ToolStick-LINDC

TOOLSTICK LIN 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 large plastic connector.

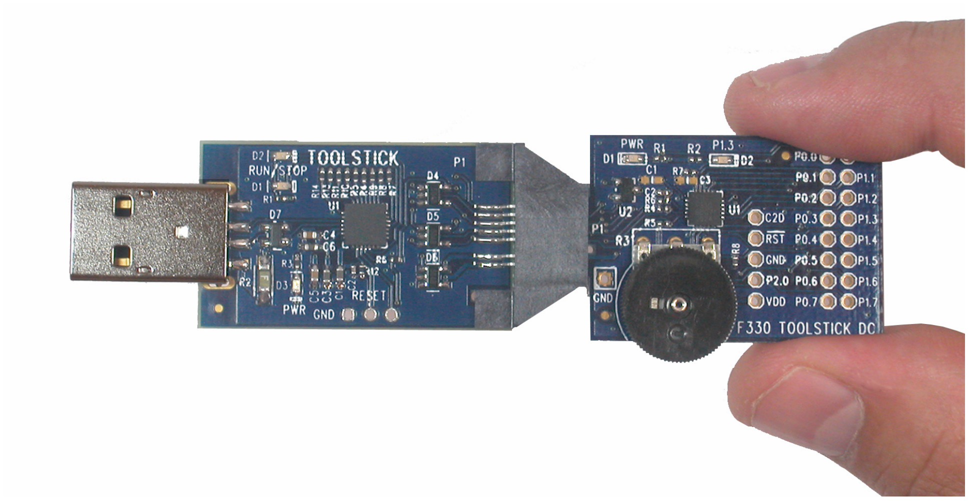


Figure 1. Proper Method of Holding the ToolStick

- Avoid directly touching any of the other components.

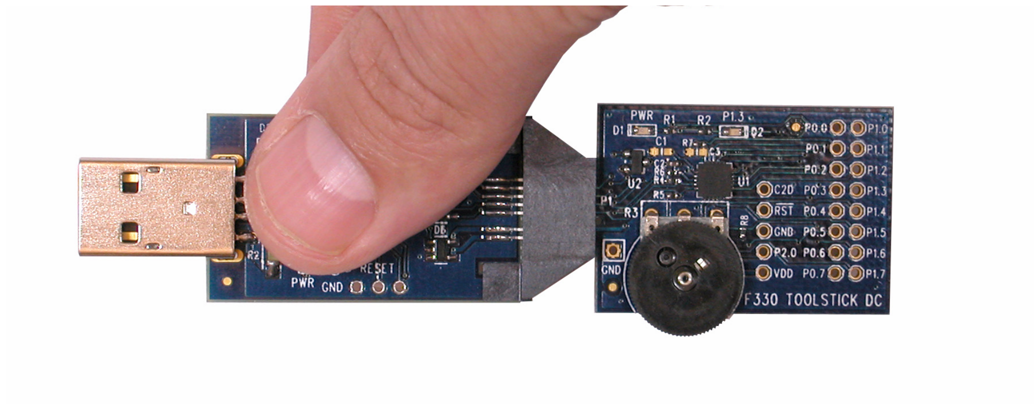


Figure 2. Improper Method of Holding the ToolStick

- 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 ToolStick-LINDC kit contains the following items:

- ToolStick LIN 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 target microcontroller and application circuitry are located on the daughter card. Some daughter cards, such as the LIN 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 LIN Daughter card includes a power LED, a LIN transceiver, a connector block for the LIN signals, and a prototyping area which provides access to all of the pins of the device. This prototyping area can be used to connect additional hardware to the microcontroller and use the daughter card as a development platform.

