PIC24 Calculator

Spring 2021

Joseph Gehloff, Mohamud Ali, Matthew Kukla, and Matthew Basken

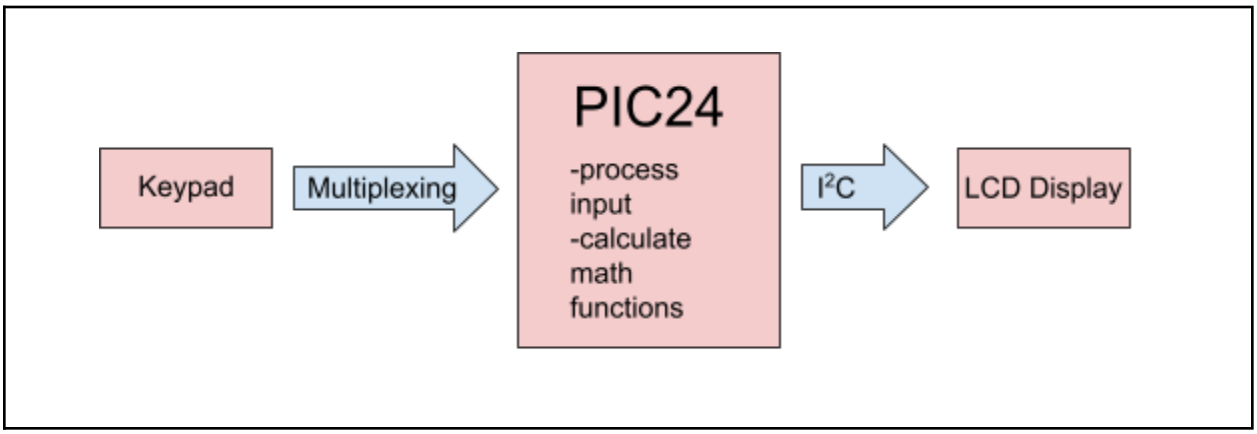
Introduction:

The objective of this project was to program a basic calculator which does simple math calculations such as addition, subtraction, multiplication and division. The calculator takes inputs from the user where the user will insert input values from the keyboard through PIC24 and PIC24 will pass those input values to the LCD display as a final output.

The goal was to use a 8-digit 7-segment display where we bought the component and described in our very first project proposal but unfortunately, we noticed lately that our LED didn't work with SPI communication that the PIC 24 takes. The biggest complication with this LED display was that it was designed with a processor that worked directly with Arduino libraries. These libraries are used to start it in initialization sequences. We tried to recreate these libraries and instructions in MPLAB by sending sets of SPI instructions through the SPI buffer. However, after countless hours with TAs and debugging on our own, we were unable to recreate these libraries. So we had to order a different component but due to the matter of time and possible delays that delivery might cause, we opted to use LCD which we used for lab 5 and 6. Also, we contacted at least two of the course TA's to let them know what has changed from our previous idea but the overall objective for the project is the same as the original one which was to program a calculator which works with the PIC24 and does simple mathematical calculations.

Hardware Description:

The hardware for this project was simple because we had the same component as we did in the previous labs particularly lab 3 and 5. The schematic we used for the wiring was a combination of lab 3 for the keyboard part and lab 5 schematic for the LCD part. Since the wiring of two labs were being combined, two of the pins used for the keypad were switched from RB2 and RB3 to RB0 and RB1 respectively.



Documentation:

The following outlines the setup function used for our calculations. The code also includes the following functions from labs 3 and 5 not specifically outlined here.

- lcd_init(), lcd_cmd(), lcd_printChar(), lcd_printStr(), and lcd_setCursor()

To set up this project, a lot of the settings from labs 3 and 5 were used. Besides the RCDIV, AD1PCFG, LATA, and TRISA settings which are in every lab, the pins RA0, RA1, RA2, and RA3 are all turned on. I2C, which is used to communicate with the LCD, is also enabled. The I2Cx module is disabled and isn't enabled until after the clock rate is set to 100 kHz. The I2C sleep state control bit and the interrupt flag are set to zero. Also included in the setup is the lcd_init() function, which initializes the LCD display.

```
void setup(void){
    CLKDIVbits.RCDIV = 0; //sets scalar to 1:1(8MHz)
    AD1PCFG = 0x9FFF; //set pins to digital
    LATA = 0xFFFF; //makes all Port A output high
    TRISA = 0xFFFF; //make Port A inputs
    CNPU1bits.CN2PUE = 1; //enables weak pull-up for CN pins
    CNPU1bits.CN3PUE = 1;
    CNPU2bits.CN30PUE = 1;
    CNPU2bits.CN29PUE = 1;

    I2C2CONbits.I2CEN = 0; //disable I2C
    I2C2BRG = 0x9D; //100kHz
    I2C2CONbits.I2CEN = 1; // enable I2C
    _I2CSIDL = 0; //continues operation in idle mode
    _MI2C2If = 0; //clears interrupt flag
    lcd_init(); //initializes the display
}
```

Table 1 (Setup Function)

