12-Key Keypad Connection to a Microcontroller

Using Standard Keypad by Grayhill Inc. of Series 96 with Model Number: 96AB2-152-R

Ali Behbehani

March, 30th 2012

-152

Abstract:

The Grayhill series 96 keypad is 3x4 or 4x4 Keypad made that uses conductive rubber to connect to PIC board traces. It works in a matrix form, so each column is represented by a pin and each row is represented by a pin. in this application note, the 3x4 pin version would be discussed, however, the 4x4 keypad would follow the same analogy. The document would represent how to use the 96AB2-152-R keypad. Some pins on the keypad would be set to high from a Microcontroller chip, other would be connect to a 5v voltage source with some pull up resistors, and there would be a code that when a button is pressed, it would read throw to find which one was pressed by finding where the circuit was shorted.

Key Words:

keypad, Grayhill series 96, 96AB2, 96AB2-152, 96AB2-152-R, keypad to microcontroller connection, keypad connection.

Table of Contents

1. Introduction:

The 12 keys keypad is widely used in many applications, some of those are: telephones and ATM machines. There are many different types of keypads and the keypad whish would be explained here would a matrix method in order to find which key is pushed. This keypad does not have pins for Vdd or Vgnd, which means it does not require a direct connection to a voltage source to perform its task. Also this keypad has 7-pins and each pin would represent a row or a column. As this keypad has 12 key, it has 3 columns and for rows. the mount of the keypad is as seen in figure 1, which shows the dimension and pins location. Moreover, the matrix of the keys related to pins is shown in figure 2. Those Figures are important in order to understand how the keypad is built, and how it would be used. Also, they shows important information that is needed to understand the pin layout and the calibration between each key and its two pins.

