Errata

CC1350 SimpleLink™ Wireless MCU Silicon Errata

ABSTRACT

This document describes the known exceptions to functional specifications (advisories) for the CC1350 SimpleLink™ device.

Table of Contents

Trademarks

1 Advisory Matrix

Table 1-1 lists all advisories and the modules affected.

Table 1-1. Advisories Matrix

MODULE	DESCRIPTION
System level	Advisory 01, RF Core CPU May Hang When Running BLE Master Command With High Throughput From Slave to Master
Digital PLL	Advisory 02, Synthesizer Calibration Might Fail on Rare Occasions
System level	Advisory 03, Reading From Flash While Performing Clock Switching Between the High-Speed Oscillators (XOSC_HF and RCOSC_HF) Will Cause the System to Hang
SSI	Advisory 04, Slave Mode Can Sample New TX Data From SYSBUS Clock Domain Using SSPCLK With No Synchronization
SSI	Advisory 05, Motorola® SPI Format Slave Mode Writes to Transmit FIFO Can Lose Data
Sensor Controller	Advisory 06, Insufficient Power Supply Recharging When Using the Sensor Controller Might Cause the System to Hang or Force a Pin Reset
PRCM	Advisory 07, Wrong Reset Source Indication
Receiver	Advisory 08, Temporary Loss of Receive Function During Continuous Receive Operation Over Long Time
Receiver	Advisory 09, Receive Sensitivity Degradation in Continuous Receive Operation Over Long Time
System level	Advisory 10, Late Time-out of Receiver Operation in Sniff Mode
System level	Advisory 11, Radio Frequency Error Glitch When Device Switches From IDLE to ACTIVE Mode While Radio is in TX or RX
System level	Advisory 12, Slow Transition Across Brown-Out Detect (BOD) Threshold Might Cause the Device to Hang
Digital IOs	Advisory 13, Limited Number of DIOs Available for the Bootloader Backdoor
General-Purpose Timer	Advisory 14, An Incorrect Value Might Be Written to the General-Purpose (GP) Timers MMRs (Memory Mapped Registers) When Simultaneously Accessing the AES (Advanced Encryption Standard) Engine from a Different Master
Radio	Advisory 15, CMD_TX_TEST May End Before Transmitting
ADC	Advisory 16, Periodic ADC Trigger at 200 kHz Rate Can Be Ignored When XOSC_HF Is Turned On or Off
ADC	Advisory 17, ADC Samples Can Be Delayed by 2 Clock Cycles (24 MHz) When XOSC_HF Is Turned On or Off, Resulting in Sample Jitter
ADC	Advisory 18, Writing Any Value to AUX_ANAIF:ADCTRIG.START Will Create an ADC Trigger

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, *CC1350F128RGZR*). 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 CC1350 device.

2.3 Package Symbolization and Revision Identification

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

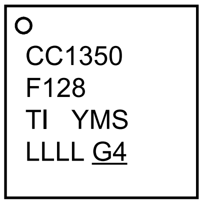

Figure 2-1. Package Symbolization

Table 2-1. Revision Identification

DEVICE REVISION CODE	SILICON REVISION
B	2.1

3 Silicon Advisories

This section lists the advisories for this silicon revision.

Advisory 01 — *RF Core CPU May Hang When Running BLE Master Command With High Throughput From Slave to Master*

Details: If the BLE master command is run, the RF core in the device may hang if all of the following criteria are fulfilled:

- The master transmitted a packet with the header field MD (cf. Bluetooth® Specification Version 4.2, Vol. 6, Part B, Section 4.2) cleared to 0
- The master then received a packet with the header field MD set to 1 and CRC OK
- The received packet is ignored as a retransmission of the last received packet, or the packet is empty (provided that the auto-flush feature is enabled; this is the setting used by the BLE stack)
- All the entries for received packet that were available at the start of the command have been filled with previously received packets, and the entries have not yet been processed and freed for re-use

In this case, the transmitter will not be enabled after receiving the packet, but the firmware defined state machine will still go into transmitter states. This causes registers to be accessed without the module having a clock, meaning that the RF core will hang until the radio is power-cycled using the internal power management on the chip.

