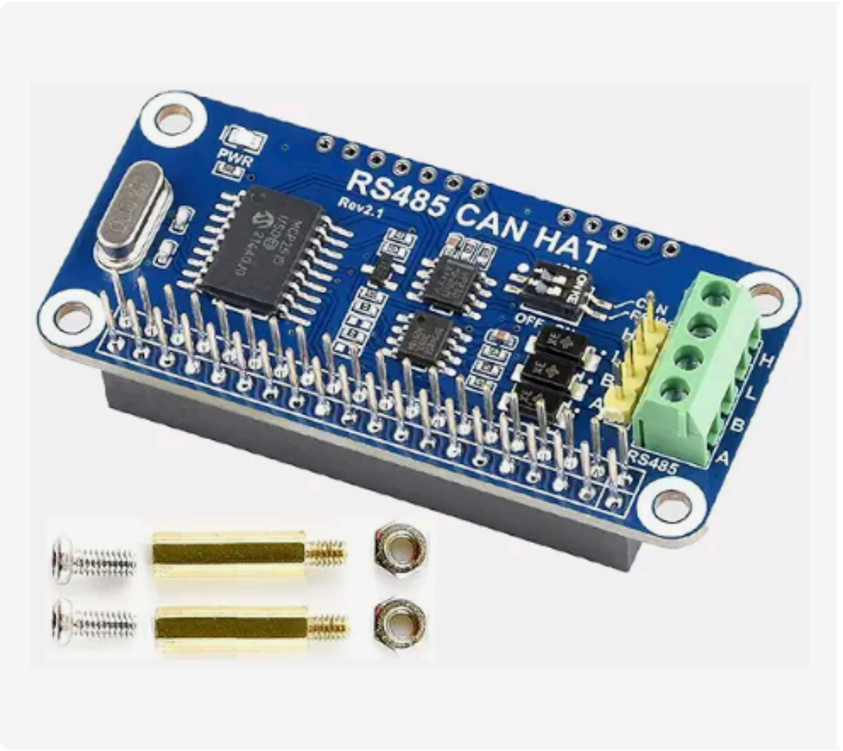
CAN RPi HAT

(Controller Area Network, Hardware Attached On Top)



NORMAN MCENTIRE

References

- https://www.waveshare.com/w/upload/2/29/RS485-CAN-HAT-user-manuakl-en.pdf

What is CAN bus?

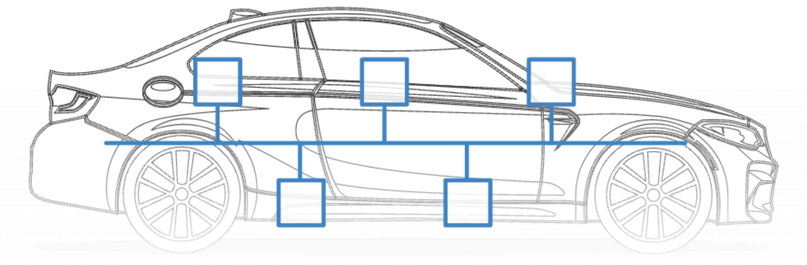
Your car is like a human body:

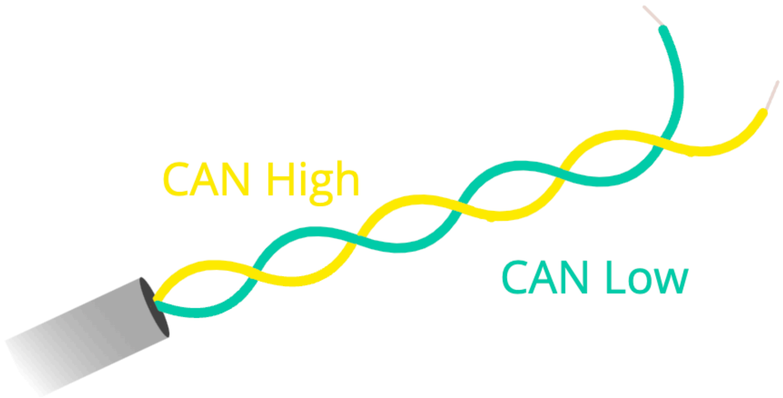
The Controller Area Network (CAN bus) is the **nervous system**, enabling communication.

In turn, 'nodes' or 'electronic control units' (ECUs) are like parts of the body, interconnected via the CAN bus. Information sensed by one part can be shared with another.

So what is an ECU?

In an automotive CAN bus system, ECUs can e.g. be the engine control unit, airbags, audio system etc. A modern car may have **up to 70 ECUs** - and each of them may have information that needs to be shared with other parts of the network.