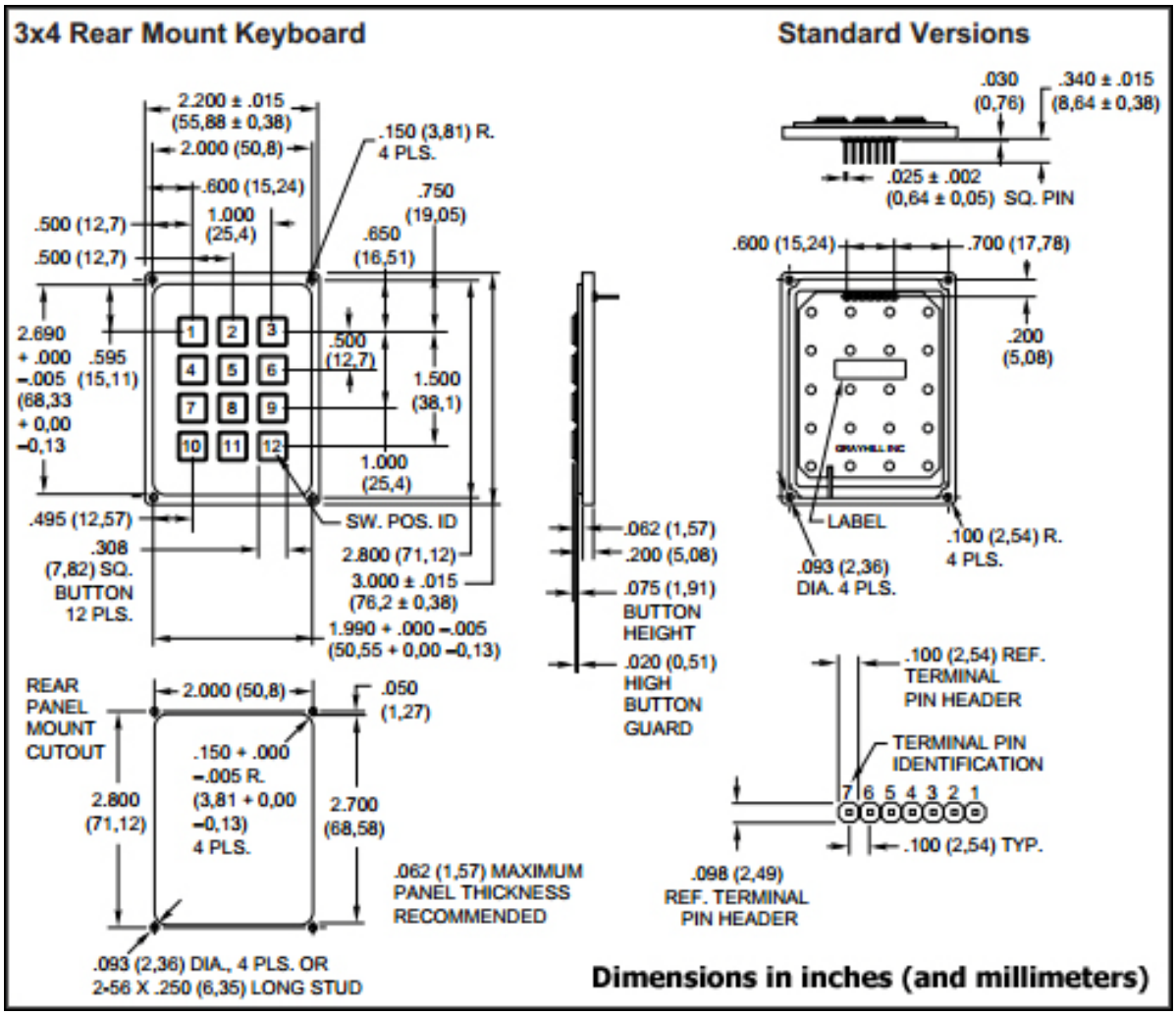

Figure 1

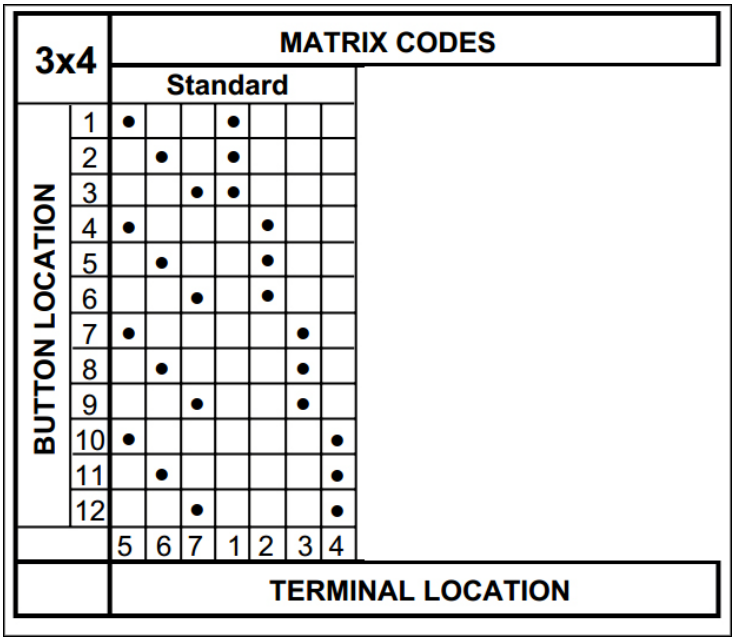
3x4
MATRIX CODES
Standard
BUTTON LOCATION
1 • • •
2 • • •
3 • • •
4 • • •
5 • • •
6 • • •
7 • • •
8 • • •
9 • • •
10 • • •
11 • • •
12 • • •
5 6 7 1 2 3 4
TERMINAL LOCATION

Figure 2

2. Objectives:

One objective of this document is to introduce the 96AB2-152-R keypad, and explains how it works. Also, it will explain the connectivity in the keypad between buttons and their pins. Another objective is to explain how to connect such a keypad into a microcontroller. Moreover, the document would explain how to communicate between the keypad and the microcontroller. A sample code would be provided in the appendix for the communication between the keypad and the microcontroller.

3. Keypad Properties:

The keypad has a 3-columns and 4-rows matrix orientation as seen in figure 2. If a key is pushed then the circuit would be shorted for those specific key pins. The short circuit would always be between a row pin and a column pin. For this specific 96AB2-152-F keypad the rows 1-4 are represented by pins 1-4 and columns 1-3 are represented by pins 5-7. For example button 1 would be represented by pins 1 and 5, so if a voltage is applied to one of the pins and a voltammeter is connected to the other pin, when the button is pushed the voltammeter would read the input voltage. From Figure 2 it would be easy to construct a table that would show each button represents what character, and which pins are shorted if that button is pressed, which is provided in Table 1

Key Number	Character	Culomn Pin	Row Pin
1	'1'	5	1
2	'2'	6	1
3	'3'	7	1
4	'4'	5	2
5	'5'	6	2
6	'6'	7	2
7	'7'	5	3
8	'8'	6	3
9	'9'	7	3
10	'*'	5	4
11	'0'	6	4
12	'#'	7	4

Table 1

4. Keypad Connection to Microcontroller:

The 4 rows of the keypad would be connected to the microcontroller pins directly without other connections. However, each of the 3 columns of the keypad would be connected into a pull up resistor (i.e. 18kΩ) which is connected into a voltage source and the microcontroller. The microcontroller pins that are used are I/O ports, so they can be programmed and used in a different way. All the pins of the keypad are connected to make sure that it would be easy to detect which pin is pushed easily. The connections between the microcontroller and the keypad are provided in figure 3. Another connection ways are possible, but for the provided one there would be an explained example on how it works in the next part.

