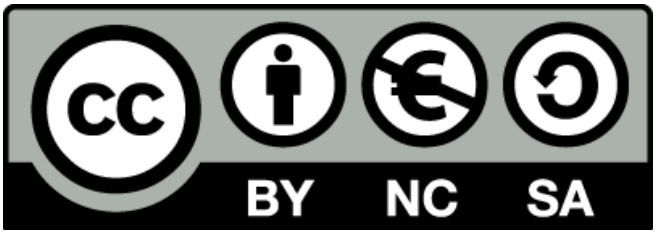
Mikrokontroller & I²C Bus

Raspberry Pi Pico
I²C Bus mit HT16K33
7-Segment Anzeige

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 – I²C Bus und dem HT16K33 7-Segment Anzeige

Mit einem Raspberry Pi Pico lassen sich die verschiedensten Hardware einfach bedienen. In diesem Tutorial möchte ich euch die Anwendung mit dem HT16K33 vorstellen. Die notwendige Hardware habe ich bereits in einem anderen Tutorial beschrieben.
Wie üblich müssen wir als erstes verschiedene Einstellungen vornehmen, in diesem Fall sagt man besser Register gesetzt werden.
Wenn die Anzeigeplatine eingeschaltet wird, ist die Anzeige nicht eingeschaltet. Es müssen drei Konfigurationswerte in den Chip schreiben werden um ihn zum Laufen zu bringen.

```
i2c_Bus.writeto(I2C_Addr_1, bytes([0x21]) )  # Einstellung Oszillator
i2c_Bus.writeto(I2C_Addr_1, bytes([0xEF]) )  # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_1, bytes([0x81]) )  # Einstellung Blinken aus, Display ein
```

Einstellung Oszillator
Ein/aus
0x21 – 0 b 0010 0001
(Oscillator ein)

Name	Command / Address / Data								Option	Description	Def.
	D15	D14	D13	D12	D11	D10	D9	D8			
System set	0	0	1	0	X	X	X	S	{S} Write only	Defines internal system oscillator on/off • {0}:Turn off System oscillator (standby mode) • {1}:Turn on System oscillator (normal operation mode)	20H

Einstellung Helligkeit
0xEF – 0 b 1110 1111
(Helligkeit auf 16/16)

D15	D14	D13	D12	D11	D10	D9	D8	ROW driver output pulse width	Def.
1	1	1	0	P3	P2	P1	P0		
1	1	1	0	0	0	0	0	1/16 duty	—
1	1	1	0	0	0	0	1	2/16 duty	—
1	1	1	0	0	0	1	0	3/16 duty	—
1	1	1	0	0	0	1	1	4/16 duty	—
1	1	1	0	0	1	0	0	5/16 duty	—
1	1	1	0	0	1	0	1	6/16 duty	—
1	1	1	0	0	1	1	0	7/16 duty	—
1	1	1	0	0	1	1	1	8/16 duty	—
1	1	1	0	1	0	0	0	9/16 duty	—
1	1	1	0	1	0	0	1	10/16 duty	—
1	1	1	0	1	0	1	0	11/16 duty	—
1	1	1	0	1	0	1	1	12/16 duty	—
1	1	1	0	1	1	0	0	13/16 duty	—
1	1	1	0	1	1	0	1	14/16 duty	—
1	1	1	0	1	1	1	0	15/16 duty	—
1	1	1	0	1	1	1	1	16/16 duty	Y

Einstellung Display ein/aus
Einstellung blinken ein/aus
0x81 – 0 b 1000 0001
(Display on, Blinking off)

Name	Command / Address / Data								Option	Description	Def.
	D15	D14	D13	D12	D11	D10	D9	D8			
Display set	1	0	0	0	X	B1	B0	D	{D} Write only	Defines Display on/off status. • {0}: Display off • {1}: Display on	80H
									{B1,B0} Write only	Defines the blinking frequency • {0,0} = Blinking OFF • {0,1} = 2HZ • {1,0} = 1HZ • {1,1} = 0.5HZ	

Die Übertragung der Daten zwischen dem Pico und den Pica Modulen erfolgt mit dem I²C Bus. Suchen wir als erstes nach den Adressen der einzelnen Pica Modulen mit der Anweisung **i2c.scan()**. Damit werden uns 4 Pica Module mit den folgenden Adressen angezeigt:

```
Scanne I2C Bus...
I2C-Geräte gefunden: 4
Dezimale Adresse: 112 | Hexadezimale Adresse: 0x70
Dezimale Adresse: 113 | Hexadezimale Adresse: 0x71
Dezimale Adresse: 114 | Hexadezimale Adresse: 0x72
Dezimale Adresse: 115 | Hexadezimale Adresse: 0x73
```

