RS485 Board (3.3V/5V) User Manual

1. Description

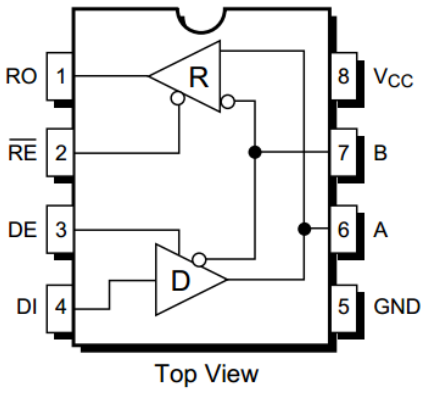
RS-485 is a serial communication protocol standard based on differential signal transmission. Comparing with RS-232 which has the transmission distance limit of 15m, RS485 can be used effectively over long distances and is widely used in many industrial applications.

RS-485 communication is a half-duplex communication using a twist pair to transmit differential signal, so it requires data receive and transmission modes switch.

- The RS485 Board is an accessory board used for adding the RS485 transceiver to your application board.
- SP485 / MAX485 are RS485 transceivers requiring 5V power supply.
- SP3485 / MAX3485 are RS485 transceivers requiring 3.3V power supply.

2. Hardware description

2.1 Pin function



SP485/MAX485/SP3485/MAX3485
Pin layout (Top view)

Pin	Name	Description
1	RO	Receiver Output
2	$\overline{RE}$	Receiver Output Enable Active LOW
3	DE	Driver Output Enable Active HIGH
4	DI	Driver Input
5	GND	Ground Connection
6	A	Driver Output/Receiver Input. Non-inverting
7	B	Driver Output/Receiver Input. Inverting
8	V_{CC}	V_{CC} < 5.25V

SP485 / MAX485 are RS485 transceivers requiring 5V power supply.
SP3485 / MAX3485 are RS485 transceivers requiring 3.3V power supply.

2.2 Function description

2.2.1 Pin description

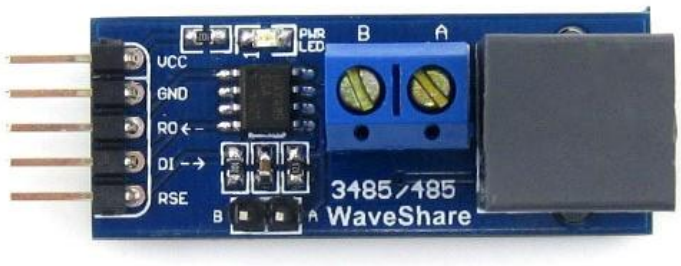


Figure 1: Module top view

Symbol	Meanings
V_{CC}	MCU power supply V_{CC}
GND	MCU power supply GND
RO	Data Receiver Output, TTL level, connected to RX of MCU
DI	Data Driver Input, TTL level, connected to TX of MCU
RSE	Active HIGH for Driver Input Enable; Active LOW for Receiver Output Enable. It can be controlled by a common IO interface.
A	Driver Output/Receiver Input. Non-inverting
B	Driver Output/Receiver Input. Inverting

Table 1: Module pins

2.2.2 Working principle

Normally, RS485 is in data receive mode.

When data transmission is required:

1) The program switches the pin RSE from LOW to HIGH;
2) UART sends data out;
3) The program switches the RS485 board from data transmission mode to data receive mode after data is sent out.
4) The flag for data sent out is provided by a specific register of UART. The program should check it to make sure if all the data is sent out.

2.3 Demos

2.3.1 Preparations

- Two RS485 boards: RS485 Board (1) and RS485 Board (2);
- Two STM32 development boards: STM32 development board (1) and STM32 development board (2) (Here we use the Open103R development board with STM32F103R as main control chip provided by Waveshare);
- Several wires or cables.

2.3.2 Processes

1) Connect the RS485 Board (1) to the STM32 development board (1), according to Table 2. And so do the RS485 Board (2) and the STM32 development board (2).

RS485 board pins	STM32 development board pins
V_{CC}	V_{CC}
GND	GND
RO	PA2
DI	PA3
RSE	PA0

Table 2: Connection between RS485 board and STM32

2) Connect an ULINK2 to the JTAG interface of Open103R development board (1), and download the program USART_accept.
3) Connect the ULINK2 to the JTAG interface of Open103R development board (2), and download the program USART_send.
4) Connect the pin A of the RS485 board (1) to the pin A of the RS485 board (2), and the pin B of the RS485 board (2) to the pin B of the RS485 board (2) with jumper cables.
   Notices: In this RS485 board, you can also use RJ11 telephone cables to connect the modules.
5) After powering up the modules, if the LEDs on the STM32 development board (1) and (2) are flickering at a same time in the same frequency, it means the RS485 communication is built up successfully.

2.3.3 Data transmission sample

```
int main(void)
{
    unsigned char i;
    USART_Configuration(); //initialize the serial port
    //initialize LED(PC9, PC10, PC11, PC12) and the pin RSE of the RS485 board
    GPIO_Configuration();
    GPIO_SetBits(GPIOA,GPIO_Pin_0);//set PA0 to 1, RS485 board is in data transmission mode
    while (1)
    {
        for(i=0;i<4;i++)
        {
            Delay(0x005fffff); //delay
            printf("%c",table[i]); //serial port sends data out, it means RS485 board is sending data out
            GPIO_Write(GPIOC,table[i]<<9); //LED lights up
        }
    }
}
```

2.3.4 Data receive sample

```
int main(void)
{
    USART_Configuration();//initialize the serial port
    // initialize LED(PC9, PC10, PC11, PC12) and the pin RES of the RS485 board
    GPIO_Configuration();
    NVIC_USART_Configuration();//set PA0 to 1, RS485 board is in data transmission mode
    GPIO_ResetBits(GPIOA,GPIO_Pin_0); // set PA0 to 0, switching RS485 board from data transmission mode to data receive mode
    while (1);
}

void USART2_IRQHandler(void) //serial port receives response interrupt function
{
    if(USART_GetITStatus(USART2, USART_IT_RXNE) != RESET)
```

```
    {
        USART_ClearITPendingBit(USART2,USART_IT_RXNE);
        //clear interrupt
        USART_SendData(USART2,
        USART_ReceiveData(USART2));//serial port sends data out
        GPIO_Write(GPIOC,USART_ReceiveData(USART2)<<9);//
        display the data received in LED
    }
}
```

2.3.5 Phenomenon

The LEDs on the STM32 development board (1) and Open103R development board(2) show the same result.