

Keywords: MAXQ1850, MAXQ1103, DS5250, DS5002, microcontroller, secure microcontroller, uC, DES, 3DES, RSA, ECDSA, Oct 17, 2008
SHA, USB smart card, ISO 7816, EMV, integrated circuit card, IC card, POS terminal, banking terminal, ATM, payment terminal, PIN pad, access control, pay tv, set top box

APPLICATION NOTE 4312

Getting Started with the MAXQ1850 Evaluation Kit (EV Kit) and the CrossWorks Compiler for the MAXQ30

Abstract: This application note describes how to create, build, and debug applications targeted to the MAXQ1850 high-performance, RISC, secure microcontroller. The examples use the MAXQ1850 evaluation kit (EV kit) and the CrossWorks C compiler available from Rowley Associates.

Introduction

Maxim Integrated Product's [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 machine. The MAXQ1850 also has a number of important security features, including:

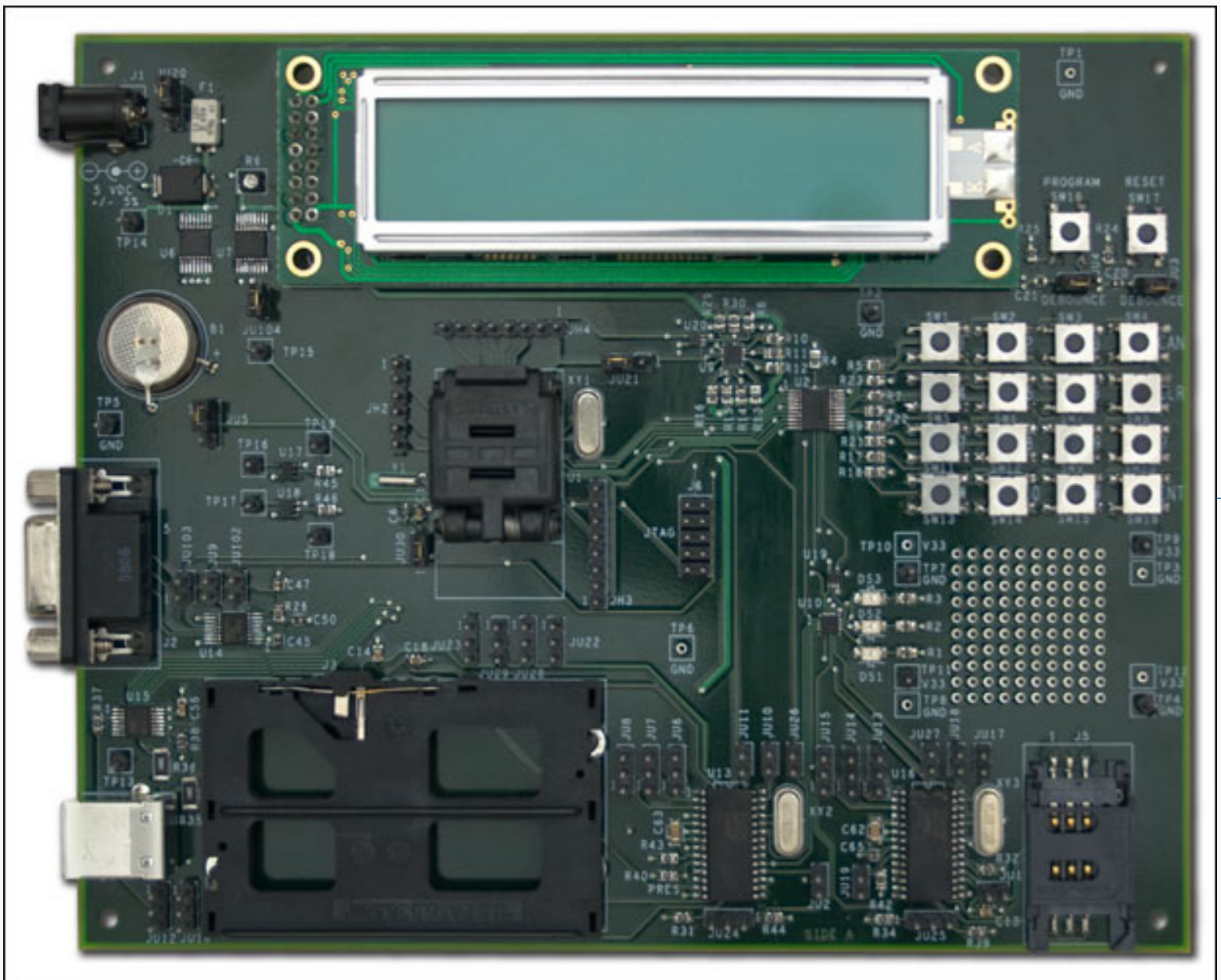
- Cryptographic accelerators supporting DES, 3DES, AES, SHA-1, SHA-224, SHA-256, RSA, DSA, and ECDSA
- True hardware random number generator
- 8KB low-leakage battery-backed NVSRAM
- Four self-destruct inputs
- Tamper detection with fast key/data erase
- Environmental sensors (e.g., temperature, voltage)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