This combination of events is unlikely in all cases and only possible when implementing a central device that may receive more packets with payload in a single connection event than the number of receive buffers that had been allocated.

Workaround(s): The user must ensure that the transmitter is enabled prior to the state machine going into transmitter states if all the aforementioned criteria are fulfilled.

Advisory 02 — *Synthesizer Calibration Might Fail on Rare Occasions*

Details: The synthesizer calibration has been observed to fail on rare occasions. This will cause any radio command that requires a calibrated synthesizer to fail.

Workaround(s): SIMPLELINK-CC13X0-SDK_2.20.00.38 and later includes an improved synthesizer start-up sequence that reduces the probability of synthesizer calibration failure.

The workaround for a failed synthesizer calibration is to calibrate again.

TI provided software stacks

For applications that use the software stacks delivered by TI, no action is required because error handling is implemented by the stacks.

Other (customer developed) software stacks

For applications that use TI-RTOS or CC13XXWARE together with customer-developed software stacks, TI recommends that proper run-time error handling is performed when executing a radio command. See the software documentation included in TI-RTOS for Microcontrollers (MCU) documentation for error codes.

Advisory 03 — *Reading From Flash While Performing Clock Switching Between the High-Speed Oscillators (XOSC_HF and RCOSC_HF) Will Cause the System to Hang*

Details: The CC1350 device contains five modules that can read from flash independently of each other. These five modules are:

- Arm® Cortex®-M3
- RF Core
- I2S
- µDMA
- Crypto (AES module)

Clock switching between XOSC_HF and RCOSC_HF can only be initiated by the Cortex-M3. While the Cortex-M3 performs clock switching, no other modules are allowed to read from the flash. The system will hang if any of the other four modules (RF core, I2S, µDMA, or Crypto) are reading from flash during this period.

Workaround(s): The user must make sure that the Cortex-M3 does not perform clock switching while any of the other four modules (RF core, I2S, µDMA, or Crypto) are reading from flash.

SimpleLink SDK (TI-RTOS)

Clock switching from RCOSC_HF to XOSC_HF is done by calling Power_setDependency (XOSC_HF). The user must register a notification to be notified when the clock switching is completed. When the notification function is called by the power driver, the clock switching is completed and it is safe for all modules to read from flash again.

Clock switching from XOSC_HF to RCOSC_HF is done by calling Power_releaseDependency (XOSC_HF). When this function returns, the clock switching is completed and it is safe for all modules to read from flash again. The TI-RTOS radio driver never reads from flash, so it is safe to perform clock switching at all times when using this driver. However, it is up to the user to make sure that no clock switching is performed if the RF Core, I2S, µDMA, or Crypto modules are reading from flash.

No RTOS

Clock switching is performed by calling the driverLib API OSCHfSourceSwitch(). When this function returns, it is safe for all modules to read from flash again. It is up to the user to make sure that no clock switching is performed if the RF core, I2S, µDMA, or Crypto modules are read from flash.

Advisory 04 — *Slave Mode Can Sample New TX Data From SYSBUS Clock Domain Using SSPCLK With No Synchronization*

Details: When the SSI is programmed to operate in slave mode, the data written to the SSI data register (SSIn:DR) in the SYSBUS clock domain can be sampled in the SSPCLK domain or without any synchronization. This sampling condition occurs when all of the following conditions are met:

- The SSI Transmit FIFO is empty.
- The SSI Data register (SSIn:DR) write access occurs as a new SPI master transfer starts.
- The SSI slave-state machine samples data to transmit.

This issue causes written data to be lost.

Workaround(s): Use TI's Unified Network Processor Interface (NPI).

Advisory 05 — *Motorola® SPI Format Slave Mode Writes to Transmit FIFO Can Lose Data*

Details: If the SSI is configured to operate in Motorola® SPI format slave mode, loss of write data can occur when the following two conditions are met:

1. A write to the SSI data register (SSIn:DR) occurs between a new SPI master transfer starting and the end of the first bit of incoming data.
2. A write to the SSI data register (SSIn:DR) occurs during the first bit of new incoming data in a back-to-back transfer sequence.

