Project title: Thermometer

1. Abstract/Summary

The project team with two members should handle a mini project within two months. We choose the thermometer as our project topic. The project is aimed at developing a thermometer by using DS18B20 temperature sensor and PIC18F4520 microcontroller board.

2. Objectives

In detail, we are going to measure the temperature in degree Celsius with 2 decimal places precision, the output will be shown in 7 segment 4 digits LED display.

The project tools included: A PIC18F4520 microcontroller board, PICkit3, 2 Dupont lines, DS18B20 sensor.

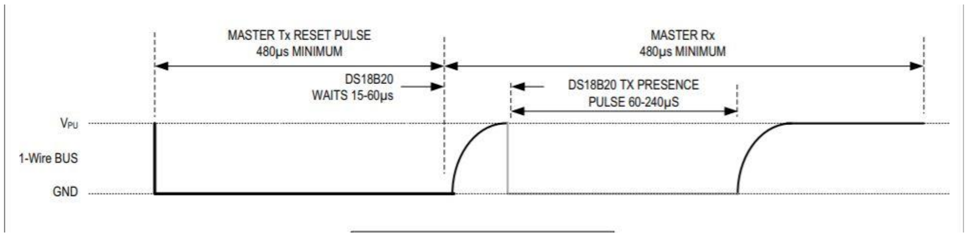
3. Introduction/Background

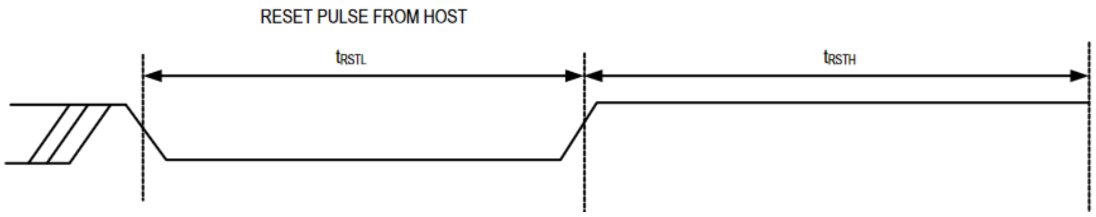
The course is introducing a new programming language Assembly language, provided a mini-project requests student to choose a topic within a given questions using mplab. The mark criteria based on the question difficulty and our degree completion. What idea to use is based on what learning experience on assembly language and self-learning, or C language can also be chosen to code.

4. Methodology/Procedure

a. Reset

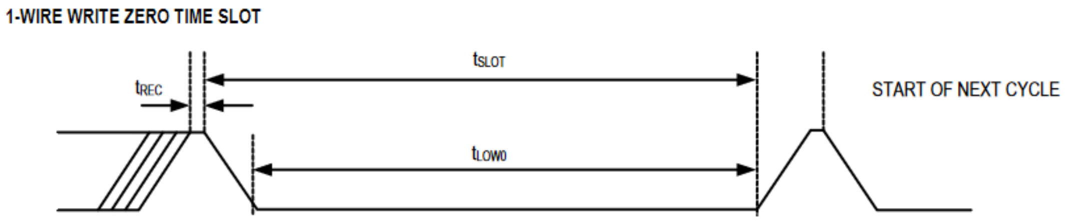
Set the TRISA to 0 (output) and the value of RA to 0 (low). And then wait for a 500us (480us minimum) time delay. After the delay had completed, set the TRISA to 1 (input) and wait for a 70us (scope 60-240 us) time delay. At that moment, we check the value of RA4. If it changed to 1 (high), it means that the reset operation has done, otherwise, it failed to reset. It waits for 430us time delay either way.