Die Anwendung von **i2c.scan()** habe ich bereits in einem anderen Tutorial erläutert.
Damit haben wir die Adressen unserer Pica Module und können diese im Programm eintragen.

```
I2C_Addr_1 = 0x70  # Angabe Adresse 1 HT16K33
I2C_Addr_2 = 0x71  # Angabe Adresse 2 HT16K33
I2C_Addr_3 = 0x72  # Angabe Adresse 3 HT16K33
I2C_Addr_4 = 0x73  # Angabe Adresse 4 HT16K33
```

Als nächste erfolgt die Einstellung des I²C Bus:

```
SCL_Pin = 21 # nach der verwendeten Hardware, sonst 1
SDA_Pin = 20 # nach der verwendeten Hardware, sonst 0
Bus = 0 # Angabe Busnummer
# Initialisierung I2C, Bus 0, sda-0(20), scl-1(21)
i2c_Bus = I2C(Bus, scl = Pin(SCL_Pin), sda = Pin(SDA_Pin), freq = 400000)
```

Dann erfolgt die Einstellung der Register für jedes Pica Modul:

```
i2c_Bus.writeto(I2C_Addr_1, bytes([0x21]))   # Einstellung Oscillator
i2c_Bus.writeto(I2C_Addr_1, bytes([0xEF]))   # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_1, bytes([0x81]))   # Einstellung Blinken aus, Display ein
```

Mit der Angabe der **bytes** kann ich verschiedene Segmente in jeder Ziffer anzeigen lassen:

```
i2c_Bus.writeto(I2C_Addr_1, bytes([0,99,0,8,0,1,0,65,0,112])) # Display 1 einschalten
```

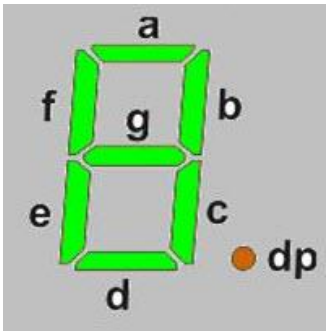
Sehen wir uns die Angabe bytes([0,99,0,8,0,1,0,65,0,112]) genauer an. Sie besteht aus 5 Gruppen mit jeweils 2 Zahlen. Jeweils 1 Paar für jede Ziffer.

1. Paar – Ziffer 1 (0, 99)
2. Paar – Ziffer 2 (0, 8)
3. Paar – Doppelpunkt ein (0, 1) - Doppelpunkt aus (0, 0)
4. Paar – Ziffer 3 (0, 65)
5. Paar – Ziffer 4 (0, 112)

Die Angaben 99, 8, 65 und 112 sind willkürliche Angaben um die Funktion zu zeigen.

Wie werden die einzelnen Segmente einer Ziffer dargestellt?

Jede einzelne Ziffer besteht aus 7 Segmenten mit der Bezeichnung von **a** bis **f** und zusätzlich dem Dezimalpunkt.



Beispiel:
Um die Ziffer **3** anzuzeigen, müssen auf der Anzeige die Segmente **a**, **b**, **c**, **d** und **g** aufleuchten.
Alle anderen Segmente sollen dunkel sein.
Daraus ergibt sich, dass die dazu notwendige Ausgabe **0100 1111** lauten muss.

Untersucht man dies für alle Ziffern, so ergibt sich folgende Tabelle:

Ziffer	Segmente	Binär	Dezimal ohne DP	Hexadezimal	Dezimal mit DP
1	b, c	0 b 0000 0110	6	0x06	134
2	a, b, d, e, g	0 b 0101 1011	91	0x5B	219
3	a, b, c, d, g	0 b 0100 1111	79	0x4F	207
4	b, c, f, g	0 b 0110 0110	102	0x66	230
5	a, c, d, f, g	0 b 0110 1101	109	0x6D	237
6	a, c, d, e, f, g	0 b 0111 1101	125	0x7D	253
7	a, b, c	0 b 0000 0111	7	0x07	135
8	a, b, c, d, e, f, g	0 b 0111 1111	127	0x7F	255
9	a, b, c, d, f, g	0 b 0110 1111	111	0x6F	239
0	a, b, c, d, e, f	0 b 0011 1111	63	0x3F	191
A	a, b, c, e, f, g	0 b 0111 0111	119	0x77	247
b	c, d, e, f, g	0 b 0111 1100	124	0x7C	252
C	a, d, e, f	0 b 0011 1001	57	0x39	185
d	b, c, d, e, g	0 b 0101 1110	94	0x5E	222
E	a, d, e, f, g	0 b 0111 1001	121	0x79	249
F	a, e, f, g	0 b 0111 0001	119	0x71	241
H	b, c, e, f, g	0 b 0111 0110	118	0x75	246
DP	h	0 b 1000 0000	128	0x80	nicht möglich

Der Dezimalpunkt wird mit Dez. 128 angesteuert. Die Zahl 3 wird mit Dez. 79 angesteuert.
Wird die Zahl 3 mit einem Dezimalpunkt angesteuert so muss 128+79=207 angegeben werden.
Egal welche Angabe man wählt es funktioniert alles.

