CMPE423 Team Project Report

Date of Submission:..........................

Submitted to: Dr. Mehmet Bodur

Submitted by Team Leader: Mahdy MOHAMED Nasr 147813

member-2: ADEMOLA OPEMIPO MAKINDE 139397

member-3: CHARLES BROWN T. DHLIWAYO 147118

member-4: M.ALI H. KHONSARI 15500208

member-5: ÇAĞRI UĞUREL 132034

Hereby signing this sheet the team leader and members declare that they know “*plagiarism is a serious crime listed in disciplinary bylaws*”, and this report does not contain plagiarism.

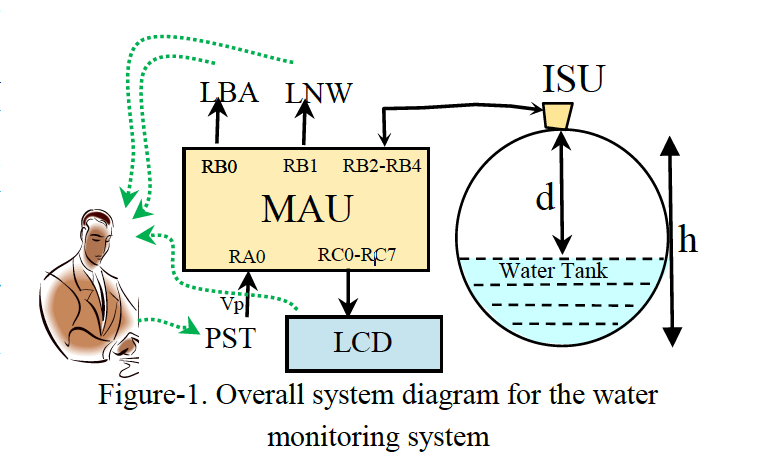
sign-1..........................

sign-2.......................... sign-3 ..........................

sign-4 .......................... sign-5 ..........................

# preliminary report

The main aim of the project is to control a water tank system in such a way to run an alarm in required situation. The solution system shall measure the water level in a cylindrical water tank that has a fixed known height and do some calculation to decide for alarm or not depending on the adjustable specified level. The project has multiple usage and mostly it will be beneficial and suitable for ordinary houses and factories especially to those who live in a developing countries and the action of bringing water takes long time. It will also help a lot the agriculture industries because primary role of water there.



This project will require a number of components to accomplish this task. The system consist of Two controllers, one to control a sensor that detect a water level and one for the main unit. Also it needs LCD display to show the accurate level of the water in tank every 5 seconds. A potentiometer to adjust the critical level. And to show the alarm and the proper notification of the overall system working it will require two LEDs. One for blink alive the will run 0.1s every second and one for alarm that will run if it necessary to do that.

For microcontrollers, PIC18F452 will be used for both main unit and sensor unit because of its power, efficiency and cost. HC - SR04 sensor will be responsible for reading the current water level. LM016L LCD is selected as a display for the water level. POT-HG (potentiometer) will be used as input for required critical water level.

The project also will require some software tools for this purpose. CC8 compiler will used for compiling C code that will be produced via software engineers and convert it to hex file that can run on a microcontroller. Also the Proteus 8 software and its designing tool (ISIS) will be used for designing the circuits and for the simulation. A machine that run a windows 7 or higher operating system is needed to for software engineers and to run the simulation.

Regarding human recourses, the system will require at least two software engineers with 40$/h minimum to program the microcontrollers also will need 2 system testing engineer for system verification with 35 $/ h. the project will need a software manager to take care of the project progress and to assign the tasks and to spot the work.

Table of Contents

[preliminary report 2](#_Toc502141664)

[Introduction 5](#_Toc502141665)

[The intelligent sensor unit (ISU) 6](#_Toc502141666)

[Task A 6](#_Toc502141667)

[Task B 7](#_Toc502141668)

[Task C 9](#_Toc502141669)

[Task D 10](#_Toc502141670)

[Task E 14](#_Toc502141671)

[Task G 15](#_Toc502141672)

[Task H 15](#_Toc502141673)

[Task I 18](#_Toc502141674)

[The main unit (MAU) 19](#_Toc502141675)

[1-making a blink alive circuit: 19](#_Toc502141676)

[2-Working with LCD unit 21](#_Toc502141677)

[3-UART Unit for receive and transmit ASKI characters. 23](#_Toc502141678)

[4-Connecting LCD and print received string from ISU 25](#_Toc502141679)

[5-Reading analog input from potentiometer 27](#_Toc502141680)

[6-Reading water level from ISU unit 29](#_Toc502141681)

[7-Turn on no water LED 30](#_Toc502141682)

[Over all System 31](#_Toc502141683)

[Circuit: 31](#_Toc502141684)

[Code of ISU: 32](#_Toc502141685)

[Code of MAU: 35](#_Toc502141686)

[Test: 39](#_Toc502141687)

[Conclusion 40](#_Toc502141688)

# **Introduction**

The system is divided on two main tasks to accomplish what is required :

1. **building the intelligent sensor unit (ISU) :**

First task (ISU) unit will be responsible for the water tank. It will be connected to the sensor that reads the water level from water tank and convert the readings to a meaningful unit (mm) and send this to main unit via UART methodology. It has to deal with the ultrasonic configuration, trigger the sensor periodically (every 5 seconds) and send the result after converting it to main unit.

1. **The main unit (MAU):**

While the second main task (MAU) will take the overall responsibilities. It will be responsible to power up the ISU unit and to read from it the readings. It will control the LEDS for blink alive and for alarm depend on the readings. It will configure the LCD and will display the result from ISU unit. It will read the input from the user via potentiometer and convert the input to a digital number that represent the specified level and can be compared to the readings from ISU unit.

# **The intelligent sensor unit (ISU)**

## Task A

The ISU unit will be used to build a simulation of prototype for an ultrasonic range sensor using the available HC-SR04 ultrasonic range sensors. At every 10 second period, the sensor shall get the pulse width of HC-SR04 output, convert it to the distance in millimeters, and transmit it serially as an EIA232 TTL level 9600 Baud ASCII 4-digit decimal number and an end of line (ASCII code 0Ah) character to a terminal. The unit is made up of the following components, in the table below:

|  |  |  |
| --- | --- | --- |
| Component Name | Value | Quantity |
| Virtual Pattern Generator | 100R | 1 |
| PIC18F252 |  | 1 |
| ISU Unit | PIC18F255 | 1 |
| Virtual Terminal |  | 1 |
| LED | 10mA | 1 |
| Oscilloscope |  | 1 |
| 4 MHz Crystal | - | 1 |

**Wiring of the sensor:**

The four pin male header provides the following terminals:

1 5V Supply

2 TRIG Trigger Pulse Input

3 ECHO Echo Pulse Output

4 0V Ground

**Electric Parameter:**

Working Voltage DC 5V

Working Current 15Ma

Workıng Frequency 40Hz

Max Range 4m

Min Range 2cm

Measuring-Angle 15 degree

Trigger Input Signal 10Us TTL pulse

Dimension 45\*20\*15mm

Terminalls Vcc, Trig, Echo, GND

## Task B

The following is a test circuit, that tests the simulation of HC-SR04 using a button swith to apply a pulse to TRIG input, and an oscilloscope to observe the duration of ECHO pulse by setting the trigger to negative edge of TRIG signal.

Change the clock time of the pattern generator in the range of 10us – 5ms at least for 10 evenly distributed values.

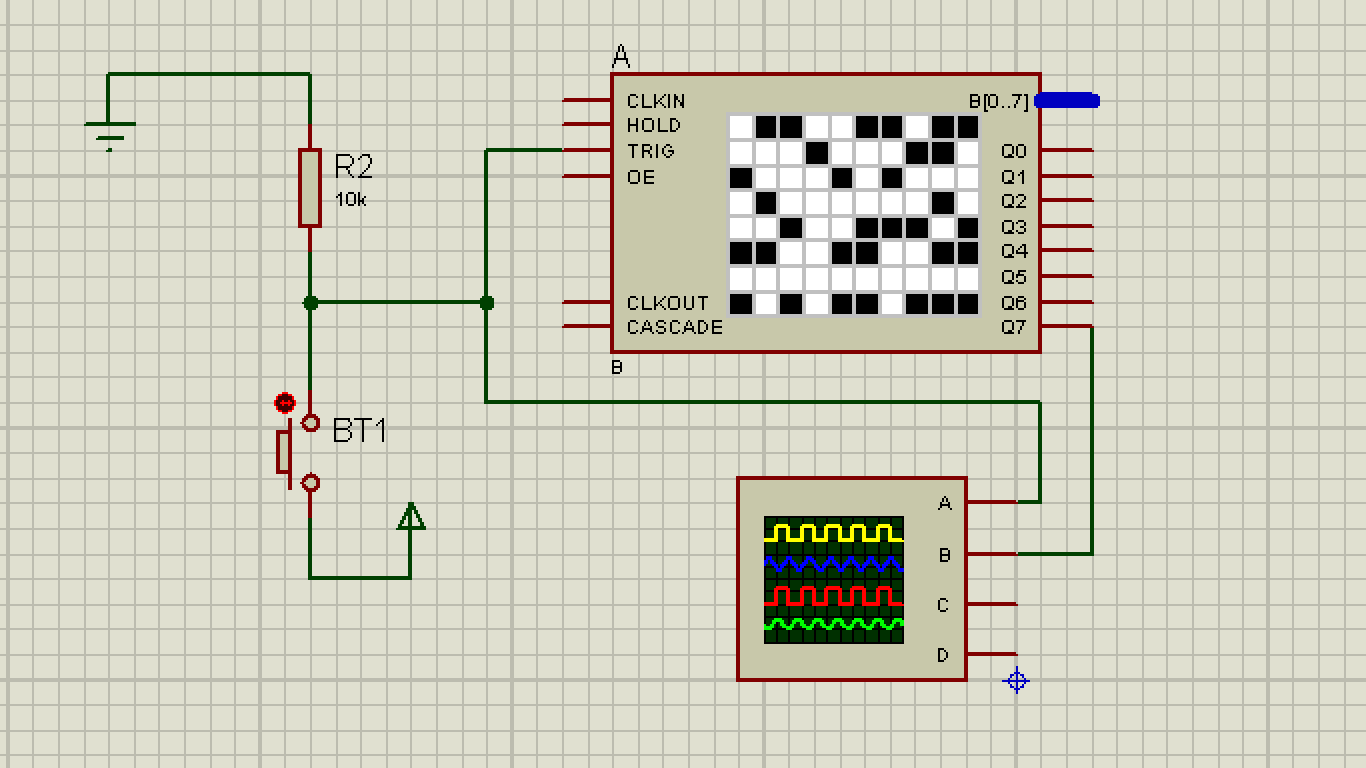
Solution:

Convert 5ms to uS :

