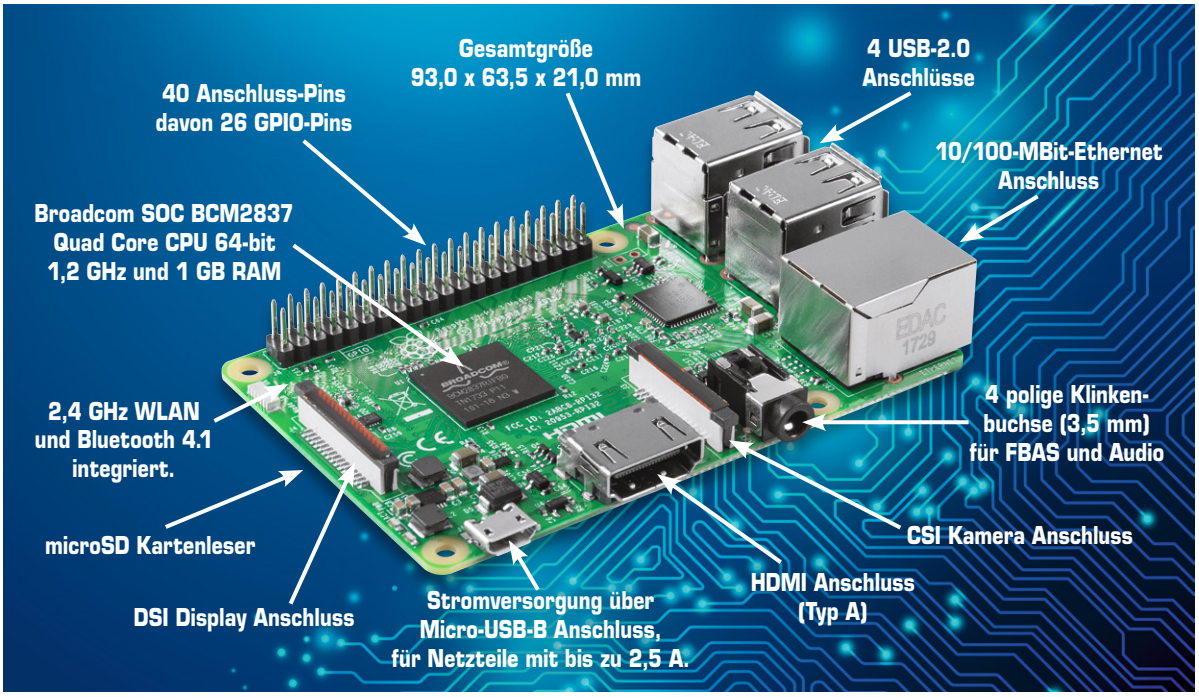
Raspberry Pi

Teil 7: Mehr Pin-Anschlüsse für den Raspberry Pi

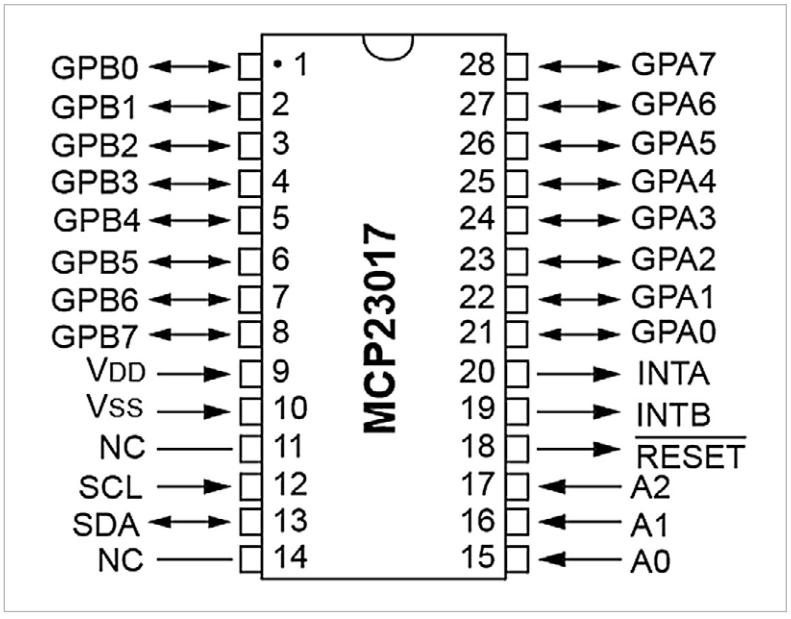
Artikelserie
im ELV Shop
#10036



Gerade in der Hausautomation, in der Sensorik und der Robotik hat man mit dem Raspberry Pi immer zu wenig Anschlüsse zur Verfügung. Besonders bei einem großen Projekt sind die wenigen echten GPIOs des Raspberry Pi schnell verbraucht. Für Abhilfe sorgt der Einsatz eines sogenannten Expanders, der bis zu 16 weitere Ein- oder Ausgänge zur Verfügung stellen kann.