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

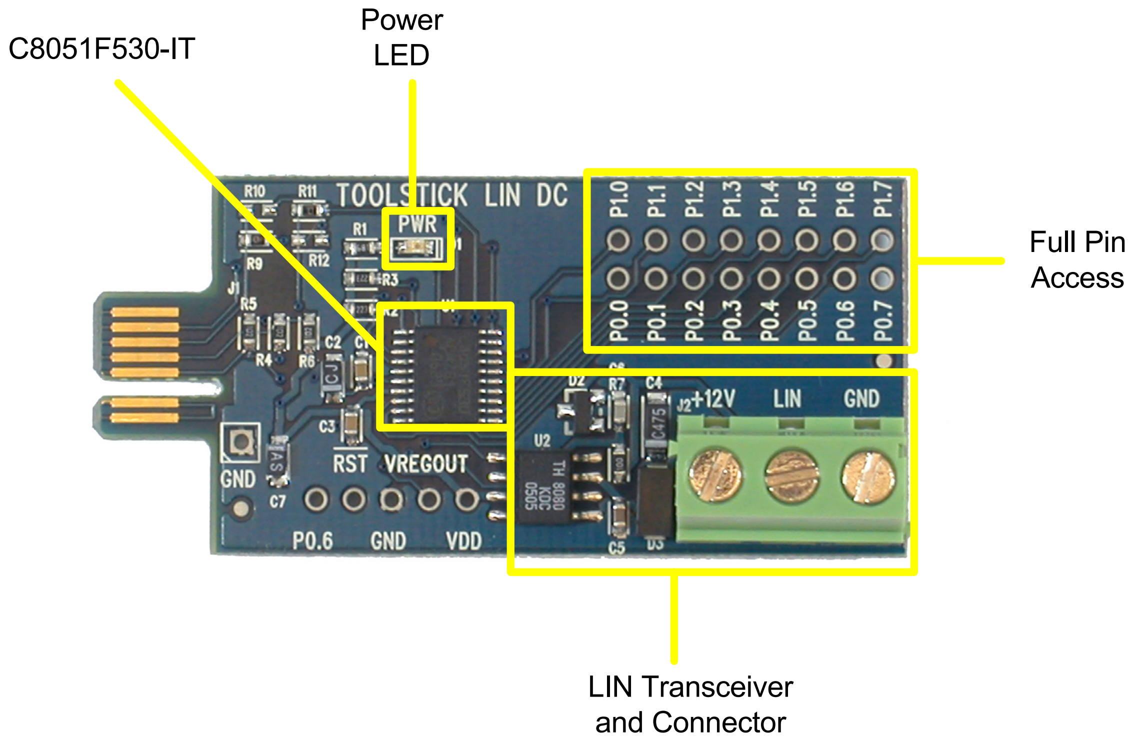


Figure 3. ToolStick LIN Daughter Card

4. Getting Started

The 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 Silicon Laboratories IDE is described in more detail in Section 5. The Keil Demo Tools include a compiler, assembler, and linker. The limits of the demo version are: 1) the resulting object code is limited to 2 kBytes and 2) the floating point library is not included. ToolStick Terminal communicates with the UART portion of the ToolStick Base Adapter to the microcontroller. It can also read/write the two GPIO pins available on the ToolStick Base Adapter.

Other useful, but optional, software that is provided on the ToolStick website includes:

- Configuration Wizard 2
- Keil uVision2 and uVision3 Drivers

Configuration Wizard 2 presents device peripheral options in a GUI and simplifies the generation of peripheral initialization code. The program is configurable to provide the output in C or assembly. The Keil uVision2 and uVision3 Drivers enable Keil uVision IDEs to debug Silicon Laboratories MCUs.

The software described above is provided in multiple download packages. The “ToolStick Development Tools” package includes the Keil Tools, example code, documentation including User’s Guides and data sheets, and the ToolStick Terminal application. The IDE, Configuration Wizard 2, and the Keil uVision Drivers are available as separate downloads. After downloading and installing the required software from www.silabs.com/toolstick, see the following sections for information regarding the Silicon Laboratories IDE and running one of the demo applications.

5. Silicon Laboratories IDE and Keil Demonstration Toolset

The Silicon Laboratories IDE integrates a source-code editor, source-level debugger, and an in-system Flash programmer. 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. IDE System Requirements

The Silicon Laboratories IDE requirements:

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

5.2. 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.pdf).

5.3. Keil Demonstration C51 C Compiler

The demonstration version of the C51 compiler is the same as the full version except code size is limited to 2 kB and the floating point library is not included. The C51 compiler reference manual can be found under the **Help** menu in the IDE or in the "*SiLabs\MCU\hlp*" directory (C51.pdf).

5.4. 3rd Party Toolsets

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

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

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

6. ToolStick LIN Daughter Card Firmware Examples

The ToolStick kit includes a few simple code examples for the LIN Daughter Card, including a simple timer example and a LIN snooper example. The timer example is stand-alone firmware that uses an on-chip Timer to measure how long the firmware is running and reports this time to the PC using ToolStick Terminal. The second example is a LIN snooper application. With the LIN snooper firmware, the LIN Daughter Card will capture the messages from an arbitrary LIN network and output the raw data to the PC using ToolStick Terminal.

