ɔeriment NO: 02A

ɲ:

To interface with 7-segment display using 8051/P89V51RD2 microcontroller.

jective:

1.To study the working of seven-segment displays.
2.To interface seven-segment display with 8051.

paratus / Software Required:

8051 / P89V51RD2 microcontroller kit
Seven-segment display
Keil µVision software (for code development).

ıeory :

The 8051's pins are broadly categorized into power and timing, control signals, and I/O ports. **Pins 40 (Vcc) ınd 20 (GND)** are for power, providing the positive supply voltage and ground, respectively. **Pins 18 XTAL2) and 19 (XTAL1)** are used to connect an external crystal oscillator, which generates the clock ;ignal for the microcontroller's internal operations.

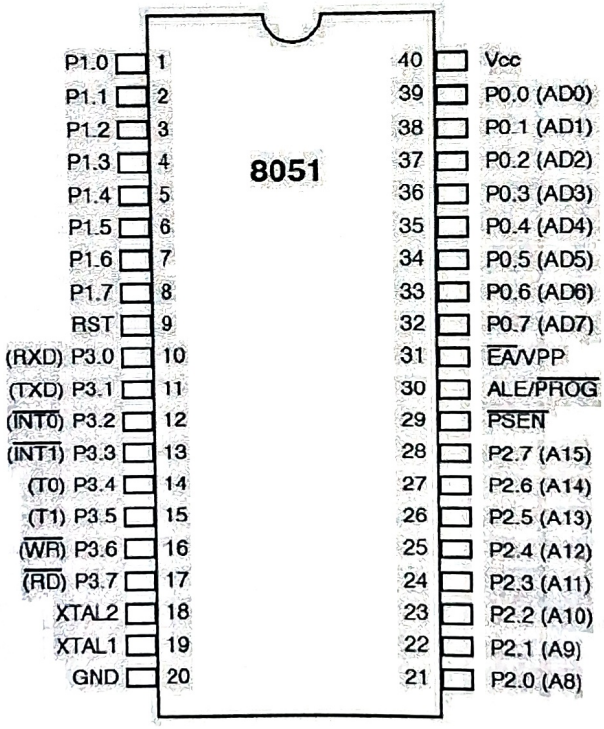


Figure 5.1 Pin Diagram of 8051 Microcontroller

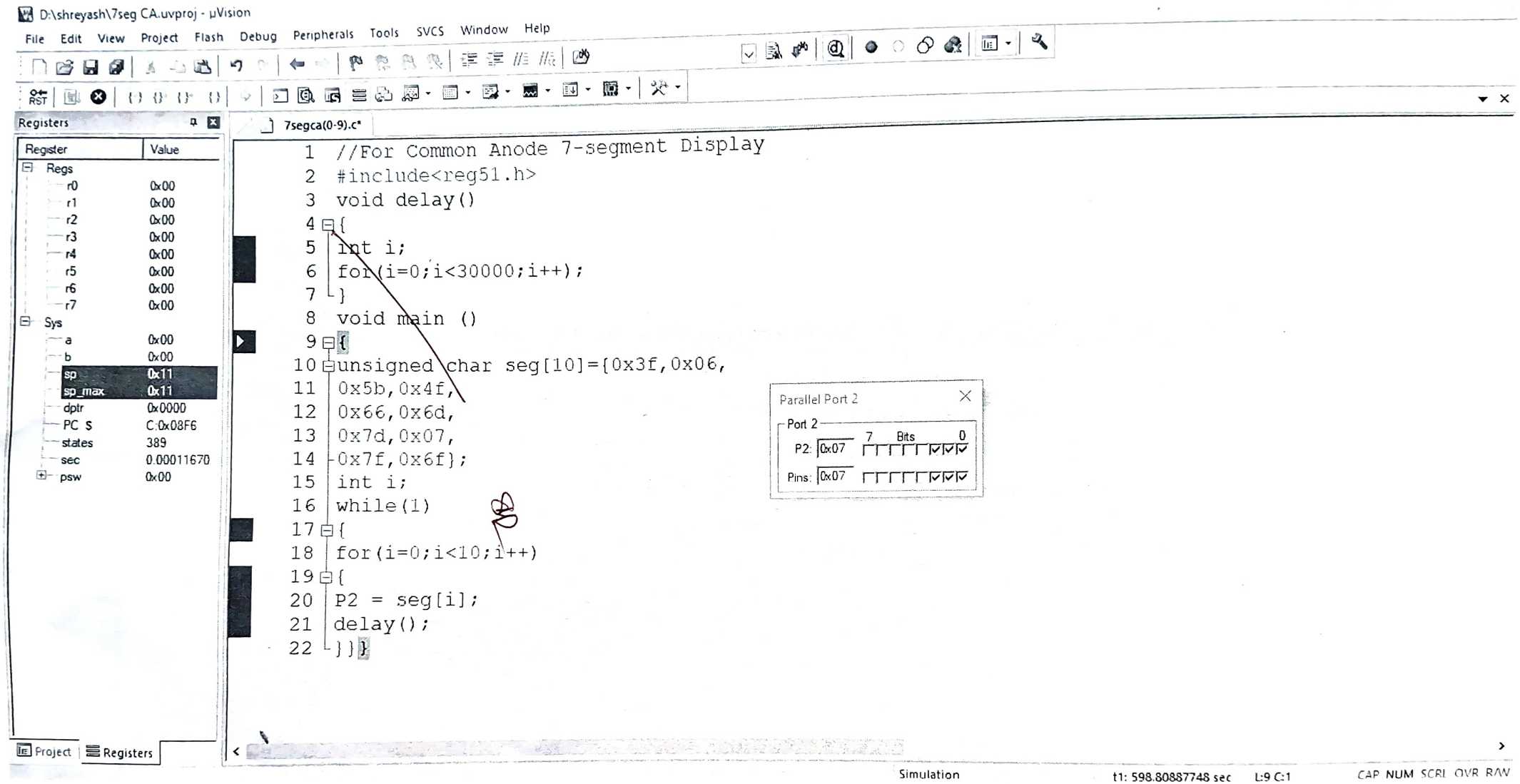
The **control signals** are essential for managing memory and program execution. **Pin 31 (EA/Vpp)** is the External Access Enable pin. When grounded, it forces the 8051 to fetch all instructions from external program memory. When connected to Vcc, it executes instructions from internal program memory until the memory address limit is reached, then automatically switches to external memory. **Pin 30 (ALE/PROG)** stands for Address Latch Enable. It is an output signal that multiplexes the low-order byte of the address from Port 0 during external memory access. **Pin 29 (PSEN)**, or Program Store Enable, is a read strobe signal for external

