Errata

F28E12x 实时 MCU 器件勘误表
器件修订版本 0

TEXAS INSTRUMENTS

摘要

本文档介绍了功能规格的已知例外情况（公告）。本文档也包含了使用说明。在使用说明中介绍了器件行为可能与假定或记录的行为不匹配的情况。这可能包括影响器件性能或功能正确性的行为。

内容

插图清单

表格清单

1 使用说明和公告模型

表 1-1 列出了所有使用说明和适用的器件修订版本。表 1-2 列出了所有公告、受影响的模块以及适用的器件修订版本。

1.1 使用说明汇总表

表 1-1. 使用说明汇总表

编号	标题	受影响的器件修订版本
		0
节 3.1.1	PIE：背对背 PIEACK 写入和手动 CPU 中断屏蔽清除之后的伪波嵌套中断	是
节 3.1.2	将嵌套中断与重复块一起使用时的注意事项	是
节 3.1.3	安全性：主要的防御层是构建芯片安全边界，从启用 JTAGLOCK 和零引脚引导至闪存功能开始	是

1.2 公告汇总表

表 1-2. 公告汇总表

模块	说明	受影响的器件修订版本
		0
ADC	ADC：如果未设置 INTxCONT（继续中断模式），中断可能会停止	是
ADC	ADC：使用 ADCCLK 小数分频器时 ADC 性能下降	是
BOR	BOR：2.45V 至 3.0V 之间的 VDDIO 可产生多个 XRSn 脉冲	是
CMPSS	CMPSS：在某些情况下，COMPxLATCH 可能无法正确清除	是
CMPSS	CMPSS：如果比较器输入引脚具有 AGPIO 功能并且 ADC 正在对输入引脚进行采样，则可能会发生 CMPSS 干扰	是
eQEP	eQEP：索引期间方向变化时位置计数器复位错误	是
闪存	闪存：不生成单个位 ECC 错误中断	是
FPU	FPU：FPU 至 CPU 寄存器移动操作之前是任何 FPU 2p 操作	是
GPIO	GPIO：漏极开路配置可以驱动短高电平脉冲	是
MCD	MCD：启用 PLL (PLLCLKEN = 1) 后，应该禁用时钟丢失检测	是
存储器	存储器：在有效存储器之外进行预取	是
PLL	PLL：：第一次尝试锁定时，PLL 可能无法按预期锁定	是
系统	系统：多次连续写入 CLKSRCCTL1 可能会导致系统死机	是
看门狗	看门狗：WDKEY 寄存器不受 EALLOW 保护	是
看门狗	看门狗：写入 WDHALTI 位会影响 PLL 锁定状态	是

2 命名规则、封装编号法和修订版本标识

2.1 器件和开发支持工具命名规则

德州仪器 (TI) 为其支持工具推荐使用三种可能的前缀指示符中的两个：TMDX 和 TMDS。这些前缀代表了产品开发的发展阶段，即从工程原型 (TMDX) 直到完全合格的生产工具 (TMDS)。

器件开发演变流程：

X 试验器件不一定代表最终器件的电气规范标准，并且可能不使用生产组装流程。

P 原型器件不一定是最终器件模型，并且不一定符合最终电气标准规范。

无 完全合格的芯片模型的生产版本。

支持工具开发演变流程：

TMDX 还未经德州仪器 (TI) 完整内部质量测试的开发支持产品。

TMDS 完全合格的开发支持产品。

X 和 P 器件和 TMDX 开发支持工具在供货时附带如下免责条款：

“开发的产品用于内部评估用途。”

生产器件和 TMDS 开发支持工具已进行完全特性描述，并且器件的质量和可靠性已经完全论证。TI 的标准保修证书适用。

预测显示原型器件 (X 或者 P) 的故障率大于标准生产器件。由于这些器件的预期最终使用故障率仍未确定，故德州仪器 (TI) 建议请勿将这些器件用于任何生产系统。请仅使用合格的生产器件。

