EFM8 Busy Bee
EFM8BB2 Errata

This document contains information on the EFM8BB2 errata. The latest available revision of this device is revision C.

For errata on older revisions, refer to the errata history section for the device. The revision information is typically specified in or near the trace code on the device. Refer to the package marking information in the data sheet for more information.

Errata effective date: May, 2019.

1. Active Errata Summary

These tables list all known errata for the EFM8BB2 and all unresolved errata in revision C of the EFM8BB2.

Table 1.1. Errata History Overview

Designator	Title/Problem	Exists on Revision:	
		B	C
BL_E101	UART Bootloader Not Available	X	X
PWR_E101	Shutdown Mode Current Spike	X	—
TIMER_E101	Timer 3/4 Chaining Mode in Suspend	X	—
UART1_E101	Some Data Patterns Cause Inadvertent LIN Break Detection	X	X
WDT_E101	Restrictions on Watchdog Timer Refresh Interval	X	X
WDT_E102	Restrictions on changing Watchdog Timer Interval	X	X

Table 1.2. Active Errata Status Summary

Errata #	Designator	Title/Problem	Workaround Exists	Affected Revision	Resolution
1	UART1_E101	Some Data Patterns Cause Inadvertent LIN Break Detection	Yes	C	—
2	WDT_E101	Restrictions on Watchdog Timer Refresh Interval	Yes	C	—
3	WDT_E102	Restrictions on changing Watchdog Timer Interval	Yes	C	—

2. Detailed Errata Descriptions

2.1 UART1_E101 – Some Data Patterns Cause Inadvertent LIN Break Detection

Description of Errata
If UART1 is used in LIN mode (LINMDE = 1 in UART1LIN), certain data patterns consisting of a byte whose MSBs resemble a start bit (e.g. 0b101xxxxx when accounting for LSB first transmission) followed by 0x0 are improperly detected as a LIN break sequence.
Affected Conditions / Impacts
Because LIN frames can have a variable length of 2, 4, or 8 bytes, the detection of a break when it is not actually sent on the LIN bus could result in application software expecting the arrival of a new frame. Furthermore, if autobaud is enabled (AUTOBDE = 1 in UART1LIN), the detection of a break at the wrong time would result in the interpretation of the next character as the sync character, which is not likely to be the required 0x55. By attempting to sync on the wrong character, the baud rate determined would be wrong, and communication with the master would be lost due to baud rate mismatch.
Workaround
There is currently no workaround for this issue. **Note:** The inadvertent triggering of the autobaud logic due to improper detection of break does not apply when UART1 is not being used in LIN mode. By following the procedure described in the reference manual, whereby it is enabled pending detection of the sync character and then disabled after the sync character is received, UART1 autobaud detection functions as expected.
Resolution
There is currently no resolution for this issue.

2.2 WDT_E101 – Restrictions on Watchdog Timer Refresh Interval

Description of Errata

If the Watchdog Timer (WDT) is enabled, firmware will periodically write an 0xA5 value to the WDTCN register to refresh the timer and prevent the watchdog reset from occurring. However, if firmware writes to WDTCN more than once during the same LFOSC0 clock period, the refresh signal may be canceled, resulting in an unintended watchdog reset when the timer expires.

Affected Conditions / Impacts

If firmware refreshes the watchdog more than once in the same LFOSC0 clock period, an unexpected watchdog reset can occur.

Workaround

Systems using the Watchdog Timer (WDT) should ensure that the WDT is refreshed no more than once per LFOSC0 clock period.

Firmware can do this by using timers to count LFOSC0 clock periods. There are three methods to accomplish this:

1. If Timer 3 is not already in use, set it up to capture on the LFOSC0 clock. In this mode, the value of the Timer 3 reload registers does not matter. Instead, the WDT refresh function should check for the 16-bit timer flag (TF3H) to be set in the reset watchdog function, which indicates that a capture event occurred. If the device has another timer that can capture on the LFOSC0 clock, then that timer may be used instead of Timer 3.

```
void refresh_wdt()
{
   // Only refresh if TF3H is set
   if (TMR3CN0 & (0x80))
   {
      WDTCN = 0xA5;
      TMR3CN0 &= ~(0x80);
   }
}
```

