Keywords: low-power microcontroller, IR carrier frequency generation, IR carrier modulation, IAR software, IAR Embedded Workbench

APPLICATION NOTE 5913

GETTING STARTED WITH THE MAXQ622 EVALUATION KIT

Abstract: This application note describes how to create, build, and debug applications targeted for the MAXQ622 low-power RISC microcontroller. The article uses the IAR Embedded Workbench toolset and C compiler available from IAR Systems.

Introduction

The MAXQ622 is a low-power microcontroller from Maxim Integrated Products. It is designed for battery-powered applications and offers low active-mode current (3.75mA typical at 12MHz). The microcontroller also features a highly efficient 16-bit single-cycle RISC processor core and flexible clocking schemes that help dynamically control performance and power consumption. The MAXQ622 is ideal for applications that require a large number of I/O pins and low power consumption.

The MAXQ622 includes a number of important features:

- Wide 1.7V to 3.6V operating voltage
- 128kB in-application programmable (IAP) flash
- 6kB data SRAM
- Two SPI interfaces, two USART interfaces, and one USB port
- 8kHz low-power nanoring wakeup timer
- IR carrier frequency generation and modulation
- Code scrambling prevents attackers from downloading software IP
- Memory protection keeps core libraries isolated and IP safe from third-party applications

The MAXQ622 evaluation (EV) kit provides a proven, reliable platform for developing low-power applications for the MAXQ622 processor. The kit includes: an IR transmitter and receiver, a USB port, 8 pushbuttons for user input, 4 LEDs for application usage, a prototyping area, and headers for accessing all the MAXQ622's I/O pins. Additionally, the provided jumpers allow convenient monitoring of the MAXQ622 processor's actual power consumption during operation.

Setting Up the MAXQ622 EV Kit

Figure 1 shows the MAXQ622 evaluation kit (EV kit) board. The EV kit contains the following hardware components:

1. MAXQ622 EV kit board
2. USB-to-JTAG/1-Wire$^{®}$ adapter
3. 10-pin JTAG ribbon cable
4. A-to-Mini-B USB cable

Use these components to implement and verify the demonstration program in this application note.

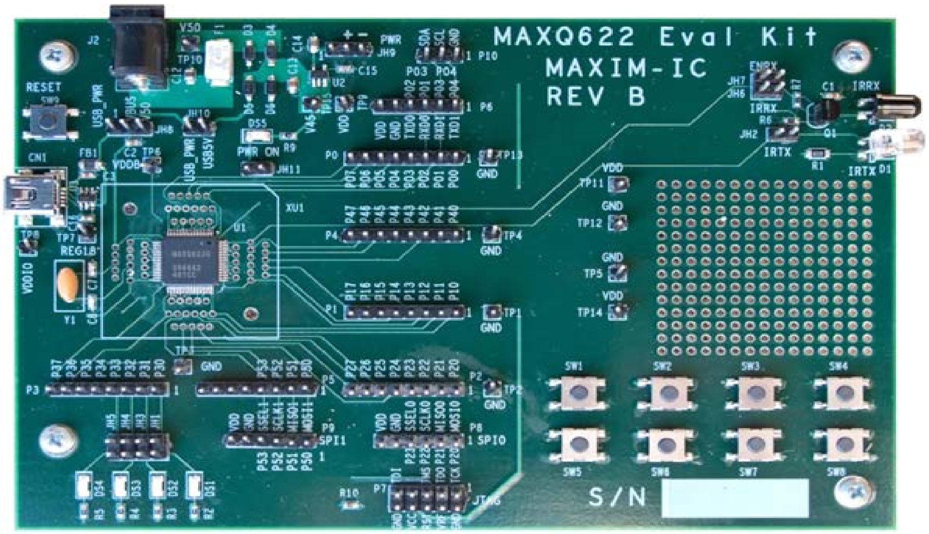


Figure 1. MAXQ622 EV kit

The MAXQ622 EV kit board has a number of jumpers to configure. For this application note, the jumpers should be configured to match the default settings, as shown in **Table 1**.