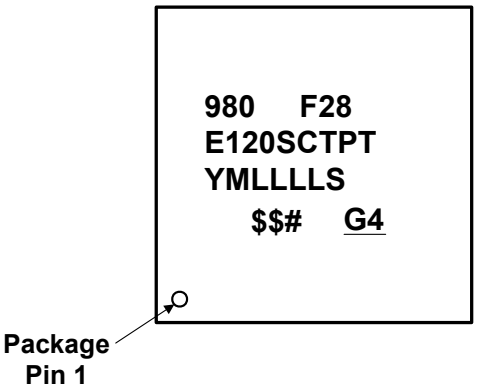
2.2 支持的器件

本文档支持以下器件：

- F28E120SC
- F28E120SB

2.3 封装编号法和修订版本标识

图 2-1、图 2-2 和 图 2-3 展示了封装编号法。表 2-1 列出了器件修订版本代码。



YMLLLLS = Lot Trace Code

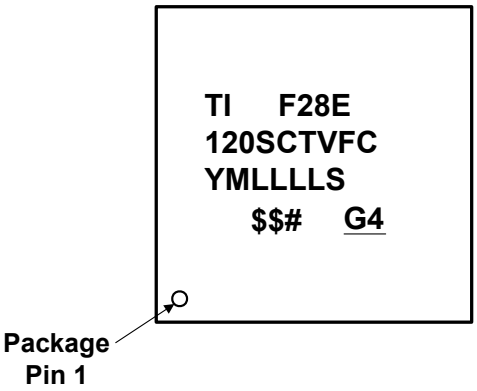
980 = TI EIA Code
YM = 2-digit Year/Month Code
LLLL = Assembly Lot Code
S = Assembly Site Code
$$ = Wafer Fab Code (one or two characters) as applicable
 # = Silicon Revision Code

G4 = Green (Low Halogen and RoHS-compliant)

备注

YM =“5A”的 F28E120SCTPT 器件错误地印成了器件版本代码 =“A”。这些器件为器件版本 0（# 为空）。

图 2-1. PT 封装的封装编号法

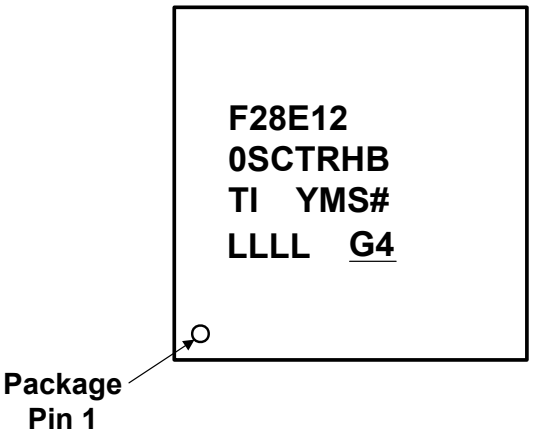


YMLLLLS = Lot Trace Code

YM = 2-digit Year/Month Code
LLLL = Assembly Lot Code
S = Assembly Site Code
$$ = Wafer Fab Code (one or two characters) as applicable
 # = Silicon Revision Code

G4 = Green (Low Halogen and RoHS-compliant)

图 2-2. VFC 封装的封装编号法



YM = 2-digit Year/Month Code
S = Assembly Site Code
 # = Silicon Revision Code
LLLL = Assembly Lot Code

G4 = Green (Low Halogen and RoHS-compliant)

图 2-3. RHB 封装的封装编号法

表 2-1. 版本标识

器件修订版本代码	器件修订版本	REVID(1) 地址：0x5D006	注释(2)
空白	0	0x0000 0001	该器件修订版本的代码为 TMX 和 TMS。

(1) 器件修订版本 ID
(2) 有关可订购器件型号，请参阅 F28E12x *实时微控制器数据表中*的“封装信息”表。

3 器件修订版本 0 使用说明和公告

本节列出了此器件修订版本的使用说明和公告。

3.1 器件修订版本 0 使用说明

本节列出了适用于器件修订版本 0 的所有使用说明。

3.1.1 PIE：背对背 PIEACK 写入和手动 CPU 中断屏蔽清除之后的伪波嵌套中断

受影响的版本：0

某些用于嵌套中断的代码序列允许 CPU 和 PIE 进入不一致状态，从而触发不必要的中断。进入该状态所需的条件为：

1. PIEACK 清除后，立即执行全局中断使能（EINT 或 ASM (" CLRC INTM") ）。
2. 嵌套中断会清除其组的一个或多个 PIEIER 位。

是否触发不必要的中断，取决于系统中其他中断的配置和时序。在大多数应用中，预计这种事件很罕见或根本不存在。如果发生这种情况，不必要的中断将是嵌套中断的 PIE 组中的第一个中断、并将在嵌套中断重新启用 CPU 中断（EINT 或 asm ("CLRC INTM") ）后触发。

权变措施：在 PIEACK 写入和 CPU 中断使能之间添加一个 NOP（无操作）。以下显示了示例代码。

```
//Bad interrupt nesting code
PieCtrlRegs.PIEACK.all = 0xFFFF;      //Enable nesting in the PIE
EINT;                                 //Enable nesting in the CPU

//Good interrupt nesting code
PieCtrlRegs.PIEACK.all = 0xFFFF;      //Enable nesting in the PIE
asm(" NOP");                          //Wait for PIEACK to exit the pipeline
EINT;                                 //Enable nesting in the CPU
```

3.1.2 将嵌套中断与重复块一起使用时的注意事项

受影响的版本：0

如果用户为了使用嵌套特性而在中断处理例程 (ISR) 内使用 EINT 指令启用中断，那么用户必须在退出 ISR 之前通过使用 DINT 汇编指令来禁用中断。否则可能会导致 RB 寄存器中的位无法正确恢复，从而导致异常代码行为。

