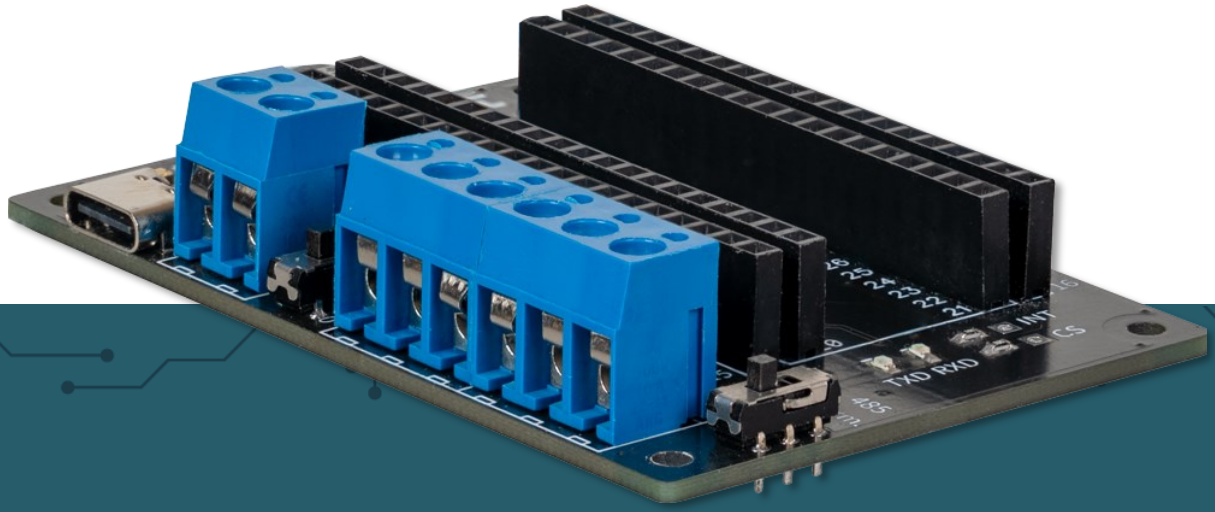


RB-P-CAN-485

Rozšiřující deska pro Raspberry Pi Pico

Pozor! Tento návod byl automaticky přeložen, v případě pochybností se prosím podívejte do anglického návodu nebo kontaktujte náš zákaznický servis.
Všechny požadavky na podporu musí být v němčině nebo angličtině.

1. OBECNÉ INFORMACE

Vážený zákazníku,
děkujeme, že jste si vybrali náš výrobek.
V následujícím textu vás seznámíme s tím, čeho je třeba dbát při uvádění do provozu a používání tohoto výrobku.´

Pokud se během používání vyskytnou neočekávané problémy, neváhejte se na nás obrátit.

2. CVYSVĚTLENÍ MODULU

V této části stručně vysvětlíme jednotlivé funkce rozšiřující desky, ke které můžete Pico připojit.

LED dioda napájení: Indikuje přítomnost napětí 5 V buď z portu USB, nebo z interního měniče napětí na desce.

USB (připojení USB-C):
Napájení: 5V.

Vstup : 6 - 12V variabilní napájení.

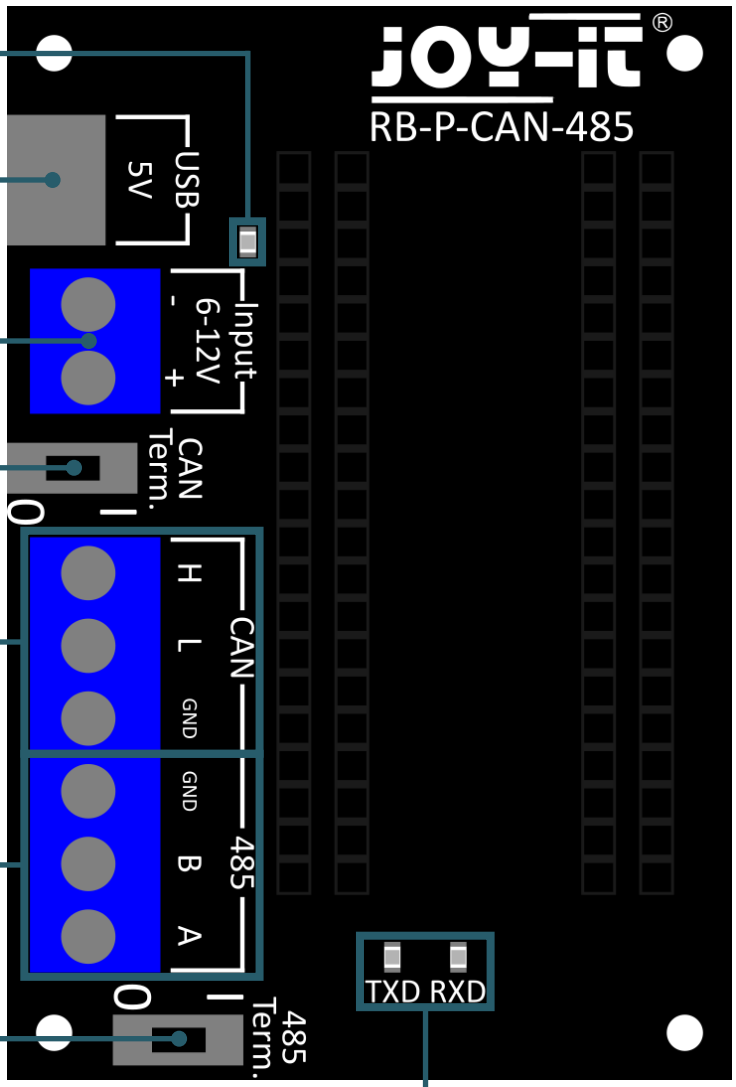
Termín CAN. (Ukončení CAN) :
Přepne rezistor 120Ω do polohy "I", který se používá pro specializované ukončení sběrnice.

