# Mighty Gecko
# EFR32MG13 Errata

This document contains information on the EFR32MG13 errata. The latest available revision of this device is revision D.

For errata on older revisions, refer to the errata history section for the device. The revision information is typically specified in or near the trace code on the device. Refer to the package marking information in the data sheet for more information.

Errata effective date: May, 2019.

# 1. Active Errata Summary

These tables list all known errata for the EFR32MG13 and all unresolved errata in revision D of the EFR32MG13.

**Table 1.1. Errata History Overview**

| Designator | Title/Problem | Exists on Revision: | | | |
|---|---|---|---|---|---|
| | | A | B | C | D |
| ADC_E213 | ADC KEEPINSLOWACC Mode | X | X | X | X |
| ADC_E222 | ADC EM2 Wakeup on a Comparator Match Disables EM2 Entry | X | — | — | — |
| ADC_E224 | ADC Warm-Up Ready Can Cause IDAC, ACMP, or CSEN to Not Function | X | X | — | — |
| ADC_E225 | Using the ADC in High Accuracy Bias Mode Will Force All Analog Peripherals to High Accuracy Bias Mode | X | X | — | — |
| ADC_E228 | Limited ADC Sampling Frequency in EM2 | X | X | X | X |
| CSEN_E201 | CSEN_DATA in Debug Mode | X | X | X | X |
| CSEN_E202 | CSEN Baseline DMA Transfers | X | X | X | X |
| CUR_E203 | Occasional Extra EM0/1 Current | X | X | X | X |
| DBG_E203 | CRC Debug Command Available | X | — | — | — |
| DBG_E204 | Debug Recovery with JTAG Does Not Work | X | X | X | X |
| EMU_E211 | Radio Clocks Remain Disabled After Voltage Scaling | X | — | — | — |
| EMU_E214 | Device Erase Cannot Occur if Voltage Scaling Level is Too Low | X | X | X | X |
| I2C_E202 | Race Condition Between Start Detection and Timeout | X | X | X | X |
| I2C_E203 | I2C Received Data Can be Shifted | X | X | X | X |
| I2C_E204 | I2C0 Does Not Meet Fast Mode Plus (Fm+) Timing at Voltage Scale Level 0 | X | X | X | X |
| I2C_E205 | Go Idle Bus Idle Timeout Does Not Bring Device to Idle State | X | X | X | X |
| I2C_E206 | Slave Holds SCL Low After Losing Arbitration | X | X | X | X |
| RMU_E202 | External Debug Access Not Available After Watchdog or Lockup Full Reset | X | X | X | X |
| RMU_E203 | AVDD Ramp Issue | X | X | — | — |
| RTCC_E203 | Potential Stability Issue with RTCC Registers | X | X | X | X |
| RTCC_E205 | Wrap Event Can Be Missed | X | X | X | X |
| TRNG_E202 | Standards Non-compliance | X | X | X | — |
| USART_E203 | DMA Can Miss USART Receive Data in Synchronous Mode | X | X | X | — |
| USART_E204 | IrDA Modulation and Transmission of PRS Input Data | X | X | X | X |

**Table 1.2. Active Errata Status Summary**

| Errata # | Designator | Title/Problem | Workaround Exists | Affected Revision | Resolution |
|---|---|---|---|---|---|
| 1 | ADC_E213 | ADC KEEPINSLOWACC Mode | No | D | — |
| 2 | ADC_E228 | Limited ADC Sampling Frequency in EM2 | No | D | — |
| 3 | CSEN_E201 | CSEN_DATA in Debug Mode | Yes | D | — |
| 4 | CSEN_E202 | CSEN Baseline DMA Transfers | Yes | D | — |
| 5 | CUR_E203 | Occasional Extra EM0/1 Current | No | D | — |
| 6 | DBG_E204 | Debug Recovery with JTAG Does Not Work | Yes | D | — |
| 7 | EMU_E214 | Device Erase Cannot Occur if Voltage Scaling Level is Too Low | Yes | D | — |
| 8 | I2C_E202 | Race Condition Between Start Detection and Timeout | Yes | D | — |
| 9 | I2C_E203 | I2C Received Data Can be Shifted | Yes | D | — |
| 10 | I2C_E204 | I2C0 Does Not Meet Fast Mode Plus (Fm+) Timing at Voltage Scale Level 0 | No | D | — |
| 11 | I2C_E205 | Go Idle Bus Idle Timeout Does Not Bring Device to Idle State | Yes | D | — |
| 12 | I2C_E206 | Slave Holds SCL Low After Losing Arbitration | Yes | D | — |
| 13 | RMU_E202 | External Debug Access Not Available After Watchdog or Lockup Full Reset | Yes | D | — |
| 14 | RTCC_E203 | Potential Stability Issue with RTCC Registers | Yes | D | — |
| 15 | RTCC_E205 | Wrap Event Can Be Missed | Yes | D | — |
| 16 | USART_E204 | IrDA Modulation and Transmission of PRS Input Data | Yes | D | — |

