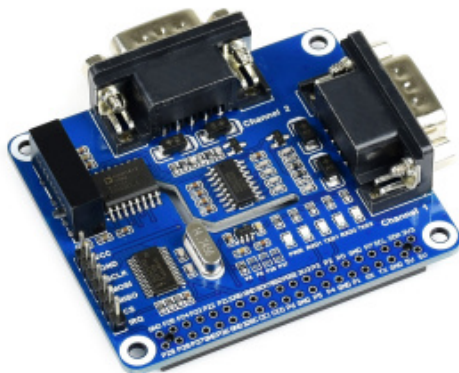


2-CH RS232 HAT

来自Waveshare Wiki

跳转至: [导航](#)、[搜索](#)

2-CH RS232 HAT



板载接口

RPi

RS232

说明

产品概述

我是专为树莓派设计的双通道隔离型RS232扩展板，采用SC16IS752+SP3232方案，内置电源隔离、ADI磁耦隔离和TVS等保护电路。

我操作简单，通过SPI接口即可控制两路RS232通信，具有通信速度快、稳定、可靠、安全等特点，可适用于工业自动化等领域。

特点

- 基于Raspberry Pi 40pin GPIO接口，适用于Raspberry Pi系列主板
- 采用SC16IS752与SP3232双芯片组合方案，可实现SPI转RS232，通信速率高达921600bps
- 板载TVS(瞬态电压抑制管)，可有效抑制电路中的浪涌电压和瞬态尖峰电压，防雷防静电
- 板载电源和串口收发指示灯，方便查看模块电源和通信状态
- 引出SPI控制接口，方便接入Arduino等主控板
- 提供完善的配套资料手册(C与python示例程序和用户手册等)

产品参数

- 串口扩展芯片：SC16IS752
- RS232收发器：SP3232
- 通信接口：SPI

- 通信速率：300~921600 bps
- 工作电压：3.3/5V
- 产品尺寸：65mm×56.5mm
- 过孔直径：3.0mm

接口说明

- 引脚功能

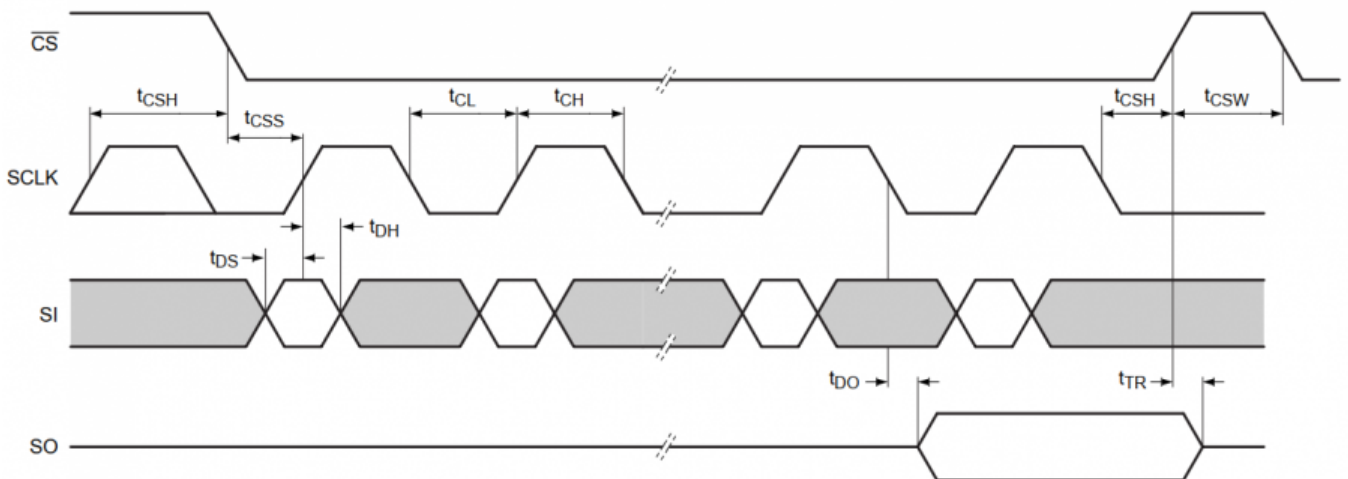
| 引脚号 | 标识 | 管脚描述 |
|-----|------|------------|
| 1 | VCC | 3.3V/5V电源正 |
| 2 | GND | 电源地 |
| 3 | SCLK | SPI时钟输入 |
| 4 | MOSI | SPI数据输入 |
| 5 | MISO | SPI数据输出 |
| 6 | CS | SPI片选信号 |
| 7 | IRQ | 中断输出 |

