ENGI E1112 Departmental Project Report: Computer Science/Computer Engineering

Gwen Pfetsch, Ivy Pan, Loren Weng

December, 2012

Abstract

The HP20b Business Consultant is a financial calculator with an open source code; the hardware as well as the software has been made public. The project centered on repurposing the functionalities of the calculator so that it can behave like a RPN (Reverse Polish Notation) calculator. Knowledge of the calculators hardware, which included its processor, the LCD display screen, and the keyboard were analyzed in order to better code the functionalities of the calculator. Using C programming language and through a series of labs, the team moved closer to making the calculator behave like that of a normal business calculator and then, a RPN calculator. With a library of usable functions to build on, the labs involved entering characters and waking up the display screen, to manipulating a keyboard, to finally entering and processing keypads and operations. Although the RPN calculator was never completely achieved, the calculator is able to take one input of a variable and the user is able to manipulate that variable with a variety of operations.

1 Introduction

Hewlett-Packard, who published the HP20b Business Consultant, publicized the source code as well as the hardware makeup, making it possible for users to reprogram and manipulate the calculator functionalities.

In order to repurpose the calculator, communication between the calculator and a computer must be established. Professor Stephen Edwards of the course has provided the team with the means of communication through a JTAG port, which is a serial pad built onto the back of the HP20b that allowed a connector to be soldered on. The USB-connect was able to transmit information between the calculator and the computer.

Through a series of lab and using C programming language, the HP20b gradually behaved more and more like a calculator with a working display screen, recognizable key presses, and (limited) functioning operations. Professor Stephen Edwards had initially provided the team with a library of working functions that the team later built upon and modified, such as lcd_put_char7, which allowed us to put a character on the screen. By building and modifying code from previous labs, each step made the calculator regain some of its original functionalities.

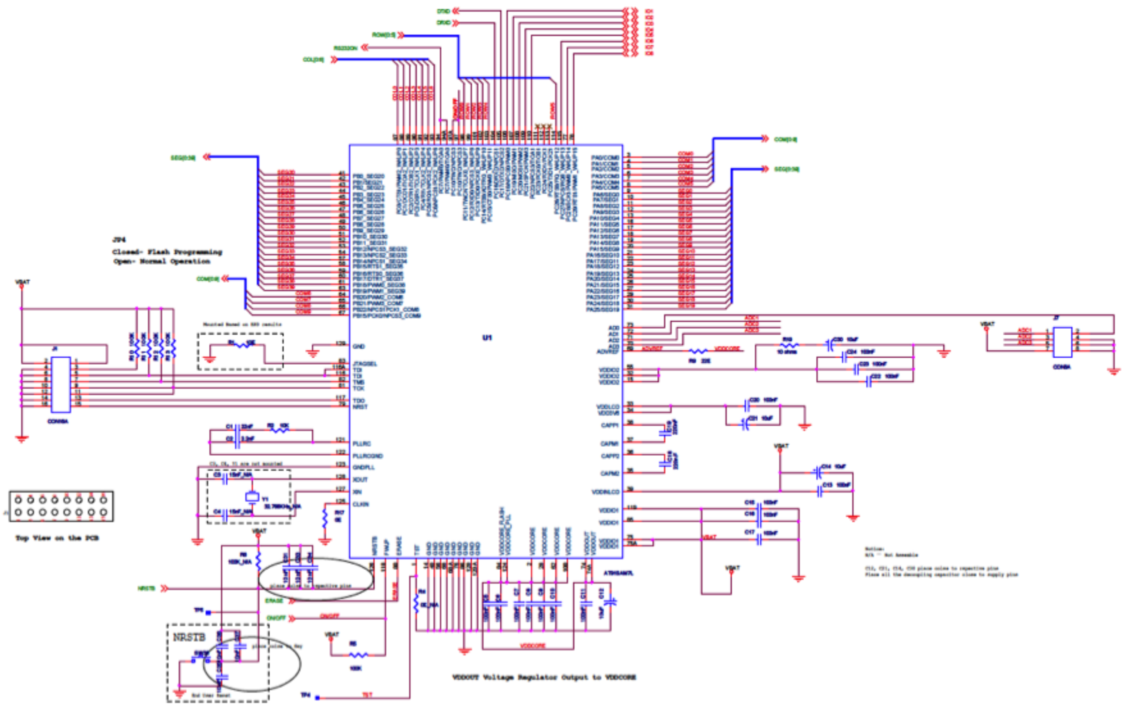
2 User Guide

The end result of the project was to make the calculator behave like an RPN calculator; although the team did not reach that result, the calculator is able to take a single input variable and the user can performs operations (+, -, *, /) on that variable. The RPN functions works by placing numbers into storage units called stacks. There are four levels in an RPN stack. When the user presses a key with an integer value, the value is stored in the first level. In order to perform an operation on that integer, the user would proceed to enter another integer, which would then be placed in the first level of the stack and the previous integer would move up one level. The user then presses an operation key and the operation is performed. Rather than pressing 2 * 3 = 6, RPN works by inputting 2 3 *. The resultant is then placed in the first level of the stack and the other two values are popped out.

