TOOLSTICK C8051F582 DAUGHTER CARD USER'S GUIDE

1. Handling Recommendations

To enable development, the ToolStick Base Adapter and daughter cards are distributed without any protective plastics. To prevent damage to the devices and/or the host PC, please take into consideration the following recommendations when using the ToolStick:

- Never connect or disconnect a daughter card to or from the ToolStick Base Adapter while the Base Adapter is connected to a PC.
- Always connect and disconnect the ToolStick Base Adapter from the PC by holding the edges of the boards.

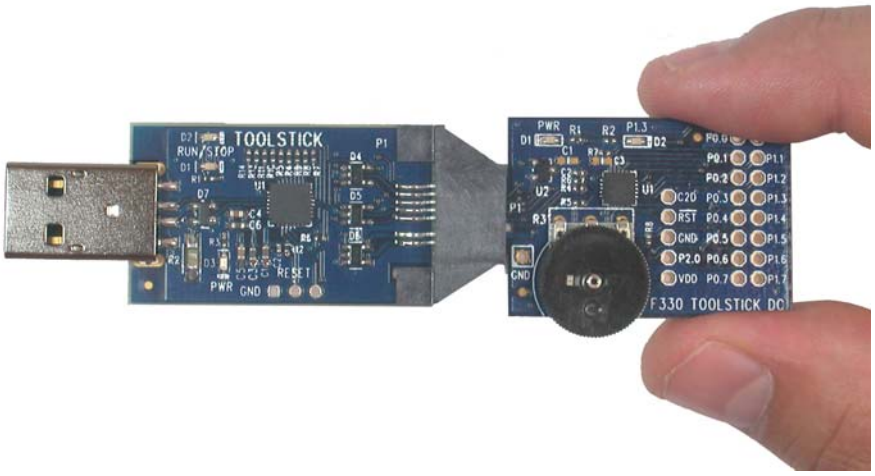


Figure 1. Proper Method of Holding the ToolStick

- Avoid directly touching any of the other components.

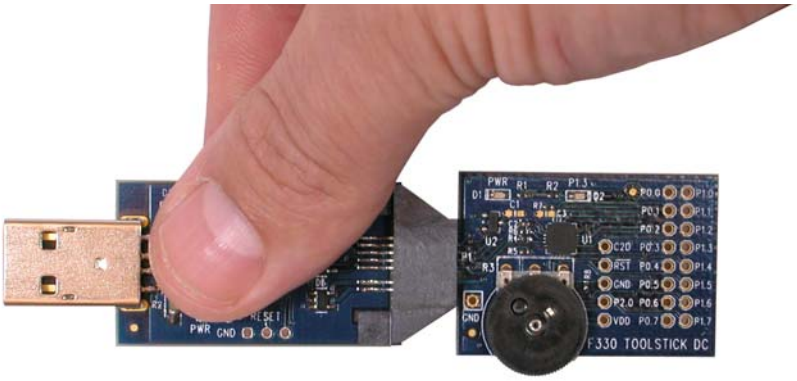


Figure 2. Improper Method of Holding the ToolStick

- Manipulate mechanical devices on the daughter cards, such as potentiometers, with care to prevent the Base Adapter or daughter card from accidentally dislodging from their sockets.

Copyright © 2008 by Silicon Laboratories

2. Contents

The ToolStick-F582DC kit contains the ToolStick C8051F582 Daughter Card.

A ToolStick daughter card requires a ToolStick Base Adapter to communicate with the PC. ToolStick Base Adapters can be purchased at www.silabs.com/toolstick.

3. ToolStick Overview

The purpose of the ToolStick is to provide a development and demonstration platform for Silicon Laboratories microcontrollers and to demonstrate the Silicon Laboratories software tools, including the Integrated Development Environment (IDE).

The ToolStick development platform consists of two components: the ToolStick Base Adapter and a daughter card. The ToolStick Base Adapter provides a USB debug interface and data communications path between a Windows PC and a target microcontroller.

The C8051F582 Daughter Card includes a CAN transceiver, a pair of LEDs, a potentiometer, a switch connected to a GPIO pin, and a small prototyping area which provides access to all of the pins of the device. This prototyping area allows additional hardware to be connected to the board to enable it as a development platform.

Figure 3 shows the ToolStick C8051F582 Daughter Card and identifies the various components.

