EFR32 Wireless Gecko *Bluetooth*® Low Energy SoC CSP EFR32BG1 Errata

This document contains information on the EFR32BG1 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 EFR32BG1 and all unresolved errata in revision C of the EFR32BG1.

Table 1.1. Errata Overview

Designator	Title/Problem	Workaround Exists	Exists on Revision:
			C
ADC_E202	Wait After POR or EM4S Wakeup	Yes	X
ADC_E206	PROGERRIF (Program Error Interrupt Flag) Will Not Clear	Yes	X
ADC_E207	ADC Scan Repeat Mode with APORT	No	X
ADC_E208	ADC Interrupt Flags	Yes	X
ADC_E209	ADC and PRS Triggers	Yes	X
ADC_E210	ADC with PRS and Software Triggers	Yes	X
ADC_E211	ADC Single Repeat Mode and Tailgating	No	X
ADC_E212	ADC with PRS in ASYNC Mode	Yes	X
ADC_E213	ADC KEEPINSLOWACC Mode	No	X
ADC_E214	Using ADC CHCONMODE with PRS	Yes	X
ADC_E215	ADC CHCONMODE Set to MAXRESP Causes Extra Latency	Yes	X
ADC_E216	ADC Conversion Start Delay	Yes	X
ADC_E217	Multiple CLK Mode Switches	Yes	X
ADC_E218	SINGLEACT and SCANACT Status Flags Delayed	Yes	X
ADC_E219	STOP Command Causing FIFO Corruption	Yes	X
ADC_E220	AUXHFRCO in ASYNC mode with ASYNC CLK in ASNEEDED mode	Yes	X
ADC_E221	ADC Temperature Sensor Must be Used in LOWACC Mode	Yes	X
ADC_E222	ADC EM2 Wakeup on a Comparator Match Disables EM2 Entry	Yes	X
ADC_E223	Delayed ADC Conversion or Warmup Start	Yes	X
ADC_E224	ADC Warm-Up Ready Can Cause ACMP to Not Function	Yes	X
ADC_E226	SCANSTOP Does Not Immediately Stop the Ongoing Sample	No	X
ADC_E227	New Conversion Triggers Cause Jitter to the Ongoing Conversions	No	X
ADC_E228	Limited ADC Sampling Frequency in EM2	No	X
CORE_E201	SYSTICK and an External Clock	Yes	X
DBG_E201	AUXHFRCO Debug Limitations	Yes	X
DBG_E202	Debug Access to ADC and LEUART not Functioning as Intended	No	X
DBG_E204	Debug Recovery with JTAG Does Not Work	Yes	X
DCDC_E202	Regulated DCDC Output Can Dip on EM2 Entry	No	X
DCDC_E203	Regulated DCDC Output Can Dip on EM2 Entry if not in LN Mode	Yes	X
DCDC_E205	Delay Required after Enabling the DC-DC Before Entering EM2/3/4H	Yes	X
DCDC_E206	Reset During Radio Operation With DC-DC Results in DVDD Brown Out	Yes	X

Designator	Title/Problem	Workaround Exists	Exists on Revision:
			C
EFR_E201	Bit Access Not Supported for Low Energy Peripherals	Yes	X
EFR_E202	Read-Clear Access for LETIMER0 and RTCC Interrupts	Yes	X
EMU_E201	High Temperature Operation	Yes	X
EMU_E204	Restrictions Writing TEMPHIGH and TEMPLOW	Yes	X
EMU_E205	Restrictions Reading TEMP	Yes	X
EMU_E207	GPIO State can be Lost During EM4 Recovery	Yes	X
EMU_E208	Occasional Full Reset After Exiting EM4H	Yes	X
EMU_E209	Potential EM2 Lock-up when using IDAC or the Debugger with the LDMA	Yes	X
EMU_E210	Potential Power-Down When Entering EM2	Yes	X
EMU_E215	Device May Brown Out After Energy Mode Transition	Yes	X
EMU_E216	EM4H I/O Retention Cannot Be Disabled	Yes	X
FLASH_E201	Potential Program Failure after Power On	Yes	X
IDAC_E201	IDAC CURSTABLE Bit Not Reliable	Yes	X
I2C_E201	I2C ABORT Command	Yes	X
I2C_E202	Race Condition Between Start Detection and Timeout	Yes	X
I2C_E203	I2C Received Data Can be Shifted	Yes	X
I2C_E205	Go Idle Bus Idle Timeout Does Not Bring Device to Idle State	Yes	X
I2C_E206	Slave Holds SCL Low After Losing Arbitration	Yes	X
I2C_E207	I2C Fails to Indicate New Incoming Data	Yes	X
LEUART_E201	Restrictions Setting TXDMAWU/RXDMAWU of LEUARTn_CTRL	Yes	X
RMU_E201	CTRL Register Reset on All Resets	Yes	X
RMU_E202	External Debug Access Not Available After Watchdog or Lockup Full Reset	Yes	X
RTCC_E201	RTCC Does Not Support Compare/Capture Wrap with Prescaler	Yes	X
RTCC_E202	RTCC Triggers to LETIMER Not Safe	Yes	X
RTCC_E203	Potential Stability Issue with RTCC Registers	Yes	X
TIMER_E201	Timer in Input Capture Mode Can Stop Counting	Yes	X
TIMER_E202	Continuous Overflow and Underflow Interrupts in Quadrature Counting Mode	Yes	X
USART_E202	Incorrect 8-bit Timer Operation in Asynchronous Mode	No	X
USART_E203	DMA Can Miss USART Receive Data in Synchronous Mode	Yes	X
USART_E204	IrDA Modulation and Transmission of PRS Input Data	Yes	X
USART_E205	Possible Data Transmission on Wrong Edge in Synchronous Mode	Yes	X
USART_E206	Additional SCLK Pulses Can Be Generated in USART Synchronous Mode	Yes	X
WDOG_E201	Clear Command is Lost Upon EM2 Entry	Yes	X

