RS485 CAN Shield

用户手册

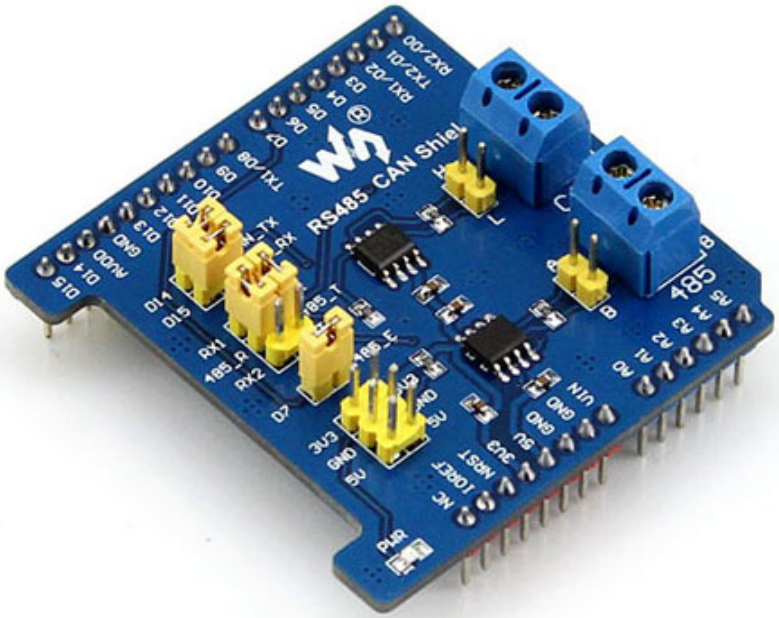
产品概述

RS485 CAN Shield 是微雪电子为 NUCLEO/XNUCLEO 开发的一款的带 RS485 和 CAN 通信功能的扩展板，具备 RS485、CAN 通信功能。

特点：

- 基于 Arduino 标准接口设计，兼容 UNO、Leonardo、NUCLEO、XNUCLEO 开发板
- 具备 RS485 功能，收发器为 MAX3485，3.3V 供电
- 具备 CAN 功能，收发器为 SN65HVD230，3.3V 供电

注意：① 使用 3.3V 供电； ② UNO、Leonardo 等 Arduino 板由于没有 CAN 硬件接口，需要通过软件模拟 CAN 总线时序才能使用。



版本：V1.4，日期：2015 年 5 月 19 日

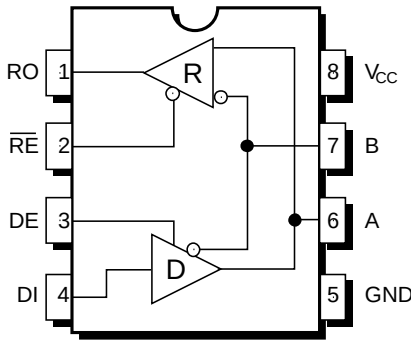
目录

1. 硬件说明

1.1. 芯片引脚功能概述

1.1.1. MAX3485

MAX3485 接口芯片是 Maxim 公司的一种 RS－485 驱动芯片。用于 RS-485 通信的低功耗收发器。采用单一电源+3.3 V 工作，采用半双工通讯方式。RO 和 DI 端分别为接收器的输出和驱动器的输入端；$\overline{RE}$和 DE 端分别为接收和发送的使能端，当$\overline{RE}$为逻辑 0 时，器件处于接收状态；当 DE 为逻辑 1 时，器件处于发送状态；A 端和 B 端分别为接收和发送的差分信号端，当 A-B>+0.2V 时，RO 输出逻辑 1；当 A-B<-0.2V 时，RO 输出逻辑 0。A 和 B 端之间加匹配电阻，一般可选 100Ω的电阻。



Top View

SP485/MAX485/SP3485/MAX3485 引脚（顶视图）

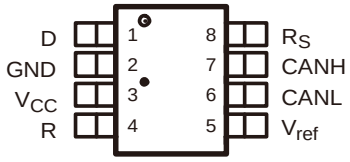
引脚	名称	说明
1	RO	接收器输出 Receiver Output
2	$\overline{RE}$	接收输出使能 Receiver Output Enable 低电平有效 Active LOW
3	DE	发送输出使能 Driver Output Enable 高电平有效 Active HIGH
4	DI	输出驱动器输入 Driver Input
5	GND	地 Ground Connection
6	A	差分信号正向端 Driver Output/Receiver Input. Non-inverting
7	B	差分信号反向端 Driver Output/Receiver Input.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

