8-bit MCU Family
C8051F93x/92x Errata

This document contains information on the errata of revision G of C8051F93x/92x.

For errata on older revisions, please refer to the errata history for the device. The device data sheet explains how to identify chip revision, either from package marking or electronically.

Errata effective date: January 12th, 2016.

1. Errata Summary

Table 1.1. Errata Status Summary

Errata #	Designator	Title/Problem	Workaround Exists	Affected Revision	Fixed Revision
1	SMBUS_E101	SMBus Hardware ACK behavior	Yes	G	—
2	CRC_E101	Writes to CRC0CN that Initiate a CRC0 Operation	Yes	G	—
3	FLASH_E101	Writes to FLSCL that Enable the Flash Read One-Shot Timer	Yes	G	—
4	PWR_E103	Reading PMU0CF upon Wakeup from Suspend Mode	Yes	G	—
5	RST_E102	VDD Monitor Disabled	Yes	G, date codes 1531 or earlier	G, date codes 1532 or later

2. Detailed Errata Descriptions

2.1 SMBUS_E101 – SMBus Hardware ACK behavior

Description of Errata

In some system management bus (SMBus) configurations, the Hardware Acknowledge mechanism of the SMBus peripheral can cause incorrect or undesired behavior. The Hardware Acknowledge mechanism is enabled when the EHACK bit (SMB0ADM.0) is set to logic 1.

The configurations to which these errata do not apply are as follows:

1. All SMBus configurations when Hardware Acknowledge is disabled
2. All **single-master** / **single-slave** SMBus configurations when Hardware Acknowledge is enabled and the MCU is operating as a **master or slave**.
3. All **multi-master** / **single-slave** SMBus configurations when Hardware Acknowledge is enabled and the MCU is operating as a **slave**.
4. All **single-master** / **multi-slave** SMBus configurations when Hardware Acknowledge is enabled and the MCU is operating as a **master**.

These errata only apply to the following configurations:

1. All **multi-slave** SMBus configurations when Hardware Acknowledge is enabled and the MCU is operating as a **slave**.
2. All **multi-master** SMBus configurations when Hardware Acknowledge is enabled and the MCU is operating as a **master**.

The following issues are present when operating as a **slave** in a **multi-slave** SMBus configuration:

1. When Hardware Acknowledge is enabled and SDA setup and hold times are not extended (EXTHOLD = 0 in the SMB0CF register), the SMBus hardware will always generate an SMBus interrupt following the ACK/NACK cycle of any slave address transmission on the bus, whether or not the address matches the conditions of SMB0ADR and SMB0MASK. The expected behavior is that an interrupt is only generated when the address matches.
2. When Hardware Acknowledge is enabled and SDA setup and hold times are extended (EXTHOLD = 1 in the SMB0CF register), the SMBus hardware will only generate an SMBus interrupt as expected when the slave address transmission on the bus matches the conditions of SMB0ADR and SMB0MASK. However, in this mode, the Start bit (STA) will be incorrectly cleared on reception of a slave address before software vectors to the interrupt service routine.
3. When Hardware Acknowledge is enabled and the ACK bit (SMB0CN.1) is set to 1, an unaddressed slave may cause interference on the SMBus by driving SDA low during an ACK cycle. The ACK bit of the unaddressed slave may be set to 1 if any device on the bus generates an ACK.

The following issue is present when operating as a **master** in a **multi-master** SMBus configuration:

If the SMBus master loses arbitration in a multi-master system, it may cause interference on the SMBus by driving SDA low during the ACK cycleof transfers which it is not participating. This will occur regardless of the state of the ACK bit (SMB0CN.1).

Affected Conditions / Impacts

When operating as a **slave** in a **multi-slave** SMBus configuration:

1. Once the CPU enters the interrupt service routine, SCL will be asserted low until SI is cleared, causing the clock to be stretched when the MCU is not being addressed. This may limit the maximum speed of the SMBus if the master supports SCL clock stretching. Incompliant SMBus masters that do not support SCL clock stretching will not recognize that the clock is being stretched. If the CPU issues a write to SMB0DAT, it will have no effect on the bus. No data collisions will occur.
2. Once the hardware has matched an address and entered the interrupt service routine, the firmware will not be able to use the Start bit to distinguish between the reception of an address byte versus the reception of a data byte. However, the hardware will still correctly acknowledge the address byte (SLA +R/W).
3. The SMBus master and the addressed slave are prevented from generating a NACK by the unaddressed slave because it is holding SDA low during the ACK cycle. There is a potential for the SMBus to lock up.

