USB 基础知识概论

版本：**0.5**

作者：crifan

邮箱：green-waste (at) 163.com

关键字

USB，概论

版本历史

版本号	时间	内容更新
0.5	2011-10-06	USB 基础知识概论

目录

图表

表格

缩写

缩写	全称	详细解释
EHCI	Enhanced Host Controller Interface	
NRZ	Non-Return-to-Zero	
NRZI	Non-Return-to-Zero Inverted	
OHCI	Open Host Controller Interface	
RZ	Return-to-Zero	
SYNC	Synchronize	
UHCI	Universal Host Controller Interface	
USB	Universal Serial Bus	通用串行总线
xHCI	eXtensible Host Controller Interface	

1. 正文之前

1.1. 此文目的

由于 USB 所涉及的知识太多，如果想要在一篇文章里，把 USB 的方方面面的内容，都解释的很清楚，那几乎是不可能的。
因此，此文目的，不是为了把 USB 的所有的事情都写出来，而是让对 USB 不懂的人，通过此文档，能对 USB 有个基本的认识，并且搞懂 USB 世界中的基本的术语的含义。
即，此文目的，是为了给不熟悉 USB 的人，一个总体的概述，以及解释一些必要的 USB 方面的基本知识。
这样，如果想要更细节的去了解 USB 的知识，也知道从哪里入手，以及如何去找相关资料区学习了。

1.2. 关于一些 USB 方面的文档

USB 很复杂，所以，如果能把复杂的东西解释的清楚的，不是很容易。
而且由于 USB 涉及知识面也很广，所以也很难简短地描述清楚。

1.2.1. 大而全的 USB 英文资料

对于众多的现存的 USB 的文章或书籍，我所见过的，能把 USB 讲的透彻的，算是
英文资料：《USB Complete》，中文翻译为《USB 大全》，目前最新版本是第四版。
其主页是：
http://www.lvr.com/usbc.htm
网上也可以找到盗版的电子版的，第三版的有中文翻译，第四版的只有英文原版。

1.2.2. 简明扼要的 USB 英文资料

另外，简明扼要地，把 USB 讲解的很清楚的，我觉得算是《USB in a Nutshell》了，网上随便都可以搜到此文的 pdf 版本，比如：
USB in a Nutshell - Making sense of the USB standard
http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf
而此文，也是主要根据此贴而来，基本可以算是《USB in a Nutshell》的中文版吧，然后另外又添加了一些必要的知识，以求把 USB 讲解的更加清楚。

1.2.3. 全系列的介绍 Linux 下的 USB 中文资料

另外，关于中文方面的资料，觉得写的比较全，解释的比较清楚的，算是 fudan_abc 写的《Linux 那些事儿系列》，是一个系列的，好多个文档的。详细资料，已整理放到这里了：
【很好的学习 Linux 驱动的教材】Linux 那些事儿系列[全][pdf]
http://bbs.chinaunix.net/thread-1977195-1-1.html

1.3. 声明

由于本人知识水平有限，错误在所难免，欢迎指正。

此文欢迎拷贝传播，但是所有权本人独有，未经许可，严谨用于其他商业等用途。

更多建议，意见，吐槽，都可以联系偶：green-waste (at) 163.com

2. USB 的来龙去脉

2.1. USB 是什么

USB 是 Universal Serial Bus 的缩写，中文译为通用串行总线。

所以，从字面意思上，善于思考的人，就问有疑问：

那么与此 USB 相比，其他还有哪些非串行的总线，以及和此通用的串行总线来说，其他还有哪些相对“不通用”的串行总线呢？

对此，借用《USB Complete》里面所总结的，关于 USB 和其他接口的区别，来解释一下：

图表 1 USB 与其他总线的异同

Table 1-1: USB is more flexible than other interfaces, which often target a specific use.

Interface	Type	Number of Devices (including PC) (max.)	Distance (max. ft)	Speed (max. bps)	Typical Use
USB 3.0	dual simplex serial	127 (per bus)	9 (typical) (up to 49 with 5 hubs)	5 G	Mass storage, video
USB 2.0	half duplex serial	127 (per bus)	16 (98 ft. with 5 hubs)	1.5M, 12M, 480M	Keyboard, mouse, drive, speakers, printer, camera
eSATA	serial	2 (port multiplier supports 16)	6	3G	Drives
Ethernet	serial	1024	1600	10G	General network communications
IEEE-1394b (FireWire 800)	serial	64	300	3.2G	Video, mass storage
IEEE-488 (GPIB)	parallel	15	60	8M	Instrumentation
I²C	synchronous serial	40	18	3.4M	Microcontroller communications
Microwire	synchronous serial	8	10	2M	Microcontroller communications
MIDI	serial current loop	2 (more with flow-through mode)	50	31.5k	Music, show control
Parallel Printer Port	parallel	2 (8 with daisy-chain support)	10–30	8M	Printers, scanners, disk drives
RS-232 (EIA/TIA-232)	asynchronous serial	2	50–100	20k (115k with some hardware)	Modem, mouse, instrumentation
RS-485 (TIA/EIA-485)	asynchronous serial	32 unit loads (some chips allow up to 256 devices)	4000	10M	Data acquisition and control systems
SPI	synchronous serial	8	10	2.1M	Microcontroller communications

