

中山大学

硕士学位论文

基于Linux的蓝牙HCI层协议的实现

姓名：汤杰

申请学位级别：硕士

专业：无线电物理

指导教师：黄晓

20080508

基于 Linux 的蓝牙 HCI 层协议的实现

专业： 无线电物理

硕士生： 汤 杰

指导老师： 黄 晓 副教授

摘 要

在信息化的当代，通信技术飞速发展，新技术和新标准不断涌现。而在众多的通信领域中，短距离无线通信在当今社会中的应用越来越广泛，作为一种新型的无线数据和语音通信的开放性标准，以保密性高、使用方便、功能强大、价格低廉，功耗低等优点，受到各行各业的青睐。另一方面，Linux 操作系统是开放源代码的代表，拥有卓越的功能和性能，而且它日趋成熟，受到巨大的嵌入式设备市场的重视，许多嵌入式应用产品都是采用 Linux 为系统平台。所以，本课题的所有的设计和实现工作都是建立在 Linux 系统的基础上。

本课题以蓝牙规范为基础，Linux 系统为平台，常见的蓝牙适配器作为实现工具。首先，分析目前 Linux 用户空间和内核空间数据交换的方式和网络协议编程，在 Linux 系统的底层上对蓝牙协议栈 BlueZ 的结构进行分析，重点对蓝牙核心规范的 HCI（主机控制接口）的研究，分析蓝牙基带层和主机的通讯机制以及通讯方法。其次，根据 HCI 协议的工作特性，分成初始化和连接两个阶段，对应为主机控制模块和连接控制模块。详细地介绍了指令分组、事件分组、数据分组的数据格式，并对每种分组类型都给出了一个具体的实例，最后，通过实际操作解析了 HCI 协议中对蓝牙设备的初始化、查询其它设备、建立链接、数据传输、断开链接的过程，并给出了软件流程图和 HCI 一般通信流程的实例。

本文在论述过程中，穿插讨论了在开发过程中遇到的困难和及其解决思路，并给出一些关键的程序代码。

实验证明，通过所设计的 HCI 协议，使得两台 PC 主机能通过蓝牙设备建立无线连接，查看对方设备信息和传输 ACL 数据。

关键词： 蓝牙， Linux， HCI， 无线连接

The realization of HCI protocol of Bluetooth on Linux system

Major: Radio Physics
Name: Tang Jie
Supervisor: Associate Professor Huang Xiao

ABSTRACT

In recent years, the developments of wireless mobile communication technology have changed with each new day; what's more, new technology and new practical standard have been established incessantly. In so many correspondence domains, short distance wireless communication is used more and more widely in our life and impacts our lives nowadays. As a new open telecommunication protocol for transmitting wireless data and voice, Bluetooth technology is more suitable to be the way to realize the wireless communication because of its characteristics of high reliability, low cost, small volume, and low power loss. The Linux system has many advantages such as stability, efficiency, wide hardware support and free, open codes provided. The high performance and excellent functions make Linux system useful and has a low cost, so that it has high practical value and market potential. Thus, Linux becomes the preferred operating system for many embedded manufacturer, and it has tremendous values and potentials in the embedded market. In the future, as a wireless media of WLAN(Wireless Local Area Network), the device or product that is designed by Bluetooth technology combining with embedded technology will be indispensable to life.

According to the analysis on the wholesome framework of specification of Bluetooth technology, this thesis develops the HCI protocol based on normal Bluetooth dongle and Linux operating system. At first, the thesis made a survey of the approach of data exchange between users-space and kernel-space, explained the structure of Bluetooth specification. Then, according to the characteristic of HCI layer, there are two parts, initialization and connection. So that, we divided HCI protocol into two processes: host control module and link control module. This thesis expatiates the format of the commands, events and data, and provides a practical illustration for every type.

The process of initializing Bluetooth equipment, inquiring and requesting create a connection to other Bluetooth equipments, data transmission and disconnection is analyzed in detail. At last, it presents flow diagram of the HCI protocol and the generic process of communication.

This thesis discourses above four parts and with some trouble and the solution in the process during the design process, and gives some important code.


The experiment results proved that the HCI protocol designed in this thesis performed very well, which reaches our prospective effect.

