Preliminary

TMS320x280x DSP
系统控制和中断

参考指南

文献编号: ZHCU003
2004 年 11 月 – 修订 2005 年 5 月

TEXAS INSTRUMENTS

内容

目录 3

序言 11

1 存储器 15

1.1 闪存和 OTP 存储器 16

1.1.1 闪存 16

1.1.2 OTP 存储器 16

1.2 闪存和 OTP 功率模式 16

1.2.1 闪存和 OTP 性能 17

1.2.2 28x 闪存管道模式 18

1.2.3 更改闪存配置寄存器的规范 19

1.3 闪存和 OTP 寄存器 20

2 代码安全模块 (CSM) 27

2.1 功能说明 28

2.2 CSM 对其它片上资源的影响 30

2.3 在用户应用程序中集成代码安全 30

2.3.1 要求安全解锁的环境 31

2.3.2 密码匹配流程 33

2.3.3 具有/不具有代码安全的器件的取消保护注意事项 34

2.4 保护安全逻辑必须执行的操作和不能执行的操作 36

2.4.1 必须执行的操作 36

2.4.2 不能执行的操作 36

2.5 CSM 功能 - 总结 36

3 时钟 37

3.1 时钟和系统控制 38

3.2 OSC 和 PLL 块 45

3.2.1 基于 PLL 的时钟模块 45

3.2.2 主振荡器失败检测 45

3.2.3 XCLKOUT 生成 48

3.2.4 PLL 控制 (PLLCR) 寄存器 48

3.2.5 PLL 控制、状态和 XCLKOUT 寄存器说明 50

3.2.6 外部参考振荡器时钟选项 52

3.3 低功率模式块 52

3.4 看门狗模块 55

3.4.1 仿真注意事项 57

3.5 32 位 CPU 定时器 0/1/2 57

4 通用输入/输出 (GPIO) 63

4.1 GPIO 模块概述 64

4.2 配置概述 64

4.3 数字通用 I/O 控制 66

4.3.1 输入鉴定 67

4.4 GPIO 和外设多路复用 68

4.5 寄存器位定义 76

5 外设帧 95
5.1 外设帧寄存器 96
5.2 EALLOW 保护寄存器 98
5.3 器件仿真寄存器 102
5.5 先写后读保护 103
6 外设中断扩展 (PIE) 105
6.1 PIE 控制器概述 106
6.1.1 中断操作顺序 106
6.2 矢量表映射 108
6.3 中断源 110
6.3.1 处理多路复用中断的规范 111
6.3.2 启用和禁用多路复用外设中断的规范 112
6.3.3 从外设到 CPU 的多路复用中断请求流程 113
6.3.4 PIE 矢量表 114
6.4 PIE 配置寄存器 122
6.5 PIE 中断寄存器 123
6.5.1 PIE 中断标志寄存器 123
6.5.2 PIE 中断启用寄存器 124
6.5.3 CPU 中断标志寄存器 (IFR) 125
6.5.4 中断启用寄存器 (IER) 和调试中断启用寄存器 (DBGIER) 126
6.6 外部中断控制寄存器 130
A 修订历史记录 133

附图目录

1-1 闪存功率模式状态图 17
1-2 闪存管道 18
1-3 闪存配置存取流程图 19
1-4 闪存选项寄存器 (FOPT) 21
1-5 闪存功率寄存器 (FPWR) 21
1-6 闪存状态寄存器 (FSTATUS) 22
1-7 闪存待机等待寄存器(FSTDBYWAIT) 23
1-8 闪存待机至活动等待计数器寄存器 (FACTIVEWAIT) 23
1-9 闪存等待状态 (FBANKWAIT) 寄存器 24
1-10 OTP 等待状态寄存器 (FOTPWAIT) 25
2-1 CSM 状态和控制寄存器 (CSMSCR) 31
2-2 密码匹配流程 (PMF) 33
3-1 时钟和复位域 38
3-2 外设时钟控制寄存器 0 (PCLKCR0) 39
3-3 外设时钟控制寄存器 1 (PCLKCR1) 39
3-4 系统控制与状态寄存器 (SCSR) 42
3-5 高速外设时钟预分频器 (HISPCP) 寄存器 43
3-6 低速外设时钟预分频器 (LOSPCP) 寄存器 44
3-7 OSC 和 PLL 块 45
3-8 振荡器失败检测逻辑图 46
3-9 XCLKOUT 生成 48
3-10 PLLCR 更改规范流程图 49
3-11 PLLCR 寄存器布局 50
3-12 PLL 状态寄存器 (PLLSTS) 51
3-13 XCLKOUT 寄存器 (XCLK) 52
3-14 低功率模式控制 0 寄存器 (LPMCR0) 54
3-15 看门狗模块 55
3-16 看门狗计数器寄存器 (WDCNTR) 56
3-17 看门狗复位 Key 寄存器 (WDKEY) 56
3-18 看门狗控制寄存器 (WDCR) 57
3-19 CPU 定时器 58
3-20 CPU 定时器中断信号和输出信号 58
3-21 TIMERxTIM 寄存器 (x = 1, 2, 3) 59
3-22 TIMERxTIMH 寄存器 (x = 1, 2, 3) 59
3-23 TIMERxPRD 寄存器 (x = 1, 2, 3) 59
3-24 TIMERxPRDH 寄存器 (x = 1, 2, 3) 60
3-25 TIMERxTCR 寄存器 (x = 1, 2, 3) 61
3-26 TIMERxTPR 寄存器 (x = 1, 2, 3) 61
3-27 TIMERxTPRH 寄存器 (x = 1, 2, 3) 62
4-1 操作模式 64
4-2 输入鉴定 67
4-3 输入限定器时钟周期 67
4-4 GPIO 端口 A MUX 1 (GPAMUX1) 寄存器 76
4-5 GPIO 端口 A MUX 2 (GPAMUX2) 寄存器 78
4-6 GPIO 端口 B MUX 1 (GPBMUX1) 寄存器 81
4-7 GPIO 端口 A 鉴定控制 (GPACTRL) 寄存器 82
4-8 GPIO 端口 B 鉴定控制 (GPBCTRL) 寄存器 83
4-9 GPIO 端口 A 鉴定选择 1 (GPAQSEL1) 寄存器 84
4-10 GPIO 端口 A 鉴定选择 2 (GPAQSEL2) 寄存器 84
4-11 GPIO 端口 B 鉴定选择 1 (GPBQSEL1) 寄存器 84
4-12 GPIO 端口 A 方向 (GPADIR) 寄存器 86
4-13 GPIO 端口 B 方向 (GPBDIR) 寄存器 86

4-14 GPIO 端口 A 上拉禁用 (GPAPUD) 寄存器 87
4-15 GPIO 端口 B 上拉禁用 (GPBPUD) 寄存器 87
4-16 GPIO 端口 A 数据 (GPADAT) 寄存器 88
4-17 GPIO 端口 B 数据 (GPBDAT) 寄存器 89
4-18 GPIO 端口 A 设置、清除和转换（GPASET、GPACLEAR 和 GPATOGGLE）寄存器 90
4-19 GPIO 端口 B 设置、清除和转换（GPBSET、GPBCLEAR 和 GPBTOGGLE）寄存器 91
4-20 GPIO XINT1、XINT2、XNMI 中断选择（GPIOXINT1SEL、GPIOXINT2SEL 和 GPIOXNMISEL）寄存器 92
4-21 GPIO 低功率模式唤醒选择 (GPIOLPMSEL) 寄存器 93
5-1 器件配置 (DEVICECNF) 寄存器 102
5-2 部件 ID 寄存器 103
5-3 REVID 寄存器 103
6-1 概述：使用 PIE 块多路复用中断 106
6-2 典型的 PIE/CPU 中断响应 - INTx.y 107
6-3 复位流程图 109
6-4 外部和 PIE 中断源 110
6-5 多路复用的中断请求流程图 113
6-6 PIECTRL 寄存器（地址 CE0） 123
6-7 PIE 中断确认寄存器 (PIEACK) 寄存器（地址 CE1） 123
6-8 PIEIFRx 寄存器 (x = 1 - 12) 123
6-9 PIEIERx 寄存器 (x = 1 - 12) 124
6-10 中断标志寄存器 (IFR) — CPU 寄存器 125
6-11 中断启用寄存器 (IER) — CPU 寄存器 127
6-12 调试中断启用寄存器 (DBGIER) — CPU 寄存器 128
6-13 外部中断 1 控制寄存器 (XINT1CR)（地址 7070h） 130
6-14 外部中断 2 控制寄存器 (XINT2CR)（地址 7071h） 130
6-15 外部 NMI 中断控制寄存器 (XNMICR) — 地址 7077h 131
6-16 外部中断 1 计数器 (XINT1CTR)（地址 7078h） 131
6-17 外部中断 2 计数器 (XINT2CTR) — 地址 7079h 132
6-18 外部 NMI 中断计数器 (XNMICTR)（地址 707Fh） 132

附表目录

1-1 闪存 / OTP 配置寄存器 20
1-2 闪存选项寄存器 (FOPT) 字段说明 21
1-3 闪存功率寄存器 (FPWR) 字段说明 21
1-4 闪存状态寄存器 (FSTATUS) 字段说明 22
1-5 闪存待机等待寄存器 (FSTDBYWAIT) 字段说明 23
1-6 闪存待机至活动等待计数器寄存器 (FACTIVEWAIT) 字段说明 23
1-7 闪存等待状态寄存器 (FBANKWAIT) 字段说明 24
1-8 OTP 等待状态寄存器 (FOTPWAIT) 字段说明 25
2-1 安全级别 28
2-2 CSM 影响的 280x 资源 30
2-3 不受 CSM 影响的 280x 资源 30
2-4 代码安全模块 (CSM) 31
2-5 CSM 状态和控制寄存器 (CSMSCR) 字段说明 31
3-1 PLL、时钟、看门狗和低功率模式寄存器 39
3-2 外设时钟控制寄存器 0 (PCLKCR0) 字段说明 40
3-3 外设时钟控制寄存器 1 (PCLKCR1) 字段说明 41
3-4 系统控制与状态寄存器 (SCSR) 字段说明 42
3-5 高速外设时钟预分频器 (HISPCP) 字段说明 43
3-6 低速外设时钟预分频器 (LOSPCP) 寄存器 44
3-7 可能的 PLL 配置模式 45
3-8 PLLCR 位说明 50
3-9 PLL 状态寄存器 (PLLSTS) 字段说明 51
3-10 XCLKOUT 寄存器 (XCLK) 字段说明 52
3-11 280x 低功率模式 52
3-12 低功率模式 53
3-13 低功率模式控制寄存器 0 (LPMCR0) 字段说明 54
3-14 看门狗计数器寄存器 (WDCNTR) 字段说明 56
3-15 看门狗复位 Key 寄存器 (WDKEY) 字段说明 56
3-16 看门狗控制寄存器 (WDCR) 字段说明 57
3-17 CPU 定时器 0、1、2 配置和控制寄存器 58
3-18 TIMERxTIM 寄存器字段说明 59
3-19 TIMERxTIMH 寄存器字段说明 59
3-20 TIMERxPRD 寄存器字段说明 60
3-21 TIMERxPRDH 寄存器字段说明 60
3-22 TIMERxTCR 寄存器字段说明 61
3-23 TIMERxTPR 寄存器字段说明 62
3-24 TIMERxTPRH 寄存器字段说明 62
4-1 GPIO 控制寄存器 65
4-2 GPIO 中断和低功率模式选择寄存器 65
4-3 GPIO 数据寄存器 66
4-4 外设输入的默认状态 69
4-5 2808 GPIO MUX 表 70
4-6 2806 GPIO MUX 表 71
4-7 2801 GPIO MUX 表 72
4-8 外设与 GPIO 交叉参考 73
4-9 GPIO 端口 A MUX 1 (GPAMUX1) 寄存器字段说明 76
4-10 GPIO 端口 A MUX 2 (GPAMUX2) 寄存器字段说明 78
4-11 GPIO 端口 B MUX 1 (GPBMUX1) 寄存器字段说明 81
4-12 GPIO 端口 B MUX 2 (GPBMUX2) 寄存器字段说明 81

4-13 GPIO 端口 A 鉴定控制 (GPACTRL) 寄存器字段说明 82
4-14 GPIO 端口 B 输入鉴定控制 (GPBCTRL) 寄存器字段说明 83
4-15 GPIO 端口 A 鉴定选择 1 (GPAQSEL1) 寄存器字段说明 84
4-16 GPIO 端口 A 鉴定选择 2 (GPAQSEL2) 寄存器字段说明 84
4.6 GPIO 端口 B 鉴定选择 1 (GPBQSEL1) 寄存器字段说明 85
4-17 GPIO 端口 B 鉴定选择 2 (GPBQSEL2) 寄存器字段说明 85
4-18 GPIO 端口 A 方向 (GPADIR) 寄存器字段说明 86
4-19 GPIO 端口 B 方向 (GPBDIR) 寄存器字段说明 86
4-20 GPIO 端口 A 内部上拉禁用 (GPAPUD) 寄存器字段说明 87
4-21 GPIO 端口 B 上拉禁用 (GPBPUD) 寄存器字段说明 87
4-22 GPIO 端口 A 数据 (GPADAT) 寄存器字段说明 88
4-23 GPIO 端口 B 数据 (GPBDAT) 寄存器字段说明 89
4-24 GPIO 端口 A 设置 (GPASET) 寄存器字段说明 90
4-25 GPIO 端口 A 清除 (GPACLEAR) 寄存器字段说明 90
4-26 GPIO 端口 A 转换 (GPATOGGLE) 寄存器字段说明 90
4-27 GPIO 端口 B 设置 (GPBSET) 寄存器字段说明 91
4-28 GPIO 端口 B 清除 (GPBCLEAR) 寄存器字段说明 91
4-29 GPIO 端口 B 转换 (GPBTOGGLE) 寄存器字段说明 91
4-30 GPIO XINT1 中断选择 (GPIOXINT1SEL) 寄存器字段说明 92
4-31 GPIO XINT2 中断选择 (GPIOXINT2SEL) 寄存器字段说明 92
4-32 GPIO XNMI 中断选择 (GPIOXNMISEL) 寄存器字段说明 92
4-33 GPIO 低功率模式唤醒选择 (GPIOLPMSEL) 寄存器字段说明 93
5-1 外设帧 0 寄存器 96
5-2 外设帧 1 寄存器 96
5-3 外设帧 2 寄存器 97
5-4 存取受 EALLOW 保护的寄存器 98
5-5 EALLOW 保护的器件仿真寄存器 98
5-6 EALLOW 保护闪存 / OTP 配置寄存器 98
5-7 EALLOW 保护代码安全模块 (CSM) 寄存器 98
5-8 EALLOW 保护的 PIE 矢量表 99
5-9 EALLOW 保护 PLL、时钟、看门狗和低功率模式寄存器 100
5-10 EALLOW 保护 GPIO MUX 寄存器 100
5-11 EALLOW 保护 eCAN-A 寄存器 101
5-12 EALLOW 保护 ePWM1 - ePWM3 寄存器 101
5-13 EALLOW 保护 ePWM4 - ePWM6 寄存器 101
5-14 器件仿真寄存器 102
5-15 DEVICECNF 寄存器字段说明 102
5.4 PARTID 寄存器字段说明 103
5-16 REVID 寄存器字段说明 103
5-17 PROTSTART 和 PROTRANGE 寄存器 103
5-18 PROTSTART 有效值 104
5-19 PROTRANGE 有效值 104
6-1 启用中断 107
6-2 中断矢量表映射 108
6-3 复位操作之后的矢量表映射 108
6-4 280x PIE 多路复用的外设中断矢量表 115
6-5 280x PIE 矢量表 117
6-6 PIE 配置和控制寄存器 122
6-7 PIECTRL 寄存器地址字段说明 123
6-8 PIE 中断确认寄存器 (PIEACK) 字段说明 123
6-9 PIEIFRx 寄存器字段说明 124

6-10 PIEIERx 寄存器 (x = 1 - 12) 字段说明 124
6-11 中断标志寄存器 (IFR) – CPU 寄存器字段说明 125
6-12 中断启用寄存器 (IER) – CPU 寄存器字段说明 127
6-13 调试中断启用寄存器 (DBGIER) – CPU 寄存器字段说明 128
6-14 外部中断 1 控制寄存器 (XINT1CR) 字段说明 130
6-15 外部中断 2 控制寄存器 (XINT2CR) 字段说明 130
6-16 外部 NMI 中断控制寄存器 (XNMICR) 字段说明 131
6-17 XNMICR 寄存器设置和中断源 131
6-18 外部中断 1 计数器 (XINT1CTR) 字段说明 131
6.7 外部中断 2 计数器 (XINT2CTR) 字段说明 132
6-19 外部 NMI 中断计数器 (XNMICTR) 字段说明 132
A-1 对修订 A 的更改 133

请先阅读

关于本手册

本参考指南适用于 TMS320x280x 系列处理器上的系统控制和中断。 包括 280x 系列中所有基于闪存、基于 ROM 和基于 RAM 的器件。

本指南描述各种 280x 数字信号处理器 (DSP) 系统控制和中断如何与外设一起工作。 包括有关以下各项的信息：

- 闪存和一次性可编程 (OTP) 存储器
- 代码安全模块 (CSM)，TMS320C28x™ 器件中集成的一种安全功能。
- 时钟机制，包括振荡器、PLL、XCLKOUT、看门狗模块和低功率模式。 另外，还描述了 32 位 CPU 定时器。
- GPIO MUX 寄存器，用于选择 280x 器件上的共享引脚的操作。
- 访问外设帧以写入和读取器件上的各种外设寄存器。
- 中断源，包括外部中断源和外设中断扩展 (PIE) 块，后者将大量中断源多路复用成较小的中断输入集。

命名惯例

本文档使用以下惯例。

- 表示十六进制数时加一个后缀 h 或一个前缀 0x。 例如，以下数字为十六进制的 40（十进制的 64）：40h 或 0x40。
- 本文档中含有寄存器的图形显示和表格说明。
  - 每个寄存器图形显示为一个分成多个字段的矩形，每个字段分别代表了此寄存器的字段。 每个域用其位名标记，域的起始位和结束位标记在标签的上面，域的读取/写入属性标记在下面， 并用图例解释了用于表示属性的符号。
  - 寄存器图形中的保留位指定一位用于将来器件扩展。

德州仪器 （TI） 提供的相关文档

下列书籍描述了 TMS320x280x 及可从 TI 网站获得的相关支持工具：

数据手册 –

SPRS230： – TMS320F2801、TMS320F2806、TMS320F2808、UCD9501 数字信号处理器数据手册

包含 F280x 器件的外引脚、信号说明以及电子和时序规范。

应用报告 –

SPRAA58： – TMS320x281x 到 TMS320x280x 迁移概述

描述了德州仪器 (TI) 的 TMS320x281x 与 TMS320x280x DSP 之间的差异，以便在将应用从 281x 迁移到 280x 的过程中提供帮助。 尽管本文档侧重从 281x 到 280x 的迁移，想要反向迁移（从 280x 到 281x）的用户也会发现本文档非常有用。

SPRA550： – 用于数字电机控制的 3.3V DSP

描述了仅使用 3.3V 电机控制器的方案，并指出对于大多数应用，3.3V 与 5V 之间不存在明显的连接问题。 还对比讨论了片上 3.3V 模数转换器 (ADC) 与 5V ADC。 概述了可以降低系统噪声和电磁干扰影响的组件布局和印刷电路板 (PCB) 设计指南。

SPRA820： – TMS320C28x DSP 在线堆栈溢出检测

介绍 TMS320C28x™ DSP 在线堆栈溢出检测的方法。 提供了包含一些函数的 C 源代码，这些函数用于在 DSP/BIOS™ 和非 DSP/BIOS 应用中执行溢出检测。

SPRA861： – RAMDISK：用户自定义的 C I/O 驱动程序示例

提供了在任意器件上使用高级 CIO 功能的复杂缓冲技术的简易方法。 本应用报告介绍了用户自定义的器件驱动程序的实施示例。

SPRA873： – 使用 TMS320F2812 DSP 和 DRV592 功率放大器的热电制冷器控制

介绍了由德州仪器 (TI) 的 TMS320F2812 数字信号处理器 (DSP) 和 DRV592 功率放大器组成的热电制冷器系统。 DSP 使用集成的 12 位模数转换器读取热敏电阻，并将脉宽调制的波形直接输出到 H 桥接的 DRV592 功率放大器，以实现数字比例积分微分反馈控制器。 全面地描述了试验系统以及软件和软件操作指南。

SPRA876： – TMS320F281x eCAN 的编程示例

包含几个编程示例，以阐述如何为不同的操作模式设置 eCAN 模块，以帮助您实现快速 eCAN 编程。附加的 SPRA876.zip 文件中包含所有项目和 CANalyzer 配置文件。

SPRA953： – IC 封装热度量

描述了传统的热度量和新的热度量，并展望其在系统级结温估值中的有关应用。

SPRA958： – 从 TMS320F281x DSP 上的内部闪存运行应用程序（修订版 B）

讨论了正确配置从片上闪存执行应用软件所需的要求。 提供了对 DSP/BIOS™ 和非 DSP/BIOS 项目的要求。 包括示例代码项目。

SPRA963： – TMS320LF24x 和 TMS320F281x 器件的可靠性数据

TMS320LF24x 和 TMS320F281x 器件应用报告的可靠性数据

SPRA989： – F2810、F2811 和 F2812 ADC 校准

描述了提高 F2810/F2811/F2812 器件上的 12 位模数转换器 (ADC) 绝对精度的方法。 本应用手册附带一个从 F2812 eZdsp 上的 RAM 执行的示例程序 (ADCcalibration.zip)。

SPRA991： – 仿真成功增强了调试和分析 - 白皮书

讨论了通过允许开发人员更有效地评估系统替代方案缩短开发周期的仿真增强。

用户指南 –

SPRU051： – TMS320x281x、280x 串行通信接口 (SCI) 参考指南

描述了 SCI 这一通常称作 UART 的两线异步串行端口。 SCI 模块支持 CPU 与其它异步外设之间的使用标准非归零 (NRZ) 格式的数字通信。

SPRU059： – TMS320x281x、280x 串行通信接口 (SPI) 参考指南

描述了 SPI（一种高速同步串行输入/输出 (I/O) 端口）它可使编程长度（1 - 16 位）的串行比特流以程序设定的比特传输速率移入和移出器件。

SPRU074： – TMS320x281x、280x 增强控制器局域网 (eCAN) 参考指南

描述了在电噪声环境下使用确定的协议与其它控制器进行串行通信的 eCAN。 使用 32 个完全可配置的邮箱和时间戳功能，eCAN 模块提供强大可靠的多功能串行通信接口。 281x DSP 中实施的 eCAN 模块与 CAN 2.0B 标准（现用）兼容。

SPRU430： – TMS320C28x DSP CPU 和指令集参考指南

描述了 TMS320C28x 定点数字信号处理器 (DSP) 的中央处理器 (CPU) 以及汇编语言指令。 它还描述了这些 DSP 上可用的仿真功能。

SPRU513： – TMS320C28x 汇编语言工具用户指南

描述了用于 TMS320C28x 器件的汇编语言工具（汇编器和其它用于开发汇编语言代码的工具）、汇编指令、宏、常用对象文件格式和符号调试指令。

SPRU514： – TMS320C28x 优化 C 编译器用户指南

描述了 TMS320C28x C/C++ 编译器。 此编译器接受 ANSI 标准 C/C++ 源代码，并为 TMS320C28x 器件生成 TMS320 DSP 汇编语言源代码。

SPRU608： – TMS320C28x 指令集仿真器技术概述

描述了 TMS320C2000 IDE 的 Code Composer Studio 内可用于仿真 C28x 内核指令集的仿真器。

SPRU625： – TMS320C28x DSP/BIOS 应用编程接口 (API) 参考指南

描述了使用 DSP/BIOS 进行的开发。

SPRU716： – TMS320x280x 模数转换器 (ADC) 参考指南

描述了如何配置和使用片上 ADC 模块，它是一个 12 位管道结构 ADC。

SPRU721： – TMS320x280x 内部集成电路 (I2C) 参考指南

描述了 TMS320x280x 数字信号处理器 (DSP) 上可用的内部集成电路 (I^2C) 模块的功能和操作。

SPRU722： – TMS320x280x 引导 ROM 参考指南

描述了 bootloader（工厂编程的引导加载软件）的用途和功能。 它还描述了器件的片上引导 ROM 的其它内容，并标识了所有信息在该存储器内的位置。

SPRU790： – TMS320x280x 增强型正交编码器脉冲 (eQEP) 参考指南

描述了 eQEP 模块，在高性能运动和定位控制系统中，该模块用于与线性或旋转增量编码器连接，以从旋转机器中获取位置、方向和速度信息。 它包括模块说明和寄存器。

SPRU791： – TMS320x280x 增强型脉宽调制器 (ePWM) 模块参考指南

描述了增强型脉宽调制器的主要应用领域，包括数字电机控制、开关模式电源控制、UPS（不间断电源）和其它形式的电源转换。

SPRU807： – TMS320x280x 增强型捕捉 (eCAP) 模块参考指南

描述了增强型捕捉模块。 它包括模块说明和寄存器。

SPRU924： – 高分辨率脉宽调制器 (HRPWM)

本文档描述了脉宽调制器的高分辨率扩展 (HRPWM) 的操作。

商标

Code Composer Studio 是 德州仪器 (TI) 的商标

TMS320C28x, Code Composer Studio 是 Texas Instruments 的商标

TEXAS INSTRUMENTS

存储器

本章描述了配置 28x 数字信号处理器 (DSP) 器件上的闪存和一次性可编程 (OTP) 存储器的等待状态和操作模式的正确顺序。 还包括有关闪存和 OTP 功率模式以及如何通过启用闪存管道模式提高闪存性能的信息。

在只有 ROM 的器件上，此信息适用于替换闪存和 OTP 的 ROM。

主题 页

1.1 闪存和 OTP 存储器 16
1.2 闪存和 OTP 功率模式 16
1.3 闪存和 OTP 寄存器 20

1.1 闪存和 OTP 存储器

本节描述如何配置两种存储器 - 闪存和一次性可编程 (OTP) 存储器。 在只有 ROM 的器件上，此信息适用于替换闪存和 OTP 的 ROM。

1.1.1 闪存

在程序和数据存储器空间中统一地映射片上闪存。 此闪存在 28x 器件上始终启用，且具有以下特性：

- 多个扇区
  可以擦除的最少量闪存是一个扇区。 具有多个扇区就可以选择让某些扇区继续处于编程状态而仅擦除特定扇区。
- 代码安全性
  闪存由代码安全模块 (CSM) 保护。 通过将密码编程到闪存中，用户可以防止未授权人员访问闪存。 请参阅第 2 章以了解有关使用代码安全模块的信息。
- 低功率模式
  为了在未使用闪存时节省功耗，提供了两个级别的低功率模式。 请参阅第 1.2 部分以了解有关可用闪存功率模式的详细信息。
- 可配置的等待状态
  可以基于 CPU 频率调节可配置的等待状态，以便在给定执行速度获得最佳性能。
- 增强的性能
  提供了闪存管道模式以提高线性代码执行的性能。

1.1.2 OTP 存储器

在程序和数据存储器空间中统一地映射 1K x 16 的一次性可编程 (OTP) 存储器块。 这样，OTP 就可用于编程数据或代码。 此存储器块与闪存不同，只能编程一次且不能擦除。

1.2 闪存和 OTP 功率模式

下列操作状态适用于闪存和 OTP 存储器：

- 复位或休眠状态
  这是器件复位之后的状态。 在此状态下，库和泵处于休眠状态（最低功率）。 闪存处于休眠状态时，CPU 对闪存或 OTP 存储器映射区进行的数据读取或操作代码提取将自动使功率模式改变为待机状态，然后改变为活动状态。 在向活动状态转换的期间，将自动使 CPU 等候。 一旦完成向活动状态的转换，CPU 存取将如常完成。
- 待机状态
  在此状态下，库和泵处于待机功率模式状态。 此种状态使用的功耗比休眠状态要大，但转换至活动或读取状态所需的时间较短。 当闪存处于待机状态时，CPU 对闪存或 OTP 存储器映射区进行的数据读取或操作代码提取将自动使功率模式改变为活动状态。 在向活动状态转换的期间，将自动使 CPU 等候。 一旦闪存 / OTP 到达活动状态，CPU 存取将如常完成。
- 活动或读取状态
  在此状态下，库和泵处于活动功率模式状态（最高功率）。 CPU 对闪存 / OTP 存储器映射区的读取或提取存取等待状态由 FBANKWAIT 和 FOTPWAIT 寄存器控制。 还可以启用称作闪存管道的预取机制来提高线性代码执行的提取性能。

> 注：
> 在引导过程中，280x 引导 ROM 对闪存中代码安全模块 (CSM) 密码所在的地址执行虚拟读取。 执行这种读取是为了解锁未存储密码的新的或已擦除的器件，以便可以执行闪存编程或将代码装入 CSM 保护的 SARAM。 在存储了密码的器件上，这种读取无效且 CSM 仍然被锁定（请参阅第 2 章以了解有关 CSM 的信息）。 这种读取产生的一个影响是闪存将从休眠（复位）状态转换至活动状态。

闪存 / OTP 库和泵总是处于相同的功率模式。 请参阅图 1-1 以了解可用功率状态的示意图。 您可以按如下方式更改当前闪存 / OTP 存储器功率状态：

- 更改至较低的功率状态

将 PWR 模式位从较高的功率模式更改为较低的功率模式。 这种更改瞬间将闪存 / OTP 槽更改为较低的功率状态。 应只用闪存 / OTP 存储器之外运行的代码存取此寄存器。

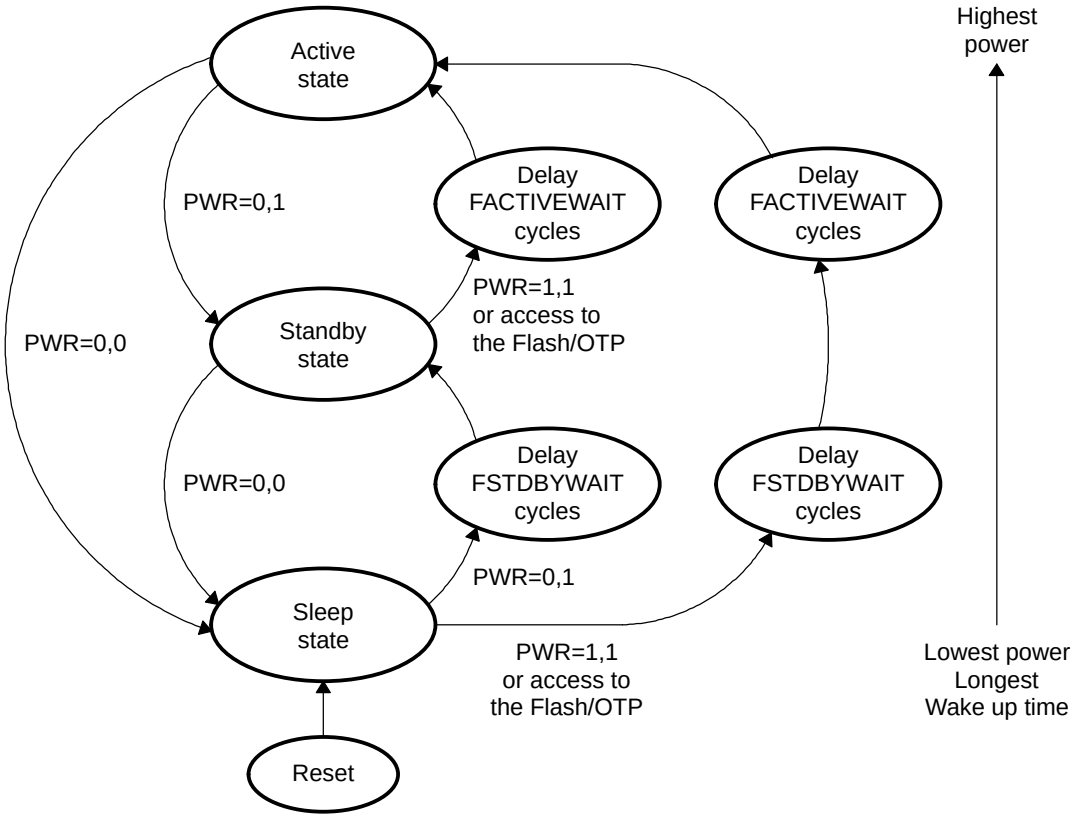
- 更改至较高的功率状态

  要从较低的功率状态更改至较高的功率状态，有两种选择。

  1. 将 FPWR 寄存器从较低的状态更改为较高的状态。 这种存取使闪存 / OTP 存储器转至较高的状态。
  2. 通过读取存取或程序操作代码提取存取来存取闪存或 OTP 存储器。 这种存取自动使闪存 / OTP 存储器转至活动状态。

从较低的功率状态转至较高的功率状态时，存在一个延迟。 请参阅图 1-1。 这种延迟是必要的，便于使闪存在较高的功率模式中稳定下来。 如果在此延迟期间发生对闪存 / OTP 存储器的任何存取，则 CPU 自动等候直至延迟结束。

图 1-1. 闪存功率模式状态图



延迟的持续时间由 FSTDBYWAIT 和 FACTIVEWAIT 寄存器确定。 从休眠状态更改至待机状态的延迟时间由 FSTDBYWAIT 寄存器确定。 从待机状态更改至活动状态的延迟时间由 FACTIVEWAIT 寄存器确定。 从休眠模式（最低功率）更改至活动模式（最高功率）的延迟时间由 FSTDBYWAIT 和 FACTIVEWAIT 确定。 这些寄存器应保持它们的默认状态。

1.2.1 闪存和 OTP 性能

CPU 对闪存 / OTP 的读取或数据提取操作可以采用下列形式之一：

- 32 位指令存取
- 16 位或 32 位数据空间读取
- 16 位程序空间读取

一旦闪存处于活动功率状态，则对存储器库映射区的读取或提取存取可以分为闪存存取或 OTP 存取两类。

主闪存阵列由行和列组成。 行包含 2048 位信息。 对闪存和 OTP 的存取可以为以下三种类型之一：

1. 闪存随机存取

   对 2048 位的行的第一次存取可以视为随机存取。

2. 闪存分页存取

对行的第一次存取可以视为随机存取，而同一行内的后续存取视为分页存取。

可以通过对 FBANKWAIT 寄存器编程来配置随机存取和分页存取的等待状态数量。 随机存取使用的等待状态的数量由 RANDWAIT 位控制，分页存取使用的等待状态的数量由 PAGEWAIT 位控制。 FBANKWAIT 寄存器默认为最差情况下的等待状态数量，因此需要初始化为合适数目的等待状态，以基于 CPU 时钟频率和闪存的存取时间提高性能。 当 PAGEWAIT 位设置为 0 时，闪存支持“零等待”存取。 这假设 CPU 速度足够低，可以满足存取时间。 要确定随机存取和分页存取的时间要求，请参阅特定器件的数据手册。

在 ROM 器件上，保留相同的等待状态配置以便与闪存器件的时序兼容。

3. OTP 存取

对 OTP 的读取或提取存取由 FOTPWAIT 寄存器中的 OTPWAIT 位控制。 存取 OTP 所用的时间比存取闪存长且没有分页模式。 与闪存一样，在只有 ROM 的器件上，OTP 被 ROM 替换，允许与 OTP 相同的等待状态配置。 要确定 OTP 存取时间要求，请参阅特定器件的数据手册。

使用闪存时请记住以下其它几个要点：

- 忽略 CPU 对闪存或 OTP 存储器映射区的写入。 它们在单个周期内完成。
- 当代码安全模块 (CSM) 受保护时，读取安全区外的闪存 / OTP 存储器映射区所用的周期数与正常存取的相同。 但是，读取操作返回 0。
- CSM 密码地址的读取通过硬件连线设置为 16 个等待状态。 PAGEWAIT 和 RANDOMWAIT 位不影响这些地址。请参阅第 2 章以了解有关 CSM 的信息。

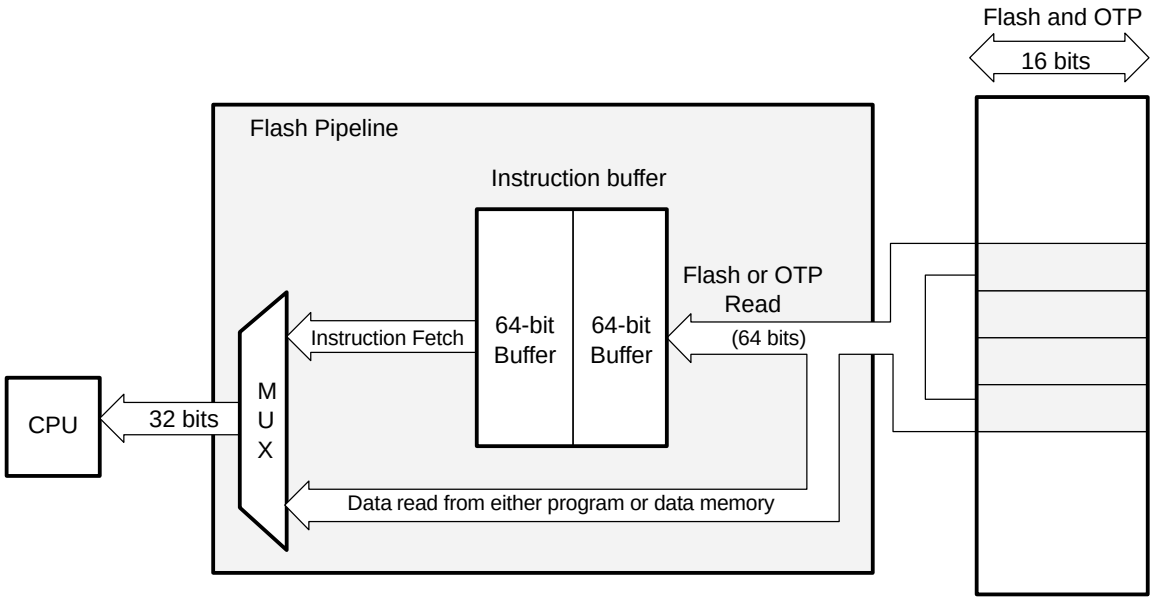
1.2.2 28x 闪存管道模式

闪存通常用于存储应用程序代码。 在代码执行期间，将从连续的存储器地址提取指令，出现不连续情况时例外。 通常，位于连续地址中的代码部分构成了应用程序代码的主要部分且称作线性代码。 为提高线性代码执行的性能，已实施了闪存管道模式。 默认情况下，闪存管道功能禁用。 在 FOPT 寄存器中设置 ENPIPE 位可以启用此模式。 闪存管道模式独立于 CPU 管道。 为了能够维持闪存器件与 ROM 器件之间的代码时序兼容性，也已在 ROM 器件上实施了闪存管道模式。

每次存取时，闪存或 OTP 的一个指令提取将读出 64 位。 存取闪存时的起始地址自动与 64 位边界对齐，以便指令位置位于要提取的 64 位以内。 启用闪存管道模式（参见图 1-2）时，指令提取读取的 64 位存储在 64 位宽 2 层深的指令预取缓冲器中。 然后，此预取缓冲器的内容被发送至 CPU 以便根据需要进行处理。

单次 64 位存取最多可以包含两条 32 位指令或四条 16 位指令。 大多数 C28x 指令为 16 位，因此对于从闪存库进行的每次 64 位指令提取，就象预取缓冲器中有多达四条指令准备通过 CPU 处理。 在 CPU 处理这些指令期间，闪存管道自动启动对闪存库的另一次存取以预取下一个 64 位。 通过这种方式，闪存管道模式在后台工作以使指令预取缓冲器尽可能满。 使用此技术，将显著提高从闪存或 OTP 执行连续代码的整体效率。

图 1-2. 闪存管道



仅当执行诸如分支、BANZ、调用或循环导致 PC 不连续时，才中止闪存管道预取。 当出现此种情况时，预取中止，并清理预取缓冲器的内容。 出现此情况时有两种可能的情形：

