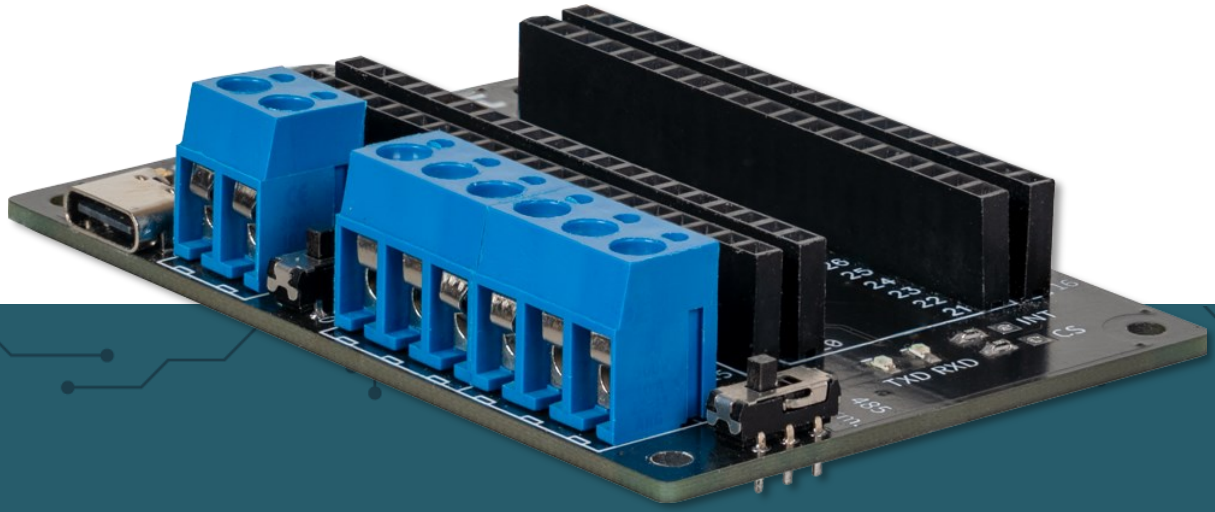


RB-P-CAN-485

Carte d'extension pour Raspberry Pi Pico

Attention! Ce manuel a été traduit automatiquement. En cas de doute, veuillez vous référer au manuel anglais ou contacter notre service clientèle.
Toutes les demandes d'assistance doivent être rédigées en allemand ou en anglais.

1. INFORMATIONS GÉNÉRALES

Cher client,
nous vous remercions d'avoir choisi notre produit.
Dans ce qui suit, nous vous indiquons ce qu'il faut observer lors de la mise en service et de l'utilisation de ce produit.

Si vous rencontrez des problèmes inattendus pendant l'utilisation, n'hésitez pas à nous contacter.

2. EXPLICATION DU MODULE

Dans cette section, nous expliquerons brièvement les différentes fonctions de la carte d'extension à laquelle vous pouvez connecter votre Pico.

LED d'alimentation : Indique la présence d'une tension de 5V provenant du port USB ou du convertisseur de tension interne de la carte.

USB (connexion USB-C) :
5V Alimentation électrique.

Entrée : alimentation variable 6 - 12V.