GPIO-Port-Erweiterung mit MCP23017 und I²C

Der MCP23017-Baustein kommuniziert mit dem Raspberry Pi per I²C und ist für kleines Geld im Elektronikhandel erhältlich. Die beiden Bänke Pin 1–8 (Reihe B) und Pin 21–28 (Reihe A) stellen jeweils die acht zusätzlichen Ein-/Ausgänge zur Verfügung.



Der MCP23017 ist ein Baustein, der die Aus- und Eingänge des Raspberry Pi erweitert. Von Pin 1–8 steht hier die Reihe B und von Pin 21–28 die Reihe A zur Verfügung, die insgesamt 16 zusätzliche Ein- oder Ausgänge liefert.

Die Beschaltung des MCP23017 und Registerinformationen sind im Datenblatt unter http://ww1.microchip.com/downloads/en/DeviceDoc/20001952C.pdf zu finden. Auf Anhieb fällt auf, dass es zwei „Bänke" an nutzbaren Pins gibt. Der MCP23017 hat eine I²C-Schnittstelle mit einer 7-Bit-Adressierung, um dort die Geräteadresse am I²C-Bus einzustellen. Dafür stehen drei Adress-Pins (Pin 15 bis 17) am MCP23017-IC zur Verfügung und natürlich benötigt der IC auch eine Spannungsversorgung (Pin 9, 18), den Masseanschluss (Pin 10) sowie die eigentliche Schnittstelle zum I²C-Bus SCL (Pin 12) und SDA (Pin 13).

Anschluss und Adressierung des MCP23017

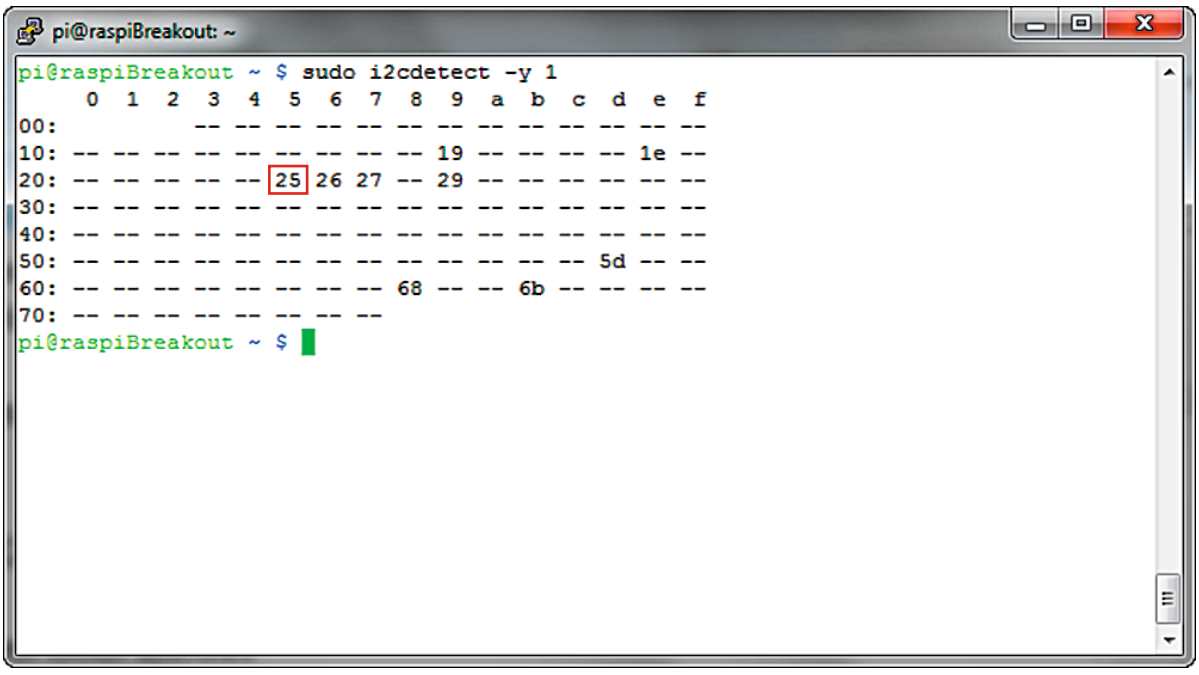
Der MCP23017-Expander wird über den I²C-Bus mit dem Raspberry Pi verbunden. Prinzipiell benötigen Sie hier neben dem GND- und 3,3-V-Anschluss noch die beiden SCL- und SDA-Pins auf der GPIO-Leiste, für die Adressierung ist die Verkabelung der drei Adress-Pins A0, A1 und A2 auf dem IC zusätzlich erforderlich, die jeweils entweder mit 3,3 V oder mit Masse bestückt werden, um die Geräteadresse festzulegen. Diese startet bei dreimal Masse mit dem Wert 0x20 als I²C-Slave-Adresse und endet bei dreimal Spannung bei Adresse 0x27.

