ENGI E1112 Departmental Project Report: Computer Science/ComputerEngineering

Joshua Boggs and Shensi Ding

May, 2012

Abstract

At the beginning of the semester, Professor Stephen Edwards informed his students that he had wiped the firmware on a set of HP 20b Calculators. Students were then assigned the daunting task of rebuilding the software from the ground up. Project partners Joshua Boggs and Shensi Ding collaborated to complete this task along with the assistance of Professor Stephen Edwards and Teaching Assistant Yoonji Shin. The following report provides details about the calculator operated on, instructions on using the calculator, and an in depth look at the software written to complete the assignment.

1 Introduction

The following documentation focuses on the HB 20B Business Calculator. Over the course of the past semester we have rewritten basic firmware for the complex calculator. This process began with writing the software to read the keyboard for input, and display pressed keys on the LCD. Next we wrote the code which would allow users to enter and edit numbers. Finally, we wrote the code which would allow user to perform calculations using reverse Polish notation.

The contents of this document include a user guide, social implications, hardware/software architectures, details of the software, our lessons learned, and criticisms of the course.

2 User Guide

The calculator we have programmed uses reverse Polish notation. In this notation, the operators come after the operands. No parentheses are used, simply number keys, operators, and the input key.

To carry out a simple operation, begin by typing your first operand into the keyboard. Then, press INPUT. The number you entered will appear on the LCD. Next, type out the second operand, and again press INPUT. Finally, press the operator you would like to use. The answer will then appear on the LCD. The following examples should illustrate this concept.

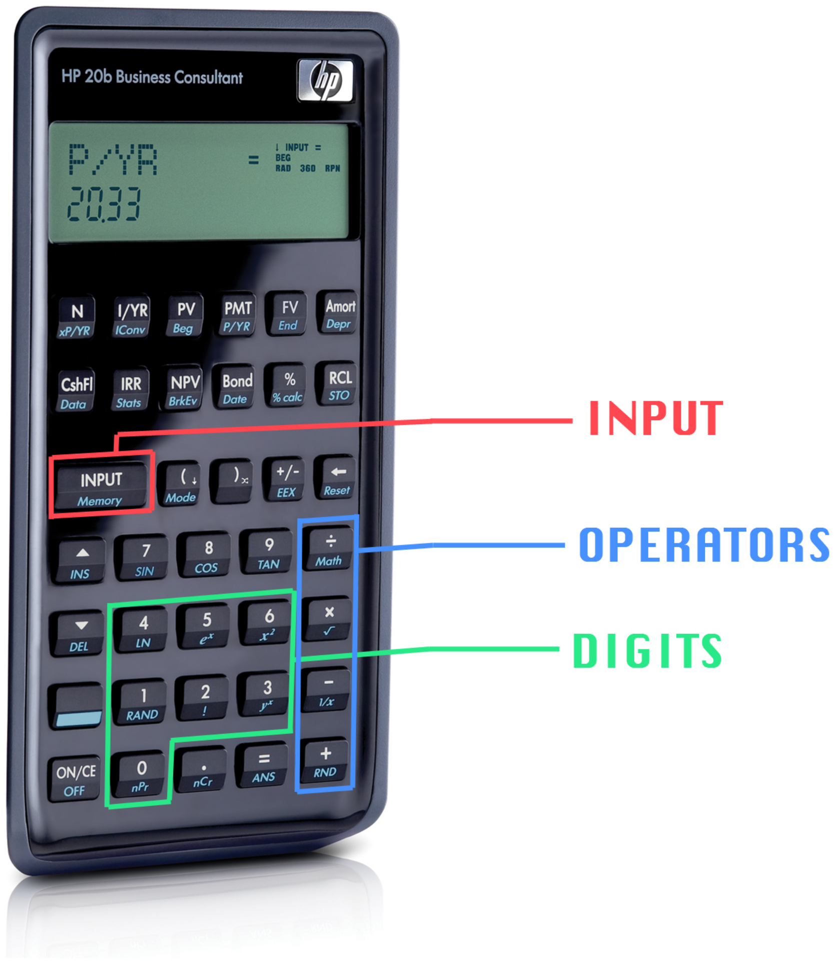


Figure 1: The HB 20B Business Calculator

Example 1: A Basic Calculation – 9 + 12

- Press the '9' digit key
- Press 'input' to save '9' as the first operand
- Press the '1' digit key
- Press the '2' digit key

- Press 'input to save '12' as the second operand
- Press the addition operator '+'

If all is entered correctly, the number '21' will appear on the LCD.
Example 2: A more complex operation – (3+5)x(7-2)

- Press the '3' digit key
- Press the 'input' key to save '3' as the first operand
- Press the '5' digit key
- Press 'input' to save '5' as the second operand
- Press the addition operator '+' to add the first two operands
- Press the '7' digit key
- Press 'input' to save '7' as the third operand
- Press the '2' digit key
- Press 'input' to save '2' as the fourth operand
- Press the subtraction operator '-' to subtract the third and fourth operands
- Press the multiplication operator 'x' to multiply the resultant operands in the stack

If all is entered correctly, the number '40' will appear on the screen.

3 Social Implications

The HP 20B Business Calculator provides efficient handheld calculations. This can simplify everyday problems and tasks. Calculations and organized and reliable, making error-prone hand calculations unnecessary. The calculator is very easy to use, and even for beginners. Physically, the calculator is thin and light for optimal portability. Aesthetically, it is sleek and stylish, without sacrificing ergonomics. Because this calculator is portable, easy to use, and drains little power, it could be used extensively in third world countries. Several fields are greatly improved by the calculator, including education, commerce, and scientific research. In the professional sector, employees of finance, real estate, insurance, accounting, and statistics may find the calculator to be particularly helpful. It is critical for all nations, including developing nations, to have access to affordable technology.

4 The Platform

The calculator consists mainly of a LCD connected to an Atmel AT91SAM7L128 processor. The calculator has been modified to connect to a computer through a JTAG port. We interact with the calculator by connecting our computers through this port.

