Errata

F29H85x, F29P58x, and F29P32x Real-Time MCUs Silicon Errata (Silicon Revisions C, B, A, 0)

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories). This document may also contain usage notes. Usage notes describe situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness.

Table of Contents

List of Figures

List of Tables

1 Usage Notes and Advisories Matrices

Table 1-1 lists all usage notes and the applicable silicon revisions. Table 1-2 lists all advisories, modules affected, and the applicable silicon revisions.

1.1 Usage Notes Matrix

Table 1-1. Usage Notes Matrix

NUMBER	TITLE	SILICON REVISIONS AFFECTED			
		0	A	B	C
Section 4.1.1	Security: New TI Keys Programmed on Silicon Revision B Devices	Yes	Yes	No	No

1.2 Advisories Matrix

Table 1-2. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED			
		0	A	B	C
ADC	ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set	Yes	Yes	Yes	Yes
CAN	CAN boot mode is not supported in silicon revision 0, A	Yes	Yes	No	No
MCAN	Message Order Inversion When Transmitting From Dedicated Tx Buffers Configured With Same Message ID	Yes	Yes	Yes	Yes
ePWM	ePWM: An ePWM Glitch can Occur if a Trip Remains Active at the End of the Blanking Window	Yes	Yes	Yes	Yes
ePWM	ePWM: ePWM TZFRC and TZCLR Events may be Missed When PERCLKDIVSEL.EPWMCLKDIV = 1	Yes	Yes	Yes	Yes
ePWM	ePWM: ESM Source for Trip Zone is not Supported	Yes	Yes	Yes	Yes
ePWM	ePWM: ePWM One-Shot/CBC Trip Event DCxEVTy.force Does Not Set the Trip Condition	Yes	Yes	Yes	Yes
ePWM	ePWM: For ePWMs Using Global Load in One-Shot Load Mode, Global Load of Registers may get Delayed When a Write to the GLDCTL2.OSHTLD Register Occurs Within 3 TBCLKs of the Global Load Event	Yes	Yes	Yes	Yes
Flash	Flash: Stand-alone CPU1/CPU3 Reset With Flash Prefetch Enabled may Cause NMI to CPU1/CPU3	Yes	Yes	No	No
FOTA	FOTA: Secure FOTA With Encryption does not Work	Yes	Yes	No	No
HRPWM	HRPWM: HRPWM High-Resolution Period Shadow to Active Loading Occurs Every ZERO Event, Even if Shadow to Active Load for Period is Set to Only Load on SYNC	Yes	Yes	Yes	Yes
HSM	HSM: HSM ROM Code Does Not Boot HSMRT Image of Size Greater Than 191KB	Yes	Yes	No	No
LIN	LIN: LIN Unable to Wake Up Using 0xF0 Wake-Up Key	Yes	Yes	Yes	Yes
MCD	MCD: Missing Clock Detect Should be Disabled When the PLL is Enabled (PLLCLKEN = 1)	Yes	Yes	Yes	Yes
MEMSS	MEMSS: Data Line Buffer (DLB) for RAM Causes Data Coherency Issue	Yes	Yes	Yes	Yes
ROM	ROM: By Default, GPIO4 is Configured as ERRORSTS by ROM Code and Driven High	Yes	No	No	No
SDFM	SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events	Yes	Yes	Yes	Yes
SDFM	SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events	Yes	Yes	Yes	Yes
SDFM	SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events	Yes	Yes	Yes	Yes
C29 CPU Subsystem	C29 CPU Subsystem: DTHE Interrupts and DMA Events Not Triggered in C29 CPU Subsystem for HS-FS Devices	Yes	Yes	No	No
System	System: Device Reset Remains Asserted When VDD Voltage Ramps Before VDDIO	Yes	Yes	Yes	Yes
System	System: Pending Misaligned Reads in the Pipeline After CPU Goes to Fault State Preventing NMI Vector Fetch	Yes	Yes	No	No
System	System: Reset will cause CPU to get stuck in BOOTROM NMI handler	Yes	Yes	No	No
System	System: Issuing device reset (XRSn) can cause unexpected fault if SYSCLKDIVSEL.PLLSYSCLKDIV = 0	Yes	Yes	Yes	Yes
System	System: Internal device reset could cause the device to be stuck in the reset loop	Yes	Yes	No	No
UART	UART: UART FIFO Gets Cleared on Continuous Debugger Reads	Yes	Yes	Yes	Yes

Table 1-2. Advisories Matrix (continued)

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED			
		0	A	B	C
VSSOSC	VSSOSC: Coupling From Adjacent Pins X1 or X2 may Prevent Proper Boot	Yes	No	No	No

2 Nomenclature, Package Symbolization, and Revision Identification

2.1 Device and Development-Support Tool Nomenclature

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 (TMDX) through fully qualified production tools (TMDS).

