Mechatronics & Embedded Microcomputer Control
ME E4058 Spring 2017

Exercise #3: Introduction to Embedded C

The goal of this exercise is an introduction to using the development system software for microcomputers available from MicroChip Inc. to develop C programs. Exercise #2 can be followed exactly as written except for 1 change. Note that all the simulation tools used to simulate Assembly programs can be used to simulate C programs.

The 3 programs written in Assembly were converted to C to illustrate writing C code. Recall that the 3 programs are:

1. UpDownCounter – When the green button on the microcomputer board is pressed and released, a counter increments. When the red button on the microcomputer board is pressed and released, a counter decrements. The count is displayed on the LEDs on the board. (Note: when a count is decremented from zero, it will indicate all ones.)

2. AtoDpolled – The voltage on the pot on the microcomputer board is read with the A/D converter in the microcomputer device and the value is displayed on the LEDs on the board.

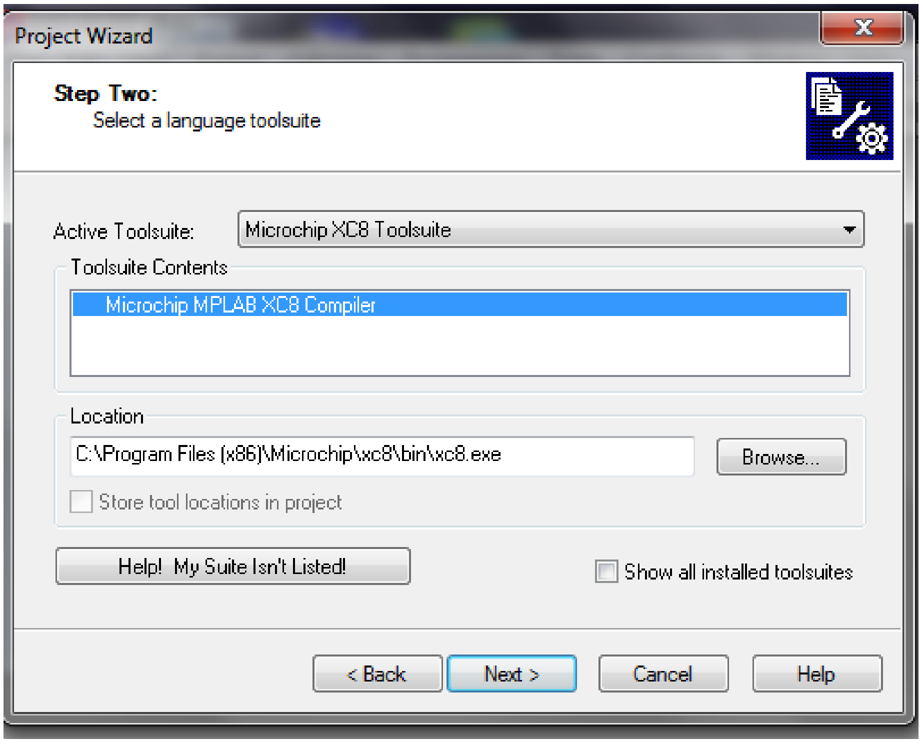
3. Timer – A counter is incremented with a one second interval. The one second software timer requires 3 registers to implement. The count is displayed on the LEDs on the board. The result is that the LEDs increment every second.

