Keywords: MAXQ1850, MAXQ1103, DS5250, DS5002, microcontroller, secure microcontroller, uC, DES, 3DES, RSA, ECDSA, SHA, USB smart card, ISO 7816, EMV, integrated circuit card, IC card, POS terminal, banking terminal, ATM, payment terminal, PIN pad, access control

APPLICATION NOTE 4312

Getting Started with DeepCover Secure Microcontroller (MAXQ1850) EV KIT and the CrossWorks Compiler for the MAXQ30

Dec 22, 2009

Abstract: This application note describes how to create, build, and debug applications targeted to the DeepCover® Secure Microcontroller (MAXQ1850). The examples use the MAXQ1850 evaluation (EV) kit and the CrossWorks C compiler available from Rowley Associates.

Introduction

Maxim's DeepCover® Secure Microcontroller (MAXQ1850) is a high-performance, high-security, low-pin-count, 32-bit RISC microcontroller designed for electronic commerce, banking, and data-security applications. The microcontroller executes 16-bit instructions and has a 32-bit data path. Most instructions execute in a single clock cycle, making the MAXQ1850 a very high-performance RISC microcontroller. The MAXQ1850 also has a number of important security features, including:

- 2048-bit modular arithmetic accelerator supporting RSA, DSA, and ECDSA
- Cryptographic hardware accelerators supporting AES-128, AES-192, AES-256, SHA-1, SHA-224, SHA-256, DES, and 3DES
- True hardware random number generator
- 8KB low-leakage battery-backed NVSRAM
- Four self-destruct inputs
- Tamper detection with fast key/data erase
- Environmental sensors (e.g., temperature, voltage) to detect out-of-range conditions

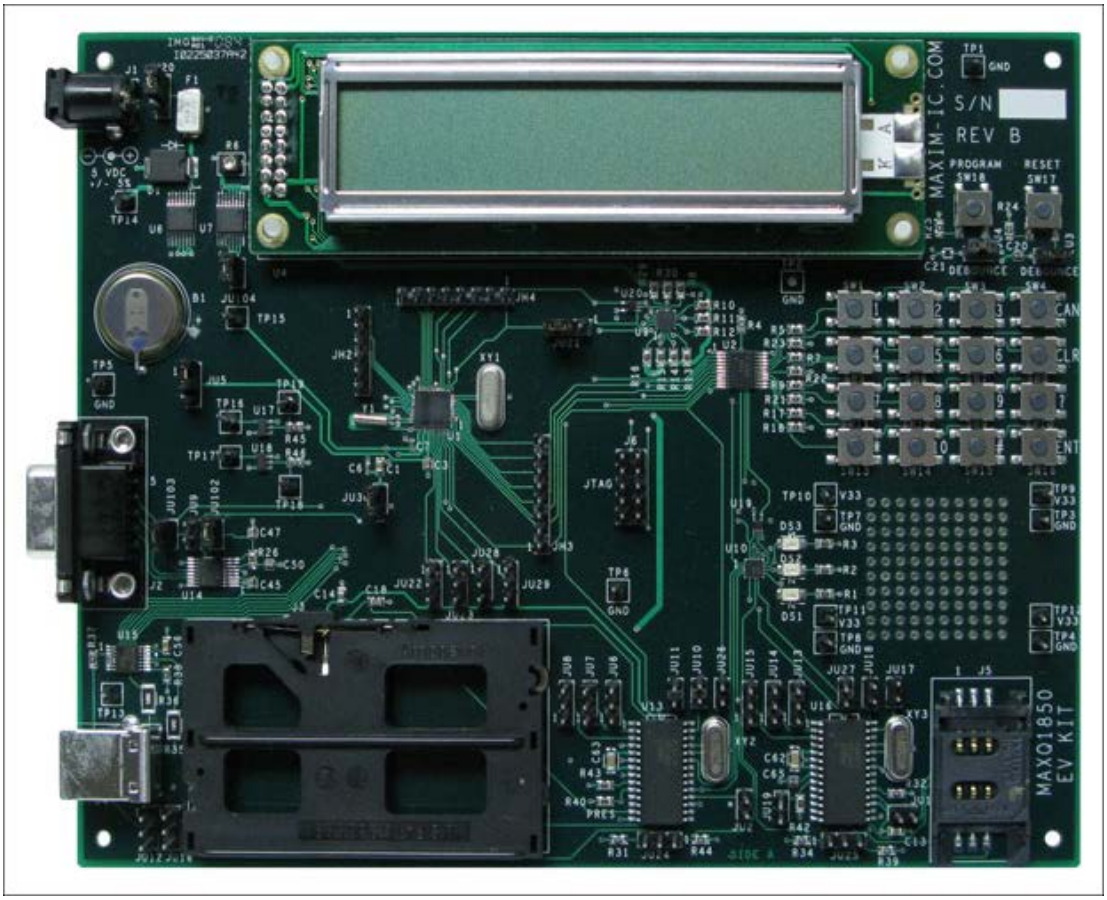
The MAXQ1850-KIT, an EV (evaluation) kit, is an ideal platform for prototyping secure applications. The EV kit provides an RS-232 serial port, two smart-card slots (one full size and one SIM card), a USB connector, an LCD screen, a 16-pushbutton keypad, and a prototyping area.