Device development evolutionary flow:

X Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

P Prototype device that is not necessarily the final silicon die and may not necessarily meet final electrical specifications.

null Production version of the silicon die that is fully qualified.

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.

X and P devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Production 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 (X or P) 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.

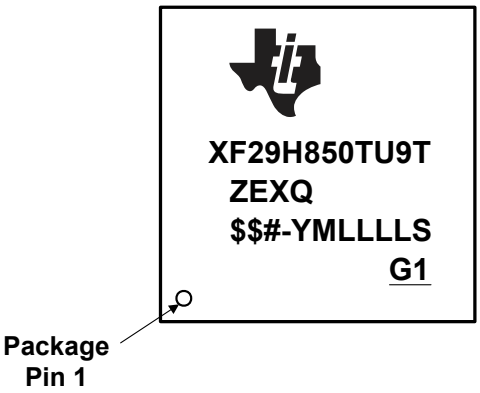
2.2 Devices Supported

This document supports the following devices:

- F29H850TU
- F29H859TU-Q1
- F29H859TM-Q1
- F29H850DU
- F29H859DU-Q1
- F29H850DM
- F29H859DM-Q1
- F29P589DU-Q1
- F29P580DM
- F29P589DM-Q1
- F29P329SM-Q1

2.3 Package Symbolization and Revision Identification

Figure 2-1, Figure 2-2, Figure 2-3, and Figure 2-4 show the package symbolization. Table 2-1 lists the silicon revision codes.

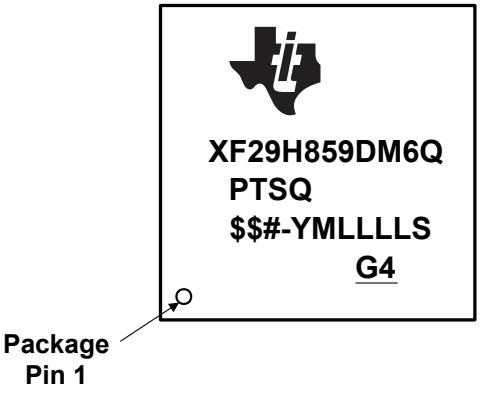


YMLLLLS = Lot Trace Code

$$ = Wafer Fab Code (one or two characters) as applicable
 # = Silicon Revision Code
YM = 2-digit Year/Month Code
LLLL = Assembly Lot Code
S = Assembly Site Code

G1 = Green (Low Halogen and RoHS-compliant)

Figure 2-1. Package Symbolization for ZEX Package

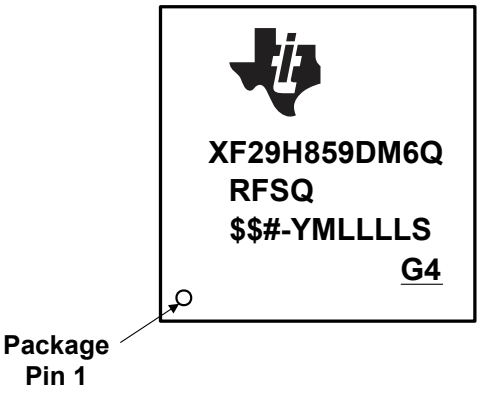


YMLLLLS = Lot Trace Code

$$ = Wafer Fab Code (one or two characters) as applicable
 # = Silicon Revision Code
YM = 2-digit Year/Month Code
LLLL = Assembly Lot Code
S = Assembly Site Code

G4 = Green (Low Halogen and RoHS-compliant)

Figure 2-2. Package Symbolization for PTS Package



YMLLLLS = Lot Trace Code

$$ = Wafer Fab Code (one or two characters) as applicable
 # = Silicon Revision Code
YM = 2-digit Year/Month Code
LLLL = Assembly Lot Code
S = Assembly Site Code

G4 = Green (Low Halogen and RoHS-compliant)

Figure 2-3. Package Symbolization for RFS Package

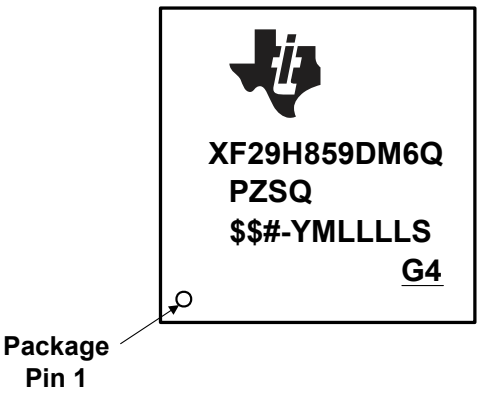


Figure 2-4. Package Symbolization for PZS Package

Table 2-1. Revision Identification

SILICON REVISION CODE	SILICON REVISION	REVID(1) Address: 0x3018 0028	COMMENTS(2)
Blank	0	0x0000 0001	This silicon revision is available as pre-production.
A	A	Not applicable(3)	This silicon revision is available as pre-production.
B	B	0x0000 0003	This silicon is available as production
C	C	0x0000 0004	This silicon is available as production

