Keywords: 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, pay tv, set top box, STB

APPLICATION NOTE 4273

Getting Started with the DeepCover Secure Microcontroller (MAXQ1103) Evaluation Kit and the CrossWorks Compiler for the MAXQ30

Sep 22, 2008

Abstract: This application note describes how to create, build, and debug applications for the DeepCover® Secure Microcontroller (MAXQ1103). This application development uses the CrossWorks C compiler, available from Rowley Associates, for the MAXQ30 platform.

Introduction

The DeepCover® Secure Microcontroller (MAXQ1103) is designed for financial terminal applications. It runs 16-bit instructions and has a 32-bit data path. Instructions run in a single machine cycle, making the microcontroller a very high-performance RISC machine. The MAXQ1103 also has a number of important security features, including:

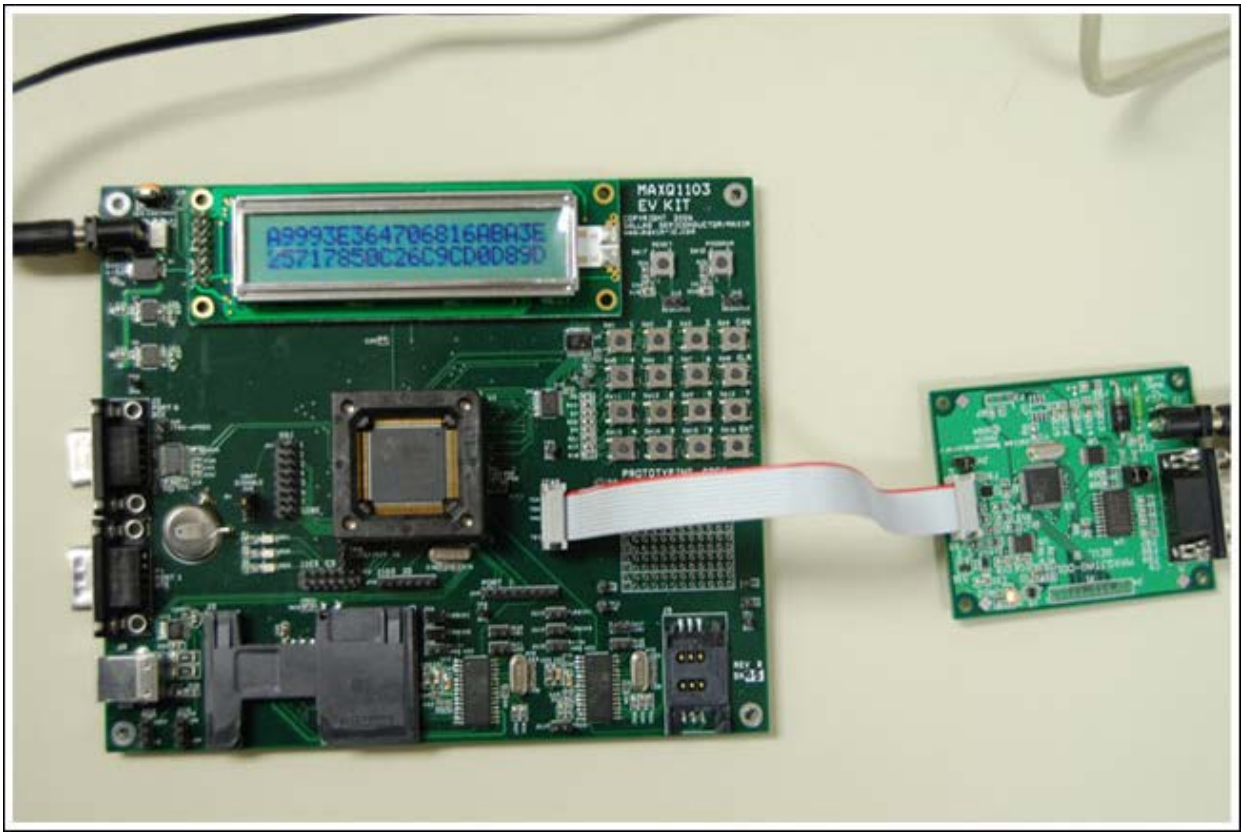
- Cryptographic accelerators supporting DES, 3DES, SHA-1, SHA-224, SHA-256, RSA, DSA, and ECDSA
- True hardware random number generator
- 1KB low-leakage battery-backed NVSRAM
- 7 tamper-detect inputs that tie to external circuitry
- Environmental sensors such as temperature and voltage out-of-range detectors

The MAXQ1103 evaluation (EV) kit provides an ideal platform for prototyping secure applications. The kit provides two serial ports, two smart card slots (one full size and one SIM card), a USB connector, an LCD screen, a 16-key keypad, and a prototyping area.