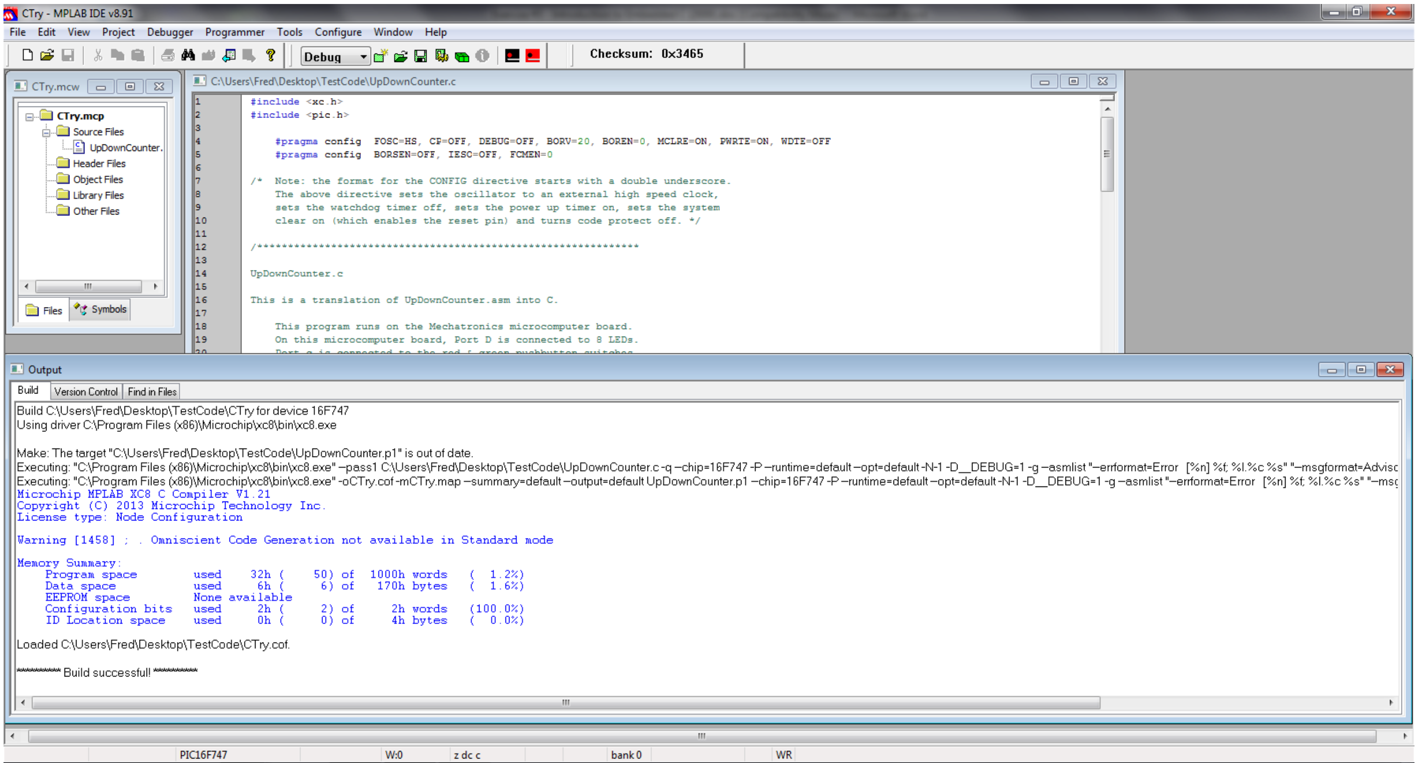
Laboratory Procedure:

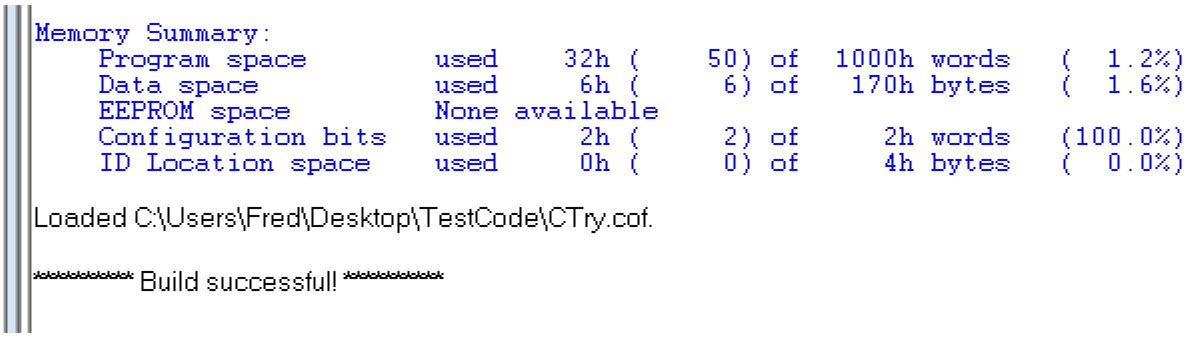
Start by setting up the project in MPLAB using the Project Wizard as before. When you get to the step 2 to select the toolsuite, you must select the ***Microchip XC8 Toolsuite*** and the ***Microchip MPLAB XC8 Compiler*** tools as shown below. For the remainder of the exercise, proceed as in exercise #2. The C program should be saved with a " .c " extension. After this is done, the editor will have color indications for comments, etc. as in Assembler.