2. Current Errata Descriptions

2.1 ADC_E202 – Wait After POR or EM4S Wakeup

Description of Errata
Attempting to take an ADC sample too soon after POR or EM4S wakeup can result in an erroneous sample being returned by the ADC.
Affected Conditions / Impacts
ADC can return erroneous sample data.
Workaround
After POR or EM4S wakeup, users must wait 150 µs before starting and using the ADC.
Resolution
There is currently no resolution for this issue.

2.2 ADC_E206 – PROGERRIF (Program Error Interrupt Flag) Will Not Clear

Description of Errata
If an invalid selection is made on APORT via POSSEL or NEGSEL by selecting both POSSEL and NEGSEL on the X bus or both on the Y bus, then PROGERRIF will set to 1. If this is followed by a valid internal selection (POSSEL set to AVDD and NEGSEL set to VSS), the PROGERRIF flag will remain 1, even though the selection is valid. This PROGERRIF flag can only be cleared by first making a valid selection of the APORT channels, then moving to an internal selection.
Affected Conditions / Impacts
If firmware attempts to clear the PROGERRIF error condition by selecting a valid non-APORT channel, the PROGERRIF bit will remain set to 1 even after firmware attempts to clear the flag.
Workaround
Firmware can clear the flag by first making a valid selection on POSSEL/NEGSEL to an APORT channel before moving to an internal selection.
Resolution
There is currently no resolution for this issue.

2.3 ADC_E207 – ADC Scan Repeat Mode with APORT

Description of Errata
If Scan repeat mode is enabled, then the ADC sets the correct APORT settings only on the first scan sequence conversion. APORT settings are cleared for all subsequent scan sequence conversions.
Affected Conditions / Impacts
Scan repeat mode does not work with the APORT.
Workaround
There is no known workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.4 ADC_E208 – ADC Interrupt Flags

Description of Errata
The SCANCMP and SINGLECMP interrupt flags in ADCn_IF are not clearable in certain scenarios. These interrupts can trigger multiple times on the same event if the event conditions are not cleared before clearing the interrupt flags.
Affected Conditions / Impacts
Multiple SCANCMP or SINGLECMP interrupts can occur from the same source.
Workaround
Once an interrupt is received, either disable the compare logic (clear CMPEN in ADCn_SINGLECTRL or ADCn_SCANCTRL) or clear the FIFO (using ADCn_SINGLEFIFOCLEAR or ADCn_SCANFIFOCLEAR) before clearing the interrupt flags.
Resolution
There is currently no resolution for this issue.

2.5 ADC_E209 – ADC and PRS Triggers

Description of Errata
Scan conversions may become dependent on Single triggers, and vice versa. If both Scan and Single channel mode are set to PRS Timed mode and both Scan and Single triggers are high at the same time before the approximation phase has started, the conversion is halted until both PRS triggers occur.
Affected Conditions / Impacts
A Scan conversion can end up waiting for a Single PRS trigger to go low before it starts the approximation phase, even if the Scan PRS trigger has already gone low, and vice versa.
Workaround
Do not set both ADC Single and Scan to use PRS Timed mode at the same time. Alternatively, if they are both set to use PRS Timed mode simultaneously, ensure that both PRS timed pulses are never high at the same time.
Resolution
There is currently no resolution for this issue.

2.6 ADC_E210 – ADC with PRS and Software Triggers

Description of Errata
Software triggered conversions are affected if the ADC is set to use PRS Timed mode, even when PRSEN is 0. If PRS Timed mode is selected using PRSMODE in ADCn_SCANCTRLX, regardless of what is set in PRSEN: 1. All channels in the scan sequence (except the first one) experience an additional 2 raw ADC_CLK cycle latency in the conversion. The same 2-cycle latency will be experienced in all subsequent conversions in repeat mode and conversions done due to oversampling in both single and scan mode. 2. If the software triggers a conversion and a PRS pulse comes in before the conversion has passed the acquisition phase, the software-triggered conversion will stall and wait for the PRS pulse to go low before starting the approximation phase.
Affected Conditions / Impacts
When using the PRS Timed mode with ADC scan, the overall scan conversion time will be longer than expected by: Extra ADC_CLK Cycles = 2 × (number of channels scanned – 1) In addition, 1 MSPS sampling will not be reachable with a software trigger, and software triggers may experience a dependency on the PRS triggers.
Workaround
There is currently no workaround for extra clocks generated by PRS Timed mode. When doing software triggered conversions, do not select PRS Timed mode in PRSMODE.
Resolution
There is currently no resolution for this issue.