Key word: Bluetooth, Linux, HCI, wireless connection

原创性声明

本人郑重声明：


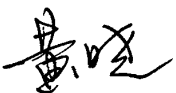
所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名： 

日期： 2008 年 5 月 8 日

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名：  导师签名： 

日期：2008年5月8日

日期：2008年5月8日

第1章 绪论

1.1 蓝牙技术的应用现状和发展前景

近年来, 蓝牙技术在无线通信技术中已经占有一个相当重要的地位, 不但数据的传输速率越来越快, 安全性、易操作性等方面都得到了相当大的改善。同时, 蓝牙芯片的体积不断的缩小, 价格也已经下降到 3 美元, 使得它成为了众多手持设备、嵌入式设备首选的近距离无线通信技术。

蓝牙 (Bluetooth) 技术, 使用全球通行的、无需申请许可的 2.4GHz 频段, 可进行实时数据和语音传输^[1], 而且有较高的传输质量, 因为采用伪随机跳频方案, 不同蓝牙微网之间的干扰会非常小。它是以低成本、短距离无线通信为基础, 为固定或移动设备的通信环境提供特别连接的通信技术。

蓝牙技术由爱立信公司开始最初的研究。1999 年, 蓝牙 1.0 技术规范发布, 采用无线接口来代替有线电缆连接, 具有很强的移植性, 采用调频技术, 成熟的纠错编码技术和基带协议。经过几年迅猛的发展, 爱立信公司与诺基亚 (Nokia)、英特尔 (Intel)、IBM 和东芝 (Toshiba) 公司组成了蓝牙特别兴趣小组 (Special Interest Group, SIG)^[2], 负责蓝牙技术标准的制定、产品测试^[3], 并协调各国蓝牙技术使用。后来该组织又陆续吸纳了微软 (Microsoft)、朗讯 (Lucent)、摩托罗拉 (Motorola)、3Com 等知名厂商。至今为止, 蓝牙 SIG 已经拥有 7000 多家成员公司。

2005 年, 蓝牙 2.0 技术^[4]诞生, 不但速度提升为 1.x 版的 3 倍, 电源消耗理论上也将降低 2/3。2006 年, 蓝牙技术得到真正广泛的发挥, 特别是通讯行业^[5], 批量生产使蓝牙芯片的成本迅速降低到 3 美元以内, 因此 2006 年可以算的上是蓝牙技术从研发转向普及的光明期。

值得注意的是, 蓝牙在手机中的普及会促进蓝牙在其他电子设备上的应用^[6], 比如笔记本电脑、电视、数码相机、打印机等。所以蓝牙的发展前景更为光明, 目前的无线解决方案中只有蓝牙技术和工业领域上的 ZigBee 成本最低, 因此在日常生活中蓝牙普及的概率将大幅度增加。

而关于 Linux 系统的应用现状和发展前景^[7-8]: 自 1991 年到现在, 经过了多年的积累改进和技术革新, Linux 已经从一个学生的玩具演变成一个成熟而稳定的操作系统内核, 他不仅拥有庞大的用户群, 还受到 IBM、HP、Sun、Intel、AMD、Sony 等 IT 巨头的青睐。各大软件公司如 CA、Veritas、BEA、Oracle、SAP、Borland

等也相继地成为了 Linux 的支持者。这些公司都确认“Linux 已经完全适合于企业级应用”。在系统平台领域, Linux 已经成为全球第二大操作系统。以往开放源代码社区都是以爱好者为主导,而现在大公司派出大量技术力量,去支援开放源代码社区。经历 16 年来, Linux 操作系统以惊人的速度在网络服务器和桌面系统发展中获得了成功,占据了大量份额,这已经是不争的事实。

全球用户越来越多的选择 Linux 操作系统并不是处于狂热而是一种理性的选择。Linux 操作系统是开放源代码、良好的可移植性、具有多任务、多用户的能力优秀的操作系统。Linux 操作系统遵循开放、自由的理念,我们能在 GNU 公共许可权限下免费获得的,能从 Internet 上免费下载安装。Linux 的源代码是开放的,任何人都能进行修改, Linux 操作系统从诞生开始就注定不是为了商业盈利目的开发的,世界上优秀的程序员和黑客都在改进 Linux 代码,使之更加完美,正如此, Linux 在技术上说是世界上最棒的操作系统。

据统计, Linux 可以帮助终端厂商节省超过 50% 的软件成本,使他们能够为市场提供更具价格优势的终端产品。除此之外, Linux 所具有的丰富的开发资源和庞大的开发者社区,也可以帮助终端厂商开发出更具个性化的产品。这两点对于那些中小规模的终端厂商尤其重要。

无论是在企业级的服务器^[9]中, PC 和笔记本电脑中,还是嵌入式设备^[10]中,以 Linux 为基础的操作系统都取得了相当大的进步并日渐深入人心。因此,在网络时代的信息高速公路上,反映了集体智慧和协作精神的 Linux,虽然未必能够短期内在桌面操作系统领域获得足够的份额,但起码在服务器市场有望与微软一争高下。