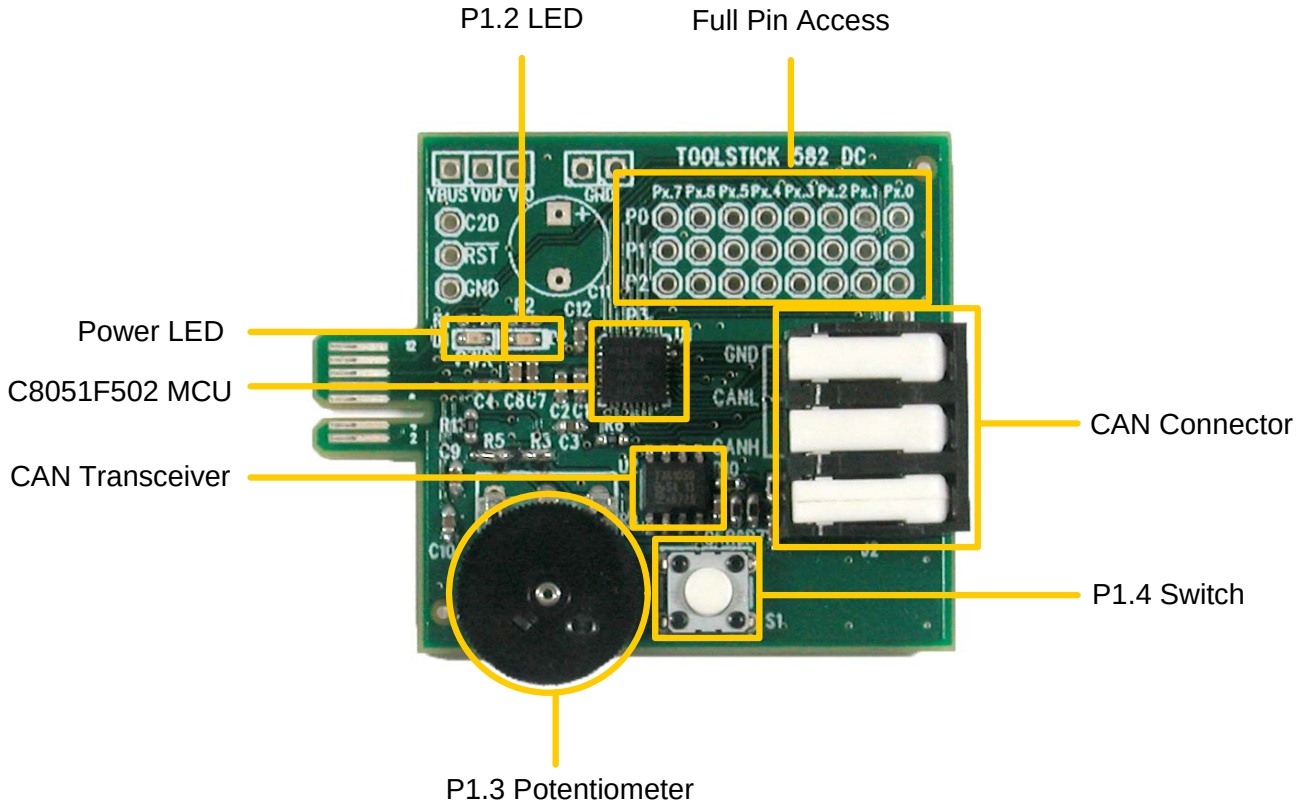


Figure 3. ToolStick C8051F582 Daughter Card

4. Getting Started

The necessary software to download, debug, and communicate with the target microcontroller must be downloaded from www.silabs.com/toolstick. The following software is necessary to build a project, download code to, and communicate with the target microcontroller:

- Silicon Laboratories Integrated Development Environment (IDE)
- Keil Demonstration Tools
- ToolStick Terminal application

The Keil Demo Tools include a compiler, assembler, and linker. See “5.2. Keil Demonstration Toolset” for more details about the demo tools. ToolStick Terminal communicates with the target microcontroller's UART through the ToolStick Base Adapter. It can also read/write two of the GPIO pins available on the ToolStick Base Adapter.

Other useful software that is provided on the Silicon Labs Downloads (www.silabs.com/mcudownloads) website includes:

- Configuration Wizard 2
- Keil uVision2 and uVision3 Drivers

The software described above is provided in several download packages. The ToolStick download package includes example code, documentation including user’s guides and data sheets, and the ToolStick Terminal application. The IDE, Keil Demonstration Tools, Configuration Wizard 2, and the Keil µVision Drivers are available as separate downloads. After downloading and installing these packages, see the following sections for information regarding the software and running one of the demo applications.

5. Software Overview

5.1. Silicon Laboratories IDE

The Silicon Laboratories IDE integrates a source-code editor, source-level debugger, and an in-system Flash programmer. See “6. ToolStick C8051F582 Daughter Card Features Demo” for detailed information on how to use the IDE. The Keil Demonstration Toolset includes a compiler, linker, and assembler and easily integrates into the IDE. The use of third-party compilers and assemblers is also supported.

5.1.1. IDE System Requirements

The Silicon Laboratories IDE requirements include:

- Pentium-class host PC running Microsoft Windows 2000 or Windows XP
- One available USB port

5.1.2. Third Party Toolsets

The Silicon Laboratories IDE has native support for many 8051 compilers. The full list of natively supported tools is:

- Keil
- IAR
- Raisonance
- Tasking
- Hi-Tech
- SDCC

Please note that the demo applications for the C8051F582 Daughter Card are written for the Keil toolset.

5.2. Keil Demonstration Toolset

5.2.1. Keil Assembler and Linker

The Keil demonstration toolset assembler and linker place no restrictions on code size.

5.2.2. Keil Demonstration C51 C Compiler

The evaluation version of the C51 compiler is the same as the full version with the following limitations:

- Maximum 4 kB code generation
- There is no floating point library installed
- When initially installed, the C51 compiler is limited to a code size of 2 kB, and programs start at code address 0x0800. Refer to “AN104: Integrating Keil Tools into the Silicon Labs IDE” for instructions to change the limitation to 4 kB and have the programs start at code address 0x0000.

5.3. Configuration Wizard 2

The Configuration Wizard 2 is a code generation tool for all of the Silicon Laboratories devices. Code is generated through the use of dialog boxes for each of the device's peripherals.

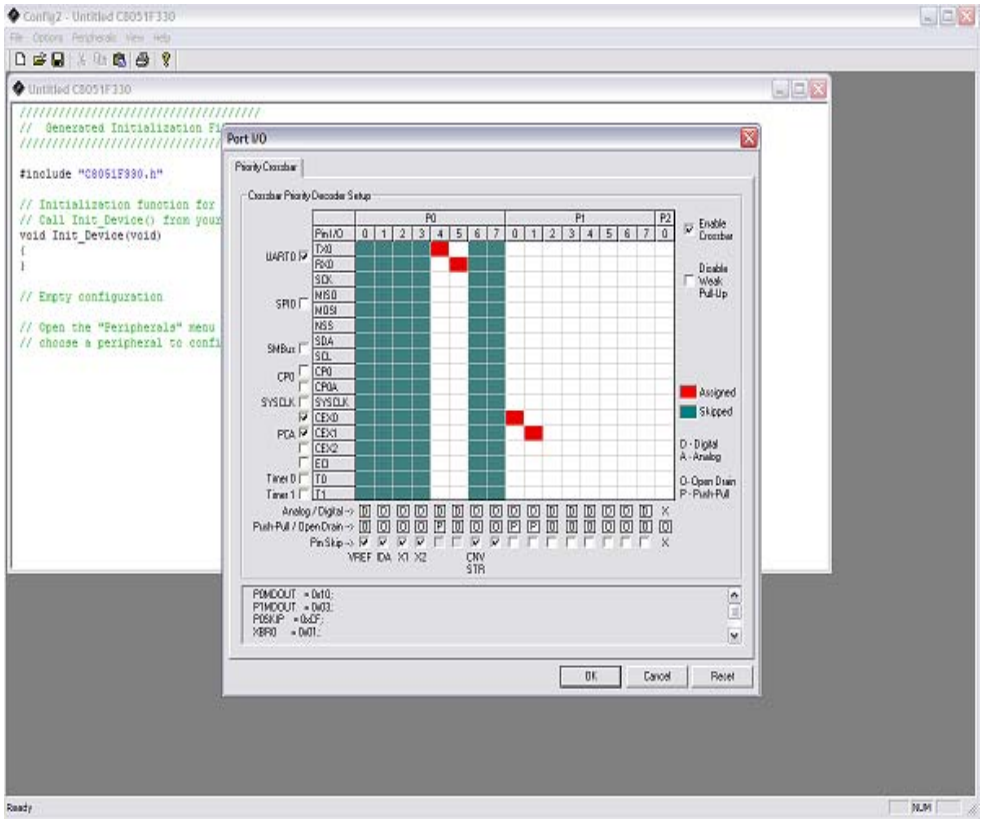


Figure 4. Configuration Wizard 2 Utility

The Configuration Wizard 2 utility helps accelerate development by automatically generating initialization source code to configure and enable the on-chip resources needed by most design projects. In just a few steps, the wizard creates complete startup code for a specific Silicon Laboratories MCU. The program is configurable to provide the output in C or assembly.

