RS485 总线通信

1. 实验目的

- 掌握 STC15W4K32S4 系列 MCU 串行口原理。
- 掌握 RS485 总线通信的硬件原理。
- 了解 RS485 总线通信的软件协议。

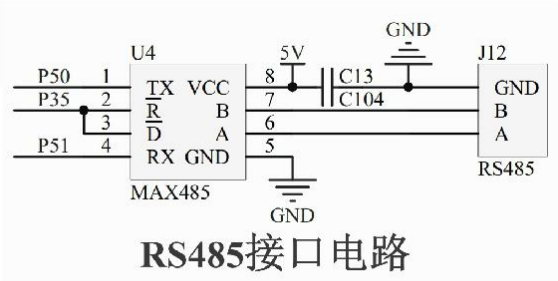
2. 实验内容

- 编写程序实现 RS485 总线收发数据。

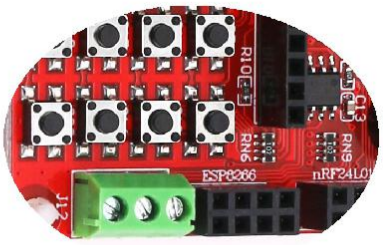
3. 硬件设计

3.1. 开发板串口硬件电路

进取者 STC15 开发板上设计了 RS485 电路（RS485 接口芯片选择的是 MAX485），具体电路及实物接口如下。



1 开发板原理图



2 开发板实物照

图 1：开发板 RS485 电路

✧ RS485 电路占用的单片机的引脚如下表：

表 1：串口电路引脚分配

UART	功能描述	对应 IO 口	说明
RXD	串口接收	P5.0	独立 GPIO
TXD	串口发送	P5.1	独立 GPIO
RE	RS485 收发使能控制	P3.5	独立 GPIO

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO。

3.2. RS485 电气性能

RS485 接口是一个物理接口，将多个 RS485 接口通过一定方式连接起来，可形成 RS485 总线。将多个 RS485 接口连接起来的方式有多种，一般不支持星型、树型或环型网络。下图是总线型连接的 RS485 系统通信框图，也是被最广泛应用的一种连接方式。

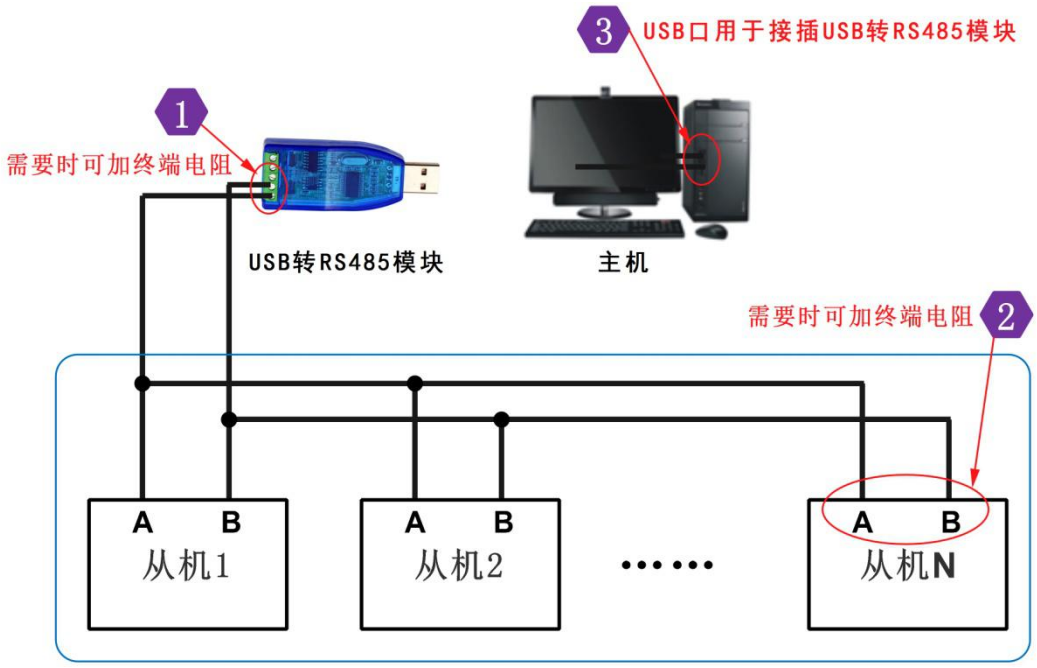


图 2： RS485 系统通信框图

✧ 注：RS485 系统通信时有几点注意事项：

1)连接电脑(主机)和 RS485 总线的可以不选择 USB 转 RS485 模块,如果电脑有 RS232 接口，也可选择 RS232 转 RS485 模块。

2）RS485 总线如果需要加终端电阻，最好的效果是总线的首尾两端各加一个。

RS485 总线是一种常规的通信总线，他不能做总线的自动仲裁，也就是不能同时发送数据以避免总线竞争，所以整个系统的通信效率必然较低，数据冗余量较大，对于速度要求高的应用场所不适应用 RS485 总线。同时由于 RS485 总线上通常只有一台主机，所以这种总线方式是典型的集中——分散型控制系统。

