



CS4101 Introduction to Embedded Systems

Lab 1: MSP430 LaunchPad IDE

Prof. Chung-Ta King

Department of Computer Science

National Tsing Hua University, Taiwan



國立清華大學

National Tsing Hua University



Introduction

- In this lab, we will learn the IDE for MSP430 LanuchPad, Code Composer Studio (CCS)
 - Learn how to set up the LaunchPad development board
 - Learn how to create a new project on CCS
 - Learn how to upload a program to the board
 - Debug a program
 - Run a program
 - Use disassembly window

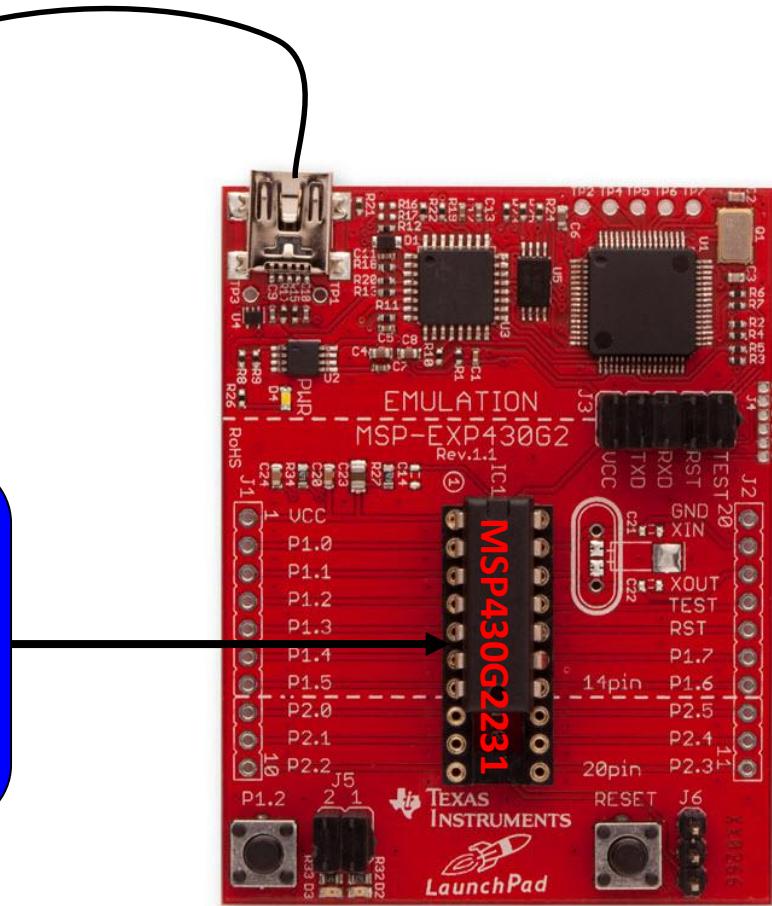


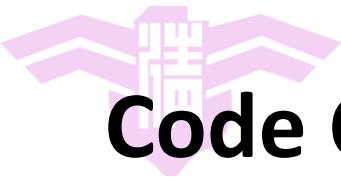


Hardware Setup



Please check the version of
the MSP430 microcontroller
and use the right header
files in your programs