CAN:
H : Připojení k vedení CAN-High.
L : Připojení k vedení CAN-Low.
GND : Zemnící připojení pro vyrovnání potenciálu mezi uzly.

485 :
A : Připojení k záporné datové lince.
B : Připojení ke kladné datové lince.
GND : Zemní připojení pro vyrovnání potenciálu.

485 Termín. (485 Ukončení) :
Přepíná rezistor 120Ω do polohy "I", který se používá pro specializované ukončení sběrnice.

TXD | RXD (stavové LED 485):
Zobrazuje aktuální stav na vysílací a přijímací lince.
Vysoký signál způsobí rozsvícení LED diod.



3. RASPBERRY PI PICO

Nyní připojte kabel micro USB k počítači a k Raspberry Pi Pico pro programování.

POZOR: Připojení USB-C na desce slouží pouze k napájení. Do počítače Raspberry Pi Pico se nepřenášejí žádná data.

K přenosu vzorových programů můžete použít vhodný vývojový program podle vlastního výběru. Doporučujeme prostředí **Thonny IDE**.

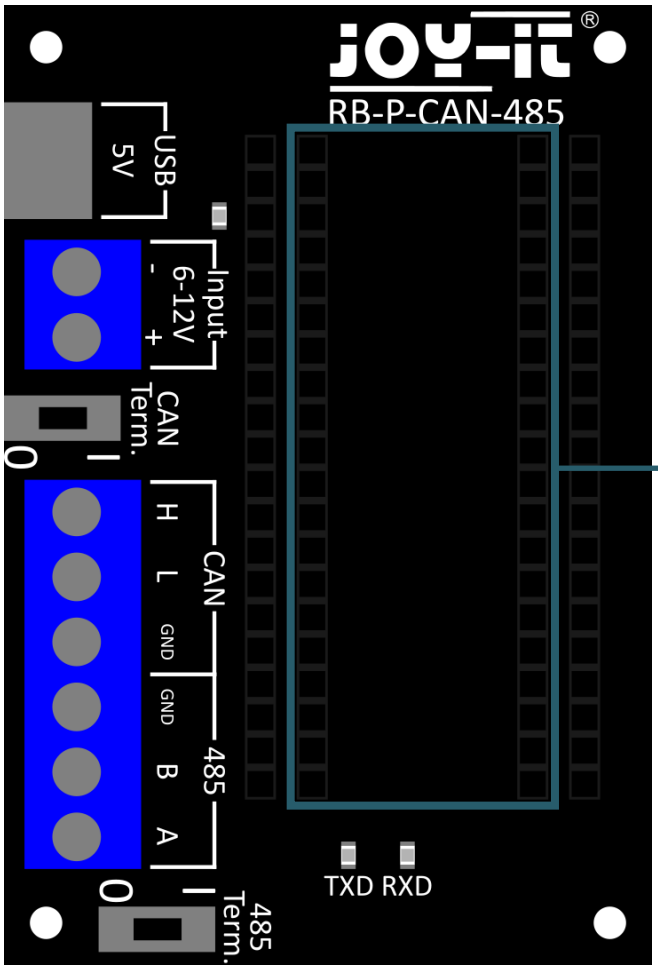
POZOR: Pokud jste ve světě mikrokontrolérů a elektroniky nováčky, nezoufejte! Připravili jsme pro vás speciální příručku pro začátečníky. Tato příručka je speciálně přizpůsobena potřebám začátečníků a vysvětluje, jak používat Raspberry Pi Pico krok za krokem.

Od základního nastavení až po realizaci projektů - v této příručce vás provedeme celým procesem. Náš průvodce obsahuje srozumitelná vysvětlení a užitečné tipy, které vám pomohou rychle a efektivně rozvinout vaše dovednosti v rozsahu práce s Raspberry Pi Pico. Naši příručku si můžete stáhnout **zde**.

