I^2C am Raspberry Pi

Guido Schmitz

Pi And More // 23. August 2012

Overview

1. Was ist I^2C?
   - Physikalisch
   - Logisch
2. Wie benutzen wir I^2C?
   - Der Portexpander aus dem Starterset
   - Beispielaufbau
   - Einrichtung (Software)
   - Inbetriebnahme

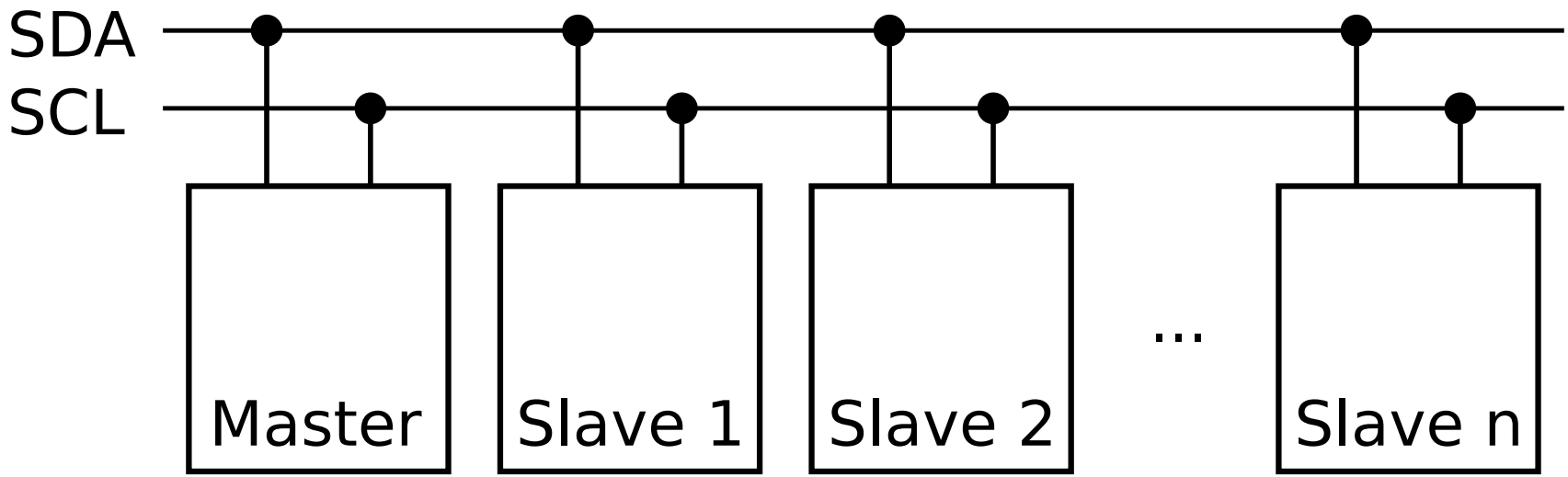
Übersicht

Grundlegendes

- I^2C = Inter-Integrated Circuit
  - sprich: "I Quadrat C" oder "I Zwei C" oder "I squared C"
  - oder auch: TWI = two wire interface (nur anderer Name)
  - oder auch: SMBus (nur marginal technisch anders)
- serieller Datenbus
  - seriell = eine Datenleitung, darauf alles nacheinander
  - Bus = mehrere Teilnehmer an gemeinsamen Leitungen

Leitungen

- zwei Leitungen
  - Serial Data Line (SDA)
  - Serial Clock Line (SCL)
- ein Master, mehrere Slaves (max. 112)



Elektrisch

- gemeinsame Referenzspannung Ground / 0V
  (auch: GND, Common, COM, V_{SS})
- Versorgungsspannung V_{CC} (auch: V_{DD}) / hier: 3.3V

