Lab #3 Low-level Character IO

1. Description

In this laboratory, the sub-routines: putchar_lcd() and getkey() are implemented. The purpose of these sub-routines is to learn about low-level IO programming which involves serial communication.

The sub-routine putchar_lcd() establishes serial communication with the LCD screen, and it takes an integer representing ASCII characters, which will be printed to the LCD. It is capable of handling special characters to perform tasks such as clearing the screen, moving the cursor back, inserting new line, etc. The sub-routine returns the integer passed within the range of 0 to 255. Anything beyond that range will return EOF as its output.

The sub-routine getkey() scans the keypad for the location of the key press, and returns its corresponding value. It does so by setting all the columns to high-Z, then setting the scanned column to low bit value. Each time the column is set to low bit, the routine checks if one of the rows outputs low bit as well. If so, we obtain the location of the specific key, which we can return the appropriate value after the key has been depressed.

The main() function tests these two sub-routines in two stages. Firstly, it calls each functions directly. Secondly, it calls printf_lcd() and fgets_keypad() which calls the sub-routines implemented in this assignment.

The limitation of this program is the length limit of 40 characters for the string inputted from fgets_keypad().

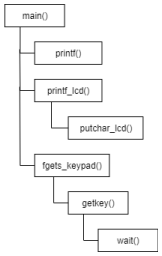

Figure 1: Function Hierarchy

2. Testing

Running main.c should include all the tests necessary to validate the two sub-routines.

1. Run main.c
2. The screen should be cleared and the screen should print the ASCII char of the integer passed inside the putchar_lcd subroutine.
3. Repeat step 1 and 2 with different values of integer passed within 0 to 255
4. Next, the there should be "EOF: -1" printed in the console since putchar_lcd() is given values outside 0 - 255 in the main routine.
5. We are now testing getkey(). Press one key in the keyboard. Check if the console prints exactly the key pressed. The LCD should only print the key pressed after the key is depressed. Check this feature by holding the press for a long time before depressing to observe if anything is printed to the screen.
6. The LCD is going to be cleared and it is going to prompt for a collection of keys. Input the keys desired.
7. Upon pressing enter, the LCD should print: "keys pressed: ¡keys entered¿". Make sure that the length of the keys entered do not exceed 40 characters.
8. The same behavior is going to be repeated to check if the sub-routines function robustly.
9. Rerun the whole program while changing the parameters passed into putchar_lcd to test different behaviors. One recommendation is to pass special characters like '\f' to putchar_lcd, which should clear the whole screen.

3. Results

The sub-routine putchar_lcd() behaves as expected. When an integer value between 0 to 255 is passed to the function, various ASCII character is printed to the screen. When special characters like \f, \n, \b, \v are inputted, it performs the instruction that corresponds to them. When a value outside 0 - 255 is passed, the console will print -1, which corresponds to EOF.

The sub-routine getkey() behaves as expected. Using printf(), the console will print exactly what has been pressed to the screen. When the key is pressed and held for a long time, only when the key is depressed that the console will print the value that the user just entered.

Using printf_lcd() will print whatever has been passed to the function, which validates the specification of putchar_lcd(). Using fgets_keypad() will return the string that is inputted from the keypad, which validates the specification of getkey(). In addition, pressing the delete key will allow the user to modify the string inputted. Once ENT is pressed, printf_lcd() will print exactly what was inputted, which validates both the sub-routines.

Given more time, we can even extend the getkey() to be compatible with other types of keypads with more characters by extending the reference table of the possible characters in the keypad. We can also extend putchar_lcd() to be compatible with other LCDs by properly establishing UART communication with other LCDs.