However, our calculator functions more like a normal business calculator that is only limited to one variable. The user can take the variable, 4, and perform operations such as (+, -, *, /) by first pressing the 4, the operation key, and then the second integer. The team was able to implement the functions by following a series of labs. The first lab allowed the user to enter an input and have it displayed on the screen. The second lab involved waking up the keyboard of the calculator so that it recognizes the location of key presses and assigning each key on the keyboard with a character. The third lab included modifying and building on the code of the previous two labs so that the user could perform operations on a variable. The last lab, although incomplete, sought to increase the complexity of the HP20b so that it could behave like an RPN calculator.

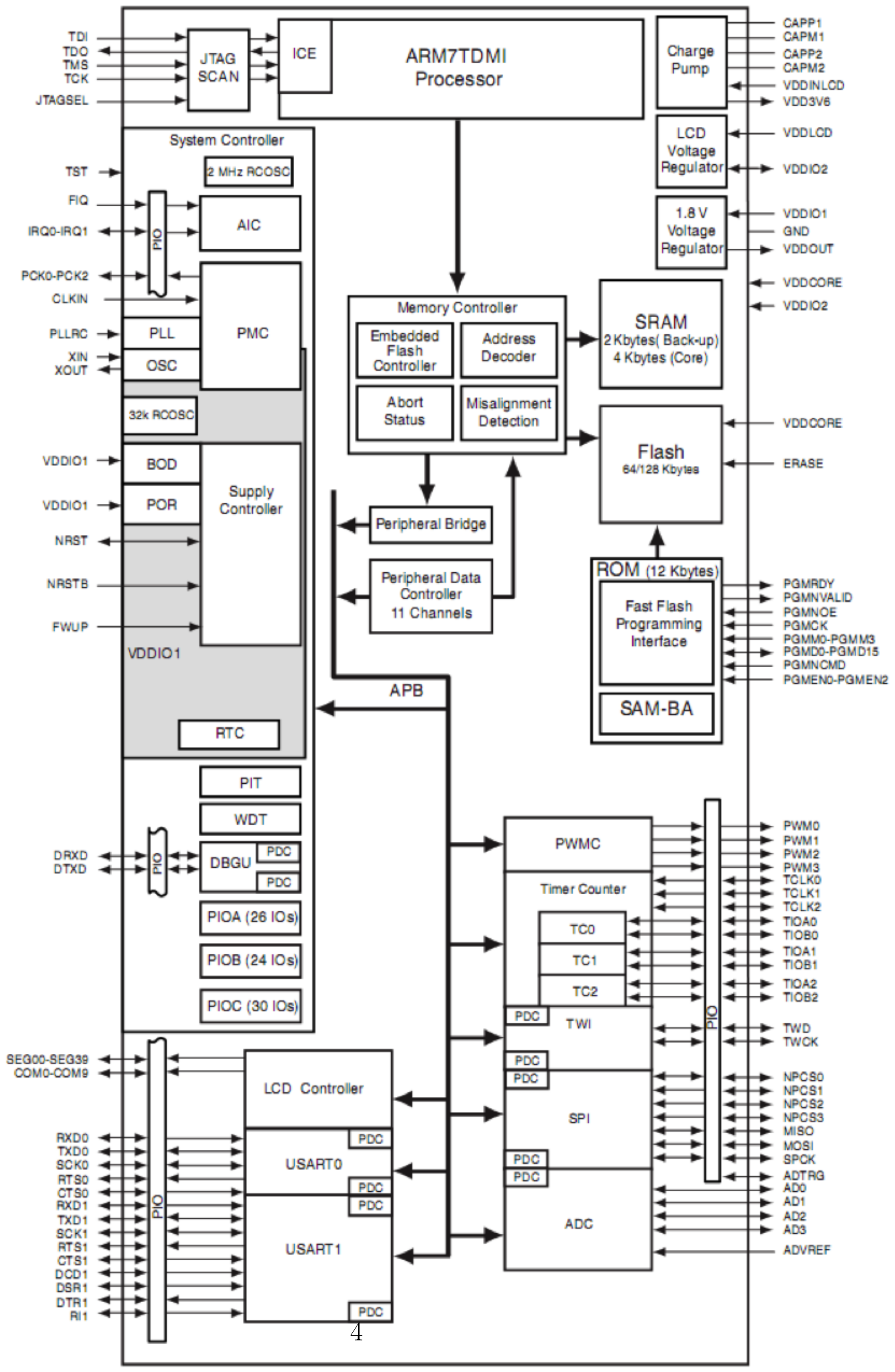
3 The Platform

The calculators logic is handled by an Atmel ATmega 91SAM7L128 microcontroller clocked at 30 MHz. The SAM7L128 is a surface-mounted, 144 pin, chip with 128 kB of onboard flash memory. The chip itself is designed around 32-bit Advanced RISC Machines (ARM) architecture, a popular industry standard. Below is a complete schematic of the 20bs pinout:

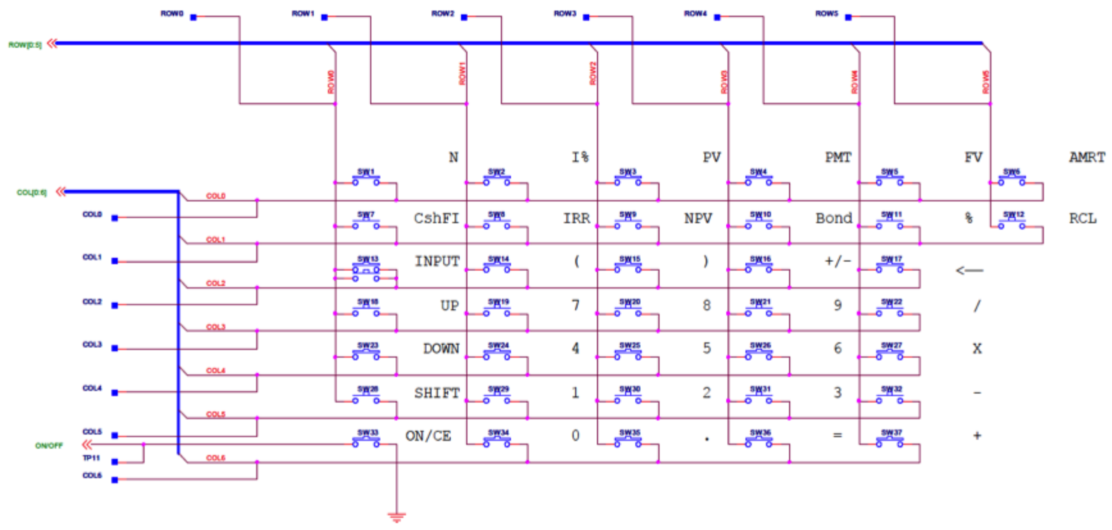


This seemingly complicated circuit diagram can be simplified into a slightly easier to interperate function block diagram found below:

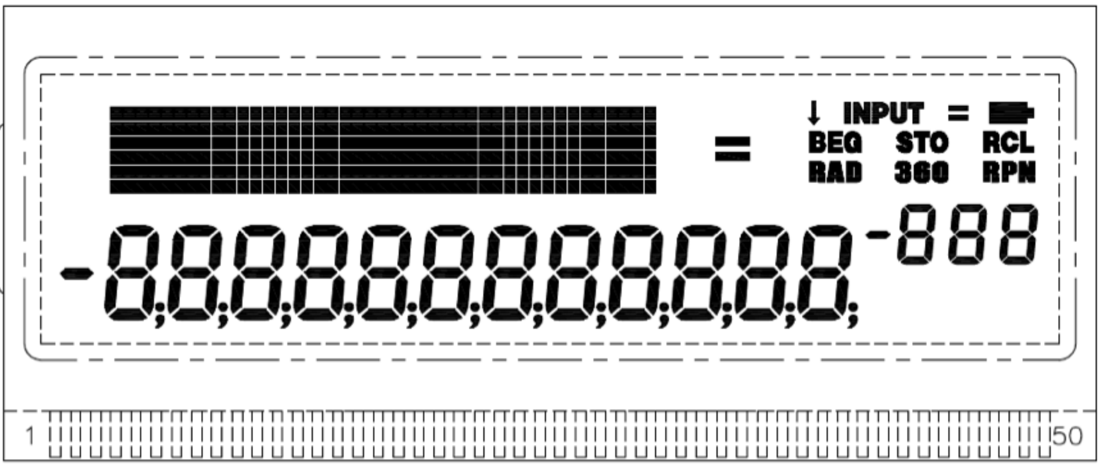
AT91SAM7L128 Block Diagram



The SAM7L128 receives its input from the calculators 37 tactile keys. Each key consists of a momentary push button connected to an input matrix of wires. When the keys are unpressed, the keyboard column is set to high. When a key is pressed, the inputs corresponding to the buttons row and column are shorted, sending a voltage through the wires and allowing the microcontroller to identify which key was pressed by running through each row and checking for low inputs. Below is a circuit diagram of the keyboard matrix:



Using input from the keyboard, the calculator can then determine the correction action to take and displays any feedback on its 400 pixel LCD. The screen itself contains 2 lines of display: the first line includes an 8-character scrolling display as well as 11 indicators to indicate for functions such as low battery of showing the unit measurement in degrees or radians; the second line includes a 12+3 digit 7-segment display with spaces in between for commas and decimals.