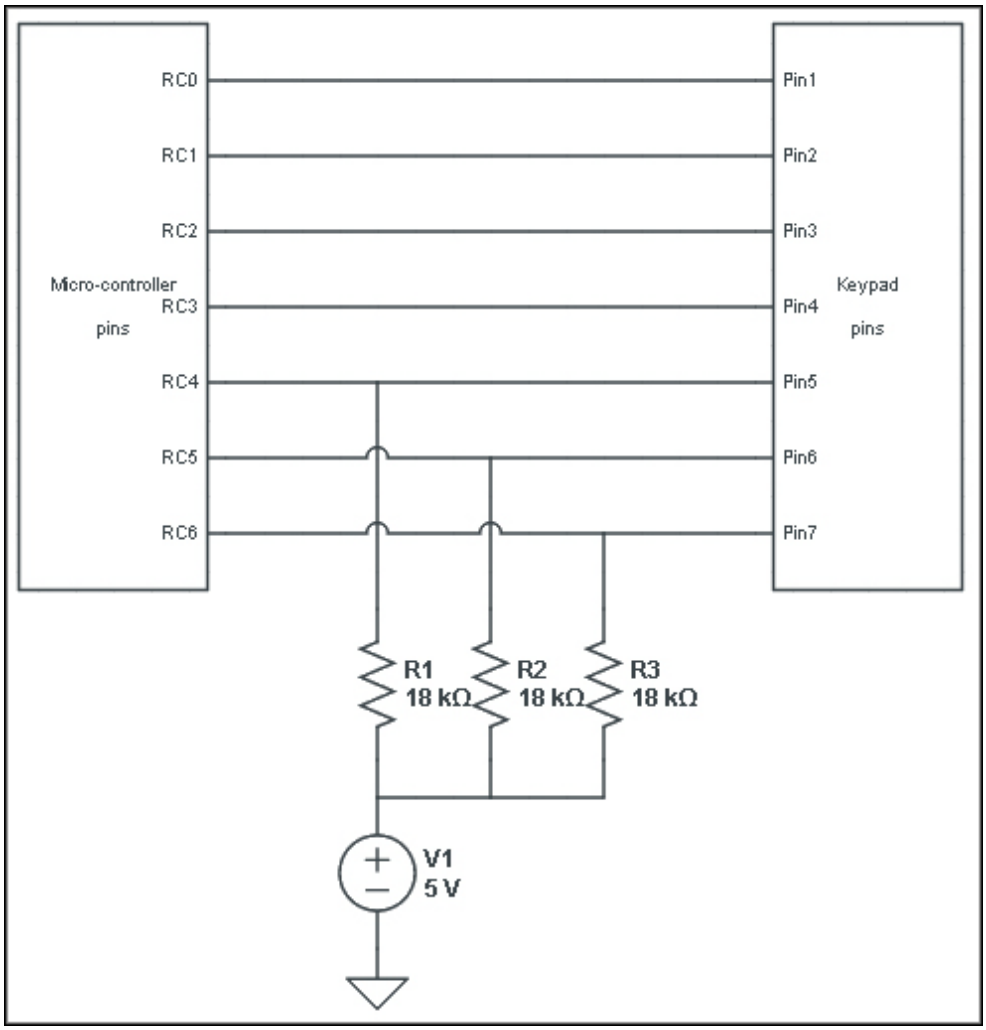

Figure 3

5. Keypad Communication with the Microcontroller:

In this example, the used microcontroller is PIC18F4520, so some other microcontrollers might have some different requirements or setting in the code, however the method is set to be general so it could be used in any microcontroller that have the appropriate pins and settings. Also, some important parameter should be set in advance to find which key is pushed. First the microcontroller pins that are connected to the rows should be set to be high output pin, and the column pins should be set to be input pins. Then in order to detect if a pin is pushed the microcontroller would set one row output to be low and then try to detect if it receives a low input from the column of the keypad. Then, it would set that row back to high and set another row to low, that would happen until it detects the low signal and convert that signal into a key character. This loop would happen in a matter of microseconds so if a button is pushed it would detect it easily.

