Ultrasonic Project: Formal Element

By Kenneth Mallabo

Last Updated: 24/05/2023

# Introduction

This report is about using the PIC16F877A, a switch, an LCD and an HCSR04 ultrasonic sensor. The goal is to measure the distance of an object using the ultrasonic sensor both in simulation and the real world. The switch is only used in the simulation to act as an object that has been placed in front of the sensor. The LCD module is used to display the distance of the object detected by the sensor. The PIC16F877A microcontroller uses C code to program all the components together.

# Techniques

There are two techniques used to calculate the measurements from the ultrasonic sensor. One uses the Timer 0 technique to count and the other uses Timer 1.

The Timer 0 technique was actually taken from this webpage (<https://www.rhydolabz.com/wiki/?p=895>). It uses a similar setup to what I needed but uses UART data to display the distance on the connected PC through HyperTerminal and it uses a different ultrasonic sensor that uses 3 pins instead of the four used for this project. Obviously, there needed to be changes to make this work. To summarise how this works, once the switch has been turned on it will trigger a pulse signal of 10 microseconds to the sensor. The timer 0 counter will reset to zero, then the sensor will echo back a response signal to the microcontroller whilst the timer 0 counter will increment until the pulse signal is gone. Afterwards, it will calculate distance using the timer 0 counter and then display the result on the LCD. This technique is the most accurate as it not only correctly displays distance, but also does so use a float to show the millimetre distance too.

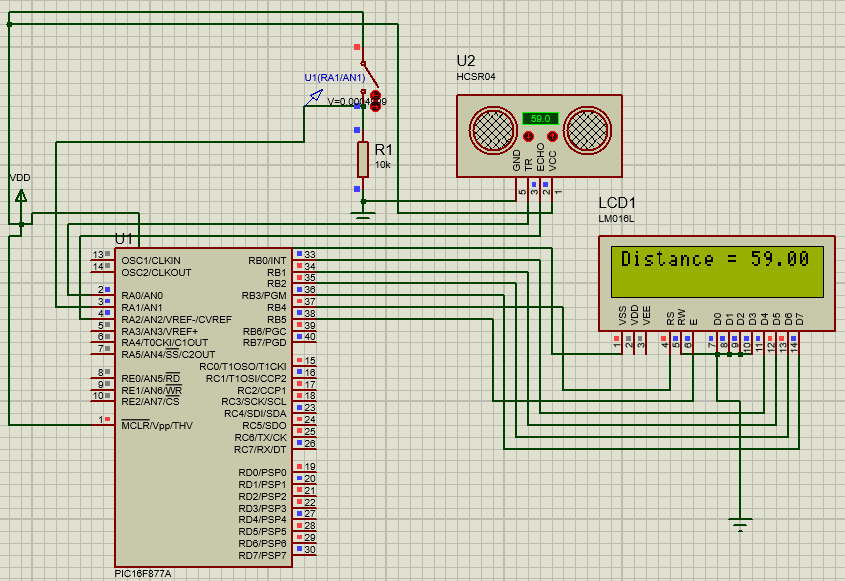
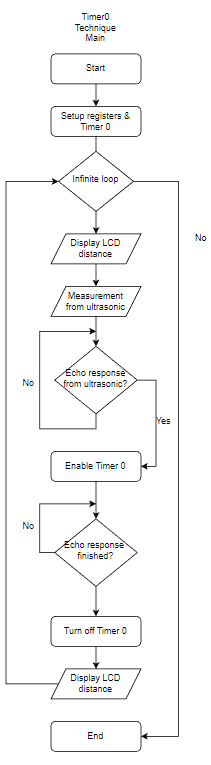


Figure : Timer 0 technique in Proteus



The Timer 1 technique was achieved using the example code called “Simple\_delay\_Ultrasonic\_V1” which was provided by Lecturer Tony Grennan on Brightspace. This code originally used the PIC16F877 which had a T1G pin which is necessary for using the Timer 1 gate control. Unfortunately, for the purposes of using the actual hardware for use in the lab, I am instead using the PIC16F877A which does not have the needed pin to perform this method. Interrupts are also not used. How this technique works is very similar to the timer 0 technique, the difference is that it uses the timer 1 counter. The prescale value used is 1:8, the internal clock source is to be divided by 4 and since the clock frequency of the PIC16F877A is 20Mhz then the input frequency is 5Mhz. The accuracy of this technique is not great in comparison to the previous technique as it is usually off by 1 cm or even more if using higher values than 60 cm.