Conclusion:

We learned interface with 7 segment display using 8051 / P89V51RD2 microcontroller also executed program for 7 segment display.

Timel Submis on (10)	Journal Present ation (10)	Perfor mance (10)	Unders tanding (10)	Oral (10)	Total (50)
10	10	10	8	7	45

Sub Teacher's Sign: _____


D:\shreyash\7seg CA.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register Value
Regs
r0 0x00
r1 0x00
r2 0x00
r3 0x00
r4 0x00
r5 0x00
r6 0x00
r7 0x00
Sys
a 0x00
b 0x00
sp 0x11
sp_max 0x11
dptr 0x0000
PC $ C:0x08F6
states 389
sec 0.00011670
psw 0x00
Project Registers
7segca(0-9).c*
1 //For Common Anode 7-segment Display
2 #include<reg51.h>
3 void delay()
4 {
5 int i;
6 for(i=0;i<30000;i++);
7 }
8 void main ()
9 {
10 unsigned char seg[10]={0x3f,0x06,
11 0x5b,0x4f,
12 0x66,0x6d,
13 0x7d,0x07,
14 0x7f,0x6f};
15 int i;
16 while(1)
17 {
18 for(i=0;i<10;i++)
19 {
20 P2 = seg[i];
21 delay();
22 }}}
Parallel Port 2
Port 2
P2: 0x07 7 Bits 0
Pins: 0x07
Simulation
t1: 598.80887748 sec
L:9 C:1
CAP NUM SCRL OVR RAW

EXPERIMENT NO. 06

Aim: write a program for Generate square wave using timer with interrupt.

Objective: To write a embedded C program to Generate square wave on Port pin using timer with interrupt

Theory:

Timers in microcontrollers are used for introducing delay, counting events, generating waveforms, and also for PWM generation.

