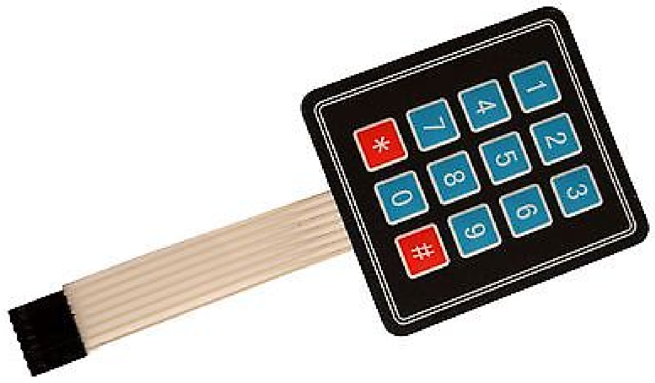
Keypads



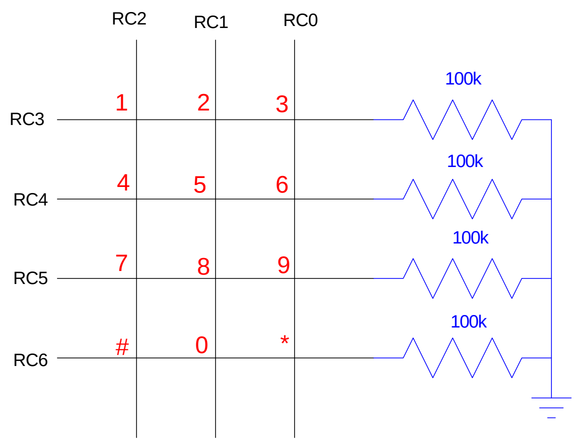
Objective:

One recurring problem with microcontrollers is how to get data to and from the controller. A keypad like the one on your calculator is one way to do this. This lecture should give you the background you need to interface a keypad to your PIC board.

Schematics:

The keypad in the lab is from www.SparkFun.com and costs about $3. It plugs right into PIRTC on your PIC board. There are other keypads out there, but the cost and easy interfacing to your board is why we're using this one. (sidelight: www.SparkFun.com has lots of good stuff for this class - check it out.)

A 3x4 keypad has seven pins: four for rows and 3 for columns. When you press a key, such as '1', row 1 is shorted to column 1. To detect this, the PIC needs to detect which pins are shorted.



One way to do this is as follows:

Connect each row to ground with a 100k resistor. Conveniently, this is already done on your PIC board. Set these four pins to input. If nothing happens, you'll read 0000 on these four pins.

Next, scan the three columns.

- Apply 5V to the first column. If any row reads a '1', the corresponding key in that column was pressed.
- Repeat for column 2 and 3.

Software:

Problem: Write a subroutine which scans PORTC for the keypad. Return 0xFF if no key was pressed. Return the key value for any other key.

```
unsigned char Keypad(void)
{
   unsigned char RESULT, i;
   RESULT = 0xFF;
   TRISC = 0xF8;
   PORTC = 4;                    // scan column #1
   for (i=0; i<10; i++);         // kill 10us to let voltages settle
   if (RC6) RESULT = 1;
   if (RC5) RESULT = 4;
   if (RC4) RESULT = 7;
   if (RC3) RESULT = 10;
   PORTC = 2;                    // scan column #2
   for (i=0; i<10; i++);         // kill 10us to let voltages settle
   if (RC6) RESULT = 2;
   if (RC5) RESULT = 5;
   if (RC4) RESULT = 8;
   if (RC3) RESULT = 0;
   PORTC = 1;                    // scan column #3
   for (i=0; i<10; i++);         // kill 10us to let voltages settle
   if (RC6) RESULT = 3;
   if (RC5) RESULT = 6;
   if (RC4) RESULT = 9;
   if (RC3) RESULT = 11;
   return(RESULT);
   }
```

There is probably a better way to do this, but part of the reason to use C is to produce code that is understandable and reusable.

Problem: Write a program that lets you input a number from -9999 to 9999. End the number with the '#' key (10) for positive, '*' for negative (11).