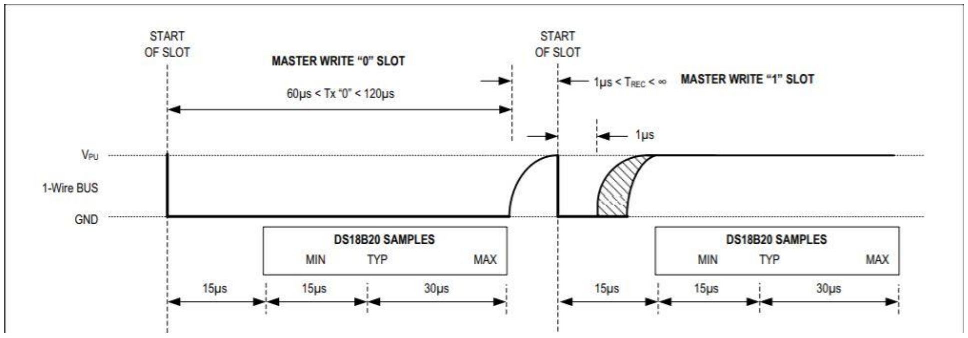




b. Write byte

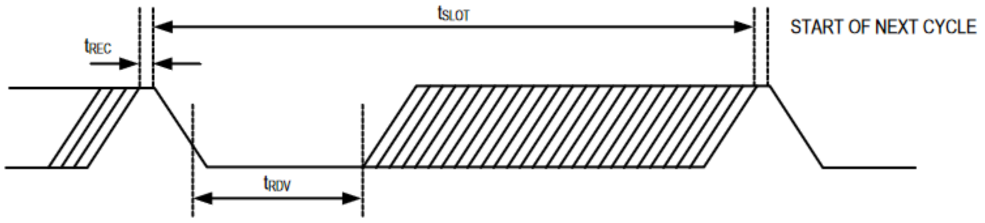
Declare the parameter as “data”. Perform a data&0x01 AND operation to get to lease significant bit. Set the TRISA to 0 (output) and the value of RA to 0 (low). Wait for a 5us time delay and then check the value of lease significant bit. If the value is 1 (high), then set the TRISA to 1 (input). Wait for a 70us time delay and then set the TRISA to 1 (input). 2 us delay afterward, perform a right 1-bit shift to data in order to get the next bit. Do the above operations 8 times to write a byte to the sensor.

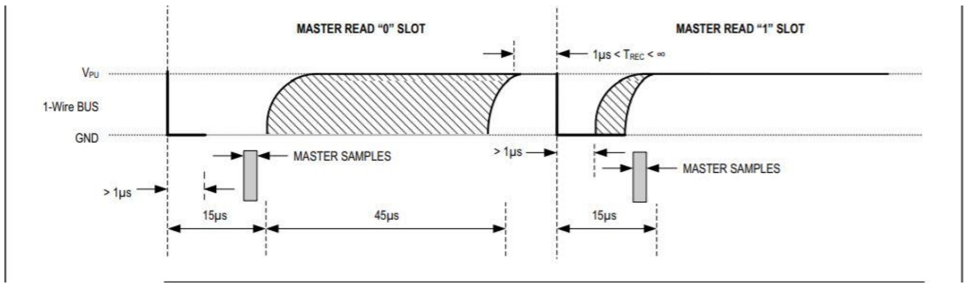




c. Read byte

Declare a return buffer “buffer” for the read byte and initialize it to 0. Perform a right 1-bit shift to buffer in order to read the next bit. Set the TRISA to 0 (output) and the value of RA to 0 (low). Wait for 6us time delay and then set the TRISA to 1 (input). Wait for 4us time delay, if the value of RA4 now equals to 1, perform a buffer|0x80 OR operation to write the most significant bit to buffer. Wait for another 6us time delay. Do the above operations 8 times to read and write a byte to the buffer. Lastly, return the variable buffer.



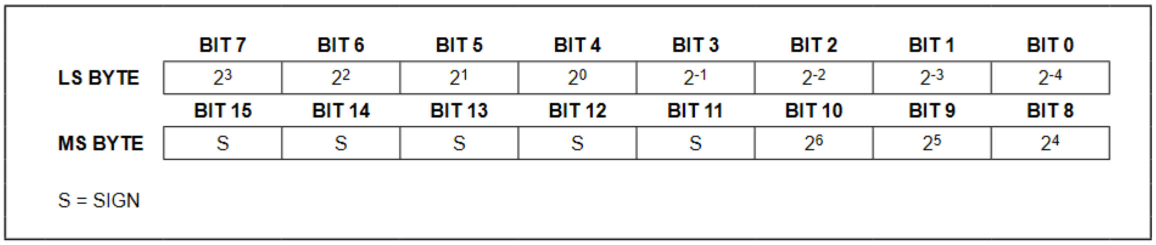


d. Byte conversion

Firstly, we must call the function reset and check whether its successful or not. If it has no error, write a byte 0xcc to skip the ROM and write a byte 0x44 to start the conversion. If it failed the reset operation, it returns 0, otherwise, return 1.

e. Read flash

Declare 2 varieties “lsb” and “msb” to store 2 bytes. We must call the function reset and check whether its successful or not. If it has no error, write a byte 0xcc to skip the ROM and write a byte 0xbe to read the 9 bits scratchpad. Assign the byte it reads to “lsb” and next byte to “msb”. Perform a lsb&0x0f to get the lease significant 4 bits as temp2 which is the decimal places. Perform (lsb>>4)|(msb<<4) to get the lease significant 4 bits of msb and most significant 4 bits of lsb, and combine them together to get the unit and tens as temp1.