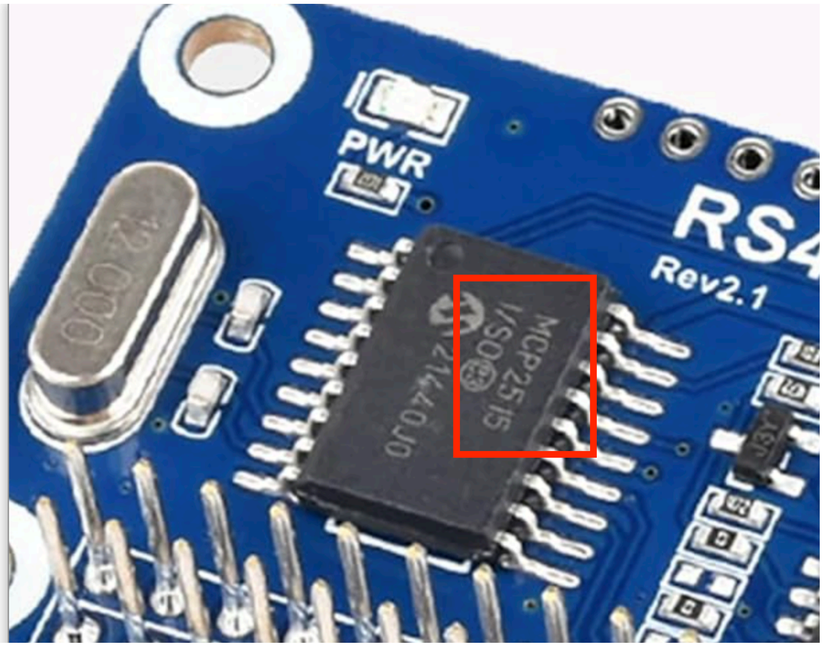


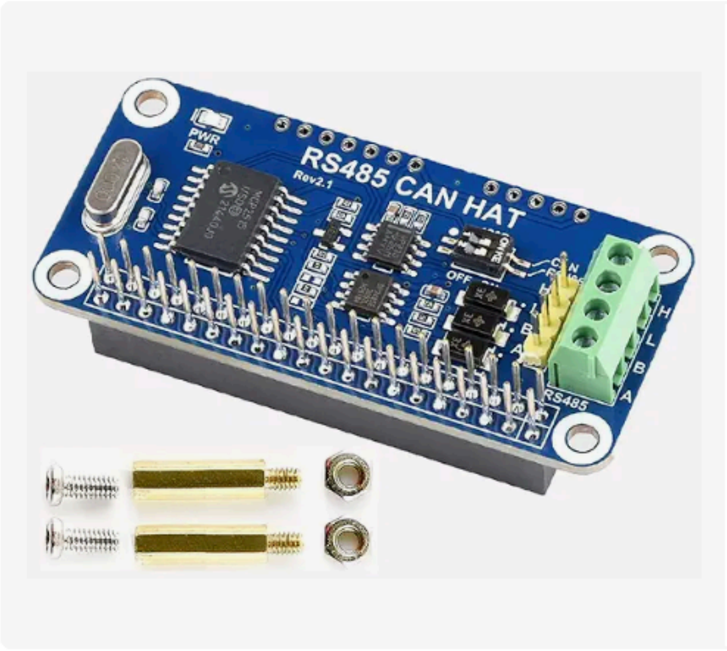
Introduction

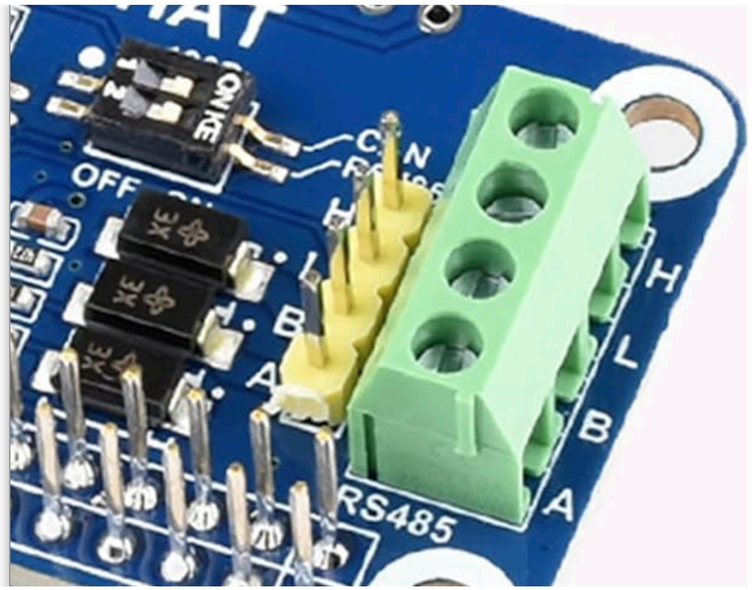
- CAN/SPI RS485/UART HAT enables RPI to communicate over long distance via CAN/RS485
  - CAN - Controller Area Network
    - SPI - Serial Peripheral Interface
      - MCP2515
      - SN65HVD230 Transceiver
  - HAT - Hardware Attached on Top
  - RS485 Uses UART for RS485 Half-Duplex
      - SP3485 Transceiver

Introduction - Uses MCP 2515 Chip

Dual Functions - SPI/CAN and UART/485



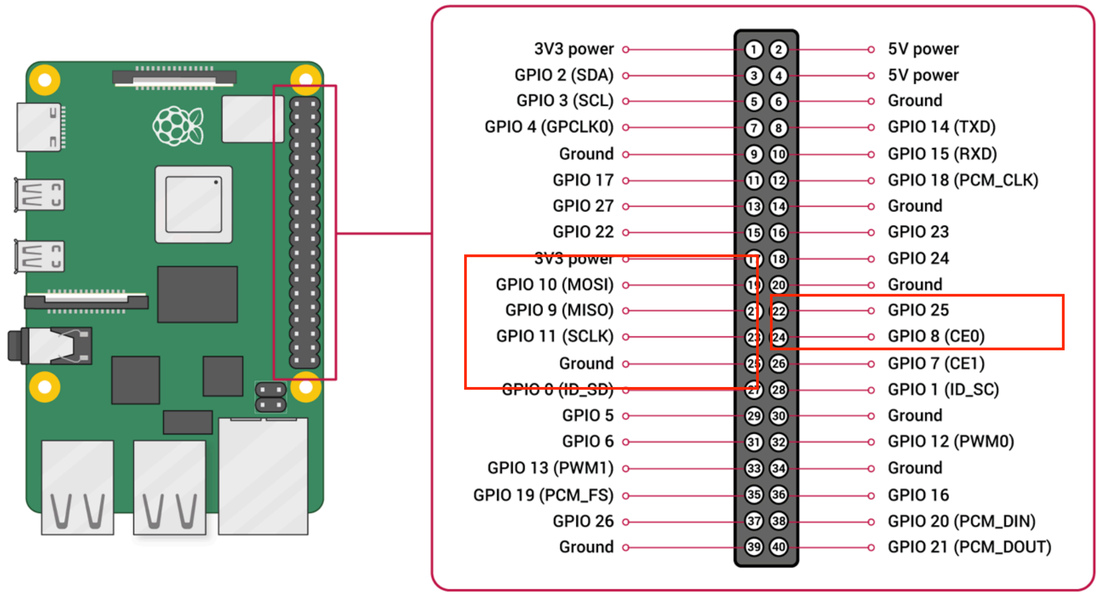




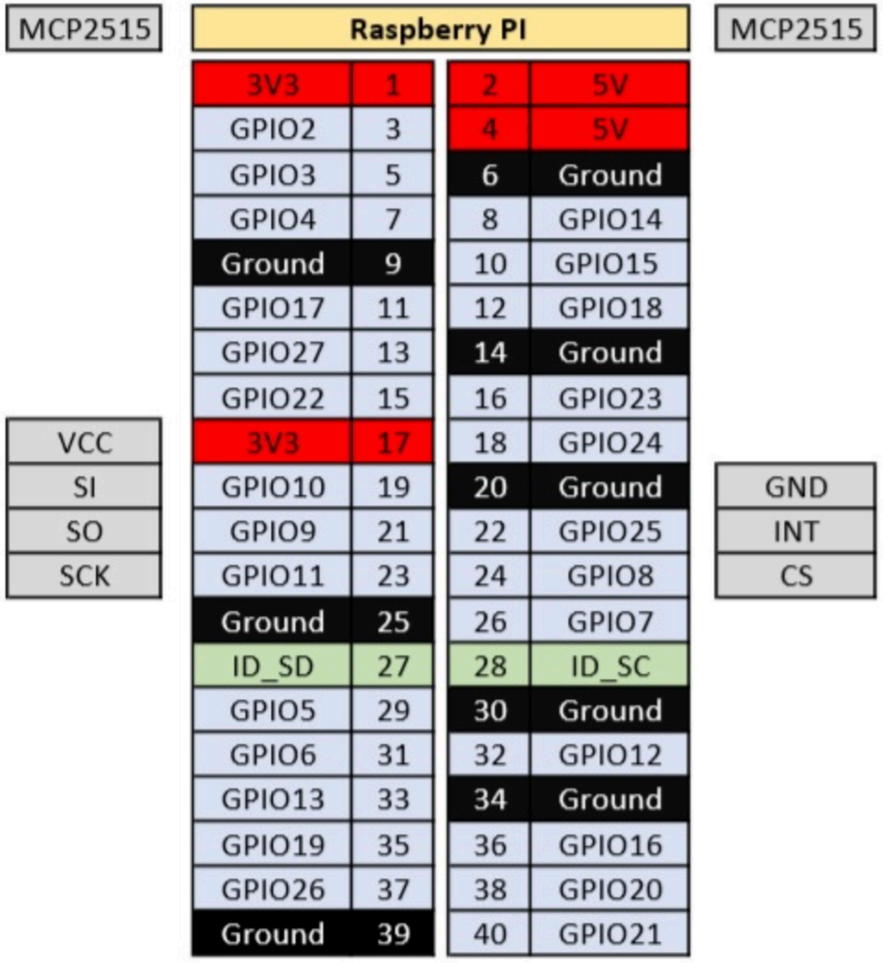
Interfaces - CAN

CAN:

