S.E. (I.T) PIC18F LAB Manual

# Ver 1.0.

# [Expt 6:Write an Embedded C program for interfacing PIC18FXXX to LED and blinking](#_bookmark0) [it using specified delay. 3](#_bookmark0)

# [Expt 7:Write an Embedded C program for ISR based buzzer on/off using Timer. 5](#_bookmark1)

# [Expt 8:Write an Embedded C program for External Interrupt input switch press,](#_bookmark2) [output at Relay. 7](#_bookmark2)

# [Expt 9:Write an Embedded C program for LCD interfacing with PIC18Fxxx 9](#_bookmark3)

# [Expt 10: Write an Embedded C program for generating PWM signal for DC/Servo](#_bookmark4) [motor on PIC18Fxxx 12](#_bookmark4)

# [Expt 11: Write an Embedded C program for PC communication using serial interface](#_bookmark5) [(UART) 14](#_bookmark5)

# [Expt 12: Write an Embedded C program for interfacing PIC18FXXX to Temperature](#_bookmark6) [sensor interfacing using ADC & display on LCD 16](#_bookmark6)

**Expt 6:Write an Embedded C program for interfacing PIC18FXXX to LED and blinking it using specified delay.**

## **Aim:** To write a C program to interface PIC18F4550 to LED and blink it with a specified delay.

**Experimental Setup:** MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure:**

**Step1:** Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2:** Write the program in C language for interfacing LEDs to PIC18F4550. **(in program properties make sure to add the 0x800 offset)**

## **Step3:** Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4:** Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5:** Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6:** Press reset button and execute the program.

**Result:** Check if the LEDs are blinking. You can change the delay and vary the blinking rate.

**Program:**

#include <p18f4550.h>

void delay(unsigned int time)

{

unsigned int i,j; for(i=0;i<time;i++)

for(j=0;j<5000;j++);

}

void main(void)

{

TRISB = 0x00; LATB = 0xFF;

while(1) //Loop forever;

{

LATB = ~LATB;

delay(200);

}

}

**Expt 7:Write an Embedded C program for ISR based buzzer on/off using Timer.**

## **Aim**: To write a C program to interface PIC18F4550 to Buzzer and switch it ON/OFF using Timer ISR..

**Experimental Setup**: MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure**:

**Step1**: Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2**: Write the program in C language for interfacing Buzzer to PIC18F4550, using Timer ISR. **(in program properties make sure to add the 0x800 offset)**

## **Step3**: Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4**: Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5**: Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6**: Press reset button and execute the program.

**Result**: Check if the buzzer is sounding ON/OFF and the ISR is getting executed with the specified timer delay. You can change the delay and vary the sounding rate.

**Program:**

#include <pic18f4550.h> /\* Contains PIC18F4550 specifications \*/ #define Buzzer LATAbits.LATA5 /\* Define buzzer pin \*/ unsigned int count = 0;

void interrupt Timer1\_ISR()

{

if(TMR1IF==1)

{

//1 ms delay time in timer TMR1L = 0x20;

TMR1H = 0xD1;

count ++;

if (count >= 1000) //measure upto 1000 ms i.e. 1 seconds

{

Buzzer = ~Buzzer; /\* Toggle buzzer pin \*/ count = 0; //reset count

}

TMR1IF = 0; //timer1 overflow flag to 0

}

}

void main()

{

TRISB=0; /\* Set as output port \*/ TRISAbits.TRISA5 = 0; //set buzzer pin RA5 as output GIE=1; /\* Enable Global Interrupt \*/

PEIE=1; /\* Enable Peripheral Interrupt \*/

TMR1IE=1; /\* Enable Timer1 Overflow Interrupt \*/ TMR1IF=0;

/\* Enable 16-bit TMR1 register,no pre-scale,internal clock, timer OFF \*/ T1CON=0x80; /\*1:8 prescale\*/

TMR1L = 0x20; TMR1H = 0xD1;

TMR1ON=1; /\* Turn ON Timer1 \*/

while(1);

}

**Expt 8:Write an Embedded C program for External Interrupt input switch press, output at Relay.**

## **Aim**: To write a C program to interface PIC18F4550 to Relay and switch it ON/OFF using input from external switch. Use ISR programming for External Interrupt.

