Mikrokontroller & I²C Bus

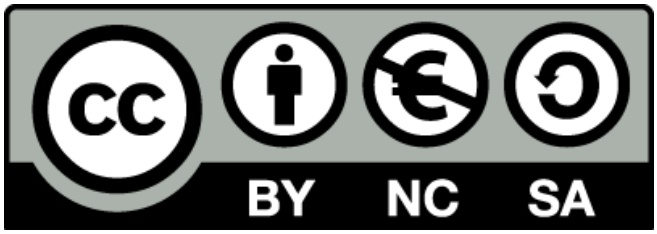
www.boxtec.ch

by AS

playground.boxtec.ch/doku.php/tutorial

Raspberry Pi Pico Porterweiterung mit dem MCP23017

Copyright



Sicherheitshinweise

Lesen Sie diese Gebrauchsanleitung, bevor Sie diesen Bausatz in Betrieb nehmen und bewahren Sie diese an einem für alle Benutzer jederzeit zugänglichen Platz auf. Bei Schäden, die durch Nichtbeachtung dieser Bedienungsanleitung verursacht werden, erlischt die Gewährleistung / Garantie. Für Folgeschäden übernehmen wir keine Haftung! Bei allen Geräten, die zu ihrem Betrieb eine elektrische Spannung benötigen, müssen die gültigen VDE-Vorschriften beachtet werden. Besonders relevant sind für diesen Bausatz die VDE-Richtlinien VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Bitte beachten Sie auch nachfolgende Sicherheitshinweise:

- Nehmen Sie diesen Bausatz nur dann in Betrieb, wenn er zuvor berührungssicher in ein Gehäuse eingebaut wurde. Erst danach darf dieser an eine Spannungsversorgung angeschlossen werden.
- Lassen Sie Geräte, die mit einer Versorgungsspannung größer als 24 V- betrieben werden, nur durch eine fachkundige Person anschließen.
- In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfewerkstätten ist das Betreiben dieser Baugruppe durch geschultes Personal verantwortlich zu überwachen.
- In einer Umgebung in der brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können, darf diese Baugruppe nicht betrieben werden.
- Im Falle eine Reparatur dieser Baugruppe, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen. Eine Reparatur des Gerätes darf nur von fachkundigen Personen durchgeführt werden.
- Spannungsführende Teile an dieser Baugruppe dürfen nur dann berührt werden (gilt auch für Werkzeuge, Messinstrumente o.ä.), wenn sichergestellt ist, dass die Baugruppe von der Versorgungsspannung getrennt wurde und elektrische Ladungen, die in den in der Baugruppe befindlichen Bauteilen gespeichert sind, vorher entladen wurden.
- Sind Messungen bei geöffnetem Gehäuse unumgänglich, muss ein Trenntrafo zur Spannungsversorgung verwendet werden
- Spannungsführende Kabel oder Leitungen, mit denen die Baugruppe verbunden ist, müssen immer auf Isolationsfehler oder Bruchstellen kontrolliert werden. Bei einem Fehler muss das Gerät unverzüglich ausser Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist.
- Es ist auf die genaue Einhaltung der genannten Kenndaten der Baugruppe und der in der Baugruppe verwendeten Bauteile zu achten. Gehen diese aus der beiliegenden Beschreibung nicht hervor, so ist eine fachkundige Person hinzuzuziehen

Bestimmungsgemäße Verwendung

- Auf keinen Fall darf 230 V~ Netzspannung angeschlossen werden. Es besteht dann Lebensgefahr!
- Dieser Bausatz ist nur zum Einsatz unter Lern- und Laborbedingungen konzipiert worden. Er ist nicht geeignet, reale Steuerungsaufgaben jeglicher Art zu übernehmen. Ein anderer Einsatz als angegeben ist nicht zulässig!
- Der Bausatz ist nur für den Gebrauch in trockenen und sauberen Räumen bestimmt.
- Wird dieser Bausatz nicht bestimmungsgemäß eingesetzt kann er beschädigt werden, was mit Gefahren, wie z.B. Kurzschluss, Brand, elektrischer Schlag etc. verbunden ist. Der Bausatz darf nicht geändert bzw. umgebaut werden!
- Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller, sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und /oder Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.
- Der Autor dieses Tutorials übernimmt keine Haftung für Schäden. Die Nutzung der Hard- und Software erfolgt auf eigenes Risiko.

