Wireless Gecko EFR32FG23 Errata

This document contains information on the EFR32FG23 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: February, 2023.

1. Errata Summary

The table below lists all known errata for the EFR32FG23 and all unresolved errata of the EFR32FG23.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision:		
			A	B	C
CUR_E302	Extra EM1 Current if FPU is Disabled	Yes	X	X	X
CUR_E303	Active Charge Pump Clock Causes High Current	Yes	X	X	—
DCDC_E302	DCDC Interrupts Block EM2/3 Entry or Cause Unexpected Wake-up	Yes	X	X	X
EMU_E305	DC-DC Refresh Time Delay	Yes	X	X	X
EUSART_E301	PRS Transmit Unavailable in Synchronous Secondary Mode	No	X	—	—
EUSART_E303	EUSART Receiver Enters Lockup State when Using Low Frequency IrDA Mode	Yes	X	X	X
EUSART_E304	Incorrect Stop Bits Lock Receiver	Yes	X	X	X
I2C_E303	I2C Fails to Indicate New Incoming Data	Yes	X	—	—
IADC_E305	FIFO Cannot Detect Eighth Entry	Yes	X	X	X
IADC_E306	Changing Gain During a Scan Sequence Causes an Errone-ous IADC Result	Yes	X	X	X
KEYSCAN_E301	Unused Rows Are Not Properly Gated Off	Yes	X	X	X
LCD_E301	LOADBUSY Status Goes Inactive Early With Prescaled Clock	Yes	X	X	X
RADIO_E306	Improper TX and RX Operation at High Temperature	Yes	X	X	X
USART_E301	Possible Data Transmission on Wrong Edge in Synchronous Mode	Yes	X	—	—
USART_E302	Additional SCLK Pulses Can Be Generated in USART Syn-chronous Mode	Yes	X	—	—
USART_E304	PRS Transmit Unavailable in Synchronous Secondary Mode	No	X	X	X

2. Current Errata Descriptions

There are no known errata for revision C of this device.

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_E305 – DC-DC Refresh Time Delay

Description of Errata
The DC-DC fast refresh delay required after exiting EM2/3 and entering continuous conduction mode (CCM) is not honored.
Affected Conditions / Impacts
When the system exits EM2/3 and the DC-DC is set to CCM, the DC-DC voltage comparator needs to be refreshed.This refresh delay is not honored when exiting EM2/3.
Workaround
Firmware must wait for at least 20 μs before enabling DC-DC CCM upon wake from EM2/3.
Resolution
There is currently no resolution for this issue.

2.4 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 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 EUSART. 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 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: 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 EUSART RX pin before enabling the EUSART. 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 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: 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 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 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: 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 EUSART RX pin before enabling the EUSART. 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 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: 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 IADC_E305 – FIFO Cannot Detect Eighth Entry

Description of Errata
The IADC is unable to detect when the eighth FIFO entry has been loaded and will not set the corresponding SINGLEFIFODVL or SCANFIFODVL flag in the IADC_IF register or request LDMA service.
Affected Conditions / Impacts
If the DVL field of IADC_SINGLEFIFOCFG or IADC_SCANFIFOCFG is set to VALID8, a FIFO full condition is not registered when the eighth FIFO entry is loaded. In particular, this means that if the LDMA is configured to empty the FIFO when it is filled with eight entries, the LDMA will never issue a request to perform the transfers necessary to do this. Similarly, the IADC will not set the IADC_IF_SINGLEFIFODVL or IADC_IF_SCANFIFODVL to request an interrupt in response to the FIFO being filled with eight entries.
Workaround
Do not configure the IADC to request an interrupt or LDMA service when all eight entries are full (IADC_SINGLEFIFOCFG_DVL = VALID8 or IADC_SCANFIFOCFG_DVL = VALID8). All other settings of the DVL field (VALID1 to VALID7) operate as expected.
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 KEYSCAN_E301 – Unused Rows Are Not Properly Gated Off

Description of Errata

Unused KEYSCAN row inputs cause the KEY bit in the KEYSCAN_IF register to be set at all times indicating a key was pressed. This prevents the interrupt flag from clearing and stops the scan procedure.

Affected Conditions / Impacts

The KEY bit in the KEYSCAN_IF register is always set when rows are left unused.

