ENGI E1112 Departmental Project Report: Computer Science/Computer Engineering

Phoenetia Browne, Eileen Li, David Boucard

May 2012

Abstract

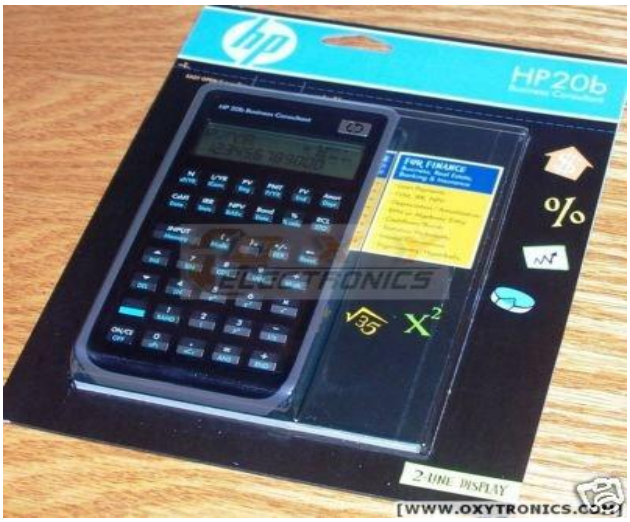
The objective of this project was to change the way a normal HP 20b business calculator operated. The calculator was basically broken apart and not functioning at the beginning and we were to write a new firmware for it using C programming. The aim was for by the end of the project the calculator would operate in RPN, which stands for Reverse Polish Notation. While working towards the goal we tried to make the calculator as user friendly and easy to use as possible.

In order to complete our final goal, we started of by creating a code that would allow the calculator to display a string of characters on the LCD screen. These characters were inputted on the computer. The next thing we did was to create a code that would make the calculator display the key that was pressed on the calculator's keypad on the LCD screen. So then the calculator was actually displaying something that was input by from the calculator itself. The last thing we did was to write a code that let the person enter numbers and the numbers would be displayed on the screen until and operation key was pressed. However, we did not get to the ultimate goal of making our calculator operate in RPN.

1 Introduction

The HP 20b business consultant is a financial calculated created by the Hewlett Packard Company in 2008[1]. It features an LCD display that can display up to 12 digits, an Atmel processor and a 6 x 7 keyboard matrix. The processor interface is also accessible which is why we were able to connect it to our computer and reprogram it to do what we wanted. The type of programming we used is called "embedded programming," because the calculator is actually a computer-controlled device, the code we created stays embedded in the calculator.

We used different aspects of C programming to complete these labs. In the second lab we the calculator had to be able to display the key that was pressed we used cell arrays and for loops. In the third that lab where the calculator would allow the user to edit the keys pressed we used pointers and structs.



2 User Guide

With our calculator, in order to display a number, press any of the number keys, and it will show up on the display screen. The maximum amount of numbers that can be displayed at once is 9 digits. The +/- key changes the sign of the number either from positive to negative, or negative to positive. The operation keys +, -, x, /, and input lets the calculator know that the user is finished inputting numbers and that the set of numbers are ready to be edited.

3 Social Implications

There are many reasons why easy to use calculators are an integral part of life at this time. Students use them to do quick calculations which they already know how to do by had, so it just speeds up the process. People use them on a daily basis for reasons such as tipping a cab driver, or checking to make sure that bank statements are correct. They are also used in more serious cases such as medicine. For example, an anesthesiologist would use it to calculate the right amount of anesthesia to give a patient so that they don't wake up during surgery. In this case calculators are used for its accuracy.

Calculators allow the world to move faster and allow for tasks to take smaller amounts of time. The calculator that we were supposed to create would be extremely user friendly for people of all ages and experience.

4 The Platform

The project used the HP-20B financial calculator which can perform basic scientific and statistical functions. It is made up of an Amtel AT91SAM7L128 embedded processor, LCD display, and simple keypad.