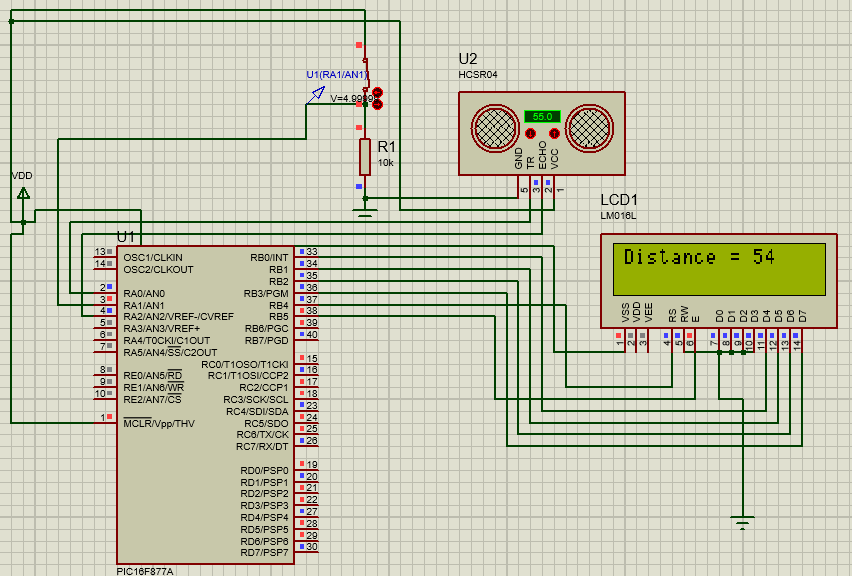
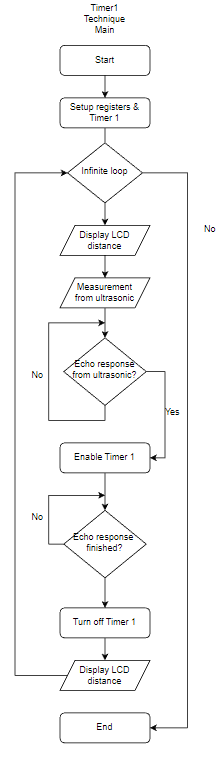


Figure : Timer 1 technique in Proteus



# Uploading to Hardware

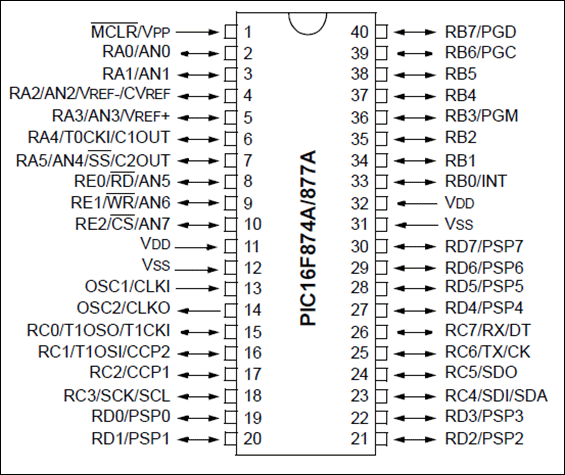
1. Connect the PICmicro MCU Multiprogrammer EB006-006-00-9 that is attached with a PIC16F877A microcontroller to the Matrix Dev Board EB083 via both of their Port A’s and Port B’s.
2. Place the HCSR04 sensor on a breadboard and then connect it to the EB083 via jumper wires. Refer to figure 3 to connect the appropriate pins together.

