

# CC1352P7 SimpleLink™ Wireless MCU Device Revision B



## ABSTRACT

This document describes the known exceptions to functional specifications (advisories) to the CC1352P7 SimpleLink™ device.

---

## Table of Contents

<b>1 Advisories Matrix</b> .....	<b>2</b>
<b>2 Nomenclature, Package Symbolization, and Revision Identification</b> .....	<b>3</b>
2.1 Device and Development Support-Tool Nomenclature.....	3
2.2 Devices Supported.....	3
2.3 Package Symbolization and Revision Identification.....	3
<b>3 Advisories</b> .....	<b>4</b>
<b>4 Revision History</b> .....	<b>18</b>

## Trademarks

SimpleLink™ and Texas Instruments™ are trademarks of Texas Instruments.  
Arm® and Cortex® are registered trademarks of Arm Limited (or its subsidiaries).  
All trademarks are the property of their respective owners.

## 1 Advisories Matrix

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

**Table 1-1. Advisories Matrix**

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		B
Radio	<a href="#">Advisory Radio_01</a> — Proprietary radio modes: spurious emissions can affect regulatory compliance	Yes
Radio	<a href="#">Advisory Radio_02</a> — High-power PA operation at temperatures below -20°C may affect the 32-kHz crystal oscillator	Yes
Power	<a href="#">Advisory Power_03</a> — Increased voltage ripple at low supply voltages when DC/DC converter is enabled	Yes
PKA	<a href="#">Advisory PKA_01</a> — Public key accelerator (PKA) interrupt line is always high when module is enabled and PKA is idle	Yes
PKA	<a href="#">Advisory PKA_02</a> — Public key accelerator (PKA) RAM is not byte accessible	Yes
I2C	<a href="#">Advisory I2C_01</a> — I <sup>2</sup> C module master status bit is set late	Yes
I2S	<a href="#">Advisory I2S_01</a> — I <sup>2</sup> S bus faults are not reported	Yes
CPU	<a href="#">Advisory CPU_01</a> — Arm® Errata #838869: Store immediate overlapping exception return operation might vector to incorrect interrupt	Yes
CPU	<a href="#">Advisory CPU_02</a> — Arm® Errata #752770: Interrupted loads to SP can cause erroneous behavior	Yes
CPU	<a href="#">Advisory CPU_03</a> — Arm® Errata #776924 VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	Yes
CPU, System	<a href="#">Advisory CPU_Sys_01</a> — The SysTick calibration value (register field CPU_SCS.STCR.TENMS) used to set up 10-ms periodic ticks is incorrect when the system CPU is running off divided down 48-MHz clock	Yes
System	<a href="#">Advisory Sys_01</a> — Device might boot into ROM serial bootloader when waking up from shutdown	Yes
System	<a href="#">Advisory Sys_05</a> — Elevated power-on-reset (POR) threshold voltage at low temperatures	Yes
System Controller	<a href="#">Advisory SYSCTRL_01</a> — Resets occurring in a specific 2-MHz period during initial power up are incorrectly reported	Yes
IO Controller	<a href="#">Advisory IOC_01</a> — Limited number of DIOs available for the bootloader backdoor	Yes
ADC	<a href="#">Advisory ADC_01</a> — Periodic ADC trigger at 200 kHz rate can be ignored when XOSC_HF is turned on or off	Yes
ADC	<a href="#">Advisory ADC_02</a> — ADC samples can be delayed by 2 or 14 clock cycles (24 MHz) when XOSC_HF is turned on or off, resulting in sample jitter	Yes
ADC	<a href="#">Advisory ADC_03</a> — Software can hang when reading the ADC FIFO if a single manual ADC trigger is generated immediately after the ADC is enabled	Yes

## 2 Nomenclature, Package Symbolization, and Revision Identification

### 2.1 Device and Development Support-Tool Nomenclature

To designate the stages in the product development cycle, Texas Instruments™ assigns prefixes to the part numbers of all devices and support tools. Each device has one of three prefixes: X, P, or null (for example, XCC1352P7). 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 (X/TMDX) through fully qualified production devices/tools (null/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.

