## soc验证相关

## 通信概念解析

### 同步通信和异步通信

发送方和接收方按照同一个时钟节拍工作就叫同步；

发送方和接收方按照自己的节拍工作就叫异步；

同步通信中，通信双方按照统一节拍工作，所以配合很好；一般需要发送方给接收方发送信息同时发送时钟信号，接收方根据发送方给它的时钟信号来安排自己的节奏。同步通信用在通信双方信息交换频率固定，或者经常通信时。

异步通信又叫异步通知。在双方通信的频率不固定时（有时3ms收发一次，有时3天才收发一次）不适合使用同步通信，而适合异步通信。异步通信时接收方不必一直在意发送方，发送方需要发送信息时会首先给接收方一个信息开始的起始信号，接收方接收到起始信号后就认为后面紧跟着的就是有效信息，才会开始注意接收信息，直到收到发送方发过来的结束标志。

### 电平信号和差分信号

电平信号和差分信号是用来描述通信线路传输方式的，即如何在通信线路上表达1和0。

电平信号的传输线中有一个参考电平线（一般是GND），然后信号线上的信号值是由信号线电平和参考电平线的电压差决定。

差分信号的传输线中没有参考电平，所有都是信号线，1和0的表达依靠信号线之间的电压差。

电平信号的2根通信线之间的电平差异容易受到干扰，传输容易失败；差分信号不容易受到干扰因此传输质量比较稳定，现代通信一般都使用差分信号。

### 并行接口和串行接口

串行、并行主要是考虑通信线的根数，就是发送方和接收方同时可以传递的信息量的多少；

譬如在电平信号下，1根参考电平线+1根信号线可以传递1位二进制；如果我们有3根线（2根信号线+1根参考线）就可以同时发送2位二进制；如果想同时发送8位二进制就需要9根线。在差分信号下，2根线（彼此差分）可以同时发送1位二进制；如果需要同时发送8位二进制，需要16根线。

似乎并行接口比串行接口要快（串行接口一次只能发送1位二进制，而并行接口一次可以发送多位二进制）要更优秀；但是实际上串行接口才是王道，用的比较广。因为更省信号线，而且对传输线的要求更低、成本更低；而且串行时可以通过提高通信速度来提高总体通信性能，不一定非得要并行。经过这么多年发展，最终胜出的是：**异步、串行、差分**，譬如USB和网络通信。

串口通信时因为是异步通信，所以通信双方必须事先约定好通信参数，这些通信参数包括：**波特率、数据位、奇偶校验位、停止位**（串口通信中起始位定义是唯一的，所以一般不用选择）

## UART协议

### 流控制（握手）

(五) 流控制Flow control: (https://zhuanlan.zhihu.com/p/58421279)

流控制，俗话说就是“握手”。流控制的作用，在不同处理性能的设备之间，数据传输之前，接收方会以“流控制”来通知发送方，是否可以继续进行接下来的数据传输。这样的应用场景多见于计算机与低性能的微控制器通信，也可见于PC与打印机之间进行的数据传输，该特点都是接收方的接收缓存已满或处理事务较慢时，从而需要流控制来告知发送方稍后再发送。

流控制的方式分别有**软件和硬件**两种。

软件的流控制方式，在UART通信中，只需RxD、TxD、GND三根即可，数据在传输过程中，依靠代码的判断处理(xon, xoff data)，并通过收发双方进行的数据交互完成控制，在现有通信物理信号线基础上，使用控制字符(ASCII表中的0x00~0x0x1F、0x7F)完成控制指令的交互。一般在私有协议下也会定义一些特殊字符设为控制指令。

硬件的流控制方式，即在原有的RxD、TxD、GND三根信号线的基础上，再增加RTS/CTS和DTR/DSR这两组信号线。

第一组线是RTS（Request toSend）和CTS（Clear toSend）。当接收方准备好接收数据，它置**高**RTS线表示它准备好了，如果发送方也就绪，它置**高**CTS，表示它即将发送数据。

第二组线是DTR（DataTerminal Ready）和DSR（Data SetReady）。这些线主要用于Modem通信。使得串口和Modem通信他们的状态。