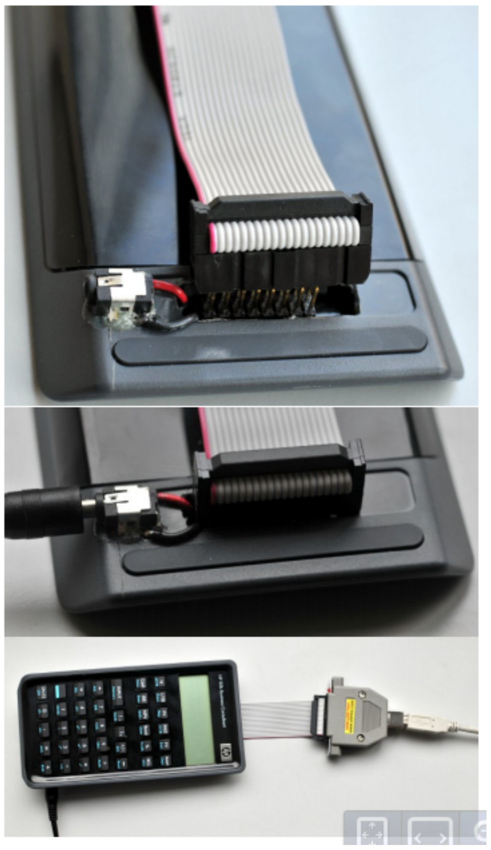


Figure 2: The JTAG port

4.1 *The Processor*

The calculator utilizes an Atmel AT91SAM7L128 processor. It is a 30MHz low-power ARM7 System with 128 Kbytes of internal high-speed flash.

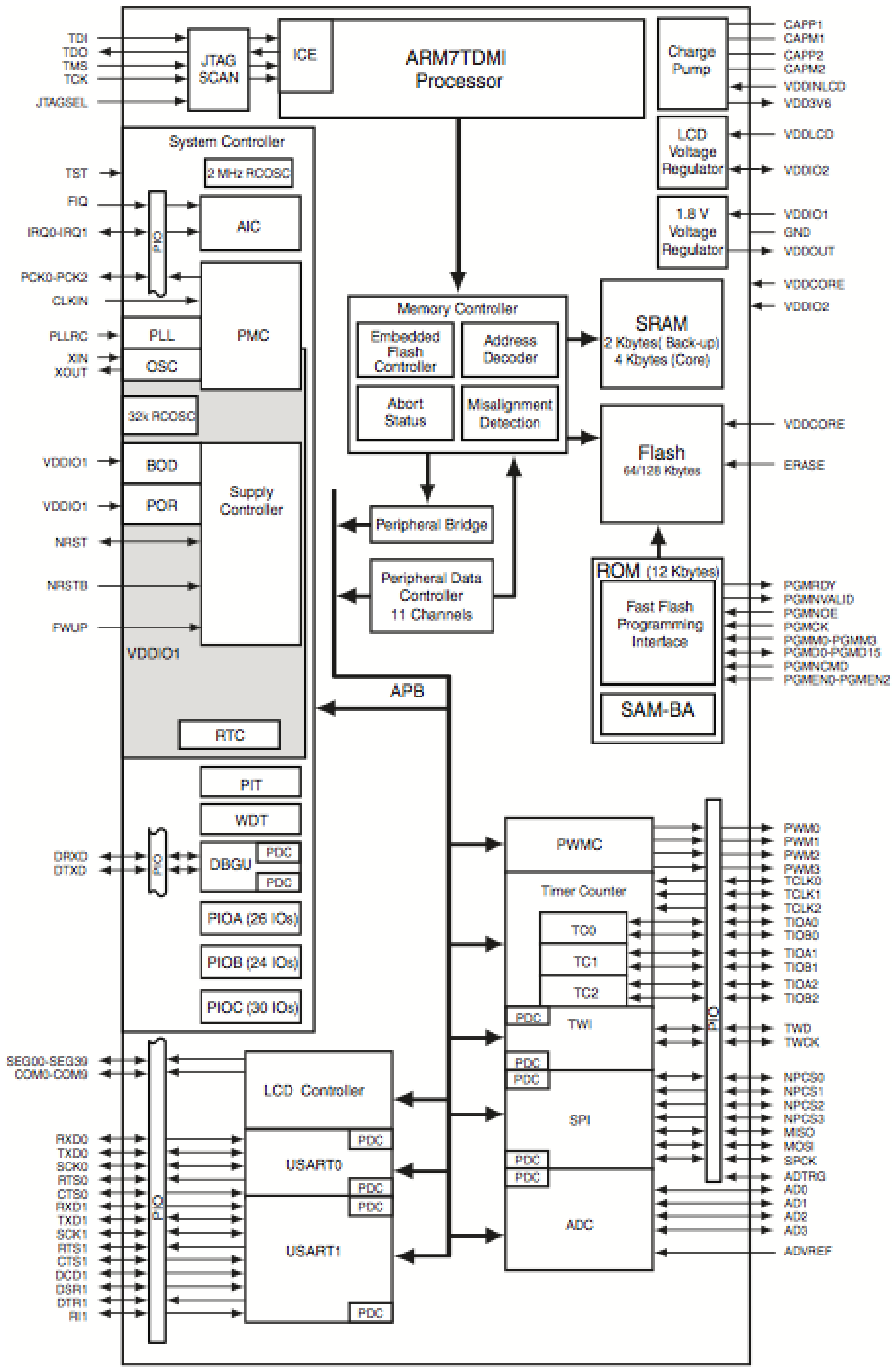


Figure 3: A block diagram of the AT91SAM7L

4.2 The LCD Display

The calculator has a large 2-line liquid crystal display. The primary section of the display, the bottom line, can show up to 12 7-segment characters at a single time, and 3 exponents. Above is an 8-character scrolling top line.

The LCD interacts with a few different functions in the code:

lcd_put_char7 – Prints the value specified in the argument.
lcd_print_int(stack[pointer]) – Prints the specified value in the stack.
lcd_print7() – Prints the value in the argument.