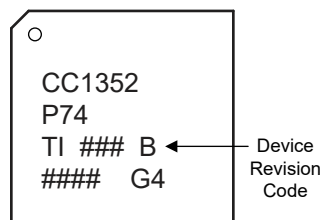
### 2.2 Devices Supported

This document supports the following device:

- [CC1352P7](#)

### 2.3 Package Symbolization and Revision Identification

Figure 2-1 and Table 2-1 describe package symbolization and the device revision code.



**Figure 2-1. Package Symbolization**

**Table 2-1. Revision Identification**

Device Revision Code	Silicon Revision
B	PG1.1

### 3 Advisories

#### Radio\_01 *Proprietary radio modes: spurious emissions can affect regulatory compliance*

---

**Revisions Affected:** Revision B

**Details:** When device internal load capacitors are used with the external 48-MHz crystal, energy couples from the crystal oscillator circuit to the RF output. This coupling causes spurious emissions at  $N \times 48$  MHz from carrier frequency. This includes, but is not limited to, the frequency bands supported by the device covered by the following regulations:

**When using the +14-dBm RF power amplifier**

- ARIB T-108 (Japan)

**When using the +20-dBm RF power amplifier**

- 470- to 510-MHz (China) Band
- FCC CFR47 Part 15 (US)
- ETSI EN 300 220 (Europe)
- ETSI EN 300 328 (Europe)
- ETSI EN 300 440 (Europe)

**Workaround:** For compliance with affected standards, external load capacitors might be needed for the 48-MHz crystal to reduce spurious emissions. Internal capacitors (default 7-pF connected capacitance) must then be disconnected internally.

This workaround is implemented by defining the following symbols in the included customer configuration file (ccfg.c) available in the SimpleLink™ CC13xx and CC26xx Software Development Kit (SDK) examples:

```
#define SET_CFG_MODE_CONF_XOSC_CAPARRAY_DELTA -128
#define SET_CFG_MODE_CONF_XOSC_CAP_MOD 0
```

#### Radio\_02 *High-Power PA Operation at Temperatures Below -20°C May Affect the 32-kHz Crystal Oscillator*

---

**Revisions Affected:** Revision B

**Details:** When using the high-power PA at temperatures below -20°C and high output power, the PA may affect the 32-kHz crystal oscillator due to RF load impedance mismatch. In this situation the crystal oscillator will stop, and provided the clock loss detector is enabled (OSC\_DIG:CTL0.CLK\_LOSS\_EN = 1), the device will reset. Antenna impedances outside of VSWR of 2:1 must be avoided in all operating scenarios.

**Workaround:**

- For applications operating below -20°C, it is of particular importance to accurately follow the reference design for the RF balun and -matching network with respect to component values and layout. Amplitude- and phase balance through the balun must be <1dB and <6 degrees, respectively.
- For PCB designs not adhering to the TI recommended guidelines or in applications where antenna impedance is subject to change due to external factors, output power should be limited to 17 dBm maximum.
- Make sure the clock loss detector is enabled, OSC\_DIG:CTL0.CLK\_LOSS\_EN = 1, to properly reset the device should the crystal oscillator be stopped.

#### Power\_03 *Increased voltage ripple at low supply voltages when DC/DC converter is enabled*

---

**Revisions Affected:** Revision B

**Power\_03**  
(continued)

***Increased voltage ripple at low supply voltages when DC/DC converter is enabled***

---

**Details:**

At supply voltages <2.0V, a hardware control module disables the DC/DC converter to maximize system efficiency. This module does not have enough hysteresis, causing approx 10 mV of ripple on the VDDR regulated power supply. Based on internal testing of the device, it is not anticipated that this erratum affects RF performance. However, these test results cannot ensure that a customer's application or end equipment will not be affected.

**Workaround:**

Use the TI-provided Power driver (PowerCC26X2.c) which automatically disables the DC/DC converter when supply voltage is <2.25V.

The workaround is available in the SimpleLink™ CC13xx and CC26xx Software Development Kit (SDK).

---

**PKA\_01**      ***Public key accelerator (PKA) interrupt line is always high when module is enabled and PKA is idle***

---

**Revisions Affected:** Revision B

**Details:** When the PKA module is enabled and idle, the interrupt line is always high and the interrupt can thus not be used as is.

**Workaround:** The workaround is to disable the PKA interrupt in the interrupt service routine while the PKA module is idle and re-enable the interrupt right after starting an operation.