Table 1. Jumper Configuration for the MAXQ622 EV Kit Board

Jumper	Setting	Effect
JH1	Open	Disconnects the board's DS1 LED cathode from the MAXQ622's port pin P3.0.
	Closed*	Connects the board's DS1 LED cathode to the MAXQ622's port pin P3.0.
JH2	Open	Disconnects the board's D1 LED IR emitter from the MAXQ622's IRTX pin.
	Closed*	Connects the board's D1 LED IR emitter to the MAXQ622's IRTX pin.
JH3	Open	Disconnects the board's DS2 LED cathode from the MAXQ622's port pin P3.1.
	Closed*	Connects the board's DS2 LED cathode to the MAXQ622's port pin P3.1.
JH4	Open	Disconnects the board's DS3 LED cathode from the MAXQ622's port pin P3.2.
	Closed*	Connects the board's DS3 LED cathode to the MAXQ622's port pin P3.2.
JH5	Open	Disconnects the board's DS4 LED cathode from the MAXQ622's port pin P3.3.
	Closed*	Connects the board's DS4 LED cathode to the MAXQ622's port pin P3.3.
JH6	Open	Disconnects the board's D2 LED IR receiver amplified signal from the MAXQ622's IRRX pin.
	Closed*	Connects the board's D2 LED IR receiver amplified signal to the MAXQ622's IRRX pin.
JH7	Open	Disconnects the IR Rx enable from the MAXQ622's P0.7 pin.
	Closed*	Connects the IR Rx enable to the MAXQ622's P0.7 pin.
JH8	1-2 Closed*	Powers the MAXQ622's VBUS supply from the USB VBUS supply (connected at CN1).
	2-3 Closed	Powers the MAXQ622's VBUS supply from the +5.0V DC supply connected at J2.
JH9	1-2 Closed*	Power the MAXQ622's VDD supply from the output of the +3.3V fixed regulator.
	2-3 Closed	Powers the board through JH9.2 (power) and JH9.3 (ground).
JH10	Open	Disconnects the USB VBUS supply (from CN1) to the input to the +3.3V regulator.
	Closed*	Connects the USB VBUS supply (from CN1) to the input to the +3.3V regulator.
JH11	Open	Disables the DS5 power LED.
	Closed*	Enables the DS5 power LED.

*Default settings.

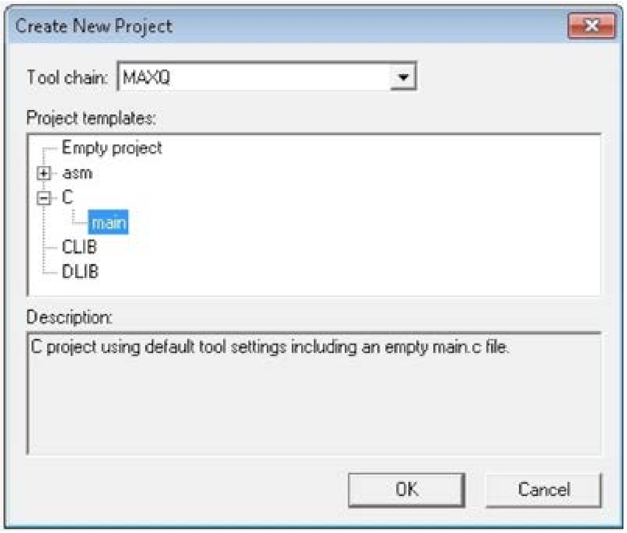
A USB-to-JTAG/1-Wire adapter has been included with the EV kit to program the MAXQ622. The installation guide for the adapter is available for download.

After installing the USB-to-JTAG/1-Wire adapter, connect the JTAG cable to the MAXQ622 EV kit board. The red stripe on the cable should connect to the side of the connector labeled pin 1 and pin 2 on the JTAG board and to the side of the connector labeled TCK-GND on the MAXQ622 EV kit board.