**Experimental Setup:** MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure**:

**Step1**: Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2**: Write the program in C language for interfacing Relay to PIC18F4550, using External Interrupt ISR. **(in program properties make sure to add the 0x800 offset)**

## **Step3**: Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4**: Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5**: Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6**: Press reset button and execute the program.

**Result**: Check if the Relay is switching ON/OFF when external interrupt switch is pressed and the ISR is getting executed .

**Program:**

#include <pic18f4550.h>

#define RELAY\_PIN LATAbits.LATA4

void interrupt extint\_isr(void)

{

unsigned int i; if(INT1F)

{

INT1F = 0;

INT1IE = 0;

RELAY\_PIN = ~RELAY\_PIN;

for(i=0; i<10000; i++); //small delay for debouncing INT1IE = 1;

}

}

int main()

{

ADCON1 = 0x0F; //set pins as Digital TRISAbits.TRISA4 = 0; //set relay pin RA4 as output TRISBbits.TRISB1 = 1; //Interrupt pin as input RELAY\_PIN = 1;

INT1IE = 1; //Enable external interrupt INT1

INTEDG1 = 0; //Interrupt on falling edge

GIE = 1; // Enable global interrupt

while(1);

}

**Expt 9:Write an Embedded C program for LCD interfacing with PIC18Fxxx.**

## **Aim**: To write a C program to interface PIC18F4550 to 16x2 Character LCD.

**Experimental Setup**: MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure**:

**Step1**: Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2**: Write the program in C language for interfacing 16x2 LCD to PIC18F4550.

**(in program properties make sure to add the 0x800 offset)**

**Step3**: Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4**: Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5**: Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6**: Press reset button and execute the program.

**Result**: Check if the characters are getting printed on the LCD screen .

**Program:**

#include <p18f4550.h>

#define LCD\_EN LATAbits.LA1 #define LCD\_RS LATAbits.LA0 #define LCDPORT LATB

void lcd\_delay(unsigned int time)

{

unsigned int i , j ;

for(i = 0; i < time; i++)

{

for(j=0;j<100;j++);

}

}

void SendInstruction(unsigned char command)

{

LCD\_RS = 0; // RS low : Instruction LCDPORT = command;

LCD\_EN = 1; // EN High lcd\_delay(10);

LCD\_EN = 0; // EN Low; command sampled at EN falling edge lcd\_delay(10);

}

void SendData(unsigned char lcddata)

{

LCD\_RS = 1; // RS HIGH : DATA

LCDPORT = lcddata;

LCD\_EN = 1; // EN High lcd\_delay(10);

LCD\_EN = 0; // EN Low; data sampled at EN falling edge lcd\_delay(10);

}

void InitLCD(void)

{

ADCON1 = 0x0F;

TRISB = 0x00; //set data port as output TRISAbits.RA0 = 0; //RS pin TRISAbits.RA1 = 0; // EN pin

SendInstruction(0x38); //8 bit mode, 2 line,5x7 dots SendInstruction(0x06); // entry mode SendInstruction(0x0C); //Display ON cursor OFF SendInstruction(0x01); //Clear display SendInstruction(0x80); //set address to 1st line

}

unsigned char \*String1 = " Microembedded"; unsigned char \*String2 = " PIC-18F Board";

void main(void)

{

ADCON1 = 0x0F;

TRISB = 0x00; //set data port as output TRISAbits.RA0 = 0; //RS pin

TRISAbits.RA1 = 0; // EN pin

SendInstruction(0x38); //8 bit mode, 2 line,5x7 dots SendInstruction(0x06); // entry mode SendInstruction(0x0C); //Display ON cursor OFF SendInstruction(0x01); //Clear display SendInstruction(0x80); //set address to 1st line

while(\*String1)

