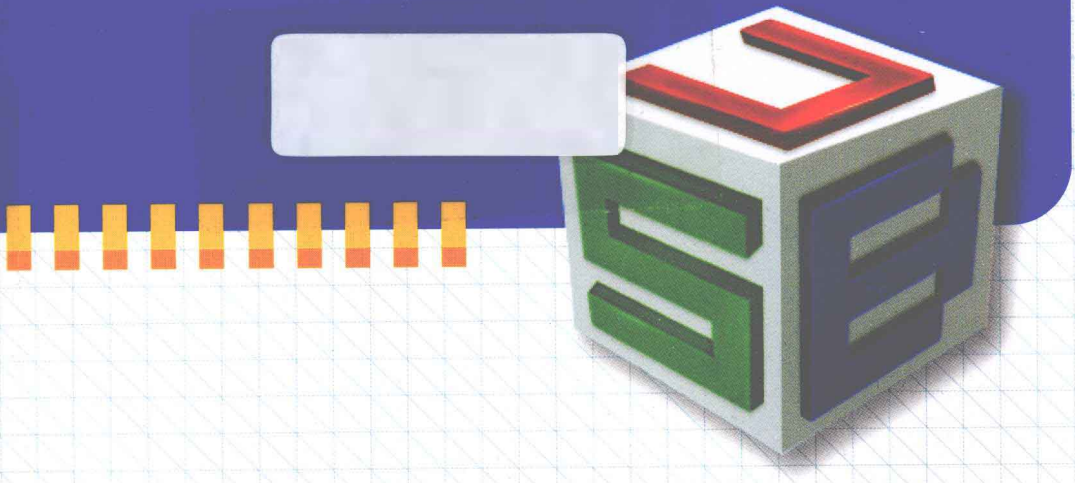
深入浅出

USB系统开发

——基于ARM Cortex-M3

主　编　王川北　刘　强

副主编　屈召贵　孙　活　蔡德洋



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

深入浅出 USB 系统开发
——基于 ARM Cortex－M3

主　编　王川北　刘　强
副主编　屈召贵　孙　活　蔡德洋

北京航空航天大学出版社

内 容 简 介

本书系统地阐述了 USB 协议、Stellaris USB 处理器的体系结构、工作原理和设计方法，并通过多个 USB 开发实例，详细介绍了 USB 开发思路、流程及编程方法，并在此基础上讲解了嵌入式 USB 主机、USB OTG 开发。全书共分 15 章：第 1 章介绍 USB 系统基础知识、基本术语、USB 基本结构、开发流程、USB 枚举、USB 描述符格式、主机和设备开发过程等；第 2 章介绍 Cortex - M3 内核的 USB 处理器，包括 USB 基本模块、工作方式、USB 寄存器操作、寄存器级编程等；第 3 章介绍使用设备驱动库函数进行 Cortex - M3 编程，包括内核操作、中断控制、GPIO 编程、USB 基本编程等；第 4 章介绍 TI 的 USB 库使用及编程；第 5～10 章介绍 USB 设备开发；第 11 章介绍 USB 主机开发；第 12 章介绍 USB OTG 开发；第 13 章介绍 USB 设备开发总结及注意事项；第 14 章介绍 USB 主机开发总结及注意事项；第 15 章是 USB 系统开发总结，包括常见概念性问题、开发问题等，阐述其产生的基本原因，并提供了解决此类问题的方案。

本书可作为高等院校电子类、仪器仪表类、控制类等专业的 USB 系统开发教材或参考用书，也可供广大从事 USB 系统开发的工程技术人员参考。

图书在版编目(CIP)数据

深入浅出 USB 系统开发 ：基于 ARM Cortex - M3 / 王川北，刘强主编. --北京 ：北京航空航天大学出版社，
2012.8

ISBN 978 - 7 - 5124 - 0872 - 2

Ⅰ. ①深… Ⅱ. ①王… ②刘… Ⅲ. ①USB 总线—串行接口 ②微控制器 Ⅳ. ①TP334.7 ②TP332.3

中国版本图书馆 CIP 数据核字(2012)第 157759 号

深入浅出 USB 系统开发——基于 ARM Cortex - M3

主　编　王川北　刘　强

副主编　屈召贵　孙　活　蔡德洋

责任编辑　陈　旭

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191)　http://www.buaapress.com.cn

