EFR32 Wireless Gecko
EFR32FG14 Errata

This document contains information on the EFR32FG14 errata. The latest available revision of this device is revision B.

Errata that have been resolved remain documented and can be referenced for previous revisions of this device.

The device data sheet explains how to identify the chip revision, either from the package marking or electronically.

Errata effective date: August, 2025.

1. Errata Summary

The following table lists all the known and unresolved errata for the EFR32FG14.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision:	
			B	
ADC_E213	ADC KEEPINSLOWACC Mode	No	X	
ADC_E228	Limited ADC Sampling Frequency in EM2	No	X	
CUR_E205	Elevated Current Consumption in EM4H and EM4S when SWCLK and SWDIO MODEn is Disabled	No	X	
DBG_E204	Debug Recovery with JTAG Does Not Work	Yes	X	
EMU_E214	Device Erase Cannot Occur if Voltage Scaling Level is Too Low	Yes	X	
EMU_E220	DECBOD Reset During Voltage Scaling After EM2 or EM3 Wakeup	Yes	X	
I2C_E202	Race Condition Between Start Detection and Timeout	Yes	X	
I2C_E203	I2C Received Data Can be Shifted	Yes	X	
I2C_E205	Go Idle Bus Idle Timeout Does Not Bring Device to Idle State	Yes	X	
I2C_E206	Slave Holds SCL Low After Losing Arbitration	Yes	X	
I2C_E207	I2C Fails to Indicate New Incoming Data	Yes	X	
LES_E201	LFPRESC Can Extend Channel Start-Up Delay	Yes	X	
RMU_E202	External Debug Access Not Available After Watchdog or Lockup Full Reset	Yes	X	
TIMER_E202	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X	
USART_E204	IrDA Modulation and Transmission of PRS Input Data	Yes	X	
USART_E205	Possible Data Transmission on Wrong Edge in Synchronous Mode	Yes	X	
USART_E206	Additional SCLK Pulses Can Be Generated in USART Synchronous Mode	Yes	X	
USART_E207	PRS Transmit Unavailable in Synchronous Slave Mode	No	X	
WDOG_E201	Clear Command is Lost Upon EM2 Entry	Yes	X	
WTIMER_E201	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X	

2. Current 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 CUR_E205 – Elevated Current Consumption in EM4H and EM4S when SWCLK and SWDIO MODEn is Disabled

Description of Errata
In EM0, EM1, EM2, and EM3, the input buffer for the SWCLK and SWDIO pins are disabled when MODEn is DISABLED for these pins, as expected. However, in EM4H and EM4S, the input buffer for the SWCLK and SWDIO pins are enabled by hardware and cannot be disabled by firmware. As a result, when MODEn is set to DISABLED for SWCLK and SWDIO and the device enters EM4H or EM4S, current consumption is elevated. To minimize EM4H/S current consumption, MODEn for these two pins should not be configured as DISABLED prior to EM4H/S entry. Instead, SWCLK should be configured as MODEn = INPUTPULL with pull-down enabled, and SWDIO should be configured as MODEn = INPUTPULL with pull-up enabled.
Affected Conditions / Impacts
Systems that use EM4H or EM4S may see elevated EM4H/EM4S current consumption up to approximately 100 uA when SWCLK and SWDIO MODEn is DISABLED.
Workaround
There is no known workaround for this issue
Resolution
There is currently no resolution for this issue.

2.4 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.

2.5 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 remain locked.
Workaround
The voltage should be set to Voltage Scale Level 2 (1.2 V) before executing the device erase. 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). 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.