Getting Started with the IAR Embedded Workbench

The IAR Embedded Workbench® is the primary IDE used for coding in C with the MAXQ622. The latest version can be obtained from the our website. IAR™ offers both time-limited and size-limited licenses of the IDE for evaluation. Download and execute the installer. Follow the directions to install the software.

After installing the software, open the application and create a new project in the current workspace. Ensure that the MAXQ tool chain is selected from the drop-down list, and open a new C project with a generated **main.c** file (**Figure 2**).



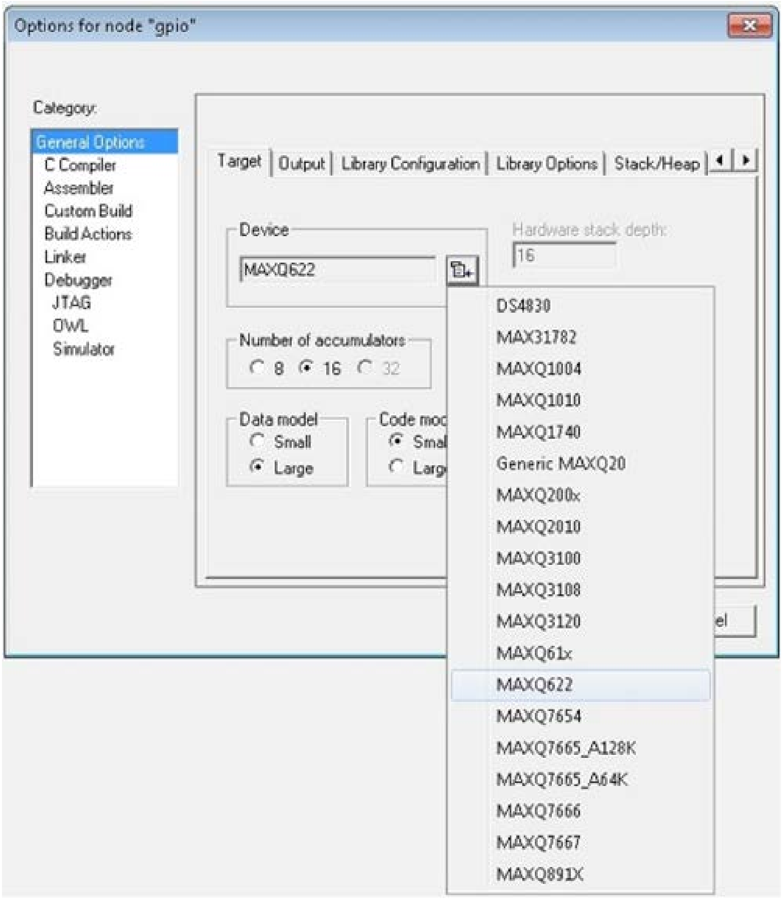
Tool chain: MAXQ

Description:

C project using default tool settings including an empty main.c file.

Figure 2. IAR new project wizard.

To configure the project to run on the MAXQ622, select **Options** from the **Project** menu or press **ALT+F7**. Select **MAXQ622** from the device drop-down menu and ensure that **CLIB** is selected under the **Library Configuration** tab (**Figure 3**).



Options for node "gpio"

Category:

Device: MAXQ622

Hardware stack depth: 16

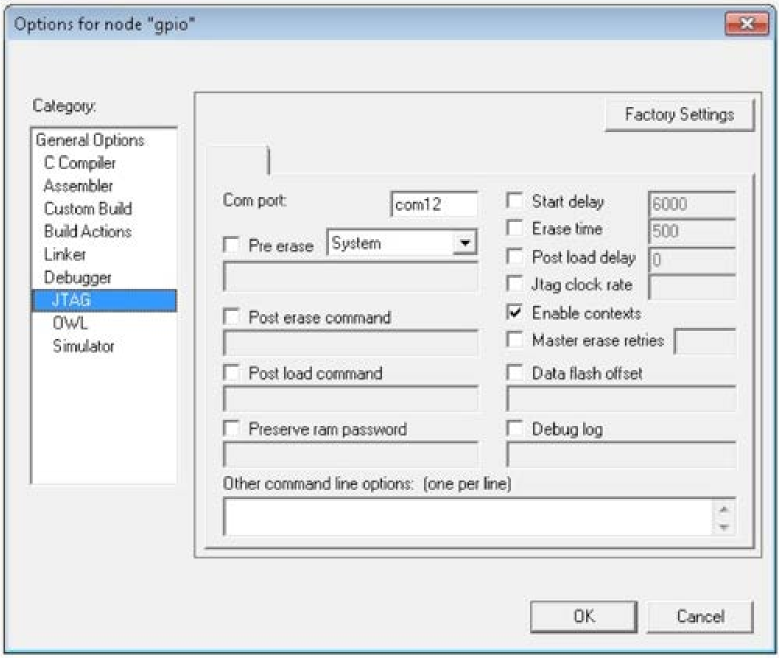
Number of accumulators: ○ 8 ● 16 ○ 32

Data model: ○ Small ● Large

Code moc: ● Smal ○ Larg

Figure 3. IAR Embedded Workbench project configuration.

Once the target device is selected, assign the com port that the JTAG adapter is using by inputting the string referencing the com port in the text field provided (**Figure 4**). Note: Use the System Device Manager to verify the com port setting.



Com port: com12

□ Pre erase System

□ Post erase command

□ Post load command

□ Preserve ram password

□ Start delay 6000

□ Erase time 500

□ Post load delay 0

□ Jtag clock rate

☑ Enable contexts

□ Master erase retries

□ Data flash offset

□ Debug log

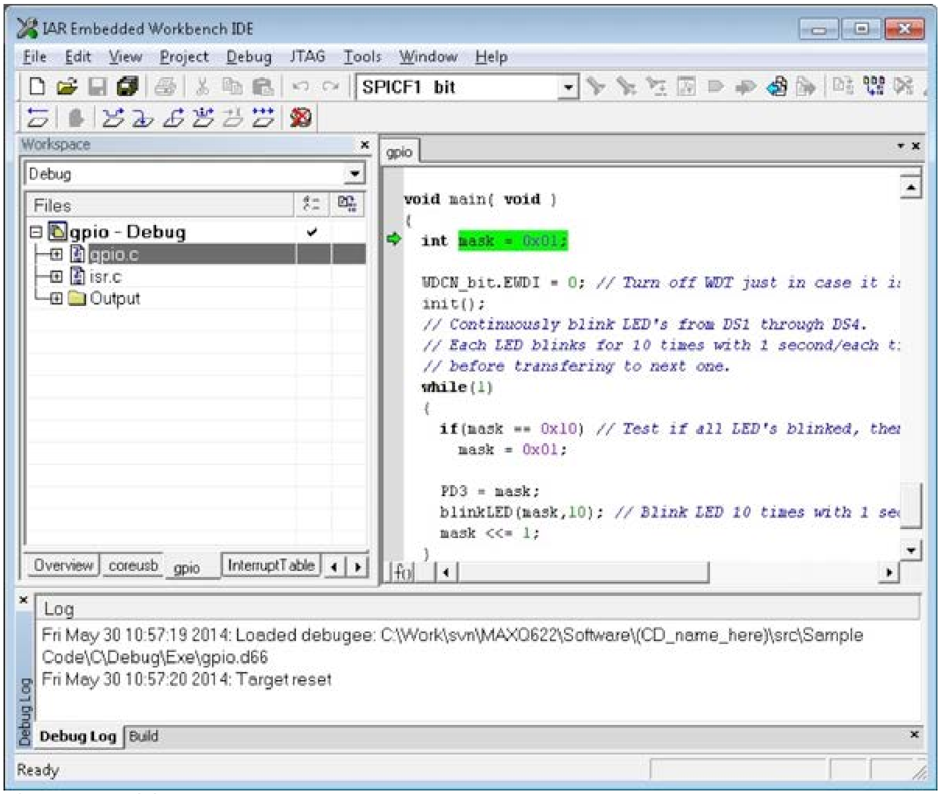
Other command line options: (one per line)

Figure 4. IAR JTAG settings.