硬件说明

控制器

本产品采用SC16IS752作为控制芯片，SC16IS752是双通道高性能的UART扩展芯片，支持SPI和I2C两种接口通信，本模块使用SPI接口。板载电源隔离、ADI磁耦隔离、板载TVS(瞬态电压抑制管)、自恢复保险丝和保护二极管和自动收发转换电路。可有效抑制电路中的浪涌电压和瞬态尖峰电压，防雷防静电，防过流过压，提高抗冲击能力，可进行信号隔离，具有可靠性高、抗干扰强、功耗低等优点。

通信协议



CS：从机片选，当CS为低电平的时候，芯片使能

SCLK：SPI通信时钟

MOSI/SI：SPI通信主机发送，从机接收

MISO/SO：SPI通信主机接收，从机发送

时序：CPHL=0，CPOL=0（SPI0）

【备注】具体关于SPI通信的相关信息，可以自行网上搜索资料了解

树莓派使用

硬件连接

| 功能引脚 | 树莓派接口 (BCM) | 描述 |
|------|----------------|------------|
| VCC | 5V | 3.3V/5V电源正 |
| GND | GND | 电源地 |
| SCLK | P21(SPI1 SCLK) | SPI时钟输入 |
| MOSI | P20(SPI1 MOSI) | SPI数据输入 |
| MISO | P19(SPI1 MISO) | SPI数据输出 |
| CS | P18(SPI1 CS) | SPI片选信号 |
| IRQ | P24 | 中断输出 |

添加驱动

- 在终端执行:

```
sudo nano /boot/config.txt
#加入如下, int_pin根据实际焊接方式设置:
dtoverlay=sc16is752-spi1,int_pin=24
#重启设备
sudo reboot
```

重启后, SC16IS752的驱动会加载到系统内核中, 此时可以运行ls /dev查看一下 将会多出如下设备:

```
pi@raspberrypi:~$ ls /dev/
autofs          gpiochip3      mapper         ram11          shm            tty19          tty34          tty5            tty8            vcs5
block           gpiomem       mem            ram12          snd            tty2           tty35          tty50          tty9            vcs6
btrfs-control  hwrng         memory_bandwidth ram13          stderr        tty20          tty36          tty51          ttyAMA0        vcs7
bus             i2c-1        mmcblk0        ram14          stdin         tty21          tty37          tty52          ttyprintk      vcsa
cachefiles     initctl       mmcblk0p1      ram15          stdout        tty22          tty38          tty53          ttySC0         vcsa1
char           input         mmcblk0p2      ram2           tty           tty23          tty39          tty54          ttySC1         vcsa2
console        kmsg         queue         ram3           tty0          tty24          tty4           tty55          urandom        vcsa3
cpu_dma_latency log           net            ram4           tty1          tty25          tty40          tty56          uinput         vcsa4
cuse           loop0         network_latency ram5           tty10         tty26          tty41          tty57          urandom        vcsa5
disk           loop1         network_throughput ram6           tty11         tty27          tty42          tty58          vchiq          vcsa6
fb0            loop2         null           ram7           tty12         tty28          tty43          tty59          vcio           vcsa7
fd             loop3         ppp            ram8           tty13         tty29          tty44          tty6           vc-mem         vcsm
full           loop4         ptmx           ram9           tty14         tty3           tty45          tty60          vcs            vhci
fuse           loop5         pts            random         tty15         tty30          tty46          tty61          vcs1           watchdog
gpiochip0      loop6         ram0           raw            tty16         tty31          tty47          tty62          vcs2           watchdog0
gpiochip1      loop7         ram1           rfkill         tty17         tty32          tty48          tty63          vcs3           zero
gpiochip2      loop-control ram10          serial         tty18         tty33          tty49          tty7           vcs4
```

在2020-05-27的树莓派系统中gpiochip3没有了变成了gpiochip2

安装库

请根据实际情况安装库, 如果你编程为C, 安装wiringPi即可; 如果是python2, 安装python2库; 如果是python3, 安装python3库;