When operating as a **master** in a **multi-master** SMBus configuration:

The SMBus master and slave participating in the transfer are prevented from generating a NACK by the MCU because it is holding SDA low during the ACK cycle. There is a potential for the SMBus to lock up.

Workaround

When operating as a **slave** in a **multi-slave** SMBus configuration:

1. The SMBus interrupt service routine should verify an address when it is received and clear SI as soon as possible if the address does not match to minimize clock stretching. To prevent clock stretching when not being addressed, enable setup and hold time extensions (EXTHOLD = 1).
2. a. Detection of Initial Start — To distinguish between the reception of an address byte at the beginning of a transfer versus the reception of a data byte when setup and hold time extensions are enabled (EXTHOLD = 1), software should maintain a status bit to determine whether it is currently inside or outside a transfer. Once hardware detects a matching slave address and interrupts the MCU, software should assume a start condition and set the software bit to indicate that it is currently inside a transfer. A transfer ends any time the STO bit is set or on an error condition (e.g., SCL Low Timeout).
   b. Detection of Repeated Start — To detect the reception of an address byte in the middle of a transfer when setup and hold time extensions are enabled (EXTHOLD = 1), disable setup and hold time extensions (EXTHOLD = 0) upon entry into a transfer and re-enable setup and hold time extensions (EXHOLD = 1) at the end of a transfer.
3. Schedule a timer interrupt to clear the ACK bit at an interval shorter than 7 bit periods when the slave is not being addressed. For example, on a 400 kHz SMBus, the ACK bit should be cleared every 17.5 µs (or at 1/7 the bus frequency, 57 kHz). As soon as a matching slave address is detected (a transfer is started), the timer which clears the ACK bit should be stopped and its interrupt flag cleared. The timer should be re-started once a stop or error condition is detected (the transfer has ended).

A code example demonstrating these workarounds can be found in Simplicity Studio using the [**Software Examples**] tile. The example can also be found in the SMBus examples folder with the following default location:

```
C:\SiLabs\MCU\Examples\C8051F93x_92x\SMBus\F93x_SMBus_Slave_Multibyte_HWACK.c
```

The SMBus examples folder, along with examples for many additional peripherals, is created when the 8-bit Silicon Labs IDE is installed. The latest version of the IDE may be downloaded from the software downloads page http://www.silabs.com/8bit-software on the Silicon Labs website.

When operating as a **master** in a **multi-master** SMBus configuration:

Disable Hardware Acknowledge (EHACK = 0) when the MCU is operating as a master in a multi-master SMBus configuration.

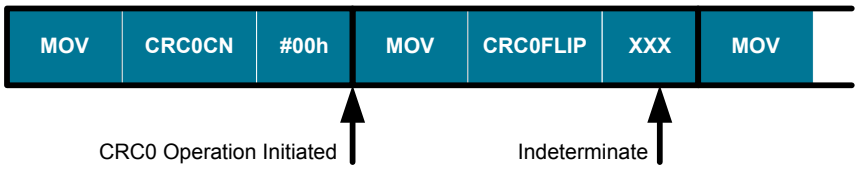
Resolution

There is currently no resolution for this issue.

2.2 CRC_E101 – Writes to CRC0CN that Initiate a CRC0 Operation

Description of Errata

The third op-code byte fetched from program memory following a write to CRC0CN that initiates a CRC0 operation is indeterminate.



Affected Conditions / Impacts

If the indeterminate op-code byte is the first or second byte in an instruction, improper code execution may result.

Workaround

Writes to CRC0CN that initiate a CRC0 operation must be immediately followed by a benign 3-byte instruction whose third byte is a "don't care." An example of such an instruction is the write of a dummy value to the CRC0FLIP register using a 3-byte MOV instruction. The value written to CRC0FLIP will be indeterminate, but this should have no effect on the system. To ensure that both instructions are executed without interruption, global interrupts should be disabled.