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

The JTAG header and power connector were added so users could change the software but retain battery power, as seen in the figure below and on the right side of the figure previous. [5]

4.1 The Processor

The calculator contains a 32-bit ARM7TDMI processor in the 7L series designed to run under low power, with 128 kilobytes of flash memory, smaller than normal for flash memory. It is also able to run on low batter based on the series of microcontrollers that allow for it. It uses just.5 mA/MHz while active and runs in a single-supply mode at 1.8 V and 100 nA while powered off. [3] The microcontroller also has a number of peripherals such as a 40-segment LCD controller, two USARTS, and an SPI. The system controller controls the power supply to ever peripheral and regulates the system's clock. The calculator belongs to the ARM7 family, one of the most widely produced 32-bit embedded processor due to the cheapness of production. [1]

During operation, the calculator has a linear three part operation of first, 'Fetch,' then 'Decode' and finally, 'Execute.' While the first part is run, the second part is decoded and the last command is fetched.

4.2 The LCD Display

The liquid crystal display (LCD) uses liquid chemicals to line up when put under electrical fields, allowing varying intensities of light to pass through to make an image. It is ultimately a matric of pixels in arranged rows and columns. [6]

Figure 4: Running the "Hello" program

The LCD of the calculator can display up to fifteen numbers, each with seven possible line segments. The twelve first numbers are large, while the last three are smaller. It has space for two negative signs, and in total, the calculator has forty possible line segments it can display. Additionally, the calculator also has space segment for commas. [2]

For class, three functions were given in a library to use. In the lab, the C source file called lcd.c has a 2-D matrix using the locations on the LCD to divide the display of each number into its respective seven parts. Secondly, lcd_put_char7 takes two parameters; the ASCII character intended to appear on the screen, ch, and the column it is intended to appear in, col. It uses the same matrix in the source file in order to display the user-inputted character on its corresponding space on the screen. The '7' in the function stands for the seven line segments available for each number. Finally, lcd_init turned on the power supply to initialize the LCD. [5]

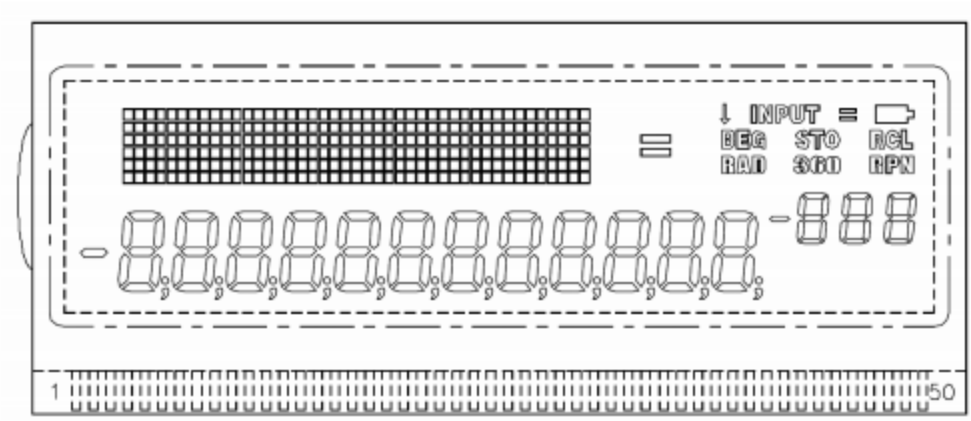


Figure 3: The LIBHP20bhp LCD Screen. It has a 43 x 6 pixel display matrix, various display indicators

4.3 Keyboard

The keyboard of the calculator is in a 6 x 7 matrix structure with leads and switches of two sheets of wiring separated by a buffer sheet. The vertical stacks are called the rows and the horizontal are called columns. [1] The processor, the SAM7: is connected to the keyboard by pins. Pressing a key connects a row and column, equalizing voltage of both row and column. Columns are initially set to high voltage and are the inputs. Rows, the outputs, take on the voltage once they are connected with the columns. When a button is pressed, and the column and row is connected, a circuit is completed. [4]