Delays in a microcontroller can be induced by either of the two ways –

1. Provide delay using software (through code using loops). But, a delay provided in this way forces the microcontroller to put all its resources for the processing of the loops, and thus blocks the code execution.
2. Providing delay using the timer in the microcontroller. In this approach, the delay is provided by loading a count in a timer. So when a timer interrupt is generated current execution will move to the ISR to serve the task. But when there is no timer interrupt it will perform another task. Therefore, we can say it is a kind of non-blocking function.

PIC18F4550 has 4 in-built timers

- Timer0 : 16-bit
- Timer1 : 16-bit
- Timer2 : 8-bit
- Timer3 : 16-bit

Out of the four timers in PIC18F4550, we will be discussing here Timer1. The working of the rest of the timers is the same. To generate a time delay using Timer1, we have to calculate a count of the desired time.

How to calculate count for Timer?

Let's take an example. Suppose we have to generate a delay of 1 ms having 8 MHz oscillator frequency of PIC18F4550. To do this, we need to find the period of the instruction cycle.

Fosc = 8 MHz

PIC18F4550 timer uses frequency as follows:

$$FTIMER = \frac{Fosc}{4}$$

PIE1: Peripheral Interrupt Enable

7	6	5	4	3	2	1	0
SPPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

Register PIE

TMR1IE: Timer1 Overflow Interrupt Enable Bit

1 = Enable the TMR1 overflow interrupt

0 = Disable the TMR1 overflow interrupt

Note: While generating a delay using ISR, the program takes more time to switch to ISR. Therefore, to avoid this we have to increase the count to compensate for the delayed time.

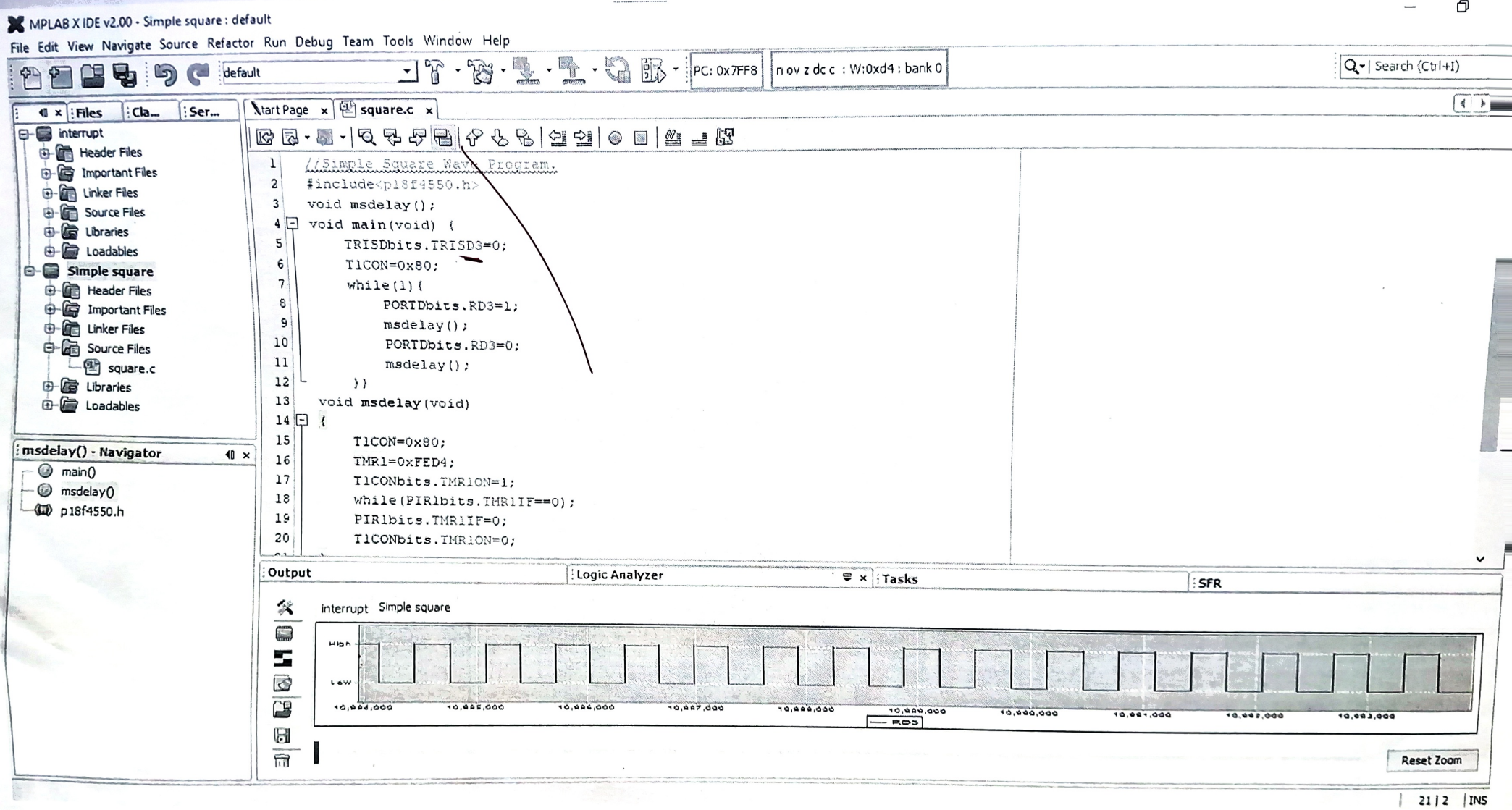
Steps for Programming PIC18F4550 Timer using Interrupt