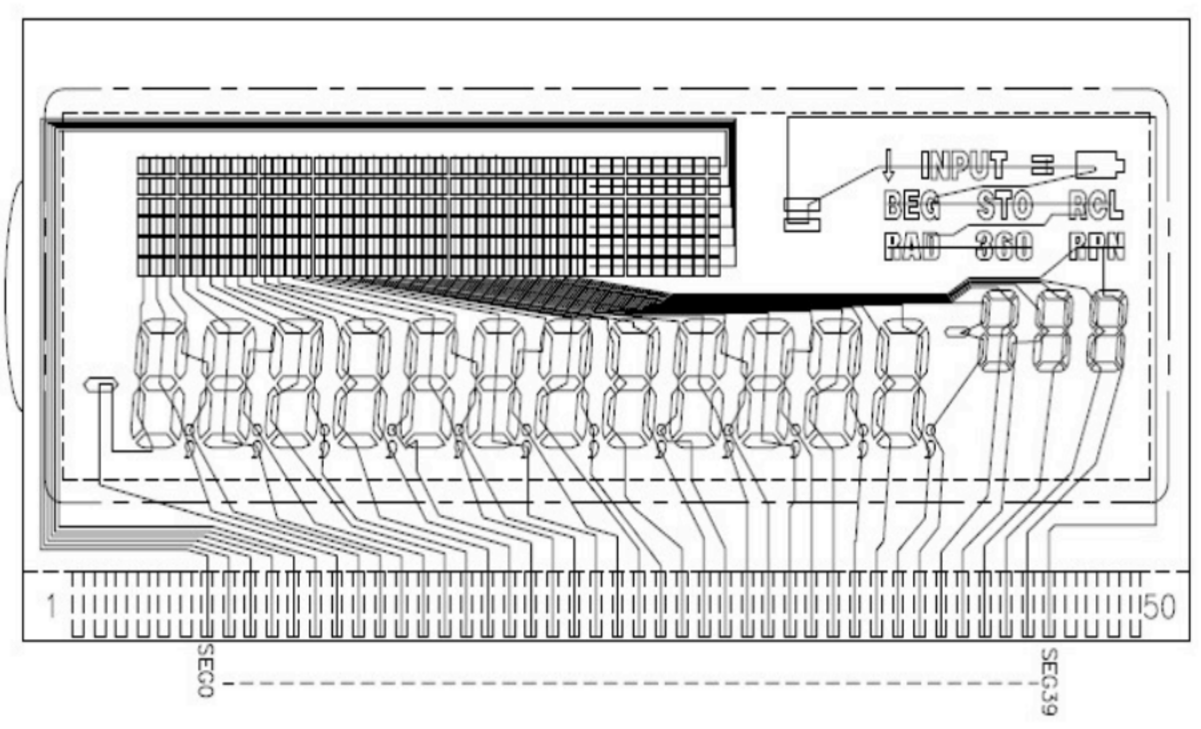
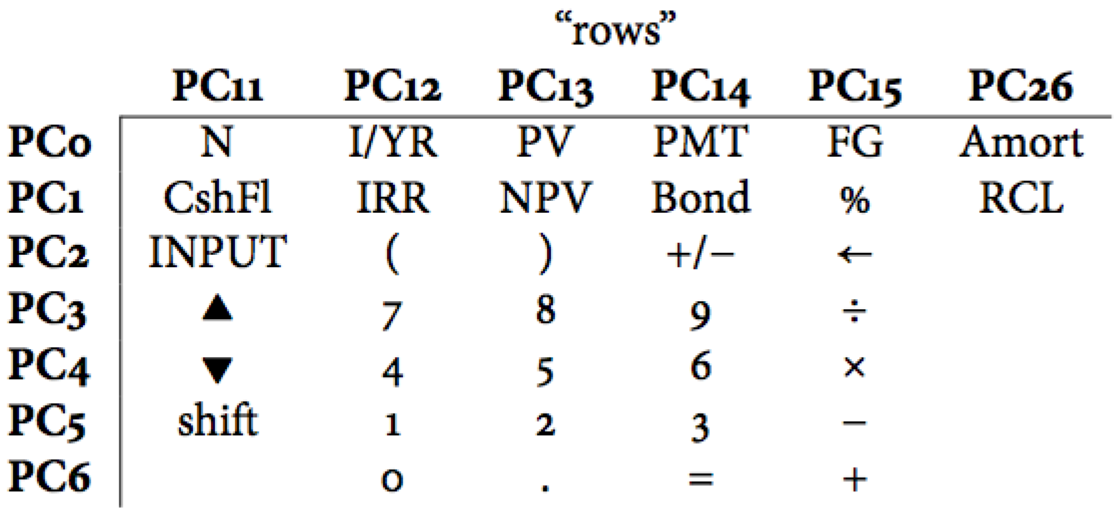


Figure 4: LCD Architecture

4.3 The Keyboard

The portion which takes up most of the real estate on the calculators facade is the numeric keyboard. It contains both digits and a plentitude of various operators. For the project, we only utilize the basic arithmetic operators, along with the digits and input key. When a key is pressed, one pin is shorted for the column, and another pin is shorted for the row. This is how we are able to read which key is pressed.



	“rows”					
	PC11	**PC12**	**PC13**	**PC14**	**PC15**	**PC26**
PC0	N	I/YR	PV	PMT	FG	Amort
PC1	CshFl	IRR	NPV	Bond	%	RCL
PC2	INPUT	(	)	+/−	←	
PC3	▲	7	8	9	÷	
PC4	▼	4	5	6	×	
PC5	shift	1	2	3	−	
PC6		0	.	=	+	

