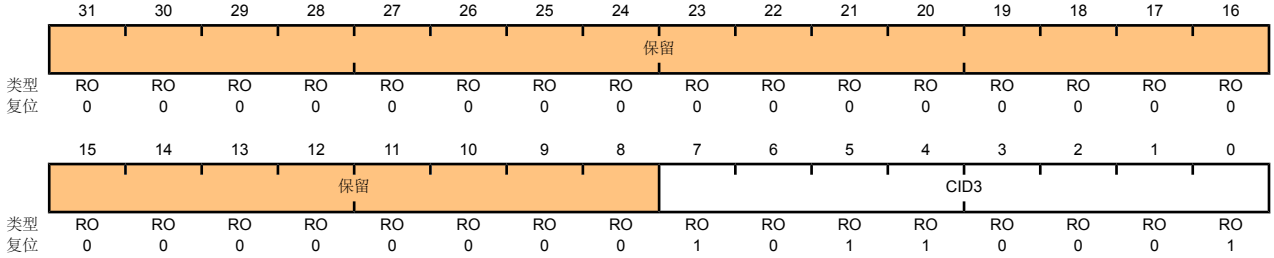
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	UART PrimeCell ID寄存器[23:16] 提供软件一个标准的跨外设标识系统。

寄存器 **25: UART PrimeCell**标识**3**（**UARTPCellID3**），偏移量 **0xFFC**

UARTPCellIDn 寄存器是硬编码的寄存器，其中的位域决定了其复位后的值。

UART PrimeCell标识3 (UARTPCellID3)

偏移量 0xFFC
类型 RO, 复位 0x0000.00B1



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID3	RO	0xB1	UART PrimeCell ID寄存器[31:24] 提供软件一个标准的跨外设标识系统。

13 同步串行接口 (SSI)

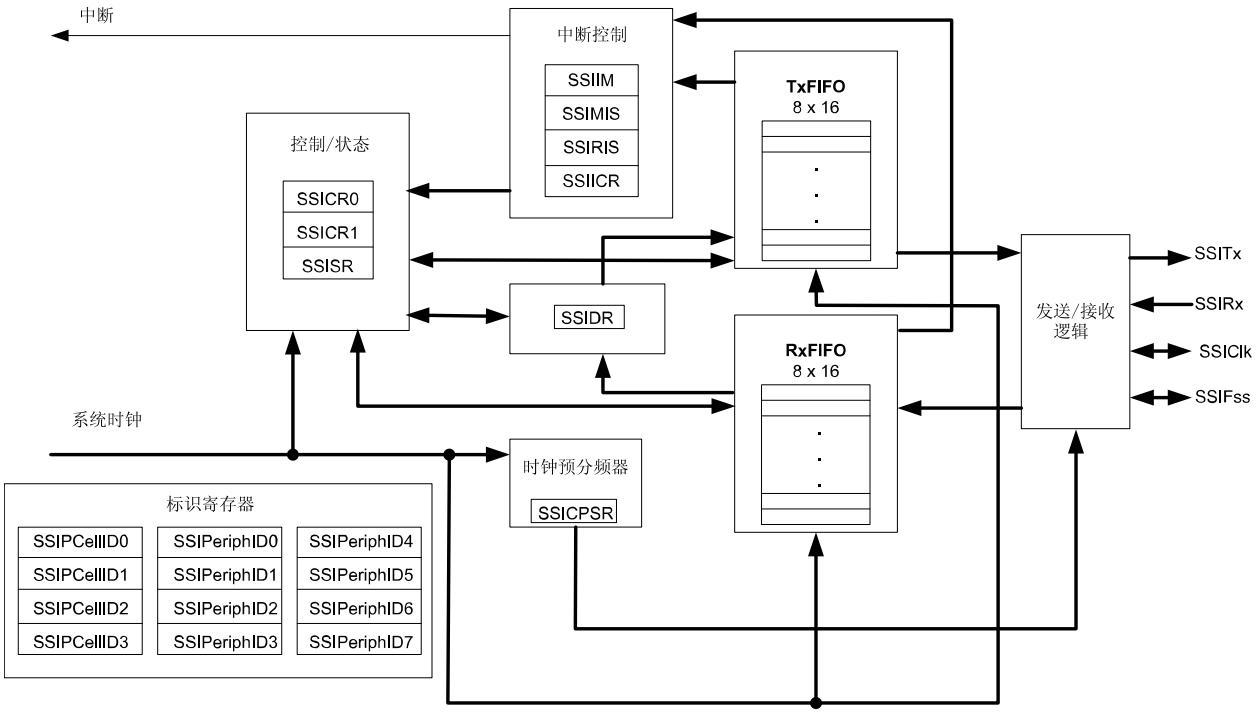
Stellaris® 微控制器包含2个同步串行接口（SSI）模块。每个SSI 是与具有Freescale SPI、MICROWIRE或德州仪器同步串行接口（SSI）的外设器件进行同步串行通信的主机或从机接口。

每个 Stellaris® SSI模块具有以下特性：

- 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、或德州仪器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 结构图

图 **13-1. SSI**模块的结构图



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2 MHz甚至更高的位速率。

串行位速率通过对50-MHz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**时钟分频**(SSICPSR)**寄存器（见 305页）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(见 298页)中设置。

输出时钟SSIClk的频率由下式来定义：

```
FSSIClk = FSysClk / (CPSDVSR * (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。当工作模式为主机模式时，系统时钟至少是SSIClk的两倍。当工作模式为从机模式时，系统时钟至少是SSIClk的12倍。

SSI的时序参数请参考 “同步串行接口（SSI）” 在 473页。

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(见 302页)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，每个都可以通过置位**SSI** 中断屏蔽 **(SSIIM)**寄存器(见 306页)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 (分别见 307页 和 308页）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：

- 德州仪器同步串行
- 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）。

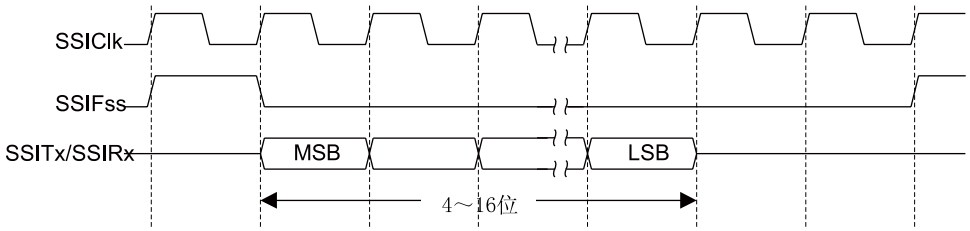
而对于德州仪器同步串行帧格式，在发送每个帧之前，SSIFss管脚会发出一个以上升沿开始并持续一个时钟周期的脉冲。在这种帧格式中，SSI和片外从器件在SSIClk的上升沿驱动各自的输出数据，并在下降沿锁存另一个器件的数据。

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 德州仪器同步串行的帧格式

图 13-2 在 289页显示了一次传输的德州仪器同步串行的帧格式。

图 **13-2. TI**同步串行的帧格式（单次传输）

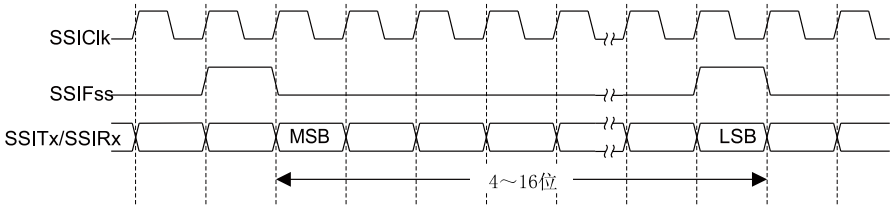


在该模式中，任何时候当SSI空闲时，SSIClk和SSIFss被强制为低电平，发送数据线SSITx为三态。一旦发送FIFO的底部入口包含数据，SSIFss就会变为高电平并持续一个SSIClk周期。 要发送的值也从发送FIFO传输到发送逻辑的串行移位寄存器中。在SSIClk的下一个上升沿，4～16位数据帧的MSB从SSITx管脚移出。同样，接收到的数据的MSB也通过片外串行从器件移到SSIRx管脚上。

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 在 290页显示了背对背（back-to-back）传输时的德州仪器（TI）同步串行帧格式。

图 13-3. TI 同步串行的帧格式（连续传输）



13.2.4.2 Freescale SPI的帧格式

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。

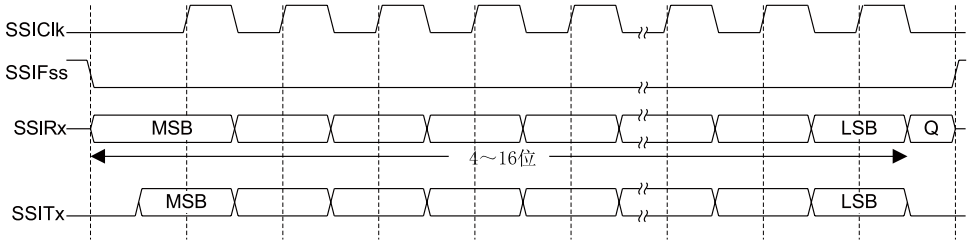
***SPH**相位控制位*

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，Freescale SPI的帧格式

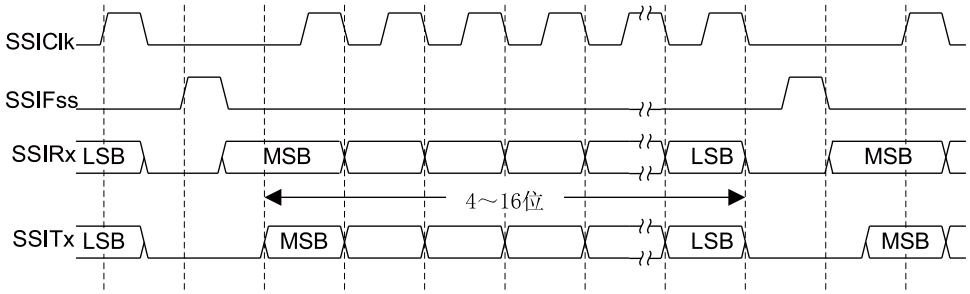
SPO=0和SPH=0时，Freescale SPI帧格式的单次和连续传输信号序列如图 13-4 在 290页 和 图 13-5 在 291页所示。

图 13-4. SPO=0和SPH=0时Freescale SPI 的帧格式 (单次传输)



注意: Q未定义。

图 13-5. SPO=0和SPH=0时Freescale SPI 的帧格式 (连续传输)



在上述配置中，当SSI处于空闲周期时：

- SSIClk 被强制变为低电平
- SSIFss 被强制变为高电平
- 发送数据线 SSITx 被强制变为低电平
- 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。

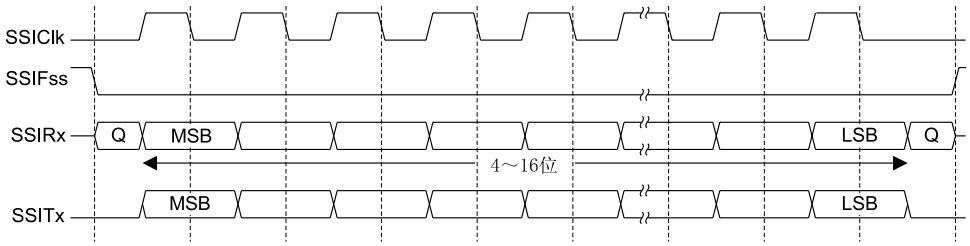
如果传输一个字，则在数据字的所有位都已传输完之后，SSIFss线在捕获到最后一个位之后的一个SSIClk周期返回到其空闲的高电平状态。

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管脚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Freescale SPI的帧格式

SPO=0和SPH=1时，Freescale SPI帧格式的传输信号序列如图 13-6 在 291页所示， 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Freescale SPI的帧格式



注意: Q未定义。

在上述配置中，当SSI处于空闲周期时：

- SSIClk 被强制变为低电平
- SSIFss 被强制变为高电平
- 发送数据线 SSITx 被强制变为低电平
- 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。主机SSITx输出被使能。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。

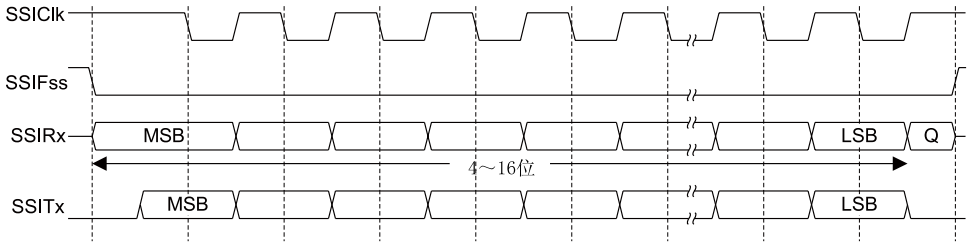
如果传输一个字，则在所有位传输完之后，SSIFss线在捕获到最后一个位之后的一个SSIClk周期返回到其空闲的高电平状态。

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Freescale SPI的帧格式

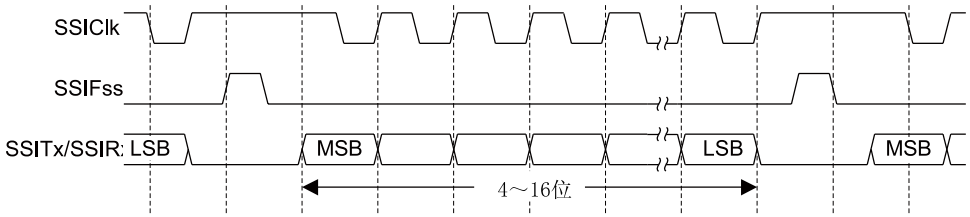
SPO=1和SPH=0时，Freescale SPI帧格式的单次和连续传输信号序列如图 13-7 在 292页 和 图 13-8 在 292页所示。

图 **13-7. SPO=1和SPH=0时Freescale SPI的帧格式（单次传输）**



注意： Q未定义。

图 **13-8. SPO=1和SPH=0时Freescale SPI的帧格式（连续传输）**



在上述配置中，当SSI处于空闲周期时：

- SSIClk 被强制变为高电平
- SSIFss 被强制变为高电平

- 发送数据线 SSITx 被强制变为低电平
- 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。

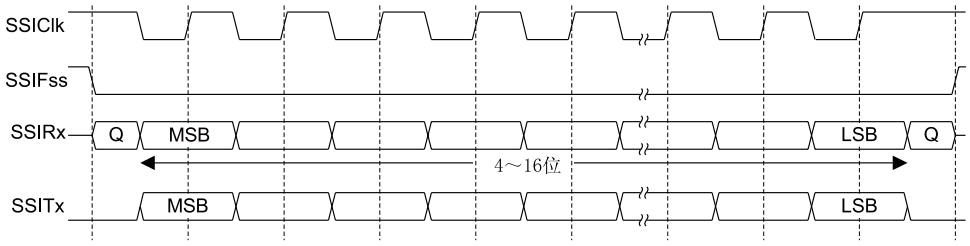
如果传输一个字，则在数据字的所有位传输完之后，SSIFss线在最后一个位传输完之后的一个SSIClk周期返回到其空闲的高电平状态。

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管脚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。

13.2.4.6 SPO=1和SPH=1时Freescale SPI的帧格式

SPO=1和SPH=1时，Freescale SPI帧格式的传输信号序列如图 13-9 在 293页所示，该图涵盖了单次和连续传输两种情况。

图 **13-9. SPO=1和SPH=1时Freescale SPI的帧格式**



注意: Q未定义。

在上述配置中，当SSI处于空闲周期时:

- SSIClk 被强制变为高电平
- SSIFss 被强制变为高电平
- 发送数据线 SSITx 被强制变为低电平
- 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。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。

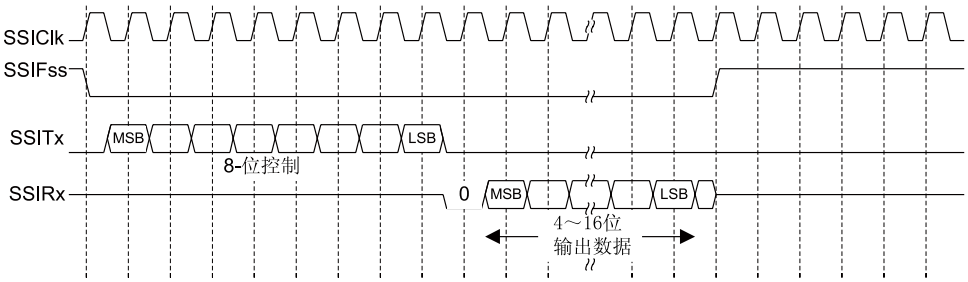
而对于连续的背对背(back-to-back)传输，SSIFss管脚保持其有效的低电平状态，直至最后一个字的最后一位捕获完成，再返回其上述的空闲状态。

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。

13.2.4.7 MICROWIRE的帧格式

图 13-10 在 294页显示了单次传输的MICROWIRE帧格式。图 13-11 在 295页 显示了该格式的背对背（back-to-back）传输情况。

图 **13-10. MICROWIRE**的帧格式 **(**单帧**)**



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。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。

在上述配置中，当SSI处于空闲周期时：

- SSIClk 被强制变为低电平
- SSIFss 被强制变为高电平
- 发送数据线 SSITx 被强制变为低电平

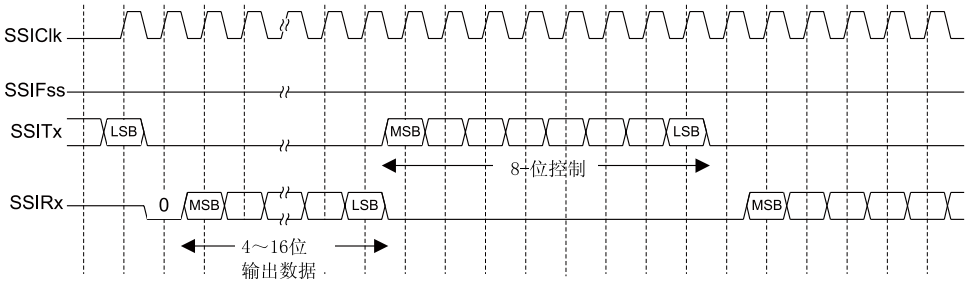
通过向发送FIFO写入一个控制字节可以触发一次传输。在SSIFss的下降沿，发送FIFO底部入口包含的值被传输到发送逻辑的串行移位寄存器中，而8位控制帧的MSB被移出到SSITx管脚上。在该控制帧的传输期间SSIFss保持低电平。SSIRx管脚保持三态。

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。

注意: 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。

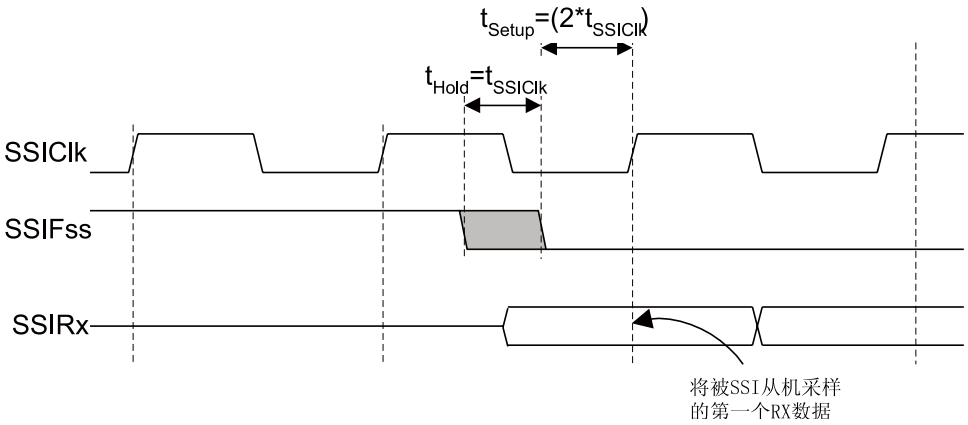
图 **13-11. MICROWIRE**的帧格式 **(**连续传输**)**



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 在 295页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）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**的帧格式，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. 确定SSI为主机还是从机：
   a. 作为主机时，将 **SSICR1** 寄存器的值设为0x0000.0000。
   b. 作为从机时 (输出使能)，将 **SSICR1** 寄存器的值设为0x0000.0004。
   c. 作为从机时 (输出禁止)，将 **SSICR1**寄存器的值设为 0x0000.000C。
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)
- 数据长度 (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)
- 1 Mbps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 / (CPSDVSR * (1 + SCR))
1x106 = 20x106 / (CPSDVSR * (1 + SCR))
```

在此情况下，如果CPSDVSR=2,，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0x0000.0000。
3. 向 **SSICPSR** 寄存器写入0x0000.0002。
4. 向 **SSICR0** 寄存器写入0x0000.09C7。
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 在 296页列出了SSI寄存器。所列偏移量是寄存器相对于SSI的基址的16进制增量：

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

注意: 在对任何控制寄存器重新编程前，必须将SSI禁止(见**SSICR1**寄存器的SSE位)。

表 **13-1. SSI** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	SSICR0	R/W	0x0000.0000	SSI控制0	298
0x004	SSICR1	R/W	0x0000.0000	SSI控制1	300
0x008	SSIDR	R/W	0x0000.0000	SSI数据	302

偏移量	名称	类型	复位	描述	见页面
0x00C	SSISR	RO	0x0000.0003	SSI状态	303
0x010	SSICPSR	R/W	0x0000.0000	SSI时钟预分频	305
0x014	SSIIM	R/W	0x0000.0000	SSI中断屏蔽	306
0x018	SSIRIS	RO	0x0000.0008	SSI原始中断状态	307
0x01C	SSIMIS	RO	0x0000.0000	SSI屏蔽后的中断状态	308
0x020	SSIICR	W1C	0x0000.0000	SSI中断清零	309
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI外设标识4	310
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI外设标识 5	311
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI外设标识6	312
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI外设标识7	313
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI外设标识0	314
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI外设标识 1	315
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI外设标识 2	316
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI外设标识 3	317
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell标识0	318
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell标识1	319
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell标识2	320
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell标识3	321

13.5 寄存器描述

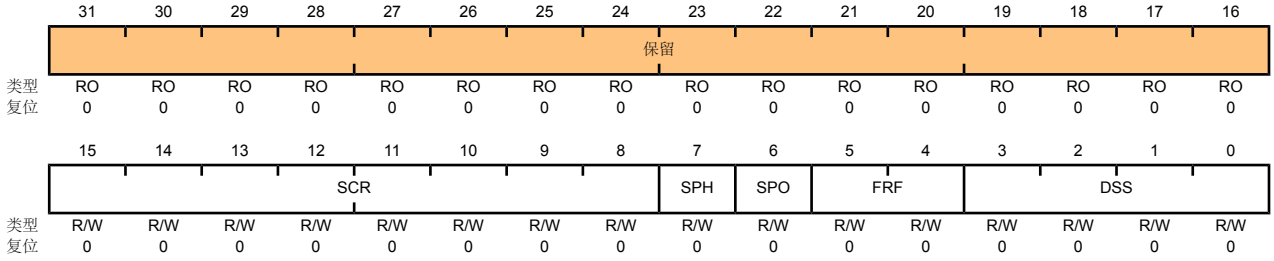
下文将按地址偏移量的数字顺序列举并描述SSI寄存器。

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。

SSI控制0 (SSICR0)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:8	SCR	R/W	0x0000	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=FSSIClk/(CPSDVSR * (1 + SCR))` 此处，CPSDVSR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。
6	SPO	R/W	0	SSI串行时钟的极性 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	0x0	SSI帧格式选择 FRF 的值定义如下: 值　帧格式 0x0 Freescale SPI的帧格式 0x1 德州仪器同步串行的帧格式 0x2 MICROWIRE的帧格式 0x3 保留

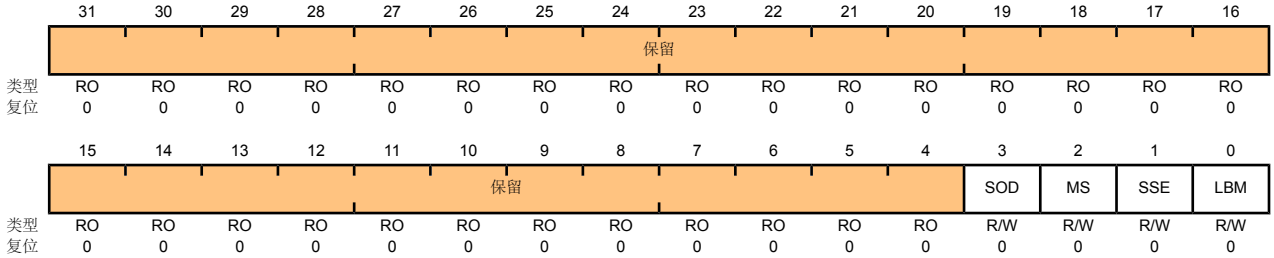
位/域	名称	类型	复位	描述
3:0	DSS	R/W	0x00	SSI数据长度选择 DSS的值定义如下: 值 数据长度 0x0-0x2 保留 0x3 4-位数据 0x4 5-位数据 0x5 6-位数据 0x6 7-位数据 0x7 8-位数据 0x8 9-位数据 0x9 10-位数据 0xA 11-位数据 0xB 12-位数据 0xC 13-位数据 0xD 14-位数据 0xE 15-位数据 0xF 16-位数据

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 (SSICR1)

偏移量 0x004
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	SOD	R/W	0	SSI从机模式输出禁止标志 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从机不驱动SSITx管脚。 该SOD值如下定义： 值 描述 0 在从机输出模式中，SSI可驱动SSITx输出。 1 在从机模式中，SSI必须不可驱动SSITx输出。
2	MS	R/W	0	SSI 主/从选择 该位选择主或从模式并且只有当SSI禁止（SSE=0）时才能修改。 该MS 值如下定义： 值 描述 0 器件配置为主机。 1 器件配置为从机。

位/域	名称	类型	复位	描述
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。 该SSE 值如下定义: 值 描述 0 SSI 操作被禁能。 1 SSI 操作被使能。 注意: 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 返回模式 将该位置位可使能回送（loopback）测试模式。 该LBM 值如下定义: 值 描述 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。在对**SSIDR**寄存器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）。

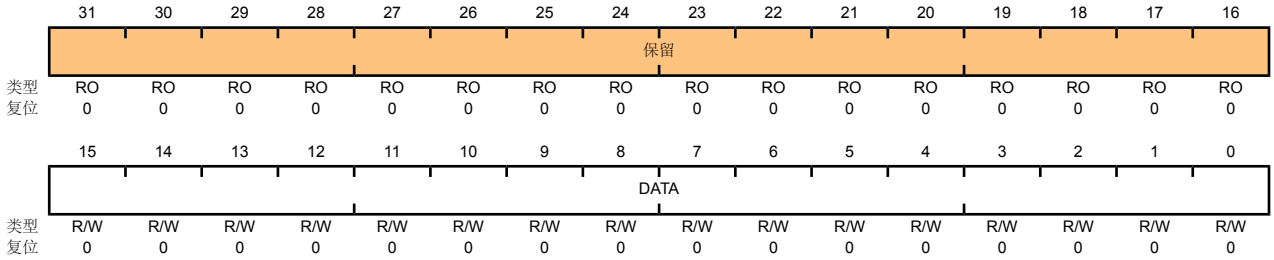
对**SSIDR**进行写操作也就是将数据写入发送FIFO的入口（由写指针来指向）。发送逻辑从发送FIFO中一次移出一个数据值。这个数据值被加载到发送串行移位器中，然后以设置好的位速率串行移出到SSITx管脚。

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
类型 R/W, 复位 0x0000.0000

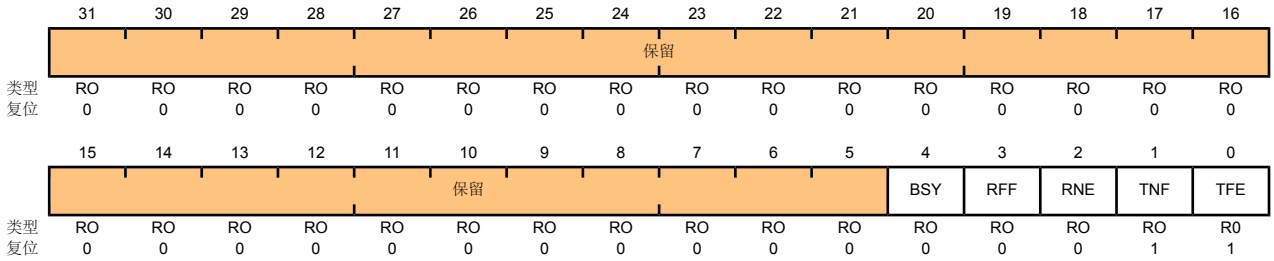


位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	DATA	R/W	0x0000	SSI接收/发送数据 对该位域的读操作即是读接收FIFO。写操作即是写发送FIFO。 当SSI设置为数据大小小于16位时，软件必须正确地调整数据。发送逻辑将忽略顶部未使用的位。接收逻辑自动调整数据。

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
类型 RO, 复位 0x0000.0003



位/域	名称	类型	复位	描述
31:5	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
4	BSY	RO	0	SSI忙位 该BSY 值如下定义： 值 描述 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满标志 该RFF 值如下定义： 值 描述 0 接收 FIFO 未满。 1 接收 FIFO 满。
2	RNE	RO	0	SSI 接收FIFO不空标志 该RNE 值如下定义： 值 描述 0 接收 FIFO为空。 1 接收 FIFO不为空。
1	TNF	RO	1	SSI发送FIFO不满标志 该TNF值如下定义： 值 描述 0 发送 FIFO满。 1 发送 FIFO未满。

位/域	名称	类型	复位	描述
0	TFE	R0	1	SSI发送FIFO空标志 该TFE值如下定义： 值 描述 0 发送 FIFO不为空。 1 发送 FIFO为空。

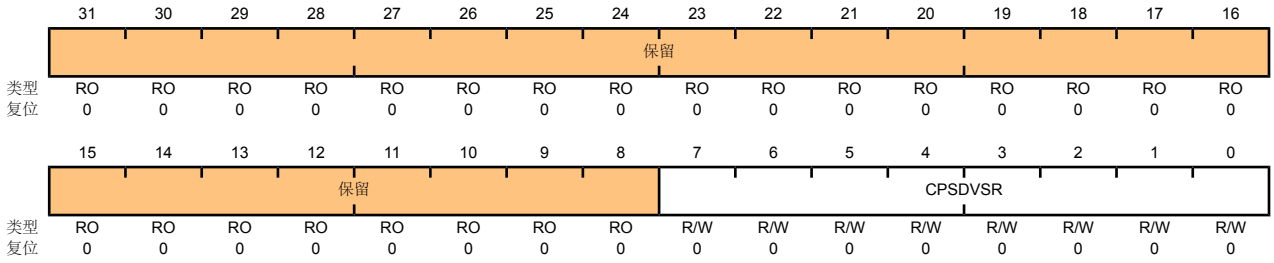
寄存器 5: SSI时钟预分频（SSICPSR），偏移量 0x010

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
类型 R/W, 复位 0x0000.0000



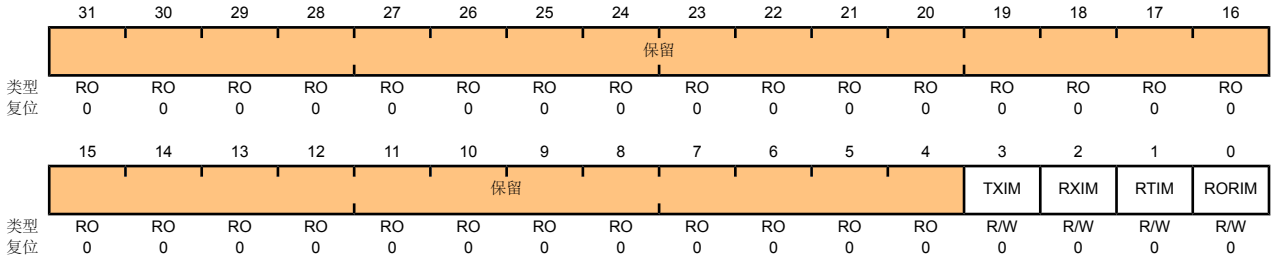
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CPSDVSR	R/W	0x00	SSI时钟预分频因子 该值必须为2-254之间的一个偶数，具体取值由SSIClk的频率决定。执行读操作时，LSB的返回值始终为0。

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
类型 R/W, 复位 0x0000.0000



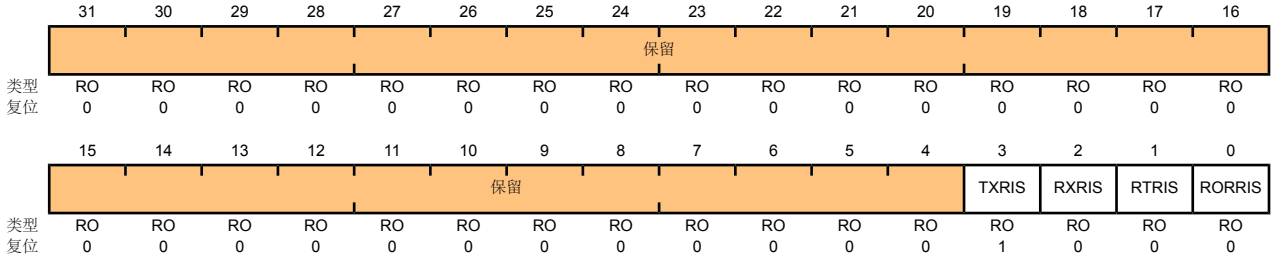
位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXIM	R/W	0	SSI发送FIFO中断屏蔽 该**TXIM** 值如下定义： 值 描述 0 TX FIFO 半满或少于半满条件中断被屏蔽。 1 TX FIFO 半满或少于半满条件中断不被屏蔽。
2	RXIM	R/W	0	SSI接收FIFO中断屏蔽 该**RXIM** 值如下定义： 值 描述 0 RX FIFO 半满或多于半满条件中断被屏蔽。 1 RX FIFO 半满或多于半满条件中断不被屏蔽。
1	RTIM	R/W	0	SSI接收超时中断屏蔽 该**RTIM**值如下定义： 值 描述 0 RX FIFO 超时中断被屏蔽。 1 RX FIFO 超时中断不被屏蔽。
0	RORIM	R/W	0	SSI接收溢出中断屏蔽 该**RORIM** 值如下定义： 值 描述 0 RX FIFO 溢出中断被屏蔽。 1 RX FIFO 溢出中断不被屏蔽。

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。

SSI原始中断状态 (SSIRIS)

偏移量 0x018
类型 RO, 复位 0x0000.0008



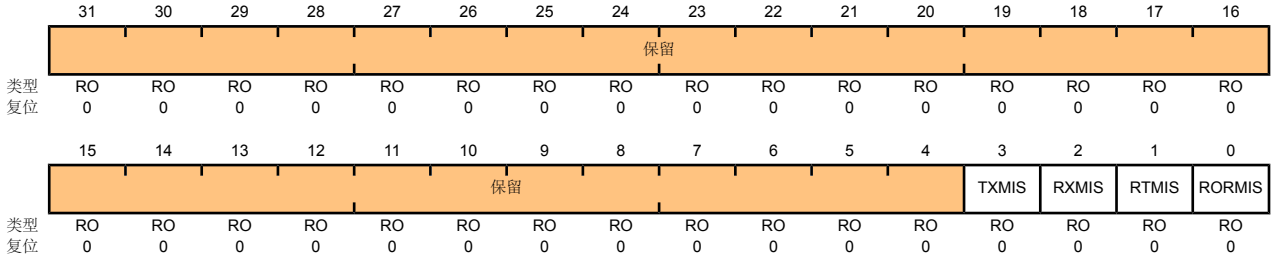
位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXRIS	RO	1	SSI发送FIFO原始中断状态 该位置位表示发送FIFO为半空或少于半空。
2	RXRIS	RO	0	SSI接收FIFO原始中断状态 该位置位表示接收FIFO为半空或多于半空。
1	RTRIS	RO	0	SSI接收超时原始中断状态 该位置位表示发生接收超时。
0	RORRIS	RO	0	SSI接收溢出原始中断状态 该位置位表示接收FIFO溢出。

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
类型 RO, 复位 0x0000.0000



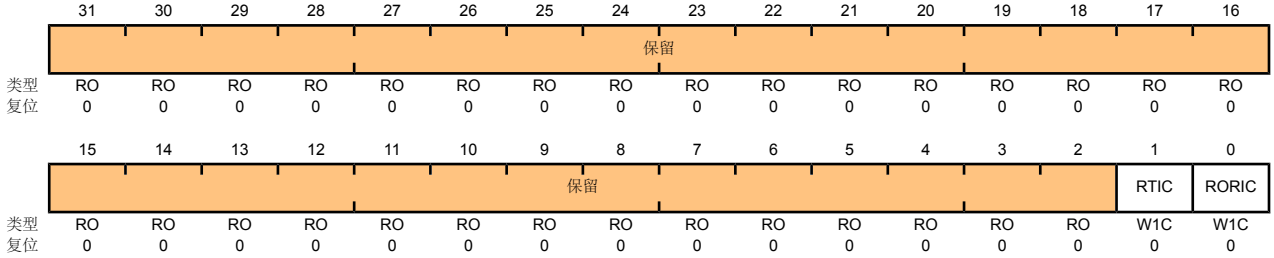
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXMIS	RO	0	SSI发送FIFO屏蔽后的中断状态 该位置位表示发送FIFO为半空或少于半空。
2	RXMIS	RO	0	SSI接收FIFO屏蔽后的中断状态 该位置位表示接收FIFO为半空或多于半空。
1	RTMIS	RO	0	SSI接收超时屏蔽后的中断状态 该位置位表示发生接收超时。
0	RORMIS	RO	0	SSI接收溢出屏蔽后的中断状态 该位置位表示接收FIFO溢出。

寄存器 9: SSI中断清零（SSIICR），偏移量 0x020

SSIICR是中断清零寄存器。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
类型 W1C, 复位 0x0000.0000



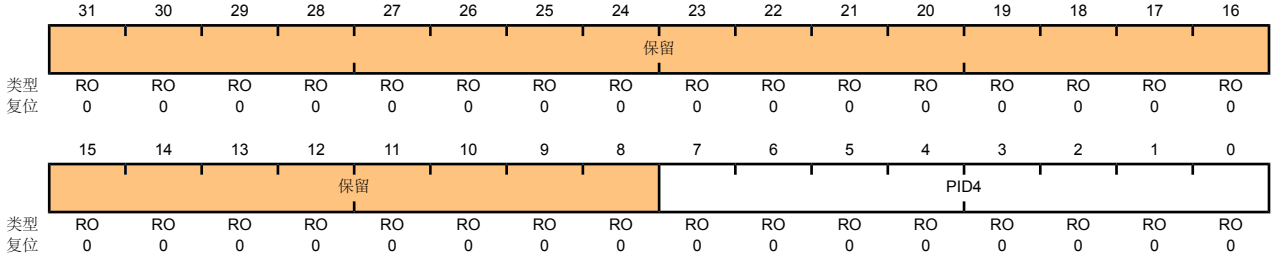
位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	RTIC	W1C	0	SSI接收超时中断清零 该RTIC 值如下定义： 值 描述 0 对中断无影响。 1 清除中断。
0	RORIC	W1C	0	SSI接收溢出中断清零 该RORIC 值如下定义： 值 描述 0 对中断无影响。 1 清除中断。

寄存器 **10: SSI**外设标识**4**（**SSIPeriphID4**），偏移量 **0xFD0**

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识4 (SSIPeriphID4)

偏移量 0xFD0
类型 RO, 复位 0x0000.0000

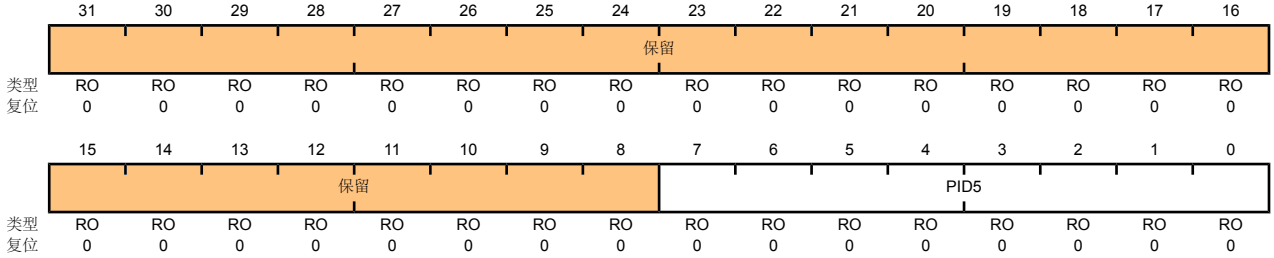


位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	SSI外设ID寄存器[7:0] 可被软件用来标识外设的存在。

寄存器 11: SSI外设标识 5（SSIPeriphID5），偏移量 0xFD4

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识 5 (SSIPeriphID5)
偏移量 0xFD4
类型 RO, 复位 0x0000.0000



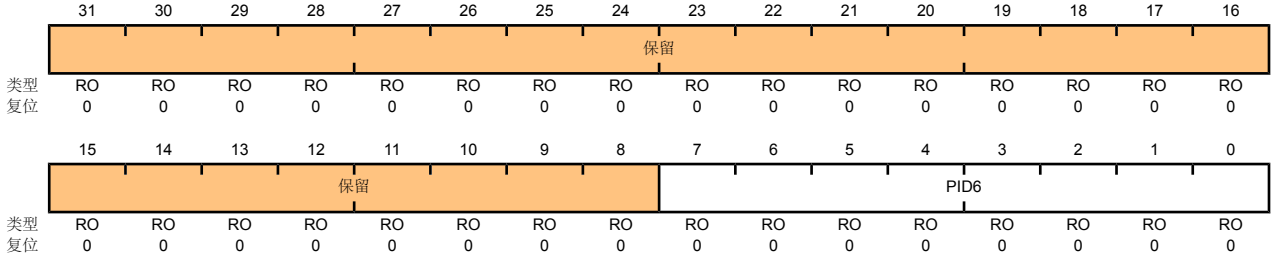
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	SSI外设ID寄存器[15:8] 可被软件用来标识外设的存在。

寄存器 **12: SSI**外设标识**6**（**SSIPeriphID6**），偏移量 **0xFD8**

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识6 (SSIPeriphID6)

偏移量 0xFD8
类型 RO, 复位 0x0000.0000

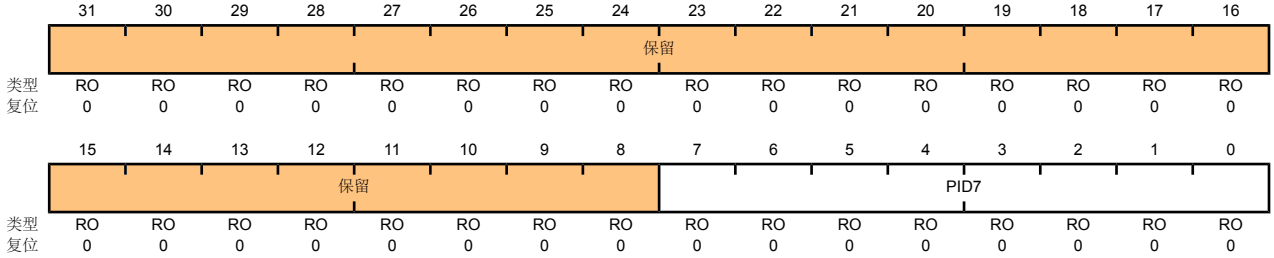


位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	SSI外设ID寄存器[23:16] 可被软件用来标识外设的存在。

寄存器 13: SSI外设标识7（SSIPeriphID7），偏移量 0xFDC

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识7 (SSIPeriphID7)
偏移量 0xFDC
类型 RO, 复位 0x0000.0000



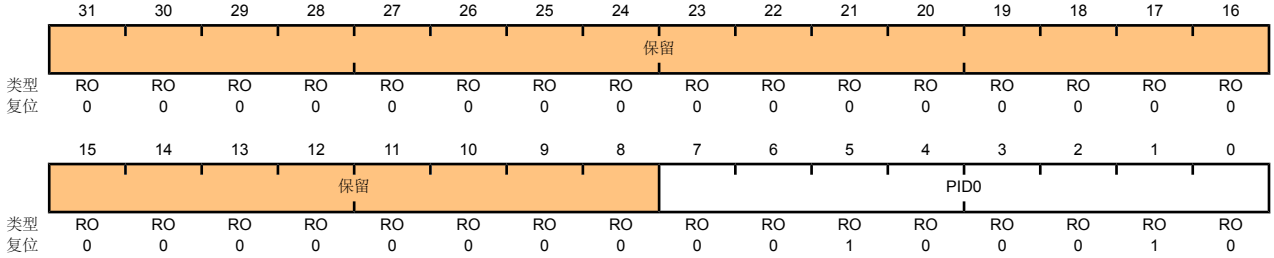
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	SSI外设ID寄存器[31:24] 可被软件用来标识外设的存在。

寄存器 **14: SSI**外设标识**0**（**SSIPeriphID0**），偏移量 **0xFE0**

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识0 (SSIPeriphID0)

偏移量 0xFE0
类型 RO, 复位 0x0000.0022

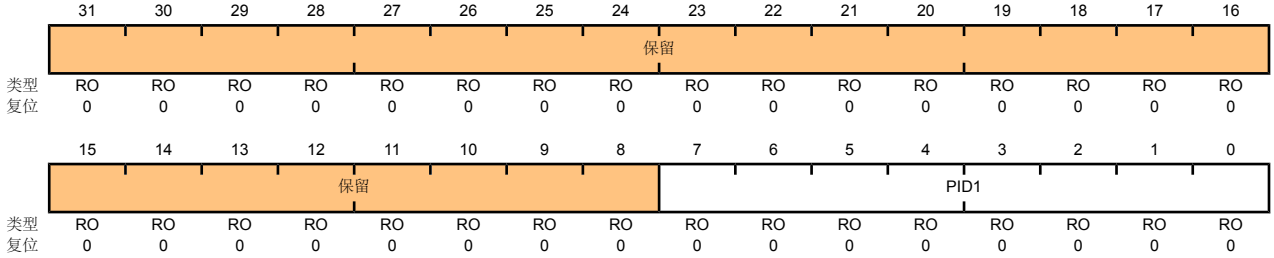


位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x22	SSI外设ID寄存器[7:0] 可被软件用来标识外设的存在。

寄存器 15: SSI外设标识 1（SSIPeriphID1），偏移量 0xFE4

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识 1 (SSIPeriphID1)
偏移量 0xFE4
类型 RO, 复位 0x0000.0000



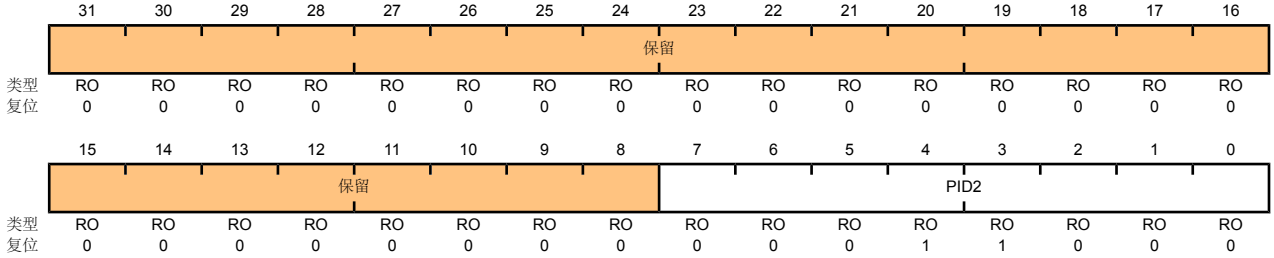
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	SSI外设ID寄存器 [15:8] 可被软件用来标识外设的存在。

寄存器 **16: SSI**外设标识 **2** （**SSIPeriphID2**），偏移量 **0xFE8**

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识 2 (SSIPeriphID2)

偏移量 0xFE8
类型 RO, 复位 0x0000.0018

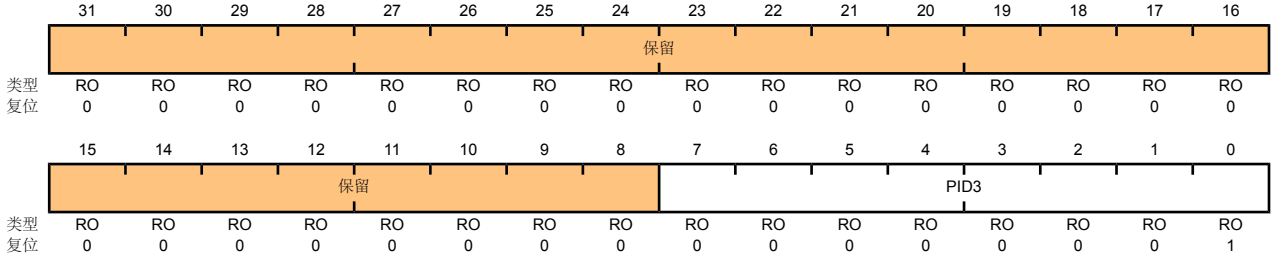


位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	SSI外设ID寄存器 [23:16] 可被软件用来标识外设的存在。

寄存器 17: SSI外设标识 3（SSIPeriphID3），偏移量 0xFEC

SSIPeriphIDn 为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI外设标识 3 (SSIPeriphID3)
偏移量 0xFEC
类型 RO, 复位 0x0000.0001



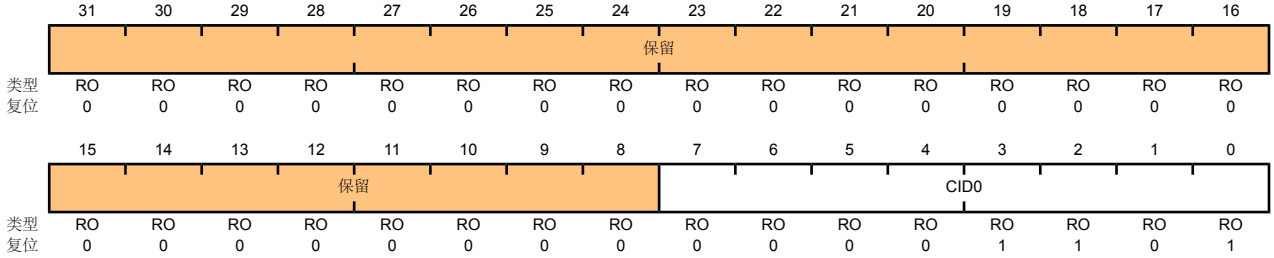
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	SSI外设ID寄存器 [31:24] 可被软件用来标识外设的存在。

寄存器 **18: SSI PrimeCell**标识**0**（**SSIPCellID0**），偏移量 **0xFF0**

SSIPCellIDn为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI PrimeCell标识0 (SSIPCellID0)

偏移量 0xFF0
类型 RO, 复位 0x0000.000D



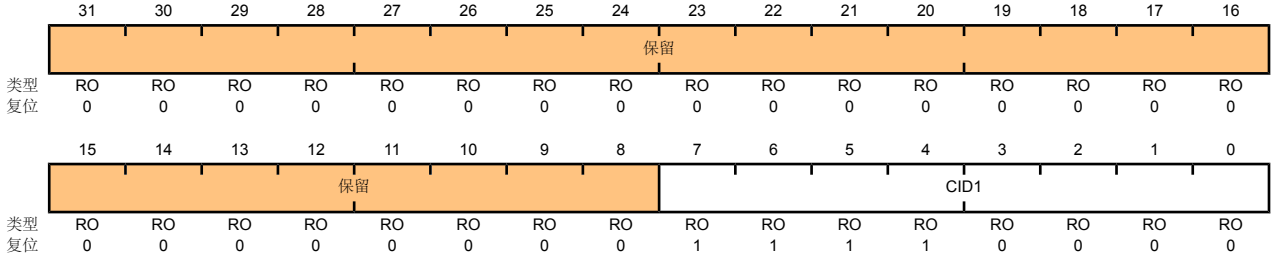
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	SSI PrimeCell ID 寄存器[7:0] 为软件提供一个标准的交叉外设识别系统。

寄存器 19: SSI PrimeCell标识1（SSIPCellID1），偏移量 0xFF4

SSIPCellIDn为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI PrimeCell标识1 (SSIPCellID1)

偏移量 0xFF4
类型 RO, 复位 0x0000.00F0



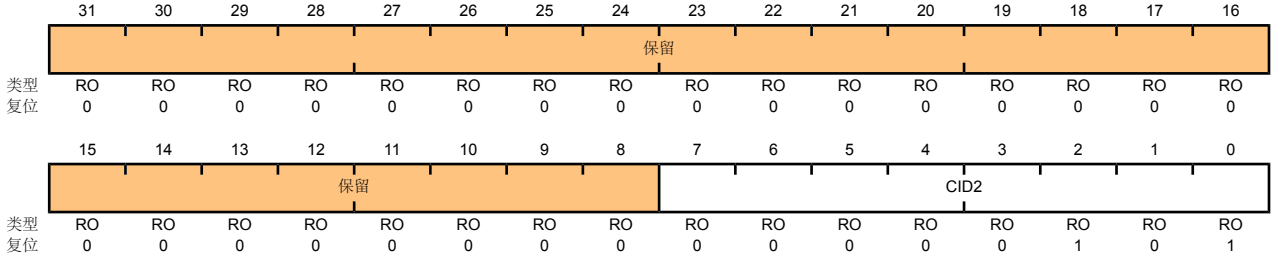
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	SSI PrimeCell ID 寄存器 [15:8] 为软件提供一个标准的交叉外设识别系统。

寄存器 **20: SSI PrimeCell**标识**2**（**SSIPCellID2**），偏移量 **0xFF8**

SSIPCellIDn为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI PrimeCell标识2 (SSIPCellID2)

偏移量 0xFF8
类型 RO, 复位 0x0000.0005



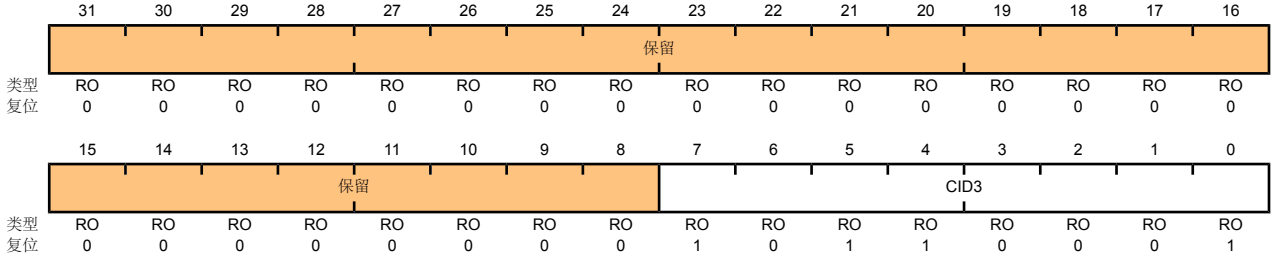
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	SSI PrimeCell ID 寄存器 [23:16] 为软件提供一个标准的交叉外设识别系统。

