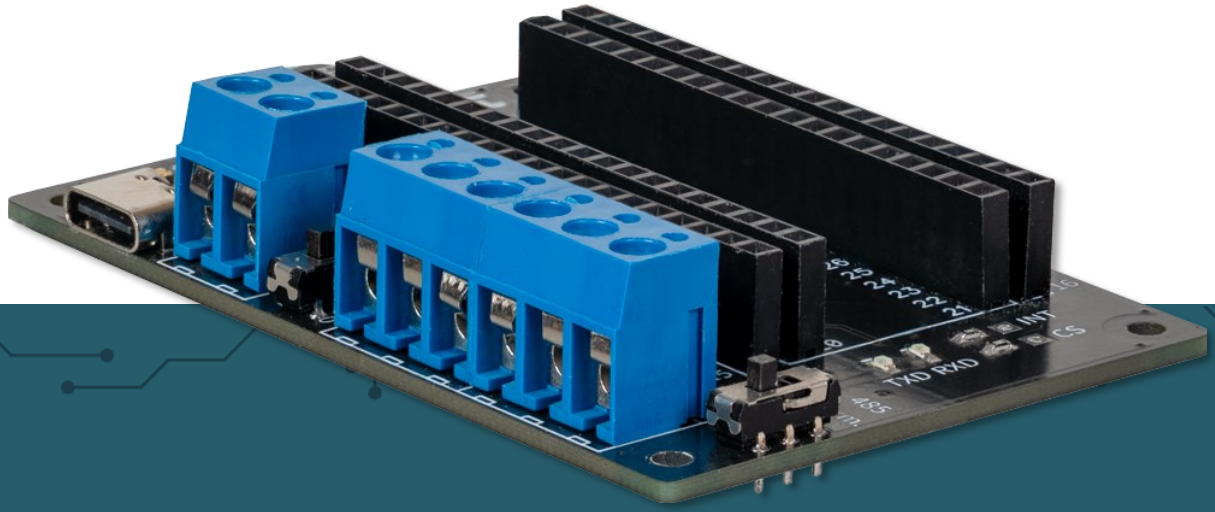


RB-P-CAN-485

Erweiterungsboard für Raspberry Pi Pico

1. ALLGEMEINE INFORMATIONEN

Sehr geehrte*r Kunde *in,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. MODULERLÄUTERUNG

In diesem Abschnitt erklären wir Ihnen kurz die einzelnen Funktionen des Erweiterungsboards, an welches Sie Ihren Pico anschließen können.

Power LED: Zeigt an wenn 5V Spannung anliegen entweder vom USB Anschluss oder vom Board internen Spannungswandler

USB (USB-C Anschluss):
5V Spannungsversorgung.

Input : 6 - 12V variable Spannungsversorgung.

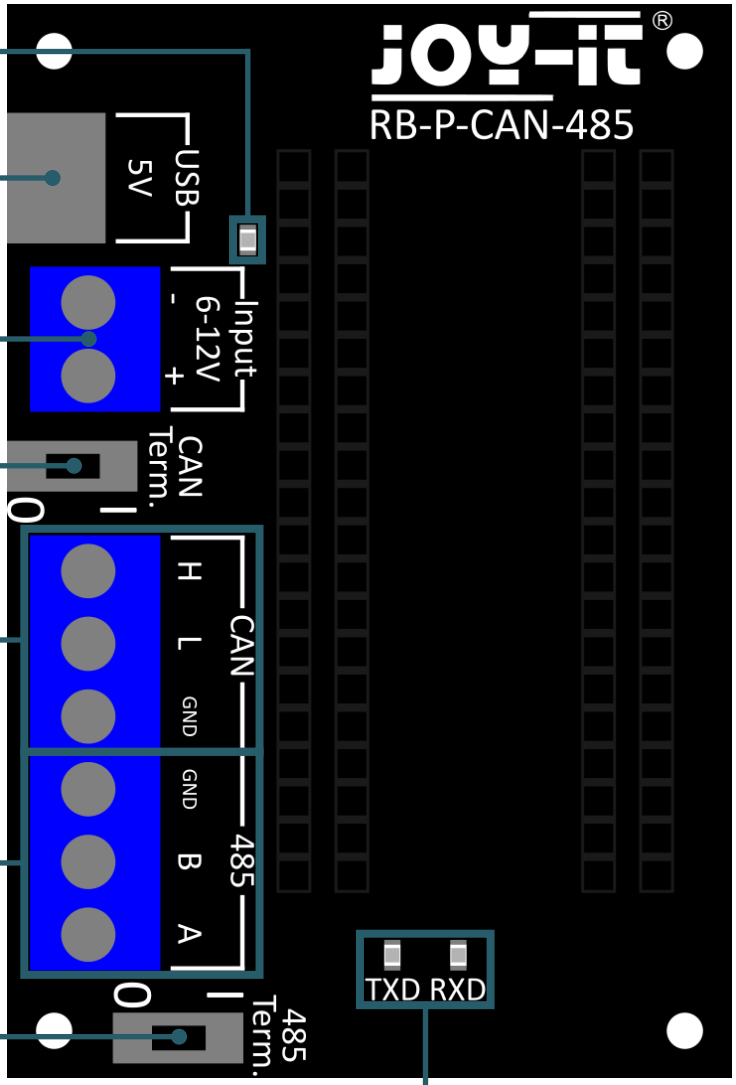
CAN Term. (CAN Termination) :
Schaltet auf der Position **"I"** einen 120Ω Widerstand dazu welcher zur Terminierung des Buses dient.

