Getting Started with the DRFG4618 Hardware Platform

by Alex Milenkovich, milenkovic@computer.org

Objectives: This tutorial will help you get started with the MSP30 IAR Assembly program development on the hardware platform DRFG4618. You will learn the following topics:

- Assembly programming
- Creating an application project using assembly programs
- Debugging using the IAR C-SPY® Debugger (Simulator)
- Program downloading on a real platform (Softbaugh DRFG 4618).

Note: It is required that students have completed the tutorial *Getting Starting With MSP430 IAR Embedded Workbench* before starting with this one.

1. Blink a LED Project: Problem Statement

This section defines the problem that will be solved by the "Blink a LED" application.

Your task is to write an assembly program that will repeatedly blink the LED1 on the Softbaugh's DRFG4618 board every second, i.e., the LED1 will be on and off for about 0.5 seconds each.

Step 1: Analyze the assignment.
In order to better understand the assignment we will first study schematics of the Softbaugh's DRFG4618 board. This board includes a TI's MSP430 microcontroller (MSP430FG4618), wireless transceiver CC2420, serial RS232 port, 4 leds (LED1-LED4), 4 switches (SW1-SW4), a potentiometer, and several extension slots that allow an easy access to all microcontroller ports. Detailed schematic of the board is provided in the following document: http://www.ece.uah.edu/~milenka/cpe323-08F/docs/DRFG4618_Schematic.pdf.

Open the DRFG4618_Schematic.pdf file and go to page 6. Please locate LED1 output port. What microcontroller port pins are connected to LED1?
Go to page 5 of the schematic and see how the LED1 port is actually connected to a physical led (a diode through a resistor).

Step 2. Develop a plan.
From configuration it is clear that if we want LED1 on, we should provide a logical '1' at the output port of the microcontroller (port P1.0), and a logical '0,' if we want LED1 to be off. We could take several approaches to solving this problem. The simplest one is to toggle the port P1.0 and have 0.5 seconds delay in software.

After initializing the microcontroller, our program will spend all its time in an infinite loop (LED1 should be repeatedly blinked). Inside a loop we will toggle the port P1.0 and then wait for approximately 0.5 seconds. The port P1.0 toggling can be done using an XOR operation of the current value of the port (P1OUT) and the constant 0x01, i.e.,

(P1OUT=P1OUT xor 0x01). Software delay of 0.5 seconds can be implemented using an empty loop with a certain number of iterations.

2. Blink a LED Project: Assembly Code

Figure 1 shows the assembly code of the blink application. Here is a short description of the assembly code for this application.

1. The comments in a single line start with a column character (;). Multi-line comments can use C-style /* comment */ notation.
2. #include <msp430xG46x.h>; This is a C-style pre-processor directive that specifies a header file to be included in the source. The header file includes all macro definitions, for example, special function register addresses (`WDTCTL`), and control bits (`WDTPW+WDTHOLD`).
3. RSEG is a segment control assembler directive that controls how code and data are located in memory. RSEG is used to mark the beginning of a relocatable code or data segment. CODE and DATA are recognized segment types that are resolved by the linker. The IAR XLINK linker can recognize any other type of segment (e.g., CSTACK for code stack).
4. How does my program execute on the MSP430. Upon powering-up the MSP430 control logic always generates a RESET interrupt request (it is the highest priority interrupt request). The value stored at the address 0xFFFE (the last word in the 64KB address space) is reserved to keep the starting address of the reset handler (interrupt service routine), and the first thing that the microcontroller does is to fetch the content from this address and put it in the program counter (PC). Thus, the starting address of our program should be stored at location 0xFFFE. The IAR linker is responsible for this step. Please notice that the symbolic name RESET (the starting address of our program) is used to load the reset vector in the interrupt vector table.
5. First instruction initializes the stack pointer register.
6. The move instruction at the StopWDT label, sets certain control bits of the watchdog timer to disable it. The watchdog timer by default will be active generating interrupt requests periodically. As this functionality is not needed in our program, we simply need to disable it.
7. Parallel ports in MSP430 microcontroller can be configured as input or output. A control register PxDIR determines whether the port x is input or output (we can configure each individual port pin). Our program drives the port pin P1.0, so it should be configured as the output. The LSB bit of the P1DIR is set to 1, while other bits are unchanged (0 by default).
8. The main loop starts at Mainloop label. The xor instruction performs toggling operation as explained above.
9. The code starting at label Wait implements software delay. To exactly calculate the software delay we need to know instruction execution time and the clock cycle time. The register R15 is loaded with 50,000. The dec.w instruciton takes 1 clock cycle to execute, and jnz L1 takes 2 clock cycles to execute (note: this can be determined by using the Simulator and the Register View CYCLECOUNTER field). The total time per iteration is 3 clock cycles. Determining clock cycle time requires in-depth understanding of the FLL-Clock module of the MSP430 is beyond the scope of this tutorial. We note that the processor clock frequency is approximately 1MHz for our configuration. The total delay is thus 50,000*3*1us=150,000us. Obviously, to meet our requirements we need to increase the number of instructions in the software delay loop (to be 10). We can do it by adding nop instructions.