如果应用程序中未使用 RPTB ASM 指令，则没有问题。对于 C 语言源代码，需要对生成的反汇编代码进行分析以验证是否发生此情况。

如果 ISR 用 C 语言编码，那么 C28x C 编译器可以处理上述问题，无需执行任何操作。如果 ISR 用 C28x 汇编语言编码，则必须遵循上述指南。

备注

CGT v15.12.2.LTS（2016 年 4 月发布）或更高版本的 CGT 封装会自动满足此要求。只有早期版本的 CGT 工具才需要添加 DINT。

3.1.3 安全性：主要的防御层是构建芯片安全边界，从启用 JTAGLOCK 和零引脚引导至闪存功能开始

受影响的版本：0

在任何情况下都不允许未经授权的代码进入器件并执行，这是器件安全的前提条件。为此，器件提供了两项功能，担心安全性的用户应该始终启用这两项功能。

- **JTAGLOCK**

  当在闪存的 USER OTP 区域中启用时，JTAGLOCK 功能会禁用对器件的资源进 JTAG 访问（例如调试器连接），阻止未经授权的一方使用 JTAG 接口将任何代码下载到器件中。启用 JTAGLOCK 后，用户仍然可以通过输入密码来允许授权方解锁，也可以通过将密码值全部设置为零来永久锁定。

- **零引脚引导至闪存**

  内置于 TI ROM 中的外部引导加载程序不会对所下载的代码执行任何身份验证。在 USER OTP 中启用零引脚引导选项以及闪存引导模式，则在引导时会阻止运行所有基于引脚的外部引导加载程序选项（例如 SCI、CAN、并行），这是因为在基本引导 ROM 执行结束后，会强制使引导过程立即跳转到内部闪存。为了保障最高的安全性，可选择安全闪存启动模式。这可以在跳转到闪存代码之前通过基本引导 ROM 对闪存代码进行预检查。

如果 JTAG 被永久锁定并且启用了零引脚引导至闪存选项，则通过 JTAG 或内置引导加载程序与器件进行通信的编程工具将失效。如果需要能够执行固件升级，则用户必须将代码预存储在闪存中以安全地管理和执行更新。

3.2 器件修订版本 0 公告

本节列出了适用于器件修订版本 0 的所有公告。

公告	***ADC：如果未设置 INTxCONT（继续中断模式），中断可能会停止***
受影响版本	0
详细信息	在 ADCINTSELxNx[INTxCONT]= 0 时，设置 ADCINTFLG 后，中断将停止，并且不会发生其他 ADC 中断。 若在 ADCINTFLGCLR 寄存器进行软件写入的同时发生 ADC 中断，则 ADCINTFLG 将意外保持为设置状态，阻止将来发生 ADC 中断。
应变方法	1. 使用 “继续中断” 模式，则 ADCINTFLG 无法阻止其他 ADC 中断： 2. 为了避免发生这种情况，请确保下一次发生 ADC 中断之前，始终有足够的时间为 ADC ISR 提供服务并清除 ADCINTFLG。 3. 清除 ADCINTFLG 时，请检查 ISR 中是否存在溢出情况。在写入到 ADCINTFLGCLR 后立即检查 ADCINTOVF；如果已设置，则再次写入 ADCINTFLGCLR 以确保 ADCINTFLG 已被清除。若再设置 ADCINTOVF 寄存器，则表示已丢失 ADC 转换中断。

```
ADCINTSEL1N2[INT1CONT] = 1;
ADCINTSEL1N2[INT2CONT] = 1;
ADCINTSEL3N4[INT3CONT] = 1;
ADCINTSEL3N4[INT4CONT] = 1;
```

```
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;        //clear INT1 flag
if(1 == AdcaRegs.ADCINTOVF.bit.ADCINT1)       //ADCINT overflow
{
    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;    //clear INT1 again
// If the ADCINTOVF condition will be ignored by the application
// then clear the flag here by writing 1 to ADCINTOVFCLR.
// If there is a ADCINTOVF handling routine, then either insert
// that code and clear the ADCINTOVF flag here or do not clear
// the ADCINTOVF here so the external routine will detect the
// condition.
//    AdcaRegs.ADCINTOVFCLR.bit.ADCINT1 = 1;  // clear OVF
}
```

公告

ADC：使用 ADCCLK 小数分频器时 ADC 性能下降

受影响版本

0

详细信息

已证明使用分数 SYSCLK 转 ADCCLK 分频器（由 ADCCTL2.PRESCALE 字段控制）会导致该器件的 ADC 性能下降。请参阅表 3-1。

表 3-1. ADCCTL2 寄存器

性能下降			
位	**字段**	**值**	**说明**
3-0	预分频	0001	ADCCLK = SYSCLK/1.5
		0003	ADCCLK = SYSCLK/2.5
		...	
性能正常			
位	**字段**	**值**	**说明**
3-0	预分频	0000	ADCCLK = SYSCLK/1.0
		0002	ADCCLK = SYSCLK/2.0
		...	

