Gökhan Seçinti
Ahmet Arış
Doğan Altan

Istanbul Technical University

Department of Computer Engineering

BLG 351E – Microcomputer Laboratory

Experiments Booklet

Fall 2017

Version 1.0.0

Contents

Introduction to CCS

Creating New Project

When Code Composer Studio starts up, select *Project - New CCS Project* from the menu as shown in Figure .

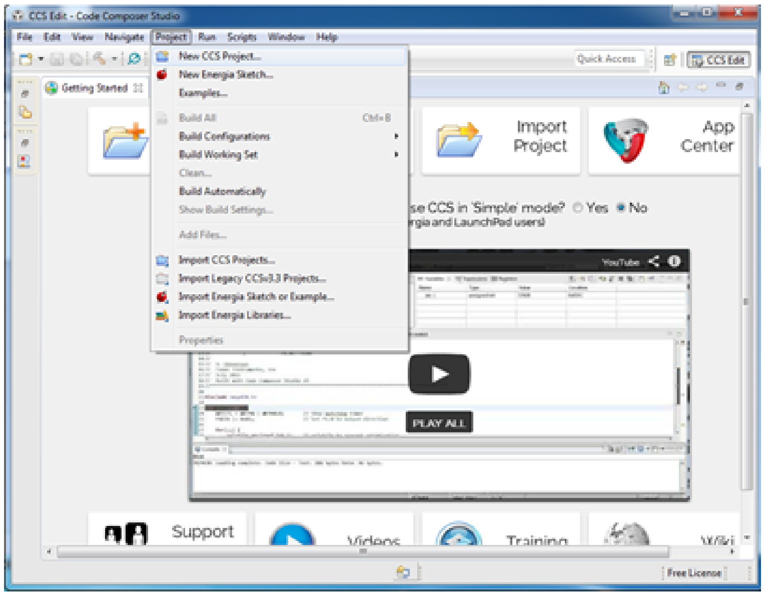
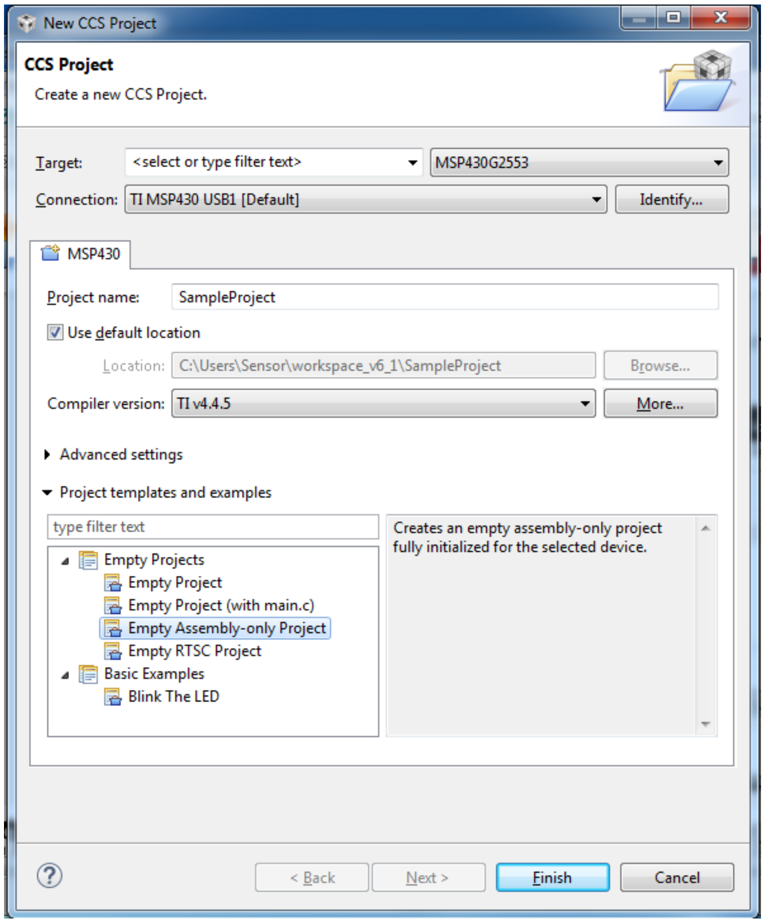


Fig. 0.1. New Project Creation Menu

On the following window which is shown in Figure , Target has to be selected as MSP430G2553. Give a proper name to the project and select the newest compiler version in the *compiler version* drop down list. Then from *the project templates and examples*, select the *Empty Assembly-only Project.*

Code Composer Studio will create **main.asm** source file for your assembly program. The structure of the "main.asm" file is shown below. During the experiments, you will place your assembly code to the section of the file which is commented as *;Main loop here* and the leave the rest of the file as it is.