发行部电话：(010)82317024　传真：(010)82328026

读者信箱：emsbook@gmail.com　邮购电话：(010)82316936

涿州市新华印刷有限公司印装　各地书店经销

*

开本：710×1 000　1/16　印张：27.25　字数：597 千字

2012 年 7 月第 1 版　2012 年 7 月第 1 次印刷　印数：4 000 册

ISBN 978 - 7 - 5124 - 0872 - 2　定价：52.00 元

前 言

USB 最初是由英特尔与微软公司倡导发起的，其最大的特点是支持热插拔和即插即用。当设备插入时，主机侦测此设备并加载所需的驱动程序，因此使用远比 PCI 和 ISA 总线方便。USB 规范第一次是于 1995 年，由 Intel、IBM、Compaq、Microsoft、NEC、Digital、North Telecom 等 7 家公司组成的 USBIF（USB Implement Forum）共同提出，USBIF 于 1996 年 1 月正式提出 USB 1.0 规范，频宽为 12 Mbps，不过因为当时支持 USB 的外围设备很少，所以主机板厂商不太把 USB Port 直接设计在主机板上。1998 年 9 月，USBIF 提出 USB 1.1 规范来修正 USB 1.0，主要修正了技术上的小细节，但传输的频宽不变，仍为 12 Mbps。USB 1.1 向下兼容 USB 1.0，因此对于一般使用者而言，感受不到 USB 1.1 与 USB 1.0 的规范差异。

2000 年 4 月，广泛使用的 USB 2.0 推出，速度达到了 480 Mbps，是 USB 1.1 的 40 倍；如今 8 个半年头过去了，USB 2.0 的速度早已无法满足应用的需要，USB 3.0 也就应运而生了，最大传输带宽高达 5.0 Gbps，也就是 625 MB/s，同时在使用 A 型的接口时向下兼容。

随着电子信息科学的飞速发展，近年来嵌入式技术的应用越来越广泛，生活中无所不在，无处不用。面对当前嵌入式行业的优越形势，业界人士掀起了学习嵌入式系统理论及应用开发的热潮，选择合适的嵌入式系统学习至关重要，嵌入式系统涉及嵌入式处理器、实时操作系统等内容。在众多的嵌入式处理器中，基于 ARM 系列的处理器得到了行业人士的认可，其中 Cortex－M3 是 ARM 公司推出的针对微控制器应用的内核，它提供了适用于众多高性能和低成本需求的嵌入式应用，成为时下 MCU 应用的热点。

本书选用 Stellaris 的 USB 控制器为控制芯片介绍 USB 系统开发，该类 USB 处理器支持 USB 主机/设备/OTG 功能，在点对点通信过程中可运行在全速和低速模式。它符合 USB 2.0 标准，包含挂起和恢复信号。它包含 32 个端点，其中包含两个用于控制传输的专用连接端点（一个用于输入，一个用于输出）以及 30 个由固件定义的端点，并带有一个大小可动态变化的 FIFO，以支持多包队列。可通过 μDMA 来访问 FIFO，将对系统软件的依赖降至最低。USB 设备启动方式灵活，可软件控制是否在启动时连接。USB 控制器遵从 OTG 标准的会话请求协议（SRP）和主机协商协议（HNP）。本书将重点讲解 Luminary（TI）的 Stellaris USB 控制器原理及应用。

