

# **C and PIC24 Assembly Language Programming**

## **5th Laboratory Report for ECE 383**

### **Microcomputers**

**Submitted by**

**Yichen Huang**

**11906882**

**Shomari Thomas**

**11672867**

**The University of Alabama**

**Tuscaloosa, Alabama 35487**

**February 19, 2020**

## Table of Contents

<b>Abstract.....</b>	<b>3</b>
<b>Introduction .....</b>	<b>4</b>
<b>Prelab .....</b>	<b>4</b>
Task 1- Basic C arithmetic operations.....	4
Task 2 – C Program “check_val”.....	5
<b>Procedure/Results.....</b>	<b>5</b>
Task 3 – Assembly Language Program check_val.....	5
Task 4 – Assembly to C Example .....	6
<b>Conclusion.....</b>	<b>6</b>
<b>Appendixes.....</b>	<b>7</b>
<b>Appendix A – Task 1 .....</b>	<b>7</b>
Source Code – “task1.c” .....	7
Figure (1 - 6).....	8
<b>Appendix B – Task 2 .....</b>	<b>12</b>
Source code – “task2.c” .....	12
Figure (7 - 18).....	12
<b>Appendix C – Task 3 .....</b>	<b>21</b>
Source code – “task3.s” .....	21
Figure (19 - 30).....	24
<b>Appendix D – Task 4 .....</b>	<b>32</b>
Source code – “task4.c” .....	32
Figure (31 - 42).....	33

## Abstract

Lab 5 was an introduction to translating basic C language code to PIC24 assembly language instructions. In order to become more familiar with this practice, we executed programs using the C language as well as their equivalent programs in PIC24 assembly language. During this lab, we implemented basic C arithmetic operations, converted PIC24 assembly language to C program language, converted C program language to PIC24 assembly language, and implemented use of PIC24 hardware.

For task 1, we used the C program provided in the lab document to create multiple C and assembly language projects implementing arithmetic operations. In task 2, we used MPLAB to create a project containing a C program that meets the specifications provided in the lab document. For task 3, we created a PIC24 assembly language program that is the equivalent of the C program from task 2. Task 4 consisted of implementing the provided PIC24 assembly language program in the lab document, writing and implementing the equivalent of the C program language, and downloading the program onto the PIC24 hardware.

In Lab 5, we became familiar with translation between C program language and PIC24 assembly language and verified the success of our programs using live results and the accuracy of the response from the PIC24 hardware after successful download of a C program equivalent to an assembly language program.

## Introduction

In Lab 5 we were introduced to translating basic PIC24 assembly language into C programming language and C programming language to PIC24 assembly language in the MPLAB Integrated Development Environment (IDE). The MPLAB Integrated Development Environment (IDE) was used as a helpful tool to simulate both the assembly language programs and the C language programs and provide the results in the form of the final state of the programs, data memory, and watch windows. Additionally, a C language program was downloaded onto a PIC24 device, giving a physical example of the capabilities of the C language program and its application in PIC24 hardware use. This lab exemplified the similarities between the capabilities of C language and PIC24 assembly language via simulation, and showed how we can use the C language, PIC24 assembly language, and MPLAB Integrated Development Environment together to solve problems.

## Prelab

### Task 1- Basic C arithmetic operations

For Exercise 1 of Lab 5, we first created a new project and new project file for use with the PIC24HJ128GP502 device in the MPLAB IDE. In the project we entered the C program provided in the lab document. After compiling the program, we watched the program memory, beginning in 0x200, data memory located in 0x800 of the file registers, and special function registers. Lastly, we opened the watch window, and after adding the SFR symbol ran the MPLAB Simulator. In the simulator, we were able to watch and correlate the changing values of the memory and watch window locations with the instructions causing the changes. We were

then able to see the value of the result and the value of the sign/negative (N), carry (C), zero (Z), and overflow (V) flags for all three of the arithmetic operations. The source code and the pictures are in [Appendix A](#).

### Task 2 – C Program “check\_val”

In Task 2, we created an MPLAB project and wrote a C language program containing three specialized variables. The variable `check_val` counts the number of one bit in a 16-bit unsigned integer named. For the `check_val` variable, the program also determines which is the first bit set. The variable `ones_count` is an 8-bit unsigned variable in which the count value should be stored. The variable `first_one` is an 8-bit unsigned variable and the location where the first bit set should be stored. Once the C program was successfully compiled, we downloaded the program onto the PIC24 hardware successfully. The source code and the pictures of the windows are in [Appendix B](#).

## Procedure/Results

### Task 3 – Assembly Language Program `check_val`

In Exercise 3, we created an MPLAB project and wrote an assembly language program equivalent to the C language program in task 2. Just as in task 2, the three variables described in the lab document followed their given requirement. The variable `check_val` counts the number of one bit in a 16-bit unsigned integer named. For the `check_val` variable, the program also determines which is the first bit set. The variable `ones_count` is an 8-bit unsigned variable in which the count value should be stored. The variable `first_one` is an 8-bit unsigned variable and

the location where the first bit set should be stored. Once the assembly language program was successfully compiled, we downloaded the program onto the PIC24 hardware successfully. The source code and pictures are in [Appendix C](#).

#### Task 4 – Assembly to C Example

In the 4th and final task of lab 5, we created an assembly project using the assembly language program provided in the lab document. Next, we executed the given program to obtain its results. We then wrote a C language program equivalent to the assembly language program provided in the lab document. Following a successful compile of the program, we downloaded our C language program onto the PIC24 device. The source code and pictures are in [Appendix D](#).

## Conclusion

We are now confident in our ability to translate basic PIC24 assembly language into C programming language and C programming language to PIC24 assembly language and simulate our results in the MPLAB Integrated Development Environment (IDE). In addition, we had success in downloading the C language programs and assembly language programs onto the PIC24 device in tasks 2, 3, and 4. The results of the simulations showed the equivalence of the C language programs and assembly language programs when translated. This lab provided an introduction C language and assembly language translation and allows for us to be confident in their use for programming the PIC24 device thereafter.

# Appendixes

## Appendix A – Task 1

Source Code – “task1.c”

```
#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=0xFF;

    u8_y=0x01;

    u16_a = 0xFFFF;

    u16_b = 0x0001;

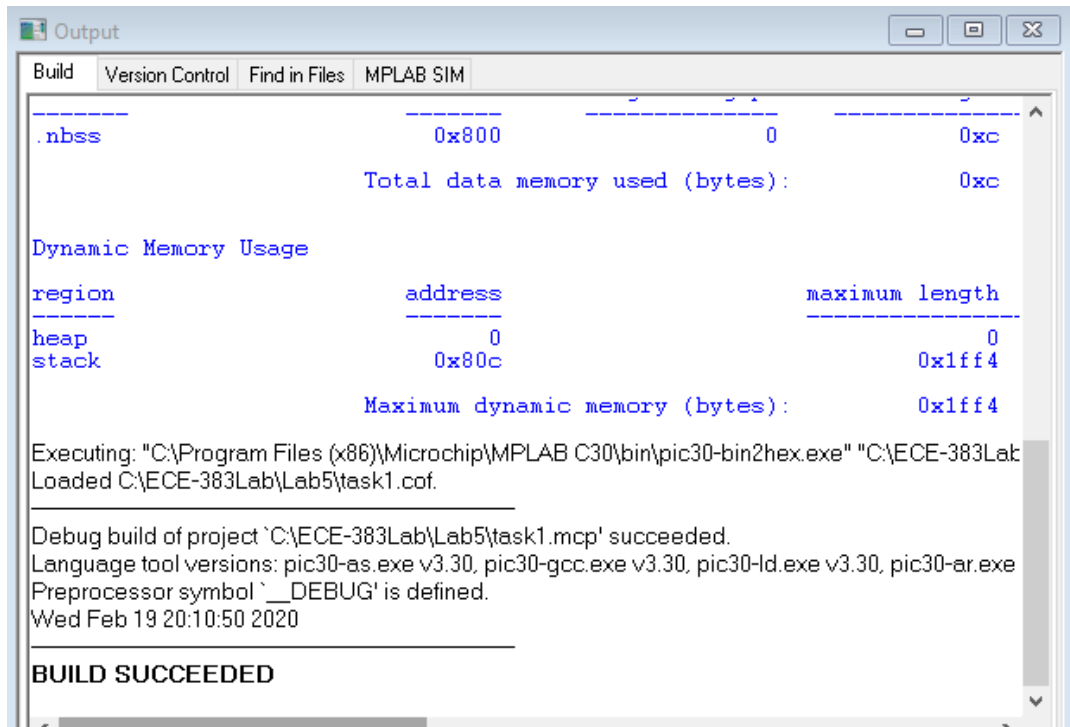
    u8_z=u8_x+u8_y;

    u16_d=(uint16) u8_x + (uint16) u8_y;

    u16_c=u16_a+u16_b;

}
```

Figure (1 - 6)

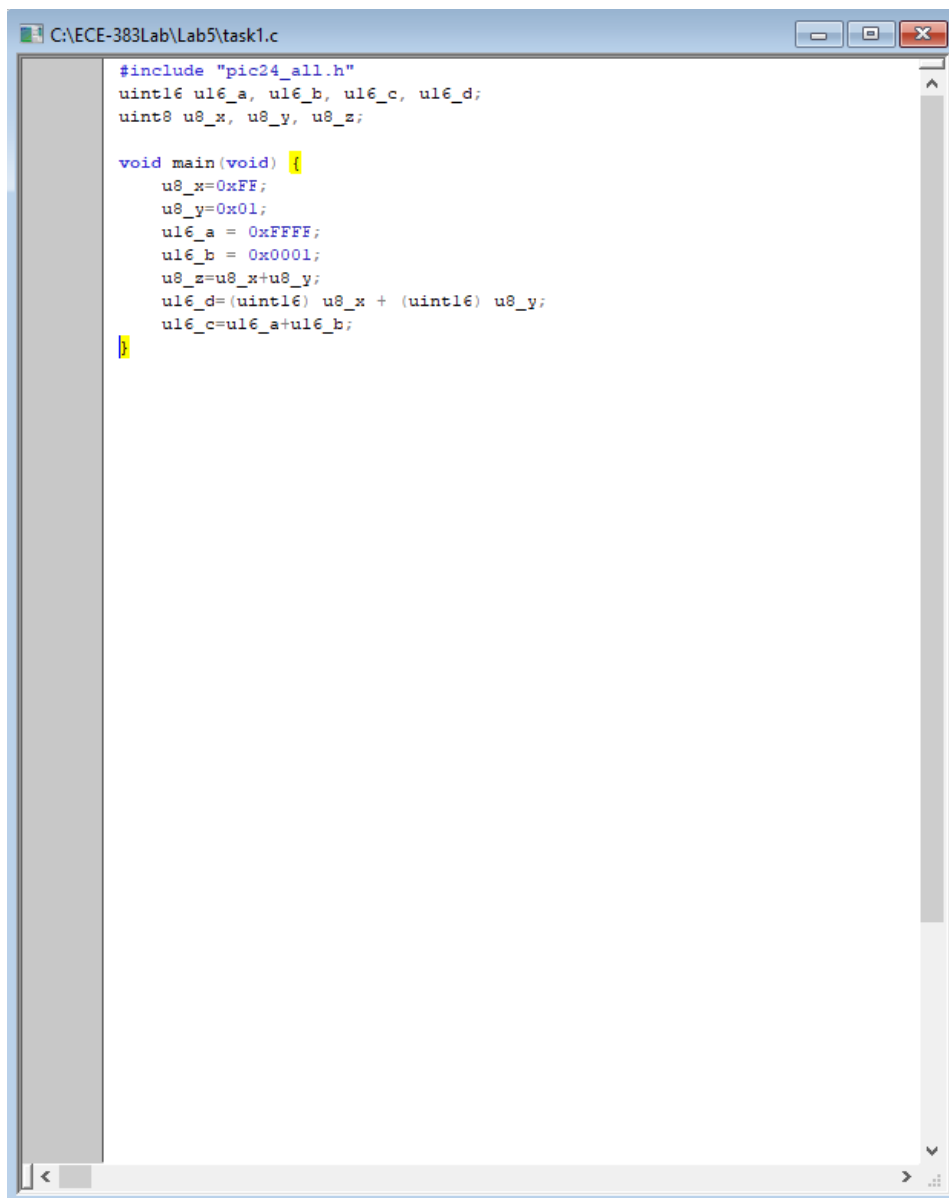


The screenshot shows the MPLAB IDE Output Window with the 'MPLAB SIM' tab selected. The output text is as follows:

```
-----  
.nbss                0x800                0                0xc  
Total data memory used (bytes):                0xc  
  
Dynamic Memory Usage  
-----  
region              address              maximum length  
-----  
heap                0                0  
stack              0x80c              0x1fff4  
Maximum dynamic memory (bytes):                0x1fff4  
  
Executing: "C:\Program Files (x86)\Microchip\MPLAB C30\bin\pic30-bin2hex.exe" "C:\ECE-383Lab\Lab5\task1.cof."  
Loaded C:\ECE-383Lab\Lab5\task1.cof.  
-----  
Debug build of project 'C:\ECE-383Lab\Lab5\task1.mcp' succeeded.  
Language tool versions: pic30-as.exe v3.30, pic30-gcc.exe v3.30, pic30-ld.exe v3.30, pic30-ar.exe  
Preprocessor symbol '__DEBUG' is defined.  
Wed Feb 19 20:10:50 2020  
-----  
BUILD SUCCEEDED
```