# 2. Detailed Errata Descriptions

## 2.1 ADC_E213 – ADC KEEPINSLOWACC Mode

| ***Description of Errata*** |
|---|
| When WARMUP-MODE in ADCn_CTRL is set to KEEPINSLOWACC, the ADC does not track the input voltage. Also, the ADC keeps the input muxes closed even during channel switching, making it not recommended to operate the ADC in KEEPINSLOWACC mode. |
| ***Affected Conditions / Impacts*** |
| KEEPINSLOWACC warmup mode does not function properly. |
| ***Workaround*** |
| There is currently no workaround for this issue. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

## 2.2 ADC_E228 – Limited ADC Sampling Frequency in EM2

| ***Description of Errata*** |
|---|
| ADC FIFO overflows occur at frequencies that are much lower than the ADC's maximum theoretical sampling rate. |
| ***Affected Conditions / Impacts*** |
| ADC sampling frequency is reduced in EM2. |
| ***Workaround*** |
| There is currently no workaround for this issue. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

## 2.3 CSEN_E201 – CSEN_DATA in Debug Mode

| ***Description of Errata*** |
|---|
| Reading CSEN_DATA in debug mode inadvertently clears pending CSEN data DMA requests. |
| ***Affected Conditions / Impacts*** |
| Reads of CSEN_DATA clear pending receive data DMA requests. This would be expected in normal operation as the DMA reads CSEN_DATA to transfer newly acquired results. These reads are intentional, but any read of CSEN_DATA, including while in debug mode, has the same effect. Thus, viewing the CSEN module registers in a debugger, such as in Simplicity Studio, can inadvertently clear pending CSEN DMA requests resulting in subsequent data being received out of order and with insertions of random data. |
| ***Workaround*** |
| Do not use a debugger to read the CSEN registers while DMA is enabled. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.4 CSEN_E202 – CSEN Baseline DMA Transfers

| ***Description of Errata*** |
|---|
| DMA transfers to CSEN_DMBASELINE do not occur in the expected order. |
| ***Affected Conditions / Impacts*** |
| When using delta modulation, a baseline value must be written to CSEN_DMBASELINE before each conversion. However, when DMA is used to do this, these writes occur after the desired conversion instead of before the conversion as is required. This means that in a given sequence of conversions serviced by DMA, the write to CSEN_DMBASELINE that should happen before conversion N is actually written in advance of conversion N + 1, leading to potentially erroneous results. |
| ***Workaround*** |
| Manually write the first value to CSEN_DMBASELINE and then use the DMA to perform subsequent baseline writes. Thus, in the case of a sequence consisting of four conversions, the first baseline value would be written to CSEN_DMBASELINE under software control (e.g. before the conversion trigger occurs). The next three values can be written by the DMA after the first and each subsequent conversion occurs.<br><br>After the final conversion, which would be the fourth in this example, the DMA will service a final write request to CSEN_DMBASELINE. This final transfer can be (1) a dummy value if no further conversions are required, (2) the initial baseline value in the case where conversions are repeated in a loop, or (3) the initial baseline value for a new, yet-to-be-triggered series of conversions. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.5 CUR_E203 – Occasional Extra EM0/1 Current

| ***Description of Errata*** |
|---|
| Occasionally when exiting EM2, a low voltage oscillator sometimes continues to run and causes the device to draw an extra ~10 μA when in EM0 or EM1. This oscillator automatically resets when entering EM2 or EM3, so the extra current draw is not present in these modes. |
| ***Affected Conditions / Impacts*** |
| Systems using EM2 may occasionally see an extra ~10 μA of current draw in EM0 or EM1. |
| ***Workaround*** |
| There is currently no workaround for this issue. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.6 DBG_E204 – Debug Recovery with JTAG Does Not Work