Workaround

Configure the GPIO_KEYSCAN_ROWSENSEnROUTE registers for any unused row inputs to the same GPIO port and pin associated with any of the row inputs that are used. For example, if rows 0, 1, and 2 are used and routed to PA05, PA06, and PA07 respectively, and rows 3, 4, and 5 are unused, the configuration could be:

```
// Routing GPIO pins PA05, PA06 and PA07 to rows 0, 1 and 2
GPIO->DBUSKEYPAD_ROWSENSE0ROUTE = 0 << _GPIO_DBUSKEYPAD_ROWSENSE0ROUTE_PORT_SHIFT |
5 << _GPIO_DBUSKEYPAD_ROWSENSE0ROUTE_PIN_SHIFT;
GPIO->DBUSKEYPAD_ROWSENSE1ROUTE = 0 << _GPIO_DBUSKEYPAD_ROWSENSE1ROUTE_PORT_SHIFT|
6 << _GPIO_DBUSKEYPAD_ROWSENSE1ROUTE_PIN_SHIFT;
GPIO->DBUSKEYPAD_ROWSENSE2ROUTE = 0 << _GPIO_DBUSKEYPAD_ROWSENSE2ROUTE_PORT_SHIFT|
7 << _GPIO_DBUSKEYPAD_ROWSENSE2ROUTE_PIN_SHIFT;

// Workaround - Connect unused rows 3, 4, and 5 to row 2 (PA07), a single used row
GPIO->KEYSCANROUTE.ROWSENSE3ROUTE = 0 << _GPIO_KEYSCAN_ROWSENSE3ROUTE_PORT_SHIFT |
7 << _GPIO_KEYSCAN_ROWSENSE3ROUTE_PIN_SHIFT;
GPIO->KEYSCANROUTE.ROWSENSE4ROUTE = 0 << _GPIO_KEYSCAN_ROWSENSE4ROUTE_PORT_SHIFT |
7 << _GPIO_KEYSCAN_ROWSENSE4ROUTE_PIN_SHIFT;
GPIO->KEYSCANROUTE.ROWSENSE5ROUTE = 0 << _GPIO_KEYSCAN_ROWSENSE5ROUTE_PORT_SHIFT |
7 << _GPIO_KEYSCAN_ROWSENSE5ROUTE_PIN_SHIFT;
```

Note that KEYSCAN_STATUS.ROW will report the same values for used and unused rows that route to the same GPIO. In the scenario above, KEYSCAN_STATUS.ROW bits 2, 3, 4, and 5 will show the same values. The unused row bits in the KEYSCAN_STATUS field should be masked so that unused row bits are set to 1, indicating a key is not pressed.

Resolution

There is currently no resolution for this issue.

2.9 LCD_E301 — LOADBUSY Status Goes Inactive Early With Prescaled Clock

Description of Errata
The LCD_STATUS_LOADBUSY bit erroneously reports completion of writes to the LCD_BACTRL, LCD_AREGA, LCD_AREGB, and LCD_SEGDn registers before synchronization is complete when LCD_CTRL_PRESCALE > 3.
Affected Conditions / Impacts
If LOADBUSY is used to gate consecutive writes to one of the affected registers, only the data associated with the last write is guaranteed to be latched into the register.
Workaround
For each write to one of the affected registers, insert a delay equal to LCD_CTRL_PRESCALE ÷ f_{LCDCLK} after LOADBUSY transitions from 1 to 0 before issuing the next write to the same register. **Note:** LOADBUSY reports when data written from the PCLK register domain into the LCD controller's low-frequency clock domain has been synchronized. It does not indicate when data written into one of the affected registers is actually driven on the LCD controller pins. In cases where writes to these registers, such as LCD_SEGDn, are intended to have the change in pin state be observable on the connected display, LOADBUSY should not be used to gate consecutive writes. Instead, the CPU should issue the register write and wait to issue the next write until a display update event or frame counter update event occurs as reported by the LCD_IF register DISPLAY or FC flag bits. Interrupts associated with these flags can and should be enabled in such cases to minimize energy use by keeping the CPU in a low-energy mode (e.g., EM2) between such consecutive register writes.
Resolution
There is currently no resolution for this issue.

2.10 RADIO_E306 – Improper TX and RX Operation at High Temperature

Description of Errata
Some devices may fail to lock to the correct RF frequency at high operating temperatures (above 120 °C).
Affected Conditions / Impacts
Devices using Gecko SDK prior to v4.0.0 at high operating temperatures may be unable to lock to the desired RF frequency. This may cause errors in the TX/RX frequency or an inability to transmit or receive data.
Workaround
Customers should use firmware provided in Gecko SDK v4.0.0 or later for proper TX/RX operation.
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. Resolved Errata Descriptions

This section contains previous errata for EFR32FG23 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 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_WARMUP-MODE = 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 be-tween 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 C devices.

3.2 EUSART_E301 — PRS Transmit Unavailable in Synchronous Secondary Mode

Description of Errata
When the EUSART 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 = 0x13 and SIGSEL = 0x4 for EUSART0).
Affected Conditions / Impacts
Systems cannot operate the EUSART in synchronous secondary mode if the PRS is used to route the transmit output to the RX (MI-SO) 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_EUSARTn_RXROUTE and GPIO_EUSARTn_TXROUTE registers are used.
Workaround
There is currently no workaround for this issue.
Resolution
This issue is resolved on revision B devices.

3.3 I2C_E303 – I^{2}C Fails to Indicate New Incoming Data

Description of Errata
A race condition exists in which the I^{2}C fails to indicate reception of new data when both user software attempts to read data from and the I^{2}C 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^{2}C to discard new incoming data bytes because RXFULL = 1 and would otherwise prevent user software from reading last byte written by the I^{2}C 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 I^{2}C 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^{2}C hardware receives the next incoming data byte.
Resolution
This issue is resolved on revision B devices.

