RS485 CAN HAT

Introduction

The RS485 CAN HAT will enables your Pi to communicate with other devices stably in long-distance via RS485/CAN functions.

Features

- Standard Raspberry Pi 40pin GPIO header for Raspberry Pi series boards.
- CAN function, onboard CAN controller MCP2515 via SPI interface, onboard transceiver SIT65HVD230DR.
- RS485 function, controlled via UART, half-duplex communication, supports automatic TX/RX control without programming, onboard transceiver SP3485.
- Onboard TVS (Transient Voltage Suppressor), effectively suppresses surge voltage and transient spike voltage in the circuit for RS485 transceiving, lightningproof & anti-electrostatic.
- Reserved control header, allows working with other control boards.

- Comes with online development resources and manual (examples in wiringPi/python).

Specification

- Operating voltage: 3.3V
- CAN chip: MCP2515
- CAN transceiver: SN65HVD230
- 485 transceiver: SP3485
- Dimensions: 65mm x 30mm
- Mounting hole size: 3.0mm

Onboard Interface

- CAN Bus:

Function PIN	Raspberry Pi (BCM)	Description
3V3	3V3	3.3V Power Input
GND	GND	Ground
SCK	SCK	SPI Clock Input
MOSI	MOSI	SPI Data Input
MISO	MISO	SPI Data Output
CS	CE0	Data/Command Selection
INT	25	Interrupt Output

- RS485 Bus:

Function PIN	Raspberry Pi (BCM)	Description
3V3	3V3	3.3V Power Input
GND	GND	Ground
RXD	RXD	UART Receives
TXD	TXD	UART Transmits
RSE	4	Tx/Rx Control

For the RSE pin, you can choose not to use it, the module factory defaults to use the hardware automatic receive and transmit.

Hardware Description

CAN Bus

The functionality of the CAN module involves handling all message reception and transmission on the CAN bus. When sending a message, it is first loaded into the appropriate message buffer and control registers. Sending operations can be initiated by setting the corresponding bits in the control registers via the SPI interface or by using the transmit enable pin. Communication status and errors can be checked by reading the respective registers.

Any messages detected on the CAN bus undergo error checking, and then are matched with user-defined filters to determine whether the message should be moved to one of the two receive buffers.

As the Raspberry Pi itself does not support the CAN bus, we can use the CAN controller with the SPI interface, along with a transceiver, to realize CAN functionality.

Microchip's MCP2515 is a CAN protocol controller that fully supports the CAN V2.0B technical specification. The device can send and receive both standard and extended data frames as well as remote frames. With two acceptance mask registers and six acceptance filter registers built in, the MCP2515 can filter out unwanted messages, reducing the overhead on the main microcontroller (MCU). The MCU connects to the device via the SPI interface, that is, the Raspberry PI connects to the chip through the SPI interface. For Raspberry Pi, using this chip does not require writing drivers; instead, simply enabling the kernel driver in the device tree is sufficient for usage.
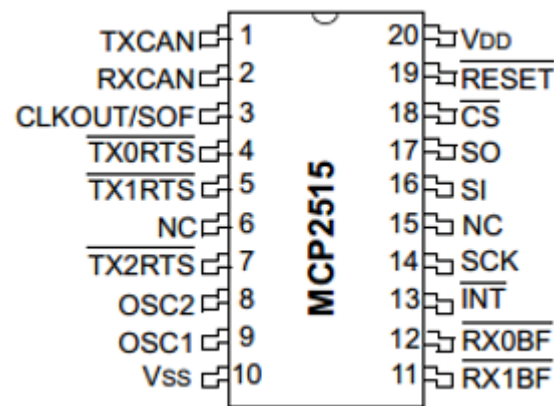
(/wiki/File:MCP2515-PIN.png)

For more details, you can refer to the user manual.

SN65HVD230 is a 3.3V CAN transceiver manufactured by Texas Instruments. This device is suitable for serial communication on CAN buses with higher communication speeds, good noise immunity, and high reliability. SN65HVD230 offers three different operating modes: high-speed, slope and standby. The mode control can be achieved through the Rs control pin. The output pin Tx of the CAN controller is connected to the data input pin D of SN65HVD230, enabling the transmission of data from this CAN node to the CAN network. Similarly, the receive pin Rx of the CAN controller is connected to the data output pin R of SN65HVD230 for receiving data.
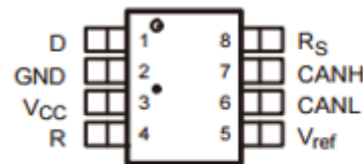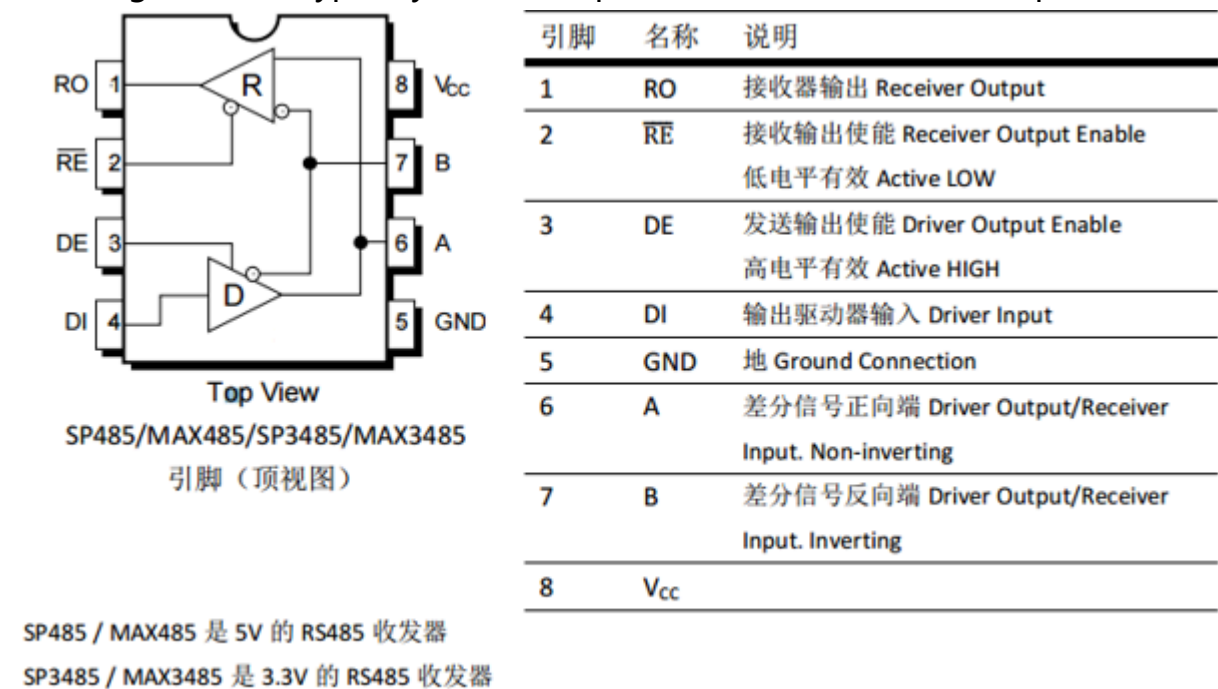

