LUMINARY MICRO™

LM3S6100 微控制器

数据手册

DS-LM3S6100-03

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

本数据手册提供了LM3S6100微控制器的相关信息，描述了围绕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 16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 33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。二进制值用一个后缀b来指示，例如，1011b。十进制值既无前缀，也无后缀。

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。 Stellaris® LM3S2000系列是针对CAN应用方案而设计的一组芯片，它在群星系列芯片的基础上扩展了Bocsh CAN网络技术——短距离工业网络里的黄金标准。 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。 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），它不但标志着ARM Cortex-M3微控制器已经具备了集成连接能力，还是唯一一系列同时集成了10/100以太网MAC和PHY物理层的ARM架构MCU。

LM3S6100 微控制器是针对工业应用方案而设计的，这些应用方案包括远程监控、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、以及火警安防等。

除此之外，LM3S6100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，LM3S6100微处理器与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

LM3S6100微控制器包含了下列特性

- 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25-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
  - 20 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
  - 64 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
  - 16 KB单周期访问的SRAM
- 通用定时器
  - 3个 通用定时器模块（GPTM），每个模块都能提供2个16位的定时器/计数器。 每个通用定时器模块都能够被设置为独立运作的定时器或事件计数器（总共有8个）可用作单个32位的定时器（最多4个）或者用作单个32位的实时时钟（RTC）以捕获事件， 或者 用作脉宽调制输出（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或单次触发模式
  - 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

■ 10/100以太网控制器

- 符合 IEEE 802.3-2002规范
- 在100 Mbps和10 Mbps速率运作下支持全双工和半双工的运作方式
- 集成10/100 Mbps收发器（PHY物理层）
- 自动MDI/MDI-X交叉校验
- 可编程MAC地址
- 节能和断电模式

■ 同步串行接口（SSI）

- 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

- 完全可编程控制的16C550型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、¼、½、¾、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

- 1个集成的模拟比较器
- 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

- GPIO
  - 高达10-30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
  - 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

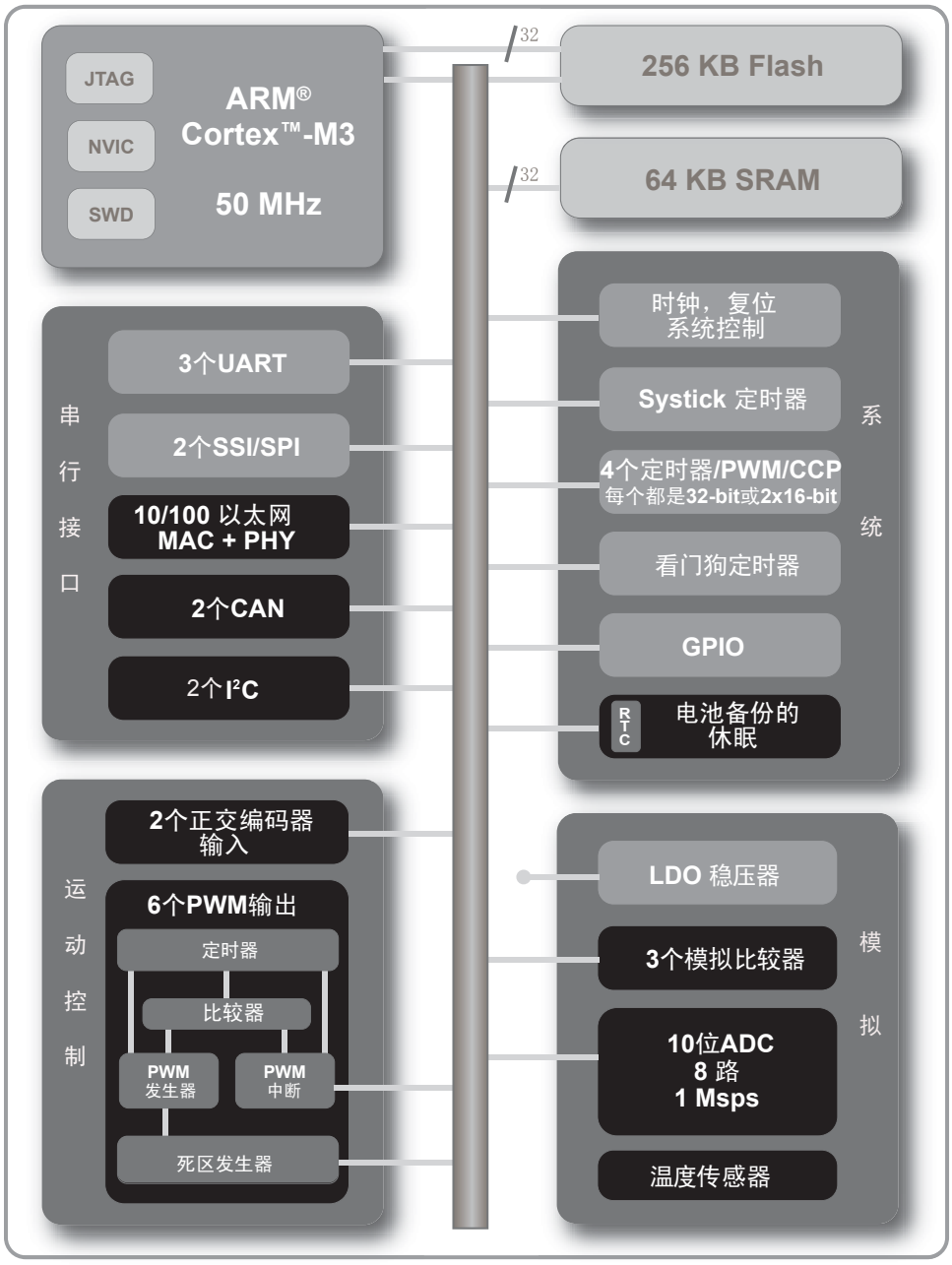
1.2 目标应用

- 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 23页给出了Stellaris® Fury-class系列器件的特性。

图 **1-1. Stellaris® Fury-class**高级方框图



1.4 功能概述

下面的小节给出了LM3S6100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 364页中找到。

1.4.1 ARM Cortex™-M3

1.4.1.1 处理器内核（见28页）

Stellaris®系列产品的所有成员（包括LM3S6100微处理器）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 28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**）

LM3S6100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20个中断上设置8个优先级。

“中断” 在 35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，LM3S6100控制器包含脉宽调制（PWM）输出。

1.4.2.1 **PWM**（见**161**页）

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LM3S6100上，通过通用定时器的运动控制特性（使用CCP管脚）可以成功地实现PWM动作控制功能。

CCP管脚（见161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。

1.4.3 串行通信外设

LM3S6100控制器支持异步和同步串行通信:

- 1个完全可编程的16C550型UART
- 1个SSI模块

1.4.3.1 UART（见209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。

LM3S6100控制器包含1个完全可编程的16C550型UART，支持高达460.8Kbps的数据传输速率。 另外，每个UART都能支持IrDA功能。 （尽管16C550型UART在功能上与16C550型UART类似，但它们的寄存器不兼容）。

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。

1.4.3.2 SSI（见246页）

同步串行接口（SSI）是一个4线双向的通信接口。

LM3S6100控制器包含1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。

这个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 。

这个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。

这个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。

1.4.3.3 以太网控制器（见279页）

以太网是一种基于帧的局域网（LAN）计算机网络技术。 以太网已经标准化为 IEEE 802.3标准。 它定义了大量的物理层连线和信号标准、媒体访问控制（MAC）/数据链路层的两种网络访问方式和一种通用的寻址格式。

Stellaris®以太网控制器由一个完全集成的媒体访问控制器（MAC）和网络物理层接口（PHY）器件组成。 以太网控制器完全符合IEEE 802.3规范并完全支持10BASE-T以及100BASE-TX标准。 除此之外，以太网控制器还支持自动MDI/MDI-X交叉校验。

1.4.4 系统外设

1.4.4.1 可编程的GPIO（见118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7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 10-30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 334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。

1.4.4.2 3个可编程的定时器（见156页）

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者定时。

Stellaris®通用定时器模块（GPTM）包含3个GPTM块。 每个GPTM块提供了2个16位的定时/计数器，可配置为两个独立的定时器或者事件计数器，还可以配置为一个32位的定时器或32位的实时时钟（RTC）。

当配置成32位模式的时候，定时器可作为单次触发定时器、周期定时器或实时时钟（RTC）运行。当配置成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位的定时器还可以被配置为事件捕获或者脉宽调制（PWM）功能。

1.4.4.3 看门狗定时器（见186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。

1.4.5 存储器外设

LM3S6100存储器提供了SRAM和Flash存储器。

1.4.5.1 SRAM（见94页）

LM3S6100静态随机存取存储器（SRAM） 控制器支持16 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。

1.4.5.2 Flash存储器（见95页）

LM3S6100Flash控制器支持64KB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。

1.4.6 其他特性

1.4.6.1 存储器映射（见33页）

存储器映射列出了指令和数据在存储器中所处的位置。LM3S6100控制器的存储器映射可以在"存储器映射" 在 33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*。

1.4.6.2 JTAG TAP控制器（见37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。

1.4.6.3 系统控制和时钟（见47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 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 333页
- “信号表” 在 334页
- “工作特性” 在 345页
- “电气特性” 在 346页
- “封装信息” 在 358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：

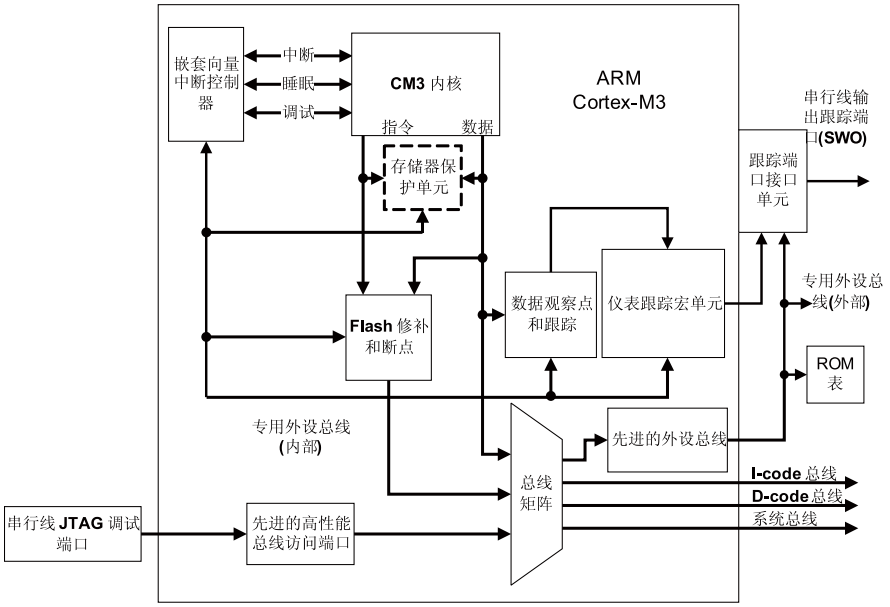
- 紧凑的内核
- Thumb-2指令集，在通常与8位和16位设备相关的存储容量中，特别是在微控制器级应用的几千字节存储量中，提供ARM内核所期望的高性能。
- 高速的应用通过Harvard结构执行，以独立的指令和数据总线为特征。
- 优越的中断处理能力，通过执行寄存器操作来实现，这些寄存器操作在处理硬件中断时使用。
- 存储器保护单元（MPU）为复杂的应用提供特权操作模式。
- 从 ARM7™ 处理器系列中移植过来，以获得更好的性能和电源效率。
- 功能齐全的调试解决方案：
  - 串行线JTAG调试端口 （SWJ-DP）
  - Flash 修补和断点（FPB）单元，用于实现断点操作
  - 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 29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。

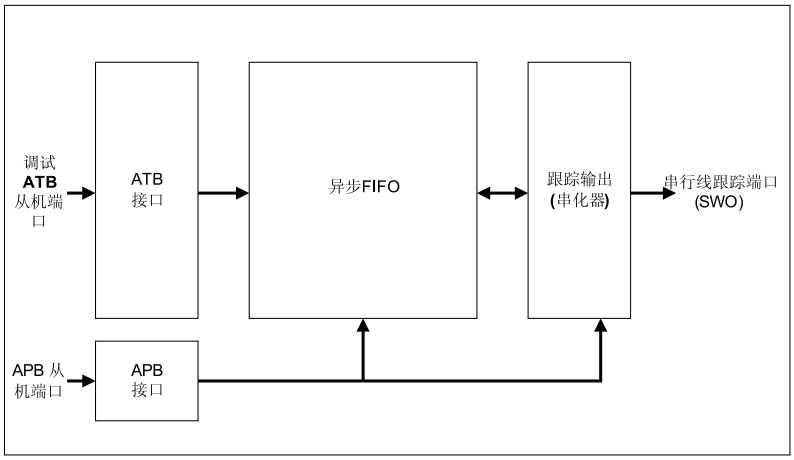
2.2.2 嵌入式跟踪宏单元 （ETM）

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 30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）

LM3S6100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。该LM3S6100微控制器支持 20 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	R/W	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0x00FFFFFF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0x00FFFFFF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

LM3S6100 控制器的存储器映射在 表 3-1 在 33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”。

注意: 在 表 3-1 在 33页中没列出的地址被保留。

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	0x1FFF.FFFF	片内Flash[b]	98
0x2000.0000	0x200F.FFFF	片内带 Bit-banded特性的 SRAM[c]	98
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。	94
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	188
0x4000.1000	0x4000.3FFF	保留	-
0x4000.4000	0x4000.4FFF	GPIO 端口 A	123
0x4000.5000	0x4000.5FFF	GPIO 端口 B	123
0x4000.6000	0x4000.6FFF	GPIO 端口 C	123
0x4000.7000	0x4000.7FFF	GPIO 端口 D	123
0x4000.8000	0x4000.8FFF	SSI0	256
0x4000.A000	0x4000.BFFF	保留	-
0x4000.C000	0x4000.CFFF	UART0	215
0x4000.F000	0x4000.FFFF	保留	-
0x4001.0000	0x4001.FFFF	保留用于将来的 FiRM 外设。	-
外设			
0x4002.2000	0x4002.3FFF	保留	-
0x4002.4000	0x4002.4FFF	GPIO 端口 E	123
0x4002.5000	0x4002.5FFF	GPIO 端口 F	123
0x4002.6000	0x4002.6FFF	GPIO 端口 G	123
0x4002.9000	0x4002.BFFF	保留	-
0x4002.E000	0x4002.FFFF	保留	-
0x4003.0000	0x4003.0FFF	定时器 0	166
0x4003.1000	0x4003.1FFF	定时器 1	166
0x4003.2000	0x4003.2FFF	定时器 2	166
0x4003.4000	0x4003.7FFF	保留	-
0x4003.9000	0x4003.BFFF	保留	-
0x4003.C000	0x4003.CFFF	模拟比较器	322
0x4003.D000	0x4003.FFFF	保留	-

起始	结束	描述	有关寄存器的详细内容，请见页面。
0x4004.3000	0x4004.7FFF	保留	-
0x4004.8000	0x4004.8FFF	以太网控制器	286
0x4004.9000	0x4004.BFFF	保留	-
0x4004.C000	0x400F.BFFF	保留	-
0x400F.D000	0x400F.DFFF	Flash 控制	98
0x400F.E000	0x400F.EFFF	系统控制	53
0x400F.F000	0x400F.FFFF	保留	-
0x4011.1000	0x4011.1FFF	保留	-
0x4012.0000	0x41FF.FFFF	保留用于不带 bit-banded 特性的外设空间。	-
0x4200.0000	0x43FF.FFFF	0x4000.0000到0x400F.FFFF的bit-band 别名。	-
0x4400.0000	0x5E32.FFFF	保留用于不带 bit-banded 特性的外设空间。	-
0x5E34.0000	0x5FFF.FFFF	保留	-
0x6000.0000	0xDFFF.FFFF	保留用于外部的器件。	-
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 35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20个中断上设置8个优先级（在 表 4-2 在 36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 36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 36页列出了 LM3S6100 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
7	SSI0
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
28	系统控制
29	Flash 控制
30	GPIO 端口 F
31	GPIO 端口 G
42	以太网控制器
44-47	保留

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。多路复用器将由Luminary Micro JTAG控制器控制，它可以对ARM、LMI和未执行的JTAG指令进行综合编程。

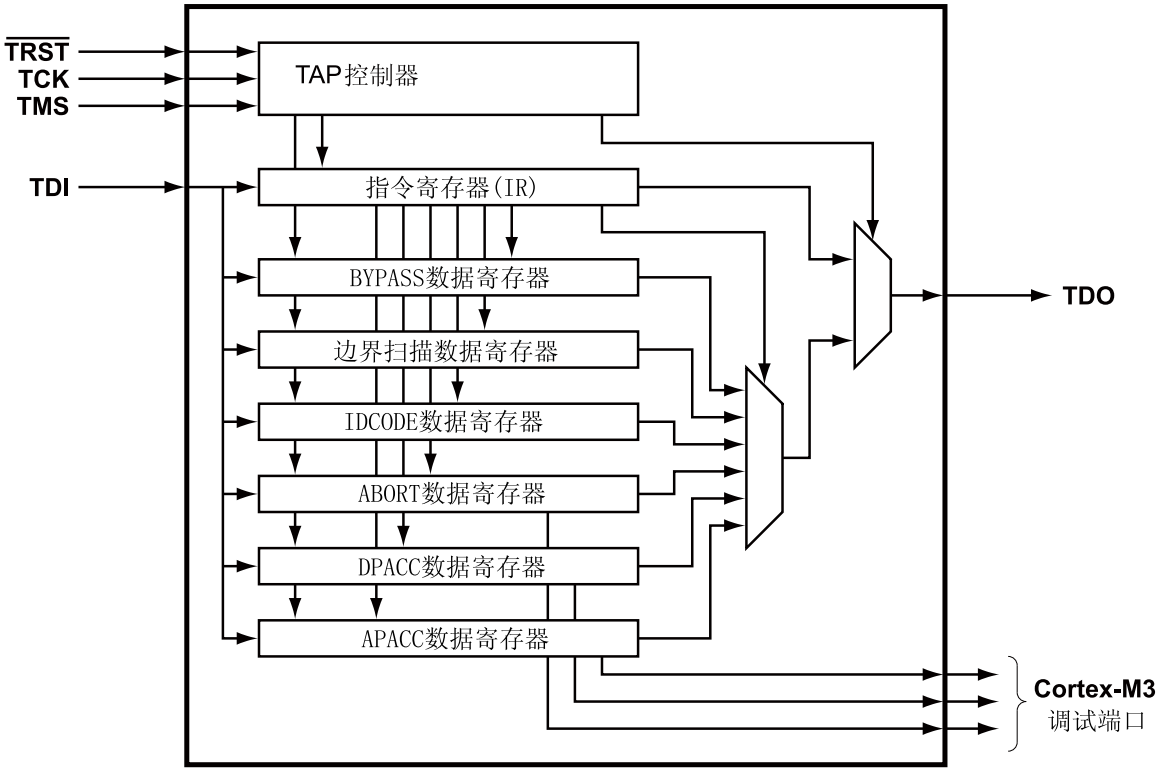
JTAG模块具有以下特性：

- 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 38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 43页）。

要了解JTAG的时序图，请参考“JTAG和边界扫描” 在 354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 39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 40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）

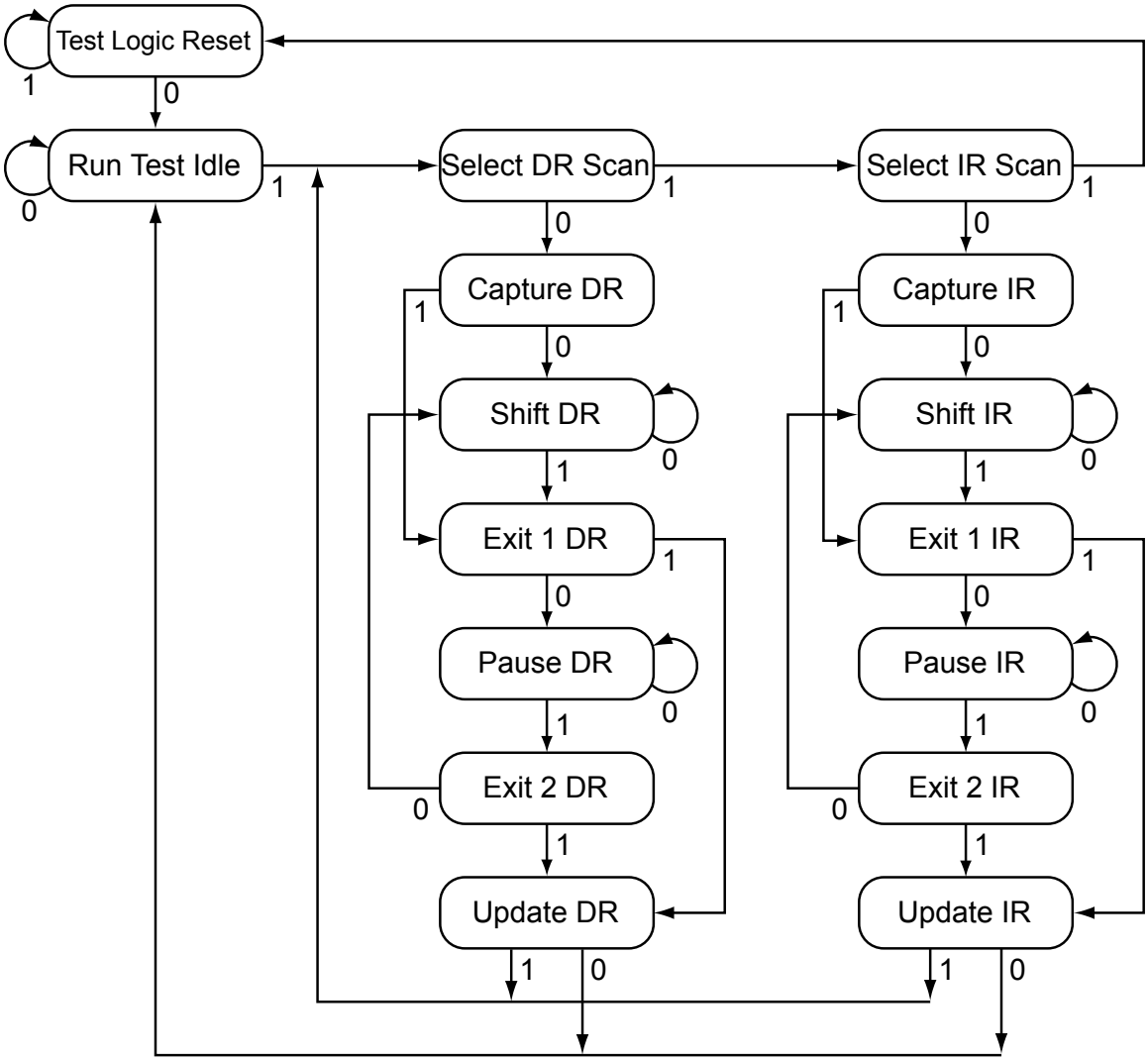
TDO管脚将一串串行信息从IR链或DR链输出。TDO的值取决于当前的TAP状态、当前指令、以及正在访问的数据链中的数据。为了在不使用JTAG端口时节省功耗，若当前并没有移出数据的活动，TDO管脚将被放置于停止的驱动状态中。因为在菊链配置中，TDO可以与另一个控制器的TDI管脚相连，所以*IEEE 标准1149.1*期望TDO的值在TCK的下降沿发生变化。

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 40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 43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 端口进行调试或板级测试，那么可腾出5个可供使用的GPIO端口。

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。

确认控制寄存器提供一个保护层，以防止对重要的硬件外设进行意外编程。对 **GPIO** 备用功能选择 **(GPIOAFSEL)** 寄存器（见 133页）保护位的写操作没有被确认进行存储直至 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 142页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 143页）的相应位已设为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 42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 43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 45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 46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 46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 45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 44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 45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。

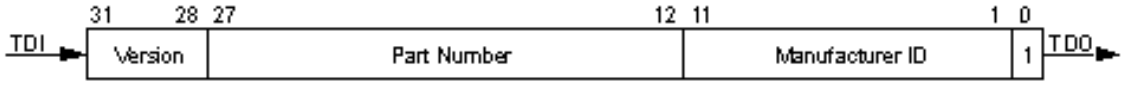
5.4.2.1 IDCODE 数据寄存器

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 45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。

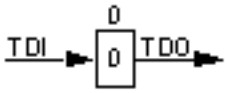
图 **5-3. IDCODE** 寄存器格式



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 45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

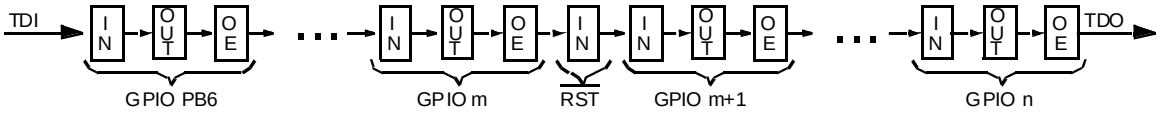


5.4.2.3 边界扫描数据寄存器

边界扫描数据寄存器的格式如图 5-5 在 45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 47页
- 局部控制，例如复位（“复位控制” 在 47页）、功率（见“功率控制” 在 49页）及时钟控制（见“时钟控制” 在 49页）
- 系统控制（运行、睡眠和深度睡眠模式），见“系统控制” 在 51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 47页。
2. 上电复位（POR），见“上电复位（POR）” 在 48页。
3. 内部掉电（BOR）检测器，见“掉电复位（BOR）” 在 48页。
4. 软件启动的复位（利用软件复位寄存器），见“软件复位” 在 49页。
5. 违反看门狗定时器复位条件，见“看门狗定时器复位” 在 49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 37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。

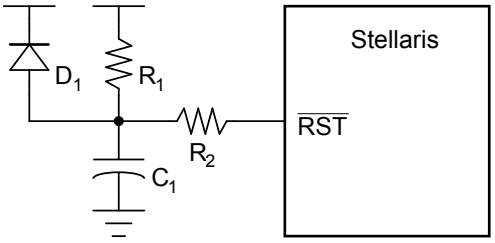
外部复位时序如图 18-9 在 357页所示。

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 48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 18-10 在 357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 18-11 在 357页所示。

6.1.2.6 软件复位

软件可以产生一个整个系统的复位，也可以复位某个特定的外设。

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see “系统控制” 在 51页）。 向一个位通道写入1来启动相应单元的复位。 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 18-12 在 357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 18-13 在 357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。支持的晶体在表 6-3 在 63页中列出。

- 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。

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～8.192 MHz 的范围之内（3.579545 MHz和8.192 MHz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 主振荡器器（**MOSC**）的晶体配置

主振荡器支持从1 MHz ～ 8.192 MHz的范围内选择晶体值。这个方法允许Luminary Micro提供优良的PLL设置。

表 6-3 在 63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 65页）。内部转换提供的转换在目标PLL VCO频率的± 1%以内。

表 6-3 在 63页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 62页和 66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 18-6 在 349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。

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**RCC/RCC2**的PWRDN位。 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 52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	54
0x004	DID1	RO	-	器件标识1	69
0x008	DC0	RO	0x003F.001F	器件功能0	71
0x010	DC1	RO	0x0000.709F	器件功能1	72
0x014	DC2	RO	0x0107.0011	器件功能2	73
0x018	DC3	RO	0x0F00.01C0	器件功能3	74
0x01C	DC4	RO	0x5000.007F	器件功能4	75
0x030	PBORCTL	R/W	0x0000.7FFD	掉电复位控制	56
0x034	LDOPCTL	R/W	0x0000.0000	LDO功率控制	57
0x040	SRCR0	RW	0x00000000	软件复位控制0	91
0x044	SRCR1	RW	0x00000000	软件复位控制1	92
0x048	SRCR2	RW	0x00000000	软件复位控制2	93
0x050	RIS	RO	0x0000.0000	原始中断状态	58
0x054	IMC	R/W	0x0000.0000	中断屏蔽控制	59
0x058	MISC	R/W1C	0x0000.0000	屏蔽后中断的状态和清零	60
0x05C	RESC	R/W	-	复位原因	61
0x060	RCC	R/W	0x07A0.3AD1	运行模式时钟配置	62
0x064	PLLCFG	RO	-	XTAL到PLL转换	65

偏移量	名称	类型	复位	描述	见页面
0x070	RCC2	R/W	0x0780.2800	运行模式时钟配置2	66
0x100	RCGC0	RW	0x00000040	运行模式时钟选通控制寄存器0	76
0x104	RCGC1	RW	0x00000000	运行模式时钟选通控制寄存器1	79
0x108	RCGC2	RW	0x00000000	运行模式时钟选通控制寄存器2	85
0x110	SCGC0	RW	0x00000040	睡眠模式时钟选通控制寄存器0	77
0x114	SCGC1	RW	0x00000000	睡眠模式时钟选通控制寄存器1	81
0x118	SCGC2	RW	0x00000000	睡眠模式时钟选通控制寄存器2	87
0x120	DCGC0	RW	0x00000040	深度睡眠模式时钟选通控制寄存器0	78
0x124	DCGC1	RW	0x00000000	深度睡眠模式时钟选通控制寄存器1	83
0x128	DCGC2	RW	0x00000000	深度睡眠模式时钟选通控制寄存器2	89
0x144	DSLPCLKCFG	R/W	0x0780.0000	深度睡眠时钟配置	68

6.4 寄存器描述

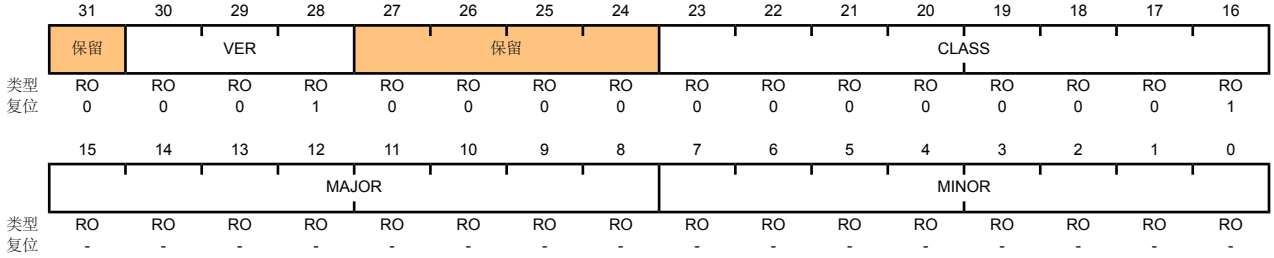
所有给出的地址都是相对于0x400F.E000的系统控制基址而言的。

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	1	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 1 Stellaris® Fury-class器件的第一版**DID0**寄存器格式。
27:24	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。
23:16	CLASS	RO	1	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 0 Stellaris® Sandstorm-class器件。 1 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 0 版本A（原始器件） 1 版本B（第一个基础层版本） 2 版本C（第二个基础层版本） 依此类推。

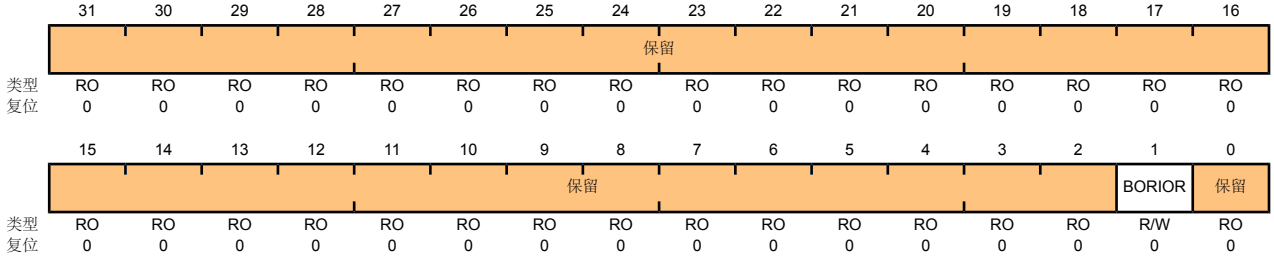
位/域	名称	类型	复位	描述
7:0	MINOR	RO	-	次版本 这个字段指定了器件的次版本号。 次版本反映了设计的金属层的改变。 MAJOR域改变时MINOR域的值重新设置。 这个域的值是数字，编码如下: 值 描述 0 原始器件，或者一个主版本更新。 1 第一个金属层改变。 2 第二个金属层改变。 依此类推。

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



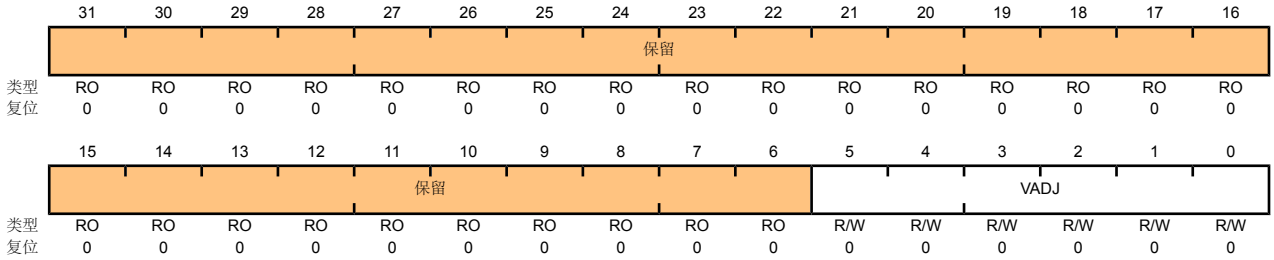
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORIOR	R/W	0	BOR中断或复位 该位控制如何将BOR事件发送给控制器。如果该位为1，发出复位信号。否则发出中断。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

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域的编程值在表 6-2 在 57页中提供。

表 **6-2. VADJ～VOUT**

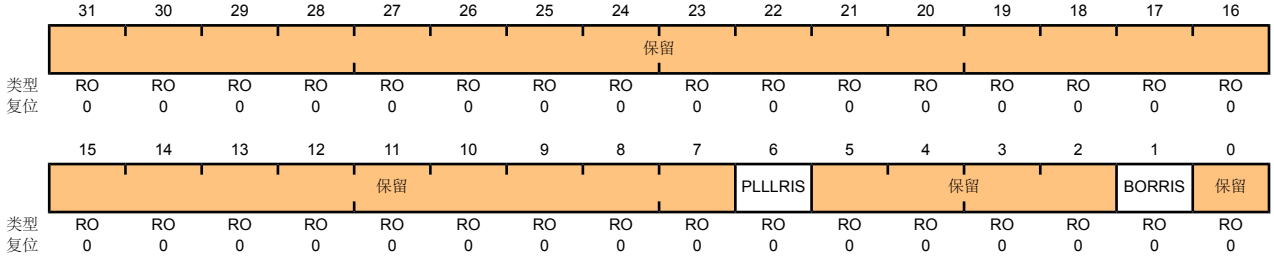
VADJ值	V_{OUT} (V)	VADJ值	V_{OUT} (V)	VADJ值	V_{OUT} (V)
0x1B	2.75	0x1F	2.55	0x03	2.35
0x1C	2.70	0x00	2.50	0x04	2.30
0x1D	2.65	0x01	2.45	0x05	2.25
0x1E	2.60	0x02	2.40	0x06-0x3F	保留

寄存器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



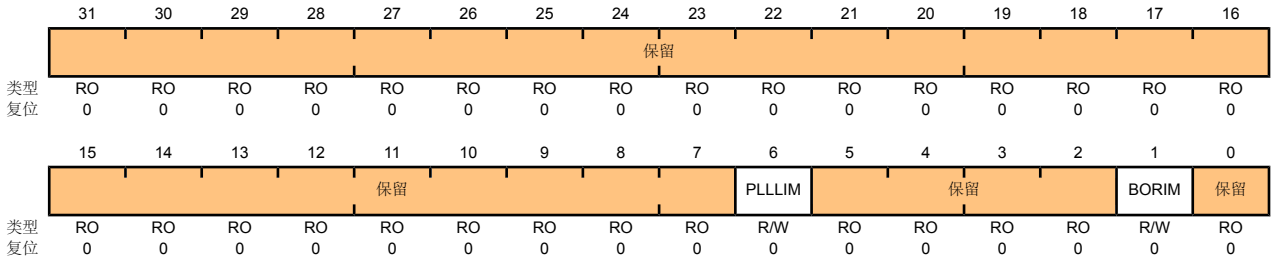
位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLRIS	RO	0	PLL锁定原始中断状态 该位在PLL T_{READY}定时器有效时置位。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORRIS	RO	0	掉电复位原始中断状态 该位是所有掉电条件的原始中断状态。 如果该位置位，则掉电条件目前有效。 这是一个来自掉电检测电路的未注册信号（unregistered signal）。 如果**IMC**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



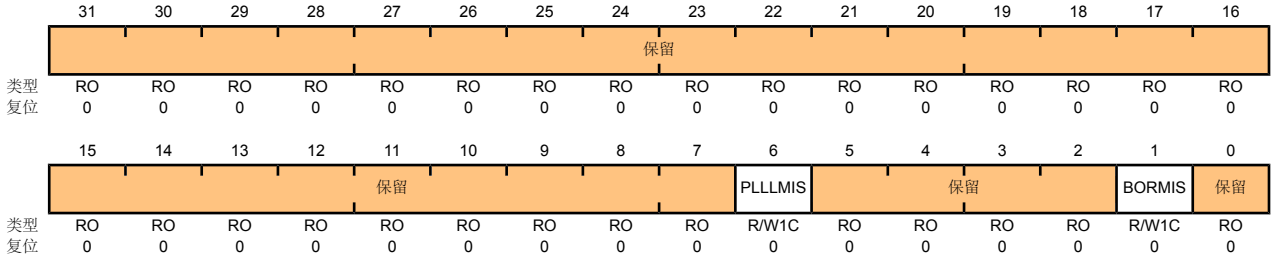
位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLIM	R/W	0	PLL锁定的中断屏蔽 该位指定限流检测是否引发一个控制器中断。 当该位置位时，如果**RIS**的PLLLRIS位置位，则产生中断；否则不产生中断。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORIM	R/W	0	掉电复位的中断屏蔽 该位指定掉电条件是否引发一个控制器中断。 当该位置位时，如果BORRIS置位，则产生中断；否则不产生中断。
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 58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



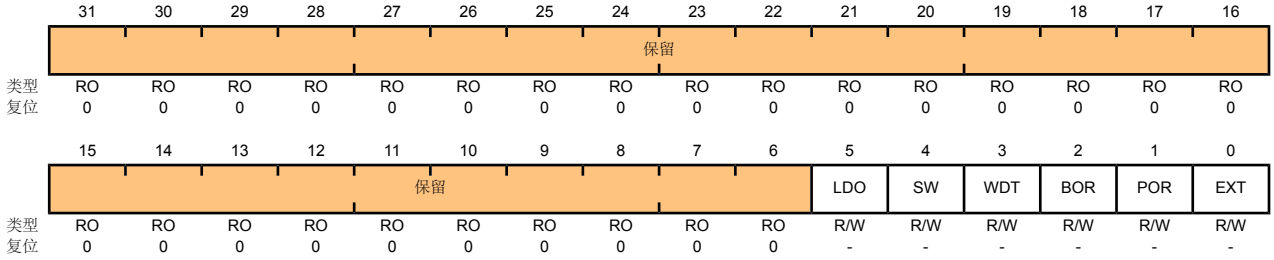
位/域	名称	类型	复位	描述
31:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PLLLMIS	R/W1C	0	PLL锁定屏蔽后的中断状态 该位在PLL T_{READY}定时器有效时置位。 通过向该位写入1来清除中断。
5:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	BORMIS	R/W1C	0	BOR屏蔽后的中断状态 BORMIS只是BORRIS和屏蔽值BORIM相与的结果。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器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



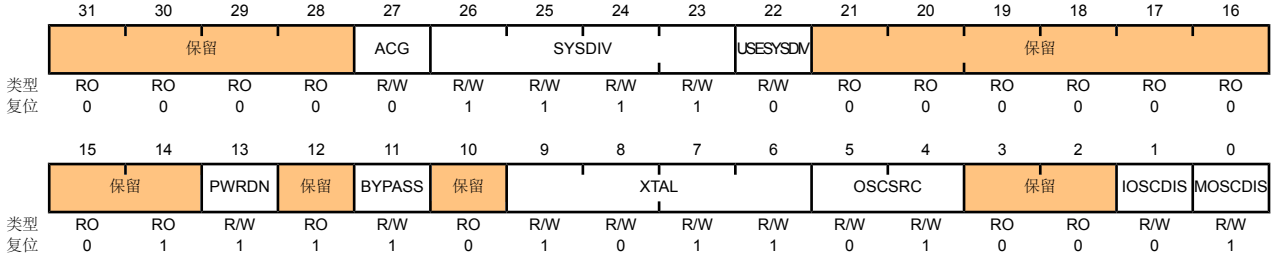
位/域	名称	类型	复位	描述
31:6	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5	LDO	R/W	-	LDO复位 该位置位时表明复位事件是LDO电路不可调引起的。
4	SW	R/W	-	软件复位 该位置位时表明复位事件是软件复位。。
3	WDT	R/W	-	看门狗定时器复位 该位置位时表明复位事件是看门狗复位。
2	BOR	R/W	-	掉电复位 该位置位时表明复位事件是掉电复位。
1	POR	R/W	-	上电复位 该位置位时表明复位事件是上电复位。
0	EXT	R/W	-	外部复位 该位置位时表明复位事件是一个外部复位（RST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0x07A0.3AD1



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。
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。 二进制值 分频值（BYPASS=1） 频率（BYPASS=0） 0000-0110 保留 保留 0011 /8 50 MHz 0111 /16 25 MHz 1000 /18 22.22 MHz 1001 /20 20 MHz 1010 /22 18.18 MHz 1011 /24 16.67 MHz 1100 /26 15.38 MHz 1101 /28 14.29 MHz 1110 /30 13.33 MHz 1111 /32 12.5 MHz（默认） 当读运行模式时钟配置（**RCC**）寄存器（见 62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。

位/域	名称	类型	复位	描述
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。
21:14	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。
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。
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。 这个域的编码如表6-3 在63页所示。
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 00 主振荡器（默认） 01 内部振荡器（默认） 10 内部振荡器/ 4（如果用作PLL的输入，则这是必须的） 11 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）。

表 **6-3.** 默认的晶体域值和**PLL**设置

晶体编号（**XTAL**二进制值）	不使用**PLL**时的晶体频率（**MHz**）	使用**PLL**时的晶体频率（**MHz**）
0000	1.000	保留
0001	1.8432	保留

晶体编号（**XTAL**二进制值）	不使用**PLL**时的晶体频率（**MHz**）	使用**PLL**时的晶体频率（**MHz**）
0010	2.000	保留
0011	2.4576	保留
0100	3.579545 MHz	
0101	3.6864 MHz	
0110	4 MHz	
0111	4.096 MHz	
1000	4.9152 MHz	
1001	5 MHz	
1010	5.12 MHz	
1011	6 MHz（复位值）	
1100	6.144 MHz	
1101	7.3728 MHz	
1110	8 MHz	
1111	8.192 MHz	

寄存器9: XTAL到PLL转换（PLLCFG），偏移量 0x064

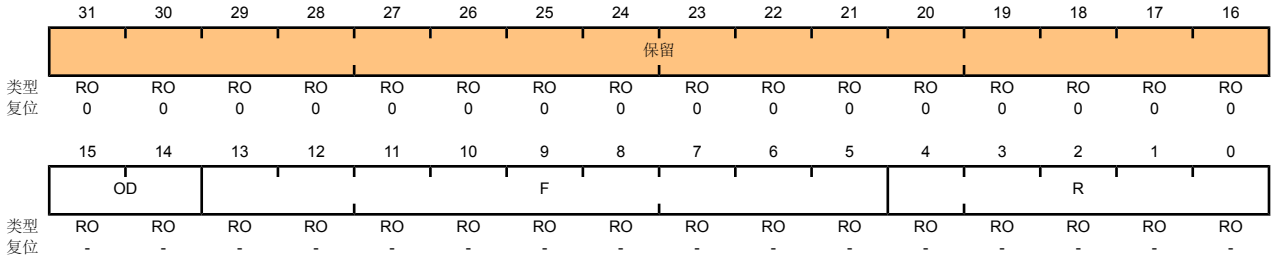
该寄存器提供将外部晶振频率转换为适当的PLL设置的方法。它在复位序列的执行过程中被初始化，并且，一旦运行模式时钟配置（**RCC**）寄存器的 XTAL域的值改变，寄存器的值就随之更新（见 62页）。

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
31:16	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。
15:14	OD	RO	-	PLL OD的值 这个域指定提供给PLL OD输入的值。
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。

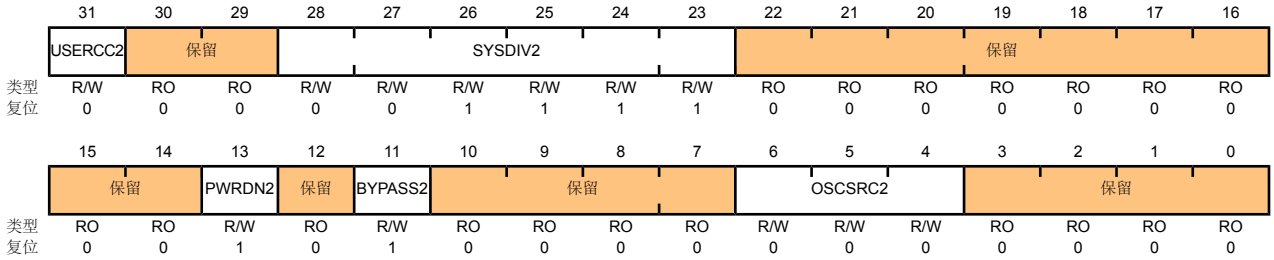
寄存器**10:** 运行模式时钟配置**2**（**RCC2**），偏移量 **0x070**

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	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。
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 111时提供的分频值是/16，而**RCC2**寄存器的SYSDIV2 域的编码为111111时提高的分频值是/64。
22:14	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	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	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。

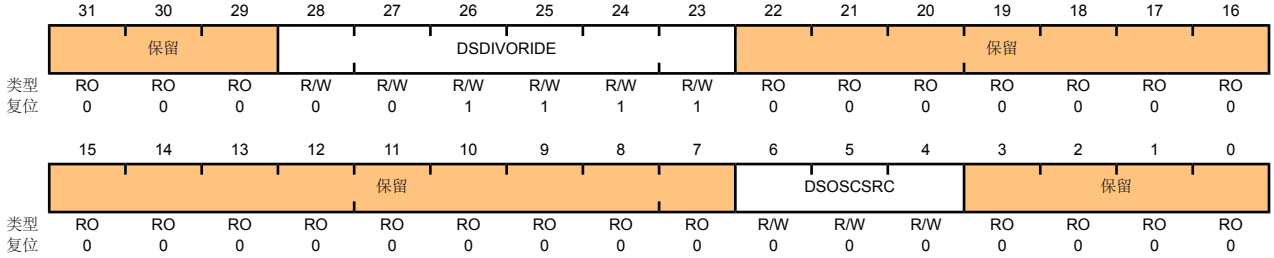
位/域	名称	类型	复位	描述
6:4	OSCSRC2	R/W	0	系统时钟源 名称 值 描述 MOSC 0 主振荡器 IOSC 1 内部振荡器 IOSC/4 2 内部振荡器 / 4 30kHz 3 30 kHz的内部振荡器 32kHz 7 32 kHz的外部振荡器
3:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

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

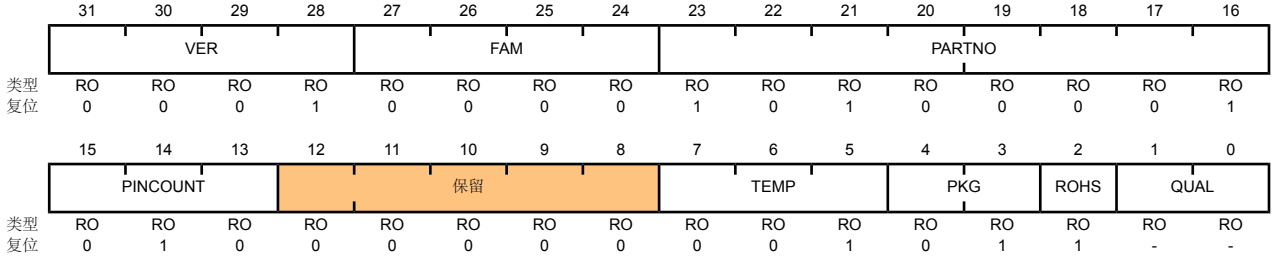


位/域	名称	类型	复位	描述
31:29	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
28:23	DSDIVORIDE	R/W	0x0F	分频器域替换 当PLL运行时出现深度睡眠，6位的系统分频器域替换。
22:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6:4	DSOSCSRC	R/W	0	时钟源 该位置位时强制IOSC在深度睡眠模式中用作时钟源。 名称 值 描述 NOORIDE 0 不替换振荡器时钟源。 IOSC 1 使用内部12MHz的振荡器作为时钟源 30kHz 3 使用30kHz的内部振荡器 32kHz 7 使用32kHz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， 以及封装类型。

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 这个位域决定 **DID1** 寄存器格式的版本。版本号是数字的。 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	0xA1	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 0xA1 LM3S6100
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。

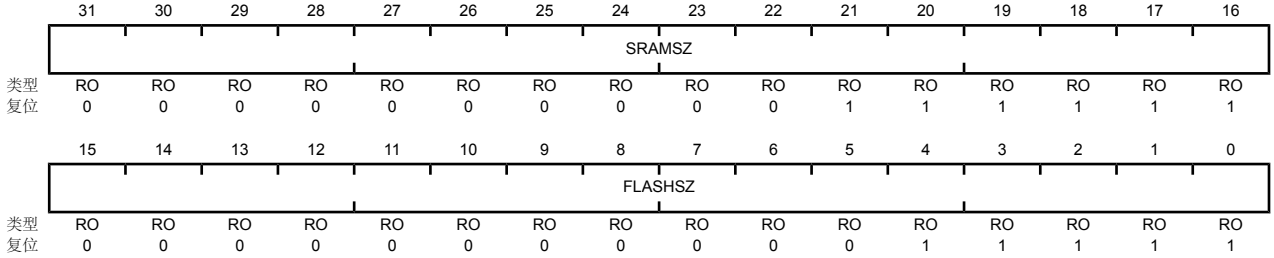
位/域	名称	类型	复位	描述
7:5	TEMP	RO	0x1	温度范围 这个位域规定了器件的温度范围。该值编码如下（所有其它编码保留）： 值 描述值 0x1 工业温度范围 (-40°C ～ 85°C)
4:3	PKG	RO	0x1	封装类型 这个位域规定了封装类型。该值编码如下（所有其它编码保留）： 值 描述值 0x1 LQFP封装
2	ROHS	RO	1	RoHS兼容性 这个位决定器件是否符合RoHS标准。A1表示器件符合RoHS标准。
1:0	QUAL	RO	-	认证情况 这个位域决定了器件的认证情况。该值编码如下（所有其它编码保留）： 值 描述值 0x0 工程样片(未经认证) 0x1 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0x003F.001F

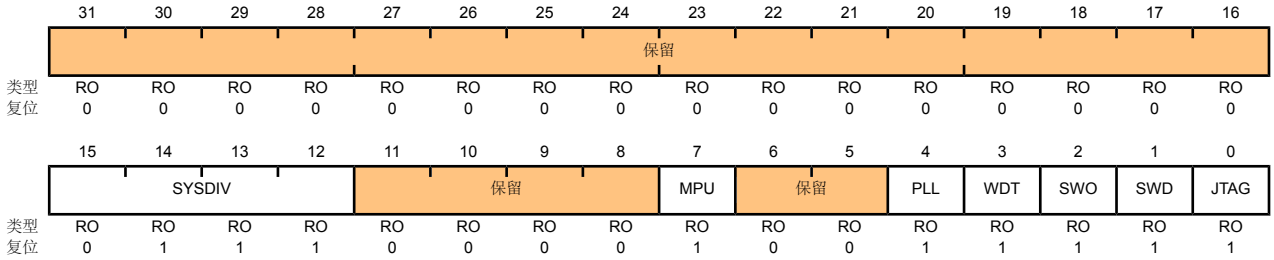


位/域	名称	类型	复位	描述
31:16	SRAMSZ	RO	0x003F	SRAM大小 表示片内SRAM存储器的大小。 值 描述值 0x003F SRAM为16KB
15:0	FLASHSZ	RO	0x001F	Flash大小 表示片内Flash存储器的大小。 值 描述值 0x001F Flash为64KB

寄存器**14:** 器件功能**1** （**DC1**）， 偏移量 **0x010**

该寄存器根据元件来预先定义并能用来验证其特性。PWM、SARADC0、MAXADCSPD、WDT, SWO、SWD和 JTAG位屏蔽**RCGC0**、**SCGC0**和 **DCGC0**寄存器。其它位为0。MAXADCSPD固定为**DC1**中指定的最大值。

器件功能1 (DC1)
偏移量0x010
类型RO, 复位0x0000.709F



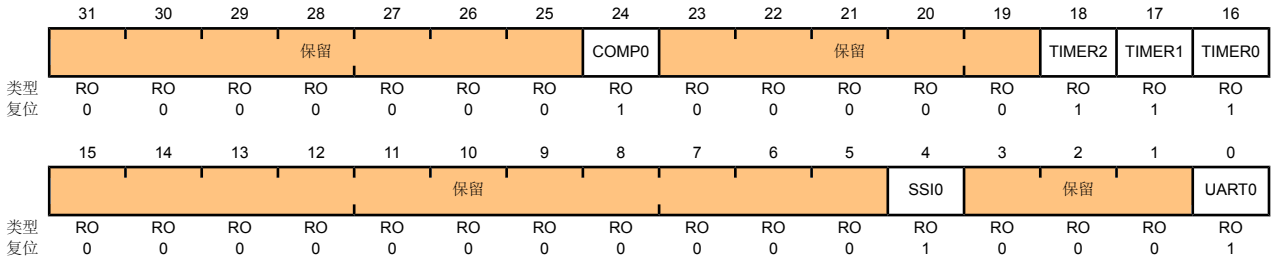
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
15:12	SYSDIV	RO	0x7	系统时钟分频器。系统时钟的最小4位分频值。复位值与硬件无关。要想了解如何使用SYSDIV位来改变系统时钟除数，请参考**RCC**寄存器。“RCC寄存器”。 值 描述值 0x7 指定一个PLL八分频的25-MHz时钟。
11:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7	MPU	RO	1	MPU存在标志。置位时表示存在Cortex-M3存储器保护单元（MPU）。有关MPU的详细内容请参考Cortex-M3技术参考手册。
6:5	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

该寄存器由元件进行预先定义并可用来验证其特性。

器件功能2 (DC2)

偏移量0x014
类型RO, 复位0x0107.0011



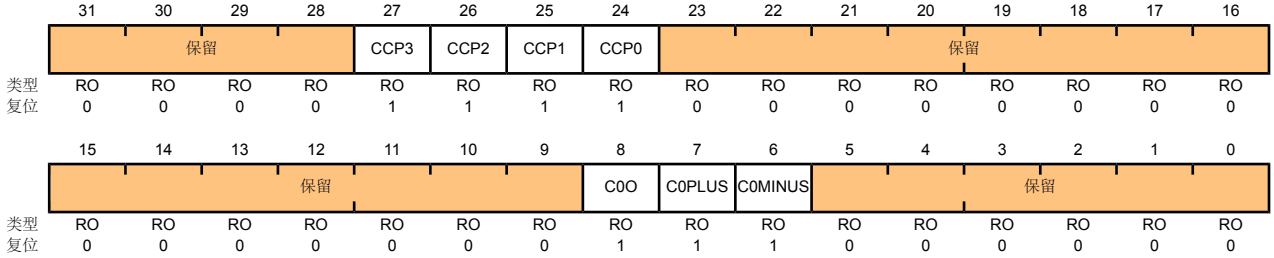
位/域	名称	类型	复位	描述
31:25	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
24	COMP0	RO	1	模拟比较器0存在标志。置位时表示存在模拟比较器0。
23:19	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
18	TIMER2	RO	1	定时器2存在标志。置位时表示存在通用定时器模块2。
17	TIMER1	RO	1	定时器1存在标志。置位时，表示存在通用定时器模块1。
16	TIMER0	RO	1	定时器0存在标志。置位时表示存在通用定时器模块0。
15:5	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-写操作过程中应该保持不变。
4	SSI0	RO	1	SSI0存在标志。置位时表示存在SSI模块0。
3:1	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。
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

