ENGI E1112 Departmental Project Report: Computer Science/Computer Engineering

Noah Stebbins, Nicholas Sun, Ga Young Lee, Luis Ramirez December, 2012

Abstract

For Art of Engineering, our group wrote firmware for an HP 20b calculator, allowing it to do more than display simple strings of characters, but also run operations and carry out commands (e.g. waiting for the user to enter a long number before returning a value). Through a series of labs, we managed to go from modifying display to defining algorithms that would define exactly what the calculator did. As we went through the labs, we developed a better understanding of the C programming language. We went from changing simple parameters, to defining methods, and finally to working with structs and pointers.

We programmed the HP 20b calculator to have a broader understanding of Computer Science as a whole. While many classes are oriented at teaching students simple syntax of a programming language, Art of Engineering is actually geared towards implementing what you learn in society. While we did not directly impact society in any way, we did develop comprehension of C by applying it to an electronic device. Thus, we took our first step as an engineer into building applications.

We built the HP 20b calculator with C through a series of four laboratory experiments. For the first one we simply modified display based on an integer argument. In the second one, we defined a method by which the calculator could display a specific key that was being pressed. In the third experiment we found a way for the calculator to store a number and an operation into a struct, which could then be used to perform operations. Finally, in the fourth experiment, we combined the results of the first three labs to create a fully functional RPN calculator.

1 Introduction

For our Art of Engineering class at Columbia University, we were supposed to program an RPN calculator using C to get it to perform simple operations for the user. While most of the code was already written for us, the “display” modification part of the code was up to us to implement. Thus, the actual code allowing us to interact with the calculator’s processor was not altered by us, but the desired behavior of the calculator was.

C was the language of choice, because it is a standard language for most electronic devices. It is still widely used, and therefore, as an engineer, it is important to learn. C also requires a comprehensive understanding of proper syntax - one mistake can prove disastrous. Thus, it is good for beginning programmers be- cause it encourages good practices.

2 User Guide

The calculator’s display is rather standard and has been explained in the summary of the Platform below. The keyboard uses a process of analyzing highs and lows, which is also further explained.

The calculator is an RPN calculator, which stands for Reverse Polish Notation. The idea behind this is that you can represent anything mathematically with- out using parentheses and by saving keystrokes. The style of an RPN calculator, however, is not conventional.

When entering computations, you must put in the first numerical value and then press INPUT. Then you enter a second value and an operation. The result is you get an evaluated mathematical expression. Thus, putting in a 5, INPUT, 6, + will return a value of 11. This is counter-intuitive to us because we want to do 5, +, 6, =. However, this is actually more efficient.

For complex situations like evaluating (3 + 5)/(7 + 6), you would divide them into sub-expressions and then divide the two sub-expressions at the very end. So you would enter a sequence of commands identical to 3, INPUT, 5, +, 7, INPUT, 6, +, /. Notice that the division operator is at the end. Running this in your calculator will actually return the correct value of 0.6153846.

3 The Platform

The software used for the HP 20b calculator is based on entry-system logic, which contains RPN and Algebraic and Chain Algebraic software. There are over 220 built-in functions and menus and prompts are included.

3.1 The Processor

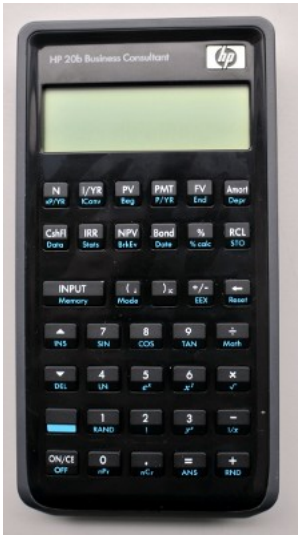

Figure 1: The HP 20b

The HP 20b is little more than a keyboard and liquid crystal display (LCD) connected to an Atmel AT91SAM7L128 processor. This mouthful of a name, which I will abbreviate to SAM7L, was given to it because it is part of Atmel's AT91SAM series of chips, which are all built around an ARM processor core ("AT" is for Atmel; "SAM" is "smart ARM core;" 91 appears to be arbitrary). The 7L series of microcontrollers are designed for low power (hence the

L), and the final 128 is a reminder that it includes 128K of flash program memory.