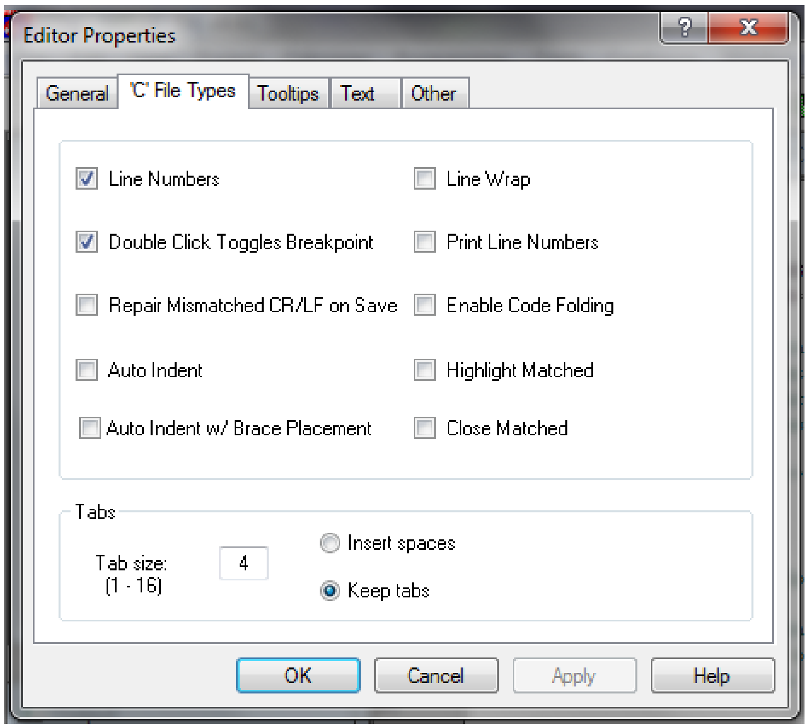
A successful compile of the UpDown Counter program will produce the memory usage map message shown below. Errors in C are not as efficiently displayed as in Assembler. The error listed on a particular line of code could have occurred in a previous or later line. Errors in C are often caused by mismatched { }.

It is often useful to display the line numbers in the C file editor. This is done by selecting editor properties ***Edit > Properties*** and then the tab ***"C" File Types.***