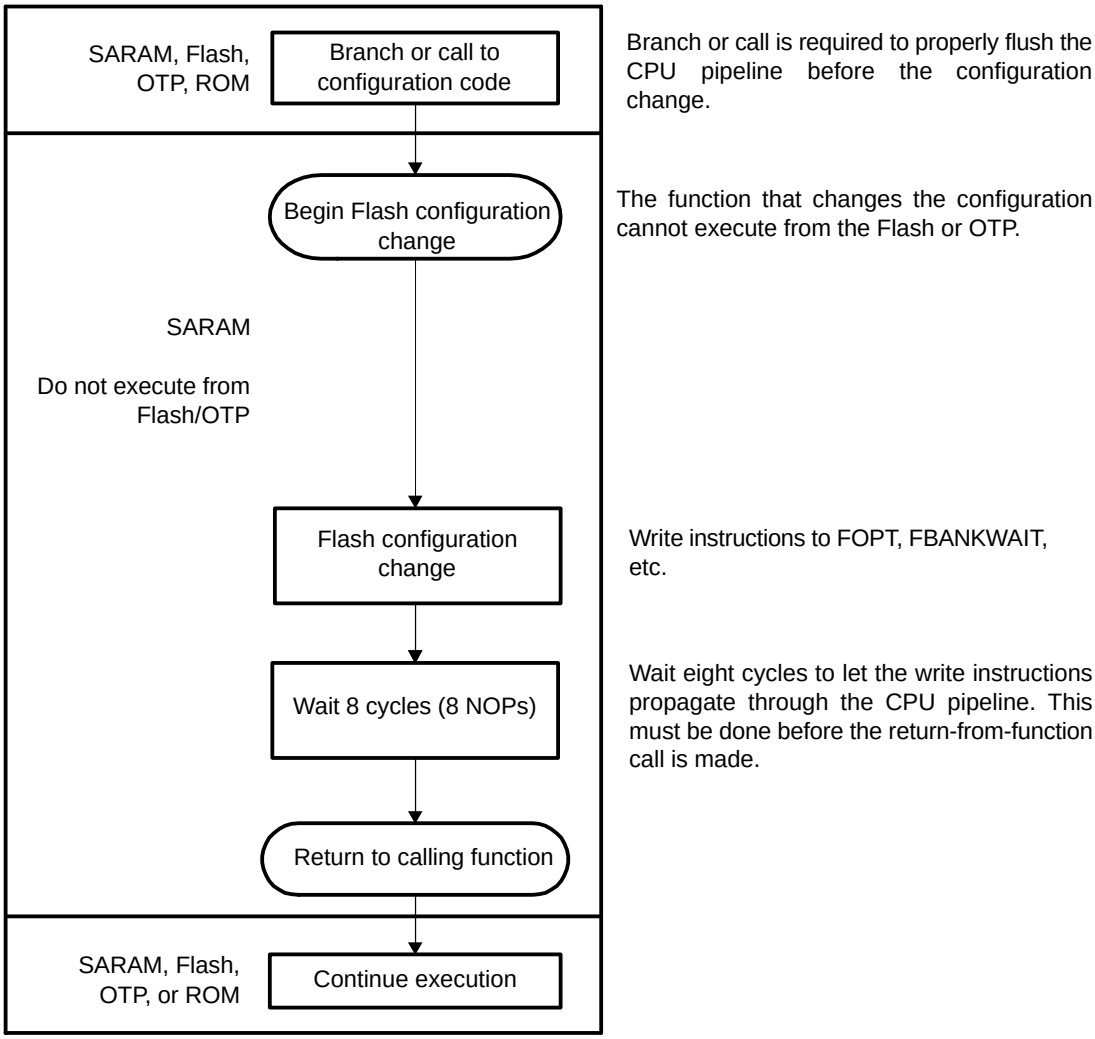
1. 如果目的地址在闪存或 OTP 内，则预取中止并在目的地址重新开始。
2. 如果目的地址在闪存和 OTP 外，则预取中止，且仅当分支转回闪存或 OTP 时才再次开始。 闪存管道预取机制仅适用于从程序空间进行的指令提取。 从数据存储器和程序存储器进行的数据读取不利用预取缓冲器能力，因此绕过了预取缓冲器。 例如，MAC、DMAC 和 PREAD 之类的指令从程序存储器读取数据值。 当发生这种读取时，将绕过预取缓冲器但不清理该缓冲器。 如果在启动数据读取操作时指令预取已在进行，则数据读取将等候直至预取完成。

1.2.3 更改闪存配置寄存器的规范

在闪存配置期间，不能对闪存或 OTP 进行任何存取。 包括仍在 CPU 管道中的指令、数据读取和指令预取操作。 为确保配置更改期间不进行任何存取，对于修改 FOPT、FPWR、FBANKWAIT 或 FOTPWAIT 寄存器的任何代码，应遵循图 1-3 中所示的规范。

此规范也适用于已用 ROM 替换闪存和 OTP 的器件上的 ROM。

图 1-3. 闪存配置存取流程图



1.3 闪存和 OTP 寄存器

The 闪存和 OTP 存储器可以由表 1-1。 所有配置寄存器均受 EALLOW 保护。 位说明位于图 1-4 至图 1-10。

表 1-1. 闪存 / OTP 配置寄存器

名称(1)(2)	地址	大小 (x16)	说明
FOPT	0x0A80	1	闪存选项寄存器
保留	0x0A81	1	保留
FPWR	0x0A82	1	闪存功率模式寄存器
FSTATUS	0x0A83	1	状态寄存器
FSTDBYWAIT (3)	0x0A84	1	闪存休眠到待机等待寄存器
FACTIVEWAIT (3)	0x0A85	1	闪存待机到活动等待寄存器
FBANKWAIT	0x0A86	1	闪存读取存取等待状态寄存器
FOTPWAIT	0x0A87	1	OTP 读取存取等待状态寄存器

(1) 这些寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解有关信息。
(2) 这些寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。
(3) 这些寄存器应保持它们的默认状态。

注:

不应通过从 OTP 或闪存运行的代码写入闪存配置寄存器，也不应在可能正在存取闪存或 OTP 时写入闪存配置寄存器。 应从在闪存 / OTP 存储器之外执行的代码执行对闪存寄存器的所有寄存器存取，且不应在闪存 / OTP 上的所有活动完成之前尝试存取。 未讨论防止此操作的硬件。

总之，可以从在闪存 / OTP 中执行的代码读取闪存寄存器；但是，不要写入这些寄存器。

仅可以通过执行 EALLOW 指令启用 CPU 对闪存配置寄存器的写入存取。 当执行 EDIS 指令时，写入存取被禁用。 这可以防止寄存器被欺骗性存取。 始终可以进行读取存取。 可以通过 JTAG 端口存取这些寄存器，而不必执行 EALLOW。 请参阅第 5.2 部分以了解有关 EALLOW 保护的信息。 这些寄存器同时支持 16 位和 32 位存取。

图 1-4. 闪存选项寄存器 (FOPT)

15	1	0
保留		ENPIPE
R-0		R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-2. 闪存选项寄存器 (FOPT) 字段说明

位	字段	值	说明(1)(2)(3)
15-1	保留		
0	ENPIPE		启用闪存管道模式位。 当设置此位时，闪存管道模式有效。 管道模式通过预取指令来提高指令提取的性能。 请参阅第 1.2.2 部分以获得详细信息。 当启用管道模式时，闪存等待状态（分页和随机）必须大于 0。 在闪存器件上，ENPIPE 影响从闪存和 OTP 进行的提取。 在 ROM 器件上，ENPIPE 影响从替换闪存和 OTP 的 ROM 块进行的提取。
		0	闪存管道模式无效。 （默认值）
		1	闪存管道模式有效。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。
(3) 当写入此寄存器时，请遵循第 1.2.3 部分。

图 1-5. 闪存功率寄存器 (FPWR)

15	2	1	0
保留		PWR	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-3. 闪存功率寄存器 (FPWR) 字段说明

位	字段	值	说明(1)(2)
15-2	保留		
1-0	PWR		闪存功率模式位。 写入这些位会更改闪存库和泵的当前功率模式。 请参阅第 1.2 部分以了解有关更改闪存库功率模式的详细信息。 在 ROM 器件上，更改 PWR 对 ROM 的功耗没有影响。 与闪存器件上完全一样，改变为待机模式或休眠模式将导致从 ROM 进行的下一次存取被延迟。
		00	库和泵休眠（最低功率）
		01	泵和库待机
		10	保留（无影响）
		11	库和泵有效（最高功率）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。

图 1-6. 闪存状态寄存器 (FSTATUS)

15				9	8
保留					3VSTAT
R-0					R/W1C-0

7	4	3	2	1	0
保留		ACTIVEWAITS	STDBYWAITS	PWRS	
R-0		R-0	R-0	R-0	

图例：R/W = 读/写；R = 只读；W1C = 写入 1 以清除；*-n* = 复位后的值

表 1-4. 闪存状态寄存器 (FSTATUS) 字段说明

位	字段	值	说明(1)(2)
15-9	保留		保留
8	3VSTAT		闪存电压 (V_{DD3VFL}) 状态锁定位。 进行设置后，此位用于指示从泵模块发出的 3VSTAT 信号转至高电平。 此位表明闪存 3.3V 电源超出允许的范围。
		0	忽略 0 的写入。
		1	当此位读取 1 时，表明闪存 3.3V 电源超出允许的范围。
			通过写入 1 清除此位。
7-4	保留		保留
3	ACTIVEWAITS		库和泵待机至活动等待计数器状态位。 此位表明相应的等待计数器是否超出访问的等待时间。
		0	此计数器当前未计时。
		1	此计数器当前正在计时。
2	STDBYWAITS		库和泵休眠至至待机等待计数器状态位。 此位表明相应的等待计数器是否超出访问的等待时间。
		0	此计数器当前未计时。
		1	此计数器当前正在计时。
1-0	PWRS		功率模式状态位。 这些位指示闪存 / OTP 当前处于哪种功率模式。
			仅在适当的定时延迟到期之后，PWRS 位才设置为新的功率模式。
		00	库和泵处于休眠模式（最低功耗）
		01	库和泵处于待机模式
		10	保留
		11	泵和库处于工作态和读取模式（最高功耗）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。

图 1-7. 闪存待机等待寄存器(FSTDBYWAIT)

15	9	8	0
保留		STDBYWAIT	
R-0		R/W-1	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-5. 闪存待机等待寄存器 (FSTDBYWAIT) 字段说明

位	字段	值	说明(1)(2)
15-9	保留	0	保留
8-0	STDBYWAIT		**此寄存器应处于默认状态。** 库和泵休眠至待机等待计数。
		111111111	511 个 SYSCLKOUT 周期（默认值）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。

图 1-8. 闪存待机至活动等待计数器寄存器 (FACTIVEWAIT)

7	4	3	0
保留		ACTIVEWAIT	
R-0		R/W-1	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-6. 闪存待机至活动等待计数器寄存器 (FACTIVEWAIT) 字段说明

位	字段	值	说明(1)(2)
15-9	保留	0	保留
8-0	ACTIVEWAIT		**此寄存器应处于默认状态。** 库和泵待机至活动等待计数：
		111111111	511 个 SYSCLKOUT 周期（默认值）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。

图 1-9. 闪存等待状态 (FBANKWAIT) 寄存器

15	12	11	8	7	4	3	0
保留		PAGEWAIT		保留		RANDWAIT	
R-0		R/W-1		R-0		R/W-1	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-7. 闪存等待状态寄存器 (FBANKWAIT) 字段说明

位	字段	值	说明(1)(2)(3)
15-12	保留		保留
11-8	PAGEWAIT		闪存分页读取等待状态 这些寄存器位指定对闪存库执行分页读取操作时的等待状态数，以 CPU 时钟周期为单位（0..15 个 SYSCLKOUT 周期）。 请参阅第 1.2.1 部分以获得详细信息。 请参阅各种器件的数据手册以了解分页闪存或 ROM 存取所需的最少时间。 必须将 RANDWAIT 设置为大于或等于 PAGEWAIT 设置的值。 没有提供硬件来检测大于 RANDWAIT 的 PAGEWAIT 值。 在 ROM 器件上，这些位影响替换闪存的 ROM 块的等待状态。
		0000	每个分页存取 0 个等待状态，每个存取总计 1 个 SYSCLKOUT 周期。 如果启用了管道模式，则 PAGEWAIT 必须大于 0。
		0001	每个分页闪存存取 1 个等待状态，或每个存取总计 2 个 SYSCLKOUT 周期。
		0010	每个分页闪存存取 2 个等待状态，或每个存取总计 3 个 SYSCLKOUT 周期。
		0011	每个分页闪存存取 3 个等待状态，或每个存取总计 4 个 SYSCLKOUT 周期。
		。 。 。	。 。 。
		1111	每个分页闪存存取 15 个等待状态，或每个存取总计 16 个 SYSCLKOUT 周期。 （默认值）
7-4	保留		保留
3-0	RANDWAIT		闪存随机读取等待状态 这些寄存器位指定对闪存库执行随机读取操作时的等待状态数，以 CPU 时钟周期为单位（0..15 个 SYSCLKOUT 周期）。 请参阅第 1.2.1 部分以获得详细信息。 请参阅各种器件的数据手册以了解随机闪存或 ROM 存取所需的最少时间。 RANDWAIT 必须大于 0。 即，必须至少使用 1 个随机等待状态。 必须将 RANDWAIT 设置为大于或等于 PAGEWAIT 设置的值。 器件将不检测和校正大于 RANDWAIT 的 PAGEWAIT 值。 在 ROM 器件上，这些位影响替换闪存的 ROM 块的等待状态。
		0000	非法值。 RANDWAIT 必须大于 0。
		0001	每个随机闪存存取 1 个等待状态，或每个存取总计 2 个 SYSCLKOUT 周期。
		0010	每个随机闪存存取 2 个等待状态，或每个存取总计 3 个 SYSCLKOUT 周期。
		0011	每个随机闪存存取 3 个等待状态，或每个存取总计 4 个 SYSCLKOUT 周期。
		。 。 。	。 。 。
		1111	每个随机闪存存取 15 个等待状态，或每个存取总计 16 个 SYSCLKOUT 周期。 （默认值）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。
(3) 当写入此寄存器时，请遵循第 1.2.3 部分。

图 1-10. OTP 等待状态寄存器 (FOTPWAIT)

15	5	4	0
保留		OTPWAIT	
R-0		R/W-1	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 1-8. OTP 等待状态寄存器 (FOTPWAIT) 字段说明

位	字段	值	说明(1)(2)(3)
15-5	保留	0	保留
4-0	OTPWAIT		OTP 读取等待状态。 这些寄存器位指定对 OTP 执行读取操作时的等待状态数，以 CPU 时钟周期为单位（1..31 个 SYSCLKOUT 周期）。 有关详细信息，请参阅“CPU 对闪存 / OTP 进行读取或提取存取”一节。 OTP 中没有分页模式。 OTPWAIT 必须大于 0。 即，必须至少使用 1 个等待状态。 请参阅各种器件的数据手册以了解 OTP 或 ROM 存取所需的最少时间。 在 ROM 器件上，这些位影响替换 OTP 的 ROM 块的等待状态。
		00000	非法值。 OTPWAIT 必须设置为 1 或更大的值。
		00001	对于每个存取总计 2 个 SYSCLKOUT 周期的情况，每个 OTP 存取将使用 1 个等待状态。
		00010	对于每个存取总计 3 个 SYSCLKOUT 周期的情况，每个 OTP 存取将使用 2 个等待状态。
		00011	对于每个存取总计 4 个 SYSCLKOUT 周期的情况，每个 OTP 存取将使用 3 个等待状态。
		。 。 。	。 。 。
		11111	对于每个存取总计 32 个 SYSCLKOUT 周期的情况，每个 OTP 存取将使用 31 个等待状态。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此寄存器受代码安全模块 (CSM) 保护。 请参阅第 2 章以获得详细信息。
(3) 当写入此寄存器时，请遵循第 1.2.3 部分。

TEXAS INSTRUMENTS

代码安全模块 (CSM)

代码安全模块 (CSM) 是集成在 28x 器件中的一种安全功能。 它防止未授权人员存取/查看片上存储器，即防止对专有代码进行复制/反向工程。

单词“ 安全 ”的意思是对片上存储器的存取进行保护。 单词“ 不安全 ”的意思是对片上存储器的存取不进行保护，即，可以通过任何方法读取存储器的内容（例如，通过如 Code Composer Studio ™之类的调试工具）。

主题 页

2.1 功能说明 28
2.2 CSM 对其它片上资源的影响 30
2.3 在用户应用程序中集成代码安全 30
2.4 保护安全逻辑必须执行的操作和不能执行的操作 36
2.5 CSM 功能 - 总结 36

2.1 功能说明

安全模块限制 CPU 对某个片上存储器的存取。 这可以有效阻止通过 JTAG 端口或外设对各种存储器进行读取和写入存取。 安全性的定义与片上存储器的存取有关并防止对专有代码或数据进行未授权的复制。

当限制 CPU 对片上安全存储器位置的存取时，器件受保护。 而此时，可能有两种保护级别，取决于程序计数器当前指向的位置。 如果代码当前正从安全存储器内运行，则只阻止通过 JTAG（即仿真器）的存取。 这允许安全代码存取安全数据。 相反，如果代码正从非安全存储器运行，则阻止对安全存储器的所有存取。 用户代码可以动态跳进和跳出安全存储器，因此允许从非安全存储器进行安全函数调用。 类似地，即使从非安全存储器运行主程序循环，也可以将中断服务例程放在安全存储器中。

由用于保护或取消保护器件的 128 位数据（8 个 16 位字）的密码保护安全性。 此密码存储在闪存或 ROM 末尾、称作密码地址的 8 个字中。

取消保护器件的方法是执行密码匹配流程 (PMF) 详情如第 2.3.2 部分。 表 2-1 显示了安全级别。

表 2-1. 安全级别

是否用正确的密码执行了 PMF?	操作模式	程序提取位置	安全说明
不支持	安全	安全存储器之外	只允许提取安全存储器
不支持	安全	安全存储器之内	CPU 具有全部存取权限。 JTAG 端口不能读取受保护的存储器内容。
支持	不安全	任何地方	CPU 和 JTAG 端口对安全存储器具有全部存取权限

密码存储在闪存 / ROM 存储器中的代码安全密码地址 (PWL)(0x003F 7FF8-0x003F 7FFF)。 这些地址存储由系统设计人员预定的密码。

如果密码地址将所有 128 位设置为 1，则表示该器件未受保护。 由于新的闪存器件已擦除闪存（全为 1），所以只需读取一下密码地址就可以使器件进入未保护模式。 如果密码地址将所有 128 位设置为 0，则不管 KEY 寄存器的内容如何，器件受保护。 不要使用全为 0 的密码，并且不要在擦除闪存期间使器件复位。 在擦除例程期间使器件复位会导致全为 0 或未知的密码。 如果在密码地址全为 0 时使器件复位，则通过执行第 2.3.2 部分。 使用全为 0 的密码将严重限制调试安全代码或重新编程闪存的能力。

注:

如果在密码地址全为 0 或为未知值时使器件复位，则除非闪存或 OTP 中嵌入了从安全 SARAM 运行闪存擦除例程的方法，否则将永久锁定该器件。 实施此过程时请务必小心谨慎，以免引入安全漏洞。

用于取消保护器件的用户可访问寄存器（8 个 16 位字）称作密钥寄存器。 这些寄存器映射到存储器空间中的地址 0x0000 0AE0 - 0x0000 0AE7 且受 EALLOW 保护。

注:

使用代码安全时保留的闪存位置

对于代码安全操作，0x3F7F80 与 0x3F7FF5 之间的所有地址不能用作程序代码或数据，但在编程代码安全密码时必须编程为 0x0000。 如果安全性不是很重要，则这些地址可用于代码或数据。

注:

代码安全模块免责声明

本器件上包含的代码安全模块 (CSM) 设计用于为相关存储器（ROM 或闪存）中存储的数据提供密码保护，且由德州仪器 (TI) 根据其标准条款和条件提供保修，遵从 TI 发布的适用于本器件的保修期规范。

但是，TI 不保证或表示 CSM 不会被危害或破坏，或不能通过其它方法存取关联的存储器中存储的数据。 而且，除了上述内容外，TI 也未对本器件的 CSM 或操作做任何保证或表示，包括任何隐含的用于特定用途的商用性或适用性保证。

在任何情况下，TI 对以任何方法使用 CSM 或本器件产生的任何必然、特殊、间接、偶然或严重伤害不负任何责任，无论 TI 是否被告知存在这种伤害的可能性。 排除的伤害包括但不限于数据丢失、信誉损失、无法使用、业务中断或其它经济损失。

2.2 CSM 对其它片上资源的影响

CSM 影响对表 2-2 所示：

表 2-2. CSM 影响的 280x 资源

地址	块
0x0000 0A80-0x0000 0A87	闪存配置寄存器
0x0000 8000-0x0000 8FFF	L0 SARAM (4K X 16)
0x0000 9000-0x0000 9FFF	L1 SARAM (4K X 16)
0x003D 7800-0x003D 7BFF	一次性可编程 (OTP) 存储器或 ROM (1K X 16)
0x003E 8000-0x003F 7FFF	闪存或 ROM（64K X 16、32 X 16 或 16 X 16）
0x003F 8000-0x003F 8FFF	L0 SARAM (4K X 16)，镜像
0x003F 9000-0x003F 9FFF	L1 SARAM (4K X 16)，镜像

代码安全模块不影响以下片上资源上的任何内容：

- 单存取 RAM (SARAM) 块未指定为安全的 - 无论器件是处于安全模式还是不安全模式，都可以自由存取这些存储器块和运行它们上面的代码。
- 引导 ROM 内容 - 的可查看性不受 CSM 影响。
- 片上外设寄存器 - 无论器件是处于安全模式还是不安全模式，外设寄存器都可以被从片上或片外存储器运行的代码初始化。
- PIE 矢量表 - 无论器件是处于安全模式还是不安全模式，都可以读取和写入矢量表。 表 2-2 与表 2-3 显示哪些片上资源受（或不受）280x 器件上的 CSM 影响。 对于其它器件，请参阅各个器件的数据表。

表 2-3. 不受 CSM 影响的 280x 资源

地址	块
0x0000 0000-0x0000 03FF	M0 SARAM (1K X 16)
0x0000 0400-0x0000 07FF	M1 SARAM (1K X16)
0x0000 0800-0x0000 0CFF	外设帧 0 (2K X 16)
0x0000 0D00-0x0000 0FFF	PIE 矢量 RAM (256 X 16)
0x0000 6000-0x0000 6FFF	外设帧 1 (4K X 16)
0x0000 A000-0x0000 BFFF	H0 SARAM (8K X 16)
0x0000 7000-0x0000 7FFF	外设帧 2 (4K X 16)
0x003F A000-0x003F BFFF	H0 SARAM (8K X 16) 镜像
0x003F F000-0x003F FFFF	引导 ROM (4K X 16)

总之，可以通过 JTAG 连接器将代码装入 表 2-3 中显示的未受保护的片上程序 SARAM，而不受代码安全模块的任何影响。 无论器件是处于安全模式还是不安全模式，都可以调试代码和初始化外设寄存器。

2.3 在用户应用程序中集成代码安全

在项目的开发阶段通常不要求代码安全；但是如果要开发强大可靠的代码，则需要代码安全。 在闪存中编程（或固化到 ROM 中）这样的代码之前，应选择一个密码来保护器件。 一旦设置了密码，则器件受保护（即在合适的地址编程密码并执行器件复位或设置 FORCESEC 位 (CSMSCR.15) 是保护器件的措施）。 从那时起，通过任何方法（通过 JTAG、从外部/片上存储器运行的代码等）存取或调试安全存储器的内容都需要提供有效的密码。 运行安全存储器之外的代码（如典型的最终客户应用中的代码）不需要密码；但是出于调试目的存取安全存储器内容时需要密码。

表 2-4. 代码安全模块 (CSM)

存储器地址	寄存器名称	复位值	寄存器说明
KEY 寄存器			
0x0000-0AE0	KEY0(1)	0xFFFF	128 位 KEY 寄存器的低位字
0x0000-0AE1	KEY1(1)	0xFFFF	128 位 KEY 寄存器的第二位字
0x0000-0AE2	KEY2(1)	0xFFFF	128 位 KEY 寄存器的第三位字
0x0000-0AE3	KEY3(1)	0xFFFF	128 位 KEY 寄存器的第四位字
0x0000-0AE4	KEY4(1)	0xFFFF	128 位 KEY 寄存器的第五位字
0x0000-0AE5	KEY5(1)	0xFFFF	128 位 KEY 寄存器的第六位字
0x0000-0AE6	KEY6(1)	0xFFFF	128 位 KEY 寄存器的第七位字
0x0000-0AE7	KEY7(1)	0xFFFF	128 位 KEY 寄存器的高位字
0x0000-0AEF	CSMSCR(1)	0x005F	CSM 状态和控制寄存器
闪存中的密码地址 (PWL) - 保留仅供 CSM 密码使用			
0x003F-7FF8	PWL0	用户自定义	128 位密码的低位字
0x003F-7FF9	PWL1	用户自定义	128 位密码的第二位字
0x003F-7FFA	PWL2	用户自定义	128 位密码的第三位字
0x003F-7FFB	PWL3	用户自定义	128 位密码的第四位字
0x003F-7FFC	PWL4	用户自定义	128 位密码的第五位字
0x003F-7FFD	PWL5	用户自定义	128 位密码的第六位字
0x003F-7FFE	PWL6	用户自定义	128 位密码的第七位字
0x003F-7FFF	PWL7	用户自定义	128 位密码的高位字

(1) 这些寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 2-1. CSM 状态和控制寄存器 (CSMSCR)

15	14 ... 7	6 ... 1	0
FORCESEC	保留	保留	SECURE
R/W-1	R-0	R-10111	R-1

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 2-5. CSM 状态和控制寄存器 (CSMSCR) 字段说明

位	字段	值	说明(1)
15	FORCESEC		写入 1 会清除 KEY 寄存器并保护器件。
		0	读取始终返回 0。
		1	清除 KEY 寄存器并保护器件。 必须按照 第 2.3.2 部分 中所述的密码匹配流程来再次取消保护器件。
14-1	保留		保留
0	SECURE		反映器件的安全状态的只读位。
		0	器件不安全（CSM 已解锁）。
		1	器件安全（CSM 已锁定）。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

2.3.1 要求安全解锁的环境

下列是要求取消保护的典型情况：

- 使用调试器（如 Code Composer Studio™）进行代码开发。
  这是产品设计阶段最常见的环境。
- 闪存编程，使用 TI 的 ，如 Code Composer Studio™ F28xx 片上闪存编程器插件。
  在代码开发和测试期间，闪存编程非常常见。 用户提供必要的密码之后，闪存实用程序禁用安全逻辑，然后尝试对闪存编程。 由于新器件附带已擦除的闪存，所以闪存实用程序不需要任何授权就可禁用新器

件中的代码安全逻辑。 但是，重新编程器件时（已包含自定义密码）要求为闪存实用程序提供密码，以便解锁器件以启用编程。 在使用 TI 提供的闪存 API 的自定义编程解决方案中，通过从安全存储器执行闪存编程算法可以避免解锁 CSM。

- 此应用程序定义的自定义环境

除了上述环境之外，在以下情况中也要求存取安全存储器内容：

- 使用片上 bootloader 将代码或数据装入安全 SARAM 或擦除/编程闪存。
- 从片上未受保护的存储器执行代码并要求存取安全存储器以获取查找表。 建议不要进行这样的操作，因为从外部代码提供密码会危及代码安全。

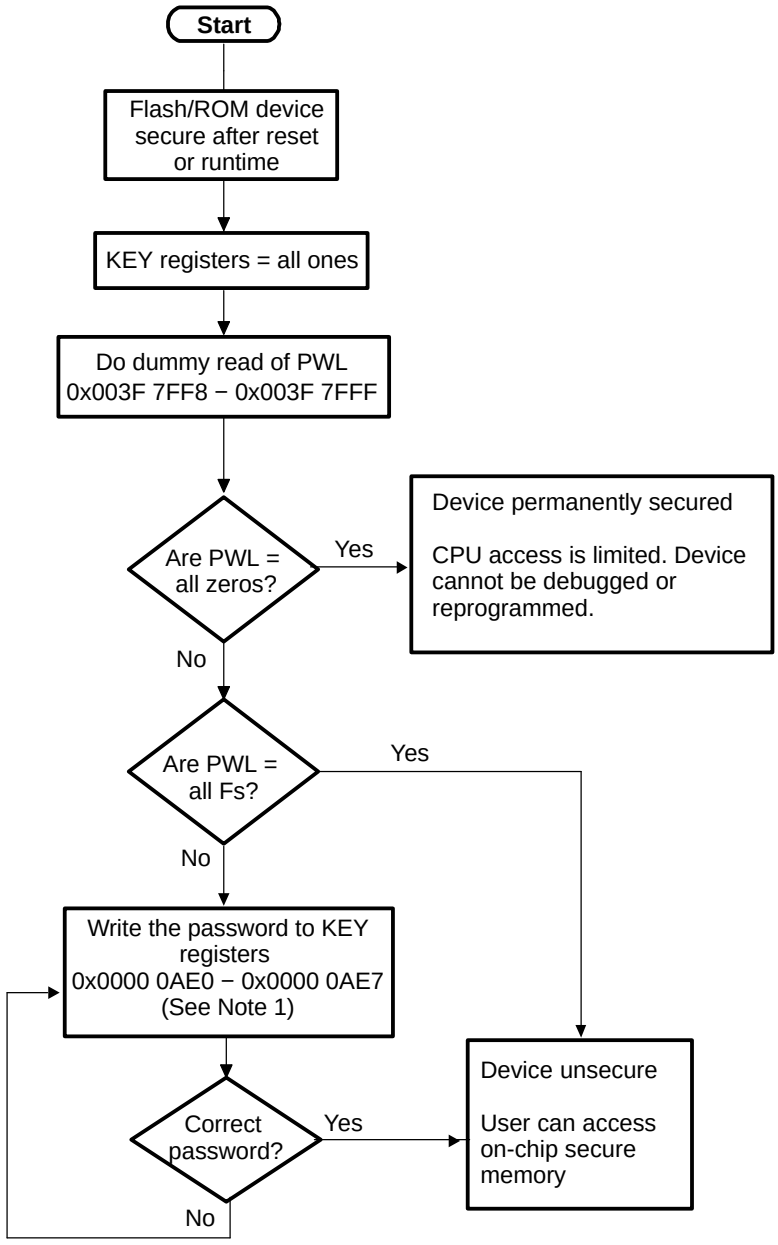
在所有上述情况中，取消保护顺序相同。 这种顺序简称为*密码匹配流程 (PMF)*。 图 2-2 说明了用户每次尝试取消保护器件时所需的操作顺序。 为说明起见，列出了一个代码示例。

2.3.2 密码匹配流程

密码匹配流程 (PMF) 本质上是一连串对密码地址 (PWL) 的 8 个虚拟读取和随后的 8 个对 。

图 2-2 显示了 PMF 如何帮助 和禁用安全逻辑。

图 2-2. 密码匹配流程 (PMF)



A KEY 寄存器受 EALLOW 保护。

2.3.3 具有/不具有代码安全的器件的取消保护注意事项

情形 1 和情形 2 提供了具有和不具有代码安全的器件的取消保护注意事项。

情形 1：具有代码安全的器件

具有代码安全的器件应在密码地址（存储器中的 0x003F 7FF8-0x003F 7FFF）存储了预定的密码。 另外，地址 0x3F7F80 - 0x3F7FF5 应全部编程为 0x0000 且不用于程序和/或数据存储。 以下是取消保护此器件的步骤：

1. 执行密码地址的虚拟读取。
2. 将密码写入 KEY 寄存器（存储器中的地址 0x0000 0AE0-0x0000 0AE7）。
3. 如果密码正确，则器件变为不受保护；否则它仍受保护。

情形 2：不具有代码安全的器件

不具有代码安全的器件应在密码地址存储 0x FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF（128 位全为 1）。以下是使用此器件的步骤：

1. 在复位时，CSM 将锁定 CSM 保护的存储器区域。
2. 执行密码地址的虚拟读取。
3. 由于密码为全 1，所以此操作将解锁所有存储器区域。 完成此操作之后，立即可以完全存取安全存储器。

注：

即使未用密码保护器件（所有密码地址全为 1），CSM 也将在复位时锁定。 因此，如果执行存取的代码从 CSM 保护的存储器区域之外执行，则仍然必须先在这些器件上执行虚拟读取操作，再读取、写入或编程安全存储器。 为了方便，引导 ROM 代码会执行这种虚拟读取。

2.3.3.1 取消保护的 C 代码示例

```
volatile int *CSM = (volatile int *)0x000AE0; //CSM register file volatile int *PWL = (volatile int *)0x3F7FF8;
//Password location volatile int tmp;

int i;

    // Read the 128-bits of the password locations (PWL)     // in flash/ROM at address 0x3F7FF8-0x3F7FFF     // If
the device is secure, then the values read will     // not actually be loaded into the temp variable, so     // this
is called a dummy read.

for (i=0; i<8; i++) tmp = *PWL++;

    // If the password locations (PWL) are all = ones (0xFFFF),     // then the device will now be unsecure. If the
password     // is not all ones (0xFFFF), then the code below is required     // to unsecure the CSM.     // Write
the 128-bit password to the KEY registers     // If this password matches that stored in the     // PWL then the CSM
will become unsecure. If it does not     // match, then the device will remain secure.     // An example password of:
    // 0x0123456789ABCDEFCDEF89AB45670123 is used.

asm("  EALLOW"); // Key registers are EALLOW protected *CSM++ = 0x0123; // Register KEY0 at 0xAE0 *CSM++ = 0x4567; //
Register KEY1 at 0xAE1 *CSM++ = 0x89AB; // Register KEY2 at 0xAE2 *CSM++ = 0xCDEF; // Register KEY3 at 0xAE3 *CSM++ =
0xCDEF; // Register KEY4 at 0xAE4 *CSM++ = 0x89AB; // Register KEY5 at 0xAE5 *CSM++ = 0x4567; // Register KEY6 at
0xAE6 *CSM++ = 0x0123; // Register KEY7 at 0xAE7 asm(" EDIS");
```

2.3.3.2 再次保护的 C 代码示例

```
volatile int *CSMSCR = 0x0000AEF;  //CSMSCR register
                                   //Set FORCESEC bit
asm("  EALLOW");                   //CSMSCR register is EALLOW protected.
*CSMSCR = 0x8000;
asm("EDIS");
```

2.4 保护安全逻辑必须执行的操作和不能执行的操作

2.4.1 必须执行的操作

- 为了简化调试和代码开发阶段，请在取消保护模式下使用器件；即在密码地址的所有 128 位中使用 1（或使用容易记住的密码）。 请在开发阶段之后固化代码时使用密码。
- 在使用闪存实用程序编程 COFF 文件之前，请再次检查密码地址中存储的密码。
- 可以在安全存储器和不安全存储器之间自由前后切换代码执行的流程，而不会危害安全性。 要在器件被保护时存取位于安全存储器中的数据变量，必须从安全存储器运行代码。
- 当使用 CSM 时，请用 0x0000 编程地址 0x3F7F80 - 0x3F7FF5。

2.4.2 不能执行的操作

- 如果需要代码安全，请不要在应用程序中除密码地址外的任何地方嵌入密码，否则可能危害安全性。
- 不要使用 128 位全 0 作为密码。 这会自动保护器件，而不管 KEY 寄存器的内容。 该器件将不可调试也不可重新编程。
- 不要在闪存阵列的擦除操作期间进行复位。 这会在密码地址留下 0 或未知值。 如果密码地址在复位期间全为 0，则器件始终受保护，而不管 KEY 寄存器的内容。
- 不要使用地址 0x3F7F80 - 0x3F7FF5 存储程序和/或数据。 当使用 CSM 时，这些地址应编程为 0x0000。

2.5 CSM 功能 - 总结

1. 在复位之后闪存被保护，直到执行了第 2.3.2 部分中描述的密码匹配流程。
2. 在闪存或 ROM 之外运行代码的标准方法是用代码编程闪存（对于 ROM 器件，在器件制造时已固化了程序）并给 DSP 上电。 由于始终允许从安全存储器提取指令，所以不管 CSM 是什么状态，即使没有执行密码匹配流程，代码也会正确工作。
3. 在器件受保护时，
4. 不能通过从非安全存储器执行代码来修改安全存储器。
5. 器件受保护的任何时间内，调试器（如 Code Composer Studio™）不能读取或写入安全存储器。
6. 当器件不受保护时，将具有从 CPU 代码和调试器存取安全存储器的全部存取权限。

TEXAS INSTRUMENTS

时钟

本节描述振荡器、PLL 和时钟机制、看门狗功能以及低功率模式。

主题 页

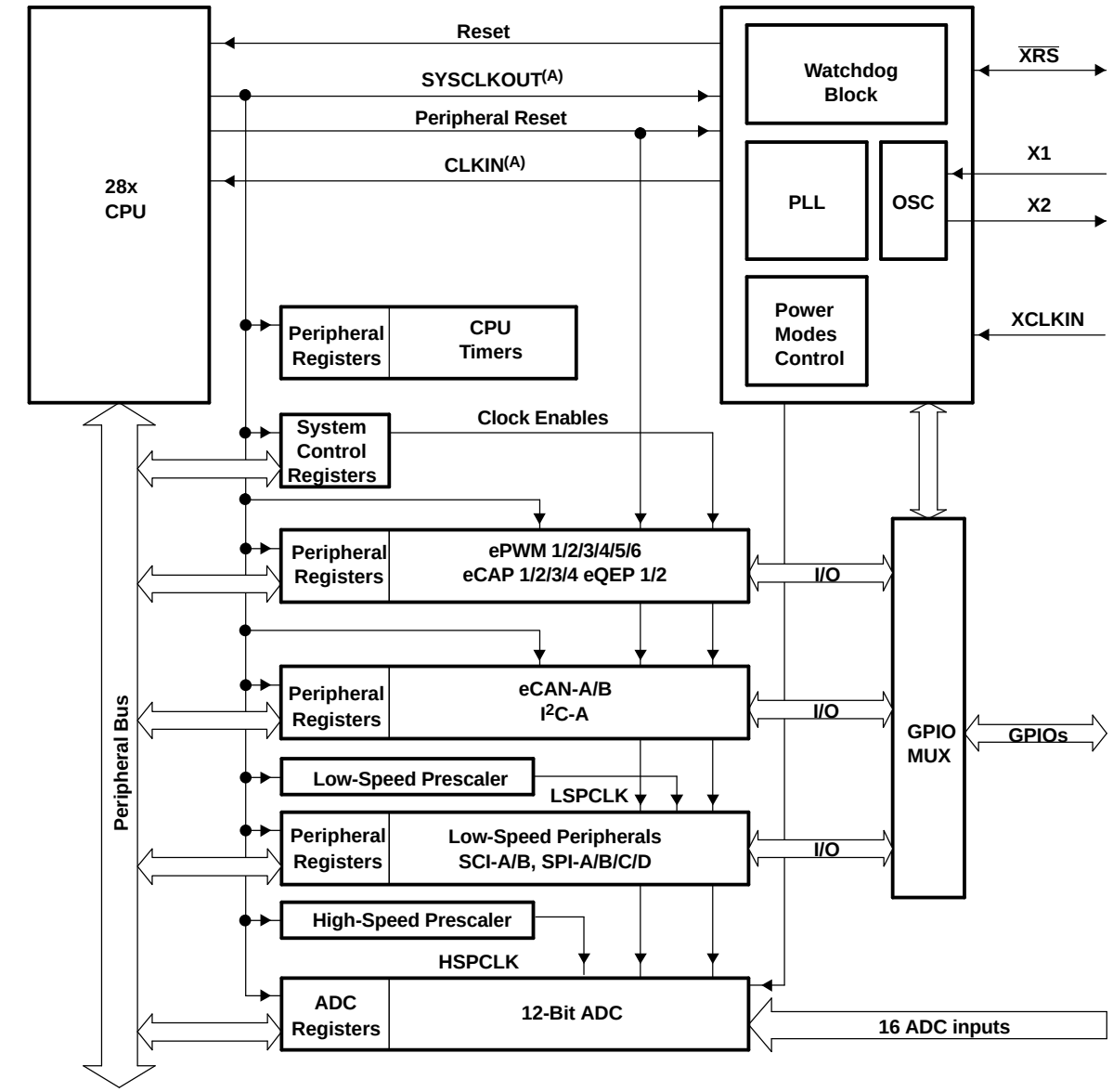
3.1 时钟和系统控制 38
3.2 OSC 和 PLL 块 45
3.3 低功率模式块 52
3.4 看门狗模块 55
3.5 32 位 CPU 定时器 0/1/2 57

3.1 时钟和系统控制

图 3-1 显示 各种时钟和复位域。

PLL、时钟、看门狗和低功率模式如表 3-1 所示。

图 3-1. 时钟和复位域



A CLKIN 是 CPU 的内部时钟。 从 CPU 传出时称作 SYSCLKOUT（即 CLKIN 与 SYSCLKOUT 频率相同）。

表 3-1. PLL、时钟、看门狗和低功率模式寄存器

名称	地址	大小 (x16)	说明(1)
XCLK	0x7010	1	XCLKOUT 引脚控制、X1 和 XCLKIN 状态寄存器
PLLSTS (2)	0x7011	1	PLL 状态寄存器
保留	0x7012 0x7019	8	
HISPCP	0x701A	1	HSPCLK 时钟的高速外设时钟预分频器寄存器
LOSPCP	0x701B	1	LSPCLK 时钟的低速外设时钟预分频器寄存器
PCLKCR0	0x701C	1	外设时钟控制寄存器 0
PCLKCR1	0x701D	1	外设时钟控制寄存器 1
LPMCR0	0x701E	1	低功率模式控制寄存器 0
保留	0x701F	1	
保留	0x7020	1	
PLLCR (2)	0x7021	1	PLL 控制寄存器
SCSR	0x7022	1	系统控制与状态寄存器
WDCNTR	0x7023	1	看门狗计数器寄存器
保留	0x7024	1	
WDKEY	0x7025	1	看门狗复位密钥寄存器
保留	0x7026 0x7028	3	
WDCR	0x7029	1	看门狗控制寄存器

(1) 此表中的所有寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 仅可由 $\overline{\text{XRS}}$ 信号或看门狗复位信号将 PLL 控制寄存器 (PLLCR) 和 PLL 状态寄存器 (PLLSTS) 复位成已知状态。 调试器或缺少时钟检测逻辑发出的复位信号无效。

PCLKCR0 和 PCLKCR1 寄存器启用/禁用到各种外设模块的时钟。 图 3-2 列出了 PCLKCR0 寄存器的位说明。

图 3-2. 外设时钟控制寄存器 0 (PCLKCR0)