本书深入浅出地讲解 USB 系统的基础知识，在内容上力求精简，快速上手，书中

大量实例来自科研实践和电子竞赛优秀作品，可作为高等院校本、专科的USB系统开发的教材，也可作为对USB系统自学和工程技术人员的参考书。

全书共分15章。

第1章：介绍USB系统基础知识、基本术语、USB基本结构、开发流程、USB枚举、USB描述符格式、主机和设备开发过程等。

第2章：介绍Cortex－M3的内核USB处理器，包括USB基本模块、工作方式、USB寄存器操作、寄存器级编程等。

第3章：介绍使用设备驱动库函数进行Cortex－M3编程，包括内核操作、中断控制、GPIO编程、USB基本编程等。

第4章：TI的USB库使用及编程。

第5～10章：介绍USB设备开发。

第11章：介绍USB主机开发。

第12章：介绍USB OTG开发。

第13章：介绍USB设备开发总结及注意事项。

第14章：介绍USB主机开发总结及注意事项。

第15章：USB系统开发总结，包括常见的概念性问题、开发问题等，阐述其产生的基本原因，并提供了解决此类问题的方案。

参加本书编写的有王川北、屈召贵、刘强、孙活、蔡德洋等。王川北负责全书的统稿工作。其中第1～8章由王川北和孙活编写；第8～15章由屈召贵和刘强编写。本书配备有实验开发装置，由王川北和蔡德洋设计。蔡德洋和孙活参与了本书各章节的审阅。

本书在编写过程得到了四川师范大学成都学院的大力支持，还得到了学院胡增江老师、汪光宅高级工程师、林信元教授、鲁顺昌教授、吴自恒教授、杨肇基教授、郭群扬工程师的大力帮助与指导。在此表示衷心感谢。

这是在非计算机、电子专业嵌入式系统课程教学改革的一个尝试，因编者水平所限，难免有不尽如人意之处，敬请读者提出宝贵意见和建议！

笔者的通信地址：四川成都郫县团结镇学院街65号，邮编：611745。

有兴趣的读者可以发送电子邮件到：paulhyde@126.com或登录：www.isjtag.com与笔者联系；也可发送邮件到：xdhydcd5@sina.com，与本书策划编辑沟通。

编 者

2012.5

目 录

第1章

USB基础

USB最初是由英特尔与微软公司倡导发起的，其最大的特点是支持热插拔和即插即用。当设备插入时，主机侦测此设备并自动加载所需的驱动程序。随着便携式设备的发展，USB的优越性更加明显，没有哪一种手机没有USB接口；没有哪一种通信接口能有USB那么方便可靠；没有哪一种通信方式的传输速度有USB那么快且用线少。本章主要介绍USB的基本理论、基础知识、基本术语、基本结构等，同时也简单介绍USB的开发流程、枚举过程等。通过本章的学习，读者可以掌握USB的基础知识，为后面USB系统开发打好基础。

1.1 USB介绍

USB，Universal Serial BUS(通用串行总线)的缩写，其中文简称为“通用串行总线”，是一个外部总线标准，用于规范计算机与外部设备的连接和通信，是应用在PC和嵌入式领域的接口技术。USB接口支持设备的即插即用和热插拔功能。USB通信方式为串行通信。在1994年底由英特尔、康柏、IBM、Microsoft等多家公司联合提出，目前使用最多的是USB 1.1和USB 2.0，几乎每一台计算机上都有4～8个USB接口头。USB 3.0已经“崛起”，并向下兼容；USB 3.0芯片已经大量生产，其相关产品已经大量投入使用。USB具有传输速度快(USB 1.1是12 Mbps，USB 2.0是480 Mbps，USB 3.0是5 Gbps)、使用方便、支持热插拔、连接灵活、独立供电等优点，可以连接鼠标、键盘、打印机、扫描仪、摄像头、闪存盘、MP3机、手机、数码相机、移动硬盘、外置光软驱、USB网卡、ADSL Modem、Cable Modem等几乎所有的外部设备。所以，掌握USB系统开发是很重要的，就像10年前，电子工程师必须掌握UART串行通信一样，不懂USB不能称其为真正的电子工程师。

常用USB接口如图1-1所示。有方型头(D型头)、扁型公头、扁型母头等，其中公头主要用于USB设备中，比如U盘使用USB扁型公头；母头主要用在USB主机上，其中USB扁型母头用得最多，比如计算机主板上全使用USB扁型母头做USB主机接头。