{

SendData(\*String1);

String1++;

}

SendInstruction(0xC0); //set address to 2nd line while(\*String2)

{

SendData(\*String2);

String2++;

}

while(1);

}

**Expt 10: Write an Embedded C program for generating PWM signal for DC/Servo motor on PIC18Fxxx.**

## **Aim**: To write a C program to interface PIC18F4550 to DC motor and varying speed using PWM signal generation.

**Experimental Setup**: MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure**:

**Step1**: Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2**: Write the program in C language for interfacing DC motor to PIC18F4550 and varying speed using PWM . **(in program properties make sure to add the 0x800 offset)**

## **Step3**: Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4**: Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5**: Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6**: Press reset button and execute the program.

**Result**: Check if the DC motor speed varies .

**Program:**

#include<p18f4550.h>

unsigned char count=0; bit TIMER,SPEED\_UP;

void timer2Init(void)

{

T2CON = 0b00000010; //Prescalar = 16; Timer2 OFF

PR2 = 0x95; //Period Register

}

void delay(unsigned int time)

{

unsigned int i,j; for(i=0;i<time;i++)

for(j=0;j<1000;j++);

}

void main(void)

{

unsigned int i;

TRISCbits.TRISC1 = 0; //RC1 pin as output

TRISCbits.TRISC2 = 0; //CCP1 pin as output LATCbits.LATC1 = 0;

CCP1CON = 0b00111100; //Select PWM mode; Duty cycle LSB CCP1CON<4:5> = <1:1>

CCPR1L = 0x0F; //Duty cycle 10%

timer2Init(); //Initialise Timer2

TMR2ON = 1; //Timer2 ON

while(1) //Loop forever

{

for(i=15;i<150;i++)

{

CCPR1L = i;

delay(100);

}

for(i=150;i>15;i--)

{

CCPR1L = i;

delay(100);

}

}

}

**Expt 11: Write an Embedded C program for PC communication using serial interface (UART).**

## **Aim**: To write a C program to interface PIC18F4550 to PC using serial communication and transmit / receive characters over it.

**Experimental Setup**: MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure**:

**Step1**: Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2**: Write the program in C language for interfacing PC to PIC18F4550 and sending ascii characters over serial communication. **(in program properties make sure to add the 0x800 offset)**

## **Step3**: Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4**: Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5**: Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6**: Press reset button and execute the program.

**Result**: connect the PC to the board using the USB cable. Start a terminal program on the PC (tera Term, Putty, hyperterminal) with the specified baud rate (9600). Check if you are getting the transmitted characters from the board and back .

**Program:**

#include<p18F4550.h> #include<stdio.h> #define Fosc 48000000UL

void InitUART(unsigned int baudrate)

{

TRISCbits.RC6 = 0; //TX pin set as output

TRISCbits.RC7 = 1; //RX pin set as input

//Non-inverted data; 8-bit baudrate generator

SPBRG = (unsigned char)(((Fosc /64)/baudrate)-1); BAUDCON = 0b00000000 ;

//Asynchronous 8-bit; Transmit enabled; Low speed baudrate select TXSTA = 0b00100000;

//Serial port enabled; 8-bit data; single receive enabled RCSTA = 0b10010000; }

void SendChar(unsigned char data)

{

while(TXSTAbits.TRMT == 0); //Wait while transmit register is empty

TXREG = data; //Transmit data

}

void putch(unsigned char data)

{

SendChar(data);

}

unsigned char GetChar(void)

{

while(!PIR1bits.RCIF); //Wait till receive buffer becomes full return RCREG; //Returned received data

}

void main(void)

{

InitUART(9600);

printf("\r\nHello MicroPIC-18F: Enter any Key from Keyboard\r\n"); while(1)

{

printf("%c! ",GetChar()); //Receive character from PC and echo back

}

while(1);

}

**Expt 12: Write an Embedded C program for interfacing PIC18FXXX to Temperature sensor interfacing using ADC & display on LCD**

## **Aim:** To write a C program to interface PIC18F4550 to a temperature sensor

(LM35) and display the temperature on LCD.

