

# First Assembly Program

EE 310/EE310L - Microcontroller - Spring 2023



NAME: Mathew Teague

DATE: 3/4/24

Please make sure you submit your assignment individually!

## Assignment # 4

<https://docs.google.com/document/d/1PB5FNCLEhWM3IZfFijavK2h4Uno6AXZn/edit?usp=sharing&ouid=111229422470150013614&rtopof=true&sd=true>

### 1. Assignment Overview

In this experiment we will be using the assembly instructions we have learned so far and writing an assembly program.

### 2. Learning Objectives

By the end of this lab, you will

- Practice with PIC assembly instructions
- Practice writing a good code
- Learn to develop a software flowchart
- Learn to draw a high-level flow diagram of the system

### 3. Review

Consider reviewing the following before you start this assignment (must be logged on SSU to open the links):

- Review the assembly instruction [command list](#).
- Review Mazidi- Chapters 2 & 3
- Review how to write [IF ELSE statements](#)
- Full datasheet for the 40PIN PDIP PIC18F46K42  
[https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18\(L\)F26-27-45-46-47-55-56-57K42-Data-Sheet-40001919G.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18(L)F26-27-45-46-47-55-56-57K42-Data-Sheet-40001919G.pdf)

**ATTENTION: THIS ASSIGNMENT MAY TAKE YOU SOME TIMES TO COMPLETE! PLEASE START EARLY.**

### 4. Materials

You need the following to complete this assignment:

- No hardware is required for this project.
- You must use MPLAB X to complete this assignment.

## 5. Background Information

In this project we develop a heating and cooling control system. The basic idea is that the user sets the desired temperature. If the environment temperature (e.g., room temp) is higher than the reference temperature, the cooling system will be turned on. On the other hand, if the environment temperature falls below the reference temperature, the heating systems will start.

## 6. Assignment

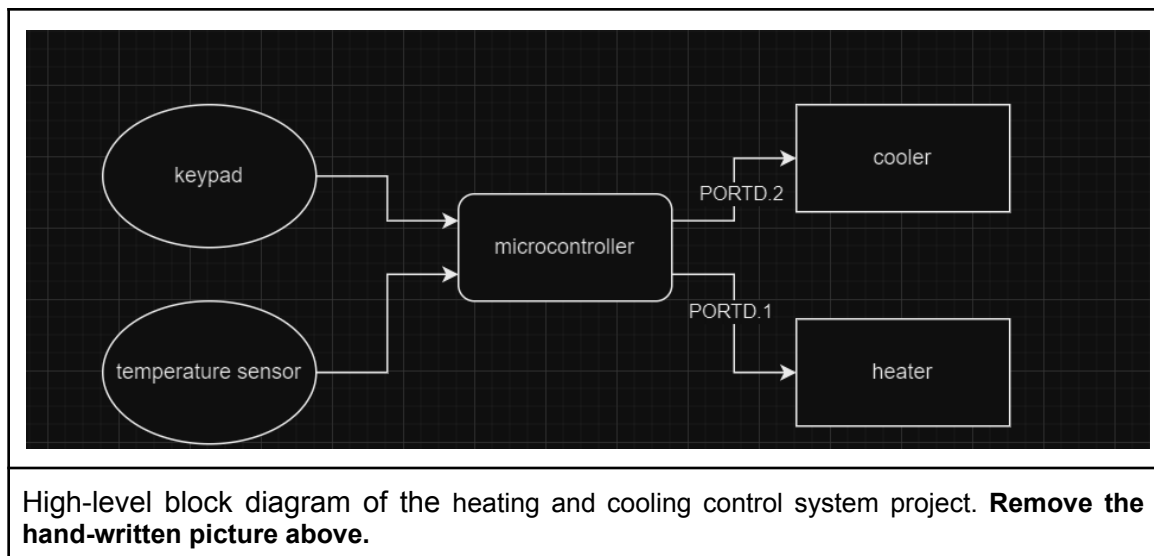
Complete the following steps.

### 6.1 System Description

Assume our cooling and heating control system consists of the following parts, as shown in the figure below:

- 1- Temperature sensor for measuring the temperature.
- 2- Keypad to enter the desired (or reference) temperature.
- 3- A microcontroller to control the heating and cooling systems.
- 4- A cooling fan with an LED that can be turned on or off using a digital signal.
- 5- A hot air blower with an LED that can be turned on or off using a digital signal.
- 6- The keypad and Sensor values must be converted to decimal and placed into appropriate registers.