When we were faced with the need to take multiplexed inputs from a keypad and perform computations based on these inputs, we needed to find a way to take any combination of inputs from the user and be able to decode them into integer values and operand instructions. This meant that in input of lets say 1-0-0-+-1-0-= needed to be broken up into a variable num1, and operand, num2, and an equals sign instruction. Now, if we knew these multiplexed values would all be in the same format, meaning tens plus tens then it would be very easy to decode individual inputs into integers. However, the inputs could be an integer of any length. To solve this problem the following if statements were used.

```
if(key == '1'){ //this is the main part of the code that will turn the individual code into a int
   lcd_printChar(0b00110001);
   if(number==0){ //is this was the first button pressed, then set this as number
      number = 1;
   }
   else{ //if this was not the first button pressed,
   number = (number+10)+1; //we shift to the next tens place and add the input (input of 1 then another 1 will result in 11)
   }
}
```

Table 1.1 (Decode Function)

The key variable is returned from the redKeypadRaw() function used in lab3. This way, we were able to take those multiplex inputs and decode them. The method for decoding works as follows. First we test for the case where this was the very first key pressed. If that is the case a variable number is set to whatever key was pressed (in this case 1). If however this was not the first key pressed and the number

already has a value assigned to it, we then enter the else statement. In this statement we simply multiply the number by 10. This effectively shifts the number to the next ten places. We then are able to add the value of the key (in this case 1) to that number. From quick examination, you can see that an input of 1 would result in a number equaling 1, and an input of 1-1 would equal 11. This same method is used for each keypad number 0-9. We also then need to handle inputs operand inputs. The following example shows how these inputs are handled.

```
if(key == 'A'||key =='B'||key=='C'||key =='D'){ //one of the action keys was pressed, now figure out which one
    num1=number;
    number=0;
    if(key =='A'){
        lcd_printChar(0b00101011);
        action = '+';
    }
    if(key =='B'){
        lcd_printChar(0b00101101);
        action = '-';
    }
    if(key =='C'){
        lcd_printChar(0b00101010);
        action = '*';
    }
    if(key =='D'){
        lcd_printChar(0b00101111);
        action = '/';
    }
```

Table 1.2 (Decode Function part 2)

Each key on the keypad used in lab3 was assigned a value. So, 'A' corresponds to '+' and so on. We test for each case and set an action key equal to that input. It is important to remember that an operand input will show up in the middle of the total instruction, so it would be best to save the operand in as a value to be used later. We can also see that num1 is set whatever value of number we have so far. Number is then reset. This way, we now have the value of num1 and the operand so we restart the process to find num2. Num2 is found in the following way.

```
if(key == 'E'){ //equals sign was pressed
    lcd_printChar(0b00111101);
    num2=number;
    result = 1;  //boolean is set
}
```

Table 1.3 (Decode Function part 3)

From this set of code we can see that if the ‘E’ is returned from the readKeypadRaw() function, we know the equals sign was pressed. We now know the instruction is complete from the user. So, we set num2 to the value of number and set a result variable to 1 (this will be important in the main function)

The last function we have is the calculator function. This function simply takes the values we determined from the decoding function and performs the computations based on these variables. This function is written as follows.

```
void calculateResult(){ //math functions
    if(action =='+'){
        number=num1+num2;
    }
    if(action =='-'){
        number=num1-num2;
    }
    if(action =='*'){
        number=num1*num2;
    }
    if(action =='/'){
        number=num1/num2;
    }
}
```

Table 1.4 Calculation function

Now, all of these functions can be combined in the main function and looped to perform these tasks on command and repeatedly. The main function is written as follows.

```
int main(void) {
    setup();
    lcd_cmd(0b00000001);//clear display
    lcd_setCursor(0,0);

    while(1){ //these steps will repeat forever
    key=readKeyPadRaw(); //find what key was pressed
    if(key != 'x') //if no key was detected (readkeypadraw returned an x) then we will not decode the values
    {
     covert_to_int(); //since we have found what key was pressed, we will now decode those values into integer values
     if(result == 1) //if the equal button was pressed
        {
         calculateResult(); //preform math functions
         displayAnswer(); //display answer on LCD (NOTE: i belive this part will take debugging, check if we need to left shift each string after
        }
    }
    }
    return 0;//never reached
}
```

Table 1.5 Main Function

We first run through the setup functions and clear the display as well as set the cursor. In the loop, we continuously read the keypad, and if that returned value is not the default ‘x’ variable (meaning no button was pressed) the code checks to see if the full instruction from the user is complete by looking at the result variable. If this variable is set to 1, we then calculate the result and display the answer. The answer is displayed in the following way.

```
void displayAnswer(){
    lcd_cmd(0b00000001);
    lcd_setCursor(0,0);
    sprintf(displayNumber, "%f", number); //these are simular to lab 6 where we take a char array and turn them into a string

    if(result==1){ //if the equals sign was input, display answer
        lcd_printStr(displayNumber);
    }
}
```

Table 1.6 Main Function

The LCD is first cleared and reset. We then use a similar method as used in lab 6 to convert the result into a string that is able to be displayed.

Basic Usage: When a key is pressed on the keypad, the value that is represented by the key lights up on the LCD. This would essentially utilize the decoding functions but not perform any of the computations. The user could input any series of inputs and be able to see that value appear on the screen. This would show how multiplexed values can be decoded into integer values, then into springs, and then displayed.

Advanced Usage: As the user enters the inputs on the keypad, the numbers they entered, as well as the operand, will be displayed on the LCD. Once the last input is entered, the output is displayed on the LCD. The advanced usage would allow the user to be able to apply all of the basic functions but then be able to perform various computations with them. This takes the idea of decoding and taking values from a multiplexed keypad and allows us to create many functions that the user can use and see on the LCD.