Inputs, Outputs, and Time

EE 474 Introduction to Embedded Systems
Lab 2

Daniel Park
Samuel Johnson
Reilly Mulligan

Introduction

The purpose of this lab was to use the Beaglebone to interact with the LCD display and to observe the effect of the Beaglebone's scheduler. In order to achieve our first goal, we wrote code to initialize the LCD, code that writes user input to pipes, and code that pushes what it reads from the pipes to the LCD. In order to achieve the latter, we wrote code that simply turns a GPIO on and off to observe lag caused by the scheduler.

Our code goes above and beyond the specifications by including a program that displays an animation on the LCD and by allowing the user to input commands instead of just text.

Hardware Connections

In order to interface with the LCD, the GPIO and the voltage pings on the Beaglebone were connected to the pins on the LCD. Specifically, there are 8 data lines and 3 command pins on the LCD that need to be connected to GPIO pins for control, and 5 pins that provide power, contrast and backlight that need to be connected to specific voltage pins.

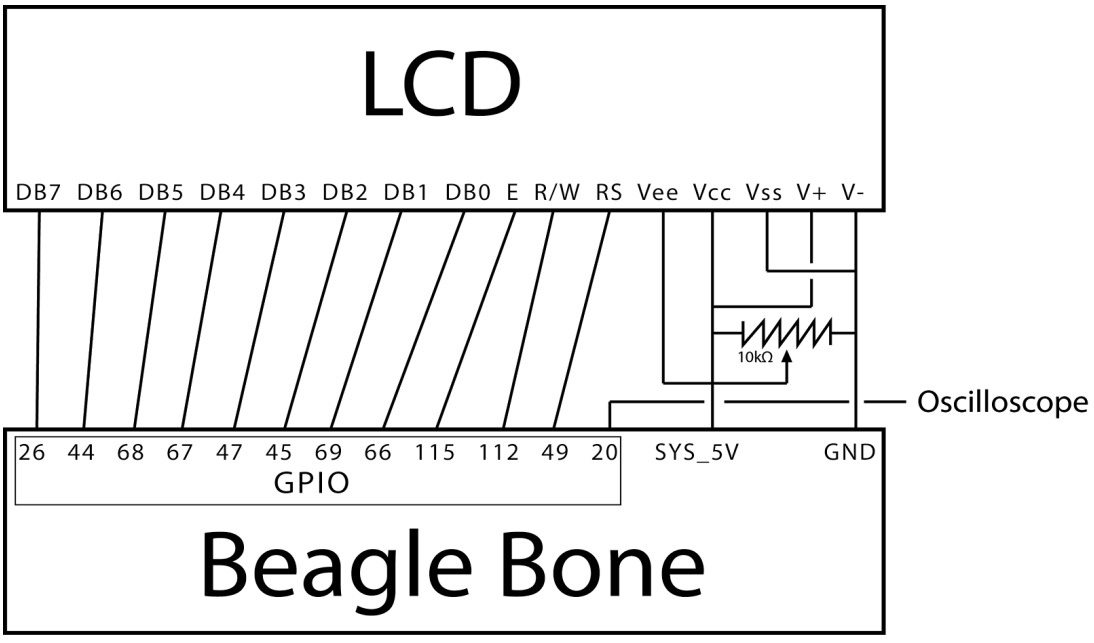

Figure 1: Circuit Wiring Overview

As shown in Figure 1 above, the GPIO pins drive the data lines on the LCD as well as the command lines. Power ports from the Beaglebone drive the power for the LCD as well as the backlight, and a potentiometer is used to control the contrast of the LCD display.

Software Overview

Our software consists of several different main programs that interact with the LCD. The following specific sections will discuss the purpose of each file in the program:

bbcommon.c

bbcommon.c is a collection of functions that handle the majority of lower level tasks on the Beaglebone, including activating the GPIOs and controlling their values and directions. Additionally, bbcommon.c contains the functions necessary to initialize the LCD and write commands and characters to it. All of our programs use bbcommon.c as a client. The most important functions of bbcomoon are in Table 1 below. Note that there are unlisted lower level functions that specifically deal with GPIO functionality.

Table 1: List of Important Functions in bbcommon.c

Function Call	**Use**
activateGPIO(int gnum)	activate specific GPIO
initializeLCD()	runs LCD initialization sequence
writeCommand(int config)	writes given command to the LCD
writeChar(char config)	writes given character to the LCD
writeString(char * str)	writes given string to the LCD

All of the commands in the table above interface directly with the LCD through GPIO pins, and control the behaviour of the display.

bbcommon.initializeLCD()

In order to use the LCD, we had to follow an initialization process laid out in Figure 2 shown below.

