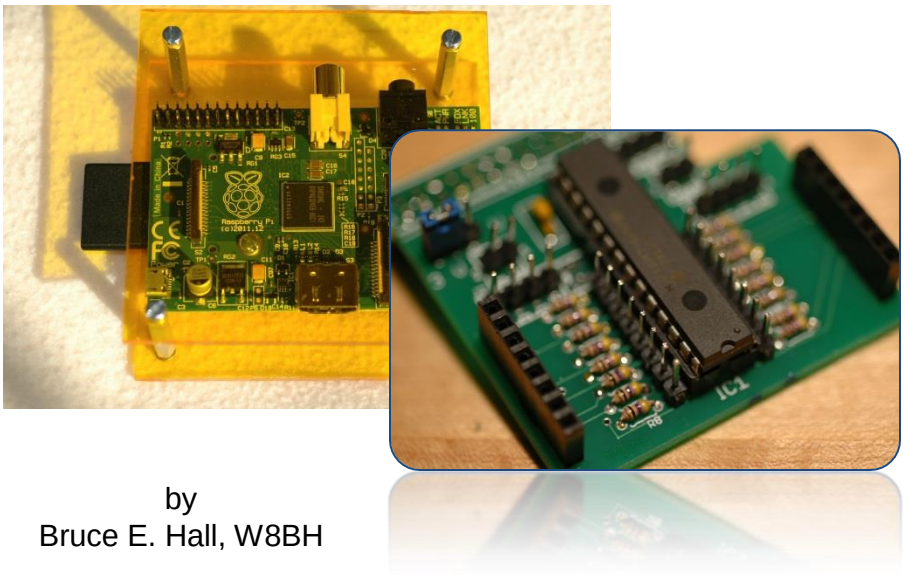


Beginner's Guide to the PI MATRIX

- Part 1-

by
Bruce E. Hall, W8BH

1) INTRODUCTION

The Pi Matrix is a fantastic tool for learning GPIO programming on the raspberry pi. Sure, you could hook up a few LEDs and switches instead, but why do that when you can program a huge matrix of 64 LEDs? If you are like me, and ready to learn about GPIO, this is the first kit you should get. Why?

- It's fun.
- It's compact: Just pop it on top of the pi
- It's clean: no need for a breadboard or wires
- It's safe(r) for your pi: does not use unbuffered GPIO pins
- It's a great excuse to get out your soldering iron.
- It's an easy way to learn about i2c on your pi
- Did I mention that it's fun?

As a huge bonus, just remove the LED matrix and you are left with 16 new, buffered I/O pins to command. This is more than the 8 unbuffered general I/O pins that you started with.