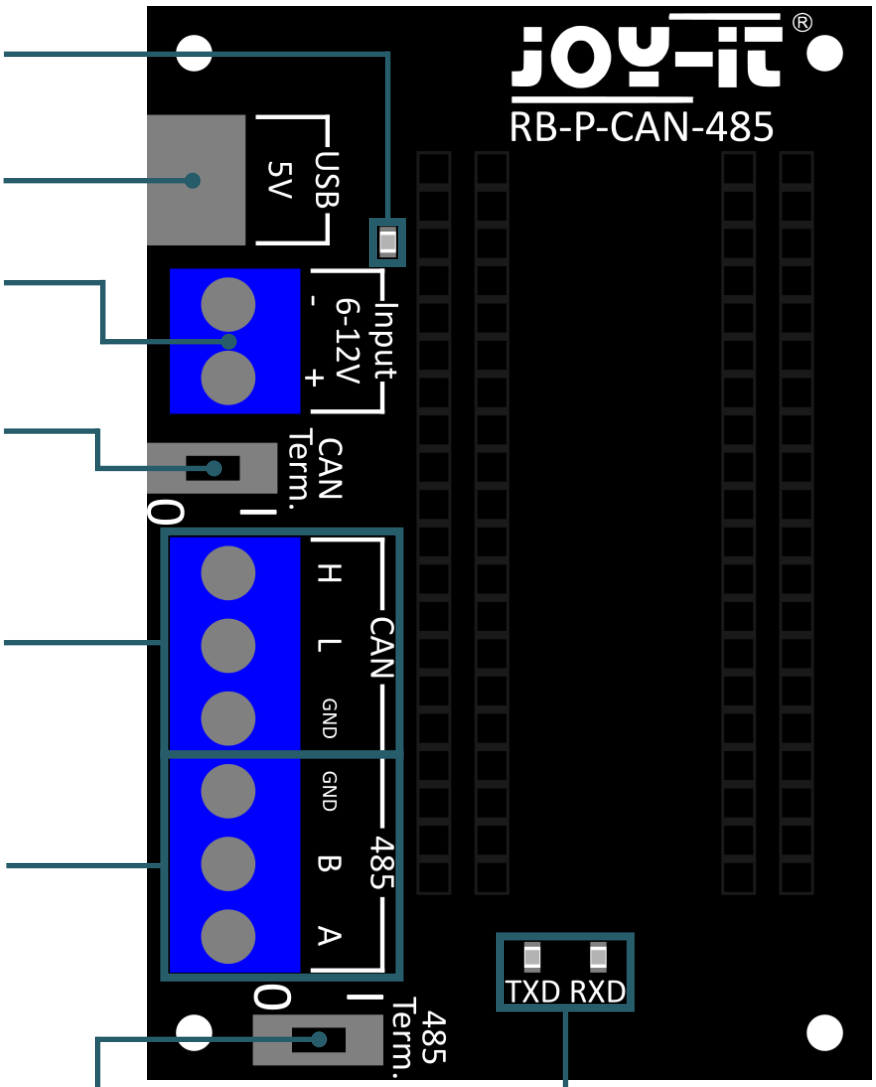
CAN Term. (terminaison CAN) :
Commute une résistance de 120Ω sur la position "I", qui est utilisée pour la terminaison spécialisée du bus.

CAN :
H : Connexion à la ligne CAN-High.
L : Connexion à la ligne CAN-Low.
GND : Connexion à la terre pour l'égalisation du potentiel entre les nœuds.

485 :
A : Connexion à la ligne de données négative.
B : Connexion à la ligne de données positive.
GND : Connexion à la terre pour l'égalisation du potentiel.

485 Term. (Terminaison 485) :
Commute une résistance de 120Ω sur la position "I", qui est utilisée pour la terminaison spécialisée du bus.

TXD | RXD (DEL d'état 485) :
Affiche l'état actuel de la ligne d'émission et de réception.
Un signal élevé entraîne l'allumage des DEL.



3. RASPBERRY PI PICO

Connectez maintenant un câble micro USB à votre ordinateur et au Raspberry Pi Pico pour la programmation.

ATTENTION : la connexion USB-C sur la carte est uniquement utilisée pour l'alimentation électrique. Aucune donnée n'est transférée au Raspberry Pi Pico.