Raspberry Pi Pico – Porterweiterung mit dem MCP23017

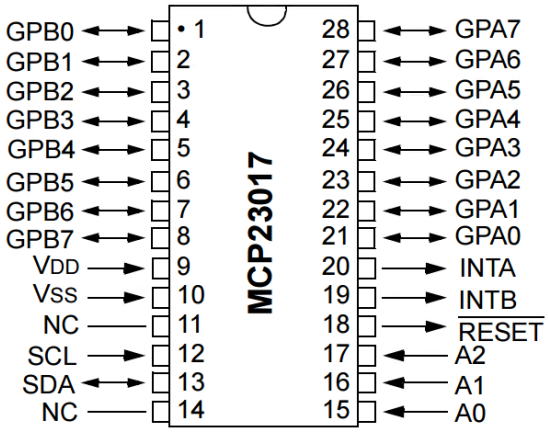
In einem anderen Tutorial habe ich bereits die Hardware zur Porterweiterung mit dem MCP 23017 ausführlich dargestellt. Die Vorstellung der Software bezog sich auf die Anwendung mit den Atmega 1284p.

In diesem Tutorial werde ich die Anwendung mit dem Raspberry Pi Pico vorstellen.

Obwohl die Register des MCP 23017 für beide ICs gleich sind, unterscheidet sich die Software durch Verwendung von Python teilweise erheblich.

Auf die allgemeine Einrichtung von Python z.B. mit Thonny werde ich in diesem Tutorial nicht eingehen.

Wenn wir uns die Hardware noch einmal genauer ansehen, so hat der MCP 23017 für uns diese relevanten Anschlüsse für die LEDs, Taster und zur Anwahl der Adresse:



Anschluss der Pins zur Adresswahl:

Pin 15 – A0
Pin 16 – A1
Pin 17 – A2

Anschluss der Taster und LEDs bei meiner Hardware:

Pin 21 – Pin 28: Port A – 8 x LED
Pin 1 – Pin 8: Port B – 8 x Taster

Adressen des MCP 23017

Die I²C-Adresse des MCP 23017 kann über die Pins A0, A1 und A2 eingestellt werden. Die Möglichen Adressen habe ich in dieser Tabelle dargestellt.

A0	A1	A2	Adresse
0	0	0	0x20
1	0	0	0x21
0	1	0	0x22
1	1	0	0x23
0	0	1	0x24
1	0	1	0x25
0	1	1	0x26
1	1	1	0x27

Die Einstellung der Adresse muss parallel dazu im Programm vorgenommen werden.

Die Bus Adresse muss auf der Hardware und im Programm gleich eingestellt werden.

Auf die Funktion der einzelnen Register werde ich in diesem Tutorial nicht eingehen.

Programm 1 – Blinken von LEDs am Port A

```
from machine import Pin, I2C
import utime
# Initialisierung I2C, Bus 0, sda-0, scl-1
SCL_Pin = 1                 # Angabe Pin SCL
SDA_Pin = 0                 # Angabe Pin SDA
Bus = 0                     # Angabe Bus Nr
Bus_Addresse = 0x22         # MCP23 017 Angabe der Bus Adresse - 0x22
i2c = I2C(Bus, scl = Pin(SCL_Pin), sda = Pin(SDA_Pin), freq = 400000)

MCP_OLATA_REG = 0x14        # Angabe der Register GPIO A
MCP_IODIRA_REG = 0x00       # steuert die Richtung der Daten am Register A

conf = [MCP_IODIRA_REG, 0]  # Richtung der Daten am PortA (0 Ausgang, 1 Eingang)

buff1 = [MCP_OLATA_REG, 0x36]       # Angabe LED/Pin nach Tabelle
buff0 = [MCP_OLATA_REG, 0x00]       # alle aus

i2c.writeto(Bus_Addresse, bytearray(conf))    # config MCP 23017 über den I2C Bus

while True:                                   # Beginn der Schleife
   i2c.writeto(Bus_Addresse, bytearray(buff1))  # schreibt an die Adresse buff 1
   utime.sleep(1)                             # Pause
   i2c.writeto(Bus_Addresse, bytearray(buff0))  # schreibt an die Adresse buff 0
   utime.sleep(1)                             # Pause
```

Angaben zu den LEDs

Beispiel 0x12 - erste Zahl (1) L5-L8 (A4-A7), zweite Zahl (2) L1-L4 (A0-A3)

erste Zahl		zweite Zahl
1 - L5 - A4	/	L1 - A0
2 - L6 - A5	/	L2 - A1
3 - L5, L6 - A4, A5	/	L1, L2 - A0, A1
4 - L7 - A6	/	L3 - A2
5 - L5, L7 - A4, A6	/	L1, L3 - A0, A2
6 - L6. L7 - A5, A6	/	L2, L3 - A1, A2
7 - L5, L6, L7 - A4, A5, A6	/	L1, L2, L3 - A0, A1, A2
8 - L8 - A7	/	L4 - A3
9 - L5, L8 - A4, A7	/	L1, L4 - A0, A3
a - L6, L8 - A5, A7	/	L2, L4 - A1, A3
b - L5, L6, L8 - A4, A5, A7	/	L1, L2, L4 - A0, A1, A3
c - L7, L8 - A6, A7	/	L3, L4 - A2, A3
d - L5, L7, L8 - A4, A6, A7	/	L1, L3, L4 - A0, A2, A3
e - L6, L7, L8 - A5, A6, A7	/	L2, L3, L4 - A1, A2, A3
f - L5, L6, L7, L8 - A4, A5, A6, A7	/	L1, L2, L3, L4 - A0, A1, A2, A3