Figure 1 Output Window





```
#include "pic24_all.h"
uint16 ul6_a, ul6_b, ul6_c, ul6_d;
uint8 u8_x, u8_y, u8_z;

void main(void) {
    u8_x=0xFF;
    u8_y=0x01;
    ul6_a = 0xFFFF;
    ul6_b = 0x0001;
    u8_z=u8_x+u8_y;
    ul6_d=(uint16) u8_x + (uint16) u8_y;
    ul6_c=ul6_a+ul6_b;
}
```

Figure 2 task1.c

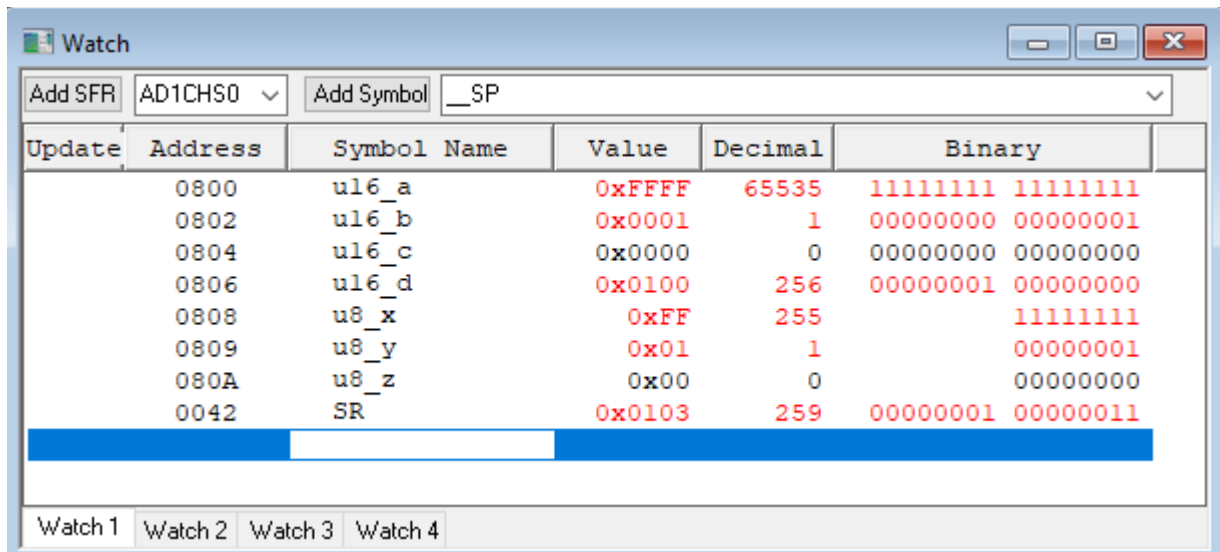


Figure 3 Watch Window

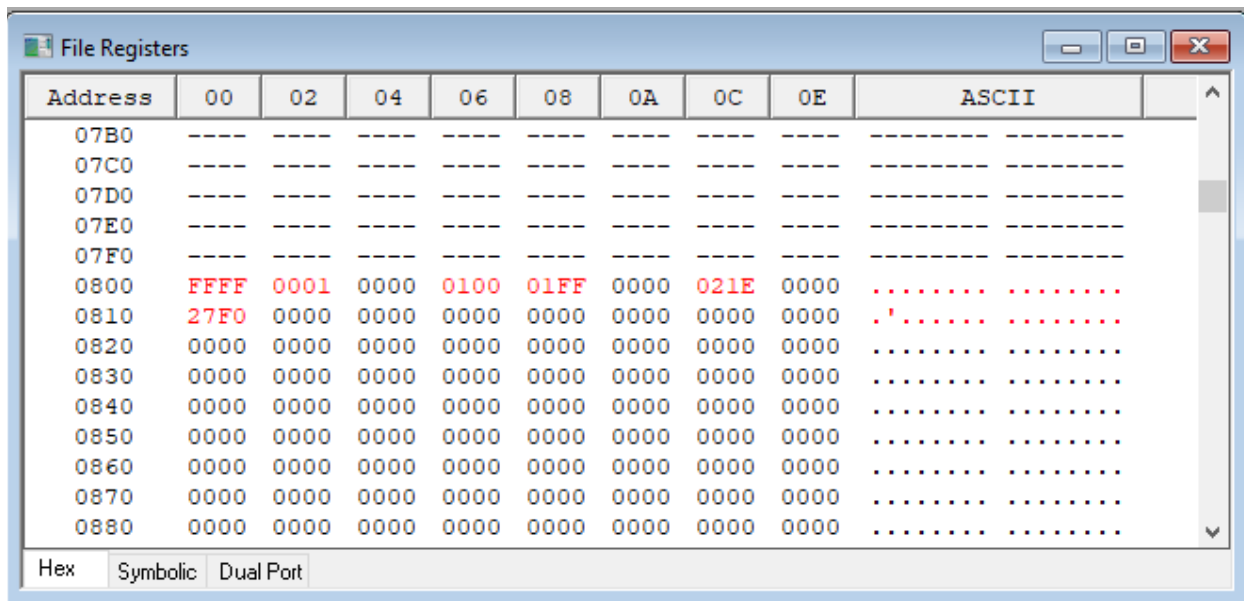


Figure 4 File Registers

Address	SFR Name	Hex
	C1BUFNT1	x0000
	C1BUFNT2	x0000
	C1RXF0EID	x0000
	C1RXF0SID	x0000
	C1RXM0EID	x0000
	C1RXM0SID	x0000
	C1RXM1EID	x0000
	C1RXM1SID	x0000
	PMDOUT1	x0000
0000	WREG0	x0000
0002	WREG1	xFFFF
0004	WREG2	x0000
0006	WREG3	x0000
0008	WREG4	x0000
000A	WREG5	x0000
000C	WREG6	x0000
000E	WREG7	x0000
0010	WREG8	x0000

Figure 5 Special Function Registers

Line	Address	Opcode	Disassembly
254	001FA	000290	_DefaultInterrupt
255	001FC	000290	_DefaultInterrupt
256	001FE	000290	_DefaultInterrupt
257	00200	2080CF	mov.w #0x80c,0x001e
258	00202	227F0E	mov.w #0x27f0,0x001c
259	00204	88010E	mov.w 0x001c,0x0020
260	00206	000000	nop
261	00208	07000C	rcall 0x000222
262	0020A	202C80	mov.w #0x2c8,0x0000
263	0020C	200001	mov.w #0x0,0x0002
264	0020E	070011	rcall 0x000232
265	00210	200000	mov.w #0x0,0x0000
266	00212	E00000	cp0.w 0x0000
267	00214	320002	bra z, 0x00021a
268	00216	020000	call 0x000000
269	00218	000000	nop
270	0021A	020294	call 0x000294
271	0021C	000000	nop
272	0021E	DA4000	ReservedBR
273	00220	FE0000	reset
274	00222	A94044	bclr.b 0x0044,#2
275	00224	200000	mov.w #0x0,0x0000
276	00226	E00000	cp0.w 0x0000
277	00228	320003	bra z, 0x000230
278	0022A	200000	mov.w #0x0,0x0000
279	0022C	8801A0	mov.w 0x0000,0x0034
280	0022E	A84044	bset.b 0x0044,#2
281	00230	060000	return
282	00232	880191	mov.w 0x0002,0x0032
283	00234	780080	mov.w 0x0000,0x0002
284	00236	EB0000	clr.w 0x0000
285	00238	370015	bra 0x000264
286	0023A	4080E2	addc.w 0x0002,#2,0x0002
287	0023C	B4A032	addc.w 0x0032
288	0023E	BA0191	tblrdl.w [0x0002],0x0006
289	00240	4080E2	addc.w 0x0002,#2,0x0002
290	00242	B4A032	addc.w 0x0032
291	00244	BA0291	tblrdl.w [0x0002],0x000a
292	00246	4080E2	addc.w 0x0002,#2,0x0002
293	00248	B4A032	addc.w 0x0032
294	0024A	EB0000	clr.w 0x0000

Figure 6 Program Memory

## Appendix B – Task 2

Source code – “task2.c”

```
#include "pic24_all.h"

uint16 check_val = 0;

uint8 ones_count = 0, first_one = 0, count = 0, set = 0;

void main (void) {

    for(check_val = 0xF508; check_val != 0; check_val = check_val / 2) {

        if (check_val & 0x0001 == 0x0001) {

            ones_count++;

            if(set == 0) {

                first_one = count;

                set = 1;

            }

        }

        count++;

    }

}
```

Figure (7 - 18)

***Simulated (7 - 12)***

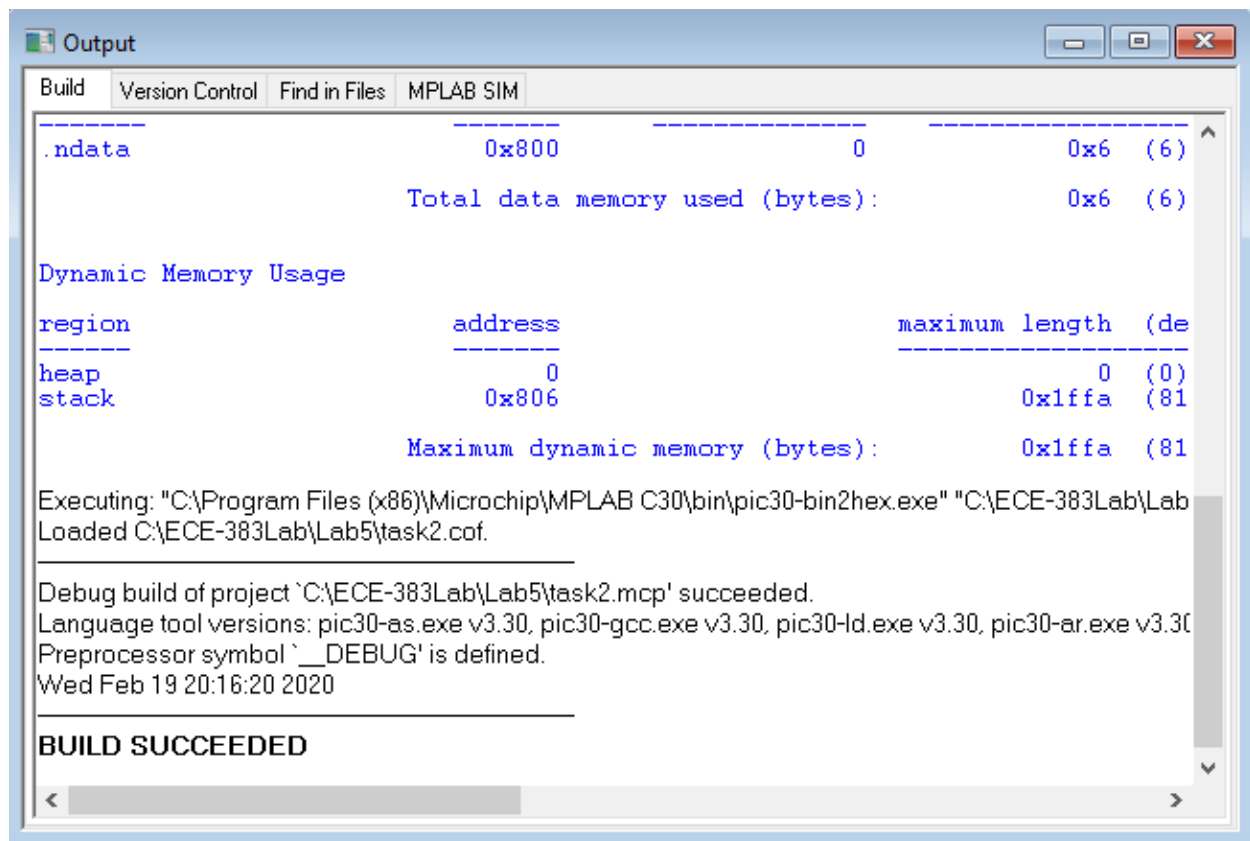


Figure 7 Output Window

The image shows a screenshot of a C code editor window. The title bar at the top reads "C:\ECE-383Lab\Lab5\task2.c". The code is as follows:

```
#include "pic24_all.h"

uint16 check_val = 0;
uint8 ones_count = 0, first_one = 0, count = 0, set = 0;

void main (void) {
    for(check_val = 0xF508; check_val != 0; check_val = check_val / 2) {
        if (check_val & 0x0001 == 0x0001) {
            ones_count++;
            if(set == 0) {
                first_one = count;
                set = 1;
            }
        }
        count++;
    }
}
```