Using the description above, draw the high-level block diagram of this system. You can use draw.io, VISIO, or any other professional software for drawing block diagrams. DO NOT use Paint or similar tools, as they are not appropriate for depicting block diagrams.



## 6.2 Microcontroller Operation

Assume the `refTemp` represents the desired environmental temperature and `measuredTemp` is the actual measured temperature using the temperature sensor. The controller should compare the two values and perform the following operations (we refer to these as the *operating requirements*)

- R1-** If `measuredTemp > refTemp` → `contReg=2` → Turn on PORTD.2, which is connected to the COOLING system
- R2-** If `measuredTemp < refTemp` → `contReg=1` → Turn on PORTD.1, which is connected to a the HEATING system
- R3-** If `measuredTemp = refTemp` → `contReg=0` → display nothing

Other system requirements include the following:

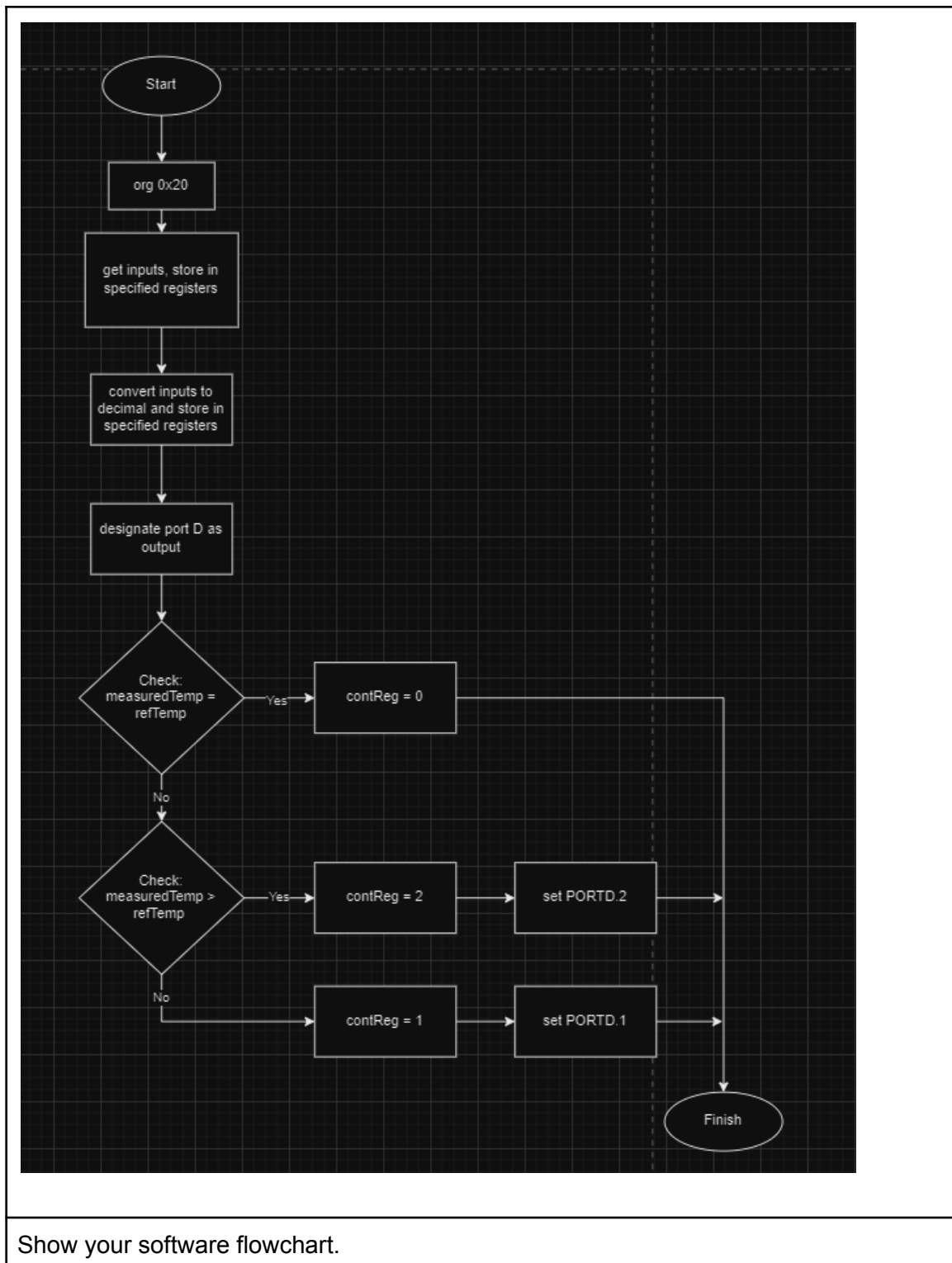
- R4-** The `refTemp` can be between +10 Degree celsius and +50 Degree celsius.
- R5-** The `measuredTemp` can be between -20 Degree celsius and +60 Degree celsius (can be negative).
- R6-** You can only use PORTD. No other ports can be used as output.
- R7-** Your program must start from register **0x20** in the program memory.
- R8-** Make sure the following variables are in the following register locations:  
`refTemp` → REG 0x20, `measuredTemp` → REG 0x21, `contReg` → REG 0x22

