SPI communication with R-PI

e-Yantra Team

February 29, 2016

Contents

1 Objective

In this tutorial we will learn the SPI (Serial Peripheral Interconnect) communication in an R-Pi.

2 Prerequisites

One should know:

- To know how to access the pins of an R-Pi.
- Some basic information regarding SPI protocol.
- Python programming skills.

3 Hardware Requirement

1. Raspberry Pi (I will be using Version 1 Model B+)
2. Power adapter(5V)

4 Software Requirement

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

5 Theory and Description

5.1 Serial communication

Serial interfaces stream their data, one single bit at a time. These interfaces can operate on as little as one wire.

Figure 1: Asynchronous serial communication

Figure shows the asynchronous serial communication because no single clock is shared between devices, hence there is no control over when data is sent or any guarantee that both sides are running at precisely the same rate.

5.2 SPI

- The Serial Peripheral Interface (SPI) is a **synchronous** communication protocol used to transfer data between micro-computers like the Raspberry Pi and peripheral devices.[2]

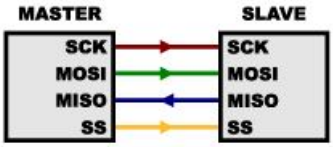

Figure 2: SPI

- These peripheral devices may be either sensors or actuators.

SPI uses 4 separate connections to communicate with the target device. These connections are:

1. Serial clock (CLK)
2. Master Input Slave Output (MISO)
3. Master Output Slave Input (MOSI) and
4. Chip Select (CS).

- The **Clock pin** sense pulses at a regular frequency, the speed at which the Raspberry Pi and SPI device agree to transfer data to each other. Master generates the clock.

- The **MISO** pin is a data pin used for the master (in this case the Raspberry Pi) to receive data from the slave. Data is read from the bus after every clock pulse.[3]

- The **MOSI** pin sends data from the Raspberry Pi to the slave.

- Finally, the **Chip Select line** chooses which particular SPI device is in use. If there are multiple SPI devices, they can all share the same CLK, MOSI, and MISO. However, only the selected device has the Chip Select line set low, while all other devices have their CS lines set high. A high Chip Select line tells the SPI device to ignore all of the commands and traffic on the rest of the bus.

5.3 Multiple slaves

There are two ways of connecting multiple slaves to an SPI bus:

- Each slave will need a separate SS line. To talk to a particular slave, master will make that slave?s SS line low and keep the rest of them high (you don?t want two slaves activated at the same time, or they may both try to talk on the same MISO line resulting in garbled data).

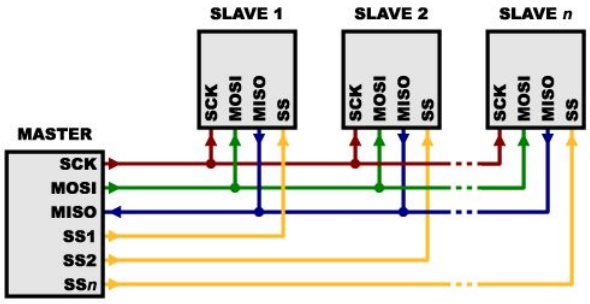

Figure 3: Multiple slave on SPI

- daisy-chained structure, with the MISO (output) of one going to the MOSI (input) of the next. In this case, a single SS line goes to all the slaves. Once all the data is sent, the SS line is raised, which causes all the chips to be activated simultaneously. This is often used for daisy-chained shift registers and addressable LED drivers.