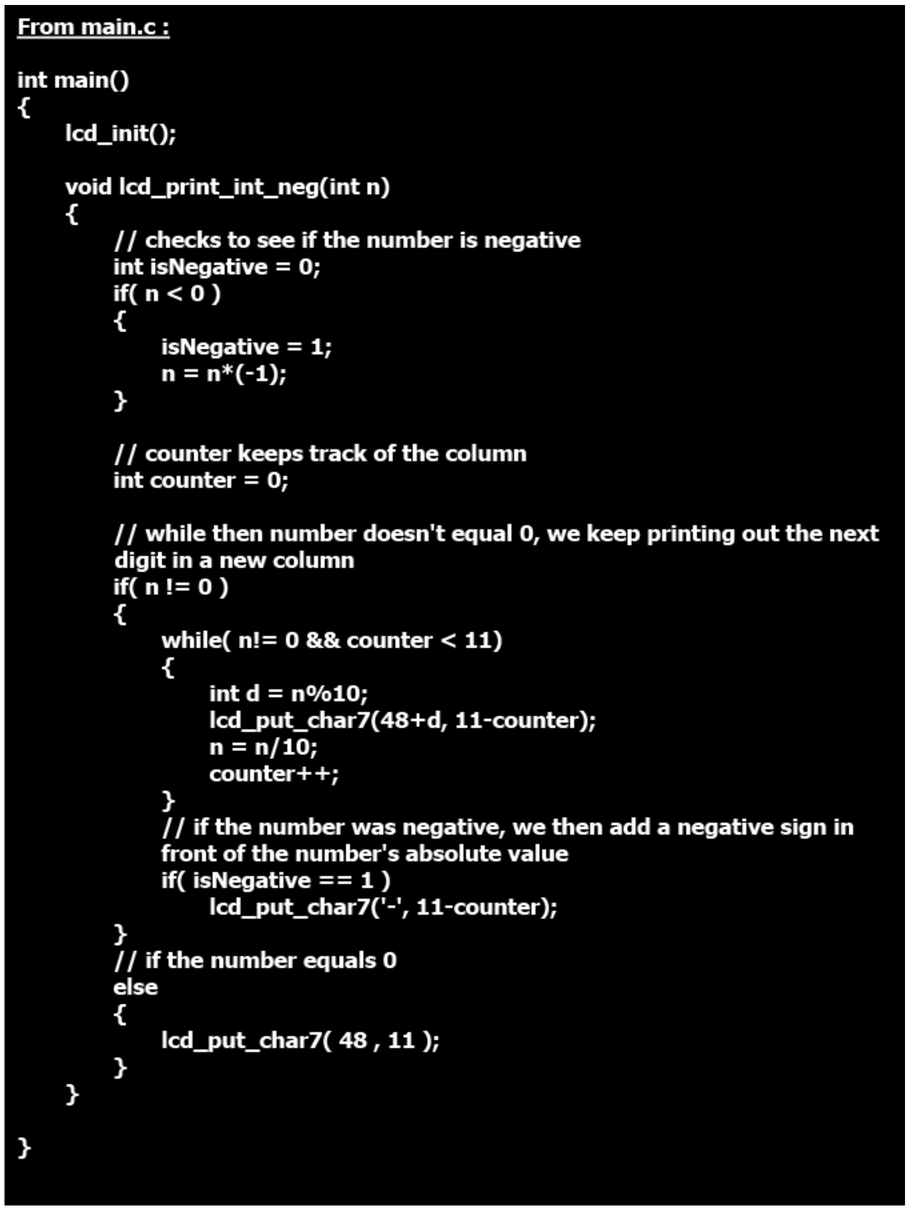
Figure 5: A simple schematic of the keyboard architecture

5 Software Details

This section includes the code we wrote to solve each lab. Explanations of the code are commented throughout the figures. In order to fix bugs, various functions and loops had to be written in the code. These are included in the figures as well. Our solutions seem to solve each of the tasks correctly and efficiently. If they were to be perfected, it is possible that they could be shorter.

5.1 Lab 1: A Scrolling Display

In this portion of the project, we wrote the software to display characters on the screen.

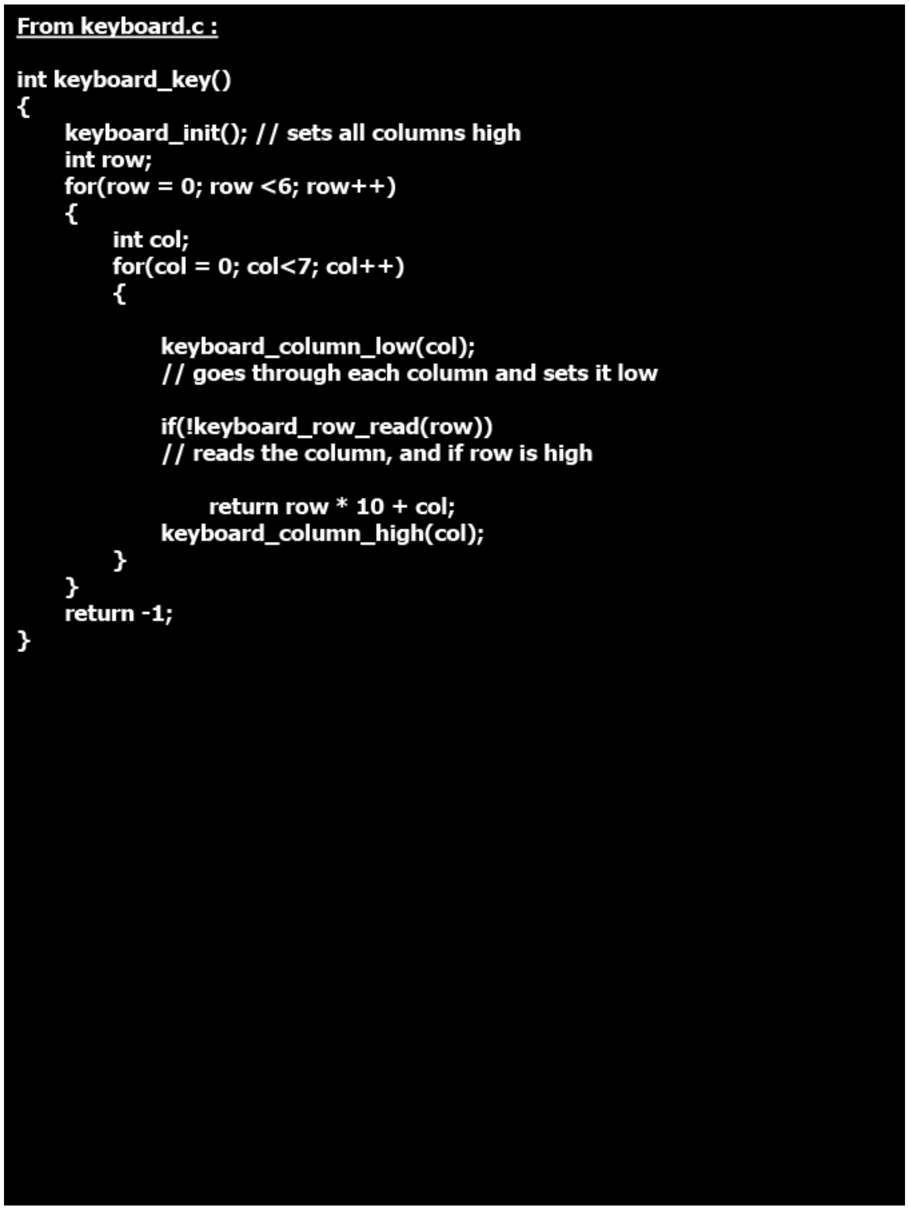


```
From main.c :

int main()
{
      lcd_init();

      void lcd_print_int_neg(int n)
      {
            // checks to see if the number is negative
            int isNegative = 0;
            if( n < 0 )
            {
                  isNegative = 1;
                  n = n*(-1);
            }

            // counter keeps track of the column
            int counter = 0;

            // while then number doesn't equal 0, we keep printing out the next
            digit in a new column
            if( n != 0 )
            {
                  while( n!= 0 && counter < 11)
                  {
                        int d = n%10;
                        lcd_put_char7(48+d, 11-counter);
                        n = n/10;
                        counter++;
                  }
                  // if the number was negative, we then add a negative sign in
                  front of the number's absolute value
                  if( isNegative == 1 )
                        lcd_put_char7('-', 11-counter);
            }
            // if the number equals 0
            else
            {
                  lcd_put_char7( 48 , 11 );
            }
      }

}
```

