Raspberry SPI

http://www.100randomtasks.com/simple-spi-on-raspberry-pi

Vorbereitungen auf dem Raspi

- Blacklist editieren:
  sudo nano /etc/modprobe.d/raspi-blacklist.conf
  '#' vor spi-bcm2708 einfügen (auskommentieren)

- Neu booten:
  sudo reboot

- Prüfen ob der SPI-Treiber geladen wurde:
  lsmod
  Es wird eine Liste der Treiber ausgegeben, spi-bcm2708 muß darin vorkommen.

- Die Python-SPI-Tools laden:
  sudo apt-get update
  sudo apt-get install python-dev

- Installation des SPI Wrappers:
  mkdir python-spi
  cd python-spi
  wget https://raw.github.com/doceme/py-spidev/master/setup.py
  wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
  sudo python setup.py install

- Test der python-spi:
  In Idle: import spidev darf keinen Fehler melden

Test SPI

Hier gibt es ein C-Programm zum Testen des Loopbacks (MISO-MOSI verbunden):

http://www.brianhensley.net/2012/07/getting-spi-working-on-raspberry-pi.html

Das korrekte Funktionieren des SPI-Taktes am CLK-Pin wird damit natürlich nicht geprüft, da der Takt nur intern benutzt wird.

Grundwissen SPI

Kommunikation über 3 Leitungen:
SCLK = Serial clock
MOSI = Master OT, Slave IN
MISO = Master IN, Slave OUT

Es können mehrere Slaves parallel am Bus hängen, der angesprochene Slave wird durch CS (Chip Select) = L ausgewählt.

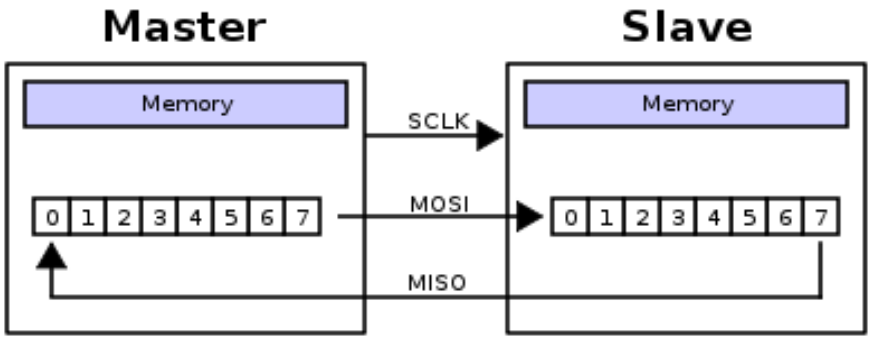
Der Raspi hat dafür 2 Chip Enable – Anschlüsse CE0 und CE1.

Es gibt 4 verschiedene SPI Modes für den Takt, der Raspi benutzt Mode 0,0.
Das heißt:

- CLK = L im Ruhezustand
- Übernahme der Datenbits mit der positiven Flanke

Default-Taktfrequenz des Raspi: 500kHz

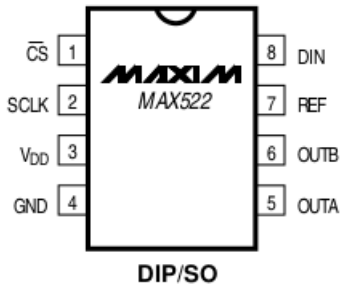
Das Senden und Empfangen geschieht immer gleichzeitig, bei jedem Takt wird ein Bit vom MOSI des Masters versendet und über MISO ein Bit des Slaves empfangen:

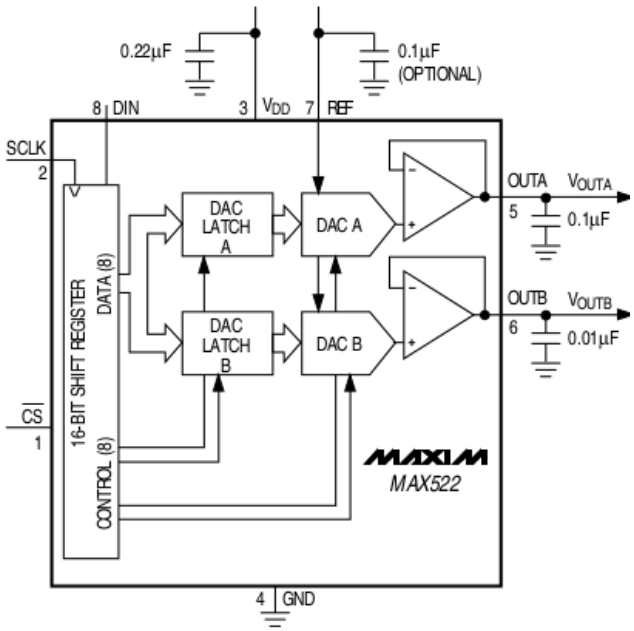


(Bildquelle: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus#Data_transmission)

Konsequenz: schickt der Master ein Kommando an den Slave, erhält er die Antwort auf die vorherige Anforderung!