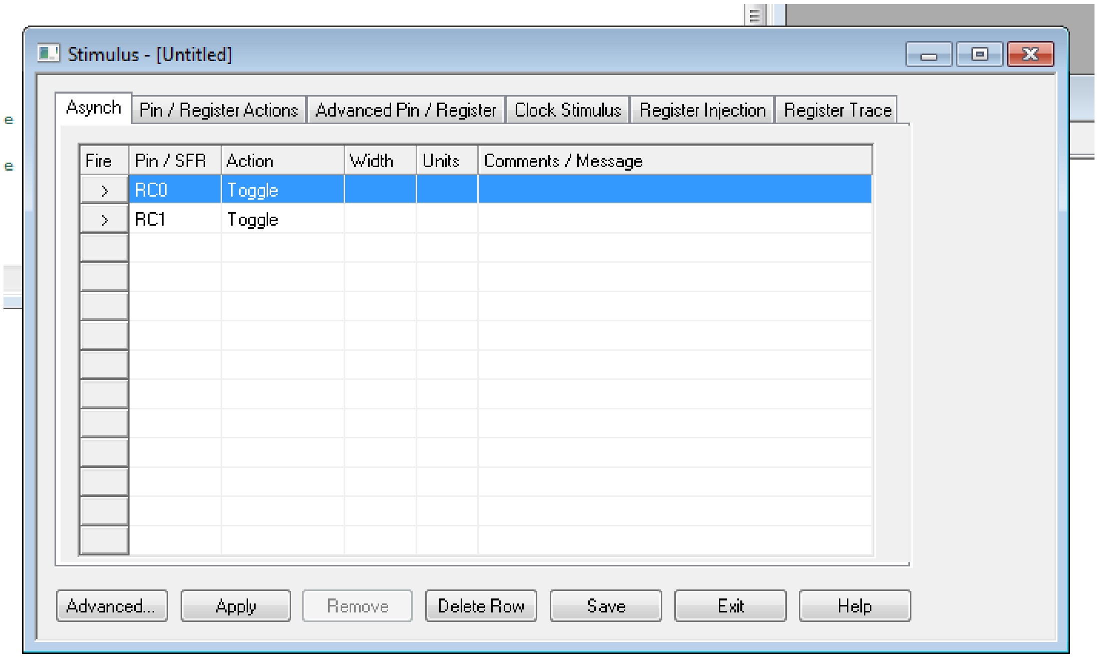


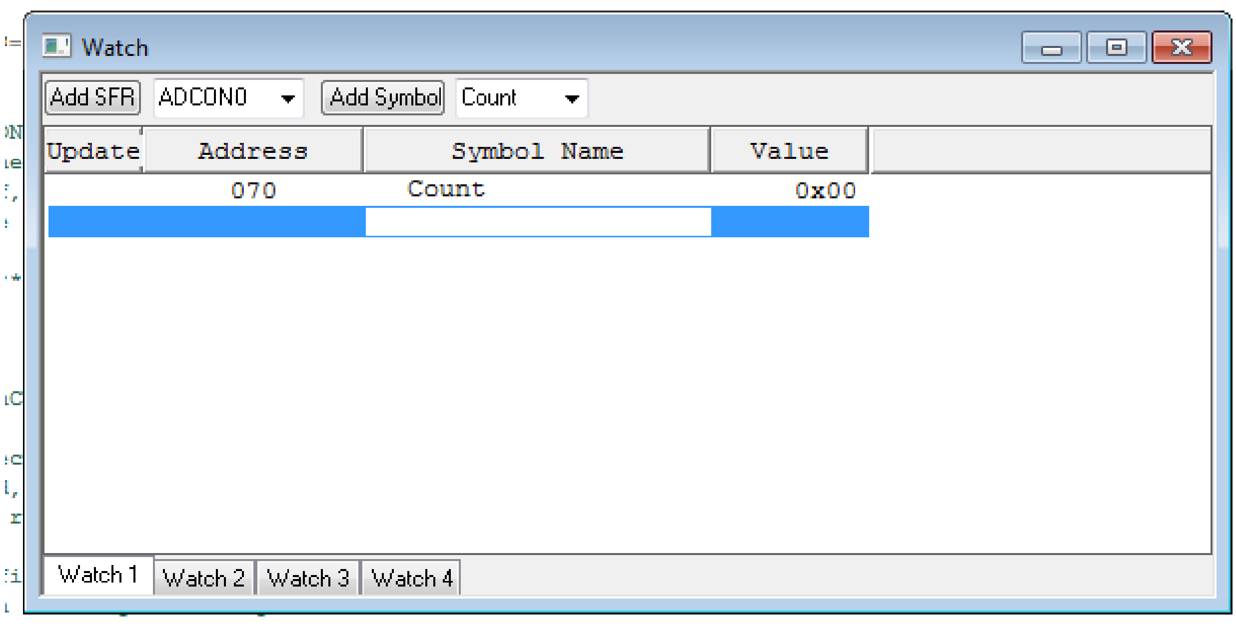




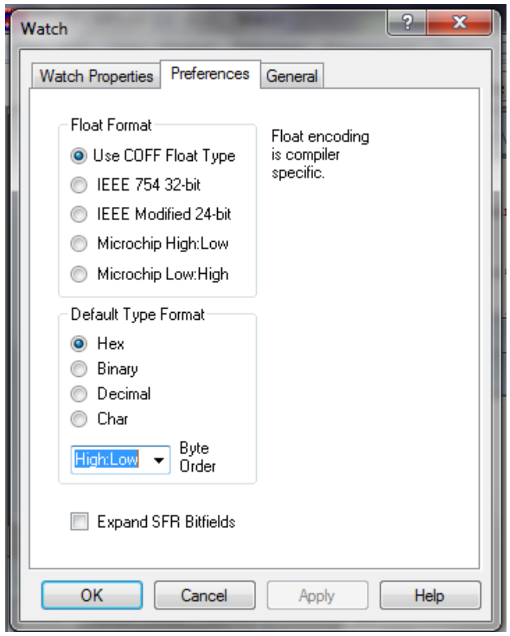
Using the Code Simulator

Using the code simulator for C is the same as Assembler. Stimulus, breakpoints and watch windows should be used as before. Recall that there are two pull downs on the top of the Watch Window. The one on the left labeled “Add SFR” can be used to add the Special Function Registers TRISC, TRISD, ADCON1, PORTC and PORTD etc. The one on the right labeled “Add Symbol” can be used to add the C symbols (for example Count below). The address shown is 070 hex. You can also use the Watch window to display the value of Special Function Registers (such as TRISC and PORTC).