The LCD is directly controlled by the SAM7L128 using all 40 proprietary LCD segment controller pins and an additional 10 general communication pins. The provided keyboard library allowed us to easily display desired information without having to individually manipulate each pin. All of this repurposing is made possible by the fact that the 20b PCB (Printed Circuit Board) provides access to both JTAG and RS-232 (serial) pads. By soldering header pins to the JTAG interface we are given direct access to the microcontroller allowing us to reprogram the onboard flash memory.

4 Software Architecture

The software architecture is made up of the Assembly Library, the LCD Library, the Keyboard Library, and the Programming level. The Assembly Library conveys instructions to the computer via the processor. The LCD is the system that displays strings and characters onto the screen. The Keyboard library is the architecture that we use to determine what key is pressed and convert that to a character that will be displayed on the screen and manipulated by the calculator program. Finally, our program will implement these libraries and interact with the user to make a working calculator.

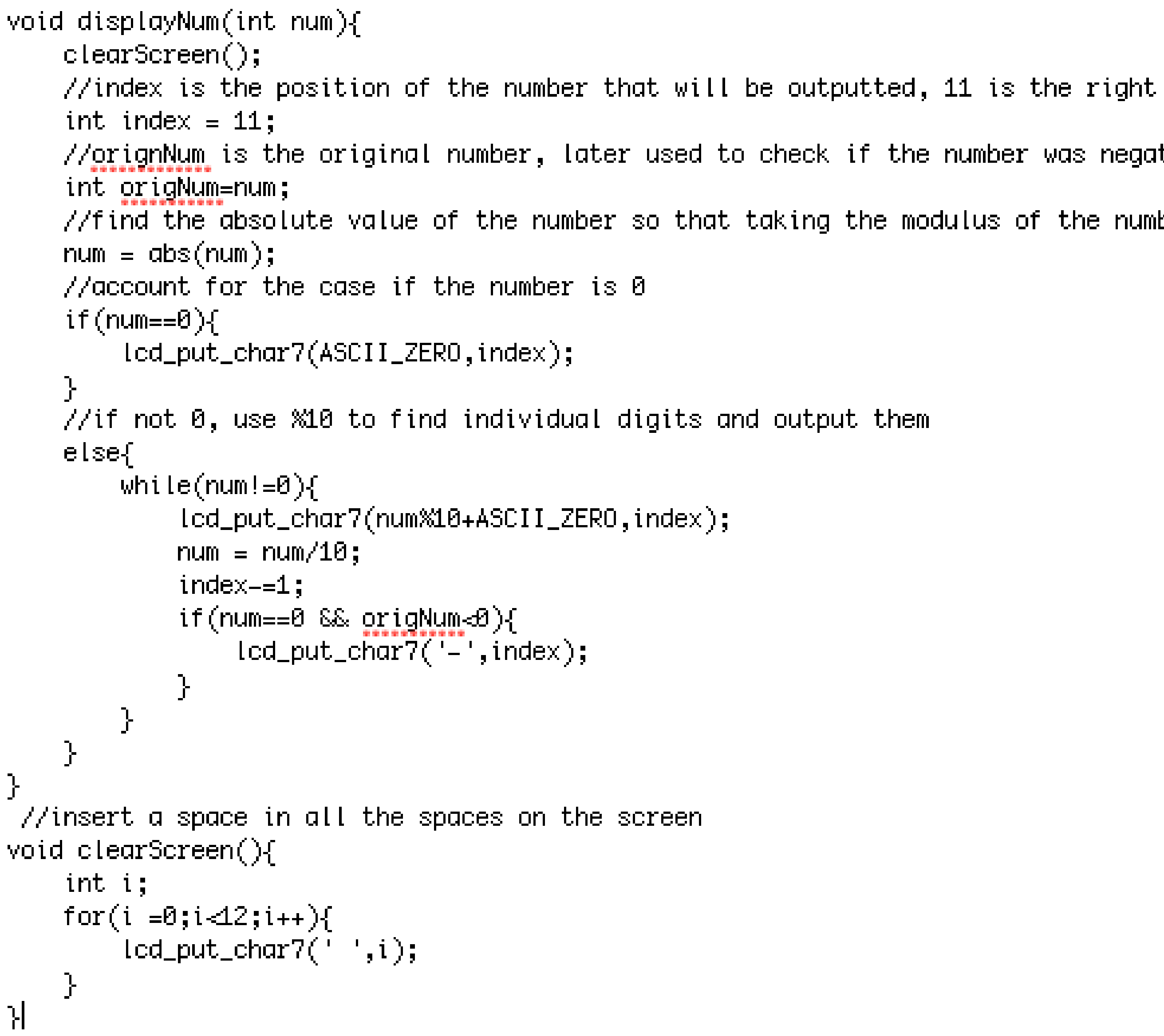
5 Software Details

5.1 Lab 1: Hello World

In this lab we initialized the Hello World program. To do this we edited Hello.c to display numbers. In the main method you set number to a number of type integer that you want to display. Then the main method runs displayNum(number). In the method displayNum(int num), it first runs clearScreen(), another method that uses a for loop to go through every space on the display and uses lcd_put_char7(, i); to put a blank space in each slot. After this clears the screen, the variable index is set to 11, which is the right of the screen. The program will put digits in right to left. A new variable origNum is made and set equal to the input number num, because num will be manipulated. num becomes the absolute value of itself so that we can take the modulus of it. An if statement is included