PIN	Raspberry Pi	Description
3V3	3V3	3.3V Power
GND	GND	Ground
SCK	SCK	SPI Clock
MOSI	MOSI	SPI Data input
MISO	MISO	SPI Data output
CS	CE0	Data/Command selection
INT	PIN22 /GPIO.6/P25	Interrupt

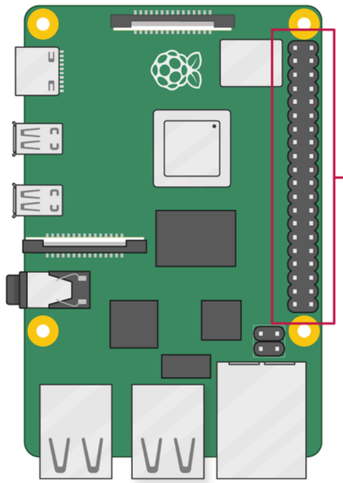


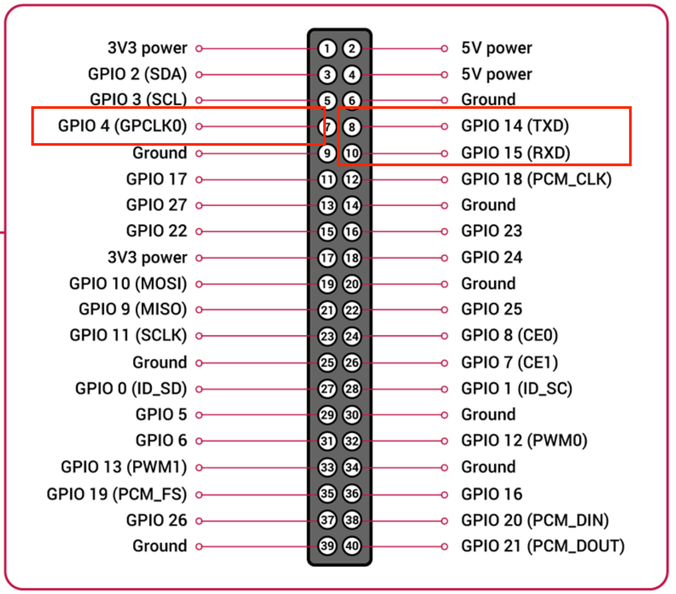
RPi 40-Pin Header - Overview



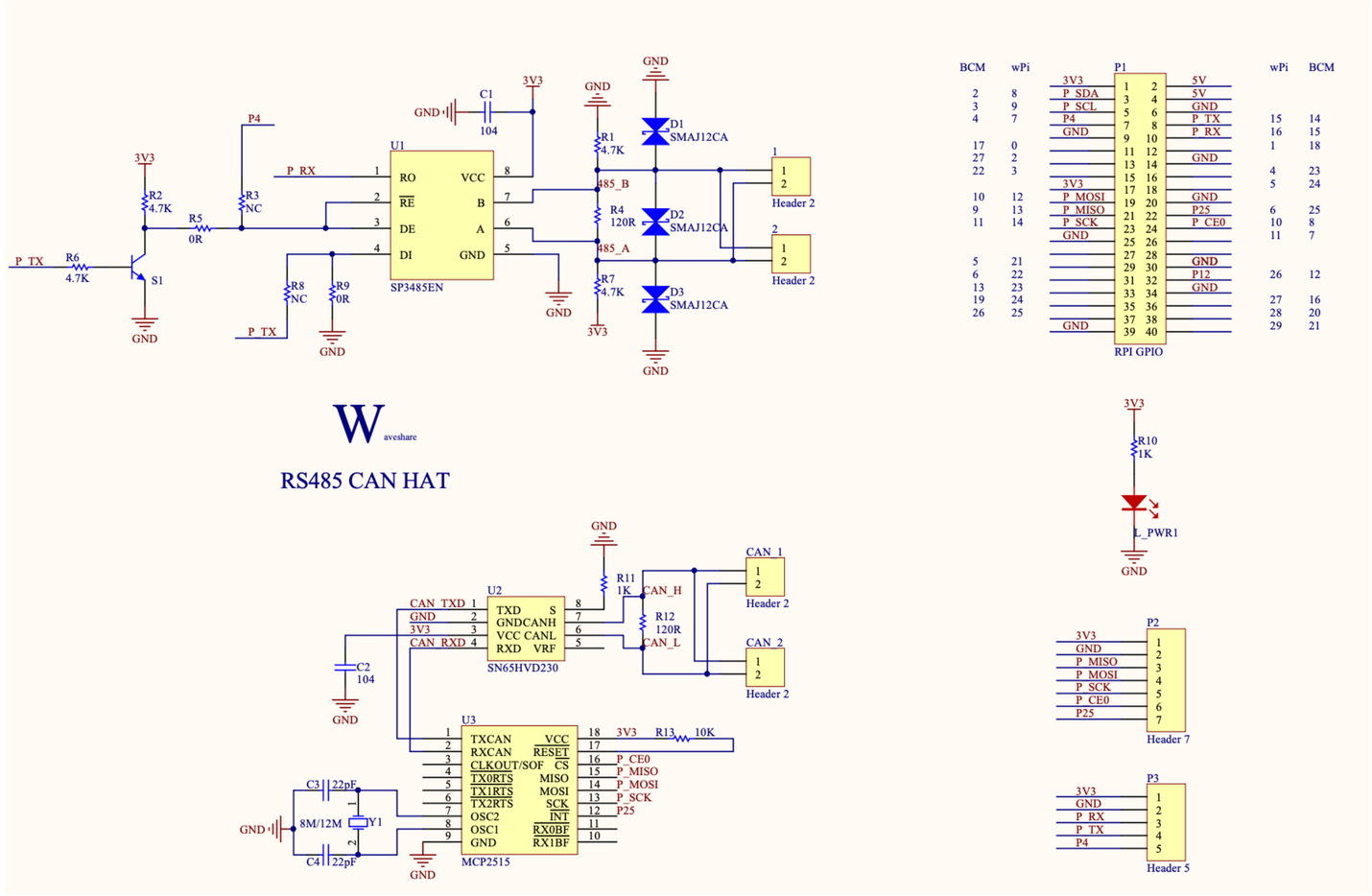
Interfaces - RS485

PIN	Raspberry Pi	Description
3V3	3V3	3.3V power
GND	GND	Ground
RXD	RXD	RS485 UART receive
TXD	TXD	RS485 UART transmit
RSE	PIN7/GPIO.7/P4	RS485 RX/TX setting





Schematic



Microchip MCP2515

MICROCHIP

MCP2515

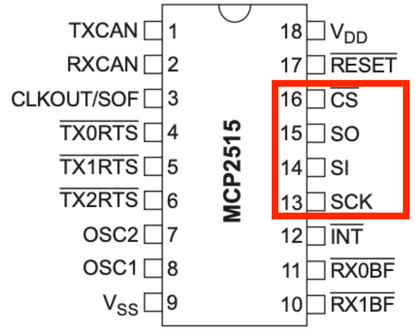
Stand-Alone CAN Controller with SPI Interface

Description

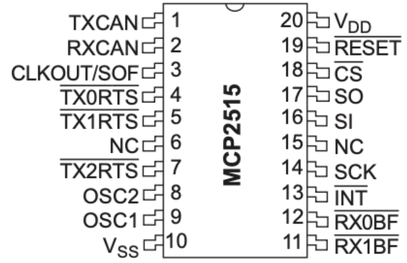
Microchip Technology's MCP2515 is a stand-alone Controller Area Network (CAN) controller that implements the CAN specification, Version 2.0B. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCU's overhead. The MCP2515 interfaces with microcontrollers (MCUs) via an industry standard Serial Peripheral Interface (SPI).

