Errata

TMS320F28004x Real-Time MCUs Silicon Errata Silicon Revisions B, A, 0

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories). This document may also contain usage notes. Usage notes describe situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness.

Table of Contents

List of Figures

List of Tables

1 Usage Notes and Advisories Matrices

Table 1-1 lists all usage notes and the applicable silicon revisions. Table 1-2 lists all advisories, modules affected, and the applicable silicon revisions.

1.1 Usage Notes Matrix

Table 1-1. Usage Notes Matrix

NUMBER	TITLE	SILICON REVISIONS AFFECTED		
		0	A	B
Section 3.1.1	PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear	Yes	Yes	Yes
Section 3.1.2	FPU32 and VCU Back-to-Back Memory Accesses	Yes	Yes	Yes
Section 3.1.3	Caution While Using Nested Interrupts	Yes	Yes	Yes
Section 3.1.4	Security: The primary layer of defense is securing the boundary of the chip, which begins with enabling JTAGLOCK and Zero-pin Boot to Flash feature	Yes	Yes	Yes

1.2 Advisories Matrix

Table 1-2. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED		
		0	A	B
FPU	FPU: FPU-to-CPU Register Move Operation Preceded by Any FPU 2p Operation	Yes	Yes	Yes
SDFM	SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events	Yes	Yes	Yes
SDFM	SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events	Yes	Yes	Yes
SDFM	SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events	Yes	Yes	Yes
SDFM	SDFM: Manchester Mode (Mode 2) Does Not Produce Correct Filter Results Under Several Conditions	Yes	Yes	Yes
eQEP	eQEP: Position Counter Incorrectly Reset on Direction Change During Index	Yes	Yes	Yes
eQEP	eQEP: eQEP Inputs in GPIO Asynchronous Mode	Yes	Yes	Yes
V_{DD} Supply:	V_{DD} Supply: During V_{DDIO} Power Up, V_{DD} May Also Rise	Yes	Yes	Yes
eCAP	eCAP: HRFRC is Not EALLOW-Protected	Yes	Yes	Yes
PLL	PLL: PLL May Not Lock on the First Lock Attempt	Yes	Yes	Yes
LPM	LPM: STANDBY Low-Power Mode is Not Supported	Yes	Yes	Yes
DCC	DCC: Single-Shot-Mode Operation May End Prematurely	Yes	Yes	Yes
BOR	BOR: VDDIO Between 2.45 V and 3.0 V can Result in Multiple XRSn Pulses	Yes	Yes	Yes
I2C	I2C: SDA and SCL Open-Drain Output Buffer Issue	Yes	Yes	Yes
I2C	I2C: Target Transmitter Mode, Standard Mode SDA Timings Limitation	Yes	Yes	Yes
ePWM	ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window	Yes	Yes	Yes
ePWM	ePWM: Trip Events Will Not be Filtered by the Blanking Window for the First 3 Cycles After the Start of a Blanking Window	Yes	Yes	Yes
ePWM	ePWM: Event Latch (DCxEVTxLAT) of "DC Event-Based CBC Trip" May not Extend Trigger Pulse as Expected When Asynchronous Path is Selected	Yes	–	–
INTOSC	INTOSC: VDDIO Powered Without VDD Can Cause INTOSC Frequency Drift	Yes	Yes	Yes
FSI	FSI: RX FIFO Spurious Overrun	Yes	Yes	Yes
	During DCAN FIFO Mode, Received Messages May be Placed Out of Order in the FIFO Buffer	Yes	Yes	Yes
Boot ROM	Boot ROM: Calling SCI Bootloader from Application	Yes	Yes	Yes

Table 1-2. Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED		
		0	A	B
Boot ROM, MPOST	Boot-ROM, MPOST: Longer Boot Time With MPOST Enabled	Yes	Yes	Yes
Memory	Memory: Prefetching Beyond Valid Memory	Yes	Yes	Yes
SYSTEM	SYSTEM: Multiple Successive Writes to CLKSRCCTL1 Can Cause a System Hang	Yes	Yes	Yes
ADC	ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set	Yes	Yes	Yes
ADC	ADC: Degraded ADC Performance With ADCCLK Fractional Divider	Yes	Yes	Yes
ADC	ADC: DMA Read of Stale Result	Yes	Yes	Yes
CLB	CLB: Back-to-Back PUSH or PULL Instructions With More Than One Active High Level Controller (HLC) Channel is not Supported	Yes	Yes	Yes
Analog Subsystem	Analog Subsystem: Software Configuration for Shared Reference Pins	Yes	Yes	–
	Analog Trim of Some TMX Devices	Yes	Yes	–
PGA	PGA: Output Filter Path is Not Supported	Yes	Yes	–
ROM	ROM: Flash API Library and FPU32 Twiddle Factor RFFT Table Not Present	Yes	Yes	–
REVID	REVID: Some TMX Revision A Devices Have an Incorrect REVID Value	–	Yes	–
GPIO	GPIO: X2/GPIO18 Pin Pullup Current During Power Up	Yes	Yes	Yes
GPIO	GPIO: Open-Drain Configuration May Drive a Short High Pulse	Yes	Yes	Yes
GPIO	GPIO: Parasitic Path to V_{SS} When Maximum V_{IH} is Exceeded in Input Mode	–	Yes	–
GPIO	GPIO: Pins may Drive High During Power Up	Yes	–	–
GPIO	GPIO: Signal Latch-up to V_{SS}	Yes	–	–
LIN	LIN: Inconsistent Sync Field Error (ISFE) Flag/Interrupt Not Set When Sync Field is Erroneous	Yes	Yes	Yes

2 Nomenclature, Package Symbolization, and Revision Identification

2.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all TMS320 MCU devices and support tools. Each TMS320™ MCU commercial family member has one of three prefixes: TMX, TMP, or TMS (for example, **TMS**320F280049). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (with TMX for devices and TMDX for tools) through fully qualified production devices and tools (with TMS for devices and TMDS for tools).

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

TMP Prototype device that is not necessarily the final silicon die and may not necessarily meet final electrical specifications.

TMS Production version of the silicon die that is fully qualified.

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully-qualified development-support product.

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Production devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (X or P) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, PZ) and temperature range (for example, S).

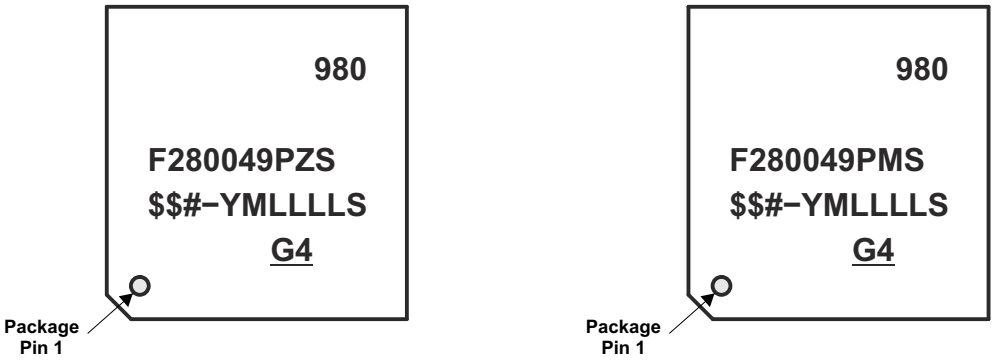
2.2 Devices Supported

This document supports the following devices:

- TMS320F280049
- TMS320F280049-Q1
- TMS320F280049C
- TMS320F280049C-Q1
- TMS320F280048-Q1
- TMS320F280048C-Q1
- TMS320F280045
- TMS320F280041
- TMS320F280041-Q1
- TMS320F280041C
- TMS320F280041C-Q1
- TMS320F280040-Q1
- TMS320F280040C-Q1

2.3 Package Symbolization and Revision Identification

Figure 2-1 and Figure 2-2 provide examples of the F28004x device markings and define each of the markings. The device revision can be determined by the symbols marked on the top of the package as shown in Figure 2-1 and Figure 2-2. Some prototype devices may have markings different from those illustrated. Figure 2-3 shows the device nomenclature.