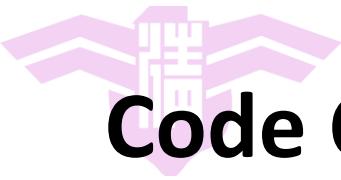




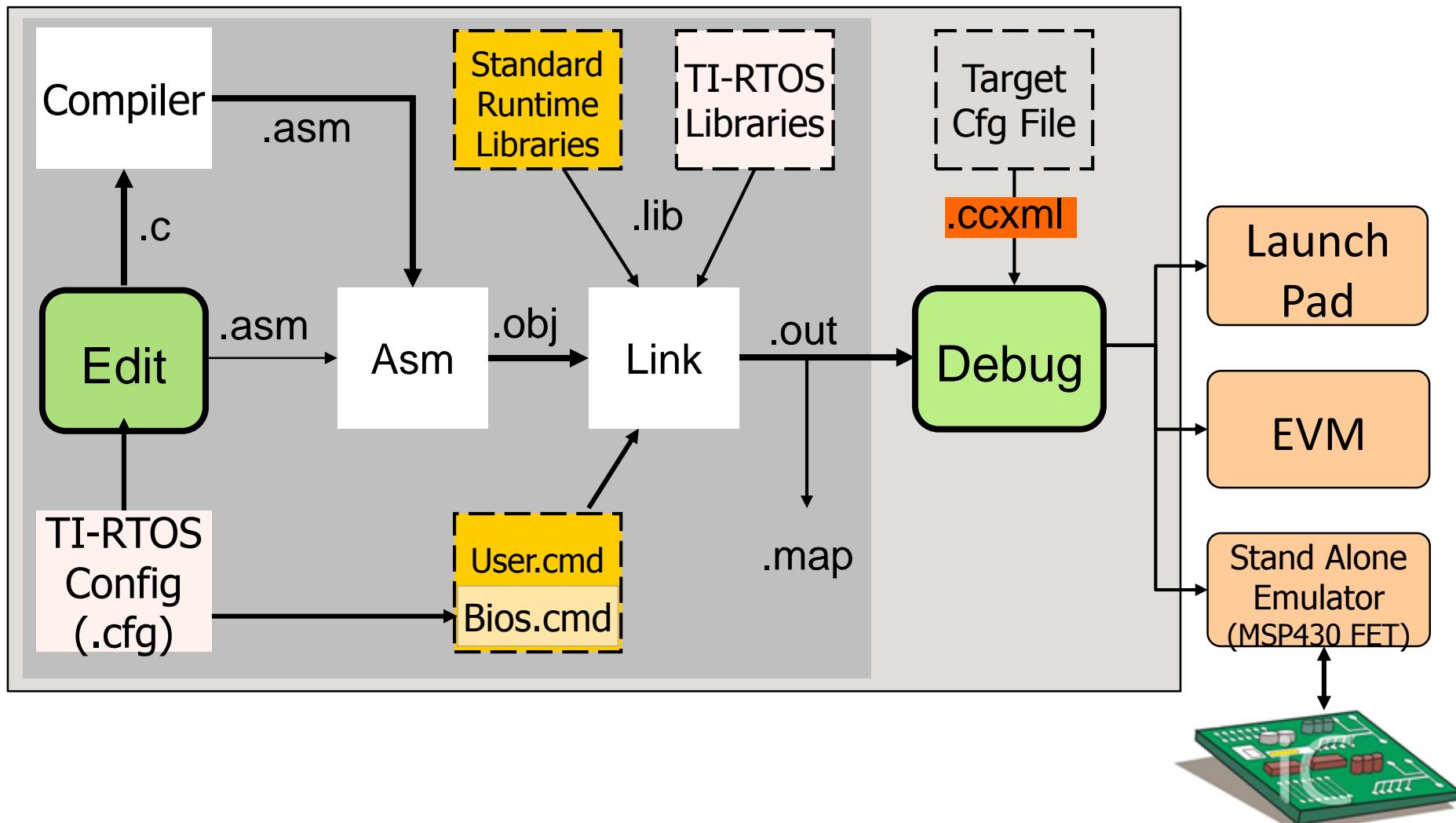
Code Composer Studio (CCS)

- An *Integrated Development Environment* (IDE) based on Eclipse
- Integrated “Debugger” and “Editor” – IDE
 - Edit and Debug have the own “perspectives” (menus, windows)
- Contains all development tools – compilers, TI-RTOS kernel and includes one target – the Simulator





Code Composer Studio (CCS)





CCS GUI – EDIT Perspective

Menus & Buttons

- Specific actions related to EDIT'ing

Perspectives

- EDIT and DEBUG

Project Explorer

- Project(s)
- Source Files

Source EDIT'ing

- Tabbed windows
- Color-coded text

Outline View

- Declarations and functions



國立清華大學

National Tsing Hua University



CCS GUI – DEBUG Perspective

The screenshot shows the CCS Debug perspective with several windows open:

- Project Explorer**: Shows the project structure for "opt_audio_sol".
- Debug**: Shows the connection status to "Spectrum Digital XDS510USB Emulator_0/C674X_0 (Suspended)". It lists the current stack frame: "isrAudio0 at isr.c:111 0x1180C554". Other frames include "ti_sysbios_family_c64p_Hwi_dispatchC_I(int) at Hwi.c:60" and "0x00000000 (no symbols are defined for 0x00000000)".
- Variables**: A table showing variables: blkCnt (unsigned short, value 60), dataIn32 (int, value 28640), and dataOut32 (int, value 109).
- Code Editor**: The file "isr.c" is open, showing C code for an interrupt service routine.
- Console**: The output window showing "Using mDDR settings".

Menus & Buttons

- Related to DEBUG'ing
- Play, Pause, Terminate

Connection Type

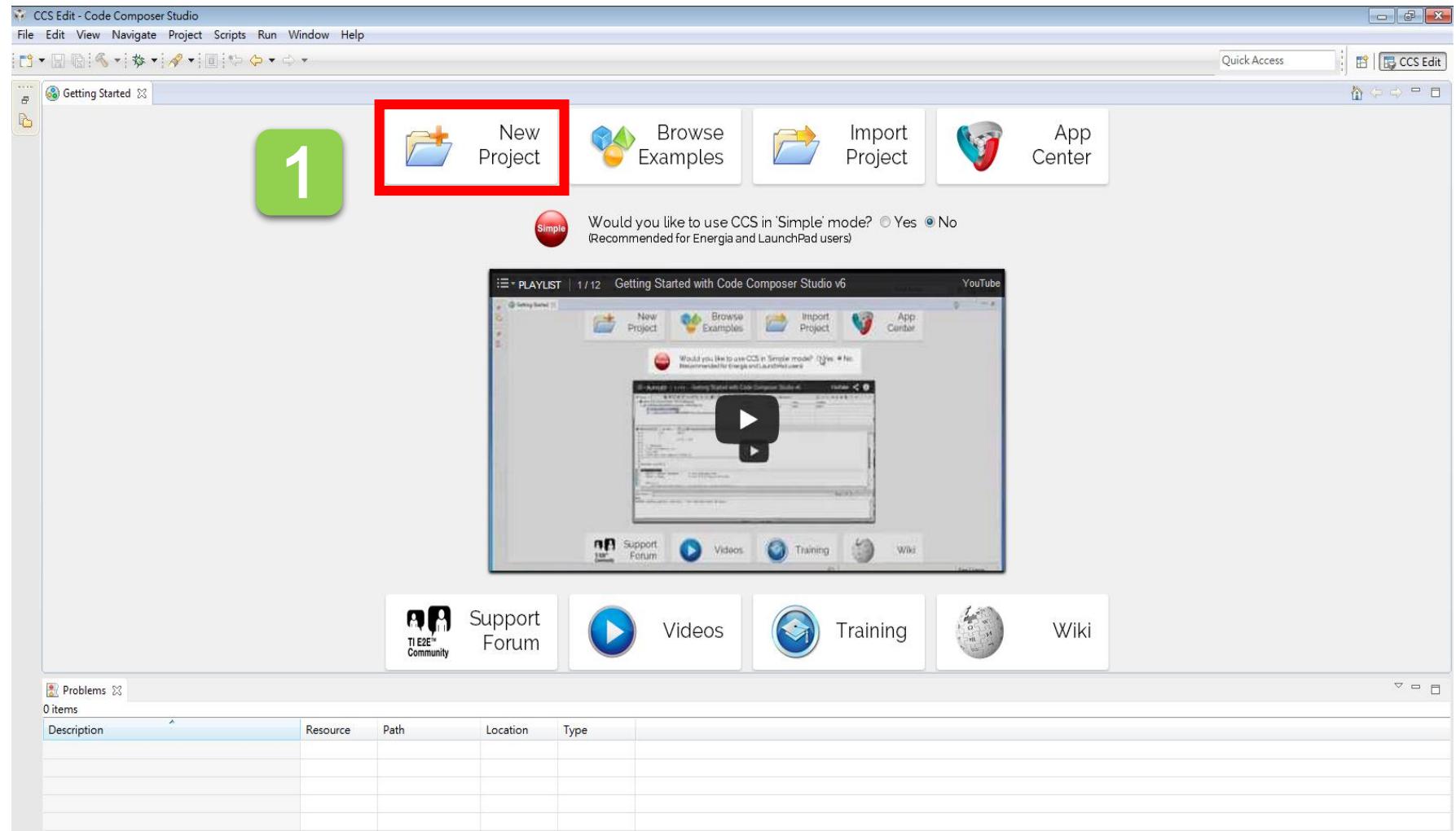
- Specified in Target Cfg file
- What options do users have when connecting to a target?
- This window also provides a “call” stack

