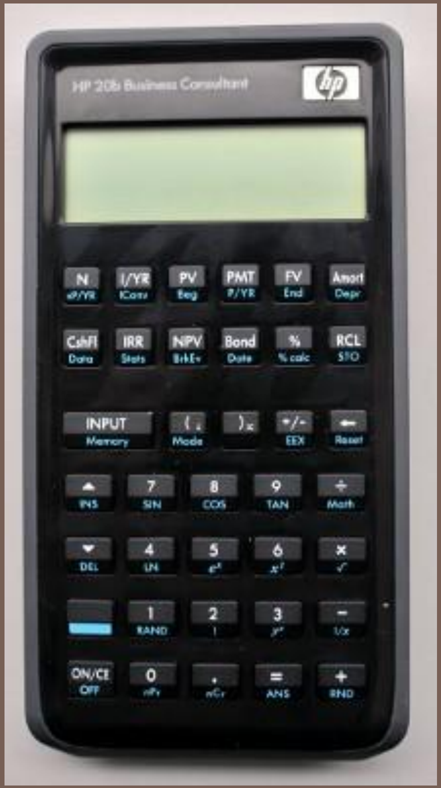
ENGE 1112 CS PROJECT

December, 2012



Arushi Gupta

Jake Kazimir

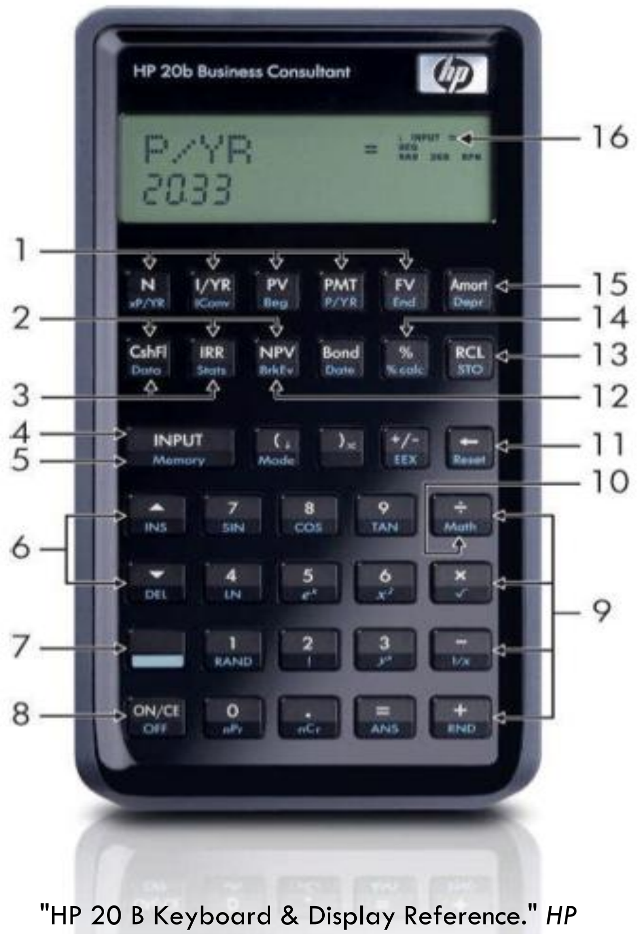
Serena Shah-Simpson

Overview

- We were given an HP 20b calculator, wiped clean of all firmware that related numbers entered on the keyboard to memory and display
- We wrote some code in C to reinstate these processes.
- This embedded programming instructed the calculator what to do under certain circumstances.

User Guide

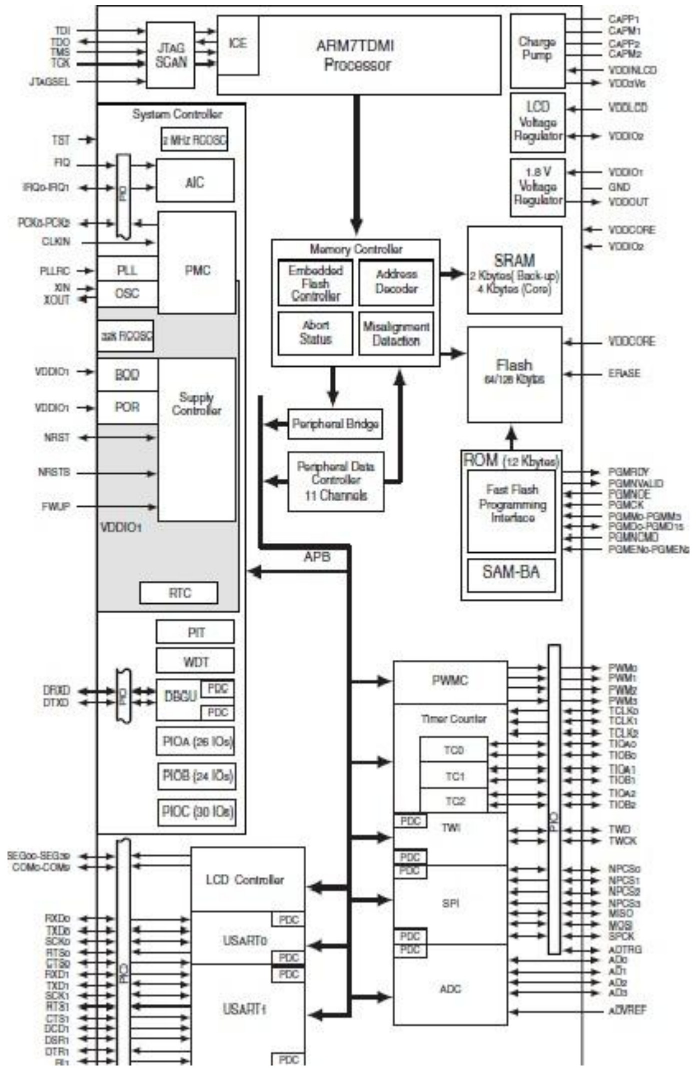
- Entering Numbers
- Clearing the Screen
- Performing operations



