# Modbus系列项目总结

## 一、MODBUS\_RTU协议新认识

1、Modbus\_RTU协议的帧结束信号是以字节与字节间间隔时间超过3.5字节时间则判定其为一帧结束的。

2、Modbus\_RTU协议具有较为严格的通讯要求，在字节与字节之间的时间长度超过1.5字节长度后其将判定该帧数据失效，直接丢弃该帧数据。

3、Modbus\_RTU协议中，每字节数据都应为11位格式，默认校验方式必须为偶校验，

无校验时，停止位应该为两位，其它校验方式为1位。

4、在通讯速率在19200Bps及以下时，严格遵守2的要求，而大于19200Bps时，建议使用字节超时时间750us，帧超时时间为1.750ms以降低CPU负担。

## 二、对于软件开发的新认识

1、软件开发时第一步应该详细了解思考项目需求，确定项目最终目标。

2、数据结构的确定十分重要，数据结构的确定应根据软件模型来确定，必须经过仔细思考，尽量做到占用空间小而能表述全部需求，同时也应该考虑其调用的频繁程度，考虑读取写入速度。

3、代码应从单一目标出发，切忌尝试在同一份代码中实现多种功能的综合。

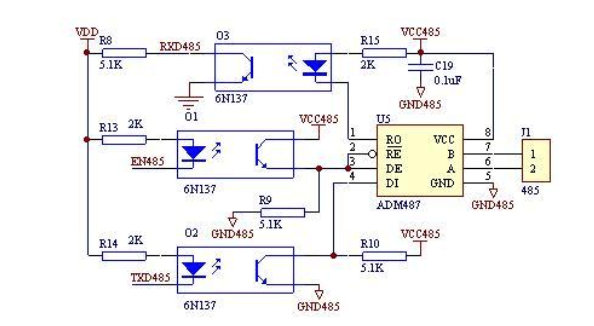
4、代码结构要合理，做到底层相关的代码与上层逻辑严格分离，形成清晰的接口函数。

5、过分强调性能是不正确的想法，应充分考虑代码的健壮性、可移植性、可修改性。

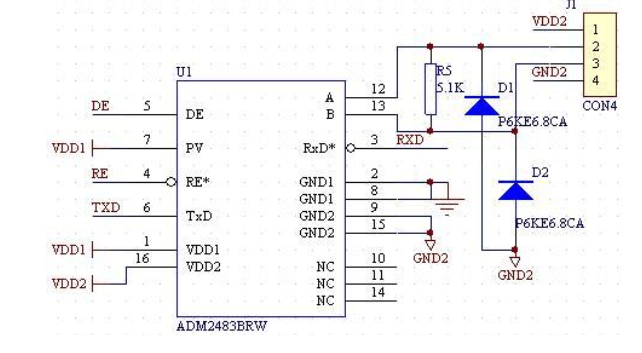
6、一份好的代码，必须有好的注释或说明文档。

## 三、对485接口的硬件设计认识

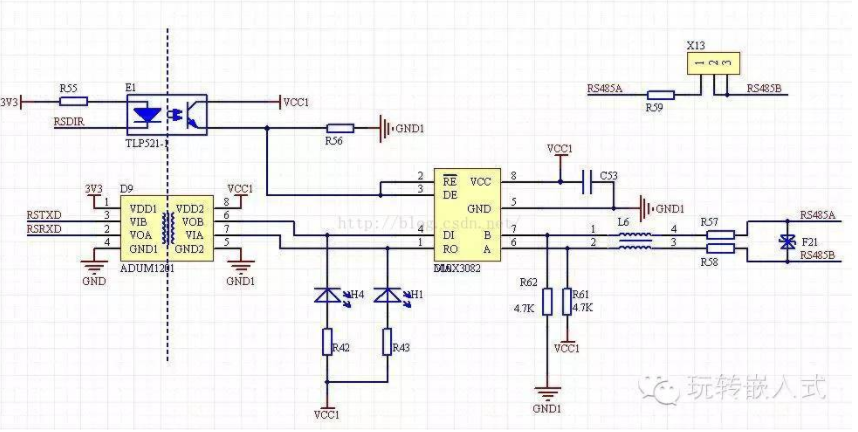
1、对于485接口隔离，了解到的实现方式有两种一种是使用高速光耦+DC-DC隔离+485芯片进行隔离，使用的光耦6N137（10M bit/s）可满足115200Bps通讯速度的要求，DC-DC隔离可采用MORNSUN出品的B0505LM-1W（根据电压需求）系列产品进行隔离（该系列部分是开环设计，既输出电压是随负责变化的，如果对电源有特殊要求，应该加入线性稳压，或者调整假负载,LDO如ME6209系列）。另外一种是使用磁隔离，有集成芯片ADM2483带磁隔离和485一体，ADM2587单电源完全隔离，也有单一的ADuM1201等只带有数字通道磁隔离的，其通讯速度十分优异，但需要注意在强磁环境下可能会出现故障。另外市面上还有直接集成好隔离电源和接口隔离的封装模块如RSM3485CT等，价格在20元至100不等。对于隔离电路其典型接法如下：