2.7 ADC_E211 – ADC Single Repeat Mode and Tailgating

Description of Errata
If the ADC Single repetitive mode is enabled using REP in ADCn_SINGLECTRL and tailgating is enabled by setting TAILGATE in ADCn_CTRL, then the ADC waits for first Scan conversion. Once that completes, a Single conversion starts, but it is stopped by the ADC before it is complete.
Affected Conditions / Impacts
Single repeat mode does not work with tailgating enabled.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.8 ADC_E212 – ADC with PRS in ASYNC Mode

Description of Errata
The ADC is currently documented as an asynchronous PRS producer. However, when the ADC is setup in ASYNC mode (ADCCLKMODE in ADCn_CTRL is set to ASYNC), the PRS outputs are no longer synchronous to the HFPERCLK.
Affected Conditions / Impacts
The ADC PRS outputs will not be properly synchronized when the ADC is in ASYNC mode.
Workaround
Use the ADC PRS outputs only when the ADC is setup to use SYNC mode (ADCCLKMODE in ADCn_CTRL is set to SYNC).
Resolution
There is currently no resolution for this issue.

2.9 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.

2.10 ADC_E214 – Using ADC CHCONMODE with PRS

Description of Errata
When CHCONMODE in ADCn_CTRL is set MAXRESP, the ADC does not work with PRS Timed mode.
Affected Conditions / Impacts
If the PRS pulse is longer than the ADC acquisition time, the input mux select lines are switched during the acquisition phase, causing the results to no longer be usable.
Workaround
PRS Timed mode should not be used with CHCONMODE in ADCn_CTRL set to MAXRESP.
Resolution
There is currently no resolution for this issue.

2.11 ADC_E215 – ADC CHCONMODE Set to MAXRESP Causes Extra Latency

Description of Errata
Setting CHCONMODE in ADCn_CTRL to MAXRESP introduces 7 extra raw ADC_CLK cycles of latency between scan conversions in a sequence.
Affected Conditions / Impacts
The 7 raw ADC_CLK cycles of extra latency impacts the time the sample is taken.
Workaround
Use CHCONMODE set to MAXSETTLE in order to avoid the extra latency.
Resolution
There is currently no resolution for this issue.

2.12 ADC_E216 – ADC Conversion Start Delay

Description of Errata
If CONVSTARTDELAYEN in both SINGLECTRLX and SCANCTRLX registers are set to 1, the ADC will look at CONVSTARTDELAY value set in SINGLECTRLX register regardless of the conversion type (SCAN or SINGLE). If a SCAN conversion is triggered in this scenario, the conversion will be delayed for the value specified in SINGLECTRLX. In all other cases, the ADC behaves as expected.
Affected Conditions / Impacts
Enabling both SINGLE and SCAN conversion start delay can result in unexpected behavior if the delay selection in the SINGLE and SCAN registers is different.
Workaround
If different CONVSTARTDELAY values are desired for SCAN and SINGLE, do not keep both SINGLE CONVSTARTDELAY and SCAN CONVSTARTDELAY enabled at the same time.
Resolution
There is currently no resolution for this issue.

2.13 ADC_E217 – Multiple CLK Mode Switches

Description of Errata
This issue can be encountered if the ADC clock (CLK) mode switches between asynchronous (ASYNC) and synchronous (SYNC) while data is being read. Specifically, this issue can occur if ADC operates in ASYNC mode with converted data being read, then some conversions are done in SYNC mode before being switched back to ASYNC mode again. FIFOCOUNT may show the wrong value when read in ASYNC mode after the last clock mode switch. Note that the recommended procedure for switching CLK modes should always be followed.
Affected Conditions / Impacts
An unexpected value may be read from FIFOCOUNT registers.
Workaround
When switching from SYNC to ASYNC mode more than once, clear the FIFOs using FIFOCLEAR before and after the CLK mode switch.
Resolution
There is currently no resolution for this issue.

2.14 ADC_E218 – SINGLEACT and SCANACT Status Flags Delayed

Description of Errata
Once the SINGELSTART/SCANSTART commands are issued, it takes a few cycles before the ADC SINGLEACT/SCANACT status flags are set.
Affected Conditions / Impacts
The status flags cannot be checked right after the conversion start commands are issued.
Workaround
Firmware that wants to check when a conversion has started after sending a software trigger will need to wait until the corresponding status flag (SINGELACT/SCANACT) goes high before proceeding.
Resolution
There is currently no resolution for this issue.

2.15 ADC_E219 – STOP Command Causing FIFO Corruption

Description of Errata
If a single or scan conversion is running and a software STOP command is issued, the conversion should stop immediately and the result should be be discarded (no FIFO updates should happen). Currently in the ADC, if a single (scan) conversion is stopped by a software STOP command and there is a scan (single) conversion pending, then the conversion will incorrectly continue and the result will be used to update the FIFO.
Affected Conditions / Impacts
Issuing the STOP command can have two different effects. If the command is sent during a conversion, the effect will be immediate if no other conversion is pending. The immediate effect means stopping the on-going conversion and discarding the current sample. If the command is sent during a conversion, the on-going conversion will finish and the current sample will be saved into the corresponding FIFO. This means that the single conversion will fully finish regardless of the ADC mode (e.g., if in the oversampling mode, the full oversampling will be executed, or if in the repetition mode, the current conversion will finish and then the repetition mode for single will be disabled). Additionally, the scan conversion will fully finish conversion of the current channel, but it will not finish the whole scan sequence (regardless of the ADC mode). In the last case, the scan FIFO will be updated with the data from channels converted so far.
Workaround
If using the STOP command, the effect will be immediate if no scan triggers were pending during the single conversion and vice versa.
Resolution
There is currently no resolution for this issue.