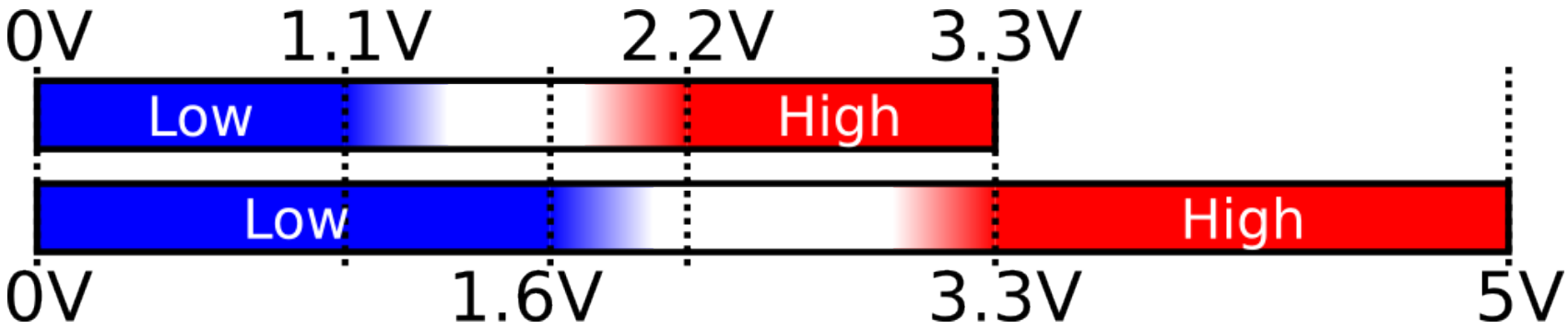
Achtung

Viele Komponenten arbeiten mit 5V Versorgungsspannung!
5V Spannung kann 3.3V Komponenten zerstören!
(insbesondere das Raspberry Pi)

- aber: I^2C unkritisch (wenn richtig angeschlossen)

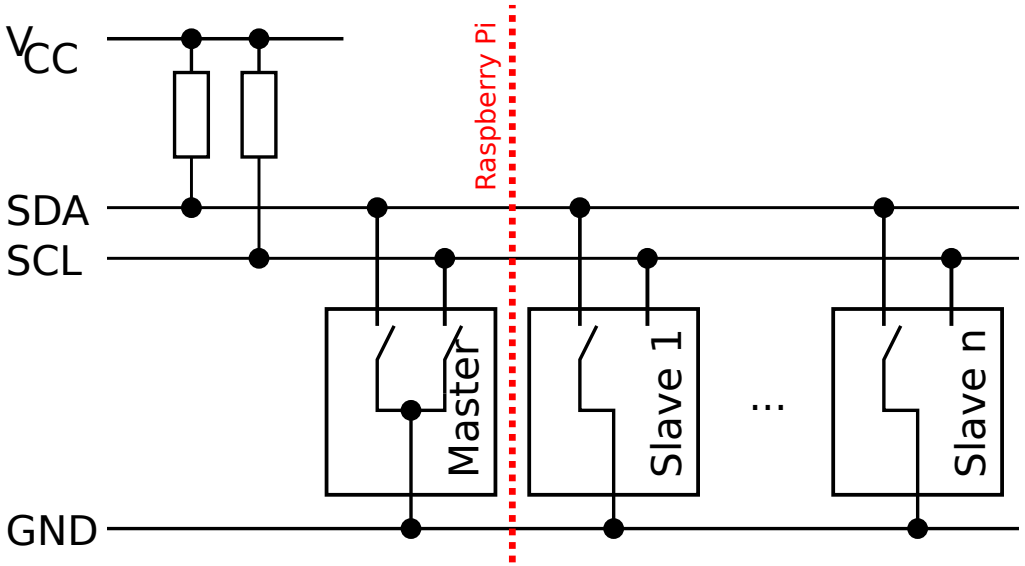
Pegel

- Spannung = Informationszustand
  - 0V = Low-Pegel = logisch 0
  - 3.3V = High-Pegel = logisch 1
- Spannung muss nicht genau passen
- Ganz grob gesagt: unteres Drittel Low, oberes Drittel High



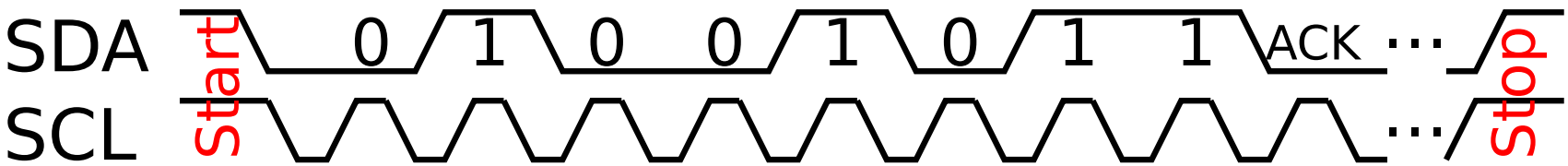
Pegelerzeugung

- I²C-Bus wird durch Pull-Up Widerstände auf 3.3V gehalten
- Pull-Up Widerstände hochohmig (mehrere $k\Omega$), im Raspberry Pi integriert
- Komponenten beeinflussen Leitung erstmal nicht (Open-Collector, Open-Drain)
- Komponenten können Leitung auf Ground ziehen



