Wireless Gecko™ EFR32FG25 Errata

This document contains information on the EFR32FG25 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: June, 2025.

1. Errata Summary

The following table lists all the known and unresolved errata for the EFR32FG25.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision	
			A	B
CUR_E302	Extra EM1 Current if FPU is Disabled	Yes	X	—
CUR_E303	Active Charge Pump Clock Causes High Current	Yes	X	—
DCDC_E301	Incorrect IPVERSION Register Value	Yes	X	X
DCDC_E302	DCDC Interrupts Block EM2/3 Entry or Cause Unexpected Wake-up	Yes	X	—
ETAMPDET_E301	E-Tamper Detect Malfunction on Writes to Upper Prescaler	Yes	X	—
EUSART_E302	Synchronous EUSART Module Disable Lockup	Yes	X	X
EUSART_E303	EUSART Receiver Enters Lockup State when Using Low Frequency IrDA Mode	Yes	X	X
EUSART_E304	Incorrect Stop Bits Lock Receiver	Yes	X	X
IADC_E306	Changing Gain During a Scan Sequence Causes an Erroneous IADC Result	Yes	X	X
PLL0_E301	USB PLL0 High Current Draw Causes Unreliable USB Functionality	No	X	—
SE_E302	DPA Countermeasure Unavailable for Some Operations	Yes	X	X

2. Current Errata Descriptions

2.1 DCDC_E301 – Incorrect IPVERSION Register Value

Description of Errata
The value returned by the IPVERSION register is incorrect and does not match the actual hardware implementation.
Affected Conditions / Impacts
Because the register addresses associated with the IPVERSION value do not match the actual hardware implementation: • writes to DCDC registers will cause incorrect module behavior, and • reads of DCDC registers will not return expected values.
Workaround
Software should use IPVERSION in conjunction with device identification data to determine the specific DCDC module implementation. Device identification can be performed by decoding the contents of the DEVINFO_PART register or by using emlib APIs, such as `SYSTEM_ChipRevisionGet()` and `SYSTEM_GetPartNumber()`.
Resolution
There is currently no resolution for this issue.

2.2 EUSART_E302 — Synchronous EUSART Module Disable Lockup

Description of Errata
The EUSART freezes and does not function if firmware: 1. Initializes the EUSART in synchronous main mode. 2. Disables the EUSART and reconfigures it to either synchronous secondary or asynchronous mode. 3. Re-enables the EUSART. 4. Transfers data. 5. Disables the EUSART. A handshake signal fails to fully propogate through the EUSART disable logic when leaving synchronous main mode.
Affected Conditions / Impacts
Systems that use the EUSART in synchronous main mode cannot simply switch to another mode because this causes the module to freeze. This issue occurs only when firmware attempts to switch from synchronous main mode to another mode. Switching between all other modes is unaffected.
Workaround

Firmware can manually generate additional clock edges after the module is disabled to fully propagate the handshake signal and allow the next disable sequence to happen as usual.
Example code
`//Work-around code//`
`uint32_t i;`
`for (i=0;i<4;i++) {`
`  EUSART0->CFG2 |= EUSART_CFG2_CLKPHA;`
`  EUSART0->CFG2 &= ~EUSART_CFG2_CLKPHA;`
`}`
`//Work-around code - END//`

Resolution
There is currently no resolution for this issue.

2.3 EUSART_E303 — EUSART Receiver Enters Lockup State when Using Low Frequency IrDA Mode

Description of Errata
When low frequency IrDA mode is enabled (EUSART_IRLFCFG_IRLFEN = 1), the receiver can block incoming traffic if it receives either a… • 0 if EUSART_CFG0_RXINV = 0 or • 1 if EUSART_CFG0_RXINV = 1 …before… • the EUSART module is enabled (EUSART_EN_EN =1), • the receiver is enabled (EUSART_CMD_RXEN =1), and • the write to enable the receiver (RXEN = 1) has been synchronized (EUSART_SYNCBUSY_RXEN = 0).
Affected Conditions / Impacts
Incoming traffic will be blocked at the EUSART receiver. Subsequent interrupts and status flags will not be set correctly.
Workaround