SP485 / MAX485 是 5V 的 RS485 收发器

SP3485 / MAX3485 是 3.3V 的 RS485 收发器

1.1.2. SN65HVD230

SN65HVD230 是德州仪器公司生产的 3.3V CAN 收发器，该器件适用于较高通信速率、良好抗干扰能力和高可靠性 CAN 总线的串行通信。SN65HVD230 具有高速、斜率和等待 3 种不同的工作模式。其工作模式控制可通过 Rs 控制引脚来实现。CAN 控制器的输出引脚 Tx 接到 SN65HVD230 的数据输入端 D，可将此 CAN 节点发送的数据传送到 CAN 网络中；而 CAN 控制器的接收引脚 Rx 和 SN65HVD230 的数据输出端 R 相连，用于接收数据。



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

2. 操作与现象

2.1. 准备工作

RS485 CAN Shield 模块两个

STM32 开发板两个（本手册用的是微雪电子的 Xnucleo 开发板，主控芯片是 STM32F103R）

杜邦线若干

2.2. 跳线说明

- D14(PB_9)、D15(PB_8)分别作为默认 CAN 的发送端和接收端。

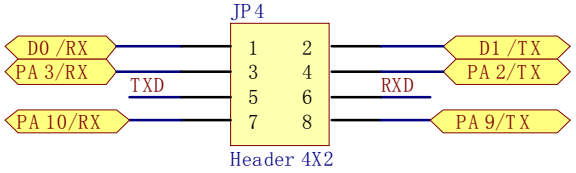
  注：PB_9,PB_8 作为 STM32 CAN1 管脚时编程需打开管脚重映射。

  GPIO_PinRemapConfig(GPIO_Remap1_CAN1, ENABLE);

- D7(PA_8)是 RS485 的发送接收使能端，高电平时为发送状态，低电平时为接收状态。

- D8(PA_9)、D2(PA_10)和 D0(PA_2)、D1(PA_3)分别是 UART1 和 UART2 的发送端和接收端。可通过 485 RXD/TXD JMP 跳线帽选择 UART1 或 UART2 作为 RS485 的输出输入端。

  注：Xnucleo 默认 PA_2、PA_3 作为串口转 USB 端口。若要用 D0、D1 作为 RS485 的串口，则还需变换 Xnucleo 中 JP4 相应跳线。用跳线帽将 1、3 管脚短接，2、4 管脚短接。Xuncleo 原理图中 JP4 串口跳线如下图所示：



- 模块间通信，CAN 端口的 H,L 分别和另一个模块的 CAN 端口 H,L 对接。RS485 端口的 A,B 分别和另一个模块的 RS485 端口 A,B 对接。

2.3. 工作原理

本测试程序采用 mbed 框架+STM32 库函数的形式,分为发送程序和接收程序两个程序。

CAN：

CAN 驱动程序采用 STM32 库函数编写，封装在 CAN.cpp 和 CAN.h 两个文件中。程序开始调用 CAN 初始化函数 `CAN_Config()`配置相关寄存器。

发送程序将要发送的数据保存在发送邮箱（`TxMessage`）中，再调用驱动函数 `CAN_Transmit(CAN1, &TxMessage)`发送出去。

而接收程序侧调用 `CAN_Receive(CAN1, CAN_FIFO0, &RxMessage);`将接收到的数据保存在接收邮箱`(RxMessage)`中。

RS485：

发送端程序控制 RS485_E 为高电平，使 RS485 处于发送状态，通过 `RS485.printf` 函数让数据通过 RS485 串口发送出去。而接收程序则开启接收中断，程序控制 RS485_E 为低电平，使 RS485 处于接收状态，中断服务函数通过 `RS485.scanf` 扫描接收到的数据。

接线：

- D14、D15 分别是默认 CAN 的发送端和接收端。
- D8、D2 是 RS485 的发送端和接收端。
- D7 是 RS485 发送接收使能端。
- D0、D1 将信息输出到 PC 端的串口。
- CAN 端口的 H,L 分别和另一个模块的 CAN 端口 H,L 对接。RS485 端口的 A,B 分别和另一个模块的 RS485 端口 A,B 对接。