该寄存器根据元件来预先定义并能用来验证其特性。

器件功能3 (DC3)

偏移量0x018
类型RO, 复位0x0F00.01C0



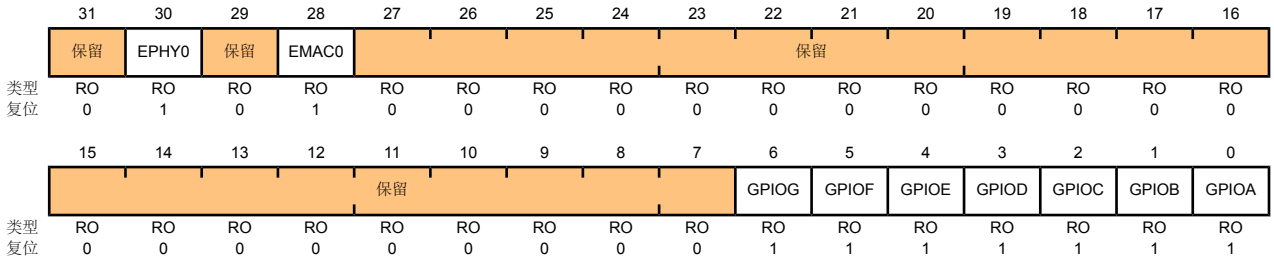
位/域	名称	类型	复位	描述
31:28	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
27	CCP3	RO	1	CCP3管脚存在标志。置位时表示存在捕获/比较/PWM管脚3。
26	CCP2	RO	1	CCP2管脚存在标志。置位时表示存在捕获/比较/PWM管脚2。
25	CCP1	RO	1	CCP1管脚存在标志。置位时表示存在捕获/比较/PWM管脚1。
24	CCP0	RO	1	CCP0管脚存在标志。置位时表示存在捕获/比较/PWM管脚0。
23:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
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。
5:0	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17: 器件功能4 （DC4）， 偏移量 0x01C

这个寄存器由器件预定义，可以用来验证特性。

器件功能4 (DC4)

偏移量0x01C
类型RO, 复位0x5000.007F



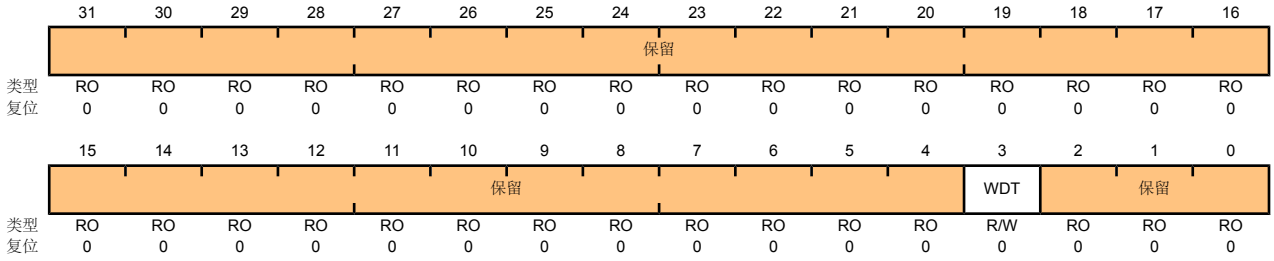
位/域	名称	类型	复位	描述
31	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
30	EPHY0	RO	1	以太网PHY存在标志。置位时表示存在以太网PHY模块0。
29	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
28	EMAC0	RO	1	以太网MAC0存在标志。置位时表示存在以太网MAC模块0。
27:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	GPIOG	RO	1	GPIO端口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)** 寄存器中的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0 (RCGC0)

偏移量0x100
类型RW, 复位0x00000040



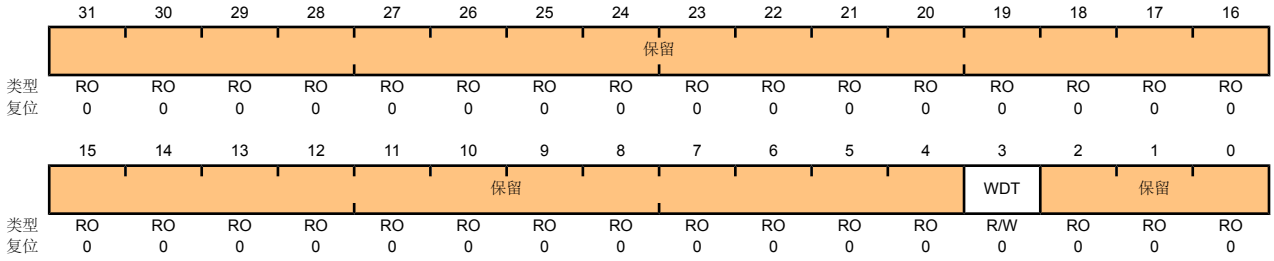
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
3	WDT	R/W	0	WDT时钟选通控制。这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

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



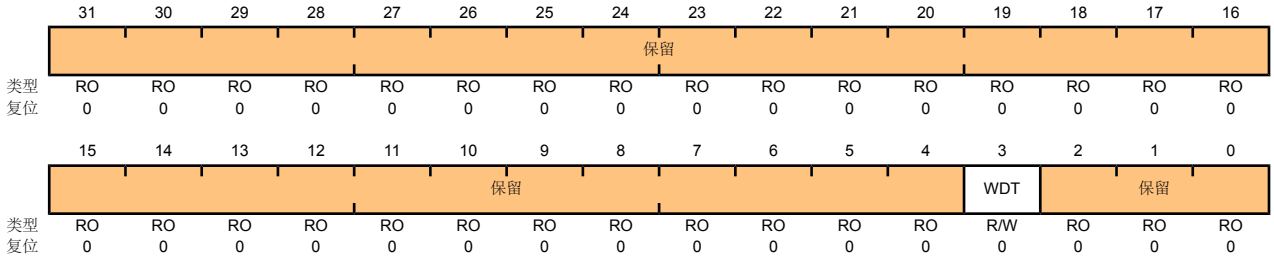
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
3	WDT	R/W	0	WDT时钟选通控制。这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

寄存器**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



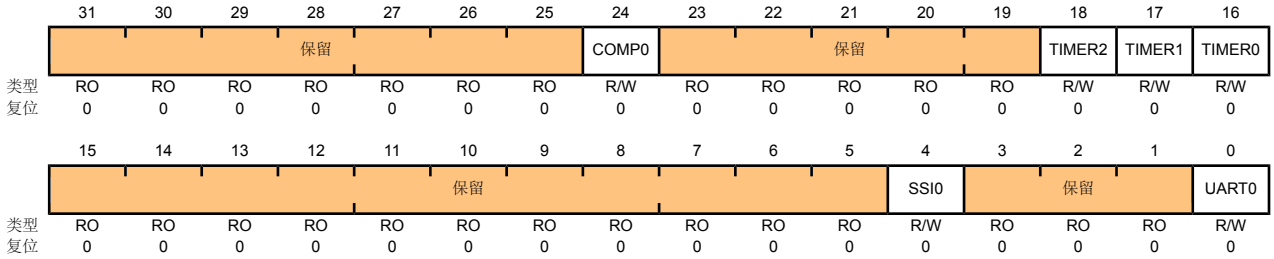
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
3	WDT	R/W	0	WDT时钟选通控制。这个位控制WDT模块的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线故障。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

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)**寄存器中的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
31:25	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。
24	COMP0	R/W	0	模拟比较器0时钟选通控制。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
23:19	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。
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。
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。
15:5	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-写操作过程中应该保持不变。
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。

位/域	名称	类型	复位	描述
3:1	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。
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1 (SCGC1)

偏移量0x114
类型RW, 复位0x00000000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	保留											SSI0	保留			UART0
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	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
31:25	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。
24	COMP0	R/W	0	模拟比较器0时钟选通控制。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。
23:19	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。
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系统错误。
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。
15:5	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-写操作过程中应该保持不变。
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。

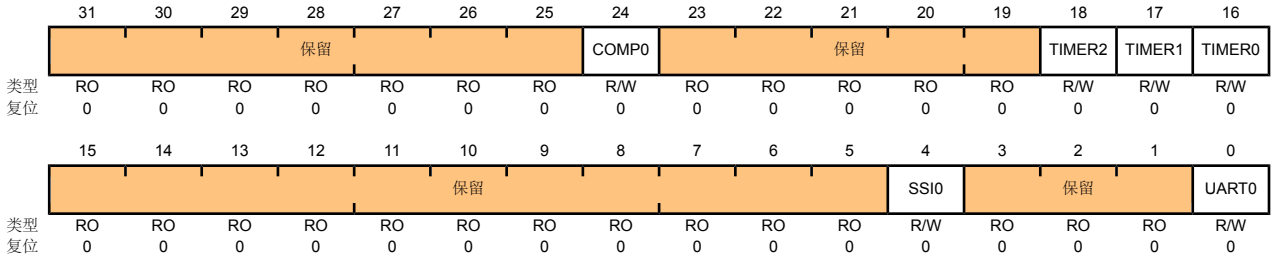
位/域	名称	类型	复位	描述
3:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
0	UART0	R/W	0	UART0时钟选通控制。这个位控制UART模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。

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1 (DCGC1)

偏移量0x124
类型RW, 复位0x00000000



位/域	名称	类型	复位	描述
31:25	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。
24	COMP0	R/W	0	模拟比较器0时钟选通控制。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。
23:19	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。
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。
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。
15:5	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-写操作过程中应该保持不变。
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。

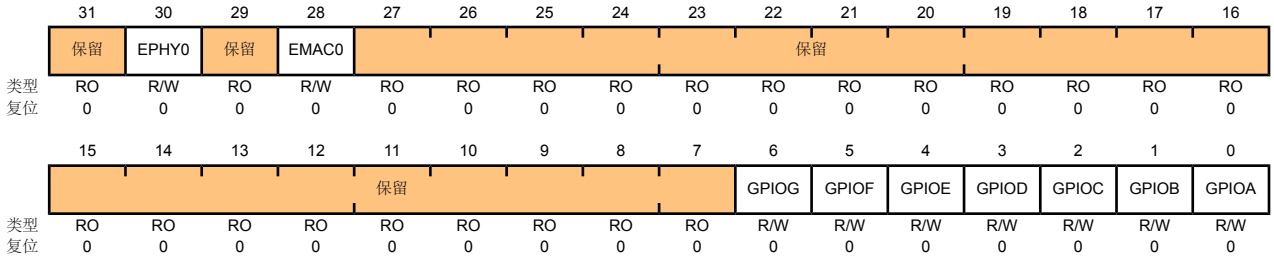
位/域	名称	类型	复位	描述
3:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
0	UART0	R/W	0	UART0时钟选通控制。这个位控制UART模块0的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。

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)**寄存器中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	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。
30	EPHY0	R/W	0	PHY0时钟选通控制。这个位控制以太网PHY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。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。
29	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。
28	EMAC0	R/W	0	MAC0时钟选通控制。这个位控制以太网MAC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。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。
27:7	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	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。

位/域	名称	类型	复位	描述
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。
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2 (SCGC2)

偏移量0x118
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	EPHY0	保留	EMAC0	保留											
类型	RO	R/W	RO	R/W	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									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
类型	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	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	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。
30	EPHY0	R/W	0	PHY0时钟选通控制。这个位控制以太网PHY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。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。
29	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。
28	EMAC0	R/W	0	MAC0A时钟选通控制。这个位控制以太网MAC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。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。
27:7	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	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。

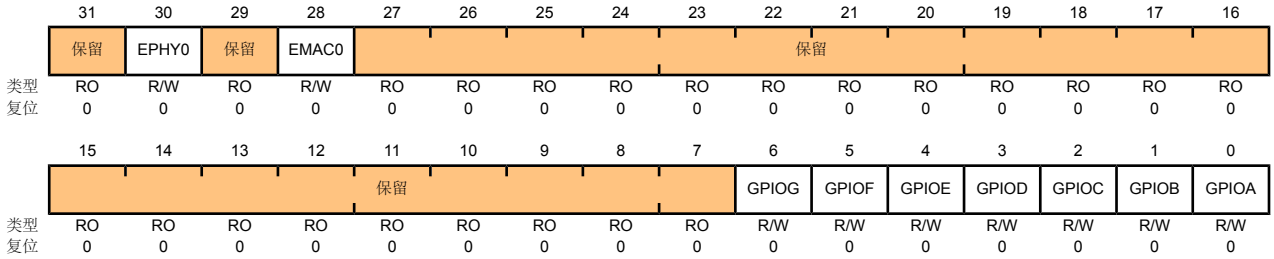
位/域	名称	类型	复位	描述
3	GPIOD	R/W	0	端口D时钟选通控制。这个位控制端口D的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
2	GPIOC	R/W	0	端口C时钟选通控制。这个位控制端口C的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
1	GPIOB	R/W	0	端口B时钟选通控制。这个位控制端口B的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，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2 (DCGC2)

偏移量0x128
类型RW, 复位0x00000000



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-写操作过程中应该保持不变。
30	EPHY0	R/W	0	PHY0时钟选通控制。这个位控制以太网PHY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。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。
29	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。
28	EMAC0	R/W	0	MAC0A时钟选通控制。这个位控制以太网MAC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。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。
27:7	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	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。

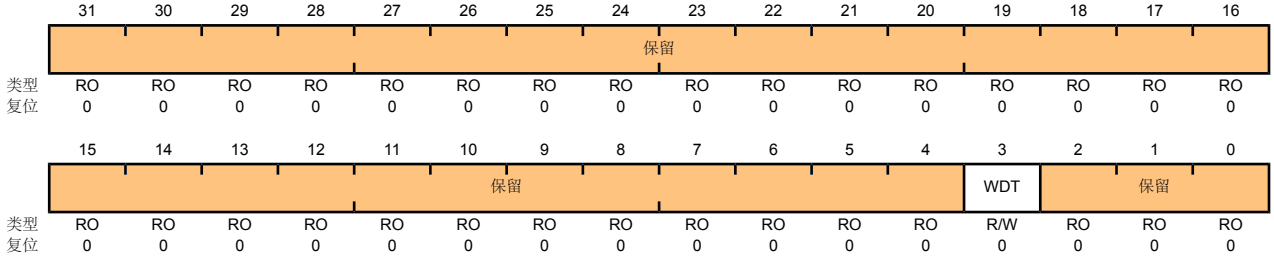
位/域	名称	类型	复位	描述
3	GPIOD	R/W	0	端口D时钟选通控制。这个位控制端口D的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
2	GPIOC	R/W	0	端口C时钟选通控制。这个位控制端口C的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功能单元不使用时钟，那么在对该单元进行读或写操作时都将返回总线错误。
1	GPIOB	R/W	0	端口B时钟选通控制。这个位控制端口B的时钟选通。如果置位，该功能单元将接收时钟并运行。否则不使用时钟且不运行。如果功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



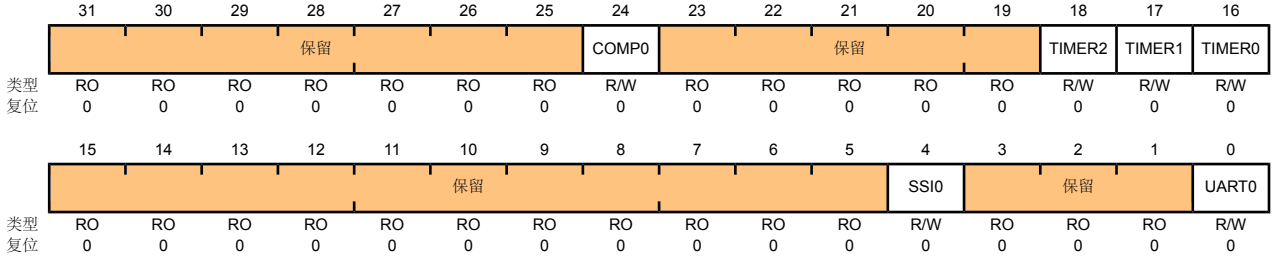
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
3	WDT	R/W	0	WDT复位控制。看门狗功能单元的复位控制。
2:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。

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



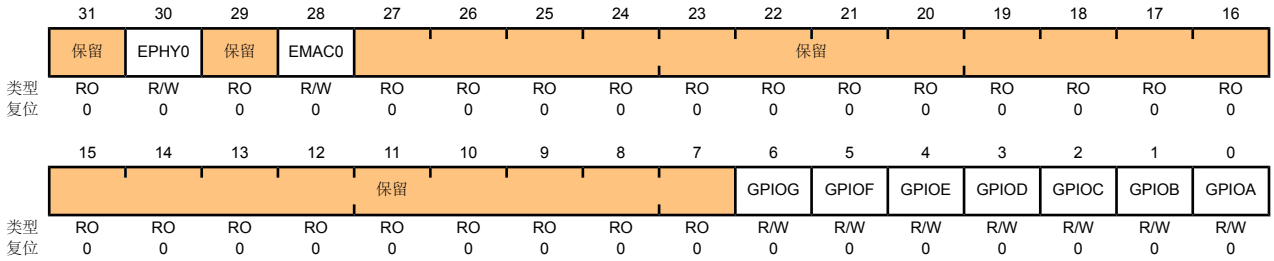
位/域	名称	类型	复位	描述
31:25	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
24	COMP0	R/W	0	模拟比较器0复位控制。模拟比较器0的复位控制。
23:19	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
18	TIMER2	R/W	0	定时器2复位控制。通用定时器模块2的复位控制。
17	TIMER1	R/W	0	定时器1复位控制。通用定时器模块1的复位控制。
16	TIMER0	R/W	0	定时器0复位控制。通用定时器模块0的复位控制。
15:5	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-写操作过程中应该保持不变。
4	SSI0	R/W	0	SSI0复位控制。SSI功能单元0的复位控制。
3:1	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。
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



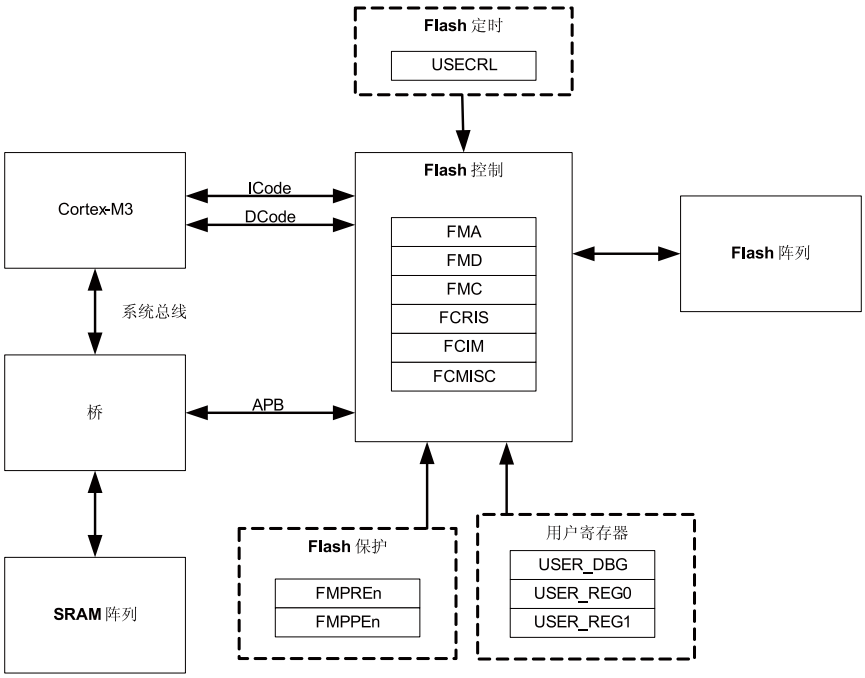
位/域	名称	类型	复位	描述
31	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
30	EPHY0	R/W	0	PHY0复位控制。以太网PHY功能单元0的复位控制。
29	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
28	EMAC0	R/W	0	MAC0复位控制。以太网MAC功能单元0的复位控制。
27:7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
6	GPIOG	R/W	0	端口G复位控制。GPIO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 内部存储器

LM3S6100微控制器带有16 KB的bit-banded SRAM和64 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。

7.1 方框图

图 **7-1. Flash**方框图



7.2 功能描述

这一节描述了Flash存储器和SRAM存储器的功能。

7.2.1 SRAM存储器

Stellaris®器件的内部SRAM位于器件存储器映射的地址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”。

7.2.2 Flash存储器

Flash是由一组可独立擦除的1KB区块所构成的。对一个区块进行擦除将使该区块的全部内容复位为1。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 360页。

7.2.2.1 Flash存储器时序

Flash的时序是由Flash控制器自动处理的。但是，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）。

7.2.2.2 Flash存储器保护

在1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 7-1 在 95页所示。

表 **7-1. 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 96页中讨论。

7.3 Flash存储器的初始化和配置

7.3.1 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**。

7.3.1.1 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。

7.3.1.2 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。

7.3.1.3 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。

7.3.2 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 7-2 在 96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 **7-2. 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。

7.4 寄存器映射

表 7-3 在 97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 **7-3.** 内部存储器寄存器映射

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	99
0x004	FMD	R/W	0x0000.0000	Flash存储器数据	100
0x008	FMC	R/W	0x0000.0000	Flash存储器控制	101
0x00C	FCRIS	RO	0x0000.0000	Flash控制器原始中断状态	103
0x010	FCIM	R/W	0x0000.0000	Flash控制器中断屏蔽	104
0x014	FCMISC	R/W1C	0x0000.0000	Flash控制器可屏蔽中断的状态和清除	105
系统控制偏移量					
0x130 和 0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash存储器保护读使能0	107
0x134 和 0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash存储器保护编程使能0	108
0x140	USECRL	R/W	0x16	USec重装	106
0x1D0	USER_DBG	RW	0xFFFF.FFFE	用户调试	109
0x1E0	USER_REG0	RW	0x8FFF.FFFF	用户寄存器0	110
0x1E4	USER_REG1	RW	0x8FFF.FFFF	用户寄存器1	111
0x204	FMPRE1	RW	0x0000.0000	Flash存储器保护读使能1	112
0x208	FMPRE2	RW	0x0000.0000	Flash存储器保护读使能2	113
0x20C	FMPRE3	RW	0x0000.0000	Flash存储器保护读使能3	114
0x404	FMPPE1	RW	0x0000.0000	Flash存储器保护编程使能1	115
0x408	FMPPE2	RW	0x0000.0000	Flash存储器保护编程使能2	116
0x40C	FMPPE3	RW	0x0000.0000	Flash存储器保护编程使能3	117

7.5 Flash寄存器描述（Flash控制偏移量）

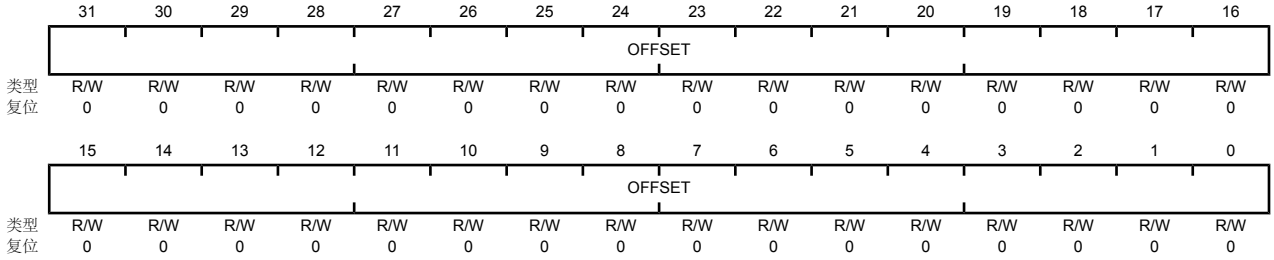
本节的剩余部分按照地址偏移的数字顺序排列和描述Flash存储器寄存器。

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



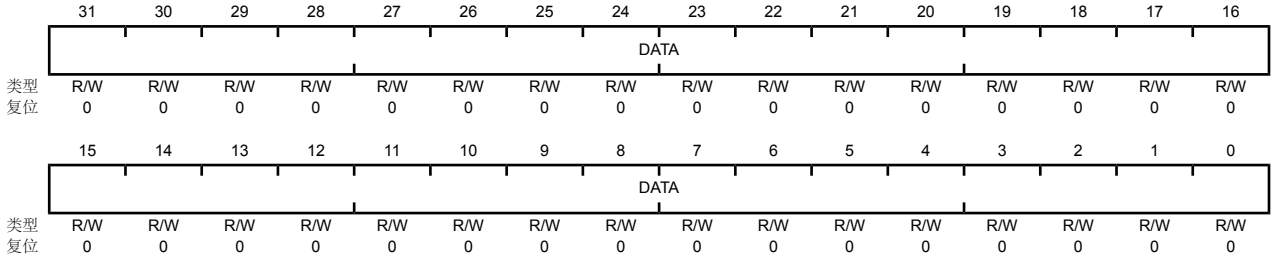
位/域	名称	类型	复位	描述
31:0	OFFSET	R/W	0x0	地址偏移量 Flash中执行操作处（非易失性寄存器除外）的地址偏移量（有关这个域的值的详见“非易失性寄存器编程” 在 96页）。

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

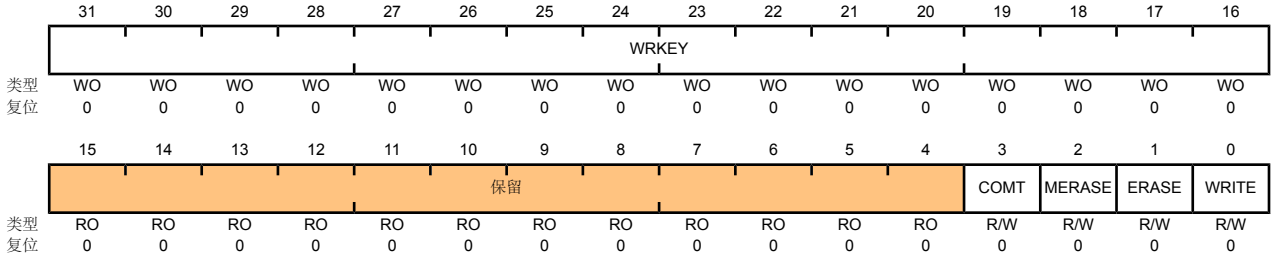
当该寄存器被写入时，Flash控制器将对**Flash**存储器地址（**FMA**）指定的位置启动适当周期数目的访问操作（见 99页）。如果是写访问，**Flash**存储器数据（**FMD**）寄存器包含的数据（见 100页）被写入。

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	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	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。

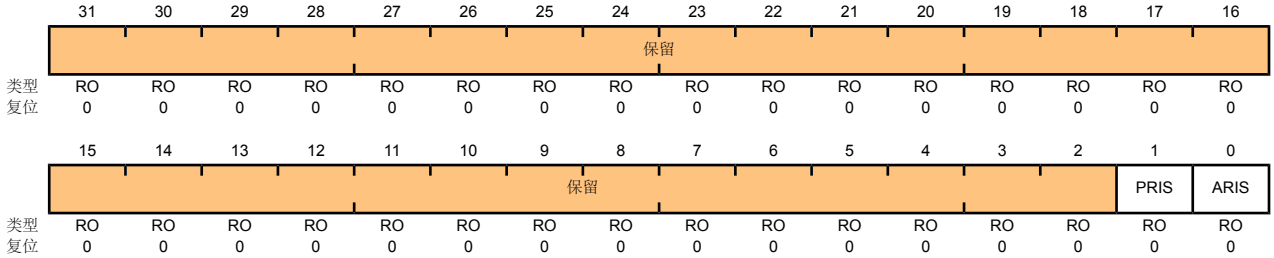
位/域	名称	类型	复位	描述
0	WRITE	R/W	0	写一个字到Flash存储器 如果该位被置位，**FMD**存放的数据写入到**FMA**的内容所指定的位置。写0对该位的状态无影响。 如果该位被读取，则提供前面写更新的状态。如果前面的写访问结束，则返回0；否则，如果写访问没有结束，返回1。 该操作所需的时间最多可达50 µs。

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



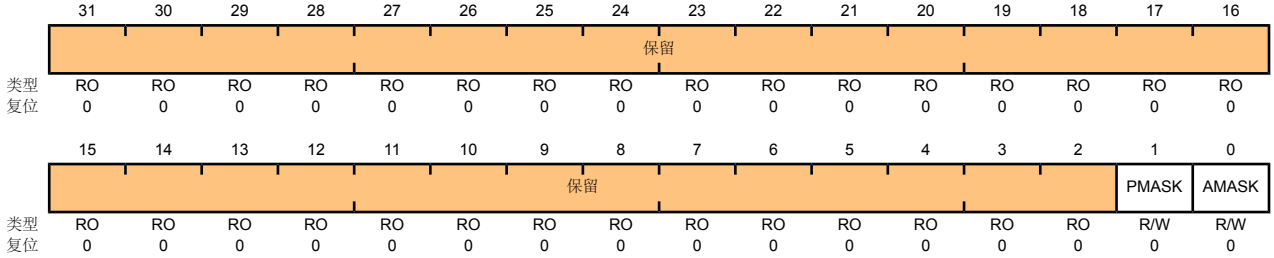
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PRIS	RO	0	编程的原始中断状态 该位表示编程周期的当前状态。如果置位，则编程周期完成；如果清零，则编程周期还没有完成。编程周期是指通过**Flash**存储器控制（**FMC**）寄存器位产生的写或擦除操作（见see 101页）。
0	ARIS	RO	0	访问的原始中断状态 该位指示Flash是否被不正确地访问。如果置位，则表示程序试图访问Flash的操作与**Flash** 存储器保护读使能（**FMPREn**）和 **Flash**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



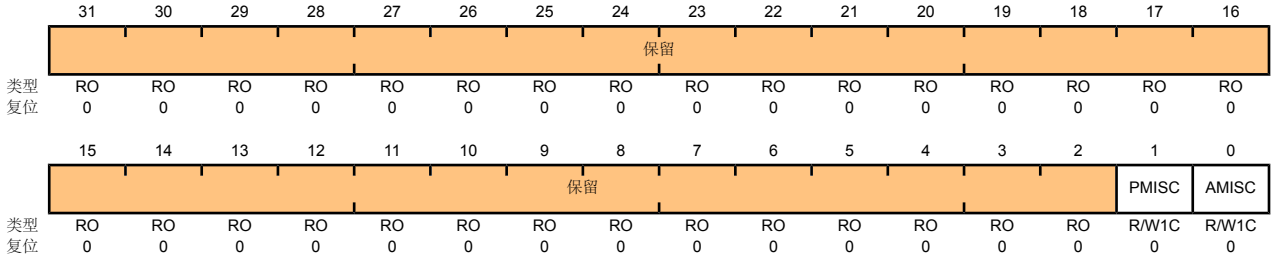
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PMASK	R/W	0	编程中断屏蔽 该位控制着编程原始中断状态向控制器的报告。如果置位，则向控制器提交编程所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。
0	AMASK	R/W	0	访问中断屏蔽。 该位控制着访问原始中断向控制器的报告。如果置位，则向控制器提交访问所产生的中断。否则，中断会被记录下来，但并不会提交到控制器。

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	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	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 103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。

7.6 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。

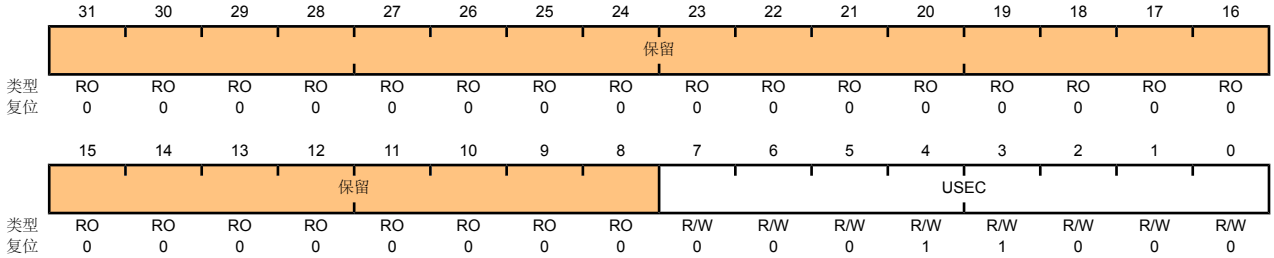
寄存器7: USec重装（USECRL），偏移量 0x140

注意: 偏移量是相对0x400F.E000的系统控制基址而言的。

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0x16



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	USEC	R/W	0x18	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 0x18 (24 MHz) 。

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

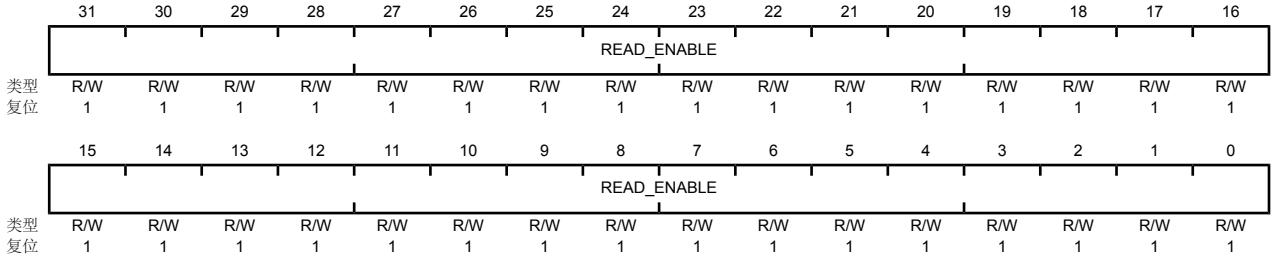
注意: 这个寄存器采用别名，用来向后兼容。

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 和 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”所示的策略。 值 描述值 0xFFFFFFFF 使能64KB Flash。

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

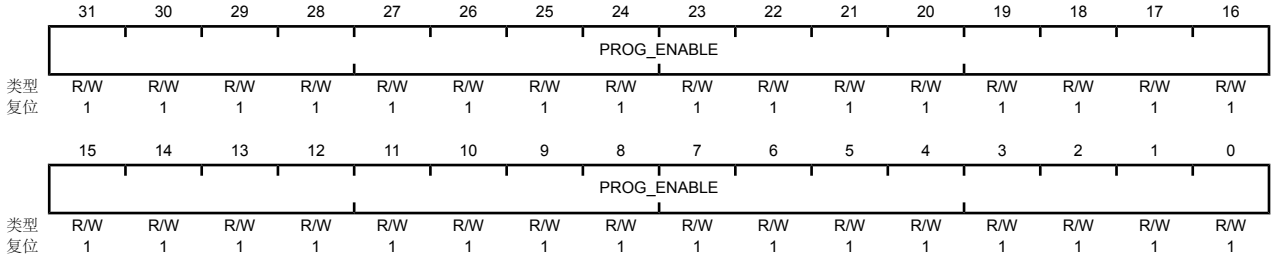
注意: 这个寄存器采用别名，用来向后兼容。

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0 (FMPPE0)

偏移量0x134 和 0x400
类型R/W, 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。结合 “Flash保护策略组合”所示的策略。 值 描述值 0xFFFFFFFF 使能64KB 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。 通过将DBG1 位变为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	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NW	DATA														
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
	DATA														DBG1	DBG0
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	0

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。若想允许调试，DBG1 位必须为1， DBG0 必须为0。
0	DBG0	R/W	0	调试控制0。要想允许调试， DBG1 位必须为1， DBG0 必须为0。

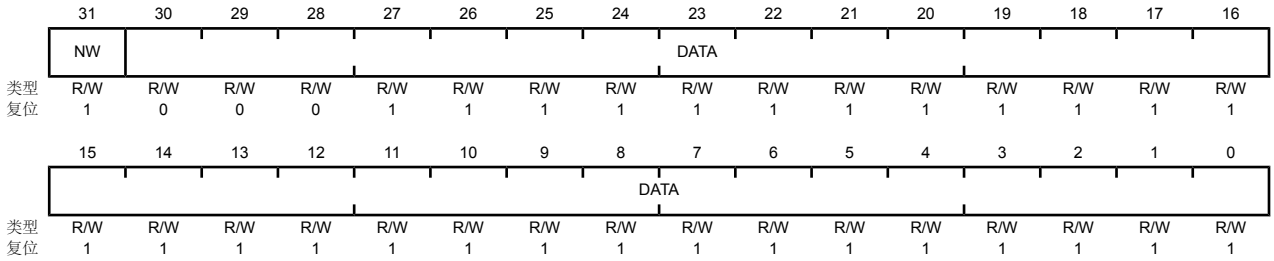
寄存器11: 用户寄存器0 （USER_REG0）， 偏移量 0x1E0

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或其它非易失器件。

用户寄存器0 (USER_REG0)

偏移量0x1E0
类型RW, 复位0x8FFF.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位双字没有被写。
30:0	DATA	R/W	0xFFFFFFF	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。

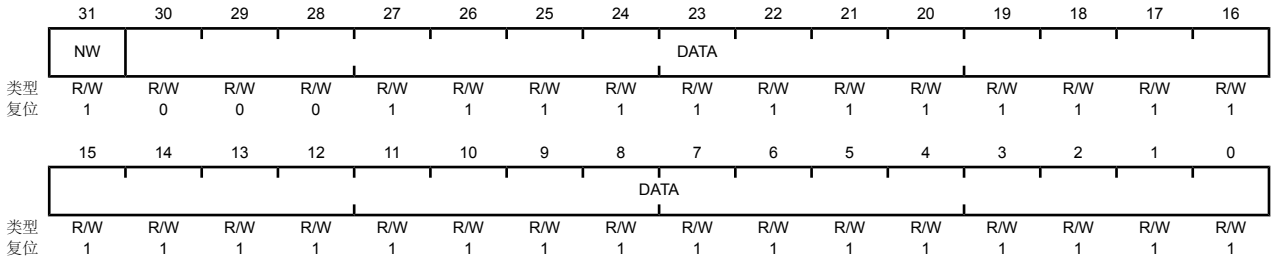
寄存器12: 用户寄存器1 （USER_REG1）， 偏移量 0x1E4

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或其它非易失器件了。

用户寄存器1 (USER_REG1)

偏移量0x1E4
类型RW, 复位0x8FFF.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位双字没有被写。
30:0	DATA	R/W	0xFFFFFFF	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。

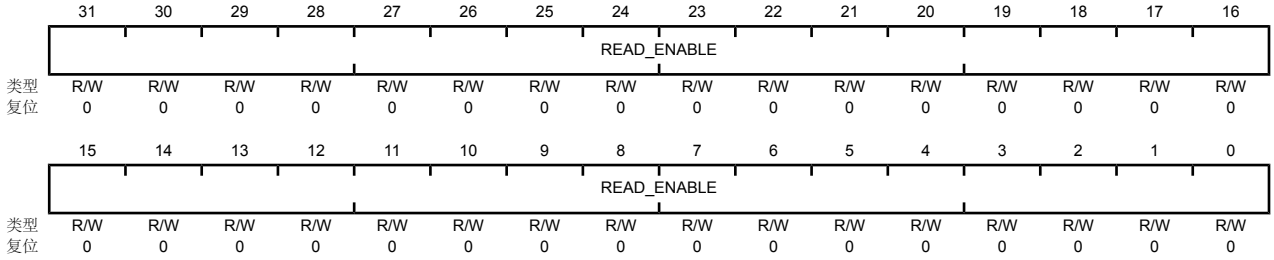
寄存器13: Flash存储器保护读使能1 （FMPRE1）， 偏移量 0x204

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0x0000.0000



位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB Flash。

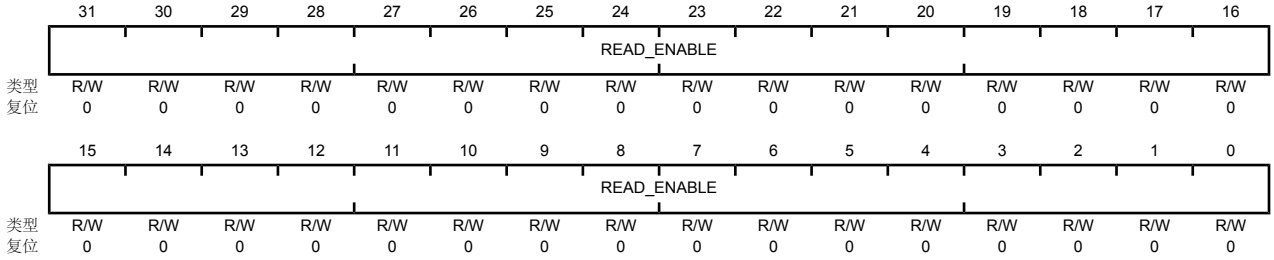
寄存器14: Flash存储器保护读使能2 （FMPRE2）， 偏移量 0x208

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0x0000.0000



位/域	名称	类型	复位	描述
31:0	READ_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB 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3 (FMPRE3)

偏移量0x20C
类型RW, 复位0x0000.0000

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 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 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
31:0	READ_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB Flash。

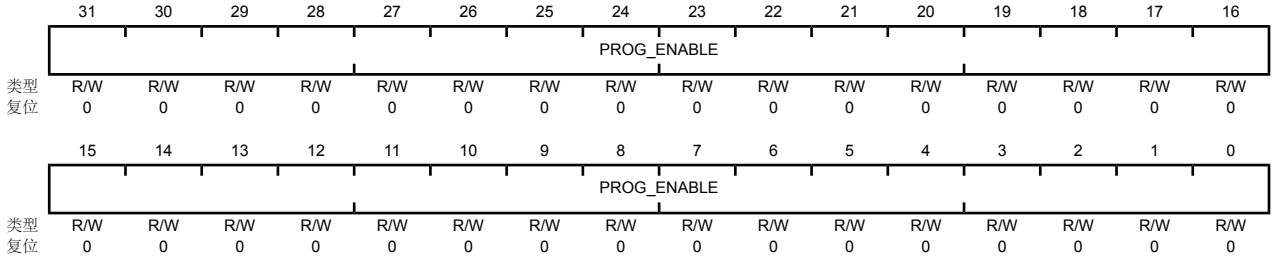
寄存器16: Flash存储器保护编程使能1 （FMPPE1）， 偏移量 0x404

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** 保持只执行位)。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0x0000.0000



位/域	名称	类型	复位	描述
31:0	PROG_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB 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** 保持只执行位)。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** 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0x0000.0000

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	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
	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	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
31:0	PROG_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB 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** 保持只执行位)。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0x0000.0000

	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	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
31:0	PROG_ENABLE	R/W	0x00000000	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 0x00000000 使能64KB Flash。

8 通用输入/输出端口（GPIO）

GPIO模块由 7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, 和端口G, ）。GPIO模块遵循FiRM规范，并且支持10-30 个可编程的输入/输出管脚，具体取决于正在使用的外设。

GPIO模块具有以下的特性：

- 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

8.1 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中的独立硬件的实例化（hardware instantiation）。LM3S6100 微控制器含有 7个端口以及 7个物理 GPIO块。

8.1.1 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。

8.1.1.1 数据方向操作

GPIO 方向 **(GPIODIR)** 寄存器（见 125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。

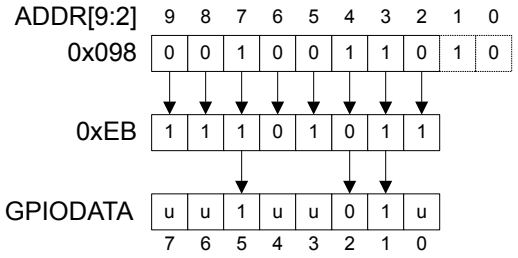
8.1.1.2 数据寄存器操作

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。

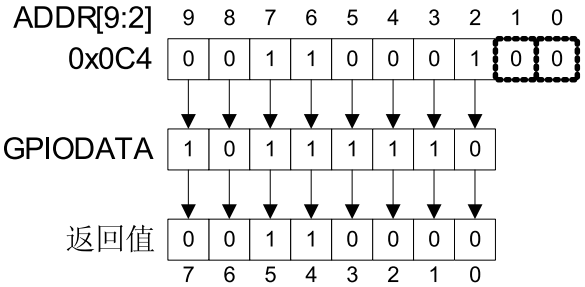
例如，将0xEB的值写入地址GPIODATA+0x098处，结果如图 8-1 在 119页所示，其中，u表示没有被写操作改变的数据。

图 **8-1. GPIODATA** 写实例



在读操作过程中，如果与数据位相关联的地址位被设为1，那么读取该值。如果与数据位相关联的地址位被设为0，那么不管它的实际值是什么，都将该值读作0。例如，读取地址GPIODATA+0x0C4处的值，结果如 图 8-2 在 119页所示。

图 **8-2. GPIODATA** 读实例



8.1.2 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 126页）
- **GPIO** 中断双边沿 **(GPIOIBE)** 寄存器（见 127页）
- **GPIO** 中断事件 **(GPIOIEV)**寄存器（见 128页）

通过**GPIO** 中断屏蔽 **(GPIOIM)**寄存器可以使能或禁能中断（见 129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 130页和 131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 132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。

8.1.3 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 133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。

8.1.4 确认（commit）控制

确认控制寄存器提供一个保护层，以防止对重要的硬件外设进行意外编程。对 **GPIO** 备用功能选择 **(GPIOAFSEL)** 寄存器（见 133页）保护位的写操作没有被确认进行存储直至 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 142页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 143页）的相应位已设为1。

8.1.5 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。

8.1.6 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。

8.2 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 8-1 在 120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 8-2 在 121页 给出了为GPIO端口的管脚2配置上升沿中断的方法。

表 **8-1. GPIO** 端口配置实例

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	?	?	?	?
数字输入 (定时器 CCP)	1	X	0	1	?	?	X	X	X	X
数字输出 (定时器 PWM)	1	X	0	1	?	?	?	?	?	?
数字输入/输出 (SSI)	1	X	0	1	?	?	?	?	?	?

配置	GPIO 寄存器位的值[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
数字输入/输出 (UART)	1	X	0	1	?	?	?	?	?	?
模拟输入（比较器）	0	0	0	0	0	0	X	X	X	X
数字输出（比较器）	1	X	0	1	?	?	?	?	?	?

a. X=忽略（无关位）

?=可以是0或1，具体取决于配置。

表 **8-2. GPIO** 中断配置实例

寄存器	期望的中断事件触发	管脚2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）

8.3 寄存器映射

表 8-3 在 122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

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 **8-3. 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	124
0x400	GPIODIR	R/W	0x0000.0000	GPIO 方向	125
0x404	GPIOIS	R/W	0x0000.0000	GPIO 中断检测	126
0x408	GPIOIBE	R/W	0x0000.0000	GPIO 中断双边沿	127
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO 中断事件	128
0x410	GPIOIM	R/W	0x0000.0000	GPIO 中断屏蔽	129
0x414	GPIORIS	RO	0x0000.0000	GPIO 原始中断状态	130
0x418	GPIOMIS	RO	0x0000.0000	GPIO 屏蔽后的中断状态	131
0x41C	GPIOICR	W1C	0x0000.0000	GPIO 中断清除	132
0x420	GPIOAFSEL		-	GPIO 备用功能选择	133
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA 驱动选择	134
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA 驱动选择	135
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA 驱动选择	136
0x50C	GPIOODR	R/W	0x0000.0000	GPIO 开漏选择	137
0x510	GPIOPUR	R/W	-	GPIO 上拉选择	138
0x514	GPIOPDR	R/W	0x0000.0000	GPIO 下拉选择	139
0x518	GPIOSLR	R/W	0x0000.0000	GPIO 斜率控制选择	140
0x51C	GPIODEN	R/W	-	GPIO 数字使能	141
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO 锁定	142
0x524	GPIOCR	-	-	GPIO 确认	143
0xFD0	GPIOPeriphID4	RO	0x0x0000.0000	GPIO 外设标识4	144
0xFD4	GPIOPeriphID5	RO	0x0x0000.0000	GPIO 外设标识 5	145
0xFD8	GPIOPeriphID6	RO	0x0x0000.0000	GPIO 外设标识 6	146
0xFDC	GPIOPeriphID7	RO	0x0x0000.0000	GPIO 外设标识7	147
0xFE0	GPIOPeriphID0	RO	0x0x0000.0061	GPIO 外设标识0	148
0xFE4	GPIOPeriphID1	RO	0x0x0000.0000	GPIO 外设标识 1	149
0xFE8	GPIOPeriphID2	RO	0x0x0000.0018	GPIO 外设标识2	150

偏移量	名称	类型	复位	描述	见页面
0xFEC	GPIOPeriphID3	RO	0x0x0000.0001	GPIO 外设标识3	151
0xFF0	GPIOPCellID0	RO	0x0x0000.000D	GPIO PrimeCell 标识0	152
0xFF4	GPIOPCellID1	RO	0x0x0000.00F0	GPIO PrimeCell 标识1	153
0xFF8	GPIOPCellID2	RO	0x0x0000.0005	GPIO PrimeCell 标识2	154
0xFFC	GPIOPCellID3	RO	0x0x0000.00B1	GPIO PrimeCell 标识3	155

8.4 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 125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。

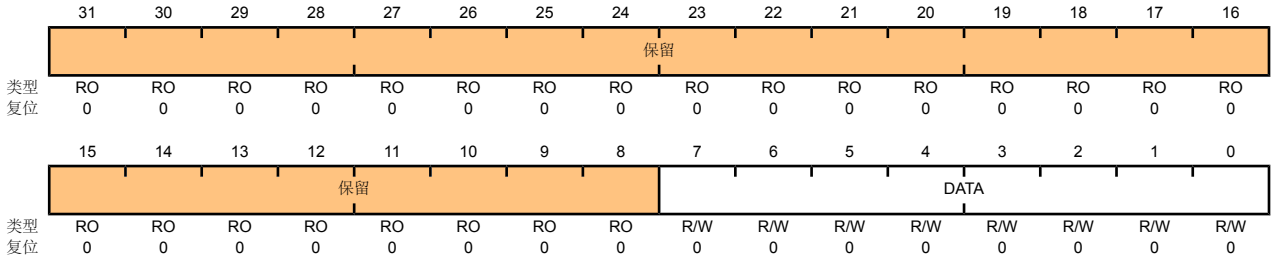
为了对**GPIODATA**寄存器执行写操作，由地址总线位[9:2]产生的相关屏蔽位必须为高电平。否则，该位的值不会被写操作改变。

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



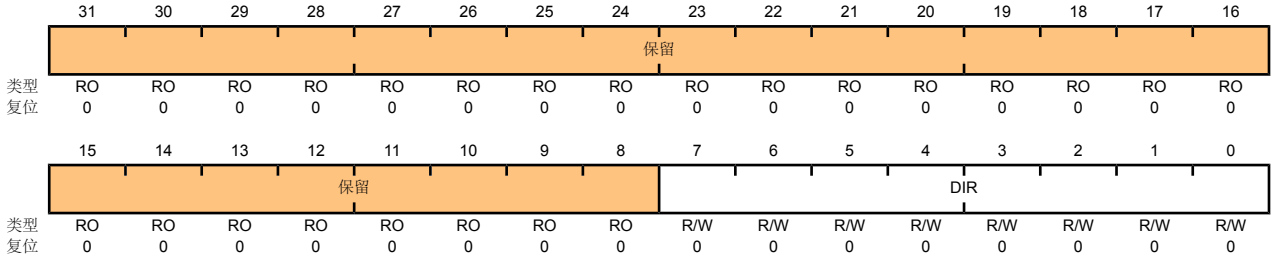
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	R/W	0	GPIO 数据 该寄存器被虚拟地映射到地址空间的256个单元中。为便于通过单独的驱动器读写这些寄存器，从这些寄存器读取的值和写入这些寄存器的值都被8条地址线`ipaddr[9:2]`屏蔽。读取该寄存器将返回其当前状态。写该寄存器仅影响那些没有被`ipaddr[9:2]`屏蔽的位和被配置成输出的位。关于读写操作的实例，请见“数据寄存器操作” 在 119页。

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



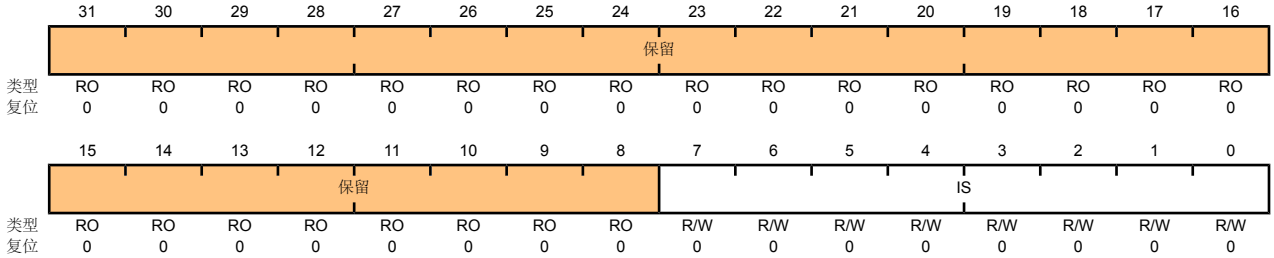
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DIR	R/W	0x00	GPIO 数据方向 0: 管脚为输入。 1: 管脚为输出。

寄存器**3: GPIO** 中断检测 （**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



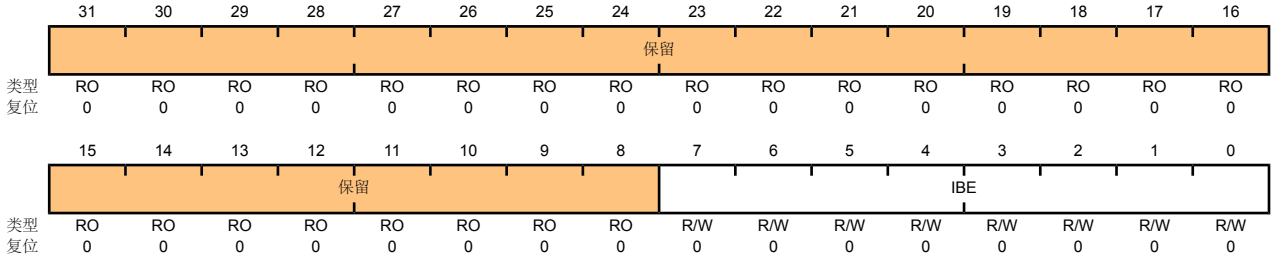
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IS	R/W	0x00	GPIO 中断检测 0: 检测的是相关管脚的边沿（边沿触发）。 1: 检测的是相关管脚的电平（电平触发）。

寄存器4: GPIO 中断双边沿（GPIOIBE），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 126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 128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



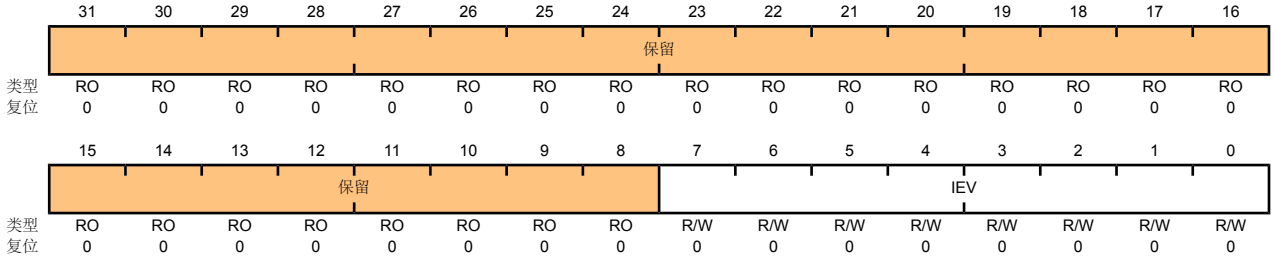
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IBE	R/W	0x00	GPIO 中断双边沿 0: 由**GPIO** 中断事件 **(GPIOIEV)**寄存器控制是否产生中断（见 128页）。 1: 相应管脚的上升沿和下降沿都会触发中断。 注意: 单边沿由**GPIOIEV**中相应的位来决定。

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 126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



