TMS320F2805x Piccolo™ MCUs Silicon Revisions A, 0

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320F2805x microcontrollers (MCUs).

The updates are applicable to:

- 80-pin Low-Profile Quad Flatpack, PN Suffix

2 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, Texas Instruments (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**320F28055). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (with TMX for devices and TMDX for tools) through fully qualified production devices and tools (with TMS for devices and TMDS for tools).

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

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

TMS Fully qualified production device

Support tool development evolutionary flow:

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

TMDS Fully qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:
"Developmental product is intended for internal evaluation purposes."

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

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

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

3 Device Markings

Figure 1 provides an example of the 2805x device markings and defines each of the markings. The device revision can be determined by the symbols marked on the top of the package as shown in Figure 1. Some prototype devices may have markings different from those illustrated. Figure 2 shows an example of the device nomenclature.

Figure 1. Example of Device Markings

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

PACKAGE MARKING CODE	SILICON REVISION	REVISION ID Address: 0x0883	COMMENTS
Blank	Indicates Revision 0	0x0000	This silicon revision is available as TMX.
A	Indicates Revision A	0x0000	This silicon revision is available as TMS.

(1) Boot-ROM contents changed between Rev. 0 silicon and Rev. A silicon. For more details, see the *TMS320x2805x Piccolo Technical Reference Manual*.

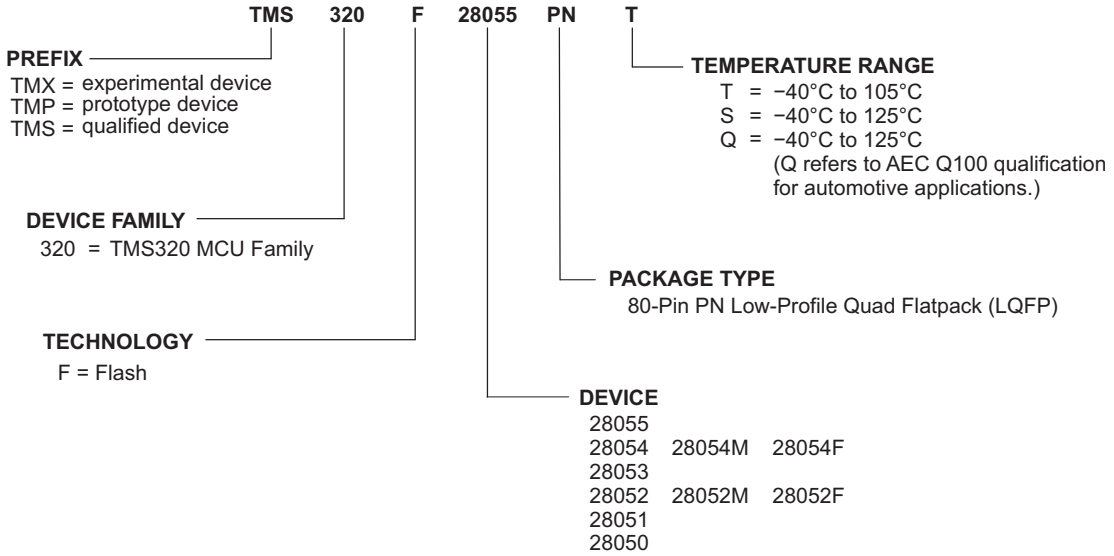

Figure 2. Device Nomenclature

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
PIE: Spurious Nested Interrupt After Back-to-Back PIEACK Write and Manual CPU Interrupt Mask Clear	Yes	Yes
CAN Bootloader: Internal Oscillator Tolerance is Not Sufficient for CAN Operation at High Temperatures	Yes	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

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 CAN Bootloader: Internal Oscillator Tolerance is Not Sufficient for CAN Operation at High Temperatures

Revision(s) Affected: 0, A

The CAN bootloader in the device's boot ROM uses the internal oscillator as the source for the CAN bit clock. At high temperatures, the frequency of the internal oscillator can deviate enough to prevent messages from being received.