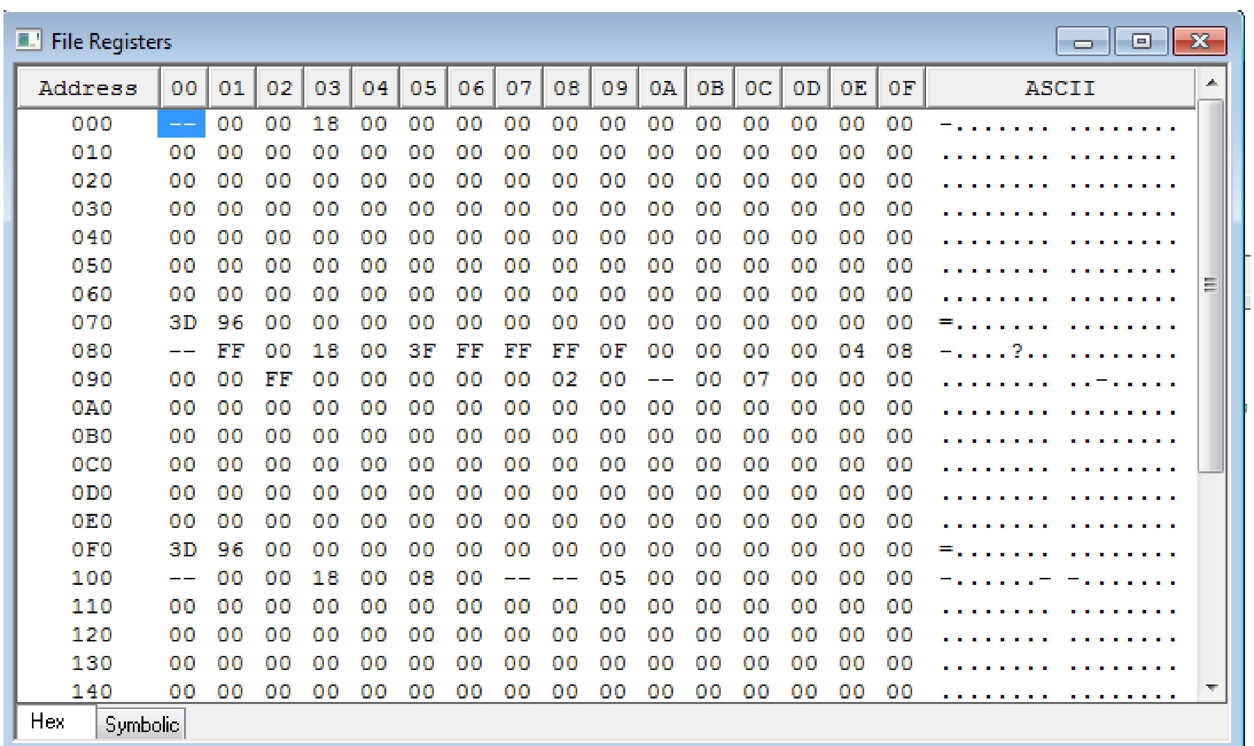




When you enter the Long Timer code you defined a variable Timer to be a long. Since Timer is defined as a long integer, the watch window will have to display 4 registers to show it. C will store the low byte in register 70, the next byte in 71, and so on. If you want to change the representation of the symbol to, for example, a signed decimal as below, you have to say that the representation is ***HighLow***. You right click in the Watch window and then select the order under ***Preferences.***



Recall that you can also view all the registers by selecting ***View>File Registers*** which shows all data memory. You can also add the three general purpose registers to the watch window by the register number or name.