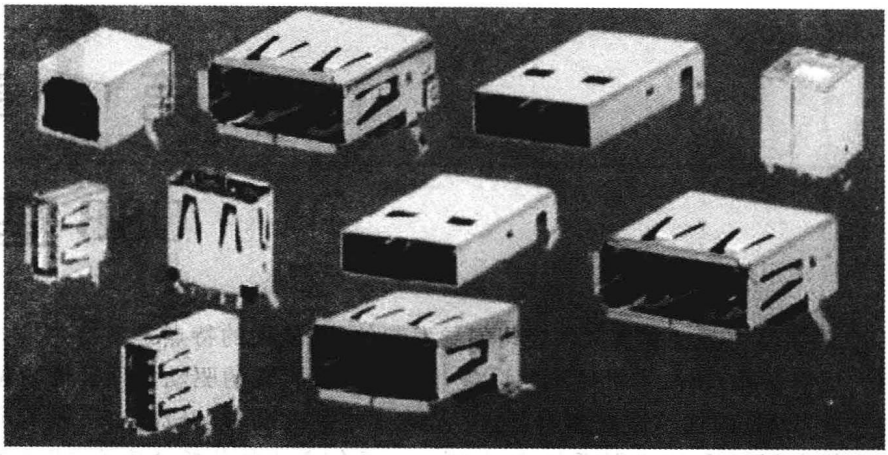


图 1-1　USB 常用接口图

1.2　USB 常用术语

学习 USB 系统开发之前，有必要先了解 USB 开发中可能会遇到的一些常用术语，有助于理解 USB 的工作模式和编程模型。USB 系统开发分为 USB 主机开发和 USB 设备(从机)开发。在一个 USB 系统中，某一个时刻，只有一个 USB 主机，其余均为 USB 设备。但是为了让一个 USB 系统既有 USB 主机功能，又有从机功能，便出现了 USB OTG。因此，USB 开发主要包括 USB 主机、USB 设备、USB OTG 系统开发。下面就一些常用术语进行详细解释：

① USB 主机：在任何一个 USB 系统中，只有一个主机。USB 和主机系统的接口称作主机控制器，主机控制器可由硬件、固件和软件综合实现。USB 主机主要与 USB 设备进行通信、交换数据、设备控制。

② USB 设备：主机的“下行”设备，为系统提供具体功能，可以是多个功能并受主机控制的外部 USB 设备。也称作 USB 外设(从机)，使用 USB B 型连接器连接。USB 主机最多可以支持 127 个 USB 设备。

③ USB OTG：简单地说，OTG 就是 On The Go，正在进行中的意思。是近几年发展起来的技术，2001 年 12 月 18 日由 USB Implementers Forum(USB IF)公布，主要应用于各种不同的设备或移动设备间的连接，进行数据交换，特别是 PDA、移动电话、消费类设备中运用得最多。改变如数码照相机、摄像机、打印机等设备间多种不同制式连接器，多达 7 种制式的存储卡间数据交换的不便。USB OTG 标准在完全兼容 USB 2.0 标准的基础上，增添了电源管理(节省功耗)功能，它允许设备既可作为主机，也可作为外设操作(两用 OTG)。OTG 两用设备完全符合 USB 2.0 标准，并可提供一定的主机检测能力，支持主机协商协议(HNP)和对话请求协议(SRP)。

在OTG中，初始主机称为A设备，设备称为B设备。简单地说，USB OTG既是USB主机也是USB设备。但是在任意时刻，只能有一个A设备(主机)。

④ 集线器：集线器(Hub)扩展USB主机所能连接设备的数量，主要用于扩展。图1-2为USB Hub使用图，从图中可以看出，一个USB Hub将一个USB接口扩展为4个，增加了USB主机连接设备的数量。Hub在USB系统中的功能就像交换机在以太网通信中的功能一样，起到数据交换、增加设备连接数量的作用。