so that if the number is 0, ASCII_ZERO, which is a constant set at 48 because this is the ASCII value of zero, is put in the index and we can end the method here. However if the number is not zero, we use a while loop. The modulus of 10 of the number is taken and put at the right-most index. Then the number becomes itself divided by 10, and the index moves to the left. This continues until the number is determined to equal 0. Because num is an integer, it will be identified as 0 when it is a decimal value less than 1. Before the while loop ends, an if statement will be read and if the original number was a negative value, a negative sign will be placed to the left of the number on the screen. This method makes use of the lcd command lcd_put_char7(). main.c:

```
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "math.h"
#define ASCII_ZERO 48

void displayNum(int);
void clearScreen();

int main()
{
  lcd_init();
    //num is the number to be displayed
    //max num = 2147483647, min num = -2147483647, limitation of C
    int number = 10000;
    displayNum(number);

}
```

keyboard.c:



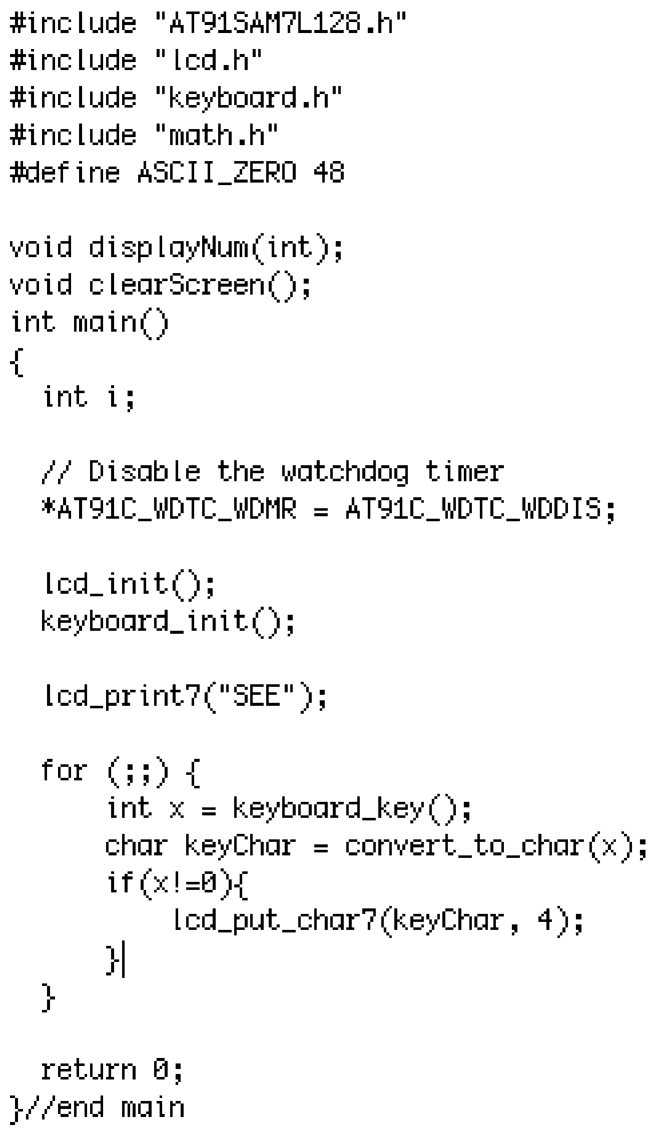
```
void displayNum(int num){
    clearScreen();
    //index is the position of the number that will be outputted, 11 is the right
    int index = 11;
    //orignNum is the original number, later used to check if the number was negat
    int origNum=num;
    //find the absolute value of the number so that taking the modulus of the numb
    num = abs(num);
    //account for the case if the number is 0
    if(num==0){
        lcd_put_char7(ASCII_ZERO,index);
    }
    //if not 0, use %10 to find individual digits and output them
    else{
        while(num!=0){
            lcd_put_char7(num%10+ASCII_ZERO,index);
            num = num/10;
            index-=1;
            if(num==0 && origNum<0){
                lcd_put_char7('-',index);
            }
        }
    }
}
 //insert a space in all the spaces on the screen
void clearScreen(){
    int i;
    for(i =0;i<12;i++){
        lcd_put_char7(' ',i);
    }
}
```