寄存器 21: SSI PrimeCell标识3（SSIPCellID3），偏移量 0xFFC

SSIPCellIDn为硬编码的寄存器，该寄存器中的位域决定了复位值。

SSI PrimeCell标识3 (SSIPCellID3)

偏移量 0xFFC
类型 RO, 复位 0x0000.00B1



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
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。

14 内部集成电路 (I²C) 接口

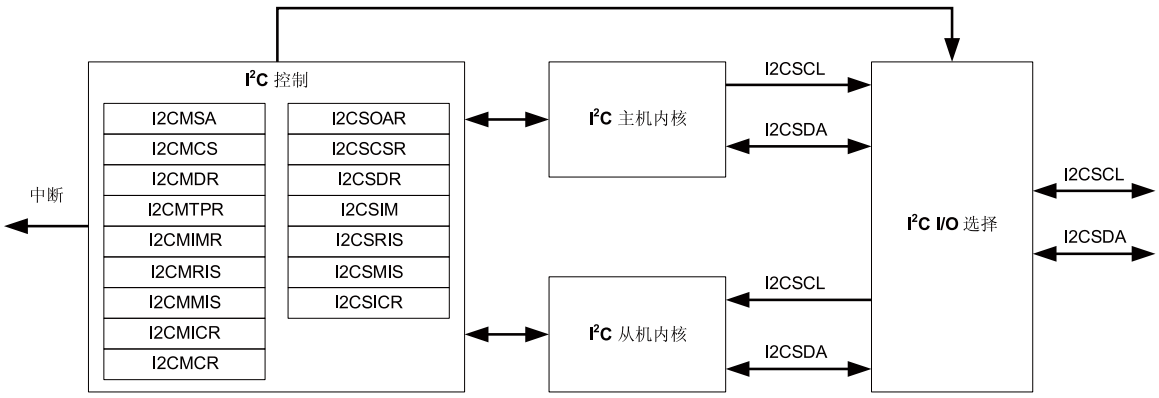
内部集成电路 (I²C) 总线通过采用两线设计（串行数据线SDA和串行时钟线SCL）来提供双向的数据传输，并连接到串行存储器(RAM和ROM)、网络设备、LCD、音频发生器等外部 I²C设备上。 I²C 总线也可在产品的开发和生产过程中用于系统的测试和诊断。LM3S2950微控制器包括1个 I²C 模块， 提供与总线上其它 I²C 器件相互作用（发送和接收）的能力。

I²C 总线上的设备可被指定为主机或从机。这个 Stellaris® I²C 模块支持这些设备作为主机或从机来发送和接收数据，也支持它们作为主机和从机的同步操作。总共有4种 I²C 模式：主机发送、主机接收、从机发送和从机接收。该 Stellaris® I²C 模块 可在两种速率下工作：标准速率(100Kbps)和高速速率(400Kbps)。

I²C 主机和从机都可以产生中断；该 I²C 主机在发送或接收操作完成（或由于错误中止）时产生中断以及 I²C 从机在主机已向其发送数据或发出请求时产生中断。

14.1 方框图

图 **14-1. I²C** 方框图

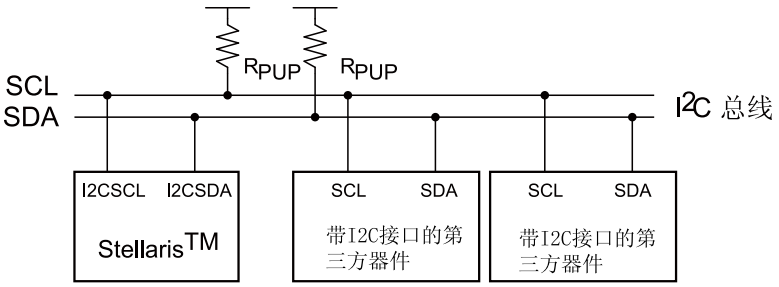


14.2 功能描述

这个I²C 模块由主机和从机两个功能组成，这两个功能均可作为独立的外设来实现。对于正确的操作，SDA和SCL管脚必须被连接到双向的开漏引脚（pad）。典型的 I²C 总线配置如 图 14-2 在 322页所示。

关于 I²C 时序框图，请见“I²C” 在 471页。

图 **14-2. I²C** 总线配置



14.2.1 I²C 总线功能概述

I²C 总线仅使用两个信号：SDA和SCL，这两个信号在 Stellaris® 微控制器中被称为I2CSDA 和 I2CSCL。SDA是双向串行数据线，SCL是双向串行时钟线。当SDA和SCL线都为高电平时，总线处于空闲状态。

I²C 总线上的每次操作都是9位，包括8个数据位和1个应答位。每次传输的字节数（在有效的起始和停止条件之间根据时间来定义，在“起始和停止条件”在323页中详述）没有限制，但每个字节后必须跟随一个应答位，并且数据必须先传输MSB位。当接收器不能接收另一个完整的字节时，它可以保持时钟线SCL为低电平并强制发送器进入等待状态。当接收器释放时钟线SCL时数据传输继续进行。

14.2.1.1 起始和停止条件

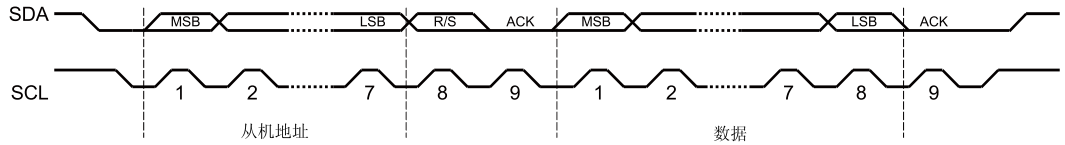
I²C 总线协议定义了两种状态来开始和结束传输：起始和停止。当SCL为高电平时，SDA线上由高到低的跳变被定义为起始条件，由低到高的跳变被定义为停止条件。总线在起始条件之后被视为忙状态，在停止条件之后被视为空闲(free)状态。见 图 14-3 在 323页。

图 **14-3.** 起始和停止条件

14.2.1.2 带有7位地址的数据格式

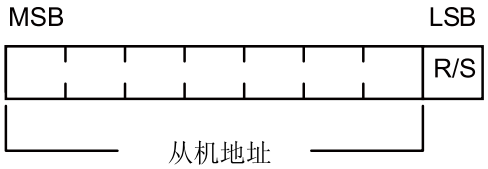
数据传输的格式如 图 14-4 在 323页所示。从机地址在起始条件之后发送。该地址为7位，后面跟的第8位是数据方向位（**I2CMSA** 寄存器中的R/S 位）。数据方向位为0表示传输（发送）；为1表示请求数据（接收）。数据传输始终由主机产生的停止条件来中止。然而，主机仍然可以在总线上通过产生重复的起始条件并寻址另一个从机进行通信，而无需先产生停止条件。因此，在一次传输过程中可能会存在各种不同组合的接收/发送格式。

图 **14-4.** 带有7位地址的完整的数据传输。



首字节的前面7位组成了从机地址（见 图 14-5 在 323页）。第8位决定了消息的方向。首字节的R/S位为0表示主机将向所选择的从机写（发送）数据，该位为1表示主机将接收来自从机的数据。

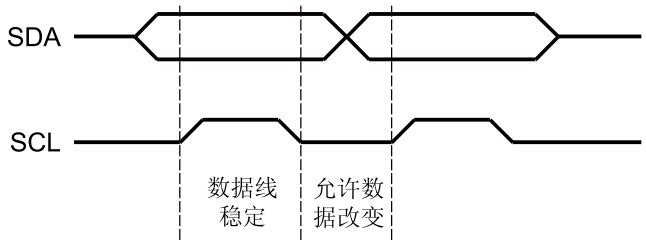
图 **14-5.** 首字节的**R/S**位



14.2.1.3 数据有效性

在时钟的高电平周期期间，SDA线上的数据必须保持稳定，数据线仅可在时钟SCL为低电平时改变（见图 14-6 在 324页）。

图 **14-6.** 在 **I²C** 总线的位传输过程中的数据有效性



14.2.1.4 应答

所有总线传输都带有所需的应答时钟周期，该时钟周期由主机产生。发送器（可以是主机或从机）在应答周期过程中释放SDA线。为了应答传输，接收器必须在应答时钟周期过程中拉低SDA。接收器在应答周期过程中发出的数据必须符合“数据有效性” 在 324页中说明的数据有效性的要求。

当从机接收器不应答从机地址时，从机必须将SDA保持在高电平状态，使得主机可产生停止条件来中止当前的传输。如果主机在传输过程中用作接收器，那么它有责任应答从机发出的每次传输。由于主机控制着传输中的字节数，因此它通过在最后一个数据字节上不产生应答来向从机发送器指示数据的结束。然后从机发送器必须释放SDA线，以便主机可以产生停止条件或重复起始条件。

14.2.1.5 仲裁

只有在总线空闲时，主机才可以启动传输。在起始条件的最少保持时间内，两个或两个以上的主机都有可能产生起始条件。在这些情况下，当SCL为高电平时仲裁机制在SDA线上产生。在仲裁过程中，第一个竞争的主机器件在SDA上设置‘1’(高电平)，而另一个主机发送‘0’(低电平）将关闭（switch off）其数据输出阶段并退出直至总线再次空闲。

仲裁可以在几个位上发生。仲裁的第一个阶段是比较地址位，如果两个主机都试图寻址相同的器件，则仲裁继续比较数据位。

14.2.2 可用的速率模式

I²C时钟速率由下列参数决定：CLK_PRD, TIMER_PRD, SCL_LP和SCL_HP。

其中：

CLK_PRD为系统时钟周期

SCL_LP 为SCL时钟的低电平阶段（固定为6）

SCL_HP 为SCL时钟的高电平阶段（固定为4）

TIMER_PRD在 **I²C** 主机定时器周期（**I2CMTPR**）寄存器中是已设定的值（见 339页）。

I²C 时钟周期的计算如下：

```
SCL_PERIOD = 2*(1 + TIMER_PRD)*(SCL_LP + SCL_HP)*CLK_PRD
```

例如：

```
CLK_PRD = 50 ns
TIMER_PRD = 2
```

```
SCL_LP=6
SCL_HP=4
```

得出的 SCL 频率为:

```
1/T = 333 Khz
```

表 14-1 在 325页给出了定时器周期、系统时钟和速率模式（标准或高速）的例子。

表 **14-1. I²C** 主机定时器周期与速率模式的例子

系统时钟	定时器周期	标准模式	定时器周期	高速模式
4 Mhz	0x01	100 Kbps	-	-
6 Mhz	0x02	100 Kbps	-	-
12.5 Mhz	0x06	89 Kbps	0x01	312 Kbps
16.7 Mhz	0x08	93 Kbps	0x02	278 Kbps
20 Mhz	0x09	100 Kbps	0x02	333 Kbps
25 Mhz	0x0C	96.2 Kbps	0x03	312 Kbps
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps
50Mhz	0x18	100 Kbps	0x06	357 Kbps

14.2.3 中断

I²C 可在观察到下列条件时产生中断:

- 主机传输完成
- 主机传输错误
- 已接收从机传输
- 已请求从机传输

I²C 主机和 I²C 模块都有各自的中断信号。当两个模块都可产生多个条件的中断时，只有一个中断信号被发送到中断控制器。

14.2.3.1 I²C 主机中断

当传输结束（发送或接收）、或在传输过程中出现错误时，I²C 主机模块产生一个中断。要使能 I²C 主机中断，软件必须写 '1' 到 **I²C** 主机中断屏蔽（**I2CMIMR**） 寄存器。当符合中断条件时，软件必须检查 **I²C**主机控制/状态（**I2CMCS**）寄存器的 ERROR 位来确认错误不是在最后一次传输中产生。如果最后一次传输没有被从机应答或如果主机由于与另一个主机竞争时丢失仲裁而被强制放弃总线的所有权，那么会发出一个错误条件。如果没有检测到错误，则应用可继续执行传输。中断通过写 '1' 到 **I²C** 主机中断清零（**I2CMICR**）寄存器来清除。

如果应用不要求使用中断，那么原始中断状态总是可通过 **I²C** 主机原始中断状态（**I2CMRIS**）寄存器来看到。

14.2.3.2 I²C 从机中断

从机模块在它接收到来自 I²C 主机的请求时产生中断。为了使能 I²C 从机中断，写 '1' 到 **I²C** 从机中断屏蔽（**I2CSIMR**）寄存器。通过检查**I²C** 从机控制/状态（**I2CSCSR**）寄存器的RREQ 和 TREQ 位，软件确定模块是否应该写（发送）数据到**I²C** 从机数据（**I2CSDR**） 寄存器或从该寄存器中读取（接

收）数据。如果从机模块在接收模式中且传输的第一个字节被接收，那么 FBR位与 RREQ位一起置位。通过写 '1' 到 **I²C** 从机中断清除 **(I2CSICR)**寄存器来清除中断。

如果应用不要求使用中断，那么原始中断状态总是可以通过 **I²C** 从机原始中断状态（**I2CSRIS**） 寄存器观察到。

14.2.4 回送操作

该 I²C 模块可放置到内部回送模式以用于诊断或调试工作。这通过置位 **I²C** 主机配置（**I2CMCR**）寄存器的 LPBK 位来完成。在回送模式中，主机和从机模块的SDA和SCL信号结合在一起。

14.2.5 命令序列流程图

该小节详述了在主机和从机模式下执行不同的 I²C 传输类型所要求的步骤。

14.2.5.1 I²C 主机命令序列

下图所示为 I²C 主机可用的命令序列。

图 **14-7.** 主机单次发送

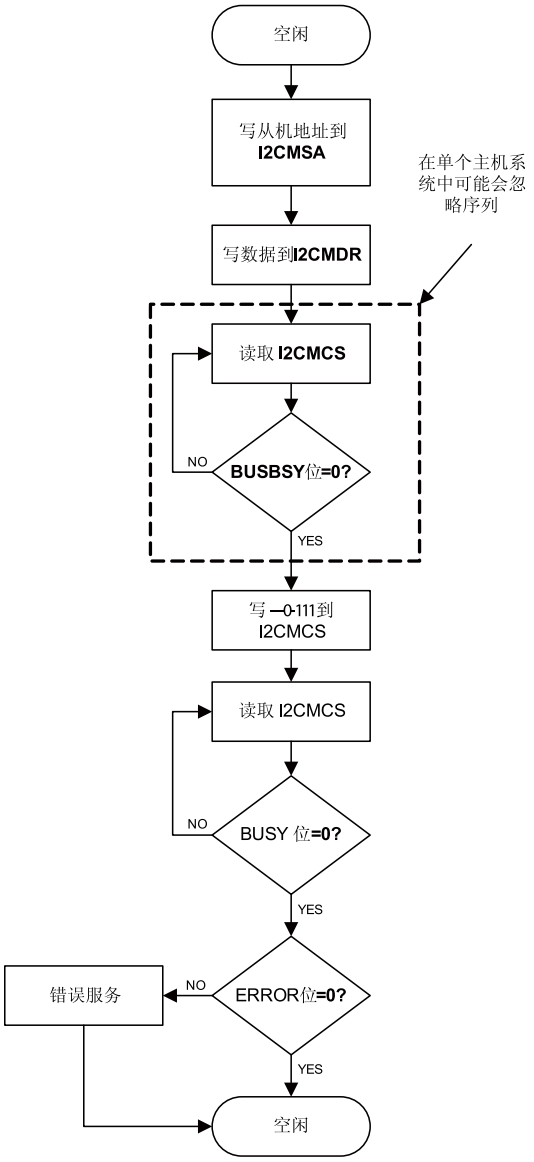

空闲
写从机地址到
I2CMSA
在单个主机系
统中可能会忽
略序列
写数据到I2CMDR
读取 I2CMCS
NO
BUSBSY位=0?
YES
写 —0-111到
I2CMCS
读取 I2CMCS
NO
BUSY 位=0?
YES
错误服务
NO
ERROR位=0?
YES
空闲

图 **14-8.** 主机单次接收

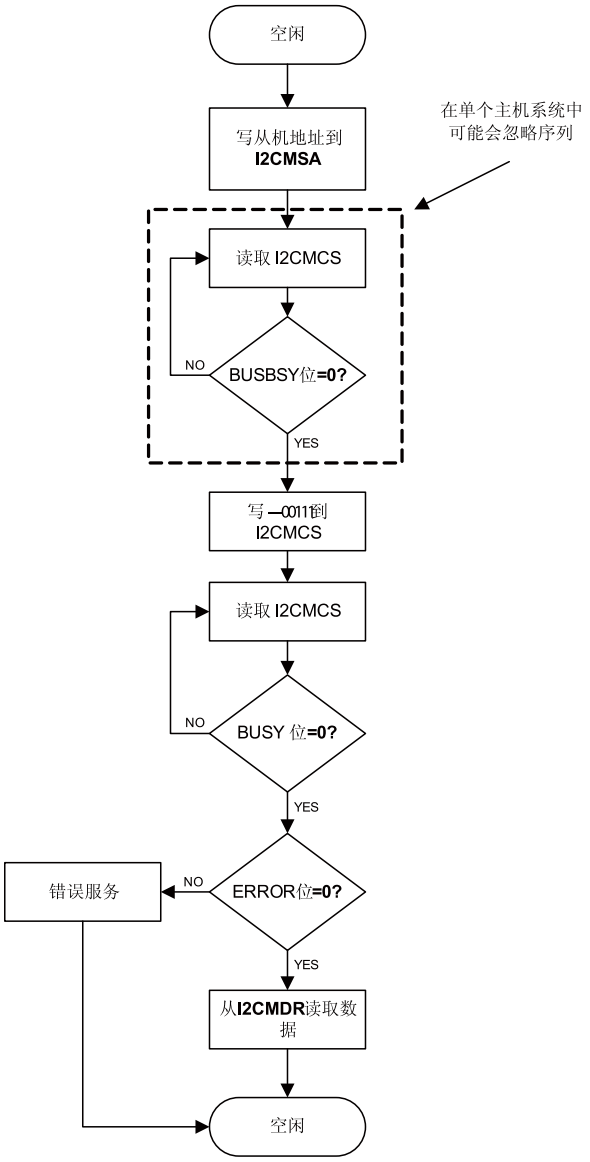

空闲
写从机地址到
I2CMSA
在单个主机系统中
可能会忽略序列
读取 I2CMCS
NO
BUSBSY位=0?
YES
写 —0011到
I2CMCS
读取 I2CMCS
NO
BUSY 位=0?
YES
错误服务
NO
ERROR位=0?
YES
从I2CMDR读取数
据
空闲

图 **14-9.** 主机突发发送

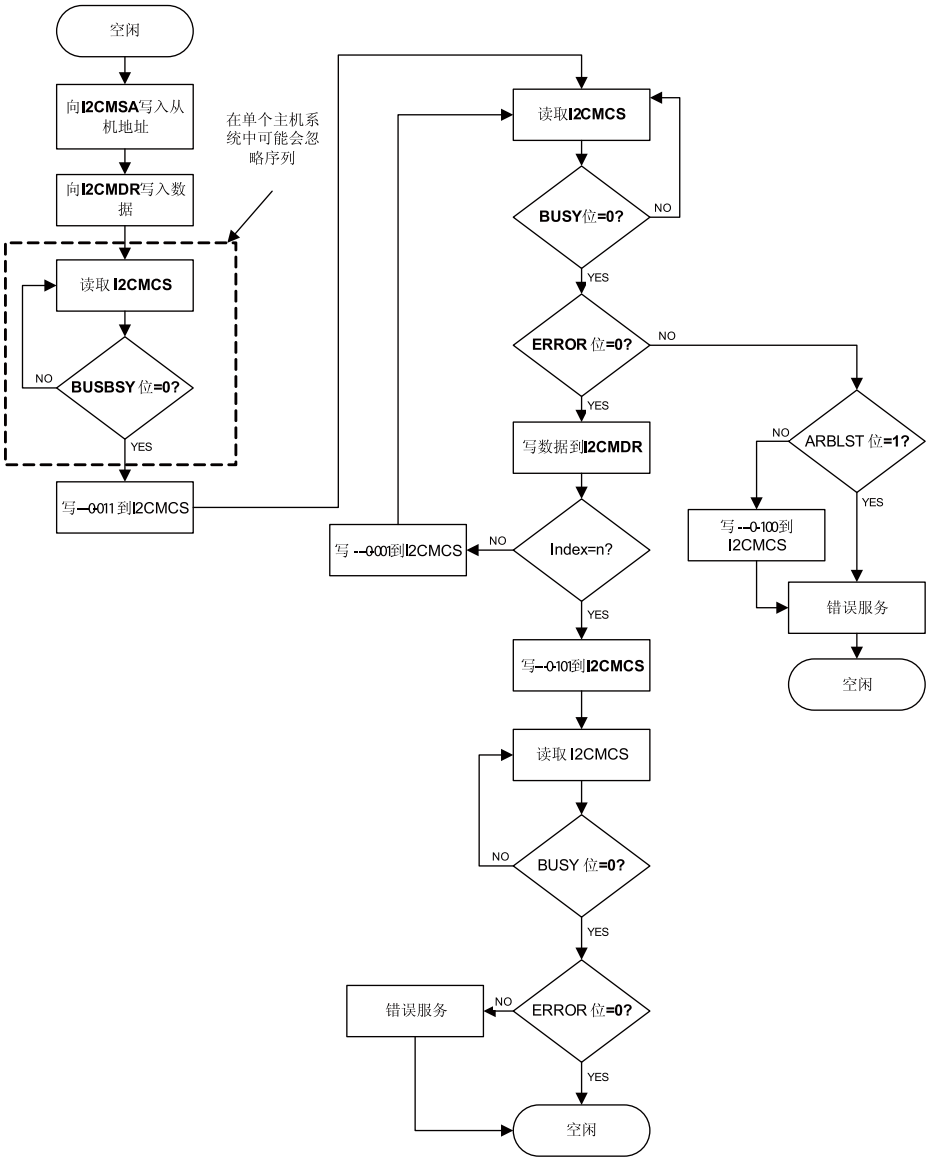

空闲
向I2CMSA写入从机地址
向I2CMDR写入数据
读取 I2CMCS
BUSBSY 位=0?
NO
YES
写—0011 到I2CMCS
在单个主机系统中可能会忽略序列
读取I2CMCS
BUSY位=0?
NO
YES
ERROR 位=0?
NO
YES
写数据到I2CMDR
Index=n?
NO
写 —0001到I2CMCS
YES
写—0101到I2CMCS
读取 I2CMCS
BUSY 位=0?
NO
YES
ERROR 位=0?
NO
错误服务
YES
空闲
ARBLST 位=1?
NO
YES
写 —0100到 I2CMCS
错误服务
空闲

图 **14-10.** 主机突发接收

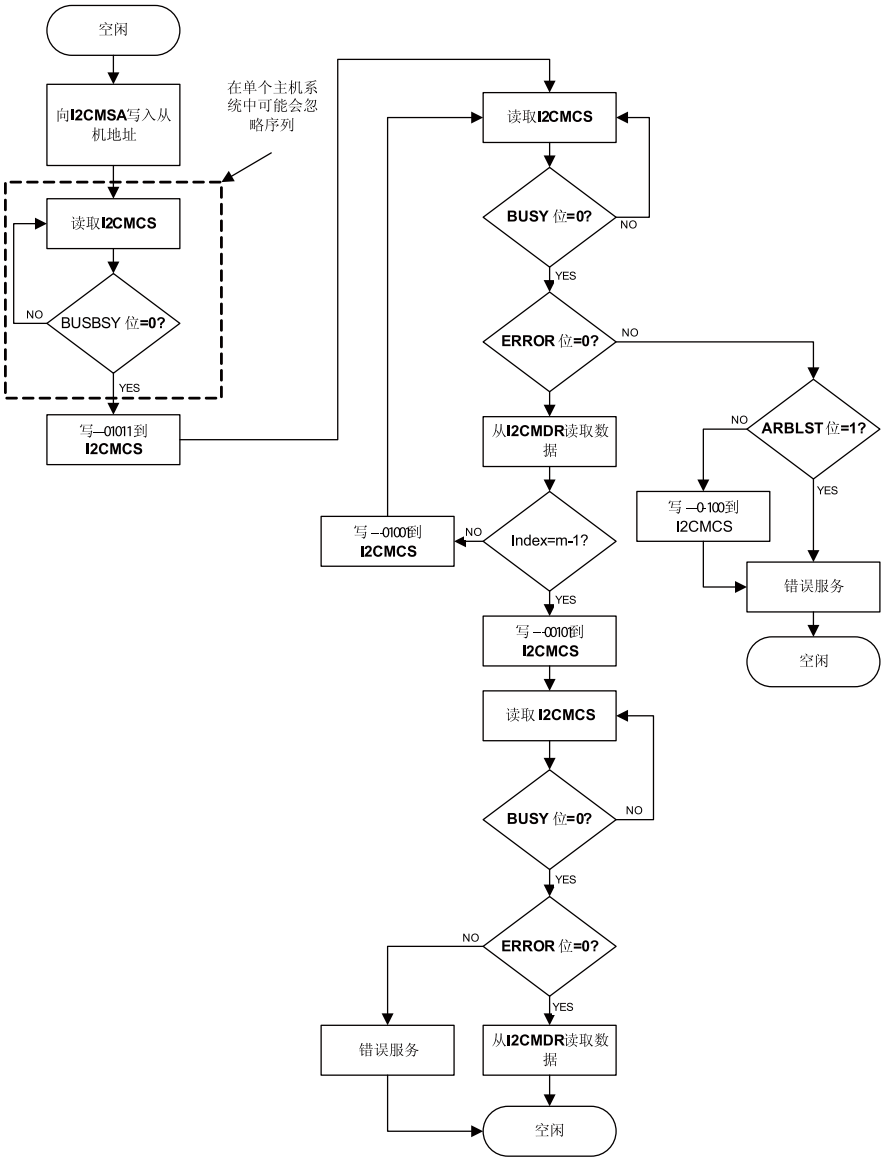


图 **14-11.** 在突发发送后主机突发接收

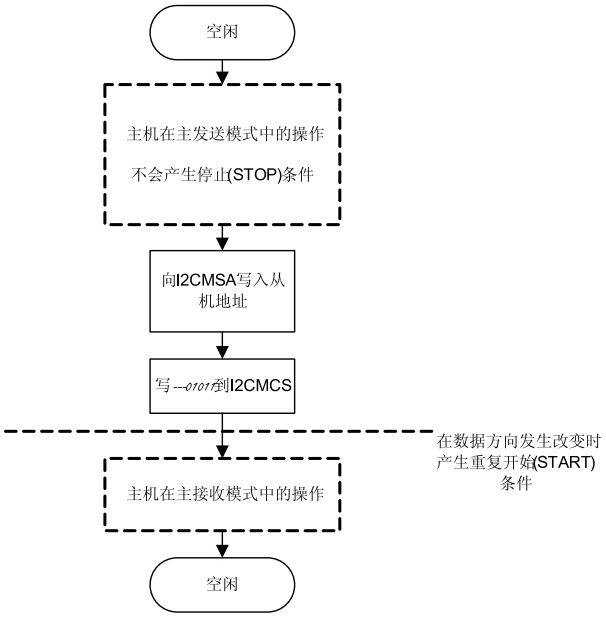
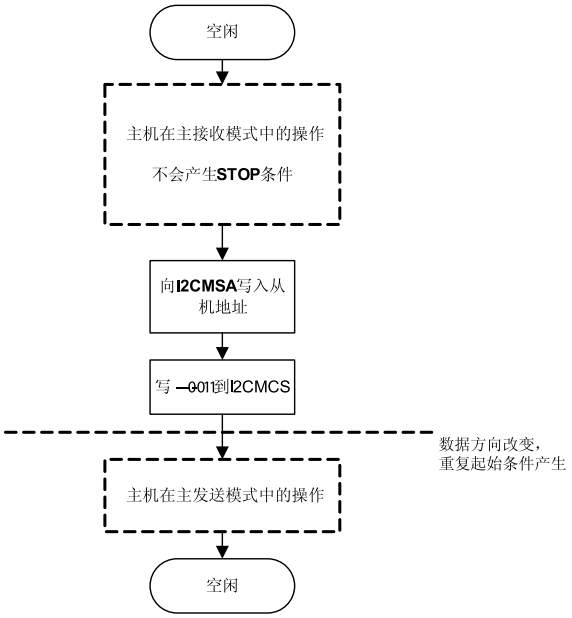


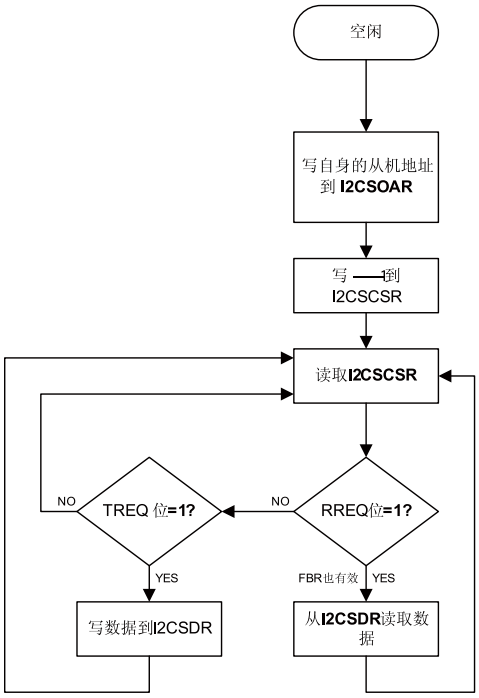
图 **14-12.** 在突发接收后主机突发发送



14.2.5.2 I²C 从机命令序列

图 14-13 在 332页说明 I²C 从机可用的命令序列。

图 **14-13.** 从机命令序列



14.3 初始化和配置

下面的例子说明了如何配置 I²C 模块作为主机来发送一个字节。这假设系统时钟为 20 MHz。

1. 通过在系统控制模块中写0x0000.1000的值到 **RCGC1** 寄存器来使能 I²C 时钟。
2. 在系统控制模块中通过 **RCGC2** 寄存器使能相应的GPIO模块的时钟。
3. 在GPIO模块中，使用 **GPIOAFSEL** 寄存器来使能相应的管脚以用于它们可选的功能。同时，确保使能相同的管脚以用于开漏操作。
4. 通过向**I2CMCR** 寄存器写0x0000.0020的值来初始化 I²C 主机。
5. 通过向**I2CMTPR**寄存器写入正确的值来设置所需的100 Kbps SCL时钟速率。写入 **I2CMTPR** 寄存器的值反映了在一个SCL时钟周期中系统时钟周期的数目。TPR值由下面的等式确定:

   ```
   TPR = (系统时钟 / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1; TPR = (20MHz / (2 * (6
   ```

   向 **I2CMTPR** 寄存器写 0x0000.0009的值。
6. 指定主机的从机地址并且下一个操作将是发送，通过向 **I2CMSA** 寄存器写 0x0000.0076的值。这设置了从机地址为 0x3B。
7. 通过向 **I2CMDR** 寄存器写入所需的数据来设置数据寄存器中准备发送的数据（字节）。
8. 通过向 **I2CMCS** 寄存器写入 0x0000.0007 的值来启动从主机到从机一个字节的数据发送（STOP, START, RUN)。
9. 等待直到传输结束，通过查询 **I2CMCS** 寄存器的 BUSBSY 位直至它已被清零。

14.4 I²C 寄存器映射

表 14-2 在 333页列出 I²C 寄存器。给出的所有地址都是相对 I²C 主机和从机的基址而言的：

- I²C 主机 0：0x4002.0000
- I²C 从机 0：0x4002.0800

表 **14-2.** 内部集成电路 **(I²C)** 接口 寄存器映射

偏移量	名称	类型	复位	描述	见页面
I²C主机					
0x000	I2CMSA	R/W	0x0000.0000	I2C 主机从地址	334
0x004	I2CMCS	R/W	0x0000.0000	I2C 主机控制/状态	335
0x008	I2CMDR	R/W	0x0000.0000	I2C 主机数据	338
0x00C	I2CMTPR	R/W	0x0000.0001	I2C 主机定时器周期	339
0x010	I2CMIMR	R/W	0x0000.0000	I2C 主机中断屏蔽	340
0x014	I2CMRIS	RO	0x0000.0000	I2C 主机原始中断状态	341
0x018	I2CMMIS	RO	0x0000.0000	I2C 主机屏蔽后的中断状态	342
0x01C	I2CMICR	WO	0x0000.0000	I2C 主机中断清除	343
0x020	I2CMCR	R/W	0x0000.0000	I2C 主机配置	344
I²C 从机					
0x000	I2CSOAR	R/W	0x0000.0000	I2C 从机自身地址	345
0x004	I2CSCSR	RO	0x0000.0000	I2C 从机控制/状态	346
0x008	I2CSDR	R/W	0x0000.0000	I2C 从机数据	348
0x00C	I2CSIMR	R/W	0x0000.0000	I2C从机中断屏蔽	349
0x010	I2CSRIS	RO	0x0000.0000	I2C 从机原始中断状态	350
0x014	I2CSMIS	RO	0x0000.0000	I2C 从机屏蔽后的中断状态	351
0x018	I2CSICR	WO	0x0000.0000	I2C 从机中断清除	352

14.5 寄存器描述（I²C 主机）

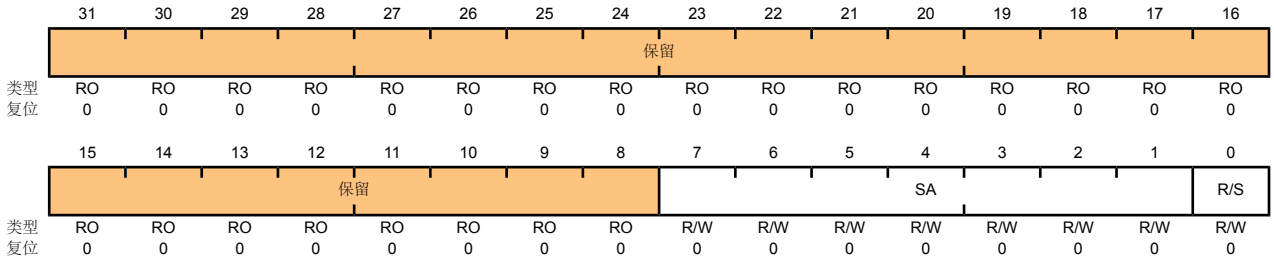
本章的以下内容将按地址偏移量的数字顺序列举并描述 I²C 主机寄存器。同见 “寄存器描述（I2C 从机）” 在 344页。

寄存器 1: I²C 主机从地址（I2CMSA），偏移量 0x000

该寄存器包括8个位：7个地址位(A6-A0)和1个接收/发送位，这个接收/发送位决定了下一个操作是接收（高电平）还是发送（低电平）。

I2C 主机从地址 (I2CMSA)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:1	SA	R/W	0	I²C 从地址 该字段表示从机地址的A6到A0。
0	R/S	R/W	0	接收/发送 R/S 位表示下一个操作是接收（高电平）还是发送（低电平）。 值 描述 0 Send. 1 Receive.

寄存器 2: I²C 主机控制/状态（I2CMCS），偏移量 0x004

对该寄存器执行写操作时访问4个控制位，执行读操作时访问7个状态位。

状态寄存器包括7个位，在执行读操作时这7个位决定了I²C 总线控制器的状态。

控制寄存器包括4个位：RUN, START, STOP和 ACK位。 START 位产生起始或重复起始条件。

STOP 位决定周期是在数据周期结束时停止，还是作为突发(burst)操作继续执行。要产生一个发送周期，就需要向 **I²C** 主机从地址（**I2CMSA**） 寄存器写入所需的地址，R/S 位设为0，并且向控制寄存器写入 ACK=X (0 或 1), STOP=1, START=1且 RUN=1 来执行操作和停止。当操作完成（或由于错误中止）时，中断管脚变为有效且数据可从**I2CMDR**寄存器中读出。当 I²C 模块在主机接收器模式下操作时，ACK位必须被设为逻辑1。这使得 I²C 总线控制器在每个字节后自动发送一个应答。当 I²C 总线控制器不再需要从机发送器发送数据时，该位必须复位。

只读状态寄存器

I2C 主机控制/状态 (I2CMCS)

偏移量 0x004
类型 RO, 复位 0x0000.0000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
域	保留															
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
域	保留									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

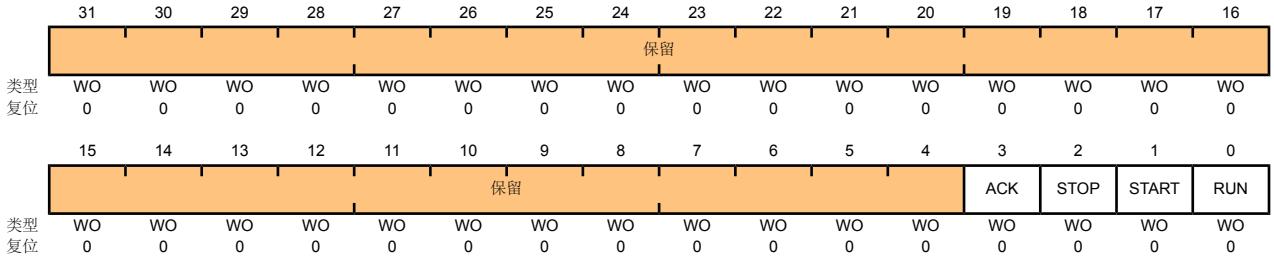
位/域	名称	类型	复位	描述
31:7	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	BUSBSY	RO	0	总线忙 该位指示 I²C 总线的状态。如果该位置位，则总线为忙状态；否则，总线为空闲状态。该位的值根据起始和停止条件来改变。
5	IDLE	RO	0	I²C 空闲 该位表示 I²C 控制器的状态。如果该位置位，则控制器为空闲；否则，控制器为非空闲。
4	ARBLST	RO	0	仲裁丢失 该位表示总线仲裁的结果。如果该位置位，则控制器丢失仲裁；否则，控制器赢得仲裁。
3	DATACK	RO	0	应答数据 该位表示上一次数据操作的结果。如果该位置位，则已发送的数据没有被应答；否则，数据被应答。
2	ADRACK	RO	0	应答地址 该位表示上一次地址操作的结果。如果该位置位，则已发送的地址没有被应答；否则，地址被应答。

位/域	名称	类型	复位	描述
1	ERROR	RO	0	错误 该位表示上一次总线操作的结果。如果该位置位，则表示上一次操作出现了错误；否则表示没有检测到错误。错误可能来自没有被应答的从机地址或没有被应答的发送数据，或者是由于控制器丢失仲裁而产生。
0	BUSY	RO	0	I²C 忙 该位表示控制器的状态。如果该位置位，则控制器为忙状态；否则，控制器为空闲状态。当 BUSY 位置位时，其它的状态位无效。

只写控制寄存器

I2C 主机控制/状态 (I2CMCS)

偏移量 0x004
类型 WO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:4	保留	WO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	ACK	WO	0	数据应答使能 当该位置位时，主机自动应答已接收的数据字节。见 表 14-3 在 337页 中的字段说明。
2	STOP	WO	0	产生停止条件 当该位置位时，产生停止条件。见 表 14-3 在 337页 中的字段说明。
1	START	WO	0	产生起始条件 当该位置位时，产生起始或重复起始条件。见 表 14-3 在 337页 中的字段说明。
0	RUN	WO	0	I²C 主机使能 当该位置位时，允许主机发送或接收数据。见 表 14-3 在 337页 中的字段说明。

表 **14-3. I2CMCS[3:0]** 字段的写操作字段说明

当前状态	I2CMSA[0]	I2CMCS[3:0]				描述
	R/S	ACK	STOP	START	RUN	
空闲	0	X[a]	0	1	1	START条件之后跟随SEND（主机进入主机发送状态）。
	0	X	1	1	1	START条件之后跟随SEND和STOP条件（主机保持在空闲状态）。
	1	0	0	1	1	START条件之后跟随带有非应答的接收操作（主机进入主机接收状态）。
	1	0	1	1	1	START条件之后跟随RECEIVE和STOP条件（主机保持在空闲状态）。
	1	1	0	1	1	START条件之后跟随 RECEIVE（主机进入主机接收状态）。
	1	1	1	1	1	非法。
	其它所有没列出的组合都不执行任何操作。					NOP
主机发送	X	X	0	0	1	发送操作（主机保持在主机发送状态）。
	X	X	1	0	0	STOP条件（主机进入空闲状态）。
	X	X	1	0	1	SEND之后跟随STOP条件（主机进入空闲状态）。
	0	X	0	1	1	重复START条件之后跟随SEND（主机保持在主机发送状态）。
	0	X	1	1	1	重复的START条件之后跟随SEND和STOP条件（主机进入空闲状态）。
	1	0	0	1	1	重复START条件之后跟随带有非应答的接收操作（主机进入主机接收状态）。
	1	0	1	1	1	重复START条件之后跟随SEND和STOP条件（主机进入空闲状态）。
	1	1	0	1	1	重复START条件之后跟随RECEIVE（主机进入主机接收状态）。
	1	1	1	1	1	非法。
	其它所有没列出的组合都不执行任何操作。					NOP
主机接收	X	0	0	0	1	带有非应答的接收操作（主机保持在主机接收状态）。
	X	X	1	0	0	STOP条件（主机进入空闲状态）。[b]
	X	0	1	0	1	RECEIVE 接收之后跟随STOP条件（主机进入空闲状态）。
	X	1	0	0	1	接收操作（主机保持在主机接收状态）。
	X	1	1	0	1	非法。
	1	0	0	1	1	重复的START条件之后跟随带有非应答的接收操作（主机保持在主机接收状态）。
	1	0	1	1	1	重复的START条件之后跟随 RECEIVE和STOP条件（主机进入空闲状态）。
	1	1	0	1	1	重复的START条件之后跟随 RECEIVE（主机保持在主机接收状态）。
	0	X	0	1	1	重复的START条件之后跟随SEND（主机进入主机发送状态）。
	0	X	1	1	1	重复的START条件之后跟随SEND和STOP条件（主机进入空闲状态）。
	其它所有没列出的组合都不执行任何操作。					NOP

a. 表格中的X说明该位可以设置成0或1。

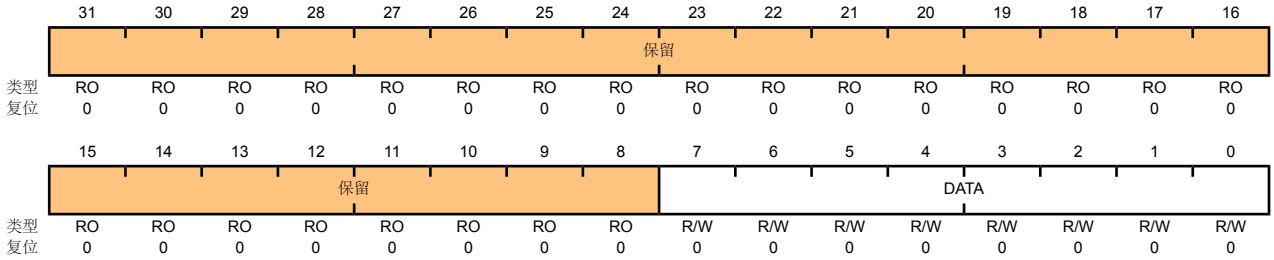
b. 在主机接收模式下，STOP条件仅在主机执行数据非应答或从机执行地址非应答后产生。

寄存器 3: I²C 主机数据（I2CMDR），偏移量 0x008

该寄存器含有在主机发送状态中准备发送的数据，以及在主机接收状态中接收到的数据。

I2C 主机数据 (I2CMDR)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



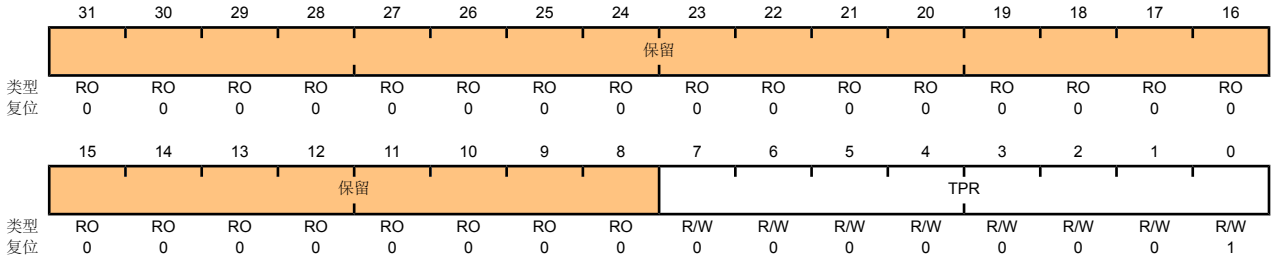
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	R/W	0x00	数据传输 操作过程中传输的数据。

寄存器 4: I²C 主机定时器周期（I2CMTPR），偏移量 0x00C

该寄存器指定SCL时钟的周期。

I2C 主机定时器周期 (I2CMTPR)

偏移量 0x00C
类型 R/W, 复位 0x0000.0001



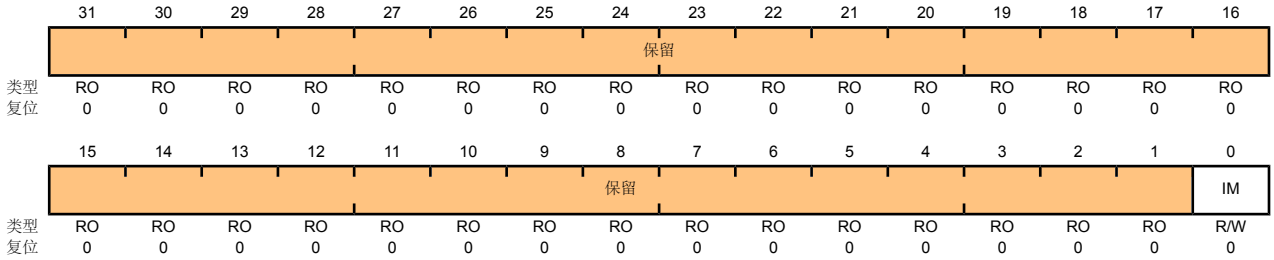
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TPR	R/W	0x1	SCL 时钟周期 该字段指定SCL时钟的周期。 `SCL_PRD = 2*(1 + TPR)*(SCL_LP + SCL_HP)*CLK_PRD` 其中: SCL_PRD 是SCL线的周期（I²C 时钟）。 TPR 是定时器周期寄存器的值（范围从1到255）。 SCL_LP 是SCL低电平周期（固定为6）。 SCL_HP 是SCL高电平周期（固定为4）。

寄存器 5: I²C 主机中断屏蔽（I2CMIMR），偏移量 0x010

该寄存器控制是否将原始中断提交到控制器中断。

I2C 主机中断屏蔽 (I2CMIMR)

偏移量 0x010
类型 R/W, 复位 0x0000.0000



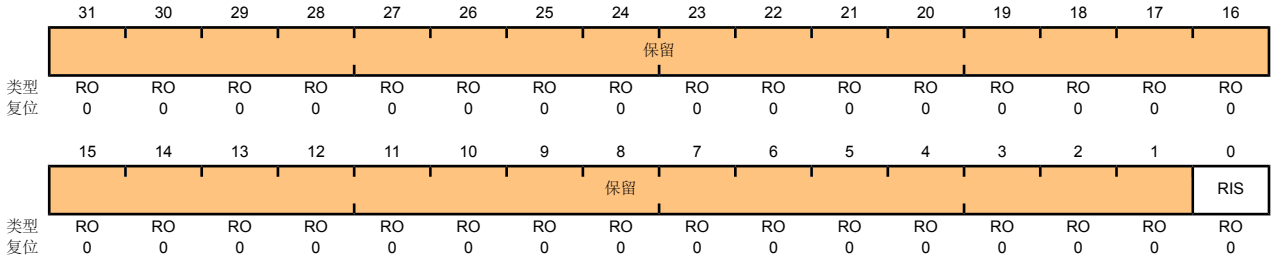
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IM	R/W	0	中断屏蔽 该位控制是否将原始中断提交到控制器中断。如果该位置位，则不屏蔽中断并且提交中断；否则，中断被屏蔽。

寄存器 6: I²C 主机原始中断状态（I2CMRIS），偏移量 0x014

该寄存器表示是否有中断正等待处理(pending)。

I2C 主机原始中断状态 (I2CMRIS)

偏移量 0x014
类型 RO, 复位 0x0000.0000



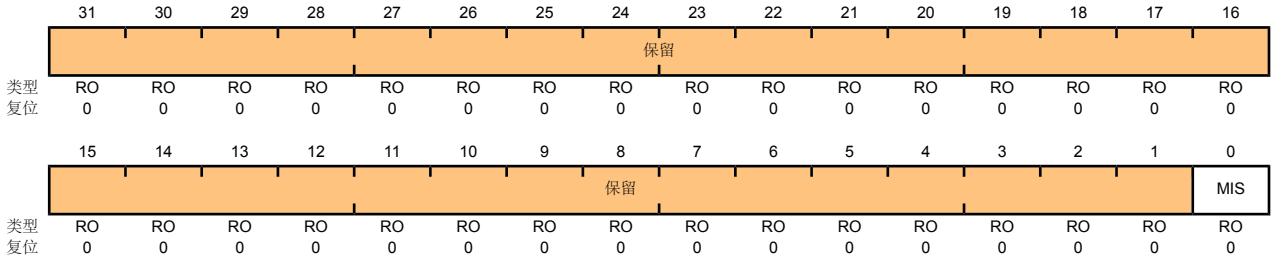
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	RIS	RO	0	原始中断状态 该位指定了 I²C 主机模块的原始中断状态（在屏蔽之前）。如果该位置位，则有中断正等待处理；否则，没有正等待处理的中断。

寄存器 7: I²C 主机屏蔽后的中断状态（I2CMMIS），偏移量 0x018

该寄存器表示是否发出中断信号。

I2C 主机屏蔽后的中断状态 (I2CMMIS)

偏移量 0x018
类型 RO, 复位 0x0000.0000

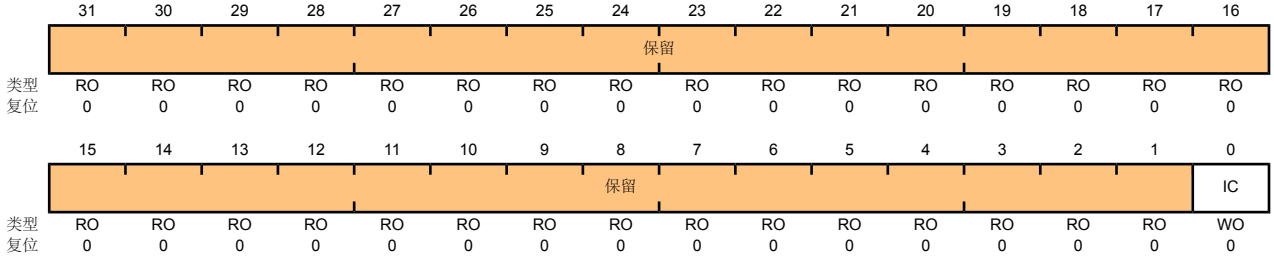


位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	MIS	RO	0	屏蔽后的中断状态 该位指定 I²C 主机模块的原始中断状态（屏蔽之后）。如果该位置位，则产生中断信号；否则，表示在该位上一次清零后没有产生中断。

寄存器 8: I²C 主机中断清除（I2CMICR），偏移量 0x01C

该寄存器清除原始中断。

I2C 主机中断清除 (I2CMICR)
偏移量 0x01C
类型 WO, 复位 0x0000.0000

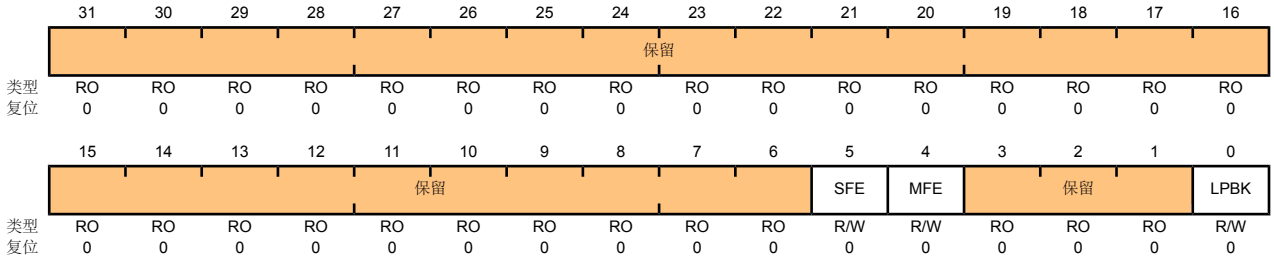


位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IC	WO	0	中断清除 该位控制原始中断的清除操作。写1清除中断；否则，写0对中断状态没有影响。读该寄存器将返回无用的数据。

寄存器 9: I²C 主机配置（I2CMCR），偏移量 0x020

该寄存器对模式（主机或从机）进行配置，并为测试模式回送(loopback)设置接口。

I2C 主机配置 (I2CMCR)
偏移量 0x020
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	SFE	R/W	0	I²C 从机功能使能 该位表示接口是否在从机模式下操作。如果该位置位，则使能从机模式；否则，禁能从机模式。
4	MFE	R/W	0	I²C 主机功能使能 该位表示接口是否在主机模式下操作。如果该位置位，则使能主机模式；否则，禁能主机模式和接口时钟。
3:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	LPBK	R/W	0	I²C 回送 该位表示接口是正常操作还是处于回送模式。如果该位置位，则设备进入测试模式回送配置中；否则，设备正常操作。

14.6 寄存器描述（I2C 从机）

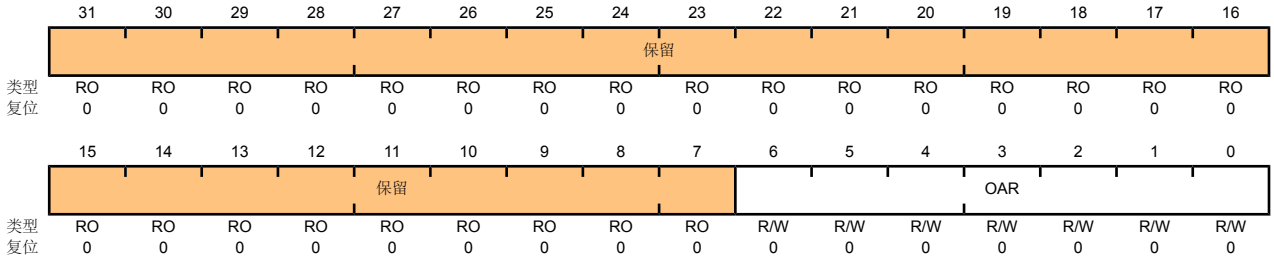
本章的以下内容将按地址偏移量的数字顺序列举并描述 I²C 从机寄存器。同见 “寄存器描述（I²C 主机）” 在 333页。