Workaround: Recalibrate the internal oscillator before invoking the CAN bootloader. This can be done in application code. For more flexibility, a wrapper function may be programmed into the device's OTP memory. See the *Using the Piccolo™ CAN Bootloader at High Temperature Application Report* for details on how to implement this workaround.

4.2 Known Design Exceptions to Functional Specifications

Table 3. Table of Contents for Advisories

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

Table 4. List of Advisories

TITLE	SILICON REVISION(S) AFFECTED	
	0	A
ADC: Initial Conversion	Yes	Yes
ADC: Temperature Sensor Minimum Sample Window Requirement	Yes	Yes
ADC: Offset Self-Recalibration Requirement	Yes	Yes
ADC: ADC can Become Non-Responsive When ADCNONOVERLAP or RESET is Written During a Conversion	Yes	Yes
Memory: Prefetching Beyond Valid Memory	Yes	Yes
eCAN: Abort Acknowledge Bit Not Set	Yes	Yes
eCAN: Unexpected Cessation of Transmit Operation	Yes	Yes
GPIO: GPIO Qualification	Yes	Yes
GPIO: GPIO Pullup Enabled on Reset	Yes	Yes
Zero-Pin Oscillator: Modification to Oscillator Frequency Parameter	Yes	Yes
Watchdog: Incorrect Operation of CPU Watchdog When WDCLK Source is OSCCLKSRC2	Yes	Yes
Oscillator: CPU Clock Switching to INTOSC2 May Result in Missing Clock Condition After Reset	Yes	Yes
eQEP: eQEP Inputs in GPIO Asynchronous Mode	Yes	Yes
eQEP: Position Counter Incorrectly Reset on Direction Change During Index	Yes	Yes
ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window	Yes	Yes

NOTE: Boot-ROM contents changed between Rev. 0 silicon and Rev. A silicon. For more details, see the *TMS320x2805x Piccolo Technical Reference Manual*.

Advisory — *ADC: Initial Conversion*

Revision(s) Affected 0, A

Details When the ADC conversions are initiated by any source of trigger in either sequential or simultaneous sampling mode, the first sample may not be the correct conversion result.

Workaround(s) For sequential mode, discard the first sample at the beginning of every series of conversions. For instance, if the application calls for a given series of conversions, SOC0→SOC1→SOC2, to initiate periodically, then set up the series instead as SOC0→SOC1→SOC2→SOC3 and only use the last three conversions, ADCRESULT1, ADCRESULT2, ADCRESULT3, thereby discarding ADCRESULT0.

For simultaneous sample mode, discard the first sample of both the A and B channels at the beginning of every series of conversions.

User application should validate if this workaround is acceptable in their application.

The following is applicable:

- For 30-MHz operation and below, this issue is fixed completely by writing a 1 to the ADCNONOVERLAP and CLKDIV2EN bits in the ADCTRL2 register. This action will give a 30-MHz ADC clock when the CPU clock = 60 MHz, and will only allow the sampling of ADC channels when the ADC is finished with any pending conversion.

Advisory — *ADC: Temperature Sensor Minimum Sample Window Requirement*

Revision(s) Affected 0, A

Details If an insufficient sample window is used, the result of a temperature sensor conversion can have a large error, making the result unreliable for the system.

Workaround(s)

1. If double-sampling of the temperature sensor is used to avoid the corrupted first sample issue, the temperature sensor result is valid. Double-sampling is equivalent to giving the sample-and-hold (S/H) circuit adequate time to charge.
2. In all other conditions, the sample-and-hold window used to sample the temperature sensor should not be less than 550 ns.

Advisory — *ADC: Offset Self-Recalibration Requirement*

Revision(s) Affected

0, A

Details

The factory offset calibration from Device_cal() may not ensure that the ADC offset remains within specifications under all operating conditions in the customer's system.