从上述表格中，表面上看，好像也没看出 USB 相对其他接口，有多么特别明显的优点，而只是看到在某些参数上，比其他某些接口参数更高，而在别的某些参数上，比其他接口低。关于细节的区别，不是此讨论的重点，此处，我们至少可以看出，除了 USB 接口外，目前已存在的接口，还是很多的，而且各种接口实际上从硬件上也是形状各异，互相也都有自己的应用领域，而且无法兼容。基于此背景，才有下面的解释，以说明，为何会出现这么个 USB 接口。

此处，简单的说，USB 就是一种接口，一种总线。

2.2. 为何要有 USB

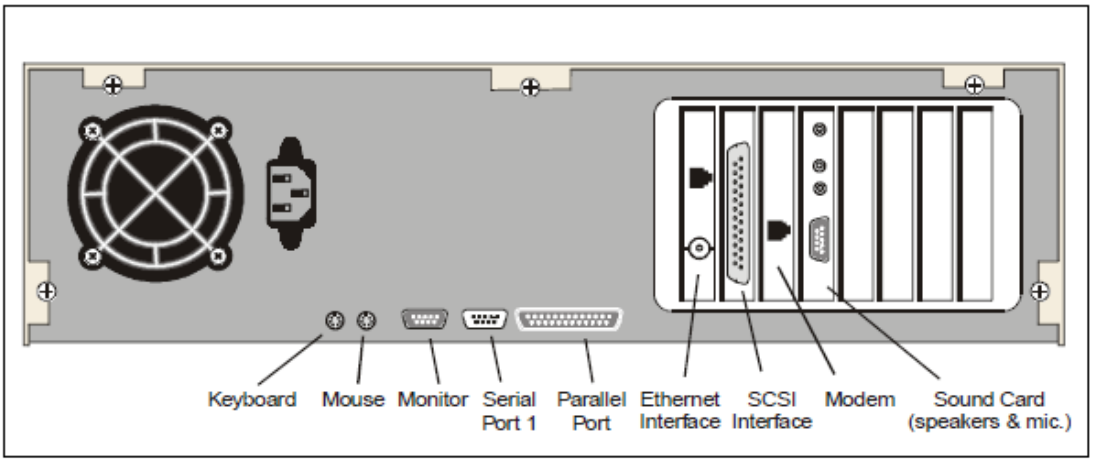
上面已经提到了，在 USB 出现之前，其实计算机领域中，已经存在众多的接口，而且不同的应用领域，已有一些相对来说是广泛使用的各种接口了。

但是，对于计算机等使用的普通用户来说，由于接口太多，而容易被搞得晕头转向。再加上各个接口从硬件形状和软件配置也都不一样，导致不兼容，为了不同的应用，而要配置多种不同的硬件接口，设置对于有些硬件接口来说，还需要手动去配置一些更细节的参数。