针对无线通信蓝牙技术无限广阔的应用前景,本实验室结合实际情况,致力于无线通信技术的研究与开发工作。

所以,本课题正是在上述的背景下提出的。

1.2 蓝牙协议的国内外研究现状

根据市场研究机构 Millward Brown 连续四年的独立调查^[11],了解消费者对 Bluetooth (蓝牙)无线技术的认知、态度和应用状况。调查访问了 2100 名年龄介乎 18 岁至 70 岁,并来自日本、美国、英国及德国的消费者。从这些受访国家显示,他们对 Bluetooth 无线技术的平均认知度由 2004 年的 60% 上升至今年的 73%。

正是因为如此,蓝牙技术已成为当今世界上的投资热点。据统计,已有数以百亿美元的资金投向了蓝牙芯片及产品开发,数以千计的厂家在全力以赴地开赴基于

蓝牙技术的产品，数以万计的工程师和技术人员热衷于蓝牙技术的研究和发展。

SIG 虽然制定出了详细的蓝牙技术规范^[12]，但是却没有对蓝牙技术在实现上进行说明。因此若想将蓝牙协议真正应用到实际产品中去，就必须将蓝牙技术规范中描述的功能加以实现到目前为止，已有的蓝牙芯片只集成了无线和基带功能，链路管理层的功能既可以由蓝牙芯片实现也可以由软件来实现，而其他高层协议还不能通过硬件实现，所以实现蓝牙协议体系中的其他高层协议，研制出符合蓝牙互操作性标准的蓝牙协议栈是蓝牙产品开发的关键。

国内外学者围绕蓝牙技术主要有以下方面的研究：蓝牙协议栈，基于蓝牙技术的嵌入式产品，组网（蓝牙微微网，蓝牙分散网）^[13]。而本课题属于协议栈开发的方向，而国内外有大量针对协议栈的文献，但是 Linux 系统上的蓝牙协议的研究和开发正慢慢起步。考虑到若要使开发出的蓝牙协议像 TCP/IP 协议那样成为蓝牙技术的通用通讯协议，就需要将该协议运行在一个开放的系统环境下。而 Linux 作为开放源代码的代表，具有符合国际通用标准、兼容性(POSIX)好、先进的网络特征、拥有多任务能力、性能稳定、可移植性好、免费等优点，是蓝牙协议开发与运行的良好平台。事实上，由于 Linux 特有优势，得到了诸如 IBM，DELL，ORACLE，SAP 等公司强有力的商业支持。

由于微软已经在其操作系统 Windows XP 中集成了蓝牙协议，鉴于微软 Windows 已经垄断了 PC 机环境，为了使我们的开发工作能够得到保护，使开发成果能够得到真正的应用，所以本课题采用开源的 Linux 操作系统。事实上，现在每个人都生活在一个充满嵌入式系统的世界里。从手表，电话到手机，PC 都有嵌入式系统的影子。

1.3 主要研究内容

✧ 研究内容

1. 分析目前 Linux 用户空间和内核空间数据交换^[14]的方式，来建立用户空间和内核空间数据交换的通道以进一步提高通信效率。熟悉 Linux 网络驱动程序的结构和层次，以及如何实现，包括硬件架构，驱动程序的基本概念和基本方法^[15]。
2. 通过本课题的研究，开发出对蓝牙规范兼容的基于嵌入式系统的蓝牙协议栈，在实验室上一届研究人员实现 Linux 系统内核空间下的 USB 蓝牙设备驱动程序的基础上，在用户空间实现 HCI 协议，通过该协议栈各模块提供的功能使两台蓝牙设备间能够达到无线通讯的目的。

◆ 创新点

1. 在 Linux 系统的底层上对蓝牙协议栈 BlueZ 的结构进行分析，重点对蓝牙核心规范的 HCI（主机控制接口）的研究，分析蓝牙基带层和主机的通讯机制以及通讯方法，在此基础上方便地进行蓝牙 HCI 层的开发，总结在该协议栈下进行编程的思路。
2. 蓝牙开发是一个比较复杂的过程，除了复杂的硬件设计外，还有针对各种应用模型的软件设计，为完成蓝牙的各种功能，软件的重要角色是不可替代的。本课题所实现的蓝牙通讯协议具有良好的移植性与扩展性，无论在理论或软件方面，都为日后创新型的嵌入式产品打好基础。
3. 常规的蓝牙协议栈一般运行于系统内核空间上，而在内核空间开发和测试难度很大，而如果在用户空间实现协议栈，把蓝牙设备映射成文件系统中的文件，使得用户空间的程序能够通过标准的系统调用来访问设备，将会避免在开发过程中可能频繁出现的系统崩溃。