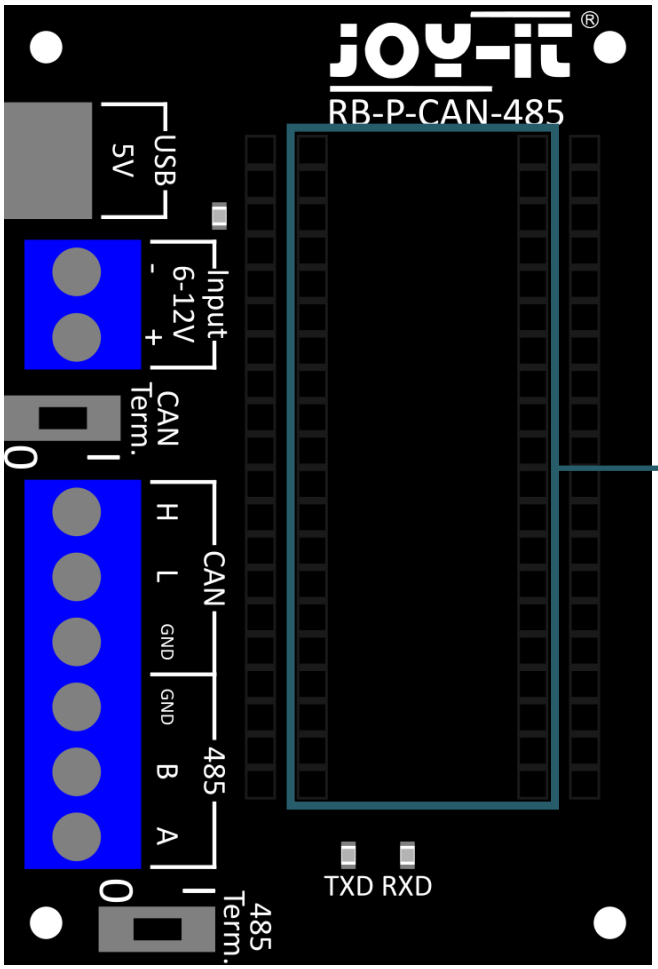
Vous pouvez utiliser un programme de développement approprié de votre choix pour transférer les exemples de programmes. Nous recommandons **l'Thonny IDE**.

ATTENTION : Si vous êtes nouveau dans le monde des microcontrôleurs et de l'électronique, ne vous inquiétez pas ! Nous avons préparé un guide spécial pour les débutants. Ce guide est spécialement adapté aux besoins des débutants et explique comment utiliser le Raspberry Pi Pico étape par étape.
De la configuration de base à l'exécution de projets, ce guide vous accompagne tout au long du processus. Notre guide comprend des explications faciles à comprendre et des conseils utiles pour vous aider à développer rapidement et efficacement vos compétences à grande échelle avec le Raspberry Pi Pico. Vous pouvez télécharger notre guide **ici**.

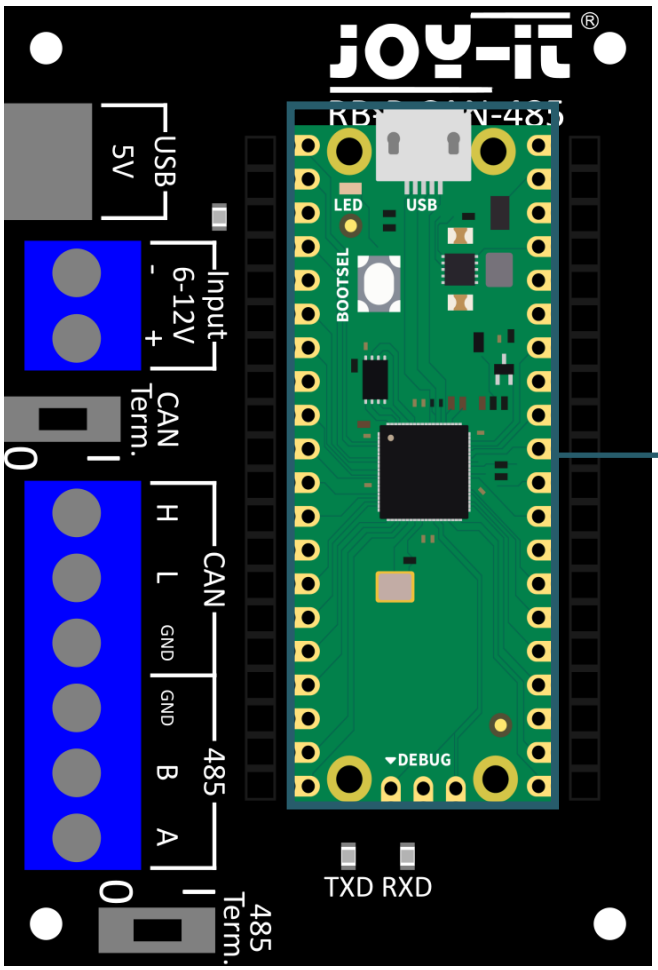
4. CONNEXION DU RASPBERRY PI PICO

Dans la section suivante, nous verrons comment connecter correctement le Raspberry Pi Pico à la carte d'extension.