The MAXQ1850 evaluation (EV) kit is an ideal platform for prototyping secure applications. The 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 EV 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 EV kit board
2. JTAG board
3. JTAG cable (connects MAXQ1850 EV kit board and JTAG board)
4. 9-pin serial cable
5. Regulated power supply (5V, $\pm 5\%$, 300mA, center positive)



[More detailed image](#) (PDF, 776KB)

Figure 1. MAXQ1850 EV kit board.

The MAXQ1850 EV kit board and JTAG board have a number of jumpers to configure. For a complete list of jumpers and their function, see the respective data sheets. For this application note, configure the jumpers as follows:

- On the MAXQ1850 EV kit board, close the following jumpers: JU3 (near Reset switch); JU4 (near Program switch); JU30 (near bottom left of processor); and JU104 (near bottom left of LCD module). Also 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 processor). All other jumpers should be open.
- On the JTAG board, close JH3. This provides 5V power from the JTAG board to the EV kit board. Jumpers JH1 and JH2 are "Don't Cares" for this configuration.

Connect the JTAG cable between the JTAG board and the MAXQ1850 kit 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 EV kit board, connect the red stripe of the cable to the side of the connector labeled pins 1 and 2. (Pin 1 can be identified by the square pad on the back of the PCB.)

Note that on an older MAXQ1850 EV kit board design, a socket may have been used for the MAXQ1850 microcontroller. If so, insert your MAXQ1850 into the socket with the lead-free indicator "+" toward the dot in one corner (bottom right) inside the socket.

Connect the 9-pin serial cable between your PC and the JTAG board. (Do not connect it to the MAXQ1850 kit board.) Finally, connect the 5V power supply to the power connector (J2) on the JTAG board. This will also supply power to the EV

kit board.

Getting Started with the CrossWorks Compiler: WalkLED

To begin using the MAXQ1850 EV kit, we will create a simple application program that blinks the three LEDs on the board. By blinking the LEDs in a fixed, repetitive sequence, they appear to be "walking" on the board. Hence the project is named WalkLED. The code for this program is shown in **Appendix A**. The code is short enough that you can manually type it in a relatively short period. However if you prefer, the source code file is also available on the EV kit's CD or for download from [Maxim](#).

The tool suite we use is CrossStudio, available from [Rowley Associates](#). When this document was created, the tool suite was CrossWorks for the MAXQ30, version 2.0.0.2008063000.2293. This was the version used to produce the screen shots in this document. To confirm the most current revision, check the Rowley & Associates website, or send an email to micro.support@maxim-ic.com.

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 WalkLED_demo and put it in the directory C:\work\maxq\MAXQ1850\WalkLED_demo.

