Project Report: Computer Science/Computer Engineering

By: David Boucard, Eileen Li, Phoenetia Browne

HP 20b Business Consultant

- Simple Calculator
- Used for:
  - Business
  - Finance
  - Real Estate
  - Accounting

Our HP 20b User Guide

- Press any number key to display
- Maximum digits to display at once is 9
- +/- changes sign of input
- Press and operation key to stop
  - +, -, x, /, input

Platform

- Basic scientific and statistical functions
- JTAG header
- Processor
- LCD
- Keyboard

gure 1: The front and back of the HP 20b calculator.

LCD Display

- lcd.c
- lcd_put_char7
- lcd_init

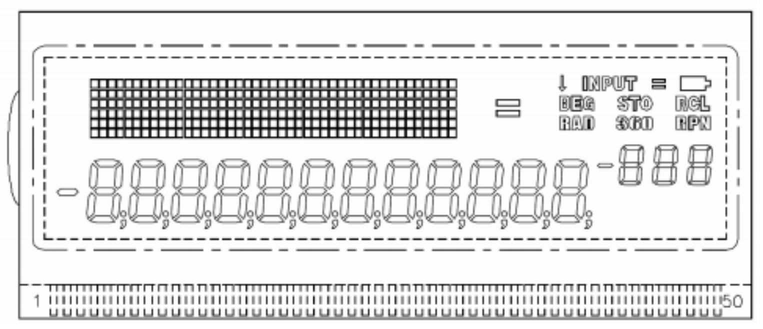


Figure 3: The LIBHP20bhp LCD Screen. It has a 43 x 6 pixel display matrix, various display indicators