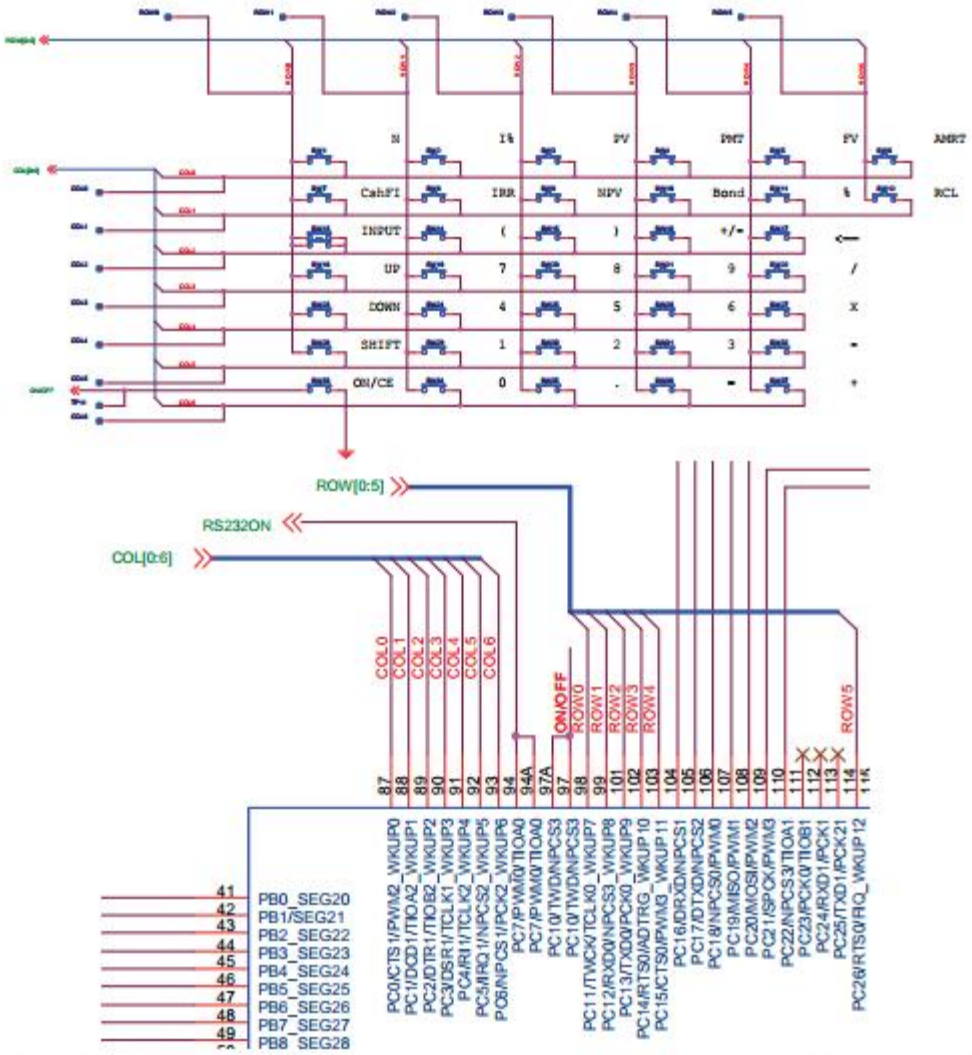


Figure 3: Schematics for the HP 20b keyboard. The top is the keyboard matrix itself; the bottom shows how the matrix is connected to the SAM7L chip. Note that the ON/CE key is separate (not part of the matrix) and that "columns" of keys are actually horizontal.

The processor identifies which key was pressed when the circuit is completed and references it to the given array of keys, and identifies which button is pressed. The function given in the second lab that allows for this is *keyboard_init* and *keyboard_column_high*, sets the keys to high voltage. The function *keyboard_column_low* also given, sets the voltage to low. Then the function *keyboard_row_read* identifies when a key is pressed when the voltage is changed, and returns which key is pressed from the array. [2]