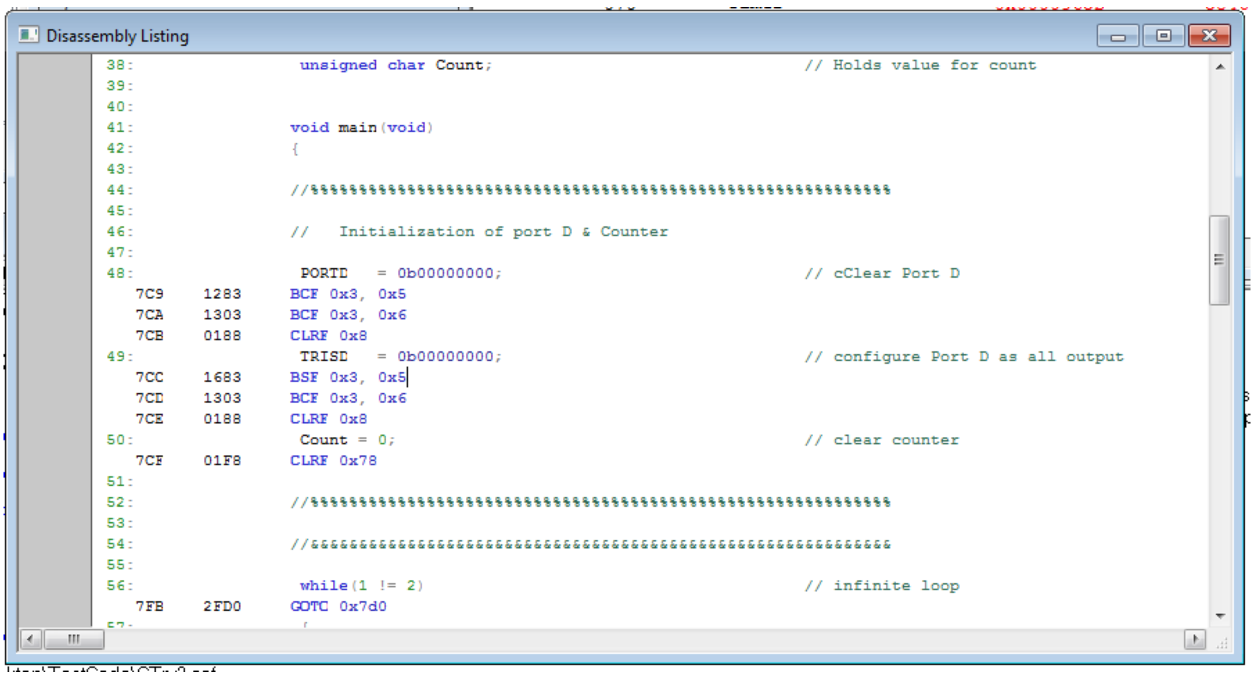


File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	00	18	00	00	00	00	00	00	00	00	00	00	00	00	-....... ....
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	3D	96	00	00	00	00	00	00	00	00	00	00	00	00	00	00	=....... ....
080	--	FF	00	18	00	3F	FF	FF	FF	0F	00	00	00	00	04	08	-....?.. ....
090	00	00	FF	00	00	00	00	00	02	00	--	00	07	00	00	00	....-.....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	3D	96	00	00	00	00	00	00	00	00	00	00	00	00	00	00	=....... ....
100	--	00	00	18	00	08	00	--	--	05	00	00	00	00	00	00	-......- -.......
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Hex Symbolic

