TMS320F28379S, TMS320F28377S, TMS320F28376S, TMS320F28375S, TMS320F28374S Delfino Microcontrollers

Silicon Errata

Literature Number: SPRZ422D
August 2014–Revised July 2016

Contents

List of Figures

List of Tables

TMS320F2837xS Delfino™ Microcontrollers

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320F2837xS microcontrollers (MCUs).

The updates are applicable to the following:

- 337-ball New Fine Pitch Ball Grid Array, ZWT Suffix
- 176-pin PowerPAD™ Thermally Enhanced Low-Profile Quad Flatpack, PTP Suffix
- 100-Pin PowerPAD Thermally Enhanced Thin Quad Flatpack, PZP Suffix

2 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] DSP devices and support tools. Each TMS320™ DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (for example, **TMS**320F28379S). 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).

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully qualified production device

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."

TMS 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 (TMX or TMP) 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, PTP) and temperature range (for example, T).

3 Device Markings

Figure 1 provides an example of the 2837xS device markings and defines each of the markings. The device revision can be determined by the symbols marked on the top of the package as shown in Figure 1. Some prototype devices may have markings different from those illustrated. Figure 2 shows an example of the device nomenclature.

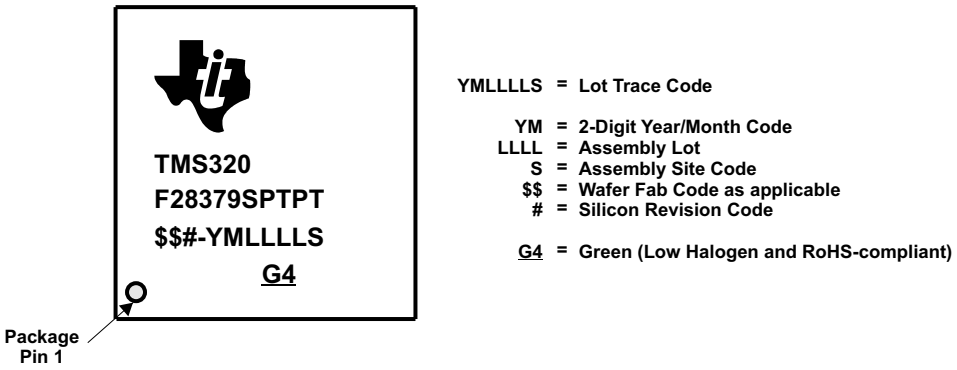

Figure 1. Example of Device Markings

Table 1. Determining Silicon Revision From Lot Trace Code

SILICON REVISION CODE	SILICON REVISION	REVID(1) Address: 0x5D00C	COMMENTS
B	B	0x0002	This silicon revision is available as TMX.
C	C	0x0003	This silicon revision is available as TMS.

