TOOLSTICK USER'S GUIDE

1. Kit Contents

The ToolStick kit contains the following items:

- ToolStick
- Silicon Laboratories Evaluation Kit IDE and Product Information CD-ROM. CD content includes:
  - Silicon Laboratories Integrated Development Environment (IDE)
  - Keil Software 8051 Development Tools (evaluation assembler, linker and C compiler)
  - Source code examples and register definition files
  - Documentation
  - Demo software
- ToolStick User's Guide (this document)

2. ToolStick Overview

The purpose of the ToolStick is to provide a way to easily evaluate the Silicon Laboratories Integrated Development Environment (IDE) and the on-chip debug capabilities of the microcontrollers.

The ToolStick is a fully contained evaluation board. The target microcontroller on the board is a Silicon Laboratories C8051F300. The target device is connected to a C8051F320 microcontroller which provides a USB debug interface between the PC and the target device. The ToolStick enumerates as a Human Interface Device and so drivers are not necessary to communicate with the device. For more information regarding the C8051F300, see the C8051F30x data sheet.

Two of the digital I/O pins on the target C8051F300 microcontroller are connected to two pairs of LEDs. These LEDs provide feedback for the user. The ToolStick kit includes two example programs that make use of the LEDs. These examples are detailed in Sections 5 and 6 of this User's Guide.

Figure 1 shows the ToolStick without the plastic case and identifies the various components.

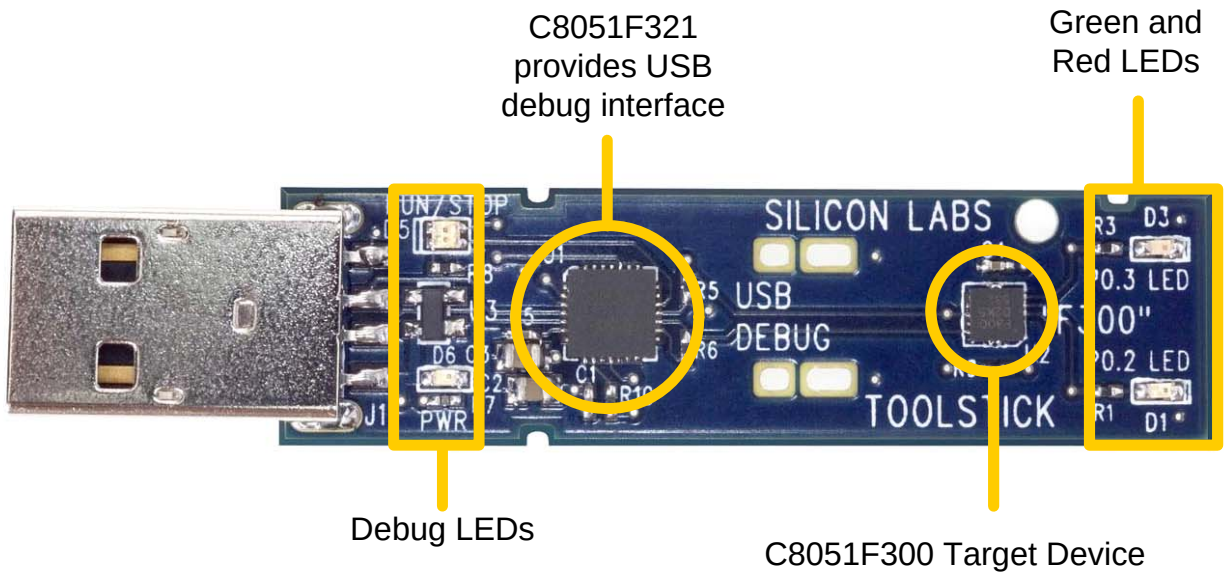


Figure 1. ToolStick Board

Copyright © 2005 by Silicon Laboratories

3. Software Setup

The included CD-ROM contains the Silicon Laboratories Integrated Development Environment (IDE), Keil software 8051 tools and additional documentation. Insert the CD-ROM into your PC's CD-ROM drive. An installer will automatically launch, allowing you to install the IDE software or read documentation by clicking buttons on the Installation Panel. If the installer does not automatically start when you insert the CD-ROM, run *autorun.exe* found in the root directory of the CD-ROM.