The workspace **maxq622evkit.eww** includes sample projects that are viewable in the **Workspace** window. Configure each project in the workspace to match the MAXQ622 by following the above steps. The projects are now ready to be compiled, loaded, and debugged on the EV Kit.

Using IAR to Debug an Application

For this application note example, activate the **gpio** project from the **Workspace** tab and ensure that the project settings described above have been properly configured. To view the source code for the project, double-click on the file **gpio.c**. The file **isr.c** is not used in this project, but it contains the interrupt callback functions for other projects included with the EV kit. To begin debugging the project, click the **Debug** button or press **Ctrl + D**. This action automatically compiles the source code and loads it onto the MAXQ622. The default debug settings will automatically place a breakpoint at the main function (**Figure 5**).



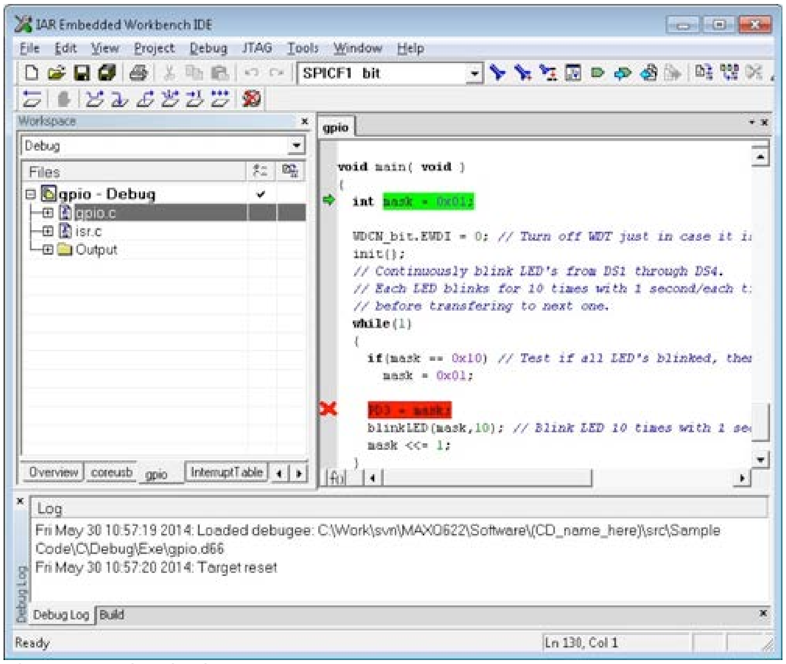
```
void main( void )
{
  int mask = 0x01;

  WDCN_bit.EWDI = 0; // Turn off WDT just in case it i:
  init();
  // Continuously blink LED's from DS1 through DS4.
  // Each LED blinks for 10 times with 1 second/each t:
  // before transfering to next one.
  while(1)
  {
    if(mask == 0x10) // Test if all LED's blinked, the
      mask = 0x01;

    PD3 = mask;
    blinkLED(mask,10); // Blink LED 10 times with 1 se
    mask <<= 1;
  }
```

Log

```
Fri May 30 10:57:19 2014: Loaded debugee: C:\Work\svn\MAXQ622\Software\(CD_name_here)\src\Sample
Code\C\Debug\Exe\gpio.d66
Fri May 30 10:57:20 2014: Target reset
```

Figure 5. IAR debug.