5.2 Lab 2: Listening to the Keyboard

To display numbers that are pressed by keys, we needed to write modifications to keyboard.c and main.c. First, we had to initialize all keys as being high so that it can detect when a key is pressed. This was done by the method all_columns_high() which uses a for loop to go through each column and implements the method all_keys_high() given to us in each column. Then we created the method keyboard_key() which returned a number that represented which key was being pressed. The method detected which key was being pressed with nested for loops. The outer for loop goes through each column. First all columns are reset to high with the method all textunderscore columns_high(). Then one column was lowered using the given method keyboard_column_low(i). Then it goes through each row to see if the key being pressed is in the intersection of the lowered row and the column it is reading. If it is, there is an if statement that returns the number and exits the loop. If this if statement is never executed, 0 is returned because this is what num is initialized as. This would mean that no key is being pressed. For a nonzero number, the number is calculated by the equation num = ((i+1)*10) + (j+1), where i would be the column of the pressed key and j is the row. The last method that we implemented was convert_to_char(int tempNum), which is called in main.c. This converts a number to a character using a key map that we constructed. This map is a list of arrays that mimics the keyboard. Using the number that was returned in keyboard_key(), this method determines its column by dividing it by 10 and subtracting 1, and its row by finding its modulus of 10 and subtracting 1.

```
void all_columns_high(){
    int i;
    for(i =0;i<7;i++){
        keyboard_column_high(i);
    }
}

int keyboard_key(){
    int i;int j;int num=0;
    for(i =0;i<7;i++){//columns
        all_columns_high();
        keyboard_column_low(i);
        for(j =0;j<6;j++){//rows
            if (!keyboard_row_read(j)){
                num=((i+1)*10)+(j+1);
                return num;
            }
        }
    }
    return num;
}
char convert_to_char(int tempNum){
    int column = tempNum/10-1;
    int row = tempNum%10-1;
    char keyMap[7][6]={{'N','i','p','P','F','A'},{'C','I','n','B','%','R'},
{'T','T','(',')','M','<'},{'^','7','8','9','/',' '},{'v','4','5','6','x',' '},
{'s','1','2','3','-',' '},{'f','0','.','=','+',' '}};
    return keyMap[column][row];
}
```

In the main method, we added an infinite for loop. A variable x is set to the value returned by keyboard_key() and then convert_to_char(x) is run on x. Unless x is 0, in which case no key was pressed, we use lcd_put_char7() to dis-



```
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"
#include "math.h"
#define ASCII_ZERO 48

void displayNum(int);
void clearScreen();
int main()
{
  int i;

  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init();

  lcd_print7("SEE");

  for (;;) {
      int x = keyboard_key();
      char keyChar = convert_to_char(x);
      if(x!=0){
          lcd_put_char7(keyChar, 4);
      }
  }

  return 0;
}//end main
```

play the character.

5.3 Lab 3: Entering and Displaying Numbers

In this lab we started to get the calculator actually working. Because we never got to doing stacks, our calculator works by having one number that we could manipulate with another by doing operations on the original. In keyboard.c, we kept everything from lab 2 and added a few more methods. One such method is keyIsOperator(), which acts as a Boolean to test if the key pressed is an operator. We also have keyIsInt() which returns true if the key is an integer. If the key is an integer, keyToInt() is run which converts the value of key to its integer value as it corresponds to the keyboard. The handler generates the stored value that is being operated on. The method starts with an if statement checking doSomething, which is set to true in another method if an operator is pressed. A few if else statements follow adjusting the value of finalInt for the case of the operator being an addition sign, subtraction, multiplication, division, or equal sign. It then returns finalInt. keyboardget_entry() is used whenever a key is pressed that will manipulate the stored value. It runs keyboard_key() and, if a key is pressed, checks if it is an operator or integer and assigns the correct character value. If the key is an operator, Boolean whatHappen is set to false. The else statement in this method is if no key was pressed. If this is run and whatHappen is true, then doSomething is set to false.