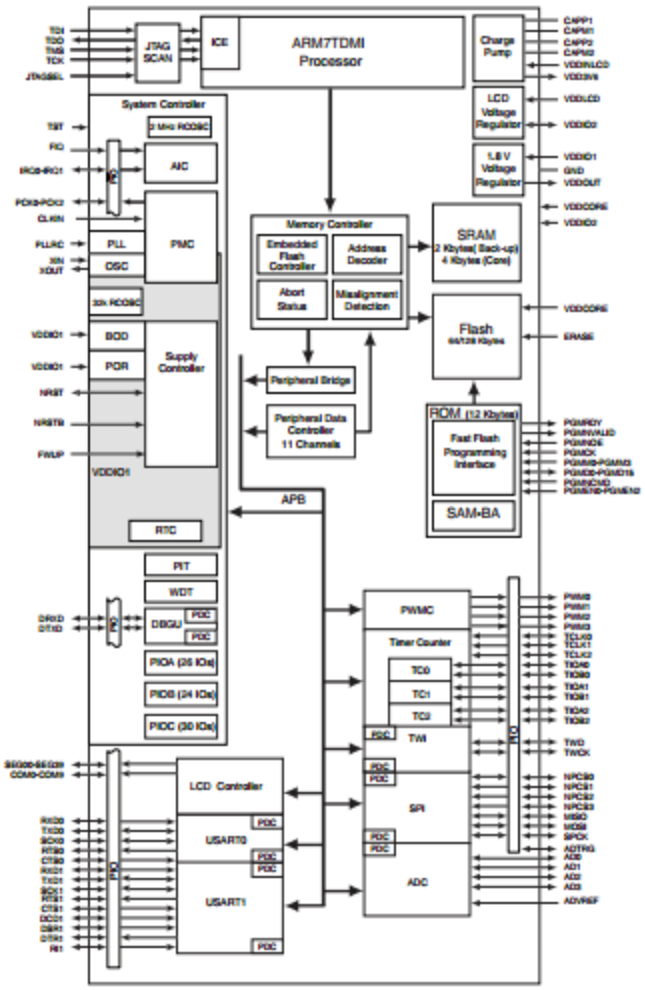


Figure 2: Block diagram of the AT91SAM7L chip. This consists of an ARM7TDMI processor core surrounded by memory, a system (clock) controller, an LCD controller, and a variety of other peripherals.

5 Software Architecture

The first lab required a function that took an integer argument and displayed it on the calculator. It had to display positive and negative numbers, and address if the user wanted too big of a number. The second lab created a function called *keyboard_key* that gave a code indicating which key was being pressed, or if not were pressed. The function had to be in *keyboard.c*. In the third and last lab our group was able to do, function *keyboard_get_entry* was added to *keyboard.c* which enabled the user to input a positive or negative integer followed by either the input button or one f the other operation keys. The function needed to wait until the user had finished entering something before if returned any information, unlike the second lab.

6 Software Details

The lab components for this project were each completed with the overarching goal of creating a fully functional calculator. Categorized, each lab implements the existing library code to produce: display functionality, user input functionality, and finally, the arithmetic functionality of the calculator. This project succeeds in implementing the first two functionalities.

6.1 Lab 1: A Scrolling Display

In order to build a usable calculator, basic display functions had to be written, allowing for integer inputs of all real numbers including negative, positive and zero integers. The task also requires the int to char method which then utilizes the ToString functions from the Standard.c file. The general method of approaching this problem is to handle three distinct types of integer inputs: negative, zero and positive and to create methods to handle each. Negative integers are briefly treated as positive integers and displayed an added '-' char. The final lab successfully handles all whole, real number inputs.

Solution for Lab 1: A Scrolling Display

```
#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 he absInput by 10, and count how many times
we can divide by 10
        absInput = absInput/10;
        digitNumber++;
    }

     absInput = abs(input); // Overwrite with the original value
so that we can do another calculation

    int i;

    for (i=0; i < digitNumber; i++)
     {
        int remainder = absInput % 10;
          // 48 == '0' we are adding 48 to convert integer to
corresponding character
        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;
}
```

6.2 Lab 2: Scanning the Keyboard

Using the actual physical layout of an average keyboard introduced in 1984 [1], the code for this lab implements an algorithm which cycles through arrays of rows and columns to distinguish between pressed keys and unpressed keys to indicate user inputs. The lab involves basic understanding of circuits and voltage/current manipulations underlying the keyboard layout scheme. The code interprets the changes in voltage as user inputs. The code also includes a two dimensional shorthand char array copy of the actual calculator input keys. This lab utilizes the voltage switches and detection to go through the calculator by column and within each column, by row. This is implemented using nested for loops.