For more information, please refer to the Configuration Wizard 2 documentation. The documentation and software are available from the Downloads webpage (www.silabs.com/mcudownloads).

5.4. Keil µVision2 and µVision3 Silicon Laboratories Drivers

As an alternative to the Silicon Laboratories IDE, the µVision debug driver allows the Keil µVision2 and µVision3 IDEs to communicate with Silicon Laboratories on-chip debug logic. In-system Flash memory programming integrated into the driver allows for rapidly updating target code. The µVision2 and µVision3 IDEs can be used to start and stop program execution, set breakpoints, check variables, inspect and modify memory contents, and single-step through programs running on the actual target hardware.

For more information, please refer to the µVision driver documentation. The documentation and software are available from the Downloads webpage (www.silabs.com/mcudownloads).

5.5. ToolStick Terminal

The ToolStick Terminal program provides the standard terminal interface to the target microcontroller's UART. However, instead of requiring the usual RS-232 and COM port connection, ToolStick Terminal uses the USB interface of the ToolStick Base Adapter to provide the same functionality.

In addition to the standard terminal functions (send file, receive file, change baud rate), two GPIO pins on the target microcontroller can be controlled using the Terminal for either RTS/CTS handshaking or software-configurable purposes (see the demo software for an example).

See Section "6.8. Using ToolStick Terminal," on page 12 for more information. The software is available on the ToolStick webpage (www.silabs.com/toolstick).

6. ToolStick C8051F582 Daughter Card Features Demo

The ToolStick kit includes a few simple code examples. The example described in this section is titled **F582DC_FeaturesDemo**. The purpose of this example is to guide a new user through the features and capabilities of the IDE and demonstrate the microcontroller's on-chip debug capabilities. The **F582DC_FeaturesDemo** example code uses the potentiometer on the daughter card to vary the blinking rate of the LED. The first part of this demo shows how to use the IDE to connect and download the firmware, view and modify registers, and use watch windows, breakpoints, and single step through code. The second part of the demo shows how to use the ToolStick Terminal to receive UART data from the daughter card and how to use the GPIO pins.

6.1. Hardware Setup

Connect the ToolStick hardware to the PC using the steps below while taking note of the recommendations in Section 1:

1. Connect the ToolStick Base Adapter to the ToolStick C8051F582 Daughter Card.
2. If available, connect the USB extension cable to the ToolStick Base Adapter.
3. Connect the ToolStick to a USB port on a PC.

See Figure 5 below for a hardware setup example using the C8051F330 ToolStick Daughter Card.

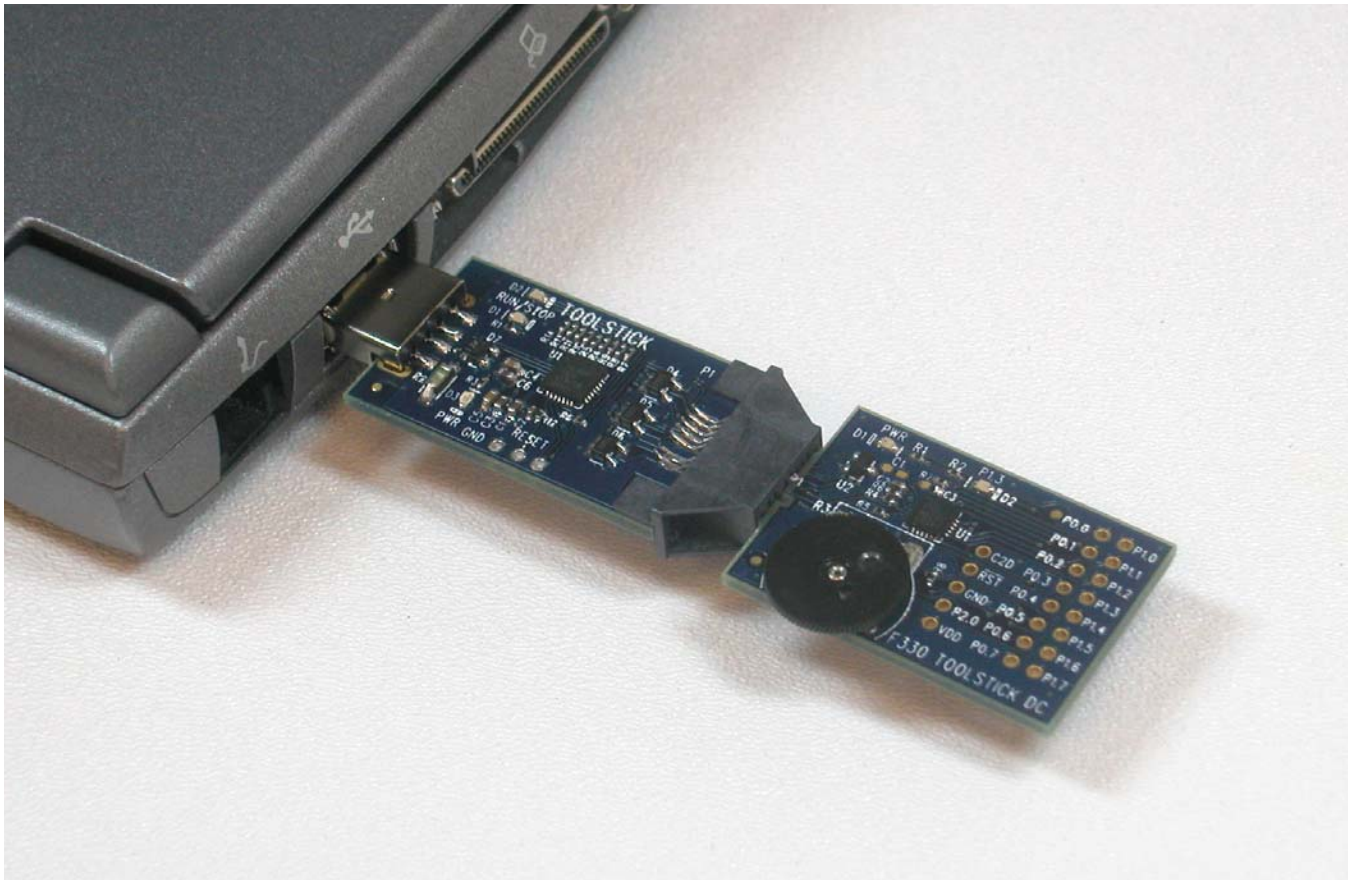


Figure 5. Hardware Setup Example

6.2. Connecting to the Device and Downloading Firmware

This section describes how to open the IDE, open and build a project, connect to a device and download the firmware.

