ToolStickUniDC

TOOLSTICK UNIVERSITY 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, consider 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 board.

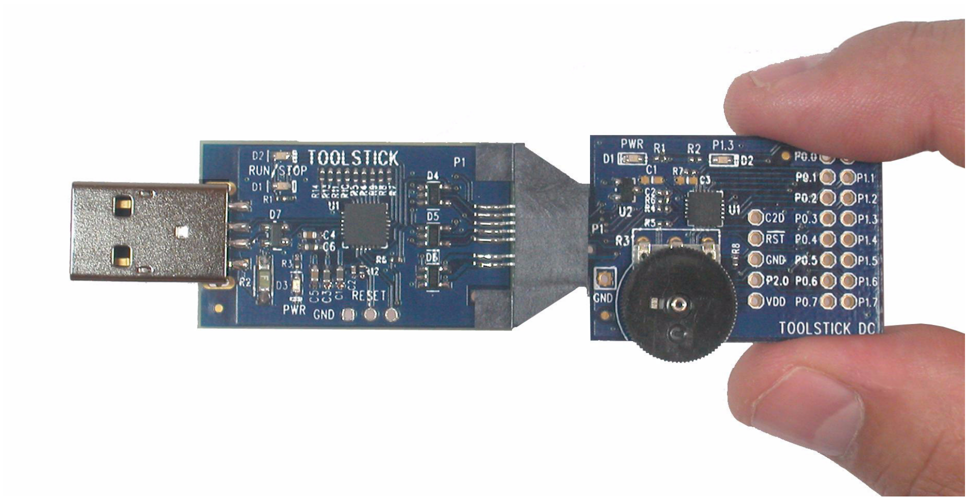


Figure 1. Proper Method of Holding the ToolStick

- Avoid directly touching any of the other components.

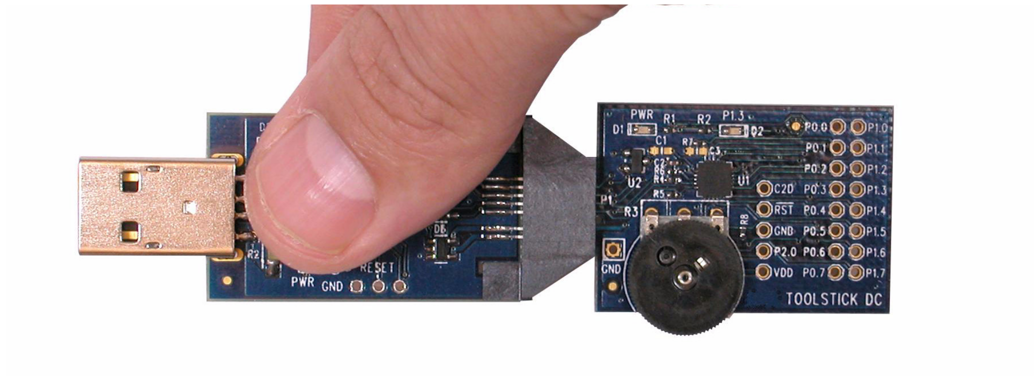


Figure 2. Improper Method of Holding the ToolStick

- 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.

2. Contents

The ToolStickUniDC kit contains the following items:

- ToolStick University Daughter Card

The ToolStickUniSK Starter Kit includes the following items:

- ToolStick Base Adapter
- ToolStick University Daughter Card
- 3-foot USB extension cable

A ToolStick daughter card requires a ToolStick Base Adapter to communicate with the PC. If the daughter card was not purchased as part of a Starter Kit, ToolStick Base Adapters can be purchased separately 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 target microcontroller and application circuitry are located on the daughter card. Some daughter cards, such as the ToolStick University Daughter Card, are used as general-purpose development platforms for the target microcontrollers, and some are used to demonstrate a specific feature or application.

The ToolStick University Daughter Card features a Silicon Laboratories C8051F020 Mixed-Signal MCU. The external circuitry on the daughter card includes a 22.1184 MHz crystal to enable UART communication, 4 LEDs, 4 push-button switches, an 8-bit DIP switch, a potentiometer, and a reset switch. Ports P0, P1, and P2 of the MCU are accessible via standard headers. A separate header is available for analog input/output that connects external signals to the ADC and comparator inputs, and DAC outputs of the MCU. A small area for prototyping is also provided that can be used to connect additional hardware to the microcontroller and use the daughter card as a development platform.