SN65HVD230 引脚
（顶视图）

引脚	名称	说明
1	D	驱动输入 Driver input
2	GND	电源地线 Ground
3	Vcc	电源线 Supply voltage
4	R	接收输出 Receiver output
5	Vref	参考输出 Reference output
6	CANL	低总线输出 Low bus output
7	CANH	高总线输出 High bus output
8	RS	工作模式控制端 Standby/slope control

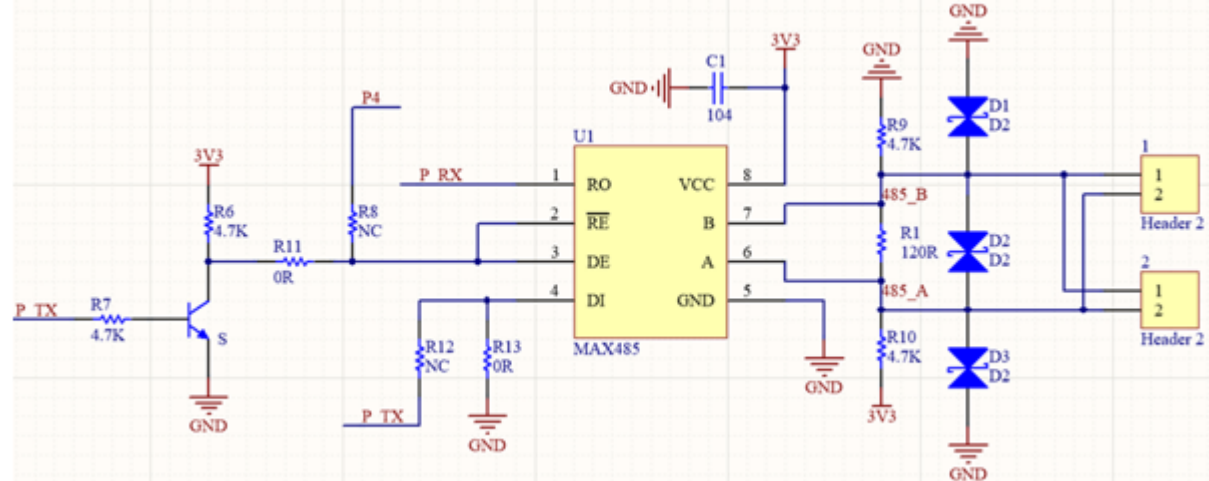(/wiki/File:RS485_CAN_HAT_(B)06.png)

RS485 Bus

The SP3485 interface chip is an RS-485 driver chip used for low-power transceiver communication on RS-485 networks. It operates on a single +3.3V power supply and employs half-duplex communication. The RO and DI pins serve as the output of the receiver and the input of the driver, respectively. The (RE) and DE pins act as the enable pins for receive and transmit operations; the device is in receive mode when (RE) is logic low, and in transmit mode when DE is logic high. The A and B pins are the differential signal lines for receive and transmit operations. When A-B > +0.2V, RO outputs logic 1; when A-B < -0.2V, RO outputs logic 0. Matching resistors, typically 100Ω, are placed between the A and B pins.

Top View
SP485/MAX485/SP3485/MAX3485
引脚（顶视图）

引脚	名称	说明
1	RO	接收器输出 Receiver Output
2	RE	接收输出使能 Receiver Output Enable 低电平有效 Active LOW
3	DE	发送输出使能 Driver Output Enable 高电平有效 Active HIGH
4	DI	输出驱动器输入 Driver Input
5	GND	地 Ground Connection
6	A	差分信号正向端 Driver Output/Receiver Input. Non-inverting
7	B	差分信号反向端 Driver Output/Receiver Input. Inverting
8	Vcc	

SP485 / MAX485 是 5V 的 RS485 收发器
SP3485 / MAX3485 是 3.3V 的 RS485 收发器

(/wiki/File:RS485_CAN_HAT_(B)0006.png)
Among them, the RE and DE pins of the SP3485 chip are set to receive and transmit;
The default factory setting of this module is to use hardware automatic sending and receiving, or you can choose to control the pins on the software to choose to send and receive, you can choose the control method by soldering the 0-ohm resistor on the board.
Hardware automatic control:

(/wiki/File:RS485_CAN_HAT_(B)07.png)

Data Rx: When P_TX is high, it's in an idle state. At this point, the transistor conducts, and the RE pin of the SP3485 chip is at a low level, enabling data reception. RO begins to receive data and forwards the data received at the 485AB port to the MCU.