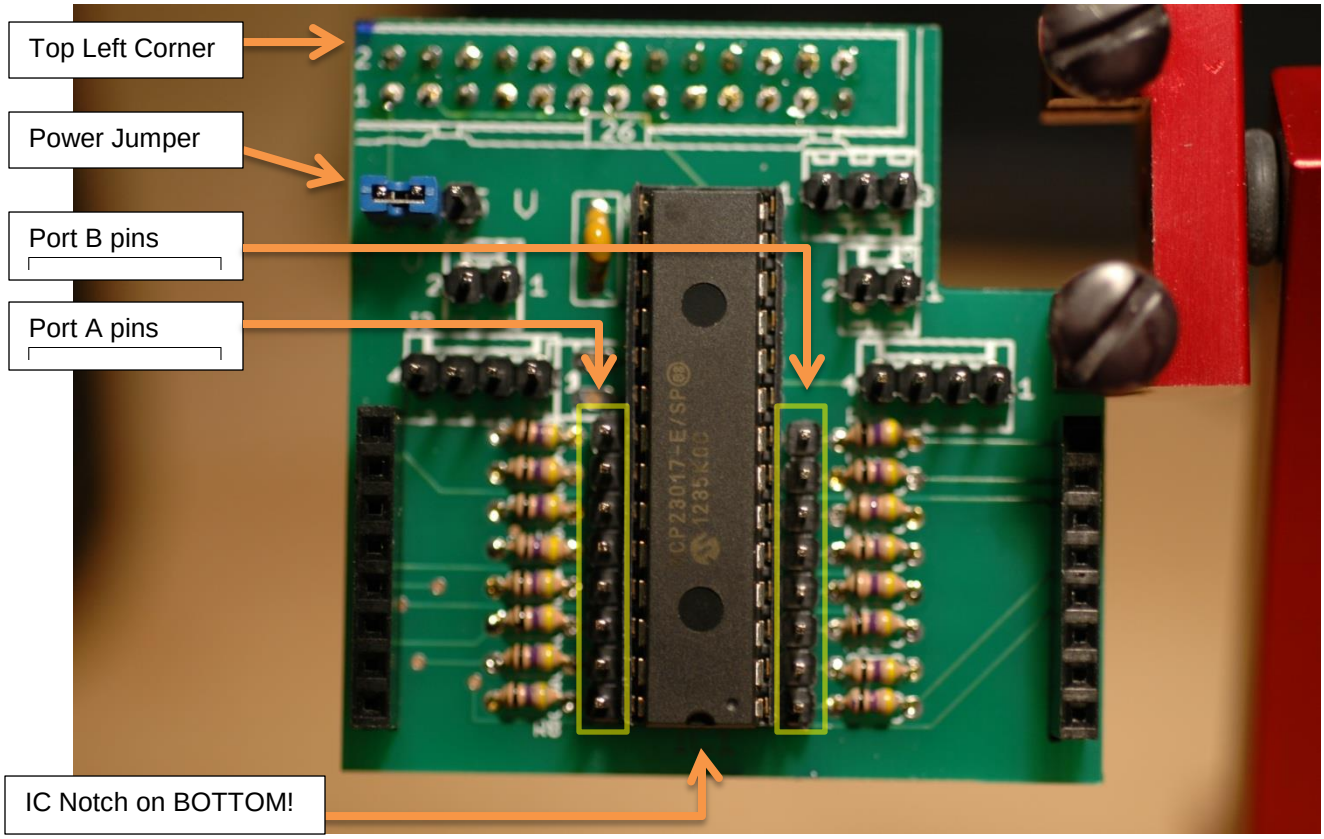
I've been playing the 'matrix' for a couple weeks now, and learned a lot. The first thing I learned is that the all of the information I needed wasn't in one spot. I was a beginner, but there weren't any beginner guides! So here are my notes.

2) CONSTRUCTION

I enjoy a good excuse to melt solder. My bag of parts arrived safely, and well packaged. This kit does not contain a huge number of parts. Check to be sure you have your matrix (of course), the MPC23017 chip & socket, the PCB, the 26-pin female connector, two female header strips, male header pins, a small header pin jumper, and a bunch of teeny-tiny resistors. Use whatever method you like for keeping parts off the floor, sorted & secure in

your workspace. Some people like egg cartons or muffin tins. I use a shoebox lid.

The kit does NOT come with any documentation. Print out the online instructions and follow along. Remember to put your 26-pin connector on the bottom of the board, and the rest of the components on the top. I found it helpful to put a small magic-marker dot on the top-side, top-left corner of the PCB, the same way that ICs are marked, so that I always knew the proper orientation. This spot is silkscreened with the numerals ‘2 1’, marking the first two pins of the 26-pin header.



Take care to mount the socket and 23017 IC with notch pointed DOWN, not up.

Nothing was difficult to solder. If you want nice, straight header pins, tack down one pin with solder, check alignment, resolder if necessary, and solder the rest when it everything looks vertical and flush with the board. Don’t solder a whole strip without checking. It also helps to immobilize the pins while you turn the board over to solder. Easier said than done! Some people like to use ‘blue tack’ (poster putty). I usually don’t have any when I need it, but carefully placed masking tape works in a pinch.

There are lots of holes for male header pins. The only header pins that you’ll need to light up the matrix are the three in the top left where the power jumper goes. The rest are not needed for the LEDs, but will be helpful if you use your PI MATRIX for other things.

Here is what all those headers are for:

a) 4 pin headers: ground connections. all of the them ground.
b) 2 pin headers: Vcc (3v3) power.
c) 3 pin header: chip voltage jumper, keep at 3v3 (jumper left/center).
d) 8 pin headers: breakout for all 16 I/O lines. Pins on left are PortA; pins on left are PortB

When everything is soldered up, install the chip (pin 1 down) and LED matrix. The matrix pin numbers are marked on the bottom: '1' and '16'. Pin 1 is top-left. If you install it upside-down it won't work right.

3) SETTING UP I2C

The PI MATRIX needs additional software installed before we can use it. In particular, we must configure the I2C bus. There are a couple good tutorials online for this: for example, check out: http://www.skpang.co.uk/blog/archives/575. What I describe here is almost exactly the same procedure.