1. Enable GIE, PEIE, TMR1IE.
2. Configure the T1CON register.
3. Clear TMR1IF Timer1 interrupt flag.
4. Load the count in TMR1 16-bit or TMR1H (higher byte) and TMR1L (lower byte).
5. Set TMR1ON to start the Timer1 operation.
6. When TMR1IF = 1, code will jump to ISR to execute it, and after execution control returns to the main program.

Conclusion:

This successfully demonstrated the generation of a square wave on a port pin of the PIC18F4550 using Timer1 with interrupt.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	07	07	44
Sub Teacher Sign:					


MPLAB X IDE v2.00 - Simple square : default
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
default
PC: 0x7FF8
n ov z dc c : W:0xd4 : bank 0
Search (Ctrl+I)
Files
Cla...
Ser...
interrupt
Header Files
Important Files
Linker Files
Source Files
Libraries
Loadables
Simple square
Header Files
Important Files
Linker Files
Source Files
square.c
Libraries
Loadables
msdelay() - Navigator
main()
msdelay()
p18f4550.h
Start Page
square.c
//Simple Square Wave Program.
#include<p18f4550.h>
void msdelay();
void main(void) {
TRISDbits.TRISD3=0;
T1CON=0x80;
while(1){
PORTDbits.RD3=1;
msdelay();
PORTDbits.RD3=0;
msdelay();
}}
void msdelay(void)
{
T1CON=0x80;
TMR1=0xFED4;
T1CONbits.TMR1ON=1;
while(PIR1bits.TMR1IF==0);
PIR1bits.TMR1IF=0;
T1CONbits.TMR1ON=0;
Output
Logic Analyzer
Tasks
SFR
interrupt Simple square
High
Low
RD3
Reset Zoom
21 | 2
INS

EXPERIMENT NO. 7

AIM: Generation of PWM signal for DC motor control

Objective: 1.To write embedded "C" program to generation of PWM signal .
2.To observe PWM signal on CRO

Apparatus: MPLAB Simulator, C18, PC, PIC18f Development KIT, USB cable, CRO

Theory: PIC18F2455/2550/4455/4550 devices all have two CCP (Capture/Compare/PWM) modules. Each module contains a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

In 28-pin devices, the two standard CCP modules (CCP1 and CCP2) operate as described in this chapter. In 40/44-pin devices, CCP1 is implemented as an Enhanced CCP module, with standard Capture and Compare modes and Enhanced PWM modes. The ECCP is "Enhanced Capture/Compare/PWM (ECCP) Module".

The Capture and Compare operations described in this chapter apply to all standard and Enhanced CCP modules.

REGISTER 15-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 Unimplemented: Read as '0'

bit 5-4 DCxB1:DCxB0: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module

Capture mode: Unused.

Compare mode:Unused.

