Gecko
EFM32PG22 Errata

This document contains information on the EFM32PG22 errata. The latest available revision of this device is revision C.

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: December, 2022.

1. Errata Summary

The table below lists all known errata for the EFM32PG22 and all unresolved errata of the EFM32PG22.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision:
			C
CUR_E302	Extra EM1 Current if FPU is Disabled	Yes	X
DCDC_E302	DCDC Interrupts Block EM2/3 Entry or Cause Unexpected Wake-up	Yes	X
EMU_E303	Watchdog Reset Hangs System Entering EM2 or EM3	Yes	X
EUART_E301	EUART Receiver Enters Lockup State when Using Low Frequency IrDA Mode	Yes	X
EUART_E302	Incorrect Stop Bits Lock Receiver	Yes	X
I2C_E303	I2C Fails to Indicate New Incoming Data	Yes	X
IADC_E306	Changing Gain During a Scan Sequence Causes an Erroneous IADC Result	Yes	X
TIMER_E301	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X
USART_E301	Possible Data Transmission on Wrong Edge in Synchronous Mode	Yes	X
USART_E302	Additional SCLK Pulses Can Be Generated in USART Synchronous Mode	Yes	X
USART_E304	PRS Transmit Unavailable in Synchronous Secondary Mode	No	X

2. Current Errata Descriptions

2.1 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
There is currently no resolution for this issue.

2.2 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
There is currently no resolution for this issue.

2.3 EMU_E303 – Watchdog Reset Hangs System Entering EM2 or EM3

Description of Errata
The chip can hang and require a hard reset (pin or power-on) to recover if: 1. The system is operating with VSCALE1 core voltage scaling (software has previously written a 1 to the EMU_CMD_EM01VSCALE1 bit), 2. The system is in the process or entering EM2 or EM3 (software has just executed the WFE or WFI instruction with the SLEEPDEEP bit in the System Control Register set), and 3. A Watchdog timeout reset is triggered.
Affected Conditions / Impacts
Systems operating with core voltage scaling can hang if a Watchdog reset occurs immediately upon EM2 or EM3 entry.
Workaround
Systems that keep the Watchdog enabled in low energy modes should, as a matter of good programming practice, service the Watchdog before entering EM2 or EM3. Calling the emlib `WDOGn_Feed()` function followed by the `WDOGn_SyncWait()` function (to ensure that the servicing write to the WDOG_CMD register completes execution) immediately before entering EM2 or EM3 will prevent a Watchdog reset that could possibly hang the system under the specified circumstances.
Resolution
There is currently no resolution for this issue.

2.4 EUART_E301 — EUART 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 EUART 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 EUART receiver and 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 EUART. Keep it this way until the receiver has been enabled and 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 EUART 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
EUART0->CFG1_SET = EUSART_CFG1_RXPRSEN;

// Enable EUART to configure Rx
EUART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Output EUART 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: EUSART_CTRL_RXINV = 1 in this workaround because the receiver input must be inverted for proper IrDA RZI operation.

- 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 EUART RX pin before enabling the EUART. Keep it this way until the receiver has been enabled and 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 EUART0 Rx to the alternate GPIO (PA00)
GPIO->EUARTROUTE[0].RXROUTE = (gpioPortA << _GPIO_EUART_RXROUTE_PORT_SHIFT) | (0 <<
_GPIO_EUART_RXROUTE_PIN_SHIFT);

// Enable EUART0 to configure Rx
EUART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Route EUART Rx to EUART_RX GPIO(EUSRT_RX_PORT & EUART_RX_PIN)
GPIO->EUARTROUTE[0].RXROUTE = (EUART_RX_PORT << _GPIO_EUART_RXROUTE_PORT_SHIFT) | (EUART_RX_PIN <<
_GPIO_EUART_RXROUTE_PIN_SHIFT);

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

  Note: EUSART_CTRL_RXINV = 1 in this workaround because the receiver input must be inverted for proper IrDA RZI operation.

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.

2.5 EUART_E302 — 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 EUART 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 EUART and force the input to the IrDA demodulator to 1 before re-enabling the EUART 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 EUART. Keep it this way until the receiver has been enabled and 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 EUART 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
EUART0->CFG1_SET = EUSART_CFG1_RXPRSEN;

// Enable EUART to configure Rx
EUART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Output EUART 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: EUSART_CTRL_RXINV = 1 in this workaround because the receiver input must be inverted for proper IrDA RZI operation.