MCP23017-Adresse	MCP23017-Pin A2 (Pin 17)	MCP23017-Pin A1 (Pin 16)	MCP23017-Pin A0 (Pin 15)	MCP23017-I²C-Adresse
000	GND	GND	GND	0x20
001	GND	GND	3V3	0x21
010	GND	3V3	GND	0x22
011	GND	3V3	3V3	0x23
100	3V3	GND	GND	0x24
100	3V3	GND	3V3	0x25
110	3V3	3V3	GND	0x26
111	3V3	3V3	3V3	0x27

7 Bit entsprechen also acht Möglichkeiten (000 bis 111) und somit auch acht unterschiedlichen verfügbaren Adressen und demnach auch maximal acht MCP23017-ICs an einem I²C-Bus, was wiederum maximal möglichen 8 x 16 = 128 Ein-/Ausgängen entspricht. Gemäß obiger Zuordnung der Adresse 0x25 werden hier Pin 15 und 17 mit 3,3 V und Pin 16 mit Masse verbunden. Pin 18 ist in allen Fällen auf 3,3 V zu legen.

I²C-Gerät MCP23017-Pin (0x25)	I²C-Gerät MCP23017	Bemerkung	Raspberry Pi (Pi Rev.1)	Raspberry Pi (Pi 1 1 Rev.2, Pi 2, Pi 3)	Raspberry-Pi-Pin	Wiring Pi
9, 15, 17, 18	VDD, A0, A2, /RESET	3,3 V	3,3 V	3,3 V	1	-
10, 16	GND, A1	Masse	GND	GND	6	-
12	SCL	I2C0_SCL	GPIO-1	GPIO-3	5	9
13	SDA	I2C0_SDA	GPIO-0	GPIO-2	3	8

Nach dem Verkabeln prüfen Sie nach dem Einschalten des Raspberry Pi die Adresse des angeschlossenen ICs. Dafür brauchen noch keine Ein-/Ausgänge des ICs bestückt zu sein. Hier geht es ausschließlich darum, zu prüfen, ob der MCP23017 ordnungsgemäß angeschlossen ist und für weitere Projekte eingesetzt werden kann.



Pin A0 und A2 auf 3V3 sowie Pin A1 auf GND: Heraus kommt in diesem Beispiel die Adresse 0x25 für den MCP-Expander. Die Adressen 0x26 und 0x27 sind in diesem Fall bereits von zwei LCD-Bildschirmen am Raspberry Pi belegt.

