Accessing IO pins of port expander interfaced with R-Pi

e-Yantra Team

July 2, 2015

Contents

1 Objective

In this tutorial we will learn to access IO pins of a port expander (MCP23017) interfaced with an R-Pi and will also perform some experiments with LED's, switches and LCD interfaced with this IC.

2 Prerequisites

- Python programming skills
- An R-Pi (with I2C; I will be using version 2)
- Interfacing an MCP23017 IC with an R-Pi should be known

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B)
2. MCP23017
3. Power adapter
4. Connecting wires
5. LED's
6. 16x2 LCD display (I will be using model HD44780U)
7. Push button
8. Resistors (330 ohms)
9. Potentiometer (10k ohms)
10. Bread board

4 Software Requirement

1. PyScripter (version 2.7 or above)
2. Mobaxterm (for windows user)

5 Theory and Description

16x2 LCD Display :

The HD44780U dot-matrix liquid crystal display controller and driver LSI displays alphanumerics,Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.[3]

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The *Command* register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The *Data* register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. [4] **Pin**

Diagram

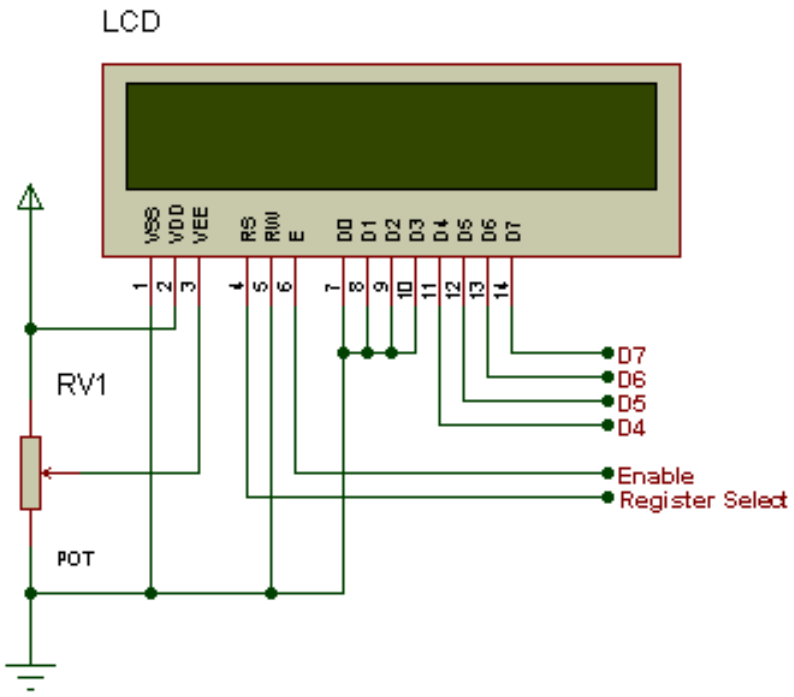

Figure 1: [5]

Pin Description

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	VEE
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7		DB0
8		DB1
9		DB2
10	8-bit data pins	DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight VCC (5V)	Led+
16	Backlight Ground (0V)	Led-

Ref: [4]

6 Experiments

In order to program the MCP23017 chip (using I2C protocol) in Python you must install the **smbus** package. Once you have installed the package you can now start programming the chip.

SMBus protocol commands:

- SMBus Read Byte: i2c_smbus_read_byte_data() This reads a single byte from a device, from a designated register.The register is specified through the Comm byte.

  S Addr Wr [A] Comm [A] S Addr Rd [A] [Data] NA P

- SMBus Write Byte: i2c_smbus_write_byte_data()

  This writes a single byte to a device, to a designated register. The register is specified through the Comm byte. This is the opposite of the Read Byte operation.

  S Addr Wr [A] Comm [A] Data [A] P

6.1 Interfacing an LED and a Switch to R-Pi using MCP23017 IC

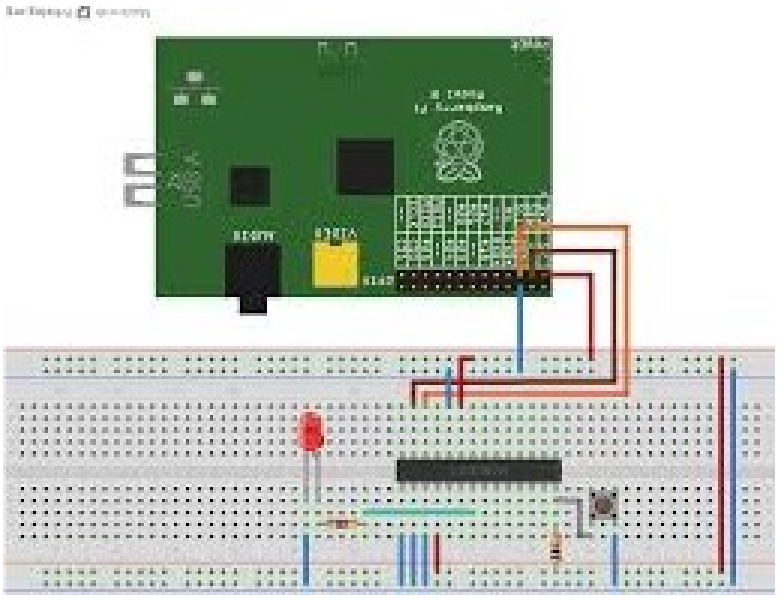

Figure 2: [2]