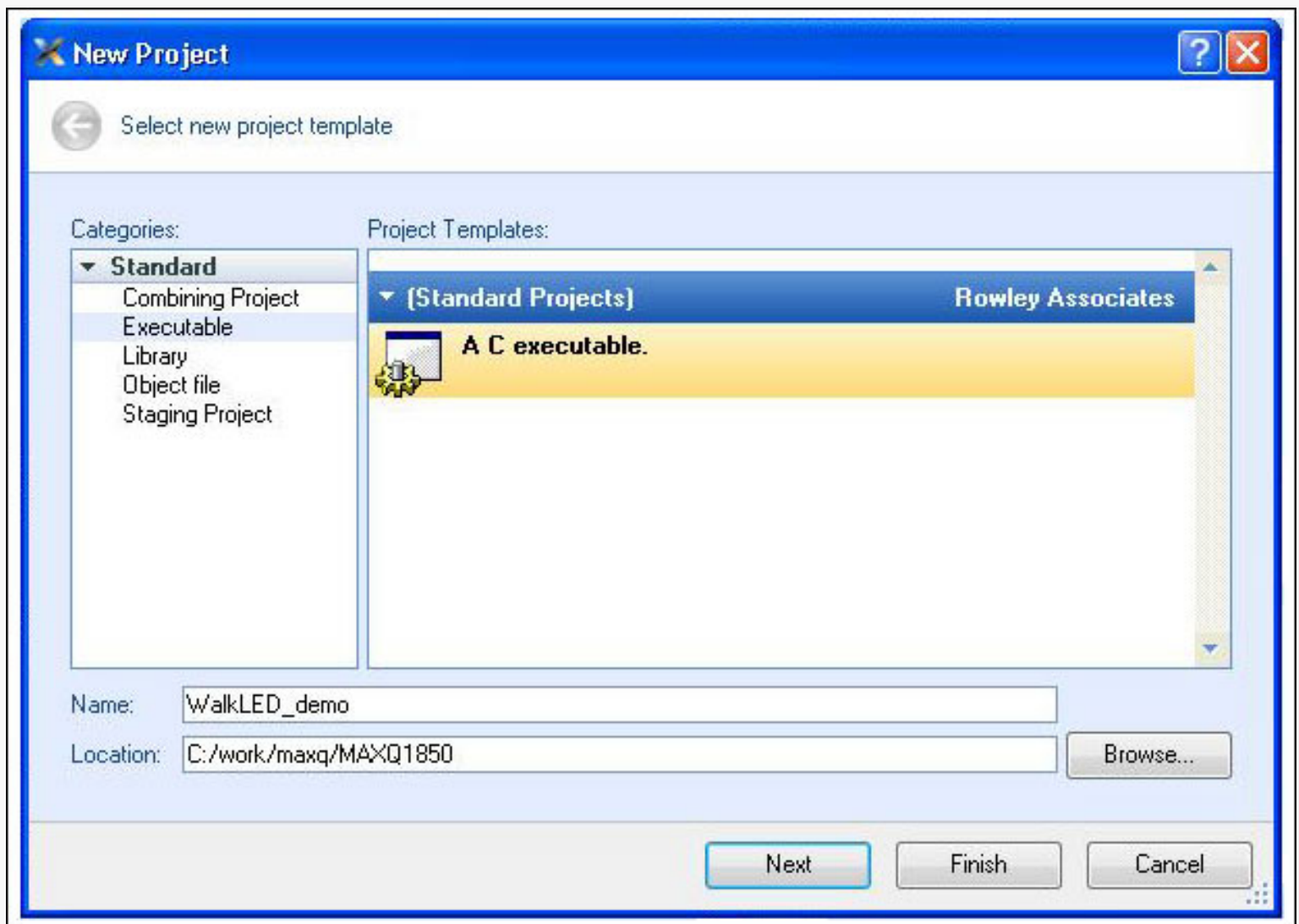


Figure 2. New project screen.

Click Next to continue, and you will see a Project Properties pop-up box (**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. Everything else on this and the subsequent pages is acceptable, so click Finish after selecting the MAXQ1850. The project will be created. If you want, you can explore other project options by clicking Next.

