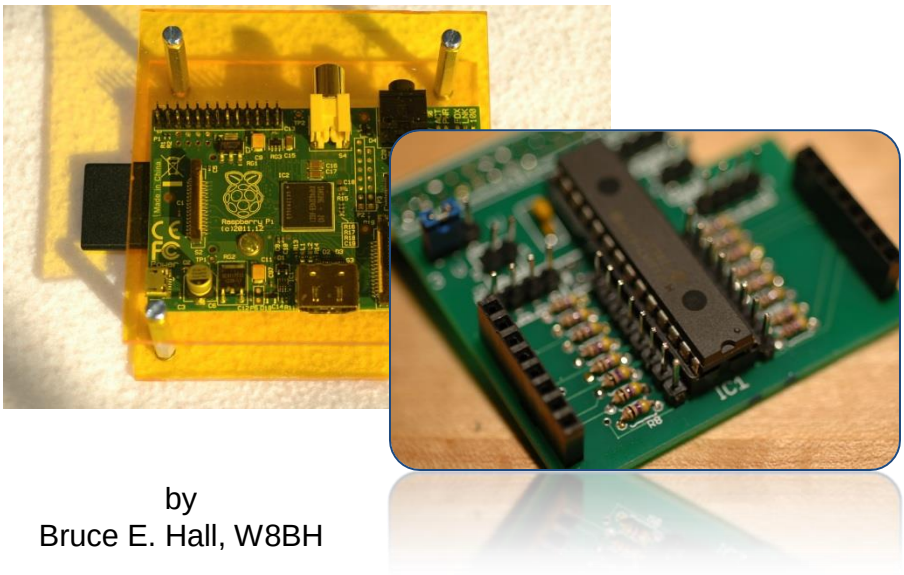


Beginner's Guide to the PI MATRIX

- Part 2-

by
Bruce E. Hall, W8BH

1) INTRODUCTION

In Part 1 of this series, we looked at how to build this great little kit, the Pi Matrix, and how to configure I2C on the raspberry pi to communicate with it. The result was a simple python script to turn all of the LEDs on & off. Exciting! But sooner or later you'll want it to do something interesting, right? In this tutorial we'll look at how the Pi Matrix works, and write code for something more YouTube friendly!

2) HOW DOES IT WORK?

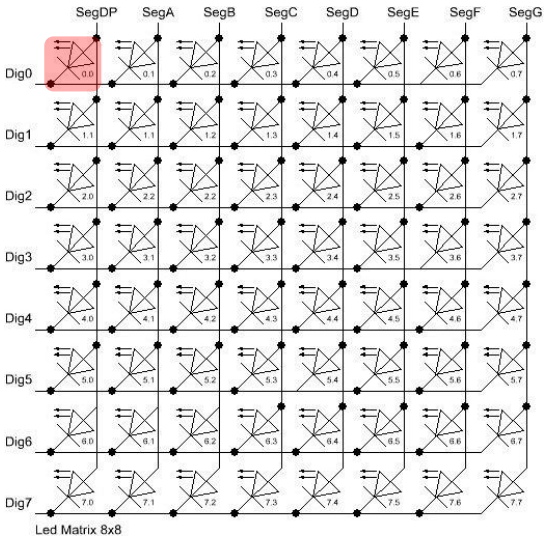
The heart of the PI MATRIX is the MCP23017 chip from Microchip. This is a really wonderful device for creating hardware interfaces. Why?

- It speaks I2C, which only needs 2 communication wires from your processor
- It buffers and protects your processor from the outside world
- It gives you a large number of I/O pins (16) to work with
- It is dirt cheap! Less than $1.50 at unit quantities

If you do something 'bad' with your connection to the outside world and fry the 23017, you've lost less than $2.00 and still have a working pi. It's a great deal.

It just so happens that the LED matrix requires 16 data lines - a perfect match. But how can you control 64 LEDs with only 16 lines? Shouldn't you need 64 lines? That's where the 'matrix' comes in. Take a look at a typical wiring diagram of these devices:

The LEDs are wired in an 8x8 matrix, where there are 8 rows (labeled ‘Dig’) and 8 columns (labeled ‘Seg’). These 8 rows and 8 columns make up our 16 connections to the outside world. To turn on an LED, apply power and ground to the corresponding row & column pins. For example, to turn on the highlighted LED at position (0,0), apply power to the anode on left-most column ‘SegDP’ and ground to the cathode on the top-most row ‘Dig 0’. Notice that each LED shares a row connection with 7 other LEDs and a column connection with 7 other LEDs.