DTR（output）是数据接收方通知对端已经准备完毕。

DST（input）是数据发送方收到对端已经准备完毕。

一个简单的规则是DTR/DSR用于表示系统通信就绪，而RTS/CTS用于单个数据包的传输。

而在实际的应用中，硬件的方式比较少用，最大的原因是消耗硬件成本，并且在当前的MCU性能下，UART的FIFO缓存器、MCU的UART事件中断等等因素，都能够完成数据的接收和存储，并且对大多数的应用场景是完全足够，因此在很多应用中已经**很少用到硬件流控制**。

### 波特率的设置

首先，UART采用异步通信方式进行，即通信双方没有时钟同步，需要通过一定手段对通信起始以及采样进行同步，才能正确发送和解析数据。

From：[嵌入式硬件通信接口协议-UART（一）协议基础 (qq.com)](https://mp.weixin.qq.com/s?__biz=MzI1MzgwMTM4Ng==&mid=2247483819&idx=1&sn=330027d374b45d2996eb48587dc76391&chksm=e9cfa76edeb82e788040980d3c579c9a7ede68accd550ce58ebf6d240e843286bcda1605b00d&scene=21" \l "wechat_redirect)

由于UART属于异步通信，在通信过程中没有同步时钟CLK来提供给接收方，接收方也就无法同步地确定每个bit的宽度，也就无法对每个bit进行正确的采样。因此接收方必须依靠设置与发送方相同的波特率参数，这样接收方对信号管脚进行采样和解码时，才能正确判断每个bit的值是“1”还是“0”，这也就是异步通信的特点。

在各类MCU的UART配置中，常用的波特率值有：

4800Bd、9600Bd、19200Bd、115200Bd，单位Bd（代表每秒的调制数，每秒传输的bit位数）。

所以每个bit的宽度可以简单的计算为波特率的倒数。

波特率

（1）波特率（bandrate），指的是串口通信的速率，也就是串口通信时每秒钟可以传输多少个二进制位。

譬如每秒种可以传输9600个二进制位（传输一个二进制位需要的时间是1/9600秒，也就是104us），波特率就是9600。

（2）串口通信的波特率不能随意设定，而应该在一些值中去选择。

一般最常见的波特率是9600或者**115200**（低端单片机如51常用9600，高端单片机和嵌入式SoC一般用115200）。

为什么波特率不可以随便指定？第一，通信双方必须事先设定相同的波特率这样才能成功通信，如果发送方和接收方按照不同的波特率通信则根本收不到，因此波特率最好是大家熟知的而不是随意指定的。第二，常用的波特率经过长久发展，就形成了共识，大家常用就是9600或者115200。

————————————————

原文链接：<https://blog.csdn.net/whddddddddddd/article/details/125091050>

### 数据位的设置

在维基百科的介绍中，数据的位数一般可有5~9位，其中各个位数的应用场景也是有区别的：

5位常用于博多码(Baudot code)

7位常用于ASCII字符

8位则用于大多数类型的数据并且与字节(Byte)匹配

6位9位少用

**收发双方定义了相同的数据位宽后，才能对信号管脚采样时正确解码到传输的数据内容。**

**先发送最低位，最后发送最高位**

### 校验位的设置

|  |  |  |
| --- | --- | --- |
| 配置方式 | 作用 | 备注 |
| None | 无校验 | 在数据传输的帧结构中没有校验位 |
| **Odd** | **奇校验** | **数据中”1”的个数是奇数，校验位被置0；**  **数据中”1”的个数是偶数，校验位被置1。** |
| **Even** | **偶校验** | **数据中”1”的个数是奇数，校验位被置1；**  **数据中”1”的个数是偶数，校验位被置0。** |
| Mark | 校验位始终是“1” | 校验位始终是“1” |
| Space | 校验位始终是“0” | 校验位始终是“0” |

### 停止位的设置

停止位可配置成**1bit、1.5bit和2bit**三种，配置成其中一种的时候，信号管脚的电平保持对应bit时长的**高电平**。

它是一个字符数据的结束标志。可以是1位、1.5位、2位的高电平。 由于数据是在传输线上定时的，并且每一个设备有其自己的时钟，很可能在通信中两台设备之间出现了小小的不同步。因此停止位不仅仅是表示传输的结束，并且提供计算机校正时钟的机会。停止位个数越多，数据传输越稳定，但是数据传输速度也越慢。

## BUS验证

### AMBA总线协议

缩写：（Advanced Microcontroller Bus Architecture）

## 参考资料：

[Uart学习笔记 - 渝雪柒柒 - 博客园 (cnblogs.com)](https://www.cnblogs.com/daisyuer/p/12362065.html)

