-----------------------------------------------------------------------------------------------------------------------------------

### 串口协议

目前较为常用的串口有9针串口（DB9）和25针串口（DB25），通信距离较近时(<12m)，可以用电缆线直接连接标准RS232端口(RS422,RS485较远)，若距离较远，需附加调制解调器（MODEM）。最为简单且常用的是三线制接法，即地、接收数据和发送数据三脚相连，本文只涉及到最为基本的接法，且直接用RS232相连。

1.DB9和DB25的常用信号脚说明

　9针串口（DB9） 25针串口（DB25）

针号 功能说明 缩写 针号 功能说明 缩写

1 数据载波检测 DCD 8 数据载波检测 DCD

2 接收数据 RXD 3 接收数据 RXD

3 发送数据 TXD 2 发送数据 TXD

4 数据终端准备 DTR 20 数据终端准备 DTR

5 信号地 GND 7 信号地 GND

6 数据设备准备好 DSR 6 数据准备好 DSR

7 请求发送 RTS 4 请求发送 RTS

8 清除发送 CTS 5 清除发送 CTS

9 振铃指示 DELL 22 振铃指示 DELL

2.RS232C串口通信接线方法（三线制）

首先，串口传输数据只要有接收数据针脚和发送针脚就能实现：同一个串口的接收脚和发送脚直接用线相连，两个串口相连或一个串口和多个串口相连

· 同一个串口的接收脚和发送脚直接用线相连 对9针串口和25针串口，均是2与3直接相连；

· 两个不同串口（不论是同一台计算机的两个串口或分别是不同计算机的串口）

上面表格是对微机标准串行口而言的，还有许多非标准设备，如接收GPS数据或电子罗盘数据，只要记住一个原则：接收数据针脚（或线）与发送数据针脚（或线）相连，彼此交叉，信号地对应相接，就能百战百胜。

3.串口调试中要注意的几点：

串口调试时，准备一个好用的调试工具，如串口调试助手、串口精灵等，有事半功倍之效果； 强烈建议不要带电插拨串口，插拨时至少有一端是断电的，否则串口易损坏。

单工、半双工和全双工的定义

　如果在通信过程的任意时刻，信息只能由一方A传到另一方B，则称为单工。

如果在任意时刻，信息既可由A传到B，又能由B传A，但只能由一个方向上的传输存在，称为半双工传输。

如果在任意时刻，线路上存在A到B和B到A的双向信号传输，则称为全双工。

电话线就是二线全双工信道。 由于采用了回波抵消技术，双向的传输信号不致混淆不清。双工信道有时也将收、发信道分开，采用分离的线路或频带传输相反方向的信号，如回线传输。

奇偶校验

串行数据在传输过程中，由于干扰可能引起信息的出错，例如，传输字符‘E’，其各位为：

0100，0101=45H

D7 D0

由于干扰，可能使位变为1，这种情况，我们称为出现了“误码”。我们把如何发现传输中的错误，叫“检错”。发现错误后，如何消除错误，叫“纠错”。

最简单的检错方法是“奇偶校验”，即在传送字符的各位之外，再传送1位奇/偶校验位。可采用奇校验或偶校验。

奇校验：所有传送的数位（含字符的各数位和校验位）中，“1”的个数为奇数，如：

1 0110，0101

0 0110，0001

偶校验：所有传送的数位（含字符的各数位和校验位）中，“1”的个数为偶数，如：

1 0100，0101

0 0100，0001

奇偶校验能够检测出信息传输过程中的部分误码（1位误码能检出，2位及2位以上误码不能检出），同时，它不能纠错。在发现错误后，只能要求重发。但由于其实现简单，仍得到了广泛使用。

有些检错方法，具有自动纠错能力。如循环冗余码（CRC）检错等。

串口通讯流控制

我们在串行通讯处理中，常常看到RTS/CTS和XON/XOFF这两个选项，这就是两个流控制的选项，目前流控制主要应用于调制解调器的数据通讯中，但对普通RS232编程，了解一点这方面的知识是有好处的。那么，流控制在串行通讯中有何作用，在编制串行通讯程序怎样应用呢？这里我们就来谈谈这个问题。

1.流控制在串行通讯中的作用

这里讲到的“流”，当然指的是数据流。数据在两个串口之间传输时，常常会出现丢失数据的现象，或者两台计算机的处理速度不同，如台式机与单片机之间的通讯，接收端数据缓冲区已满，则此时继续发送来的数据就会丢失。现在我们在网络上通过MODEM进行数据传输，这个问题就尤为突出。流控制能解决这个问题，当接收端数据处理不过来时，就发出“不再接收”的信号，发送端就停止发送，直到收到“可以继续发送”的信号再发送数据。因此流控制可以控制数据传输的进程，防止数据的丢失。 PC机中常用的两种流控制是硬件流控制（包括RTS/CTS、DTR/CTS等）和软件流控制XON/XOFF（继续/停止），下面分别说明。

2.硬件流控制

硬件流控制常用的有RTS/CTS流控制和DTR/DSR（数据终端就绪/数据设置就绪）流控制。

硬件流控制必须将相应的电缆线连上，用RTS/CTS（请求发送/清除发送）流控制时，应将通讯两端的RTS、CTS线对应相连，数据终端设备（如计算机）使用RTS来起始调制解调器或其它数据通讯设备的数据流，而数据通讯设备（如调制解调器）则用CTS来起动和暂停来自计算机的数据流。这种硬件握手方式的过程为：我们在编程时根据接收端缓冲区大小设置一个高位标志（可为缓冲区大小的75％）和一个低位标志（可为缓冲区大小的25％），当缓冲区内数据量达到高位时，我们在接收端将CTS线置低电平（送逻辑0），当发送端的程序检测到CTS为低后，就停止发送数据，直到接收端缓冲区的数据量低于低位而将CTS置高电平。RTS则用来标明接收设备有没有准备好接收数据。

常用的流控制还有还有DTR/DSR（数据终端就绪/数据设置就绪）。我们在此不再详述。由于流控制的多样性，我个人认为，当软件里用了流控制时，应做详细的说明，如何接线，如何应用。

3.软件流控制

由于电缆线的限制，我们在普通的控制通讯中一般不用硬件流控制，而用软件流控制。一般通过XON/XOFF来实现软件流控制。常用方法是：当接收端的输入缓冲区内数据量超过设定的高位时，就向数据发送端发出XOFF字符（十进制的19或Control-S，设备编程说明书应该有详细阐述），发送端收到XOFF字符后就立即停止发送数据；当接收端的输入缓冲区内数据量低于设定的低位时，就向数据发送端发出XON字符（十进制的17或Control-Q），发送端收到XON字符后就立即开始发送数据。一般可以从设备配套源程序中找到发送的是什么字符。

应该注意，若传输的是二进制数据，标志字符也有可能在数据流中出现而引起误操作，这是软件流控制的缺陷，而硬件流控制不会有这个问题。

-----------------------------------------------------------------------------------------------------------------------------------

### PC9针RS232串口互连

PC机上的公插头定义如下：

1 CD Carrier Detect

2 RXD Receive Data

3 TXD Transmit Data

4 DTR Data Terminal Ready (数据中断就绪)

5 GND System Ground

6 DSR Data Set Ready (数据准备就绪)

7 RTS Request to Send

8 CTS Clear to Send

9 RI Ring Indicator

双机互联时：

Receive Data 2——3 Transmit Data

Transmit Data 3——2 Receive Data

Data Terminal Ready 4——6+1 Data Set Ready + Carrier Detect

System Ground 5——5 System Ground

Data Set Ready + Carrier Detect 6+1——4 Data Terminal Ready

Request to Send 7——8 Clear to Send

Clear to Send 8——7 Request to Send

-----------------------------------------------------------------------------------------------------------------------------------

### RS232C电平标准

EIA-RS-232C对电器特性、逻辑电平和各种信号线功能都作了规定：

在TXD和RXD上：逻辑1(MARK)=-3V～-15V

逻辑0(SPACE)=+3V～+15V

在RTS、CTS、DSR、DTR和DCD等控制线上：

信号有效（接通，ON状态，正电压）=+3V～+15V

信号无效（断开，OFF状态，负电压）=-3V～-15V

以上规定说明了RS-323C标准对逻辑电平的定义。对于数据（信息码）：逻辑“1”（传号）的电平低于-3V，逻辑“0”（空号）的电平高于+3V；对于控制信号；接通状态（ON）即信号有效的电平高于+3V，断开状态(OFF)即信号无效的电平低于-3V，也就是当传输电平的绝对值大于3V时，电路可以有效地检查出来，介于-3V～+3V之间的电压无意义，低于-15V或高于+15V的电压也认为无意义，因此，实际工作时，应保证电平在±(3～15)V之间。

TTL（晶体管-晶体管逻辑）电平（比如i2c通信）

TTL电平信号被利用的最多是因为通常数据表示采用二进制规定，+5V等价于逻辑“1”，0V等价于逻辑“0”，这被称做TTL（晶体管-晶体管逻辑电平）信号系统，这是计算机处理器控制的设备内部各部分之间通信的标准技术。

TTL电平信号对于计算机处理器控制的设备内部的数据传输是很理想的，首先计算机处理器控制的设备内部的数据传输对于电源的要求不高以及热损耗也较低，另外TTL电平信号直接与集成电路连接而不需要价格昂贵的线路驱动器以及接收器电路；再者，计算机处理器控制的设备内部的数据传输是在高速下进行的，而TTL接口的操作恰能满足这个要求。TTL型通信大多数情况下，是采用并行数据传输方式，而并行数据传输对于超过10英尺的距离就不适合了。这是由于可靠性和成本两面的原因。因为在并行接口中存在着偏相和不对称的问题，这些问题对可靠性均有影响。

TTL输出高电平>2.4V，输出低电平<0.4V。在室温下，一般输出高电平是3.5V，输出低电平是0.2V。最小输入高电平和低电平：输入高电平>=2.0V，输入低电平<=0.8V，噪声容限是0.4V。

从以上定义可以看出，这个两个电平是有不同的规定的，因此在使用到了这两种电平的接口进行通信时需经过转换处理，以便进行正确的数据传输。

-----------------------------------------------------------------------------------------------------------------------------------

### 串口流控协议

1. 发送端发RTS；
2. 接收端发出DTR
3. 发端TXD穿数据。

-----------------------------------------------------------------------------------------------------------------------------------

### SPI接口

SPI（Serial Peripheral Interface）总线是Motorola公司提出的一个同步串行外设接口，用于CPU与各种外围器件进行全双工、同步串行通讯。

SPI使用4条线：串行时钟线（SCK）、主机输入/从机输出线（MISO）、主机输出/从机输入线（MOSI）和低电平有效的使能信号线（/CS）。有的SPI接口芯片带有中断信号线(INT)或 、有的SPI接口芯片没有主机输出/从机输入数据线MOSI。

这样，仅需3、4根数据线和控制线即可扩展具有SPI接口的各种I/O器件。

它的主要特征如下：

* 全双工，串行数据同时输入和输出。
* 1.05Mbit/s的最大主机数据传输速率。
* 4种可编程的主机数据传输速率。
* 可编程串行时钟极性与相位。
* 数据传送结束标志。
* 总线竞争保护。

当一个主控机通过SPI与几种不同的串行I/O芯片相连时，必须使用每片的允许控制端/CS1, /CS2, /CS3。**未选中的芯片，输出端应处于高阻态**。如果芯片本身没有三态控制端，**则应外加三态门**。只有选中的芯片，SCK脉冲才把串行数据移入；未选中时，SCK对芯片无影响。**若没有允许控制端，则应在外围用门电路对SCK进行控制**，然后再加到芯片的时钟输入端。当然，如果SPI总线上只连接一个芯片，就不需要考虑以上的问题。

下图出了SPI典型的时序图。可以看到SPI总线数据的传输格式是高位（MSB）在前，低位（LSB）在后。对于不同的串行接口芯片，它们的时钟时序是不同的，有的是在时钟SCK的上升沿接收数据，在下降沿发送数据，而有的器件则相反，因此，需要参考具体器件的数据手册。



SPI接口的全称是"Serial Peripheral Interface",意为串行外围接口,是Motorola首先在其MC68HCXX系列处理器上定义的。SPI接口主要应用在EEPROM,FLASH,实时时钟,AD转换器,还有数字信号处理器和数字信号解码器之间。

SPI接口是在CPU和外围低速器件之间进行同步串行数据传输,在主器件的移位脉冲下,数据按位传输,高位在前,地位在后,为全双工通信,数据传输速度总体来说比I2C总线要快,速度可达到几Mbps。

SPI接口是以主从方式工作的,这种模式通常有一个主器件和一个或多个从器件,其接口包括以下四种信号：

（1）MOSI – 主器件数据输出,从器件数据输入

（2）MISO – 主器件数据输入,从器件数据输出

（3）SCLK – 时钟信号,由主器件产生

（4）/CS – 从器件使能信号,由主器件控制

-----------------------------------------------------------------------------------------------------------------------------------

### USB1.1协议的工作流程

1. 上电，10ms复位
2. HOST向DEVICE发送第一个‘要求设备描述符’的SETUP包 (80，06，00，01，00，00，40，00)。其中要求的数据长度是无效的。
3. DEVICE根据接收到的SETUP包，向HOST只发送一个包。包内包含了‘设备描述符’的实际长度。
4. HOST根据设备描述符，向DEVICE发送一个包含地址的包，设置DEVICE地址(2~127)。
5. DEVICE根据地址设置自己的状态；HOST以后会在指定地址访问DEVICE。
6. 10ms复位
7. HOST第二次发送‘要求设备描述符’的SETUP包 (80，06，00，01，00，00，12，00)。不过这次要求的数据长度是实际的数据长度。
8. DEVICE每次传送8个字节到HOST，直到把需要的数据传送完毕。
9. HOST发送第一个‘要求配置描述符’的SETUP包，要求DEVICE发送设备的配置描述符。
10. DEVICE分多次将配置描述符发送给HOST。
11. 像这样，HOST依次得到DEVICE的接口描述符和节点描述符。
12. 最后HOST向DEVICE发送设备配置SETUP包 (00，09，01，00，00，00，00，00)，DEVICE处理此事件后，将允许DEVICE的所有节点进入工作状态。

-----------------------------------------------------------------------------------------------------------------------------------

2005-06-02

### How do applications communicate with USB peripherals?

### To communicate with a USB peripheral, a PC needs two things: a device driver that knows how to communicate with the PC's USB drivers and an application that knows how to talk to the device driver. Some peripherals can use drivers that are built into Windows. Others may require a custom driver.

-----------------------------------------------------------------------------------------------------------------------------------

### How do USB peripherals communicate?

USB peripherals typically use a combination of hardware and embedded code to communicate with PCs. In this book, I show how to write the code that enables **Windows to identify a device** and **load the appropriate device driver**, as well as the code required for exchanging data with applications.

-----------------------------------------------------------------------------------------------------------------------------------

### How do I decide whether my peripheral can use bus power, or whether it needs its own supply?

A. big advantage to USB is that many peripherals can be powered entirely from the bus. Find out whether your device can use this capability and how to manage power use, especially for devices that use battery power.

-----------------------------------------------------------------------------------------------------------------------------------

### How can I be sure that my device will operate as smoothly as possible for its end users?

On the peripheral side, smooth operation requires understanding the specification's requirements and how the device can meet them. In the PC, proper operation requires a correctly structured information (INF) file that enables Windows to identify the device and software that knows how to communicate with the device as efficiently as possible.

-----------------------------------------------------------------------------------------------------------------------------------

### Benefits of USB: Automatic configuration.

When a user connects a USB peripheral to a powered system, Windows automatically detects the peripheral and loads

the appropriate software driver. The first time the peripheral connects, Windows may prompt the user to insert a disk with driver software, but other than that, installation is automatic. There's no need to locate and run a setup program or restart the system before using the peripheral.

-----------------------------------------------------------------------------------------------------------------------------------

### Benefits of USB: No user settings.

USB peripherals don't have user-selectable settings such as port addresses and interrupt-request (IRQ) lines. Available IRQ lines are in short supply on PCs, and not having to allocate one for a new peripheral is often reason enough to use USB.

-----------------------------------------------------------------------------------------------------------------------------------

### USB总线供电电流最大值：

USB可以为500mA以内的外设供电。

-----------------------------------------------------------------------------------------------------------------------------------

### USB总线电平

USB full speed(12Mbps)电平为3V(1), 0V(0); USB high speed (480Mbps)电平为400mV(1), 0V(0).

-----------------------------------------------------------------------------------------------------------------------------------

### USBVBUS负载电容规定：

In order to meet both the USB 2.0 and USB OTG (On The Go) specifications while avoiding USB supply under-voltage conditions resulting from the current limit slew rate (100mA/μS) limitations of the USB bus, the CHG\_IN bypass capacitance value must to be between 1μF and 4.7μF for the ACT5805.

-----------------------------------------------------------------------------------------------------------------------------------

### Speed of USB

USB supports three bus speeds: high speed at 480 Megabits per second, full speed at 12 Megabits per second, and low speed at 1.5 Megabits per second. Every USB-capable PC supports low and full speeds. High speed was added in version 2.0 of the specification, and requires USB 2.0-capable hardware on the motherboard or an expansion card.

These speeds are signaling speeds, or the bit rates supported by the bus. The rates of data transfer that individual devices can expect are lower. In addition to data, the bus must carry status, control, and error-checking signals. Plus, multiple peripherals may be sharing the bus. The theoretical maximum rate for a single transfer is over 53 Megabytes per second at high speed, about 1.2 Megabytes per second at full speed, and 800 bytes per second at low speed.

Why three speeds? Low speed was included for two reasons. Low-speed peripherals can often be built more cheaply. And for mice and devices that require flexible cables, low-speed cables can be more flexible because they don't require as much shielding. Full speed is comparable to or better than the speeds attainable with existing serial and parallel ports and can serve as a replacement for these.

-----------------------------------------------------------------------------------------------------------------------------------

### Flexibility

USB's four transfer types and three speeds make it feasible for many types of peripherals.

-----------------------------------------------------------------------------------------------------------------------------------

### Peripheral Support (USB Device的要求)

On the peripheral side, each USB device's hardware must include a controller chip that handles the details of USB communications. Some controllers are complete microcomputers that include a CPU and memory to store device-specific code that runs inside the peripheral. Others handle only USB-specific tasks, with a data bus that connects to another microcontroller that performs non-USB related functions and communicates with the USB controller as needed.

The peripheral is responsible for responding to requests to send and receive configuration data, and for reading and writing other data when requested. In some chips, some functions are microcoded in hardware and don't need to be programmed.

Many USB controllers are based on popular architectures such as Intel's 8051, with added circuits and machine codes to support USB.

-----------------------------------------------------------------------------------------------------------------------------------

### USB的OTG选择

1. USB OTG的选择由第五个IDpin决定。A设备(主机)的ID引脚接GND，B设备(从机)的ID引脚悬空。当OTG设备检测到接地的ID引脚时，标识默认为A设备。检测到ID引脚的为B设备。A设备提供VBUS。

2. OTG设备有省电模式，VBUS会关闭。

3. 用USB滤波器保护设备 高速USB与低速USB 针对USB工作在高速和低速两种模式，可以在外围设备那一侧，改变上拉电阻器的位置来做到。全速器件是用上拉电阻器接到数据线的D+那端，对於低速器件，则是把上拉电阻器接到数据线中D-那端上

-----------------------------------------------------------------------------------------------------------------------------------

### USB OTG说明

USB On-The-Go 在 USB 规格中增加了以下几点：

+ 双重功能设备 (设备可以扮演主机端或外设)

+ 可以切换 USB 主机与外设功能的主机协商协议 (HNP, Host Negotiation Protocol)

+ 对话请求协议 (SRP, Session Request Protocol)

+ 高低功耗之外的微功耗选择

+ 微型连接器

备注：On-The-Go 设备非但能与其他 On-The-Go 设备连接，由于內嵌 USB 主机端与 USB 外设功能，因此当它连接到一个普通 USB 外设时，它可以扮演 USB 主机端的角色，而当它连接到一个 USB 主机端時，则可以扮演 USB 外设角色。

USB OTG的硬件特点:

USB操作简单，即插即用，是免费公开的标准。而且Windows OS和许多RTOS都提供USB驱动程序，当今市场上已有许多USB设备，USB可以说是无处不在。在移动领域，手机或PDA等便携式设备，也需要作为Modem连接到PC上进行文件传输，目前许多移动终端生产厂家，都已将USB接口加到新产品上，但它们也只能作为PC外设实现与PC的数据交换。

USB OTG扩展了便携式设备之间的连通性，限定了主机(Host)功能，在传统的USB外设上增加了主机的功能，适应点对点的连接。在硬件方面，它也添加了更小的连接器和电缆，其中包括袖珍的A插头(Mini-A)、袖珍的A插槽(Mini-A)和袖珍的AB插槽(Mini-AB)，这些连接器比通常的USB标准连接器Standard-A和Standard-B小很多，更适于便携式设备。OTG也新增了Mini-A至Standard-B和Mini-A至Mini-B两种连接电缆。

OTG设备目前有两种：两用OTG设备 (Dual-BoleDevice)和外设式OTG设备(Peripheral-Only Device)。外设式OTG设备是传统的USB外设，它只能使用Mini-B插槽，不能使用Mini-AB插槽。而两用OTG设备既可作为主机，也可作为外设。作为主机的两用OTG设备不需要支持所有的USB外设，但当它与PC相连时，就只能作为外设使用。两用OTG设备都有一个Mini-AB插槽，可依照主从设定，通过Mini-A或Mini-B连接器实现点对点的连接。为了易于区分两用设备，通常Mini-A为白色，多为主控接口，插入Mini-A的设备为A-设备(A-Device)；Mini-B为黑色，多为从属接口，插入Mini-B的设备为B-设备(B-Device)；Mini-AB为灰色，多为双重角色接口。

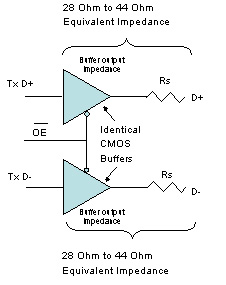
USB OTG通过五脚接口和五线电缆传输数据。除了传统的“VBUS”、“D+”、“D-”、“GND”四个针脚外，第五个针脚为“ID”脚，ID脚决定了初始化的主从角色：在Mini-A插头中，ID短路接地，所连设备被初始化为主；在Mini-B插头中，ID悬空，所连设备被初始化为从。在软件控制下，主从角色也可以利用主设备转换协议 (HNP)进行互换。

主设备可提供低功率输出，最小为8mA的电流到USB电缆，当不传输时可关掉VBUS。在VBUS关闭时，如从属设备需要使用总线，则可请求主设备开启VBUS，为VBUS重新供电。低功耗可支持USB设备在电池供电的情况下正常工作，延长电池寿命，节省能源。

w-----------------------------------------------------------------------------------------------------------------------------------

### USB串联匹配电阻

由于信号引脚的特征输出阻抗实际上比正确匹配负载所需的总体阻抗小很多，所以必须在这些引脚上串接电阻。许多收发器供应商建议串接电阻阻抗为33Ω。而实际上，上拉电阻阻值只要能保证传输线一侧测出的总阻抗在28-44Ω范围内就可以.



在上行位的收发器有两种不同配置，一个用于低速传输，一个用于全速或高速传输。当配置全速数据传输时，1.5kΩ ± 5%的上拉电阻会在D+ 线和3.3V之间进行连接。如果收发器有提供3.3V上拉电压，最好使用这个电压。

当进行低速数据传输配置时，外设需要设定为发出"Differential 0" 状态信号。在此情况下，"Differential 0" 状态定义为当D+ 小于VOL (max)，D- 大于VOH (min)。这是通过在D- 线和3.3V之间接上1.5kΩ ± 5% 上拉电阻来实现的。同样地，许多收发器也因此提供上拉电压。使用此配置时，外设一旦连接到主机上，1.5kΩ上拉电阻便会抵消15kΩ下拉电阻的反偏作用，

-----------------------------------------------------------------------------------------------------------------------------------

### I2S 的定义

I2S, or Inter-IC Sound, is an electrical bus interface standard used for connecting digital audio devices together. It is most commonly used to carry PCM information between the CD transport and the DAC in a CD player. The I2S bus separates clock and data signals, resulting in a very low jitter connection. Jitter can cause distortion in a digital-to-analog converter. The bus consists of three lines, a clock line, a word select line, and a multiplexed data line.

-----------------------------------------------------------------------------------------------------------------------------------

### USB和IEEE 1394接口的比较

两者的传输速率不同。USB的传输速率现在只有12Mbps/s，只能连接键盘、鼠标与麦克风等低速设备，而IEEE1394可以使用400Mbap/s，可以用来连接数码相机、扫描仪和信息家电等需要高速率的设备。而后来，推出了USB2.0，虽然有所赶上IEEE1394，但是1394的流量还可以增加至1G。

两者的结构不同。USB在连接时必须至少有一台电脑，并且必须需要HUB来实现互连，整个网络中最多可连接127台设备，每个外设间距离（线缆长度）可达5米。IEEE1394并不需要电脑来控制所有设备，也不需要HUB，IEEE1394可以用网桥连接多个IEEE1394网络，也就是说在用IEEE1394实现了63台IEEE1394设备之后也可以用网桥将其他的IEEE1394网络连接起来，达到无限制连接。

两者的智能化不同。IEEE1394网络可以在其设备进行增减时自动重设网络。USB是以HUB来判断连接设备的增减了。两者的应用程度不同。现在USB已经被广泛应用于各个方面，几乎每台PC主板都设置了USB接口，USB2.0也会进一步加大USB应用的范围。IEEE1394现在只被应用于音频、视频等多媒体方面。以下是几种数据接口的列表比较：

IEEE1394接口详细说明

IEEE1394总线是一种目前为止最快的高速串行总线，最高的传输速度为400Mbps/s。对于各种需要大量带宽的设备提供了专门的优化，接口可以同时连接63个不同设备，IEEE1394同USB一样，支持带电插拨设备。IEEE1394支持即插即用，现在的WIN98 SE、WIN2000、WIN ME、WIN XP都对IEEE1394支持的很好，在这些操作系统中用户不用再安装驱动程序，也能使用IEEE1394设备。

火线（IEEE1394）支持的传输速率有100Mbps,200Mbps，400Mbps，将来会提升到800Mbps，1Gbps，1.6Gbps。不需要控制器，可以实现对等传输，最大连线4.5米，大于4.5米可采用中继设备支持，同样支持即插即用。火线是目前唯一支持数字摄录机的总线。IEEE1394既可作为外部总线，又可成为内部总线使用，不过由于已经有了PCI这样历史悠久的总线存在，而且现在PCI正向64位过渡，各厂商并不愿意做总线上的调整改动，所以市面上的IEEE1394是作为外部总线连接外设使用。 它的缺点主要表现于两个方面：应用少。现在支持IEEE1394的设备也不太多，只有一些数码相机与MP3等一些使用高带宽的设备使用IEEE1394。其它的设备其实也用不了那么高的带宽。IEEE1394总线需要占用大量的资源，所以需要高速度的CPU。

-----------------------------------------------------------------------------------------------------------------------------------

### USB枚举过程

（1）集线器检测新设备

主机集线器监视着每个端口的信号电压，当有新设备接入时便可觉察。（集线器端口的两根信号线的每一根都有15kΩ的下拉电阻，而每一个设备在D+都有一个1.5kΩ的上拉电阻。当用USB线将PC和设备接通后，设备的上拉电阻使信号线的电位升高，因此被主机集线器检测到。）

（2）主机知道了新设备连接后

每个集线器用中断传输来报告在集线器上的事件。当主机知道了这个事件，它给集线器发送一个Get\_Status请求来了解更多的消息。返回的消息告诉主机一个设备是什么时候连接的。

（3）集线器重新设置这个新设备

当主机知道有一个新的设备时，主机给集线器发送一个Set\_Feature请求，请求集线器来重新设置端口。集线器使得设备的USB数据线处于重启（RESET）状态至少10ms。

（4）集线器在设备和主机之间建立一个信号通路

主机发送一个Get\_Status请求来验证设备是否激起重启状态。返回的数据有一位表示设备仍然处于重启状态。当集线器释放了重启状态，设备就处于默认状态了，即设备已经准备好通过Endpoint 0 的默认流程响应控制传输。即设备现在使用默认地址0x0与主机通信。

（5）集线器检测设备速度

集线器通过测定那根信号线（D+或D-）在空闲时有更高的电压来检测设备是低速设备还是全速设备。（全速和高速设备D+有上拉电阻，低速设备D-有上拉电阻）。

以下，需要USB的firmware进行干预

（6）获取最大数据包长度

PC向address 0发送USB协议规定的Get\_Device\_Descriptor命令，以取得却缺省控制管道所支持的最大数据包长度，并在有限的时间内等待USB设备的响应，该长度包含在设备描述符的bMaxPacketSize0字段中，其地址偏移量为7，所以这时主机只需读取该描述符的前8个字节。注意，主机一次只能列举一个USB设备，所以同一时刻只能有一个USB设备使用缺省地址0。

以下操作雷同，不同操作系统设定时延是不一样的，比如说win2k大概是几毫秒，如果没有反应就再发送一次命令，重复三次。

（7）主机分配一个新的地址给设备

主机通过发送一个Set\_Address请求来分配一个唯一的地址给设备。设备读取这个请求，返回一个确认，并保存新的地址。从此开始所有通信都使用这个新地址。

（8）主机向新地址重新发送Get\_Device\_Descriptor命令，此次读取其设备描述符的全部字段，以了解该设备的总体信息，如VID，PID。

（9）主机向设备循环发送Get\_Device\_Configuration命令，要求USB设备回答，以读取全部配置信息。

（10）主机发送Get\_Device\_String命令，获得字符集描述（unicode），比如产商、产品描述、型号等等。

（11）此时主机将会弹出窗口，展示发现新设备的信息，产商、产品描述、型号等。

（12）根据Device\_Descriptor和Device\_Configuration应答，PC判断是否能够提供USB的Driver，一般win2k能提供几大类的设备，如游戏操作杆、存储、打印机、扫描仪等，操作就在后台运行。但是Win98却不可以，所以在此时将会弹出对话框，索要USB的Driver。

（13）加载了USB设备驱动以后，主机发送Set\_Configuration（x）命令请求为该设备选择一个合适的配置(x代表非0的配置值)。如果配置成功，USB设备进入“配置”状态，并可以和客户软件进行数据传输。

此时，常规的USB完成了其必须进行的配置和连接工作。查看注册表，能够发现相应的项目已经添加完毕，至此设备应当可以开始使用。不过，USB协议还提供了一些用户可选的协议，设备如果不应答，也不会出错，但是会影响到系统的功能

-----------------------------------------------------------------------------------------------------------------------------------

### What is AMBA

AMBA is a well-established specification for processor bus architectures. It stands for Advanced Microcontroller Bus Architecture. The AMBA architecture standardizes the on-chip connection of different IPs and thus enables IP reusability.

AMBA specification defines three defferent buses: ASB, AHB and APB. AHB stands for Advanced High-performance Bus. This is the premiere bus used for high performance system modules such as memory controllers and Interrupt controllers. ASB stands for Advanced System Bus. This is an older version which has been superseded by AHB. Oki's ARM micros always use the AHB bus. APB stands for Advanced Peripheral Bus. It is a simple lower performance and low power bus used for low speed peripherals. Macrocells designed to interface with AMBA can be seen as building blocks which can be reused in future designs and mixed and matched in different combinations to realize complex systems in a shorter period of time.

-----------------------------------------------------------------------------------------------------------------------------------

### 串口偷电

4，7脚加二极管并联接到7805上，小功率使用应该没问题。好像总电流不能超过6.5mA，否则pc会死机重起。

-----------------------------------------------------------------------------------------------------------------------------------

### LCD RGB和CPU接口讨论

RGB是专业绘图用的，不是专业显卡看不出区别来,MPU接口方式：显示数据写入DDRAM，常用于静止图片显示。下面我们来看看什么是RGB接口，

RGB接口方式：显示数据不写入DDRAM，直接写屏，速度快，常用于显示视频或动画用。

只有TFT模块才有RGB接口。

MPU接口方式：LCD只有数据线，读写线，片选，复位。

RGB接口方式：LCD有数据线，行场扫描线，clk线。

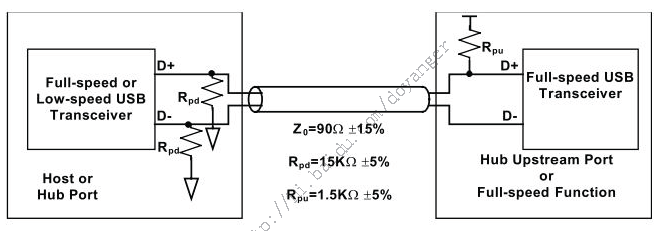
最主要的区别是RGB接口需要行场扫描线，clk线。而MPU接口不需要。

RGB模式：大屏采用较多的模式，数据位传输也有6位，16位和18位之分。其中包含4个重要的控制信号VSYNC、HSYNC、DCLK 和VDEN，分别用于帧、行、像素的数据传输。剩下就是数据线。它的优缺点正好和MCU模式相反。

-----------------------------------------------------------------------------------------------------------------------------------

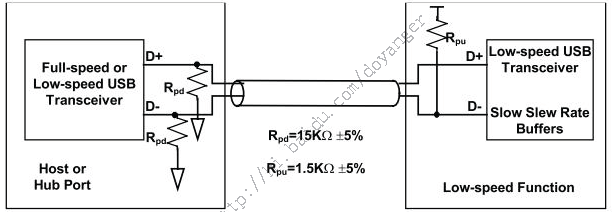
### USB设备识别

我们知道USB2.0向下兼容USB1.x，即高速2.0的hub能支持所有的速度类型的设备，而USB1.x的hub不能支持高速设备（High Speed Device）。因此，如果高速设备挂到USB1.x的hub上， 那该设备只能工作在全速模式下。不管是hub还是设备（device），对于速度的区分是非常 重要的，否则，后续的通信根本无法进行。  
  
**全速和低速识别**  
　　  
　　根据规范，全速（Full Speed）和低速（Low Speed）很好区分，因为在设备端有一个1.5k的上拉电阻，当设备插入hub或上电（固定线缆的USB设备）时， 有上拉电阻的那根数据线就会被拉高，hub根据D+/D-上的电平判断所挂载的是全速设备还是低速设备。如下两图：



USB全 速设备上电连接

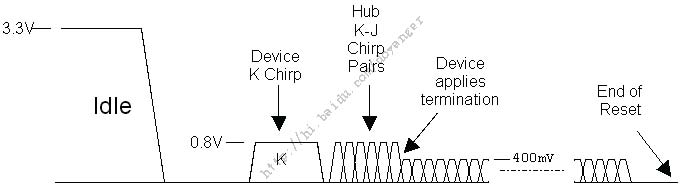
（Full-speed Device Cable and Resistor Connections）



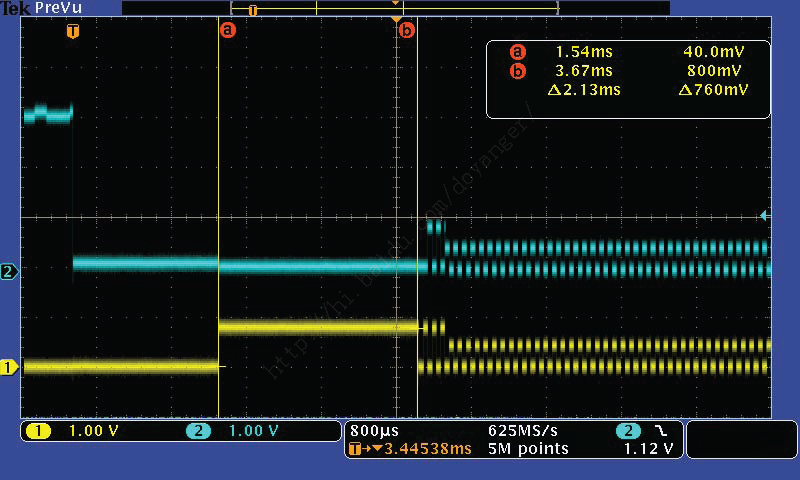
USB低速设备上电连接

（Low-speed Device Cable and Resistor Connections）

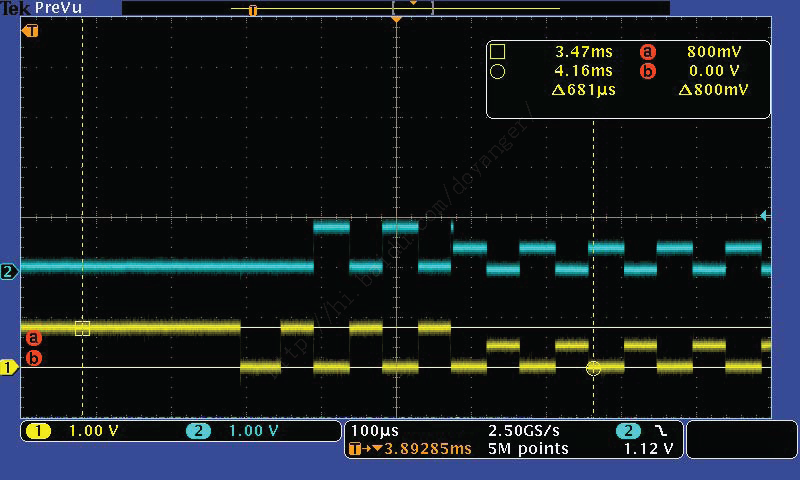
**高速识别**  
　　  
　　USB全速/低速识别相当简单，但USB2.0，USB1.x就一对数据线，不能像全速/低速那样仅依靠数据线上拉电阻位 置就能识别USB第三种速度：高速。因此对于高速设备的识别就显得稍微复杂些。  
  
　　高速设备初始是以一个全 速设备的身份出现的，即和全速设备一样，D+线上有一个1.5k的上拉电阻。USB2.0的hub把它当作一个全速设备，之后，hub和设备通过一系列握手信号确认双方的身份。在这里对速度的检测是双向的，比如高速的hub需要检测所挂上来的设备是高速、全速还是低速，高速的设备需要检测所 连上的hub是USB2.0的还是1.x的，如果是前者，就进行一系列动作切到高速模式工作，如果是后者，就以全速模式工作。  
  
　　下图展示了一个高速设备 连到USB2.0 hub上的情形：



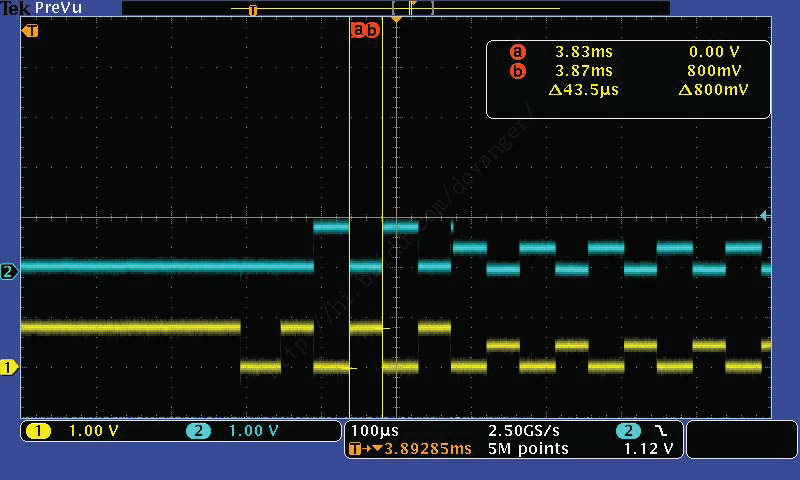
　　hub检测到有设备插入/上电时，向主机 通报，主机发送Set\_Port\_Feature请求让hub复位新插入的设备。设备复位操作是hub通过驱动数据线到复位状态SE0(Single- ended 0，即D+和D-全为低电平)，并持续至少10ms。  
  
　　高速设备看到复位信号后，通过内部的电流源向D-线持续灌大小 为17.78mA电流。因为此时高速设备的1.5k上拉电阻还未撤销，在hub端，全速/低速驱动器形成一个阻抗为 45欧姆(Ohm)的终端电阻，2电阻并联后仍是45欧姆左右的阻抗，所以在hub端看到一个约800mV的电压（45欧姆\*17.78mA），这就是Chirp K信号。Chirp K信号的持续时间是1ms~7ms。  
  
　　在hub端，虽然下达 了复位信号，并一直驱动着SE0，但USB2.0的高速接收器一直在检测Chirp K信号，如果没有Chirp K信号看到，就继续复位操作,直到复位结束，之后就在全速模式下操作。如果只是一个全速的hub，不支持高速操作，那么该hub不理会设备发送的Chirp K信号，之后设备也不会切换到高速模式。  
  
　　设备发送的Chirp K信号结束后100us内，hub必须开始回复一连串的KJKJKJ....序列，向设备表明这是一个USB2.0的hub。这里的KJ序列是连续的，中间不能间断，而且每个K或J的持续时间在40us~60us之间。KJ序列停止后的100~500us内结束复位操作。hub发送Chirp KJ序列的方式和设备一样，通过电流源向差分数据线交替灌17.78mA的电流实现。  
  
　　再回到设备端来。设备检测到6个hub发出的Chirp信号后（3对KJ序列），它必须在500us内切换到高速模式。切换动作有：  
1. 断开1.5k的上拉电阻。  
2. 连接D+/D-上的高速终端电阻（high-speed termination），实际上就是全速/低速差分驱动器。  
3. 进入默认的高速状态。  
　　执行1，2两步后，USB信号线上看到的现象就 发生变化了：hub发送出来的Chirp KJ序列幅值降到了原先的一半，400mV。这是因为设备端挂载新的终端电阻后，配上原先hub端的终端电阻，并联后的阻抗是22.5欧姆。400mV就是由17.78mA\*22.5Ohm得来。以后高速操作 的信号幅值就是400mV而不像全速/低速那样的3V。  
  
　　至此，高速设备与USB2.0 hub握手完毕，进行后续的480Mbps高速信号通信。  
  
最后附上几幅实际USB高速识别的示波器抓 图，图中蓝色信号是D+,黄色信号是D-。  
  
1.数据线D+在T点之前挂上1.5K电 阻，在T点被host拉成SE0状态。在近2ms后，设备发送第一个Chirp K，向host通知说：我是一个高速设备，如果可能，请用高速方式与我通信。其幅度是800mV（17.78mA \* (45Ohm ||1.5kOhm) = 800mV，见上文解释）。在这里，Chirp K的持续时间是2.13ms（a，b两点之间）。



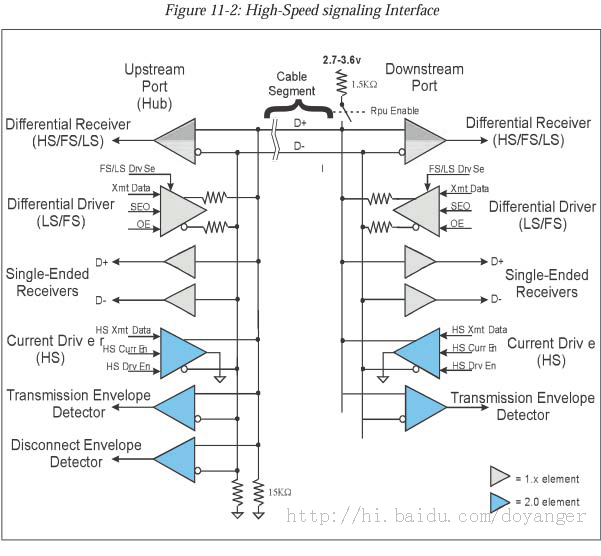
2.这幅图显示了host发出的chirp KJ信号的幅度，头几个KJ是800mv（a，b之间），随后的是400mV。图中可以看到设备在收到第三个chirp J（蓝色短条）后马上把1.5k电阻取消，导致chirp J的幅值下降到400mV。(17.78mA \* (45Ohm ||45Ohm) = 17.78mA \* 22.5Ohm = 400mV)



3.量测了一个chirp J的宽度：43.5us。



 最 后附上一张来自Don Anderson的USB System Architecture里的USB HS接口图：



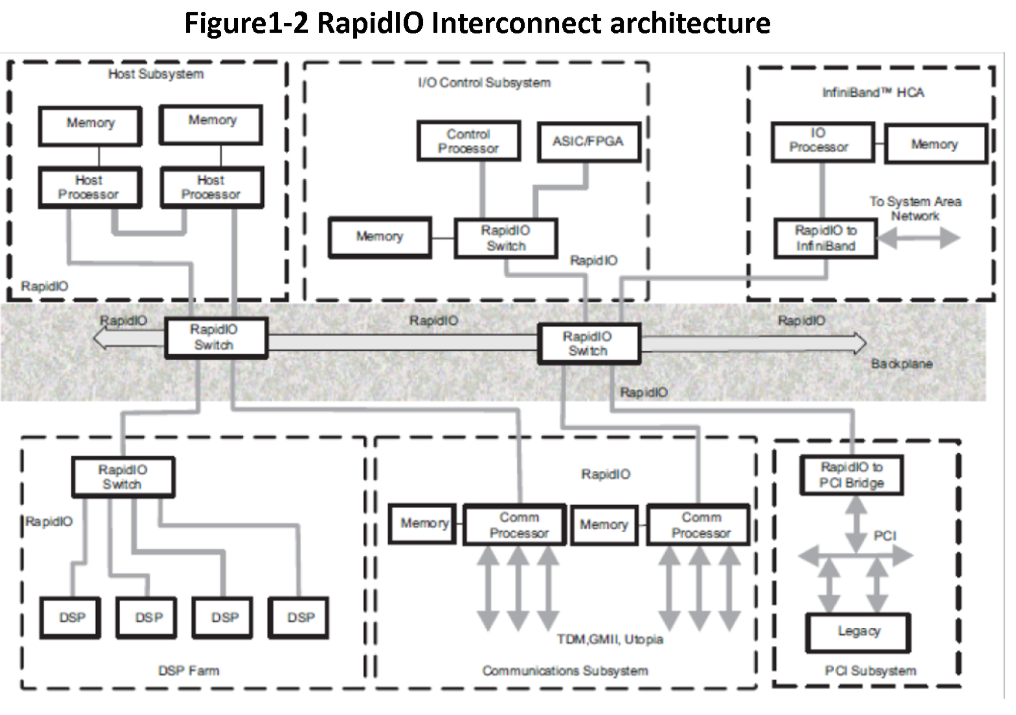
-----------------------------------------------------------------------------------------------------------------------------------

### SRIO接口

1. 简介:

SRIO是面向嵌入式系统开发提出的高可靠、高性能、基于包交换的新一代高速互联技术，已于2004年被国际标准化组织(ISO)和国际电工协会(IEC)批准为ISO/IECDIS 18372标准。SRIO则是面向串行背板、DSP和相关串行数据平面连接应用的串行RapidIO接口。串行RapidIO包含一个3层结构的协议，即物理层、传输层、逻辑层。物理层定义电气特性、链路控制、低级错误管理、底层流控制数据；传输层定义包交换、路由和寻址机制；逻辑层定义总体协议和包格式。可以实现最低引脚数量，采用DMA传输，支持复杂的可扩展拓扑，多点传输；可选的1.25 Gbps、2.5 Gbps、3.125 Gbps和5Gbps四种速度能满足不同应用需求，是未来十几年中嵌入式系统互联的最佳选择之一。

1. SRIO结构
   1. 层次结构: SRIO模块由三层构建而成。逻辑层：确定终端处理传输的协议，包括包的格式。包含终端处理传输（transaction）的必要信息，如传输类型、大小、物理地址。传输层 定义了在系统中正确路由信息包的寻址方案。包含系统中终端相互传输包（packet）的信息，如寻址。物理层 包含设备级的接口信息，如电气特性、错误管理数据和基本的流量控制数据。包含物理设备之间相互传递包（packet）时所需的信息，如电接口，流的控制。传输层与逻辑层和物理层是上下兼容的
   2. RapidIO互联结构：定义为一个独立于物理层实现的分组交换协议。Figure1-2显示的是SRIO互联系统



* 1. 物理层 1x/4x LP-Serial（长浮点串口） 规格：

现在有两种SRIO贸易协会认可的物理层规格:

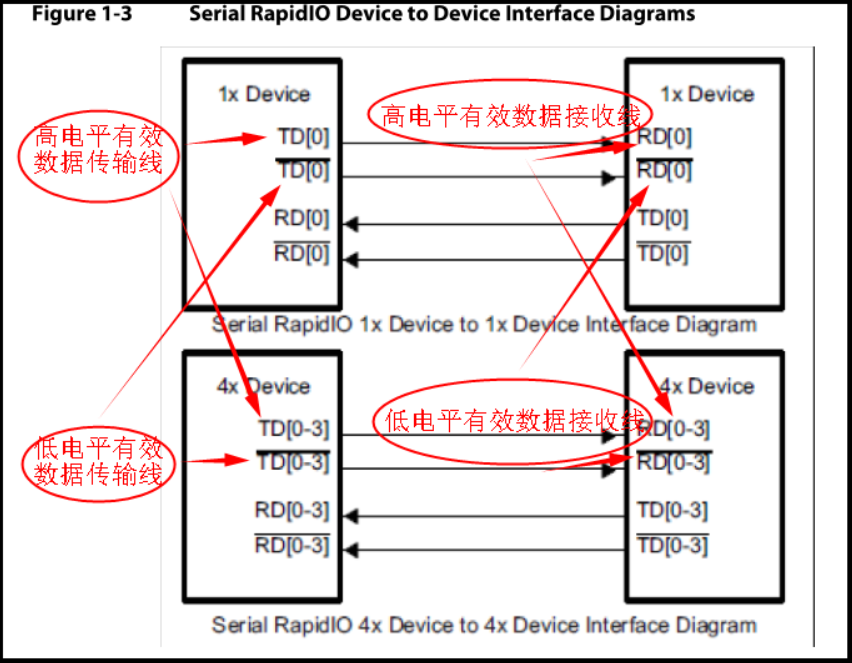
1、 8/16 LP-LVDS

2、 1x/4x LP-Serial

第一种规格是点对点同步时钟源DDR接口；第二种规格是点对点，交流耦合，时钟恢复接口。而且两种规格不兼容。SRIO遵从第二种规格，即1x/4x LP-Serial规格，SRIO中的串行/解串技术也是由这种规格分配的。该规格适用于4个频率点，即1.25,2.5,3.125和5Gbps，这定义了每个I/O差分对的总带宽。有一个8位/ 10位编码方案，确保时钟恢复电路的充足的数据转换。由于8位/ 10位编码方案的开销，每个差分对各自的有效数据带宽是1，2，2.5，4 Gbps。SRIO只同时为1X和4X的port指定频率。一个1X port被定义为一个TX（transanction）和一个RX（receive）的差分对。类似于一个IO口的差分对。4 X port就是4个1X的组合。一个4X port也可以被配置为4个1X port。

SRIO提供了支持从1G到16G带宽的可升级接口。

Figure1-3显示了怎样连接两个1X设备（或两个4X设备），一个设备的positive transmit data line (TDx) 【高电平有效传输数据线TDx】和另一个设备的positive receive data line (RDx)相连，低电平有效传输数据线 TDx和低电平有效接收数据线RDx相连。



* 1. s

1. SRIO中RapidIO支持的功能：

1、遵从RapidIO互联协议REV2.1.1；

2、LP- Serial协议REV2.1.1；

3、4个可操作的1Xports；

　　2个可操作的2Xports，使用2Xport时可以用2个1Xport或1个2Xport；

　　1个可操作的4Xport，使用4Xport时可以用1个4Xport或4个1Xport；

4、在TI SerDes（串行/解串器）集成时钟恢复功能；

5、运行不同的port时使用不同的波特率，但是只支持倍频，比如2.5G和5G是支持的，但是3.125G和5G是不支持的；

6、硬件错误处理，包括CRC（循环冗余校验(Cyclic Redundancy Check, CRC)）；

7、差分CML（Current-Mode Logic 电流型逻辑（电路））信号支持AC和DC耦合；

8、能对不用的port进行掉电（powerdown）；

9、支持1.25, 2.5, 3.125, and 5 Gbps rates；

10、支持读，写，写/反馈，流数据的写入，out-going Atomic(什么意思？外向原子？),维护操作；

11、向CPU产生中断（门铃包和内部调度）；

12、支持8位和16位的设备ID；

13、支持接收34位的地址；

14、支持产生34位，50位和66位地址；

15、支持数据类型：位，半字，字，双字；

16、是BIG-ENDIAN（big endian是指低地址存放最高有效字节（MSB），与之对应的是small endian）模式的；

17、DirectIO传输；

18、消息（Message）传递；

19、数据payload最高256位；

20、单个消息最多包含16个包；

21、时钟域切换支持Elastic Store FIFO（FIFO弹性存储）；

22、Short Run and Long Run compliant（不太懂，百度得到的结果只有经济学中的短期和长期，long run指的是不变成本和可变成本都能快速可变的时期，随着市场的自身的调节，市场的供求能达到平衡；short run是不变成本很难改变，商品的价格与不变成本具有粘滞性的时期；short和long是相对而言的，不一定short就比long小。T-T我不是在看DSP吗T\_T？）

23、支持错误管理拓展；

24、支持拥挤（多个核同时访问同一地址）控制扩展；

25、支持多点ID；

26、支持长短控制符；

27、支持IDLE1和IDLE2；（IDLE为系统挂起命令，是为了节省系统消耗）

28、基于优先级和频率的严格优先级段交至协议单元；

以下是不支持的功能

1、不支持全局内存共享协议（GSM）；

2、不兼容8/16 LP-LVDS规格；

3、不支持RapidIO原子操作目标；

1. 规格：

支持的两个REV2.1.1规格上面已经提到过了，不再赘述。

serial RapidIO AC specification（串行RapidIO交流协议）两种驱动，称为long run和short run。

long run协议适用于至少50cm长的长底板应用和至少两个连接设备的应用；

short run协议适用于低功率应用，它通常用于同一个板子上或者底板较短的连接上。

这两类规格的不同在于驱动的Vod。（什么是Vod，很明显不是点播技术）

Target atomic operations(目标原子操作)包括增长，减少，测试和交换，置位和清除操作在L2缓存或寄存器中是不支持的。

对外部设备的Atomic request operations（原子请求操作）是支持的。

（atomic到底是个啥啊？）

1. s

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------------