Beispiel 1: MAX522 DA-Wandler





Verbindungen Raspi – MAX522:

+3V3 – VDD
CS0 – CS\
CLK – SCLK
MOSI – DIN

Am MAX522:

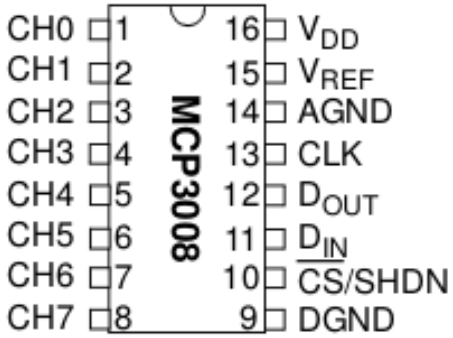
REF – VDD (Referenz 3.3V)
OUTA gepuffert mit C = 100nF! (siehe Schaltplan / Datenblatt) (sonst wilde Schwingungen)

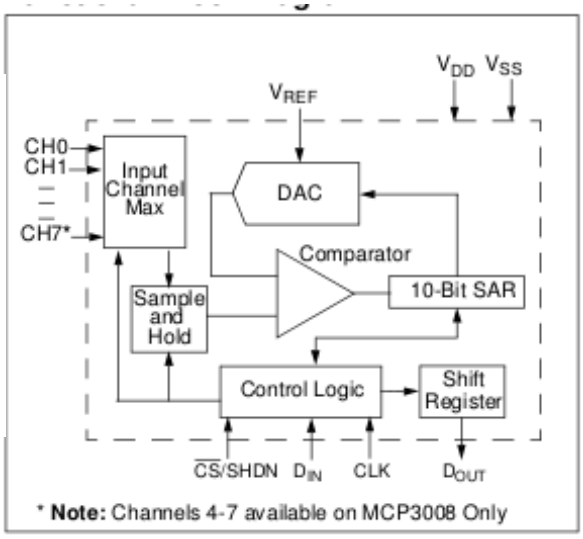
Das Beispielprogramm fragt einen Wert 0...255 ab und schickt diesen an den DA-Wandler A. Die gesendeten Daten bestehen immer aus einem Kommandobyte und dem Analogwert 0...255. Das Kommandobyte 49 im Beispiel ergibt sich aus dem Datenblatt (benutze DAC A, schalte B aus).

```
import spidev
spi = spidev.SpiDev()
spi.open(0,0)       # use chip select CE0, channel 0

while True:
     strvalue = raw_input("value (0...255, q=quit):")

     if strvalue == 'q':
          break
     else:
          value = int (strvalue)
          dummy = spi.xfer2([49,value])
spi.close()
```

Achtung: Bug!
Wenn vorher im gleichen oder einem anderen Programm ein Pin der SPI-Schnittstelle benutzt wurde, funktioniert SPI nicht. (Getestet 6.11.2013 mit SCLK)
In diesem Fall muß der Raspi zuerst neu gebootet werden.

Beispiel 2: MCP3008 AD-Wandler





Verbindungen Raspi – MCP3008:

+3V3 – VDD
CS0 – CS\
CLK – CLK
MOSI – DIN
MISO - DOUT

Am MCP3008:

REF – VDD (Referenz 3.3V)

Test mit einem Potentiometer an 3.3V, Schleifer am CH0-Eingang

Das Beispielprogramm liest jede Sekunde einen Wert von CH0 und gibt ihn aus:

```
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)           # use chip select CE0, channel 0

while True:
     try:
          chan = 0
          response = spi.xfer2([1, 128+(chan<<4), 0])
          value = (response[1] & 3) * 256 + response[2]
          print value
          time.sleep(1)
     except KeyboardInterrupt:
          spi.close()
```

Zum Vertändnis der Kommandos und Auswertung siehe Datenblatt.
Ganz kurz:
Der Master sendet und empfängt immer 3 Bytes.

Gesendete Bytes:
das erste ist immer 1, beim zweiten wird der Kanal ausgewählt, das dritte ist egal.

Empfangene Bytes:
das erste wird verworfen, dass zweite enthält rechtsbündig die beiden höchstwertigsten Bits (herausmaskiert durch "&3"), die restlichen Bits stehen im dritten empfangenen Byte.