The purpose of the rest of this chapter purpose is to guide a new user through the features and capabilities of the IDE and demonstrate the microcontroller's on-chip debug capabilities using the timer example. The first part of 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. The second part of the demo shows how to use ToolStick Terminal to receive UART data from the daughter card.

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 LIN 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 4 below for an example hardware setup.

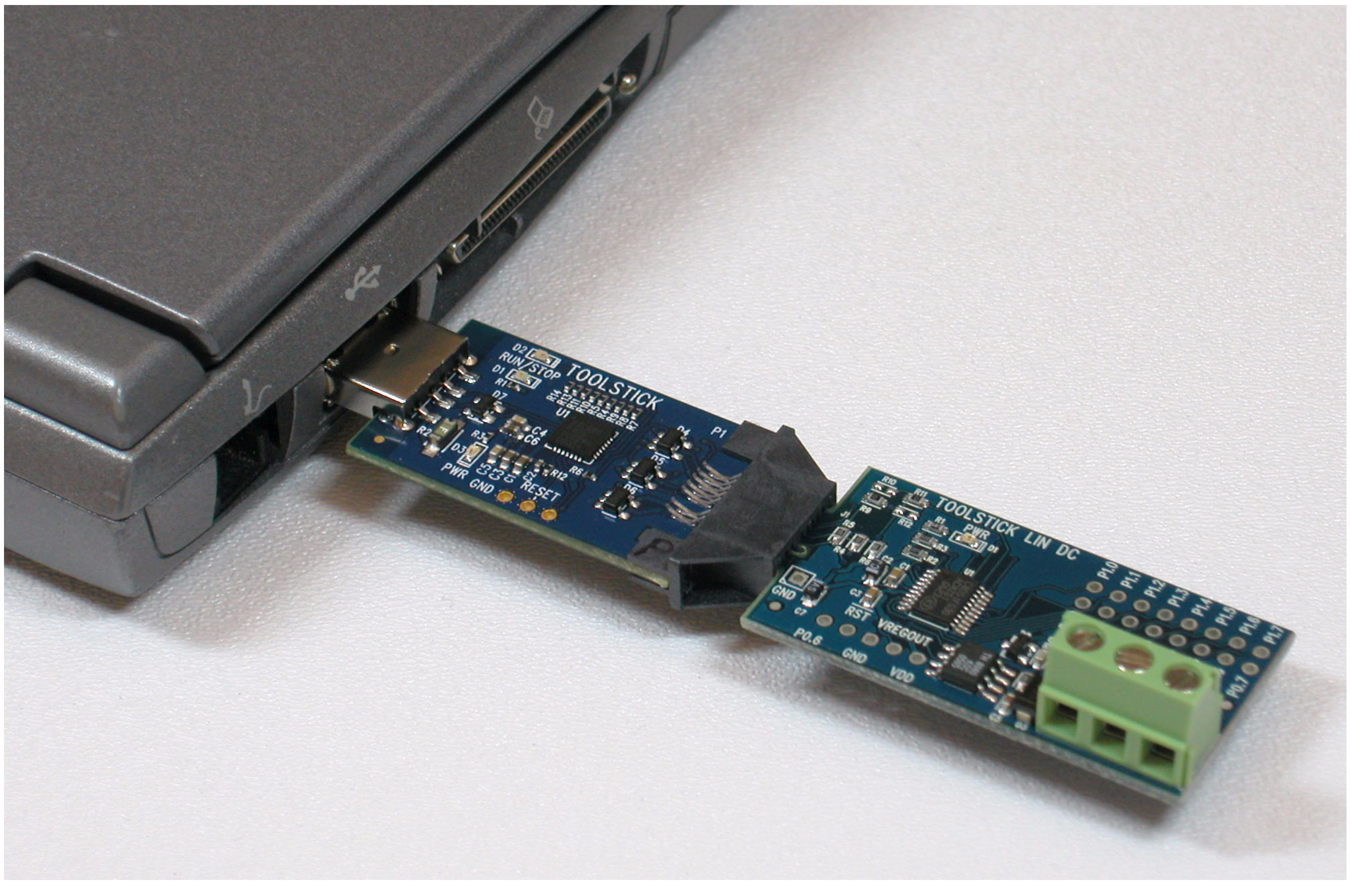


Figure 4. Hardware Setup Example

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\LINDC\Firmware\.*
4. Select **F530DC_SoftwareTimer.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 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 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 output of the firmware is sent to the UART and is captured by ToolStick Terminal. Since ToolStick Terminal is not connected at the moment, the output must be viewed using the IDE. Viewing the data in the IDE is discussed in Section 6.4. The debug commands in the IDE (single-step, multiple-step, set breakpoint, and others) are disabled when the device is running.
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. Enabling and Using Watch Windows

To view and modify variables in code, the IDE provides Watch Windows. 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 example code, the global variable **Seconds** is one variable of three variables that stores the time since the firmware has begun executing.

