STM32串口通信校验问题

STM32串口通信校验问题
这里以串口作为传输媒介，
介绍下怎样来发送接收一个完整的数据包。
过程涉及到封包与解包。
设计一个良好的包传输机制很有利于数据传输的稳定性以及正确性。
串口只是一种传输媒介，
这种包机制同时也可以用于SPI,I2C的总线下的数据传输。
在单片机通信系统（多机通信以及PC与单片机通信）中，是很常见的问题。
一、根据帧头帧尾或者帧长检测一个数据帧
1、帧头+数据+校验+帧尾
这是一个典型的方案，
但是对帧头与帧尾在设计的时候都要注意，也就是说帧头、帧尾不能在所传输的数据域中出现，一旦出现可能就被误判。
如果用中断来接收的话，程序基本可以这么实现：

```
unsigned char recstatu;//表示是否处于一个正在接收数据包的状态
unsigned char ccnt;     //计数
unsigned char packerflag;//是否接收到一个完整的数据包标志
unsigned char rxbuf[100];//接收数据的缓冲区
void UartHandler()
{
    unsigned char tmpch;
    tmpch = UARTRBR; //uartrbr
    if(tmpch 是包头)                    //检测是否是包头
    {
        recstatu = 1;
        ccnt   = 0 ;
        packerflag = 0;
        return ;
    }
    if(tmpch是包尾)                     //检测是否是包尾
    {
        recstatu = 0;
        packerflag = 1;                 //用于告知系统已经接收到一个完整的数据包
        return ;
    }
   if(recstatu ==1)                     //是否处于接收数据包状态
    {
        rxbuf[ccnt++] = tmpch;
    }
}
```

上面也就是接收一个数据包，
但是再次提醒，包头和包尾不能在数据域中出现，一旦出现将会出现误判。
另外一个。数据的校验算法是很必要的，
在数据传输中，由于受到干扰，很难免有时出现数据错误，
加上校验码可在发现数据传输错误时，可以要求数据的另一方重新发送，或是进行简单的丢弃处理。
校验算法不一定要很复杂，普通的加和，异或，以及循环冗余都是可以的。
我上面的接收程序在接收数据时，已经将包头和包尾去掉，这些可以根据自己的需求加上，关键是要理解原理。
上述包协议出现了以下的几种变种：
1.1 帧头+数据长度+数据+校验值
1.2包长+校验值
上面两种其实都是知道了数据包的长度，然后根据接收字节的长度来判断一个完整的数据包。
例如，定义一个数据包的长度为256字节，那我们就可以一直接收，直到接收到256个字节，就认为是一个数据包。
但是，会不会存在问题呢？比如说从机向主机发送数据，发送了一半，掉电，重启，开机后继续发送，这很明显接收到的数据就不对了，所以此时很有必要定义一个超限时间，比如我们可以维护下面这样的一个结构体。

```
struct  uartrd{
char  rd[ 256];
unsigned int timeout;
```

}
成员变量rd用来存放接收到的数据字节；
成员变量timeout用来维护超时值，
这里主要讨论这个。这个数值怎么维护呢，可以用一个定时器来维护，以可以放在普通的滴答中断里面来维护，也可以根据系统运行一条指令的周期，在自己的循环中来维护，给其设置个初值，比如说100，当有第一个数据到来以后，timeout在指定的时间就会减少1，减少到0时，就认为超时，不论是否接收到足够的数据，都应该抛弃。

二、根据接收超时来判断一个数据包
2.1 数据+校验
核心思想是
如果在达到一定的时间没有接受到数据，就认为数据包接收完成。
modbus协议里就有通过时间间隔来判断帧结束的。
具体实现是要使用一个定时器，
在接收到第一个数据时候，开启定时器，
在接收到一个数据时候，就将定时器清零，
让定时器重新开始计时，
如果设定的超时时间到（超时时间长度可以设置为5个正常接收的周期），
则认为在这一段时间内没有接受到新的数据，就认为接收到一个完整的数据包了。
流程大体如下图所示：

进行一个简单的小的总结，上述几种方法都还是较为常用的，在具体的实现上，可以根据具体的实际情况，设计出具体的通讯协议。
数据校验位，有时候感觉不出来其重要性，但是一定要加上，对数据进行一个相关的验证还是必要的。
现在很在MCU都带有FIFO，DMA等功能，所以有时候利用上这些特性，可以设计出更好的通讯方式。
有的人问在接受串口数据时候是应该中断一次接收一个，还是进入中断后接收一段数据呢，我认为应该中断接收一个，因为CPU是很快的，至少对于串口是这样，在接受每个数据的间隔期间，处理器还是可以做些其他工作的。这是在裸机下的模型。
在多线程中，那就可以直接建立一个数据接收线程。