Figure 6: The solution for Lab 1: A Scrolling Display

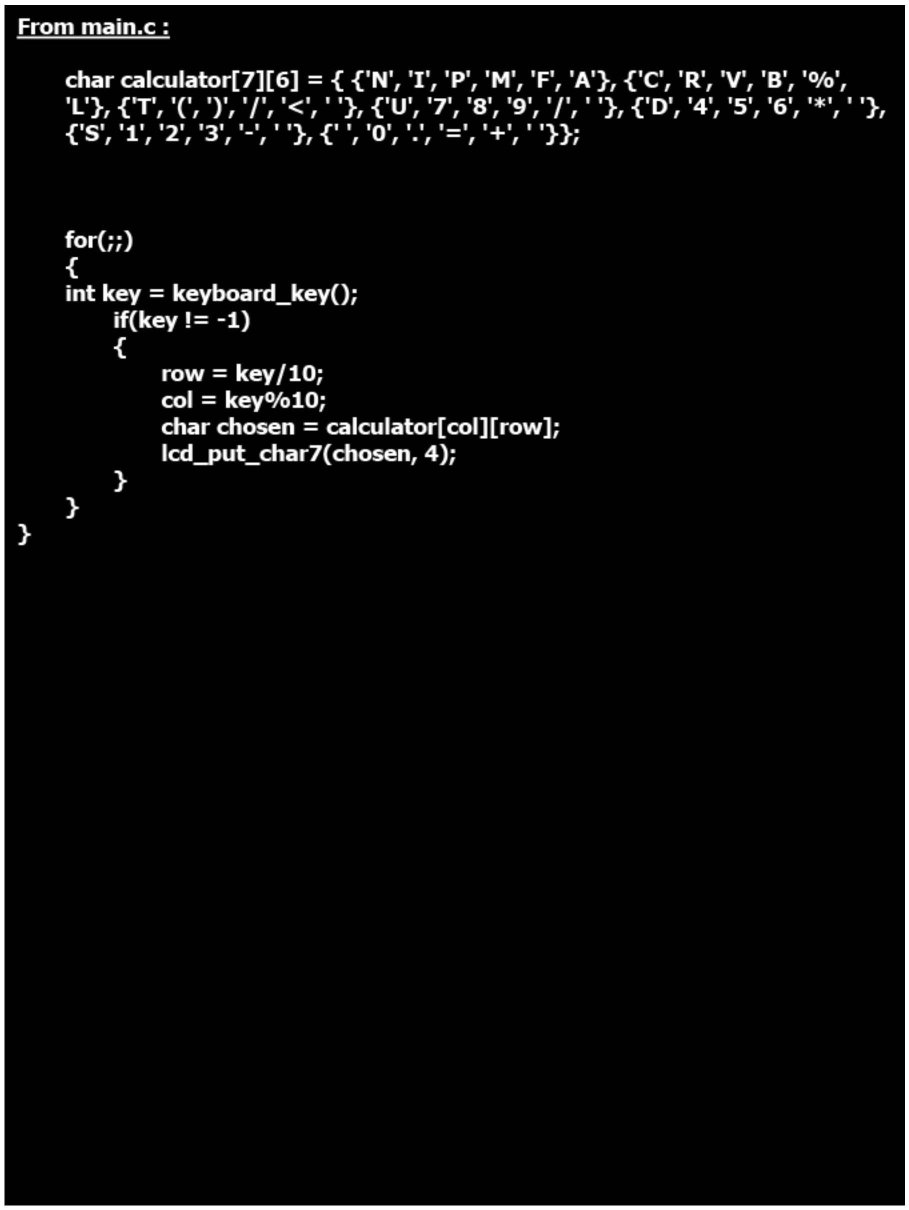
5.2 Lab 2: Scanning the Keyboard

For the next portion of the project, we wrote the code to respond to a pressed key by displaying that key on the screen.


From keyboard.c :

```
int keyboard_key()
{
      keyboard_init(); // sets all columns high
      int row;
      for(row = 0; row <6; row++)
      {
            int col;
            for(col = 0; col<7; col++)
            {

                  keyboard_column_low(col);
                  // goes through each column and sets it low

                  if(!keyboard_row_read(row))
                  // reads the column, and if row is high

                        return row * 10 + col;
                  keyboard_column_high(col);
            }
      }
      return -1;
}
```

Figure 7: The solution for Lab 2: Scanning the Keyboard (Part 1)



