Silicon Errata

SWRZ078B–January 2018–Revised March 2019

CC1312R SimpleLink™ Wireless MCU Device Revision E

This document describes the known exceptions to functional specifications (advisories) to the CC1312R SimpleLink™ device.

Topic **Page**

1 Advisories Matrix

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

Table 1. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		E
Radio	Advisory Radio_01 — Proprietary Radio Modes: Spurious Emissions Can Affect Regulatory Compliance	Yes
Power	Advisory Power_03 — Increased Voltage Ripple at Low Supply Voltages When DC/DC Converter is Enabled	Yes
PKA	Advisory PKA_01 — Public Key Accelerator (PKA) Interrupt Line is Always High When Module is Enabled and PKA is Idle	Yes
PKA	Advisory PKA_02 — Public Key Accelerator (PKA) RAM is Not Byte Accessible	Yes
I2C	Advisory I2C_01 — I^2C Module Master Status Bit is Set Late	Yes
I2S	Advisory I2S_01 — I^2S Bus Faults are Not Reported	Yes
CPU	Advisory CPU_01 — Arm® Errata #838869: Store Immediate Overlapping Exception Return Operation Might Vector to Incorrect Interrupt	Yes
CPU	Advisory CPU_02 — Arm® Errata #752770: Interrupted Loads to SP Can Cause Erroneous Behavior	Yes
CPU	Advisory CPU_03 — Arm® Errata #776924 VDIV or VSQRT Instructions Might Not Complete Correctly When Very Short ISRs are Used	Yes
CPU, System	Advisory 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	Yes
System	Advisory Sys_01 — Device Might Boot Into ROM Serial Bootloader When Waking Up From Shutdown	Yes
System Controller	Advisory SYSCTRL_01 — Resets Occurring in a Specific 2-MHz Period During Initial Power Up are Incorrectly Reported	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, XCC1312R). 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:

- CC1312R

2.3 Package Symbolization and Revision Identification

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

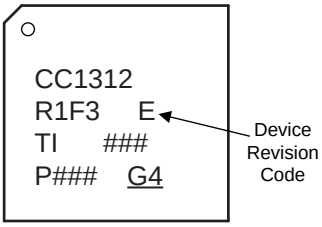

Figure 1. Package Symbolization for Silicon Revision E

Table 2. Revision Identification

DEVICE REVISION CODE	SILICON REVISION
E	PG2.1

3 Device Revision E Advisories

This section lists the advisories for this silicon revision.

3.1 Advisories

Table 3. Device Revision E Advisory List

Advisory Radio_01 *Proprietary Radio Modes: Spurious Emissions Can Affect Regulatory Compliance*

Revisions Affected: Revision E and earlier

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 × 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)

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 all Software Development Kit (SDK) examples:
#define SET_CCFG_MODE_CONF_XOSC_CAPARRAY_DELTA -128
#define SET_CCFG_MODE_CONF_XOSC_CAP_MOD 0

Advisory Power_03 *Increased Voltage Ripple at Low Supply Voltages When DC/DC Converter is Enabled*

Revisions Affected: Revision E and earlier

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 <2.2V.

The workaround is available in all SDK versions.

Advisory PKA_01 *Public Key Accelerator (PKA) Interrupt Line is Always High When Module is Enabled and PKA is Idle*

Revisions Affected: Revision E and earlier

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 following SimpleLink software development kit (SDK) versions:

- SimpleLink CC13x2 SDK 1.60.00.xx and later

Advisory PKA_02 *Public Key Accelerator (PKA) RAM is Not Byte Accessible*

Revisions Affected: Revision E and earlier

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 following SimpleLink software development kit (SDK) versions:

- SimpleLink CC13x2 SDK 1.60.00.xx and later

Advisory I2C_01 *I²C Module Master Status Bit is Set Late*

Revisions Affected: Revision E and earlier

Details: The I2C.MSTAT[0] bit is not set immediately after writing to the I2C.MCTRL register. This can lead an I²C 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 all Software Development Kit (SDK) versions.

Advisory I2S_01 *I²S Bus Faults are Not Reported*

Revisions Affected: Revision E and earlier

Details: The I²S module will not set the bus error interrupt flag (I2S0.IRQFLAGS.BUS_ERR) if an I²S 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 I²S 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.

Advisory CPU_01 *Arm® Errata #838869: Store Immediate Overlapping Exception Return Operation Might Vector to Incorrect Interrupt*

Revisions Affected: Revision E and earlier

Details:

Configurations Affected:

This erratum only affects systems where writeable memory locations can exhibit more than one wait state (system SRAM do 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 pended by a level-based interrupt that is cleared by C's handler then interrupt C will be pended 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.

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.

Advisory CPU_02 *Arm® Errata #752770: Interrupted Loads to SP Can Cause Erroneous Behavior*

Revisions Affected: Revision E and earlier

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.

Advisory CPU_03 *Arm® Errata #776924: VDIV or VSQRT Instructions Might Not Complete Correctly When Very Short ISRs are Used*

Revisions Affected: Revision E and earlier

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 VQSRT 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.

Advisory 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 E and earlier

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.

Advisory Sys_01 *Device Might Boot Into ROM Serial Bootloader When Waking Up From Shutdown*

Revisions Affected: Revision E and earlier

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 CC13x2, CC26x2 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.

Advisory SYSCTRL_01 *Resets Occurring in a Specific 2-MHz Period During Initial Power Up are Incorrectly Reported*

Revisions Affected: Revision E and earlier

Details: 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.

Workaround: None

Trademarks

SimpleLink, Texas Instruments are trademarks of Texas Instruments.
Arm, Cortex are registered trademarks of Arm Limited (or its subsidiaries).

Revision History

Changes from Original (January 2018) to B Revision **Page**

- Changed Device Revision to "E" 1
- Changed package symbolization image to reflect Device Revision "E" 4
- Deleted Advisory RadioFW_01 5
- Deleted Advisory RadioFW_03 5
- Deleted Advisory RadioFW_04 5
- Deleted Advisory RadioFW_05 5
- Deleted Advisory RadioFW_06 5
- Deleted Advisory RadioFW_11 5
- Deleted Advisory RadioFW_12 5
- Deleted Advisory RadioFW_16 5
- Deleted Advisory Radio_Osc_01 5
- Deleted Advisory Power_02 5
- Deleted Advisory Power_04 5
- Deleted Advisory Power_05 5
- Deleted Advisory Sys_02 5
- Added a more detailed description to Radio_01 5
- Added clarification and example to Advisory I2S_01 8
- Added link to technical reference manual in Advisory Sys_01 14
- Added clarification to Advisory SYSCTRL_01 14

IMPORTANT NOTICE AND DISCLAIMER

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

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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