Setting up the MAXQ1850 EV Kit

The MAXQ1850-KIT board is shown in **Figure 1**. The following hardware components are contained in the EV kit package, and are used for implementing this application note:

1. MAXQ1850-KIT board
2. JTAG board
3. JTAG cable (connects MAXQ1850-KIT board and JTAG board)
4. 9-pin serial cable
5. Two regulated power supplies (5V, ±5%, 300mA, center positive)

Note: Rev C kits come with a 1000mA supply.



More detailed image (PDF, 5.6MB)
Figure 1. MAXQ1850-KIT board (Rev B).

The MAXQ1850-KIT board has a number of jumpers to configure. For a complete list of jumpers and their function, see the data sheet. For this application note, configure the jumpers as follows:

- JU30 (near bottom left of the MAXQ1850).
- Connect pin 1 (square pad on PCB) and pin 2 on jumpers JU5 (near battery) and JU20 (near power input).
- Connect pins 2 and 3 on jumper JU21 (near top right of the MAXQ1850).
- All other jumpers should be open.

Note: some examples included with the MAXQ1850-KIT CD require different jumper settings. Refer to the example's README.txt file for jumper placement.

Connect the JTAG cable between the JTAG board (P2) and the MAXQ1850-KIT (J6) board. On the JTAG board, the red stripe of the cable should connect to the side of the connector labeled pin 1 and pin 2. On the MAXQ1850-KIT board, connect the red stripe of the cable to pin 1, **TCK**. (Pin 1 can be identified by the square pad on the back of the PCB.)

Connect the 9-pin serial cable between your PC and the JTAG board. Finally connect the 5V power. On Rev A and B kits, connect a power supply to the power connector (J2) on the JTAG board and to the EV kit board (J1). On Rev C boards first close jumper JU31 on the EV kit board, and JH3 on the JTAG board. Then connect the power supply to the EV kit board (J1). The Rev C board will supply power for the JTAG board.

Getting Started with the CrossWorks Compiler

To begin using the MAXQ1850-KIT, we will create a simple application program that blinks the 3 LEDs on the board. By blinking the LEDs in a fixed, repetitive sequence, they appear to be "walking" on the board. The source code is available on the EV kit's CD or for download from Maxim. Unzip this archive to: C:\MAXQ1850\LED.

The tool suite that Maxim uses is CrossStudio, available from Rowley Associates. This document was created using the CrossWorks IDE for the MAXQ30, version 2.0.0.2009012302.4045. To confirm the most current available revision, check the Rowley Associates website, or contact Maxim. Refer to **Appendix A** for more information about Rowley CrossWorks.

To create a new solution, go to **File → New → New Project**. From the **New Project** pop-up, fill in the **Name** and **Location** boxes at the bottom, and select the **Executable** category and **A C executable** from the **Project Templates** window (**Figure 2**). We will call our project **LED** and put it in the directory **C:\MAXQ1850\LED**.