(1) Silicon Revision ID
(2) For orderable device numbers, see the PACKAGING INFORMATION table in the *F29H85x, F29P58x, and F29P32x Real-Time Microcontrollers* data sheet.
(3) Refer to the Silicon Revision Code in the Package Symbolization figures.

3 Silicon Revision B Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

3.1 Silicon Revision B Usage Notes

This section lists all the usage notes that are applicable to silicon revision B and earlier silicon revisions.

3.2 Silicon Revision B Advisories

This section lists all the advisories that are applicable to silicon revision B and earlier silicon revisions.

Advisory ***ADC: Interrupts may Stop if INTxCONT (Continue-to-Interrupt Mode) is not Set***

Revisions Affected

0, A, B, C

Details

If ADCINTSELxNx[INTxCONT] = 0, then interrupts will stop when the ADCINTFLG is set and no additional ADC interrupts will occur.

When an ADC interrupt occurs simultaneously with a software write of the ADCINTFLGCLR register, the ADCINTFLG will unexpectedly remain set, blocking future ADC interrupts.

Workarounds

1. Use Continue-to-Interrupt Mode to prevent the ADCINTFLG from blocking additional ADC interrupts:

```
ADCINTSEL1N2[INT1CONT] = 1;
ADCINTSEL1N2[INT2CONT] = 1;
ADCINTSEL3N4[INT3CONT] = 1;
ADCINTSEL3N4[INT4CONT] = 1;
```

2. Ensure there is always sufficient time to service the ADC ISR and clear the ADCINTFLG before the next ADC interrupt occurs to avoid this condition.
3. Check for an overflow condition in the ISR when clearing the ADCINTFLG. Check ADCINTOVF immediately after writing to ADCINTFLGCLR; if it is set, then write ADCINTFLGCLR a second time to ensure the ADCINTFLG is cleared. The ADCINTOVF register will be set, indicating an ADC conversion interrupt was lost.

```
AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;        //clear INT1 flag
if(1 == AdcaRegs.ADCINTOVF.bit.ADCINT1)       //ADCINT overflow
{
    AdcaRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;    //clear INT1 again
// If the ADCINTOVF condition will be ignored by the application
// then clear the flag here by writing 1 to ADCINTOVFCLR.
// If there is a ADCINTOVF handling routine, then either insert
// that code and clear the ADCINTOVF flag here or do not clear
// the ADCINTOVF here so the external routine will detect the
// condition.
//    AdcaRegs.ADCINTOVFCLR.bit.ADCINT1 = 1;  // clear OVF
}
```

Advisory	***Message Order Inversion When Transmitting From Dedicated Tx Buffers Configured With Same Message ID***
Revisions Affected	0, A, B, C
Details	Multiple Tx Buffers are configured with the same Message ID. Transmission of these Tx buffers is requested sequentially in ascending order with a delay between the individual Tx requests. Depending on the delay between the individual Tx requests, the Tx Buffers may not be transmitted in the expected ascending order of the Tx Buffer number.
Workarounds	First, write the group of Tx messages with same Message ID to the Message RAM. Then, request transmission of all of these messages concurrently by a single write access to **TXBAR**. Use the Tx FIFO instead of dedicated Tx Buffers for the transmission of several messages with the same Message ID in a specific order.

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

Revisions Affected

0, A, B, C

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 3-1 illustrates the time period which could result in an undesired ePWM output.

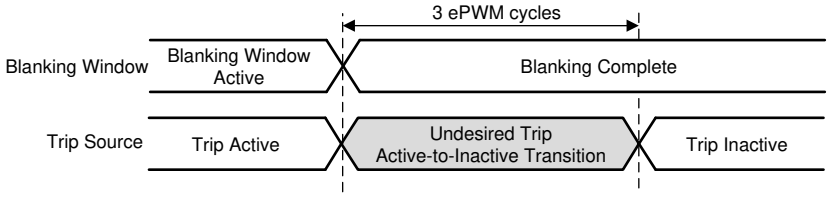


Figure 3-1. Undesired Trip Event and Blanking Window Expiration

Figure 3-2 illustrates the two potential ePWM outputs possible if the trip event ends within 1 cycle before or 3 cycles after the blanking window closes.