"HP 20 B Keyboard & Display Reference." *HP 20b Keyboard & Display Reference : HP Calculator : Educalc.net.* N.p., n.d. Web. 17 Dec. 2012.

Platform - Processor

- The Atmel AT91SAM7L128 processor (aka SAM7L)
- Surrounded mainly by memory and peripherals.
- System controller that controls the clocks and power supply



Platform- LCD Display

- Two line display
- Library functions:
  - *lcd_init*: Turned on the display's power supply
  - *lcd_put_char7*: Displayed a specified character or number (entered in ASCII code) in a specified position on the LCD display

Platform - Keyboard

- connected to the SAM7L chip
- library functions:
  - *keyboard_init*: Set all the columns high with pull-up resistors on the rows
  - *keyboard_column_high*: set a specified column high
  - *keyboard_column_low*: set a specified column low
  - *keyboard_row_read*: returned true if specified row was high, return false if low

Software Architecture

- Keyboard_key
  - returns an integer if a key is pressed.
- DepressedKey
  - takes this integer as input and returns it only when the key as gone from being pressed to being not pressed.
- Pressed_key
  - takes the value from depressed_key and prints the appropriate symbol to the display. This process is repeated inside an infinite loop in the main method.

Software Detail - Lab 1

```
void printFunction(int NUM) {
 clearScreen();  //clear screen
 int num_position [11];       //int array to hold nums
 int n;                       //initialize index for printing NUM
 int x;                       //initialize index for storing NUM
 int ASCII_CORRECTION = 48;    //correction

 if(NUM<0){
  lcd_put_char7('-',0);       //place negative sign
  NUM=-NUM;                   //make positive
  x=1;                        //index begins at 1
  n=1;
 }
 else{
  x=1;
  n=1;
 }
 while(NUM >= 1){
   num_position[x] = NUM%10;     //next char
   NUM = NUM/10;                //since int will cut off
   x++;                         //add 1 to index
 }
if(NUM==0){
   lcd_put_char7(0+ASCII_CORRECTION, 11);
 }
 for(n; n<x; n++){
   lcd_put_char7(num_position[n]+ASCII_CORRECTION,12-n);
      //print
 }
}
```

Software Detail - Lab 2

```
int keyboard_key(){
 int key[2]={-1,-1};        //key coordinate array to be
       returned
 int j=0;
 int i=0;
  for (;;) {                    //infinite loop
   keyboard_init();                //set all to high
     for(j=0; j<7; j++){
       keyboard_column_low(j);      //"look here" column low
       for (i = 0 ; i < 6 ; i++){    //iterate through rows
         if (!keyboard_row_read(i)){
           key[1]=i;              //i=row
           key[0]=j;              //j=column
         }
       }
       keyboard_column_high(j);    //reset column high
  }
```

```
 int returnkey=key[1]*10+key[0]; //keeps track of
column, row
   if(key[1]!=-1){
     return returnkey;            //returns [column, row] as an
integer
   }
   else{
     lcd_print7("NADA");          //nothing is being pressed
   }
}
}
void pressed_key(int x){
   int i=x/10; //i is row
   int j=x%10; //j is column
    int CALC_KEYBOARD[7][6]={{0,0,0,0,0,0},
                       {0,0,0,0,0,0},
                       {0,0,0,0,0,0},
                       {0,7,8,9,0,0},
                       {0,4,5,6,0,0},
                       {0,1,2,3,0,0},
                       {0,0,0,0,0,0}};
    int y=CALC_KEYBOARD[j][i];
   lcd_put_char7(y+'0', 11);  //prints
   clearScreen();             //clears screen
```