寄存器 10: I²C 从机自身地址（I2CSOAR），偏移量 0x000

该寄存器包括7个地址位，以识别 I²C 总线上的Stellaris® I²C 设备。

I2C 从机自身地址 (I2CSOAR)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:7	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6:0	OAR	R/W	0x00	I²C 从机自身地址 该字段表示从机地址的A6到A0。

寄存器 11: I²C 从机控制/状态（I2CSCSR），偏移量 0x004

对寄存器执行写操作时访问1个控制位，执行读操作时访问3个状态位。

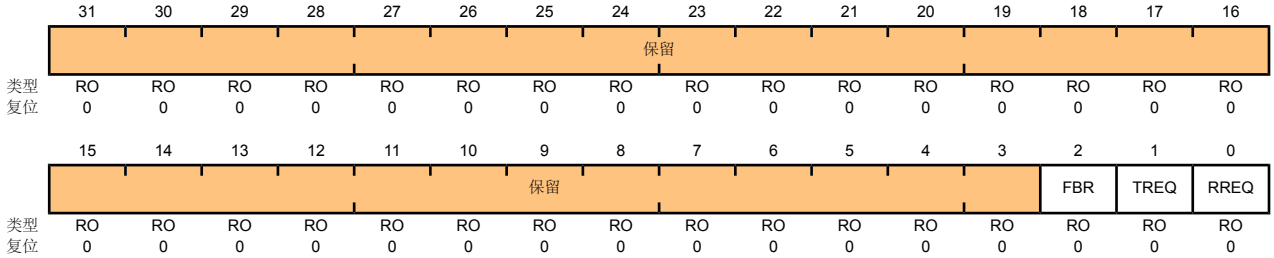
只读状态寄存器包括3个位：FBR, RREQ和TREQ 位。接收到的第一字节（FBR）位仅在Stellaris®器件检测到其自身从地址并接收到第一个来自I²C主机的数据字节之后才会置位。 接收请求（RREQ） 位表示 Stellaris® I²C 设备已接收到来自I²C 主机的数据字节。从 **I²C** 从机数据（**I2CSDR**）寄存器中读取一个数据字节来清零RREQ 位。 发送请求 （TREQ）位表示 Stellaris® I²C 设备作为从机发送器被寻址。写一个数据字节到 **I²C** 从机数据 **(I2CSDR)** 寄存器来清零 TREQ 位。

只写控制寄存器包括一个位：DA位。 DA 位使能和禁能 Stellaris® I²C 从机操作。

只读状态寄存器

I2C 从机控制/状态 (I2CSCSR)

偏移量 0x004
类型 RO, 复位 0x0000.0000

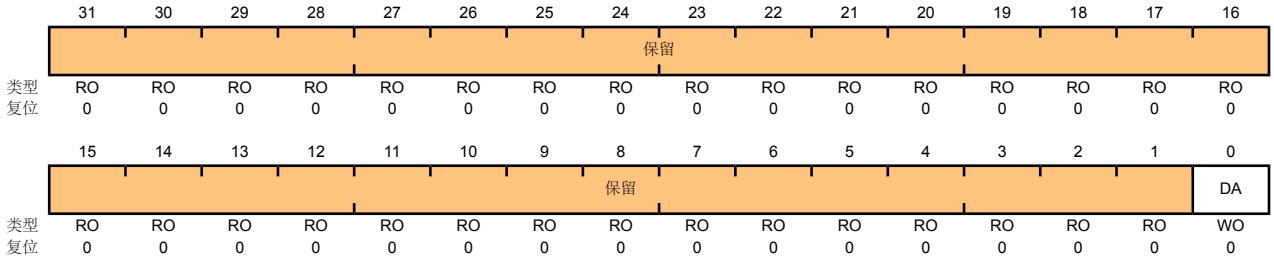


位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	FBR	RO	0	接收到的第一个字节 表示接收到的紧跟从机自身地址后的第一个字节。该位仅在 RREQ 位置位时有效，并且从**I2CSDR** 寄存器读完数据后自动清零。 注意: 该位不用于从机发送操作。
1	TREQ	RO	0	发送请求 该位表示与未处理（outstanding）的发送请求相关的 I²C从机的状态。如果该位置位，则 I²C已作为从机发送器被寻址，并使用扩展时钟将主机延时，直至数据已被写入**I2CSDR**寄存器。否则，没有未处理的发送请求。
0	RREQ	RO	0	接收请求 该位表示与未处理的接收请求相关的 I²C从机的状态。如果该位置位，则 I²C 单元含有来自 I²C 主机的未处理的接收数据，并使用扩展时钟来延时主机直至数据已从**I2CSDR** 寄存器中读取。否则，没有未处理的接收数据。

只写控制寄存器

I2C 从机控制/状态 (I2CSCSR)

偏移量 0x004
类型 WO, 复位 0x0000.0000



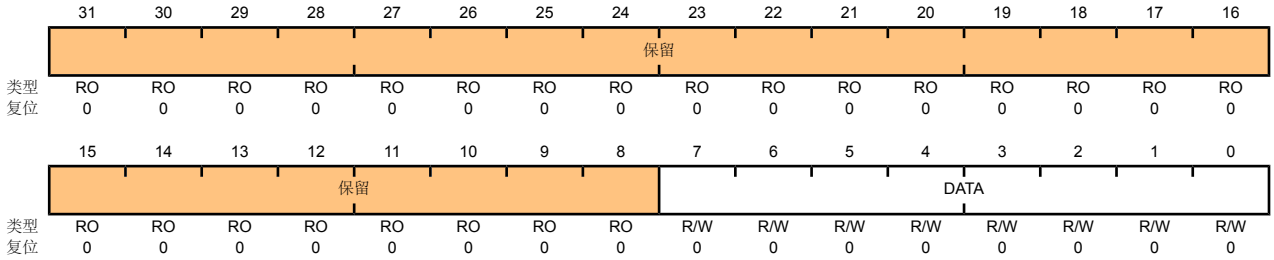
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	DA	WO	0	设备有效 值 描述 0 禁能 I^2C 从机操作。 1 使能 I^2C 从机操作。

寄存器 12: I²C 从机数据（I2CSDR），偏移量 0x008

该寄存器含有在从机发送状态中准备发送的数据，以及在从机接收状态中接收到的数据。

I2C 从机数据 (I2CSDR)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



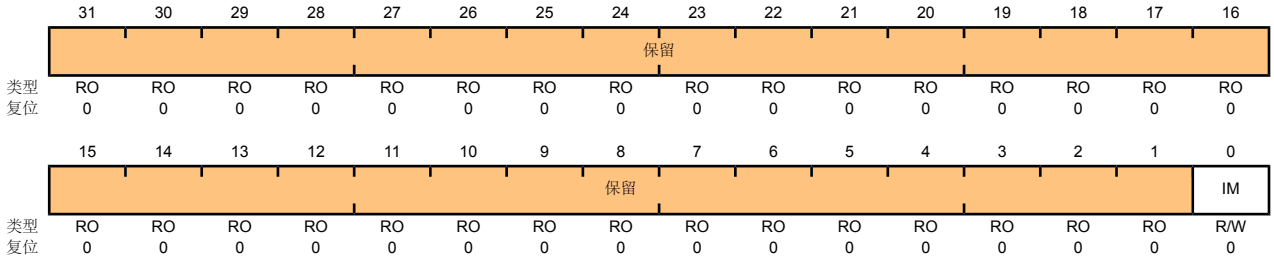
位/域	名称	类型	复位	描述
31:8	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	R/W	0x0	传输的数据 该字段包含了在从机接收或发送操作中传输的数据。

寄存器 13: I²C从机中断屏蔽（I2CSIMR），偏移量 0x00C

该寄存器控制是否将原始中断提交到控制器中断。

I2C从机中断屏蔽 (I2CSIMR)

偏移量 0x00C
类型 R/W, 复位 0x0000.0000



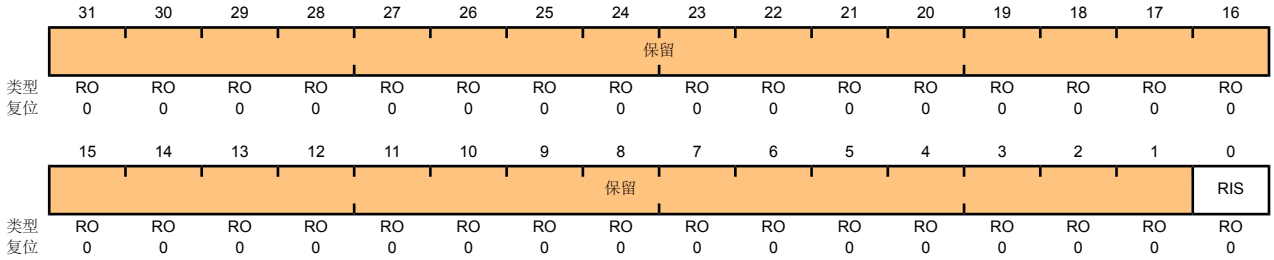
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IM	R/W	0	中断屏蔽 该位控制是否将原始中断提交到控制器中断。如果该位置位，则不屏蔽中断并且提交中断；否则，中断被屏蔽。

寄存器 14: I²C 从机原始中断状态（I2CSRIS），偏移量 0x010

该寄存器表示是否有中断正等待处理(pending)。

I2C 从机原始中断状态 (I2CSRIS)

偏移量 0x010
类型 RO, 复位 0x0000.0000



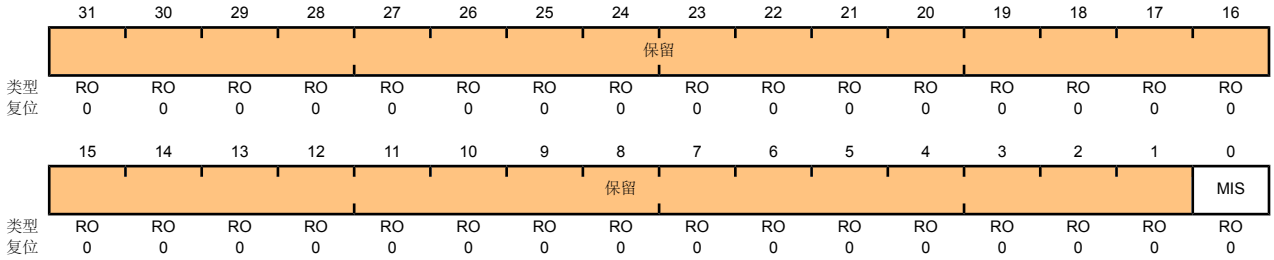
位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	RIS	RO	0	原始中断状态 该位表示 I²C 从机模块的原始中断状态（在屏蔽之前）。如果该位置位，则有中断正等待处理；否则，没有正等待处理的中断。

寄存器 15: I²C 从机屏蔽后的中断状态（I2CSMIS），偏移量 0x014

该寄存器表示是否发出中断信号。

I2C 从机屏蔽后的中断状态 (I2CSMIS)

偏移量 0x014
类型 RO, 复位 0x0000.0000

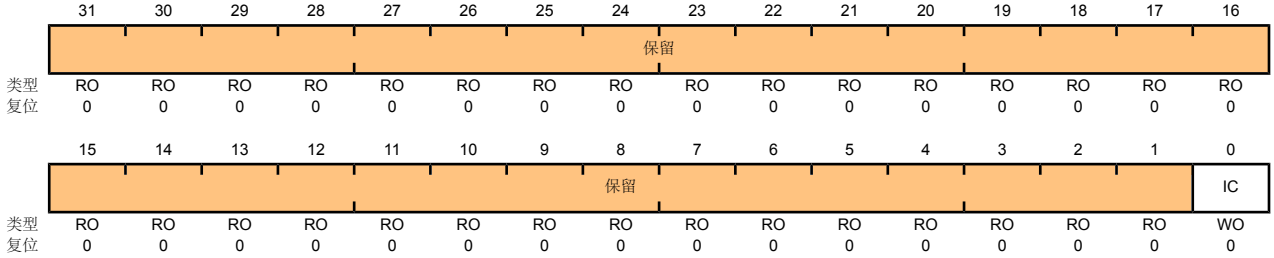


位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	MIS	RO	0	屏蔽后的中断状态 该位表示 I²C 从机模块的原始中断状态（在屏蔽之后）。 如果该位置位，则产生中断信号；否则，表示在该位上一次清零后没有产生中断。

寄存器 16: I²C 从机中断清除（I2CSICR），偏移量 0x018

该寄存器清除原始中断。

I2C 从机中断清除 (I2CSICR)
偏移量 0x018
类型 WO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IC	WO	0	清除中断 该位控制原始中断的清除操作。写1清除中断；写0对中断状态没有影响。读该寄存器将返回无用的数据。

15 控制器局域网 （CAN）模块

15.1 控制器局域网概述

控制器局域网（CAN）是一种用于连接电子控制单元（ECU）的多主站共用型串行总线标准。CAN特别适用于电磁干扰和其它电子噪声强的环境，它可以使用像RS-485这样的平衡差分线或者更稳定可靠的双绞线。CAN最初是专门面向汽车的，后来也使用在许多嵌入式控制应用中（比如：工业和医疗）。当总线长度小于40米时位速率可高达1Mbps。位速率会随着节点之间距离的增加而降低（例如：总线长度为500m时位速率为125 Kbps）。

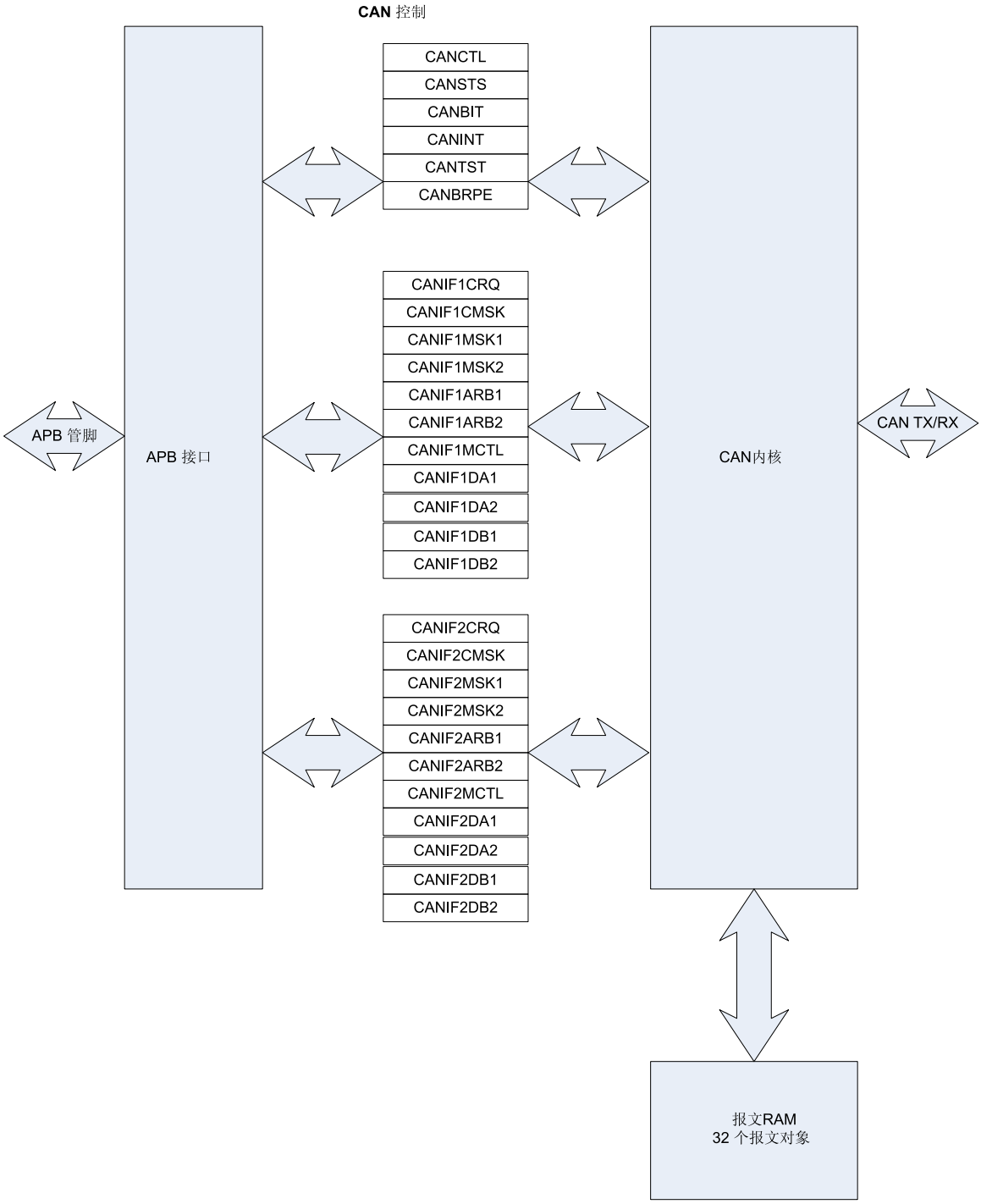
15.2 控制器局域网的特性

Stellaris® CAN 模块具有以下特性：

- CAN 协议版本 2.0 part A/B
- 位速率高达1 Mbps
- 具有32个报文对象
- 每个报文对象都具有自己的标识符屏蔽码
- 可屏蔽中断
- 针对时间触发的CAN（TTCAN）应用，可选择禁止自动重发送模式
- 自测试操作具有可编程的回送模式
- 具有可编程的 FIFO 模式
- 通过CAN0Tx和CAN0Rx管脚与外部CAN PHY无缝连接

15.3 控制器局域网的结构图

图 **15-1. CAN** 模块的结构图



15.4 控制器局域网的功能描述

CAN 模块支持 CAN 2.0 A /B协议。支持包括具有11位标识符（标准帧）或29位标识符（扩展帧）的数据帧、远程帧、错误帧以及超载帧的报文传输。传输速率可以编程为1 Mbps。

CAN 模块主要由3个部件组成：

- CAN协议控制器和报文处理器
- 报文存储器
- CAN寄存器接口

协议控制器从CAN总线传输和接收串行数据，并将数据传递到报文处理器。接着，报文处理器根据当前的滤波和报文对象存储器中的标识符，将该信息载入合适的报文对象。报文处理器还负责根据CAN总线上的事件来产生中断。

报文对象存储器由32个相同的存储块组成，这些存储块保存了每个报文对象当前的配置信息、状态和实际数据。我们可以通过CAN报文对象寄存器接口来访问它们。由于不能通过Stellaris®存储器映射直接访问报文存储器，Stellaris® CAN 控制器会提供一个接口来与报文存储器通信。

CAN 报文对象寄存器接口提供了两个寄存器组来与报文对象通信。由于不能直接访问报文对象存储器，所以必须使用这两个接口来读写各个报文对象。当多个对象包含需要处理的新信息时，这两个报文对象接口允许并行访问CAN控制器报文对象。

15.4.1 初始化

软件初始化在发送器的错误计数超过255时发生，我们可以通过置位**CAN** 控制 **(CANCTL)** 寄存器中的`INIT`位、软件或硬件复位，或通过脱离总线来启动它。在`INIT` 置位时, 所有CAN总线的报文传输都被中止，而且CAN发送输出的状态为隐性电平（逻辑1）。进入初始化状态并不会改变CAN控制器、报文对象或错误计数器的配置。但是，某些配置寄存器我们只能在初始化状态时才可访问。

为了初始化CAN控制器，应该设置 **CAN** 位定时 **(CANBIT)** 寄存器并对每个报文对象进行配置。如果不需要某个报文对象，那么只需通过清零**CANIFnARB2**寄存器中的`MsgVal`位将它设置成无效就足够了。否则，整个报文对象必须被初始化，因为报文对象的场（fields）可能包含引起意外结果的无效信息。当**CANCTL**寄存器中的 `INIT` 和 `CCE` 位置位时，通过访问 **CAN** 位定时 **(CANBIT)** 寄存器和 **CAN** 波特率预分频器扩展 **(CANBRPE)** 寄存器来配置位定时被使能。如果想退出初始化状态，必须清零 `INIT` 位。然后，内部位流处理器（BSP）在它参与总线动作和启动报文传输前会等待11个连续隐性位（总线空闲）序列的出现，以便让自己同步于CAN总线上的数据传输。报文对象的初始化独立于初始化状态，并且可以在不工作时（on the fly）完成，但是在BSP启动报文传输前报文对象应该全部配置成特定的标志符或设置成无效。要想在正常工作期间改变报文对象的配置，可以将**CANIFnARB2**寄存器中的 `MsgVal` 位设为0 （无效）。当配置完成时, `MsgVal`再次被设为1 (有效)。

15.4.2 操作

一旦CAN模块被初始化，并且**CANCTL** 寄存器中的`INIT`位重新设为0，CAN模块自身将同步于CAN总线，并启动报文传输。在接收报文时，如果报文通过了报文处理器的滤波，就会存储在它们相应的报文对象中。整个报文（包括所有仲裁位、数据长度码和8个数据字节）都存储在报文对象中。如果使用了标识符屏蔽位(**CANIFnMSKn** 寄存器中的`Msk` 位），那么在报文对象中可能会覆盖被屏蔽为“无关”的仲裁位。

CPU通过CAN接口寄存器(**CANIFnCRQ**、 **CANIFnCMSK**、 **CANIFnMSKn**、**CANIFnARBn**、**CANIFnMCTL**、 **CANIFnDAn**和 **CANIFnDBn**)可以在任意时刻读写每个报文。报文处理器保证在出现同时访问的情况下数据的一致性。

报文对象的发送受管理CAN硬件的软件的控制。这些可以是用来一次数据传输的报文对象，也可以是以多周期方式响应的永久性报文对象。永久性报文对象设置了所有仲裁和控制，并且只更新数据

字节。为启动发送，**CANTXRQn** 寄存器中的TxRqst 位和**CANNWDAn** 寄存器中的NewDat 位必须置位。如果多个发送报文被分配给了同一个报文对象（在报文对象不够时），整个报文对象必须在请求发送前被配置。

同一时刻可以请求发送任意数量的报文对象；它们根据内部的优先级进行发送，其优先级基于报文对象的报文标识符。报文可以在任意时刻被更新或者设置成无效，即使是在它们请求的发送仍然被挂起时。当报文在其挂起发送启动前被更新时，旧的数据会被丢弃。根据报文对象的配置情况，接收含匹配标识符的远程帧会自动请求发送报文。

有两组CAN接口寄存器 (**CANIF1x**和**CANIF2x**) 被用来访问报文RAM中的报文对象。 CAN控制器将传输到报文RAM和从报文RAM那里传输调整成传输到该寄存器和从该寄存器那里传输。这两组寄存器的功能是独立且相同的，并且可以用来排队等待处理。

15.4.3 发送报文对象

如果CAN模块的内部发送移位寄存器准备装载，并且如果CAN接口寄存器和报文RAM之间无数据传输，那么优先级最高并且含有挂起发送请求的有效报文对象将被报文处理器载入发送移位寄存器，然后开始发送。报文对象的 NewDat 位被复位，并且可以在 **CANNWDAn** 寄存器中查看其状态。在成功发送后，如果自开始发送起就没有新数据写入报文对象，那么**CANIFnMCTL** 寄存器中的TxRqst 位将被复位。如果**CANIFnMCTL** 寄存器 中的 TxIE 位置位，那么**CANIFnMCTL** 寄存器中的 IntPnd 位会在成功发送后置位。如果CAN模块丢失了仲裁或者如果在发送期间发生错误，那么一旦CAN总线再次空闲就会重新发送报文。如果与此同时，优先级最高的报文发送发出了请求，那么报文将按照它们的优先级顺序进行发送。

15.4.4 配置发送报文对象

表 15-1 在 356页规定了接收报文对象的位的设置。

表 **15-1.** 发送信息对象的位设置

寄存器	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
位	MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
值	1	appl	appl	appl	1	1	0	0	0	appl	0	appl	0

CANIFnARBn 寄存器中的Xtd 和 ID 位域都被应用程序置位。它们定义标识符和待发报文的类型。如果使用了一个11位标识符 (标准帧) , 该标识符将会被设置为**CANIFnARB1**的 [28:18] 位域，**CANIFnARBn** 的17:0位不用。

如果 TxIE 位置位, IntPnd 位在成功发送报文对象后会置位。

如果RmtEn 位置位，匹配的接收远程帧将会使得TxRqst 位置位，并且含有报文对象数据的数据帧会自动应答远程帧。

CANIFnMCTL 寄存器中的DLC 位被应用程序置位。TxRqst 和 RmtEn 在数据有效之前可能不会置位。

CAN屏蔽寄存器 (**CANIFnMSKn**中的Msk 位 , **CANIFnMCTL**寄存器中的UMask 位，以及**CANIFnMSK2** 寄存器中的 MXtd位 和 MDir 位) 可用来 (UMask=1) 允许带相似标识符的远程帧组将 TxRqst位置位。Dir 位不应该被屏蔽。

15.4.5 更新发送报文对象

CPU在任意时刻都可以通过CAN接口寄存器来更新发送报文对象的数据字节，并且 MsgVal 位和 TxRqst 位在更新前都不必复位。

即使只是将一部分数据字节更新，相应的 **CANIFnDAn**或**CANIFnDBn** 寄存器的4个字节都必须在寄存器内容被传输到报文对象之前有效。在CPU写入新数据字节之前，不是 CPU 将4个字节全部写入

CANIFnDAn 或 **CANIFnDBn** 寄存器，就是将报文对象传输到 **CANIFnDAn** 或 **CANIFnDBn** 寄存器 。

若只是想更新报文对象中的数据，就将WR、NewDat、DataA和DataB位都写入 **CAN IFn** 命令屏蔽 **(CANIFnMSKn)**寄存器，然后写**CAN IFn Data**寄存器，接着将报文对象的数目写入**CAN IFn**命令请求 **(CANIFnCRQ)**寄存器, 这样就同时更新了数据字节和TxRqst位。

在数据更新时，为了防止 TxRqst在已进行的发送的末尾复位，NewDat必须和 TxRqst一起置位。当 NewDat 和 TxRqst一起置位时，一旦开始新的发送NewDat 就会复位。

15.4.6 接受接收的报文对象

当到来的报文其仲裁场和控制场 (ID + Xtd + RmtEn + DLC) 完全移入CAN 模块时, 模块具有报文处理功能，开始对报文RAM进行扫描，以找到匹配的有效报文对象。在通过扫描报文RAM来获取匹配的报文对象时，接收过滤单元从内核装载仲裁位。接着报文对象1的仲裁和屏蔽场（包括 MsgVal、UMask、 NewDat和 EoB) 被装载到接收过滤单元，并与移位寄存器的仲裁场进行比较。后面的报文对象也依此进行，直到发现匹配的报文对象或到达报文RAM的末端。如果匹配出现，则停止扫描，然后报文处理器将根据接收帧的类型进行处理。

15.4.7 接收数据帧

报文处理器将来自CAN模块接收移位寄存器的报文存储到报文RAM中相应的报文对象中。它将数据字节、所有仲裁位和数据长度码存储到相应的报文对象中。即使使用了仲裁屏蔽寄存器，也可以通过执行这些动作来让数据字节和标识符保持连接。 CANIFnMCTL.NewDat 位置位，表明接收到了新数据。在读取报文对象时，CPU应该将CANIFnMCTL.NewDat复位 ，以向控制器表明已经接收了报文，并且缓冲器可以接收更多报文。如果CAN控制器接收了报文并且 CANIFnMCTL.NewDat 位已经置位, 那么 MsgLst 位将会置位，以表明之前的数据已丢失。如果 CANIFnMCTL.RxIE 位置位，CANIFnMCTL.IntPnd 位也会置位， **CANINT**中断寄存器因而会指向正在接收报文的报文对象。在刚刚接收请求数据帧时，该报文对象的 CANIFnMCTL.TxRqst 位复位，以便阻止发送远程帧。

15.4.8 接收远程帧

在接收远程帧时，必须考虑匹配报文对象的三种不同配置：

- Dir = 1 (方向 =发送), RmtEn = 1, UMask = 1 或0

  在接收到匹配的远程帧时，该报文对象的TxRqst 位会置位。剩余的报文对象保持不变。

- Dir = 1 (方向 = 发送), RmtEn = 0, UMask = 0

  在接收到匹配的远程帧时，该报文对象的 TxRqst 位保持不变；远程帧被忽略。这个远程帧被禁止并且不能自动应答或表明这个远程帧曾经出现过。

- Dir = 1 (方向 = 发送), RmtEn = 0, UMask = 1

  在接收到匹配的远程帧时，该报文对象的 TxRqst 位复位。移位寄存器的仲裁场和控制场 (ID + Xtd + RmtEn + DLC) 被存储到报文RAM中的报文对象中，并且该报文对象的 NewDat 位将置位。报文对象的数据场保持不变；远程帧像接收的数据帧那样被处理。这对于来自另一CAN器件的远程数据请求来说非常有用，因为Stellaris®控制器没有包含可用的数据。软件必须填充数据并人工响应帧。

15.4.9 接收/发送优先级

报文对象的接收/发送优先级由报文编号决定。报文对象1的优先级最高，报文对象32的优先级最低。如果有超过1个发送请求被挂起，那么将按顺序发送报文对象，报文编号最低的报文对象最先发送。这不能和报文标识符相混淆，因为优先级是由CAN总线强加的。这就意味着，如果报文对象1和报文

对象2都含有需要发送的有效报文，那么将首先发送报文对象1，而不用考虑信息报文本身的报文标识符。

15.4.10 配置接收报文对象

表 15-2 在 358页规定了接收报文对象的位的设置。

表 **15-2.** 接收报文对象的位设置

寄存器	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
位	MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
值	1	appl	appl	appl	1	0	0	0	appl	0	0	0	0

CANIFnARBn 寄存器中的Xtd 和 ID 位域都被应用程序置位。它们定义了标识符和所接受接收报文的类型。如果使用了11位标识符 (标准帧) ，它将被编程为**CANIFnARB1**的位[28:18]，并且位[17:0]被CAN控制器忽略。当接收到一个含11位标识符的数据帧时，位[17:0]被设为0。

如果 RxIE 位置位，那么 IntPnd 位将在接收数据帧被接受并被存储到报文对象中时置位。

当报文处理器将数据帧存储在报文对象中时，它存储了接收数据长度码和8个数据字节。如果数据长度码小于8，那么报文对象剩余的字节将被非特定值覆盖。

CAN屏蔽寄存器 (**CANIFnMSKn**中的Msk 位, **CANIFnMCTL**寄存器中的UMask 位，以及**CANIFnMSK2** 寄存器中的 MXtd 位和 MDir 位) 可用来 (UMask=1) 允许接收带有相似标识符的数据帧组。典型应用中不应该将 Dir 位屏蔽。

15.4.11 处理接收报文对象

因为报文处理器状态机保证了数据的一致性，所以 CPU 在任意时刻都可以通过CAN接口寄存器读取接收报文。

通常，CPU首先会向**CAN IFn** 命令屏蔽 **(CANIFnCMSK)** 寄存器写入 0x007F，然后向**CAN IFn** 命令请求 **(CANIFnCRQ)** 寄存器写入报文对象的编号 。那个组合将整个接收报文从报文RAM传输到报文缓冲寄存器 (**CANIFnMSKn**, **CANIFnARBn**和 **CANIFnMCTL**)。另外， NewDat 和 IntPnd 位在报文RAM中清零，确定已读取报文，并清除了该报文对象产生的挂起中断。

如果报文对象在接收过滤中使用屏蔽，那么仲裁位会指示接收的是哪个匹配报文。

NewDat 的实际值指示自上次读取该报文对象起是否接收了新报文。MsgLst 的实际值指示自上次读取该报文对象起是否接收了多于1个报文。MsgLst 不会自动复位。

通过使用远程帧，CPU可以从CAN总线上的另一CAN节点处请求新数据。置位接收对象的 TxRqst 位会启动发送带接收对象标识符的远程帧。远程帧触发其它CAN节点启动发送匹配数据帧。如果在可以发送远程帧之前接收到匹配数据帧，那么 TxRqst 位会自动复位。这样，当CAN总线上的其它器件比预期的要早发送数据时就不会丢失数据了。

15.4.12 中断处理

如果多个中断被挂起，**CAN** 中断 **(CANINT)** 寄存器将指向优先级最高的挂起中断，而不用考虑它们的时间顺序如何。中断会一直挂起，直至CPU将它清除。

状态中断的优先级最高。在报文中断之间，报文对象的中断优先级随报文编号的升高而降低。报文中断可以通过清零报文对象的IntPnd 位被清除。状态中断可以通过读取 **CAN** 状态**(CANSTS)** 寄存器被清除。

CANINT 寄存器中的中断标识符 IntId 指明了中断的原因。在没有挂起中断时，寄存器将该值保持为0。如果**CANINT** 的值不为0，那么就有中断被挂起。如果IE 位在 **CANCTL** 寄存器中置位，那么

CPU的中断线是激活的。中断线一直保持激活，直到 **CANINT** 为 0、所有中断源都被清除、 (中断原因复位)，或直到IE复位（禁止来自CAN控制器的中断）。

因为CAN模块已更新，所以**CANINT**寄存器的值为0x8000 时表示有中断被挂起，但是不必改变**CANSTS**寄存器 (错误中断或状态中断)。这表明出现一个新错误中断或一个新状态中断。写访问可以清零**CANSTS**寄存器中的RxOK、TxOK和LEC标志， 但只有通过读取**CANSTS**寄存器才能清除状态中断源。

IntId 指向优先级最高的挂起报文中断。**CANCTL** 寄存器中的 SIE 位决定状态寄存器的改变是否会引发中断。**CANCTL** 寄存器中的 EIE 位决定CAN控制器的任一中断是否真的会向微控制器中断控制器产生中断。 即使当 IE 位被置为0，**CANINT** 中断寄存器也会被更新。

在处理报文中断源时有两种可能。第一个可能是通过读取**CANINT** 中断寄存器中的 IntId 位来决定挂起优先级最高的中断；第二个可能是通过读取 **CAN** 报文中断挂起 **(CANMSGnINT)** 寄存器来查看所有含挂起中断的报文对象。

通过读取属于中断源的报文，中断服务程序可以读取报文同时还可以通过置位**CAN IFn** 命令屏蔽 **(CANIFnCMSK)** 寄存器中的ClrIntPnd 位来复位报文对象的 IntPnd。 当 IntPnd 位被清零时，**CANINT** 寄存器将包含下一个带挂起中断的报文对象的报文编号。

15.4.13 位定时配置错误的注意事项

即使CAN的位定时配置中的细微错误不会立即造成故障，但也会大大地降低CAN网络的性能。在许多情况下，CAN位同步会修改CAN位定时的错误配置，使之控制在只会偶然产生错误帧。但是，在仲裁时，当两个或两个以上CAN节点同时试图发送帧时，采样点位置不当可能会使得其中一个发送器变成错误认可（error passive）状态。对于这种偶发错误的分析，必须要详细了解CAN节点内的CAN位同步以及CAN节点对CAN总线的相互作用。

15.4.14 位时间和位速率

CAN系统支持的位速率范围：1 Kbps～1000 Kbps。CAN网络的每个成员都有自己的时钟发生器。即使CAN节点振荡器的周期可能不同，但对于每个CAN节点来说，位时间的定时参数都可以单独配置，产生一个共同的位速率。

由于温度或电压的变化以及元件的损耗会引起频率发生小变动，所以这些振荡器不会绝对地稳定。但只要这些变动保持在振荡器特定的容忍范围，CAN节点就可以通过周期性与位流同步来补偿不同的位速率。

根据CAN规范，位时间被分成4个时间段(见 图 15-2 在 360页)：同步段、传播时间段、相位缓冲段1和相位缓冲段2。每个段由具体、可编程数量的时间份额（time quanta）组成 (见 表 15-3 在 360页)。时间份额是位时间的基本时间单元，它的长度 (tq)由CAN控制器的系统时钟(fsys)和波特率预分频器(BRP)定义：

```
tq = BRP / fsys
```

CAN模块的系统时钟fsys是其CAN模块时钟 (CAN_CLK)输入的频率。

同步段Sync_Seg 是位时间的一部分，在此段内期望有一个CAN总线电平边沿出现；如果边沿出现在 Sync_Seg 之外，那么它与Sync_Seg 之间的长度叫做沿 *相位误差* 。

传播时间段 Prop_Seg 用于补偿CAN网络内部的物理延迟时间。

相位缓冲段 Phase_Seg1 和 Phase_Seg2 包围了采样点。

(重)同步跳转宽度（SJW）决定重同步会将采样点移动多远，移动距离的上限由用于补偿沿相位误差的相位缓冲段给定。

通过不同的位时间配置可以得到指定的位速率，但是为了CAN网络可以正常的工作，必须考虑物理延迟时间和振荡器的容限。

图 **15-2. CAN** 的位时间

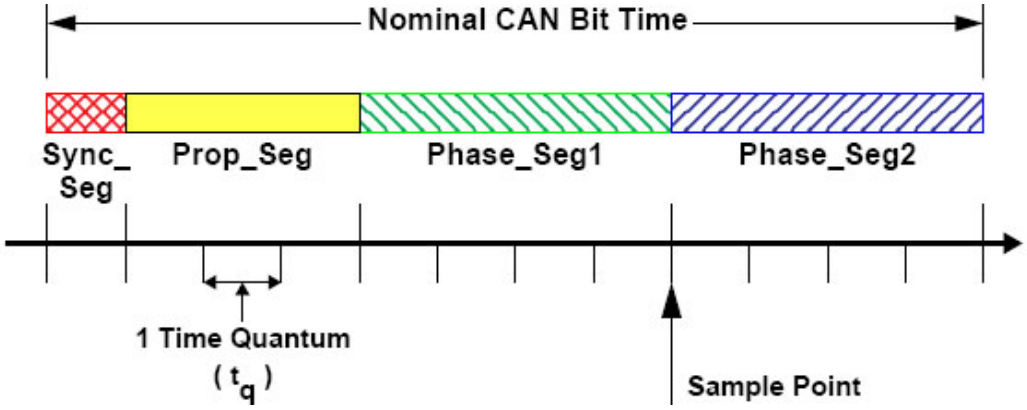


表 **15-3. CAN** 协议范围[a]

参数	范围	说明
BRP	[1 .. 32]	定义时间份额的长度 t_q
Sync_Seg	1 t_q	固定长度，总线输入与系统时钟同步
Prop_Seg	[1 .. 8] t_q	补偿物理延迟时间
Phase_Seg1	[1 .. 8] t_q	可通过同步暂时延长
Phase_Seg2	[1 .. 8] t_q	可通过同步暂时缩短
SJW	[1 .. 4] t_q	不能比任一相位缓冲段长

a. 该表描述了CAN协议要求的最小可编程范围。

位定时配置的编程是由**CANBIT** 寄存器中的2个寄存器字节来完成的。 Prop_Seg 与Phase_Seg1 的和(作为 TSEG1) 与 Phase_Seg2 (作为 TSEG2) 组合成1个字节, 而 SJW 和 BRP 组合成另一个字节。

在这些位定时寄存器中， TSEG1、TSEG2、SJW和 BRP四个位域必须编程为一个小于其函数值的数字值；所以其值不属于[1..n]范围，而属于[0..n-1]范围。那样的话，例如：SJW ([1..4]的函数范围)只用两个位来表示。因此，位时间的长度是（编程值）：

```
[TSEG1 + TSEG2 + 3] tq
```

或 (函数值):

```
[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq
```

位定时寄存器中的数据是CAN协议控制器的配置输入。波特率预分频器(由BRP配置) 决定时间份额（位时间的基本时间单元）的长度；位定时逻辑 (由TSEG1、 TSEG2和 SJW配置) 决定位时间内时间份额的数目。

位时间的处理、采样点位置的计算以及偶然同步都由CAN控制器控制，并且每个时间份额估算一次。

CAN控制器可以将报文翻译成帧，也可以将帧翻译成报文。它产生并丢弃附着的固定格式位，插入和提取填充位，计算和检查CRC代码，执行错误管理，并决定使用那种类型的同步。它在采样点处估算，并处理采样的总线输入位。由采样点开始，用于计算下一个将要发送位(即数据位、 CRC 位、填充位、错误标志或空闲位) 的时间叫做信息处理时间 (IPT)。

IPT 根据应用程序的不同而不同， 但不会长于 2 tq；CAN的IPT是0 tq。其长度是Phase_Seg2编程长度的下限。在同步情况下， Phase_Seg2 可能会缩短成小于IPT，但这并不会影响总线定时。

15.4.15 计算位定时参数

通常，位时序配置的计算从目标位速率或位时间开始。作为结果的位时间 (1/位速率) 必须是系统时钟周期的整数倍。

位时间可由4～25个时间份额组成。通过不同的组合可得到目标位时间，允许重复以下步骤。

要定义的第一部分位时间是 Prop_Seg。其长度视系统测量的延迟时间而定。必须为可扩展的CAN总线系统定义最大的总线长度和最大的节点延迟。 Prop_Seg 的结果时间被转换成时间份额(四舍五入成最接近 tq的整数倍)。

Sync_Seg 是1 tq 长（固定的）， 两个相位缓冲段为 (位时间 - Prop_Seg - 1) tq 。如果剩余的tq是偶数，那么相位缓冲段的长度相同，即Phase_Seg2 = Phase_Seg1, 否则 Phase_Seg2 = Phase_Seg1 + 1。

还必须考虑 Phase_Seg2 的最小额定长度。Phase_Seg2 不能比CAN控制器的信息处理时间短，在 [0..2] tq范围内，视实际的执行情况而定。

同步跳转宽度的长度被设置为最大值，是4和 Phase_Seg1之中的最小值。

结果配置所需的振荡器容限范围通过以下算式计算得到：

```
(1 -df) x fnom <= fosc <= (1+ df) x fnom
```

此处，

- df = 振荡器频率的最大极限值
- fosc = 实际的振荡器频率
- fnom = 额定的振荡器频率

最大频率范围必须考虑以下等式：

```
df <= (Phase_Seg1,Phase_Seg2)min/ 2 x (13 x tbit - Phase_Seg2)
dfmax = 2 x df x fnom
```

此处，

- Phase_Seg1 和 Phase_Seg2 取自表 15-3 在 360页
- tbit = 位时间
- dfmax = 两个振荡器之间的最大差值

如果可以包含一个以上的配置，那么该配置允许选择最高的振荡器容限范围。

含不同系统时钟的CAN节点要求不同配置的位速率相同。在CAN网络中会对整个网络计算一次传播时间，这个时间与最长延迟时间的节点有关。

CAN 系统的振荡器容错范围受最低容错范围的节点限制。

这个计算表明了这样一种事实：为了找到一种协议兼容的CAN位定时配置，必须缩短总线宽度或降低位速率，抑或增加振荡器频率的稳定性。

将结果配置写入 **CAN** 位定时**(CANBIT)**寄存器：

```
(Phase_Seg2-1)&(Phase_Seg1+Prop_Seg-1)&(同步跳转宽度-1)&(预分频-1)
```

15.4.15.1 范例：高波特率的位定时

在这个实例中， CAN_CLK 的频率为10 MHz, BRP 为 0, 而位速率为1 Mbps。

```
tq 100 ns = tCAN_CLK
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 600 ns = 6 × tq
tSJW 100 ns = 1 × tq
tTSeg1 700 ns = tProp + tSJW
tTSeg2 200 ns = Information Processing Time + 1 × tq
tSync-Seg 100 ns = 1 × tq
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2
tolerance for CAN_CLK 0.39 % =
   min(PB1,PB2)/ 2 × (13 x bit time - PB2) =
   0.1us/ 2 x (13x 1us - 2us)
```

在上述实例中，串联的位时间参数是(2-1)3&(7-1)4&(1-1)2&(1-1)6，并且**CANBIT**被编程为0x1600。

15.4.15.2 范例：低波特率的位定时

在该实例中，CAN_CLK 的频率为2 MHz, BRP 为 1,位速率为100 Kbps。

```
tq 1 ms = 2 × tCAN_CLK delay of bus driver 200 ns delay of receiver circuit 80 ns d
```

在该实例中，串联的位时间参数是(4-1)3&(5-1)4&(4-1)2&(2-1)6，并且**CANBIT**被编程为0x34C1。

15.5 控制器局域网络的寄存器映射

表 15-4 在 362页列出了寄存器。所有给出的地址都是相对于CAN基址的：

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000

所有访问都是以字（32位）为边界。

表 **15-4. CAN** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	CANCTL	R/W	0x0000.0001	CAN 控制	364
0x004	CANSTS	R/W	0x0000.0000	CAN 状态	366
0x008	CANERR	RO	0x0000.0000	CAN 错误计数器	368
0x00C	CANBIT	R/W	0x0000.2301	CAN位定时	369
0x010	CANINT	RO	0x0000.0000	CAN 中断	370
0x014	CANTST	R/W	0x0000.0000	CAN 测试	371
0x018	CANBRPE	R/W	0x0000.0000	CAN波特率预分频扩展	372
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1命令请求	373
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 命令屏蔽	374

偏移量	名称	类型	复位	描述	见页面
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 屏蔽 1	377
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1屏蔽2	378
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 仲裁 1	379
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 仲裁 2	380
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 报文控制	381
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 数据 A1	383
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 数据 A2	383
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 数据 B1	383
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 数据 B2	383
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 命令请求	373
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 命令屏蔽	374
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 屏蔽1	377
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2屏蔽2	378
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 仲裁 1	379
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 仲裁 2	380
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 报文控制	381
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 数据 A1	383
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 数据 A2	383
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 数据 B1	383
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 数据 B2	383
0x100	CANTXRQ1	RO	0x0000.0000	CAN发送请求1	384
0x104	CANTXRQ2	RO	0x0000.0000	CAN 发送请求2	384
0x120	CANNWDA1	RO	0x0000.0000	CAN新数据1	385
0x124	CANNWDA2	RO	0x0000.0000	CAN新数据2	385
0x140	CANMSG1INT	RO	0x0000.0000	CAN报文1中断挂起	386
0x144	CANMSG2INT	RO	0x0000.0000	CAN报文2中断挂起	386
0x160	CANMSG1VAL	RO	0x0000.0000	CAN报文1有效	387
0x164	CANMSG2VAL	RO	0x0000.0000	CAN 报文2有效	387

15.6 寄存器描述

下文按照地址偏移量的数字顺序列出并描述了CAN寄存器。**CANIF1x** 和 **CANIF2x**这两组接口寄存器用于访问报文RAM中的报文对象 。这两组寄存器的功能是相同的，并且都用来排队等待处理。

寄存器 **1: CAN** 控制（**CANCTL**）寄存器，偏移量 **0x000**

该控制寄存器用来初始化模块并使能测试模式和中断。

脱离总线恢复序列（见CAN 规范版本）。2.0) 不能通过置位或复位`INIT`被缩短。如果器件脱离了总线，它将置位`INIT`，中止所有总线动作。一旦CPU将`INIT`清零后，器件在恢复正常工作之前会等待出现129 个总线空闲(129 * 11个连续高电平)。在脱离总线恢复序列的末端，错误管理计数器复位。

在`INIT`位复位后的等待期间，每次都要监视一个连续为11个高电平的序列， 将`Bit0Error`码写入**CANSTS**状态寄存器，使得 CPU 不但可以轻易地检测 CAN 总线总是为显性状态还是持续被打乱，还可以对脱离总线恢复序列的过程进行监控。

CAN 控制 (CANCTL)

偏移量 0x000
类型 R/W, 复位 0x0000.0001

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	保留															
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	保留								Test	CCE	DAR	保留	EIE	SIE	IE	INIT
类型	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

位/域	名称	类型	复位	描述
31:8	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	Test	R/W	0	测试模式使能 0: 正常操作 1: 测试模式
6	CCE	R/W	0	配置改变使能 0: 不允许对 **CANBIT** 寄存器进行写访问。 1: 如果`INIT` 位为1，则允许对**CANBIT**寄存器进行写访问。
5	DAR	R/W	0	禁止自动重发送 0: 允许自动重发送被打乱的信息。 1: 禁止自动重发送。
4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	EIE	R/W	0	错误中断使能 0: 禁止不会产生错误状态中断。 1: 使能**CANSTS**寄存器中 `Boff`位或 `EWarn` 位的改变会产生中断。
2	SIE	R/W	0	状态改变中断使能 0: 禁止不会产生状态改变中断。 1: 使能中断在报文被成功发送或接收后，或者在检测到CAN总线错误后产生。**CANSTS** 寄存器中`TxOk` 位或 `RxOk` 位的改变会产生中断。

位/域	名称	类型	复位	描述
1	IE	R/W	0	CAN中断使能位 0: 禁止中断 1: 使能中断。
0	INIT	R/W	1	初始化 0: 正常工作 1: 初始化开始。

寄存器 **2: CAN** 状态 （**CANSTS**）寄存器，偏移量 **0x004**

状态寄存器包含了中断服务方面的信息，如脱离总线（Bus-off）、错误计数阈值和错误类型。

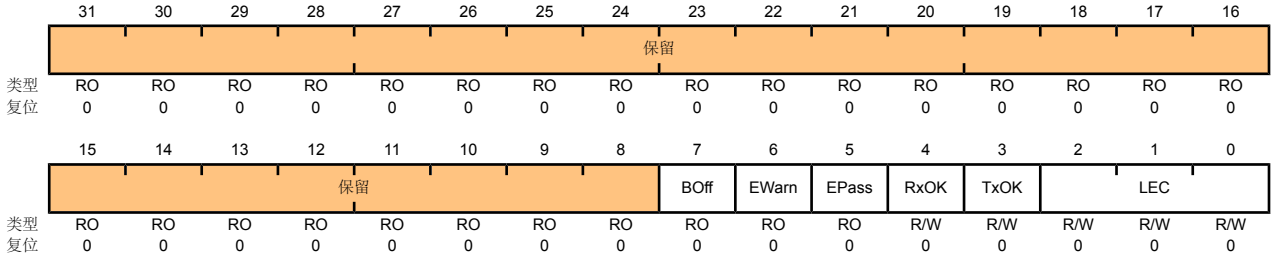
LEC 域指示了CAN总线上最后一次发生的错误的类型。这个域在无错地传输（接收或发送）完报文后会清零。CPU会写入未使用的错误代码7以检查更新情况。

假定**CAN**控制 **(CANCTL)** 寄存器中相应的使能位被置位，错误中断将由 BOff 位和 EWarn 位产生，状态改变中断将由 RxOk位、 TxOk位和 LEC 位产生。EPass 位的改变或者对 RxOk位、TxOk位或LEC位执行写操作都不会产生中断。

在CAN 中断(CANINT)挂起时，读取**CAN**状态 **(CANSTS)** 寄存器会将 **CAN** 中断**(CANINT)** 寄存器清零。

CAN 状态 (CANSTS)

偏移量 0x004
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	BOff	RO	0	脱离总线状态 0: 模块不是处于脱离总线状态。 1: 模块处于脱离总线状态。
6	EWarn	RO	0	警告状态 0: 两个错误计数器都低于错误警告极限值96。 1: 至少有一个错误计数器达到错误警告极限值96。
5	EPass	RO	0	错误认可 0:CAN模块处于错误激活状态，即接收或发送错误计数值小于等于127。 1: CAN模块处于错误认可状态，即接收或发送错误计数值大于127。
4	RxOK	R/W	0	成功接收报文 0: 由于这个位最后复位为 0，所以没有成功接收到报文。 1: 由于这个最后复位为0，所以成功接收到报文，与接收过滤的结果无关。 该位从未被CAN模块复位。

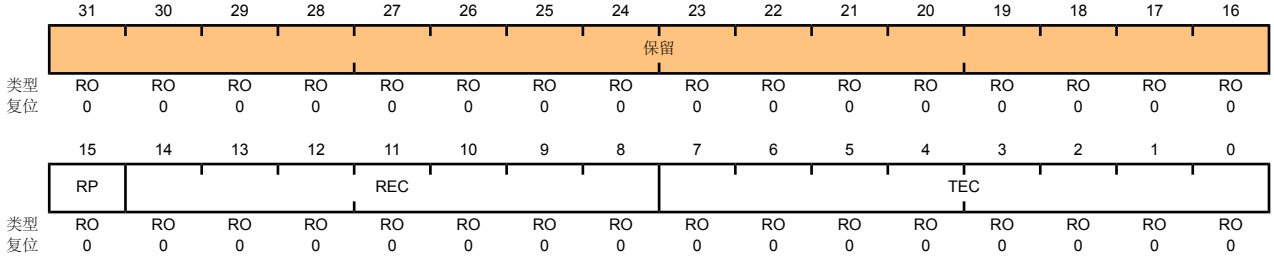
位/域	名称	类型	复位	描述
3	TxOK	R/W	0	成功发送报文 0: 由于该位最后复位为0，所以没有成功发送报文。 1: 由于该位最后复位为0，所以成功无错地发送了报文，并且被其它1个或1个以上节点应答。 该位从未被CAN模块复位。
2:0	LEC	R/W	0x0	最后一次错误代码 这表示的是CAN总线上最后一次发生的错误的类型。 值 定义 0x0 无错 0x1 填充错误 一个序列中有超过5个相同极性的位出现在接收报文中，这是接收报文所不能允许的。 0x2 格式错误 接收帧的固定格式部分包含错误格式。 0x3 应答错误 另一节点没有应答发送的报文。 0x4 位1错误 在发送报文时，CAN控制器对数据线进行监控以探测任一冲突。在发送仲裁场时，数据冲突是仲裁协议的一部分。在发送其它帧/场时，数据冲突被视作错误。 位1错误表示器件想发送高电平（逻辑1），但是监控到的总线值却是低电平（逻辑0）。 0x5 位0错误 位0错误表示器件想发送低电平（逻辑0），但是监控到的总线值却是高电平（逻辑1）。 在脱离总线恢复期间，每次监控到含11个高电平的序列后这个状态位就会置位。这使得CPU可以在不扰乱总线的情况下对脱离总线恢复序列的进程进行监控。 0x6 CRC错误 CRC 校验和在接收报文中是错误的，表示所计算得到的接收值与数据计算得到的CRC值不相同。 0x7 不使用 当 LEC 位为该值时, 自CPU将该值写入LEC后都没有探测到 CAN 总线事件。

寄存器 **3: CAN** 错误计数器（**CANERR**）寄存器，偏移量**0x008**

该寄存器包含了错误计数器值，可以用来分析错误原因。

CAN 错误计数器 (CANERR)

偏移量 0x008
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	RP	RO	0	接收错误认可 0: 接收错误计数器的值低于错误认可标准（小于等于127）。 1: 接收错误计数器的值达到错误认可标准（大于等于128）。
14:8	REC	RO	0x0	接收错误计数器 接收器错误计数器的状态 (0 到127)。
7:0	TEC	RO	0x0	发送错误计数器 发送错误计数器的状态 (0 到 255)。