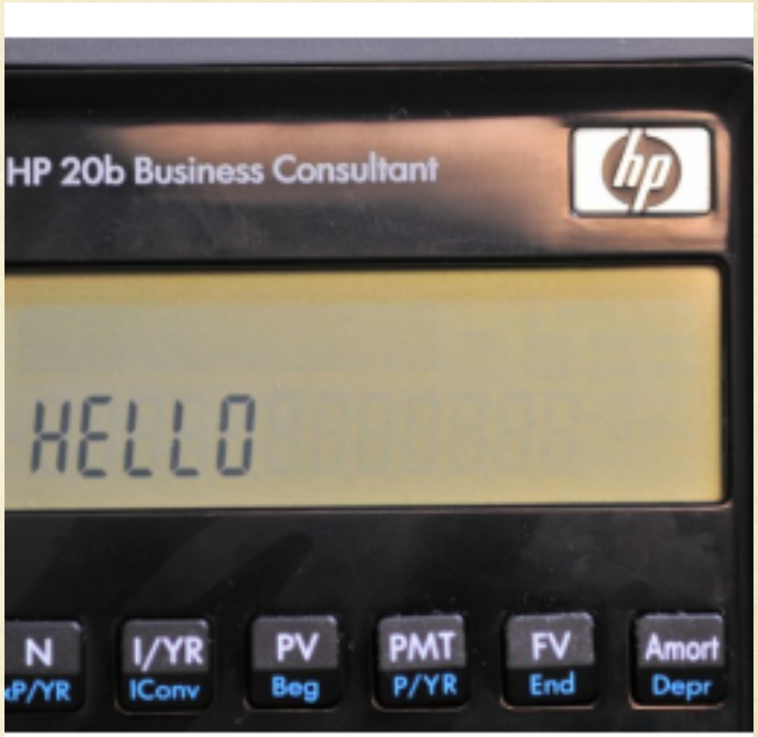


Figure 4: Running the "Hello" program

Keyboard

- *keyboard_column_high*
- *keyboard_column_low*
- *keyboard_row_read*

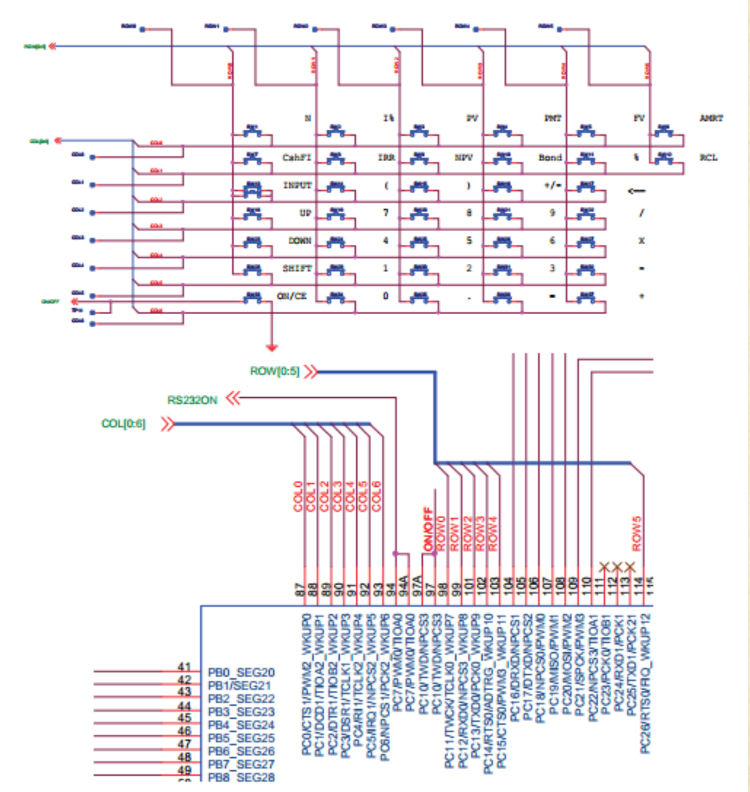


Figure 3: Schematics for the HP 20b keyboard. The top is the keyboard matrix itself; the bottom shows how the matrix is connected to the SAM7L chip. Note that the ON/CE key is separate (not part of the matrix) and that "columns" of keys are actually horizontal.

