Interface IO port expander with R-Pi

e-Yantra Team

June 23, 2015

Contents

1 Objective

In this tutorial we will learn how to interface a port expander i.e. MCP23017 IC with an R-Pi to increase the number of input/output pins.

2 Prerequisites

One should have:

- Python programming skills
- Some knowledge about performing basic programs to access GPIO pins of an R-Pi
- Some basic information regarding different serial communication protocols

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 2 Model B)
2. Power adapter
3. MCP23017 IC
4. Connecting wires
5. Bread board

4 Software Requirement

MobaXterm (for windows user)

5 Theory and Description

The no. of GPIO pins in an R-Pi is less(merely 26 in R-Pi 2) and so interfacing sensors and other modules would consume most of the pins thus setting a limit for I/O access. Therefore in order to increase the no. of GPIO pins we can interface a port expander to an R-Pi.

A port expander is a hardware device designed to allow a user to utilize more than one device on a single port at one time. For example, the device may allow seven devices to connect to one serial port.[1]

5.1 I2C

IC (Inter-Integrated Circuit), pronounced I-squared-C, is a multi-master, multi-slave, single-ended, serial computer bus used for attaching lower-speed peripheral ICs to processors and micro controllers.[2]

The two IC signals are called 'serial data (SDA) and serial clock (SCL). There is no need of chip select (slave select) or arbitration logic in this. Virtually any number of slaves and any number of masters can be connected onto these 2 signal lines and communicate between each other using a protocol that defines:

- 7-bits slave addresses: each device connected to the bus has got such a unique address
- data divided into 8-bit bytes
- a few control bits for controlling the communication start, end, direction and for an acknowledgement mechanism.

The data rate has to be chosen between 100 kbps, 400 kbps and 3.4 Mbps, respectively called standard mode, fast mode and high speed mode. Some IC variants include 10 kbps (low speed mode) and 1 Mbps (fast mode +) as valid speeds.

Physically, the IC bus consists of the 2 active wires SDA and SCL and a ground connection. The active wires are both bi-directional.[3]

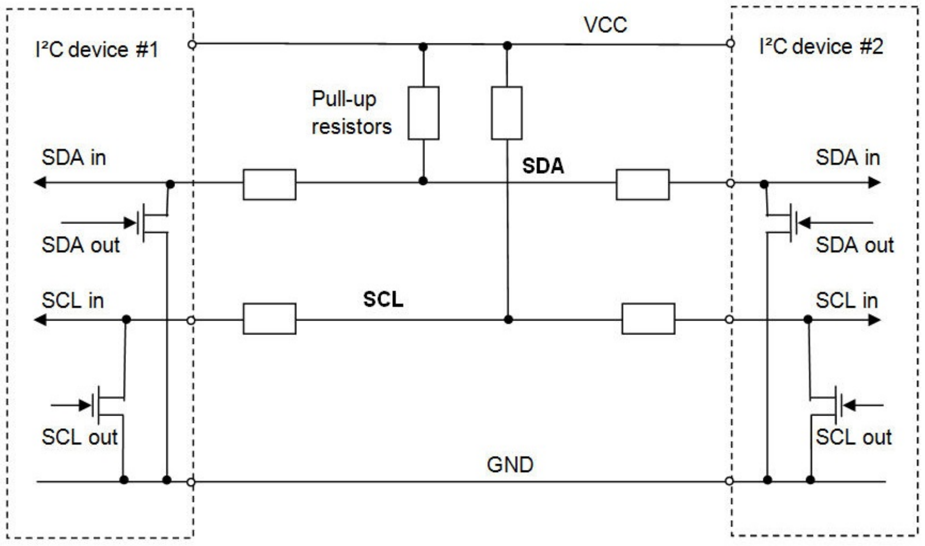


Figure 1: [3]

The I2C protocol specification states that the IC that initiates a data transfer on the bus is considered the Bus Master. Consequently, at that time, all the other ICs are regarded to be Bus Slaves.

Data transfer using I2C

1. First, the master will issue a START condition. This acts as an Attention signal to all of the connected devices. All ICs on the bus will listen to the bus for incoming data.

2. Then the master sends the ADDRESS of the device it wants to access, along with an indication whether the access is a Read or Write operation. Having received the address, all ICs will compare it with their own address. If it doesnt match, they simply wait until the bus is released by the stop condition. If the address matches, however, the chip will produce a response called the ACKNOWLEDGE signal.

3. Once the master receives the acknowledgement, it can start transmitting or receiving DATA.When all is done, the master will issue the STOP condition. This is a signal that states the bus has been released and that the connected ICs may expect another transmission to start any moment.

