Raspberry Pi Recipes Part #4

Set the Table for Raspberry (S)PI

Last time we looked at the UART Serial Interface of the Raspberry Pi's Expansion Header. In this project we'll take a look at one of Raspberry Pi's other serial interfaces—the SPI Bus.

By **Tony Dixon** (UK)

SPI Interface

The Serial Peripheral Interface (SPI) is the second of the three serial interfaces you'll find on the Raspberry Pi's Expansion Header. The other two interfaces are the UART Serial Interface (see Part 3) and the I²C interface.

Table 1 details the Expansion Header signals. The SPI interface is found on pins 19 (MOSI), 21 (MISO), and 23 (SCK), with the two SPI Chip Enable signals found on pins 24 (CE0) and 26 (CE1).

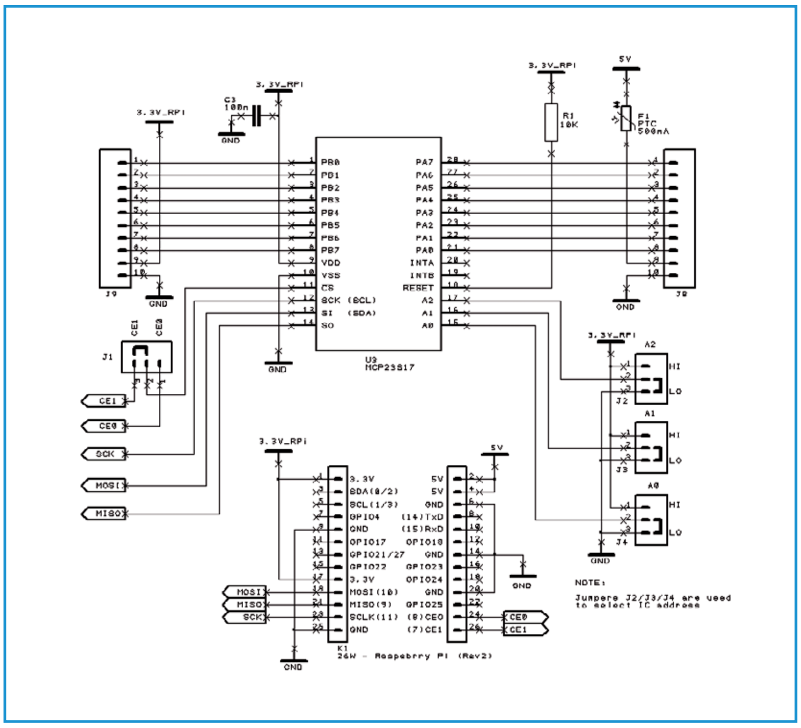
SPI is designed to interface other devices with a minimum number of pins in a Master/Slave arrangement. For our examples the Raspberry Pi will always be the SPI Master. Physically, the SPI interface normally requires four signals to function properly. These signals are *Master Out, Slave In* (MOSI), *Master In, Slave Out* (MISO), *Serial Clock* (SCK) and *Chip Enable* (CEx).

Port Expander Hardware

In our Raspberry Pi SPI project we'll be expanding the number of the RPI GPIOs by adding a Port Expander. In particular we'll be using a 16-channel MCP23S17 Port Expander from Microchip.

Figure 1 shows a simple circuit. The chip is connected to the RPi's SPI interface, and we have the option of using either one of the two SPI chip enable signals (CE0 or CE1) via a jumper.

Figure 1. Schematic for Raspberry Pi MCP23S17 Port Expander.