- 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 EUART RX pin before enabling the EUART. Keep it this way until the receiver has been enabled and 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 EUART0 Rx to the alternate GPIO (PA00)
GPIO->EUARTROUTE[0].RXROUTE = (gpioPortA << _GPIO_EUART_RXROUTE_PORT_SHIFT) | (0 <<
_GPIO_EUART_RXROUTE_PIN_SHIFT);

// Enable EUART0 to configure Rx
EUART0->EN_SET = EUSART_EN_EN;

// Enable Rx
EUART0->CMD = EUSART_CMD_RXEN;

// Wait until Rx enable is synchronized
while ((EUART0->SYNCBUSY & EUSART_SYNCBUSY_RXEN) != 0U) {}

// Route EUART Rx to EUART_RX GPIO(EUSRT_RX_PORT & EUART_RX_PIN)
GPIO->EUARTROUTE[0].RXROUTE = (EUART_RX_PORT << _GPIO_EUART_RXROUTE_PORT_SHIFT) | (EUART_RX_PIN <<
_GPIO_EUART_RXROUTE_PIN_SHIFT);

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

  Note: EUSART_CTRL_RXINV = 1 in this workaround because the receiver input must be inverted for proper IrDA RZI operation.

Resolution

There is currently no resolution for this issue.

2.6 I2C_E303 – I²C Fails to Indicate New Incoming Data

Description of Errata
A race condition exists in which the I²C fails to indicate reception of new data when both user software attempts to read data from and the I²C 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 I²C to discard new incoming data bytes because RXFULL = 1 and would otherwise prevent user software from reading last byte written by the I²C 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 also sets 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 I²C 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 I²C hardware receives the next incoming data byte.
Resolution
There is currently no resolution for this issue.

2.7 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, erronous IADC results may occur.
Resolution
There is currently no resolution for this issue.

2.8 TIMER_E301 — 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 contiunously 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 contiunously even if the interrupt flag (TIMER_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 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 increment or decrement 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_IEN_OF)
  TIMER0->CNT += 1;

if (intFlags & TIMER_IEN_UF)
  TIMER0->CNT -= 1;
```

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.

2.9 USART_E301 — Possible Data Transmission on Wrong Edge in Synchronous Mode

Description of Errata
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 if the USART is configured to operate in synchronous mode with 1. USART_CLKDIV_DIV = 0 (clock = $f_{HFPERCLK}$ ÷ 2), 2. USART_CTRL_CLKPHA = 0, 3. USART_TIMING_CSHOLD = 1 and 4. 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 the chip select hold time (USART_TIMING_CSHOLD = 1).
Affected Conditions / Impacts
Reception of each data bit by the secondary is tied to a specific clock edge. Therefore, the late transmission by the main of the first bit of a word may cause the secondary to receive the incorrect data, especially if the data setup time for the secondary 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 operation in both the CLKPOL = CLKPHA = 0 and CLKPOL = CLKPHA = 1 modes.
Resolution
There is currently no resolution for this issue.

2.10 USART_E302 — 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 main 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 secondary 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 secondary would lose synchronization with the main 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.

2.11 USART_E304 — PRS Transmit Unavailable in Synchronous Secondary Mode

Description of Errata
When the USART is configured for synchronous secondary operation, the transmit output (MISO) is not driven if the signal is routed to a pin using the PRS producer (e.g., SOURCESEL = 0x20 and SIGSEL = 0x4 for USART0).
Affected Conditions / Impacts
Systems cannot operate the USART in synchronous secondary mode if the PRS is used to route the transmit output to the RX (MISO) pin. Operation is not affected in main 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 GPIO_USARTn_RXROUTE and GPIO_USARTn_TXROUTE registers are used.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

3. Revision History

Revision 0.2

December, 2022

- Updated the workaround in I2C_E303.
- Added CUR_E302, DCDC_E302, EUART_E301, EUART_E302,IADC_E306 and USART_E304.
- Replaced select terms with inclusive lexicon.

Revision 0.1

August, 2020

- 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