For more details, see the Synchronous Serial Interface (SSI) chapter in the *CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual*.

Workaround(s): Use TI's Unified Network Processor Interface (NPI).

Advisory 06 — *Insufficient Power Supply Recharging When Using the Sensor Controller Might Cause the System to Hang or Force a Pin Reset*

Details: When the CC1350 device goes into standby, a time interval must be set for the initial period between VDDR recharges based on temperature and certain device-specific trims. However, if the Sensor Controller wakes up and puts the device back into standby without waking the CM3, the initial recharge period programmed by the CM3 will be used again by the hardware. If the temperature has increased sufficiently between the time the CM3 puts the device into standby and the time the Sensor Controller puts the device into standby, the recharge period programmed may be too long. VDDR may drop below the permitted minimum threshold due to the temperature induced increased leakage. This may cause an inconsistent internal state in the device or cause the device to hang, forcing a pin reset.

Workaround(s): If the device is put in standby while using the Sensor Controller, the user must ensure that the initial recharge period programmed by the CM3 is more conservative than what is strictly necessary at the current temperature. The longest duration that the CM3 sleeps at a specific temperature gradient, defines a maximum limit for temperature increase between recalculations of the initial recharge period. The sleep duration can be altered such that a decrease in the converged recharge period matches the margin subtracted from the initial recharge period.

The following parameters are needed for the workaround:

- The user’s maximum expected temperature gradient in °C/s
- The upper bound of the user’s expected operating temperature range in °C
- The initial recharge period set at the upper bound of the user’s expected operating temperature range in SCLK_LF clock periods
- A table of converged recharge periods in SCLK_LF clock periods at varying temperatures decreasing from the upper bound of the user’s expected operating temperature
- The margin in SCLK_LF clock periods that the user wants to subtract from the regular initial recharge period. The higher this value, the longer the device can sleep before requiring a wakeup to recalculate the initial recharge period. However, higher values will lead to slower convergence of the recharge algorithm towards the ideal recharge period at a given temperature.

Based on the parameters above, the user can make the following calculations to determine the maximum duration the CM3 can sleep before waking up to recalculate the initial recharge period:

- Add the margin to the converged recharge period at the upper bound of your operating temperature.
- Find the value in the table of converged recharge periods that comes closest to this value and determine the temperature this value occurs at.
- Find the temperature difference between that temperature and the upper bound of your expected operating temperature range.
- Divide this temperature delta by your maximum expected temperature gradient.

The result of the calculations above is a time value in seconds. This value is the longest time the device can safely stay in standby before a CM3 wakeup must be forced if the Sensor Controller is active.

To apply the workaround, convert this duration from seconds to system ticks. When using SIMPLELINK-CC13X0-SDK_1.60.00.21 or later, apply this value into PowerCC26XX_Config.maxStandbyDuration, set PowerCC26XX_Config.enableMaxStandbyDuration to true, and set PowerCC26XX_Config.vddrRechargeMargin to the value chosen earlier.

Advisory 07 — *Wrong Reset Source Indication*

Details: The field RESET_SRC in the register AON_SYSCTL:RESETCTL shows the source of the last system reset. Occurrence of one of the reset sources may trigger several other reset sources as essential parts of the system are undergoing a reset. This field will report the source of the reset (not the other resets that are consequence of the system reset). To support this feature, the actual register is not captured before the reset source is released. If a new reset source is triggered within a window of four 32-kHz periods after the previous reset source was released, the register field RESET_SRC may indicate power-on reset as source regardless of the actual reset source.

Workaround(s): None

Advisory 08 — *Temporary Loss of Receive Function During Continuous Receive Operation Over Long Time*

Details: The CC1350 modem has a mismatch in the data rates between two modules. If a CC1350 device operates in continuous receive mode for a long period of time without receiving any packets, the mismatch might cause a temporary loss of proper RF reception. This temporary loss occurs at a rate that depends on operating frequency and data rate settings, and lasts for a short period of time before the receive functionality self-recovers. For Bluetooth® Low Energy (BLE) channels with 1 Mbps or Coded PHY, the issue will not occur earlier than 2.8 seconds after start of the receiver, and for BLE channels with 2 Mbps PHY, the issue will not occur earlier than 1.4 seconds after start of the receiver. The only BLE link layer state that allows continuous receive operation of that duration is the scanning state operating on an advertising channel with 1 Mbps or coded PHY. For BLE advertising channels and 1 Mbps or coded PHY, the issue will not occur earlier than 5 seconds after start of the receiver.