**Experimental Setup:** MicroPIC18F board, USB cable, Power supply adaptor, MPLABx IDE, PICLoader software.

**Procedure:**

**Step1:** Open MPLABX IDE on the PC for program development and create a new project and save it in a new folder.

**Step2:** Write the program in C language for interfacing temperature sensor (LM35) to PIC18F4550 and display result on LCD. **(in program properties make sure to add the 0x800 offset)**

## **Step3:** Build the program and create hex file. In case of errors correct program and rebuild to create hex file.

**Step4:** Prepare the experimental setup by connecting the MicroPIC18F board to the PC using USB cable. Power ON the Board. Check for the USBtoSerial COMx allocated by the PC.

**Step5:** Using the PICLoader Software flash the hex file in the PIC18F4550.

**Step6:** Press reset button and execute the program.

**Result:** Check if the temperature values are displayed on the LCD.

**Program:**

#include <pic18f4550.h> #include <stdio.h>

#define LCD\_EN LATAbits.LA1 #define LCD\_RS LATAbits.LA0 #define LCDPORT LATB

unsigned char str[16];

void lcd\_delay(unsigned int time)

{

unsigned int i , j ;

for(i = 0; i < time; i++)

{

for(j=0;j<100;j++);

}

}

void SendInstruction(unsigned char command)

{

LCD\_RS = 0; // RS low : Instruction LCDPORT = command;

LCD\_EN = 1; // EN High lcd\_delay(10);

LCD\_EN = 0; // EN Low; command sampled at EN falling edge lcd\_delay(10);

}

void SendData(unsigned char lcddata)

{

LCD\_RS = 1; // RS HIGH : DATA

LCDPORT = lcddata;

LCD\_EN = 1; // EN High lcd\_delay(10);

LCD\_EN = 0; // EN Low; data sampled at EN falling edge lcd\_delay(10);

}

void InitLCD(void)

{

ADCON1 = 0x0F;

TRISB = 0x00; //set data port as output TRISAbits.RA0 = 0; //RS pin TRISAbits.RA1 = 0; // EN pin

SendInstruction(0x38); //8 bit mode, 2 line,5x7 dots SendInstruction(0x06); //entry mode SendInstruction(0x0C); //Display ON cursor OFF SendInstruction(0x01); //Clear display SendInstruction(0x80); //set address to 0

}

void LCD\_display(unsigned int row, unsigned int pos, unsigned char \*ch)

{

if(row==1)

SendInstruction(0x80 | (pos-1)); else

SendInstruction(0xC0 | (pos-1));

while(\*ch)

SendData(\*ch++);

}

void ADCInit(void)

{

//ADC channel 7 input TRISEbits.RE2 = 1;

//Ref voltages Vdd & Vss; AN0 - AN7 channels Analog ADCON1 = 0b00000111;

//Right justified; Acquisition time 4T; Conversion clock Fosc/64 ADCON2 = 0b10101110;

}

unsigned short Read\_Temp(void)

{

ADCON0 = 0b00011101; //ADC on; Select channel;

GODONE = 1; //Start Conversion

while(GO\_DONE == 1 ); //Wait till A/D conversion is complete return ADRES; //Return ADC result

}

int main(void)

{

unsigned int temp;

InitLCD();

ADCInit();

LCD\_display(1,1,"Temperature:"); while(1)

{

temp = Read\_Temp();

temp = ((temp \* 500) / 1023); sprintf(str,"%d'C ",temp); LCD\_display(2,1,str); lcd\_delay(9000);

}

return 0;

}