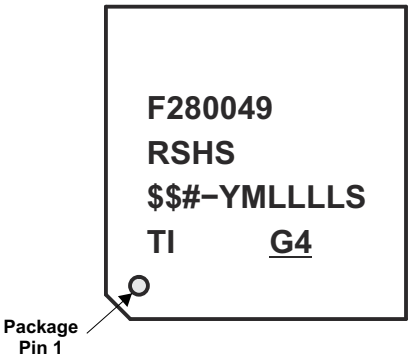


YMLLLLS = Lot Trace Code

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

G4 = Green (Low Halogen and RoHS-compliant)

Figure 2-1. Examples of Device Markings for PM and PZ Packages



YMLLLLS = Lot Trace Code

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

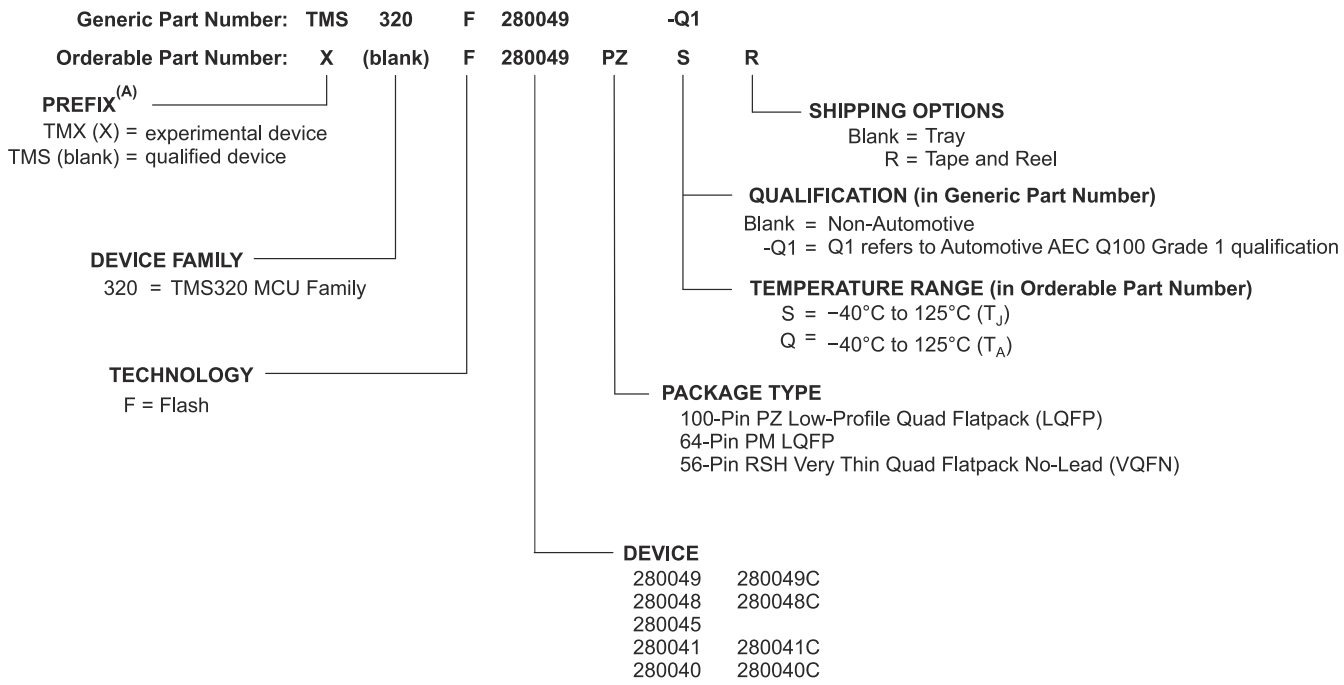
G4 = Green (Low Halogen and RoHS-compliant)

Figure 2-2. Example of Device Markings for RSH Package

Table 2-1. Determining Silicon Revision From Lot Trace Code

SILICON REVISION CODE	SILICON REVISION	REVID(1) Address: 0x5D00C	COMMENTS(2)
Blank	0	0x0000 0000	This silicon revision is available as TMX.
A	A	0x0000 0001	This silicon revision is available as TMX.
B	B	0x0000 0002	This silicon revision is available as TMX and TMS.

(1) Silicon Revision ID
(2) For orderable device numbers, see the PACKAGING INFORMATION table in the *TMS320F28004x Real-Time Microcontrollers* data sheet.



A. Prefix X is used in orderable part numbers.

Figure 2-3. Example of Device Nomenclature

3 Silicon Revision B Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

3.1 Silicon Revision B Usage Notes

This section lists all the usage notes that are applicable to silicon revision B [and earlier silicon revisions].

3.1.1 PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear

Revisions Affected: 0, A, B

Certain code sequences used for nested interrupts allow the CPU and PIE to enter an inconsistent state that can trigger an unwanted interrupt. The conditions required to enter this state are:

1. A PIEACK clear is followed immediately by a global interrupt enable (EINT or asm(" CLRC INTM")).
2. A nested interrupt clears one or more PIEIER bits for its group.

Whether the unwanted interrupt is triggered depends on the configuration and timing of the other interrupts in the system. This is expected to be a rare or nonexistent event in most applications. If it happens, the unwanted interrupt will be the first one in the nested interrupt's PIE group, and will be triggered after the nested interrupt re-enables CPU interrupts (EINT or asm(" CLRC INTM")).

Workaround: Add a NOP between the PIEACK write and the CPU interrupt enable. Example code is shown below.

```
//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 FPU32 and VCU Back-to-Back Memory Accesses

Revisions Affected: 0, A, B

This usage note applies when a VCU memory access and an FPU memory access occur back-to-back. There are three cases:

Case 1. Back-to-back memory reads: one read performed by a VCU instruction (VMOV32) and one read performed by an FPU32 instruction (MOV32).

If an R1 pipeline phase stall occurs during the first read, then the second read will latch the wrong data. If the first instruction is not stalled during the R1 pipeline phase, then the second read will occur properly.

The order of the instructions—FPU followed by VCU or VCU followed by FPU—does not matter. The address of the memory location accessed by either read does not matter.

Case 1 Workaround: Insert one instruction between the two back-to-back read instructions. Any instruction, except a VCU or FPU memory read, can be used.

Case 1, Example 1:

```
VMOV32  VR1,mem32       ; VCU memory read
NOP                     ; Not a FPU/ VCU memory read
MOV32   R0H,mem32       ; FPU memory read
```

Case 1, Example 2:

```
VMOV32  VR1,mem32       ; VCU memory read
VMOV32  mem32, VR2      ; VCU memory write
MOV32   R0H,mem32       ; FPU memory read
```

Case 2. Back-to-back memory writes: one write performed by a VCU instruction (VMOV32) and one write performed by an FPU instruction (MOV32).

If a pipeline stall occurs during the first write, then the second write can corrupt the data. If the first instruction is not stalled in the write phase, then no corruption will occur.

The order of the instructions—FPU followed by VCU or VCU followed by FPU—does not matter. The address of the memory location accessed by either write does not matter.

Case 2 Workaround: Insert two instructions between the back-to-back VCU and FPU writes. Any instructions, except VCU or FPU memory writes, can be used.

Case 2, Example 1:

```
VMOV32  mem32,VR0       ; VCU memory write
NOP                     ; Not a FPU/VCU memory write
NOP                     ; Not a FPU/VCU memory write
MOV32   mem32,R3H       ; FPU memory write
```

Case 2, Example 2:

```
VMOV32  mem32,VR0       ; VCU memory write
VMOV32  VR1,  mem32     ; VCU memory read
NOP
MOV32   mem32,R3H       ; FPU memory write
```

Case 3. Back-to-back memory writes followed by a read or a memory read followed by a write. In this case, there is no interaction between the two instructions. No action is required.

Workaround: See Case 1 Workaround and Case 2 Workaround.

3.1.3 Caution While Using Nested Interrupts

Revisions Affected: 0, A, B

If the user is enabling interrupts using the EINT instruction inside an interrupt service routine (ISR) in order to use the nesting feature, then the user must disable the interrupts before exiting the ISR. Failing to do so may cause undefined behavior of CPU execution.

3.1.4 Security: The primary layer of defense is securing the boundary of the chip, which begins with enabling JTAGLOCK and Zero-pin Boot to Flash feature

Revisions Affected: 0, A, B

Device security relies on the premise that unauthorized code is not allowed to enter the device and execute under any circumstances. To that end, the device provides two features that a user concerned about security should always enable.

- **JTAGLOCK**

  When enabled in the USER OTP area of flash, the JTAGLOCK feature disables JTAG access (for example, debugger connection) to resources on the device, blocking an unauthorized party from using the JTAG interface to download any code into the device. When JTAGLOCK is enabled, the user can still allow an authorized party to unlock it by entering a password, or they can lock it permanently by programming a password value of all all-zeros.

- **Zero-pin Boot to Flash**

  The external bootloaders built into the TI ROM do not perform any authentication of the downloaded code. Enabling the Zero-pin boot option along with a flash boot mode in the USER OTP blocks all pin-based external bootloader options (for example, SCI, CAN, Parallel) from running at boot by forcing the boot process to jump immediately to internal flash after the base boot ROM execution concludes. For highest security, the Secure Flash boot mode can be chosen. This enables a pre-check of the flash code by the base boot ROM before jumping to it.

If JTAG is locked permanently and the Zero-pin Boot to Flash option is enabled, programming tools that communicate with the device through JTAG or the built-in bootloaders will not work. If the ability to perform firmware upgrades is desired, the user must pre-store code in flash to securely manage and perform the update.

3.2 Silicon Revision B Advisories

This section lists all the advisories that are applicable to silicon revision B [and earlier silicon revisions].

Advisory ***FPU: FPU-to-CPU Register Move Operation Preceded by Any FPU 2p Operation***

Revisions Affected 0, A, B

Details

This advisory applies when a multicycle (2p) FPU instruction is followed by a FPU-to-CPU register transfer. If the FPU-to-CPU read instruction source register is the same as the 2p instruction destination, then the read may be of the value of the FPU register before the 2p instruction completes. This occurs because the 2p instructions rely on data-forwarding of the result during the E3 phase of the pipeline. If a pipeline stall happens to occur in the E3 phase, the result does not get forwarded in time for the read instruction.

The 2p instructions impacted by this advisory are MPYF32, ADDF32, SUBF32, and MACF32. The destination of the FPU register read must be a CPU register (ACC, P, T, XAR0...XAR7). This advisory does not apply if the register read is a FPU-to-FPU register transfer.

In the example below, the 2p instruction, MPYF32, uses R6H as its destination. The FPU register read, MOV32, uses the same register, R6H, as its source, and a CPU register as the destination. If a stall occurs in the E3 pipeline phase, then MOV32 will read the value of R6H before the MPYF32 instruction completes.

Example of Problem:

```
   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
```

Figure 3-1 shows the pipeline diagram of the issue when there are no stalls in the pipeline.

	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	

Figure 3-1. Pipeline Diagram of the Issue When There are no Stalls in the Pipeline

Advisory (continued) ***FPU: FPU-to-CPU Register Move Operation Preceded by Any FPU 2p Operation***

Figure 3-2 shows the pipeline diagram of the issue if there is a stall in the E3 slot of the instruction I1.

	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

Figure 3-2. Pipeline Diagram of the Issue if There is a Stall in the E3 Slot of the Instruction I1

Workaround

Treat MPYF32, ADDF32, SUBF32, and MACF32 in this scenario as 3p-cycle instructions. Three NOPs or non-conflicting instructions must be placed in the delay slot of the instruction.

The C28x Code Generation Tools v.6.2.0 and later will both generate the correct instruction sequence and detect the error in assembly code. In previous versions, v6.0.5 (for the 6.0.x branch) and v.6.1.2 (for the 6.1.x branch), the compiler will generate the correct instruction sequence but the assembler will not detect the error in assembly code.

Example of Workaround:

```
   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
```

Figure 3-3 shows the pipeline diagram with the workaround in place.

Advisory (continued) ***FPU: FPU-to-CPU Register Move Operation Preceded by Any FPU 2p Operation***

	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	

Figure 3-3. Pipeline Diagram With Workaround in Place

Advisory	***SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events***
Revisions Affected	0, A, B
Details	When SDFM comparator settings—such as filter type, lower/upper threshold, or comparator OSR (COSR) settings—are dynamically changed during run time, spurious comparator events will be triggered. The spurious comparator event will trigger a corresponding CPU interrupt, CLA task, ePWM X-BAR events, and GPIO output X-BAR events if configured appropriately.
Workaround	When comparator settings need to be changed dynamically, follow the procedure below to ensure spurious comparator events do not generate a CPU interrupt, CLA task, or X-BAR events (ePWM X-BAR/GPIO output X-BAR events): 1. Disable the comparator filter. 2. Delay for at least a latency of the comparator filter + 3 SD-Cx clock cycles. 3. Change comparator filter settings such as filter type, COSR, or lower/upper threshold. 4. Delay for at least a latency of the comparator filter + 5 SD-Cx clock cycles. 5. Enable the comparator filter.

Advisory	***SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events***
Revisions Affected	0, A, B
Details	When SDFM data settings—such as filter type or DOSR settings—are dynamically changed during run time, spurious data-filter-ready events will be triggered. The spurious data-ready event will trigger a corresponding CPU interrupt, CLA task, and DMA trigger if configured appropriately.
Workaround	When SDFM data filter settings need to be changed dynamically, follow the procedure below to ensure spurious data-filter-ready events are not generated: 1. Disable the data filter. 2. Delay for at least a latency of the data filter + 3 SD-Cx clock cycles. 3. Change data filter settings such as filter type and DOSR. 4. Delay for at least a latency of the data filter + 5 SD-Cx clock cycles. 5. Enable the data filter.

Advisory ***SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events***

Revisions Affected 0, A, B

Details

Back-to-back writes to SDCPARMx register bit fields CEVT1SEL, CEVT2SEL, and HZEN within three SD-modulator clock cycles can potentially corrupt the SDFM state machine, resulting in spurious comparator events, which can potentially trigger CPU interrupts, CLA tasks, ePWM XBAR events, and GPIO output X-BAR events if configured appropriately.

Workaround

Avoid back-to-back writes within three SD-modulator clock cycles or have the SDCPARMx register bit fields configured in one register write.

Advisory ***SDFM: Manchester Mode (Mode 2) Does Not Produce Correct Filter Results Under Several Conditions***

Revisions Affected 0, A, B

Details

The Manchester decoding algorithm samples the Manchester bitstream with SYSCLK in a calibration window of 1024 SDx_Dy signal transitions. The derived clock from the Manchester bitstream is used to sample for data in the subsequent calibration window cycle.

There are several scenarios that can cause large errors in the filter results:

- Any single noise event on SDx_Dy can corrupt the decoded Manchester clock and cause subsequent data to be sampled at an incorrect frequency.
- If the Manchester bitstream clock rate is a near exact integer multiple of SYSCLK, then an occasional Manchester bit can be skipped when the phases of the Manchester stream and internal SYSCLK drift past each other in phase before the next 1024 transition calibration window becomes effective. Deviations in duty cycle from 50% of the Manchester clock also need to be accounted for to ensure the longer Manchester pulses are not an integer multiple of SYSCLK. This situation can be unavoidable if the clock sources for either the SD modulator or this device have a wide variation since a wide range of keep out frequencies become problematic
- If the Manchester edge delay variation between rising and falling (duty cycle of the bitstream) is greater than one SYSCLK, then the SDFM clock decode algorithm can incorrectly identify the clock period as shorter than it is.

Workarounds

The workarounds available are:

- Avoid using Manchester mode and consider using Mode 0, which provides the best filter performance under noisy conditions. This is the recommended workaround.
- Avoid any noise on the Manchester bitstream and avoid integer multiples of SYSCLK for the selected Manchester clock source. A precision clock source for the modulator and this device must be used.
- Ensure rising and falling edge delays (high and low pulses) are within one SYSCLK of each other in length.
- Design an application-level algorithm that is robust against occasional incorrect SDFM results.

Advisory	***eQEP: Position Counter Incorrectly Reset on Direction Change During Index***
Revisions Affected	0, A, B
Details	While using the PCRM = 0 configuration, if the direction change occurs when the index input is active, the position counter (QPOSCNT) could be reset erroneously, resulting in an unexpected change in the counter value. This could result in a change of up to ±4 counts from the expected value of the position counter and lead to unexpected subsequent setting of the error flags. While using the PCRM = 0 configuration [that is, Position Counter Reset on Index Event (QEPCTL[PCRM] = 00)], if the index event occurs during the forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation. If the direction change occurs while the index pulse is active, the module would still continue to look for the relative quadrature transition for performing the position counter reset. This results in an unexpected change in the position counter value. The next index event without a simultaneous direction change will reset the counter properly and work as expected.
Workaround	Do not use the PCRM = 0 configuration if the direction change could occur while the index is active and the resultant change of the position counter value could affect the application. Other options for performing position counter reset, if appropriate for the application [such as Index Event Initialization (IEI)], do not have this issue.

Advisory	***eQEP: eQEP Inputs in GPIO Asynchronous Mode***
Revisions Affected	0, A, B
Details	If any of the eQEP input pins are configured for GPIO asynchronous input mode via the GPxQSELn registers, the eQEP module may not operate properly because the eQEP peripheral assumes the presence of external synchronization to SYSCLKOUT on inputs to the module. For example, QPOSCNT may not reset or latch properly, and pulses on the input pins may be missed. For proper operation of the eQEP module, input GPIO pins should be configured via the GPxQSELn registers for synchronous input mode (with or without qualification), which is the default state of the GPxQSEL registers at reset. All existing eQEP peripheral examples supplied by TI also configure the GPIO inputs for synchronous input mode. The asynchronous mode should not be used for eQEP module input pins.
Workaround	Configure GPIO inputs configured as eQEP pins for non-asynchronous mode (any GPxQSELn register option except “11b = Asynchronous”).

Advisory ***V_{DD} Supply: During V_{DDIO} Power Up, V_{DD} May Also Rise***

Revisions Affected

0, A, B

Details

A leakage current from V_{DDIO} to V_{DD} is present when the V_{DD} supply is below approximately 0.5 V. This causes the V_{DD} voltage to rise to approximately 0.5 V when V_{DDIO} is powered. This is observed when the device is configured to use either the internal VREG (VREGENZ tied to V_{SS}) or an external 1.2-V regulator (VREGENZ tied to V_{DDIO}), and there is a significant delay (about 1 ms) between the power up of V_{DDIO} and V_{DD} from external regulators or the ramp time of V_{DDIO} is greater than 1 ms when in internal VREG mode.

This does not impact device functionality once the external 1.2-V or internal 1.2-V supply begins to ramp. See the *TMS320F28004x Real-Time Microcontrollers* data sheet for power sequencing requirements.

Workaround

If this early voltage on V_{DD} is a problem for system-level supervisor circuits, then minimize the delay between ramping the 3.3-V V_{DDIO} and 1.2-V V_{DD} rails. If the internal VREG is used, decrease the ramp time of the 3.3-V V_{DDIO} supply to 1 ms or less.

Advisory	***eCAP: HRFRC is Not EALLOW-Protected***
Revisions Affected	0, A, B
Details	The HRFRC register is not EALLOW-protected. Issuing the EALLOW and EDIS instructions to write to this register is not required. To enable software reuse on other devices where HRFRC is EALLOW-protected, using EALLOW and EDIS is recommended.
Workaround	None

Advisory	***PLL: PLL May Not Lock on the First Lock Attempt***
Revisions Affected	0, A, B
Details	The PLL may not start properly at device power up. The PLLSTS[LOCKS] bit is set, but the PLL does not produce a clock. Once the PLL has started properly, the PLL can be disabled and reenabled with no issues and will stay locked. However, the PLL lock problem could reoccur on a subsequent power-up cycle. If the SYSPLL has not started properly and is selected as the CPU clock source, the CPU will stop executing instructions. The occurrence rate of this transient issue is low. After an initial occurrence, this issue may not be subsequently observed in the system again. Implementation of the workaround reduces the rate of occurrence.