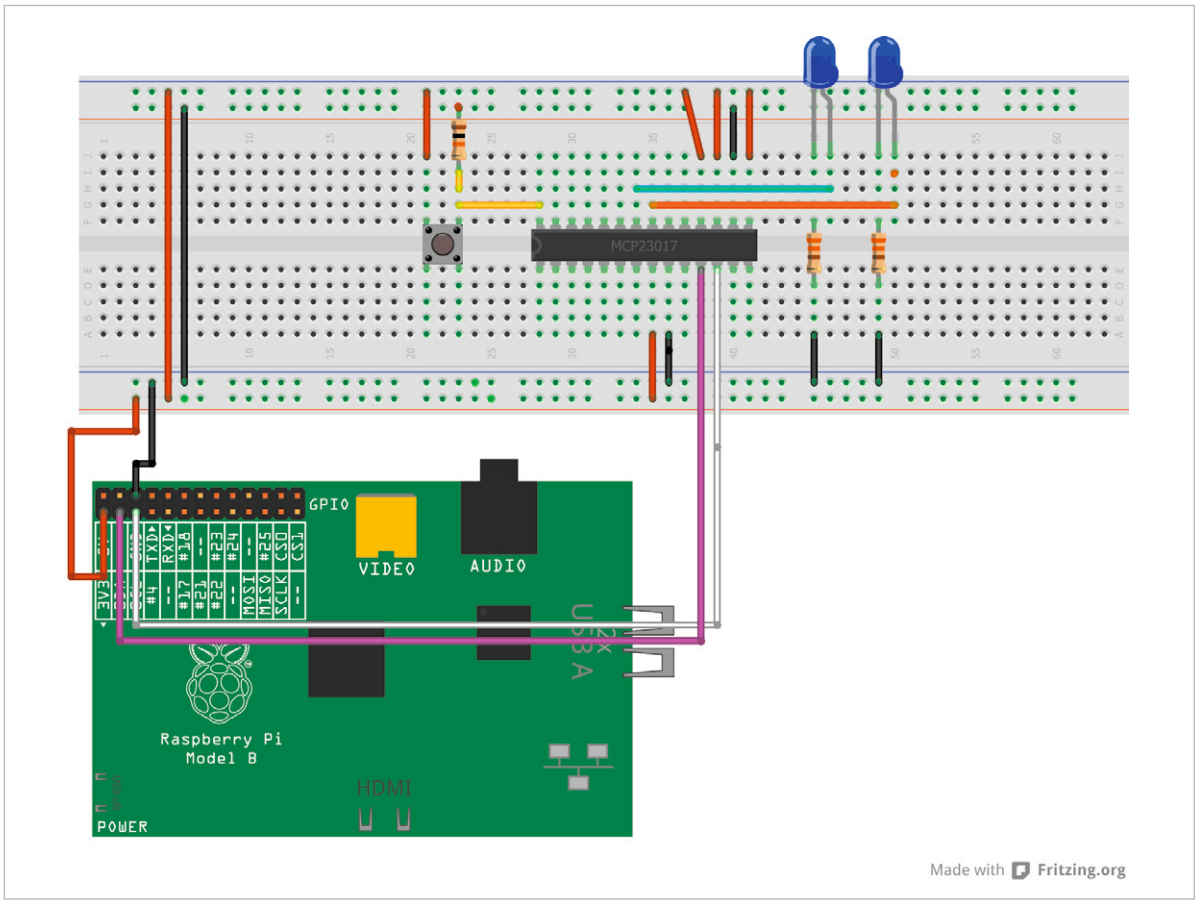
Nach dem Hochfahren des Raspberry Pi kommen somit die I²C-Tools zum Einsatz: Über das *i2cdetect*-Kommando lassen Sie sich die auf dem I²C-Bus verfügbaren Geräte anzeigen. Wird der MCP23017-Expander angezeigt, lassen sich wie gewohnt auf dem Terminal die *i2cset*- und *i2cget*-Befehle verwenden, um auf die entsprechenden Register zuzugreifen.

LED-/Schalter-Projekt mit dem MCP23017

Ist das grundsätzliche Funktionieren des MCP23017 am I²C-Bus des Raspberry Pi sichergestellt, können Sie auf dem Steckboard schon mal eine Schaltung realisieren, die sämtliche Anwendungszwecke im Zusammenhang mit dem MCP23017 abdeckt: Am Anschluss GPA7 wird ein Eingang in Form eines Schalters konfiguriert, zwei angeschlossene LEDs an den Anschluss-Pins GPA0 und GPA1 werden als Ausgang konfiguriert.

MCP23017-Pin	MCP23017-Bezeichnung	Gekoppelt mit ...	OLATA	Bit	Funktion
21	GPA0	Anode LED1	0x14	00000001	Ausgang
22	GPA1	Anode LED2	0x14	00000010	Ausgang
28	GPA7	Schalter	0x14	10000000	Eingang
9, 15, 17, 18	VDD, A0, A2, /RESET	Pi, Pin 1	-	-	-
10, 16	GND, A1	Pi, Pin 6	-	-	-
12	SCL	Pi, Pin 5	-	-	-
13	SDA	Pi, Pin 3	-	-	-

Für die Adressierung ist in diesem Beispiel die Belegung 0x25 eingestellt, die sich auf dem Steckboard einfach durch die Verbindung von 3,3 V bzw. GND an die Adress-Pins A0–A2 einstellen lässt. Auf dem Steckboard sieht das Testprojekt schematisch nun wie folgt aus:



In der Abbildung ist exemplarisch der Aufbau mit einem Raspberry Pi 1 dargestellt. Die in dem Beispiel verwendeten Anschlusspunkte sind bei den neueren Raspberry-Pi-Varianten identisch vorhanden.

Wie gewohnt benötigen LEDs einen passenden Vorwiderstand, der um die 200–330 Ohm (Ω) betragen sollte. Auch der Schalter wird mit einem Widerstand 10 kΩ versehen und zieht den Schalter auf Masse, falls dieser betätigt wird.