	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

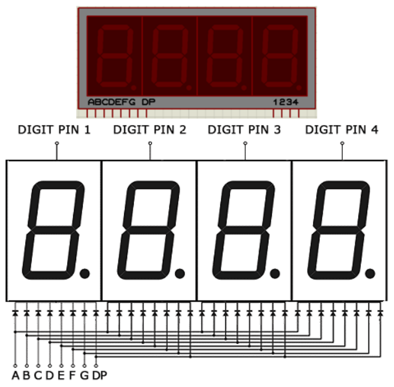
S = SIGN

f. Display

Set ADCON1=0x07 to set all pin to digital. Set the TRISB to 0 (output) and the value of RB to 0 (low) in order to switch the digits we are going to display. Only 3 bits are used in this situation. When RB=000, the first digit will be displayed. When RB=100, the second digit will be displayed. When RB=010, the third digit will be displayed. When RB=110, the fourth digit will be displayed.

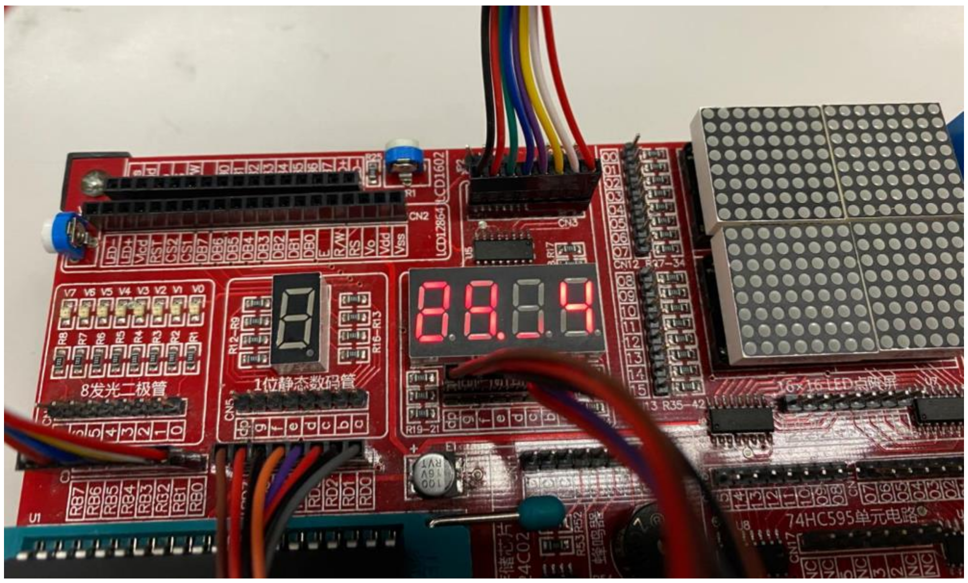
Set the TRISD to 0 (output) and the value of RD to 0 (low) in order to enable to LED segments. Declare an LED enabler array “number” {0x3F,0x06,0x5B,0x4F,0x66, 0x6D,0x7D,0x07,0x7F,0x6F} for the unit and tens digit of temperature. Declare 2 LED enabler arrays “number1” and “number2” for the decimal places based on the calculation. Since temp2 is only stored in 4bits, we can compute the actual temperature in an easy way. For example, value of temp2 is 12 (1100), the actual temperature should be 2^-1+2^-2=0.75. We put the 0x07 (led enabler of 7) and 0x6D (led enabler of 5) respectively in both arrays 12th index places.

Call the convert and readflash function. set RB=000 and perform temp1/10 to get the tens digit and set PORTD to number[temp1/10] to show the digit on the first LED. Wait for a 2ms time delay. Set RB=100 and perform temp1%10 to get the unit digit and set PORTD to number[temp1%10] to show the digit on the second LED. Set RB=010 and set PORTD to number1[temp2] to show the digit on the third LED. Wait for a 2ms time delay. Set RB=110 and set PORTD to number2[temp2] to show the digit on the fourth LED. Wait for a 2ms time delay and keeping doing from the beginning.



5. Observations/Results

   The result works as our expectation. The temperature in degree Celsius with 2 decimal places precision will be shown in 7 segment 4 digits LED display. 2ms time delay between each of the segments to ensure it would not blink too quickly and the user is not able to notice the changes.