Solution:

```
int GetNumber(void)
{
   unsigned int RESULT;
   unsigned int KEY;

   RESULT = 0;
   KEY = 0;
   while(KEY != 10) {
      while(Keypad() == 0xFF);
      KEY = Keypad();
      while(Keypad() != 0xFF);
      if (KEY < 10) RESULT = (RESULT * 10) + KEY;
```

```
            }
         if (KEY == 11) RESULT = -RESULT;
         return(RESULT);
         }
```

Note several things:

- The first while statement causes the program to kick out when you hit the '#' key.
- The second while statement waits until you press a key.
- The third while statement waits until you release that key.
- It doesn't matter that the variable RESULT appears in Keypad() and GetNumber(). Each subroutine has local variables that other subroutines can't see - and hence don't care about.
- By making these subroutines, you can call these from several locations in your code. You just call a subroutine from within itself (GetNumber() can't call GetNumber(). That results in an infinite loop.

Fun with Keypads: Alarm Clock

Input a number from 0000 to 9999.

When you press the star button, have the PIC processor count down, displaying the time as XXX,X seconds. When you reach 000.0 seconds, turn on the lights on PORTA.

Solution:

```
// Main Routine

void main(void)
{
   unsigned int i, j;
   int TIME, X, RUN, TEMP;

   TRISA = 0;
   TRISB = 0xFF;
   TRISC = 0xF8;
   TRISD = 0;
   TRISE = 0;
   TRISA = 0;
   ADCON1 = 15;

   PORTA = 0;

   LCD_Init();                  // initialize the LCD

   LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MSG0[i]);
   Wait_ms(2000);
   LCD_Inst(1);

   TIME = 0;
   X = 0;
   RUN = 0;

   LCD_Move(0,0);  LCD_Write('T');
   LCD_Move(1,0);  LCD_Write('X');
```

```
        while(1) {
           TEMP = ReadKey();

           if (TEMP < 10) X = (X*10) + TEMP;

           if (TEMP == 10) {
              TIME = X;
              X = 0;
              RUN = 1;
              }

           if (RUN) {
              while(TIME) {
                 TIME = TIME - 1;
                 LCD_Move(0,5);  LCD_Out(TIME,1);
                 Wait_ms(100);
                 }
              PORTA = 0xFF;
              Wait_ms(1000);
              PORTA = 0;
              RUN = 0;
              }

           LCD_Move(1,5);  LCD_Out(X, 0);
           LCD_Move(0,5);  LCD_Out(TIME,1);

           Wait_ms(100);

           }

        }
```

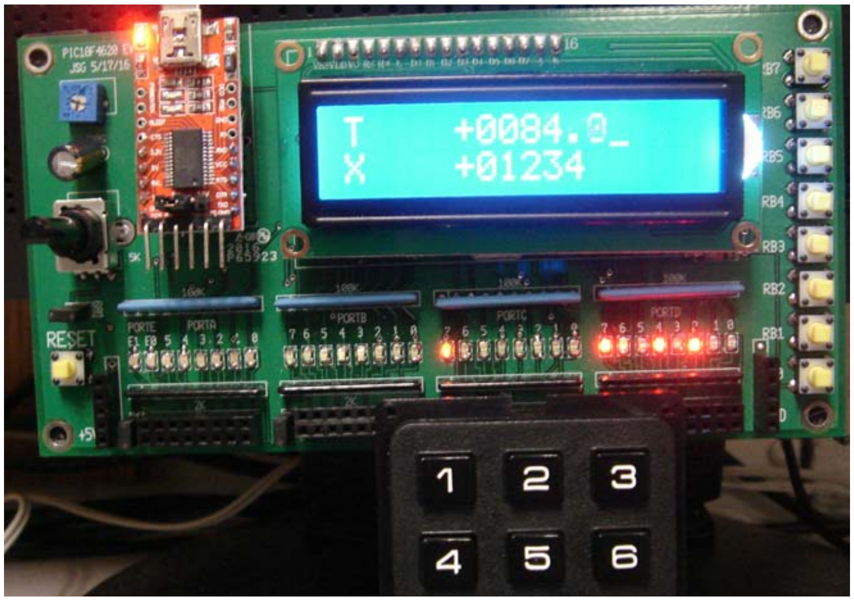
Comments:

The blue section of code reads the keypad - allowing you to input numbers from 0000 to 9999.

When you press the star button '*', whatever number you input is stored in TIME. The red section of code then counts down to zero at one count every 100ms.

When you reach 0000.0 seconds, the lights on PORTA turn on for one second.

The code then repeats and waits for another time.



Keypad Alarm Clock: The keypad connects to PORTC pins 0..6. The number you type in is displayed as X. When you press the * key, the time is displayed as T and it counts down to 0000.0 seconds. When it reaches zero, the lights on PORTA turn on for one second.

