ECE383: Microcomputers – Lab 5
C and PIC24 Assembly Language Programming

Goals: The goals of this lab are to introduce students to basic C language programming and equivalent PIC24 assembly language programming.

Introduction

This lab introduces basic C language programs and equivalent PIC24 assembly language programs. The tasks in this lab are:

- Implement programming tasks using the C language.
- Implement equivalent programs using the PIC24 assembly language.

This lab requires you to capture portions of the screen. For windows computers, you can use the "Snipping Tool" application, though other third party tools are also available. Additionally, you will need to capture video demos of your code operating on hardware. These videos should be captured using your smart phone or similar recording device.

As always, read through the entire lab and review any supplied files before starting to work on the lab deliverables. The deliverables have you verify computations (by hand) performed by the assembly language or C program. In all cases, make it easy for the TA to verify your computations by showing each step of your work and keeping it logically organized.

NOTE: When writing MPLAB assembly language programs, do not use variable names **a** or **b** as these are reserved names.

This lab will make use of several C header files discussed in the textbook. These files should be located in the *C:\microchip\lib\include* directory, which were to be installed as a part of the MPLAB install process with the **pic24_code_examples_mplab.zip** file.

1. TASK 1: Basic C arithmetic operations

For this task you will create multiple C and assembly language projects implementing arithmetic operations. Perform the following steps:

- Start the MPLAB IDE. Use *Project->Project Wizard* for the creation of an MPLAB project.
  - **Step One:** Select the device *PIC24HJ128GP502*.
  - **Step Two:** Select the Active Toolsuite as *Microchip C30 Toolsuite*.
  - **Step Three:** *Create a New Project File* in a unique directory on the local hard disk (*C:\temp\task1.mcp* as an example).
  - **Step Four:** Skip the addition of existing files to the project.

- After the project is open, use *Project->Build Options* to add the *C:\microchip\lib\include* directory to the *Include Search Path* directory as shown in Figure 1 below.