The problem has been fixed in SIMPLELINK-CC13X0-SDK_2.20.00.38 and later.

Workaround(s): **Bluetooth® Low Energy designs**

Use SIMPLELINK-CC13X0-SDK_2.20.00.38 or later. For customers using an SDK that pre-dates 2.20, you may experience some packet loss unless you limit the scan window to 5 seconds or lower.

Sub-1 GHz designs

Use SIMPLELINK-CC13X0-SDK_2.20.00.38 or later. For customers using an SDK that pre-dates 2.20, the operational mismatch between the two modules in the modem and resulting temporary loss in the receiver can be avoided by updating the fractFreq value in the radio commands to generate correct settings for the radio.

SmartRF™ Studio v2.8.0 or later can be used to generate the correct fractFreq value for the radio at each operating frequency and data rate. This (new) fracFreq value can be directly applied to settings generated by older SmartRF™ Studio versions. Except for avoiding the temporary loss in the receiver, RF functionalities and performance are not affected by updating the fractFreq value.

Advisory 09 — *Receive Sensitivity Degradation in Continuous Receive Operation Over Long Time*

Details: The CC1350 receiver uses an advanced and dynamic synchronization word search algorithm to optimize the reception of wanted packets and minimize false packet detection. If a CC1350 device operates in continuous receive mode for a long period of time, the dynamic synchronization word search algorithm might in some rare cases lead to unusually high internal threshold values and thereby degraded receive sensitivity. The sensitivity degradation probability depends on the operating noise and interfering conditions and the auto-correlation properties of the synchronization word.

Workaround(s): Use SIMPLELINK-CC13X0-SDK_2.20.00.38 or later. For customers using an SDK that pre-dates 2.20, a restart of the receiver resets all internal threshold values and ensures optimal performance of the synchronization word search operation and best receive sensitivity.

Advisory 10 — *Late Time-out of Receiver Operation in Sniff Mode*

Details: The receive sniff mode commands might run the receiver longer than specified for certain settings, leading to increased average current consumption.

When running CMD_PROP_RX_SNIFF or CMD_PROP_RX_ADV_SNIFF, and the following conditions are met:

- csEndTrigger is set to something other than TRIG_NEVER and
- the channel is busy when the trigger defined by csEndTrigger and csEndTime is observed,

then the carrier sense operation will always end, regardless of csConf.busyOp.

After this, the receiver will run until a packet is received or the command times out due to endTrigger.

Workaround(s): Use SIMPLELINK-CC13X0-SDK_2.20.00.38 or later.

Advisory 11 — *Radio Frequency Error Glitch When Device Switches From IDLE to ACTIVE Mode While Radio is in TX or RX*

Details: If the device switches from IDLE to ACTIVE mode (for example, if the CPU starts executing code) while the radio is transmitting or receiving, it can result in a frequency deviation error in the modulated signal or erroneous signal reception.

Workaround(s): Keep the flash ON in IDLE by using the TI Power driver API:
Power_setConstraint(PowerCC26XX_NEED_FLASH_IN_IDLE)

Advisory 12 — *Slow Transition Across Brown-Out Detect (BOD) Threshold Might Cause the Device to Hang*

Details:

For applications using non-rechargeable (primary) battery, the issue described in this advisory would potentially occur only at end-of-life of the battery, and therefore a workaround is not necessary as the battery would anyway need to be replaced, triggering a power-on reset.

If the VDDS supply voltage is held in the BOD threshold region (approximately 1.78 V), the device might on rare occasions end up in a lock-up state. The current draw is approximately 2.25 mA in this state. The device will not exit this state by increasing the VDDS supply voltage above the BOD threshold. To get out of this state, a pin reset must be performed or the VDDS supply voltage must be decreased below the power-on reset (POR) threshold (1.0 V), triggering a POR reset.

