Debugging Techniques for the C8051F9xx and Si10xx Device Families

Relevant Devices

This application note applies to the following device families:
C8051F91x/90x, C8051F93x/2x, C8051F96x, C8051F99x/98x, Si100x, Si101x, and Si102x/3x

1. Introduction

The C8051F9xx and Si10xx families of MCUs enable operation down to 0.9 V and have an ultra-low-power Sleep mode. While this product family has many of the same features and debug capabilities of other Silicon Laboratories C8051F microcontrollers, there are some differences related to its ability to operate down to 0.9 V and support an ultra-low-power sleep mode that impact the debug process. This document highlights debugging techniques specific to the C8051F9xx and Si10xx families. Following the guidelines in this document will ensure a smooth debugging experience.

2. Development Tools

The C8051F9xx and Si10xx families do not require any special development tools other than those used for developing systems with any Silicon Laboratories C8051F microcontroller. However, it is important to ensure that you are using the latest version of the development tools provided by Silicon Laboratories.

The following development tool revisions are required to enable full debug capability on C8051F9xx and Si10xx devices:

- Silicon Labs IDE—V3.80 or later
- Keil µVision Driver—V3.20 or later
- Flash Programming Utilities—V3.50 or later
- Production Programming Utility—V2.00 or later

3. Measuring Supply Current

When designing low-power embedded systems, it is a good idea to measure the system supply current throughout development to ensure that the final product’s power consumption meets the design goals. Figure 1 shows a typical current meter modeled as a resistor to account for the meter’s internal resistance.

When the MCU enters its “Halt” debug state, supply current increases to approximately 5 mA. Depending on the source resistance of the current meter, this can cause a large voltage drop across the meter, which triggers an MCU brownout reset. Upon experiencing a brownout reset condition, the MCU and the IDE will become unsynchronized, and the debug session will be automatically ended by the IDE. Figure 2 shows the typical error message displayed after a brownout reset.

When establishing a debug connection to the MCU, it is recommended to place the current meter into its “mA” mode and disable the “auto ranging” feature in order to minimize the meter’s internal resistance and reduce the likelihood of triggering an MCU brownout reset.

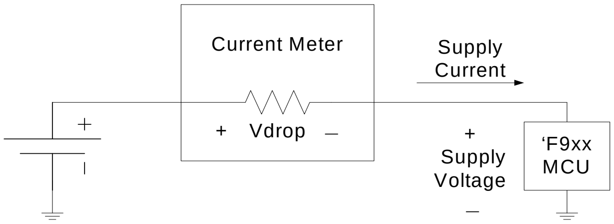
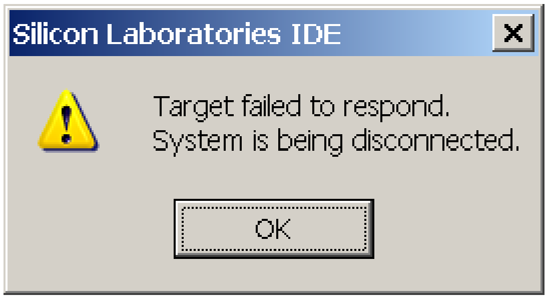


Figure 1. Internal Resistance of a Current Meter



Target failed to respond.
System is being disconnected.

Figure 2. IDE Error Message (Brownout Reset)

After a debug connection has been established, the Run/Stop button may be used to switch the MCU between the “Run” and “Halt” debug states. In the “Run” state, the Run/Stop button turns red, and the MCU supply current is determined by application code. The device may enter low-power modes, such as Suspend and Sleep, and supply current can drop below 1 µA while the MCU is in the “Run” debug state. In order to view the contents of memory or SFRs, the MCU must be placed in the “Halt” debug state. While halted, the Run/Stop button turns green, and the MCU supply current is approximately 5 mA.

Most current meters have a “µA” mode, which allows a small current to be measured with high resolution. This mode is very useful when measuring the Sleep or Suspend Mode current of the device. The internal resistance of the current meter is very high in this mode to allow an accurate current measurement. Before placing the current meter in its “uA” setting, make sure that the MCU is in the “Run” debug state (Run/Stop button is red) and that the MCU has entered its low power state.

The current meter should be switched back to its “mA” mode before waking the MCU up from its low-power state or placing it in the “Halt” debug state by pressing the Run/Stop button in the IDE while it is red.