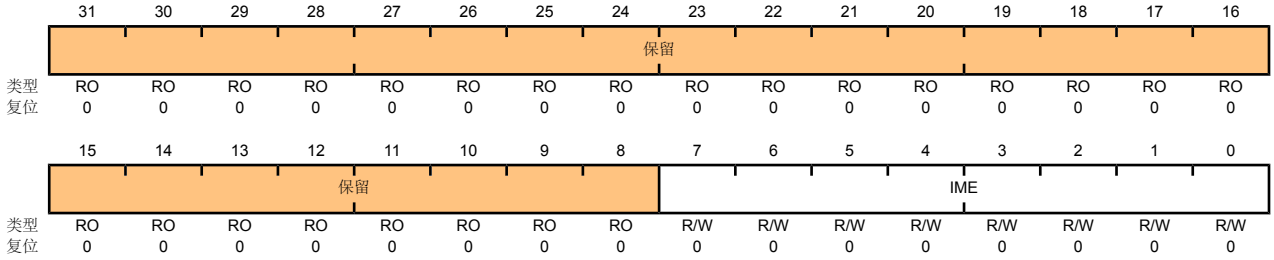
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IEV	R/W	0x00	GPIO 中断事件 0: 相应管脚上的下降沿或低电平触发中断。 1: 相应管脚的上升沿或高电平触发中断。

寄存器6: GPIO 中断屏蔽（GPIOIM），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



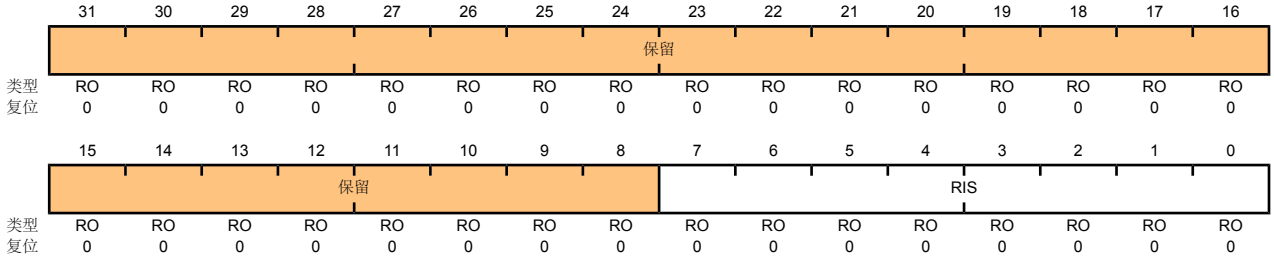
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	IME	R/W	0x00	GPIO 中断屏蔽使能 0: 相应管脚的中断被屏蔽。 1: 相应管脚的中断未被屏蔽。

寄存器7: GPIO 原始中断状态（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 129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	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	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）。 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。

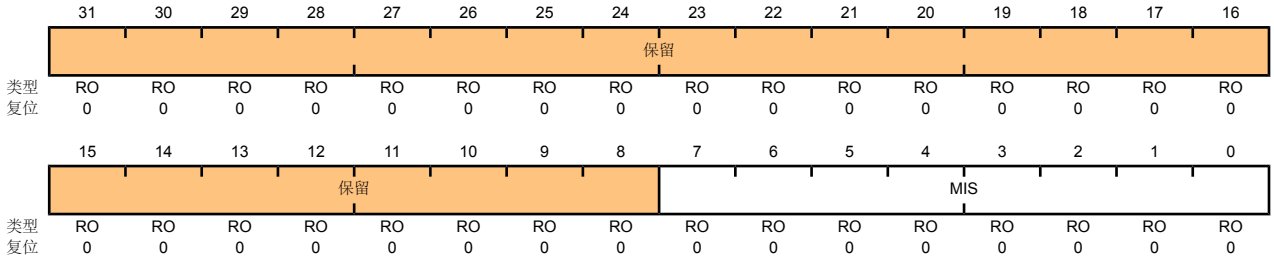
寄存器8: GPIO 屏蔽后的中断状态（GPIOMIS），偏移量 0x418

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



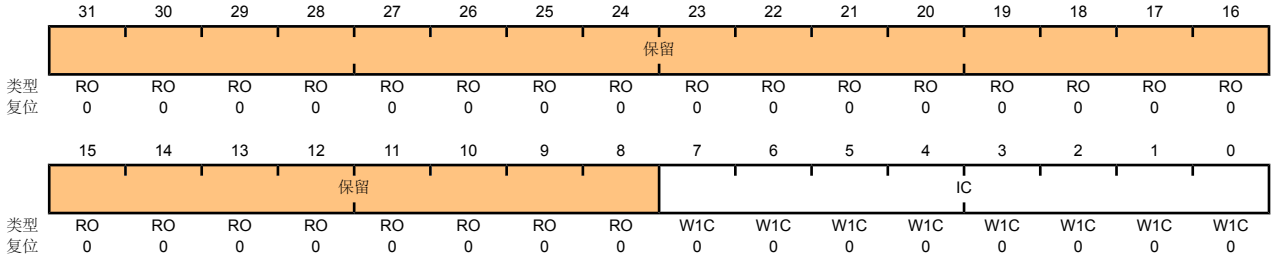
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	MIS	RO	0x00	GPIO 屏蔽后的中断状态 相应管脚上中断已屏蔽的值。 0: 相应的GPIO线路的中断未被激活。 1: 相应的GPIO线路发出中断。

寄存器9: GPIO 中断清除（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	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	IC	W1C	0x00	GPIO 中断清除 0: 相应的中断未受影响。 1: 相应的中断被清除。

寄存器10: GPIO 备用功能选择（GPIOAFSEL），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）。

确认控制寄存器提供一个保护层，以防止对重要的硬件外设进行意外编程。对 **GPIO** 备用功能选择 **(GPIOAFSEL)** 寄存器（见 133页）保护位的写操作没有被确认进行存储直至 **GPIO** 锁定 **(GPIOLOCK)** 寄存器（见 142页）已被解锁且 **GPIO** 确认 **(GPIOCR)** 寄存器（见 143页）的相应位已设为1。

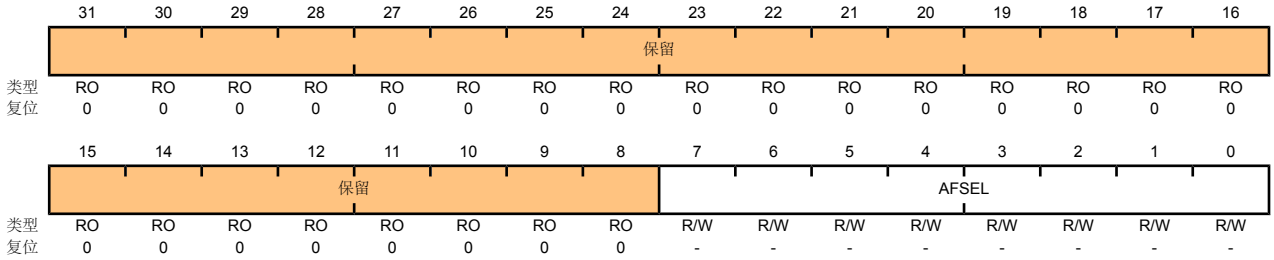
重要: 除了5个JTAG/SWD管脚（PB7 和PC[3:0]）之外，所有GPIO管脚默认都是三态管脚（**GPIOAFSEL=0, GPIODEN=0, GPIOPDR=0,** 且 **GPIOPUR**=0）。JTAG/SWD 管脚默认为 JTAG/SWD 功能（**GPIOAFSEL**=1, **GPIODEN=1** 且 **GPIOPUR**=1）。通过上电复位（POR）或外部复位（RST），可以让这两组管脚都回到其默认状态。

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, 复位-



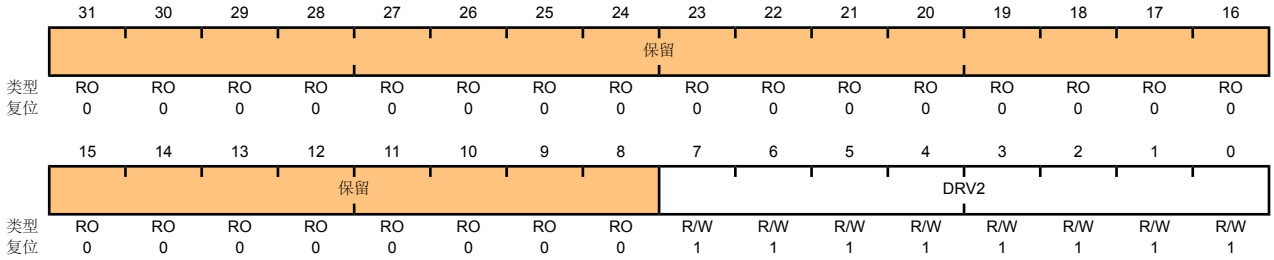
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读–修改–写操作过程中应当保持不变。
7:0	AFSEL	R/W	-	GPIO 备用功能选择 0: 软件控制相应的GPIO线（GPIO模式）。 1: 硬件控制相应的GPIO线（备用的硬件功能）。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**, **GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD功能。正因为这样，对于GPIO端口B，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



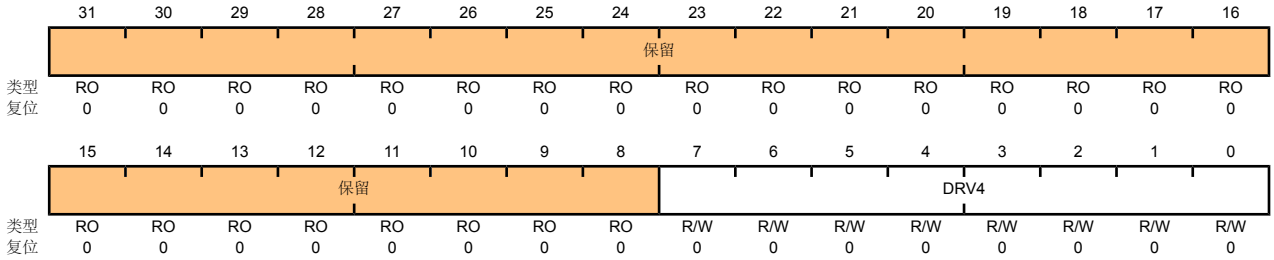
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV2	R/W	0xFF	输出端口 2-mA 驱动使能 向**GPIODR4[n]** 或 **GPIODR8[n]**写入1，都会使相应的2-mA使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

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



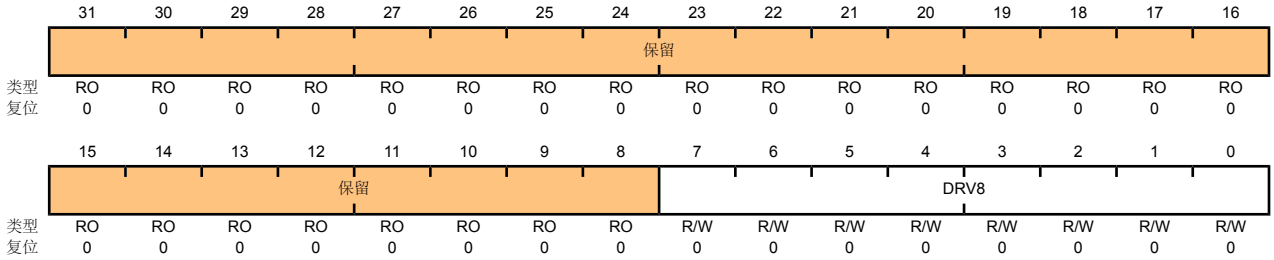
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV4	R/W	0x00	输出端口 4-mA 驱动使能 向**GPIODR2[n]** 或 **GPIODR8[n]**写入1，都会使相应的4-mA使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

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



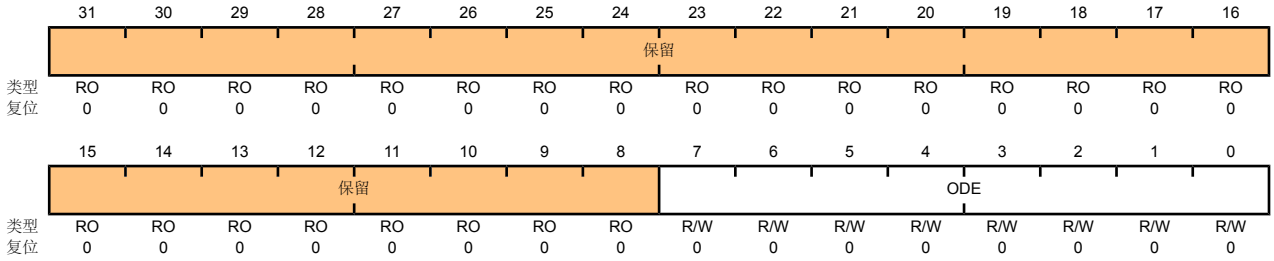
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DRV8	R/W	0x00	输出端口 8-mA 驱动使能 向**GPIODR2[n]** 或 **GPIODR4[n]**写入1，都会使相应的8-mA使能位清零。这种修改会在写操作之后的第二个时钟周期生效。

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 141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。

GPIO 开漏选择 (GPIOODR)

偏移量0x50C
类型R/W, 复位0x0000.0000



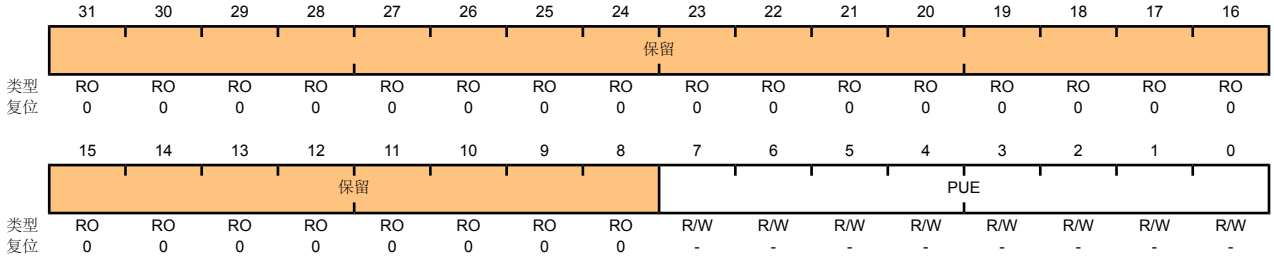
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	ODE	R/W	0x00	输出端口开漏使能 0: 开漏配置被禁能。 1: 开漏配置被使能。

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 139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-



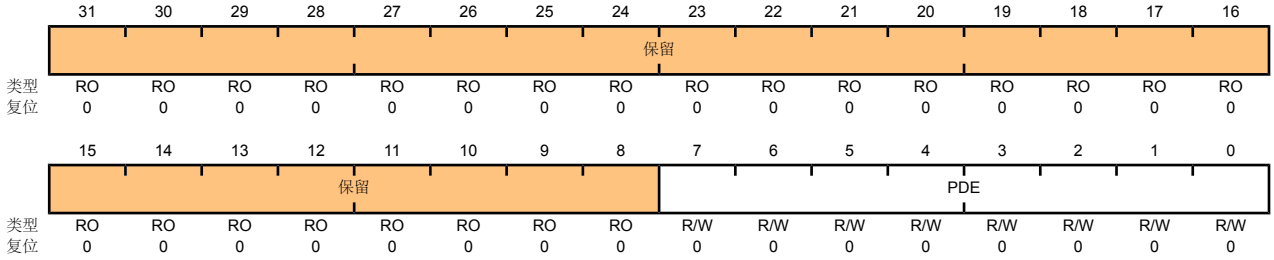
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PUE	R/W	-	端口弱上拉使能 向**GPIOPDR[n]**写1会清零相应的**GPIOPUR[n]**使能位。这种修改会在写操作之后的第二个时钟周期生效。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOAFSEL**, **GPIOPUR** 和 **GPIODEN** 寄存器默认的复位值都是0x0000.0000。这5个管脚默认为JTAG/SWD功能。正因为这样，对于GPIO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 138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



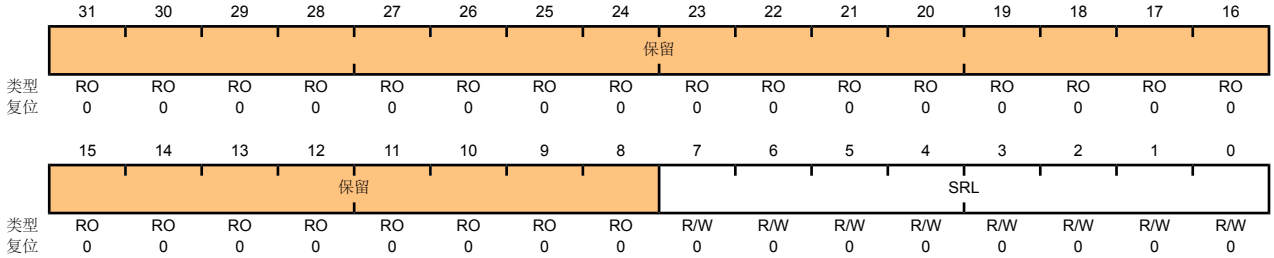
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PDE	R/W	0x00	端口弱下拉使能 向**GPIOPUR[n]**写1会清零相应的**GPIOPDR[n]**使能位。这种修改会在写操作之后的第二个时钟周期生效。

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 136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



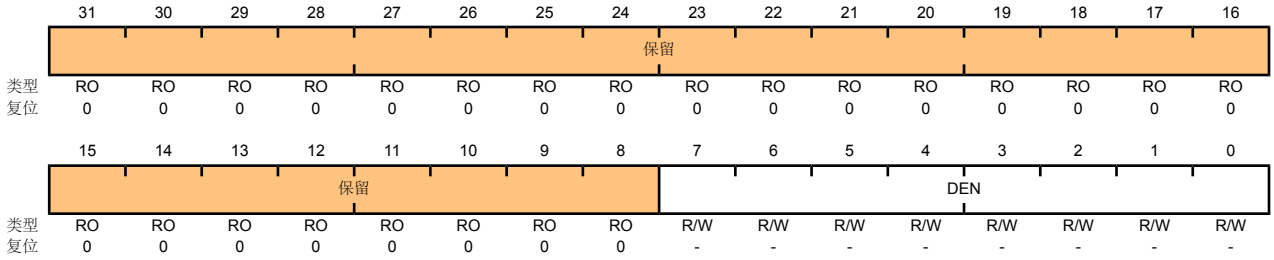
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	SRL	R/W	0	斜率限制使能（仅为8-mA驱动） 0: 斜率控制被禁能。 1: 斜率控制被使能。

寄存器18: GPIO 数字使能（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	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	DEN	R/W	-	数字使能 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 143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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LOCK															
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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LOCK															
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	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
31:0	LOCK	R/W	0x00000001	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: 锁定的：0x00000001 解锁的：0x00000000

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。

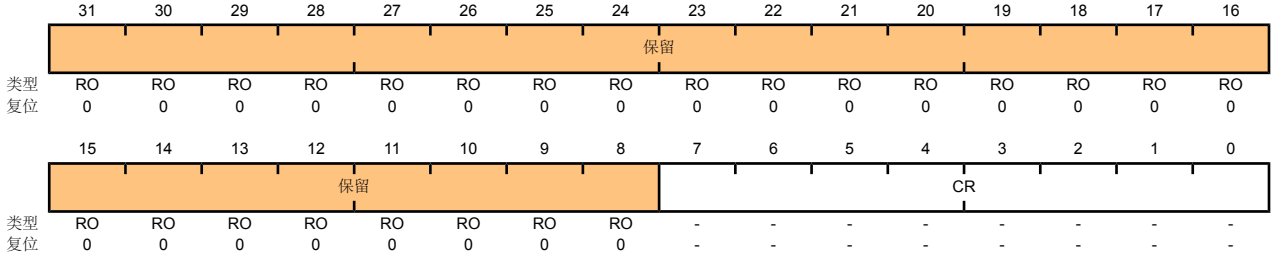
GPIOCR 寄存器的内容只有在 **GPIOLOCK** 寄存器解锁时才能被修改。如果**GPIOLOCK**寄存器被锁定，那么写 GPIOCR 寄存器将被忽略。

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-



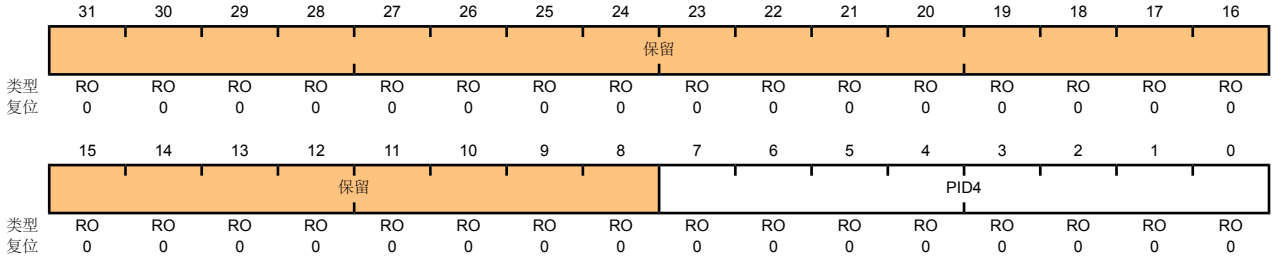
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CR	-	-	GPIO 确认（commit） 在按位（bit-wise）基础上，任何设置的位允许相应的GPIOAFSEL位被设为其备用功能。 注意: 对于除5个JTAG/SWD管脚（PB7 和 PC[3:0]）之外的所有GPIO管脚，**GPIOCR**寄存器默认的寄存器类型都是RO。这5个管脚是当前仅受**GPIOCR**寄存器保护的GPIO。正因为这样，对于GPIO 端口B7 和 GPIO 端口C[3:0]的寄存器类型是R/W。 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0x0x0000.0000



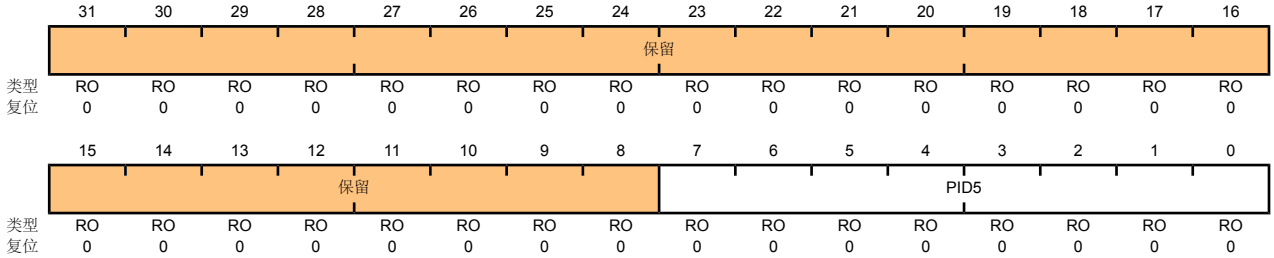
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	GPIO 外设ID寄存器 [7:0]

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0x0x0000.0000



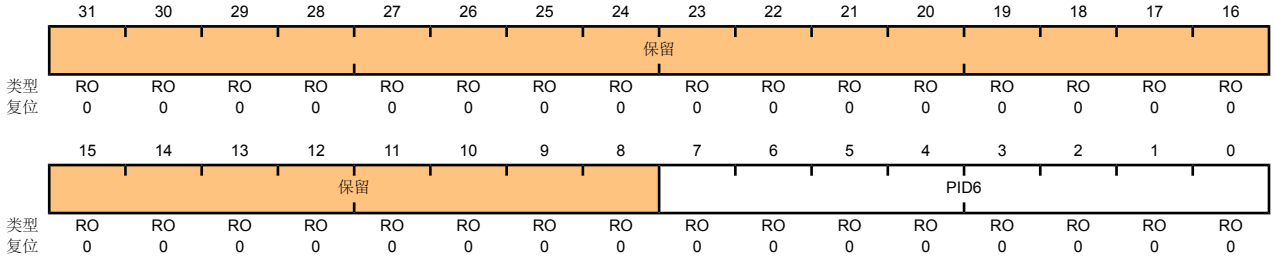
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	GPIO 外设ID寄存器 [15:8]

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0x0x0000.0000



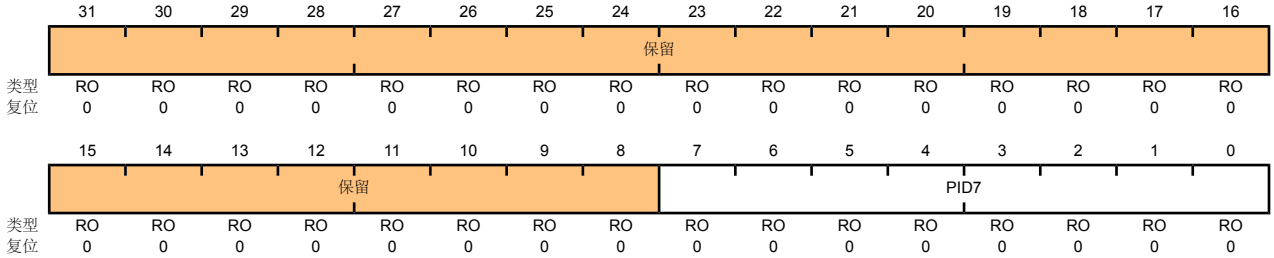
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	GPIO 外设ID寄存器 [23:16]

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0x0x0000.0000



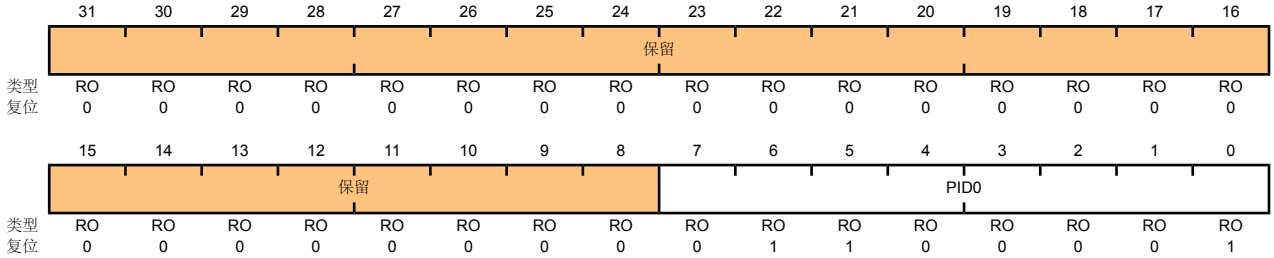
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	GPIO 外设ID寄存器 [31:24]

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0x0x0000.0061



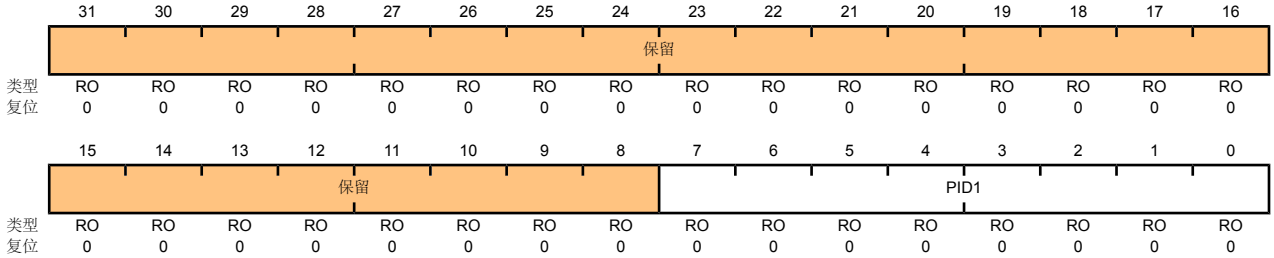
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x61	GPIO 外设ID寄存器 [7:0] 可被软件用来标识该外设是否存在。

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0x0x0000.0000



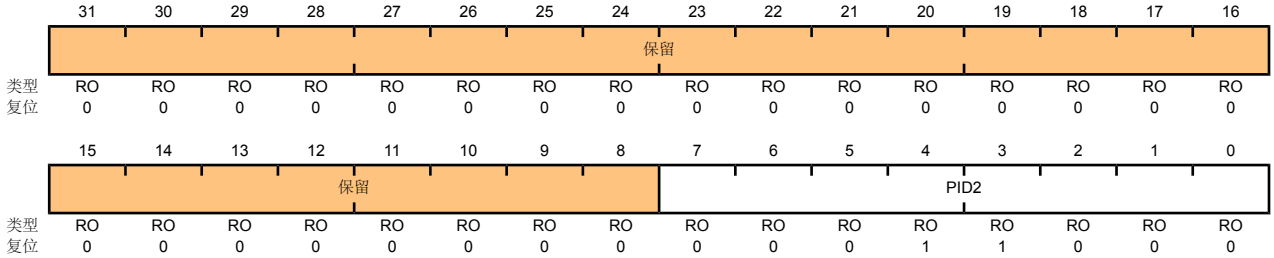
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	GPIO 外设ID寄存器 [15:8] 可被软件用来标识该外设是否存在。

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0x0x0000.0018



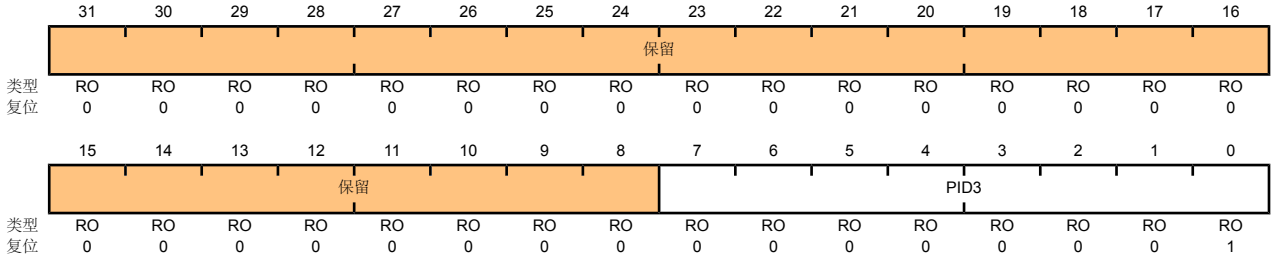
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	GPIO 外设ID寄存器 [23:16] 可被软件用来标识该外设是否存在。

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0x0x0000.0001



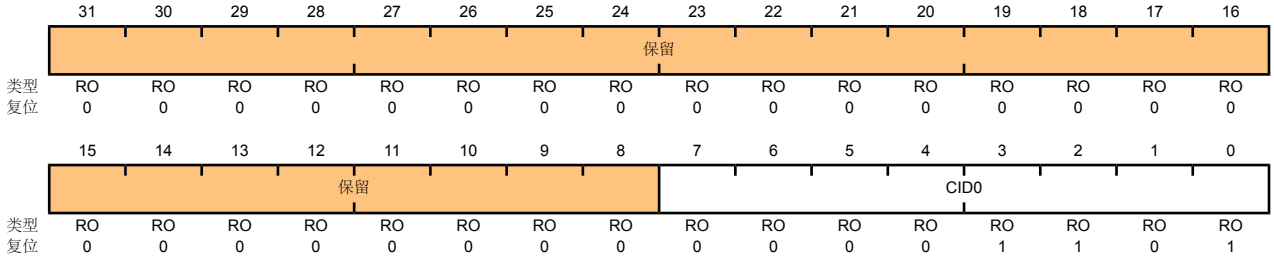
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	GPIO 外设ID寄存器 [31:24] 可被软件用来标识该外设是否存在。

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0x0x0000.000D



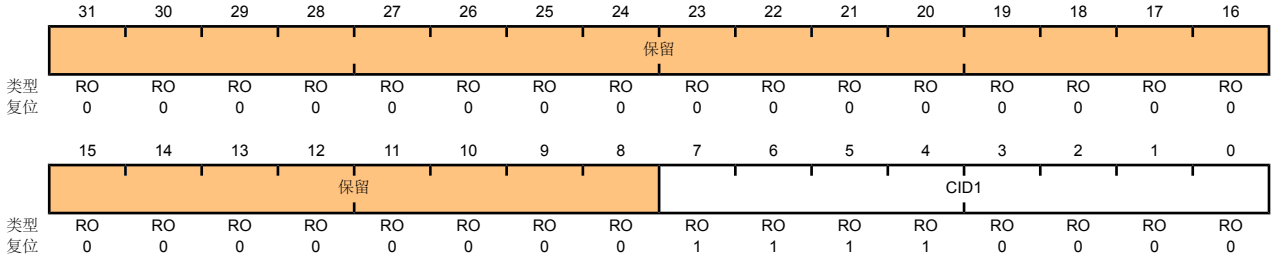
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	GPIO PrimeCell ID寄存器 [7:0] 向软件提供一个标准的交叉外设标识系统。

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0x0x0000.00F0



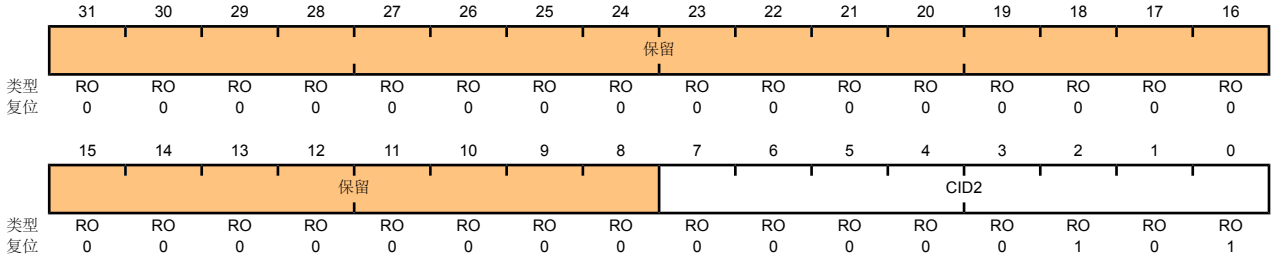
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	GPIO PrimeCell ID寄存器 [15:8] 向软件提供一个标准的交叉外设标识系统。

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0x0x0000.0005



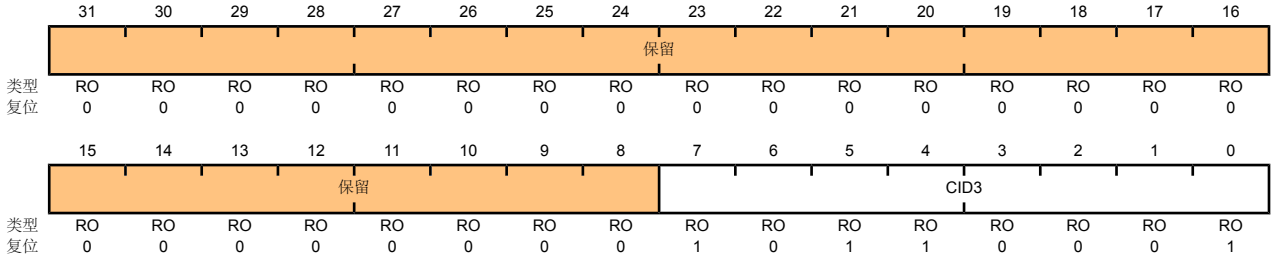
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	GPIO PrimeCell ID寄存器 [23:16] 向软件提供一个标准的交叉外设标识系统。

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0x0x0000.00B1



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	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。

9 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 3个 GPTM 模块 (定时器0, 定时器1和定时器 2)。每个GPTM模块包含两个16位的定时/计数器（称作TimerA和TimerB）。用户可以将它们配置成独立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1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。

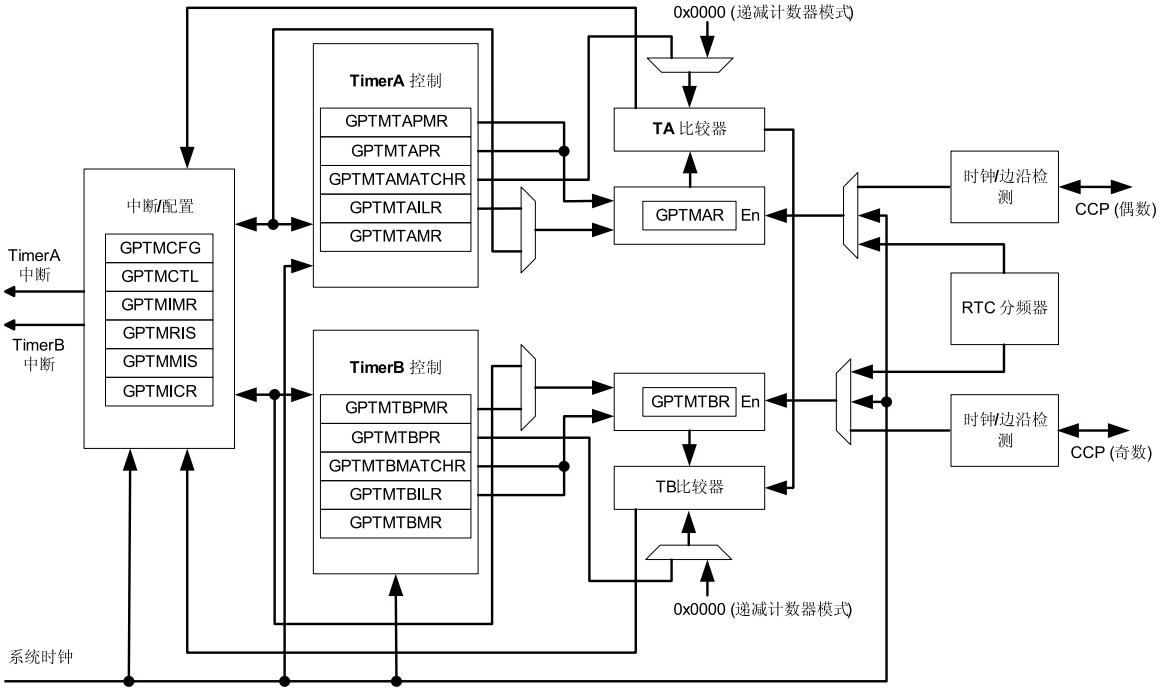
通用定时器模块是 Stellaris® 微控制器的一个定时资源。其它定时器资源还包括系统定时器 (SysTick) (见 “系统定时器 (SysTick)” 在 31页)。

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。

9.1 结构图

图 **9-1. GPTM**模块的结构图



9.2 功能描述

每个GPTM模块的主要元件包括两个自由运行的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 167页)、**GPTM TimerA** 模式 **(GPTMTAMR)** 寄存器 (见 168页)和 **GPTM TimerB** 模式 **(GPTMTBMR)** 寄存器 (见 169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。

9.2.1 GPTM 复位条件

GPTM模块复位后处于未激活状态，所有控制寄存器均被清零，同时进入默认状态。计数器TimerA和TimerB连同与它们对应的装载寄存器：**GPTM TimerA** 间隔装载（**GPTMTAILR**） 寄存器（见 176页） 和 **GPTM TimerB** 间隔装载 （**GPTMTBILR**）寄存器（见 177页）一起初始化为0xFFFF。预分频计数器：**GPTM TimerA** 预分频**(GPTMTAPR)** 寄存器 (见 180页) 和**GPTM TimerB** 预分频 **(GPTMTBPR)** 寄存器 (见 181页)初始化为0x00。