CAN :
H : Anschluss zur CAN-High Leitung.
L : Anschluss zur CAN-Low Leitung.
GND : Masseanschluss zum Potentialausgleich zwischen Nodes.

485 :
A : Anschluss zur negativen Datenleitung.
B : Anschluss zur positiven Datenleitung.
GND : Masseanschluss zum Potentialausgleich.

485 Term. (485 Termination) :
Schaltet auf der Position **"I"** einen 120Ω Widerstand dazu welcher zur Terminierung des Buses dient.

TXD | RXD (485 Status LEDs):
Zeigt den aktuellen Status auf der Transmit und Receive Leitung an.
Ein High Signal lässt die LEDs aufleuchten.



3. RASPBERRY PI PICO

Schließen Sie zur Programmierung nun ein Micro-USB Kabel an Ihren Computer und an den Raspberry Pi Pico an.

ACHTUNG! Der USB-C Anschluss auf dem Board dient ausschließlich der Spannungsversorgung. Hierbei werden keine Daten an den Raspberry Pi Pico übertragen.

Zur Übertragung der Beispielprogramme können Sie ein geeignetes Entwicklungsprogramm Ihrer Wahl verwenden. Wir empfehlen hier die **Thonny IDE**.

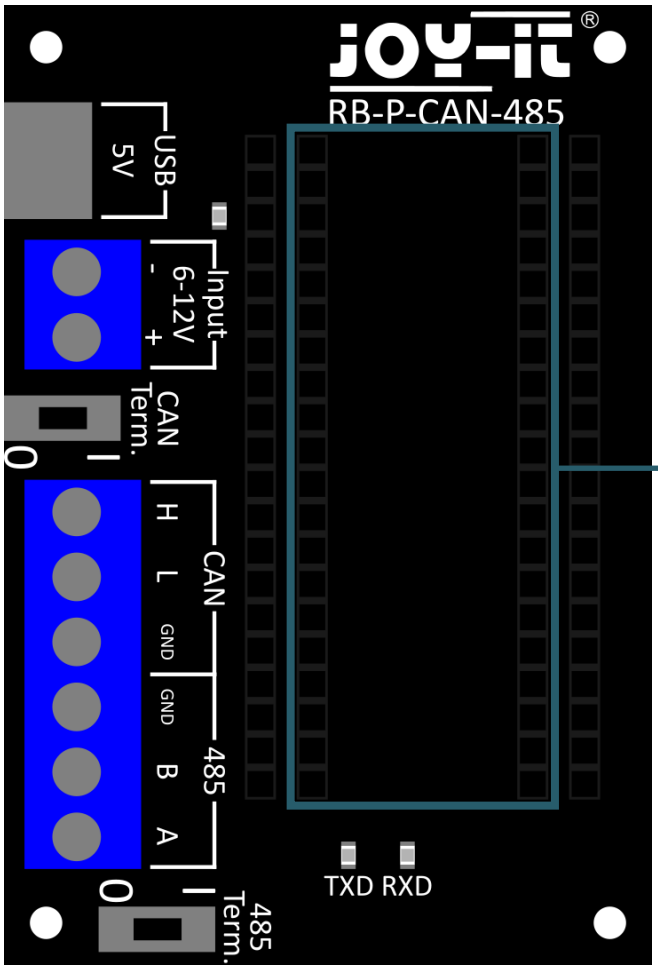
ACHTUNG! Wenn Sie neu in der Welt der Mikrocontroller und Elektronik sind, keine Sorge ! Wir haben eine spezielle Anfängeranleitung für Sie vorbereitet. Diese Anleitung ist speziell auf die Bedürfnisse von Anfängern zugeschnitten und erklärt Ihnen die Verwendung des Raspberry Pi Picos Schritt für Schritt.