2.16 ADC_E220 – AUXHFRCO in ASYNC mode with ASYNC CLK in ASNEEDED mode

Description of Errata
ADC sampling in the ASYNC ASNEEDED mode when running from the AUXHFRCO can temporarily upset voltage references used in certain analog components. This issue effects the BOD trip point, DCDC voltage accuracy, ACMP reference accuracy, IDAC output current accuracy, and low voltage digital supply voltage accuracy.
Affected Conditions / Impacts
If the ADC is being used in ASYNC mode with AUXHFRCO, ASYNC CLK cannot be used in ASNEEDED mode.
Workaround
If the ADC is being used in ASYNC mode with AUXHFRCO, ASYNC CLK must be set to ALWAYSON mode.
Resolution
There is currently no resolution for this issue.

2.17 ADC_E221 – ADC Temperature Sensor Must be Used in LOWACC Mode

Description of Errata
The ADC temperature sensor used in HIGHACC mode can temporarily upset voltage references used in certain analog components. This issue effects the BOD trip point, DCDC voltage accuracy, ACMP reference accuracy, IDAC output current accuracy, and low voltage digital supply voltage accuracy.
Affected Conditions / Impacts
If the ADC is being used to take a reading from its internal temperature sensor, ADCn_BIASPROG.GPBIASACC = 0 cannot be used.
Workaround
Use ADCn_BIASPROG.GPBIASACC = 1 when taking temperature measurements.
Resolution
There is currently no resolution for this issue.

2.18 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
There is currently no resolution for this issue.

2.19 ADC_E223 – Delayed ADC Conversion or Warmup Start

Description of Errata
When a new conversion trigger is received from PRS or a software start, the ADC is expected to either: 1. Immediately start warmup (if ADCn_CTRL_WARMUPMODE is set to NORMAL, KEEPINSTANDBY, or KEEPINSLOWACC). 2. Immediately start the conversion (if in KEEPADCWARM warmup mode). This expected behavior does not occur if the ADC prescaler (ADCn_CTRL_PRESC) is set to a non-zero value, as the start of the ADC warmup or conversion gets delayed by the number of ADC clock cycles specified by the PRESC field.
Affected Conditions / Impacts
Systems using the ADC clock prescaler will see a delay after a start-of-conversion or start-of-warmup trigger and the actual conversion or warmup sequence.
Workaround
For systems that cannot tolerate a delay between the start-of-conversion and actual conversion or before ADC warmup, set ADCn_CTRL_PRESC to 0 and use the prescalars in the CMU to prescale the incoming ADC_CLK.
Resolution
There is currently no resolution for this issue.

2.20 ADC_E224 – ADC Warm-Up Ready Can Cause ACMP to Not Function

Description of Errata
The ACMP module uses 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 ACMP module 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 ACMP module in conjunction with the ADC can see intermittent failures where these modules do not operate.
Workaround
To work around this issue, enable the ACMP module before enabling the ADC. This will ensure the handshaking logic between the ADC and other modules functions correctly.
Resolution
There is currently no resolution for this issue.

2.21 ADC_E226 – SCANSTOP Does Not Immediately Stop the Ongoing Sample

Description of Errata
The stop behavior of conversions differs between scan mode (SCANSTOP in ADCn_CMD) or single conversion mode (SINGLESTOP in ADCn_CMD). 1. If the SINGLESTOP command is sent during a single conversion, the effect will be immediate if no other conversion is pending. In this case, the ADC immediately stops the on-going conversion and discards the current sample. 2. If the SCANSTOP command is sent during a scan conversion, the on-going conversion will finish and the current sample will be saved into the corresponding FIFO. If oversampling mode is enabled (RES in ADCn_SCANCTRL set to OVS), the full oversampling will be completed for this sample. If in repetition mode (REP in ADCn_SCANCTRL set to 1), the current conversion will finish and then the repetition mode for the single channel will be disabled. Once the current conversion completes, the rest of the scan sequence aborts.
Affected Conditions / Impacts
Systems issuing a SCANSTOP command to the ADC may see an additional conversion in scan mode.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.22 ADC_E227 – New Conversion Triggers Cause Jitter to the Ongoing Conversions

Description of Errata
In some cases, an ADC conversion can take one conversion clock longer in the worst case if a new conversion trigger occurs while another conversion is ongoing. The results of the conversion are unaffected.
Affected Conditions / Impacts
Systems issuing conversion triggers while conversions are ongoing may see slightly longer conversion times.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.23 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.

2.24 CORE_E201 – SYSTICK and an External Clock

Description of Errata
The core allows two different clock sources for the SysTick counter. The first one is the core free-running clock, which operates correctly. The second source uses the 32 kHz from the RTCC. This pulse width of this clock is not wide enough, which results in missed SysTick counts.
Affected Conditions / Impacts
Firmware should not use the external clock source for the SysTick counter.
Workaround
Use the core free-running clock for the SysTick, which is the default selection.
Resolution
There is currently no resolution for this issue.

2.25 DBG_E201 – AUXHFRCO Debug Limitations

Description of Errata
The AUXHFRCO is the default debug clock, set by DBG in CMU_DBGCLKSEL. Using AUXHFRCO as the debug clock while entering EM2 has the potential of corrupting the system, causing some registers in the TPIU to not retain their value.
Affected Conditions / Impacts
Firmware should not use the AUXHFRCO as the debug clock while entering EM2.
Workaround
When using AUXHFRCO as the debug clock, it must be stopped before entering the EM2 power mode. Alternatively, select another clock source as the debug clock before entering EM2.
Resolution
There is currently no resolution for this issue.