1. If the device is running, stop execution using the **Stop** button or use the **Debug** → **Stop** menu option.
2. Scroll to the Global Variables section of the firmware and right-click on the variable **Seconds**. In the context menu that appears, select **Add Seconds** 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.
3. **Start** and **Stop** the device a few times. See that the value of the Seconds is incremented each second that the firmware is running.
4. 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.
5. **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.

6.5. Viewing and Modifying Registers

Registers on the device can be also viewed and modified when the device is in a halted state using windows similar to watch windows. 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 Timers Debug Window and disable Timer2 directly from the IDE.

1. Open the Timers Debug Window from the **View** → **Debug Windows** → **SFR's** → **Timers** menu option. The Timers Debug Window appears on the right-hand side of the IDE. In this window, the TMR2CN register is shown. This register is used to enable and configure Timer2. Timer2 is used to to keep the time for this example. When the firmware is running, the **TMR2CN** register reads as 0x44 indicating that the Timer is running.
2. In the debug window, change the value of **TMR2CN** from 0x44 to 0x40. This value turns off Timer2 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 **TMR2CN** value.
5. Notice that starting and stopping the device does not increment the **Seconds** variable.
6. Re-enable the timer by writing 0x44 to the **TMR2CN** 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 C8051F530 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.

6.6. 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 explains how to set a breakpoint on the line of source code that increments the **Seconds** variable.

1. If the device is running, stop execution using the **Stop** button or use the **Debug** → **Stop** menu option.
2. Scroll to the Timer2_ISR function and right-click on the variable “**Seconds**”. See the comment in the code for exact line. 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 **Seconds** variable increments once for every time the processor is started.

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 **Seconds** 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 the UART on the target microcontroller 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 change the Baud Rate to **9600** and click “OK.” The rest of the default settings are correct for the C8051F530 Software Timer firmware.
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. Once connected, the **Receive Data** window displays the time the firmware has been running.
6. When ToolStick Terminal connects to the device, it resets it. The firmware starts keeping time from the reset and outputs the time 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. One of these GPIO pins is connected to port pin P1.1 on the C8051F530. The following steps describe how to change the level of one of the GPIO pins and signal the target microcontroller. When the port pin level is low, the firmware halts the software clock.

1. In ToolStick Terminal, under Pin State Configuration, select “**Set GPIO0 Logic Low**” and click on “**Set Selected Pin States**.” The firmware reads this pin in the Timer2_ISR() and stops incrementing the time if the pin is logic low.
2. See the ToolStick Terminal reports that the clock is halted.
3. Change the GPIO0 pin state back to **Logic High** and notice that the firmware resumes incrementing the time.

The firmware on the C8051F530 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 **F530DC_SoftwareTimer** example firmware, the ToolStick download package also includes demo projects named **F530DC_LIN_Snooper** and **F530DC_ADC0_TemperatureSensor**. The instructions for running these demos 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\LINDC\Firmware\* folder.

8. Using the C8051F530 Daughter Card as a Development Platform

The prototyping area on the ToolStick C8051F530 daughter card makes it easy to interface to external hardware. All of the digital I/O pins are available so it possible to create a complete system.

8.1. C8051F530 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 C8051F530 that are connected to other components:

- P0.0, P0.1—These pins are connected to the LIN transceiver (U2).
- P0.3, P0.4, P0.5—These pins are connected directly to the ToolStick Base Adapter for UART communication.
- P1.1, P1.2—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.

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

8.2. VREF Capacitor

On the C8051F530 devices, if VREF is generated internally, it is output to port pin P0.0. For VREF stability, it is highly recommended to place a capacitor on the VREF output pin. The firmware examples for the daughter card use VDD as VREF, so no external capacitor on P0.0 is necessary for proper operation.

8.3. C2 Pin Sharing

On the C8051F530, the debug pins, C2CK, and C2D, are shared with the pins /RST and P2.0 respectively. The daughter card includes the resistors necessary to enable pin sharing which allow the /RST and P0.6 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.