9.2.2 32-位定时器工作模式

注意： 编号为奇数和偶数的CCP管脚均用于16位模式。仅编号为偶数的CCP管脚用于32位模式。

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

通过向**GPTM** 配置 **(GPTMCFG)** 寄存器写入0（单次触发/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。 在这两种配置中，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 176页
- **GPTM TimerB** 间隔装载（**GPTMTBILR**） 寄存器 [15:0]，见 177页
- **GPTM TimerA**（**GPTMTAR**） 寄存器 [15:0]，见 184页
- **GPTM TimerB** （**GPTMTBR**） 寄存器 [15:0]，见 185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]
```

9.2.2.1 32-位单次触发/周期定时器模式

在32位单次触发和周期定时器模式中，TimerA和TimerB寄存器连在一起被配置为32位递减计数器。然后根据写入**GPTM TimerA**模式 **(GPTMTAMR)**寄存器 (见 168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 170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 173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 175页)执行写操作将其清零。如果**GPTM** 中断屏蔽 **(GPTIMR)** 寄存器 (见 172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 174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。

9.2.2.2 32-位实时时钟定时器模式

在实时时钟（RTC）模式中，TimerA和TimerB寄存器连在一起被配置为32位递增计数器。在首次选择RTC模式时，计数器装载的值为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 178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。

9.2.3 16-位定时器的工作模式

通过向**GPTM**配置 **(GPTMCFG)** 寄存器 (见 167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。

9.2.3.1 16-位单次触发/周期定时器模式

在16位单次触发/周期定时器模式中，定时器被配置为带可选的8位预分频器的16位递减计数器，预分频器可有效地将定时器的计数范围扩大到24位。选择单次触发模式还是周期模式由写入**GPTMTnMR**寄存器中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25-MHz （时钟周期Tc=20 ns）作为标准进行计算。

表 **9-1.** 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	2.6214	mS
00000001	2	5.2428	mS
00000010	3	27.8642	mS
----------	--	--	--
11111100	254	665.8458	mS
11111110	255	668.4672	mS
11111111	256	671.0886	mS

a. Tc表示时钟周期。

9.2.3.2 16-位输入边沿计数模式

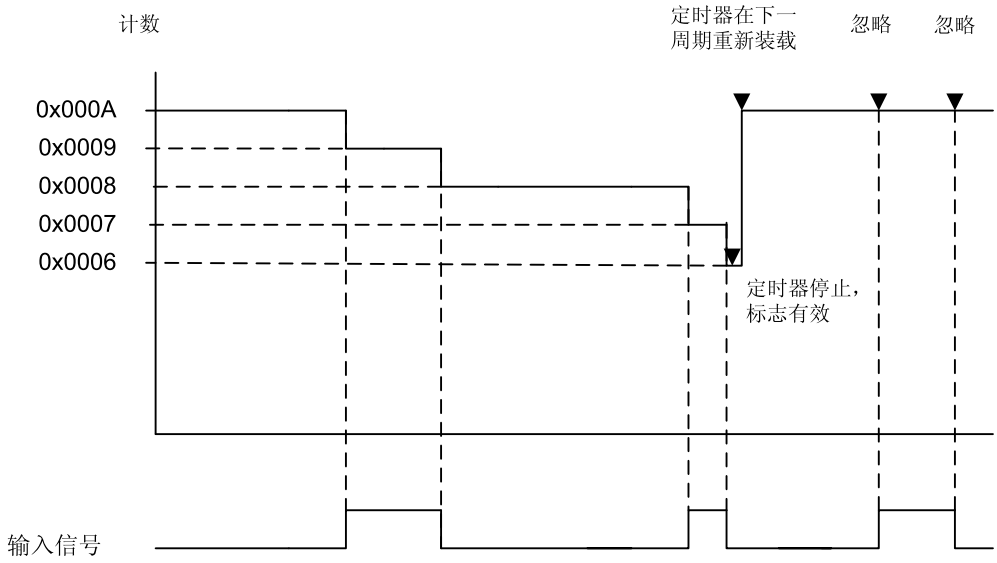
在边沿计数模式中，定时器被配置为能够捕获3种事件类型的递减计数器，这3种事件类型为上升沿、下降沿、或上升/下降沿。为了把定时器设置为边沿计数模式，**GPTMTnMR**寄存器的TnCMR 位必须设为0。定时器计数时所采用的边沿类型由**GPTMCTL**寄存器的TnEVENT位域决定。在初始化过程中，需对**GPTM Timern** 匹配 **(GPTMTnMATCHR)**寄存器进行配置，以便**GPTMTnILR**寄存器和**GPTMTnMATCHR**寄存器之间的差值等于必须计算的边沿事件的数目。

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 9-2 在 160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 **9-2. 16**位输入边沿计数模式实例



9.2.3.3 16-位输入边沿定时模式

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。

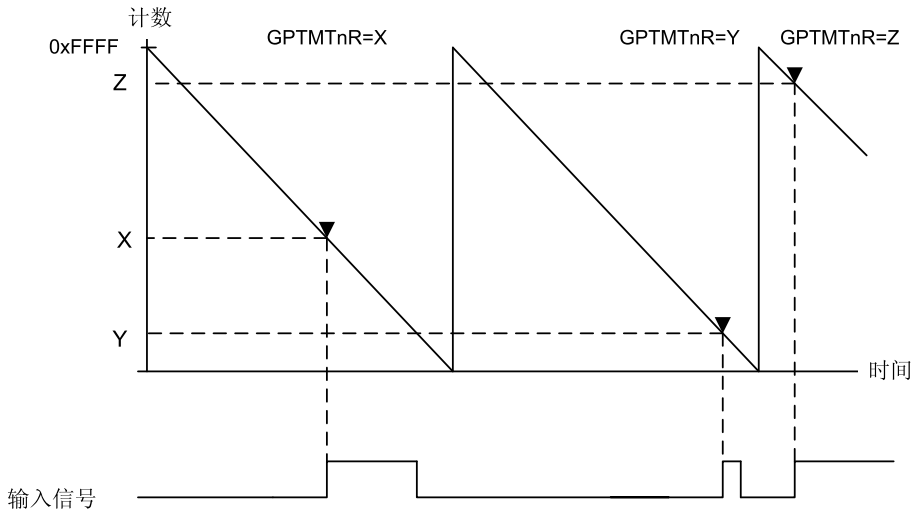
在软件写**GPTMCTL**寄存器的TnEN 位时，定时器使能并用于事件捕获。在检测到所选的输入事件时，从**GPTMTnR**寄存器中捕获**Tn**计数器的当前值，且该值可通过控制器来读取。 然后GPTM让CnERIS位有效（如果中断没有被屏蔽，则也让CnEMIS位有效）。

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9-3在161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 **9-3. 16**位输入边沿定时模式实例



9.2.3.4 16-位**PWM**模式

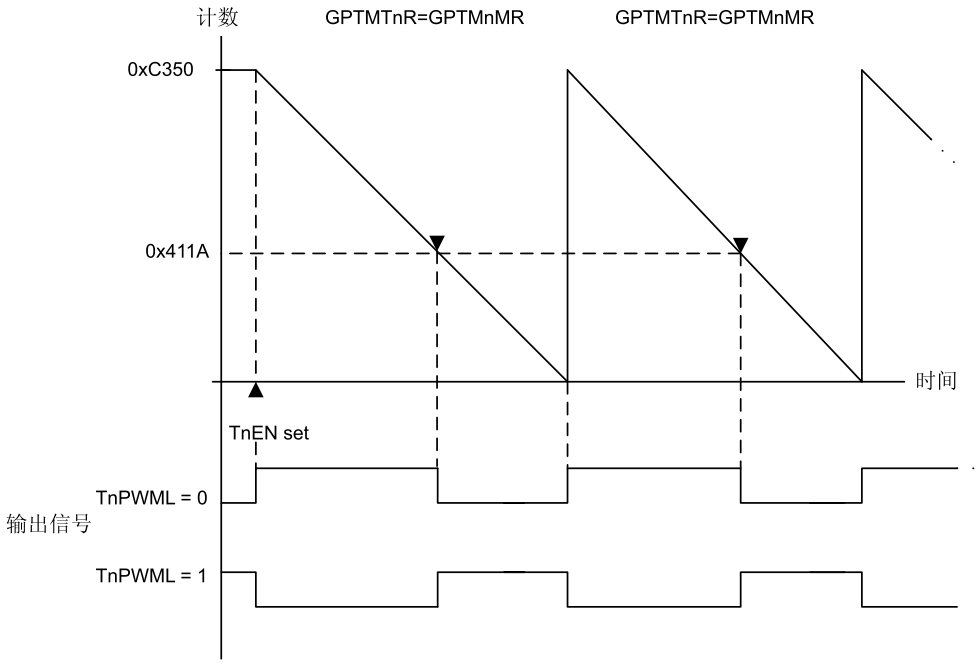
GPTM支持简单的PWM生成模式。在PWM模式中，定时器配置为递减计数器，初值由**GPTMTnILR**定义。通过将**GPTMTnMR**寄存器的TnAMS位置为0x1、TnCMR位置为0x0、TnMR位域置为0x2来使能PWM模式。

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9-4在162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 **9-4. 16-**位**PWM**模式实例



9.3 初始化和配置

在使用通用定时器时，外设时钟必须使能，该操作通过将**RCGC1**寄存器中的 TIMER0、 TIMER1和 TIMER2 位置位来实现。

针对每种支持的定时器模式，本节提供了模块的初始化以及配置示例。

9.3.1 32-位单次触发/周期定时器模式

将GPTM配置为32位单次触发和周期模式的步骤如下:

1. 确保定时器在发生任何变化之前先禁止（将**GPTMCTL**寄存器的TAEN位清零）。
2. 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 162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。

9.3.2 32-位实时时钟（RTC）模式

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

1. 确保定时器在发生任何变化之前先禁止（TAEN位清零）。
2. 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。

9.3.3 16-位单次触发/周期定时器模式

将定时器配置为16位单次触发和周期模式的步骤如下:

1. 确保定时器在发生任何变化之前先禁止（TnEN位清零）。
2. 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 163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。

9.3.4 16-位输入边沿计数模式

将定时器配置为输入边沿计数模式的步骤如下:

1. 确保定时器在发生任何变化之前先禁止（TnEN位清零）。

2. 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 164页到step 9 在 164页才能重新使能定时器。

9.3.5 16-位输入边沿定时模式

将定时器配置为输入边沿定时模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（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。

9.3.6 16-位**PWM**模式

将定时器配置为PWM模式的步骤如下：

1. 确保定时器在发生任何变化之前先禁止（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. 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。

9.4 寄存器映射

表 9-2 在 165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：

- 定时器0: 0x4003.00000x4003.0000
- 定时器1: 0x4003.10000x4003.1000
- 定时器2: 0x4003.20000x4003.2000

表 **9-2.** 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	0x0x0000.0000	GPTM 配置	167
0x004	GPTMTAMR	R/W	0x0x0000.0000	GPTM TimerA 模式	168
0x008	GPTMTBMR	R/W	0x0x0000.0000	GPTM TimerB 模式	169
0x00C	GPTMCTL	R/W	0x0x0000.0000	GPTM 控制	170
0x018	GPTMIMR	R/W	0x0x0000.0000	GPTM 中断屏蔽	172
0x01C	GPTMRIS	RO	0x0x0000.0000	GPTM 原始中断状态	173
0x020	GPTMMIS	RO	0x0x0000.0000	GPTM屏蔽后的中断状态	174
0x024	GPTMICR	W1C	0x0x0000.0000	GPTM中断清零	175
0x028	GPTMTAILR	R/W	0x0000.FFFF（16位模式） 和 0xFFFF.FFFF（32位模式）	GPTM TimerA间隔装载	176
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB间隔装载	177
0x030		R/W	0x0000.FFFF（16位模式） 和 0xFFFF.FFFF（32位模式）		178
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB匹配	179
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA预分频	180
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB预分频	181
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA预分频匹配	182
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB预分频匹配	183

偏移量	名称	类型	复位	描述	见页面
0x048	GPTMTAR	RO	0x0000.FFFF （16位模式） 和 0xFFFF.FFFF（32位模式）	GPTM TimerA	184
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	185

9.5 寄存器描述

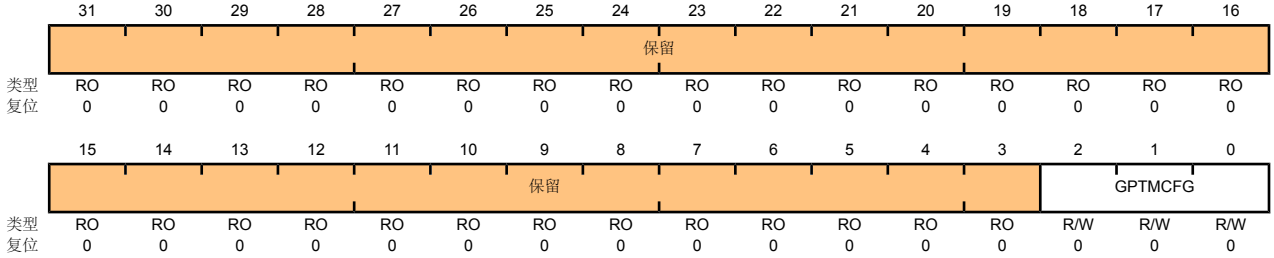
下文将按地址偏移量的数字顺序列出GPTM寄存器，并对它们进行描述。

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0x0x0000.0000



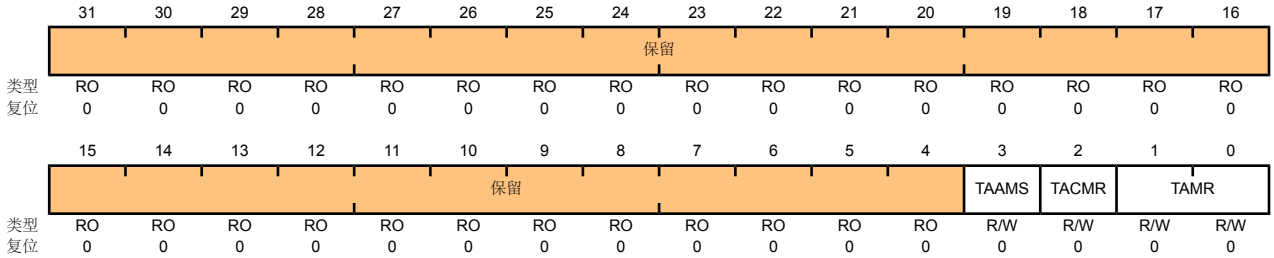
位/域	名称	类型	复位	描述
31:3	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
2:0	GPTMCFG	R/W	0	GPTM 配置 0x0: 32-位定时器配置 0x1: 32-位实时时钟 (RTC) 计数器配置 0x2:保留 0x3: 保留 0x4-0x7: 16-位定时器配置，其功能由**GPTMTAMR** 和 **GPTMTBMR**的 1:0 位来控制。

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GPTM为16位PWM模式中时，TAAMS位设为0x1，TACMR位设为0x0，TAMR位域设为0x2。

GPTM TimerA 模式 (GPTMTAMR)

偏移量0x004
类型R/W, 复位0x0x0000.0000



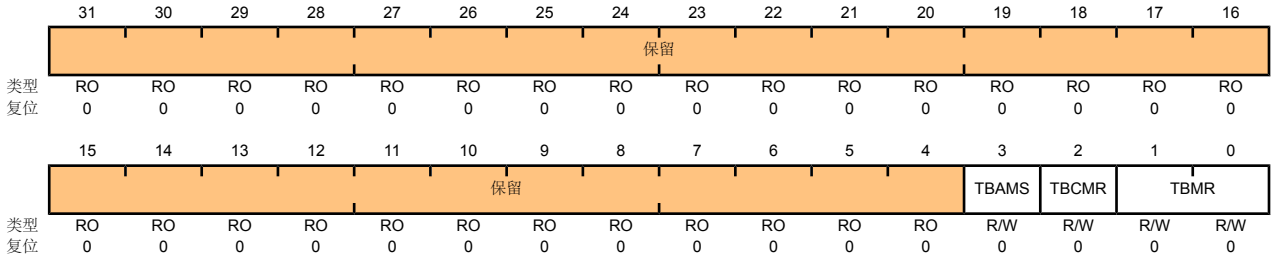
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TAAMS	R/W	0	GPTM TimerA 备用模式选择 0: 使能捕获模式。 1: 使能PWM模式。 注意: 要使能PWM模式，必须清零 TACMR 位并将 TAMR 位域置为0x2。
2	TACMR	R/W	0	GPTM TimerA 捕获模式 0: 边沿-计数模式 1: 边沿-定时模式
1:0	TAMR	R/W	0	GPTM TimerA 模式 0x0: 保留 0x1: 单次触发定时器模式 0x2: 周期定时器模式 0x3: 捕获模式 定时器模式基于**GPTMCFG**寄存器（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 位域置为 0x2。

GPTM TimerB 模式 (GPTMTBMR)

偏移量0x008
类型R/W, 复位0x0x0000.0000



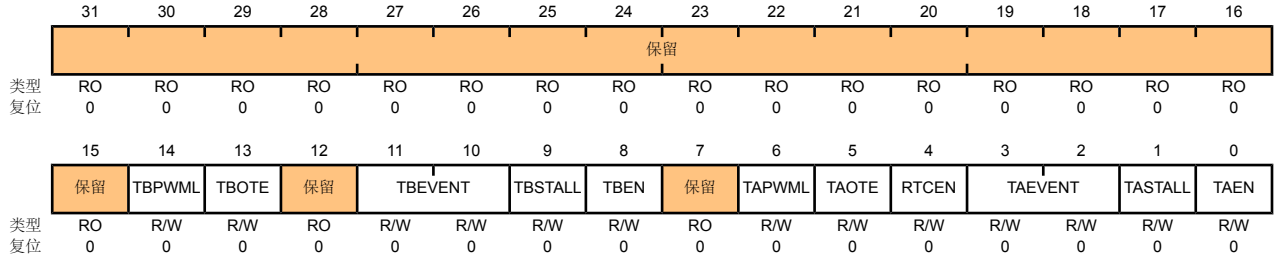
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TBAMS	R/W	0	GPTM TimerB 备用模式选择 0: 使能捕获模式。 1: 使能PWM模式。 注意: 要使能 PWM 模式，你必须清零 TBCMR 位，并将 TBMR 位域置为 0x2。
2	TBCMR	R/W	0	GPTM TimerB 捕获模式 0: 边沿-计数模式 1: 边沿-定时模式
1:0	TBMR	R/W	0	GPTM TimerB 模式 0x0: 保留 0x1: 单次触发定时器模式 0x2: 周期定时器模式 0x3: 捕获模式 定时器模式基于**GPTMCFG**寄存器的 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，并使能其它诸如定时器停止和输出触发信号等特性。

GPTM 控制 (GPTMCTL)

偏移量0x00C
类型R/W, 复位0x0x0000.0000



位/域	名称	类型	复位	描述
31:15	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。
14	TBPWML	R/W	0	GPTM TimerB PWM 输出电平 0: 输出不改变。 1: 输出反相。
13	TBOTE	R/W	0	GPTM TimerB 输出触发使能标志 0: TimerB 输出触发禁止。 1: TimerB 输出触发使能。
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	0	GPTM TimerB 事件模式 00: 上升沿。 01: 下降沿。 10: 保留 11: 上升/下降沿
9	TBSTALL	R/W	0	GPTM TimerB 停止使能标志 0: 禁止TimerB 停止。 1: 使能TimerB 停止。
8	TBEN	R/W	0	GPTM TimerB使能标志 0: TimerB 禁止。 1: TimerB使能并开始计数，或根据**GPTMCFG**寄存器使能捕获逻辑。
7	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

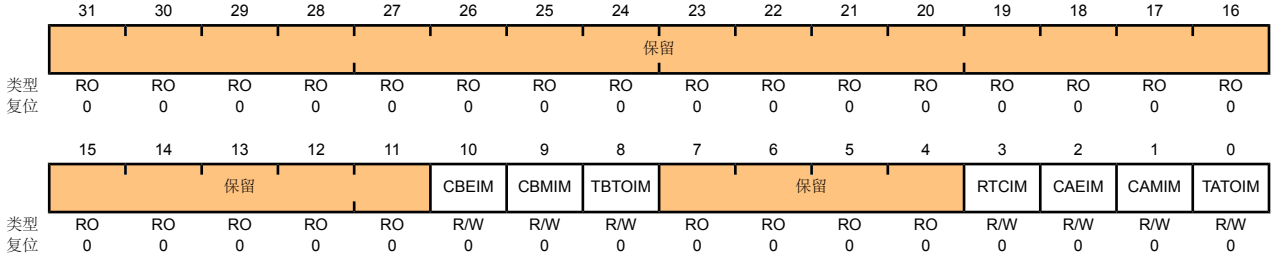
位/域	名称	类型	复位	描述
6	TAPWML	R/W	0	GPTM TimerA的PWM输出电平 0: 输出不改变。 1: 输出反相。
5	TAOTE	R/W	0	GPTM TimerA 的输出触发使能标志 0: TimerA的输出触发禁止 1: TimerA的输出触发使能。
4	RTCEN	R/W	0	GPTM RTC 使能标志 0: RTC计数禁止。 1: RTC计数使能。
3:2	TAEVENT	R/W	0	GPTM TimerA 事件模式 00: 上升沿。 01: 下降沿。 10: 保留 11: 上升/下降沿
1	TASTALL	R/W	0	GPTM TimerA 停止使能标志 0: 禁止TimerA停止。 1: 使能TimerA 停止。
0	TAEN	R/W	0	GPTM TimerA使能标志 0: TimerA禁止。 1: TimerA使能并开始计数，或根据**GPTMCFG**寄存器使能捕获逻辑。

寄存器5: GPTM 中断屏蔽（GPTMIMR），偏移量 0x018

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

GPTM 中断屏蔽 (GPTMIMR)

偏移量0x018
类型R/W, 复位0x0x0000.0000



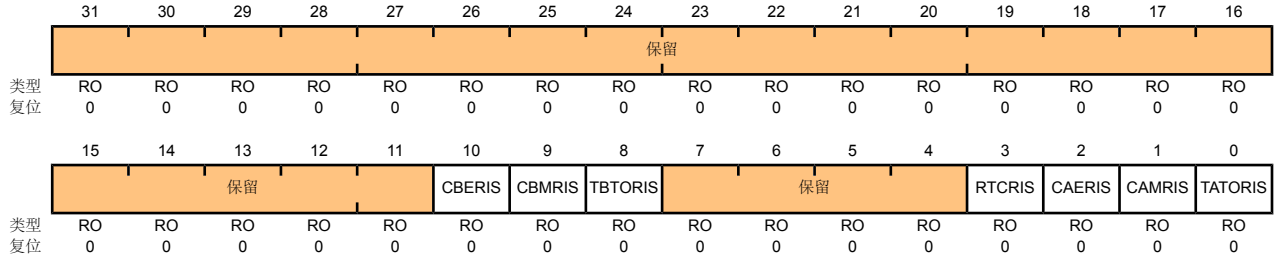
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBEIM	R/W	0	GPTM CaptureB事件的中断屏蔽标志 0: 中断禁止。 1: 中断使能。
9	CBMIM	R/W	0	GPTM CaptureB匹配的中断屏蔽标志 0: 中断禁止。 1: 中断使能。
8	TBTOIM	R/W	0	GPTM TimerB超时的中断屏蔽标志 0: 中断禁止。 1: 中断使能。
7:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	RTCIM	R/W	0	GPTM RTC的中断屏蔽标志 0: 中断禁止。 1: 中断使能。
2	CAEIM	R/W	0	GPTM CaptureA事件的中断屏蔽标志 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 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 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0x0x0000.0000



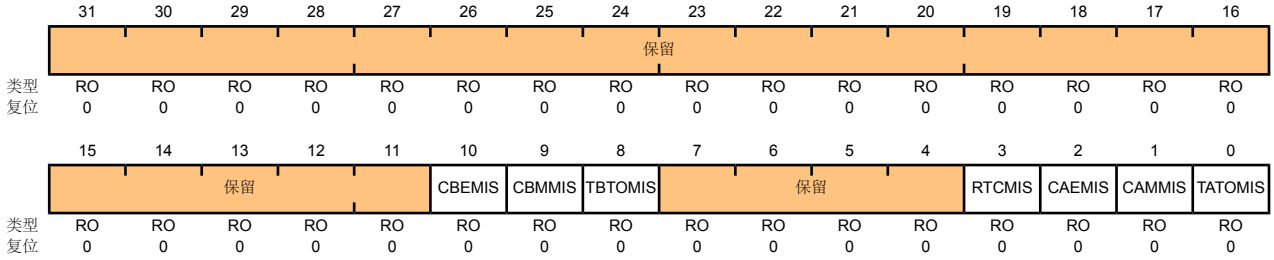
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBERIS	RO	0	GPTM CaptureB事件的原始中断标志 该位表示屏蔽之前CaptureB事件的中断状态。
9	CBMRIS	RO	0	GPTM CaptureB匹配的原始中断标志 该位表示屏蔽之前CaptureB匹配的中断状态。
8	TBTORIS	RO	0	GPTM TimerB超时的原始中断标志 该位表示屏蔽之前TimerB超时的中断状态。
7:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	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0x0x0000.0000



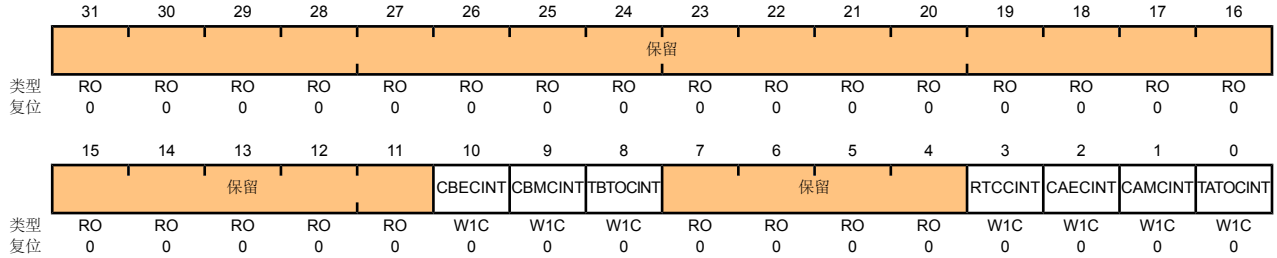
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBEMIS	RO	0	GPTM CaptureB事件屏蔽后的中断标志 该位表示屏蔽之后CaptureB事件的中断状态。
9	CBMMIS	RO	0	GPTM CaptureB匹配屏蔽后的中断标志 该位表示屏蔽之后CaptureB匹配的中断状态。
8	TBTOMIS	RO	0	GPTM TimerB超时屏蔽后的中断标志 该位表示屏蔽之后TimerB超时的中断状态。
7:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	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0x0x0000.0000



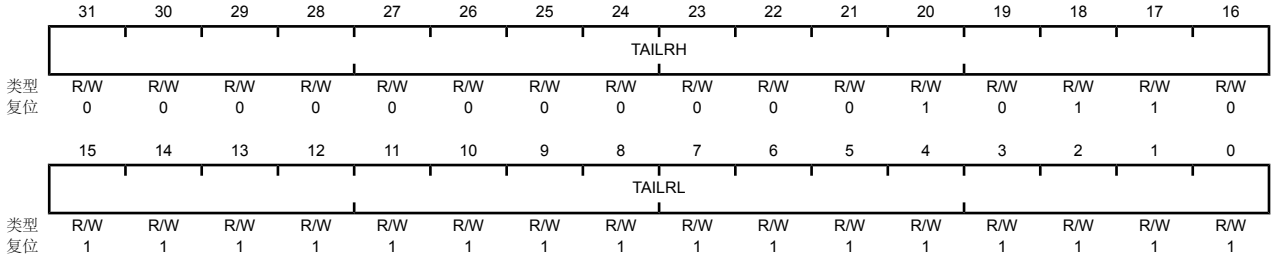
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	CBECINT	W1C	0	GPTM CaptureB事件的中断清零标志 0: 中断不受影响。 1: 中断清零。
9	CBMCINT	W1C	0	GPTM CaptureB匹配的中断清零标志 0: 中断不受影响。 1: 中断清零。
8	TBTOCINT	W1C	0	GPTM TimerB超时的中断清零标志 0: 中断不受影响。 1: 中断清零。
7:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－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

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

该寄存器用来将起始计数值装入定时器。当GPTM配置为其中一种32位模式时，**GPTMTAILR**作为32位寄存器使用（高16位对应于**GPTM TimerB** 间隔装载（**GPTMTBILR**）寄存器的值）。在16位模式中，该寄存器的高16位读作0，不影响**GPTMTBILR**寄存器的状态。

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）



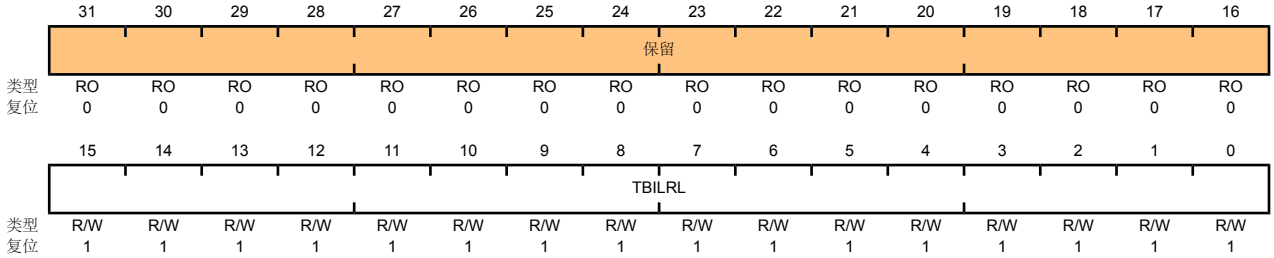
位/域	名称	类型	复位	描述
31:16	TAILRH	R/W	0xFFFF （32位模式） 0x0000 （16位模式）	GPTM TimerA间隔装载寄存器的高字节 当通过**GPTMCFG**寄存器配置为32位模式时，**GPTM TimerB**间隔装载（**GPTMTBILR**）寄存器通过写操作来装载该值。读操作时返回**GPTMTBILR**的当前值。 在16位模式中，该位域在读操作时返回0，不影响**GPTMTBILR**寄存器的状态。
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA间隔装载寄存器的低字节 在16位和32位模式中，对该位域执行写操作将装载TimerA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



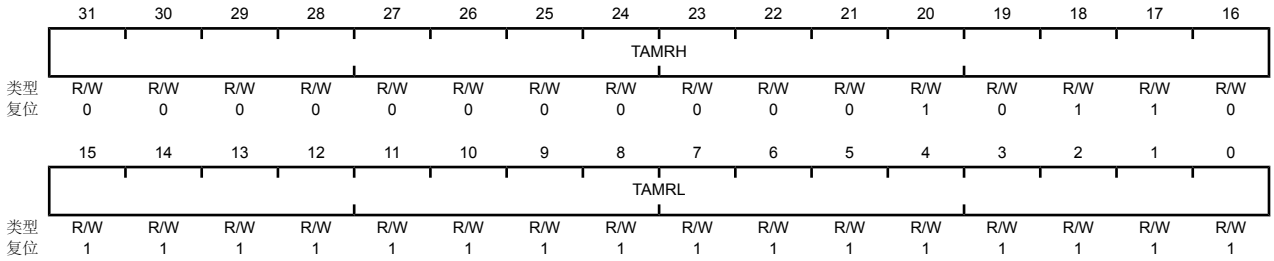
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB间隔装载寄存器 当GPTM没有被配置为32位定时器时，对该位域执行写操作将更新**GPTMTBILR**的值。在32位模式中，写操作被忽略，读操作返回**GPTMTBILR**的当前值。

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、16位PWM和输入边沿计数模式。

()

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



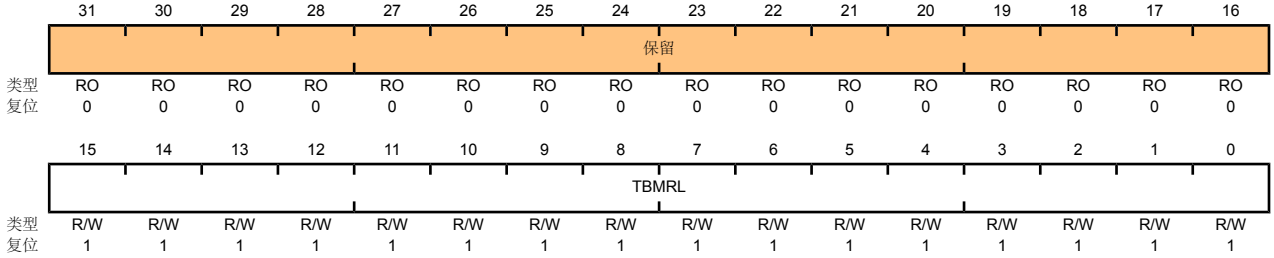
位/域	名称	类型	复位	描述
31:16	TAMRH	R/W	0xFFFF (32-位模式) 0x0000 (16-位模式)	GPTM TimerA匹配寄存器的高字节 当通过**GPTMCFG**寄存器配置为32位实时时钟（RTC）模式时，该值与**GPTMTAR**的高半字进行比较，来确定匹配事件。 在16位模式中，对该位域的读操作返回0，不影响**GPTMTBMATCHR**寄存器的状态。
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA匹配寄存器的低字节 当通过**GPTMCFG**寄存器配置为32位实时时钟（RTC）模式时，该值与**GPTMTAR**的低半字进行比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



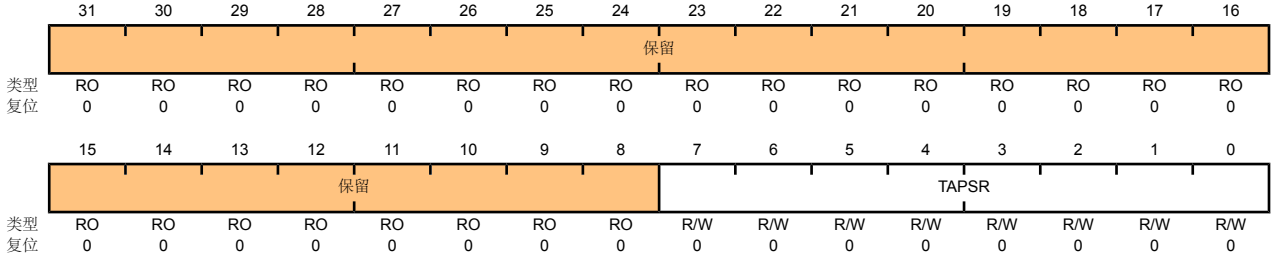
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB匹配寄存器的低字节 配置为PWM模式时，该值与**GPTMTBILR**一起，确定输出PWM信号的占空比。 配置为边沿计数模式时，该值与**GPTMTBILR**一起，确定需计数多少边沿事件数。总的边沿事件数等于**GPTMTBILR**与该值的差。

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



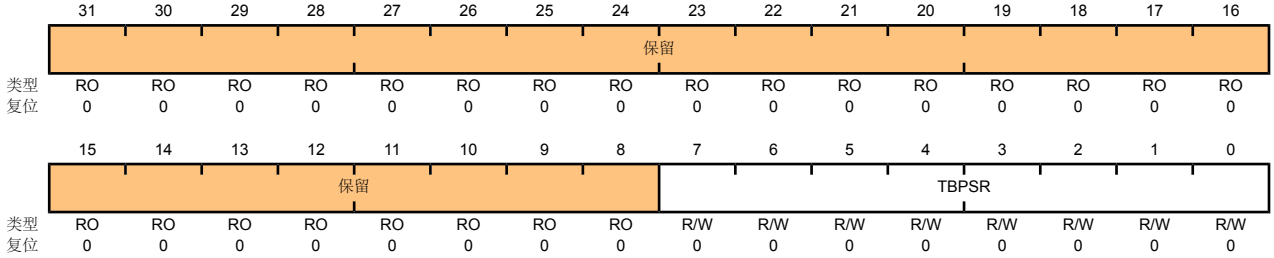
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TAPSR	R/W	0	GPTM TimerA 预分频 寄存器通过写操作载入该值。执行读操作将返回寄存器的当前值。 详细信息和实例请参考 表 9-1 在 159页 。

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



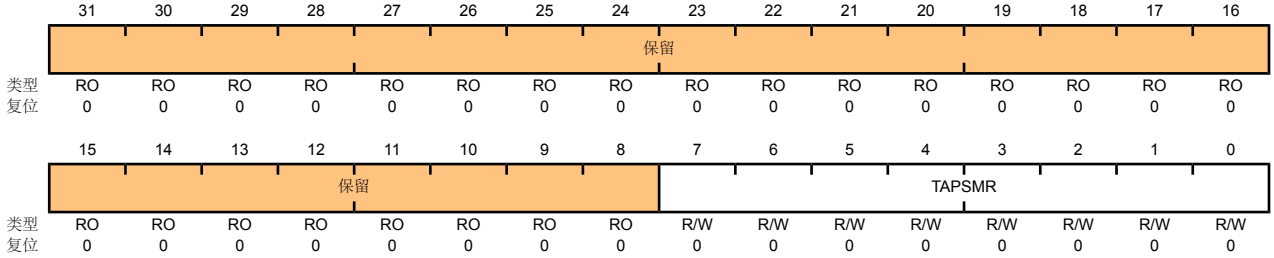
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TBPSR	R/W	0	GPTM TimerB预分频 寄存器通过写操作载入该值。执行读操作将返回寄存器的当前值。 详细信息和实例请参考 表 9-1 在 159页 。

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



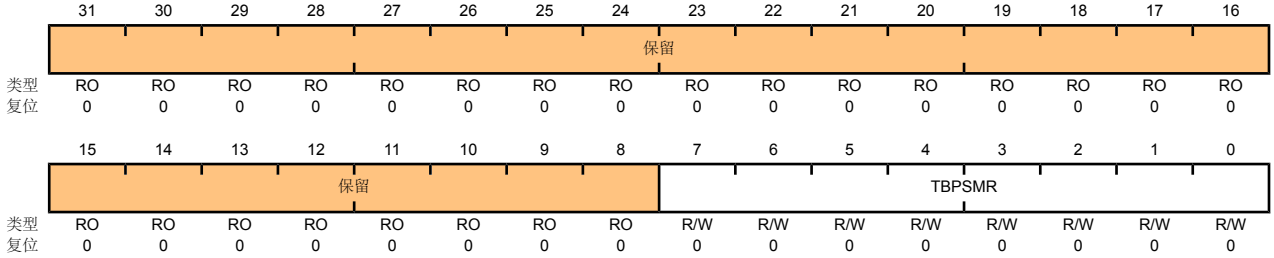
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TAPSMR	R/W	0	GPTM TimerA预分频匹配 该值与**GPTMTAMATCHR**一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

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

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



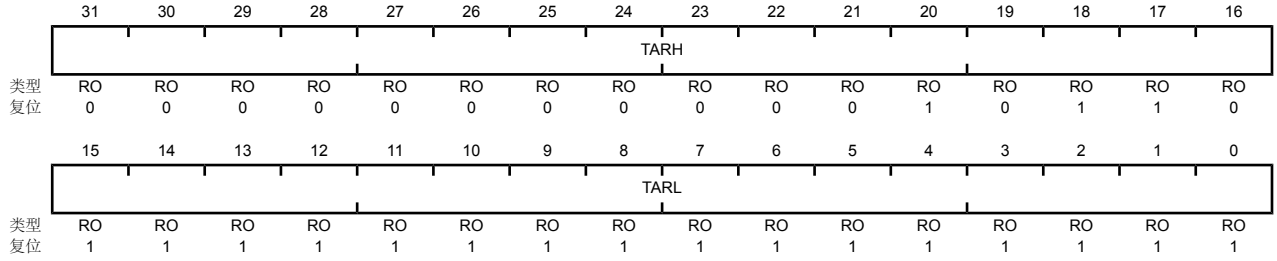
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	TBPSMR	R/W	0	GPTM TimerB预分频匹配 该值与**GPTMTBMATCHR**一起使用，以便在使用预分频器的情况下检测定时器匹配事件。

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

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

GPTM TimerA (GPTMTAR)

偏移量0x048
类型RO, 复位0x0000.FFFF （16位模式） 和 0xFFFF.FFFF（32位模式）



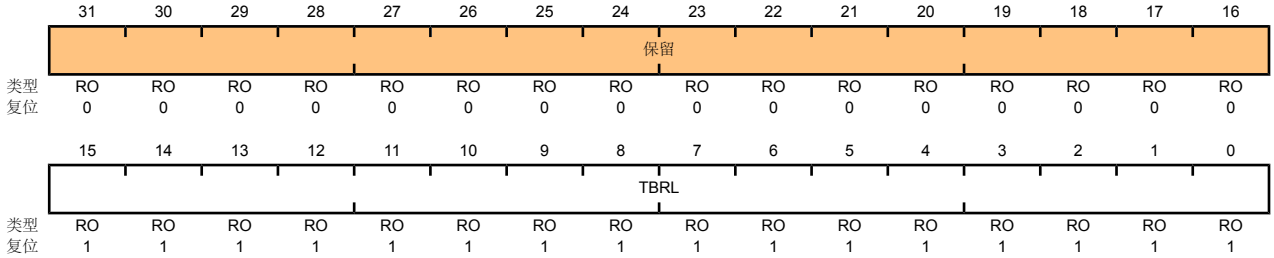
位/域	名称	类型	复位	描述
31:16	TARH	RO	0xFFFF （32位模式） 0x0000 （16位模式）	GPTM TimerA寄存器高字节 如果**GPTMCFG**配置为32位模式，则对该位域执行读操作将获得TimerB的值。如果**GPTMCFG**配置为16位模式，则读操作返回0。
15:0	TARL	RO	0xFFFF	GPTM TimerA寄存器高字节 读取该位域将返回**GPTM TimerA**计数寄存器的当前值，但输入边沿计数模式除外，在该模式中，读操作返回上一次边沿事件的时间戳（timestamp）。

寄存器18: GPTM TimerB（GPTMTBR），偏移量 0x04C

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

GPTM TimerB (GPTMTBR)

偏移量0x04C
类型RO, 复位0x0000.FFFF



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	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）。

10 看门狗定时器

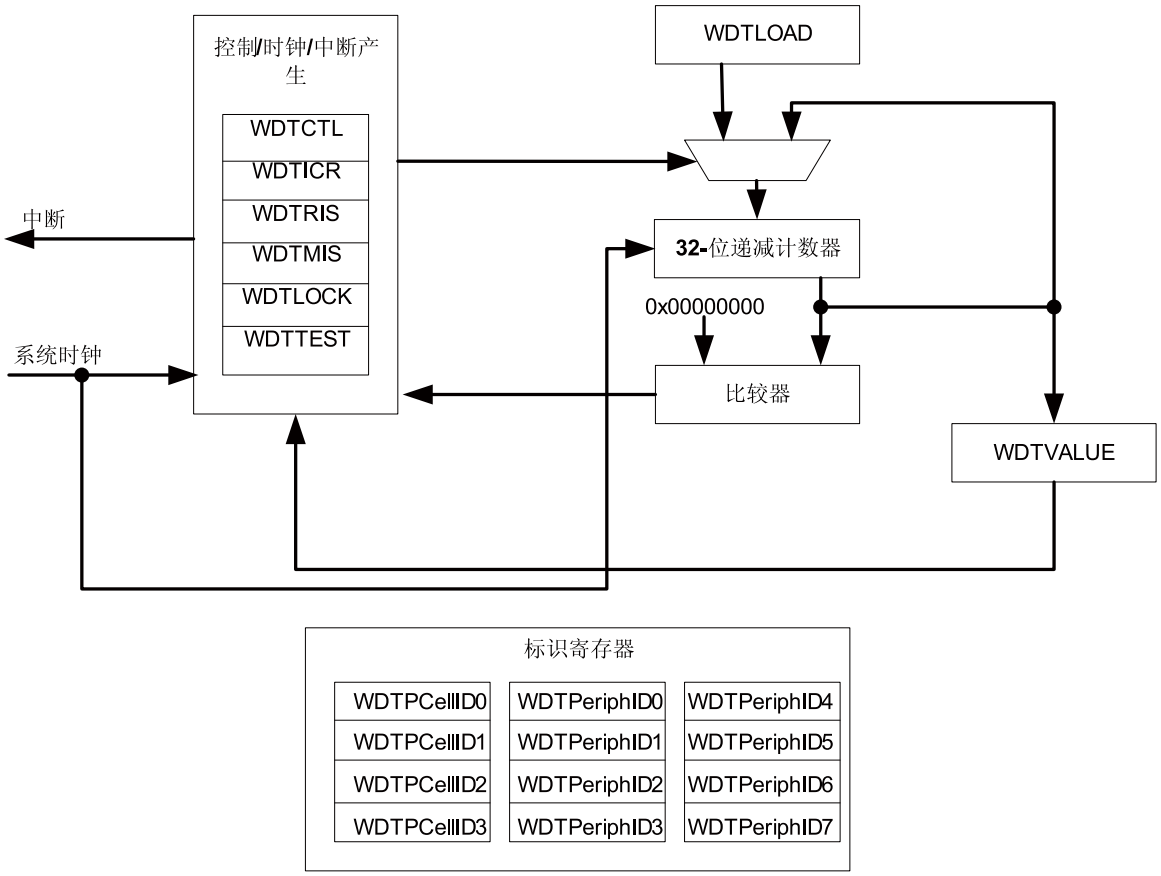
看门狗定时器在到达超时值时可能会产生不可屏蔽的中断（NMIs）或复位。当系统由于软件错误而无法响应或外部器件不是以期望的方式响应时，使用看门狗定时器可重新获得控制权。

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。

10.1 结构图

图 **10-1. WDT**模块的结构图



10.2 功能描述

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

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

在清除第一个超时中断之前，如果定时器的值再次递减为0，且复位信号已使能（通过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。

10.3 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。

10.4 寄存器映射

表 10-1 在 187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 **10-1.** 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	189
0x004	WDTVALUE	RO	0xFFFF.FFFF	看门狗值	190
0x008	WDTCTL	R/W	0x0000.0000	看门狗控制	191
0x00C	WDTICR	WO	-	看门狗中断清零	192
0x010	WDTRIS	RO	0x0000.0000	看门狗原始中断状态	193
0x014	WDTMIS	RO	0x0000.0000	看门狗屏蔽后的中断状态	194
0x418	WDTTEST	R/W	0x0000.0000	看门狗测试	195
0xC00	WDTLOCK	R/W	0x0000.0000	看门狗锁定	196
0xFD0	WDTPeriphID4	RO	0x0000.0000	看门狗外设标识 4	197
0xFD4	WDTPeriphID5	RO	0x0000.0000	看门狗外设标识 5	198
0xFD8	WDTPeriphID6	RO	0x0000.0000	看门狗外设标识 6	199
0xFDC	WDTPeriphID7	RO	0x0000.0000	看门狗外设标识 7	200
0xFE0	WDTPeriphID0	RO	0x0000.0005	看门狗外设标识0	201

偏移量	名称	类型	复位	描述	见页面
0xFE4	WDTPeriphID1	RO	0x0000.0018	看门狗外设标识1	202
0xFE8	WDTPeriphID2	RO	0x0000.0018	看门狗外设标识2	203
0xFEC	WDTPeriphID3	RO	0x0000.0001	看门狗外设标识3	204
0xFF0	WDTPCellID0	RO	0x0000.000D	看门狗PrimeCell标识 0	205
0xFF4	WDTPCellID1	RO	0x0000.00F0	看门狗PrimeCell标识1	206
0xFF8	WDTPCellID2	RO	0x0000.0005	看门狗PrimeCell标识2	207
0xFFC	WDTPCellID3	RO	0x0000.00B1	看门狗PrimeCell标识3	208

10.5 寄存器描述

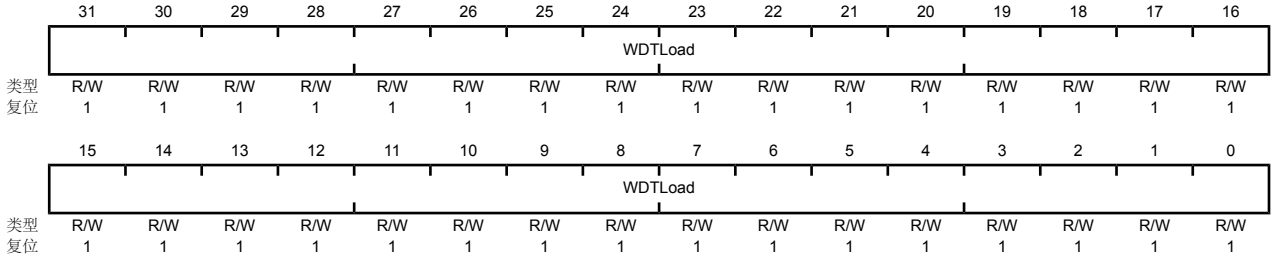
下文将按地址偏移量的数字顺序列出WDT寄存器，并对它们进行描述。

寄存器1: 看门狗装载 （**WDTLOAD**），偏移量 **0x000**

该寄存器存放的是供32位计数器使用的32位间隔值。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



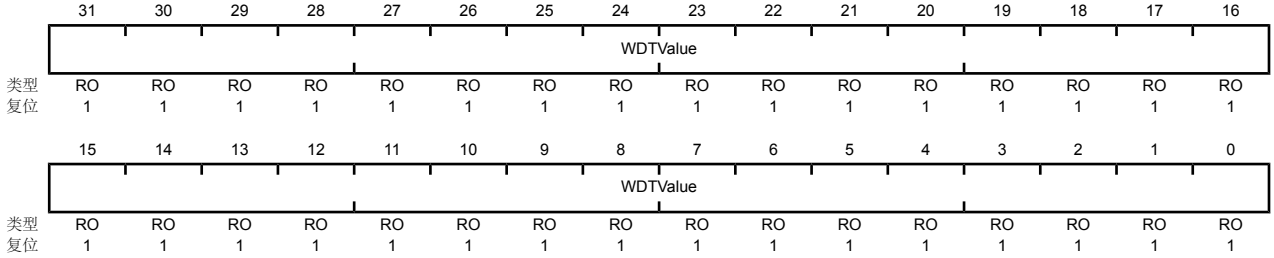
位/域	名称	类型	复位	描述
31:0	WDTLoad	R/W	0xFFFF.FFFF	看门狗装载值

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。

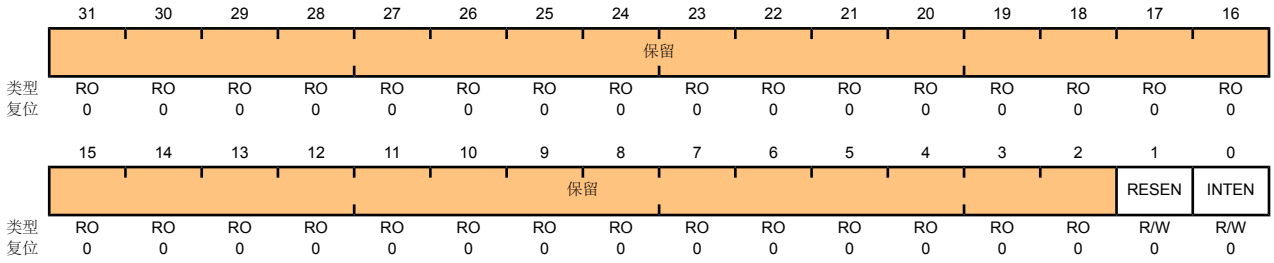
寄存器3: 看门狗控制（WDTCTL），偏移量 0x008

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



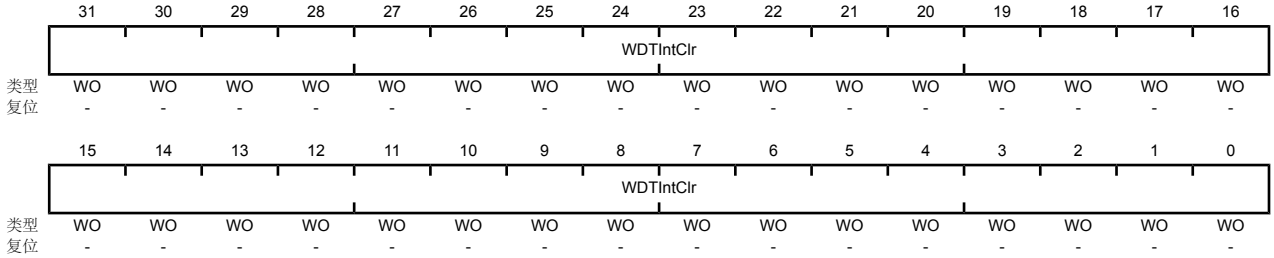
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	RESEN	R/W	0	看门狗复位使能标志 0: 禁止。 1: 使能看门狗模块复位输出。
0	INTEN	R/W	0	看门狗中断使能标志 0: 中断事件禁能（一旦该位被置位，则只能通过硬件复位来清零该位）。 1: 中断事件使能。一旦被使能，之后所有的写操作都会被忽略。

寄存器**4:** 看门狗中断清零（**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-



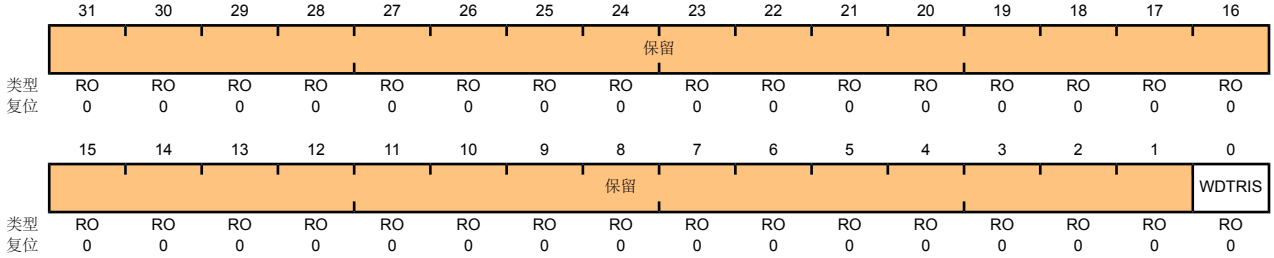
位/域	名称	类型	复位	描述
31:0	WDTIntClr	WO	-	清除看门狗中断。

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



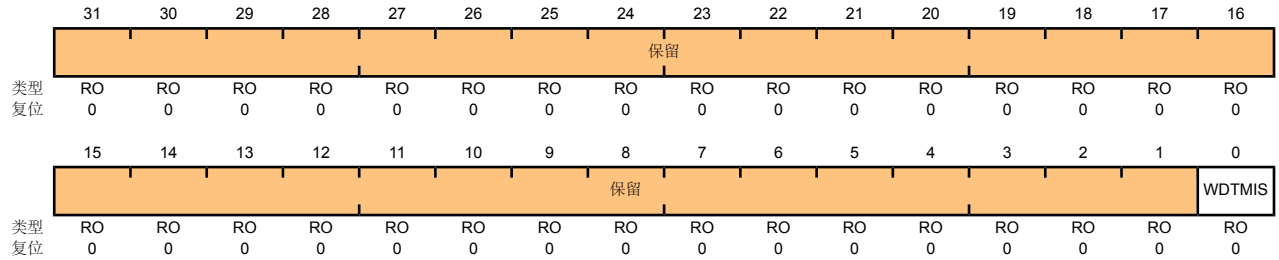
位/域	名称	类型	复位	描述
31:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	WDTRIS	RO	0	看门狗原始中断状态 给出**WDTINTR**的原始中断状态（在屏蔽之前）。.

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



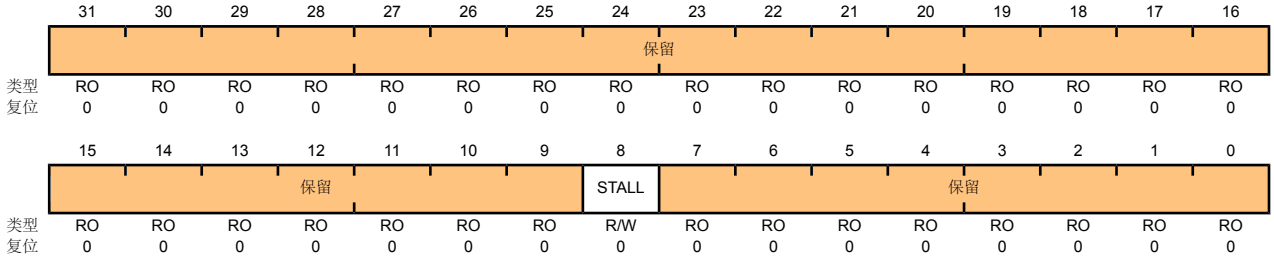
位/域	名称	类型	复位	描述
31:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	WDTMIS	RO	0	看门狗屏蔽后的中断状态 给出**WDTINTR**中断在屏蔽后的中断状态。

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



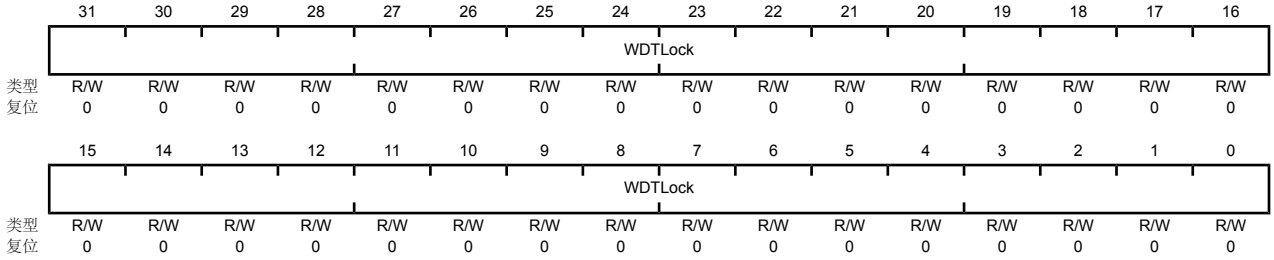
位/域	名称	类型	复位	描述
31:9	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
8	STALL	R/W	0	看门狗停止使能标志 当设为1时，如果调试器使 Stellaris®微控制器停止，则看门狗定时器也会停止计数。而一旦微控制器重新启动，则看门狗定时器也会恢复计数。
7:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器**8:** 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



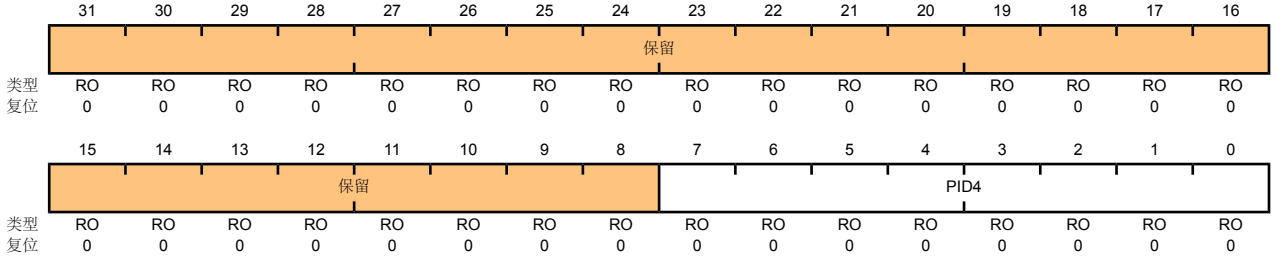
位/域	名称	类型	复位	描述
31:0	WDTLock	R/W	0x0000	看门狗锁定 写入值0x1ACC.E551 允许看门狗寄存器执行写操作。写入其它值则重新进行锁定，防止对任意寄存器进行更新。 读取该寄存器时返回以下值： 锁定: 0x0000.0001 未锁定: 0x0000.0000

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



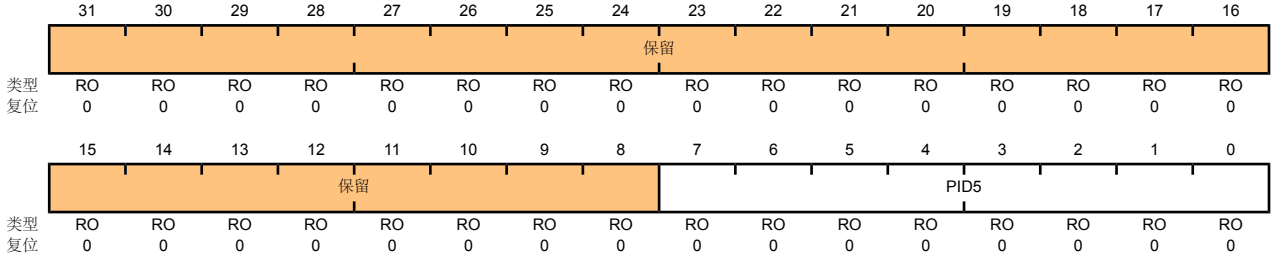
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	WDT 外设ID寄存器[7:0]

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



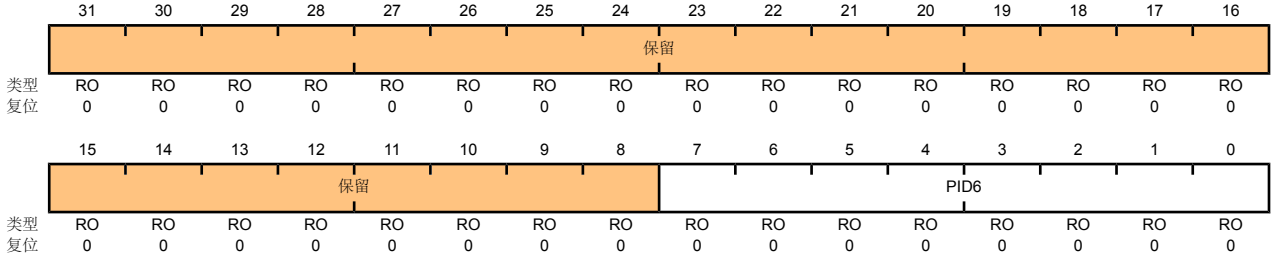
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	WDT外设ID寄存器[15:8]

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



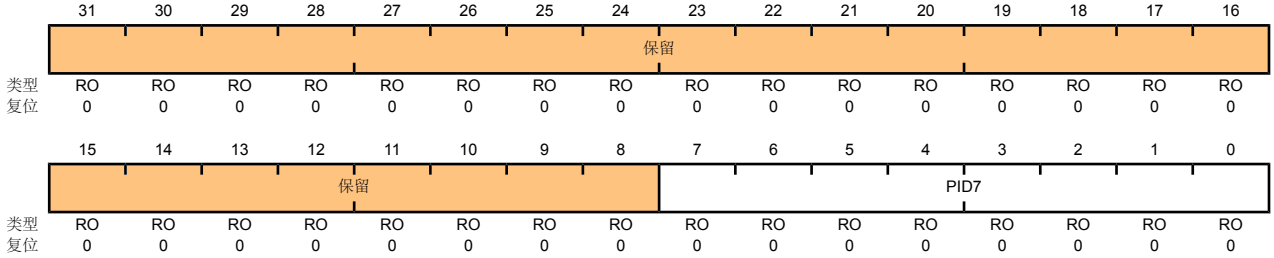
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	WDT外设ID寄存器[23:16]

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



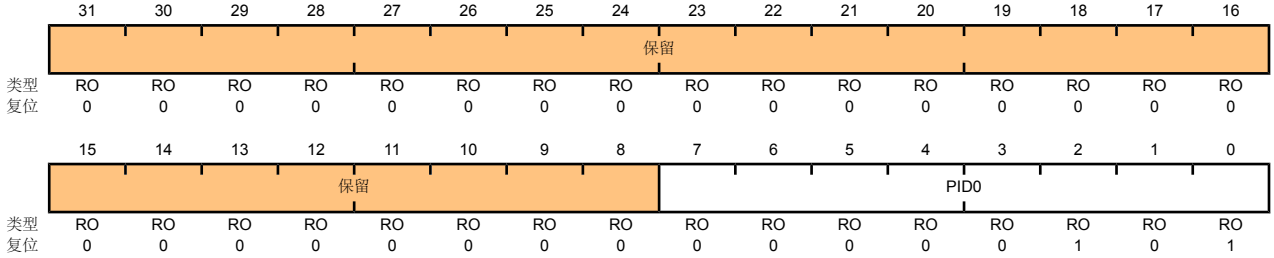