Data Tx: When P_TX is pulled low, indicating the start of data transmission, the transistor is cut off, and the DE pin is set to a high level, enabling data transmission. At this point, if the data being transmitted is '1', the transistor will be conducting. Although reception becomes valid, the chip remains in a high-impedance state during the transmission phase, so it continues to maintain the transmission state, allowing for the normal transmission of '1'.

Note: the use of an automatic transceiver due to the on-off speed of the transistor, will lead to the automatic transceiver baud rate that can not be too high, if you need a very high baud rate, it is recommended to use the manual transceiver.

Working with Raspberry Pi

If you use bookworm system, only the lgpio library is available, bcm2835 and wiringPi libarary cannot be installed or used. Please note that the python library does not need to be installed, you can directly run the

demo.

BCM2835

```
#Open the Raspberry Pi terminal and run the following commands:
sudo wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.71.tar.gz
sudo tar zxvf bcm2835-1.71.tar.gz
cd bcm2835-1.71/
sudo ./configure && sudo make && sudo make check && sudo make install
# For more information, please refer to the official website: http://www.airspayce.com/mike
m/bcm2835/
```

wiringPi

```
#Open the Raspberry Pi terminal and run the following commands:
cd
sudo apt-get install wiringpi
#For Raspberry Pi systems after May 2019 (those earlier may not require execution), an upgr
ade may be necessary:
wget https://files.waveshare.com/wiki/common/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# Run gpio -v and version 2.52 will appear. If it does not appear, there is an installation
error

#Bullseye branch system uses the following command:
sudo git clone https://github.com/WiringPi/WiringPi
cd WiringPi
sudo ./build
gpio -v
# Run gpio -v and version 2.70 will appear. If it does not appear, there is an installation
error
```

lgpio

```
sudo su
wget https://github.com/joan2937/lg/archive/master.zip
unzip master.zip
cd lg-master
sudo make install
# For more information, please refer to the official website: https://github.com/gpiozero/l
g
```

Python3

```
sudo apt-get update
sudo apt-get install python3-serial
sudo apt-get install python3-can
sudo apt install python3-rpi-lgpio
```

CAN Usage

This demo uses two Raspberry Pi and two RS485 CAN HATs.

Provide python and c demos.

Preparation

Insert the module into the Raspberry Pi, modify the start-up script "config.txt".

```
sudo nano /boot/config.txt
sudo nano /boot/firmware  config.txt #Bookworm/Ubuntu
```

Add the following content at the file:

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000
```

"oscillator=12000000" is the onboard oscillator with the size of 12M as shown below:
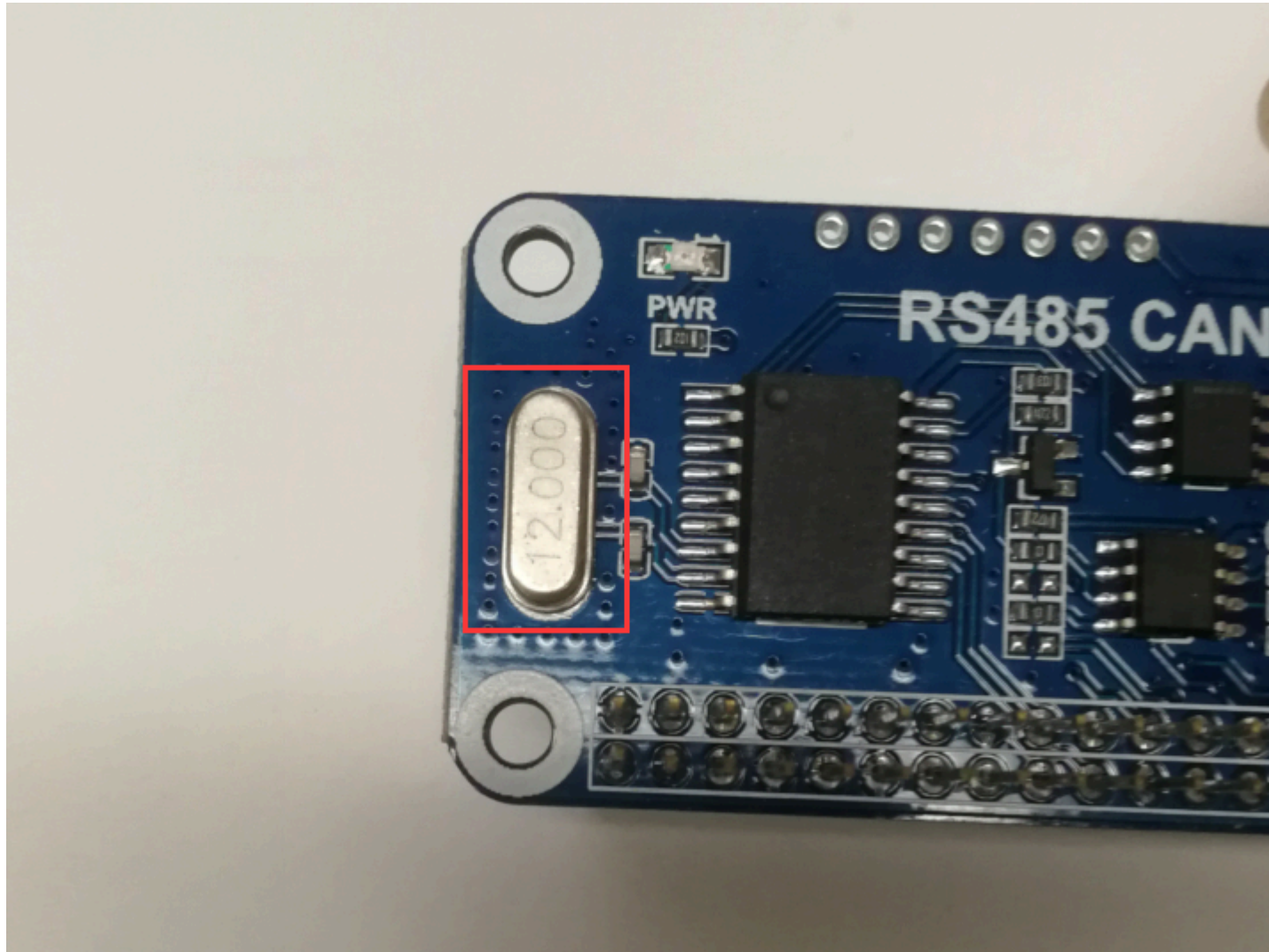
(/wiki/File:RS485_CAN_HAT_cry12.png)

- If the purchase date is earlier than August 2019, please use the following:

As shown in the figure, the red box is the 8M crystal oscillator.
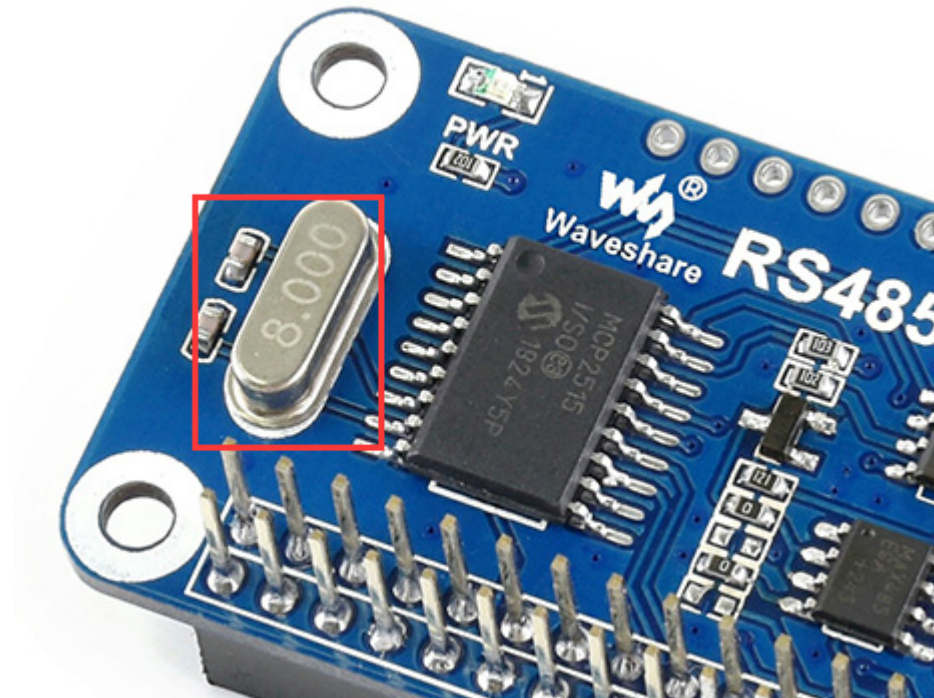
(/wiki/File:RS485_CAN_HAT_cry.png)

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=8000000,interrupt=25,spimaxfrequency=1000000
```