Setting Up the MAXQ1103 EV Kit

The MAXQ1103 EV kit is shown in **Figure 1**. The following hardware components are required to perform the work discussed in this application note:

1. MAXQ1103 EV kit board
2. JTAG board
3. JTAG cable (connects the MAXQ1103 EV kit board and JTAG board)
4. 9-pin serial cable
5. Regulated power supply (5V, ±5%, 300mA, center positive)



More detailed image (PDF, 14.4MB)
Figure 1. MAXQ1103 EV kit.

The MAXQ1103 EV kit and JTAG board each have jumpers to configure; see their respective data supplied with the EV kit for the full list of jumpers and functions. For this application note, configure the jumpers in the following way:

- On the MAXQ1103 EV kit board, close jumper JU1 and connect the two upper pins of JU5 (the two pins closest to the JU5 label). All other jumpers should be open for now. If any of JU6 through JU18 are closed, it is probably OK. These are configurations for smart card communication, which this application note will not cover.
- On the JTAG board, close JH1 and JH2, and open JH3.

Connect the JTAG cable between the JTAG board and the MAXQ1103 kit board. The red cable should connect to the side labeled pin1 and pin2 on the JTAG board, and to the TCK-GND side on the MAXQ1103 kit board.

Note that older MAXQ1103 EV kits may have sockets for the MAXQ1103 IC. If so, insert your MAXQ1103 into the socket with the IC's markings upside down (the lead-free indicator "+" should be in the upper right side).

Connect the 9-pin serial cable between your PC and the JTAG board. Do not connect it to the MAXQ1103 kit board. Connect the power supplies to the two boards.

Getting Started with the CrossWorks Compiler: Blinky

Instead of "Hello World," we begin by building a simple application that blinks an LED on the MAXQ1103 kit board.

The tool suite we use is CrossStudio, available from Rowley Associates. At this time the current version of the tool suite is CrossWorks for the MAXQ30 (version 2.0.0.2008063000.2293), and this was used to produce the screen shots included in this document. To check on the latest revision, go online to the Rowley Associates website or contact us through our Maxim Support Center.

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 "A C executable" from the Project Templates window (**Figure 2**). We will call our project BlinkyDemo and put it in the directory C:\work\maxq\maxq1103\blinky.