PWM mode:These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 CCPxM3:CCPxM0: CCPx Module Mode Select bits

0000= Capture/Compare/PWM disabled (resets CCPx module)

0001= Reserved

with RB3.Changing the pin assignment of CCP2 does not automatically change any requirements for configuring the port pin. Users must always verify that the appropriate TRIS register is configured correctly for CCP2 operation, regardless of where it is located.

INTERACTIONS BETWEEN CCP1 AND CCP2 FOR TIMER RESOURCES

CCP/ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
Timer 2	PWM

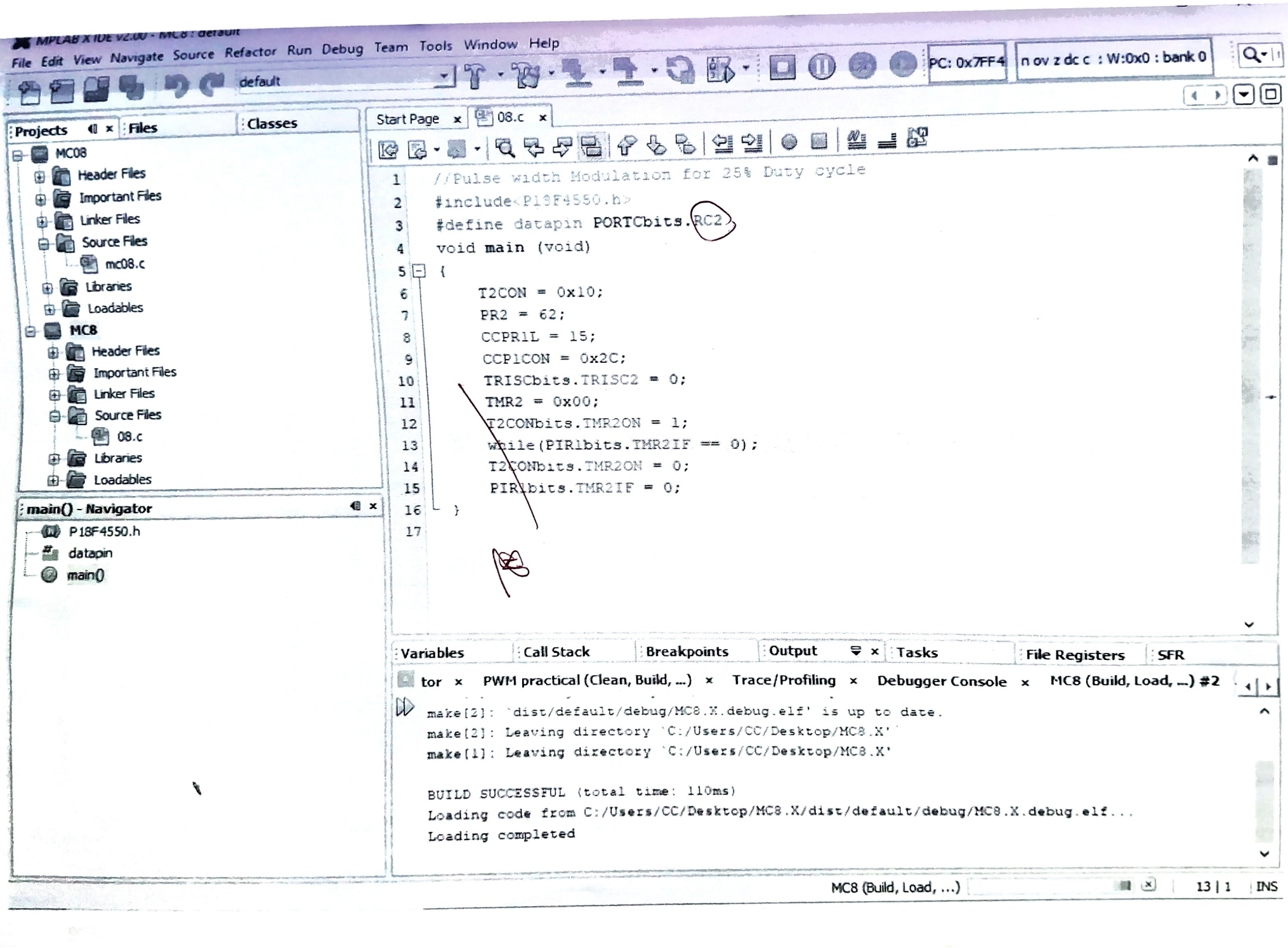
Algorithm:-