Von der Grundkonfiguration bis hin zur Ausführung von Projekten - in dieser Anleitung begleiten wir Sie durch den gesamten Prozess. Unsere Anleitung umfasst leicht verständliche Erklärung und nützliche Tipps um Ihnen zu helfen, Ihre Fähigkeiten im Umfang mit dem Raspberry Pi Pico schnell und effektiv zu entwickeln. Unsere Anleitung können Sie **hier** herunterladen.

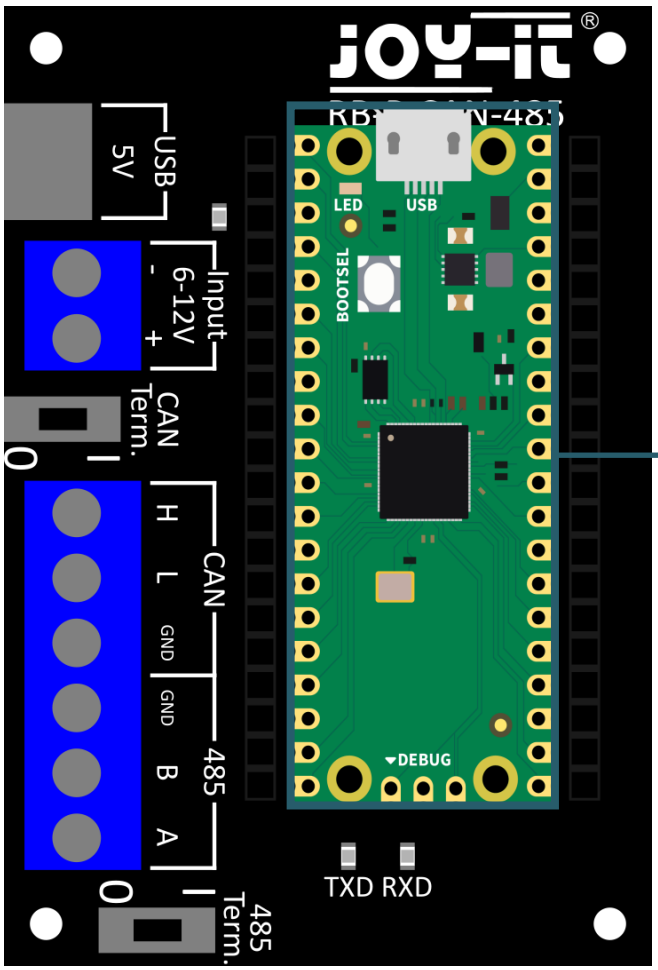
4. ANSCHLUSS DES RASPBERRY PI PICO

Im folgenden Abschnitt befassen wir uns damit, wie der Raspberry Pi Pico richtig an das Erweiterungsboard angeschlossen wird.

Im folgenden sehen Sie eine Abbildung wie der Raspberry Pi Pico auf das Erweiterungsboard gesteckt werden muss.



Steckplatz für den Raspberry Pi Pico: Hier wird der Raspberry Pi Pico aufgesteckt.



Wenn der Raspberry Pi Pico aufgesteckt wurde sollte Ihr Steckplatz nun so aussehen.

5. CODEBEISPIEL RS485

Nachdem Sie ihren Raspberry Pi Pico nun eingerichtet haben können Sie die folgenden zwei Codebeispiele für die Kommunikation über die RS485 Schnittstelle auf Ihren Raspberry Pi Pico hochladen. Alternativ können Sie die Codebeispiele auch **hier** herunterladen.

Codebeispiel des Senders:

```
from machine import UART, Pin
import time

uart0 = UART(0, baudrate=115200, tx=Pin(12), rx=Pin(13))
c=0

txData = b'RS485 send test...\r\n'
uart0.write(txData)
print('RS485 send test...')
time.sleep(0.1)

while True:
    c=c+1
    time.sleep(0.5)
    print (c)#shell output
    uart0.write("{}\r\n".format(c))
```

Codebeispiel des Empfängers:

```
from machine import UART, Pin
import time

uart0 = UART(0, baudrate=115200, tx=Pin(12), rx=Pin(13))

flag = 1
txData = 'RS485 receive test...\r\n'
uart0.write(txData)
print('RS485 receive test...')
time.sleep(0.1)

while True:
    rxData = bytes()
    while uart0.any() > 0:
        rxData = uart0.read()
        print(rxData)
```

6. CODEBEISPIEL CAN

Nachdem Sie ihren Raspberry Pi Pico nun eingerichtet haben können Sie die folgenden zwei Codebeispiele für die Kommunikation über die CAN Schnittstelle auf Ihren Raspberry Pi Pico hochladen.