As shown in the figure:

- Pin 9 (VDD) is connected to 3.3V
- Pin 10 (VSS) is connected to Ground
- Pin 12 (SCL) is connected to Pin 5 on the Pi GPIO
- Pin 13 (SDA) is connected to Pin 3 on the Pi GPIO
- Pin 18 (Reset) should be set high for normal operation so we connect this to 3.3V
- Pins 15, 16 & 17 (A0-A2) determine the number assigned to this device. We are only using one device so we will give it a binary zero by setting all three of these pins to 0 (ground)
- Led is connected to GPA0 and switch is connected to GPA7

Code

```
import smbus # module to access i2c based interfaces
import time

bus = smbus.SMBus(1) # Rev 2 Pi uses 1

DEVICE = 0x20 # Device address (A0-A2)
IODIRA = 0x00 # Pin direction register
OLATA  = 0x14 # Register for outputs
GPIOA  = 0x12 # Register for inputs


# all bits of IODIRA register are set to 0 meaning GPA pins are outputs
bus.write_byte_data(DEVICE,IODIRA,0x00)

# Set all 7 output bits of port A to 0
bus.write_byte_data(DEVICE,OLATA,0)

try:
 while True:
  input = bus.read_byte_data(DEVICE,GPIOA) #read status of GPIO register
                                           #switch status
  if input & 0x80 == 0x80: # switch pressed i.e. input = True
    bus.write_byte_data(DEVICE,OLATA,1) # led glows
    time.sleep(1)

  # Set all bits to zero
  bus.write_byte_data(DEVICE,OLATA,0)

except KeyboardInterrupt:
        pass
        bus.write_byte_data(DEVICE,OLATA,0) # in case of keyboard
                        # interrupt set port A pins to zero
```

6.2 Interfacing an LCD to an R-Pi using MCP23017 IC

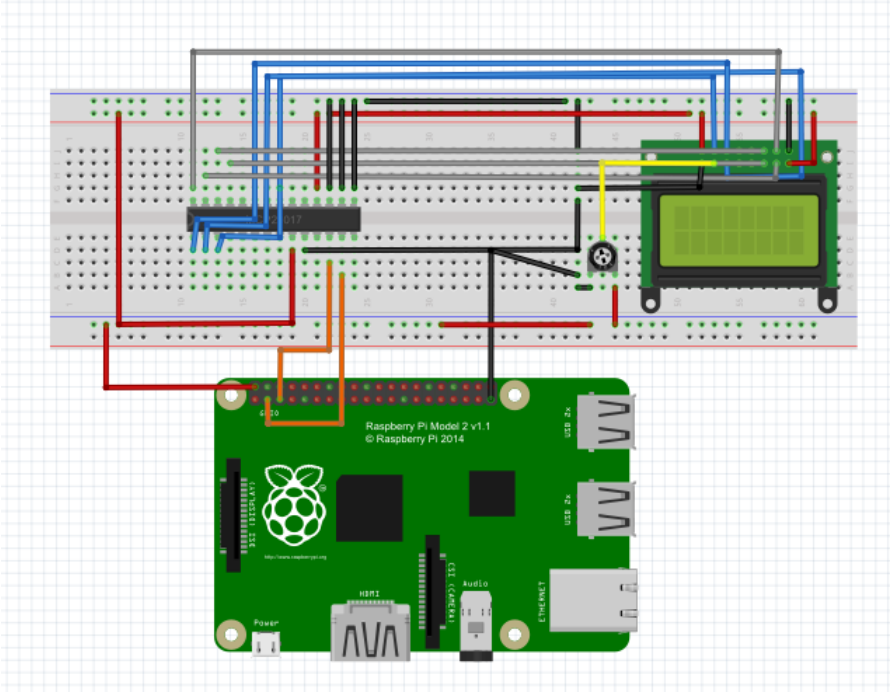

Figure 3: [2]

As shown in the figure:

- Pin 9 (VDD) is connected to 5V (Red)
- Pin 10 (VSS) is connected to Ground (Black)
- Pin 12 (SCL) is connected to Pin 5 on the Pi GPIO (Orange)
- Pin 13 (SDA) is connected to Pin 3 on the Pi GPIO (Orange)
- Pin 18 (Reset) should be set high for normal operation so we connect this to 5V (Red)
- Pins 15, 16 & 17 (A0-A2) determine the number assigned to this device. We are only using one device so we will give it a binary zero by setting all three of these pins to 0 (ground) (Black)
- RS,RW and Enable pins of the LCD are connected to GPB0,GPB1 and GPB2 respectively.
- Data pins D7,D6,D5 and D4 are connected to GPS7,GPA6,GPA5 and GPA4 respectively.