(It is always showing an unclear look if we take a photo on it)

6. Discussion and Future work
   a. Delay part (choosing hardware/device)

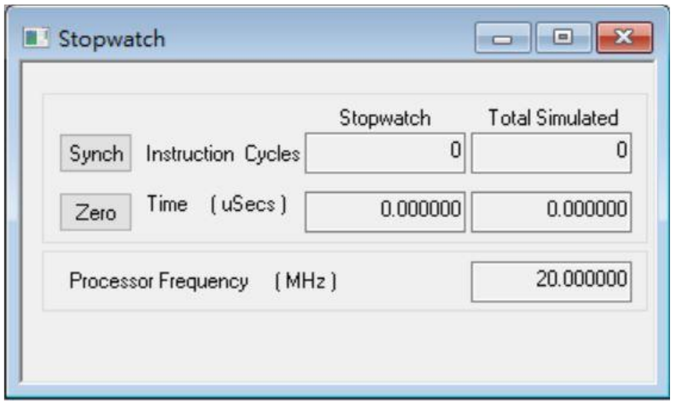
Based on what devices are used to simulate the code in the MPLAB, there will be a difference between delay detection. For instance, when computer in lab is used, the difference among delay detection can be up to 20 times from initial values; On the other hand, when own computer is used, the difference among delay detection is only 18 times from initial values. Inaccurate or delay settings causes errors on temperature (error = different from real temperature).

b. Calculate the time used for every value increment and decrement (software/mplab)

(This step is to pursue more precise temperature detection, can be skipped)

(The results based on what device is used)

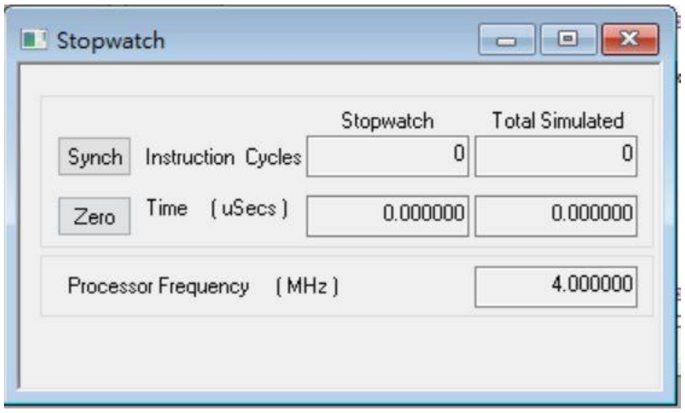
There is a time delay for every value getting increased or decreased, but we don’t know how many times used for every while-loop (or for-loop, but less to be used) runs. Hence, before the main function() and temperature part start, we have to use stopwatch function to see how many times for values run:


Stopwatch

		Stopwatch	Total Simulated
Synch	Instruction Cycles	0	0
Zero	Time (uSecs)	0.000000	0.000000

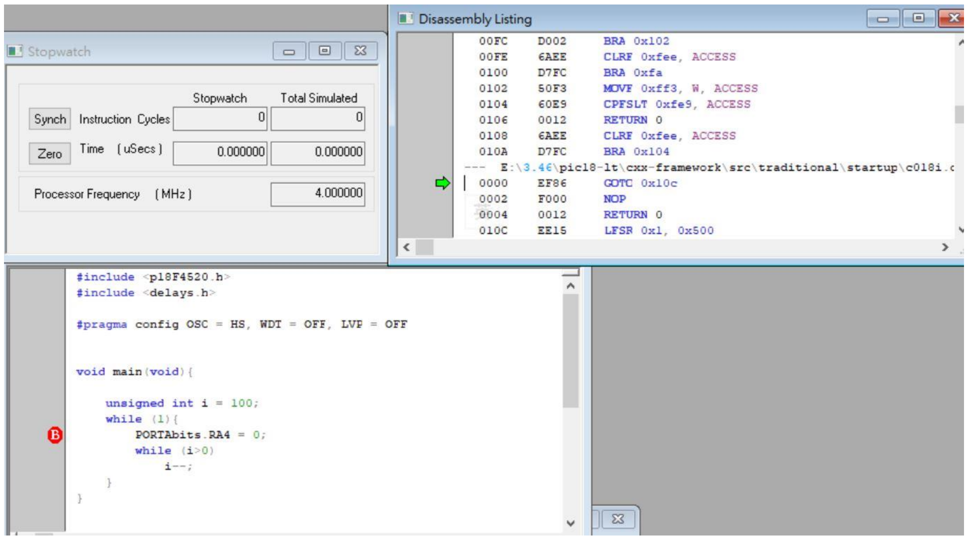
Processor Frequency (MHz) 20.000000

Firstly, we have to set 'Processor Drequency' from 20 MHz to 4 MHz as our PIC18F4520 is using 4 MHz.



```
                    Stopwatch    Total Simulated
Instruction Cycles          0                  0
Time ( uSecs )       0.000000           0.000000
Processor Frequency ( MHz )             4.000000
```

After that, we code a very simple function to test the simulation.



```
                    Stopwatch    Total Simulated
Instruction Cycles          0                  0
Time ( uSecs )       0.000000           0.000000
Processor Frequency ( MHz )             4.000000
```