The code defines a `main` function that iterates through the bits of the hexadecimal value `0xF508` by repeatedly dividing it by 2. For each iteration, it checks if the least significant bit is 1. If it is, it increments `ones_count`. If this is the first time a 1 is encountered, it sets `first_one` to the current `count` and marks `set` as 1. The `count` variable is incremented for every bit checked, regardless of whether it is 1 or 0. The loop terminates when `check_val` becomes 0.

Figure 8 task2.c

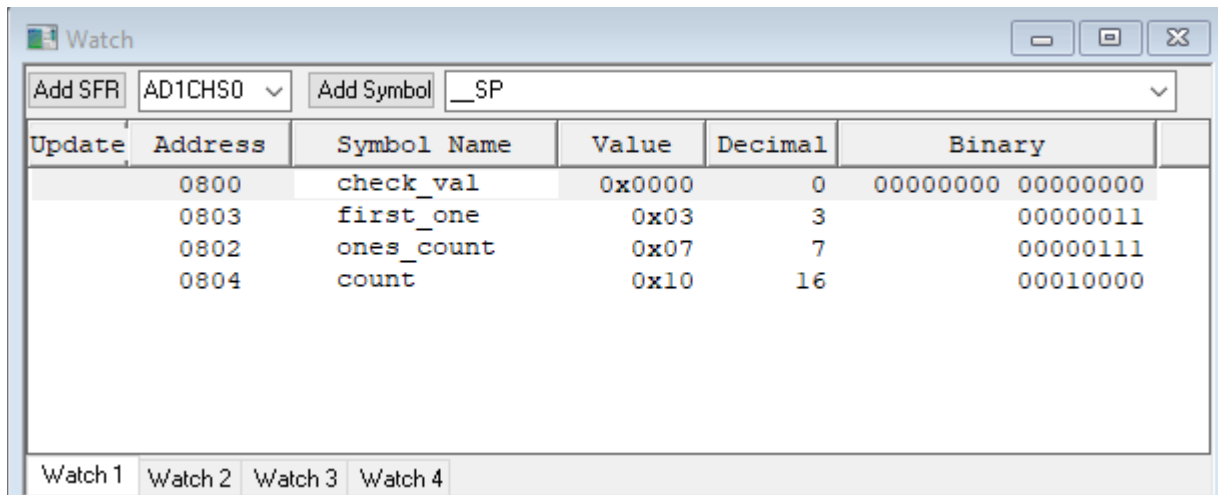


Figure 9 Watch Window

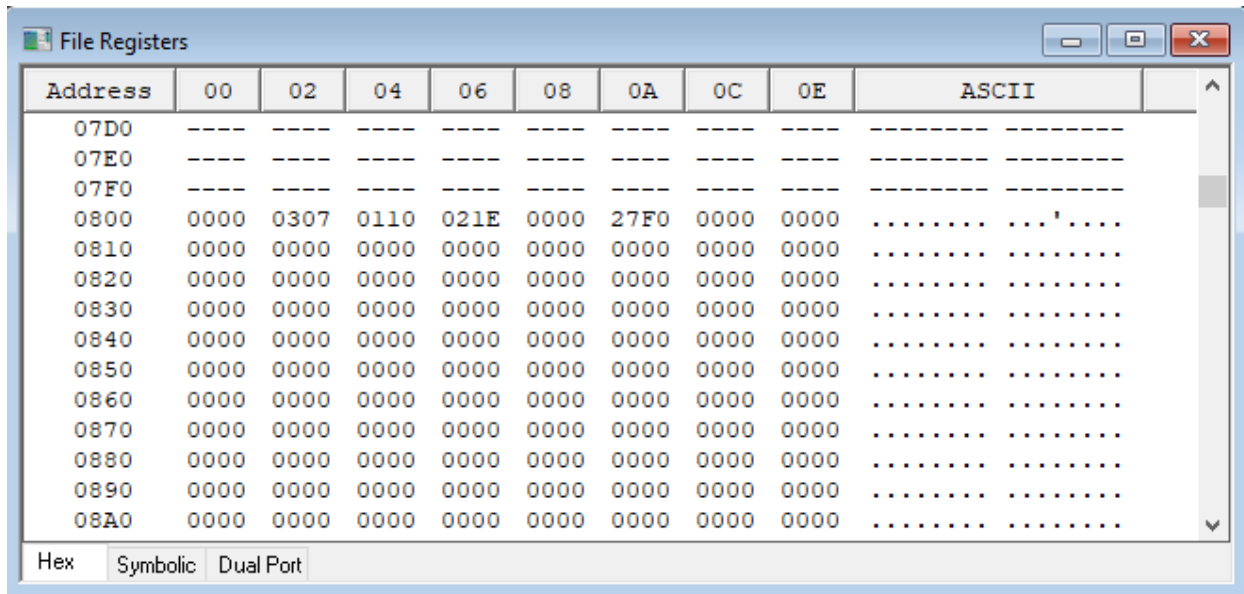
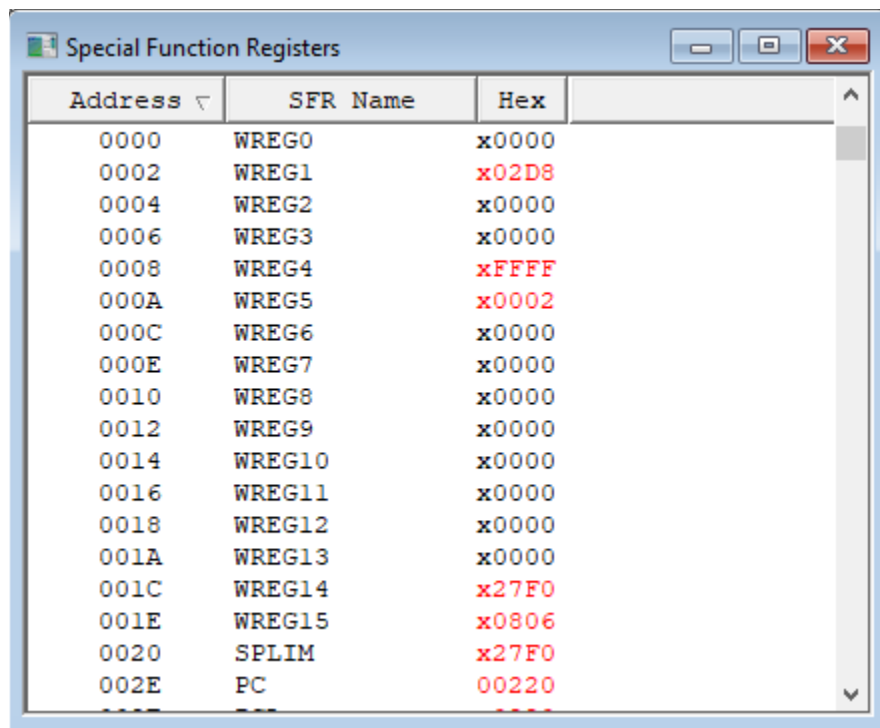


Figure 10 File Registers

Program Memory				
	Line	Address	Opcode	Disassembly
	257	00200	20806F	mov.w #0x806,0x001e
	258	00202	227F0E	mov.w #0x27f0,0x001c
	259	00204	88010E	mov.w 0x001c,0x0020
	260	00206	000000	nop
	261	00208	07000C	rcall 0x000222
	262	0020A	202CE0	mov.w #0x2ce,0x0000
	263	0020C	200001	mov.w #0x0,0x0002
	264	0020E	070011	rcall 0x000232
	265	00210	200000	mov.w #0x0,0x0000
	266	00212	E00000	cp0.w 0x0000
	267	00214	320002	bra z, 0x00021a
	268	00216	020000	call 0x000000
	269	00218	000000	nop
	270	0021A	020294	call 0x000294
	271	0021C	000000	nop
	272	0021E	DA4000	ReservedBR
→	273	00220	FE0000	reset
	274	00222	A94044	bclr.b 0x0044,#2
	275	00224	200000	mov.w #0x0,0x0000
	276	00226	E00000	cp0.w 0x0000
	277	00228	320003	bra z, 0x000230
	278	0022A	200000	mov.w #0x0,0x0000
	279	0022C	8801A0	mov.w 0x0000,0x0034
	280	0022E	A84044	bset.b 0x0044,#2
	281	00230	060000	return
	282	00232	880191	mov.w 0x0002,0x0032
	283	00234	780080	mov.w 0x0000,0x0002
	284	00236	EB0000	clr.w 0x0000
	285	00238	370015	bra 0x000264
	286	0023A	4080E2	add.w 0x0002,#2,0x0002
	287	0023C	B4A032	addc.w 0x0032

Figure 11 Program Memory





Address	SFR Name	Hex
0000	WREG0	x0000
0002	WREG1	x02D8
0004	WREG2	x0000
0006	WREG3	x0000
0008	WREG4	xFFFF
000A	WREG5	x0002
000C	WREG6	x0000
000E	WREG7	x0000
0010	WREG8	x0000
0012	WREG9	x0000
0014	WREG10	x0000
0016	WREG11	x0000
0018	WREG12	x0000
001A	WREG13	x0000
001C	WREG14	x27F0
001E	WREG15	x0806
0020	SPLIM	x27F0
002E	PC	00220