You can see the Assembly code generated by your C program by selecting ***View>Disassembly Listing.***



```c
#include <xc.h>
#include <pic.h>

    #pragma config  FOSC=HS, CP=OFF, DEBUG=OFF, BORV=20, BOREN=0, MCLRE=ON, PWRTE=ON, WDTE=OFF
    #pragma config  BORSEN=OFF, IESO=OFF, FCMEN=0

/*  Note: the format for the CONFIG directive starts with a double underscore.
    The above directive sets the oscillator to an external high speed clock,
    sets the watchdog timer off, sets the power up timer on, sets the system
    clear on (which enables the reset pin) and turns code protect off. */

/***************************************************************

UpDownCounter.c

This is a translation of UpDownCounter.asm into C.

    This program runs on the Mechatronics microcomputer board.
    On this microcomputer board, Port D is connected to 8 LEDs.
    Port c is connected to the red & green pushbutton switches.

    This program increments a file register Count every time
    the green pushbutton switch (PortC pin 0) is pressed.
    The program decrements the file register Count every time
    the red pushbutton switch (PortC pin 1) is pressed.
    The value of Count is displayed on the LEDs connected
    to Port D.

    The net result is that LEDs should increment or decrement
    in a binary manner every time a switch is pressed.

***************************************************************/

/* Variable declarations */

#define PORTBIT(adr,bit)        ((unsigned)(&adr)*8+(bit))

// The function PORTBIT is used to give a name to a bit on a port
// The variable RC0 could have equally been used

    static bit      greenButton     @   PORTBIT(PORTC,0);
    static bit      redButton    @   PORTBIT(PORTC,1);

    char Count, i;

void    SwitchDelay (void)                      // Waits for switch debounce

{
    for (i=200; i > 0; i--) {}                  // 1200 us delay
}

void    main (void)
{

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    PORTD   = 0B00000000;                       // Clear Port D output latches
    TRISD   = 0B00000000;                       // Configure Port D as all output
    TRISC   = 0B11111111;                       // Configure Port C as all input

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

    while(1 != 2)                               // Infinite loop
    {

        if(greenButton == 1)                    // If green press...
        {
            while(greenButton == 1){}           // Wait for release
            SwitchDelay();                      // Let switch debounce
            Count++;                            // Increment Count
```

```
            PORTD = Count;                      // Display Count value on PORTD LEDs
        }

        else if(redButton == 1)                 // If red press...
        {
            while(redButton == 1){}             // Wait for release
            SwitchDelay();                      // Let switch debounce
            Count--;                            // Decrement Count
            PORTD = Count;                      // Display Count value on PORTD LEDs
        }
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

}
```

```c
#include <xc.h>
#include <pic.h>

    #pragma config  FOSC=HS, CP=OFF, DEBUG=OFF, BORV=20, BOREN=0, MCLRE=ON, PWRTE=ON, WDTE=OFF
    #pragma config  BORSEN=OFF, IESO=OFF, FCMEN=0

/*  Note: the format for the CONFIG directive starts with a double underscore.
    The above directive sets the oscillator to an external high speed clock,
    sets the watchdog timer off, sets the power up timer on, sets the system
    clear on (which enables the reset pin) and turns code protect off. */


/***************************************************************

AtoDpolled.c

This is a translation of AtoDpolled.asm into C.

    This program illustrates the operation of the PIC16F74's
    Analog to Digital (A/D) converter. One A/D Channel is selected.
    The A/D is polled to determine when it is finished. The program
    sits in a continuous loop until th G0 bit in register ADCON0
    ndicates that the A/D conversion has completed.

    The A/D is configured in ADCON0 and ADCON1 as follows:
        Vref = +5V internal
        A/D Osc. =  8 * oscillator period