Figure 2 shows a block diagram of the SAM7L chip. It looks complicated, but is essentially a single standard processor surrounded by memory and a wide variety of peripherals, most of which we will not use. You should be aware of the system controller, which, through software, controls the clock and power supply of each peripheral. This makes it possible to save energy by not powering on unneeded peripherals, but can also make a peripheral appear not to work if you neglect to turn on its power.

3.2 The LCD Display

The LCD display of the calculator includes 1.5 lines by 12 characters and 3 exponent characters. There is a bottom line of 7 segments, and a scrolling top line of 8 characters.

The *lcd_put_char7* function is the foundation to display on the calculator. The function takes in two parameters, the first one a character and the second one a number. The function then returns the character at the given index numerical value. So putting 'H' and 0 as your two parameters would print H at the 0th value of the calculator. This means that you would have a calculator displaying H on the far left-hand side.

3.3 The Keyboard

The keyboard works as follows. Initially, all columns are set to a low voltage, and then the rows are tested to see if they have a low or high voltage. If a row is low as well then the column is set to high and the position at that specific row and column value is returned, allowing the user to see display for a corresponding character.

4 Software Architecture

I have several files in my lab that aid in producing the final result as a whole. The keyboard.h file is a template for the keyboard.c file, which is in charge of how things are displayed and how they are stored. The lcd.h file is a template for the lcd.c file, which is in charge of mandating the laws of display. It is thanks to this file that we can even see any output at all on our display. After all, the computer needs to instruct the calculator on how it will display its data.

The main.c file is responsible for executing everything else and linking to everything else. It is thanks to the main.c file that we have an interface by which we can easily see what part each file plays in the final product. The Makefile sets the rules for compilation, which we can then do with a simple MAKE FLASH in the terminal.