Package Types

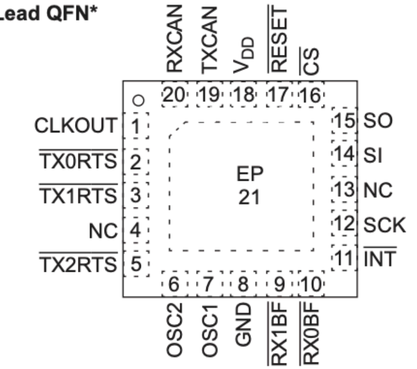
18-Lead PDIP/SOIC



20-Lead TSSOP



20-Lead QFN*

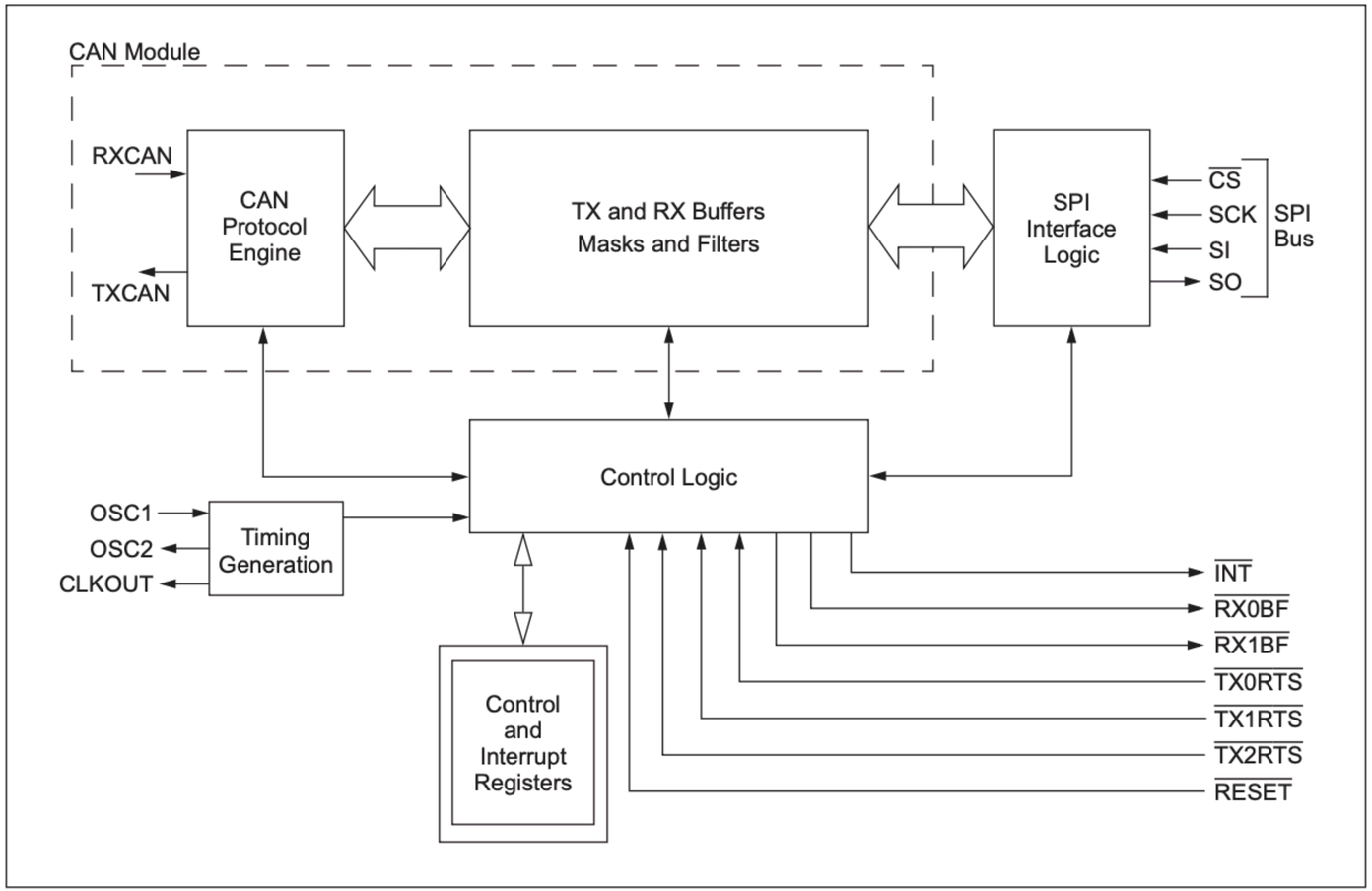


* Includes Exposed Thermal Pad (EP); see Table 1-1.