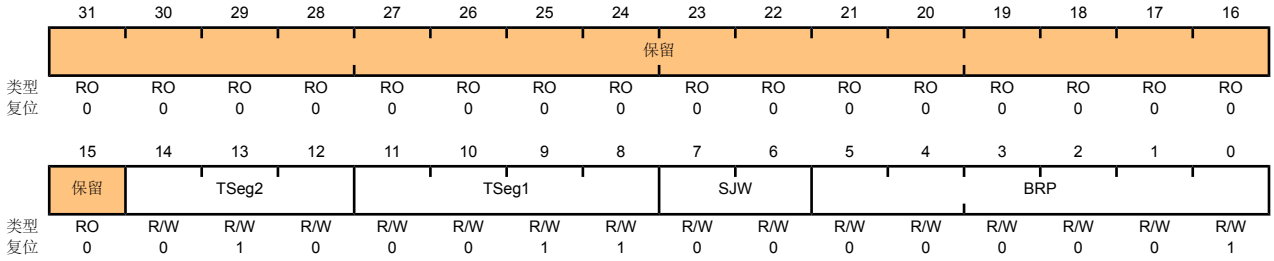
寄存器 4: CAN位定时（CANBIT）寄存器，偏移量 0x00C

该寄存器用来编程位宽和位份额。这些值被编程为系统时钟频率。该寄存器被**CANCTL**寄存器中的CCE和INIT位写使能。

通过一个8 MHz 的CAN模块时钟 (CAN_CLK)，该寄存器的复位值0x230将CAN配置成位速率等于500 Kbps。

CAN位定时 (CANBIT)

偏移量 0x00C
类型 R/W, 复位 0x0000.2301



位/域	名称	类型	复位	描述
31:15	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
14:12	TSeg2	R/W	0x2	采样点之后的时间段 0x00-0x07: 硬件在实际应用中对该值的解释要比此处编程的值大1。 因此，例如复位值0x2表示 Phase_Seg2 包含3（即2+1）个位时间份额 (见 图 15-2 在 360页)。位时间份额由BRP定义。
11:8	TSeg1	R/W	0x3	采样点之前的时间段 0x00-0x0F: 硬件在实际中对该值的解释要比此处编程的值大1。 因此，例如复位值0x3表示 Phase_Seg1 (see 图 15-2 在 360页)包含4（即3+1）个位时间份额。位时间份额由BRP定义。
7:6	SJW	R/W	0x0	(重)同步跳转宽度 0x00-0x03: 硬件在实际应用中对该值的解释要比此处编程的值大1。 在帧起始（SOF）过程中，如果CAN控制器检测到相位误差（偏差），它可以通过改变SJW的值来调节TSeg2或TSeg1 的长度。因此复位值0调节了1个位时间份额的长度。
5:0	BRP	R/W	0x1	波特率预分频器 0x00-0x03F: 产生位时间份额时，振荡器频率的分频因子。位时间由多个这种份额组成。硬件在实际应用中对该值的解释要比此处编程的值大1。 BRP 定义了组成1个位时间份额要多少CAN时钟周期, 因此其复位值是2（即1+1）个位时间份额。 **BRPRE**寄存器可以用来进一步分割位时间。

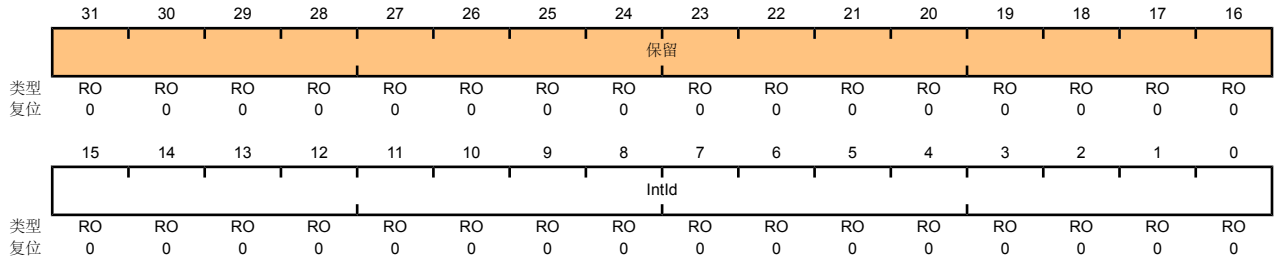
寄存器 **5: CAN** 中断（**CANINT**）寄存器，偏移量 **0x010**

这个寄存器用来指示中断源。

如果多个中断被挂起，**CAN** 中断 **(CANINT)** 寄存器将指向优先级最高的挂起中断，而不用考虑它们的时间顺序如何。中断会一直挂起，直至CPU将它清除。如果 `IntId` 位不是 0x0000 (缺省) 并且 **CANCTL**寄存器中的`IE`位没有置位，那么中断是激活的。在中断源复位时或直到`IE`复位，中断线会一直保持激活，直至 `IntId` 位变回0x0000 。

CAN 中断 (CANINT)

偏移量 0x010
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	IntId	RO	0x0000	中断标识符 该域的编号指明了中断源。

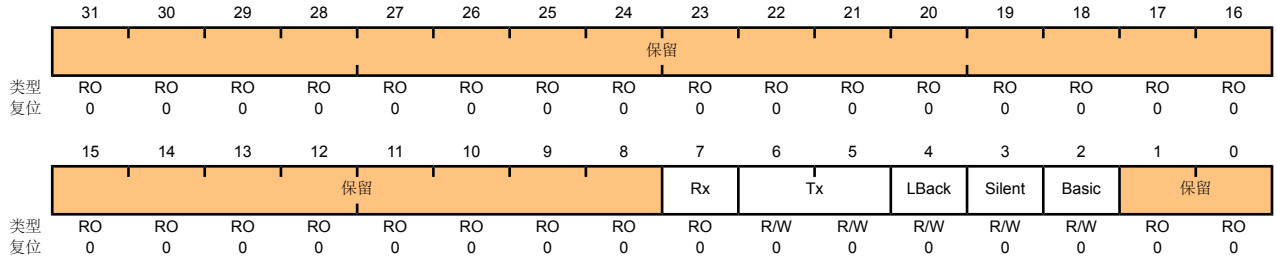
值	定义
0x0000	没有挂起中断
0x0001-0x0020	引起中断的报文对象的编号
0x0021-0x7FFF	不使用
0x8000	状态中断
0x8001-0xFFFF	不使用

寄存器 6: CAN 测试（CANTST）寄存器，偏移量 0x014

这是用于自测试和外部管脚访问的测试模式寄存器。它被 **CANCTL**寄存器中的Test位写使能。 不同的测试功能可以组合在一起，但是当Tx位不等于 0x0时，它会扰乱报文发送。

CAN 测试 (CANTST)

偏移量 0x014
类型 R/W, 复位 0x0000.0000



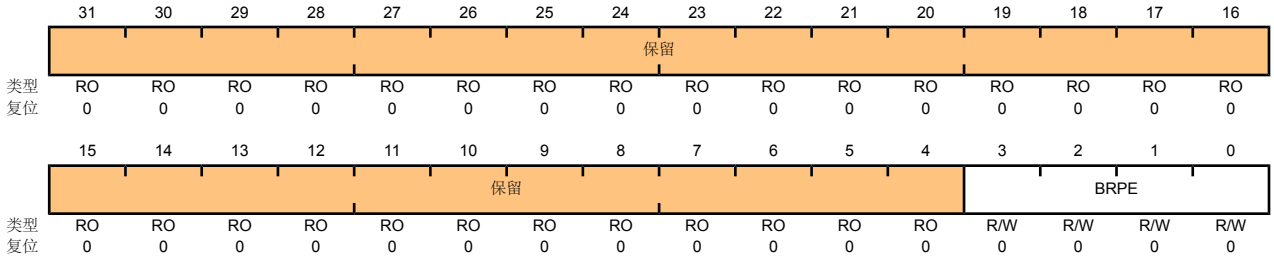
位/域	名称	类型	复位	描述
31:8	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	Rx	RO	0	接收观测 显示CANnRx 管脚上的值。
6:5	Tx	R/W	0x0	发送控制 CANnTx管脚的覆盖控制。 值 描述 00 CAN_TX 受 CAN模块 (缺省)控制 01 CAN_TX管脚上驱动的采样点信号 10 CAN_TX 驱动一个低电平值 11 CAN_TX 驱动一个高电平值
4	LBack	R/W	0	回送模式 0: 禁止 1: 使能
3	Silent	R/W	0	静止模式 不发送数据；监控总线。也称为总线监控模式。 0: 禁止 1: 使能
2	Basic	R/W	0	基本模式 0: 禁止 1: 将 **CANIF1**寄存器用作发送缓冲器，将 **CANIF2** 寄存器用作接收缓冲器。
1:0	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 **7: CAN**波特率预分频扩展（**CANBRPE**）寄存器，偏移量 **0x018**

该寄存器使用**CANBIT** 寄存器中的BRP 位对位时间进行近一步分频。 它被**CANCTL** 寄存器中的 CCE 位写使能。

CAN波特率预分频扩展 (CANBRPE)

偏移量 0x018
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:4	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3:0	BRPE	R/W	0x0	波特率预分频扩展。 0x00-0x0F: 最高可将 BRP 位扩展到1023。硬件在实际应用中对该位的解释要比 BRPE (MSBs) 和 BRP (LSBs) 编程的值大1。

寄存器 8: CAN IF1命令请求（CANIF1CRQ），偏移量 0x020

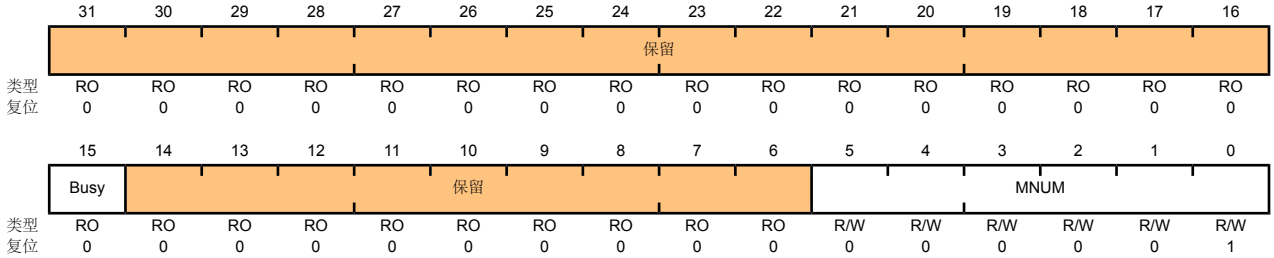
寄存器 9: CAN IF2 命令请求（CANIF2CRQ），偏移量 0x080

该寄存器在MNUM位被更新时用来启动传输。 其 Busy 位表示信息正从CAN接口寄存器传输到内部报文RAM。

MNUM位一写入报文对象编号，报文传输就开始。执行这个写操作后， Busy 位被自动设为1，指示传输正在进行。在等待了3到6个CAN_CLK周期后，接口寄存器和报文RAM之间的传输结束，Busy 位变回0。

CAN IF1命令请求 (CANIF1CRQ)

偏移量 0x020
类型 RO, 复位 0x0000.0001



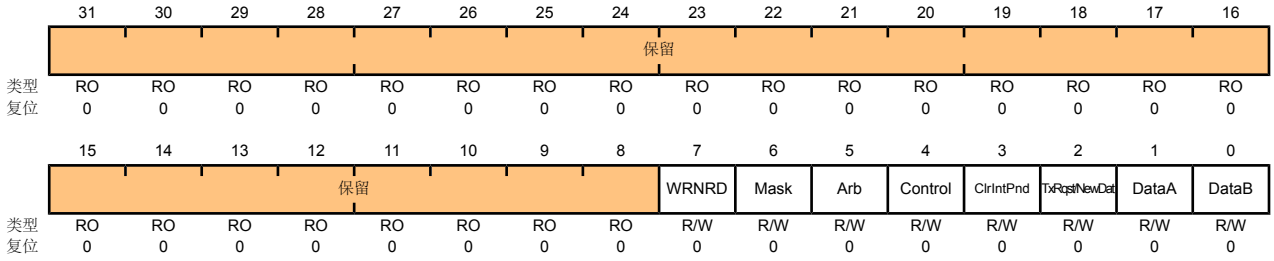
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	Busy	RO	0x0	忙标志 0: 当读/写动作结束时复位。 1: 在对该寄存器中的报文编号进行写操作时置位。
14:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5:0	MNUM	R/W	0x01	报文编号 在报文RAM中的32个报文对象中挑出一个进行数据传输。报文对象的编号是：1到32。 值 描述 0x00 0 不是有效的报文编号；它被解释为0x20或对象32。 0x01-0x20 表示指定的报文对象1到报文对象32。 0x21-0x3F 不是有效的报文编号；这些值被替换，并且被解释为0x01-0x1F。

寄存器 **10: CAN IF1** 命令屏蔽（**CANIF1CMSK**），偏移量 **0x024**
寄存器 **11: CAN IF2** 命令屏蔽（**CANIF2CMSK**），偏移量 **0x084**

命令屏蔽寄存器规定传输方向并决定哪个缓冲寄存器是数据传输的源与目标。

CAN IF1 命令屏蔽 (CANIF1CMSK)

偏移量 0x024
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:8	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	WRNRD	R/W	0	写，不读 0: 读。将 **CAN** 命令请求 （**CANIFnCRQ**） 寄存器指定的报文对象地址传输到CAN报文缓冲寄存器 （**CANIFnMSK1**、 **CANIFnMSK2**、 **CANIFnARB1**、 **CANIFnARB2**、 **CANIFnCTL**、**CANIFnDA1**、 **CANIFnDA2**、 **CANIFnDB1**和 **CANIFnDB2**）。 1: 写。将数据从报文缓冲寄存器传输到由 **CANIFnCRQ** 寄存器指定的报文对象的地址中。
6	Mask	R/W	0x0	访问屏蔽位 当 `WRNRD=1` (写入)时: 0: 屏蔽位不变。 1: 传输 `IDMask` + `Dir` + `MXtd` 到报文对象。 当 `WRNRD=0` (读取)时: 0: 屏蔽位不变。 1: 传输报文对象的 `IDMask` + `Dir` + `MXtd` 到接口寄存器。
5	Arb	R/W	0x0	访问仲裁位 当 `WRNRD=1` (写入)时: 0: 仲裁位不变。 1: 传输 `ID` + `Dir` + `Xtd` + `MsgVal` 到报文对象。 当 `WRNRD=0` (读取)时: 0: 仲裁位不变。 1: 传输 `ID` + `Dir` + `Xtd` + `MsgVal` 到报文缓冲寄存器。

位/域	名称	类型	复位	描述
4	Control	R/W	0x0	访问控制位 当 WRNRD=1 (写入)时: 0: 控制位不变。 1:传输控制位到报文对象。 当 WRNRD=0 (读取)时: 0: 控制位不变。 1: 传输控制位到报文缓冲寄存器。
3	ClrIntPnd	R/W	0x0	清零中断挂起位 注意: 在写入 (WRNRD=1)时不使用这个位。 0: **CANIFnMCTL** 寄存器中的IntPnd 位保持不变。 1: 将报文对象中**CANIFnMCTL** 寄存器中的 IntPnd 位清零。
2	TxRqst/NewDat	R/W	0x0	访问发送请求或新数据 当 WRNRD=1 (写入)时: 访问发送请求位 0: TxRqst 位不变。 1: TxRqst位置位 注意: 如果通过编程这个TxRqst位来请求发送，那么**CANIFnMCTL** 寄存器中相同的 TxRqst位被忽略。 当 WRNRD=0 (读取)时: 访问新数据位 0: NewDat 位不变。 1: 报文对象中的NewDat 位被清零。 注意: 读取报文对象这个动作可以与复位IntPnd和 NewDat动作同时进行。那些被传输到**CANIFnMCTL** 寄存器的位总是反映了它们被复位之前的状态。
1	DataA	R/W	0x0	访问数据字节 0 到3 当 WRNRD=1 (写入)时: 0: 数据字节 0-3 不变。 1: 传输数据字节0-3 (**CANIFnDA1** 和 **CANIFnDA2**) 到报文对象。 当 WRNRD=0 (读取)时: 0: 数据字节 0-3 不变。 1: 传输报文对象中的数据字节 0-3 到**CANIFnDA1**和 **CANIFnDA2**。

位/域	名称	类型	复位	描述
0	DataB	R/W	0x0	访问数据字节 4 到 7 当 WRNRD=1 (写入)时: 0: 数据字节 4-7 不变。 1: 传输数据字节 4-7 (**CANIFnDB1** 和 **CANIFnDB2**) 到报文对象。 当 WRNRD=0 (读取)时: 0: 数据字节 4-7 不变。 1: 传输报文对象中的数据字节 4-7 到 **CANIFnDB1** 和 **CANIFnDB2**。

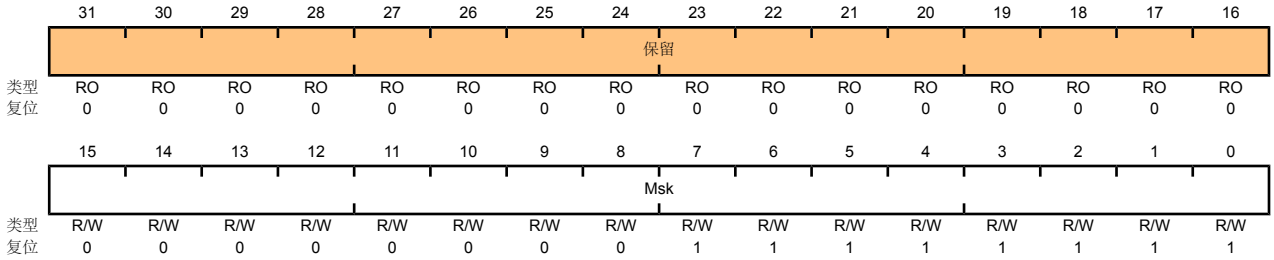
寄存器 12: CAN IF1 屏蔽 1（CANIF1MSK1），偏移量 0x028

寄存器 13: CAN IF2 屏蔽1（CANIF2MSK1），偏移量 0x088

该寄存器提供的屏蔽信息连同数据 (**CANIFnDAn**)、仲裁信息 (**CANIFnARBn**)和控制信息 (**CANIFnMCTL**)一起保存到报文RAM中的报文对象。屏蔽位连同**CANIFnARBn**寄存器中的ID 位用来接收滤波。额外的屏蔽信息包含在 **CANIFnMSK2** 寄存器中。

CAN IF1 屏蔽 1 (CANIF1MSK1)

偏移量 0x028
类型 RO, 复位 0x0000.FFFF



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	Msk	R/W	0xFF	标志符屏蔽码 0: 报文对象中对应的标识符位（ID）不能禁止接收滤波中的匹配。 1: 对应的标识符位（ID）用于接收滤波。

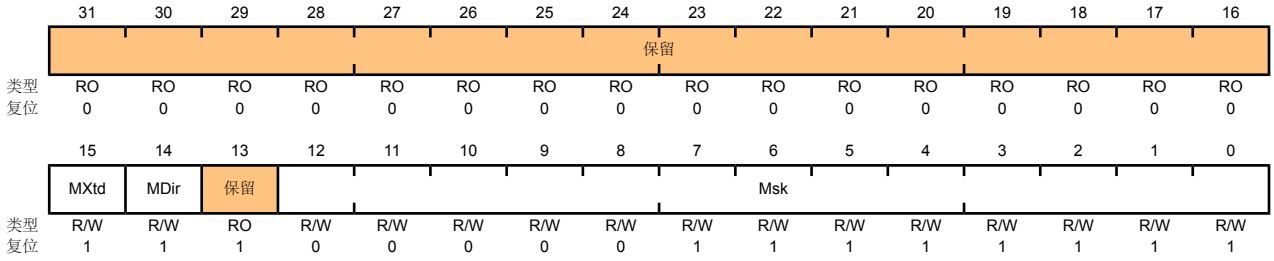
寄存器 **14: CAN IF1**屏蔽**2**（**CANIF1MSK2**），偏移量 **0x02C**

寄存器 **15: CAN IF2**屏蔽**2**（**CANIF2MSK2**），偏移量 **0x08C**

该寄存器保存了**CANIFnMSK1** 寄存器相关的屏蔽信息。

CAN IF1屏蔽2 (CANIF1MSK2)

偏移量 0x02C
类型 RO, 复位 0x0000.FFFF



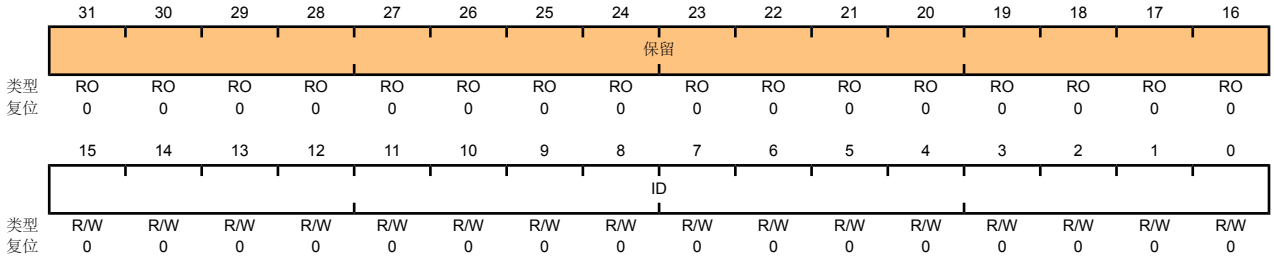
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	MXtd	R/W	0x1	屏蔽码扩展标识符 0: 扩展标识符位 (**CANIFnARB2**寄存器中的`Xtd`) 不会影响接收滤波。 1: 扩展标识符位 `Xtd` 用于接收滤波。
14	MDir	R/W	0x1	屏蔽报文方向 0: 报文方向位 (**CANIFnARB2** 寄存器中的`Dir`) 不会影响接收滤波。 1: 报文方向位 `Dir` 用于接收滤波。
13	保留	RO	0x1	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
12:0	Msk	R/W	0xFF	标志符屏蔽码 0: 报文对象中对应的标识符位（ID）不能禁止接收滤波中的匹配。 1: 对应的标识符位（ID）用于接收滤波。

寄存器 16: CAN IF1 仲裁 1（CANIF1ARB1），偏移量 0x030
寄存器 17: CAN IF2 仲裁 1（CANIF2ARB1），偏移量 0x090

这些寄存器保存接收滤波的标识符。

CAN IF1 仲裁 1 (CANIF1ARB1)

偏移量 0x030
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	ID	R/W	0x00	报文标识符 这个位域和**CANIFnARB2** 寄存器中的 `ID` 位域一起用于建立报文标识符。ID[28:0] 是扩展帧，ID[28:18] 是标准帧。

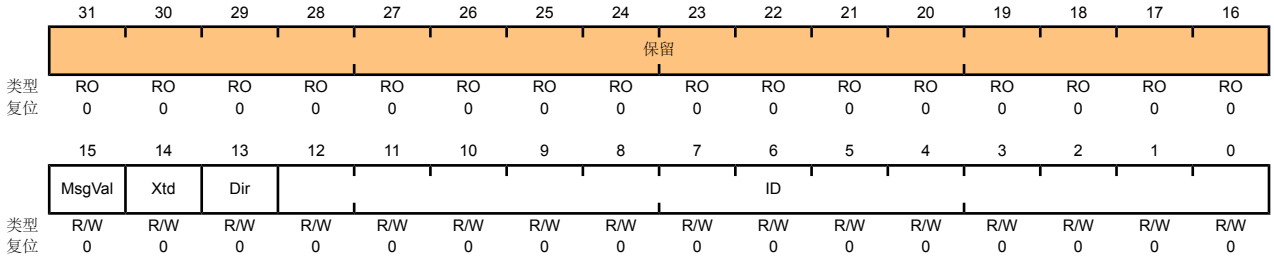
寄存器 **18: CAN IF1** 仲裁 **2**（**CANIF1ARB2**），偏移量 **0x034**

寄存器 **19: CAN IF2** 仲裁 **2**（**CANIF2ARB2**），偏移量 **0x094**

这些寄存器保存接收滤波的信息。

CAN IF1 仲裁 2 (CANIF1ARB2)

偏移量 0x034
类型 RO, 复位 0x0000.0000



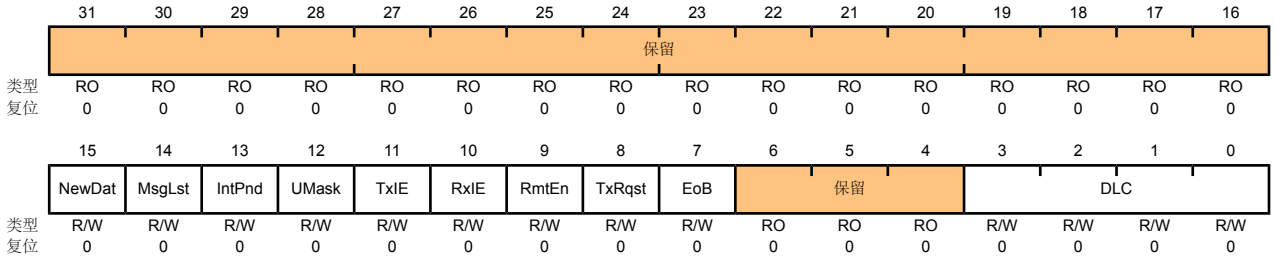
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	MsgVal	R/W	0x0	报文有效 0: 该报文对象被报文处理器忽略。 1: CAN控制器内的报文处理器将对该报文对象进行配置，并且将其考虑在内。 在初始化过程中以及**CANCTL** 寄存器中的 `Init`位清零前，所有未用的报文对象应该将该位清零。 在**CANIFnARBn** 寄存器中的`ID` 位、**CANIFnARB2** 寄存器中的 `Xtd` 和 `Dir` 位、或者**CANIFnMCTL** 寄存器中的`DLC` 位中的任一位被修改之前，或者不再需要报文对象时，也必须将`MsgVal` 位清零。
14	Xtd	R/W	0x0	扩展标识符 0: 11-位标准标识符将用于该报文对象。 1: 29-位扩展标识符将用于该报文对象。
13	Dir	R/W	0x0	报文方向 0: 接收`TxRqst`发出后，带有这个报文对象的标识符的远程帧将被发送。在接收到带有匹配标识符的数据帧后，该报文将被存储到该报文对象。 1: 发送`TxRqst`发出后，各自的报文对象将被当作数据帧发送。在接收到含匹配标识符的远程帧后，该报文对象的`TxRqst` 位会置位(若`RmtEn=1`)。
12:0	ID	R/W	0x0	报文标识符 和**CANIFnARB1** 寄存器中的 `ID` 位一起被用来建立报文标识符。ID[28:0] 是扩展帧，ID[28:18] 是标准帧。

寄存器 20: CAN IF1 报文控制（CANIF1MCTL），偏移量 0x038
寄存器 21: CAN IF2 报文控制 （CANIF2MCTL），偏移量 0x098

该寄存器保存了将要发送给报文RAM的报文对象相关的控制信息。

CAN IF1 报文控制 (CANIF1MCTL)

偏移量 0x038
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15	NewDat	R/W	0x0	新数据 0: 自上次该标志被CPU清零以来，报文处理器就没有写新数据到报文对象的数据部分。 1: 报文处理器或CPU已经写新数据到报文对象的数据部分。
14	MsgLst	R/W	0x0	报文丢失 0 : 自上次该位被CPU清零以来就没有丢失报文。 1: 在 NewDat 置位时，报文处理器存储新报文到这个对象内；CPU 丢失了报文。 这个位仅在**CANIFnARB2** 寄存器中的 Dir 位被置为 0 (接收)时才有效。
13	IntPnd	R/W	0x0	中断挂起 0: 该报文对象不是中断源。 1: 该报文对象是中断源。如果此时包含一个优先级更高的中断，那么 **CAN** 中断 **(CANINT)** 寄存器中的中断标识符将指向该报文对象。
12	UMask	R/W	0x0	使用接收屏蔽码 0: 屏蔽被忽略。 1: 将屏蔽码 (Msk、 MXtd和 MDir) 用于接收滤波。
11	TxIE	R/W	0x0	发送中断使能 0: 在成功发送帧后，**CANIFnMCTL** 寄存器中的 IntPnd 位不发生改变。 1: 在成功发送帧后，**CANIFnMCTL** 寄存器中的IntPnd 位置位。
10	RxIE	R/W	0x0	接收中断使能 0: 在成功接收帧后，**CANIFnMCTL** 寄存器中的IntPnd 位不发生改变。 1: 在成功接收帧后，**CANIFnMCTL** 寄存器中的IntPnd 位置位。

位/域	名称	类型	复位	描述
9	RmtEn	R/W	0x0	远程使能 0: 在接收到远程帧后，**CANIFnMCTL** 寄存器中的 TxRqst 位不发生改变。 1: 在接收到远程帧后， **CANIFnMCTL** 寄存器中的TxRqst 位置位。
8	TxRqst	R/W	0x0	发送请求 0: 该报文对象不等待发送。 1: 已请求发送该报文对象，但还没完成。
7	EoB	R/W	0x0	缓冲结束 0: 报文对象属于FIFO缓冲器，并且不是FIFO缓冲器最后一个报文对象。 1: 单个报文对象或FIFO缓冲器最后一个报文对象。 该位用来串连两个或两个以上报文对象（多达32个）以创建FIFO缓冲器。当为单个报文对象(因而不属于FIFO缓冲器)时，这个位必须置为1。
6:4	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3:0	DLC	R/W	0x0	数据长度码 值 描述 0x0-0x8 指定数据帧中的字节数。 0x9-0xF 缺省情况下是含8字节的数据帧。 报文对象的**CANIFnMCTL** 寄存器中的 DLC 位必须和其它节点上含相同标识符的对应对象一样被定义。 在信息处理器存储数据帧时，它将DLC 写入由接收报文给定的值。

寄存器 **22: CAN IF1** 数据 **A1**（**CANIF1DA1**），偏移量 **0x03C**

寄存器 **23: CAN IF1** 数据 **A2**（**CANIF1DA2**），偏移量 **0x040**

寄存器 **24: CAN IF1** 数据 **B1**（**CANIF1DB1**），偏移量 **0x044**

寄存器 **25: CAN IF1** 数据 **B2**（**CANIF1DB2**），偏移量 **0x048**

寄存器 **26: CAN IF2** 数据 **A1**（**CANIF2DA1**），偏移量 **0x09C**

寄存器 **27: CAN IF2** 数据 **A2**（**CANIF2DA2**），偏移量 **0x0A0**

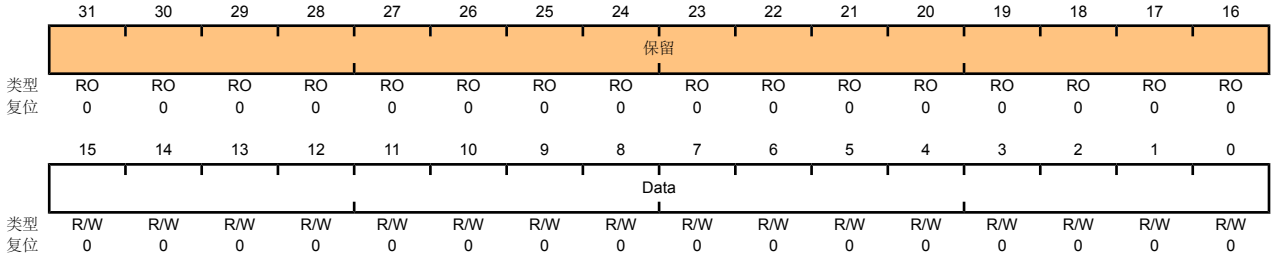
寄存器 **28: CAN IF2** 数据 **B1**（**CANIF2DB1**），偏移量 **0x0A4**

寄存器 **29: CAN IF2** 数据 **B2**（**CANIF2DB2**），偏移量 **0x0A8**

这些寄存器包含要发送的数据或已接收的数据。在CAN 数据帧中， 数据字节0是要发送或接收的第一个字节，而数据字节7是要发送或接收的最后一个字节。在CAN的串行位流中，各字节的MSB首先被发送。

CAN IF1 数据 A1 (CANIF1DA1)

偏移量 0x03C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	Data	R/W	0x00	Data **CANIFnDA1**寄存器包含数据字节1和0; **CANIFnDA2**包含数据字节3和2; **CANIFnDB1**包含数据字节5和4; 以及**CANIFnDB2**包含数据字节7和6。

寄存器 **30: CAN**发送请求**1**（**CANTXRQ1**），偏移量 **0x100**

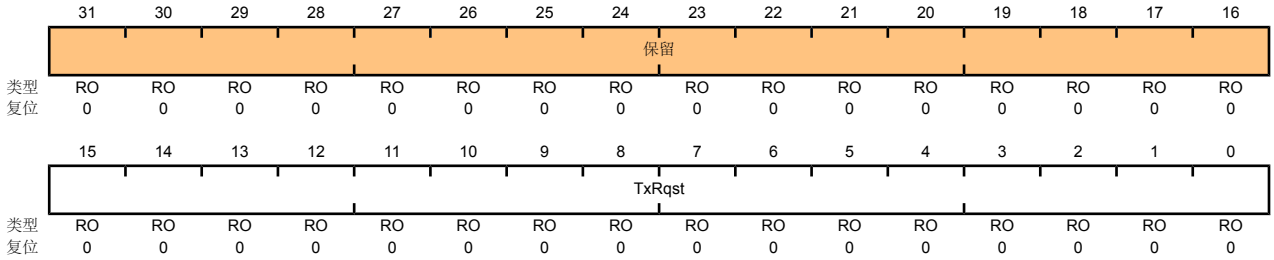
寄存器 **31: CAN** 发送请求**2**（**CANTXRQ2**），偏移量 **0x104**

CANTXRQ1 和 **CANTXRQ2** 寄存器保存了32个报文对象的 `TxRqst` 位。通过读取这些位，CPU 可以找出将发送请求挂起的报文对象。某一特定报文对象的`TxRqst` 位可以通过以下3种方式改变：(1) CPU 通过 **CAN IFn** 报文控制 **(CANIFnMCTL)** 寄存器 (2) 在接收到远程帧后，通过报文处理器状态机 (3) 在成功发送后，通过报文处理器状态机。

CANTXRQ1 寄存器包含报文RAM中第一个16报文对象的 `TxRqst` 位； **CANTXRQ2** 寄存器包含第二个16报文对象的 `TxRqst` 位。

CAN发送请求1 (CANTXRQ1)

偏移量 0x100
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TxRqst	RO	0x00	发送请求位 (所有报文对象的) 0: 报文对象不等待发送。 1: 已请求发送报文对象，但还没完成。

寄存器 32: CAN新数据1 （CANNWDA1），偏移量 0x120

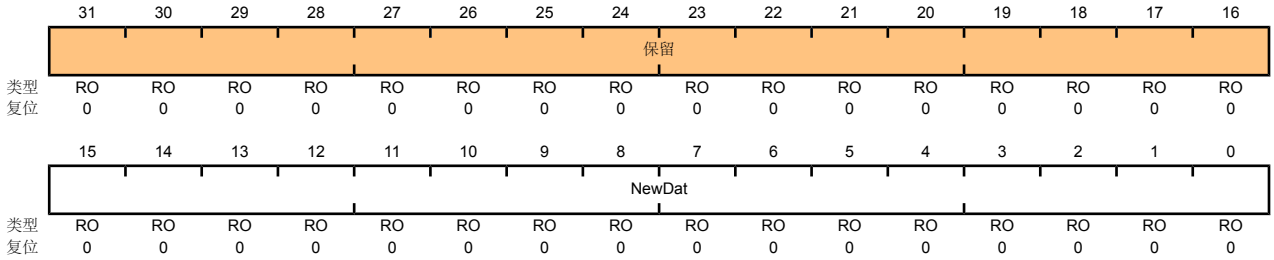
寄存器 33: CAN新数据2 （CANNWDA2），偏移量 0x124

CANNWDA1 和 **CANNWDA2** 寄存器包含32个报文对象的 NewDat位。通过读取这些位，CPU可以找出将数据部分更新的报文对象。某一特定报文对象的NewDat 位可以通过以下三种方式改变：(1) CPU 通过 **CAN IFn** 报文控制 **(CANIFnMCTL)** 寄存器 (2) 在接收数据帧后，通过报文处理器状态机 (3) 在成功发送后，通过报文处理器状态机。

CANNWDA1 寄存器包含报文RAM中第一个16报文对象的 NewDat 位； **CANNWDA2**寄存器包含第二个16报文对象的 NewDat 位。

CAN新数据1 (CANNWDA1)

偏移量 0x120
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	NewDat	RO	0x00	新数据位 (所有报文对象的) 0: 自上次该标志被CPU清零以来，报文处理器就没有写新数据到报文对象的数据部分。 1: 报文处理器或CPU已经写新数据到报文对象的数据部分。

寄存器 **34: CAN**报文**1**中断挂起（**CANMSG1INT**），偏移量 **0x140**

寄存器 **35: CAN**报文**2**中断挂起（**CANMSG2INT**），偏移量 **0x144**

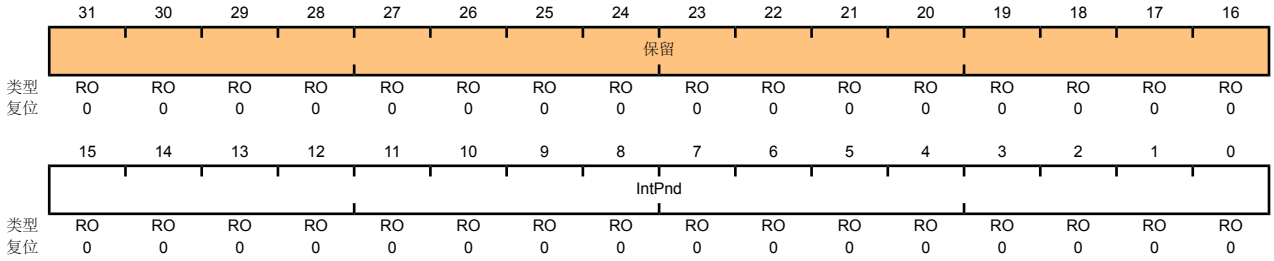
CANMSG1INT 和 **CANMSG2INT** 寄存器保存了32个报文对象的 `IntPnd` 位。通过读取这些位，CPU可以找出挂起中断的报文对象。某一特定报文对象的`IntPnd` 位可以通过以下两种方式改变：(1) CPU 通过 **CAN IFn** 报文控制 **(CANIFnMCTL)** 寄存器 (2) 在接收或发送完帧之后，通过报文处理器状态机。

这个域也可以在 **CAN** 中断 **(CANINT)** 寄存器中编码。

CANMSG1INT 寄存器包含报文RAM中第一个16报文对象的 `IntPnd` 位； **CANMSG2INT** 寄存器包含第二个16报文对象的 `IntPnd` 位。

CAN报文1中断挂起 (CANMSG1INT)

偏移量 0x140
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	IntPnd	RO	0x00	中断挂起位 (所有报文对象的) 0: 该报文对象不是中断源。 1: 该报文对象是中断源。

寄存器 36: CAN报文1有效 （CANMSG1VAL），偏移量 0x160

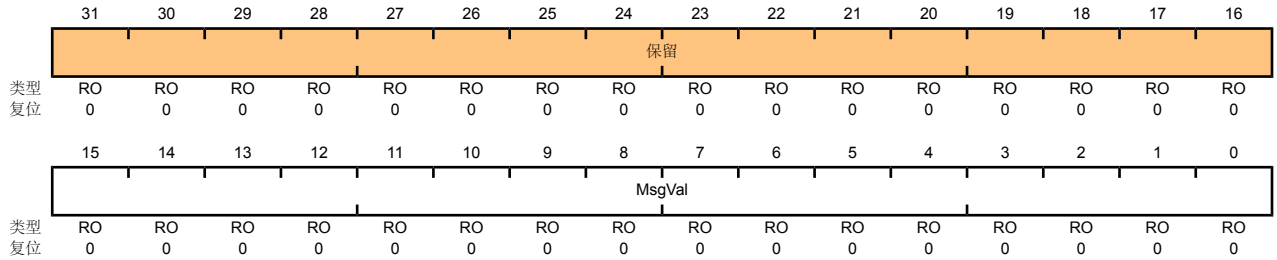
寄存器 37: CAN 报文2有效（CANMSG2VAL），偏移量 0x164

CANMSG1VAL 和 **CANMSG2VAL** 寄存器保存了32个报文对象的 `MsgVal` 位。通过读取这些位，CPU可以找出哪个报文对象有效。某一特定报文对象的报文值可以通过 **CAN IFn** 报文控制 **(CANIFnMCTL)**寄存器改变。

CANMSG1VAL寄存器包含报文RAM中第一个16报文对象的`MsgVal`位；**CANMSG2VAL**寄存器包含报文RAM中第二个16报文对象的`MsgVal`位。

CAN报文1有效 (CANMSG1VAL)

偏移量 0x160
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x0000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	MsgVal	RO	0x00	报文有效位 (所有报文对象的) 0: 该报文对象没有被配置，并且被报文处理器忽略。 1: 报文处理器对该报文对象进行配置并应将其考虑在内。

16 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。

LM3S2950 控制器提供3个独立的集成模拟比较器，可配置模拟比较器来驱动输出或产生中断。

注意: 不是所有比较器都可以选择驱动输出管脚。详见“比较器工作模式”表。

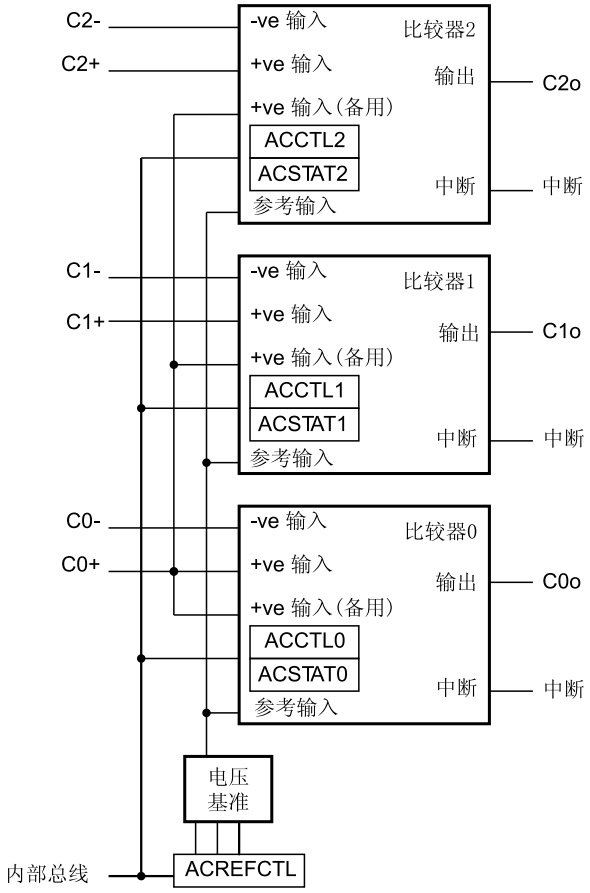
比较器可将测试电压与下面的其中一种电压相比较：

- 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，或可以使用比较器通过中断 通知应用让它开始捕获采样序列。

16.1 结构图

图 **16-1.** 模拟比较器模块的结构图



16.2 功能描述

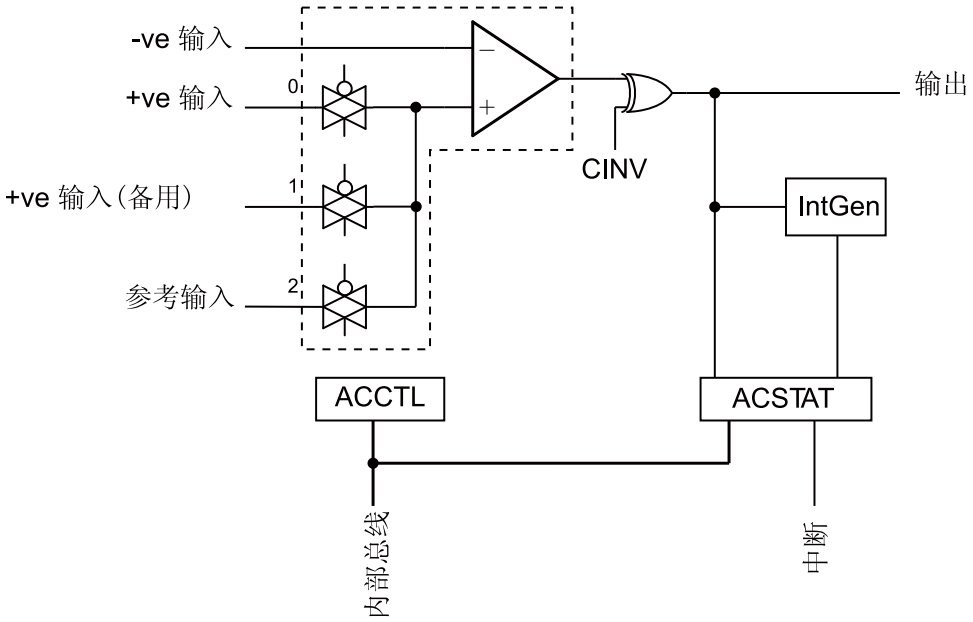
重要: 建议将模拟输入管脚的数字-输入使能 (GPIO模块的GPIODEN 位) ，以防止从I/O端口中吸收过量的电流。

比较器通过比较VIN-和VIN+输入来产生输出VOUT。

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

如 图 16-2 在 389页所示，VIN- 的输入源是一个外部输入电压。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 **16-2.** 比较单元的结构



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。 比较器的工作模式请参考“比较器工作模式”表。

通常，会在内部使用比较器输出来产生控制器中断。但比较器也可以用来驱动外部管脚。

重要: 在使用模拟比较器之前必须置位某些寄存器位值。比较器输入和输出管脚的正确端口配置在“比较器工作模式”表中有描述。

表 **16-1.** 比较器**0**的工作模式

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

表 **16-2.** 比较器**1**的工作模式

ACCNTL1	比较器 **1**			
ASRCP	**VIN-**	**VIN+**	输出	中断
00	C1-	C1o/C1+[a]	C1o/C1+	是
01	C1-	C0+	C1o/C1+	是
10	C1-	Vref	C1o/C1+	是
11	C1-	保留	C1o/C1+	是

a. C1o 和 C1+ 信号共用一个管脚，且该管脚只可以使用其中一个信号。

表 **16-3.** 比较器**2**的工作模式

ACCNTL2	比较器 **2**			
ASRCP	**VIN-**	**VIN+**	输出	中断
00	C2-	C2o/C2+[a]	C2o/C2+	是
01	C2-	C0+	C2o/C2+	是
10	C2-	Vref	C2o/C2+	是
11	C2-	保留	C2o/C2+	是

a. C2o 和 C2+ 信号共用一个管脚，且该管脚只可以使用其中一个信号。

16.2.1 内部参考编程

内部的参考结构如图 16-3 在 390页所示。该结构通过一个配置寄存器(**ACREFCTL**)来控制。表 16-4 在 390页 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 **16-3.** 比较器内部参考结构

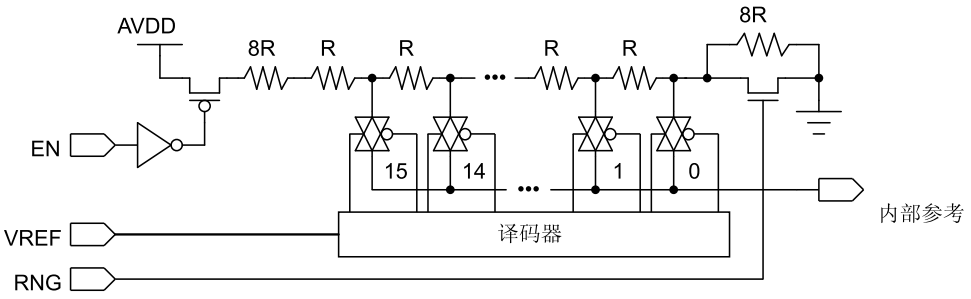


表 **16-4.** 内部参考电压和**ACREFCTL**位域的值

ACREFCTL 寄存器		基于**VREF**位域的输出参考电压
EN 位的值	**RNG** 位的值	
EN=0	RNG=X	无论VREF为任何值,地面(GND)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。

ACREFCTL 寄存器		基于VREF位域的输出参考电压
EN 位的值	RNG 位的值	
EN=1	RNG=0	阶梯电阻的总阻值为32R。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 \quad VREF$ 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。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \times V_{REF}$ 在该模式中内部参考电压的范围是0.0－2.0625V。

16.3 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0x0010.0000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0x0000.030C,从而将内部电压参考配置为1.65V。
4. 向**ACCTL0**寄存器写入0x0000.040C，从而将比较器0配置为使用内部电压参考，且不将C0o管脚上的输出反相。
5. 延迟一段时间。
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。

16.4 寄存器映射

表 16-5 在 392页列出了比较器寄存器。表中所列偏移量是寄存器地址相对于模拟比较器基址0x4003.C000的十六进制地址增量。

表 **16-5.** 模拟比较器 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x00	ACMIS	R/W1C	0x0000.0000	模拟比较器屏蔽后的中断状态	393
0x04	ACRIS	RO	0x0000.0000	模拟比较器原始中断状态	394
0x08	ACINTEN	R/W	0x0000.0000	模拟比较器中断使能	395
0x10	ACREFCTL	R/W	0x0000.0000	模拟比较器参考电压控制	396
0x20	ACSTAT0	RO	0x0000.0000	模拟比较器状态0	397
0x24	ACCTL0	R/W	0x0000.0000	模拟比较器控制0	398
0x40	ACSTAT1	RO	0x0000.0000	模拟比较器状态1	397
0x44	ACCTL1	R/W	0x0000.0000	模拟比较器控制1	398
0x60	ACSTAT2	RO	0x0000.0000	模拟比较器状态2	397
0x64	ACCTL2	R/W	0x0000.0000	模拟比较器控制2	398

16.5 寄存器描述

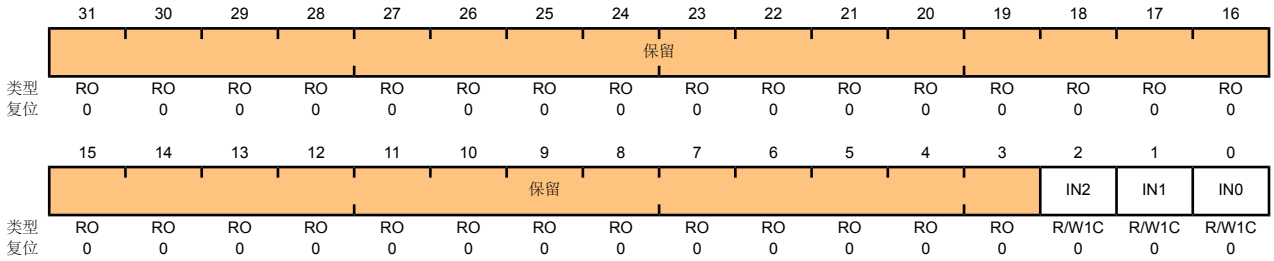
本节内容将按地址偏移量的数字顺序来列举并且描述模拟比较器寄存器。

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的（经过屏蔽后的）中断状态。

模拟比较器屏蔽后的中断状态 (ACMIS)

偏移量 0x00
类型 R/W1C, 复位 0x0000.0000

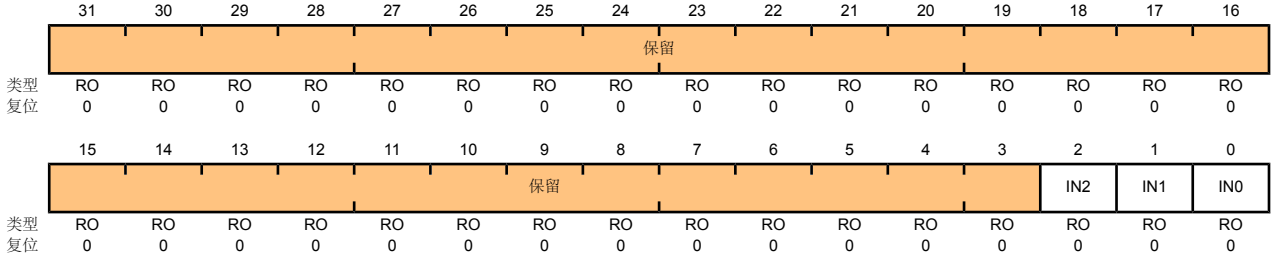


位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IN2	R/W1C	0	比较器2屏蔽后的中断状态 给出该中断屏蔽后的中断状态。向该位写1可以清零挂起中断。
1	IN1	R/W1C	0	比较器1屏蔽后的中断状态 给出该中断屏蔽后的中断状态。向该位写1可以清零挂起中断。
0	IN0	R/W1C	0	比较器0屏蔽后的中断状态 给出该中断屏蔽后的中断状态。向该位写1可以清零挂起中断。

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)
偏移量 0x04
类型 RO, 复位 0x0000.0000



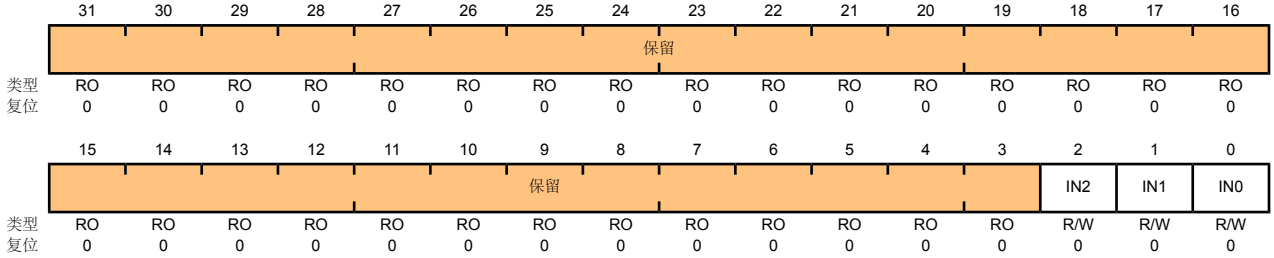
位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IN2	RO	0	比较器2中断状态 当该位置位时，表示中断已通过比较器2产生。
1	IN1	RO	0	比较器1中断状态 当该位置位时，表示中断已通过比较器1产生。
0	IN0	RO	0	比较器0中断状态 当该位置位时，表示中断已通过比较器0产生。

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)