2. If any timer is already in use, is clocked from the LFOSC0, and the low overflow flag is not already in use, firmware can check the low byte overflow flag (TFnL) to ensure at least one clock period has passed. For example, using Timer 3:

```
void init_wdt()
{
  // whatever code needed to initialized watchdog

  // intentionally set the TF3L flag (assuming SFRPAGE is correct)
  TMR3CN0 |= 0x40;
}

void refresh_wdt()
{
  static uint8_t last_tmr3l = 0;

  if ( (TMR3CN0 & 0x40) || (last_tmr3l != TMR3L) )
  {
    WDTCN = 0xA5;
    TMR3CN0 &= ~0x40;
    last_tmr3l = TMR3L;
  }
}
```

3. If the application already has an accurate and reliable time base, use that timer to establish a minimum WDT refresh interval that is longer than one LFOSC0 clock period in duration, similar to method (2) above as appropriate.

See the Knowledge Base article on this errata for more information, including examples of these firmware workarounds: https://www.silabs.com/community/mcu/8-bit/knowledge-base.entry.html/2016/11/28/wdt_e101_-_restricti-Vqe5.

Note: The LFOSC0 does not halt while debugging. This can cause the timer overflow flag to be set more quickly than expected when debugging the watchdog refresh function.

Resolution

There is currently no resolution for this issue.

2.3 WDT_E102 – Restrictions on changing Watchdog Timer Interval

Description of Errata
A watchdog reset can occur when the Watchdog Timer (WDT) is disabled.
Affected Conditions / Impacts
If the WDT timeout interval is changed from a higher interval to a lower interval, regardless if the WDT is enabled or disabled, a watchdog reset can occur
Workaround
This can be resolved by refreshing and disabling the WDT before changing the WDT timeout interval from a higher interval to lower interval. Following is the sequence of code that needs to be followed when changing the WDT interval. **Note:** User must insert the code to wait. It is not explicity added in the above sequence as it depends on the divided LFOSC0 clock and the SYSCLK clock selected by the user.
Resolution
There is currently no resolution for this issue.

```
void change_interval()
{
   WDTCN = 0xA5;          // Refresh WDT
   // Insert code to wait for 2 divided LFOSC0 clock periods
   WDTCN = 0xDE;          // Disable WDT (first key)
   WDTCN = 0xAD;          // Disable WDT (second key)
   // Insert code to wait for 3 divided LFOSC0 clock periods
   WDTCN = WDT_interval   // Change the current WDT interval to a lower interval with the MSB cleared to 0
   // Insert code to wait for 1 SYSCLK clock period
}
```

3. Errata History

This section contains the errata history for EFM8BB2 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 Errata History Summary

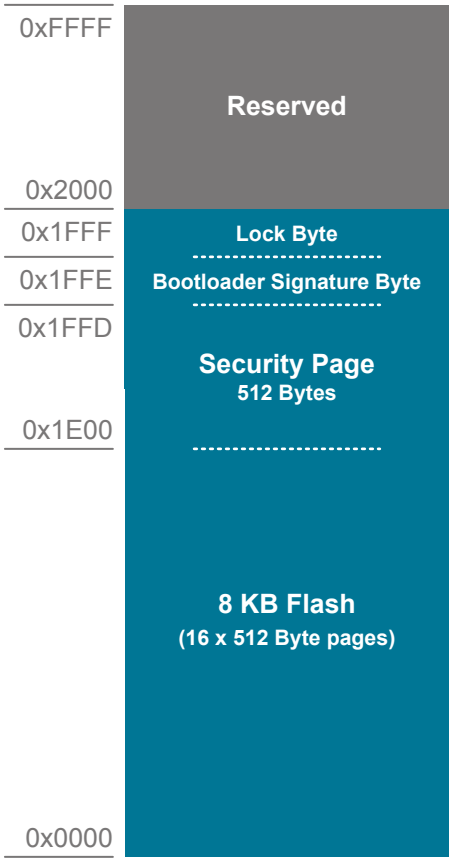
This table lists all resolved errata for the EFM8BB2.