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

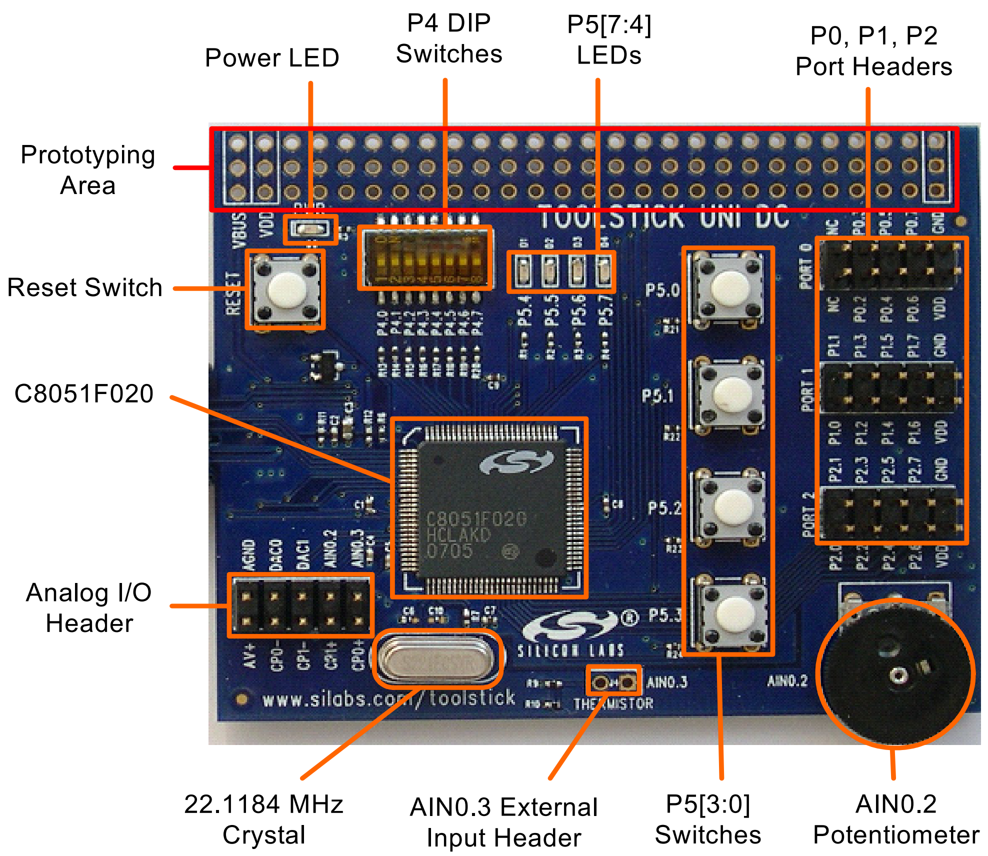


Figure 3. ToolStick University Daughter Card (Rev 002)

4. Getting Started

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

- Silicon Laboratories Integrated Development Environment (IDE)
- Keil Demonstration Toolset
- ToolStick Virtual Tools

The Keil Demo Toolset includes a compiler, assembler, and linker. See "5.2.2. Keil Demonstration C51 C Compiler," on page 4 for more details about the demo tools. ToolStick Virtual Tools include the ToolStick Terminal application that allows serial communication with the target microcontroller's UART through the ToolStick Base Adapter. Also included as part of the ToolStick Virtual Tools are the ToolStick Virtual LCD and the ToolStick Virtual Oscilloscope. The ToolStick Virtual Tools are described in detail in “AN333: ToolStick Virtual Tools User’s Guide”.

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

- Configuration Wizard 2
- Keil uVision Drivers

All of the above software is described in more detail in “5. Software Overview”.

To simplify the download process, the necessary software described above is provided in two download packages. The IDE Download package includes the Silicon Labs IDE and the Keil Demo Toolset. The ToolStick Virtual Tools Download package includes the ToolStick virtual tools, example code, and documentation including user’s guides and data sheets. After downloading (from www.silabs.com/mcuniversity) and installing these two 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 University Daughter Card Demo," on page 7 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 other third-party compilers and assemblers is also supported.

5.1.1. IDE System Requirements

The Silicon Laboratories IDE requirements:

- Pentium-class host PC running Microsoft Windows 2000 or newer.
- One available USB port.
- 64 MB RAM and 40 MB free HD space recommended.

5.1.2. 3rd 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 ToolStick University Daughter Card are written for the Keil and SDCC toolsets.

5.2. Keil Demonstration Toolset

5.2.1. Keil Assembler and Linker

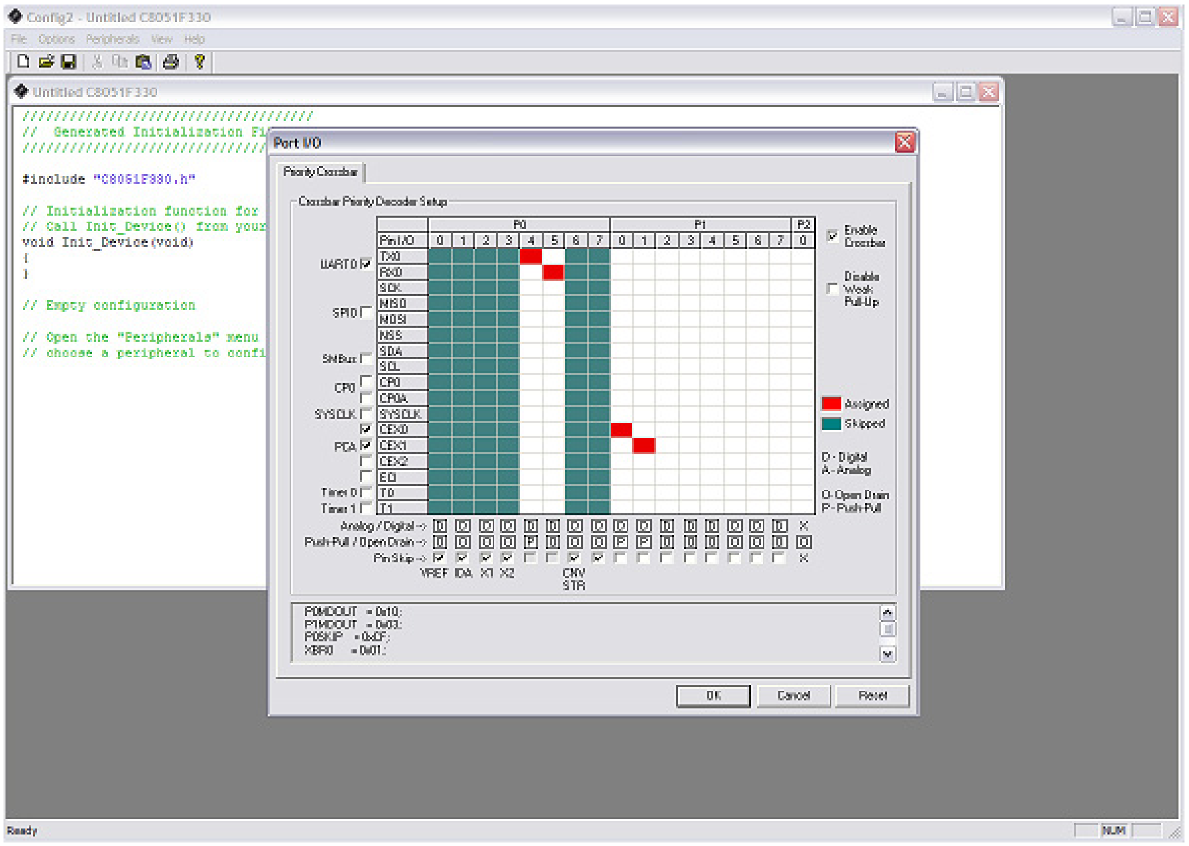
The assembler and linker that are part of the Keil Demonstration Toolset are the same versions that are found in the full Keil Toolset. The complete assembler and linker reference manual can be found on-line under the **Help** menu in the IDE or in the "*SiLabs\MCU\hlp*" directory (A51.chm).