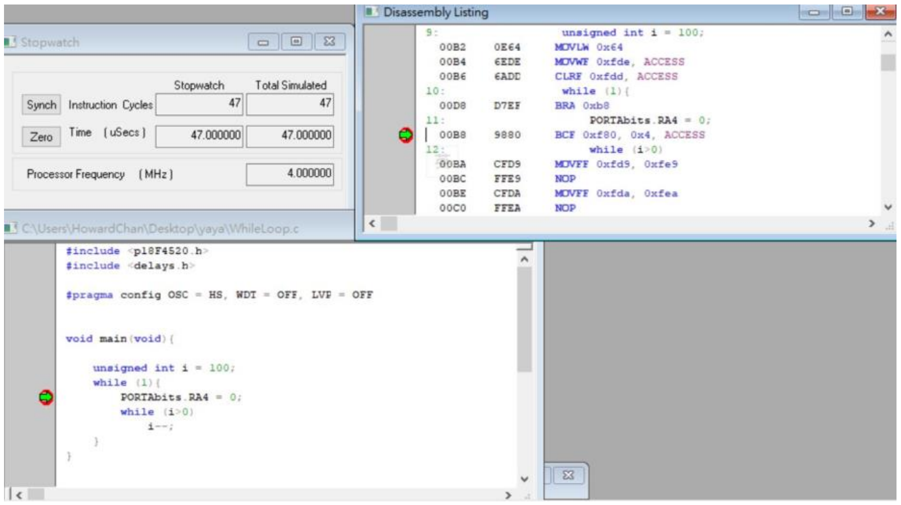
```
00FC     D002     BRA 0x102
00FE     6AEE     CLRF 0xfee, ACCESS
0100     D7FC     BRA 0xfa
0102     50F3     MOVF 0xff3, W, ACCESS
0104     60E9     CPFSLT 0xfe9, ACCESS
0106     0012     RETURN 0
0108     6AEE     CLRF 0xfee, ACCESS
010A     D7FC     BRA 0x104
---  E:\3.46\pic18-lt\cxx-framework\src\traditional\startup\c018i.c
0000     EF86     GOTO 0x10c
0002     F000     NOP
0004     0012     RETURN 0
010C     EE15     LFSR 0x1, 0x500
```

```c
#include <p18F4520.h>
#include <delays.h>

#pragma config OSC = HS, WDT = OFF, LVP = OFF


void main(void){

    unsigned int i = 100;
    while (1){
        PORTAbits.RA4 = 0;
        while (i>0)
            i--;
    }
}
```



```
                    Stopwatch    Total Simulated
Instruction Cycles         47                 47
Time ( uSecs )      47.000000          47.000000
Processor Frequency ( MHz )             4.000000
```

```
9:                        unsigned int i = 100;
  00B2     0E64     MOVLW 0x64
  00B4     6EDE     MOVWF 0xfde, ACCESS
  00B6     6ADD     CLRF 0xfdd, ACCESS
10:                       while (1){
  00D8     D7EF     BRA 0xb8
11:                           PORTAbits.RA4 = 0;
  00B8     9880     BCF 0xf80, 0x4, ACCESS
12:                           while (i>0)
  00BA     CFD9     MOVFF 0xfd9, 0xfe9
  00BC     FFE9     NOP
  00BE     CFDA     MOVFF 0xfda, 0xfea
  00C0     FFEA     NOP
```

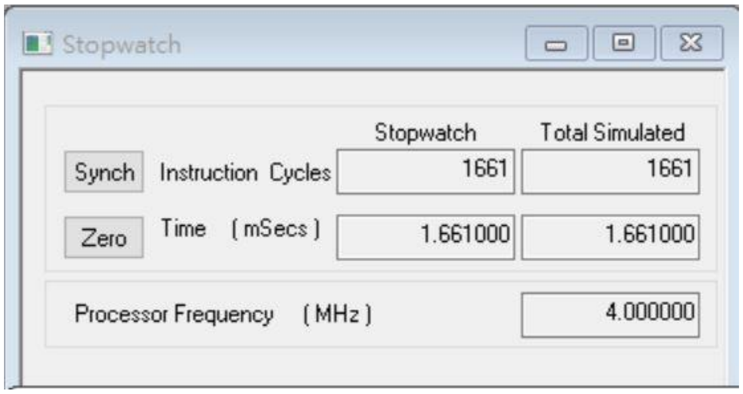
```c
#include <p18F4520.h>
#include <delays.h>

#pragma config OSC = HS, WDT = OFF, LVP = OFF


void main(void){

    unsigned int i = 100;
    while (1){
        PORTAbits.RA4 = 0;
        while (i>0)
            i--;
    }
}
```

Keep pressing F7 (Step into function until the pointer pointing to PORTAbits.RA4, record the current Instruction Cycles (47) and time runs (47u secs)), start running whole while loop until the pointer pointing the PORTAbits.RA4 again.