Figure : PIC16F877A pin diagram (source: https://components101.com/microcontrollers/pic16f877a-pin-diagram-description-features-datasheet)

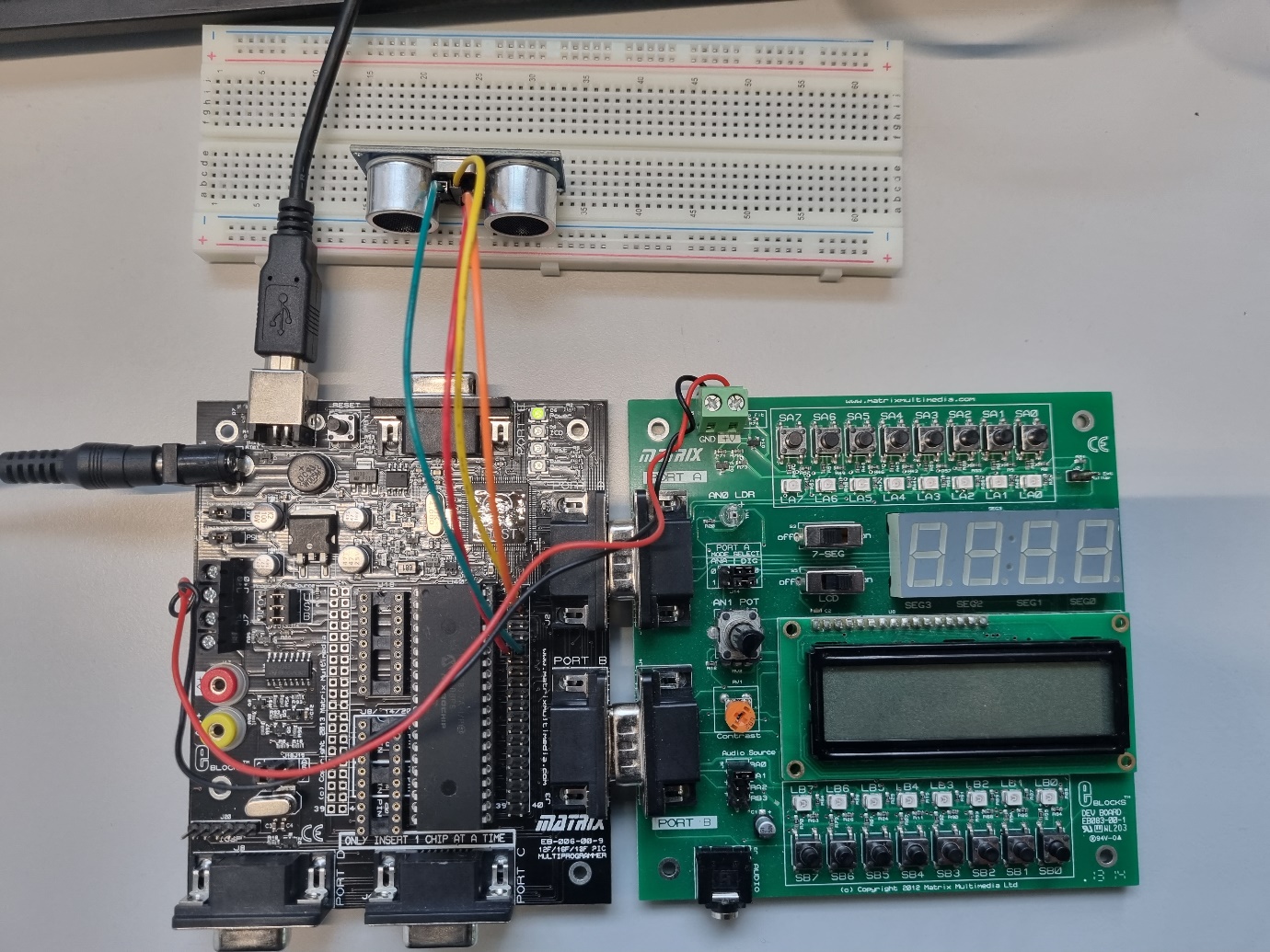
1. Connect a 12V power supply unit to the EB006-006-00-9 and then a USB connection between the EB083 and a PC. The physical setup should look like figure 4. 

Figure : Physical ultrasonic project on the lab

1. Next, we need the hex file of the program. This can be retrieved by going to Proteus and then building or running the program. After that, right-click on the microcontroller, select “Edit Properties”, click on the folder icon and then copy the “debug.hex” file. Paste it somewhere where you could easily access it.
2. Open the mLoader program in order to upload the hex code to the EB006-006-00-9. Enter the full file path of the hex code for the “File Name”. Click on the “Autodetect” button.
3. Finally click “Send” and if it had worked then the final step was to press “Execute”.
4. The LCD should be showing the distance between a detected object and the Ultrasonic sensor however, in Figure 5 the value is stuck on 0 despite a mouse being placed nearby it.