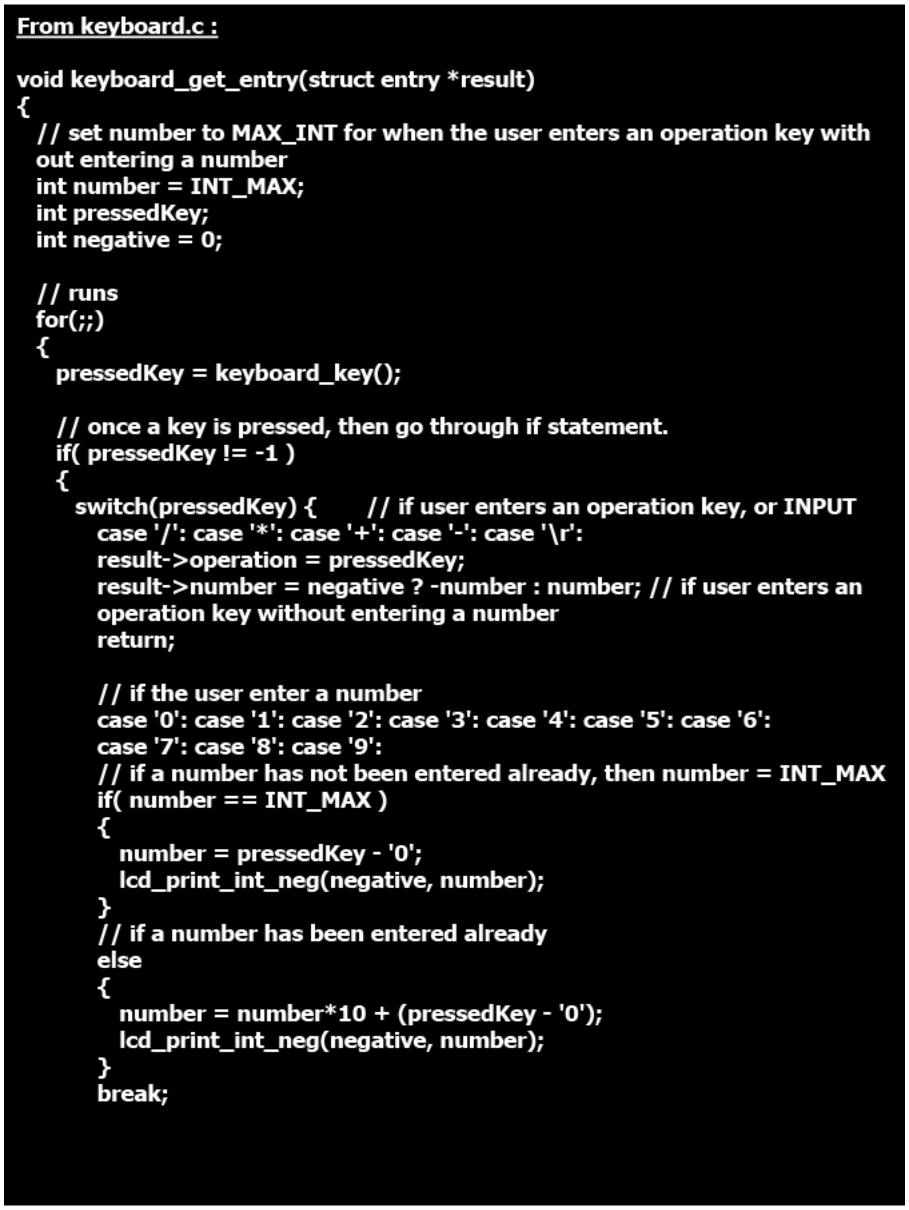
From main.c :

```
    char calculator[7][6] = { {'N', 'I', 'P', 'M', 'F', 'A'}, {'C', 'R', 'V', 'B', '%',
    'L'}, {'T', '(', ')', '/', '<', ' '}, {'U', '7', '8', '9', '/', ' '}, {'D', '4', '5', '6', '*', ' '},
    {'S', '1', '2', '3', '-', ' '}, {' ', '0', '.', '=', '+', ' '}};



    for(;;)
    {
    int key = keyboard_key();
        if(key != -1)
        {
            row = key/10;
            col = key%10;
            char chosen = calculator[col][row];
            lcd_put_char7(chosen, 4);
        }
    }
}
```

Figure 8: The solution for Lab 2: Scanning the Keyboard (Part 2)

5.3 Lab 3: Entering and Displaying Numbers

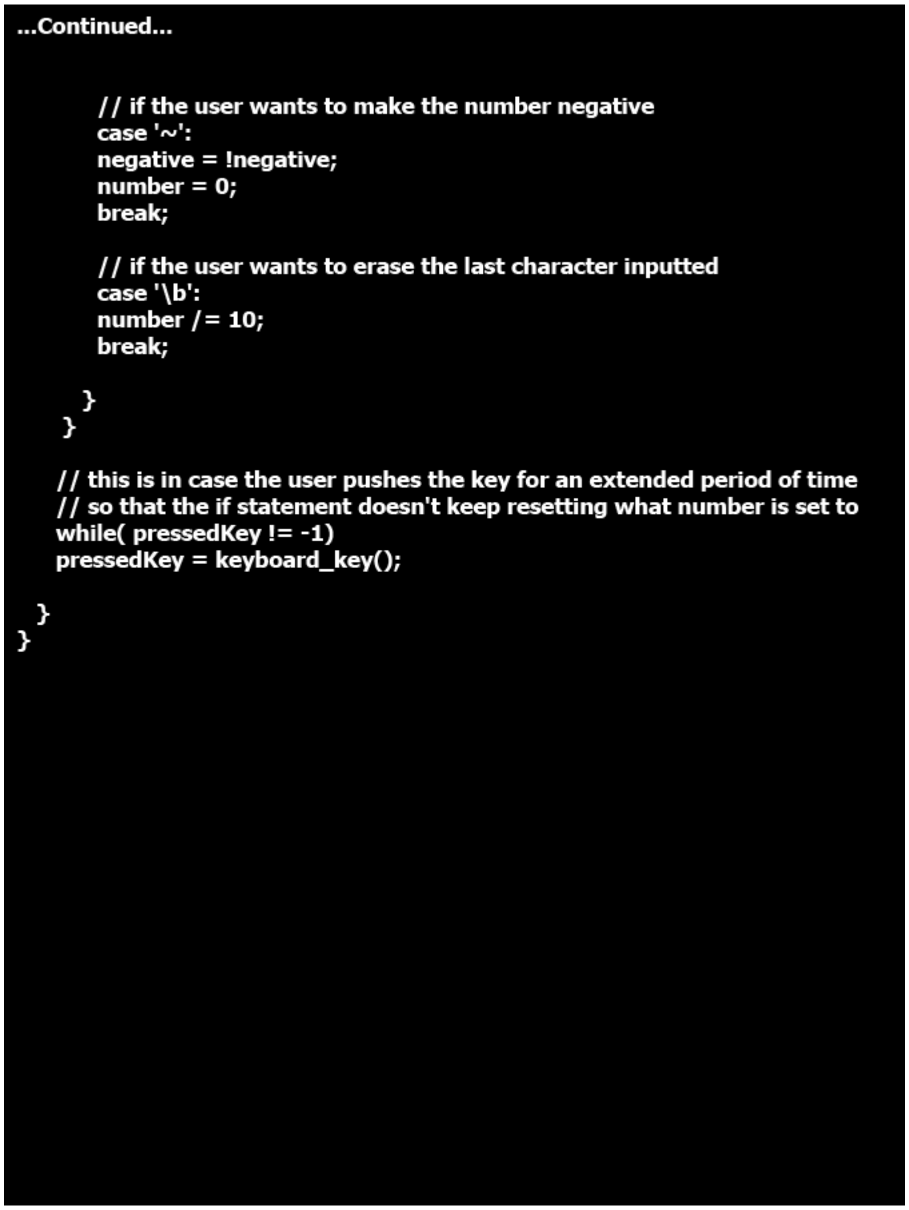
The next portion of the code we worked on allows users to enter and edit numbers. In this portion of the code, we implemented the useful switch function, which we were taught to use in class.



