

Experiment No. 4

Introduction to MicroC Program

Simulation Lab – ENEE4104

Section: Saturday 2:00-5:00pm

Instructor: Ashraf Rimawi

Teaching Assistant: Dalal Hamdan

Student Name: Haitham Da’ana

Student ID: 1121331

**Date: 22/10/16**

**Abstract:**

The aim of this experiment was familiarize students with the MicroC Pro by simulating and testing out some basic, and straight-forward programs/codes, and run them on a PIC Microcontroller simulator using programs such as “Proteus”, without loading them on a real PIC Microcontroller.

**Introduction:**

**PIC Microcontrollers** are chips family that mainly consists of I/O ports, memory, processor, and converters like Analog-to-Digital Converters/Digital-to-Analog Converters and so on, and some other features, such as, Pulse Width Modulation (PMW) modules. These Microcontrollers can be programmed by higher level language like C, and as any processor, it has its own instruction set, thus, it requires a special compilers; e.g. mikroC PRO for PIC.

**microC PRO** is a software that is used to write programs for Microcontrollers, its official language is C, but this program has its own compiler that is specially made to compile the code to fit the processor of the PIC Microcontrollers since, they differ from the ordinary computer processors like Intel, and has their own instruction set.

**Procedure**

PART I:

1. microC Pro was opened, and a new project was created with the following specifications:
2. The device was chosen to be **P16F84A**
3. The clock was chosen to be 4MHz
4. The following code was written in microC PRO:

// Lcd pinout settings

sbit LCD\_RS at RB4\_bit;

sbit LCD\_EN at RB5\_bit;

 sbit LCD\_D7 at RB3\_bit;

 sbit LCD\_D6 at RB2\_bit;

 sbit LCD\_D5 at RB1\_bit;

 sbit LCD\_D4 at RB0\_bit;

 // Pin direction

sbit LCD\_RS\_Direction at TRISB4\_bit;

sbit LCD\_EN\_Direction at TRISB5\_bit;

sbit LCD\_D7\_Direction at TRISB3\_bit;

 sbit LCD\_D6\_Direction at TRISB2\_bit;

 sbit LCD\_D5\_Direction at TRISB1\_bit;

sbit LCD\_D4\_Direction at TRISB0\_bit;

void main()

{

TRISA = 0x00; // set all pins of port A as output

TRISB = 0x00; // set all pins of port B as output

while(1)

{ lcd\_init(); // initialize the lcd lcd\_out(1,1,"ENEE413 EXP#3"); Lcd\_Cmd(\_LCD\_CURSOR\_OFF);

}

}

1. After the code was written, it got built right away, in order to extract the hex file of the project!

PARTI – Simulation:

1. Proteus software was opened and the following circuit was drawn as shown in Figure below:



1. code was loaded on the PIC16F84A Module on Proteus software, by double clicking the model on the connection board, the hex file output of the mikroC program was loaded onto proteus, and clock was set to 4MHz as it was set in the original mikroC code.
2. The simulation was started, and snap shot of the output was taken.

PART II:

1. microC Pro was opened, and a new project was created with the following specifications:
2. The device was chosen to be **P16F877A**
3. The clock was chosen to be 8MHz
4. The following code was written in microC PRO:
5. After the code was written, it got built right away, in order to extract the hex file of the project!

PART II – Simulation:

1. Proteus software was opened and the following circuit was drawn:

unsigned int a;

void main() {

TRISA = 0xFF; // PORTA is input

TRISC = 0; // PORTC is output

TRISB = 0; // PORTB is output

do {

a = ADC\_Read(0); // Get 10-bit results of AD conversion PORTB = a; // Send lower 8 bits to PORTB

PORTC = a >> 8; // Send 2 most significant bits to RC1, RC0 } while(1);

}



1. The output of the microC code hex file was loaded onto the Proteus PIC16F877A model, with clock set to 8MHz
2. The circuit was simulated and the results of simulation were observed and snap shot of the output was taken

**Results**

1. **PART I:**



1. **PART II:**
2. With pot set to 100%:



All LED’s are on.

1. **Pot set to 0%:**

None of the LED’s is on



**Discussion**

1. We observed that in PART I, that microC program has its own libraries for various of electronic parts such as the LCD which has its own library and instructions(functions)

That helps and eases the programming operation.

1. In part two we observed that the Analog to Digital Converter is limited to 10 bits, which may result in errors if we want to make a system to be highly accurate, but in this case it’s not important
2. The code in part two consists of an infinite while loop which will never ever end since it’s ending option can’t be attained, and also the analog input is just a fraction of the bias/reference voltage, ranging from between

Vfull > Vfraction > 0

1. When the analog input is at maximum full range input the microC is programmed then to drive all the LED’s at its defined outputs, which these LED’s will represent the digital equivalent code of the analog input