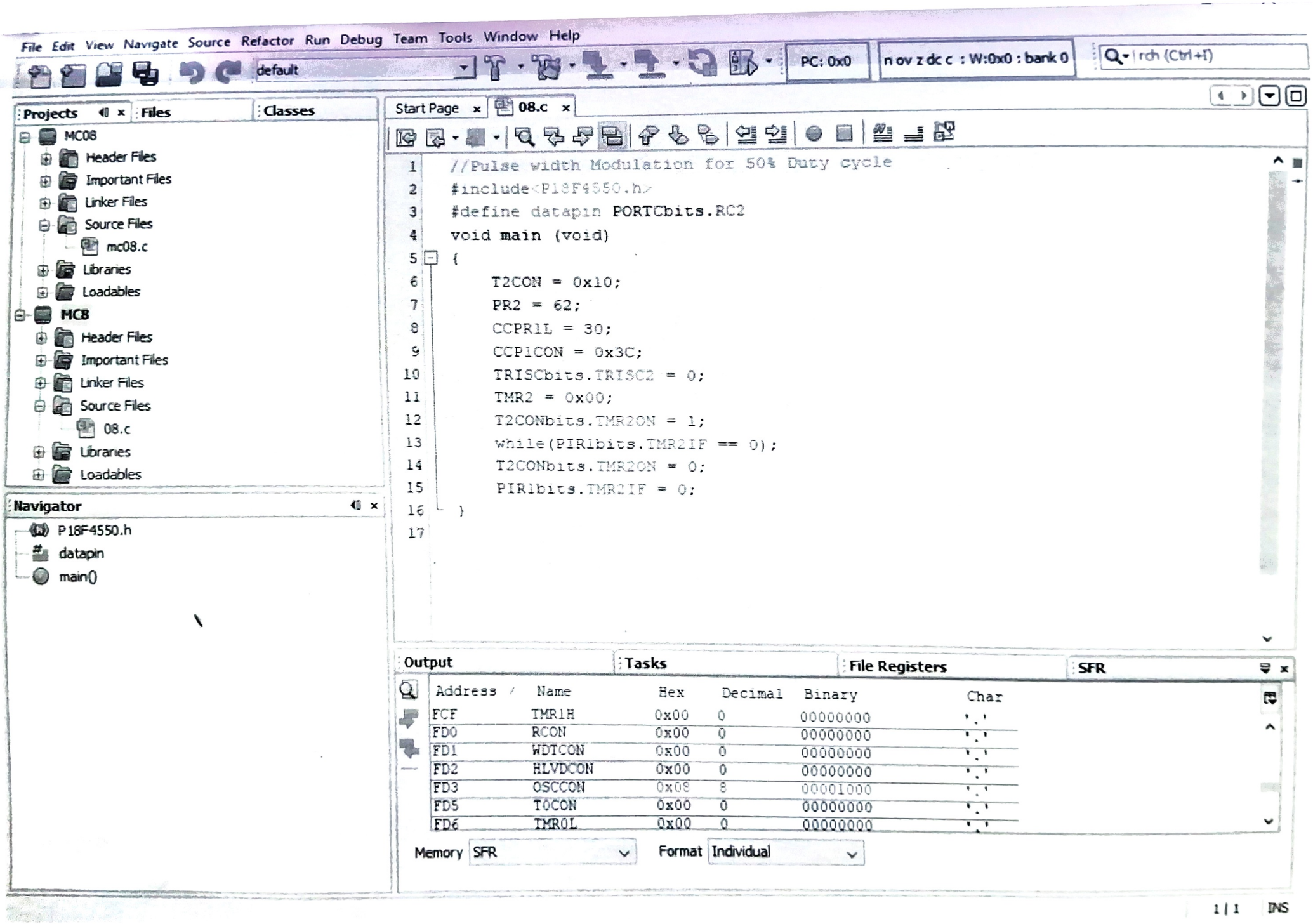
1) Start
2) Initialize code ,duty =0
3) Select PWM mode, configure CCPON register.
4) Set PWM period in PR2 Register (PR2=FF)
5) Declare port c as a output port.
6) Start timer 2 & load T2ON register with 0*7E
7) Declare While loop.
8) Take the code value from port B.
9) If the key 1 is pressed increases the duty cycle.
10) Load the new value of CCP1L & CCP1CON Register.
11) If key 2 is pressed decreases the duty cycle.
12) Load the new value & CCP1L & CCP1CON register.
13) Call Delay.
14) Initialize for Loop With (i=0;i<1000;i++)
15) Initialize for loop with (j=0;j<1000;j++)
16) STOP.

