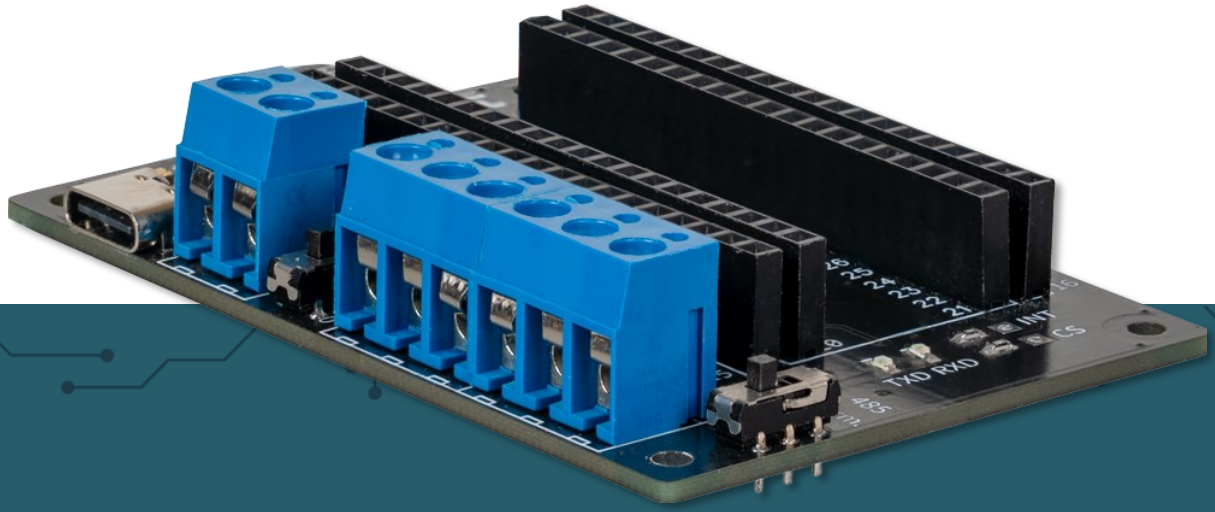


RB-P-CAN-485

Expansion board for Raspberry Pi Pico

Joy-IT powered by SIMAC Electronics GmbH - Pascalstr. 8 - 47506 Neukirchen-Vluyn - www.joy-it.net

1. GENERAL INFORMATION

Dear customer,
thank you very much for choosing our product.
In the following, we will introduce you to what to observe while starting up and using this product.

Should you encounter any unexpected problems during use, please do not hesitate to contact us.

2. MODULE EXPLANATION

In this section, we will briefly explain the individual functions of the expansion board to which you can connect your Pico.

Power LED: Indicates when 5V voltage is present either from the USB port or from the board's internal voltage converter

USB (USB-C connection):
5V Power supply.

Input : 6 - 12V variable power supply.

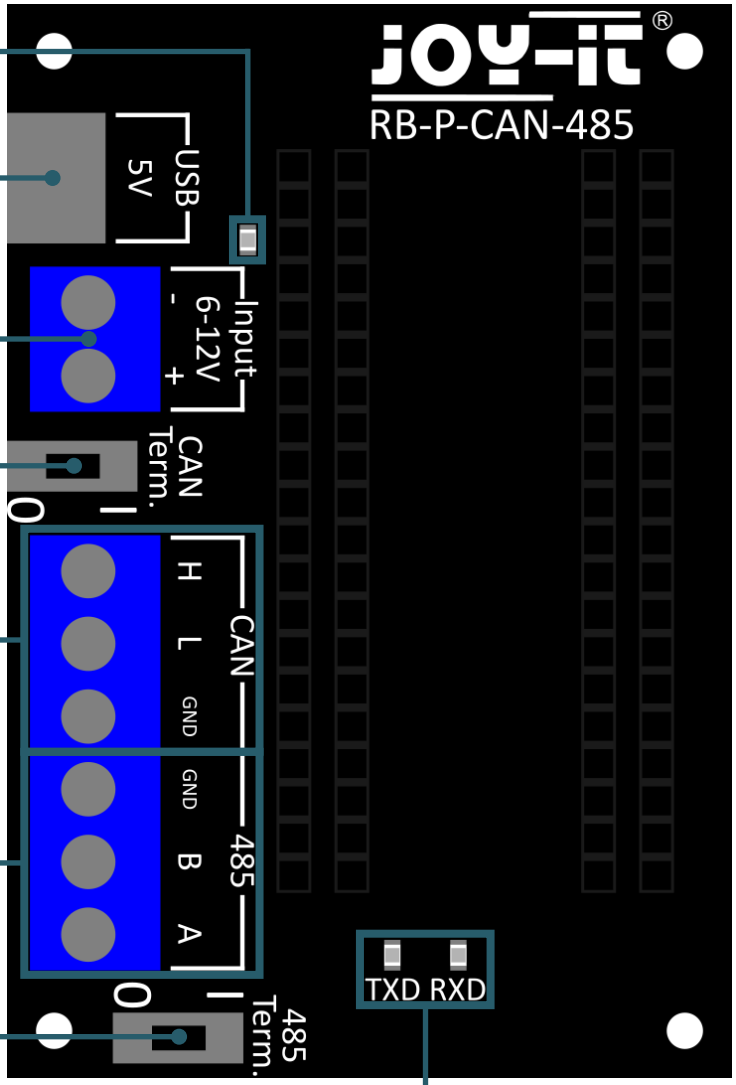
CAN Term. (CAN Termination) :
Switches a 120Ω resistor to the "I" position, which is used for specialized termination of the bus.