5.2.2. Keil Demonstration C51 C Compiler

The demonstration version of the C51 compiler is the same as the full version with these main limitations: (1) Maximum 2 kB code generation (2) Floating point library not included (3) Programs start at code address 0x0800. A complete list of limitations can be found on the Keil website, www.keil.com/demo/limits.asp. The C51 compiler reference manual can be found under the **Help** menu in the IDE or in the "*SiLabs\MCU\hlp*" directory (C51.chm).

5.3. Configuration Wizard 2

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



```
////////////////////////////////////////
//  Generated Initialization Fi
////////////////////////////////

#include "C8051F330.h"

// Initialization function for
// Call Init_Device() from your
void Init_Device(void)
{
}

// Empty configuration

// Open the "Peripherals" menu
// choose a peripheral to confi
```

```
P0MDOUT   = 0x10;
P1MDOUT   = 0x03;
P0SKIP   = 0xCF;
XBR0    = 0x01;
```

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 for 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 code.

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

5.4. Keil uVision Silicon Laboratories Drivers

As an alternative to the Silicon Laboratories IDE, the uVision debug driver allows the Keil uVision IDE to communicate with Silicon Laboratories on-chip debug logic. In-system Flash memory programming integrated into the driver allows for rapid updating of target code. The uVision IDE 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 uVision driver documentation. The documentation and software are available from the downloads webpage.

5.5. ToolStick Virtual Tools

The ToolStick Virtual Tools include the following applications:

- ToolStick Terminal
- ToolStick Virtual LCD
- ToolStick Virtual Oscilloscope

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. The ToolStick Terminal integrated into the ToolStick Virtual Tools uses a baud rate of 230400 bps with a data format of 8-N-1 (8 data bits, No parity bit, 1 stop bit), and RTS/CTS hardware handshaking enabled.

The ToolStick Virtual LCD application creates a simple LCD panel on the PC that can be controlled by sending commands and data through the target microcontroller’s UART interface via the ToolStick Base Adapter.

The ToolStick Virtual Oscilloscope application allows the PC to be used as an oscilloscope by sending data points to be plotted through the target microcontroller’s UART interface via the ToolStick Base Adapter.

See “AN333: ToolStick Virtual Tools User’s Guide” included in the ToolStick Virtual Tools Download package for instructions on using the ToolStick Virtual Tools.

6. ToolStick University Daughter Card Demo

The ToolStick Virtual Tools download package includes a simple code example. The example described in this section is titled **UniDC_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 **UniDC_FeaturesDemo** example code simply lights up an LED when a push button is pushed. This demo shows how to use the IDE to connect and download the firmware, view and modify registers, use watch windows, use breakpoints, and single-step through code.

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 University 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.

Figure 5 shows an example hardware setup.