Figure 12 Special Function Register

### On PIC24 Microcontroller (13 - 18)

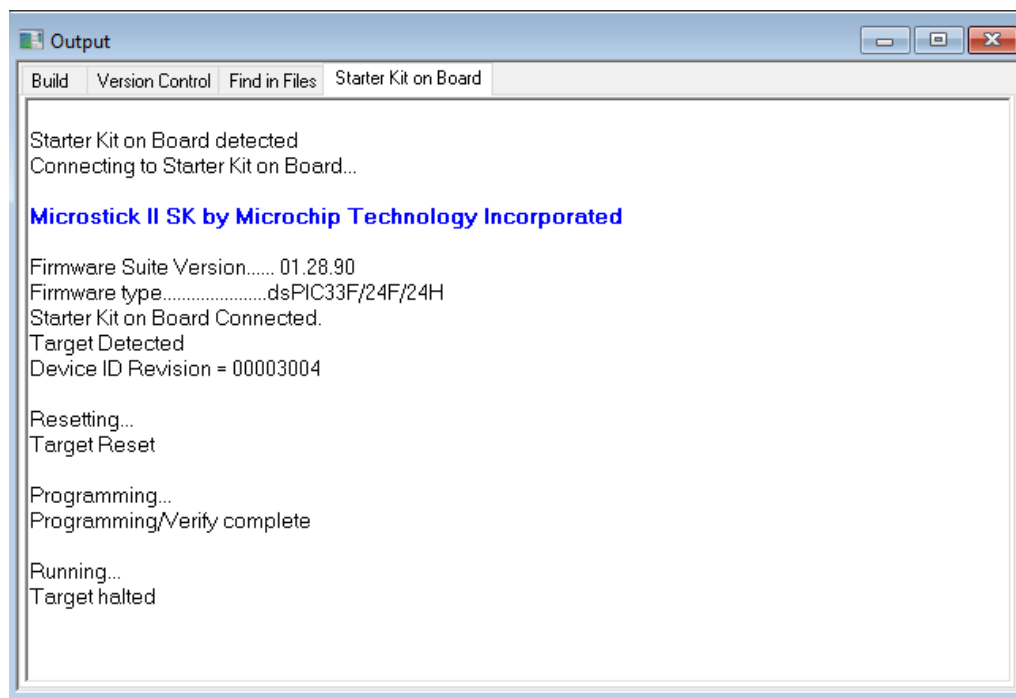
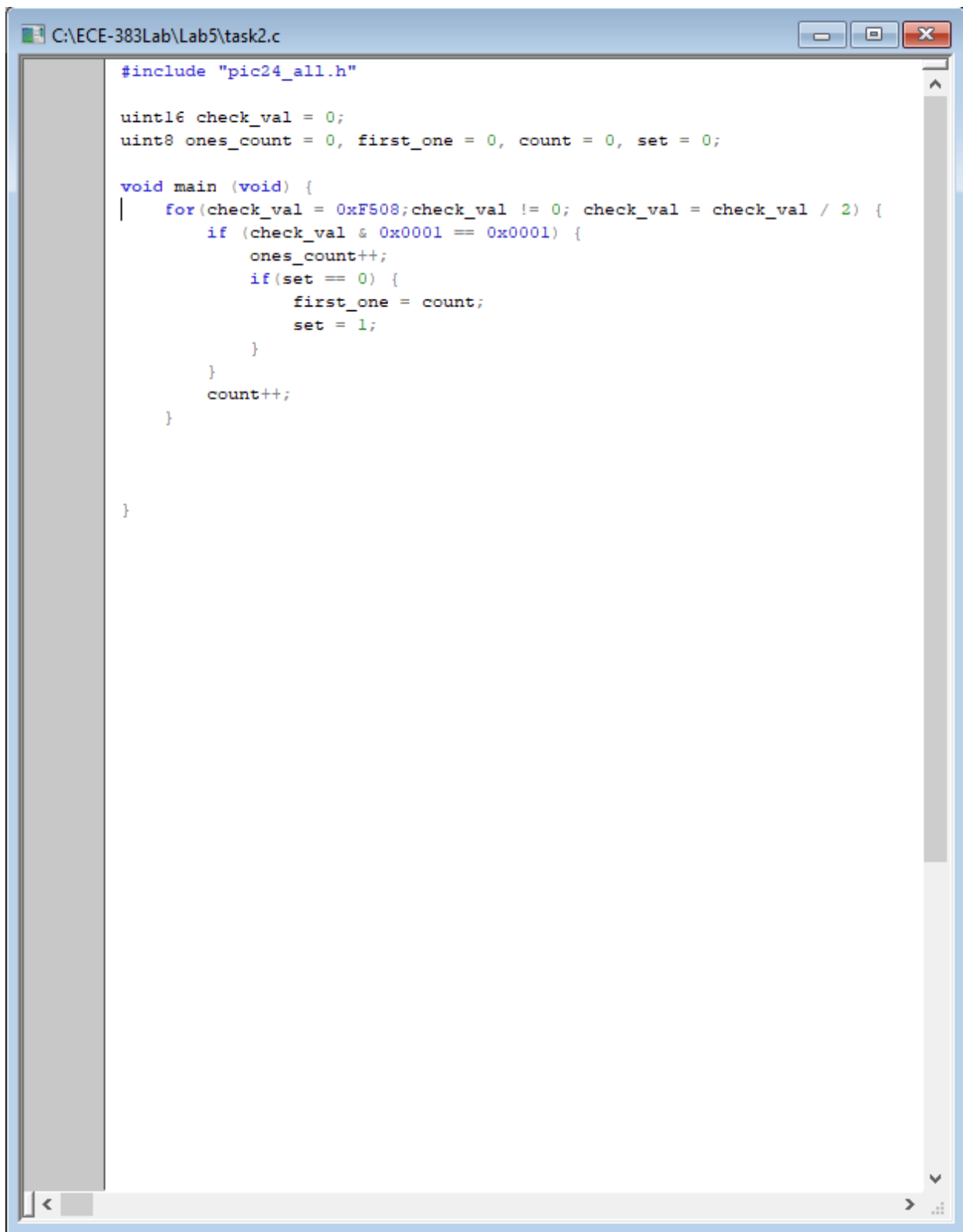


Figure 13 Output Window



```
#include "pic24_all.h"

uint16 check_val = 0;
uint8 ones_count = 0, first_one = 0, count = 0, set = 0;

void main (void) {
    for(check_val = 0xF508; check_val != 0; check_val = check_val / 2) {
        if (check_val & 0x0001 == 0x0001) {
            ones_count++;
            if(set == 0) {
                first_one = count;
                set = 1;
            }
        }
        count++;
    }
}
```

Figure 14 task2.c

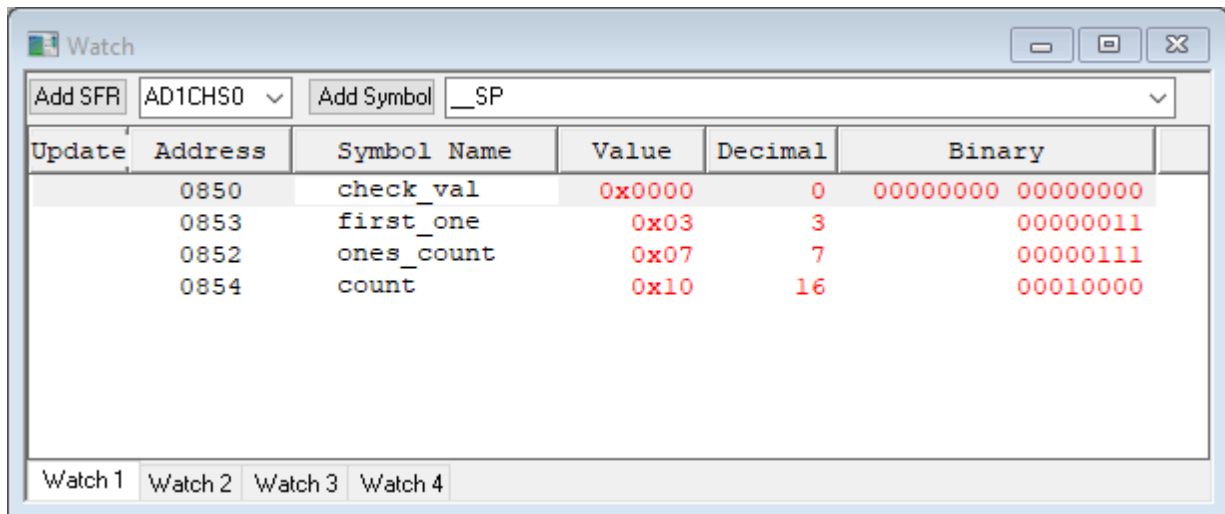


Figure 15 Watch Window

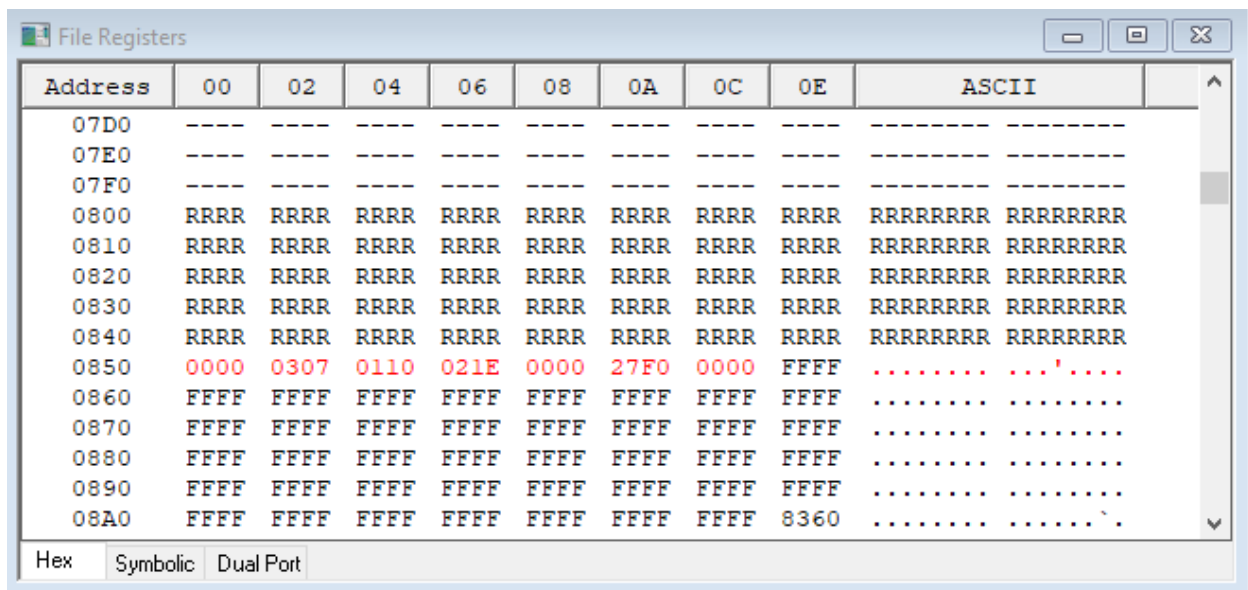


Figure 16 File Registers

Line	Address	Opcode	Disassembly
259	00204	88010E	mov.w 0x001c,0x0020
260	00206	000000	nop
261	00208	07000C	rcall 0x000222
262	0020A	202CE0	mov.w #0x2ce,0x0000
263	0020C	200001	mov.w #0x0,0x0002
264	0020E	070011	rcall 0x000232
265	00210	200000	mov.w #0x0,0x0000
266	00212	E00000	cp0.w 0x0000
267	00214	320002	bra z, 0x00021a
268	00216	020000	call 0x000000
269	00218	000000	nop
270	0021A	020294	call 0x000294
271	0021C	000000	nop
272	0021E	DA4000	ReservedBR
273	00220	FE0000	reset
274	00222	A94044	bclr.b 0x0044,#2
275	00224	200000	mov.w #0x0,0x0000
276	00226	E00000	cp0.w 0x0000
277	00228	320003	bra z, 0x000230
278	0022A	200000	mov.w #0x0,0x0000
279	0022C	8801A0	mov.w 0x0000,0x0034
280	0022E	A84044	bset.b 0x0044,#2
281	00230	060000	return
282	00232	880191	mov.w 0x0002,0x0032
283	00234	780080	mov.w 0x0000,0x0002
284	00236	EB0000	clr.w 0x0000
285	00238	370015	bra 0x000264
286	0023A	4080E2	add.w 0x0002,#2,0x0002
287	0023C	B4A032	addc.w 0x0032
288	0023E	BA0191	tblrdl.w [0x0002],0x0006
289	00240	4080E2	add.w 0x0002,#2,0x0002

Figure 17 Program Memory

Address	SFR Name	Hex
0000	WREG0	x0000
0002	WREG1	x02D8
0004	WREG2	x0000
0006	WREG3	x0000
0008	WREG4	xFFFF
000A	WREG5	x0002
000C	WREG6	x0000
000E	WREG7	x0000
0010	WREG8	x0000
0012	WREG9	x0000
0014	WREG10	x0000
0016	WREG11	x0000
0018	WREG12	x0000
001A	WREG13	x0000
001C	WREG14	x27F0
001E	WREG15	x0856
0020	SPLIM	x27F0
002E	PC	00220

