PIC Programs

4	WAP for interfacing button, LED, relay and buzzer as follows - On pressing button1 relay and buzzer is turned ON and LED‘s start chasing from left to right; When button 2 is pressed relay and buzzer is turned OFF and LED‘‘s start chasing from left to right. (Group B: 6)
5	Interfacing of LCD to PIC 18FXXXX. (Group B: 7)
6	Generate square wave using timer with interrupt. (Group B: 9)
7	Interfacing serial port with PC both side communication. (Group C: 11)
8	Generation of PWM signal for DC Motor control. (Group C: 13)

```
/*****************************************************************************
* Expt - 4 WAP for interfacing button, LED, relay and buzzer as follows -
Description: This file contains Interface With Relay, Buzzer, Switch, Led's.
A. when button 1 is pressed relay and buzzer is turned ON and LED’s start chasing from left to
right  B. when button 2 is pressed relay and buzzer is turned OFF and Led start chasing from right
to left  *****************************************************************************/
#include <p18f4520.h>
#include <delays.h>

#pragma config OSC = HS // High-speed oscillator
#pragma config WDT = OFF //Watchdog Timer disabled
#pragma config LVP = OFF // Low-voltage Programming disabled
#pragma config PBADEN = OFF


#define BUZZER PORTAbits.RA3 //Buzzer connected to PORTA 3rd PIN


 #define SWITCH0 PORTBbits.RB0 //Switch0 connected to PORTB 0th PIN


#define SWITCH1 PORTBbits.RB1 //Switch1 connected to PORTB 1st PIN

void main(void)
{
        TRISA = 0x00; // RA3,OutPut Direction
        TRISB = 0xff; // RB0,B1 Input Direction
        TRISD = 0x00; // [RD0-3=LED's][RD4,5=Relay1,2]OutPut Direction
        PORTD = 0xff; // [RD0-3=LED's][RD4,5=Relay1,2] Initialise as 0xff
        BUZZER = 0x00;

while(1)
{
 if(SWITCH0==0) // Condition for 1st switch   {
 while(1)
        {
```

```
            BUZZER =1; // Buzzer On
            PORTD = 0x37; //(Relay1&2 = 1)&(LED's sequence L-to-R =0111=7)  Delay10KTCYx(100); //
            400mSDelay
            PORTD = 0x3B; // (LED's sequence Left to Right=1011=B)  Delay10KTCYx(100);
            PORTD = 0x3D;
            Delay10KTCYx(100);
            PORTD = 0x3E;
            Delay10KTCYx(100);
            if(SWITCH1==0) // check if 2nd switch is pressed
            break;
 }
            }

 else if(SWITCH1==0) // Condition for 2nd switch   {
            while(1)
            {
            BUZZER =0; // Buzzer Off
            PORTD = 0xcE; //(Relay1&2 = 0)&(LED's seq R-to Left=1110=E)   Delay10KTCYx(100);
            PORTD = 0xcD; // LED's sequence Right to Left=1101=D  Delay10KTCYx(100);
            PORTD = 0xcB;
            Delay10KTCYx(100);
            PORTD = 0xc7;
            Delay10KTCYx(100);
                        if(SWITCH0==0) // check if 1st switch is pressed
                        break;
                        }
            }
 }
 }


/*****************************************************************************************
 * Expt – 5 LCD connections:
  RC0 RC1 RE2 RE1 RE0 RC2 RS EN D7 D6 D5 D4 4 bit interface is used.
 *****************************************************************************************/
 /*****************************************************************************
Expt – 5 LCD connections:
RC0 RC1 RE2 RE1 RE0 RC2
RS EN D7 D6 D5 D4
4 bit interface is used.