CAN :
H : Connection to the CAN-High line.
L : Connection to the CAN-Low line.
GND : Ground connection for potential equalization between nodes.

485 :
A : Connection to the negative data line.
B : Connection to the positive data line.
GND : Ground connection for potential equalization.

485 Term. (485 Termination) :
Switches a 120Ω resistor to the "I" position, which is used for specialized termination of the bus.

TXD | RXD (485 status LEDs):
Displays the current status on the Transmit and Receive line.
A high signal causes the LEDs to light up.



3. RASPBERRY PI PICO

Now connect a micro USB cable to your computer and to the Raspberry Pi Pico for programming.

ATTENTION: The USB-C connection on the board is only used for the power supply. No data is transferred to the Raspberry Pi Pico.

You can use a suitable development program of your choice to transfer the sample programs. We recommend the **Thonny IDE**.

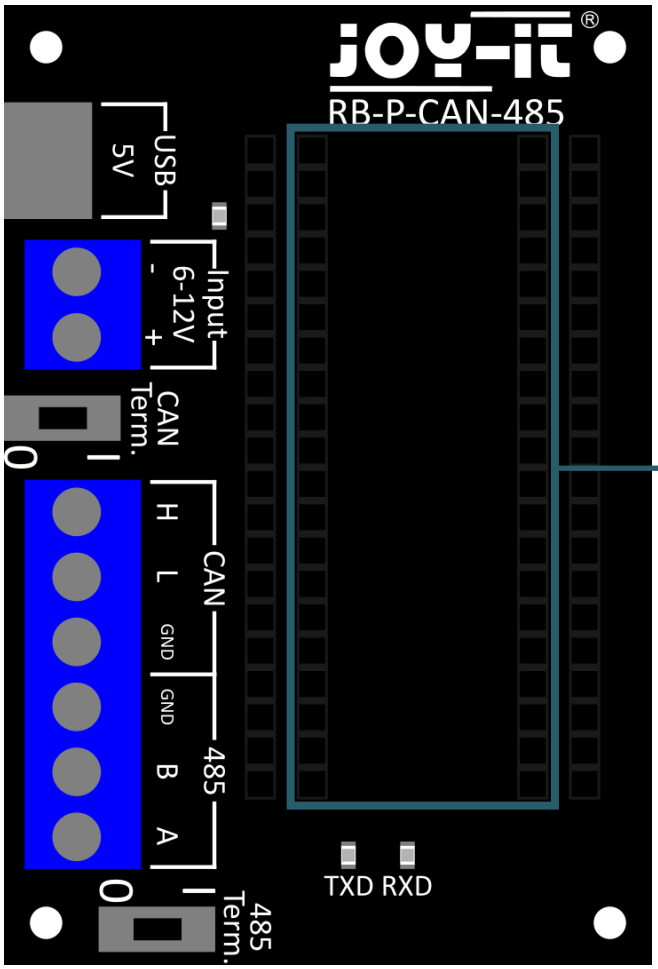
ATTENTION: If you are new to the world of microcontrollers and electronics, don't worry! We have prepared a special beginner's guide for you. This guide is specially tailored to the needs of beginners and explains how to use the Raspberry Pi Pico step by step.

From the basic configuration to the execution of projects - in this guide we walk you through the whole process. Our guide includes easy-to-understand explanations and useful tips to help you develop your skills at scale with the Raspberry Pi Pico quickly and effectively. You can download our guide **here**.