SPI Interface

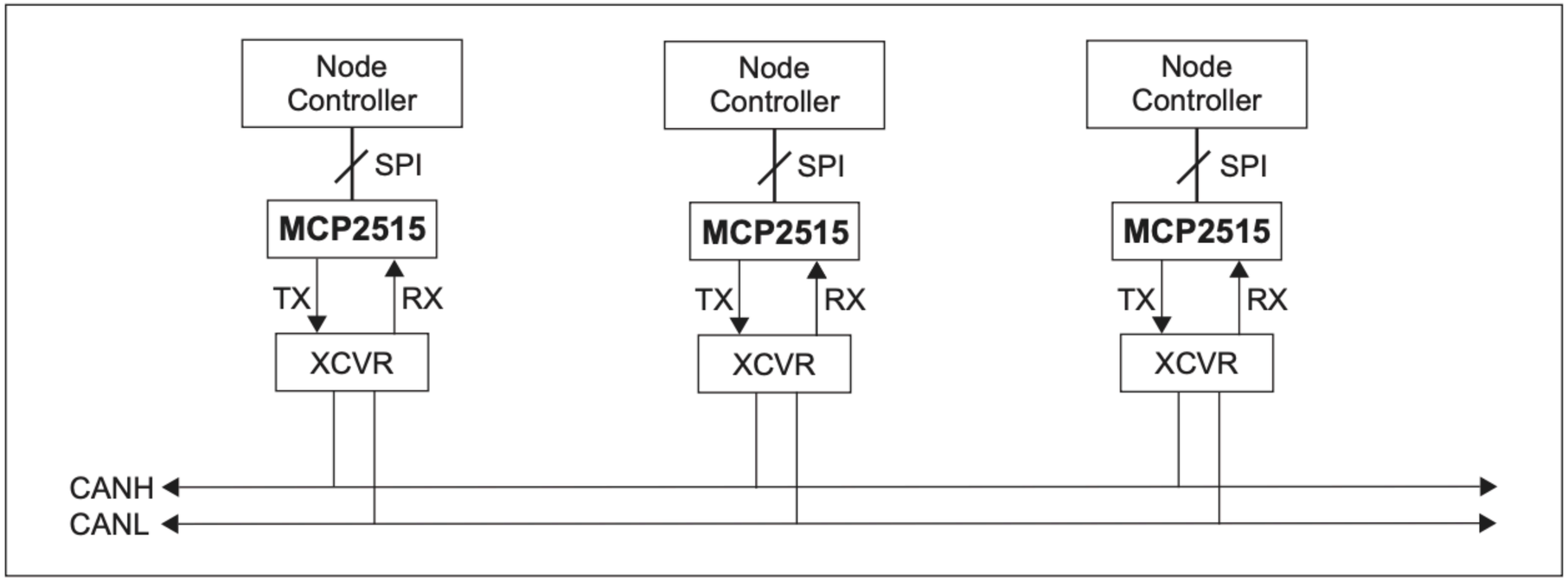
Microchip MCP2515 Block Diagram

FIGURE 1-1: BLOCK DIAGRAM

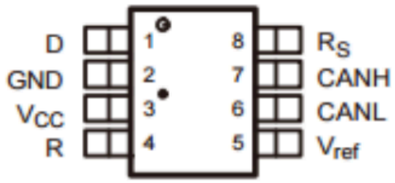


Microchip MCP2515 Block Diagram

FIGURE 1-2: EXAMPLE SYSTEM IMPLEMENTATION



SN65HVD230 CAN Transceiver

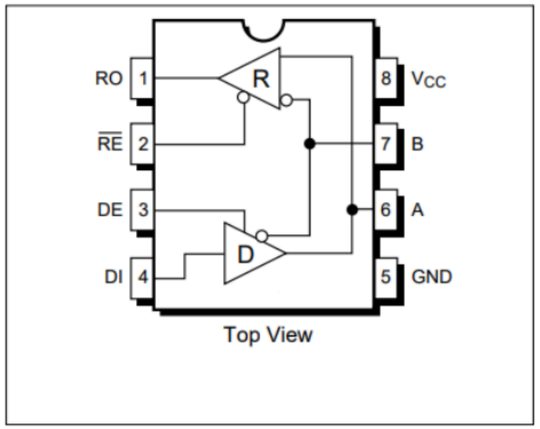


SN65HVD230 引脚

（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	V_{CC}	电源线 Supply voltage
4	R	接收输出 Receiver output
5	V_{ref}	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

SP3481 RS485 Transceiver



SP3481/SP3485
Pinout (Top View)

PIN FUNCTION

Pin 1 – RO – Receiver Output.

Pin 2 – $\overline{RE}$ – Receiver Output Enable Active LOW.

Pin 3 – DE – Driver Output Enable Active HIGH.

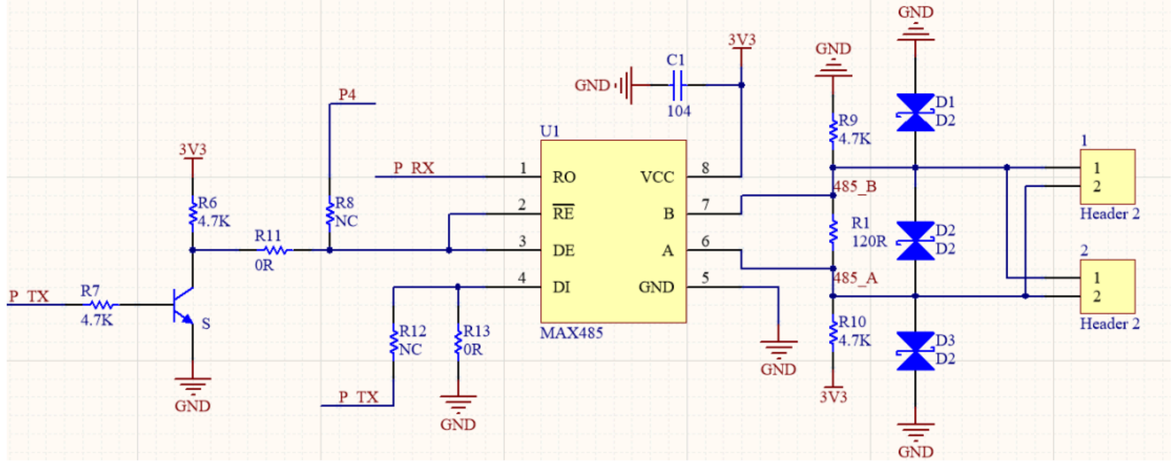
Pin 4 – DI – Driver Input.

Pin 5 – GND – Ground Connection.

Pin 6 – A – Driver Output/Receiver Input Non-inverting.

Pin 7 – B – Driver Output/Receiver Input Inverting.

Pin 8 – V_{CC}



Library Installation Overview

- Required Libraries
  - wiringPi
  - bcm2835
- Python Libraries
  - sudo apt-get install python-pip
  - sudo pip install python-can

wiringPi Library - Intro

- GPIO library written for BCM2835,36,37 SOC devices
- Designed for those familiar with Arduino
- Both C Library and command-line utility (gpio)

NOTE: Project official deprecated HOWEVER still in wide use!
(Alternative projects: pigpio)

wiringPi Library - Key Features

- Simply coding syntax similar to Arduino
- Comprehensive GPIO Control
  - Read/Write, Read Analog, PWM (Pulse Width Modulation)
- Support for popular I/O expanders (MCP23xxx, etc.)
- Command-line Utility
  - gpio
- ISR Handling
  - ISR = Interrupt Service Routine
- Networking
  - Allows RPI to send data over the network

wiringPi Library - Basic Functions

- wiringPiSetup() - must call this first
- pinMode() - INPUT, OUTPUT, PWM_OUTPUT, GPIO_CLOCK
- pullUpDnControl()
- pwmWrite()
- digitalWrite(), digitalRead()
- analogRead(), analogWrite()

wiringPi Library - To Install - 1st Try

```
$ sudo apt-get install wiringpi
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package wiringpi is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'wiringpi' has no installation candidate
```

wiringPi Library - To Install - 2nd Try

```
$ git clone https://github.com/WiringPi/WiringPi
Cloning into 'WiringPi'...
remote: Enumerating objects: 1733, done.
remote: Counting objects: 100% (615/615), done.
remote: Compressing objects: 100% (114/114), done.
remote: Total 1733 (delta 555), reused 502 (delta 501), pack-reused 1118
Receiving objects: 100% (1733/1733), 803.57 KiB | 3.55 MiB/s, done.
Resolving deltas: 100% (1188/1188), done.

$ cd WiringPi/

$ ls
build           debian           devLib    gpio     newVersion  pins       update   version.h  wiringPiD
COPYING.LESSER  debian-template  examples  INSTALL  People      README.md  VERSION  wiringPi

$ find . | wc -l
264
```

wiringPi Library - find .