偏移量 0x08
类型 R/W, 复位 0x0000.0000

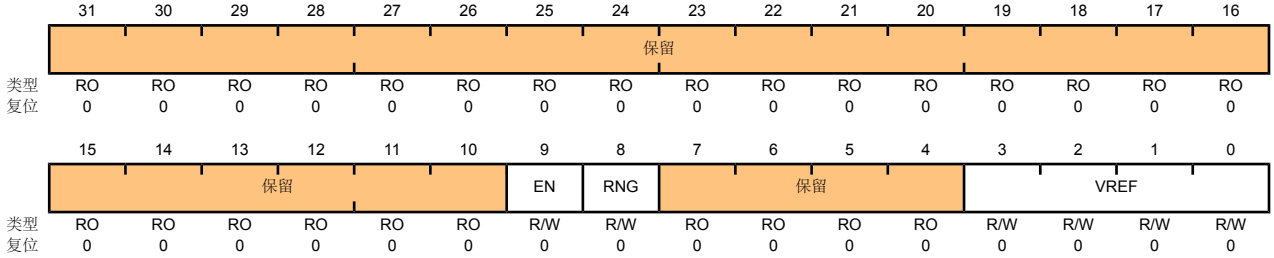


位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IN2	R/W	0	比较器2中断使能位 当该位置位时，使能比较器2输出的控制器中断。
1	IN1	R/W	0	比较器1中断使能位 当该位置位时，使能比较器1输出的控制器中断。
0	IN0	R/W	0	比较器0中断使能位 当该位置位时，使能比较器0输出的控制器中断。

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，以及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 0x10
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:10	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
9	EN	R/W	0	阶梯电阻使能位 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	阶梯电阻范围 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	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3:0	VREF	R/W	0x00	阶梯电阻的电压参考 VREF 位域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 16-4 在 390页。

寄存器 **5:** 模拟比较器状态**0**（**ACSTAT0**），偏移量 **0x20**

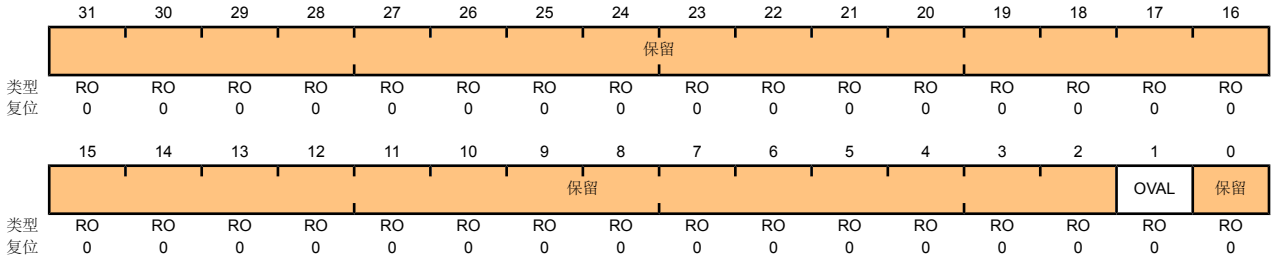
寄存器 **6:** 模拟比较器状态**1**（**ACSTAT1**），偏移量 **0x40**

寄存器 **7:** 模拟比较器状态**2**（**ACSTAT2**），偏移量 **0x60**

这些寄存器指示比较器的当前输出值。

模拟比较器状态0 (ACSTAT0)

偏移量 0x20
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	OVAL	RO	0	比较器输出值 OVAL 位指示比较器的当前输出值。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器 **8:** 模拟比较器控制**0**（**ACCTL0**），偏移量 **0x24**

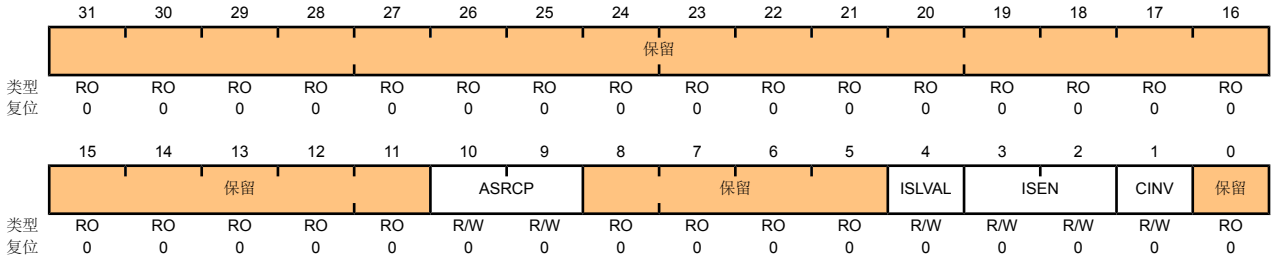
寄存器 **9:** 模拟比较器控制**1**（**ACCTL1**），偏移量 **0x44**

寄存器 **10:** 模拟比较器控制**2**（**ACCTL2**），偏移量 **0x64**

这些寄存器用来配置 比较器的输入和输出。

模拟比较器控制0 (ACCTL0)

偏移量 0x24
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:11	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10:9	ASRCP	R/W	0x00	模拟正电压源 ASRCP 位域指定了到比较器VIN+端口的输入电压源。该位域的编码如下： 值 功能 0x0 管脚值 0x1 C0+的管脚值 0x2 内部电压参考 0x3 保留
8:5	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
4	ISLVAL	R/W	0	电平触发中断值 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	0x0	中断触发方式 ISEN 位域指定了产生中断的比较器输出的检测方式。检测条件如下： 值 功能 0x0 电平检测，见 ISLVAL 0x1 下降沿 0x2 上升沿 0x3 上升/下降沿

位/域	名称	类型	复位	描述
1	CINV	R/W	0	比较器输出反相 CINV位有条件地将比较器的输出反相。如果该位为0，那么比较器的输出不变。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反相。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

17 脉宽调制器 (PWM)

脉宽调制（PWM）是一项功能强大的技术，它是一种对模拟信号电平进行数字化编码的方法。在脉宽调制中使用高分辨率计数器来产生方波，并且可以通过调整方波的占空比来对模拟信号电平进行编码。PWM通常使用在开关电源（switching power）和电机控制中。

Stellaris® PWM模块由 3个 PWM发生器模块 1个控制模块组成。每个 PWM发生器模块包含1个定时器（16位递减或先递增后递减计数器），2个PWM比较器，PWM信号发生器，死区发生器和中断选择器。而控制模块决定了PWM信号的极性，以及将哪个信号传递到管脚。

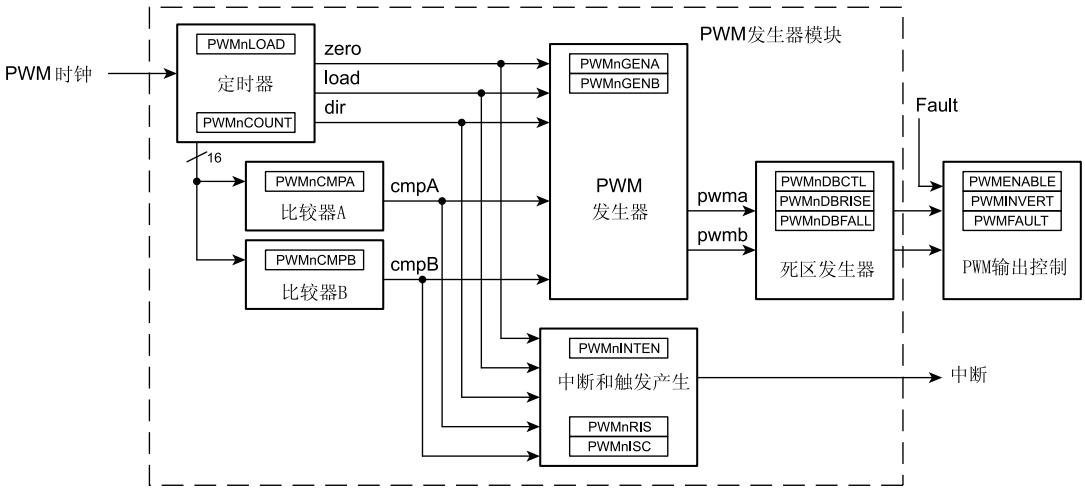
每个 PWM发生器模块产生两个PWM信号，这两个PWM信号可以是独立的信号（基于同一定时器因而频率相同的独立信号除外），也可以是一对插入了死区延迟的互补（complementary）信号。这些PWM发生模块的输出信号在传递到器件管脚之前由输出控制模块管理。

Stellaris® PWM模块具有极大的灵活性。它可以产生简单的PWM信号，如简易充电泵需要的信号；也可以产生带死区延迟的成对PWM信号，如供半-H桥（half-H bridge）驱动电路使用的信号。3个发生器模块也可产生3相反相器桥所需的完整6通道门控。

17.1 结构图

图 17-1 在 400页给出了 Stellaris® PWM模块的结构图。该LM3S2950控制器包含3个发生器模块(PWM0, PWM1和PWM2)，并产生6个独立的PWM信号或3对带死区延时的PWM信号。

图 **17-1. PWM**模块的结构图



17.2 功能描述

17.2.1 PWM定时器

每个PWM发生器的定时器都有两种工作模式：递减计数模式或先递增后递减计数模式。在递减计数模式中，定时器从装载值开始计数，计数到零时又返回到装载值并继续递减计数。在先递增后递减计数模式中，定时器从0开始往上计数，一直计数到装载值，然后从装载值递减到零，接着再递增到装载值，依此类推。通常，递减计数模式是用来产生左对齐或右对齐的PWM信号，而先递增后递减计数模式是用来产生中心对齐的PWM信号。

PWM定时器输出3个信号，这些信号在生成PWM信号的过程中使用：方向信号（在递减计数模式中，该信号始终为低电平，在先递增后递减计数模式中，则是在低高电平之间切换）；当计数器计

数值为0时，一个宽度等于时钟周期的高电平脉冲；当计数器计数值等于装载值时，一个宽度等于时钟周期的高电平脉冲。注：在递减计数模式中，零脉冲之后紧跟着一个装载脉冲。

17.2.2 PWM比较器

每个PWM发生器含两个比较器，用于监控计数器的值；当比较器的值与计数器的值相等时，比较器输出宽度为单时钟周期的高电平脉冲。在先递增后递减计数模式中，比较器在递增和递减计数时都要进行比较，因此必须通过计数器的方向信号来限定。这些限定脉冲在生成PWM信号的过程中使用。如果任一比较器的值大于计数器的装载值，则该比较器永远不会输出高电平脉冲。

图17-2在401页显示的是计数器处于递减计数模式时的行为以及这些脉冲之间的关系。图17-3在402页显示的是计数器处于先递增后递减计数模式时的行为以及这些脉冲之间的关系。

图 **17-2. PWM**递减计数模式

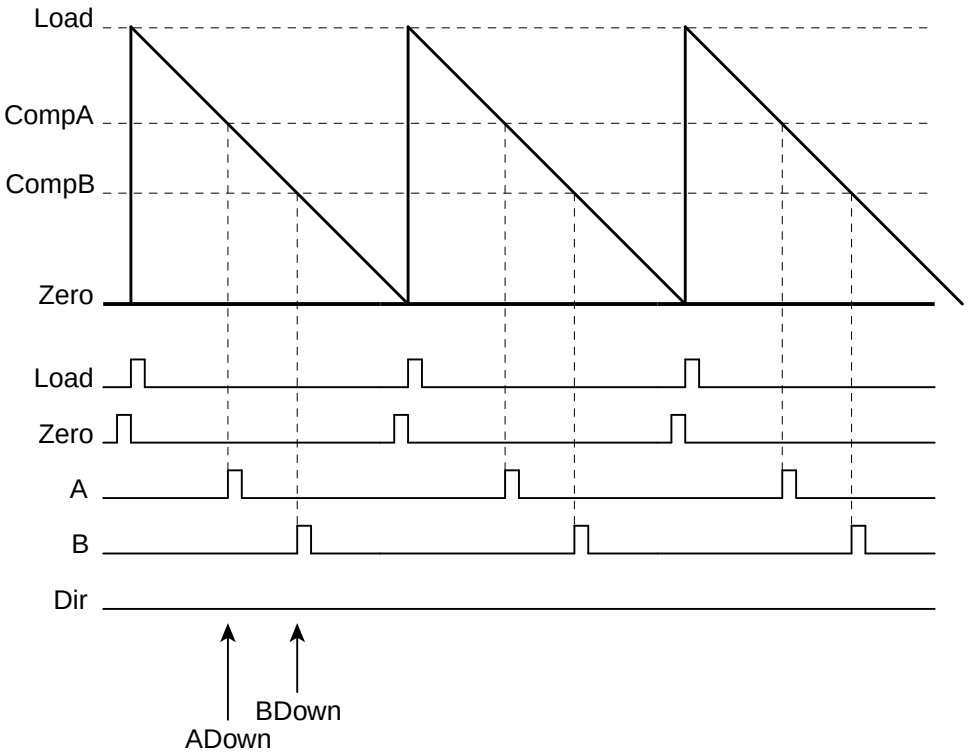
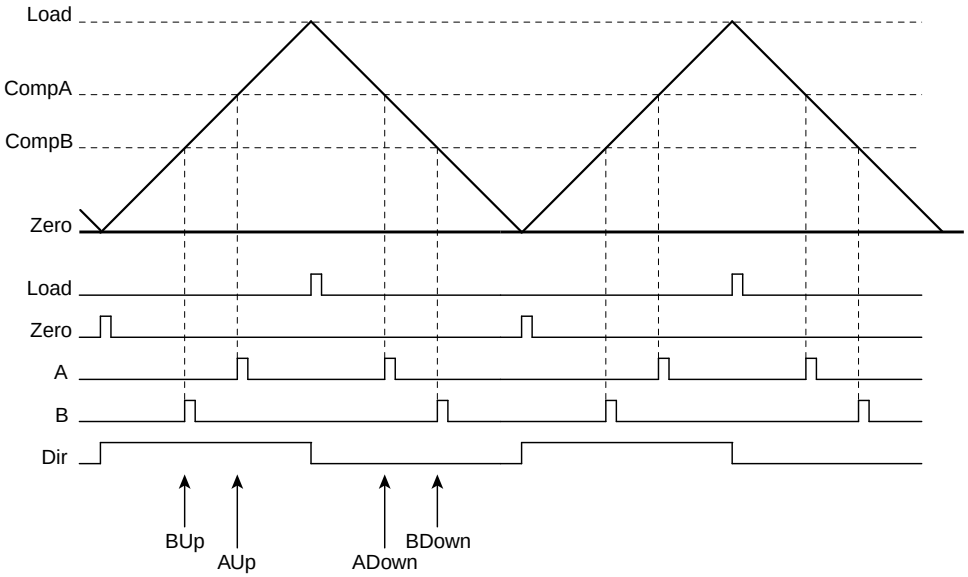


图 **17-3. PWM**先递增后递减计数模式

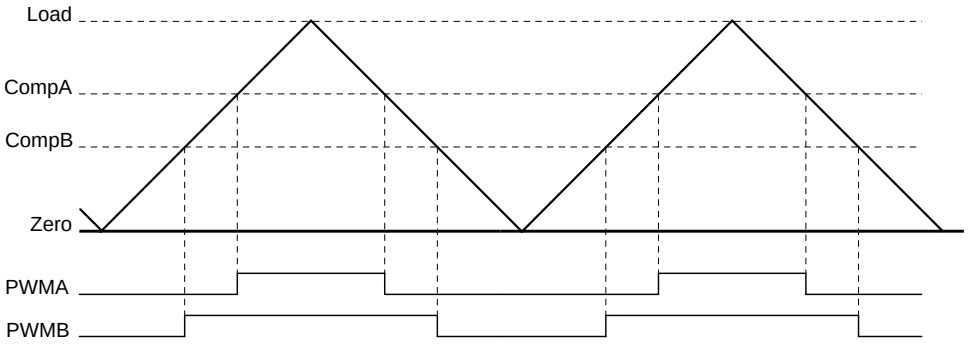


17.2.3 PWM信号发生器

PWM发生器捕获这些脉冲（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：零、装载、匹配A递减、匹配B递减。在先递增后递减计数模式中，能够影响PWM信号的事件有6个：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或匹配B事件与零或装载事件重合时，它们可以被忽略。如果匹配A与匹配B事件重合，则第一个信号PWMA只根据匹配A事件生成，第二个信号PWMB只根据匹配B事件生成。

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。这些动作可用来产生一对不同位置和不同占空比的PWM信号，这对信号可以重叠或不重叠。图17-4 在402页 显示的就是在先递增后递减计数模式产生的一对中心对齐、含不同占空比的重叠PWM信号。

图 **17-4.** 在先递增后递减计数模式中产生**PWM**信号



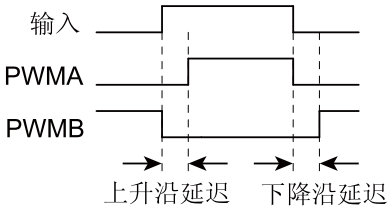
在该示例中，第一个PWM发生器设置为在出现匹配A递增事件时驱动为高电平，出现匹配A递减事件时驱动为低电平，并忽略其它4个事件。第二个发生器设置为在出现匹配B递增事件时驱动为高电平，出现匹配B递减事件时驱动为低电平，并忽略其它4个事件。改变比较器A的值可改变PWMA信号的占空比，改变比较器B的值可改变PWMB信号的占空比。

17.2.4 死区发生器

PWM发生器产生的两个PWM信号被传递到死区发生器。如果死区发生器禁能，则PWM信号只简单地通过该模块，而不会发生改变。如果死区发生器使能，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。

通过上文，我们由此看出：PWMA和PWMB是一对高电平有效的信号，并且其中一个信号总是为高电平，但在跳变处的那段可编程延迟时间除外，都为低电平。这样这两个信号便可用来驱动半-H桥（half-H bridge），又由于它们带有死区延迟，因而还可以避免冲过电流（shoot through current）破坏电力电子（power electronics）。图 17-5 在 403页 显示了死区发生器对输入PWM信号的影响。

图 **17-5. PWM**死区发生器



17.2.5 中断 选择器

PWM发生器还捕获相同的4个（或6个）计数器事件，并使用它们来产生中断。用户可以选择这些事件中的任一个或一组作为中断源；只要其中一个所选事件发生就会产生中断。选择的事件不同，在PWM信号内产生中断 的位置也不同。注：中断 都是基于原始（raw）事件的；而不考虑死区发生器在PWM信号边沿上产生的延迟。

17.2.6 同步方法

具有全局复位功能，该功能可同时复位PWM发生器中的任何或全部计数器。如果多个PWM发生器使用相同的计数器装载值来配置，那么可以保证PWM发生器也具有相同的计数值（这不表示PWM发生器必须在其同步之前被配置）。这样，通过那些信号边沿之间的已知关系可产生2个以上的PWM信号，因为计数器总是具有相同的值。

在PWM发生器中，要对计数器装载值和比较器匹配值进行更新有两种方法。一种是立即更新，计数器计数一到零就立即使用新值。由于要等计数器计数到零才能使用新值，因而在更新过程中定义了一个约定（guaranteed）行为，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，它要等全局同步更新信号有效才使用新值，同步更新信号有效时，计数器一到零就立即使用新值。第二种方法可以同时对多个PWM发生器中的多项进行更新，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先在原来的值上运行。 装载和比较器匹配的更新方法可以在各个PWM发生器模块中单独配置。当那些模块中的定时器同步时，通常可以在PWM发生器模块中使用同步更新机制，尽管该机制正常工作时不要求这个。

17.2.7 故障状态

影响PWM模块的外部条件有两个；一个是故障管脚的信号输入，另一个是由调试器引发的控制器中止。我们可以采用两种机制来处理这些情况，一是强制将输出信号变为无效（inactive）状态，以及/或者让PWM定时器停止运行。

每个输出信号都带有一个故障位。若故障位置位，则故障输入信号将会使相应的输出信号变为无效状态。如果无效状态指的是信号能够长期停留的安全状态，那么这样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。此外，故障条件还可以产生控制器中断。

用户可以将PWM发生器配置为在停止条件期间停止计数。T也可以选择让计数器一直运行，直到计数值为零才停止，或计数值为零时继续计数和重装。停止状态不会产生控制器中断。

17.2.8 输出控制模块

每个PWM发生器模块产生的是两个原始PWM信号，输出控制模块在PWM信号进入管脚之前要对其最后的状态进行控制。通过一个寄存器就能够对实际传递到管脚的PWM信号进行修改。例如，通过对寄存器执行写操作来修改PWM信号（而无需通过修改反馈控制回路来修改各个PWM发生器），以实现无电刷直流电机通信。同样地，故障控制也能够禁能所有的PWM信号。能够对任一PWM信号执行最终的反相操作，使得默认高电平有效的信号变为低电平有效。

17.3 初始化和配置

以下是对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%，假定系统时钟为20MHz。这个实例假定系统时钟为20 MHz。

1. 使能PWM时钟，通过写0x0010.0000到系统控制模块中的 **RCGC0** 寄存器来实现。
2. 使能对应GPIO模块的时钟，通过系统控制模块中的**RCGC2**寄存器来实现。
3. 在GPIO模块中，根据管脚的具体功能，使用**GPIOAFSEL**寄存器来使能相应的管脚。
4. 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 2(000)。
5. 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 0x0000.0000。
   - 向**PWM0GENA**寄存器写入0x0000.008C。
   - 向**PWM0GENB**寄存器写入0x0000.080C。
6. 设置周期。若要得到25KHz频率，即周期为1/25,000（40µs）。PWM时钟源为10MHz；由系统时钟2分频得到。这意味着一个定时器周期要等于400个PWM时钟周期。然后用这个值来设置**PWM0LOAD**寄存器。在递减计数模式中，将**PWM0LOAD**寄存器的Load位域设置为请求的周期减1。
   - 向**PWM0LOAD**寄存器写入0x0000.018F。
7. 将PWM0管脚的脉冲宽度设置为25%占空比。
   - 向**PWM0CMPA**寄存器写入0x0000.012B。
8. 将PWM1 管脚的脉冲宽度设置为75%占空比。
   - 向**PWM0CMPB**寄存器写入0x0000.0063。
9. 启动PWM发生器0中的定时器。
   - 向**PWM0CTL**寄存器写入0x0000.0001。
10. 使能PWM输出。
    - 向**PWMENABLE**寄存器写入0x0000.0003。

17.4 寄存器映射

表 17-1 在 405页列出了PWM寄存器表中所列偏移量是寄存器地址相对于PWM基址0x4002.8000的十六进制增量。

表 **17-1. PWM** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	PWMCTL	R/W	0x0000.0000	PWM主控	407
0x004	PWMSYNC	R/W	0x0000.0000	PWM时基同步	408
0x008	PWMENABLE	R/W	0x0000.0000	PWM输出使能	409
0x00C	PWMINVERT	R/W	0x0000.0000	PWM输出反相	410
0x010	PWMFAULT	R/W	0x0000.0000	PWM输出故障	411
0x014	PWMINTEN	R/W	0x0000.0000	PWM中断使能	412
0x018	PWMRIS	RO	0x0000.0000	PWM原始中断状态	413
0x01C	PWMISC	R/W1C	0x0000.0000	PWM中断状态和清除	414
0x020	PWMSTATUS	RO	0x0000.0000	PWM状态	415
0x040	PWM0CTL	R/W	0x0000.0000	PWM0控制	416
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0中断 使能	418
0x048	PWM0RIS	RO	0x0000.0000	PWM0 原始中断状态	420
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0中断状态和清零	421
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0装载	422
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 计数器	423
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0比较器A	424
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0比较器B	425
0x060	PWM0GENA	R/W	0x0000.0000	PWM0发生器A控制	426
0x064	PWM0GENB	R/W	0x0000.0000	PWM0发生器B控制	429
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0死区控制	432
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0死区上升沿延迟	433
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0死区下降沿延迟	434
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 控制	416
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1中断 使能	418
0x088	PWM1RIS	RO	0x0000.0000	PWM1 原始中断状态	420
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 中断状态和清零	421
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1装载	422
0x094	PWM1COUNT	RO	0x0000.0000	PWM1计数器	423
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1比较器A	424

偏移量	名称	类型	复位	描述	见页面
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1比较器B	425
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1发生器A控制	426
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1发生器B控制	429
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1死区控制	432
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1死区上升沿延迟	433
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1死区下降沿延迟	434
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2控制	416
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2中断使能	418
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 原始中断状态	420
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2中断状态和清零	421
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2装载	422
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2计数器	423
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2比较器A	424
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2比较器B	425
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2发生器A控制	426
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2发生器B控制	429
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2死区控制	432
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2死区上升沿延迟	433
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2死区下降沿延迟	434

17.5 寄存器描述

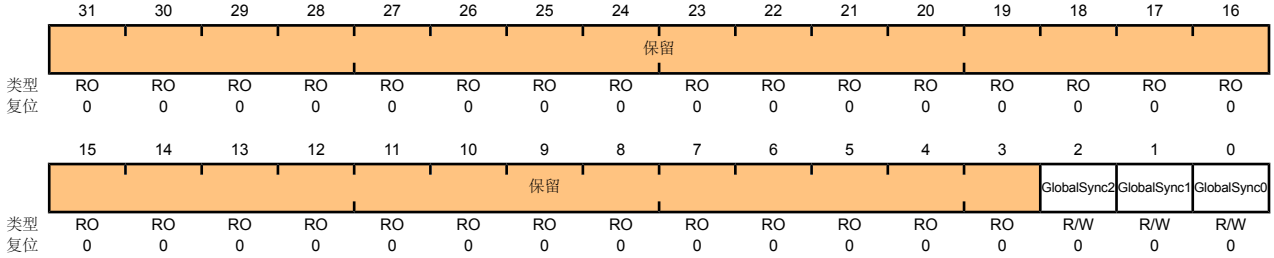
下文将按地址偏移量的数字顺序列出PWM寄存器，并对它们进行描述。

寄存器 1: PWM主控 （PWMCTL），偏移量 0x000

该寄存器对PWM发生模块进行主控。

PWM主控 (PWMCTL)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



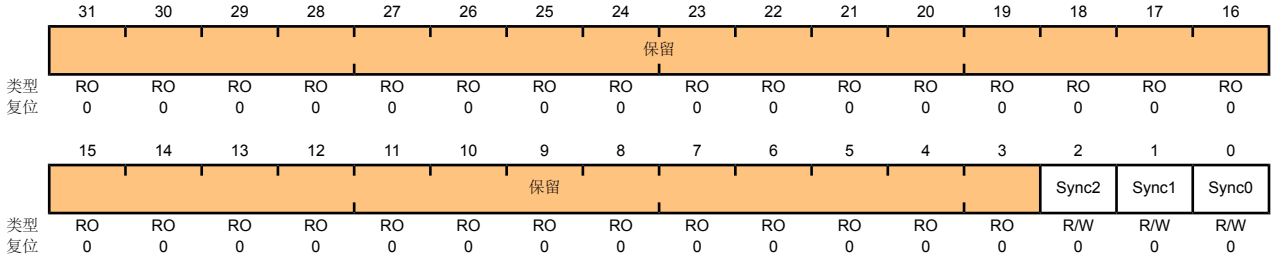
位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	GlobalSync2	R/W	0	更新PWM发生器2 与 `GlobalSync0`相同，但用于PWM发生器2。
1	GlobalSync1	R/W	0	更新PWM发生器1 与 `GlobalSync0`相同，但用于PWM发生器1。
0	GlobalSync0	R/W	0	更新PWM发生器0 该位置位时，装载或PWM发生器0中的比较寄存器在下一次对应的计数器变为0时才进行更新。更新结束时该位自动清零，而不能由软件来清零。

寄存器 **2: PWM**时基同步（**PWMSYNC**），偏移量 **0x004**

该寄存器提供一种对PWM发生模块中的计数器进行同步的方法。向该寄存器的某个位写**1**会让指定的计数器复位为零，而如果同时向多个位写**1**会让多个计数器同时复位。复位之后，寄存器中的位自动清零，读这些位时如果返回零，则表示同步已完成。

PWM时基同步 (PWMSYNC)

偏移量 0x004
类型 R/W, 复位 0x0000.0000



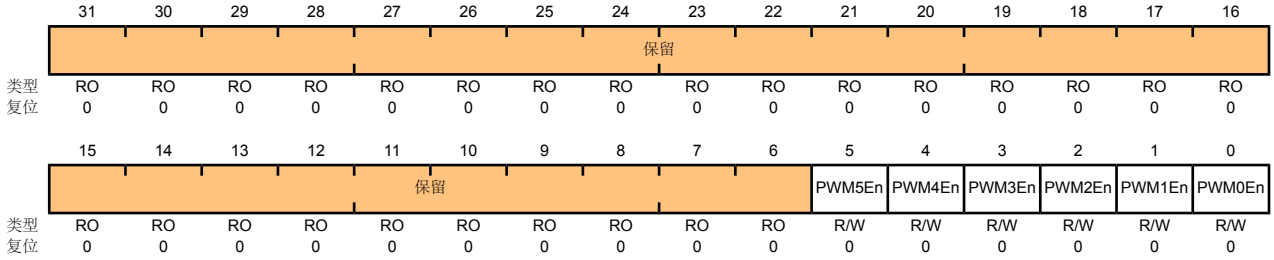
位/域	名称	类型	复位	描述
31:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	Sync2	R/W	0	复位发生器**2**计数器 对PWM发生器**2**的计数器进行复位。
1	Sync1	R/W	0	复位发生器**1**计数器 对PWM发生器**1**的计数器进行复位。
0	Sync0	R/W	0	复位发生器**0**计数器 对PWM发生器**0**的计数器进行复位。

寄存器 3: PWM输出使能（PWMENABLE），偏移量 0x008

该寄存器主要控制将哪一个已产生的PWM信号输出到器件管脚。如果禁止PWM输出，PWM信号就不会传递给管脚，此时可以继续生成PWM信号（例如在时基同步时）。当寄存器中的位置位时，将对应的PWM信号传递到由**PWMINVERT**寄存器控制的输出级（output stage）。没有置位时，PWM信号用0代替，同样可以传递到输出级。

PWM输出使能 (PWMENABLE)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



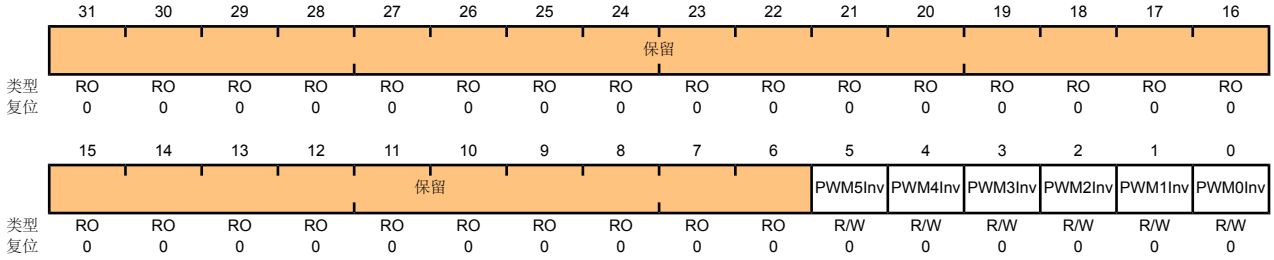
位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	PWM5En	R/W	0	PWM5输出使能 该位置位时，允许将产生的信号PWM5传递到器件管脚。
4	PWM4En	R/W	0	PWM4输出使能 该位置位时，允许将产生的信号PWM4传递到器件管脚。
3	PWM3En	R/W	0	PWM3输出使能 该位置位时，允许将产生的信号PWM3传递到器件管脚。
2	PWM2En	R/W	0	PWM2输出使能 该位置位时，允许将产生的信号PWM2传递到器件管脚。
1	PWM1En	R/W	0	PWM1输出使能 该位置位时，允许将产生的信号PWM1传递到器件管脚。
0	PWM0En	R/W	0	PWM0输出使能 该位置位时，允许将产生的信号PWM0传递到器件管脚。

寄存器 **4: PWM**输出反相（**PWMINVERT**），偏移量 **0x00C**

该寄存器对器件管脚上的PWM信号的极性进行控制。由死区模块生成的PWM信号为高电平有效，通过本寄存器可以将其变为低电平有效。而被禁能的PWM通道也通过输出反相器被传递（假如这样配置的话），这样，不活动的通道也保持了其正确的极性。

PWM输出反相 (PWMINVERT)

偏移量 0x00C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	PWM5Inv	R/W	0	PWM5信号反相 该位置位时，将产生的信号PWM5反相。
4	PWM4Inv	R/W	0	PWM4信号反相 该位置位时，将产生的信号PWM4反相。
3	PWM3Inv	R/W	0	PWM3信号反相 该位置位时，将产生的信号PWM3反相。
2	PWM2Inv	R/W	0	PWM2信号反相 该位置位时，将产生的信号PWM2反相。
1	PWM1Inv	R/W	0	PWM1信号反相 该位置位时，将产生的信号PWM1反相。
0	PWM0Inv	R/W	0	PWM0信号反相 该位置位时，将产生的信号PWM0反相。

寄存器 5: PWM输出故障（PWMFAULT），偏移量 0x010

该寄存器用来控制出现故障情况时的PWM输出行为。故障输入和调试事件均可看作故障条件。在故障条件下，每个PWM信号均可以直接通过，而无需修改或驱动为低电平。对于配置为直通（pass-through）的输出来说，处理相应PWM发生器的调试事件还决定是否继续生成PWM信号。

若在输出反相器之前对故障条件进行控制，那么如果通道被配置成反相，在故障条件下驱动为低电平的PWM信号将被反相（即，该管脚在存在故障情况时驱动为高电平）。

PWM输出故障 (PWMFAULT)

偏移量 0x010
类型 R/W, 复位 0x0000.0000

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
域	保留															
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
域	保留										Fault5	Fault4	Fault3	Fault2	Fault1	Fault0
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

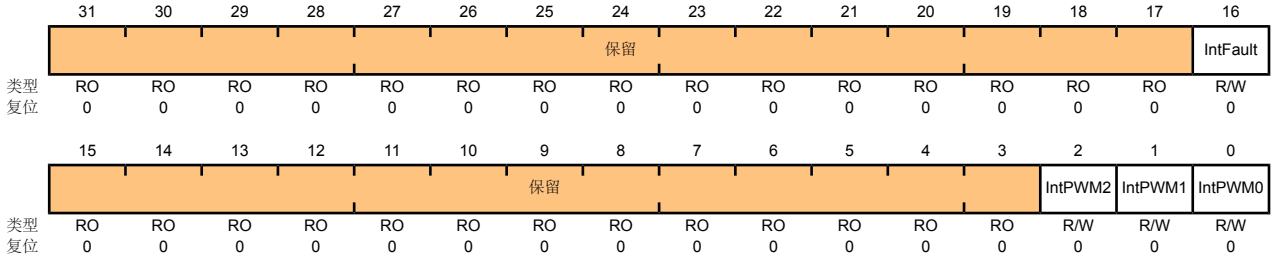
位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	Fault5	R/W	0	PWM5 在故障时驱动为低电平 该位置位时，输出信号PWM5在故障状态上驱动为低电平。
4	Fault4	R/W	0	PWM4在故障时驱动为低电平 该位置位时，输出信号PWM4在故障状态上驱动为低电平。
3	Fault3	R/W	0	PWM3在故障时驱动为低电平 该位置位时，输出信号PWM3在故障状态上驱动为低电平。
2	Fault2	R/W	0	PWM2在故障时驱动为低电平 该位置位时，输出信号PWM2在故障状态上驱动为低电平。
1	Fault1	R/W	0	PWM1在故障时驱动为低电平 该位置位时，输出信号PWM1在故障状态上驱动为低电平。
0	Fault0	R/W	0	PWM0在故障时驱动为低电平 该位置位时，输出信号PWM0在故障状态上驱动为低电平。

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)

偏移量 0x014
类型 R/W, 复位 0x0000.0000



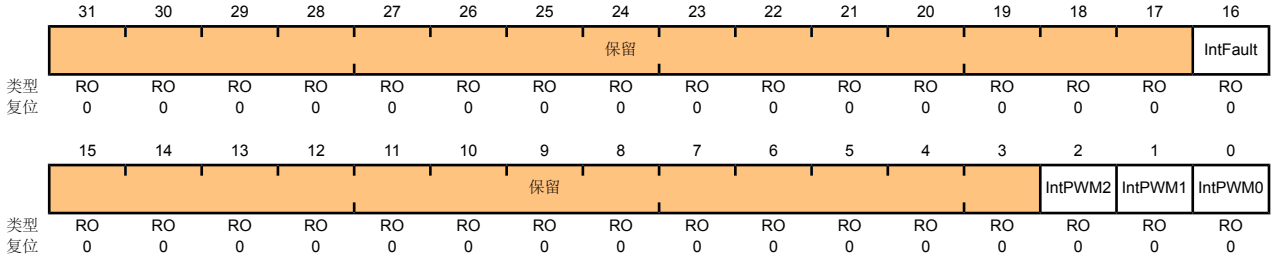
位/域	名称	类型	复位	描述
31:17	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
16	IntFault	R/W	0	故障中断使能位 如果该位为1，则在故障输入有效时产生中断。
15:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IntPWM2	R/W	0	PWM2中断使能位 如果该位为1，则在PWM发生器2模块发出中断时产生中断。
1	IntPWM1	R/W	0	PWM1中断使能位 如果该位为1，则在PWM发生器1模块发出中断时产生中断。
0	IntPWM0	R/W	0	PWM0中断使能位 如果该位为1，则在PWM发生器0模块发出中断时产生中断。

寄存器 7: PWM原始中断状态（PWMRIS），偏移量 0x018

该寄存器提供的是已发出的中断源的当前设置，而不管它们是否被提交给控制器。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(见 414页)来清零。PWM发生器中断只简单地反映PWM发生器的状态，它们通过PWM发生器模块中的中断状态寄存器进行清零。寄存器中设为1的位表示活动的事件，0位表示所述事件处于不活动状态。

PWM原始中断状态 (PWMRIS)

偏移量 0x018
类型 RO, 复位 0x0000.0000



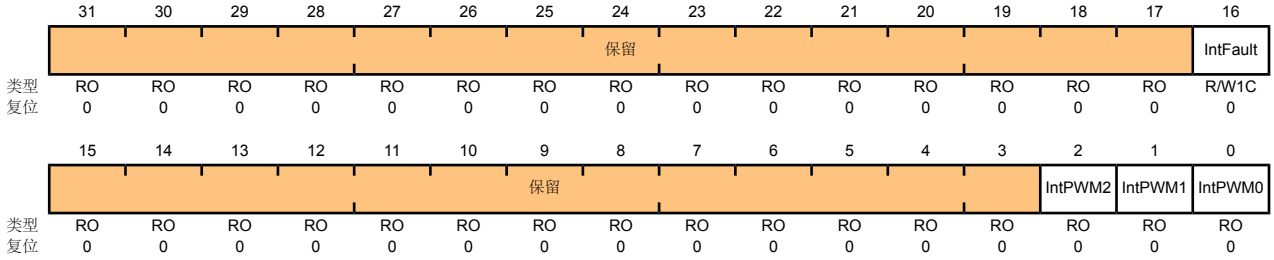
位/域	名称	类型	复位	描述
31:17	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
16	IntFault	RO	0	故障中断有效 表示故障输入端有效。
15:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IntPWM2	RO	0	PWM2中断有效 表示PWM发生器2模块正发出中断。
1	IntPWM1	RO	0	PWM1中断有效 表示PWM发生器1模块正发出中断。
0	IntPWM0	RO	0	PWM0中断有效 表示PWM发生器0模块正发出中断。

寄存器 **8: PWM**中断状态和清除（**PWMISC**），偏移量 **0x01C**

该寄存器汇总了单个PWM发生器模块的中断状态。寄存器中的位为1，表示对应的发生器模块正在发出中断。而通过查询每个模块中的单个中断状态寄存器可以确定中断原因和清除中断。对于故障中断来说，向该位写1即可将锁存的中断状态清零。

PWM中断状态和清除 (PWMISC)

偏移量 0x01C
类型 R/W1C, 复位 0x0000.0000



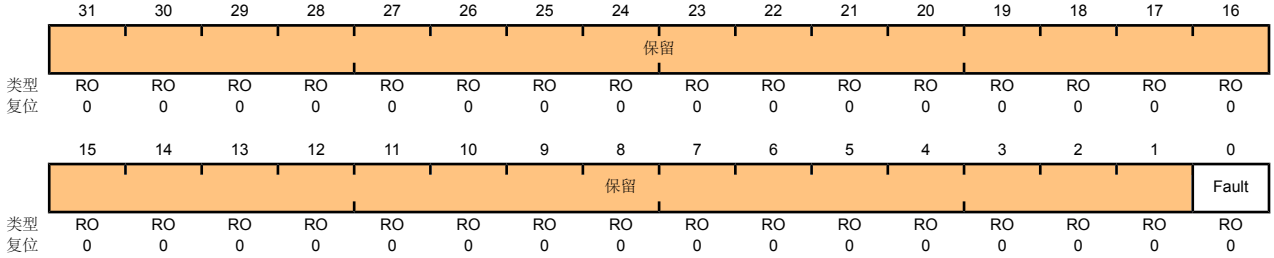
位/域	名称	类型	复位	描述
31:17	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
16	IntFault	R/W1C	0	故障中断有效 表示故障输入是否正在发出中断。
15:3	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2	IntPWM2	RO	0	PWM2中断状态 表示PWM发生器2模块是否正在发出中断。
1	IntPWM1	RO	0	PWM1中断状态 表示PWM发生器1模块是否正在发出中断。
0	IntPWM0	RO	0	PWM0中断状态 表示PWM发生器0模块是否正在发出中断。

寄存器 9: PWM状态（PWMSTATUS），偏移量 0x020

该寄存器用于显示故障输入信号的状态。

PWM状态 (PWMSTATUS)

偏移量 0x020
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	Fault	RO	0	故障中断状态 该位置位表示故障输入端变有效。

寄存器 **10: PWM0**控制（**PWM0CTL**），偏移量 **0x040**

寄存器 **11: PWM1** 控制 （**PWM1CTL**），偏移量 **0x080**

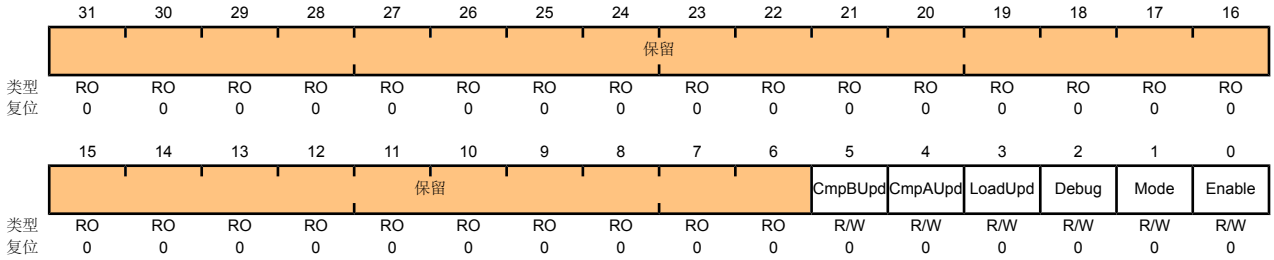
寄存器 **12: PWM2**控制（**PWM2CTL**），偏移量 **0x0C0**

这些寄存器用来配置PWM信号发生模块 (PWM0CTL控制PWM发生器0模块，依此类推)。寄存器的更新模式、调试模式、计数模式和模块使能模式都是通过这些寄存器控制。该模块可以产生两个独立的PWM信号（来自同一个计数器），或一对增加了死区延迟的PWM信号。

PWM0模块产生PWM0和PWM1输出信号，PWM1 模块产生PWM2和PWM3输出信号，PWM2模块产生PWM4和PWM5输出信号。

PWM0控制 (PWM0CTL)

偏移量 0x040
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	CmpBUpd	R/W	0	比较器B更新方法 和CmpAUpd 相同，但仅适用于比较器B寄存器。
4	CmpAUpd	R/W	0	比较器A更新方法 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下一次计数器到达0时反映给比较器。如果为1，则将延迟寄存器更新，直到已通过**PWM**主机控制 **(PWMCTL)** 寄存器 (见 407页)请求了同步更新之后，下一次计数器到达0时才反映给比较器。
3	LoadUpd	R/W	0	装载寄存器更新方法 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下一次计数器到达0时反映给计数器。如果为1，则将延迟寄存器更新，直到已通过**PWM**主机控制 **(PWMCTL)**寄存器请求了同步更新之后，下一次计数器到达0时才反映给计数器。
2	Debug	R/W	0	调试模式 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，并当不再处于调试模式时，再次继续运行。如果为1，则计数器一直运行。
1	Mode	R/W	0	计数模式 计数器的计数模式。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，则计数器从0开始递增计数到装载值，然后再递减到0，接着重复该过程（先递增后递减计数模式）。

位/域	名称	类型	复位	描述
0	Enable	R/W	0	PWM模块使能 PWM发生模块的主机使能。如果该位为0，则整个模块禁止，并且不计时。如果为1，则模块使能并产生PWM信号。

寄存器 **13: PWM0**中断 使能 （**PWM0INTEN**），偏移量 **0x044**

寄存器 **14: PWM1**中断 使能（**PWM1INTEN**），偏移量 **0x084**

寄存器 **15: PWM2**中断使能（**PWM2INTEN**），偏移量 **0x0C4**

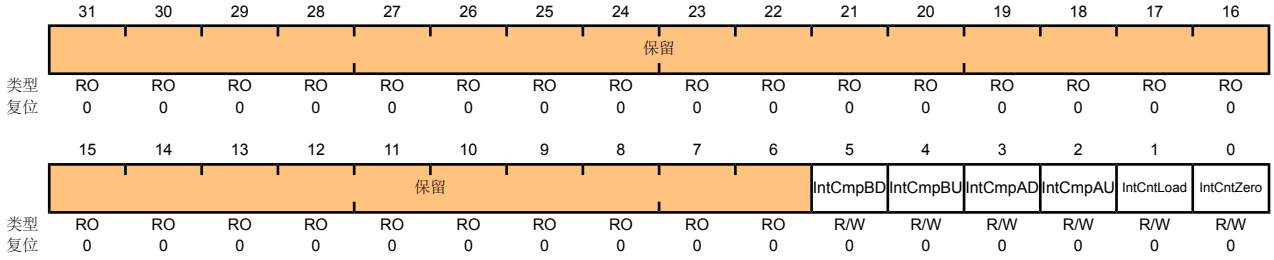
这些寄存器控制PWM发生器 (**PWM0INTEN** 控制PWM发生器0模块，依此类推)产生中断 的能力。能够引起中断的事件包括：

- 计数器的计数值等于装载寄存器的装载值
- 计数器的计数值等于零
- 递增计数时，计数器的计数值等于比较器A寄存器中的值
- 递减计数时，计数器的计数值等于比较器A寄存器中的值
- 递增计数时，计数器的计数值等于比较器B寄存器中的值
- 递减计数时，计数器的计数值等于比较器B寄存器中的值

上述事件的任何组合都可以产生中断。

PWM0中断 使能 (PWM0INTEN)

偏移量 0x044
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	IntCmpBD	R/W	0	计数器=比较器B并递减计数时产生中断 如果该位为1，则当计数器的计数值等于比较器B的值并且计数器正在递减计数时，产生中断。
4	IntCmpBU	R/W	0	计数器=比较器B并递增计数时产生中断 如果该位为1，则当计数器的计数值等于比较器B的值并且计数器正在递增计数时，产生中断。
3	IntCmpAD	R/W	0	计数器=比较器A并递减计数时产生中断 如果该位为1，则当计数器的计数值等于比较器A的值并且计数器正在递减计数时，产生中断。
2	IntCmpAU	R/W	0	计数器=比较器A并递增计数时产生中断 如果该位为1，则当计数器的计数值等于比较器A的值并且计数器正在递增计数时，产生中断。

位/域	名称	类型	复位	描述
1	IntCntLoad	R/W	0	计数器=装载值时产生中断 如果该位为1，则当计数器的计数值等于**PWMnLOAD**寄存器的值时，产生中断。
0	IntCntZero	R/W	0	计数器=0时产生中断 如果该位为1，则当计数器的计数值为零时产生中断。

寄存器 **16: PWM0** 原始中断状态（**PWM0RIS**），偏移量 **0x048**

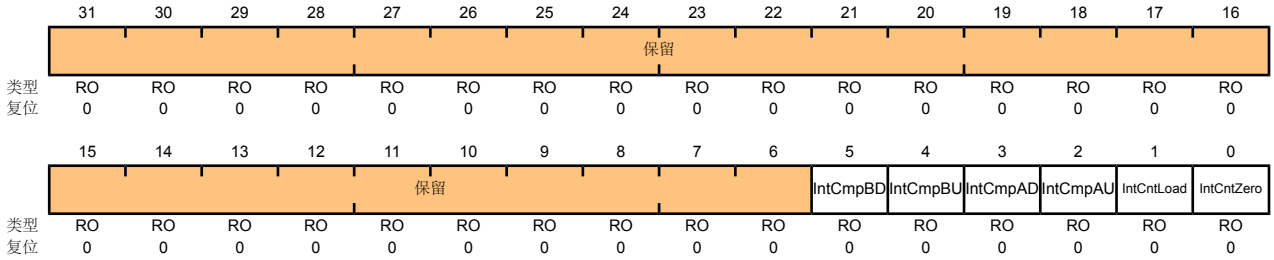
寄存器 **17: PWM1** 原始中断状态 （**PWM1RIS**），偏移量 **0x088**

寄存器 **18: PWM2** 原始中断状态（**PWM2RIS**），偏移量 **0x0C8**

这些寄存器显示的是 已发出的中断源的当前状态，而不管中断是否被提交给控制器 (**PWM0RIS**控制PWM发生器0模块，依此类推)。寄存器中的位为1表示已发生锁存事件；为0表示所述事件没有发生。

PWM0 原始中断状态 (PWM0RIS)

偏移量 0x048
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	IntCmpBD	RO	0	比较器B递减中断状态 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	比较器B递增中断状态 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	比较器A递减中断状态 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	比较器A递增中断状态 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	计数器=装载值时的中断状态 表示计数器已与**PWMnLOAD**寄存器的值相等。
0	IntCntZero	RO	0	计数器=0时的中断状态 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

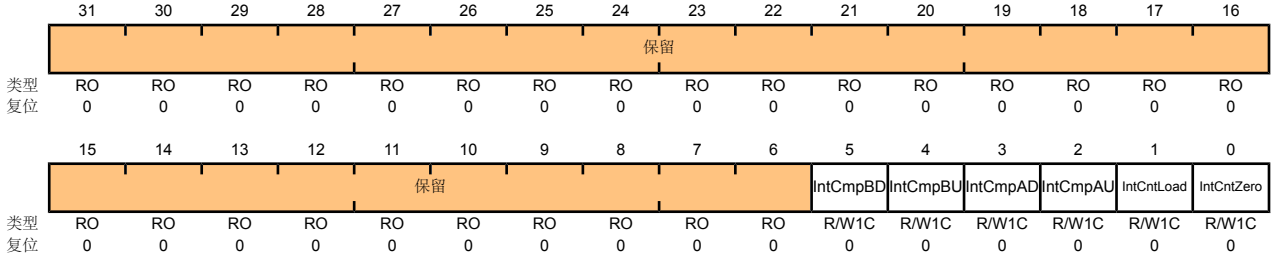
寄存器 20: PWM1 中断状态和清零 （PWM1ISC），偏移量 0x08C

寄存器 21: PWM2中断状态和清零（PWM2ISC），偏移量 0x0CC

这些寄存器提供的是 已发出给控制器的中断源的当前设置(**PWM0ISC**控制PWM发生器0模块，依此类推)。寄存器中的位为1表示已发生锁存事件；为0表示所述事件没有发生。它们都是R/W1C，即向某个位写1将使对应的中断原因清零。

PWM0中断状态和清零 (PWM0ISC)

偏移量 0x04C
类型 R/W1C, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:6	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	IntCmpBD	R/W1C	0	比较器B递减中断 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	比较器B递增中断 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	比较器A递减中断 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	比较器A递增中断 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	计数器=装载时的中断 表示计数器已与**PWMnLOAD**寄存器的值相等。
0	IntCntZero	R/W1C	0	计数器=0时的中断 表示计数器已等于0。

寄存器 **22: PWM0**装载（**PWM0LOAD**），偏移量 **0x050**

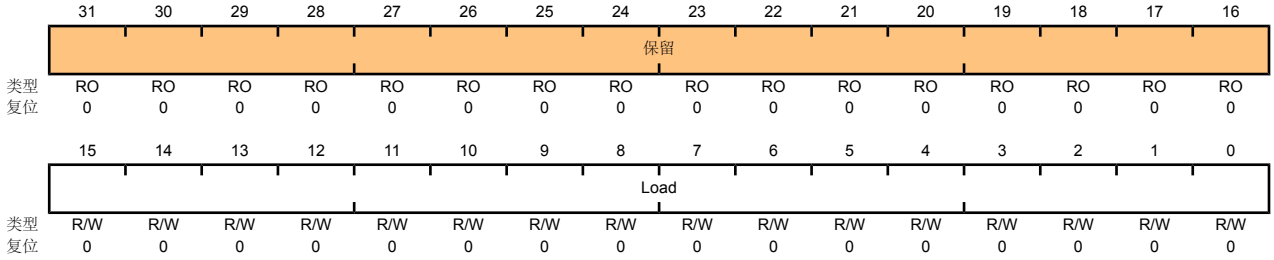
寄存器 **23: PWM1**装载（**PWM1LOAD**），偏移量 **0x090**

寄存器 **24: PWM2**装载（**PWM2LOAD**），偏移量 **0x0D0**

这些寄存器包含的是 PWM计数器的装载值 (**PWM0LOAD** 控制PWM发生器0模块，依此类推)。根据计数模式，该值可以在计数器到达零之后加载到计数器，也可以在计数器递减到零之后，作为递增计数的极限值。如果装载值的更新模式为立即模式，则该值在下一次计数器到达零时使用；如果为同步模式，则在通过**PWM**主控**(PWMCTL)**寄存器(见 407页)请求了同步更新之后，下一次计数器到达零时使用。如果在实际更新装载值之前重写该寄存器，则之前的值不再使用并丢失。

PWM0装载 (PWM0LOAD)

偏移量 0x050
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	Load	R/W	0	计数器装载值 计数器的装载值。

寄存器 25: PWM0 计数器（PWM0COUNT），偏移量 0x054

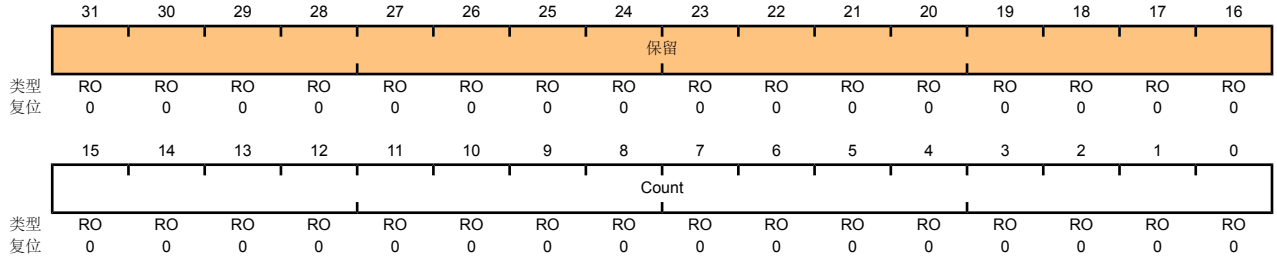
寄存器 26: PWM1计数器（PWM1COUNT），偏移量 0x094

寄存器 27: PWM2计数器（PWM2COUNT），偏移量 0x0D4

这些寄存器包含的是PWM计数器的当前值(**PWM0COUNT**是PWM发生器0模块的值，依此类推)。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，见 426页 和 429页），或驱动中断（通过**PWMnINTEN**寄存器，见 418页）。当该值为零时，将产生具有相同功能的脉冲。