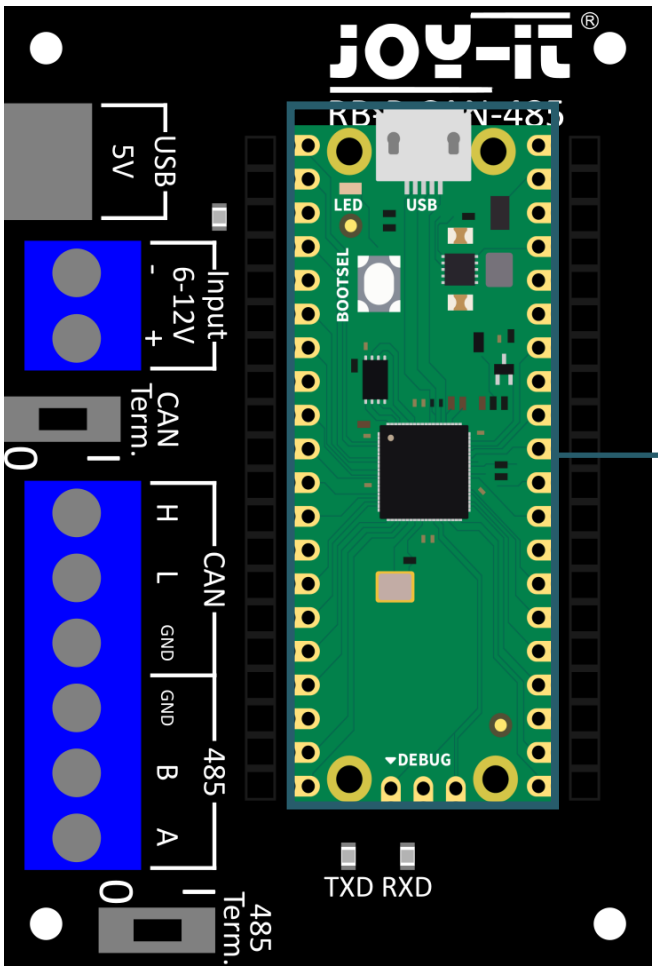
4. PŘIPOJENÍ RASPBERRY PI PICO

V následující části se podíváme na to, jak správně připojit Raspberry Pi Pico k rozšiřující desce.

Níže vidíte ilustraci, jak je třeba Raspberry Pi Pico připojit k rozšiřující desce.



Slot pro Raspberry Pi Pico:
Zde se připojuje Raspberry Pi Pico.



Pokud je Raspberry Pi Pico připojeno, měl by váš slot nyní vypadat takto.

5. PŘÍKLAD KÓDU RS485

Nyní, když jste nastavili Raspberry Pi Pico, můžete do něj nahrát následující dva příklady kódu pro komunikaci přes rozhraní RS485. Případně si příklady kódu můžete stáhnout také **zde**.

Příklad kódu vysílače:

```
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))
```

Příklad kódu přijímače:

```
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. PŘÍKLAD KÓDU MŮŽE

Nyní, když jste nastavili Raspberry Pi Pico, můžete nahrát následující dva příklady kódu pro komunikaci přes rozhraní CAN do Raspberry Pi Pico.

Pro použití těchto příkladů je nutná odpovídající knihovna. Používáme knihovnu **MicroPython_CAN_BUS_MCP2515** od společnosti **Longan-Labs**, která je zveřejněna pod **licencí MIT**.

Příklad kódu vysílače z knihovny:

```
'''
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
```

Příklad kódu přijímače z knihovny:

```
'''
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. DALŠÍ INFORMACE

Naše informační povinnosti a povinnosti zpětného odběru podle zákona o elektrických a elektronických zařízeních (ElektroG)

Symbol na elektrických a elektronických zařízeních: 

Tento přeškrtnutý odpadkový koš znamená, že elektrická a elektronická zařízení nepatří do domovního odpadu. Staré spotřebiče musíte odevzdat na sběrném místě.

Před odevzdáním odpadních baterií a akumulátorů, které nejsou přiloženy k odpadnímu zařízení, je třeba je od něj oddělit.

Možnosti zpětného odběru:

Jako koncový uživatel můžete při nákupu nového zařízení bezplatně odevzdat staré zařízení (které v podstatě plní stejnou funkci jako nové zařízení zakoupené u nás) k likvidaci.

Malá zařízení, jejichž vnější rozměry nepřesahují 25 cm, lze likvidovat v běžném množství v domácnosti nezávisle na zakoupení nového zařízení.

Možnost odevzdání v provozovně naší společnosti v otevírací době:

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

Možnost vrácení ve vaší oblasti:

V případě potřeby vám zašleme známku na balík, s níž nám můžete zařízení bezplatně vrátit. Kontaktujte nás prosím e-mailem na adrese Service@joy-it.net nebo telefonicky.

Informace o balení:

Pokud nemáte vhodný obalový materiál nebo nechcete použít svůj vlastní, kontaktujte nás a my vám zašleme vhodný obal.

8. PODPORA

Pokud se po nákupu ještě vyskytnou nějaké nevyřešené otázky nebo problémy, poskytneme vám podporu prostřednictvím e-mailu, telefonu a našeho systému ticketové podpory.

Email: **service@joy-it.net**
Systém vstupenek: **http://support.joy-it.net**
Telefon: +49 (0)2845 9360-50 (Po - Čt: 09:00 - 17:00 hodin,
Pá: 09:00 - 14:30 hodin)

Další informace naleznete na našich webových stránkách:
www.joy-it.net

Zveřejněno na: 27.03.2024