Zur Benutzung dieser Beispiele wird die dazugehörige Bibliothek benötigt. Wir benutzen hier die **MicroPython_CAN_BUS_MCP2515** Bibliothek von **Longan-Labs**, welche unter der **MIT Lizenz** veröffentlicht wurde.

Codebeispiel des Senders aus der Bibliothek:

```
'''
Comments reworked by Joy-IT, 25.03.2024
----------------------------------------------
A simple example to send data to a CAN bus.
'''

# Import necessary libraries
import sys  # System-specific parameters and functions
import time  # Time access and conversions

# Import Can, CanError, CanMsg, and CanMsgFlag classes from the custom
library 'canbus'
from canbus import Can, CanError, CanMsg, CanMsgFlag

# Create an instance of the Can class to interface with the CAN bus
can = Can()

# Initialize the CAN interface
ret = can.begin()  # Begin method initializes the CAN interface and returns
a status code
if ret != CanError.ERROR_OK:  # Check if the initialization was successful
    print("Error to initialize CAN!")  # Print error message if initializa-
tion failed
```

```
    sys.exit(1)  # Exit the script with an error code if CAN interface
couldn't be initialized
print("Initialized successfully!")  # Print success message if initializati-
on was successful

# Main loop to send data to the CAN bus
while True:
    data = b"\x12\x34\x56\x78\x9A\xBC\xDE\xF0"  # Data to be sent over the
CAN bus in bytes format

    # Create a standard format frame CAN message
    msg = CanMsg(can_id=0x123, data=data)  # can_id is the identifier for
the CAN message, data is the bytes to send
    error = can.send(msg)  # Send the CAN message and store the error status
    if error == CanError.ERROR_OK:  # Check if the message was sent success-
fully
        print('1------------------------------')  # Print a delimiter if the
message was sent successfully

    # Create an extended format frame CAN message
    msg = CanMsg(can_id=0x12345678, data=data, flags=CanMsgFlag.EFF)  # EFF
flag indicates an extended frame format
    error = can.send(msg)  # Send the CAN message and store the error status
    if error == CanError.ERROR_OK:  # Check if the message was sent success-
fully
        print('2------------------------------')  # Print a delimiter if the
message was sent successfully

    time.sleep(1)  # Wait for 1 second before sending the next set of messa-
ges
```

Codebeispiel des Empfängers aus der Bibliothek:

```
'''
Comments reworked by Joy-IT, 25.03.2024
------------------------------------------
A simple example to receive data from a CAN bus.
'''

# Import necessary libraries
import sys  # System-specific parameters and functions
import time  # Time access and conversions

# Import the Can and CanError classes from the custom library/module
'canbus'
from canbus import Can, CanError

# Create a Can object instance for interfacing with the CAN bus
can = Can()

# Initialize the CAN interface
ret = can.begin()  # Begin method initializes the CAN interface and returns
a status code
if ret != CanError.ERROR_OK:  # Check if the initialization was successful
    print("Error to initialize CAN!")  # Print error message if initializa-
tion failed
```

```
    sys.exit(1)  # Exit the script with an error code if CAN interface
couldn't be initialized
print("Initialized successfully!")  # Print success message if initializati-
on was successful

# Main loop to receive data from the CAN bus
while True:
    if can.checkReceive():  # Check if there is data to receive on the CAN
bus
        error, msg = can.recv()  # Receive data from the CAN bus; returns an
error code and the message object
        if error == CanError.ERROR_OK:  # Check if data was received without
errors
            # Print received message details
            print('------------------------------')
            print("can id: %#x" % msg.can_id)  # Print the CAN ID in hexade-
cimal format
            print("is rtr frame:", msg.is_remote_frame)  # Print whether the
message is a Remote Transmission Request (RTR) frame
            print("is eff frame:", msg.is_extended_id)  # Print whether the
message uses an Extended Frame Format (EFF)
            print("can data hex:", msg.data.hex())  # Print the message data
in hexadecimal format
            print("can data dlc:", msg.dlc)  # Print the Data Length Code
(DLC), indicating the number of data bytes in the message
    else:
        time.sleep(0.1)  # If no data to receive, wait for 0.1 seconds befo-
re checking again
```

7. SONSTIGE INFORMATIONEN

Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

Symbol auf Elektro- und Elektronikgeräten:

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

Rückgabemöglichkeiten:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte, bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Möglichkeit Rückgabe in Ihrer Nähe:
Wir senden Ihnen eine Paketmarke zu, mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

Informationen zur Verpackung:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

8. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen, stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: **service@joy-it.net**
Ticket-System: **http://support.joy-it.net**
Telefon: +49 (0)2845 9360-50 (Mo - Do: 09:00 - 17:00 Uhr, Fr: 09:00 - 14:30 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
www.joy-it.net

Veröffentlicht: 27.03.2024