EFR32 Wireless Gecko EFR32MG12 Errata

This document contains information on the EFR32MG12 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 package marking or electronically.

Errata effective date: September, 2020.

1. Errata Summary

The table below lists all known errata for the EFR32MG12 and all unresolved errata in revision C of the EFR32MG12.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision:		
			A	B	C
ADC_E213	ADC KEEPINSLOWACC Mode	No	X	X	X
ADC_E222	ADC EM2 Wakeup on a Comparator Match Disables EM2 Entry	Yes	X	—	—
ADC_E224	ADC Warm-Up Ready Can Cause IDAC, ACMP, or CSEN to Not Function	Yes	X	X	—
ADC_E228	Limited ADC Sampling Frequency in EM2	No	X	X	X
CSEN_E201	CSEN_DATA in Debug Mode	Yes	X	X	X
CSEN_E202	CSEN Baseline DMA Transfers	Yes	X	X	X
CUR_E203	Occasional Extra EM0/1 Current	No	X	X	X
DBG_E204	Debug Recovery with JTAG Does Not Work	Yes	X	X	X
DCDC_E204	Potential Brownout when Enabling the DCDC from Off to Low Power Mode	Yes	X	X	X
EMU_E209	Potential EM2 Lock-up when using IDAC or the Debugger with the LDMA	Yes	X	—	—
EMU_E211	Radio Clocks Remain Disabled After Voltage Scaling	Yes	X	—	—
EMU_E212	Delay Required Between Successive Voltage Scaling Commands	Yes	X	X	X
EMU_E214	Device Erase Cannot Occur if Voltage Scaling Level is Too Low	Yes	X	X	X
EMU_E220	DECBOD Reset During Voltage Scaling After EM2 or EM3 Wakeup	Yes	X	X	X
I2C_E202	Race Condition Between Start Detection and Timeout	Yes	X	X	X
I2C_E203	I2C Received Data Can be Shifted	Yes	X	X	X
I2C_E205	Go Idle Bus Idle Timeout Does Not Bring Device to Idle State	Yes	X	X	X
I2C_E206	Slave Holds SCL Low After Losing Arbitration	Yes	X	X	X
I2C_E207	I2C Fails to Indicate New Incoming Data	Yes	X	X	X
LES_E201	LFPRESC Can Extend Channel Start-Up Delay	Yes	X	X	X
RADIO_E210	BLE Receiver False Detection Determination	No	X	X	X
RAM_E201	Timing Issues in Upper 192 kB of RAM	No	X	—	—
RMU_E202	External Debug Access Not Available After Watchdog or Lockup Full Reset	Yes	X	X	X
RMU_E203	AVDD Ramp Issue	Yes	X	X	—
RTCC_E203	Potential Stability Issue with RTCC Registers	Yes	X	X	X
RTCC_E204	Disabling the RTCC Backup RAM may Consume Extra Current	Yes	X	X	X

Designator	Title/Problem	Workaround Exists	Exists on Revision:		
			A	B	C
RTCC_E205	Wrap Event Can Be Missed	Yes	X	X	X
TIMER_E202	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X	X	X
USART_E201	USART DMA Transactions Fail with Slow Peripheral Clocks	No	X	X	X
USART_E203	DMA Can Miss USART Receive Data in Synchronous Mode	Yes	X	X	X
USART_E204	IrDA Modulation and Transmission of PRS Input Data	Yes	X	X	X
USART_E205	Possible Data Transmission on Wrong Edge in Synchronous Mode	Yes	X	X	X
USART_E206	Additional SCLK Pulses Can Be Generated in USART Synchronous Mode	Yes	X	X	X
VDAC_E201	VDAC Output Drives All APORT Buses Simultaneously	Yes	X	X	X
VDAC_E202	PRS Outputs Not Generated when Interrupt Flag is Set	Yes	X	X	X
WDOG_E201	Clear Command is Lost Upon EM2 Entry	Yes	X	X	X
WTIMER_E201	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X	X	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 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. Therefore, 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. 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 DCDC_E204 – Potential Brownout when Enabling the DCDC from Off to Low Power Mode

Description of Errata
A brownout event (DVDD BOD) can occur if the DC-DC converter is enabled from off to Low Power mode.
Affected Conditions / Impacts
Firmware cannot set REGPWRSEL before setting EMU->DCDCCTRL.DCDCMODE=LOWPOWER.
Workaround
To workaround this issue, firmware can: 1. Set the EMU->DCDCPWRCTRL.REGPWRSEL bitfield to 1 to select DVDD. 2. Set the EMU->DCDCCTRL.DCDCMODE bitfield to LOWNOISE. 3. Wait for the DCDCLNRUNNING interrupt (or wait for the DCDCLNRUNNING bit to be set in the EMU_IF register). 4. Set the EMU->DCDCCTRL.DCDCMODE bitfield to LOWPOWER. This workaround is included in v5.1.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.8 EMU_E212 – Delay Required Between Successive Voltage Scaling Commands