2.26 DBG_E202 – Debug Access to ADC and LEUART not Functioning as Intended

Description of Errata
ADC and LEUART registers that have a side-effect during a read access (i.e., LEUART_RXDATA or other registers that pop data read) continue to execute triggered actions when an attached debugger is performing read accesses. The intended behavior is to halt execution of pop actions when a debugger is attached.
Affected Conditions / Impacts
Some ADC and LEAURT registers will pop data from their respective buffers on read accesses from the debugger.
Workaround
The user needs to be aware that while debugging, reads via the debugger have the same affect as reads by the CPU during normal mode for these registers. To avoid the debugger triggering a data pop from the buffer, the user should avoid setting a memory window over the actionable register(s), as the reads to fetch the data for the debugger memory view would trigger the data pop from the buffer.
Resolution
There is currently no resolution for this issue.

2.27 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.

2.28 DCDC_E202 – Regulated DCDC Output Can Dip on EM2 Entry

Description of Errata
The regulated output on DVDD, when using the DCDC, can dip up to 4% during EM2 entry.
Affected Conditions / Impacts
External components operating from DVDD, when regulated by the DCDC, may suffer a depressed supply by up to 4%, which can last approximately 1 ms. Operation of the device is not affected when this occurs.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.29 DCDC_E203 – Regulated DCDC Output Can Dip on EM2 Entry if not in LN Mode

Description of Errata
The regulated output on DVDD, when using the DCDC, can dip up to approximately 9% during EM2 entry if the DCDC has not completed a transition into LN mode. Note that if a switch to LN mode completes prior to entry to EM2, the DCDC can exhibit the behavior described in DCDC_E202.
Affected Conditions / Impacts
External components operating from DVDD, when regulated by the DCDC, may suffer a depressed supply by up to approximately 9%, which can last approximately 1 ms. A BOD reset of the device is possible, but unlikely.
Workaround
Firmware should wait to see LNRUNNING set after initiating a transition of the DCDC into LN mode, before attempting to enter EM2. This workaround is included in v5.0.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.30 DCDC_E205 – Delay Required after Enabling the DC-DC Before Entering EM2/3/4H

Description of Errata
The DC-DC hardware state machine may malfunction if the device transitions to EM2, EM3, or EM4H before the DC-DC starts running Low Noise mode after being enabled.
Affected Conditions / Impacts
Systems using the DC-DC converter should wait ~300 µs until the LNRUNNING status bit in the DCDCSTATUS register is set before transitioning to EM2, EM3, or EM4H after enabling the DC-DC converter.
Workaround
Firmware should wait for the DCDCLNRUNNING bit to be set in the EMU_IF register before transitioning to EM2, EM3, or EM4H after enabling the DC-DC converter. This workaround is included in v5.1.0 or later of the Gecko SDK.
Resolution
There is currently no resolution for this issue.

2.31 DCDC_E206 – Reset During Radio Operation With DC-DC Results in DVDD Brown Out

Description of Errata

During radio operation with the DC-DC enabled, the protocol stacks enable radio interference minimization features which change the source of the DC-DC clock to reduce the impact of the DC-DC switching frequency on the radio. When a soft reset occurs (e.g., from a Pin or WDOG), the minimal interference DC-DC clock stops, causing the DC-DC to stop switching and charging up the output voltage, which results in a brown-out on the DVDD supply. The device then resets to its default start-up DC-DC configuration with the bypass switch enabled (i.e., DVDD=VREGVDD), and execution continues.

As a result of this sequence, the original reset cause indicated in the RMU_RSTCAUSE register will be masked by the brown-out reset (DVDDBOD) flag.

Affected Conditions / Impacts

Systems using the radio and DC-DC converter that are expecting to read valid results from the RMU_RSTCAUSE register for EM4RST, WDOGRST, SYSREQRST, LOCKUPRST, or EXTRST resets will see a brown-out reset event, instead.

Workaround

To prevent the DVDD brown-out, firmware can immediately enable the bypass switch early in code execution. For example, in the `SystemInit()` function in `system_efr32xg1b.c`, add the following lines of code:

```
BUS_RegBitWrite(&EMU->DCDCCLIMCTRL, _EMU_DCDCCLIMCTRL_BYPLIMEN_SHIFT, 1);
EMU->DCDCCTRL = (EMU->DCDCCTRL & ~_EMU_DCDCCTRL_DCDCMODE_MASK) | emuDcdcMode_Bypass;
*(volatile uint32_t *)(0x400E3074) &= ~(0x1UL << 0);
*(volatile uint32_t *)(0x400E3060) &= ~(0x1UL << 28);
```

Then, add the following line at the beginning of `main()`:

```
BUS_RegBitWrite(&EMU->DCDCCLIMCTRL, _EMU_DCDCCLIMCTRL_BYPLIMEN_SHIFT, 0);
```

This will disable the bypass current limit, which was enabled in the first line of the `SystemInit()` sequence. This limit prevents excessive current through the bypass when transitioning across large steps between DVDD and VDD, but consumes ~10uA of current that is no longer necessary once DVDD has ascended to near VDD.