When the debugger is connected, there will be an offset of approximately 3 uA on all current measurements. To remove this offset, disconnect from the target device in the IDE, and remove the 10-pin debug ribbon cable from the target board.

4. Sleep/Suspend Mode Debugging

Firmware that places the MCU in Sleep or Suspend mode may be debugged in the same way as firmware that does not place the MCU in Sleep or Suspend. A few precautions will ensure a smooth debugging experience:

- If firmware has code that places the MCU in Sleep mode, do not place a breakpoint on the instruction immediately following the write to PMU0CF that places the device into the low power mode. When waking up from Sleep mode, the MCU must execute at least one instruction before resuming full debug functionality.
- Upon wake-up from Sleep or Suspend mode, firmware should not allow the MCU to re-enter the low-power mode for a period of at least 15 µs if the wake-up source was a falling edge on the reset pin. The MCU requires 15 µs to respond to a pin reset or to detect C2 debug traffic. If software places the MCU back into the low-power mode in less than 15 µs, there is a possibility that pin reset and C2 debug events will not be detected.
- The transition from the “Run” debug state to the “Halt” debug state (in which all code execution is stopped) takes approximately 3–5 µs. During this time, the MCU is executing code. Under most debug conditions, a delay of 3–5 µs is not noticeable by the user. However, if firmware places the MCU in a low-power mode (e.g., Idle, Suspend, or Sleep) and the MCU is in the “Run” debug state, pressing the Run/Stop button in the IDE will cause the MCU to wake up from the low-power state and transition to the “Halt” debug state. Once the MCU fully transitions to the “Halt” debug state and the graphical user interface in the IDE is updated to match the MCU state, the user will notice that the PC counter did not stop at the instruction immediately following the instruction that placed the MCU in the low-power mode. Instead, the user will notice that the PC counter has executed 3–5 µs worth of instructions before stopping. To prevent the MCU from executing code after waking up from the low-power mode, place a breakpoint at the location where code execution should stop.
- On ‘F912, ‘F902, ‘F99x, ‘F98x, and Si101x devices, the POR supply monitor may be disabled to achieve an ultra-low sleep mode current. When disabled, any device reset will re-enable the supply monitor and trigger a power-on reset. This prevents the IDE from being able to connect to a device while its POR supply monitor is disabled. Applications that disable the POR supply monitor should insert a 3–5 second delay loop in software before disabling the VBAT supply monitor in order to be able to connect to the device using the IDE.

The proper way of connecting to a device when its POR supply monitor is disabled is:

1. Trigger any device reset (e.g., reset pin, etc).
2. Within 3–5 seconds, initiate a connection using the IDE. Note: If a reset is not triggered before attempting to connect, then the first connection attempt will fail. Subsequent connection attempts occurring within 3–5 seconds of each other should succeed.

5. One Cell Mode Debugging

Note: This section only applies to C8051F91x/90x, C8051F93x/2x, C8051F99x/98x, Si100x, and Si101x.

In one-cell mode, debugging should be fully functional given that the power supply source impedance is small enough to handle short bursts of high current flow. Since battery current increases as the V_{BAT} voltage decreases, it is recommended to debug with a V_{BAT} voltage of 1.5 V or higher, especially when a current meter is connected to the system at the same time as debugging with the IDE. This input supply voltage can be achieved by using fresh batteries or a bench power supply.

You can connect to the device using the IDE when the V_{BAT} supply voltage is less than 1.5 V; however, there is a possibility that the IDE will not be able to connect if the source impedance of the power supply becomes too high (e.g., current meter or batteries that are near expiration).

The VDD/DC+ supply voltage can be changed on-the-fly in software when the MCU is used in one-cell mode. The USB Debug Adapter measures the supply voltage of the device and tries to match it when generating logic HIGH and LOW values.

If the MCU supply voltage is increased rapidly (e.g., from 1.9 V to 3.3 V) while single-stepping through the code, the USB debug adapter may try to communicate with the MCU using the lower supply voltage. If the logic levels of the old supply voltage fall outside the VIH/VIL thresholds of the new supply voltage, communication may fail and the debug session may end. To avoid ending the debug session, avoid single-stepping through code that modifies the supply voltage. Instead, allow the MCU to be in the “Run” debug state while executing these instructions and for a period of one second following the supply voltage transition.