DEBUG Windows

- Watch Variables
- Memory Browser
- PC execution point
- Console Window

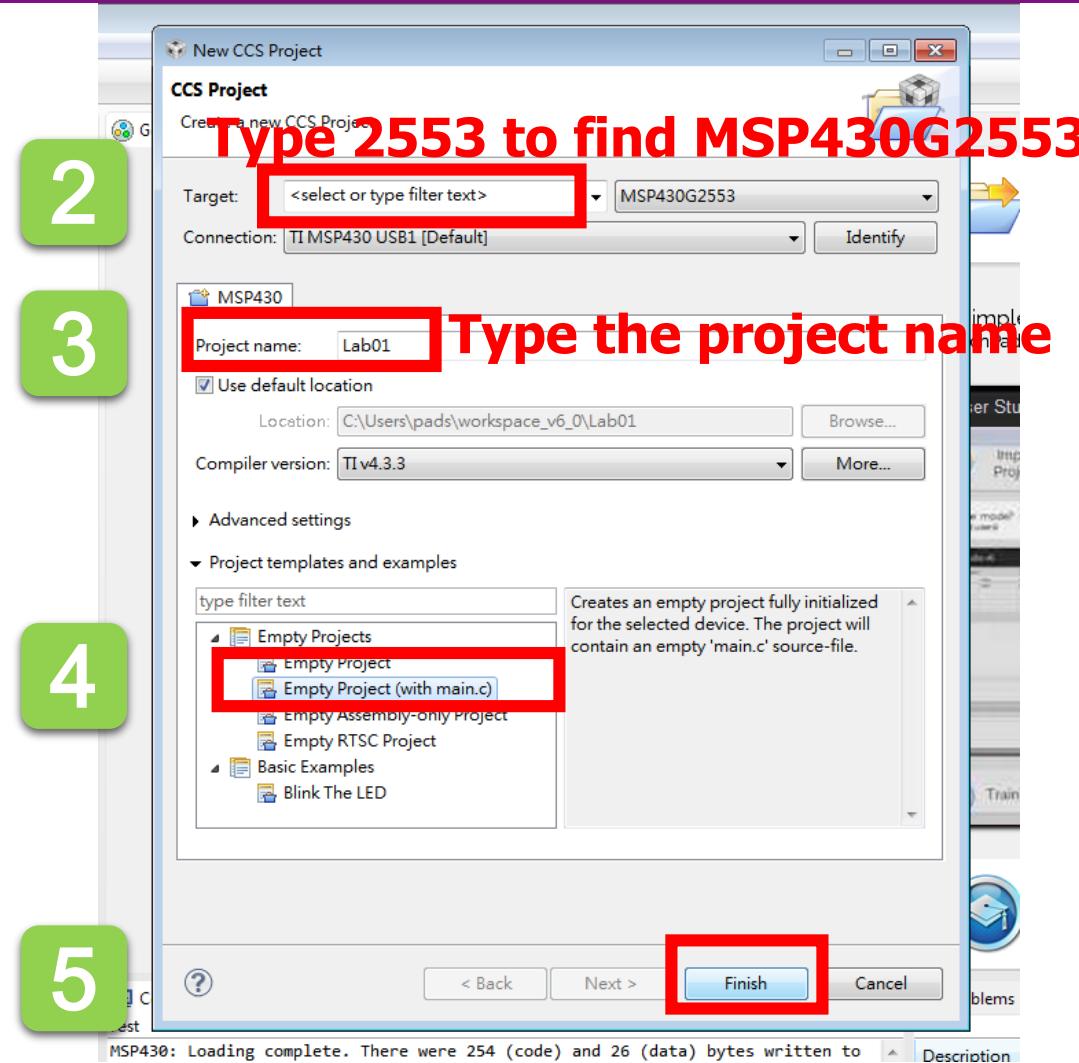


Create a New CCS Project





Create a New CCS Project





Build and Load the Project

- Three buttons on the horizontal toolbar control code generation. You can move your mouse over each button to read their descriptions.