Software Detail – Lab 3 (Uh-Oh)

```
char screenDisplay[12]={'a','a','a','a','a','a','a','a','a','a','a','a'};
int keyboard_key(){
    int key[2]={-1,-1};              //key coordinate array to be returned
    int j=0;
    int i=0;
    for(;;) {                        //infinite loop
        keyboard_init();                 //set all to high
        for(j=0; j<7; j++){
            keyboard_column_low(j);      //set "look here" column to low
            for (i = 0 ; i < 6 ; i++){   //iterate through rows
                if (!keyboard_row_read(i)){
                    key[1]=i;            //i=row
                    key[0]=j;            //j=column
                    break;
                }
            }
            keyboard_column_high(j);  //reset "look here" column to high
        }
        if(key[1]==0 && key[0]==0){
            clearScreen();
            int x=0;
            for(x; x<12; x++){
                screenDisplay[x]='a';
            }
        }
        if(key[1] == -1 && key[2] == -1){
            return -1;                //if nothing is being pressed, return -1
        }
int returnkey=key[1]*10+key[0];    //keeps track of column, row
        if(key[1]!=-1){
            return key               //returns [column, row] as an integer
        }
    }
}

int pressed_key(int x){
    int i=x/10; //i is row
    int j=x%10; //j is column

    CALC_KEYBOARD[7][6]={{'0','0','0','0','0','0'},
        {'0','0','0','0','0','0'},
        {'I','0','0','0','0','0'},
        {'0','7','8','9','%','0'},
        {'0','4','5','6','*','0'},
        {'0','1','2','3','-','0'},
        {'0','0','0','=','+','0'}};

    int z=0;      //=INDEX-1;
    for(z=0; z<12; z++){
        if(screenDisplay[z]=='a'){
          screenDisplay[z]=CALC_KEYBOARD[j][i];
        }
    }
    printScreen();                  //will print screenDisplay[]
  }
}
```

Software Detail – Lab 3 (It Gets Worse)

```
void keyboard_get_entry(struct entry *result)
{
    result->number = num;
    lcd_print_int(num);
    result-> operation = op;
    //    lcd_put_char7(op);
}
 int depressedKey(int x){
    if(x!=-1) {
        tempCurrent = -1;
    }
    else {
        tempCurrent = 1;
    }
    if(tempPrevious ==-1 && tempCurrent!=-1) {
        return x;
    }
    else{
return -1;
      }
    tempPrevious = tempCurrent;
}
```

```
int pressed_key(int x){
    int i=x/10; //i is row
    int j=x%10; //j is column
    if((i == 6 && j == 4)||(j==5))
    {
        if(j==5 && i==4){
            op= '%';
            result = make_struct(op, num);
            return;
        }
        else if(j==5 && i==5){
            op= '*';
            result = make_struct(op, num);
            return;
        }
        else if(j==5 && i==6){
            op= '-';
            result = make_struct(op, num);
            return;
        }
        else if(j==5 && i==7){
            op= '+';
            result = make_struct(op, num);
            return;
        }
```

```
else{
     op= '=';
     result = make_struct(op, num);
     return;
 }
 num=0;
 for(q=0;q<screenDisplay.length;q++){
     num=((Integer)screenDisplay[q])*10^q+num;
 }
         return num;
 }
 if (x!=-1) {
    char
CALC_KEYBOARD[7][6]={{'0','0','0','0','0','0'},
        {'0','0','0','0','0','0'},
        {'I','0','0','0','0','0'},
        {'0','7','8','9','%','0'},
        {'0','4','5','6','*','0'},
        {'0','1','2','3','-','0'},
        {'0','0','0','=','+','0'}};
    int z=0;      //=INDEX-1;
    for(z=0; z<12; z++){
        if(screenDisplay[z]=='a'){
           screenDisplay[z]=CALC_KEYBOARD[j][i];
        }
    }
    printScreen();
  }
}
```

Software Detail – Lab 4 (still really Lab 3 and still not completed)

```
void keyboard_key(){
 int j=0;
 int i=0;
 int x=-1;
 int y=-1;
 for (;;) {                    //infinite loop
  while(!keyboard_init());  //wrong parameter**
   keyboard_init();              //set all to high
   for(j=0; j<7; j++){
    keyboard_column_low(j);     //"look here" column low
    for (i = 0 ; i < 6 ; i++){    //iterate through rows
     if (!keyboard_row_read(i)){
      x=i;                    //i=row
      y=j;                    //j=column
      break;                   //exit for loop
     }
    }
    keyboard_column_high(j);   //"look here" column high
  }
 int count=0;
 if (x < 0 && y<0)
      return;
   else if(x==2 && y==5){
     //delete (reset number)
   }
   else if((x>=1 && x<=3)&&(y>=3&&y<=5)){
     //print number
   }
   else if (asdlf;asd;lf){
     //do math
   }
   else {
      count = count+1;
      number[11-count]=CALC_KEYBOARD[y][x];
      int z=0;
      for(z;z<11;z++)
      {
        lcd_put_char7(number[z]+'0', z);  //prints
      }
    }
   }
 }
```

**would have been easier to make keyboard_key return a boolean and use this boolean as a while() parameter to tell if should print again

**note in this code that the rapid printing problem has not been fixed

Lessons Learned

- Choosing the best return type
- Always have a working piece of code which can test if the code is buggy or if the calculator battery died.
- Remember to check all the links connecting the calculator to the computer before assuming that the battery died.
- The longer the code and the more necessary it is to copy paste, the greater the likelihood that there is a simpler solution.
- Remember to look at edge cases.
- It’s much easier to unit test smaller methods than figure out everything that went wrong at the end.