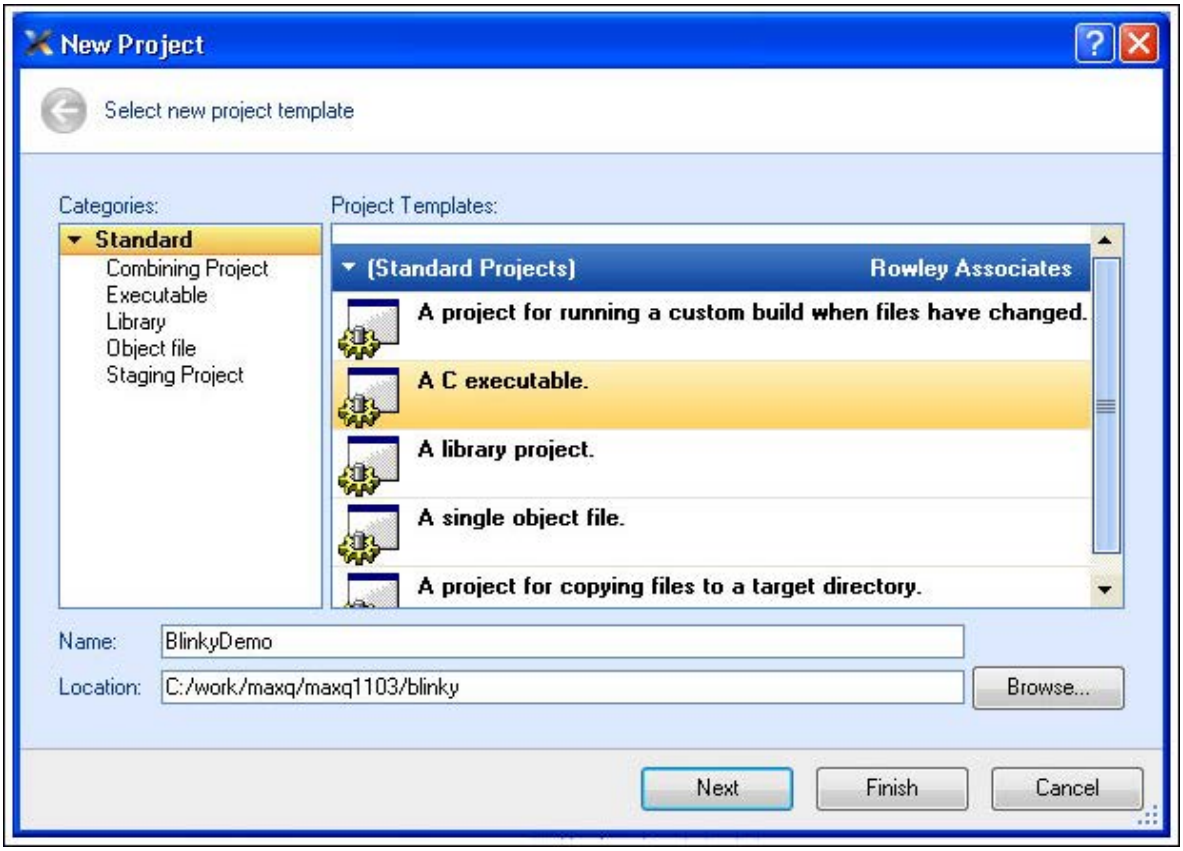


Figure 2. Select "A C executable" and fill in the project name and location.

Click Next to continue, and you will see a Project Properties pop-up box. The default values are good here, so click Finish and the project will be created. (You can also click Next to explore other options; this project uses all default values for these options.)

When the project is created, there will be a new project in the Project Explorer box (**Figure 3**), 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.

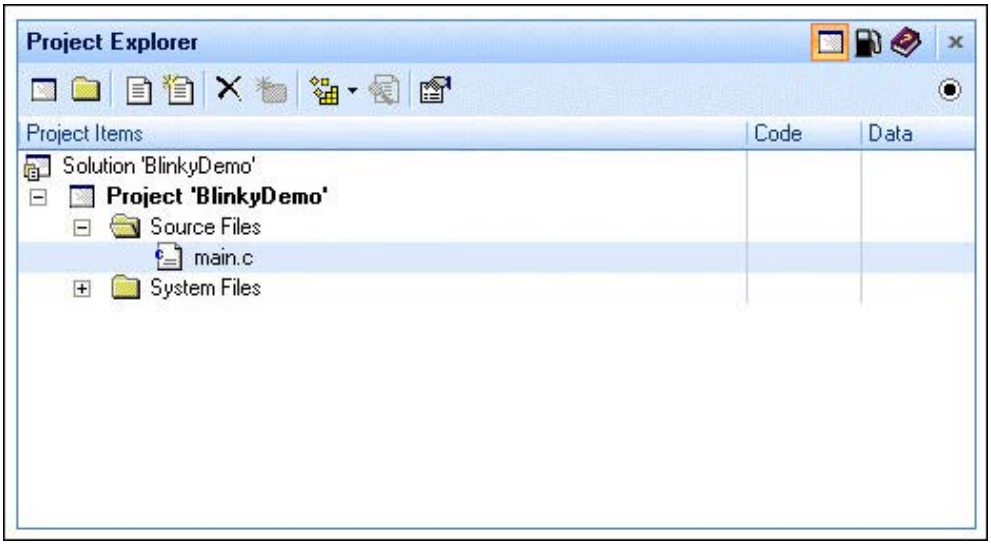


Figure 3. Project Explorer window.