Workaround(s)

- To ensure that the offset remains within the data sheet's “single recalibration” specifications, perform the AdcOffsetSelfCal() function after Device_cal() has completed and the ADC has been configured.
- To ensure that the offset remains within the data sheet's “periodic recalibration” specifications, perform the AdcOffsetSelfCal() function periodically with respect to temperature drift.

For more details on AdcOffsetSelfCal(), refer to the ADC Zero Offset Calibration section in the "Analog-to-Digital Converter and Comparator" chapter of the *TMS320x2805x Piccolo Technical Reference Manual*.

Advisory

ADC: ADC can Become Non-Responsive When ADCNONOVERLAP or RESET is Written During a Conversion

Revision(s) Affected

0, A

Details

The ADC can get into a non-responsive state when the ADCCTL2[ADCNONOVERLAP] is modified while a conversion is in progress. When in this condition, no further conversion from the ADC will be possible without a device reset.

There are two different ways to cause this condition:

- Writing to ADCCTL2[ADCNONOVERLAP] while a conversion is in progress.
- Writing to ADCCTL1[RESET] while a conversion is in progress.

Workaround(s)

Follow this sequence to modify ADCCTL2[ADCNONOVERLAP] or write ADCCTL1[RESET]:

1. Set all SOC trigger sources ADCSOCxCTL[TRIGSEL] to 0.
2. Set all ADCINTSOCSEL1/2 to 0.
3. Ensure there is not another SOC pending (This can be accomplished by polling SOC Flags).
4. Wait for all conversions to complete.
   a. ADCCTL2[CLKDIV2EN] = 0, ADCCTL2[CLKDIV4EN] = x → (ACQPS + 13) * 1 SYSCLKs
   b. ADCCTL2[CLKDIV2EN] = 1, ADCCTL2[CLKDIV4EN] = 0 → (ACQPS + 13) * 2 SYSCLKs
   c. ADCCTL2[CLKDIV2EN] = 1, ADCCTL2[CLKDIV4EN] = 1 → (ACQPS + 13) * 4 SYSCLKs
5. Modify ADCCTL2[ADCNONOVERLAP] or write ADCCTL1[RESET].

An example code follows.

```
EALLOW;
// Set all SOC trigger sources to software
AdcRegs.ADCSOC0CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC1CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC2CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC3CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC4CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC5CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC6CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC7CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC8CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC9CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC10CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC11CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC12CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC13CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC14CTL.bit. TRIGSEL  = 0;
AdcRegs.ADCSOC15CTL.bit. TRIGSEL  = 0;

// Set all ADCINTSOCSEL1/2 to 0.
    AdcRegs.ADCINTSOCSEL1.bit.SOC0  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC1  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC2  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC3  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC4  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC5  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC6  = 0;
    AdcRegs.ADCINTSOCSEL1.bit.SOC7  = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC8  = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC9  = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC10 = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC11 = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC12 = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC13 = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC14 = 0;
    AdcRegs.ADCINTSOCSEL2.bit.SOC15 = 0;

// Ensure there is not another SOC pending
while (AdcRegs.ADCSOCFLG1.all != 0x0);

// Wait for conversions to complete
// Delay time based on ACQPS = 6 , ADCCTL2[CLKDIV2EN] = 1, ADCCTL2[CLKDIV4EN] = 0
// 7 + 13 ADC Clocks = 20 ADCCLKS -> 40 SYSCLKS
asm(" RPT#40||NOP");
// ADCCTL2[ADCNONOVERLAP] = <new value>;
// ADCCTL1[RESET] = 1;

EDIS;
```

Advisory — *Memory: Prefetching Beyond Valid Memory*

Revision(s) Affected

0, A

Details

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

Workaround

The prefetch queue is 8 x16 words in depth. Therefore, code should not come within 8 words of the end of valid memory. This restriction applies to all memory regions and all memory types (flash, OTP, SARAM) on the device. Prefetching across the boundary between two valid memory blocks is all right.