10. The final program code that satisfies the requirements is shown in

```
;*******************************************************************************
;   MSP430xG46x Demo - Software Toggle P1.0
;
;   Description: Toggle P1.0 by xor'ing P1.0 inside of a software loop.
;   ACLK = 32.768kHz, MCLK = SMCLK = default DCO
;
;                MSP430xG461x
;             -----------------
;         /|\|                 |
;          | |                 |
;          --|RST              |
;            |                 |
;            |             P1.0|-->LED
;
;   Alex Milenkovich, milenkovic@computer.org
;
;*******************************************************************************
#include  <msp430xG46x.h>
;-------------------------------------------------------------------------------
            RSEG    CSTACK                  ; Define stack segment
;-------------------------------------------------------------------------------
            RSEG    CODE                    ; Assemble to Flash memory
;-------------------------------------------------------------------------------
RESET       mov.w   #SFE(CSTACK),SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT
SetupP5     bis.b   #001h,&P1DIR            ; P1.0 output
                                            ;
Mainloop    xor.b   #001h,&P1OUT            ; Toggle P1.0
Wait        mov.w   #050000,R15             ; Delay to R15
L1          dec.w   R15                     ; Decrement R15
            jnz     L1                      ; Delay over?
            jmp     Mainloop                ; Again
                                            ;
;-------------------------------------------------------------------------------
            COMMON  INTVEC                  ; Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     RESET_VECTOR            ; MSP430 RESET Vector
            DW      RESET                   ;
            END
;*******************************************************************************
```

Figure 1. MSP430 Assembly Code for Blinking a LED.

```
;*******************************************************************************
;   MSP430xG46x Demo - Software Toggle P1.0
;
;   Description: Toggle P1.0 by xor'ing P1.0 inside of a software loop.
;   ACLK = 32.768kHz, MCLK = SMCLK = default DCO
;
;                MSP430xG461x
;             -----------------
;         /|\|                 |
;          | |                 |
;          --|RST              |
;            |                 |
;            |             P1.0|-->LED
;
;   Alex Milenkovich, milenkovic@computer.org
;
;*******************************************************************************
#include  <msp430xG46x.h>
;-------------------------------------------------------------------------------
            RSEG    CSTACK                  ; Define stack segment
;-------------------------------------------------------------------------------
            RSEG    CODE                    ; Assemble to Flash memory
;-------------------------------------------------------------------------------
RESET       mov.w   #SFE(CSTACK),SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop WDT
SetupP5     bis.b   #001h,&P1DIR            ; P1.0 output
                                            ;
Mainloop    xor.b   #001h,&P1OUT            ; Toggle P1.0
Wait        mov.w   #050000,R15             ; Delay to R15
L1          dec.w   R15                     ; Decrement R15
            nop
            nop
            nop
            nop
            nop
            nop
            nop
            jnz     L1                      ; Delay over?
            jmp     Mainloop                ; Again
                                            ;
;-------------------------------------------------------------------------------
            COMMON  INTVEC                  ; Interrupt Vectors
;-------------------------------------------------------------------------------
            ORG     RESET_VECTOR            ; MSP430 RESET Vector
            DW      RESET                   ;
            END
;*******************************************************************************
```

Figure 2. MSP430 Assembly Code for Blinking a LED every 1s.

3. Creating an Application Project

Please consult the tutorial on how to create a new workspace and project.

Step 1. Choose Project>Create New Project. Select Empty project and save it as blink_asm.

Step 2. Add the source file msp430xG46x_blink.s43 to the project. Choose Project>Add Files.

Step 3. Choose Project>Options and set all categories as explained in the Getting Starting With MSP430 IAR Embedded Workbench tutorial.