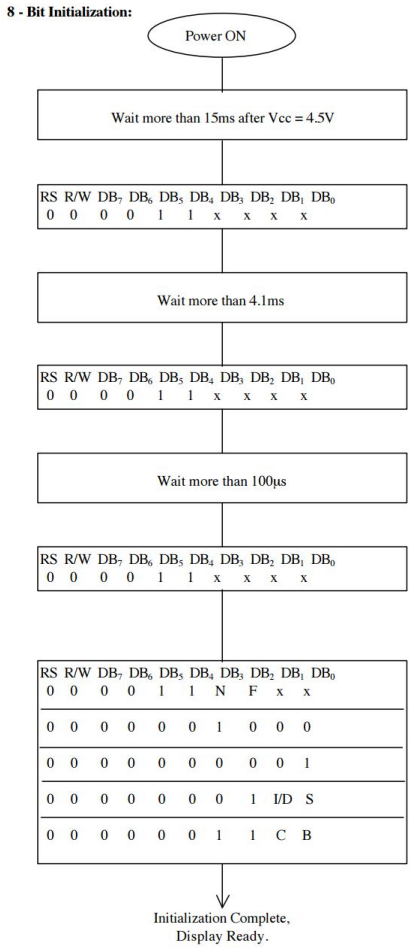

Figure 2: Initialization Procedure for 8-bit input LCD display

This procedure involved sending bit values from the GPIO pins into the LCD data pins as shown in Figure 2. In order to ease this process, functions that are listed in Table 1 were called to configure the data bits for each step.

lcdlistener.exe

lcdlistener.exe is an executable that creates 2 pipes (text_pipe and command_pipe) and continually checks if anything has been written to either of them. It does this by running in a loop and using select() to see if either of the pipes can be read from. If one of the pipes can be read from, lcdlistener will push its contents to the appropriate register on the LCD (the data register for text_pipe and the command register for command_pipe). lcdlistener runs indefinitely unless a user ends it. See Figure 3 for the complete setup.

input.exe

input.exe is a program that makes use of the pipes created to write information to the LCD. Specifically, this program allows the user to directly control the LCD in real time using the command line on the host computer. In addition to the ability to write specific characters to the LCD, the user is given the ability to move the cursor, clear the display and change lines through typing commands that can be read by the program. A list of possible commands are shown in Table 2.

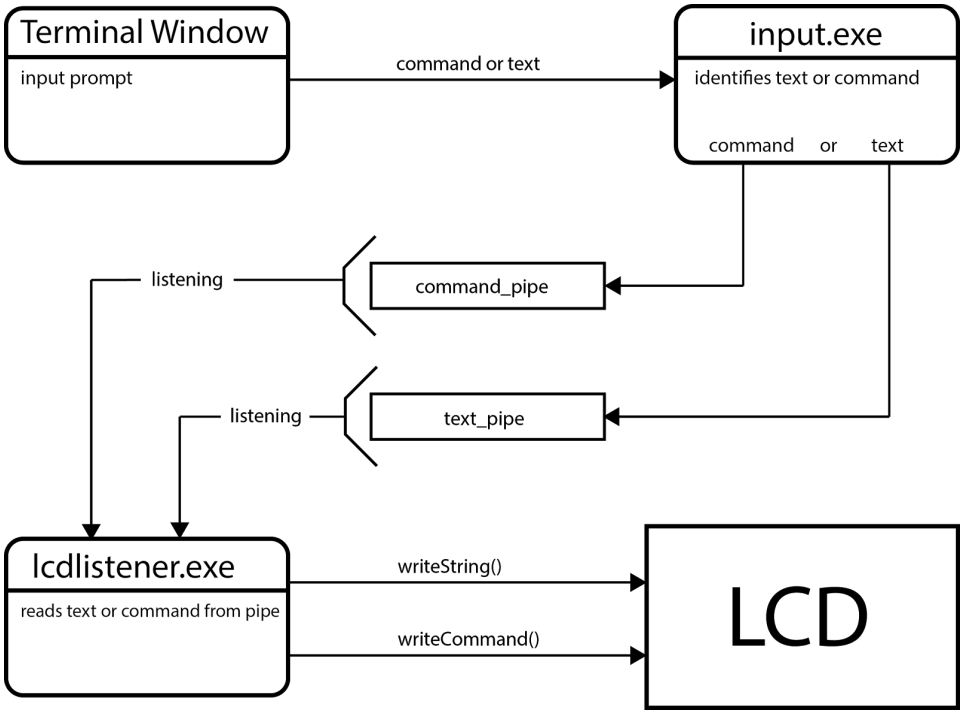

Figure 3: Block Schematic and Command Flow Chart of the LCD Program

Table 2: User commands for input.exe

Input	***Command***
/clear	Clears the LCD screen
/b	Backspace
/tl	Moves cursor to top line
/bl	Moves cursor to bottom line
/scl	Shifts cursor left
/scr	Shifts cursor right
/sdl	Shifts display left
/sdr	Shifts display right

getmoney.exe

getmoney.exe is an executable that produces moving text on the LCD. A '$' character scrolls along the top line, resetting when it reaches the end of the display. The string "GET MONEY" is always displayed in the bottom left corner of the screen. This was accomplished by writing the characters in the appropriate part of the screen, delaying, clearing the buffer, and writing characters to the string in a new position. This makes the text appear to be moving. Placement of the '$' was determined by the various loops in the code.