Ist die Schaltung auf dem Steckboard umgesetzt, prüfen Sie nach dem Einschalten des Raspberry Pi nochmals die Adresse des angeschlossenen ICs. Ist der IC ordnungsgemäß als I²C-Gerät gemeldet, nehmen Sie die beiden LEDs und den Schalter in Betrieb. Dafür sind verschiedene Registerbefehle für die Steuerung und Kontrolle notwendig.

MCP23017-Registerkontrolle und Adressierung

Die grundsätzliche Herangehensweise bei der Adressierung des MCP23017-ICs ist folgende: Zunächst legen Sie über eine Bitfolge die Konfiguration einer oder mehrerer Pins fest, ob diese jeweils als Ein- oder Ausgang fungieren sollen. Ist ein Pin beispielsweise als Ausgang definiert, können Sie im nächsten Schritt genau diesen Pin auf High setzen, d. h. den Ausgang „einschalten". Im Datenblatt des MCP23017 ist dies wie folgt dargestellt: Jedes Register ist einer Hexadezimal-Adresse (zweite Spalte in Table 1-6 Datenblatt) zugeordnet. Die Steuerung des jeweiligen Registers erfolgt über die Bit-Manipulation von Bit 0 bis Bit 7.

TABLE 1-6: CONTROL REGISTER SUMMARY (IOCON.BANK = 0)

Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IPOLA	02	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
IPOLB	03	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
GPINTENA	04	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPINTENB	05	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
DEFVALA	06	DEF7	DEF6	DEF5	DEF4	DEF3	DEF2	DEF1	DEF0	0000 0000
DEFVALB	07	DEF7	DEF6	DEF5	DEF4	DEF3	DEF2	DEF1	DEF0	0000 0000
INTCONA	08	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	0000 0000
INTCONB	09	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	0000 0000
IOCON	0A	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
IOCON	0B	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
GPPUA	0C	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPPUB	0D	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
INTFA	0E	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	0000 0000
INTFB	0F	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	0000 0000
INTCAPA	10	ICP7	ICP6	ICP5	ICP4	ICP3	ICP2	ICP1	ICP0	0000 0000
INTCAPB	11	ICP7	ICP6	ICP5	ICP4	ICP3	ICP2	ICP1	ICP0	0000 0000
GPIOA	12	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
GPIOB	13	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	14	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
OLATB	15	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

Eine Menge Informationen aus dem Datenblatt des MCP23017: Interessant für die Programmierung sind vor allem die Zeilen mit den Hex-Adressen 00 und 01 sowie 12–15.

In der nachstehenden sortierten Übersicht sind die „Zuständigkeiten" der einzelnen Funktionen den jeweiligen GPA-/GPB-Anschlüssen auf Basis des Datenblattes etwas übersichtlicher dargestellt.

Adresse(hex)	Name	Funktion	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	IODIRA	In or Out GPA	GPA7	GPA6	GPA5	GPA4	GPA3	GPA2	GPA1	GPA0
0x01	IODIRB	In or Out GPB	GPB7	GPB6	GPB5	GPB4	GPB3	GPB2	GPB1	GPB0
0x12	GPIOA	On Off GPA	GPA7	GPA6	GPA5	GPA4	GPA3	GPA2	GPA1	GPA0
0x13	GPIOB	On Off GPB	GPB7	GPB6	GPB5	GPB4	GPB3	GPB2	GPB1	GPB0
0x14	OLATA	On Off GPA	GPA7	GPA6	GPA5	GPA4	GPA3	GPA2	GPA1	GPA0
0x15	OLATB	On Off GPB	GPB7	GPB6	GPB5	GPB4	GPB3	GPB2	GPB1	GPB0

Damit können Sie sämtliche Anschlüsse von GPA0–GPA7 und GPB0–GPB7 je nach (Zeilen-)Zuordnung entweder als Ein- oder Ausgang definieren und auf High- oder Low-Pegel schalten. Falls der Anschluss als Eingang definiert wurde, kann dieser selbstverständlich über das in der Tabelle gegliederte Adressschema ausgelesen werden.