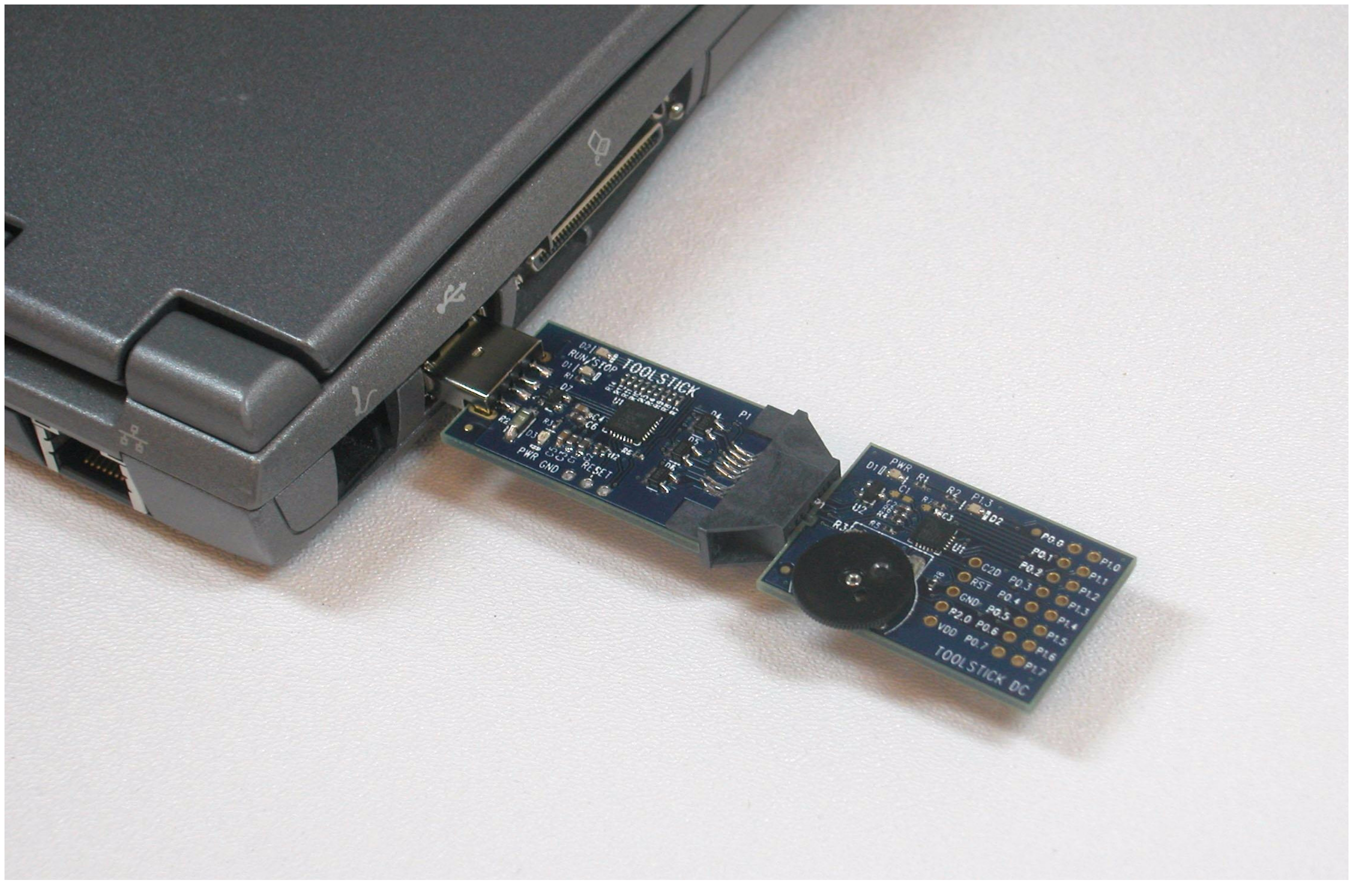
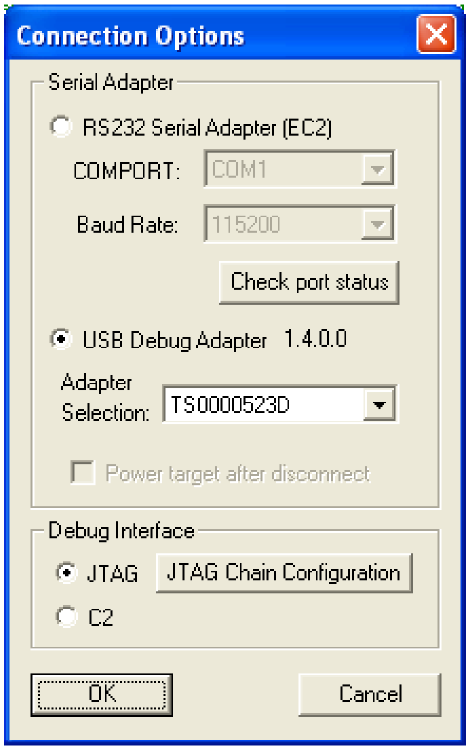


Figure 5. Hardware Setup Example

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 the default location *C:\SiLabs\MCU\ToolStick\UniversityDC\Firmware\FeaturesDemo\.*
4. Select **UniDC_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 **JTAG** for the Debug Interface, and then click **OK** as shown in Figure 6.


Connection Options

Serial Adapter

RS232 Serial Adapter (EC2) [✓ USB Debug Adapter 1.4.0.0]

COMPORT: COM1

Baud Rate: 115200

Adapter Selection: TS0000523D

Power target after disconnect

Debug Interface

[✓ JTAG] C2

Figure 6. Connection Options Screen

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 were 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. One green LED (P5.4) on the daughter card will light up. 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, Press the push button labeled P5.0 to see another LED (P5.7) light up.

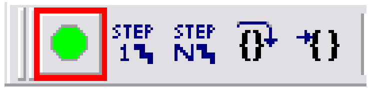


Figure 7. Go Button

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.

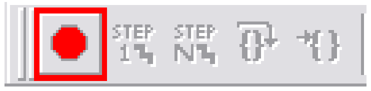


Figure 8. Stop Button

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 Ports Debug Window and light up an LED directly from the IDE.

1. Open the Ports Debug Window from the **View** → **Debug Windows** → **SFR's** → **Ports** menu option shown in Figure 9. The Ports Debug window appears on the right-hand side of the IDE. In this window, the P74OUT register is shown. This register sets the output mode of ports 4 through 7.

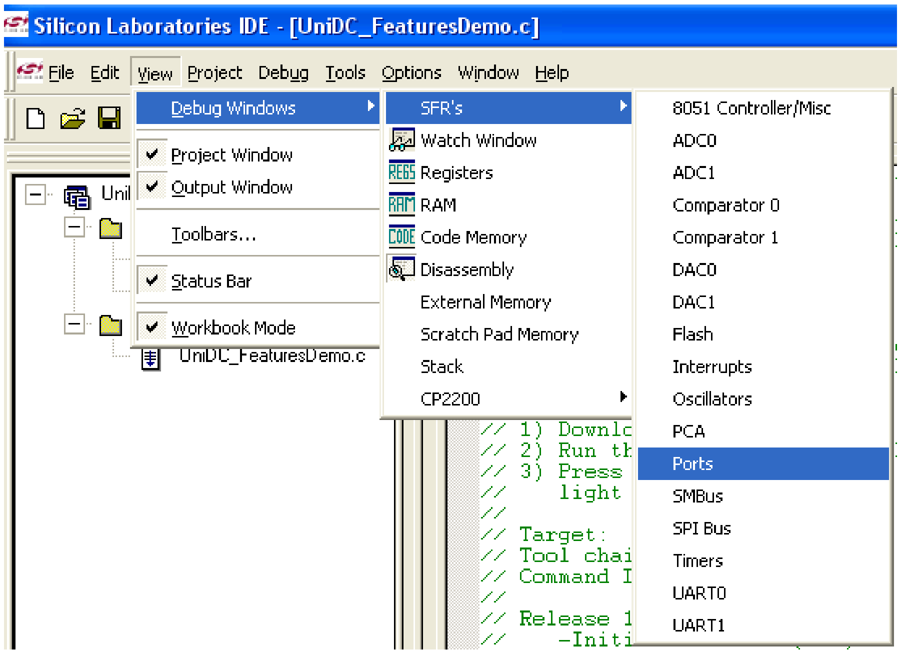
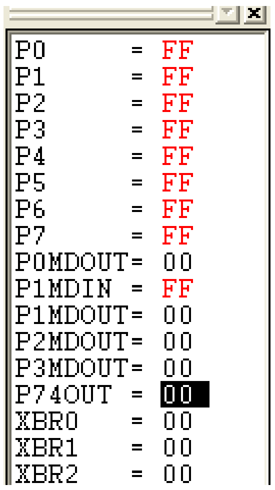