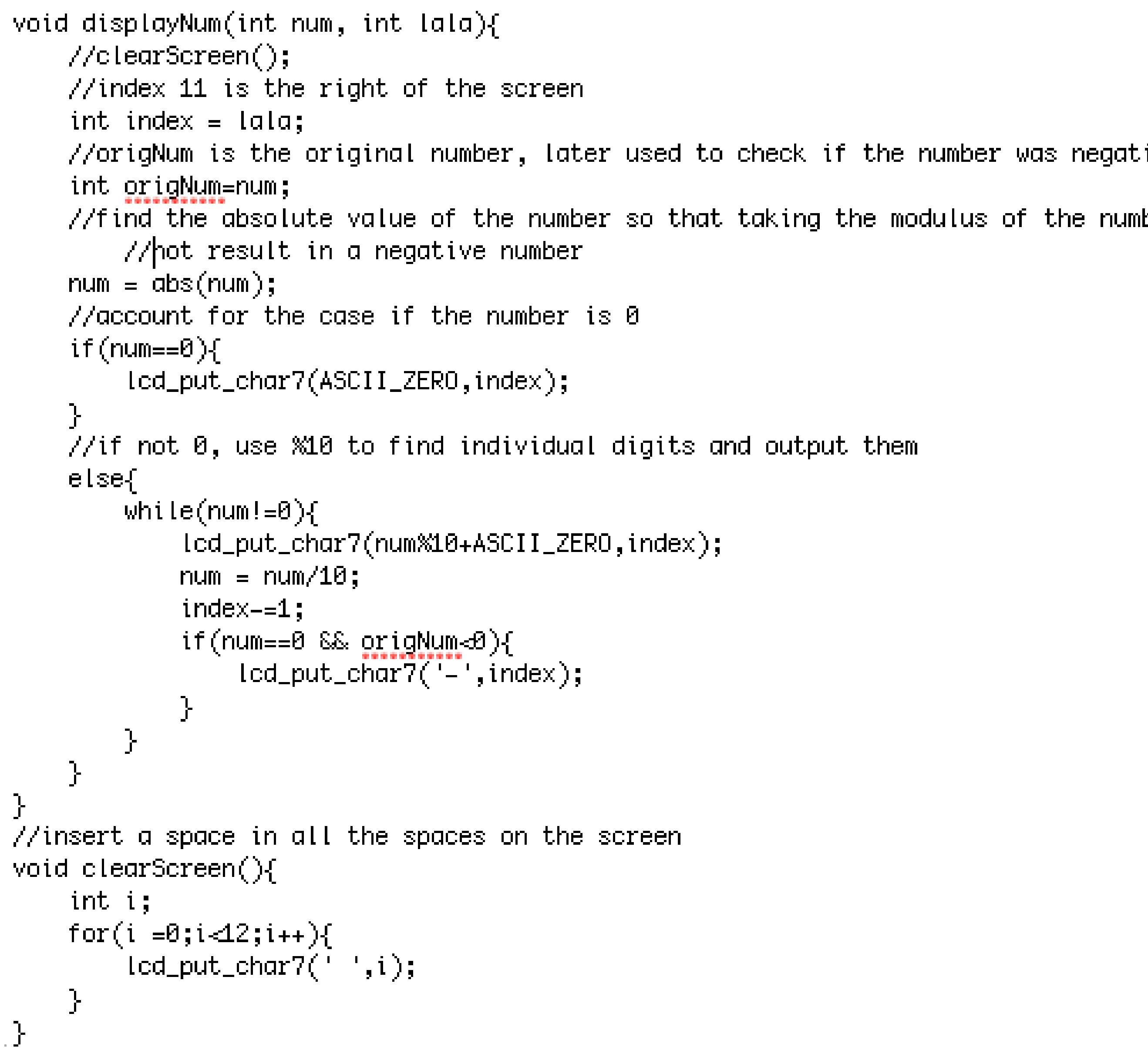
```
#include "AT91SAM7L128.h"

#define KEYBOARD_COLUMNS 0x7f
#define KEYBOARD_ROWS 0x400fc00
#define INT_MAX 2147483647

const unsigned char keyboard_row_index[] = {11,12,13,14,15,26};
int tempInt=0;
char tempChar='+';
int resultInt=0;
char resultChar='+';
int doSomething = 0;
int whatHappen = 0;

int finalInt= 0;

/*struct entry{
    char operation='';
    int number=0;
}*/

void keyboard_init()
{
    tempInt=0;
    resultInt =0;
    finalInt =0;
    resultChar = '+';
    tempChar = '+';
    doSomething =0;
  // 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

}
```

```
int handler(){
    /*
     if(addStack){
     addStack = 0;
     stack.add(resultInt);
     }
     */
    if(doSomething){
        doSomething =0;
        tempInt = 0;
//code if time to use stacks; we didn't get to that point
        /*
         tempInt2=stack.pop();
         tempInt1 = stack.pop();
         if(resultChar =='+'){
            stack.add(tempInt1+tempInt2);
         }
         else if(resultChar =='-'){
            stack.add(tempInt1-tempInt2);
         }
         else if(resultChar =='*'){
            stack.add(tempInt1*tempInt2);
         }
         else if(resultChar =='/'){
            stack.add(tempInt1/tempInt2);
         }
         */
        if(resultChar =='+'){
            finalInt +=resultInt;
        }
        else if(resultChar =='-'){
            finalInt -=resultInt;
        }
        else if(resultChar =='*'){
            finalInt =resultInt*finalInt;
        }
        else if(resultChar =='/'){
            finalInt = 1000;//(int)(((double)finalInt)/resultInt);
        }
        else if(resultChar =='='){
            finalInt = 1000;//(int)(((double)finalInt)/resultInt);
        }

        resultInt = 0;
        return finalInt;//stack.peek();
    }
    return finalInt;
}
```

```
int keyboard_get_entry(){
    int getKey = keyboard_key();
    if(getKey !=0){
        if(keyIsInt(getKey) && tempInt == resultInt){
            tempInt*=10;
            tempInt +=keyToInt(getKey);
        }
        else if(keyIsOperator(getKey)){
            tempChar = convert_to_char(getKey);
            whatHappen = 1;
        }
        /*else if(keyIsENTER(getKey)){
            addStack =1;
        }*/
    }
    else{
        resultInt = tempInt;
        //if(finalInt ==0){finalInt = resultInt;}
        resultChar = tempChar;
        if(whatHappen){doSomething =1;whatHappen=0;}
    }
    return resultInt;
}
```