Solution for Lab 2: Scanning the Keyboard

Main.c

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

//Write a function keyboard_key that returns a code that
indicates either which key is currently being pressed or that no
key
//is being pressed. Add this function to keyboard.c.
//Modify the code in main.c to use this function to report which
key is being pressed.

int main()
{
      int i, j;
      int returnCode;
      char pressedKey;

      char keys[COLUMNCOUNT][ROWCOUNT] =
```

```
      {{'N', 'I', 'P', 'P', 'F', 'A'},
           {'C', 'I', 'N', 'B', '%', 'R'},
           {'I', '(', ')', '+', '<' },
           {'^', '7', '8', '9', '/' },
           {'D', '4', '5', '6', '*'},
           {' ', '1', '2', '3', '-'},
           {'O', '0', '.', '=', '+'}};

      lcd_init();
      keyboard_init();

      lcd_print7("SEE");

      keyboard_column_low(0);

      for (;;)
      {
           returnCode = keyboard_key(); // returnCode contains i
* 10 + j
           j=returnCode%10;
           i=returnCode/10;
           pressedKey = keys[i][j];

           lcd_put_char7(pressedKey, 4);
      }

  return 0;
}
```

Keyboard.h

```
#ifndef _KEYBOARD_H
#  define KEYBOARD_H

#define COLUMNCOUNT 7
#define ROWCOUNT 6

/* Keyboard Layout

   Rows and columns on HP 20b schematic are reversed from what
you'd expect

   Columns are outputs; rows are read

            Row0  Row1  Row2 Row3 Row4 Row5
            PC11  PC12  PC13 PC14 PC15 PC26

Col0  PC0     N   I/YR  PV   PMT   FV  Amort
Col1  PC1   CshFI  IRR  NPV  Bond  %    RCL
Col2  PC2   INPUT   (    )   +/-   <-
```

```
Col3  PC3    UP     7    8    9    /
Col4  PC4   DOWN    4    5    6    *
Col5  PC5   SHIFT   1    2    3    -
Col6  PC6           0    .    =    +

  ON/CE is separate
 */

// Initialize the keyboard and set all columns high with pullups
on the rows
extern void keyboard_init(void);

// Set the given column high
extern void keyboard_column_high(int column);

// Set the given column low
extern void keyboard_column_low(int column);

// Return true if the row is high, false otherwise
extern int keyboard_row_read(int row);

// Returns the code that indicates which (and if) key has been
pressed
extern int keyboard_key();

#endif
```

Keyboard.c

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

#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00


const unsigned char keyboard_row_index[] = {11,12,13,14,15,26};

void keyboard_init()
{
  // Initialize the keyboard: Columns are outputs, rows are
inputs
  AT91C_BASE_PMC->PMC_PCER = (uint32) 1 << AT91C_ID_PIOC; // Turn
on PIOC clock
  AT91C_BASE_PIOC->PIO_PER = KEYBOARD_ROWS | KEYBOARD_COLUMNS; //
Enable control
  AT91C_BASE_PIOC->PIO_PPUDR = KEYBOARD_COLUMNS; // Disable
pullups on columns
  AT91C_BASE_PIOC->PIO_OER = KEYBOARD_COLUMNS;   // Make columns
outputs
```

```
  AT91C_BASE_PIOC->PIO_PPUER = KEYBOARD_ROWS;    // Enable
pullups on rows
  AT91C_BASE_PIOC->PIO_ODR = KEYBOARD_ROWS;      // Make rows
inputs

  AT91C_BASE_PIOC->PIO_SODR = KEYBOARD_COLUMNS;  // Drive all
columns high

}

void keyboard_column_high(int column)
{
  AT91C_BASE_PIOC->PIO_SODR = 1 << column;
}

void keyboard_column_low(int column)
{
  AT91C_BASE_PIOC->PIO_CODR = 1 << column;
}

int keyboard_row_read(int row)
{
  return (AT91C_BASE_PIOC->PIO_PDSR) & (1 <<
keyboard_row_index[row]);
}

int keyboard_key()
{
      int i, j;
      int pressedKey;

      // set every column to high to initialize
      for(i = 0; i < COLUMNCOUNT; i++)
      {
            keyboard_column_high(i);
      }

      // Check each column and set one column at a time to 'low'
      for(i=0; i < COLUMNCOUNT; i++)
      {
            keyboard_column_low(i);
            // Check each row
            for(j=0; j<ROWCOUNT; j++)
            {
                  //If the row is low, it means button has been
pressed
                  if(!keyboard_row_read(j))//if it is high, then
statement will be false, program will skip if statement
                  {
                        keyboard_column_high(i);
                        pressedKey = i * 10 + j;
```

```
                        return pressedKey; // return the value
pressedKey back to main function and terminate the function.
                }
            }
      }
      return -1; // return -1 if no key has been pressed
}
```