| ***Description of Errata*** |
|---|
| The debug recovery algorithm of holding down pin reset, issuing a System Bus Stall AAP instruction, and releasing the reset pin does not work when using the JTAG debug interface. When using the JTAG debug interface, the core will continue to execute code as soon as the reset pin is released. |
| ***Affected Conditions / Impacts*** |
| The debug recovery sequence will not work when using the JTAG debug interface. |
| ***Workaround*** |
| Use the Serial Wire debug interface to implement the debug recovery sequence. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.7 EMU_E214 – Device Erase Cannot Occur if Voltage Scaling Level is Too Low

| ***Description of Errata*** |
|---|
| The device erase logic does not check the Voltage Scale Level prior to attempting a device erase. If using Voltage Scale Level 0 (1 V), the device may not be able to erase the flash. This results in a potentially unlockable device if operating at Voltage Scale Level 0 (1 V). |
| ***Affected Conditions / Impacts*** |
| It is possible that the flash is only partially erased when performing the operation at Voltage Scale Level 0 (1 V). If this results in the debug lock bit not clearing, a locked part doesn't unlock after the partial erasure (which it is intended to do), and the part remains locked. If subsequent erasures continue to fail, the part would remained locked. |
| ***Workaround*** |
| The voltage should be set to Voltage Scale Level 2 (1.2 V) before executing the device erase.<br><br>For systems that don't lock the debug interface, the user can follow the debug recovery procedure to halt the CPU before it has a chance to execute code in software to avoid the code scaling the voltage. The device erase can then be executed at Voltage Scale Level 2 (1.2 V) (the power-on default voltage of the device).<br><br>For systems that do lock the debug interface, firmware can implement a mechanism whereby it can voltage scale or unlock debug access if its defined authentication method is passed. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.8 I2C_E202 – Race Condition Between Start Detection and Timeout

| ***Description of Errata*** |
|---|
| There is a race condition where the Bus Idle Timeout counter may clear the busy status of the I2C bus after a start condition. |
| ***Affected Conditions / Impacts*** |
| Software may attempt another I2C start if it thinks the bus is idle. This may disrupt the I2C bus. After the Bus Idle Timeout feature has triggered, it will not detect another idle condition. |
| ***Workaround*** |
| Software can wait for any of the following conditions before starting an I2C transaction:<br>• The received address match interrupt indicates that the I2C bus is busy. Software should serve this transaction and proceed accordingly. Software can ignore the wrong busy status.<br>• The SSTOPIF interrupt flag indicates that the I2C bus has returned to the idle state.<br>• A defined, system-dependent amount of time to wait after bus activity to ensure that the bus is in idle state. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.9 I2C_E203 – I2C Received Data Can be Shifted

| ***Description of Errata*** |
| --- |
| If SDA falls between detection of the start condition and the first rising edge of SCL, the I2C state machine clears the start condition that was just detected, causing the state machine counter to count the rising edge of SCL earlier than it was detected. This causes the received data to be out of sync and the acknowledge phase to occur one SCL clock cycle earlier than expected, thus corrupting the integrity of the I2C bus.<br><br>There are two ways in which the falling condition on SDA can potentially happen:<br>• In multi-master systems, one master initiates a start condition and then drives SDA high shortly before another master drives SDA low to indicate a start condition.<br>• In a single master system, if SDA is high from the last bit of the previous transaction, the master initiates a start condition and then drives SDA low because the MSB of the new address is low. |
| ***Affected Conditions / Impacts*** |
| I2C operation in slave mode or multi-master mode. |
| ***Workaround*** |
| This depends on whether the system is multi- or single-master. There is no workaround for multi-master cases. In a single master system, the state of SDA may not change unless a new address is being sent, such that the falling condition on SDA would not be observed. Whether or not this is the case is dependent on the implementation of the particular I2C master. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.10 I2C_E204 – I2C0 Does Not Meet Fast Mode Plus (Fm+) Timing at Voltage Scale Level 0

| ***Description of Errata*** |
| --- |
| I2C0 cannot meet Fast Mode Plus (Fm+) timing specifications when the device is operating at Voltage Scale Level 0 (`EMU_STATUS_VSCALE = VSCALE0`). Attempting to do so can result in false NACK/START/STOP conditions during communication. |
| ***Affected Conditions / Impacts*** |
| I2C0 is unable to support Fm+ timing at Voltage Scale Level 0 (`EMU_STATUS_VSCALE = VSCALE0`). |
| ***Workaround*** |
| There is no workaround for I2C0, and this erratum affects only I2C0 on a given device. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.11 I2C_E205 – Go Idle Bus Idle Timeout Does Not Bring Device to Idle State