I2C drivers are disabled by default. Let's enable them. We'll need to edit a couple system files, install a couple software packages, and then reboot.

From the command line, enter:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf'
```

Look for an entry like 'blacklist i2c-bcm2708'. Add a hash mark '#' at the beginning of the line: '#blacklist…' Press CTRL X then y to save and exit.

Next edit the modules file by:

```
sudo nano /etc/modules
```

Add i2c-dev on a new line, then save and exit.

Next, install some tools and python support:

```
sudo apt-get update
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Now add a new user to the i2c group:

```
sudo adduser pi i2c
```

Shut down your pi now:

```
sudo halt
```

Turn off the power and seat the PI MATRIX board on raspberry pi's 26-pin GPIO header. The board should lay across the top of the pi, not away from it. Cross your fingers, turn on the power, and log back in to the pi account.

From the command line, enter:

```
sudo i2cdetect -y 1
```

(Use 0 instead of 1 if you have an older pi. Older pi's don't have any mounting holes on the PCB; newer ones have 2 mounting holes.)

This tool will listen for I2C devices, and report their addresses. If all goes well, you should see all blanks except for a '20' on the second line. This indicates that your pi is communicating with the MCP23017 (address 0x20) on the PI MATRIX. Congratulations!

If you don't see the '20', remove the matrix and start troubleshooting. Nothing that follows will work until you've established I2C communication.

4) BABY STEPS

Now that I2C is working, its time to talk to the LEDs! The i2cset utility in the i2c-tools package lets us send data. The format is "i2cset -y <bus number> <chip addr> <data addr> <value>". Option -y is not required, but turns off a lot of scary-sounding warnings. We will send data to the PI MATRIX one byte at a time. The bus number is 0 for older pi's and 1 for new ones. Mine is newer. Try these, in the order shown, omitting the comments:

```
i2cset -y 1 0x20 0x00 0x00      #set port A to all outputs
i2cset -y 1 0x20 0x01 0x00      #set port B to all outputs
i2cset -y 1 0x20 0x13 0x00      #set all port B (row) pins low
ic2set -y 1 0x20 0x12 0xFF      #set all port A (column) pins high; LEDs turn on!
ic2set -y 1 0x20 0x12 0x00      #enter this to turn the LEDs off
```

Did you see the all of the LEDs turn on? If not, reseat the matrix board and/or LEDs, and make sure that you've oriented it correctly (Pin 1 of the LED matrix towards the top-left of the pi-matrix board. The matrix board is mounted over the pi, not away from it.).

5) MATRIX PROGRAMMING FOR DUMMIES

It's time to try some real programming in Python. Why Python? Other languages, like C, would work equally well. But I don't know Python and this project a good excuse to learn something about it. Try looking at some examples on github for raspberry pi programs that use i2c. I found them most of them, like Adafruit's I2C library, very complicated and confusing! I prefer starting simple, getting small stuff to work, then adding to it. If you like this method too then read on.

The simplest way to try out some Python is to use the interactive mode. From the bash prompt just type: python

The python prompt is a triple chevron '>>>'. From here we enter our python code, line by line, and the interpreter works as we go. Enter the following:

```
>>> import smbus
>>> addr = 0x20
>>> dira = 0x00
>>> dirb = 0x01
>>> porta = 0x12
>>> portb = 0x13
>>> bus = smbus.SMBus(1)
```

If you did it right, you shouldn't get any feedback. The first line adds our I2C library for python, the next five are constants for addressing the MCP23017 chip, and the last one establishes a variable to write to/from the bus. Now we just have to send data to the matrix board:

```
>>> bus.write_byte_data(addr,dira,0x00)
>>> bus.write_byte_data(addr,dirb,0x00)
>>> bus.write_byte_data(addr,portb,0x00)
>>> bus.write_byte_data(addr,porta,0xFF)
```

The first two lines set all 16 I/O pins to output. The next line takes all of the row pins low, and the last line takes all of the column pins high. If you didn't make any spelling mistakes, all the LEDs should be on now. To turn them off, just zero out the port A register:

```
>>> bus.write_byte_data(addr,porta,0x00)
```

Once you get that working, it isn't too hard to make an honest-to-goodness Python script. I've written a short one (my very first Python), based entirely on the 12 lines of code above. Try typing it using or favorite editor, or copy it from http://w8bh.net/pi/hello-matrix.py. You can run it from bash if you make it executable, like this:

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

In time-honored 'hello-world' style, this script will flash the LEDs on your display. Notice that the LEDs might remain on, depending when you quit. Run it again, or use one of the methods above to turn off the LEDs. Have fun! In Part 2 of this series we will do some more interesting displays on the matrix.