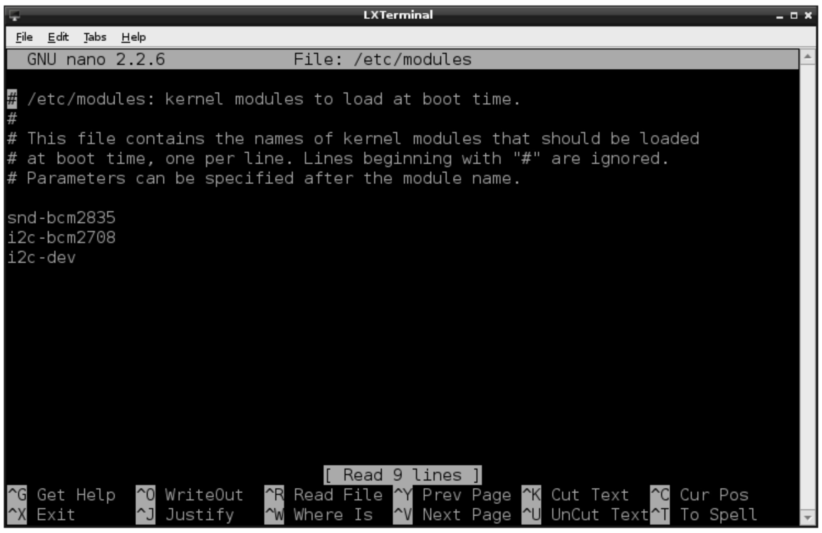
4. When a master wants to receive data from a slave, it proceeds the same way, but sets the R/W bit at a logical one. Once the slave has acknowledged the address, it starts sending the requested data, byte

by byte. After each data byte, it is up to the master to acknowledge the received data.[3]

5.2 Enabling I2C interface on R-Pi

1. Open MobaXterm then open LXTerminal(if using LXDE desktop) else on the terminal window type *sudo nano /etc/modules*

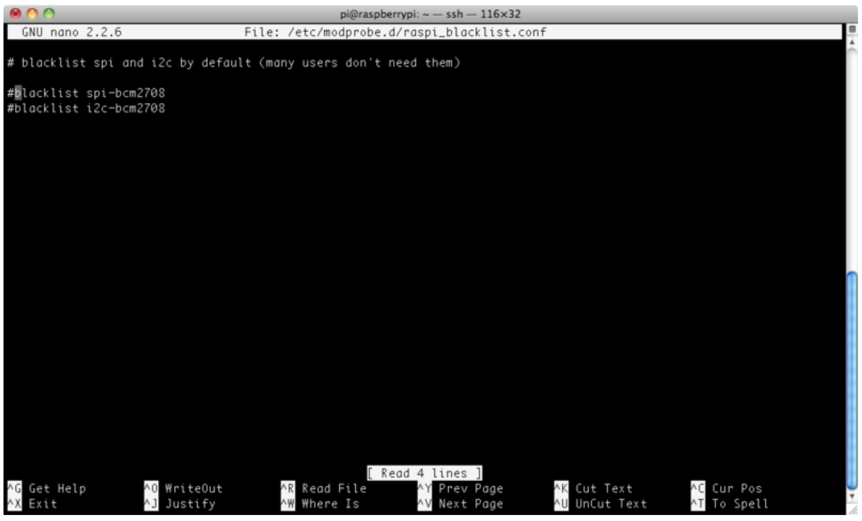
2. A file opens, add the following two lines to the end of the file: *i2c-bcm2708* and *i2c-dev*. Then save the file by Ctrl+X , Y.



```
  GNU nano 2.2.6              File: /etc/modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
i2c-bcm2708
i2c-dev






                                [ Read 9 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Figure 2: [4]

3. In case you have a file called */etc/modprobe.d/raspi-blacklist.conf* then edit the file as shown below(comment the 2 lines)



```
pi@raspberrypi: ~ — ssh — 116×32
  GNU nano 2.2.6                  File: /etc/modprobe.d/raspi_blacklist.conf

# blacklist spi and i2c by default (many users don't need them)

#blacklist spi-bcm2708
#blacklist i2c-bcm2708

                                             [ Read 4 lines ]
^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

Figure 3: [4]

Note: In case you don't have this file continue with the further steps.

4. Recent versions of Raspberry Pi need an update to*/boot/config.txt* file. Type the command *sudo nano /boot/config.txt* and edit the file by adding the following two lines at the end of the file: $dtparam\bar{i}2c1\bar{o}n$ and $dtparam\bar{i}2c\_arm\bar{o}n$

5. After all this is done reboot the system by using the command *sudo reboot*

5.3 MCP23017

It is a 16 bit I/O expander with serial interface. Some of its features are:

- 16-bit remote bidirectional I/O port (I/O pins default to input)
- High-speed I2C interface (100 kHz, 400 kHz, 1.7 MHz)
- Three hardware address pins to allow up to eight devices on the bus
- Two pins to communicate with a master controller i.e. SCL and SDA

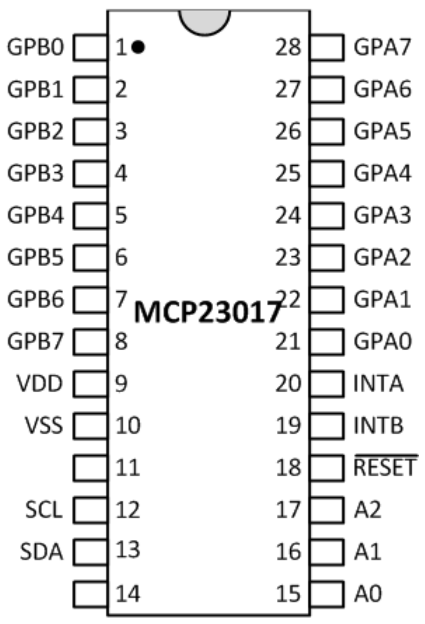


Figure 4: [5]

For further information you refer the Appendix section.