The last two challenging requirements:

- R9-** The decimal equivalent value of `refTemp` must be stored in Registers 0x60,61,62. For example, if the `refTemp=44d` then `0x62=0x00,0x61=04,0x60=04`.
- R10-** The decimal equivalent value of `measuredTemp` must be stored in Registers 0x70,71,72. For example, if the `refTemp=94d` then `0x72=0x00,0x71=09,0x70=04`.

System requirements.

Show your software flowchart. You can use Idraw.net, VISIO, or any other professional software for drawing flowcharts. DO NOT use Paint or similar tools, as they are not appropriate for displaying flowcharts.



### 6.3 Code Header

All programs should include, at or near the beginning of the program, a comment block with at least the following information: the programmer's name and appropriate identification, the date of writing (and possibly dates and purpose of major modifications), the name of the program, the

computer and operating system for which it was written. Include information about compiling the program (i.e., the compile line commands). An example program header block in ASM might look as follows:

```

1  ;-----
2  ;   Title: Heating and Cooling System Controller
3  ;-----
4  ;Purpose: This program controls a heating/cooling system
5  ;Dependencies: NONE
6  ;Compiler: MPLAB X IDE v6.20
7  ;Author: Mathew Teague
8  ;OUTPUTS:
9  ;   PORTD.1 - Heating system
10 ;   PORTD.2 - Cooling system
11 ;INPUTS:
12 ;   keypad -> refTemp
13 ;   temperature sensor -> measuredTemp
14 ;Versions:
15 ;   V1.0: 2/29/2024 - First Version
16 ;   V2.0: 3/4/2024 - I had to overhaul the code to match the updated
17 ;                   assignemnt requirements
18 ;   V2.1: 3/5/2024 - Added a code block that accounts for negative inputs
19 ;   V2.2: 3/6/2024 - Changed some of the register names so the code is
20 ;                   is easier to understand
21 ;                   - Added code for converting the inputs into decimal
22 ;                   - Changed how PORTD is set
23 ;                   - Finished comments
24 ;   V2.3: 3/6/2024 - Changed how the registers names were defined
25 ;                   - Polished old comments and added new ones
26 ;                   - Fixed how negative numbers are displayed
27 ;                   - Finalized header
28 ;   V2.4: 3/6/2024 - Finalized header again
29 ;                   - Fixed PORTD setup problem involving ANSEL

```

Take a snapshot of your program comment block - **REMOVE THE SAMPLE CODE**

## 6.4 Assigning the Inputs, Outputs & Other Constants

In your program you need to define the inputs, outputs and other constants as follow:

```

#-----
# PROGRAM INPUTS
#-----
#define measuredTempInput    45 ; this is the input value
#define refTempInput        25 ; this is the input value

#-----
# REGISTERS
#-----
#define measuredTempREG      0x20 ; this is
#define refTempREG          0x21 ; this is

. . . etc.

```

```
#-----
# PROGRAM OUTPUTS
#-----
#define measuredTempDecimal 0x70 ; this is the input value
#define refTempInputDecima ; this is the input value
#define HEATER PORTD,2
#define COOLER PORTD,1
```

and so on...

## 6.5 Write the Main Code

The best way to approach this assignment is to break down the problem into smaller pieces:

1- First, assign the start of the code in the program memory and define all the constants with proper register numbers.

2- Assign some arbitrary values to the inputs: `refTemp=0x5` & `measuredTemp=6`. Write your code such that requirements R1-R3 are met. Here is the basic pseudo example code for this part (this is not the solution to this problem!):

```
If measuredTemp=refTemp then
    set contReg=0
    GOTO LED_OFF
If measuredTemp>refTemp then
    set contReg=2
    GOTO LED_HOT
ELSE
    set contReg=1
    GOTO LED_COOL

LED_HOT {do something or nothing}
LED_COOL {do something or nothing}
LED_OFF {do something or nothing}
```

**NOTE: your code and your software flowchart MUST match! So, make sure your flowchart properly represents what you have coded.**