2.6 EMU_E220 – DECBOD Reset During Voltage Scaling After EM2 or EM3 Wakeup

Description of Errata
An infrequent, asynchronous and unrelated internal event can intermittently delay normal BOD state-machine transition sequencing during voltage scaling from VSCALE0 (1.0 Vdc) to VSCALE2 (1.2 Vdc) when emerging from EM2/EM3 to EM0. This delay can cause erroneous DECBOD resets on some devices.
Affected Conditions / Impacts
Systems operating with core voltage scaling can experience a decouple voltage brownout reset (DECBOD) when exiting EM2 or EM3.
Workaround
Systems that use core voltage scaling need to enter EM2 or EM3 via a RAM executed wait for interrupt instruction with interrupts disabled. After wakeup and before voltage scaling, the system should delay for 14 µs, then check if an EMU calibration is active, and if so, wait for it to complete. **Note:** This workaround is included in `em_emu.c` in the v4.5 or later of the Gecko SDK. It is recommended to workaround this issue by using the latest Gecko SDK version.

```
#define EMU_STATUS_CALIBRATION  (0x1UL << 28)
// Execute from RAM with interrupts disabled
__WFI();
ERRATA_FIX_EMU_E220_DELAY_CYCLES(); //delay 14 microseconds
while (EMU->STATUS & EMU_STATUS_CALIBRATION);
```

Resolution
There is currently no resolution for this issue.

2.7 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: • 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. • The SSTOPIF interrupt flag indicates that the I2C bus has returned to the idle state. • 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.

2.8 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. There are two ways in which the falling condition on SDA can potentially happen: • 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. • 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.

2.9 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 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.

2.10 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.

2.11 I2C_E207 – I^2C Fails to Indicate New Incoming Data

Description of Errata
A race condition exists in which the I^2C fails to indicate reception of new data when both user software attempts to read data from and the I^2C hardware attempts to write data to the I2C_RXFIFO in the same cycle.
Affected Conditions / Impacts
When this race condition occurs, the RXFIFO enters an invalid state in which both I2C_STATUS_RXDATAV = 0 and I2C_STATUS_RXFULL = 1. This causes the I^2C to discard new incoming data bytes because RXFULL = 1 and would otherwise prevent user software from reading last byte written by the I^2C hardware to RXFIFO because RXDATAV = 0.
Workaround
User software can recognize and clear this invalid RXDATAV = 0 and RXFULL = 1 condition by performing a dummy read of the RXFIFO (I2C_RXDATA). This restores the expected RXDATAV = 1 and RXFULL = 0 condition. The dummy read will also set the RXUFIF flag bit, which should be ignored and cleared. The data from this read can be discarded, and user software can now read the last byte written by the I^2C hardware to the RXFIFO (the byte which caused the invalid RXDATAV = 0 and RXFULL = 1 condition). No data will be lost as long as user software completes this recovery procedure (performing the dummy read and then reading the remaining valid byte in the RXFIFO) before the I^2C hardware receives the next incoming data byte.
Resolution
There is currently no resolution for this issue.

2.12 LES_E201 — LFPRESC Can Extend Channel Start-Up Delay

Description of Errata
Setting LESENSE_TIMCTRL_LFPRESC to a value other than DIV1 may delay channel start-up longer than the number of $LFACLK_{LESENSE}$ clock cycles specified by LESENSE_TIMCTRL_STARTDLY.
Affected Conditions / Impacts
Delaying channel start-up delays the subsequent excitation and measurement phases and may have an impact on the data returned by the LESENSE.
Workaround
If a channel start-up delay is used (LESENSE_TIMCTRL_STARTDLY > 0), LESENSE_TIMCTRL_LFPRESC must be set to DIV1.
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: • Software should configure peripherals to either LIMITED or EXTENDED mode if full debugger functionality is needed after a watchdog or lockup reset. • When using FULL reset mode, appending at least 9 idle clock cycles to the last debug command will allow the transaction to complete. • A power cycle or hard pin reset will restore normal operation.
Resolution
There is currently no resolution for this issue.