Kommunikation

- Master kommuniziert abwechselnd mit einem Slave (sowohl im Sinne, dass sich Master und Slave abwechseln, als auch, dass sich Slaves untereinander abwechseln)
- Ein Block: 8 Bit Daten (1 Byte) und danach ein Acknowledge (ACK, von Slave)
- dabei: zuerst höchstwertigstes Bit
- SDA wird gelesen, wenn SCL kurzzeitig High



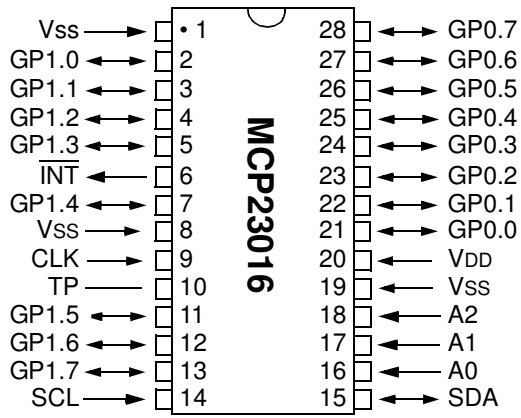
Protokoll

- Zunächst sendet Master die Adresse des Slaves (Wer?)
  - Adresse (7 Bit) kann am Slave eingestellt werden, üblicherweise erster Teil fest
  - 8tes Bit gibt an, ob der Master eine Information lesen (1) oder schreiben (0) will
- zweiter Block (vom Master) gibt an um welche Information es sich handelt (Was?)
- dritter Block (bei Lesen von Slave, bei Schreiben vom Master) Daten (Das!)

Übersicht

MCP23016 - Allgemein

- 16 Ein-/Ausgänge auf 2 Ports à 8 Pins
- jeder Pin wahlweise entweder Ein- oder Ausgang
- I²C
- 5V
- schützt Raspberry Pi
- genaue Informationen siehe Datenblatt

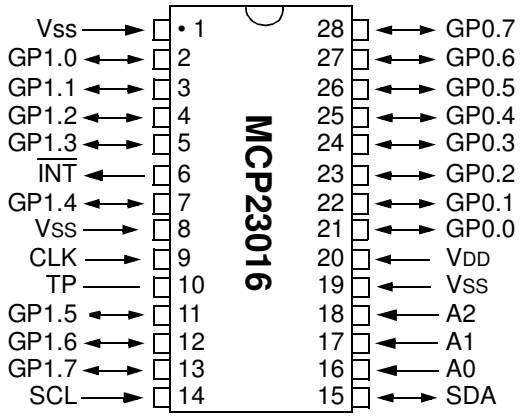


MCP23016 - I²C

- Adresse: 0100XXX (erste vier Bits fest) Anschlüsse A2,A1,A0 bestimmen letzte drei Bits
- Welche Register (s.o. Was?):

0x00	Port 0, Werte
0x01	Port 1, Werte
0x06	Port 0, Richtung
0x07	Port 1, Richtung

(Richtung: 0=Ausgang, 1=Eingang)



Beispielschaltung

- Raspberry Pi
- Portexpander MCP23016
- LED (abgeflachte Seite ist -)
- Sonstiges: Vorwiderstand für LED, Widerstand + Kondensator für MCP23016

Vorbereitung des Raspberry Pi (1)

- Vorraussetzung: installiertes Raspbian
- Installation der Software:

```
sudo apt-get update
sudo apt-get install python-smbus i2c-tools
```

- in /etc/modprobe.d/raspi-blacklist.conf die Zeile

```
blacklist i2c-bcm2708
```

ändern in

```
#blacklist i2c-bcm2708
```

Vorbereitung des Raspberry Pi (2)

- in /etc/modules folgende Zeilen ergänzen:

```
i2c-dev
i2c-bcm2708
```

- I²C für Benutzer freischalten:

```
sudo adduser pi i2c
```

- Neustarten

Python (1)

Ein kleines Python-Programm:

```
# I2C laden
import smbus
bus = smbus.SMBus(0)
# Port 1: erster Pin als Ausgang
bus.write_byte_data(0x20, 0x07, 0b11111110)
# Port 1: erster Pin auf High
bus.write_byte_data(0x20, 0x01, 0b00000001)
# Port 1: erster Pin auf Low
bus.write_byte_data(0x20, 0x01, 0b00000000)
```

Tipp: kann man direkt in Python eintippen (Befehl: python)

Python (2)

```
# Port 0: Pins abfragen
x = bus.read_byte_data(0x20, 0x00)
```