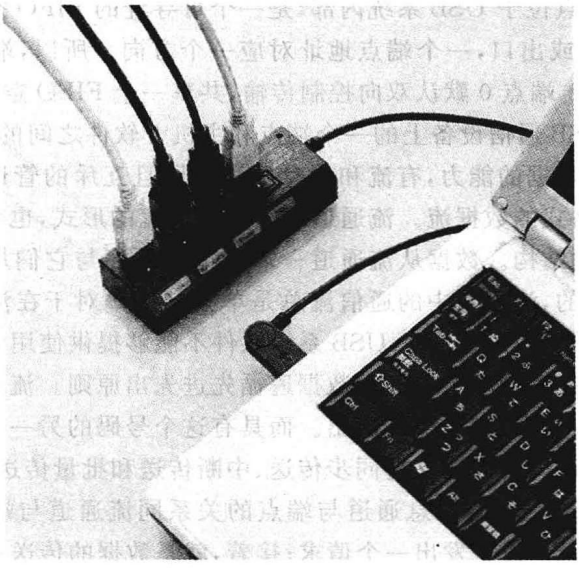


图1-2 USB Hub使用图

⑤ SIE：串行接口引擎，USB控制器内部的“核心”，将低级信号转换成字节，以供控制器使用，并负责处理底层协议，如填充位、CRC生成和校验，且可发出错误报告。物理层的数据功能是根据USB协议规定把USB物理接口上的电信号解码为USB数据。一个USB设备系统由以下几部分组成：进行数据处理的微处理器部分；USB SIE(串行接口引擎)部分；用于数据存储的存储器部分；进行数据编码与外部相连的USB收发器。串行接口引擎完成发送和接收，不对数据内容进行分析。SIE的主要功能是对USB总线上传输的数据进行编码与解码；检测与产生信息包的起始SOP标志和信息包的结束EOP标志；对数据进行串/并转换；检测接收信息包、传送信息包；串行接口引擎是USB设备系统必不可少的接口模块，它的可靠性是USB设备正确执行后续操作的前提。

⑥ 端点：是主机与设备之间通信的目的或来源。控制端点可以双向传输数据，而其他端点只能在单方向传输数据。主机和设备的通信最终作用于设备的各个端点上，是主机与设备间通信流的一个逻辑终端。每个USB设备有一个唯一的地址，这个地址是在设备连上主机时，由主机分配的，而设备中的每个端点在设备内部有唯一

的端点号。这个端点号是在设计设备时给定的。每个端点都是一个简单的连接点，或者支持数据流进设备，或者支持其流出设备，两者不可兼得。基于PnP机制，设备被枚举时，它必须向主机报告各个端点的特性，包括端点号、通信方向、端点支持的最大包大小、带宽要求等（其中端点支持的最大包大小叫作数据有效负载）。每个设备必须有端点0，用于设备枚举和对设备进行一些基本的控制功能。除了端点0，其余的端点在设备配置之前不能与主机通信，只有向主机报告这些端点的特性并被确认后才能被激活。端点位于USB系统内部，是一个可寻址的FIFO空间。类似于高速公路收费口的入口或出口，一个端点地址对应一个方向。所以，端点2-IN与端点2-OUT完全不同。端点0默认双向控制传输，共享一个FIFO空间。