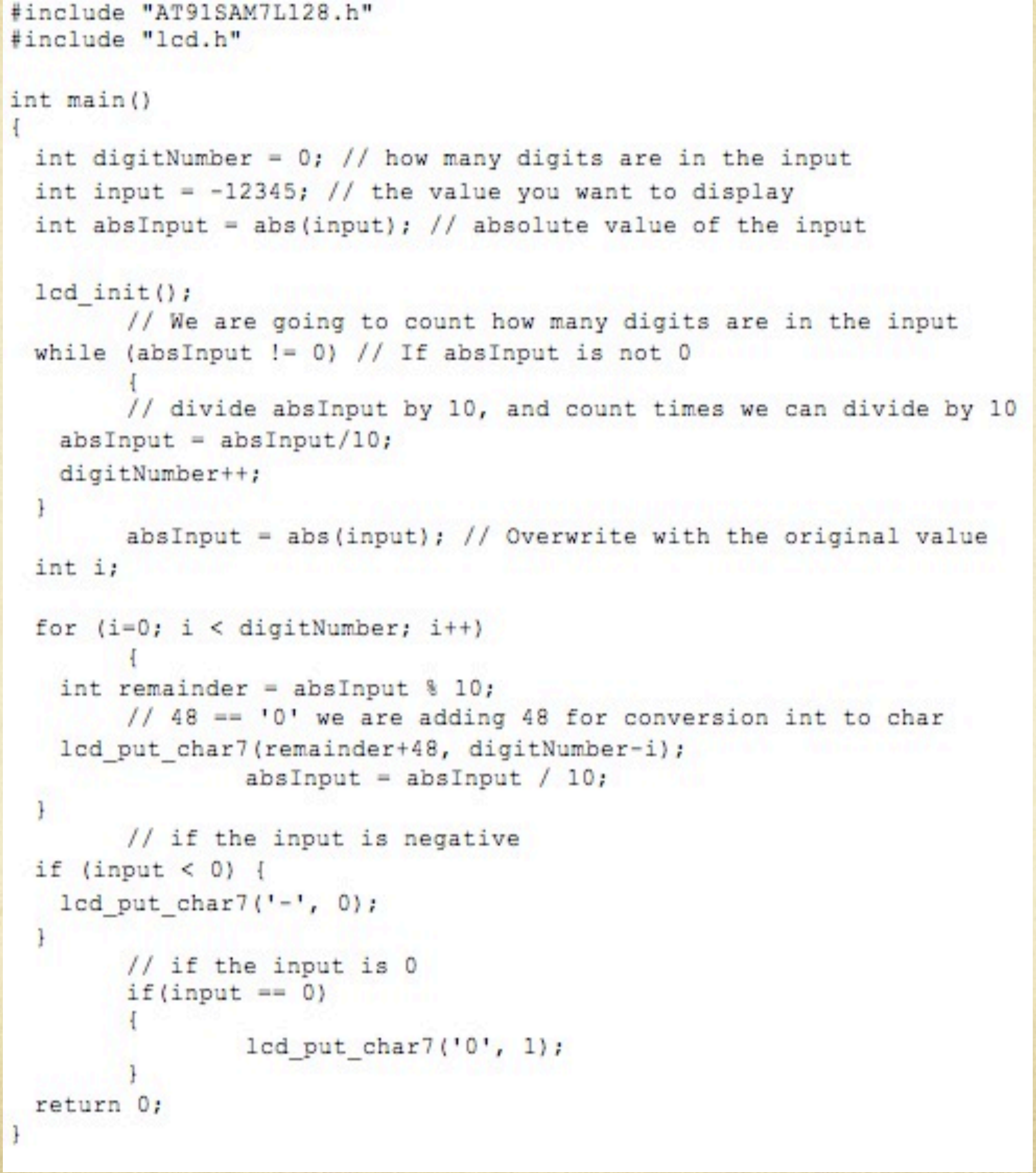
Software Details

- The developed software works towards :

  - Display Functionality
  - Scanning Functionality
  - Input Functionality

Display Functionality

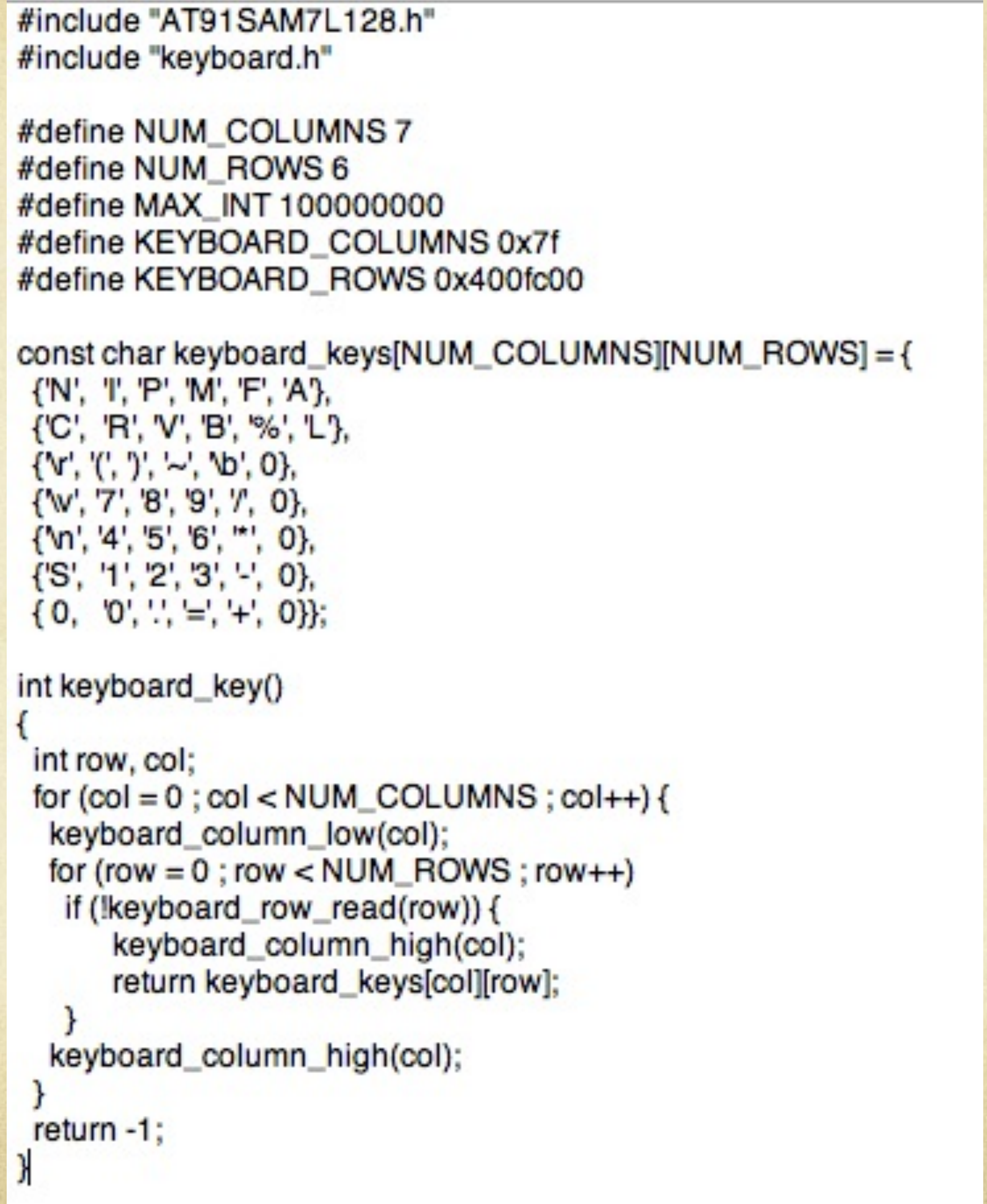
- 6.1 Lab 1: A Scrolling Display
- The general method of creating a display for integers is to handle the three distinct types of integer inputs: negative, zero and positive and to create methods to handle each.



```
#include "AT91SAM7L128.h"
#include "lcd.h"

int main()
{
  int digitNumber = 0; // how many digits are in the input
  int input = -12345; // the value you want to display
  int absInput = abs(input); // absolute value of the input

  lcd_init();
        // We are going to count how many digits are in the input
  while (absInput != 0) // If absInput is not 0
        {
        // divide absInput by 10, and count times we can divide by 10
    absInput = absInput/10;
    digitNumber++;
  }
        absInput = abs(input); // Overwrite with the original value
  int i;

  for (i=0; i < digitNumber; i++)
        {
    int remainder = absInput % 10;
        // 48 == '0' we are adding 48 for conversion int to char
    lcd_put_char7(remainder+48, digitNumber-i);
                absInput = absInput / 10;
  }
        // if the input is negative
  if (input < 0) {
    lcd_put_char7('-', 0);
  }
        // if the input is 0
        if(input == 0)
        {
                lcd_put_char7('0', 1);
        }
  return 0;
}
```