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	WDT外设ID寄存器[31:24]

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



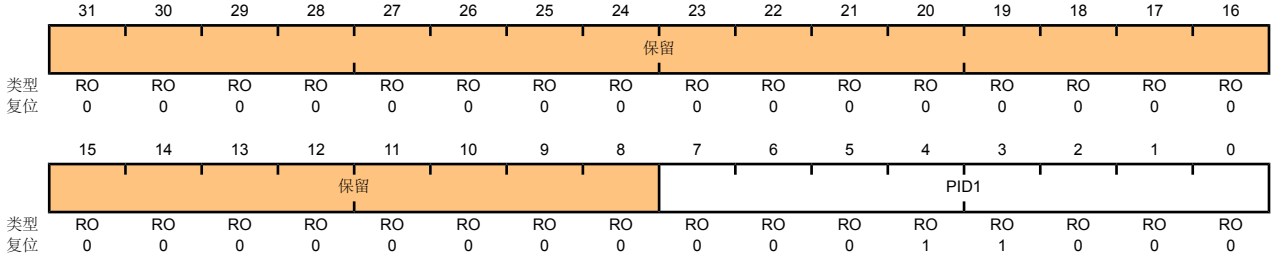
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x05	看门狗标识ID寄存器[7:0]

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



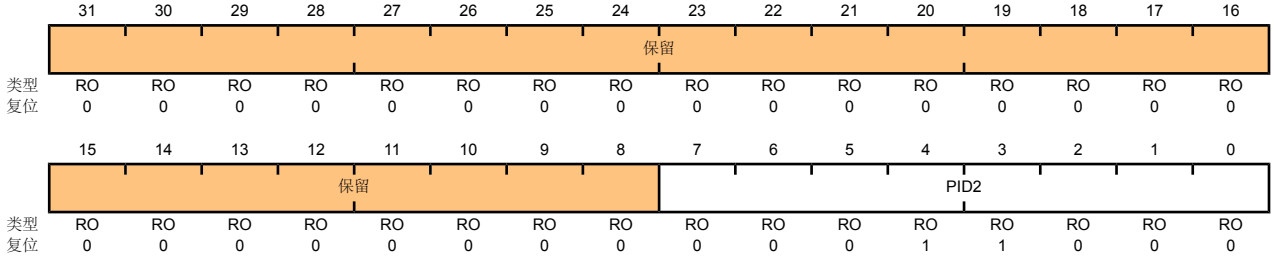
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x18	看门狗外设ID寄存器[15:8]

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



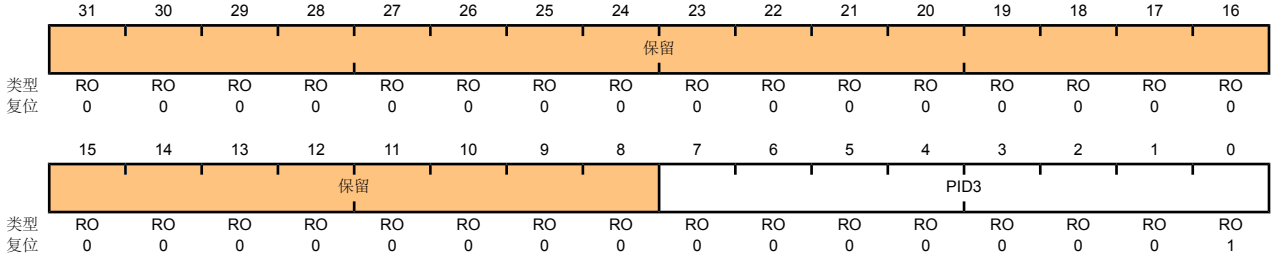
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	看门狗外设ID寄存器[23:16]

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



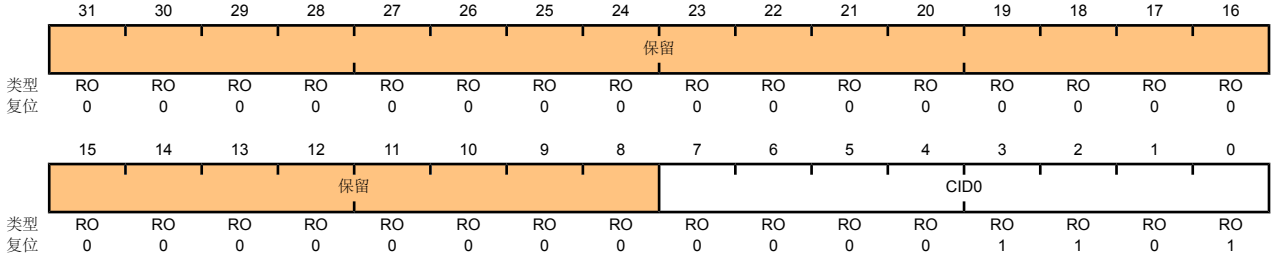
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	看门狗外设ID寄存器[31:24]

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



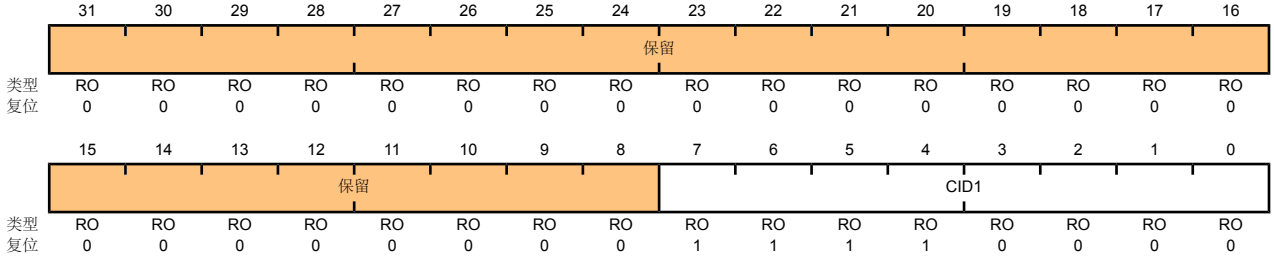
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	看门狗PrimeCell ID寄存器[7:0]

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



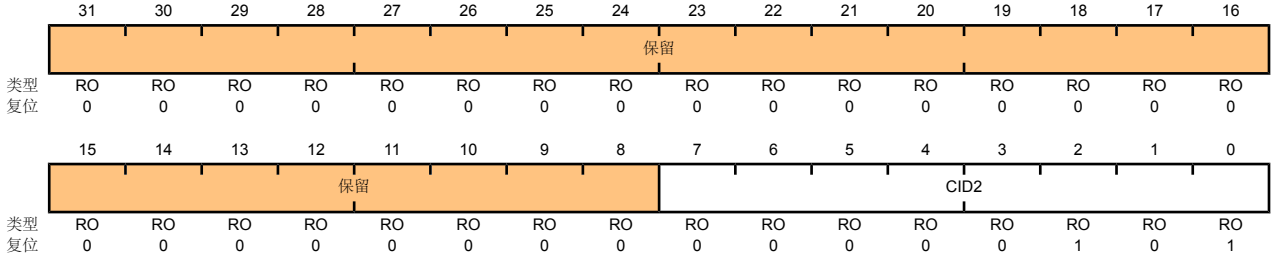
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	看门狗PrimeCell ID寄存器[15:8]

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



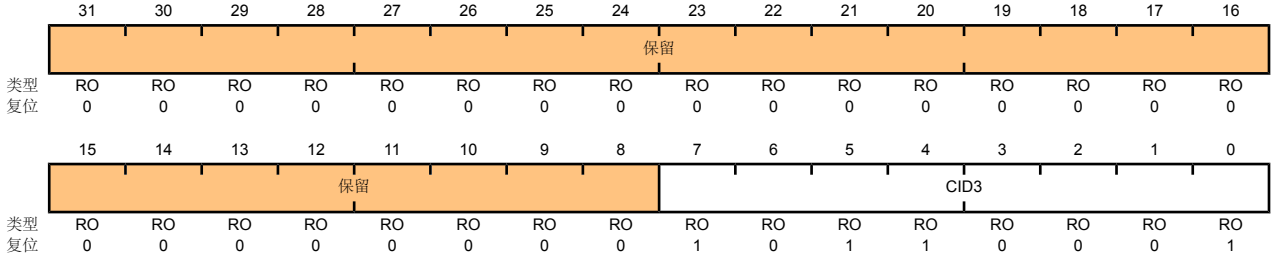
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	看门狗PrimeCell ID寄存器[23:16]

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	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	CID3	RO	0xB1	看门狗PrimeCell ID寄存器[31:24]

11 通用异步收发器 (UART)

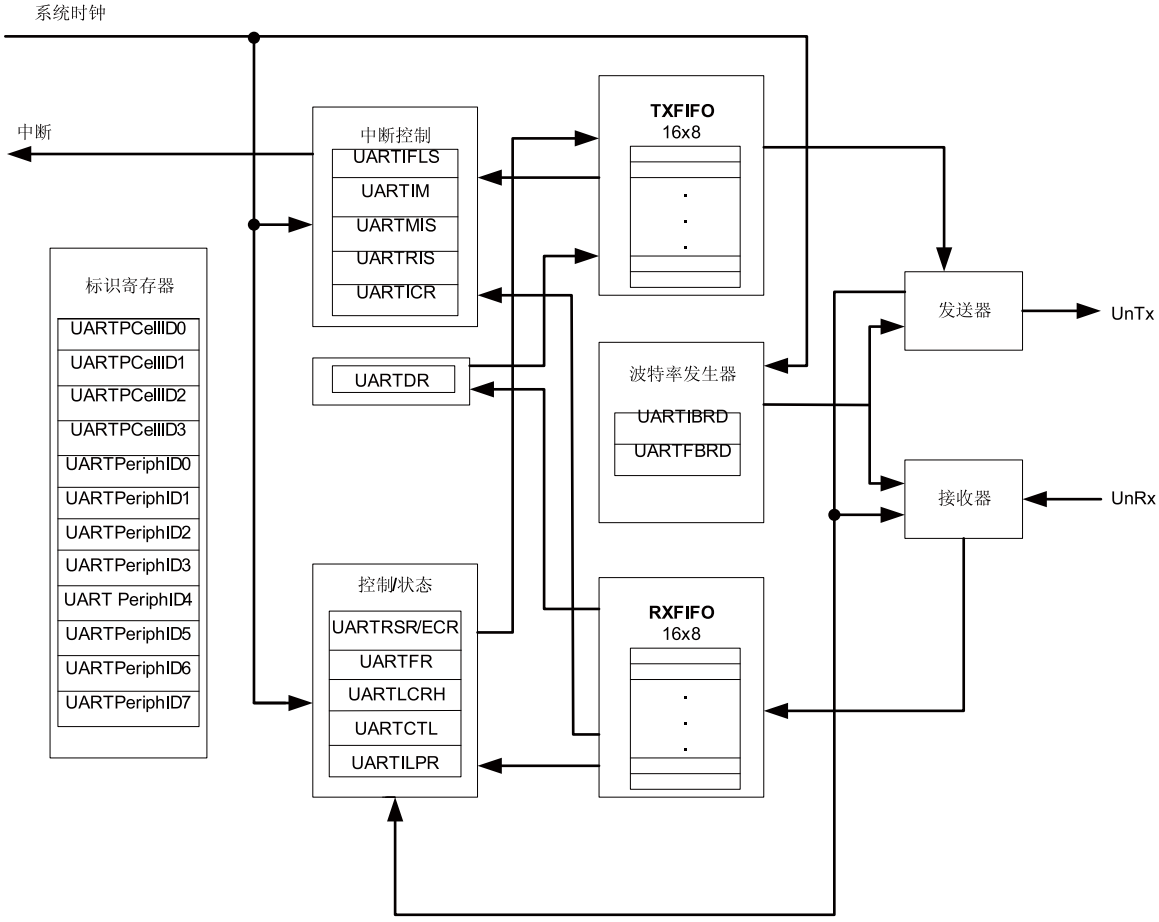
Stellaris® 通用异步收发器 (UART) 具有完全可编程、16C550型串行接口的特性。该LM3S6100 控制器含有 1个 UART模块。

该 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460.8Kbps
- 标准的异步通信位：起始位、停止位和奇偶校验位（parity）
- 检测错误的起始位
- 线中止（Line-break）的产生和检测
- 完全可编程的串行接口特性：
  - 5、 6、 7或 8 个数据位
  - 偶校验、奇校验、粘着或无奇偶校验位的产生/检测
  - 产生1或2个停止位
- 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。

11.1 结构图

图 **11-1. UART**模块的结构图



11.2 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 225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。

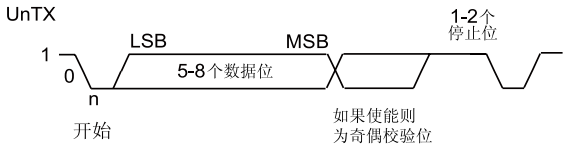
UART外设还包含一个串行红外（SIR）编码器/解码器模块，这个模块可以通过与一个红外收发器连接来实现IrDA SIR物理层规范。通过UARTCTL寄存器对SIR功能进行编程。

11.2.1 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 11-2 在 211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 **11-2. UART**字符帧



11.2.2 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 221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 222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 223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**

11.2.3 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 219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 210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 217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。

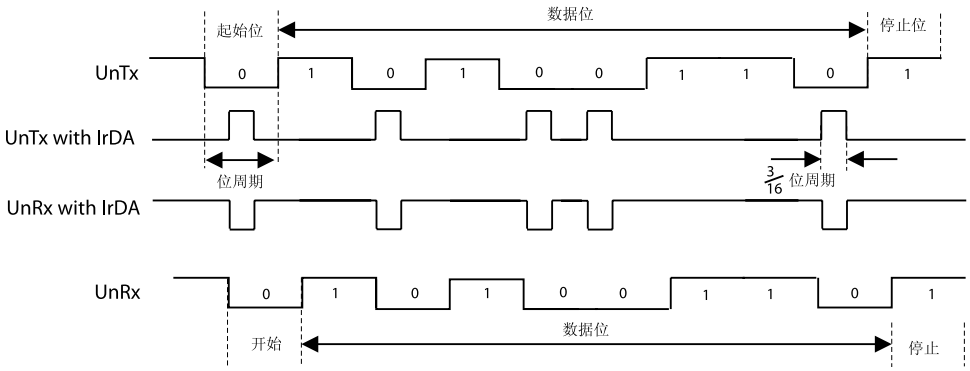
11.2.4 串行红外 **(SIR)**协议

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 11-3 在 212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 **11-3. 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。

11.2.5 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 216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**(223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 219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 227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。

11.2.6 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 231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 228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 230页)便可查询到该状态。

只需把**UART**中断清除**(UARTICR)** 寄存器 (见 232页)中相应的位置位，便可以清除中断（**UARTMIS**和**UARTRIS**寄存器的中断）。

11.2.7 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 225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。

11.2.8 **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。

11.3 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211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 221页) 的DIVINT位域应该设为10。加载到**UARTFBRD** 寄存器(见 222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。

11.4 寄存器映射

表 11-1 在 215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

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 225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 **11-1. UART**寄存器映射

偏移量	名称	类型	复位	描述	见页面
0x000	UARTDR	RO	0x0000.0000	UART数据	216
0x004	UARTRSR/ UARTECR	R/W	0x0000.0000	UART接收状态/错误清除	217
0x018	UARTFR	RO	0x0000.0090	UART标志	219
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA 低功耗寄存器	220
0x024	UARTIBRD	R/W	0x0000.0000	UART整数波特率除数	221
0x028	UARTFBRD	R/W	0x0000.0000	UART小数波特率除数	222
0x02C	UARTLCRH	R/W	0x0000.0000	UART线控	223
0x030	UARTCTL	R/W	0x0000.0300	UART控制	225
0x034	UARTIFLS	R/W	0x0000.0012	UART中断的FIFO深度选择	227
0x038	UARTIM	R/W	0x0000.0000	UART中断屏蔽	228
0x03C	UARTRIS	RO	0x0000.000F	UART原始中断状态	230
0x040	UARTMIS	RO	0x0000.0000	UART屏蔽后的中断状态	231
0x044	UARTICR	W1C	0x0000.0000	UART中断清除	232
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART外设标识4	234
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART外设标识5	235
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART外设标识6	236
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART外设标识7	237
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART外设标识0	238
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART外设标识1	239
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART外设标识2	240
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART外设标识3	241
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell 标识0	242
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell标识1	243
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell标识2	244
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell标识3	245

11.5 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**

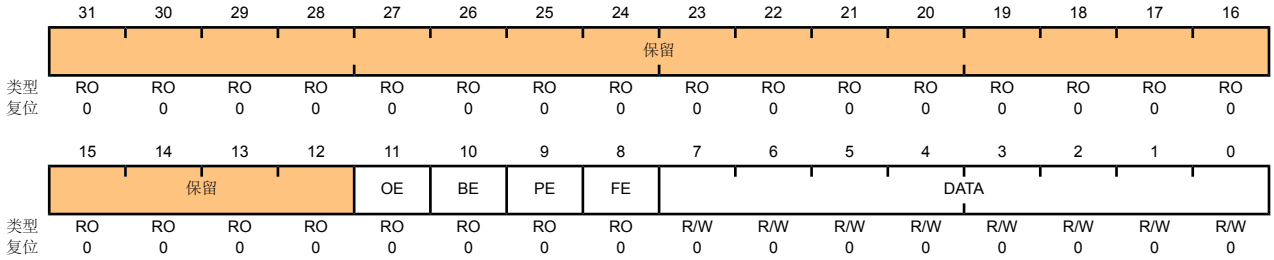
该寄存器是数据寄存器（FIFO的接口）。

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

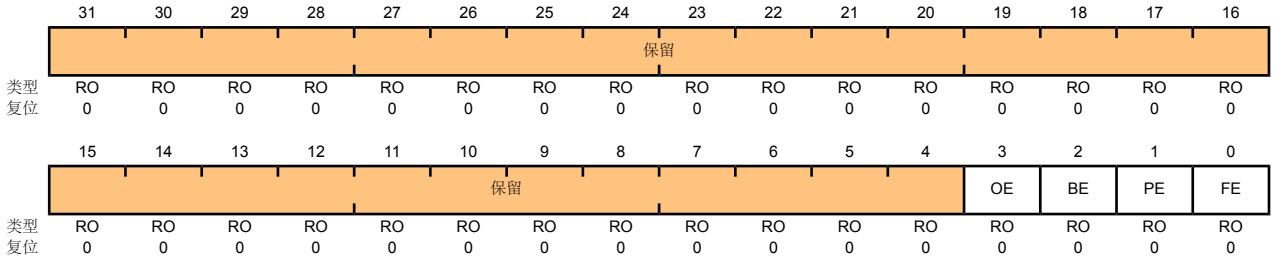
接收状态除了可以从**UARTDR**寄存器中读取之外，还可以从**UARTRSR**寄存器中读取。如果从该寄存器读取状态，与入口相对应的状态信息将在读取**UARTRSR**前先从**UARTDR**读取。当溢出的情况发生时，溢出的状态信息将被立即置位。

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



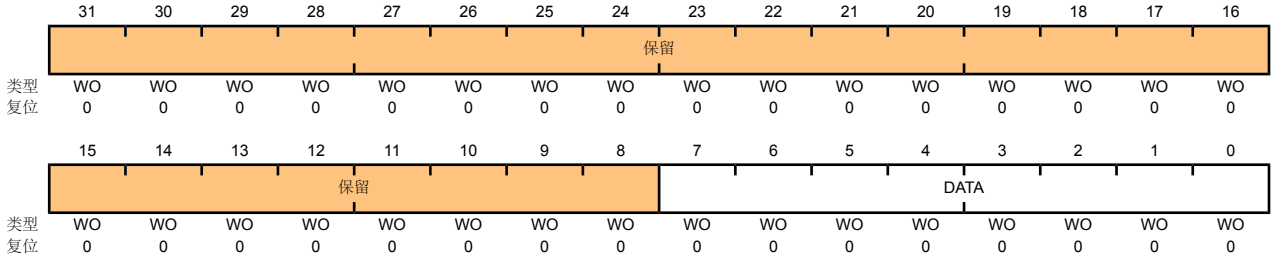
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读–修改–写操作过程中应当保持不变。 **UARTRSR**寄存器不能被写。
3	OE	RO	0	UART溢出错误标志 当FIFO满且接收到新的数据时该位被设为1。对**UARTECR**进行写操作会将该位清零。 由于在FIFO满时不再有数据写入，所以FIFO内容将保持有效，只有移位寄存器的内容会被覆盖。此时，CPU必须读取数据以便将FIFO清空。
2	BE	RO	0	UART中止错误（Break Error） 在检测到中止的情况时该位会被设为1，表示接收数据输入在长于一个完整字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



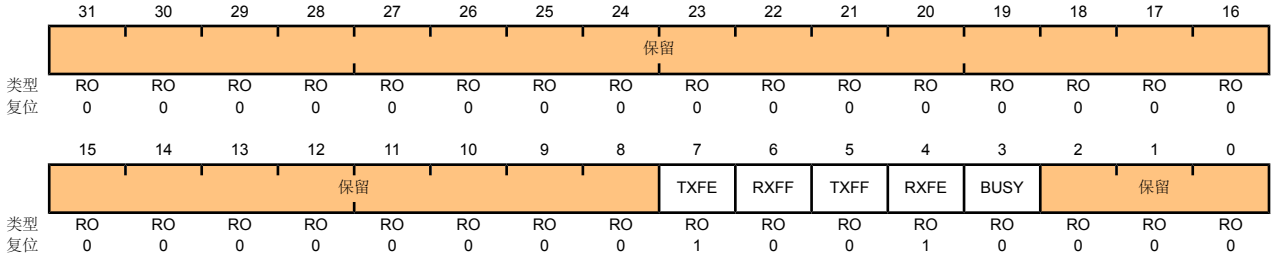
位/域	名称	类型	复位	描述
31:8	保留	WO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DATA	WO	0	错误清除标志 将任意数据写入该寄存器会将帧、奇偶校验、中止和溢出标志清零。

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：

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

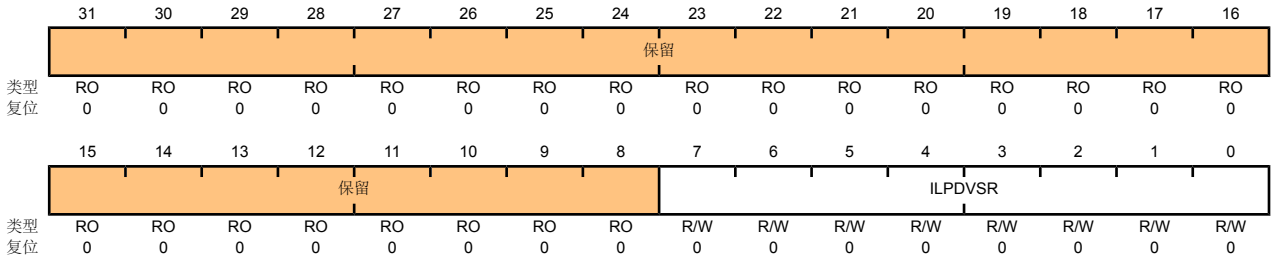
此处， $F_{IrLPBaud16}$ 额定值为 1.8432 MHz。

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



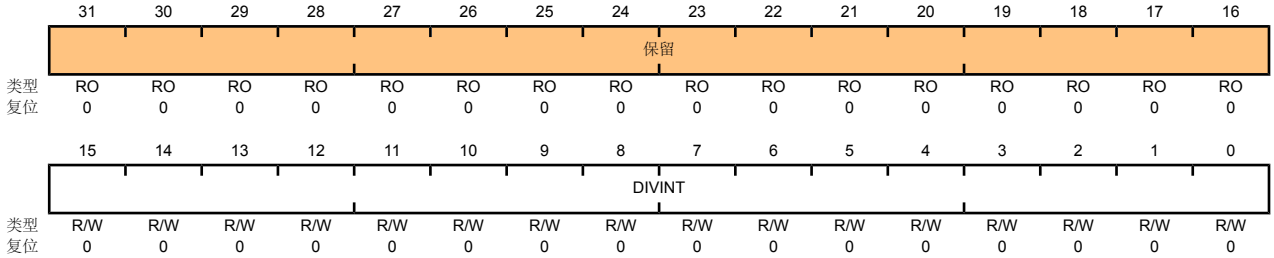
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读–修改–写操作过程中应当保持不变。
7:0	ILPDVSR	R/W	0x0000	IrDA低功耗除数 这是一个8-位低功耗除数。

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 211页。

UART整数波特率除数 (UARTIBRD)

偏移量0x024
类型R/W, 复位0x0000.0000



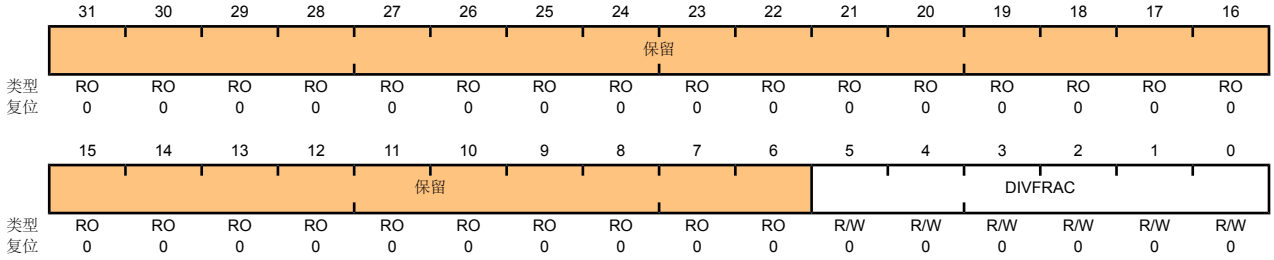
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	DIVINT	R/W	0x0000	整数波特率除数

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 211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	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	DIVFRAC	R/W	0x00	小数波特率除数

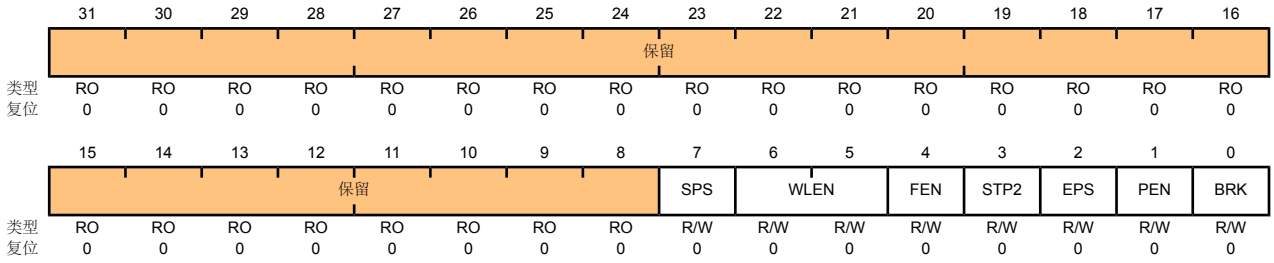
寄存器7: UART线控（UARTLCRH），偏移量 0x02C

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： 0x3: 8位 0x2: 7位 0x1: 6位 0x0: 5 位 (缺省值)
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。

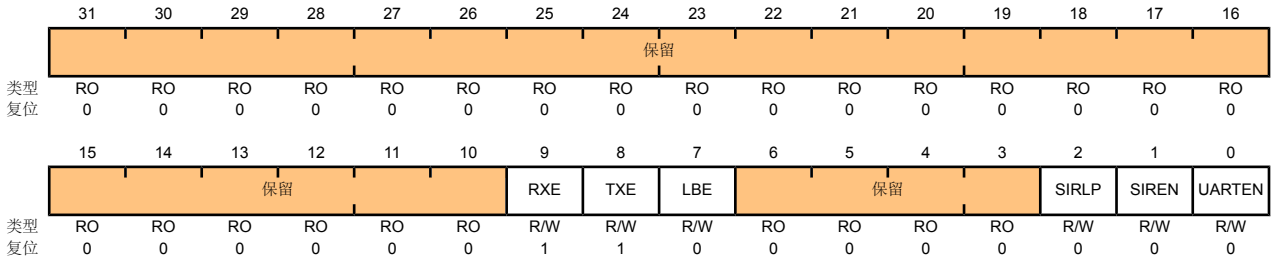
寄存器8: UART控制（UARTCTL），偏移量 0x030

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 220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

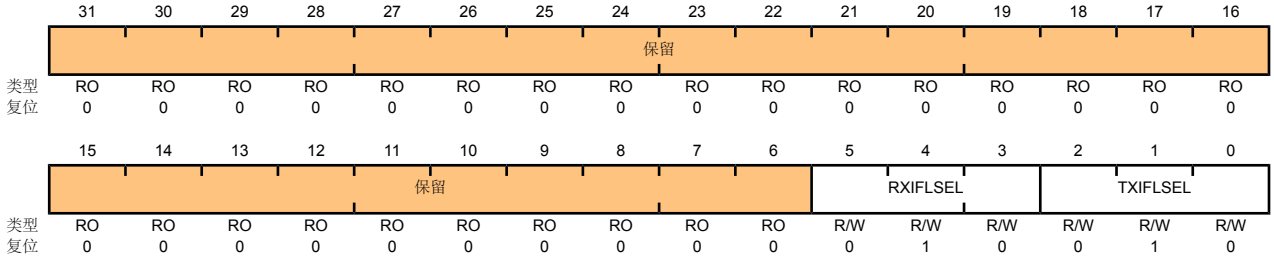
UARTIFLS寄存器是中断的FIFO深度（level）选择寄存器。你可以使用该寄存器来定义**UARTRIS**寄存器中TXRIS和RXRIS位触发（中断）时的FIFO深度。

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	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。
5:3	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: 000: RX FIFO ≥ 全满时的1/8 001: RX FIFO ≥ 全满时的¼ 010: RX FIFO ≥ 全满时的½ (缺省值) 011: RX FIFO ≥ 全满时的¾ 100: RX FIFO ≥ 全满时的7/8 101-111: 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: 000: TX FIFO ≤ 全满时的1/8 001: TX FIFO ≤ 全满时的¼ 010: TX FIFO ≤ 全满时的½ (缺省值) 011: TX FIFO ≤ 全满时的¾ 100: TX FIFO ≤ 全满时的7/8 101-111: 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	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					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	保留			
类型	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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
31:11	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。
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。

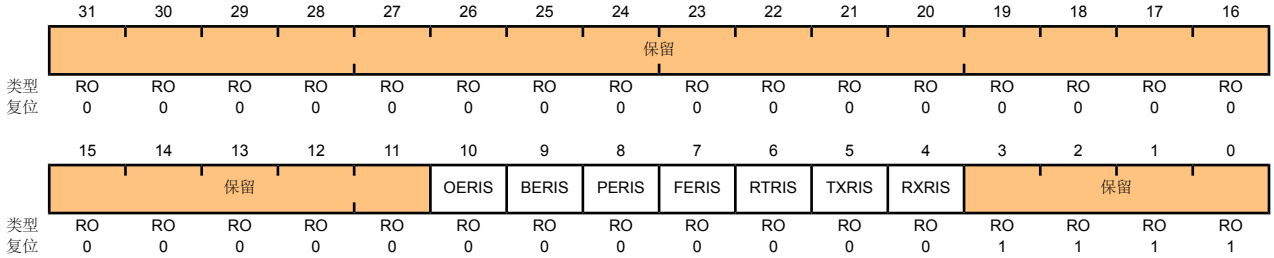
位/域	名称	类型	复位	描述
4	RXIM	R/W	0	UART接收中断屏蔽位 读取时，返回RXIM中断屏蔽位的当前值。 该位置位时，可以将RXIM中断传递到中断控制器。
3:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器**11: 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



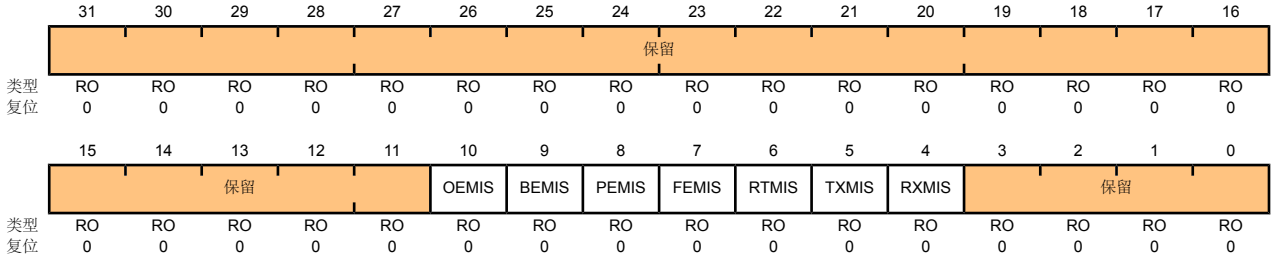
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OERIS	RO	0	UART溢出错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
9	BERIS	RO	0	UART中止（break）错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
8	PERIS	RO	0	UART奇偶校验错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
7	FERIS	RO	0	UART帧错误的原始中断状态 显示接收中断的原始（屏蔽前）中断状态。
6	RTRIS	RO	0	UART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



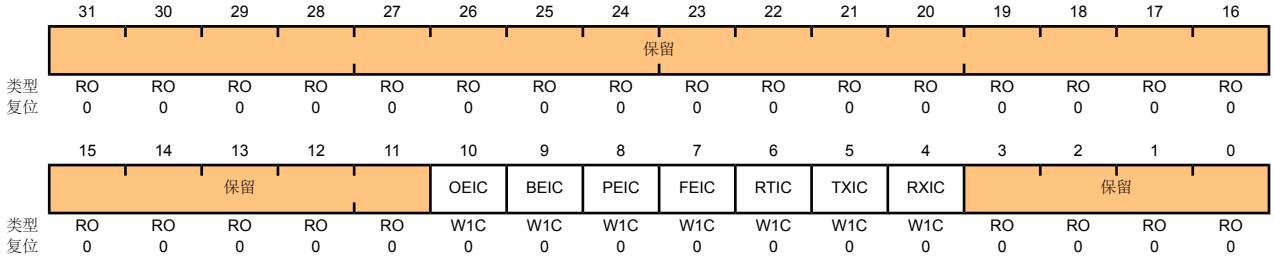
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10	OEMIS	RO	0	UART溢出错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
9	BEMIS	RO	0	UART 中止（break）错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
8	PEMIS	RO	0	UART奇偶校验错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
7	FEMIS	RO	0	UART帧错误屏蔽后的中断状态 显示接收屏蔽后的中断状态。
6	RTMIS	RO	0	UART接收超时屏蔽后的中断状态 显示接收屏蔽后的中断状态。
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	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。
10	OEIC	W1C	0	溢出错误中断清除 0: 对中断不起作用。 1: 清除中断。
9	BEIC	W1C	0	中止（break）错误中断清除 0: 对中断不起作用。 1: 清除中断。
8	PEIC	W1C	0	奇偶校验错误中断清除 0: 对中断不起作用。 1: 清除中断。
7	FEIC	W1C	0	帧错误中断清除 0: 对中断不起作用。 1: 清除中断。
6	RTIC	W1C	0	接收超时中断清除 0: 对中断不起作用。 1: 清除中断。
5	TXIC	W1C	0	发送中断清除 0: 对中断不起作用。 1: 清除中断。
4	RXIC	W1C	0	接收中断清除 0: 对中断不起作用。 1: 清除中断。

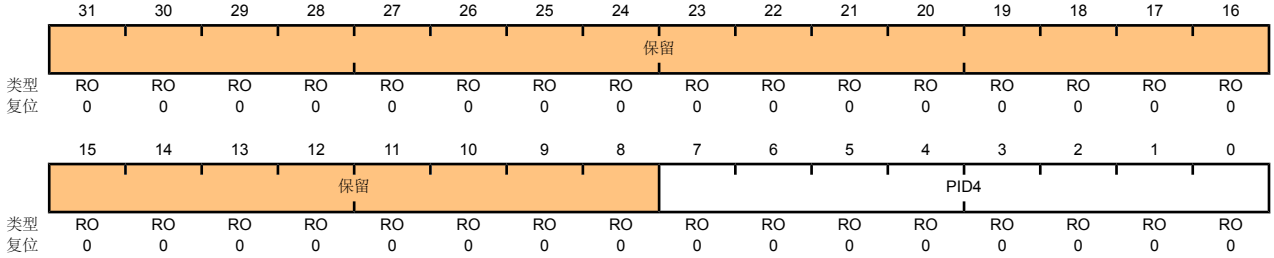
位/域	名称	类型	复位	描述
3:0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器**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



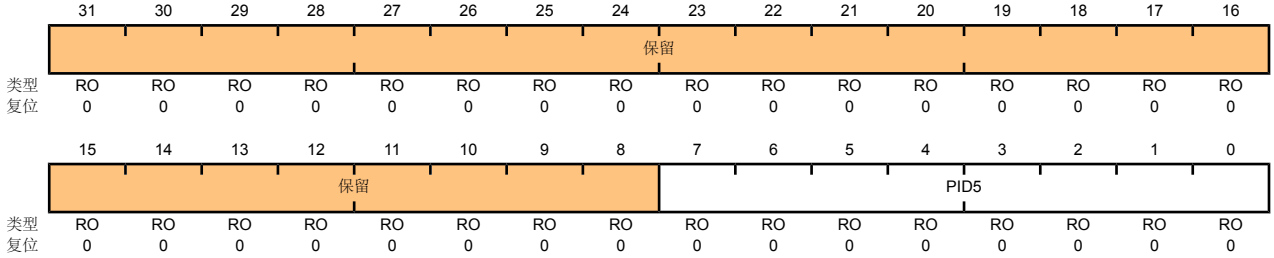
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	UART外设ID寄存器[7:0] 可被软件用来标识该外设的存在与否。

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



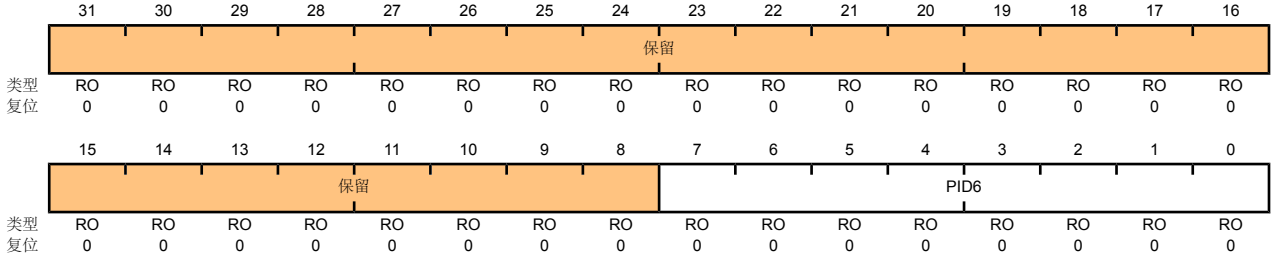
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	UART外设ID寄存器[15:8] 可被软件用来标识该外设的存在与否。

寄存器**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



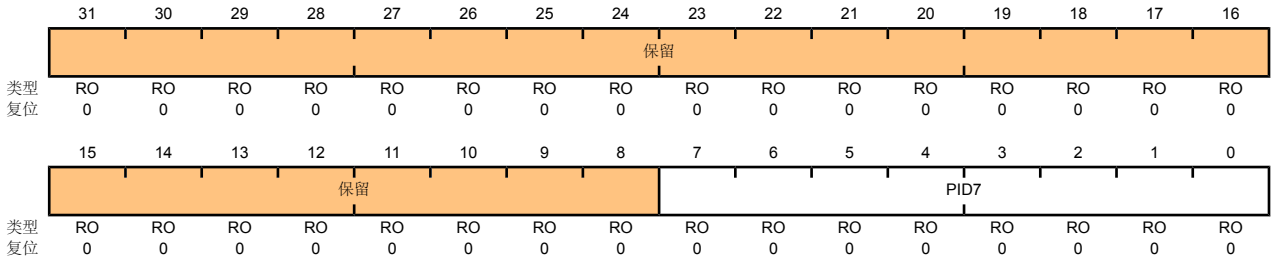
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	UART外设ID寄存器[23:16] 可被软件用来标识该外设的存在与否。

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



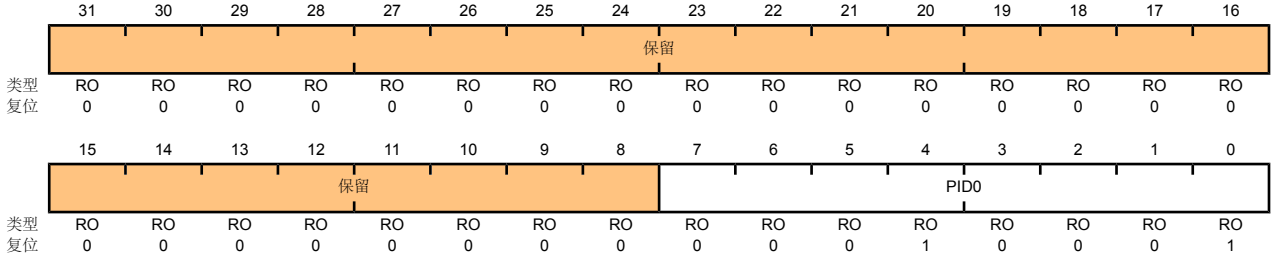
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	UART外设ID寄存器[31:24] 可被软件用来标识该外设的存在与否。

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



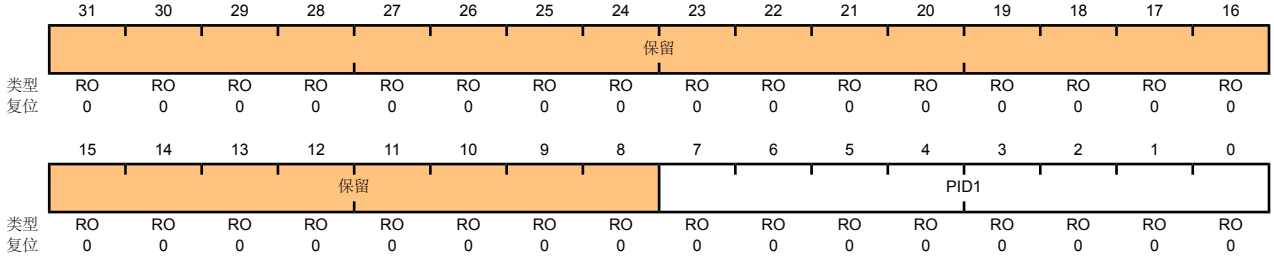
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x11	UART外设ID寄存器[7:0] 可被软件用来标识该外设的存在与否。

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



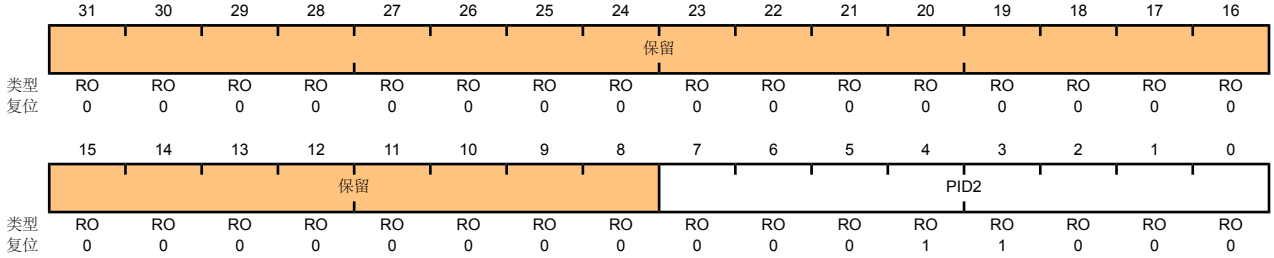
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	UART外设ID寄存器[15:8] 可被软件用来标识该外设的存在与否。

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



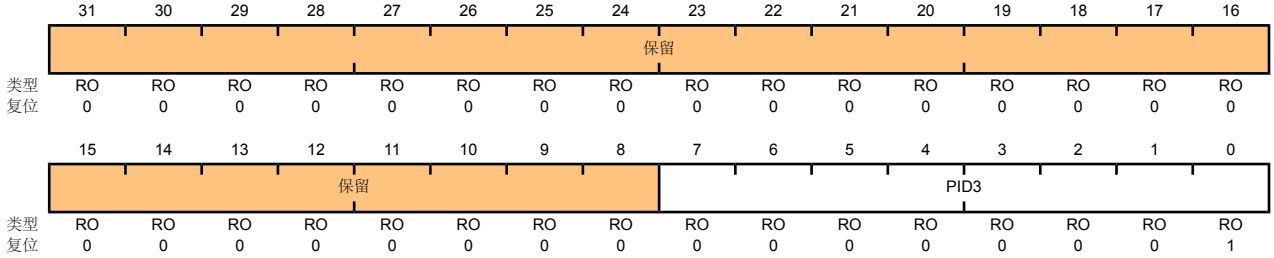
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	UART外设ID寄存器[23:16] 可被软件用来标识该外设的存在与否。

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



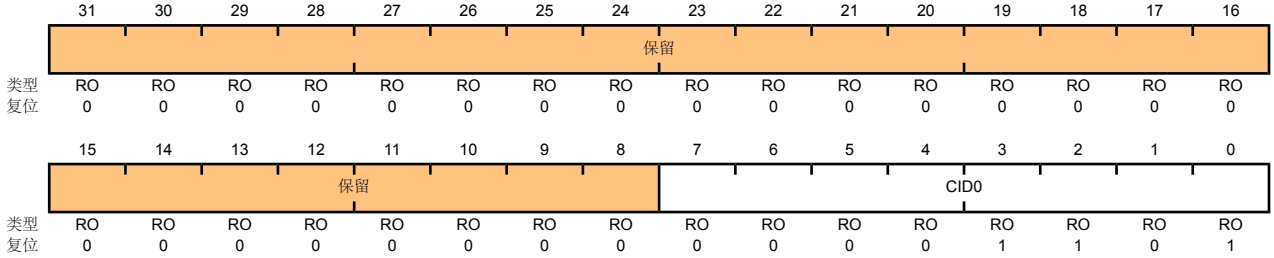
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	UART外设ID寄存器[31:24] 可被软件用来标识该外设的存在与否。

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



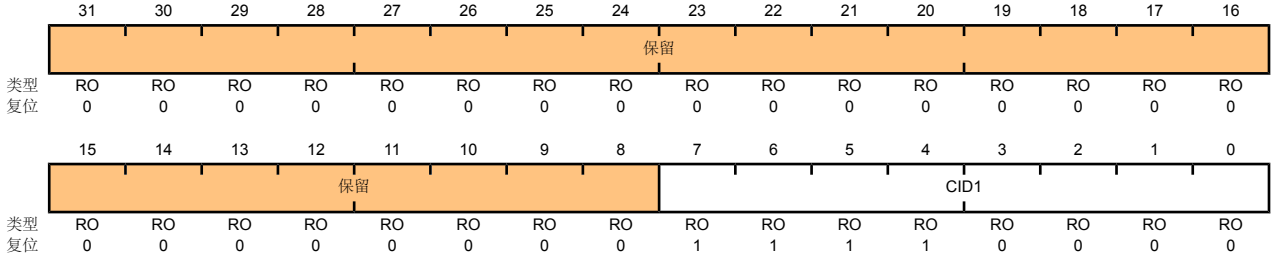
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	UART PrimeCell ID 寄存器[7:0] 提供软件一个标准的跨外设标识系统。

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



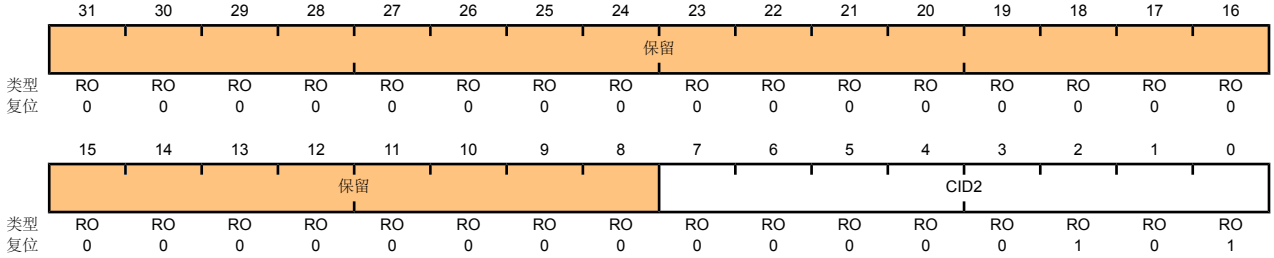
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	UART PrimeCell ID寄存器[15:8] 提供软件一个标准的跨外设标识系统。

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



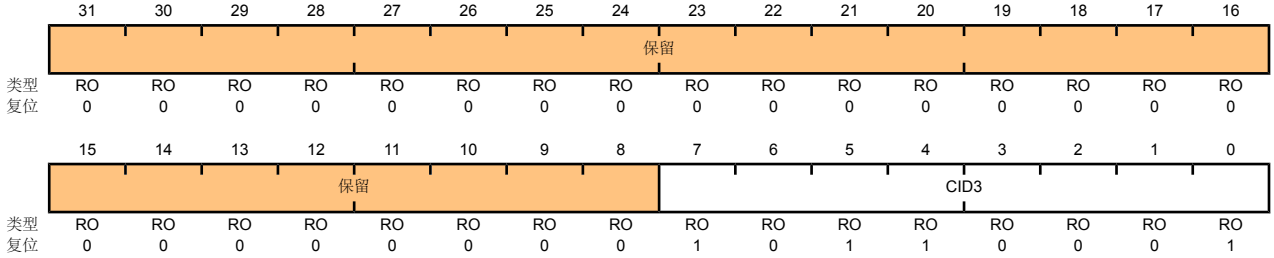
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	UART PrimeCell ID寄存器[23:16] 提供软件一个标准的跨外设标识系统。

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	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	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。

12 同步串行接口 (SSI)

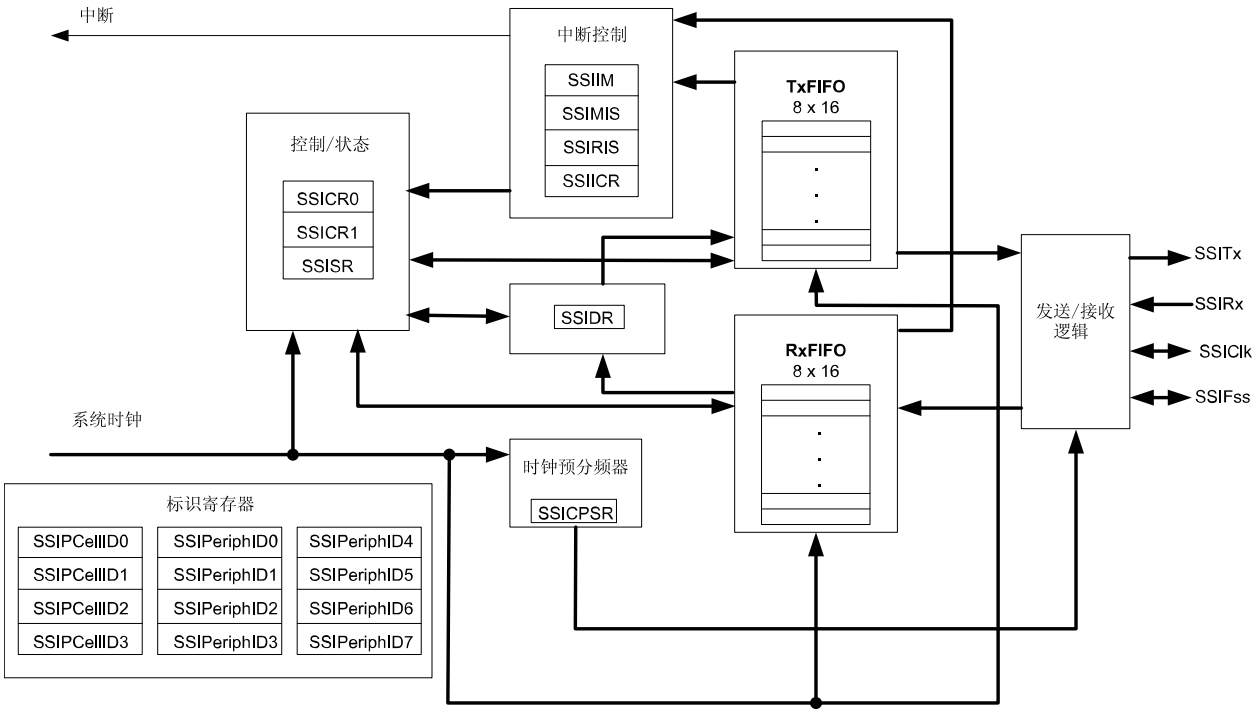
Stellaris® 同步串行接口 (SSI) 是与具有Freescale SPI、MICROWIRE或德州仪器同步串行接口（SSI）的外设器件进行同步串行通信的主机或从机接口。

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

12.1 结构图

图 **12-1. SSI**模块的结构图



12.2 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。

12.2.1 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25-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 262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 257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12.5MHz，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 353页。

12.2.2 FIFO操作

12.2.2.1 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 260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。

12.2.2.2 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）。

12.2.3 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 263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 264页 和 265页）中读取。

12.2.4 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）。

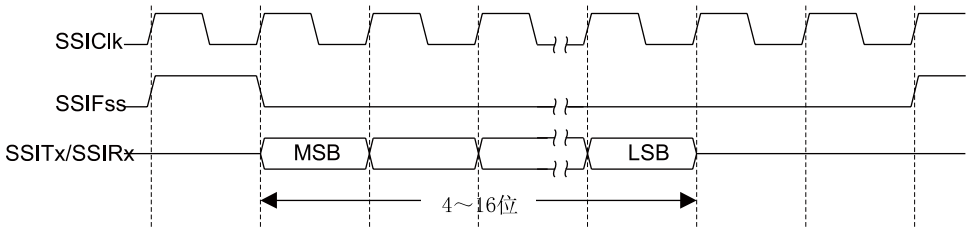
而对于德州仪器同步串行帧格式，在发送每个帧之前，SSIFss管脚会发出一个以上升沿开始并持续一个时钟周期的脉冲。在这种帧格式中，SSI和片外从器件在SSIClk的上升沿驱动各自的输出数据，并在下降沿锁存另一个器件的数据。

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。

12.2.4.1 德州仪器同步串行的帧格式

图 12-2 在 248页显示了一次传输的德州仪器同步串行的帧格式。

图 **12-2. TI**同步串行的帧格式（单次传输）

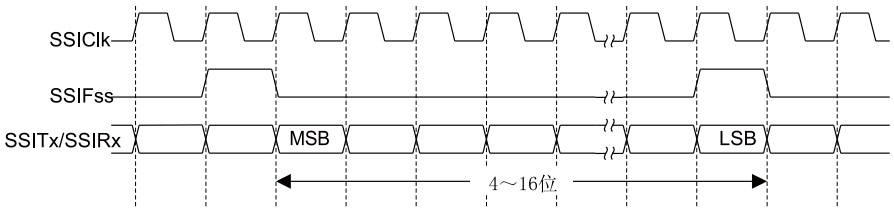


在该模式中，任何时候当SSI空闲时，SSIClk和SSIFss被强制为低电平，发送数据线SSITx为三态。一旦发送FIFO的底部入口包含数据，SSIFss就会变为高电平并持续一个SSIClk周期。 要发送的值也从发送FIFO传输到发送逻辑的串行移位寄存器中。在SSIClk的下一个上升沿，4～16位数据帧的MSB从SSITx管脚移出。同样，接收到的数据的MSB也通过片外串行从器件移到SSIRx管脚上。

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 12-3 在 249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 12-3. TI 同步串行的帧格式（连续传输）



12.2.4.2 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。

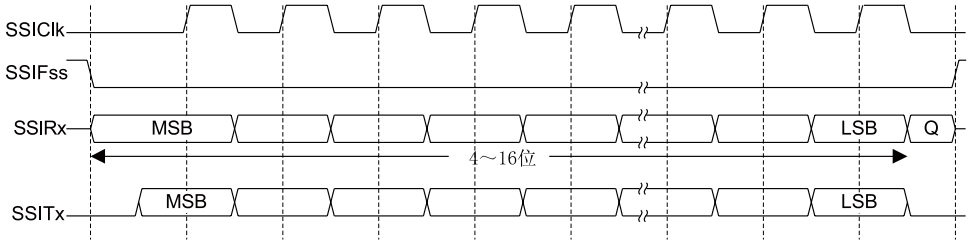
***SPH**相位控制位*

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。

12.2.4.3 SPO=0和SPH=0时，Freescale SPI的帧格式

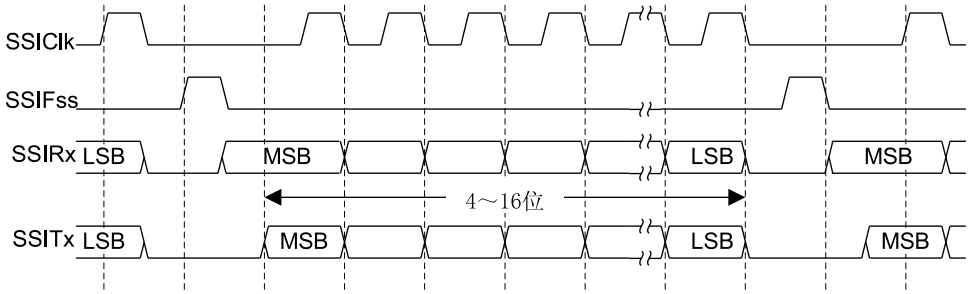
SPO=0和SPH=0时，Freescale SPI帧格式的单次和连续传输信号序列如图 12-4 在 249页 和 图 12-5 在 250页所示。

图 12-4. SPO=0和SPH=0时Freescale SPI 的帧格式 (单次传输)



注意: Q未定义。

图 12-5. 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。

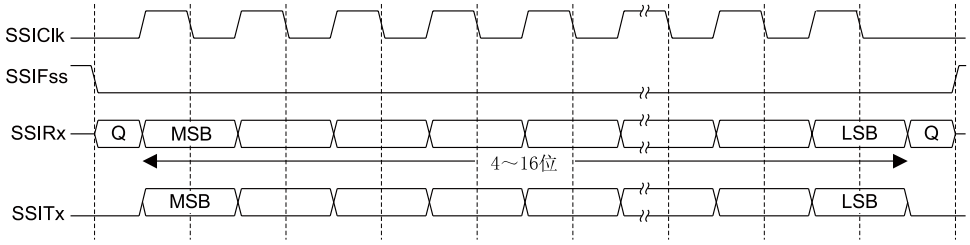
如果传输一个字，则在数据字的所有位都已传输完之后，SSIFss线在捕获到最后一个位之后的一个SSIClk周期返回到其空闲的高电平状态。

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。

12.2.4.4 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 12-6 在 250页所示， 该图涵盖了单次和连续传输这两种情况。

图 12-6. 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。

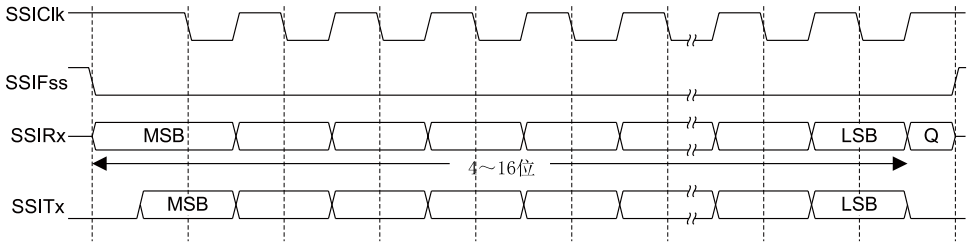
如果传输一个字，则在所有位传输完之后，SSIFss线在捕获到最后一个位之后的一个SSIClk周期返回到其空闲的高电平状态。

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。

12.2.4.5 SPO=1和SPH=0时Freescale SPI的帧格式

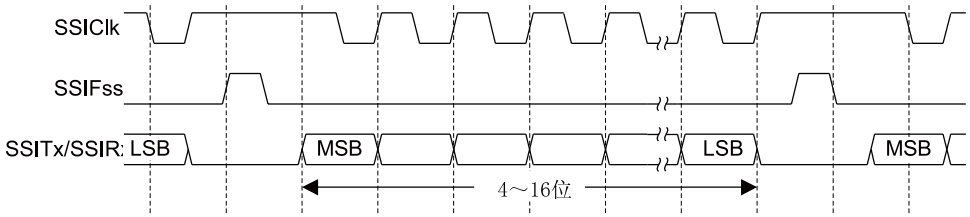
SPO=1和SPH=0时，Freescale SPI帧格式的单次和连续传输信号序列如图 12-7 在 251页 和 图 12-8 在 251页所示。

图 **12-7. SPO=1和SPH=0时Freescale SPI的帧格式（单次传输）**



注意： Q未定义。

图 **12-8. 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。

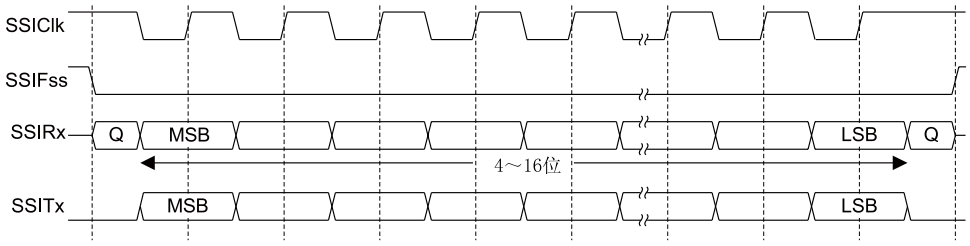
如果传输一个字，则在数据字的所有位传输完之后，SSIFss线在最后一个位传输完之后的一个SSIClk周期返回到其空闲的高电平状态。

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。

12.2.4.6 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 12-9 在 252页所示，该图涵盖了单次和连续传输两种情况。

图 **12-9. 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。

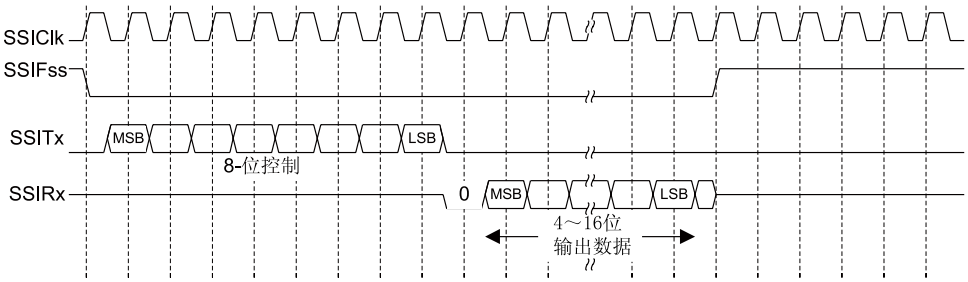
而对于连续的背对背(back-to-back)传输，SSIFss管脚保持其有效的低电平状态，直至最后一个字的最后一位捕获完成，再返回其上述的空闲状态。

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。

12.2.4.7 MICROWIRE的帧格式

图 12-10 在 253页显示了单次传输的MICROWIRE帧格式。图 12-11 在 254页 显示了该格式的背对背（back-to-back）传输情况。

图 **12-10. 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

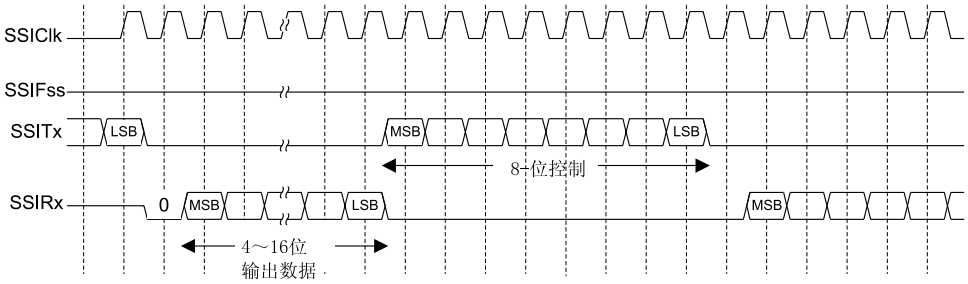
通过向发送FIFO写入一个控制字节可以触发一次传输。在SSIFss的下降沿，发送FIFO底部入口包含的值被传输到发送逻辑的串行移位寄存器中，而8位控制帧的MSB被移出到SSITx管脚上。在该控制帧的传输期间SSIFss保持低电平。SSIRx管脚保持三态。

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。

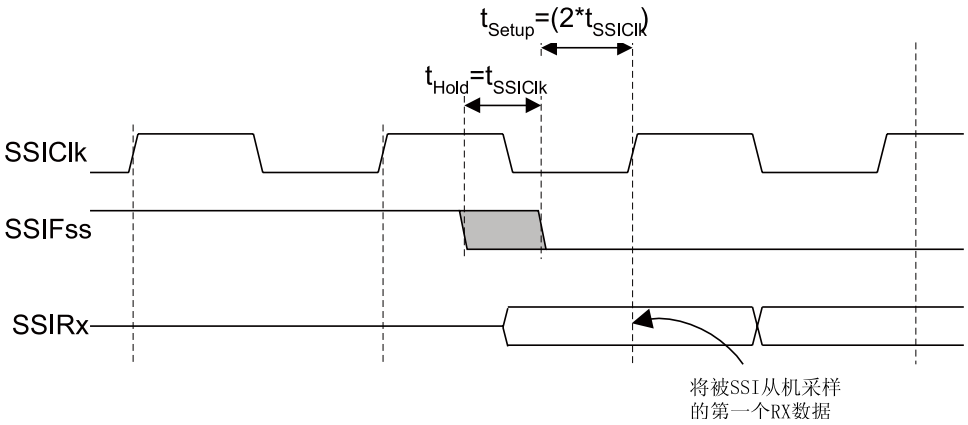
图 12-11. MICROWIRE的帧格式 (连续传输)



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 12-12 在 254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 12-12. MICROWIRE的帧格式，输入建立和保持时间要求



12.3 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。

12.4 寄存器映射

表 12-1 在 255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

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 **12-1. 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	257
0x004	SSICR1	R/W	0x0000.0000	SSI控制1	259
0x008	SSIDR	R/W	0x0000.0000	SSI数据	260
0x00C	SSISR	RO	0x0000.0003	SSI状态	261

偏移量	名称	类型	复位	描述	见页面
0x010	SSICPSR	R/W	0x0000.0000	SSI时钟预分频	262
0x014	SSIIM	R/W	0x0000.0000	SSI中断屏蔽	263