| ***Description of Errata*** |
|---|
| When the I2C is operating as a slave, if the bus idle timeout is active (`I2Cn_CTRL_BITO != 0`) and the go idle on bus timeout feature is enabled (`I2Cn_CTRL_GIBITO = 1`), the bus idle interrupt flag (`I2Cn_IF_BITO`) sets upon timeout, but the receiver does not enter the idle state. |
| ***Affected Conditions / Impacts*** |
| The I2C receiver needs to detect a START condition to recover from the bus idle timeout state. If there is other, undefined activity on the bus after the timeout, the receiver will not recover as expected. |
| ***Workaround*** |
| The `I2Cn_CTRL_EN` bit can be toggled from 1 to 0 and back to 1 again in order to resume normal operation. Alternatively, a START condition issued by any other master on the bus (including the EFM32/EFR32 device) will reset the receiver and return it to normal operation. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.12 I2C_E206 – Slave Holds SCL Low After Losing Arbitration

| ***Description of Errata*** |
|---|
| If, while transmitting data as a slave, arbitration is lost, SCL is unintentionally held low for an indefinite period of time. |
| ***Affected Conditions / Impacts*** |
| The winner of arbitration cannot use the bus because SCL is never released. |
| ***Workaround*** |
| If the $I^2C$ arbitration lost flag is asserted (I2C_IF_ARBLOST = 1) in slave mode (I2C_STATE_MASTER = 0), application software needs to wait for at least one SCL high time and then issue the transmission abort command (set I2C_CMD_ABORT = 1), thus releasing SCL. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.13 RMU_E202 – External Debug Access Not Available After Watchdog or Lockup Full Reset

| ***Description of Errata*** |
|---|
| When a reset is triggered in full-reset mode, a debugger will not be able to read AHB-AP or ARM core registers. |
| ***Affected Conditions / Impacts*** |
| Systems using the full reset mode for watchdog or lockup resets will see limited debugging capability after one of these resets triggers. |
| ***Workaround*** |
| There are three possible workarounds:<br>• Software should configure peripherals to either LIMITED or EXTENDED mode if full debugger functionality is needed after a watchdog or lockup reset.<br>• When using FULL reset mode, appending at least 9 idle clock cycles to the last debug command will allow the transaction to complete.<br>• A power cycle or hard pin reset will restore normal operation. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.14 RTCC_E203 – Potential Stability Issue with RTCC Registers

| ***Description of Errata*** |
| --- |
| RTCC_LOCK and RTCC_POWERDOWN have the potential to be momentarily unstable under some PCLK, Low Energy Peripheral Clock, and APB write scenarios. This stability issue resolves in approximately 160 ns as the write completes with the assertion of the APB clock pulse. |
| ***Affected Conditions / Impacts*** |
| A write to RTCC_LOCK or RTCC_POWERDOWN may have unintended effects if the write is completed with the Low Energy Peripheral clock enabled (RTCC in the CMU_LFECLKEN0 register is set to 1). |
| ***Workaround*** |
| To avoid this stability issue, configure the RTCC_LOCK and RTCC_POWERDOWN registers with the Low Energy Peripheral clock disabled (RTCC in the CMU_LFECLKEN0 register is cleared to 0).<br><br>This workaround is included in v5.1.0 or later of the Gecko SDK. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.15 RTCC_E205 – Wrap Event Can Be Missed

| ***Description of Errata*** |
| --- |
| The RTCC main counter can miss a CC1 wrap event (CCV1TOP bitfield in the RTCC_CTRL register set to 1) if one of the following registers are written in the same cycle as the wrap event: RTCC_CTRL, RTCC_CNT, RTCC_TIME, RTCC_DATE, RTCC_PRECNT, RTCC_IFC, RTCC_IFS, RTCC_CCx_CCV, RTCC_CCx_CTRL, RTCC_CCx_TIME, RTCC_CCx_DATE, RTCC_CMD, RTCC_RETx_REG. |
| ***Affected Conditions / Impacts*** |
| Systems using the CC1 wrap event feature may miss events if an affected register is written immediately before a wrap occurs. |
| ***Workaround*** |
| There are two workarounds to this issue:<br><br>• Do not use the CC1 wrap event feature (CCV1TOP in RTCC_CTRL should be cleared to 0).<br>• Alternatively, do not write to any of the affected registers when the counter is about to wrap. This means that firmware must check that RTCC_CNT is not close to RTCC_CC1_CCV before writing the register. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