Nummer	Anschluss	Zeile	Wert (hex)	Wert (dez)	Wert (binär)
1	GPA0	0x14	0x01	1	00000001
2	GPA1	0x14	0x02	2	00000010
3	GPA2	0x14	0x04	4	00000100
4	GPA3	0x14	0x08	8	00001000
5	GPA4	0x14	0x10	16	00010000
6	GPA5	0x14	0x20	32	00100000
7	GPA6	0x14	0x40	64	01000000
8	GPA7	0x14	0x80	128	10000000
9	GPB0	0x15	0x01	1	00000001
10	GPB1	0x15	0x02	2	00000010
11	GPB2	0x15	0x04	4	00000100
12	GPB3	0x15	0x08	8	00001000
13	GPB4	0x15	0x10	16	00010000
14	GPB5	0x15	0x20	32	00100000
15	GPB6	0x15	0x40	64	01000000
16	GPB7	0x15	0x80	128	10000000

Standardmäßig sind die Pins des Raspberry Pi als Eingang definiert. Möchten Sie beispielsweise alle Pins der Anschlussreihe A hingegen als Ausgang festlegen, dann navigieren Sie in der Tabelle zu Zeile IODIRA (Input Output Direction A) und verwenden die zugeordnete Adresse 0x00, in der Sie den Wert 0x00 (Ausgang) schreiben.

Befehl	Bus-Parameter	Bus-Wert	Adresse MCP23017-Baustein	Zeile/Adresse	Wert in Hex
i2cset	-y	1	0x25	0x00	0x00

Damit setzt sich der Terminal-Befehl für Anschlussreihe A wie folgt zusammen:

```
i2cset -y 1 0x25  0x00 0x00 # IODIRA
```

Analog für die Anschlussreihe B:

```
i2cset -y 1 0x25  0x01 0x00 # IODIRB
```

Einen Mix der Anschlüsse – sprich das gleichzeitige Schalten mehrerer Ausgänge – erreichen Sie durch das Addieren der Binärzahlen: Angenommen, GPA7 soll als Eingang und die Anschlüsse GPA0–GPA6 als Ausgang fungieren. Dies entspricht laut der Tabelle dem Binärwert 10000000, also dem 0x80-Hex-Adressanteil im Befehl:

```
i2cset -y 1 0x25 0x00 0x80 # GPA7 Input / GPA0- GPA6 Output
```

Damit haben Sie die Richtung der Anschlüsse festgelegt. Im nächsten Schritt setzen Sie die Werte für den bzw. die Ausgänge, mit denen Sie Schaltvorgänge wie mit einem „normalen" GPIO-Anschluss des Raspberry Pi vornehmen können.

Schalten der zusätzlichen GPIO-Ausgänge

Für das Schalten der definierten Ausgänge sind für die Reihe A die Zeilen OLATA/GPIOA, für die Reihe B die Zeilen OLATB oder auch GPIOB zuständig. Für die Steuerung ist es prinzipiell egal, ob Sie bei der Steuerung der Anschlussreihe A im Kommando Zeile 0x12 oder 0x14 verwenden – beide haben im Endeffekt die gleiche Funktion. Dasselbe gilt hier für die Anschlussreihe B, wo 0x13 oder 0x15 analog zu verwenden sind. In diesem Beispiel wird Zeile 14 (OLATA) verwendet, um die LED an Anschluss 0 (GPA0) einzuschalten.

```
i2cset -y 1 0x25  0x14 0x01
```

Damit ist GPA0 nun auf High (3,3 V) und die LED sollte leuchten. Umgekehrt funktioniert das Ausschalten. Hier weisen Sie dem Anschluss den Wert 0 zu:

```
i2cset -y 1 0x25  0x14 0x00
```

Möchten Sie zusätzlich zu GPA0 auch GPA1 leuchten lassen, muss laut Tabelle Bit 0 und Bit 1 gesetzt sein (Binär 0011 = Dezimal 3 = Hex 0x03).

```
i2cset -y 1 0x25  0x14 0x03
```

Soll hingegen nur GPA1 und nicht mehr GPA0 leuchten, dann ist das Kommando

```
i2cset -y 1 0x25  0x14 0x02
```

das richtige (Hex 2 = Dezimal 2 = Binär 0011). Um alle angeschlossen LEDs auszuschalten, setzen Sie das Bit auf den Wert 0 zurück.

```
i2cset -y 1 0x25  0x14 0x00
```

Nun haben Sie die grundsätzlichen Funktionen des MCP23017 auf dem Terminal getestet, im nächsten Schritt setzen Sie die Schaltung in Python um, damit der MCP23017-IC auch in anderen Elektronikprojekten eingesetzt werden kann.

MCP23017 mit Python programmieren

In Sachen Zugriff auf die Register des MCP23017-ICs werden im ersten nachfolgendem Beispiel in der zu erstellenden Datei *mcp23071-step1.py* sämtliche Pins an der Anschlussreihe A des ICs als Ausgang konfiguriert. Anschließend werden über die For-Schleife die dort verfügbaren acht Anschlüsse nacheinander angesteuert und über dem Befehl *i2cbus.write_byte_data(I2C_MPC23017,OLATA,pinNr)* nacheinander eingeschaltet.