For example: if button `8' is pushed the microcontroller would set the first row to low, then detect if an input goes low, which none would because `8' is connected to row 3 which is high and then the shorted output between column 2 and row 3 would still be high. Then the microcontroller will reset row 1 to high and set row 2 to low, which would not change any input. Then, the microcontroller would reset row 2 to high and set row 3 to low, and try to detect if one of the columns inputs is low, which the 2nd column would be low, because when `8' is pressed column 2 would be shorted with row 3 and since row 3 is low, and column 2 is connected to it and to a pull up resistor, the resistor would pull the voltage away from the pin of column 2 which makes the microcontroller detects a low input in column 2 when row 3 is set to low, then it would now that `8' is pressed.

A sample code for the loop detection method would be provided at the appendix however, it does not have all the information about setting the pins. It was provided to make the used loop detector easier to understand.

6. Summary:

The document provides some introduction about the 96AB2-152-R keypad with the appropriate figures that are provided from its datasheet. Also, the properties of the keypad were introduced to make it easier to understand how it works. An example connection with a microcontroller is explained to help in understanding how to connect such a device. Then the communication between the microcontroller and the keypad was explained and provided with an explained example. This all helps in setting up the keypad to work properly.

7. References:

http://media.digikey.com/pdf/Data%20Sheets/Grayhill%20PDFs/96%20Series.pdf

http://www.egr.msu.edu/classes/ece480/capstone/ForMiniprojects/

http://ww1.microchip.com/downloads/en/DeviceDoc/39631a.pdf

8. Appendix:

```
while(i == 0)
{
   PORTCbits.RC0 = 0;  //set row 1 low
   delay();
   j = 0;
   while(!PORTCbits.RC4)   //check if column 1 is low
   {
      if(j == 0)
      {
         cX = '1';
         LCD_PutChar (cX);
         j = 1;
      }
   }
   j = 0;
   while(!PORTCbits.RC5)   //check if column 2 is low
   {
      if(j == 0)
      {
         cX = '2';
         LCD_PutChar (cX);
         j = 1;
      }
   }
   j = 0;
   while(!PORTCbits.RC6)   //check if column 3 is low
   {
      if(j == 0)
{
         cX = '3';
         LCD_PutChar (cX);
         j = 1;
      }
   }
   PORTCbits.RC0 = 1;  //set row 1 high
   PORTCbits.RC1 = 0;  //set row 2 low
   delay();
   j = 0;
   while(!PORTCbits.RC4)   //check if column 1 is low
   {
```

```
    if(j == 0)
    {
        cX = '4';
        LCD_PutChar (cX);
        j = 1;
    }
}
j = 0;
while(!PORTCbits.RC5)   //check if column 2 is low
{
    if(j == 0)
    {
        cX = '5';
        LCD_PutChar (cX);
        j = 1;
    }
}

j = 0;
while(!PORTCbits.RC6)   //check if column 3 is low
{
    if(j == 0)
    {
        cX = '6';
        LCD_PutChar (cX);
        j = 1;
    }
}
PORTCbits.RC1 = 1;  //set row 2 high
PORTCbits.RC2 = 0;  //set row 3 low
delay();
j = 0;
while(!PORTCbits.RC4)   //check if column 1 is low
{
    if(j == 0)
    {
        cX = '7';
        LCD_PutChar (cX);
        j = 1;
    }
}
j = 0;
while(!PORTCbits.RC5)   //check if column 2 is low
{
```

```
    if(j == 0)
    {
        cX = '8';
        LCD_PutChar (cX);
        j = 1;
    }
}
j = 0;
while(!PORTCbits.RC6)   //check if column 3 is low
{
    if(j == 0)
    {
        cX = '9';
        LCD_PutChar (cX);
        j = 1;
    }
}
PORTCbits.RC2 = 1;  //set row 3 high
PORTCbits.RC3 = 0;  //set row 4 low
j = 0;
delay();
while(!PORTCbits.RC4)   //check if column 1 is low
{
    if(j == 0)
    {
        cX = '*';
        LCD_PutChar (cX);
        j = 1;
    }
}
j = 0;
while(!PORTCbits.RC5)   //check if column 2 is low
{
    if(j == 0)
    {
        cX = '0';
        LCD_PutChar (cX);
        j = 1;
    }
}
j = 0;
while(!PORTCbits.RC6)   //check if column 3 is low
{
    if(j == 0)
```

```
      {
         cX = '#';
         LCD_PutChar (cX);
         j = 1;
      }
   PORTCbits.RC3 = 1;//set row 4 high
   j = 0;
   delay();
   }
}
```