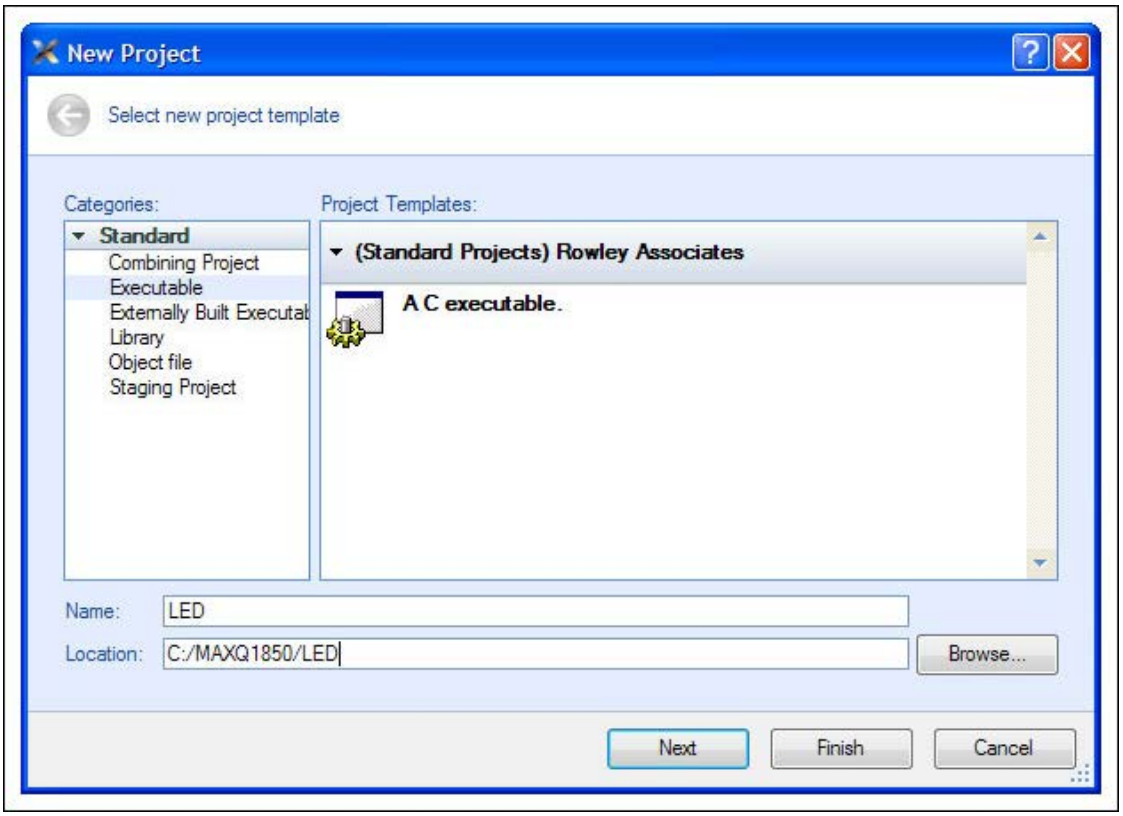


Figure 2. New project screen.

Click **Next** to continue, and you will see the **Project Properties** window (**Figure 3**). It is possible that the **Target Processor** is currently the MAXQ1103. By double clicking on the processor part number, you can select the MAXQ1850 processor. Click Next again and remove the check mark from **Project files → `main.c`** (**Figure 4**). We will be adding our own files. Click Finish to create the project.

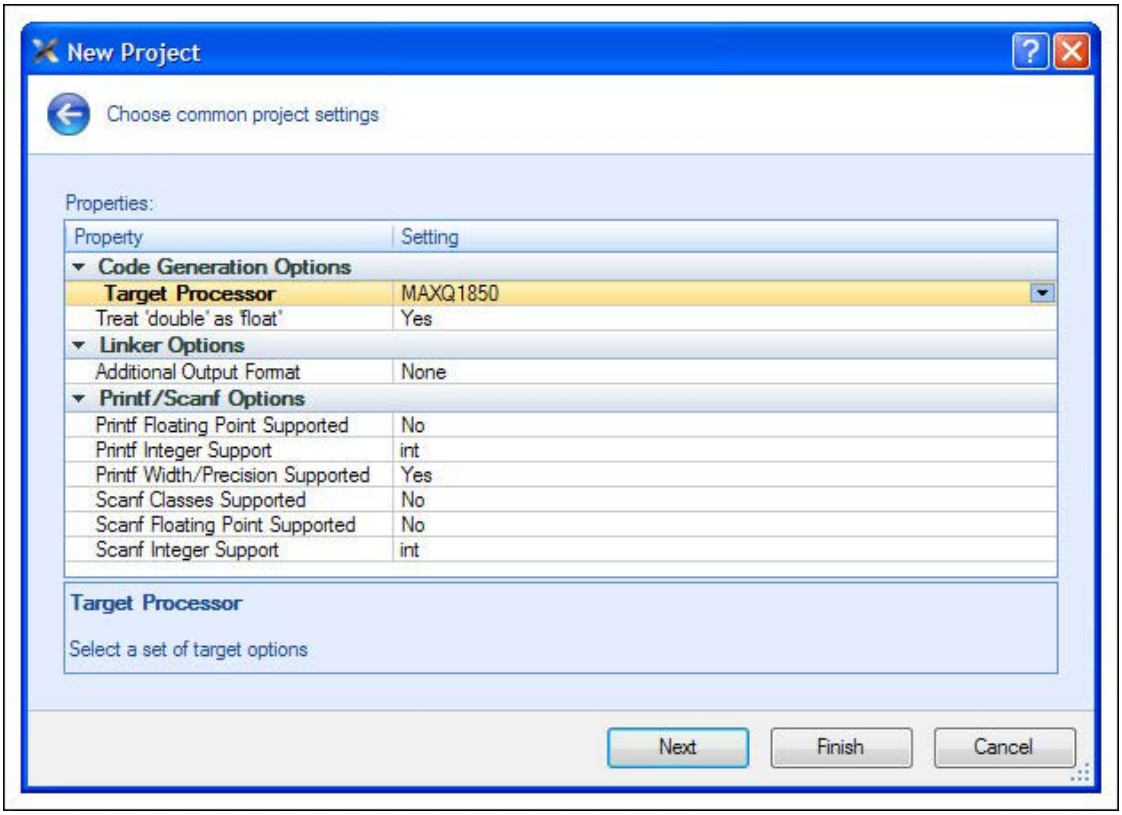


Figure 3. Selecting the MAXQ1850 processor.