(1) Silicon Revision ID

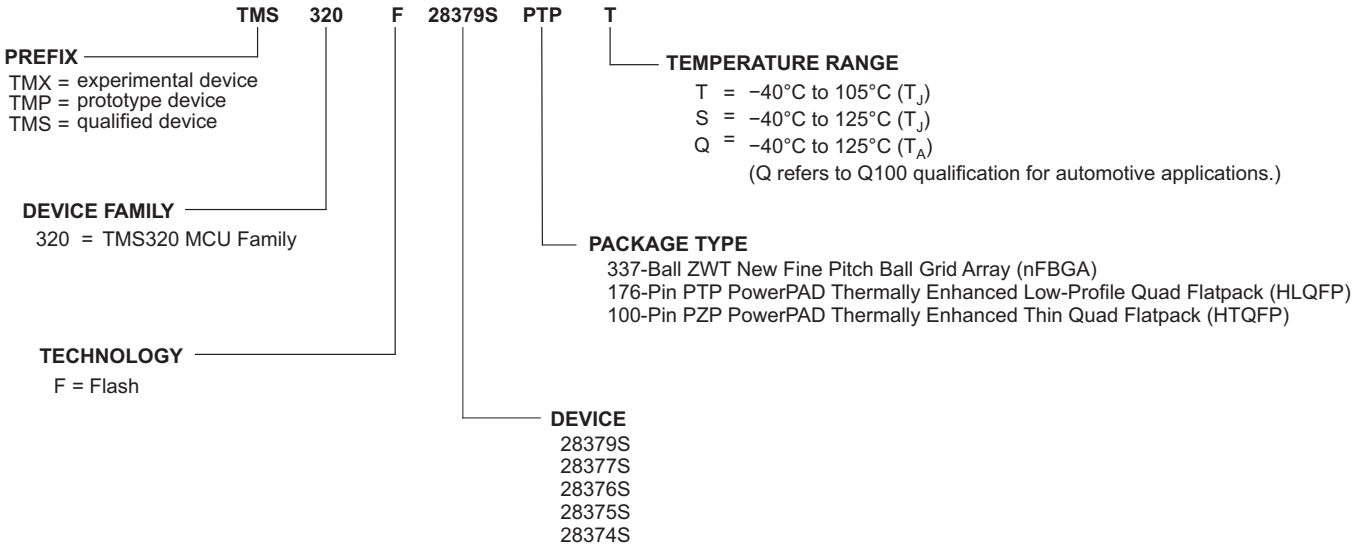

Figure 2. Example of Device Nomenclature

4 Usage Notes and Known Design Exceptions to Functional Specifications

4.1 Usage Notes

Usage notes highlight and describe particular situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness. These usage notes will be incorporated into future documentation updates for the device (such as the device-specific data sheet), and the behaviors they describe will not be altered in future silicon revisions.

Table 2 shows which silicon revision(s) are affected by each usage note.

Table 2. List of Usage Notes

TITLE	SILICON REVISION(S) AFFECTED	
	B	C
PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear	Yes	Yes