6.3 Lab 3: Entering and Displaying Numbers

Once display and scanning functionality are implemented, the calculator is ready to take actual inputs. The next goal of the project is to take in two integer inputs and an operation argument to prepare for the arithmetic operations. This portion of the lab requires significant use of pointers to store input data as the user presses multiple keys. The inputs are stored as continually updated int. The user input is taken as either an integer key, operation key, or delete key. The '-' operation can alternate as a subtraction function or a negation operator if an integer has not been input yet. The delete key alters the display while also modifying the current integer input only when an integer input has been made. These inputs can then ready to be sent to perform basic arithmetic functions on integer inputs.

Solution for Lab 3: Entering and Displaying Numbers

Main.c

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

int main()
{
  struct entry entry;
  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init();

      for(;;)
      {
         keyboard_get_entry(&entry);

         lcd_print7("EX          ");
         lcd_put_char7(entry.operation,  4);

         int temp = entry.number; //struct that defines data from
temp
         if(temp < 0)
         {
             lcd_put_char7('-', 6);
```

```
            temp = -temp;
        }

        lcd_put_char7(temp % 10 + '0', 11); //takes the remainder
and adds to the ASCII value
        temp = temp/10;
        lcd_put_char7(temp % 10 + '0', 10);

     }

  return 0;
}
```

Part of lcd.c

```
// Display a string on the 7-segment display starting from the
leftmost column
void lcd_print7(const char *c)
{
  int column;
  char ch;
  for (column = 0 ; column < LCD_7_COLUMNS && (ch = *c++) ;
++column)
    lcd_put_char7(ch, column);
}

// Display a right-justified integer with an optional leading
negative sign
void lcd_print_int_neg(int negative, unsigned int n)
{
  int column = LCD_7_COLUMNS - 4; // Start rightmost
  do {
    lcd_put_char7(n%10 + '0', column--);
    n /= 10;
  } while (n);
  if (negative) lcd_put_char7('-', column--);
  while (column >= 0) lcd_put_char7(' ', column--);
}

// Display a right-justified signed integer
void lcd_print_int(int n) {
  if (n < 0) lcd_print_int_neg(1, -n);
  else lcd_print_int_neg(0, n);
}
```

Keyboard.c

```
#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 unsigned char keyboard_row_index[] = {11,12,13,14,15,26};

/* Character codes returned by keyboard_key */

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}};

void keyboard_init()
{
  // Initialize the keyboard: Columns are outputs, rows are
inputs
  AT91C_BASE_PMC->PMC_PCER = (uint32) 1 << AT91C_ID_PIOC; // Turn
on PIOC clock
  AT91C_BASE_PIOC->PIO_PER = KEYBOARD_ROWS | KEYBOARD_COLUMNS; //
Enable control
  AT91C_BASE_PIOC->PIO_PPUDR = KEYBOARD_COLUMNS; // Disable
pullups on columns
  AT91C_BASE_PIOC->PIO_OER = KEYBOARD_COLUMNS;   // Make columns
outputs
  AT91C_BASE_PIOC->PIO_PPUER = KEYBOARD_ROWS;    // Enable
pullups on rows
  AT91C_BASE_PIOC->PIO_ODR = KEYBOARD_ROWS;      // Make rows
inputs