1. Open the Silicon Laboratories IDE from the **Start** → **Programs** → **Silicon Laboratories** menu.
2. In the IDE, go to **Project** → **Open Project**.
3. Browse to *C:\SiLabs\MCU\ToolStick\F582DC\Firmware*.
4. Select **F582DC_FeaturesDemo.wsp** and click OK.
5. In the IDE, select **Project** → **Rebuild Project**.
6. Go to **Options** → **Connection Options**.
7. Select "**USB Debug Adapter**" for the Serial Adapter and "**C2**" for the Debug Interface, and then click "OK".
8. Go to **Debug** → **Connect**.
9. Download the code using the **download button** on the menu bar or use alt-D.

Once these steps are completed, the firmware is built into an object file (step 5) and downloaded to the device (step 9). The device is now ready to begin executing code. If all of these steps are followed successfully, the "Go" option is enabled in the Debug menu. A green circle icon in the IDE toolbar also indicates that the device is ready to run. If one of the steps leads to an error, make sure that the ToolStick is properly inserted in a USB port and start again with step 6.

6.3. Running and Stopping Code Execution

Once the IDE is connected to the device and the firmware is loaded, the IDE can start and stop the code execution. The following steps can be performed using the buttons on the toolbar or using the options in the Debug menu.

1. To start code execution, click the green "**Go**" button on the toolbar or use the **Debug** → **Go** menu option. The green LED on the daughter card will start to flash. The debug commands in the IDE (single-step, multiple-step, set breakpoint, and others) are disabled when the device is running. While the firmware is running, the potentiometer on the daughter card can be rotated to alter the blinking speed of the LED. The button labeled S1 can also be pressed to toggle the ADC on and off. When the ADC is off, the blink rate or brightness of the LED will not change with the potentiometer.



2. To stop code execution, click the red "**Stop**" button on the toolbar or use the **Debug** → **Stop** menu option. The device will halt code execution and all of the registers and pins on the device will hold their state.

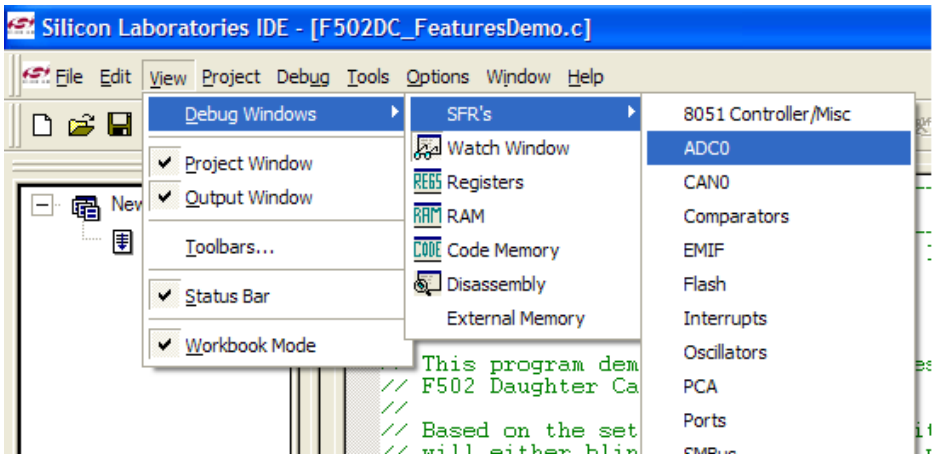


All debug windows and watch windows are refreshed when the device is stopped. If any of the values in these windows have changed since the last time the device was halted, the new value is shown in red text instead of black text.

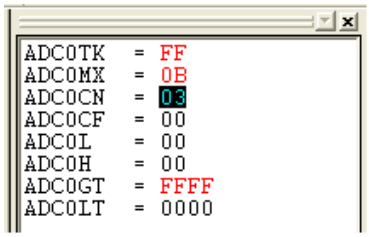
6.4. Viewing and Modifying Registers

All registers on the device can be viewed and modified when the device is in a halted state. The registers are grouped together according to which peripheral or part of hardware they belong. As an example, this guide shows how to open the ADC0 Debug Window and disable the ADC0 directly from the IDE.

1. Open the ADC0 Debug Window from the **View** → **Debug Windows** → **SFR's** → **ADC0** menu option. The ADC0 Debug Window appears on the right-hand side of the IDE. In this window, the ADC0CN register is shown. This register is used to enable and configure the on-chip ADC. When the firmware is running, the ADC0CN register reads as 0x83 indicating that the ADC is running.



2. In the debug window, change the value of **ADC0CN** from 0x83 to 0x03. This value turns off the ADC on the target microcontroller.



3. To write this new value to the device, select **Refresh** from the Debug Menu or click the **Refresh** button in the toolbar.



4. Click "**Go**" to resume running the device with the new **ADC0CN** value.
5. Turn the potentiometer on the daughter card and notice that it has no effect on the blinking rate of the LED.
6. Re-enable the ADC by writing 0x83 to the **ADC0CN** and clicking the **Refresh** button.