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

Revision(s) Affected: B, C

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
```

4.2 Known Design Exceptions to Functional Specifications

Table 3. Table of Contents for Advisories

Table 4 shows which silicon revision(s) are affected by each advisory.

Table 4. List of Advisories

TITLE	SILICON REVISION(S) AFFECTED	
	B	C
Analog Trim of Some TMX Devices	Yes	
ADC: ADC Post-Processing Block Limit Compare	Yes	Yes
ADC: ADC Offset Trim in Different Modes	Yes	Yes
ADC: Random Conversion Errors	Yes	
ADC: ADC PPB Event Trigger (ADCxEVT) to ePWM Digital Compare Submodule	Yes	
ADC: 12-Bit Switch Resistance	Yes	
ADC: 12-Bit Input Capacitance When Switching Channel Groups	Yes	
$\overline{\text{XRS}}$ may Toggle During Power Up	Yes	
USB: USB DMA Event Triggers are not Supported	Yes	Yes
VREG: VREG Will be Enabled During Power Up Irrespective of VREGENZ	Yes	
Flash: A Single-Bit ECC Error May Cause Endless Calls to Single-Bit-Error ISR	Yes	Yes
ePIE: Spurious VCU Interrupt (ePIE 12.6) Can Occur When First Enabled	Yes	
eQEP: Position Counter Incorrectly Reset on Direction Change During Index	Yes	Yes
PLL: May Not Lock On the First Lock Attempt	Yes	Yes
SDFM: Data Filter Output Does Not Saturate at Maximum Value With Sinc3 and OSR = 256	Yes	Yes
SDFM: Spurious Data Acknowledge Event When Data Filter is Configured and Enabled for the First Time	Yes	Yes
SDFM: Spurious Data Acknowledge Event When Data Filter is Synchronized Using PWM FILRES Signal	Yes	Yes
SDFM: Comparator Filter Module may Generate Spurious Over-Value and Under-Value Conditions	Yes	Yes
SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events	Yes	Yes
SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events	Yes	Yes
FPU: FPU-to-CPU Register Move Operation Preceded by Any FPU 2p Operation	Yes	Yes
FPU: LUF, LVF Flags are Invalid for the EINVF32 and EISQRTF32 Instructions	Yes	Yes
Memory: Prefetching Beyond Valid Memory	Yes	Yes
CMPSS: COMPxLATCH May Not Clear Properly Under Certain Conditions	Yes	Yes
CMPSS: Ramp Generator May Not Start Under Certain Conditions	Yes	Yes
Boot ROM: Device Will Hang During Boot if X1 Clock Source is not Present	Yes	

Advisory — *Analog Trim of Some TMX Devices*

Revision(s) Affected B

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

TRIM	REGISTER	IMPACT OF TRIM VALUE EQUAL TO ZERO
ADC reference	AnalogSubsysRegs.ANAREFTRIMA AnalogSubsysRegs.ANAREFTRIMB AnalogSubsysRegs.ANAREFTRIMC AnalogSubsysRegs.ANAREFTRIMD	Degraded performance of the ADC for all specifications.
ADC linearity	AdcaRegs.ADCINLTRIM1-6 AdcbRegs.ADCINLTRIM1-6 AdccRegs.ADCINLTRIM1-6 AdcdRegs.ADCINLTRIM1-6	Degraded INL and DNL specifications of the ADC in 16-bit mode. No workaround available.
ADC offset	AdcaRegs.ADCOFFTRIM AdcbRegs.ADCOFFTRIM AdccRegs.ADCOFFTRIM AdcdRegs.ADCOFFTRIM	Degraded performance of the ADC offset error specification.
Internal oscillator	AnalogSubsysRegs.INTOSC1TRIM AnalogSubsysRegs.INTOSC2TRIM	Degraded frequency accuracy and temperature drift of the internal oscillators.
Buffered DAC offset	DacaRegs.DACTRIM DacbRegs.DACTRIM DaccRegs.DACTRIM	Degraded offset error specification of the buffered DAC. No workaround available.

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

If the **ADC reference trim** registers contain all zeros, write the static reference trim value of 0x7BDD to the reference trim register for all ADCs.

Missing **ADC offset trim** can be generated by following the instructions in the “ADC Zero Offset Calibration” section of the *TMS320F2837xS Delfino 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 — *ADC: ADC Post-Processing Block Limit Compare*