- 安装wiringPi

```
sudo apt-get install wiringpi
#对于树莓派4B可能需要进行升级;
cd /tmp
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
# 运行 gpio -v 会出现2.52版本, 如果没有出现说明安装出错
```

■ 安装python2库

```
sudo apt-get update
sudo apt-get install python-pip
sudo pip install RPi.GPIO
sudo apt-get install python-serial
```

■ 安装python3库

```
sudo apt-get update
sudo apt-get install python3-pip
sudo pip3 install RPi.GPIO
sudo apt-get install python3-serial
```

测试

■ 下载并运行测试例程

```
sudo apt-get install p7zip-full
wget http://www.waveshare.net/w/upload/3/35/2-CH_RS232_HAT_Code.7z
7z x 2-CH_RS232_HAT_Code.7z -o./2-CH_RS232_HAT
sudo chmod 777 -R 2-CH_RS232_HAT
cd 2-CH_RS232_HAT/2-CH_RS232_HAT_Code/
```

■ 另外也可以下载我们Github上面的工程:

```
sudo git clone https://github.com/waveshare/2-CH-RS232-HAT
cd 2-CH_RS232_HAT_Code/
```

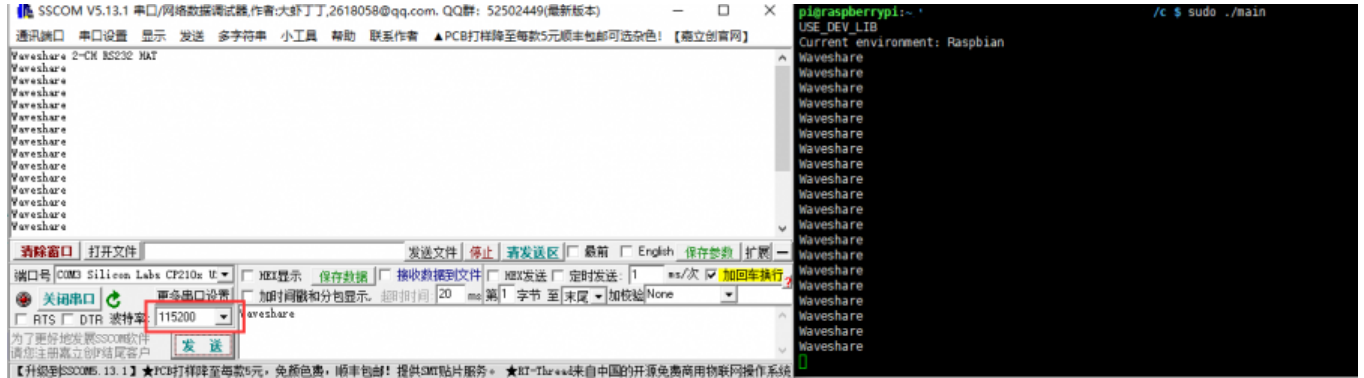
*注意目录稍微有些不一样

■ C 程序

```
cd c
make clean
make
sudo ./main
```

预期效果：

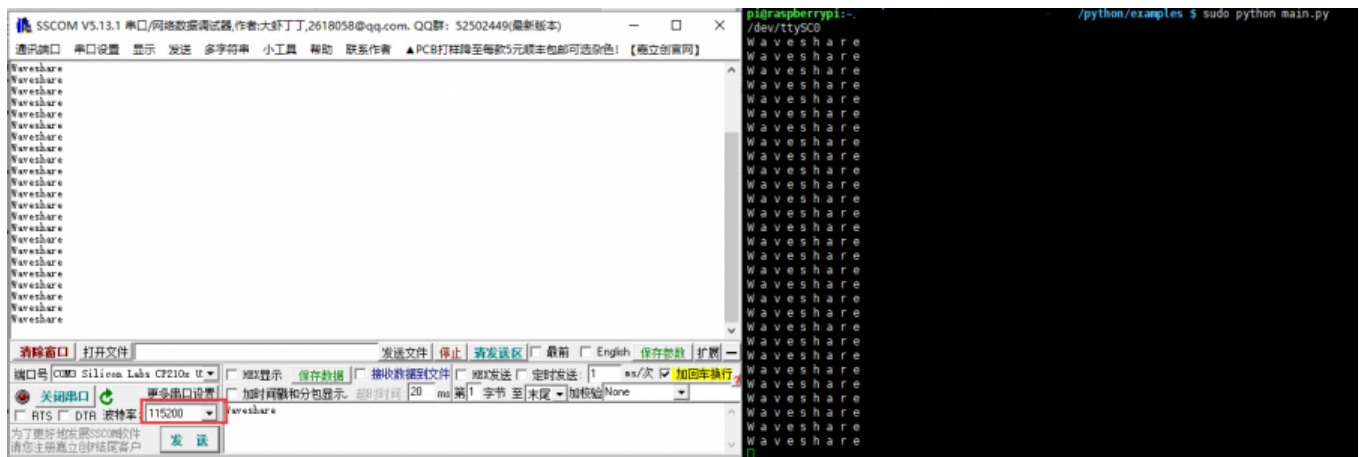
通道1通过转串口线连接电脑，打开电脑串口设置波特率115200，电脑发送任何数据，模块将数据返回（如果结尾不加回车换行，可能会导致终端无显示）。