Code

```
import smbus # python module to access i2c based interfaces
import time

bus = smbus.SMBus(1) # Rev 2 Pi uses 1
DEVICE = 0x20 # Device address (A0-A2)
IODIRA = 0x00 # Pin direction register
IODIRB = 0x01 # Pin direction register
OLATA  = 0x14 # Register for outputs
OLATB  = 0x15 # Register for outputs

# Function name :initialize_mcp
# Input : None
# Output : Initializes MCP23017 IC
# Example call: initialize_mcp()
def initialize_mcp():
    # all bits of IODIRA and IODIRB register are set to 0 meaning GPA
        # and GPB pins are outputs
    bus.write_byte_data(DEVICE,IODIRA,0x00)
    bus.write_byte_data(DEVICE,IODIRB,0x00)

    # Set all 7 output bits of port A and port B to 0
    bus.write_byte_data(DEVICE,OLATA,0)
    bus.write_byte_data(DEVICE,OLATB,0)

    return

# Function name :cmdset4bit
# Input : None
# Output : Sets the 16x2 LCD in 4 bit mode
# Logic : Command 0x30 is sent thrice and command 0x20 is sent
#         once to initialize the LCD
# Example call: cmdset4bit()
def cmdset4bit():
    time.sleep(1.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)
    bus.write_byte_data(DEVICE,OLATA,0b00110000)
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

    time.sleep(1.0/1000)
    bus.write_byte_data(DEVICE,OLATA,0b00110000)
```

```
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

    time.sleep(1.0/1000)
    bus.write_byte_data(DEVICE,OLATA,0b00110000)
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

    time.sleep(1.0/1000)
    bus.write_byte_data(DEVICE,OLATA,0b00100000)
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

    return

# Function name : convert
# Input : A list of hexadecimal values (commands or data)
# Output : The function returns a list with corresponding decimal
#          equivalent of the hexadecimal values
# Logic : A 2 digit hexadecimal value is separated into 1 digit
#         each and a zero is appended at the end of each digit
#         (eg: '0f' is converted to '00' and 'f0')
# Example call: convert(lst)
def convert(lst):
    fc = []
    l = len(lst)
    for i in range(0,l):
        j = lst[i]
        temp1 = j[0] + '0'
        temp2 = j[1] + '0'
        t = int(temp1,16) # returns the decimal form of a string (temp1)
        u = int(temp2,16)
        fc.append(t)
        fc.append(u)

    return fc # output list

# Function name : stringconvert
# Input : A string i.e. data to be displayed on LCD
# Output : The function returns a list with corresponding hexadecimal
#          equivalent of every character in a string
```

```
# Logic : This function converts a character in a string into hex
#         format and appends the converted hex value into a list for
#         further processing. (eg: 'A' is converted to '41' i.e. the
#         hex value of the character )
# Example call: stringconvert(s)
def stringconvert(s):
    s1 = []
    l1 = len(s)
    for i in range(0,l1):
        j = ord(string[i]) # ord() function returns the decimal
                           # equivalent of an ascii character
        f = hex(j)
        s1.append(f)

    l2 = len(s1)
    newlst = []
    for i in range (0,l2):
        s2 = s1[i]
        s3 = s2[2]
        s4 = s2[3]
        sf = s3 + s4
        newlst.append(sf)

    return newlst # output list

# Function name : conversion
# Input : 2 lists a data list and a command list
# Output : The function returns 2 lists 'data' and 'cmd' that
#         contain data and commands in the form that can be
#         directly given as a pin output of MCP23017 IC
# Example call: conversion(com,s)
def conversion(com,s):
    conv_string = stringconvert(s)
    cmd = convert(com)
    data = convert(conv_string)

    return data,cmd # output lists

# Function name : lcd_start
# Input : None
# Output : MCP23017 is initialized and lcd is set in 4 bit mode.
# Example call: lcd_start()
def lcd_start():
    initialize_mcp()
```

```
    time.sleep(5.0/100)
    cmdset4bit()

# Function name : commandwrt
# Input : A list with commands in converted form (hexadecimal to
#         decimal format)
# Output : Commands are sent to LCD display one by one
# Logic : For sending commands RS pin of an LCD is set to 0 and R/W
#         pin of an LCD is set to 0(for write operation) and a high
#         to low enable pulse is applied every time a command is sent
# Example call: commandwrt(cmd)
def commandwrt(cmd):
    l = len(cmd)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)
    for i in range(0,l):
        time.sleep(1.0/1000)
        bus.write_byte_data(DEVICE,OLATA,cmd[i])
        bus.write_byte_data(DEVICE,OLATB,0b00000100)
        time.sleep(5.0/1000)
        bus.write_byte_data(DEVICE,OLATB,0b00000000)

    return

# Function name : datawrt
# Input : A list with data in converted form (string to decimal format)
# Output : Data (a string) is sent to LCD display one by one
# Logic : For sending data RS pin of an LCD is set to 1 and R/W pin
#         of an LCD is set to 0(for write operation) and a high to
#         low enable pulse is applied every time data is sent
# Example call: datawrt(data)
def datawrt(data):
    l = len(data)
    for i in range(0,l):
        time.sleep(1.0/1000)
        bus.write_byte_data(DEVICE,OLATA,data[i])
        bus.write_byte_data(DEVICE,OLATB,0b00000101)
        time.sleep(5.0/1000)
        bus.write_byte_data(DEVICE,OLATB,0b00000001)

    return

# Function name : lcdclear
# Input : None
# Output : LCD screen gets cleared(command = 0x01)
```

```
# Example call: lcdclear()
def lcdclear():
    time.sleep(1.0/1000)
    bus.write_byte_data(DEVICE,OLATA,0b00000000)
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

    bus.write_byte_data(DEVICE,OLATA,0b00010000)
    bus.write_byte_data(DEVICE,OLATB,0b00000100)
    time.sleep(5.0/1000)
    bus.write_byte_data(DEVICE,OLATB,0b00000000)

try:
  lcd_start()
  com = ['28','01','0f','06'] # a list of commands
  s = "ABCD" # data to be displayed

  data,cmd = conversion(com,s)
  commandwrt(cmd)
  datawrt(data)


except KeyboardInterrupt:
        pass
        lcdclear()
```