From keyboard.c :

```
void keyboard_get_entry(struct entry *result)
{
  // set number to MAX_INT for when the user enters an operation key with
  out entering a number
  int number = INT_MAX;
  int pressedKey;
  int negative = 0;

  // runs
  for(;;)
  {
    pressedKey = keyboard_key();

    // once a key is pressed, then go through if statement.
    if( pressedKey != -1 )
    {
      switch(pressedKey) {      // if user enters an operation key, or INPUT
        case '/': case '*': case '+': case '-': case '\r':
        result->operation = pressedKey;
        result->number = negative ? -number : number; // if user enters an
        operation key without entering a number
        return;

        // if the user enter a number
        case '0': case '1': case '2': case '3': case '4': case '5': case '6':
        case '7': case '8': case '9':
        // if a number has not been entered already, then number = INT_MAX
        if( number == INT_MAX )
        {
          number = pressedKey - '0';
          lcd_print_int_neg(negative, number);
        }
        // if a number has been entered already
        else
        {
          number = number*10 + (pressedKey - '0');
          lcd_print_int_neg(negative, number);
        }
        break;
```

Figure 9: The solution for Lab 3: Entering and Displaying Numbers (Part 1)

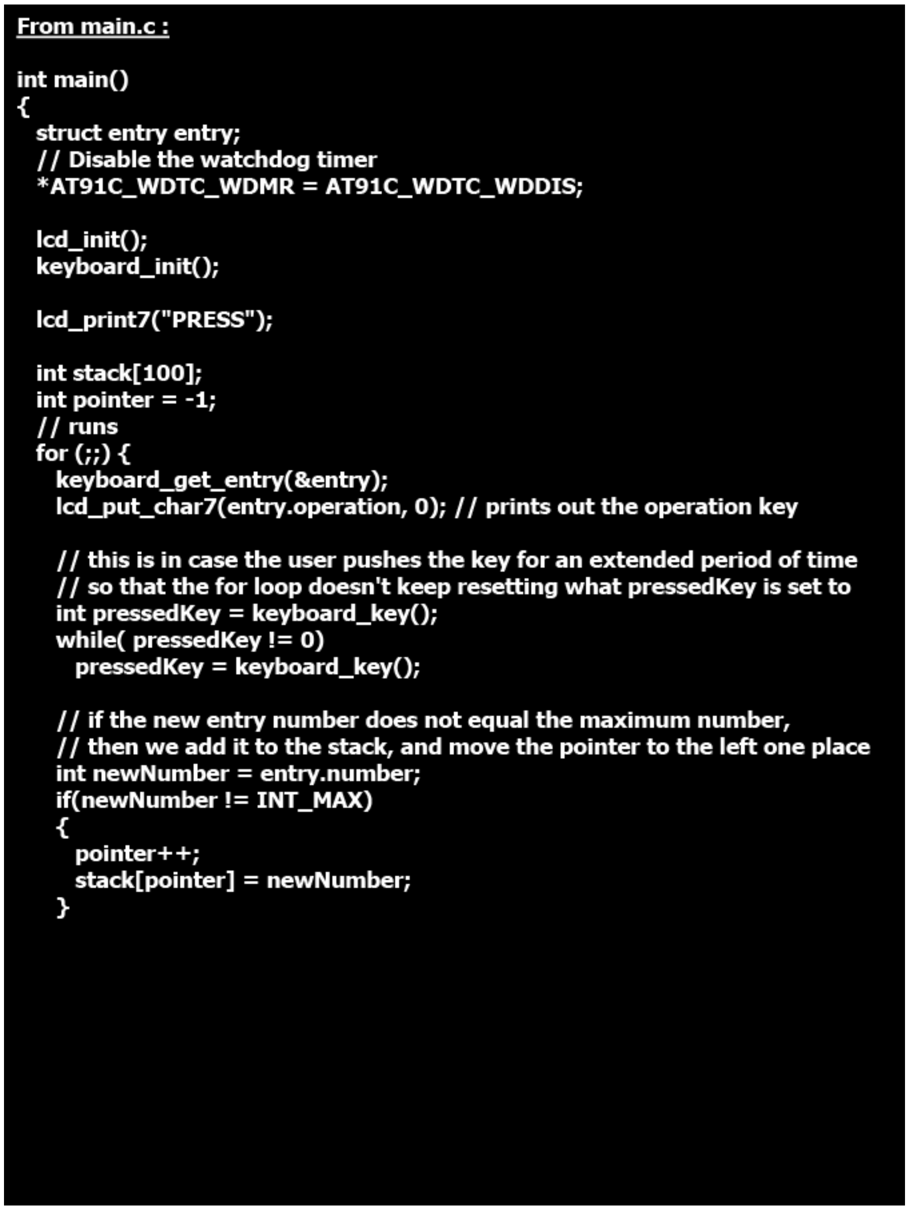


```
...Continued...


        // if the user wants to make the number negative
        case '~':
        negative = !negative;
        number = 0;
        break;

        // if the user wants to erase the last character inputted
        case '\b':
        number /= 10;
        break;

      }
    }

    // this is in case the user pushes the key for an extended period of time
    // so that the if statement doesn't keep resetting what number is set to
    while( pressedKey != -1)
    pressedKey = keyboard_key();

  }
}
```

Figure 10: The solution for Lab 3: Entering and Displaying Numbers (Part 2)