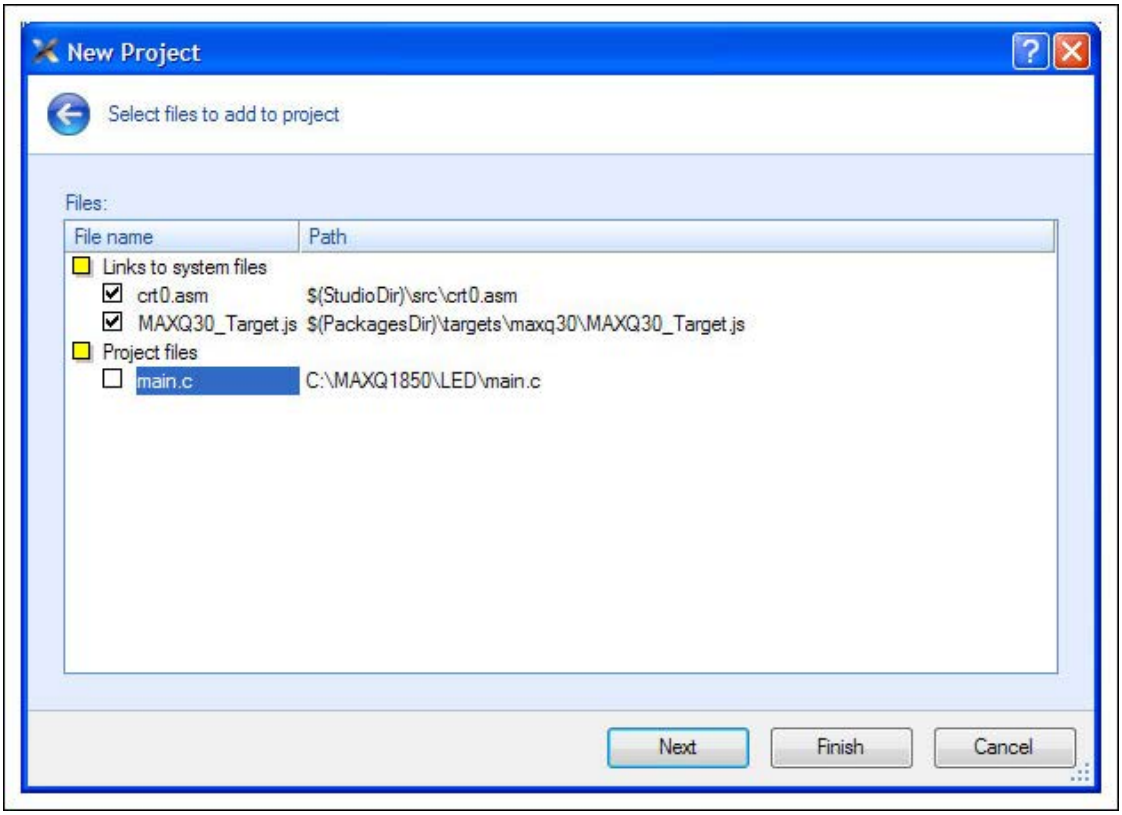


Figure 4. New project options.

When the project is created, there will be a new project in the **Project Explorer** window (**Figure 5**), usually located in the upper right side of the application window. Expand project LED and you will see two folders, **Source Files** and **System Files**. Right click Source Files and select **Add Existing File**. Browse to **C:\MAXQ1850\LED** and select both `main.c` and `spi.c`, then click **Open**.

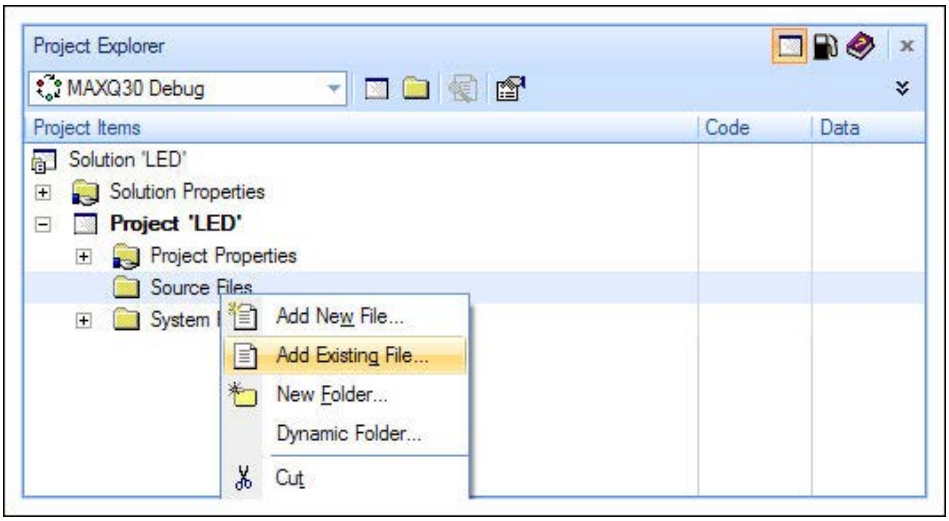


Figure 5. Project Explorer window.