*****************************************************************************/
#include <p18f4520.h>
#include <stdio.h>
#include <delays.h>
#include "LCD.h"
#pragma config OSC = HS // High-speed oscillator
#pragma config WDT = OFF // Watchdog Timer disabled
#pragma config LVP = OFF // Low-voltage Programming disabled
/*
Delay10KTCYx(1)4mS
Delay10KTCYx(2)8mS
*/
#define rs LATCbits.LATC0
```

```
#define en LATCbits.LATC1

unsigned char arr1[] = "Hello World";
unsigned char arr2[] = "Microcontroller";

void MyDelay(unsigned int Count)
{
while(Count)
{
Count-- ;
}
}
void lcdcmd1(unsigned char command)
{
LATCbits.LATC2 = (command) & 0x1; //RC2=1
LATEbits.LATE0 = (command >> 1 ) & 0x1; //RE0=1
LATEbits.LATE1 = (command >> 2) & 0x1; //RE1=0
LATEbits.LATE2 = (command >> 3 ) & 0x1; //RE2=0
en=0;
rs=0;
Delay10KTCYx(1);
en=1;
Delay10KTCYx(1);
en=0;
Delay10KTCYx(1);
}
void lcdcmd(unsigned char value)
{
/*
example value command =0x38 =>
Require to send command lik this Way:
highernibble = 0 0 1 1 | 0 0 0 0

Microcontrollers (2019 Course) T.E. (E&TC)

D7 D6 D5 D4 | D3 D2 D1 D0

RE2 RE1 RE0 RC2

lowernibble = 1 0 0 0 | 0 0 0 0
D7 D6 D5 D4 | D3 D2 D1 D0
here D0-D3 is not used because of 4-Bit DATA LINES
So we have to send data(0x3 & 0x8) on D4-D7 Data lines two times
*/
char lowernibble=0,highernibble=0;
//Exmaple Value =0x38
lowernibble = value & 0x0f; //lowernibble = 0x08
highernibble = value & 0xf0; //highernibble = 0x30
highernibble = (highernibble >>4) & 0x0f ; //highernibble = 0x03
```

```
lcdcmd1(highernibble);
lcdcmd1(lowernibble);
Delay10KTCYx(2);
}
void lcddata1(unsigned char data)
{
//data = 0x38=>0x03,0x08 nibble
LATCbits.LATC2 = (data) & 0x1; //RC2=1
LATEbits.LATE0 = (data >> 1 ) & 0x1; //RE0=1
LATEbits.LATE1 = (data >> 2) & 0x1; //RE1=0
LATEbits.LATE2 = (data >> 3 ) & 0x1; //RE2=0
rs=1;
Delay10KTCYx(1);
en=0;
Delay10KTCYx(1);
en=1;
Delay10KTCYx(1);
en=0;
Delay10KTCYx(1);
}
void lcddata(unsigned char value)
{
/*
exmple value command =0x38 =>
Require to send command lik this Way:
highernibble = 0 0 1 1 | 0 0 0 0
D7 D6 D5 D4 | D3 D2 D1 D0

RE2 RE1 RE0 RC2

lowernibble = 1 0 0 0 | 0 0 0 0
D7 D6 D5 D4 | D3 D2 D1 D0
here D0-D3 is not used because of 4-Bit DATA LINES
So we have to send data(0x3 & 0x8) on D4-D7 Data lines two times
*/
char lowernibble=0,highernibble=0;
//Exmaple Value =0x38
lowernibble = value & 0x0f; //lowernibble = 0x08
highernibble = value & 0xf0; //highernibble  = 0x30
highernibble = (highernibble >>4) & 0x0f ; //highernibble = 0x03
lcddata1(highernibble);
lcddata1(lowernibble);
Delay10KTCYx(2);
}
void lcdinit()
{
//Configure OutPut Pin
TRISEbits.RE0 = 0; //OUTPUT DIR OF RE0
TRISEbits.RE1 = 0; //OUTPUT DIR OF RE1
TRISEbits.RE2 = 0; //OUTPUT DIR OF RE2
```

```
TRISCbits.RC0 = 0; //OUTPUT DIR OF RC0
TRISCbits.RC1 = 0; //OUTPUT DIR OF RC1
TRISCbits.RC2 = 0; //OUTPUT DIR OF RC2
//Make Output Value =0 to all Pins
PORTEbits.RE0 = 0;
PORTEbits.RE1 = 0;
PORTEbits.RE2 = 0;
PORTCbits.RC0 = 0;
PORTCbits.RC1 = 0;
PORTCbits.RC1 = 0;
Delay10KTCYx(1);
lcdcmd(0x03);
Delay10KTCYx(1);
lcdcmd(0x03);
Delay10KTCYx(2);
lcdcmd(0x03);
Delay10KTCYx(2);
lcdcmd(0x02);
Delay10KTCYx(2);
lcdcmd(0x28);
lcdcmd(0x08);
lcdcmd(0x0c);
lcdcmd(0x06);
}
void DisplayRow (int row, char *str)
{
/*
pass pointer to 16 character string
displayes the message on line1 or line2 of LCD, depending on
whether row is 1 or 2.