Range is 10uS – 5000uS

Scale: 5000 - 10= 4999 uS

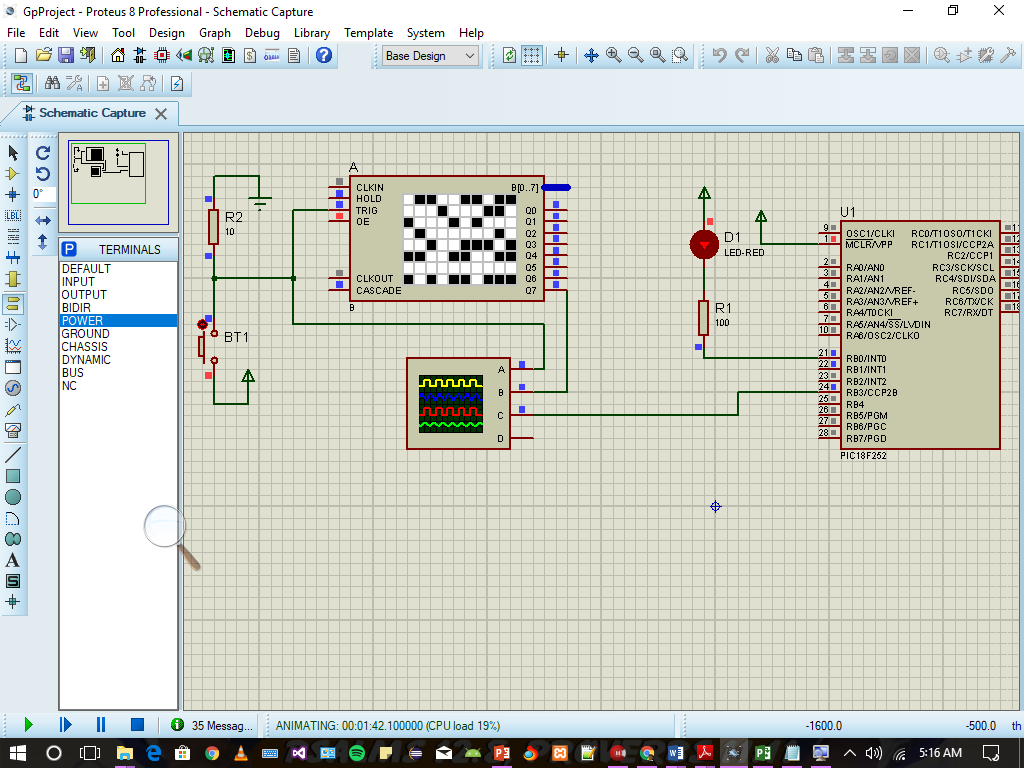
* 4999/10 = 499 separation between each distributed value



|  |  |  |
| --- | --- | --- |
| **Clock Time (of Pattern Generator) uS** | **ECHO Pulse Time (Tp)** | **Distance (Tp X 0.340)** |
| 10 | 100 | 34 |
| 598.1 | 6 | 2.04 |
| 1098 (1096) | 10 | 3.4 |
| 1597 | 16 | 5.44 |
| 2096 | 21 | 7.14 |
| 2595 | 26 | 8.84 |
| 3094 | 31 | 10.54 |
| 3593 | 36 | 12.24 |
| 4092 | 41 | 13.94 |
| 4591 <=5ms | 46 | 15.64 |

## Task C

Add on your test circuit a PIC18F252 (uC) and a Blinkalive led. On this second test circuit write a program code with 0.1 ms timebase that can implement blinkalive function, and build the simulation of the circuit to test it in ISIS environment.



Timer Setting Calculation:

For blinking 1 time every 10s

Count= 10/0.0001 = 100 000 CC

🡪 = 1.53 < PS

PS=2

Nc= Ncc/PS 🡪 100 000/2 = 50 000

To count 50 000 Prescaled clock cycles we set TMR0H = - Nc/256 & TMR0L = - Nc%256

Code:

void BlinkAlive(void)

{

char bcount;

if(bcount==0) PORTB.0=0; //LED ON

else PORTB.0=1; //LED OFF

bcount++;

if(bcount>9) bcount=0;

}

....

BlinkAlive();

Do{ T0CON = 0b100000000; TMR0H = - 50 000/256; TMR0L = - 50 000%256;

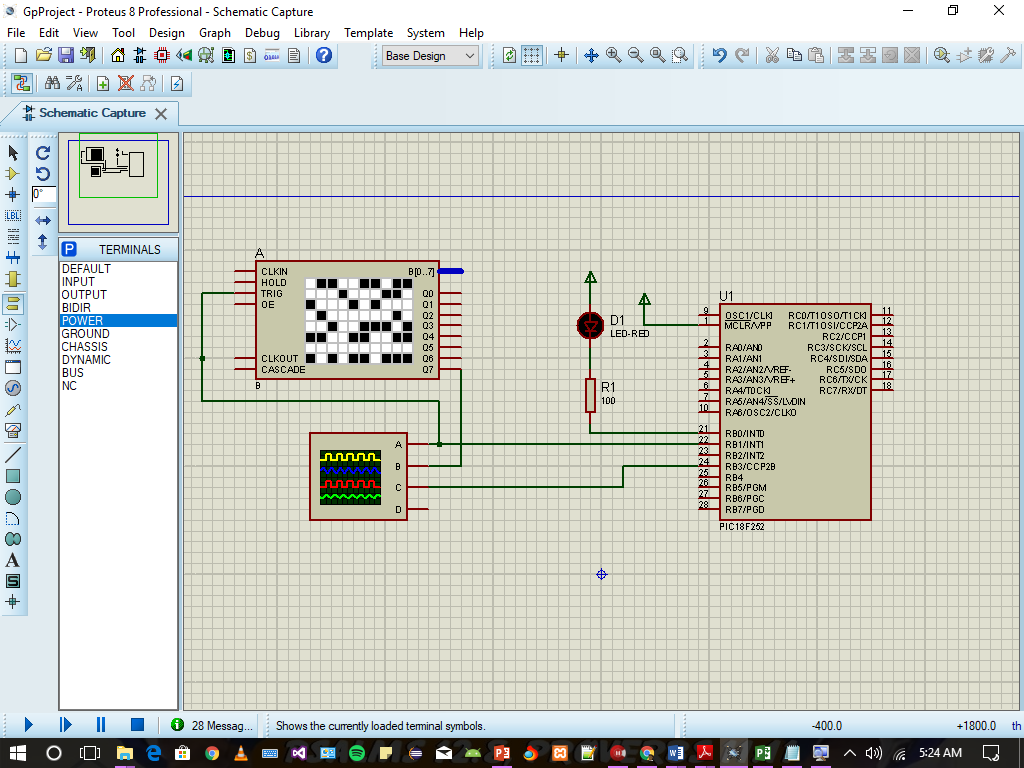
.......

}while(!TMR0IF);

## Task D

On your next test circuit remove the TRIG button, and connect TRIG to your uC. Modify the main program by adding a counter to count 0.1s periods for 10 seconds. Add a code to generate an approximately 20us trig pulse once at every 10 seconds. Use lst file to determine the period of TRIG pulse. Determine the exact period of the TRIG pulse by oscilloscope.

TRIG connected to Micro-Controller Schema



Code Tasks:

Adds counter to count 0.1s periods for 10 seconds.

Generates an 20Us TRIG pulse once at every 10 seconds

#define ECHOPin PORTB.2

#define ECHOTrs TRISB.2

#define TXPin PORTB.3

#define TXTrs TRISB.3

#define NC 50000

uns16 t0,t1,tp;

char T100m;

void BlinkAlive(void)

{ char bcount; if (bcount==0) PORTB.0=0; //LED ON

else PORTB.0=1; //LED OFF

bcount++;if (bcount>9) bcount=0; }

void ECHO\_WAIT\_UP()

{ECHOTrs=1; // Input for ECHO at RB2

do{}while(!ECHOPin);// portb.2=0}

void ECHO\_WAIT\_DN()

{ ECHOTrs=1; // Input for ECHO at RB2

do{}while(ECHOPin); //portb.2=1 }

void main(void)

{//Configuration

TRISB.0=0; //BLINKALIVE OUTPUT AT RB0

TRISB.1=0; TRISB.3=0;

TRISB.1=0; // OUTPUT TRIG (PULSE IS HIGH)

do{

T0CON=0b10000000; // set TMR0 for 0.1 sec

TMR0H=-NC/256; TMR0L=-NC%256;TMR0IF=0;

BlinkAlive();

++T100m; //increment by one to count 10s time period.

if(T100m>100)//true, then 10s is over

{

T100m=0; //clear to start count again

//TRIG passed 20us pulse

PORTB.1=1; //Output high for Trig

char i=20/3;

do{}while(--i);

PORTB.1=0;