After saving and exiting, restart the Raspberry Pi:

```
sudo reboot
```

After rebooting, run the command to see if the initialization was successful:

```
dmesg | grep -i '\(can\|spi\)'
```

(/wiki/File:RS485_CAN_HAT_CAN1.png)

If the module is not connected the following may be prompted:

(/wiki/File:RS485_CAN_HAT_CAN2.png)

Please check if the module is connected. Whether to enable SPI and turn on the MCP2515
kernel driver. Whether reboot is performed.

Make sure that both sides of the Raspberry Pi are handled this way, and connect the H and L of
the two modules correspondingly.

If you are using another CAN device, make sure that the lines H-H, and L-L are connected.

- Open CAN:

```
sudo ip link set can0 up type can bitrate 1000000
sudo ifconfig can0 txqueuelen 65536
sudo ifconfig can0 up
```

- For more CAN kernel commands, you can refer to:

```
https://www.kernel.org/doc/Documentation/networking/can.txt (https://www.kernel.org/doc/Doc
umentation/networking/can.txt)
```

- View ifconfig:

```
ifconfig
```

```
pi@raspberrypi:~ $ ifconfig
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10
(UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.80  netmask 255.255.252.0  broadcast 192.168.11.255
        inet6 fe80::7af1:9de7:6749:e572  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:e6:84:86  txqueuelen 1000  (Ethernet)
        RX packets 7044  bytes 673465 (657.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 218  bytes 28256 (27.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(/wiki/File:RS485_CAN_HAT_cry16.png)

Simple Test

- This demonstration uses a Raspberry Pi, an RS485 CAN HAT (B) module and a USB-CAN-A module. Python and C language programs are provided:

(/wiki/File:Pi-can-a.png)

- Open CAN:
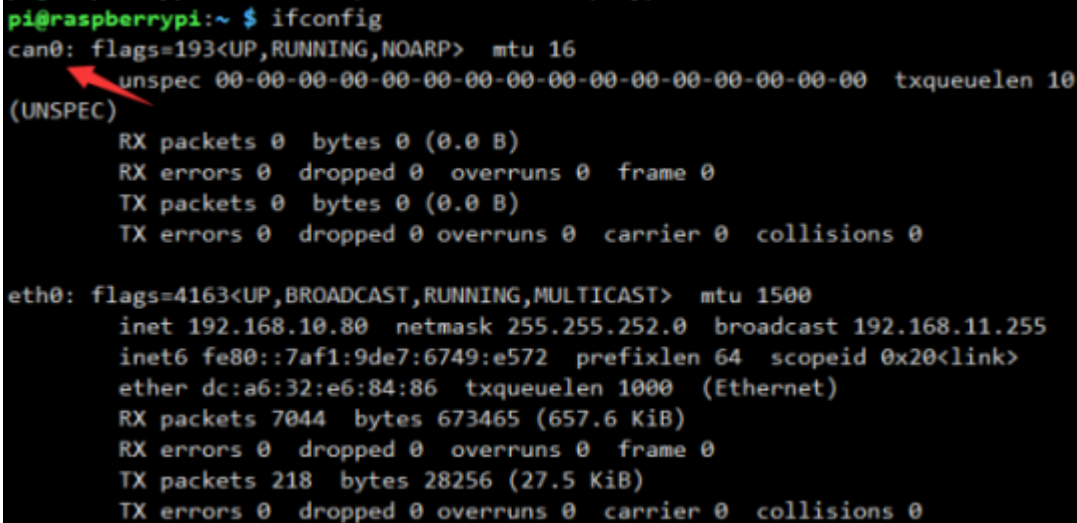
```
sudo ip link set can0 up type can bitrate 1000000
sudo ifconfig can0 txqueuelen 65536
```

- For more CAN kernel commands, please refer to:

```
https://www.kernel.org/doc/Documentation/networking/can.txt (https://www.kernel.org/doc/Doc
umentation/networking/can.txt)
```

- Check ifconfig:

```
ifconfig
```


(/wiki/File:RS485_CAN_HAT_cry16.png)

- Install can-utils:

```
sudo apt-get install can-utils
```

- Receive:

The command for inputting the receiving data in the terminal:

```
candump can0
```

The receiving tool is blocked. When running the tool without parameters, it will always be in the receiving state. Use Ctrl+C to exit.
A description of the parameters of the tool can be used:

```
candump -h
```

- Sending

Input the command for sending the data in the terminal:

```
cansend can0 000#11.22.33.44
```

The result is as shown below:

(/wiki/File:RS485_CAN_HAT_B_Rasp0112.png)

Demo Download

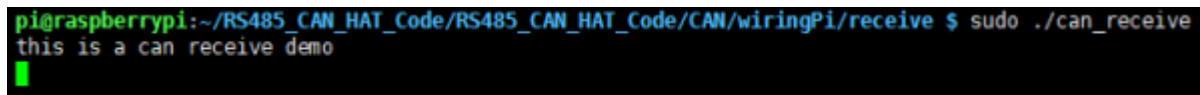Run the following content in the Raspberry Pi terminal:

```
sudo apt-get install unzip
wget https://files.waveshare.com/upload/4/4e/RS485_CAN_HAT_Code.zip
unzip RS485_CAN_HAT_Code.zip
sudo chmod 777 -R RS485_CAN_HAT_Code/
```

C

- Block receiving: Open the terminal on the Raspberry Pi and run:

```
cd RS485_CAN_HAT_Code/CAN/wiringPi/receive/
make clean
make
sudo ./can_receive
```

The receiving demo is blocked, and it ends as soon as the data is read.

(/wiki/File:RS485_CAN_HAT_B_Rasp10.png)

- Send: the Raspberry Pi opens the terminal and runs:

```
cd RS485_CAN_HAT_Code/CAN/wiringPi/send/
make clean
make
sudo ./can_send
```

```
pi@raspberrypi:~/RS485_CAN_HAT_Code/RS485_CAN_HAT_Code/CAN/wiringPi/send $ sudo ./can_send
this is a can send demo
can_id  = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
```

(/wiki/File:RS485_CAN_HAT_B_Rasp11.png)

At this point, receive the message corresponding to id:

```
pi@raspberrypi:~/RS485_CAN_HAT_Code/RS485_CAN_HAT_Code/CAN/wiringPi/receive $ sudo ./can_receive
this is a can receive demo
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
```

(/wiki/File:RS485_CAN_HAT_B_Rasp12.png)

python

Raspberry Pi opens the terminal and runs:

```
cd RS485_CAN_HAT_Code/CAN/python/
#Run reception first:
sudo python reveive.py
#The sending terminal:
sudo python send.py
```

How to Use It with Other CAN devices

1. Ensure that the hardware wiring is correct, i.e., H-H and L-L connections.

2. Ensure that the baud rate settings are the same on both sides, the default demo sets the baud rate to 100K.

(/wiki/File:RS485_CAN_HAT_B_Rasp13.png)

3. Ensure that the CAN IDs on both sides are the same, otherwise it will not be possible to receive.

4. If there is frame loss when sending data for a long time, you can try to reduce the baud rate to solve the problem.

RS485 Usage

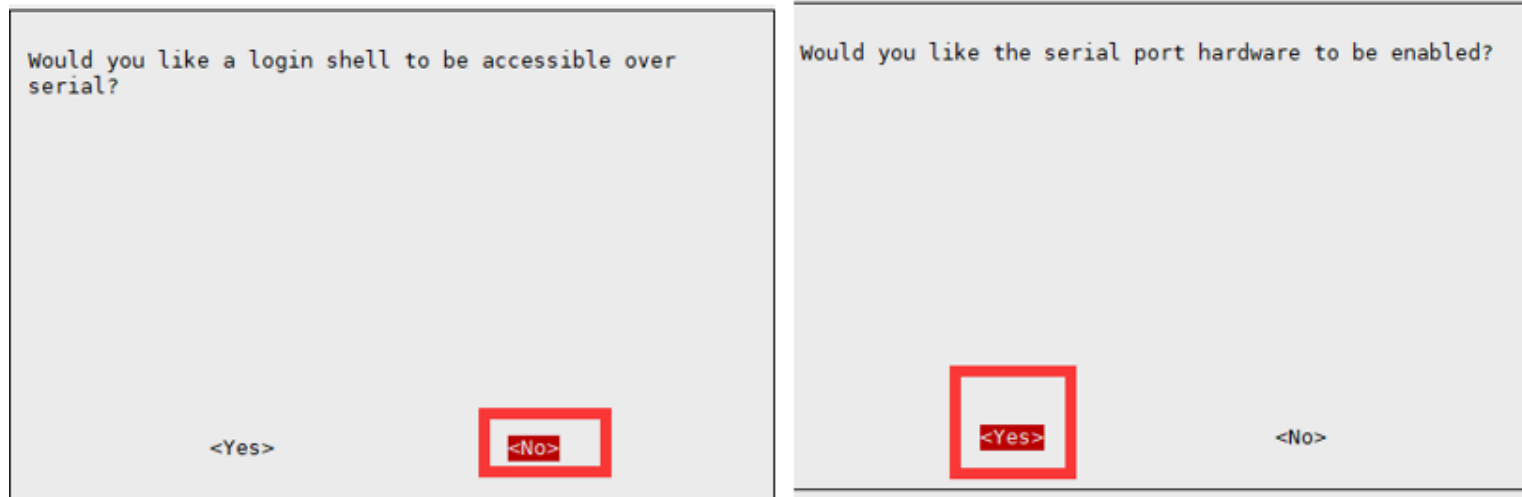This demo uses two Raspberry Pi and two RS485 CAN HAT modules.

Provide Python and wiringPi demos.

Enable UART Interface

Open the Raspberry Pi terminal and enter the following command to enter the configuration interface