        A/D Channel = AN0 (RA0)

    Hardware for this program is the Mechatronics microcomputer board.
    The program converts the potentiometer value on RA0 and displays it as
    an 8 bit binary value on Port D.

***************************************************************/

// Variable declarations

    char Temp;                                  // Variable for delay loop


void    SetupDelay(void)                        // Delay loop
{
    for (Temp=1; Temp > 0; Temp--) {}           // 17 us delay
}

void    initAtoD(void)                          // Initialize A/D
{
    ADCON1  = 0b00000100;                       // RA0,RA1,RA3 analog inputs, rest digital
    ADCON0  = 0b01000001;                       // Select 8* oscillator, analog input 0, turn on
    SetupDelay();                               // Delay a bit before starting A/D conversion
    GO  = 1;                                    // Start A/D
}

//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

void    main(void)
{

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//  Initialization of ports and A/D

    PORTD   = 0b00000000;                       // Set Port D low
    TRISD   = 0b00000000;                       // Configure Port D as all output

    initAtoD();

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

    while(1 != 2)                           // Infinite loop
```

```
    {
        while(GO == 1){}                        // Wait here until A/D conversion is done
        while(GO == 1){}                        // Make sure A/D has finished
        PORTD   = ADRESH;                       // Display A/D value on Port D LED's
        GO  = 1;                                // Restart A/D
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

}
```

```c
#include <xc.h>
#include <pic.h>

    #pragma config  FOSC=HS, CP=OFF, DEBUG=OFF, BORV=20, BOREN=0, MCLRE=ON, PWRTE=ON, WDTE=OFF
    #pragma config  BORSEN=OFF, IESO=OFF, FCMEN=0

/*  Note: the format for the CONFIG directive starts with a double underscore.
    The above directive sets the oscillator to an external high speed clock,
    sets the watchdog timer off, sets the power up timer on, sets the system
    clear on (which enables the reset pin) and turns code protect off. */

/***************************************************************

Timer.c

This is a translation of Timer.asm into C.

    This program illustrates using a long integer (32 bits) to form a timer
    with a one second period.

    Since a "for" delay loop is 26 usec using an unsigned long number
    (determined in the simulator), you need approximately 38,461 loops to
    equal 1 second. It therefore requires at least 2 8-bit registers to store
    this large a number. An unsigned long is 4 registers. (Note: if you use an
    unsigned integer to hold the value, the for loop is quicker and the value
    needed does not fit into the 2 registers.)

    Hardware for this program is the Mechatronics microcomputer board.
    The program counts the seconds and displays the count as an 8 bit
    binary value on Port D. The LEDs on Port D should therefore increment
    by one every second.

***************************************************************/

// Variable declarations

    unsigned long Timer;                            // Holds value for 1 sec timer
    unsigned char Count;                            // Holds value for count


void    main(void)
{

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//  Initialization of port D & Counter

    PORTD   = 0b00000000;                           // cClear Port D
    TRISD   = 0b00000000;                           // configure Port D as all output
    Count = 0;                                      // clear counter

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

    while(1 != 2)                                   // infinite loop
    {
        for (Timer = 38461; Timer > 0; Timer--) {}  // 1s delay loop
            Count++;                                // increment count
            PORTD = Count;                          // display count value on Port D

//      could also have done PORTD++ to display count value on Port D LEDs

    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

}
```