Examples

Library C Source Code

gpio CLI Source Code

Debian Packaging

Networking Daemon

```
find
.
./examples
./examples/serialTest.c
./examples/blink.rtb
./examples/blink.sh
. . .
./wiringPi
./wiringPi/mcp3422.c
./wiringPi/mcp23s08.h
./wiringPi/wiringPiI2C.h
./wiringPi/mcp23016.h
./wiringPi/wiringSerial.c
./wiringPi/wpiExtensions.c
. . .
./gpio/gpio.c
./gpio/Makefile
./gpio/readall.c
. . .
./debian-template
./debian-template/wiringPi
./debian-template/wiringPi/DEBIAN
./debian-template/wiringPi/DEBIAN/postrm
./debian-template/wiringPi/DEBIAN/control
./debian-template/wiringPi/DEBIAN/postinst
. . .
./wiringPiD
./wiringPiD/daemonise.c
./wiringPiD/network.c
./wiringPiD/runRemote.h
./wiringPiD/runRemote.c
./wiringPiD/Makefile
. . .
```

This is a great template for your own projects

wiringPi Library - README.md

```
Ports
-----

wiringPi has been wrapped for multiple languages:

* Node - https://github.com/WiringPi/WiringPi-Node
* Perl - https://github.com/WiringPi/WiringPi-Perl
* PHP - https://github.com/WiringPi/WiringPi-PHP
* Python - https://github.com/WiringPi/WiringPi-Python
* Ruby - https://github.com/WiringPi/WiringPi-Ruby
```

wiringPi Library - build - Part 1

```
#!/bin/sh -e

# build
#       Simple wiringPi build and install script
#
#       Copyright (c) 2012-2015 Gordon Henderson
#################################################################################
# This file is part of wiringPi:
#       A "wiring" library for the Raspberry Pi
#
#    wiringPi is free software: you can redistribute it and/or modify
#    it under the terms of the GNU Lesser General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
```

wiringPi Library - build - Part 2

```
check_make_ok() {
  if [ $? != 0 ]; then
    echo ""
    echo "Make Failed..."
    echo "Please check the messages and fix any problems. If you're still stuck,"
    echo "then raise a GitHub issue with the output and as many details as you can"
    echo "  https://github.com/WiringPi/WiringPi/issues"
    echo ""
    exit 1
  fi
}
```

wiringPi Library - build - Part 3

```
if [ x$1 = "xclean" ]; then
  cd wiringPi
  echo -n "wiringPi:   "; make clean
  cd ../devLib
  echo -n "DevLib:     "; make clean
. . .

if [ x$1 = "xuninstall" ]; then
  cd wiringPi
  echo -n "wiringPi: " ; $sudo make uninstall
  cd ../devLib
  echo -n "DevLib:   " ; $sudo make uninstall
  cd ../gpio
  echo -n "gpio:     " ; $sudo make uninstall
  exit
fi
. . .
# Only if you know what you're doing!

if [ x$1 = "xdebian" ]; then
  here=`pwd`
  deb_destdir=${here}/debian-template/wiringPi
  cd debian-template/wiringPi
. .
```

wiringPi Library - build - Part 4

WiringPi Library Build

```
if [ x$1 != "x" ]; then
  echo "Usage: $0 [clean | uninstall]"
  exit 1
fi

  echo "wiringPi Build script"
  echo "====================="
  echo

  hardware=`fgrep Hardware /proc/cpuinfo | head -1 | awk '{ print $3 }'`

echo
  echo "WiringPi Library"
  cd wiringPi
  $sudo make uninstall
  if [ x$1 = "xstatic" ]; then
    make -j5 static
    check_make_ok
    $sudo make install-static
  else
    make -j5
    check_make_ok
    $sudo make install
  fi
  check_make_ok
```

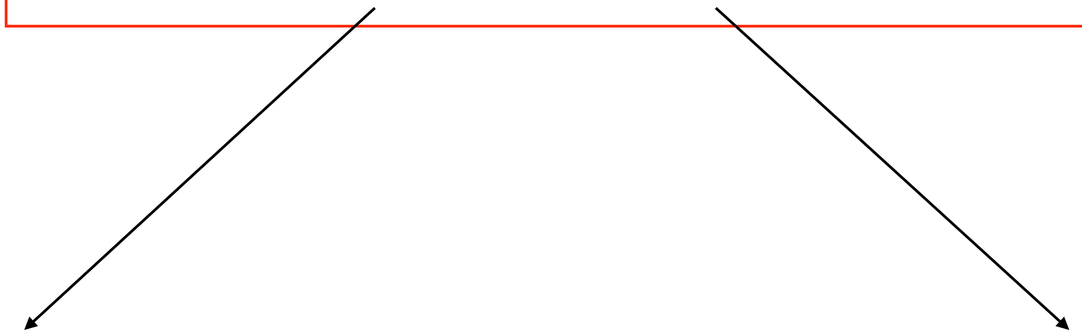
wiringPi Library - build - Part 4
Wiring Devices Library

```
echo
echo "WiringPi Devices Library"
cd ../devLib
$sudo make uninstall
if [ x$1 = "xstatic" ]; then
  make -j5 static
  check_make_ok
  $sudo make install-static
else
  make -j5
  check_make_ok
  $sudo make install
fi
check_make_ok
```

wiringPi Library - build - Part 5a
GPIO Utility

```
echo
echo "GPIO Utility"
cd ../gpio
make -j5
check_make_ok
$sudo make install
check_make_ok
```

```
ls gpio/
COPYING.LESSER  gpio.1  gpio.c  Makefile  pintest  readall.c  test.sh
```



```
head gpio/gpio.1
.TH GPIO 1 "March 2018" wiringPi "Command-Line access to Raspberry Pi's GPIO"

.SH NAME
gpio \- Command-line access to Raspberry Pi's GPIO

.SH SYNOPSIS
.B gpio
.B \-v
.PP
.B gpio
```

See Makefile on next slide

wiringPi Library - build - Part 5b

GPIO Utility Makefile

```
DESTDIR?=/usr
PREFIX?=/local

ifneq ($V,1)
Q ?= @
endif

#DEBUG  = -g -O0
DEBUG = -O2
CC  ?= gcc
INCLUDE = -I$(DESTDIR)$(PREFIX)/include
CFLAGS  = $(DEBUG) -Wall -Wextra $(INCLUDE) -Winline -pipe $(EXTRA_CFLAGS)

LDFLAGS = -L$(DESTDIR)$(PREFIX)/lib
LIBS    = -lwiringPi -lwiringPiDev -lpthread -lrt -lm -lcrypt
```

wiringPi Library - build - Part 5c

GPIO Utility Makefile

```
# May not need to  alter anything below this line
###############################################################################

SRC = gpio.c readall.c

OBJ = $(SRC:.c=.o)

all:   gpio

version.h:  ../VERSION
  $Q echo Need to run newVersion above.

gpio:  $(OBJ)
  $Q echo [Link]
  $Q $(CC) -o $@ $(OBJ) $(LDFLAGS) $(LIBS)

.c.o:
  $Q echo [Compile] $<
  $Q $(CC) -c $(CFLAGS) $< -o $@
```