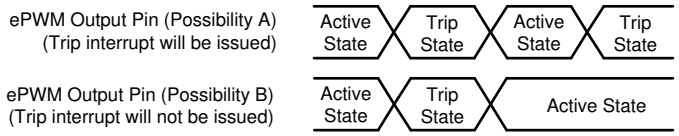


Figure 3-2. Resulting Undesired ePWM Outputs Possible

Workaround

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

Advisory ***ePWM: ePWM TZFRC and TZCLR Events may be Missed When PERCLKDIVSEL.EPWMCLKDIV = 1***

Revisions Affected

0, A, B, C

Details

The TZFRC bit is used for software-forced trip events, while the TZCLR bit is used for clearing the trip-zone events. On devices with EPWMCLKDIV, the TZFRC and TZCLR write may be missed and leave the output unaffected if PERCLKDIVSEL.EPWMCLKDIV is programmed to 1. This bit is programmed to 1 by default (EPWMCLK is PLLSYSCLK/2).

Workaround

1. Configure **EPWMCLK = PLLSYSCLK (PERCLKDIVSEL.EPWMCLKDIV = 0)**.
2. If the user has to configure **EPWMCLK = PLLSYSCLK/2 (PERCLKDIVSEL.EPWMCLKDIV = 1)**, select one of the reserved mux inputs of EPWMXBAR to be used for the trip using the following driverlib software sequence.

PWMXBAR → Digital Compare → Trip Zone

1. Configure both trip zone actions for Digital Compare Output A Event 1/2 Action On EPWMxA and Digital Compare Output B Event 1/2 Action On EPWMxB.
   - EPWM_setTripZoneAction()
2. Configure the input signals for TRIPIN1-15 or the ORed Combinational logic of TRIPIN1-15.
   - EPWM_selectDigitalCompareTripInput()
3. Configure the Digital Compare condition for DCAEVT1/2 and DCBEVT1/2.
   - EPWM_setTripZoneDigitalCompareEventCondition()
4. Configure the PWMXBAR Input to the Digital Compare Submodule to reserved.
   - XBAR_selectEpwmXbarInputSource()

Application Code

To trip the PWMs, you can invert the PWMXBAR state using XBAR_invertOutputSignal().

Advisory	***ePWM: ESM Source for Trip Zone is not Supported***
Revisions Affected	0, A, B, C
Details	An ESM_GEN_EVENT coming from the ESM may not properly trip the ePWM, and should not be used. The following path should not be used to trip PWM: ESM Subsystem (SYS ESM) → ESM_GEN_EVENT → PWM XBAR → ePWM Digital Compare → ePWM Trip Zone
Workaround	Use the error source to generate an NMI interrupt from the respective ESM CPU instance. Within the NMI interrupt, issue a software write using the Driverlib function below to force a trip on the PWM output. EPWM_forceTripZoneEvent()

Advisory	***ePWM: ePWM One-Shot/CBC Trip Event DCxEVTy.force Does Not Set the Trip Condition***
Revisions Affected	0, A, B, C
Details	The DCxEVTy.force signal can be used as input for one-shot trip events and cycle-by-cycle trip events. Setting DCxCTL[EVT1/2FRCSYNCSEL] = 1 (passes the DCxEVT signal through the async path) does not properly set the one-shot or cycle-by-cycle trip event condition, and the trip event is missed.
Workaround	For devices with DCxCTL.EVT1/2FRCSYNCSEL and DCxCTL.EVT1/2LATSEL, there are two options. Otherwise, Sync Path is the only option available for use in one-shot or cycle-by-cycle trip events. **Async Path** (passes the DCxEVT signal through the async path) The following configurations must be made to latch the one-shot or cycle-by-cycle trip condition through the async path: 1. DCxCTL.EVT1/2FRCSYNCSEL = 1 2. DCxCTL.EVT1/2LATSEL = 1 **Sync Path** (passes the DCxEVT signal through the sync path) 1. DCxCTL.EVT1/2FRCSYNCSEL = 0

Advisory ***ePWM: For ePWMs Using Global Load in One-Shot Load Mode, Global Load of Registers may get Delayed When a Write to the GLDCTL2.OSHTLD Register Occurs Within 3 TBCLKs of the Global Load Event***

Revisions Affected

0, A, B, C

Details

When a write to the GLDCTL2.OSHTLD register bit occurs within 3 TBCLKs of the global load event (configured by GLDCTL.GLDMODE), the global load of registers (configured by GLDCFG) may get delayed, which can create unintended waveforms.

Workaround

To avoid this issue, any writes to the GLDCTL2.OSHTLD register bit needs to be a minimum of 3 TBCLK cycles before the configured global load event.

