**Introduction:**

Problem Statement:

The microsystem design course final project allowed for the choice of a user defined project. Using experience from previous tutorials and labs, a few options were discussed. After discussion, a choice was made to use a microphone input into the PIC887 analog input and an HD44780 LCD display to display a current and average sound level.

The primary objectives for this project were interfacing the microphone with the PIC887 microcontroller, and creating interface drivers for the HD44780 LCD display. Secondary objectives included soldering all connections for the microphone breakout board and LCD display, interfacing the microphone with the analog to digital converter, manipulating the microphone data allowing it to convey useful sound level values, and displaying sound level data via characters on the LCD display.

Background Information:

The HD44780 display requires an initialization sequence prior to using the LCD to display character values. The backlight on the LCD display requires a larger current source, therefore a 9 volt battery was used, with a voltage divider circuit to allow for correct voltage. Also, the contrast for the LCD display is controlled by voltage via the onboard potentiometer. The microphone breakout board utilizes a small electret microphone and a Texas Instruments OPA344 op amp to amplify the analog signal from the microphone.

**Methods and Procedures:**

Project Sequence:

The project sequence is quite extensive. The project was broken into two distinct work sections, so that two people were able to work primarily on different subsections. These subsections are: microphone interfacing and data manipulation, and LCD driver, interfacing, and data display. Later, the two functioning codes were combined and interfaced with eachother.

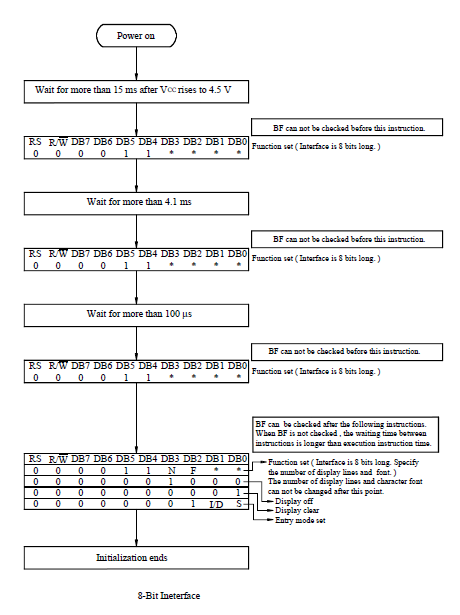
First it is important to create a template for the entire project. All of the user defined variables are laid out in the beginning, allowing the addition of more variables as needed. Utilizing templates and code examples from previous labs allowed for implementation of various functions modified to suit the needs of the final project. An interrupt service routine is used to handle the necessary interrupts, followed by an init subroutine to initialize necessary ports for input and outputs. Following are various subroutines used for the filter average, LCD functions, and character conversion.

The microphone project sequence is as follows. The microphone breakout board physical connections were soldered to the PIC887 microcontroller. The breakout board interfaces via the analog input port RA1 receiving an analog signal from the microphone driven opamp on the breakout board. This signal is fed into the analog to digital converter, allowing it to be converted to a 10-bit binary number. A new value is pulled from the A2D converter every time Timer0 overflows, it is then loaded into the current level and averaged with the average level. The values taken from the A2D converter are left aligned, making the stored values the eight most significant bits. This data is then manipulated and converted to two HEX values, one for each of the two digit characters, using a lookup table to send the correct character value to the LCD display.

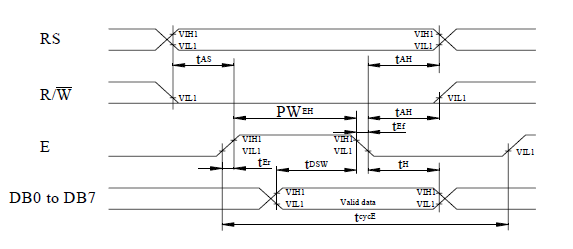
A table was developed to show possible values and equivalent character values for the LCD display. Below you will see a table describing possible decimal values, equivalent binary values, equivalent hexadecimal values, and converted hexadecimal character values for the LCD display.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Decimal** | **Binary** | **Equivalent Level Value** | **Equivalent level HEX Value** | **Equivalent LCD Value** |
| 0 | 0 | 0 | 0 | 0x30 |
| 1 | 1 | 1 | 1 | 0x31 |
| 2 | 10 | 2 | 2 | 0x32 |
| 3 | 11 | 3 | 3 | 0x33 |
| 4 | 100 | 4 | 4 | 0x34 |
| 5 | 101 | 5 | 5 | 0x35 |
| 6 | 110 | 6 | 6 | 0x36 |
| 7 | 111 | 7 | 7 | 0x37 |
| 8 | 1000 | 8 | 8 | 0x38 |
| 9 | 1001 | 9 | 9 | 0x39 |
| 10 | 1010 | 10 | A | 0x41 |
| 11 | 1011 | 11 | B | 0x42 |
| 12 | 1100 | 12 | C | 0x43 |
| 13 | 1101 | 13 | D | 0x44 |
| 14 | 1110 | 14 | E | 0x45 |
| 15 | 1111 | 15 | F | 0x46 |

The LCD project sequence is as follows. In order to get the LCD out of the box and interfaced into a system can be broken down into three parts. First, research confirming that the HD44780 protocol could be implemented physically to the PIC16F887; given the number of ports and by choosing the 5V model this was accomplished. Immediately the physical connections were made and the supporting circuits were constructed. Secondly, the timing constraints and initialization sequence needed to be followed. The use of both lines on the screen was selected as needed by the mission statement of the project. The only resource used in this step was the data sheet specified from the supplier. Upon completing the initialization sequence, the concept of sending data to the LCD and commands to the LCD was understood. At this point, drivers for the screen could be drafted in a modular fashion allowing for more convenient programming methods.



The above shows a flow chart for the initialization sequence for the HD44780 LCD display. It is important to note this flow chart is followed closely in the code for the initialization sequence.



Above shows the proper timing sequence for the HD 44780 LCD display. It is important to note that the enable bit is high prior to the data bits being sent. Also, note that the R/W is never set because the project only requires data to be written to the LCD display. RS determines whether a command or data is to be written.

|  |  |  |
| --- | --- | --- |
| Hex Value | Delay Time | units |
| 0xFF | 196.3 | ms |
| 0xFA | 192.4 | ms |
| 0xAA | 130.6 | ms |
| 0x77 | 91.2 | ms |
| 0x19 | 18.5 | ms |
| 0x17 | 17 | ms |
| 0x15 | 15.4 | ms |
| 0x14 | 14.6 | ms |
| 0x09 | 6.1 | ms |
| 0x05 | 3.01 | ms |
| 0x03 | 1.5 | ms |
| 0x02 | 777 | us |
| 0x01 | 4 | us |

Here you find a table for measure time delays for the function DelayW called significantly in the LCD\_Init function.

**Results and Discussion:**