Note: This code can be imported into any other python code for further usage (eg: for displaying sensor values on lcd).

7 Appendix

7.1 Raspberry Pi 2 Pin-out Diagram

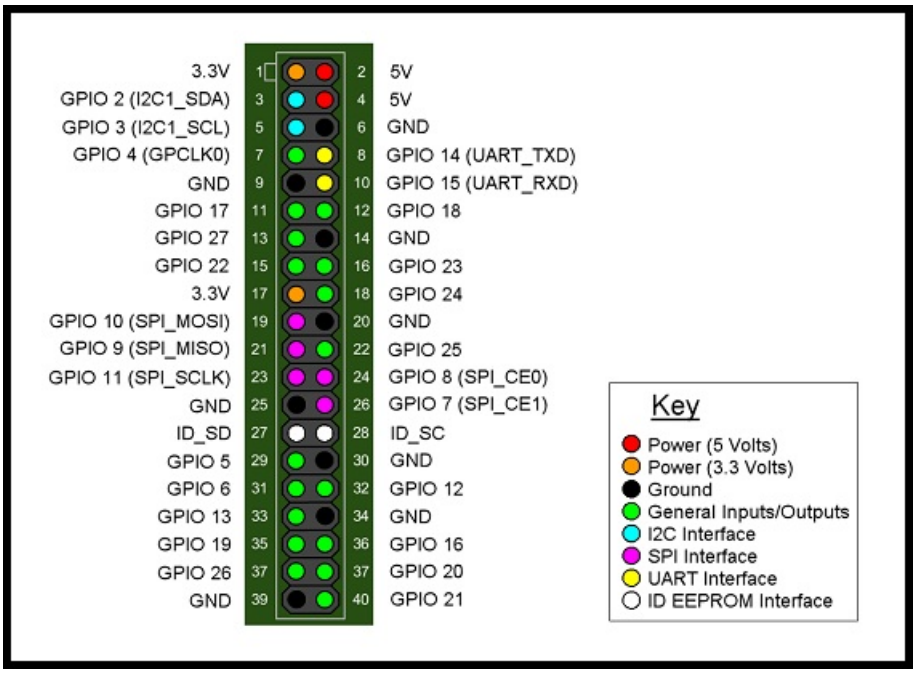

Figure 4: [5]

7.2 MCP23017 datasheet

`http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf`

8 References

1. https://www.kernel.org/doc/Documentation/i2c/smbus-protocol
2. http://dangerousprototypes.com/wp-content/media/2013/04/mcp23017test_bb-600x458.png
3. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
4. http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet
5. http://data.designspark.info/uploads/images/53bc258dc6c0425cb44870b50ab30621