Workaround 1 (Recommended): Synchronous ISR to PWMs

1. Generate an ISR on a known Event Trigger Interrupt event (that is, CMPC, CMPD event, and so forth).
2. Ensure a minimum of 3 TBCTR cycles are kept between the global load event and when the ISR writes to GLDCTL2.OSHTLD.
   a. When writing to the OSHTLD bit, ensure this is non-interruptible code by disabling the interrupts.

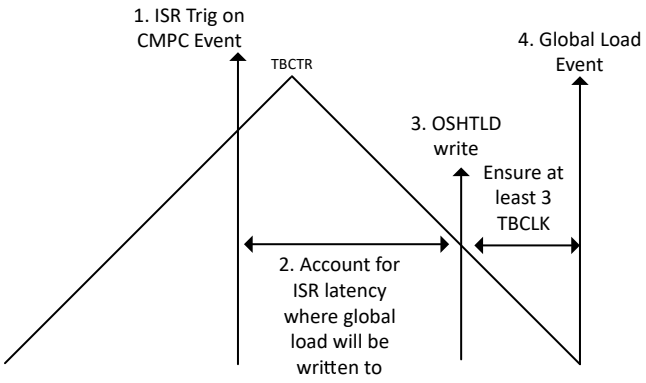


Figure 3-3. Example of Synchronous ISR Loading Method

Workaround 2: Asynchronous updates to PWM registers

1. Read from the TBCTR of the PWM instance that is enabled for global loading before writing to the GLDCTL2.OSHTLD register.
2. Ensure at least 3 cycles are kept between the global load event and writes to GLDCTL2.OSHTLD.
   a. Ensure the code reading from TBCTR and writing to the OSHTLD bit is non-interruptible code by disabling interrupts.

Advisory	***HRPWM: HRPWM High-Resolution Period Shadow to Active Loading Occurs Every ZERO Event, Even if Shadow to Active Load for Period is Set to Only Load on SYNC***
Revisions Affected	0, A, B, C
Details	When high-resolution period is enabled (HRPCTL[HRPE] = 1), and shadow-to-active loading is configured to load on sync event (TBCTL2[PRDLDSYNC] = 0x1 or 0x2), the load occurs every ZERO event instead of on the sync event.
Workaround	When using high-resolution period, the sync event is ignored, and shadow-to-active loading occurs every ZERO event. If high-resolution period is disabled, the sync event is not ignored, and shadow-to-active loading occurs on a sync event.

Advisory	***LIN: LIN Unable to Wake Up Using 0xF0 Wake-Up Key***
Revisions Affected	0, A, B, C
Details	This issue only applies when using the LIN as commander. If the LIN attempts to send 0xF0 as a wake-up key, the internal state machine becomes stuck and will not recover, blocking future transmissions.
Workaround	Send an unused LIN header instead of the 0xF0 wake-up signal. The reserved identifier 0x3E or any other unused identifier can be used. The break field within the header acts as a valid LIN wake-up command to other LIN nodes and disables the internal LIN POWERDOWN bit.

Advisory	***MCD: Missing Clock Detect Should be Disabled When the PLL is Enabled (PLLCLKEN = 1)***
Revisions Affected	0, A, B, C
Details	The PLL has a limp mode feature to provide a slow PLLRAWCLK output even if its input OSCCLK is absent. Independently, the Missing Clock Detect (MCD) circuit will forcibly switch the system clock source to INTOSC1 when a missing OSCCLK input is detected. The MCD mux to switch between these system clock sources is not ensured to be glitch-free when both clock sources (PLLRAWCLK and INTOSC1) are still active. In rare cases, this may lead to unpredictable device behavior during a missing clock failure event.
Workarounds	When the PLL is used by the system (PLLCLKEN = 1), disable the MCD by writing MCDCR.MCLKOFF = 1. The Dual Clock Comparator (DCC) circuit can be configured to quickly detect if the SYSCLK frequency drops outside the desired frequency to its limp mode due to a missing clock event. When the system is operating in PLL bypass mode (PLLCLKEN = 0), the MCD circuit can still be used to detect missing clock events and switch the clock source to INTOSC1.

Advisory	***MEMSS: Data Line Buffer (DLB) for RAM Causes Data Coherency Issue***
Revisions Affected	0, A, B, C
Details	When the Data Line Buffer (DLB) is enabled (by default) and two CPUs perform simultaneous read/write operations to the same RAM address location, the read operation may receive stale data instead of new data in certain conditions.
Workaround	The user should disable the DLB using the configuration bit in the MEM_DLB_CONFIG register if the RAM block is shared between multiple CPUs.