PWM0 计数器 (PWM0COUNT)

偏移量 0x054
类型 RO, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	Count	RO	0x00	计数值 计数器的当前值。

寄存器 **28: PWM0**比较器**A**（**PWM0CMPA**），偏移量 **0x058**

寄存器 **29: PWM1**比较器**A**（**PWM1CMPA**），偏移量**0x098**

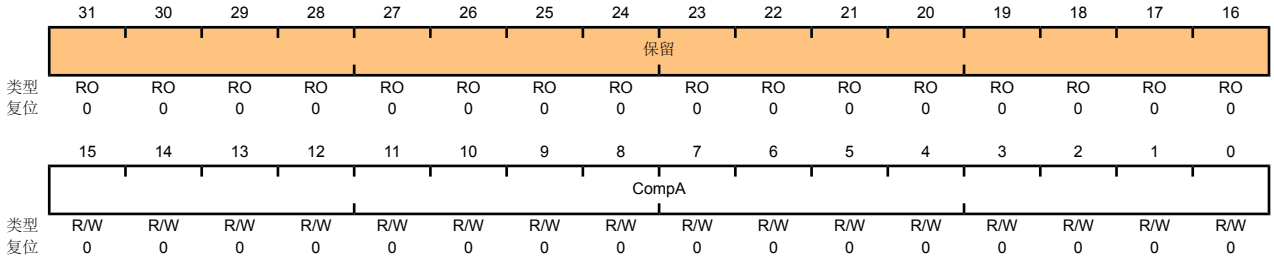
寄存器 **30: PWM2**比较器**A**（**PWM2CMPA**），偏移量 **0x0D8**

这些寄存器包含的是 与计数器进行比较的值 (**PWM0CMPA** 控制PWM发生器0模块，依此类推)。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），或驱动中断（通过**PWMnINTEN**寄存器）。如果该寄存器的值大于 **PWMnLOAD**寄存器(见 422页)的值，则始终不输出脉冲。

对于比较器A来说，如果更新模式为立即模式（根据**PWMnCTL**寄存器的CmpAUpd位），那么该寄存器的16位CompA将在下一次计数器到达零时使用。如果为同步更新，则在通过**PWM**主控**(PWMCTL)**寄存器 (见 407页)请求了同步更新之后，下一次计数器到达零时使用。如果在实际的更新之前重写该寄存器，则之前的值不再使用并丢失。

PWM0比较器A (PWM0CMPA)

偏移量 0x058
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	CompA	R/W	0x00	比较器A的值 与计数器进行比较的值。

寄存器 31: PWM0比较器B（PWM0CMPB），偏移量 0x05C

寄存器 32: PWM1比较器B（PWM1CMPB），偏移量 0x09C

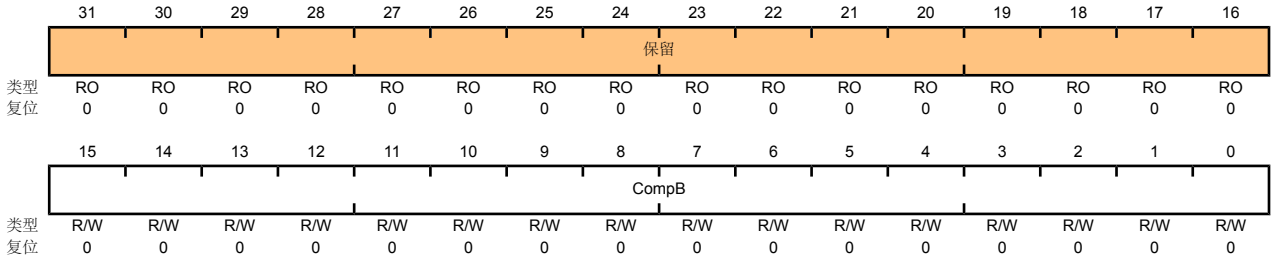
寄存器 33: PWM2比较器B（PWM2CMPB），偏移量 0x0DC

这些寄存器包含的是 与计数器进行比较的值 (**PWM0CMPB** 控制PWM发生器0模块，依此类推)。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），或驱动中断（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来说，如果更新模式为立即模式（根据 **PWMnCTL**寄存器的CmpBUpd位），那么该寄存器的16位CompB将在下一次计数器到达零时使用。如果为同步更新，则在通过**PWM**主控 **(PWMCTL)** 寄存器 (见 407页)请求了同步更新之后，下一次计数器到达零时使用。如果在实际的更新之前重写该寄存器，则之前的值不再使用并丢失。

PWM0比较器B (PWM0CMPB)

偏移量 0x05C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:16	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	CompB	R/W	0x00	比较器B的值 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**

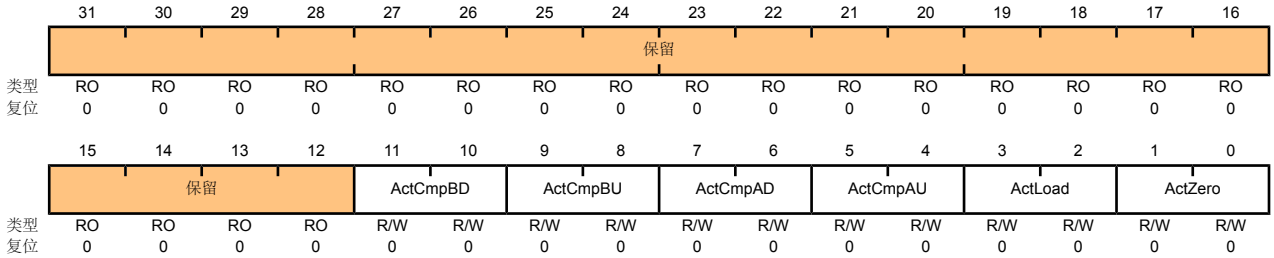
这些寄存器 根据计数器的装载输出脉冲和零输出脉冲以及比较器的比较A脉冲和比较B脉冲来控制PWMnA信号的产生 (**PWM0GENA** 控制PWM发生器0模块，依此类推)。当计数器处于递减计数模式时，只出现4个事件；当处于先递增后递减模式时，共出现6个事件。这些事件在确定PWM信号产生的位置及信号的占空比时具有极大的灵活性。

PWM0GENA 寄存器控制 PWM0A 信号的产生; **PWM1GENA**信号, PWM1A 信号的产生; 以及 **PWM2GENA**信号, PWM2A 信号的产生。

如果零或装载事件与比较A或比较B事件相同，则发生零或装载动作，比较A或比较B忽略。如果比较A事件与比较B事件相同，则发生比较A动作，比较B忽略。

PWM0发生器A控制 (PWM0GENA)

偏移量 0x060
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:12	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11:10	ActCmpBD	R/W	0x0	比较器B递减时的动作 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。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。

位/域	名称	类型	复位	描述
9:8	ActCmpBU	R/W	0x0	比较器B递增时的动作 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。只在**PWMnCTL**寄存器(见 416页)的Mode位设置为1时才发生。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
7:6	ActCmpAD	R/W	0x0	比较器A递增时的动作 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。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
5:4	ActCmpAU	R/W	0x0	比较器A递增时的动作 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。只在**PWMnCTL**寄存器的Mode位设置为1时才发生。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
3:2	ActLoad	R/W	0x0	计数器=装载值时的动作 当计数器的值与装载值相等时采取的动作。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。

位/域	名称	类型	复位	描述
1:0	ActZero	R/W	0x0	计数器=0时的动作 当计数器的值为零时采取的动作。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 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

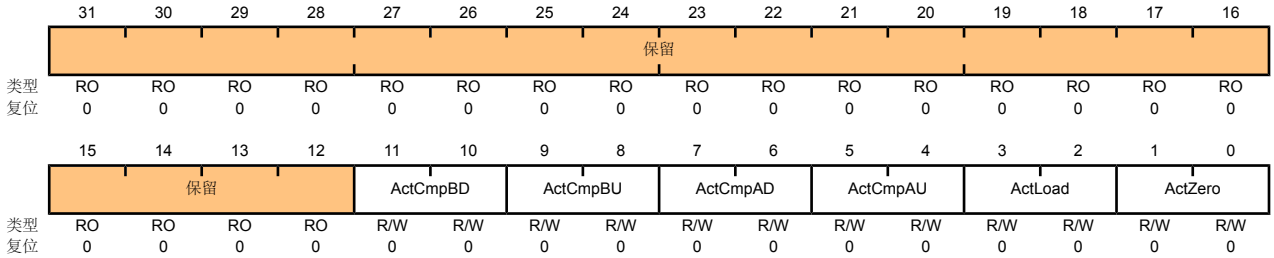
这些寄存器 根据来自计数器的装载输出脉冲和零输出脉冲以及来自比较器的比较A脉冲和比较B脉冲来控制PWMnB信号的产生 (**PWM0GENB**控制PWM发生器0模块，依此类推)。当计数器处于递减计数模式时，只出现4个事件；当处于先递增后递减模式时，共出现6个事件。这些事件在确定PWM信号产生的位置及信号的占空比时具有极大的灵活性。

PWM0GENB寄存器控制 PWM0B 信号的产生; **PWM1GENB**信号、PWM1B 信号; 以及 **PWM2GENB**信号、PWM2B 信号的产生。

如果零或装载事件与比较A或比较B事件相同，则发生零或装载动作，比较A或比较B忽略。如果零或装载事件与比较A或比较B事件相同，则发生零或装载动作，比较A或比较B忽略。

PWM0发生器B控制 (PWM0GENB)

偏移量 0x064
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:12	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11:10	ActCmpBD	R/W	0x0	比较器B递减时的动作 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。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。

位/域	名称	类型	复位	描述
9:8	ActCmpBU	R/W	0x0	比较器B递增时的动作 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。只在**PWMnCTL**寄存器的Mode位设置为1时才发生。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
7:6	ActCmpAD	R/W	0x0	比较器A递增时的动作 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。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
5:4	ActCmpAU	R/W	0x0	比较器A递增时的动作 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。只在**PWMnCTL**寄存器的Mode位设置为1时才发生。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。
3:2	ActLoad	R/W	0x0	计数器=装载值时的动作 当计数器的值与装载值相等时采取的动作。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。

位/域	名称	类型	复位	描述
1:0	ActZero	R/W	0x0	计数器=0时的动作 当计数器的值为零时采取的动作。 下表定义了输出信号上事件的作用。 值 描述 0x0 不动作。 0x1 将输出信号反相。 0x2 将输出信号设置为0。 0x3 将输出信号设置为1。

寄存器 **40: PWM0**死区控制 （**PWM0DBCTL**），偏移量 **0x068**

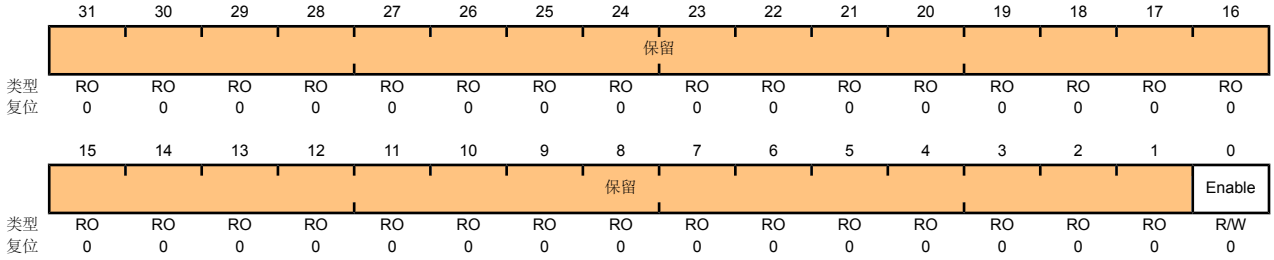
寄存器 **41: PWM1**死区控制 （**PWM1DBCTL**），偏移量 **0x0A8**

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PWM0A和PWM0B信号来产生信号PWM0和 PWM1信号。当死区发生器被禁能时，PWM0A信号直接通过死区模块成为PWM0信号，PWM0B信号直接通过死区模块成为PWM1信号。当死区发生器使能以及结果波形反相时，忽略PWM0B信号；延迟PWM0A信号的上升沿来产生PWM0，延迟时间由**PWM0DBRISE**寄存器(见 433页)的值确定，并延迟PWM0A信号的下降沿来产生PWM1信号，延迟时间由**PWM0DBFALL**寄存器 (见 434页)的值确定。 采用与上文相同的方式, 通过PWM1A 和 PWM1B信号来产生PWM2和PWM3 信号， 通过PWM2A和PWM2B信号来产生PWM4和 PWM5信号。

PWM0死区控制 (PWM0DBCTL)

偏移量 0x068
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:1	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	Enable	R/W	0	死区发生器使能 该位置位时，死区发生器将死区插入到输出信号中；为零时，只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**

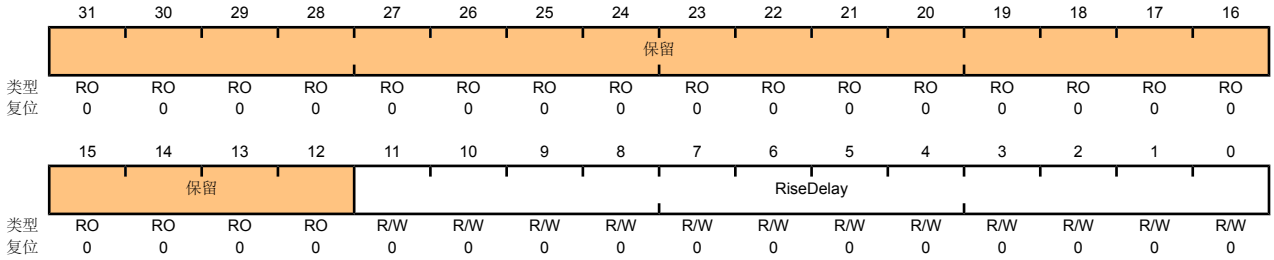
寄存器 **44: PWM1**死区上升沿延迟（**PWM1DBRISE**），偏移量 **0x0AC**

寄存器 **45: PWM2**死区上升沿延迟（**PWM2DBRISE**），偏移量 **0x0EC**

PWM0DBRISE 寄存器包含的是在生成PWM0信号时，信号PWM0A上升沿延迟的时钟数。如果通过**PWMnDBCTL**寄存器将死区发生器禁能，那么将忽略寄存器**PWM0DBRISE**。如果该寄存器的值大于PWM输入信号的高电平宽度，则上升沿将延迟一整个高电平的时间，使得输出没有高电平。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。采用与上文相同的方式，通过 PWM1A 上升沿延迟来产生 PWM2 ，并通过PWM2A上升延迟来产生 PWM4 。

PWM0死区上升沿延迟 (PWM0DBRISE)

偏移量 0x06C
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:12	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11:0	RiseDelay	R/W	0	死区上升延迟 上升沿延迟的时钟数。

寄存器 **46: PWM0**死区下降沿延迟 （**PWM0DBFALL**），偏移量 **0x070**

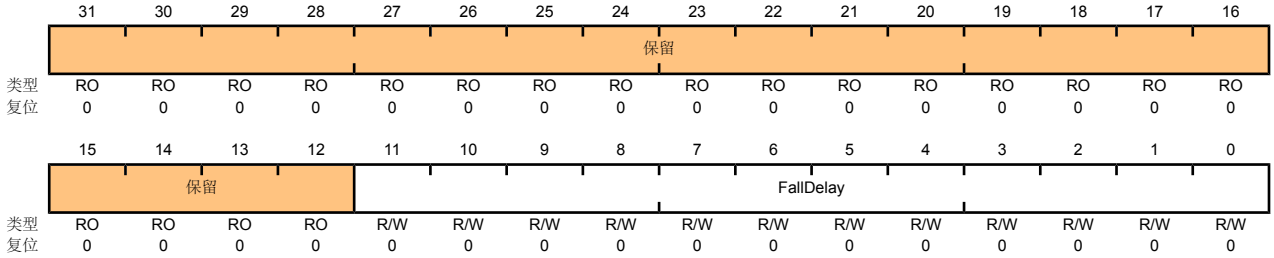
寄存器 **47: PWM1**死区下降沿延迟（**PWM1DBFALL**），偏移量 **0x0B0**

寄存器 **48: PWM2**死区下降沿延迟（**PWM2DBFALL**），偏移量 **0x0F0**

PWM0DBFALL寄存器包含的是在生成PWM1信号时，PWM0A信号下降沿延迟的时钟数。如果死区发生器被禁能，那么将忽略该寄存器。如果该寄存器的值大于PWM输入信号的低电平宽度，则下降沿将延迟一整个低电平的时间，导致输出没有低电平。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。采用与上文相同的方式，通过PWM1A下降沿延迟来产生PWM3，并通过PWM2A下降沿延迟来产生 PWM5。

PWM0死区下降沿延迟 (PWM0DBFALL)

偏移量 0x070
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:12	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
11:0	FallDelay	R/W	0x00	死区下降沿延迟 下降沿延迟的时钟数。

18 正交编码接口 (QEI)

正交编码器（又名双通道增量式编码器），用于将线性位移转换成脉冲信号。通过监控脉冲的数目和两个信号的相对相位，用户可以跟踪旋转的位置、方向和速度。此外还有第三个通道，称为索引信号，可用来对位置计数器进行复位，以确定绝对位置。

Stellaris® 正交编码器接口 (QEI) 模块 对正交编码器轮产生的代码进行解码，将它们解释成位置对时间的积分，并确定旋转的方向。另外，它还能够捕获编码器轮运转时的大致速度。

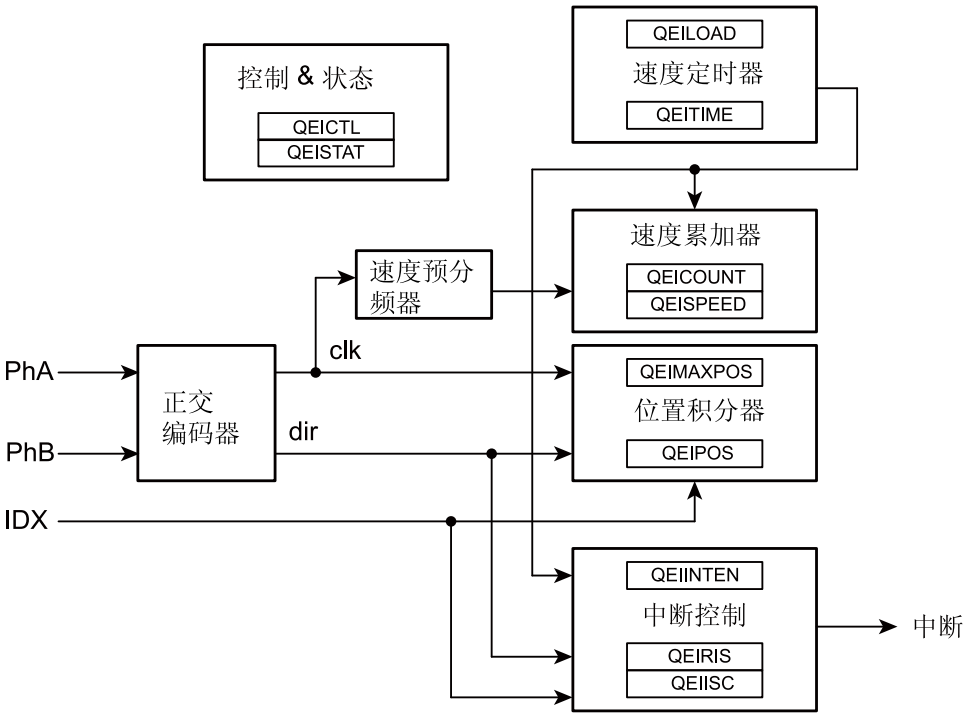
Stellaris® 正交编码器具有以下特性：

- 使用位置积分器来跟踪编码器的位置
- 使用内置定时器来捕获速度
- QEI在下列事件发生时将产生中断：
  - 检测到索引脉冲
  - 速度定时器发生计满返回事件
  - 旋转方向发生改变
  - 检测到正交错误

18.1 结构图

图 18-1 在 435页显示了Stellaris® QEI模块的结构图。

图 **18-1. QEI**结构图



18.2 功能描述

QEI模块对正交编码器轮产生的两位格雷码（gray code）进行解码，将它们解释成位置对时间的积分并确定旋转的方向。另外，它还能够捕获编码器轮运转时的大致速度。

虽然必须在使能速度捕获前使能位置积分器，但仍然可以单独使能位置积分器和速度捕获。PhA 和PhB这两个相位信号在被QEI模块解码前可以进行交换，以改变正向和反向的意义和纠正系统的错误接线（miswiring）。另外，相位信号也可以解释为时钟和方向信号，将它们作为某些编码器的输出。

QEI模块支持两种信号操作模式：正交相位模式和时钟/方向模式。在正交相位模式中，编码器产生两个相位差为90度的时钟信号；它们的边沿关系被用来确定旋转方向。在时钟/方向模式中，编码器产生一个时钟信号和一个方向信号，分别表示步长和旋转方向。这两种模式的选择由**QEI**控制**(QEICTL)**寄存器 (见 439页)中的SigMode位确定。

在将QEI模块设置为使用正交相位模式（SigMode位为0）时，位置积分器的捕获模式可设置成在PhA信号的上升和下降沿或是在PhA和PhB的上升和下降沿对位置计数器进行更新。在PhA和PhB的上升和下降沿上更新位置计数器提供更高精度的数据（更多位置计数），但位置计数器的计数范围却相对变少了。

当PhA的边沿超前于PhB的边沿时，位置计数器加1。当PhA的边沿滞后于PhB的边沿时，位置计数器减1。当一对上升沿和下降沿出现在其中一个相位上，而在其它相位上没有任何边沿时，这表示旋转方向已经发生了改变。

位置计数器遇到下列其中一种情况时将自动复位：1、检测到索引脉冲；2、位置计数器的值达到最大值。复位模式由**QEI**控制 **(QEICTL)**寄存器的ResMode位确定。

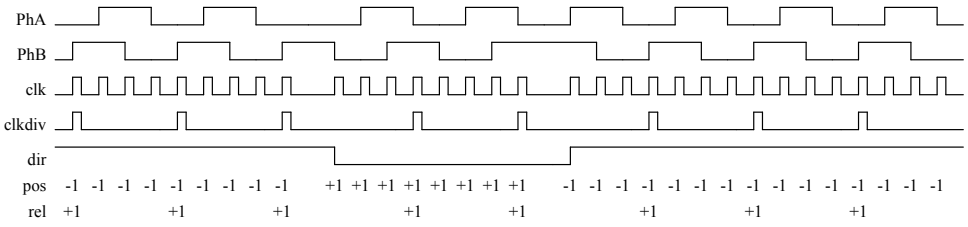
当ResMode位为0时，位置计数器在检测到索引脉冲时复位。在该模式下，位置计数器的值限制在[0:N-1]内，N为编码器轮旋转一圈得到的相位边沿数。**QEIMAXPOS** 寄存器必须设置为N-1，这样，从位置0反向就可以使位置计数器移到N-1。在该模式中，一旦出现索引脉冲，位置寄存器就包含了编码器相对于索引（或发起）位置的绝对位置的信息。

当ResMode 位为1时，位置计数器的范围限制在[0:M]内，M为可编程的最大值。在该模式中，位置计数器将忽略索引脉冲。

速度捕获包含一个可配置的定时器和一个计数寄存器。定时器在给定时间周期内对相位边沿进行计数（使用与位置积分器相同的配置）。控制器通过**QEISPEED**寄存器来获得上一个时间周期内的边沿计数值，而当前时间周期的边沿计数在**QEICOUNT**寄存器中进行累加。当前时间周期一结束，在该段时间内计得的边沿总数便可以从**QEISPEED**寄存器中获得（上一个值丢失）。这时**QEICOUNT**复位为0，并在一个新的时间周期开始计数。在给定时间周期内所计得的边沿数目与编码器的速度成正比例。

图 18-2 在 436页显示了Stellaris®正交编码器如何将相位输入信号转换为时钟脉冲、方向信号，以及速度预分频器如何操作（在4分频模式中）。

图 **18-2.** 正交编码器和速度预分频器的操作



定时器的周期由**QEILOAD**寄存器中指定的定时器装载值来确定。定时器到达0时可触发一次中断，硬件将**QEILOAD**的值重新装载到定时器中，并继续递减计数。在编码器的速度较低的情况下，需要

一个较长的定时器周期，以便捕获足够多的边沿，从而使得结果有意义。在编码器速度较高的情况下，可以使用较短的定时器周期也可以使用速度预分频器。

可以使用下面的等式将速率计数器的值转换为rpm（每分钟的转数）：

```
rpm = (clock * (2 ^ VelDiv) * Speed * 60) ÷ (Load * ppr * edges)
```

此处：

clock 表示控制器的时钟速率

ppr 表示实际编码器旋转一圈的脉冲数

edge 为2或4，根据**QEICTL**寄存器中设置的捕获模式来决定（CapMode设为0时，edge为2，CapMode为1时edge为4）。

例如，有一个运行速率为600rpm的电机。在电机上连接一个每转可产生2048个脉冲的正交编码器，这样，每转可获得8192个相位边沿。当相位预分频器设置为1分频（即VelDiv设置为0）并在PhA和PhB边沿上计时时，每秒可获得81920个脉冲（电机每秒转动10次）。如果定时器的时钟频率为10,000Hz，装载值为2,500（可定时1/4s），则每次更新定时器时，可计得20,480个脉冲。使用上述等式：

```
rpm = (10000 * 1 * 20480 * 60) ÷ (2500 * 2048 * 4) = 600 rpm
```

现在，假设电机速率增加到3000rpm。这时正交编码器每秒产生409,600个脉冲，即每1/4s可产生102,400个脉冲。再次使用上述等式：

```
rpm = (10000 * 1 * 102400 * 60) ÷ (2500 * 2048 * 4) = 3000 rpm
```

由于某些立即数可能会超过32位整数，因此，在计算这个等式时要特别注意。在上例中，时钟为10,000，除法器为2,500，我们可以将这两个值预先除以100（如果它们在编译时是常数），因此这两个值就变为100和25。事实上，如果它们在编译时是常数，则可将它们简化为只简单地乘以4，而又由于边沿计数因子为4，因此两个值刚好抵消。

重要: 简化编译时的常量因子和简化计算该等式时的处理请求同是控制该等式的立即数的最好方法。

通过选择定时器装载值使得该式子的除数为2的n次幂，这样便可避免除法操作；而只需一个简单的移位操作。在编码器每转产生的脉冲数为2的n次幂时，我们很容易会想到选择2的n次幂作为装载值。而对于其它编码器，必须选择合适的装载值，使得load、ppr、edges的乘积非常接近2的n次幂。例如，每转100个脉冲的编码器，其装载值可设为82，这样，除数便为32,800，该值比2^{14}大0.09%；在此情况下，通常15次移位就已足够接近于除法操作的结果。如果要求绝对精度，则可以使用控制器的除法指令。

QEI模块能够在出现以下事件时产生控制器中断：相位错误、方向改变、接收到索引脉冲、速度定时器发生计满返回事件。该模块还提供标准屏蔽、原始中断状态、中断状态，以及中断清零功能。

18.3 初始化和配置

下面的例子说明了如何配置正交编码器模块来读回绝对位置：

1. 通过写0x0000.0100到系统控制模块中的**RCGC1**寄存器来使能QEI时钟。
2. 通过系统控制模块中的**RCGC2**寄存器来使能对应GPIO模块的时钟。
3. 在GPIO模块中，使用**GPIOAFSEL**寄存器来使能对应管脚的第二功能。

4. 将正交编码器配置为捕获两个信号的边沿，并在索引脉冲复位时保存绝对位置的信息。使用1000-线编码器，每条线有4个边沿，因此每转一圈产生4000个脉冲；位置计数器从0开始计数，所以将最大位置计数值设置为3999（0xF9F）。
   - 向**QEICTL**寄存器写入0x0000.0018。
   - 向**QEIMAXPOS**寄存器写入0x0000.0F9F。
5. 将QEICTL寄存器的位0置位以使能正交编码器。
6. 延迟一段时间。
7. 读取**QEIPOS**寄存器以获取编码器的位置信息。

18.4 寄存器映射

表 18-1 在 438页列出了QEI寄存器。所有列出的地址都是相对于QEI模块基址的16进制变量：

- QEI0: 0x4002.C000

表 **18-1. QEI** 寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	QEICTL	R/W	0x0000.0000	QEI控制	439
0x004	QEISTAT	RO	0x0000.0000	QEI状态	441
0x008	QEIPOS	R/W	0x0000.0000	QEI位置	442
0x00C	QEIMAXPOS	R/W	0x0000.0000	QEI最大位置值	443
0x010	QEILOAD	R/W	0x0000.0000	QEI定时器装载	444
0x014	QEITIME	RO	0x0000.0000	QEI定时器	445
0x018	QEICOUNT	RO	0x0000.0000	QEI速度计数器	446
0x01C	QEISPEED	RO	0x0000.0000	QEI速度	447
0x020	QEIINTEN	R/W	0x0000.0000	QEI中断使能	448
0x024	QEIRIS	RO	0x0000.0000	QEI原始中断状态	449
0x028	QEIISC	R/W1C	0x0000.0000	QEI中断状态和清零	450

18.5 寄存器描述

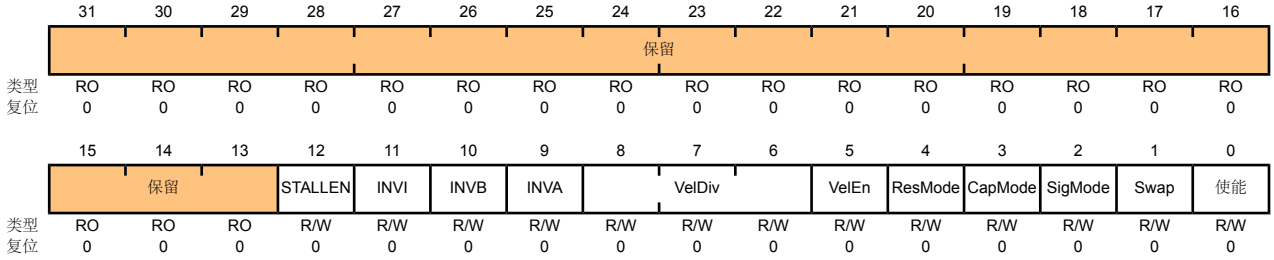
下面将按地址偏移量的数字顺序列出QEI的寄存器并对它们进行了描述。

寄存器 1: QEI控制（QEICTL），偏移量 0x000

该寄存器含有QEI模块的配置信息。正交编码器和速度捕获模块可单独使能；在捕获速度时，必须使能正交编码器，但在不需要知道速度的应用中可以禁止捕获速率功能。另外，相位信号的解码、相位交换、位置更新模式、位置复位模式和速率预分频器也都通过该寄存器进行设置。

QEI控制 (QEICTL)

偏移量 0x000
类型 R/W, 复位 0x0000.0000



位/域	名称	类型	复位	描述
31:13	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
12	STALLEN	R/W	0	停止 QEI 如果该位置位，则QEI在微控制器发出Halt信号时停止。
11	INVI	R/W	0	索引脉冲反相 该位置位时，将输入索引脉冲反相。
10	INVB	R/W	0	PhB反相 该位置位时，将PhB输入反相。
9	INVA	R/W	0	PhA反相 该位置位时，将PhA输入反相。
8:6	VelDiv	R/W	0x0	预分频速度 输入正交脉冲在传输到QEICOUNT累加器之前先进行预分频。该位域可设置为以下值: 值 预分频 0x0 ÷1 0x1 ÷2 0x2 ÷4 0x3 ÷8 0x4 ÷16 0x5 ÷32 0x6 ÷64 0x7 ÷128
5	VelEn	R/W	0	捕获速度 该位置位时，正交编码器的速度捕获功能使能。

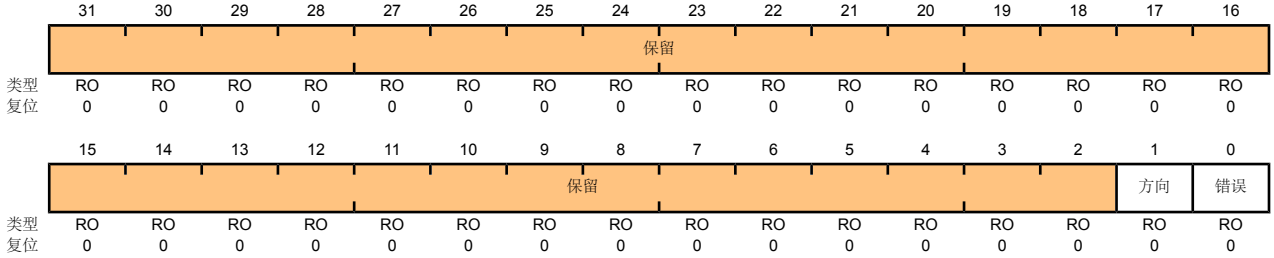
位/域	名称	类型	复位	描述
4	ResMode	R/W	0	复位模式 位置计数器的复位模式。该位为0时，位置计数器在到达最大值时复位；为1时，位置计数器在捕获到索引脉冲时复位。
3	CapMode	R/W	0	捕获模式 捕获模式定义在该位置进行计数的相位边沿。该位为0时，只对PhA的边沿进行计数；为1时，同时对PhA和PhB边沿进行计数，这样位置分辨率加倍，但计数范围却也减少一半。
2	SigMode	R/W	0	信号模式 该位为1时，信号PhA和PhB为时钟和方向信号；为0时，为正交相位信号。
1	Swap	R/W	0	交换信号 将信号PhA和PhB进行交换。
0	使能	R/W	0	使能QEI 使能正交编码器模块。

寄存器 2: QEI状态（QEISTAT），偏移量 0x004

该寄存器提供QEI模块操作相关的状态。

QEI状态 (QEISTAT)

偏移量 0x004
类型 RO, 复位 0x0000.0000



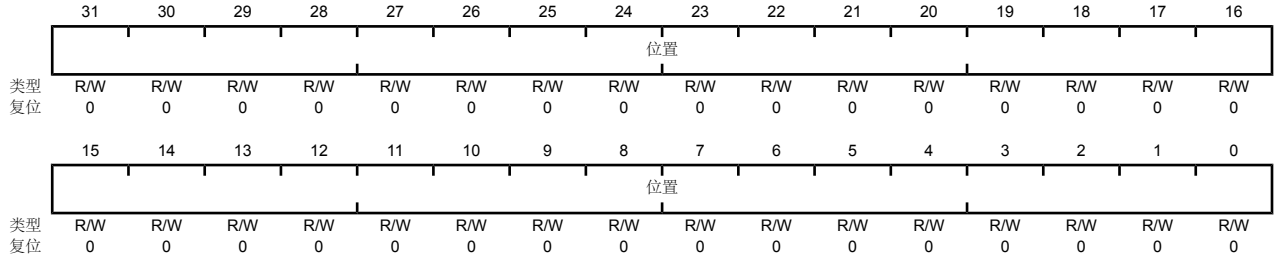
位/域	名称	类型	复位	描述
31:2	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	方向	RO	0	旋转方向 表示编码器旋转的方向。 该 方向值如下定义： 值 描述 0 正向旋转 1 反向旋转
0	错误	RO	0	错误检测 表示在格雷码序列中检测到错误（即，两个信号同时改变）。

寄存器 **3: QEI**位置（**QEIPOS**），偏移量 **0x008**

该寄存器包含位置积分器的当前值。寄存器的值在QEI相位输入端输入脉冲时更新，也可以通过写操作将其设置为一个指定值。

QEI位置 (QEIPOS)

偏移量 0x008
类型 R/W, 复位 0x0000.0000



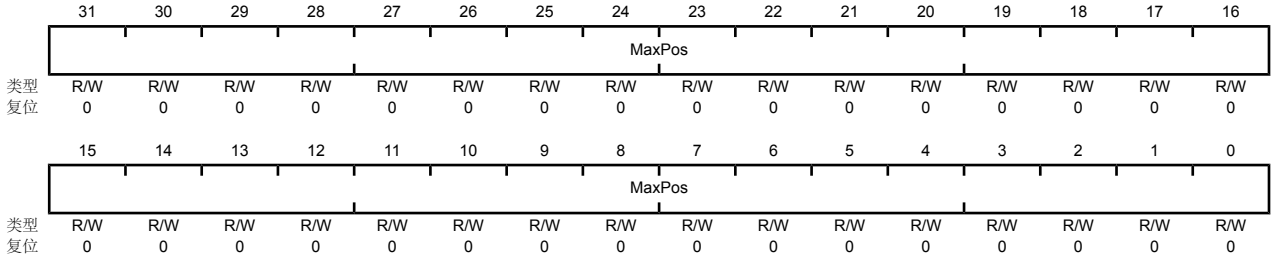
位/域	名称	类型	复位	描述
31:0	位置	R/W	0x00	当前的位置积分值 位置积分器的当前值。

寄存器 4: QEI最大位置值（QEIMAXPOS），偏移量 0x00C

该寄存器包含位置积分器的最大值。当正向旋转时，如果位置寄存器的值加1后超过该寄存器的值，则位置寄存器复位为0。当反向旋转时，如果执行减1操作的位置寄存器，其值已到达0，则位置寄存器复位为该寄存器中的值。

QEI最大位置值 (QEIMAXPOS)

偏移量 0x00C
类型 R/W, 复位 0x0000.0000



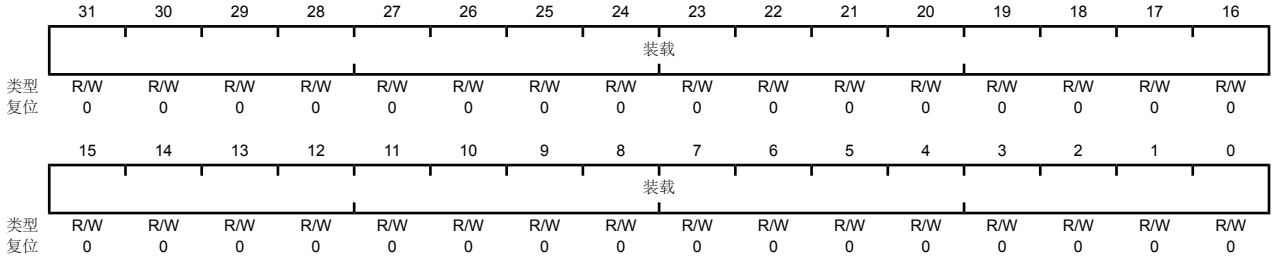
位/域	名称	类型	复位	描述
31:0	MaxPos	R/W	0x00	最大的位置积分值 位置积分器的最大值。

寄存器 **5: QEI**定时器装载（**QEILOAD**），偏移量 **0x010**

该寄存器包含速度定时器的装载值。由于该值在定时器到达0之后的时钟周期内装入定时器，因此，它应该比目标周期内的时钟数小1。即，如果希望每个定时周期内有2000个时钟，则该寄存器中的值应为1999。

QEI定时器装载 (QEILOAD)

偏移量 0x010
类型 R/W, 复位 0x0000.0000



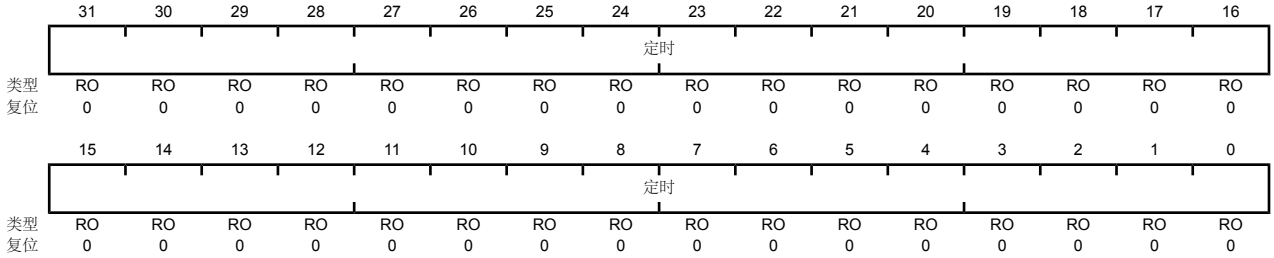
位/域	名称	类型	复位	描述
31:0	装载	R/W	0x00	速度定时器的装载值 速度定时器的装载值。

寄存器 6: QEI定时器（QEITIME），偏移量 0x014

该寄存器包含速度定时器的当前值。当**QEICTL**的VelEn位为0时，这个计数器不执行加1操作。

QEI定时器 (QEITIME)

偏移量 0x014
类型 RO, 复位 0x0000.0000



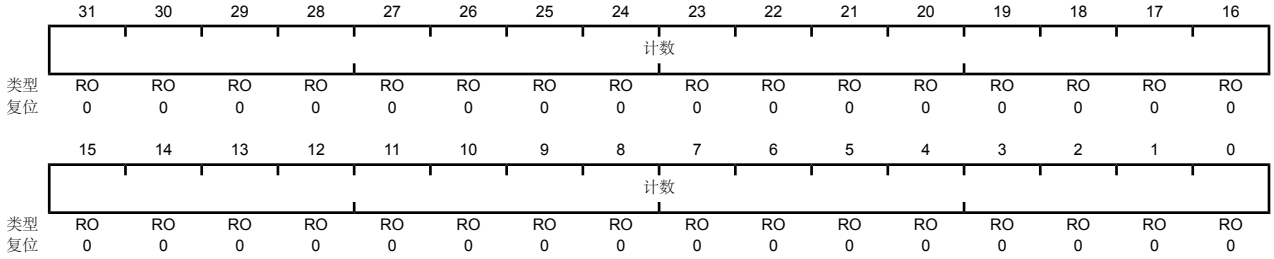
位/域	名称	类型	复位	描述
31:0	定时	RO	0x00	速度定时器的当前值 速度定时器的当前值。

寄存器 **7: QEI**速度计数器（**QEICOUNT**），偏移量 **0x018**

该寄存器包含在当前时间周期内，正在计数的速率脉冲的个数。这是正在运行时的脉冲总数，因此，它所对应的时间周期的精度不可知（即该寄存器的读操作未必与**QEITIME**寄存器返回的时间对应，因为两次读操作之间的时间间隔（window of time）非常小，在这段时间内任一个值都可能发生改变）。**QEISPEED** 寄存器应该用来确定编码器的实际速度；它只用来存储信息。当**QEICTL**的VelEn为0时，计数器不执行加1操作。

QEI速度计数器 (QEICOUNT)

偏移量 0x018
类型 RO, 复位 0x0000.0000



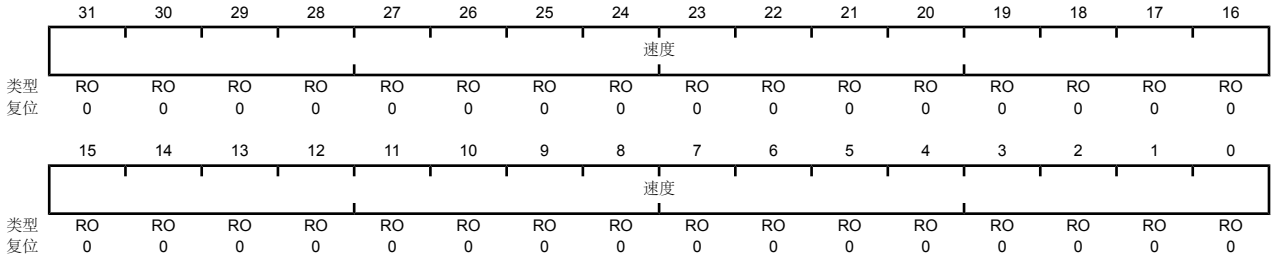
位/域	名称	类型	复位	描述
31:0	计数	RO	0x00	速度脉冲计数 在速度定时器周期内，编码器正在运行时的脉冲总数。

寄存器 8: QEI速度（QEISPEED），偏移量 0x01C

该寄存器包含最近测得的正交编码器的速度。它与上一个定时周期内计得的脉冲数相对应。当**QEICTL**的 VelEn为0时，该寄存器不会更新。

QEI速度 (QEISPEED)

偏移量 0x01C
类型 RO, 复位 0x0000.0000



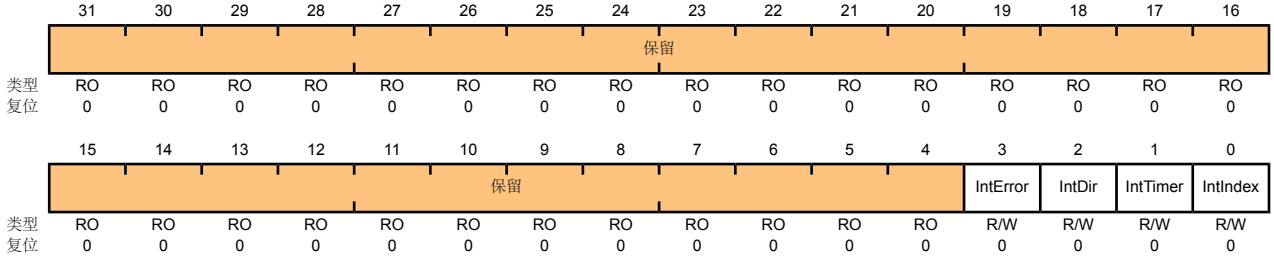
位/域	名称	类型	复位	描述
31:0	速度	RO	0x00	速度 测得的正交编码器的速度，以一个周期时间内的脉冲数来计量。

寄存器 **9: QEI**中断使能（**QEIINTEN**），偏移量 **0x020**

该寄存器用于使能QEI模块的各个中断。如果将该寄存器中的对应位设为1，则向控制器发出中断。

QEI中断使能 (QEIINTEN)

偏移量 0x020
类型 R/W, 复位 0x0000.0000



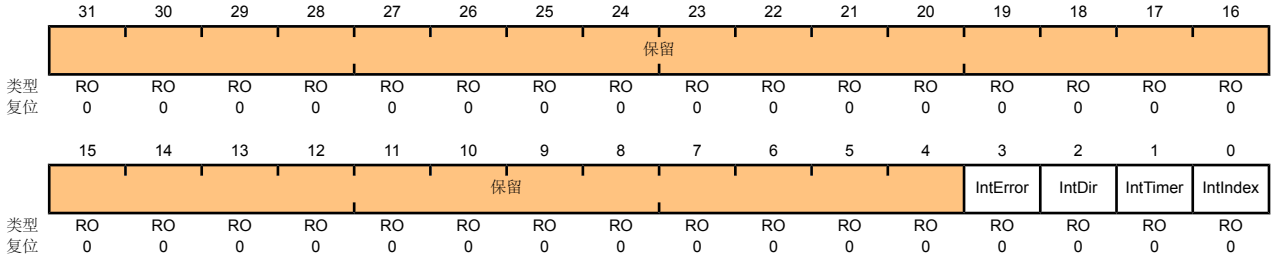
位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	IntError	R/W	0	相位错误中断使能位 如果该位为1，则在检测到相位错误时产生中断。
2	IntDir	R/W	0	方向改变中断使能位 如果该位为1，则在改变方向时产生中断。
1	IntTimer	R/W	0	定时器计满返回中断使能位 如果该位为1，则在速度定时器发生计满返回事件时产生中断。
0	IntIndex	R/W	0	索引脉冲检测中断使能位 如果该位为1，则在检测到索引脉冲时产生中断。

寄存器 10: QEI原始中断状态（QEIRIS），偏移量 0x024

该寄存器提供已发出中断的中断源的当前设置，而不管它们是否将中断提交到了控制器（通过**QEIINTEN**寄存器进行设置）。寄存器中的位为1表示已发生锁存事件，为0表示事件没有发生。

QEI原始中断状态 (QEIRIS)

偏移量 0x024
类型 RO, 复位 0x0000.0000



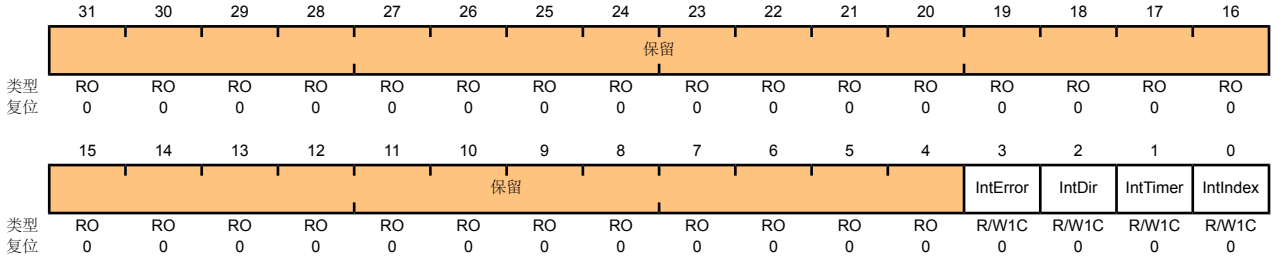
位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	IntError	RO	0	相位错误检测 表示检测到相位错误。
2	IntDir	RO	0	方向改变检测 表示方向已发生改变。
1	IntTimer	RO	0	速度定时器计满返回位 表示速度定时器已经计满准备返回。
0	IntIndex	RO	0	索引脉冲出现位 表示已出现索引脉冲。

寄存器 **11: QEI**中断状态和清零（**QEIISC**），偏移量 **0x028**

该寄存器提供已发出给控制器的中断源的当前设置。寄存器中的位为1表示已发生锁存事件，为0表示事件没有发生。该寄存器是R/W1C类型，即向某个位写1可将对应的中断原因清零。

QEI中断状态和清零 (QEIISC)

偏移量 0x028
类型 R/W1C, 复位 0x0000.0000

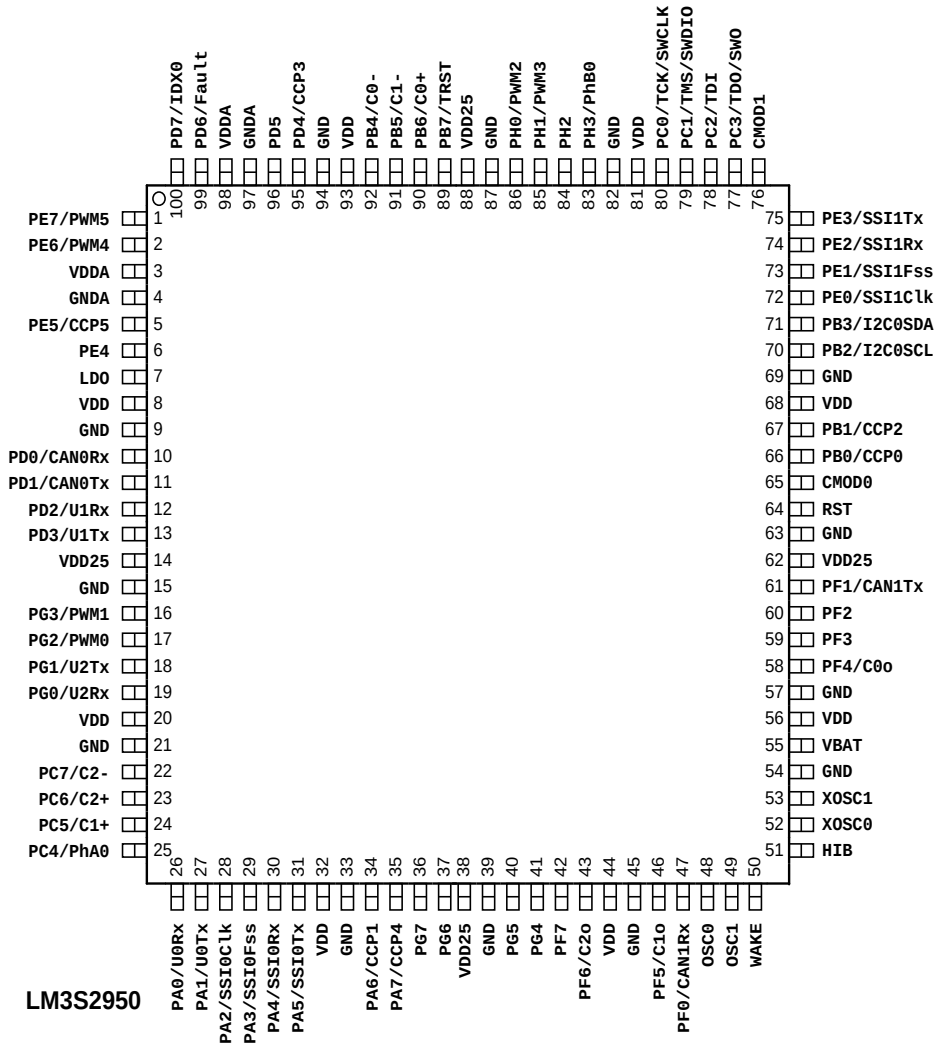


位/域	名称	类型	复位	描述
31:4	保留	RO	0x00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	IntError	R/W1C	0	相位错误中断 表示检测到相位错误。
2	IntDir	R/W1C	0	方向改变中断 表示方向已发生改变。
1	IntTimer	R/W1C	0	速度定时器计满返回中断 表示速度定时器已经计满准备返回。
0	IntIndex	R/W1C	0	索引脉冲中断 表示已出现索引脉冲。

19 管脚图

图 19-1 在 451页显示管脚图和管脚到信号名称的映射。

图 **19-1.** 管脚连接图



20 信号表

下表列出了每个管脚可用的信号。通过**GPIOAFSEL**寄存器，由软件来使能管脚功能。

重要: 默认情况下所有多路复用管脚均为GPIO，5个JTAG管脚(PB7和PC[3:0])除外，它们默认用作JTAG功能。

表 20-1 在 452页显示管脚到信号名称的映射，包括信号的功能特性。表 20-2 在 456页按信号名称的字母顺序列出信号。

表 20-3 在 460页按功能将信号分组，GPIO管脚除外。表 20-4 在 464页列出了GPIO管脚和它们可选的功能。

表 **20-1.** 按管脚编号排列的信号

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
1	PE7	I/O	TTL	GPIO端口E位7
	PWM5	O	TTL	PWM 5
2	PE6	I/O	TTL	GPIO端口E位6
	PWM4	O	TTL	PWM 4
3	VDDA	-	电源	模拟电路（ADC、模拟比较器等）的正电源（3.3V）。这些电源与VDD独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
4	GNDA	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
5	PE5	I/O	TTL	GPIO端口E位5
	CCP5	I/O	TTL	捕获/比较/PWM 5
6	PE4	I/O	TTL	GPIO端口E位4
7	LDO	-	电源	低压差稳压器输出电压。这个管脚在管脚和GND之间需要一个1µF或更大的外部电容。当使用片内LDO给逻辑电路提供电源时，除了去耦电容（decoupling capacitor）外，LDO管脚还必须连接到板极的VDD25管脚。
8	VDD	-	电源	I/O和某些逻辑的电源正极。
9	GND	-	电源	逻辑和I/O管脚的地参考。
10	PD0	I/O	TTL	GPIO端口D位0
	CAN0Rx	I	TTL	CAN模块0接收
11	PD1	I/O	TTL	GPIO端口D位1
	CAN0Tx	O	TTL	CAN模块0发送
12	PD2	I/O	TTL	GPIO端口D位2
	U1Rx	I	TTL	UART模块1接收。当在IrDA模式下时，该信号具有IrDA调制。
13	PD3	I/O	TTL	GPIO端口D位3
	U1Tx	O	TTL	UART模块1发送。当在IrDA模式时，该信号具有IrDA调制。
14	VDD25	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
15	GND	-	电源	逻辑和I/O管脚的地参考。