关于 USB 出现之前，计算机领域中的接口太多太繁杂，可以用下面这张，关于 PC 机箱背后的接口的图片来说明：

图表 2 PC 机箱后面的众多接口

Figure 1-2: Connectors at Backplane

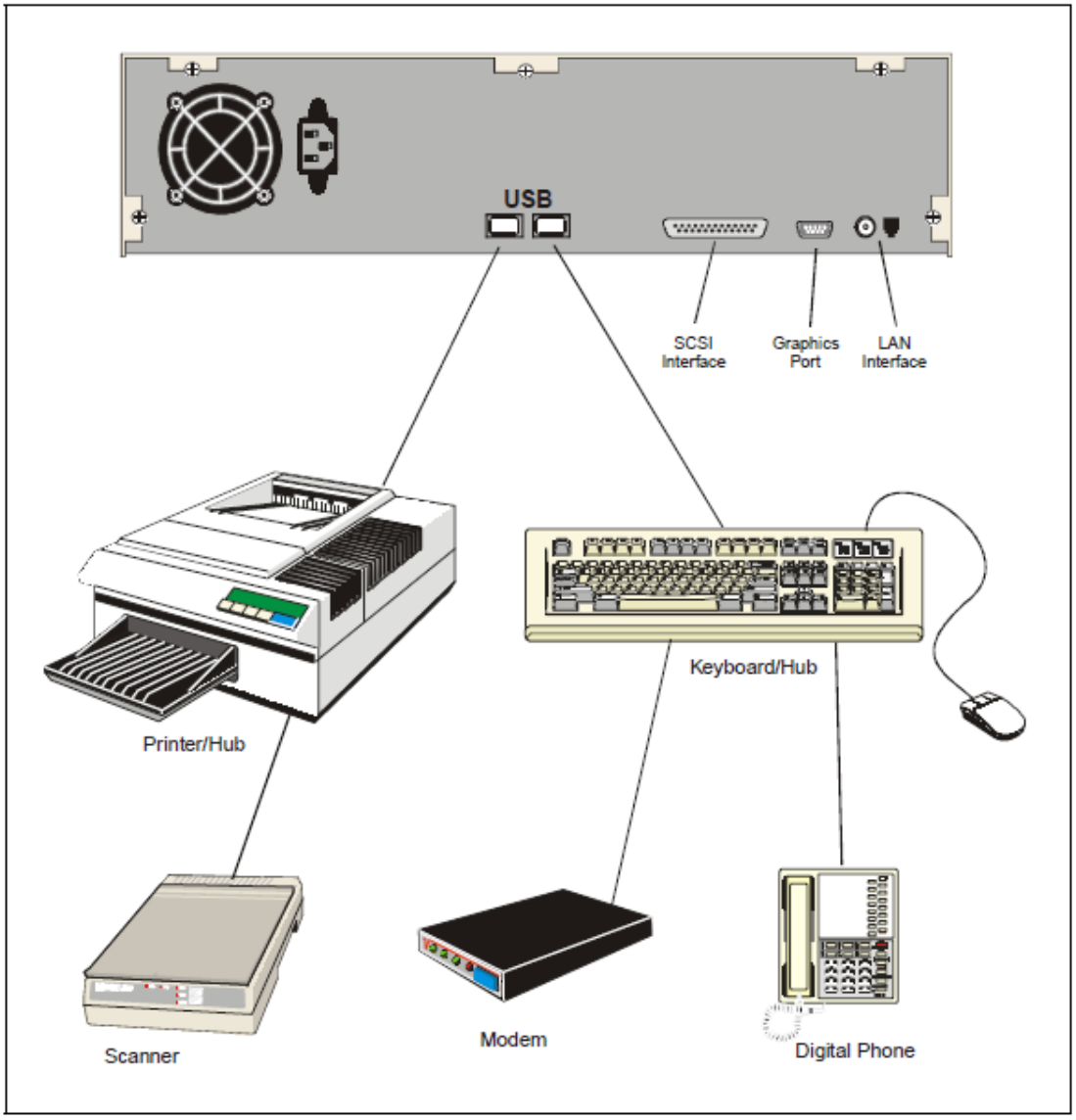


所以，总的来说，在 USB 出现之前，各种接口太多，而且都不太容易使用，互相之间的兼容性也较差，因此，才出现了 USB。

而万能的 USB 接口出现的话，整个 PC 机箱背后的接口，就不那么繁杂，显得清静多了：

图表 3 有了 USB 接口之后的 PC 机箱背后的接口

Figure 1-3: USB Device Connections



USB 出现的最初的目的，根据 USB 规范中的解释，是为了：

（1） 将 PC 和电话能连起来：由于大家都认识到，下一代的应用，肯定是实现计算机设备和通讯设备的完美融合。而且，为了实现移动领域内的人机数据的交互，也需要方便且不贵的连接方案。但是，计算机领域和通讯领域却是各自为政的发展，没有考虑互联性。由此，USB 的出现，就是为了解决这一类互联问题的。

（2） 方便用户使用：以前的一些设备，多数不支持即插即用，而且很多设备还需要懂行的用户去手动配置，然后才可以正常工作，而 USB 的出现，使得用户不用关心设备的细节，不需要去另外再配置什么参数，直接插上就可以用了，而且还支持即插即用，很是方便。

（3） 接口扩展性要好：之前的众多接口，导致不同的应用，需要使用不同的接口，很是繁琐。USB 的出现，支持众多的应用，都使用统一的 USB 的接口，方便了用户，不需要再搞懂各种接口的用途和差异。

总的来说，USB 的出现，是希望通过此单个的 USB 接口，同时支持多种不同的应用，而且用户用起来也很方便，直接插上就能用了，也方便不同的设备的之间的互联。
说白了，就相当于在之前众多的接口之上，设计出一个 USB 这么个万能的接口，以后各种外设，都可以用这一种接口即可。
这估计也是 USB 的名称中的 Universal 通用的这一个词的来历吧。

3. USB 相关的基础知识

在介绍 USB 协议的细节知识之前，有很多相关的软硬件的基础知识，需要了解一下。

3.1. USB 相关的硬件

USB 设备，从物理上的逻辑结构来说，包含了主机 Host 端和设备 Device 端。
其中，主机 Host 端，有对应的硬件的 USB 的主机控制器 Host Controller，而设备端，连接的是对应的 USB 设备。

3.1.1. USB 控制器类型：OHCI，UHCI，EHCI，xHCI

由于历史原因，导致 USB 的主机控制器，出现了多种不同的类型，即 OHCI 和 UHCI，EHCI，和 xHCI。
不论是那种 USB 主机控制器，简称主控，都是符合对应的 USB 的规范的，都是实现了对应的 USB 规范中所规定的 USB 主控所要的那些事情的。只不过是不同的 USB 主控的类型，有着不同的特点。
下面对这些不同类型的 USB 主控制器，进行简要的解释。

3.1.1.1. OHCI 和 UHCI

OHCI，Open Host Controller Interface，创立者是 Compaq，Microsoft 和 National Semiconductor。
UHCI，Universal Host Controller Interface，创立者是 Intel。

两者之间的相同点是：
不论是 OHCI 还是 UHCI 都是对应于 USB 1.1 的标准的，都是完全符合 USB 协议标准的。

区别在于：
只是各自的实现方式有些略微不同而已。当然对应的具体的性能，也略有差别，具体的差异，，和实际的应用有关系。
但是本身 OHCI 和 UHCI 的区别在于：
虽然都是实现了 USB1.1 协议规范，但是在功能划分上，OHCI 更多地把要做的事情，用硬件来实现，因此，实现 OHCI 的 USB 控制器的软件驱动的开发工作，相对要容易些，软件要做的事情，相对较少。
对应地，OHCI 更多地应用在扩展卡，尤其是嵌入式领域中，常见的很多开发板中的 USB 的控制器，很多都是 OHCI 的。
而 UHCI 把更多的功能，留给了软件，相对来说，软件做的事情，即负担要重些。但是实现对应的 UHCI 的硬件的 USB 控制器，价格上，就相对便宜些。
对应地，UHCI 更多地应用在 PC 机中的主板上的 USB 控制器。