The lock-up state is triggered if a brown-out-detect (BOD) event occurs during specific stages of the boot code execution. There are two critical, narrow time windows, each of approximately 10 ns duration, and both of these time windows occur within 100 µs to 1 ms after the reset event that started the boot code execution. Typically, this can happen when the supply voltage is ramped slowly across the BOD threshold. Supply resistance, in combination with device startup current will then pull the VDDS supply voltage below the BOD threshold multiple times as the device turns on and off due to resets.

For Li-Ion and NiMH rechargeable batteries, a first level protection disconnecting the chip VDDS supply would typically prevent the device from entering this state during battery discharge as the device power supply would fall below the POR threshold.

Workaround(s):

The following workarounds must be implemented:

The specified operating supply voltage range for the device is 1.8 V to 3.8 V. When using rechargeable batteries, the battery protection system must ensure that either:

- The device supply voltage remains at or above the minimum operating supply voltage (1.8 V) once powered on, or
- If the device supply is discharged below the minimum operating supply voltage (1.8 V), the device must be reset (pin or power-on reset) when the supply is charged above the minimum operating supply voltage (1.8 V) again.

Advisory 13 — *Limited Number of DIOs Available for the Bootloader Backdoor*

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_{x-1}, even if higher DIO numbers are available for the device.

Workarounds: There are no workaround for this issue.

Advisory 14 — *An Incorrect Value Might Be Written to the General-Purpose (GP) Timers MMRs (Memory Mapped Registers) When Simultaneously Accessing the AES (Advanced Encryption Standard) Engine from a Different Master*

Details: When writing data to the GP Timer MMRs from one master while simultaneously accessing the AES modules from another master (read/write), an incorrect value might be captured in the GP Timer MMRs. In some cases, the incorrect value is replaced by the correct one after two clock cycles, but not always. No issue is seen when accessing the modules from the same master.

Workaround 1: Avoid accesses to AES by other masters while writing to the GP Timer MMRs. This can be accomplished by acquiring the relevant semaphores (depending on drivers/stacks) before writing to the GP Timer.

Note that the BLE stack executes crypto operations from ISR context. Other software must release the semaphores from a higher-priority ISR to avoid deadlocks.

Workaround 2: Verify the value written to the GP Timer MMR by reading it back in software. Correct the value if necessary.

Advisory 15 — *CMD_TX_TEST May End Before Transmitting*

Details: The CMD_TX_TEST may end before transmitting anything if it follows another command in a chain, unless the command directly in front of CMD_TX_TEST is CMD_FS.

Workaround 1: Insert a CMD_FS directly before any CMD_TX_TEST in the chain.

Workaround 2: Send the CMD_TX_TEST as a standalone command instead of using a chain.

Advisory 16 *Periodic ADC Trigger at 200 kHz Rate Can Be Ignored When XOSC_HF Is Turned On or Off*

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).

SCLK_HF switches from RCOSC_HF to XOSC_HF at different times compared to the ADC clock. 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 1.5% (1 clock cycle 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 1/24 µs or more.

Advisory 17 — *ADC Samples Can Be Delayed by 2 clock Cycles (24 MHz) When XOSC_HF Is Turned On or Off, Resulting in Sample Jitter*

Details: There is no dedicated clock source selection for the ADC clock. 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).

SCLK_HF switches from RCOSC_HF to XOSC_HF at different times compared to the ADC clock. This leads to sample jitter.

Workaround 1: Use asynchronous sampling

- 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.

Advisory 18 — *Writing Any Value to AUX_ANAIF:ADCTRIG.START Will Create an ADC Trigger*

Details: The original register documentation states that you need to write a ‘1’ to AUX_ANAIF:ADCTRIG.START to manually trigger the ADC. This is not correct, as any write will do this.

Workaround: Avoid writing to AUX_ANAIF:ADCTRIG.START unless an ADC trigger should be generated.

4 Revision History

Changes from Revision C (October 2018) to Revision D (December 2020) **Page**

- Updated the numbering format for tables, figures, and cross-references throughout the document.................. 2
- Added Advisories 13 through 18.. 2