Workaround	TI recommends doing lock sequences in succession until the PLL is in locked state when the PLL is configured for the first time after power up. The lock sequence is: disable the PLL, start the PLL, wait for the LOCKS bit to set, and validate the PLL frequency using the Dual Clock Comparator (DCC). After the PLL is observed to be running, it can be selected as the CPU clock source. TI recommends using the C2000Ware SysCtl_setClock() function, which also includes implementation of this workaround, to set the PLL clock. Details on DCC usage are in the C2000Ware SysCtl_IsPLLValid() function. The workaround can also be applied at the system level by a supervisor resetting the device if it is not responding.

Advisory	***LPM: STANDBY Low-Power Mode is Not Supported***
Revisions Affected	0, A, B
Details	The STANDBY low-power mode is not supported.
Workaround	The IDLE or HALT low-power modes can be used for power reduction. See the *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual* for information on implementing these modes. If IDLE is used, additional power reduction can be optionally achieved through software by one or all of these methods: • Decrease the SYSCLK frequency: – Change the SYSCLK source to OSCCLK by configuring SYSPLLCTL1[PLLCLKEN] = 0. – Change the SYSCLKDIVSEL register to a higher divider. • Disable peripheral clocks through the PCLKCRx register.

Advisory	***DCC: Single-Shot-Mode Operation May End Prematurely***
Revisions Affected	0, A, B
Details	In single-shot mode, DCCSTATUS[DONE] or DCCSTATUS[ERROR] may be prematurely set. When this occurs, DCC results are invalid.
Workaround	Any of the following conditions ends DCC operation prematurely. TI recommends rerunning DCC if *any* of the below conditions are met. • DCCSTATUS[DONE] = 1 and (DCCCNT1 > 0 or DCCCNT0 > 0 or DCCVALID0 > 0) • DCCSTATUS[ERROR] = 1 and DCCCNT1 > 0 and DCCVALID0 > 0

Advisory ***BOR: VDDIO Between 2.45 V and 3.0 V can Result in Multiple XRSn Pulses***

Revisions Affected

0, A, B

Details

The BOR can generate repeating XRSn assertions and deassertions when the VDDIO supply voltage is between 2.45 V and 3.0 V. It is recommended that the XRSn pin *not* be used directly as a reset to any other devices in the system.

The F28004x BOR is effective for internally holding the device in a known reset state, even when these XRSn pulses are occurring. The device will not branch to application code or bootloaders, and all other pins will be held in their reset state until the VDDIO supply rises above 3.0 V.

Workarounds

1. Ignore the extra XRSn transitions during power up, power down, and BOR events. The extra XRSn pulses will have no effect on the F28004x device operation itself.
2. If XRSn pulses would cause undesired system behavior with other system components, then do not use XRSn to drive other devices. An external voltage supervisor can be used for these applications.
3. For applications that need to avoid these pulses during normal power up and power down:
   a. Power up: Follow the $t_{VDDIO\text{-}RAMP}$ requirement in the Recommended Operating Conditions table of the *TMS320F28004x Real-Time Microcontrollers* data sheet; no extra XRSn low pulses will occur.
   b. Power Down: To avoid any deassertion of XRSn during power down, design the power supply so that VDDIO passes through the range from 3.0 V to 2.45 V within 25 µs. If some voltage rise on XRSn is acceptable, then the time constant of the RC circuit implemented on XRSn can be calculated to ensure the voltage does not rise above a system-specified threshold.

Advisory ***I2C: SDA and SCL Open-Drain Output Buffer Issue***

Revisions Affected

0, A, B

Details

The SDA and SCL outputs are implemented with push-pull 3-state output buffers rather than open-drain output buffers as required by I2C. While it is possible for the push-pull 3-state output buffers to behave as open-drain outputs, an internal timing skew issue causes the outputs to drive a logic-high for a duration of 0–5 ns before the outputs are disabled. The unexpected high-level pulse will only occur when the SCL or SDA outputs transition from a driven low state to a high-impedance state and there is sufficient internal timing skew on the respective I2C output.

This short high-level pulse injects energy in the I2C signals traces, which causes the I2C signals to sustain a period of ringing as a result of multiple transmission line reflections. This ringing should not cause an issue on the SDA signal because it only occurs at times when SDA is expected to be changing logic levels and the ringing will have time to damp before data is latched by the receiving device. The ringing may have enough amplitude to cross the SCL input buffer switching threshold several times during the first few nanoseconds of this ringing period, which may cause clock glitches. This ringing should not cause a problem if the amplitude is damped within the first 50 ns because I2C devices are required to filter their SCL inputs to remove clock glitches. Therefore, it is important to design the PCB signal traces to limit the duration of the ringing to less than 50 ns. One possible solution is to insert series termination resistors near the SCL and SDA terminals to attenuate transmission line reflections.

This issue may also cause the SDA output to be in contention with the slave SDA output for the duration of the unexpected high-level pulse when the slave begins its ACK cycle. This occurs because the slave may already be driving SDA low before the unexpected high-level pulse occurs. The glitch that occurs on SDA as a result of this short period of contention does not cause any I2C protocol issue but the peak current applies unwanted stress to both I2C devices and potentially increases power supply noise. Therefore, a series termination resistor located near the respective SDA terminal is required to limit the current during the short period of contention.

A similar contention problem can occur on SCL when connected to I2C slave devices that support clock stretching. This occurs because the slave is driving SCL low before the unexpected high-level pulse occurs. The glitch that occurs on SCL as a result of this short period of contention does not cause any I2C protocol issue because I2C devices are required to apply a glitch filter to their SCL inputs. However, the peak current applies unwanted stress to both I2C devices and potentially increases power supply noise. Therefore, a series termination resistor located near the respective SCL terminal is required to limit the current during the short period of contention.

If another master is connected, the unexpected high-level pulses on the SCL and SDA outputs can cause contention during clock synchronization and arbitration. The series termination resistors described above will also limit the contention current in this use case without creating any I2C protocol issue.

Workaround

Insert series termination resistors on the SCL and SDA signals and locate them near the SCL and SDA terminals. The SCL and SDA pullup resistors should also be located near the SCL and SDA terminals. The placement of the series termination resistor and pullup resistor should be connected as shown in Figure 3-4.

Advisory (continued) ***I2C: SDA and SCL Open-Drain Output Buffer Issue***

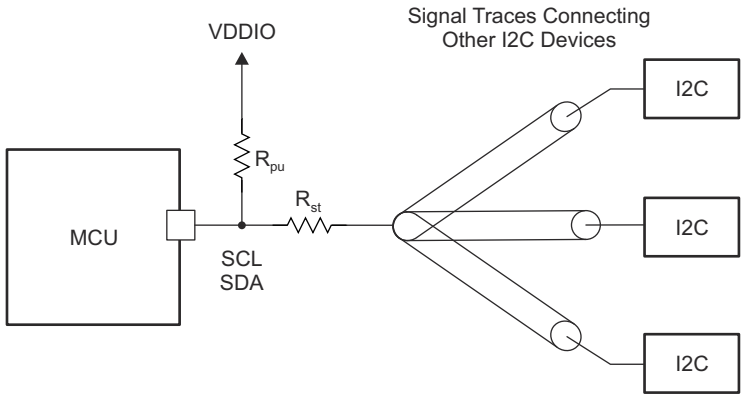


Figure 3-4. Placement of Series Termination Resistor and Pullup Resistor

Advisory ***I2C: Target Transmitter Mode, Standard Mode SDA Timings Limitation***

Revision Affected 0, A, B

Details

The I2C peripheral present on the MCU is a Fast-mode device; it will clock-stretch the SCL (Clock) line when used with a Standard-mode host.

There is a requirement from the I2C Specification for a Fast-mode device used in a Standard-mode system to meet $t_{SU:DAT}$ (data set-up time) + $t_{r(max)}$ (rise time) before releasing the SCL line. See Footnote 4 of the "Characteristics of the SDA and SCL bus lines for Standard, Fast, and Fast-mode Plus I^2C-bus devices" table in the NXP Semiconductors *I^2C-bus specification and user manual* (UM10204).

However, the C2000 I2C clock-stretches the SCL line by a fixed amount = 6 * f_{mod} Clock (I2C Clock rate of the C2000) in the above scenario. When the C2000™ microcontroller is acting as a target transmitter with a Standard-mode host, it is possible for the clock line (SCL) to be released by the C2000 before the data (SDA) is ready, if the t_r of SDA is too long.

The "Pull-up resistor sizing" section in the NXP Semiconductors *I^2C-bus specification and user manual* (UM10204) gives more details on choosing the appropriate PU resistor (R_p), based on the rise time (t_r) and bus capacitance (C_b) shown in Equation 1.

$$R_{p(max)} = \frac{t_r}{0.8473 \times C_b} \quad (1)$$

Workaround

1. **Reducing t_r with a strong pullup**
   In order to ensure that $t_{SU:DAT}$ + $t_{r(max)}$ is met, the user can configure the pullup resistance on the SDA line such that it meets the constraints listed in the SDA Data Rise Time Requirement column of Table 3-1 based on the value of f_{mod} Clock in their system. This will ensure that the data present on the SDA line is valid when the C2000 releases the SCL signal.

   Table 3-2 gives suggested R_p resistor values for a given f_{mod} Clock (MHz) and C_b (bus capacitance). For other values of C_b, please use Equation 1 to calculate the value of R_p needed in the system.

Table 3-1. Data Rise Time Requirements for C2000 as Target Transmitter with Standard-Mode Host