Scanning Functionality

- 6.2 Lab 2: Scanning the Keyboard
- Implements an algorithm which cycles through arrays of rows and columns The code interprets the changes in voltage as user inputs. This is implemented using nested for loops.

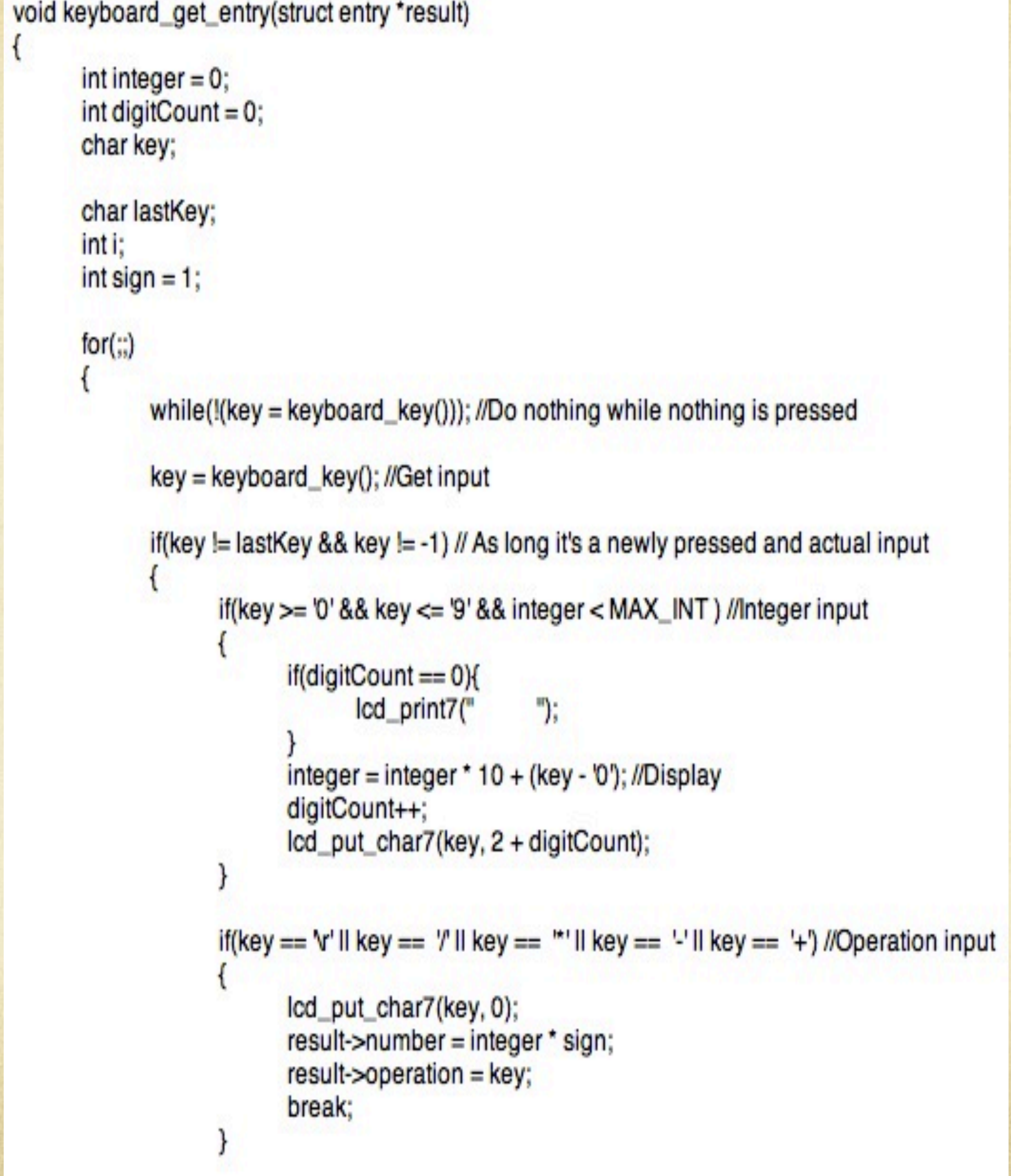


```
#include "AT91SAM7L128.h"
#include "keyboard.h"

#define NUM_COLUMNS 7
#define NUM_ROWS 6
#define MAX_INT 100000000
#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00

const char keyboard_keys[NUM_COLUMNS][NUM_ROWS] = {
 {'N',  'I', 'P', 'M', 'F', 'A'},
 {'C',  'R', 'V', 'B', '%', 'L'},
 {'\r', '(', ')', '~', '\b', 0},
 {'\v', '7', '8', '9', '/',  0},
 {'\n', '4', '5', '6', '*',  0},
 {'S',  '1', '2', '3', '-',  0},
 { 0,   '0', '.', '=', '+',  0}};

int keyboard_key()
{
 int row, col;
 for (col = 0 ; col < NUM_COLUMNS ; col++) {
  keyboard_column_low(col);
  for (row = 0 ; row < NUM_ROWS ; row++)
   if (!keyboard_row_read(row)) {
      keyboard_column_high(col);
      return keyboard_keys[col][row];
   }
  keyboard_column_high(col);
 }
 return -1;
}
```