  AT91C_BASE_PIOC->PIO_SODR = KEYBOARD_COLUMNS;  // Drive all
columns high

}

void keyboard_column_high(int column)
{
  AT91C_BASE_PIOC->PIO_SODR = 1 << column;
}

void keyboard_column_low(int column)
{
  AT91C_BASE_PIOC->PIO_CODR = 1 << column;
}

int keyboard_row_read(int row)
```

```
{
  return (AT91C_BASE_PIOC->PIO_PDSR) & (1 <<
keyboard_row_index[row]);
}

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;
}

void keyboard_get_entry(struct entry *result)
{

      int integer = 0;
      int digitCount = 0;
      char key;
      char lastKey = -1;
      int i;
      int sign = 1;


      for(;;)
      {
            key = keyboard_key();

            if(key != lastKey && key != -1 )
            {
                  // When the key entered is a number
                  if(key >= '0' && key <= '9' && integer < MAX_INT
)
                  {
                 if(!(lastKey == '0' && key == '0' && digitCount
== 0))
                  {
                      if(digitCount == 0)
                      {
                          lcd_print7("           ");
                      }
                      integer = integer * 10 + (key - '0');
                      digitCount++;
                      lcd_put_char7(key, 2 + digitCount);
                  }
```

```
            }

                // When the entered key is operation keys
                if(integer != 0 && (key == '\r' || key ==  '/' ||
key ==  '*' || key ==  '-' || key ==  '+'))
                {
                     lcd_put_char7(key, 0);

                     result->number = integer*sign;
                     result->operation = key;

                     break;
                }

                // When the entered key is backspace
                if(key=='\b' && digitCount>0)
                {
                     integer=integer/10;

                     lcd_put_char7(' ', 2 + digitCount);
                     digitCount--;
                }

                //When the entered key is +/-
                if(key == '~')
                {
                if(integer == 0)
                {
                    lcd_print7("           ");
                }
                     //lcd_print7("+/-");
                     sign = sign* -1;
                     if(sign == -1)
                     {
                          lcd_put_char7('-', 2);
                     }
                     else {
                          lcd_put_char7(' ', 2);
                     }

                }
                lastKey = key;
           }
      }
}
```

7 Lessons Learned

For this project group, the lab presented a two-fold challenge of learning C programming syntax and computer science methodology with no prior programming experience while completing lab projects in a timely manner. The best way to learn is to

fail and try again, which happened on multiple occasions during the course. Overall, there were many valuable lessons learned in this course from creating sound, concise algorithms to maintaining clear and concise communication of ideas. Perhaps this project may have been simple for more experienced students, but for fresh inductees to computer science, the HP20b Calculator project is an great challenge and introduction to algorithmic problem solving and programming.

8 Criticism of the Joy of Engineering Lab

While the Computer Engineering/Computer Science Lab was one of the more fulfilling lab sections offered this semester, it was lacking in one aspect. While the structure of the course was very well organized, there was little flexibility for wide range of students entering the lab. Some students were clearly unchallenged and underwhelmed while others, like our group were unable to complete the entire project. A more detailed description of the lab section's required prior experience and expected student skill level would have been useful in lab section selection.

References

[1] Introduction to Keyboards: Physical Keyboard Layout. Online http://www-01.ibm.com/software/globalization/topics/keyboards/physical.html

[2] "HP 20b Business Consultant Financial Calculator Overview." HP: Hewlett-Packard. Web. 09 Dec. 2011. <http://h10010.www1.hp.com/wwpc/us/en/sm/WF05a/215348-215348- 64232-20036-215349-3732534.html>.

[3] Hewlett-Packard Company. The RPN Stack. HP 20b Business Consultant Financial Calculator Manual. 18. Web. 9 Dec. 2011. <http://h10010.www1.hp.com/wwpc/pscmisc/vac/us/product_pdfs/HP_20b_Online_Manual.pdf>.

[4] HP Business Consultant Financial Calculator. Digital image. Web. 06 May 2012. <http://www.collectibles-articles.com/antique/collectible-image-large/hp-20b-business-consultant-financial-calculator-new_330337963278.jpg>.

[5] Edwards, Stephen. "Repurposing an HP Calculator." Web. 05 May 2012. <http://www.cs.columbia.edu/~sedwards/classes/2011/gateway-fall/keyboard.pdf>.

[6] Hp-20b repurposing project. Online http://www.wiki4hp.com/doku.php?id=20b:repurposing_project.