基于 485 总线的评分系统
实验报告

班级：计科 2104

学号：202108010426

姓名：冷长佼

目录

1 实验项目

1.1 项目名称

基于 485 总线的评分系统

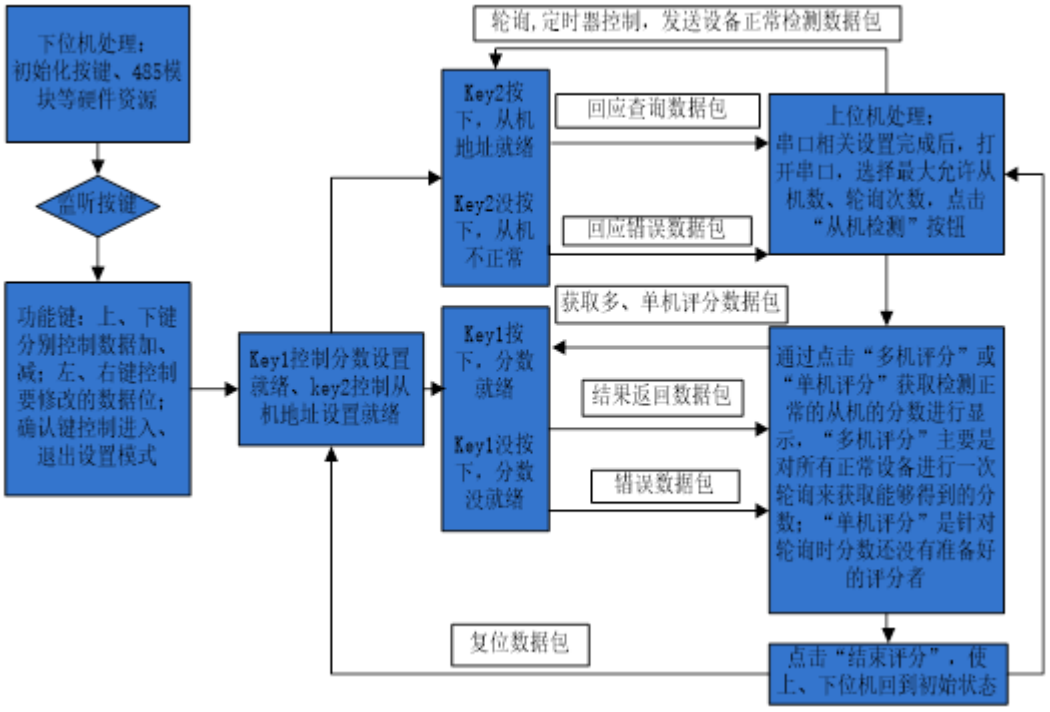
1.2 实验目的

通过本案例加深理解 RS485 通信方式，实现上位机的主控制器与所有的下位机进行通信。

1.3 实验资源

PC、STC 学习板等

1.4 实验简介



1.5 实验通信协议

本案例模拟 Modbus 协议，采用主、从技术，上位机的主控制器可以与所有的下位机通信，也可以单独与一个指定的下位机通信。模拟 Modbus 协议中，上下位机的数据包都只含 5 个字节，其基本格式为：数据包头（0x5A）+地址码（广播地址/从机地址）+功能码+携带数据（一个字节）+校验码字节，携带数据部分可以扩充多个字节，可以视情况进行修改。数据包具体定义如下：

（1）主机检测从机是否正常相关数据包：（主机与单个从机设备通信）

A、设备正常检测数据包：

方向：上位机----->下位机

数据包消息：数据包头+从机地址+检测功能码（Fun_CheckSlave）+自定义内容（Check_Content）+校验字节

功能：查询下位机是否正常。正常，下位机发送回应查询数据包；不正常，则下位机不予回应；数据传输过程发生错误，下位机发送回应错误数据包，上位机可以通过设置多次轮询来重新检测该设备是否正常；

B、回应查询数据包：

方向：下位机----->上位机

数据包消息：数据包头+从机地址+检测功能码（Fun_CheckSlave）+自定义内容（接收自主机 Check_Content）+校验字节

C、回应错误数据包：

方向：下位机----->上位机

数据包消息：数据包头+从机地址+检测功能码（Fun_CheckSlave）+错误码（ErrorInfo）+校验字节

（2）主机获取从机评分相关数据包：（主机与单个从机设备通信）

D、获取多、单机评分数据包：

方向：上位机----->下位机

数据包消息：数据包头+检测正常从机地址（0x00）+读下位机功能码（Fun_ReadInfo）+从机地址+校验字节

功能：对检测正常的设备，进行一次轮询，获取评分已经准备好的从机的分数。对于单机直

接进行通信，没有轮询。

E、结果返回数据包：

方向：下位机----->上位机

数据包消息：数据包头+从机地址+读下位机功能码（Fun_ReadInfo）+从机返回的分数值+校验字节（分数值>100:表示上面提及的未准备好，回应错误数据包）

（3）此轮评分结束相关数据包

F、复位数据包：（主机与所有从机通信）

方向：上位机----->下位机

数据包消息：数据包头+广播地址+复位功能码（Fun_Reset）+从机返回的分数值（0x00）+校验字节

功能：指示所有正常连接的从机进行复位操作，准备下一轮的评分。

2 实验任务

2.1 实验任务一

任务名称：B 级任务

2.1.1 实验步骤

1、 阅读程序系统流程框图，明确双机通信的功能需求。

2、 熟悉上一节中模拟 MODBUS 协议的数据包结构，相关功能码及附加数据定义

功能码	读下位机功能码	0X03
	检测功能码	0X08
	地址错误功能码	0X10
	复位功能码	0X01
附加数据	错误码	0X6F
	包头	0X5A
	广播地址	0X00
	自定义内容	0X13

3、 回顾 485 总线数据收发实验，搭建双机通信电路。参考上一节内容确保 STC 从机编号和评分设定完成后，按下 KEY2、KEY3 按键标志，第 1 位和第 8 位 LED 灯被点亮。

4、 PC 端串口设置如下：

串口波特率：9600　数据位：8 位　　校验位：无　停止位：1

5、 所编写的 PC 端程序应参考上一节中的通信协议完成一次完整的评分过程：

- 需要包含串口的设置
- 主节点发起从机检测过程：发送指定从机编号正常检测数据包，判断回应查询数据包是否符合上一节中的通信协议。

- 主机获取从机评分过程：发送指定从机评分相关数据包，判断回应查询数据包是否符合上一节中的通信协议。
- 主机发起结束评分的过程，若复位成功，STC 从机上第 1 位和第 8 位 LED 灯会熄灭。
- 展示串口相关信息，展示检测到的从机编号和从机的评分等。

2.1.2 程序代码（注意代码规范）

(1)串口读写

串口读写参考串口实验中老师给出的参考代码改写，为了方便读写，定义了 serial 类，包含文件句柄和 epoll 事件。并定义相关函数。在构造函数中对事件监听进行设置，读写函数中，将读取和写入的数据都保存在 vector 中。

写入数据时直接调用 write 进行写入，count 计数保证数据全部写入。读取数据时，等待事件后进行读取，为了保证读取足够字节的数据，循环进行读取。在前面的串口实验中，单片机连续重复发出数据，因此可以通过多次读取保证读取一定字节的数据，在本次实验中，由于单片机只发送一次数据，因此 while 循环没有实际意义。

```
std::vector<unsigned char> serial::myRead(size_t n) const {
    size_t count = 0;
    std::vector<unsigned char> buffer(n);
    while (count < n) {
        epoll_event event1 = event;
        // 等待数据，指定超时值，避免无限期阻塞等待，没有等到则不读取
        if(epoll_wait(epfd, &event1, 1, 5000) == 0) continue;
        // 读取数据，然后根据读取到的数据数量决定是否需要继续读取
        count += ::read(board, &buffer[count], n-count);
    }
    return buffer;
}

void serial::myWrite(const std::vector<unsigned char> &data) const {
    size_t count = 0;
    while (count < data.size()) {
        // 向串口写入数据
        count += ::write(board, &data[count], data.size() - count);
    }
}
```

(2)评分功能实现

设备检测功能

设备检测需要先发送设备检测数据包，然后接受从机数据包，进行数据分析。设备检测功能通过 check 函数实现。发送的数据为协议定义的 5a + 从机地址 + 检测功能码(08) + 自定义内容 check_content(13) + 校验码。其中地址为用户输入并作为参数传入该函数，校验码使用累加和，将传入的地址加上其他位的值得到。然后将地址和校验码插入 vector 中，写入数据。发送数据后睡眠 1s 等待从机作出反应，然后读取数据。首先检验校验码是否正确，错误则直接返回宏定义的 check_failed。 如果校验正确，判断是否得到正确的查询回应，即与主机发送的数据是否相同，如果相同返回 1，不相同则判断回应查询是否得到了错误回应(0x6f)，并返回 0。

```
#define error -1
#define check_failed -2
```

```
//设备正常检测：5a + 从机地址 + 检测功能码(08) + check_content(13) + 校验码
int check(serial &serial1,int addr){
    int check_code = 117 + addr, ret;               //校验码为累加和
    vector<uchar> code = {0x5a, 0x08, 0x13};
    vector<uchar> rec;
    code.insert(code.begin()+1, (uchar)addr);           //插入从机地址
    code.push_back((uchar)check_code);              //插入校验码
    /* 写入并接收回应数据包 */
    serial1.myWrite(code);
    sleep(1);
    rec = serial1.myRead(5);
    if(rec[0] + rec[1] + rec[2] + rec[3] != rec[4]) return check_failed; //校验不通过
    if(format(rec) == format(code))                 //检验接收数据包是否与发送相同，相同则从机地址正确
        ret = 1;
    else if(rec[3] == 0x6f) ret = 0;                    //回应错误
    return ret;
}
```

获取分数功能

```
//获取从机分数：5a + 00 + 读取功能码03 + 从机地址 + 校验字节
int get_score(serial &serial1, int addr){
    int check_code = 93 + addr, ret;                //校验码为累加和
    vector<uchar> code = {0x5a, 0x00, 0x03};
    vector<uchar> rec;
    code.insert(code.begin()+3, (uchar)addr);           //插入从机地址
    code.push_back((uchar)check_code);              //插入校验码
    /* 写入并接收回应数据包 */
    serial1.myWrite(code);
    sleep(1);
    rec = serial1.myRead(5);
    if(rec[0] + rec[1] + rec[2] + rec[3] != rec[4]) return check_failed; //校验不通过
    if(rec[3] == 0x6f) ret = error;                 //回应错误
    else
        ret = (int)rec[3];                      //转为数字
    return ret;
}
```

从机复位

获取分数功能定义 get_score 函数实现。与检测设备类似，先发送查询分数的数据包，然后获取从机回应的数据包进行分析。具体实现与设备检测类似，通过传入的从机地址计算校验码，然后将数据写入串口。写入后再读取串口数据，检验校验码，检查是否错误，如果没有异常，将分数值转换为 10 进制数字的整型返回。

复位功能定义 reset 函数实现。复位只需要发送数据，不需要读取数据，且数据是固定的 5a0001005b，因此每次只需要向串口写入该数据。经过测试，从机不一定能收到主机发送的数据，并成功复位，因此通过多次发送复位数据包的方式，确保从机能够复位。其实在实现设备检测和获取分数时，也存在从机无法成功接收数据而无相应的情况。但是由于设备检测和获取分数需要读取从机发送的数据，如果重复向从机发送数据，从机回复数据时就会产生总线冲突，导致数据错误，因此前两个功能没有重复向串口写入数据。

```
//从机复位：5a + 广播地址00 + 复位功能码01 + 00 + 校验字节
void reset(serial &serial1){
    vector<uchar> code = {0x5a, 0x00, 0x01,0x00,0x5b};
    /* 发送数据包 */
    for(int i=0;i<500;i++) {
        serial1.myWrite(code);
    }
    cout<<"从机已复位"<<endl;
    return;
}
```

(3)主函数

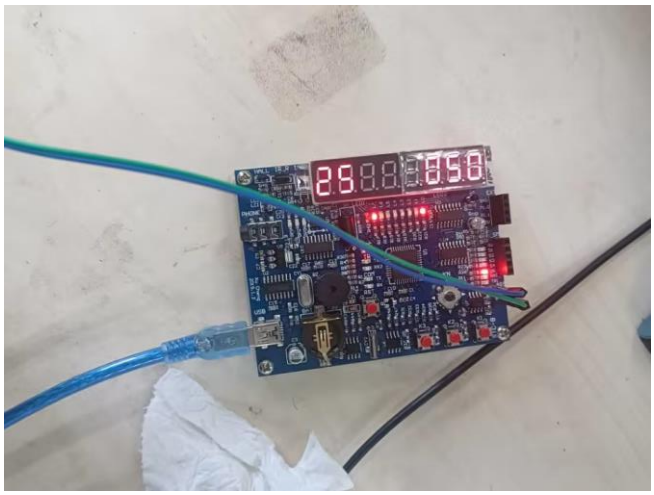
main 函数中接收用户输入，主要接收从机地址，复位命令和退出程序的命令。并调用上述实现的函数进行与从机的通信。在调用函数后对返回值进行检查，判断是否发生了错误，并对不同的错误给出输出。

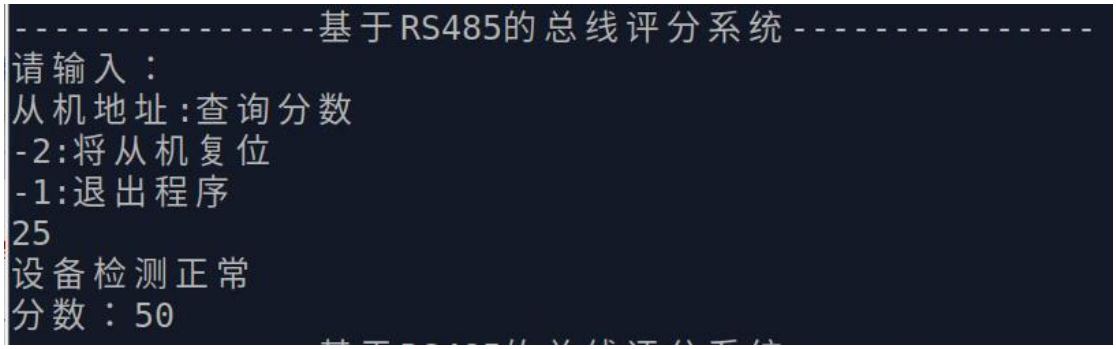
```
int main() {
    serial serial1("/dev/ttyUSB0", B9600);
    int addr, check_ret, score;
    while(1){
        cout<<"---------------基于RS485的总线评分系统---------------"<<endl;
        cout<<"请输入：\n从机地址:查询分数\n-2:将从机复位\n-1:退出程序"<<endl;
        cin>>addr;
        if(addr == -1) break;
        if(addr == -2) {
            reset(serial1);
            continue;
        }
        check_ret = check(serial1, addr);
        if(check_ret == check_failed) cout<<"数据校验错误"<<endl;
        else if(check_ret == 1){
            cout<<"设备检测正常"<<endl;
            sleep(1);
            score = get_score(serial1, addr);
            if(score == check_failed)
                cout<<"数据校验错误"<<endl;
            else if(score == error)
                cout<<"回应错误"<<endl;
            else
                cout<<"分数: "<<score<<endl;
        }
        else if(check_ret == error){
            cout<<"回应错误"<<endl;
            continue;
        }
    else{
        cout<<"数据传输错误"<<endl;
    }
    }
    return 0;
}
```

2.1.3 运行结果分析

运行界面：

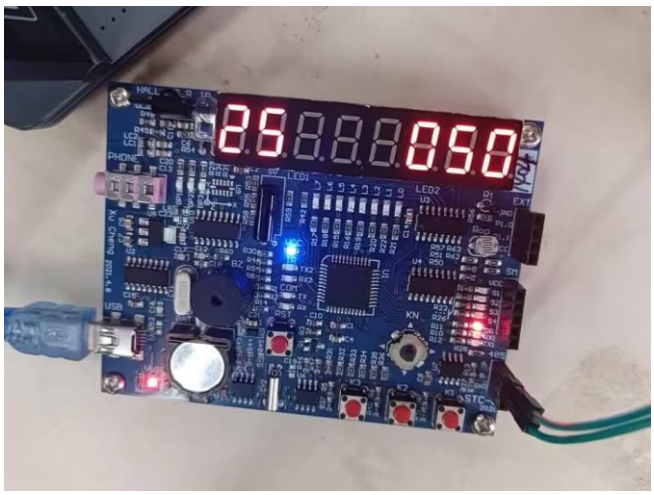
将单片机程序下载到单片机，将主机与从机分别连接到计算机，在主机连接的计算机上运行程序。从机首先设置一个地址和分数，设置地址为 25，分数为 50：





在主机连接的计算机上运行程序，输入从机地址，尝试获取分数，获取到了正确的分数。

输入复位命令-2，将从机复位：



从机复位成功，第 1 位和第 8 位 LED 灯熄灭。

运行结果分析

通过测试，程序可以完成评分功能。但发送数据后，从机不一定能够接收数据并发出回应，因此需要多次尝试检测设备，获取分数才能获取数据。推测这与单片机程序有关，单片机程序不能及时接收到主机发送的数据，也只发送一次数据给主机，导致通信存在困难。程序的设计应该是正确的。

2.2 实验任务二

任务名称：A 级任务

2.2.1 实验步骤

在 B 级任务基础上，扩充程序功能如：允许最大从机数、轮询次数、错误数据包的处理、统计多人评分的平均分等。（答辩评分的主要依据）

2.2.2 程序代码（注意代码规范）

在实验任务一代码的基础上修改

增加第三个选项：从 1 开始查询地址并查询分数

```
int main() {
    serial serial1("/dev/ttyUSB0", B9600);
    int addr, check_ret, score;
    while(1){
        lop: cout<<"---------------基于RS485的总线评分系统---------------"<<endl;
        cout<<"请输入：\n-3:从1开始查询地址并查询分数\n-2:将从机复位\n-1:退出程序"<<endl;
        cin>>addr;
        if(addr == -1) goto bye;
        else if(addr == -2) {
            reset(serial1);
            continue;
        }
```

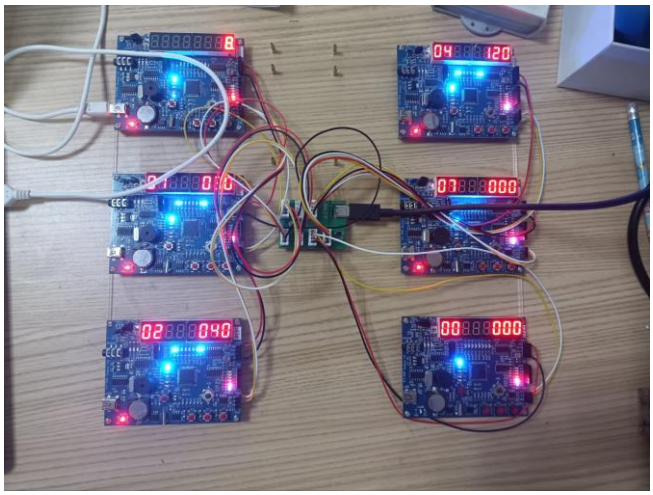
Main 函数中轮询

```
    if(addr == -3){
    int index = 0;
    for(addr = 1;addr<100;addr++ ){
    score = check_failed;
        check_ret = check(serial1, addr);
        if(check_ret == check_failed) {cout<<"数据校验错误,尝试"<<addr+1<<endl;continue;}
        else if(check_ret == 1){
            cout<<"设备检测正常"<<endl;
        mache[index].num =  addr;
        score = get_score(serial1, addr);
            if(score == check_failed)
                cout<<"数据校验错误"<<endl;
            else if(score == error)
                cout<<"回应错误"<<endl;
            else {
        cout<<"分数: "<<score<<endl;
        mache[index].score = score;
        index++;
}
```

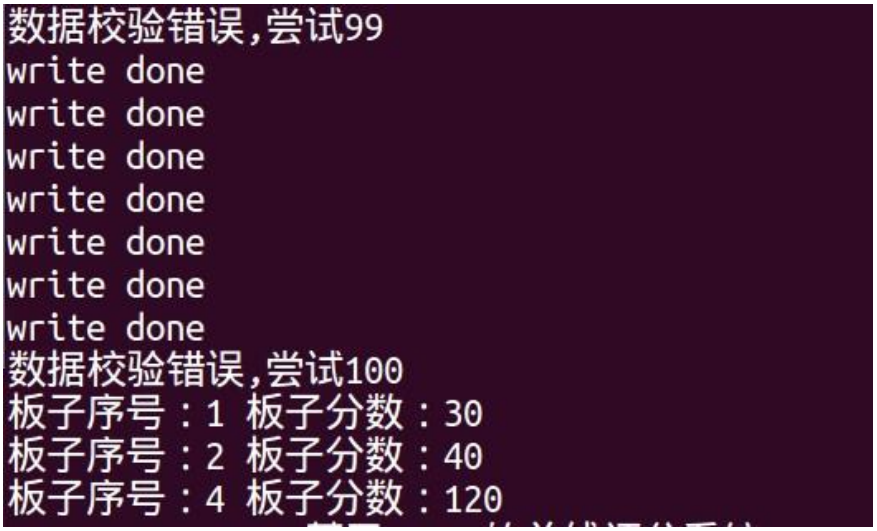
2.2.3 运行结果分析

运行界面：

多机检测，分别设置地址为 1，2,4,7，分数为 30，40,120,0。



从 1 开始轮询，检测得



运行结果分析

七号板子没有检测到，代码任有不足。在实验时经常出现 input/output 这个错误，也会出现检测不到 1 号板子的情况，可能是因为连接线的接触不良导致。

3 总结

3.1 小组分工，个人任务总结

代码分析与改进：胡盼阳，韦华喜

实验报告撰写：冷长佼

实验操作完成：胡盼阳，韦华喜，冷长佼

3.2 心得体会

通过本次实验，复习并加深了对 RS485 总线通信的理解，也进一步熟悉了使用 c++进行串口的数据读写，以及时间监听等机制。成功实现了评分系统。在实现的过程中与同学配合，将串口及事件监听作为类封装参考了同学的设计，数据的收发及解析主要结合实验资料中的定义完成。最后的多级评分尝试在原程序的基础上进行实现，但是由于单片机不能每次都准确的接收数据并进行响应，没有测试成功。通过串口相关的几次实验，我对单片机与计算机的串口通信以及基于 RS485 的通信都有了深入的了解。为后续相关知识的学习建立了基础。