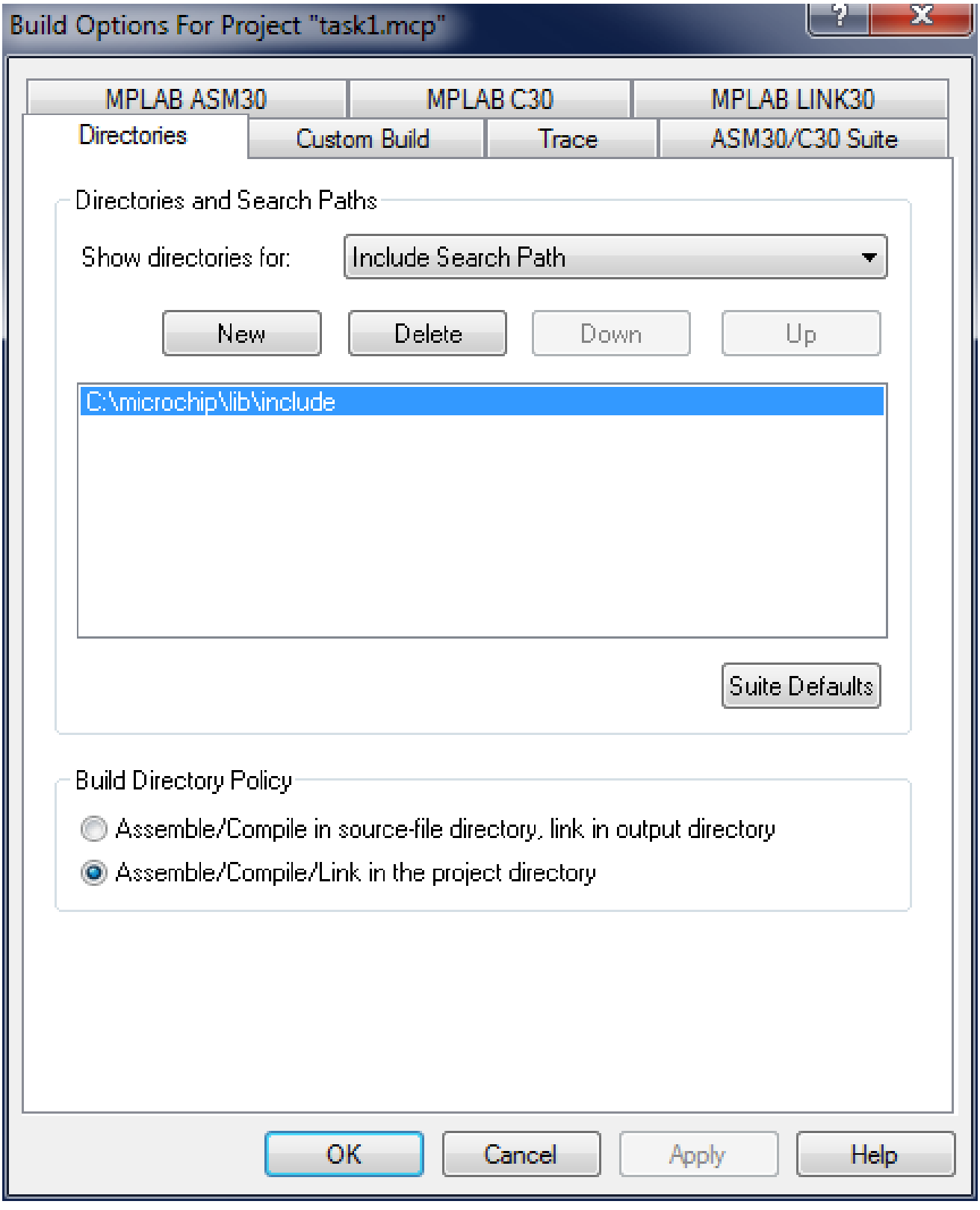

Figure 1. Include Search Path Directory.

- Enter the C program below and save it with the name *task1.c* as a part of the project.

```
#include "pic24_all.h"

uint16 u16_a, u16_b, u16_c, u16_d;
uint8  u8_x,  u8_y,  u8_z;

void main(void) {
  u8_x=0xFE;
  u8_y=0x02;
  u16_a = 0xFFFE;
  u16_b = 0x0002;

  u8_z=u8_x+u8_y;
  u16_d=(uint16) u8_x + (uint16) u8_y;
  u16_c=u16_a+u16_b;
}
```

- Use *Project->Build All* to compile the program. If the source file is not already open, double-click on the *task1.c* source file to open it.
- After the project is compiled, use *View->Program Memory* and open the program memory window. Scroll the window until you find your program in memory. Your program should begin at location 0x200.
- Use *View->File Registers* to view data memory. Scroll to location 0x800, which is where your variables will start.
- Use *View->Special Function Registers* to view the special function registers (these will appear as WREG0-WREG15, etc).
- Open a watch window *(View->Watch)* and use *Add Symbol* to watch variable values of the all the program variables. Use *Add SFR* to watch the **SR** (status register) special function register value.
- Use *Debugger->Select Tool->MPLAB Sim* to select the MPLAB Simulator.
- Use *Debugger->Step Into (F7)* to single step through the program. Watch both the memory locations and watch window locations, and correlate their changing values with the instructions being executed.
- **For all three of the arithmetic operations, record the value of the result and the value of the sign/negative (N), carry (C), zero (Z), and overflow (V) flags**. The value of the flags can be determined from the SR register and are also indicated at the bottom of the MPLAB IDE.

  Deliverable 1: For all three of the arithmetic operations for TASK 1, enter the value of the sign/negative (N), carry (C), zero (Z), and overflow (OV) flags of the STATUS REGISTER. Each flag value should be entered as a single bit (0 or 1).

  Deliverable 2: In your own words, what is type-casting in the context of the C-code program for TASK 1? Where was type-casting used in the C-code program and what was its effect? Why is type-casting useful when writing C-code?

  Deliverable 3: Upload a screen-capture of the MPLAB IDE environment that clearly shows that complete set of results of program execution for TASK 1. The complete set of results shows the filename of the program, that the program built successfully, and that the special function registers, data memory

locations, and program memory have the expected values. Additionally, make sure that your name and CWID are shown as text either as a comment in the program source code or in an external text editor that is open (but do not cover any of the necessary deliverables).

2. TASK 2: C Program check_val

Create an MPLAB project (*task2.mcp*) containing a C program (*task2.c*) that counts the number of one bits in a 16-bit unsigned integer named **`check_val`**. The count value should be stored in an 8-bit unsigned variable named **`ones_count`**. The program should also determine which is the first bit set in the **`check_val`** variable. The location of the first bit set should be stored in an 8-bit unsigned variable named **`first_one`**. For example, if **`check_val`**=0xF508 then the computed values should be **`ones_count`**=7 and **`first_one`**=3. You must download your program to your PIC24 hardware (i.e. the Microstick II) and demonstrate the execution of your program on hardware to the TA.

Hint: Use a **for** loop and shift right 1 bit on every iteration of the loop to simplify testing of a bit value.

Deliverable 4: Upload a screen-capture of the MPLAB IDE environment that clearly shows that complete set of results of program execution for TASK 2. The complete set of results shows the filename of the program, that the program built successfully, and that the special function registers, data memory locations or data variables have the expected values. Additionally, make sure that your name and CWID are shown as text either as a comment in the program source code or in an external text editor that is open (but do not cover any of the necessary deliverables).