Note: When programming in C, the dummy value written to CRC0FLIP should be a non-zero value. This prevents the compiler from generating the following instruction sequence:

```
CLR A
MOV CRC0FLIP, A
```

When programming in C, the disassembly should be checked to ensure the compiler generated the following instruction sequence:

```
MOV CRC0FLIP, #AAh  ;where #AAh is the non-zero dummy value.
```

Resolution

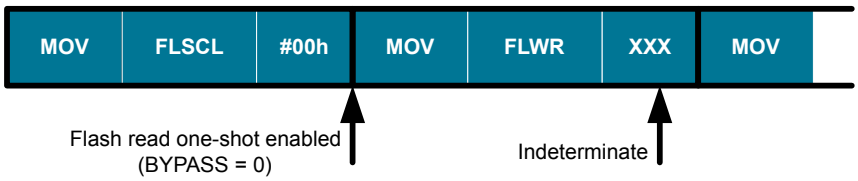
This behavior is described in revision 1.1 and later data sheets.

2.3 FLASH_E101 – Writes to FLSCL that Enable the Flash Read One-Shot Timer

Description of Errata

The Flash read one-shot timer is enabled on reset and reduces supply current when operating at system clock frequencies below 10 MHz. When operating at system clock frequencies above 10 MHz, power consumption can be minimized if the one-shot timer is disabled by setting the BYPASS bit (FLSCL.6) to logic 1. The Flash read one-shot timer should be re-enabled by clearing the BYPASS bit (FLSCL.6) to logic 0 when entering a low power mode or if the system clock frequency is reduced below 10 MHz.

The third op-code byte fetched from program memory following a write to FLSCL that enables the Flash read one-shot timer is indeterminate.



Affected Conditions / Impacts

If the indeterminate op-code byte is the first or second byte in an instruction, improper code execution may result.

Workaround

Writes to FLSCL that enable the Flash read one-shot timer must be immediately followed by a benign 3-byte instruction whose third byte is a “don’t care.” An example of such an instruction is the write of a dummy value to the FLWR register using a 3-byte MOV instruction. All writes to the FLWR register are “don’t care” and have not effect on the system. To ensure that both instructions are executed without interruption, global interrupts should be disabled.

Note: When programming in C, the dummy value written to FLWR should be a non-zero value. This prevents the compiler from generating the following instruction sequence:

```
CLR A
MOV FLWR, A
```

When programming in C, the disassembly should be checked to ensure the compiler generated the following instruction sequence:

```
MOV FLWR, #AAh ; where #AAh is the non-zero dummy value.
```

Resolution

This behavior is described in revision 1.1 and later data sheets.

2.4 PWR_E103 – Reading PMU0CF upon Wakeup from Suspend Mode

Description of Errata

Upon wakeup from suspend mode, the power management unit requires two system clocks in order to update the wake-up source flags in the PMU0CF register.

Affected Conditions / Impacts

If the PMU0CF register is read in the first two clock cycles following the wakeup from Suspend Mode, all wakeup source flags will contain a value of zero.