4. Silicon Laboratories Integrated Development Environment (IDE)

The Silicon Laboratories IDE integrates a source-code editor, source-level debugger, and in-system Flash programmer. The use of third-party compilers and assemblers is also supported. This kit includes the Keil Software A51 macro assembler, BL51 linker and evaluation version C51 C compiler. These tools can be used from within the Silicon Laboratories IDE.

4.1. System Requirements

The Silicon Laboratories IDE requirements:

- Pentium-class host PC running Microsoft Windows 98SE, Windows 2000, or Windows XP.
- One available USB port.
- 64 MB RAM and 40 MB free HD space recommended.

4.2. Assembler and Linker

A full-version Keil A51 macro assembler and BL51 banking linker are included with the kit and are installed during IDE installation. 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).

4.3. Evaluation C51 C Compiler

An evaluation version of the Keil C51 C compiler is included with the kit and is installed during IDE installation. The evaluation version of the C51 compiler is the same as the full professional 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).

4.4. 3rd Party Toolsets

The ToolStick demos are written for the Keil toolset that is provided on the CD. The Silicon Laboratories IDE has native support for many other 8051 compilers. The full list of natively supported tools is:

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

5. Blink_LED Demo

The ToolStick kit includes two simple code examples. The first example is titled Blink_LED. 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 Blink_LED example code alternately turns on the red and green LEDs on the ToolStick board. The following steps describe how to connect and download the firmware, view and modify registers, use watch windows, use breakpoints, and single step through code.

5.1. 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. Connect the ToolStick to a USB port on the PC
3. In the IDE, go to Project → Open Project
4. Browse to *C:\SiLabs\MCU\Demos\ToolStick\Blink_LED*
5. Select Blink_LED.wsp and click OK
6. In the IDE, select Project → Rebuild Project
7. Go to Options → Connection Options
8. Select “USB Debug Adapter” for the Serial Adapter and “C2” for the Debug Interface, and then click “OK”
9. Go to Debug → Connect
10. 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 6) and downloaded to the device (step 10). 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 3.

5.2. 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 and red LEDs on the ToolStick will start to alternately flash. The debug commands on 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 pin on the device will hold their state. When the ToolStick firmware is stopped, one of the LEDs will be on and the other will be off.

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.

5.3. Viewing and Modifying Registers

All registers on the device can be viewed and modified when the device is in a halt 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 modify the status of the port pins directly from the IDE.

1. Open the Ports Debug Window from the View → Debug Windows → SFR’s → Ports menu option. The Ports Debug Window appears on the right-hand side of the IDE. The logic value of the Port 0 pins is stored in the P0 register. If the value of P0 is 0xF7, port pin P0.2 is high and the red LEDs on the ToolStick are on and the green LEDs are off. If the value of P0 is 0xFB, port pin P0.3 is high and the green LEDs are on and the red LEDs are off.
2. To enable both LEDs at the same time, the logic level of pins P0.2 and P0.3 need to be high. In the Ports Debug Window, double-click on the P0 value to move the cursor. Change the value of P0 to 0xFF, then select Debug → Refresh. This will write the value of 0xFF directly to the Port 0 pins and the pins will reflect the value immediately. This can be confirmed by looking at the ToolStick.
3. Click on the “Go” button or select Debug → Go. The device will resume execution and the LEDs will continue to blink synchronously.
4. Click on the “Stop” button.
5. In the Ports Debug Window, change the value of P0 to 0xF7 or 0xFB, and then click the Refresh button. This will have the LEDs blink alternately as before.

Changing the values of registers does not require recompiling the code or redownloading the firmware. At any time, the device can be halted and the values of the registers can be changed. The firmware will continue execution using the new values. This capability greatly speeds up the debugging process.