Deliverable 5: Upload the C-code source file you wrote to meet the TASK 2 requirements. Remember also that comments are required for all source code to detail the intent of the code. Source code without comments will be given a zero.

Deliverable 6: Upload a video that captures your TASK 2 program running on the PIC24 hardware (the Microstick II). This video needs to show the successful build, download, and execution of the code. During execution of the code, set breakpoints to halt your code and show the successful shifting of bits in your **for** loop. The correct final values in the **ones_count** and **first_one** variables must be shown after the complete program execution. During this video, audio must also be captured where you describe each step for the TA to follow. At the end of your video also make sure to include a final shot that shows your ACT card or similar photo ID. Videos without this final detail will be given a grade of zero.

3. TASK 3: Assembly Language Program check_val

Create an MPLAB project (*task3.mcp*) containing an assembly language program (*task3.s*) that implements the equivalent of Task 2. You must download your program to your PIC24 hardware and demonstrate the execution of your program on hardware to the TA.

Deliverable 7: Upload a screen-capture of the MPLAB IDE environment that clearly shows that complete set of results of program execution for TASK 3. The complete set of results shows the filename of the program, that the program built successfully, and that the special function registers, data memory locations or data variables have the expected values. Additionally, make sure that your name and CWID are shown as text either as a comment in the program source code or in an external text editor that is open (but do not cover any of the necessary deliverables).

Deliverable 8: Upload the assembly language source file you wrote to meet the TASK 3 requirements. Remember also that comments are required for all source code to detail the intent of the code. Source code without comments will be given a zero.

Deliverable 9: Upload a video that captures your TASK 3 program running on the PIC24 hardware (the Microstick II). This video needs to show the successful build, download, and execution of the code. During execution of the code, set breakpoints to halt your code and show the successful shifting of bits in your **for** loop. The correct final values in the **ones_count** and **first_one** variables must be shown after the complete program execution. During this video, audio must also be captured where you describe each step for the TA to follow. At the end of your video also make sure to include a final shot that shows your ACT card or similar photo ID. Videos without this final detail will be given a grade of zero.

4. TASK 4: Assembly to C Example

Create a new assembly project and input the assembly code as shown below. See the program results after executing the program. Create an MPLAB project (*task4.mcp*) containing a C program (*task4.c*) that implements the equivalent of the assembly program.

```
      .include "p24Hxxxx.inc"
      .global __reset

      .bss              ; Uninitialized data section
                        ; Variables start at location 0x0800
x:       .space 2       ; Allocating space (two bytes) to variable.
y:       .space 2       ; Allocating space (two bytes) to variable.
count:   .space 1       ; Allocating space (one byte) to variable.

;.....................................................................
; Code Section in Program Memory
;.....................................................................

      .text                         ; Start of Code section
__reset:                            ; first instruction located at __reset label
        mov #__SP_init, w15         ; Initialize the Stack Pointer
        mov #__SPLIM_init,W0
        mov W0, SPLIM               ; Initialize the stack limit register
                                    ; __SP_init set to be after allocated data

; User Code starts here.
           mov #0x3, w0
           mov.b wreg, count
           mov #0x1, w1
           mov w1, x
           mov #0x3, w2
           mov w2, y
top:
           cp0.b count
           bra z, done

           cp w1, w2
           bra nz, next
           inc w2, w2
           mov w2, y
next:
           cp w1, w2
           bra GEU, next2
           add #0x2, w1
           mov w1, x
next2:
           dec.b count
           bra top

done:      goto done  ; Place holder for last line of executed code

.end                  ; End of program code in this file
```

You must download your program to your PIC24 hardware and demonstrate the execution of your program on hardware to the TA.

Deliverable 10: Upload a screen-capture of the MPLAB IDE environment that clearly shows that complete set of results of program execution for TASK 4. The complete set of results shows the filename of the program, that the program built successfully, and that the special function registers, data memory locations or data variables have the expected values. Additionally, make sure that your name and CWID are shown as text either as a comment in the program source code or in an external text editor that is open (but do not cover any of the necessary deliverables).

Deliverable 11: Upload the C-code source file you wrote to meet the TASK 3 requirements. Remember also that comments are required for all source code to detail the intent of the code. Source code without comments will be given a zero.

Deliverable 12: Upload a video that captures your TASK 4 program running on the PIC24 hardware (the Microstick II). This video needs to show the successful build, download, and execution of the code. During the execution of this code, audio must also be captured while you step through each individual line of code and describe what it is doing and which assembly language instruction (or instructions) it is implementing. At the end of your video also make sure to include a final shot that shows your ACT card or similar photo ID. Videos without this final detail will be given a grade of zero.