0x018	SSIRIS	RO	0x0000.0008	SSI原始中断状态	264
0x01C	SSIMIS	RO	0x0000.0000	SSI屏蔽后的中断状态	265
0x020	SSIICR	W1C	0x0000.0000	SSI中断清零	266
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI外设标识4	267
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI外设标识 5	268
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI外设标识6	269
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI外设标识7	270
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI外设标识0	271
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI外设标识 1	272
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI外设标识 2	273
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI外设标识 3	274
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell标识0	275
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell标识1	276
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell标识2	277
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell标识3	278

12.5 寄存器描述

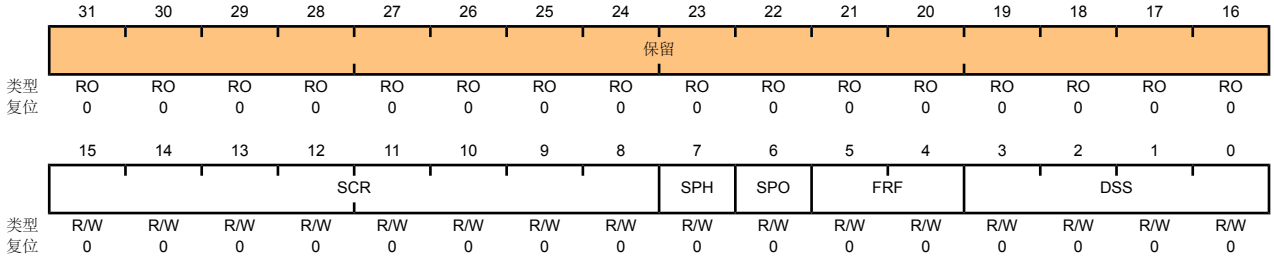
下文将按地址偏移量的数字顺序列举并描述SSI寄存器。

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	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。
15:8	SCR	R/W	0	SSI串行时钟速率 SCR的值用来产生SSI的发送和接收位速率。该位速率为: BR=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	0	SSI帧格式选择 FRF 的值定义如下: FRF值 帧格式 00 Freescale SPI的帧格式 01 德州仪器同步串行的帧格式 10 MICROWIRE的帧格式 11 保留

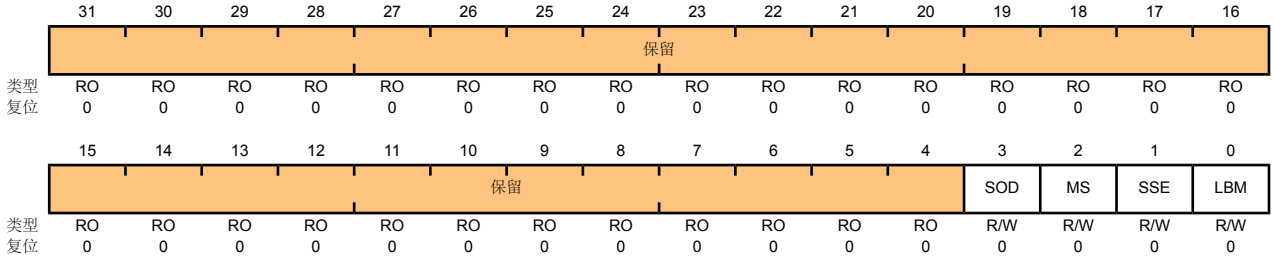
位/域	名称	类型	复位	描述
3:0	DSS	R/W	0	SSI数据长度选择 DSS的值定义如下: DSS值 数据长度 0000-0010 保留 0011 4-位数据 0100 5-位数据 0101 6-位数据 0110 7-位数据 0111 8-位数据 1000 9-位数据 1001 10-位数据 1010 11-位数据 1011 12-位数据 1100 13-位数据 1101 14-位数据 1110 15-位数据 1111 16-位数据

寄存器2: SSI控制1（SSICR1），偏移量 0x004

SSICR1 为控制寄存器1，其位域用来控制SSI模块内的各种功能。主机和从机模式功能就由该寄存器控制。

SSI控制1 (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	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	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。 0: SSI SSI能够在从机输出模式中驱动SSITx 输出。 1: SSI不可以在从机模式中驱动SSITx输出。
2	MS	R/W	0	SSI 主/从选择 该位选择主或从模式并且只有当SSI禁止（SSE=0）时才能修改。 0: 器件配置为主机。 1: 器件配置为从机。
1	SSE	R/W	0	SSI同步串行端口使能 将该位置位可使能SSI操作。 0: SSI操作禁止 1: SSI操作使能。 注意: 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。 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）。

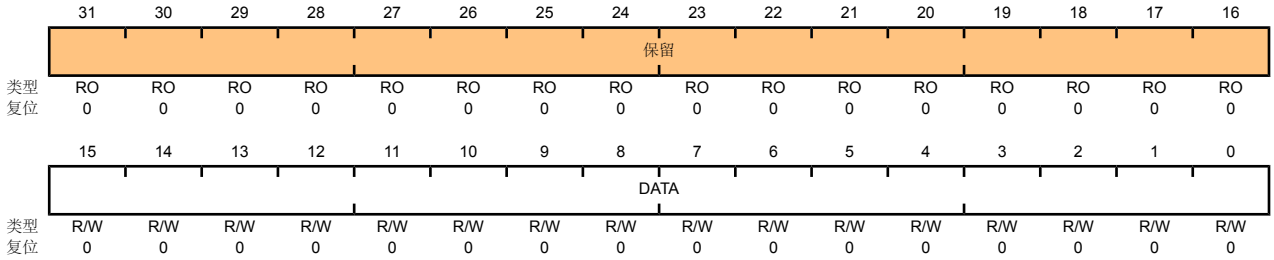
对**SSIDR**进行写操作也就是将数据写入发送FIFO的入口（由写指针来指向）。发送逻辑从发送FIFO中一次移出一个数据值。这个数据值被加载到发送串行移位器中，然后以设置好的位速率串行移出到SSITx管脚。

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

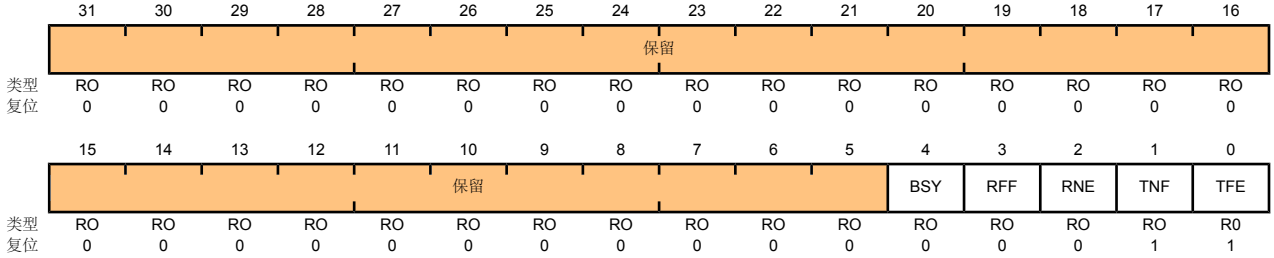


位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	DATA	R/W	0	SSI接收/发送数据 对该位域的读操作即是读接收FIFO。写操作即是写发送FIFO。 当SSI设置为数据大小小于16位时，软件必须正确地调整数据。发送逻辑将忽略顶部未使用的位。接收逻辑自动调整数据。

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	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	BSY	RO	0	SSI忙位 0: SSI空闲。 1: SSI当前正在发送和/或接收帧，或者发送FIFO不为空。
3	RFF	RO	0	SSI接收FIFO满标志 0: 接收FIFO未满 1: 接收FIFO满
2	RNE	RO	0	SSI 接收FIFO不空标志 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 0: 发送FIFO满。 1: 发送FIFO未满。
0	TFE	R0	1	SSI发送FIFO空标志 0: 发送FIFO不为空。 1: 发送FIFO为空。

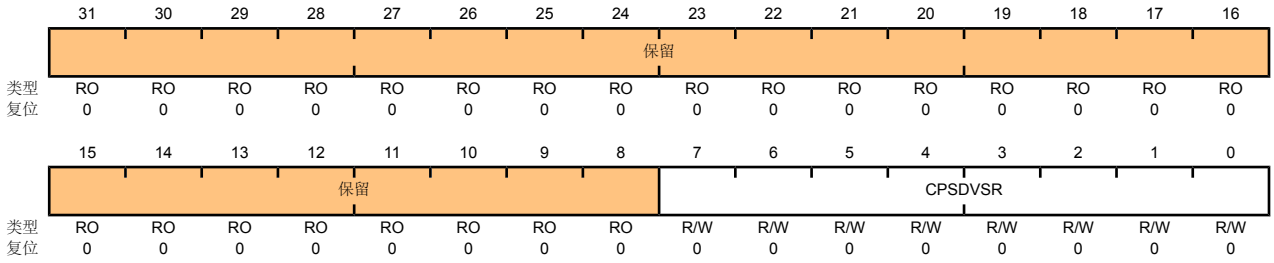
寄存器**5: SSI**时钟预分频（**SSICPSR**），偏移量 **0x010**

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	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	CPSDVSR	R/W	0	SSI时钟预分频因子 该值必须为2-254之间的一个偶数，具体取值由SSIClk的频率决定。执行读操作时，LSB的返回值始终为0。

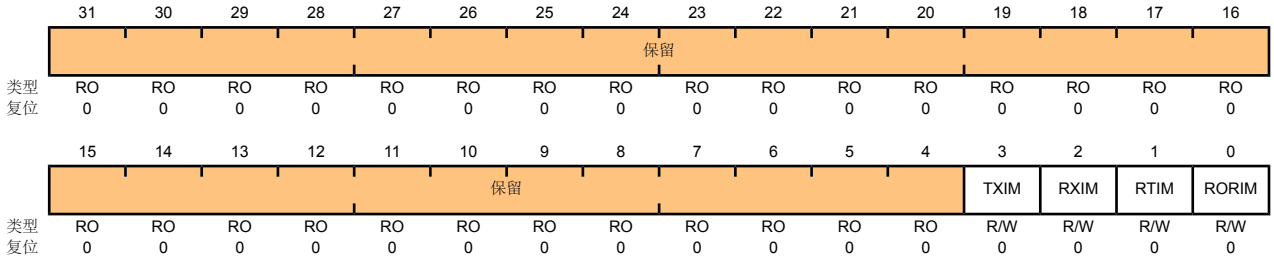
寄存器6: SSI中断屏蔽（SSIIM），偏移量 0x014

SSIIM为中断屏蔽置位或清零寄存器。它是一个读/写寄存器，复位时所有位都清零。

对该寄存器执行读操作将获得相关中断屏蔽的当前值。向寄存器的特定位写1将设置屏蔽，使得中断能够读出。写0可清除对应的中断屏蔽。

SSI中断屏蔽 (SSIIM)

偏移量0x014
类型R/W, 复位0x0000.0000



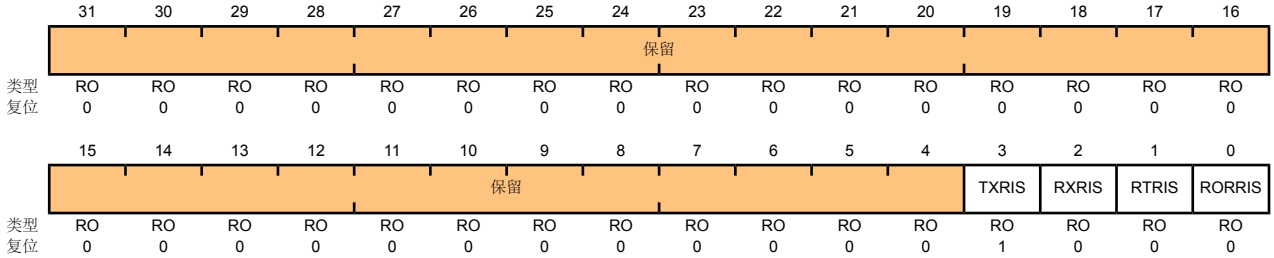
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXIM	R/W	0	SSI发送FIFO中断屏蔽 0: TX FIFO半空或少于半空条件中断被屏蔽。 1: TX FIFO半空或少于半空条件没有被屏蔽。
2	RXIM	R/W	0	SSI接收FIFO中断屏蔽 0: RX FIFO半满或少于半满条件中断被屏蔽。 1: RX FIFO半满或少于半满条件中断没有被屏蔽。
1	RTIM	R/W	0	SSI接收超时中断屏蔽 0: RX FIFO超时中断被屏蔽。 1: RX FIFO超时中断没有被屏蔽。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

寄存器**7: SSI**原始中断状态（**SSIRIS**），偏移量 **0x018**

SSIRIS为原始中断状态寄存器。对其执行读操作可获得对应中断在屏蔽之前的当前原始中断状态值。写操作无效。

SSI原始中断状态 (SSIRIS)

偏移量0x018
类型RO, 复位0x0000.0008



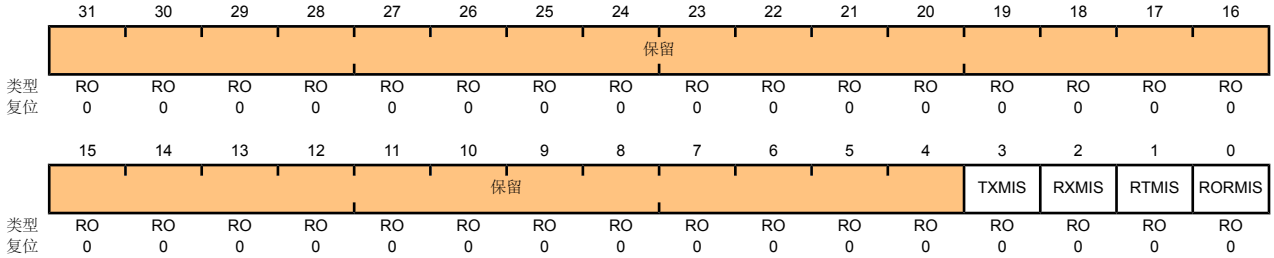
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXRIS	RO	1	SSI发送FIFO原始中断状态 该位置位表示发送FIFO为半空或少于半空。
2	RXRIS	RO	0	SSI接收FIFO原始中断状态 该位置位表示接收FIFO为半空或多于半空。
1	RTRIS	RO	0	SSI接收超时原始中断状态 该位置位表示发生接收超时。
0	RORRIS	RO	0	SSI接收溢出原始中断状态 该位置位表示接收FIFO溢出。

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



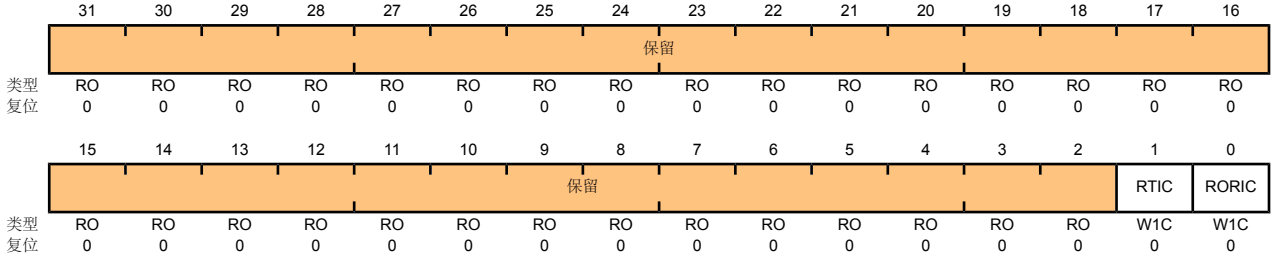
位/域	名称	类型	复位	描述
31:4	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
3	TXMIS	RO	0	SSI发送FIFO屏蔽后的中断状态 该位置位表示发送FIFO为半空或少于半空。
2	RXMIS	RO	0	SSI接收FIFO屏蔽后的中断状态 该位置位表示接收FIFO为半空或多于半空。
1	RTMIS	RO	0	SSI接收超时屏蔽后的中断状态 该位置位表示发生接收超时。
0	RORMIS	RO	0	SSI接收溢出屏蔽后的中断状态 该位置位表示接收FIFO溢出。

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



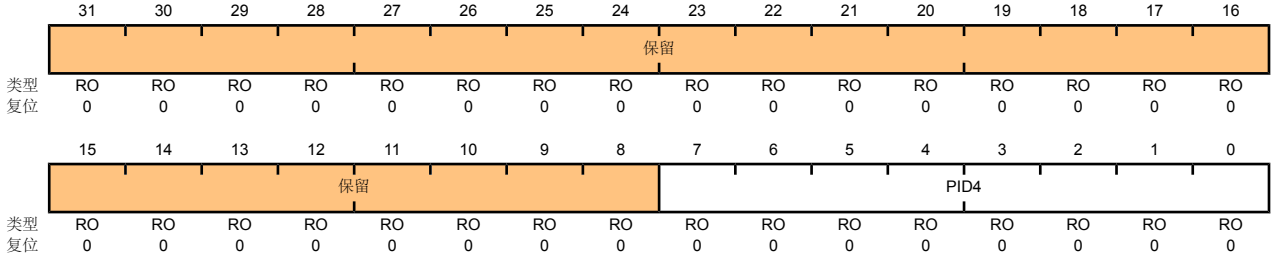
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	RTIC	W1C	0	SSI接收超时中断清零 0: 对中断无影响 1: 清零中断。
0	RORIC	W1C	0	SSI接收溢出中断清零 0: 对中断无影响 1: 清零中断。

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



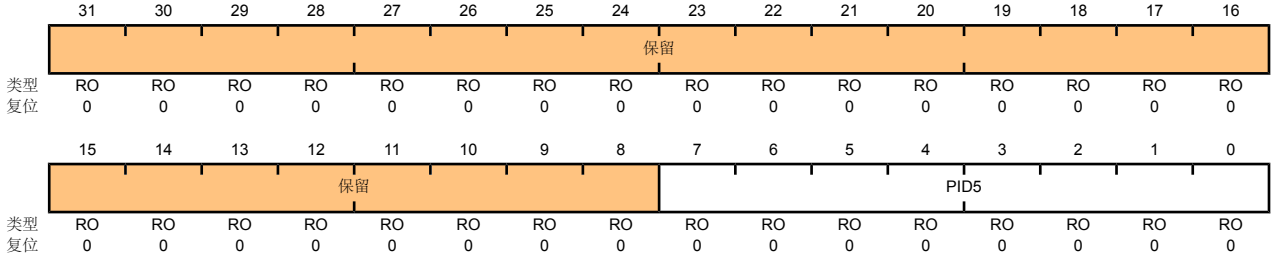
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID4	RO	0x00	SSI外设ID寄存器[7:0] 可被软件用来标识外设的存在。

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



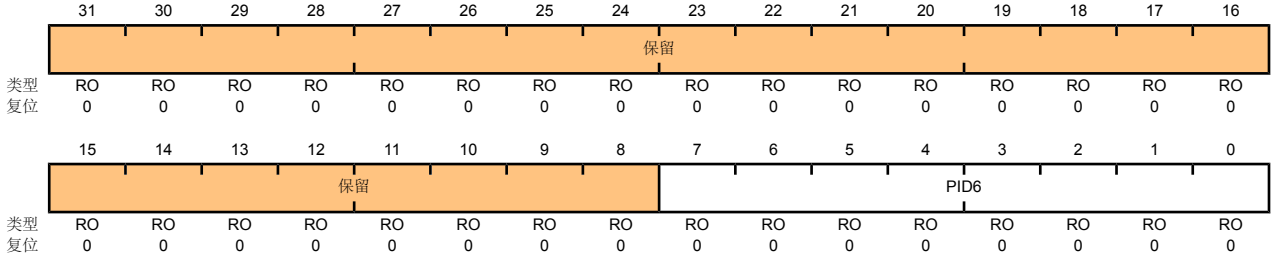
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID5	RO	0x00	SSI外设ID寄存器[15:8] 可被软件用来标识外设的存在。

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



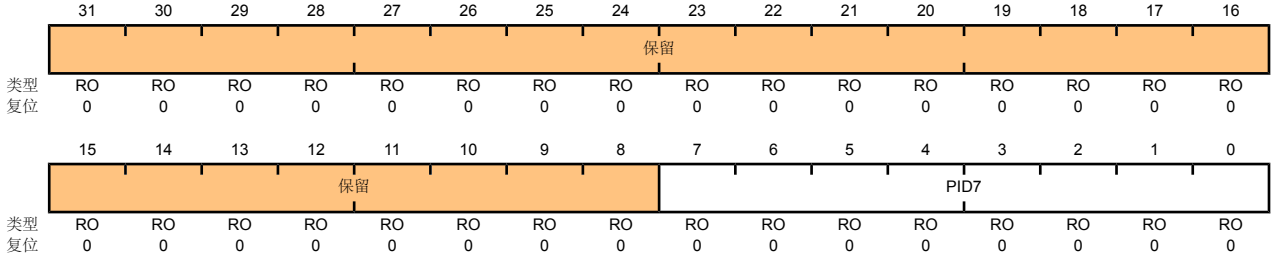
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID6	RO	0x00	SSI外设ID寄存器[23:16] 可被软件用来标识外设的存在。

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



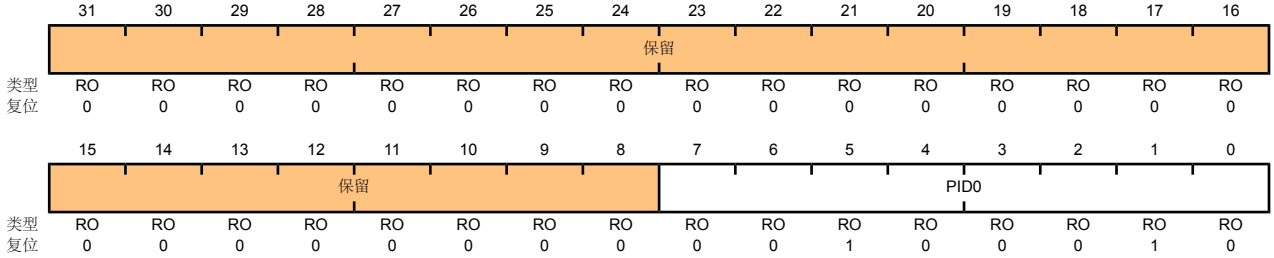
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID7	RO	0x00	SSI外设ID寄存器[31:24] 可被软件用来标识外设的存在。

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



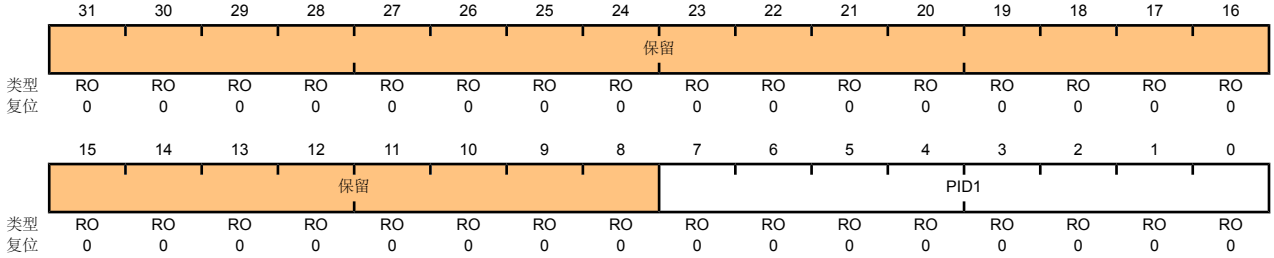
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID0	RO	0x22	SSI外设ID寄存器[7:0] 可被软件用来标识外设的存在。

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



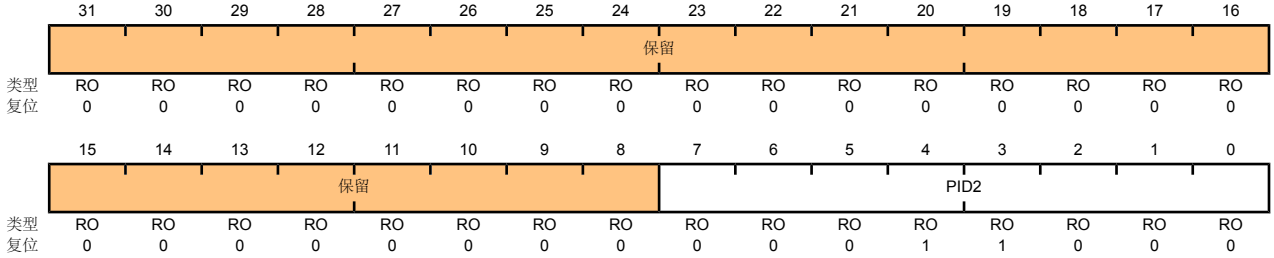
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID1	RO	0x00	SSI外设ID寄存器 [15:8] 可被软件用来标识外设的存在。

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



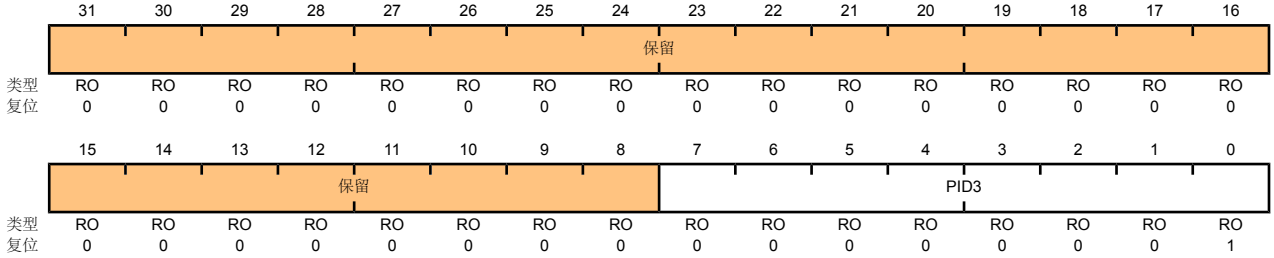
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID2	RO	0x18	SSI外设ID寄存器 [23:16] 可被软件用来标识外设的存在。

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



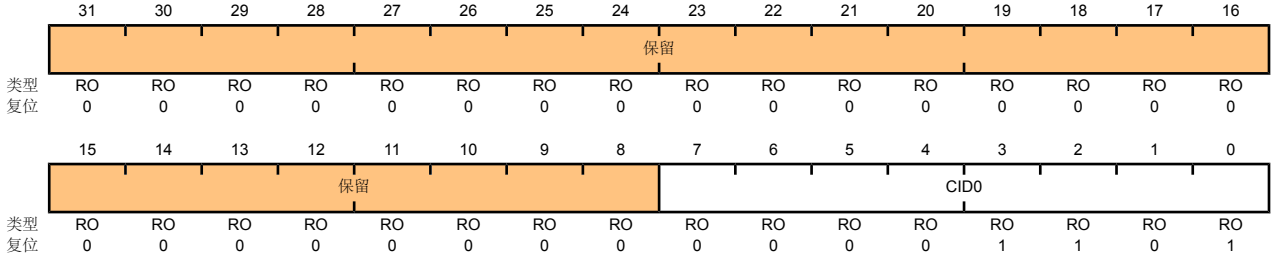
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	PID3	RO	0x01	SSI外设ID寄存器 [31:24] 可被软件用来标识外设的存在。

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



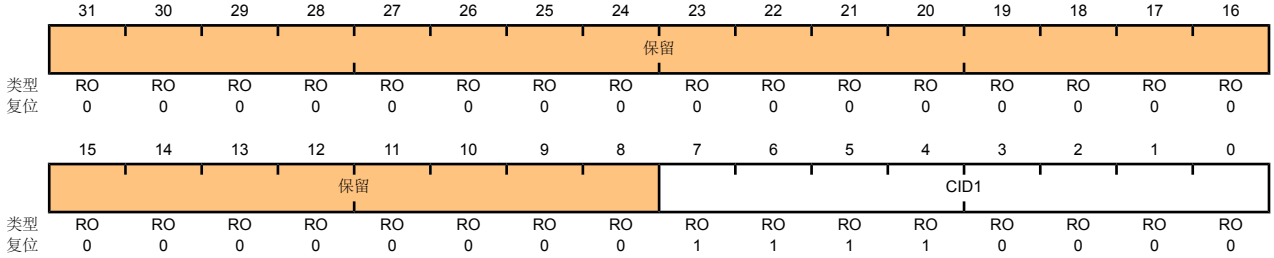
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID0	RO	0x0D	SSI PrimeCell ID 寄存器[7:0] 为软件提供一个标准的交叉外设识别系统。

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



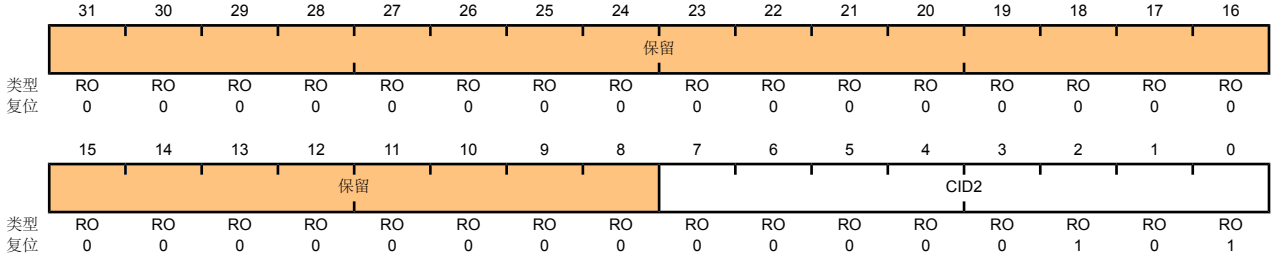
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID1	RO	0xF0	SSI PrimeCell ID 寄存器 [15:8] 为软件提供一个标准的交叉外设识别系统。

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



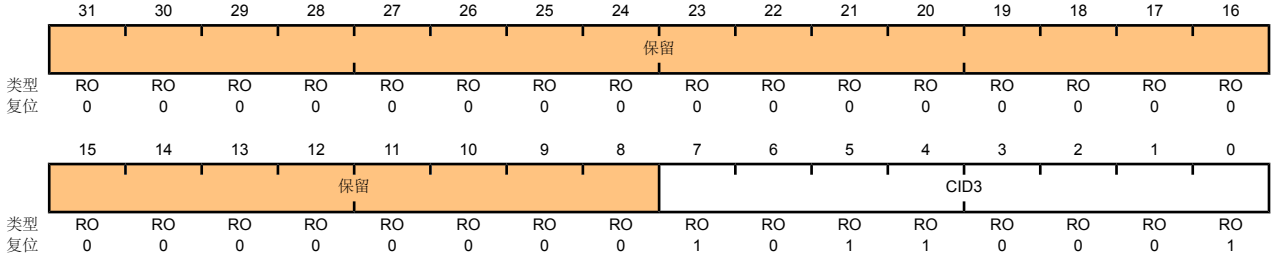
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	CID2	RO	0x05	SSI PrimeCell ID 寄存器 [23:16] 为软件提供一个标准的交叉外设识别系统。

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	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	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。

13 以太网控制器

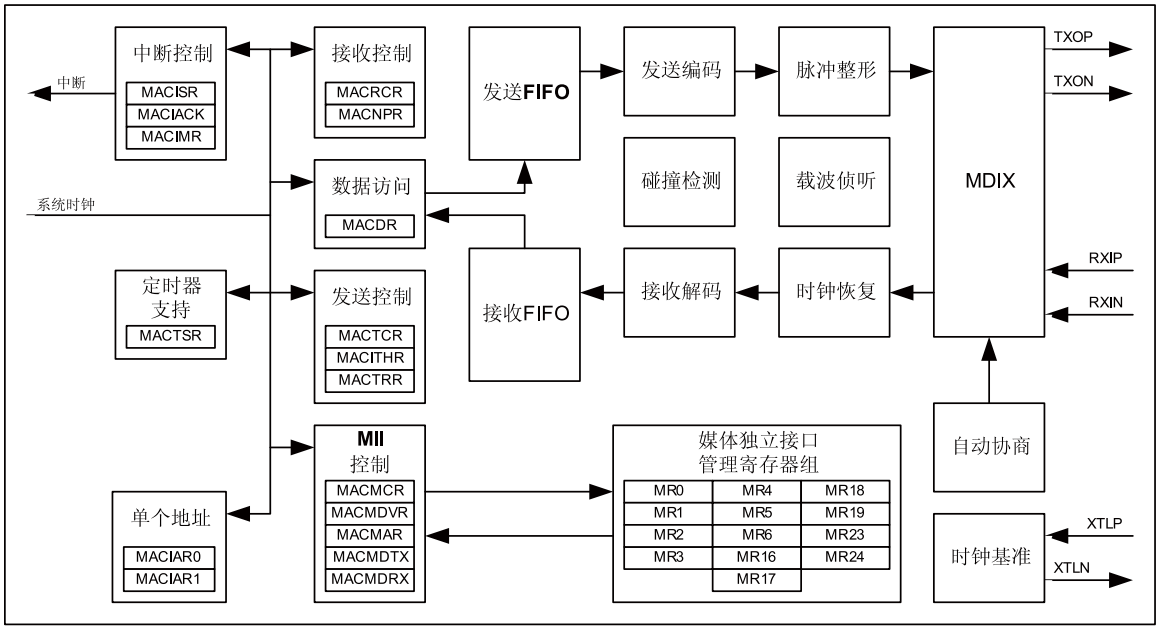
Stellaris® 以太网控制器由一个完全集成的媒体访问控制器（MAC）和网络物理（PHY）接口器件组成。 以太网控制器遵循*IEEE 802.3*规范，完全支持10BASE-T和100BASE-TX标准。

以太网控制器模块具有以下特性：

- 遵循*IEEE 802.3-2002 规范*
  - 遵循10BASE-T/100BASE-TX IEEE-802.3。 只需要一个双路1:1隔离变压器就能与线路相连
  - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX 扰码器/解扰器
  - 全功能的自协商
- 多种工作模式
  - 全双工和半双工100 Mbps
  - 全双工和半双工10 Mbps
  - 节电和掉电模式
- 高度可配置
  - 可编程MAC地址
  - LED活动选择
  - 支持混杂模式
  - CRC错误拒绝控制
  - 用户可配置的中断
- 物理媒体操作
  - 自动MDI/MDI-X交叉校验
  - 寄存器可编程的发送幅度
  - 自动极性校正和10BASE-T信号接收

13.1 方框图

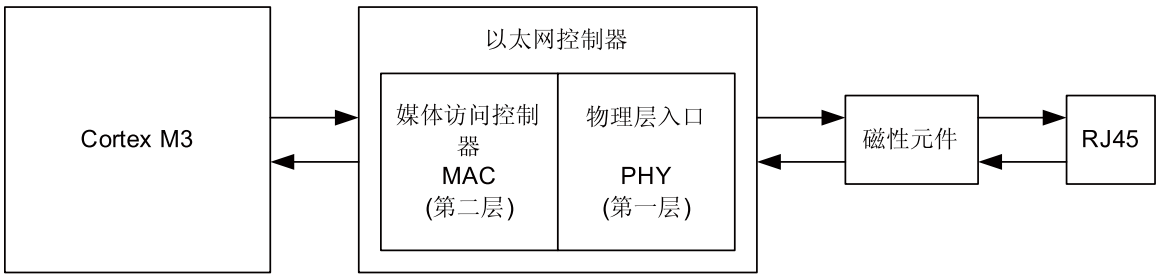
图 **13-1.** 以太网控制器方框图



13.2 功能描述

如图 13-2 在 280页中所示，以太网控制器在功能上被划分为两层或两个模块——媒体访问控制器（MAC）层和网络物理（PHY）层。它们与ISO模型的第2和第1层相对应。以太网控制器的基本接口是到MAC层的一个简单总线接口。 MAC层提供了以太网帧的发送和接收处理。 MAC层还通过一个内部的媒体独立接口（MII）给PHY模块提供接口。

图 **13-2.** 以太网控制器



13.2.1 内部 **MII**操作

为了MII管理接口的正确工作，MDIO信号必须通过一个10k Ω 的上拉电阻连接到+3.3 V的电源。 不连接这个上拉电阻将阻止这个内部MII上的管理传输起作用。 注意：通过MII的数据传输可能仍然起作用，因为默认情况下PHY层将自协商链路参数。

为了使MII管理接口正确工作，内部时钟必须被向下分频，使频率从系统时钟变为一个不大于2.5MHz的频率。 **MACMDV**寄存器包含用来下调系统时钟的分频器。 有关这个寄存器使用的详细情况见 298页。

13.2.2 PHY配置/操作

以太网控制器中的物理层（PHY）包括集成的ENDEC、扰码器/解扰器、双速时钟恢复和全功能自协商功能。 发送器包含一个片内脉冲整形器和一个线路驱动器。 接收器有一个自适应均衡器和一个校准时钟及恢复数据所需的基线恢复电路。 在100BASE-TX应用中，收发器采用5类非屏蔽双绞线（Cat-5 UTP）；在10BASE-T应用中，收发器采用3类非屏蔽双绞线（Cat-3 UTP）。 以太网控制器通过双路1:1隔离变压器连接到线路介质（line media）。 无需外部滤波器。

13.2.2.1 时钟选择

PHY有一个片内晶体振荡器，这个振荡器也可由一个外部振荡器驱动。 在这种模式的工作中，XTLPPHY和XTLNPHY 管脚之间应该连接一个25MHz的晶体。 或者，XTLP管脚也可以连接一个外部25-MHz的时钟输入。 在这种模式的工作中，不需要任何晶体，XTLN管脚必须连接到地。

13.2.2.2 自协商

PHY支持*IEEE 802.3*标准中第28条的自协商功能，可以在铜线电缆上执行10/100Mbps的操作。 这个功能可以通过寄存器设置来使能。 复位后，自协商功能默认开启，**MR0**寄存器的ANEGEN位为高。 软件可以通过写 ANEGEN 位禁止自协商功能。 在自协商过程中**MR4** 寄存器的内容通过快速链路脉冲编码发送给PHY的连接方。

一旦自协商结束，**MR18**寄存器的DPLX和 RATE位就反映了实际速度和选择的双工。 如果由于某种原因自协商未能建立一个链路，则**MR18**寄存器的ANEGF位就会将该情况反映出来，自协商从头开始重新启动。 向**MR0**寄存器的RANEG位写1也能使自协商重新启动。

13.2.2.3 极性校正

PHY能够为10BASE-T执行自动或手动极性翻转，也具有自协商功能。 **MR16**寄存器的位4和5（RVSP0L 和AP0L）控制着这个特性。 默认是自动模式，该模式下AP0L为低，RVSP0L 指明检测电路是否已经将输入信号翻转了。 要进入手动模式， AP0L应当被设置成高， RVSP0L控制着信号的极性。

13.2.2.4 MDI/MDI-X配置

PHY支持IEEE 802.3 2002中定义的自动 MDI/MDI-X 配置。 这就使得在连接到另一个器件（例如，集线器）时无需使用交叉电缆。 通过 **MR24**寄存器中的设置来控制算法。 有关这些设置的其它详细信息请参考 321页。

13.2.2.5 LED 指示器

PHY支持2个LED信号，它们可以用来指示以太网控制器操作的各种状态。 这两个信号对应LED0和LED1管脚。 默认情况下，这些管脚配置用作GPIO信号（PF3和PF2）。 为了使PHY层驱动这两个信号，信号必须被重新配置成硬件功能。 其它的信息请参考GPIO一章。 这些管脚的功能通过PHY层的**MR23**寄存器来编程。 有关如何编程这些LED功能的其它信息请参考 320页。

13.2.3 MAC配置/操作

13.2.3.1 以太网帧格式

以太网数据由以太网帧来传送。 基本的帧格式如图 13-3 在 282页所示。

图 **13-3.** 以太网帧

帧的7个字段从左到右被发送。 帧的位按照最低有效位到最高有效位的方向被发送。

- 前导码

  物理层信号电路使用前导码字段来实现与接收到的帧的时序同步。 前导码的长度为7个字节。

- 起始帧分界符（SFD）

  SFD字段在前导码模式之后，指示帧的开始。 其值为1010.1011。

- 目标地址（DA）

  这个字段指定数据帧的目标地址。 DA的LSB决定地址是一个单个地址（0）还是组/多播地址（1）。

- 源地址（SA）

  源地址字段识别帧启动的站。

- 长度/类型字段

  这个字段的意义由它的数值来决定。 2个字节中的第1个字节是最高有效字节。 这个字段可以解释成长度或类型码。数据字段的最大长度为1500字节。如果长度/类型字段的值小于或等于1500（十进制），则该字段的值就是MAC客户数据的字节数。 如果该字段的值大于或等于1536（十进制），则字段代表的就是类型。 协议标准未定义长度/类型字段的值在1500和1536之间时代表的含义。 如果长度/类型字段的值大于1500（十进制），MAC模块就认定该字段代表的是类型。

- 数据

  数据字段是一个0～1500字节的序列。 由于提供了高度的数据透明度，所以任何值都可以出现在该字段中。 最小的帧尺寸必须满足IEEE标准的要求。 如果必要，可以通过添加一些额外的位来延长数据字段（一次填充）。 填充字段的长度可以为0～46个字节。 数据字段和填充字段长度之和的最小值必须为46个字节。 虽然MAC模块自动插入填充的操作可以通过一个寄存器写来禁能，但是，如果需要，操作仍可执行。 对于MAC模块内核来说，发送/接收的数据可以多于1500字节，不会报告“帧太长”错误。 取而代之的是，在接收到的帧太大而不适合以太网控制器的RAM时报告FIFO溢出错误。

- 帧校验序列（FCS）

  帧校验序列传送循环冗余校验（CRC）值。 这个字段的值使用CRC-32算法通过目标地址、源地址、长度/类型、数据和填充字段计算得到。 MAC模块每次计算半个字节的FCS值。 对于发送的帧，这个字段由MAC层自动插入，除非通过**MACTCTL**寄存器的CRC位将其禁能了。 对于接收到的帧，这个字段被自动校验。 如果FCS校验未通过，帧就不能放置到RX FIFO中，除非FCS校验通过**MACRCTL**寄存器的BADCRC位被禁能。

13.2.3.2 MAC层FIFO

一个2K字节的TX FIFO提供给以太网帧的发送，可以用来存放单个帧。 虽然IEEE 802.3规范限制一个以太网帧的净负荷区的大小为1500字节，但以太网控制器并没有给出这样的限制。 整个缓冲区都可以使用，净负荷区高达2032字节。

一个2K字节的RX FIFO提供给以太网帧接收，可以用来保存多个帧（最多可高达31个帧）。 如果接收到一个帧而RX FIFO没有足够的空间来存放，则会指示溢出错误。

有关TX和RX FIFO分布的详细信息，请参考表 13-1 在 283页。 请注意以下所描述的TX FIFO和RX FIFO分布之间的不同。 对于TX FIFO来说，第一个FIFO字中的数据长度字段指的是以太网帧的数据净负荷，如第5个到第n个FIFO位置中所示。 对于RX FIFO来说，帧长度字段是包括FCS和帧长度字节在内的接收到的以太网的总长度。 还要注意，如果FCS的产生通过**MACTCTL**寄存器的 CRC位被禁止，则FIFO中的最后一个字必须是已经写入FIFO的帧的FCS字节。

需要注意的还有：如果数据净负荷区的长度不是4的倍数，则FCS字段将和FIFO中的字重叠。 但是，对于RX FIFO，下个数据帧的开始总是位于一个字的边界处。

表 **13-1. TX & RX FIFO** 的组织结构

FIFO 字读/写序列	字位域	**TX FIFO**（写）	**RX FIFO**（读）
第1个字	7:0	数据长度LSB	帧长度LSB
	15:8	数据长度 MSB	帧长度MSB
	23:16	DA字节1	
	31:24	DA字节 2	
第2个字	7:0	DA字节3	
	15:8	DA字节4	
	23:16	DA字节5	
	31:24	DA字节6	
第3个字	7:0	SA 字节1	
	15:8	SA字节2	
	23:16	SA字节3	
	31:24	SA字节4	
第4个字	7:0	SA字节5	
	15:8	SA字节6	
	23:16	长度/类型MSB	
	31:24	长度/类型LSB	
第5～n个字	7:0	数据字节n	
	15:8	数据字节 n+1	
	23:16	数据字节n+2	
	31:24	数据字节n+3	
最后一个字	7:0	FCS 1（如果 CRC的产生在MACTCTL中被禁止）	FCS 1
	15:8	FCS 2 （如果 CRC的产生在MACTCTL中被禁止）	FCS 2
	23:16	FCS 3 （如果 CRC的产生在MACTCTL中被禁止）	FCS 3
	31:24	FCS 4 （如果 CRC的产生在MACTCTL中被禁止）	FCS 4

13.2.3.3 以太网发送选择

以太网控制器可以在发送帧结束时自动产生和插入帧校验序列（FCS）。 这由**MACTCTL**寄存器的CRC位来控制。 出于测试的目的，为了产生一个带有无效CRC的帧，这个特性可以被禁止。

IEEE 802.3规范要求以太网帧的净负荷区的大小最小为46字节。 如果装入FIFO的净负荷数据区小于最小的46字节，则以太网控制器可以配置成自动填充数据区。 这个特性由**MACTCTL**寄存器的PADEN位来控制。

在MAC层，发送器可以通过使用**MACTCTL**寄存器的 DUPLEX位配置成既执行全双工，又执行半双工操作。

13.2.3.4 以太网接收选择

利用**MACRCTL**寄存器的BADCRC位，以太网控制器可以配置成拒绝到来的带有无效帧校验序列字段的以太网帧。

以太网接收器也可以用**MACRCTL**寄存器的PRMS和AMUL域配置成混杂模式和多播模式。 如果这些模式都被禁止，那么只有带有广播地址的以太网帧或与编程到**MACIA0**和**MACIA1**寄存器的MAC地址相匹配的帧被放置到RX FIFO中。

13.2.4 中断

在下面的一个或多个条件出现时以太网控制器产生中断：

- 一个空RX FIFO接收到一个帧
- 出现了帧发送错误
- 成功发送完一个帧
- RX FIFO中没有空间时接收到了一个帧（FIFO溢出）
- 接收到一个伴随一个或多个错误条件的帧（例如，FCS失败）
- MAC和PHY层之间的MII管理传输已经结束
- 一个或多个下面的PHY层条件出现：
  - 自协商结束
  - 远程故障
  - 连接状态改变
  - 连接方应答
  - 并行检测故障
  - 接收到页
  - 接收错误
  - 检测到Jabber事件

13.3 初始化和配置

要使用以太网控制器，外设必须通过置位RCGC2寄存器的ETH位来使能。 然后，使用以下步骤来配置以太网控制器执行基本的操作。

1. 编程**MACDIV**寄存器在内部MII上获得一个2.5MHz的时钟（或更小的时钟）。 假设系统时钟为20MHz，则**MACDIV**的值就是4。

2. 编程**MACIA0**和**MACIA1**寄存器进行地址过滤。
3. 使用值0x16编程**MACTCTL**寄存器，实现自动CRC产生、填充和全双工操作。
4. 使用值0x08编程**MACRCTL**寄存器来拒绝带有坏FCS的帧。
5. 通过置位**MACTCTL**和**MACRCTL**寄存器的LSB来使能发送器和接收器。
6. 要发送一个帧，就使用**MACDATA**寄存器将该帧写入TX FIFO。然后置位**MACTR**寄存器的NEWTX位启动发送过程。当NEWTX位被清零后，TX FIFO就可用于下个帧的发送。
7. 要接收一个帧，就必须等到**MACNP** 寄存器的NPR域为非零值。然后使用**MACDATA**寄存器开始将帧从RX FIFO中读出。当帧（包括FCS字段在内）被读取后，NPR域的值应当减1。当RX FIFO中没有帧时，NPR域将读出为零。

13.4 以太网寄存器映射

表 13-2 在 285页列出了以太网MAC寄存器。所有给出的地址都是相对于0x4004.8000的以太网MAC基址而言的。

*IEEE 802.3*标准指定了一个寄存器集合，用来控制和集中PHY的状态。这些寄存器被共同称为MII管理寄存器，在*IEEE 802.3* 规范的22.2.4节中对它们进行了详细描述。表 13-2 在 285页列出了MII管理寄存器。给出的所有地址都是绝对地址，可以直接写入**MACMCTL**寄存器的REGADR域。寄存器0～15的格式由IEEE规范定义，为所有PHY实现（PHY implementation）所共用。存在的唯一不同是某些特性，特定的PHY可能支持、也可能不支持。寄存器16～31是厂商特有的寄存器，用来支持厂商PHY实现特有的特性。未列出的厂商特有的寄存器被保留。

表 **13-2.** 以太网寄存器映射

偏移量	名称	类型	复位	描述	见页面
以太网**MAC**					
0x000	MACRIS	RO	0x0000.0000	以太网MAC原始中断状态	287
0x000	MACIACK	W1C	0x0000.0000	以太网MAC中断应答	289
0x004	MACIM	R/W	0x0000.007F	以太网MAC中断屏蔽	290
0x008	MACRCTL	R/W	0x0000.0008	以太网MAC接收控制	291
0x00C	MACTCTL	R/W	0x0000.0000	以太网MAC发送控制	292
0x010	MACDATA	R/W	0x0000.0000	以太网MAC数据	293
0x014	MACIA0	R/W	0x0000.0000	以太网MAC单个地址0	294
0x018	MACIA1	R/W	0x0000.0000	以太网MAC单个地址1	295
0x01C	MACTHR	R/W	0x0000.003F	以太网MAC阈值	296
0x020	MACMCTL	R/W	0x0000.0000	以太网MAC管理控制	297
0x024	MACMDV	R/W	0x0000.0080	以太网MAC管理分频器	298
0x028	MACMADD	RO	0x0000.0000	以太网MAC管理地址	299
0x02C	MACMTXD	R/W	0x0000.0000	以太网MAC管理发送数据	300
0x030	MACMRXD	R/W	0x0000.0000	以太网MAC管理接收数据	301
0x034	MACNP	RO	0x0000.0000	以太网MAC的包数目	302

偏移量	名称	类型	复位	描述	见页面
0x038	MACTR	R/W	0x0000.0000	以太网MAC发送请求	303
MII管理					
0x00	MR0	R/W	0x3100	以太网PHY管理寄存器0 – 控制	304
0x01	MR1	RO	0x7849	以太网PHY管理寄存器1 – 状态	306
0x02	MR2	RO	0x000E	以太网PHY管理寄存器2 – PHY标识符1	308
0x03	MR3	RO	0x7237	以太网PHY管理寄存器3 – PHY标识符2	309
0x04	MR4	R/W	0x01E1	以太网PHY管理寄存器4 – 自协商通告	310
0x05	MR5	RO	0x0000	以太网PHY管理寄存器5 – 自协商连接方基页能力	312
0x06	MR6	RO	0x0000	以太网PHY管理寄存器6 – 自协商扩展	313
0x10	MR16	R/W	0x0140	以太网PHY管理寄存器16 – 厂商特定	314
0x11	MR17	R/W	0x0000	以太网PHY管理寄存器17 – 中断控制/状态	316
0x12	MR18	RO	0x0000	以太网PHY管理寄存器18 – 诊断	318
0x13	MR19	R/W	0x4000	以太网PHY管理寄存器19 – 收发器控制	319
0x17	MR23	R/W	0x0010	以太网PHY管理寄存器23 – LED配置	320
0x18	MR24	R/W	0x00C0	以太网PHY管理寄存器24 –MDI/MDIX控制	321

13.5 以太网MAC寄存器描述

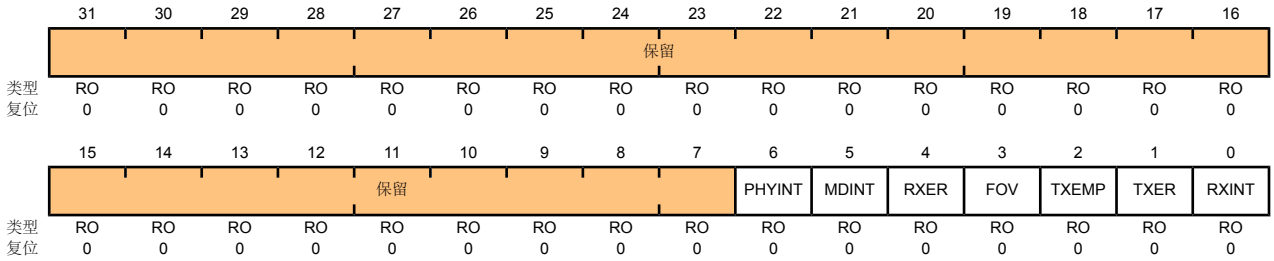
本节的剩余部分按照地址偏移量的数值顺序列出和描述以太网MAC寄存器。

寄存器1: 以太网MAC原始中断状态（MACRIS），偏移量 0x000

MACRIS寄存器是中断状态寄存器。 读操作时，该寄存器给出了屏蔽之前相应中断的当前状态值。

以太网MAC原始中断状态 (MACRIS)

偏移量0x000
类型RO, 复位0x0000.0000



位/域	名称	类型	复位	描述
31:7	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	PHYINT	RO	0x0	PHY中断 该位置位时表明PHY层的一个已使能的中断产生了。 必须读取PHY的MR17来确定触发这个中断的特定PHY事件。
5	MDINT	RO	0x0	MII传输结束 该位置位时表明MII接口上的传输（读或写）已经成功完成。
4	RXER	RO	0x0	接收错误 这个位表明接收器碰到了错误。 可以造成这个中断位置位的可能错误有: ■ 在一个帧接收的过程中出现接收错误（仅限于100 Mb/s的情况）。 ■ 由于对齐错误造成帧不是整数个字节（dribble bits）。 ■ 帧的CRC没有通过FCS校验。 ■ 当长度/类型字段解释为长度字段时与帧数据大小不符。
3	FOV	RO	0x0	FIFO溢出 该位置位时表明接收FIFO遇到溢出错误。
2	TXEMP	RO	0x0	发送FIFO为空 该位置位时表明包已被发送，TX FIFO为空。
1	TXER	RO	0x0	发送错误 该位置位时表明发送器遇到错误。 可以造成这个中断位置位的可能错误有: ■ 存放在TX FIFO中的数据长度字段超过2032。 这个错误出现时帧不发送。 ■ 在等待重传（backoff）过程中重新发送的尝试次数超出了最大限制16。

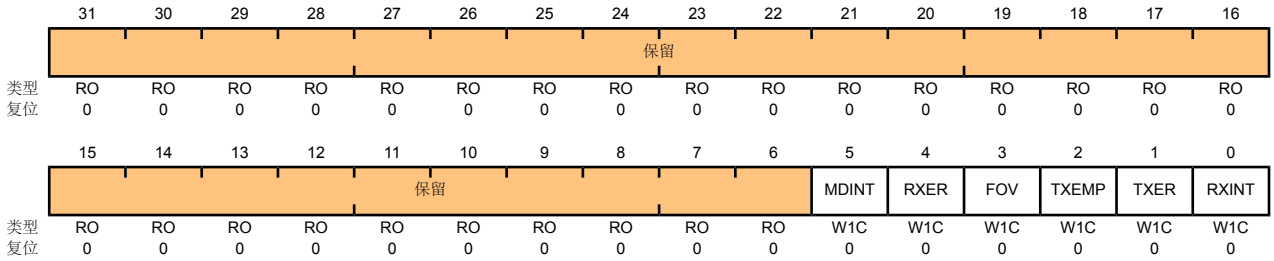
位/域	名称	类型	复位	描述
0	RXINT	RO	0x0	接收到包 该位置位时表明至少已经接收到一个包，并且包已经存放到接收FIFO中。

寄存器2: 以太网MAC中断应答（MACIACK），偏移量 0x000

向这个寄存器的任何位写入1会清零以太网**MAC**原始中断状态（**MACRIS**）寄存器中相应的中断位。

以太网MAC中断应答 (MACIACK)

偏移量0x000
类型W1C, 复位0x0000.0000



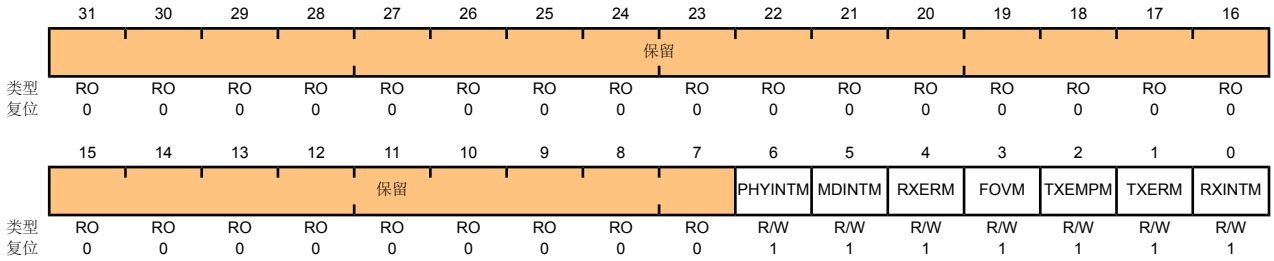
位/域	名称	类型	复位	描述
31:6	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PHYINT	W1C	0x0	清除PHY中断 向PHYINT位写入1清除从**MACRIS**寄存器读出的PHYINT 中断。
5	MDINT	W1C	0x0	清除MII传输结束中断 向MDINT位写入1清除从 **MACRIS** 寄存器读出的MDINT中断。
4	RXER	W1C	0x0	清除接收错误中断 写1到 RXER位清除从**MACRIS**寄存器读出的RXER中断。
3	FOV	W1C	0x0	清除FIFO溢出中断 向FOV位写入1清除从 **MACRIS**寄存器读出的FOV 中断。
2	TXEMP	W1C	0x0	清除发送FIFO为空中断 向TXEMP位写入1清除从 **MACRIS** 寄存器读出的TXEMP 中断。
1	TXER	W1C	0x0	清除发送错误中断 向TXER位写入1清除从**MACRIS**寄存器读出的TXER中断并复位TX FIFO写指针。
0	RXINT	W1C	0x0	清除接收到包中断 向RXINT位写入1清除从**MACRIS**寄存器读出的RXINT中断。

寄存器3: 以太网MAC中断屏蔽（MACIM），偏移量 0x004

这个寄存器允许软件来使能/禁能以太网MAC中断。 写0禁能中断，写1使能中断。

以太网MAC中断屏蔽 (MACIM)

偏移量0x004
类型R/W, 复位0x0000.007F



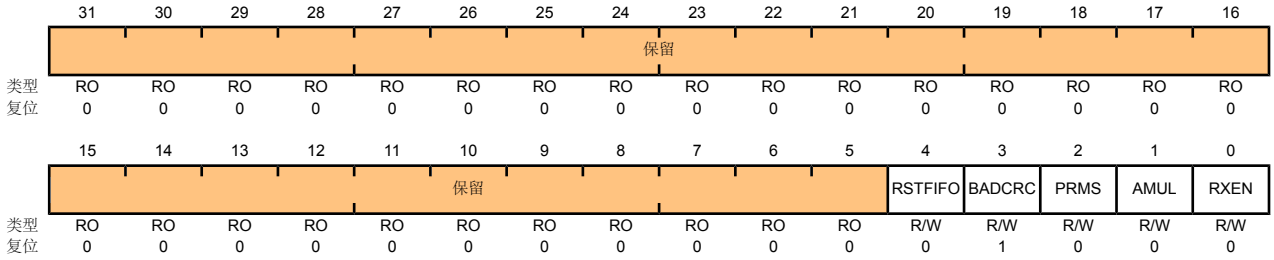
位/域	名称	类型	复位	描述
31:7	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
6	PHYINTM	R/W	1	屏蔽PHY中断 PHYINTM位屏蔽 **MACRIS**寄存器的 PHYINT 位防止它有效。
5	MDINTM	R/W	1	屏蔽MII传输结束中断 MDINTM位屏蔽**MACRIS** 寄存器的MDINT位防止它有效。
4	RXERM	R/W	1	屏蔽接收错误中断 RXERM位屏蔽**MACRIS** 寄存器的RXER位防止它有效。
3	FOVM	R/W	1	屏蔽FIFO溢出中断 FOVM位屏蔽**MACRIS**寄存器的FOV位防止它有效。
2	TXEMPM	R/W	1	屏蔽发送FIFO为空中断 TXEMPM位屏蔽**MACRIS**寄存器的TXEMP 位防止它有效。
1	TXERM	R/W	1	屏蔽发送错误中断 TXERM位屏蔽**MACRIS**寄存器的TXER位防止它有效。
0	RXINTM	R/W	1	屏蔽接收到包中断 RXINTM位屏蔽**MACRIS**寄存器的RXINT位防止它有效。

寄存器4: 以太网MAC接收控制（MACRCTL），偏移量 0x008