The source code that is auto-generated is quite simple, and we only need to add a few lines to make our blinking application work. Copy the following application code (replacing everything currently in the `main.c` file).

```
#include <maxq1103.h>
#include <inmaxq.h>

void delayms(unsigned long count)
{
  unsigned int x;
  while (count > 0)
  {
    for (x=0;x<2500;x++)
    {
      __no_operation();
    }
    count--;
  }
}

void main(void)
{
  // set port 0 to all output
  PD0 = 0xff;
  while (1)
  {
    // toggle bits 0, 1, 7
    P00 = P00 ^ 0x83;
    delayms(500);
  }
}
```

When we run this application, we will expect to see LEDs DS1, DS2, and DS3 (located to the lower left of the MAXQ1103 on the kit board) blink on for about 0.5s and off for about 0.5s. Note that the "`delayms`" function is not exactly a millisecond, but it is close enough for the purposes of a blinky application.

Before we can run the demo, we need to build it. Select Build ⟶ Build BlinkyDemo. You can also press F7 to build. As long as everything built correctly, you will see the message Build Complete with check mark beside it in the Output window (**Figure 4**). If there are errors, make sure that you entered the code correctly.

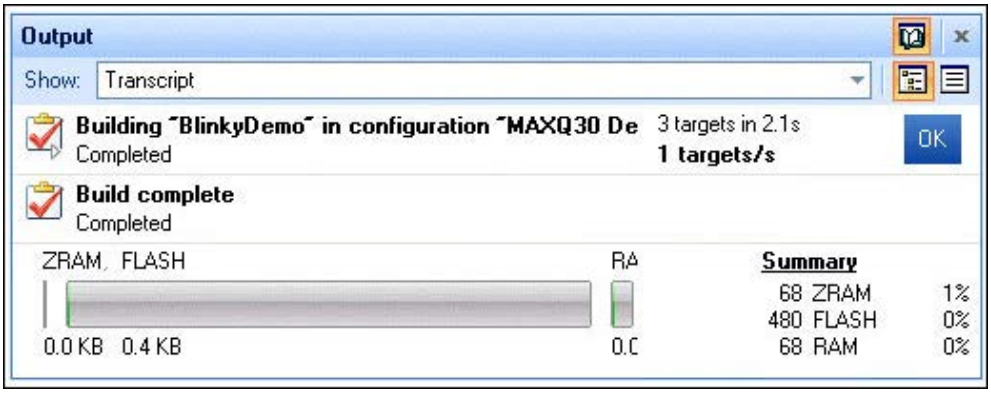


Figure 4. Output after project build.

To run the application, go to Debug ⟶ Step Over. You can also press F10 or click the icon in the toolbar with the downward arrow (**Figure 5**).

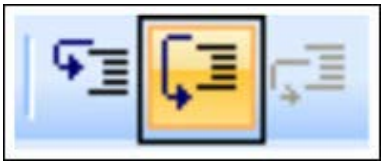


Figure 5. Step Over button.

CrossStudio will download the application to the MAXQ1103 through the JTAG board while status messages appear in the Output window. The application will begin to run and pause on the first line of code (denoted by the yellow arrow in the left margin). To run the application, click the button that looks like a "play" button (or go to Debug ⟶ Go). Now verify that the LEDs on the MAXQ1103 board are blinking. You can try to alter the application somewhat. Try to blink the LEDs in sequence, or try varying the amount of time that they are lit so the blinking becomes faster and slower over time.

Using CrossStudio to Debug an Application

Now we can explore some of the debugging capabilities of the MAXQ1103 and the CrossStudio tool. The MAXQ1103 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 MAXQ1103 also has a 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 MAXQ1103 microcontrollers that are deployed in sensitive applications.

We now return to the original Blinky application, and change the delay from 500 to 5 in the main function:

```
delayms(5);
```

Now build and run the application. Note that the LEDs appear to be continuously lit, rather than blinking on and off. This is what happened when the simple demo code was first written and run.

An initial question arose quickly: "Were the lights actually continuously on, or were they blinking too fast to be seen?" If the lights were continuously on, then considerably more review of schematics and pin assignments would be needed to ensure that everything was OK. If the LEDs were simply blinking fast (too fast for one to see the intervals clearly), then only the timings needed adjustment, a task that should probably have been expected. To answer this question, we now use CrossStudio's debug tools.

Press the Pause button (or go to Debug ⟶ Break). A yellow arrow will appear where the code stopped.

Chances are that the code stopped in the 'for' loop of the `delayms()` function (see **Figure 6**).

```
void delayms(unsigned long count)
{
  unsigned int x;
  while (count > 0)
  {
    for (x=0;x<2500;x++)
    {
      __no_operation();
    }
    count--;
  }
}
```

Figure 6. Code stopped execution in the `delayms()` function.