Das komplette Programm:

```
from machine import I2C, Pin
import utime
# Vorbereitung Bus, Angabe Bus Nr und Pins
I2C_Addr_1 = 0x70  # Angabe Adresse 1 HT16K33
I2C_Addr_2 = 0x71  # Angabe Adresse 2 HT16K33
I2C_Addr_3 = 0x72  # Angabe Adresse 3 HT16K33
I2C_Addr_4 = 0x73  # Angabe Adresse 4 HT16K33
SCL_Pin = 1 # oder 21
SDA_Pin = 0 # oder 20
Bus = 0 # Angabe Busnummer
# Initialisierung I2C, Bus 0, sda-0, scl-1
i2c_Bus = I2C(Bus, scl = Pin(SCL_Pin), sda = Pin(SDA_Pin), freq = 400000)
# Adresse 1
i2c_Bus.writeto(I2C_Addr_1, bytes([0x21]))   # Einstellung Oscillator
i2c_Bus.writeto(I2C_Addr_1, bytes([0xEF]))   # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_1, bytes([0x81]))   # Einstellung Blinken aus, Display ein
```

```
# Adresse 2
i2c_Bus.writeto(I2C_Addr_2, bytes([0x21]))  # Einstellung Oscillator
i2c_Bus.writeto(I2C_Addr_2, bytes([0xEF]))  # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_2, bytes([0x81]))  # Einstellung Blinken aus, Display ein
# Adresse 3
i2c_Bus.writeto(I2C_Addr_3, bytes([0x21]))  # Einstellung Oscillator
i2c_Bus.writeto(I2C_Addr_3, bytes([0xEF]))  # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_3, bytes([0x81]))  # Einstellung Blinken aus, Display ein
# Adresse 4
i2c_Bus.writeto(I2C_Addr_4, bytes([0x21]))  # Einstellung Oscillator
i2c_Bus.writeto(I2C_Addr_4, bytes([0xEF]))  # Einstellung Helligkeit
i2c_Bus.writeto(I2C_Addr_4, bytes([0x81]))  # Einstellung Blinken aus, Display ein

while True: # Endlos Schleife Beginn
    # Muster der Segmente [ S1-0, S1-?, S2-0, S2-?, P-0, P-?, S3-0, S3-?, S4-0, S4-? ]
    # Ansteuerung Modul 1
    i2c_Bus.writeto(I2C_Addr_1, bytes([0,134,0,219,0,1,0,207,0,230])) # Display 1 einschalten
    i2c_Bus.writeto(I2C_Addr_2, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 2 alles aus
    i2c_Bus.writeto(I2C_Addr_3, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 3 alles aus
    i2c_Bus.writeto(I2C_Addr_4, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 4 alles aus
    utime.sleep(1)
    # Ansteuerung Modul 2
    i2c_Bus.writeto(I2C_Addr_2, bytes([0,237,0,253,0,01,0,135,0,255])) # Display 2 einschalten
    i2c_Bus.writeto(I2C_Addr_1, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 1 alles aus
    i2c_Bus.writeto(I2C_Addr_3, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 3 alles aus
    i2c_Bus.writeto(I2C_Addr_4, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 4 alles aus
    utime.sleep(1)
    # Ansteuerung Modul 3
    i2c_Bus.writeto(I2C_Addr_3, bytes([0,239,0,191,0,0,0,247,0,252])) # Display 3 einschalten
    i2c_Bus.writeto(I2C_Addr_1, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 1 alles aus
    i2c_Bus.writeto(I2C_Addr_2, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 2 alles aus
    i2c_Bus.writeto(I2C_Addr_4, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 4 alles aus
    utime.sleep(1)
    # Ansteuerung Modul 4
    i2c_Bus.writeto(I2C_Addr_4, bytes([0,185,0,222,0,1,0,255,0,246])) # Display 4 einschalten
    i2c_Bus.writeto(I2C_Addr_1, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 1 alles aus
    i2c_Bus.writeto(I2C_Addr_2, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 2 alles aus
    i2c_Bus.writeto(I2C_Addr_3, bytes([0,0,0,0,0,0,0,0,0,0])) # Display 3 alles aus
    utime.sleep(1)
```

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