Revision(s) Affected B, C

Details When using a non-zero offset reference in the ADC post-processing block (PPB), the resultant ADCPPBxRESULT can be signed. TRIPHI or TRIPLO limit compares do not function correctly with this result if it is signed.

Workaround(s) When using the TRIPHI or TRIPLO limit compares, leave the offset reference as zero. The offset reference (and zero compare) can be used as long as the limit compares are disabled.

If the limit compares, the offset reference, and the zero-crossing compare are to be used at the same time, then two PPBs can be used. Both PPBs should be configured to use the same SOC. One PPB can implement the TRIPHI and/or TRIPLO limit compares while the other can implement offset reference subtraction and zero-crossing detection.

Advisory — *ADC: ADC Offset Trim in Different Modes*

Revision(s) Affected B, C

Details A different offset trim is required when switching between 12-bit and 16-bit resolution and when switching between single-ended and differential signaling mode.

Workaround(s) Whenever setting the resolution or signal mode of the ADC, use the “AdcSetMode” function in controlSUITE™. This will ensure the correct trims are loaded into the offset trim register. Note that on start-up, trims will be loaded for 12-bit, single-ended operation.

Advisory — *ADC: Random Conversion Errors*

Revision(s) Affected B

Details The ADC may have errors at a rate as high as 1 in $10^{6.5}$ ADC conversions in 12-bit mode and as high as 1 in $10^{8.75}$ conversions in 16-bit mode. When a conversion error occurs, it will be a significant random jump in the digital output of the ADC without a corresponding change in the ADC input voltage, otherwise known as a “sparkle code”. The magnitude of this jump will typically be in the range of 20 LSBs to 200 LSBs; however, larger or smaller jumps may occur.

Workaround(s) For the revisions affected, the error rate will be lower than 1 error in $10^{14.5}$ ADC conversions for both 12-bit mode and 16-bit mode when all of the following configurations are used:

- The S+H duration is at least 320 ns
- ADCCLK is 40 MHz or less
- ADCCLK prescale is a whole number: /1.0, /2.0, /3.0, /4.0, /5.0, /6.0, /7.0, or /8.0
- The value of 0x7000 is written to memory locations 0x0000 743F, 0x0000 74BF, 0x0000 753F, and 0x0000 75BF (writing this value is only valid when the ADCCLK prescale is a whole number).

Advisory — *ADC: ADC PPB Event Trigger (ADCxEVT) to ePWM Digital Compare Submodule*

Revision(s) Affected B

Details The ADCxEVT trigger to the ePWM digital compare submodule may not be detected by the ePWM.

Workaround(s) The ADCxEVT can generate an ADCx_EVT interrupt to the PIE. The ISR can be used to perform the desired task in software.

Advisory — *ADC: 12-Bit Switch Resistance*

Revision(s) Affected B

Details The ADC input model should be used to select the sample-and-hold (S+H) duration for each ADC input. For the revisions affected, the 12-bit input model under-estimates the value of the sampling switch resistance (R_{on}). A R_{on} value of 2 kΩ should be used to select the S+H duration for these revisions.

Workaround(s) For the revisions affected, the S+H duration should be chosen to account for the additional switch resistance.

Advisory — *ADC: 12-Bit Input Capacitance When Switching Channel Groups*

Revision(s) Affected B