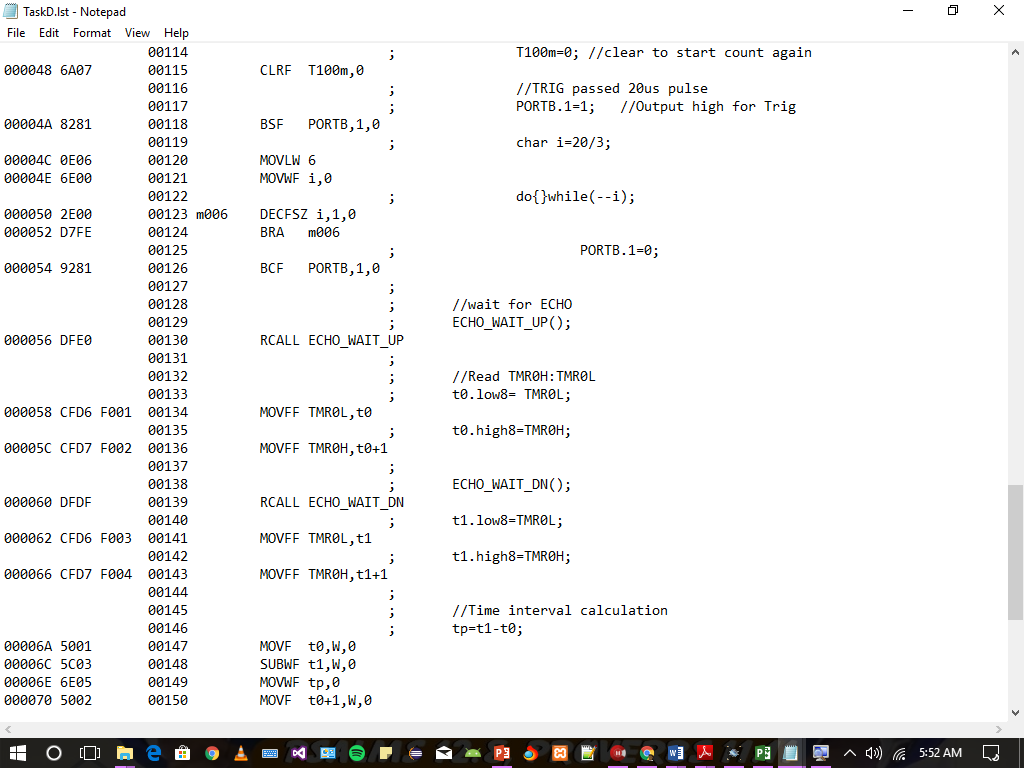
ECHO\_WAIT\_UP(); //wait for ECHO

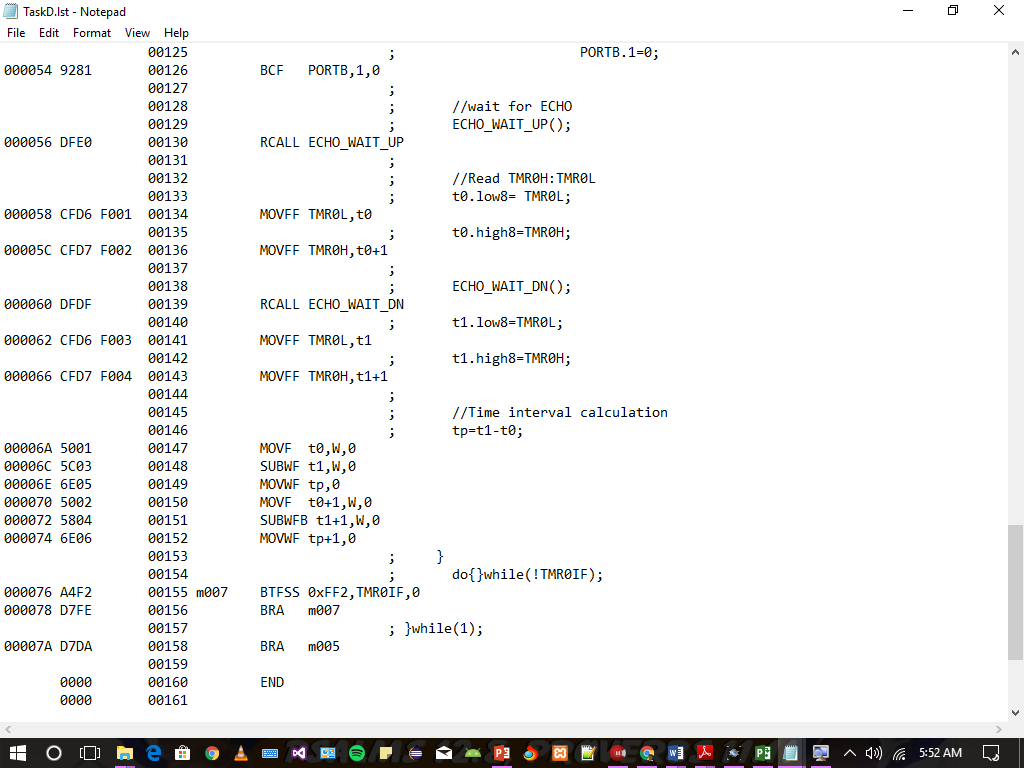
//Read TMR0H:TMR0L t0.low8= TMR0L; t0.high8=TMR0H;

ECHO\_WAIT\_DN(); t1.low8=TMR0L; t1.high8=TMR0H;

//Time interval calculation tp=t1-t0;} do{}while(!TMR0IF); }while(1); }

**Read 20uS Pulse from Lst file:**

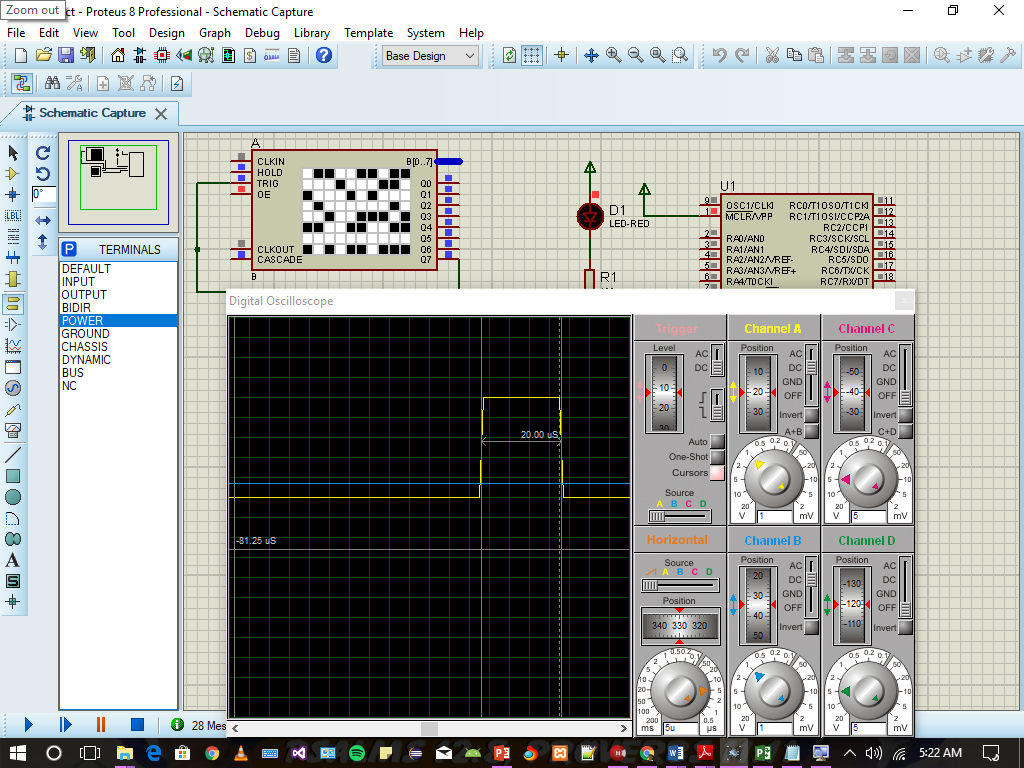




**Pulse count starts from BSF PORTB,1,0 till BRA m007 with every instruction being 1CC and every “BRA” instruction is 2 CC.**

**Total Clock Cycles =20 CC**

**Read The Exact Period (20Us) of TRIG PULSE from Oscilloscope :**



## Task E

Write necessary code to read the timer at the start and at the end of the ECHO pulse. Develop a solution to detect failed sensors when ECHO takes longer than 40 ms

void ECHO\_WAIT\_UP()

{ECHOTrs=1; // Input for ECHO at RB2

do{}while(!ECHOPin);// portb.2=0}

void ECHO\_WAIT\_DN()

{ECHOTrs=1; // Input for ECHO at RB2

do{}while(ECHOPin); //portb.2=1}

....

....

....

//wait for ECHO

ECHO\_WAIT\_UP();

//Read TMR0H:TMR0L

t0.low8= TMR0L;

t0.high8=TMR0H;

ECHO\_WAIT\_DN();

t1.low8=TMR0L;

t1.high8=TMR0H;

//Time interval calculation

tp=t1-t0;

//TO DETECT FAILED SENSOR Check if Tp is greater than 40ms if its true BLINKALIVE SHOULD STOP WORKING

if(tp>40)

TRISB.0=1; //TURN OFF BLINKALIVE

## Task G

Using #include <math24F.h>

//wait for ECHO

ECHO\_WAIT\_UP();

//Read TMR0H:TMR0L

t0.low8= TMR0L;

t0.high8=TMR0H;

ECHO\_WAIT\_DN();

t1.low8=TMR0L;

t1.high8=TMR0H;

//Time interval calculation

tp=t1-t0;

//timer calculation

TXhex(t0.high8);TXhex(t0.low8);TXchar('/');

TXhex(t1.high8);TXhex(t1.low8);

TXchar('/');

dmm=tp\*0.34;

WREG=84;multiply(tp.high8);

dmm.low8=PRODL;dmm.high8=PRODH;

multiply(tp.low8);

dmm.low8+=PRODH;

if(Carry)++dmm.high8;

## Task H

char a5,a4,a3,a2,a1;

void u16toASCII(char \*ap , uns16 dmm) {

uns16 i=dmm;

\*ap='0'; while(i>9999){++\*ap; i-=10000;} ++ap;

\*ap='0'; while(i>999){++\*ap; i-=1000;} ++ap;

\*ap='0'; while(i>99){++\*ap; i-=100;} ++ap;

\*ap='0'; while(i>9){++\*ap; i-=10;} ++ap; \*ap='0'+i;

++ap; \*ap=0 ;

}

void Int16AsciiConvert(uns16 n)

{

uns16 i=n;

a5='0'; while(i>9999){++a5; i-=10000;}

a4='0'; while(i>999){++a4; i-=1000;}

a3='0'; while(i>99){++a3; i-=100;}

a2='0'; while(i>9){++a2; i-=10;}

a1='0'+ i;

}

void TXASCII()

{

TXchar(a5);TXchar(a4);TXchar(a3);

TXchar(a2);TXchar(a1); TXchar(0x0D);

}

void main(void)

{

//Configuration

TRISB.0=0; //OUTPUT AT RB0

TRISB.1=0;

TRISB.3=0;

TRISB.1=0; // OUTPUT TRIG

TRISB.4=1;

do{

T0CON=0b10000000; // set TMR0 for 0.1 sec

TMR0H=-NC/256; TMR0L=-NC%256;TMR0IF=0;

BlinkAlive();

++T100m;

//Button Tasks

if(T100m>100)

{

T100m=0;

//TRIG passed 20us pulse

PORTB.1=1; //Output high for Trig

char i=20/3;

do{}while(--i);

PORTB.1=0;

//wait for ECHO

ECHO\_WAIT\_UP();

//Read TMR0H:TMR0L

t0.low8= TMR0L;

t0.high8=TMR0H;

ECHO\_WAIT\_DN();

t1.low8=TMR0L;

t1.high8=TMR0H;

//Time interval calculation

tp=t1-t0;

//timer calculation

TXhex(t0.high8);TXhex(t0.low8);TXchar('/');

TXhex(t1.high8);TXhex(t1.low8);

TXchar('/');

dmm=tp\*0.34;

WREG=84;multiply(tp.high8);

dmm.low8=PRODL;dmm.high8=PRODH;

multiply(tp.low8);

dmm.low8+=PRODH;

if(Carry)++dmm.high8;

arrypoint=&arry[0];

u16toASCII(arry,dmm);

Int16AsciiConvert(dmm);

\*arrypoint=13;

++arrypoint;

\*arrypoint=0;

arrypoint=&arry[0];

while(\*arrypoint){TXchar(\*arrypoint); ++arrypoint;};

TXASCII();