6) PYTHON SCRIPT:

```
#!/usr/bin/python

#----------------------------------------------------------------------
#
#      hello-matrix.py
#      Version 1.0
#
#      Purpose :  Flashes All LEDs on the Pi Matrix Board
#
#      Author  :  Bruce E. Hall <bhall66@gmail.com>
#      Date    :  02 Feb 2013
#
#----------------------------------------------------------------------

import smbus                                #python access to I2C bus
import time                                 #for timing delays

# Definitions for the MCP23017 chip
ADDR   = 0x20                               #I2C bus address of the 23017
DIRA   = 0x00                               #PortA I/O direction register
DIRB   = 0x01                               #PortB I/O direction register
PORTA  = 0x12                               #PortA data register
PORTB  = 0x13                               #PortB data register

#
# Program Start
#
print "Hello, Pi Matrix!  (press Ctrl-C to stop)"

bus = smbus.SMBus(1)                        #Use '1' for newer Pi; 0 for oldies
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
bus.write_byte_data(ADDR,PORTB,0x00)        #set row outputs low (cathodes to
ground)

while (True):                               #loop until user halts program
    bus.write_byte_data(ADDR,PORTA,0xFF)    #set column outputs high = turn on LEDs
    time.sleep(0.5)                         #keep LEDs on for a while
    bus.write_byte_data(ADDR,PORTA,0x00)    #set column outputs low = turn off LEDs
    time.sleep(0.5)                         #keep LEDs off for a while
```

7) REFERENCE

Board orientation matters. I am assuming the following orientation:

a) Pins 1 & 2 of the 26-pin header are on the TOP/LEFT corner of the matrix board.
b) The LED matrix is oriented the same way, with pin1 on the left and towards the top of the pcb.
c) LED rows go left to right, with row 1 being at the top and row 8 at the bottom.
d) LED columns go up to down, with column 1 on the left and column 8 on the right

It is easy to get the orientation messed up, since your pi may not be oriented the same way. The rows & columns are correct if you look at the pi with GPIO header on top (an SD card on left, USB/Ethernet on right),

With this orientation, the LED matrix has the following pinout*:

Pin 1 = Col 4	Pin 16 = Row 8
Pin 2 = Col 2	Pin 15 = Row 7
Pin 3 = Row 2	Pin 14 = Col 7
Pin 4 = Row 3	Pin 13 = Row 1
Pin 5 = Col 1	Pin 12 = Col 5
Pin 6 = Row 5	Pin 11 = Row 6
Pin 7 = Col 3	Pin 10 = Row 4
Pin 8 = Col 6	Pin 9 = Col 8

*Datasheets for this 2088B-style 8x8 matrix show a different pinout because they use a different orientation.

The LEDs are oriented with the anode on the columns and cathode on the rows. To light an LED you must raise the corresponding column high and take the row low.

The MCP23017 is mounted notch-down. Looking from the top, here is the data pin layout. Both 8-bit data ports are broken out on male header pins on each side of the chip:

A0 (pin 21) – goes to Col 8	B7 (pin 8) – goes to Row 8
A1 (pin 22) – goes to Col 7	B6 (pin 7) – goes to Row 7
A2 (pin 23) – goes to Col 6	B5 (pin 6) – goes to Row 6
A3 (pin 24) – goes to Col 5	B4 (pin 5) – goes to Row 5
A4 (pin 25) – goes to Col 4	B3 (pin 4) – goes to Row 4
A5 (pin 26) – goes to Col 3	B2 (pin 3) – goes to Row 3
A6 (pin 27) – goes to Col 2	B1 (pin 2) – goes to Row 2
A7 (pin 28) – goes to Col 1	B0 (pin 1) – goes to Row 1

Each data pin goes to an LED row or column via a current limiting resistor. If you use the male header breakouts for other projects, please note that the bit order is not the same: on the left (portA) it goes from bit 0 to bit 7, but on the right (port B) it goes in the reverse direction from bit 7 to bit 0.