3- After ensuring that the logic works properly and the `contReg` has the correct values write the code to setup the PORTD properly. Here is the basic pseudo code for this part::

```
LED_HOT {
    Display H
    TURN ON hotAir}
    TURN OFF coolAir
LED_COOL {
    TURN OFF hotAir
    TURN OFF coolAir}
LED_OFF {Display nothing & TURN OFF all}
```

```

91      MOVLW    measuredTemp    ;check if measuredTemp is negative before converting
92      ADDLW    0x0              ;to decimal
93      BNN      _NOTNEGATIVE
94      NEGF     measuredTempReg ;switch to positive if negative
95
96      _NOTNEGATIVE:
97      ;Convert refTemp to decimal and store in regs
98      ;I repurposed an algorithm from the Mazidi textbook (pg 164)
99      MOVLW    refTemp          ;converting refTemp first
100     MOVWF    NUME              ;load numerator
101     MOVLW    MYDEN
102     CLRF     QU,1              ;clear quotient reg
103     refD_1:
104     INCF     QU,1              ;increment QU for every subtraction
105     SUBWF    NUME              ;NUME - WREG, store in wreg
106     BC       refD_1           ;if NUME > 0, repeat
107     ADDWF    NUME              ;put number back in NUME
108     DECF     QU,1              ;decrement QU since we reversed the subtraction
109     MOVFF    NUME,refD1        ;save the low digit
110     MOVFF    QU,NUME           ;repeat one more time
111     CLRF     QU               ;clear QU
112     refD_2:
113     INCF     QU,1
114     SUBWF    NUME              ;sub WREG from NUME value
115     BC       refD_2           ; (C = 1 for positive)
116     ADDWF    NUME              ;once too many
117     DECF     QU,1
118     MOVFF    NUME,refD2        ;save middle digit
119     MOVFF    QU,refD3          ;save high digit, always 0 for our case
120
121     ;Convert measuredTemp to decimal and store in regs
122     MOVFF    measuredTempReg, NUME
123     MOVLW    MYDEN
124     CLRF     QU,1              ;clear quotient reg
125     measuredD_1:
126     INCF     QU,1              ;increment QU for every subtraction
127     SUBWF    NUME              ;NUME - WREG, store in wreg
128     BC       measuredD_1      ;if NUME > 0, repeat
129     ADDWF    NUME              ;put number back in NUME
130     DECF     QU,1              ;decrement QU since we reversed the subtraction
131     MOVFF    NUME,measuredD1   ;save the low digit
132     MOVFF    QU,NUME           ;repeat one more time
133     CLRF     QU               ;clear QU
134     measuredD_2:
135     INCF     QU,1
136     SUBWF    NUME              ;sub WREG from NUME value, store in WREG
137     BC       measuredD_2      ; (C = 1 for positive)
138     ADDWF    NUME              ;once too many
139     DECF     QU,1
140     MOVFF    NUME,measuredD2   ;save middle digit
141     MOVFF    QU,refD3          ;save high digit, always 0 for our case
142
143     MOVLW    measuredTemp      ;make sure measuredTempReg has correct value,
144     MOVWF    measuredTempReg   ;in case it was negative
145

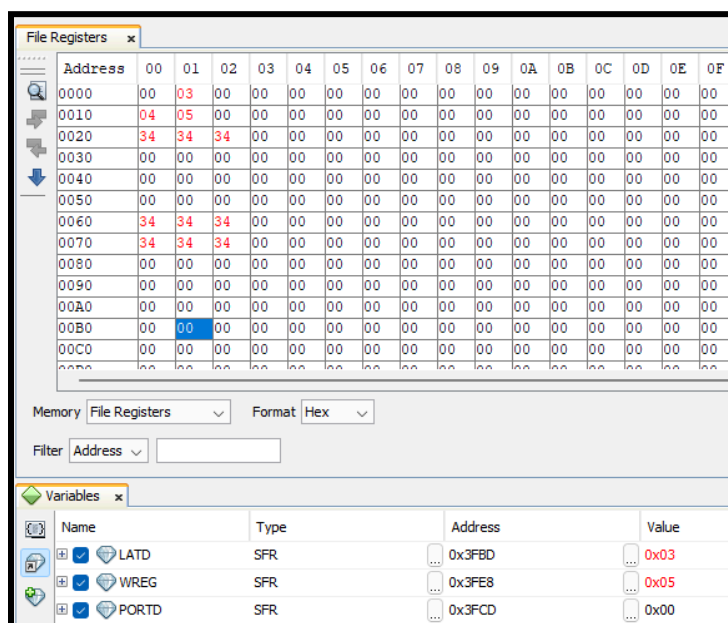
```

Show a snapshot of your program where you convert the HEX value to decimal and place them in REG 60-62 and 70-72. **Do not include your header block! Make sure you have proper comments in your code; don't add too many details.**

## 6.6 Debugging the Program

In order to properly debug your program you can arrange the FILE REGISTER and VARIABLE windows as shown below.