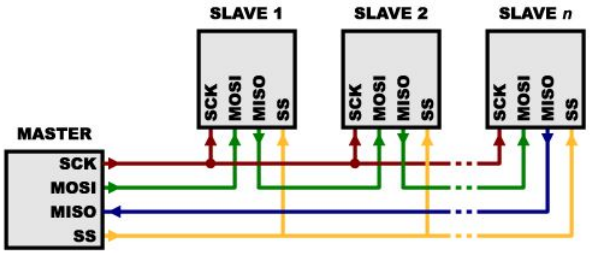

Figure 4: Daisy chain structure

5.4 SPI on R-Pi

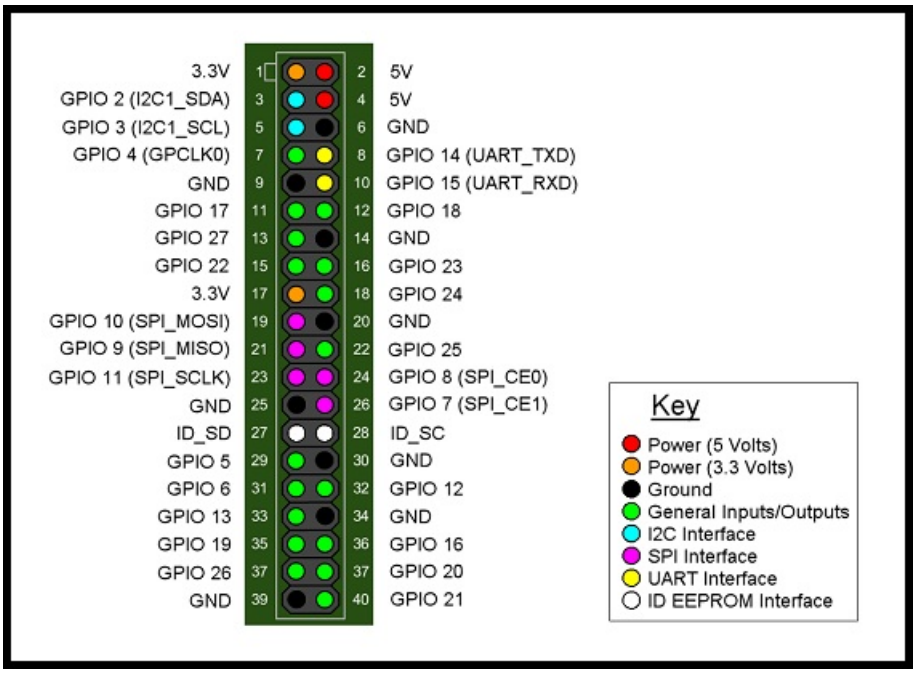

Figure 5: Pin Header on R-Pi

The Raspberry Pi has SPI bus which can be enabled on pin no 19,21,23,24 and 26 shown by pink dots. It is a synchronous serial data link standard and is used for short distance single master communication between devices. As far as the Pi is concerned this is usually relevant to certain sensors and add-on boards.

The default Raspbian image disables SPI by default so before you can use it the interface must be enabled. This can be done uses either of two methods. I?ll describe both methods but the first one is probably easier and quicker.

6 Enabling SPI on R-Pi

6.1 Method 1: Enabling using raspi-config

Run the following command:
Sudo raspi-config

This will launch the raspi-config utility. Select option 8 “Advanced Options”.

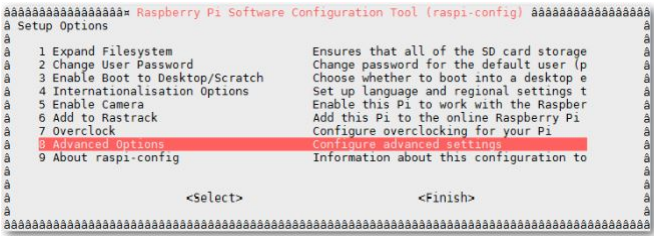

Figure 6: [1]

Select the “SPI” option.

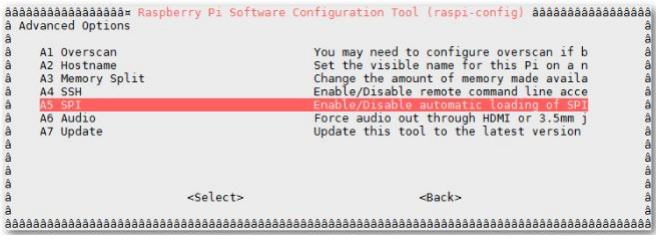

Figure 7: [2]

Set the option to “Yes”.

Figure 8: [3]

Select “OK”.

Figure 9: [4]

Select “Finish”.