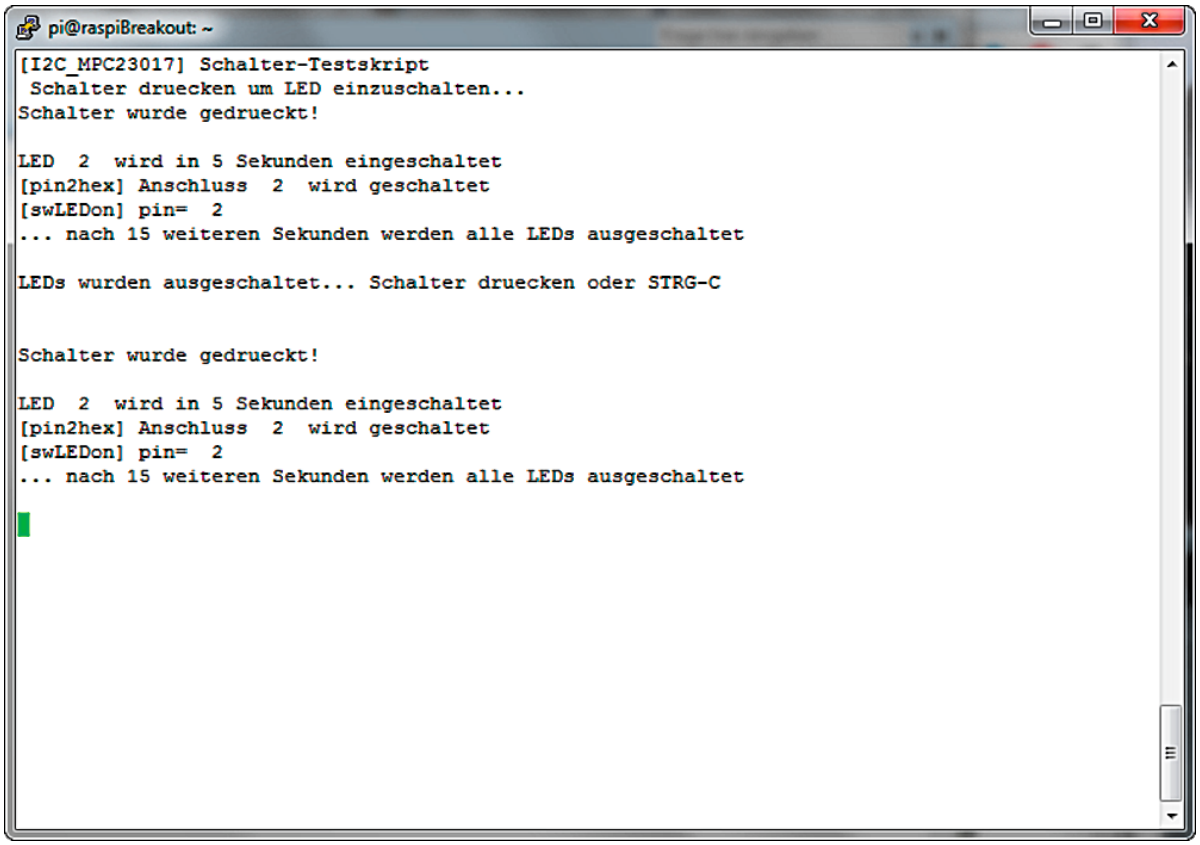
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# --------------------------------------------------------------------------
#  mcp23071-step1.py
import smbus
import time
#
#i2cbus = smbus.SMBus(0)  # Rev 1 Pi -> 0
i2cbus = smbus.SMBus(1) # Rev 2 Pi -> 1
# ----------------------------------------------------------
I2C_MPC23017 = 0x25 # Addresse ueber A0-A2 eingestellt
IODIRA = 0x00 # Pin Richtung Register A
IODIRA = 0x01 # Pin Richtung Register B
OLATA  = 0x14 # Register fuer Ausgang A
GPIOA  = 0x12 # Register fuer Eingang A
OLATB  = 0x15 # Register fuer Ausgang B
GPIOB  = 0x13 # Register fuer Eingang B
# ----------------------------------------------------------
# Alles auf Ausgang bei Anschlussreihe A
# -> IODIRA register auf 0x00
i2cbus.write_byte_data(I2C_MPC23017,IODIRA,0x00)
#
# Ausgang von allen Ausgaengen auf LOW schalten
i2cbus.write_byte_data(I2C_MPC23017,OLATA,0)
for pinNr in range(1,8):
  # Zaehlung ab 1 da Wert 0 alles aus.
  # Zaehlung ab 001 bis 111 (7)
  i2cbus.write_byte_data(I2C_MPC23017,OLATA,pinNr)
  print("Ausgang Pin-Nr.", pinNr, " an OLATA wird eingeschaltet")
  time.sleep(1)
# Alles ausschalten
i2cbus.write_byte_data(I2C_MPC23017,OLATA,0)
# ----------------------------------------------------------
```

Soll der eine oder andere Anschluss des MCP23017-ICs auch als Eingang verwendet werden, ist der Status des entsprechenden Registers permanent zu überwachen, damit auch der Schaltvorgang bemerkt wird. Anhand des obigen konkreten Schaltungsbeispiel mit den beiden LEDs und Schalter wurde nun folgende Logik implementiert: Wurde Schalter an Eingang GPA7 gedrückt, schalte beide LEDs an Ausgang GPA0 und GPA1 ein, ansonsten bleiben die angeschlossenen LEDs aus. In Python lässt sich diese Aufgabe vereinfacht wie folgt realisieren und kann dann in die bereits erstellte Datei *mcp23071-step1.py* eingebaut werden.