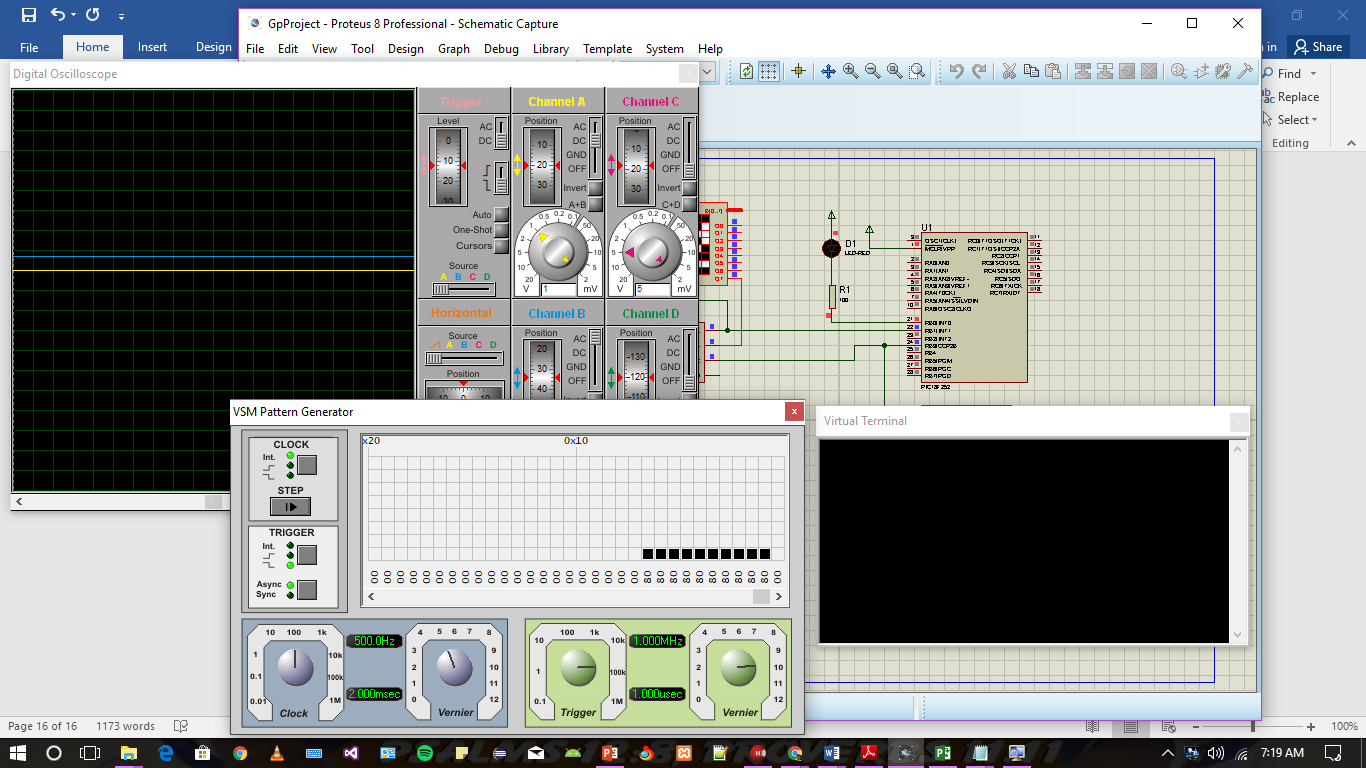
}

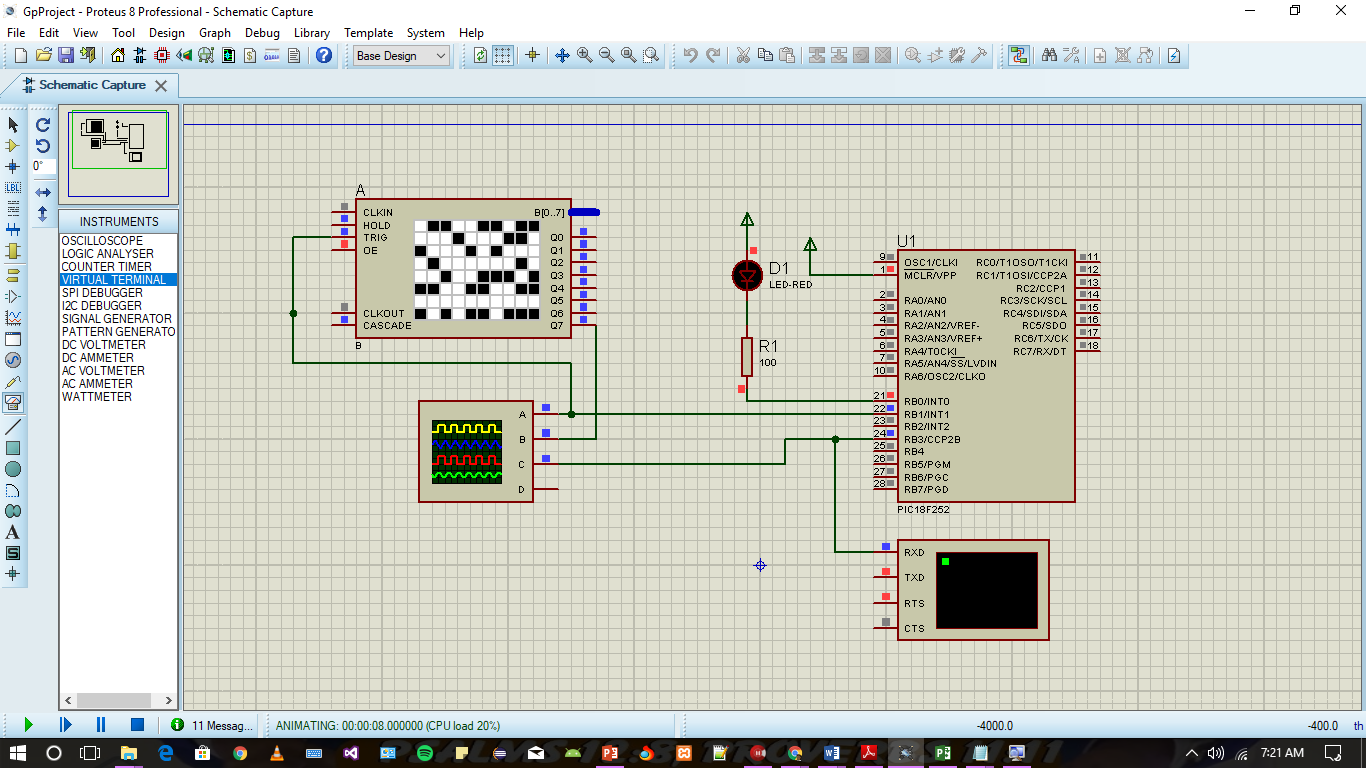
do{}while(!TMR0IF);

}while(1);

}

## Task I



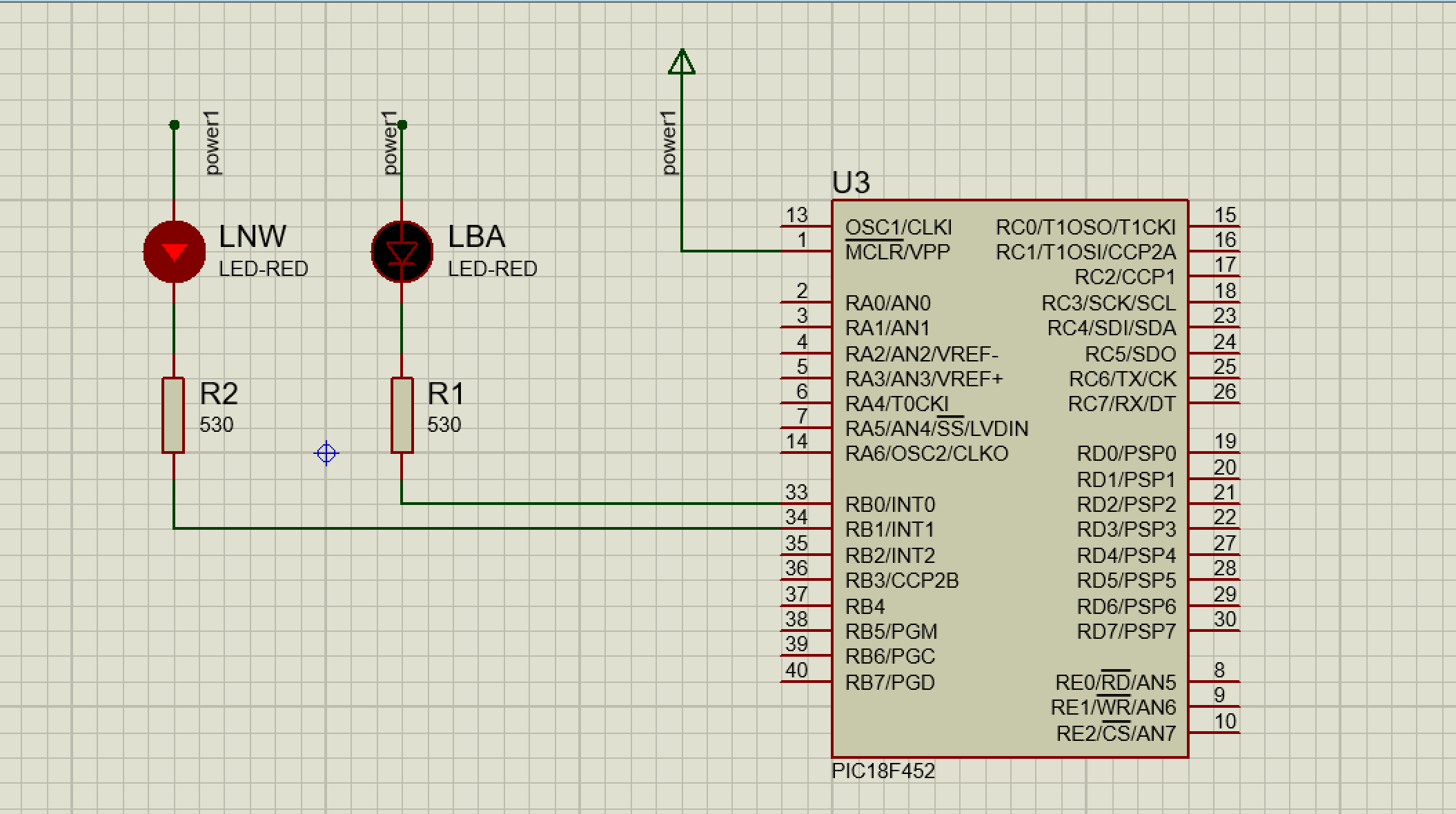


# The main unit (MAU)

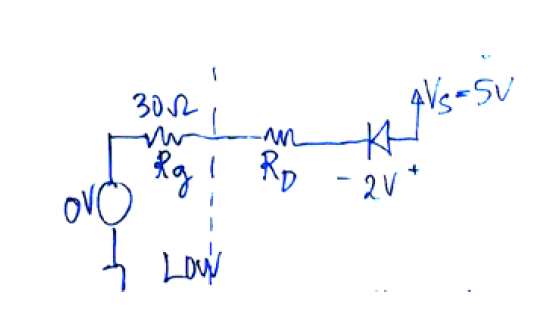
## 1-making a blink alive circuit:

The system require an blink alive functionality, so the system has time base with 0.1 seconds and the blink alive led LBA should blink for 0.1 seconds and be dark for 0.9 seconds. The LBA is connected to PORTB.0 and LNW is connected to PORTB.1. For the timing we used the TMR0 unit for this purpose for its accuracy and independency.

The calculation for timer, PIC18F452 run at 4MHz with T = 1us per clock cycle. So, making 0.1s (100 000 us) require to count (0.1s)/(1us) => 100 000 clock cycles. TMR0 run with 16bit at most, so we will run the pre-scaler with ceil(100 000/2^16) = 2. So the TMR0CON = 0b1.0.0.0.0.000; and the required new number of clock cycles needed after pre-scaler will be 100 000/2 = 50 000. And we will assign the 50 000 into TMR0 by setting the low first and the high second of TMR0.



Connection for LEDs into the circuit, we connected the anode of the LED to +5V and connect the cathode to the PIC via pull down resistance. The LEDS have an 2.2 forward voltage. And the PIC resistance calculated as follows: PIC Th resistance = 30 ohm.



so to calculate the Rd we used this rule : *Id*=(*Vs* – *Vd*)/(*Rg*+*Rd*); => *Rd* = (*Vs* – *Vd*)/*Id* – *Rg* which result for Id = 5 mA, Rd = 2.8V/5mA – 30 ohm = **530ohm**.