Details The ADC input model should be used to select the sample-and-hold (S+H) duration for each ADC input. For the revisions affected, if the currently converting channel is an even-numbered channel and the previously converted channel was an odd-numbered channel (or vice versa), then the 12-bit input model will not accurately predict ADC input performance. Under these conditions, an additional capacitance should be added to the model. This capacitance has a value of 11.5 pF and should be placed between the S+H switch and R_{on} as shown in Figure 3.

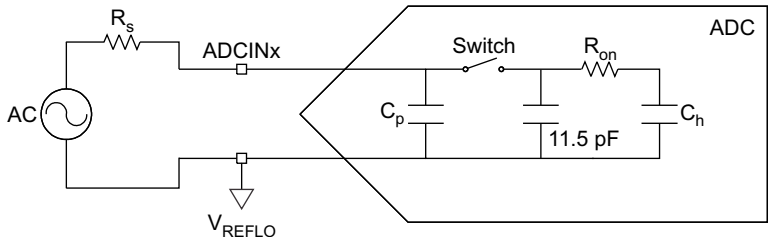

Figure 3. Single-Ended Input Model

Workaround(s) For the revisions affected, when subsequent conversions switch between channel groups, the S+H duration should be chosen to account for the additional capacitance.

Advisory — *$\overline{\text{XRS}}$ may Toggle During Power Up*

Revision(s) Affected B

Details During device power up, the $\overline{\text{XRS}}$ pin may toggle high prematurely. After the V_{DDIO} and V_{DD} supplies reach the recommended operation conditions, the $\overline{\text{XRS}}$ pin behavior will be per the pin description. This is only an issue with the external state of the $\overline{\text{XRS}}$ pin. Internally, the device will be held in reset by the POR logic until the supplies are within an acceptable range and $\overline{\text{XRS}}$ is high.

Workaround(s) Disregard $\overline{\text{XRS}}$ activity on the board prior to supplies reaching recommended operating conditions.

Advisory — *USB: USB DMA Event Triggers are not Supported*

Revision(s) Affected B, C

Details The USB module generates inadvertent extra DMA requests, causing the FIFO to overflow (on IN endpoints) or underflow (on OUT endpoints). This causes invalid IN DATA packets (larger than the maximum packet size) and duplicate receive data.

Workaround(s) None

Advisory — *VREG: VREG Will be Enabled During Power Up Irrespective of VREGENZ*

Revision(s) Affected B

Details During power up of the 3.3-V V_{DDIO}, the internal Voltage Regulator (VREG) will be active until the 1.2-V V_{DD} supply reaches approximately 0.7 V. After this time, the VREGENZ pin tied to V_{DDIO} will disable the internal VREG. This will not impact device operation.

Workaround(s) None

Advisory — *Flash: A Single-Bit ECC Error May Cause Endless Calls to Single-Bit-Error ISR*

Revision(s) Affected B, C

Details When a single-bit ECC error is detected, the CPU executes the single-bit-error interrupt service routine (ISR). When the ISR returns, the same instruction that caused the first error is fetched again. If the ECC error threshold (ERR_THRESHOLD.THRESHOLD) is 0, then the same error is detected and another ISR is executed. This continues in an endless loop. This sequence of events only occurs if the error is caused by a program fetch operation, not a data read.

Workaround(s) Set the error threshold bit-field (ERR_THRESHOLD.THRESHOLD) to a value greater than or equal to 1. Note that the default value of the threshold bit-field is 0.

Advisory — *ePIE: Spurious VCU Interrupt (ePIE 12.6) Can Occur When First Enabled*

Revision(s) Affected

B

Details

The VCU-II can power up in a state which incorrectly sets the VCU VSTATUS[DIVE] error bit and, subsequently PIEIFR12[INTx6], when the CPU is released from reset. When the VCU interrupt enable PIEIER12[INTx6] is enabled for the first time by the application, a spurious interrupt can occur due to the erroneous pending interrupt.

Workaround(s)

Before enabling VCU interrupt 12.6, execute the following instructions to avoid the spurious interrupt.

