Errata

TMS320F281x DSPs Silicon Errata

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320F2810, TMS320F2811, and TMS320F2812 digital signal processors (DSPs).

The updates are applicable to:

- 128-pin Low-Profile Quad Flatpack (LQFP) [PBK suffix]
- 176-pin LQFP [PGF suffix]
- 179-ball MicroStar BGA™ [GHH suffix]
- 179-ball lead-free MicroStar BGA [ZHH suffix]
- 179-ball MicroStar BGA™ [GBB suffix]
- 179-ball lead-free MicroStar BGA™ [ZAY suffix]

2 Device and Development Tool Support 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** 320F2812). 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 (TMX/TMDX) through fully qualified production devices/ tools (TMS/TMDS).

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, GHH) and temperature range (for example, A).

3 Device Markings

Figure 3-1 provides an example of the TMS320F281x 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 3-1. Some prototype devices may have markings different from those illustrated. Figure 3-2 shows an example of device nomenclature.

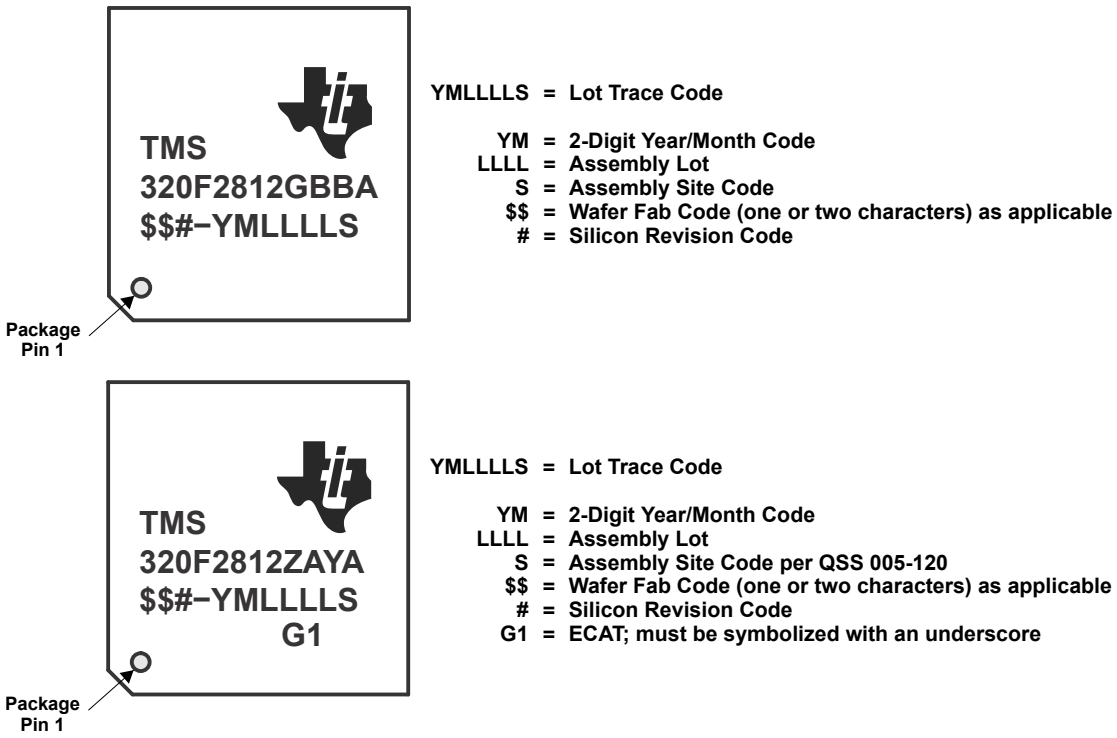


Figure 3-1. Example of Device Markings

Table 3-1. Determining Silicon Revision From Lot Trace Code

SILICON REVISION CODE	SILICON REVISION	REVISION ID Address: 0x0883	COMMENTS
Blank (no second letter in prefix)	Indicates Revision 0	0x0000	This silicon revision is available as TMX only.
A	Indicates Revision A	0x0001	This silicon revision is available as TMX only.
B	Indicates Revision B	0x0002	Internal
C	Indicates Revision C	0x0003	TMP/TMX/TMS
D	Indicates Revision D	0x0003	Internal
E	Indicates Revision E	0x0005	Production device (TMS)
F	Indicates Revision F	0x0006	Internal
G	Indicates Revision G	0x0007	Production device (TMS)

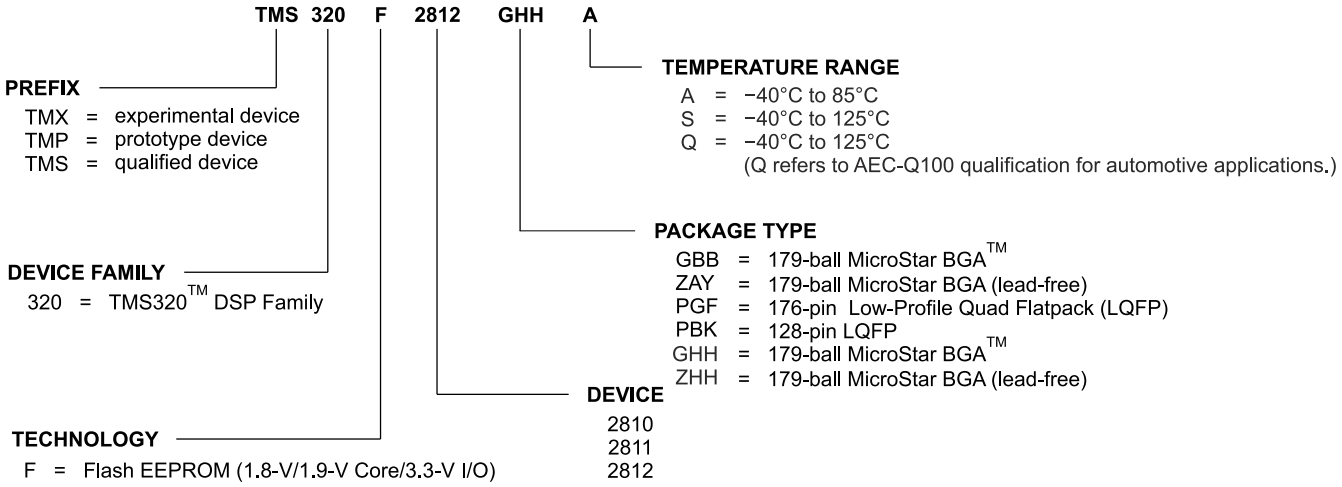


Figure 3-2. Example of Device Nomenclature

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 4-1 shows which silicon revision(s) are affected by each usage note.