3.1.1.1.1. 为何 Intel 设计的 UHCI 把更多的任务都留给软件实现？

对于两者的区别和实际的应用，自己分析，不难发现，其是有着内在的逻辑关系的。
因此，作为 UHCI 的创立者 Intel，创立了 UHCI，把更多的 USB 需要做的事情，留给了软件，这样就可以实现出相对教便宜的 USB 的主控制器了，可以用于 PC 端的 CPU 所对应的主板上，便宜的 USB 主控，当然相对市场来说，更容易多卖出去一点，有利于市场推广。PC 主板卖出的多了， 自然对应的 Intel 的 CPU，也会多卖点，Intel 自然可以赚更多的钱了。

3.1.1.1.2. 为何嵌入式系统中的 USB 主控多用 OHCI，而非 UHCI？

而对应的 Compaq，Microsoft 和 National Semiconductor 所创立的 OHCI，由于把更多的 USB 要做的事情，都用硬件实现了，这样对应的软件驱动所要做的事情，就少了，这样就有利于实现对应的 OHCI 的 USB 主控的驱动了，这点对于嵌入式系统来说，尤其重要，因为本身嵌入式系统就是资源有限，所以要尽量少的利用其他资源，比如 CPU 资源，去实现特点的功能，所以，倾向于采用对软件资源要求少的 OHCI，而不是 UHCI，否则用了 UHCI 的 USB 主控的话，需要实现对应的驱动，软件要做的事情太多，不利于在嵌入式系统这有限的资源环境下实现。

3.1.1.1.3. OHCI 和 UHCI 技术细节上的区别

关于 OHCI 和 UCHI 在技术细节方面，更加详细的区别主要有这些：
（1）单帧内的 stage 的个数
对于控制传输来说：
OHCI：在单个帧内，可以调度多个 stage；
UHCI：在单个帧内，只调度一个 stage。
（2）单帧内的 transaction 的个数
对于最大数据包大小小于 64 字节的 Bulk 端点来说：
OHCI：单个帧内，可能会有多个 transaction。
UHCI：单个帧内，不超过一个 transaction；
（3）轮询的频率
OHCI：，即使端点描述符中，已经指定了最大延迟是 255ms，但是 OHCI 主控还是会，至少每 32ms 就去轮询一次中断端点；
UHCI：UHCI 主控可以支持，但是不是必须要支持，更低频率地轮询。

3.1.1.2. EHCI

EHCI，Enhanced Host Controller Interface。
简单说就是，EHCI 定义了 USB 2.0 的主机控制器的规范，定义了 USB 2.0 的主控，需要包括哪些硬件实现，需要实现哪些功能，其也对应着对应的系统软件，所面对的是哪些接口。EHCI 对 USB 主控的定义，详细到了寄存器的级别了，即定义了你 USB 主控，都要实现哪些对应的功能和对应的寄存器有哪些，分别是何种功能等。然后对应的软件驱动人员，去写 USB 主控的驱动的时候，也就清楚有哪些可以利用的系统资源，如何去使用这些资源，读取，设置对应的寄存器，实现对应的功能了。

对应的 EHCI 规范，可以去 Intel 的官网找到：
EHCI Specification
http://www.intel.com/technology/usb/ehcispec.htm

3.1.1.3. xHCI

xHCI，Extensible Host Controller Interface
同 EHCI 是针对 USB 2.0 类似，xHCI 是针对的 USB 3.0 规范。
也是定义了 USB 3.0 主控需要如何实现，需要包含哪些功能，也是提供了寄存器级别的定义。对应的 xHCI 规范，可以去 Intel 的官网找到：
Extensible Host Controller Interface (xHCI) Specification for USB 3.0
http://www.intel.com/technology/usb/xhcispec.htm

3.1.1.4. OHCI，UHCI，EHCI，xHCI 的区别和联系

针对上述的解释，对 USB 的不同类型的主机控制器，简要概括如下：

表格 1 不同 USB 控制器类型 OHCI，UHCI，EHCI，xHCI 的区别和联系

USB 主机控制器类型	共同点	区别			
		对应的 USB 的协议和支持的速率	创立者	功能划分	常用于
OHCI	都实现了对应的 USB 的规范中所要求的功能	USB 1.1=Low Speed 和 Full Speed	Compaq，Microsoft 和 National Semiconductor	硬件功能>软件功能=>硬件做的事情更多，所以实现对应的软件驱动的任务，就相对较简单	扩展卡，嵌入式开发板的 USB 主控*
UHCI			Intel	软件功能 > 硬件功能=>软件的任务重，可以使用较便宜的硬件的 USB 控制器	PC 端的主板上的 USB 主控
EHCI		USB 2.0=High Speed	Intel	定义了 USB 2.0 主控中所要实现何种功能，以及如何实现	各种 USB 2.0 主控
xHCI		USB 3.0=Super Speed	Intel	定义了 USB 3.0 主控中所要实现何种功能，以及如何实现	各种 USB 3.0 主控

3.1.2.USB 接口的引脚定义

USB 接口的物理上的对应的引脚和对应含义等，可用下表概括：

表格 2 USB 1.x/2.0 的引脚定义

引脚	名称	电缆颜色	描述
1	VBUS	Red	+5 V，电源
2	D−	White	Data −，数据线
3	D+	Green	Data +，数据线
4	GND	Black	Ground，接地