⑦ 管道：是USB通信设备上的一个端点和主机上软件之间的联系，体现了主机缓存和端点间传送数据的能力，有流和消息两种不同且互斥的管道通信格式。流指不具有USB定义格式的数据流。流通道中的数据是流的形式，也就是该数据的内容不具有USB要求的结构。数据从流通道一端流进的顺序与它们从流通道另一端流出时的顺序是一样的，流通道中的通信流总是单方向的。对于在流通道中传送的数据，USB认为它来自同一个客户。USB系统软件不能够提供使用同一流通道的多个客户的同步控制。在流通道中传送的数据遵循先进先出原则。流管道只能连到一个固定号码的端点上，或者流进，或者流出。而具有这个号码的另一个方向的端点可以被分配给其他流通道。流通道支持同步传送、中断传送和批量传送；消息指具有某种USB定义的格式的数据流。消息通道与端点的关系同流通道与端点的关系是不同的。首先，主机向USB设备发出一个请求；接着，就是数据的传送；最后，是一个状态阶段。为了能够容纳请求/数据/状态的变化，消息通道要求数据有一个格式，此格式保证了命令能够被可靠地传送和确认。消息通道允许双方向的信息流，虽然大多数的通信流是单方向的。特别地，默认控制通道也是一个消息通道。在某一时刻，一个端点只能为单个信息请求服务。主机上的多个客户软件通过默认管道能产生请求，但它们以先进先出的顺序被送到端点上。在响应主机事务的数据和状态阶段，端点能控制信息的流动。只有当端点上的当前信息处理完成后，端点才会正常地发送下一个信息。一个设备信息管道在两个方向（IN或OUT标记）要求有一个单一的设备端点号。对于每个方向，USB不允许信息管道与不同的端点号相联系。

⑧ 设备接口：此接口非物理接口，是一种与主机通信的信道管理。设备接口中包含多个端点，与主机进行通信。此接口相当于一个USB系统功能，这个功能可能会使用到多个端点。比如，一个USB设备既有USB键盘功能，又有U盘功能，这种设备就具备两个接口功能，每一个接口都要使用多个端口。

⑨ 描述符：是一个数据结构，使主机了解设备的格式化信息。打个比方，就像平时填写申请表一样，格式与内容都有规定，必须按规定格式和字数编写，否则不予理睬。每一个描述符可能包含整个设备的信息，或是设备中的一个组件（接口）。标准USB设备有5种描述符：设备描述符、配置描述符、字符串描述符、接口描述符及端

点描述符。

⑩ 枚举：USB主机通过一系列命令要求USB设备发送描述符信息，从而知道设备具有什么功能、属于哪一类设备、要占用多少带宽、使用哪类传输方式及数据量的大小，只有主机确定了这些信息之后，设备才能真正地开始工作。打个比方，这就相当于申请处理过程，让你提交什么资料，就必须得提交什么资料，并且必须按申请表规定的格式与字符数进行填写，否则，申请无效。所以枚举对主机与设备进行通信是非常重要的，枚举不成功，主机就无法了解设备，USB设备也不能称其为USB设备。

⑪ 3种传输速率：低速模式传输速率为1.5 Mbps，多用于键盘和鼠标；全速模式传输速率为12 Mbps；高速模式传输速率为480 Mbps。

⑫ 输入：相对主机而言，如果设备输入端点发送的数据为设备发送数据到主机。设备发送数据，主机接收数据。

⑬ 输出：相对主机而言，如果设备输出端点接收的数据为主机发送到设备的数据。设备接收数据，主机发送数据。

⑭ HCD：包含主机控制器和根HUB的硬件，为程序员提供了由硬件实现定义的接口主机控制器设备（HCD）。而实际上HCD在计算机上表现为端口和内存的映射。USB 1.0和USB 1.1的标准都有两个HCD标准，分别是康柏公司的开放主机控制器接口（OHCI）和Intel公司的通用主机控制器接口（UHCI），而VIA公司的USB芯片采纳了Intel的UHCI标准；其他USB芯片大多使用康柏公司的OHCI标准。它们的主要区别是UHCI更加依赖软件驱动，因此对CPU要求更高，但是自身的硬件会更廉价。它们的并存导致操作系统开发和硬件厂商都必须在两个方案上开发和测试，从而导致费用上升。因此USB IF在USB 2.0的设计阶段坚持只能有一个实现规范，这就是扩展主机控制器接口（EHCI）。因为EHCI只支持全速传输，所以EHCI控制器包括4个虚拟的全速或者慢速控制器。这里同样是Intel和VIA使用虚拟UHCI，其他一般使用OHCI控制器。某些版本的Windows上，打开设备管理器，如果设备说明中有“增强”（“Enhanced”），就能够确认它是USB 2.0版的。而在Linux系统中，命令lspci能够列出所有的PCI设备，而USB会分别命名为OHCI、UHCI或者EHCI。列出为16位地址的为EHCI，32位的为OHCI。命令lsusb能够显示所有USB设备的信息，命令dmesg能够显示OS启动时关于USB设备的信息。