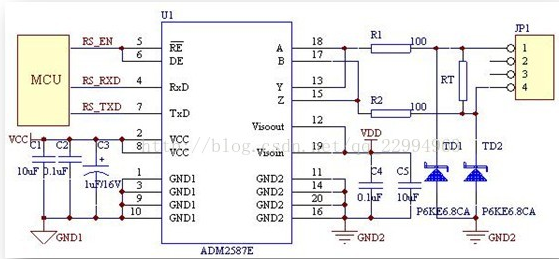
（1）光耦隔离电路



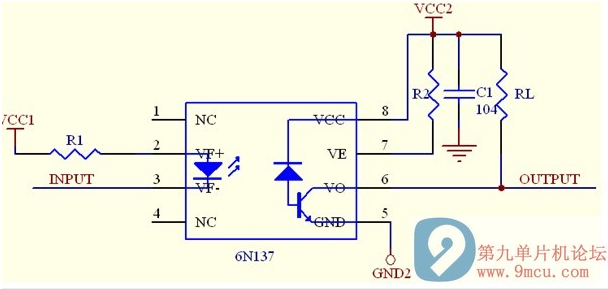
（2）磁隔离集成芯片电路



（3）磁通道隔离集

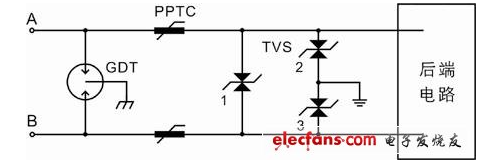


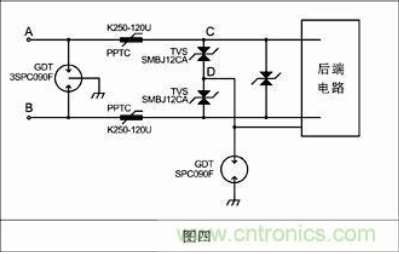
（4）单电源完全隔离



（5）6N137典型电路

2、对于485电路接口的防雷击设计，主要运用到三种原件陶瓷放电管GDT，自恢复保险丝PPTC，瞬态抑制二极管TVS。





（6）防雷击电路

当雷击发生时，感应过电压由两端引入，GDT做一级防护，此时过电压被大大削弱到数百伏左右，再经过PPTC限浪,TVS做二次限压，使到后端电路的电压被箝制在8V左右，从而实现对后端电路的保护。TVS2/3做共模保护，TVS1做差模保护。

3、对于485电路自动收发电路



3、 Stm32f10x系列的PC13，PC14和PC15引脚通过电源开关进行供电，因此这三个引脚作为输出引脚时有以下限制： 作为输出脚时只能工作在2MHz模式下 ,最大驱动负载为30pF ,同一时间，三个引脚中只有一个引脚能作为输出引脚。PC15似乎无法作为输入(未彻底实验)。

4、环形缓冲区： 环形缓冲区，顾名思义就是一个环状的存储数据的区域，其空间使用数组进行构造（链表也可以）。环形缓冲区特点是读和写可以是分开的，写入数据之后可以先不去读取，等到需要读取的时候再去读取，并且数据一经读取之后下次就不能再去读取（当然也可以实现重复读取的效果，不过大多用作一次性读取），等于说是一次性的读取。

假设一个长度为256字节的数组，构建出一个环形缓冲区，当写操作进行到数组的第256项之后，再一次写入就会回到第0个进行写入；同样读操作是读取到数组的第256项时，再一次进行读取就会回到数组的第一项。是谓环形缓冲。

作用1：用作需要大量写入并且大量一次性读写的数据缓存区。比如视频的写入读取：在视频播放的时候需要不断的进行写入读取操作，而且数据一经读出就会显示出来，下次就不再需要已经读出的数据了。使用环形缓冲区可以满足这个要求，并且实现读写分别进行，而且节省了空间。

作用2：用作进程间通信，减少加锁开销

由于环形缓冲区的读写分开特性，当两个线程进行通信的时候，可以采用环形缓冲区进行交流，一个进程读取，一个进程写入，由于读写的位置不同，并不需要加锁进行并发控制，也就减少了锁的时间开销。

在read位置前面的一个字节始终不能够写入数据，但是却很轻易的区分出来缓冲区是满还是空。