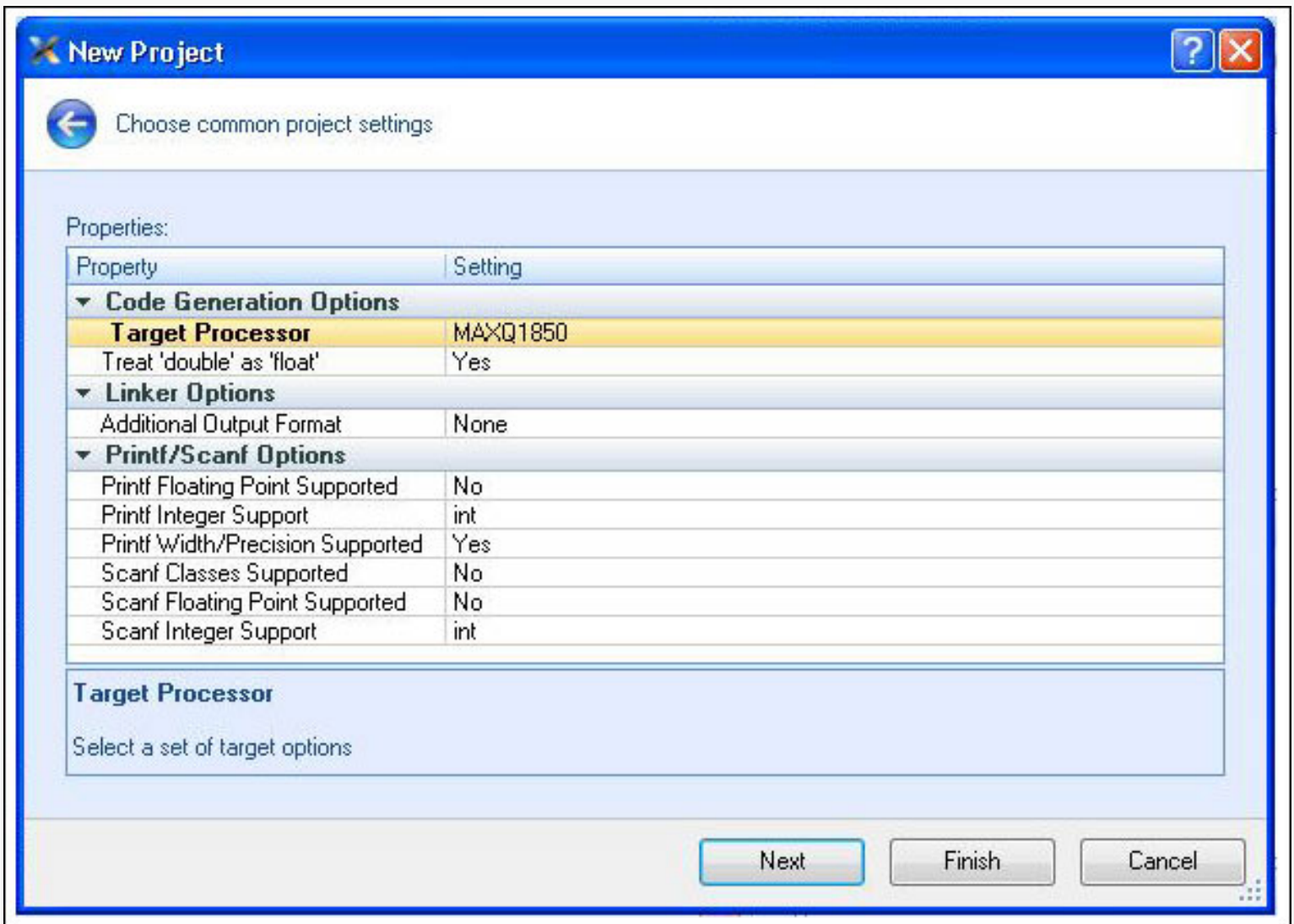


Figure 3. Selecting the MAXQ1850 processor.

When the project is created, there will be a new project in the Project Explorer window (Figure 4), usually located in the upper right side of the application window. Open it, and you will see two folders, Source Files and System Files. Open the Source Files and you will see `main.c`, your application source code. Double click it to open.

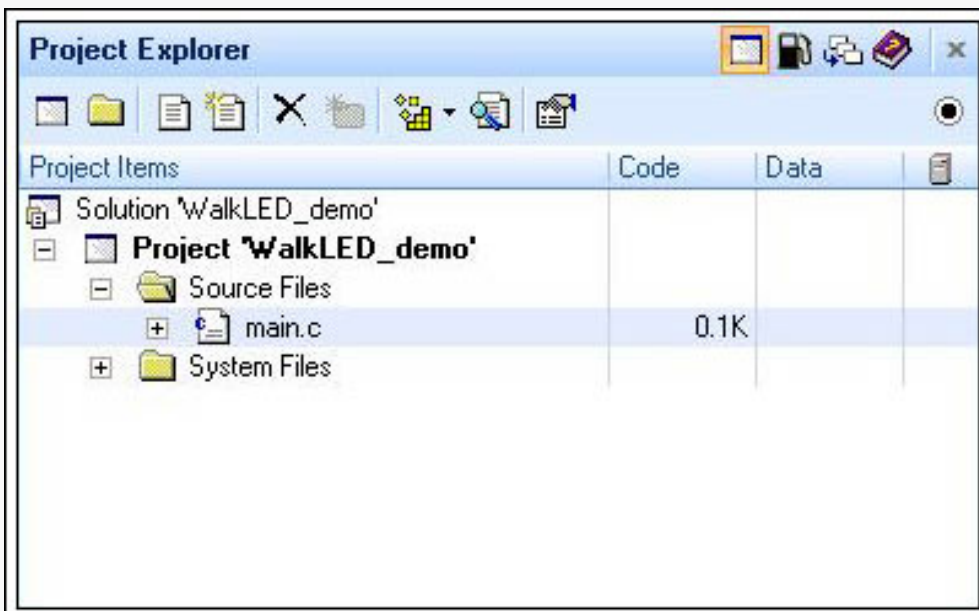


Figure 4. Project Explorer window.

Now type the application of Appendix A, or cut and paste from one of the sources mentioned above, into the `main.c` file,

thus replacing all of its current contents.

When this application is executed, you should see LEDs DS1, DS2, and DS3 (located immediately to the left of the prototyping area on the kit board) blink on and off in sequence. However before the application can be run, it must be "built." Select Build → Build WalkLED_demo, or alternatively, you can press F7. If everything builds properly, you will see the message "Build complete" with a check mark beside it in the Output window (**Figure 5**). If there are errors, make sure that you entered the code correctly.