To avoid entering the lockup state, use one of the workarounds mentioned below:

- When the receiver (RX) input is routed through the PRS:

  Force the input to the IrDA demodulator to high by using the PRS before enabling EUSART. Keep it this way until the receiver has been enabled and the EUSART_CMD_RXEN bit is synchronized. See the following code sequence for an example of how to do this:

```
// Output logic 0 through PRS Channel that is connected to EUSART RX GPIO
PRS->ASYNC_CH[0].CTRL = PRS_ASYNC_CH_CTRL_FNSEL_LOGICAL_ZERO |
                    PRS_ASYNC_CH_CTRL_SOURCESEL_GPIO | PRS_ASYNC_CH_CTRL_SIGSEL_GPIOPIN0;

// Select PRS as input to RX.
EUSART0->CFG1_SET = EUSART_CFG1_RXPRSEN;

// Enable EUSART to configure Rx
EUSART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUSART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUSART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Output EUSART RX pin through PRS Channel
PRS->ASYNC_CH[0].CTRL = (PRS->ASYNC_CH[0].CTRL & ~_PRS_ASYNC_CH_CTRL_FNSEL_MASK) |
                                    PRS_ASYNC_CH_CTRL_FNSEL_A;
```

  Note: For proper IrDA RZI operation, the receiver input must be inverted, so EUSART_CTRL_RXINV = 1 in this workaround.

- When the receiver (RX) input is not routed through the PRS:

  Force the input to the IrDA demodulator to high by using a GPIO pin other than the current EUSART RX pin before enabling the EUSART. Keep it this way until the receiver has been enabled and the EUSART_CMD_RXEN bit is synchronized. See the following code sequence for an example of how to do this:

```
// Configure alternate GPIO (PA00) used for workaround to output 0
GPIO_PinModeSet(gpioPortA, 0, gpioModePushPull, 0);

// Route EUSART0 Rx to the alternate GPIO (PA00)
GPIO->EUSARTROUTE[0].ROUTEEN = (GPIO->EUSARTROUTE[0].ROUTEEN & ~GPIO_EUSART_ROUTEEN_RXPEN);
GPIO->EUSARTROUTE[0].RXROUTE = (gpioPortA << _GPIO_EUSART_RXROUTE_PORT_SHIFT) | (0 <<
_GPIO_EUSART_RXROUTE_PIN_SHIFT);
GPIO->EUSARTROUTE[0].ROUTEEN |= GPIO_EUSART_ROUTEEN_RXPEN;

// Enable EUSART0 to configure Rx
EUSART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUSART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUSART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Route EUSART Rx to EUSART_RX GPIO(EUSART_RX_PORT & EUSART_RX_PIN)
GPIO->EUSARTROUTE[0].ROUTEEN = (GPIO->EUSARTROUTE[0].ROUTEEN & ~GPIO_EUSART_ROUTEEN_RXPEN);
GPIO->EUSARTROUTE[0].RXROUTE = (EUSART_RX_PORT << _GPIO_EUSART_RXROUTE_PORT_SHIFT) | (EUSART_RX_PORT <<
_GPIO_EUSART_RXROUTE_PIN_SHIFT);
GPIO->EUSARTROUTE[0].ROUTEEN |= GPIO_EUSART_ROUTEEN_RXPEN;

// Disable alternate GPIO (PA00) used for workaround
GPIO_PinModeSet(gpioPortA, 0, gpioModeDisabled, 0);
```

  Note: For proper IrDA RZI operation, the receiver input must be inverted, so EUSART_CTRL_RXINV = 1 in this workaround.

To exit the lockup state, disable the EUART and force the input to the IrDA demodulator to 1 before re-enabling the EUART by using steps mentioned above.

Resolution

There is currently no resolution for this issue.