Table 4-1. List of Usage Notes

TITLE	SILICON REVISION(S) AFFECTED(1)							
	0	A	B	C	D	E	F	G
PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear	Y	Y	Y	Y	Y	Y	Y	Y

(1) Y = Yes

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

Revision(s) Affected: 0, A, B, C, D, E, F and G

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
```

5 Known Design Exceptions to Functional Specifications

Table 5-1 shows which silicon revision(s) are affected by each advisory.

Table 5-1. List of Advisories

TITLE	SILICON REVISION(S) AFFECTED(1)							
	0	A	B	C	D	E	F	G
Memory: Prefetching Beyond Valid Memory	Y	Y	Y	Y	Y	Y	Y	Y
Memory: Program Reads From Flash/ROM Memory	Y	Y	Y	Y	Y	Y	Y	Y
Memory: Flash and OTP Prefetch Buffer Overflow	Y	Y	Y	Y	Y	Y	Y	Y
Memory: Set Device Emulation Register Bits for On-Chip RAM Performance	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
Memory: OTP Memory	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
XINTF: XBANK Does Not Properly Extend an Access	Y	Y	Y	Y	Y	Y	Y	Y
XINTF: XREADY Signal is not Sampled Properly When Using Asynchronous Sampling Mode	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
SCI: Incorrect Operation of SCI in Address Bit Mode	Y	Y	Y	Y	Y	Y	Y	Y
SCI: Bootloader Does Not Clear the ABD Bit After Auto-Baud Lock	Y	Y	Y	Y	Y	Y	Y	Y
SCI: Bootloader Does Not Clear the ABD Bit Before Auto-Baud Lock	Y	Y	Y	Y	Y	Y	Y	Y
eCAN: Abort Acknowledge Bit Not Set	Y	Y	Y	Y	Y	Y	Y	Y
eCAN: CPU Access to the eCAN Registers may Fail if it is in Conflict With an eCAN Access to the eCAN Registers	Y	Y	Y	Y	Y	Y	Y	Y
eCAN: Unexpected Cessation of Transmit Operation	Y	Y	Y	Y	Y	Y	Y	Y
WD: WDFLAG Bit Does Not Work as Intended	Y	Y	Y	Y	Y	Y	Y	Y
WD: A Low Output on GPIOF14 Can Disable the PLL and Watchdog if the Watchdog Fires a Reset	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
ADC: EOS BUF1/2 Bits in ADCST Corrupted at the End of Conversion of Sequencer 1/2 When INT MOD SEQ1/2 is Enabled	Y	Y	Y	Y	Y	Y	Y	Y
ADC: Reserved Bits in Autosequence Status Register (ADCASEQSR)	Y	Y	Y	Y	Y	Y	Y	Y
ADC: Sequencer Reset While Dual Sequencers Are Running	Y	Y	Y	Y	Y	Y	Y	Y
ADC: Result Register Update Delay	Y	Y	Y	Y	Y	Y	Y	Y
ADC: Device Has Higher Gain Error Than the Design Goal of 1% FSR on All of the B0−B7 Channels	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
ADC: Device Has Higher Offset Error Than the Design Goal (0.5 to 1%) on Some Channels	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
ADC: Device Has Higher Non-Linearity Than the Design Goal of 2 LSBs	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
McBSP: Receive FIFO Read Conflict	Y	Y	Y	Y	Y	Y	Y	Y
McBSP: Read Operations Decrement the McBSP FIFO	Y	Y	Y	Y	Y	Y	Y	Y
SPI: Slave-Mode Operation	Y	Y	Y	Y	Y	Y	Y	Y
Clocking: Logic-High Level for XCLKIN Pin	Y	Y	Y	Y	Y	Y	Y	Y
EV: QEP Circuit	Y	Y	Y	Y	Y	Y	Y	Y
QEP: QEP Inputs in GPIO Asynchronous Mode	Y	Y	Y	Y	Y	Y	Y	Y
DEVICE-ID: Register of the Silicon Same for Revision C and Revision D	N/A	N/A	N/A	N/A	Y	N/A	N/A	N/A
PLL: PLL x4 and x8 Multiplier Ratios	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A
Low-Power Modes − STANDBY Mode	Y	Y	N/A	N/A	N/A	N/A	N/A	N/A

(1) Y = Yes; N/A = Not Applicable

Advisory	*Memory: Prefetching Beyond Valid Memory*
Revision(s) Affected	0, A, B, C, D, E, F and G
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(s)	The prefetch queue is 8x16 words in depth. Therefore, code should not come within 8 words of the end of valid memory. This restriction applies to all memory regions and all memory types (Flash/ROM, OTP, SARAM, XINTF) on the device. Prefetching across the boundary between two valid memory blocks is fine. 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.

Advisory	*Memory: Program Reads From Flash/ROM Memory*
Revision(s) Affected	0, A, B, C, D, E, F and G
Details	When an interrupt occurs while program code is executing instructions from the address range 0x3F7FF0 through 0x3F7FF7, it is possible that subsequent data reads from the Flash/ROM will return all zeros.
Workaround(s)	Do not place program code within this address range. This range can be used for data variable storage.

Advisory

Memory: Flash and OTP Prefetch Buffer Overflow

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

This advisory applies to code executing from flash or OTP with the flash `prefetch` buffer enabled.

The flash prefetch buffer may overflow if a SBF or BF instruction is within eight 16-bit words preceding an operation using indirect or direct program-memory addressing. The window for which this can occur is shown below:

```
Address
Offset
0x0000    BF LSW (32-bit opcode)
0x0001    BF MSW or SBF (16-bit opcode)
-------------------------------------
0x0002    SBF/BF + 1 word     //
0x0003    SBF/BF + 2 words    //
0x0004    SBF/BF + 3 words    // If an instruction within this window
0x0005    SBF/BF + 4 words    // uses program-memory addressing, it
0x0006    SBF/BF + 5 words    // can cause the flash prefetch buffer to
0x0007    SBF/BF + 6 words    // overflow.
0x0008    SBF/BF + 7 words    //
0x0009    SBF/BF + 8 words    //
-------------------------------------
0x0010    SBF/BF + 9 words
```

Whether or not an overflow actually occurs depends on the instruction sequence, flash wait states and CPU pipeline stalls. If an overflow occurs, it will result in execution of invalid opcodes. Instructions that use program-memory addressing are MAC/XMAC, DMAC/XMACD, QMACL, IMACL, PREAD/XPREAD and PWRITE/XPWRITE.

Workaround(s)

1. Hand-coded assembly:

Use the SB/B instructions instead of SBF/BF for code targeted to execute from flash or OTP. The SB/B instructions are more efficient in wait-stated memory so a performance improvement may also be seen. In addition, the –flash_prefetch_warn compiler option can be used to issue a warning if the assembly code violates this erratum.

2. Compiler-generated assembly:

Compiler versions prior to V15.12.0.LTS:

Use the compiler switch -me to force the compiler to generate SB/B instructions instead of SBF/BF instructions. In heavily wait stated memory the SB/B instructions are more efficient than SBF/BF. In SARAM the SBF/BF instructions are more efficient. Therefore, this switch should be applied as follows:

- Use the compiler switch -me on source code that runs from flash or OTP.
- Do not use the compiler switch -me on source code that runs from SARAM.
- Use -me If a file contains functions that runs from flash as well as functions that run from SARAM.

The -me switch is available in C28x compiler v4.1.4 and later.

Compiler V15.12.0.LTS and later:

Indicate which functions will run from SARAM using the --ramfunc=on option or the *_attribute_*((ramfunc)). The compiler will only generate SBF/BF instructions within these functions.

The –me switch is deprecated and no longer has any effect.

Advisory ***Memory: Set Device Emulation Register Bits for On-Chip RAM Performance***

Revision(s) Affected

0 and A

Details

To get the best performance of on-chip RAM blocks M0/M1/L0/L1/H0, the internal control register bits have to be enabled. The bits are in the Device Emulation Registers.

Workaround(s)

All device initialization code should include the following register updates. These are EALLOW-protected registers.

Register Address	Value
0x950	0x0300
0x951	0x0300
0x952	0x0300
0x953	0x0300
0x954	0x0300

Code Example:

```
EALLOW
MOVL      XAR1,#0x0950
MOVL      XAR2,#0x0300
MOV       *XAR1++,AR2
MOV       *XAR1++,AR2
MOV       *XAR1++,AR2
MOV       *XAR1++,AR2
MOV       *XAR1++,AR2
EDIS
```

The Code Composer GEL init files will initialize these for emulation and debug environment. From the next silicon revision onward, this initialization is automatically done upon reset.

Advisory ***Memory: OTP Memory***

Revision(s) Affected

0 and A

Details

The 1K-word OTP memory is not available.

Workaround(s)

This is fixed in the next revision of the silicon.

Advisory ***XINTF: XBANK Does Not Properly Extend an Access***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

When XTIMCLK is not equal to SYSCLKOUT, the XBANK logic may not properly delay a pending access. This occurs for some combinations of XINTF zone wait states and XBANK delay cycles. There are two cases when this occurs.

Case 1: When XTIMCLK = 1/2 SYSCLKOUT and XCLKOUT = XTIMCLK

A pending access may not be delayed by the XBANK logic if either:

- WLEAD + WACTIVE + WTRAIL <= XBANK[BCYC] or
- RLEAD + RACTIVE + RTRAIL <= XBANK[BCYC]

Where WLEAD, WACTIVE, WTRAIL, RLEAD, RACTIVE, RTRAIL are defined as shown in Table 5-2.

Table 5-2. Pending Access Relationships

	X2TIMING = 0	X2TIMING = 1
WLEAD	XTIMING x [XWRLEAD]	XTIMING x [XWRLEAD] x 2
WACTIVE	XTIMING x [XWRACTIVE] + 1	XTIMING x [XWRACTIVE] x 2 + 1
WTRAIL	XTIMING x [XWRTRAIL]	XTIMING x [XWRTRAIL] x 2
RLEAD	XTIMING x [XRDLEAD]	XTIMING x [XRDLEAD] x 2
RACTIVE	XTIMING x [XRDACTIVE] + 1	XTIMING x [XRDACTIVE] x 2 + 1
RTRAIL	XTIMING x [XRDTRAIL]	XTIMING x [XRDTRAIL] x 2

In Table 5-2, XTIMINGx refers to the XTIMING register for Zone x. When XBANK delay cycles are added between two accesses, Zone x refers to the first zone in the sequence. For example: if XBANK[BANK] = 7, then delay cycles will be added to any access into or out of Zone 7. This means:

- Access to Zone 0 followed by Zone 7: the timing of Zone 0 is critical.
- Access to Zone 1 followed by Zone 7: the timing of Zone 1 is critical.
- Access to Zone 7 followed by Zone 0: the timing of Zone 7 is critical.

Thus, the timing of any zone involved in bank switching must be considered.

Case 2) When XTIMCLK = 1/2 SYSCLKOUT and XCLKOUT = 1/2 XTIMCLK:

A pending access may not be delayed properly by the XBANK logic if XBANK[BCYC] = 4 or XBANK[BCYC] = 6.

Workaround(s)

Case 1) If XTIMCLK = 1/2 SYSCLKOUT and XCLKOUT = XTIMCLK, then select:

- XBANK[BCYC] <= WLEAD + WACTIVE + WTRAIL and
- XBANK[BCYC] <= RLEAD + RACTIVE + RTRAIL

When XBANK delay cycles are added between two accesses, the timing restriction applies to the first zone accessed as described earlier. The timing of any zone involved in bank switching must be considered.

Table 5-3 shows examples of valid XBANK[BCYC] selections. This list is not exhaustive.

Advisory (continued)

XINTF: XBANK Does Not Properly Extend an Access

Table 5-3. Examples of Valid XBANK Selections

XWRLEAD XRDLEAD	WRACTIVE XRDACTIVE	XWRTRAIL XRDTRAIL	X2TIMING	WLEAD + WACTIVE + WTRAIL	Choose XBANK[BCYC]
1	2	1	0	5	< 5
1	3	1	0	6	< 6
2	3	1	0	7	< 7
1	0	1	1	3	< 3
1	1	0	1	5	< 5
1	1	1	1	7	< 7

Case 2: If XTIMCLK = 1/2 SYSCLKOUT and XCLKOUT = 1/2 XTIMCLK, then select:

- XBANK[BCYC] != 4 and
- XBANK[BCYC] != 6

Advisory

XINTF: XREADY Signal is not Sampled Properly When Using Asynchronous Sampling Mode

Revision(s) Affected

0 and A

Details

In case of asynchronous ready mode, if the XREADY signal is high within the Lead period, then access will complete in the number of cycles programmed in LEAD + ACTIVE + TRAIL counters even if XREADY goes low before the start of the ACTIVE period. In this case, XREADY is not being used properly to extend the access.

Workaround(s)

Try one of the following possible workarounds:

- Ensure that the XREADY signal is not low at the start of an access when using asynchronous sampling mode. If the XINTF sees the XREADY signal low from the start of an access, then the ACTIVE period will be extended as desired.
- Use the XTIMING register wait-state values to extend the access such that timings are met without using XREADY.
- Use the synchronous XREADY sampling mode. This problem is not observed in synchronous mode.

This issue is fixed in the next revision of the silicon.

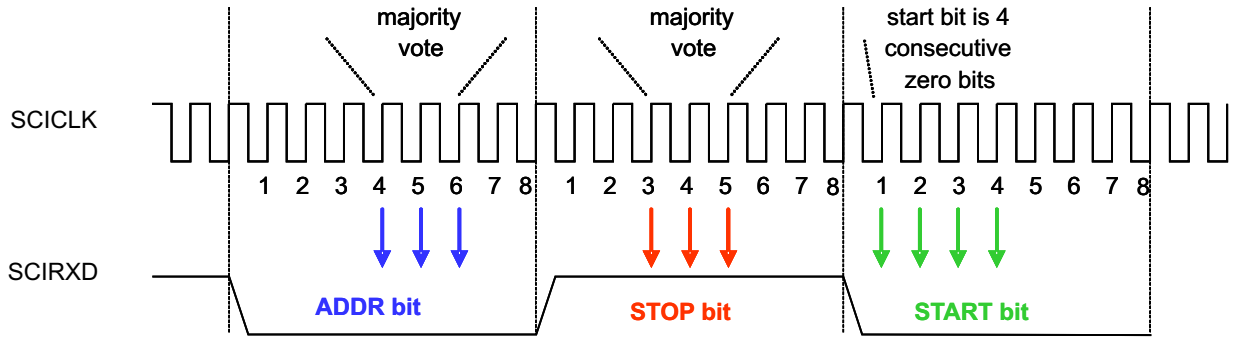
Advisory ***SCI: Incorrect Operation of SCI in Address Bit Mode***

Revision(s) Affected 0, A, B, C, D, E, F and G

Details

The SCI does not look for the STOP bit after the ADDR bit. Instead, the SCI starts looking for the start bit beginning on sub-sample 6 of the ADDR bit. Slow rise time from the ADDR bit to the STOP bit can cause a false START bit to occur since the 4th sub-sample for the start bit may be sensed low.

Expected Operation:



Erroneous Operation:

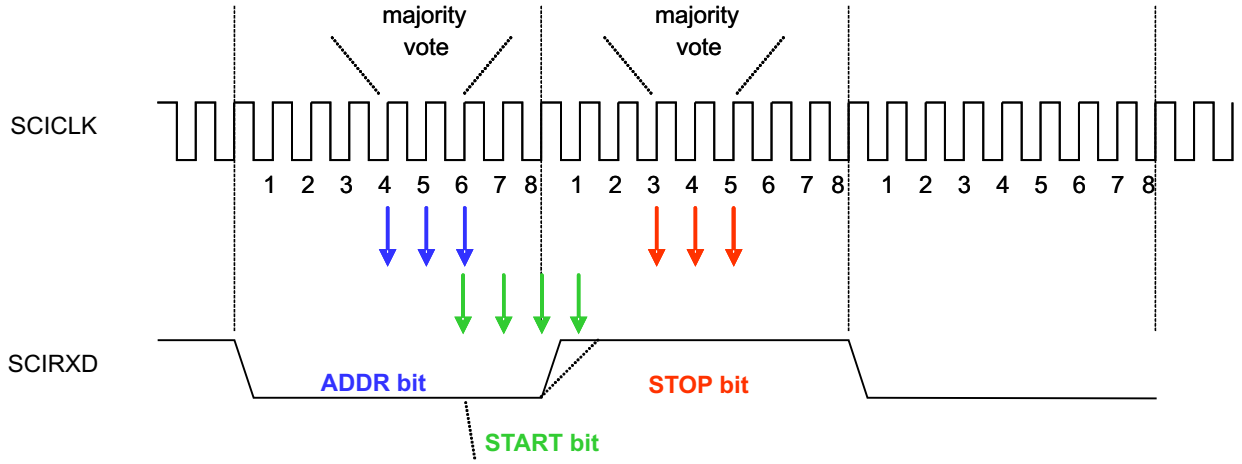


Figure 5-1. Difference Between Expected and Erroneous Operation of START Bit

Workaround(s)

Program the baud rate of the SCI to be slightly slower than the actual. This will cause the 4th sub-sample of the false START bit to be delayed in time, and therefore occur more towards the middle of the STOP bit (away from the signal transition region). The amount of baud-slowing needed depends on the rise time of the signal in the system. Alternatively, the IDLE mode of the SCI module may be used, if applicable.

Advisory ***SCI: Bootloader Does Not Clear the ABD Bit After Auto-Baud Lock***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

The SCI ROM bootloader code does not clear the auto-baud detect (ABD) bit in the SCIFFCT register after the auto-baud process completes. If the SCI-A port is used after the bootloader is executed, transmit interrupts (SCITXINTA) will not be able to occur, nor will the auto-baud lock feature of SCI-A work correctly.

Workaround(s)

If the SCI bootloader has been executed, the user's application code should clear the ABD bit by writing a 1 to ABD CLR (bit 14) in the SCIFFCT register before enabling the SCITXINTA interrupt, and before using the auto-baud feature.

Advisory ***SCI: Bootloader Does Not Clear the ABD Bit Before Auto-Baud Lock***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

The SCI ROM bootloader code does not correctly clear the Auto-Baud Detect (ABD) bit in the SCIFFCT register before the auto-baud process begins. The bootloader code fragment is shown below:

```
// Prepare for autobaud detection
// Set the CDC bit to enable autobaud detection
// and clear the ABD bit
SCIARegs.SCIFFCT.all = 0x2000;
```

The comments incorrectly state that the ABD bit is cleared. The ABD bit is cleared by writing a 1 to the ABD_CLR bit (bit 14) of the SCIFFCT register. This situation does not hinder operation from power up or reset because the ABD bit is cleared by default after reset. If, however, the bootloader is invoked a second time from software, then the ABD bit will not be cleared and autobaud lock will not occur properly.

Workaround(s)

If the bootloader is going to be re-invoked by software, the user's code must first clear the ABD bit before calling the bootloader. To do this, write a 1 to the ABD CLR bit (bit 14) in the SCIFFCT register.

Advisory ***eCAN: Abort Acknowledge Bit Not Set***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

After setting a transmission request reset (TRR) register bit to abort a message, there are some rare instances where the TRRn and TRSn bits will clear without setting the abort acknowledge (AAn) bit. The transmission itself is correctly aborted, but no interrupt is asserted and there is no indication of a pending operation.

In order for this rare condition to occur, all of the following conditions must happen:

1. The previous message was not successful, either because of lost arbitration or because no node on the bus was able to acknowledge it or because an error frame resulted from the transmission. The previous message need not be from the same mailbox in which a transmit abort is currently being attempted.
2. The TRRn bit of the mailbox should be set in a CPU cycle immediately following the cycle in which the TRSn bit was set. The TRSn bit remaining set due to incompletion of transmission satisfies this condition as well—that is, the TRSn bit could have been set in the past, but the transmission remains incomplete.
3. The TRRn bit must be set in the exact SYSCLKOUT cycle where the CAN module is in idle state for one cycle. The CAN module is said to be in idle state when it is not in the process of receiving/transmitting data.

If these conditions occur, then the TRRn and TRSn bits for the mailbox will clear t_{clr} SYSCLKOUT cycles after the TRR bit is set where:

$$t_{clr} = ((\text{mailbox\_number})*2)+3 \text{ SYSCLKOUT cycles}$$

The TAn and AAn bits will not be set if this condition occurs. Normally, either the TA or AA bit sets after the TRR bit goes to zero.

Workaround(s)

When this problem occurs, the TRRn and TRSn bits will clear within t_{clr} SYSCLKOUT cycles. To check for this condition, first disable the interrupts. Check the TRRn bit t_{clr} SYSCLKOUT cycles after setting the TRRn bit to make sure it is still set. A set TRRn bit indicates that the problem did not occur.

If the TRRn bit is cleared, it could be because of the normal end of a message and the corresponding TAn or AAn bit is set. Check both the TAn and AAn bits. If either of the bits is set, then the problem did not occur. If they are both zero, then the problem did occur. Handle the condition like the interrupt service routine would except that the AAn bit does not need clearing now.

If the TAn or AAn bit is set, then the normal interrupt routine will happen when the interrupt is re-enabled.

Advisory ***eCAN: CPU Access to the eCAN Registers may Fail if it is in Conflict With an eCAN Access to the eCAN Registers***

Revision(s) Affected 0, A, B, C, D, E, F and G

Details

If contention exists between the CPU and the eCAN controller for access to certain eCAN register areas, a CPU read may erroneously read all zeros (0x00000000), and a CPU write may erroneously fail to execute. Specifically:

- **Case 1:** If the CPU reads the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers) at the same time that the eCAN controller is accessing (reading or writing) the LAM/MOTO/MOTS register area, the CPU may erroneously read all zeros (0x00000000).
- **Case 2:** If the CPU writes to the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH register) at the same time that the eCAN controller is accessing (reading or writing) the LAM/MOTO/MOTS register area, the CPU write may fail to execute.
- **Case 3:** If the CPU reads the LAM/MOTO/MOTS register area at the same time that the eCAN controller is accessing (reading or writing) the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers), the CPU may erroneously read all zeros (0x00000000).
- **Case 4:** If the CPU writes to the LAM/MOTO/MOTS register area at the same time that the eCAN controller is accessing (reading or writing) the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers), the CPU write may fail to execute.

Workaround(s)

Workarounds for each of the four cases are as follows:

- **Case 1:** For all CPU reads from the eCAN mailbox RAM area, check to see if the read returns all zeros. If so, the CPU should perform a second read. If the second read returns zero as well, then the data is correctly zero. If the second read returns a non-zero value, then the second data is the correct value. Note that interrupts must be disabled during the consecutive CPU reads. See NOTE 5.
- **Case 2:** For all CPU writes to the eCAN mailbox RAM area, the CPU should write the data twice. Note that interrupts must be disabled during the consecutive CPU writes. See NOTE 5.
- **Case 3:** For all CPU reads from the LAM/MOTO/MOTS register area, check to see if the read returns all zeros. If so, the CPU should perform a second read. If the second read returns zero as well, then the data is correctly zero. If the second read returns a non-zero value, then the second data is the correct value. Note that interrupts must be disabled during the consecutive CPU reads. See NOTE 5.
- **Case 4:** For all CPU writes to the LAM/MOTO/MOTS register area, the CPU should write the data twice with a minimum of 4 CPU cycles in between the writes. Note that interrupts must be disabled during the consecutive CPU writes. See NOTE 5.

Advisory (continued)

eCAN: CPU Access to the eCAN Registers may Fail if it is in Conflict With an eCAN Access to the eCAN Registers

Note

1. An example of the eCAN controller reading the LAM/MOTO/MOTS register area is a read of the LAMn register to check if a received message passes the acceptance mask filtering criterion. This happens during reception of a frame.
2. An example of the eCAN controller writing to the LAM/MOTO/MOTS register area is a write to the MOTSn register to update the timestamp upon successful transmission of a frame.
3. An example for the eCAN controller attempting to read the mailbox RAM area (MSGID, MSGCTRL, MDL, and MDH registers) is right before transmission.
