**Subroutine call & MPLAB debugger**

*Lab #4*

**Phillip May,** 1639294

**ABSTRACT**

*In this lab I programmed to the PIC16F887 and learned how to use the debugger..*

Experiment Performed on **12 October 2017**

Report Submitted on **19 October 2017**



Department of Computer

Engineering Technology

**INTRODUCTION TO MICROPROCESSORS (247-302)**

*Day Yann Fong*

Table of Contents

[Introduction 3](#_Toc496168544)

[Equipement 3](#_Toc496168545)

[Procedure 3](#_Toc496168546)

[Results and Discussion 4](#_Toc496168547)

[Conclusion 7](#_Toc496168548)

[References 7](#_Toc496168549)

# Introduction

In this lab I will learn about to use the debugger function for the pickit3 in combination with PIC16F887 in order to help me debug code by adding breakpoints to see what part of my logic is problematic. I also learn about using subroutine calls. I also look at some other files generated by the compiler like the .lst file.

# Equipement

• PIC16F887

• Pickit3

• 2 100nF capacitors

• 4 LED

• 4 270ohm resistors

•10Kohm resistor

•1 push button

•Connecting wires

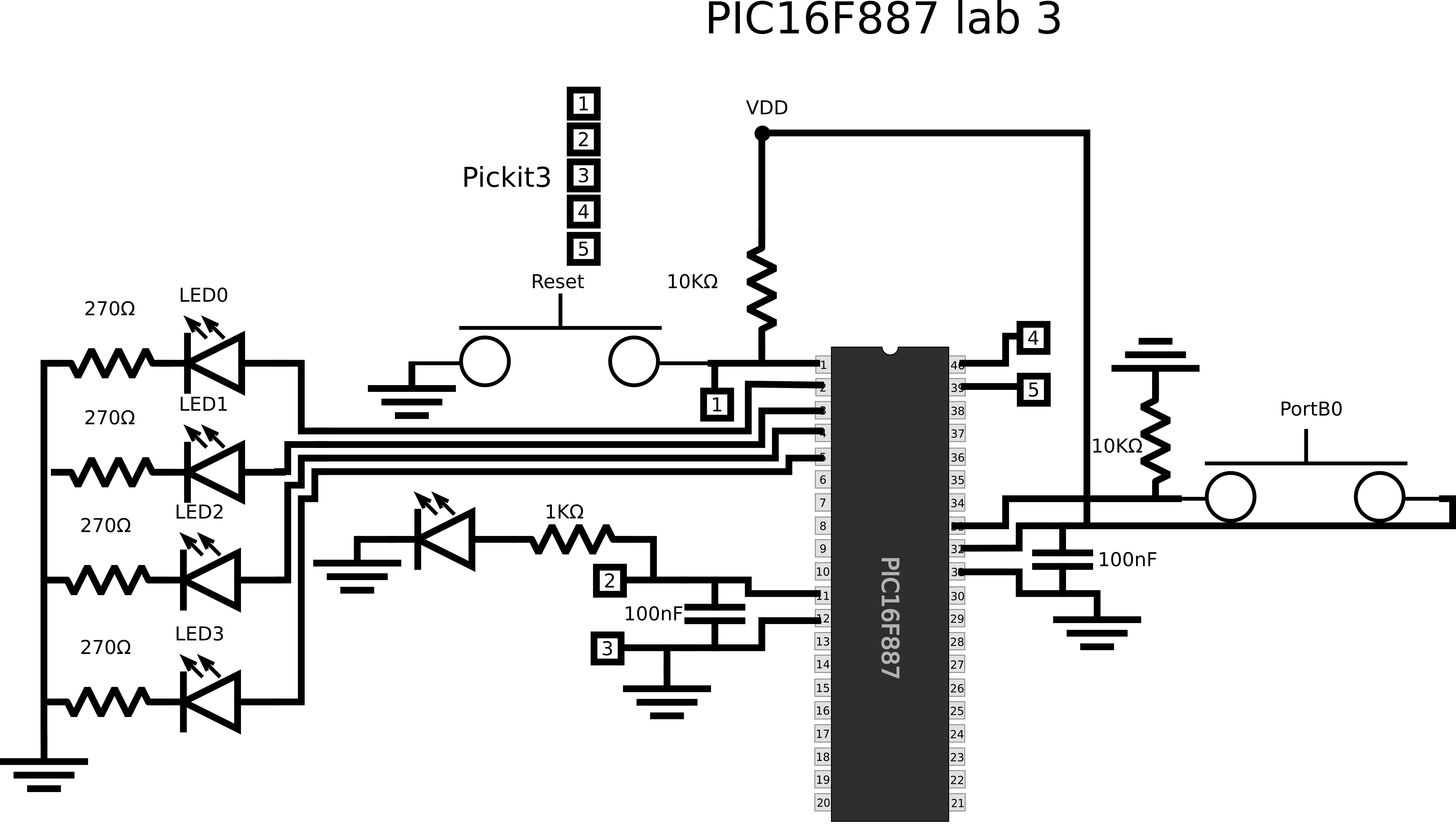
# Procedure

The procedure is provided in the lab instructions.