2.4 EUSART_E304 — Incorrect Stop Bits Lock Receiver

Description of Errata
When low frequency IrDA mode is enabled (EUSART_IRLFCFG_IRLFEN = 1), the receiver can block incoming traffic if it receives either a… • 0 if EUSART_CFG0_RXINV = 0 or • 1 if EUSART_CFG0_RXINV = 1 …when it is expecting a stop bit.
Affected Conditions / Impacts
Incoming traffic will be blocked at the EUSART receiver. Subsequent interrupts and status flags will not be set correctly.
Workaround

To avoid receiver lock-up in the application firmware caused by formatting errors in the received data, change the receiver GPIO pin routing to force the input to the IrDA demodulator to 1 for the anticipated period of time during which such data can be received.

To exit the lockup state, disable the EUSART and force the input to the IrDA demodulator to 1 before re-enabling the EUSART by using one of the workarounds mentioned below:

- When the receiver (RX) input is routed through the PRS:

  Force the input to the IrDA demodulator to high by using the PRS before enabling EUSART. Keep it this way until the receiver has been enabled and the EUSART_CMD_RXEN bit is synchronized. See the following code sequence for an example of how to do this:

```
// Output logic 0 through PRS Channel that is connected to EUSART RX GPIO
PRS->ASYNC_CH[0].CTRL = PRS_ASYNC_CH_CTRL_FNSEL_LOGICAL_ZERO |
                        PRS_ASYNC_CH_CTRL_SOURCESEL_GPIO | PRS_ASYNC_CH_CTRL_SIGSEL_GPIOPIN0;

// Select PRS as input to Rx
EUSART0->CFG1_SET = EUSART_CFG1_RXPRSEN;

// Enable EUSART to configure Rx
EUSART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUSART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUSART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Output EUSART RX through PRS Channel
PRS->ASYNC_CH[0].CTRL = (PRS->ASYNC_CH[0].CTRL & ~_PRS_ASYNC_CH_CTRL_FNSEL_MASK) |
                        PRS_ASYNC_CH_CTRL_FNSEL_A;
```

  Note: For proper IrDA RZI operation, the receiver input must be inverted, so EUSART_CTRL_RXINV = 1 in this workaround.

- When the receiver (RX) input is not routed through the PRS:
  Force the input to the IrDA demodulator to high by using a GPIO pin other than the current EUSART RX pin before enabling the EUSART. Keep it this way until the receiver has been enabled and the EUSART_CMD_RXEN bit is synchronized. See the following code sequence for an example of how to do this:

```
// Configure alternate GPIO (PA00) used for workaround to output 0
GPIO_PinModeSet(gpioPortA, 0, gpioModePushPull, 0);

// Route EUSART0 Rx to the alternate GPIO (PA00)
GPIO->EUSARTROUTE[0].RXROUTE = (gpioPortA << _GPIO_EUSART_RXROUTE_PORT_SHIFT) | (0 <<
_GPIO_EUSART_RXROUTE_PIN_SHIFT);

// Enable EUSART0 to configure Rx
EUSART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUSART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUSART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Route EUSART Rx to EUSART_RX GPIO(EUSRT_RX_PORT & EUSART_RX_PIN)
GPIO->EUSARTROUTE[0].RXROUTE = (EUSART_RX_PORT << _GPIO_EUSART_RXROUTE_PORT_SHIFT) | (EUSART_RX_PIN <<
_GPIO_EUSART_RXROUTE_PIN_SHIFT);

// Disable alternate GPIO (PA00) used for workaround
GPIO_PinModeSet(gpioPortA, 0, gpioModeDisabled, 0);
```

  Note: For proper IrDA RZI operation, the receiver input must be inverted, so EUSART_CTRL_RXINV = 1 in this workaround.

Resolution

There is currently no resolution for this issue.

