Errata

CC2340R5 SimpleLink™ Wireless MCU Device Revision B

ABSTRACT

This document describes the known exceptions to functional specifications (advisories) to the CC2340R5 SimpleLink™ device.

Table of Contents

Trademarks

SimpleLink™ and Texas Instruments™ are trademarks of Texas Instruments.
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
SPI	Advisory SPI_04 —Hang scenario with SPI waiting for CPU intervention forever	Yes
ADC	Advisory ADC_08 —ADC BUSY bit not cleared in repeat single, sequence and repeat sequence conversion modes.	Yes
ADC	Advisory ADC_09—ADC can have random conversion errors.	Yes
BATMON	Advisory BATMON_01 —Incorrect temperature measurement	Yes
CKM	Advisory CLK_01—Bluetooth Low Energy link may not be maintained when using LFOSC only	Yes
I2C	Advisory I2C_02—SDA and SCL open-drain output buffer issue	Yes
GPIO	Advisory GPIO_01—Open-drain configuration can drive a short high pulse	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, XCC2340R5). 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 is still undefined. Only qualified production devices are to be used.

2.2 Devices Supported

This document supports the following device:

- CC2340R5

2.3 Package Symbolization and Revision Identification

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

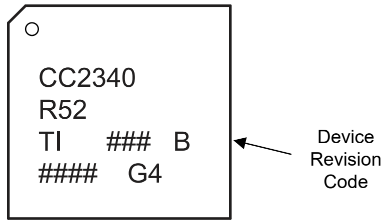

Figure 2-1. Package Symbolization

Table 2-1. Revision Identification

Device Revision Code	Silicon Revision
B	PG2.0

3 Advisories

SPI_04 ***Hang scenario with SPI waiting for CPU intervention forever.***

Revisions Affected B

Details When the CPU is reading or writing the SPI FIFO using FIFO level triggers to generate interrupts, the system can hang. After the first interrupt is serviced, the FIFO level can permanently be below or above the configured threshold and not generate a subsequent CPU interrupt. This can lead to a hang scenario with SPI waiting for CPU intervention forever.

Workaround

1. Use polling of FIFO status within SPI and do not rely on FIFO level configured interrupts, or
2. Use only empty/overflow interrupts and don't use FIFO level configured interrupts, or
3. Use FIFO level configured interrupts along with empty (for TXFIFO) and overflow (for RXFIFO) as a failsafe to avoid hang scenarios.

ADC_08 ***ADC BUSY bit not cleared in repeat single, sequence and repeat sequence conversion modes.***

Revisions Affected B

Description When ADC is configured in repeat single, sequence or repeat sequence conversion modes with trigger policy as trigger next in the MEMCTLx register, software attempting to stop the conversion sequence by clearing ENC bit does not clear BUSY bit in the STATUS register. In the case of sequence conversion mode with trigger next policy, the BUSY bit is cleared at the end of the conversion sequence.

Workaround To stop the conversions and to clear the BUSY bit in the above mentioned ADC operating scenario, the following software sequence can be followed.

1. Write CTL0.ENC = 0
2. Change CTL1.TRIGSRC to SOFTWARE
3. Write CTL1.SC=1

ADC_09 *ADC can have random conversion errors.*

Revisions Affected B

Description

ADC can have errors at the rate as high as 1 in 330 million ADC conversions. When a conversion error occurs, the error results in a jump in the digital output of the ADC without a corresponding change in the ADC input voltage, otherwise known as a ‘sparkle code’. The magnitude of the jump is 64 LSBs higher or lower than the expected ADC output when ADC is used in 12-bit resolution setting. The magnitude of the jump decreases to +/-16 LSBs for 10-bit resolution and +/-4 LSBs when set to 8-bit resolution.

Workaround

The error rate can be reduced to 1 error in 80 billion ADC conversions by setting ADC.DEBUG1:CTRL[10:9] bits high.

Other software workarounds like a best-out-of-three, where out of three consecutive samples, the one with the highest standard deviation is discarded and the other two averaged to generate the ADC output can also be considered.

Software averaging of 16 consecutive ADC outputs decreases the deviation of the ADC output to +/- 4 LSBs when set to 12-bit resolution.

These workarounds will be incorporated into a future release of the SimpleLink™ Low Power F3 software development kit (SDK) for the CC23xx devices.

BATMON_01 ***Incorrect temperature measurement.***

Revisions Affected B

Description BATMON can report incorrect temperatures when hysteresis is enabled. To prevent potential incorrect temperature reports, the user must always disable BATMON hysteresis.

Workaround Hysteresis is controlled by the PMUD.CLT[2] HYST_EN bit.

Hysteresis is enabled by default (reset value = 1) and, therefore, must actively be disabled during boot.

Hysteresis can be disabled by clearing the PMUD.CLT[2] HYST_EN bit using the following command:

HWREG(PMUD_BASE + PMUD_O_CTL) = (PMUD_CTL_CALC_EN | PMUD_CTL_MEAS_EN)

This workaround is incorporated into SIMPLELINK-LOWPOWER-F3-SDK versions >= 7.40.xx

CLK_01 ***Bluetooth Low Energy link can not be maintained when using LFOSC only***

Revisions Affected B

Description A small percentage of devices do not maintain a Bluetooth Low Energy link if LFOSC is used as a sleep clock due to random timing error above 500 PPM.