Figure 18 Special Function Registers

## Appendix C – Task 3

Source code – “task3.s”

```
.include "p24Hxxxx.inc"

.global __reset

.bss

check_val:      .space 2
ones_count:    .space 1
first_one:      .space 1
count:          .space 1
set:            .space 1

.text

__reset:

    mov #__SP_init, w15
    mov #__SPLIM_init, W0
    mov W0, SPLIM

;-----C code-----
;uint16 check_val = 0;
;uint8 ones_count = 0, first_one = 0, count = 0, set = 0;
;void main (void) {
;    for(check_val = 0xFFFF;check_val != 0; check_val = check_val / 2) {
;        if (check_val & 0x0001 == 0x0001) {
;            ones_count++;
;            if(set == 0) {
```

```

;                                first_one = count;
;                                set = 1;
;                                }
;                                }
;                                count++;
;                                }
;                                }
;                                }

;-----code start here-----

; check_val = 0xF508
    mov #0xF508, w0
    mov wreg, check_val

;ones_count = 0
    clr.b ones_count

;first_one = 0
    clr.b first_one

;count = 0
    clr.b count

;set = 0
    clr.b set

top:
    mov check_val w0;
    bra Z, end_loop;
    and #0x0001, w0;

```

```

bra Z, end_if1;

    inc.b ones_count

    mov.b set, wreg

    bra NZ, end_if2

        mov.b count, wreg

        mov.b wreg, first_one

        mov.b #0x01, w0

        mov.b wreg, set

    end_if2:

end_if1:

inc.b count

lsr check_val

bra top

end_loop:

done: goto done

.end

```

Figure (19 - 30)

**Simulated (19 - 24)**

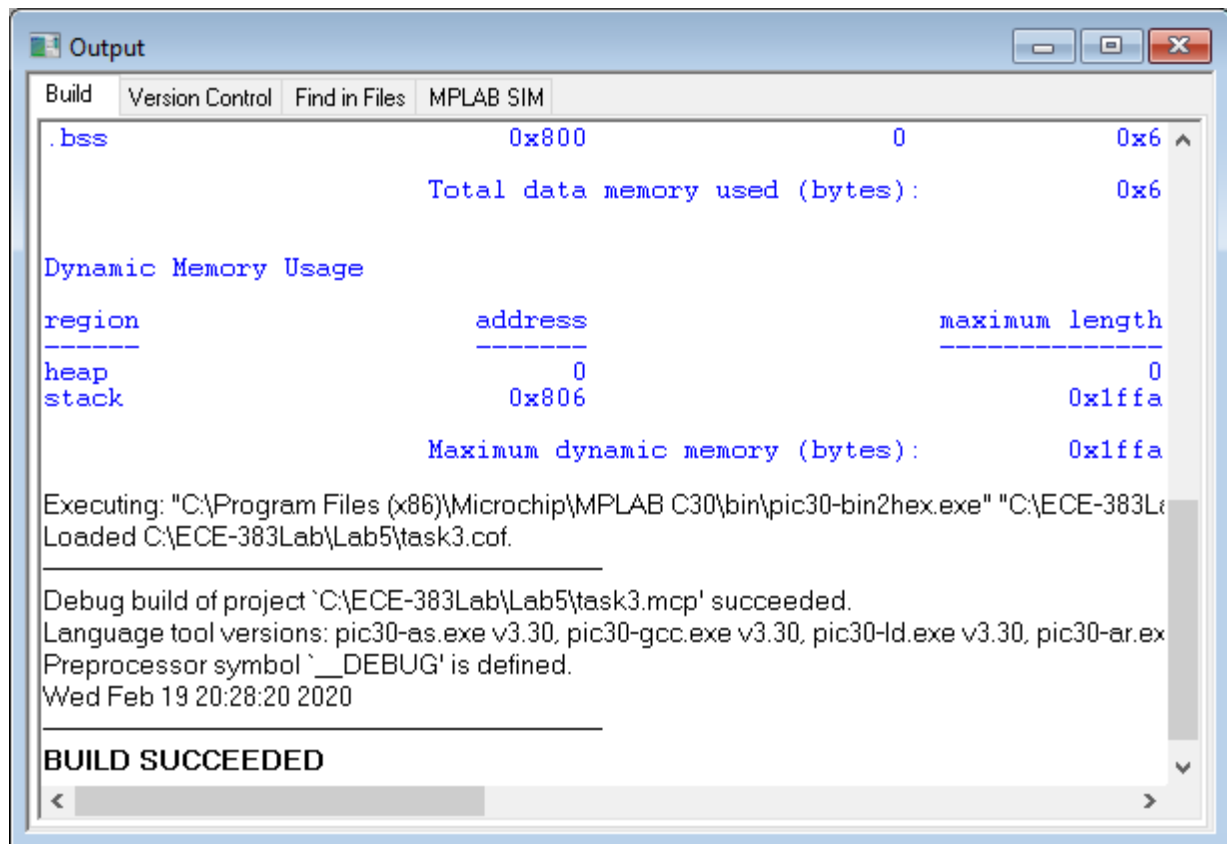


Figure 19 Output Window



```

C:\ECE-383Lab\Lab5\task3.s
.include "p24Hxxxx.inc"
.global __reset
.bss
check_val:    .space 2
ones_count:   .space 1
first_one:    .space 1
count:        .space 1
set:          .space 1

.text
__reset:
    mov #__SP_init, w15
    mov #__SPLIM_init, W0
    mov W0, SPLIM

;-----C code-----
;uint16 check_val = 0;
;uint8 ones_count = 0, first_one = 0, count = 0, set = 0;

;void main (void) {
;    for(check_val = 0xFFFF;check_val != 0; check_val = check_val / 2) {
;        if (check_val & 0x0001 == 0x0001) {
;            ones_count++;
;            if(set == 0) {
;                first_one = count;
;                set = 1;
;            }
;        }
;        count++;
;    }
;}

;-----code start here-----

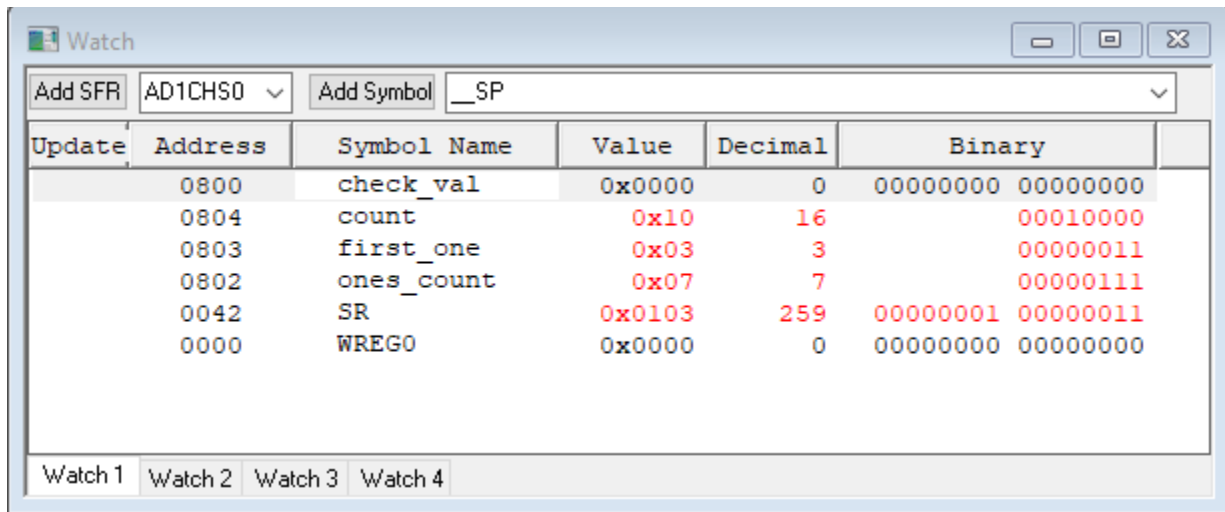
; check_val = 0xF508
;mov #0xF508, w0
;mov wreg, check_val
;ones_count = 0
;clr.b ones_count
;first_one = 0
;clr.b first_one
;count = 0
;clr.b count
;set = 0
;clr.b set

top:
    mov check_val, w0;
    bra Z, end_loop;
    and #0x0001, w0;
    bra Z, end_if1;
    inc.b ones_count
    mov.b set, wreg
    bra NZ, end_if2
    mov.b count, wreg
    mov.b wreg, first_one
    mov.b #0x01, w0
    mov.b wreg, set
    end_if2:
    end_if1:
    inc.b count
    lsr check_val
    bra top

end_loop:
    Hone: goto done
.end

```

Figure 20 task3.s

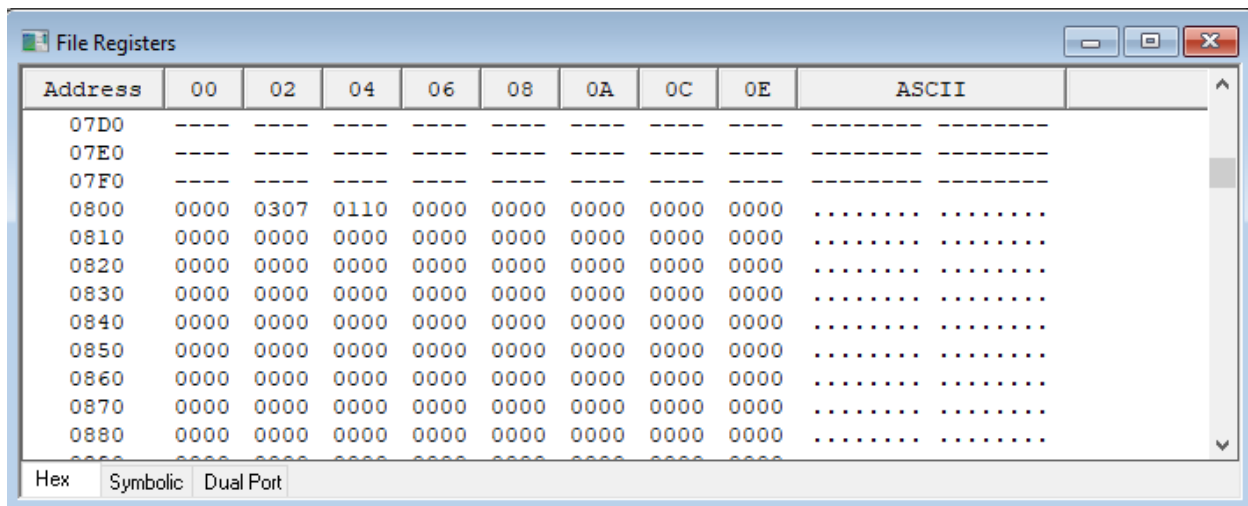


Watch window showing memory addresses and symbol values. The window has a title bar 'Watch' and a toolbar with minimize, maximize, and close buttons. Below the toolbar is a section with 'Add SFR' (set to AD1CHS0) and 'Add Symbol' (set to \_\_SP). The main table has columns: Update, Address, Symbol Name, Value, Decimal, and Binary. The data is as follows:

Update	Address	Symbol Name	Value	Decimal	Binary
	0800	check_val	0x0000	0	00000000 00000000
	0804	count	0x10	16	00010000
	0803	first_one	0x03	3	00000011
	0802	ones_count	0x07	7	00000111
	0042	SR	0x0103	259	00000001 00000011
	0000	WREG0	0x0000	0	00000000 00000000

At the bottom, there are tabs for Watch 1, Watch 2, Watch 3, and Watch 4, with Watch 1 selected.