15	14	13	12	11	10	9	8
ECANBENCLK	ECANAENCLK	保留		SCIBENCLK	SCIAENCLK	SPIBENCLK	SPIAENCLK
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SPIDENCLK	SPICENCLK	保留	I2CAENCLK	ADCENCLK	TBCLKSYNC	保留	
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

图 3-3. 外设时钟控制寄存器 1 (PCLKCR1)

15	14	13	12	11	10	9	8
EQEP2ENCLK	EQEP1ENCLK	保留		ECAP4ENCLK	ECAP3ENCLK	ECAP2ENCLK	ECAP1ENCLK
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
保留		EPWM6ENCLK	EPWM5ENCLK	EPWM4ENCLK	EPWM3ENCLK	EPWM2ENCLK	EPWM1ENCLK
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 3-2. 外设时钟控制寄存器 0 (PCLKCR0) 字段说明

位	字段	值	说明(1)
15	ECANBENCLK		ECAN-B 时钟启用。 此位在 2806 和 2801 器件上保留，写入将被忽略。
		0	不为 eCAN-B 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAN-B 模块计时。
14	ECANAENCLK		ECAN-A 时钟启用
		0	不为 eCAN-A 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAN-A 模块计时。
13-12	保留		保留
11	SCIBENCLK		SCI-B 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 SCI-B 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SCI-B 模块计时。
10	SCIAENCLK		SCI-A 时钟启用
		0	不为 SCI-A 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SCI-A 模块计时。
9	SPIBENCLK		SPI-B 时钟启用
		0	不为 SPI-B 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SPI-B 模块计时。
8	SPIAENCLK		SPI-A 时钟启用
		0	不为 SPI-A 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SPI-A 模块计时。
7	SPIDENCLK		SPI-D 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 SPI-D 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SPI-D 模块计时。
6	SPICENCLK		SPI-C 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 SPI-C 模块计时。 （默认值）(2)
		1	由低速时钟 (LSPCLK) 为 SPI-C 模块计时。
5	保留	0	保留
4	I2CAENCLK		I^2C 时钟启用
		0	不为 I^2C 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 I^2C 模块计时。
3	ADCENCLK		ADC 时钟启用
		0	不为 ADC 模块计时。 （默认值）(2)
		1	由高速时钟 (HSPCLK) 为 ADC 模块计时。
2	TBCLKSYNC		ePWM 模块时基时钟 (TBCLK) 同步：使用户可以将所有启用的 ePWM 模块与时基时钟 (TBCLK) 进行全局同步：
		0	停止所有启用的 ePWM 模块时钟。 （默认值）
		1	使用校准的 TBCLK 的第一个上升沿启动所有启用的 ePWM 模块时钟。 为了完美地同步 TBCLK，每个 ePWM 模块的 TBCTL 寄存器中的预分频器位必须进行相同设置。 启用 ePWM 时钟的正确过程如下： 1. 在 PCLKCR1 寄存器中启用 ePWM 模块时钟。 2. 将 TBCLKSYNC 设置为 0。 3. 配置预分频器值和 ePWM 模式。 4. 将 TBCLKSYNC 设置为 1。
1-0	保留		保留

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 如果未使用外设块，则可关闭到该外设的时钟以便使功耗降至最低。

表 3-3. 外设时钟控制寄存器 1 (PCLKCR1) 字段说明

位	字段	值	说明(1)
15	EQEP2ENCLK		eQEP2 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 eQEP2 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eQEP2 模块计时。
14	EQEP1ENCLK		eQEP1 时钟启用。
		0	不为 eQEP1 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eQEP1 模块计时。
13-12	保留		保留
11	ECAP4ENCLK		eCAP4 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 eCAP4 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAP4 模块计时。
10	ECAP3ENCLK		eCAP3 时钟启用。 此位在 2801 器件上保留，写入将被忽略。
		0	不为 eCAP3 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAP3 模块计时。
9	ECAP2ENCLK		eCAP2 时钟启用。
		0	不为 eCAP2 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAP2 模块计时。
8	ECAP1ENCLK		eCAP1 时钟启用。
		0	不为 eCAP1 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 eCAP1 模块计时。
7-6	保留		保留
5	EPWM6ENCLK		ePWM6 时钟启用。 (3)此位在 2801 器件上保留，写入将被忽略。
		0	不为 ePWM6 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM6 模块计时。
4	EPWM5ENCLK		ePWM5 时钟启用。 (3)此位在 2801 器件上保留，写入将被忽略。
		0	不为 ePWM5 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM5 模块计时。
3	EPWM4ENCLK		ePWM4 时钟启用。 (3)此位在 2801 器件上保留，写入将被忽略。
		0	不为 ePWM4 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM4 模块计时。
2	EPWM3ENCLK		ePWM3 时钟启用。 (3)
		0	不为 ePWM3 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM3 模块计时。
1	EPWM2ENCLK		ePWM2 时钟启用。 (3)
		0	不为 ePWM2 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM2 模块计时。
0	EPWM1ENCLK		ePWM1 时钟启用。 (3)
		0	不为 ePWM1 模块计时。 （默认值）(2)
		1	由系统时钟 (SYSCLKOUT) 为 ePWM1 模块计时。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 如果未使用外设块，则可关闭到该外设的时钟以便使功耗降至最低。
(3) 要在 ePWM 模块内启动 ePWM 时基时钟 (TBCLK)，必须还设置 PCLKCR0 中的 TBCLKSYNC 位。

系统控制和状态寄存器 (SCSR) 包含看门狗改写位和看门狗中断启用/禁用位。 图 3-4 描述了 SCSR 寄存器的位功能。

图 3-4. 系统控制与状态寄存器 (SCSR)

15							8
保留							
R-0							
7	6	5	4	3	2	1	0
保留					WDINTS	WDENINT	WDOVERRIDE
R-0					R-1	R/W-0	R/W1C-1

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

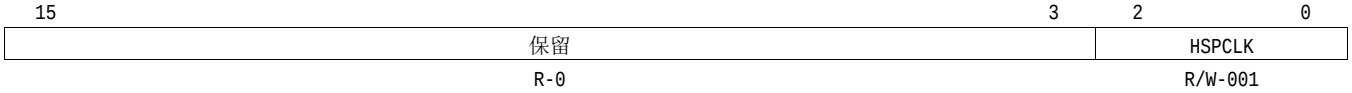
表 3-4. 系统控制与状态寄存器 (SCSR) 字段说明

位	字段	值	说明(1)
15-3	保留		
2	WDINTS		看门狗中断状态位。 WDINTS 反映来自看门狗时钟的 $\overline{\text{WDINT}}$ 信号的当前状态。 WDINTS 跟随 $\overline{\text{WDINT}}$ 状态两个 SYSCLKOUT 周期。 如果看门狗中断用于将器件从 IDLE 或 STANDBY 低功率模式唤醒，请使用 WDINTS 位以确保 $\overline{\text{WDINT}}$ 不会在试图返回 IDLE 或 STANDBY 模式之前活动。
		0	看门狗中断信号 ($\overline{\text{WDINT}}$) 活动。
		1	看门狗中断信号 ($\overline{\text{WDINT}}$) 不活动。
1	WDENINT		看门狗中断启用。
		0	启用看门狗复位 ($\overline{\text{WDRST}}$) 输出信号且禁用看门狗中断 ($\overline{\text{WDINT}}$) 输出信号。 这是复位 ($\overline{\text{XRS}}$) 时的默认状态。 当看门狗中断发生时，$\overline{\text{WDRST}}$ 信号将保持低电平 512 个 OSCCLK 周期。 如果在 $\overline{\text{WDINT}}$ 为低电平时清除 WDENINT 位，则将立即进行复位。 可以读取 WDINTS 位来确定 $\overline{\text{WDINT}}$ 信号的状态。
		1	禁用 $\overline{\text{WDRST}}$ 输出信号且启用 $\overline{\text{WDINT}}$ 输出信号。 当看门狗中断发生时，$\overline{\text{WDINT}}$ 信号将保持低电平 512 个 OSCCLK 周期。 如果看门狗中断用于将器件从 IDLE 或 STANDBY 低功率模式唤醒，请使用此位确保 $\overline{\text{WDINT}}$ 不会在试图返回 IDLE 或 STANDBY 模式之前活动。
0	WDOVERRIDE		看门狗改写
		0	写入 0 无效。 如果此位被清除，则它保持此状态直到出现复位。 用户可以读取此位的当前状态。
		1	可以更改看门狗控制 (WDCR) 寄存器的看门狗禁用 (WDDIS) 位的状态。 如果通过写入 1 清除了 WDOVERRIDE 位，则不能修改 WDDIS 位。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

高速外设和低速外设时钟预分频（ 和 ）寄存器分别用于配置高速和低速外设时钟。 请参阅图 3-5以了解 HISPCP 位布局，并参阅图 3-6以了解 LOSPCP 布局。

图 3-5. 高速外设时钟预分频器 (HISPCP) 寄存器



图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-5. 高速外设时钟预分频器 (HISPCP) 字段说明

位	字段	值	说明(1)
15-3	保留		保留
2-0	HSPCLK		这些位配置高速外设时钟 (HSPCLK) 相对于 SYSCLKOUT 的比率： 如果 HISPCP(2)≠ 0，则 HSPCLK = SYSCLKOUT/(HISPCP X 2) 如果 HISPCP = 0，则 HSPCLK = SYSCLKOUT
		000	高速时钟 = SYSCLKOUT/1
		001	高速时钟 = SYSCLKOUT/2（复位默认值）
		010	高速时钟 = SYSCLKOUT/4
		011	高速时钟 = SYSCLKOUT/6
		100	高速时钟 = SYSCLKOUT/8
		101	高速时钟 = SYSCLKOUT/10
		110	高速时钟 = SYSCLKOUT/12
		111	高速时钟 = SYSCLKOUT/14

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此等式中的 HISPCP 代表 HISPCP 寄存器中位 2:0 的值。

图 3-6. 低速外设时钟预分频器 (LOSPCP) 寄存器

15	3	2	0
保留		LSPCLK	
R-0		R/W-010	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-6. 低速外设时钟预分频器 (LOSPCP) 寄存器

位	字段	值	说明(1)
15-3	保留		保留
2-0	LSPCLK		这些位配置低速外设时钟 (LSPCLK) 相对于 SYSCLKOUT 的比率：
			如果 LOSPCP(2)≠ 0，则 LSPCLK = SYSCLKOUT/(LOSPCP X 2)
			如果 LOSPCP = 0，则 LSPCLK = SYSCLKOUT
		000	低速时钟 = SYSCLKOUT/1
		001	低速时钟 = SYSCLKOUT/2
		010	低速时钟 = SYSCLKOUT/4（复位默认值）
		011	低速时钟 = SYSCLKOUT/6
		100	低速时钟 = SYSCLKOUT/8
		101	低速时钟 = SYSCLKOUT/10
		110	低速时钟 = SYSCLKOUT/12
		111	低速时钟 = SYSCLKOUT/14

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 此等式中的 LOSPCP 代表 LOSPCP 寄存器中位 2:0 的值。

3.2 OSC 和 PLL 块

片上振荡器和锁相环路 (PLL) 块为器件提供时钟信号，并控制低功率模式 (LPM) 的进入。

3.2.1 基于 PLL 的时钟模块

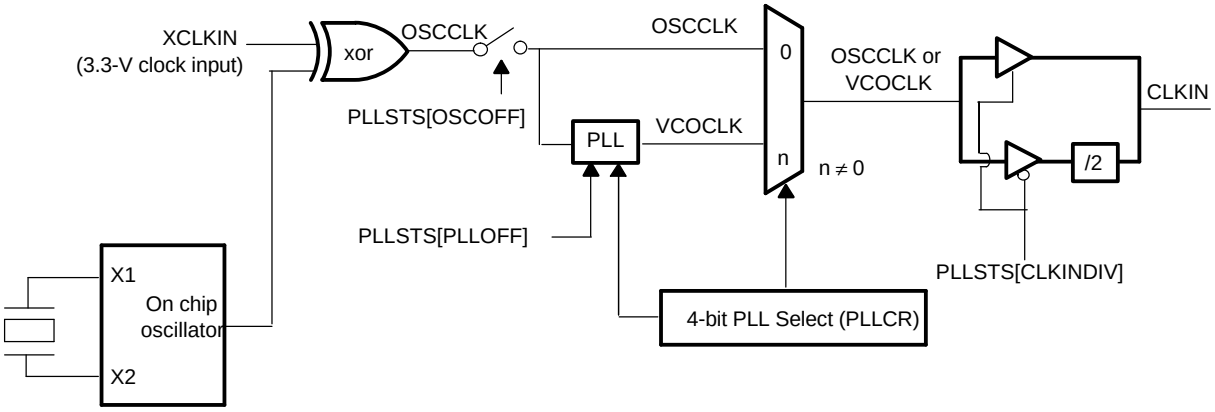
280x 器件具有一个片上 。 PLL 具有 4 位比率控制以选择不同的 CPU 时钟频率。

基于 PLL 的时钟模块提供两种操作模式：

- 晶振操作：
  此模式允许使用外部 为器件提供时基。 晶体连接到 X1/X2 引脚，XCLKIN 连接到低电平。
- 外部时钟源操作：
  此模式允许绕过内部振荡器。 器件时钟根据 引脚上的外部时钟源输入生成。 必须将 X1 连接到低电平并让 X2 悬空。 在这种情况下，外部振荡器时钟连接到 XCLKIN 引脚，允许使用 3.3V 时钟源。

图 3-7 显示 280x 上的 OSC 和 PLL 块。

图 3-7. OSC 和 PLL 块



The 允许使用 X1 和 X2 引脚将晶体连接到 280x 器件。 如果未使用晶体，则可以将外部振荡器直接连接到 XCLKIN 引脚，X2 引脚保留悬空，X1 连接到低电平。 请参阅 *TMS320F2808*、*TMS320F2806*、*TMS320F2801*、*UCD9501 数字信号处理器数据手册*（文献编号 SPRS230）。

表 3-7. 可能的 PLL 配置模式

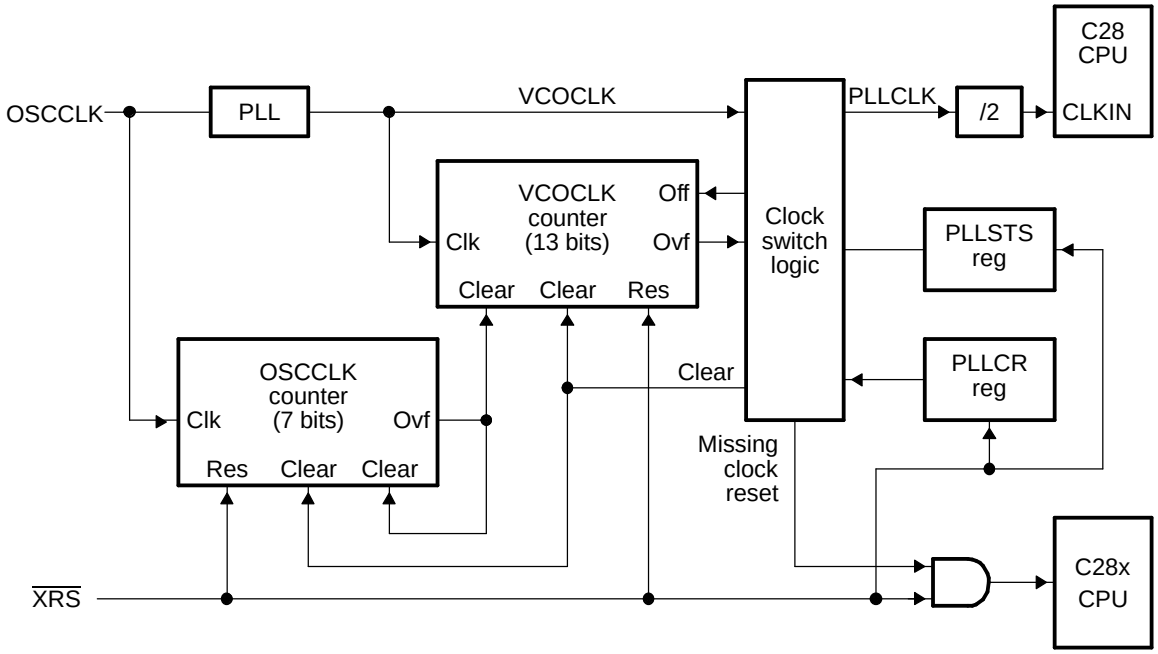
PLL 模式	注释	SYSCLKOUT
PLL 关闭	由在 PLLSTS 寄存器中设置 PLLOFF 位的用户调用。 在此模式中，PLL 块被禁用。 这对降低系统噪声和低功率操作非常有用。 必须先将 PLLCR 寄存器设置为 0x0000（PLL 旁路），再进入此模式。 CPU 时钟 (CLKIN) 等于 X1/X2 或 XCLKIN 上的输入时钟除以 2。	OSCCLK/2
PLL 旁路	PLL 旁路是上电或外部复位 ($\overline{XRS}$) 之后的默认 PLL 配置。 当 PLLCR 寄存器设置为 0x0000 时或在修改 PLLCR 寄存器之后 PLL 锁定至新频率时，选择此模式。 在此模式中，PLL 本身被旁路，但未关闭。	OSCCLK/2
启用 PLL	通过将非零值 n 写入 PLLCR 寄存器实现。 通过写入 PLLCR，器件将切换到 PLL 旁路方式直到 PLL 锁定。	OSCCLK*n/2

3.2.2 主振荡器失败检测

由于振动，到 DSP 的外部时钟源可能断开且无法为器件计时。 当 PLL 未被禁用时，主振荡器失败逻辑允许器件检测此情况并默认为本节中描述的已知状态。

使用了两个计数器来监控是否存在 OSCCLK 信号，如图 3-8。 来自 X1/X2 或 XCLKIN 输入的 OSCCLK 信号本身使第一个计数器递增。 当未关闭 PLL 时，PLL 时钟输出的 VCOCLK 使第二个计数器递增。 这些计数器配置为当 7 位 OSCCLK 计数器溢出时，就清除 13 位 VCOCLK 计数器。 在正常操作模式下，只要存在 OSCCLK，VCOCLK 计数器就永远不会溢出。

图 3-8. 振荡器失败检测逻辑图



如果缺少 OSCCLK 输入信号，则 PLL 将输出默认的“跛行模式”频率且 VCOCLK 计数器继续递增。 由于缺少 OSCCLK 信号，OSCCLK 计数器将不递增，因此不会定期清除 VCOCLK 计数器。 最终，VCOCLK 计数器溢出且器件将 CPU 的 CLKIN 输入切换到 PLL 的跛形模式输出频率 (VCOCLK/2)。 振荡器失败检测逻辑然后使 CPU、外设和其它器件逻辑复位。

除了使器件复位之外，缺少振荡器逻辑还会设置 PLLSTS[MCLKSTS] 寄存器位。 当 MCLKCSTS 位为 1 时，这表示缺少振荡器检测逻辑使部件复位且 CPU 现在以一半的跛形模式频率运行。 在复位之后应检查 PLLSTS 寄存器中的 MCLKSTS 状态位，以确定是否是由于发生缺少时钟情况而使器件复位，如果是，则采取适当的措施。可以通过将 1 写入到 PLLSTS[MCLKCLR] 位来使缺少时钟状态复位。 这将使缺少时钟检测电路和计数器复位。 如果在写入 MCLKCLR 位之后 OSCCLK 仍然丢失，则 VCOCLK 计数器再次溢出且将重复上述过程。

请记住下列预防措施和限制：

- **当更改 PLL 控制寄存器时请遵循正确的规范。**
  当修改 PLLCR 寄存器时，请始终遵循图 3-10 中概述的规范。
- **当器件正以跛形模式操作时，请不要写入 PLLCR 寄存器。**
  当写入 PLLCR 寄存器时，器件会将 CPU 的 CLKIN 输入切换到 OSCCLK/2。 当以跛形模式操作时，OSCCLK 可能不存在且到系统的时钟将停止。 在写入 PLLCR 寄存器之前，请始终检查 PLLSTS[MCLKSTS] 位是否为 0，如图 3-10。
- **没有外部时钟时，看门狗不工作。**
  当没有 OSCCLK 时，看门狗不工作且不能生成复位信号。 尚未添加特殊硬件来使看门狗在缺少 OSCCLK 时切换到跛形模式时钟。
- **跛形模式不会在上电模式时工作。**
  如果在上电时缺少 OSCCLK，PLL 不会生成跛形模式时钟。 仅在初始存在 OSCCLK 时，PLL 才会生成跛形模式时钟。
- **当器件正以跛形模式操作时，请不要进入 HALT 低功率模式。**
  如果在器件正以跛形模式操作时进入 HALT 模式，则系统可能挂起且可能无法退出 HALT 模式。 因此，在进入 HALT 模式之前，请始终检查 PLLSTS[MCLKSTS] 位是否为 0。

下面描述了在各种操作模式下缺少时钟检测逻辑的情形：

- **PLL 旁路模式**
  当 PLL 控制寄存器设置为 0x0000 时，PLL 被旁路且 OSCCLK/2 直接连接到 CPU 的输入时钟 CLKIN。 如果检测到缺少 OSCCLK，器件将自动切换到 PLL、设置缺少时钟检测状态位并生成缺少时钟复位信号。 器件现在将以 1/2 的 PLL 跛形模式频率运行。

- 启用 PLL 模式

  当 PLL 控制寄存器为非零值（PLLCR = n，其中 n ≠ 0x0000），则 PLL 被启用。 在此模式中，1/2 的 PLL 输出频率 (n*OSCCLK/2) 连接到 CPU 的 CLKIN。 如果检测到缺少 OSCCLK，将设置缺少时钟检测状态位且器件将生成缺少时钟复位信号。 器件现在将以 1/2 的 PLL 跛形模式频率运行。

- PLL 当前正在锁定

  在此模式中，已修改 PLLCR 且 PLL 已开始锁定。 当 PLL 正在锁定时，PLLSTS 寄存器中的 PLLLOCKS 位将为 0。 在锁定期间，到 CPU 的 CLKIN 连接到 OSCCLK/2。 如果缺少 OSCCLK，器件将使 CPU 的输入时钟切换到 1/2 的 PLL 跛形模式频率。 另外，将设置缺少时钟状态位且器件将生成缺少时钟复位信号。 器件现在将以 1/2 的 PLL 跛形模式频率运行。

- STANDBY 低功率模式

  在此模式中，到 CPU 的 CLKIN 停止。 如果检测到缺少输入时钟，将设置缺少时钟状态位且器件将生成缺少时钟复位信号。 如果出现此情况时 PLL 处于旁路模式，将自动为 CPU 发送 1/2 的 PLL 跛形频率。 器件现在将以 1/2 的 PLL 跛形模式频率运行。

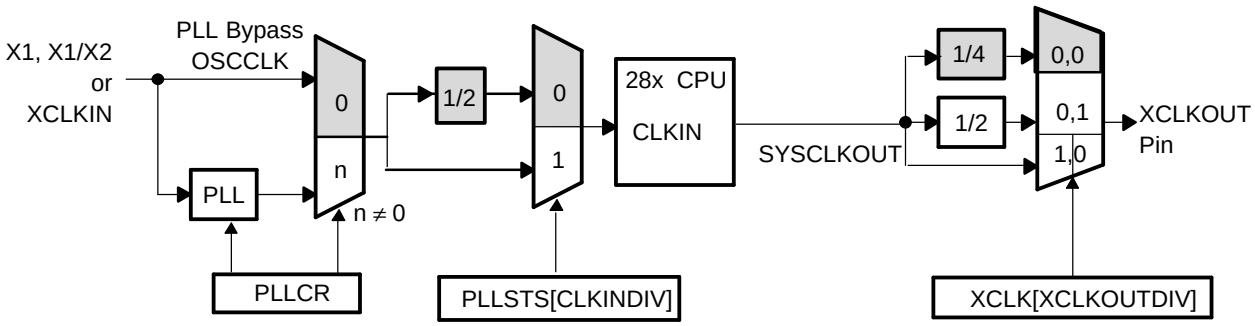
- HALT 低功率模式

  在 HALT 低功率模式中，到器件的所有时钟被关闭。 当器件退出 HALT 模式时，振荡器和 PLL 将上电。 仅在完成它们的上电之后，才会启用用于检测缺少输入时钟（VCOCLK 和 OSCCLK）的计数器。 如果 VCOCLK 计数器溢出，将设置缺少时钟检测状态位且器件将生成缺少时钟复位信号。 如果出现此情况时 PLL 处于旁路模式，将自动为 CPU 发送 1/2 的 PLL 跛形频率。 器件现在将以 1/2 的 PLL 跛形模式频率运行。

3.2.3 XCLKOUT 生成

XCLKOUT 信号直接来源于系统时钟 SYSCLKOUT，如图 3-9。 XCLKOUT 可以等于 SYSCLKOUT、它的一半或四分之一，具体情况由 XCLK 寄存器中的 XCLKOUTDIV 位确定。 默认情况下，在复位时，XCLKOUT = SYSCLKOUT/4 或 XCLKOUT = OSCCLK/8。

图 3-9. XCLKOUT 生成



Default at reset

当复位信号有效时，XCLKOUT 信号有效。 由于当复位信号为低电平时 XCLKOUT 应为 SYSCLKOUT/4，因此可以在调试期间监控此信号以检测器件是否被正确计时。 XCLKOUT 引脚上没有内部上拉或下拉电路。

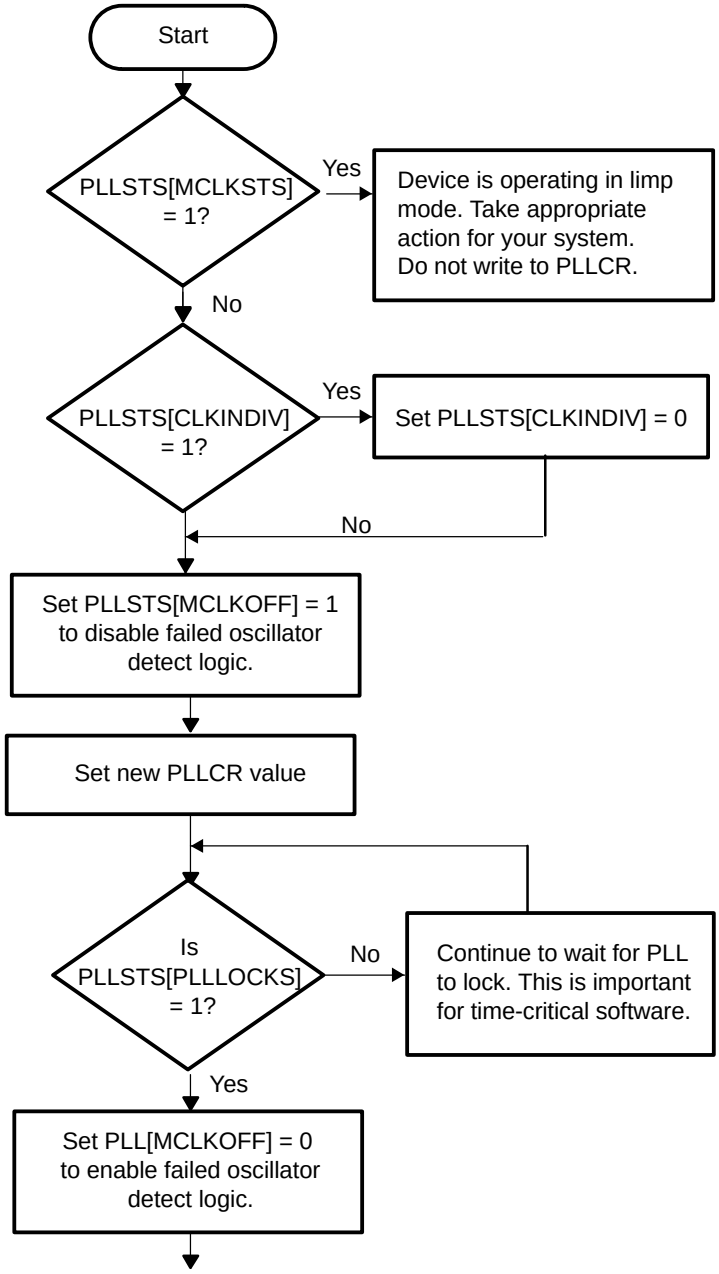
如果未使用 XCLKOUT，可以通过将 XCLK 寄存器中的 XCLKOUTDIV 位设置为 1,1 来关闭它。

3.2.4 PLL 控制 (PLLCR) 寄存器

PLLCR 寄存器用于更改器件的 PLL 乘法器。 当 CPU 写入 DIV 位时，PLL 逻辑将 CPU 时钟 (CLKIN) 切换到 OSCCLK/2。 一旦 PLL 稳定且已锁定在新指定的频率，PLL 就将 CLKIN 切换到新值，如表 3-8。 当发生这种情况时，将设置 PLLSTS 寄存器中的 PLLLOCKS 位，以指示 PLL 已完成锁定且器件现在以新频率运行。 用户软件可以监控 PLLLOCKS 位以确定 PLL 何时完成锁定。

无论何时写入 PLLCR 寄存器时，都请遵循图 3-10 中的规范。

图 3-10. PLLCR 更改规范流程图



3.2.5 PLL 控制、状态和 XCLKOUT 寄存器说明

PLLCR 寄存器中的 DIV 字段控制 PLL 是否被旁路且在它未被旁路时设置 PLL 时钟比率。 PLL 旁路是复位之后的默认模式。

如果 PLL 正以设置的 PLLSTS [MCLKSTS] 位所指示的跛形模式操作，请不要写入 DIV 字段。 请参阅图 3-10。

图 3-11. PLLCR 寄存器布局

15	4	3	0
保留		DIV	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-8. PLLCR 位说明

DIV 值	说明(1)
0000（PLL 旁路）	CLKIN = OSCCLK/2
0001	CLKIN = (OSCCLK*1)/2
0010	CLKIN = (OSCCLK*2)/2
0011	CLKIN = (OSCCLK*3)/2
0100	CLKIN = (OSCCLK*4)/2
0101	CLKIN = (OSCCLK*5)/2
0110	CLKIN = (OSCCLK*6)/2
0111	CLKIN = (OSCCLK*7)/2
1000	CLKIN = (OSCCLK*8)/2
1001	CLKIN = (OSCCLK*9)/2
1010	CLKIN = (OSCCLK*10)/2
1011 - 1111	保留

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 3-12. PLL 状态寄存器 (PLLSTS)

15							8
保留							
R-0							

7	6	5	4	3	2	1	0
保留	MCLKOFF	OSCOFF	MCLKCLR	MCLKSTS	PLLOFF	保留	PLLLOCKS
R-0	R/W-0	R/W-0	R=0/W-0	R-0	R/W-0	R/W-0	R-1

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-9. PLL 状态寄存器 (PLLSTS) 字段说明

位	字段	值	说明(1)(2)
15-7	保留		保留
6	MCLKOFF		缺少时钟检测关闭位
		0	启用主振荡器失败检测逻辑。 （默认值）
		1	禁用主振荡器失败检测逻辑。 不想受检测电路影响的用户可以使用此模式。 例如，外部时钟关闭。
5	OSCOFF		振荡器时钟关闭位
		0	来自 X1/X2 或 XCLKIN 的 OSCCLK 信号馈送至 PLL 块。 （默认值）
		1	来自 X1/X2 或 XCLKIN 的 OSCCLK 信号未馈送至 PLL 块。 此模式对于测试主振荡器失败检测逻辑非常有用。 此模式不关闭内部振荡器。
4	MCLKCLR		缺少时钟清除位
		0	写入 0 无效。 此位的读数始终为 0。
		1	强制缺少时钟检测电路清零和复位。 如果仍然缺少 OSCCLK，则检测电路将再次生成传送至系统的复位信号、设置缺少时钟状态位 (MCLKSTS) 且由 PLL 以跛形模式频率驱动 CPU。
3	MCLKSTS		缺少时钟状态位。 复位后检查此位的状态，以确定是否检测到缺少振荡器的情况。 在正常情况下，此位应为 0。 对此位的写入将被忽略。 通过写入 MCLKCLR 位或强制实施外部复位，可以清除此位。
		0	表示正常操作。 未检测到缺少时钟的情况。
		1	表示检测到缺少 OSCCLK。 主振荡器失败检测逻辑已重置此器件，并且 CPU 当前已由按跛形模式频率运行的 PLL 计时。
2	PLLOFF		PLL 关闭位 此位关闭 PLL. 有助于系统噪声测试。 此模式应只在 PLLCR 寄存器设置为 0x0000 时使用。
		0	PLL 打开（默认值）
		1	PLL 关闭 设置了 PLLOFF 位时，PLL 模块应保持断电。 器件必须处于 PLL 旁路模式 (PLLCR = 0x0000)，然后才能将 1 写入 PLLOFF。 当 PLL 关闭 (PLLOFF = 1) 时，不要为 PLLCR 写入非零值。 当 PLLOFF = 1 时，STANDBY 和 HALT 低功率模式将如预期正常工作。 在从 HALT 或 STANDBY 唤醒之后，PLL 模块将继续断电。
1	保留		此位保留且必须始终写入为 0。 不要将 1 写入此位。
0	PLLLOCKS		PLL 锁定状态位
		0	指示已写入 PLLCR 寄存器且 PLL 当前已锁定。 由 OSCCLK/2 为 CPU 计时，直至 PLL 锁定。
		1	指示 PLL 已锁定且现在已稳定。

(1) 只能由 $\overline{\text{XRS}}$ 信号或看门狗复位信号使此寄存器复位到它的默认状态。 缺少时钟或调试器复位信号不能使它复位。
(2) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 3-13. XCLKOUT 寄存器 (XCLK)

15	12	11	8
XCLKINCNT		X1CNT	
R/W-0		R/W-0	

7	5	4	3	2	1	0
保留		XCLKINDAT	X1DAT	XCLKOUTDAT	XCLKOUTDIV	
R-0		W-0	R-0	R-0	R/W-0	

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 3-10. XCLKOUT 寄存器 (XCLK) 字段说明

位	字段	值	说明(1)
15-12	XCLKINCNT		此位保留仅供德州仪器 (TI) 使用。
11-8	X1CNT		此位保留仅供德州仪器 (TI) 使用。
7-5	保留		
4	XCLKINDAT		此位保留仅供德州仪器 (TI) 使用。
3	X1DAT		此位保留仅供德州仪器 (TI) 使用。
2	XCLKOUTDAT		此位保留仅供德州仪器 (TI) 使用。
1	XCLKOUTDIV		XCLKOUT 分频比。 这两位选择 XCLKOUT 频率相对于 SYSCLKOUT 频率的比率。 比率为：
		00	XCLKOUT = SYSCLKOUT/4（默认值）
		01	XCLKOUT = SYSCLKOUT/2
		10	XCLKOUT = SYSCLKOUT
		11	XCLKOUT = 关闭（引脚处于高阻抗模式）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

3.2.6 外部参考振荡器时钟选项

TI 建议客户让谐振器/晶体供应商提供他们的器件与 DSP 芯片一起工作的信息。 振谐器/晶体供应商具有调谐振谐电路的设备和专业技术。 供应商也可以建议客户注意选择正确的振谐组件值以在整个操作范围内提供正确的启动和稳定性。

3.3 低功率模式块

The 280x 器件上的低功率模式与 240x 器件类似。 表 3-11 总结了各种模式。

各种低功率模式的操作如表 3-12。

有关进入和退出低功率模式的准确时序，请参阅 *TMS320F2808*、*TMS320F2806*、*TMS320F2801*、*UCD9501 数字信号处理器数据手册*（文献编号 SPRS230）。

表 3-11. 280x 低功率模式

模式	LPMCR0[1:0]	OSCCLK	CLKIN	SYSCLKOUT	退出(1)
IDLE	00	打开	打开	打开(2)	$\overline{\text{XRS}}$、 看门狗中断、 任何启用的中断、 XNMI
STANDBY	01	打开 （看门狗仍在运行）	关闭	关闭	$\overline{\text{XRS}}$、 看门狗中断、 GPIO 端口 A 信号、 调试器(3)
HALT	1X	关闭 （振荡器和 PLL 关闭、看门狗不工作）	关闭	关闭	$\overline{\text{XRS}}$、 GPIO 端口 A 信号、 调试器(3)

(1) “退出”列列出哪些信号或在哪些情况下会退出低功率模式。 此信号必须保持低电平足够长时间以便器件识别中断。 否则，不退出 IDLE 模式且器件回到指示的低功率模式。

(2) 28x 上的 IDLE 模式的行为与 24x/240x 的不同。 在 28x 上，CPU 的时钟输出 (SYSCLKOUT) 仍然工作，而在 24x/240x 上，该时钟关闭。

(3) 在 28x 上，即使到 CPU 的时钟 (CLKIN) 关闭，JTAG 端口仍然可以工作。

表 3-12. 低功率模式

模式	说明
IDLE 模式：	任何启用的中断或 NMI 可以导致退出此模式。 在此模式期间，LPM 块本身不执行任何任务。
STANDBY 模式：	如果 LPMCR0 寄存器中的 LPM 位设置为 01，则当执行 IDLE 指令时，器件进入 STANDBY 模式。 在 STANDBY 模式中，CPU 的时钟输入 (CLKIN) 被禁用，这会禁用从 SYSCLKOUT 派生的所有时钟。 振荡器、PLL 和看门狗仍将工作。在进入 STANDBY 模式之前，应执行以下任务： • 在 PIE 模块中启用 WAKEINT 中断。 此中断同时连接至看门狗和低功率模式模块中断。 • 如有需要，请在 GPIOLPMSEL 寄存器中指定其中一个 GPIO 端口 A 信号来唤醒器件。 GPIOLPMSEL 寄存器是 GPIO 模块的一部分。 除了所选的 GPIO 信号之外，如果在 LPMCRO 寄存器中启用了 $\overline{\text{XRS}}$ 输入和看门狗中断，则它们也可以将器件从 STANDBY 模式唤醒。 • 在 LPMCR0 寄存器中为将唤醒器件的信号选择输入鉴定。 当所选的外部信号变低时，它必须保持低电平如 LPMCR0 寄存器中的鉴定期所指定的若干 OSCCLK 周期。 如果在此期间采样到信号为高电平，则鉴定将重新启动。 在鉴定期结束时，PLL 启用 CPU 的 CLKIN，且在 PIE 块中锁定 WAKEINT 中断。 CPU 然后响应 WAKEINT 中断（如果已启用该中断）。
HALT 模式：	如果 LPMCR0 寄存器中的 LPM 位设置为 1x，则当执行 IDLE 指令时，器件进入 HALT 模式。 在 HALT 模式中，所有器件时钟以及 PLL 和振荡器关闭。 在进入 HALT 模式之前，应执行以下任务： • 在 PIE 模块中启用 WAKEINT 中断。 此中断同时连接至看门狗和低功率模式模块中断。 • 如有需要，请在 GPIOLPMSEL 寄存器中指定其中一个 GPIO 端口 A 信号来唤醒器件。 GPIOLPMSEL 寄存器是 GPIO 模块的一部分。 除了所选的 GPIO 信号之外，XRS 输入也会将器件从 STANDBY 模式唤醒。 **当器件以跛形模式 (PLLSTS[MCLKSTS] = 1) 操作时，不要进入 HALT 低功率模式。** 如果在器件以跛形模式操作时尝试进入 HALT 模式，则系统可能挂起，并且您可能无法退出 HALT 模式。 因此，在进入 HALT 模式之前，请始终检查 PLLSTS[MCLKSTS] 位是否为 0。 当所选的外部信号变低时，它被以异步方式馈送到 LPM 块。 振荡器打开并开始上电。 您必须保持该信号为低电平足够长时间以便振荡器完成上电。 当信号再次驱动回高电平时，此信号将异步释放 PLL，并且它将开始锁定。 一旦 PLL 已锁定，它会在 CPU 响应 WAKEINT 中断（如果已启用）时将 CLKIN 馈送到 CPU。

低功率模式由 （图 3-14）的图像所示。

图 3-14. 低功率模式控制 0 寄存器 (LPMCR0)

15	14	8	7	2	1	0
WDINTE	保留		QUALSTDBY		LPM	
R/W-0	R-0		R/W-1		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-13. 低功率模式控制寄存器 0 (LPMCR0) 字段说明