*/
int k ;
//Either Line 1 select or select Line 2
if (row == 1)
{
lcdcmd(0x80) ;
}
if(row == 2)
{
lcdcmd(0xc0) ;
}
//After Selection of LCD Line send Data from 0 to 15 Char
for(k = 0 ; k < 16 ; k ++)
{
if (str[k])
lcddata(str[k]) ;
else
break ;

}
```

```
while(k < 16)
{
lcddata(' ') ;
k ++ ;
}
}
void main (void)
{

TRISB = 0XFF ; // INPUT FROM KEYs
TRISD = 0x00 ; // OUTPUT(LEDs)
PORTD = 0x00 ;

TRISD = 0x00;
PORTD = 0x00;

MyDelay(100) ;
lcdinit();

PORTD = 0x00;

while(1)
{

DisplayRow (1,arr1);
DisplayRow (2,arr2);
}
}
```

```
/*****************************************************************************************
* Description: Header files containing 16x2 character lcd display functions.
*****************************************************************************************
*/ #ifndef LCD_H
#define LCD_H

extern void lcddata(unsigned char value);
extern void lcddata(unsigned char value);
extern void lcdinit();
extern void DisplayRow (int row, char *str);

#endif //LCD_H
```

```
/*********************************************************************
* Expt - 6 Generate square wave using timer with interrupt.
Description: This program is related to led blinking on port D.
*********************************************************************
*/ /*
Cal = for 20MHz oscillator and 256 prescalar, the effective instruction execution speed is
(4*256)/20×10^6  which will give you 51.2uS (micro second) per instruction.
```

So for 1 second 19625(it is 1/51.2uS).
Since we know interrupt happens only when the timer buffer overflows,
for a 16 bit timer the preset value to achieve 1 second for overflow will be 65535 – 19625(0x4c4b) = 0xb3b4.
*/

```
#include <p18f4520.h>
#include <delays.h>


#pragma config OSC = HS // High-speed oscillator
#pragma config WDT = OFF // Watchdog Timer disabled
#pragma config LVP = OFF // Low-voltage Programming disabled
#pragma config PBADEN = OFF


#define BUZZER PORTAbits.RA3 //Buzzer connected to PORTA 3rd PIN

unsigned char TimerOnflag=0, Num=0;

void Timer0Init()
{
	T0CON = 0x87;  ; // Prescaler of 256, Timer Prescaler assigned , Increment on low to high transition,
			//16 BIT MODE, Turns timer on

	TMR0H =0xB3;
	TMR0L = 0xB4;
	// Interrupt Setup
	RCONbits.IPEN = 1; // Interrupt priority enabled

	INTCON = 0xE0;  // Enables Global Interrupts, Peripheral Interrupts, Enables overflow Interrupt, Sets
			//overflow bit to zero

	INTCON2bits.TMR0IP = 1; // Timer 0 Interrupt is high priority
}

void TMRISR (void) ;
#pragma code InterruptVectorHigh=0x08

void InterruptVectorHigh (void)
{
	_asm
	goto TMRISR
	_endasm
}
#pragma code
#pragma interrupt TMRISR
void TMRISR (void)	//Timer ISR
{
	if(INTCONbits.TMR0IF == 1)
	{
		INTCONbits.TMR0IF=0; //Clear Interrupt Flag to prevent reinitialisation of ISR
		TMR0H =0xB3;
		TMR0L = 0xB4;
		if(TimerOnflag)
```

```
                {
                        PORTD = 0x00;
                        TimerOnflag=0;
                        BUZZER =0;
                        Delay10KTCYx(700);

                }
                else
                {
                        PORTD = 0xff;
                        TimerOnflag=1;
                        BUZZER =0;
                        Delay10KTCYx(700);
                }
        }
}

void main (void)
{
        Timer0Init();
        TimerOnflag=0;
        TRISD = 0x00; //Port configured as Output
        PORTD = 0x00; //PORTD = 0x00

        while(1)
        {
           BUZZER =1;
        }
}
```

```
/*****************************************************************************************
* Expt 7: Interfacing serial port with PC both side communication.