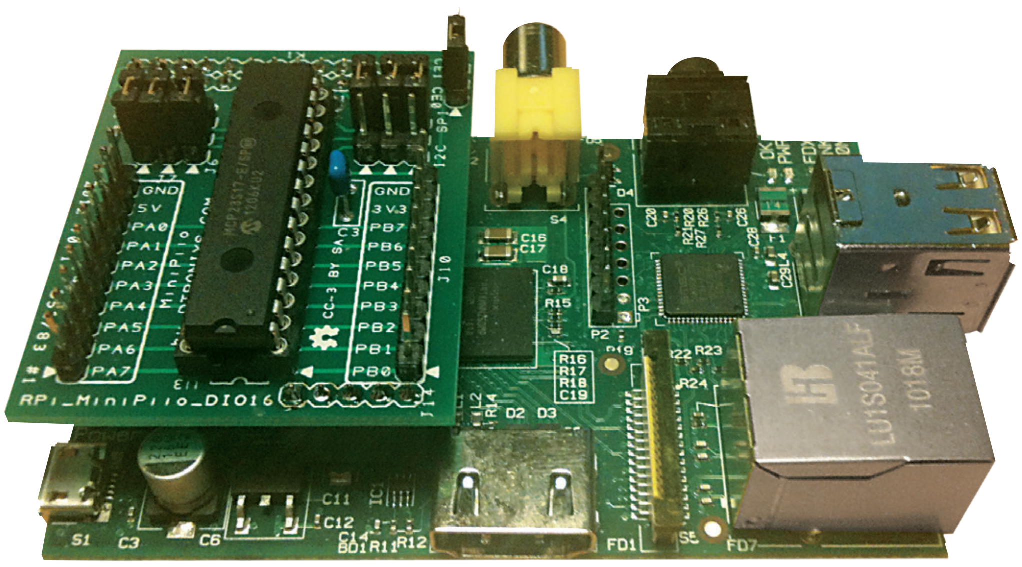


Figure 2.
Pi and MCP23S17 Add-On Board.

Figure 2 depicts our hardware setup where we are using a small add-on board [3] to provide our MC23S17 interface.

Installing Python's SPI Library

Our programming language of choice for these project examples is Python 2. As we discovered last time, Python is already installed as standard in the Raspbian distribution. However, there is no provision for the SPI interface. To fix this we will need to install the SPI Python wrapper / library, so let's start an LXterminal session (as shown in **Figure 3**) and type the following commands:

```
cd ~

git clone git://github.com/doceme/
   py-spidev

cd py-spidev/

sudo python setup.py install

{or
sudo apt-get install git-core
   python-dev
sudo apt-get install python-pip
sudo pip install spidev)
```

Once installed, we'll need to do a little housekeeping and tell Raspbian that we intend to use the hardware SPI interface. The hardware SPI is disabled by default, so we have to change this by editing the blacklist file:

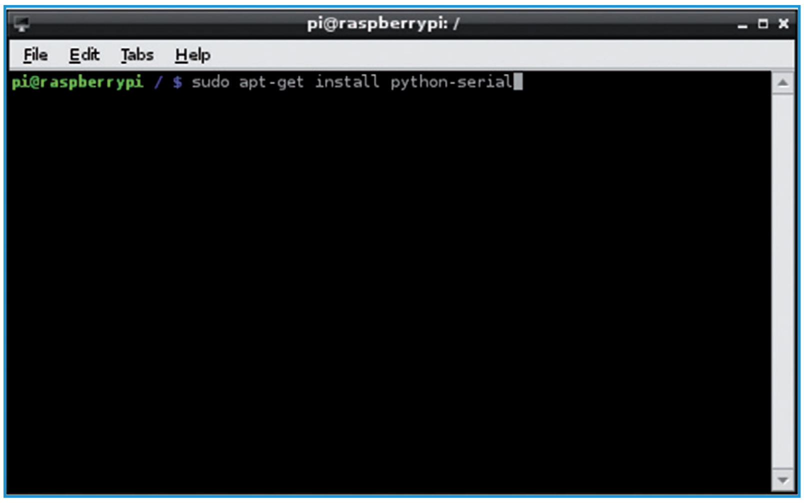
```
sudo nano /etc/modprobe.d/
   raspi-blacklist.conf
```

Find the text line with ***blacklist spi-bcm2708***, insert a # (hash) at the start of the line to comment out the statement, and then save the file. Once saved, let's do a quick reboot by typing:

```
sudo reboot
```

Once we've rebooted, we can start a new LXTerminal session and type...

Figure 3.
LXTerminal.

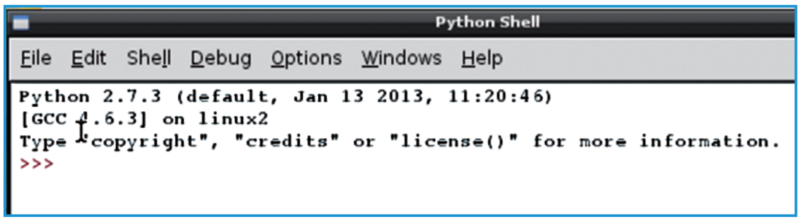


```
pi@raspberrypi / $ sudo apt-get install python-serial
```

```
ls /dev/spi*
```