2.14 TIMER_E202 — Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode

Description of Errata
When the TIMER is configured to operate in quadrature decoder mode with the overflow interrupt enabled and the counter value (TIMER_CNT) reaches the top value (TIMER_TOP), the overflow interrupt is requested continuously even if the interrupt flag (TIMER_IF_OF) is cleared. Similarly, if the underflow interrupt is enabled and the counter value reaches zero, the underflow interrupt is requested continuously even if the interrupt flag (TIMER_IF_UF) is cleared. The interrupt can be cleared only after the counter value has incremented or decremented so that the overflow or underflow condition no longer applies.
Affected Conditions / Impacts
Because the counter is clocked by its CC0 and CC1 inputs in quadrature decoder mode and not the prescaled HFPERCLK, overflow and underflow events remain latched as long as TIMER_CNT remains at the value that triggered the overflow or underflow condition. Until the counter is no longer in the overflow or underflow condition, it is not possible to clear the associated interrupt flag.
Workaround

Short of disabling the relevant interrupts, the simplest workaround is to manually change TIMER_CNT so that the overflow or underflow condition no longer exists. Insert the following or similar code in the interrupt handler for the timer in question (TIMER0 in this case) to do this:
`uint32 intFlags = TIMER_IntGet(TIMER0);`
 	
`if((intFlags & TIMER_IF_OF) && (TIMER0->CNT == TIMER0->TOP))`
`  TIMER0->CNT = 0;`
 	
`if((intFlags & TIMER_IF_UF) && (TIMER0->CNT == 0x0))`
`  TIMER0->CNT = TIMER0->TOP;`
 	
It may be necessary for firmware to account for this adjustment in calculations that include the counter value.

Resolution
There is currently no resolution for this issue.

2.15 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. 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.

2.16 USART_E205 — Possible Data Transmission on Wrong Edge in Synchronous Mode

Description of Errata
If the USART is configured to operate in synchronous mode with... 1. USART_CLKDIV_DIV = 0 (clock = $f_{HFPERCLK} \div 2$) 2. USART_CTRL_CLKPHA = 0 3. USART_TIMING_CSHOLD = 1 ...and data is loaded into the transmit FIFO (say, by the LDMA) at the exact same time as the USART state machine begins to insert the requested one bit time extension of chip select hold time (USART_TIMING_CSHOLD = 1), the first bit of the new data word is incorrectly transmitted on the leading clock edge of the subsequent data bit and not the trailing clock edge of the current data bit.
Affected Conditions / Impacts
Reception of each data bit by the slave is tied to a specific clock edge, thus the late transmission by the master of the first bit of a word may cause the slave to receive the incorrect data, especially if the data setup time for the slave approaches or exceeds one half the shift clock period.
Workaround
Because there is no way to specifically time a write to the transmit FIFO such that it does not occur when the USART state machine changes state, use one of the following workarounds to avoid the risk for data corruption described above: • Set USART_CLK_DIV > 0. • Use USART_TIMING_CSHOLD = 0 or USART_TIMING_CSHOLD > 1. • Use USART_CTRL_CLKPHA = 1. This option is particularly useful with SPI flash memories as many support operations in both the CLKPOL = CLKPHA = 0 and CLKPOL = CLKPHA = 1 modes.
Resolution
There is currently no resolution for this issue.

2.17 USART_E206 — Additional SCLK Pulses Can Be Generated in USART Synchronous Mode

Description of Errata
When inter-character spacing is enabled (USART_TIMING_ICS > 0) and USART_CTRL_CLKPHA = 1 in synchronous master mode, an extra clock pulse is generated after each frame transmitted except the last (that frame which when sent results in both the transmit FIFO and transmit shift register being empty).
Affected Conditions / Impacts
The extra clock pulse generated at the end of the first frame would cause a slave device to clock in the first bit of the next frame it expects to receive even though the USART is not yet driving that data. The slave would lose synchronization with the master and erroneously receive all frames after the first.
Workaround
Do not enable inter-character spacing when CLKPHA = 1. If a delay between frames is necessary, insert one manually with a software delay loop. Data cannot be transmitted using DMA in this case.
Resolution
There is currently no resolution for this issue.