### 2.16 USART_E204 — IrDA Modulation and Transmission of PRS Input Data

| ***Description of Errata*** |
|---|
| If the USART IrDA modulator is configured to accept input from a PRS channel, the incoming data stream will not be transmitted because the required clock from the baud rate generator is never enabled. |
| ***Affected Conditions / Impacts*** |
| It is not possible for the USART IrDA modulator to directly transmit data from a source other than the USART's own transmitter. The USART_IRCTRL_IRPRSEN bit should remain at its reset state of 0. |
| ***Workaround*** |
| Assuming the data to be sent via the PRS is also data that could be received by the EFM32/EFR32 USART, then the data can be received using the USART's PRS RX feature (USART_INPUT_RXPRS = 1), stored in RAM (e.g. using DMA), and then transmitted with IrDA mode enabled. In cases where IrDA operation is transmit-only, the PRS RX data can be received on the same USART doing the transmission. If IrDA operation is bidirectional, then another USART must be used to receive the PRS data.<br><br>If the data to be sent is in some other format (e.g. pulses from a timer output), then there is no direct way to transmit it using the IrDA modulator. It would be necessary to capture the data in some other way and reformat it as serial data timed according to the clock generated by the USART. |
| ***Resolution*** |
| There is currently no resolution for this issue. |

# 3. Errata History

This section contains the errata history for EFR32MG13 devices.

For errata on the latest revision, refer to the beginning of this document. The device data sheet explains how to identify chip revision, either from package marking or electronically.

## 3.1 Errata History Summary

This table lists all resolved errata for the EFR32MG13.

**Table 3.1. Errata History Status Summary**

| Errata # | Designator | Title/Problem | Workaround Exists | Affected Revision | Resolution |
|---|---|---|---|---|---|
| 1 | ADC_E222 | ADC EM2 Wakeup on a Comparator Match Disables EM2 Entry | Yes | A | B |
| 2 | ADC_E224 | ADC Warm-Up Ready Can Cause IDAC, ACMP, or CSEN to Not Function | Yes | B | C |
| 3 | ADC_E225 | Using the ADC in High Accuracy Bias Mode Will Force All Analog Peripherals to High Accuracy Bias Mode | No | B | C |
| 4 | DBG_E203 | CRC Debug Command Available | No | A | B |
| 5 | EMU_E211 | Radio Clocks Remain Disabled After Voltage Scaling | Yes | A | B |
| 6 | RMU_E203 | AVDD Ramp Issue | Yes | B | C |
| 7 | TRNG_E202 | Standards Non-compliance | Yes | C | D |
| 8 | USART_E203 | DMA Can Miss USART Receive Data in Synchronous Mode | Yes | C | D |

## 3.2 Detailed Errata Descriptions

### 3.2.1 ADC_E222 – ADC EM2 Wakeup on a Comparator Match Disables EM2 Entry

***Description of Errata***

If the ADC wakes up the system from EM2 on a comparator flag match (CMPEN must be set in SINGLECTRL/SCANCTRL), the wake-up handler will not be able to clear this EM2 wakeup request. This results in the core immediately exiting EM2 on subsequent EM2 entry.

***Affected Conditions / Impacts***

Systems using the ADC comparator flag match may not be able to enter EM2.

***Workaround***

To clear the wakeup request, the wakeup handler must do one of the following:

- Disable CMPEN in the SINGLECTRL/SCANCTRL register.
- Reset the ADC FIFO.
- Continue performing conversions until an incoming conversion does not pass the CMP threshold set in CMPTHR.

Once one of these conditions has been met, the comparator can be re-enabled (if it was disabled) and the core can enter EM2.

***Resolution***

This issue is resolved in revision B devices.