Note that PORTD may not toggle due to lack of correct setting (we will talk about this in the next lab). If you don't observe any changes in PORTD please examine LATD as shown in the figure below.



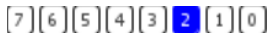
## 6.7 Testing your Software


It is important you make sure your project works for all the values within the given ranges, per R4-R5. Show a snapshot of REG20-REG24 for the following test cases:


Case 1: <b>EXAMPLE</b> refTemp = 15d measuredTemp = 5d			
REG62=00,01, 05	REG72=0,00,05	<div> <div>7</div> <div>6</div> <div>5</div> <div>4</div> <div>3</div> <div>2</div> <div>1</div> <div>0</div> </div>	REG22=0xF ,0x5, 0x1
Snapshot of REG60-62	Snapshot of REG70-72	Snapshot of your LED PORT	Snapshot of REG22-22

**NOTE: If it is easier just write the values of the registers instead of pasting the snapshots.**



<b>Case 2:</b> <code>refTemp = 15d</code> <code>measuredTemp = 20d</code>			
0,1,5	0,2,0		0xF, 0x14, 0x2
Snapshot of REG60-62	Snapshot of REG70-72	Snapshot of your LED PORT	Snapshot of REG20-25

<b>Case 3:</b> <code>refTemp = 15d</code> <code>measuredTemp = -5d</code>			
0,1,5	0,0,5		0F,FB,1
Snapshot of REG60-62	Snapshot of REG70-72	Snapshot of your LED PORT	Snapshot of REG20-25

<b>Case 4:</b> <code>refTemp = 15d</code> <code>measuredTemp = 0d</code>			
0,1,5	0,0,0		0F,0,1
Snapshot of REG60-62	Snapshot of REG70-72	Snapshot of your LED PORT	Snapshot of REG20-25

**NOTE: You will not receive any points if your code does not work properly or if you do not follow the requirements. Make sure you get your code to work first and you meet all the requirements. You must demonstrate your project to the instructor to receive a grade.**

## 7. GitHub

Once you have completed the code and debugged it. Submit the final code to your GitHub, in the Assignment folder. Make sure you name it properly. You should demonstrate your code to your instructor during the demo.

<a href="https://github.com/mteaguu/Microcontroller_EE310">https://github.com/mteaguu/Microcontroller_EE310</a>
Place the link to your GihHub page here.

## 8. Additional Features

In this problem we were given a number of different requirements. However, in order to improve the system's performance it is necessary to add more specific requirements. For example, what if the Reference Temp is set to a number larger than 60? What if the temperature sensor is not

reading any values, etc.? Define FIVE additional requirements that can improve the system. For each case explain how the system can be improved.

New Requirement:	How does it improve the system
EXAMPLE: Add a second LED PORT to display and show the reference temperature on the second display.	This can help the user to always know what the reference temperature is at any given time.
1- Add some 7 seg displays to display the decimal numbers we converted	This would help the user identify whether or not the microcontroller is getting the correct values from the sensor and the keypad
2- Make the program automatically restart whenever a new value is put in the keypad	The program needs to be restarted whenever a new value is given to refTemp, so this would change would just make the user's life easier
3- Keep checking temperature after turning on heater/cooler, so it can be turned off once the temperatures match	Right now, the program doesn't stop heating/cooling ever. This change would make the program far more practical
4- Make an output for when the two temperatures are equal	There isn't an indicator for when the values are equal, so the user can't tell if the microcontroller is just not working
5- Add audio for when the heater and cooler are turned on/off	If my previous suggestions were implemented, this would make it so the user could know if the heating/cooling system is working without having to look at the displays. Also, they would know if it's about to get colder or warmer.

## 9. Survey Questions

Answer the following questions, please:

Survey question	Response
On a scale of 1-10 how did you like this exercise? (10 is the best, 1 is the worst)	6
On a scale of 1-10 how much did you learn as a result of completing this exercise? (10 = plenty; 1=very little)	10
How many hours did you spend completing this exercise?	about 7.5

## 10. References

[1] Complete Electronics Self-Teaching Guide with Projects | Earl Boysen, Harry Kybett.  
ISBN: 978-1-118-28232-8 July 2012