TMS320R2811 and TMS320R2812 Digital Signal Processors

Silicon Errata

Literature Number: SPRZ226E
June 2004–Revised September 2016

Contents

List of Figures

List of Tables

TMS320R281x Silicon Errata

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320R2811 and TMS320R2812 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]

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**320R2812). 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).

MicroStar BGA, TMS320 are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

3 Device Markings

Figure 1 provides examples of the TMS320R281x 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 device nomenclature.

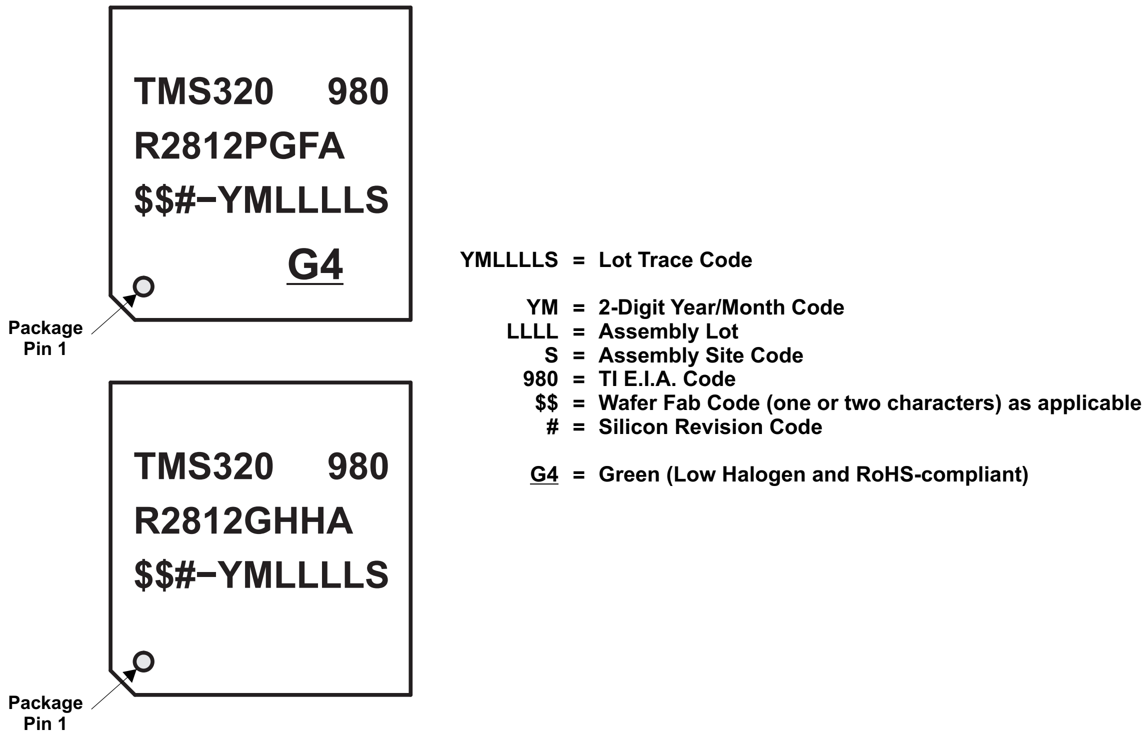


Figure 1. Examples of Device Markings

Table 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 TMS only.
B	Indicates Revision B	0x0002	This silicon revision is available as TMS only.