Figure 9. Ports Debug Window

2. In the debug window, change the value of **P74OUT** to 0x00. This value turns off all four LEDs on P5[7:4].



```
P0      = FF
P1      = FF
P2      = FF
P3      = FF
P4      = FF
P5      = FF
P6      = FF
P7      = FF
P0MDOUT= 00
P1MDIN  = FF
P1MDOUT= 00
P2MDOUT= 00
P3MDOUT= 00
P74OUT  = 00
XBR0    = 00
XBR1    = 00
XBR2    = 00
```

Figure 10. Debug Window

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

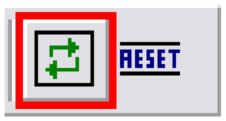


Figure 11. Refresh Button

4. Click “**Go**” to resume running the device with the new **P74OUT** value.
5. Press the push button, **P5.0**, and notice that it has no effect on the LED P5.7.
6. Re-enable the P5 output mode by writing 0x08 to the **P74OUT** and clicking the **Refresh** button.
7. Then, also try changing the upper four bits of the **P5** register and click the **Refresh** button to directly light up the LEDs.

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 C8051F020 data sheet for the definitions and usage for all registers.

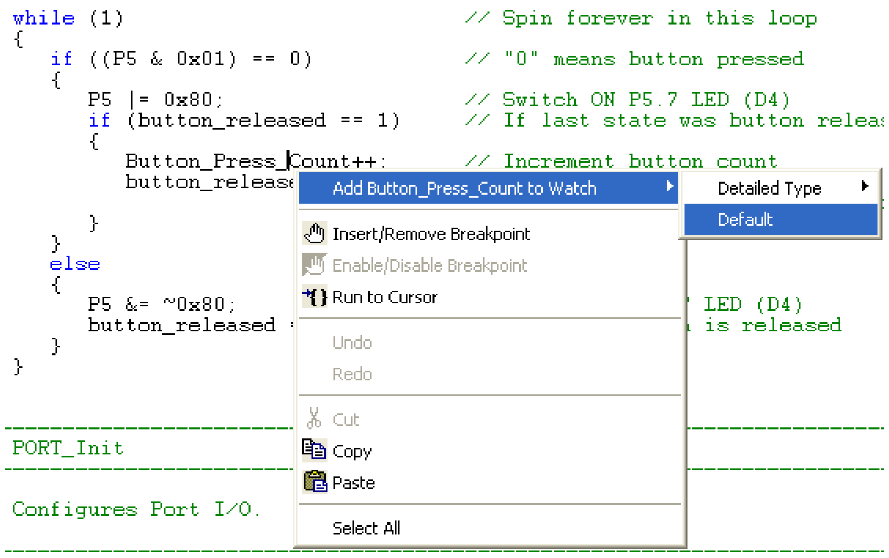
The debug windows for other sets of registers are found in the **View** → **Debug Windows** → **SFR’s** menu.

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 **UniDC_FeaturesDemo** example code, the variable **Button_Press_Count** is a counter that stores the number of times the P5.0 button was pressed.

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 **UniDC_FeaturesDemo.c** to open the source file.

3. Scroll to the **main** function (line 69) and right-click on the variable “**Button_Press_Count**”. In the context menu that appears, select “**Add Button_Press_Count to Watch**” and then choose “**Default**.” 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.



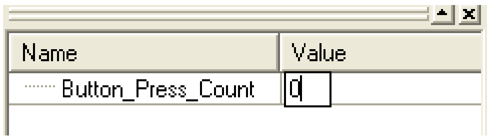
```
while (1)                          // Spin forever in this loop
{
   if ((P5 & 0x01) == 0)           // "0" means button pressed
   {
      P5 |= 0x80;                  // Switch ON P5.7 LED (D4)
      if (button_released == 1)    // If last state was button releas
      {
         Button_Press_Count++;     // Increment button count
         button_release
      }
   }
   else
   {
      P5 &= ~0x80;                             LED (D4)
      button_released                          is released
   }
}

------------------------------------------------------------------------
PORT_Init
------------------------------------------------------------------------

Configures Port I/O.
------------------------------------------------------------------------
```

Figure 12. Adding Variable to Watch Window

4. **Start** code execution and press button P5.0 a few times. **Stop** the device and see that the value of the **Button_Press_Count** is incremented.
5. When the device is halted, click on the value field in the watch window shown in Figure 13, and change the value to 0. Then, click the **Refresh** button or select **Debug** → **Refresh** to write the new value to the device.