The workaround is implemented in the TI-provided cryptography drivers (ECDHCC26X2.c, ECDSACC26X2.c, ECJPAKECC26X2.c\_list.c) in the SimpleLink™ CC13xx and CC26xx software development kit (SDK).

---

**PKA\_02**      ***Public key accelerator (PKA) RAM is not byte accessible***

---

**Revisions Affected:** Revision B

**Details:** When accessing the PKA RAM, the RAM is not byte accessible. If a single byte is accessed (read or written), 4 bytes will be accessed instead.

**Workaround:** The workaround is to use word access (4 bytes) when accessing the PKA RAM.

The workaround is implemented in the TI-provided cryptography drivers (ECDHCC26X2.c, ECDSACC26X2.c, ECJPAKECC26X2.c\_list.c) in the SimpleLink™ CC13xx and CC26xx software development kit (SDK).

---

**I2C\_01** *I<sup>2</sup>C module master status bit is set late*

---

**Revisions Affected:** Revision B

**Details:** The I2C.MSTAT[0] bit is not set immediately after writing to the I2C.MCTRL register. This can lead an I<sup>2</sup>C master to believe it is no longer busy and continuing to write data.

**Workaround:** Add four NOPs between writing to the MCTRL register and polling the MSTAT register.

The workaround is implemented in the TI-provided I2C Master driver (I2CCC26XX.c) and in the I2C driver Library APIs (driverlib/i2c.c).

The workaround is available in the SimpleLink™ CC13xx and CC26xx Software Development Kit (SDK).

---

**I2S\_01** *I<sup>2</sup>S bus faults are not reported*

---

**Revisions Affected:** Revision B

**Details:** The I<sup>2</sup>S module will not set the bus error interrupt flag (I2S0.IRQFLAGS.BUS\_ERR) if an I<sup>2</sup>S read or write causes a system bus fault that results from access to illegal addresses (usage error).

**Workaround:** Software must ensure that memory area used by the I<sup>2</sup>S DMA is accessible, meaning that the memory is powered on and the system bus is connected.

As an example; The TI-provided SPI driver SPICC26X2DMA.c will ensure that the flash memory is kept accessible also in Idle power mode if the transmit buffer address starts with 0x0 to ensure no bus faults occur. A similar approach needs to be taken if writing a peripheral driver utilizing I2S.

---

**CPU\_01**      ***Arm® Errata #838869: Store immediate overlapping exception return operation might vector to incorrect interrupt***


---

**Revisions Affected:** Revision B

**Details:**      **Configurations Affected:**

This erratum only affects systems where writeable memory locations can exhibit more than one wait state (system SRAM does not have wait states).

The Arm® Cortex®-M4 processor includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

**Conditions:**

- The handler for interrupt A is being executed.
- Interrupt B, of the same or lower priority than interrupt A, is pending.
- A store with immediate offset instruction is executed to a bufferable location:
 

```
STR/STRH/STRB <Rt>, [<Rn>,#imm]
STR/STRH/STRB <Rt>, [<Rn>,#imm]!
STR/STRH/STRB <Rt>, [<Rn>,#imm]
```
- Any number of additional data-processing instructions can be executed.
- A BX instruction is executed that causes an exception return.
- The store data has wait states applied to it such that the data is accepted at least two cycles after the BX is executed.
  - Minimally this is two cycles if the store and the BX instruction have no additional instructions between them.
  - The number of wait states required to observe this erratum needs to be increased by the number of cycles between the store and the interrupt service routine exit instruction.
- Before the bus accepts the buffered store data, another interrupt C is asserted which has the same or lower priority as A, but a greater priority than B.

**Implications:**

The processor should execute interrupt handler C, and on completion of handler C the processor should execute the handler for B. If the previously listed conditions are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C, and then twice executing the handler for B. The first time the handler for B is executed it will be at the priority level for interrupt C. If interrupt C is pending by a level-based interrupt that is cleared by C's handler then interrupt C will be pending again after the handler for B has completed and the handler for C will be executed. If interrupt C is level based, then this interrupt will eventually become re-pending and subsequently be handled. If interrupt C is a single pulse interrupt, there is a possibility that this interrupt will be lost.



**CPU\_01** (continued) **Arm® Errata #838869: Store immediate overlapping exception return operation might vector to incorrect interrupt**

This bug is triggered in a rare condition. In cases where STORE experiences more than 2 wait cycles, workarounds must be used by the software developer.