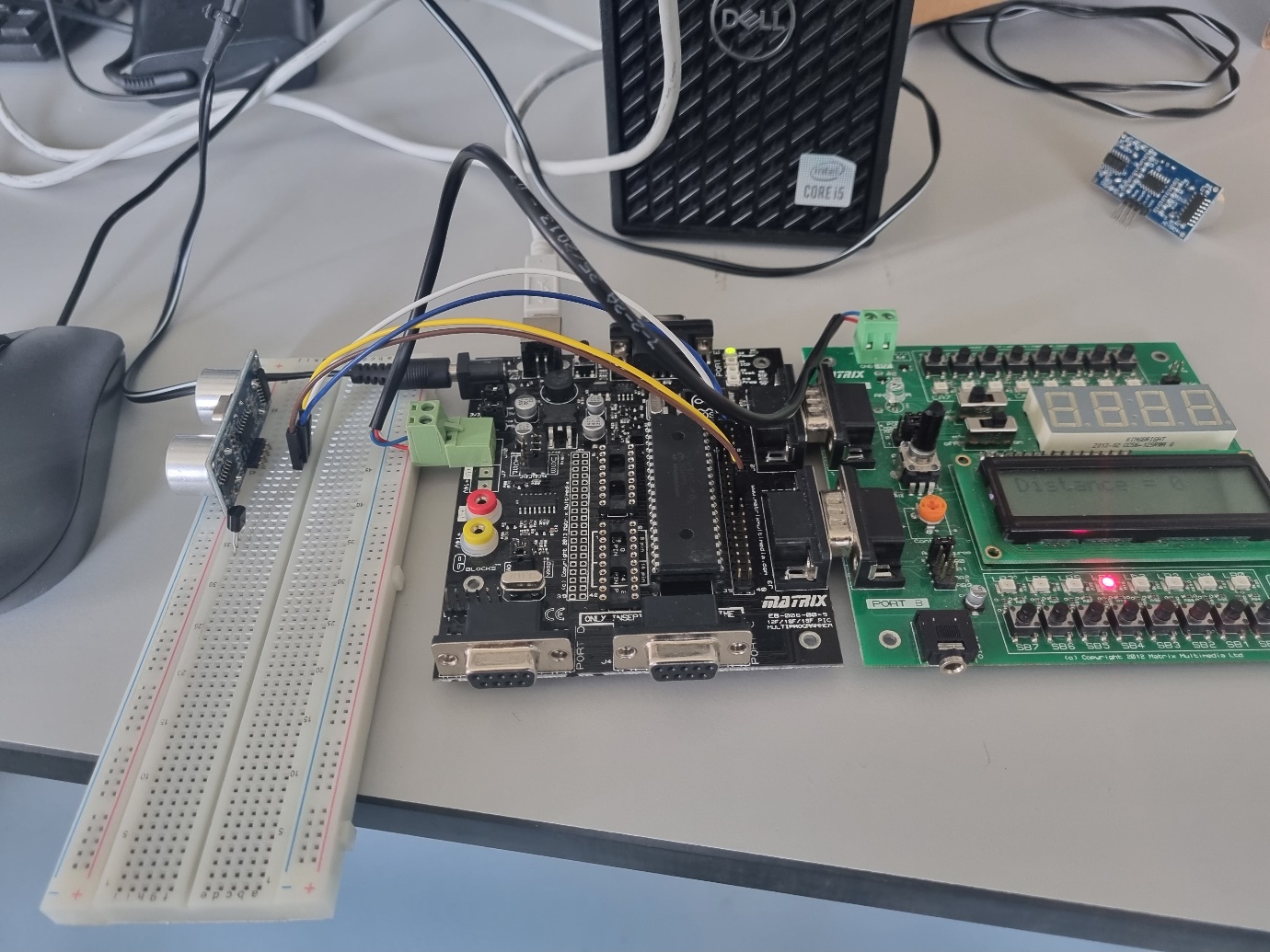


Figure : LCD not properly displaying sensor values

# References

microdigisoft. (2021). *[PIC16F877A]: Timers in PIC Microcontroller and Application* [online]. Available from: <https://microdigisoft.com/pic16f877a-timers-in-pic-microcontroller-and-application/> [accessed 2 April 2023].

rhydoLABZ. (2013). *Measure Distance with UltraSonic Distance Sensor (PWM O/P)* [online]. Available from: <https://www.rhydolabz.com/wiki/?p=895> [accessed 3 April 2023].