Workaround A software workaround are available in the SimpleLink F3 SDK >= 8.10.xx to allow devices to operate in the broadcaster, observer and peripheral roles when using LFOSC only. When the software workaround is used, the device can see short periods of operation with reduced throughput and increased power consumption when the timing error occurs. This software workaround does not support the central role. To completely avoid the consequences of increased power consumption and connection throughput or support the Central role, TI recommends using an external 32.768kHz crystal.

I2C_01 *SDA and SCL Open-Drain Output Buffer Issue*

Revisions Affected B

Description

The SDA and SCL outputs are implemented with push-pull 3-state output buffers rather than open-drain output buffers as required by I^2C. While it is possible for the push-pull 3-state output buffers to behave as open-drain outputs, an internal timing skew issue causes the outputs to drive a logic-high for a duration of approximately 1ns–2ns before the outputs are disabled. The unexpected high-level pulse only occurs when the SCL or SDA outputs transition from a driven low state to a high-impedance state.

This short high-level pulse injects energy in the I^2C signals traces, which causes the I^2C signals to sustain a period of ringing as a result of multiple transmission line reflections. This ringing should not cause an issue on the SDA signal because it only occurs at times when SDA is expected to be changing logic levels and the ringing will have time to damp before data is latched by the receiving device. The ringing can have enough amplitude to cross the SCL input buffer switching threshold several times during the first few nanoseconds of this ringing period, which can cause clock glitches. This ringing should not cause a problem if the amplitude is damped within the first 50ns because I^2C devices are required to filter the SCL inputs to remove clock glitches. Therefore, it is important to design the PCB signal traces to limit the duration of the ringing to less than 50ns. One possible way to reduce the ringing is to insert series termination resistors near the SCL and SDA terminals to attenuate transmission line reflections.

This issue may also cause the SDA output to be in contention with the target SDA output for the duration of the unexpected high-level pulse when the target begins an ACK cycle. This occurs because the target may already be driving SDA low before the unexpected high-level pulse occurs. The glitch that occurs on SDA as a result of this short period of contention does not cause any I^2C protocol issue but the peak current applies unwanted stress to both I^2C devices and potentially increases power supply noise. Therefore, a series termination resistor located near the respective SDA terminal is required to limit the current during the short period of contention.

A similar contention problem can occur on SCL when connected to I^2C target devices that support clock stretching. This occurs because the target is driving SCL low before the unexpected high-level pulse occurs. The glitch that occurs on SCL as a result of this short period of contention does not cause any I^2C protocol issue because I^2C devices are required to apply a glitch filter to their SCL inputs. However, the peak current applies unwanted stress to both I^2C devices and potentially increases power supply noise. Therefore, a series termination resistor located near the respective SCL terminal is required to limit the current during the short period of contention.

If another controller is connected, the unexpected high-level pulses on the SCL and SDA outputs can cause contention during clock synchronization and arbitration. The series termination resistors described above also limit the contention current in this use case without creating any I^2C protocol issue.

Workaround

Insert series termination resistors on the SCL and SDA signals and locate them near the SCL and SDA terminals along with the SCL and SDA pullup resistors.

The ringing can also be reduced by controlling the output to use minimum drive strength and reduced slew rate. These options are only configurable on pins that support high drive output. Standard drive pins have no configuration options.

GPIO_01 ***Open-Drain Configuration Can Drive a Short High Pulse***

Revisions Affected B

Description

Each DIO can be configured to an open-drain mode using the IOCx register.

However, an internal device timing issue may cause the GPIO to drive a logic-high for approximately 1ns–2ns during the transition into or out of the high-impedance state. This undesired high-level can cause the GPIO to be in contention with another open-drain driver on the line if the other driver is simultaneously driving low. The contention is undesirable because it applies stress to both devices and results in a brief intermediate voltage level on the signal. This intermediate voltage level may be incorrectly interpreted as a high level if there is not sufficient logic filtering present in the receiver logic to filter this brief pulse.

Workaround

If contention is a concern, do not use the open-drain functionality of the GPIOs; instead, emulate open-drain mode in software. Open-drain emulation can be achieved by setting the GPIO data (DOUT31_0.DIOx) to a static 0 and driving the GPIO output enable (DOE31_0.DIOx) to enable or disable the drive low. For an example implementation, see the code below.

```
#include <ti/devices/cc23x0r5/driverlib/gpio.h>
    /* Call driver init functions */
    GPIO_init();

    //Set GPIO data (DOUT31_0.DIOx) to static 0
    GPIOClearDio(CONFIG_GPIO_LED_0);
    while(1) //loop below toggles the LED on and off every 1 second
    {
    GPIOSetOutputEnableDio(CONFIG_GPIO_LED_0, 1);
    sleep(1);
    GPIOSetOutputEnableDio(CONFIG_GPIO_LED_0, 0);
    sleep(1);
    }
```

4 Revision History

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

Changes from February 24, 2024 to August 11, 2024 (from Revision C (February 2024) to Revision D (August 2024)) **Page**

Changes from October 24, 2023 to February 24, 2024 (from Revision B (October 2023) to Revision C (February 2024)) **Page**

Changes from April 1, 2023 to October 23, 2023 (from Revision A (April 2023) to Revision B (October 2023)) **Page**

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