Figure 21 watch window



File Registers window showing memory addresses and ASCII values. The window has a title bar 'File Registers' and a toolbar with minimize, maximize, and close buttons. The main table has columns: Address, 00, 02, 04, 06, 08, 0A, 0C, 0E, and ASCII. The data is as follows:

Address	00	02	04	06	08	0A	0C	0E	ASCII
07D0	----	----	----	----	----	----	----	----	-----
07E0	----	----	----	----	----	----	----	----	-----
07F0	----	----	----	----	----	----	----	----	-----
0800	0000	0307	0110	0000	0000	0000	0000	0000	.....
0810	0000	0000	0000	0000	0000	0000	0000	0000	.....
0820	0000	0000	0000	0000	0000	0000	0000	0000	.....
0830	0000	0000	0000	0000	0000	0000	0000	0000	.....
0840	0000	0000	0000	0000	0000	0000	0000	0000	.....
0850	0000	0000	0000	0000	0000	0000	0000	0000	.....
0860	0000	0000	0000	0000	0000	0000	0000	0000	.....
0870	0000	0000	0000	0000	0000	0000	0000	0000	.....
0880	0000	0000	0000	0000	0000	0000	0000	0000	.....

At the bottom, there are tabs for Hex, Symbolic, and Dual Port, with Hex selected.

Figure 22 File Register

Special Function Registers		
Address	SFR Name	Hex
0000	WREG0	x0000
0002	WREG1	x0000
0004	WREG2	x0000
0006	WREG3	x0000
0008	WREG4	x0000
000A	WREG5	x0000
000C	WREG6	x0000
000E	WREG7	x0000
0010	WREG8	x0000
0012	WREG9	x0000
0014	WREG10	x0000
0016	WREG11	x0000
0018	WREG12	x0000
001A	WREG13	x0000
001C	WREG14	x0000
001E	WREG15	x0806
0020	SPLIM	x27F0

Figure 23 Special Function Registers

Program Memory

Line	Address	Opcode	Disassembly	
255	001FC	00025E	_DefaultInterrupt	
256	001FE	00025E	_DefaultInterrupt	
257	00200	880191	mov.w 0x0002,0x0032	
258	00202	780080	mov.w 0x0000,0x0002	
259	00204	EB0000	clr.w 0x0000	
260	00206	370015	bra 0x000232	
261	00208	4080E2	add.w 0x0002,#2,0x0002	
262	0020A	B4A032	addc.w 0x0032	
263	0020C	BA0191	tblrdl.w [0x0002],0x0006	
264	0020E	4080E2	add.w 0x0002,#2,0x0002	
265	00210	B4A032	addc.w 0x0032	
266	00212	BA0291	tblrdl.w [0x0002],0x000a	
267	00214	4080E2	add.w 0x0002,#2,0x0002	
268	00216	B4A032	addc.w 0x0032	
269	00218	EB0200	clr.w 0x0008	
270	0021A	DE2B47	lsr 0x000a,#7,0x000c	
271	0021C	B207F5	and.w #0x7f,0x000a	
272	0021E	E12C60	cp.b 0x000a,#0	
273	00220	3A0004	bra nz, 0x00022a	
274	00222	EB5900	clr.b [0x0004++]	
275	00224	E90183	dec.w 0x0006,0x0006	
276	00226	3EFFFF	bra gtu, 0x000222	
277	00228	370004	bra 0x000232	
278	0022A	E12861	cp.w 0x000a,#1	
279	0022C	320001	bra z, 0x000230	
280	0022E	EB8200	setm.w 0x0008	
281	00230	070004	rcall 0x00023a	
282	00232	BA0111	tblrdl.w [0x0002],0x0004	
283	00234	E00002	cp0.w 0x0004	
284	00236	3AFFE8	bra nz, 0x000208	
285	00238	060000	return	
286	0023A	BA5931	tblrdl.b [0x0002++],[0x0004++]	
287	0023C	E90183	dec.w 0x0006,0x0006	
288	0023E	32000C	bra z, 0x000258	
289	00240	BA5921	tblrdl.b [0x0002--],[0x0004++]	
290	00242	E90183	dec.w 0x0006,0x0006	
291	00244	320008	bra z, 0x000256	
292	00246	E00004	cp0.w 0x0008	
293	00248	3A0003	bra nz, 0x000250	
294	0024A	4080E2	add.w 0x0002,#2,0x0002	
295	0024C	B4A032	addc.w 0x0032	
296	0024E	37FFFF	bra 0x00023a	
Opcode Hex	Machine	Symbolic	PSV Mixed	PSV Data

Figure 24 Program Memory

## On PIC24 Microcontroller (25 - 30)

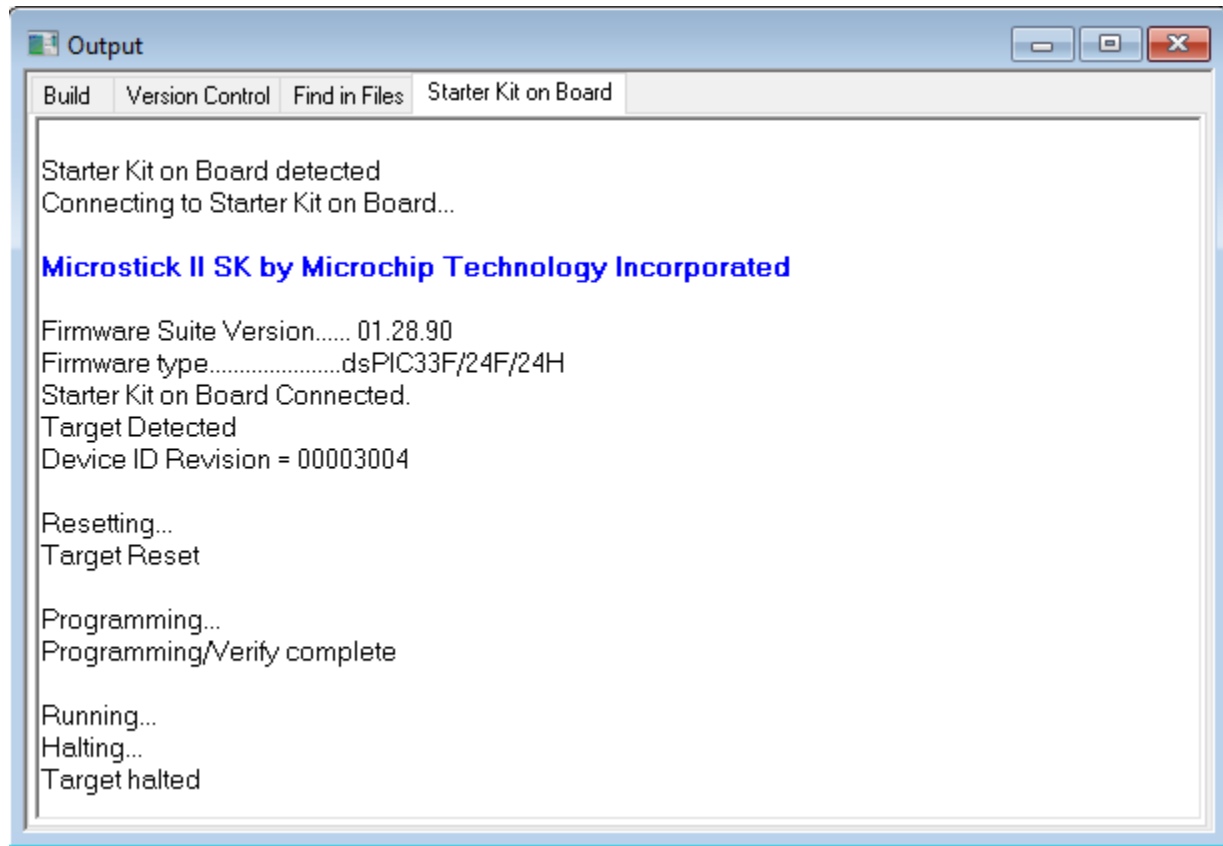


Figure 25 Output Window

```
C:\ECE-383Lab\Lab5\task3.s

.include "p24Hxxxx.inc"
.global __reset
.bss
check_val:      .space 2
ones_count:     .space 1
first_one:      .space 1
count:          .space 1
set:            .space 1

.text
__reset:
    mov #__SP_init, w15
    mov #__SPLIM_init, W0
    mov W0, SPLIM

;-----C code-----
;uint16 check_val = 0;
;uint8 ones_count = 0, first_one = 0, count = 0, set = 0;

;void main(void) {
;    for(check_val = 0xFFFF; check_val != 0; check_val = check_val / 2) {
;        if (check_val & 0x0001 == 0x0001) {
;            ones_count++;
;            if (set == 0) {
;                first_one = count;
;                set = 1;
;            }
;        }
;        count++;
;    }
;}

;-----code start here-----

; check_val = 0xF508
    mov #0xF508, w0
    mov wreg, check_val
;ones_count = 0
    clr.b ones_count
;first_one = 0
    clr.b first_one
;count = 0
    clr.b count
;set = 0
    clr.b set

top:
    mov check_val, w0;
    bra Z, end_loop;
    and #0x0001, w0;
    bra Z, end_if1;
    inc.b ones_count
    mov.b set, wreg
    bra NZ, end_if2
    mov.b count, wreg
    mov.b wreg, first_one
    mov.b #0x01, w0
    mov.b wreg, set
    end_if2:
    end_if1:
    inc.b count
    lsr check_val
    bra top

end_loop:
    Hone: goto done
.end
```

Figure 26 task3.s

Update	Address	Symbol Name	Value	Decimal	Binary
	0850	check_val	0x0000	0	00000000 00000000
	0854	count	0x10	16	00010000
	0853	first_one	0x03	3	00000011
	0852	ones_count	0x07	7	00000111
	0042	SR	0x0103	259	00000001 00000011
	0000	WREG0	0x0000	0	00000000 00000000

Watch 1 Watch 2 Watch 3 Watch 4

Figure 27 Watch Window

Address	00	02	04	06	08	0A	0C	0E	ASCII
0800	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRRRRRR RRRRRRRR
0810	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRRRRRR RRRRRRRR
0820	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRRRRRR RRRRRRRR
0830	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRRRRRR RRRRRRRR
0840	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRR	RRRRRRRR RRRRRRRR
0850	0000	0307	0110	FFFF	FFFF	FFFF	FFFF	FFFF	.....
0860	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	.....
0870	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	.....
0880	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	.....
0890	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	.....
08A0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	8360	.....
08B0	E05D	8D71	5747	40B0	FFFF	FFFF	FFFF	FFFF	] .q.GW.@ .....

Hex Symbolic Dual Port

Figure 28 File Registers

Address	SFR Name	Hex
0000	WREG0	x0000
0002	WREG1	x0000
0004	WREG2	x0000
0006	WREG3	x0000
0008	WREG4	x0000
000A	WREG5	x0000
000C	WREG6	x0000
000E	WREG7	x0000
0010	WREG8	x0000
0012	WREG9	x0000
0014	WREG10	x0000
0016	WREG11	x0000
0018	WREG12	x0000
001A	WREG13	x0000
001C	WREG14	x0000
001E	WREG15	x0856
0020	SPLIM	x27F0

Figure 29 Special Function Registers

Program Memory				
	Line	Address	Opcode	Disassembly
	256	001FE	00025E	_DefaultInterrupt
	257	00200	880191	mov.w 0x0002,0x0032
	258	00202	780080	mov.w 0x0000,0x0002
	259	00204	EB0000	clr.w 0x0000
	260	00206	370015	bra 0x000232
	261	00208	4080E2	add.w 0x0002,#2,0x0002
	262	0020A	B4A032	addc.w 0x0032
	263	0020C	BA0191	tblrdl.w [0x0002],0x0006
	264	0020E	4080E2	add.w 0x0002,#2,0x0002
	265	00210	B4A032	addc.w 0x0032
	266	00212	BA0291	tblrdl.w [0x0002],0x000a
	267	00214	4080E2	add.w 0x0002,#2,0x0002
	268	00216	B4A032	addc.w 0x0032
	269	00218	EB0200	clr.w 0x0008
	270	0021A	DE2B47	lsr 0x000a,#7,0x000c
	271	0021C	B207F5	and.w #0x7f,0x000a
	272	0021E	E12C60	cp.b 0x000a,#0
	273	00220	3A0004	bra nz, 0x00022a
	274	00222	EB5900	clr.b [0x0004++]
	275	00224	E90183	dec.w 0x0006,0x0006
	276	00226	3EFFFF	bra gtu, 0x000222
	277	00228	370004	bra 0x000232
	278	0022A	E12861	cp.w 0x000a,#1
	279	0022C	320001	bra z, 0x000230
	280	0022E	EB8200	setm.w 0x0008
	281	00230	070004	rcall 0x00023a
	282	00232	BA0111	tblrdl.w [0x0002],0x0004
	283	00234	E00002	cp0.w 0x0004
	284	00236	3AFFE8	bra nz, 0x000208
	285	00238	060000	return
	286	0023A	BA5931	tblrdl.b [0x0002++],[0x0004++]
	287	0023C	E90183	dec.w 0x0006,0x0006
	288	0023E	32000C	bra z, 0x000258
	289	00240	BA5921	tblrdl.b [0x0002--],[0x0004++]
	290	00242	E90183	dec.w 0x0006,0x0006
	291	00244	320008	bra z, 0x000256
	292	00246	E00004	cp0.w 0x0008
	293	00248	3A0003	bra nz, 0x000250
	294	0024A	4080E2	add.w 0x0002,#2,0x0002
	295	0024C	B4A032	addc.w 0x0032
	296	0024E	37FFF5	bra 0x00023a
	297	00250	BAD911	tblrdh.b [0x0002],[0x0004++]
Opcode Hex	Machine	Symbolic	PSV Mixed	PSV Data