- This erratum does not apply for TI-RTOS interrupts, which ensures that no *store with immediate offset* occurs within the last 5 instructions of the interrupt routine. See the following files included in all SDKs for further implementation details:
  - kernel/tirtos/packages/ti/sysbios/family/arm/m3/Hwi\_asm\*.sv7M
- Zero-latency interrupts in TI-RTOS (bypassing the kernel) and the no-RTOS examples in the SDK are affected by this erratum.

**Workarounds: Software not using the Memory Protection Unit (MPU):**

For software not using the Memory Protection Unit (MPU), the workaround can be to disable CPU write buffering (register CPU\_SCS.ACTLR.DISDEFWBUF) at the cost of significantly reduced execution speed.

**All other cases (recommended workaround):**

Ensure a DSB instruction occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

**ARMCC:**

```
...
__schedule_barrier(); __asm{DSB}; __schedule_barrier(); }
```

**GCC:**

```
...
__asm volatile ("dsb 0xf" ::: "memory"); }
```

---

**Note**

The workaround for this bug will **not** be added automatically by the compiler.

---

---

**CPU\_02**      **Arm® Errata #752770: Interrupted loads to SP can cause erroneous behavior**


---

**Revisions Affected:** Revision B

**Details:** An interrupt occurring during the data-phase of a single word load to the stack-pointer (SP/R13) can cause an erroneous behavior of the device. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions that can result in the load transaction being repeated are:

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!
- LDR SP,[Rn,#imm]
- LDR SP,[Rn]
- LDR SP,[Rn,Rm]

The affected instructions that can result in the stack-pointer being loaded from an incorrect memory address are:

- LDR SP,[Rn],#imm
- LDR SP,[Rn,#imm]!

**Conditions:**

- An LDR is executed, with SP/R13 as the destination.
- The address for the LDR is successfully issued to the memory system.
- An interrupt is taken before the data has been returned and written to the stack-pointer.

**Implications:**

Unless the load is being performed to device memory or strongly-ordered memory, there should be no implications from the repetition of the load.

- In the unlikely event that the load is being performed to device memory or strongly-ordered memory, the repeated read can result in the final stack-pointer value being different than had only a single load been performed.
- Interruption of the two write-back forms of the instruction can result in both the base register value and the final stack-pointer value being incorrect. This can result in apparent stack corruption and subsequent unintended modification of memory.

**Workaround:** Most compilers ensure this bug is not triggered by not emitting the affected instruction sequence and not using the instructions in the compiler runtime libraries. This includes:

- IAR from v6.21
- All versions of TI's Arm compiler (CCS)

A workaround for both issues can be implemented by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

If repeated reads are acceptable, then the base register update issue may be worked around by performing the stack-pointer load without the base increment followed by a subsequent ADD or SUB instruction to perform the appropriate update to the base register.

## CPU\_03

### ***Arm® Errata #776924: VDIV or VSQRT instructions might not complete correctly when very short ISRs are used***

**Revisions Affected:** Revision B

**Details:**

On the Arm® Cortex®-M4F processor, the VDIV and VSQRT instructions take 14 cycles to execute. When an interrupt is taken a VDIV or VSQRT instruction is not terminated, and completes its execution while the interrupt stacking occurs. If lazy context save of floating point state is enabled then the automatic stacking of the floating point context does not occur until a floating point instruction is executed inside the interrupt service routine.

Lazy context save is enabled by default. When it is enabled, the minimum time for the first instruction in the interrupt service routine to start executing is 12 cycles. In certain timing conditions, and if there is only one or two instructions inside the interrupt service routine, then the VDIV or VSQRT instruction might not write its result to the register bank or to the FPSCR.

**Conditions::**

- The floating point unit is present and enabled
- Lazy context saving is not disabled
- A VDIV or VSQRT is executed
- The destination register for the VDIV or VSQRT is one of s0 - s15
- An interrupt occurs and is taken.
- The interrupt service routine being executed does not contain a floating point instruction.
- An interrupt return is executed 14 cycles after the VDIV or VSQRT is executed.

A minimum of 12 of these 14 cycles are utilized for the context state stacking, which leaves 2 cycles for instructions inside the interrupt service routine, or 2 wait states applied to the entire stacking sequence (which means that it is not a constant wait state for every access).