■ PYTHON 程序

```
cd python
cd examples
sudo python main.py
```

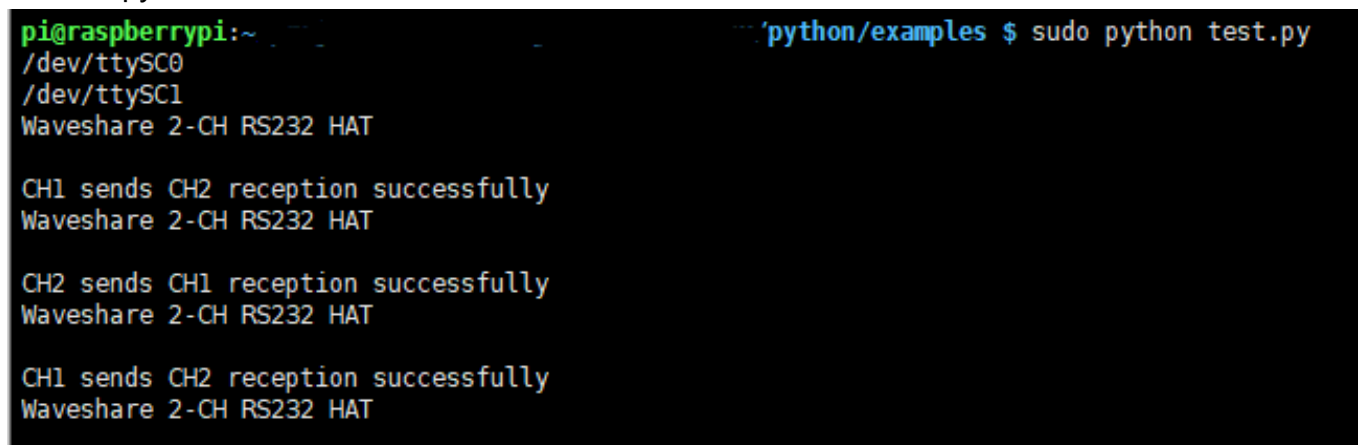
■ 通道1通过转串口线连接电脑，打开电脑串口设置波特率115200，电脑发送任何数据，模块将数据返回（如果结尾不加回车换行，可能会导致终端无显示）。



此外提供一个测试程序：

```
sudo python test.py
```

运行test.py，通过母对母串口交叉线把通道1与通道2连接



资料

文档

- 原理图

程序

- 示例程序
- Github

数据手册

- SC16IS752

FAQ

问题：232通信不正常？

1. 检查树莓派的串口通信是否开启了流控；
2. 确定232的线连接正确；
3. 可以先使用USB to 232设备与2-CH RS232 HAT通信，保证树莓派的设置没有问题；
4. 检查串口通信参数的奇数偶数位校验和波特率的设置。

问题：树莓派装了ubuntu系统可以使用么，为什么在boot文件夹找不到config.txt文件？

1. 树莓派装了主流的ubuntu系统可以使用；
2. Ubuntu 的config.txt文件通常在/boot/firmware 文件夹
3. 或者将树莓派的SD卡通过读卡器在电脑下（或者其它能识别SD卡的主机）读取和更改config.txt文件。