Description of Errata
Issuing two successive unsuccessful voltage scaling commands may cause the device to brown out.
Affected Conditions / Impacts
Systems using voltage scaling should use one of the two workarounds to avoid an undesired brown out.
Workaround
There are two workarounds for this problem: 1. Always wait for the VSCALEDONE interrupt when voltage scaling up or down. This is a very easy solution to implement, but may cause a delay of up to 25 µs when scaling up and up to 3 µs of delay when scaling down. 2. For systems sensitive to delays or long voltage scaling up time: a. When scaling down, wait for the VSCALEDONE (or VSCALEBUSY status) interrupt (up to 3 µs of delay). b. When scaling up, successive voltage scaling commands can be issued (no 25 µs delay required). Successive voltage scaling commands should never be issued when scaling down. This workaround is included in v5.1.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.9 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.

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.

2.10 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. Additionally, the EMU writes shown below should be added around EM2/EM3 entry and exit and after voltage scaling completes. This prevents the BOD state-machine transition signals from being delayed. This workaround adds 2.7 µs to the voltage scaling operation.

Note: This workaround is included in `em_emu.c` in the v2.7.4.0 or later of the Gecko SDK. It is recommended to workaround this issue by using the latest Gecko SDK version.

```
// Execute from RAM with interrupts disabled
*(uint32_t *) (EMU_BASE + 0x1A4) |= 0x1f << 10;
__WFI();
*(uint32_t *) (EMU_BASE + 0x14C) |=  0x01 << 31;
// Enable Interrupts and return to flash execution

. . . .

// After voltage scaling is complete
*(uint32_t *) (EMU_BASE + 0x14C) &= ~(0x01 << 31);
EMU->IFC = 0xFFFFFFFF;
```

Resolution

There is currently no resolution for this issue.

2.11 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.

2.12 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.

2.13 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.

2.14 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.

2.15 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 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.

2.16 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.

2.17 RADIO_E210 – BLE Receiver False Detection Determination

Description of Errata
The BLE receiver is susceptible to making a false detection determination when there is on-channel noise but no BLE signal present. Because packet arrival is determined by 2 - 3 preamble symbols, sample noise cannot be averaged down. This makes the probability of false detection higher. When the receiver is busy processing the signal based upon this false detection, an actual BLE packet can be missed.
Affected Conditions / Impacts
This issue can lead to a high BLE packet error rate (PER) and poor interoperability performance when interfacing with other BLE transmitters that may introduce noise in the channel before transmitting the preamble symbols.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.18 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.

2.19 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). This workaround is included in v5.1.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.20 RTCC_E204 – Disabling the RTCC Backup RAM may Consume Extra Current

Description of Errata
Disabling the RTCC backup RAM may cause higher EM4H current draw due to the internal power structure of the backup RAM.
Affected Conditions / Impacts
Systems disabling the RTCC backup RAM may see additional current consumption.
Workaround
Firmware should keep the RTCC backup RAM retained. Leakage for the backup RAM is low (~3 nA typical), so the impact is slight. This workaround is included in v5.1.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.21 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: • Do not use the CC1 wrap event feature (CCV1TOP in RTCC_CTRL should be cleared to 0). • 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.

2.22 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.

2.23 USART_E201 — USART DMA Transactions Fail with Slow Peripheral Clocks

Description of Errata
USART DMA transactions will fail when the USART peripheral clock is slower than the DMA clock and IGNORESREQ is cleared to 0.
Affected Conditions / Impacts
Systems will not be able to use the DMA with a USART running from a slow clock when IGNORESREQ is cleared to 0.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.24 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: • Synchronous master sample delay is enabled (USARTn_CTRL_SMSDELAY = 1) to improve timing at higher clock rates. • The receive FIFO is already full, and the receive data DMA request (USARTnRXDATAV) is asserted. • 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. • The transmit data DMA request arrives before the receive data DMA request (the transmit FIFO empties before the receive data DMA request is asserted). • A higher priority peripheral DMA request arrives while processing the transmit data DMA request but before the receive data DMA request is processed. 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
There is currently no resolution for this issue.

2.25 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.

2.26 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 = $f_{HFPERCLK}$ ÷ 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.

2.27 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.