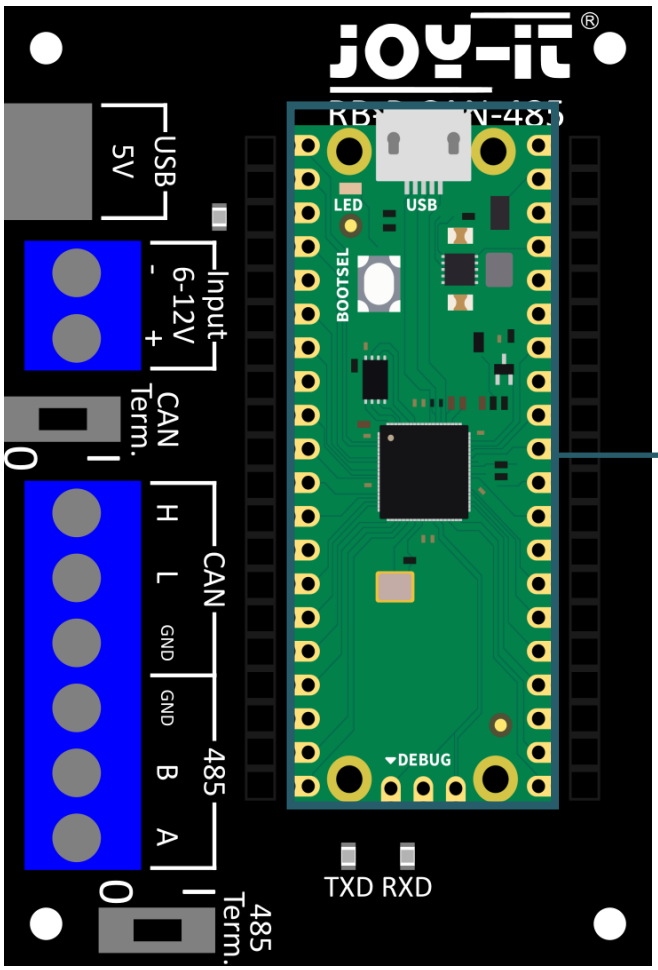
4. CONNECTION OF THE RASPBERRY PI PICO

In the following section, we will look at how to connect the Raspberry Pi Pico to the expansion board correctly.

Below you can see an illustration of how the Raspberry Pi Pico must be plugged into the expansion board.



Slot for the Raspberry Pi Pico:
The Raspberry Pi Pico is plugged in here.



If the Raspberry Pi Pico has been plugged in, your slot should now look like this.

5. CODE EXAMPLE RS485

Now that you have set up your Raspberry Pi Pico, you can upload the following two code examples for communication via the RS485 interface to your Raspberry Pi Pico. Alternatively, you can also download the code examples **here**.

Code example of the transmitter:

```
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))
```

Code example of the receiver:

```
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. CODE EXAMPLE CAN

Now that you have set up your Raspberry Pi Pico, you can upload the following two code examples for communication via the CAN interface to your Raspberry Pi Pico.

The corresponding library is required to use these examples. We use the **MicroPython_CAN_BUS_MCP2515** library from **Longan-Labs**, which is published under the **MIT license**.

Code example of the transmitter from the library:

```
'''
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
```

Code example of the receiver from the library:

```
'''
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. ADDITIONAL INFORMATION

Our information and take-back obligations according to the Electrical and Electronic Equipment Act (ElektroG)

Symbol on electrical and electronic equipment:

This crossed-out dustbin means that electrical and electronic appliances do not belong in the household waste. You must return the old appliances to a collection point.

Before handing over waste batteries and accumulators that are not enclosed by waste equipment must be separated from it.

Return options:

As an end user, you can return your old device (which essentially fulfills the same function as the new device purchased from us) free of charge for disposal when you purchase a new device.

Small appliances with no external dimensions greater than 25 cm can be disposed of in normal household quantities independently of the purchase of a new appliance.

Possibility of return at our company location during opening hours:

SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn, Germany

Possibility of return in your area:

We will send you a parcel stamp with which you can return the device to us free of charge. Please contact us by email at Service@joy-it.net or by telephone.

Information on packaging:

If you do not have suitable packaging material or do not wish to use your own, please contact us and we will send you suitable packaging.

8. SUPPORT

If there are still any issues pending or problems arising after your purchase, we will support you by e-mail, telephone and with our ticket support system.

Email: **service@joy-it.net**
Ticket system: **http://support.joy-it.net**
Telephone: +49 (0)2845 9360-50 (Mon - Thur: 09:00 - 17:00 o‘clock, Fri: 09:00 - 14:30 o‘clock)

For further information please visit our website:
www.joy-it.net

Veröffentlicht: 27.03.2024