# Appendix

// CONFIG

#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR disabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; low-voltage programming enabled)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

#include <xc.h>

#include <stdio.h>

#include <stdlib.h>

#include <iso646.h>

#include <stdint.h>

#include <pic16f877a.h>

#define \_XTAL\_FREQ 19660800 //THIS SHOULD BE CHANGED TO 19660800 Hz for the lab boards, 20000000 for proteus

#include<pic.h>

// lcd initialise

#define RS RB4

#define EN RB5

#define D4 RB0

#define D5 RB1

#define D6 RB2

#define D7 RB3

#include "lcd.h";

// pwm define

#define START\_PULSE\_TIMER() T0CS=0 //echo wait PULSEIN timer start bit

#define STOP\_PULSE\_TIMER() T0CS=1 //echo wait PULSEIN timer stop bit

#define PORT\_CONFIG() TRISC=0;TRISD=0

#define INTERRUPT\_ENABLE() GIE=1; PEIE=1; T0IF=0; T0IE=1

//pwm variables

double PulseInTime=0; //variable calculating the echo travel time

double cmDistance; //variable calculating the distance within limits

unsigned int tDistance; //temperary variable for holding the distance value

unsigned int TimeCnt=0; //timer0 overflow count

char DArr[8]; // array holding the calculated distance for display

// lcd variables

int j;

int i;

int a;

long distance; // variable where the the reflection time of the ultrasound is stored

\_\_CONFIG( FOSC\_HS & WDTE\_OFF & PWRTE\_OFF & CP\_OFF & BOREN\_ON & LVP\_OFF & CPD\_OFF & WRT\_OFF & DEBUG\_OFF);

unsigned char counter,clk\_cnt=0,clk\_sec=0,seconds=0,up\_date=0;

void setup\_hardware (void)

{

PORTA=0x00; // switch & ultrasonic

TRISB = 0x00 ; // set all of PORTB for output

//TRISBbits.TRISB0=1; // But make least significant bit an input for the interrupt input if using that

TRISA=0xFE; // set PORTA for inputs originally FE

//TRISC0=TRISC1=TRISC2=0;

ADCON1 =0xFE; // enable portA as analogue

INTCONbits.GIE=1; //Enable the interrupt system

INTCONbits.INTE=1; //

OPTION\_REGbits.INTEDG=0; // 0 or 1 depending on what 'edge' of signal at RBo

INTCONbits.PEIE=1; // The peripheral register enable bit that applies to all 'other' peripherals

// that can not be processed through INTCON directly such as TIMER1 and use PIE/PIR

//TIMER 0 SET\_UP Comment out if using timer 1

//INTCONbits.TMR0IE=0;

//INTCONbits.TMR0IF=0;

//INTCONbits.PEIE=1; //not required for the basic TO but T1 would require this

//TIMER 1 SET\_UP Comment out if using timer 0

PIE1bits.TMR1IE=1;

PIR1bits.TMR1IF=0;

T1CON=0B00000000;

T1CONbits.T1CKPS0=1; // THE BITS FOR PRESCLAER SET TO 11

T1CONbits.T1CKPS1=1; // which is 8

T1CONbits.TMR1CS=0; // THIS BIT ENSURES TIMER CLOCK IS SYSTEMS Fosc/4

OPTION\_REGbits.PS0=1; //option register set

OPTION\_REGbits.PS1=1;

OPTION\_REGbits.PS2=1;

OPTION\_REGbits.PSA=0;

OPTION\_REGbits.T0CS=0;

//the following lines relate to the external interrupt if it was being used

//OPTION\_REGbits.INTEDG=0; // The interrupting signal is caused by a 0 to 1 transition (edge triggered)

// so the INT EDGE select bit is set to 1. A '0' in this option bit would allow a 1 to 0 EDGE

// to cause the interrupt.