这个寄存器使能软件来配置接收模块和控制从物理媒体接收到的帧的类型。 重点注意的是：当接收模块使能时，即使AMUL 位不置位，所有在目标地址字段中带有一个FF-FF-FF-FF-FF-FF 广播地址的有效帧都将被接收到，并存放到RX FIFO中。

以太网MAC接收控制 (MACRCTL)

偏移量0x008
类型R/W, 复位0x0000.0008

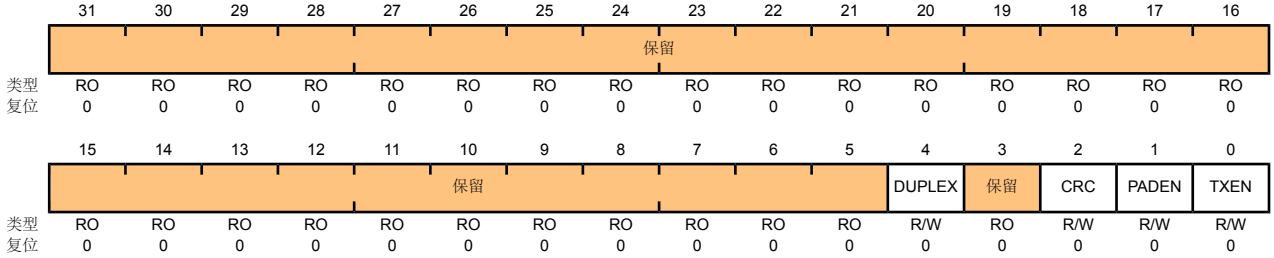


位/域	名称	类型	复位	描述
31:5	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
4	RSTFIFO	R/W	0x0	清空接收FIFO 该位置位时清空接收FIFO。 这个操作在执行软件初始化时执行。 建议先禁能接收器（RXEN = 0），然后再启动复位（RSTFIFO = 1）。这个序列将清空和复位RX FIFO。
3	BADCRC	R/W	0x1	使能拒绝坏CRC BADCRC位使能拒绝一个带有计算错误的CRC的帧。
2	PRMS	R/W	0x0	使能混杂模式 PRMS位使能混杂模式（Promiscuous mode），接受所有有效的帧，不管它们的目标地址是什么。
1	AMUL	R/W	0x0	使能多播帧 AMUL位使能物理媒体多播帧的接收。
0	RXEN	R/W	0x0	使能接收器 RXEN 位使能以太网接收器。 当该位为低时，接收器被禁能，物理媒体上的所有帧都被忽略。

寄存器5: 以太网MAC发送控制（MACTCTL），偏移量 0x00C

这个寄存器使能软件来配置发送模块，控制帧被放置到物理媒体上。

以太网MAC发送控制 (MACTCTL)
偏移量0x00C
类型R/W, 复位0x0000.0000



位/域	名称	类型	复位	描述
31:5	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。
4	DUPLEX	R/W	0x0	使能双工模式 该位置位时使能双工模式，允许同步发送和接收。
3	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。
2	CRC	R/W	0x0	使能CRC产生 该位置位时使能CRC的自动产生以及在包末尾的放置。 如果该位不置位，则放置到TX FIFO中的帧将在它们写入FIFO的同时被发送。
1	PADEN	R/W	0x0	使能包填充 该位置位时使能不符合最小帧尺寸要求的包的自动填充。
0	TXEN	R/W	0x0	使能发送器 置位时使能发送器。 该位为0时禁能发送器。

寄存器6: 以太网MAC数据（MACDATA），偏移量 0x010

这个寄存器使能软件来访问TX和RX FIFO。

读这个寄存器返回存放在读指针指示位置处RX FIFO中的数据。

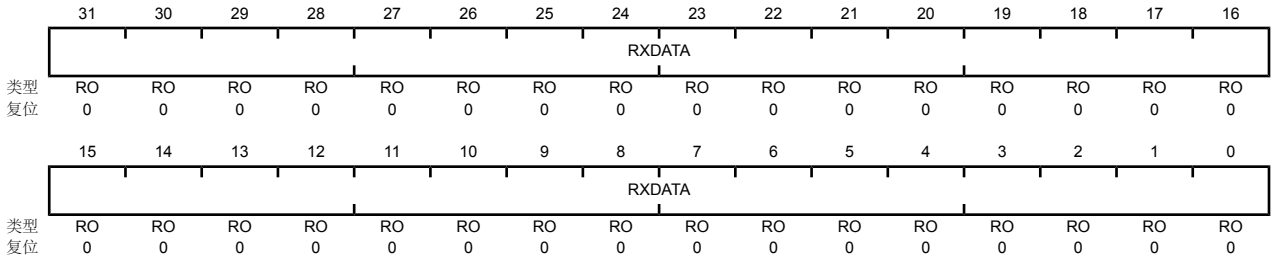
写这个寄存器将数据保存到写指针指示的位置处的TX FIFO中。 然后写指针自动递增指向下个TX FIFO单元。

在这里没有这样的一种机制，可以随机访问RX或FX FIFO中的字节。 数据必须从RX FIFO中顺序读出，然后保存到缓冲区中执行进一步的处理。 一旦执行完一个读操作，FIFO中的数据就不能再次被读取。 数据必须被顺序地写入TX FIFO。 如果在将帧放置到TX FIFO中时出错，那么写指针可以通过写**MACIACK**寄存器的TXER位复位到指向TX FIFO起始处，数据被再次写入。

只读寄存器

以太网MAC数据 (MACDATA)

偏移量0x010
类型RO, 复位0x0000.0000

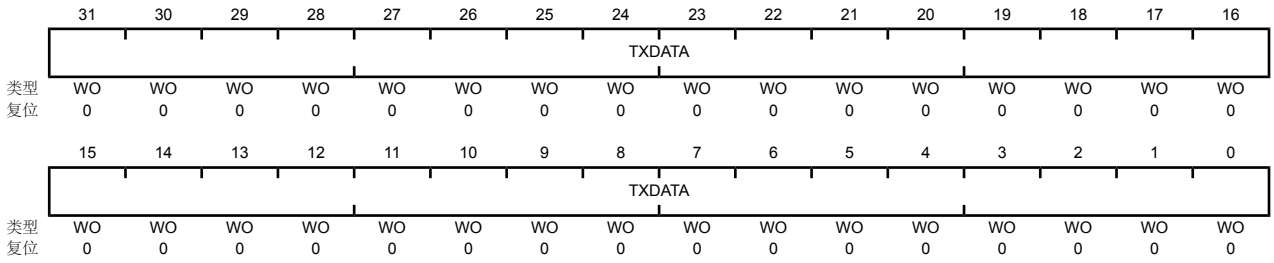


位/域	名称	类型	复位	描述
31:0	RXDATA	RO	0x0	接收FIFO数据 RXDATA位代表存放在RX FIFO中的接下来的4个字节的数据。

只写寄存器

以太网MAC数据 (MACDATA)

偏移量0x010
类型WO, 复位0x0000.0000



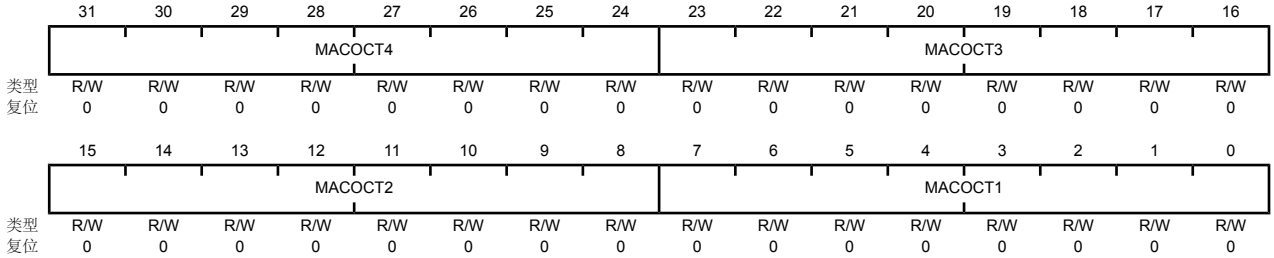
位/域	名称	类型	复位	描述
31:0	TXDATA	WO	0x0	发送FIFO数据 TXDATA位代表要发送出去的放置在TX FIFO中的接下来的4个字节的数据。

寄存器7: 以太网MAC单个地址0（MACIA0），偏移量 0x014

这个寄存器使能软件来编程网络接口卡（NIC）硬件MAC地址的前4个字节。 将6字节的IAR与到来的目标地址字段相比较来确定是否应该接收帧。

以太网MAC单个地址0 (MACIA0)

偏移量0x014
类型R/W, 复位0x0000.0000



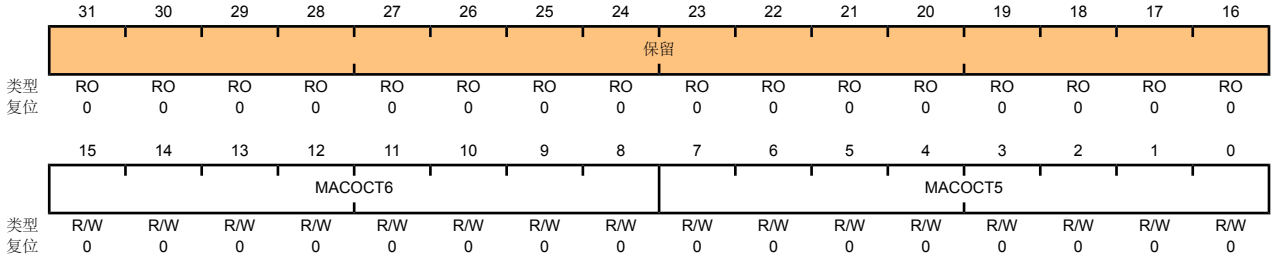
位/域	名称	类型	复位	描述
31:24	MACOCT4	R/W	0x0	MAC地址字节4 MACOCT4位代表MAC地址的第4个字节，该字节用来唯一确定每个以太网控制器。
23:16	MACOCT3	R/W	0x0	MAC地址字节3 MACOCT3位代表MAC地址的第3个字节，该字节用来唯一确定每个以太网控制器。
15:8	MACOCT2	R/W	0x0	MAC地址字节2 MACOCT2 位代表MAC地址的第2个字节，该字节用来唯一确定一个以太网控制器。
7:0	MACOCT1	R/W	0x0	MAC字节地址1 MACOCT1位代表MAC地址的第1个字节，该字节用来唯一确定每个以太网控制器。

寄存器8: 以太网MAC单个地址1（MACIA1），偏移量 0x018

这个寄存器使能软件编程以太网接口卡（NIC）的硬件MAC地址的最后2个字节。 将6字节的IAR与到来的目标地址字段相比较来确定是否应该接收帧。

以太网MAC单个地址1 (MACIA1)

偏移量0x018
类型R/W, 复位0x0000.0000



位/域	名称	类型	复位	描述
31:16	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。
15:8	MACOCT6	R/W	0x0	MAC地址字节6 MACOCT6位代表第6个字节的MAC地址，该字节用来唯一确定每个以太网控制器。
7:0	MACOCT5	R/W	0x0	MAC地址字节5 MACOCT5位代表第5个字节的MAC地址，该字节用来唯一确定每个以太网控制器。

寄存器9: 以太网MAC阈值（MACTHR），偏移量 0x01C

这个寄存器使能软件来设置阈值级别（threshold level），帧的发送在此阈值级别处开始。 如果THRESH被设置成复位值0x3F，则发送直到 **MACTR**寄存器的NEWTX位置位时才开始。 这可以有效禁止早发送特性。

向THRESH写入除‘全1’之外的任何值来使能早发送特性。 一旦TX FIFO中数据的字节计数到达这个级别，帧发送就开始。 当THRESH被设置成‘全0’时，帧发送在4个字节（1次写）保存到TX FIFO中之后开始。 THRESH位域的值每次都等到另外32个字节（8次写）的数据保存到TX FIFO之后才递增。 因此，THRESH的值0x01要等待36个字节数据的写入；THRESH的值0x02要等待68个字节数据的写入。 总之，当下面的条件满足时启动早发送：

字节的数目 ≥ 4 (*THRESH* x 8 + 1)

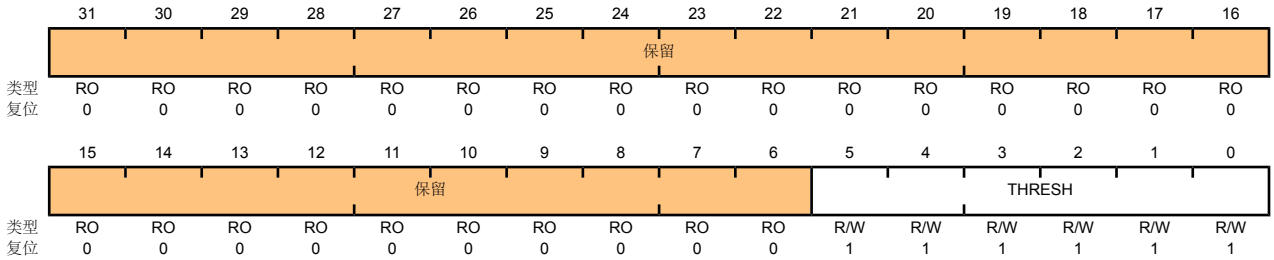
到达阈值级别的效果等同于置位 **MACTR**寄存器的NEWTX位。 帧的发送开始，然后数据长度字段指示的字节数被发送到物理媒体上。 由于未执行欠载检查，尾指针可能可以到达和经过TX FIFO的写指针处。 这会造成不确定的值被写入到物理媒体，而不是帧的末尾处。 因此，软件必须确保足够的总线宽度来写TX FIFO。

如果一个帧比需要发送的阈值级别小，则必须用一个明确的写操作来置位**MACTR** 寄存器的NEWTX位。 这样，即使还未达到阈值限制，也启动帧的发送。

如果阈值级别设置地过小，发送器可能会欠载执行。 一旦发生这种情况，发送帧将被终止，指示一个发送错误。

以太网MAC阈值 (MACTHR)

偏移量0x01C
类型R/W, 复位0x0000.003F



位/域	名称	类型	复位	描述
31:6	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。
5:0	THRESH	R/W	0x3F	阈值 THRESH 位代表早发送阈值。 一旦TX FIFO中的数据量超过这个值，包发送就开始。

寄存器10: 以太网MAC管理控制（MACMCTL），偏移量 0x020

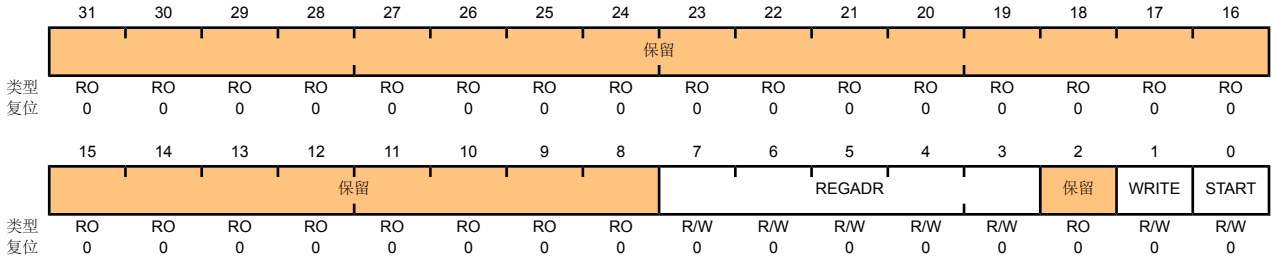
这个寄存器使能软件来控制数据传输到以太网PHY的MII管理寄存器以及来自MII管理寄存器的数据的传输。每个这样的寄存器的地址、名称、类型、复位配置和功能描述请见 表 13-2 在 285页和“MII管理寄存器描述” 在 303页。

为了启动一个来自MII管理寄存器的*读*传输，WRITE位必须在START位被写入1的同一个周期内被写入0。

为了启动一个到MII管理寄存器的*写*传输，WRITE 位必须在START位被写入1的同一个周期内被写入1。

以太网MAC管理控制 (MACMCTL)

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	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。
7:3	REGADR	R/W	0x0	MII寄存器地址 REGADR位域代表的是下个MII管理接口传输的MII管理寄存器地址。
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	WRITE	R/W	0x0	MII寄存器传输类型 WRITE位代表的是下个MII管理接口传输的操作。 如果WRITE被置位，下个操作将是写操作；否则是读操作。
0	START	R/W	0x0	MII寄存器传输使能 START位代表的是下个MII管理接口传输的启动。 当该位被写入1时，位于REGADR的MII寄存器将被读出（WRITE=0）或写入（WRITE=1）。

寄存器11: 以太网MAC管理分频器（MACMDV），偏移量 0x024

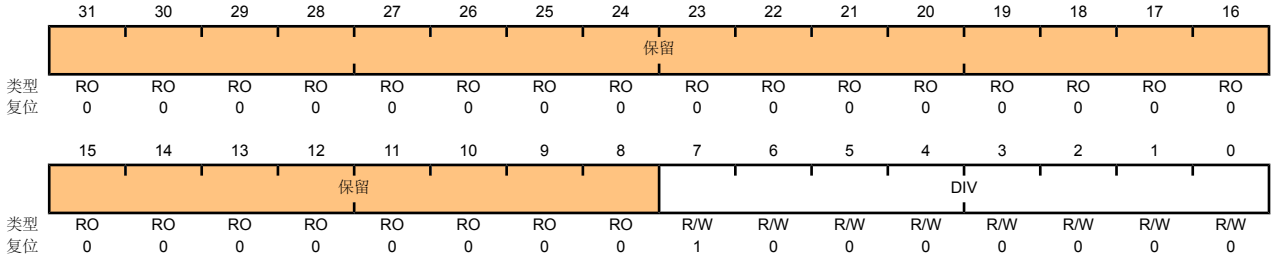
这个寄存器使能软件为管理数据时钟（MDC）设置时钟分频器。 这个时钟用来同步系统和MII管理寄存器之间的读和写传输。 MDC时钟的频率可以通过下式计算出来：

$$F_{mdc} = F_{ipclk} / (2 * (MACMDVR + 1))$$

时钟分频器必须被写入这样一个值，该值必须保证MDC时钟不超过2.5MHz的频率。

以太网MAC管理分频器 (MACMDV)

偏移量0x024
类型R/W, 复位0x0000.0080



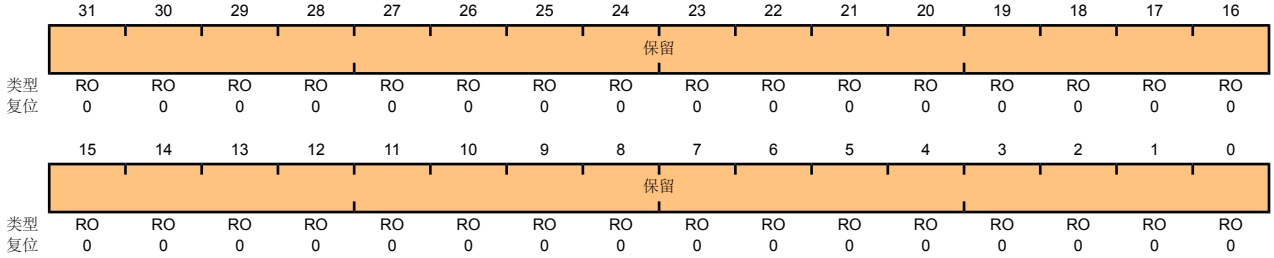
位/域	名称	类型	复位	描述
31:8	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
7:0	DIV	R/W	0x80	时钟分频器 DIV位用来设置MDC时钟的时钟分频器，MDC时钟供MAC和PHY两者在串行MII接口上互相发送数据使用。

寄存器12: 以太网MAC管理地址（MACMADD），偏移量 0x028

这个寄存器使能软件为下个MII管理寄存器传输选择PHY的地址。

以太网MAC管理地址 (MACMADD)

偏移量0x028
类型RO, 复位0x0000.0000



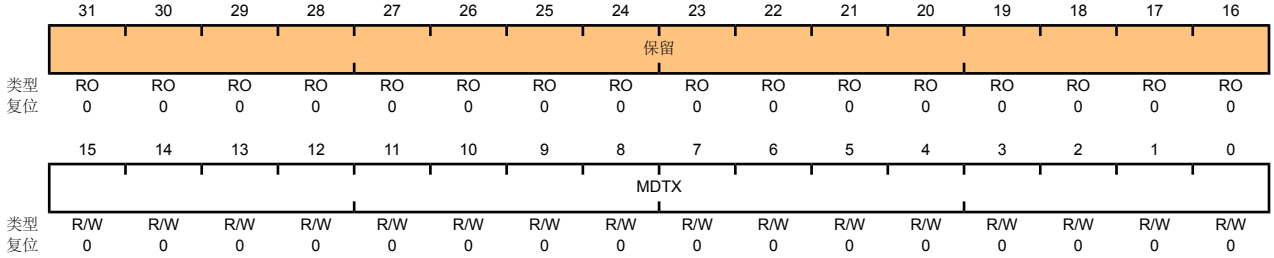
位/域	名称	类型	复位	描述
31:0	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器13: 以太网MAC管理发送数据（MACMTXD），偏移量 0x02C

这个寄存器存放着写入MII管理寄存器的下个值。

以太网MAC管理发送数据 (MACMTXD)

偏移量0x02C
类型R/W, 复位0x0000.0000



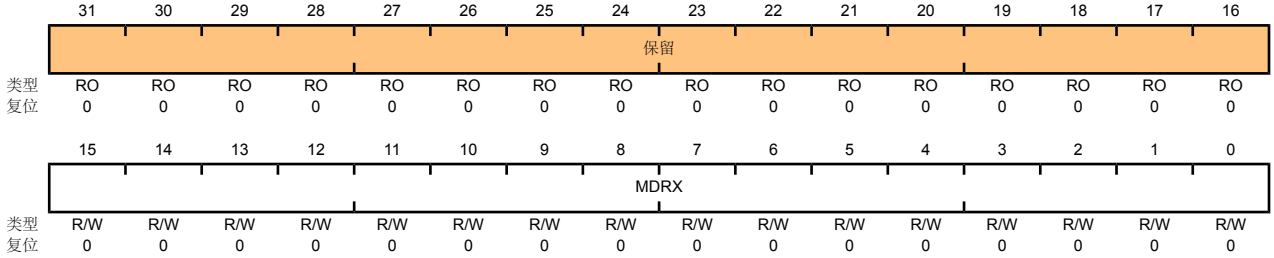
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	MDTX	R/W	0x0	MII寄存器发送数据 MDTX位代表将被写入下个MII管理传输的数据。

寄存器14: 以太网MAC管理接收数据（MACMRXD），偏移量 0x030

这个寄存器保存着从MII管理寄存器读出的最后一个值。

以太网MAC管理接收数据 (MACMRXD)

偏移量0x030
类型R/W, 复位0x0000.0000



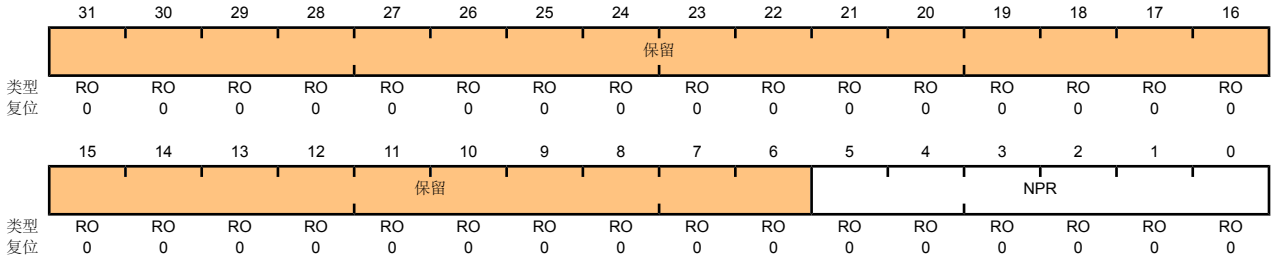
位/域	名称	类型	复位	描述
31:16	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
15:0	MDRX	R/W	0x0	MII寄存器接收数据 MDRX位代表在前面的MII管理传输中读出的数据。

寄存器15: 以太网MAC的包数目（MACNP），偏移量 0x034

这个寄存器保存着当前RX FIFO中帧的数量。 当NPR为‘全0’时，RX FIFO中没有帧，这时 RXINT位不置位。 当NPR是任何其它值时，RX FIFO中至少有一个帧，**MACRIS**寄存器的RXINT位被置位。

以太网MAC的包数目 (MACNP)

偏移量0x034
类型RO, 复位0x0000.0000



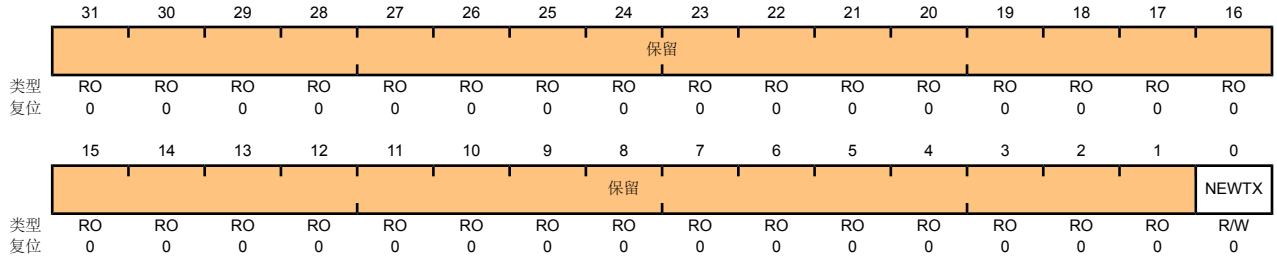
位/域	名称	类型	复位	描述
31:6	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
5:0	NPR	RO	0x0	接收FIFO中包的数量 NPR位代表保存在RX FIFO中的包的数量。 当NPR域大于0时，**MACRIS**寄存器的RXINT中断将有效。

寄存器16: 以太网MAC发送请求（MACTR），偏移量 0x038

这个寄存器使能软件来启动帧发送，将当前位于TX FIFO中的帧发送到物理媒体。 一旦帧已经从TX FIFO发送到媒体或遇到一个发送错误，NEWTX位就由硬件自动清零。

以太网MAC发送请求 (MACTR)

偏移量0x038
类型R/W, 复位0x0000.0000



位/域	名称	类型	复位	描述
31:1	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。
0	NEWTX	R/W	0x0	新的发送 该位置位时，一旦包被放置到TX FIFO中，NEWTX位就启动一次以太网发送。 一旦发送结束，该位就被清零。 如果正在使用早发送（见**MACTHR**寄存器），这个位就不需要被置位。

13.6 MII管理寄存器描述

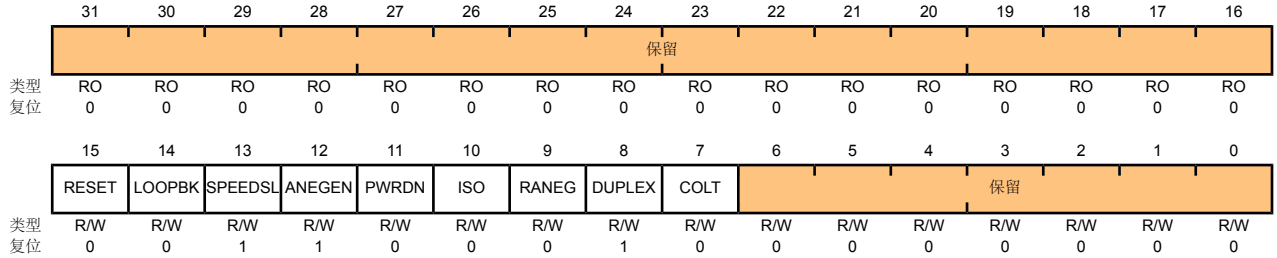
*IEEE 802.3标准*指定了一个寄存器集合，用来控制和集中PHY的状态。 这些寄存器被共同称为MII管理寄存器。 给出的所有地址都是绝对地址。 未列出的地址保留。

寄存器17: 以太网PHY管理寄存器0 – 控制（MR0），偏移量 0x00

这个寄存器使能软件来配置PHY的操作。这些寄存器的默认设置将PHY初始化成一个没有配置的正常的工作模式。

以太网PHY管理寄存器0 – 控制 (MR0)

偏移量0x00
类型R/W, 复位0x3100



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-写操作过程中应该保持不变。
15	RESET	R/W	0	复位寄存器 该位置位时将寄存器复位为默认状态并重新初始化内部状态机。一旦复位操作结束，该位就被硬件清零。
14	LOOPBK	R/W	0	环回模式 该位置位时使能环回模式的操作。接收电路和物理媒体分隔开，发送内容通过接收电路（代替物理媒体）被发送回来。
13	SPEEDSL	R/W	1	速度选择 1：使能100 Mb/s模式的操作（100BASE-TX）。 0：使能10 Mb/s模式的操作（10BASE-T）。
12	ANEGEN	R/W	1	自协商使能 该位置位时使能自协商处理。
11	PWRDN	R/W	0	掉电 该位置位时使PHY进入一个低功耗状态。
10	ISO	R/W	0	分隔 当该位置位时，发送数据通路和接收数据通路分隔开，忽略这些总线上的所有信号。
9	RANEG	R/W	0	重启自协商 该位置位时重启自协商处理。一旦开始重新启动，这个位就被硬件清零。
8	DUPLEX	R/W	1	设置双工模式 1：使能全双工模式的操作。该位可以由软件在手动配置过程中置位，也可以通过自协商处理置位。 0：使能半双工模式的操作。

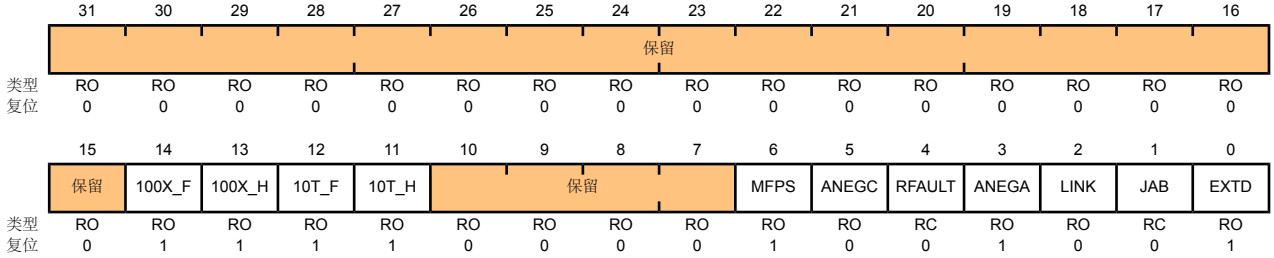
位/域	名称	类型	复位	描述
7	COLT	R/W	0	冲突检测 该位置位时使能冲突检测模式的操作。 COLT位在发送启动后有效；一旦发送终止，它就变为无效。
6:0	保留	R/W	0x00	写操作时作为0被写入，读操作时读出的内容被忽略。

寄存器18: 以太网PHY管理寄存器1 – 状态（MR1），偏移量 0x01

这个寄存器使能软件来确定PHY的功能和正确执行PHY的初始化和操作。

以太网PHY管理寄存器1 – 状态 (MR1)

偏移量0x01
类型RO, 复位0x7849



位/域	名称	类型	复位	描述
31:15	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。
14	100X_F	RO	1	100BASE-TX全双工模式 该位置位时表明PHY能够支持100BASE-TX全双工模式。
13	100X_H	RO	1	100BASE-TX半双工模式 该位置位时表明PHY能够支持100BASE-TX半双工模式。
12	10T_F	RO	1	10BASE-T全双工模式 该位置位时表明PHY能够支持10BASE-T全双工模式。
11	10T_H	RO	1	10BASE-T半双工模式 该位置位时表明PHY能够支持10BASE-T半双工模式。
10:7	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	MFPS	RO	1	前导码去除的管理帧 该位置位时表明管理接口能够接收前导码去除的管理帧。
5	ANEGC	RO	0	自协商结束 该位置位时表明自协商过程已经结束，自协商协议定义的扩展寄存器有效。
4	RFAULT	RC	0	远程故障 该位置位时表明检测到一个远程故障条件。 即使故障条件已经不存在了，这个位仍然保持置位，直至被读出。
3	ANEGA	RO	1	自协商 该位置位时表明PHY有能力执行自协商。
2	LINK	RO	0	连接建立 该位置位时表明PHY已经建立了一个有效连接。

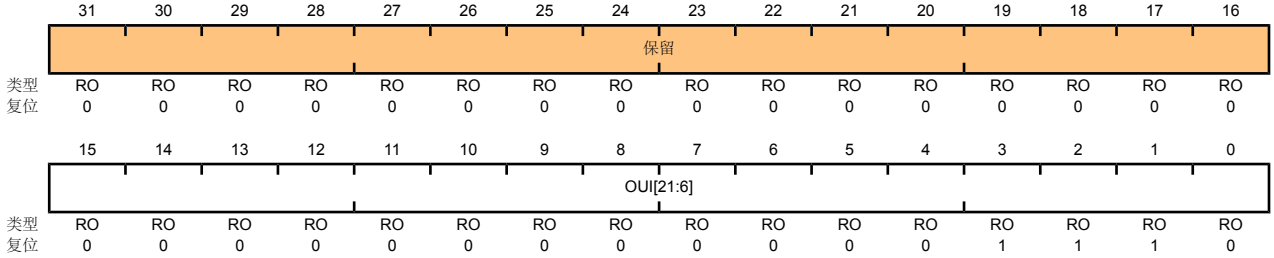
位/域	名称	类型	复位	描述
1	JAB	RC	0	Jabber条件 该位置位时表明PHY已经检测到一个Jabber条件。即使Jabber条件已经不存在了，这个位仍然保持置位，直至被读出。
0	EXTD	RO	1	扩展功能 该位置位时表明PHY提供一个扩展功能集，它可以通过扩展寄存器集来访问。

寄存器19: 以太网PHY管理寄存器2 – PHY标识符1（MR2），偏移量 0x02

这个寄存器连同**MR3**一起提供一个32位的值来指示制造商、模型和版本信息。

以太网PHY管理寄存器2 – PHY标识符1 (MR2)

偏移量0x02
类型RO, 复位0x000E



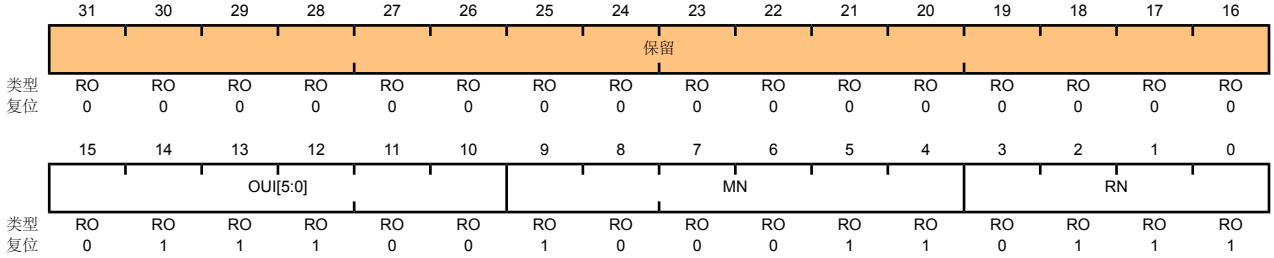
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
15:0	OUI[21:6]	RO	0x000E	组织唯一标识符[21:6] 这个域连同管理寄存器**3**的`OUI[5:0]`域一起组成指示PHY制造商的组织唯一标识符（OUI）。

寄存器20: 以太网PHY管理寄存器3 – PHY标识符2（MR3），偏移量 0x03

这个寄存器连同**MR2**一起提供一个32位的值来指示制造商、模型和版本信息。

以太网PHY管理寄存器3 – PHY标识符2 (MR3)

偏移量0x03
类型RO, 复位0x7237



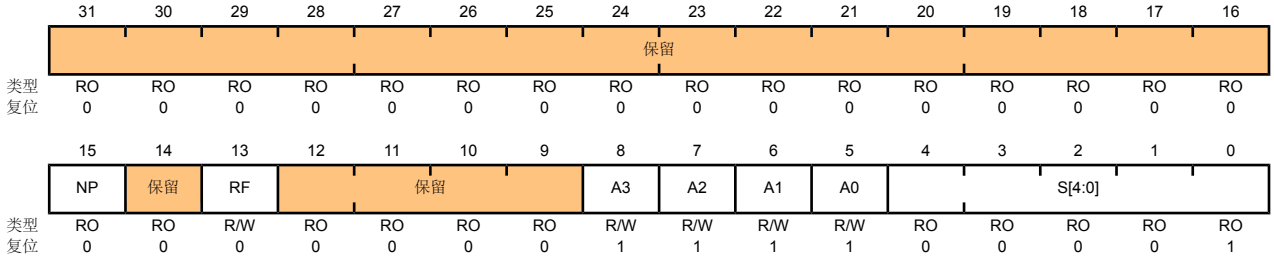
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
15:10	OUI[5:0]	RO	0x1C	组织唯一标识符[5:0] 这个域连同管理寄存器**2**的`OUI[21:6]`域一起组成指示PHY制造商的组织唯一标识符。
9:4	MN	RO	0x23	模型编号 `MN`域代表PHY的模型编号。
3:0	RN	RO	0x7	版本号 `RN`域代表PHY的版本号。

寄存器21: 以太网PHY管理寄存器4 – 自协商通告（MR4），偏移量 0x04

这个寄存器提供了自协商过程中使用的PHY通告的能力。 位8:5代表技术能力域位A[7:0]。 这个字段可以由软件覆盖来自协商为一个备用的公共技术。 写这个寄存器不会产生任何影响，直到自协商被重新启动。

以太网PHY管理寄存器4 – 自协商通告 (MR4)

偏移量0x04
类型R/W, 复位0x01E1



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-写操作过程中应该保持不变。
15	NP	RO	0	下页 该位置位时表明PHY能够下页交换来提供更详细的有关PHY功能的信息。
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	RF	R/W	0	远程故障 该位置位时向连接方指示碰到了一个远程故障条件。
12:9	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－写操作过程中应当保持不变。
8	A3	R/W	1	技术能力域[3] 该位置位时表明PHY支持100Base-TX全双工信号协议。 如果软件想确保不使用这个模式，该位可以写入0，自协商用**MR0**寄存器的RANEG位重新启动。
7	A2	R/W	1	技术能力域[2] 该位置位时表明PHY支持100Base-T半双工信号协议。 如果软件想确保不使用这个模式，该位可以写入0，重新启动自协商。
6	A1	R/W	1	技术能力域[1] 该位置位时表明PHY支持10Base-T全双工信号协议。 如果软件想确保不使用这个模式，该位可以写入0，重新启动自协商。
5	A0	R/W	1	技术能力域[0] 该位置位时表明PHY支持10Base-T半双工信号协议。 如果软件想确保不使用这个模式，该位可以写入0，重新启动自协商。

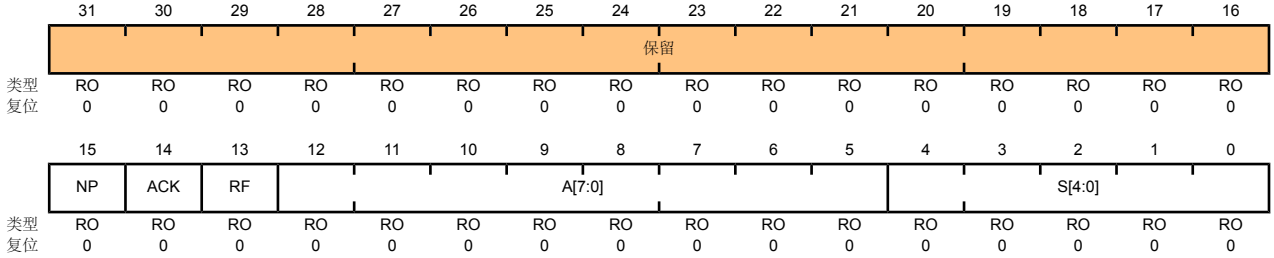
位/域	名称	类型	复位	描述
4:0	S[4:0]	RO	0x01	选择器域 S[4:0]域为PHY之间的通信对32条可能的消息进行编码。 这个域硬编码为0x01，表明Stellaris® PHY遵循*IEEE 802.3*。

寄存器22: 以太网PHY管理寄存器5 – 自协商连接方基页能力（MR5），偏移量0x05

这个寄存器提供在自协商过程中接收和保存的连接方PHY通告的能力。

以太网PHY管理寄存器5 – 自协商连接方基页能力 (MR5)

偏移量0x05
类型RO, 复位0x0000



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-写操作过程中应该保持不变。
15	NP	RO	0	下页 该位置位时表明连接方的PHY能够下页交换来提供更详细的有关PHY功能的信息。
14	ACK	RO	0	应答 该位置位时表明器件已经成功接收到连接方在自协商过程中通告的能力。
13	RF	RO	0	远程故障 用作一个标准传输机制，用来传输简单的故障信息。
12:5	A[7:0]	RO	0x00	技术能力域 A域编码PHY支持的单个技术。 见**MR4**寄存器。
4:0	S[4:0]	RO	0x00	选择器域 S域为PHY之间的通信对可能的消息进行编码。

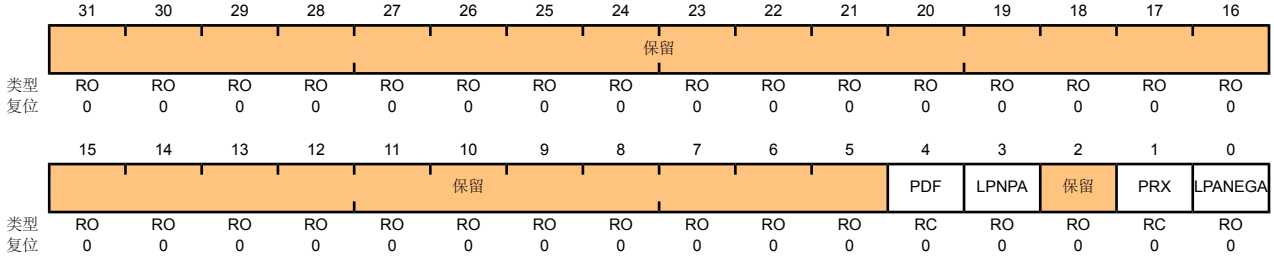
值	含义
0x00	保留
0x01	IEEE Std 802.3
0x02	IEEE Std 802.9 ISLAN-16T
0x03	IEEE Std 802.5
0x04	IEEE Std 1394
0x05 – 0x1F	保留

寄存器23: 以太网PHY管理寄存器6 – 自协商扩展（MR6），偏移量 0x06

该寄存器使能软件来确定自协商后PHY和连接方的自协商和下页功能。

以太网PHY管理寄存器6 – 自协商扩展 (MR6)

偏移量0x06
类型RO, 复位0x0000



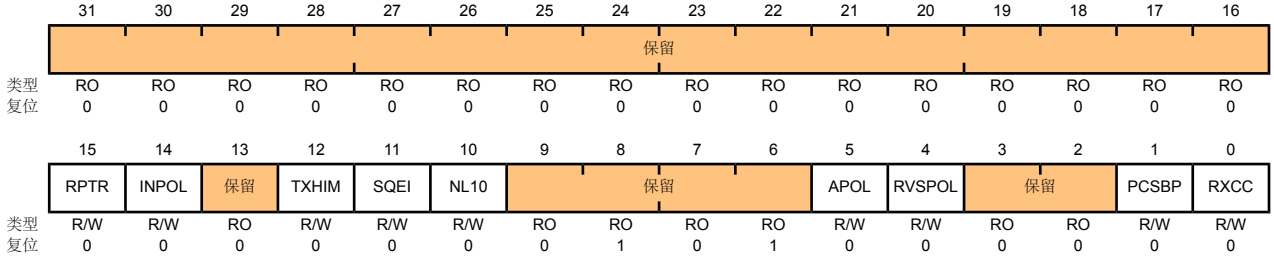
位/域	名称	类型	复位	描述
31:5	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
4	PDF	RC	0	并行检测故障 该位置位时表明在连接时检测到多于一种技术。 该位在读出时被清零。
3	LPNPA	RO	0	连接方具有下页功能。 该位置位时表明连接方具有下页功能。
2	保留	RO	0x000	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	PRX	RC	0	接收到新的页 该位置位时表明从连接方接收到一个新的页并保存到合适的单元中。 该位在寄存器被读出之前一直保持置位。
0	LPANEGA	RO	0	连接方具有自协商功能。 该位置位时表明连接方具有自协商功能。

寄存器24: 以太网PHY管理寄存器16 – 厂商特定（MR16），偏移量 0x10

该寄存器使能软件来配置PHY的厂商特定模式的操作。

以太网PHY管理寄存器16 – 厂商特定 (MR16)

偏移量0x10
类型R/W, 复位0x0140



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-写操作过程中应该保持不变。
15	RPTR	R/W	0	转发器模式 该位置位时使能转发器模式的操作。 在这个模式中，不允许全双工通信，载波检测信号只响应接收活动。 如果PHY被配置成10Base-T模式，SQE测试功能就被禁止。
14	INPOL	R/W	0	中断极性 1：设置PHY中断的极性为高电平有效。 0：设置PHY中断的极性为低电平有效。 重要: 由于媒体访问控制器希望PHY的中断低电平有效，因此为了确保操作正确该位必须总是被写入0。
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－写操作过程中应当保持不变。
12	TXHIM	R/W	0	发送高阻模式 该位置位时使能发送器高阻模式。 在这种模式下，TXOP和TXON发送器管脚进入高阻状态。 RXIP和RXIN管脚保持全部功能。
11	SQEI	R/W	0	SQE禁止测试 该位置位时禁止10Base-T SQE测试。 当该位为0时，通过在一个帧发送结束后产生一个冲突脉冲来执行SQE测试。
10	NL10	R/W	0	自然环回模式 该位置位时使能10Base-T自然环回模式。 当10Base-T模式使能时，这会使PHY接收到的发送数据返回到接收数据通路。
9:6	保留	RO	0x05	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

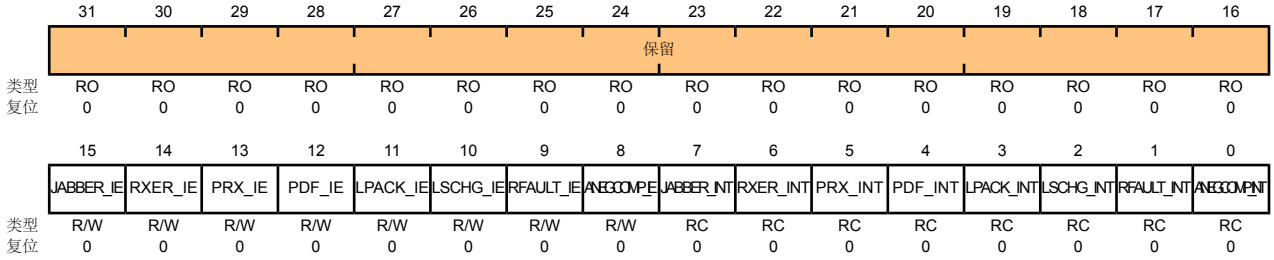
位/域	名称	类型	复位	描述
5	APOL	R/W	0	自动极性禁止 该位置位时禁止PHY的自动极性功能。 如果该位为0，当PHY处于10Base-T模式时，PHY会因为自协商过程中的一个错误极性连接而自动翻转接收到的信号。
4	RVSPOL	R/W	0	接收数据极性 这一位指示接收数据脉冲是否正在被翻转。 如果APOL位为0，则RVSPOL位只可读，指示自动极性电路是否正在翻转极性。 此时，RVSPOL位为1时表明接收数据被翻转；为0时表明接收数据未被翻转。 如果APOL位被置位，则RVSPOL位可写，软件可以强制接收数据被翻转。 将RVSPOL设置为1强制接收数据翻转；设置为0不翻转接收数据。
3:2	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	PCSBP	R/W	0	PCS旁路 该位置位时使能PCS的旁路和100Base-TX模式中的扰码/解扰功能。 这个模式只有在自协商禁能而100Base-T模式使能的情况下才有效。
0	RXCC	R/W	0	接收时钟控制 该位置位时，如果PHY配置为100Base-TX模式，使能接收时钟控制的节电模式。如果当前没有接收到来自物理媒体的任何数据，该节电模式就关断接收时钟来节省功耗。当PCSBP被使能时，该模式不应该使用；当 **MR0**寄存器的LOOPBK 位被置位时，该模式被自动禁止。

寄存器25: 以太网PHY管理寄存器17 – 中断控制/状态（MR17），偏移量 0x11

这个寄存器为控制和观察触发**MACRIS**寄存器中PHY中断的事件提供一些方法。 这个寄存器也可以通过MII串行接口被用在轮询模式中，作为一个通过一个寄存器地址来观察PHY中关键事件的方法。 位0～7是状态位，每一位根据一个事件被置位。 这些位在寄存器读出后被清零。 当寄存器的位8～15被置位时，使能低字节中相应的位来通知**MACRIS**寄存器中一个PHY中断的出现。

以太网PHY管理寄存器17 – 中断控制/状态 (MR17)

偏移量0x11
类型R/W, 复位0x0000



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-写操作过程中应该保持不变。
15	JABBER_IE	R/W	0	Jabber中断使能 该位置位时，在PHY检测到一个Jabber条件时使能系统中断。
14	RXER_IE	R/W	0	接收错误中断使能 该位置位时，在PHY检测到一个接收错误时使能系统中断。
13	PRX_IE	R/W	0	页接收中断使能 该位置位时，在PHY接收到一个新的页时使能系统中断。
12	PDF_IE	R/W	0	并行检测故障中断使能 该位置位时，在PHY检测到一个并行检测故障时使能系统中断。
11	LPACK_IE	R/W	0	LP应答中断使能 该位置位时，当在自协商过程中接收到带应答位的FLP突发时使能系统中断。
10	LSCHG_IE	R/W	0	连接状态改变中断使能 该位置位时，当连接状态从OK变为FAIL时使能系统中断。
9	RFAULT_IE	R/W	0	远程故障中断使能 该位置位时，当连接方通知一个远程故障条件时使能系统中断。
8	ANEGCOMP_IE	R/W	0	自协商结束中断使能 该位置位时，当自协商序列成功结束时使能系统中断。
7	JABBER_INT	RC	0	Jabber事件中断 该位置位时表明10Base-T电路已经检测到一个Jabber事件。

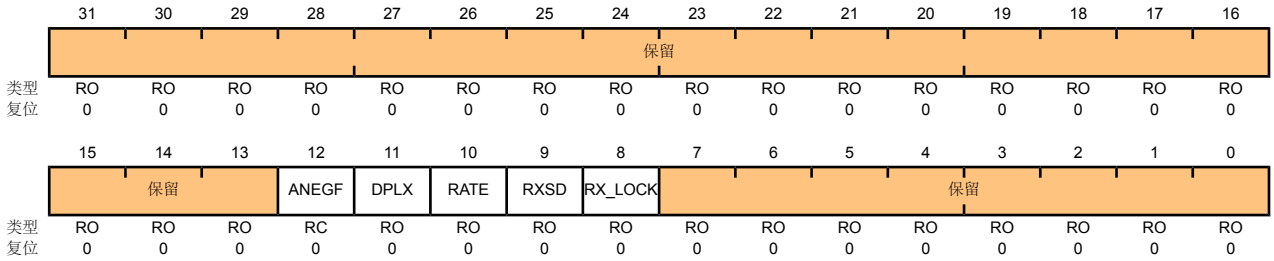
位/域	名称	类型	复位	描述
6	RXER_INT	RC	0	接收错误中断 该位置位时表明PHY已经检测到一个接收错误。
5	PRX_INT	RC	0	页接收中断 该位置位时表明在自协商过程中已经从连接方接收到一个新的页。
4	PDF_INT	RC	0	并行检测故障中断 该位置位时表明在自协商过程中PHY已经检测到一个并行检测故障。
3	LPACK_INT	RC	0	LP应答中断 该位置位时表明在自协商过程中已经接收到一个带置位应答位的FLP突发。
2	LSCHG_INT	RC	0	连接状态改变中断 该位置位时表明连接状态已经从OK变成了FAIL。
1	RFAULT_INT	RC	0	远程故障中断 该位置位时表明连接方已经发信号通知一个远程故障条件。
0	ANEGCOMP_INT	RC	0	自协商结束中断 该位置位时表明自协商序列已经成功结束。

寄存器26: 以太网PHY管理寄存器18 – 诊断（MR18），偏移量 0x12

这个寄存器使能软件来诊断前面自协商的结果。

以太网PHY管理寄存器18 – 诊断 (MR18)

偏移量0x12
类型RO, 复位0x0000



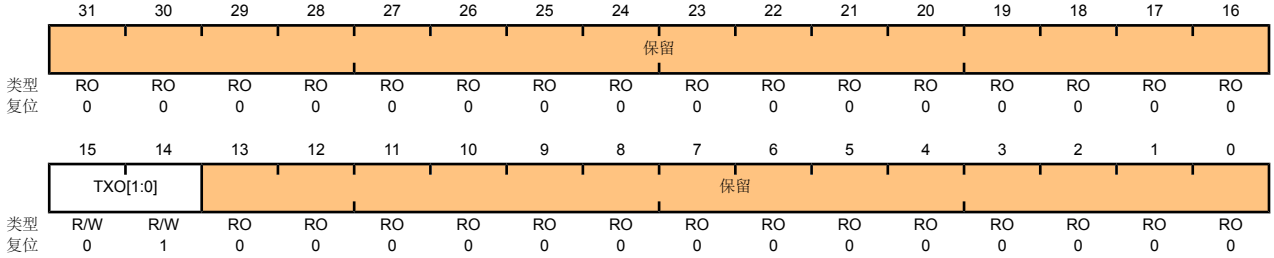
位/域	名称	类型	复位	描述
31:13	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
12	ANEGF	RC	0	自协商失败 该位置位时表明在自协商过程中没有找到公共的技术，自协商失败。 该位在读出前一直保持置位。
11	DPLX	RO	0	双工模式 该位置位时表明全双工是自协商过程中找到的最大共同特性（highest common denominator）。 否则，半双工是找到的最大共同特性。
10	RATE	RO	0	速度 该位置位时表明100Base-TX是自协商过程中找到的最大共同特性。 否则，10Base-TX是找到的最大共同特性。
9	RXSD	RO	0	接收检测 该位置位时表明接收信号检测已经出现（100Base-TX模式）或者已经检测到Manchester编码的数据（10Base-T模式）。
8	RX_LOCK	RO	0	接收PLL锁定 该位置位时表明接收PLL已经锁定到接收信号上，以便在所选的速度下工作 （10Base-T或 100Base-TX）。
7:0	保留	RO	00	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器27: 以太网PHY管理寄存器19 – 收发器控制（MR19），偏移量 0x13

该寄存器使能软件设置发送输出的增益来补偿变压器损耗。

以太网PHY管理寄存器19 – 收发器控制 (MR19)

偏移量0x13
类型R/W, 复位0x4000



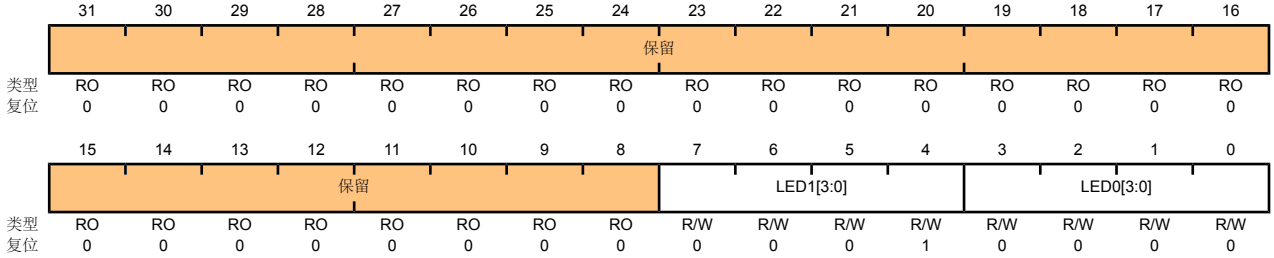
位/域	名称	类型	复位	描述
31:16	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
15:14	TXO[1:0]	R/W	1	发送幅度选择 TX0域设置发送输出幅度来说明发送变压器的插入损耗。 值 含义 00 插入损耗为0.0dB时设置的增益 01 插入损耗为0.4dB时设置的增益 10 插入损耗为0.8dB时设置的增益 11 插入损耗为1.2dB时设置的增益
13:0	保留	RO	0x0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器28: 以太网PHY管理寄存器23 – LED配置（MR23），偏移量0x17

该寄存器使能软件来选择LED的触发源。

以太网PHY管理寄存器23 – LED配置 (MR23)

偏移量0x17
类型R/W, 复位0x0010



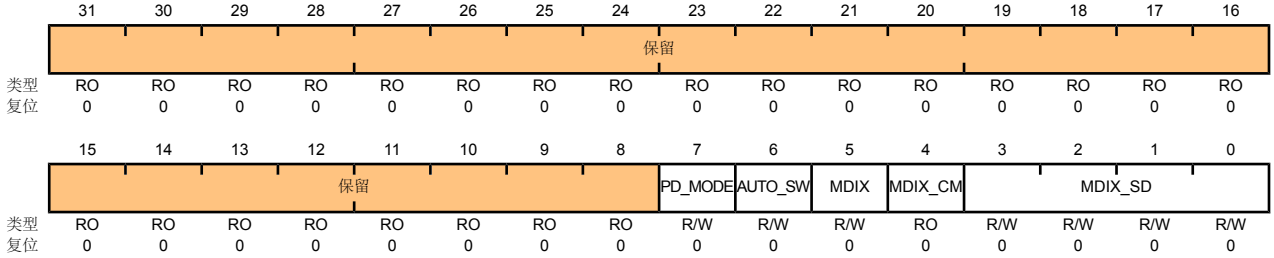
位/域	名称	类型	复位	描述
31:8	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读-修改-写操作过程中应该保持不变。
7:4	LED1[3:0]	R/W	1	LED1源 LED1域选择LED1信号的触发源。 值 含义 0000 连接良好 0001 RX或TX活动（默认的LED1状态） 0010 TX活动 0011 RX活动 0100 冲突 0101 100BASE-TX模式 0110 10BASE-T模式 0111 全双工 1000 连接良好& 闪烁＝RX或TX活动
3:0	LED0[3:0]	R/W	0	LED0源 LED0域选择LED0信号的触发源。 值 含义 0000 连接良好（默认的LED0状态） 0001 RX或TX活动 0010 TX活动 0011 RX活动 0100 冲突 0101 100BASE-TX模式 0110 10BASE-T模式 0111 全双工 1000 连接良好& 闪烁＝RX或TX活动

寄存器29: 以太网PHY管理寄存器24 –MDI/MDIX控制（MR24），偏移量 0x18

这个寄存器使能软件来控制MDI/MDIX多路复用器的行为和它的切换功能。

以太网PHY管理寄存器24 –MDI/MDIX控制 (MR24)

偏移量0x18
类型R/W, 复位0x00C0



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	PD_MODE	R/W	0	并行检测模式 该位置位时使能并行检测模式，并在自协商未使能时允许自动切换工作。
6	AUTO_SW	R/W	0	自动切换使能 该位置位时使能MDI/MDIX多路复用器的自动切换。
5	MDIX	R/W	0	自动切换配置 该位置位时表明MDI/MDIX多路复用器处于交叉（MDIX）配置。 当该位为0时，表明多路复用器处于直通（MDI）配置。 当AUTO_SW位为1时，MDIX位只可读。 当AUTO_SW位为0时，MDIX位可读/写，并且可以手动配置。
4	MDIX_CM	RO	0	自动切换完成 该位置位时表明自动切换序列已经结束。 如果该位为0，则表明序列还未完成或自动切换被禁能。
3:0	MDIX_SD	R/W	0	自动切换种子（Auto-Switch Seed） 这个域为切换算法提供最初的种子（seed）。这个种子直接影响分别写入位[3:0]的尝试次数[5,4]。 值0将种子设为0x5。

14 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。

LM3S6100 控制器提供一个模拟比较器，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。

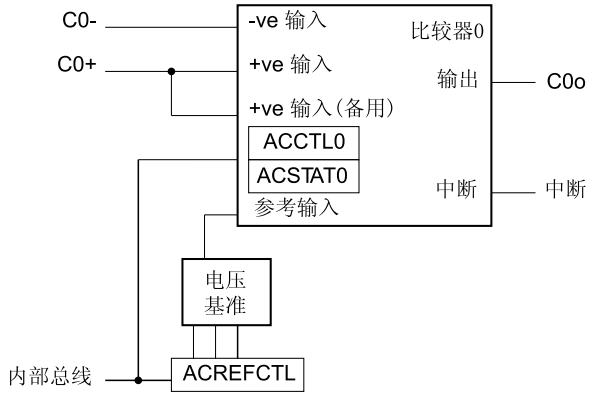
比较器可将测试电压与下面的其中一种电压相比较：

- 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。

14.1 结构图

图 **14-1.** 模拟比较器模块的结构图



14.2 功能描述

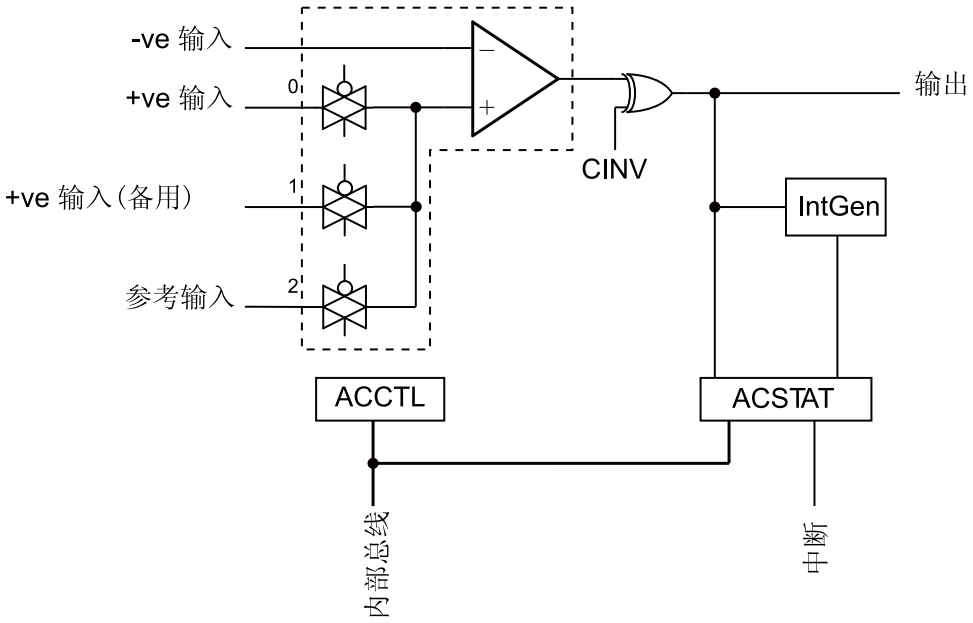
重要: 建议将模拟输入管脚的数字-输入使能 (GPIO模块的GPIODEN 位) ，以防止从I/O端口中吸收过量的电流。

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 14-2 在 323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 **14-2.** 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 **14-1.** 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

14.2.1 内部参考编程

内部的参考结构如图 14-3 在 324页所示。该结构通过一个配置寄存器(**ACREFCTL**)来控制。表 14-2 在 324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 **14-3.** 比较器内部参考结构

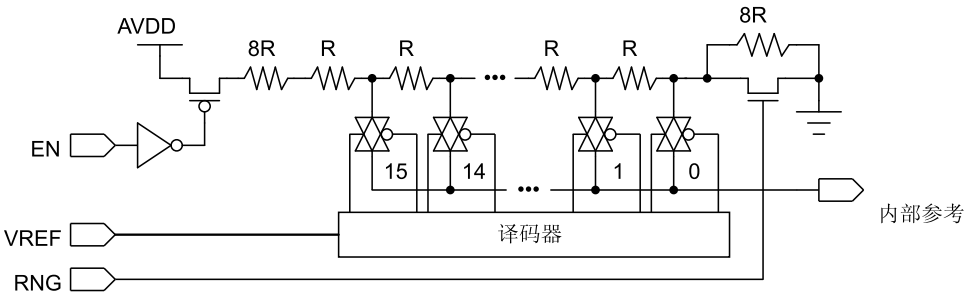