2.18 USART_E207 — PRS Transmit Unavailable in Synchronous Slave Mode

Description of Errata
When the USART is configured for synchronous slave operation, the transmit output (MISO) is not driven if the signal is routed to a pin using the PRS producer (for example, SOURCESEL = 0x10 and SIGSEL = 0x5 for USART0 on some devices).
Affected Conditions / Impacts
Systems cannot operate the USART in synchronous slave mode if the PRS is used to route the transmit output to the RX (MISO) pin. Operation is not affected in master mode when the transmit output is routed to the TX (MOSI) pin using the PRS producer nor is operation affected in any mode when the USARTn_ROUTELOC0_RXLOC and USARTn_ROUTELOC0_TXLOC bit fields are used.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.19 WDOG_E201 – Clear Command is Lost Upon EM2 Entry

Description of Errata
If the device enters EM2 while the clear command is still being synchronized, the watchdog counter may not be cleared as expected.
Affected Conditions / Impacts
If the watchdog counter is not cleared as expected, the device can encounter a watchdog reset.
Workaround
Wait for WDOG_SYNCBUSY_CMD to clear before entering EM2. Note that WDOG can be clocked from one of the low-frequency clock sources and will require additional time to enter EM2 when implementing this workaround.
Resolution
There is currently no resolution for this issue.

2.20 WTIMER_E201 — Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode

Description of Errata
When the WTIMER is configured to operate in quadrature decoder mode with the overflow interrupt enabled and the counter value (WTIMER_CNT) reaches the top value (WTIMER_TOP), the overflow interrupt is requested continuously even if the interrupt flag (WTIMER_IF_OF) is cleared. Similarly, if the underflow interrupt is enabled and the counter value reaches zero, the underflow interrupt is requested contiunously even if the interrupt flag (WTIMER_IF_UF) is cleared. Only after the counter value has incremented or decremented so that the overflow or underflow condition no longer applies can the interrupt be cleared.
Affected Conditions / Impacts
Because the counter is clocked by its CC0 and CC1 inputs in quadrature decoder mode and not the prescaled HFPERCLK, overflow and underflow events remain latched as long WTIMER_CNT remains at the value that triggered the overflow or underflow condition. Until the counter is no longer in the overflow or underflow condition, it is not possible to clear the associated interrupt flag.
Workaround

Short of disabling the relevant interrupts, the simplest workaround is to manually change WTIMER_CNT so that the overflow or underflow condition no longer exists. Insert the following or similar code in the interrupt handler for the timer in question (WTIMER0 in this case) to do this:
`uint32 intFlags = TIMER_IntGet(WTIMER0);`
 	
`if((intFlags & WTIMER_IF_OF) && (WTIMER0->CNT == WTIMER0->TOP))`
`  WTIMER0->CNT = 0;`
 	
`if((intFlags & WTIMER_IF_UF) && (WTIMER0->CNT == 0x0))`
`  WTIMER0->CNT = WTIMER0->TOP;`
 	
It may be necessary for firmware to account for this adjustment in calculations that include the counter value.

Resolution
There is currently no resolution for this issue.

3. Revision History

Revision 0.7

August, 2025

- Updated workaround for EMU_E220.
- Added CUR_E205.
- Clarified workaround for I2C_E207.

Revision 0.6

September, 2020

- Added I2C_E207, USART_E206 and WDOG_E201.

Revision 0.5

April, 2020

- Added EMU_E220.

Revision 0.4

January, 2020

- Added I2C_E202, I2C_E203, I2C_E205, LES_E201, TIMER_E202, USART_E204, USART_E205, and WTIMER_E201.
- Migrated to new errata document format.

Revision 0.3

August, 2018

- Added EMU_E214 and I2C_E206.
- Removed TRNG_E202.

Revision 0.2

May, 2018

- Added TRNG_E202.
- Updated the workaround in RMU_E202.

Revision 0.1

October, 2017

- Initial release.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com