RILEY POOLE
ME 477 – LAB REPORT 03
WINTER 2019

I. DESCRIPTION

The primary purpose of this program was to test the functionality of my getchar() and putchar_lcd() functions. These functions are the lowest level functions we will be writing in this course and complete the goals of the first three labs to introduce us to increasingly low-level routines.

THE MAJOR TASKS PERFORMED BY THIS PROGRAM WERE

- Test the functionality of putchar_lcd() directly by passing letters and escape sequences directly
- Test the out of range function of putchar_lcd()
- Test the functionality of getkey() by returning a character from the keypad
- Collect an entire string using fgets_keypad()
- Write an entire string using printf_lcd()

THE LIMITATIONS OF THIS PROGRAMS CAPABILITY ARE

- Putchar_lcd() is limited to only 4 escape sequences f,b,v,n
- Putchar_lcd() is limited to the not extended ascii (<256)
- Using the Uart_Write() function requires an array argument that could be substituted with a single variable which would be faster to initialize
- The getkey() algorithm must check each column separately which takes a lot of time compared to a parallel connection to the keypad
- The wait() function is not accurate and is implementation dependent

EXPLAIN ANY ALGORITHMS

The getkey() algorithm uses the high and low impedance characteristics of the MyRio I/O to check each row and column for key presses. This is accomplished by first setting all except for one of the columns to a high impedance read mode. Then systematically each row is read to detect a low signal. All the rows and columns are connected to pull-up resistors and if a low signal is detected this must mean that one of the keys have been depressed. The channel of the row and column are then used with a table to return the pressed key.

HIERARCHICAL STRUCTURE
Implementation ready functions are marked with an ‘x’

main
- x MyRio_IsNotSuccess
- x printf_lcd
  - x va_start
  - x vsnprintf
  - putchar_lcd
  - x va_end
- putchar_lcd
  - x Uart_Open
  - x Uart_Write
  - x Dio_ReadBit
- getkey
  - **x** Dio_ReadBit
- x MyRio_Close

putchar_lcd
- **x** Uart_Open
- x Uart_Write
- x Dio_ReadBit

getkey
- **x** Dio_ReadBit

FUNCTIONS OF THE PROGRAM

- **main()**
  - Prototype:
    - int main(void)
  - Purpose:
    - to test the functionality of getkey() and putchar_lcd()
  - Rationale for creation:
    - Need a procedural section to layout testing environment
  - Inputs and parameters:
    - No inputs or parameters used
  - Return:
    - int – status of the MyRio_Open() or MyRio_Close
    - outputs of getkey() and putchar_lcd() to the console and the LCD (see pseudocode)
- **putchar_lcd()**
  - Prototype:
    - int putchar_lcd(int input)
  - Purpose:
    - put a single character on the lcd screen and check for errors
  - Rationale for creation:
    - need a low level routine to allow other output functions such as printf_lcd()
  - Inputs and parameters:
    - int input – character to be written to the lcd;
  - Return:
    - int – the character written to the lcd
    - int – EOF to signal an error
- **getkey()**
  - Prototype:
    - char getkey(void)
  - Purpose:
    - Bring in a single character from the keypad and return it to the caller
  - Rationale for creation:
    - Need a low level routine to allow other input functions such as fgets_keypad()
  - Inputs and parameters:
    - None;
  - Return:
    - character input on the keypad

II. TESTING

To make testing easier, I have written main to show all aspects of the basic functionality of putchar_lcd() and getkey() where the user just needs to press enter with a few inputs.

Testing putchar_lcd() is done by essentially passing the possible legal values into it as well as ones that are outside the range. We then check that the correct character is displayed as well as an error message for the ones out of range. Getkey() is tested by calling it for an input key and then returning that key to the PC console.

TESTING PUTCHAR_LCD()

Just press <ENT> 8 times and check the output matches

1. \f 'A'
2. \b
3. \n
4. 'A'
5. \v
6. \f 'A'
7. '256'

Output:

1.	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**
A_	**A**	**A**	**A**	**A**	**_**	**A_**	**Error**
		_	**A_**	**A**			**Console**

TESTING GETKEY()
Steps:
1. <ENT>
2. <any key>
3. Should print key in step 2 to the lcd

TESTING FGETS_KEYPAD()
Steps:
1. enter <any combination of keys 80 characters or less>
2. Should print the string to the lcd

TESTING PRINTF_LCD() WITH ESCAPE SEQUENCES

Just press <ENT> until the lcd reads “END”

1. \f
2. \v
3. \n
4. \b

Output:
1. **"This is a test of f"**
2. **"123s is a test**
   of v" ; places 123 at the start of This is a test of v
3. **"This is a test**
   of n" ; place "of n" on a new line
4. **"This is a tes**
   of b" ; places destructive backspace over t

III. RESULTS

HOW SUCCESSFULLY THE PROGRAM RUNS AND UNSOLVED PROBLEMS

The program runs without any obvious errors. The sequence of <ENT> inputs for the tester could be more fluid. It also may be difficult to see what the program is testing without having read the rest of this document. I don't like how fgets_keypad() returns a " " instead of a clear indication of EOF to the PC console which I think makes it ambiguous.

SPECIFIC QUESTIONS IN THE ASSIGNMENT

There were no specific questions asked in this lab.

POSSIBLE IMPROVEMENTS AND EXTENSIONS

- Allow putchar to use more escape sequences
- Allow putchar and lcd to use extended ascii or unicode
- Check the keys faster with assembly code or use parallel interface
- Make a better wait() function

IV. ALGORITHMS AND PSEUDOCODE

```
int putchar_lcd(int input)
{
      set an initialization flag;
      check the flag to see if putchar_lcd has been called before

      if it has not been called before
            initialize the UART
            set the initialization flag true;
            write characters to the lcd checking for escape sequences

      if the value of "input" is within 0 to 255{
            check if "input" is an escape sequence

      if it is an escape sequence
            save "input" to the one character array
            write the one character array to the UART

      else if the value of "input" is out of range
            return the error code EOF
}

int getkey()
{
      set an initialization flag

      if getkey has not been called before
            initialize the 8 channels
            set the initialization flag high
            for each column
                  set all the the columns to read high-Z
                  write the current column to low
                  for each row
                        set the current row to read high-Z
                        if the current row is low
                              set the low_bit_flag high
                              break
                  wait to make sure it is not erroneous;
                  if the low_bit_flag is high
                        break
      while the button is still being held (current row is low)
            wait to make sure it is released
            return the key based on the table defined
}
```

```
int main(void)
{
      testing putchar_lcd()
            print to lcd \f
            put A to lcd
            print to lcd \b
            print to lcd \n
            print to lcd A
            print to lcd \v
            print to lcd \f again
            print to lcd A

      check that a value outside the range 0-255 cannot be used
            print to lcd the value 256
            print to lcd the value -1

      testing getkey()
            call getkey() to get a character
            print this character to the lcd

      collect an entire string using printf_lcd()
            call fgets_keypad to collect a string
            print the string to lcd

      print an entire string using printf_lcd()
            print \f in a string
            print \v in a string
            print \n in a string
            print \b in a string
}
```