Table 3.1. Errata History Status Summary

Errata #	Designator	Title/Problem	Workaround Exists	Affected Revision	Resolution
1	BL_E101	UART Bootloader Not Available	No	C	C date code 1601
2	PWR_E101	Shutdown Mode Current Spike	No	B	C
3	TIMER_E101	Timer 3/4 Chaining Mode in Suspend	Yes	B	C

3.2 Detailed Errata Descriptions

3.2.1 BL_E101 – UART Bootloader Not Available

Description of Errata
The data sheet mentions a UART bootloader in device flash. This bootloader is not available on revision B devices and revision C devices with date code prior to 1601.
Affected Conditions / Impacts
Systems intending to use a UART bootloader will need to implement and download a custom bootloader to the devices received from the factory. The factor booloader in AN945 will not work on revision B devices and revision C devices with date code prior to 1601. Devices with the factory bootloader and Bootloader Signature Byte support will use the byte immediately before the Lock Byte as a Bootloader Signature Byte to determine if the bootloader is present in flash. For example, in a device with 8 KB of flash:    For applications that do not use the bootloader, the Bootloader Signature Byte can be any value other than 0xA5 to enable normal operation. Note that the devices placed on a Starter Kit board may not have the Bootloader Signature Byte support included, so these parts may behave differently than loose parts ordered separately.
Workaround
A bootloader is not required for normal operation. However, if a bootloader is required by the application, a custom-written bootloader can be downloaded to devices received from the factory. The factory bootloader will not work on revision B devices and revision C devices with date code prior to 1601. Systems using the device should not write the Bootloader Signature Byte to 0xA5 when the intent is to not use the bootloader.
Resolution

This issue will be resolved in revision C devices with date code 1601 and later.

More information on the bootloader can be found in the device data sheet and in *AN945: "EFM8 Factory Bootloader User Guide"*. Application notes can be found on the Silicon Labs website (www.silabs.com/8bit-appnotes) and in Simplicity Studio using the [**Application Notes**] tile.

3.2.2 PWR_E101 – Shutdown Mode Current Spike

Description of Errata
Revision C devices placed in Shutdown mode will experience a momentary supply current spike after entering this low power mode. The magnitude of the spike is about 1.5 mA with a 3.6 V supply; it is smaller with lower supply voltages. This current spike will dissipate and the device will settle to the expected current consumption over time.
Affected Conditions / Impacts
Systems using the Shutdown energy mode will experience a momentary current spike before the system settles to the expected current consumption.
Workaround
There is currently no workaround for this issue.
Resolution
This issue is resolved in revision C devices.

3.2.3 TIMER_E101 – Timer 3/4 Chaining Mode in Suspend

Description of Errata
The Timer 3/4 32-bit counter on revision B devices will not switch to the low frequency oscillator (LFOSC0) after entering Suspend mode if the system clock divider is set to a value of divide-by-4 or greater.
Affected Conditions / Impacts
Systems using the Timer 3/4 32-bit counter in chained mode while in Suspend should use the recommended system clock divider settings to ensure proper operation.
Workaround
When using the Timer 3/4 32-bit counter in Suspend mode, set the system clock divider to the divide-by-1 or divide-by-2 settings before entering Suspend mode.
Resolution
This issue is resolved in revision C devices.

4. Revision History

Revision 0.7

May, 2019

- Added UART1_E101.

Revision 0.6

November, 2018

- Merged errata history and errata into one document.
- Moved BL_E101 from Active Errata Summary to Errata History.
- Updated the second workaround in WDT_E101.
- Updated Knowledge Base article link in WDT_E101.
- Added WDT_E102.

Revision 0.5

September, 2016

- Added WDT_E101.

Revision 0.4

November, 2015

- Updated UART Bootloader Not Available errata:
  - Added designator BL_E101.
  - Updated fixed revision to A, date code 1601 and later.
- Updated latest revision to C.
- Moved PWR_E101 and TIMER_E101 from the errata to the errata history.

Revision 0.3

June, 2015

- Updated UART Bootloader Not Available errata.
  - Updated affected condition with expected behavior.
  - Updated workaround with warning to not write 0xA5 to Bootloader Signature Byte.