权变措施

使用偶数预分频时钟分频器值。偶数预分频值会产生整数时钟分频，而不会影响 ADC 性能。

公告 **BOR：2.45V 至 3.0V 之间的 VDDIO 可产生多个 XRSn 脉冲**

受影响版本

0

详细信息

当 VDDIO 电源电压介于 2.45V 和 3.0V 之间时，BOR 会生成重复的 XRSn 置为有效和置为无效 。建议*不要*直接将 XRSn 引脚用作系统中任何其他器件的复位。

即使在这些 XRSn 脉冲发生时，F28E12x BOR 也能有效地在内部将器件保持在已知的复位状态。在 VDDIO 供电电压升至 3.0 V 以上之前，该器件不会转移应用代码或引导加载程序，所有其他引脚都将保持其复位状态。

应变方法

1. 上电、断电和 BOR 事件期间，忽略额外的 XRSn 切换。额外的 XRSn 脉冲将不会影响 F28E12x 器件本身的运行。
2. 如果 XRSn 脉冲会导致其他系统元件出现不良的系统行为，则请勿使用 XRSn 驱动其他器件。这些应用可使用外部电压监控器。
3. 对于需要在正常上电和断电期间避免这些脉冲的应用：
   a. 上电：请遵循 *F28E12x 实时微控制器*数据表的建议运行条件表中的 SR_{SUPPLY} 要求；不会出现额外的 XRSn 低电平脉冲。
   b. 断电：为避免 XRSn 在断电期间失效，在对电源进行设计时应确保 VDDIO 在 25 µs 内支持 3.0V 至 2.45V 的电压范围 。如果 XRSn 上的电压上升是可以接受的，则可以计算 XRSn 上实现的 RC 电路的时间常数，以确保电压不会上升到系统指定的阈值以上。

公告 **CMPSS：在某些情况下，COMPxLATCH 可能无法正确清除**

受影响版本

0

详细信息

CMPSS 锁存路径旨在在本地锁存器 (COMPxLATCH) 中保持跳闸状态，直到由软件（通过 COMPSTSCLR）或 PWMSYNC 清除。

在信号经过数字滤波器数字化和鉴定后，COMPxLATCH 由比较器输出间接设置。比较器输出到达 COMPxLATCH 的预期最大延时可以用 CMPSS 模块时钟周期表示为：

LATENCY = 1 + (1 x FILTER_PRESCALE) + (FILTER_THRESH x FILTER_PRESCALE)

当 COMPxLATCH 由软件或 PWMSYNC 清除时，锁存本身根据需要清除，但 COMPxLATCH 之前的数据路径可能不会反映额外延时数的模块时钟周期的比较器输出值。如果在 COMPxLATCH 被清除时数字滤波器输出解析为逻辑 1，则锁存将在下一个时钟周期再次设置。

应变方法

在清除 COMPxLATCH 之前，让数字滤波器输出解析为逻辑 0。

如果软件清除了 COMPxLATCH，那么在清除锁存之前可以通过 COMPSTS 寄存器确认数字滤波器的输出状态。对于较大的延时值会产生不可容忍的延时的情况，可以通过重新初始化数字滤波器（通过 CTRIPxFILCTL）来刷新滤波器 FIFO。

如果 COMPxLATCH 被 PWMSYNC 清除，设计用户应用程序时应使得比较器跳闸条件在 PWMSYNC 生成前能至少清除延时周期。

公告

CMPSS：如果比较器输入引脚具有 AGPIO 功能并且 ADC 正在对输入引脚进行采样，则可能会发生 CMPSS 干扰

受影响版本

0

详细信息

需要特别注意特定模拟输入引脚的用例组合，如表 3-2 所示。如此表所示，对于 CMPSS 输入、ADC 采样和 AGPIO 的组合，需要使用特殊注意事项或权变措施。

表 3-2. 特定模拟输入引脚的用例组合

特定模拟引脚上使用的功能	使用的元件				
CMPSS 比较器输入	是	-	是	-	是
ADC 采样	是	是	-	是	是
AGPIO 模拟引脚类型	是	是	是	-	-
AIO 模拟引脚类型	-	-	-	是	是
结果	**需要权变措施**	**无需特殊分析或权变措施**			

AGPIO 模拟引脚路径包含一个额外的 53Ω 串联开关。这会创建一个由 ADC 和 CMPSS 比较器共享的低电容隔离式节点，如图 3-1 所示。当 ADC 对通道进行采样时，该节点可能会受到干扰（取决于 ADC 采样保持电容器上先前存储的电压），这种干扰可能会导致高达 50ns 的错误 CMPSS 事件。为了适应这种潜在的干扰，可以实施以下权变措施。