Step 4. Select the source file msp430xG46x_blink.s43 and compile it. (Choose Project>Compile). Click on the list file msp430xG46x_blink.lst to see report generated by the compiler.

Step 5. Setting the linker options. In Project>Options, select Linker category. Check the box for "Override default program entry" and select "Defined by application." This step notifies linker where to find the beginning of the program.

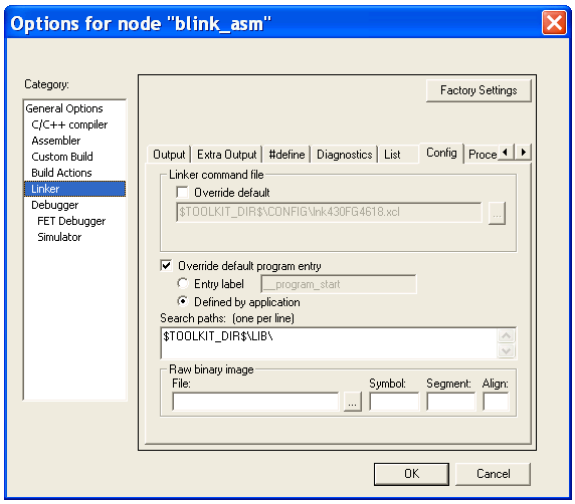

Figure 3. Linker category.

Step 6. Link the program. Choose Project>Make. The application is now ready for debugging and for download to the Softbaugh board. Click on blink_asm.map to see report from the linker.

4. Program Simulation and Debugging

In this section we will discuss how to simulate program execution. will simulate program execution.

Step 1: Go to Project>Options and select Debugger category. In the Driver box, choose Simulator (Figure 4). Click OK.

Step 2: Choose Project>Debug. Ignore warning. You will see the EW display as shown in Figure 5.

Step 3: Step through the program. Observe the Disassembly, Register View, and Memory View windows. Answer the following questions.
What is the starting address of the program?
How many clock cycles does each instruction take to execute?

Observe the contents of memory location and registers as you step through the program. What is the content of the memory location at the address 0xFFFE?

What are addresses of the special-purpose registers P1DIR and P1OUT? Monitor the contents of these locations as you walk through your program. Set breakpoints to move easier through your program.

Step 4: Choose Stop Debugging.

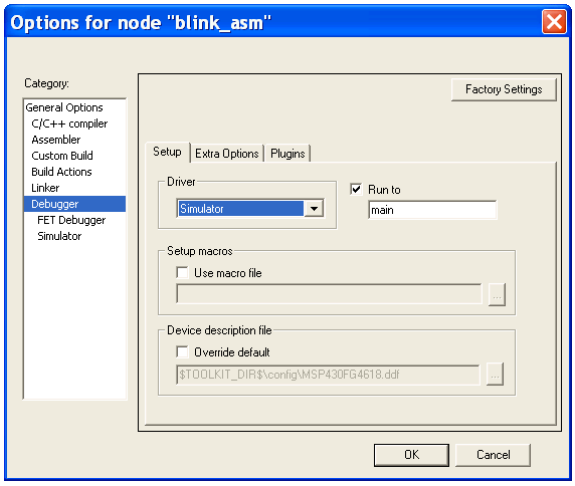

Figure 4. Debugger category.