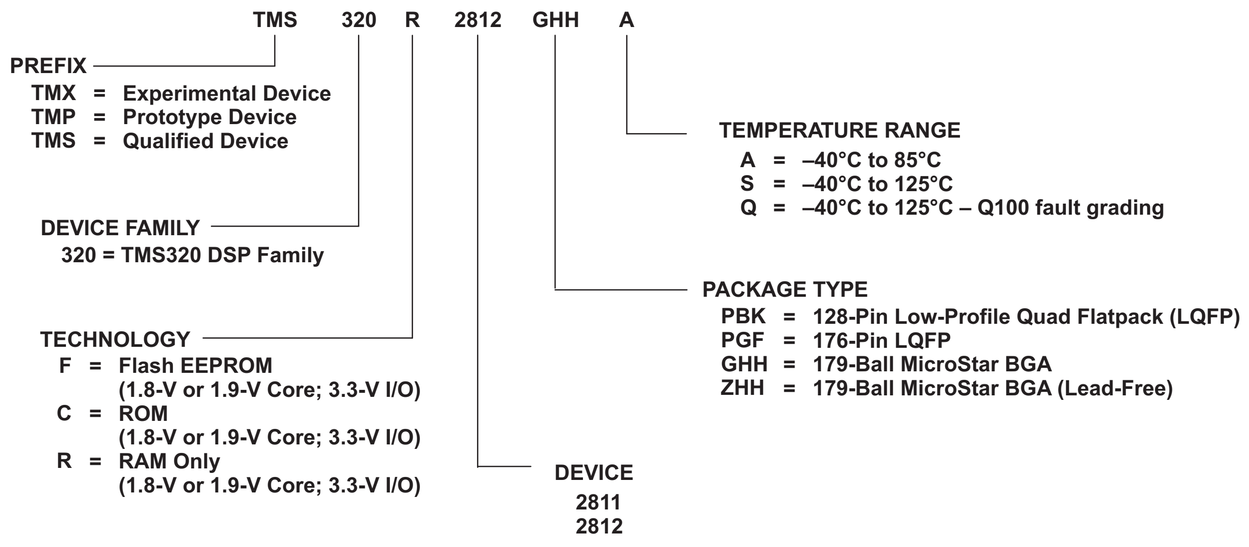

TMS 320 R 2812 GHH A
PREFIX
TMX = Experimental Device
TMP = Prototype Device
TMS = Qualified Device
DEVICE FAMILY
320 = TMS320 DSP Family
TECHNOLOGY
F = Flash EEPROM (1.8-V or 1.9-V Core; 3.3-V I/O)
C = ROM (1.8-V or 1.9-V Core; 3.3-V I/O)
R = RAM Only (1.8-V or 1.9-V Core; 3.3-V I/O)
DEVICE
2811
2812
PACKAGE TYPE
PBK = 128-Pin Low-Profile Quad Flatpack (LQFP)
PGF = 176-Pin LQFP
GHH = 179-Ball MicroStar BGA
ZHH = 179-Ball MicroStar BGA (Lead-Free)
TEMPERATURE RANGE
A = –40°C to 85°C
S = –40°C to 125°C
Q = –40°C to 125°C – Q100 fault grading

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		
	0	**A**	**B**
PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear	Y	Y	Y
Incorrect Reception of KeyValue During Bootloading	Y	Y	Y