Figure 30 Program Memory

## Appendix D – Task 4

Source code – “task4.c”

```
#include "pic24_all.h"

uint16 x,y;

uint8 count;

void main(void) {

    x = 1;

    y = 3;

    for(count = 3; count>0; count--) {

        if((x-y) == 0) {

            y++;

        }

        if (x < y) {

            x = x + 2;

        }

    }

}
```



Figure (31 - 42)

### **Simulated (31 - 36)**

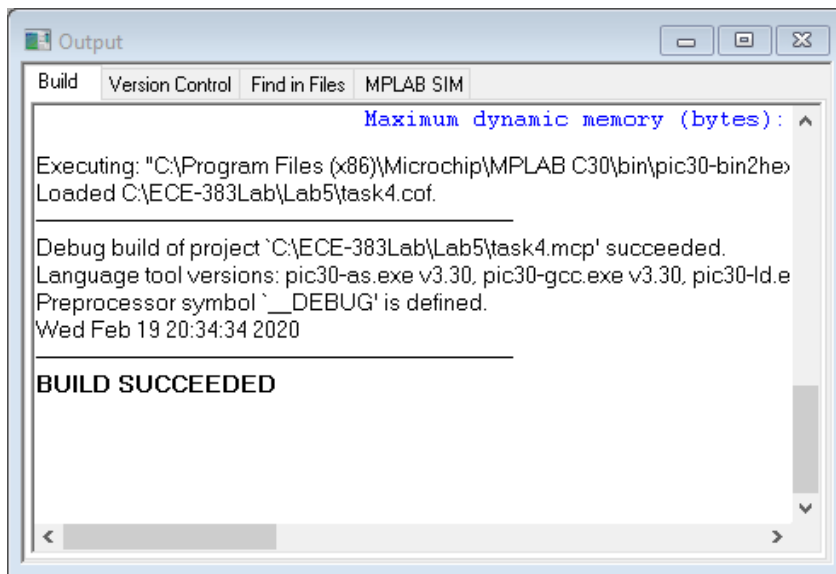


Figure 31 Output Window

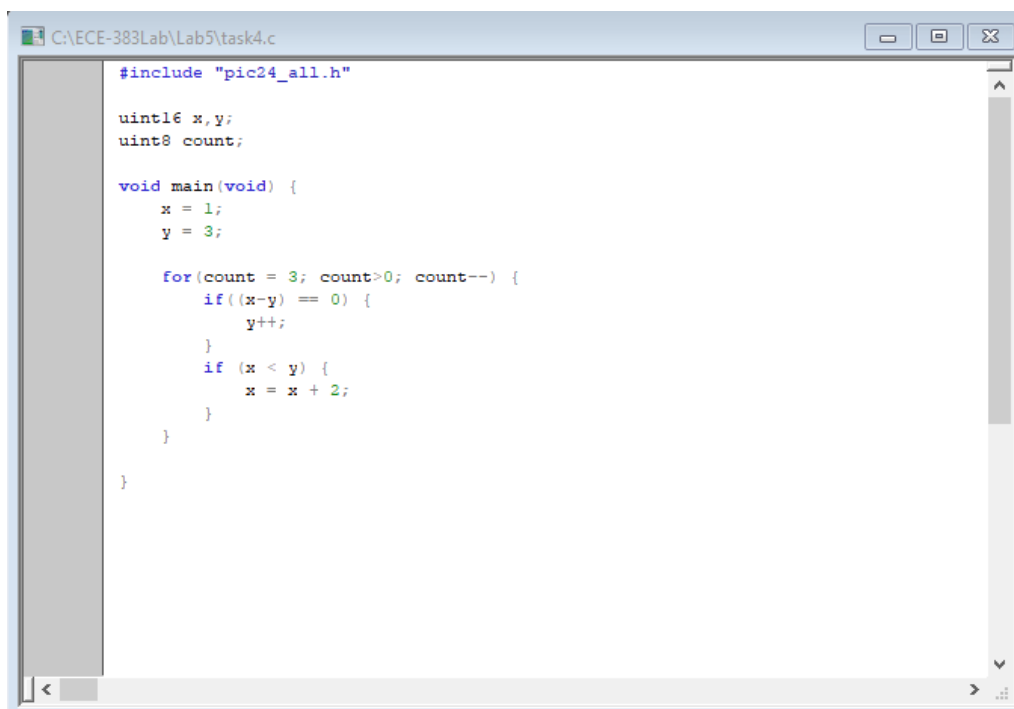


Figure 32 task4.c

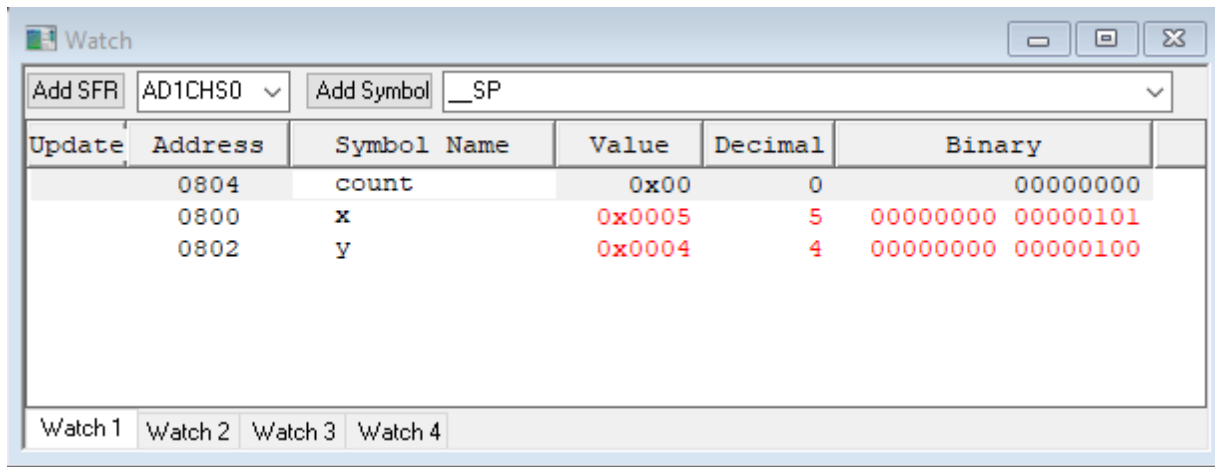


Figure 33 watch window

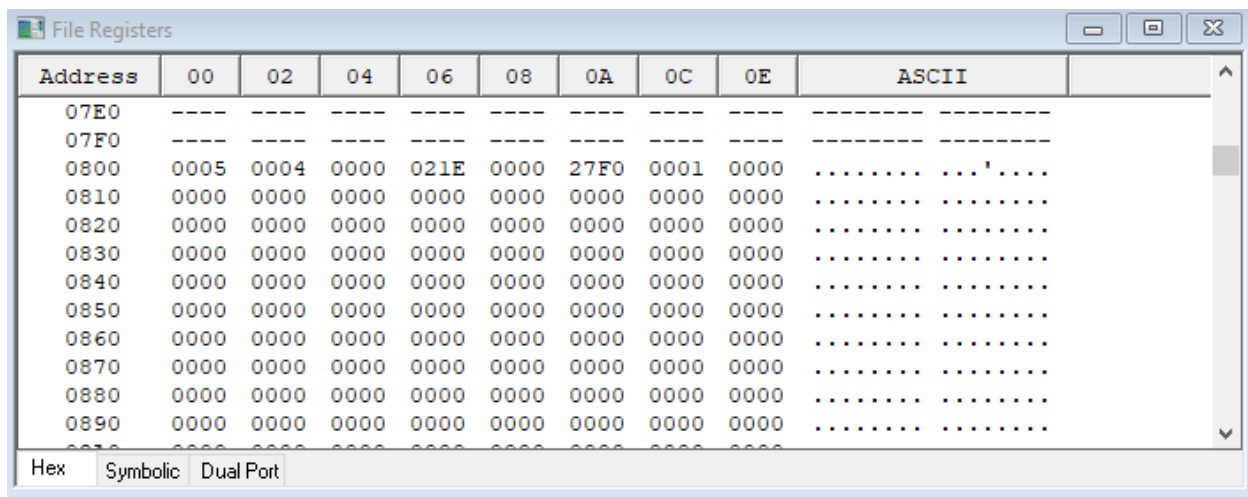


Figure 34 File Registers

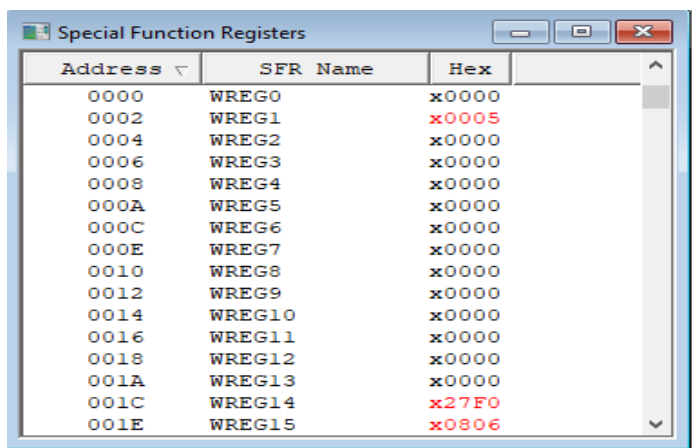


Figure 35 Special Function Register

Program Memory				
	Line	Address	Opcode	Disassembly
	258	00202	227F0E	mov.w #0x27f0,0x001c
	259	00204	88010E	mov.w 0x001c,0x0020
	260	00206	000000	nop
	261	00208	07000C	rcall 0x000222
	262	0020A	202D00	mov.w #0x2d0,0x0000
	263	0020C	200001	mov.w #0x0,0x0002
	264	0020E	070011	rcall 0x000232
	265	00210	200000	mov.w #0x0,0x0000
	266	00212	E00000	cp0.w 0x0000
	267	00214	320002	bra z, 0x00021a
	268	00216	020000	call 0x000000
	269	00218	000000	nop
	270	0021A	020294	call 0x000294
	271	0021C	000000	nop
	272	0021E	DA4000	ReservedBR
→	273	00220	FE0000	reset
	274	00222	A94044	bclr.b 0x0044,#2
	275	00224	200000	mov.w #0x0,0x0000
	276	00226	E00000	cp0.w 0x0000
	277	00228	320003	bra z, 0x000230
	278	0022A	200000	mov.w #0x0,0x0000
	279	0022C	8801A0	mov.w 0x0000,0x0034
	280	0022E	A84044	bset.b 0x0044,#2
	281	00230	060000	return
	282	00232	880191	mov.w 0x0002,0x0032
	283	00234	780080	mov.w 0x0000,0x0002
	284	00236	EB0000	clr.w 0x0000
	285	00238	370015	bra 0x000264
	286	0023A	4080E2	add.w 0x0002,#2,0x0002
	287	0023C	B4A032	addc.w 0x0032
	288	0023E	BA0191	tblrdl.w [0x0002],0x0006
	289	00240	4080E2	add.w 0x0002,#2,0x0002
	290	00242	B4A032	addc.w 0x0032
	291	00244	BA0291	tblrdl.w [0x0002],0x000a
	292	00246	4080E2	add.w 0x0002,#2,0x0002
	293	00248	B4A032	addc.w 0x0032