### 3.2.2 ADC_E224 – ADC Warm-Up Ready Can Cause IDAC, ACMP, or CSEN to Not Function

| ***Description of Errata*** |
| --- |
| The IDAC, ACMP, or CSEN modules use use the warm up timing module in the ADC to determine when the peripherals are ready for use. However, if the ADC is enabled first, this timing module can fail to properly handshake with a low probability, causing the IDAC, ACMP, or CSEN modules to never finish warming up. The ADC is not affected by this issue and will always be available after it is enabled. |
| ***Affected Conditions / Impacts*** |
| Systems using the IDAC, ACMP, or CSEN modules in conjunction with the ADC can see intermittent failures where these modules do not operate. |
| ***Workaround*** |
| To work around this issue, enable the IDAC, ACMP, or CSEN modules before enabling the ADC. This will ensure the handshaking logic between the ADC and other modules functions correctly. |
| ***Resolution*** |
| This issue is resolved in revision C devices. |

### 3.2.3 ADC_E225 – Using the ADC in High Accuracy Bias Mode Will Force All Analog Peripherals to High Accuracy Bias Mode

| ***Description of Errata*** |
| --- |
| Using the ADC in high-accuracy bias mode (GPBIASACC in ADCn_BIASPROG cleared to 0) forces all other analog peripherals into high-accuracy bias mode. These peripherals will then draw additional current.<br>The data sheet current consumption numbers are current specified with this additional current consumption included. When the updated devices are available, the device data sheet will be updated with the reduced current consumption specifications. |
| ***Affected Conditions / Impacts*** |
| Systems using the ADC in high-accuracy bias mode may see higher current consumption than expected when other analog peripherals not using high-accuracy bias mode are in use. |
| ***Workaround*** |
| There is currently no workaround for this issue. |
| ***Resolution*** |
| This issue is resolved in revision C devices. |

### 3.2.4 DBG_E203 – CRC Debug Command Available

| ***Description of Errata*** |
| --- |
| The CRC Debug Command is always available, even when the Debug Access Port (DAP) is locked, enabling outside systems to request a CRC of the memory at any time. The memory of the device is not accessible when the DAP is locked. |
| ***Affected Conditions / Impacts*** |
| External systems may CRC the memory contents at any time. |
| ***Workaround*** |
| There is currently no workaround for this issue. |
| ***Resolution*** |
| This issue is resolved in revision B devices. |

### 3.2.5 EMU_E211 – Radio Clocks Remain Disabled After Voltage Scaling

***Description of Errata***

To avoid the radio clocks from causing issues at low voltage, hardware automatically disables the radio clocks while scaling down to Voltage Scale Level 0 (1 V). However, this lock is never released, disabling the radio until the next full reset.

***Affected Conditions / Impacts***

If the device voltage is scaled below Voltage Scale Level 2 (1.2 V), then the device scales the voltage back up to use the radio, the radio will not function.

***Workaround***

To workaround this issue, do not use voltage scaling when using the radio.

***Resolution***

This issue is resolved in revision B devices.

### 3.2.6 RMU_E203 – AVDD Ramp Issue

***Description of Errata***

The device may not properly start during power-on or restart when a voltage droop (brown out) occurs on AVDD. The failure is intermittent.

For example configurations and waveforms that are more likely to result in this issue, see the following Knowledge Base article:

http://community.silabs.com/t5/32-bit-MCU-Knowledge-Base/RMU-E203-AVDD-Ramp-Issue/ta-p/197340

To detect this failure state, place a GPIO toggle at the beginning of `main()` in the device firmware. When this failure occurs, the pin will not be toggling as expected, as the device is not executing any code.

***Affected Conditions / Impacts***

Systems may intermittently see the device fail to start, reset, or respond. The current draw of the device in this state is ~100 μA and DECOUPLE will be fully powered (~1.2 V). The device will not execute any code in this state.

***Workaround***

This issue can be resolved with a hardware workaround where an external circuit holds the reset pin low during power-on or brown out until AVDD reaches 1.8 V. For brown out, the reset pin must be configured to hard reset mode. This can be accomplished as part of the firmware image programmed to the device (lock bits area) or using the following code:

```
// Clears the CLW0 bit to enable Hard reset
void enable_hardreset()
{
  uint32_t value;
  uint32_t newvalue;
  value = *(uint32_t *)0xFE041E8;
  newvalue = value & ~(1 << 2);
  MSC_WriteWord((uint32_t *)0xFE041E8, &newvalue, 4);
}
```

There is currently no software workaround for all potential failure mechanisms. The software workaround included in the Knowledge Base article will prevent failure in some scenarios. See the Knowledge Base article for more information:

http://community.silabs.com/t5/32-bit-MCU-Knowledge-Base/RMU-E203-AVDD-Ramp-Issue/ta-p/197340

***Resolution***

This issue is resolved in revision C devices.