2.5 IADC_E306 – Changing Gain During a Scan Sequence Causes an Erroneous IADC Result

Description of Errata
Differences in the ANALOGGAIN setting within multiple IADC_CFGx groups during a scan sequence introduces a transient condition that may result in an inaccurate IADC conversion.
Affected Conditions / Impacts
The result of the IADC scan measurement may not match the expected result for the voltage present on the pin during the conversion.
Workaround
Both 1 and 2 shown below must be implemented. 1. If there is a difference in the ANALOGGAIN setting between IADC_CFGx groups during a scan sequence, the IADC_SCHEDx clock prescaler must also change to an appropriate setting. This forces a warmup state (5 µs delay) in between ANALOGGAIN changes. Note that the same IADC_SCHEDx clock prescaler value may be an appropriate setting for both ANALOGGAIN settings, but to force the warmup delay, the IADC_SCHEDx must have different values. 2. The first and last entry of a scan group should use IADC_CFG0, which is the default configuration of the IADC at the start and end of a scan conversion sequence. If CONFIG1 is used at the start and end of the scan group, erroneous IADC results may occur.
Resolution
There is currently no resolution for this issue.

2.6 SE_E302 – DPA Countermeasure Unavailable for Some Operations

Description of Errata
Differential power analysis (DPA) countermeasures for ECDH on Curve25519, ECDH on Curve448, and EdDSA signing on Curve25519 are unavailable due to a lack of hardware support on all Series 2 devices with a Hardware Secure Engine (HSE).
Affected Conditions / Impacts
A successful DPA attack may be possible if the impacted algorithms are implemented in a customer's product. However, a DPA attack is not an easy/straightforward attack as it requires specific equipment, many traces, physical access to the device, and some control over device operation. If a successful DPA attack occurs, an attacker may be able to gain access to confidential information, such as private keys or encrypted communications between devices.
Workaround
No fix is available to provide the affected DPA countermeasures on Series 2 devices. Refer to Security Advisory A-00000534 for mitigation recommendations, which include refreshing key pairs or using a key pair only once to reduce the risk of a successful DPA attack.
Resolution
There is currently no resolution for this issue.

3. Resolved Errata Descriptions

This section contains previous errata for EFR32FG25 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 the package marking or electronically.

3.1 CUR_E302 – Extra EM1 Current if FPU is Disabled

Description of Errata
When the Floating Point Unit (FPU) is disabled, the on-demand Fast Startup RC Oscillator (FSRCO) remains on after an energy mode transition from EM0 to EM1 is complete. This leads to higher current consumption in EM1.
Affected Conditions / Impacts
The enabled FSRCO increases EM1 current consumption by approximately 500 µA.
Workaround
Always enable the FPU at the beginning of code execution via the Coprocessor Access Control Register (CPACR) in the System Control Block (SCB) as shown below: `SCB->CPACR |= ((3 << 20) | (3 << 22));`
Resolution
This issue is resolved on revision B devices.

3.2 CUR_E303 – Active Charge Pump Clock Causes High Current

Description of Errata
When the ACMP0, ACMP1, or IADC0 peripherals are active, the clock to the internal analog mux charge pump may also be activated, resulting in extra supply current.
Affected Conditions / Impacts
• ACMP0 and ACMP1: The charge pump clock is activated whenever either module is enabled via the ACMPn_EN_EN bit or when enabled by the LESENSE state machine. • IADC0: The charge pump clock is activated when any portion of the IADC analog circuitry is on. When IADC_CTRL_WARMUPMODE = KEEPINSTANDBY or KEEPWARM, the clock is activated as long as the IADC is enabled via the IADC_EN_EN bit. When IADC_CTRL_WARMUPMODE = NORMAL, the clock is activated only during warmup and conversion and will be shut down between conversions. • The extra current is from a shared block and increases supply current by an approximate total of 25 µA when any of the above conditions are true.
Workaround
No workaround exists to entirely eliminate the extra current. The impact of the current can be reduced by duty-cycling the peripheral. The average system supply current increase depends on the total percentage of time the peripheral(s) is/are active. For example, if only ACMP0 is used and enabled for 10% of the time, the average supply current increase is about 2.5 µA.
Resolution
This issue is resolved on revision B devices.