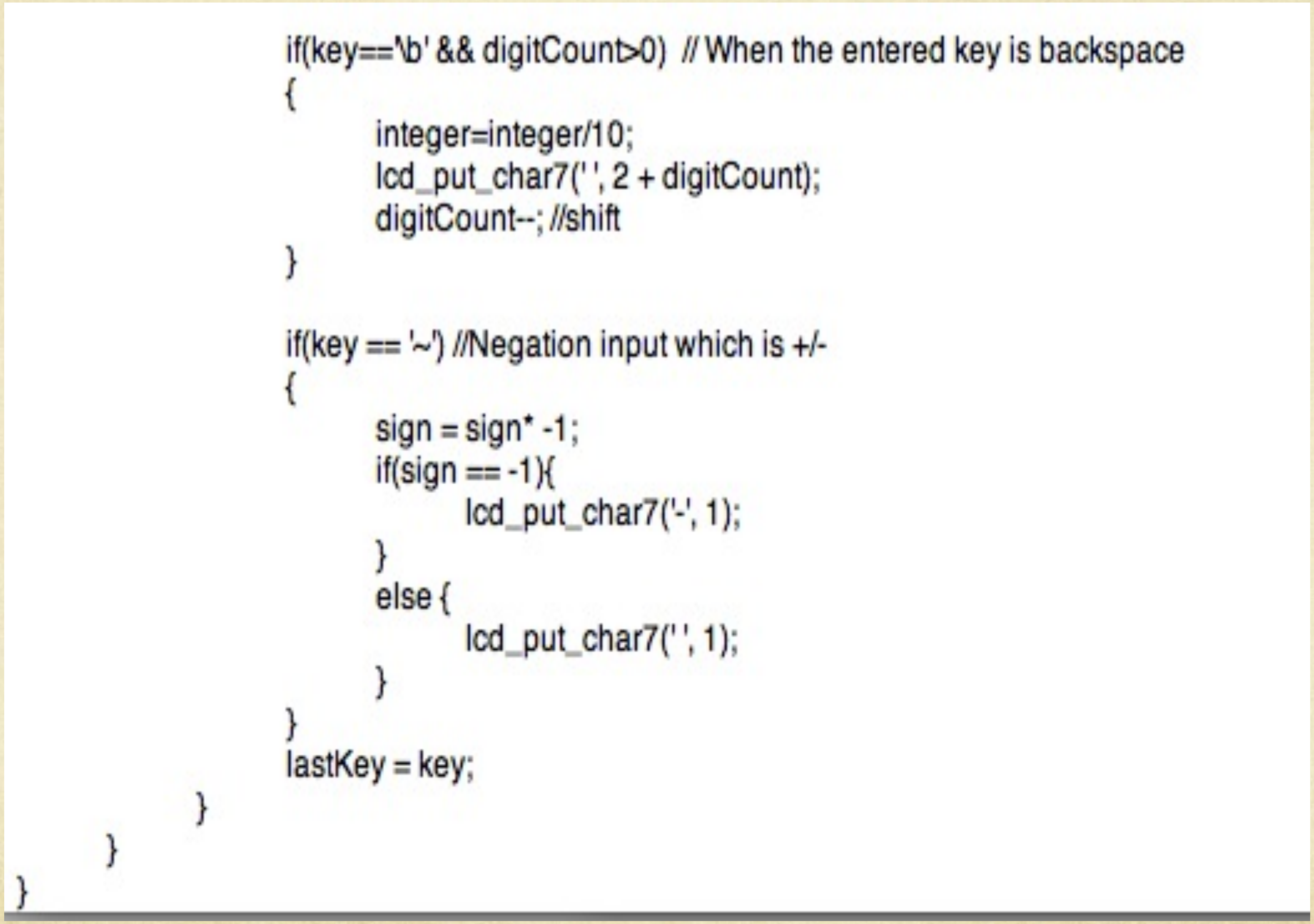
Input Functionality

- 6.3 Lab 3: Entering and Displaying Numbers

- The next goal of the project is to take in many integer inputs and an operation arguments to prepare for the arithmetic operations. The three main types of inputs were handled as: integer keys, operation keys, and the delete key. Each modifies both the display and the input user data.



```
void keyboard_get_entry(struct entry *result)
{
        int integer = 0;
        int digitCount = 0;
        char key;

        char lastKey;
        int i;
        int sign = 1;

        for(;;)
        {
                while(!(key = keyboard_key())); //Do nothing while nothing is pressed

                key = keyboard_key(); //Get input

                if(key != lastKey && key != -1) // As long it's a newly pressed and actual input
                {
                        if(key >= '0' && key <= '9' && integer < MAX_INT ) //Integer input
                        {
                                if(digitCount == 0){
                                        lcd_print7("        ");
                                }
                                integer = integer * 10 + (key - '0'); //Display
                                digitCount++;
                                lcd_put_char7(key, 2 + digitCount);
                        }

                        if(key == '\r' || key ==  '/' || key ==  '*' || key ==  '-' || key ==  '+') //Operation input
                        {
                                lcd_put_char7(key, 0);
                                result->number = integer * sign;
                                result->operation = key;
                                break;
                        }
```



```
            if(key=='\b' && digitCount>0)  // When the entered key is backspace
            {
                integer=integer/10;
                lcd_put_char7(' ', 2 + digitCount);
                digitCount--; //shift
            }

            if(key == '~') //Negation input which is +/-
            {
                sign = sign* -1;
                if(sign == -1){
                    lcd_put_char7('-', 1);
                }
                else {
                    lcd_put_char7(' ', 1);
                }
            }
            lastKey = key;
        }
    }
}
```

Lessons
Learned

Questions?