```
// Clear VCU divide by zero status
asm(" VCLRDIVE");
// Clear PIE interrupt for VCU
PieCtrlRegs.PIEIFR12.bit.INTx6 = 0;
```

Beginning with revision C silicon, the Boot ROM will perform the above workaround before branching to the application.

Advisory — *eQEP: Position Counter Incorrectly Reset on Direction Change During Index*

Revision(s) Affected

B, C

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(s)

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 — *PLL: May Not Lock on the First Lock Attempt*

Revision(s) Affected

B, C

Details

On a very small number of devices, the PLL may not start properly at device power up or wake up from Hibernate. The PLLSTS[LOCKS] bit is set, but the PLL does not produce a clock.

Once the PLL has properly started, the PLL can be disabled and re-enabled with no issues and will stay locked. However, the PLL lock problem could re-occur on a subsequent power-up or Hibernate cycle.

If the SYSPLL has not properly started and is selected as the CPU clock source, the CPU will stop executing instructions.

This advisory applies to both PLLs, with a different workaround for each.

Workaround(s)

SYSPLL Workaround:

Repeated lock attempts will reduce the likelihood of seeing the condition on the final attempt. TI recommends a minimum of five lock sequences in succession when the PLL is configured the first time after a power up. A lock sequence means disabling the PLL, starting the PLL locking, and waiting for the LOCKS bit to set. After the final sequence, the clock source is switched to use the PLL output as normal.

The Watchdog timer can be used to detect that the condition has occurred because it is not clocked by the PLL output. The Watchdog should be enabled before selecting the PLL as the clock source and configured to reset the device. If the PLL is not producing a clock, the Watchdog will reset the device and the user initialization software will therefore repeat the PLL initialization.

Many applications do not have a different initialization sequence for a Watchdog-initiated reset; for these applications, no further action is required. For applications that do use a different device initialization sequence when a Watchdog reset is detected, a flag can be used to identify the Watchdog reset as a PLL cause. The SYSDBGCTL[BIT_0] bit (which is bit 0 at 0x0005D12C) can be set active during the PLL lock sequence and used to distinguish a Watchdog PLL retry attempt versus a different Watchdog reset source.

See the ControlSUITE InitSysPll() function for an example implementation of this workaround.

AUXPLL Workaround:

CPU Timer 2 can be used to detect that the AUXPLL is active before it is used as a clock source for USB. If the AUXPLL is not active, repeat the lock attempt in the same manner as the SYSPLL workaround.

See the ControlSUITE InitAuxPll() function for an example implementation of this workaround.

NOTE: The USB Boot Mode does not implement the previous workarounds. Applications using USB Boot will need to implement any retry attempts at the system level.

Advisory — *SDFM: Data Filter Output Does Not Saturate at Maximum Value With Sinc3 and OSR = 256*

Revision(s) Affected B, C

Details If the differential input of the Sigma-Delta Filter Module (SDFM) is greater than or equal to +FSR (full-scale differential voltage input range), then the output of the SDFM clips with a stream of ones. When this stream of ones is fed to a data filter that is configured as a sinc3 filter with an OSR = 256, the output of the filter does not saturate at the maximum value (16777215 in 32-bit mode or 32767 in 16-bit mode); but, instead roll over to the minimum value (–16777216 in 32-bit mode or –32768 in 16-bit mode).

Workaround(s) Maintain the differential input of the SDFM in the specified linear input range as specified in the modulator data sheet.

Advisory — *SDFM: Spurious Data Acknowledge Event When Data Filter is Configured and Enabled for the First Time*

Revision(s) Affected B, C

Details When the SDFM data filter is configured and enabled for the first time, it is possible to get one spurious data acknowledge event (AFx) before the data filter settles to give correct digital data. Subsequent data acknowledge events (AFx)/DMA events occur correctly as per data filter configuration.

Workaround(s) Do the following:

1. Configure and enable the SDFM data filter.
2. Delay for at least latency of data filter + 5 SD-Cx clock cycles.
3. Enable SDFM data acknowledge interrupts/DMA events.

Advisory — *SDFM: Spurious Data Acknowledge Event When Data Filter is Synchronized Using PWM FILRES Signal*

Revision(s) Affected B, C

Details When the SDFM data filters are synchronized using the PWM FILRES signal, it is possible to get a spurious data acknowledge event (AFx) before the data filter settles to give correct digital data. Subsequent data acknowledge events (AFx) occur correctly as per data filter configuration before the next PWM FILRES signal.

Workaround(s) Do the following:

1. Choose any PWMx to work in the same time base as the PWM that generates the FILRES pulse.
2. PWMx should also interrupt the CPU/CLA at least 1.2 µs after the PWM FILRES pulse gets applied in order to clear the SDIFLG register that may be set because of the spurious data acknowledge event.
3. SDFM_CPUISR or SDFM_CLATask:
   (a) Collect the required number of samples, N, after the FILRES pulse.
   (b) If the number of samples is less than or equal to N, clear the SDIFLG register; otherwise, do not clear the SDIFLG register to prevent further SDFM interrupts.

Advisory — *SDFM: Comparator Filter Module may Generate Spurious Over-Value and Under-Value Conditions*

Revision(s) Affected

B, C

Details

When interrupts are enabled in the SDFM comparator module, it may trigger spurious over-value (SDIFLG.IEHx, x = 1 to 4) or under-value (SDIFLG.IELx, x = 1 to 4) conditions. These are depicted as IELx and IEHx interrupt outputs in the “Block Diagram of One Filter Module” figure in the *TMS320F2837xS Delfino Microcontrollers Technical Reference Manual*.

Workaround(s)

These erroneous interrupts can be eliminated by implementing the following workaround:

- Comparator OSR (COSR) value should be greater than or equal to 5.
- After changing COSR, wait for at least latency of comparator filter and 5 SD-Cx cycles before enabling comparator interrupts SDCPARMx.IEH and SDCPARMx.IEL.

Advisory — *SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events*

Revision(s) Affected

B, C

Details

When SDFM comparator settings—such as filter type, lower/upper threshold, or COSR settings—are changed while low-level/high-level events are enabled, a spurious comparator lower threshold (or) higher threshold event will be triggered.

Workaround(s)

When comparator settings need to be changed dynamically, follow the procedure below:

1. Disable the SDFM comparator interrupts.
2. Change comparator settings such as lower/upper threshold, filter type, or COSR.
3. Comparator OSR (COSR) value should be greater than or equal to 5.
4. Delay for at least a latency of comparator filter + 5 SD-Cx clock cycles.
5. Enable the SDFM comparator interrupts.

Advisory — *SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events*

Revision(s) Affected

B, C

Details

When SDFM data filter settings—such as filter type or DOSR settings—are changed while the data filter and its data acknowledge events are enabled, spurious data acknowledge events will be triggered.

Workaround(s)

When data filter settings need to be changed dynamically, follow the procedure below:

1. Disable the SDFM data filter.
2. Change data filter settings such as filter type or DOSR.
3. Delay for at least a latency of data filter + 5 SD-Cx clock cycles.
4. Enable the SDFM data filter.

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

Revision(s) Affected

B, C

Details

This advisory applies when a multi-cycle (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 E2 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, XT, 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
```