2.3.1. 发送端程序说明

CAN：程序开始调用 CAN 初始化函数，配置相关寄存器。CAN 通信侧建立一个发送邮箱 TXMsg，将要发送的数据保存在邮箱中，再调用驱动函数发送出去。

RS485：控制 RS485_E 为高电平，使 RS485 处于发送状态，通过连接到 RS485 的串口将数据发送出去。

```
#include "mbed.h"
#include "CAN.h"

Serial pc(D1,D0);           //serial print message
Serial RS485(D8, D2);       //RS485_TX RS485_RX
DigitalOut RS485_E(D7);     //RS485_E

CanTxMsg TxMessage;
uint8_t TransmitMailbox = 0;
int i =0,j=0;

int main() {
    CAN_Config();//CAN 初始化
    RS485_E = 1;//使能 RS485 发送状态

    /* TxMessage */   //设置发送邮箱数据
    TxMessage.StdId = 0x10;
    TxMessage.ExtId = 0x1234;
    TxMessage.RTR=CAN_RTR_DATA;
    TxMessage.IDE=CAN_ID_STD;
    TxMessage.DLC=8;
    TxMessage.Data[0] = 'C';
    TxMessage.Data[1] = 'A';
    TxMessage.Data[2] = 'N';
    TxMessage.Data[3] = ' ';
    TxMessage.Data[4] = 'T';
    TxMessage.Data[5] = 'e';
    TxMessage.Data[6] = 's';
    TxMessage.Data[7] = 't';

    pc.printf( "**** This is a RS485_CAN_Shield Send test program
****\r\n");

    while(1) {

        RS485.printf("ncounter=%d ",j);//RS485 发送数据
        wait(1);
        TransmitMailbox = CAN_Transmit(CAN1, &TxMessage);//CAN 发送数据

        i = 0;
```

```
        while((CAN_TransmitStatus(CAN1, TransmitMailbox) != CANTXOK) &&
(i != 0xFFF)){
            i++;
        }

        if(i == 0xFFF){
            pc.printf("\r\can send fail\r\n");//等待超时，发送失败
        }
        else{
            pc.printf("\r\nCAN send TxMessage successfully \r\n");
            //发送成功
        }
        pc.printf("\r\nRS485 send: counter=%d\r\n",j++);//打印发送内容
        pc.printf("The CAN TxMsg: %s\r\n",TxMessage.Data);

        wait(1);
    }
}
```

2.3.2. 接收端程序说明

CAN：程序开始调用 CAN 初始化函数，配置相关寄存器。接收端查询 FIFO 中是否有数据，有的话，则将接收到的数据保存到接收邮箱 RxMessage 中，再通过串口打印出来。

RS485：使能 RS485 接收中断函数，控制 RS485_E 为低电平，使 RS485 处于接收状态，中断服务函数通过 RS485.scanf 扫描接收到的数据。

```
#include "mbed.h"
#include "CAN.h"

Serial pc(D1,D0);           //serial print message
Serial RS485(D8, D2);       //RS485_TX RS485_RX
DigitalOut RS485_E(D7);     //RS485_E

CanRxMsg RxMessage;         //RxMessage
char s[1024];

void callback()//RS485 接收中断处理函数
{// Note: you need to actually read from the serial to clear the RX interrupt
    RS485.scanf("%s",s);//保存接收数据
```

```
    pc.printf("\r\nRS485 Receive:%s \r\n",s);//打印接收信息
}

int main() {

    CAN_Config();//CAN 初始化
    RS485.attach(&callback);//开启 RS485 接收中断
    RS485_E = 0;//使能接收状态
    pc.printf( "**** This is a can receive test program ****\r\n");
    while(1) {
        while(CAN_MessagePending(CAN1, CAN_FIFO0) < 1)//等待数据到来
        {
        }
        CAN_Receive(CAN1, CAN_FIFO0, &RxMessage);//CAN 接收数据
        pc.printf("The CAN RxMsg: %s\r\n",RxMessage.Data);//打印接收数据
    }
}
```

2.3.3. 实验现象

发送端串口输出：

```
**** This is a RS485_CAN_Shield Send test program ****

CAN send TxMessage successfully

RS485 send: counter=0
The CAN TxMsg: CAN Test

CAN send TxMessage successfully

RS485 send: counter=1
The CAN TxMsg: CAN Test

CAN send TxMessage successfully

RS485 send: counter=2
The CAN TxMsg: CAN Test
```

接收端串口输出：

```
**** This is a can receive test program ****
```

```
RS485 Receive:ncounter=0
The CAN RxMsg: CAN Test

RS485 Receive:ncounter=1
The CAN RxMsg: CAN Test

RS485 Receive:ncounter=2
The CAN RxMsg: CAN Test

RS485 Receive:ncounter=3
The CAN RxMsg: CAN Test
```