In dieser Angabe steckt das Register A und welche LED leuchten soll:

```
buff1 = [MCP_OLATA_REG, 0x36]      ( L5, L6, L2, L3 )
```

An Hand der Tabelle dürfte die Zuordnung der LEDs kein Problem sein. Bitte die Reihenfolge der Zahlen beachten. Taster schalten nach GND, LEDs liegen über einen Widerstand an GND.

Im Netz und verschiedenen Handbücher habe ich einige Anleitung zum Schalten der LEDs gefunden. Leider war die Beschreibung oder die Zuordnung der Pins/LEDs teilweise gar nicht vorhanden oder kaum beschrieben.
Das Auslesen der Taster habe ich in keinem Beitrag gefunden. Teilweise habe ich Vorschläge bekommen, es zu machen wie beim PCF8574 oder in C. Leider sind die Unterschiede zu Python recht erheblich und die Vorschläge kaum zu verwenden.
In der Dokumentation zu Python gibt es ebenfalls einiges an Info dazu. Leider alles sehr allgemein und nicht ausführlich.

Mit diesem Programm können Taste(n) gedrückt werden. Die Tasten schalten nach GND und werden vom MCP23017 auf +5V gelegt. Die Auswahl der Taste wird mit if (data[0] ^ 0xff) & **0x01**: vorgenommen. Die gewünschte Taste wird mit **0x01** vorgenommen. Im Anschluss stelle ich eine Tabelle vor mit der Auswahl der möglichen Tastenkombinationen.

```
import machine
from machine import I2C, Pin
import utime

# Initialisierung I2C, Bus 0, sda-0, scl-1, Adresse 0x22, Frequenz 400kHz
SCL_Pin = 1                 # Angabe Pin SCL
SDA_Pin = 0                 # Angabe Pin SDA
Bus = 0                     # Angabe Bus Nr.
MCP_Address = 0x22          # MCP23017 Angabe der Bus Adresse - 0x22
i2c = I2C(Bus, scl = Pin(SCL_Pin), sda = Pin(SDA_Pin), freq = 400000)

MCP_GPIOA = 0x12            # Spiegelt den Wert am Anschluss A wider.
MCP_GPIOB = 0x13            # Spiegelt den Wert am Anschluss B wider.
MCP_OLATA = 0x14            # Schalte Ausgänge Port A.
MCP_OLATB = 0x15            # Schalte Ausgänge Port B.
MCP_IODIRA = 0x00           # Steuert die Richtung der Daten-E/A für Anschluss A.
MCP_IODIRB = 0x01           # Steuert die Richtung der Daten-E/A für Anschluss B.
MCP_IPOLA = 0x02            # Polarität der entsprechenden GPIO-Port-Bits für Port A.
MCP_IPOLB = 0x03            # Polarität der entsprechenden GPIO-Port-Bits für Port B.
MCP_GPINTENA = 0x04         # Steuert den Interrupt-on-change für jeden Pin von Anschluss A.
MCP_GPINTENB = 0x05         # Steuert den Interrupt-on-change für jeden Pin von Anschluss B.
MCP_DEFVALA = 0x06          # Steuert Interrupt-on-Change für Anschluss A.
MCP_DEFVALB = 0x07          # Steuert Interrupt-on-Change für Anschluss B.
MCP_INTCONA = 0x08          # Steuert Pin-Wert Interrupt-on-change für Anschluss A
MCP_INTCONB = 0x09          # Steuert Pin-Wert Interrupt-on-change für Anschluss B
MCP_IOCON = 0x0A            # Steuert das Gerät
MCP_GPPUA = 0x0C            # Schaltet Pull-up-Widerstände für Port A auf 5V
MCP_GPPUB = 0x0D            # Schaltet Pull-up-Widerstände für Port B auf 5V
MCP_INTFA = 0x0E            # Spiegelt Pins von Anschluss A wider
MCP_INTFB = 0x0F            # Spiegelt Pins des Anschlusses B wider
MCP_INTCAPA = 0x10  # Wert Anschluss A zum Zeitpunkt des Auftretens der Unterbrechung
MCP_INTCAPB = 0x11  # Wert Anschluss B zum Zeitpunkt des Auftretens der Unterbrechung

# Achtung Angabe der Pins in Hex
confA = [MCP_IODIRA, 0x00]      # Steuert die Richtung am Port A, 0 Ausgang, 1 Eingang
```

```
confB = [MCP_IODIRB, 0xff]      # Steuert die Richtung am Port B, 0 Ausgang, 1 Eingang
confC = [MCP_GPPUB, 0xff]       # Schaltet Pull-up-Widerstände für Port B auf 5V

buff1 = [MCP_OLATA, 0x50]                       # Angabe Zahl, Pins an nach Tabelle
buff0 = [MCP_OLATA, 0x00]                       # Angabe 0, alle Pins aus
buff2 = [MCP_OLATA, 0x05]                       # Angabe Zahl, Pins an nach Tabelle
buff3 = [MCP_OLATA, 0x03]                       # Angabe Zahl, Pins an nach Tabelle

i2c.writeto(MCP_Address, bytearray(confA))      # Port A als Ausgang
i2c.writeto(MCP_Address, bytearray(confB))      # Port B als Eingang
i2c.writeto(MCP_Address, bytearray(confC)) # Schaltet Pull-up-Widerstände für Port B auf 5V

while True:                                     # Beginn der Schleife
    data=i2c.readfrom_mem(MCP_Address, MCP_GPIOB, 1)
    print(data) # Anzeige zur Kontrolle am PC
    print()
    if (data[0] ^ 0xff) & 0x01:                 # in der 0x01 nach & liegen die Taster
        i2c.writeto(MCP_Address, bytearray(buff2))  # schreibt an die Adresse buff 1
        utime.sleep(0.5)                        # Pause
    else:
        i2c.writeto(MCP_Address, bytearray(buff3))  # schreibt an die Adresse buff 0
        utime.sleep(0.5)                        # Pause

    # Kontrolle Anzeige mit LED
    i2c.writeto(MCP_Address, bytearray(buff1))  # schreibt an die Adresse buff 1
    utime.sleep(0.5)                            # Pause
    i2c.writeto(MCP_Address, bytearray(buff0))  # schreibt an die Adresse buff 0
    utime.sleep(0.5)                            # Pause
```