wiringPi Library - build - Part 5d

GPIO Utility Makefile

```
.PHONY: clean
clean:
  $Q echo "[Clean]"
  $Q rm -f $(OBJ) gpio *~ core tags *.bak

.PHONY: tags
tags:  $(SRC)
  $Q echo [ctags]
  $Q ctags $(SRC)

.PHONY: install
install: gpio
  $Q echo "[Install]"
  $Q mkdir -p  $(DESTDIR)$(PREFIX)/bin
  $Q cp gpio   $(DESTDIR)$(PREFIX)/bin
ifneq ($(WIRINGPI_SUID),0)
  $Q chown root.root$(DESTDIR)$(PREFIX)/bin/gpio
  $Q chmod 4755  $(DESTDIR)$(PREFIX)/bin/gpio
endif
  $Q mkdir -p  $(DESTDIR)$(PREFIX)/share/man/man1
  $Q cp gpio.1   $(DESTDIR)$(PREFIX)/share/man/man1
```

wiringPi Library - build - Part 6

Commented Out By Default: Daemon and Examples

```
# echo
# echo "wiringPi Daemon"
# cd ../wiringPiD
# make -j5
# check_make_ok
# $sudo make install
# check_make_ok

# echo
# echo "Examples"
# cd ../examples
# make
# cd ..
```

wiringPi Library - build - Part 7

Message About Using -lwiringPi

```
echo
echo All Done.
echo ""
echo "NOTE: To compile programs with wiringPi, you need to add:"
echo "    -lwiringPi"
echo "  to your compile line(s) To use the Gertboard, MaxDetect, etc."
echo "  code (the devLib), you need to also add:"
echo "    -lwiringPiDev"
echo "  to your compile line(s)."
echo ""
```

C Libraries - WiringPi

C

Install WiringPi Library

```
#Open the Raspberry Pi terminal and run the following command
cd
sudo apt-get install wiringpi
#For Raspberry Pi systems after May 2019 (earlier than that can be executed without), an upgrade may b
e required:
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# Run gpio -v and version 2.52 will appear, if it doesn't it means there was an installation error

# Bullseye branch system using the following command:
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
. /build
gpio -v
# Run gpio -v and version 2.70 will appear, if it doesn't it means there was an installation error
```

C Libraries - Building WiringPi

```
./build
wiringPi Build script
=====================


WiringPi Library
[UnInstall]
[Compile] wiringPi.c
[Compile] wiringSerial.c
. . .

WiringPi Devices Library
[UnInstall]
[Compile] ds1302.c
. . .

GPIO Utility
[Compile] gpio.c
[Compile] readall.c
[Link]
[Install]

All Done.

NOTE: To compile programs with wiringPi, you need to add:
    -lwiringPi
  to your compile line(s) To use the Gertboard, MaxDetect, etc.
  code (the devLib), you need to also add:
    -lwiringPiDev
  to your compile line(s).
```

Running gpio Command

```
$ gpio -v
gpio version: 2.70
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Pi 4B, Revision: 04, Memory: 2048MB, Maker: Sony
  * Device tree is enabled.
  *--> Raspberry Pi 4 Model B Rev 1.4
  * This Raspberry Pi supports user-level GPIO access.
```

C Libraries - bcm2835

Install bcm2835

```
#Open the Raspberry Pi terminal and run the following command
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.71.tar.gz
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/mikem/bcm2835/
```

Python Libraries

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip install RPi.GPIO
sudo apt-get install python3-smbus
```

/usr/local/include

```
$ ls /usr/local/include/
ads1115.h    htu21d.h     mcp23016.h     mcp3002.h  piGlow.h       scrollPhat.h  wiringPiI2C.h
bmp180.h     lcd128x64.h  mcp23016reg.h  mcp3004.h  pigpiod_if2.h  sn3218.h      wiringPiSPI.h
drcNet.h     lcd.h        mcp23017.h     mcp3422.h  pigpiod_if.h   softPwm.h     wiringSerial.h
drcSerial.h  max31855.h   mcp23s08.h     mcp4802.h  pigpio.h       softServo.h   wiringShift.h
ds1302.h     max5322.h    mcp23s17.h     pcf8574.h  piNes.h        softTone.h    wpiExtensions.h
ds18b20.h    maxdetect.h  mcp23x0817.h   pcf8591.h  pseudoPins.h   sr595.h
gertboard.h  mcp23008.h   mcp23x08.h     piFace.h   rht03.h        wiringPi.h
```

/usr/local/include/wiringPi.h

```
#ifndef  __WIRING_PI_H__
#define  __WIRING_PI_H__
. . .
// Pin modes

#define  INPUT        0
#define  OUTPUT       1
#define  PWM_OUTPUT     2
. . .
extern const char *piModelNames    [21] ;
extern const char *piRevisionNames [16] ;
extern const char *piMakerNames    [16] ;
extern const int   piMemorySize    [ 8] ;
. . .

// Function prototypes
// c++ wrappers thanks to a comment by Nick Lott
// (and others on the Raspberry Pi forums)

#ifdef __cplusplus
extern "C" {
#endif

. . .
extern void wiringPiVersion (int *major, int *minor) ;
extern int  wiringPiSetup       (void) ;
. . .
extern          void pinMode             (int pin, int mode) ;
. .

extern          void digitalWrite        (int pin, int value) ;
. .
```

gpio-test.c - Source Code

```
#include <stdio.h>
#include <wiringPi.h>
#include <unistd.h>

int main() {
   puts("Test of wiringPi");

   wiringPiSetup();

   int major = 42;
   int minor = 42;;
   wiringPiVersion(&major, &minor);
   printf("major:minor: %d:%d\n", major, minor);

   int gpio21 = 21;
   pinMode(gpio21, OUTPUT);

   while (1) {
      puts("LED ON");
      digitalWrite(gpio21, 1) ;
      sleep(10);
      puts("LED OFF");
      digitalWrite(gpio21, 0) ;
      sleep(10);
   }

   return 0;
}
```

gpio-test.c - Building and Running

```
$ gcc -Wall -o gpio-test gpio-test.c
/usr/bin/ld: /tmp/ccbtxOZU.o: in function `main':
gpio-test.c:(.text+0x14): undefined reference to `wiringPiSetup'
/usr/bin/ld: gpio-test.c:(.text+0x38): undefined reference to `wiringPiVersion'
/usr/bin/ld: gpio-test.c:(.text+0x60): undefined reference to `pinMode'
/usr/bin/ld: gpio-test.c:(.text+0x6c): undefined reference to `digitalWrite'

$ gcc -Wall -o gpio-test gpio-test.c -lwiringPi

$ ldd gpio-test
   linux-vdso.so.1 (0xbefe2000)
   /usr/lib/arm-linux-gnueabihf/libarmmem-${PLATFORM}.so => /usr/lib/arm-linux-gnueabihf/libarmmem-v7l.so (0xb6ec4000)
   libwiringPi.so => /usr/local/lib/libwiringPi.so (0xb6e91000)
   libc.so.6 => /lib/arm-linux-gnueabihf/libc.so.6 (0xb6d3d000)
   libm.so.6 => /lib/arm-linux-gnueabihf/libm.so.6 (0xb6cce000)
   libpthread.so.0 => /lib/arm-linux-gnueabihf/libpthread.so.0 (0xb6ca2000)
   librt.so.1 => /lib/arm-linux-gnueabihf/librt.so.1 (0xb6c8a000)
   libcrypt.so.1 => /lib/arm-linux-gnueabihf/libcrypt.so.1 (0xb6c3e000)
   /lib/ld-linux-armhf.so.3 (0xb6ed9000)