In main.c we first changed the infinite for loop. There are two values x and z; z is the stored value and x is the value that changes z. Display num is first run for z, the display number, then for x.

```c
#include "AT91SAM7L128.h"
#include "lcd.h"
#include "keyboard.h"
#include "math.h"
#define ASCII_ZERO 48

void displayNum(int, int);
void clearScreen();
int main()
{
  int i;

  // Disable the watchdog timer
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;

  lcd_init();
  keyboard_init();

  lcd_print7("SEE");//test print

    int y=0;
  for (;;) {
      int x = keyboard_get_entry();//temporary value to perform operation on z
      int z = handler();//stored variable to represent current number

      displayNum(z,5);
      if(x!=0){
          if(y!=x){
              y=x;
              displayNum(x,9);
          }//prevents from displaying 0

      }
  }

  return 0;
}
```



```
void displayNum(int num, int lala){
    //clearScreen();
    //index 11 is the right of the screen
    int index = lala;
    //origNum is the original number, later used to check if the number was negati
    int origNum=num;
    //find the absolute value of the number so that taking the modulus of the numb
        //not result in a negative number
    num = abs(num);
    //account for the case if the number is 0
    if(num==0){
        lcd_put_char7(ASCII_ZERO,index);
    }
    //if not 0, use %10 to find individual digits and output them
    else{
        while(num!=0){
            lcd_put_char7(num%10+ASCII_ZERO,index);
            num = num/10;
            index-=1;
            if(num==0 && origNum<0){
                lcd_put_char7('-',index);
            }
        }
    }
}
//insert a space in all the spaces on the screen
void clearScreen(){
    int i;
    for(i =0;i<12;i++){
        lcd_put_char7(' ',i);
    }
}
```

6 Lessons Learned

This lab was a great experience for us. As aspiring engineers we were able to see how to implement a program that has a real purpose. All of us are currently studying computer programming, and to have this project while studying Java or Python was interesting, because we could compare languages and see a good application of our programming skills. Group work was also a learning process in this lab. We had to split up the work so that everyone was working easily, so those who didn't necessarily have a huge part in writing the code would in turn work on the presentation, hardware, etc.

7 Criticism of the Course

A criticism we have is the difficulty of the project. Because none of us could code in C we had a hard time with the learning curve. We weren't able to finish the last lab as well. It was also hard to complete a lab of this magnitude in a group of four only meeting once a week. We also had a group member drop out of the course, and so the inconsistency of a group member not being there was tough. Otherwise we enjoyed the lab. It was a good, although difficult, introduction and we all enjoyed learning matlab as well.