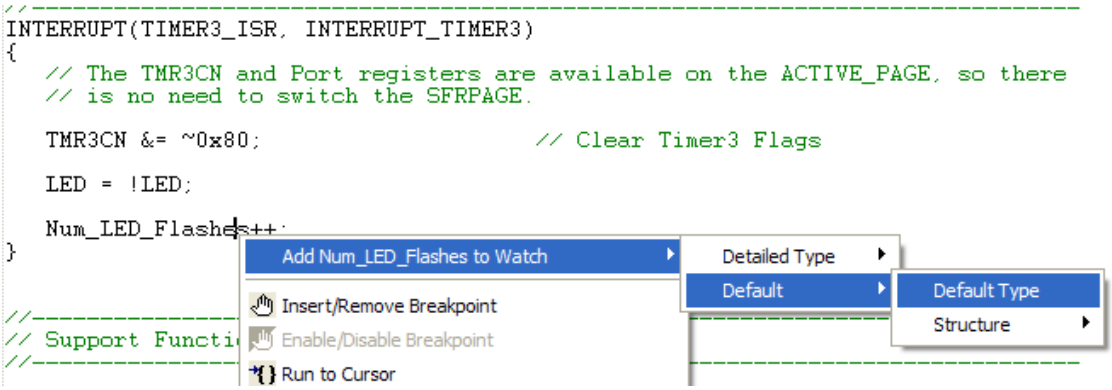
Changing the values of registers does not require recompiling the code and redownloading the firmware. At any time, the device can be halted and the values of the registers can be changed. After selecting "Go", the firmware will continue execution using the new values. This capability greatly speeds up the debugging process. See the data sheet for the C8051F582 device for the definitions and usage for all registers.

The debug windows for other sets of registers are found in the **View** → **Debug Windows** → **SFR's** menu.

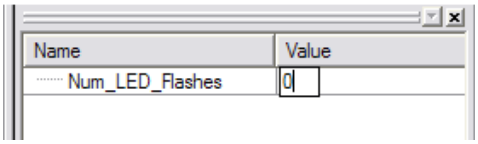
6.5. Enabling and Using Watch Windows

The Debug Windows in the View menu are used to view and modify hardware registers. To view and modify variables in code, the IDE provides Watch Windows. Just as with register debug windows, variables in the watch windows are updated each time the device is halted. This section of the User's Guide explains how to add a variable to the watch window and modify the variable. In the F582_FeatureDemo example code, the variable **Num_LED_Flashes** is a counter that stores the number of times the LED blinks.

1. If the device is running, stop execution using the "**Stop**" button or use the **Debug** → **Stop** menu option.
2. In the File View on the left-hand side of the IDE, double-click on **F582DC_FeaturesDemo.c** to open the source file.
3. Scroll to the **Timer3_ISR()** function and right-click on the variable "**Num_LED_Flashes**." In the context menu that appears, select "**Add Num_LED_Flashes to Watch**" and then choose "**Default** → **Default Type**." On the right-hand portion of the IDE, the watch window appears and the variable is added. The current value of the variable is shown to the right of the name.



4. **Start** and **stop** the device a few times. See that the value of the **Num_LED_Flashes** is incremented each time the LED blinks.
5. When the device is halted, click on the value field in the watch window and change the value to 0. Then click the **Refresh** button or select **Debug** → **Refresh** to write the new value to the device.



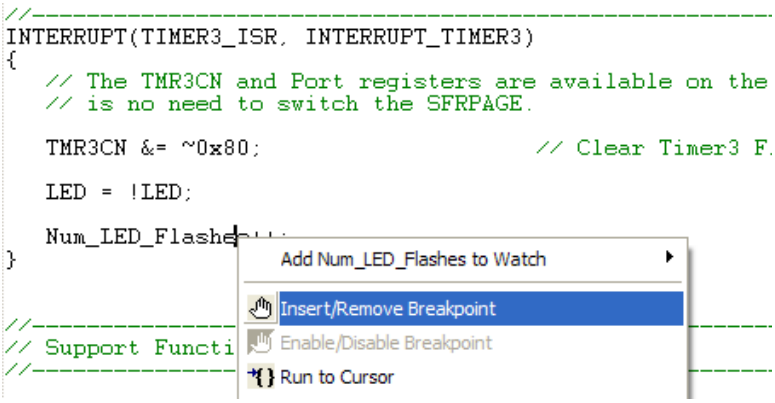
6. **Start** and **stop** the device a few times to watch the variable increment starting at 0.

Changing the values of variables does not require recompiling the code and redownloading the firmware. At any time, the device can be halted and the values of the variables can be changed. The firmware will continue execution using the new values.

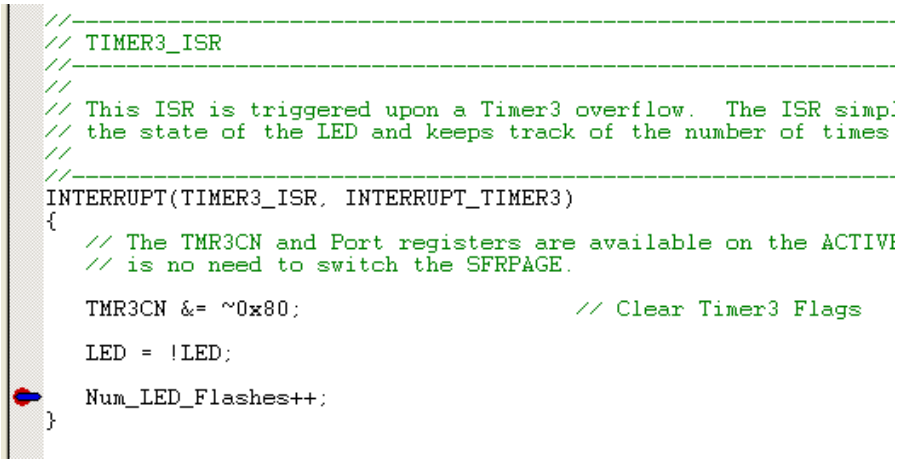
6.6. Setting and Running to Breakpoints

The Silicon Laboratories microcontroller devices support up to four hardware breakpoints. A breakpoint is associated with a specific line of code. When the processor reaches a hardware breakpoint, the code execution stops, and the IDE refreshes all debug and watch windows. The on-chip debug hardware allows for breakpoints to be placed on any line of executable code, including code in Interrupt Service Routines. This section provides steps to set a breakpoint on the line of source code that increments the **Num_LED_Flashes** variable.

1. If the device is running, stop execution using the "**Stop**" button or use the **Debug** → **Stop** menu option.
2. Scroll to the **Timer3_ISR()** function and right-click on the variable "**Num_LED_Flashes**". In the context menu that appears, select "**Insert/Remove Breakpoint**." On the left side of the line in the editor window, a red circle is added to indicate a breakpoint is placed on the source line.



3. Click the "**Go**" button or select the **Debug** → **Go** menu option.
4. After a short time, the IDE will show that the device is halted. A blue line will be placed in the editor window to indicate where the code execution has stopped.