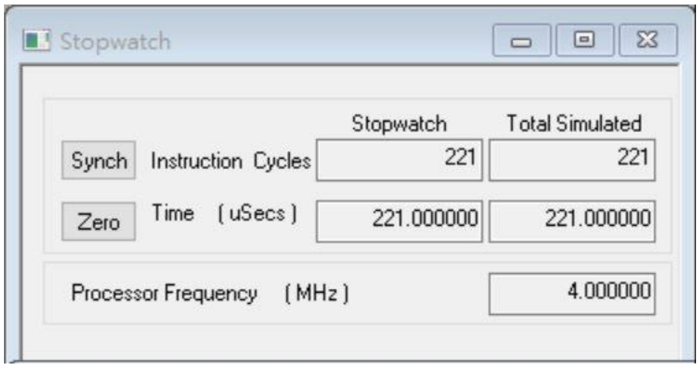


	Stopwatch	Total Simulated
Instruction Cycles	1661	1661
Time (mSecs)	1.661000	1.661000
Processor Frequency (MHz)		4.000000

(Total run time after while-loop for unsigned integer ‘i’ equals to 100)

1.661 msecs --> 1661 usecs minus(-) 47 usecs then divided by 100, we get 16.14 usecs per one loop (for value ‘i' from 100 to 0).

Repeat this part, but we choose ‘i’=10 this time.



	Stopwatch	Total Simulated
Instruction Cycles	221	221
Time (uSecs)	221.000000	221.000000
Processor Frequency (MHz)		4.000000

221 usecs minus(-) 47 usecs then divided by 10, we get 17.4 usecs per one loop (for value ‘i’ from 10 to 0)

For cases of 100 and 10 decrement, we pick ‘18’ for delay every function.

If the delay needs 540 usecs, we pick unsigned integer ‘i’ = 30.

c. Decimal places array

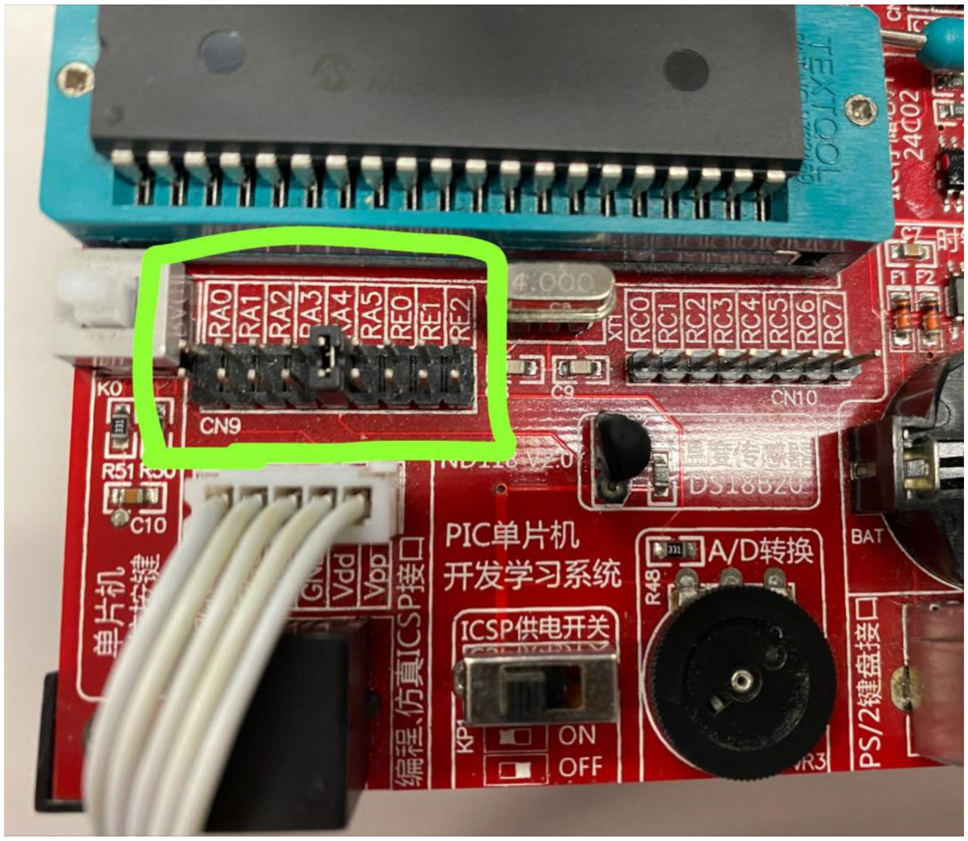
Calculate the decimal places automatically instead of the calculation manually. Previously we hardcoded the LED enabler in the 2 arrays number1 and number2, but in fact, it can be computed automatically by doing the division and bit shifting operations.