```
def switchLED(pin=1):
  i2cbus.write_byte_data(I2C_MPC23017,getIODIR(pin),0x80) # GPA7 Eingang
  while True:
    # Register GPIOA auslesen
    gpSwitch = i2cbus.read_byte_data(I2C_MPC23017,getREG(pin))
    if(gpSwitch & 0b10000000 == 0b10000000): # hex 0x80
      print("Schalter wurde gedrueckt!\n")
      print("LED ", pin ," wird in 5 Sekunden eingeschaltet")
      time.sleep(5)
      swLEDon(pin2hex(pin))
      print("... nach 15 weiteren Sekunden werden alle LEDs ausgeschaltet\n")
      time.sleep(15)
      swLEDoff()
```

Aus Platzgründen wurde das komplette Listing nicht abgedruckt, lediglich die „Überwachung" des Registers in Form der Funktion *switchLED*. Diese besteht aus einer While-Schleife, die den Status des Anschlusses GPA7 (Hex-Adresse 0x80) überwacht. Über das If-Konstrukt mit der Und-Verknüpfung mit dem Binärwert der Hex-Adresse wird festgestellt, ob die Taste gedrückt wurde oder nicht. Nach dem Start der While-Schleife ist der

Wert der Variable *gpSwitch* gleich 0. Nach dem Drücken des Schalters sollte *gpSwitch* den Wert 10000000 (Dezimal 128, Hex 80) annehmen. In diesem Fall wird über die Funktion *swLEDon(pin2hex(pin))* die angegebene LED eingeschaltet.



Dank der Sleep-Funktion läuft das Skript in diesem Bereich automatisch ab: Nach fünf Sekunden wird die angeschlossene LED eingeschaltet, nach weiteren 15 Sekunden wird sie wieder abgeschaltet. Anschließend wird die While-Schleife wieder aktiv und wartet darauf, dass jemand auf den Schalter drückt.

Mit dem MCP23017-IC stehen Ihnen nun deutlich mehr Anschlüsse und Möglichkeiten am Raspberry Pi zur Verfügung. Je nach Problemstellung lassen sich bis zu acht solcher ICs am Raspberry Pi betreiben – hier sind eine gut durchdachte Strategie und eine passende Platinenbestückung notwendig, damit Sie bis zu 128 zusätzliche Ein-/Ausgänge organisatorisch und im Python-Code gut handhaben.

Beachten Sie beim Ausbau mit den Port-Expandern auch, dass Sie hier je nach Anzahl der ICs eine entsprechende Stromversorgung bereitstellen. Die 3,3-Volt-Leitung des Raspberry Pi liefert im stabilen Betrieb rund 51 mA, was für sämtliche Ports und Anschlüsse gilt, die 3,3 Volt ziehen.

Reichen die Anschlüsse am I²C-Port mit den 128 Ein-/Ausgängen nicht aus, lässt sich nahezu derselbe IC auch in der SPI-Bauform (MCP23S17-IC) am Raspberry Pi einsetzen. Dort stehen zwei SPI-Kanäle (CE0 und CE1) zur Verfügung, die jeweils acht solcher MCP23S17-ICs aufnehmen können. Damit stehen nochmals weitere 256 Ein- und Ausgänge zur Verfügung – mit 384 Anschlüssen ist das Potential mit dem MCP23017 bzw. MCP23S17 dann ausgeschöpft. ELV