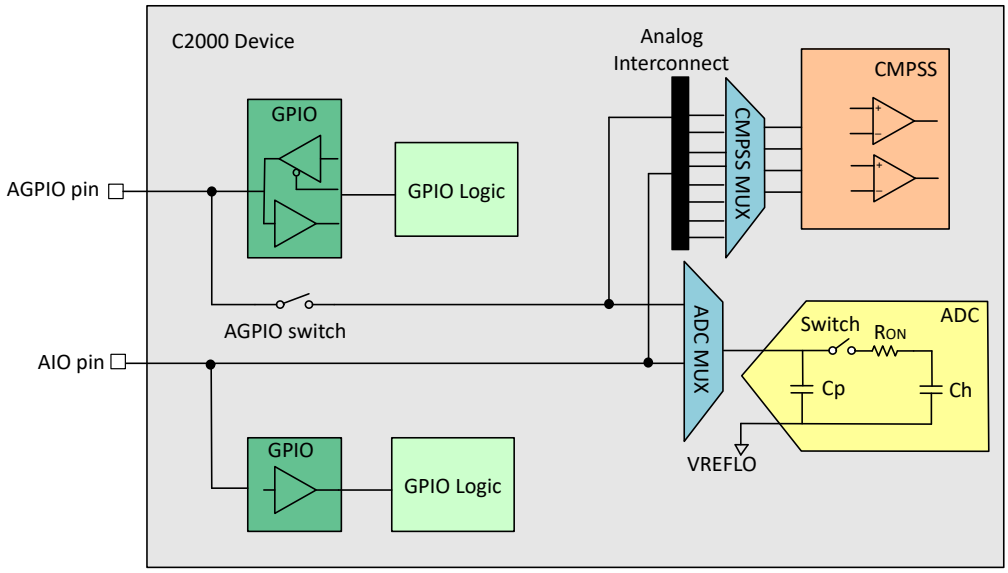


图 3-1. 具有 AGPIO 和 AIO 模拟引脚类型的模拟子系统图

应变方法

1. 对于同时需要 ADC 和 CMPSS 的模拟通道，使用不同的引脚（即 AIO 引脚类型）。
2. 使用设置为 50ns 或更大的 CMPSS 数字滤波器，从而将滤除临时干扰。
3. 预处理 ADC 的采样保持电容器，干扰不会导致误跳闸。例如，在读取受影响的通道之前，立即对 ADC 上不同通道的 3.3V 连接执行虚拟读取，从而使干扰为正向，远离误跳闸。如果误跳闸极性反转，则将使用 0V 信号的相反虚拟读取。

公告 **eQEP: 索引期间方向变化时位置计数器复位错误**

受影响版本

0

详细信息

在使用 PCRM = 0 的配置时，若索引输入处于活动状态时方向发生变化，则位置计数器 (QPOSCNT) 可能会错误地复位，从而导致计数器值发生意外变化。若如此，可能会与位置计数器的预期值相差多达 ±4 个数值，并随后意外设置错误标志。

在使用 PCRM = 0 的配置时[即发生索引事件时位置计数器复位 (QEPCTL[PCRM] = 00)]，若在正向运动期间发生索引事件，则位置计数器在下一个 eQEP 时钟时复位为 0。若在反向移动期间发生索引事件，则位置计数器将在下一个 eQEP 时钟时复位为 QPOSMAX 寄存器中的值。eQEP 外围设备会记录第一个索引标记 (QEPSTS[FIMF]) 的出现情况以及 QEPSTS 寄存器中第一个索引事件标记 (QEPSTS[FIDF]) 方向。此外，其还会记录第一个索引标记上的正交边沿以便在索引事件复位操作中使用相同的相对正交跃迁。

如果在索引脉冲活动时方向发生改变，则该模块仍将继续查找相对正交跃迁以执行位置计数器复位操作。若如此，位置计数器的值会发生意外变化。

未同时发生方向改变的下一个索引事件将正确复位计数器并按预期工作。

应变方法

在索引处于活动状态时，若方向发生变化且位置计数器值的相应变化可能会影响应用程序，请勿使用 PCRM = 0 的配置。

若适用于此应用程序，则位置计数器复位所运用的其他选项[例如索引事件初始化(IEI)]不会出现此问题。

公告	*闪存：不生成单个位 ECC 错误中断*
受影响版本	0
详细信息	如果单个位 ECC 错误阈值配置为 0，则当存在单个位错误时不会生成单个位错误中断。
权变措施	将错误阈值位字段（FLASH_ECC_REGS ERR_THRESHOLD.ERR_THRESHOLD 字段）设置为大于或等于 1 的值。请注意：阈值位字段的默认值为 0。

公告

FPU：FPU 至 CPU 寄存器移动操作之前是任何 FPU 2p 操作

受影响版本

0

详细信息

当多周期 (2p) FPU 指令后跟 FPU 到 CPU 寄存器转移时，此公告适用。如果 FPU 到 CPU 读取指令源寄存器与 2p 指令目标相同，则在 2p 指令完成之前读取的可能是 FPU 寄存器的值。这是因为 2p 指令依赖于流水线 E3 阶段期间结果的数据转发。如果在 E3 阶段碰巧发生流水线停滞，则读取指令不会及时转发结果。