### 3.2.7 TRNG_E202 — Standards Non-compliance

| ***Description of Errata*** |
| --- |
| The TRNG module may either fail to generate random numbers or generate random numbers with AIS-31 error flags. This is because the TRNG has 2 potential issues:<br>• Non-Compliance with NIST SP800-90B<br>The TRNG entropy source may not be sufficient to pass the start-up tests and will not place any data in the FIFO.<br>• Non-Compliance with AIS-31<br>The TRNG entropy source may be sufficient to pass the start-up tests, but insufficient to pass the AIS-31 test. It will place some data in the FIFO and indicate an AIS-31 error.<br>In both cases, the TRNG will cause the mbed TLS random number generator to return an error code and no data.<br>The TRNG module, therefore, is non-functional and should not be used. |
| ***Affected Conditions / Impacts*** |
| Application software that uses the mbed TLS random number generator may return no data either with or without an error code. Software that accesses the TRNG module directly by using the public CMSIS registers will either receive no data or data with AIS-31 error flags. |
| ***Workaround*** |
| There is no workaround that is NIST SP800-90B or AIS-31 compliant. Silicon Labs has implemented the use of alternative entropy sources to add entropy to the mbedTLS entropy collector in Gecko SDK version 2.3. The TRNG module should not be used directly. Customer application code should rather use the mbedTLS entropy collector API in Gecko SDK version 2.3 and later. |
| ***Resolution*** |
| This issue is resolved in revision D devices. |

### 3.2.8 USART_E203 — DMA Can Miss USART Receive Data in Synchronous Mode

| ***Description of Errata*** |
| --- |
| If the USART is operating in synchronous mode, it can drop received data before the DMA has a chance to read it under the following conditions:<br>• Synchronous master sample delay is enabled (USARTn_CTRL_SMSDELAY = 1) in order to improve timing at higher clock rates.<br>• The receive FIFO is already full, and the receive data DMA request (USARTnRXDATAV) is asserted.<br>• The transmit shift register is clocking out the last frame to be sent, the transmit FIFO is empty, and the transmit data DMA request (USARTnTXBL) is asserted.<br>• The transmit data DMA request arrives before the receive data DMA request (the transmit FIFO empties before the receive data DMA request is asserted).<br>• A higher priority peripheral DMA request arrives while processing the transmit data DMA request but before the receive data DMA request is processed.<br>Because the incoming peripheral DMA request has higher priority than the USART DMA requests but cannot interrupt a DMA request that is already in progress (the transmit data DMA request), it will be processed before the receive data DMA request, thus causing the USART to drop an incoming frame (or frames) since the receive FIFO is already full. |
| ***Affected Conditions / Impacts*** |
| In systems that use the USART in synchronous mode with the master sample delay feature (USARTn_CTRL_SMSDELAY = 1) and that use the DMA to manage both the USART transmitter and receiver, as well as other peripherals with higher request priorities, the USART can drop an incoming frame (or frames) if the DMA is not able to process the receive data requests to empty the receive FIFO when it is full. |
| ***Workaround*** |
| Assign a higher priority to the DMA channel servicing the receive data DMA requests such that it is processed before the channel servicing transmit data DMA requests and any channels servicing requests associated with any other peripherals that could potentially stall a USART synchronous transfer that is already in progress. Set LDMA_CHx_CTRL_IGNORESREQ = 1 for the transmit data channel so that the LDMA accumulates multiple requests from the transmitter and services them with a single transfer cycle. This causes the LDMA to fill the USART transmitter's FIFO only when it is empty instead of each and every time space becomes available. |
| ***Resolution*** |
| This issue is resolved in revision D devices. |

# 4. Revision History

**Revision 0.8**

May, 2019

- Updated the latest revision to revision D.
- TRNG_E202 and USART_E203 resolved and moved to 3. Errata History.

**Revision 0.7**

November, 2018

- Added ADC_E228, CSEN_E201, CSEN_E202, EMU_E214, I2C_E202, I2C_E203, I2C_E204, I2C_E205, I2C_E206, USART_E203, and USART_E204.

**Revision 0.6**

May, 2018

- Added TRNG_E202.

**Revision 0.5**

March, 2018

- Updated the workaround in RMU_E202.

**Revision 0.4**

January, 2018

- Removed PLFRCO_E201.

**Revision 0.3**

August, 2017

- Updated the latest revision to revision C.
- Added PLFRCO_E201.