⑮ USB电源：USB接头提供一组5 V的电压，可作为USB设备的电源。实际上，设备接收到的电源可能会低于5 V，只略高于4 V。USB规范要求在任何情形下，电压均不能超过5.25 V；在最坏情形下（经由USB供电HUB所连接的LOW POWER设备）电压均不能低于4.375 V，一般情形下电压会接近5 V。一个USB的根集线器最多只能提供500 mA的电流。如此的电流已足以驱动许多电子设备，不过连接在总线供电Hub的所有设备需要共享500 mA的电流额度。一个由总线供电的设备可以使用到它所连接到Hub的所有电能。总线供电的Hub可以将电源供

给连接在 Hub 上的所有设备，不过 USB 的规范只允许总线供电的 Hub 下游串接一层总线供电的设备，因此，总线供电的 Hub 下游不允许再串接另一个由总线供电的 Hub。许多 Hub 有外加电源，因此可以提供电源给下游的设备，不会消耗总线上的电源。若设备需要的电压超过 5 V，或是需要电流超过 500 mA，都需要使用外加电源。相对于之前其他沟通介面仅能传递信息资料，高电压 USB 插槽本身还能提供 5 V 的主动电压及 0.5 A 的电流，因此对于一些小型设备而言，可以不必再外接电源供应装置，就能利用来自 USB 插槽的电力顺利运作。利用这个特点，也有厂商开发出适当的排线，将 USB 拿来当作供电插座使用，例如作为移动电话的充电器；或是供给小型电灯的电力需要，而与原本用来连接计算机的主要用途无关。

1.3 USB 设备开发流程

图 1-3 为 USB 设备开发流程图。USB 设备开发流程相当简单，①首先确认 USB 系统的开发类型，是 USB 主机、USB 设备还是 OTG；②如果确定是USB设备，必须明确该设备的类型：HID、UDIO、CDC、HUB、IMAGE 等；③查找相关设备手册，确定其描述符；④完成描述符后，编写 USB 枚举程序，观察是否枚举成功，如果枚举成功，此设备开发已经完成大部分；⑤编写应用程序，在枚举成功后，主要是进行数据处理，编写应用程序。以上就是 USB 设备开发的主要流程，其中最主要的是枚举过程，枚举不成功，该设备就不能称为 USB 设备，更不能完成 USB 设备所赋予的任务。

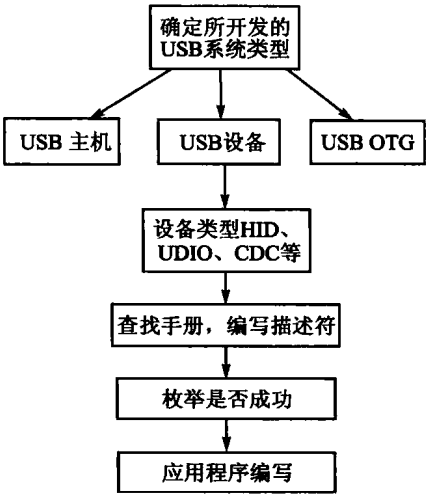


图 1-3 USB 设备开发流程图

1.4 USB 设备枚举

对于 USB 设备开发来说，最重要的是枚举，即让主机知道设备的相关信息。若枚举不成功，则设备无法识别、更不能使用。本节主要讲枚举过程，有关设备的其他信息，请参阅 USB 官方协议手册。USB 设备的属性通过一组描述符来反映它们，这些描述符是具有一定格式的数据结构，主机软件可通过 GET_DESCRIPTOR 请求获取这些描述符。每一个描述符的第一个字节表明本描述符的长度，其后是一个字节的描述符类型信息。如果描述符中的长度域值小于描述符的定义长度，此描述符被认为是非法的，不能被主机接收；如果返回描述符中的长度域值大于描述符的定义长度，则过长部分被忽略。

1.4.1　USB 设备请求

设备描述符是表征对该设备及所有设备配置起全程作用的信息。在设备枚举时，主机使用 GET_DESCRIPTOR 控制指令直接从设备端点 0 读取该描述符，一个 USB 设备只能有一个设备描述符。USB 设备与主机连接时会发出 USB 请求命令。每个请求命令数据包由 8 个字节(5 个字段)组成，具有相同的数据结构。表 1－1 为 USB 设备请求数据包格式表：

表 1－1　USB 设备请求数据包格式表

偏移量	域	大　小	值	描　述
0	bmRequestType	1	位	请求特征
1	bRequest	1	值	请求命令
2	wValue	2	值	请求不同，含义不同
4	wIndex	2	索引	请求不同，含义不同
6	wLength	2	值	数据传输阶段，为数据字节数

标准的 USB 设备请求命令是用在控制传输中“初始设置步骤”里的数据包阶段(即 DATA0，由 8 个字节构成)。标准 USB 设备请求命令共有 11 个，大小都是 8 个字节，具有相同的结构，由 5 个字段构成(字段是标准请求命令的数据部分)，结构如下(括号中的数字表示字节数，首字母 bm、b、w 分别表示位图、字节和双字节)：

bmRequestType(1)＋bRequest(1)＋wvalue(2)＋wIndex(2)＋wLength(2)

在 USB 2.0 规范里定义的标准 USB 设备请求结构体数据头如下：

```
typedef struct
{
    //定义请求的方向和类型
    unsigned char bmRequestType;
    //请求类型
    unsigned char bRequest;
    //根据请求而定其实际含义
    unsigned short wValue;
    //根据请求而定，通常提供索引或偏移
   unsigned short wIndex;
    //在数据传输阶段时，指示传输数据的字节数
    unsigned short wLength;
}
tUSBRequest;
```

(1) bmRequestType 参数

```
//位7,说明请求的传输方向
//输入
#define USB_RTYPE_DIR_IN          0x80
//输出
#define USB_RTYPE_DIR_OUT         0x00
//位6:5,定义请求的类型
#define USB_RTYPE_TYPE_M          0x60
#define USB_RTYPE_VENDOR          0x40
#define USB_RTYPE_CLASS           0x20
#define USB_RTYPE_STANDARD        0x00
```

USB_RTYPE_TYPE_M 用于提取请求中的 6：5 位，并通过 6：5 位判断请求类型，大多数为 0x00，即标准请求(USB_RTYPE_STANDARD)。

```
// 位4：0,定义接收者
#define USB_RTYPE_RECIPIENT_M     0x1f
#define USB_RTYPE_OTHER           0x03
#define USB_RTYPE_ENDPOINT        0x02
#define USB_RTYPE_INTERFACE       0x01
#define USB_RTYPE_DEVICE          0x00
```

USB_RTYPE_RECIPIENT_M 用于提取请求中的 4：0 位，并能通过 4：0 位判断请求的接收者。0x00，设备；0x01，接口；0x02，端点；0x03 为其他请求。

(2) bRequest 参数

```
//标准请求的请求类型
#define USBREQ_GET_STATUS         0x00
#define USBREQ_CLEAR_FEATURE      0x01
#define USBREQ_SET_FEATURE        0x03
#define USBREQ_SET_ADDRESS        0x05
#define USBREQ_GET_DESCRIPTOR     0x06
#define USBREQ_SET_DESCRIPTOR     0x07
#define USBREQ_GET_CONFIG         0x08
#define USBREQ_SET_CONFIG         0x09
#define USBREQ_GET_INTERFACE      0x0a
#define USBREQ_SET_INTERFACE      0x0b
#define USBREQ_SYNC_FRAME         0x0c
```

当 bmRequestType 为标准请求(即设备请求的第 7 位为 0)时，bRequest(第 6：5 位)参数提示当前请求类型，以上请求类型经常在枚举时使用。

(3) wValue 参数

① USBREQ_CLEAR_FEATURE 和 USBREQ_SET_FEATURE 请求命令时：