<u>Button Name</u>	<u>Description</u>
1 Build	Incremental build and link of only modified source files
2 Rebuild	Full build and link of all source files
3 Debug	Automatically build, link, load and launch debug-session

- Click the “Build” button and watch the tools run in the Console window. Check for any errors in the Problems window.





Build and Load the Project

- CCS can automatically save modified source files, build the program, open the debug perspective view for debugging, download the program to the target, and run the program at the beginning of the main() function
 - Click on the “**Debug**” button (green bug) or
Click Target → Debug Active Project





Debug Environment

- The basic buttons that control the debug environment are located in the top of CCS:



- At this point you should still be at the beginning of main(). Click the **Run** button to run the code.
 - Notice that the LEDs are toggling, as expected.





End Debug Session and Close Project

- The **Terminate All** button will terminate the active debug session, close the debugger and return CCS to the “C/C++ Perspective” view
 - Click **Target** → **Terminate All** or use the **Terminate All** button
- Next, close the project by right-clicking on project-name in the **C/C++ Projects** window and select **Close Project**.





Use Disassembly Window in Debug

The screenshot shows the Code Composer Studio interface during a debug session. A red box highlights the 'Window' menu at the top left, and another red box highlights the 'Disassembly' option in the 'Show View' submenu. An orange arrow points from the 'Disassembly' menu entry to the 'Disassembly' window on the right. The 'Disassembly' window is also highlighted with a red border. The window displays assembly code for the main function, starting with:

```
main():
  c05e: 8321 DECD.W SP
  24 WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
  c060: 40B2 5A80 0120 MOV.W #0x5a80,&Watchdog_Timer_WDTCTL
  25 P1DIR |= 0x01; // Set P1.0 to output direction
  c066: D3D2 0022 BIS.B #1,&Port_1_2_P1DIR
  30 P1OUT ^= 0x01; // Toggle P1.0 using exclusive-OR
  $C$L1:
  c06a: E3D2 0021 XOR.B #1,&Port_1_2_P1OUT
  32 i = 10000; // SW Delay
  c06e: 40B1 2710 0000 MOV.W #0x2710,0x0000(SP)
  33 do i--;
  $C$L2:
  c074: 8391 0000 DEC.W 0x0000(SP)
  35 }
  c078: 9381 0000 TST.W 0x0000(SP)
  c07c: 27F6 JEQ ($C$L1)
  c07e: 3FFA JMP ($C$L2)
  179 {
    _c_int00(), _c_int00_noexit():
  c080: 4031 0400 MOV.W #0x0400,SP
  182 if(_system_pre_init() != 0) _auto_init();
  c084: 12B0 C080 CALL #_system_pre_init
  c088: 930C TST.W R12
  c08a: 2402 JEQ ($C$L2)
  c08c: 12B0 C000 CALL #_auto_init
```

The bottom status bar indicates: MSP430 - Loading complete. There were 184 (code) and 26 (data) bytes written to FLASH. The expected RAM usage is 80 (uninitialized data + stack) bytes.