Look at the Locals window on the right (if it is not visible, go to Debug ⟶ Debug Windows ⟶ Locals). It should show the current values for the variables "x" and "count." Now press the Step Over button several times. You should see the value of x increase in the Locals window. (One could continue to press Step Over until the loop ends, but that would take "forever.")

The real question to answer now became apparent, "Were the lights blinking?" So, set a breakpoint on the line `delayms(5)` in function `main` by clicking the small triangle to the left on this line of code. It will turn into a red circle (**Figure 7**). Now run the application again (Debug ⟶ Go or the play button). The application will run to that point and halt. Now click Go several times and you should see the lights blink on and off as you click. This verifies that the lights were blinking, but too fast for our eyes to see it.

```
  PD0 = 0xff;
  while (1)
  {
    // toggle bits 0, 1, 7
    P00 = P00 ^ 0x83;
    delayms(5);
  }
}
```

Figure 7. Breakpoint added.

We now take this opportunity to explore more debug features. Press the Step Over button several times and you will execute three lines of code in sequence: `while(1)`, `P00 = P00 ^ 0x83`, and `delayms(5)`. You will also see the lights blink as you pass the `P00` line. Now press the Step Into button (**Figure 8**) while paused on the `delayms(5)` line and you will go into the `delayms()` function (rather than Step Over, which executes the whole function).

Figure 8. Step Into button.

You can also change variables (and registers) while running. Clear all the breakpoints (Debug → Breakpoints → Clear All Breakpoints) and click Go. Click Pause and you should be stopped in the middle of the `delayms()` function again. Note the values of "x" and "count." Now try setting x to 2499 (click on the value that it displays for x, and enter 2499 when highlighted). Now do several Step Over or Step Into actions and you should see the loop end and the value of "count" also decremented.

There are some other debugging features that may be interesting:

- Debug → Disassembly will show you a mix of C code and the generated assembly code. This will let you step through the assembly code instead of the C code, but also let you know where you are in your C code.
- Debug → Debug Windows → Call Stack will show you the functions that have been called to get your application to its current point. If you pause while in the `delayms()` function, you should see something like **Figure 9**.
- Stop debugging by Debug → Stop and look at the Targets window to the right. Make sure that Maxim Serial JTAG Adapter is bolded and look below at the configuration options. If you use a serial port other than the default COM1, here is the location to change that option.

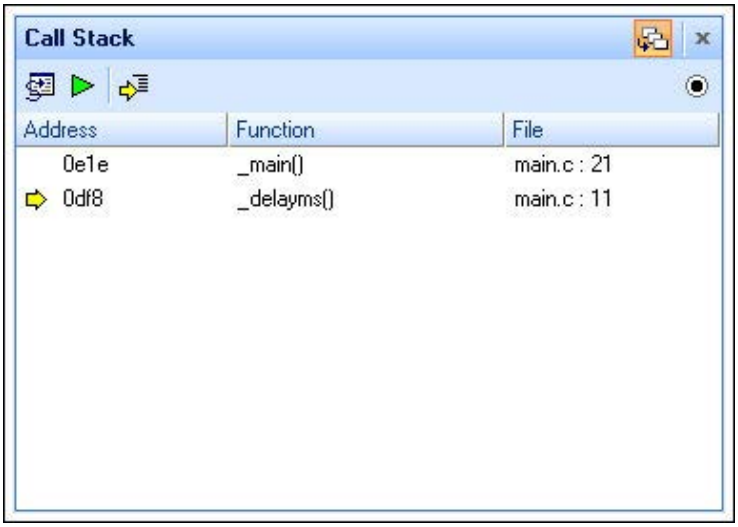


Figure 9. Call Stack while running in the `delayms()` function.

For More Information

Software libraries and reference designs are in progress by engineers at Maxim. Contact Tech Support for the latest information on available libraries and tools, or if you have any problems with this application note.

Related Parts		
DS5002	Secure Microprocessor Chip	Free Samples
DS5250	High-Speed Secure Microcontroller	
DS8007	Multiprotocol Dual 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 4273: http://www.maximintegrated.com/an4273
APPLICATION NOTE 4273, AN4273, AN 4273, APP4273, Appnote4273, Appnote 4273