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

5.4. 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. The variable T2_Overflow_Count is a counter that stores the number of times the LEDs blink.

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 BLINK_LED.c to open the source file. The source file might already be open.
3. Scroll to the bottom of the source file and right-click on the variable "T2_Overflow_Count". In the context menu that appears, select "Add T2_Overflow_Count to Watch" and then choose "Default." On the right-hand portion of the IDE, the watch window appears and the T2_Overflow_Count 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 T2_Overflow_Count is incremented each time the LEDs blink.
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 from 0.

Changing the values of variables does not require recompiling the code or 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.

5.5. 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 T2_Overflow_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 bottom of the source file and right-click on the variable "T2_Overflow_Count". 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 the processor a few more times. Notice that the LEDs blink once for every time the processor is started and the T2_Overflow_Counter also increments by one.

5.6. 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 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 a breakpoint is not already set on the line of code that increments the T2_Overflow_Count variable, set the breakpoint using the steps described in Section 5.5.
2. Start the processor using the “Go” button and wait until 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. PWM Demo

In addition to the demo Blink_LED example software, the ToolStick CD also includes a demo project named PWM_LED. This demo firmware uses the hardware Programmable Counter Array modules on the C8051F300 in 8-bit PWM mode to drive the LEDs on the ToolStick.

The project and source files for the demo can be found in the *C:\SiLabs\MCU\Demos\ToolStick\PWM* folder.

7. Configuration Wizard 2

Configuration Wizard 2 helps accelerate development by automatically generating initialization source code to configure and enable the on-chip resources needed by most design projects. Configuration Wizard 2 is installed as part of the demo installation.

This chapter of the User’s Guide describes the basic features and capabilities of Configuration Wizard 2.

7.1. Configuration Wizard 2 Options

When Configuration Wizard 2 is opened, the program lists the device families that it supports. Based on the target device, first select the device family and then the specific part number.

The default source file that appears initially includes the header file for the device. As the initialization code is generated for each hardware peripheral, it is added to source file.

Configuration Wizard 2 can output the initialization code in either C or assembly. The selection between C and assembly is done using the Options → Code Format menu option.

7.2. Generating Initialization Code

As an example, the following steps show how to generate initialization code for the Programmable Counter Array module on a C8051F300. The code generated is similar to the PWM PCA initialization example code described in Section 6.

1. Open Configuration Wizard from the Start → Programs → Silicon Laboratories menu.
2. Select C8051F30x as the Device Family and C8051F300 as the Part Number.
3. Select the Peripherals → PCA menu option. The top part of the new window has the peripheral options. The lower part shows the initialization code.
4. In the PCA0 tab, select “Enable PCA0.” Notice the code that is inserted in the bottom half of the window.
5. In the Module 0 tab, select “8-bit Pulse Width Modulator”
6. In the Module 1 tab, select “8-bit Pulse Width Modulator”
7. In the Module 2/WDT tab, uncheck “Enable Watchdog Timer”
8. Click “OK.” The initialization code is inserted in a new function PCA_Init(). This function can be directly inserted in to a final project.