3.3 DCDC_E302 – DCDC Interrupts Block EM2/3 Entry or Cause Unexpected Wake-up

Description of Errata
Regardless of the setting of the DCDC Interrupt Enable (DCDC_IEN) register, if the DCDC interrupt is enabled in the NVIC, the BYPSW, WARM, RUNNING, or TMAX interrupt requests can wake the device from EM2/3 or prevent it from entering EM2/3.
Affected Conditions / Impacts
The errata is limited to the BYPSW, WARM, RUNNING, or TMAX requests as reflected in the DCDC Interrupt Flag (DCDC_IF) register, which also function as wake-up sources from EM2/3. When the NVIC DCDC interrupt is enabled: • If the corresponding DCDC_IEN bit for one of these interrupt requests is 1 and that condition occurs, then an interrupt **will** occur, and the CPU will branch to the DCDC IRQ handler. • If the corresponding DCDC_IEN bit for one of these interrupt requests is 0 and that condition occurs, then an interrupt **will not** occur. • If any one of these four interrupt conditions occurs, regardless of the setting of its corresponding DCDC_IEN bit, the device **will** wake from EM2/3 and/or be prevented from entering EM2/3. If the corresponding IEN is 0, an interrupt **will not** occur even though the EM2/3 wakeup event has occurred.
Workaround
To prevent unwanted wake-up from or blocked entry into EM2/3, disable the DCDC interrupt using `NVIC_DisableIRQ(DCDC_IRQn)` before entering EM2/3 and re-enable the DCDC interrupt using `NVIC_EnableIRQ(DCDC_IRQn)` after EM2/3 wake-up.
Resolution
This issue is resolved on revision B devices.

3.4 ETAMPDET_E301 – E-Tamper Detect Malfunction on Writes to Upper Prescaler

Description of Errata
Writing to ETAMPDET → CLKPRESCVAL.UPPERPRESC to divide down the peripheral clock can cause the ETAMPDET peripheral to potentially malfunction.
Affected Conditions / Impacts
Regardless of the oscillator used, writing a non-zero value to the upper prescaler (ETAMPDET → CLKPRESCVAL.UPPERPRESC) and putting the ETAMPDET peripheral through one enable-disable-enable cycle can lead to the E-Tamper Detect module to end up in an unrecoverable state.
Workaround
Set ETAMPDET → CLKPRESCVAL.UPPERPRESC to 0. Not writing to the upper prescaler and using a low-frequency oscillator will lead to the module consuming additional current since it will not be possible to divide the clock down to its lowest value. To mitigate the increased current consumption from a higher ETAMPDET peripheral clock frequency, ULFRCO should be used as the clock source to divide the clock down to a lower frequency. Using the ULFRCO with the right lower prescaler settings will also help generate a pseudo-random bit stream (PRBS) that is below 100 Hz.
Resolution
This issue is resolved on revision B devices.

3.5 PLL0_E301 – USB PLL0 High Current Draw Causes Unreliable USB Functionality

Description of Errata
When the PLL0 is enabled to clock the USB module, the device will draw approximately 200 µA more current than expected. Over time, this leads to a condition in which the USB becomes non-functional because the clock appears to stop. Time to failure is dependent on the DVDD voltage, with failure occurring immediately if DVDD = 3.8V, and some time later at lower DVDD voltages.
Affected Conditions / Impacts
Enabling the PLL0 causes the device to draw higher current than expected, which results in unreliable USB module functionality. Once the device shows this failure, it cannot be recovered by lowering the DVDD voltage level.
Workaround
There is currently no workaround for this issue.
Resolution
This issue is resolved on revision B devices.

4. Revision History

Revision 0.5

June, 2025

- Added EUSART_E302.
- Added SE_E302.

Revision 0.4

December, 2022

- Added EUSART_E303 and EUSART_E304.

Revision 0.3

May, 2022

- Added CUR_E303.
- Added IADC_E306.
- Added PLL0_E301.
- Updated latest device revision to revision B.

Revision 0.2

October, 2021

- Added DCDC_E302.
- Added ETAMPDET_E301.

Revision 0.1

July, 2021

- 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