Name	Value
Button_Press_Count	0

Figure 13. Modifying Variable Value in Watch Window

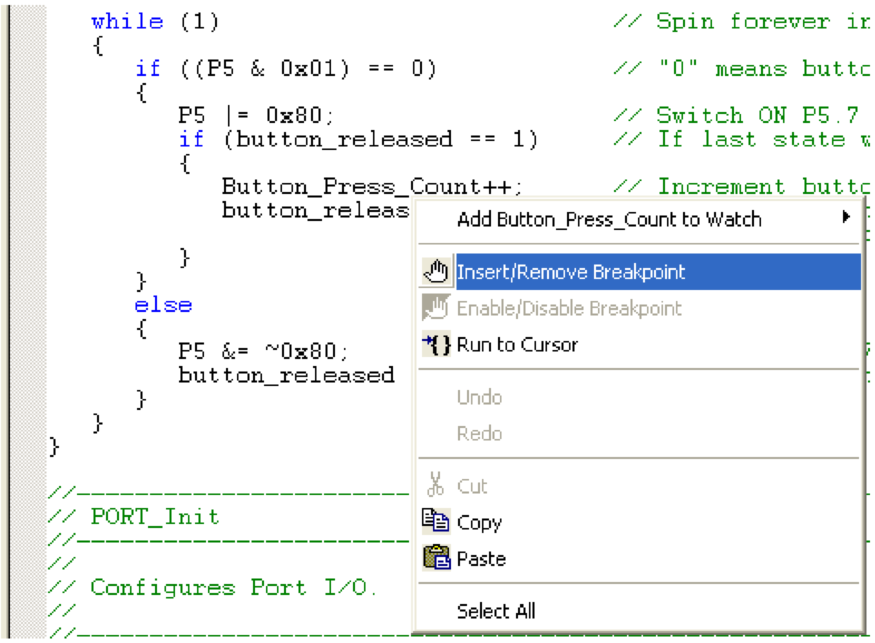
6. Repeat step 4 to watch the variable increment starting at 0.

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 variable values changed. The firmware will continue to execute using the new values.

6.6. Setting and Running to Breakpoints

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, 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 **Button_Press_Count** variable.

1. If the device is running, stop execution using the “**Stop**” button or use the **Debug** → **Stop** menu option.
2. Scroll to the **main** function (line 69) and right-click on the **Button_Press_Count** variable. In the context menu shown in Figure 14, select **Insert/Remove Breakpoint**. On the left side of the line in the editor window shown in Figure 15, a red circle is added to indicate that a breakpoint has been placed on the source line.

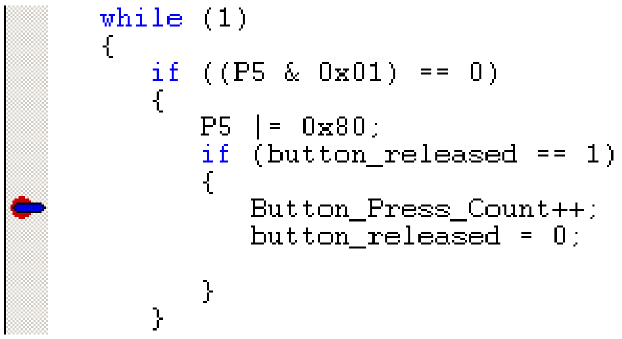


```
   while (1)                              // Spin forever ir
   {
      if ((P5 & 0x01) == 0)               // "0" means buttc
      {
         P5 |= 0x80;                      // Switch ON P5.7
         if (button_released == 1)        // If last state w
         {
            Button_Press_Count++;         // Increment buttc
            button_releas
         }
      }
      else
      {
         P5 &= ~0x80;
         button_released
      }
   }
}

//-----------------------
// PORT_Init
//-----------------------
//
// Configures Port I/O.
//
//-----------------------
```

Figure 14. Inserting Breakpoint

3. Click the **Go** button or select the **Debug** → **Go** menu option.
4. Press the button labeled P5.0. 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.



```
   while (1)
   {
      if ((P5 & 0x01) == 0)
      {
         P5 |= 0x80;
         if (button_released == 1)
         {
            Button_Press_Count++;
            button_released = 0;

         }
      }
```

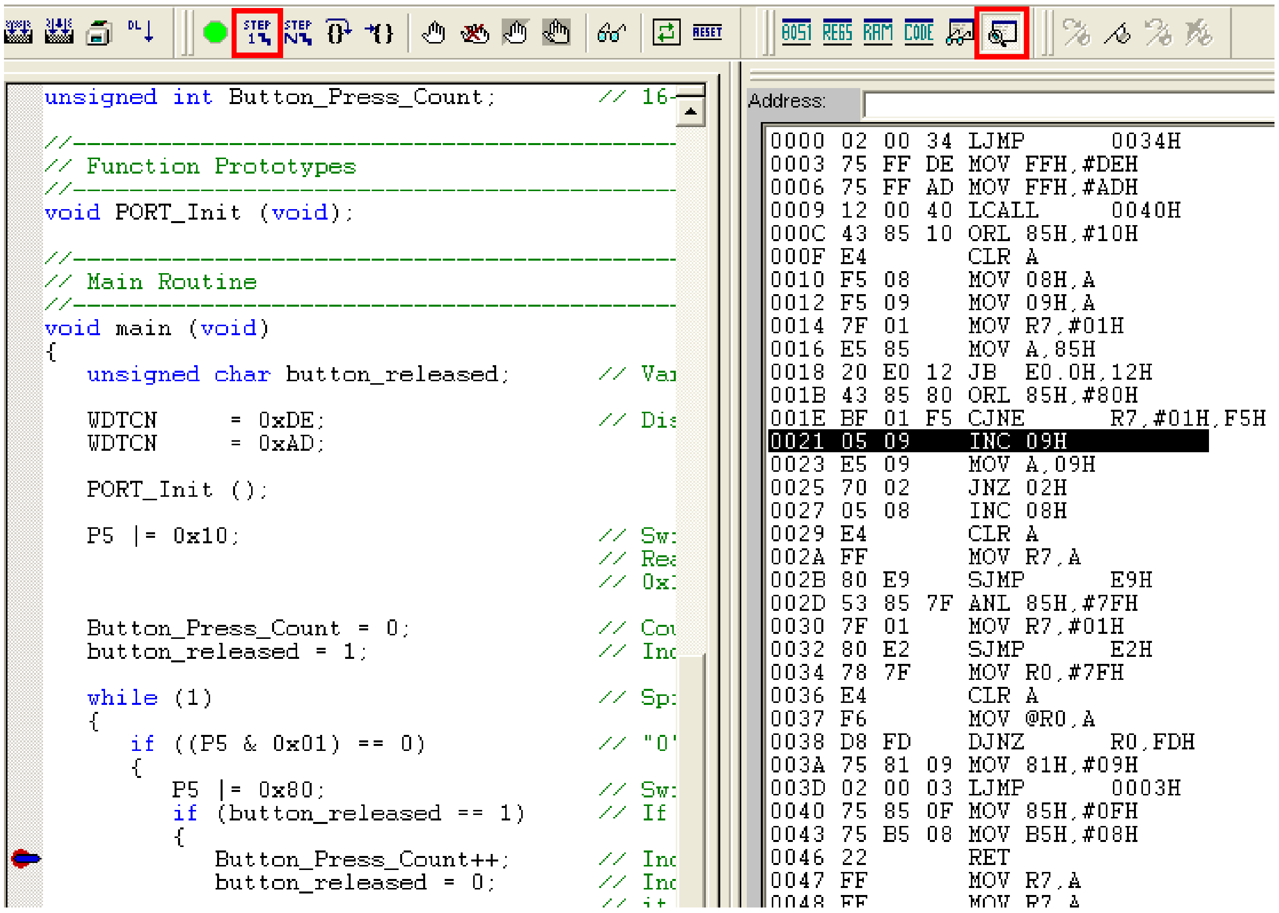
Figure 15. Execution Stopped at Breakpoint