受此公告影响的 2p 指令是 MPYF32、ADDF32、SUBF32 和 MACF32。FPU 寄存器读取的目标必须是一个 CPU 寄存器（ACC、P、T、XAR0...XAR7）。如果寄存器读取是 FPU 到 FPU 寄存器转移，则此公告不适用。

在下面的示例中，2p 指令 MPYF32 使用 R6H 作为其目标。FPU 寄存器读取 MOV32 使用同一个寄存器 R6H 作为其源，并使用 CPU 寄存器作为目标。如果在 E3 流水线阶段发生停滞，则 MOV32 将在 MPYF32 指令完成之前读取 R6H 的值。

问题示例：

```
   MPYF32 R6H, R5H, R0H  ; 2p FPU instruction that writes to R6H
|| MOV32 *XAR7++, R4H
   F32TOUI16R R3H, R4H   ; delay slot
   ADDF32 R2H, R2H, R0H
|| MOV32 *--SP, R2H      ; alignment cycle
   MOV32 @XAR3, R6H      ; FPU register read of R6H
```

图 3-2 展示了流水线中没有停滞时的问题流水线图。

	Instruction	F1	F2	D1	D2	R1	R2	E	W		Comments
		FPU pipeline--->				R1	R2	E1	E2	E3	
I1	MPYF32 R6H, R5H, R0H || MOV32 *XAR7++, R4H	I1									
I2	F32TOUI16R R3H, R4H	I2	I1								
I3	ADDF32 R3H, R2H, R0H || MOV32 *--SP, R2H	I3	I2	I1							
I4	MOV32 @XAR3, R6H	I4	I3	I2	I1						
			I4	I3	I2	I1					
				I4	I3	I2	I1				
					I4	I3	I2	I1			
						I4	I3	I2	I1		
							I4	I3	I2	**I1**	I4 samples the result as it enters the R2 phase. The product R6H=R5H*R0H (I1) finishes computing in the E3 phase, but is **forwarded** as an operand to I4. This makes I4 appear to be a 2p instruction, but I4 actually takes 3p cycles to compute.
								I4	I3	I2	
									I4	I3	

图 3-2. 流水线中没有停滞时的问题流水线图

公告（续）

FPU：FPU 至 CPU 寄存器移动操作之前是任何 FPU 2p 操作

图 3-3 展示了指令 I1 的 E3 时隙中存在停滞时的问题流水线图。

	Instruction	F1	F2	D1	D2	R1	R2	E	W		Comments
		FPU pipeline-->				R1	R2	E1	E2	E3	
I1	MPYF32 R6H, R5H, R0H || MOV32 *XAR7++, R4H	I1									
I2	F32TOUI16R R3H, R4H	I2	I1								
I3	ADDF32 R3H, R2H, R0H || MOV32 *--SP, R2H	I3	I2	I1							
I4	MOV32 @XAR3, R6H	I4	I3	I2	I1						
			I4	I3	I2	I1					
				I4	I3	I2	I1				
					I4	I3	I2	I1			
						I4	I3	I2	I1		
							I4	I3	I2	I1 (STALL)	I4 samples the result as it enters the R2 phase, but I1 is stalled in E3 and is unable to forward the product of R5H*R0H to I4 (R6H does not have the product yet due to a design bug). So, I4 reads the old value of R6H.
							I4	I3	I2	I1	There is no change in the pipeline as it was stalled in the previous cycle. I4 had already sampled the old value of R6H in the previous cycle.
								I4	I3	I2	Stall over

图 3-3. 指令 I1 的 E3 时隙中存在停滞时的问题流水线图

权变措施

在本例中，应将 MPYF32、ADDF32、SUBF32 和 MACF32 视为 3p 周期指令。必须将三个 NOP 或非冲突指令放置在指令的延迟时隙中。

C28x 代码生成工具 v.6.2.0 及更高版本都将生成正确的指令序列并检测汇编代码中的错误。在以前的版本 v6.0.5（适用于 6.0.x 分支）和 v.6.1.2（适用于 6.1.x 分支）中，编译器将生成正确的指令序列，但汇编器不会检测汇编代码中的错误。

解决方法示例：

```
   MPYF32 R6H, R5H, R0H
|| MOV32 *XAR7++, R4H      ; 3p FPU instruction that writes to R6H
   F32TOUI16R R3H, R4H     ; delay slot
   ADDF32 R2H, R2H, R0H
|| MOV32 *--SP, R2H        ; delay slot
   NOP                     ; alignment cycle
   MOV32 @XAR3, R6H        ; FPU register read of R6H
```