Advisory	***SDFM: Dynamically Changing Threshold Settings (LLT, HLT), Filter Type, or COSR Settings Will Trigger Spurious Comparator Events***
Revisions Affected	0, A, B, C
Details	When SDFM comparator settings—such as filter type, lower/upper threshold, or comparator OSR (COSR) settings—are dynamically changed during run time, spurious comparator events will be triggered. The spurious comparator event will trigger a corresponding CPU interrupt, ePWM X-BAR events, and GPIO output X-BAR events if configured appropriately.
Workaround	When comparator settings need to be changed dynamically, follow the procedure below to ensure spurious comparator events do not generate a CPU interrupt or X-BAR events (ePWM X-BAR/GPIO output X-BAR events): 1. Disable the comparator filter. 2. Delay for at least a latency of the comparator filter + 3 SD-Cx clock cycles. 3. Change comparator filter settings such as filter type, COSR, or lower/upper threshold. 4. Delay for at least a latency of the comparator filter + 5 SD-Cx clock cycles. 5. Enable the comparator filter.

Advisory	***SDFM: Dynamically Changing Data Filter Settings (Such as Filter Type or DOSR) Will Trigger Spurious Data Acknowledge Events***
Revisions Affected	0, A, B, C
Details	When SDFM data settings—such as filter type or DOSR settings—are dynamically changed during run time, spurious data-filter-ready events will be triggered. The spurious data-ready event will trigger a corresponding CPU interrupt and DMA trigger if configured appropriately.
Workaround	When SDFM data filter settings need to be changed dynamically, follow the procedure below to ensure spurious data-filter-ready events are not generated: 1. Disable the data filter. 2. Delay for at least a latency of the data filter + 3 SD-Cx clock cycles. 3. Change data filter settings such as filter type and DOSR. 4. Delay for at least a latency of the data filter + 5 SD-Cx clock cycles. 5. Enable the data filter.

Advisory	***SDFM: Two Back-to-Back Writes to SDCPARMx Register Bit Fields CEVT1SEL, CEVT2SEL, and HZEN Within Three SD-Modulator Clock Cycles can Corrupt SDFM State Machine, Resulting in Spurious Comparator Events***
Revisions Affected	0, A, B, C
Details	Back-to-back writes to SDCPARMx register bit fields CEVT1SEL, CEVT2SEL, and HZEN within three SD-modulator clock cycles can potentially corrupt the SDFM state machine, resulting in spurious comparator events, which can potentially trigger CPU interrupts, ePWM XBAR events, and GPIO output X-BAR events if configured appropriately.
Workaround	Avoid back-to-back writes within three SD-modulator clock cycles or have the SDCPARMx register bit fields configured in one register write.

Advisory	System: Device Reset Remains Asserted When VDD Voltage Ramps Before VDDIO

Revisions Affected

0, A, B, C

Details

The device XRSn reset signal can remain in a low (reset asserted) state when the VDD supply voltage is ramped up before or simultaneously with the VDDIO supply. As a result, the device fails to boot.

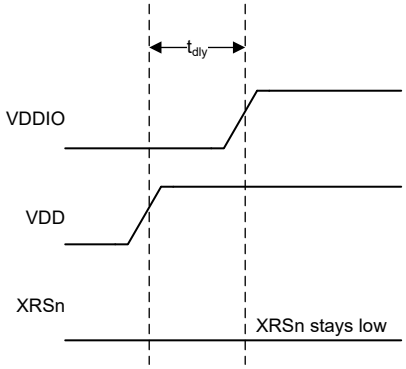


Figure 3-4. Incorrect Power-up Sequence Leading to Stuck Reset

Workaround

Ensure the VDDIO supply is fully ramped up at least 1ms before ramping up the VDD supply voltage.

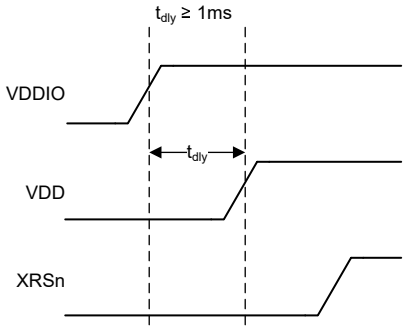


Figure 3-5. Correct Power-up Sequence With Reset Release

Advisory	***System: Issuing device reset (XRSn) can cause unexpected fault if SYSCLKDIVSEL.PLLSYSCLKDIV = 0***
Revisions Affected	0, A, B, C
Details	Issuing device reset (XRSn - internal or external) can put the device into unexpected fault state when SYSCLKDIVSEL.PLLSYSCLKDIV = 0 (divide by 1).
Workaround	Do not use SYSCLKDIVSEL.PLLSYSCLKDIV = 0 (divide by 1). Instead, use SYSCLKDIVSEL.PLLSYSCLKDIV = 1 (divide by 2) or higher.