The code for this Task :-

#pragma chip PIC18F452

#define on 0

#define off 1

#define NC 50000

#define LBA PORTB.0

void delay100ms()

{

T0CON=0b10000000; // set TMR0 for 0.1 sec

TMR0H=-NC/256; TMR0L=-NC%256;TMR0IF=0;

do{}while(!TMR0IF);

}

void blink(void) {

static char BAcount;

if(BAcount == 0) LBA = on;

else LBA = off;

++BAcount;

if(BAcount > 9) BAcount = 0;

}

void initPorts(){ TRISB = 0; }

void main(void)

{

initPorts();

do {

blink();

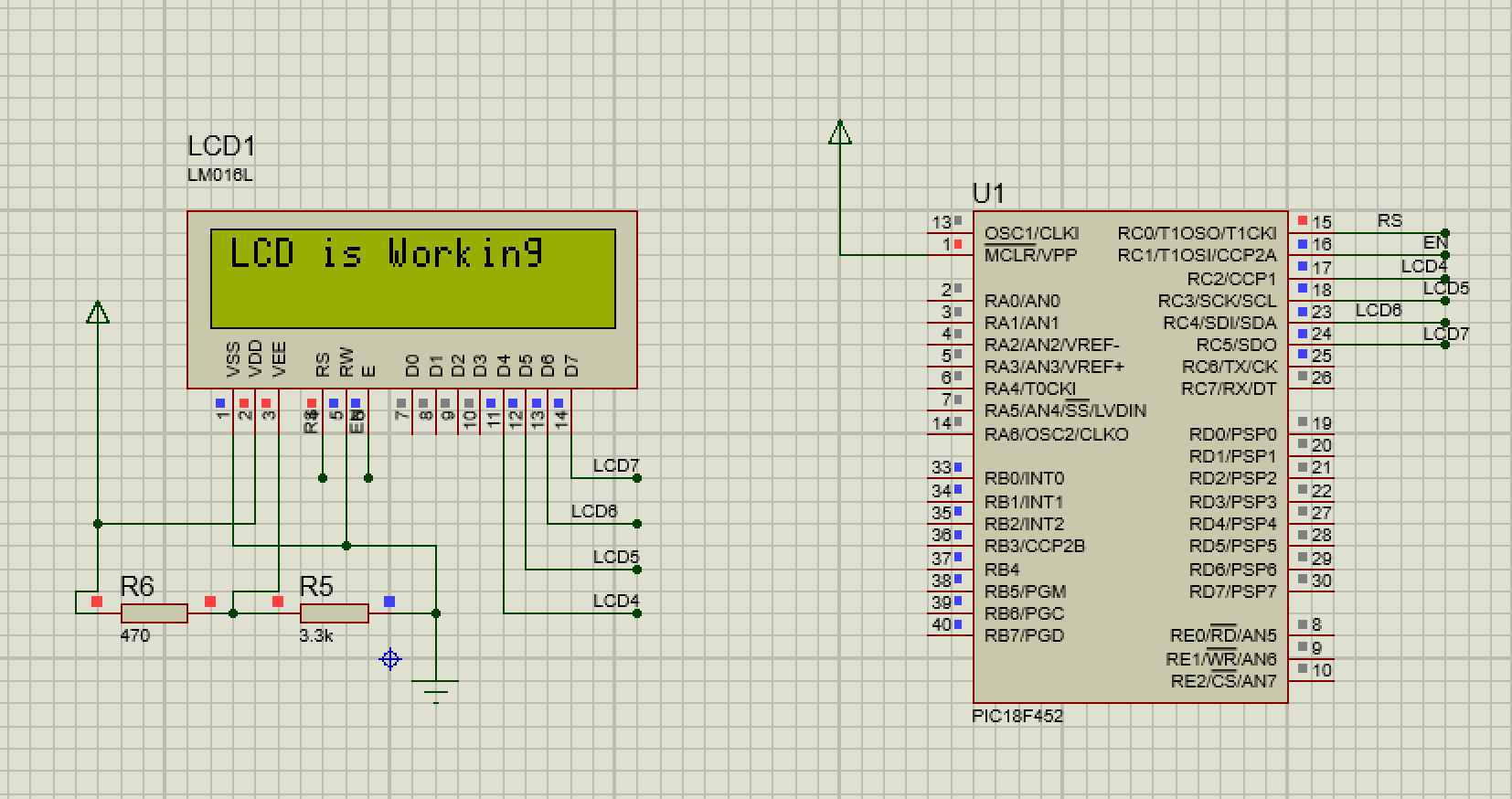
delay100ms();

} while(1);

}

## 2-Working with LCD unit

The LCD is connected on Nibble mode (4 bit data lines). It connected via PORTC. LCD D4 to PORTC.2, LCD D5 to PORTC.3, LCD D6 to PORTC.4 and LCD D7 to PORTC.5. Also LCD RS (LCD reset control) to PORTC.0 and LCD EN (LCD enable control) to PORTC. The LCD needs 100ms to reset. So using the previous task for 100ms delay for reset will make it work. TMR0 will run in 16 bit mode with pres-scaler 2 with NC = 50 000 which means 100ms.



Using the LCD ready functions like Print LCD and print char, the LCD shall be able to run properly. Like these functions:

void LCDNibble(char Ch)

{

LCD4T=0; LCD5T=0; LCD6T=0; LCD7T=0;

LCDRT=0; LCDET=0; LCDE = 0;

if(LCDC) LCDR=0; else LCDR = 1;

if(Ch.4) LCD4=1; else LCD4=0; if(Ch.5) LCD5=1; else LCD5=0;

if(Ch.6) LCD6=1; else LCD6=0; if(Ch.7) LCD7=1; else LCD7=0;

LCDE=1; LCDw2u(2); LCDE=0;

LCD4=0;nop();LCD5=0;nop();LCD6=0;nop();LCD7=0; LCDw2u(5);

}

void PrintLCD(const char \*Ch)

{

char WC,WP=0; LCDS=0 ; LCDC=0 ; LCDE=0 ;

do{ WC=Ch[WP]; WP++;

if(WC){ LCDC=0;

if(WC==0xFF) LCDS ^=1 ;

else { LCDC=WC.7||LCDS; if(WC<4) LCDC=1;

LCDNibble( WC & 0xF0);

if(LCDS && WC==0x28){//set mode takes 3ms time

char T=12;do{LCDw2u(0);}while(--T);}

LCDNibble( swap(WC) & 0xF0 );

if(WC<4){char T=12; do{ LCDw2u(0);}while(--T);}

LCDw2u(20); } }

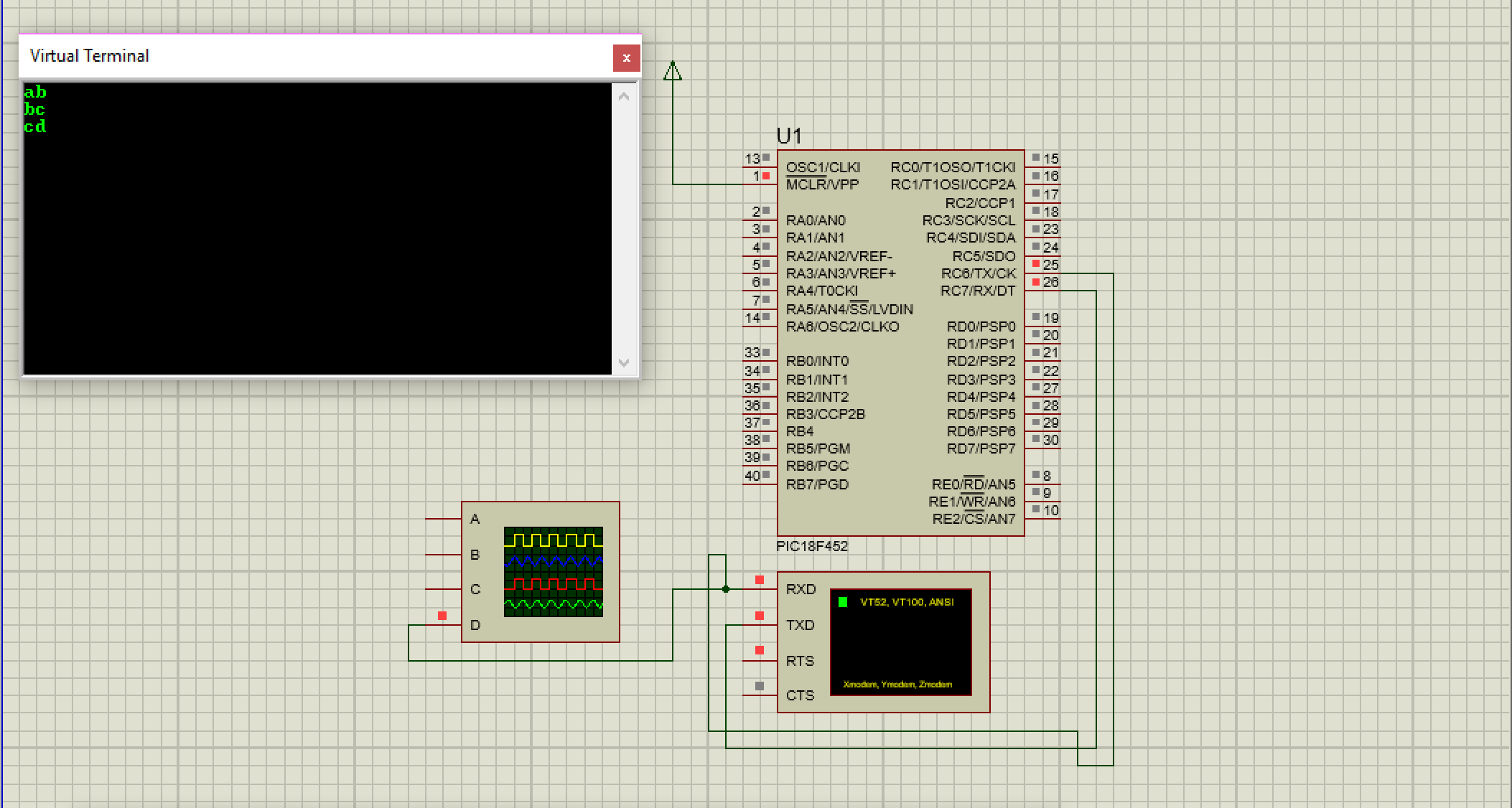
}while(WC);

}

LCD is tested by trying to print on it “LCD is working” sentence. And after running the simulation it printed it out. The LCD require a power of 5V on Vdd pin also 0V on Vss pin.

## 3-UART Unit for receive and transmit ASKI characters.

In the UART unit we used PORTC.6 for transmitting and PORTC.7 for receiving. In this task we test this unit via virtual terminal, so we can transmit to the unit and receive from unit. The Virtual terminal sited to 9600 baud rate and also other settings for simulation as follows: “{PRIMITIVE=DIGITAL} {MODDLL=VTERM.DLL} {BAUDRATE=9600} {DATABITS=8} {PARITY=NONE} {STOPBITS=1} {XONXOFF=0} {SIGPOL=0} {CTLPOL=0} {TRACE=1}”.