When this application is executed, you should see LEDs DS1, DS2, and DS3 (located immediately to the left of the prototyping area on the EV kit board) blink on and off in sequence. However before the application can be run, it must be "built." Select **Build → Build LED**, or alternatively press **F7**. If everything builds properly, you will see the

message **Build complete** with a check mark beside it in the **Output** window (**Figure 6**).

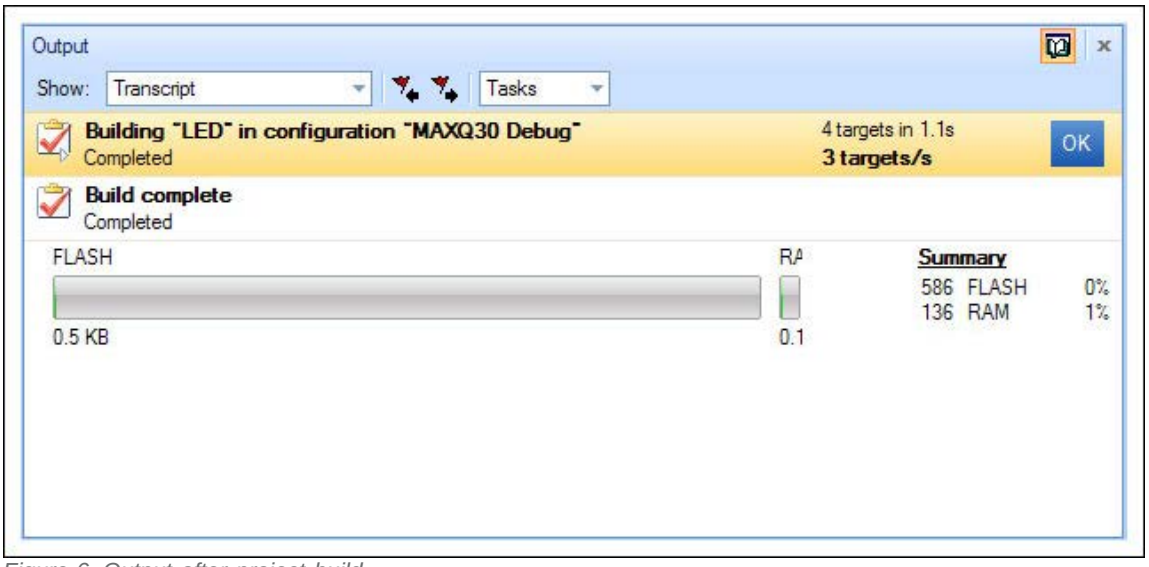


Figure 6. Output after project build.

Before attempting to load the application, set up the JTAG Debug Port in the **Targets** window. Double click **Maxim Serial JTAG Adapter** changing it to bold font, and look below at the **Properties Window** for information. You will see a list of properties and their settings as shown in **Figure 7**. Set **Connection → Port Name** to the serial port to which the JTAG board is connected.

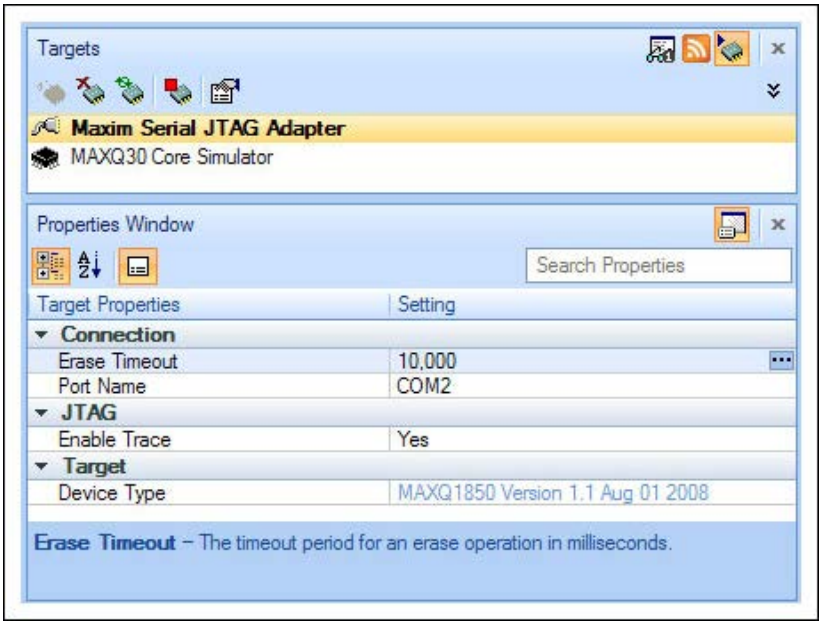


Figure 7. Target Properties Window.