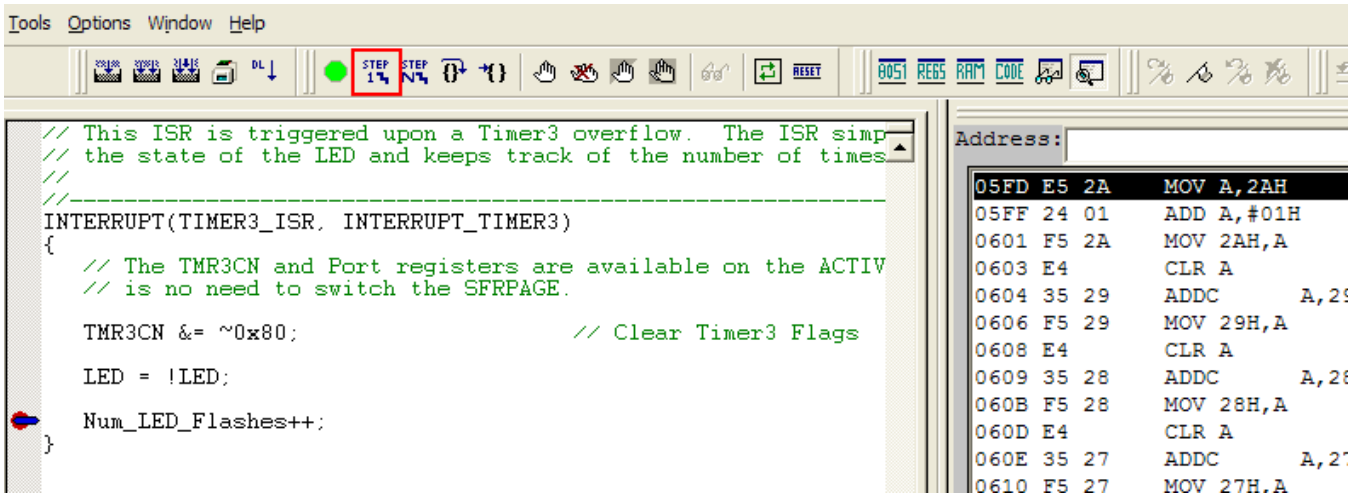


5. **Start** and **stop** the processor a few more times. Notice that the LED blinks once for every time the processor is started and the **Num_LED_Flashes** variable also increments by one.

6.7. Single-Stepping Through Firmware

The IDE supports the ability to single-step through firmware one assembly instruction at a time. The IDE reads the Flash from the device, converts the instructions to assembly and displays them in a disassembly window. The following steps show how to open the disassembly window and single step through firmware.

1. If there is already not a breakpoint set on line of code that increments the **Num_LED_Flashes** variable, set the breakpoint using the steps described in Section 6.6.
2. Start the processor using the "**Go**" button and wait till it stops on the breakpoint.
3. Select **View** → **Debug Windows** → **Disassembly**. The disassembly window will appear on the right-hand side of the IDE, if it is not already open.
4. To execute one assembly instruction at a time, click the "**Step**" button on the toolbar or select the **Debug** → **Step** menu option. The highlighted line in the disassembly window indicates the next instruction to be executed. The blue line marker in the editor window will stay on the same .C source line until all of the assembly instructions are completed.

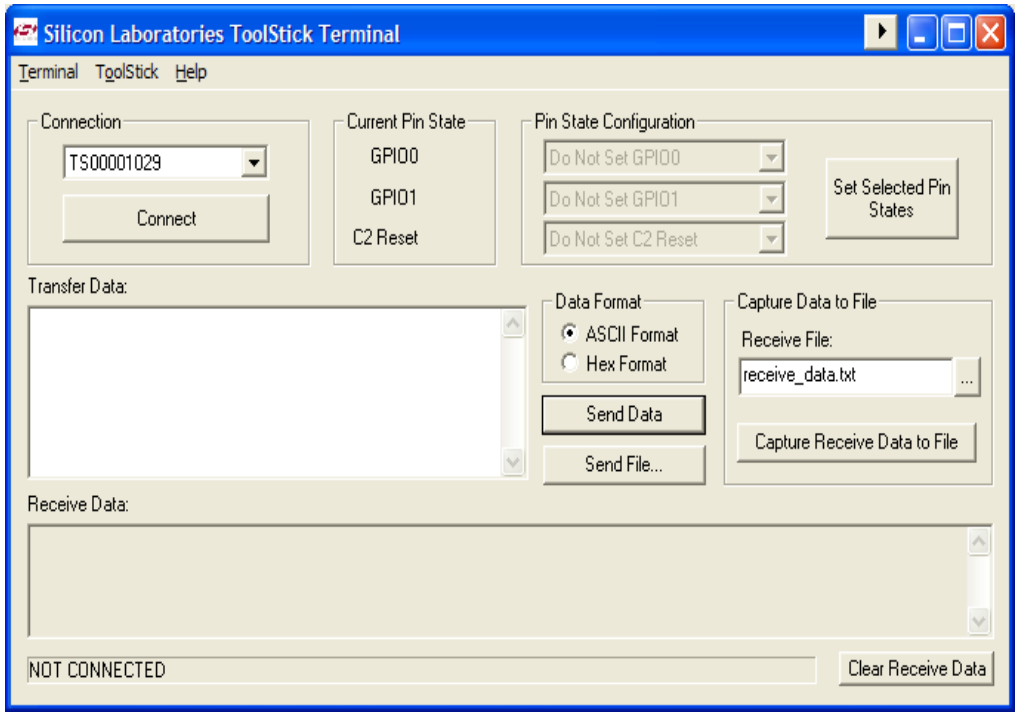


The disassembly window has three columns. The left column is the address of the instruction in Flash. The middle column is the instruction in hex. The right column is the disassembled instruction. The Disassembly debug window and the capability to single-step through firmware allows a developer to see exactly what instructions are executed and their output.

6.8. Using ToolStick Terminal

This section describes how to use ToolStick Terminal to communicate with UART from the PC to the daughter card through the ToolStick Base Adapter.

1. If the Silicon Laboratories IDE is open, close the IDE. The IDE and the ToolStick Terminal cannot communicate with the daughter card simultaneously.
2. Open ToolStick Terminal from the **Start** → **Programs** → **Silicon Laboratories** menu.