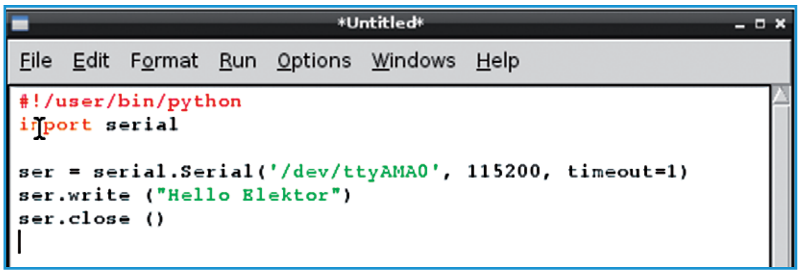
...to check that we have two SPI devices list (one for each SPI Chip Select signal), and we should have:

```
/dev/spidev0.0
/dev/spidev0.1
```



```
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>>
```

Figure 4. IDLE Python Shell.



```
#!/user/bin/python
import serial

ser = serial.Serial('/dev/ttyAMA0', 115200, timeout=1)
ser.write ("Hello Elektor")
ser.close ()
```

Figure 5. IDLE Editor.

Example Program: mcp23s17.py

With spidev installed we're now going to write a small test program to illuminate LEDs wired to the Port Expander GPIO.

Double-click the IDLE icon on your Pi's desktop to start the Python Shell and IDLE (**Figure 4**).

Select File Option from the menu and create a new program. This will start the IDLE editor.

In the IDLE editor, **Figure 5**, type the program as shown in the **Listing**.

Once you've typed the program, make sure you save it, then switch to an LXTerminal session and type the following command to make your program an executable:

```
chmod +x mcp23s17.py
```

Once done, you can run your program by typing the following command:

```
sudo ./mcp23s17.py
```

(130211)

Listing

```
#!/usr/bin/python

import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)

while True:
    spi.xfer([0,0,0])     # turn all lights off
    time.sleep(1)
    spi.xfer([1,255,254]) # turn all lights on
    time.sleep(1)
```

Quick Summary of spidev commands

`spi.open (0,0)`	opens SPI bus 0 using CE0.
`spi.open (0,1)`	opens SPI bus 0 using CE1.
`spi.close ()`	disconnects the object from the interface.
`spi.writebytes ([array of bytes])`	writes an array of bytes to SPI device.
`spi.readbytes (len)`	reads *len* bytes from SPI device.
`spi.xfer2 ([array of bytes])`	sends an array of bytes keeping the CEx asserted the whole time.
`spi.xfer ([array of bytes])`	sends an array of bytes de-asserting and re-asserting the CEx with every byte transmitted.

Internet Links

[1] www.raspberrypi.org

[2] www.github.com/doceme/py-spidev

[3] www.dtronixs.com

Table 1. Expansion Header Pin Out

Pin Name	Pin Function	Alternative	RPi.GPIO
P1-02	5.0V	-	-
P1-04	5.0V	-	-
P1-06	GND	-	-
P1-08	GPIO14	UART0_TXD	RPi.GPIO8
P1-10	GPIO15	UART0_RXD	RPi.GPIO10
P1-12	GPIO18	PWM0	RPi.GPIO12
P1-14	GND	-	-
P1-16	GPIO23		RPi.GPIO16
P1-18	GPIO24		RPi.GPIO18
P1-20	GND	-	-
P1-22	GPIO25		RPi.GPIO22
P1-24	GPIO8	**SPI0_CE0_N**	RPi.GPIO24
P1-26	GPIO7	**SPI0_CE1_N**	RPi.GPIO26

Pin Name	Board Revision 1		Board Revision 2	
	Pin Function	Alternative	Pin Function	Alternative
P1-01	3.3V	-	3.3V	-
P1-03	GPIO0	I2C0_SDA	GPIO2	I2C1_SDA
P1-05	GPIO1	I2C0_SCL	GPIO3	I2C1_SCL
P1-07	GPIO4	GPCLK0	GPIO4	GPCLK0
P1-09	GND	-	GND	-
P1-11	GPIO17	RTS0	GPIO17	RTS0
P1-13	GPIO21		GPIO27	
P1-15	GPIO22		GPIO22	
P1-17	3.3V	-	3.3V	-
P1-19	GPIO10	SPI0_MOSI	GPIO10	SPI0_MOSI
P1-21	GPIO9	SPI0_MISO	GPIO9	SPI0_MISO
P1-23	GPIO11	SPI0_SCLK	GPIO11	SPI0_SCLK
P1-25	GND	-	GND	-

Note: I2C0_SDA and I2C0_SCL (GPIO0 & GPIO1) and I2C1_SDA and I2C1_SCL (GPIO2 & GPIO3) have 1.8-kΩ pull-up resistors to 3V3.