Configuring URAT require to calculate which SPBRG will be used. Since it is 9600 baud rate, so if we set BRGH we will get SPBRG = (4MHz/16/9600) – 1 => 25.04 => 25 because it needs to be integer from 0 to 255. The real Baud rate with BRGH = 1 is (4MHz/(25+1)/16) = 9615.4 Baud. For BRGH = 0, SPBRG = (4MHz/64/9600) – 1 => 5.5 => 6. And real Baud is (4MHz/(6+1)/64) = 8928 Baud.

For this comparison we calculated the Errors of each of them, for BRGH = 1 the error = (|9615.4 – 9600|)/9600 = 0.0016 = 0.16% error. For BRGH = 0 the error = (|8929-9600|)/9600 = 0.07 = 7% which is not acceptable because it is more than 4%. So we choose setting with BRGH = 1.

The code for this section:

#pragma chip PIC18F452

void InitUART(){

TRISC.6=0;TRISC.7=1;

BRGH=1; SPBRG=25; TXEN=1;CREN=1;SPEN=1;

}

void main(void){

InitUART();

char tmp;

do{

if(RCIF){

tmp = RCREG;

{

do{}while(!TXIF);

++tmp;TXREG=tmp;

}

}

{int16 i=14285; do{}while(--i);} // 0.1 sec

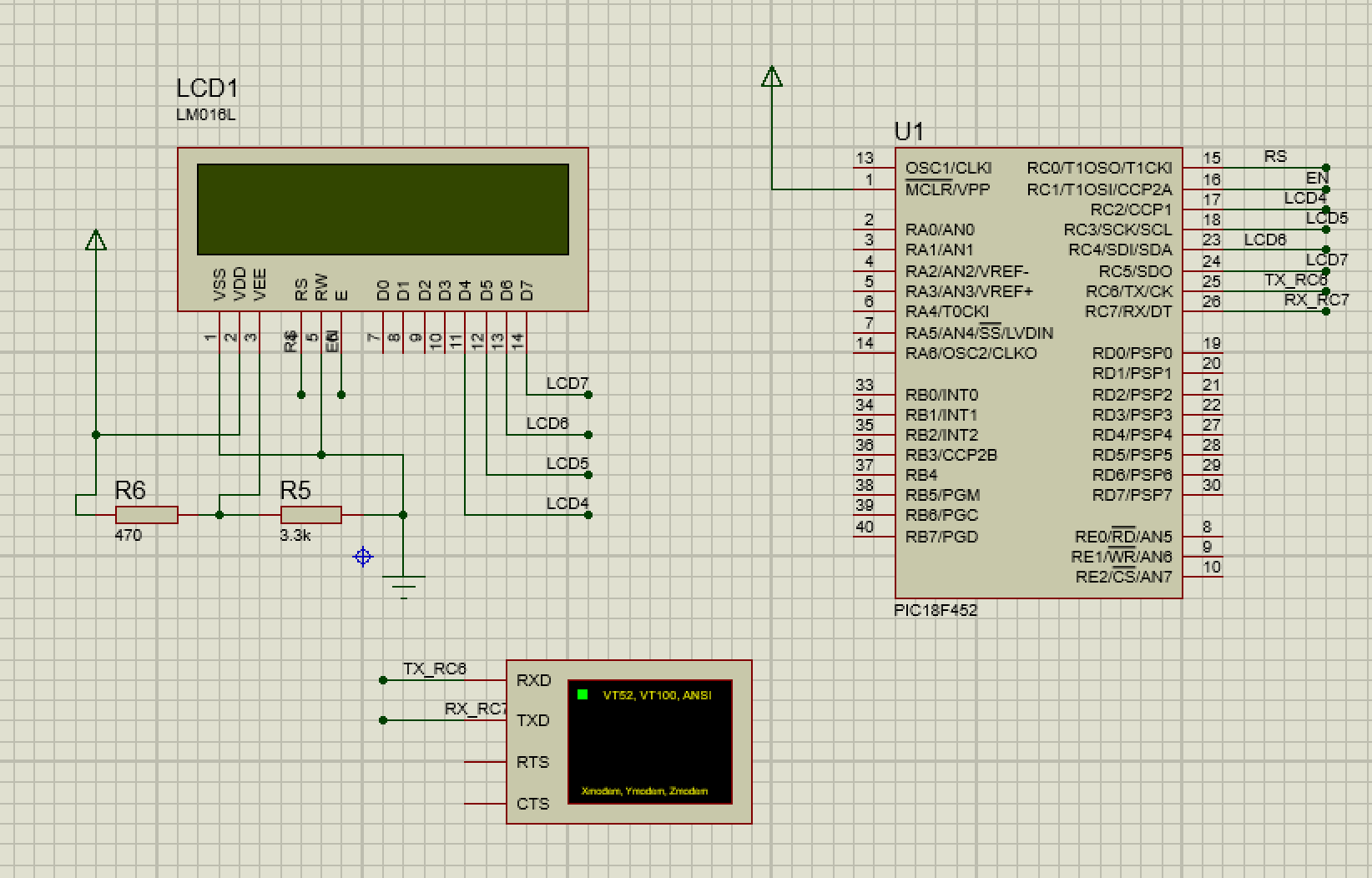
}while(1);

}

The test is passed for this circuit and it shown when setting the virtual terminal to echo typed chars, it received the next chars immediately. The test cared by type ‘a’ and receive ‘b’ and second test via typing ‘b’ and receive ‘c’ and last test were transmit by typing ‘c’ and got ‘d’. So all tests is passed successfully.

## 4-Connecting LCD and print received string from ISU

In this part, the LCD is used to test receiving string via UART unit from ISU and display it. Also the LCD test the result of conversion this string to integer and convert it back to string so it can be displayed on LCD. The circuit is combination of previous two circuits.



The virtual terminal is used as simulation for ISU unit to send a string. And the end of string char used for testing is ‘a’ char because virtual terminal don’t accept to enter 0x0D. this part required to define 3 new functions, recieveString(char \*a), str2i(char \*a) and i2a(char \*).

First one (resiceveString) is used to receive a string from ISU unit (virtual terminal in this case). The cod as follow:

void recieveString(char \*a)

{

if(!RCIF || string\_ready) return ;

char tmp = RCREG;

//0x0D

if(tmp == 'a'){

\*btr = '\0'; btr = a;

string\_ready = 1;

return;

}

\*btr = tmp;

++btr;

}

second one is for converting the input string to an integer so it can be manipulated, so str2i() play this role:

uns16 str2i(char \*a) {

char i=0,tt;

uns16 res=0;

//tt=a[i]-'0';res += tt\*mult;

while(i<5){res\*=10;res +=a[i]-'0';i++;}

return res;

//2,0,0,0,0 original in ISU

//0,0,0,0,2 recieved

}

last function is to convert any integer to a string that can be printed on LCD. And a2i() function was responsible for this :

void i2a(uns16 k, char \*a){

char i = 5;

while(k){ --i;a[i]=k%10; a[i]+='0'; k/=10;}

while(i){--i;a[i]='0';}

}

for testing this values is entered to virtual terminals and the LCD displayed these strings.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEST | Input/V-Terminal | Output1/LCD | Output2/LCD | result |
| 1 | “00002a” | 00002 | 00002 | passed |
| 2 | “12345a” | 12345 | 12345 | passed |

## 5-Reading analog input from potentiometer

setting ADC require to set the ADCON1 which require to chose PCFG to 1110 because we need only A0 for analog input. Also making TRISA.0 = 1 for input of POT meter. ADC will work on 8bit accuracy so the ADCON1 = 0b0.0.00.1110, and ADCON1 for internal RC oscillator and specifying A0 is 0b11.000.0.0.1. the function that set the configuration is as follow:

void initADC() {

ADCON0 = 0b11.000.0.0.1;

ADCON1 = 0b0.0.00.1110;

}

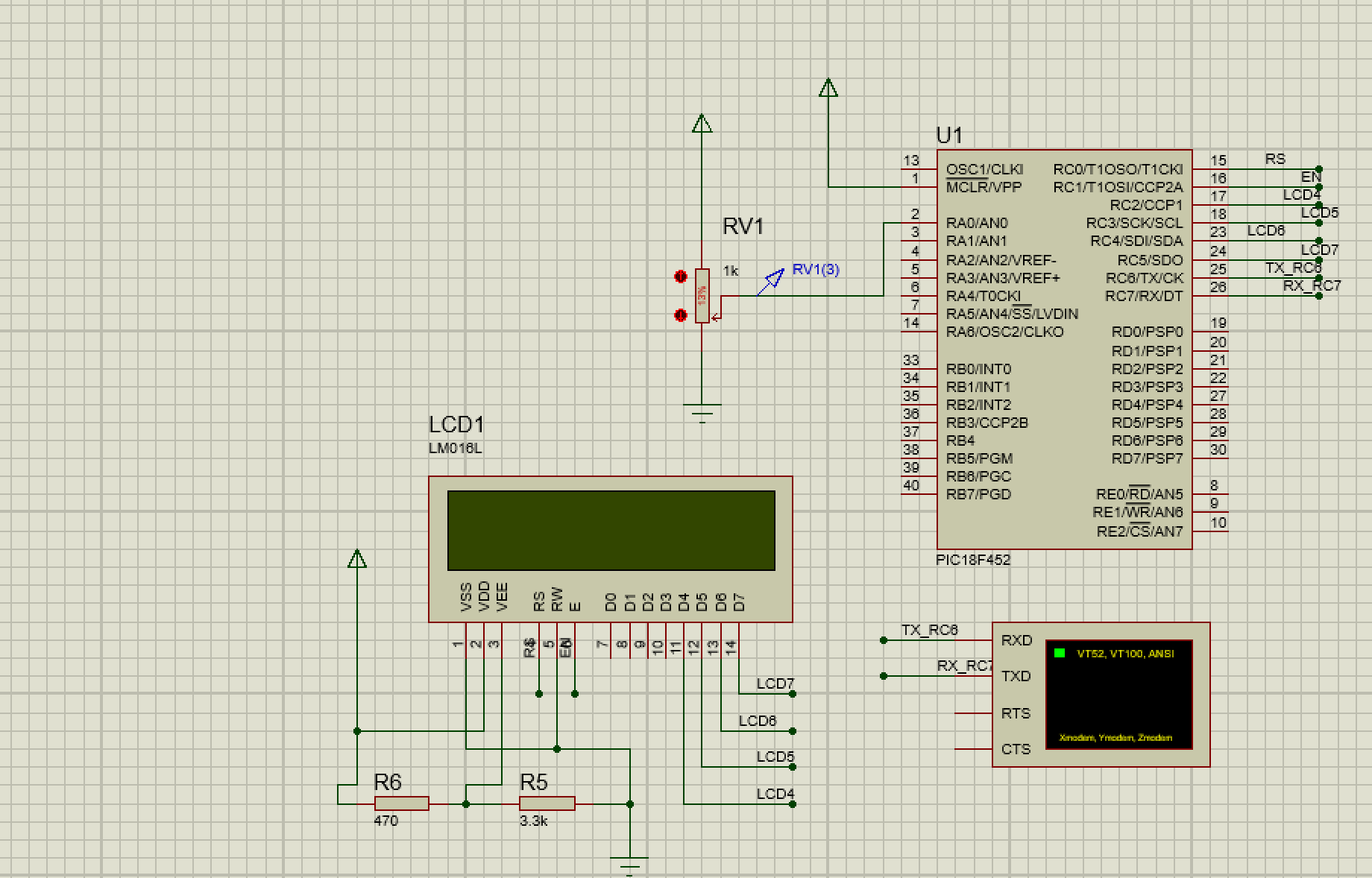
Also to start the conversion there is function that change ADCON0.2 to 1 and wait until it become 0:

void startADC(){

ADC\_go\_done=1; do{}while(ADC\_go\_done); // wait ADC to be over

}

so the circuit for ADC POT and for LCD after connection will look like:



To read the input result from ADC we use ADRESH register and after multiply by 100 and divide by 255 (because this is the maximum for 8bit accuracy) the result will be the percent. To show the ‘%’ symbol on LCD we add before end of string this symbol.