3. Go to the **ToolStick** → **Settings** menu.
4. Under "Pin Settings", change GPIO0 / RTS to "**GPIO Output - Push Pull**" and click "OK." The rest of the default settings are correct for the C8051F582 Features Demo.
5. In the top, left-hand corner of the Terminal application, available devices are shown in the drop-down Connection menu. Click "**Connect**" to connect to the device. In the "**Receive Data**" window, text indicating the blink rate of the LED will appear.
6. Turn the potentiometer on the daughter card and see that the blink rate is updated on the daughter card and the new blink rate is printed to the Terminal.

In addition to the standard two UART pins (TX and RX), there are two GPIO/UART handshaking pins on the ToolStick Base Adapter that are connected to two port pins on the target microcontroller. ToolStick Terminal is used to configure and read/write these pins. For the **F582DC_FeaturesDemo**, one of these GPIO pins is connected to an external interrupt pin on the C8051F582. The following steps describe how to change the level of one of the GPIO pins and trigger an interrupt on the target microcontroller. The interrupt forces the firmware to switch modes and send a pulse-width modulated (PWM) signal to the LED instead of blinking the LED using an on-chip Timer.

1. In ToolStick Terminal, under Pin State Configuration, select "**Set GPIO0 Logic Low**" and click on "**Set Selected Pin States**." This changes the level of the GPIO0 pin from **Logic High** to **Logic Low** and triggers a level-sensitive interrupt on the microcontroller.
2. In the Receive window, see that the printed text has changed to indicate the LED PWM duty cycle.
3. Turn the potentiometer on the daughter card to change the brightness of the LED on the daughter card.
4. Change the GPIO0 pin state back to **Logic High** and notice that the firmware switches back to blinking the LED.

The firmware on the C8051F582 target microcontroller does not need to be customized to use the UART and communicate with ToolStick Terminal. The firmware on the microcontroller should write to the UART as it would in any standard application and all of the translation is handled by the ToolStick Base Adapter.

7. Additional Demo Examples

In addition to the **F582DC_FeaturesDemo** example firmware, the ToolStick download package also includes a demo project named **F582DC_ADC0_TemperatureSensor**. The instructions for running this demo can be found at the top of the source file. The project and source files for these demos can be found in the *C:\SiLabs\MCU\ToolStick\F582DC\Firmware*\ folder.

8. Using the C8051F582 Daughter Card as a Development Platform

The prototyping area on the ToolStick C8051F582 daughter card makes it easy to interface to external hardware. All of the digital I/O pins are available so it is possible to create a complete system.

8.1. C8051F582 Pin Connections

It is important to note that if external hardware is being added, some of the existing components on the board can interfere with the signaling. The following is a list of port pins on the C8051F582 that are connected to other components:

- P0.0—This pin is connected to a 1.0 uF capacitor, which serves as a decoupling capacitor for the on-chip voltage reference. The capacitor C3 can be removed to disconnect it from the pin.
- P0.4, P0.5—These pins are connected directly to the ToolStick Base Adapter for UART communication.
- P0.6, P0.7—These pins are connected to the TXD and RXD pins of the CAN transceiver. The CAN transceiver can be removed to disconnect it from the pins.
- P1.0, P1.1—These pins are connected directly to the ToolStick Base Adapter's GPIO pins. By default, these GPIO pins on the Base Adapter are high-impedance pins so they will not affect any signaling. Configuring these pins on the Base Adapter to output pin or handshaking pins could affect signaling.
- P1.2—This pin is connected to the anode of the green LED on the daughter card. The LED or the R2 resistor can be removed to disconnect the LED from the pin.
- P1.3—This pin is connected to the output of the potentiometer. The 0 ohm resistor R5 can be removed to disconnect the potentiometer from the pin.
- P1.4—This pin is connected to the P1.4 switch. The switch can be removed to disconnect it from the pin.
- P1.5—This pin is connected to the mode select pin on the CAN transceiver. The CAN transceiver can be removed to disconnect it from the pins.

See the daughter card schematic in Section 10 for more information.

8.2. C2 Pin Sharing

On the C8051F582, the debug pins, C2CK, and C2D, are shared with the pins /RST and P3.0 respectively. The daughter card includes the resistors necessary to enable pin sharing, which allow the /RST and P3.0 pins to be used normally while simultaneously debugging the device. See Application Note "AN124: Pin Sharing Techniques for the C2 Interface" at www.silabs.com for more information regarding pin sharing.

9. Information Locations

Example source code is installed by default in the "C:*SiLabs**MCU**ToolStick**F582DC**Firmware*" directory during the ToolStick installation.

Documentation for the ToolStick kit, including this User's Guide, can be found in the *C:\SiLabs\MCU\ToolStick\Documentation* and the *C:\SiLabs\MCU\ToolStick\F582DC\Documentation* directories.

The installer for the ToolStick software is available at www.silabs.com/toolstick.

