6/10

ECE 341 Lab 6

Zachary DeLuca

March 06 2023

---

1 Introduction

Technically, the "cerebot" is a PIC32 board.

In this lab we were able to make a library that allows us to write characters or strings to the LCD display. We will use the peripheral libraries in the Cerebot files in order to use its PMP. The PMP is required to interface with the controller on the LCD board, as it accepts a set of parallel inputs. The other important task the PMP takes care of is the timing as the LCD controller clock is only 10kHZ whereas the cerebot runs its peripheral clock at 10MHz. The PMP initiates the handshake in order to keep the user from having to bit bang each of the instructions and data bits to and from the board. This lab is complete once the board displays the correct strings that are inputted in the main function.

Programmable addr lines, control lines, etc. -1

2 Implementation

2.1 Listing 6.1

The first listing shows the initialization of the LCD board that is to be done in the system init file. If this project is meant to create a library to be imported into any project that needs an LCD display part, then one thing needs to be addressed. The delay used here is a hardware assisted delay, meaning the intiialization of the Timer1 and the function that allows us to use it need to be incorporated with the library's h and c files instead of having

Listing 6.1

```
void LCDinit(){
int cfg1 = PMP_ON|PMP_READ_WRITE_EN|PMP_READ_POL_HI|PMP_WRITE_POL_HI;
int cfg2 = PMP_DATA_BUS_8 | PMP_MODE_MASTER1 |
PMP_WAIT_BEG_4 | PMP_WAIT_MID_15 | PMP_WAIT_END_4;
int cfg3 = PMP_PEN_0; // only PMA0 enabled
int cfg4 = PMP_INT_OFF; // no interrupts used
mPMPOpen(cfg1, cfg2, cfg3, cfg4);

Timer1_delay(50);
PMPSetAddress(0);
PMPMasterWrite(0x38);

Timer1_delay(50);
PMPSetAddress(0);
PMPMasterWrite(0x0F);

Timer1_delay(50);
PMPSetAddress(0);
PMPMasterWrite(0x01);
Timer1_delay(50);
}
```

This simply shows how to implement the initialization flow shown on page 11 of the lab handout.

2.2 Listing 6.2

The next listing is the functions for reading and writing to the LCD controller board. These are given in the lab handout but are shown here for the sake of showing the differences made to the base model of the code. The only difference made was an extra instance of waiting for the busy flag to clear in the write function. In retrospect this was probably not necessary, but the code has been proven to work so the code stays as it is in order to keep the LCD appeased. The only thing the extra delay would cause is that the LCD writes imperceptibly slower, which is not a concern for a device whose sole purpose is to be seen by human eyes. The busy() function is a user written function that simply contains the while(busyLCD()); loop.

Listing 6.2

```
void LCD_puts(char *char_string)
{
while(*char_string) // Look for end of string NULL character
{
LCD_putc(*char_string); // Write character to LCD
char_string++; // Increment string pointer
}
} //End of LCD_puts

char readLCD(int addr)
{
PMPSetAddress(addr); // Set LCD RS control
mPMPMasterReadByte(); // initiate dummy read sequence
return mPMPMasterReadByte();// read actual data
} // End of readLCD

void writeLCD(int addr, char c)
{
busy(); // Wait for LCD to be ready
PMPSetAddress(addr); // Set LCD RS control
PMPMasterWrite(c); // initiate write sequence
} // End of writeLCD
```

2.3 Listing 6.3

The main meat of the user defined function is the LCD_putc function, which is listed below in Listing 6.3. It takes in the input char and first checks if it is a control char, such as "\r" or "\n" and if it is then it executes the control function instead of printing the char. If it is not a control char, it then checks if it has reached the point that it needs to wrap around. This part cause problems but was able to be fixed by recalling the function after writing the new address function.

Listing 6.3

```
void LCD_putc(char intake){
int location;
location = readLCD(0);
if(location > 0xF && location < 0x40){
PMPSetAddress(0);
writeLCD(0,0xC0);
if(intake!=' ')
LCD_putc(intake);
}
else if (location > 0x4F){
writeLCD(0,0x80);
}
if(intake == '\r'){
if(location < 0xF){
writeLCD(0,0x80);
}
else{
writeLCD(0,0xC0);
}
}
else if(intake == '\n'){
if(location < 0xF){
writeLCD(0,0x80+location+0x40);
}
else{
writeLCD(0,0x80+location-0x40);
}
}
if(intake != '\r' && intake!='\n'){
writeLCD(1,intake);
}
Timer1_delay(5);
}
```

2.4 Listing 6.4

The next listing is simply a clear screen function thrown together for convenience.

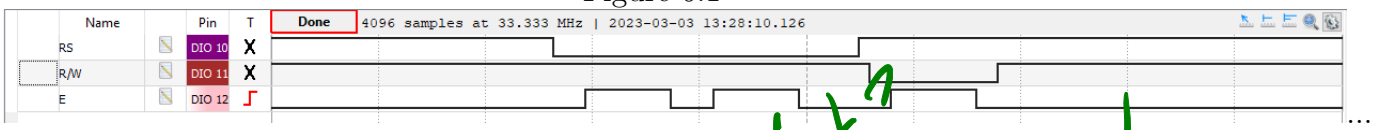
Listing 6.4

```
void clearLCD(){
writeLCD(0,0x01);
writeLCD(0,0x80);
}
```

3 Testing and Verification

For testing and verifying this lab, we needed to print the two statements given in the handout. The first being the question: "Does Dr J prefer PIC32 or FPGA??" and the second was an ASCII encoded phrase that read: "Answer: Neither!". For personal testing purposes, additional tests were done to check the functionality of the screen clearing, the control characters and the wrapping feature. The timing of the system was checked using the waveforms, and the result is printed below:

Figure 6.1



measure this? Does it scale!

4 Conclusion

This lab finished successfully with the LCD being able to display everything asked of it, including and exceeding the prompt phrases from the handout. The experience with the libraries indicate that using the PMP is much preferable to the bit-banging method as it does not require as much waiting and manual synchronizing of the differently clocked chips. This is helpful when interacting with the LCD controller. The interaction is with the LCD controller as it allows for writing the data to the display to be handled by something that can keep characters displayed without burdening the master chip. Having memory on the LCD instead of driving it straight from the Cerebot means that the data can be written once and let stand instead of having to constantly try to drive data on a differently clocked peripheral. With this setup, the data can be displayed as fast as once every clock(probably), meaning 10000 times a second, provided its being read from the LCD memory. The data being written to the controller is slower. Either way, the display is meant for human eyes which are much slower than that so the difference the busy periods make up is unnoticeable.

what?!.. -2 what rate did you actually measure?