Revision 0.2

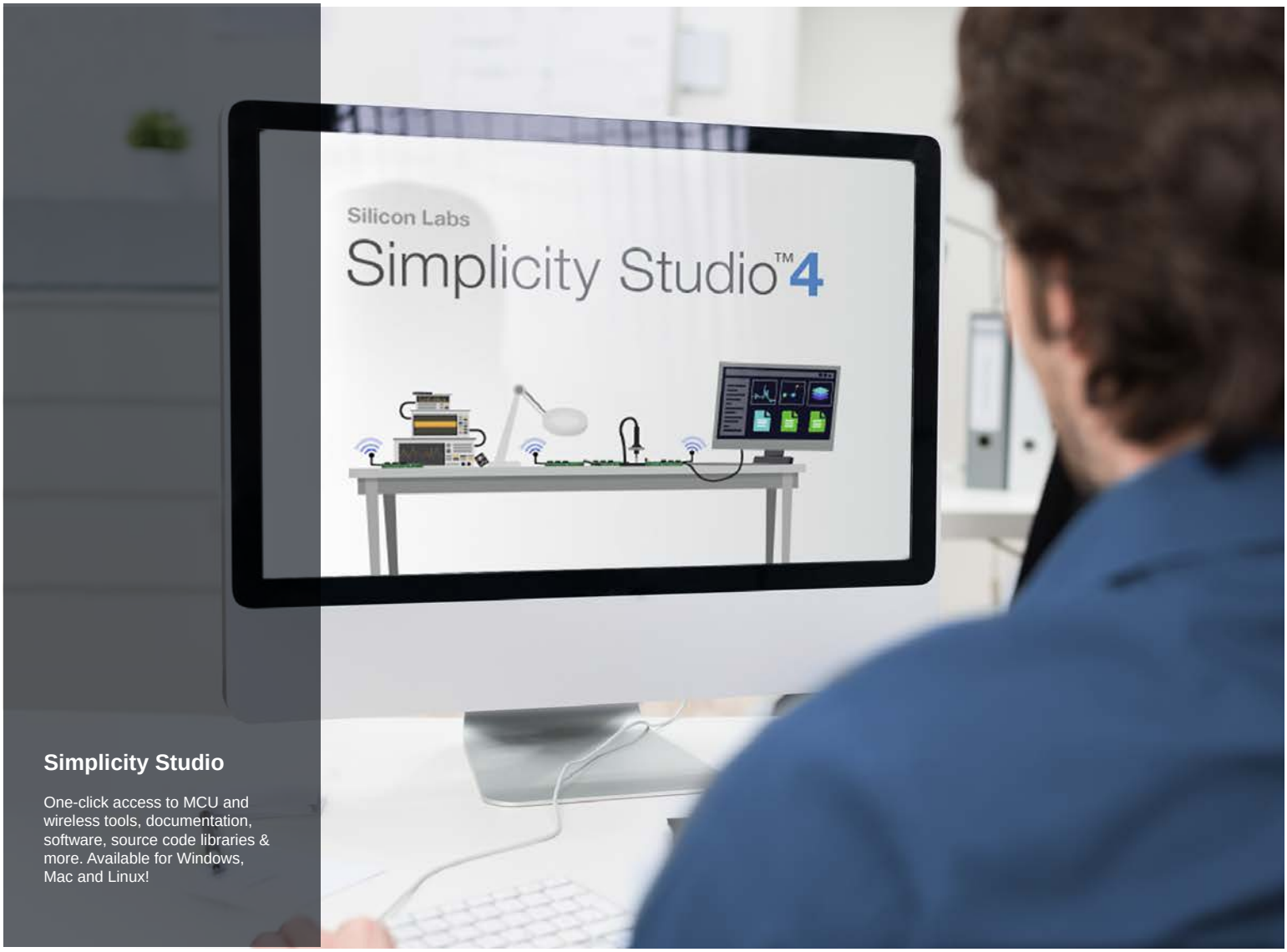
April, 2015

- Added Shutdown Mode Current Spike and Timer 3/4 Chaining Mode in Suspend errata.

Revision 0.1

February, 2015

- 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 and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com