10. C8051F582 Daughter Card Schematic

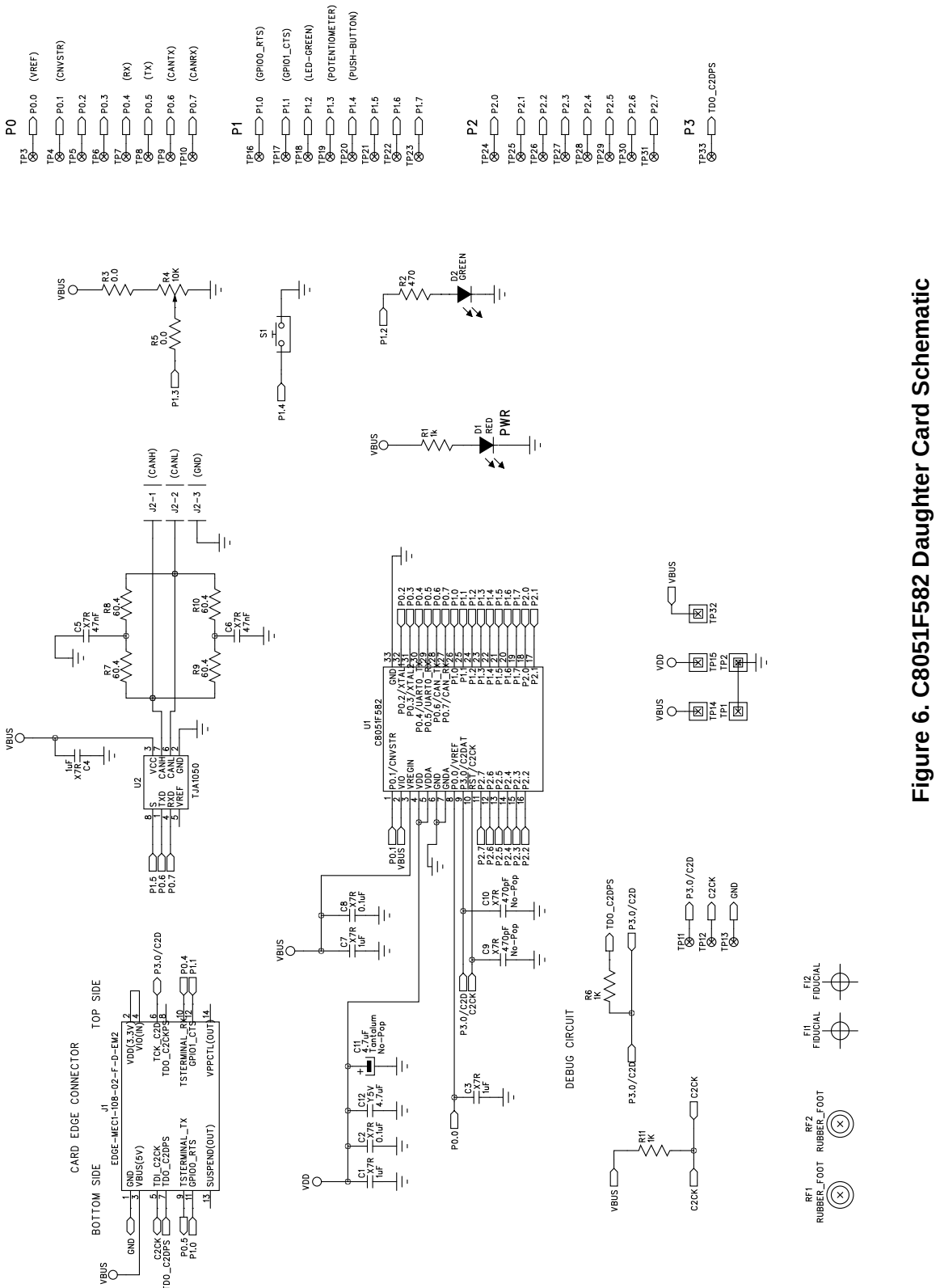
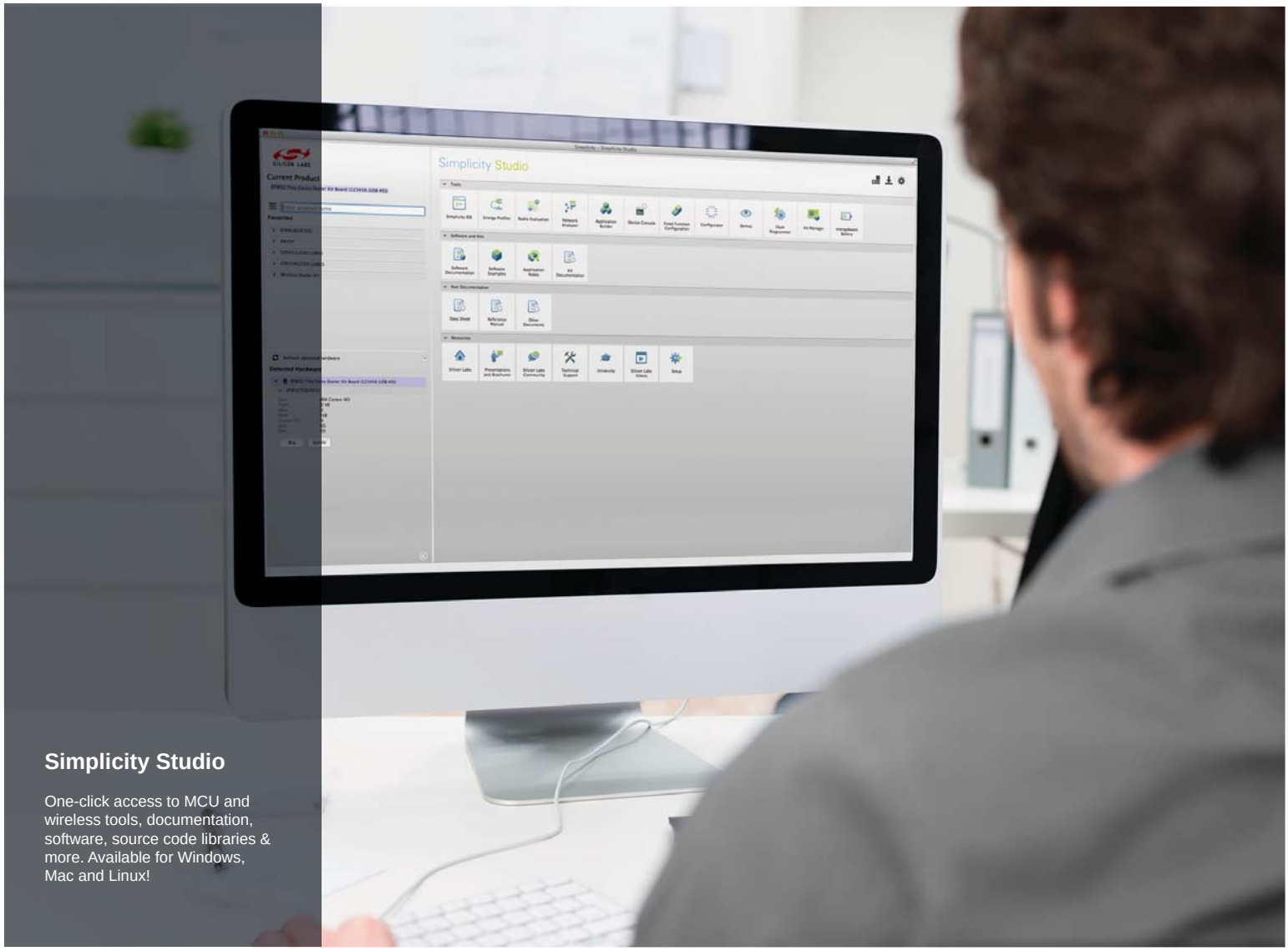


Figure 6. C8051F582 Daughter Card Schematic



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 Laboratories 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 Laboratories 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 Laboratories 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 Laboratories 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 must not be used within any Life Support System without the specific written consent of Silicon Laboratories. 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 Laboratories products are generally not intended for military applications. Silicon Laboratories 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, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem ®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. 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 LABS

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

http://www.silabs.com