2.28 VDAC_E201 — VDAC Output Drives All APORT Buses Simultaneously

Description of Errata
When VDACn_OPAx_OUT.APORTOUTEN is set, the VDACn/OPAx will drive all its connected APORT buses (BUSAY, BUSBY, BUSCY, and BUSDY) instead of only the APORT bus selected via APORTOUTSEL. However, the VDACn/OPAx APORT request signals do correspond to the programmed APORTOUTSEL value and therefore, the APORT conflict registers (OPAxAPORTCONFLICT in VDACn_STATUS, OPAxAPORTCONFLICTIF in VDACn_IF, and VDACn_APORTCONFLICT) will not reflect potential conflicts on the erroneously driven APORT buses. If any other peripherals (or other VDAC channel or other OPAMP) are using either of the above APORT buses at the same time, this can lead to contention on the APORT bus and possible high current consumption.
Affected Conditions / Impacts
Systems attempting to use multiple APORT buses and the VDAC may see contention.
Workaround
If none of the other APORT clients (e.g., ADC, ACMP, OPAx input muxes, etc.) use BUSAY, BUSBY, BUSCY, and BUSDY, then no problem exists and the potential simultaneous driving of these buses by VDACn/OPAx can be ignored. Alternatively, the VDACn/OPAx can be configured to use direct connections of its main or alternative outputs to certain pins, thereby bypassing the APORT. Direct output connections can be enabled by programming MAINOUTEN=1 and/or ALTOUTEN=1 (while keeping APORTOUTEN=0) in the VDACn_OPAx_OUT register. The device data sheet lists the available main output and alternative output connections to pins per VDAC output or OPAMP.
Resolution
There is currently no resolution for this issue.

2.29 VDAC_E202 — PRS Outputs Not Generated when Interrupt Flag is Set

Description of Errata
The conversion done (CD) PRS outputs from the DAC are tied to the interrupt flags. As long as the interrupt flag is set, no PRS output will be generated. When the first conversion done (CD) event occurs, the VDAC will set the interrupt flag and generate one PRS pulse. As long as the interrupt flag is set, any new conversion done events will not generate a new PRS pulse. After software clears the flag, the next conversion done event will generate a PRS pulse. Clearing the interrupt flag itself will not generate a pulse. Any CD event that occurs while the flag is set will be ignored.
Affected Conditions / Impacts
Systems attempting to use the DAC PRS outputs should ensure the interrupt flags are cleared.
Workaround
Firmware should clear the conversion done flag immediately after entering the interrupt service routine. This will allow the next conversion done event to generate a PRS pulse.
Resolution
There is currently no resolution for this issue.

2.30 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.

2.31 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 contiunously 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. Resolved Errata Descriptions

This section contains previous errata for EFR32MG12 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 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. When 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 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 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.

3.3 EMU_E209 – Potential EM2 Lock-up when using IDAC or the Debugger with the LDMA

Description of Errata
The device can lock up if firmware updates the IDAC output just before entering EM2 while the LDMA module is enabled. Similarly, the device can lock up if the Debugger is connected and the firmware enters EM2 while the LDMA module is enabled.
Affected Conditions / Impacts
Systems using the LDMA and IDAC or LDMA and Debugger may no longer function properly after attempting to enter EM2.
Workaround
Two workarounds exist: 1. If LDMA functionality in EM2 is not needed, firmware can disable the DMA via the CMU->HFBUSCLKEN* LDMA bit before entering EM2. 2. If LDMA functionality in EM2 is needed, wait for the IDAC output to settle before entry into EM2 or disconnect the debugger before entry into EM2.
Resolution
This issue is resolved in revision B devices.

3.4 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.

3.5 RAM_E201 – Timing Issues in Upper 192 kB of RAM

Description of Errata
The upper 192 kB of internal RAM has an issue where the hold timing is not always met. This may result in DMA issues when targeting these addresses in RAM. Normal CPU accesses to data at the affected RAM addresses do not have this issue.
Affected Conditions / Impacts
This issue may result in DMA issues when targeting the upper 192 kB of internal RAM.
Workaround
DMA accesses up to the 64 kB boundary should not have any issues. There is no workaround for the upper 192 kB of RAM.
Resolution
This issue is resolved in revision B devices.

3.6 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.

4. Revision History

Revision 0.8

September, 2020

- Added I2C_E207, USART_E206 and WDOG_E201.

Revision 0.7

April, 2020

- Added EMU_E220.

Revision 0.6

December, 2019

- Updated the workaround in RMU_E202.
- Added CSEN_E201, CSEN_E202, I2C_E202, I2C_E203, I2C_E205, I2C_E206, LES_E201, RADIO_E210, TIMER_E202, USART_E203, USART_E204, USART_E205, and WTIMER_E201.
- Migrated to new errata document format.

Revision 0.5

October 23rd, 2017

- Updated the latest revision to revision C.
- Moved ADC_E222, ADC_E224, EMU_E209, EMU_E211, RAM_E201, and RMU_E203 to the errata history section.

Revision 0.41

May 26th, 2017

- Renamed BOD_E202 to RMU_E203 and adjusted the wording.
- Reworded RMU_E202 to remove mention of specific core architectures.

Revision 0.4

May 15th, 2017

- Added ADC_E224, BOD_E202, and RMU_E202.

Revision 0.3

February 20th, 2017

- Updated the latest revision to revision B.
- Added DBG_E204, EMU_E214, USART_E201, VDAC_E201, and VDAC_E202.
- Removed CMU_E202, DBG_E203, and EMU_E213.
- Slightly adjusted the wording of the ADC_E213 description.

Revision 0.2

December 2nd, 2016

- Added CUR_E203, DBG_E203, RAM_E201, and RTCC_E205.

Revision 0.1

October 14th, 2016

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

http://www.silabs.com