In general this means that if the memory system inserts wait states for stack transactions then this erratum cannot be observed.

**Implications:**

The VDIV or VSQRT instruction does not complete correctly and the register bank and FPSCR are not updated, meaning that these registers hold incorrect, out of date, data.

For hand-written assembly code inside interrupt routines, this erratum should be considered.

**Workarounds:**

A workaround is only required if the floating point unit is present and enabled. A workaround is not required if the memory system inserts one or more wait states to every stack transaction.

When using TI-RTOS interrupts, all interrupt service routines will contain more than the 2 instructions and no workaround is required.

In all other cases, one of the following two workarounds must be implemented:

**Workaround 1:** Disable lazy context save of floating point state by clearing LSPEN to 0 (bit 30 of the FPCCR at address 0xE000EF34).

**Workaround 2:** Ensure that every interrupt service routine contains more than 2 instructions in addition to the exception return instruction.

**CPU\_Sys\_01**      *The SysTick calibration value (register field CPU\_SCS.STCR.TENMS) used to set up 10-ms periodic ticks is incorrect when the system CPU is running off divided down 48-MHz clock*

---

**Revisions Affected:** Revision B

**Details:** When using the Arm® Cortex® SysTick timer, the TENMS register field (CPU\_SCS.STCR.TENMS) will always shows the value corresponding to a 48-MHz CPU clock, regardless of the CPU division factor.

**Workarounds:** One of the following two workarounds must be implemented:

**Workaround 1:** Do not use a divided down system CPU clock. In general, power savings are maximized by completing a task at full clock speed and then stopping the system CPU entirely after the task is complete.

**Workaround 2:** Read the system CPU division factor from the PRCM.CPUCLKDIV.RATIO register and compensate the TENMS field in software based on this value.

TI-provided drivers do not offer any functionality to divide the system CPU clock.

---

**Sys\_01**                      ***Device might boot into ROM serial bootloader when waking up from shutdown***

---

**Revisions Affected:** Revision B

**Details:**

For the conditions given below, the device will boot into and execute the ROM serial bootloader when waking up from Shutdown power mode. Intended behavior is to execute the application image. The prerequisites for this erratum to happen are:

- The wake up from Shutdown must be caused by toggling or noise on the JTAG TCK pin and not by a GPIO event.
- The Customer Configuration Section (CCFG) must have configured the bootloader with the following field values:
  - BOOTLOADER\_ENABLE = 0xC5 (Bootloader enabled)
  - BL\_ENABLE = 0xC5 (Bootloader pin backdoor enabled)
  - BL\_PIN\_NUMBER = n (any valid DIO number)

With the above prerequisites, the bootloader will be entered in the following cases:

- The CCFG bootloader pin level (BL\_LEVEL) is set to 0x0 (active low) AND the input buffer enable for the DIO defined in BL\_PIN\_NUMBER is disabled in register IOC.IOCFGn.IE. If the input buffer is not enabled, the DIO level will always read 0 and bootloader will be entered.
- The input buffer controlled by IOC.IOCFGn.IE is enabled and the DIO input value is the same level as the CCFG bootloader pin level (BL\_LEVEL) when entering Shutdown (GPIO input values are latched when entering Shutdown).

Please refer to the ICEMelter chapter in the [CC13x2x7, CC26x2x7 SimpleLink™ Wireless MCU Technical Reference Manual](#) for details on how noise entering the JTAG TCK pin can wake up the device.

**Workarounds:**

One of the following workarounds must be implemented:

- If input buffer is not enabled, use only active high bootloader pin level (BL\_LEVEL)
- If input buffer is enabled, ensure DIO input pin level is not the same as bootloader pin level (BL\_LEVEL) when entering Shutdown.