To load the application use **Debug → Go**, or press **F5**. The program will load onto the EV kit with JTAG and pause at the start of the `main` function. If the **Go** option is not available or the program does not load, refer to **Appendix B** for troubleshooting.

To run the application from this point, select **Debug → Go** (or click the button that looks like a **Play** button). Now verify that the LEDs on the MAXQ1850-KIT board are blinking. You may want to alter the application somewhat now;

try blinking the LEDs in reverse sequence, or vary the amount of time that they are lit so the blinking becomes faster or slower.

Using CrossStudio to Debug an Application

Now we will explore some of the debugging capabilities of the MAXQ1850 and the CrossStudio tool. The MAXQ1850 has a built-in JTAG engine that allows debugging on the actual silicon, thus eliminating the need for expensive emulators or potentially inaccurate simulators. Note that the MAXQ1850 also has a security locking mechanism that will prevent JTAG from working when the part has been locked. This ensures that the JTAG debug engine is not a security threat on MAXQ1850 devices deployed in sensitive applications.

Consider the LED application. First quit the current debugging session using **Debug ⟶ Stop**. As an experiment, change the delay counter from `200000` to `2000` in the `Delay` function within `main.c` function.

```
for(i=0;i < 2000; i++) ;
```

Now build and run the application by selecting **Build ⟶ Build and Debug**. The toolset will rebuild the project, load the new program, and start its execution. Note that the LEDs now appear to be continuously lit, rather than blinking on and off.

By selecting the **Pause** button (or select **Debug ⟶ Break**), the program execution will halt at its current line of code, and a yellow arrow will appear in the left margin. You should expect the code to stop in the `for` loop of the `Delay` function (see **Figure 8**), since this program spends a significant portion of its time here.

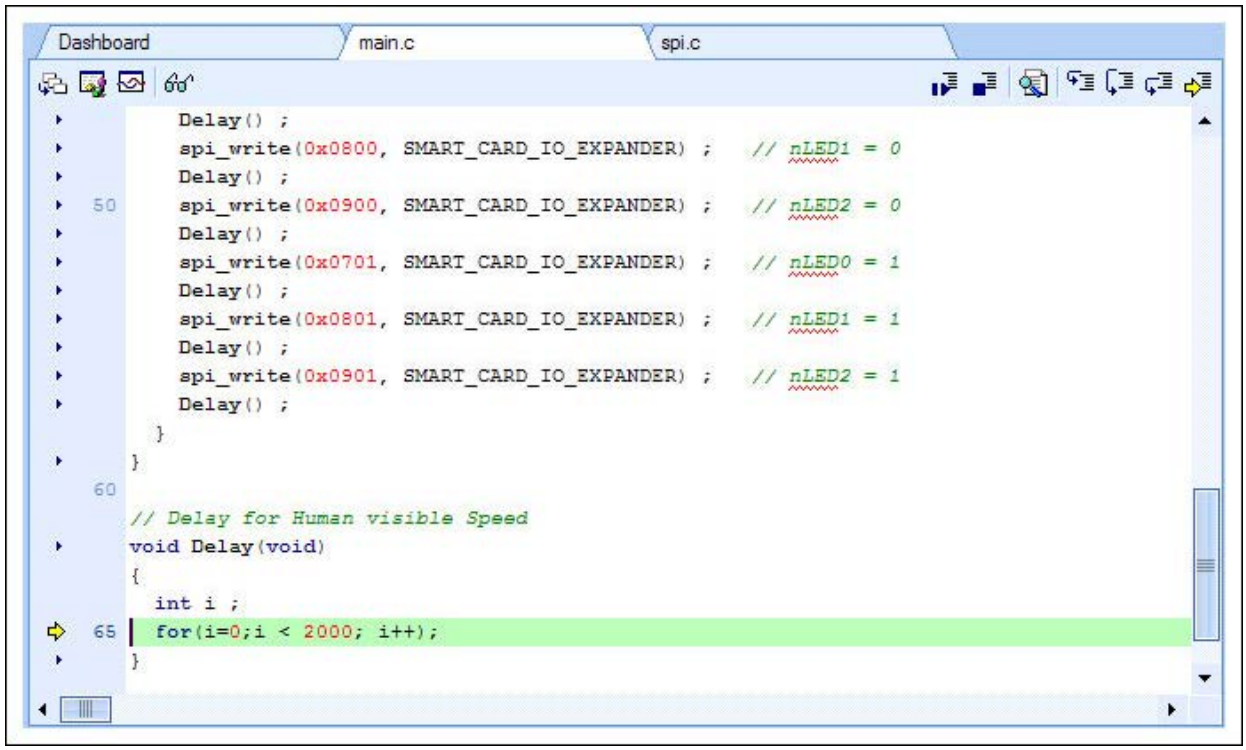


Figure 8. Code stopped execution in the `Delay` function.