Workaround

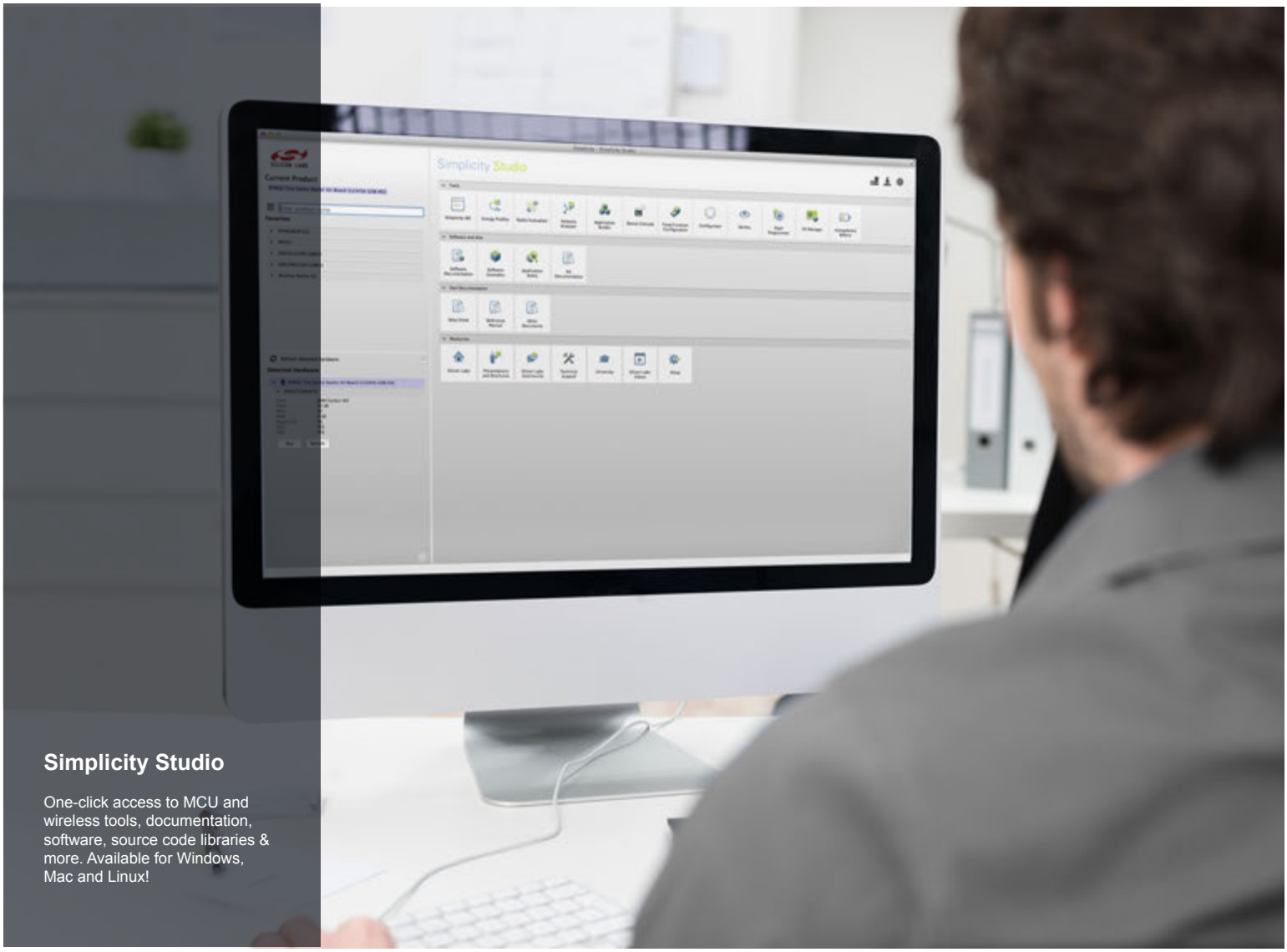
When waking up from suspend, wait at least 2 system clock cycles before reading the PMU0CF register to determine the cause of the wakeup.

Resolution

This behavior is described in revision 1.1 and later data sheets.

2.5 RST_E102 – VDD Monitor Disabled

Description of Errata
When a C8051F93x/92x device is subjected to a slowly decaying VDD ramp (for example, 100 µV/sec), oscillations on the /RST pin caused by the VDD monitor can result in the VDD monitor getting disabled. The oscillations result from the slow reaction time of the VDD monitor hysteresis circuit combined with VDD ripple caused by the changing device current demands as it transitions from its operating state to the reset state. The oscillation behavior is exacerbated by: 1. Slow VDD decay timing resulting from powering the device from a discharging battery or super capacitor, for example. 2. A high active supply current in comparison to the supply current of the device when it is held in reset. This can be caused by high system clock frequency or high GPIO sourcing load. 3. Device dependencies, with some part-to-part variations that affect the probability of the failure occurring.
Affected Conditions / Impacts
The VDD monitor enable bit is unique in that it is only affected by a power-on reset (POR) (which sets the bit to a ‘1’) and an SFR write, which can clear the bit to ‘0’ or set it to ‘1’ under software control. All other reset sources have no effect on the VDD monitor enable bit. Thus, if any action sets the bit to a ‘0’, it will remain ‘0’ until a POR occurs or software sets the bit to a ‘1’.
Workaround
Firmware can enable the VDD monitor as the first instruction executed after a reset. On systems written in C, this means editing the startup routine (i.e. STARTUP.A51 for Keil) to enable the VDD monitor as the first instruction. This will minimize any duration that the system is operating while the VDD monitor is disabled.
Resolution
Fixed in revision G devices with date codes of 1532 and later.



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