4. An example for the eCAN controller attempting to write to the mailbox RAM area (MSGID, MSGCTRL, MDL, and MDH registers) is right after reception.
5. A C callable assembly implementation of the workaround can be downloaded from the TI Website.

Advisory

eCAN: Unexpected Cessation of Transmit Operation

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

In rare instances, the cessation of message transmission from the eCAN module has been observed (while the receive operation continues normally). This anomalous state may occur without any error frames on the bus.

Workaround(s)

The Time-out feature (MOTO) of the eCAN module may be employed to detect this condition. When this occurs, set and clear the CCR bit (using the CCE bit for verification) to remove the anomalous condition.

Advisory ***WD: WDFLAG Bit Does Not Work as Intended***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

The WDFLAG bit in F281x devices cannot be used to reliably distinguish a watchdog-initiated reset from a power-on (or warm) reset. This is because the device expects the $\overline{\text{XRS}}$ pin to be pulled high (by the external reset circuit) within 4 SYSCLKOUT cycles (8 OSCLK cycles at power up) after the end of the watchdog-initiated reset pulse, which is 512 OSCCLK cycles. Most of the external $\overline{\text{XRS}}$ circuits cannot provide the fast rise time requirement (due to the capacitance); therefore, the WDFLAG bit should not be used in applications.

Workaround(s)

None. This bit should not be used.

Advisory ***WD: A Low Output on GPIOF14 Can Disable the PLL and Watchdog if the Watchdog Fires a Reset***

Revision(s) Affected

0 and A

Details

If, during program execution, the XF_ $\overline{\text{XPLLDIS}}$/GPIOF14 signal is changed to either of the following:

- A general-purpose output and driven low
- The XF functionality and driven low

and a watchdog reset occurs, then the low output state of the XF_ $\overline{\text{XPLLDIS}}$/GPIOF14 pin will be latched into the $\overline{\text{XPLLDIS}}$ signal. The result of this is that the PLL and the reset function of the watchdog will be disabled. The watchdog itself is not disabled.

Workaround(s)

One of the following workarounds can be used:

- Do not toggle XF/GPIOF14 in user code. Instead, use another GPIO signal for status.
- Set the watchdog to fire an interrupt instead of reset.

This is fixed in the next revision of the silicon.

Advisory ***ADC: EOS BUF1/2 Bits in ADCST Corrupted at the End of Conversion of Sequencer 1/2 When INT MOD SEQ1/2 is Enabled***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

Setting the INT MOD SEQx bit in ADCTRL2 as per the user guide should result in the ADC wrapper generating INT SEQx at the end of every other conversion rather than at the end of every conversion. Also, EOS BUFx will be set at the end of every conversion to track the status of the SEQx in use. However, a conversion on SEQ1 will cause EOS_BUF2 to be set if INT_MOD_SEQ2 is enabled for that sequencer, even if INT_MOD_SEQ1 is not enabled.

For example, if INT_MOD_SEQ2 is set, a conversion on SEQ1 will cause the EOS_BUF2 bit to be set incorrectly. This will cause INT SEQ2 to be set incorrectly after the next SEQ2 completion. If EOS_BUF2 is already set (from previous SEQ conversion), a conversion on SEQ1 will cause EOS_BUF2 to be cleared causing the interrupt to be missed. The above relationship is also true for SEQ2 affecting SEQ1. In all cases, the sequencers do work correctly, with the exception that EOS BUFx gets corrupted.

Workaround(s)

Do not use the INT_MOD_SEQx feature if both SEQ1 and SEQ2 will be used before two completions of sequence have completed on the INT_MOD_SEQ selected sequencer.

Advisory ***ADC: Reserved Bits in Autosequence Status Register (ADCASEQSR)***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

SEQ2 STATE2–0 and SEQ1 STATE3–0 bit fields (bits 6 through 0) are the pointers of SEQ2 and SEQ1, respectively. These bits are reserved for TI testing and should not be used in customer applications.

Workaround(s)

None

Advisory ***ADC: Sequencer Reset While Dual Sequencers Are Running***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

In the TMS320F2812/TMS320F2811/TMS320F2810 on-chip ADC, there are two sequencers for performing ADC conversions; SEQ1 and SEQ2. If one of the sequencers is reset while the other sequencer is running, it will result in the running sequencer never completing its current sequence. The sequencer busy bit (bit 3/bit 2 in ADCST register) for the sequencer will remain active and an “End-of-sequence (EOS)” interrupt for the running sequencer will never be generated. For example, if SEQ1 is reset while SEQ2 is performing a sequence, then SEQ2 will never complete.

Workaround(s)

If dual sequencers are enabled, then the software handling the ADC module should make sure that SEQ1 BSY/ SEQ2 BSY bits are not set before performing a reset of either sequencer.

Advisory ***ADC: Result Register Update Delay***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

The ADC result status flags INT_SEQ1 and INT_SEQ2 bit fields (bits 0 and 1, respectively) in the ADC_ST_FLG register indicate the availability of new ADC results after conversions and initiation of the ADC interrupts.

The update of the ADC result register requires one extra ADC cycle to complete after the status flags INT_SEQ1 and INT_SEQ2 bit(s) are set. The result of reading the result register prior to this extra cycle will result in old data being read (reset value/previous conversion result).

If auto-sequencers are enabled with a non-zero value in the MAXCONV register, the last result register update takes an additional ADC cycle from the time the INT_SEQ1 or INT_SEQ2 flag is set.

Workaround(s)

Delay the read of the ADC result register(s) by at least one ADC clock period. This delay can be implemented by using software delay loops.

If the ADC result register(s) are read using the ADC interrupt, rather than polling, the wait period introduced by the ISR (interrupt service routine) could minimize the delay needed in software. This ISR branching delay is generally greater than 8 SYSCLKOUT cycles.

The ratio of the ADC clock (ADCCLK) to the CPU clock (SYSCLKOUT) determines the size of the software delay. For example, if ADCCLK = 10 MHz, the software delay should be at least 100 ns.

Timing example to estimate the software delay:

1. Get the HSPCLK prescaler value − HISPCP
2. Get the ADCCLK prescaler value − ADCCLKPS
3. Get the CPS (ADCCTRL1[7]) value − CPS
4. Software wait-period in CPU cycles (SYSCLKOUT) before the ADC result register read is defined as:

```
Software wait = (HISPCP *2) * (ADCCLKPS * 2) * (CPS +1) cycles
If HISPCP or ADCCLKPS is 0, then the respective terms should be (HISPCP +1)
or (ADCCLKPS+1)
```

Advisory ***ADC: Device Has Higher Gain Error Than the Design Goal of 1% FSR on All of the B0−B7 Channels***

Revision(s) Affected 0 and A

Details

The device has a higher gain error than the design goal of 1% FSR on all of the B0−B7 channels. The gain error varies across channels A0−A7 and B0−B7.

Based on the current data obtained on B group channels, all B group channels show a uniform gain error as high as 2 to 3%.

Workaround(s)

The channel-to-channel gain error data across channels are listed in Table 5-4. This should help in calibrating in software or hardware. This was fixed in Revision B silicon.

Table 5-4. Channel-to-Channel Offset Error Data Across Channels (176-Pin PGF)

ADC CHANNELS	A0	A1	A2	A3	A4	A5	A6	A7	B0	B1	B2	B3	B4	B5	B6	B7
Gain Error in %	0.20	0.18	0.52	0.53	0.53	0.55	0.54	0.54	2.92	2.92	2.92	2.93	2.93	2.93	2.93	2.97
Offset in LSB Counts	14.80	15.64	4.86	9.82	5.82	– 14.57	– 13.98	– 31.78	20.94	21.98	22.48	23.39	22.39	23.14	23.64	24.91

Note

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory ***ADC: Device Has Higher Offset Error Than the Design Goal (0.5 to 1%) on Some Channels***

Revision(s) Affected 0 and A

Details

Based on the current data obtained on all channels, some channels show an offset error as high as 1%.

Workaround(s)

The channel-to-channel offset error data across channels are listed in Table 5-4. This should help in calibrating in software or hardware. This was fixed in Revision B silicon.

Note

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory	***ADC: Device Has Higher Non-Linearity Than the Design Goal of 2 LSBs***
Revision(s) Affected	0 and A
Details	Based on the current data obtained on all channels, some channels show non-linearity as high as 12 LSBs in the mid-scale range. That is, the mid-range conversions will be off by about 12 LSB counts.
Workaround(s)	This issue is corrected in the next revision of the silicon. The following option could be used to correct for INL errors only for the TMX Revision A silicon. The INL issue is across all channels. Use the ADC results only for 9-bit data accuracy on this revision of the silicon and ignore the rest of the bits. This will mitigate the INL effect in the application provided the algorithm can tolerate 9-bit accuracy.

Note

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory ***McBSP: Receive FIFO Read Conflict***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

The McBSP peripheral operates with or without FIFOs. The receive FIFO has interrupt generation logic that initiates interrupts based on the 5-bit FIFO status bits (12−8) and interrupt level bits (4−0) in the MFFRX register.

If the CPU reads the receive FIFO while the McBSP module writes new data into the FIFO, there is a potential conflict. The CPU read will not be stalled and read data will not be valid. The FIFO write gets the priority. The receive FIFO will be updated after every word is received in DRR2/1 registers. The DRR2/1 register update time will primarily depend on the word size and CLKR rate. For example, for 8-bit word, it should be typically 8 times the CLKR cycle time. This conflict will be more pronounced if data transferred on the receive channel is back-to-back with no delays between words.

Workaround(s)

The receive FIFOs should be read based on receive interrupts and within the next word receive time. To avoid the read conflict, additional checks could be used before initiating receive FIFO read. In most McBSP configurations, the FSR is a receiving sync pulse either active high or low (based on the FSR polarity bit) and will go inactive during word transfer time. These active and inactive phases can be detected by checking the FSR flag bit MCFFST (bit 3) register or checking the status of the FSR pin. See the FSR flag bit description for details.

Advisory ***McBSP: Read Operations Decrement the McBSP FIFO***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

A read operation from any of the following locations will cause the McBSP receive FIFO contents to decrement by 1, as if the McBSP DRR1 register had been read:

- 0x7001 Reserved
- 0x7401 EV−A T1CNT
- 0x7C01 Reserved

The actual value read from the location is correct and is not affected by this issue.

Workaround(s)

1. Ensure that the McBSP receive FIFO is empty before performing any read operation from any of these addresses.
2. If McBSP traffic is common in the application and a timer count needs to be monitored, consider using a timer other then EV Timer1.

Advisory	*SPI: Slave-Mode Operation*
Revision(s) Affected	0, A, B, C, D, E, F and G
Details	When in slave mode, the SPI does not resynchronize received words based on SPISTE. A spurious SPICLK pulse could therefore throw the data stream out of sync.
Workaround(s)	If the circuit board is not noisy enough to generate spurious SPICLK pulses, then this is not an issue. If noise is an issue, then the McBSP in SPI-slave mode may be used, since the McBSP resynchronizes on each new word.

Advisory	*Clocking: Logic-High Level for XCLKIN Pin*
Revision(s) Affected	0, A, B, C, D, E, F and G
Details	This advisory is applicable only when an external oscillator is used to clock the device. The X1/XCLKIN pin is referenced to the core power supply (V_{DD}), rather than the 3.3-V I/O supply (V_{DDIO}). Therefore, the logic-high level for the input clock should not exceed V_{DD}. This requirement remains the same for future silicon revisions as well.
Workaround(s)	A clamping diode may be used to clamp a buffered clock signal to ensure that the logic-high level does not exceed V_{DD} (1.8 V or 1.9 V). Otherwise, 1.8-V oscillators may be used.

Advisory ***EV: QEP Circuit***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

After a DSP reset, the QEP module fails to detect the first transition that occurs on QEP input pins. (This problem also manifests itself when an external clock is used for the EV timers.) Therefore, if the first transition occurs after a GP timer has been initialized and enabled as the QEP counter (that is, to use QEP as source of clock), the first transition will not be counted by the GP timer. The result is an error of one count in the GP timer out of a total of 1024 counts for a 256-line encoder, or 4096 counts for a 1024-line encoder. However, the issue is not a concern under any of the following conditions:

1. *The first transition happens **before** the GP timer is initialized and enabled as QEP counter.* This ensures that all transitions are counted after initialization.
2. *After the first index pulse is received and if the index pulse is used to recalibrate the GP Timer (through capture interrupt).* The recalibration corrects the error in the GP timer; therefore, from the time the first index pulse is received, the QEP counter becomes accurate.

Workaround(s)

1. Make the first transition happen before the GP timer is initialized and enabled as QEP counter. This is usually the case because typically the rotor shaft is locked to a known position before the GP timer is initialized. Locking the rotor shaft will generate transitions on QEP input pins, unless the rotor shaft is exactly aligned to the known position (which is a rare case). Disturbing the rotor shaft on purpose takes care of the rare case.
2. Use the index pulse of the encoder to recalibrate the GP timer used as QEP counter.
3. The counter has to be forced to count before the application actually uses the QEP. During initialization, configure the internal clock (HSPCLK) to be the counter source. After the first count is done, the counter should be reconfigured for external signals (QEP/TCLKIN) and reset to 0. Now the counter will also count the first edge of the QEP.

Advisory ***QEP: QEP Inputs in GPIO Asynchronous Mode***

Revision(s) Affected

0, A, B, C, D, E, F and G

Details

If any of the QEP input pins are configured for GPIO asynchronous input mode via the GPxQSELn registers, the QEP module may not operate properly. For example, QPOSCNT may not reset or latch properly, and pulses on the input pins may be missed. This is because the QEP peripheral assumes the presence of external synchronization to SYSCLKOUT on inputs to the module.

For proper operation of the QEP module, input GPIO pins should be configured via the GPxQSELn registers for synchronous input mode (with or without qualification). This is the default state of the GPxQSEL registers at reset. All existing QEP peripheral examples supplied by TI also configure the GPIO inputs for synchronous input mode.

The asynchronous mode should not be used for QEP module input pins.

Workaround(s)

Configure GPIO inputs configured as QEP pins for non-asynchronous mode (any GPxQSELn register option except “11b = Asynchronous”).

Advisory	***DEVICE-ID: Register of the Silicon Same for Revision C and Revision D***
Revision(s) Affected	D
Details	The DEVICE-ID register of the revision D silicon contains the same value (0x0003) as that of the revision C silicon.
Workaround(s)	The next revision (revision E) of the silicon has a DEVICE-ID value of 0x0005.

Advisory	***PLL: PLL x4 and x8 Multiplier Ratios***
Revision(s) Affected	0 and A
Details	When the PLL multiplier is set to x4 or x8 (by writing 0004 or 0008, respectively, in the PLLCR register), the watchdog is re-enabled and resets the device upon a WD overflow. With noisy board conditions, this problem may be observed with other PLL multipliers as well.
Workaround(s)	Do not use these multiplier values for these revisions. This is fixed in the next revision of the silicon.

Advisory	***Low-Power Modes − STANDBY Mode***
Revision(s) Affected	0 and A
Details	When the device is put into STANDBY mode, the watchdog is re-enabled and resets the device upon a WD overflow.
Workaround(s)	Do not use the STANDBY mode for these revisions. This is fixed in the next revision of the silicon.

6 Documentation Support

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

For more information regarding the TMS320F281x devices, please see the following publication:

- *TMS320F281x Digital Signal Processors Data Manual*

7 Trademarks

MicroStar BGA™ and TMS320™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

8 Revision History

Changes from September 18, 2020 to December 19, 2023 (from Revision S (September 2020) to Revision T (December 2023)) **Page**

- **Global:** Changed document title from *TMS320F2810, TMS320F2811, TMS320F2812 DSPs* to *TMS320F281x DSPs Silicon Errata*..... 1
- Figure 3-1 (Examples of Device Markings): Updated figure.....2
- Updated Workaround for Advisory; Memory: Flash and OTP Prefetch Buffer Overflow..... 7

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