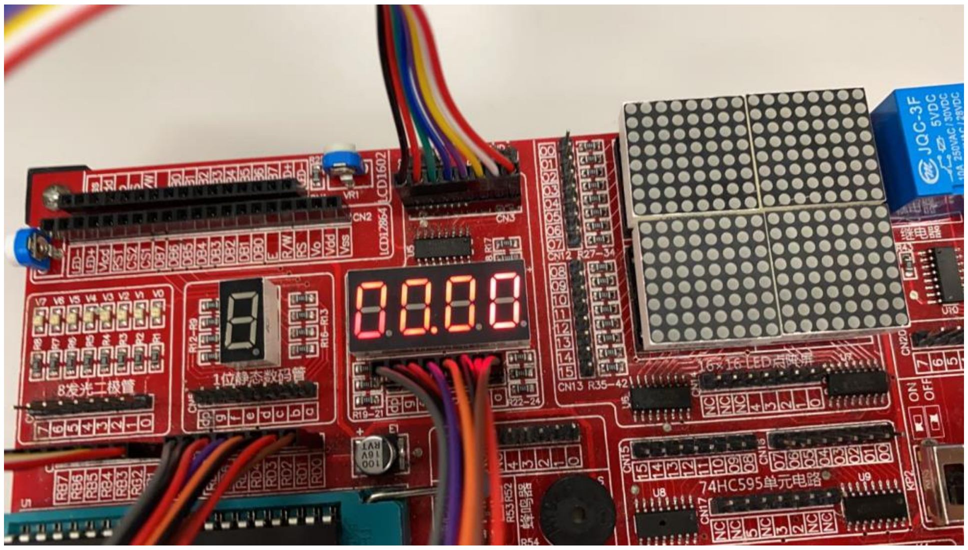
As the decimal value, for first decimal value, since it is a Hexadecimal value, we can firstly draw a truth table:

2^-1	2^-2	2^-3	2^-4	Total	Transfer
0	0	0	0	0	0
0	0	0	1	0.0625	0
0	0	1	0	0.125	1
0	0	1	1	0.1875	1
0	1	0	0	0.25	2
0	1	0	1	0.3125	3
0	1	1	0	0.375	3
0	1	1	1	0.4375	4
1	0	0	0	0.5	5
1	0	0	1	0.5625	5
1	0	1	0	0.625	6
1	0	1	1	0.6875	6
1	1	0	0	0.75	7
1	1	0	1	0.8125	8
1	1	1	0	0.875	8
1	1	1	1	0.9375	9

d. To be care

Must make sure that we do not insert the DS18F20 in the wrong way. Be aware that insert it in opposite way does shorted, burn out and damaged the sensor. Also, make sure you put the jumper on the RA4 pin, otherwise it does not work at all. Missing this step causes failure of doing temperature detection.





(This figure shows a case if we didn't pin RA4, as part D mentions, the display showing 00.00)

7. Conclusions

To conclude, the mini project request to build up a small function within a limited time, the project is also found to be difficult for a lack of knowledge of the PIC8F4520 microcontroller, and who have a lack of an experience to code.

8. References

DS18B20 Programmable Resolution 1-Wire Digital Thermometer. (2019). https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Microchip Technology. (2008). PIC18F2420/2520/4420/4520 Data Sheet. https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf.

9. Appendix

All arrays for the digits display

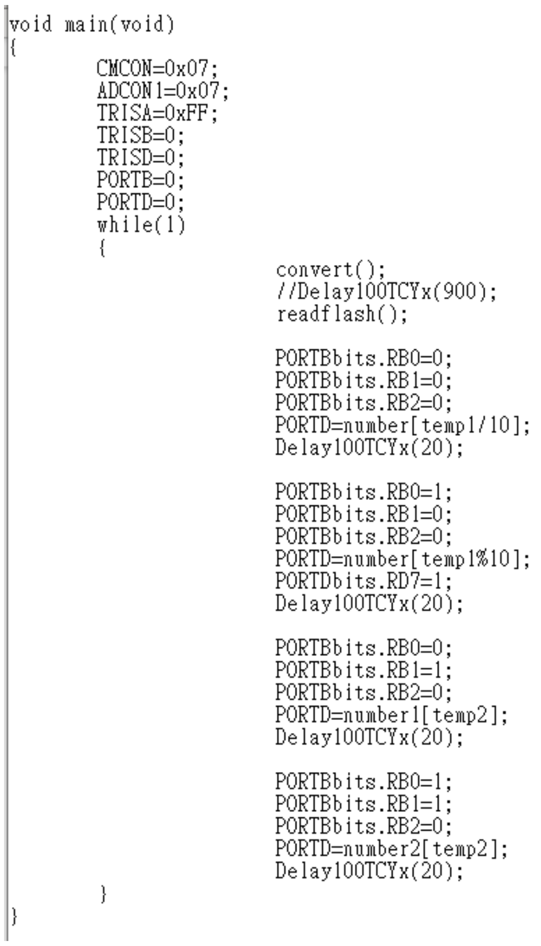
```
unsigned char number[10]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};

unsigned char number1[16]={0x3F,0x3F,0x06,0x06,0x5B,0x4F,0x4F,0x66,0x6D,
0x6D,0x7D,0x7D,0x07,0x7F,0x7F,0x6F};

unsigned char number2[16]={0x3F,0x7D,0x4F,0x6F,0x6D,0x06,0x7F,0x4F,0x3F,0x7D,
0x4F,0x6F,0x6D,0x06,0x7F,0x66};
```