5. **Start** the processor a few more times, and press the button when the processor is executing. Notice that the IDE halts execution whenever the button is pressed. For every time the processor is started and the button is pressed, the **Button_Press_Count** 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 not a breakpoint already set on the line of code that increments the **Button_Press_Count** variable, set the breakpoint using the steps described in Section 6.6.
2. Start the processor using the **Go** button, and press the button labeled P5.0 on the daughter card to stop execution on the breakpoint.
3. Select **View** → **Debug Windows** → **Disassembly**, or click the **Disassembly Window** button on the toolbar. The disassembly window shown in Figure 16 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.



```
unsigned int Button_Press_Count;              // 16-

//-----------------------------------------------
// Function Prototypes
//-----------------------------------------------
void PORT_Init (void);

//-----------------------------------------------
// Main Routine
//-----------------------------------------------
void main (void)
{
   unsigned char button_released;             // Va

   WDTCN      = 0xDE;                         // Dis
   WDTCN      = 0xAD;

   PORT_Init ();

   P5 |= 0x10;                                // Sw
                                              // Rea
                                              // 0x

   Button_Press_Count = 0;                    // Cou
   button_released = 1;                       // Inc

   while (1)                                  // Sp
   {
      if ((P5 & 0x01) == 0)                   // "0'
      {
         P5 |= 0x80;                          // Sw
         if (button_released == 1)            // If
         {
            Button_Press_Count++;             // Inc
            button_released = 0;              // Inc
```

```
0000 02 00 34 LJMP      0034H
0003 75 FF DE MOV FFH,#DEH
0006 75 FF AD MOV FFH,#ADH
0009 12 00 40 LCALL     0040H
000C 43 85 10 ORL 85H,#10H
000F E4       CLR A
0010 F5 08    MOV 08H,A
0012 F5 09    MOV 09H,A
0014 7F 01    MOV R7,#01H
0016 E5 85    MOV A,85H
0018 20 E0 12 JB  E0.0H,12H
001B 43 85 80 ORL 85H,#80H
001E BF 01 F5 CJNE      R7,#01H,F5H
0021 05 09    INC 09H
0023 E5 09    MOV A,09H
0025 70 02    JNZ 02H
0027 05 08    INC 08H
0029 E4       CLR A
002A FF       MOV R7,A
002B 80 E9    SJMP      E9H
002D 53 85 7F ANL 85H,#7FH
0030 7F 01    MOV R7,#01H
0032 80 E2    SJMP      E2H
0034 78 7F    MOV R0,#7FH
0036 E4       CLR A
0037 F6       MOV @R0,A
0038 D8 FD    DJNZ      R0,FDH
003A 75 81 09 MOV 81H,#09H
003D 02 00 03 LJMP      0003H
0040 75 85 0F MOV 85H,#0FH
0043 75 B5 08 MOV B5H,#08H
0046 22       RET
0047 FF       MOV R7,A
0048 FF       MOV R7,A
```

Figure 16. Disassembly Window

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 allow a developer to see exactly what instructions are executed and their output.

7. Additional Demo Examples

In addition to the **UniDC_FeaturesDemo** example firmware, the ToolStick download package also includes a demo project called **UniDC_VirtualTools_Demo**. The instructions for running this demo can be found at the top of the source file. The ToolStick Virtual Tools are described in “AN333: ToolStick Virtual Tools User’s Guide”.

The project and source files for the demos can be found in the *C:\SiLabs\MCU\ToolStick\UniversityDC\Firmware\* default directory.

8. Using the ToolStick University Daughter Card as a Development Platform

The prototyping area on the ToolStick University Daughter Card makes it easy to interface to external hardware. Ports P0, P1, and P2 are available at the headers. Also available are the analog input/output signals at header J5. Wires can be soldered between components in the prototyping area and the headers to add interface circuitry.