Sample Code: Blinking the Red LED

```
#include <msp430x2231.h>
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;      // Stop watchdog timer
    P1DIR |= 0x01;                // Set P1.0 as output

    for(;;) {
        volatile unsigned int i; // prevent optimization

        P1OUT ^= 0x01;          // Toggle P1.0 using XOR
        i = 10000;               // SW Delay
        do i--;
        while(i != 0);
    }
    return 0;
}
```





Lab 1

- Upload sample code to the board and run it. The red LED should blink.
- **Basic 1:** Modify the code to blink the green and red LED, which are located at **Port 1 Bit 0** and **Bit 6..**

```
WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer  
P1DIR = 0x41; 0100 0001 // P1.0 & 6 outputs (red & green LEDs)
```

- If we want to turn on green LED, we should assign P1OUT to 0x40, which is 01000000 in binary.

```
while(1) { 0100 0000  
    P1OUT = 0x40; // P1.6 on (green LED)  
    _delay_cycles(100);  
    P1OUT = 0; // green LED off  
    _delay_cycles(5000);  
}
```





Lab 1

- **Basic 2:**
 - Modify the sample code to remove the **volatile** keyword in declaring the variable i. Observe how the LaunchPad behaves.
- **Basic 3:**
 - To explain the behavior of LaunchPad, let us study the assembly code of the two programs.
 - Use the disassembly window to obtain the assembly code of the original sample code and the modified sample code
 - Observe the differences of these two assembly programs and explain the behavior of LaunchPad.





Assembly for Not Using volatile

```
24     WDTCTL = WDTPW | WDTHOLD;  
      main():  
c078:  40B2 5A80 0120      MOV.W  #0x5a80, &Watchdog_Timer_WDTCTL  
25     P1DIR |= 0x01;  
c07e:  D3D2 0022          BIS.B  #1, &Port_1_2_P1DIR  
31     P1OUT ^= 0x01;  
      $C$L1:  
c082:  E3D2 0021          XOR.B  #1, &Port_1_2_P1OUT  
34     do i--;  
c086:  3FFD                JMP    ($C$L1)  
65     func_epilog_7:      POP    r4  
      __mspabi_func_epilog(), __mspabi_func_epilog_7():  
c088:  4134                POP.W  R4  
66     func_epilog_6:      POP    r5  
      __mspabi_func_epilog_6():  
c08a:  4135                POP.W  R5
```



國立清華大學

National Tsing Hua University



Assembly for Using volatile

```
23     int main(void) {
main():
c05e:  8321          DECD.W  SP
24     WDTCTL = WDTPW | WDTHOLD;
c060:  40B2 5A80 0120  MOV.W   #0x5a80,&Watchdog_Timer_WDTCTL
25     P1DIR |= 0x01;
c066:  D3D2 0022  BIS.B   #1,&Port_1_2_P1DIR
31     P1OUT ^= 0x01;
        $C$L1:
c06a:  E3D2 0021  XOR.B   #1,&Port_1_2_P1OUT
33     i = 10000;
c06e:  40B1 2710 0000  MOV.W   #0x2710,0x0000(SP)
34     do i--;
        $C$L2:
c074:  8391 0000  DEC.W   0x0000(SP)
36     }
c078:  9381 0000  TST.W   0x0000(SP)
c07c:  27F6          JEQ    ($C$L1)
c07e:  3FFA          JMP    ($C$L2)
```





Assembly for Using volatile

```
179  {
    _c_int00(), _c_int00_noexit():
c080:  4031 0400          MOV.W   #0x0400,SP
182      if(_system_pre_init() != 0) _auto_init();
c084:  12B0 C0B0          CALL    #_system_pre_init
c088:  930C                TST.W   R12
c08a:  2402                JEQ    ($C$L2)
c08c:  12B0 C000          CALL    #_auto_init
183      main(0);
        $C$L2:
c090:  430C                CLR.W   R12
c092:  12B0 C05E          CALL    #main
184      abort();
c096:  12B0 C0B4          CALL    #abort
65      func_epilog_7:     POP     r4
    __mspabi_func_epilog(), __mspabi_func_epilog_7():
c09a:  4134                POP.W   R4
66      func_epilog_6:     POP     r5
    __mspabi_func_epilog_6():
```





Grading Policies

- DEMO in the lab ----- 100%
- DEMO within a week (before lab starts) ----- 80%
- DEMO after a week (before lab starts) ----- 60%
- After two weeks (after lab starts)----- 0%

- TA office hour:
 - Mon 10:00 – 12:00
 - Tue 13:00 – 15:00
 - Wed 10:00 – 12:00