//INTCONbits.INTF=0; // Clear the external interrupt flag as a precaution

//INTCONbits.INTE=1; // set the ENABLE bit to activate external interrupt

//INTCONbits.GIE=0 // I have disabled interrupts

//; // THE GLOBAL INTERRUPT ENABLE/MASK BIT MUST BE SET to allow any CPU interrupt

// NOW BEFORE 'TURNING ON' TIMER 1 MODULE TO START COUNTING...

// LOAD A VALUE TO START COUNT FROM (NOT JUST 0000 TO FFFF).

// ASSUMING THAT Fosc= 20MHz which is divided by 4 and pre-scaler 8....

// (4/20e6)\*8 = 1.6 microseconds i.e the period of clock signals applied to TIMER 1.

//T1CONbits.TMR1ON; // So for 1 second this would mean 1sec/1e-6 = 625,000 counts..obviously multiple overflows

// of a 16 bit count (max single overflow of 65,536). So either use a lower frequncy XTAL...

// or as before, with TIMER 0 programs, 'count off' a number of interrupts. Clearly you require

// an integer value (no fractional/decimal) and must remember the max value of a single count is 65,536.)

// dividing 625,000 by 10 gives 62,500 which is an integer value less than 65,536.

TMR1L=0xDC; // SO THAT MEANS YOU LOAD -62,500 (2'S COMPLEMENT) WHICH IS hex 0BDC.

TMR1H=0x0B;

//T1CONbits.TMR1ON=1; //And now turn on TIMER 1 TO START COUNT

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE FOLLOWING IS FOR THE TIMER 1 GATE CONTROL.....there are a lot of registers that need to be carefully checked

//T1CONbits.T1GINV=1; //This is in the T1 CONTROL REG and if '1' then TIMER 1 COUNTS WHEN INPUT AT GATE =1

//T1CONbits.TMR1GE=1; // This 'gate enables' i.e. the timer count is controlled by the gate at pin PORT B 5

//Also, comparator C2 control register has a bit which must be set to 1 for the T1 Gate input to be from the pin (T1G)

//CM2CON1bits.T1GSS=1;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

// Interrupt function Remember there is only 1 function invoked by an 'interrupt' even though there are various possible sources

// of interrupt. This example has only 1 used, the timer 0 (often it and external interrupt are used)

//The following line is a generic name for ISR routine

//void \_\_interrupt() interruptServiceRoutine(void)

//{

// }

//But below it is named for the timer since we are only using that

// Interrupt function

// NOTE:Comment out the interrupt function if using timer 0, otherwise don't if using timer 1

void \_\_interrupt()timer\_isr(void)

{

if(PIR1bits.TMR1IF==1) // THIS IF CLAUSE IS TRUE IF THE INTERRUPT CAME FROM TIMER 1

{

TMR1L=0xDC; //Reset the timer for the next count

TMR1H=0x0B;

PIR1bits.TMR1IF=0; //Reset the flag in the PIR register

clk\_cnt++;

if(clk\_cnt==7) // originally 20?--> // Check for the 7th interrupt? NOTE!!!!

{ // Yes it was set to 20 to 'slow down' the demo to 2 seconds for observation purposes

clk\_cnt=0; //Reset

clk\_sec++; // count off another second

up\_date=1; // message flag to main routine

}

}

// The simple purpose of the external Interrupt is to reset SECONDS to zero

else // If it was not the TIMER that interrupted

{ INTCONbits.INTF=0; // It must be EXTERNAL INTERRUPT SO CLEAR THAT FLAG

T1CONbits.TMR1ON=0; // Turn off timer 1 for a moment...possibly could be done first

clk\_sec=0; // Clear seconds

clk\_cnt=0; // and reset this to ensure it starts from a true zero condition

up\_date=1; // message flag to main routine as we want this 'zero' to be displayed

TMR1L=0xDC; //Reset the timer for the next count

TMR1H=0x0B;

T1CONbits.TMR1ON=1; //And turn on the TIMER 1 again to continue

} // You could just have cleared the TIMER flag and cleared seconds...but the output will probably

} // never show zero or only very momentarily...this is a 'full reset' on interrupt switch press

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : Delay10Us

Description : Delay function for 10 micro-sec

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay10Us()

{

int dCnt;

for(dCnt=0;dCnt<3;dCnt++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : Delay\_100MicroSec

Description :Delay function for 100 micro-sec

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_100MicroSec()