Additional information on the workaround and examples provided is available from the following KB article URL:

http://community.silabs.com/t5/Mesh-Knowledge-Base/DCDC-E206-Reset-During-Radio-Operation-With-DC-DC-Results-in/ta-p/193802

Note: This workaround may not suffice for a pin reset initiated by a button press, which may be on the order of 50-300 ms. Because the device won't come out of reset to execute the workaround until the reset button is released, the DVDD supply will discharge during the entire duration the button is pressed, which will likely result in a DVDD brown-out.

Resolution

There is currently no resolution for this issue.

2.32 EFR_E201 – Bit Access Not Supported for Low Energy Peripherals

Description of Errata
Bit set and clear operations do not work properly for Low Energy Peripherals including WDOG, PCNT0, LEUART0, LETIMER0, and RTCC.
Affected Conditions / Impacts
To implement bit set or bit clear operations with Low Energy Peripherals, firmware must execute a read-modify-write operation address on the peripheral's registers.
Workaround
To implement bit set or bit clear operations with Low Energy Peripherals (WDOG, PCNT0, LEUART0, LETIMER0, and RTCC), firmware must execute a read-modify-write operation address on the peripheral's registers.
Resolution
There is currently no resolution for this issue.

2.33 EFR_E202 – Read-Clear Access for LETIMER0 and RTCC Interrupts

Description of Errata
The automatic read-clear mechanism for the LETIMER0 and RTCC modules does not actually clear the module interrupts.
Affected Conditions / Impacts
Firmware must be written to manually clear the interrupts for the LETIMER0 and RTCC modules.
Workaround
Firmware must read the LETIMER0 and RTCC interrupts using the module IFS register and clear the interrupts manually by writing to the module IFC register.
Resolution
There is currently no resolution for this issue.

2.34 EMU_E201 – High Temperature Operation

Description of Errata
The performance of analog peripherals at high temperatures (above 50 °C) may change over time. Firmware should periodically adjust the calibration of the analog peripherals to compensate for this behavior. This issue affects the BOD trip point, dc-dc output voltage accuracy, ACMP reference accuracy, and IDAC output current accuracy in EM2 through EM4H power modes. This does not affect operation of these peripherals in the EM0, EM1, or EM4S power modes.
Affected Conditions / Impacts
The performance of analog peripherals at high temperatures may change over time.
Workaround
The TEMPDRV module in emdrv addresses this issue and should be included in application firmware. This module is automatically included for systems using Silicon Labs software stacks. For systems not using a Silicon Labs stack (i.e., writing code from scratch or using software examples as a starting point), firmware should include the TEMPDRV module and call the `TEMPDRV_Init()` function to improve high temperature operation (above 50 °C). The module documentation can be found in Simplicity Studio and contains more information about the firmware solution. See application note, AN1027: EFR32xG1 and EFM32PG1/JG1 High-Temperature Operation for more information.
Resolution
There is currently no resolution for this issue.

2.35 EMU_E204 – Restrictions Writing TEMPHIGH and TEMPLOW

Description of Errata
Writing TEMPHIGH and TEMPLOW in EMU_TEMPLIMITS at certain times can cause erroneous interrupts to occur.
Affected Conditions / Impacts
Writing to TEMPHIGH or TEMPLOW in EMU_TEMPLIMITS at any time may cause the TEMPHIGH and TEMPLOW interrupt flags in EMU_IF to trigger incorrectly.
Workaround
Firmware should only write TEMPHIGH and TEMPLOW within 250 ms of receiving a TEMPLOW, TEMPHIGH, or TEMP interrupt.
Resolution
There is currently no resolution for this issue.

2.36 EMU_E205 – Restrictions Reading TEMP

Description of Errata
Reads of TEMP in EMU_TEMP may not always return the correct value.
Affected Conditions / Impacts
Reading TEMP in EMU_TEMP at any time may read an incorrect value.
Workaround
Firmware should restrict its reading of TEMP to the following scenarios: 1. Read the TEMP field multiple times until the same value is returned in two consecutive reads. 2. Read TEMP within 250 ms of receiving a TEMPLOW, TEMPHIGH, or TEMP interrupt.
Resolution
There is currently no resolution for this issue.

2.37 EMU_E207 – GPIO State can be Lost During EM4 Recovery

Description of Errata
Firmware can configure the I/O state to be retained when exiting EM4 by setting EM4IORETMODE in EMU_EM4CTRL. The desired behavior is that firmware can restore the state of the I/Os in EM0 after exit from EM4 via reset while the I/O state is maintained. It is possible for a GPIO saving a non-5 V tolerant (non-OVT) configuration pull-down configuration to lose the pull-down state if 5 V (OVT) tolerance is disabled using GPIO_Px_OVTDIS before restoring the pull-down configuration.
Affected Conditions / Impacts
Restoring the GPIO after an EM4 wakeup improperly can result in the I/O pull-down configuration being lost.
Workaround
The loss of the pull-down state can be avoided by re-enabling the pull-down before disabling the Over Voltage Tolerance for a GPIO using GPIO_Px_OVTDIS.
Resolution
There is currently no resolution for this issue.