From here, navigate through the program using the Debug toolbar functions. To set more breakpoints throughout the code, double-click in the margin to the left of the line to place a break in the program (**Figure 6**).


IAR Embedded Workbench IDE
File Edit View Project Debug JTAG Tools Window Help
SPICF1 bit
Workspace
Debug
Files
gpio - Debug
gpio.c
isr.c
Output
Overview coreusb gpio InterruptTable
gpio
Debug Log Build
Ready
Ln 130, Col 1

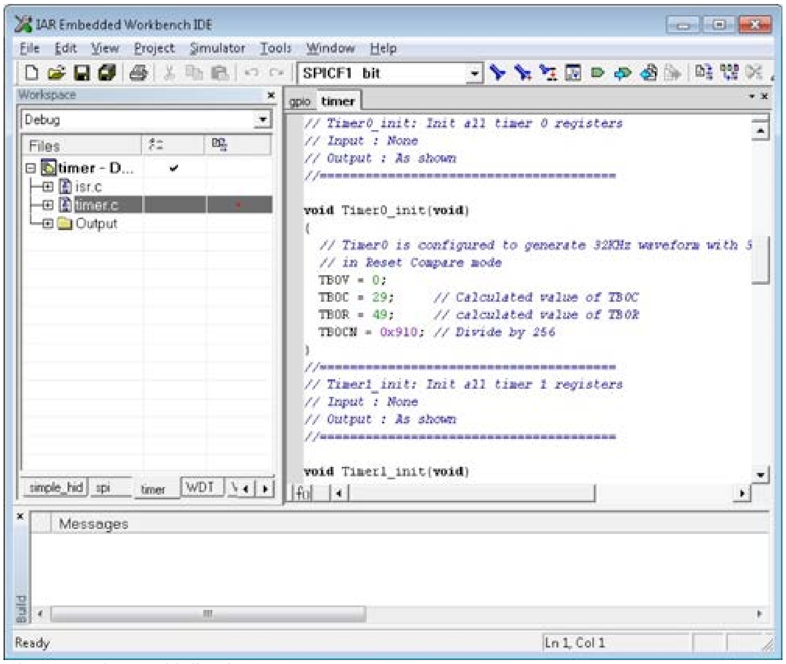
```
void main( void )
{
  int mask = 0x01;

  WDCN_bit.EWDI = 0; // Turn off WDT just in case it i
  init();
  // Continuously blink LED's from DS1 through DS4.
  // Each LED blinks for 10 times with 1 second/each t
  // before transfering to next one.
  while(1)
  {
    if(mask == 0x10) // Test if all LED's blinked, the
      mask = 0x01;

    PD3 = mask;
    blinkLED(mask,10); // Blink LED 10 times with 1 se
    mask <<= 1;
  }
```

```
Log
Fri May 30 10:57:19 2014: Loaded debugee: C:\Work\svn\MAXQ622\Software\(CD_name_here)\src\Sample
Code\C\Debug\Exe\gpio.d66
Fri May 30 10:57:20 2014: Target reset
```

Figure 6. IAR breakpoints.

The **gpio** example uses a timer on the MAXQ622 to blink 4 LEDs. The first four pins are wired to these LEDs and can be manipulated by the software. By shifting the mask variable, the position of the high bit will move through the mask to the bit for each LED (Figure 6). The PD3 register controls the direction of the port 3 pins. Setting these bits to 1 configures them for output, allows the voltage to be raised high, and turns on the LEDs.