图 3-4 展示了带解决方法的流水线图。

公告（续）

FPU：FPU 至 CPU 寄存器移动操作之前是任何 FPU 2p 操作

	Instruction	F1	F2	D1	D2	R1	R2	E	W		Comments
		FPU pipeline-->				R1	R2	E1	E2	E3	
I1	MPYF32 R6H, R5H, R0H || MOV32 *XAR7++, R4H	I1									
I2	F32TOUI16R R3H, R4H	I2	I1								
I3	ADDF32 R3H, R2H, R0H || MOV32 *--SP, R2H	I3	I2	I1							
I4	NOP	I4	I3	I2	I1						
I5	MOV32 @XAR3, R6H	I5	I4	I3	I2	I1					
			I5	I4	I3	I2	I1				
				I5	I4	I3	I2	I1			
					I5	I4	I3	I2	I1		
						I5	I4	I3	I2	I1 (STALL)	Due to one extra NOP, I5 does not reach R2 when I1 enters E3; thus, forwarding is not needed.
						I5	I4	I3	I2	I1	There is no change due to the stall in the previous cycle.
							I5	I4	I3	I2	I1 moves out of E3 and I5 moves to R2. R6H has the result of R5H*R0H and is read by I5. There is no need to forward the result in this case.
								I5	I4	I3	

图 3-4. 带解决方法的流水线图

公告 **GPIO：漏极开路配置可以驱动短高电平脉冲**

受影响版本

0

详细信息

每个 GPIO 都可以使用 GPxODR 寄存器配置为漏极开路模式。但是，内部器件时序问题可能会导致 GPIO 在进入或退出高阻抗状态期间将逻辑高电平驱动至高达 0 – 10 ns。

如果另一个驱动器同步驱动低电平，这种不必要的高电平可能会导致 GPIO 与线路上的另一个漏极开路驱动器发生争用。我们不希望出现此争用，因为它会对两个器件施加应力，并导致信号上出现短暂的中间电压电平。如果接收器逻辑中没有足够的逻辑滤波来滤除此短暂脉冲，则该中间电压电平可能会被错误地解读为高电平。

权变措施

如果存在争用问题，请勿使用 GPIO 的漏极开路功能；请改为在软件中仿真漏极开路模式。通过将 GPIO 数据 (GPxDAT) 设置为静态 0 并切换 GPIO 方向位 (GPxDIR) 来启用和禁用驱动低电平，可以实现漏极开路仿真。有关实现示例，请参阅以下代码。

```
void main(void)
{ ...

  // GPIO configuration
     EALLOW;                                  // disable pullup
     GpioCtrlRegs.GPxPUD.bit.GPIOx = 1;       // disable open-drain mode
     GpioCtrlRegs.GPxODR.bit.GPIOx = 0;       // set GPIO to drive static 0 before
                                              // enabling output
     GpioDataRegs.GPxCLEAR.bit.GPIOx = 1;
     EDIS;
  ...

  // application code
  ...

  // To drive 0, set GPIO direction as output
     GpioCtrlRegs.GPxDIR.bit.GPIOx = 1;

  // To tri-state the GPIO(logic 1),set GPIO as input
     GpioCtrlRegs.GPxDIR.bit.GPIOx = 0;
}
```

公告	***MCD：启用 PLL (PLLCLKEN = 1) 后，应该禁用时钟丢失检测***
受影响版本	0
详细信息	PLL 具有跛行模式功能，可提供慢速 PLLRAWCLK 输出（即使缺少其输入 OSCCLK）。另外，当检测到缺少 OSCCLK 输入时，时钟丢失检测 (MCD) 电路会强制将系统时钟源切换到 INTOSC1。当两个时钟源（PLLRAWCLK 和 INTOSC1）都仍然有效时，无法确保在这些系统时钟源之间切换的 MCD 多路复用器无干扰。在极少数情况下，这可能会导致在时钟丢失故障事件期间出现不可预测的器件行为。
应变方法	当系统使用 PLL (PLLCLKEN = 1) 时，可通过写入 MCDCR.MCLKOFF = 1 来禁用 MCD。 可对双时钟比较器 (DCC) 电路进行配置，以快速检测 SYSCLK 频率是否由于时钟丢失事件而降至低于所需频率，导致进入跛行模式。 当系统以 PLL 旁路模式 (PLLCLKEN = 0) 运行时，仍可使用 MCD 电路来检测时钟丢失事件并将时钟源切换到 INTOSC1。

公告 ***存储器：在有效存储器之外进行预取***

受影响版本

0

详细信息

C28x CPU 预取指令的范围超出其流水线中当前正在活动的指令范围。如果预取发生在有效存储器结束之后，则 CPU 可能会接收到无效的操作码。

权变措施

M1 – 预取队列的深度为 8 x16 字。因此，代码不应在有效存储器末尾的 8 个字以内。可以在两个有效存储器块之间实现跨边界预取。

示例 1：M1 在地址 0x7FF 处结束，后面不跟随另一个存储器块。M1 中代码的存储地址不应超过 0x7F7。地址 0x7F8-0x7FF 不应用于代码。

示例 2：M0 结束于地址 0x3FF，有效存储器 (M1) 紧随其后。M0 中的代码可存储在 0x3FF 及以下的地址。代码也可以交叉进入 M1，最高到地址 0x7F7（含地址 0x7F7）。