Im Programm wird die Lib mcp23017.py nicht verwendet. Habe alle relevanten Register angegeben und die verwendeten im Programm eingetragen.

Angaben zu den Tastern:
Beispiel **0x12** - erste Zahl (1) T5-T8 (B4-B7), zweite Zahl (2) T1-T4 (B0-B3)

erste Zahl		zweite Zahl
1 - **T5** - B4	/	T1 - B0
2 - T6 - B5	/	**T2** - B1
3 - T5, T6 - B4, B5	/	T1, T2 - B0, B1
4 - T7 - B6	/	T3 - B2
5 - T5, T7 - B4, B6	/	T1, T3 - B0, B2
6 - T6, T7 - B5, B6	/	T2, T3 - B1, B2
7 - T5, T6, T7 - B4, B5, B6	/	T1, T2, T3 - B0, B1, B2
8 - T8 - B7	/	T4 - B3
9 - T5, T8 - B4, B7	/	T1, T4 - B0, B3
a - T6, T8 - B5, B7	/	T2, T4 - B1, B3
b - T5, T6, T8 - B4, B5, B7	/	T1, T2, T4 - B0, B1, B3
c - T7, T8 - B6, B7	/	T3, T4 - B2, B3
d - T5, T7, T8 - B4, B6, B7	/	T1, T3, T4 - B0, B2, B3
e - T6, T7, T8 - B5, B6, B7	/	T2, T3, T4 - B1, B2, B3
f - T5, T6, T7, T8 - B4, B5, B6, B7	/	T1, T2, T3, T4 - B0, B1, B2, B3

Beispiel:
Im Programm: **if (data[0] ^ 0xff) & 0x12:**
Ausgewählte Taster: **0x12**
Erste Zahl: **1** Ausgewählte Taster: **T5**
Zweite Zahl: **2** Ausgewählte Taster: **T2**

Alle Programme habe ich mit einem Raspberry Pi Pico und der angegeben Hardware getestet.

Einige Teile des Textes wurden zur besseren Übersicht farblich gestaltet.
Die Nutzung erfolgt auf eigenes Risiko.
Ich wünsche viel Spaß beim Bauen und programmieren
Achim

myroboter@web.de

http://www.elektronik-kompendium.de/sites/raspberry-pi/2611031.htm
http://www.elektronik-kompendium.de/sites/raspberry-pi/2612271.htm

.. und weiter Beiträge.

Mein besonderer Dank gilt deets Sirius3 für ihre Hilfe. Leider sind mir ihre Namen nicht bekannt.