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
16	PG3	I/O	TTL	GPIO端口G位3
	PWM1	O	TTL	PWM 1
17	PG2	I/O	TTL	GPIO端口G位2
	PWM0	O	TTL	PWM 0
18	PG1	I/O	TTL	GPIO端口G位1
	U2Tx	O	TTL	UART 2发送。当在IrDA模式下时，该信号具有IrDA调制。
19	PG0	I/O	TTL	GPIO端口G位0
	U2Rx	I	TTL	UART 2接收。当在IrDA模式下时，该信号具有IrDA调制。
20	VDD	-	电源	I/O和某些逻辑的电源正极。
21	GND	-	电源	逻辑和I/O管脚的地参考。
22	PC7	I/O	TTL	GPIO端口C位7
	C2-	I	模拟	模拟比较器2负极输入
23	PC6	I/O	TTL	GPIO端口C位6
	C2+	I	模拟	模拟比较器正极输入
24	PC5	I/O	TTL	GPIO端口C位5
	C1+	I	模拟	模拟比较器正极输入
25	PC4	I/O	TTL	GPIO端口C位4
	PhA0	I	TTL	QEI模块0相位A
26	PA0	I/O	TTL	GPIO端口A位0
	U0Rx	I	TTL	UART模块0接收。当在IrDA模式时，该信号具有IrDA调制。
27	PA1	I/O	TTL	GPIO端口A位1
	U0Tx	O	TTL	UART模块0发送。当在IrDA模式时，该信号具有IrDA调制。
28	PA2	I/O	TTL	GPIO端口A位2
	SSI0Clk	I/O	TTL	SSI模块0时钟
29	PA3	I/O	TTL	GPIO端口A位3
	SSI0Fss	I/O	TTL	SSI模块0帧
30	PA4	I/O	TTL	GPIO端口A位4
	SSI0Rx	I	TTL	SSI模块0接收
31	PA5	I/O	TTL	GPIO端口A位5
	SSI0Tx	O	TTL	SSI模块0发送
32	VDD	-	电源	I/O和某些逻辑的正电源。
33	GND	-	电源	逻辑和I/O管脚的地参考。
34	PA6	I/O	TTL	GPIO端口A位6
	CCP1	I/O	TTL	捕获/比较/PWM 1
35	PA7	I/O	TTL	GPIO端口A位7
	CCP4	I/O	TTL	捕获/比较/PWM 1
36	PG7	I/O	TTL	GPIO端口G位7
37	PG6	I/O	TTL	GPIO端口G位6
38	VDD25	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
39	GND	-	电源	逻辑和I/O管脚的地参考。
40	PG5	I/O	TTL	GPIO端口G位5
41	PG4	I/O	TTL	GPIO端口G位4
42	PF7	I/O	TTL	GPIO端口F位7
43	PF6	I/O	TTL	GPIO端口F位6
	C2o	O	TTL	模拟比较器2输出
44	VDD	-	电源	I/O和某些逻辑的正极电源。
45	GND	-	电源	逻辑和I/O管脚的地参考。
46	PF5	I/O	TTL	GPIO端口F位5
	C1o	O	TTL	模拟比较器1输出
47	PF0	I/O	TTL	GPIO端口F位0
	CAN1Rx	I	TTL	CAN模块1接收
48	OSC0	I	模拟	主振荡器晶体输入或外部时钟参考输入。
49	OSC1	O	模拟	主振荡器晶体输出。
50	$\overline{\text{WAKE}}$	I	OD	当有效时外部输入将处理器从休眠模式中唤醒。
51	$\overline{\text{HIB}}$	O	TTL	该输出指示处理器在休眠模式下。
52	XOSC0	I	模拟	休眠模块振荡器晶体输入或外部时钟参考输入。注意这是用于休眠模块RTC的4.19-MHz晶体或32.768-kHz振荡器。见**HIBCTL**寄存器的CLKSEL位。
53	XOSC1	O	模拟	休眠模块振荡器晶体输出。
54	GND	-	电源	逻辑和I/O管脚的地参考。
55	VBAT	-	电源	休眠模块的电源供应源。它通常连接到电池的正极端并用作备用电池/休眠模块电源供应器的电源。
56	VDD	-	电源	I/O和某些逻辑的正极电源。
57	GND	-	电源	逻辑和I/O管脚的地参考。
58	PF4	I/O	TTL	GPIO端口F位4
	C0o	O	TTL	模拟比较器0输出
59	PF3	I/O	TTL	GPIO端口F位3
60	PF2	I/O	TTL	GPIO端口F位2
61	PF1	I/O	TTL	GPIO端口F位1
	CAN1Tx	O	TTL	CAN模块1发送
62	VDD25	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
63	GND	-	电源	逻辑和I/O管脚的地参考。
64	$\overline{\text{RST}}$	I	TTL	系统复位输入
65	CMOD0	I/O	TTL	CPU模式位0。输入必须设为逻辑0（地）；其它编码保留。
66	PB0	I/O	TTL	GPIO端口B位0
	CCP0	I/O	TTL	捕获/比较/PWM 0
67	PB1	I/O	TTL	GPIO端口B位1
	CCP2	I/O	TTL	捕获/比较/PWM 2
68	VDD	-	Power	I/O和某些逻辑的正极电源。

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
69	GND	-	电源	逻辑和I/O管脚的地参考。
70	PB2	I/O	TTL	GPIO端口B位2
	I2C0SCL	I/O	OD	I2C模块0时钟
71	PB3	I/O	TTL	GPIO端口B位3
	I2C0SDA	I/O	OD	I2C模块0数据
72	PE0	I/O	TTL	GPIO端口E位0
	SSI1Clk	I/O	TTL	SSI模块1时钟
73	PE1	I/O	TTL	GPIO端口E位1
	SSI1Fss	I/O	TTL	SSI模块1帧
74	PE2	I/O	TTL	GPIO端口E位2
	SSI1Rx	I	TTL	SSI模块1接收
75	PE3	I/O	TTL	GPIO端口E位3
	SSI1Tx	O	TTL	SSI模块1发送
76	CMOD1	I/O	TTL	CPU模式位1。输入必须设为逻辑0（地）；其它编码保留。
77	PC3	I/O	TTL	GPIO端口C位3
	TDO	O	TTL	JTAG TDO和SWO
	SWO	O	TTL	JTAG TDO和SWO
78	PC2	I/O	TTL	GPIO端口C位2
	TDI	I	TTL	JTAG TDI
79	PC1	I/O	TTL	GPIO端口C位1
	TMS	I/O	TTL	JTAG TMS和SWDIO
	SWDIO	I/O	TTL	JTAG TMS和SWDIO
80	PC0	I/O	TTL	GPIO端口C位0
	TCK	I	TTL	JTAG/SWD CLK
	SWCLK	I	TTL	JTAG/SWD CLK
81	VDD	-	电源	I/O和某些逻辑的正极电源。
82	GND	-	电源	逻辑和I/O管脚的地参考。
83	PH3	I/O	TTL	GPIO端口H位3
	PhB0	I	TTL	QEI模块0相位B
84	PH2	I/O	TTL	GPIO端口H位2
85	PH1	I/O	TTL	GPIO端口H位1
	PWM3	O	TTL	PWM 3
86	PH0	I/O	TTL	GPIO端口H位0
	PWM2	O	TTL	PWM 2
87	GND	-	电源	逻辑和I/O管脚的地参考。
88	VDD25	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
89	PB7	I/O	TTL	GPIO端口B位7
	TRST	I	TTL	JTAG TRSTn
90	PB6	I/O	TTL	GPIO端口B位6
	C0+	I	模拟	模拟比较器0正极输入

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
91	PB5	I/O	TTL	GPIO端口B位5
	C1-	I	模拟	模拟比较器1负极输入
92	PB4	I/O	TTL	GPIO端口B位4
	C0-	I	模拟	模拟比较器0负极输入
93	VDD	-	电源	I/O和某些逻辑的正极电源。
94	GND	-	电源	逻辑和I/O管脚的地参考。
95	PD4	I/O	TTL	GPIO端口D位4
	CCP3	I/O	TTL	捕获/比较/PWM 3
96	PD5	I/O	TTL	GPIO端口D位5
97	GNDA	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度减少VDD上的电气噪声，使其不影响模拟功能。
98	VDDA	-	电源	模拟电路（ADC、模拟比较器等）的电源正极(3.3V)。这些电源与VDD独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
99	PD6	I/O	TTL	GPIO端口D位6
	Fault	I	TTL	PWM错误
100	PD7	I/O	TTL	GPIO端口D位7
	IDX0	I	TTL	QEI模块0索引

表 **20-2.** 按信号名称排列的信号

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
C0+	90	I	模拟	模拟比较器0正极输入
C0-	92	I	模拟	模拟比较器0负极输入
C0o	58	O	TTL	模拟比较器0输出
C1+	24	I	模拟	模拟比较器正极输入
C1-	91	I	模拟	模拟比较器1负极输入
C1o	46	O	TTL	模拟比较器1输出
C2+	23	I	模拟	模拟比较器负极输入
C2-	22	I	模拟	模拟比较器2负极输入
C2o	43	O	TTL	模拟比较器2输出
CAN0Rx	10	I	TTL	CAN模块0接收
CAN0Tx	11	O	TTL	CAN模块0发送
CAN1Rx	47	I	TTL	CAN模块1接收
CAN1Tx	61	O	TTL	CAN模块1发送
CCP0	66	I/O	TTL	捕获/比较/PWM 0
CCP1	34	I/O	TTL	捕获/比较/PWM 1
CCP2	67	I/O	TTL	捕获/比较/PWM 2
CCP3	95	I/O	TTL	捕获/比较/PWM 3
CCP4	35	I/O	TTL	捕获/比较/PWM 1
CCP5	5	I/O	TTL	捕获/比较/PWM 5
CMOD0	65	I/O	TTL	CPU模式位0。输入必须设为逻辑0（地）；其它编码保留。

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
CMOD1	76	I/O	TTL	CPU模式位1。输入必须设为逻辑0（地）；其它编码保留。
Fault	99	I	TTL	PWM错误
GND	9	-	电源	逻辑和I/O管脚的地参考。
GND	15	-	电源	逻辑和I/O管脚的地参考。
GND	21	-	电源	逻辑和I/O管脚的地参考。
GND	33	-	电源	逻辑和I/O管脚的地参考。
GND	39	-	电源	逻辑和I/O管脚的地参考。
GND	45	-	电源	逻辑和I/O管脚的地参考。
GND	54	-	电源	逻辑和I/O管脚的地参考。
GND	57	-	电源	逻辑和I/O管脚的地参考。
GND	63	-	电源	逻辑和I/O管脚的地参考。
GND	69	-	电源	逻辑和I/O管脚的地参考。
GND	82	-	电源	逻辑和I/O管脚的地参考。
GND	87	-	电源	逻辑和I/O管脚的地参考。
GND	94	-	电源	逻辑和I/O管脚的地参考。
GNDA	4	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
GNDA	97	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
HIB	51	O	TTL	该输出指示处理器在睡眠模式下。
I2C0SCL	70	I/O	OD	I2C模块0时钟
I2C0SDA	71	I/O	OD	I2C module 0 data
IDX0	100	I	TTL	QEI模块0索引
LDO	7	-	电源	低压差稳压器输出电压。这个管脚在管脚和GND之间需要一个1µF或更大的外部电容。当使用片内LDO给逻辑电路提供电源时，除了去耦电容（decoupling capacitor）外，LDO管脚还必须连接到板极的VDD25管脚。
OSC0	48	I	模拟	主振荡器晶体输入或外部时钟参考输入。
OSC1	49	O	模拟	主振荡器晶体输出。
PA0	26	I/O	TTL	GPIO端口A位0
PA1	27	I/O	TTL	GPIO端口A位1
PA2	28	I/O	TTL	GPIO端口A位2
PA3	29	I/O	TTL	GPIO端口A位3
PA4	30	I/O	TTL	GPIO端口A位4
PA5	31	I/O	TTL	GPIO端口A位5
PA6	34	I/O	TTL	GPIO端口A位6
PA7	35	I/O	TTL	GPIO端口A位7
PB0	66	I/O	TTL	GPIO端口B位0
PB1	67	I/O	TTL	GPIO端口B位1
PB2	70	I/O	TTL	GPIO端口B位2
PB3	71	I/O	TTL	GPIO端口B位3

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
PB4	92	I/O	TTL	GPIO端口B位4
PB5	91	I/O	TTL	GPIO端口B位5
PB6	90	I/O	TTL	GPIO端口B位6
PB7	89	I/O	TTL	GPIO端口B位7
PC0	80	I/O	TTL	GPIO端口C位0
PC1	79	I/O	TTL	GPIO端口C位1
PC2	78	I/O	TTL	GPIO端口C位2
PC3	77	I/O	TTL	GPIO端口C位3
PC4	25	I/O	TTL	GPIO端口C位4
PC5	24	I/O	TTL	GPIO端口C位5
PC6	23	I/O	TTL	GPIO端口C位6
PC7	22	I/O	TTL	GPIO端口C位7
PD0	10	I/O	TTL	GPIO端口D位0
PD1	11	I/O	TTL	GPIO端口D位1
PD2	12	I/O	TTL	GPIO端口D位2
PD3	13	I/O	TTL	GPIO端口D位3
PD4	95	I/O	TTL	GPIO端口D位4
PD5	96	I/O	TTL	GPIO端口D位5
PD6	99	I/O	TTL	GPIO端口D位6
PD7	100	I/O	TTL	GPIO端口D位7
PE0	72	I/O	TTL	GPIO端口E位0
PE1	73	I/O	TTL	GPIO端口E位1
PE2	74	I/O	TTL	GPIO端口E位2
PE3	75	I/O	TTL	GPIO端口E位3
PE4	6	I/O	TTL	GPIO端口E位4
PE5	5	I/O	TTL	GPIO端口E位5
PE6	2	I/O	TTL	GPIO端口E位6
PE7	1	I/O	TTL	GPIO端口E位7
PF0	47	I/O	TTL	GPIO端口F位0
PF1	61	I/O	TTL	GPIO端口F位1
PF2	60	I/O	TTL	GPIO端口F位2
PF3	59	I/O	TTL	GPIO端口F位3
PF4	58	I/O	TTL	GPIO端口F位4
PF5	46	I/O	TTL	GPIO端口F位5
PF6	43	I/O	TTL	GPIO端口F位6
PF7	42	I/O	TTL	GPIO端口F位7
PG0	19	I/O	TTL	GPIO端口G位0
PG1	18	I/O	TTL	GPIO端口G位1
PG2	17	I/O	TTL	GPIO端口G位2
PG3	16	I/O	TTL	GPIO端口G位3
PG4	41	I/O	TTL	GPIO端口G位4
PG5	40	I/O	TTL	GPIO端口G位5

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
PG6	37	I/O	TTL	GPIO端口G位6
PG7	36	I/O	TTL	GPIO端口G位7
PH0	86	I/O	TTL	GPIO端口H位0
PH1	85	I/O	TTL	GPIO端口H位1
PH2	84	I/O	TTL	GPIO端口H位2
PH3	83	I/O	TTL	GPIO端口H位3
PhA0	25	I	TTL	QEI模块0相位A
PhB0	83	I	TTL	QEI模块0相位B
PWM0	17	O	TTL	PWM 0
PWM1	16	O	TTL	PWM 1
PWM2	86	O	TTL	PWM 2
PWM3	85	O	TTL	PWM 3
PWM4	2	O	TTL	PWM 4
PWM5	1	O	TTL	PWM 5
$\overline{\text{RST}}$	64	I	TTL	系统复位输入。
SSI0Clk	28	I/O	TTL	SSI模块0时钟
SSI0Fss	29	I/O	TTL	SSI模块0帧
SSI0Rx	30	I	TTL	SSI模块0接收
SSI0Tx	31	O	TTL	SSI模块0发送
SSI1Clk	72	I/O	TTL	SSI模块1时钟
SSI1Fss	73	I/O	TTL	SSI模块1帧
SSI1Rx	74	I	TTL	SSI模块1接收
SSI1Tx	75	O	TTL	SSI模块1发送
SWCLK	80	I	TTL	JTAG/SWD CLK
SWDIO	79	I/O	TTL	JTAG TMS和SWDIO
SWO	77	O	TTL	JTAG TDO和SWO
TCK	80	I	TTL	JTAG/SWD CLK
TDI	78	I	TTL	JTAG TDI
TDO	77	O	TTL	JTAG TDO和SWO
TMS	79	I/O	TTL	JTAG TMS和SWDIO
$\overline{\text{TRST}}$	89	I	TTL	JTAG TRSTn
U0Rx	26	I	TTL	UART模块0接收。当在IrDA模式下时，该信号具有IrDA调制。
U0Tx	27	O	TTL	UART模块0发送。当在IrDA模式下时，该信号具有IrDA调制。
U1Rx	12	I	TTL	UART模块1接收。当在IrDA模式下时，该信号具有IrDA调制。
U1Tx	13	O	TTL	UART模块1发送。当在IrDA模式下时，该信号具有IrDA调制。
U2Rx	19	I	TTL	UART 2 接收。当在IrDA模式下时，该信号具有IrDA调制。
U2Tx	18	O	TTL	UART 2 发送。当在IrDA模式下时，该信号具有IrDA调制。

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
VBAT	55	-	电源	休眠模块的电源供应源。它通常连接到电池的正极端并用作备用电池/休眠模块电源供应器的电源。
VDD	8	-	电源	I/O和某些逻辑的正极电源。
VDD	20	-	电源	I/O和某些逻辑的正极电源。
VDD	32	-	电源	I/O和某些逻辑的正极电源。
VDD	44	-	电源	I/O和某些逻辑的正极电源。
VDD	56	-	电源	I/O和某些逻辑的正极电源。
VDD	68	-	电源	I/O和某些逻辑的正极电源。
VDD	81	-	电源	I/O和某些逻辑的正极电源。
VDD	93	-	电源	I/O和某些逻辑的正极电源。
VDD25	14	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
VDD25	38	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
VDD25	62	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
VDD25	88	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
VDDA	3	-	电源	模拟电路（ADC、模拟比较器等）的电源正极(3.3 V)。 这些电源与VDD独立，以最大限度地减少 VDD上的电气噪声，使其不影响模拟功能。
VDDA	98	-	电源	模拟电路（ADC、模拟比较器等）的电源正极(3.3 V)。 这些电源与VDD独立，以最大限度地减少 VDD上的电气噪声，使其不影响模拟功能。
WAKE	50	I	OD	当有效时外部输入将处理器从休眠模式中唤醒。
XOSC0	52	I	模拟	睡眠模块振荡器晶体输入或外部时钟参考输入。注意这是用于睡眠模块RTC的4.19-MHz晶体或32.768-kHz振荡器。见**HIBCTL**寄存器的CLKSEL位。
XOSC1	53	O	模拟	休眠模块振荡器晶体输出。

表 **20-3.** 按功能划分的信号，**GPIO**除外

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
I2C	I2C0SCL	70	I/O	OD	I2C模块0时钟
	I2C0SDA	71	I/O	OD	I2C模块0数据
JTAG/SWD/SWO	SWCLK	80	I	TTL	JTAG/SWD CLK
	SWDIO	79	I/O	TTL	JTAG TMS和SWDIO
	SWO	77	O	TTL	JTAG TDO和SWO
	TCK	80	I	TTL	JTAG/SWD CLK
	TDI	78	I	TTL	JTAG TDI
	TDO	77	O	TTL	JTAG TDO和SWO
	TMS	79	I/O	TTL	JTAG TMS和SWDIO

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
PWM	PWM0	17	O	TTL	PWM 0
	PWM1	16	O	TTL	PWM 1
	PWM2	86	O	TTL	PWM 2
	PWM3	85	O	TTL	PWM 3
	PWM4	2	O	TTL	PWM 4
	PWM5	1	O	TTL	PWM 5
	错误	99	I	TTL	PWM错误
Power	GNDA	97	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
QEI	IDX0	100	I	TTL	QEI模块0索引
	PhA0	25	I	TTL	QEI模块0相位A
	PhB0	83	I	TTL	QEI模块0相位B
SSI	SSI0Clk	28	I/O	TTL	SSI模块0时钟
	SSI0Fss	29	I/O	TTL	SSI模块0帧
	SSI0Rx	30	I	TTL	SSI模块0接收
	SSI0Tx	31	O	TTL	SSI模块0发送
	SSI1Clk	72	I/O	TTL	SSI模块1时钟
	SSI1Fss	73	I/O	TTL	SSI模块1帧
	SSI1Rx	74	I	TTL	SSI模块1接收
	SSI1Tx	75	O	TTL	SSI模块1发送
UART	U0Rx	26	I	TTL	UART模块0接收。当在IrDA模式下时，该信号具有IrDA调制。
	U0Tx	27	O	TTL	UART模块0发送。当在IrDA模式下时，该信号具有IrDA调制。
	U1Rx	12	I	TTL	UART模块1接收。当在IrDA模式下时，该信号具有IrDA调制。
	U1Tx	13	O	TTL	UART模块1发送。当在IrDA模式下时，该信号具有IrDA调制。
	U2Rx	19	I	TTL	UART 2 接收。当在IrDA模式下时，该信号具有IrDA调制。
	U2Tx	18	O	TTL	UART 2 发送。当在IrDA模式下时，该信号具有IrDA调制。
控制器局域网	CAN0Rx	10	I	TTL	CAN模块0接收
	CAN0Tx	11	O	TTL	CAN模块0发送
	CAN1Rx	47	I	TTL	CAN模块1接收
	CAN1Tx	61	O	TTL	CAN模块1发送

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
模拟比较器	C0+	90	I	模拟	模拟比较器0正极输入
	C0-	92	I	模拟	模拟比较器0负极输入
	C0o	58	O	TTL	模拟比较器0输出
	C1+	24	I	模拟	模拟比较器正极输入
	C1-	91	I	模拟	模拟比较器1负极输入
	C1o	46	O	TTL	模拟比较器1输出
	C2+	23	I	模拟	模拟比较器正极输入
	C2-	22	I	模拟	模拟比较器2负极输入
	C2o	43	O	TTL	模拟比较器2输出

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
电源	GND	9	-	电源	逻辑和I/O管脚的地参考。
	GND	15	-	电源	逻辑和I/O管脚的地参考。
	GND	21	-	电源	逻辑和I/O管脚的地参考。
	GND	33	-	电源	逻辑和I/O管脚的地参考。
	GND	39	-	电源	逻辑和I/O管脚的地参考。
	GND	45	-	电源	逻辑和I/O管脚的地参考。
	GND	54	-	电源	逻辑和I/O管脚的地参考。
	GND	57	-	电源	逻辑和I/O管脚的地参考。
	GND	63	-	电源	逻辑和I/O管脚的地参考。
	GND	69	-	电源	逻辑和I/O管脚的地参考。
	GND	82	-	电源	逻辑和I/O管脚的地参考。
	GND	87	-	电源	逻辑和I/O管脚的地参考。
	GND	94	-	电源	逻辑和I/O管脚的地参考。
	GNDA	4	-	电源	模拟电路（ADC、模拟比较器等）的地参考。这些电源与GND独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
	HIB	51	O	TTL	该输出指示处理器在休眠模式下。
	LDO	7	-	电源	低压差稳压器输出电压。这个管脚在管脚和GND之间需要一个1µF或更大的外部电容。 当使用片内LDO给逻辑电路提供电源时，除了去耦电容（decoupling capacitor）外，LDO管脚还必须连接到板极的VDD25管脚。
	VBAT	55	-	电源	休眠模块的电源供应源。它通常连接到电池的正极端并用作备用电池/休眠模块电源供应器的电源。
	VDD	8	-	电源	I/O和某些逻辑的正极电源。
	VDD	20	-	电源	I/O和某些逻辑的正极电源。
	VDD	32	-	电源	I/O和某些逻辑的正极电源。
	VDD	44	-	电源	I/O和某些逻辑的正极电源。
	VDD	56	-	电源	I/O和某些逻辑的正极电源。
	VDD	68	-	电源	I/O和某些逻辑的正极电源。
	VDD	81	-	电源	I/O和某些逻辑的正极电源。
	VDD	93	-	电源	I/O和某些逻辑的正极电源。
	VDD25	14	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
	VDD25	38	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
	VDD25	62	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
	VDD25	88	-	电源	大多数逻辑功能（包括处理器内核和大部份外设）的电源正极。
	VDDA	3	-	电源	模拟电路（ADC、模拟比较器等）的电源正极(3.3 V)。这些电源与VDD独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
	VDDA	98	-	电源	模拟电路（ADC、模拟比较器等）的电源正极(3.3 V)。这些电源与VDD独立，以最大限度地减少VDD上的电气噪声，使其不影响模拟功能。
	WAKE	50	I	OD	当有效时外部输入将处理器从休眠模式中唤醒。

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
系统控制; 时钟	CMOD0	65	I/O	TTL	CPU模式位0。输入必须设为逻辑0（地）；其它编码保留。
系统控制;时钟	CMOD1	76	I/O	TTL	CPU模式位1。输入必须设为逻辑0（地）；其它编码保留。
	OSC0	48	I	模拟	主振荡器晶体输入或外部时钟参考输入。
	OSC1	49	O	模拟	主振荡器晶体输出。
	RST	64	I	TTL	系统复位输入。
	TRST	89	I	TTL	JTAG TRSTn
	XOSC0	52	I	模拟	休眠模块振荡器晶体输入或外部时钟参考输入。注意这是用于休眠模块RTC的4.19-MHz晶体或32.768-kHz振荡器。见**HIBCTL**寄存器的CLKSEL位。
	XOSC1	53	O	模拟	休眠模块振荡器晶体输出。
通用定时器	CCP0	66	I/O	TTL	捕获/比较/PWM 0
	CCP1	34	I/O	TTL	捕获/比较/PWM 1
	CCP2	67	I/O	TTL	捕获/比较/PWM 2
	CCP3	95	I/O	TTL	捕获/比较/PWM 3
	CCP4	35	I/O	TTL	捕获/比较/PWM 1
	CCP5	5	I/O	TTL	捕获/比较/PWM 5

表 **20-4. GPIO**管脚和备用功能

GPIO管脚	管脚编号	多路复用功能	多路复用功能
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	CCP1	
PA7	35	CCP4	
PB0	66	CCP0	
PB1	67	CCP2	
PB2	70	I2C0SCL	
PB3	71	I2C0SDA	
PB4	92	C0-	
PB5	91	C1-	
PB6	90	C0+	
PB7	89	TRST	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	PhA0	
PC5	24	C1+	

GPIO管脚	管脚编号	多路复用功能	多路复用功能
PC6	23	C2+	
PC7	22	C2-	
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PD4	95	CCP3	
PD5	96		
PD6	99	错误	
PD7	100	IDX0	
PE0	72	SSI1Clk	
PE1	73	SSI1Fss	
PE2	74	SSI1Rx	
PE3	75	SSI1Tx	
PE4	6		
PE5	5	CCP5	
PE6	2	PWM4	
PE7	1	PWM5	
PF0	47	CAN1Rx	
PF1	61	CAN1Tx	
PF2	60		
PF3	59		
PF4	58	C0o	
PF5	46	C1o	
PF6	43	C2o	
PF7	42		
PG0	19	U2Rx	
PG1	18	U2Tx	
PG2	17	PWM0	
PG3	16	PWM1	
PG4	41		
PG5	40		
PG6	37		
PG7	36		
PH0	86	PWM2	
PH1	85	PWM3	
PH2	84		
PH3	83	PhB0	

21 工作特性

表 **21-1.** 温度特性

特性	符号	值	单位
工作温度范围[a]	T_A	-40 ～ +85	°C

a. 最大的存储温度为150°C。

表 **21-2.** 热特性

特性	符号	值	单位
热电阻（结点到环境）[a]	Θ_{JA}	55.3	°C/W
平均结点温度[b]	T_J	$T_A + (P_{AVG} \bullet \Theta_{JA})$	°C

a. 结点到环境的热敏电阻θ_{JA}由封装模拟器（ package simulator）提供。

b. 功耗由温度决定。

22 电气特性

22.1 DC特性

22.1.1 最大额定值

最大额定值是指器件应该遵循的极限值，超过最大额定值可能会造成器件永久损坏。

注意: 不能保证器件在最大额定值下能正确工作。

表 **22-1.** 最大额定值

特性[a]	符号	值		单位
		Min	**Max**	
I/O电源电压（V_{DD}）	V_{DD}	0	4	V
内核电源电压（V_{DD25}）	V_{DD25}	0	4	V
模拟电源电压（V_{DDA}）	V_{DDA}	0	4	V
电池电源电压（V_{BAT}）	V_{BAT}	0	4	V
输入电压	V_{IN}	-0.3	5.5	V
每个输出管脚的最大电流	I	-	25	mA

a. 测量电压都是相对GND而言的。

重要: 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；但还是建议对该高阻抗电路采用正规的预防措施，以避免在具体的应用中电压超过最大额定电压。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。

22.1.2 建议的直流工作条件

表 **22-2.** 建议的直流工作条件

参数	参数名称	**Min**	**Nom**	**Max**	单位
V_{DD}	I/O电源电压	3.0	3.3	3.6	V
V_{DD25}	内核电源电压	2.25	2.5	2.75	V
V_{DDA}	模拟电源电压	3.0	3.3	3.6	V
V_{BAT}	电池电源电压	2.3	3.0	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}	施密特触发器输入的高电平输入电压	0.8 * V_{DD}	-	V_{DD}	V
V_{SIL}	施密特触发器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}	高电平拉电流（source current），V_{OH}=2.4 V				
	2mA驱动	2.0	-	-	mA
	4mA驱动	4.0	-	-	mA
	8mA驱动	8.0	-	-	mA

参数	参数名称	Min	Nom	Max	单位
I_{OL}	低电平灌电流（sink current），V_{OL}=0.4 V				
	2mA驱动	2.0	-	-	mA
	4mA驱动	4.0	-	-	mA
	8mA驱动	8.0	-	-	mA

22.1.3 片内低压差（LDO）稳压器特性

表 **22-3. LDO**稳压器特性

参数	参数名称	Min	Nom	Max	单位
V_{LDOOUT}	可编程内部（逻辑）电源输出值	2.25	2.5	2.75	V
	输出电压精度	-	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}	每一步的编程增量电压	-	50	-	mV
C_{LDO}	内部电源的外部滤波器电容的大小	1.0	-	3.0	µF

22.1.4 功率规范

下面的表格中指定的功率测量结果是使用SRAM的内核处理器在下列条件下得到的（特殊情况见注释）：

- V_{DD} = 3.3 V
- V_{DD25} = 2.50 V
- V_{BAT} = 3.0 V
- V_{DDA} = 3.3 V
- 温度 = 25°C
- 时钟源（MOSC） =3.579545 MHz的晶振
- 主振荡器（MOSC） = 使能
- 内部振荡器（IOSC） = 禁能

表 **22-4.** 详细的功率规范

参数	参数名称	条件	3.3 V V_{DD}, V_{DDA}, V_{DDPHY}		2.5 V V_{DD25}		3.0 V V_{BAT}		单位
			Nom	Max	Nom	Max	Nom	Max	
I_{DD_RUN}	运行模式1（Flash循环）	V_{DD25} = 2.50 V 代码= while(1){} 在Flash中执行 外设 = 全部开启 系统时钟 = 50 MHz（带PLL）	3	挂起[a]	108	挂起[a]	0	挂起[a]	mA
	运行模式2（Flash循环）	V_{DD25} = 2.50 V 代码= while(1){} 在Flash中执行 外设 = 全部关闭 系统时钟 = 50 MHz（带PLL）	0	挂起[a]	53	挂起[a]	0	挂起[a]	mA
	运行模式1（SRAM循环）	V_{DD25} = 2.50 V 代码= while(1){} 在SRAM中执行 外设 = 全部开启 系统时钟 = 50 MHz（带PLL）	3	挂起[a]	102	挂起[a]	0	挂起[a]	mA
	运行模式2（SRAM循环）	V_{DD25} = 2.50 V 代码= while(1){} 在SRAM中执行 外设 = 全部关闭 系统时钟 = 50 MHz（带PLL）	0	挂起[a]	47	挂起[a]	0	挂起[a]	mA
I_{DD_SLEEP}	睡眠模式	V_{DD25} = 2.50 V 外设 = 全部关闭 系统时钟 = 50 MHz（带PLL）	0	挂起[a]	17	挂起[a]	0	挂起[a]	mA
$I_{DD\_DEEPSLEEP}$	深度睡眠模式	LDO = 2.25 V 外设 = 全部关闭 系统时钟 = IOSC30KHZ/64	0.14	挂起[a]	0.18	挂起[a]	0	挂起[a]	mA
$I_{DD\_HIBERNATE}$	休眠模式	V_{BAT} = 3.0 V V_{DD} = 0 V V_{DD25} = 0 V V_{DDA} = 0 V V_{DDPHY} = 0 V 外设 = 全部关闭 系统时钟 = 关闭 休眠模式 = 32 kHz	0	挂起[a]	0	挂起[a]	16	挂起[a]	µA

a. 挂起结束。

22.1.5 Flash存储器特性

表 **22-5. Flash**存储器特性

参数	参数名称	Min	Nom	Max	单位
PE_{CYC}	故障出现之前可保证的编程/擦除周期数[a]	10,000	100,000	-	周期

参数	参数名称	Min	Nom	Max	单位
T_{RET}	平均温度为85°C下的数据保存时间	10	-	-	年
T_{PROG}	字编程时间	20	-	-	µs
T_{ERASE}	页擦除时间	20	-	-	ms
T_{ME}	整体擦除时间	200	-	-	ms

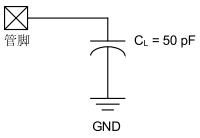
a. 编程/擦除周期定义成位从1变为0再变为1（1-> 0 -> 1）。

22.2 AC特性

22.2.1 负载条件

除非特别说明，否则下列条件对所有时间测量来说都是正确的。时序是在4-mA驱动强度下测量得到的。

图 **22-1.** 负载条件



22.2.2 时钟

表 **22-6.** 锁相环（**PLL**）特性

参数	参数名称	Min	Nom	Max	单位
$f_{ref\_crystal}$	晶体参考值[a]	3.579545	-	8.192	MHz
f_{ref_ext}	外部时钟基准[a]	3.579545	-	8.192	MHz
f_{pll}	PLL频率[b]	-	400	-	MHz
T_{READY}	PLL锁定时间	-	-	0.5	ms

a. 确切的晶体值由编程到运行模式时钟配置（**RCC**）寄存器XTAL域的值决定。

b. PLL的频率由硬件根据 **RCC**寄存器 XTAL域的值自动算出。

表 **22-7.** 时钟特性

参数	参数名称	Min	Nom	Max	单位
f_{IOSC}	内部12MHz的振荡器频率	8.4	12	15.6	MHz
$f_{IOSC30KHZ}$	内部30KHz的振荡器频率	21	30	39	KHz
f_{XOSC}	休眠模式的振荡器频率	-	4.194304	-	MHz
f_{XOSC_XTAL}	休眠振荡器的晶体参考值	-	4.194304	-	MHz
f_{XOSC_EXT}	休眠模块的外部时钟基准	-	32.768	-	KHz
f_{MOSC}	主振荡器频率	1	-	8	MHz
t_{MOSC_per}	主振荡器周期	125	-	1000	ns
$f_{ref\_crystal\_bypass}$	使用主振荡器作为晶体参考（PLL处于旁路模式）	1	-	8	MHz
$f_{ref\_ext\_bypass}$	外部时钟基准（PLL处于旁路模式）	0	-	50	MHz
f_{system_clock}	系统时钟	0	-	50	MHz

表 **22-8.** 晶体特性

参数名称	值				单位
频率	8	6	4	3.5	MHz
频率容限	±50	±50	±50	±50	ppm
老化	±5	±5	±5	±5	ppm/yr
振荡模式	并行	并行	并行	并行	
温度稳定性（0～ 85 °C）	±25	±25	±25	±25	ppm
动态电容（典型）	27.8	37.0	55.6	63.5	pF
动态电感（典型）	14.3	19.1	28.6	32.7	mH
等效串联电阻（最大）	120	160	200	220	Ω
滤波电容（最大）	10	10	10	10	pF
负载电容（典型）	16	16	16	16	pF
驱动级别（典型）	100	100	100	100	µW

22.2.3 模拟比较器

表 **22-9.** 模拟比较器特性

参数	参数名称	Min	Nom	Max	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	50	-	-	dB
T_{RT}	响应时间	-	-	1	µs
T_{MC}	比较器模式变成输出有效	-	-	10	µs

表 **22-10.** 模拟比较器电压参考特性

参数	参数名称	Min	Nom	Max	单位
R_{HR}	分辨率的高范围	-	$V_{DD}/32$	-	LSB
R_{LR}	分辨率的低范围	-	$V_{DD}/24$	-	LSB
A_{HR}	绝对精度的高范围	-	-	±1/2	LSB
A_{LR}	绝对精度的低范围	-	-	±1/4	LSB

22.2.4 I^2C

表 **22-11.** I^2C特性

参数编号	参数	参数名称	Min	Nom	Max	单位
I1[a]	t_{SCH}	起始条件保持时间	36	-	-	系统时钟
I2[a]	t_{LP}	时钟低电平时间	36	-	-	系统时钟
I3[b]	t_{SRT}	I2CSCL/I2CSDA上升时间 （V_{IL} =0.5 V～V_{IH} =2.4 V）	-	-	（见注b）	ns
I4[a]	t_{DH}	数据保持时间	2	-	-	系统时钟
I5[c]	t_{SFT}	I2CSCL/I2CSDA下降时间（V_{IH} =2.4 V～ V_{IL} =0.5 V）	-	9	10	ns
I6[a]	t_{HT}	时钟高电平时间	24	-	-	系统时钟
I7[a]	t_{DS}	数据建立时间	18	-	-	系统时钟
I8[a]	t_{SCSR}	起始条件的建立时间（只用于重复的起始条件）	36	-	-	系统时钟

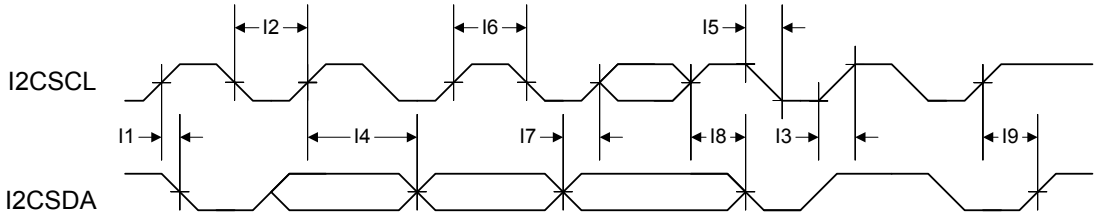
参数编号	参数	参数名称	Min	Nom	Max	单位
I9[a]	t_{SCS}	停止条件的建立时间	24	-	-	系统时钟

a. 值取决于**I²C**主机定时器周期（**I2CMTPR**）寄存器的TPR位设定的值；最大I2CSCL频率下设定的TPR值（TPR=0x2）会获得最小的输出时序，如上表所示。I²C接口用来调节实际的数据转换时间，使之移到I2CSCL低电平周期的中间。实际位置受TPR设定值的影响；但是，上表中给出的数目是最小值。

b. 由于I2CSCL和I2CSDA都是开漏类型的输出，控制器只能将输出驱动为低电平，因此，I2CSCL或I2CSDA到达高电平需要花费的时间由外部信号的容抗和上拉电阻值决定。

c. 在标称的50pF负载下指定。

图 **22-2. I²C时序**



22.2.5 休眠模块

由于休眠模块旨在关断主器件所有其它功能部件的供电，所以需要对系统实现方面进行特殊的考虑。系统电源分布和系统的接口必须被驱动为 0 V_{DC}，或者用HIB控制的相同稳压器关断。

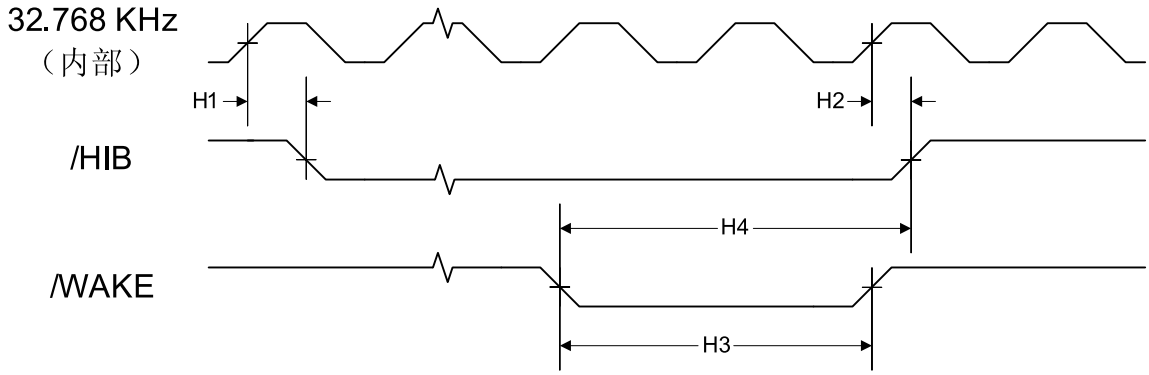
HIB控制的稳压器需要一个小于或等于250µs的稳定时间。

表 **22-12.** 休眠模块特性

参数编号	参数	参数名称	Min	Nom	Max	单位
H1	t_{HIB_LOW}	内部32.768KHz的时钟基准的上升沿到/HIB有效	-	200	-	µs
H2	t_{HIB_HIGH}	内部32.768KHz的时钟基准上升沿到/HIB无效	-	30	-	µs
H3	t_{WAKE_ASSERT}	/WAKE有效时间	62	-	-	µs
H4	$t_{WAKETOHIB}$	/WAKE有效到/HIB无效	62	-	124	µs
H5	t_{XOSC_SETTLE}	XOSC稳定时间[a]	20	-	-	ms
H6	$t_{HIB\_REG\_WRITE}$	完成写HIB模块中非易失性寄存器的时间	92	-	-	µs
H7	$t_{HIB\_TO\_VDD}$	HIB 在 VDD 和 VDD25 的最小操作电压下无效	-	-	250	µs

a. 这个参数极容易受PCB布线和走线长度的影响，从而使得这个参数时间更长一些。在PCB设计过程要特别注意，将走线长度和RLC（电阻、电感、电容）减到最小。

图 **22-3.** 休眠模块时序



22.2.6 同步串行接口（SSI）

表 **22-13. SSI**特性

参数编号	参数	参数名称	Min	Nom	Max	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 **22-4. TI**帧格式（**FRF=01**）的**SSI**时序，单次传输的时序测量

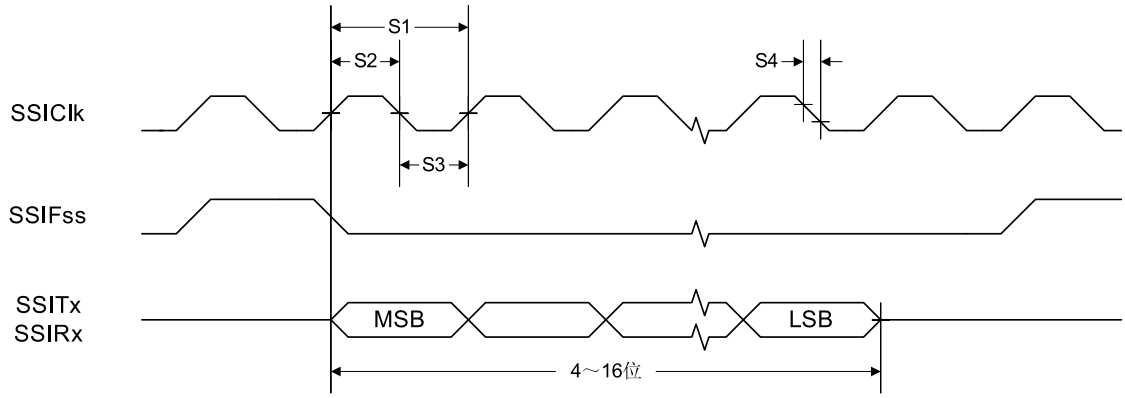


图 **22-5. MICROWIRE**帧格式（**FRF=10**）的**SSI**时序，单次传输

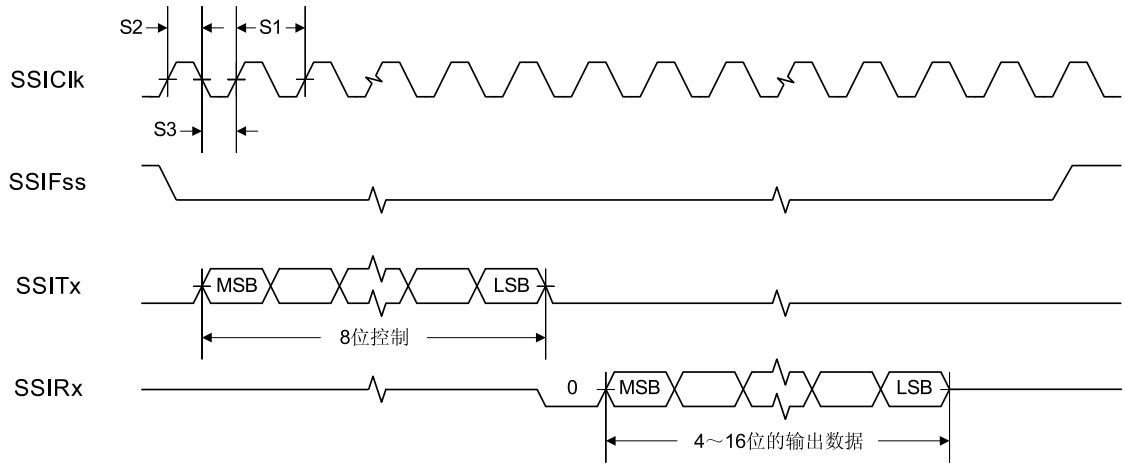
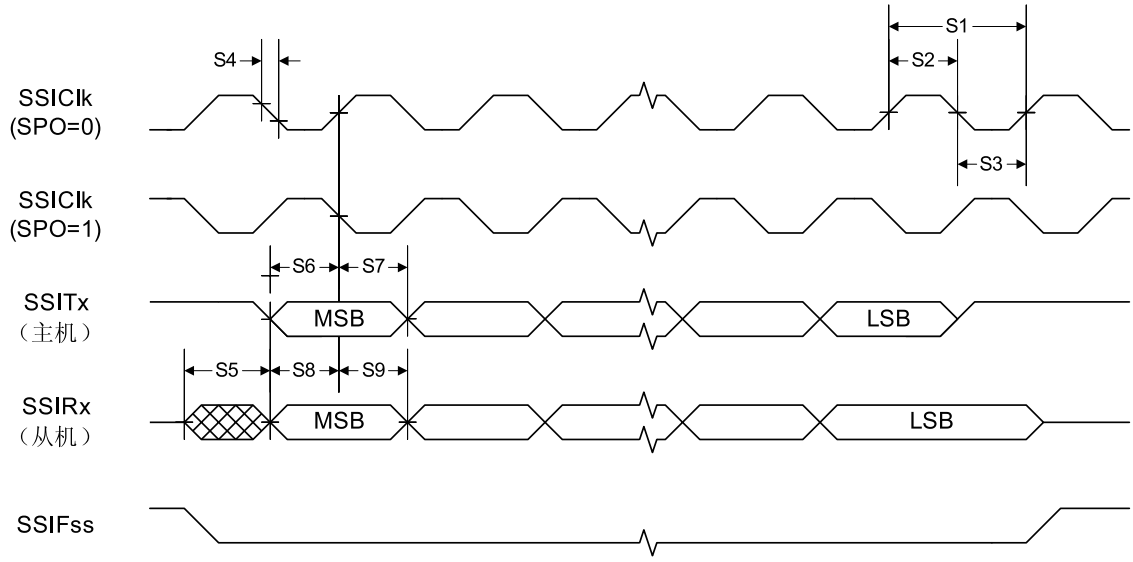


图 **22-6. SPI**帧格式（**FRF=00**），**SPH=1**



22.2.7 JTAG和边界扫描

表 **22-14. JTAG**特性

参数编号	参数	参数名称	Min	Nom	Max	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	-	t_{TCK}	-	ns
J4	t_{TCK_HIGH}	TCK时钟高电平时间	-	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上升沿为止的TMS建立时间	20	-	-	ns
J8	t_{TMS_HLD}	TCK上升沿开始算起的TMS保持时间	20	-	-	ns
J9	t_{TDI_SU}	到TCK上升沿为止的TDI建立时间	25	-	-	ns
J10	t_{TDI_HLD}	从TCK上升沿开始算起的TDI保持时间	25	-	-	ns
J11 t_{TDO_ZDV}	由TCK下降沿开始计算，高阻态到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
J12 t_{TDO_DV}	从TCK下降沿开始计算，两次数据有效之间的时间	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，数据有效到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
		带斜率控制的8-mA驱动		7	9	ns
J14	t_{TRST}	TRST生效的时间	100	-	-	ns

参数编号	参数	参数名称	Min	Nom	Max	单位
J15	t_{TRST_SU}	到TCK上升沿为止的TRST建立时间	10	-	-	ns

图 **22-7. JTAG**测试时钟输入时序

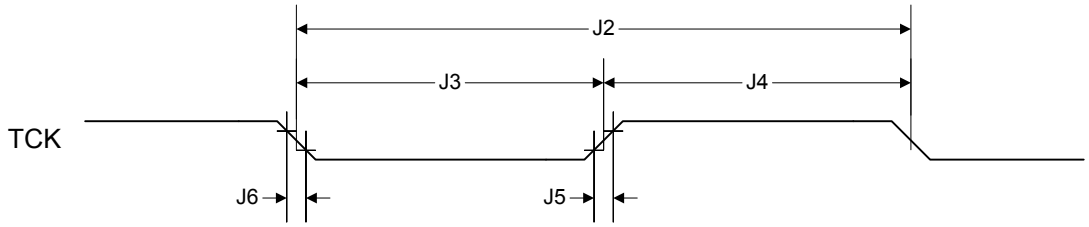


图 **22-8. JTAG**测试访问端口（**TAP**）时序

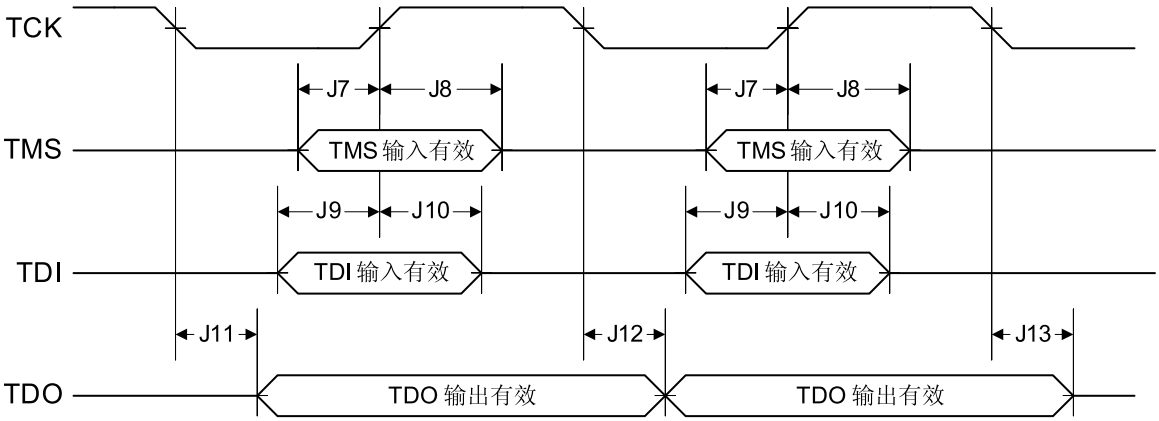
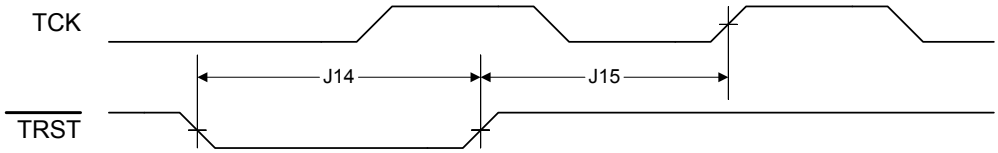


图 **22-9. JTAG TRST**时序



22.2.8 通用I/O口

注意: 所有GPIO口最大可承受5V的电压。

表 **22-15. GPIO**特性

参数	参数名称	条件	Min	Nom	Max	单位
t_{GPIOR}	GPIO上升时间（V_{DD}从20%上升到80%的时间）	2-mA驱动	-	17	26	ns
		4-mA驱动		9	13	ns
		8-mA驱动		6	9	ns
		带斜率控制的8-mA驱动		10	12	ns
t_{GPIOF}	GPIO下降时间（V_{DD}从80%下降到20%的时间）	2-mA驱动	-	17	25	ns
		4-mA驱动		8	12	ns
		8-mA驱动		6	10	ns
		带斜率控制的8-mA驱动		11	13	ns