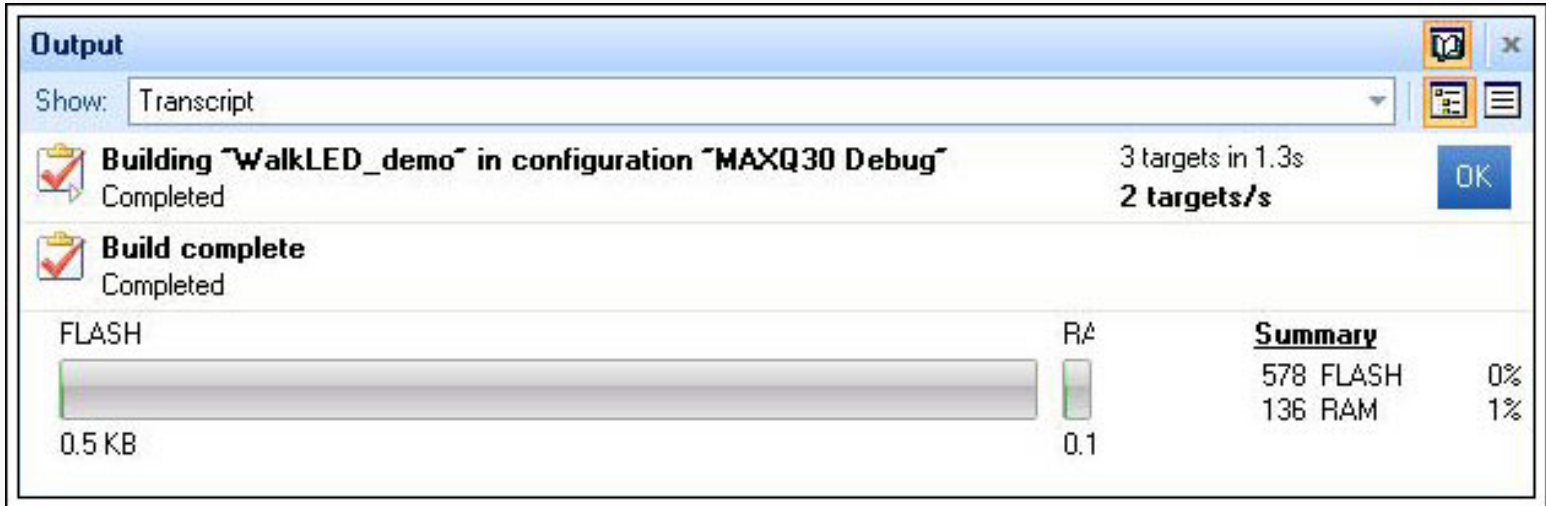


Figure 5. Output after project build.

You can now run the application. To do so, go to Debug → Build and Debug, and the program will start, then continue running. Clicking on the Pause button (Break Execution, as above and left of the code window) will halt the program at its current location and wait for the user's next action. If the Build and Debug option is not available, go to the "targets" window; click on the Maxim Serial JTAG Adapter; and then click on the Connect button (top leftmost corner of the window). This will connect the toolset to the EV kit through the JTAG board, and enable the debugging capability. Once the toolset's debugging capability is enabled, you can begin the debug process by simply clicking on the Step Over button shown in **Figure 6**.



Figure 6. Step Over button.

When the program is started with the Step Over button, CrossStudio will download the application to the MAXQ1850 through the JTAG board while status messages appear in the Output window. The application will begin to run and then pause on the first line of code (denoted by the yellow arrow in the left margin). 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 board are blinking. You may want to alter the application somewhat now; try to blink the LEDs in reverse sequence, or vary the amount of time that they are lit so the blinking becomes faster and 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.