表格 3 USB 3.0 的引脚定义

Pin	Color	Signal name ('A' connector)	Signal name ('B' connector)
1	Red	VBUS	
2	White	D−	
3	Green	D+	
4	Black	GND	
5	Blue	StdA_SSRX−	StdA_SSTX−
6	Yellow	StdA_SSRX+	StdA_SSTX+
7	Shield	GND_DRAIN	
8	Purple	StdA_SSTX−	StdA_SSRX−
9	Orange	StdA_SSTX+	StdA_SSRX+
Shell	Shell	Shield	

3.1.3.USB 的接口（connector）类型

由于 USB 的产生就是为了支持众多种应用的，而由于各种应用中，对于硬件接口的大小也有一些限制，比如有些小型设备或者移动式设备中，接口不能太大等，所以而设计出多种类型的接口，用于不同的应用。

在介绍插头和插座之前，先多解释一下，基本的叫法。

插头，plug，对应的也叫公口，即插别人的；

插座，receptacle，对应也叫做母口，即被插的；

对上述解释，想多了的，面壁去；没想多的，继续看技术介绍。

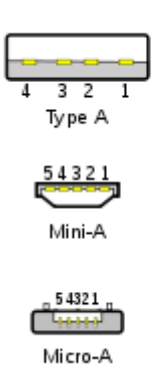
下面就来简单的介绍一下不同的 USB 接口类型，即各种不同的插头插座：

USB 的接口类型，根据接口形状不同，主要可以分为三大类：

（1）普通的硬件直接叫做 Type，（2）然后有小型版本的叫 Mini 迷你的，（3）和更加小的，叫做 Micro 微小的，其中每一种大类中，又都可以分为两类，A 类（Type A）和 B 类（Type B）。

下面就用表格的形式，详细对比 USB 的各种接口，包括对应的插头和插座：

图表 4 USB 接口分类

USB 接口（插头）概览	大的分类	细分	特点	插头图片示例	对应的插座	常见用途
	Type	Type A	长方形			普通 PC 端
		Type B	内部正方近乎形，外部是梯形			USB 设备的接口
	Mini	Mini A	小型版的梯形			数码相机，移动硬盘等移动设备
		Mini B	小型版的长方形			
	Micro	Micro A	比 Mini 更扁			手机等移动设备
		Micro B				

注：

1. 目前多数手机厂商已宣布统一使用 Micro USB 接口作为手机充电器标准接口。

3.2. USB 相关的软件

如果某个 USB 设备正常工作，除了对应的硬件之外，还需要对应的软件支持。

3.2.1. USB 设备端的固件（Firmware）

而对于 USB 设备端来说，内部是需要有对应的设备端的驱动，常常称其为固件 Firmware，其实现了对应的设备端的 USB 所要做的事情，主要是相应一些标准的请求，完成对应的数据读取和写入等。

3.2.2.USB 主机（Host）端的 USB 驱动和软件

对应的，主机 Host 端，也需要对应的驱动，此部分驱动，不论是 Linux 下，还是 Windows 下，都已经实现了常见的驱动了，所以一般来说，很少需要驱动开发者再去写相关的驱动。

3.2.3.其他一些 USB 测试和协议分析等软件

在设备驱动开发阶段或者 USB 出现问题，需要调试的时候，往往就需要一些调试工具了。一般来说，都包含了对应的 USB 硬件测试工具，加上对应的软件工具，去捕获对应 USB 总线上的数据，即所谓的 USB 抓包，然后再去分析抓取出来的数据，是否是期望的，是否符合 USB 协议的规范定义。

我所见过的一些 USB 抓包工具有：

1. Ellisys 的 USB Explorer 260 硬件 加上对应的 USB 软件 Ellisys USB Analysis Software，实现 USB 数据抓包和分析。
2. Catalyst Enterprises 公司的硬件，加上对应的软件 SBAE USB，实现 USB 数据捕获和分析。

另外，当然也有一些 USB 开发相关的工具，比如：

1. usbview 用于查看 USB 设备的详细信息。
2. USB20CV 等用来做 USB 兼容性测试的工具，具体工具下载和详细解释可以去 USB 官网找到：

http://www.usb.org/developers/tools/

3. bus hound，好像是个纯软件的 USB 抓包工具，据说不支持 USB 枚举过程的抓包。具体没用过，只是听说过。

4. USB 协议概览

4.1. USB 2.0 协议内容概览

USB 协议，由于涉及内容太多，所以在此一个文档中解释清楚，是不现实的。
此处能做的和要做的，就是对于 USB 协议简明地介绍一下关于 USB 本身协议部分的内容。

当前最新的 USB 协议，已经发展到 USB 3.0 了。但是主流的 USB 设备和技术，还是以 USB 2.0 居多。所以此文，主要是以 USB 2.0 为基础来解释 USB 协议的基础知识，当然，会在相关内容涉及到 USB 3.0 的时候，也把 USB 3.0 的相关内容添加进来。
关于 USB 2.0 和 USB 3.0 等 USB 的协议规范，可以去官网下载：
http://www.usb.org/developers/docs/

其实，说实话，不论是谁，如果开始看 USB 协议的时候，发现单独对于 USB 2.0 规范本身这一个文档来说，竟然都有 650 页，而如果再加上，新的 USB 3.0 规范的 482 页，和其他一些辅助的 USB 相关的规范定义等文档，即使你是英语为母语的人，如果要看完这么多页的协议规范，估计也会吐的，更别说我们中国人了。
所以，此处，就来简单以 USB 2.0 规范为例，分析一下，具体其都主要包含哪些内容，然后你会发现，其实和 USB 协议本身相关的内容，相对则不会那么多，大概只有 97 页左右的内容，是我们所要关心的。
下面就来分析看看，USB 2.0 的规范中，具体都包含了哪些内容：