Vous trouverez ci-dessous une illustration de la manière dont le Raspberry Pi Pico doit être branché sur la carte d'extension.



Emplacement pour le Raspberry Pi Pico :
Le Raspberry Pi Pico est branché ici.



Si le Raspberry Pi Pico a été branché, votre emplacement devrait ressembler à ceci.

5. EXEMPLE DE CODE RS485

Maintenant que vous avez configuré votre Raspberry Pi Pico, vous pouvez télécharger les deux exemples de code suivants pour la communication via l'interface RS485 sur votre Raspberry Pi Pico. Vous pouvez également télécharger les exemples de code **ici**.

Exemple de code de l'émetteur :

```
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))
```

Exemple de code du récepteur :

```
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. L'EXEMPLE DE CODE PEUT

Maintenant que vous avez configuré votre Raspberry Pi Pico, vous pouvez télécharger les deux exemples de code suivants pour la communication via l'interface CAN sur votre Raspberry Pi Pico.

La bibliothèque correspondante est nécessaire pour utiliser ces exemples. Nous utilisons la bibliothèque **MicroPython_CAN_BUS_MCP2515** de **Longan-Labs**, qui est publiée sous la **licence MIT**.

Exemple de code de l'émetteur de la bibliothèque :

```
'''
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
```

Exemple de code du récepteur de la bibliothèque :

```
'''
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. INFORMATIONS COMPLÉMENTAIRES

Nos obligations d'information et de reprise conformément à la loi sur les équipements électriques et électroniques (ElektroG)

Symbole sur les équipements électriques et électroniques :

Cette poubelle barrée signifie que les appareils électriques et électroniques ne doivent pas être jetés dans les ordures ménagères. Vous devez remettre les appareils usagés à un point de collecte.

Avant de remettre les déchets, les piles et les accumulateurs qui ne sont pas inclus dans les déchets doivent être séparés de ces derniers.

Possibilités de retour :

En tant qu'utilisateur final, vous pouvez retourner gratuitement votre ancien appareil (qui remplit essentiellement la même fonction que le nouvel appareil acheté chez nous) pour qu'il soit éliminé lors de l'achat d'un nouvel appareil.

Les petits appareils dont les dimensions extérieures ne dépassent pas 25 cm peuvent être éliminés dans le cadre de la gestion normale des déchets ménagers, indépendamment de l'achat d'un nouvel appareil.

Possibilité de retour dans nos locaux pendant les heures d'ouverture :

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

Possibilité de retour dans votre région :

Nous vous enverrons un timbre de colis avec lequel vous pourrez nous renvoyer l'appareil gratuitement. Veuillez nous contacter par courrier électronique à l'adresse Service@joy-it.net ou par téléphone.

Informations sur l'emballage :

Si vous ne disposez pas de matériel d'emballage approprié ou si vous ne souhaitez pas utiliser le vôtre, veuillez nous contacter et nous vous enverrons un emballage approprié.

8. SOUTIEN

Si des questions restent en suspens ou si des problèmes surviennent après votre achat, nous vous aiderons par courrier électronique, par téléphone et par le biais de notre système d'assistance par tickets.

Email: **service@joy-it.net**
Système de tickets: **http://support.joy-it.net**
Téléphone : +49 (0)2845 9360-50 (Lun - Jeu : 09:00 - 17:00 heures,
Vendredi : 09:00 - 14:30 heures)

Pour de plus amples informations, veuillez consulter notre site web :
www.joy-it.net

Publié: 27.03.2024