Advisory	***UART: UART FIFO Gets Cleared on Continuous Debugger Reads***
Revisions Affected	0, A, B, C
Details	The UART IP treats debugger and CPU reads in the same way. As a result, on reading the UART_DR register continuously from CCS, the FIFO gets cleared before the code actually reads it.
Workaround	Do not keep the memory browser open during UART data transfers.

4 Silicon Revision A Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

4.1 Silicon Revision A Usage Notes

Silicon revision-applicable usage notes have been found on a later silicon revision. For more details, see Silicon Revision B Usage Notes.

4.1.1 Security: New TI Keys Programmed on Silicon Revision B Devices

Revisions Affected: 0, A

New FEK (File Encryption Key) values have been programmed on silicon revision B units. Customers are required to change their keys and use OTP KW version and TIFS (TI Foundational Security) package 1.2.1. Silicon revision B units will only work with the version of the packages described.

4.2 Silicon Revision A Advisories

Silicon revision-applicable advisories have been found on a later silicon revision. For more details, see Silicon Revision B Advisories.

Advisory	***Flash: Stand-alone CPU1/CPU3 Reset With Flash Prefetch Enabled may Cause NMI to CPU1/CPU3***
Revisions Affected	0, A
Details	If Flash prefetch is enabled, a stand-alone reset issued to CPU1 or CPU3 (for example, debug reset from CCS) may cause an NMI to the CPU because of an uncorrectable ECC error. Below are the sources for a stand-alone CPU reset. CPU1: 1. Debug reset 2. HSM → CPU1.RSn CPU3: 1. Debug reset 2. HSM → CPU3.RSn 3. CPU3 WD Reset 4. CPU3 NMIWD Reset 5. SSU_CPU3_CFG_REGS → CPU_RST_CTRL.SW_SYSRSN
Workaround	Disable Flash prefetch (FRIx_INTF_CTRL.PREFETCH_EN = 0) before issuing a stand-alone reset to the CPU. CPU1 → FRI1_INTF_CTRL.PREFETCH_EN = 0 CPU3 → FRI3_INTF_CTRL.PREFETCH_EN = 0 FRIx_INTF_CTRL.PREFETCH_EN is writable from CPU1.LINK2 and CPU1/3 debugger if ZONE1 is enabled.

Advisory	***FOTA: Secure FOTA With Encryption does not Work***
Revisions Affected	0, A
Details	A bug in the boot sequence prevents the application of encryption and key derivation on the Firmware-Over-The-Air (FOTA) image.
Workaround	None. This is fixed in silicon revision B.

Advisory	***HSM: HSM ROM Code Does Not Boot HSMRT Image of Size Greater Than 191KB***
Revisions Affected	0, A
Details	Hardware Security Module (HSM) ROM does not boot HSMRT image of size greater than 191KB in Flash Boot Mode.
Workaround	Keep HSMRT image size less than 191KB.

Advisory	***C29 CPU Subsystem: DTHE Interrupts and DMA Events Not Triggered in C29 CPU Subsystem for HS-FS Devices***
Revisions Affected	0, A
Details	In the High-Security, Field-Securable (HS-FS) device life-cycle state, cryptographic engines can be automatically assigned to the C29 CPU subsystem using a boot certificate extension option. When this option is configured in the certificate, the engines are mapped to the C29 CPU, but the corresponding interrupt signals and DMA events are not routed to the C29 CPU, and thus do not trigger when running a C29 application.
Workaround	To detect events generated by the cryptographic accelerator engines, poll the respective interrupt or DMA status register.

Advisory	***System: Pending Misaligned Reads in the Pipeline After CPU Goes to Fault State Preventing NMI Vector Fetch***
Revisions Affected	0, A
Details	The NMI handler fails to execute when three or more back-to-back C29 CPU faults caused by misaligned reads occur. When more than two faults are in the CPU pipeline, the CPU does not fetch the NMI vector as expected.
Workaround	Use ERAD-SEC counter: 1. Choose ESM_GEN_EVENT as input to EPWMXBAR. 2. Configure the ERAD-SEC1 counter in start-stop mode: start event as EPWMXBAR event, stop event as SEC1 event itself. This counter will be counting SYSCLK cycles. 3. Configure the ERAD-SEC reference register to generate a match event and trigger an NMI (INT_EN and NMI_EN bits in SEC_CNTL register) on a count of 50. 4. Configure ESM CPU1 to generate an NMI on the ERAD_CPU1_NMI event.