2.38 EMU_E208 – Occasional Full Reset After Exiting EM4H

Description of Errata
Exiting EM4H may occasionally cause a full reset causing loss of RTCC counters and result in longer wakeup times on the order of power-on wakeup times.
Affected Conditions / Impacts
When waking up from EM4H, the system does not wait long enough for the BODs to settle before enabling it as a reset source. Even if the power is stable on DECOUPLE, AVDD, or DVDD, the part may see a BOD reset. The reset is reflected in the RESETCAUSE register.
Workaround

The work around is to disable the BOD prior to EM4H entry. This work around is automatically implemented when the `EMU_EnterEM4()` function is called in the Gecko SDK (versions v4.4.0 or later).
`__disable_irq();`
`*(volatile uint32_t *)(EMU_BASE + 0x190)  = 0x0000ADE8UL;`
`*(volatile uint32_t *)(EMU_BASE + 0x198) |= (0x1UL << 7);`
The BODs will automatically be enabled after EM4H wakeup, but disabling the BOD will also result in disabling the EM4BOD protection. POR reset will still be valid. This work around is not needed for EM4S.
Note: Use the `RMU_ResetCauseGet()` function in emlib on EM4 wakeup to differentiate between a BOD reset and EM4RST. For Gecko SDK versions earlier than v5.0.0, the return value from `RMU_ResetCauseGet()` will be 0x0000001C (BOD reset) instead of 0x00010000 (EM4RST) when waking up from EM4H.

Resolution
There is currently no resolution for this issue.

2.39 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
There is currently no resolution for this issue.

2.40 EMU_E210 – Potential Power-Down When Entering EM2

Description of Errata
On the rare occasion when the firmware starts a transition to EM2 while the EMU is updating its Temperature Sensor measurement, and then within a 300 ns window of entering EM2 the system is woken up to EM0 or EM1, the device's internal power system can be erroneously disabled, powering down the device. When this occurs, the device will reset and will set the DECBOD flag in the RMU_RSTCAUSE register.
Affected Conditions / Impacts
Systems that do not use the emdrv TEMPDRV module may experience a power down on these very rare EM2 timing transitions.
Workaround
The TEMPDRV module in emdrv and `CHIP_Init()` (SDK version v5.0.0 or later) addresses this issue and should be included in application firmware. This module is automatically included for systems using Silicon Labs software stacks. For systems not using a Silicon Labs stack (i.e., writing code from scratch or using software examples as a starting point), firmware should include the TEMPDRV module and call the `TEMPDRV_Init()` and `CHIP_Init()` functions to prevent this issue from occurring. The module documentation can be found in Simplicity Studio and contains more information about the firmware solution.
Resolution
There is currently no resolution for this issue.

2.41 EMU_E215 – Device May Brown Out After Energy Mode Transition

Description of Errata
The logic controlling the internal voltage regulator may rarely transition to an incorrect state after a transition to EM0 or EM1. When this occurs, the device will reset and will set the DECBOD flag in the RMU_RSTCAUSE register.
Affected Conditions / Impacts
Systems that do not use the emdrv TEMPDRV module may experience a brown out on these rare EM0 or EM1 timing transitions.
Workaround
The TEMPDRV module in emdrv (SDK version v5.0.0 or later) addresses this issue and should be included in application firmware. This module is automatically included for systems using Silicon Labs software stacks. For systems not using a Silicon Labs stack (i.e., writing code from scratch or using software examples as a starting point), firmware should include the TEMPDRV module and call the `TEMPDRV_Init()` function to prevent this issue from occurring. The module documentation can be found in Simplicity Studio and contains more information about the firmware solution.
Resolution
There is currently no resolution for this issue.

2.42 EMU_E216 – EM4H I/O Retention Cannot Be Disabled

Description of Errata
Even if I/O retention is disabled by clearing EM4IORETMODE in EMU_EM4CTRL to the DISABLE setting, the affected devices behave as if I/O retention is configured as SWUNLATCH, meaning that the pin state is retained until software writes the EM4UNLATCH bit in the EMU_CMD register to remove retention.
Affected Conditions / Impacts
Systems that disable I/O retention mode may still see pins retain state until software writes the EM4UNLATCH bit in the EMU_CMD register to remove retention.
Workaround
Software should always perform an I/O unlatch after exiting EM4H by calling the `EMU_UnlatchPinRetention()` function.
Resolution
There is currently no resolution for this issue.

2.43 FLASH_E201 – Potential Program Failure after Power On

Description of Errata
There is very small probability that, with some devices, the first program of flash after a power on cycle or wake-up from EM2, EM3, EM4H, or EM4S may not take effect unless the flash has first been programmed or erased. This issue affects all devices, though the probability of the failure occurring on each device may differ.
Affected Conditions / Impacts
After a power-on, the first program of flash initiated by firmware may not take effect.
Workaround
To ensure the flash is programmed correctly, firmware should program the flash, verify the flash contents, and reprogram the flash, if necessary. This workaround is included in v4.4.0 or later of the Gecko SDK. Note that typical word (32-bit) programming time is 26 µs, while it only takes several tens of ns to read and verify, so the overhead is minimal. An alternative workaround is to erase the flash page before programming after a power on cycle or wake-up from EM2, EM3, EM4S, or EM4H.
Resolution
There is currently no resolution for this issue.

2.44 IDAC_E201 – IDAC CURSTABLE Bit Not Reliable

Description of Errata
The IDAC CURSTABLE flag in IDAC_STATUS may erroneously report that the current output is stable.
Affected Conditions / Impacts
Systems using the IDAC should not rely on the CURSTABLE to determine if the output is stable.
Workaround
The CURSTABLE bit must not be used. Firmware must wait the minimum required IDAC settling time listed in the data sheet.
Resolution
There is currently no resolution for this issue.