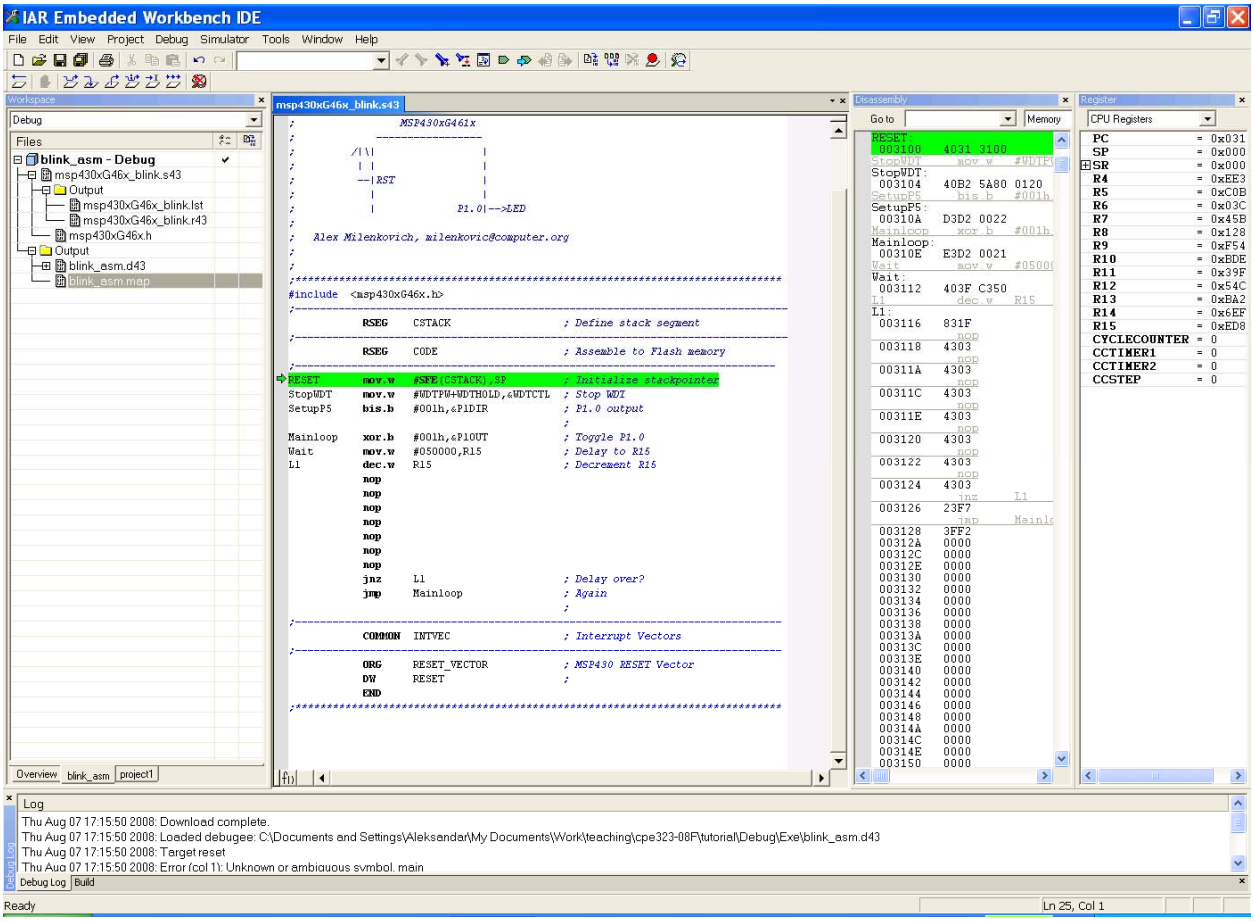

Figure 5. IAR Simultor.

5. Program Download

Step 1: Choose Project>Options, Debugger category. Select FET Debugger (Figure 6) and click OK.

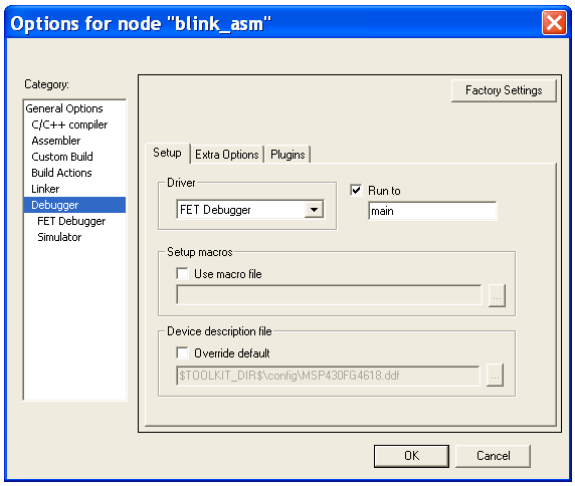

Figure 6. Debugger category.

Step 2. Start debugger. The executable is downloaded to the board through JTAG programmer.

Step 3. Choose Debug>Go. Observe the LED1 on the board.

Step 4. Stop the debugger. Experiment on your own.

6. Assignments

1. Write an assembly program that periodically toggles (LED1-LED4) every 2 seconds (all leds are on for 1 second, and off for 1 second). Demonstrate your program on the DRFG4618 platform.
2. Write a C program that periodically turns-on the leds (LED1-LED4) in sequence as shown below. Demonstrate your program on the DRFG4618 platform.