New CCS Project

CCS Project

Create a new CCS Project.

Target: <select or type filter text> MSP430G2553

Connection: TI MSP430 USB1 [Default]

MSP430

Project name: SampleProject

☑ Use default location

Location: C:\Users\Sensor\workspace_v6_1\SampleProject

Compiler version: TI v4.4.5

Advanced settings

Project templates and examples

- Empty Projects
  - Empty Project
  - Empty Project (with main.c)
  - Empty Assembly-only Project
  - Empty RTSC Project
- Basic Examples
  - Blink The LED

Creates an empty assembly-only project fully initialized for the selected device.

Fig. 0.2. New Project Window

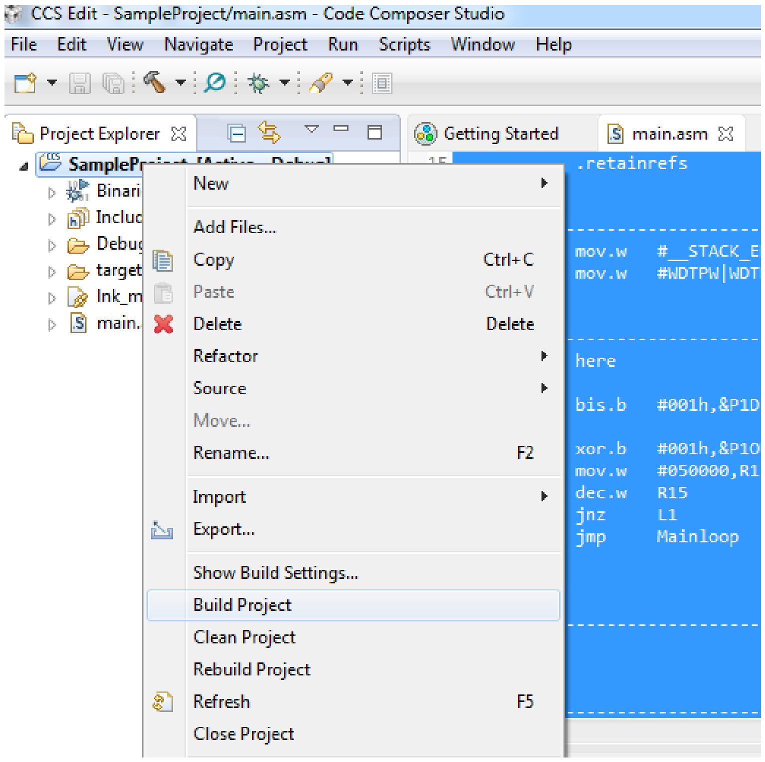
```
;-------------------------------------------
; MSP430 Assembler Code Template
;  for use with TI Code Composer Studio
;-------------------------------------------
; Include device header file
        .cdecls C,LIST,"msp430.h"
;-------------------------------------------
 ; Export program entry-point to
 ; make it known to linker.
        .def    RESET
        .def    ISR
;-------------------------------------------
        .text           ; Assemble into program memory.
        .retain         ; Override ELF conditional linking
                        ; and retain current section.
        .retainrefs     ; And retain any sections that have
                        ; references to current section.
;-------------------------------------------
; Initialize stackpointer
RESET       mov.w   #__STACK_END,SP

; Stop watchdog timer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL

;-------------------------------------------
; Main loop here
;-------------------------------------------

;-------------------------------------------
; Stack Pointer definition
;-------------------------------------------
        .global __STACK_END
        .sect   .stack

;-------------------------------------------
; Interrupt Vectors
;-------------------------------------------
        .sect   ".reset"       ; MSP430 RESET Vector
        .short  RESET
        .sect  ".int03"
        .short  ISR
```

Running the Project

Building Your Program

Before loading your program to the board, you first need to build your program by right clicking on the project and selecting Build Project from the menu as shown in Figure .