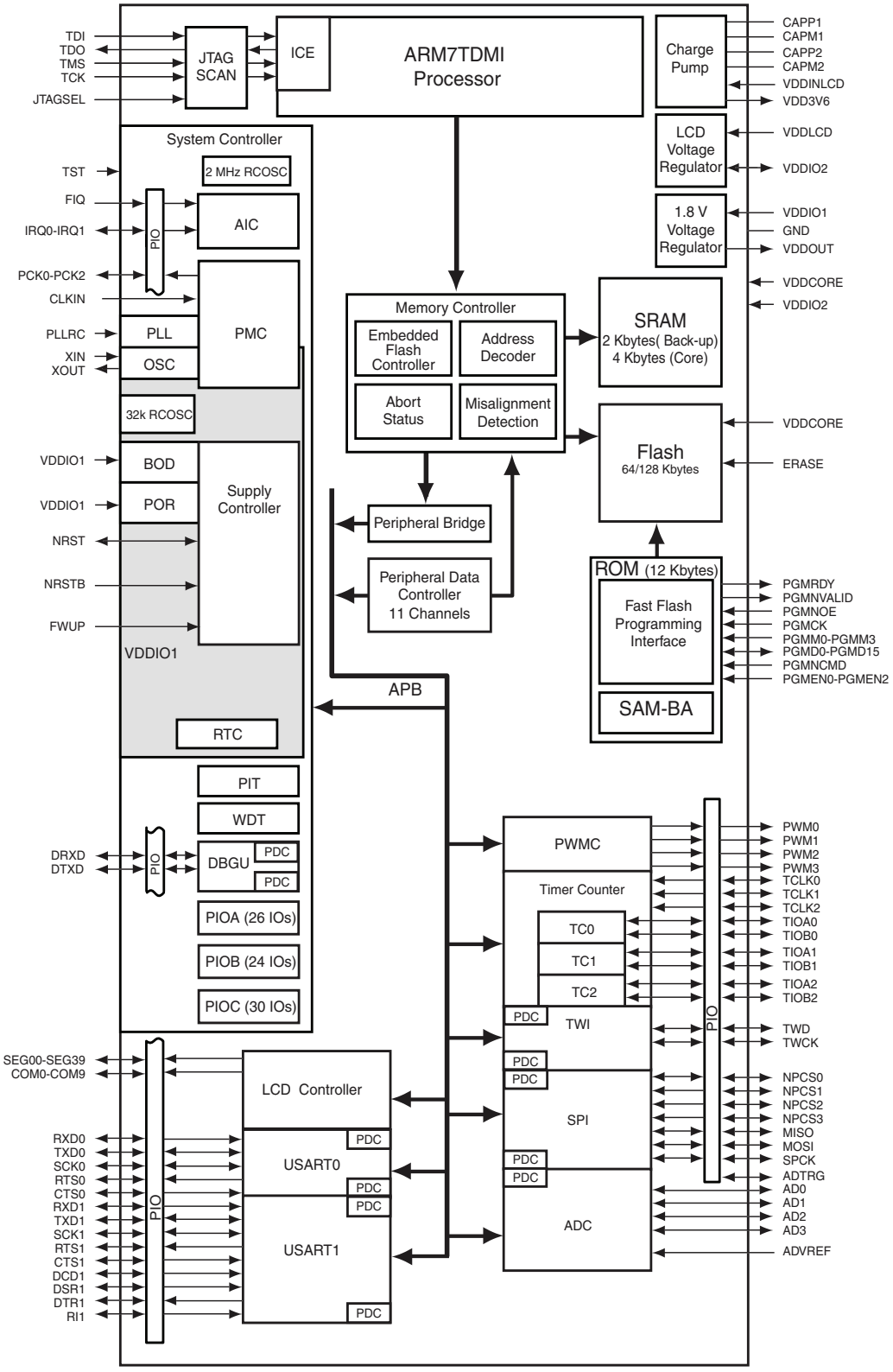

Figure 2: A block diagram of the AT91SAM7L microcontroller that is at the heart of the HP 20b

5 Software Details

In this section, I have included my group's best solutions to the questions posed by the labs and I will explain how they work, my motivations for them, whether I would do anything differently, etc.

5.1 Lab 1: A Scrolling Display

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

void clearScreen() {
    int i;
    for(i=0; i<11; i++){
        lcd_put_char7(' ',i);}
}

void printOutput(int number) {
    clearScreen();

    int counter, remain; char character;
    int position = 11;
    int absArg = abs(number);

    for(counter = absArg; counter >= 1; counter /= 10) {
        remain = counter % 10;
        character = '0' + remain;
        lcd_put_char7(character, position);
        position--;
    }

    if(number < 0) {
        lcd_put_char7('-', 0);
    }

    if(number == 0) {
        lcd_put_char7('0', 11);
    }
}

int main()
{
  lcd_init();

    printOutput(23444);
    printOutput(-0);

  return 0;
}
```

Goal: display a numerical argument in the calculator

Clear screen assigns empty spaces in every position in the screen. So it makes the screen ready to accept the number.
After cleaning up the screen the variable takes the absolute value of input number. In the for loop, the number is kept divided by 10 and the remainder is stored in the right most position until the computer gets the one digit value of the highest digit number. For example, when you enter 205, the first result from for loop is 5 and it is shown in the right most position 11. And the remaining number 20 goes through the same process and 0 in position 10 and 2 in position 9. In case when the input number is negative, the number goes through the same process because the code takes the absolute value of the number prior to the for loop.
To deal with the negative sign, lcd_put_char7('-', 0) displays '-' in the left most position. Also, if input is zero, the screen shows 0 in the rightmost position by lcd_put_char7('0', 11)

5.2 Lab 2: Scanning the Keyboard

```
char keyboard_key() {

int i; int rowPosition = 2; int colPosition = 5; // if nothing is pressed at that point

char keyboard[7][6] = {{'N', 'Y', 'V', 'P', 'F', 'A'}, {'C', 'R', 'K', 'B', '%', 'C'},