表格 4 USB 2.0 协议的内容组成

章节	名称	内容描述	页数
1	介绍	介绍了为何要有 USB 以及 USB 协议内容的涵盖范围。此章节最重要的信息就是，引用了 USB Device Class 规范。不需要看。	2
2	术语和缩写	名词解释，一般的协议都会有这一章节的。无需看。	8
3	背景	说明了 USB 的来由，以及目的是为了是 USB 的用户，注意不是为了是 USB 的开发者 ,更加容易使用。介绍了 Low ,Full , High Speed 三种不同的速度以及对应的应用领域。所以也不需要看。	4
4	系统架构综述	可以从这章开始看。此章介绍了 USB 系统的基本架构，包括拓扑关系，数据速度，数据流类型，基本的电气规范。	10
5	USB 数据流模型	此章开始介绍 USB 中数据是如何流向的。其先介绍了端点和管道，然后对控制，中断，等时，批量四种传输类型进行了详细阐述。其中，重要的一点是，要搞懂每种传输类型，当然，这对于初学者来说可能会有那么一点难。	60

6	机械的	此章详细介绍了 USB 的两种标准的连接头，即接口的类型，其中需要了解的一点是，A 类接口旨在用于数据向下流的（downstream），而 B 类接口旨在用于数据向上流的（upstream）。因此，你应该知道，不应该也不可能去将一个 USB 线，连到两个都是 upstream 的端口上。而所有的 full 或 high speed 的 USB 线，都是可拔插的，而低速的 USB 线，应该是焊死的。如果你不是 USB 接口的制造商，那么就没必要细看这章，而只需要大概浏览一下其中关于 USB 的接口类型的相关内容即可。	33
7	电子的	此章详解了 USB 总线上的电子信号，包括线阻，上下沿的时间，驱动者和接受者的规范定义，以及比特位编码，比特位填充等。此章中需要知道的，更重要的一点是，关于使用电阻在数据线上的偏压，去实现 USB 设备的速度类型检测，以及设备是总线供电还是自供电。除非你是在晶元级别上设计 USB 数据收发模块的相关人士，否则都可以直接跳过此章节。而正常的 USB 设备的数据手册中，都会有相关的解释，说明关于 USB 总线阻抗需要匹配电阻的阻值是多少。	75
8	协议层	此章，从字节的级别，解释了 USB 数据包的细节，包括了同步，PID，地址，端点，CRC 域。多数的开发人员都还没注意到这部分的底层的协议层，因为 USB 的设备中的硬件 IC，会帮你做这些事情的。然而，多学习和了解一些关于报告状态和握手协议方面的知识，还是有必要的。。	45
9	USB 设备框架工作	此章，是整个 USB 协议中，用到的最多的一章。此章详细阐述了 USB 总线枚举的过程，以及一些 USB Request 的详细语法和含义，比如 set address，get descriptor 等，这些相关内容在一起，就构成了最常用的 USB 的协议层，也是通常 USB 编程人员和开发者所看到的这一层。此章节，必须详细阅读和学习。	36
10	USB 主机的硬件和软件	此章介绍了和 USB Host 相关的知识。包括了数据帧 frame 和微帧 microframe 的产生，主机控制器的需求，软件机制和 USB 的驱动模型等。如果你不是去设计 USB Host 的话，那么就直接跳过此章即可。	23
11	Hub 规范	此章定义了 USB Hub 相关的规范，包括了 Hub 的配置，分离传输，Hub 类的标准描述符等。同理，如果你不是去设计 USB Hub，那么也可以忽略此章。	143

注：

1. 关于第九章=chapter 9=ch9，多说明一下，由于其特殊性，特殊在于大部分和 USB 协

议相关的内容，都在此章节内，所以，你会在其他地方看到有关此 ch9 的说法。比如 Linux 源码中关于 USB 协议实现的部分的代码，会看到有对应的头文件是：

include\linux\usb\ch9.h

此文件，就是指的是 USB 规范中的 chapter 9，第九章。

这也意味着，以后其他人如果谈及 USB 的话，说到第九章，指的就是此 USB 规范中的 chapter 9，因为其包含了 USB 协议的软件实现所有关的多数的内容。

所以，由上述总结，我们可以看出：

对于只是为 USB 外设开发驱动的开发者的话，那么有关的章节只有：

- 4 系统架构综述
- 5 USB 数据流模型
- 9 USB 设备框架工作
- 10 USB 主机的硬件和软件

如果是对于 USB 外设的电子设计研发人员，有关系的章节有：

- 4 系统架构综述
- 5 USB 数据流模型
- 6 机械的
- 7 电子的

这么一来，如果是打算做 USB 设备驱动开发的话，其实我们要看的，只是 USB 2.0 规范中的一部分，大概有 10+60+36+23=129 页，所以，相对来说，还算能够接受，至少比要看那 600 页，要少了很多。

当然，这其中的内容，也还是不少。

而下面这些内容，就是将其中最基本的内容，精简出来，以方便想要快速了解 USB 基础知识的人，尽快地，更加清晰地，了解到 USB 的基础知识。

4.2. USB 协议的版本和支持的速度

USB 协议，也像其他协议一样，经历过很多个版本，但是正式发布出来的，主要有 4 个。

其中，从开始的 USB 1.1，发展到后来的 USB 2.0，以及最新的协议版本是 USB 3.0.