{

int dCnt; // delay count

for(dCnt=1;dCnt<35;dCnt++); // 100Us approx

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : DelayMs

Description : Delay function for n milli-sec, n=1,2,3....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DelayMs(unsigned char Cnt)

{

int dCnt;

while(Cnt>0)

{

for(dCnt=0;dCnt<10;dCnt++)

Delay\_100MicroSec();

Cnt--;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : TIMER0\_INIT

Description :Timer0 initialization

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TIMER0\_INIT()

{

PSA =1; // set timer0 minimum prescale

TMR0=0; // timer0 count register

T0CS=1; // timer0 start bit

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : TriggerPulse

Description :Generating pulse for sensor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TriggerPulse()

{

RA0=0;

TRISA0=0; // pulse signal line made O/P for trigger

Delay10Us();

RA0=1; // H\_to\_L transition. ie,

//Trigger Pulse to the Ultrasonic range finder module

Delay10Us();

RA0=0;

TimeCnt=0; // clear timer overflow count

PulseInTime=0;

TMR0=0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : PulseInMode

Description :Calculating & displaying distance

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PulseInMode()

{

while(!RA2); // wait for PulseIn signal to go HIGH

START\_PULSE\_TIMER(); // start echo PulseIn timer T0CS=0

while(RA2); // wait for PulseIn signal to go LOW

STOP\_PULSE\_TIMER(); // stop echo PulseIn timer T0CS=1

PulseInTime= TMR0; // timer calculations [return time in Us]

PulseInTime= (double)( ( (PulseInTime\* 0.2) + (TimeCnt \* 51.2)) );

cmDistance = (double) PulseInTime/58; // timer value in Us

// divided by 58 gives the cm distance

tDistance =(unsigned int)cmDistance; // display of the calculated distance

// lcd header

Lcd\_Clear();

char str[20];

Lcd\_Clear();

sprintf(str, "Distance = %f", cmDistance);

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String(str);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function : interrupt isr

Description :interrupt service

NOTE: Comment this function out if using timer 1!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

void interrupt isr()

{

if(T0IF==1) // timer overflow interrupt

{

T0IF=0; // clear timer interrupt flag

TimeCnt++; // timer overflow count increment

}

}

\*/

void main(void)

{

setup\_hardware() ; //initialise the various registers used

seconds = 0;

Lcd\_Init(); // lcd header

PORT\_CONFIG(); // PORT configurations

TIMER0\_INIT(); // Timer initialisation

INTERRUPT\_ENABLE(); // Interrupt enable

Lcd\_Clear();

char str[20];

Lcd\_Clear();

sprintf(str, "Distance = %f", cmDistance);

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String(str);

while(1)

{

/\*\*\*\*pulse width modulation technique source: https://www.rhydolabz.com/wiki/?p=895

\*\*\*\*comment or uncomment code for either technique\*\*\*/

/\*

while(RA1==0) // comment this loop out if using real hardware

{

// do nothing while the switch is off

}

TriggerPulse(); // Trigger mode

PulseInMode(); // PulseIn mode

DelayMs(250); //wait before next cycle

DelayMs(250); //wait before next cycle

while(RA1==1); // comment this loop out if using real hardware

tDistance = 0;

}

\*/

/\*\*\*timer 1 technique & delay timing technique

\*\*\*\*comment or uncomment code for either technique\*\*\*/

clk\_sec=0;

distance = 0;

Lcd\_Clear();

char str[80];

sprintf(str, "Distance = %d", clk\_sec);

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String(str);

//while(RA1==0) // comment this loop out if using real hardware

//{

// do nothing while swich is off

//} //WAIT FOR SWITCH PRESS

RA0=1; //PULSE TRIGGER FOR 10 US

\_\_delay\_us(10);

RA0=0; // TURN OFF PULSE TRIGGER

while(RA2==0); //WAIT FOR RA2 TO GO HIGH

while(RA2==1)

{

TMR1IF=1; // turn on counter for timer1

}

TMR1IF=0; // turn off counter for timer1

Lcd\_Clear();

sprintf(str, "Distance = %d", clk\_sec);

Lcd\_Set\_Cursor(1,1);

Lcd\_Write\_String(str);

\_\_delay\_us(10000);

//while(RA1==1) // comment this loop out if using real hardware

//{

// do nothing else while swich is on

//}

}

}