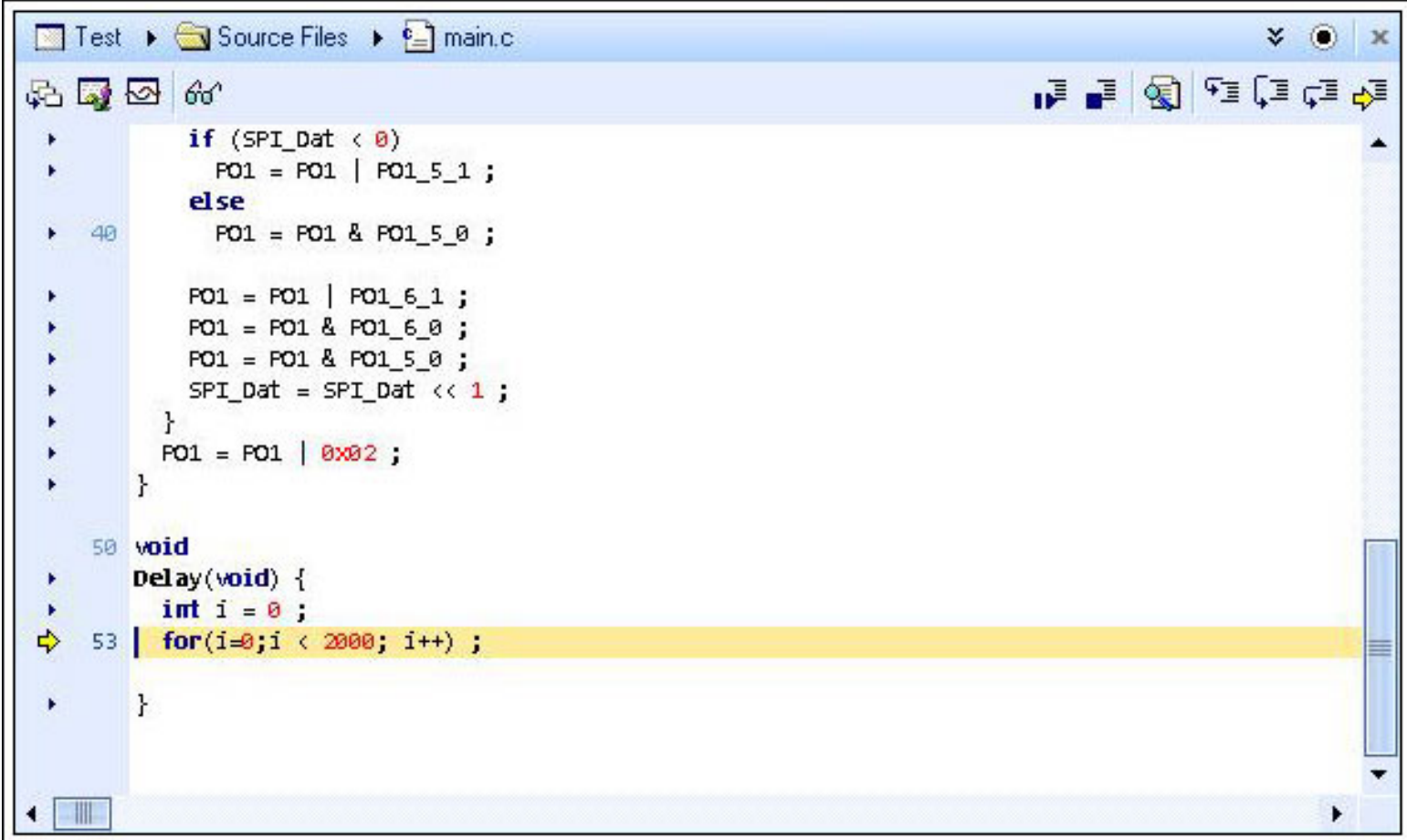
Consider the WalkLED_demo application. As an experiment, change the delay counter from 200000 to 2000 in the Delay function within main.c function.

```
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 7**), since this program spends a significant portion of its time here.



```
Test > Source Files > main.c
if (SPI_Dat < 0)
    PO1 = PO1 | PO1_5_1 ;
else
    40 PO1 = PO1 & PO1_5_0 ;

    PO1 = PO1 | PO1_6_1 ;
    PO1 = PO1 & PO1_6_0 ;
    PO1 = PO1 & PO1_5_0 ;
    SPI_Dat = SPI_Dat << 1 ;
}
PO1 = PO1 | 0x02 ;
}