6 Experiment

In order to start programming the port expander(MCP23017) , we need to interface the IC with R-Pi in the following way:

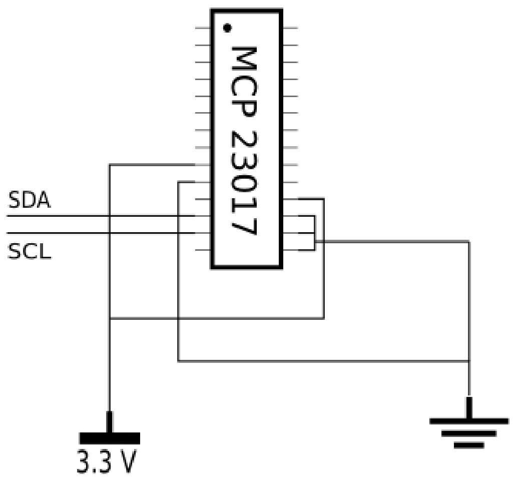


Figure 5: [6]

- Connect the VDD pin and RESET on MCP23017 to 3.3v pin on an R-Pi
- Connect the VSS pin on MCP23017 and A0,A1,A2 pin to GND pin on an R-Pi(If you are using 1 port expander IC then we connect the 3 address pins A0, A1 and A2 pins to GND)
- Connect the SCL and SDA pins from the MCP23017 IC to the respective pins on R-Pi

Please refer the Appendix section before making the necessar connections.

After making all the necessary connections type the following command on the terminal *sudo i2cdetect -y 1* (to see the device addresses of MCP23017 IC's interfaced)

Note : For a version 1 R-Pi with RAM lesser than 512MB use the command *sudo i2cdetect -y 1*

7 Appendix

7.1 Raspberry Pi 2 Pin-out Diagram

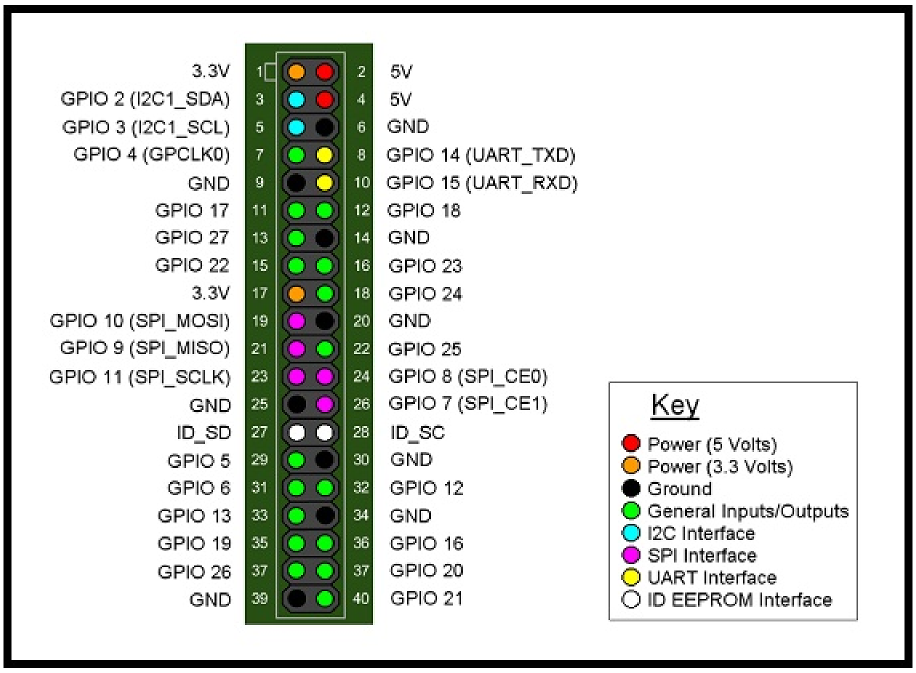


Figure 6: [7]

7.2 MCP23017 datasheet

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

8 References

1. http://www.computerhope.com/jargon/p/portexpa.htm
2. https://en.wikipedia.org/wiki/I%C2%B2C
3. http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
4. https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c
5. https://www.mathworks.com/examples/matlab/4547-add-digital-i-o-pins-to-raspberry-pi-hardware-using-mcp23017
6. https://camo.githubusercontent.com/c80be3d0c9e3146cd09e15b7ddf07dbb8b7d40b8/687474703a2f2f692e696d6775722e636f6d2f4575524e722e706e67
7. http://data.designspark.info/uploads/images/53bc258dc6c0425cb44870b50ab30621