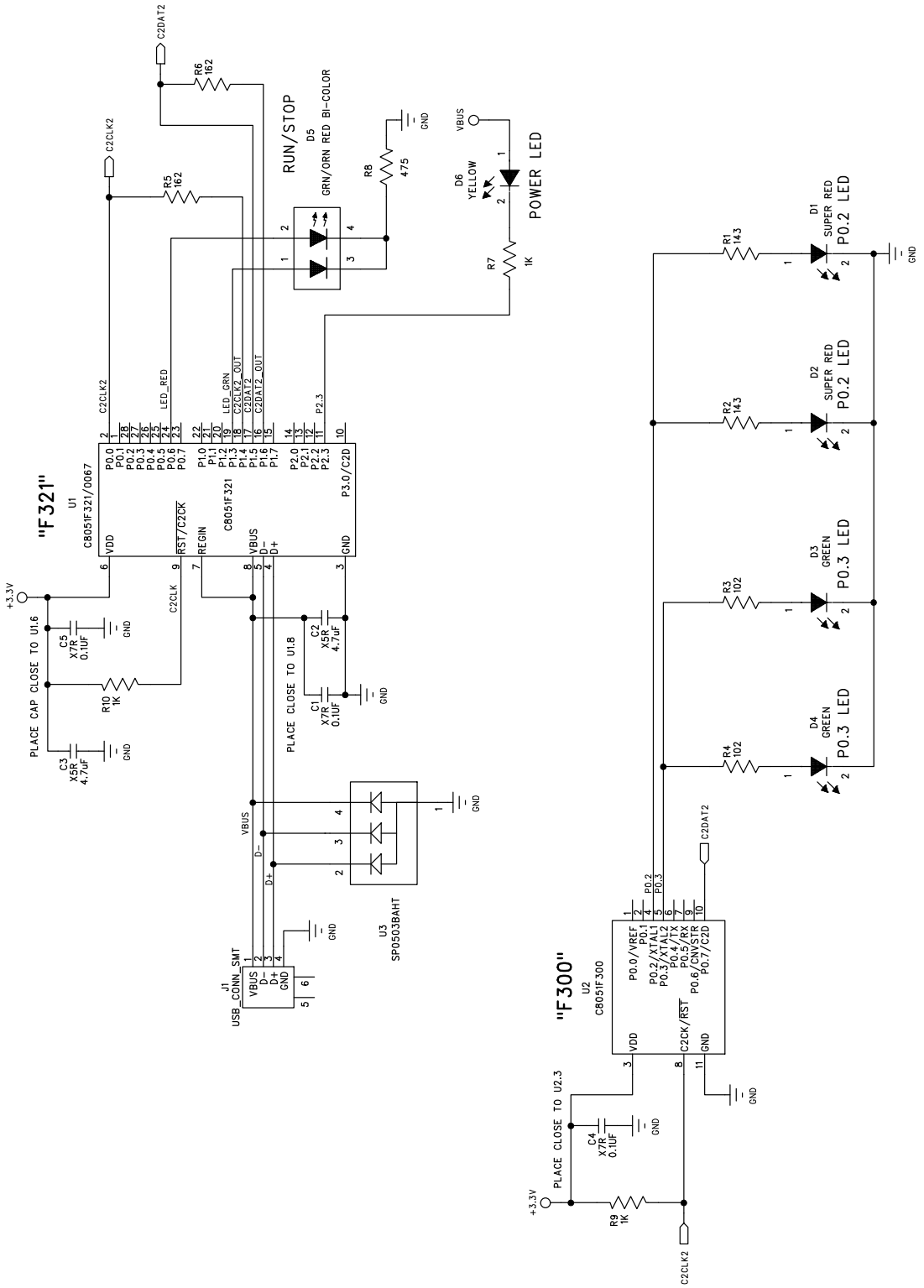
Configuration Wizard 2 can be used the configure the oscillators, port pins and other peripherals in a similar manner. The Peripherals menu of Configuration Wizard is customized based on the device family and part number. The options in the menu show which peripherals are supported. More information regarding any peripheral can be found in the device family’s respective data sheet.