Workaround(s)

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
```

Advisory — *FPU: LUF, LVF Flags are Invalid for the EINVF32 and EISQRTF32 Instructions*

Revision(s) Affected B, C

Details This advisory applies to the EINVF32 and EISQRTF32 instructions. The expected results for these instructions are correct; however, the underflow (LUF) and overflow (LVF) flags are not. These flags are invalid and should not be used.

Workaround(s) Do not rely on the LUF and LVF flags to catch underflow/overflow conditions resulting from the EINVF32 and EISQRTF32 instructions. Instead, check the operands for the following conditions (in code) before using each instruction:

EINVF32	Divide by 0
EISQRTF32	Divide by 0, Divide by a negative input

Disregard the contents of the LUF and LVF flags by saving the flags to the stack before calling the instruction, and subsequently restoring the values of the flags once the instruction completes.

```
MOV32                 *SP++,STF    ; Save off current status flags
EISQRTF32/EINVF32                  ; Execute operation
NOP                                ; Wait for operations to complete
MOV32                 STF,*--SP    ; Restore previous status flags
```

If the PIE interrupts are tied to the LUF and LVF flags, disable the interrupts (at the PIE) before using either the EINVF32 or EISQRTF32 instruction. Check to see if the LUF and LVF flags are set; if they are, a variable can be set to indicate that a false LUF/LVF condition is detected. Clear the flags in the STF (FPU status flag) before re-enabling the interrupts.

Once the interrupts are re-enabled at the PIE, the interrupt may occur (if the LUF/LVF interrupt lines were asserted by either of the two instructions) and execution branches to the Interrupt Service Routine (ISR). Check the flag to determine if a false condition has occurred; if it has, disregard the interrupt.

Do not clear the PIE IFR bits (that latch the LUF and LVF flags) directly because an interrupt event on the same PIE group (PIE group 12) may inadvertently be missed.

Here is an example:

```
_flag_LVFLUF_set    .usect ".ebss",2,1,1
    ...
    MOV32   *SP++,STF                       ; Save off current status flags
    ; Load the PieCtrlRegs page to the DP
    MOVW    DP, #_PieCtrlRegs.PIEIER12.all
    ; Zero out PIEIER12.7/8, i.e. disable LUF/LVF interrupts
    AND     @_PieCtrlRegs.PIEIER12.all, #0xFF3F
    EISQRTF32/EINVF32                       ; Execute operation
    MOVL    XAR3, #_flag_LVFLUF_set         ; Wait for operation to complete
    MOV32   *+XAR3[0], STF                  ; save STF to _flag_LVFLUF_set
    AND     *+XAR3[0], #0x3                 ; mask everything but LUF/LVF
    ; Clear Latched overflow, underflow flag
    SETFLG  LUF=0, LVF=0
    ; Re-enable PIEIER12.7/8, i.e. re-enable the LUF/LVF interrupts
    OR      @_PieCtrlRegs.PIEIER12.all, #0x00C0
    MOV32   STF,*--SP                       ; Restore previous status flags
```

In the ISR,