Description: This program sends & receives data on serial port
*****************************************************************************************
*/ #include <p18f4520.h>
#include <stdio.h>
#include <delays.h>

#pragma config OSC = HS // High-speed oscillator
#pragma config WDT = OFF // Watchdog Timer disabled
#pragma config LVP = OFF // Low-voltage Programming disabled

void InitUSART()
{
        TRISCbits.TRISC6 = 1; //make Tx Pin of UART
        TRISCbits.TRISC7 = 1; //make Rx Pin of UART
        SPBRG = 31 ; //9600 baud @ 20MHz(HS MODE)
        TXSTA = 0x20; //Transmitter Enable
        RCSTAbits.SPEN =1; //Serial Port Enble
        RCSTAbits.CREN =1; //Continues Receive Enable bit
}
```

```
unsigned char getchar(void)
{
        unsigned char ch;
        while (!(PIR1bits.RCIF==1)); //wait until there's a character to be read ch=RCREG;
                                     //then read from the Receiver Buffer Register return ch;
}

unsigned char putchar(unsigned char ch)
{
        while (!(PIR1bits.TXIF==1)); //wait until Transmitter Buffer Register Empty TXREG=ch;
                                     //then write to the Transmitter Buffer Register return ch;
}

void main (void)
{
        InitUSART() ;
        printf("Hello PIC18f4520!\r\n") ;
        while(1)
        {
         putchar(getchar());
        }
}
```

/**

* **Expt 8: Generation of PWM signal for DC Motor control.**

Description: This program is related to DC on PWM.

****

```
*/ #include <p18f4520.h>
#include <delays.h>

#pragma config OSC = HS // High-speed oscillator
#pragma config WDT = OFF // Watchdog Timer disabled
#pragma config LVP = OFF // Low-voltage Programming disabled
#pragma config PBADEN = OFF

void main()
{
        //RC1= interfacing with STK
        //RC3= interfacing with Ultralite
        unsigned char dc ;
        TRISC = 0 ; // set PORTC as output RC1,RC2,RC3 Configure as a PWM PORTC = 0 ; // clear
        PORTC
        /*
         * configure CCP module as 10000 Hz PWM output
         */
        PR2 = 0b01111100 ;
        T2CON = 0b00000101 ;
        CCP1CON = 0b00001100 ;
        CCP2CON = 0b00111100 ;

        for(;;) // forever
        {
                /*
```

```
                * PWM resolution is 10 bits
                * don't use last 2 less significant bits CCPxCON,
                * so only CCPRxL have to be touched to change duty cycle
                */
               for(dc = 0 ; dc < 128 ; dc++)
               {
                        CCPR1L = dc ;
                        CCPR2L = 128 - dc ;
                        Delay10KTCYx(50);//200mS
               }
               for(dc = 127 ; dc > 0 ; dc--)
               {
                        CCPR1L = dc ;
                        CCPR2L = 128 - dc ;
                        Delay10KTCYx(50);//200mS

               }
        }
}
```