5.4 Lab 4: An RPN Calculator

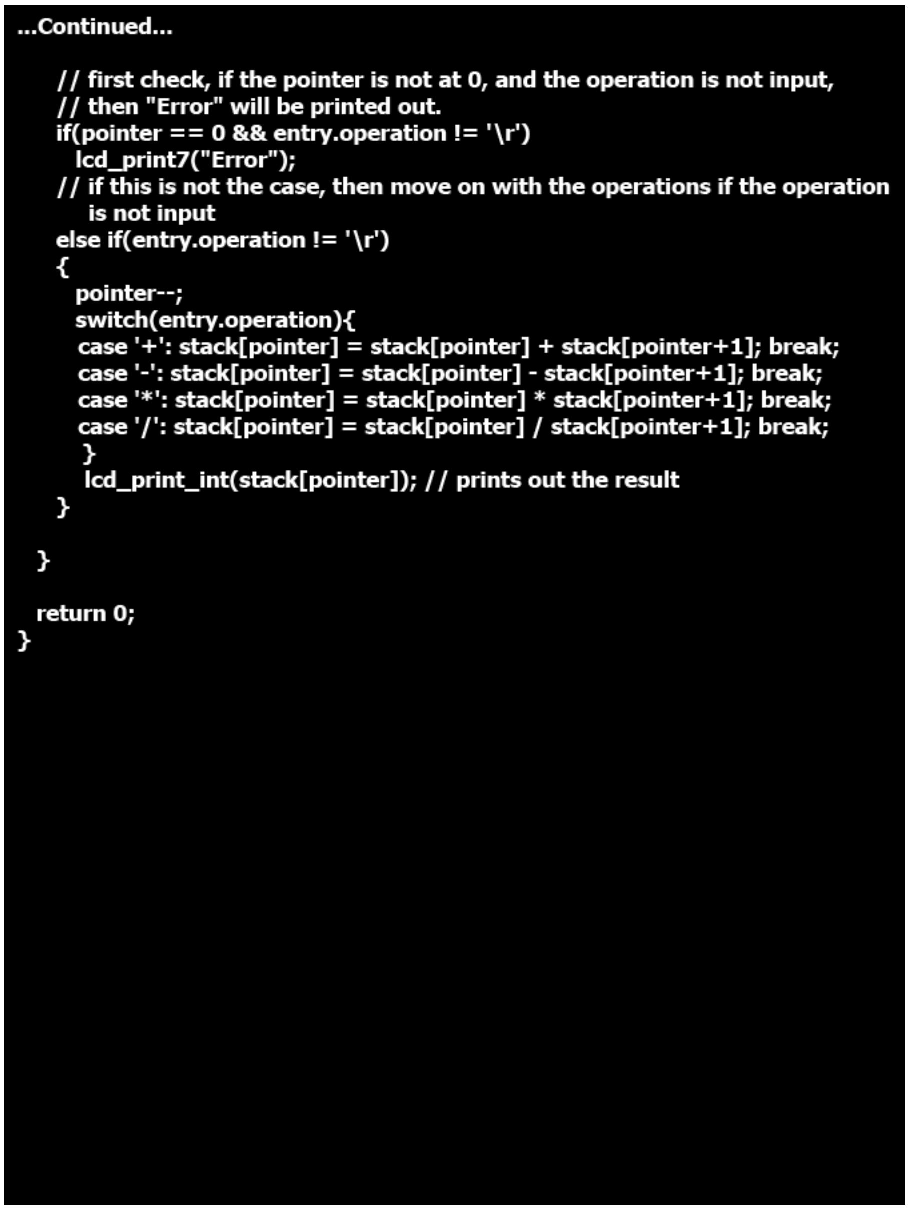
The final portion of our software building endeavors allows users to perform calculations using reverse Polish notation. This required implementing an array stack to hold onto various operands being manipulated by users.



From main.c :

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

  lcd_init();
  keyboard_init();

  lcd_print7("PRESS");

  int stack[100];
  int pointer = -1;
  // runs
  for (;;) {
    keyboard_get_entry(&entry);
    lcd_put_char7(entry.operation, 0); // prints out the operation key

    // this is in case the user pushes the key for an extended period of time
    // so that the for loop doesn't keep resetting what pressedKey is set to
    int pressedKey = keyboard_key();
    while( pressedKey != 0)
      pressedKey = keyboard_key();

    // if the new entry number does not equal the maximum number,
    // then we add it to the stack, and move the pointer to the left one place
    int newNumber = entry.number;
    if(newNumber != INT_MAX)
    {
      pointer++;
      stack[pointer] = newNumber;
    }
```

Figure 11: The solution for Lab 4: An RPN Calculator (Part 1)



```
...Continued...

    // first check, if the pointer is not at 0, and the operation is not input,
    // then "Error" will be printed out.
    if(pointer == 0 && entry.operation != '\r')
      lcd_print7("Error");
    // if this is not the case, then move on with the operations if the operation
        is not input
    else if(entry.operation != '\r')
    {
      pointer--;
      switch(entry.operation){
      case '+': stack[pointer] = stack[pointer] + stack[pointer+1]; break;
      case '-': stack[pointer] = stack[pointer] - stack[pointer+1]; break;
      case '*': stack[pointer] = stack[pointer] * stack[pointer+1]; break;
      case '/': stack[pointer] = stack[pointer] / stack[pointer+1]; break;
       }
       lcd_print_int(stack[pointer]); // prints out the result
    }

  }

  return 0;
}
```

Figure 12: The solution for Lab 4: An RPN Calculator (Part 2)

6 Lessons Learned

We learned a great deal about how to edit prewritten code to accomplish a given goal. It seems that in the professional world, a lot of work like this will be necessary. Accuracy and efficiency were key, and we were able to get a first-hand experience at dealing with buggy code. Bugs are not good! It is important to consider all possible scenarios and error-prone cases when writing code.

We advise that all future students fully understand the existing code and hardware system before beginning to write their own code. This will make the coding process less difficult, and it will improve the overall learning experience.

7 Criticism of the Course

The course assumes that everyone knows computer programming. Acknowledging that it is difficult to appeal to students with varying levels of experience, perhaps there could be some more instruction on programming for less experienced students. We would have also appreciated an intro to C++ before we began the first lab, because even the more skilled students were mostly only trained in Java. The code reviews were also helpful, and it would be nice to do some more of this for future classes. Overall, the class was a great learning experience, and we enjoyed doing the hands-on work.