Opcode Hex
Machine
Symbolic
PSV Mixed
PSV Data

Figure 36 Program Memory

## On PIC24 Microcontroller (37 - 42)

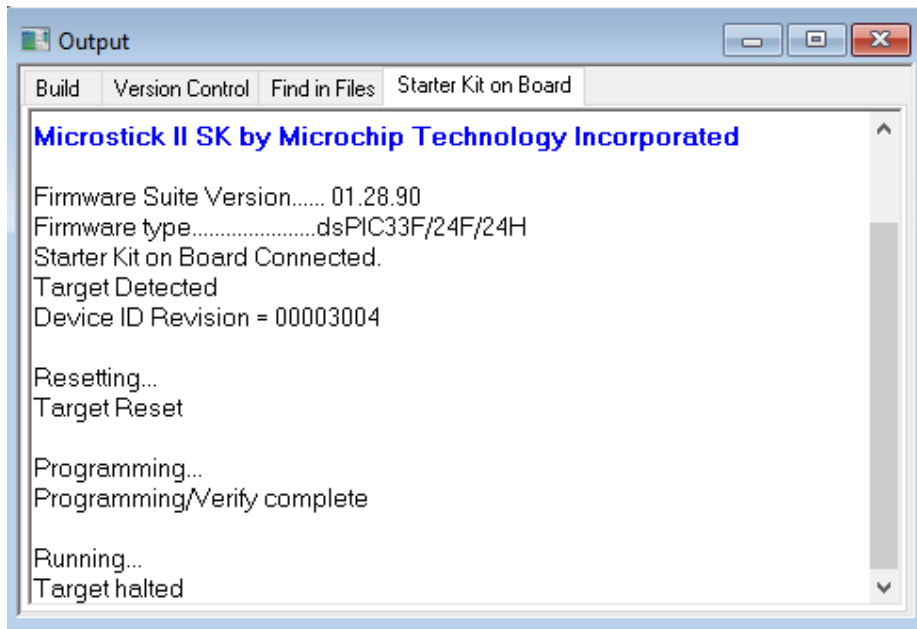


Figure 37 Output Window

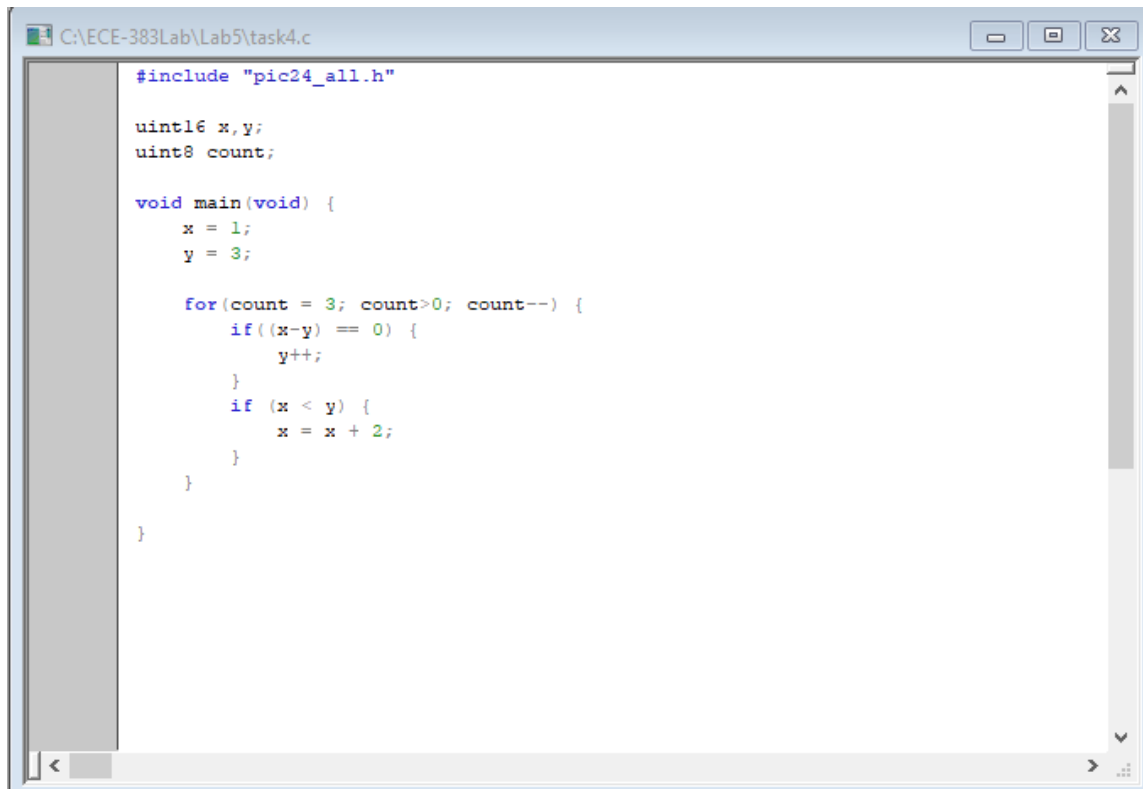


Figure 38 task4.c

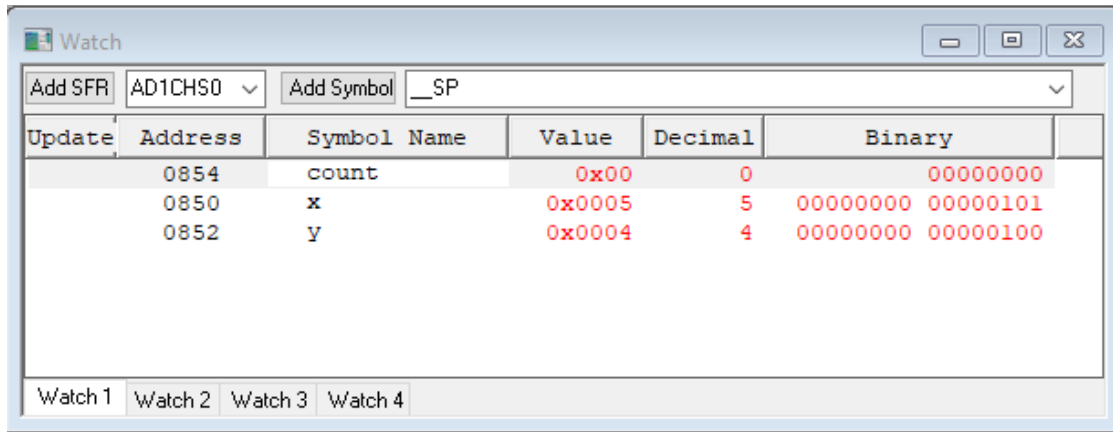


Figure 39 Watch Window

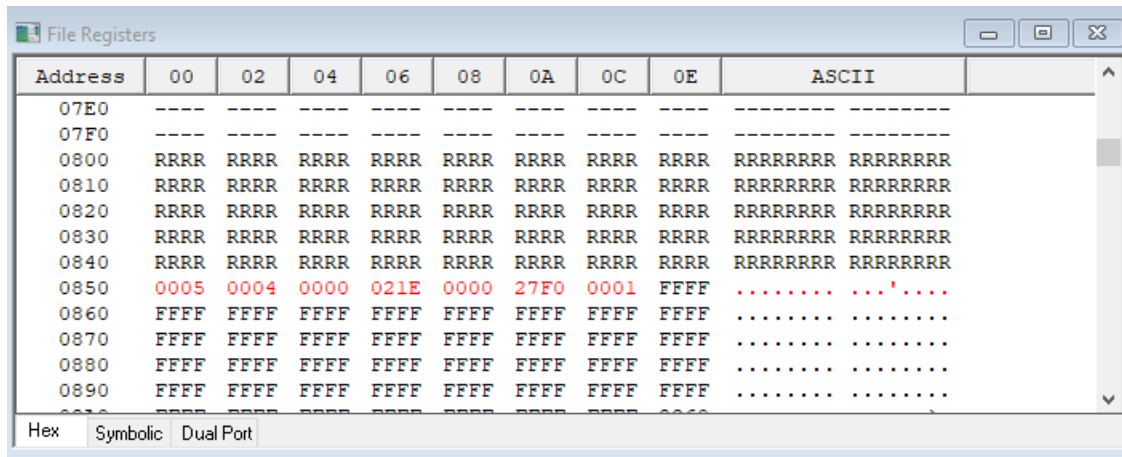


Figure 40 File Registers

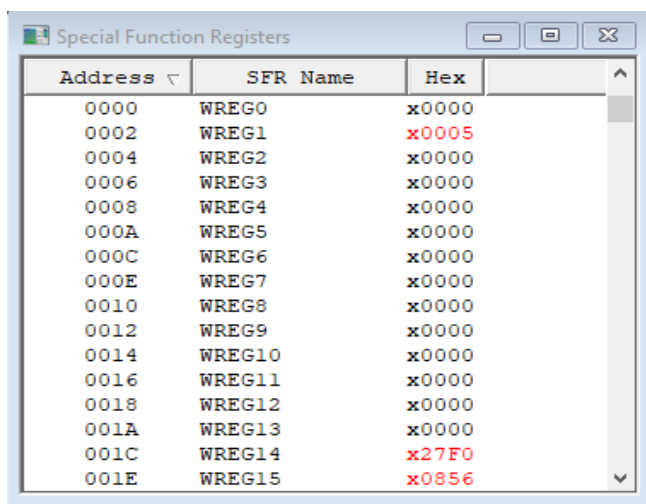


Figure 41 Special Function Registers

Program Memory				
	Line	Address	Opcode	Disassembly
	255	001FC	000290	_DefaultInterrupt
	256	001FE	000290	_DefaultInterrupt
	257	00200	20856F	mov.w #0x856,0x001e
	258	00202	227F0E	mov.w #0x27f0,0x001c
	259	00204	88010E	mov.w 0x001c,0x0020
	260	00206	000000	nop
	261	00208	07000C	rcall 0x000222
	262	0020A	202D00	mov.w #0x2d0,0x0000
	263	0020C	200001	mov.w #0x0,0x0002
	264	0020E	070011	rcall 0x000232
	265	00210	200000	mov.w #0x0,0x0000
	266	00212	E00000	cp0.w 0x0000
	267	00214	320002	bra z, 0x00021a
	268	00216	020000	call 0x000000
	269	00218	000000	nop
	270	0021A	020294	call 0x000294
	271	0021C	000000	nop
	272	0021E	DA4000	ReservedBR
➔	273	00220	FE0000	reset
	274	00222	A94044	bclr.b 0x0044,#2
	275	00224	200000	mov.w #0x0,0x0000
	276	00226	E00000	cp0.w 0x0000
	277	00228	320003	bra z, 0x000230
	278	0022A	200000	mov.w #0x0,0x0000
	279	0022C	8801A0	mov.w 0x0000,0x0034
	280	0022E	A84044	bset.b 0x0044,#2
	281	00230	060000	return
	282	00232	880191	mov.w 0x0002,0x0032
	283	00234	780080	mov.w 0x0000,0x0002
	284	00236	EB0000	clr.w 0x0000
	285	00238	370015	bra 0x000264
	286	0023A	4080E2	add.w 0x0002,#2,0x0002
	287	0023C	B4A032	addc.w 0x0032
	288	0023E	BA0191	tblrdl.w [0x0002],0x0006
	289	00240	4080E2	add.w 0x0002,#2,0x0002
	290	00242	B4A032	addc.w 0x0032
<div> <div>←</div> <div></div> <div>→</div> </div>				
<div> <div>Opcode Hex</div> <div>Machine</div> <div>Symbolic</div> <div>PSV Mixed</div> <div>PSV Data</div> </div>				

Figure 42 Program Memory