

**Lab 8 Report**

By

**Michel Sfeir**

Presented to

Dr. Edgard Chammas

Report for Microcontrollers Lab

Faculty of engineering

University of Balamand

**How it was done:**

In Lab 8 we designed a graphical user interface for communicating between the computer and the EasyPic board. It included reading temperature and A/D results from the board, as well as the status of the switch RA4. It also allowed us to send commands from the PC to the board: Turning on and off all of PORTD’s LED lights and sending 2-line messages to the LCD. As an additional assignment we had to code in the toggling of a Buzzer whenever a checkbox in the GUI was pressed.

To do this, 3 steps were required:

* Firstly, the Setup function had to be changed. As I decided to run the buzzer using timer interrupts, the Setup function had to setup timer0, enable global interrupts, and setup RC2 as a digital output pin. For these purposes the following code was added to it:

ANSELCbits.ANSC2 = 0; // RC2 is a digital i/o

TRISCbits.TRISC2 = 0; // RC2 is a digital output pin

T0CON = 0b11010010; // Run internal clock with prescaler 1:8

INTCONbits.TMR0IE = 0; // Initially the timer interrupt is not enabled

INTCONbits.GIE = 1; // Global Interrupt Enable on

It is important to set TMR0IE = 0 at the start so that the buzzer is off by default, until the checkbox is pressed.

* Secondly, the interrupt function to turn on the buzzer must be coded. Since we are aiming for a 50% duty cycle, we do not need to have separate timers for turning RC2 on and off, and instead we can simply count once and toggle it.

**#pragma code ISR = 0x0008**

**#pragma interrupt ISR**

**void ISR(void){**

**TMR0L = 256 - 125; // f = 1/8 MHz, T = 8 us; T \* 125 = 1000 us \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//f = 1000 Hz/2 = 500 Hz**

INTCONbits.T0IF = 0; // Acknowledge interrupt

**PORTCbits.RC2 = !PORTCbits.RC2; // Toggle RC2 to create a square \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//\_\_\_wave with 50% duty cycle**

**}**

Each time, TMR0L will count 125 cycles, with each cycle being 8 micro seconds because of the 1:8 prescaler used above. This means an interrupt will happen every 1000 micro seconds, and toggle RC2. Since a period requires two interrupts, the period of this signals is 2000 micro seconds, meaning the frequency is the desired 500 Hz.

* Finally, when a use presses the checkbox we need to turn this whole process on. To do so we simply need to allow Timer 0 to do its interrupts. That is done with the following code:

**case BuzzerOnOff:**

**INTCONbits.TMR0IE = !INTCONbits.TMR0IE; // Toggle timer interrupt \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//\_every time the checkbox is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//\_pressed**

**break;**

With these changes in place, we could see that the buzzer turns on and off every time the checkbox is pressed

The functionality is demonstrated in this video on my youtube channel: <https://youtu.be/GpPlu7rF0zo>

**Full Code:**

Here is the full code used in this project:

#include    <p18cxxx.h>

#include    <LCD4lib.h>

#include    <Hyperterm.h>

void Setup(void);

void setupBuzzer(void);

void ReadLM35(void);

void ReadAnalogPot(void);

void BuzzerOn(void);

char txPacket[4];

char rxPacket[1 + 32];

enum cmds {

    SendData, GetLEDs, GetLCDdata, BuzzerOnOff

};

char T, V;

void main(void) {

    Setup();

    while (1) {

        while (!PIR1bits.RCIF);

        switch (RCREG) {

            case SendData:

                //Send data to the PC

                ReadAnalogPot();

                txPacket[0] = ADRESH;

                txPacket[1] = ADRESL;

                txPacket[2] = (PORTAbits.RA4) ? 1 : 0;

                ReadLM35();

                txPacket[3] = ADRESH;

                TxPacket(txPacket, 4);

                break;

            case GetLEDs:

                RxPacket(rxPacket, 1);

                PORTD = rxPacket[0];

                break;

            case BuzzerOnOff:

                INTCONbits.TMR0IE = !INTCONbits.TMR0IE; // Toggle timer interrupt every time the checkbox is pressed

                break;

            case GetLCDdata:

                //Receive LCD data and print it

                RxPacket(rxPacket, 32);

                DispVarStr(&rxPacket[0],Ln1Ch0,16);

                DispVarStr(&rxPacket[16],Ln2Ch0,16);

                break;

        }

    }

}

void Setup(void) {

    InitUART();

    InitLCD();

    ANSELD = 0x00; // Port D is a digital I/O

    TRISD = 0x00; // PORTD pins are output

    VREFCON0 = 0b10010000;

    ANSELCbits.ANSC2 = 0; // RC2 digital i/o

    TRISCbits.TRISC2 = 0; // RC2 digital output pin

    T0CON = 0b11010010; // Run internal clock with prescaler 1:8

    INTCONbits.TMR0IE = 0; // Initially the timer interrupt is not enabled

    INTCONbits.GIE = 1; // Global Interrupt Enable on

}

void ReadLM35(void) {

    ADCON0 = 0b00011001;

    Delay10KTCYx(5);

    ADCON1 = 0b00001000;

    ADCON2 = 0b00100101;

   ADCON0bits.GO = 1;

   while (ADCON0bits.GO);

}

void ReadAnalogPot(void) {

    ADCON0 = 0b00001001;

    Delay10KTCYx(5);

    ADCON1 = 0b00000000;

    ADCON2 = 0b10100101;

    ADCON0bits.GO = 1;

    while (ADCON0bits.GO);

}

#pragma code ISR = 0x0008

#pragma interrupt ISR

void ISR(void){

    TMR0L = 256 - 125; // f = 1/8 MHz, T = 8 us; T \* 125 = 1000 us; f = 1000 Hz/2 = 500 Hz

    INTCONbits.T0IF = 0; // Acknowledge interrupt

    PORTCbits.RC2 = !PORTCbits.RC2; // Toggle RC2 to create a square wave with 50% duty cycle

}