RILEY POOLE
ME 477 – LAB REPORT 02
WINTER 2019

I. DESCRIPTION

This program prompts the user for input strings on the LCD display. They are then stored and sent to be displayed on the PC console. This source file is mostly just to test the functionality of the getchar_keypad() function.

The function getchar_keypad() is essentially a branch of three “if” statements that change a buffer pointer and a while loop that checks for the user pressing the enter key.

THE MAJOR TASKS PERFORMED BY THIS PROGRAM WERE

1. Prompt the user for two input strings and wait for terminating enter key press
2. Send the input strings to the PC console display

THE LIMITATIONS OF THIS PROGRAMS CAPABILITY ARE

1. There is no easy way to clear the display with a single keypress
2. Up and Down functionality is not exploited rather it simply returns ‘[‘ or ‘]’
3. Key presses can only be displayed on the LCD screen not the PC console at the same time

EXPLAIN ANY ALGORITHMS

The getchar_keypad() function uses an algorithm to determine if the input character is an escape key. This is accomplished with a switch statement and checking for the allowed escaped sequences. If an escape sequence is not entered, then it puts the value of the input character into the buffer unaltered.

HIERARCHICAL STRUCTURE

The overall structure of the program is to break up the calling and outputting tasks into separate functions and then call the two procedurally in main().

Functions Previously found to be implementation ready are marked with an x_

- main
  - x_MyRio_Open
  - x_MyRio_IsNotSuccess
  - x_Printf_lcd
    - x_putchar_lcd
  - fgets_keypad
    - getchar_keypad
  - x_Printf
  - x_MyRio_Close
  - x_wait_ent
- getchar_keypad
  - x_getkey
  - x_putchar_lcd

FUNCTIONS OF THE PROGRAM

- **MAIN()**
  - Prototype:
    - int main(void)
  - Purpose:
    - To use the function getchar_keypad() and test its operation.
  - Rationale for creation:
    - Need a procedural section to layout testing environment.
  - Inputs and parameters:
    - No inputs or parameters.
  - Return:
    - int – status of the MyRio_Open() or MyRio_Close
  - Calls:
    - MyRio_Open()
    - MyRio_IsNotSuccess()
    - printf_lcd()
    - fgets_keypad()
    - wait_ent()
    - printf()
    - MyRio_Close()

- **GETCHAR_KEYPAD()**
  - Prototype:
    - int getchar_keypad(void)
  - Purpose:
    - Provide minimal user interface for input/output on the UART keypad. Keypresses are stored in a buffer and displayed on the LCD.
  - Rationale for creation:
    - Need a higher level abstraction of getkey() and putchar_lcd()
    - Need minimal user interface that accepts and returns keypresses
  - Inputs and parameters:
    - None
  - Return:
    - If there are no characters in buffer when called
      - No return, control only
      - Creates buffer of inputted characters
    - If there is more than one character in the buffer when called
      - int – returns the character pointed to in the buffer
    - If there is exactly one character in the buffer when called
      - int – returns EOF because that must be the last character of the buffer always.
  - Calls:
    - getkey()
    - putchar_lcd()

- **WAIT_ENT()**
  - Prototype:
    - void wait_ent(void)
  - Purpose:
    - Pause the program and wait for user to press <ENT>
  - Rationale for creation:
    - Cumbersome to write these two lines so many times
  - Inputs and parameters:
    - None
  - Return:
    - None
  - Calls:
    - printf_lcd()
    - getkey()

II. TESTING

It can be seen from the call tree hierarchy that the purpose of this program is to test the functionality of getchar_keypad() through the use of fgets_keypad(). Testing should include at least:

1. Pressing enter makes getchar_keypad() return the buffer back to the calling program one character at a time
   a. **Set #define test_getchar to 0; compile and run program** This will call fgets_keypad() which recursively calls getchar_keypad()
   b. Type <1><2><3><4><5><6><7><8><9><0><up><down><.><-> on keypad and press enter
   c. Type <1><2><3> then press <backspace> and hit enter
   d. Check that the return on the PC console is "1234567890[].- 12"
2. Check that getchar_keypad() returns EOF when there is a single character in the buffer
   a. Set #define test_getchar to 1; compile and run the program
   b. Type <1><ENT>
      i. check that "1 ÿ" is displayed on the PC console
   c. Type <ENT><ENT>
      i. check that "ÿ ÿ" is displayed on the PC console

Note: ÿ is the EOF character (255 base 10)

III. RESULTS

HOW SUCCESSFULLY THE PROGRAM RUNS AND UNSOLVED PROBLEMS

The program runs as intended. It is able to take in two strings with fgets_keypad(). The user created function getchar_keypad() works very similarly to the one used in the standard C library. This is shown by running a different section of main that runs getchar_keypad() directly. The delete function works as a destructive backspace. It is designed to not do anything if pressed as the first character (i.e. n = 0).

SPECIFIC QUESTIONS IN THE ASSIGNMENT

I did not see specific questions to be answered in the lab.

POSSIBLE IMPROVEMENTS AND EXTENSIONS

Something that bothers me about getchar_keypad() is that the statically declared buffer does not clear itself after it has been totally used. It seems more natural to be able to call the buffer only once.

Something that bothers me about fgets_keypad() is that it does not give a clear indication when the EOF character has been received. For example if <ENT> is pressed before characters are entered it would make sense for fgets_keypad to return EOF instead of pushing a blank space to the console.