Conclusion:

In this experiment for generation of PWM Signal for DC Motor Control we use CCP Module in PWM Mode. we observed when button 1 is pressed then increase the duty cycle & vice versa condition when button 2 is pressed like that we can control speed of DC Motor.

Timely submission(10)	Journal Presentation(10)	Performance(10)	Understanding(10)	Oral(10)	Total (50)
10	10	10	08	07	45
Sub Teacher Sign:					


MPLAB X IDE v2.00 - MC8 : default
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
default
PC: 0x7FF4
n ov z dc c : W:0x0 : bank 0
Projects
Files
Classes
MC08
Header Files
Important Files
Linker Files
Source Files
mc08.c
Libraries
Loadables
MC8
Header Files
Important Files
Linker Files
Source Files
08.c
Libraries
Loadables
main() - Navigator
P18F4550.h
datapin
main()
Start Page
08.c
1 //Pulse width Modulation for 25% Duty cycle
2 #include<P18F4550.h>
3 #define datapin PORTCbits.RC2
4 void main (void)
5 {
6 T2CON = 0x10;
7 PR2 = 62;
8 CCPR1L = 15;
9 CCP1CON = 0x2C;
10 TRISCbits.TRISC2 = 0;
11 TMR2 = 0x00;
12 T2CONbits.TMR2ON = 1;
13 while(PIR1bits.TMR2IF == 0);
14 T2CONbits.TMR2ON = 0;
15 PIR1bits.TMR2IF = 0;
16 }
17
Variables
Call Stack
Breakpoints
Output
Tasks
File Registers
SFR
tor
PWM practical (Clean, Build, ...)
Trace/Profiling
Debugger Console
MC8 (Build, Load, ...) #2
make[2]: `dist/default/debug/MC8.X.debug.elf' is up to date.
make[2]: Leaving directory `C:/Users/CC/Desktop/MC8.X'
make[1]: Leaving directory `C:/Users/CC/Desktop/MC8.X'
BUILD SUCCESSFUL (total time: 110ms)
Loading code from C:/Users/CC/Desktop/MC8.X/dist/default/debug/MC8.X.debug.elf...
Loading completed
MC8 (Build, Load, ...)
13 | 1
INS


File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
default
PC: 0x0
n ov z dc c : W:0x0 : bank 0
rch (Ctrl+I)
Projects
Files
Classes
MC08
Header Files
Important Files
Linker Files
Source Files
mc08.c
Libraries
Loadables
MC8
Header Files
Important Files
Linker Files
Source Files
08.c
Libraries
Loadables
Navigator
P18F4550.h
datapin
main()
Start Page
08.c
1 //Pulse width Modulation for 50% Duty cycle
2 #include<P18F4550.h>
3 #define datapin PORTCbits.RC2
4 void main (void)
5 {
6 T2CON = 0x10;
7 PR2 = 62;
8 CCPR1L = 30;
9 CCP1CON = 0x3C;
10 TRISCbits.TRISC2 = 0;
11 TMR2 = 0x00;
12 T2CONbits.TMR2ON = 1;
13 while(PIR1bits.TMR2IF == 0);
14 T2CONbits.TMR2ON = 0;
15 PIR1bits.TMR2IF = 0;
16 }
17
Output
Tasks
File Registers
SFR
Address / Name Hex Decimal Binary Char
FCF TMR1H 0x00 0 00000000 '.'
FD0 RCON 0x00 0 00000000 '.'
FD1 WDTCON 0x00 0 00000000 '.'
FD2 HLVDCON 0x00 0 00000000 '.'
FD3 OSCCON 0x08 8 00001000 '.'
FD5 T0CON 0x00 0 00000000 '.'
FD6 TMR0L 0x00 0 00000000 '.'
Memory SFR
Format Individual
1 | 1
INS