8.1. C8051F020 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 C8051F020 MCU that are connected to other components:

- P0.0, P0.1—These pins are connected directly to the ToolStick Base Adapter for UART communication. These two pins are not available on the header labeled PORT 0 (J1).
- P3.6, P3.7—These pins are connected directly to the ToolStick Base Adapter’s GPIO pins for use as UART handshaking signals. The ToolStick Terminal included in the Virtual Tools application configures the GPIO pins on the Base Adapter as RTS/CTS handshaking pins. This means that the pin, P3.7 (CTS), should be configured as push-pull output, and P3.6 (RTS) should be configured as open-drain input on the ToolStick University daughter card.
- P5[3:0]—These four pins are connected to push-button switches on the daughter card. The series resistors, R21 through R24, can be removed to disconnect the push-button switch from the pin.
- P5[7:4]—These four pins are connected to the cathode terminals of the green LEDs on the daughter card. The LEDs, D1 through D4, or the series resistors, R1 through R4, can be removed to disconnect the LED from the pin.
- P4—All eight pins of port P4 are connected to the eight-bit DIP switch on the daughter card. The series resistors, R13 through R20, can be removed to disconnect the DIP switch from the port pin.
- AIN0.2—This pin is connected to the output of the potentiometer. The potentiometer can be removed to disconnect it from the pin.

See the daughter card schematic in Figure 17 for more information.

9. Board Revision Information

Revision SA-TS003PCB-**001** of the ToolStick University Daughter card has the labels for the CP0- and CP1- pins swapped in the header, J5. This silk-screen error has been fixed on revision SA-TS003PCB-**002**. Figure 3 on page 3 shows the revised board with the labels in the correct positions. The revision number is located on the back side of the daughter card.

10. Information Locations

Example source code is installed by default in the “*C:\SiLabs\MCU\ToolStick\UniversityDC\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\UniversityDC\Documentation* default directories.

The installer for the ToolStick Virtual Tools software is available at www.silabs.com/mcuniversity.

11. ToolStick University Daughter Card Schematic

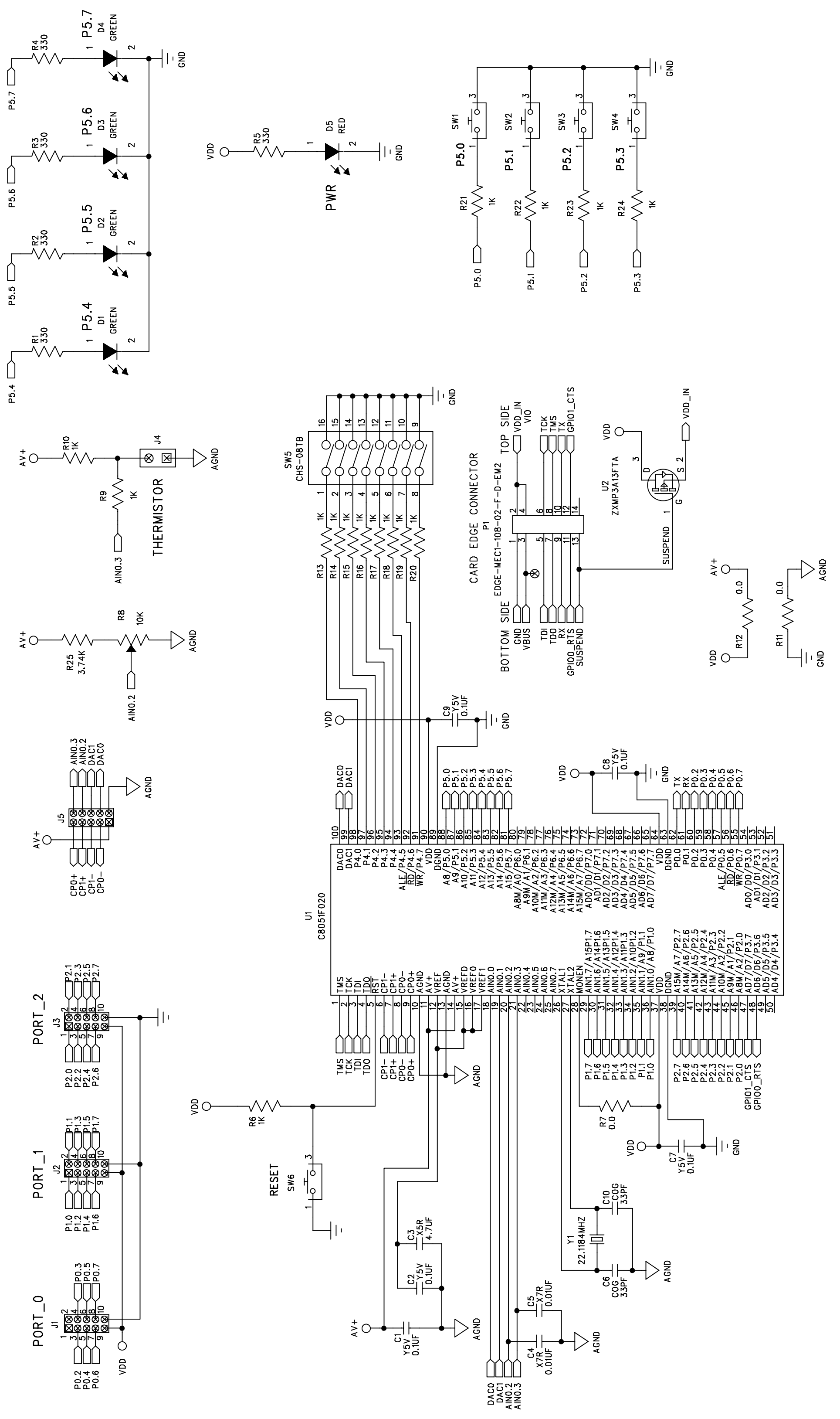


Figure 17. Daughter Card Schematic

Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Email: MCUinfo@silabs.com
Internet: www.silabs.com