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

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

Advisory *eCAN: Abort Acknowledge Bit Not Set*

Revision(s) Affected 0, A

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 the message 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 or 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} = [(mailbox_number) * 2] + 3 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 one 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: Unexpected Cessation of Transmit Operation*

Revision(s) Affected 0, A

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 — *GPIO: GPIO Qualification*

Revision(s) Affected 0, A

Details If a GPIO pin is configured for "n" SYSCLKOUT cycle qualification period (where 1 ≤ n ≤ 510) with "m" qualification samples (m = 3 or 6), it is possible that an input pulse of [n * m – (n – 1)] width may get qualified (instead of n * m). The occurrence of this incorrect behavior depends upon the alignment of the asynchronous GPIO input signal with respect to the phase of the internal prescaled clock, and hence, is not deterministic. The probability of this kind of wrong qualification occurring is "1/n".

Worst-case example:

If n = 510, m = 6, a GPIO input width of (n * m) = 3060 SYSCLKOUT cycles is required to pass qualification. However, because of the issue described in this advisory, the minimum GPIO input width which may get qualified is [n * m – (n – 1)] = 3060 – 509 = 2551 SYSCLKOUT cycles.

Workaround(s) None. Ensure a sufficient margin is in the design for input qualification.

Advisory — *GPIO: GPIO Pullup Enabled on Reset*

Revision(s) Affected 0, A

Details Pullup is enabled on reset for the following pins: GPIO20, GPIO21, GPIO24, GPIO30, GPIO31, GPIO40, and GPIO42.

Workaround(s) If these pins being pulled high (upon reset) is an issue in the application, use an external pulldown resistor to overdrive the default state of the internal pullups. A value of 1.8 kΩ is adequate to achieve this. However, TI recommends that the design be validated for proper operation to ensure that other components connected to these pins operate correctly in the presence of this pulldown resistor.

Advisory — *Zero-Pin Oscillator: Modification to Oscillator Frequency Parameter*

Revision(s) Affected

0, A

Details

The zero-pin oscillator is now specified with the center frequency at a defined temperature and temperature coefficient to calculate the absolute frequency at any operational temperature. Customers will need to check their temperature profile to ensure the zero-pin oscillator meets their requirement.

Workaround(s)

If the frequency output does not meet a needed tolerance, software compensation can be used to achieve improved accuracy at any temperature. For accuracy specification, see the *TMS320F2805x Piccolo™ Microcontrollers Data Manual*.

Advisory — *Watchdog: Incorrect Operation of CPU Watchdog When WDCLK Source is OSCCLKSRC2*

Revision(s) Affected 0, A

Details When OSCCLKSRC2 is used as the clock source for CPU watchdog, the watchdog may fail to generate a device reset intermittently.

Workaround(s) WDCLK should be sourced only from OSCCLKSRC1 (INTOSC1). The CPU may be sourced from OSCCLKSRC2 or OSCCLKSRC1 (INTOSC1).

Advisory

Oscillator: CPU Clock Switching to INTOSC2 May Result in Missing Clock Condition After Reset

Revision(s) Affected

0, A

Details

After at least two system resets (not including power-on reset), when the application code attempts to switch the CPU clock source to internal oscillator 2, a missing clock condition will occur, and the clock switching will fail under the following conditions:

- X1 and X2 are unused (X1 is always tied low when unused).
- GPIO38 (muxed with TCK and XCLKIN) is used as JTAG TCK pin only.
- JTAG emulator is disconnected.

The missing clock condition will recover only after a power-on reset when the failure condition occurs.