All functions related to the DS18B20

All main function



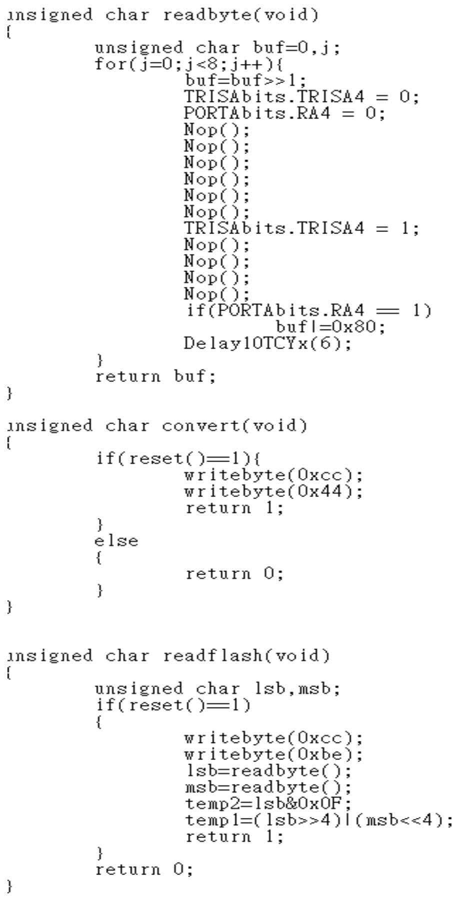
```
void main(void)
{
        CMCON=0x07;
        ADCON1=0x07;
        TRISA=0xFF;
        TRISB=0;
        TRISD=0;
        PORTB=0;
        PORTD=0;
        while(1)
        {
                        convert();
                        //Delay100TCYx(900);
                        readflash();

                        PORTBbits.RB0=0;
                        PORTBbits.RB1=0;
                        PORTBbits.RB2=0;
                        PORTD=number[temp1/10];
                        Delay100TCYx(20);

                        PORTBbits.RB0=1;
                        PORTBbits.RB1=0;
                        PORTBbits.RB2=0;
                        PORTD=number[temp1%10];
                        PORTDbits.RD7=1;
                        Delay100TCYx(20);

                        PORTBbits.RB0=0;
                        PORTBbits.RB1=1;
                        PORTBbits.RB2=0;
                        PORTD=number1[temp2];
                        Delay100TCYx(20);

                        PORTBbits.RB0=1;
                        PORTBbits.RB1=1;
                        PORTBbits.RB2=0;
                        PORTD=number2[temp2];
                        Delay100TCYx(20);
        }
}
```



```
unsigned char readbyte(void)
{
        unsigned char buf=0,j;
        for(j=0;j<8;j++){
                buf=buf>>1;
                TRISAbits.TRISA4 = 0;
                PORTAbits.RA4 = 0;
                Nop();
                Nop();
                Nop();
                Nop();
                Nop();
                Nop();
                TRISAbits.TRISA4 = 1;
                Nop();
                Nop();
                Nop();
                Nop();
                if(PORTAbits.RA4 == 1)
                        buf|=0x80;
                Delay10TCYx(6);
        }
        return buf;
}

unsigned char convert(void)
{
        if(reset()==1){
                writebyte(0xcc);
                writebyte(0x44);
                return 1;
        }
        else
        {
                return 0;
        }
}


unsigned char readflash(void)
{
        unsigned char lsb,msb;
        if(reset()==1)
        {
                writebyte(0xcc);
                writebyte(0xbe);
                lsb=readbyte();
                msb=readbyte();
                temp2=lsb&0x0F;
                temp1=(lsb>>4)|(msb<<4);
                return 1;
        }
        return 0;
}
```