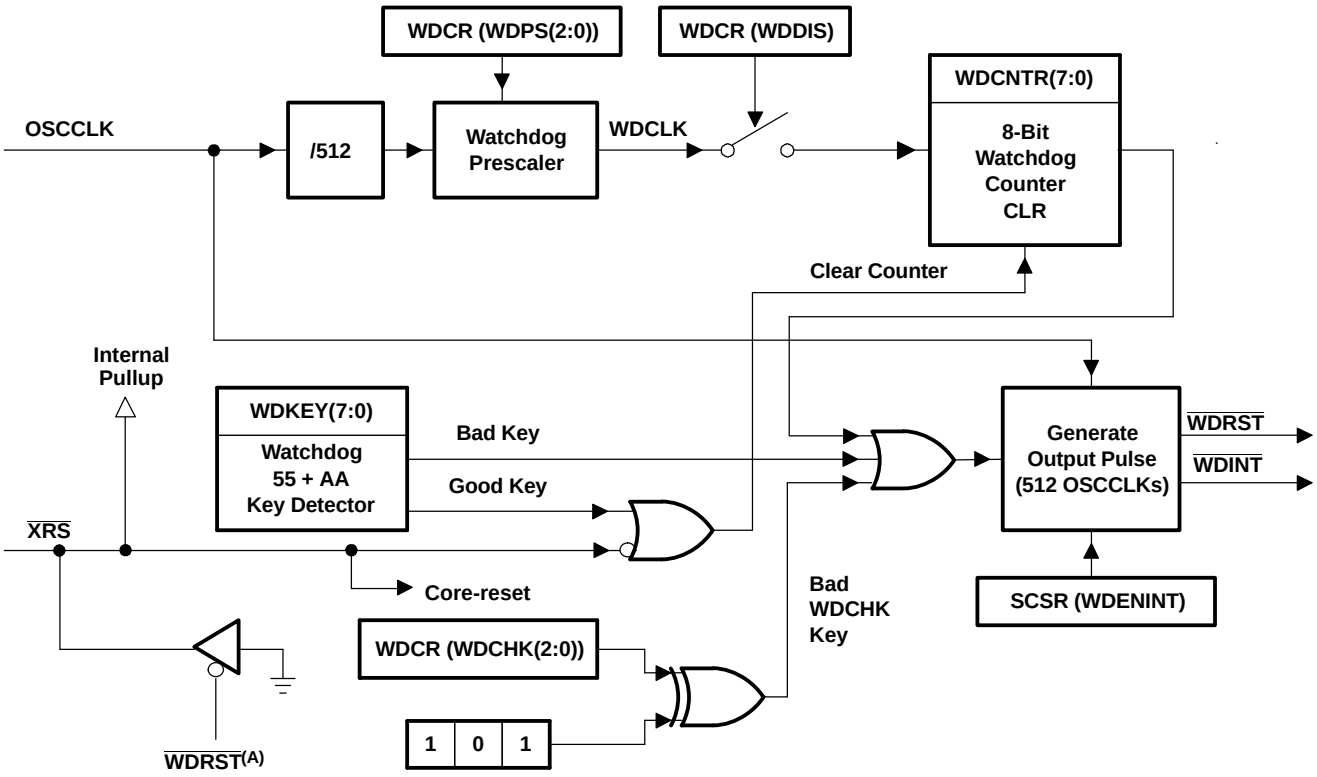
位	字段	值	说明(1)
15	WDINTE		看门狗中断启用
		0	不允许看门狗中断将器件从待机模式（STANDBY）中唤醒。 （默认值）
		1	允许看门狗将器件从 STANDBY 唤醒。 还必须在 SCSR 寄存器中启用看门狗中断。
14-8	保留		保留
7-2	QUALSTDBY		选择用于鉴定所选 GPIO 输入的 OSCCLK 时钟周期数，该输入将器件从 STANDBY 模式唤醒。 仅当处于 STANDBY 模式时，才使用此鉴定。 在 GPIOLPMSEL 寄存器中指定了可将器件从 STANDBY 唤醒的 GPIO 信号。
		000000	2 个 OSCCLK（默认值）
		000001	3 个 OSCCLK
		。 。 。	。 。 。
		111111	65 个 OSCCLK
1-0	LPM(2)		这些位设置器件的低功率模式。
		00	将低功率模式设置为 IDLE（默认值）
		01	将低功率模式设置为 STANDBY
		10	将低功率模式设置为 HALT 模式(3)
		11	将低功率模式设置为 HALT 模式(3)

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。
(2) 仅当执行 IDLE 指令时，低功率模式位 (LPM) 才生效。 因此，在执行 IDLE 指令之前，必须将 LPM 位设置为合适的模式。
(3) 如果在器件正以跛形模式操作时进入 HALT 模式，则系统可能挂起且可能无法退出 HALT 模式。 因此，在进入 HALT 模式之前，请始终检查 PLLSTS[MCLKSTS] 位是否为 0。

3.4 看门狗模块

280x 上的看门狗模块与 240x 和 281x 器件上使用的看门狗块类似。 每当 8 位的看门狗加计数器到达其最大值时，看门狗模块会生成一个 512 个振荡器时钟 (OSCCLK) 宽的输出脉冲。 要防止这种情况，用户可以禁用该计数器，或者必须通过编写软件定期将一个 0x55 + 0xAA 序列写入至看门狗密钥寄存器中，从而使看门狗计数器复位。 图 3-15 显示了看门狗模块内的各种功能块。

图 3-15. 看门狗模块



A 当发生看门狗复位时，WDRST 和 XRS 信号被驱动为低电平 512 个 OSCCLK 周期。 同样，如果启用了看门狗中断，则当发生中断时，WDINT 信号将被驱动为低电平 512 个 OSCCLK 周期。

可以在 SCSR 寄存器中将看门狗配置为在看门狗计数器达到其最大值时使器件复位 (WDRST) 或发出中断 (WDINT)。 每种情况的行为描述如下：

- 复位模式：

  如果看门狗配置为使器件复位，则当看门狗计数器达到其最大值时，WDRST 信号将把器件复位 (XRS) 引脚拉低 512 个 OSCCLK 周期。 如果在 WDINT 低电平有效时将看门狗重新配置为复位模式，则将立即使器件复位。 当使看门狗从中断模式切换到复位模式时，这非常重要。 在将看门狗重新配置为复位模式之前，可以读取 SCSR 寄存器中的 WDINTS 位以确定 WDINT 信号的当前状态。

- 中断模式：

  如果看门狗配置为触发中断，则将使 WDINT 信号驱动为低电平 512 个 OSCCLK 周期，从而采用 PIE 中的 WAKEINT 中断（如果已在 PIE 模块中启用该中断）。 在 WDINT 下降沿的边缘触发看门狗中断。 因此，如果在 WDINT 变为无效之前重新启用 WAKEINT 中断，则将不能立即获得另一个中断。 下一个 WAKEINT 中断将出现在下一次看门狗超时的时候。

  看门狗中断信号使看门狗可以用作从 IDLE 和 STANDBY 低功率模式的唤醒信号。 如果看门狗中断用于从 IDLE 或 STANDBY 低功率模式情况唤醒，则在试图返回 IDLE 或 STANDBY 模式之前请确保 WDINT 信号再次回到高电平。 当生成看门狗中断时，WDINT 信号将保持低电平 512 个 OSCCLK 周期。 可以通过读取 SCSR 寄存器中的看门狗中断状态位 (WDINTS) 来确定 WDINT 的当前状态。

  在 STANDBY 模式中，器件上的所有外设关闭。 继续工作的唯一外设是看门狗。 看门狗模块使振荡器时钟关闭。 WDINT 信号被馈送到 LPM 块以便它可以将器件从 STANDBY 唤醒（如已启用）。 有关详细信息，请参阅器件数据手册中的“低功率模式块”部分。

  在 IDLE 模式中，WDINT 信号可以生成到 CPU 的中断（PIE 中的 WAKEINT 中断）以使 CPU 退出 IDLE 模式。

在 HALT 模式中，不能使用此功能，因为振荡器（和 PLL）关闭，因此看门狗也关闭。

图 3-16. 看门狗计数器寄存器 (WDCNTR)

15	8	7	0
保留		WDCNTR	
R-0		R-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-14. 看门狗计数器寄存器 (WDCNTR) 字段说明

位	字段	说明
15-8	保留	保留
7-0	WDCNTR	这些位包含 WD 计数器的当前值。 此 8 位计数器按看门狗时钟 (WDCLK) 频率持续递增。 如果计数器溢出，则看门狗启动复位。 如果用有效组合写入 WDKEY 寄存器，则该计数器复位为零。 可以在 WDCR 寄存器中配置看门狗时钟频率。

图 3-17. 看门狗复位 Key 寄存器 (WDKEY)

15	8	7	0
保留		WDKEY	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-15. 看门狗复位 Key 寄存器 (WDKEY) 字段说明

位	字段	说明(1)
15-8	保留	保留
7-0	WDKEY	写入 0x55 并接着写入 0xAA 会导致清除 WDCNTR 位。 写入任何其它值导致立即生成看门狗复位信号。 读取会返回 WDCR 寄存器的值。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 3-18. 看门狗控制寄存器 (WDCR)

15				8
保留				

7	6	5	3	2	0
WDFLAG	WDDIS	WDCHK		WDPS	
R/W1C-0	R/W-0	R/W-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-16. 看门狗控制寄存器 (WDCR) 字段说明

位	字段	值	说明(1)
15-8	保留		保留
7	WDFLAG		看门狗复位状态标志位
		0	$\overline{XRS}$ 引脚或上电导致复位。 该位保持被锁定状态，直至写入 1 以清除该条件。 忽略 0 的写入。
		1	指示看门狗复位 ($\overline{WDRST}$) 生成了复位条件。 。
6	WDDIS		看门狗禁用。 在复位时，启用了看门狗模块。
		0	启用看门狗模块。 仅当 SCSR2 寄存器中的 WDOVERRIDE 位设置为 1 时，才可以修改 WDDIS。
		1	禁用看门狗模块
5-3	WDCHK		看门狗检查。 每当对此寄存器执行写入时，必须始终向这些位写入 1,0,1。 如果启用了看门狗，则写入任何其它值导致 CPU 立即复位。 从这三位读回的值始终为零 (0, 0, 0)。
2-0	WDPS		看门狗预分频。 这些位配置看门狗计数器时钟 (WDCLK) 相对于 OSCCLK/512 的比率：
		000	WDCLK = OSCCLK/512/1
		001	WDCLK = OSCCLK/512/1
		010	WDCLK = OSCCLK/512/2
		011	WDCLK = OSCCLK/512/4
		100	WDCLK = OSCCLK/512/8
		101	WDCLK = OSCCLK/512/16
		110	WDCLK = OSCCLK/512/32
		111	WDCLK = OSCCLK/512/64

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

当 $\overline{XRS}$ 线路为低电平时，WDFLAG 位被强制为低电平。 仅当检测到 $\overline{WDRST}$ 信号的上升沿（在同步和一个 8192 SYSCLKOUT 周期延迟之后）且 $\overline{XRS}$ 信号为高电平时，才设置 WDFLAG 位。 如果当 $\overline{WDRST}$ 变为高电平时 $\overline{XRS}$ 信号为低电平，则 WDFLAG 位保持为 0。 在典型应用中，$\overline{WDRST}$ 信号连接至 $\overline{XRS}$ 输入。 因此，要区分看门狗复位和外部器件复位，外部复位的持续时间必须比看门狗脉冲长。

3.4.1 仿真注意事项

在各种调试情况下，看门狗模块的行为如下：

CPU 暂停：	当 CPU 暂停时，看门狗时钟 (WDCLK) 暂停。
自由运行模式：	当 CPU 处于自由运行模式时，看门狗模块继续正常操作。
实时单步模式：	当 CPU 处于实时单步模式时，看门狗时钟 (WDCLK) 暂停。 即使在实时中断内，看门狗也保持暂停。
实时自由运行模式：	当 CPU 处于实时自由运行模式时，看门狗正常运行。

3.5 32 位 CPU 定时器 0/1/2

本节描述 280x 器件上的 3 个 32 位 定时器（图 3-19）(TIMER0/1/2)。

CPU 定时器 1 和 2 保留供 TI 软件（如 DSP-BIOS）使用。 可以在用户应用程序中使用 CPU 定时器 0。

在 280x 器件中，CPU 定时器中断信号（TINT0、TINT1、TINT2）的连接如图 3-20。

图 3-19. CPU 定时器

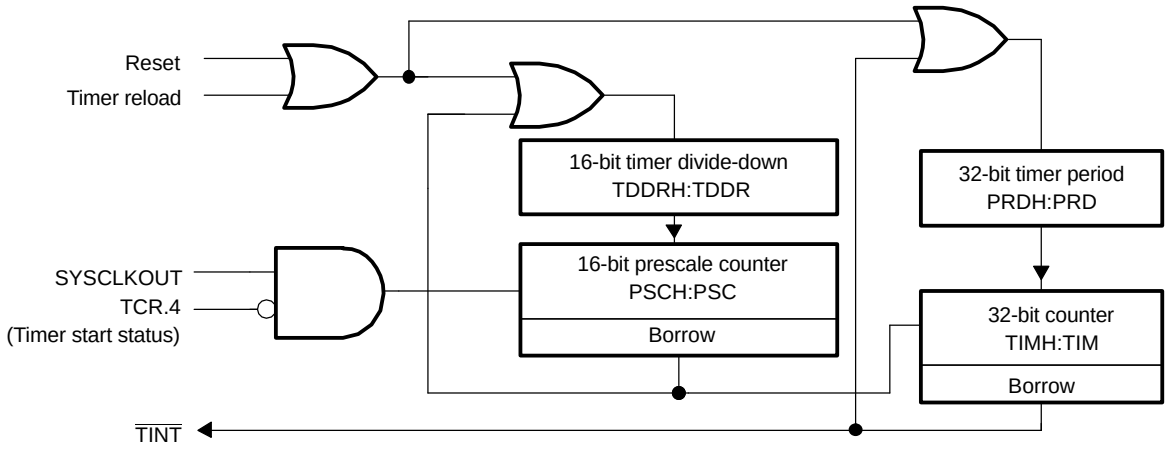
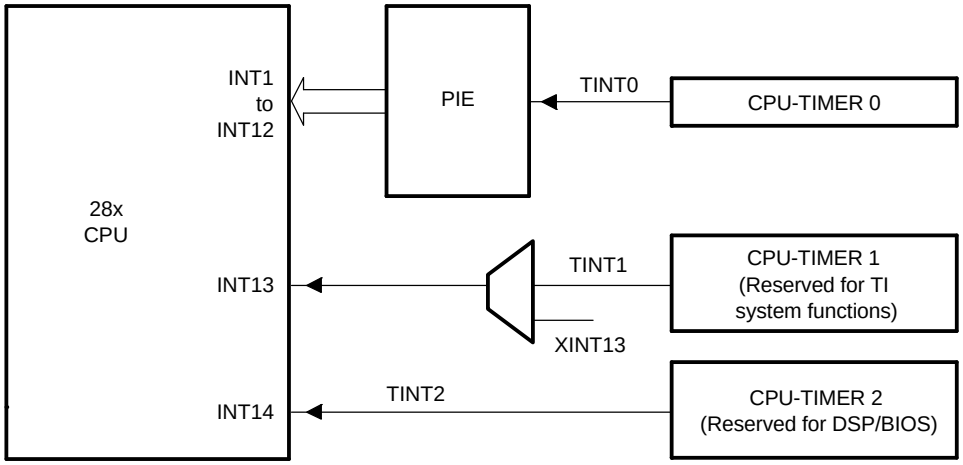


图 3-20. CPU 定时器中断信号和输出信号



A 定时器寄存器连接到 28x 处理器的存储器总线。

B 定时器的时序与处理器时钟的 SYSCLKOUT 同步。

CPU 定时器的通常操作如下：32 位计数器寄存器 TIMH:TIM 装入周期寄存器 PRDH:PRD 中的值。 计数器寄存器按 28x 的 SYSCLKOUT 频率递减。 当计数器到达 0 时，定时器中断输出信号生成中断脉冲。 表 3-17 中列出的寄存器用于配置定时器。

表 3-17. CPU 定时器 0、1、2 配置和控制寄存器

名称	地址	大小 (x16)	说明
TIMER0TIM	0x0C00	1	CPU 定时器 0，计数器寄存器
TIMER0TIMH	0x0C01	1	CPU 定时器 0，计数器寄存器高电平
TIMER0PRD	0x0C02	1	CPU 定时器 0，期间寄存器
TIMER0PRDH	0x0C03	1	CPU 定时器 0，期间寄存器高电平
TIMER0TCR	0x0C04	1	CPU 定时器 0，控制寄存器
保留	0x0C05	1	
TIMER0TPR	0x0C06	1	CPU 定时器 0，预分频寄存器
TIMER0TPRH	0x0C07	1	CPU 定时器 0，预分频寄存器高电平
TIMER1TIM	0x0C08	1	CPU 定时器 1，计数器寄存器
TIMER1TIMH	0x0C09	1	CPU 定时器 1，计数器寄存器高电平
TIMER1PRD	0x0C0A	1	CPU 定时器 1，期间寄存器

表 3-17. CPU 定时器 0、1、2 配置和控制寄存器(接上表)

名称	地址	大小 (x16)	说明
TIMER1PRDH	0x0C0B	1	CPU 定时器 1，期间寄存器高电平
TIMER1TCR	0x0C0C	1	CPU 定时器 1，控制寄存器
保留	0x0C0D	1	
TIMER1TPR	0x0C0E	1	CPU 定时器 1，预分频寄存器
TIMER1TPRH	0x0C0F	1	CPU 定时器 1，预分频寄存器高电平
TIMER2TIM	0x0C10	1	CPU 定时器 2，计数器寄存器
TIMER2TIMH	0x0C11	1	CPU 定时器 2，计数器寄存器高电平
TIMER2PRD	0x0C12	1	CPU 定时器 2，期间寄存器
TIMER2PRDH	0x0C13	1	CPU 定时器 2，期间寄存器高电平
TIMER2TCR	0x0C14	1	CPU 定时器 2，控制寄存器
保留	0x0C15	1	
TIMER2TPR	0x0C16	1	CPU 定时器 2，预分频寄存器
TIMER2TPRH	0x0C17	1	CPU 定时器 2，预分频寄存器高电平

图 3-21. TIMERxTIM 寄存器 (x = 1, 2, 3)

15	0
TIM	
R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-18. TIMERxTIM 寄存器字段说明

位	字段	说明
15-0	TIM	CPU 定时器计数器寄存器 (TIMH:TIM)：TIM 寄存器保持定时器的当前 32 位计数的低 16 位。 TIMH 寄存器保持定时器的当前 32 位计数的高 16 位。 TIMH:TIM 每隔 (TDDRH:TDDR+1) 个时钟周期减 1，其中 TDDRH:TDDR 是定时器预分频下来的值。 当 TIMH:TIM 递减到零时，将用 PRDH:PRD 寄存器中包含的周期值重载 TIMH:TIM 寄存器。 生成定时器中断 (TINT) 信号。

图 3-22. TIMERxTIMH 寄存器 (x = 1, 2, 3)

15	0
TIMH	
R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-19. TIMERxTIMH 寄存器字段说明

位	字段	说明
15-0	TIMH	请参阅 TIMERxTIM 的说明。

图 3-23. TIMERxPRD 寄存器 (x = 1, 2, 3)

15	0
PRD	
R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-20. TIMERxPRD 寄存器字段说明

位	字段	说明
15-0	PRD	CPU 定时器周期寄存器 (PRDH:PRD)：PRD 寄存器保持 32 位周期的低 16 位。 PRDH 寄存器保持 32 位周期的高 16 位。 当 TIMH:TIM 递减到零时，在下一个定时器输入时钟周期（预分频器的输出）开始时，将用 PRDH:PRD 寄存器中包含的周期值重载 TIMH:TIM 寄存器。 当设置了定时器控制寄存器 (TCR) 中的定时器重载位 (TRB) 时，也会将 PRDH:PRD 内容载入 TIMH:TIM。

图 3-24. TIMERxPRDH 寄存器 (x = 1, 2, 3)

15	0
PRDH	
R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-21. TIMERxPRDH 寄存器字段说明

位	字段	说明
15-0	PRDH	请参阅 TIMERxPRD 的说明。

图 3-25. TIMERxTCR 寄存器 (x = 1, 2, 3)

15	14	13	12	11	10	9	8
TIF	TIE	保留		FREE	SOFT	保留	
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0	

7	6	5	4	3	0
保留		TRB	TSS	保留	
R-0		R/W-0	R/W-0	R-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-22. TIMERxTCR 寄存器字段说明

位	字段	值	说明
15	TIF		CPU 定时器中断标志。
		0	CPU 定时器未递减至零。 忽略 0 的写入。
		1	当 CPU 定时器递减至零时，将设置此标志。 将 1 写入此位会清除该标志。
14	TIE		CPU 定时器中断启用。
		0	禁用 CPU 定时器中断。
		1	启用 CPU 定时器中断。 如果定时器递减至零，且设置了 TIE，则定时器发出它的中断请求。
13-12	保留		保留
11-10	FREE SOFT		CPU 定时器仿真模式：当在高级语言调试器中遇到断点时，这些位是确定定时器状态的特殊仿真位。 如果 FREE 位设置为 1，则遇到软件断点时，定时器继续运行（即，自由运行）。 在这种情况下，SOFT 为无关。 但是如果 FREE 为 0，则 SOFT 生效。 在这种情况下，如果 SOFT = 0，则定时器在下次 TIMH:TIM 递减时停机。 如果 SOFT 位为 1，则定时器在 TIMH:TIM 递减至零时停机。
		FREE SOFT	CPU 定时器仿真模式
		0 0	在 TIMH:TIM 下一次递减之后停止（硬停止）
		0 1	在 TIMH:TIM 递减至 0 之后停止（软停止）
		1 0	自由运行
		1 1	自由运行 在 SOFT STOP 模式中，定时器在关闭之前生成中断（因为到达 0 是中断产生的条件）。
9-6	保留		保留
5	TRB		CPU 定时器重载位。
		0	TRB 位始终读取为 0。 忽略 0 的写入。
		1	当将 1 写入 TRB 时，TIMH:TIM 载入 PRDH:PRD 中的值，且预分频器计数器 (PSCH:PSC) 载入定时器分出寄存器 (TDDRH:TDDR) 中的值。
4	TSS		CPU 定时器停止状态位。 TSS 是停止或启动 CPU 定时器的 1 位标志。
		0	读数 0 指示 CPU 定时器正在运行。 要启动或重新启动 CPU 定时器，请将 TSS 设置为 0。 在复位时，TSS 被清零且 CPU 定时器立即启动。
		1	读数 1 指示 CPU 定时器已停止。 要停止 CPU 定时器，请将 TSS 设置为 1。
3-0	保留		保留

图 3-26. TIMERxTPR 寄存器 (x = 1, 2, 3)

15	8	7	0
PSC		TDDR	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-23. TIMERxTPR 寄存器字段说明

位	字段	说明
15-8	PSC	CPU 定时器预分频计数器 这些位保持定时器的当前预分频计数。 对于 PSCH:PSC 值大于 0 的每个定时器时钟源周期，PSCH:PSC 减 1。 PSCH:PSC 到达 0 之后一个定时器时钟（定时器预分频器的输出）周期，PSCH:PSC 载入 TDDRH:TDDR 的内容，且定时器计数器寄存器 (TIMH:TIM) 减 1。 每当软件设置定时器重载位 (TRB) 时，也会重载 PSCH:PSC。 可以通过读取寄存器检查 PSCH:PSC，但不能直接设置它。 必须从定时器分出寄存器 (TDDRH:TDDR) 获取它的值。 在复位时，PSCH:PSC 设置为 0。
7-0	TDDR	CPU 定时器分出。 每隔 (TDDRH:TDDR + 1) 个定时器时钟源周期，定时器计数器寄存器 (TIMH:TIM) 减 1。 在复位时，TDDRH:TDDR 位清零。 要将总的定时器计数增加一个整数因子，请将此因子减 1 之后写入 TDDRH:TDDR 位。 当预分频器计数器 (PSCH:PSC) 值为 0 时，一个定时器时钟源周期之后，TDDRH:TDDR 的内容重载 PSCH:PSC，且 TIMH:TIM 减 1。 每当软件设置定时器重载位 (TRB) 时，TDDRH:TDDR 也会重载 PSCH:PSC。

图 3-27. TIMERxTPRH 寄存器 (x = 1, 2, 3)

15	8	7	0
PSCH		TDDRH	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 3-24. TIMERxTPRH 寄存器字段说明

位	字段	说明
15-8	PSCH	请参阅 TIMERxTPR 的说明。
7-0	TDDRH	请参阅 TIMERxTPR 的说明。

TEXAS INSTRUMENTS

通用输入/输出 (GPIO)

GPIO MUX 寄存器用于选择 280x 器件上的共享引脚操作。 这些引脚按其通用 I/O 名称命名，即 GPIO0 - GPIO34。 可以分别选择这些引脚作为数字 I/O 操作，或将它们分别连接到三个外设 I/O 信号之一（通过 GPAMUX1/2 和 GPBMUX1 寄存器）。 如果选择为数字 I/O 模式，则提供寄存器来配置引脚方向（通过 GPADIR 和 GPBDIR 寄存器）。 还可以鉴定输入信号以除去不需要的噪音（通过 GPAQSEL1/2、GPBQSEL1/2、GPACTRL 和 GPBCTRL 寄存器）。

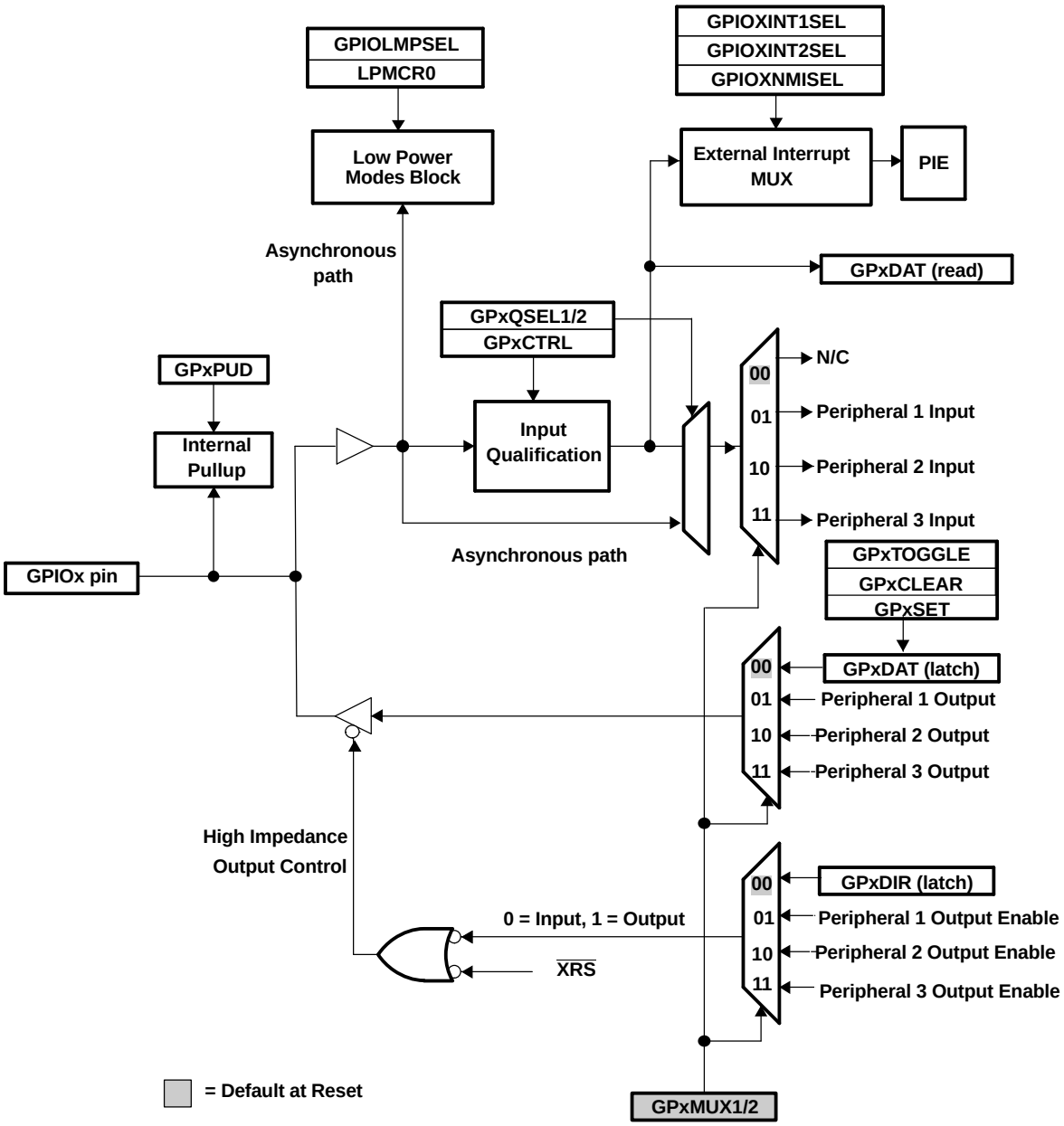
主题 页

4.1 GPIO 模块概述 64
4.2 配置概述 64
4.3 数字通用 I/O 控制 66
4.4 GPIO 和外设多路复用 68
4.5 寄存器位定义 76

4.1 GPIO 模块概述

在 280x 器件上，除了单个引脚位 IO 能力之外，每个 GPIO 引脚上还可以多路复用多达三个独立的外设信号。 280x 器件上有两个 32 位的 IO 端口。 端口 A 包含 GPIO0-GPIO31，端口 B 包含 GPIO32-GPIO34。 端口 B 上的其余 IO 当前保留供将来扩展使用。 图 4-1 显示了 GPIO 模块的基本操作模式。

图 4-1. 操作模式



A x 代表端口，即 A 或 B。 例如，GPxDIR 指 GPADIR 或 GPBDIR 寄存器，具体取决于所选的特定 GPIO 引脚。

B 在相同的存储器位置存取 GPxDAT 锁定/读取。

4.2 配置概述

引脚功能指定、输入鉴定和外部中断（XINT1、XINT2 和 XNMI）源全由 GPIO 配置控制寄存器控制。 另外，可以指定引脚将器件从 HALT 和 STANDBY 低功率模式唤醒并可以启用/禁用内部上拉电阻。 表 4-1 与表 4-2 列出了用于配置 GPIO 引脚以匹配系统要求的寄存器。

表 4-1. GPIO 控制寄存器

名称(1)	地址	大小 (x16)	寄存器说明
GPACTRL	0x6F80	2	GPIO A 控制寄存器 (GPIO0-GPIO31)
GPAQSEL1	0x6F82	2	GPIO A 限定器选择 1 寄存器 (GPIO0-GPIO15)
GPAQSEL2	0x6F84	2	GPIO A 限定器选择 2 寄存器 (GPIO16-GPIO31)
GPAMUX1	0x6F86	2	GPIO A MUX 1 寄存器 (GPIO0-GPIO15)
GPAMUX2	0x6F88	2	GPIO A MUX 2 寄存器 (GPIO16-GPIO31)
GPADIR	0x6F8A	2	GPIO A 方向寄存器 (GPIO0-GPIO31)
GPAPUD	0x6F8C	2	GPIO A 上拉禁用寄存器 (GPIO0-GPIO31)
保留	0x6F8E -0x6F8F	2	
GPBCTRL	0x6F90	2	GPIO B 控制寄存器 (GPIO32-GPIO34)
GPBQSEL1	0x6F92	2	GPIO B 限定器选择 1 寄存器 (GPIO32-GPIO34)
GPBQSEL2	0x6F94	2	保留
GPBMUX1	0x6F96	2	GPIO B MUX 1 寄存器 (GPIO32-GPIO34)
GPBMUX2	0x6F98	2	保留
GPBDIR	0x6F9A	2	GPIO B 方向寄存器 (GPIO32-GPIO34)
GPBPUD	0x6F9C	2	GPIO B 上拉禁用寄存器 (GPIO32-GPIO34)
保留	0x6F9E -0x6FB0	34	

(1) 此表中的所有寄存器受 EALLOW 保护。 请参阅第 5.2 部分

表 4-2. GPIO 中断和低功率模式选择寄存器

名称(1)	地址	大小 (x16)	寄存器说明
GPIOXINT1SEL	0x6FE0	1	XINT1 源选择寄存器 (GPIO0-GPIO31)
GPIOXINT2SEL	0x6FE1	1	XINT2 源选择寄存器 (GPIO0-GPIO31)
GPIOXNMISEL	0x6FE2	1	XNMI 源选择寄存器 (GPIO0-GPIO31)
保留	0x6FE3 - 0x6FE7	5	
GPIOLPMSEL	0x6FE8	1	LPM 唤醒源选择寄存器 (GPIO0-GPIO31)

(1) 此表中的所有寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

要计划如何配置 GPIO 模块，请考虑以下步骤：

1. 计划器件外引脚：

   通过引脚多路复用方案，280x 器件在给 35 个 GPIO (GPIO0-GPIO34) 引脚指定功能时提供了灵活性。 在开始之前，请查看可用于每个引脚的外设选项，并为您的特定系统计划外引脚。 引脚将用作通用输入输出 (GP I/O) 还是用作三个可用外设功能之一？ 了解此信息将帮助确定如何进一步配置引脚。

   表 4-5、表 4-6 和表 4-7 分别显示 2808、2806 和 2801 器件的 MUX 选项。 表 4-9 按外设对 MUX 表进行分类。 此表可用于快速标识可指定为特定外设功能的 GPIO 引脚。

2. 启用或禁用内部上拉电阻：

要启用或禁用内部上拉电阻，请写入 GPIO 上拉禁用（GPAPUD 和 GPBPUD）寄存器中相应的位。 对于可充当 ePWM 输出引脚 (GPIO0-GPIO11) 的引脚，默认情况下禁用了内部上拉电阻。 所有其它 GPIO 引脚在默认情况下启用了上拉电阻。

3. 选择输入鉴定：

   如果引脚将用作输入，请指定所需的输入鉴定（如果存在）。 在 GPACTRL、GPBCTRL、GPAQSEL1/2 和 GPBQSEL1 寄存器中指定输入鉴定。 默认情况下，所有输入信号只与 SYSCLKOUT 同步。

4. 选择引脚功能：

   配置 GPAMUX1/2 和 GPBMUX1 寄存器以使引脚为 GP I/O 或三个可用外设功能之一。 默认情况下，所有 GPIO 引脚在复位时配置为通用输入引脚。

5. 对于数字通用 I/O，请选择引脚的方向：

   如果引脚配置为 GP I/O，请在 GPADIR 和 GPBDIR 寄存器中将引脚的方向指定为输入或输出。 默认情况下，所有 GP I/O 引脚为输入。 要将引脚从输入更改为输出，首先通过为 GPACLEAR、GPBCLEAR、GPASET、GPBSET、GPATOGGLE 或 GPBTOGGLE 寄存器写入合适的值来给输出锁定载入

要驱动的值。 一旦载入了输出锁定，就可以通过 GPADIR 和 GPBDIR 寄存器将引脚方向从输入更改为输出。 在复位时将清除所有引脚的输出锁定。

6. 选择低功率模式唤醒源：

指定哪些引脚（如果有的话）能够将器件 HALT 和 STANDBY 低功率模式唤醒。 在 GPIOLPMSEL 寄存器中指定这些引脚。

7. 选择外部中断源：

指定 XINT1、XINT2 和 XNMI 中断的源。 对于每个中断，可以指定其中一个端口 A 信号作为源。 通过在 GPIOXINT1SEL、GPIOXINT2SEL 和 GPIOXNMISEL 寄存器中指定源完成此操作。 可以在 XINT1CR、XINT2CR 和 XNMICR 寄存器中配置中断的极性，如第 6.6 部分。

4.3 数字通用 I/O 控制

对于配置为 GP I/O 的引脚，可以通过使用下列寄存器更改引脚上的值：

表 4-3. GPIO 数据寄存器

名称	地址	大小 (x16)	寄存器说明
GPADAT	0x6FC0	2	GPIO A 数据寄存器 (GPIO0-GPIO31)
GPASET	0x6FC2	2	GPIO A 设置寄存器 (GPIO0-GPIO31)
GPACLEAR	0x6FC4	2	GPIO A 清除寄存器 (GPIO0-GPIO31)
GPATOGGLE	0x6FC6	2	GPIO A 转换寄存器 (GPIO0-GPIO31)
GPBDAT	0x6FC8	2	GPIO B 数据寄存器 (GPIO32-GPIO34)
GPBSET	0x6FCA	2	GPIO B 设置寄存器 (GPIO32-GPIO34)
GPBCLEAR	0x6FCC	2	GPIO B 清除寄存器 (GPIO32-GPIO34)
GPBTOGGLE	0x6FCE	2	GPIO B 转换寄存器 (GPIO32-GPIO34)
保留	0x70FC - 0x70FF	4	

- GPADAT、GPBDAT 寄存器：

  每个 I/O 端口具有一个数据寄存器。 数据寄存器中的每一位对应一个 GPIO 引脚。 不管引脚如何配置（GPIO 或外设功能），数据寄存器中的相应位在鉴定后都会反映引脚的当前状态。 写入 GPADAT 或 GPBDAT 寄存器会清除或设置相应的输出锁定，且如果引脚启用为通用输出（GPIO 输出），该引脚还将驱动为低电平或高电平。 如果引脚未配置为 GPIO 输出，则将锁定该值但不会驱动引脚。 仅当引脚稍后配置为 GPIO 输出，才会将锁定的值驱动至引脚。

  当使用 GPxDAT 寄存器更改输出引脚的电平时，请务必小心不要错误地更改另一引脚的电平。 例如，如果您打算通过使用读取 - 修改 - 写入指令写入 GPADAT 寄存器第 0 位来更改 GPIOA0 的输出锁定电平。 如果另一个 I/O 端口 A 信号在该指令的读取和写入阶段之间更改了电平，则可能出现问题。 也可以更改输出锁定的状态。 另外可以通过使用 GPxSET、GPxCLEAR 和 GPxTOGGLE 寄存器载入输出锁定来避免这种情况。

- GPASET、GPBSET 寄存器：

  设置寄存器用于将指定的 GPIO 引脚驱动为高电平，而不会扰乱其它引脚。 每个 I/O 端口具有一个设置寄存器且每位对应一个 GPIO 引脚。 设置寄存器始终读回 0。 如果相应的引脚配置为输出，则将 1 写入设置寄存器中的该位会将输出锁定设置为高电平且将相应的引脚驱动为高电平。 如果引脚未配置为 GPIO 输出，则将锁定该值但不会驱动引脚。 仅当引脚稍后配置为 GPIO 输出，才会将锁定的值驱动至引脚。 向设置寄存器中的任何位写入 0 无效。

- GPACLEAR、GPBCLEAR 寄存器：

  清除寄存器用于将指定的 GPIO 引脚驱动为低电平，而不会扰乱其它引脚。 每个 I/O 端口具有一个 清除寄存器。 清除寄存器始终读回 0。 如果相应的引脚配置为通用输出，则将 1 写入清除寄存器中的相应位将清除输出锁定且会将引脚驱动为低电平。 如果引脚未配置为 GPIO 输出，则将锁定该值但不会驱动引脚。 仅当引脚稍后配置为 GPIO 输出，才会将锁定的值驱动至引脚。 向清除寄存器中的任何位写入 0 无效。

- GPATOGGLE、GPBTOGGLE 寄存器：

  切换寄存器用于将指定的 GPIO 引脚驱动为相反的电平，而不会扰乱其它引脚。 每个 I/O 端口具有一个寄存器。 切换寄存器始终读回 0。 如果相应的引脚配置为输出，则将 1 写入切换寄存器中的该位会将输出锁定翻转且向相反的方向拉动相应的引脚。 即，如果输出引脚已驱动为低电平，则将 1 写入切换寄存器中的相应位会将该引脚拉高。 同样，如果输出引脚已为高电平，则将 1 写入切换寄存器中的相应位会将该引脚拉低。 如果引脚未配置为 GPIO 输出，则将锁定该值但不会驱动引脚。 仅当引脚稍后配置为

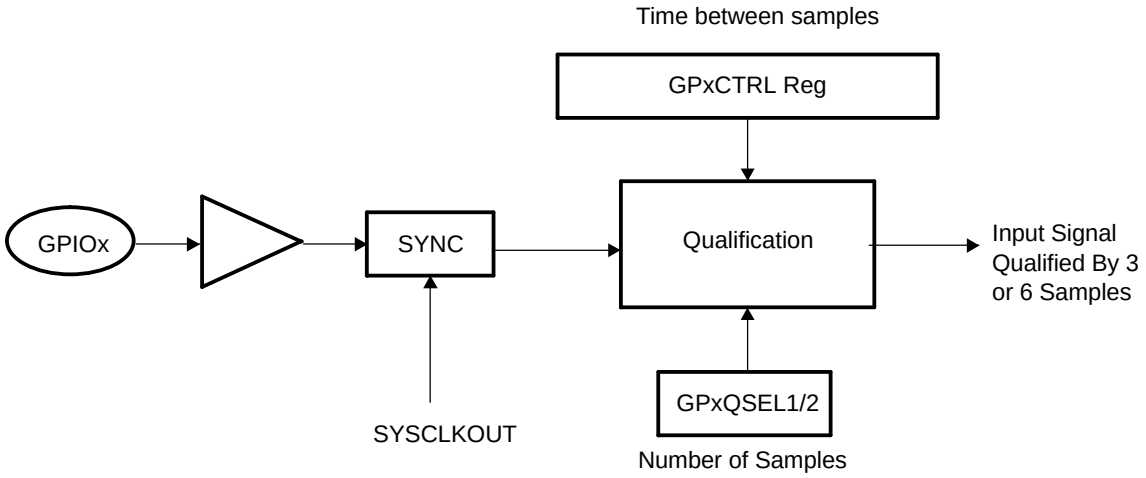
GPIO 输出，才会将锁定的值驱动至引脚。 向转换寄存器中的任何位写入 0 无效。

4.3.1 输入鉴定

输入鉴定在 280x 器件上非常灵活。 可以通过配置 GPAQSEL1/2 和 GPBQSEL1 寄存器选择每个 GPIO 引脚的输入鉴定类型，方法为以下三种选项之一：

- 只与 SYSCLKOUT 同步：
  在复位时，这是所有 GPIO 引脚的默认方式。 在此模式中，输入信号只与输入系统时钟 (SYSCLKOUT) 同步。 不对该信号执行任何进一步的鉴定。
- 使用采样窗口的鉴定：
  在此模式中，输入信号首先与系统时钟 (SYSCLKOUT) 同步，再按指定数目的周期鉴定，然后才允许输入更改。 图 4-2 与图 4-3 显示如何执行输入鉴定以消除不需要的噪声。