f_{mod} Clock (MHz)	f_{mod} Period (ns)	SCL Clock-Stretch Delay from C2000 I2C (ns): (6*f_{mod} Clock)	Data Set-up Time (ns): $t_{SU:DAT}$ (Standard Mode)	SDA Data Rise Time Requirement (ns): t_r
7	142.9	857	250	607
8	125	750		500
9	111	666		416
10	100	600		350
11	90.9	545		295
12	83.3	500		250

Advisory (continued) ***I2C: Target Transmitter Mode, Standard Mode SDA Timings Limitation***

Table 3-2. Pullup Resistor (R_p) Values for Common Bus Capacitances (C_b)

f_{mod} Clock (MHz)	SDA Data Rise Time Requirement (ns): t_r	R_p (kΩ) for C_b = 100 pF	R_p (kΩ) for C_b = 200 pF	R_p (kΩ) for C_b = 300 pF	R_p (kΩ) for C_b = 400 pF
7	607	7.1	3.5	2.3	1.7
8	500	5.9	2.9	1.9	1.4
9	416	4.9	2.4	1.6	1.2
10	350	4.1	2.0	1.3	1.0
11	295	3.4	1.7	1.1	0.8
12	250	2.9	1.4	0.9	0.7

2. **t_r = 1000 ns**
   This workaround is not preferred due to restrictions in general I2C usage, use Workaround 1 when possible.
   If the system is such that it requires 1000 ns of rise time on the SDA line, the C2000 I2C f_{mod} Clock can be configured to 4.8 MHz so the clock-stretching (6 * f_{mod} Clock) satisfies this requirement. This results in t_r = (1/4.8 MHz) * 6 = 1000 ns. This workaround is only valid in systems where the C2000 I2C is the target on the I2C bus. Note that 4.8 MHz is outside the data sheet's required range of 7 MHz to 12 MHz for f_{mod} Clock. Using f_{mod} at 4.8 MHz, even though it is outside of the data sheet's required range, will work for the C2000 I2C in Target mode on a Standard-mode host bus. Using f_{mod} = 4.8 MHz in any other configurations except the one listed in this workaround will cause other timing parameters to be violated and is not allowed.

Advisory ***ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window***

Revisions Affected 0, A, B

Details

The blanking window is typically used to mask any PWM trip events during transitions which would be false trips to the system. If an ePWM trip event remains active for less than three ePWM clocks after the end of the blanking window cycles, there can be an undesired glitch at the ePWM output.

Figure 3-5 illustrates the time period which could result in an undesired ePWM output.

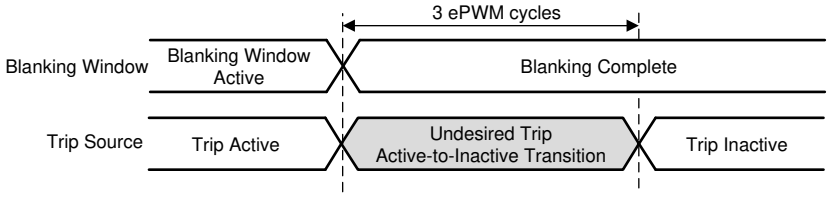


Figure 3-5. Undesired Trip Event and Blanking Window Expiration

Figure 3-6 illustrates the two potential ePWM outputs possible if the trip event ends within 1 cycle before or 3 cycles after the blanking window closes.

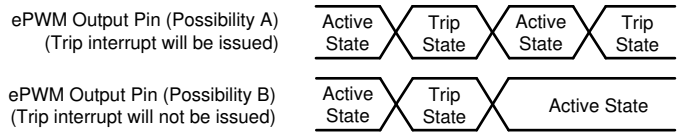


Figure 3-6. Resulting Undesired ePWM Outputs Possible

Workaround

Extend or reduce the blanking window to avoid any undesired trip action.

Advisory ***ePWM: Trip Events Will Not be Filtered by the Blanking Window for the First 3 Cycles After the Start of a Blanking Window***

Revisions Affected 0, A, B

Details

The Blanking Window will not blank trip events for the first 3 cycles after the start of a Blanking Window. DCEVTFILT may continue to reflect changes in the DCxEVTy signals. If DCEVTFILT is enabled, this may impact subsequent subsystems that are configured (for example, the Trip Zone submodule, TZ interrupts, ADC SOC, or the PWM output).

Workaround

Start the Blanking Window 3 cycles before blanking is required. If a Blanking Window is needed at a period boundary, start the Blanking Window 3 cycles before the beginning of the next period. This works because Blanking Windows persist across period boundaries.

Advisory	***INTOSC: VDDIO Powered Without VDD Can Cause INTOSC Frequency Drift***
Revisions Affected	0, A, B

Details

The "D" revision of the *TMS320F28004x Real-Time Microcontrollers* data sheet (SPRS945D) has updated power sequencing requirements. Revision "C" and earlier revisions of the data sheet did not require VDDIO and VDD to be powered on and powered off at the same time when using an external supply source for VDD.

If VDDIO is powered on while VDD is not powered, there will be an accumulating and persistent downward frequency drift for INTOSC1 and INTOSC2. The rate of drift accumulated will be greater when VDDIO is powered without VDD at high temperatures.

As a result of this drift, the INTOSC1 and INTOSC2 internal oscillator frequencies could fall below the minimum values specified in the data sheet. This would impact applications using INTOSC2 as the clock source for the PLL, with the system operating at a lower frequency than expected.

Workarounds

1. Use the internal VREG or internal DCDC, which will ensure VDD is powered when VDDIO is present.
2. When using an external VDD source, always keep VDDIO and VDD powered together.
3. Use the external X1 and X2 crystal oscillators as the PLL clock source. The crystal oscillator does not have any drift related to VDDIO and VDD supply sequencing.

Advisory ***FSI: RX FIFO Spurious Overrun***

Revisions Affected

0, A, B

Details

A buffer overrun is asserted when the last location of the FIFO is written.

Workarounds

Two possible workarounds are available.

1. Set up the communication between the transmitting and receiving modules in such a way that the maximum number of data words received, before the first data word is read, is 15 (not 16). Under this condition, buffer overrun behavior is reliable.
2. If the application must fill all 16 data words in the receive buffer before the first data word is read (NWORD packet with 16 words), then the following sequence can be used:
   - Ignore RX buffer overrun RX_EVT_STS.BUF_OVERRUN.
   - On RX_EVT_STS.FRAME_DONE, read RX_BUF_PTR_STS.CURR_WORD_CNT and check that it is 16.
   - Use DMA or software to move the data out of the RX buffer.
   - Read RX_BUF_PTR_STS.CURR_WORD_CNT and check that it is 0.
   - Clear the RX_EVT_STS.FRAME_DONE Flag by writing a 1 to RX_EVT_CLR.FRAME_DONE.

Advisory	***During DCAN FIFO Mode, Received Messages May be Placed Out of Order in the FIFO Buffer***
Revisions Affected	0, A, B
Details	In DCAN FIFO mode, received messages with the same arbitration and mask IDs are supposed to be placed in the FIFO in the order in which they are received. The CPU then retrieves the received messages from the FIFO via the IF1/IF2 interface registers. Some messages may be placed in the FIFO out of the order in which they were received. If the order of the messages is critical to the application for processing, then this behavior will prevent the proper use of the DCAN FIFO mode.
Workaround	Use the DMA to read out the FIFO via the IF3 register. Each time a message is received into the FIFO, the data is also copied to the IF3 register, and a DMA request Is generated to the DMA module to read out the data.

Advisory	***Boot ROM: Calling SCI Bootloader from Application***
Revisions Affected	0, A, B
Details	The ROM SCI bootloader uses autobaud lock to lock the baud rate. The SCI baud rate is split between two registers, SCILBAUD and SCIHBAUD. The ROM SCI bootloader expects SCIHBAUD to contain its default reset value of zero. If the ROM SCI bootloader is called from an application that modified the contents of SCIHBAUD to be non-zero, then the SCI will not autobaud-lock and the SCI bootloader will not execute.
Workaround	Clear SCIHBAUD to zero before calling the ROM SCI Bootloader.

Advisory	***Boot-ROM, MPOST: Longer Boot Time With MPOST Enabled***
Revisions Affected	0, A, B
Details	When MPOST functionality is enabled through user OTP, the boot-ROM execution time is around 5 seconds. This is because the boot-ROM executes MPOST with INTOSC clock of 10 MHz, which results in longer boot time.
Workaround	Avoid using boot-ROM supported MPOST and perform necessary memory self-tests as part of application initialization.

Advisory ***Memory: Prefetching Beyond Valid Memory***

Revisions Affected

0, A, B

Details

The C28x CPU prefetches instructions beyond those currently active in its pipeline. If the prefetch occurs past the end of valid memory, then the CPU may receive an invalid opcode.

Workarounds

M1, GS3 – The prefetch queue is 8 x16 words in depth. Therefore, code should not come within 8 words of the end of valid memory. Prefetching across the boundary between two valid memory blocks is all right.

Example 1: M1 ends at address 0x7FF and is not followed by another memory block. Code in M1 should be stored no farther than address 0x7F7. Addresses 0x7F8–0x7FF should not be used for code.

Example 2: M0 ends at address 0x3FF and valid memory (M1) follows it. Code in M0 can be stored up to and including address 0x3FF. Code can also cross into M1, up to and including address 0x7F7.

Flash – The prefetch queue is 16 x16 words in depth. Therefore, code should not come within 16 words of the end of valid memory; otherwise, it generates a Flash ECC uncorrectable error.

Table 3-3. Memories Impacted by Advisory

MEMORY TYPE	ADDRESSES IMPACTED
M1	0x0000 07F8–0x0000 07FF
GS3	0x0001 3FF8–0x0001 3FFF
Flash	0x0009 FFF0–0x0009 FFFF

Advisory ***SYSTEM: Multiple Successive Writes to CLKSRCCTL1 Can Cause a System Hang***

Revisions Affected

0, A, B

Details

When the CLKSRCCTL1 register is written more than once without delay between writes, the system can hang and can only be recovered by an external XRSn reset or Watchdog reset. The occurrence of this condition depends on the clock ratio between SYSCLK and the clock selected by OSCCLKSRCSEL, and may not occur every time.

If this issue is encountered while using the debugger, then after hitting pause, the program counter will be at the Boot ROM reset vector.

Implementing the workaround will avoid this condition for any SYSCLK to OSCCLK ratio.

Workaround

Add a software delay of 300 SYSCLK cycles using an NOP instruction after every write to the CLKSRCCTL1 register.

Example:

```
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 and later revisions will have this workaround implemented.

Advisory ***ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set***

Revisions Affected

0, A, B

Details

If ADCINTSELxNx[INTxCONT] = 0, then interrupts will stop when the ADCINTFLG is set and no additional ADC interrupts will occur.

When an ADC interrupt occurs simultaneously with a software write of the ADCINTFLGCLR register, the ADCINTFLG will unexpectedly remain set, blocking future ADC interrupts.

Workarounds

1. Use Continue-to-Interrupt Mode to prevent the ADCINTFLG from blocking additional ADC interrupts:

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

2. Ensure there is always sufficient time to service the ADC ISR and clear the ADCINTFLG before the next ADC interrupt occurs to avoid this condition.
3. Check for an overflow condition in the ISR when clearing the ADCINTFLG. Check ADCINTOVF immediately after writing to ADCINTFLGCLR; if it is set, then write ADCINTFLGCLR a second time to ensure the ADCINTFLG is cleared. The ADCINTOVF register will be set, indicating an ADC conversion interrupt was lost.

```
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
}
```

Advisory ***ADC: Degraded ADC Performance With ADCCLK Fractional Divider***

Revisions Affected

0, A, B

Details

Using fractional SYSCLK-to-ADCCLK dividers (controlled by the ADCCTL2.PRESCALE field) has been shown to cause degradation in ADC performance on this device. See Table 3-4.

Table 3-4. ADCCTL2 Register

REDUCED PERFORMANCE			
BIT	**FIELD**	**VALUE**	**DESCRIPTION**
3–0	PRESCALE	0001	ADCCLK = SYSCLK/1.5
		0003	ADCCLK = SYSCLK/2.5
		...	
NORMAL PERFORMANCE			
BIT	**FIELD**	**VALUE**	**DESCRIPTION**
3–0	PRESCALE	0000	ADCCLK = SYSCLK/1.0
		0002	ADCCLK = SYSCLK/2.0
		...	

Workaround

Use even PRESCALE clock divider values. Even PRESCALE values result in integer clock dividers which do not impact the ADC performance.

Advisory ***ADC: DMA Read of Stale Result***

Revisions Affected

0, A, B

Details

The ADCINT flag can be set before the ADCRESULT value is latched (see the t_{LAT} and $t_{INT(LATE)}$ columns in the ADC Timings table of the *TMS320F28004x Real-Time Microcontrollers* data sheet). The DMA can read the ADCRESULT value as soon as 3 cycles after the ADCINT trigger is set. As a result, the DMA could read a prior ADCRESULT value when the user expects the latest result if all of the following are true:

- The ADC is in late interrupt mode.
- The ADC operates in a mode where $t_{INT(LATE)}$ occurs 3 or more cycles before t_{LAT} (ADCCTL2 [PRESCALE] > 2).
- The DMA is triggered from the ADCINT signal.
- The DMA immediately reads the ADCRESULT value associated with that ADCINT signal without reading any other values first.
- The DMA was idle when it received the ADCINT trigger.

Only the DMA reads listed above could result in reads of stale data; the following non-DMA methods will always read the expected data:

- The ADCINT flag triggers a CLA task.
- The ADCINT flag triggers a CPU ISR.
- The CPU polls the ADCINT flag.

Workaround

Trigger two DMA channels from the ADCINT flag. The first channel acts as a dummy transaction. This will result in enough delay that the second channel will always read the fresh ADC result.

Advisory	***CLB: Back-to-Back PUSH or PULL Instructions With More Than One Active High Level Controller (HLC) Channel is not Supported***
Revisions Affected	0, A, B
Details	This issue only affects HLC operations if using back-to-back PUSH or PULL instructions with more than one active HLC channel.
Workaround	Reorder the HLC instructions so that every PUSH instruction is followed by a non-PUSH instruction. The same applies for the PULL instruction. If only one HLC channel is active, this issue does not occur.

Advisory ***GPIO: X2/GPIO18 Pin Pullup Current During Power Up***

Revisions Affected

0, A, B

Details

During power up, a pullup current of approximately 1.8 mA will be seen on X2/GPIO18. This pin will revert to input mode and operate per the pin description by the time XRSn releases (transitions to high).

Workaround

None

Advisory ***GPIO: Open-Drain Configuration May Drive a Short High Pulse***

Revisions Affected

0, A, B

Details

Each GPIO can be configured to an open-drain mode using the GPxODR register. However, an internal device timing issue may cause the GPIO to drive a logic-high for up to 0–10 ns during the transition into or out of the high-impedance state.

This undesired high-level may cause the GPIO to be in contention with another open-drain driver on the line if the other driver is simultaneously driving low. The contention is undesirable because it applies stress to both devices and results in a brief intermediate voltage level on the signal. This intermediate voltage level may be incorrectly interpreted as a high level if there is not sufficient logic-filtering present in the receiver logic to filter this brief pulse.

Workaround

If contention is a concern, do not use the open-drain functionality of the GPIOs; instead, emulate open-drain mode in software. Open-drain emulation can be achieved by setting the GPIO data (GPxDAT) to a static 0 and toggling the GPIO direction bit (GPxDIR) to enable and disable the drive low. For an example implementation, see the code below.

```
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;
}
```

Advisory ***LIN: Inconsistent Sync Field Error (ISFE) Flag/Interrupt Not Set When Sync Field is Erroneous***

Revisions Affected 0, A, B

Details

During LIN communications, if the Sync field received (on RX) is erroneous (that is, if the Sync field receives any value other than 0x55), the LIN does not set the ISFE Flag in the SCIFLR.ISFE register or trigger the ISFE interrupt. Communication gets terminated without data being received or the RX receive interrupt being set. There is no way for an application to detect an error in the Sync field. The application can detect if the Sync field is completely blank or if the Sync field is not received within the given tolerances (as explained in the *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*), but the application cannot detect any error in the value of Sync field.

Workarounds

Method 1: Keep polling the SCIFLR.RXRDY flag and time out if it is not set within a certain amount of time.

Use the following steps as a guideline:

1. Poll for the SCIFLR.BUSY flag to set.
2. Once the BUSY flag goes high, poll for the SCIFLR.RXRDY flag. Concurrently within this loop, also have a SW timeout, which times out and exits the loop if the RXRDY flag is not set within a user-defined time interval.

Method 2: Configure the CPU timer to interrupt if the RX interrupt is not triggered. This method does not use CPU bandwidth.

Use the following steps as a guideline:

1. Configure XINT to trigger an ISR when the LINRX goes from high to low (indicating LIN is busy).
2. Inside the XINT ISR, configure the CPU timer, which starts timing the frame completion.
3. If the frame is received correctly with the correct Sync field, it should trigger the LIN RX ISR, inside which you can turn off the timer so that you do not get a false timeout.
4. If the frame is not received correctly, it does not trigger the LIN RX ISR but triggers the CPU timer ISR (timeout occurred), which indicates an error in the Sync field.

4 Silicon Revision A Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

4.1 Silicon Revision A Usage Notes

Silicon revision-applicable usage notes have been found on a later silicon revision. For more details, see Silicon Revision B Usage Notes.

4.2 Silicon Revision A Advisories

Silicon revision-applicable advisories have been found on a later silicon revision. For more details, see Silicon Revision B Advisories.

Advisory ***Analog Subsystem: Software Configuration for Shared Reference Pins***

Revisions Affected 0, A

Details

Smaller pin-count packages of the F28004x device family have combined VREFHI pins. Software configuration bits are provided in the ANAREFPP register to disable all but one of the ganged references. This allows correct operation of internal reference mode in these circumstances. On production (TMS) devices, the Boot ROM will write these bits, and no further action will be required from the user. However, on some TMX devices, this write will not occur.

Workaround

For TMX devices, the user should do the following writes one time before trying to configure the references for internal reference mode:

- 100-pin PZ package: The value 0x0002 should be written to ANAREFPP.
- 64-pin PM package: The value 0x0003 should be written to ANAREFPP.
- 56-pin RSH package: The value 0x0003 should be written to ANAREFPP.

Advisory ***Analog Trim of Some TMX Devices***

Revisions Affected 0, A

Details

Some TMX samples may not have analog trims programmed. This could degrade the performance of the ADC, buffered DAC, internal oscillators, PGA, and internal voltage regulator. A value of all zeros in these trim registers will have the following impact.

TRIM	REGISTER	IMPACT OF TRIM VALUE EQUAL TO ZERO
ADC offset	AdcaRegs.ADCOFFTRIM AdcbRegs.ADCOFFTRIM AdccRegs.ADCOFFTRIM	Degraded performance of the ADC offset error specification.
ADC reference	AnalogSubsysRegs.ANAREFTRIMA AnalogSubsysRegs.ANAREFTRIMB AnalogSubsysRegs.ANAREFTRIMC	Degraded performance of the ADC for all specifications. No workaround available.
ADC linearity	AdcaRegs.ADCINLTRIM2-3 AdcbRegs.ADCINLTRIM2-3 AdccRegs.ADCINLTRIM2-3	Degraded INL and DNL specifications of the ADC. No workaround available.
Internal oscillator	AnalogSubsysRegs.INTOSC1TRIM AnalogSubsysRegs.INTOSC2TRIM	Degraded frequency accuracy and temperature drift of the internal oscillators.