```
sudo raspi-config
Select Interfacing Options -> Serial, disable shell access, and enable the hardware serial
port
```
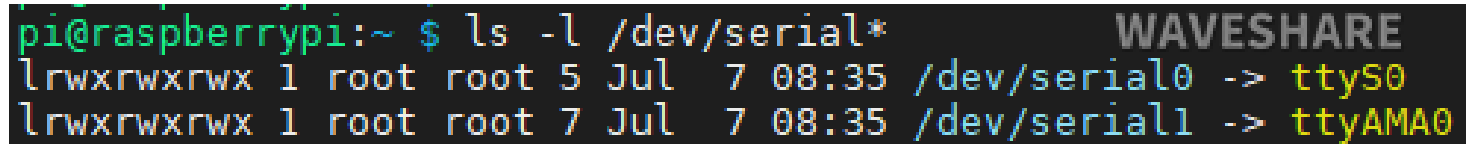
For Raspberry Pi 3B users, the serial port is used for Bluetooth and needs to be commented out:

```
#dtoverlay=pi3-miniuart-bt
```

For Raspberry Pi 5/2B/zero and some Raspberry Pi that have been changed in configuration, the user's serial device number may be ttyAMA0; you can use the following command line to confirm, serial0 for the selection of the serial device number, as follows:

```
ls -l /dev/serial*
```


(/wiki/File:RM520n-gl_faq90.png)

And then reboot the Raspberry Pi:

```
sudo reboot
```

Make sure that both sides of the Raspberry Pi are handled this way, connecting the two modules A and B correspondingly.

If using other 485 devices, make sure to connect A-A and B-B.

C

- Blocking reception, the Raspberry Pi opens a terminal and runs:

```
cd RS485_CAN_HAT_Code/485/WiringPi/receive
make clean
make
sudo ./485_receive
```

The receiving demo is blocked until the data is read and then it ends.

(/wiki/File:RS485_CAN_HAT_B_RS011.png)

- Send, the Raspberry Pi opens the terminal and runs:

```
cd RS485_CAN_HAT_Code/485/WiringPi/send
make clean
make
sudo ./485_send
```

At this point the receiver receives the demo:
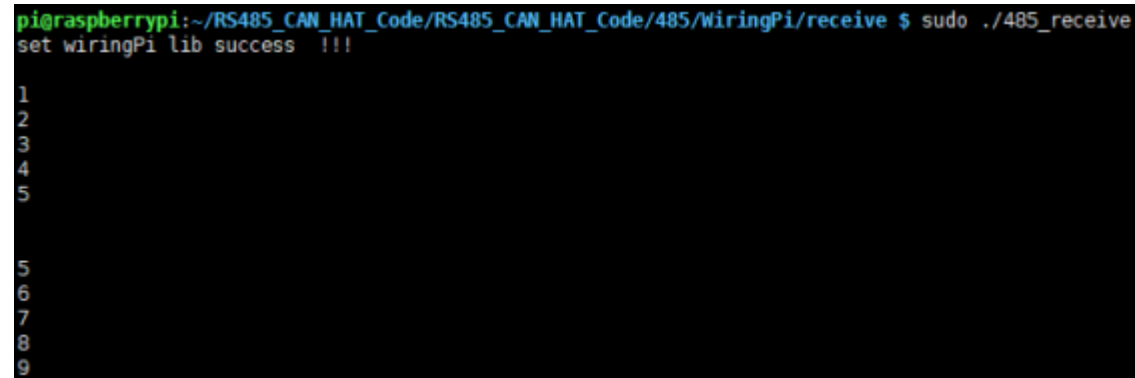


```
pi@raspberrypi:~/RS485_CAN_HAT_Code/RS485_CAN_HAT_Code/485/WiringPi/receive $ sudo ./485_receive
set wiringPi lib success  !!!

1
2
3
4
5


5
6
7
8
9
```

(/wiki/File:RS485_CAN_HAT_B_RS012.png)

python Demo

```
cd RS485_CAN_HAT_Code/485/python/
#Run reception first:
sudo python receive.py
#Sending terminal:
sudo python send.py
```

Troubleshooting

If the 485 communication is abnormal, please try the following steps:

1. Verify that logging into the Raspberry Pi shell from serial is disabled;
2. Determine the hardware version of the Raspberry Pi, if it is a Raspberry Pi ZERO/3B, the serial port in the program may need to be modified to /dev/ttyAMA0;
3. Determine whether 485's A,B correspond one-to-one with the controlled 485 devices A, B;
4. You can first use a USB to 485 device to communicate with the RS485 CAN HAT to ensure that the Raspberry Pi is set up correctly.

Resources

Documents

- User Manual (https://files.waveshare.com/upload/2/29/RS485-CAN-HAT-user-manuakl-en.pdf)
- Schematic (https://files.waveshare.com/upload/1/1d/RS485_CAN_HAT_Schematic.pdf)

Demo code

- Demo code (https://files.waveshare.com/upload/4/4e/RS485_CAN_HAT_Code.zip)
- X3-Pi Demo code (https://files.waveshare.com/upload/4/4e/RS485_CAN_HAT_X3.zip)

Datasheet

- MCP2515 (https://files.waveshare.com/upload/8/83/MCP2515.pdf)
- SN65HVD230 (https://files.waveshare.com/upload/8/82/SN65HVD230.pdf)
- SP3481_SP3485 (https://files.waveshare.com/upload/3/36/SP3481_SP3485.pdf)

3D Drawing

- 3D Drawing (https://files.waveshare.com/upload/5/52/RS485_CAN_HAT_3D_Drawing.zip)

FAQ

Question: Imprint?

Answer:
- The current version is 12M, you can view the front of the module:

 (/wiki/File:RS485_CAN_HAT_cry12.png)

Then the corresponding command in config.txt is:

```
dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000
```

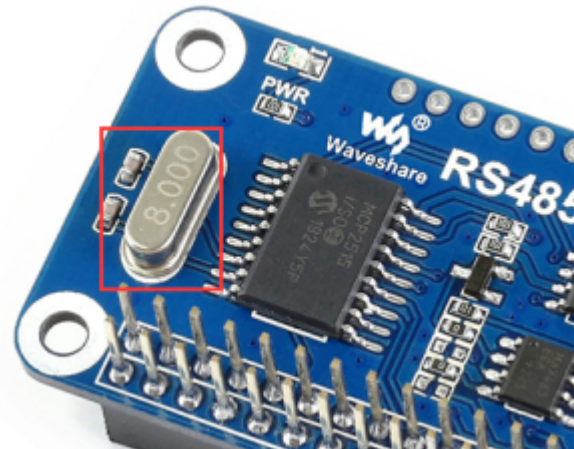- If you have an old version, there should be an 8M crystal oscillator, as shown below:

 (/wiki/File:RS485_CAN_HAT_cry.png)

Then the corresponding command in config.txt is:

```
dtoverlay=mcp2515-can0,oscillator=8000000,interrupt=25,spimaxfrequency=1000000
```

Question:Why can't I control the module with UART on Raspberry Pi, can't open ttyS0 through minicom, prompting no ttyS0?

Answer:

Open the Raspberry Pi terminal and enter the following command to access the configuration screen:

```
sudo raspi-config
Select Interfacing Options -> Serial to turn off shell access and turn on the hardware serial port
```
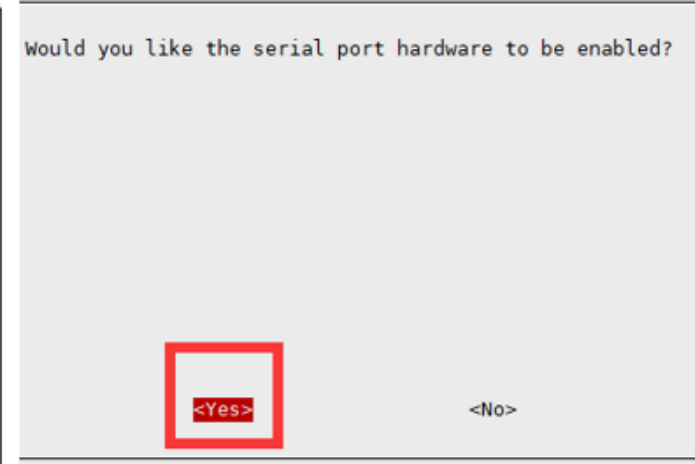

(/wiki/File:L76X_GPS_Module_rpi_serial.png)

- Raspberry Pi 5/3B/2B/zero, the user serial device number is ttyAMA0; you can use the following command line to confirm that serial0 is the selected serial device number, as follows:

```
ls -l /dev/serial*
```

```
pi@raspberrypi:~ $ ls -l /dev/serial*                    WAVESHARE
lrwxrwxrwx 1 root root 5 Jul  7 08:35 /dev/serial0 -> ttyS0
lrwxrwxrwx 1 root root 7 Jul  7 08:35 /dev/serial1 -> ttyAMA0
```
(/wiki/File:RM520n-gl_faq90.png)

- For Raspberry Pi 3B/2B/Zero, the serial port is typically used for Bluetooth. When you comment out this setting, it defaults to S0. However, Raspberry Pi 4 does not have this statement by default.

```
#dtoverlay=pi3-miniuart-bt
```

And then reboot the Raspberry Pi:

```
sudo reboot
```

Question:CAN can't send and receive data?

Answer:
1. Make sure the baud rates on both sides are the same;
2. The fixed frame ID is set in the demo: 0X123, please set the sending and receiving CAN ID of the other end of your CAN to be x0123;

Question:With or without isolation?

Answer:
- This is an entry-level 485 and CAN, both without isolation.

Question:485 Communication is abnormal, what should I do?

Answer:

1. Determine the hardware version of the Raspberry Pi, if it is the Raspberry Pi ZERO/3B, the serial port in the program needs to be modified to /dev/ttyAMA0;
2. Check whether the serial communication of the Raspberry Pi has enabled flow control;
3. Determine whether A and B of 485 correspond to the controlled 485 devices A and B one by one;
4. You can use the USB to 485 devices to communicate with the RS485 CAN HAT first to ensure that there is no problem with the settings of the Raspberry Pi;
5. Check the setting of odd and even bit parity of serial communication parameters.

Question:How to configure the config.txt file when the Ubuntu system is installed on the Raspberry Pi?

Answer:

- 1. In Mainstream Ubuntu system, the config.txt files are usually in the /boot/firmware folder:

```
sudo nano /boot/firmware  config.txt
```

Add the following content at the end:

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000
enable uart=1
```

- 2. Or use the SD card of the Raspberry Pi to read and change the config.txt file under the computer (or other hosts that can recognize the SD card) through the card reader.

Question:After connecting the sensor to RS485, the corresponding program does not receive data?