The test taken with 10 different POT values and the result in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| TEST | POT settings | LCD output | result |
| 1 | 50% | 50% | passed |
| 2 | 51% | 50% | Passed (small error) |
| 3 | 52% | 52% | passed |
| 4 | 29% | 29% | passed |
| 5 | 13% | 12% | Passed (small error) |
| 6 | 80% | 80% | passed |
| 7 | 75% | 75% | passed |
| 8 | 7% | 6% | Passed (small error) |
| 9 | 30% | 30% | passed |
| 10 | 99% | 98% | Passed (small error) |

## 6-Reading water level from ISU unit

For reading water level , we connected the UART unit task with previous one that has a POT meter. The system every 2 second convert the water level integer to percent by subtracting h (1000mm) from it and divide by h then multibly by 100 to get the percent out of 100.

iwp = h - iwl;

htmp = h / 100;iwp/=htmp;

as what shown instead of multiplying with 100 we divide the h by 100 to save the important digits and to not overflow the uns16. Also for POT meter we did the same thing that we did in previous task :

ipot = ADRESH;

ipot\*=100;ipot/=255;

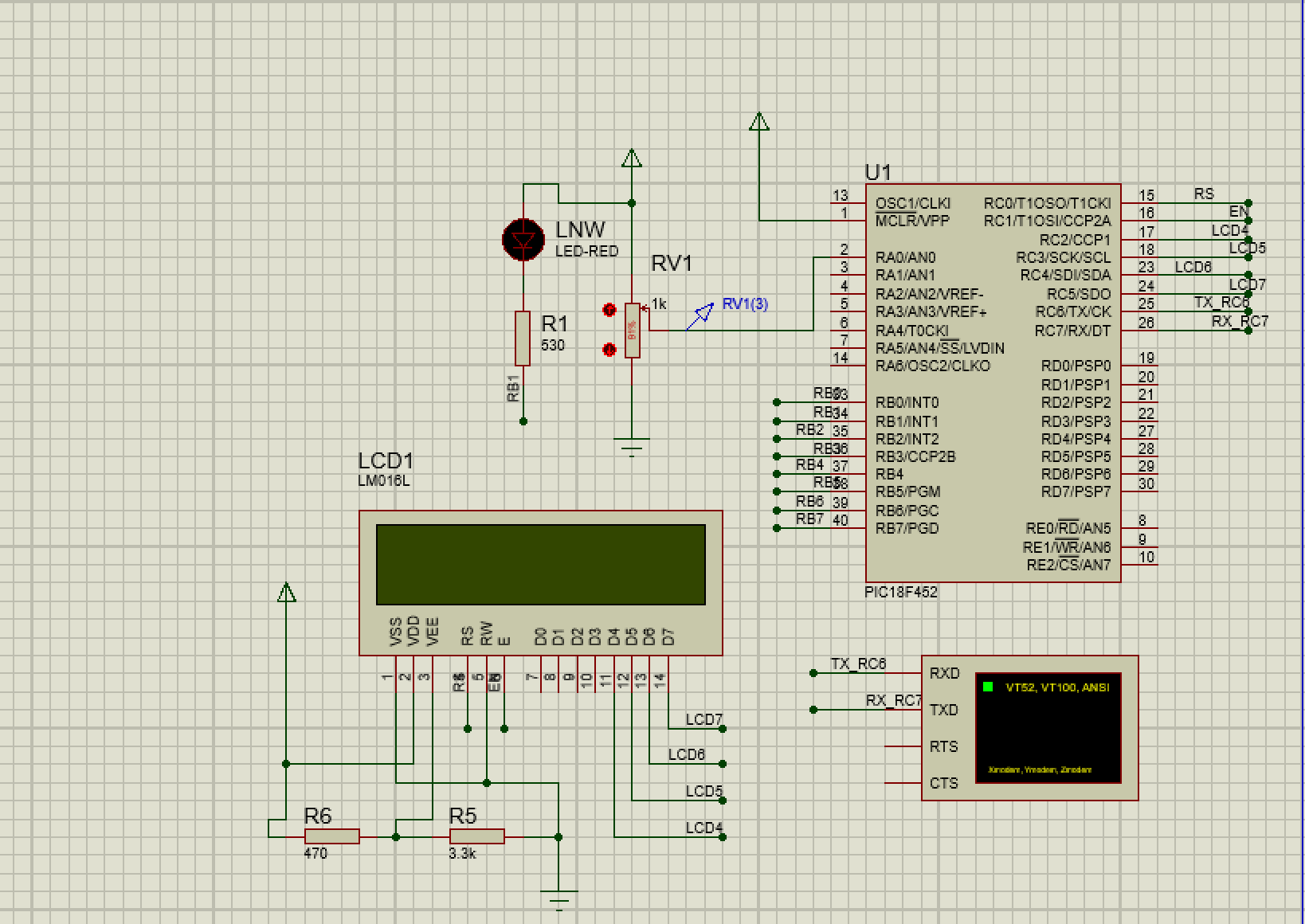
we didn’t do the same because at maximum it will be 25500 which can fit in uns16.

The test is performed against this circuit for these values:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TEST | POT settings | V-Terminal | LCD Water | LCD POT | result |
| 1 | 50% | “00100a” | 50% | 90% | passed |
| 2 | 51% | “00200a” | 50% | 80% | passed |
| 3 | 52% | “00300a” | 52% | 70% | passed |
| 4 | 29% | “00400a” | 29% | 60% | passed |
| 5 | 13% | “00500a” | 12% | 50% | passed |
| 6 | 80% | “00600a” | 80% | 40% | passed |
| 7 | 75% | “00700a” | 75% | 30% | passed |
| 8 | 7% | “00800a” | 6% | 20% | passed |
| 9 | 30% | “00900a” | 30% | 10% | passed |
| 10 | 99% | “00000a” | 98% | 100% | passed |

## 7-Turn on no water LED

Here we use the previous result to make an action depend on the POT percent and on Water percent by comparing between them. So we turn NWL on if W%<POT% otherwise Turn the LED OF. We have to configure TRISB.1 for output by making it 0. The result circuit is :



the code that control the LNW is :

if(iwp<ipot)

LNW=on;

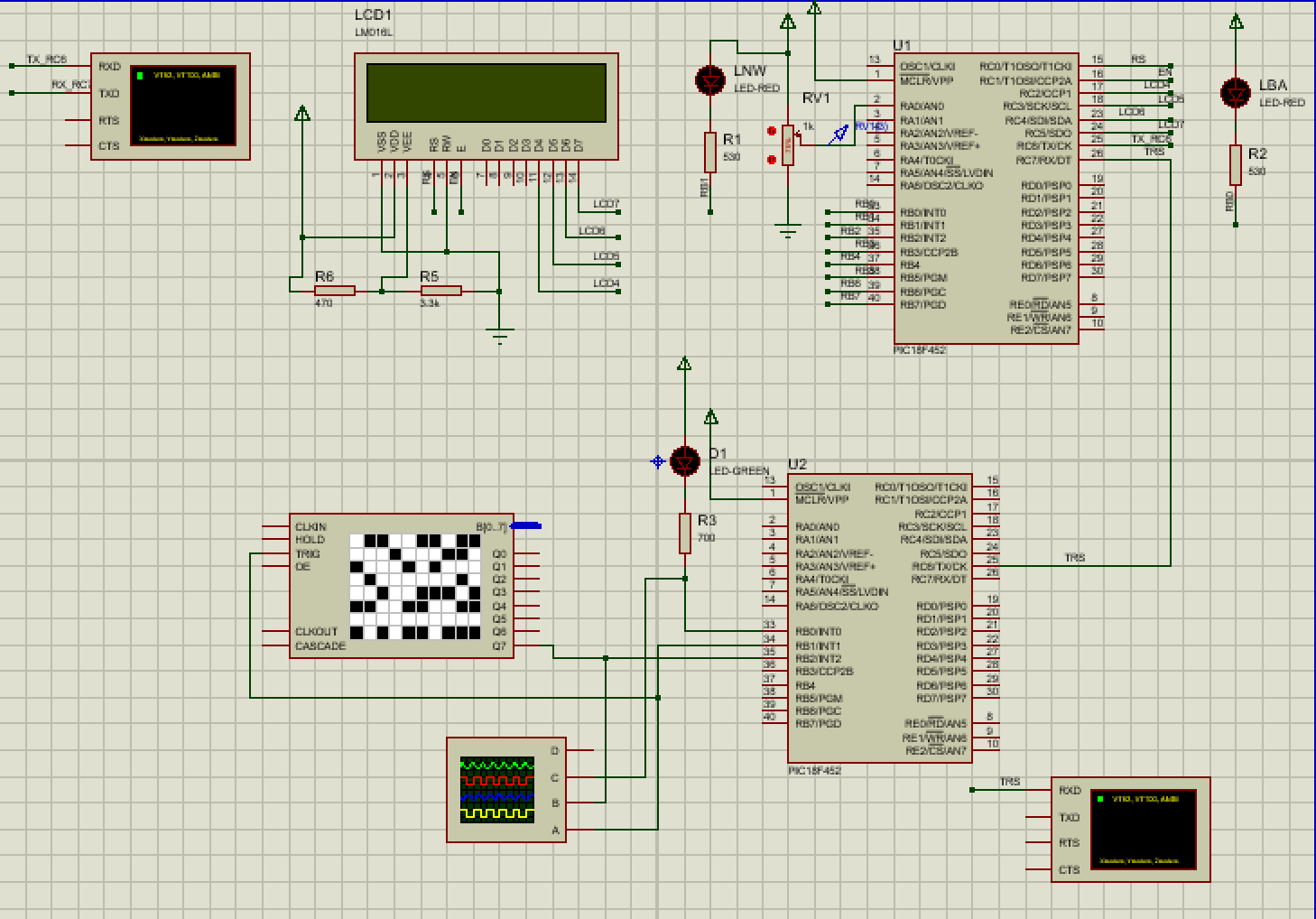
else

LNW=off;

LNW is defined as #define LNW PORTB.1

# Over all System

## Circuit:



## Code of ISU:

#pragma chip PIC18F452

#define NC 50000

char T100ms,blinkT100ms;

//#include "math24F.h"

// return from interrupt

// on lowgoing edge of BT4 give 0.1s pulse from RB0

// RB7 is for blinkalive LED

// TIMER0 generates 100 ms interrupts.