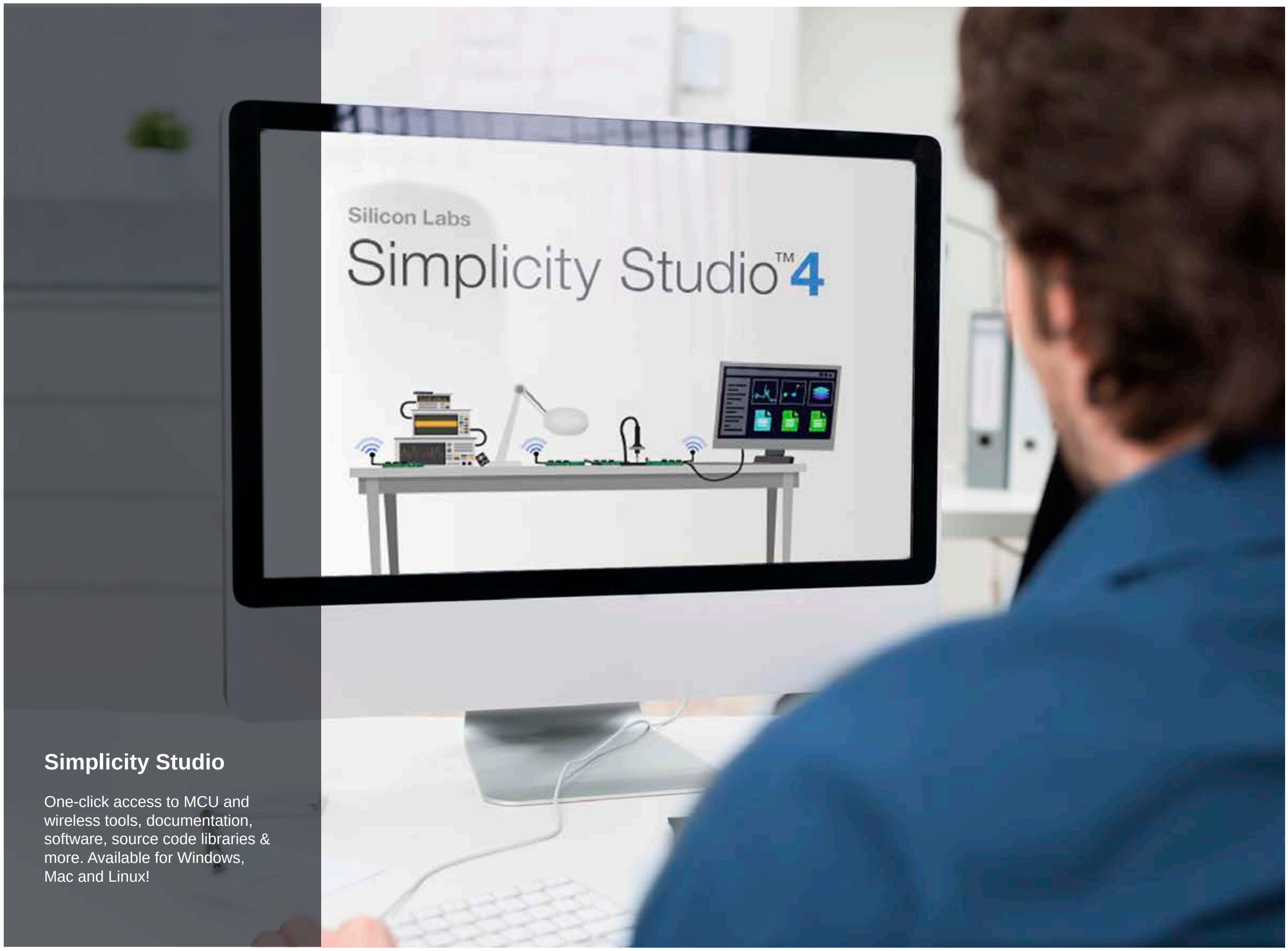
Document Change List

Revision 0.2 to Revision 0.3

- Added support for C8051F99x/98x devices.

Revision 0.3 to Revision 0.4

- Added support for 'F96x, Si100x, Si101x, and Si102x/3x devices.



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

Disclaimer
Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information
Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world’s most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

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

http://www.silabs.com