During the build process, compiler may see portions of your code unnecessary and remove them or try to optimize your code by taking other actions. In order to see the results of your code without any compiler modifications, you need to deactivate the compiler optimization. You can do this by right clicking on your project and selecting the Properties. In the Properties Window,



```
.retainrefs

mov.w   #__STACK_E
mov.w   #WDTPW|WDT

here

bis.b   #001h,&P1D

xor.b   #001h,&P1O
mov.w   #050000,R1
dec.w   R15
jnz     L1
jmp     Mainloop
```

Fig. 0.3. Building the project

from the Build tab, select MSP430 Compiler and then Optimization section. In optimization section, select optimization level as **off** and click ok.

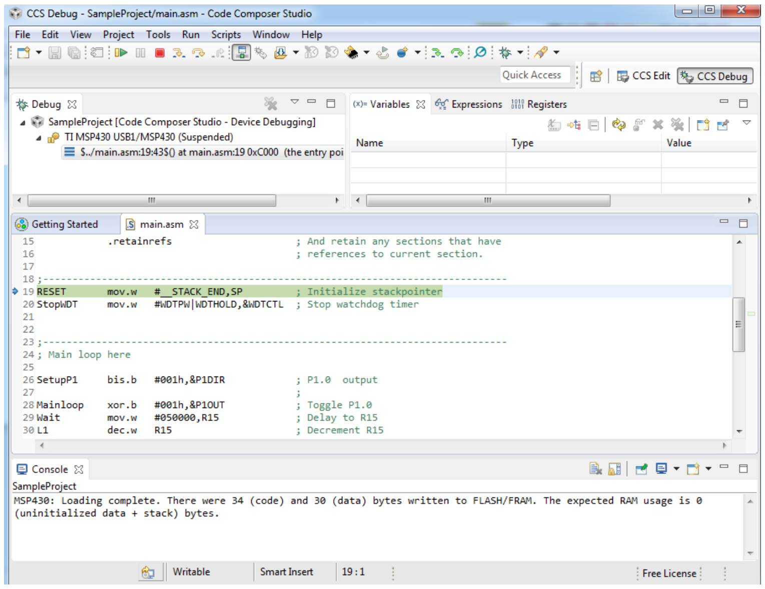
Debugging and Loading

If build process does not complain about any errors, then your program is ready to be loaded to the board. If your board is not connected to the PC via USB connection, you should connect it now. In order to load your program and debug it, press F11 or select Run - Debug from the menu. Code Composer Studio will load your program to the board and you will see the Debugging view right now as shown in the Figure .

When the program is loaded to the board, then we have chance run the assembly code step by step by either pressing F5 on the keyboard or using the debugging menu (i.e., step into, step over) as shown in Figure .

You can use the step into and step over to execute the current assembly code line by line. The effect of these commands and your assembly code can be seen in the registers of the MSP430. You can view the contents of the all registers by Registers tab of the Debugging View which is shown in Figure .

When you click resume on debugging menu, your program starts running on the board. The debugging view can be terminated by simply clicking on Terminate on debugging menu or pressing Ctrl+F2.



```
SampleProject [Code Composer Studio - Device Debugging]
  TI MSP430 USB1/MSP430 (Suspended)
    $../main.asm:19:43$() at main.asm:19 0xC000  (the entry poi
```

```
            .retainrefs                     ; And retain any sections that have
                                            ; references to current section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop watchdog timer


;-------------------------------------------------------------------------------
; Main loop here

SetupP1     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
```

```
SampleProject
MSP430: Loading complete. There were 34 (code) and 30 (data) bytes written to FLASH/FRAM. The expected RAM usage is 0
(uninitialized data + stack) bytes.
```