<b>Sys_05</b>	<b><i>Elevated power-on-reset (POR) threshold voltage at low temperatures</i></b>
<b>Revisions Affected</b>	Revision B
<b>Details</b>	<p>When powering up the device from 0V at temperatures &lt; 0°C, the power-on-reset (POR) circuit may not release reset until VDDS reaches 2.3V, and not at 1.8V as intended. After POR has released the reset, an affected device will continue to operate at voltages down to 1.8V.</p> <p>This behavior is only observed during power up and does not occur when the device is subjected to an external pin reset, wake-up from shutdown, or watchdog reset.</p> <p>The occurrence is rare and is only observed on very few devices.</p>
<b>Workaround</b>	<p><b>Workaround 1:</b> Power-up the devices at VDDS &gt; 2.3V when operating at temperatures below 0°C.</p> <p>or</p> <p><b>Workaround 2:</b> Power-up the device at VDDS &lt; 2.3V, trigger the Reset-pin from a host MCU or external circuitry.</p> <p>In addition, when operating the device in external regulator mode, workaround 2 must be implemented. Please note that VDDS must not exceed 1.95V in external regulator mode. <u><i>This is applicable only to devices that have external regulator mode support.</i></u></p>
<b>SYCTRL_01</b>	<b><i>Resets occurring in a specific 2-MHz period during initial power up are incorrectly reported</i></b>
<b>Revisions Affected:</b>	Revision B
<b>Details:</b>	<p>If a reset occurs in a specific 2-MHz period during initial power-up (boot), the reset source in AON_PMCTL.RESETCTL.RESET_SRC is reported as PWR_ON regardless of the reset source. This means that there is a window of 0.5 μs during boot where a reset can be incorrectly reported.</p>
<b>Workaround:</b>	None

---

**IOC\_01** *Limited number of DIOs available for the bootloader backdoor*

---

**Revisions Affected:** Revision B

**Details:** The highest possible DIO number that can be used for the bootloader backdoor is limited to the number of available GPIOs minus 1. The bootloader backdoor pin is configured through SET\_CCFG\_BL\_CONFIG\_BL\_PIN\_NUMBER in ccfg.c. That means that if the device has x GPIOs, the highest DIO number that can be selected for the bootloader backdoor is DIO<sub>x-1</sub>, even if higher DIO numbers are available for the device.

**Workarounds:** There are no workaround for this issue.

## ADC\_01 *Periodic ADC trigger at 200 kHz rate can be ignored when XOSC\_HF is turned on or off*

**Revisions Affected:** Revision B

**Details:** There is no dedicated clock source selection for the ADC clock. The clock is derived from either XOSC\_HF or RCOSC\_HF, but defaults to XOSC\_HF-derived clock whenever this is turned on.

When the ADC clock source is switched from RCOSC\_HF to XOSC\_HF-derived clock, the clock will stop for 2 cycles (24 MHz).

When the ADC clock source is switched from XOSC\_HF-derived clock to RCOSC\_HF-derived clock, the clock will stop for additionally 12 clock cycles, as the RCOSC\_HF-derived clock is not ready when switch is done.

The fact that the clock is stopped, together with the difference in frequency between XOSC\_HF and RCOSC\_HF, may cause the ADC sampling and conversion to finish too late to catch the next trigger.

**Workaround 1:** Use asynchronous sampling.

The sampling period after the issue occurs can be reduced by up to 20% (12 + 1 clock cycles at 24 MHz)

To use the ADC in asynchronous mode from the Sensor Controller:

Call `adcEnableAsync()` to enable the ADC, instead of `adcEnableSync()`

Example:

```
adcEnableAsync(ADC_REF_FIXED, ADC_TRIGGER_AUX_TIMER0);
```

To use the ADC in asynchronous mode from the System CPU, by using the ADCBuf driver:

```
ADCBuf_Params params;
ADCBufCC26X2_ParamsExtension paramsExtension;

ADCBuf_Params_init(&params);
ADCBufCC26X2_ParamsExtension_init(&paramsExtension);

paramsExtension.samplingMode = ADCBufCC26X2_SAMPING_MODE_ASYNCHRONOUS;
params.custom = &paramsExtension;
```

To use the ADC in asynchronous mode from the System CPU, by using DriverLib API:

Call `AUXADCEnableAsync()` to enable the ADC, instead of `AUXADCEnableSync()`

Example:

```
AUXADCEnableAsync(AUXADC_REF_FIXED, AUXADC_TRIGGER_GPT0A);
```

Please note the difference between the asynchronous and synchronous ADC modes:

- In asynchronous mode, the ADC trigger ends the sampling period (which started immediately after the previous conversion), and starts conversion.
- In synchronous mode, the ADC trigger starts the sampling period (with configurable duration), followed by conversion.