3.4 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
This issue is resolved on revision B devices.

3.5 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
This issue is resolved on revision B devices.

4. Revision History

Revision 0.9

February, 2023

- Added LCD_E301.

Revision 0.8

December, 2022

- Added EUSART_E303 and EUSART_E304.
- Fixed workaround routing for KEYSCAN_E301.

Revision 0.7

June, 2022

- Updated latest device revision to revision C.
- Added IADC_E306 and KEYSCAN_E301.
- Resolved CUR_E303.

Revision 0.6

January, 2022

- Added CUR_E303.

Revision 0.5

December, 2021

- Added RADIO_E306.
- Updated the resolution of USART_E304.
- Replaced select terms with inclusive lexicon.

Revision 0.4

September, 2021

- Added CUR_E302 and DCDC_E302.
- Clarified the revision history.

Revision 0.3

June, 2021

- Updated latest device revision to revision B.
- Added EMU_E305.
- Resolved EUSART_E301, I2C_E303, USART_E301, USART_E302 and USART_E304.

Revision 0.2

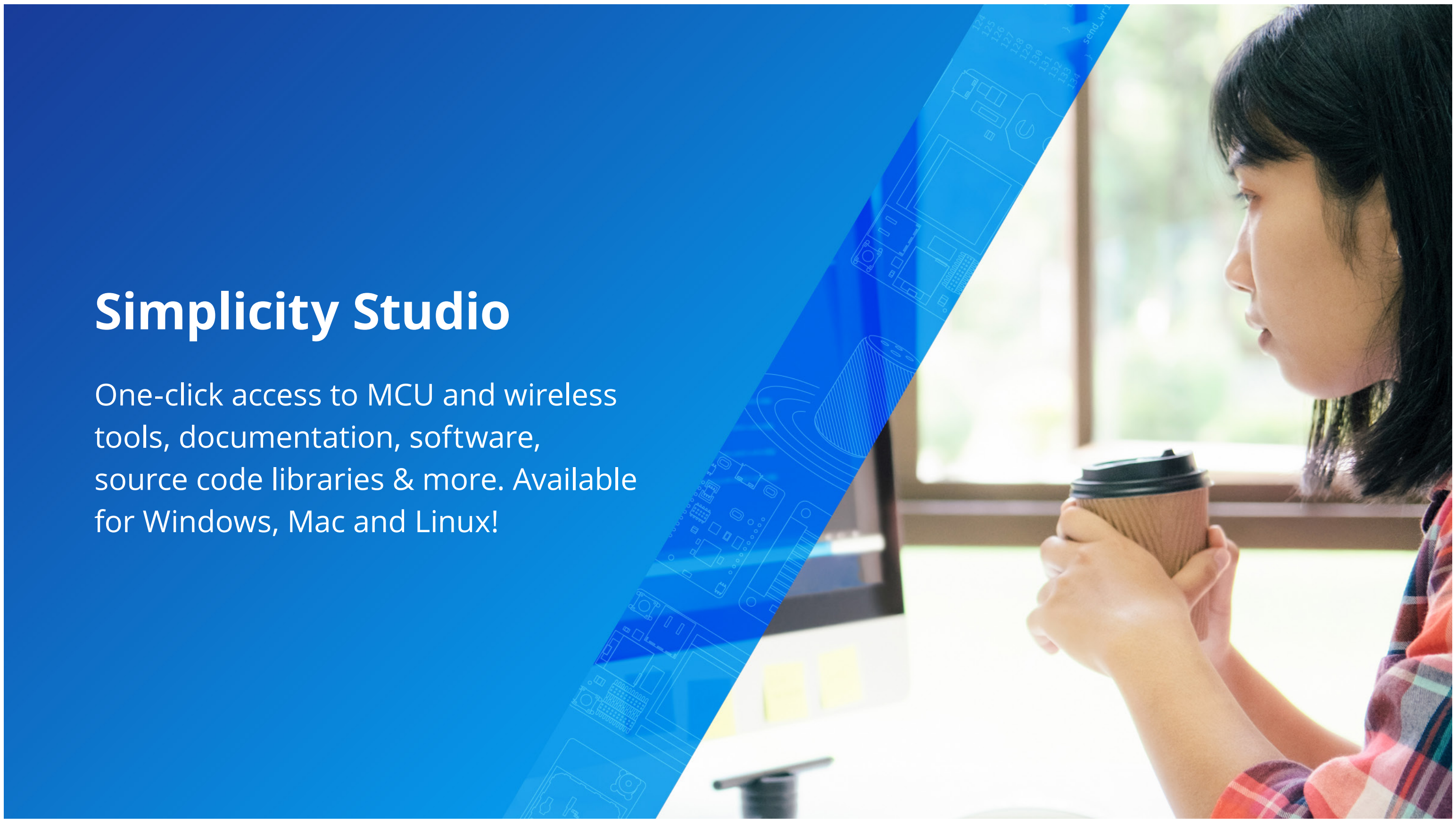
January, 2021

- Updated the workaround in I2C_E303.

Revision 0.1

September, 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