图 4-2. 输入鉴定

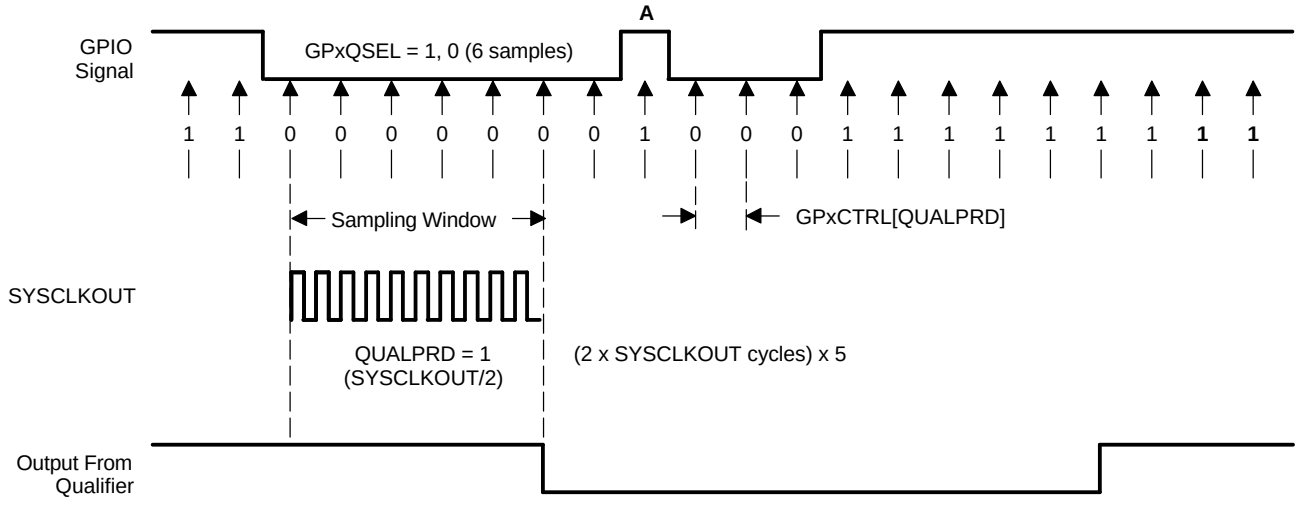


要鉴定信号，使输入与 SYSCLKOUT 同步，然后再定期采样。 采样周期由 GPACTRL 和 GPBCTRL 寄存器中的 QUALPRD 位指定，且可以按 8 个信号一组进行配置。 这指定采样之间相对于 SYSCLKOUT 的时间量。

采样窗口（或信号采样次数）为 3 倍采样或 6 倍采样宽，在 GPAQSEL1/2 和 GPBQSEL1 寄存器中指定。 当 3 个或 6 个连续周期相同时，表示已更改输入。 因为输入信号为异步，所以在采样窗口开始之前，与 SYSCLKOUT 同步可能有多达一个 SYSCLKOUT 周期的延迟。

对于图 4-3 中显示的示例，GPxQSEL1/2 寄存器中的采样窗口已配置为 6 采样鉴定。 采样窗口的宽度为 (2 X SYSCLKOUT) X 5 周期。 在图 4-3 中，输入限定器会忽略短时脉冲波形干扰 (A)。

图 4-3. 输入限定器时钟周期



GPxCTRL 寄存器中的 QUALPRD 位字段指定 0x00 到 0xFF 的采样周期。 当 QUALPRD = 0,0 时，输入鉴定周期等于 SYSCLKOUT。 对于任何其它值 n，将以 2*n 个 SYSCLKOUT 周期的速率采样鉴定。 即，将每隔 2n

个 SYSCLKOUT 周期对 GPIO 引脚采样一次。 对于要检查更改的输入限定器，输入应如 GPxQSEL1/2 寄存器中所指定的那样稳定 3 次或 6 次采样。

- 无同步（异步输入）：
  此模式用于不需要输入同步的外设或外设本身执行同步的情况。 示例包括通信端口 SCI、SPI、eCAN 和 I^2C。 另外，让 ePWM 跳匣区域 ($\overline{TZ1}$-$\overline{TZ6}$) 信号独立于 SYSCLKOUT 工作成为可能。 如果引脚用作通用数字输入引脚，则异步选项无效。 如果引脚配置为 GPIO 输入且在 GPxQSEL1/2 中选择了异步选项，则鉴定默认为与 SYSCLKOUT 同步。

4.4 GPIO 和外设多路复用

280x 器件可以在通用输入/输出 (GPIO) 端口的每个引脚上多路复用多达三个不同的外设功能。 这使您可以获取和选择将对特定应用效果最佳的外设混合方案。

表 4-5、表 4-6 和表 4-6 概述 2808、2806 和 2801 器件的可能 MUX 组合，按 GPIO 引脚分类。 第二列指示器件上的引脚的 I/O 名称。 由于 I/O 名称是唯一的，所以它是标识特定引脚的最佳方法。 因此，本节中的寄存器说明仅指特定引脚的 GPIO 名称。 第一列中指示了控制每个引脚的选择的 MUX 寄存器和特定位。

例如，通过写入 GPAMUX[15:14] 控制 GPIO7 引脚的 MUX。 通过写入这些位，该引脚配置为 GPIO7 或三个外设功能之一。 2808 上的 GPIO7 引脚可以配置如下：

GPAMUX1[15:14] 位设置	所选的引脚功能
如果 GPAMUX1[15:14] = 0,0	引脚配置为 GPIO7
如果 GPAMUX1[15:14] = 0,1	引脚配置为 EPWM4B (O)
如果 GPAMUX1[15:14] = 1,0	引脚配置为 SPISTED (I/O)
如果 GPAMUX1[15:14] = 1,1	引脚配置为 ECAP2 (I/O)

280x 系统中的所有器件具有相同的多路复用方案。 唯一的差异是如果外设在特定器件上不可用，则 MUX 选择将在该器件上保留且不应使用。 如果选择未映射到外设的保留 GPIO MUX 配置，则将根据引脚和 GPxMUX1/2 寄存器设置出现以下两种结果之一：

- 引脚将处于输入模式且不会被驱动。 引脚的电平将由该引脚上的内部上拉电阻的状态确定。 在器件 MUX 表（表 4-5、表 4-6 和表 4-7 ）中这指示为 “ 保留， 引脚未驱动 ”。
- 引脚将处于输出模式且被驱动为低电平。 在器件 MUX 表中，这指示为 “ 保留， 引脚被驱动为低电平 ”。

某些外设可以通过 MUX 寄存器指定至多个引脚。 例如，CAP1 功能可以根据如下所示的各个系统要求指定至 GPIO5 或 GPIO24 引脚：

指定至 CAP1 的引脚		MUX 配置
选择 1	GPIO5	GPAMUX[11:10] = 1,1
或选择 2	GPIO24	GPAMUX2[17:16] = 0,1

如果没有引脚配置为外设的输入，或多个引脚配置为同一外设的输入，则该外设的输入将默认为 0 或 1，如表 4-4。 例如，如果 ECAP1 指定给 GPIO5 和 GPIO24，则 eCAP1 外设的输入将默认为高电平状态，如 表 4-4 中所示，且该输入将不会连接到 GPIO5 或 GPIO24。

表 4-4. 外设输入的默认状态

外设输入	说明	默认输入 (1)
$\overline{TZ1}$-$\overline{TZ6}$	跳匣区域 1-6	1
EPWMSYNCI	ePWM 同步输入	0
ECAP1-ECAP4	eCAP1-4 输入	1
EQEP1A, EQEP2A	eQEP 输入	1
EQEP1I, EQEP2I	eQEP 索引	1
EQEP1S, EQEP2S	eQEP 选通	1
SPICLKA - SPICLKD	SPI-A - SPI-D	1
SPISTEA - SPISTED	启用 SPI-A - SPI-D 发送	0
SPISIMOA - SPISIMOD	SPI-A - SPI-D 从属输入、主输出	1
SPISOMIA - SPISOMID	SPI-A - SPI-D 从属输出、主输入	1
SCIRXDA - SCIRXDB	SCI-A - SCI-B 接收	1
CANRXA- CANRXB	eCAN-A - eCAN-B 接收	1
SDAA	I2C 数据	1
SCLA1	I2C 时钟	1

(1) 如果已在 GPxMUX1/2 寄存器中给外设功能指定多个引脚或未指定任何引脚，则将给外设输入指定此值。

表 4-5、表 4-6 和表 4-7 分别显示 2808、2806 和 2801 器件的 MUX 选项。 表 4-9 按外设对 MUX 表进行分类。 此表可用于快速标识可指定为特定外设功能的 GPIO 引脚。

表 4-5. 2808 GPIO MUX 表

GPAMUX1/2[1] 寄存器位	复位时的默认值 主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
		GPAMUX1		
1-0	GPIO0	EPWM1A (O)	保留。 未驱动引脚。[2]	保留。 未驱动引脚。
3-2	GPIO1	EPWM1B (O)	SPISIMOD (I/O)	保留。 未驱动引脚。
5-4	GPIO2	EPWM2A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
7-6	GPIO3	EPWM2B (O)	SPISOMID (I/O)	保留。 未驱动引脚。
9-8	GPIO4	EPWM3A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
11-10	GPIO5	EPWM3B (O)	SPICLKD (I/O)	ECAP1 (I/O)
13-12	GPIO6	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	SPISTED (I/O)	ECAP2 (I/O)
17-16	GPIO8	EPWM5A (O)	CANTXB (O)	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	EPWM5B (O)	SCITXB (O)	ECAP3 (I/O)
21-20	GPIO10	EPWM6A (O)	CANRXB (I)	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	EPWM6B (O)	SCIRXB (I)	ECAP4 (I/O)
25-24	GPIO12	$\overline{\text{TZ1}}$ (I)	CANTXB (O)	SPISIMOB (I/O)
27-26	GPIO13	$\overline{\text{TZ2}}$ (I)	CANRXB (I)	SPISOMIB (I/O)
29-28	GPIO14	$\overline{\text{TZ3}}$ (I)	SCITXB (O)	SPICLKB (I/O)
31-30	GPIO15	$\overline{\text{TZ4}}$ (I)	SCIRXB (I)	SPISTEB (I/O)
		GPAMUX2		
1-0	GPIO16	SPISIMOA (I/O)	CANTXB (O)	$\overline{\text{TZ5}}$ (I)
3-2	GPIO17	SPISOMIA (I/O)	CANRXB (I)	$\overline{\text{TZ6}}$ (I)
5-4	GPIO18	SPICLKA (I/O)	SCITXB (O)	保留。 未驱动引脚。
7-6	GPIO19	SPISTEA (I/O)	SCIRXB (I)	保留。 未驱动引脚。
9-8	GPIO20	EQEP1A (I)	SPISIMOC (I/O)	CANTXB (O)
11-10	GPIO21	EQEP1B (I)	SPISOMIC (I/O)	CANRXB (I)
13-12	GPIO22	EQEP1S (I/O)	SPICLKC (I/O)	SCITXB (O)
15-14	GPIO23	EQEP1I (I/O)	SPISTEC (I/O)	SCIRXB (I)
17-16	GPIO24	ECAP1 (I/O)	EQEP2A (I)	SPISIMOB (I/O)
19-18	GPIO25	ECAP2 (I/O)	EQEP2B (I)	SPISOMIB (I/O)
21-20	GPIO26	ECAP3 (I/O)	EQEP2I (I/O)	SPICLKB (I/O)
23-22	GPIO27	ECAP4 (I/O)	EQEP2S (I/O)	SPISTEB (I/O)
25-24	GPIO28	SCIRXDA (I)	保留。 未驱动引脚。	$\overline{\text{TZ5}}$ (I)
27-26	GPIO29	SCITXDA (O)	保留。 未驱动引脚。	$\overline{\text{TZ6}}$ (I)
29-28	GPIO30	CANRXA (I)	保留。 未驱动引脚。	保留。 未驱动引脚。
31-30	GPIO31	CANTXA (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
		GPBMUX1		
1-0	GPIO32	SDAA (I/OC)	EPWMSYNCI (I)	$\overline{\text{ADCSOCAO}}$ (O)
3-2	GPIO33	SCLA (I/OC)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
5-4	GPIO34	保留。 未驱动引脚。	保留。 未驱动引脚。	保留。 未驱动引脚。

(1) GPxMUX1/2 指引脚的合适的 MUX 寄存器：如 GPAMUX1、GPAMUX2 或 GPBMUX1。
(2) 短语 “保留。 未驱动引脚”意思是没有给此 GPxMUX1/2 寄存器设置指定外设。 如果选择了它，则引脚将成为输入且不会被驱动。这是用于未来扩展的保留配置。

表 4-6. 2806 GPIO MUX 表

GPxMUX1/2[1] 寄存器位	复位时的默认值 主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
		GPAMUX1		
1-0	GPIO0	EPWM1A (O)	保留。 未驱动引脚。[2]	保留。 未驱动引脚。
3-2	GPIO1	EPWM1B (O)	SPISIMOD (I/O)	保留。 未驱动引脚。
5-4	GPIO2	EPWM2A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
7-6	GPIO3	EPWM2B (O)	SPISOMID (I/O)	保留。 未驱动引脚。
9-8	GPIO4	EPWM3A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
11-10	GPIO5	EPWM3B (O)	SPICLKD (I/O)	ECAP1 (I/O)
13-12	GPIO6	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	SPISTED (I/O)	ECAP2 (I/O)
17-16	GPIO8	EPWM5A (O)	保留。 未驱动引脚。	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	EPWM5B (O)	SCITXB (O)	ECAP3 (I/O)
21-20	GPIO10	EPWM6A (O)	保留。 未驱动引脚。	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	EPWM6B (O)	SCIRXB (I)	ECAP4 (I/O)
25-24	GPIO12	$\overline{\text{TZ1}}$ (I)	保留。 未驱动引脚。	SPISIMOB (I/O)
27-26	GPIO13	$\overline{\text{TZ2}}$ (I)	保留。 未驱动引脚。	SPISOMIB (I/O)
29-28	GPIO14	$\overline{\text{TZ3}}$ (I)	SCITXB (O)	SPICLKB (I/O)
31-30	GPIO15	$\overline{\text{TZ4}}$ (I)	SCIRXB (I)	SPISTEB (I/O)
		GPAMUX2		
1-0	GPIO16	SPISIMOA (I/O)	保留。 未驱动引脚。	$\overline{\text{TZ5}}$ (I)
3-2	GPIO17	SPISOMIA (I/O)	保留。 未驱动引脚。	$\overline{\text{TZ6}}$ (I)
5-4	GPIO18	SPICLKA (I/O)	SCITXB (O)	保留。 未驱动引脚。
7-6	GPIO19	SPISTEA (I/O)	SCIRXB (I)	保留。 未驱动引脚。
9-8	GPIO20	EQEP1A (I)	SPISIMOC (I/O)	保留。 未驱动引脚。
11-10	GPIO21	EQEP1B (I)	SPISOMIC (I/O)	保留。 未驱动引脚。
13-12	GPIO22	EQEP1S (I/O)	SPICLKC (I/O)	SCITXB (O)
15-14	GPIO23	EQEP1I (I/O)	SPISTEC (I/O)	SCIRXB (I)
17-16	GPIO24	ECAP1 (I/O)	EQEP2A (I)	SPISIMOB (I/O)
19-18	GPIO25	ECAP2 (I/O)	EQEP2B (I)	SPISOMIB (I/O)
21-20	GPIO26	ECAP3 (I/O)	EQEP2I (I/O)	SPICLKB (I/O)
23-22	GPIO27	ECAP4 (I/O)	EQEP2S (I/O)	SPISTEB (I/O)
25-24	GPIO28	SCIRXDA (I)	保留。 未驱动引脚。	$\overline{\text{TZ5}}$ (I)
27-26	GPIO29	SCITXDA (O)	保留。 未驱动引脚。	$\overline{\text{TZ6}}$ (I)
29-28	GPIO30	CANRXA (I)	保留。 未驱动引脚。	保留。 未驱动引脚。
31-30	GPIO31	CANTXA (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
		GPBMUX1		
1-0	GPIO32	SDAA (I/OC)	EPWMSYNCI (I)	$\overline{\text{ADCSOCAO}}$ (O)
3-2	GPIO33	SCLA (I/OC)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
5-4	GPIO34	保留。 未驱动引脚。	保留。 未驱动引脚。	保留。 未驱动引脚。

(1) GPxMUX1/2 指引脚的合适的 MUX 寄存器：如 GPAMUX1、GPAMUX2 或 GPBMUX1。
(2) 短语 “ 保留。 未驱动引脚 ” 意思是没有给此 GPxMUX1/2 寄存器设置指定外设。 如果选择了它，则引脚将成为输入且不会被驱动。这是用于未来扩展的保留配置。

表 4-7. 2801 GPIO MUX 表

GPxMUX1/2[1] 寄存器位	复位时的默认值 主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
		GPAMUX1		
1-0	GPIO0	EPWM1A (O)	保留。 未驱动引脚。[2]	保留。 未驱动引脚。
3-2	GPIO1	EPWM1B (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
5-4	GPIO2	EPWM2A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
7-6	GPIO3	EPWM2B (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
9-8	GPIO4	EPWM3A (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
11-10	GPIO5	EPWM3B (O)	保留。 未驱动引脚。	ECAP1 (I/O)
13-12	GPIO6	保留。 引脚被驱动为低电平。[3]	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	保留。 引脚被驱动为低电平。	保留。 未驱动引脚。	ECAP2 (I/O)
17-16	GPIO8	保留。 引脚被驱动为低电平。	保留。 未驱动引脚。	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	保留。 引脚被驱动为低电平。	保留。 未驱动引脚。	保留
21-20	GPIO10	保留。 引脚被驱动为低电平。	保留。 未驱动引脚。	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	保留。 引脚被驱动为低电平。	保留。 未驱动引脚。	保留。 未驱动引脚。
25-24	GPIO12	$\overline{\text{TZ1}}$ (I)	保留。 未驱动引脚。	SPISIMOB (I/O)
27-26	GPIO13	$\overline{\text{TZ2}}$ (I)	保留。 未驱动引脚。	SPISOMIB (I/O)
29-28	GPIO14	$\overline{\text{TZ3}}$ (I)	保留。 未驱动引脚。	SPICLKB (I/O)
31-30	GPIO15	$\overline{\text{TZ4}}$ (I)	保留。 未驱动引脚。	SPISTEB (I/O)
		GPAMUX2		
1-0	GPIO16	SPISIMOA (I/O)	保留。 未驱动引脚。	$\overline{\text{TZ5}}$ (I)
3-2	GPIO17	SPISOMIA (I/O)	保留。 未驱动引脚。	$\overline{\text{TZ6}}$ (I)
5-4	GPIO18	SPICLKA (I/O)	保留。 未驱动引脚。	保留。 未驱动引脚。
7-6	GPIO19	SPISTEA (I/O)	保留。 未驱动引脚。	保留。 未驱动引脚。
9-8	GPIO20	EQEP1A (I)	保留。 未驱动引脚。	保留。 未驱动引脚。
11-10	GPIO21	EQEP1B (I)	保留。 未驱动引脚。	保留。 未驱动引脚。
13-12	GPIO22	EQEP1S (I/O)	保留。 未驱动引脚。	保留。 未驱动引脚。
15-14	GPIO23	EQEP1I (I/O)	保留。 未驱动引脚。	保留。 未驱动引脚。
17-16	GPIO24	ECAP1 (I/O)	保留。 未驱动引脚。	SPISIMOB (I/O)
19-18	GPIO25	ECAP2 (I/O)	保留。 未驱动引脚。	SPISOMIB (I/O)
21-20	GPIO26	保留	保留。 未驱动引脚。	SPICLKB (I/O)
23-22	GPIO27	保留	保留。 未驱动引脚。	SPISTEB (I/O)
25-24	GPIO28	SCIRXDA (I)	保留。 未驱动引脚。	$\overline{\text{TZ5}}$ (I)
27-26	GPIO29	SCITXDA (O)	保留。 未驱动引脚。	$\overline{\text{TZ6}}$ (I)
29-28	GPIO30	CANRXA (I)	保留。 未驱动引脚。	保留。 未驱动引脚。
31-30	GPIO31	CANTXA (O)	保留。 未驱动引脚。	保留。 未驱动引脚。
		GPBMUX1		
1-0	GPIO32	SDAA (I/OC)	EPWMSYNCI (I)	$\overline{\text{ADCSOCAO}}$ (O)
3-2	GPIO33	SCLA (I/OC)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
5-4	GPIO34	保留。 未驱动引脚。	保留。 未驱动引脚。	保留。 未驱动引脚。

(1) GPxMUX1/2 指引脚的合适的 MUX 寄存器；如 GPAMUX1、GPAMUX2 或 GPBMUX1。
(2) 短语 “保留。 未驱动引脚” 意思是没有给此 GPxMUX1/2 寄存器设置指定外设。 如果选择了它，则引脚将成为输入且不会被驱动。 这是用于未来扩展的保留配置。
(3) 短语 “保留。 将引脚驱动为低电平。” 意思是没有给此 GPxMUX1/2 寄存器设置指定外设。 如果选择了它，则引脚将成为输出且不会被驱动为低电平。 这是用于未来扩展的保留配置。

表 4-8 是按外设功能而不是按 GPIO 功能分类的 GPxMUX1/2 寄存器表。 此表显示 2808 器件上可用的外设的超集。 某些显示的外设在其它器件上将不可用且应视为保留。 在表 4-5、表 4-6 和表 4-7 。 此表可用于快速标识可指定为特定外设功能的 GPIO 引脚。 建议根据特定器件的数据手册交叉检查最终选择。

表 4-8. 外设与 GPIO 交叉参考

主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
	转换 A 的 ADC 外部启动		
GPIO8	EPWM5A	CANTXB	**$\overline{\text{ADCSOCAO}}$**
GPIO32	SDAA	EPWMSYNCI	**$\overline{\text{ADCSOCAO}}$**
	转换 B 的 ADC 外部启动		
GPIO10	EPWM6A	CANRXB	**$\overline{\text{ADCSOCBO}}$**
GPIO10	EPWM6A	CANRXB	**$\overline{\text{ADCSOCBO}}$**
GPIO33	SCLA	EPWMSYNCO	**$\overline{\text{ADCSOCBO}}$**
	eCAN-A		
GPIO30	**CANRXA**	保留	保留
GPIO31	**CANTXA**	保留	保留
	eCAN-B		
GPIO8	EPWM5A	**CANTXB**	$\overline{\text{ADCSOCAO}}$
GPIO10	EPWM6A	**CANRXB**	$\overline{\text{ADCSOCBO}}$
GPIO12	$\overline{\text{TZ1}}$	**CANTXB**	SPISIMOB
GPIO13	$\overline{\text{TZ2}}$	**CANRXB**	SPISOMIB
GPIO16	SPISIMOA	**CANTXB**	$\overline{\text{TZ5}}$
GPIO17	SPISOMIA	**CANRXB**	$\overline{\text{TZ6}}$
GPIO20	EQEP1A	SPISIMOC	**CANTXB**
GPIO21	EQEP1B	SPISOMIC	**CANRXB**
	eCAP1		
GPIO5	EPWM3B	SPICLKD	**ECAP1**
GPIO24	**ECAP1**	EQEP2A	SPISIMOB
	eCAP2		
GPIO7	EPWM4B	SPISTED	**ECAP2**
GPIO25	**ECAP2**	EQEP2B	SPISOMIB
	eCAP3		
GPIO9	EPWM5B	SCITXB	**ECAP3**
GPIO26	**ECAP3**	EQEP2I	SPICLKB
	eCAP4		
GPIO11	EPWM6B	SCIRXB	**ECAP4**
GPIO27	**ECAP4**	EQEP2S	SPISTEB

表 4-8. 外设与 GPIO 交叉参考(接上表)

主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
	ePWM1-6		
GPIO0	**EPWM1A**	保留	保留
GPIO1	**EPWM1B**	SPISIMOD	保留
GPIO2	**EPWM2A**	保留	保留
GPIO3	**EPWM2B**	SPISOMID	保留
GPIO4	**EPWM3A**	保留	保留
GPIO5	**EPWM3B**	SPICLKD	ECAP1
GPIO6	**EPWM4A**	EPWMSYNCI	EPWMSYNCO
GPIO7	**EPWM4B**	SPISTED	ECAP2
GPIO8	**EPWM5A**	CANTXB	$\overline{\text{ADCSOCAO}}$
GPIO9	**EPWM5B**	SCITXB	ECAP3
GPIO10	**EPWM6A**	CANRXB	$\overline{\text{ADCSOCBO}}$
GPIO11	**EPWM6B**	SCIRXB	ECAP4
	ePWM 同步输入		
GPIO6	EPWM4A	**EPWMSYNCI**	EPWMSYNCO
GPIO32	SDAA	**EPWMSYNCI**	$\overline{\text{ADCSOCAO}}$
	ePWM 同步输出		
GPIO6	EPWM4A	EPWMSYNCI	**EPWMSYNCO**
GPIO33	SCLA	**EPWMSYNCO**	$\overline{\text{ADCSOCBO}}$
	eQEP1		
GPIO20	**EQEP1A**	SPISIMOC	CANTXB
GPIO21	**EQEP1B**	SPISOMIC	CANRXB
GPIO22	**EQEP1S**	SPICLKC	SCITXB
GPIO23	**EQEP1I**	SPISTEC	SCIRXB
	eQEP2		
GPIO24	ECAP1	**EQEP2A**	SPISIMOB
GPIO25	ECAP2	**EQEP2B**	SPISOMIB
GPIO26	ECAP3	**EQEP2I**	SPICLKB
GPIO27	ECAP4	**EQEP2S**	SPISTEB
	I^2C-A		
GPIO32	**SDAA**	EPWMSYNCI	$\overline{\text{ADCSOCAO}}$
GPIO33	**SCLA**	EPWMSYNCO	$\overline{\text{ADCSOCBO}}$
	SCI-A		
GPIO28	**SCIRXDA**	保留	$\overline{\text{TZ5}}$
GPIO29	**SCITXDA**	保留	$\overline{\text{TZ6}}$
	SCI-B		
GPIO9	EPWM5B	**SCITXB**	ECAP3
GPIO11	EPWM6B	**SCIRXB**	ECAP4
GPIO14	$\overline{\text{TZ3}}$	**SCITXB**	SPICLKB
GPIO15	$\overline{\text{TZ4}}$	**SCIRXB**	SPISTEB
GPIO18	SPICLKA	**SCITXB**	保留
GPIO19	SPISTEA	**SCIRXB**	保留
GPIO22	EQEP1S	SPICLKC	**SCITXB**
GPIO23	EQEP1I	SPISTEC	**SCIRXB**

表 4-8. 外设与 GPIO 交叉参考(接上表)

主 I/O 功能 （GPxMUX1/2 位=0,0）	外设选择 1 （GPxMUX1/2 位=0,1）	外设选择 2 （GPxMUX1/2 位=1,0）	外设选择 3 （GPxMUX1/2 位=1,1）
	SPI-A		
GPIO16	**SPISIMOA**	CANTXB	**$\overline{TZ5}$**
GPIO17	**SPISOMIA**	CANRXB	**$\overline{TZ6}$**
GPIO18	**SPICLKA**	SCITXB	保留
GPIO19	**SPISTEA**	SCIRXB	保留
	SPI-B		
GPIO12	**$\overline{TZ1}$**	CANTXB	**SPISIMOB**
GPIO13	**$\overline{TZ2}$**	CANRXB	**SPISOMIB**
GPIO14	**$\overline{TZ3}$**	SCITXB	**SPICLKB**
GPIO15	**$\overline{TZ4}$**	SCIRXB	**SPISTEB**
GPIO24	ECAP1	EQEP2A	**SPISIMOB**
GPIO25	ECAP2	EQEP2B	**SPISOMIB**
GPIO26	ECAP3	EQEP2I	**SPICLKB**
GPIO27	ECAP4	EQEP2S	**SPISTEB**
	SPI-C		
GPIO20	EQEP1A	**SPISIMOC**	CANTXB
GPIO21	EQEP1B	**SPISOMIC**	CANRXB
GPIO22	EQEP1S	**SPICLKC**	SCITXB
GPIO23	EQEP1I	**SPISTEC**	SCIRXB
	SPI-D		
GPIO1	EPWM1B	**SPISIMOD**	保留
GPIO3	EPWM2B	**SPISOMID**	保留
GPIO5	EPWM3B	**SPICLKD**	ECAP1
GPIO7	EPWM4B	**SPISTED**	ECAP2
	跳匣区域 1-6		
GPIO12	**$\overline{TZ1}$**	CANTXB	SPISIMOB
GPIO13	**$\overline{TZ2}$**	CANRXB	SPISOMIB
GPIO14	**$\overline{TZ3}$**	SCITXB	SPICLKB
GPIO15	**$\overline{TZ4}$**	SCIRXB	SPISTEB
GPIO16	SPISIMOA	CANTXB	**$\overline{TZ5}$**
GPIO17	SPISOMIA	CANRXB	**$\overline{TZ6}$**
GPIO28	SCIRXDA	保留	**$\overline{TZ5}$**
GPIO29	SCITXDA	保留	**$\overline{TZ6}$**

4.5 寄存器位定义

图 4-4. GPIO 端口 A MUX 1 (GPAMUX1) 寄存器

31 30	29 28	27 号 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND- R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-9. GPIO 端口 A MUX 1 (GPAMUX1) 寄存器字段说明

位	字段	值	说明(1)
31-30	GPIO15		将 GPIO15 引脚配置为：
		00	GPIO15 (I/O)，通用 I/O 15（默认值）
		01	$\overline{TZ4}$ (I)，跳匣区域 4
		10	SCIRXB (I)，SCI-B 接收。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISTEB (I/O)，启用 SPI-B 发送。
29-28	GPIO14		将 GPIO14 引脚配置为：
		00	GPIO14 (I/O)，通用 I/O 14（默认值）
		01	$\overline{TZ3}$ (I)，跳匣区域 3
		10	SCITXB (O)，SCI-B 发送
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPICLKB (I/O)，SPI-B 时钟输入
27-26	GPIO13		将 GPIO13 引脚配置为：
		00	GPIO13 (I/O)，通用 I/O 13（默认值）
		01	$\overline{TZ2}$ (I)，跳匣区域 2
		10	CANRXB (I)，eCAN-B 接收
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISOMIB (I/O)
25-24	GPIO12		将 GPIO12 引脚配置为：
		00	GPIO12 (I/O)，通用 I/O 12（默认值）
		01	$\overline{TZ1}$ (I)，跳匣区域 1
		10	CANTXB (O)，eCAN-B 发送。
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISIMOB (I/O)，SPI-B 从属输入、主输出。
23-22	GPIO11		将 GPIO11 引脚配置为：
		00	GPIO11 (I/O)，通用 I/O 11（默认值）
		01	EPWM6B (O)，ePWM 6 输出 B。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且将被驱动为低电平。
		10	SCIRXB (I)，SCI-B 接收。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	ECAP4 (I/O)，eCAP4。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-9. GPIO 端口 A MUX 1 (GPAMUX1) 寄存器字段说明(接上表)

位	字段	值	说明(1)
21-20	GPIO10		将 GPIO10 引脚配置为：
		00	GPIO10 (I/O)，通用 I/O 10（默认值）
		01	EPWM6A (O)，ePWM6 输出 A。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且不会被驱动为低电平。
		10	CANRXB (I)，eCAN-B 接收
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	$\overline{\text{ADCSOCBO}}$ (O)，转换 B 的 ADC 启动
19-18	GPIO9		将 GPIO9 引脚配置为：(1)
		00	GPIO9 (I/O)，通用 I/O 9（默认值）
		01	EPWM5B (O)，ePWM5 输出 B。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且不会被驱动为低电平。
		10	SCITXB (O)，SCI-B 发送
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	ECAP3 (I/O)，eCAP3。
17-16	GPIO8		将 GPIO8 引脚配置为：
		00	GPIO8 (I/O)，通用 I/O 8（默认值）
		01	EPWM5A (O)，ePWM5 输出 A。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且不会被驱动为低电平。
		10	CANTXB (O)，eCAN-B 发送。
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	$\overline{\text{ADCSOCAO}}$ (O)，转换 A 的 ADC 启动
15-14	GPIO7		将 GPIO7 引脚配置为：
		00	GPIO7 (I/O)，通用 I/O 7（默认值）
		01	EPWM4B (O)，ePWM4 输出 B。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且将被驱动为低电平。
		10	SPISTED (I/O)，启用 SPI-D 发送。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	ECAP2 (I/O)，eCAP2
13-12	GPIO6		将 GPIO6 引脚配置为：
		00	GPIO6 (I/O)，通用 I/O 6（默认值）
		01	EPWM4A (O)，ePWM4 输出 A。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输出且不会被驱动为低电平。
		10	EPWMSYNCI (I)，ePWM 同步输入
		11	EPWMSYNCO (O)，ePWM 同步输出
11-10	GPIO5		将 GPIO5 引脚配置为：
		00	GPIO5 (I/O)，通用 I/O 5（默认值）
		01	EPWM3B (O)，ePWM3 输出 B。
		10	SPICLKD (I/O)，SPI-D 时钟。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	ECAP1 (I/O)，eCAP1
9-8	GPIO4		将 GPIO4 引脚配置为：
		00	GPIO4 (I/O)，通用 I/O 4（默认值）
		01	EPWM3A (O)，ePWM3 输出 A。
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。

表 4-9. GPIO 端口 A MUX 1 (GPAMUX1) 寄存器字段说明(接上表)

位	字段	值	说明(1)
7-6	GPIO3		将 GPIO3 引脚配置为：
		00	GPIO3 (I/O)，通用 I/O 3（默认值）
		01	EPWM2B (O)，ePWM2 输出 B。
		10	SPISOMID (I/O)，SPI-D 从属输出、主输入。 在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
5-4	GPIO2		将 GPIO2 引脚配置为：
		00	GPIO2 (I/O) 通用 I/O 2（默认值）
		01	EPWM2A (O)，ePWM2 输出 A。
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
3-2	GPIO1		将 GPIO1 引脚配置为：
		00	GPIO1 (I/O) 通用 I/O 1（默认值）
		01	EPWM1B (O)，ePWM1 输出 B。
		10	SPISIMOD (I/O)，SPI-D 从属输入、主输出。 在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
1-0	GPIO0		将 GPIO0 引脚配置为：
		00	GPIO0 (I/O)，通用 I/O 0（默认值）
		01	EPWM1A (O)，ePWM1 输出 A。
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。

图 4-5. GPIO 端口 A MUX 2 (GPAMUX2) 寄存器

31 30	29 28	27 号 26	25 24	23 22	21 20	19 18	17 16
GPIO31	GPIO30	GPIO29	GPIO128	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-10. GPIO 端口 A MUX 2 (GPAMUX2) 寄存器字段说明

位	字段	值	说明(1)
31-30	GPIO31		将 GPIO31 引脚配置为：
		00	GPIO31 (I/O) 通用 I/O 31（默认值）
		01	CANTXA (O)，eCAN-A 发送
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
29-28	GPIO30		将 GPIO30 引脚配置为：
		00	GPIO30 (I/O) 通用 I/O 30（默认值）
		01	CANRXA (I)，eCAN-A 接收
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-10. GPIO 端口 A MUX 2 (GPAMUX2) 寄存器字段说明(接上表)

位	字段	值	说明(1)
27-26	GPIO29		将 GPIO29 引脚配置为：
		00	GPIO29 (I/O) 通用 I/O 29（默认值）
		01	SCITXDA (O)，eSCI-D 发送
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	$\overline{TZ6}$ (I)，跳匣区域 6
25-24	GPIO28		将 GPIO28 引脚配置为：
		00	GPIO28 (I/O) 通用 I/O 28（默认值）
		01	SCIRXDA (I)，eSCI-A 接收
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	$\overline{TZ5}$ (I)，跳匣区域 5
23-22	GPIO27		将 GPIO27 引脚配置为：
		00	GPIO27 (I/O)，通用 I/O 27（默认值）
		01	ECAP4 (I/O), eCAP4。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		10	EQEP2S (I/O), eQEP2 选通。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISTEB (I/O)，启用 SPI-B 发送。
21-20	GPIO26		将 GPIO26 引脚配置为：
		00	GPIO26 (I/O)，通用 I/O 26（默认值）
		01	ECAP3 (I/O), eCAP3。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		10	EQEP2I (I/O), eQEP2 索引。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPICLKB (I/O)，SPI-B 时钟。
19-18	GPIO25		将 GPIO25 引脚配置为：
		00	GPIO25 (I/O)，通用 I/O 25（默认值）
		01	ECAP2 (I/O), eCAP2
		10	EQEP2B (I)，eQEP2 输入 B。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISOMIB (I/O)，SPI-B 从属输出、主输入。
17-16	GPIO24		将 GPIO24 引脚配置为：
		00	GPIO24 (I/O)，通用 I/O 24（默认值）
		01	ECAP1 (I/O), eCAP1
		10	EQEP2A (I)，eQEP2 输入 A。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SPISIMOB (I/O)，SPI-B 从属输入、主输出。
15-14	GPIO23		将 GPIO23 引脚配置为：
		00	GPIO23 (I/O)，通用 I/O 23（默认值）
		01	EQEP1I (I/O), eQEP1 索引。
		10	SPISTEC (I/O)，启用 SPI-C 发送。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	SCIRXB (I/O)，SCI-B 时钟。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。

表 4-10. GPIO 端口 A MUX 2 (GPAMUX2) 寄存器字段说明(接上表)

位	字段	值	说明(1)
13-12	GPIO22		将 GPIO22 引脚配置为：
		00	GPIO22 (I/O)，通用 I/O 22（默认值）
		01	EQEP1S (I/O), eQEP1 选通。
		10	SPICLKC (I/O)，SPI-C 时钟。
			在 2801 器件上保留此选项。如果选择了它，则引脚将成为输入且不会被驱动。
		11	SCITXB (O)，SCI-B 发送
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
11-10	GPIO21		将 GPIO21 引脚配置为：
		00	GPIO21 (I/O)，通用 I/O 21（默认值）
		01	EQEP1B (I)，eQEP1 输入 B。
		10	SPISOMIC (I/O)，SPI-C 从属输出、主输入。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	CANRXB (I)，eCAN-B 接收
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
9-8	GPIO20		将 GPIO20 引脚配置为：
		00	GPIO20 (I/O) 通用 I/O 22（默认值）
		01	EQEP1A (I)，eQEP1 输入 A。
		10	SPISIMOC (I/O)，SPI-C 从属输入、主输出。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	CANTXB (O)，eCAN-B 发送。
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
7-6	GPIO19		将 GPIO19 引脚配置为：
		00	GPIO19 (I/O)，通用 I/O 19（默认值）
		01	SPISTEA (I/O)，启用 SPI-A 发送。
		10	SCIRXB (I)，SCI-B 接收。
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
5-4	GPIO18		将 GPIO18 引脚配置为：
		00	GPIO18 (I/O)，通用 I/O 18（默认值）
		01	SPICLKA (I/O)，SPI-A 时钟。
		10	SCITXB (O)，SCI-B 发送
			在 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
3-2	GPIO17		将 GPIO17 引脚配置为：
		00	GPIO17 (I/O)，通用 I/O 17（默认值）
		01	SPISOMIA (I/O)，SPI-A 从属输出、主输入。
		10	CANRXB (I)，eCAN-B 接收
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	$\overline{\text{TZ6}}$ (I)，跳匣区域 6
1-0	GPIO16		将 GPIO16 引脚配置为：
		00	GPIO16 (I/O)，通用 I/O 16（默认值）
		01	SPISIMOA (I/O)，SPI-A 从属输入、主输出
		10	CANTXB (O)，eCAN-B 发送。
			在 2806 和 2801 器件上保留此选项。 如果选择了它，则引脚将成为输入且不会被驱动。
		11	$\overline{\text{TZ5}}$ (I)，跳匣区域 5

图 4-6. GPIO 端口 B MUX 1 (GPBMUX1) 寄存器

31	6	5	4	3	2	1	0
保留		GPIO34		GPIO33		GPIO32	
R-0		R/W-0		R/W-0		R/W-0	

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-11. GPIO 端口 B MUX 1 (GPBMUX1) 寄存器字段说明

位	字段	值	说明(1)
31-6	保留		保留
5-4	GPIO34		将 GPIO34 引脚配置为：
		00	GPIO34 (I/O)，通用 I/O 34（默认值）
		01	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		10	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
		11	保留。 如果选择了此选项，则引脚将成为输入且不会被驱动。
3-2	GPIO33		将 GPIO33 引脚配置为：
		00	GPIO33 (I/O)，通用 I/O 33（默认值）
		01	SCLA (I/O)，I2C 时钟。
		10	EPWMSYNCO (O)，ePWM 同步输出
		11	$\overline{\text{ADCSOCBO}}$ (O)
1-0	GPIO32		将 GPIO32 引脚配置为：
		00	GPIO32 (I/O)，通用 I/O 32（默认值）
		01	SDAA (I/O)，I2C 数据。
		10	EWPMSYNCI (I)，ePWM 同步输入
		11	$\overline{\text{ADCSOCAO}}$ (O)，转换 A 的 ADC 启动

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-12. GPIO 端口 B MUX 2 (GPBMUX2) 寄存器字段说明

位	字段	类型	复位	说明
31-0	保留	R/W	0,0	保留

图 4-7. GPIO 端口 A 鉴定控制 (GPACTRL) 寄存器

31	24	23	16
QUALPRD3		QUALPRD2	
R/W-0		R/W-0	
15	**8**	**7**	**0**
QUALPRD1		QUALPRD0	
R/W-0		R/W-0	

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-13. GPIO 端口 A 鉴定控制 (GPACTRL) 寄存器字段说明