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

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
```

4.1.2 Incorrect Reception of KeyValue During Bootloading

Revision(s) Affected: 0, A, B

If an invalid KeyValue is received in SCI, SPI, or Parallel GPIO boot options, control would pass from the boot-ROM code to “Reserved” address space in the device memory map. The reserved space contains TI test code that could activate external peripheral pins belonging to PWM and communication modules. Care should be taken in the design of the boot-up process to ensure correct delivery of the KeyValue and subsequent code words. In the case of SPI or Parallel GPIO bootloading options, absence of the external device (supplying the code words) would lead to an invalid KeyValue situation explained above.

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[1]

TITLE	SILICON REVISION(S) AFFECTED		
	0	**A**	**B**
Memory: Prefetching Beyond Valid Memory	Y	Y	Y
XINTF: XBANK Does Not Properly Extend an Access	Y	Y	Y
SCI: Incorrect Operation of SCI in Address Bit Mode	Y	Y	Y
SCI: Bootloader Does Not Clear the ABD Bit After Auto-Baud Lock	Y	Y	Y
SCI: Bootloader Does Not Clear the ABD Bit Before Auto-Baud Lock	Y	Y	Y
eCAN: Abort Acknowledge Bit Not Set	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
eCAN: Unexpected Cessation of Transmit Operation	Y	Y	Y
WD: WDFLAG Bit Does Not Work as Intended	Y	Y	Y
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
ADC: Reserved Bits in Autosequence Status Register (ADCASEQSR)	Y	Y	Y
ADC: Sequencer Reset While Dual Sequencers Are Running	Y	Y	Y
ADC: Result Register Update Delay	Y	Y	Y
McBSP: Receive FIFO Read Conflict	Y	Y	Y
McBSP: Read Operations Decrement the McBSP FIFO	Y	Y	Y
SPI: Slave-Mode Operation	Y	Y	Y
Clocking: Logic-High Level for XCLKIN Pin	Y	Y	Y
EV: QEP Circuit	Y	Y	Y
QEP: QEP Inputs in GPIO Asynchronous Mode	Y	Y	Y

[1] Y = Yes; N/A = Not Applicable

Advisory *Memory: Prefetching Beyond Valid Memory*

Revision(s) 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.

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 (XINTF) and all memory types (SARAM) 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 *XINTF: XBANK Does Not Properly Extend an Access*

Revision(s) Affected 0, A, B

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.

Table 5. 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, 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 6 shows examples of valid XBANK[BCYC] selections. This list is not exhaustive.

Table 6. 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:

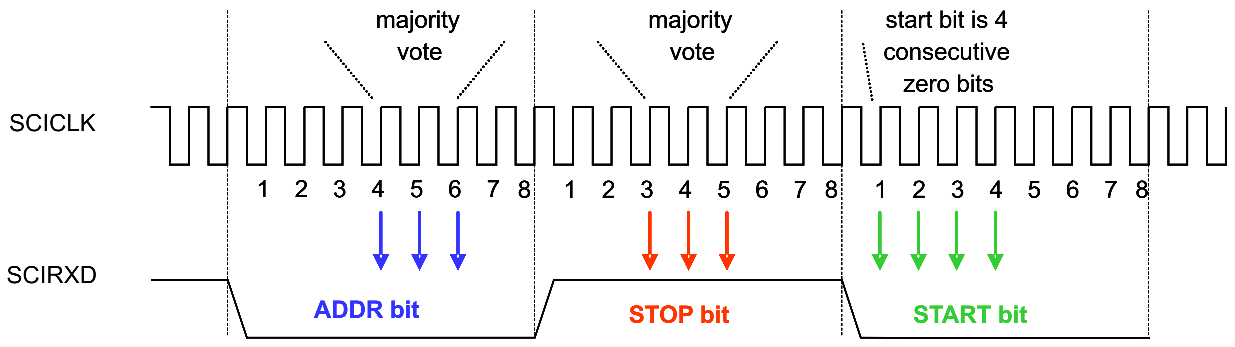
- XBANK[BCYC] != 4 and
- XBANK[BCYC] != 6

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

Revision(s) Affected 0, A, B

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:

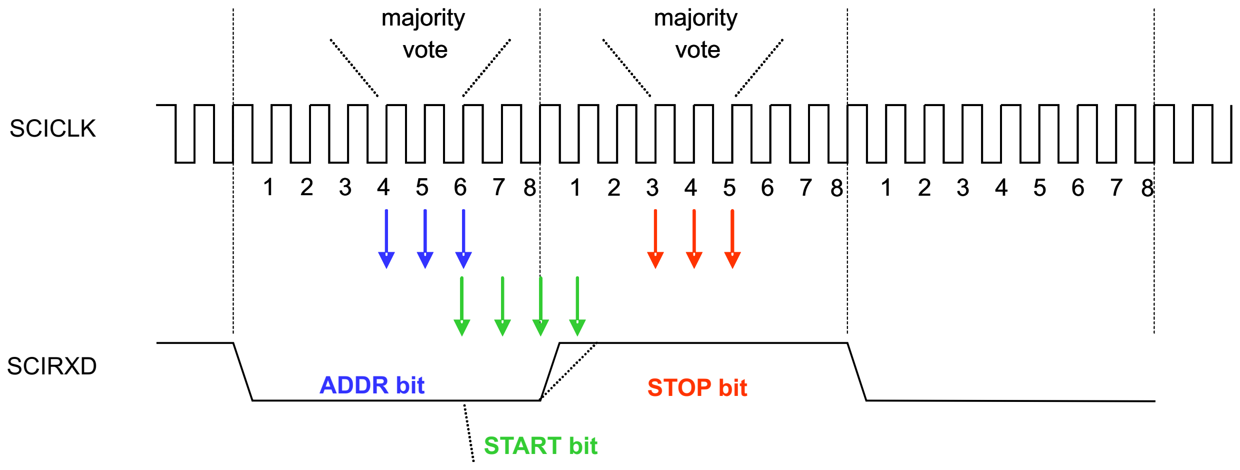


Figure 3. Difference Between Expected and Erroneous Operation of START Bit

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

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

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

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

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.

NOTES:

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

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

Details The WDFLAG bit in R281x 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 $\overline{XRS}$ 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 $\overline{XRS}$ 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 *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

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

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

Details In the TMS320R2812/TMS320R2811 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

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 — *McBSP: Receive FIFO Read Conflict*

Revision(s) Affected 0, A, B

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

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

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

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

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

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”).

5 Documentation Support

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

For further information regarding the TMS320R281x devices, please see the following publication:

- *TMS320R2811, TMS320R2812 Digital Signal Processors Data Manual*

Revision History

Changes from November 12, 2015 to September 28, 2016 (from D Revision (November 2015) to E Revision) **Page**

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		