Workaround(s)

Before switching the CPU clock source to INTOSC2 via the OSCCLKSRCSEL and OSCCLKSRC2SEL bits in the CLKCTL register, the user must toggle the XCLKINOFF and XTALOSCOFF bits in the CLKCTL register as illustrated in the below sequence:

```
CLKCTL |= 0x6000;     // XCLKINOFF = 1, XTALOSCOFF = 1
CLKCTL &=~0x6000;     // XCLKINOFF = 0, XTALOSCOFF = 0
CLKCTL |= 0x6000;     // XCLKINOFF = 1, XTALOSCOFF = 1
CLKCTL &=~0x6000;     // XCLKINOFF = 0, XTALOSCOFF = 0
CLKCTL |= 0x6000;     // XCLKINOFF = 1, XTALOSCOFF = 1
```

Once the above procedure is executed, then the OSC2 selection switches can be configured.

If the JTAG emulator is connected, and GPIO38 (TCK) is toggling, then the above procedure is unnecessary, but will do no harm.

If no clock is applied to GPIO38, TI also recommends that a strong pullup resistor on GPIO38 be added to V_{DDIO}.

Advisory *eQEP: eQEP Inputs in GPIO Asynchronous Mode*

Revision(s) Affected 0, A

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

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

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

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

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

Revision(s) Affected 0, A

Details While using the PCRM = 0 configuration, if the direction change occurs when the index input is active, the position counter (QPOSCNT) could be reset erroneously, resulting in an unexpected change in the counter value. This could result in a change of up to ±4 counts from the expected value of the position counter and lead to unexpected subsequent setting of the error flags.

While using the PCRM = 0 configuration [that is, Position Counter Reset on Index Event (QEPCTL[PCRM] = 00)], if the index event occurs during the forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

If the direction change occurs while the index pulse is active, the module would still continue to look for the relative quadrature transition for performing the position counter reset. This results in an unexpected change in the position counter value.

Workaround(s) Do not use the PCRM = 0 configuration if the direction change could occur while the index is active and the resultant change of the position counter value could affect the application.

Other options for performing position counter reset, if appropriate for the application [such as Index Event Initialization (IEI)], do not have this issue.

Advisory

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

Revision(s) Affected

0, A

Details

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

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

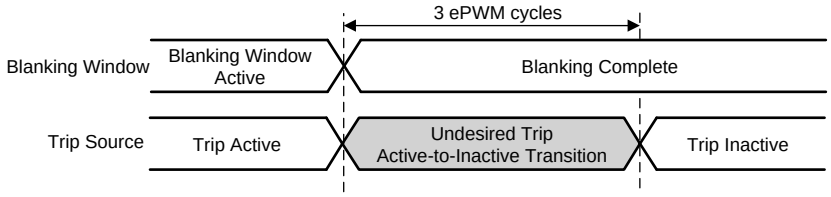

Figure 3. Undesired Trip Event and Blanking Window Expiration

Figure 4 illustrates the two potential ePWM outputs possible if the trip event ends within 1 cycle before or 3 cycles after the blanking window closes.

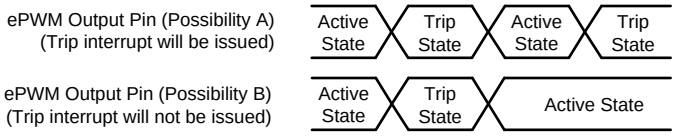

Figure 4. Resulting Undesired ePWM Outputs Possible

Workaround(s)

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

5 Documentation Support

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

For more information regarding the TMS320F2805x Piccolo devices, see the following documents:

- *TMS320F2805x Piccolo™ Microcontrollers Data Manual*
- *TMS320x2805x Piccolo Technical Reference Manual*

Trademarks

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

Revision History

Changes from March 22, 2016 to September 28, 2018 (from D Revision (March 2016) to E Revision) **Page**

- Section 4.2 (Known Design Exceptions to Functional Specifications): Added ADC: ADC can Become Non-Responsive When ADCNONOVERLAP or RESET is Written During a Conversion advisory. 8
- Section 4.2: Added ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window advisory. 17

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

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