不过这三个版本都是针对的是有线的（corded）设备来说的，在 USB 2.0 和 USB 3.0 之间，发布过一个是针对无线设备的 USB 协议，叫做 USB Wireless，也被称为 USB 2.5。

其中，USB 1.1 中所支持的速度是低速（Low Speed）的 1.5Mbits/s，全速（Full Speed）的 12Mbits/s，而 USB 2.0 提高了速度至高速（High Speed）的 480Mbits/s，而最新的 USB 3.0，支持超高速（Super Speed）的 5Gbits/s。

下面简要总结一下，各个 USB 协议版本的演化历史：

表格 5 USB 协议的版本的演化

USB 协议		针对的设备	对应的速度		备注
版本号	发布日期		名称	速率	
1.1	1998 年 8 月	有线的	Low Speed Full Speed	1.5Mbits/s=192KB/s 12Mbits/s=1.5MB/s	
2.0	2000 年 4 月	有线的	High Speed	480Mbits/s=60MB/s	
2.5	2010 年 9 月	无线的			Wireless USB 1.1
3.0	2008 年 11 月	有线的	Super Speed	5.0Gbits/s=640MB/s	

4.2.1.为何 USB 的速度，最开始没有设计的更快些？

有人会问，既然 USB 技术本身可以设计成速度更快的，为何最开始不把 USB 的设计成速度更快的呢？比如，最开始为啥没有把 USB 设计成 2.0 的那样的速度呢？

那是因为，任何规范和协议，都离不开当时的背景。关于 USB 的速度发展，其有其自身的考虑。

比如最开始的 USB 1.1，对于低速的 1.5Mbits/s 的速度，虽然速度很低，但是由于此速度，主要用于 USB 鼠标，键盘等低速设备，所以本身就够用了，而且速度低还有个好处，那就是对于电磁辐射 EMI 的抗干扰能力较强些，而使得设计和制造对应的硬件设备的成本要降低些，比如可以使用相对便宜的陶瓷振荡器（resonator）做晶振（crystal）。

而后来的 USB 2.0,的出现，则是为了满足人民群众日益增长的对于高速速度传输方面的需求，比如你从 MP3 里面拷贝歌曲出来，如果是 USB 1.1，那么实际效果最快也就 1MB 左右，而如果是 USB 2.0，平均效果大概有 3MB/s,5MB/s，性能好的可达 10MB/s，20MB/s，所以，如果拷贝个 1G 的东西，相当于 USB 1.1 要 1 小时左右，而 USB 2.0 只要 1 分钟左右。因为如果没有 USB 2.0 的出现的话，那么现在的人们，早就放弃了 USB 了，因为谁也忍受不了这个太慢的速度。所以为了满足大家的需求，才有了 USB 2.0 的出现。

而对于最新的 USB 3.0，同理，也是为了满足现在的一些及以后的可能的需求，即希望拷贝蓝光光碟的内容到硬盘上，动辄都是几个 G 的内容，以 USB 2.0 的速度，那怎么说也得个几分钟，而有了 USB 3.0 后，就有望实现，几秒或者几十秒，哗的一下，就把多少个 G 的东西，拷贝传输到别的介质上了。当然，这只是理论上的，实际的 USB 3.0 的速度，受到 USB 设备的硬件本身能力，和对应的软件驱动，以及所设计的介质不同，而会有不同的速度。

4.3. USB 系统的核心是 Host

USB 是 Host 端控制整个的总线数据传输的。单个 USB 总线上，只能有一个 Host。USB 协议本身，是不支持多个 Host 端的。

不过，相对有点特殊的是，USB 为了支持多个设备互相，而不需要另外接 Host，比如一个数码相机和一个打印机，希望把打印机和数码相机直接相连接，然后就可以实现通过 USB，把数据从数码相机传送到打印机中，打印机就可以打印了。

此处，OTG 引入了一个新的概念，HNP（Host Negotiation Protocol），主机协商协议，允许两个设备之间互相协商谁去当 Host。不过，即使在 OTG 中，也只是同一时刻，只存在单个的 Host，而不允许存在多个 Host 的。

USB 中的 Host 端，负责所有底层的数据传输的控制以及数据带宽的安排调度。

底层数据的发送，是使用一种基于令牌环的协议，通过不同类型的传输方法而实现的。

USB 的设备连接方式，即拓扑方式，是星形的，所以，在需要连接多个设备的时候，常需要用到对应的 Hub。这种星形拓扑的好处是，对于每一个设备的供电情况，都可以控制，也可以同时支持多个不同类型的设备。而万一某设备电流过高了，也不会影响到其他设备。

同一 USB 总线中，最多可连接 127 个设备。当然，实际中你所看到的，往往都比较少，至少普通用户用到的，也就是几个的数量级。常见的是一个 USB Host，提供，2,3 个或者 4,5 个接口。而 USB Host 的内部实现，也有最开始的只有 1 个 USB 主控 Host Controller，变得有些内部包含 2 个甚至更多个 USB 主控，以提高所接的 USB 设备的带宽，即速率，因为单个 USB 主控所接多个 USB 设备的话，那么多个设备是共享此 USB 主控的带宽的。

4.4. USB 中用 NRZI 来编码数据

之前已经介绍过了 USB 的引脚定义了，但是对于其中的 USB 2.0 的两根数据线 D+和 D-所对应的数据传输，却没有详细介绍。此处就是介绍，在此串行数据线中，数据是如何被编码和传送的。