概括一下，RS485 接口或者说 RS485 总线具有以下电气性能：

1）传输方式：主从式，半双工。

2）传输速率：可配置（最高传输速率为 10Mbps）。

3）最大挂接节点数：一般最大支持 32 个节点（如果使用特制的 RS485 芯片，可以达到 128 个或者 256 个节点，最大的可以支持到 400 个节点）。

4）最大传输距离（不加中继）：受传输速率、通信电缆是否双绞、通信电缆芯线截面积等因素影响，一般在 100Kbps 的传输速率下，才可以达到最大的通信距离约为 1219 米。

3.3. RS485 通信协议

RS485 接口本身只是硬件接口，硬件通信接口建立后，在进行数据传输的设备之间需要约定一个数据协议，以使接收端能够解析收到的数据，这便是“协议”的概念。

通讯协议按照有无统一的标准来划分，可分成两种：

1）标准通信协议。此有统一标准的协议格式，如“ModBus”协议，标准的协议内容全面，包含的内容很多，但不易理解。

2）自定义通信协议或者说私有通信协议。用户可根据自己的需求，灵活定义一种协议，简单实用，这便是“自定义协议”。

✧ 注：“ModBus”协议也有 ASCII 模式和 RTU 模式两种，这两种又有很大差别，用户需根据自己实际需求来选择最合适自己的协议。

3.4. RS485 电路设计

RS485 电路总体上可以分为隔离型与非隔离型。隔离型比非隔离型在抗干扰、系统稳定性等方面都有更出色的表现，但有一些场合也可以用非隔离型。

非隔离型的电路非常简单，只需一个 RS485 芯片直接与 MCU 的串行通信口和一个 GPIO 控制口连接就可以。（开发板使用的就是最简单的非隔离型电路）

在某些工业控制领域，由于现场情况十分复杂，各个节点之间存在很高的共模电压。虽然 RS485 接口采用的是差分传输方式，具有一定的抗共模干扰的能力，但当共模电压超过 RS485 接收器的极限接收电压，即大于+12V 或小于－7V 时，接收器就再也无法正常工作了，严重时甚至会烧毁芯片和仪器设备。

解决此类问题的方法是通过 DC-DC 将系统电源和 RS485 收发器的电源隔离；通过隔离器件将信号隔离，彻底消除共模电压的影响。

实现隔离型 RS485 电路设计的方案可分为：

1）传统方式：用高速光耦、带隔离的 DC-DC、RS485 芯片构筑电路。

2）使用二次集成芯片，如 ADM2483、ADM257E 等。

✧ 注：隔离型 RS485 具体电路不在此介绍，还是那句话，根据自己的需求和成本来定方案。

4. 软件设计

RS485 通信本质还是异步串行通信，这和 UART 通信一样，不同之处是，RS485 接收和发送需要专用使能引脚控制，软件设计时需要注意。（软件控制 MAX485 芯片 RE 引脚切换为接收或发送模式时，注意软件适当延时，以满足硬件切换的时间需求。）

4.1.RS485 总线数据收发实验（串口 3）

✧ 注：本节的实验源码是在“实验 2-8-1：串口 1 收发实验（P3.0 和 P3.1）”的基础上修改。本节对应的实验源码是：“实验 2-16：RS485 总线数据收发实验（串口 3）”。

4.1.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 2：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	. c	外部串行口有关的用户自定义函数。
2	delay	. c	包含用户自定义延时函数。

4.1.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1.  #include "delay.h"
2.  #include "uart.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 3：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

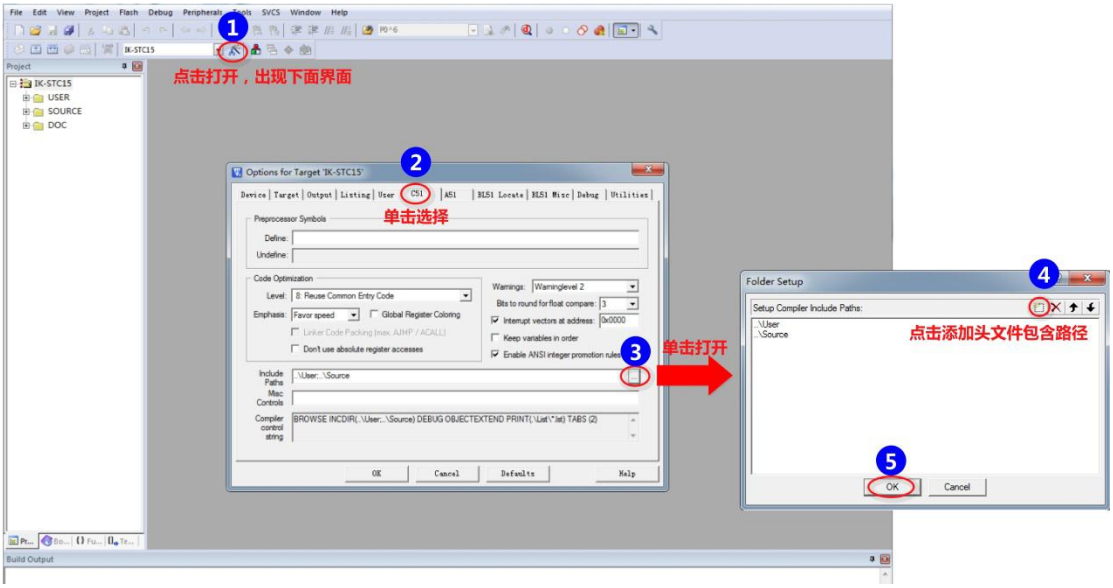


图 3：添加头文件包含路径

4.1.3. 编写代码

首先，在 uart.c 文件中编写串口 3 的初始化函数 Uart3_Init，代码如下。

程序清单：串口 3 初始化函数

```
/*************************************************************************
 * 描  述 : 串口 3 初始化函数
 * 入  参 : 无
 * 返回值 : 无
备注：波特率 9600bps    晶振 11.0592MHz
 *************************************************************************/
void Uart3_Init(void)
{
    P_SW2|=S3_S;          //选择 P5.0、P5.1 为串口 3 使用
    S3CON |= 0x10;        //启动串行接收器
    S3CON &= 0x30;        //8 位数据,可变波特率,串口 3 选择定时器 2 为波特率发生器
    AUXR |= 0x04;           //定时器 2 时钟为 Fosc,即 1T
    T2L = 0xE0;           //设定定时初值
    T2H = 0xFE;           //设定定时初值
    AUXR |= 0x10;           //启动定时器 2
    IE2 |= 0x08;        // 串口 3 中断打开
}
```

然后，编写串口 3 发送数据函数，把要发送的字节存放于数据缓存寄存器中，直到数据发送完成，代码如下。

程序清单：数据发送函数函数

```
/*************************************************************************
 * 描  述 : 串口 3 发送数据函数
```

```
3.   * 入   参 : uint8 数据
4.   * 返回值 : 无
5.   ******************************************************************************/
6.  void SendDataByUart3(uint8 dat)
7.  {
8.      S3BUF = dat;                    //写数据到 UART 数据寄存器
9.          while(!(S3CON&S3TI));        //在停止位没有发送时，S3TI 为 0 即一直等待
10.     S3CON&=~S3TI;                   //清除 S3CON 寄存器对应 S3TI 位（该位必须软件清零）
11. }
```

之后，编写串口 3 的中断服务函数，将接收的数据存放到用户自定义变量 uart3temp 中，代码如下。

程序清单：中断服务函数

```
1.  /*********************************************************************************
2.   * 描   述 : 串口 3 中断服务函数
3.   * 入   参 : 无
4.   * 返回值 : 无
5.   ******************************************************************************/
6.  void Uart3() interrupt UART3_VECTOR using 1
7.  {
8.      IE2 &= 0xF7;                      // 串口 3 中断关闭
9.      Flag=TRUE;                        //接收到数据,接收标识符有效
10.     if (S3CON & S3RI)                 //串行接收到停止位的中间时刻时，该位置 1
11.   {
12.       S3CON &= ~S3RI;                //清除 S3CON 寄存器对应 S3RI 位(该位必须软件清零)
13.            uart3temp = S3BUF;
14.    }
15.   if (S3CON & S3TI)                  //在停止位开始发送时，该位置 1
16.    {
17.       S3CON &= ~S3TI;                //清除 S3CON 寄存器对应 S3TI 位(该位必须软件清零)
18.    }
19.     IE2 |= 0x08;                      // 串口 3 中断打开
20. }
```

最后，用户定义一个自定义函数 UART3_Tx485_Puts，该函数将接收的数据原样返回去并加上回车符，注意 RS485 发送完后要将 RE 引脚控制为接收模式，切换引脚发送还是接收时有软件延时。主函数 main 在主循环中调用该函数。具体代码如下。

代码清单：用户函数 UART3_Tx485_Puts

```
1.  /*********************************************************************************
2.   * 描   述 : RS485 接收到数据后发送出去
3.   * 入   参 : 无
```

```
4.   * 返回值 : 无
5.   ******************************************************************************/
6.  void USART3_Tx485_Puts(void)
7.  {
8.    if(Flag)       //有新数据通过串口被接收到
9.      {
10.         rs485_dr=1;                                  //控制 485 发送
11.         delay_ms(1);                                 //延时 1ms，不可省去
12.         IE2 &= 0xF7;                                       //串口 3 中断关闭
13.         SendDataByUart3(uart3temp);                  //发送字符
14.         SendDataByUart3(0x0D);                       //发送换行符
15.         SendDataByUart3(0x0A);                       //发送换行符
16.         IE2 |= 0x08;                                 //串口 3 中断打开
17.         delay_ms(1);                                 //延时 1ms，不可省去
18.         rs485_dr=0;                                  //控制 485 接收
19.         Flag=FALSE;                                  //清除接收标识符
20.   }
21. }
```

代码清单：主函数

```
1.  int main()
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////
9.      P3M1 &= 0xDF;   P3M0 &= 0xDF;                    //设置 P3.5 为准双向口
10.     P5M1 &= 0xFC; P5M0 &= 0xFC;                    //设置 P5.0~P5.1 为准双向口
11.
12.     rs485_dr=0;             //控制 485 接收
13.     Uart3_Init();           //串口 3 初始化
14.     EA = 1;                 //总中断打开
15.
16.     while(1)
17.     {
18.       USART3_Tx485_Puts();    //RS485 接收到 1 个字符后返回该字符
19.     }
20. }
```

4.1.4. 硬件连接

本实验需要用到 USB 转 RS485 转换器，以实现开发板通过 RS485 接口与 PC 通信，实验连接图如下。

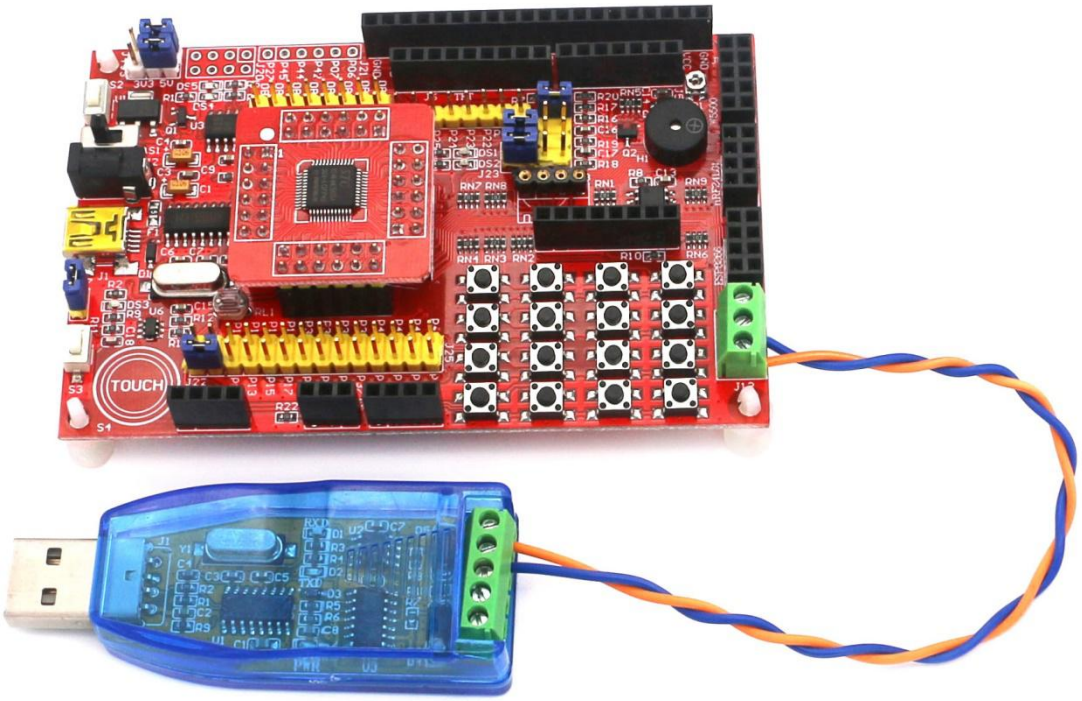


图 4：RS485 通信实验连接图

4.1.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-16：RS485 总线数据收发实验（串口 3）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\RS485\projec”目录下的工程“RS485.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“RS485.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，打开串口调试助手，波特率设置为 9600，在发送区发送什么字符在接收区就会收到什么字符（注意是字符不是字符串）。