

# gPod Accessible Blood Glucose Meter

Week 7

February 27-March 3

David Price

## Work Completed

This week was spent working towards integration of the LCD screen and the speech module to the microprocessor. Starting with the LCD screen I used some sample code included with the compiler and other examples found on the internet. Initially the code did not work with the LCD screen. I later discovered that the code lacked the proper initialization routines for the LCD. The following code shows the initialization routine required for the LCD.

```
void lcd_init()
{
    TRISENABLE= OUTPUT; //set RA2 as output
    TRISDATA=OUTPUTS; //set RC0-RC7 as outputs
    TRISR5=OUTPUT; //set RB7 as output
    TRISRW=OUTPUT; //set RB6 as output

    RS = 0; // write control bytes
    DelayMs(15); // power on delay
    DATA = 0x3; // attention!
    DelayMs(5);
    DATA = 0x3; // attention!
    DelayUs(160);
    DATA = 0x3; // attention!
    DelayUs(160);
    DATA=0x38; // Interface length:8 bit mode, 2lines , 5x7 font
    DATA =0x08; // Display off, cursor off, blink off
    DATA=0x01; // clear display
    DATA = 0x0F; // display on, Cursor on, blink on
    ENABLE=1; // set enable clock high
    ENABLE=0; // set enable clock low
    DATA= 0x06; //Entry mode, cursor move right
    ENABLE=1;
    ENABLE=0;
}
```

After implementing this initialization routine the LCD showed a blank screen with a blinking cursor. The next step was to write a string to the LCD. The code to write a string to the LCD requires a write character function looped for the length of the string to write each character. The following code describes this process.

```

void LCDwritechar (char c)
{
    int cnum;
    cnum = c;
    DATA=cnum;
    RS=1;
    ENABLE=1;
    ENABLE=0;
}

void LCDwritestring (const char* s)
{
    while(*s)
        LCDwritechar(*s++);
}

```

Using this code to write the string “hello world” to the LCD did not work at first. After observing the scope output of the Enable and RS pins I learned that the RS pulse was not being held high through the enable pulse. The RS pulse is set high to tell the LCD to write characters to the screen and low to accept instructions. The RS pulse was originally setup on the RA0 pin but there is a problem with the digital output on those pins according to Microchip. My solution was to move the RS and RW outputs to the RB 7 and RB 6 pins. After doing this the LCD worked properly. I have some doubts that the second row of the display we found is working and will be ordering a newer screen to eliminate this problem. Figure 1 shows the LCD screen displaying the “hello world” phrase.



**Figure 1, LCD screen displaying “hello world” as commanded by the microprocessor.**

I also spent time trying to interface the speech module with the microprocessor. First I worked with the computer and LabView to understand the data format required to make the module work. LabView was configured for 38400 baud, 8 data bits, no parity, and 2 stop bits. I used a port monitor to observe the commands being sent by the SP03 software. Table 1 shows the command sequence required to make the module speak the word “Hello.”

Command Sent	Response Read	Comment
80	01	Start of command sequence
00	00	Volume (Full)
04	04	Speech Pitch (4)
02	02	Speech Speed (2)
48	H	Text string follows until 1A command received by the module. <b>Note:</b> The text buffer is 85 bytes in size so the text string is limited to 81 characters long.
65	e	
6C	l	
6C	l	
6F	o	
1A	1A	Ends text string
00	00	Speak the text

**Table 1, Command sequence required for speech module.**

I wrote a LabView program to simulate the command sequence needed to make the speech module work. The LabView program correctly communicates with the speech module using the computer's RS232 port. Work will be done after the break to communicate with the speech module using the microprocessor.

### **Future Work**

This week I will continue to work to integrate the speech module to the microprocessor and integrate the LCD screen with the A/D converter. I need to find out how to do RS232 communication with the microprocessor at 38400 baud and 2 stop bits. Currently I only have code to communicate using 1 stop bit.

I also will work to display the voltage value from the A/D on the LCD screen. In order to do this I need to take the 10 bit binary number from the A/D converter and convert it to a BCD (binary coded decimal) value. Once I have the BCD value I can add 0x30 to the number to convert it to the ASCII representation needed to display the voltage as a decimal number on the LCD screen. Once I can display the raw voltage level on the LCD I will add an algorithm to convert the voltage level to a glucose measurement.

### **Project Review**

The work on the LCD and speech module was extremely productive. I now have a full understanding of the LCD screen and how to communicate with the speech module. Matt and Mike spent time working with the barcode scanner to try and understand how to interface it to our meter.

### **Hours Worked**

Hours spent on the project for Week 7: **27 Hours**