Answer:

The sensor may send hex data (sometimes it is necessary to send hex data to the sensor to request data), follow the steps below to send and receive hex data:

```
wget https://files.waveshare.com/upload/0/00/RS485-CAN-HAT-For-Hex.zip (https://files.wav
eshare.com/upload/0/00/RS485-CAN-HAT-For-Hex.zip)
unzip RS485-CAN-HAT-For-Hex.zip
sudo chmod 777 RS485-CAN-HAT-For-Hex.zip
cd RS485-CAN-HAT-For-Hex
#Receive Hex
sudo python3 RS485-CAN-HAT-send-hex.py
#Send Hex
sudo python3 RS485-CAN-HAT-receive-hex\.py
```

Question:CAN communication will get slower and slower, and then you can only often re-can port, sometimes indeed the speed of the program is quite slow?

Answer:

If it means that the output part of data processing becomes slower, it should help to improve the performance of the host computer; you can also consider clearing the cache before each read to read only the latest data for processing.

Question:Do I need to connect GND for RS485 communication?

Answer:

Can be used without connecting to the GND, connected to the GND signal may be more stable, connected to GND to obtain better performance, reliability, and anti-interference ability. For long-distance communication. Hence, it is recommended to connect the GND.

Question:If the installation of Python libraries fails on the latest Raspberry Pi system, what should you do?

Answer:

After entering "myenv", install the corresponding Python library. In the "myenv" environment, all commands are executed without "sudo".

```
sudo apt install python3-venv
python3 -m venv myenv &&source myenv/bin/activate
source myenv/bin/activate
pip install requests
```

Question:Why is the external USB-CAN box communication abnormal?

Answer:
- The CAN rate of the host and the slave should be the same.
- Both ends of the 120 ohm balancing resistors are turned on.
- The frame ID of the transmitter should correspond to the filter ID of the receiver.

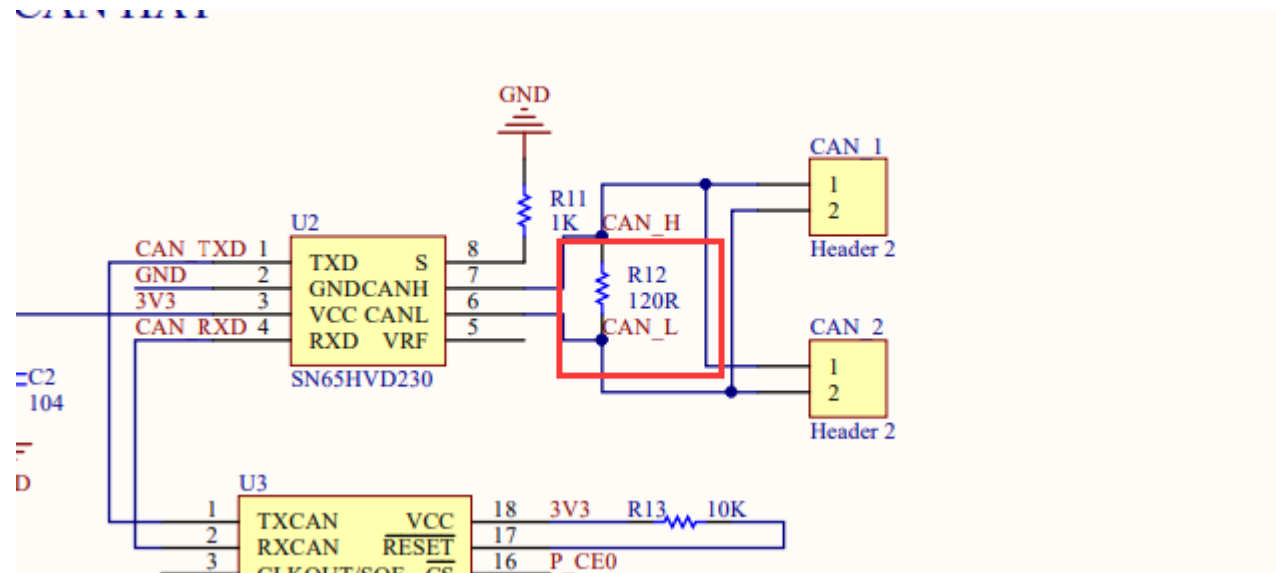## Question:I am looking for some documentation of the register for ModBus Comunicatons (Address)?

Answer:

The RS485 CAN HAT is an adapter only.

For Modbus transmit, you can only transmit modbus command directly.

Question:Does product RS485 CAN HAT for Raspberry Pi have a terminating resistor for the can bus?

Answer:

Yes, it does have the terminal resistor.

(/wiki/File:Rs485-can-hat-faq.png)

Question:The installation of wiringpi library is abnormal on 64-bit system, how can I install it?

Answer:

- Download 64-bit Arm wiringpi

```
wget https://gitee.com/LJYSCIENTIST/raspberry-pi-software-storage/raw/master/wiringpi-2.6
0-1_arm64.deb
```

- Install 64-bit Arm wiringpi

```
sudo dpkg -i wiringpi-2.60-1_arm64.deb
```

Question:How can I verify its functions if I only have one module?

Answer:
- It is possible to verify the functionality with loopback mode:
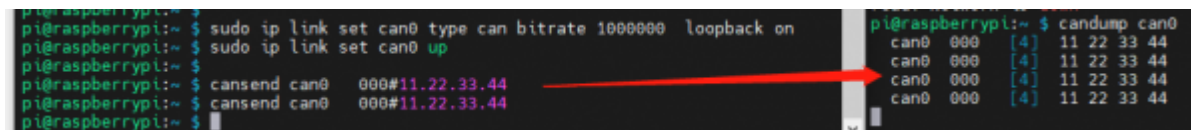- Install can-utils.

```
sudo apt-get install can-utils
```

- Open can0 and set loopback mode:

```
sudo ip link set can0 down
sudo ip link set can0 type can bitrate 1000000 loopback on
sudo ip link set can0 up type can bitrate 1000000
```

- Open two terminal windows, test the self-transmission of can0:

```
candump can0
cansend can0  000#11.22.33.44
```


(/wiki/File:Can_loopback.png)

Question:Does it support Raspberry Pi 4B?

Answer:

Yes, it supports.

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.

Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

Submit Now (https://service.waveshare.com/)