Fig. 0.4. New Debugging View

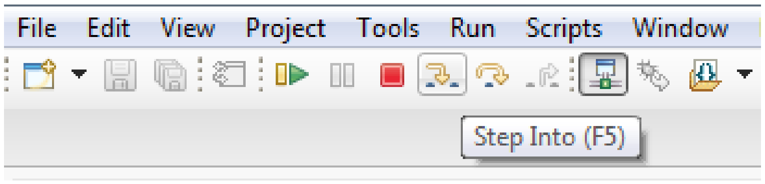
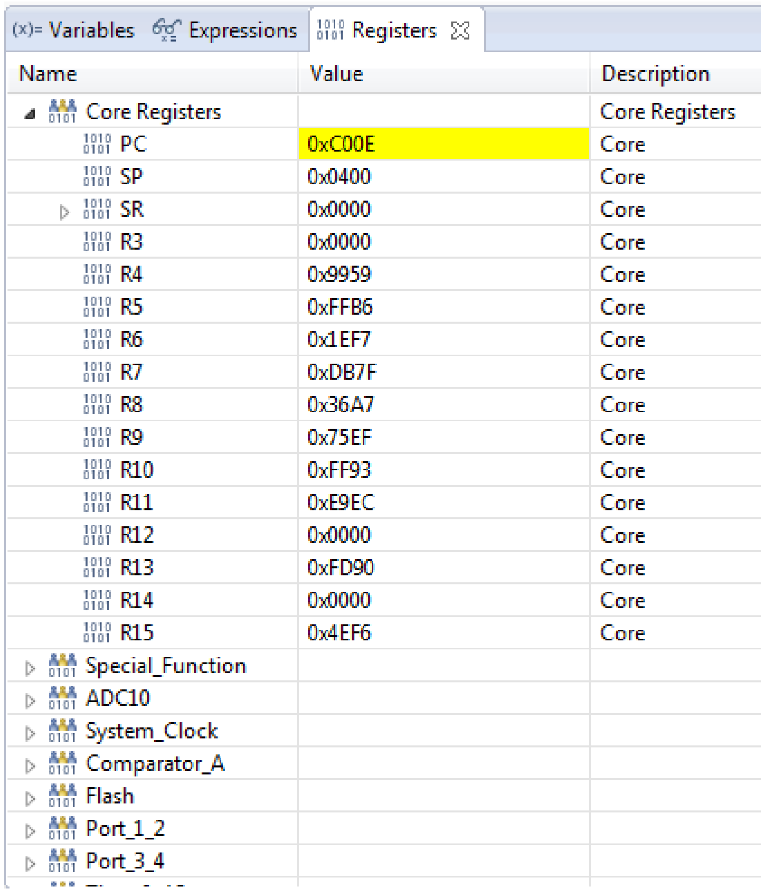


Fig. 0.5. Debugging Menu



Name	Value	Description
Core Registers		Core Registers
PC	0xC00E	Core
SP	0x0400	Core
SR	0x0000	Core
R3	0x0000	Core
R4	0x9959	Core
R5	0xFFB6	Core
R6	0x1EF7	Core
R7	0xDB7F	Core
R8	0x36A7	Core
R9	0x75EF	Core
R10	0xFF93	Core
R11	0xE9EC	Core
R12	0x0000	Core
R13	0xFD90	Core
R14	0x0000	Core
R15	0x4EF6	Core
Special_Function		
ADC10		
System_Clock		
Comparator_A		
Flash		
Port_1_2		
Port_3_4		

Fig. 0.6. Registers Tab

1

Basic Assembly Coding

1.1 Introduction and Preliminary

This lab aims to introduce the MSP430 Education Board, MSP430G2553 microcontroller and MSP430 assembly language.

Before coming to the laboratory, you should investigate the following material .

- BLG351E-Microcomputer_lab_intro
- 1_MSP430_introduction
- 4_MSP430_Education_Board_Document
- MSP430 Instruction Set

1.2 Experiment

1.2.1 Experiment - Part 1

Write the following assembly code to the place marked with ”Main loop here” on your main.asm file.

```
SetupP1      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
```

1.2.2 Experiment - Part 2

Modify your code in Part 1 in such a way that LEDs connected through Port 1 are turned on and off sequentially as shown below.

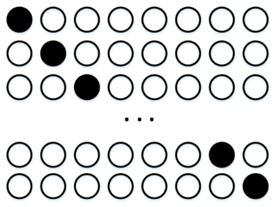


Fig. 1.1. Sequential Operation of the LEDs

1.2.3 Experiment - Part 3

Modify your code in Part 2 in such a way that LEDs connected through Port 1 are turned on and off sequentially as shown below.

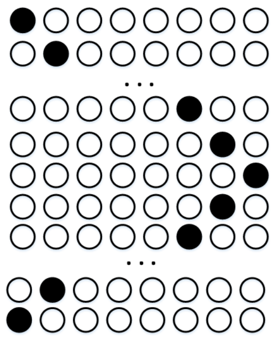


Fig. 1.2. Sequential Operation of the LEDs

1.3 Report

Your report should contain your program code (with explanations) for Part 1, Part 2 and Part 3.