50 void
Delay(void) {
    int i = 0 ;
    53 | for(i=0;i < 2000; i++) ;
}
}
```

Figure 7. 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. 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 (to the right of the Step Into button in **Figure 6**), the program is executed until the `Delay` function is exited and execution is returned to its calling function `main.c`.

You could also 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 8**). Now run the application again (**Debug** → **Go** or the Play button). The application will run to that point and halt.

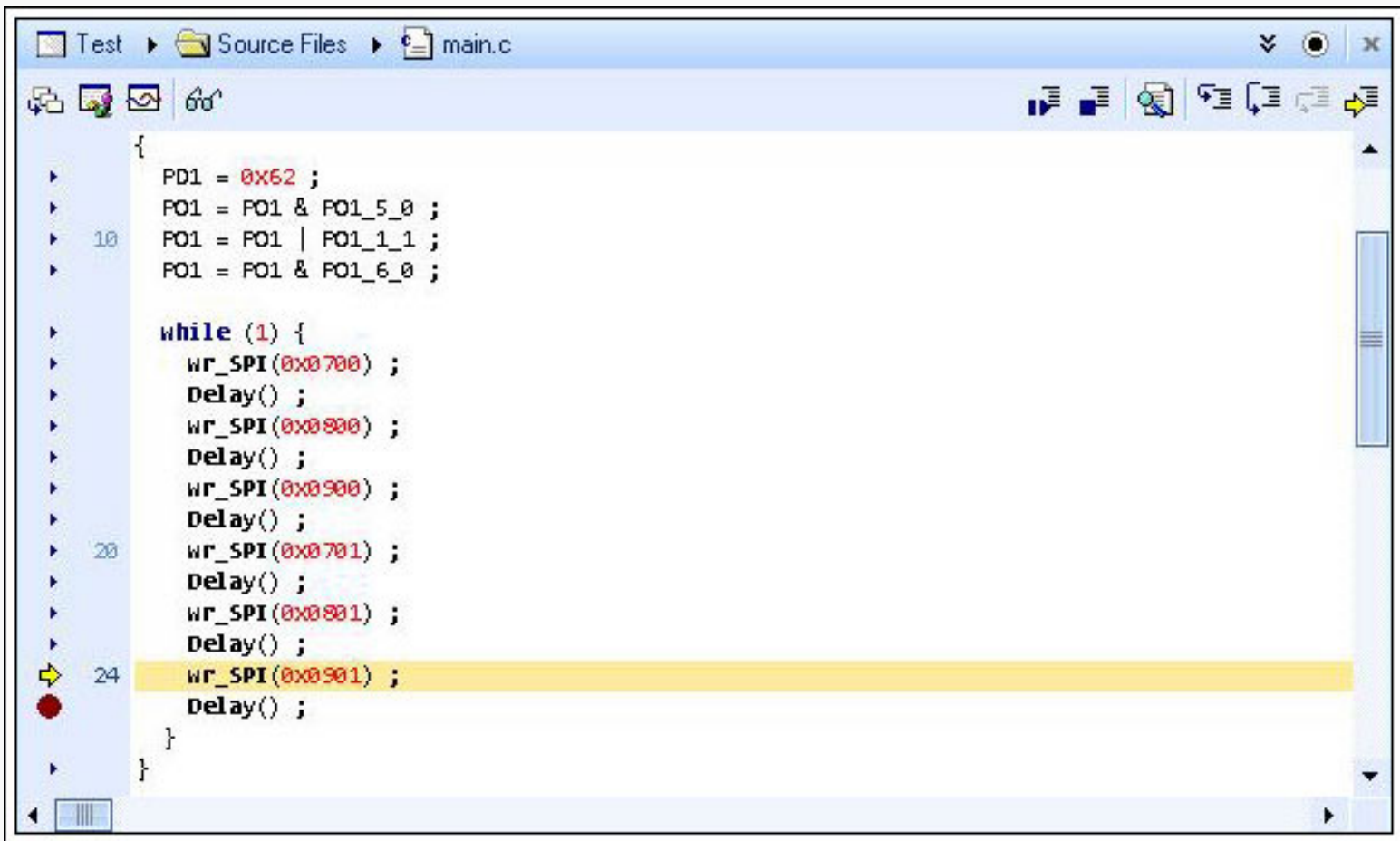


Figure 8. Breakpoint added.

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 (Figure 9). 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.



Figure 9. Step Into button.

It is also possible to 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.
- Stop debugging by using **Debug** → **Stop** and look at the Targets window to the right. Make sure that Maxim Serial JTAG Adapter Properties appears in bold font and look below at the Properties Window for information. If Maxim Serial JTAG Adapter Properties is not displayed, select it from the drop-down menu. You will see a list of properties and their settings in Figure 11. Use the scroll bar to see all available information. One property under the Connection heading is the Port Name. If you use a serial port other than the default COM1, here is where you can change the location.



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

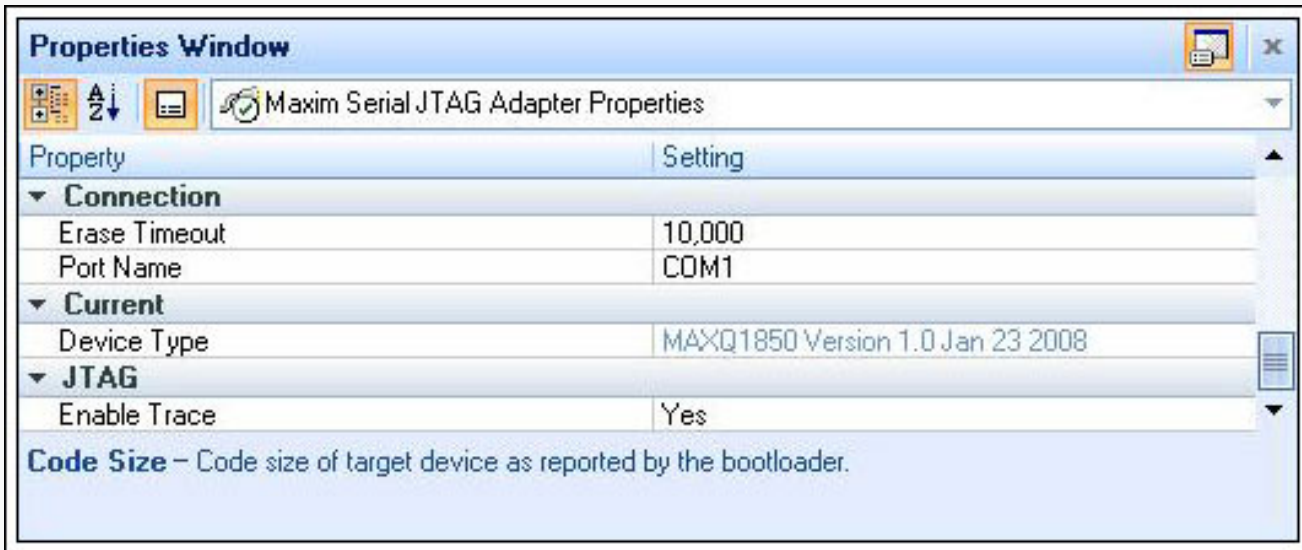


Figure 11. Properties Window.

For More Information

Software libraries and reference designs are currently in development by engineers at Maxim. Contact [microcontroller.support@maxim-ic.com](mailto:support@maxim-ic.com) for the latest information on available libraries and tools, or if you have any problems with this application note.

Appendix A. WalkLED main.c Source Code