Scheduler and Jitter of LCD

Observing the Beaglebone scheduler's workings and its limitations requires looking at voltage switching behavior from one of the GPIO pins (GPIO 20 in our lab experiment) using the oscilloscope. With the switching program running concurrently with others, the jitter arises because the scheduler is slicing the time into fragments and assigning it to the multiple programs in the Beaglebone, including the switching program itself. Figure 4 shows the voltage switching operation without the jitter while Figure 5 contains jitter during the program. This jitter is evident by the observed late switching response of the voltages through the scope.

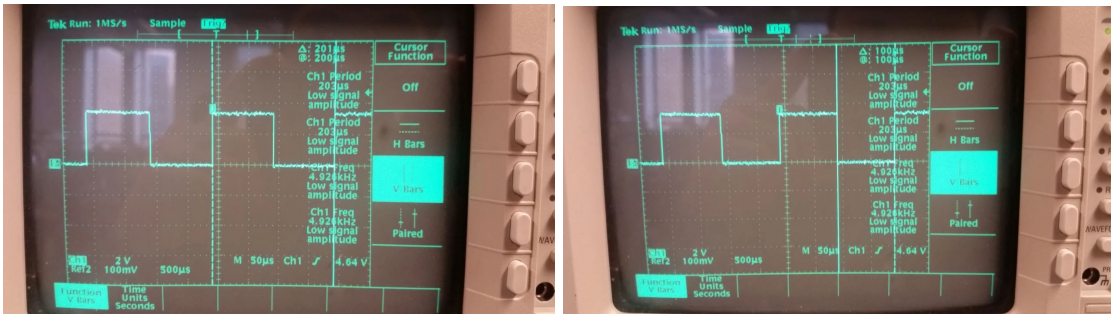

Figure 4: Voltage from GPIO20 pin without jitter

Quantitatively, the RMS voltage value fluctuates with every other period because there are times when the low values switch to the high value a bit too late, decreasing the duty cycle and lowering the RMS value. Likewise, when the high value switch to the low value a bit too late, the duty cycle increases along with the RMS value. Specifically, the program has a 200 microsecond period with 50% duty cycle. During jitter, the period fluctuates between 250 to 275 microseconds and the duty cycle fluctuates between 40 to 75 percent.

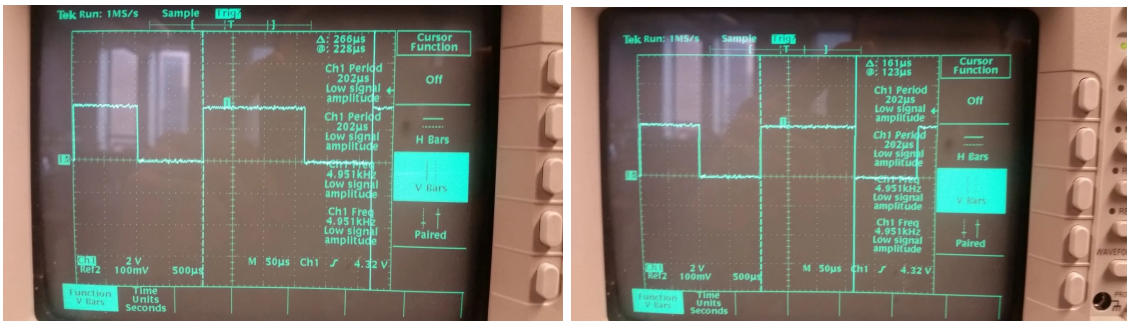

Figure 5: Voltage from GPIO20 pin with scheduling jitter

Conclusion

Through this project our group learned many different aspects of writing code that interfaces with other processes and external modules. Some of the most challenging parts of creating the working code and hardware had to deal with specific timings that needed to be met. For example, the boot sequence of the LCD required codes in a specific order with intentional timing. Once initialized, the commands and text written to the LCD also required specific timing.

There was also the challenge of constructing a working named pipe for the user-side program to communicate with the LCD. This is a very convenient way to transfer data between independently running programs, but has a limitation on its speed. We discovered this when trying to run “GetMoney.exe” through the pipe. The animation wouldn’t run because of the speed limitation on data transfer through the pipe. Also, having one program listen to 2 pipes at the same time proved challenging and took time to set up, but ultimately our solution worked very well.

We were able to overcome these timing challenges and create two working programs: GetMoney.exe and input.exe. In particular, input.exe turns the LCD into a very primitive text display that can be written to, while GetMoney shows the possibility of animation on the display. These both demonstrate how the LCD can be manipulated through software in a fun and useful way.