位	字段	值	说明(1)
31-24	QUALPRD3		指定 GPIO24 到 GPIO31 的鉴定采样周期。 采样周期是鉴定采样之间相对于 SYSCLKOUT 的时间量。 在 GPAQSEL1/2 寄存器中定义了采样次数。
		0x00	QUALPRD = SYSCLKOUT
		0x01	QUALPRD = SYSCLKOUT/2
		0x02	QUALPRD = SYSCLKOUT/4
		。 。 。	。 。 。
		0xFF	QUALPRD = SYSCLKOUT/510
23-16	QUALPRD2		指定 GPIO16 到 GPIO23 的鉴定采样周期。 采样周期是鉴定采样之间相对于 SYSCLKOUT 的时间量。 在 GPAQSEL1/2 寄存器中定义了采样次数。
		0x00	QUALPRD = SYSCLKOUT
		0x01	QUALPRD = SYSCLKOUT/2
		0x02	QUALPRD = SYSCLKOUT/4
		。 。 。	。 。 。
		0xFF	QUALPRD = SYSCLKOUT/510
15-8	QUALPRD1		指定 GPIO8 到 GPIO15 的鉴定采样周期。 采样周期是鉴定采样之间相对于 SYSCLKOUT 的时间量。 在 GPAQSEL1/2 寄存器中定义了采样次数。
		0x00	QUALPRD = SYSCLKOUT
		0x01	QUALPRD = SYSCLKOUT/2
		0x02	QUALPRD = SYSCLKOUT/4
		。 。 。	。 。 。
		0xFF	QUALPRD = SYSCLKOUT/510
7-0	QUALPRD0		指定 GPIO0 到 GPIO7 的鉴定采样周期。 采样周期是鉴定采样之间相对于 SYSCLKOUT 的时间量。 在 GPAQSEL1/2 寄存器中定义了采样次数。
		0x00	QUALPRD = SYSCLKOUT
		0x01	QUALPRD = SYSCLKOUT/2
		0x02	QUALPRD = SYSCLKOUT/4
		。 。 。	。 。 。
		0xFF	QUALPRD = SYSCLKOUT/510

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-8. GPIO 端口 B 鉴定控制 (GPBCTRL) 寄存器

31	8	7	0
保留		QUALPRD0	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-14. GPIO 端口 B 输入鉴定控制 (GPBCTRL) 寄存器字段说明

位	字段	值	说明(1)
31-8	保留		保留
7-0	QUALPRD0		指定 GPIO32 到 GPIO34 的鉴定采样周期。 采样周期是鉴定采样之间相对于 SYSCLKOUT 的时间量。 在 GPBQSEL1 寄存器中定义了要进行的采样次数。
		0x00	QUALPRD = SYSCLKOUT
		0x01	QUALPRD = SYSCLKOUT/2
		0x02	QUALPRD = SYSCLKOUT/4
		。 。 。	。 。 。
		0xFF	QUALPRD = SYSCLKOUT/510

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-9. GPIO 端口 A 鉴定选择 1 (GPAQSEL1) 寄存器

31 30	29 28	27 号 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-15. GPIO 端口 A 鉴定选择 1 (GPAQSEL1) 寄存器字段说明

位	字段	值	说明(1)
31-0	GPIO15-GPIO0		选择 GPIO0 到 GPIO15 的输入鉴定类型。 每个 GPIO 输入的输入鉴定由如 图 4-9。
		00	只与 SYSCLKOUT 同步。 对外设和 GPIO 引脚都有效。
		01	使用 3 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPACTRL 寄存器中可以指定采样之间的时间。
		10	使用 6 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPACTRL 寄存器中可以指定采样之间的时间。
		11	异步。 （没有同步或鉴定）。 此选项仅适用于配置为外设的引脚。 如果引脚配置为 GPIO 输入，则此选项与 0,0 或 “ 与 SYSCLKOUT 同步 ” 相同。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-10. GPIO 端口 A 鉴定选择 2 (GPAQSEL2) 寄存器

31 30	29 28	27 号 26	25 24	23 22	21 20	19 18	17 16
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-16. GPIO 端口 A 鉴定选择 2 (GPAQSEL2) 寄存器字段说明

位	字段	值	说明(1)
31-0	GPIO31-GPIO16		选择 GPIO16 到 GPIO31 的输入鉴定类型。 每个 GPIO 输入的输入鉴定由如 图 4-10。
		00	只与 SYSCLKOUT 同步。 对外设和 GPIO 引脚都有效。
		01	使用 3 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPACTRL 寄存器中可以指定采样之间的时间。
		10	使用 6 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPACTRL 寄存器中可以指定采样之间的时间。
		11	异步。 （没有同步或鉴定）。 此选项仅适用于配置为外设的引脚。 如果引脚配置为 GPIO 输入，则此选项与 0,0 或 “ 与 SYSCLKOUT 同步 ” 相同。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-11. GPIO 端口 B 鉴定选择 1 (GPBQSEL1) 寄存器

31 6	5 4	3 2	1 0
保留	GPIO34	GPIO33	GPIO32
R-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4.6. GPIO 端口 B 鉴定选择 1 (GPBQSEL1) 寄存器字段说明

位	字段	值	说明(1)
31-6	保留		
5-0	GPIO34-GPIO32		选择 GPIO32 到 GPIO34 的输入鉴定类型。 每个 GPIO 输入的输入鉴定由如 图 4-11。
		00	只与 SYSCLKOUT 同步。 对外设和 GPIO 引脚都有效。
		01	使用 3 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPBCTRL 寄存器中可以指定采样之间的时间。
		10	使用 6 次采样的鉴定。 对配置为 GPIO 或外设功能的引脚有效。 在 GPBCTRL 寄存器中可以指定采样之间的时间。
		11	异步。 （没有同步或鉴定）。 此选项仅适用于配置为外设的引脚。 如果引脚配置为 GPIO 输入，则此选项与 0,0 或 “ 与 SYSCLKOUT 同步 ” 相同。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-17. GPIO 端口 B 鉴定选择 2 (GPBQSEL2) 寄存器字段说明

位	字段	说明
31-0	保留	保留供未来扩展

当引脚在合适的 MUX 寄存器中配置为 GPIO 时，GPADIR 和 GPBDIR 寄存器控制这些引脚的方向。 方向寄存器对配置为外设功能的引脚无效。

图 4-12. GPIO 端口 A 方向 (GPADIR) 寄存器

31	30	29	28	27 号	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-18. GPIO 端口 A 方向 (GPADIR) 寄存器字段说明

位	字段	值	说明(1)
31-0	GPIO31-GPIO0		当指定的引脚在合适的 GPAMUX1 或 GPAMUX2 寄存器中配置为 GPIO 时，控制 GPIO 端口 A 引脚的方向。
		0	将 GPIO 引脚配置为输入。 （默认值）
		1	将 GPIO 引脚配置为输出。
			在此引脚上驱动 GPADAT 输出锁定中的当前值。 要在将引脚从输入更改为输出之前初始化 GPADAT 锁定，请使用 GPASET、GPACLEAR 和 GPATOGGLE 寄存器。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-13. GPIO 端口 B 方向 (GPBDIR) 寄存器

31	3	2	1	0
保留		GPIO34	GPIO33	GPIO32
R-0		R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-19. GPIO 端口 B 方向 (GPBDIR) 寄存器字段说明

位	字段	值	说明(1)
31-3	保留		
2-0	GPIO34-GPIO32		当指定的引脚在合适的 GPBMUX1 或 GPBMUX2 寄存器中配置为 GPIO 时，控制 GPIO 端口 B 引脚的方向（输入或输出）。
		0	将 GPIO 引脚配置为输入。 （默认值）
		1	将 GPIO 引脚配置为输出。
			在此引脚上驱动 GPBDAT 输出锁定中的当前值。 要在将引脚从输入更改为输出之前初始化 GPBDAT 锁定，请使用 GPBSET、GPBCLEAR 和 GPBTOGGLE 寄存器。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

上拉禁用 (GPxPUD) 寄存器允许您指定哪些引脚应启用内部上拉电阻。 可配置为 ePWM 输出 (GPIO0-GPIO11) 的引脚上的内部上拉电阻在外部复位信号 ($\overline{XRS}$) 为低电平时全部被异步禁用。 所有其它引脚上的内部上拉电阻在复位时被启用。 当退出复位时，上拉电阻保持它们的默认状态，直到您在软件中通过写入此寄存器启用或禁用它们。 上拉配置同时适用于配置为 I/O 的引脚和配置为外设功能的引脚。

图 4-14. GPIO 端口 A 上拉禁用 (GPAPUD) 寄存器

31	30	29	28	27 号	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-20. GPIO 端口 A 内部上拉禁用 (GPAPUD) 寄存器字段说明

位	字段	值	说明 (1)
31-0	GPIO31-GPIO0		配置所选 GPIO 端口 A 引脚上的内部上拉电阻。 每个 GPIO 引脚与此寄存器中的一位相对应，如 图 4-14。
		0	启用指定引脚上的内部上拉电阻。 （GPIO12-GPIO31 的默认值）
		1	禁用指定引脚上的内部上拉电阻。 （GPIO0-GPIO11 的默认值）

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-15. GPIO 端口 B 上拉禁用 (GPBPUD) 寄存器

31	3	2	1	0
保留		GPIO34	GPIO33	GPIO32
R-0		R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-21. GPIO 端口 B 上拉禁用 (GPBPUD) 寄存器字段说明

位	字段	值	说明 (1)
31-3	保留		保留
2-0	GPIO34-GPIO32		配置所选 GPIO 端口 B 引脚上的内部上拉电阻。 每个 GPIO 引脚与此寄存器中的一位相对应，如 图 4-15。
		0	启用指定引脚上的内部上拉电阻。 （默认值）
		1	禁用指定引脚上的内部上拉电阻。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

不管引脚处于哪种模式，GPIO 数据寄存器都会指示 GPIO 引脚的当前状态。 如果引脚已启用为 GPIO 输出，则写入此寄存器会将各个 GPIO 引脚设置为高电平或低电平，否则写入的值被锁定但被忽略。 输出寄存器锁定的状态将保持为其当前状态，直到下一个写入操作。 复位会将所有位和锁定值清零。 从 GPADAT 和 GPBDAT 寄存器读取的值反映引脚（鉴定之后）的状态，而不是 GPADAT 或 GPBDAT 寄存器的输出锁定状态。

通常 DAT 寄存器用于读取引脚的当前状态。 要轻松地修改引脚的输出电平，请参阅 SET、CLEAR 和 TOGGLE 寄存器。

图 4-16. GPIO 端口 A 数据 (GPADAT) 寄存器

31	30	29	28	27 号	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值(1)

(1) x = 复位之后 GPADAT 寄存器的状态未知。 它取决于引脚在复位之后的电平。

表 4-22. GPIO 端口 A 数据 (GPADAT) 寄存器字段说明

位	字段	值	说明
31-0	GPIO31-GPIO0		每位对应一个 GPIO 端口 A 引脚 (GPIO0-GPIO31)，如 图 4-16。
		0	读取 0 指示引脚的当前状态为低，而不管该引脚的配置模式。 如果在合适的 GPAMUX1/2 和 GPADIR 寄存器中将引脚配置为 GPIO 输出，则写入 0 将强制输出为 0；否则，该值被锁定但不用于驱动引脚。
		1	读取 1 指示引脚的当前状态为高，而不管该引脚的配置模式。 如果在合适的 GPAMUX1/2 和 GPADIR 寄存器中将引脚配置为 GPIO 输出，则写入 1 将强制输出为 1；否则，该值被锁定但不用于驱动引脚。

图 4-17. GPIO 端口 B 数据 (GPBDAT) 寄存器

31	3	2	1	0
保留		GPIO34	GPIO33	GPIO32
R-0		R/W-x	R/W-x	R/W-x

LEGEND- R/W = 读/写；R = 只读；-*n* = 复位后的值(1)

(1) x = 复位之后 GPBDAT 寄存器的状态未知。 它取决于引脚在复位之后的电平。

表 4-23. GPIO 端口 B 数据 (GPBDAT) 寄存器字段说明

位	字段	值	说明
31-3	保留		保留
2-0	GPIO34-GPIO32		每位对应一个 GPIO 端口 B 引脚 (GPIO32-GPIO34)，如 图 4-17
		0	读取 0 指示引脚的当前状态为低，而不管该引脚的配置模式。 如果在合适的 GPBMUX1 和 GPBDIR 寄存器中将引脚配置为 GPIO 输出，则写入 0 将强制输出为 0；否则，该值被锁定但不用于驱动引脚。
		1	读取 1 指示引脚的当前状态为高，而不管该引脚的配置模式。 如果在合适的 GPBMUX1 和 GPBDIR 寄存器中将引脚配置为 GPIO 输出，则写入 1 将强制输出为 1；否则，该值被锁定但不用于驱动引脚。

图 4-18. GPIO 端口 A 设置、清除和转换（GPASET、GPACLEAR 和 GPATOGGLE）寄存器

31	30	29	28	27 号	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 4-24. GPIO 端口 A 设置 (GPASET) 寄存器字段说明

位	字段	值	说明
31-0	GPIO31-GPIO0		每个 GPIO 端口 A 引脚 (GPIO0-GPIO31) 与此寄存器中的一位相对应，如 图 4-18。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制将各个输出数据锁定设置为高电平。 如果引脚配置为 GPIO 输出，则它将被驱动为高电平。 如果引脚未配置为 GPIO 输出，则锁定将设置为高电平但不会驱动引脚。

表 4-25. GPIO 端口 A 清除 (GPACLEAR) 寄存器字段说明

位	字段	值	说明
31-0	GPIO31 - GPIO0		每个 GPIO 端口 A 引脚 (GPIO0-GPIO31) 与此寄存器中的一位相对应，如 图 4-18。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制将各个输出数据锁定设置为低电平。 如果引脚配置为 GPIO 输出，则它将被驱动为低电平。 如果引脚未配置为 GPIO 输出，则锁定将被清除但不会驱动引脚。

表 4-26. GPIO 端口 A 转换 (GPATOGGLE) 寄存器字段说明

位	字段	值	说明
31-0	GPIO31-GPIO0		每个 GPIO 端口 A 引脚 (GPIO0-GPIO31) 与此寄存器中的一位相对应，如 图 4-18。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制各个输出数据锁定从当前状态转换。 如果引脚配置为 GPIO 输出，则将向当前状态的相反方向驱动。 如果引脚未配置为 GPIO 输出，则锁定将被转换但不会驱动引脚。

图 4-19. GPIO 端口 B 设置、清除和转换（GPBSET、GPBCLEAR 和 GPBTOGGLE）寄存器

31	3	2	1	0
保留		GPIO34	GPIO33	GPIO32
R-0		R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-27. GPIO 端口 B 设置 (GPBSET) 寄存器字段说明

位	字段	值	说明
31-3	保留		保留
2-0	GPIO34-GPIO32		每个 GPIO 端口 B 引脚 (GPIO32-GPIO34) 与此寄存器中的一位相对应，如 图 4-19。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制将各个输出数据锁定设置为高电平。 如果引脚配置为 GPIO 输出，则它将被驱动为高电平。 如果引脚未配置为 GPIO 输出，则锁定将设置为高电平但不会驱动引脚。

表 4-28. GPIO 端口 B 清除 (GPBCLEAR) 寄存器字段说明

位	字段	值	说明
31-3	保留		保留
2-0	GPIO34-GPIO32		每个 GPIO 端口 B 引脚 (GPIO32-GPIO34) 与此寄存器中的一位相对应，如 图 4-19。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制将各个输出数据锁定设置为低电平。 如果引脚配置为 GPIO 输出，则它将被驱动为低电平。 如果引脚未配置为 GPIO 输出，则锁定将被清除但不会驱动引脚。

表 4-29. GPIO 端口 B 转换 (GPBTOGGLE) 寄存器字段说明

位	字段	值	说明
31-3	保留		保留
2-0	GPIO34-GPIO32		每个 GPIO 端口 B 引脚 (GPIO32-GPIO34) 与此寄存器中的一位相对应，如 图 4-19。
		0	忽略 0 的写入。 此寄存器始终读回 0。
		1	写入 1 会强制各个输出数据锁定从当前状态转换。 如果引脚配置为 GPIO 输出，则将向当前状态的相反方向驱动。 如果引脚未配置为 GPIO 输出，则锁定将被清除但不会驱动引脚。

图 4-20. GPIO XINT1、XINT2、XNMI 中断选择（GPIOXINT1SEL、GPIOXINT2SEL 和 GPIOXNMISEL）寄存器

15	5	4	0
保留		GPIOSEL	
R-0		R/W-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-30. GPIO XINT1 中断选择 (GPIOXINT1SEL) 寄存器字段说明

位	字段	值	说明(1)
15-5	保留		保留
4-0	GPIOSEL		选择的端口 A GPIO 信号 (GPIO0 - GPIO31) 将用作 XINT1 中断源。 另外，还可在 XINT1CR 寄存器中配置中断，如 第 6.6 部分。
		00000	选择 GPIO0 引脚作为 XINT1 中断源（默认值）
		00001	选择 GPIO1 引脚作为 XINT1 中断源
		。 。 。	。 。 。
		11110	选择 GPIO30 引脚作为 XINT1 中断源
		11111	选择 GPIO31 引脚作为 XINT1 中断源

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-31. GPIO XINT2 中断选择 (GPIOXINT2SEL) 寄存器字段说明

位	字段	值	说明(1)
15-5	保留		保留
4-0	GPIOSEL		选择的端口 A GPIO 信号 (GPIO0 - GPIO31) 将用作 XINT2 中断源。 另外，还可在 XINT2CR 寄存器中配置中断，如 第 6.6 部分。 要使用该信号作为转换的 ADC 启动，请在 ADCTRL2 寄存器中启用它。 ADCSOC 始终是上升沿敏感。
		00000	选择 GPIO0 引脚作为 XINT2 中断源（默认值）
		00001	选择 GPIO1 引脚作为 XINT2 中断源
		。 。 。	。 。 。
		11110	选择 GPIO30 引脚作为 XINT2 中断源
		11111	选择 GPIO31 引脚作为 XINT2 中断源

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

表 4-32. GPIO XNMI 中断选择 (GPIOXNMISEL) 寄存器字段说明

位	字段	值	说明(1)
15-5	保留		保留
4-0	GPIOSEL		选择的端口 A GPIO 信号 (GPIO0 - GPIO31) 将用作 XNMI 中断源。 另外，还可在 XNMICR 寄存器中配置中断，如 第 6.6 部分。
		00000	选择 GPIO0 引脚作为 XNMI 中断源（默认值）
		00001	选择 GPIO1 引脚作为 XNMI 中断源
		。 。 。	。 。 。
		11110	选择 GPIO30 引脚作为 XNMI 中断源
		11111	选择 GPIO31 引脚作为 XNMI 中断源

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

图 4-21. GPIO 低功率模式唤醒选择 (GPIOLPMSEL) 寄存器

31	30	29	28	27 号	26	25	24
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 4-33. GPIO 低功率模式唤醒选择 (GPIOLPMSEL) 寄存器字段说明

位	字段	值	说明(1)
31-0	GPIO31 - GPIO0		低功率模式唤醒选择。 此寄存器中的每个位对应一个 GPIO 端口 A 引脚 (GPIO0 - GPIO31)，如 图 4-21。
		0	如果该位被清除，则相应引脚上的信号将对 HALT 和 STANDBY 低功率模式无效。
		1	如果各个位设置为 1，则相应引脚上的信号能够将器件从 HALT 和 STANDBY 低功率模式唤醒。

(1) 此寄存器受 EALLOW 保护。 请参阅第 5.2 部分以了解详细信息。

TEXAS INSTRUMENTS

外设帧

本章描述外设帧以及器件仿真寄存器。

主题 页

5.1 外设帧寄存器 96
5.2 EALLOW 保护寄存器 98
5.3 器件仿真寄存器 102
5.5 先写后读保护 103

5.1 外设帧寄存器

280x 器件包含三个外设寄存器空间。 这些空间分类如下：

- 外设帧 0：这些是直接映射到 CPU 存储器总线的外设。 请参阅表 5-1。
- 外设帧 1：这些是映射到 32 位外设总线的外设。 请参阅表 5-2。
- 外设帧 2：这些是映射到 16 位外设总线的外设。 请参阅表 5-3。

表 5-1. 外设帧 0 寄存器

名称	地址范围	大小 (x16)	存取类型(1)
器件仿真寄存器	0x0880 0x09FF	384	EALLOW 保护
闪存寄存器(2)	0x0A80 0x0ADF	96	EALLOW 保护。 受 CSM 保护
代码安全模块寄存器	0x0AE0 0x0AEF	16	EALLOW 保护
CPU 定时器 0/1/2 寄存器	0x0C00 0x0C3F	64	未受 EALLOW 保护
PIE 寄存器	0x0CE0 0x0CFF	32	未受 EALLOW 保护
PIE 矢量表	0x0D00 0x0DFF	256	EALLOW 保护

(1) 如果寄存器受 EALLOW 保护，则在执行 EALLOW 指令之前不能执行写入。 EDIS 指令禁用写入以防止杂散代码或指针破坏寄存器内容。
(2) 闪存寄存器也受代码安全性模块 (CSM) 保护。

表 5-2. 外设帧 1 寄存器

名称	地址范围	大小 (x16)	存取类型(1)
eCANA 寄存器	0x6000 0x60FF	256 (128 x 32)	某些 eCAN 控制寄存器（以及在其它 eCAN 控制寄存器中选定的位）受 EALLOW 保护。
eCANA 邮箱 RAM	0x6100 0x61FF	256 (128 x 32)	未受 EALLOW 保护
eCANB 寄存器	0x6200 0x62FF	256 (128 x 32)	某些 eCAN 控制寄存器（以及在其它 eCAN 控制寄存器中选定的位）受 EALLOW 保护。
eCANB 邮箱 RAM	0x6300 0x63FF	256 (128 x 32)	未受 EALLOW 保护
ePWM1 寄存器	0x6800 0x683F	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
ePWM2 寄存器	0x6840 0x687F	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
ePWM3 寄存器	0x6880 0x68BF	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
ePWM4 寄存器	0x68C0 0x68FF	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
ePWM5 寄存器	0x6900 0x693F	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
ePWM6 寄存器	0x6940 0x697F	64 (32 x 32)	某些 ePWM 寄存器受 EALLOW 保护。 请参阅第 第 5.2 部分 节。
保留	0x69800x69FF	128	
eCAP1 寄存器	0x6A00 0x6A1F	32 (16 x 32)	未受 EALLOW 保护
eCAP2 寄存器	0x6A20 0x6A3F	32 (16 x 32)	未受 EALLOW 保护
eCAP3 寄存器	0x6A40 0x6A5F	32 (16 x 32)	未受 EALLOW 保护
eCAP4 寄存器	0x6A60 0x6A7F	32 (16 x 32)	未受 EALLOW 保护

(1) 外设帧 1 允许 16 位和 32 位存取。 所有 32 位存取与偶数地址边界对齐。

表 5-2. 外设帧 1 寄存器(接上表)

名称	地址范围	大小 (x16)	存取类型(1)
保留	0x6A80 0x6AFF	32 (16 x 32)	未受 EALLOW 保护
eQEP1 寄存器	0x6B00 0x6B3F	64 (32 x 32)	未受 EALLOW 保护
eQEP2 寄存器	0x6B40 0x6B7F	64 (32 x 32)	未受 EALLOW 保护
GPIO 控制寄存器	0x6F80 0x6FBF	128 (64 x 32)	EALLOW 保护
GPIO 数据寄存器	0x6FC0 0x6FDF	32 (16 x 32)	未受 EALLOW 保护
GPIO 中断和 LPM 选择寄存器	0x6FE0 0x6FFF	32 (16 x 32)	EALLOW 保护

表 5-3. 外设帧 2 寄存器

名称	地址范围	大小 (x16)	存取类型(1)
系统控制寄存器	0x7010 0x702F	32	EALLOW 保护
SPI-A 寄存器	0x7040 0x704F	16	未受 EALLOW 保护
SCI-A 寄存器	0x7050 0x705F	16	未受 EALLOW 保护
ADC 寄存器	0x7100 0x711F	32	未受 EALLOW 保护
SPI-B 寄存器	0x7740 0x774F	16	未受 EALLOW 保护
SCI-B 寄存器	0x7750 0x775F	16	未受 EALLOW 保护
SPI-C 寄存器	0x7760 0x776F	16	未受 EALLOW 保护
SPI-D 寄存器	0x7780 0x778F	16	未受 EALLOW 保护
I2C 寄存器	0x7900 0x793F	64	未受 EALLOW 保护

(1) 外设帧 2 只允许 16 位存取。 所有 32 位存取被忽略（可能返回或写入无效数据）。

5.2 EALLOW 保护寄存器

280x 器件上的几个控制寄存器受 EALLOW 保护机制保护以防止虚假的 CPU 写入。 状态寄存器 1 (ST1) 中的 EALLOW 位指示保护的状态是否如表 5-4。

表 5-4. 存取受 EALLOW 保护的寄存器

EALLOW 位	CPU 写入	CPU 读取	JTAG 写入	JTAG 读取
0	忽略	允许	允许(1)	允许
1	允许	允许	允许	允许

(1) 可以通过 JTAG 端口覆盖 EALLOW 位，从而允许在调试期间从 Code Composer Studio 界面对受保护的寄存器进行全面的访问。

在复位时 EALLOW 位被清除以启用 EALLOW 保护。 在受保护时，CPU 对受保护寄存器进行的所有写入被忽略且只允许 CPU 读取、JTAG 读取和 JTAG 写入。 如果设置了此位，则通过执行 EALLOW 指令可以允许 CPU 自由写入受保护的寄存器。 在修改寄存器之后，可以通过执行 EDI 指令清除 EALLOW 位使它们再次受保护。

以下列寄存器受 EALLOW 保护：

- 器件仿真寄存器
- 闪存寄存器
- CSM 寄存器
- PIE 矢量表
- 系统控制寄存器
- GPIO MUX 寄存器
- 某些 eCAN 寄存器

表 5-5. EALLOW 保护的器件仿真寄存器

名称	地址	大小 (x16)	说明
DEVICECNF	0x0880 0x0881	2	器件配置寄存器
PROTSTART	0x0884	1	块保护起始地址寄存器
PROTRANGE	0x0885	1	块保护范围地址寄存器

表 5-6. EALLOW 保护闪存 / OTP 配置寄存器

名称	地址	大小 (x16)	说明
FOPT	0x0A80	1	闪存选项寄存器
FPWR	0x0A82	1	闪存功率模式寄存器
FSTATUS	0x0A83	1	状态寄存器
FSTDBYWAIT	0x0A84	1	闪存休眠到待机等待状态寄存器
FACTIVEWAIT	0x0A85	1	闪存待机到活动等待状态寄存器
FBANKWAIT	0x0A86	1	闪存读取存取等待状态寄存器
FOTPWAIT	0x0A87	1	OTP 读取存取等待状态寄存器

表 5-7. EALLOW 保护代码安全模块 (CSM) 寄存器

寄存器名称	地址	大小 (x16)	寄存器说明
KEY0	0x0AE0	1	128 位 KEY 寄存器的低位字
KEY1	0x0AE1	1	128 位 KEY 寄存器的第二位字
KEY2	0x0AE2	1	128 位 KEY 寄存器的第三位字

表 5-7. EALLOW 保护代码安全模块 (CSM) 寄存器(接上表)

寄存器名称	地址	大小 (x16)	寄存器说明
KEY3	0x0AE3	1	128 位 KEY 寄存器的第四位字
KEY4	0x0AE4	1	128 位 KEY 寄存器的第五位字
KEY5	0x0AE5	1	128 位 KEY 寄存器的第六位字
KEY6	0x0AE6	1	128 位 KEY 寄存器的第七位字
KEY7	0x0AE7	1	128 位 KEY 寄存器的高位字
CSMSCR	0x0AEF	1	CSM 状态和控制寄存器

表 5-8. EALLOW 保护的 PIE 矢量表

名称	地址	大小 (x16)	说明
未使用	0x0D00	2	保留
	0x0D02		
	0x0D04		
	0x0D06		
	0x0D08		
	0x0D0A		
	0x0D0C		
	0x0D0E		
	0x0D10		
	0x0D12		
	0x0D14		
	0x0D16		
	0x0D18		
INT13	0x0D1A	2	外部中断 13 (XINT13) 或 CPU 定时器 1（供 RTOS 使用）
INT14	0x0D1C	2	CPU 定时器 2（供 RTOS 使用）
DATALOG	0x0D1E	2	CPU 数据记录中断
RTOSINT	0x0D20	2	CPU 实时操作系统中断
EMUINT	0x0D22	2	CPU 仿真中断
NMI	0x0D24	2	外部不可屏蔽中断
ILLEGAL	0x0D26	2	非法操作
USER1	0x0D28	2	用户定义的陷阱
。	。	。	。
USER12	0x0D3E	2	用户定义的陷阱
INT1.1	0x0D40	2	第 1 组中断矢量
.	.	.	
INT1.8	0x0D4E	2	
.	.	.	第 2 组中断矢量
.	.	.	到第 11 组中断矢量
.	.	.	
INT12.1	0x0DF0	2	第 12 组中断矢量
.	.	.	
INT12.8	0x0DFE	2	

表 5-9. EALLOW 保护 PLL、时钟、看门狗和低功率模式寄存器

名称	地址	大小 (x16)	说明
XCLK	0x7010	1	XCLKOUT 引脚控制、X1 和 XCLKIN 状态寄存器
PLLSTS	0x7011	1	PLL 状态寄存器
HISPCP	0x701A	1	HSPCLK 时钟的高速外设时钟预分频器寄存器
LOSPCP	0x701B	1	HSPCLK 时钟的低速外设时钟预分频器寄存器
PCLKCR0	0x701C	1	外设时钟控制寄存器 0
PCLKCR1	0x701D	1	外设时钟控制寄存器 1
LPMCR0	0x701E	1	低功率模式控制寄存器 0
PLLCR	0x7021	1	PLL 控制寄存器
SCSR	0x7022	1	系统控制与状态寄存器
WDCNTR	0x7023	1	看门狗计数器寄存器
WDKEY	0x7025	1	看门狗复位密钥寄存器
WDCR	0x7029	1	看门狗控制寄存器

表 5-10. EALLOW 保护 GPIO MUX 寄存器

名称	地址	大小 (x16)	说明
GPACTRL	0x6F80	2	GPIO A 控制寄存器 (GPIO0 至 GPIO31)
GPAQSEL1	0x6F82	2	GPIO A 限定器选择 1 寄存器 (GPIO0 至 GPIO15)
GPAQSEL2	0x6F84	2	GPIO A 限定器选择 2 寄存器 (GPIO16 至 GPIO31)
GPAMUX1	0x6F86	2	GPIO A MUX 1 寄存器 (GPIO0 至 GPIO15)
GPAMUX2	0x6F88	2	GPIO A MUX 2 寄存器 (GPIO16 至 GPIO31)
GPADIR	0x6F8A	2	GPIO A 方向寄存器 (GPIO0 至 GPIO31)
GPAPUD	0x6F8C	2	GPIO A 上拉禁用寄存器 (GPIO0 至 GPIO31)
GPBCTRL	0x6F90	2	GPIO B 控制寄存器 (GPIO32 至 GPIO35)
GPBQSEL1	0x6F92	2	GPIO B 限定器选择 1 寄存器 (GPIO32 至 GPIO35)
GPBQSEL2	0x6F94	2	保留
GPBMUX1	0x6F96	2	GPIO B Mux 1 寄存器 (GPIO32 至 GPIO35)
GPBMUX2	0x6F98	2	保留
GPBDIR	0x6F9A	2	GPIO B 方向寄存器 (GPIO32 至 GPIO35)
GPBPUD	0x6F9C	2	GPIO B 上拉禁用寄存器 (GPIO32 至 GPIO35)
GPIOXINT1SEL	0x6FE0	1	XINT1 GPIO 输入选择寄存器 (GPIO0 至 GPIO31)
GPIOXINT2SEL	0x6FE1	1	XINT2 GPIO 输入选择寄存器 (GPIO0 至 GPIO31)
GPIOXNMISEL	0x6FE2	1	XNMI GPIO 输入选择寄存器 (GPIO0 至 GPIO31)
GPIOLPMSEL	0x6FE8	2	LPM GPIO 选择寄存器 (GPIO0 至 GPIO31)

表 5-11. EALLOW 保护 eCAN-A 寄存器

名称	eCAN-A 地址	eCAN-B 地址	大小 (x16)	说明
CANMC	0x6014	0x6214	2	主控制寄存器(1)
CANBTC	0x6016	0x6216	2	位定时配置寄存器(2)
CANGIM	0x6020	0x6220	2	全局中断屏蔽寄存器(3)
CANMIM	0x6024	0x6224	2	邮箱中断屏蔽寄存器
CANTSC	0x602E	0x622E	2	时间戳计数器
CANTIOC	0x602A	0x622A	1	CANTXA 引脚的 I/O 控制寄存器(4)
CANRIOC	0x602C	0x622C	1	CANRXA 引脚的 I/O 控制寄存器(5)

(1) 只保护 CANMC[15-9] 和 [7-6] 位
(2) 只保护 BCR[23-16] 和 [10-0] 位
(3) 只保护 CANGIM[17-16]、[14-8] 和 [2-0] 位
(4) 只保护 IOCONT1[3]
(5) 只保护 IOCONT2[3]

表 5-12. EALLOW 保护 ePWM1 - ePWM3 寄存器

名称	ePWM1 地址	ePWM2 地址	ePWM3 地址	大小 (x16)	寄存器说明
TZSEL	0x6812	0x6852	0x6892	1	跳匣区域选择寄存器
TZDCSEL	0x6813	0x6853	0x6893	1	跳匣区域数字比较器选择寄存器
TZCTL	0x6814	0x6854	0x6894	1	跳匣区域控制寄存器
TZEINT	0x6815	0x6855	0x6895	1	跳匣区域启用中断寄存器
TZCLR	0x6817	0x6857	0x6897	1	跳匣区域清除寄存器
TZFRC	0x6818	0x6858	0x6898	1	跳匣区域强制寄存器
HRCNFG	0x6820	0x6860	0x68A0	1	HRPWM 配置寄存器

表 5-13. EALLOW 保护 ePWM4 - ePWM6 寄存器

名称	ePWM4 地址	ePWM5 地址	ePWM6 地址	大小 (x16)	寄存器说明
TZSEL	0x68D2	0x6912	0x6952	1	跳匣区域选择寄存器
TZDCSEL	0x68D3	0x6913	0x6953	1	跳匣区域数字比较器选择寄存器
TZCTL	0x68D4	0x6914	0x6954	1	跳匣区域控制寄存器
TZEINT	0x68D5	0x6915	0x6955	1	跳匣区域启用中断寄存器
TZCLR	0x68D7	0x6917	0x6957	1	跳匣区域清除寄存器
TZFRC	0x68D8	0x6918	0x6958	1	跳匣区域强制寄存器
HRCNFG	0x68E0	无	无	1	HRPWM 配置寄存器

5.3 器件仿真寄存器

这些寄存器用于控制 C28x CPU 的保护模式和监视某些关键器件信号。 表 5-14 中定义了这些寄存器。

表 5-14. 器件仿真寄存器

名称	地址	大小 (x16)	说明
DEVICECNF	0x0880 0x0881	2	器件配置寄存器
PARTID	0x0882	1	部件 ID 寄存器
REVID	0x0883	1	修订 ID 寄存器
PROTSTART	0x0884	1	块保护起始地址寄存器
PROTRANGE	0x0885	1	块保护范围地址寄存器

图 5-1. 器件配置 (DEVICECNF) 寄存器

31	20	19	18	17	16
保留		ENPROT	保留		
R-0		R/W-1	R-111		

15	6	5	4	3	2	0
保留		XRS	保留	VMAPS	保留	
R-0		R-P	R-0	R-P	R/W-011	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 5-15. DEVICECNF 寄存器字段说明

位	字段	值	说明
31-20	保留		保留
19	ENPROT		启用读 - 写保护模式位。
		0	禁用读 - 写保护模式
		1	根据 PROTSTART 和 PROTRANGE 寄存器的指定启用读写保护
18-6	保留		保留
5	XRS		复位输入信号状态。 此位直接连接到 $\overline{\text{XRS}}$ 输入引脚。
4	保留		保留
3	VMAPS		VMAP 配置状态。 此位指示 VMAP 的状态。
2-0	保留		保留

图 5-2. 部件 ID 寄存器

15	0
REVID	
R	

表 5.4. PARTID 寄存器字段说明

位	字段	类型	复位	值	说明
15-0	PARTID	R	(1)		这些 16 位指定器件的部件号：
				0x002C	F2801
				0x0034	F2806
				0x003C	F2808
					所有其它值保留或由其它器件使用。

(1) 复位值取决于器件，如寄存器说明中所示。

图 5-3. REVID 寄存器

15	0
REVID	
R	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 5-16. REVID 寄存器字段说明

位	字段	复位	说明
15-0	REVID		这些 16 位指定特定部件的芯片修订号。 此编号在芯片的第一个修订版本上始终从 0x0000 开始且在后续修订上递增。
		0x0000	修订 0（用于第一个芯片）
		0x0001	修订 A
		0x0002	修订 B 等

5.5 先写后读保护

PPROTSTART 和 PROTRANGE 寄存器设置具有 CPU 先“写”后“读”操作保护的存储器地址范围（操作按顺序而不按它们的自然管道次序进行）。 对某些外设操作，这是必需的保护。

示例：下列代码行对寄存器 1 (REG1) 位置执行写入，然后下一条指令执行从寄存器 2 (REG2) 位置的读取。在处理器存储器总线上，在禁用块保护的情况下，写入之前先发出读取操作，如下所示。

```
MOV @REG1,AL         ---------+ TBIT @REG2,#BIT_X    ---------|-------> Read                                +------->
Write
```

如果启用了块保护，则读取将一直等到写入发生，如下所示：

```
MOV    @REG1,AL         ---------+ TBIT    @REG2,#BIT_X    ---------|-----+                                  +-----|--
-> Write                                    +---> Read
```

表 5-17. PROTSTART 和 PROTRANGE 寄存器

名称	地址	大小	类型	复位	说明
PROTSTART	0x0884	16	R/W	0x0100(1)	PROTSTART 寄存器设置相对于处理器低 22 位地址范围的 16 个最高位的起始地址。 因此，最小分辨率为 64 个字。
PROTRANGE	0x0885	16	R/W	0x00FF(1)	PROTRANGE 寄存器设置块大小（从起始地址），从 64 个字开始并以 2 的倍数递增（64、128、256、512、1K、2K、4K、8K、16K、.... 和 2M）。

(1) 在复位时将选择这些寄存器的默认值以覆盖存储器映射（地址范围 0x4000-0x8000）的外设帧 1、外设帧 2 和 XINTF 区域 1 区域。

表 5-18. PROTSTART 有效值

		(1)寄存器位															
起始地址	寄存器值	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 0000	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0000 0040	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x0000 0080	0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0000 00C0	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
0x003F FF00	0xFFFC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0x003F FF40	0xFFFD	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0x003F FF80	0xFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0x003F FFC0	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(1) 计算寄存器值的最快方法是将所需的块起始地址除以 64。

表 5-19. PROTRANGE 有效值

		(1)寄存器位															
块大小	寄存器值	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
256	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
512	0x0007	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1K	0x000F	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
256K	0x0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
512K	0x1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1M	0x3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2M	0x7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4M	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(1) 并非所有寄存器值都有效。 PROTSTART 地址值必须为范围值的倍数。 例如：如果块大小设置为 4K，则起始地址只能位于任何 4K 边界。

TEXAS INSTRUMENTS

外设中断扩展 (PIE)

外设中断扩展 (PIE) 块将大量中断源多路复用成较小的中断输入集。 PIE 块可以支持 96 个单个中断，这些中断被分为 8 个一组的块。 每组馈入 12 条内核中断线路（INT1 到 INT12）中的一条。 96 个中断中的每一个中断都由其存储在专用 RAM 块中的矢量支持，您可以修改专用 RAM 块。 CPU 在对中断提供服务时自动提取合适的中断矢量。 它用 9 个 CPU 时钟周期提取矢量并保存重要的 CPU 寄存器。 因此，CPU 可以快速响应中断事件。 可以通过硬件和软件控制中断的优先级。 每个中断都可以在 PIE 块内启用/禁用。