```
#include <maxq1850.h>
#include <inmaxq.h>
#include "WalkLED.h"

void
main(void)
{
    PD1 = 0x62 ; // PD1 = In Out Out In   In In Out In
    PO1 = PO1 & PO1_5_0 ; // U10 Din (P1.5) = 0
    PO1 = PO1 | PO1_1_1 ; // U10 CS (P1.1) = 1
    PO1 = PO1 & PO1_6_0 ; // U10 SCLK (P1.6) = 0
}
```

```

while (1) {
    wr_SPI(0x0700) ;    // U10 P7 (nLED0) = 0
    Delay() ;
    wr_SPI(0x0800) ;    // U10 P8 (nLED1) = 0
    Delay() ;
    wr_SPI(0x0900) ;    // U10 P9 (nLED2) = 0
    Delay() ;
    wr_SPI(0x0701) ;    // U10 P7 (nLED0) = 1
    Delay() ;
    wr_SPI(0x0801) ;    // U10 P8 (nLED1) = 1
    Delay() ;
    wr_SPI(0x0901) ;    // U10 P9 (nLED2) = 1
    Delay() ;
}
}

void
wr_SPI(short SPI_Dat)    // Write data to SPI™ device U10
{
    int i;

    PO1 = PO1 & PO1_1_0 ; // U10 CS (P1.1) = 0

    for(i=0;i<16;i++) {    // Set Din for each data bit
        if (SPI_Dat < 0)
            PO1 = PO1 | PO1_5_1 ;    // Din = 1
        else
            PO1 = PO1 & PO1_5_0 ;    // Din = 0

        PO1 = PO1 | PO1_6_1 ;    // SCLK = 1
        PO1 = PO1 & PO1_6_0 ;    // SCLK = 0
        PO1 = PO1 & PO1_5_0 ;    // Din = 0
        SPI_Dat = SPI_Dat << 1 ;    // Shift in next bit
    }
    PO1 = PO1 | 0x02 ;    // CS = 1
}

void
Delay(void) {    // Delay to make LEDs visible
    int i = 0 ;
    for(i=0;i < 200000; i++) ;
}

```

SPI is a trademark of Motorola, Inc.

Application note 4312: www.maxim-ic.com/an4312

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest? [Sign up for EE-Mail™](#).

Related Parts

DS5250: [QuickView](#) -- [Abridged Data Sheet](#)

DS8007: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS8023: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS8024: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

DS8113: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

MAXQ1103: [QuickView](#) -- [Abridged Data Sheet](#)

MAXQ1850: [QuickView](#) -- [Abridged Data Sheet](#)

AN4312, AN 4312, APP4312, Appnote4312, Appnote 4312

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal