公告

昵称：yooooooo
园龄：10年
粉丝：413
关注：3
+加关注

<	2025年9月	>

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

积分与排名

积分 - 1498462
排名 - 171

随笔分类

Android(66)
ARM(30)
Audio(12)
Bluetooth(3)
Bootup(39)
BPF(17)
BUS(1)
C Programming(17)
Camera(26)
CPU(100)
DevOps(5)
Display Driver(25)
Driver(29)

USB编码方式（NRZI）及时钟同步方式

阅读目录

正文

回到顶部

1.概述

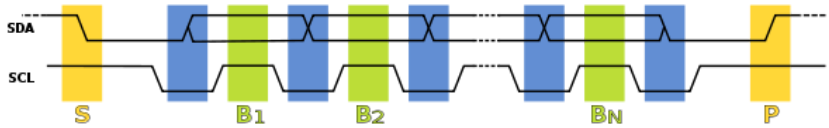
在同步通讯系统中，两个设备通讯则需要同步信号，同步信号分为时钟同步信号和自同步信号两种，时钟同步方式在通讯链路上具有时钟信号（IIC、SPI），自同步方式在通讯链路中没有同步信号（PCIE、USB），自同步方式常常适用于高速通讯系统中。

回到顶部

2.USB自同步传输

首先，USB 的数据是串行发送的，就像 UART、I2C、SPI 等等，连续的01 信号只通过一根数据线发送给接受者。但是因为发送者和接收者运行的频率不一样，信号的同步就是个问题，比如，接受者接收到了一个持续一段时间的低电平，无法得知这究竟是代表了 5 个 0 还是 1000 个 0。

一个解决办法，就是在传输数据信号的同时，附加一个时钟信号，用来同步两端的传输，接受者在时钟信号的辅助下对数据信号采样，就可以正确解析出发送的数据了，比如 I2C 就是这样做的，SDA 来传输数据，SCL 来传输同步时钟：



虽然这样解决了问题，但是却需要附加一根时钟信号线来传输时钟。因为USB没有时钟信号，有没有不需要附加的时钟信号，也能保持两端的同步呢?

有的，这就是 RZ 编码（Return-to-zero Code），也叫做归零编码。

回到顶部

3. RZ 编码（Return-to-zero Code）

RZ 编码（Return-to-zero Code），也叫做归零编码。在 RZ 编码中，正电平代表逻辑 1，负电平代表逻辑 0，并且，每传输完一位数据，信号返回到零电平，也就是说，信号线上会出现 3 种电平：正电平、负电平、零电平：

File System(34)
Git(2)
更多

随笔档案

2025年9月(12)
2025年8月(25)
2025年7月(20)
2025年6月(39)
2025年5月(32)
2025年4月(13)
2025年3月(17)
2025年2月(12)
2025年1月(35)
2024年12月(50)
2024年11月(33)
2024年10月(59)
2024年9月(53)
2024年8月(89)
2024年7月(40)
更多

阅读排行榜

1. Android WebView 的使用(超详细用法)(44344)
2. 网络流媒体协议之——RTSP协议(23346)
3. squashfs文件系统(17375)
4. C语言函数不定参数实现方式(16461)
5. Linux RCU 机制详解(15054)
6. 高通adsp架构下sensor(14424)
7. Linux audio驱动模型(13594)
8. ALSA声卡驱动的DAPM（一）-DPAM详解(12174)
9. ELF文件结构描述(11775)
10. 电池中的NTC功能是什么？【转】(11473)

评论排行榜

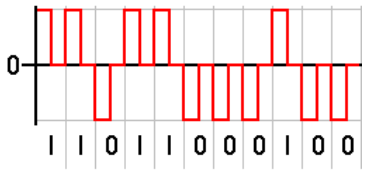
1. C语言函数不定参数实现方式(5)
2. HiGV ui代码流程(4)
3. 什么是重定位？为什么需要重定位？【转】(4)
4. 如何使用C语言的面向对象(4)
5. Linux audio驱动模型(4)
6. USB Type-C的工作原理与技术分析(3)
7. USB通信协议深入理解(3)
8. 弱符号__attribute__((weak))(3)
9. 符号解析与重定位(3)
10. I2C通讯协议(3)

推荐排行榜

1. ELF文件结构描述(9)
2. V4L2 driver -整体架构(8)
3. 网络流媒体协议之——RTSP协议(7)
4. Linux内存描述之概述--Linux内存管理(一)(7)
5. Android Recovery升级原理(6)
6. Linux audio驱动模型(6)
7. Android WebView 的使用(超详细用法)(5)
8. 弱符号__attribute__((weak))(5)
9. Linux RCU 机制详解(5)
10. VFS四大对象之二 struct inode(5)

最新评论

1. Re:【ARM Cache 及 MMU 系列文章 6 -- Cache 寄存器 CTR_EL0 CLIDR CCSIDR CSSELR 使用详解 1】
@su_su_su_cheng 您好，我在linux系统上使用lscpu看到的LLC大小为3M

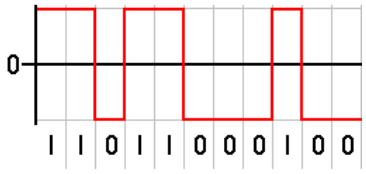


这样虽然省了时钟数据线，但是还是有缺点的，因为在 RZ 编码中，大部分的数据带宽，都用来传输“归零”而浪费掉了。

回到顶部

4. NRZ 编码（Non-return-to-zero Code）

去掉这个归零步骤，NRZ 编码（Non-return-to-zero Code）就出现了，和 RZ 的区别就是 NRZ 是不需要归零的：



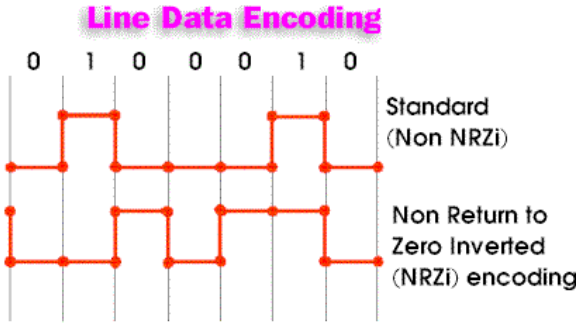
这样，浪费的带宽又回来了，不过又丧失宝贵的自同步特性了，貌似我们又回到了原点，继续往下看。

回到顶部

5.NRZI 编码（Non-Return-to-Zero Inverted Code）

NRZI 编码（Non-Return-to-Zero Inverted Code）和 NRZ 的区别就是 NRZI 用信号的翻转代表一个逻辑，信号保持不变代表另外一个逻辑。

USB 传输的编码就是 NRZI 格式，在 USB 中，电平翻转代表逻辑 0，电平不变代表逻辑1:



翻转的信号本身可以作为一种通知机制，而且可以看到，即使把 NRZI 的波形完全翻转，所代表的数据序列还是一样的，对于像 USB 这种通过差分线来传输的信号尤其方便~

现在再回到那个同步问题：

的确，NRZ 和 NRZI 都没有自同步特性，但是可以用一些特殊的技巧解决。

比如，先发送一个同步头，内容是 0101010 的方波，让接受者通过这个同步头计算出发送者的频率，然后再用这个频率来采样之后的数据信号，就可以了。

回到顶部

6.USB使用NRZI编码同步原理

在 USB 中，每个 USB 数据包，最开始都有个同步域（SYNC），这个域固定为 0000 0001，这个域通过 NRZI 编码之后，就是一串方波（NRZI 遇 0 翻转，遇 1 不变），接受者可以用这个 SYNC 域来同步之后的数据信号。

此外，因为在 USB 的 NRZI 编码下，逻辑 0 会造成电平翻转，所以接受者在接受数据的同时，根据接收到的翻转信号不断调整同步频率，保证数据传输正确。

但是，这样还是会有一个问题，就是虽然接受者可以主动和发送者的频率匹配，但是两者之间总会有误差。

假如数据信号是 1000 个逻辑 1，经过 USB 的 NRZI 编码之后，就是很长一段没有变化的电平，在这种情况下，即使接受者的频率和发送者相差千分之一，就会造成把数据采样成 1001 个或者 999 个 1了。

回到顶部

7.USB中用Bit-Stuffing来同步时钟信号

USB 对这个问题的解决办法，就是强制插 0，也就是传说中的 bit-stuffing，**如果要传输的数据中有 6个连续的 1，发送前就会在第 6 个 1 后面强制插入一个 0，让发送的信号强制出现翻转**，从而强制接受者进行频率调整。

接受者只要删除 6 个连续 1 之后的 0，就可以恢复原始的数据了。

7.1.9 Bit Stuffing

In order to ensure adequate signal transitions, bit stuffing is employed by the transmitting device when sending a packet on USB (see Figure 7-32 and Figure 7-34). A zero is inserted after every six consecutive ones in the data stream before the data is NRZI encoded, to force a transition in the NRZI data stream. This gives the receiver logic a data transition at least once every seven bit times to guarantee the data and clock lock. Bit stuffing is enabled beginning with the Sync Pattern. The data "one" that ends the Sync Pattern is counted as the first one in a sequence. Bit stuffing by the transmitter is always enforced, except during high-speed EOP. If required by the bit stuffing rules, a zero bit will be inserted even if it is the last bit before the end-of-packet (EOP) signal.

The receiver must decode the NRZI data, recognize the stuffed bits, and discard them.

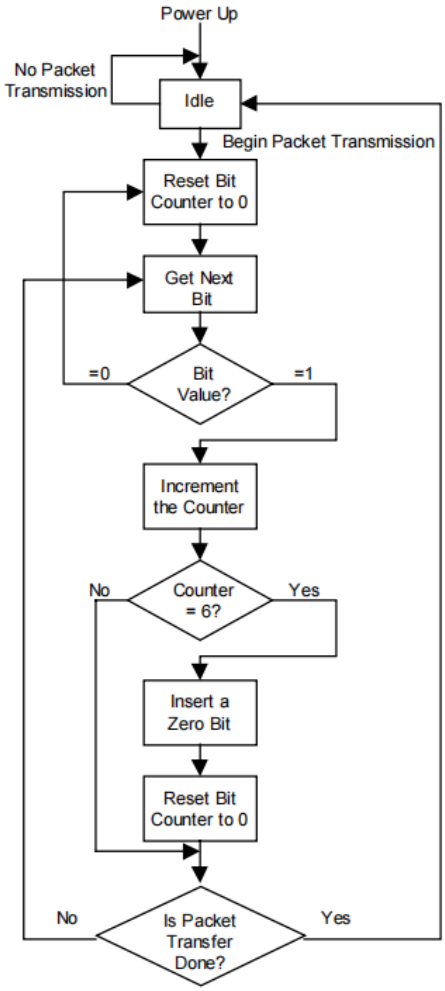


Figure 7-34. Flow Diagram for Bit Stuffing

如果您觉得阅读本文对您有帮助，请点一下**"推荐"**按钮，您的**"推荐"**将是我最大的写作动力！

分类：USB

« 上一篇： USB 同步字段中高速同步字段和低速全速同步字段的区别
» 下一篇： USB3.0与USB2.0编码方式的区别