表 **14-2.** 内部参考电压和**ACREFCTL**位域的值

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。
EN=1	RNG=0	阶梯电阻的总阻值为32R。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 \ VREF$ 在该模式中内部参考电压的范围是0.825－2.37V。
	RNG=1	阶梯电阻的总阻值为24R。 $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \times V_{REF}$ 在该模式中内部参考电压的范围是0.0－2.0625V。

14.3 初始化和配置

下面的例子展示了应如何配置模拟比较器才能从内部寄存器中读回其输出值。

1. 向系统控制模块中的**RCGC1**寄存器写入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。

14.4 寄存器映射

表 14-3 在 325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 **14-3.** 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	326
0x04	ACRIS	RO	0x0000.0000	模拟比较器原始中断状态	327
0x08	ACINTEN	R/W	0x0000.0000	模拟比较器中断使能	328
0x10	ACREFCTL	R/W	0x0000.0000	模拟比较器参考电压控制	329
0x20	ACSTAT0	RO	0x0000.0000	模拟比较器状态0	330
0x24	ACCTL0	R/W	0x0000.0000	模拟比较器控制0	331

14.5 寄存器描述

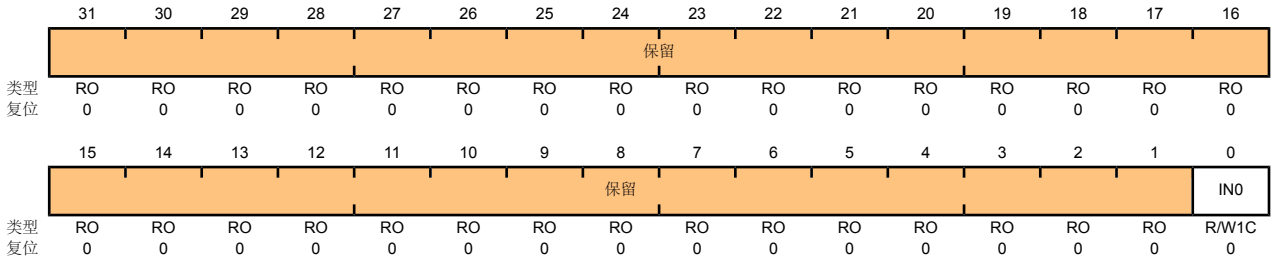
本节内容将按地址偏移量的数字顺序来列举并且描述模拟比较器寄存器。

寄存器1: 模拟比较器屏蔽后的中断状态（ACMIS），偏移量0x00

该寄存器汇总了比较器的（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



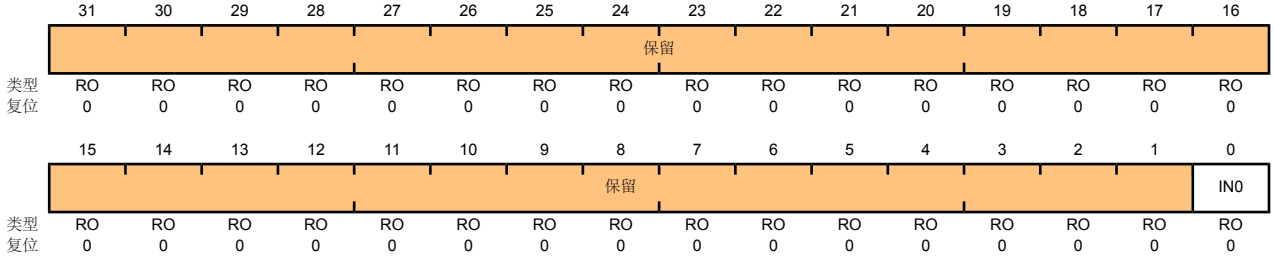
位/域	名称	类型	复位	描述
31:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IN0	R/W1C	0	比较器0屏蔽后的中断状态 给出该中断屏蔽后的中断状态。向该位写1可以清零挂起中断。

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



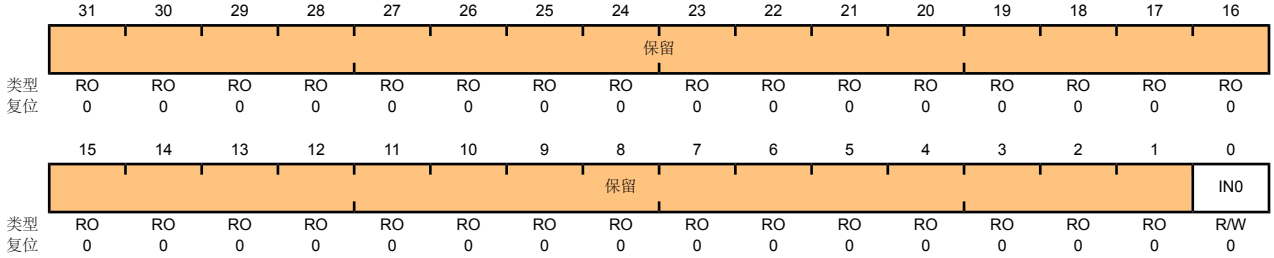
位/域	名称	类型	复位	描述
31:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IN0	RO	0	比较器0中断状态 当该位置位时，表示中断已通过比较器0产生。

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



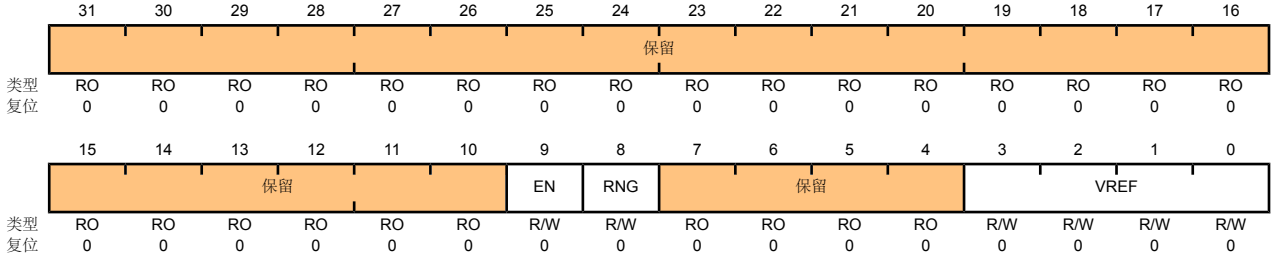
位/域	名称	类型	复位	描述
31:1	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
0	IN0	R/W	0	比较器0中断使能位 当该位置位时，使能比较器0输出的控制器中断。

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



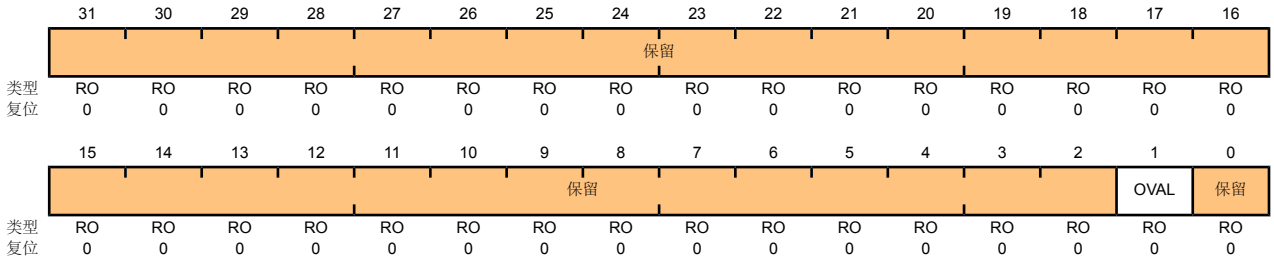
位/域	名称	类型	复位	描述
31:10	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
9	EN	R/W	0	阶梯电阻使能位 EN 位指示阶梯电阻(resistor ladder)是否已上电。如果该位为0，则阶梯电阻未上电。如果该位为1，则阶梯电阻被连接到模拟V_{DD}。 该位复位为0使得在未使用和未编程的情况下内部参考消耗的功率总量最小。
8	RNG	R/W	0	阶梯电阻范围 RNG 位指示阶梯电阻的范围。如果该位为0，则阶梯电阻的总电阻为32R。如果该位为1，则阶梯电阻的总电阻为24R。
7:4	保留	RO	0	软件不应该依赖保留位的值。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。
3:0	VREF	R/W	0	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 14-2 在 324页。

寄存器5: 模拟比较器状态0（ACSTAT0），偏移量 0x20

该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



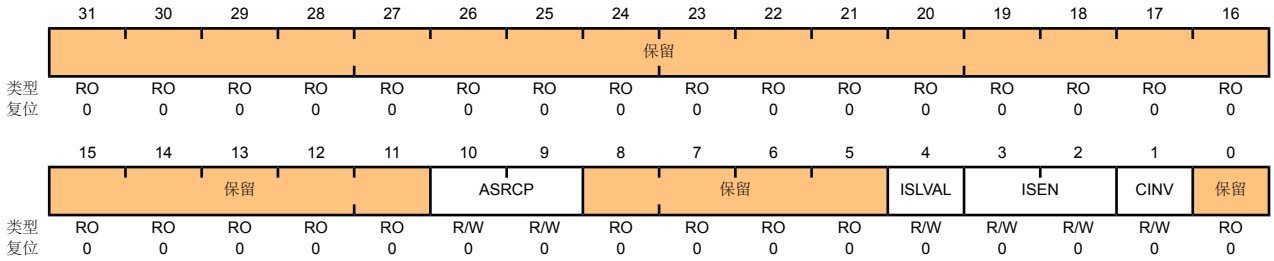
位/域	名称	类型	复位	描述
31:2	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
1	OVAL	RO	0	比较器输出值 0VAL 位指示比较器的当前输出值。
0	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。

寄存器6: 模拟比较器控制0（ACCTL0），偏移量 0x24

该寄存器用来配置 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



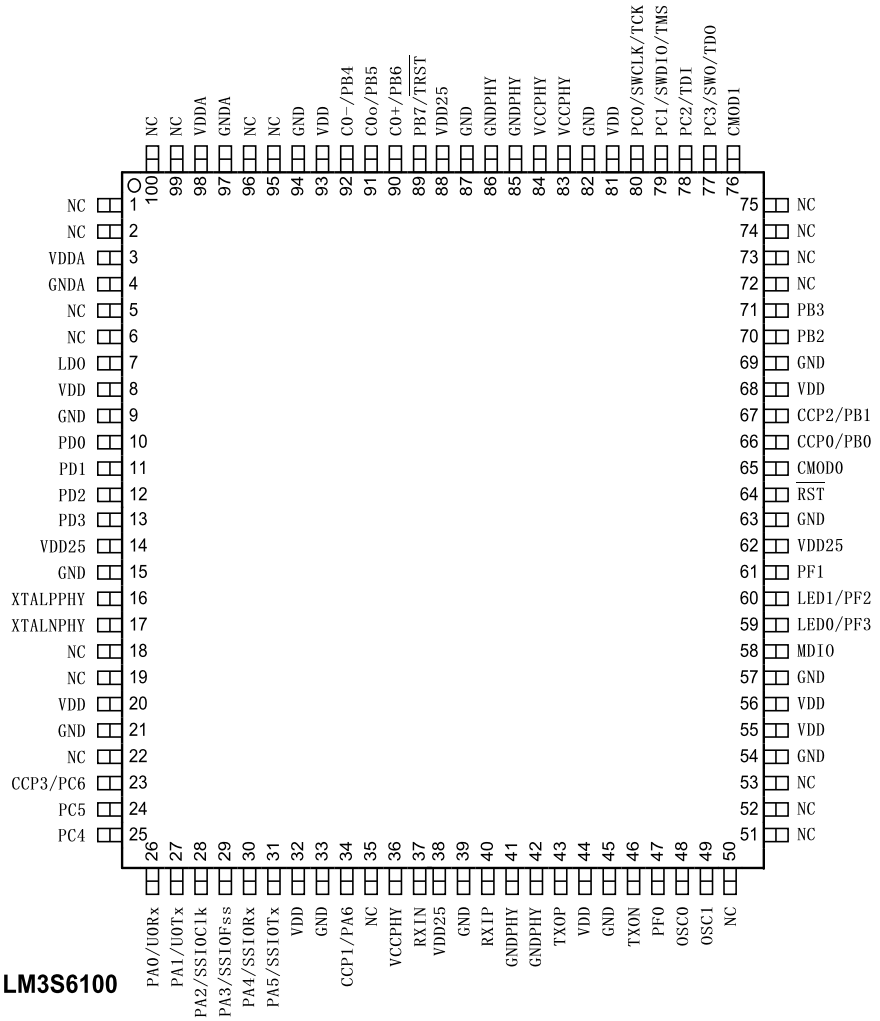
位/域	名称	类型	复位	描述
31:11	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
10:9	ASRCP	R/W	0	模拟正电压源 ASRCP 位域指定了到比较器VIN+端口的输入电压源。该位域的编码如下： ASRCP 功能 00 管脚值 01 C0+的管脚值 10 内部电压参考 11 保留
8:5	保留	RO	0	软件不应该依赖保留位的值。为了兼容未来的器件，保留位的值在读－修改－写操作过程中应当保持不变。
4	ISLVAL	R/W	0	电平触发中断值 ISLVAL 位指定了在电平检测模式中能够使中断产生的输入电平的感应值。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	0	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： ISEN 功能 00 电平检测，见 ISLVAL 01 下降沿 10 上升沿 11 上升/下降沿
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。

位/域	名称	类型	复位	描述
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。

15 管脚图

图 15-1 在 333页显示管脚图和管脚到信号名称的映射。

图 **15-1.** 管脚连接图



16 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 16-1 在 334页显示管脚到信号名称的映射，包括信号的功能特性。表 16-2 在 337页按信号名称的字母顺序列出信号。

表 16-3 在 341页按功能将信号分组，GPIO管脚除外。表 16-4 在 343页列出了GPIO管脚和它们可选的功能。

表 **16-1.** 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	NC	-	-	不连接
2	NC	-	-	不连接
3	VDDA	-	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。
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	NC	-	-	不连接
6	NC	-	-	不连接
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
11	PD1	I/O	TTL	GPIO端口D位1
12	PD2	I/O	TTL	GPIO端口D位2
13	PD3	I/O	TTL	GPIO端口D位3
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。
16	XTALPPHY	O	TTL	以太网PHY的XTALP。
17	XTALNPHY	I	TTL	以太网PHY的XTALN。
18	NC	-	-	不连接
19	NC	-	-	不连接
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	NC	-	-	不连接
23	CCP3	I/O	TTL	捕获/比较/PWM 3
	PC6	I/O	TTL	GPIO端口C位6

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
24	PC5	I/O	TTL	GPIO端口C位5
25	PC4	I/O	TTL	GPIO端口C位4
26	PA0	I/O	TTL	GPIO端口A位0
	U0Rx	I	TTL	UART模块0接收。当在IrDA模式下时，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模式下时，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和某些逻辑的电源正极。
33	GND	-	电源	逻辑和I/O管脚的地参考。
34	CCP1	I/O	TTL	捕获/比较/PWM 1
	PA6	I/O	TTL	GPIO端口A位6
35	NC	-	-	不连接
36	VCCPHY	I	TTL	以太网PHY的VCC。
37	RXIN	I	模拟	以太网PHY的RXIN。
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。
39	GND	-	电源	逻辑和I/O管脚的地参考。
40	RXIP	I	模拟	以太网PHY的RXIP。
41	GNDPHY	I	TTL	以太网PHY的GND。
42	GNDPHY	I	TTL	以太网PHY的GND。
43	TXOP	O	模拟	以太网PHY的TXOP。
44	VDD	-	电源	I/O和某些逻辑的电源正极。
45	GND	-	电源	逻辑和I/O管脚的地参考。
46	TXON	O	模拟	以太网PHY的TXON。
47	PF0	I/O	TTL	GPIO端口F位0
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	NC	-	-	不连接
51	NC	-	-	不连接
52	NC	-	-	不连接
53	NC	-	-	不连接
54	GND	-	电源	逻辑和I/O管脚的地参考。
55	VDD	-	Power	I/O和某些逻辑的电源正极。
56	VDD	-	电源	I/O和某些逻辑的电源正极。

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
57	GND	-	电源	逻辑和I/O管脚的地参考。
58	MDIO	I/O	TTL	以太网PHY的MDIO。
59	LED0	O	TTL	MII LED 0
	PF3	I/O	TTL	GPIO端口F位3
60	LED1	O	TTL	MII LED 1
	PF2	I/O	TTL	GPIO端口F位2
61	PF1	I/O	TTL	GPIO端口F位1
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	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	CCP0	I/O	TTL	捕获/比较/PWM 0
	PB0	I/O	TTL	GPIO端口B位0
67	CCP2	I/O	TTL	捕获/比较/PWM 2
	PB1	I/O	TTL	GPIO端口B位1
68	VDD	-	电源	I/O和某些逻辑的电源正极。
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
71	PB3	I/O	TTL	GPIO端口B位3
72	NC	-	-	不连接
73	NC	-	-	不连接
74	NC	-	-	不连接
75	NC	-	-	不连接
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
	SWO	O	TTL	JTAG TDO和SWO
	TDO	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
	SWDIO	I/O	TTL	JTAG TMS和SWDIO
	TMS	I/O	TTL	JTAG TMS和SWDIO
80	PC0	I/O	TTL	GPIO端口C位0
	SWCLK	I	TTL	JTAG/SWD CLK
	TCK	I	TTL	JTAG/SWD CLK
81	VDD	-	电源	I/O和某些逻辑的电源正极。
82	GND	-	电源	逻辑和I/O管脚的地参考。
83	VCCPHY	I	TTL	以太网PHY的VCC。
84	VCCPHY	I	TTL	以太网PHY的VCC。
85	GNDPHY	I	TTL	以太网PHY的GND。

管脚编号	管脚名称	管脚类型	缓冲区类型	描述
86	GNDPHY	I	TTL	以太网PHY的GND。
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	C0+	I	模拟	模拟比较器0正极输入
	PB6	I/O	TTL	GPIO端口B位6
91	C0o	O	TTL	模拟比较器0输出
	PB5	I/O	TTL	GPIO端口B位5
92	C0-	I	模拟	模拟比较器0负极输入
	PB4	I/O	TTL	GPIO端口B位4
93	VDD	-	电源	I/O和某些逻辑的电源正极。
94	GND	-	电源	逻辑和I/O管脚的地参考。
95	NC	-	-	不连接
96	NC	-	-	不连接
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	NC	-	-	不连接
100	NC	-	-	不连接

表 **16-2.** 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	91	O	TTL	模拟比较器0输出
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	23	I/O	TTL	捕获/比较/PWM 3
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。
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。
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。

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
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。
GNDPHY	41	I	TTL	以太网PHY的GND。
GNDPHY	42	I	TTL	以太网PHY的GND。
GNDPHY	85	I	TTL	以太网PHY的GND。
GNDPHY	86	I	TTL	以太网PHY的GND。
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。
LED0	59	O	TTL	MII LED 0
LED1	60	O	TTL	MII LED 1
MDIO	58	I/O	TTL	以太网PHY的MDIO。
NC	1	-	-	不连接
NC	2	-	-	不连接
NC	5	-	-	不连接
NC	6	-	-	不连接
NC	18	-	-	不连接
NC	19	-	-	不连接
NC	22	-	-	不连接
NC	35	-	-	不连接
NC	50	-	-	不连接
NC	51	-	-	不连接
NC	52	-	-	不连接
NC	53	-	-	不连接
NC	72	-	-	不连接
NC	73	-	-	不连接
NC	74	-	-	不连接
NC	75	-	-	不连接
NC	95	-	-	不连接
NC	96	-	-	不连接
NC	99	-	-	不连接
NC	100	-	-	不连接

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
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
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
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
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
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
RST	64	I	TTL	系统复位输入。
RXIN	37	I	模拟	以太网PHY的RXIN。
RXIP	40	I	模拟	以太网PHY的RXIP
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
SWCLK	80	I	TTL	JTAG/SWD CLK
SWDIO	79	I/O	TTL	JTAG TMS和SWDIO
SWO	77	O	TTL	JTAG TDO和SWO

管脚名称	管脚编号	管脚类型	缓冲区类型	描述
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
TRST	89	I	TTL	JTAG TRSTn
TXON	46	O	模拟	以太网PHY的TXON。
TXOP	43	O	模拟	以太网PHY的TXOP。
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。
VCCPHY	36	I	TTL	以太网PHY的VCC。
VCCPHY	83	I	TTL	以太网PHY的VCC。
VCCPHY	84	I	TTL	以太网PHY的VCC。
VDD	8	-	电源	I/O和某些逻辑的电源正极。
VDD	20	-	电源	I/O和某些逻辑的电源正极。
VDD	32	-	电源	I/O和某些逻辑的电源正极。
VDD	44	-	电源	I/O和某些逻辑的电源正极。
VDD	55	-	Power	I/O和某些逻辑的电源正极。
VDD	56	-	电源	I/O和某些逻辑的电源正极。
VDD	68	-	电源	I/O和某些逻辑的电源正极。
VDD	81	-	电源	I/O和某些逻辑的电源正极。
VDD	93	-	电源	I/O和某些逻辑的电源正极。
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。
XTALNPHY	17	I	TTL	以太网PHY的XTALN。
XTALPPHY	16	O	TTL	以太网PHY的XTALP。

表 **16-3.** 按功能分组的信号，**GPIO**管脚除外

功能	管脚名称	管脚编号	管脚类型	缓冲区类型	描述
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
Power	GND	9	-	电源	逻辑和I/O管脚的地参考。
	VDD	55	-	Power	I/O和某些逻辑的电源正极。
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
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。
以太网PHY	GNDPHY	41	I	TTL	以太网PHY的GND。
	GNDPHY	42	I	TTL	以太网PHY的GND。
	GNDPHY	85	I	TTL	以太网PHY的GND。
	GNDPHY	86	I	TTL	以太网PHY的GND。
	LED0	59	O	TTL	MII LED 0
	LED1	60	O	TTL	MII LED 1
	MDIO	58	I/O	TTL	以太网PHY的MDIO。
	RXIN	37	I	模拟	以太网PHY的RXIN。
	RXIP	40	I	模拟	以太网PHY的RXIP。
	TXON	46	O	模拟	以太网PHY的TXON
	TXOP	43	O	模拟	以太网PHY的TXOP
	VCCPHY	36	I	TTL	以太网PHY的VCC
	VCCPHY	83	I	TTL	以太网PHY的VCC
	VCCPHY	84	I	TTL	以太网PHY的VCC
	XTALNPHY	17	I	TTL	以太网PHY的XTALN
	XTALPPHY	16	O	TTL	以太网PHY的XTALP
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	91	O	TTL	模拟比较器0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	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。
	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。
	VDD	8	-	电源	I/O和某些逻辑的电源正极。
	VDD	20	-	电源	I/O和某些逻辑的电源正极。
	VDD	32	-	电源	I/O和某些逻辑的电源正极。
	VDD	44	-	电源	I/O和某些逻辑的电源正极。
	VDD	56	-	电源	I/O和某些逻辑的电源正极。
	VDD	68	-	电源	I/O和某些逻辑的电源正极。
	VDD	81	-	电源	I/O和某些逻辑的电源正极。
	VDD	93	-	电源	I/O和某些逻辑的电源正极。
	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。

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。
	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
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	23	I/O	TTL	捕获/比较/PWM 3

表 **16-4. GPIO**管脚和备用功能

GPIO管脚	管脚编号	复用的功能	复用的功能
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	CCP1	
PB0	66	CCP0	
PB1	67	CCP2	
PB2	70		
PB3	71		
PB4	92	C0-	
PB5	91	C0o	
PB6	90	C0+	
PB7	89	TRST	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25		
PC5	24		
PC6	23	CCP3	
PD0	10		
PD1	11		
PD2	12		
PD3	13		
PF0	47		
PF1	61		

GPIO管脚	管脚编号	复用的功能	复用的功能
PF2	60	LED1	
PF3	59	LED0	

17 工作特性

表 **17-1.** 温度特性

特性	符号	值	单位
工作温度范围[a]	T_A	-40 to +85	°C

a. 最大的存储温度为150°C。

表 **17-2.** 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。

18 电气特性

18.1 DC特性

18.1.1 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 **18-1.** 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
以太网PHY电源电压（V_{CCPHY}）	V_{CCPHY}	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。

18.1.2 建议的 直流工作条件

表 **18-2.** 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_{CCPHY}	以太网PHY电源电压	3.0	3.3	3.6	V
V_{IH}	高电平输入电压	2.0	-	5.0	V
V_{IL}	低电平输入电压	-0.3	-	1.3	V
V_{SIH}	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

18.1.3 片内低压差（LDO）稳压器特性

表 **18-3. 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

18.1.4 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_{DDA} = 3.3 V
- V_{DDPHY} = 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 **18-4.** 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}		单位
			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 = 25 MHz（带PLL）	48	挂起[a]	64	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 = 25 MHz（带PLL）	5	挂起[a]	33	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 = 25 MHz（带PLL）	48	挂起[a]	56	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 = 25 MHz（带PLL）	5	挂起[a]	26	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 = 25 MHz（带PLL）	5	挂起[a]	12	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	4.6	挂起[a]	0.21	挂起[a]	mA

a. 挂起结束。

18.1.5 Flash存储器特性

表 **18-5. 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
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

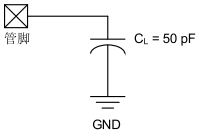
a. 编程/擦除周期定义成位从1变为0再变为1（1-> 0 -> 1）。

18.2 AC特性

18.2.1 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 **18-1.** 负载条件



18.2.2 时钟

表 **18-6.** 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 **18-7.** 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_{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	-	25	MHz
f_{system_clock}	系统时钟	0	-	25	MHz

表 **18-8.** 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

参数名称	值				单位
负载电容（典型）	16	16	16	16	pF
驱动级别（典型）	100	100	100	100	µW

18.2.3 模拟比较器

表 **18-9.** 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 **18-10.** 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

18.2.4 以太网控制器

表 **18-11. 100BASE-TX**发送器特性[a]

参数名称	Min	Nom	Max	单位
峰值输出幅度	950	-	1050	mVpk
输出幅度对称性	0.98	-	1.02	mVpk
输出超调	-	-	5	%
上升/下降时间	3	-	5	ns
上升/下降时间失衡	-	-	500	ps
占空比失真	-	-	-	ps
抖动	-	-	1.4	ns

a. 在变压器线路端（线路端）测得。

表 **18-12. 100BASE-TX**发送器特性（信息）[a]

参数名称	Min	Nom	Max	单位
回波损耗（Return loss）	16	-	-	dB
开路电感	350	-	-	µs

a. 表中的规范只是作为信息被提供。它们主要由外部变压器和用于测量的终端电阻决定。

表 **18-13. 100BASE-TX**接收器特性

参数名称	Min	Nom	Max	单位
信号检测有效阈值	600	700		mVppd
信号检测无效阈值	350	425	-	mVppd

参数名称	Min	Nom	Max	单位
差分输入电阻	20	-	-	kΩ
抖动容限（pk-pk）	4	-	-	ns
基线漂移跟踪	-75	-	+75	%
信号检测有效时间	-	-	1000	µs
信号检测无效时间	-	-	4	µs

表 **18-14. 10BASE-T**发送器特性[a]

参数名称	Min	Nom	Max	单位
峰值差分输出信号	2.2	-	2.8	V
谐波含量	27	-	-	dB
连接脉冲宽度	-	100	-	ns
空闲开始脉冲宽度	-	300 350	-	ns

a. 为模板测试Manchester编码的数据脉冲、连接脉冲和空闲开始脉冲，使用*IEEE 802.3*中14条的过程。

表 **18-15. 10BASE-T**发送器特性（信息）[a]

参数名称	Min	Nom	Max	单位
输出回波损耗（return loss）	15	-	-	dB
输出阻抗平衡	29-17log(f/10)	-	-	dB
峰值共模输出电压	-	-	50	mV
共模抑制能力	-	-	100	mV
共模抑制抖动	-	-	1	ns

a. 表中的规范只是作为信息被提供。它们主要由外部变压器和用于测量的终端电阻决定。

表 **18-16. 10BASE-T**接收器特性

参数名称	Min	Nom	Max	单位
DLL相位获取时间	-	10	-	BT
抖动容限（pk-pk）	30	-	-	ns
输入静噪阈值	500	600	700	mVppd
输入非静噪阈值	275	350	425	mVppd
差分输入电阻	-	20	-	kΩ
位错误率	-	10^{-10}	-	-
共模抑制能力	25	-	-	V

表 **18-17.** 隔离变压器[a]

名称	值	条件
匝比	1 CT : 1 CT	+/- 5%
开路电感	350 uH (min)	@ 10 mV, 10 kHz
漏电感	0.40 uH (max)	@ 1 MHz (min)
绕组间电容	25 pF (max)	
DC电阻	0.9 Ohm (max)	
插入损耗	0.4 dB (typ)	0-65 MHz

名称	值	条件
HIPOT	1500	Vrms

a. 在线路接口需要2个简单的1:1隔离变压器。为了超越FCC的要求，推荐使用带集成共模扼流圈的变压器。该表给出了建议的传输线变压器的特性。

注意： 100Base-TX幅度规范可以承受0.4dB的变压器损耗。对于带高插入损耗的传输线变压器，1.2dB的插入损耗可以通过选择**MR19**寄存器的发送幅度选择（TX0）位的合适设置来补偿。

表 **18-18.** 以太网参考晶体[a]

名称	值	条件
频率	25.00000	MHz
负载电容[b]	4[c]	pF
频率容限	±50	PPM
老化	±2	PPM/yr
温度稳定性（0°～70°）	±5	PPM
振荡模式	并联谐振，基本模式	
温度为25° C ±2° C；驱动级别＝0.5 mW		
驱动级别（典型）	50-100	µW
滤波电容（最大）	10	pF
动态电容（最小）	10	fF
Serious resistance（最大）	60	Ω
杂散响应（最大）	500kHz内 >（主响应－5 dB）	

a. 如果内部晶体振荡器被使用，则选择具有下列特性的一个晶体。

b. XTLP/XTLN两端连接的等效差分电容。

c. 如果使用带更大负载的晶体，则应当增加一个连接到地的滤波电容来弥补等效的电容偏差。

图 **18-2.** 外部**XTLP**振荡器特性

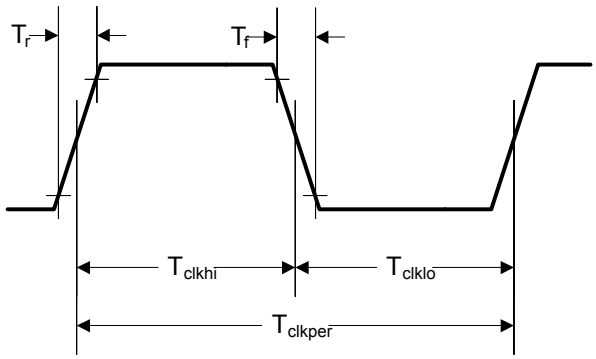


表 **18-19.** 外部**XTLP**振荡器特性

参数名称	符号	Min	Nom	Max	单位
XTLN输入低电压	$XTLN_{ILV}$	-	-	0.8	-
XTLP频率[a]	$XTLP_f$	-	25.0	-	-

参数名称	符号	Min	Nom	Max	单位
XTLP周期[b]	T_{clkper}	-	40	-	-
XTLP占空比	$XTLP_{DC}$	40 40	-	60 60	%
上升/下降时间	T_r , T_f	-	-	4.0	ns
绝对抖动		-	-	0.1	ns

a. IEEE 802.3频率容限±50ppm。

b. IEEE 802.3频率容限±50ppm。

18.2.5 同步串行接口（SSI）

表 **18-20. 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 **18-3. TI**帧格式（**FRF=01**）的**SSI**时序，单次传输的时序测量

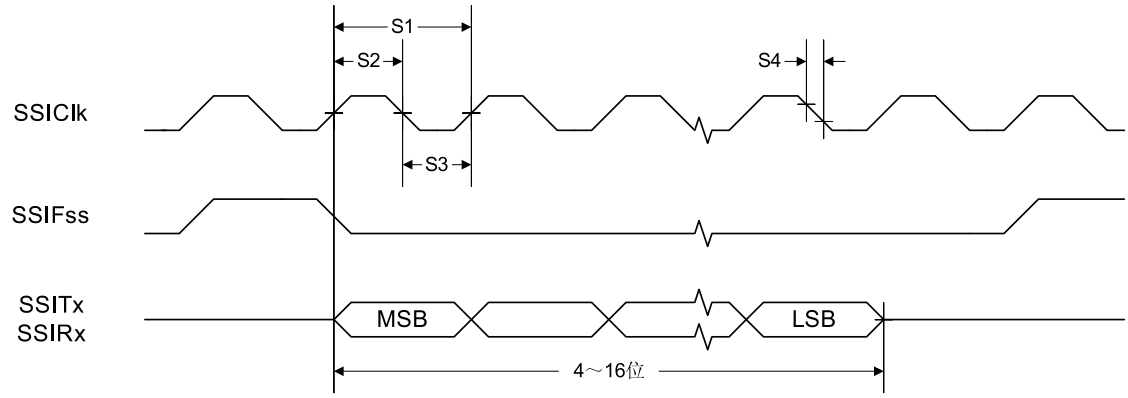


图 **18-4. MICROWIRE**帧格式（**FRF=10**）的**SSI**时序，单次传输

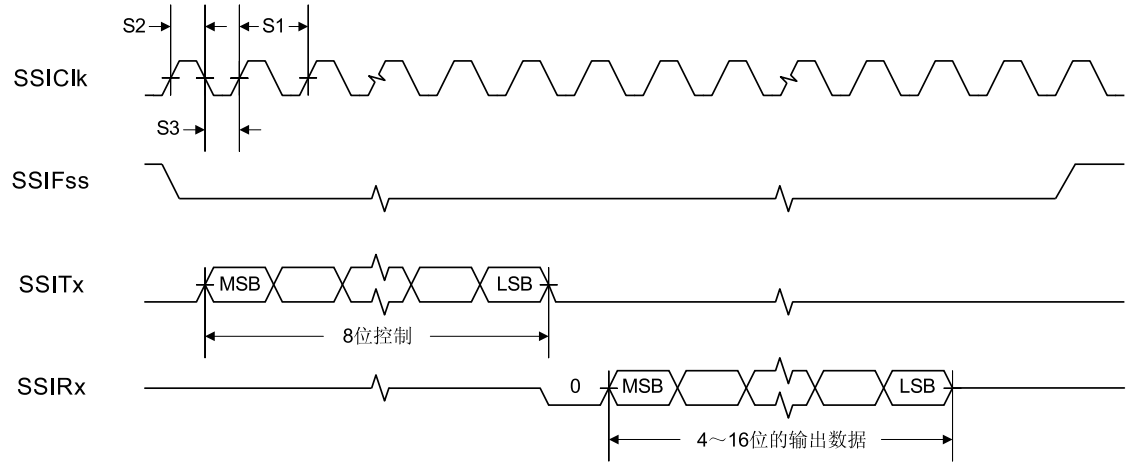
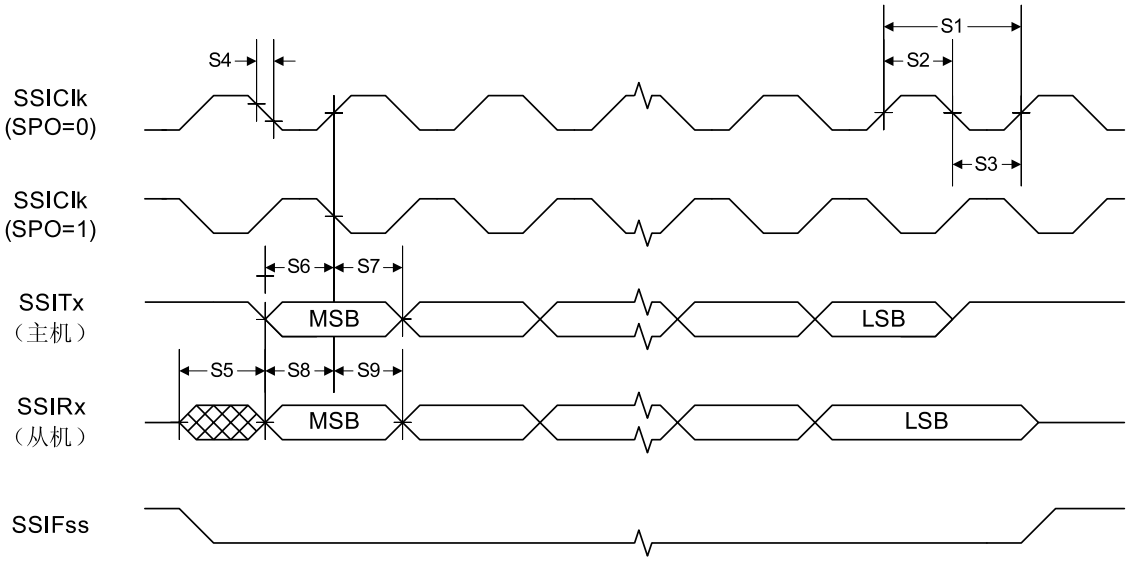


图 **18-5. SPI**帧格式（**FRF=00**），**SPH=1**



18.2.6 JTAG和边界扫描

表 **18-21. 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

参数编号	参数	参数名称	Min	Nom	Max	单位
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
J15	t_{TRST_SU}	到TCK上升沿为止的TRST建立时间	10	-	-	ns

图 **18-6. JTAG**测试时钟输入时序

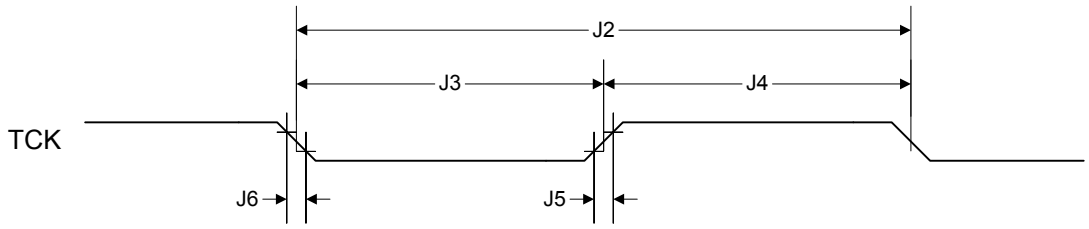


图 **18-7. JTAG**测试访问端口（**TAP**）时序

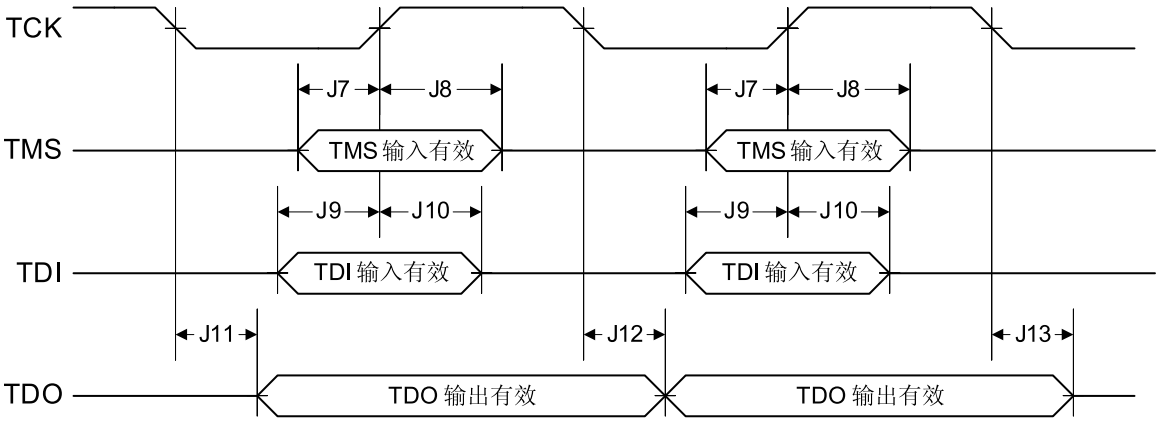
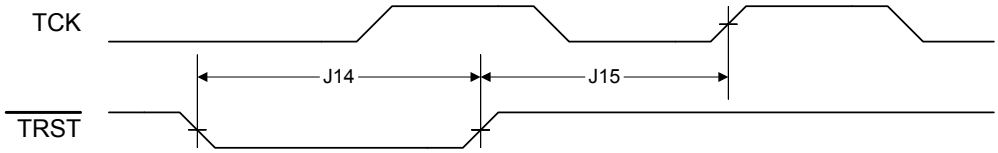


图 **18-8. JTAG TRST**时序



18.2.7 通用I/O口

注意: 所有GPIO口最大可承受5V的电压。

表 **18-22. 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

18.2.8 复位

表 **18-23.** 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（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RST脉冲宽度	2	-	-	µs

a. 20 * t_{MOSC_per}

图 **18-9.** 外部复位时序（**RST**）

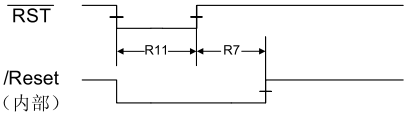


图 **18-10.** 上电复位时序

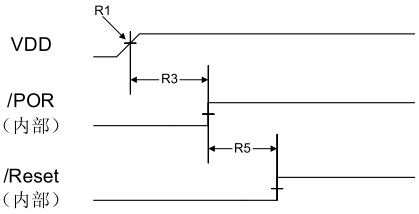


图 **18-11.** 掉电复位时序

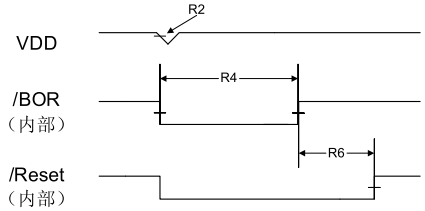


图 **18-12.** 软件复位时序

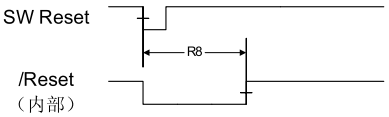
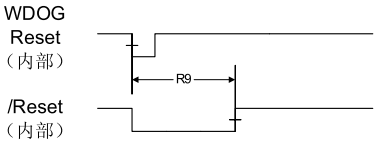
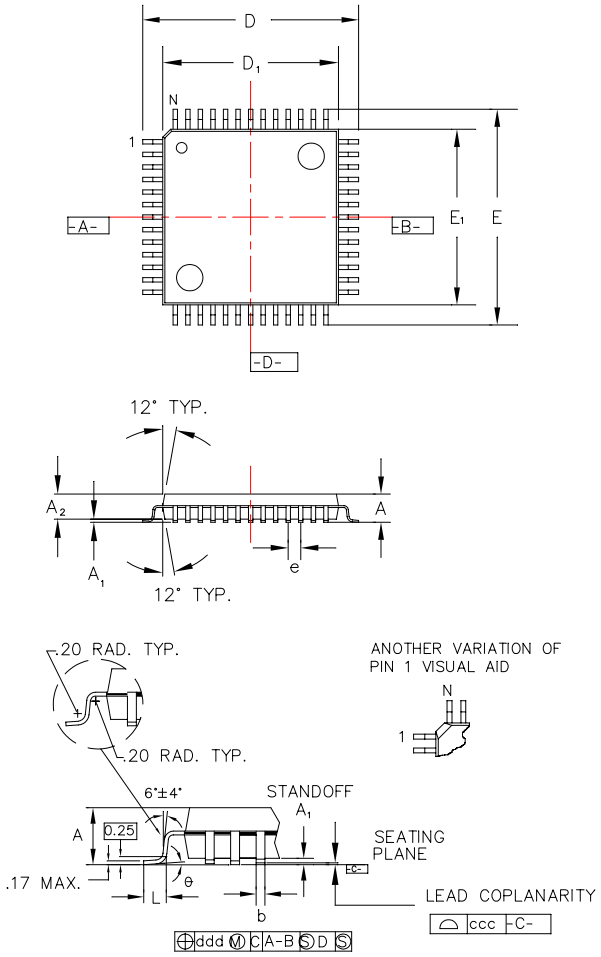


图 **18-13.** 看门狗复位时序



19 封装信息

图 **19-1. 100-**脚 **LQFP**封装



注释

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。
4. L/F: Eftec 64T Cu 或等效，0.127 mm (0.005") 或 0.152 mm (0.006") 厚度。
5. 使用变量 BED 以用于本体尺寸。

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。该加载程序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。这点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。.

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。具体内容详见传输协议中的“SSI格式”。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。这种限定会在与Flash相互作用的命令中进一步讨论。

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 订购和联系信息

B.1 订购信息

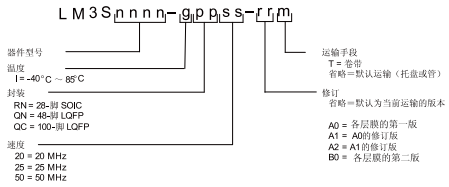


表 **B-1.** 器件订购信息

可订购的器件型号	描述
LM3S6100-IQC25	Stellaris® LM3S6100微控制器

B.2 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。

B.3 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

C 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 712 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