Advisory	***System: Reset will cause CPU to get stuck in BOOTROM NMI handler***
Revisions Affected	0, A
Details	On the F29P32x device ADCE is not present, but BOOTROM code accesses the ADCE register; thus, the CPU gets an access fault and gets stuck in BOOTROM NMI handler which can cause issues such as a flash programming error.
Workaround	• This issue does not happen on power-up; thus, the user should uncheck the “Reset target before flash programming/operations” option within the Flash Settings of CCS. • Disable WD in application to avoid any device reset. Also, do not issue any reset from the application. • Power cycle the device if the device needs to be reset.

Advisory	***System: Internal device reset could cause the device to be stuck in the reset loop***
Revisions Affected	0, A
Details	On a Rev 0/A device, in case of an internal device reset, which toggles XRSn, the HSM clock divider (HSMCLKDIVSEL. HSMCLKDIV) gets reset to 0 (/1) first and then the rest of the clock logic (like PLL settings) gets reset. This effectively clocks the HSM subsystem with SYSCLK = 200MHz for a couple of cycles which is out of HSM spec frequency (100MHz). This can cause HSM to go into an undefined state and device to be stuck in reset loop due to watchdog reset. This issue is fixed on Rev B and subsequent revisions.
Workarounds	None. Need to power cycle the device in this error condition. Recommendation is to switch application to use die Rev C.

Advisory	***CAN boot mode is not supported in silicon revision 0, A.***
Revisions Affected	0, A
Details	CAN boot mode option is not implemented in bootrom in early silicon revisions 0 and A.
Workaround	None. Use silicon revisions B or later if CAN boot mode is required.

5 Silicon Revision 0 Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

5.1 Silicon Revision 0 Usage Notes

Silicon revision-applicable usage notes have been found on a later silicon revision. For more details, see Silicon Revision B Usage Notes.

5.2 Silicon Revision 0 Advisories

Silicon revision-applicable advisories have been found on a later silicon revision. For more details, see Silicon Revision A Advisories and Silicon Revision B Advisories.

Advisory	***ROM: By Default, GPIO4 is Configured as ERRORSTS by ROM Code and Driven High***
Revisions Affected	0
Details	The ROM code configures a GPIO pin for the ERRORSTS function after the device reset (XRSn) based on SECCFG field settings. By default, the ROM code configures the GPIO4 pin as the ERRORSTS pin. The ERRORSTS (GPIO4) pin will be controlled by the Error Signaling Module (ESM) and will be driven high by default (no error). This high state can cause a board issue if the GPIO4 pin is used to drive critical system functions. For example, if GPIO4 is used for the EPWM3_A function, a high-power FET can be turned on inadvertently and cause damage to the board.
Workaround	1. Avoid using GPIO4 for critical system functions on the board (for example, driving high-power FETs). 2. If GPIO4 must be used for critical system functions, choose a different GPIO for ERRORSTS by appropriately configuring the SECCFG field. Refer to the empty_driverlib_project_secure example in SDK to see how to change the ERRORSTS pinmux option. Since the ROM code configures the GPIO pin for the ERRORSTS function on device reset (XRSn) only, but the ERRORSTS pinmux configuration gets cleared by a debugger reset as well, the user needs to issue a full device reset (XRSn) to rerun the ROM code configuration of ERRORSTS.

Advisory ***VSSOSC: Coupling From Adjacent Pins X1 or X2 may Prevent Proper Boot***

Revisions Affected

0

Details

For silicon revision 0 devices, VSSOSC noise can induce INTOSC2 glitches, which can put the device into a fault state when INTOSC2 is a clock source for the device. INTOSC2 is used as a primary clock to the device during boot, flash programming, and any other time as configured by the application.

Fast edge rates on the adjacent X1 or X2 pins can induce noise on the VSSOSC pin and must be avoided for silicon revision 0 devices. Devices after silicon revision 0 do not have this sensitivity.

Workaround

On silicon revision 0 devices, do not use fast edges on the X1 or X2 pins. For example, do not use a single-ended crystal to drive X1.

Table 5-1. Clock Source Options

APPLICATION CLOCK SOURCE	SILICON REVISION 0	SILICON REVISION A	SILICON REVISION B
X1–X2 crystal	Preferred	Acceptable	Acceptable
X1 single-ended input from external oscillator	Avoid	Acceptable	Acceptable
INTOSC2	Acceptable if the wider INTOSC frequency accuracy is acceptable		

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 F29H85x, F29P58x, and F29P32x devices, see the following documents:

- *F29H85x, F29P58x, and F29P32x Real-Time Microcontrollers* data sheet
- *F29H85x and F29P58x Real-Time Microcontrollers Technical Reference Manual*

7 Trademarks

All trademarks are the property of their respective owners.

8 Revision History

Changes from May 30, 2026 to July 30, 2026 (from Revision D (May 2026) to Revision E (July 2026)) Page

- Added advisory 28

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, 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025