Look at the Locals window on the right. (If it is not visible, go to **Debug ⟶ Debug Windows ⟶ Locals**.) This window will show the current value for the `i` variable. Now press the **Step Over** button, or **Debug ⟶ Step Over**. Let the program run for a second, and then press the **Pause** button again. You should see that the value of `i` has increased.

To exit the function, we could continue to press the **Step Over** button until the loop ends, but that will take a long time. By simply pressing the **Step Out** button or **Debug ⟶ Step Out**, the program is executed until the `Delay` function is exited and execution is returned to its calling function `main.c`.

You can achieve a similar result by setting a breakpoint. To set a breakpoint on any line calling the `Delay` function within the function `main.c`, click on the small triangle to the left of one of these lines of code. It will become a red circle (**Figure 9**). Now run the application again (**Debug ⟶ Go** or the **Play** button). The application will run to that point and halt.

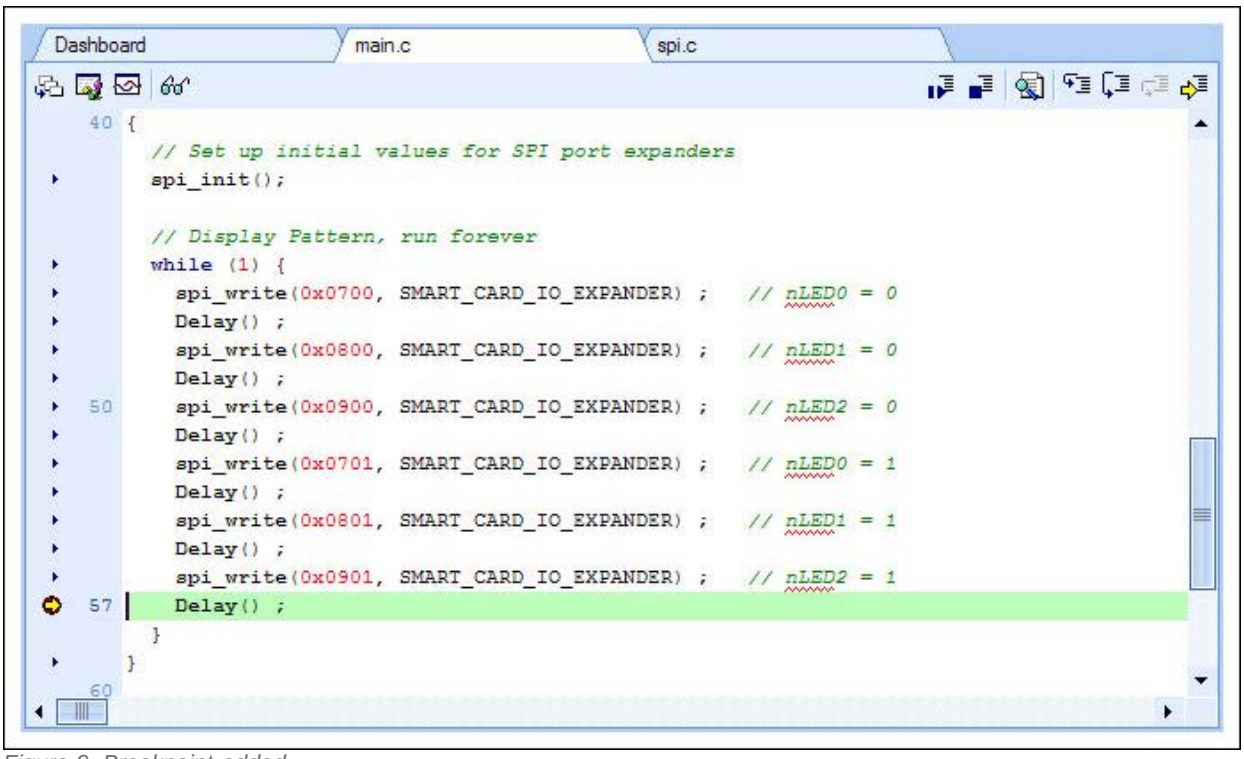


Figure 9. Breakpoint added.