**Workaround 2:** Ensure that XOSC\_HF is not turned on or off while the ADC is used.

**Workaround 3:** Increase the sampling period by (12+1)/24  $\mu$ s or more.



---

**ADC\_02**      ***ADC samples can be delayed by 2 or 14 clock cycles (24 MHz) when XOSC\_HF is turned on or off, resulting in sample jitter***


---

**Revisions Affected:** Revision B

**Details:** There is no dedicated clock source selection for the ADC clock. The clock is derived from either XOSC\_HF or RCOSC\_HF, but defaults to XOSC\_HF-derived clock whenever this is turned on.

When the ADC clock source is switched from RCOSC\_HF to XOSC\_HF-derived clock, the clock will stop for 2 cycles (24 MHz).

When the ADC clock source is switched from XOSC\_HF-derived clock to RCOSC\_HF-derived clock, the clock will stop for additionally 12 clock cycles, as the RCOSC\_HF-derived clock is not ready when switch is done.

SCLK\_HF switches from RCOSC\_HF to XOSC\_HF at different times compared to ADC clock. This leads to sample jitter.

**Workaround 1:** Use asynchronous sampling

- This will reduce the delay of 14 clock cycles down to 2 clock cycles.
- Using asynchronous sampling and an external trigger source (GPIO input pin) will eliminate the delay completely

To use the ADC in asynchronous mode from the Sensor Controller:

Call `adcEnableAsync()` to enable the ADC, instead of `adcEnableSync()`

Example:

```
adcEnableAsync(ADC_REF_FIXED, ADC_TRIGGER_AUX_TIMER0);
```

To use the ADC in asynchronous mode from the System CPU, by using the ADCBuf driver:

```
ADCBuf_Params params;
ADCBufCC26X2_ParamsExtension paramsExtension;

ADCBuf_Params_init(&params);
ADCBufCC26X2_ParamsExtension_init(&paramsExtension);

paramsExtension.samplingMode = ADCBufCC26X2_SAMPING_MODE_ASYNCHRONOUS;
params.custom = &paramsExtension;
```

To use the ADC in asynchronous mode from the System CPU, by using DriverLib API:

Call `AUXADCEnableAsync()` to enable the ADC, instead of `AUXADCEnableSync()`

Example:

```
AUXADCEnableAsync(AUXADC_REF_FIXED, AUXADC_TRIGGER_GPT0A);
```

Please note the difference between the asynchronous and synchronous ADC modes:

- In asynchronous mode, the ADC trigger ends the sampling period (which started immediately after the previous conversion), and starts conversion.
- In synchronous mode, the ADC trigger starts the sampling period (with configurable duration), followed by conversion

**Workaround 2:** Ensure that XOSC\_HF is not turned on or off while the ADC is used.

---

**ADC\_03**                      **Software can hang when reading the ADC FIFO if a single manual ADC trigger is generated immediately after the ADC is enabled**


---

**Revisions Affected:** Revision B

**Details:**                      There is no dedicated clock source selection for the ADC clock. The clock is derived from either XOSC\_HF or RCOSC\_HF, but defaults to XOSC\_HF-derived clock whenever this is turned on.

When the ADC clock source is switched from RCOSC\_HF to XOSC\_HF-derived clock, the clock will stop for 2 cycles (24 MHz).

When the ADC clock source is switched from XOSC\_HF-derived clock to RCOSC\_HF-derived clock, the clock will stop for additionally 12 clock cycles, as the RCOSC\_HF-derived clock is not ready when switch is done.

The additional 12 clock cycles introduces a race between trigger-event and ADC trigger-detector to get out of reset.

**Workaround 1:**              TI software adds a short delay at the end of the function that enables the ADC.

- If using the ADC through the System CPU (TI drivers or DriverLib API): Use SimpleLink CC13x2 and CC26x2 SDK 5.30 or later.
- If using ADC through the Sensor Controller (ADC resource): Use Sensor Controller Studio 2.8.0 or later.

**Workaround 2:**              Ensure that XOSC\_HF is not turned on or off while the ADC is used.

#### 4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

**Changes from May 26, 2021 to January 18, 2022 (from Revision \* (May 2021) to Revision A (January 2022))**

	<b>Page</b>
• Removed Advisory GPTM_01 and Advisory SRAM_01.....	<a href="#">2</a>

---

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

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

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2022, Texas Instruments Incorporated