Advisory (continued) ***Analog Trim of Some TMX Devices***

TRIM	REGISTER	IMPACT OF TRIM VALUE EQUAL TO ZERO
Buffered DAC offset	DacaRegs.DACTRIM DacbRegs.DACTRIM	Degraded offset error specification of the buffered DAC. No workaround available.
PGA gain and offset	PGAGAIN3TRIM PGAGAIN6TRIM PGAGAIN12TRIM PGAGAIN24TRIM	Degraded performance of the PGA gain and offset error specifications. No workaround available.

Workarounds

The following workarounds can be used for improved performance, though it still may not meet data sheet specifications.

Missing **ADC offset trim** can be generated by following the instructions in the ADC Zero Offset Calibration section of the *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*.

If the **internal oscillator trim** contains all zeros, the user can adjust the lowest 10 bits of the oscillator trim register between 1 (minimum) and 1023 (maximum) while observing the system clock on the XCLOCKOUT pin.

Advisory	***PGA: Output Filter Path is Not Supported***
Revisions Affected	0, A
Details	The PGA module includes an embedded series-resistor signal path (R_{FILTER}) for implementing a low-pass filter at the PGA_OF pin. This R_{FILTER} signal path should not be used or enabled on the affected revisions. The alternate functions shared with R_{FILTER} on the PGA_OF pin are not affected. For example, ADC input signals A2 and B6 are still available on PGA1_OF.
Workaround	None

Advisory	***ROM: Flash API Library and FPU32 Twiddle Factor RFFT Table Not Present***
Revisions Affected	0, A
Details	In the affected revisions, the Flash API library and the FPU32 twiddle factor for the 1024-pt RFFT table are not present in the ROM. For details on the ROM contents, see the Memory Maps section of the ROM Code and Peripheral Booting chapter in the *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*.
Workaround	None

Advisory	***REVID: Some TMX Revision A Devices Have an Incorrect REVID Value***
Revision Affected	A
Details	Some early TMX Revision A devices have an incorrect value in REVID (address 0x0005_D00C). The REVID incorrectly indicates the Revision 0 value (0x0000_0000) instead of the correct Revision A value (0x0000_0001). Software that uses REVID to distinguish between Revision 0 and Revision A will not function as intended. There are no other functional impacts due to this erratum. Applications that do not use REVID in software will work properly as any other Revision A device. Lot Trace Code affected: • 65AVVDW • 66ALSXW
Workaround	The device markings on the package are correct and can be used to identify the device revision.

Advisory ***GPIO: Parasitic Path to V_{SS} When Maximum V_{IH} is Exceeded in Input Mode***

Revision Affected

A

Details

If a voltage greater than maximum V_{IH} (V_{DDIO} + 0.3 V) is applied to the GPIO pins listed below, an internal parasitic path from the pin to V_{SS} may be turned on. This parasitic current can impact the functional operation of the pin. This is more likely to occur at high temperature. The parasitic path will be removed when the IO is driven below V_{IL}. The path will not reactivate until another overvoltage event occurs.

- GPIO16
- GPIO17
- GPIO24
- GPIO25
- GPIO26
- GPIO27
- GPIO35 (TDI)
- GPIO37 (TDO)
- GPIO40
- GPIO41
- GPIO42
- GPIO43

Pins configured for output-only mode (with no other drivers on the pin) will not see an overvoltage condition at the pin and are not affected by this advisory.

Pins configured in input or bidirectional mode can see an overvoltage condition in three primary ways:

1. The input is driven by a low-impedance driver that can generate a large overshoot at the input due to impedance mismatch without compensating termination.
2. The input sees large transients from external noise sources that rise above V_{DDIO} + 0.3 V at the pin.
3. The input is driven by a device powered by a different voltage regulator. When receiving voltages from another voltage domain, the system design should always keep voltages below maximum V_{IH}. However, due to the increased possibility of the voltage being temporarily greater than V_{DDIO} + 0.3 V due to a noise event or if there is improper supply sequencing, then this advisory will apply.

Workaround

If any of the above conditions apply for an input or bidirectional pin, insert a series resistor between the signal and the input pin. The series resistor should be placed close to the input pin.

If the overvoltage is due to overshoot (situations #1 or #2 above), a series resistor of 100 Ω or greater should be used.

If the overvoltage might be sustained (situation #3 above), a series resistor of 220 Ω or greater should be used.

5 Silicon Revision 0 Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

5.1 Silicon Revision 0 Usage Notes

Silicon revision-applicable usage notes have been found on a later silicon revision. For more details, see Silicon Revision B Usage Notes.

5.2 Silicon Revision 0 Advisories

Silicon revision-applicable advisories have been found on a later silicon revision. For more details, see Silicon Revision B Advisories.

Advisory	***GPIO: Pins may Drive High During Power Up***
Revision Affected	0
Details	During power up, the following pins will temporarily be in output mode and drive high. These pins will properly revert to input mode and operate per the pin description by the time XRSn releases (transitions to high). • GPIO0 • GPIO1 • GPIO2 • GPIO3 • GPIO4 • GPIO5 • GPIO11 • GPIO13 • GPIO29 • GPIO33
Workaround	None

Advisory ***GPIO: Signal Latch-up to V_{SS}***

Revision Affected

0

Details

The ESD structures on the pins listed below can be unintentionally turned on during functional operation, which will pull the pins to V_{SS}. There will be approximately 40 mA of additional current on the V_{DDIO} supply for each output pin in this condition.

- GPIO16
- GPIO17
- GPIO24
- GPIO25
- GPIO26
- GPIO27
- GPIO35 (TDI)
- GPIO37 (TDO)
- GPIO40
- GPIO41
- GPIO42
- GPIO43

The condition has not been observed below 70°C under normal operation. This condition can occur in input or output mode and with any of the mux functions. Designs with lightly loaded pins and fast switching signals are more likely to see the condition. Pins not bonded out in smaller pin-count packages can also enter the latch-up condition if they are toggled.

The latch-up condition can be ended by toggling the IO at a lower temperature.

Workarounds

Four workaround options are:

1. Avoid using these pins on the revision affected.
2. Avoid high-temperature operations on the revision affected.
3. If the pin is configured as an input or output:

   Place a capacitor of 300 pF or greater between each of these pins and ground, placed as closely as possible to the device. This will slow down the fast signal and avoid triggering the condition. Larger capacitors will be more effective at filtering the transient but must be balanced against the system-level timing requirements of these pins.

   For input pins, a smaller capacitor may be possible when used in combination with option 4.
4. If the pin is configured as an input:

   Connect a resistor in series with any other components on the board such that the total resistance of the driver plus the resistor is 1 kΩ or greater. The goal is to eliminate fast voltage transient seen at the pin. This will also limit the DC current if the ESD structure is activated due to noise.

Advisory	***ePWM: Event Latch (DCxEVTxLAT) of "DC Event-Based CBC Trip" May not Extend Trigger Pulse as Expected When Asynchronous Path is Selected***
Revision Affected	0
Details	DCxEVTxLAT may lose the captured trigger event for an asynchronous input upon deassertion. When an asynchronous trigger is deasserted, it is expected that the flop holds the value until there is a Clear event. Since the trigger is asynchronous with respect to the clock of the flop, there is a possibility that the flop may get cleared during deassertion. This would result in a loss of event latch function.
Workaround	None

6 Documentation Support

For more information regarding the F28004x devices, see the following documents:

- *TMS320F28004x Real-Time Microcontrollers* Data Sheet
- *TMS320F28004x Real-Time Microcontrollers Technical Reference Manual*

7 Trademarks

TMS320™ is a trademark of Texas Instruments.
C2000™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

8 Revision History

Changes from September 1, 2022 to February 22, 2024 (from Revision G (September 2022) to Revision H (February 2024)) **Page**

- *Devices Supported* section: Updated section....4
- Added Security: The primary layer of defense is securing the boundary of the chip, which begins with enabling JTAGLOCK and Zero-pin Boot to Flash feature Usage Note.... 9
- Added I2C: Target Transmitter Mode, Standard Mode SDA Timings Limitation advisory....24

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265