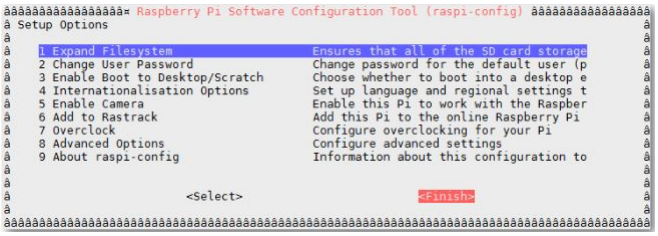

Figure 10: [5]

Reboot for the changes to take effect :
sudo reboot

SPI is now enabled.

6.2 Method 2: Enabling by editing file manually

To enable hardware SPI on the Pi we need to make a modification to a system file :
sudo nano /boot/config.txt

Add the following line at the bottom :
dtparam=spi=on

Use CTRL-X, then Y, then RETURN to save the file and exit. Reboot using the following :
sudo reboot

6.3 Checking if SPI is enabled

To check if the SPI module is loaded by the system run the following command :
lsmod

You should see "spi_bcm2708" or "spi_bcm2835" listed in the output. You can use the following command to filter the list and make it easier to spot the spi entry :

$$lsmod \mid grepspi\_ \tag{1}$$

SPI is now enabled.

6.4 Install Python SPI Wrapper

In order to read data from the SPI bus in Python we can install a library called 'py-spidev'. To install it we first need to install 'python-dev':
sudo apt-get install python2.7-dev

Then to finish we can download 'py-spidev' and compile it ready for use, run the following commands:
wget https://github.com/Gadgetoid/py-spidev/archive/master.zip
unzip master.zip
rm master.zip
cd py-spidev-master
sudo python setup.py install
cd ..

You should now be ready to either communicate with add-on boards using their own libraries (e.g. the PiFace) or other SPI devices.

7 Programming for SPI

7.1 Settings

There are some options when setting up the interface. These options must match those of the device talking to; check the device?s datasheet to see what it requires.

- **bits_per_word:** Property that gets / sets the bits per word.
  Range - 8..16.
- **cshigh:** Property that gets / sets if the CS is active high.
- **loop:** Property that gets / sets loopback configuration. This is used to test the pcb for example. Anything that gets received will be echoed back.
- **lsbfirst:** Property that gets / sets if LSB should be transferred first or last.
  NOTE: Needs Boolean value. However, this property seems to be read only and the value dependences on the endianess of the controller / cpu. The Raspberry Pi can only send MSB first, so you may need to convert the byte(s) manually.
- **max_speed_hz:** Property that gets / sets the maximum bus speed in Hz.
- **mode:** Property that gets / sets the SPI mode as two bit pattern of Clock Polarity and Phase [CPOL—CPHA]
  Range - 0 to 3.
- **threewire:** Property that gets / sets SI/SO signals shared so you have only 1 data line

7.2 Commands

1. **open:**
   Syntax: open(bus, device)
   Description: Connects the object to the specified SPI device. open(X,Y) will open /dev/spidev-X.Y.

2. **readbytes:**
   Syntax: read(len)
   Returns: [values]
   Description: Read len bytes from SPI device.

3. **writebytes:**
   Syntax: write([values])
   Description: Write bytes to SPI device.

4. **xfer:**
   Syntax: xfer([values])
   Returns: [values]
   Description: Perform SPI transaction.CS will be released and reactivated between blocks. delay specifies delay in sec between blocks.

5. **xfer2:**
   Syntax: xfer2([values])
   Returns: [values]
   Description: Perform SPI transaction. CS will be held active between blocks.

6. **close:**
   Syntax: close()
   Description: Disconnects the object from the interface.

Example: Open SPI and write a byte (0xAA) to it each 0.1 second until you cancel it with Ctrl+C.

Code

```
import spidev
import time

spi = spidev.SpiDev()                   # create spi object
spi.open(0, 1)                          # open spi port 0, device (CS) 1
try:
 while True:
 resp = spi.xfer2([0xAA])               # transfer one byte
 time.sleep(0.1)                        # sleep for 0.1 seconds
 #end while
except KeyboardInterrupt:               # Ctrl+C pressed, so
 spi.close()                            #   close the port before exit
#end try
```

8 References

1. https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
2. www.en.wikipedia.org/wiki/SerialPeripheralInterface
3. www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/
4. developers.google.com/edu/python/
5. http://tightdev.net/SpiDev_Doc.pdf