2.45 I2C_E201 – I2C ABORT Command

Description of Errata
For on-going transactions, the ABORT command in I2Cn_CMD may cause the I2C module to lock up indefinitely if it is issued in the middle of an I2C transaction.
Affected Conditions / Impacts
During on-going transactions, the ABORT command can cause the I2C receive module FSM to hang, locking up the I2C transaction indefinitely.
Workaround
The ABORT command should only be used after the I2C module is enabled in order to instruct the I2C module that the bus is IDLE. To use the ABORT command during a transaction, the I2C module should be disabled by clearing EN in I2Cn_CTRL.
Resolution
There is currently no resolution for this issue.

2.46 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.

2.47 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.

2.48 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.

2.49 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.

2.50 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.

2.51 LEUART_E201 – Restrictions Setting TXDMAWU/RXDMAWU of LEUARTn_CTRL

Description of Errata
Changing the value of TXDMAWU while TXEN = 1 or RXDMAWU while RXEN = 1 in LEUARTn_CMD could potentially cause unpredictable behavior.
Affected Conditions / Impacts
If the TXDMAWU field is updated while TXEN = 1 or the RXDMAWU field is updated while RXEN =1, a spurious DMA wake-up event could be created.
Workaround
Firmware should first disable the receive or transmit path using RXEN or TXEN in LEAURTn_CMD before changing RXDMAWU or TXDMAWU.
Resolution
There is currently no resolution for this issue.

2.52 RMU_E201 – CTRL Register Reset on All Resets

Description of Errata
The RMU_CTRL register is reset to default state on every system reset, not only POR and hard pin reset as stated in the reference manual.
Affected Conditions / Impacts
The RMU_CTRL register is reset to default state on every system reset, not only POR and hard pin reset as stated in the reference manual.
Workaround
Firmware should always update RMU_CTRL after a reset. Note that the LOCKUP reset is disabled by default.
Resolution
There is currently no resolution for this issue.

2.53 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.

2.54 RTCC_E201 – RTCC Does Not Support Compare/Capture Wrap with Prescaler

Description of Errata
When the RTCC is configured with a prescaler, the CCV1 top value enable feature enabled by setting CCV1TOP in RTCC_CTRL fails to wrap the counter when RTCC_CNT is equal to RTCC_CC1_CCV, as intended.
Affected Conditions / Impacts
Using CCV1TOP with a prescaled RTCC may result in the RTCC not wrapping at the desired time.
Workaround
Do not use a prescaler with the RTCC when using the CCV1TOP feature.
Resolution
There is currently no resolution for this issue.

2.55 RTCC_E202 – RTCC Triggers to LETIMER Not Safe

Description of Errata
The LETIMER inputs from the RTCC are connected to the LFECLK domain, while the LETIMER itself is clocked from the LFACLK domain. This results in synchronization issues between the RTCC inputs to the LETIMER and the LETIMER.
Affected Conditions / Impacts
RTCC triggers to LETIMER are not safe and can result in undefined behavior.
Workaround
Do not use RTCC triggers for LETIMER. PRS channels can be used as an alternative.
Resolution
There is currently no resolution for this issue.

2.56 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.

2.57 TIMER_E201 – Timer in Input Capture Mode Can Stop Counting

Description of Errata
When RISEA/FALLA is configured to RELOADSTART and then changed to any other mode at the same time as when a pulse from CC0 or PRS input occurs, the counter could stop running.
Affected Conditions / Impacts
The timer may not count properly in all situations.
Workaround
To prevent input pulses while changing RISEA/FALLA: 1. If using PRS, before changing RISEA/FALLA from RELOADSTART to any other value, change the input to some other PRS input by using PRSSEL in TIMERn_CC0_CTRL. After clearing RISEA/FALLA, set PRSSEL back to the original value 2. If using CC0 in, set GPIO mode to DISABLE for that pin before changing RISEA/FALLA. Then, set it back to INPUT mode after changing RISEA/FALLA.
Resolution
There is currently no resolution for this issue.

2.58 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.

2.59 USART_E202 — Incorrect 8-bit Timer Operation in Asynchronous Mode

Description of Errata
The 8-bit Timer logic that creates events within the USART works correctly in synchronous (USART) mode, but is not correct for asynchronous (UART) mode. As a result, it is not recommended to use the 8-bit Timer feature with asynchronous mode on affected devices.
Affected Conditions / Impacts
Systems using the USART module in asynchronous (UART) mode should not use the 8-bit Timer feature.
Workaround
There is currently no workaround for this issue.
Resolution
There is currently no resolution for this issue.

2.60 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.

2.61 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.

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

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

2.63 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.

2.64 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.

3. Revision History

Revision 0.4

September, 2020

- Added I2C_E202, I2C_E203, I2C_E205, I2C_E206, I2C_E207, TIMER_E202, USART_E203, USART_E204, USART_E205, USART_E206 and WDOG_E201.
- Migrated to new errata document format.

Revision 0.3

March 2018

- Updated the workaround description for EMU_E210.
- Updated the description and workaround description for RMU_E202.
- Updated revision history format.

Revision 0.2

July 21st, 2017

- Updated the workaround description for EMU_E208.
- Added ADC_E224, ADC_E226, ADC_E227, DBG_E204, DCDC_E205, DCDC_E206, EMU_E215, EMU_E216, RMU_E202, RTCC_E203, and USART_E202.
- Merged errata history and errata into one document.

Revision 0.1

November 11th, 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 LABS

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

http://www.silabs.com