List of Figures

List of Tables

TMS320F281x, TMS320C281x Silicon Errata

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 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**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 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 1. Some prototype devices may have markings different from those illustrated. Figure 2 shows an example of device nomenclature.

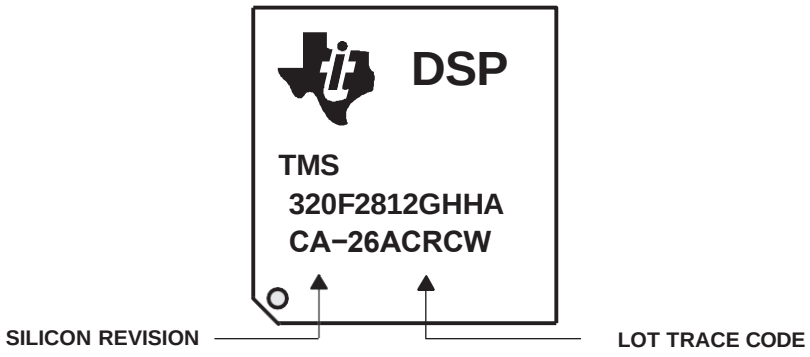

Figure 1. Example of Device Markings

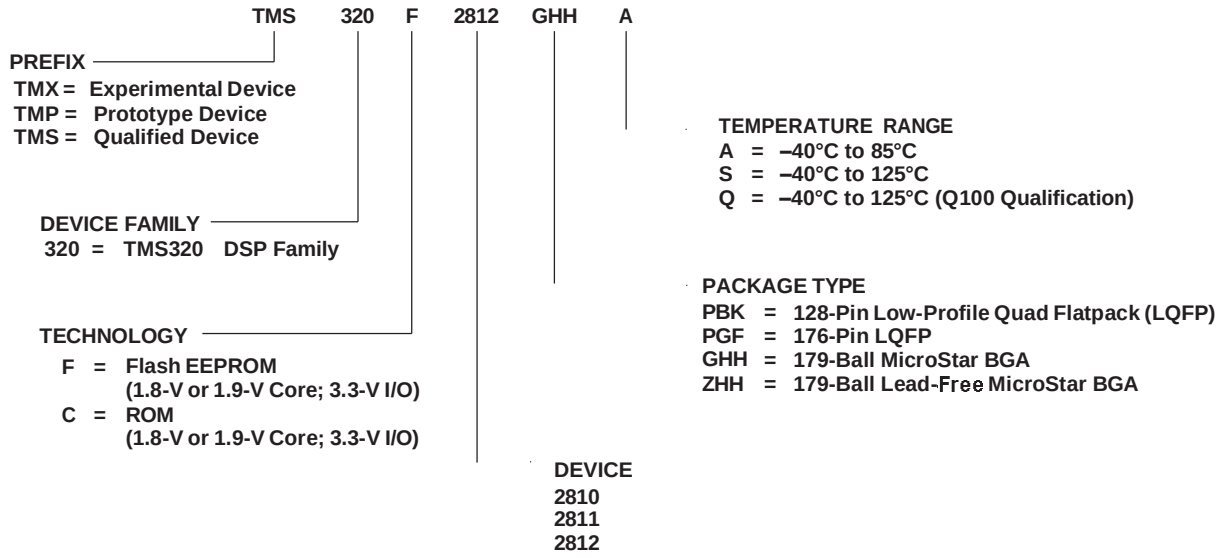

Figure 2. Example of Device Nomenclature

Table 1. Determining Silicon Revision From Lot Trace Code (F281x)

SECOND LETTER IN PREFIX OF LOT TRACE CODE	SILICON REVISION	REVISION ID (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)

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 and Table 3 show which silicon revision(s) are affected by each usage note.

Table 2. List of Usage Notes for F281x(1)

TITLE	SILICON REVISION(S) AFFECTED							
	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

Table 3. List of Usage Notes for C281x(1)

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

(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: TMS320F281x: 0, A, B, C, D, E, F and G
TMS320C281x: 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
```

4.2 Known Design Exceptions to Functional Specifications

Table 4. Table of Contents for Advisories

Table 5 and Table 6 show which silicon revision(s) are affected by each advisory.

Table 5. List of Advisories for F281x[1]

TITLE	SILICON REVISION(S) AFFECTED							
	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
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
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

Table 6. List of Advisories for C281x[1]

TITLE	SILICON REVISION(S) AFFECTED		
	0	A	B
Memory: Prefetching Beyond Valid Memory	Y	Y	Y
Memory: Program Reads From Flash/ROM Memory	Y	Y	Y
Memory: Flash and OTP Prefetch Buffer Overflow	Y	Y	Y
Memory: Set Device Emulation Register Bits for On-Chip RAM Performance	N/A	N/A	N/A
Memory: OTP Memory	N/A	N/A	N/A
XINTF: XBANK Does Not Properly Extend an Access	Y	Y	Y
XINTF: XREADY Signal is not Sampled Properly When Using Asynchronous Sampling Mode	N/A	N/A	N/A
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
WD: WDFLAG Bit Does Not Work as Intended	Y	Y	Y
WD: A Low Output on GPIOF14 Can Disable the PLL and Watchdog if the Watchdog Fires a Reset	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
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
ADC: Device Has Higher Gain Error Than the Design Goal of 1% FSR on All of the B0−B7 Channels	N/A	N/A	N/A
ADC: Device Has Higher Offset Error Than the Design Goal (0.5 to 1%) on Some Channels	N/A	N/A	N/A
ADC: Device Has Higher Non-Linearity Than the Design Goal of 2 LSBs	N/A	N/A	N/A
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
DEVICE-ID: Register of the Silicon Same for Revision C and Revision D	N/A	N/A	N/A
PLL: PLL x4 and x8 Multiplier Ratios	N/A	N/A	N/A
Low-Power Modes − STANDBY Mode	N/A	N/A	N/A

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

Advisory — *Memory: Prefetching Beyond Valid Memory*

Revision(s) Affected

TMS320F281x: 0, A, B, C, D, E, F and G

TMS320C281x: 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 and all memory types (Flash/ROM, OTP, SARAM, XINTF) on the device. Prefetching across the boundary between two valid memory blocks is ok.

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

TMS320F281x: 0, A, B, C, D, E, F and G

TMS320C281x: 0, A, B

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

TMS320F281x: 0, A, B, C, D, E, F and G

TMS320C281x: 0, A, B

Details

This advisory applies to code executing from flash or OTP with the flash prefetch buffer enabled. On ROM devices, this applies to the ROM that replaces flash and OTP.

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 include 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:

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 the C28x compiler as of V4.1.4 and V5.0 beta3.