Let’s see how our program from part 1 works. To light up all of the LEDs we send the hexadecimal value of 0xFF to the Port A, which go to the column pins of the matrix. 0xFF is the binary value of ‘11111111’. Each bit and therefore each column input will be at logic 1. We also must send the value of 0x00 to Port B, which go to the rows. The Port B binary value is ‘00000000’. Each bit and therefore each row input will be at logic 0 (ground). Result: all the columns (anodes) are high and all the rows (cathodes) are low, so the LEDs turn on.

3) CODING FOR COLUMNS

We can make more interesting displays by changing the row and column inputs. For instance: instead of taking all of the column inputs high, what would happen if we only took one column high? Answer: only the LEDs in that column would turn on. The others would stay dark.

Let’s write a routine to turn on a single column. To turn on the first column, we just need to turn on the lowest bit. This is easy: just write the value of ‘00000001’ (0x01) to Port A like this:

```
bus.write_byte_data(ADDR,PORTB,0x01)
```

Now what if we wanted to turn on the third column? We would need to write the binary value ‘00000100’ (which equals 0x04) to Port A

```
bus.write_byte_data(ADDR,PORTA,0x04)
```

Later versions of python will let us write values as binary literals like this: ‘0b00000100’ but we’ll stick to the equivalent value in hexadecimal notation ‘0x04’ for compatibility. If we start numbering our columns at zero, we can generalize the value mathematically as: 2^n Column 0 equals 1, column 2 is 2, column 2 is 4, column 3 is 8, and so on. To code this value we use a sneaky trick, the left-shift operator ‘<<’, like this:

```
bus.write_byte_data(ADDR,PORTA,1<<column)
```

And that's all we should need except for one small problem: the column pins for the LED matrix are wired in opposite direction of the PortA pins. To correct for this we'll use the right-shift operator instead of the left, like this:

```
bus.write_byte_data(ADDR,PORTA,0x80>>column)
```

You don't really have to do this last step, as long as you're willing to look at the matrix upside-down. It's all a matter of orientation!

4) CODING FOR ROWS

Now let's do the same thing for rows. Easy, right? Yes and No. We can use the same technique, but electrically we need to selectively bring our row to logic 0, not logic 1. For example, to turn on row 3 we'll need to send the binary value '11111011' to PortB. The third bit from the left is logic 0, and everything else is logic one. Notice that all of the bits are flipped, or inverted, compared to what we used in the column method. In fact, we can write the binary value '11111011' as the inverse of '00000100'. In Python, the inverse operator is the tilde (~). Our row coding looks like this:

```
bus.write_byte_data(ADDR,PORTB,~(1<<row))
```

5) CODING FOR INDIVIDUAL LEDs

It's easier than you think. All we have to do is combine the code for selecting the row and selecting the column!

```
bus.write_byte_data(ADDR,PORTA,0x80>>column)
bus.write_byte_data(ADDR,PORTB,~(1<<row))
```

We now have enough material for another demo program, copied below. I've grouped each type of demo into its own routine. Try typing this code using your favorite editor, or copy it from http://w8bh.net/pi/matrix-part2.py. You can run it from bash if you make it executable, like this:

```
chmod +x matrix-part2.py
./matrix-part2.py
```

Have fun! In Part 3 of this series we will do some more intricate matrix displays.