                       {'I', '(', ')', 'O', '<'},
                       {'^', '7', '8', '9', '%'},
                       {'v', '4', '5', '6', 'x'},
                       {'s', '1', '2', '3', '-'},
                       {' ', '0', '.', '=', '+'}};
   //set all columns to high

for (i = 0; i < 7; i++) { keyboard_column_high(i);}

int j, k;

for (j = 0; j < 7; j++) {

keyboard_column_low(j);

          for (k = 0; k < 6; k++) {

                    if (!keyboard_row_read(k)) {return keyboard[j][k]; }
```

```
            }
            keyboard_column_high(j);
    }
}
```

The figure above shows my code for lab 2. Keyboard_key() works as follows: you first loop through all of the columns and you set them all to high. Initially you assume the worst (i.e. that the value you are trying to access is not in the calculator's keyboard). Thus, the rowPosition is set to two and the colPosition is set to five. You then loop through all of the rows for each column. If the row is also low, then you have to return both the value of the row and the value of the column.

We then created a three-dimensional array of all of the possible input values on a calculator and we return the element in that three-dimensional array that has a row position and a column position equal to that of the two low values.

This function shown is invoked by main.c and is located inside of the keyboard.c file.

We tested it with a wide range of values. We saw what would happen if you kept pushing down the button (it would keep displaying) and we saw what would happen if you tried to hit two values (only one would be displayed). Thus, we accounted for error.

5.3 Lab 3: Entering and Displaying Numbers

NOTE*:* We did not have the chance to do a large portion of this assignment so we did as much as we could and explain it to our maximum ability

```
void keyboard_get_entry(struct entry *result)
{

   int entry=0;
          int negative = 0;

  for (;;) {
//make some loop that calls keyboard_key() to check if it is true so as to see if whether a key is being pressed.  While a key is pressed keep looping through and calling keyboard_key() until it is no longer being pressed, so once keyboard_key() is no longer true and then take that value.  Every instance of keyboard_key() should be replaced by a variable initialized within the loop

   if (keyboard_key() >= 0 && keyboard_key() <= 9)
   {
      entry= entry * 10 + (keyboard_key() – ‘0’);
             result->number = negative ? -entry : entry;
           }
//check the cases that can be pushed
   else if (keyboard_key() == ‘~’) //if the plus minus sign is pushed, make negative true.
{
              negative = 1;
       }
else if (keyboard_key() == '\r' || keyboard_key()  == '+' || keyboard_key()  == '-' || keyboard_key()  == '*' || keyboard_key() == '/') // if one of these are pressed exit and return what is in the pointer
    {

              result->operation = keyboard_key();

       return;

    }
```

lcd_print_int_neg(negative, entry);

}

The user is going to enter a string of numbers which create one large number by looping through some sort of display modification where the numbers stay on the screen and allow for the addition of other numbers

When you have finished typing the number that you desire, you enter an operation. the program recognizes this operation and it returns the function, storing the operation and the number in struct

These are passed to the main function where you can test that this works correctly by displaying the number and the operation

6 Lessons Learned

In this lab I learned how to not only program in C, but how to implement C on electronic devices, and that I feel, is the most important thing I learned. Anybody can write a hello program in C, but to actually use knowledge to construct a device is wholly different.

For future students, I advise that you work with people who work just as hard if not harder than you do, and make sure that everyone in the team knows their role in the group. If someone has extensive experience with C, have him explain to the other group members what he has learned so they are caught up-to-speed. Leave no group member behind. Have no group member do all of the work.

Honestly, I wish I knew in the beginning of class how to manage my time. College is all about project-like assignments and procrastination just does not work. The most important thing I feel like I have learned is to start projects on the day they are assigned. In the case of Art of Engineering, this means pushing the group forward on the first day you start working on the lab. Instead of

being lazy in class since Art of Engineering is on Friday, it would be wiser to just deal with the fact that you have class and start working.

7 Criticism of the Course

I liked the fact that we were learning how to apply computer science to an actual electronic device. Not only were we learning a new programming language, but we also had the chance to implement it on a calculator to define the behavior of the calculator.

I did not like the large leaps between the labs. In my opinion, the labs were too difficult for beginning C programming. We spent a majority of our time just trying to figure out what we were supposed to do, and when we finally understood what the lab directions were asking, we were behind.

In the future, I would recommend making more labs, but having them have smaller requests. For example, instead of creating a fully functional RPN calculator in four labs, you could make it six labs, but have each lab designed to complete in about a week. That way, presentations would be continuous and momentum for completing more work would be higher.

The code reviews were somewhat helpful. The feedback was very good and I learned a lot about minimizing code and making code more readable while maintaining overall function. I also learned about the balance of commenting a program thanks to feedback. Simpler labs would allow for simpler presentations.