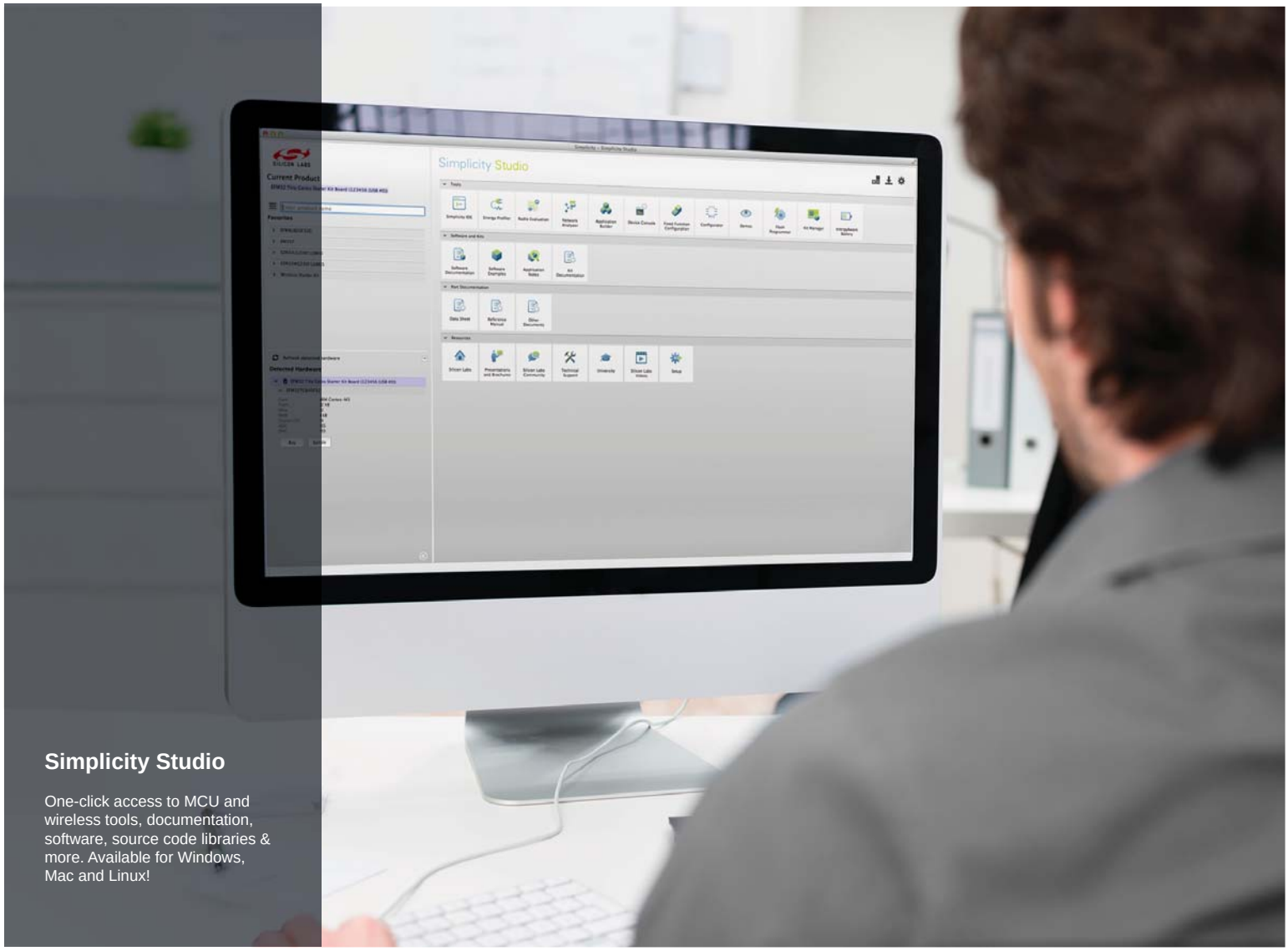
8. Information Locations

Example source code is installed in the “C:*SiLabs**MCU**Demos**ToolStick*” directory during IDE installation.

Documentation for the ToolStick kit, including this User’s Guide, can be found in the C:*SiLabs**MCU**Demos**ToolStick**Documentation*\ folder.

9. Schematic


"F321"
"F300"
RUN/STOP
POWER LED
P0.2 LED
P0.3 LED
PLACE CAP CLOSE TO U1.6
PLACE CLOSE TO U1.8
PLACE CLOSE TO U2.3
C2CLK2
C2DAT2
LED_RED
LED_GRN
VBUS
GND
+3.3V



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