主题 页

6.1 PIE 控制器概述 106
6.2 矢量表映射 108
6.3 中断源 110
6.4 PIE 配置寄存器 122
6.5 PIE 中断寄存器 123
6.6 外部中断控制寄存器 130

6.1 PIE 控制器概述

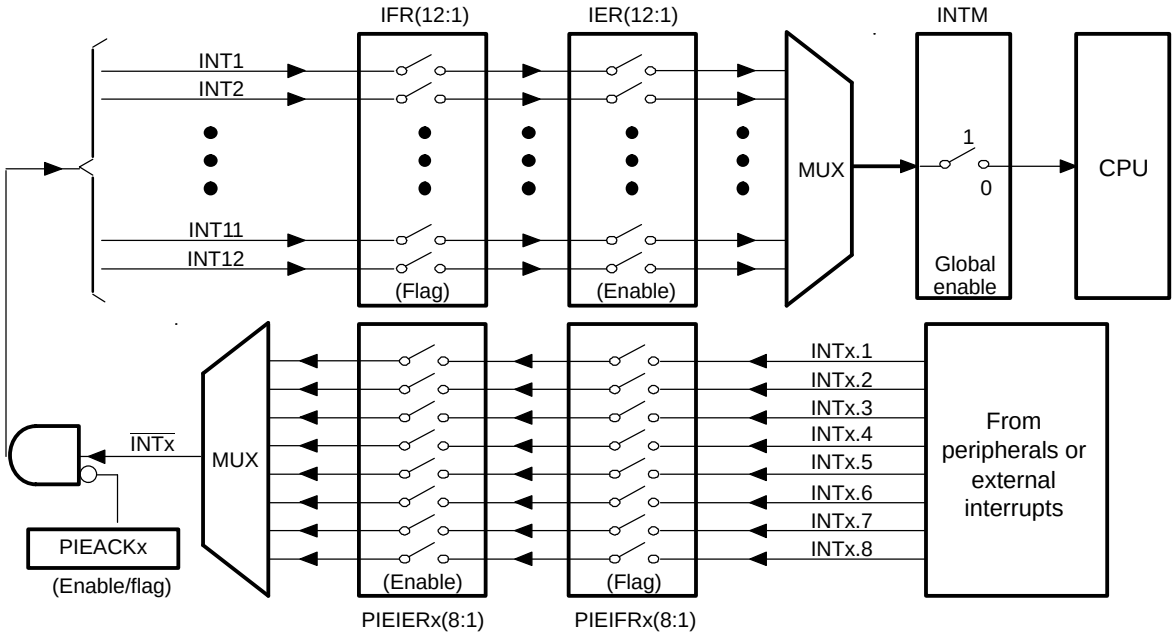
在 CPU 级别上，28x CPU 支持 1 个不可屏蔽中断 (NMI) 和 16 个可屏蔽的按优先级排列的中断请求（INT1-INT14、RTOSINT 和 DLOGINT）。 28x 器件具有许多外设，每个外设能够生成一个或多个中断以响应外设级别的许多事件。 因为 CPU 没有足够的能力在 CPU 级别处理所有外设中断请求，所以需要一个集中式外设中断扩展 (PIE) 控制器来仲裁来自各种源（如外设和其它外部引脚）的中断请求。

PIE 矢量表用于存储系统内每个中断服务例程 (ISR) 的地址（矢量）。 每个中断源（包括所有多路复用的和非多路复用的中断）有一个矢量。 在器件初始化期间将填充矢量表并且可以在操作期间进行更新。

6.1.1 中断操作顺序

图 6-1 显示了所有多路复用的 PIE 中断的中断操作顺序。 没有多路复用的中断源直接馈送到 CPU。

图 6-1. 概述：使用 PIE 块多路复用中断



- 外设级别

  外设中发生会产生中断的事件。 在该特定外设的寄存器中设置了对应该事件的中断标志 (IF) 位。

  如果设置了相应的中断启用 (IE) 位，则外设生成中断请求发送到 PIE 控制器。 如果未在外设级别启用中断，则 IF 保持设置直到被软件清除。 如果稍后启用了中断且仍然设置了中断标志，则向 PIE 发出中断请求。

  必须手动清除外设寄存器内的中断标志。 有关详细信息，请参阅特定外设的外设参考指南。

- PIE 级别

  PIE 模块将 8 个外设和外部引脚中断多路复用为一个 CPU 中断。 这些中断分为 12 组：PIE 组 1 - PIE 组 12。 组内的中断多路复用为一个 CPU 中断。 例如，PIE 组 1 多路复用为 CPU 中断 1 (INT1)，PIE 组 12 多路复用为 CPU 中断 12 (INT12)。 连接到其余 CPU 中断的中断源未多路复用。 对于非多路复用的中断，PIE 将请求直接传递给 CPU。

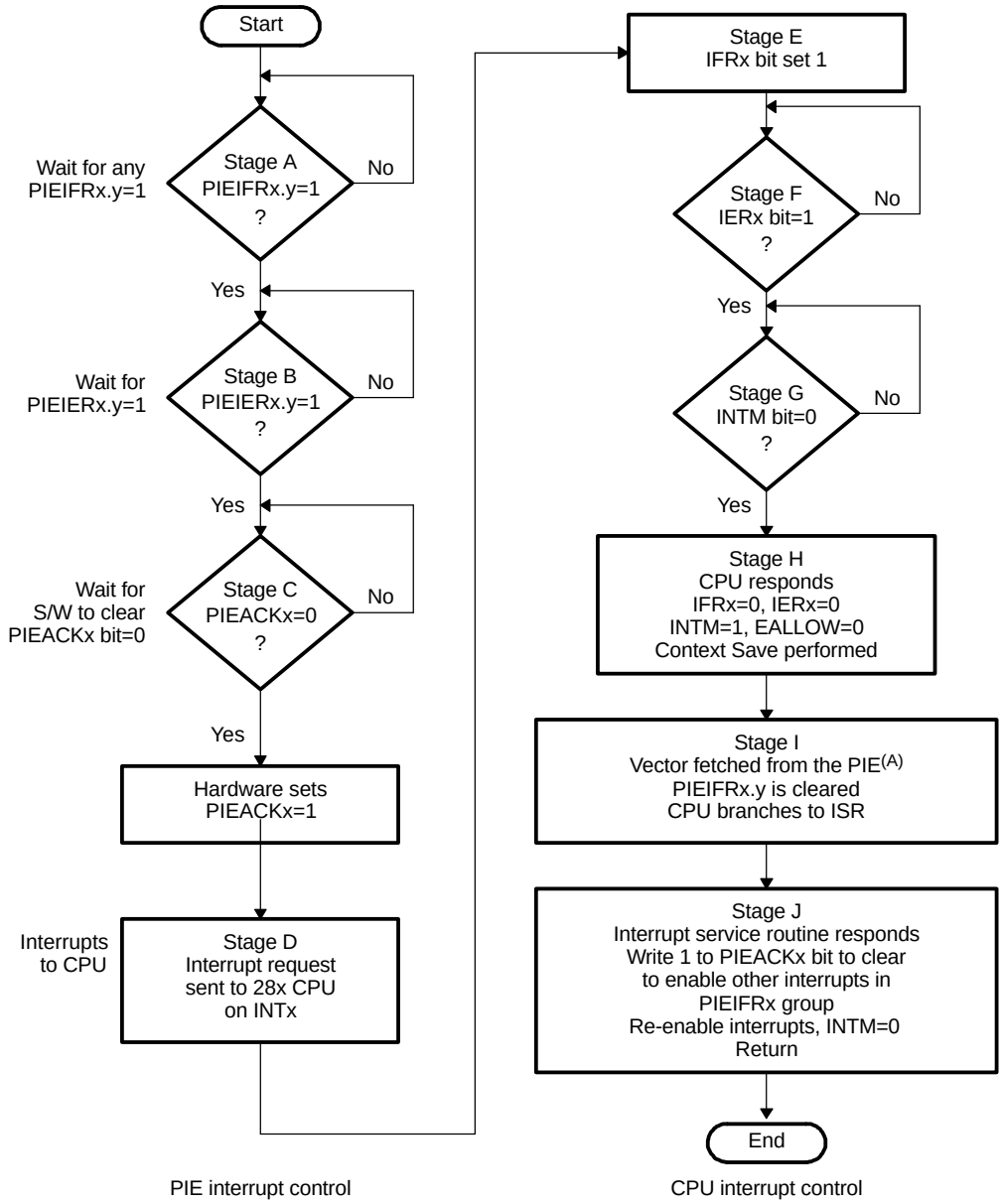
  对于多路复用中断源，PIE 模块中的每个中断组具有一个关联的标志寄存器 (PIEIFRx) 和启用 (PIEIERx) 寄存器（x = PIE 组 1 - PIE 组 12）。 每位（表示为 y）对应组内 8 个多路复用中断之一。 因此 PIEIFRx.y 和 PIEIERx.y 对应 PIE 组 x (x = 1-12) 内的中断 y (y = 1-8)。 另外，每个 PIE 中断组有一个确认位 (PIEACK)，称作 PIEACKx (x = 1-12)。 图 6-2 阐述了各种 PIEIFR 和 PIEIER 寄存器情况下 PIE 硬件的行为。

  一旦对 PIE 控制器发出请求，将设置相应的 PIE 中断标志 (PIEIFRx.y) 位。 如果还为给定中断设置了 PIE 中断启用 (PIEIERx.y) 位，则 PIE 检查相应的 PIEACKx 位以确定 CPU 是否已为来自该组的中断做好准备。 如果已为该组清除了 PIEACKx 位，则 PIE 向 CPU 发送中断请求。 如果设置了 PIEACKx，则 PIE 等待直到它被清除，再发送 INTx 的请求。 有关详细信息，请参阅第 6.3 节。

- CPU 级别

一旦向 CPU 发出了请求，将设置相应 INTx 的 CPU 级别中断标志 (IFR) 位。 在 IFR 中锁定了标志之后，直到在 CPU 中断启用 (IER) 寄存器或调试中断启用寄存器 (DBGIER) 和全局中断屏蔽 (INTM) 位中启用了相应的中断，才会为它提供服务。

图 6-2. 典型的 PIE/CPU 中断响应 - INTx.y



A 注： 对于多路复用中断，PIE 响应已标志且启用的具有最高优先级的中断。 如果没有已标志且启用的中断，则响应组内具有最高优先级的中断（INTx.1，其中 x 为 PIE 组）。 请参阅第 第 6.3.3 部分 节以了解详细信息。

如 表 6-1 所示，启用 CPU 级别的可屏蔽中断的要求取决于所用的中断处理进程。 在大多数时间进行的标准进程中，不会使用 DBGIER 寄存器。 当 28x 处于实时仿真模式且 CPU 停机时，将使用另一个进程。 在此特殊情况下，将使用 DBGIER 且忽略 INTM 位。 如果 DSP 处于实时模式且 CPU 正在运行，则将使用标准中断处理进程。

表 6-1. 启用中断

中断处理进程	启用中断，如果...
标准	INTM = 0 且 IER 中的位为 1

表 6-1. 启用中断(接上表)

中断处理进程	启用中断，如果...
DSP 处于实时模式且停机	IER 中的位为 1 且 DBGIER 为 1

CPU 然后准备为中断提供服务。 *TMS320C28x DSP CPU 指令集参考指南*（文献编号 SPRU430）中详细描述了此准备进程。 在准备期间，将清除相应的 CPU IFR 和 IER 位，清除 EALLOW 和 LOOP，设置 INTM 和 DBGM，清理管道且存储返回地址，并执行自动背景保存。 然后从 PIE 模块提取 ISR 的矢量。 如果中断请求来自多路复用中断，则 PIE 模块使用组 PIEIERx 和 PIEIFRx 寄存器来解码需要服务的中断。 第 第 6.3.3 部分。

直接从 PIE 中断矢量表提取执行的中断服务例程的地址。 PIE 内可用的 96 个中断每个都有一个 32 位的矢量。 当提取中断矢量时，会自动清除 PIE 模块内的中断标志 (PIEIFRx.y)。 但是，当准备好接收来自 PIE 组的多个中断时，必须手动清除给定中断组的 PIE 确认位。

6.2 矢量表映射

在 28xx 器件上，中断矢量表可以映射到存储器中 5 个不同的位置。 实际上，只将 PIE 矢量表映射用于 280x 器件。

此矢量映射由下列模式位/信号控制：

VMAP: VMAP 位于状态寄存器 1 ST1（位 3）。 器件复位将此位设置为 1。 可以通过写入 ST1 或 SETC/CLRC VMAP 指令修改此位的状态。 对于正常操作，请保留此位被设置。

M0M1MAP: M0M1MAP 位于状态寄存器 1 ST1（位 11）。 器件复位将此位设置为 1。 可以通过写入 ST1 或 SETC/CLRC M0M1MAP 指令修改此位的状态。 对于正常的 28xx 正常操作，应保留此位被设置。 M0M1MAP = 0 保留仅供 TI 测试使用。

ENPIE: ENPIE 位于 PIECTRL 寄存器（位 0）。 在复位时此位的默认值设置为 0（禁用 PIE）。 在复位之内可以通过写入 PIECTRL 寄存器（地址 0x0000 0CE0）修改此位的状态。

使用这些位和信号时，可能的矢量表如表 6-2。

表 6-2. 中断矢量表映射

矢量 MAPS	矢量提取来源	地址范围	VMAP	M0M1MAP	ENPIE
M1 矢量(1)	M1 SARAM 块	0x000000-0x00003F	0	0	X
M0 矢量(1)	M0 SARAM 块	0x000000-0x00003F	0	1	X
BROM 矢量	引导 ROM 块	0x3FFFC0-0x3FFFFF	1	X	0
PIE 矢量	PIE 块	0x000D00-0x000DFF	1	X	1

(1) 矢量映射 M0 和 M1 矢量仅供保留模式使用。 在 28x 器件上它们用作 SARAM。

M1 和 M0 矢量表映射保留仅供 TI 测试使用。 当使用其它矢量映射时，M0 和 M1 存储器块作为 SARAM 块处理且可以不受任何限制自由使用。

在器件复位操作之后，矢量表的映射将如表 6-3。

表 6-3. 复位操作之后的矢量表映射

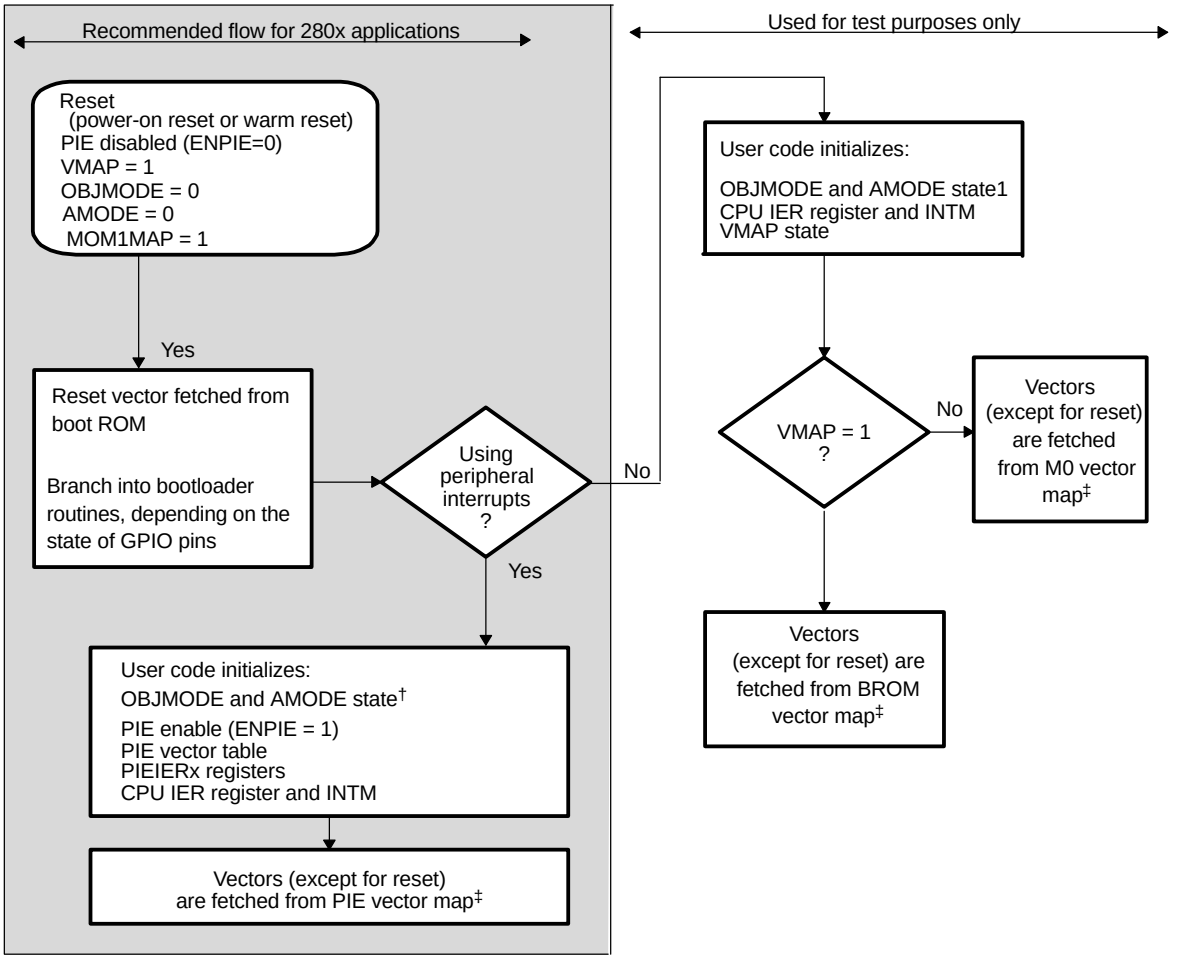
矢量 MAPS	复位提取来源	地址范围	VMAP (1)	M0M1MAP (1)	ENPIE (1)
BROM 矢量 (2)	引导 ROM 块	0x3FFFC0-0x3FFFFF	1	1	0

(1) 在 28x 器件上，VMAP 和 M0M1MAP 模式在复位时设置为 1。 复位时，ENPIE 模式强制为 0。
(2) 始终从引导 ROM 提取复位矢量。

在复位和引导完成之后，应由用户的代码初始化 PIE 矢量表。 然后，应用程序启用 PIE 矢量表。 从那时开始从 PIE 矢量表提取中断矢量。 注：当发生复位时，始终从矢量表提取复位矢量，如表 6-3。 复位后，始终禁用 PIE 矢量表。

图 6-3 阐述了选择矢量表映射的进程。

图 6-3. 复位流程图



A 28x CPU 的兼容性操作模式由状态寄存器 1 (ST1) 中的 OBJMODE 和 AMODE 位的组合确定：

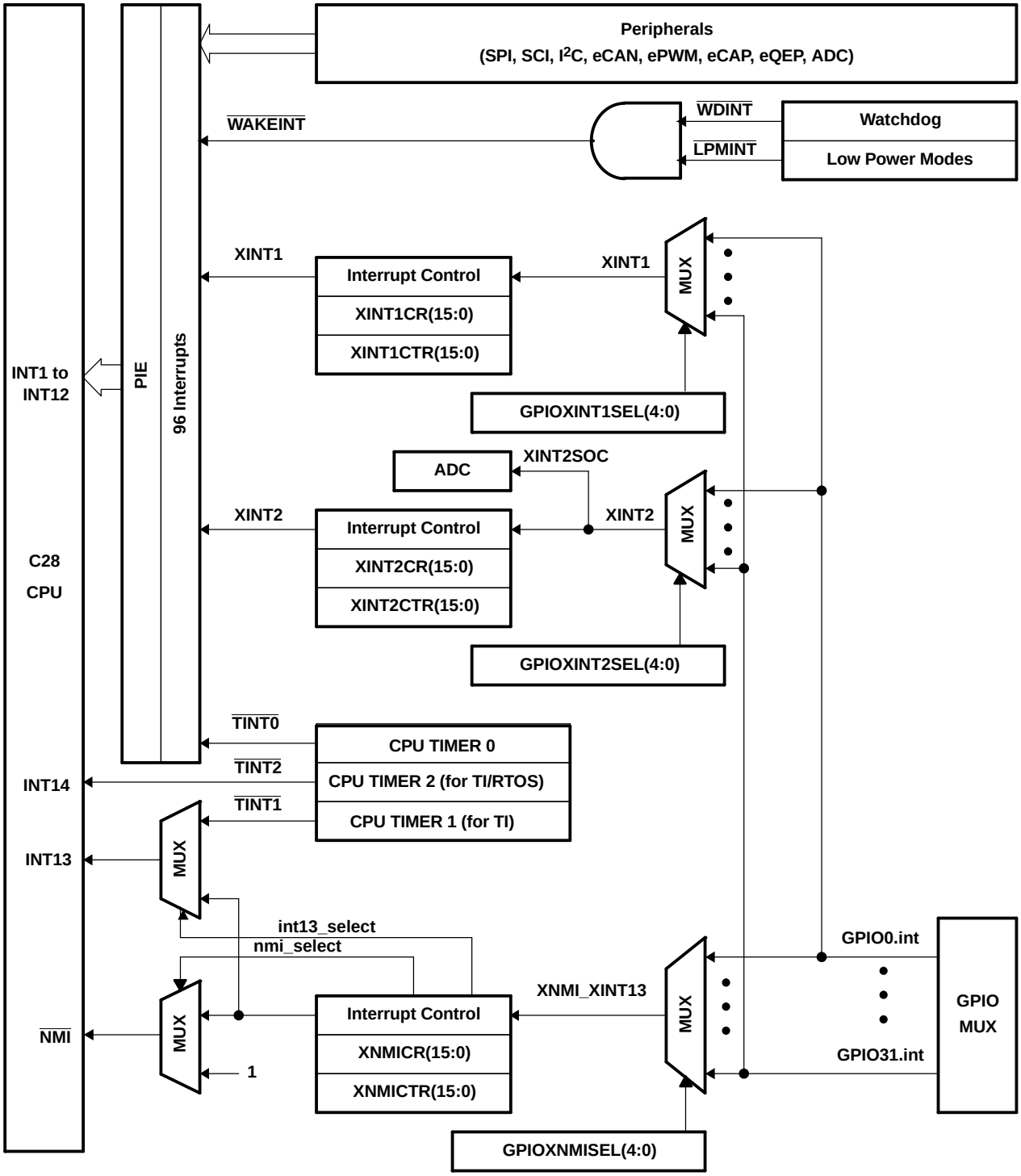
操作模式	OBJMODE	AMODE	
C28x 模式	1	0	
C2xLP 源兼容	1	1	
C27x 目标兼容	0	0	（复位时的默认值）

B 始终从引导 ROM 提取复位矢量。

6.3 中断源

图 6-4 显示 280x 器件内的各种中断源如何多路复用。 在所有 28x 器件上，此多路复用方案可能不完全相同。 有关详细信息，请参阅特定器件的数据手册。

图 6-4. 外部和 PIE 中断源



A 在 GPIO MUX 中，XINT1、XINT2 和 XNMI 信号同步，并且可以选择由用户可编程时钟周期数目限定。 这可以过滤出来自输入源的假信号。 有关详细信息，请参阅 GPIO MUX 一节。

6.3.1 处理多路复用中断的规范

PIE 模块将 8 个外设和外部引脚中断多路复用为一个 CPU 中断。 这些中断分为 12 组：PIE 组 1 - PIE 组 12。 每组均具有关联的启用 PIEIER 和标志 PIEIFR 寄存器。 这些寄存器用于控制中断到 CPU 的流动。 PIE 模块还使用 PIEIER 和 PIEIFR 寄存器来解码 CPU 应转移到的中断服务例程。

当清除 PIEIFR 和 PIEIER 寄存器内的位时，应遵循以下三个主要规则：

规则 1：切勿通过软件清除 PIEIFR 位

当发生对 PIEIFR 寄存器的写入或读-修改-写操作时，可能丢失引入的中断。 要清除 PIEIFR 位，必须为暂挂中断提供服务。 如果想要清除 PIEIFR 位而不执行正常的服务例程，请遵循以下规范：

1. 设置 EALLOW 位以允许修改 PIE 矢量表。
2. 修改 PIE 矢量表以便外设服务例程的矢量指向临时 ISR。 此临时 ISR 将只执行从中断 (IRET) 操作的返回。
3. 启用中断以便将由临时 ISR 对中断提供服务。
4. 为临时中断例程提供服务之后，将清除 PIEIFR 位。
5. 修改 PIE 矢量表以将外设的服务例程重新映射到正确的服务例程。
6. 清除 EALLOW 位。

规则 2：通过软件划分中断优先级的规范

请使用*用 C 语言编写的 C280x C/C++ 头文件和外设示例*（文献编号 SPRC191）中介绍的方法。

a. 使用 CPU IER 寄存器作为全局优先级，使用各个 PIEIER 寄存器作为组优先级。 在此事例中，仅在中断内修改 PIEIER 寄存器。 另外，仅修改了与所服务中断相同的组的 PIEIER。 此修改是在 PIEACK 位保持从 CPU 返回的其它中断时进行的。

b. 对来自非相关组的中断提供服务时，切勿禁用组的 PIEIER 位。

规则 3：使用 PIEIER 禁用中断

如果 PIEIER 寄存器用于启用然后禁用中断，则必须遵循第 6.3.2 部分 中所述的规范。

6.3.2 启用和禁用多路复用外设中断的规范

启用或禁用中断的正确规范是通过使用外设中断启用/禁用标志完成的。 PIEIER 和 CPU IER 寄存器的主要用途是通过软件划分同一 PIE 中断组内的中断的优先级。 软件包*用 C 语言编写的 C280x C/C++ 头文件和外设示例*（文献编号 SPRC191）中包括了阐述这种通过软件划分中断优先级的方法的示例。

如果需要在此背景之外清除 PIEIER 寄存器内的位，则应遵循以下两个规范。 第一种方法保留关联的 PIE 标志寄存器以便不丢失中断。 第二种方法清除关联的 PIE 标志寄存器。

方法 1：使用 PIEIERx 寄存器以禁用中断并保留关联的 PIEIFRx 标志。

要清除 PIEIERx 寄存器内的位而同时保留 PIEIFRx 寄存器中的相关标志，应遵循以下规范：

a. 禁用全局中断 (INTM = 1)。
b. 清除 PIEIERx.y 位以禁用给定外设的中断。 可以为同一组内的一个或多个外设执行此操作。
c. 等待 5 个周期。 此延迟是必需的，它能确保在 CPU IFR 寄存器内为正在进入 CPU 的任何中断设置了标志。
d. 清除外设组的 CPU IFRx 位。 这是 CPU IFR 寄存器上的安全操作。
e. 清除外设组的 PIEACKx 位。
f. 启用全局中断 (INTM = 0)。

方法 2：使用 PIEIERx 寄存器以禁用中断并清除关联的 PIEIFRx 标志。

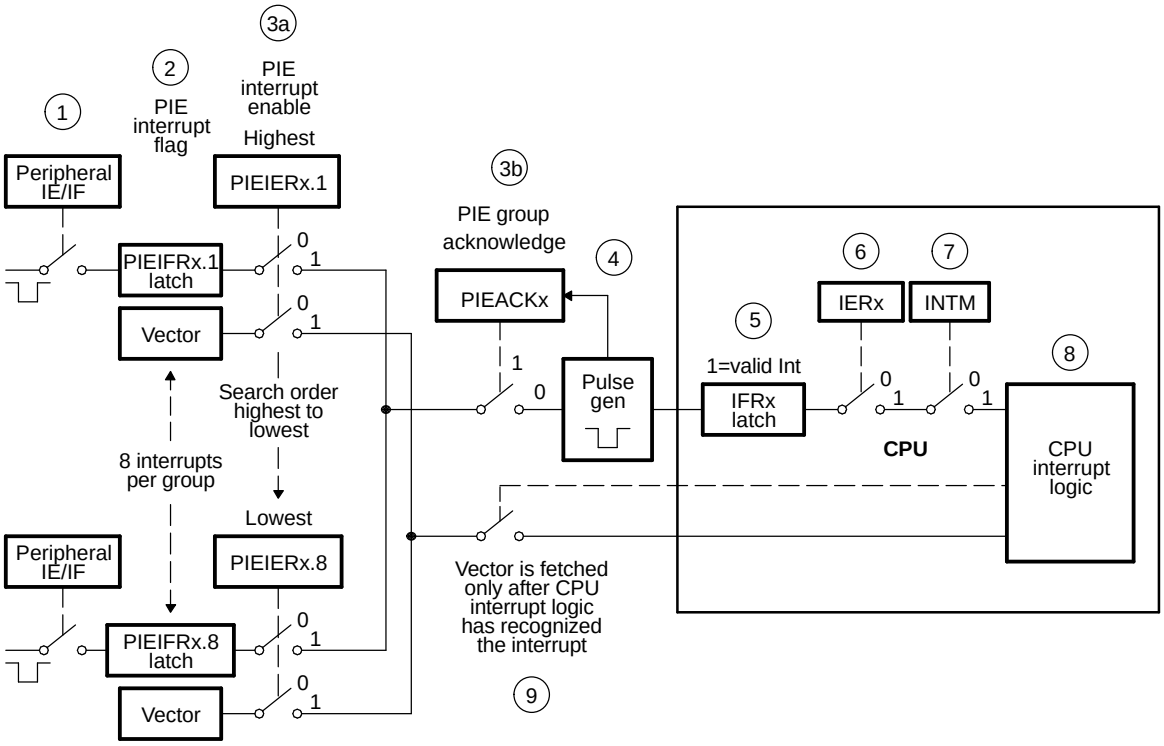
要执行外设中断的软件复位并清除 PIEIFRx 寄存器和 CPU IFR 寄存器中的关联标志，应遵循以下规范：

1. 禁用全局中断 (INTM = 1)。
2. 设置 EALLOW 位。
3. 修改 PIE 矢量表以便将特定外设中断的矢量临时映射到空的中断服务例程 (ISR)。 此空的 ISR 将只执行从中断 (IRET) 操作的返回。 这是清除单个 PIEIFRx.y 位而不丢失任何来自组内其它外设的中断的安全方法。
4. 在外设寄存器上禁用外设中断。
5. 启用全局中断 (INTM = 0)。
6. 等待空的 ISR 例程对来自外设的任何暂挂中断提供服务。
7. 禁用全局中断 (INTM = 1)。
8. 修改 PIE 矢量表以将外设矢量表映射回其原始 ISR。
9. 清除 EALLOW 位。
10. 清除给定外设的 PIEIER 位。
11. 清除给定外设组的 IFR 位（这是 CPU IFR 寄存器上的安全操作）。
12. 清除 PIE 组的 PIEACK 位。
13. 启用全局中断。

6.3.3 从外设到 CPU 的多路复用中断请求流程

图 6-5 显示了流程，其中的步骤用带圆圈的编号表示。 在图的后面对步骤进行了描述。

图 6-5. 多路复用的中断请求流程图



步骤 1：

PIE 组内的任何外设或外部中断生成中断。 如果在外设模块内启用了这些中断，则向 PIE 模块发送中断请求。

步骤 2：

PIE 模块识别 PIE 组 x 内的中断 y (INTx.y) 已发出中断且已锁定适当的 PIE 中断标志位：PIEIFRx.y = 1。

步骤 3：

要将中断请求从 PIE 发送到 CPU，以下两个条件必须同时为真：

3a：必须设置了正确的启用位 (PIEIERx.y = 1) 且

3b：该组 的 PIEACKx 位必须已清除。

步骤 4：

如果 3a 和 3b 中的两个条件同时为真，则中断请求被发送到 CPU 且再次设置确认位 (PIEACKx = 1)。 PIEACKx 位将保持设置，直到您清除它以指示可以将来自该组的其它中断从 PIE 发送到 CPU。

步骤 5：

设置 CPU 中断标志位 (CPU IFRx = 1) 以指示 CPU 级别的暂挂中断 x。

步骤 6 和步骤 7：

如果已启用该 CPU 中断（CPU IER 位 x = 1 或 DBGIER 位 x = 1）且全局中断屏蔽已清除 (INTM = 0)，则 CPU 将为 INTx 提供服务

步骤 8：

CPU 识别中断，然后执行自动背景保存、清除 IER 位、设置 INTM 并清除 EALLOW。 *TMS320C28x DSP CPU 和指令集参考指南*（文献编号 SPRU430）中描述了 CPU 准备服务中断时所执行的所有步骤。

步骤 9：

CPU 然后将向 PIE 请求合适的变量。 对于多路复用中断，PIE 模块使用 PIEIERx 和 PIEIFRx 寄存器中的当前值来解码应使用哪个矢量地址。 可能有以下两种情况：

a. 组内既在 PIEIERx 寄存器中启用且在 PIEIFRx 中标志为暂挂的具有最高优先级的中断的向量被提取且用作转移地址。 在这种方式中，如果在步骤 7 之后为用更高优先级的启用的中断设置了标志，是将首先使用该中断。

b. 如果没有启用组内带有标志的中断，则 PIE 将用组内最高优先级的中断的矢量响应。 那是用于 INTx.1 的转移地址。 此行为对应 28x TRAP 或 INT 指令。

注:

因为 PIEIERx 寄存器用于确定哪个矢量将用于分支，所以当清除 PIEIERx 寄存器内的位时务必小心。 第 第 6.3.2 部分。 未按照这些步骤可能导致在图 6-5 中第 5 步中断传递到 CPU 之后 PIEIERx 寄存器发生更改。 在这种情况下，除非有其它暂挂且启用的中 断，否则 PIE 将象执行了 TRAP 或 INT 指令那样响应。

这时，PIEIFRx.y 位被清除，且 CPU 转移到从 PIE 提取的中断矢量。

6.3.4 PIE 矢量表

PIE 矢量表（请参阅表 6-5）由 256 x 16 的 SARAM 块组成，如果未使用 PIE 模块，该块也可用作 RAM（仅在数据空间中）。 复位时，PIE 矢量表内容未定义。 CPU 固定了 INT1 到 INT12 的中断优先级。 PIE 控制每组 8 个中断的优先级。 例如，如果 INT1.1 与 INT8.1 同时出现，两个中断由 PIE 模块同时提供给 CPU，则 CPU 先给 INT1.1 提供服务。 如果 INT1.1 与 INT1.8 同时出现，则 INT1.1 先发送给 CPU，INT1.8 紧随其后。 中断优先级划分是在中断处理的矢量提取期间执行的。

TRAP 1 到 TRAP 12 指令或 INTR INT1 到 INTR INT12 指令从每组的第一个位置（INTR1.1 到 INT12.1）提取矢量。 与 OR IFR 类似，如果设置了相应的中断标志，则 #16 位操作导致从 INTR1.1 到 INTR12.1 位置提取矢量。 所有其它 TRAP、INTR、OR IFR 和 #16 位操作从相应的表位置提取矢量。 应避免对 INTR1 到 INTR12 执行这样的操作。 TRAP #0 操作返回矢量值 0x000000。矢量表受 EALLOW 保护。

在表 6-4 中的 96 个可能的多路复用中断中，当前使用了 43 个中断。 其余中断保留供未来的器件使用。 如果在 PIEIFRx 级别启用了这些保留的中断，则可将它们用作软件中断，前提是组内没有任何中断被外设使用。 否则，如果在修改 PIEIFR 时意外清除了来自外设的中断的标志，则可能丢失这些中断。

总之，在以下两种安全的情况下，保留的中断可用作软件中断：

1. 组内没有外设发出中断。
2. 未给该组指定外设中断。 例如，PIE 组 11 和 12 没有连接任何外设。

表 6-4 中显示了对连接到 PIE 模块的外设和外设中断进行的中断分组。 表中的每行显示多路复用为特定 CPU 中断的 8 个中断。 整个 PIE 矢量表，包括多路复用和非多路复用中断，如 表 6-5 所示。

表 6-4. 280x PIE 多路复用的外设中断矢量表

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1.y	WAKEINT	TINT0	ADCINT	XINT2	XINT1	保留	SEQ2INT	SEQ1INT
	LPM/WD	TIMER.0	(ADC)			-	(ADC)	(ADC)
	0xD4E	0xD4C	0xD4A	0xD48	0xD46	0xD44	0xD42	0xD40
INT2.y	保留	保留	EPWM6_ TZINT	EPWM5_TZINT	EPWM4_TZINT	EPWM3_TZINT	EPWM2_TZINT	EPWM1_TZINT
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD5E	0xD5C	0xD5A	0xD58	0xD56	0xD54	0xD52	0xD50
INT3.y	保留	保留	EPWM6_INT	EPWM5_INT	EPWM4_INT	EPWM3_INT	EPWM2_INT	EPWM1_INT
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD6E	0xD6C	0xD6A	0xD68	0xD66	0xD64	0xD62	0xD60
INT4.y	保留	保留	保留	保留	ECAP4_INT	ECAP3_INT	ECAP2_INT	ECAP1_INT
	-	-	-	-	(eCAP4)	(eCAP3)	(eCAP2)	(eCAP1)
	0xD7E	0xD7C	0xD7A	0xD78	0xD76	0xD74	0xD72	0xD70
INT5.y	保留	保留	保留	保留	保留	保留	EQEP2_INT	EQEP1_INT
	-	-	-	-	-	-	(eQEP2)	(eQEP1)
	0xD8E	0xD8C	0xD8A	0xD88	0xD86	0xD84	0xD82	0xD80
INT6.y	SPITXINTD	SPIRXINTD	SPITXINTC	SPIRXINTC	SPITXINTB	SPIRXINTB	SPITXINTA	SPIRXINTA
	(SPI-D)	(SPI-D)	(SPI-C)	(SPI-C)	(SPI-B)	(SPI-B)	(SPI-A)	(SPI-A)
	0xD9E	0xD9C	0xD9A	0xD98	0xD96	0xD94	0xD92	0xD90
INT7.y	保留	保留	保留	保留	保留	保留	保留	保留
	0xDAE	0xDAC	0xDAA	0xDA8	0xDA6	0xDA4	0xDA2	0xDA0
INT8.y	保留	保留	保留	保留	保留	保留	I2CINT1A	I2CINT2A
	-	-	-	-	-	-	(I2C-A)	(I2C-A)
	0xDBE	0xDBC	0xDBA	0xDB8	0xDB6	0xDB4	0xDB2	0xDB0
INT9.y	ECAN1INTB	ECAN0INTB	ECAN1INTA	ECAN0INTA	SCITXINTB	SCIRXINTB	SCITXINTA	SCIRXINTA
	(CAN-B)	(CAN-B)	(CAN-A)	(CAN-A)	(SCI-B)	(SCI-B)	(SCI-A)	(SCI-A)
	0xDCE	0xDCC	0xDCA	0xDC8	0xDC6	0xDC4	0xDC2	0xDC0
INT10.y	保留	保留	保留	保留	保留	保留	保留	保留
	0xDDE	0xDDC	0xDDA	0xDD8	0xDD6	0xDD4	0xDD2	0xDD0
INT11.y	保留	保留	保留	保留	保留	保留	保留	保留
	0xDEE	0xDEC	0xDEA	0xDE8	0xDE6	0xDE4	0xDE2	0xDE0
INT12.y	保留	保留	保留	保留	保留	保留	保留	保留
	0xDFE	0xDFC	0xDFA	0xDF8	0xDF6	0xDF4	0xDF2	0xDF0

表 6-5. 280x PIE 矢量表

名称	VECTOR ID[1]	地址[2]	大小 (x16)	说明[3]	CPU 优先级	PIE 组优先级
复位	0	0x0000 0D00	2	始终从引导 ROM 中的位置 0x003F FFC0 提取复位。	1（最高）	-
INT1	1	0x0000 0D02	2	未使用。 请参阅 PIE 组 1	5	-
INT2	2	0x0000 0D04	2	未使用。 请参阅 PIE 组 2	6	-
INT3	3	0x0000 0D06	2	未使用。 请参阅 PIE 组 3	7	-
INT4	4	0x0000 0D08	2	未使用。 请参阅 PIE 组 4	8	-
INT5	5	0x0000 0D0A	2	未使用。 请参阅 PIE 组 5	9	-
INT6	6	0x0000 0D0C	2	未使用。 请参阅 PIE 组 6	10	-
INT7	7	0x0000 0D0E	2	未使用。 请参阅 PIE 组 7	11	-
INT8	8	0x0000 0D10	2	未使用。 请参阅 PIE 组 8	12	-
INT9	9	0x0000 0D12	2	未使用。 请参阅 PIE 组 9	13	-
INT10	10	0x0000 0D14	2	未使用。 请参阅 PIE 组 10	14	-
INT11	11	0x0000 0D16	2	未使用。 请参阅 PIE 组 11	15	-
INT12	12	0x0000 0D18	2	未使用。 请参阅 PIE 组 12	16	-
INT13	13	0x0000 0D1A	2	外部中断 13 (XINT13) 或 CPU 定时器 1（供 TI/RTOS 使用）[4]	17	-
INT14	14	0x0000 0D1C	2	CPU 定时器 2（供 TI/RTOS 使用）	18	-
DATALOG	15	0x0000 0D1E	2	CPU 数据记录中断	19（最低）	-
RTOSINT	16	0x0000 0D20	2	CPU 实时操作系统中断	4	-
EMUINT	17	0x0000 0D22	2	CPU 仿真中断	2	-
NMI	18	0x0000 0D24	2	外部不可屏蔽中断	3	-
ILLEGAL	19	0x0000 0D26	2	非法操作	-	-
USER1	20	0x0000 0D28	2	用户定义的陷阱	-	-
USER2	21	0x0000 0D2A	2	用户定义的陷阱	-	-
USER3	22	0x0000 0D2C	2	用户定义的陷阱	-	-
USER4	23	0x0000 0D2E	2	用户定义的陷阱	-	-
USER5	24	0x0000 0D30	2	用户定义的陷阱	-	-
USER6	25	0x0000 0D32	2	用户定义的陷阱	-	-
USER7	26	0x0000 0D34	2	用户定义的陷阱	-	-
USER8	27 号	0x0000 0D36	2	用户定义的陷阱	-	-
USER9	28	0x0000 0D38	2	用户定义的陷阱	-	-

(1) 该矢量 ID 供 DSP/BIOS 使用。
(2) 始终从引导 ROM 中的位置 0x003F FFC0 提取复位。
(3) PIE 矢量表内的所有位置受 EALLOW 保护。
(4) CPU 定时器 1 保留供 TI 软件使用。 但是中断 XINT13 可以由客户应用程序自由使用。

表 6-5. 280x PIE 矢量表(接上表)