```
__interrupt void fpu32_luf_lvf_isr (void)
{
 // Check the flag for whether the LUF, LVF flags set by
  // either EISRTF32 or EINVF32
  if((flag_LVFLUF_set & 0x3U) != 0U)
  {
    //Reset flag
    flag_LVFLUF_set = 0U;
    // Do Nothing
  }
  else
  {
    //If flag_LVFLUF_set was not set then this interrupt
    // is the legitimate result of an overflow/underflow
    // from an FPU operation (not EISQRTF32/EINVF32)
    ...
    // Handle Overflow/Underflow condition
    ...
    ...
    ...
  }
  // Ack the interrupt and exit
}
```

Advisory — *Memory: Prefetching Beyond Valid Memory*

Revision(s) Affected

B, C

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.

Workaround

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.

Table 5. Memories Impacted by Advisory

MEMORY TYPE	ADDRESSES IMPACTED	F28377S F28375S	F28376S F28374S
M1	0x0000 07F8–0x0000 07FF	Yes	Yes
GS11	0x0001 7FF8–0x0001 7FFF	No	Yes
GS15	0x0001 BFF8–0x0001 BFFF	Yes	N/A
Flash	0x000B FFF8–0x000B FFFF	No	Yes

Advisory — *CMPSS: COMPxLATCH May Not Clear Properly Under Certain Conditions*

Revision(s) Affected

B, C

Details

The CMPSS latched path is designed to retain a tripped state within a local latch (COMPxLATCH) until it is cleared by software (via COMPSTSCLR) or by PWMSYNC.

COMPxLATCH is set indirectly by the comparator output after the signal has been digitized and qualified by the Digital Filter. The maximum latency expected for the comparator output to reach COMPxLATCH may be expressed in CMPSS module clock cycles as:

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

When COMPxLATCH is cleared by software or by PWMSYNC, the latch itself is cleared as desired, but the data path prior to COMPxLATCH may not reflect the comparator output value for an additional LATENCY number of module clock cycles. If the Digital Filter output resolves to a logical 1 when COMPxLATCH is cleared, the latch will be set again on the following clock cycle.

Workaround(s)

Allow the Digital Filter output to resolve to logical 0 before clearing COMPxLATCH.

If COMPxLATCH is cleared by software, the output state of the Digital Filter can be confirmed through the COMPSTS register prior to clearing the latch. For instances where a large LATENCY value produces intolerable delays, the filter FIFO may be flushed by reinitializing the Digital Filter (via CTRIPxFILCTL).

If COMPxLATCH is cleared by PWMSYNC, the user application should be designed such that the comparator trip condition is cleared at least LATENCY cycles before PWMSYNC is generated.

Advisory — *CMPSS: Ramp Generator May Not Start Under Certain Conditions*

Revision(s) Affected

B, C

Details

The Ramp Generator is designed to produce a falling-ramp DAC reference that is synchronized with a PWMSYNC signal. Upon receiving a PMWSYNC signal, the Ramp Generator will start to decrement its DAC value. When COMPSTS[COMPHSTS] is asserted by a trip event, the Ramp Generator will stop decrementing its DAC value.

If COMPSTS[COMPHSTS] is asserted simultaneously with a PWMSYNC signal, the desired behavior is for the PWMSYNC signal to take priority such that the Ramp Generator starts to decrement in the new EPWM cycle. Instead of the desired behavior, the COMPSTS[COMPHSTS] trip condition will take priority over PWMSYNC such that the Ramp Generator stops decrementing for a full EPWM cycle until the next PWMSYNC signal is detected.

Workaround(s)

Avoid COMPSTS[COMPHSTS] trip conditions when PWMSYNC is generated. For example, peak current mode control applications can limit the PWM duty cycle to a maximum value that will avoid simultaneous COMPSTS[COMPHSTS] and PWMSYNC assertions.

Advisory — *Boot ROM: Device Will Hang During Boot if X1 Clock Source is not Present*

Revision(s) Affected

B

Details

The device boot code will attempt to configure the device using X1 as the clock source. When X1 is not present, the device boot will hang. This advisory applies to any system which is designed to use the INTOSC as the primary clock source with no clock on X1 during boot. This issue only affects some silicon revision B devices and it will be fixed in all future silicon revisions.

Workaround(s)

Apply external clock source to X1 on silicon revision B devices, even if using INTOSC as the application clock source.

5 Documentation Support

For device-specific data sheets and related documentation, visit the TI web site at: http://www.ti.com.

For more information regarding the TMS320F2837xS Delfino devices, see the following documents:

- *TMS320F2837xS Delfino™ Microcontrollers Data Manual*
- *TMS320F2837xS Delfino Microcontrollers Technical Reference Manual*

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from October 22, 2015 to July 21, 2016 (from C Revision (October 2015) to D Revision) **Page**

- Section 4.1 (Usage Notes): Removed "PLL: PLL Must be Locked Twice in Succession After PLLEN is Set" usage note. This is replaced with the new "PLL: May Not Lock on the First Lock Attempt" advisory. 6
- Section 4.2 (Known Design Exceptions to Functional Specifications): Added PLL: May Not Lock on the First Lock Attempt advisory. 14
- Section 4.2: Updated Details of SDFM: Dynamically Changing Threshold Settings, Filter Type, or COSR Settings Will Trigger Spurious Comparator Events advisory. 16
- Section 4.2: Updated Workaround(s) of SDFM: Dynamically Changing Data Filter Settings Will Trigger Spurious Data Acknowledge Events advisory..... 16
- Section 4.2: Added Memory: Prefetching Beyond Valid Memory advisory..... 20

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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