6) PYTHON SCRIPT for PART2:

```
#!/usr/bin/python

########################################################################
#
#      matrix-part2:  test of Pi Matrix board
#
#      This application will test out your Pi LED 8x8 Matrix board
#      by selectively turning on and off each row, column, and pixel.
#
#      Use the concepts from these simple test routines
#      to make more interesting displays.
#
#      Author:  Bruce E. Hall <bhall66@gmail.com>
#      Date  :  03 Feb 2013
#
########################################################################

import smbus               #gives us a connection to the I2C bus
import time                #for timing delays

#Definitions for the MCP23017 chip
ADDR   = 0x20              #The I2C address of our chip
DIRA   = 0x00              #PortA I/O direction, by pin. 0=output, 1=input
DIRB   = 0x01              #PortB I/O direction, by pin. 0=output, 1=input
PORTA  = 0x12              #Register address for PortA
PORTB  = 0x13              #Register address for PortB

#
#      Lower Level LED Display routines
#   Write data directly to the Pi Matrix board
#

def Init23017 ():
       #Set up the 23017 for 16 output pins
       bus.write_byte_data(ADDR,DIRA,0x00);  #all zeros = all outputs on PortA
       bus.write_byte_data(ADDR,DIRB,0x00);  #all zeros = all outputs on PortB

def TurnOffLEDS ():
       bus.write_byte_data(ADDR,PORTA,0x00)  #set all columns low
       bus.write_byte_data(ADDR,PORTB,0x00)  #set all rows low

def TurnOnLEDS ():
       bus.write_byte_data(ADDR,PORTA,0xFF)  #set all columns low
       bus.write_byte_data(ADDR,PORTB,0x00)  #set all rows low

def SetLED (row,col):
       #turn on an individual LED at (row,col).  All other LEDS off.
       bus.write_byte_data(ADDR,PORTA,0x80>>col)
       bus.write_byte_data(ADDR,PORTB,~(1<<row))

def SetColumn (col):
       #turn on all LEDs in the specified column. expects input of 0-7
       bus.write_byte_data(ADDR,PORTB,0x00)
       bus.write_byte_data(ADDR,PORTA,0x80>>col)

def SetRow (row):
       #turn on all LEDs in the specified row.  expects input of 0-7
       bus.write_byte_data(ADDR,PORTA,0xFF)
       bus.write_byte_data(ADDR,PORTB,~(1<<row))
```

```
#
#      Routines for testing our Pi Matrix board
#      Each one calls on one or more of the low-level routines above
#


def Pause():
       #Turn off LEDS and wait a while between cases
       TurnOffLEDS();
       time.sleep(0.5);

def FlashLEDS (delay):
       #Flash all of the LEDS on/off for the specified time
       TurnOnLEDS()
       time.sleep(delay)
       TurnOffLEDS()
       time.sleep(delay)

def LEDtest (delay):
       #Turn on all 64 LEDS for the specified time delay, in seconds
       print "\nLighting all LEDS for %d seconds" % delay
       TurnOnLEDS()                    #turn them all on
       time.sleep(delay)               #wait a while
       Pause()                                      #then turn them off

def FlashTest (numCycles,delay):
       print "\nFlash all of the LEDS"
       for count in range(0,numCycles):
              FlashLEDS(delay)
       Pause()

def ColumnTest (numCycles):
       #Turn on each Column.
       #Keep PortB (rows) low & set each bit in PortA (columns) high
       #This will actually light LEDS in reverse order, col 7 to col0,
       #because Port A bit 0 is wired to Col7, A1 to Col6,..., A7 to Col0
       print "\nTurn on Columns 0 to 7"
       for count in range(0,numCycles):
              print "...cycle",count+1
              for col in range(0,8):
                     SetColumn(col)
                     time.sleep(0.5)
              time.sleep(1)
       Pause()

def RowTest (numCycles):
       #Turn on each row, from row 0 to row 7
       #Keep PortA (columns) high & selectively turn a bit in PortB (rows) low
       print "\nTurn on Rows 0 to 7"
       for count in range(0,numCycles):
              print "...cycle",count+1
              for row in range(0,8):
                     SetRow(row)
                     time.sleep(0.5)
              time.sleep(1)
       Pause()
```

```
def PixelTest (numCycles):
       #Flash each LED according to its (row,col) coordinate
       print "\nFlash each LED in (row,col) order"
       for count in range(0,numCycles):
              for row in range(0,8):
                     for col in range(0,8):
                            SetLED(row,col)
                            time.sleep(0.1)
       Pause()


#
#      Here is the program body: call each of the text routines in turn.
#      Global variable 'bus' is our connection to the I2C bus
#

print "\nPi Matrix test program starting"
bus = smbus.SMBus(1);             #Use '1' for newer Pi boards; 0 for oldies
Init23017()                       #Set all 16 I/O pins as output
LEDtest(5)                        #Turn on all LEDS to verify connectivity
FlashTest(25,0.1)                 #Flash them all for a little fun
ColumnTest(3)                     #Sequentially turn on Columns 0 to 7
RowTest(3)                        #Sequentially turn on Rows 0 to 7
PixelTest(1)                      #Flash each pixel in row,col order
FlashLEDS(0.1)                    #Visual end of test
print "\nDone."
```