名称	VECTOR ID[1]	地址[2]	大小 (x16)	说明[3]		CPU 优先级	PIE 组优先级
USER10	29	0x0000 0D3A	2	用户定义的陷阱		-	-
USER11	30	0x0000 0D3C	2	用户定义的陷阱		-	-
USER12	31	0x0000 0D3E	2	用户定义的陷阱		-	-
PIE 组 1 矢量 - 多路复用为 CPU INT1							
INT1.1	32	0x0000 0D40	2	SEQ1INT	(ADC)	5	1（最高）
INT1.2	33	0x0000 0D42	2	SEQ2INT	(ADC)	5	2
INT1.3	34	0x0000 0D44	2	保留		5	3
INT1.4	35	0x0000 0D46	2	XINT1		5	4
INT1.5	36	0x0000 0D48	2	XINT2		5	5
INT1.6	37	0x0000 0D4A	2	ADCINT	(ADC)	5	6
INT1.7	38	0x0000 0D4C	2	TINT0	（CPU-定时器 0）	5	7
INT1.8	39	0x0000 0D4E	2	WAKEINT	LPM/WD	5	8（最低）
PIE 组 2 矢量 - 多路复用为 CPU INT2							
INT2.1	40	0x0000 0D50	2	EPWM1_TZINT	(EPWM1)	6	1（最高）
INT2.2	41	0x0000 0D52	2	EPWM2_TZINT	(EPWM2)	6	2
INT2.3	42	0x0000 0D54	2	EPWM3_TZINT	(EPWM3)	6	3
INT2.4	43	0x0000 0D56	2	EPWM4_TZINT	(EPWM4)	6	4
INT2.5	44	0x0000 0D58	2	EPWM5_TZINT	(EPWM5)	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6_TZINT	(EPWM6)	6	6
INT2.7	46	0x0000 0D5C	2	保留		6	7
INT2.8	47	0x0000 0D5E	2	保留		6	8（最低）
PIE 组 3 矢量 - 多路复用为 CPU INT3							
INT3.1	48	0x0000 0D60	2	EPWM1_INT	(EPWM1)	7	1（最高）
INT3.2	49	0x0000 0D62	2	EPWM2_INT	(EPWM2)	7	2
INT3.3	50	0x0000 0D64	2	EPWM3_INT	(EPWM3)	7	3
INT3.4	51	0x0000 0D66	2	EPWM4_INT	(EPWM4)	7	4
INT3.5	52	0x0000 0D68	2	EPWM5_INT	(EPWM5)	7	5
INT3.6	53	0x0000 0D6A	2	EPWM6_INT	(EPWM6)	7	6
INT3.7	54	0x0000 0D6C	2	保留		7	7
INT3.8	55	0x0000 0D6E	2	保留		7	8（最低）
PIE 组 4 矢量 - 多路复用为 CPU INT4							
INT4.1	56	0x0000 0D70	2	ECAP1_INT	(ECAP1)	8	1（最高）
INT4.2	57	0x0000 0D72	2	ECAP2_INT	(ECAP2)	8	2

表 6-5. 280x PIE 矢量表(接上表)

名称	VECTOR ID[1]	地址[2]	大小 (x16)	说明[3]		CPU 优先级	PIE 组优先级
INT4.3	58	0x0000 0D74	2	ECAP3_INT	(ECAP3)	8	3
INT4.4	59	0x0000 0D76	2	ECAP4_INT	(ECAP4)	8	4
INT4.5	60	0x0000 0D78	2	保留		8	5
INT4.6	61	0x0000 0D7A	2	保留		8	6
INT4.7	62	0x0000 0D7C	2	保留		8	7
INT4.8	63	0x0000 0D7E	2	保留		8	8（最低）
PIE 组 5 矢量 - 多路复用为 CPU INT5							
INT5.1	64	0x0000 0D80	2	EQEP1_INT	(EQEP1)	9	1（最高）
INT5.2	65	0x0000 0D82	2	EQEP1_INT	(EQEP2)	9	2
INT5.3	66	0x0000 0D84	2	保留		9	3
INT5.4	67	0x0000 0D86	2	保留		9	4
INT5.5	68	0x0000 0D88	2	保留		9	5
INT5.6	69	0x0000 0D8A	2	保留		9	6
INT5.7	70	0x0000 0D8C	2	保留		9	7
INT5.8	71	0x0000 0D8E	2	保留		9	8（最低）
PIE 组 6 矢量 - 多路复用为 CPU INT6							
INT6.1	72	0x0000 0D90	2	SPIRXINTA	(SPI-A)	10	1（最高）
INT6.2	73	0x0000 0D92	2	SPITXINTA	(SPI-A)	10	2
INT6.3	74	0x0000 0D94	2	SPIRXINTB	(SPI-B)	10	3
INT6.4	75	0x0000 0D96	2	SPITXINTB	(SPI-B)	10	4
INT6.5	76	0x0000 0D98	2	SPIRXINTC	(SPI-C)	10	5
INT6.6	77	0x0000 0D9A	2	SPITXINTC	(SPI-C)	10	6
INT6.7	78	0x0000 0D9C	2	SPIRXINTD	(SPI-D)	10	7
INT6.8	79	0x0000 0D9E	2	SPITXINTD	(SPI-D)	10	8（最低）
PIE 组 7 矢量 - 多路复用为 CPU INT7							
INT7.1	80	0x0000 0DA0	2	保留		11	1（最高）
INT7.2	81	0x0000 0DA2	2	保留		11	2
INT7.3	82	0x0000 0DA4	2	保留		11	3
INT7.4	83	0x0000 0DA6	2	保留		11	4
INT7.5	84	0x0000 0DA8	2	保留		11	5
INT7.6	85	0x0000 0DAA	2	保留		11	6
INT7.7	86	0x0000 0DAC	2	保留		11	7
INT7.8	87	0x0000 0DAE	2	保留		11	8（最低）
PIE 组 8 矢量 - 多路复用为 CPU INT8							

表 6-5. 280x PIE 矢量表(接上表)

名称	VECTOR ID[1]	地址[2]	大小 (x16)	说明[3]		CPU 优先级	PIE 组优先级
INT8.1	88	0x0000 0DB0	2	I2CINT1A	I2C-A	12	1（最高）
INT8.2	89	0x0000 0DB2	2	I2CINT2A	I2C-A	12	2
INT8.3	90	0x0000 0DB4	2	保留		12	3
INT8.4	91	0x0000 0DB6	2	保留		12	4
INT8.5	92	0x0000 0DB8	2	保留		12	5
INT8.6	93	0x0000 0DBA	2	保留		12	6
INT8.7	94	0x0000 0DBC	2	保留		12	7
INT8.8	95	0x0000 0DBE	2	保留		12	8（最低）
PIE 组 9 矢量 - 多路复用为 CPU INT9							
INT9.1	96	0x0000 0DC0	2	SCIRXINTA	(SCI-A)	13	1（最高）
INT9.2	97	0x0000 0DC2	2	SCITXINTA	(SCI-A)	13	2
INT9.3	98	0x0000 0DC4	2	SCIRXINTB	(SCI-B)	13	3
INT9.4	99	0x0000 0DC6	2	SCITXINTB	(SCI-B)	13	4
INT9.5	100	0x0000 0DC8	2	ECAN0INTA	(eCAN-A)	13	5
INT9.6	101	0x0000 0DCA	2	ECAN1INTA	(eCAN-A)	13	6
INT9.7	102	0x0000 0DCC	2	ECAN0INTA	(eCAN-B)	13	7
INT9.8	103	0x0000 0DCE	2	ECAN1INTA	(eCAN-B)	13	8（最低）
PIE 组 10 矢量 - 多路复用为 CPU INT10							
INT10.1	104	0x0000 0DD0	2	保留		14	1（最高）
INT10.2	105	0x0000 0DD2	2	保留		14	2
INT10.3	106	0x0000 0DD4	2	保留		14	3
INT10.4	107	0x0000 0DD6	2	保留		14	4
INT10.5	108	0x0000 0DD8	2	保留		14	5
INT10.6	109	0x0000 0DDA	2	保留		14	6
INT10.7	110	0x0000 0DDC	2	保留		14	7
INT10.8	111	0x0000 0DDE	2	保留		14	8（最低）
PIE 组 11 矢量 - 多路复用为 CPU INT11							
INT11.1	112	0x0000 0DE0	2	保留		15	1（最高）
INT11.2	113	0x0000 0DE2	2	保留		15	2
INT11.3	114	0x0000 0DE4	2	保留		15	3
INT11.4	115	0x0000 0DE6	2	保留		15	4
INT11.5	116	0x0000 0DE8	2	保留		15	5
INT11.6	117	0x0000 0DEA	2	保留		15	6
INT11.7	118	0x0000 0DEC	2	保留		15	7

表 6-5. 280x PIE 矢量表(接上表)

名称	VECTOR ID[1]	地址[2]	大小 (x16)	说明[3]	CPU 优先级	PIE 组优先级
INT11.8	119	0x0000 0DEE	2	保留	15	8（最低）
PIE 组 12 矢量 - 多路复用为 INT12 INT11						
INT12.1	120	0x0000 0DF0	2	保留	16	1（最高）
INT12.2	121	0x0000 0DF2	2	保留	16	2
INT12.3	122	0x0000 0DF4	2	保留	16	3
INT12.4	123	0x0000 0DF6	2	保留	16	4
INT12.5	124	0x0000 0DF8	2	保留	16	5
INT12.6	125	0x0000 0DFA	2	保留	16	6
INT12.7	126	0x0000 0DFC	2	保留	16	7
INT12.8	127	0x0000 0DFE	2	保留	16	8（最低）

6.4 PIE 配置寄存器

控制 PIE 模块的功能的寄存器如表 6-6。

表 6-6. PIE 配置和控制寄存器

名称	地址	大小 (x16)	说明
PIECTRL	0x0000-0CE0	1	PIE 控制寄存器
PIEACK	0x0000-0CE1	1	PIE，确认寄存器
PIEIER1	0x0000-0CE2	1	PIE，INT1 组启用寄存器
PIEIFR1	0x0000-0CE3	1	PIE，INT1 组标志寄存器
PIEIER2	0x0000-0CE4	1	PIE，INT2 组启用寄存器
PIEIFR2	0x0000-0CE5	1	PIE，INT2 组标志寄存器
PIEIER3	0x0000-0CE6	1	PIE，INT3 组启用寄存器
PIEIFR3	0x0000-0CE7	1	PIE，INT3 组标志寄存器
PIEIER4	0x0000-0CE8	1	PIE，INT4 组启用寄存器
PIEIFR4	0x0000-0CE9	1	PIE，INT4 组标志寄存器
PIEIER5	0x0000-0CEA	1	PIE，INT5 组启用寄存器
PIEIFR5	0x0000-0CEB	1	PIE，INT5 组标志寄存器
PIEIER6	0x0000-0CEC	1	PIE，INT6 组启用寄存器
PIEIFR6	0x0000-0CED	1	PIE，INT6 组标志寄存器
PIEIER7	0x0000-0CEE	1	PIE，INT7 组启用寄存器
PIEIFR7	0x0000-0CEF	1	PIE，INT7 组标志寄存器
PIEIER8	0x0000-0CF0	1	PIE，INT8 组启用寄存器
PIEIFR8	0x0000-0CF1	1	PIE，INT8 组标志寄存器
PIEIER9	0x0000-0CF2	1	PIE，INT9 组启用寄存器
PIEIFR9	0x0000-0CF3	1	PIE，INT9 组标志寄存器
PIEIER10	0x0000-0CF4	1	PIE，INT10 组启用寄存器
PIEIFR10	0x0000-0CF5	1	PIE，INT10 组标志寄存器
PIEIER11	0x0000-0CF6	1	PIE，INT11 组启用寄存器
PIEIFR11	0x0000-0CF7	1	PIE，INT11 组标志寄存器
PIEIER12	0x0000-0CF8	1	PIE，INT12 组启用寄存器
PIEIFR12	0x0000-0CF9	1	PIE，INT12 组标志寄存器

6.5 PIE 中断寄存器

图 6-6. PIECTRL 寄存器（地址 CE0）

15	1	0
PIEVECT		ENPIE
R-0		R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-7. PIECTRL 寄存器地址字段说明

位	字段	值	说明
15-1	PIEVECT		这些位指示 PIE 矢量表中提取矢量的地址。 忽略地址的最低位且仅显示地址的位 1 至 15。 可以读取矢量值以确认产生了矢量提取的中断。 **例如：**如果 PIECTRL = 0x0D27，则提取了地址 0x0D26 中的矢量（非法操作）。
0	ENPIE		启用从 PIE 矢量表提取矢量。 **注：** 即使复位矢量被启用，也不会从 PIE 提取。 该矢量将始终从引导 ROM 中提取。
		0	如果此位设置为 0，则将禁用 PIE 模块并从引导 ROM 中的 CPU 矢量表提取矢量。 即使在禁用 PIE 块时，也可以访问所有 PIE 块的寄存器（PIEACK、PIEIFR 和 PIEIER）。
		1	当 ENPIE 设置为 1 时，除复位之外的所有矢量均从 PIE 矢量表提取。 始终从引导 ROM 提取复位矢量。

图 6-7. PIE 中断确认寄存器 (PIEACK) 寄存器（地址 CE1）

15	12	11	0
保留		PIEACK	
R-0		R/W1C-1	

说明：R/W1C = 读/写入 1 以清除；R = 只读；-*n* = 复位后的值

表 6-8. PIE 中断确认寄存器 (PIEACK) 字段说明

位	字段	值	说明
15-12	保留		保留
11-0	PIEACK		PIEACK 中的每位对应特定的 PIE 组。 位 0 对应 PIE 组 2 中多路复用为 $\overline{\text{INT1}}$ 的中断，依此类推直到位 11，它对应多路复用为 CPU $\overline{\text{INT12}}$ 的 PIE 组 12
		位 x = 0 (1)	如果位读数为 0，则它指示 PIE 可以从相应的组发送中断到 CPU。 忽略 0 的写入。
		位 x = 1	读数为 1 指示来自相应组的中断已发送给 CPU，且当前阻止来自该组的所有其它中断。 如果该组有一个中断正暂挂，将 1 写入相应的中断位将清除该位并能使 PIE 模块驱动一个脉冲进入 CPU 中断输入。

(1) 位 x = PIEACK 位 0 - PIEACK 位 11。 位 0 对应 CPU $\overline{\text{INT1}}$，依此类推直到位 11，它对应 CPU $\overline{\text{INT12}}$

6.5.1 PIE 中断标志寄存器

有 12 个 PIEIFR 寄存器，每个对应 PIE 模块使用的一个 CPU 中断 (INT1-INT12)。

图 6-8. PIEIFRx 寄存器 (x = 1 - 12)

15	8
保留	
R-0	

7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-9. PIEIFRx 寄存器字段说明

位	字段	说明
15-8	保留	保留
7	INTx.8	这些寄存器位指示中断当前是否处于活动状态。 它们的工作方式与 CPU 中断标志寄存器非常相似。 当中断处于活动状态时，将设置相应的寄存器位。 当中断被服务或为寄存器位写入 0 时，该位将被清除。 也可以读取此寄存器以确定哪些中断正在活动或暂挂。 x = 1 - 12。 INTx 表示 CPU INT1 至 INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	在中断处理的中断矢量提取期间，PIEIFR 寄存器位被清除。
3	INTx.4	硬件对 PIEIFR 寄存器的存取优先级比 CPU 高。
2	INTx.3	
1	INTx.2	
0	INTx.1	

注:

切勿清除 PIEIFR 位。 在读 - 修改 - 写操作期间，可能丢失中断。 请参阅第 第 6.3.1 部分 节 以了解清除已设置标志的中断的方法。

6.5.2 PIE 中断启用寄存器

有 12 个 PIEIER 寄存器，每个对应 PIE 模块使用的一个 CPU 中断 (INT1-INT12)。

图 6-9. PIEIERx 寄存器 (x = 1 - 12)

15							8
保留							
R-0							
7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 6-10. PIEIERx 寄存器 (x = 1 - 12) 字段说明

位	字段	说明
15-8	保留	保留
7	INTx.8	这些寄存器位分别启用组内的中断，且工作方式与内核中断启用寄存器非常相似。 位设置为 1 将启用对相应中断的服务。 位设置为 0 将禁用对中断的服务。 x = 1 - 12。 INTx 表示 CPU INT1 至 INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

注:

在正常操作期间清除 PIEIER 位时务必小心谨慎。 请参阅第 第 6.3.2 部分 节 节以了解处理这些位的正确规范。

6.5.3 CPU 中断标志寄存器 (IFR)

CPU 中断标志寄存器 (IFR) 是 16 位的 CPU 寄存器，用于标识和清除暂挂的中断。 IFR 包含 CPU 级别的所有可屏蔽中断（INT1-INT14、DLOGINT 和 RTOSINT）的标志位。 当启用 PIE 时，PIE 模块多路复用 INT1-INT12 的中断源。

当请求可屏蔽中断时，相应的外设控制寄存器中的标志位设置为 1。 如果相应的屏蔽位也为 1，则中断请求发送至 CPU，并在 IFR 中设置相应的标志。 这指示该中断正暂挂或等待确认。

要标识暂挂的中断，请使用 ，然后测试堆栈上的值。 使用 OR IFR 指令设置 IFR 位并使用 AND IFR 指令手动清除暂挂的中断。 可以使用 AND IFR #0 指令或通过硬件复位清除所有暂挂的中断。

以下事件也会 清除 IFR 标志：

- CPU 确认了该中断。
- 28x 器件复位。

注：

1. 要清除 CPU IFR 位，必须给它写入 0 而不是 1。
2. 当可屏蔽中断被确认时，仅自动清除 IFR 位。 不清除相应外设控制寄存器中的标志位。 如果应用程序要求清除控制寄存器标志，则必须由软件清除该位。
3. 当 INTR 指令请求中断且设置了相应的 IFR 位时，CPU 不自动清除该位。 如果应用程序要求清除 IFR 位，则必须由软件清除该位。
4. IMR 和 IFR 寄存器与内核级别中断有关。 所有外设在各自的控制/配置寄存器中均有自己的中断屏蔽和标志位。 请注意，一个内核级别中断下分组了若干外设中断。

图 6-10. 中断标志寄存器 (IFR) — CPU 寄存器

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 6-11. 中断标志寄存器 (IFR) — CPU 寄存器字段说明

位	字段	值	说明
15	RTOSINT		实时操作系统标志。 RTOSINT 是 RTOS 中断的标志。
		0	没有暂挂的 RTOS 中断
		1	至少一个 RTOS 中断正暂挂。 将 0 写入此位以清零和清除中断请求
14	DLOGINT		数据记录中断标志。 DLOGINT 是数据记录中断的标志。
		0	没有暂挂的 DLOGINT。
		1	至少一个 DLOGINT 中断正暂挂。 将 0 写入此位以清零和清除中断请求
13	INT14		中断 14 标志。 INT14 是连接到 CPU 中断级别 INT14 的中断的标志。
		0	没有暂挂的 INT14 中断
		1	至少一个 INT14 中断正暂挂。 将 0 写入此位以清零和清除中断请求
12	INT13		中断 13 标志。 INT13 是连接到 CPU 中断级别 INT13I 的中断的标志。
		0	没有暂挂的 INT13 中断
		1	至少一个 INT13 中断正暂挂。 将 0 写入此位以清零和清除中断请求
11	INT12		中断 12 标志。 INT12 是连接到 CPU 中断级别 INT12 的中断的标志。
		0	没有暂挂的 INT12 中断
		1	至少一个 INT12 中断正暂挂。 将 0 写入此位以清零和清除中断请求

表 6-11. 中断标志寄存器 (IFR) — CPU 寄存器字段说明(接上表)

位	字段	值	说明
10	INT11		中断 11 标志。 INT11 是连接到 CPU 中断级别 INT11 的中断的标志。
		0	没有暂挂的 INT11 中断
		1	至少一个 INT11 中断正暂挂。 将 0 写入此位以清零和清除中断请求
9	INT10		中断 10 标志。 INT10 是连接到 CPU 中断级别 INT10 的中断的标志。
		0	没有暂挂的 INT10 中断
		1	至少一个 INT6 中断正暂挂。 将 0 写入此位以清零和清除中断请求
8	INT9		中断 9 标志。 INT9 是连接到 CPU 中断级别 INT6 的中断的标志。
		0	没有暂挂的 INT9 中断
		1	至少一个 INT9 中断正暂挂。 将 0 写入此位以清零和清除中断请求
7	INT8		中断 8 标志。 INT8 是连接到 CPU 中断级别 INT6 的中断的标志。
		0	没有暂挂的 INT8 中断
		1	至少一个 INT8 中断正暂挂。 将 0 写入此位以清零和清除中断请求
6	INT7		中断 7 标志。 INT7 是连接到 CPU 中断级别 INT7 的中断的标志。
		0	没有暂挂的 INT7 中断
		1	至少一个 INT7 中断正暂挂。 将 0 写入此位以清零和清除中断请求
5	INT6		中断 6 标志。 INT6 是连接到 CPU 中断级别 INT6 的中断的标志。
		0	没有暂挂的 INT6 中断
		1	至少一个 INT6 中断正暂挂。 将 0 写入此位以清零和清除中断请求
4	INT5		中断 5 标志。 INT5 是连接到 CPU 中断级别 INT5 的中断的标志。
		0	没有暂挂的 INT5 中断
		1	至少一个 INT5 中断正暂挂。 将 0 写入此位以清零和清除中断请求
3	INT4		中断 4 标志。 INT4 是连接到 CPU 中断级别 INT4 的中断的标志。
		0	没有暂挂的 INT4 中断
		1	至少一个 INT4 中断正暂挂。 将 0 写入此位以清零和清除中断请求
2	INT3		中断 3 标志。 INT3 是连接到 CPU 中断级别 INT3 的中断的标志。
		0	没有暂挂的 INT3 中断
		1	至少一个 INT3 中断正暂挂。 将 0 写入此位以清零和清除中断请求
1	INT2		中断 2 标志。 INT2 是连接到 CPU 中断级别 INT2 的中断的标志。
		0	没有暂挂的 INT2 中断
		1	至少一个 INT2 中断正暂挂。 将 0 写入此位以清零和清除中断请求
0	INT1		中断 1 标志。 INT1 是连接到 CPU 中断级别 INT1 的中断的标志。
		0	没有暂挂的 INT1 中断
		1	至少一个 INT1 中断正暂挂。 将 0 写入此位以清零和清除中断请求

6.5.4 中断启用寄存器 (IER) 和调试中断启用寄存器 (DBGIER)

IER 是 16 位的 CPU 寄存器。 IER 包含所有可屏蔽 CPU 中断级别（INT1-INT14、RTOSINT 和 DLOGINT）的启用位。 IER 中没有包括 NMI 和 XRS；因此，IER 对这些中断无效。

可以读取 IER 以标识已启用或禁用的中断级别，并且可以写入 IER 以启用或禁用中断级别。 要启用中断级别，请使用 OR IER 指令将其相应的 IER 位设置为 1。 要禁用中断级别，请使用 AND IER 指令将其相应的 IER 位设置为 0。 当中断被禁用时，无论 INTM 位的值是什么，都不会确认它。 当中断被启用时，如果相应的 IFR 位为 1 且 INTM 位为 0，则会确认它。

除非使用的是实时操作系统，否则当使用 OR IER 和 AND IER 指令修改 IER 位时，请确保它们不会修改位 15 (RTOSINT) 的状态。

当对 硬件中断提供服务或执行 时，将自动清除相应的 IER 位。 当 请求中断时，不会自动清除相应的 IER 位。 在使用 TRAP 指令的情况下，如果需要清除该位，必须通过中断服务例程来完成。

在复位时，所有 IER 位清零，以禁用所有可屏蔽 CPU 级别中断。

IER 寄存器如图 6-11 所示，图形下方为位说明。

图 6-11. 中断启用寄存器 (IER) — CPU 寄存器

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 6-12. 中断启用寄存器 (IER) — CPU 寄存器字段说明

位	字段	值	说明
15	RTOSINT		启用实时操作系统中断。 RTOSINT 启用或禁用 CPU RTOS 中断。
		0	禁用级别 INT6
		1	启用级别 INT6
14	DLOGINT		启用数据记录中断。 DLOGINT 启用或禁用 CPU 数据记录中断。
		0	禁用级别 INT6
		1	启用级别 INT6
13	INT14		启用中断 14。 INT14 启用或禁用 CPU 中断级别 INT14。
		0	禁用级别 INT14
		1	启用级别 INT14
12	INT13		启用中断 13。 INT13 启用或禁用 CPU 中断级别 INT13。
		0	禁用级别 INT13
		1	启用级别 INT13
11	INT12		启用中断 12。 INT12 启用或禁用 CPU 中断级别 INT12。
		0	禁用级别 INT12
		1	启用级别 INT12
10	INT11		启用中断 11。 INT11 启用或禁用 CPU 中断级别 INT11。
		0	禁用级别 INT11
		1	启用级别 INT11
9	INT10		启用中断 10。 INT10 启用或禁用 CPU 中断级别 INT10。
		0	禁用级别 INT10
		1	启用级别 INT10
8	INT9		启用中断 9。 INT9 启用或禁用 CPU 中断级别 INT9。
		0	禁用级别 INT9
		1	启用级别 INT9
7	INT8		启用中断 8。 INT8 启用或禁用 CPU 中断级别 INT8。
		0	禁用级别 INT8
		1	启用级别 INT8
6	INT7		启用中断 7。 INT7 启用或禁用 CPU 中断级别 INT7。
		0	禁用级别 INT7
		1	启用级别 INT7
5	INT6		启用中断 6。 INT6 启用或禁用 CPU 中断级别 INT6。
		0	禁用级别 INT6
		1	启用级别 INT6
4	INT5		启用中断 5。INT5 启用或禁用 CPU 中断级别 INT5。
		0	禁用级别 INT5
		1	启用级别 INT5

表 6-12. 中断启用寄存器 (IER) — CPU 寄存器字段说明(接上表)

位	字段	值	说明
3	INT4		启用中断 4。INT4 启用或禁用 CPU 中断级别 INT4。
		0	禁用级别 INT4
		1	启用级别 INT4
2	INT3		启用中断 3。INT3 启用或禁用 CPU 中断级别 INT3。
		0	禁用级别 INT3
		1	启用级别 INT3
1	INT2		启用中断 2。INT2 启用或禁用 CPU 中断级别 INT2。
		0	禁用级别 INT2
		1	启用级别 INT2
0	INT1		启用中断 1。INT1 启用或禁用 CPU 中断级别 INT1。
		0	禁用级别 INT1
		1	启用级别 INT1

仅当 CPU 在实时仿真模式下停机时，才使用 。 DBGIER 中启用的中断定义为对时间要求极其严格的中断。 当 CPU 在实时模式下停机时，服务的中断仅为 IER 中也启用的对时间要求极其严格的中断。 如果 CPU 在实时仿真模式下运行，则使用标准的中断处理进程且忽略 DBGIER。

与 IER 相似，可以读取 DBGIER 以标识已启用或禁用的中断，并且可以写入 DBGIER 以启用或禁用中断。 要启用中断，请将其相应的位设置为 1。 要禁用中断，请将其相应的位设置为 0。 使用 读取 DBGIER，并使用 POP DBGIER 写入 DBGIER 寄存器。 复位时，将所有 DBGIER 位设置为 0。

图 6-12. 调试中断启用寄存器 (DBGIER) — CPU 寄存器

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

图例：R/W = 读/写；R = 只读；*-n* = 复位后的值

表 6-13. 调试中断启用寄存器 (DBGIER) — CPU 寄存器字段说明

位	字段	值	说明
15	RTOSINT		启用实时操作系统中断。 RTOSINT 启用或禁用 CPU RTOS 中断。
		0	禁用级别 INT6
		1	启用级别 INT6
14	DLOGINT	。	启用数据记录中断。 DLOGINT 启用或禁用 CPU 数据记录中断。
		0	禁用级别 INT6
		1	启用级别 INT6
13	INT14	。	启用中断 14。 INT14 启用或禁用 CPU 中断级别 INT14
		0	禁用级别 INT14
		1	启用级别 INT14
12	INT13		启用中断 13。 INT13 启用或禁用 CPU 中断级别 INT13。
		0	禁用级别 INT13
		1	启用级别 INT13
11	INT12		启用中断 12。 INT12 启用或禁用 CPU 中断级别 INT12。
		0	禁用级别 INT12
		1	启用级别 INT12

表 6-13. 调试中断启用寄存器 (DBGIER) — CPU 寄存器字段说明(接上表)

位	字段	值	说明
10	INT11		启用中断 11。 INT11 启用或禁用 CPU 中断级别 INT11。
		0	禁用级别 INT11
		1	启用级别 INT11
9	INT10		启用中断 10。 INT10 启用或禁用 CPU 中断级别 INT10。
		0	禁用级别 INT10
		1	启用级别 INT10
8	INT9		启用中断 9。 INT9 启用或禁用 CPU 中断级别 INT9。
		0	禁用级别 INT9
		1	启用级别 INT9
7	INT8		启用中断 8。 INT8 启用或禁用 CPU 中断级别 INT8。
		0	禁用级别 INT8
		1	启用级别 INT8
6	INT7		启用中断 7。 INT7 启用或禁用 CPU 中断级别 INT7。
		0	禁用级别 INT7
		1	启用级别 INT7
5	INT6		启用中断 6。 INT6 启用或禁用 CPU 中断级别 INT6。
		0	禁用级别 INT6
		1	启用级别 INT6
4	INT5		启用中断 5。INT5 启用或禁用 CPU 中断级别 INT5。
		0	禁用级别 INT5
		1	启用级别 INT5
3	INT4		启用中断 4。INT4 启用或禁用 CPU 中断级别 INT4。
		0	禁用级别 INT4
		1	启用级别 INT4
2	INT3		启用中断 3。INT3 启用或禁用 CPU 中断级别 INT3。
		0	禁用级别 INT3
		1	启用级别 INT3
1	INT2		启用中断 2。INT2 启用或禁用 CPU 中断级别 INT2。
		0	禁用级别 INT2
		1	启用级别 INT2
0	INT1		启用中断 1。INT1 启用或禁用 CPU 中断级别 INT1。
		0	禁用级别 INT1
		1	启用级别 INT1

6.6 外部中断控制寄存器

某些器件支持三个屏蔽的 XINT1、XINT2 和 XINT13。 XINT13 与一个非可屏蔽中断 XNMI 多路复用。 可以选择每个这些外部中断为负沿或正沿触发，还可以启用或禁用它们（包括 XNMI）。 屏蔽的中断还包含 16 位自由运行的递增计数器，当检测到有效的中断沿时，该计数器复位为 0。 此计数器可用于准确设置中断的时间戳。

图 6-13. 外部中断 1 控制寄存器 (XINT1CR)（地址 7070h）

15	4	3	2	1	0
保留		极性		保留	启用
R-0		R/W-0		R-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-14. 外部中断 1 控制寄存器 (XINT1CR) 字段说明

位	字段	值	说明
15-4	保留		读取返回 0；写入无影响。
3-2	极性		此读/写位决定是在引脚信号的上升沿还是下降沿生成中断。
		00	在下降沿（高向低转换）生成中断
		01	在上升沿（低向高转换）生成中断
		10	在下降沿（高向低转换）生成中断
		11	在下降沿和上升沿（高向低转换和低向高转换）都生成中断
1	保留		读取返回 0；写入无影响
0	启用		此读/写位启用或禁用外部中断 XINT1。
		0	禁用中断
		1	启用中断

图 6-14. 外部中断 2 控制寄存器 (XINT2CR)（地址 7071h）

15	4	3	2	1	0
保留		极性		保留	启用
R-0		R/W-0		R-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-15. 外部中断 2 控制寄存器 (XINT2CR) 字段说明

位	字段	值	说明
15-4	保留		读取返回 0；写入无影响。
3-2	极性		此读/写位决定是在引脚信号的上升沿还是下降沿生成中断。
		00	在下降沿（高向低转换）生成中断
		01	在上升沿（低向高转换）生成中断
		10	在下降沿（高向低转换）生成中断
		11	在下降沿和上升沿（高向低转换和低向高转换）都生成中断
1	保留		读取返回 0；写入无影响
0	启用		此读/写位启用或禁用外部中断 XINT2。
		0	禁用中断
		1	启用中断

图 6-15. 外部 NMI 中断控制寄存器 (XNMICR) — 地址 7077h

15	4	3	2	1	0
保留		极性		选择	启用
R-0		R/W-0		R-0	R/W-0

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-16. 外部 NMI 中断控制寄存器 (XNMICR) 字段说明

位	字段	值	说明
15-4	保留		读取返回 0；写入无影响。
3-2	极性		此读/写位决定是在引脚信号的上升沿还是下降沿生成中断。
		00	在下降沿（高向低转换）生成中断
		01	在上升沿（低向高转换）生成中断
		10	在下降沿（高向低转换）生成中断
		11	在下降沿和上升沿（高向低转换和低向高转换）都生成中断
1	选择		选择 INT13 的源
		0	定时器 1 连接到 INT13
		1	XNMI_XINT13 连接到 INT13
0	启用		此读/写位启用或禁用外部中断 NMI
		0	禁用 XNMI 中断
		1	启用 XNMI 中断

XNMI 控制寄存器 (XNMICR) 可用于启用或禁用到 CPU 的 NMI 中断。 另外，可以选择 INT13 CPU 中断的源。 如 图 6-4 所示，INT13 中断的源可以是内部 CPU 定时器 1或指定给 XNMI 的外部 GPIO 信号。

在 F280x 器件上，CPU 定时器 1 保留供 TI 软件使用。 但是，INT13 中断仍然可以连接到 XNMI_XINT13 供客户使用。

表 6-17 显示了 XNMICR 寄存器设置与 28x CPU 的中断源之间的关系。

表 6-17. XNMICR 寄存器设置和中断源

XNMICR	寄存器位	28x CPU 中断		时间戳
启用	选择	NMI 源	INT13 源	(XNMICTR)
0	0	禁用	CPU 定时器 1	无
0	1	禁用	XNMI	无
1	0	XNMI	CPU 定时器 1	XNMI
1	1	禁用	XNMI	XNMI

对于每个外部中断，还有一个 16 位的计数器，每当检测到中断沿时，该计数器复位为 0x000。 这些计数器可用于准确设置发生中断的时间戳。

图 6-16. 外部中断 1 计数器 (XINT1CTR)（地址 7078h）

15	0
INTCTR 15-8	
R-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-18. 外部中断 1 计数器 (XINT1CTR) 字段说明

位	字段	说明
15-0	INTCTR	这是时钟频率为 SYSCLKOUT 的自由运行的 16 位递增计数器。 当检测到有效的中断沿时，计数器的值复位为 0x0000，然后继续计数，直到检测到下一个有效的中断沿。 当中断被禁用时，该计数器停止。 该计数器为自由运行的计数器，当达到最大值时将回绕到 0。 该计数器为只读寄存器，只能由有效的中断沿或通过复位来复位为 0。

图 6-17. 外部中断 2 计数器 (XINT2CTR) — 地址 7079h

15	0
INTCTR[15-0]	
R-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6.7. 外部中断 2 计数器 (XINT2CTR) 字段说明

位	字段	说明
15-0	INTCTR	这是时钟频率为 SYSCLKOUT 的自由运行的 16 位递增计数器。 当检测到有效的中断沿时，计数器的值复位为 0x0000，然后继续计数，直到检测到下一个有效的中断沿。 当中断被禁用时，该计数器停止。 该计数器为自由运行的计数器，当达到最大值时将回绕到 0。 该计数器为只读寄存器，只能由有效的中断沿或通过复位来复位为 0。

图 6-18. 外部 NMI 中断计数器 (XNMICTR)（地址 707Fh）

15	0
INTCTR[15-0	
R-0	

图例：R/W = 读/写；R = 只读；-*n* = 复位后的值

表 6-19. 外部 NMI 中断计数器 (XNMICTR) 字段说明

位	字段	说明
15-0	INTCTR	这是时钟频率为 SYSCLKOUT 的自由运行的 16 位递增计数器。 当检测到有效的中断沿时，计数器的值复位为 0x0000，然后继续计数，直到检测到下一个有效的中断沿。 当中断被禁用时，该计数器停止。 该计数器为自由运行的计数器，当达到最大值时将回绕到 0。 该计数器为只读寄存器，只能由有效的中断沿或通过复位来复位为 0。

TEXAS INSTRUMENTS

Appendix A

ZHCU003 – 2004 年 11 月 – 修 订 2005 年 5 月

修订历史记录

本文档将 SPRU712 修订为 SPRU712A。 附录部分仅列出在最新版本中所做的修订。 修订范围限定于如表 A-1 所示的技术方面的更改。

表 A-1. 对修订 A 的更改

位置	更改
第 1.2 部分	添加了描述复位后引导 ROM 虚拟读取 CSM 密码对闪存功率模式的影响的附注。
图 1-3	修改了图形，在其中删除了 XINTF
表 1-1	添加了 FSTDBYWAIT 和 FACTIVEWAIT 应保持其默认状态的附注。
表 1-4	在 3VSTAT 的描述中将 3V 更改为 3.3V。
第 2.1 部分	添加了有关使用零代码安全密码的影响的信息。
表 2-2	添加了 CSM 上的闪存配置寄存器表对其它片上资源的影响，并校正了闪存/ROM 的起始地址和闪存/ROM 的大小
表 2-4	校正了存储器中的 PWL 的行并添加了 CSMSCR 的复位值。
图 2-1	校正了 FORCESEC 的类型和复位值。
表 2-5	更新了 FORCESEC 的说明。
第 2.3.3 部分	添加了有关没有编程密码的器件行为的澄清。
第 2.3.3.2 部分	将 CSM 更改为 CSMSCR 且将地址更新为 0xAEF。
第 2.4.2 部分	将 “ 清除闪存 ” 更改为 “ 擦除闪存 ” 。 添加了以下澄清：如果在擦除闪存时发生复位，则 PWL 可能为零或未知。
第 2.5 部分	添加了以下澄清：已将 “ 当器件受保护时不能修改安全存储器。 ” 更改为 “ 当器件受保护时不能由从非安全存储器执行的代码修改安全存储器。 ” 。删除了对微计算机模式的参考。
表 3-4	更新了 WDINTS 位字段的说明。
图 3-7	更新了 PLL 插图以显示 PLL 旁路的正确路径和 PLL 之后 1/2 的位置。
表 3-7	更新了不同 PLL 模式的说明。 校正了 PLL 关闭时 SYSCLKOUT 与 OSCCLK 的关系。
第 3.2.2 部分	添加了在不同操作情况下跛形模式的限制和行为的列表。
第 3.2.3 部分	删除了 XCLKOUT 可以用作输入的注释。 此模式保留供 TI 使用。
图 3-9	更新了 PLL 插图以正确显示 PLL 旁路和 PLL 之后 1/2 的位置。 校正了 SYSCLKOUT 与 XCLKOUT 的比率为 1:1 时 XCLKOUTDIV 的值。
表 3-1	添加了有关 PLLCR 和 PLLSTS 如何复位的第二个注意事项。
图 3-12	添加了有关 PLLSTS 寄存器复位的注意事项。
表 3-10	已将 XCLKINCNT、X1CNT、XCLKINDAT、X1DAT 和 XCLKOUTDAT 标记为保留仅供 TI 使用。 删除了 XCLKOUT 可以用作输入的注释。
表 3-12	已将 LPMCR 寄存器更改为 LPMCR0 寄存器。
第 3.4 部分	可以由 WDINTS 位确定 $\overline{\text{WDINT}}$ 的状态，而不是以前声明的 WDENINT。 添加了：当发出看门狗中断时，$\overline{\text{WDINT}}$ 将保持低电平 512 个 OSCCLK 周期。 更新了复位模式和中断模式的说明。
图 3-15	给图形注意事项添加了：如果启用了看门狗中断，则 $\overline{\text{WDINT}}$ 也将保持 512 个 OSCCLK 周期。
表 3-12、表 3-13	添加了警告：当设置了缺少时钟状态位时，不要进入 HALT 低功率模式。
图 3-18	对 WDCHK 字段（位 5-3）进行了添加，以指示这些位读回的数为 0,0,0。
表 4-5、表 4-6、表 4-7、表 4-9、表 4-10、 表 4-11	添加了以下信息：描述在 GPxMUX1/2（x = A 或 B）寄存器中选择了保留的位置时引脚的行为。
图 4-3	添加了图形之前的文本并修改了图形
表 4-11	已将 GPIO 端口 B MUX1 寄存器中的位 7-6 更改为保留。

表 A-1. 对修订 A 的更改(接上表)

位置	更改
表 4-14	更改了位 7-0 的位说明
第 4.6 部分	已将位 7-6 更改为保留
表 4-17	已将 GPBQSEL2 寄存器更改为保留
表 4-19	已将 GPBDIR 中的位 3 更改为保留
表 4-20	已将 GPAPUD 位 12-31 的复位状态从 1 更改为 0：启用内部上拉电阻。
表 4-21	已将 GPBPUD 中的位 3 更改为保留。 已将位 0、1、2 (GPIO32-GPIO34) 的复位状态更改为 0：启用内部上拉电阻。
表 5-12、表 5-13	添加了 HRCNFG 寄存器。
表 6-5	已将中断 USER0-USER11 的名称更改为 USER1-USER12。 这将与其它文档保持一致。 添加了有关未保留 XINT13 的注意事项。
表 6-7	删除了对 XINTF 区域 7 和 XMPNMC 的参考，因为 280x 器件上没有 XINTF。