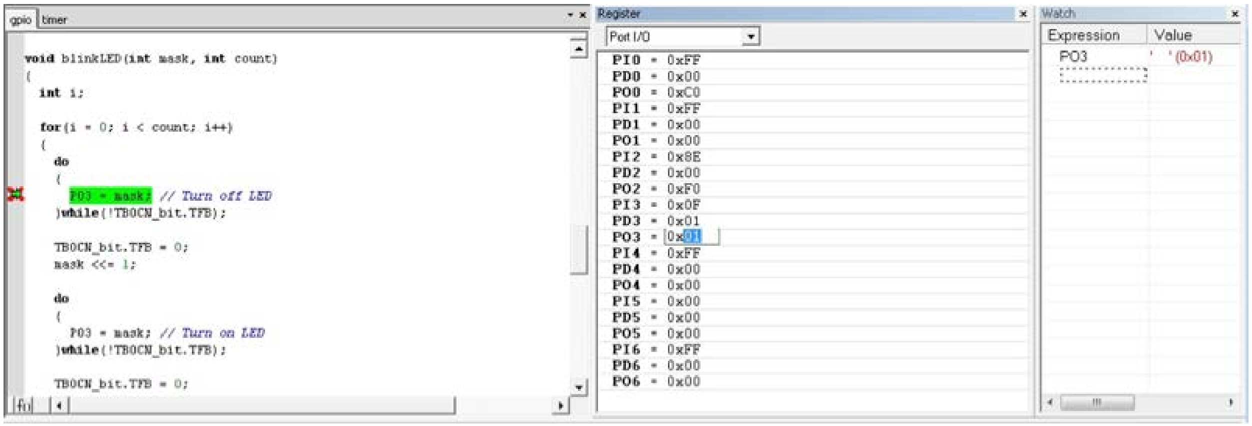
A timer in the MAXQ622 is then used to control the blink rate of the LEDs. The timer initializes to count at a constant rate (Sysclk/256) from 0 up to 0xB71B. The default Sysclk is 12MHz, making the timer raise its flag every second (**Figure 7**).



```
// Timer0_init: Init all timer 0 registers
// Input : None
// Output : As shown
//========================================

void Timer0_init(void)
{
  // Timer0 is configured to generate 32KHz waveform with 5
  // in Reset Compare mode
  TBOV = 0;
  TBOC = 29;      // Calculated value of TBOC
  TBOR = 49;      // calculated value of TBOR
  TBOCN = 0x910; // Divide by 256
}
//========================================
// Timer1_init: Init all timer 1 registers
// Input : None
// Output : As shown
//========================================

void Timer1_init(void)
```

Figure 7. Timer Initialization.

The IAR software allows the user to view variable and register values while debugging. In order to look at a register or variable, right-click on the expression and select **Add to Watch**. When a breakpoint halts execution or the **Break** button is pressed, these values are updated and can be changed by the user (**Figure 8**). Also, view registers by selecting the **View** menu and clicking **Register**.



```
void blinkLED(int mask, int count)
{
  int i;

  for(i = 0; i < count; i++)
  {
    do
    {
      PO3 = mask; // Turn off LED
    }while(!TBOCN_bit.TFB);

    TBOCN_bit.TFB = 0;
    mask <<= 1;

    do
    {
      PO3 = mask; // Turn on LED
    }while(!TBOCN_bit.TFB);

    TBOCN_bit.TFB = 0;
```

```
PI0 = 0xFF
PD0 = 0x00
PO0 = 0xC0
PI1 = 0xFF
PD1 = 0x00
PO1 = 0x00
PI2 = 0x8E
PD2 = 0x00
PO2 = 0xF0
PI3 = 0x0F
PD3 = 0x01
PO3 = 0x01
PI4 = 0xFF
PD4 = 0x00
PO4 = 0x00
PI5 = 0x00
PD5 = 0x00
PO5 = 0x00
PI6 = 0xFF
PD6 = 0x00
PO6 = 0x00
```

Expression	Value
PO3	' ' (0x01)

Figure 8. blinkLED() function/IAR Register and Watch.

Related Parts		
MAXQ622	16-Bit Microcontrollers with Infrared Module and Optional USB	Free Samples

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

Application Note 5913: http://www.maximintegrated.com/en/an5913
APPLICATION NOTE 5913, AN5913, AN 5913, APP5913, Appnote5913, Appnote 5913