USB 所传输的数据，用的数据编码方式是 NRZI（Non-Return-to-Zero Inverted），其具体的含义解释，个人觉得某人解释的很清楚，所以在此转载其对应的这篇文章：

USB 的 NRZI 编码

http://galeki.is-programmer.com/posts/10054.html

这两天继续看 USB 相关的内容，准备用纯软件实现一下 USB 设备传输，为将来的项目打好基础。

首先碰到的就是这个 NRZI 编码的问题了，基础太薄弱，看了一上午总算明白了大概。

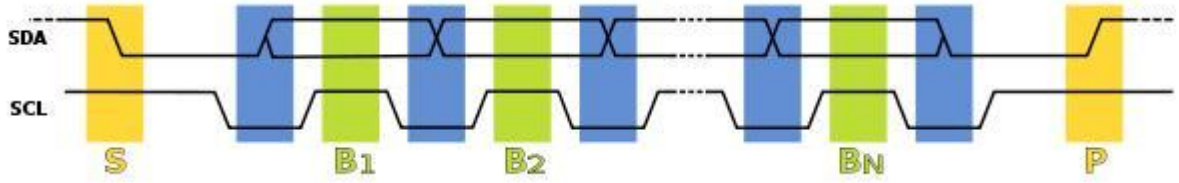
首先，USB 的数据是串行发送的，就像 UART、I2C、SPI 等等，连续的 01 信号只通过一根数据线发送给接受者。

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：

图表 5 I2C 数据编码格式

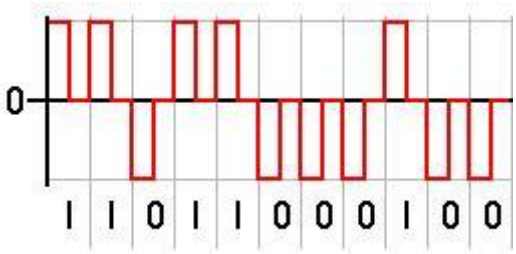


虽然这样解决了问题，但是却需要附加一根时钟信号线来传输时钟。有没有不需要附加的时钟信号，也能保持两端的同步呢？

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

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：

图表 6 归零编码

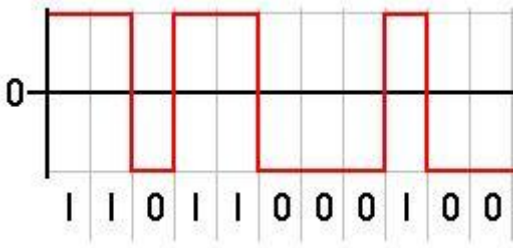


从图上就可以看出来，因为每位传输之后都要归零，所以接收者只要在信号归零后采样即可，这样就不在需要单独的时钟信号。实际上， RZ 编码就是相当于把时钟信号用归零编码在了数据之内。这样的信号也叫做自同步（self-clocking）信号。

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。

那么，我们去掉这个归零步骤，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：

图表 7 非归零编码

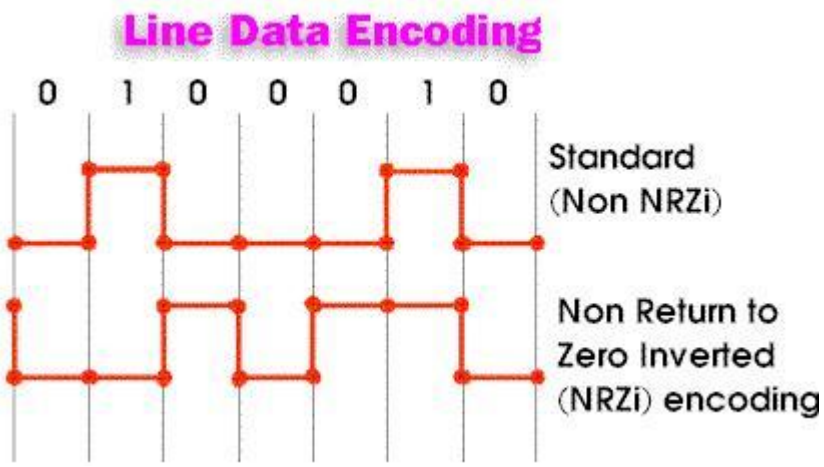


这样，浪费的带宽又回来了，不过又丧失宝贵的自同步特性了，貌似我们又回到了原点，其实这个问题也是可以解决的，不过待会儿再讲，先看看什么是 NRZI：

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：

图表 8 NRZ 和 NRZI



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。

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（复习下前面：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，所以接收者在接收数据的同时，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，就是虽然接收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 个了。

4.4.1. 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 7 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。

5. 引用文章

官方的 USB 2.0 规范
http://www.usb.org/developers/docs/usb_20.zip

USB in a Nutshell - Making sense of the USB standard
http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf

USB Complete
http://www.lvr.com/usbc.htm

【很好的学习 Linux 驱动的教材】Linux 那些事儿系列[全][pdf]
http://bbs.chinaunix.net/thread-1977195-1-1.html

USB 接口
http://zh.wikipedia.org/zh/USB

USB
http://en.wikipedia.org/wiki/USB

USB 2.0 A 型、B 型、Mini 和 Micro 接口定义及封装
http://www.metsky.com/archives/474.html

Wireless USB Documents
http://www.usb.org/developers/wusb/docs/

USB 的 NRZI 编码
http://galeki.is-programmer.com/posts/10054.html

USB 3.0
http://en.wikipedia.org/wiki/USB_3.0