**Revision 0.2**

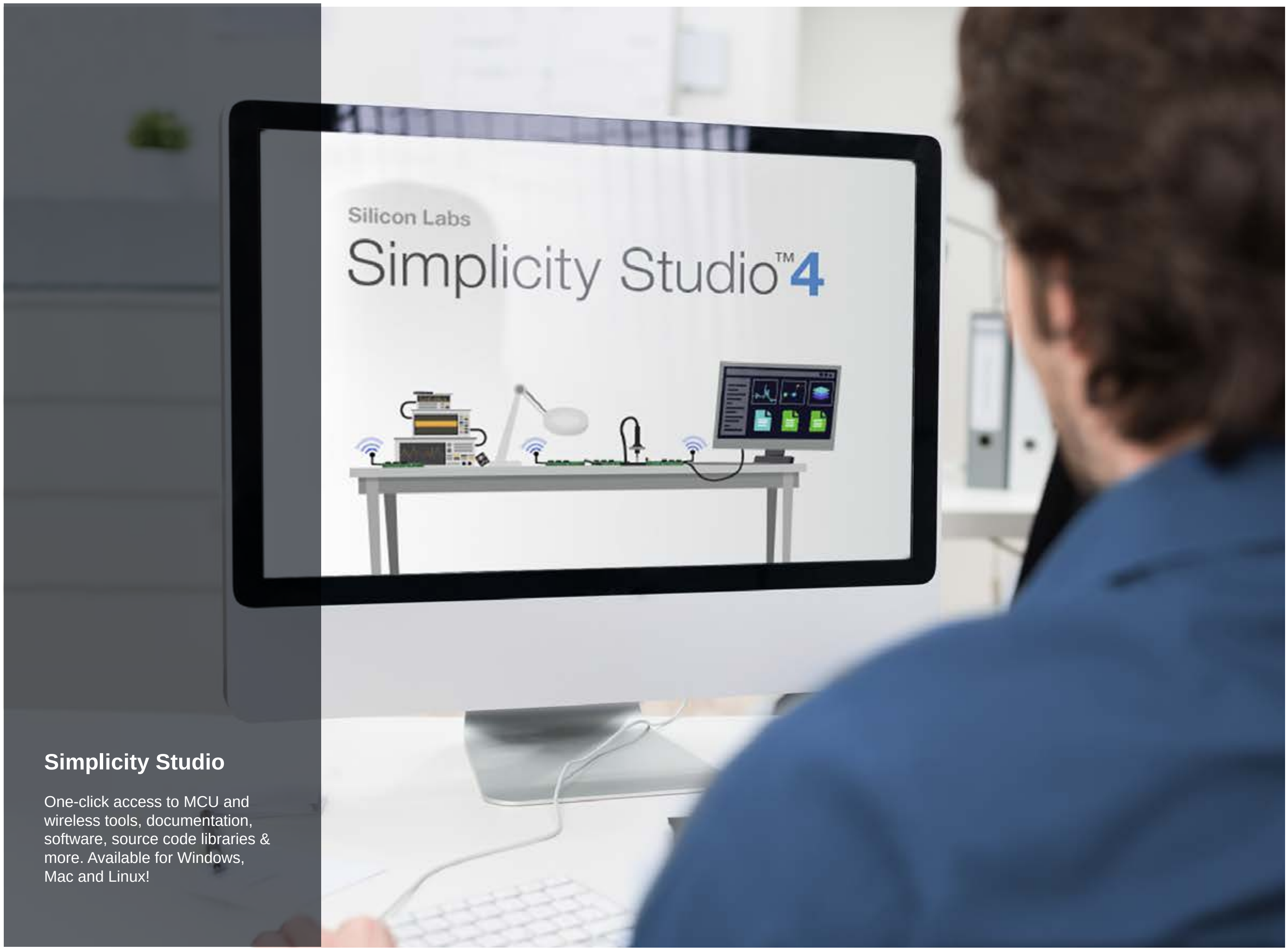
May, 2017

- Added ADC_E224, ADC_E225, DBG_E204, RMU_E202, and RMU_E203.

**Revision 0.1**

November, 2016

- Initial release.




**IoT Portfolio**
*www.silabs.com/IoT*

**SW/HW**
*www.silabs.com/simplicity*

**Quality**
*www.silabs.com/quality*

**Support and Community**
*community.silabs.com*



**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**http://www.silabs.com**