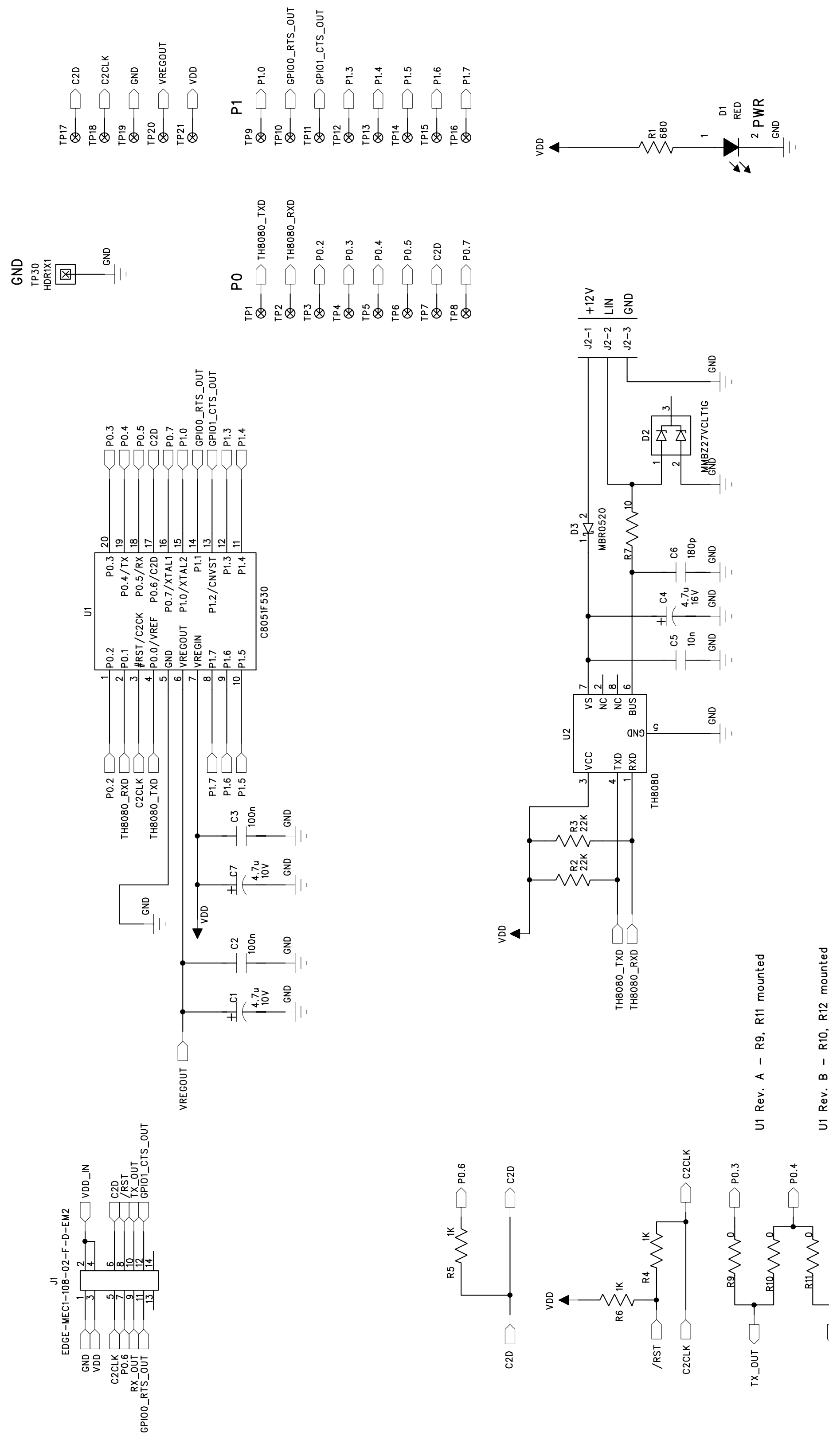
9. Information Locations

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

Documentation for the ToolStick kit can be found in the C:\SiLabs\MCU\ToolStick\Documentation\ directory.

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

10. C8051F530 Daughter Card Schematic



Document Change List

Revision 0.1 to Revision 0.2

- Updated paths in Section "9. Information Locations," on page 10.

NOTES:

Contact Information

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 532-5353
Email: mcuapps@silabs.com
Internet: www.silabs.com