We will now explore more debug features. Press the **Step Over** button several times. A line of C source code will be executed with every press. You will see the LEDs blink as you pass the lines that control each one. When paused at one of the `Delay()` lines, press the **Step Into** button or **Debug ⟶ Step Into**. This will step into this function and halt execution at its first executable line. As demonstrated earlier, you can exit the `Delay()` function with one click by pressing the **Step Out** button.

It is also possible change variables (and registers) while running. Click **Go** then click **Pause**, and the program should be stopped in the middle of the `Delay()` function again. Note the value of i Now, try setting `i` to `1998`. (Click on the value that it displays for `i` and enter `1998` when highlighted.) Click the **Step Into** button and you should see the loop end because the terminal value of `i` is reached.

There are some other debugging features that may be of interest:

- **Debug ⟶ Disassembly** will display a mix of C code and the generated assembly code. This lets the user step through the assembly code instead of the C code, and also shows the C code as it executes.
- **Debug ⟶ Debug Windows ⟶ Call Stack** will display the functions that have been called for the application to reach its current point. If execution is paused while in the `Delay()` function, the display will look something like **Figure 10**.

Figure 10. Call Stack while running in the `Delay()` function.

For More Information

An EV kit CD is available with example code for all the MAXQ1850's features. To request this CD or if you have any problems with this application note, please contact Microcontroller Support.

Formal documentation is available for Rowley CrossWorks for MAXQ30 here.

Appendix A. Rowley CrossStudio

Rowley CrossWorks IDE Version

The LED example and all of the examples found on the EV Kit CD were developed with the most recent version of Rowley CrossWorks for MAXQ30, Release 2.0.0.2009012309.4045. It is **VERY IMPORTANT** to use this version of the IDE. Project files from this version of the IDE are **NOT** compatible with older versions. To see what release you have, open **CrossStudio → Help → About CrossStudio**.

Obtaining Rowley CrossWorks for MAXQ30

The CrossStudio for MAXQ30 is available from Rowley at their website: http://www.crossstudio.co.uk/maxq/index.htm

Under "Latest Downloads → CrossWorks for MAXQ30" choose "Version 2.0 for Windows." Download and install the executable.

Rowley provides a 30-day evaluation period for the CrossWorks IDE. Follow the instructions.

To purchase a license for Rowley CrossWorks for MAXQ30. You can also email at sales@rowley.co.uk or phone +44(0)1453 549536.

Appendix B. Troubleshooting

Compiling Issues with Example Projects

If issues occur while attempting to compile an example from an included project file, the most likely problem is version incompatibilities in the Rowley CrossWorks compiler. This incompatibility can cause strange behavior such as: project not building, not generating correct output files, failure to initiate debug session, debug buttons grayed out, errors stating that the vendor package is not installed, etc.

To correct this problem, recreate the project file with the currently installed IDE. First remove the current .hzp and .hzs files from the project directory. Follow the instructions above or from section D of the EV Kit CD README.txt file. The project should now build and debug correctly.

Issues with the Serial-to-JTAG Board

If Rowley CrossWorks reports problems connecting to the MAXQ1850-KIT with the Maxim serial JTAG adapter, try the following steps one at a time:

- Verify that the JTAG programming cable is correctly oriented. Make sure that the red stripe goes to pin 1 on both the serial JTAG adapter board (Connector P2) and the MAXQ1850 (Connector J6). Pin 1 can be verified by the square pad on the back of the EV kit PCB.
- Verify that both the serial JTAG adapter board and the EV kit have power connected. The serial JTAG adapter board LED DS1 should be lit; on the MAXQ1850-KIT board the top row of the LCD screen should have some faint squares visible.
- Verify that the correct serial communications port is selected in Rowley CrossWorks. Double click Maxim serial JTAG adapter from the **Targets** window, verify that the port name corresponds to the serial port connected to the serial JTAG adapter board.
- Disconnect the power cable from both the serial JTAG adapter board, and the EV kit. Right click on **Maxim Serial JTAG Adapter** in the **Targets** window, and choose **Disconnect**. Reconnect both power cables, and select **Connect**.
- Reload the firmware on the serial JTAG adapter board, following the instructions in application note 4027, "How to Update the Firmware in the MAXQ Serial-to-JTAG Board."

Related Parts		
DS5002	Secure Microprocessor Chip	Free Samples
DS5250	High-Speed Secure Microcontroller	
DS8007	Multiprotocol Dual Smart Card Interface	Free Samples
DS8023	Smart Card Interface	Free Samples
DS8024	Smart Card Interface	Free Samples
DS8113	Smart Card Interface	Free Samples
MAXQ1103	DeepCover Secure Microcontroller with Rapid Zeroization Technology and Cryptography	
MAXQ1850	DeepCover Secure Microcontroller with Rapid Zeroization Technology and Cryptography	Free Samples

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 4312: http://www.maximintegrated.com/an4312
APPLICATION NOTE 4312, AN4312, AN 4312, APP4312, Appnote4312, Appnote 4312