./gpio-test
Test of wiringPi
major:minor: 2:70
LED ON
LED OFF
```

Now to test SPI (Serial Peripheral Interface)

Edit /boot/config.txt

> Note:
>
> There are two versions. The only difference betweem these version are the crystal. The old one use 8M and the line added to config.txt file should be:

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

The new one use 12M crystal and the line should be

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

Reboot

```
$ sudo reboot
Connection to 192.168.4.34 closed by remote host.
Connection to 192.168.4.34 closed.
bash-3.2$
```

```
$ dmesg | grep CAN
[    7.562495] CAN device driver interface

$ dmesg | grep spi0
[    7.601713] mcp251x spi0.0 can0: MCP2515 successfully initialized.
```

Bring up CAN Interface

```
$ ifconfig -a
can0: flags=128<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=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e4:5f:01:7d:fc:ea  txqueuelen 1000  (Ethernet)
        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


$ sudo ip link set can0 up type can bitrate 500000

$ ifconfig can0
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
```

To Automatically Bring Up

To automatically bring up the interface on boot, edit your /etc/network/interfaces file and add the following:

```
auto can0
iface can0 inet manual
    pre-up /sbin/ip link set can0 type can bitrate 500000 triple-
sampling on restart-ms 100
    up /sbin/ifconfig can0 up
    down /sbin/ifconfig can0 down
```

sudo apt install can-utils

```
$ sudo apt install can-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libirs-export161 libisccfg-export163 policycoreutils selinux-utils
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  can-utils
```

dpkg -L can-utils - Part 1

CAN-utils is a collection of extremely useful debugging tools using the SocketCAN interface. It includes applications such as:

- **candump** – Dump can packets – display, filter and log to disk.
- **canplayer** – Replay CAN log files.
- **cansend** – Send a single frame.
- **cangen** – Generate random traffic.
- **canbusload** – display the current CAN bus utilisation.

CAN-utils source can be obtained from the GitHub repository: **https://github.com/linux-can/can-utils**

```
$ dpkg -L can-utils
/.
/usr
/usr/bin
/usr/bin/asc2log
/usr/bin/bcmserver
/usr/bin/can-calc-bit-timing
/usr/bin/canbusload
/usr/bin/candump
/usr/bin/canfdtest
/usr/bin/cangen
/usr/bin/cangw
/usr/bin/canlogserver
/usr/bin/canplayer
/usr/bin/cansend
/usr/bin/cansequence
/usr/bin/cansniffer
```

dpkg -L can-utils - Part 1

Virtual CAN interface

While you can develop and test code on real hardware, Linux also provides a virtual CAN interface driver (vcan.ko) that acts like a loopback port. When used with **CAN-Utils**, it makes development and testing a breeze. No more trying to work out if your issue is hardware or software related.

To create a virtual CAN network interface called vcan0:

```
sudo ip link add dev vcan0 type vcan
sudo ifconfig vcan0 up
```

can_receive.c

can_receive.c - Part 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>

int main()
{
    int ret;
    int s, nbytes;
    struct sockaddr_can addr;
    struct ifreq ifr;
    struct can_frame frame;

    memset(&frame, 0, sizeof(struct can_frame));

    system("sudo ip link set can0 type can bitrate 100000");
    system("sudo ifconfig can0 up");
    printf("this is a can receive demo\r\n");
```

can_receive.c - Part 2

```
//1.Create socket
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
if (s < 0) {
    perror("socket PF_CAN failed");
    return 1;
}

//2.Specify can0 device
strcpy(ifr.ifr_name, "can0");
ret = ioctl(s, SIOCGIFINDEX, &ifr);
if (ret < 0) {
    perror("ioctl failed");
    return 1;
}

//3.Bind the socket to can0
addr.can_family = PF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));
if (ret < 0) {
    perror("bind failed");
    return 1;
}
```

can_receive.c - Part 3

```
    //4.Define receive rules
    struct can_filter rfilter[1];
    rfilter[0].can_id = 0x123;
    rfilter[0].can_mask = CAN_SFF_MASK;
    setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));

    //5.Receive data and exit
    while(1) {
        nbytes = read(s, &frame, sizeof(frame));
        if(nbytes > 0) {
            printf("can_id = 0x%X\r\ncan_dlc = %d \r\n", frame.can_id, frame.can_dlc);
            int i = 0;
            for(i = 0; i < 8; i++)
                printf("data[%d] = %d\r\n", i, frame.data[i]);
            break;
        }
    }

    //6.Close the socket and can0
    close(s);
    system("sudo ifconfig can0 down");

    return 0;
}
```

can_receive.c - Makefile

```
CC = gcc
CFLAGS = -Wall -g -O0

can_receive:can_receive.c
	$(CC) $(CFLAGS) -o $@ $^

clean:
	$(RM) can_receive .*.sw?
```

can_receive.c - Running Code

```
$ make
gcc -Wall -g -O0 -o can_receive can_receive.c

$ ./can_receive
this is a can receive demo
```

can_send

can_send.c - Part 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>

int main()
{
    int ret;
    int s, nbytes;
    struct sockaddr_can addr;
    struct ifreq ifr;
    struct can_frame frame;
    memset(&frame, 0, sizeof(struct can_frame));

    system("sudo ip link set can0 type can bitrate 100000");
    system("sudo ifconfig can0 up");
    printf("this is a can send demo\r\n");
```

can_send.c - Part 2

```
//1.Create socket
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
if (s < 0) {
    perror("socket PF_CAN failed");
    return 1;
}

//2.Specify can0 device
strcpy(ifr.ifr_name, "can0");
ret = ioctl(s, SIOCGIFINDEX, &ifr);
if (ret < 0) {
    perror("ioctl failed");
    return 1;
}

//3.Bind the socket to can0
addr.can_family = AF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));
if (ret < 0) {
    perror("bind failed");
    return 1;
}
```

can_send.c - Part 3

```
//4.Disable filtering rules, do not receive packets, only send
setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, NULL, 0);

//5.Set send data
frame.can_id = 0x123;
frame.can_dlc = 8;
frame.data[0] = 1;
frame.data[1] = 2;
frame.data[2] = 3;
frame.data[3] = 4;
frame.data[4] = 5;
frame.data[5] = 6;
frame.data[6] = 7;
frame.data[7] = 8;

printf("can_id  = 0x%X\r\n", frame.can_id);
printf("can_dlc = %d\r\n", frame.can_dlc);
int i = 0;
for(i = 0; i < 8; i++)
    printf("data[%d] = %d\r\n", i, frame.data[i]);
```

can_send.c - Part 4

```
    //6.Send message
    nbytes = write(s, &frame, sizeof(frame));
    if(nbytes != sizeof(frame)) {
        printf("Send Error frame[0]!\r\n");
        system("sudo ifconfig can0 down");
    }

    //7.Close the socket and can0
    close(s);
    system("sudo ifconfig can0 down");
    return 0;
}
```