22.2.9 复位

表 **22-16.** 复位特性

参数编号	参数	参数名称	Min	Nom	Max	单位
R1	V_{TH}	复位阈值	-	2.0	-	V
R2	V_{BTH}	掉电阈值	2.85	2.9	2.95	V
R3	T_{POR}	上电复位超时	-	10	-	ms
R4	T_{BOR}	掉电超时	-	500	-	µs
R5	T_{IRPOR}	POR之后内部复位的超时	6	-	11	ms
R6	T_{IRBOR}	BOR之后内部复位的超时[a]	0	-	1	µs
R7	T_{IRHWR}	硬件复位（$\overline{RST}$管脚）后内部复位的超时	0	-	1	ms
R8	T_{IRSWR}	软件启动的系统复位之后的内部复位的超时 [a]	2.5	-	20	µs
R9	T_{IRWDR}	看门狗复位之后的内部复位的超时[a]	2.5	-	20	µs
R10	$T_{VDDRISE}$	电源电压（V_{DD}）的上升时间（0V～3.3V）	-	-	100	ms
R11	T_{MIN}	最小$\overline{RST}$脉冲宽度	2	-	-	µs

a. 20 * t_{MOSC_per}

图 **22-10.** 外部复位时序（**$\overline{RST}$**）

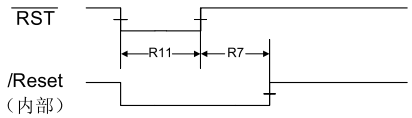


图 **22-11.** 上电复位时序

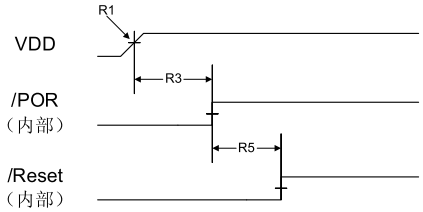


图 **22-12.** 掉电复位时序

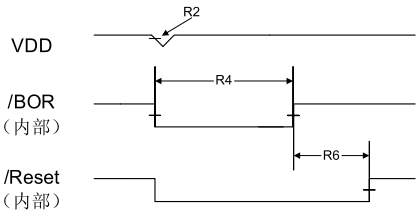


图 **22-13.** 软件复位时序

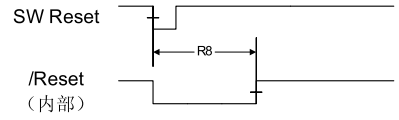
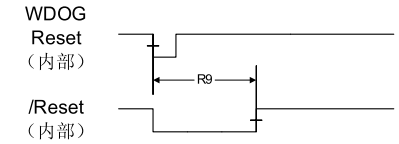
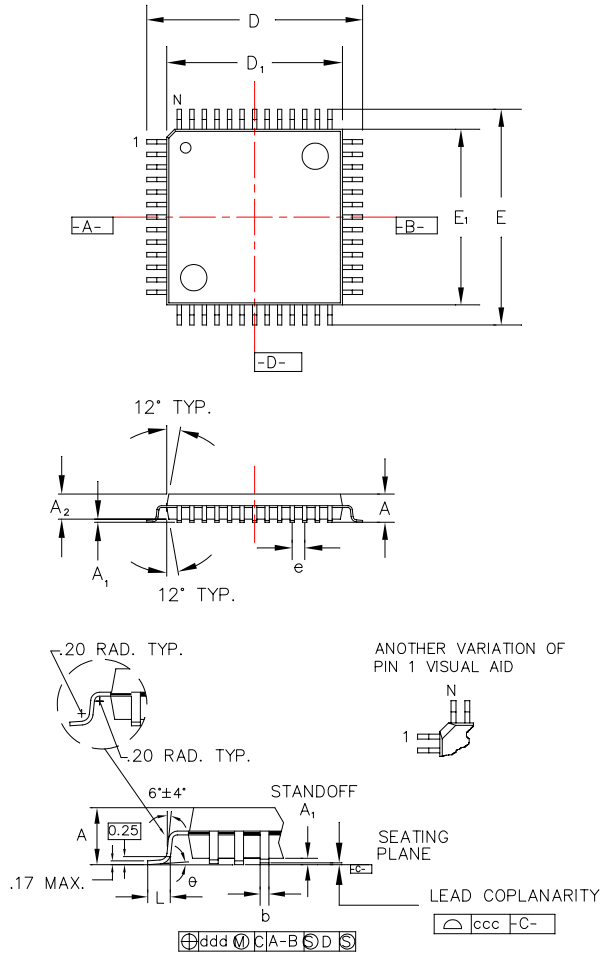


图 **22-14.** 看门狗复位时序



23 封装信息

图 **23-1. 100-**脚 **LQFP**封装



注意: 下列注释应用于封装图。

1. 所有尺寸均用mm表示。
2. 显示的尺寸是带有指定容差的标称尺寸。
3. 尺寸长度 'L'在底板以上0.25mm的标准板处测量。

本体 +2.00 mm 占地面积，1.4 mm封装厚度。		
符号	引脚	100L
A	最大	1.60
A_1		最小0.05/ 最大0.15
A_2	±0.05	1.40
D	±0.20	16.00
D_1	±0.05	14.00

E	±0.20	16.00
E_1	±0.05	14.00
L	±0.15/-0.10	0.60
e	BASIC	0.50
b	±0.05	0.22
θ	===	0˚~7˚
ddd	最大	0.08
ccc	最大	0.08
JEDEC 参考图		MS-026
变量标志符		BED

A 串行Flash加载程序

A.1 串行Flash加载程序

用户通过使用Stellaris® 串行Flash加载程序，无需使用调试接口，就可将代码下载到Flash存储器中。串行Flash加载程序通过一个简单的包接口（packet interface）与器件进行同步通信。Flash加载程序使用晶振，无需使能PLL，所以其速率取决于所使用的晶体。可以使用的两个串行接口分别是UART0和SSI接口。为方便起见，这两种串行接口的数据格式和通信协议都相同。

A.2 接口

用户一旦通过其中一个串行接口与Flash加载程序建立通信后，就立即使用包接口，并且一直使用，直至Flash加载程序复位或新代码掌握控制权。例如，一旦用户使用SSI端口进行通信后，通过UART与Flash的通信就会立刻被禁止，直到器件复位。

A.2.1 UART

通用异步收发器（UART）的通信采用的是固定的串行格式：8个数据位、无奇偶校验位、1个停止位。通信速率可以由Flash加载程序自动检测，它可以是主控器和器件所支持的任意有效波特率。自动检测序列要求波特率不能高于运行该串行Flash加载程序的电路板所具有的晶体频率的1/32。这实际上和通过硬件限制Stellaris®器件上所有UART的最大波特率的情况是一样的，该波特率如下计算：

```
最大波特率 = 系统时钟频率 / 16
```

要确定波特率，串行Flash加载程序首先必须确定自己的晶体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的波特率配置成和主控器一样。通过这种波特率自动检测技术，主机可以采用任意有效的波特率与器件进行通信。

在执行这种自动同步功能时，主控器必须向Flash加载程序发送两个均为0x55的字节。这样就可以向Flash加载程序生成一系列脉冲，这些脉冲可用来计算对UART进行编程以与主控器的波特率相同时所需的比率。在主控器发送了用于同步的格式之后，它将试图从UART中读回一个字节数据。Flash加载程序返回0xCC来表示波特率检测成功。如果在至少2倍于传输两个字节所需的时间内还没有接收到该字节，那么主控器可以重新发送另外一种格式：0x55，0x55，并再次等待0xCC字节，直至Flash加载程序确认已经正确接收到同步格式。例如，等待数据从Flash加载程序返回的时间应该为2*（20(位/同步)/波特率(位/同步)）。要得到115200的波特率，时间必须设为2*（20/115200）或0.35ms。

A.2.2 SSI

同步串行接口（SSI）端口也使用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也设为1。有关该传输协议的格式的详细内容请见 SSI小节 “帧格式” 在 289页。与UART类似，SSI接口对硬件也有一定的要求，限制SSI时钟可以运行的最大速率。这样，SSI最大的时钟速率为运行Flash加载程序的电路板所具有的晶体频率的1/12。由于主器件作为主控器，因此，时钟直接由主控器提供，Flash加载程序上的SSI不需要确定时钟。

A.3 信息包处理

除UART自动波特率（auto-baud）之外的所有通信，都是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）。接收和发送信息包所使用的格式都相同，包括对信息包的成功接收和未接收作出应答时所使用的方法也相同。

A.3.1 信息包的格式

所有从器件中发送和接收的信息包均使用下列字节封包格式：

```
struct
{
 unsigned char ucSize;
 unsigned char ucCheckSum;
 unsigned char Data[];
};
```

ucSize	接收到的第一个字节，包含的是总的传输长度，包括数据长度以及校验和字节。
ucChecksum	保存的只是对数据缓冲区中的字节进行简单校验和而得的结果。算法是： Data[0]+Data[1]+…+ Data[ucSize-3].
Data	打算供器件使用的原始数据，它按照某些命令接口的形式进行格式化。此时，该数据缓冲区应向器件发送或从器件接收ucSize–2 个数据字节。

A.3.2 信息包的发送

实际的信息包字节可以一个一个单独地发送，也可以一次全部发送；唯一的局限性就是引发Flash存储器访问的命令应该限定下载的字节数，以防止在Flash编程器件丢失字节。这种限定会在描述串行 flash 加载程序命令 COMMAND_SEND_DATA 中进一步讨论(见“COMMAND_SEND_DATA (0x24)” 在 482页)。

一旦主控器对数据包进行正确的格式化之后，数据包就必须通过UART或SSI接口发送出去。然后主控器轮询UART或SSI接口以查找从器件中返回的第一个非零数据。该非零字节可以是来自器件的ACK（0xCC）或（ACK）的NAK（0x33），表明成功接收到信息包，或未能接收到信息包。但这并不表示位于信息包数据部分的已发送命令的实际内容就是有效的，它仅仅表示已经成功接收到信息包。

A.3.3 信息包的接收

Flash加载程序发送数据包与接收数据包的格式相同。它在发送第一个实际的数据字节前会发送一个前导零。第一个非零字节表示紧跟校验和字节之后的信息包的长度，最后跟着数据本身。在Flash加载程序发送第一个非零字节之后，数据之间便没有间隔。一旦与Flash加载程序通信的器件接收完所有字节，它就必须对信息包作出ACK或NAK应答以指示发送是否成功。如果器件向Flash加载程序发送的是NAK，那么它将重新发送失败的命令，并再次请求发送数据。由于Flash加载程序只将接收到的第一个非0数据作为有效响应，所以，如有必要，主控器会在向下发送ACK/NAK信号给Flash加载程序前先发送前导零。为了从Flash加载程序中接收数据或向其发送数据，SSI接口就需要使用这种填充零操作。

A.4 命令

下文将对能够发送到Flash加载程序的命令列表进行详细介绍。数据的第一个字节始终为其中的一个已定义命令，后面跟着的是由被发送命令定义的数据或参数。

A.4.1 COMMAND_PING (0X20)

本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命令含3个字节，COMMAND_PING的值为0x20，且一个字节的校验和即为该字节本身，因此，Byte[1]也为0x20。 由于ping命令不含真正的返回状态，因此ACK应答可以解释为成功地对Flash加载程序执行了ping操作。

A.4.2 COMMAND_GET_STATUS (0x23)

本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才发送，以确保之前的命令发送成功，或者如果失败，则可确保正确地对失败进行响应。本命令首先需要信息包数据中的一个字节，然后对含有状态码的一个字节数据的信息包进行读操作。最后一步便是是对已接收数据作出ACK或NAK应答，以便Flash加载程序知道是否已经读取完数据。

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.4.3 COMMAND_DOWNLOAD (0x21)

本命令要发送到Flash加载程序，以表明存储数据的位置以及在COMMAND_SEND_DATA 之后应发送的字节数。本命令由均先传输最高位（MSB）的2个32位值组成。第一个32位值是对数据进行编程的起始地址，第二个32位值是要发送的数据的字节数。本命令还可以对整个编程空间触发一次擦除操作，因此本命令与其他命令相比，所花费的时间将更长。这样，接收电路板的ACK/NAK响应所需的时间也将更长。本命令后面必须紧跟一个COMMAND_GET_STATUS命令，以确保程序（Program）地址和（Program）大小对于运行Flash加载程序的器件是有效的。

发送本命令的包格式如下所示：

```
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.4.4 COMMAND_SEND_DATA (0x24)

本命令的后面应该只紧跟一个COMMAND_DOWNLOAD命令，或者如果需要更多数据，还可以紧跟另外一个COMMAND_SEND_DATA命令。连续的发送数据命令会使地址自动递增，并从先前的位置继续编程。调用者应该将数据的传输限制为一个信息包最多含8个字节，以便对Flash成功编程，并且不会让串行接口输入缓冲区出现溢出状况。一旦接收完由COMMAND_DOWNLOAD命令指示的字节数，本命令就会终止编程。每次调用该功能时，它之后都必须紧跟一个COMMAND_GET_STATUS命令，以确保数据成功编入Flash中。如果Flash加载程序向本命令发送一个NAK应答，Flash加载程序将不会使当前地址递增，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.4.5 COMMAND_RUN (0x22)

本命令用于告知Flash加载程序从作为本命令中的参数进行传递的地址处执行。本命令由一个32位值组成，该值被解释为执行时的地址。32位值首先发送最高位（MSB），并且在给定地址处真正执行代码之前，Flash加载程序会向主机返回一个ACK信号作为应答。这样，主控器就知道是否已成功接收该命令，以及代码是否正在运行。

```
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.4.6 COMMAND_RESET (0x25)

本命令用于通知Flash加载程序件进行复位。这在下载一个覆写Flash加载程序的新映象（image），以及希望从一个完全复位的状态启动时是非常有用。与COMMAND_RUN命令不同的是，COMMAND_RESET命令允许硬件读取初始的堆栈指针以及为新代码设立初始堆栈指针。如果出现重大的错误，或者是主器件想重新与Flash加载程序通信时，本命令还可以用来复位Flash加载程序。

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

Flash加载程序在对运行Flash加载程序的器件实际执行软件复位前，会向主控器返回一个ACK信号作为应答。这样，主控器就知道该命令已被成功接收，器件将被复位。

B 寄存器快速参考

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
系统控制															
DID0, 类型 **RO,** 偏移量 **0x000,** 复位 **-**															
	VER							CLASS							
MAJOR								MINOR							
PBORCTL, 类型 **R/W,** 偏移量 **0x030,** 复位 **0x0000.7FFD**															
														BORIOR	
LDOPCTL, 类型 **R/W,** 偏移量 **0x034,** 复位 **0x0000.0000**															
										VADJ					
RIS, 类型 **RO,** 偏移量 **0x050,** 复位 **0x0000.0000**															
									PLLLRIS					BORRIS	
IMC, 类型 **R/W,** 偏移量 **0x054,** 复位 **0x0000.0000**															
									PLLLIM					BORIM	
MISC, 类型 **R/W1C,** 偏移量 **0x058,** 复位 **0x0000.0000**															
									PLLLMIS					BORMIS	
RESC, 类型 **R/W,** 偏移量 **0x05C,** 复位 **-**															
										LDO	SW	WDT	BOR	POR	EXT
RCC, 类型 **R/W,** 偏移量 **0x060,** 复位 **0x07AE.3AD1**															
				ACG	SYSDIV			USESYSDIV		USEPWMDIV	PWMDIV				
		PWRDN		BYPASS	XTAL			OSCSRC						IOSCDIS	MOSCDIS
PLLCFG, 类型 **RO,** 偏移量 **0x064,** 复位 **-**															
		F									R				
RCC2, 类型 **R/W,** 偏移量 **0x070,** 复位 **0x0780.2800**															
USERCC2			SYSDIV2												
		PWRDN2		BYPASS2					OSCSRC2						
DSLPCLKCFG, 类型 **R/W,** 偏移量 **0x144,** 复位 **0x0780.0000**															
			DSDIVORIDE												
									DSOSCSRC						
DID1, 类型 **RO,** 偏移量 **0x004,** 复位 **-**															
VER				FAM				PARTNO							
PINCOUNT								TEMP			PKG		ROHS	QUAL	
DC0, 类型 **RO,** 偏移量 **0x008,** 复位 **0x00FF.007F**															
SRAMSZ															
FLASHSZ															
DC1, 类型 **RO,** 偏移量 **0x010,** 复位 **0x0310.30DF**															
						CAN1	CAN0				PWM				
MINSYSDIV								MPU	HIB		PLL	WDT	SWO	SWD	JTAG
DC2, 类型 **RO,** 偏移量 **0x014,** 复位 **0x070F.1137**															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
			I2C0				QEI0			SSI1	SSI0		UART2	UART1	UART0
DC3, 类型 **RO,** 偏移量 **0x018,** 复位 **0x3F00.FFFF**															
		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0								
PWMFAULT	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DC4, 类型 **RO,** 偏移量 **0x01C,** 复位 **0x0000.00FF**															
								GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
RCGC0, 类型 **RW,** 偏移量 **0x100,** 复位 **0x00000040**															
						CAN1	CAN0				PWM				
									HIB			WDT			
SCGC0, 类型 **RW,** 偏移量 **0x110,** 复位 **0x00000040**															
						CAN1	CAN0				PWM				
									HIB			WDT			
DCGC0, 类型 **RW,** 偏移量 **0x120,** 复位 **0x00000040**															
						CAN1	CAN0				PWM				
									HIB			WDT			
RCGC1, 类型 **RW,** 偏移量 **0x104,** 复位 **0x00000000**															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
			I2C0				QEI0			SSI1	SSI0		UART2	UART1	UART0
SCGC1, 类型 **RW,** 偏移量 **0x114,** 复位 **0x00000000**															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
			I2C0				QEI0			SSI1	SSI0		UART2	UART1	UART0
DCGC1, 类型 **RW,** 偏移量 **0x124,** 复位 **0x00000000**															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
			I2C0				QEI0			SSI1	SSI0		UART2	UART1	UART0
RCGC2, 类型 **RW,** 偏移量 **0x108,** 复位 **0x00000000**															
								GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, 类型 **RW,** 偏移量 **0x118,** 复位 **0x00000000**															
								GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, 类型 **RW,** 偏移量 **0x128,** 复位 **0x00000000**															
								GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, 类型 **RW,** 偏移量 **0x040,** 复位 **0x00000000**															
						CAN1	CAN0				PWM				
									HIB			WDT			
SRCR1, 类型 **RW,** 偏移量 **0x044,** 复位 **0x00000000**															
					COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
			I2C0				QEI0			SSI1	SSI0		UART2	UART1	UART0
SRCR2, 类型 **RW,** 偏移量 **0x048,** 复位 **0x00000000**															
								GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
休眠模块															
HIBRTCC, 类型 **RO,** 偏移量 **0x000,** 复位 **0x0000.0000**															
RTCC															
RTCC															
HIBRTCM0, 类型 **R/W,** 偏移量 **0x004,** 复位 **0xFFFF.FFFF**															
RTCM0															
RTCM0															
HIBRTCM1, 类型 **R/W,** 偏移量 **0x008,** 复位 **0xFFFF.FFFF**															
RTCM1															
RTCM1															
HIBRTCLD, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0xFFFF.FFFF**															
RTCLD															
RTCLD															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIBCTL, 类型 **R/W**, 偏移量 **0x010**, 复位 **0x0000.0000**															
								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM, 类型 **R/W**, 偏移量 **0x014**, 复位 **0x0000.0000**															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRIS, 类型 **RO**, 偏移量 **0x018**, 复位 **0x0000.0000**															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBMIS, 类型 **RO**, 偏移量 **0x01C**, 复位 **0x0000.0000**															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBIC, 类型 **R/W1C**, 偏移量 **0x020**, 复位 **0x0000.0000**															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRTCT, 类型 **R/W**, 偏移量 **0x024**, 复位 **0x0000.7FFF**															
TRIM															
HIBDATA, 类型 **R/W**, 偏移量 **0x030- 0x12C**, 复位 **0x0000.0000**															
RTD															
RTD															
内部存储器 **Flash**控制偏移量															
FMA, 类型 **R/W**, 偏移量 **0x000**, 复位 **0x0000.0000**															
														OFFSET	
OFFSET															
FMD, 类型 **R/W**, 偏移量 **0x004**, 复位 **0x0000.0000**															
DATA															
DATA															
FMC, 类型 **R/W**, 偏移量 **0x008**, 复位 **0x0000.0000**															
WRKEY															
												COMT	MERASE	ERASE	WRITE
FCRIS, 类型 **RO**, 偏移量 **0x00C**, 复位 **0x0000.0000**															
														PRIS	ARIS
FCIM, 类型 **R/W**, 偏移量 **0x010**, 复位 **0x0000.0000**															
														PMASK	AMASK
FCMISC, 类型 **R/W1C**, 偏移量 **0x014**, 复位 **0x0000.0000**															
														PMISC	AMISC
内部存储器 系统控制偏移量															
USECRL, 类型 **R/W**, 偏移量 **0x140**, 复位 **0x31**															
								USEC							
FMPRE0, 类型 **R/W**, 偏移量 **0x130 and(cn) 0x200**, 复位 **0xFFFF.FFFF**															
READ_ENABLE															
READ_ENABLE															
FMPPE0, 类型 **R/W**, 偏移量 **0x134 and(cn) 0x400**, 复位 **0xFFFF.FFFF**															
PROG_ENABLE															
PROG_ENABLE															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_DBG, 类型 **RW,** 偏移量 **0x1D0,** 复位 **0xFFFF.FFFE**															
NW	DATA														
DATA														DBG1	DBG0
USER_REG0, 类型 **RW,** 偏移量 **0x1E0,** 复位 **0xFFFF.FFFF**															
NW	DATA														
DATA															
USER_REG1, 类型 **RW,** 偏移量 **0x1E4,** 复位 **0xFFFF.FFFF**															
NW	DATA														
DATA															
FMPRE1, 类型 **RW,** 偏移量 **0x204,** 复位 **0xFFFF.FFFF**															
READ_ENABLE															
READ_ENABLE															
FMPRE2, 类型 **RW,** 偏移量 **0x208,** 复位 **0xFFFF.FFFF**															
READ_ENABLE															
READ_ENABLE															
FMPRE3, 类型 **RW,** 偏移量 **0x20C,** 复位 **0xFFFF.FFFF**															
READ_ENABLE															
READ_ENABLE															
FMPPE1, 类型 **RW,** 偏移量 **0x404,** 复位 **0xFFFF.FFFF**															
PROG_ENABLE															
PROG_ENABLE															
FMPPE2, 类型 **RW,** 偏移量 **0x408,** 复位 **0xFFFF.FFFF**															
PROG_ENABLE															
PROG_ENABLE															
FMPPE3, 类型 **RW,** 偏移量 **0x40C,** 复位 **0xFFFF.FFFF**															
PROG_ENABLE															
PROG_ENABLE															
通用输入/输出端口（**GPIO**）															
GPIODATA, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
								DATA							
GPIODIR, 类型 **R/W,** 偏移量 **0x400,** 复位 **0x0000.0000**															
								DIR							
GPIOIS, 类型 **R/W,** 偏移量 **0x404,** 复位 **0x0000.0000**															
								IS							
GPIOIBE, 类型 **R/W,** 偏移量 **0x408,** 复位 **0x0000.0000**															
								IBE							
GPIOIEV, 类型 **R/W,** 偏移量 **0x40C,** 复位 **0x0000.0000**															
								IEV							
GPIOIM, 类型 **R/W,** 偏移量 **0x410,** 复位 **0x0000.0000**															
								IME							
GPIORIS, 类型 **RO,** 偏移量 **0x414,** 复位 **0x0000.0000**															
								RIS							
GPIOMIS, 类型 **RO,** 偏移量 **0x418,** 复位 **0x0000.0000**															
								MIS							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOICR, 类型 **W1C,** 偏移量 **0x41C,** 复位 **0x0000.0000**															
								IC							
GPIOAFSEL, 类型 **R/W,** 偏移量 **0x420,** 复位 **-**															
								AFSEL							
GPIODR2R, 类型 **R/W,** 偏移量 **0x500,** 复位 **0x0000.00FF**															
								DRV2							
GPIODR4R, 类型 **R/W,** 偏移量 **0x504,** 复位 **0x0000.0000**															
								DRV4							
GPIODR8R, 类型 **R/W,** 偏移量 **0x508,** 复位 **0x0000.0000**															
								DRV8							
GPIOODR, 类型 **R/W,** 偏移量 **0x50C,** 复位 **0x0000.0000**															
								ODE							
GPIOPUR, 类型 **R/W,** 偏移量 **0x510,** 复位 **-**															
								PUE							
GPIOPDR, 类型 **R/W,** 偏移量 **0x514,** 复位 **0x0000.0000**															
								PDE							
GPIOSLR, 类型 **R/W,** 偏移量 **0x518,** 复位 **0x0000.0000**															
								SRL							
GPIODEN, 类型 **R/W,** 偏移量 **0x51C,** 复位 **-**															
								DEN							
GPIOLOCK, 类型 **R/W,** 偏移量 **0x520,** 复位 **0x0000.0001**															
								LOCK							
								LOCK							
GPIOCR, 类型 **-,** 偏移量 **0x524,** 复位 **-**															
								CR							
GPIOPeriphID4, 类型 **RO,** 偏移量 **0xFD0,** 复位 **0x0000.0000**															
								PID4							
GPIOPeriphID5, 类型 **RO,** 偏移量 **0xFD4,** 复位 **0x0000.0000**															
								PID5							
GPIOPeriphID6, 类型 **RO,** 偏移量 **0xFD8,** 复位 **0x0000.0000**															
								PID6							
GPIOPeriphID7, 类型 **RO,** 偏移量 **0xFDC,** 复位 **0x0000.0000**															
								PID7							
GPIOPeriphID0, 类型 **RO,** 偏移量 **0xFE0,** 复位 **0x0000.0061**															
								PID0							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPeriphID1, 类型 **RO,** 偏移量 **0xFE4,** 复位 **0x0000.0000**															
								PID1							
GPIOPeriphID2, 类型 **RO,** 偏移量 **0xFE8,** 复位 **0x0000.0018**															
								PID2							
GPIOPeriphID3, 类型 **RO,** 偏移量 **0xFEC,** 复位 **0x0000.0001**															
								PID3							
GPIOPCellID0, 类型 **RO,** 偏移量 **0xFF0,** 复位 **0x0000.000D**															
								CID0							
GPIOPCellID1, 类型 **RO,** 偏移量 **0xFF4,** 复位 **0x0000.00F0**															
								CID1							
GPIOPCellID2, 类型 **RO,** 偏移量 **0xFF8,** 复位 **0x0000.0005**															
								CID2							
GPIOPCellID3, 类型 **RO,** 偏移量 **0xFFC,** 复位 **0x0000.00B1**															
								CID3							
通用定时器															
GPTMCFG, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
													GPTMCFG		
GPTMTAMR, 类型 **R/W,** 偏移量 **0x004,** 复位 **0x0000.0000**															
												TAAMS	TACMR	TAMR	
GPTMTBMR, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
												TBAMS	TBCMR	TBMR	
GPTMCTL, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0x0000.0000**															
	TBPWML	TBOTE		TBEVENT		TBSTALL	TBEN		TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
GPTMIMR, 类型 **R/W,** 偏移量 **0x018,** 复位 **0x0000.0000**															
					CBEIM	CBMIM	TBTOIM					RTCIM	CAEIM	CAMIM	TATOIM
GPTMRIS, 类型 **RO,** 偏移量 **0x01C,** 复位 **0x0000.0000**															
					CBERIS	CBMRIS	TBTORIS					RTCRIS	CAERIS	CAMRIS	TATORIS
GPTMMIS, 类型 **RO,** 偏移量 **0x020,** 复位 **0x0000.0000**															
					CBEMIS	CBMMIS	TBTOMIS					RTCMIS	CAEMIS	CAMMIS	TATOMIS
GPTMICR, 类型 **W1C,** 偏移量 **0x024,** 复位 **0x0000.0000**															
					CBECINT	CBMCINT	TBTOCINT					RTCCINT	CAECINT	CAMCINT	TATOCINT
GPTMTAILR, 类型 **R/W,** 偏移量 **0x028,** 复位 **0x0000.FFFF**（**16**位模式） 和 **0xFFFF.FFFF**（**32**位模式）															
								TAILRH							
								TAILRL							
GPTMTBILR, 类型 **R/W,** 偏移量 **0x02C,** 复位 **0x0000.FFFF**															
								TBILRL							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPTMTAMATCHR, 类型 **R/W,** 偏移量 **0x030,** 复位 **0x0000.FFFF**（**16**位模式） 和 **0xFFFF.FFFF**（**32**位模式）															
TAMRH															
TAMRL															
GPTMTBMATCHR, 类型 **R/W,** 偏移量 **0x034,** 复位 **0x0000.FFFF**															
TBMRL															
GPTMTAPR, 类型 **R/W,** 偏移量 **0x038,** 复位 **0x0000.0000**															
								TAPSR							
GPTMTBPR, 类型 **R/W,** 偏移量 **0x03C,** 复位 **0x0000.0000**															
								TBPSR							
GPTMTAPMR, 类型 **R/W,** 偏移量 **0x040,** 复位 **0x0000.0000**															
								TAPSMR							
GPTMTBPMR, 类型 **R/W,** 偏移量 **0x044,** 复位 **0x0000.0000**															
								TBPSMR							
GPTMTAR, 类型 **RO,** 偏移量 **0x048,** 复位 **0x0000.FFFF** （**16**位模式） 和 **0xFFFF.FFFF**（**32**位模式）															
TARH															
TARL															
GPTMTBR, 类型 **RO,** 偏移量 **0x04C,** 复位 **0x0000.FFFF**															
TBRL															
看门狗定时器															
WDTLOAD, 类型 **R/W,** 偏移量 **0x000,** 复位 **0xFFFF.FFFF**															
WDTLoad															
WDTLoad															
WDTVALUE, 类型 **RO,** 偏移量 **0x004,** 复位 **0xFFFF.FFFF**															
WDTValue															
WDTValue															
WDTCTL, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
														RESEN	INTEN
WDTICR, 类型 **WO,** 偏移量 **0x00C,** 复位 **-**															
WDTIntClr															
WDTIntClr															
WDTRIS, 类型 **RO,** 偏移量 **0x010,** 复位 **0x0000.0000**															
															WDTRIS
WDTMIS, 类型 **RO,** 偏移量 **0x014,** 复位 **0x0000.0000**															
															WDTMIS
WDTTEST, 类型 **R/W,** 偏移量 **0x418,** 复位 **0x0000.0000**															
							STALL								
WDTLOCK, 类型 **R/W,** 偏移量 **0xC00,** 复位 **0x0000.0000**															
WDTLock															
WDTLock															
WDTPeriphID4, 类型 **RO,** 偏移量 **0xFD0,** 复位 **0x0000.0000**															
								PID4							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTPeriphID5, 类型 **RO,** 偏移量 **0xFD4,** 复位 **0x0000.0000**															
								PID5							
WDTPeriphID6, 类型 **RO,** 偏移量 **0xFD8,** 复位 **0x0000.0000**															
								PID6							
WDTPeriphID7, 类型 **RO,** 偏移量 **0xFDC,** 复位 **0x0000.0000**															
								PID7							
WDTPeriphID0, 类型 **RO,** 偏移量 **0xFE0,** 复位 **0x0000.0005**															
								PID0							
WDTPeriphID1, 类型 **RO,** 偏移量 **0xFE4,** 复位 **0x0000.0018**															
								PID1							
WDTPeriphID2, 类型 **RO,** 偏移量 **0xFE8,** 复位 **0x0000.0018**															
								PID2							
WDTPeriphID3, 类型 **RO,** 偏移量 **0xFEC,** 复位 **0x0000.0001**															
								PID3							
WDTPCellID0, 类型 **RO,** 偏移量 **0xFF0,** 复位 **0x0000.000D**															
								CID0							
WDTPCellID1, 类型 **RO,** 偏移量 **0xFF4,** 复位 **0x0000.00F0**															
								CID1							
WDTPCellID2, 类型 **RO,** 偏移量 **0xFF8,** 复位 **0x0000.0005**															
								CID2							
WDTPCellID3, 类型 **RO,** 偏移量 **0xFFC,** 复位 **0x0000.00B1**															
								CID3							
通用异步收发器 **(UART)**															
UARTDR, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
				OE	BE	PE	FE	DATA							
UARTRSR/ UARTECR, 类型 **RO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
												OE	BE	PE	FE
UARTRSR/ UARTECR, 类型 **WO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
								DATA							
UARTFR, 类型 **RO,** 偏移量 **0x018,** 复位 **0x0000.0090**															
								TXFE	RXFF	TXFF	RXFE	BUSY			
UARTILPR, 类型 **R/W,** 偏移量 **0x020,** 复位 **0x0000.0000**															
								ILPDVSR							
UARTIBRD, 类型 **R/W,** 偏移量 **0x024,** 复位 **0x0000.0000**															
DIVINT															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UARTFBRD, 类型 **R/W,** 偏移量 **0x028,** 复位 **0x0000.0000**															
										DIVFRAC					
UARTLCRH, 类型 **R/W,** 偏移量 **0x02C,** 复位 **0x0000.0000**															
								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
UARTCTL, 类型 **R/W,** 偏移量 **0x030,** 复位 **0x0000.0300**															
						RXE	TXE	LBE					SIRLP	SIREN	UARTEN
UARTIFLS, 类型 **R/W,** 偏移量 **0x034,** 复位 **0x0000.0012**															
										RXIFLSEL			TXIFLSEL		
UARTIM, 类型 **R/W,** 偏移量 **0x038,** 复位 **0x0000.0000**															
					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM				
UARTRIS, 类型 **RO,** 偏移量 **0x03C,** 复位 **0x0000.000F**															
					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS				
UARTMIS, 类型 **RO,** 偏移量 **0x040,** 复位 **0x0000.0000**															
					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS				
UARTICR, 类型 **W1C,** 偏移量 **0x044,** 复位 **0x0000.0000**															
					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC				
UARTPeriphID4, 类型 **RO,** 偏移量 **0xFD0,** 复位 **0x0000.0000**															
								PID4							
UARTPeriphID5, 类型 **RO,** 偏移量 **0xFD4,** 复位 **0x0000.0000**															
								PID5							
UARTPeriphID6, 类型 **RO,** 偏移量 **0xFD8,** 复位 **0x0000.0000**															
								PID6							
UARTPeriphID7, 类型 **RO,** 偏移量 **0xFDC,** 复位 **0x0000.0000**															
								PID7							
UARTPeriphID0, 类型 **RO,** 偏移量 **0xFE0,** 复位 **0x0000.0011**															
								PID0							
UARTPeriphID1, 类型 **RO,** 偏移量 **0xFE4,** 复位 **0x0000.0000**															
								PID1							
UARTPeriphID2, 类型 **RO,** 偏移量 **0xFE8,** 复位 **0x0000.0018**															
								PID2							
UARTPeriphID3, 类型 **RO,** 偏移量 **0xFEC,** 复位 **0x0000.0001**															
								PID3							
UARTPCellID0, 类型 **RO,** 偏移量 **0xFF0,** 复位 **0x0000.000D**															
								CID0							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UARTPCellID1, 类型 **RO,** 偏移量 **0xFF4,** 复位 **0x0000.00F0**															
								CID1							
UARTPCellID2, 类型 **RO,** 偏移量 **0xFF8,** 复位 **0x0000.0005**															
								CID2							
UARTPCellID3, 类型 **RO,** 偏移量 **0xFFC,** 复位 **0x0000.00B1**															
								CID3							
同步串行接口 **(SSI)**															
SSICR0, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
SCR								SPH	SPO	FRF		DSS			
SSICR1, 类型 **R/W,** 偏移量 **0x004,** 复位 **0x0000.0000**															
												SOD	MS	SSE	LBM
SSIDR, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
DATA															
SSISR, 类型 **RO,** 偏移量 **0x00C,** 复位 **0x0000.0003**															
											BSY	RFF	RNE	TNF	TFE
SSICPSR, 类型 **R/W,** 偏移量 **0x010,** 复位 **0x0000.0000**															
								CPSDVSR							
SSIIM, 类型 **R/W,** 偏移量 **0x014,** 复位 **0x0000.0000**															
												TXIM	RXIM	RTIM	RORIM
SSIRIS, 类型 **RO,** 偏移量 **0x018,** 复位 **0x0000.0008**															
												TXRIS	RXRIS	RTRIS	RORRIS
SSIMIS, 类型 **RO,** 偏移量 **0x01C,** 复位 **0x0000.0000**															
												TXMIS	RXMIS	RTMIS	RORMIS
SSIICR, 类型 **W1C,** 偏移量 **0x020,** 复位 **0x0000.0000**															
														RTIC	RORIC
SSIPeriphID4, 类型 **RO,** 偏移量 **0xFD0,** 复位 **0x0000.0000**															
								PID4							
SSIPeriphID5, 类型 **RO,** 偏移量 **0xFD4,** 复位 **0x0000.0000**															
								PID5							
SSIPeriphID6, 类型 **RO,** 偏移量 **0xFD8,** 复位 **0x0000.0000**															
								PID6							
SSIPeriphID7, 类型 **RO,** 偏移量 **0xFDC,** 复位 **0x0000.0000**															
								PID7							
SSIPeriphID0, 类型 **RO,** 偏移量 **0xFE0,** 复位 **0x0000.0022**															
								PID0							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIPeriphID1, 类型 **RO,** 偏移量 **0xFE4,** 复位 **0x0000.0000**															
								PID1							
SSIPeriphID2, 类型 **RO,** 偏移量 **0xFE8,** 复位 **0x0000.0018**															
								PID2							
SSIPeriphID3, 类型 **RO,** 偏移量 **0xFEC,** 复位 **0x0000.0001**															
								PID3							
SSIPCellID0, 类型 **RO,** 偏移量 **0xFF0,** 复位 **0x0000.000D**															
								CID0							
SSIPCellID1, 类型 **RO,** 偏移量 **0xFF4,** 复位 **0x0000.00F0**															
								CID1							
SSIPCellID2, 类型 **RO,** 偏移量 **0xFF8,** 复位 **0x0000.0005**															
								CID2							
SSIPCellID3, 类型 **RO,** 偏移量 **0xFFC,** 复位 **0x0000.00B1**															
								CID3							
内部集成电路 **(I²C)** 接口															
I²C主机															
I2CMSA, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
								SA							R/S
I2CMCS, 类型 **RO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
I2CMCS, 类型 **WO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
												ACK	STOP	START	RUN
I2CMDR, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
								DATA							
I2CMTPR, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0x0000.0001**															
								TPR							
I2CMIMR, 类型 **R/W,** 偏移量 **0x010,** 复位 **0x0000.0000**															
															IM
I2CMRIS, 类型 **RO,** 偏移量 **0x014,** 复位 **0x0000.0000**															
															RIS
I2CMMIS, 类型 **RO,** 偏移量 **0x018,** 复位 **0x0000.0000**															
															MIS
I2CMICR, 类型 **WO,** 偏移量 **0x01C,** 复位 **0x0000.0000**															
															IC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2CMCR, 类型 **R/W,** 偏移量 **0x020,** 复位 **0x0000.0000**															
										SFE	MFE				LPBK
内部集成电路 **(I²C)** 接口 **I²C** 从机															
I2CSOAR, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
									OAR						
I2CSCSR, 类型 **RO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
													FBR	TREQ	RREQ
I2CSCSR, 类型 **WO,** 偏移量 **0x004,** 复位 **0x0000.0000**															
															DA
I2CSDR, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
								DATA							
I2CSIMR, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0x0000.0000**															
															IM
I2CSRIS, 类型 **RO,** 偏移量 **0x010,** 复位 **0x0000.0000**															
															RIS
I2CSMIS, 类型 **RO,** 偏移量 **0x014,** 复位 **0x0000.0000**															
															MIS
I2CSICR, 类型 **WO,** 偏移量 **0x018,** 复位 **0x0000.0000**															
															IC
控制器局域网 （**CAN**）模块															
CANCTL, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0001**															
								Test	CCE	DAR		EIE	SIE	IE	INIT
CANSTS, 类型 **R/W,** 偏移量 **0x004,** 复位 **0x0000.0000**															
								BOff	EWarn	EPass	RxOK	TxOK	LEC		
CANERR, 类型 **RO,** 偏移量 **0x008,** 复位 **0x0000.0000**															
RP	REC							TEC							
CANBIT, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0x0000.2301**															
	TSeg2			TSeg1				SJW		BRP					
CANINT, 类型 **RO,** 偏移量 **0x010,** 复位 **0x0000.0000**															
IntId															
CANTST, 类型 **R/W,** 偏移量 **0x014,** 复位 **0x0000.0000**															
								Rx	Tx		LBack	Silent	Basic		
CANBRPE, 类型 **R/W,** 偏移量 **0x018,** 复位 **0x0000.0000**															
												BRPE			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CANIF1CRQ, 类型 **RO,** 偏移量 **0x020,** 复位 **0x0000.0001**															
Busy										MNUM					
CANIF2CRQ, 类型 **RO,** 偏移量 **0x080,** 复位 **0x0000.0001**															
Busy										MNUM					
CANIF1CMSK, 类型 **RO,** 偏移量 **0x024,** 复位 **0x0000.0000**															
								WRNRD	Mask	Arb	Control	ClrIntPnd	TxRqstNewDat	DataA	DataB
CANIF2CMSK, 类型 **RO,** 偏移量 **0x084,** 复位 **0x0000.0000**															
								WRNRD	Mask	Arb	Control	ClrIntPnd	TxRqstNewDat	DataA	DataB
CANIF1MSK1, 类型 **RO,** 偏移量 **0x028,** 复位 **0x0000.FFFF**															
Msk															
CANIF2MSK1, 类型 **RO,** 偏移量 **0x088,** 复位 **0x0000.FFFF**															
Msk															
CANIF1MSK2, 类型 **RO,** 偏移量 **0x02C,** 复位 **0x0000.FFFF**															
MXtd	MDir		Msk												
CANIF2MSK2, 类型 **RO,** 偏移量 **0x08C,** 复位 **0x0000.FFFF**															
MXtd	MDir		Msk												
CANIF1ARB1, 类型 **RO,** 偏移量 **0x030,** 复位 **0x0000.0000**															
ID															
CANIF2ARB1, 类型 **RO,** 偏移量 **0x090,** 复位 **0x0000.0000**															
ID															
CANIF1ARB2, 类型 **RO,** 偏移量 **0x034,** 复位 **0x0000.0000**															
MsgVal	Xtd	Dir	ID												
CANIF2ARB2, 类型 **RO,** 偏移量 **0x094,** 复位 **0x0000.0000**															
MsgVal	Xtd	Dir	ID												
CANIF1MCTL, 类型 **RO,** 偏移量 **0x038,** 复位 **0x0000.0000**															
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB				DLC			
CANIF2MCTL, 类型 **RO,** 偏移量 **0x098,** 复位 **0x0000.0000**															
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB				DLC			
CANIF1DA1, 类型 **R/W,** 偏移量 **0x03C,** 复位 **0x0000.0000**															
Data															
CANIF1DA2, 类型 **R/W,** 偏移量 **0x040,** 复位 **0x0000.0000**															
Data															
CANIF1DB1, 类型 **R/W,** 偏移量 **0x044,** 复位 **0x0000.0000**															
Data															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CANIF1DB2, 类型 **R/W,** 偏移量 **0x048,** 复位 **0x0000.0000**															
Data															
CANIF2DA1, 类型 **R/W,** 偏移量 **0x09C,** 复位 **0x0000.0000**															
Data															
CANIF2DA2, 类型 **R/W,** 偏移量 **0x0A0,** 复位 **0x0000.0000**															
Data															
CANIF2DB1, 类型 **R/W,** 偏移量 **0x0A4,** 复位 **0x0000.0000**															
Data															
CANIF2DB2, 类型 **R/W,** 偏移量 **0x0A8,** 复位 **0x0000.0000**															
Data															
CANTXRQ1, 类型 **RO,** 偏移量 **0x100,** 复位 **0x0000.0000**															
TxRqst															
CANTXRQ2, 类型 **RO,** 偏移量 **0x104,** 复位 **0x0000.0000**															
TxRqst															
CANNWDA1, 类型 **RO,** 偏移量 **0x120,** 复位 **0x0000.0000**															
NewDat															
CANNWDA2, 类型 **RO,** 偏移量 **0x124,** 复位 **0x0000.0000**															
NewDat															
CANMSG1INT, 类型 **RO,** 偏移量 **0x140,** 复位 **0x0000.0000**															
IntPnd															
CANMSG2INT, 类型 **RO,** 偏移量 **0x144,** 复位 **0x0000.0000**															
IntPnd															
CANMSG1VAL, 类型 **RO,** 偏移量 **0x160,** 复位 **0x0000.0000**															
MsgVal															
CANMSG2VAL, 类型 **RO,** 偏移量 **0x164,** 复位 **0x0000.0000**															
MsgVal															
模拟比较器															
ACMIS, 类型 **R/W1C,** 偏移量 **0x00,** 复位 **0x0000.0000**															
													IN2	IN1	IN0
ACRIS, 类型 **RO,** 偏移量 **0x04,** 复位 **0x0000.0000**															
													IN2	IN1	IN0
ACINTEN, 类型 **R/W,** 偏移量 **0x08,** 复位 **0x0000.0000**															
													IN2	IN1	IN0
ACREFCTL, 类型 **R/W,** 偏移量 **0x10,** 复位 **0x0000.0000**															
						EN	RNG					VREF			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACSTAT0, 类型 **RO,** 偏移量 **0x20,** 复位 **0x0000.0000**															
														OVAL	
ACSTAT1, 类型 **RO,** 偏移量 **0x40,** 复位 **0x0000.0000**															
														OVAL	
ACSTAT2, 类型 **RO,** 偏移量 **0x60,** 复位 **0x0000.0000**															
														OVAL	
ACCTL0, 类型 **R/W,** 偏移量 **0x24,** 复位 **0x0000.0000**															
					ASRCP						ISLVAL	ISEN		CINV	
ACCTL1, 类型 **R/W,** 偏移量 **0x44,** 复位 **0x0000.0000**															
					ASRCP						ISLVAL	ISEN		CINV	
ACCTL2, 类型 **R/W,** 偏移量 **0x64,** 复位 **0x0000.0000**															
					ASRCP						ISLVAL	ISEN		CINV	
脉宽调制器 **(PWM)**															
PWMCTL, 类型 **R/W,** 偏移量 **0x000,** 复位 **0x0000.0000**															
													GlobalSync2	GlobalSync1	GlobalSync0
PWMSYNC, 类型 **R/W,** 偏移量 **0x004,** 复位 **0x0000.0000**															
													Sync2	Sync1	Sync0
PWMENABLE, 类型 **R/W,** 偏移量 **0x008,** 复位 **0x0000.0000**															
										PWM5En	PWM4En	PWM3En	PWM2En	PWM1En	PWM0En
PWMINVERT, 类型 **R/W,** 偏移量 **0x00C,** 复位 **0x0000.0000**															
										PWM5Inv	PWM4Inv	PWM3Inv	PWM2Inv	PWM1Inv	PWM0Inv
PWMFAULT, 类型 **R/W,** 偏移量 **0x010,** 复位 **0x0000.0000**															
										Fault5	Fault4	Fault3	Fault2	Fault1	Fault0
PWMINTEN, 类型 **R/W,** 偏移量 **0x014,** 复位 **0x0000.0000**															
															IntFault
													IntPWM2	IntPWM1	IntPWM0
PWMRIS, 类型 **RO,** 偏移量 **0x018,** 复位 **0x0000.0000**															
															IntFault
													IntPWM2	IntPWM1	IntPWM0
PWMISC, 类型 **R/W1C,** 偏移量 **0x01C,** 复位 **0x0000.0000**															
															IntFault
													IntPWM2	IntPWM1	IntPWM0
PWMSTATUS, 类型 **RO,** 偏移量 **0x020,** 复位 **0x0000.0000**															
															Fault
PWM0CTL, 类型 **R/W,** 偏移量 **0x040,** 复位 **0x0000.0000**															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM1CTL, 类型 **R/W,** 偏移量 **0x080,** 复位 **0x0000.0000**															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM2CTL, 类型 **R/W,** 偏移量 **0x0C0,** 复位 **0x0000.0000**															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM0INTEN, 类型 **R/W,** 偏移量 **0x044,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1INTEN, 类型 **R/W,** 偏移量 **0x084,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM2INTEN, 类型 **R/W,** 偏移量 **0x0C4,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0RIS, 类型 **RO,** 偏移量 **0x048,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1RIS, 类型 **RO,** 偏移量 **0x088,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM2RIS, 类型 **RO,** 偏移量 **0x0C8,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0ISC, 类型 **R/W1C,** 偏移量 **0x04C,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1ISC, 类型 **R/W1C,** 偏移量 **0x08C,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM2ISC, 类型 **R/W1C,** 偏移量 **0x0CC,** 复位 **0x0000.0000**															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0LOAD, 类型 **R/W,** 偏移量 **0x050,** 复位 **0x0000.0000**															
Load															
PWM1LOAD, 类型 **R/W,** 偏移量 **0x090,** 复位 **0x0000.0000**															
Load															
PWM2LOAD, 类型 **R/W,** 偏移量 **0x0D0,** 复位 **0x0000.0000**															
Load															
PWM0COUNT, 类型 **RO,** 偏移量 **0x054,** 复位 **0x0000.0000**															
Count															
PWM1COUNT, 类型 **RO,** 偏移量 **0x094,** 复位 **0x0000.0000**															
Count															
PWM2COUNT, 类型 **RO,** 偏移量 **0x0D4,** 复位 **0x0000.0000**															
Count															
PWM0CMPA, 类型 **R/W,** 偏移量 **0x058,** 复位 **0x0000.0000**															
CompA															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM1CMPA, 类型 **R/W,** 偏移量 **0x098,** 复位 **0x0000.0000**															
CompA															
PWM2CMPA, 类型 **R/W,** 偏移量 **0x0D8,** 复位 **0x0000.0000**															
CompA															
PWM0CMPB, 类型 **R/W,** 偏移量 **0x05C,** 复位 **0x0000.0000**															
CompB															
PWM1CMPB, 类型 **R/W,** 偏移量 **0x09C,** 复位 **0x0000.0000**															
CompB															
PWM2CMPB, 类型 **R/W,** 偏移量 **0x0DC,** 复位 **0x0000.0000**															
CompB															
PWM0GENA, 类型 **R/W,** 偏移量 **0x060,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM1GENA, 类型 **R/W,** 偏移量 **0x0A0,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM2GENA, 类型 **R/W,** 偏移量 **0x0E0,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM0GENB, 类型 **R/W,** 偏移量 **0x064,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM1GENB, 类型 **R/W,** 偏移量 **0x0A4,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM2GENB, 类型 **R/W,** 偏移量 **0x0E4,** 复位 **0x0000.0000**															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM0DBCTL, 类型 **R/W,** 偏移量 **0x068,** 复位 **0x0000.0000**															
															Enable
PWM1DBCTL, 类型 **R/W,** 偏移量 **0x0A8,** 复位 **0x0000.0000**															
															Enable
PWM2DBCTL, 类型 **R/W,** 偏移量 **0x0E8,** 复位 **0x0000.0000**															
															Enable
PWM0DBRISE, 类型 **R/W,** 偏移量 **0x06C,** 复位 **0x0000.0000**															
				RiseDelay											
PWM1DBRISE, 类型 **R/W,** 偏移量 **0x0AC,** 复位 **0x0000.0000**															
				RiseDelay											
PWM2DBRISE, 类型 **R/W,** 偏移量 **0x0EC,** 复位 **0x0000.0000**															
				RiseDelay											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM0DBFALL, 类型 **R/W**, 偏移量 **0x070**, 复位 **0x0000.0000**															
				FallDelay											
PWM1DBFALL, 类型 **R/W**, 偏移量 **0x0B0**, 复位 **0x0000.0000**															
				FallDelay											
PWM2DBFALL, 类型 **R/W**, 偏移量 **0x0F0**, 复位 **0x0000.0000**															
				FallDelay											
正交编码接口 **(QEI)**															
QEICTL, 类型 **R/W**, 偏移量 **0x000**, 复位 **0x0000.0000**															
			STALLEN	INVI	INVB	INVA	VelDiv			VelEn	ResMode	CapMode	SigMode	Swap	使能
QEISTAT, 类型 **RO**, 偏移量 **0x004**, 复位 **0x0000.0000**															
														方向	错误
QEIPOS, 类型 **R/W**, 偏移量 **0x008**, 复位 **0x0000.0000**															
位置															
位置															
QEIMAXPOS, 类型 **R/W**, 偏移量 **0x00C**, 复位 **0x0000.0000**															
MaxPos															
MaxPos															
QEILOAD, 类型 **R/W**, 偏移量 **0x010**, 复位 **0x0000.0000**															
装载															
装载															
QEITIME, 类型 **RO**, 偏移量 **0x014**, 复位 **0x0000.0000**															
定时															
定时															
QEICOUNT, 类型 **RO**, 偏移量 **0x018**, 复位 **0x0000.0000**															
计数															
计数															
QEISPEED, 类型 **RO**, 偏移量 **0x01C**, 复位 **0x0000.0000**															
速度															
速度															
QEIINTEN, 类型 **R/W**, 偏移量 **0x020**, 复位 **0x0000.0000**															
												IntError	IntDir	IntTimer	IntIndex
QEIRIS, 类型 **RO**, 偏移量 **0x024**, 复位 **0x0000.0000**															
												IntError	IntDir	IntTimer	IntIndex
QEIISC, 类型 **R/W1C**, 偏移量 **0x028**, 复位 **0x0000.0000**															
												IntError	IntDir	IntTimer	IntIndex

C 订购和联系信息

C.1 订购信息

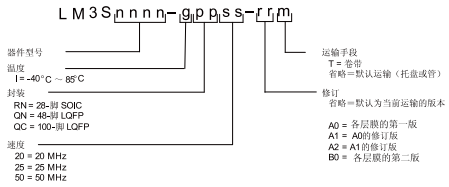


表 **C-1.** 器件订购信息

可订购的器件型号	描述
LM3S2950-IQC50	Stellaris® LM3S2950 微控制器
LM3S2950-IQC50(T)	Stellaris® LM3S2950 微控制器

C.2 套件

Luminary Micro Stellaris®系列提供了工程师需要快速开发的硬件和软件工具。

- 参考设计套件通过提供可以运行的硬件和完整的文档来加速产品开发，这些完整的文档包含了硬件设计文件：

  http://www.luminarymicro.com/products/reference_design_kits/

- 在购买之前，开发套件提供了评估Stellaris®微控制器的低成本和有效方式：

  http://www.luminarymicro.com/products/evaluation_kits/

- 开发套件为你提供所有需要开发的工具和原型嵌入式应用:

  http://www.luminarymicro.com/products/boards.html

最近可供货的工具请见 Luminary Micro 网站或咨询您的 Luminary Micro 分销商。

C.3 公司信息

Luminary Micro公司设计、推广和销售基于ARM Cortex-M3的微控制器（MCU）。位于Austin Texas（奥斯汀德克萨斯）的Luminary Micro公司是Cortex-M3的主要合作伙伴，生产了世界上首款实现Cortex-M3处理器的芯片。Luminary Micro引进的Stellaris®系列的器件提供32-位的性能，价格与目前8位和16位微控制器设计的价格相同。一个基于ARM技术的MCU的入门级价格为1.00美元，Luminary Micro的Stellaris产品线实现了标准化，无需进行未来结构体系的升级或软件工具的修改。

Luminary Micro公司108 Wild Basin, Suite 350 Austin, TX 78746 主要联系电话：+1-512-279-8800。

C.4 支持信息

如需获得有关Luminary Micro产品的支持，请联系：

support@luminarymicro.com +1-512-279-8800, 分机号码3

D 周立功公司相关信息

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 12 楼 F4　邮编：510630

电话：(020)38730972　38730976　38730916　38730917　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　87569917

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 A 座 1207-1208 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）1611 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：成都市一环路南二段 1 号数码同人港 401 室（磨子桥立交西北角）

电话：(028) 85439836　85437446

传真：(028) 85437896

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)83781788（5 线）

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　87881295

传真：(029)87880865