RPN Calculator: (RPN.C) Build a calculator which has four operations: +. -, *, /

Program your PIC to operated as an RPN calculator with the following functions:

*	#	RB5	RB4	RB3	RB2	RB1	RB0
Enter	undo	-	clear	*	/	+	-

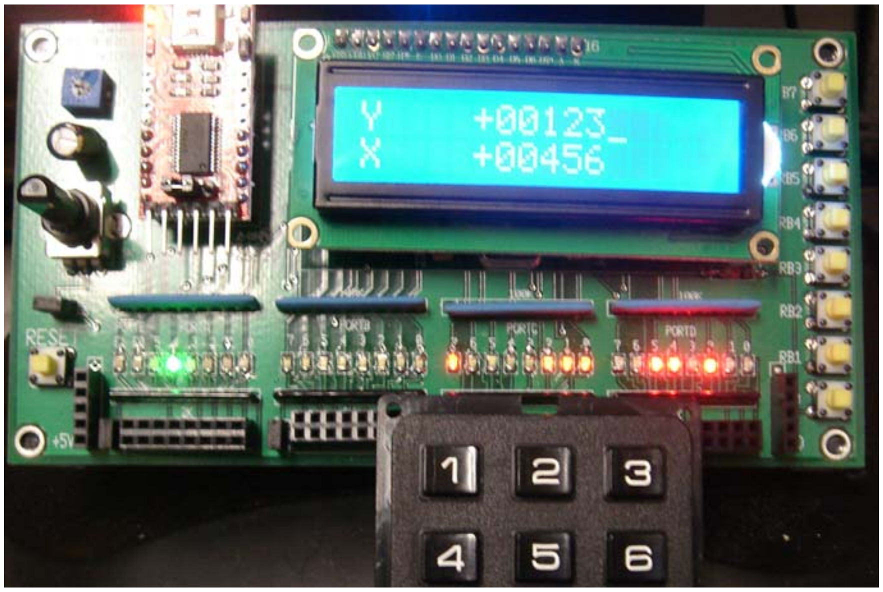
Solution: First, modify the GetKey function so it recognizes other buttons (RB0..RB3)

```
char GetKey(void)
{
   int i;
   unsigned char RESULT;
   TRISC = 0xF8;
   RESULT = 0xFF;
   PORTC = 4;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 1;
   if (RC5) RESULT = 4;
   if (RC4) RESULT = 7;
   if (RC3) RESULT = 10;
   PORTC = 2;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 2;
   if (RC5) RESULT = 5;
   if (RC4) RESULT = 8;
   if (RC3) RESULT = 0;
   PORTC = 1;
   for (i=0; i<100; i++);
   if (RC6) RESULT = 3;
```

```
        if (RC5) RESULT = 6;
        if (RC4) RESULT = 9;
        if (RC3) RESULT = 11;
        if (RB0) RESULT = 12;
        if (RB1) RESULT = 13;
        if (RB2) RESULT = 14;
        if (RB3) RESULT = 15;
        if (RB4) RESULT = 16;
        PORTC = 0;
        return(RESULT);
     }
```

In the main routine, check what button you pressed to know what to do:

```
   while(1) {
      TEMP = ReadKey();

      if (TEMP < 10) X = (X*10) + TEMP;               // Input number

      if (TEMP == 10) {                               // * Enter (push)
         T = Z;
         Z = Y;
         Y = X;
         X = 0;
         }
      if (TEMP == 11) {           // # undo
         X = X / 10;
         }
      if (TEMP == 12) {           // RB0 subtract
         X = Y - X;
         Y = Z;
         Z = T;
         }
      if (TEMP == 13) {           // RB1 add
         X = X + Y;
         Y = Z;
         Z = T;
         }
      if (TEMP == 14) {           // RB2 divide
         X = Y / X;
         Y = Z;
         Z = T;
         }
      if (TEMP == 15) {           // RB3 multiply
         X = X * Y;
         Y = Z;
         Z = T;
         }
      if (TEMP == 16) {           // RB4 clear
         X = 0;
         }

      LCD_Move(1,5);  LCD_Out(X, 0);
      LCD_Move(0,5);  LCD_Out(Y, 0);

      }

   }
```



RPN Calculator. The keypad connects to PORTC pins 0..6. The number you input displays as X. You can push X onto the stack (Y). Operations are addition (RB1) subtraction (RB0), multiplication (RB4) and division (RB3)