// Fosc=4MHz, 0.1s=100000cc., PS=2, Nc= 100000/2

#define BALEDPin PORTB.0

#define BALEDTrs TRISB.0

#define TrigPin PORTB.1

#define TrigTrs TRISB.1

#define ECHOPin PORTB.2

#define ECHOTrs TRISB.2

void InitPorts(void){

BALEDTrs = 0;//PORTB.0 = 0;

TrigTrs = 0;//PORTB.1 = 0;

ECHOTrs = 1;//PORTB.2 = 1;

TRISC.6=0;

TRISC.7=1;

} // RC7..RC0 output

void InitUART(){

TRISC.6=0;TRISC.7=1;

BRGH=1; SPBRG=25; TXEN=1;CREN=1;SPEN=1;

}

void Blink(void){

//BALEDPin can used instade of PORTB.0;

static char BAcount;

if(BAcount==0) BALEDPin=0; // LED on

else BALEDPin=1; // LED off

BAcount++; if(BAcount>9) BAcount=0;

}

void bauddelay(void){

char i=(104-8)/3 ;

do{}while(--i);

}

void transmit(uns16 dmm)

{

char a[6];

a[0]='0'; while(dmm>10000){++a[0]; dmm-=10000;}

a[1]='0'; while(dmm>1000){++a[1]; dmm-=1000;}

a[2]='0'; while(dmm>100){++a[2]; dmm-=100;}

a[3]='0'; while(dmm>10){++a[3]; dmm-=10;}

a[4]='0'+dmm;

a[5]=0x0D;

char i,y;

for(i=0; i<7; ++i){

do{}while(!TXIF); TXREG=a[i];

y=8;while(--y){}

}

}

void set100ms() {

// set Timer0 for 0.1sec

TMR0H=-NC/256; TMR0L=-NC%256; TMR0IF=0;

T0CON=0b1000.0000;

}

void main(void)

{

char PPB, PB, PBpushed;

T100ms=0;

InitPorts();

InitUART();

PB=PORTB;

do{

set100ms();

PPB=PB; PB=PORTB; PBpushed=~PB&PPB;

if (T100ms >= 50) {

T100ms = 0;

TrigPin = 1; // trig

char i = 20/3;

do{}while(--i);

TrigPin = 0;

do{}while(!ECHOPin);

uns16 t1,t2,td; t1.low8 = TMR0L;t1.high8 = TMR0H;

do{}while(ECHOPin && !TMR0IF );

t2.low8 = TMR0L;t2.high8 = TMR0H;

td = t2 - t1;

uns16 dmm;

// dmm=tp\*0.34;

//multiplyconstfrac(tp,84,dmm);

if(!TMR0IF){

WREG=td.low8; multiply(87);

dmm.low8=PRODH; WREG=td.high8;

multiply(87); dmm.low8 += PRODL;

if(Carry) ++PRODH; dmm.high8=PRODH;

}else{

dmm = 1000;

}

transmit(dmm);

}

do{}while(!TMR0IF);

++T100ms;

Blink();

}while(1); /\* now mainloop takes around 20 cc only \*/

}

## Code of MAU:

#pragma chip PIC18F452

#define CLR\_LCD; PrintLCD("\xff\x02\x82\x28\x01\x03\x0C\x06");

#define NC 50000

#define h 1000

#define on 0

#define off 1

#define LNW PORTB.1

#define LBA PORTB.0

char ustr[16];

char bstr[16];

char T100ms,Tbase;

char rstr[8];

char \*btr;

char io;

char string\_ready;

uns16 iwl;

void str2i();

#pragma origin 0x08

void ISR(){

static char WX,SX,BX;

WX=WREG;SX=STATUS;BX=BSR;

if(TMR0IF){

T0CON=0b10000.000; // set TMR0 for 0.1 sec

TMR0H=-NC/256; TMR0L=-NC%256;TMR0IF=0;

T100ms++;Tbase=1;

}

if(RCIF){

char t;

t = RCREG;

// RCIF=0;

if(t == 0x0D){

btr=rstr;

str2i();

}else if(t!=6){

\*btr = t;

++btr;

}

}

STATUS = SX;BSR=BX;WREG=WX;

retint();

}

bit ADC\_go\_done @ ADCON0.2;

//Global variables for LCD

// Data and tris bits, each bit may be defined at any io pin.

bit LCD4 @PORTC.2, LCD5 @PORTC.3, LCD6 @PORTC.4, LCD7 @PORTC.5,

LCD4T @TRISC.2, LCD5T @TRISC.3, LCD6T @TRISC.4, LCD7T @TRISC.5;

bit LCDR @PORTC.0, LCDRT @TRISC.0, // Control RS and its tris

LCDE @PORTC.1, LCDET @TRISC.1, // Control E and its tris

LCDS, LCDC ; // single control char, and control toggle flags

// Procedures related to LCD

void LCDw2u(char W){ // W x 3 microsecond for 4MHz.

WREG =W; do{ }while( --WREG ); }

void LCDNibble(char Ch)

{

LCD4T=0; LCD5T=0; LCD6T=0; LCD7T=0;

LCDRT=0; LCDET=0; LCDE = 0;

if(LCDC) LCDR=0; else LCDR = 1;

if(Ch.4) LCD4=1; else LCD4=0; if(Ch.5) LCD5=1; else LCD5=0;

if(Ch.6) LCD6=1; else LCD6=0; if(Ch.7) LCD7=1; else LCD7=0;

LCDE=1;

LCDw2u(2);

LCDE=0;

LCD4=0;nop();LCD5=0;nop();LCD6=0;nop();LCD7=0; LCDw2u(5);

}

void PrintLCD(const char \*Ch)

{

char WC,WP=0; LCDS=0 ; LCDC=0 ; LCDE=0 ;

do{ WC=Ch[WP]; WP++;

if(WC){ LCDC=0;

if(WC==0xFF) LCDS ^=1 ;

else { LCDC=WC.7||LCDS; if(WC<4) LCDC=1;

LCDNibble( WC & 0xF0);

if(LCDS && WC==0x28){//set mode takes 3ms time

char T=12;do{LCDw2u(0);}while(--T);}

LCDNibble( swap(WC) & 0xF0 );

if(WC<4){char T=12; do{ LCDw2u(0);}while(--T);}

LCDw2u(20); } }

}while(WC);

}

char\* i2a(uns16 k, char \*a){

char i = 5;

if(!k){

--i;a[i]=0;

}

while(k){ --i;a[i]=k%10; a[i]+='0'; k/=10;}

while(i){--i;a[i]=' ';}

a+=5;

return a;

}

void str2i() {

static char iy;

iy=0;

iwl=0;

/\*do{}while(!TXIF);TXREG=rstr[0];

do{}while(!TXIF);TXREG=rstr[1];

do{}while(!TXIF);TXREG=rstr[2];

do{}while(!TXIF);TXREG=rstr[3];

do{}while(!TXIF);TXREG=rstr[4];

do{}while(!TXIF);TXREG=0x0D;\*/

//tt=a[i]-'0';res += tt\*mult;

//res\*=10;res +=a[i]-'0';

//do{}while(!TXIF);TXREG=rstr[i];

while(iy<5){iwl\*=10;

iwl +=rstr[iy]-'0';++iy;}

if(iwl>1000)

iwl=1000;

//2,0,0,0,0 original in ISU

//0,0,0,0,2 recieved

}

void initADC() {

ADCON0 = 0b11.000.0.0.1;

ADCON1 = 0b0.0.00.1110;

}

void startADC(){

ADC\_go\_done=1; do{}while(ADC\_go\_done); // wait ADC to be over

}

void InitPorts(void){

TRISB.0=0; // RB0 Blink alive LED out

TRISB.1=0;

TRISA.0 = 1;//POT input

TRISC.6=0; // TX UART

TRISC.7=1; //TR UART

} // RC7..RC0 output

void InitUART(){

TRISC.6=0;TRISC.7=1;

BRGH=1; SPBRG=25; TXEN=1;CREN=1;SPEN=1;

}

void initINT(){

TMR0IE = 1;TMR0IP=0;TMR0IF=0;

RCIE = 1; RCIF=0;RCIP=0;

IPEN = 0;GIE=1;PEIE=1;

}

char\* strcpy( char \*a, const char \*b)

{

char t;

do{

t=\*b;

\*a=t;

++a; ++b;

}while(t);

return --a;

}

void blink(void) {

static char BAcount;

if(BAcount == 0) LBA = on;

else LBA = off;

++BAcount;

if(BAcount > 9) BAcount = 0;

}

void main(void){

initINT();

InitPorts();

InitUART();

initADC();

Tbase=0;

LNW = off;

TMR0IF=1;

T100ms = 0;

//btr = rstr;

btr=rstr;

string\_ready = 0;

uns16 ipot=0,iwp=0,htmp;

iwl=0;

{char i=200; do{LCDw2u(250);}while(--i);}//reset

CLR\_LCD;

PrintLCD("\x80Welcome...");

char \*pt;

do{

if(T100ms>50) {

pt = strcpy(ustr,"\x80L Water=");

iwp = h - iwl;

htmp = h / 100;iwp/=htmp;

pt = i2a(iwp,pt);

\*pt='%';pt++;\*pt=0x0D;

PrintLCD(ustr);

startADC();

ipot = ADRESH;

ipot\*=100;ipot/=255;

pt = strcpy(bstr,"\xC0L POT =");

pt = i2a(ipot,pt);

\*pt='%';pt++;\*pt=0;

PrintLCD(bstr);

if(iwp<ipot)

LNW = on;

else

LNW = off;

T100ms = 0;

}

if(Tbase){

blink();

Tbase=0;

}

}while(1);

}

# Test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| test | Sensor echo | Water % | Pot % | alarm |
| 1 | 1ms | 82% | 75% | No |
| 2 | 1.5ms | 73% | 75% | Yes |
| 3 | 2ms | 64% | 50% | No |
| 4 | 2.5ms | 58% | 50% | No |
| 5 | 3ms | 48% | 50% | Yes |
| 6 | 3.5ms | 39% | 50% | Yes |
| 7 | 4ms | 33% | 20% | No |
| 8 | 4.5ms | 23% | 20% | No |
| 9 | always high | 0% | 20% | Yes |

# Conclusion

This system used almost all what we learned on the CMPE423 course. We made a time base with TMRO with and without interrupts also we used the UART unit to transmit from and to microcontrollers. Also we made a communication mechanism between microcontroller. Learned how to use analog to digital converter, configure and working with LCD display also program using CC8E and simulate on protus8. Making a complete project allowed us to learn the complete system well.

Although we got troubles in using UART to transfer the strings from and to UC also we encountered a problem in converting number from string to int and visa versa. We stopped also in uns16 intermediate overflow but with a small algebra we did it correctly at the end.

Here is out work distribution table:

|  |  |  |
| --- | --- | --- |
| Task | Name of member | action |
| Install protues | M.ALI H. KHONSARI | Implementing |
| Simulation of sensor | CHARLES BROWN | Implementing, testing |
| Time calculation ISU | ÇAĞRI UĞUREL | Implementing |
| ISU UART | CHARLES BROWN | Implementing, testing |
| ISU | CHARLES BROWN | testing |
| Blink alive ISU + MAU | ÇAĞRI UĞUREL | Implementing, testing |
| MAU UART | MAHDY MOHAMED NASR | Implementing, testing |
| MAU LCD | MAHDY MOHAMED NASR | Implementing, testing |
| MAU ADC | MAHDY MOHAMED NASR | Implementing, testing |
| MAU calculatoins | M.ALI H. KHONSARI | Implementing |
| Over all testing | ADEMOLA MAKINDE | testing |