表 3-3. 受公告影响的存储器

内存类型	受影响的地址
M1	0x0000 07F8-0x0000 07FF

公告 ***PLL：第一次尝试锁定时，PLL 可能无法按预期锁定***

受影响版本

0

详细信息

第一次尝试锁定时，PLL 可能无法正确锁定。PLLSTS[LOCKS] 位被置位，但 PLL 锁定在意外且随机的频率。在随后禁用并重新启用 PLL 时，可能会再次出现 PLL 锁定问题。

如果 SYSPLL 未正确锁定并被选择作为 CPU 时钟源，则由于锁定的频率不明确，CPU 将出现异常行为。

这一瞬态问题的发生率很低。禁用并重新启用 PLL 时，可能会在系统中观察到该问题。权变措施是进行重试，直到 PLL 锁定在正确的频率。

应变方法

TI 建议连续重试 PLL 锁定序列，直到 PLL 处于锁定状态并经 DCC 验证在预期频率下工作。

锁定序列为：禁用 PLL、启动 PLL、等待将 LOCKS 位置位，以及使用双时钟比较器 (DCC) 验证 PLL 频率。观察到 PLL 在正确的频率下运行后，可以选择它作为 CPU 时钟源。

TI 建议在 C2000Ware v6.00.01 或更高版本中使用 SysCtl_setClock() 函数，该函数还包括此权变措施的实现，以重试并将 PLL 时钟设置为用户定义的限值。

有关 DCC 用法的详细信息，请参见 C2000Ware SysCtl_IsPLLValid() 函数。如果器件无法正常工作，也可以由监控器在系统级别复位该器件，从而应用该权变措施。

公告 **系统：多次连续写入 CLKSRCCTL1 可能会导致系统挂起**

受影响版本

0

详细信息

在 写入 CLKSRCCTL1 寄存器时，在写入操作之间多次未延迟，则系统可能会发生挂起的情况，并且只能通过外部 XRSn 复位或 Watchdog 复位才能恢复。在 SYSCLK 与 OSCCLKSRCSEL 所选时钟之间的时钟比有问题时，就会出现这种情况，然而，并非每次都会出现这个问题。

如果在使用调试器时遇到此问题，那么在点击暂停后，程序计数器将位于 Boot ROM 复位向量处。

实施此权变措施，可避免由 SYSCLK 与 OSCCLK 时钟比引起的这种情况。

权变措施

每次写入 CLKSRCCTL1 寄存器后，使用 NOP 指令添加 300 个 SYSCLK 周期的软件延迟。

示例：

```
ClkCfgRegs.CLKSRCCTL1.bit.INTOSC2OFF=0;          // Turn on INTOSC2
asm("   RPT #250 || NOP");                       // Delay of 250 SYSCLK Cycles
asm("   RPT #50 || NOP");                        // Delay of 50 SYSCLK Cycles
ClkCfgRegs.CLKSRCCTL1.bit.OSCCLKSRCSEL = 0;      // Clk Src = INTOSC2
asm("   RPT #250 || NOP");                       // Delay of 250 SYSCLK Cycles
asm("   RPT #50 || NOP");                        // Delay of 50 SYSCLK Cycles
```

C2000Ware_3_00_00_00 和更高版本将实施此权变措施。

公告	***看门狗：WDKEY 寄存器不受 EALLOW 保护***
受影响版本	0
详细信息	WDKEY 寄存器不受 EALLOW 保护。无需发出 EALLOW 和 EDIS 指令来写入该寄存器。要在 WDKEY 受 EALLOW 保护的其他器件上启用软件重用，建议使用 EALLOW 和 EDIS。
权变措施	无

公告	***看门狗：写入 WDHALTI 位会影响 PLL 锁定状态***
受影响版本	0
详细信息	一旦系统 PLL 被锁定 (SYSPLLSTS.LOCKS = 1)，对 CLKSRCCTL1.WDHALTI 进行写入会使 PLL 解锁 (SYSPLLSTS.LOCKS = 0)。
权变措施	在锁定 PLL 之前执行 WDHALTI 配置。

4 文档支持

欲获得器件专用数据表和相关文档，请访问 TI 网站：https://www.ti.com。

欲了解有关 F28E12x 器件的更多信息，请参阅以下文档：

- *F28E12x 实时微控制器数据表*
- *F28E12x 实时微控制器技术参考手册*

5 商标

所有商标均为其各自所有者的财产。

6 修订历史记录

Changes from JULY 28, 2025 to OCTOBER 31, 2025 (from Revision * (July 2025) to Revision 0 (October 2025)) **Page**

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月