# Results and Discussion

Part A: Using subroutines

Note based on circuit from last lab:



3. Modify code to make 4-bit binary counter

Code with comments included with lab.

5. Attach a copy of the .lst file and answer the following related questions.

Copy is attached with lab report.

a) Is there any memory space assigned to the “include” directive? Why?

It doesn’t anything to the values on the side like location, value, object, code and line. So it seems logical to assume it does not take any memory space.

b) What is the meaning of the hex code at the left column, beside the variables definition in “cblock” directive?

It seems that it’s the addresses of the user defined variables in the file register. Given that they are 20 and 21 and I set the starting address of the code as 0x20.

c) What is the implication of the directive “ORG”

It sets the next line as location 00 and seems to increment in bytes used for every line of code because comments don’t cause it to increase. Also labels don’t seem to cause to increment further confirming this theory.

d) List the address of all the labels in your program.

|  |  |
| --- | --- |
| Location (address) | Label |
| 0000 | START |
| 0013 | Mainstart |
| 0018 | LEDoncycleouter |
| 001C | LEDoncycle |
| 0022 | LEDoffcycleouter |
| 0026 | LEDoffcycleinner |

e) What is the total size of your program code?

002B is the final address which in bytes would be 43 bytes of code in base 10.

Part B: Introduction to MPLAB Debugging features

6. Refer to the User’s Guide on how to start debugging your code.

You press the debug run icon and it will compile a debug version of your project which you can set breakpoints on and step through.

7. Explain the various debugger step through options.



From left to right:

The first one step over executes one line of code unless it’s a function call in which case it executes the entire thing and then stops at the line immediately after.

The second one is step into it also executes one line at a time and if it’s a function call it goes to the first line of the function instead of skipping the entire thing allowing you to debug functions if needed.

The third button is to run code and pause at whatever line your text editing cursor is at in the window. This is useful as it means you don’t need to set a breakpoint for one off checks to make sure something is working.

The final buttons sets the PC counter to whatever line you have selected using your cursor. This is useful if you want to debug a specific part of code but skip something like a real time delay before doing a specific check.

8. Suggest a debugging solution to the following scenarios.

a) You want to find out if the value of PORTA correctly updated.

Add a breakpoint right after the code that is intended to modify PORTA and create a watch for PORTA to make it easier to see if the value changed.

b) You want to find out about changes to PCL and PCLATH when a subroutine calls and returns.

Add PCL and PCLATH to watches, set a breakpoint right a subroutine is called and ends so you can step into it and step over the return part of the code and see how affects the registers.

c) You want to monitor the change of your counter variable as a delay loop.

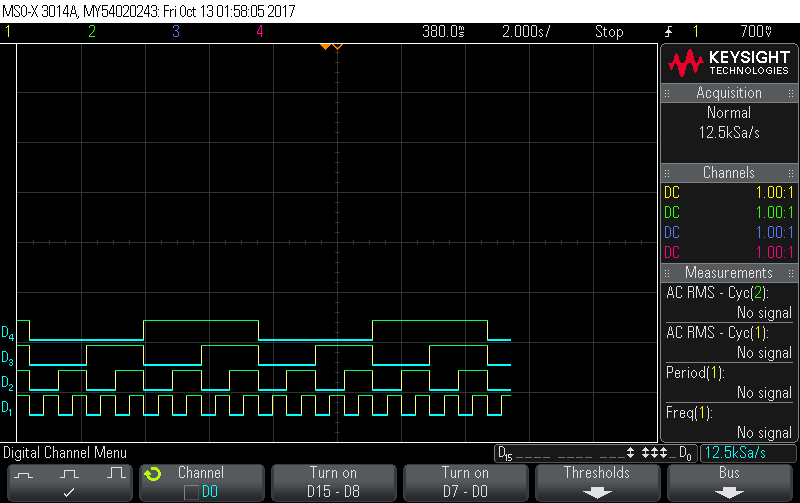
You can add your counter to a watch and set a breakpoint right before and after your delay loop. If it gets stuck between the start and then the loop does not end.

Logic analyzer VS oscilloscope

The oscilloscope is used to look at analog waveforms over time and see changes if the waveform. So if you have a waveform with an irregular waveform the oscilloscope would be useful.

The Logic analyzer is to look at digital logic systems over time. Unlike the oscilloscope it can only read a 0 or 1 from its inputs. While this may seem strictly worse, it has the advantage of having many more channels and having the user interface optimized for logic analysis making it easier to analyze what is going on in the circuit.

Screenshot of 4 bit synchronous counter from logic analyzer:



# Conclusion

In this lab I learn how to use the PIC16F887 debugger and made a 4 bit synchronous counter using subroutines. As with last lab my code has lots of possible improvements none of which I made while turning my delay part into a subroutine.

# References

|  |  |
| --- | --- |
| [1] | Microchip Technology Inc., "MPASM Assembler, MPLINK Object Linker, MPLIB Object Librarian User's Guide," Microchip Technology Inc., 2013. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/33014L.pdf. [Accessed 20 9 2017]. |