1.4 论文结构

本论文具体安排如下：

第一章，介绍了本课题的背景与意义。

第二章，由浅入深，首先对蓝牙协议栈进行简介，接着重点分析 HCI 协议，以及本课题的总体设计方案。

第三章，详细分析了 Linux 下的基于 USB 的蓝牙驱动程序。

第四章，通过对主机控制模块在 HCI 协议中位置的介绍，阐述了该模块的设计和实现。

第五章，通过对连接模块在 HCI 协议中位置的介绍，阐述了该模块的设计和实现。

第六章，对实现过程中的操作进行了介绍，包括调试、改进和结果分析。

第七章，对本文进行概括总结，并对今后相关的研究和开发工作进行展望。

第2章 总体设计方案

2.1 蓝牙协议栈概述

协议就是设备间交换信息所遵循的规则。任何类型的网络拓扑结构，都有一套协议或规则来详细定义消息如何在链路上传输，蓝牙技术标准也不例外。于是，SIG 制定了蓝牙技术规范，方便开发基于具有可互操作的无线模块和交互式服务应用。

为了实现互操作，在远程设备上的对应应用程序必须运行在相同的协议栈上。不同的应用可运行于不同协议栈。并不是所有的应用程序都使用全部的协议。相反，应用程序一般只使用协议栈中的某些部分。所以，蓝牙协议采用了类似于 OSI 的分层体系结构。

蓝牙技术标准的开放性使设备制造商的应用程序可以方便地使用遵守蓝牙技术标准的软硬件系统。在蓝牙技术规范协议的上面，制造商或者研究人员还可以自由地实现他们自己的或是通用的应用协议。完整的协议栈层次结构如图 2-1 所示。

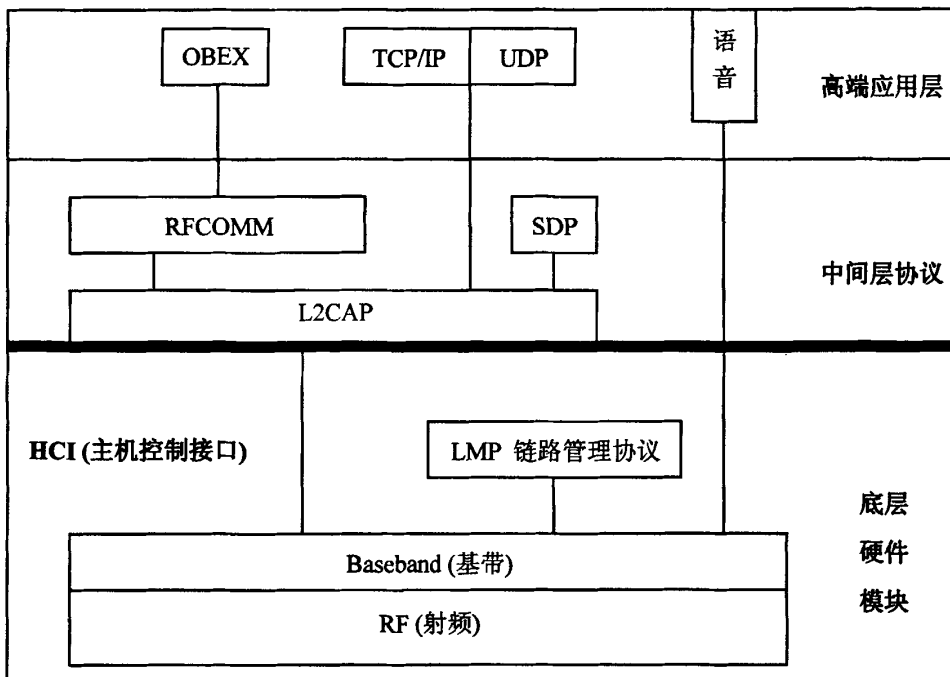


图 2-1 蓝牙协议栈

SIG 将蓝牙协议栈体系结构中的协议分为四层^[16]:

- (1) 核心协议: 基带 (baseband), 链路管理器 (LMP), 逻辑链适配协议 (L2CAP), 服务发现协议 (SDP)。
- (2) 电缆替代协议: 串行仿真协议 (FRCOMM)。
- (3) 电话传送控制协议: 电话控制协议 (TCS-BIN), AT 指令集。
- (4) 可选协议: PPP, TCP/UDP/IP, 对象交换协议 (OBEX), 无线应用协议 (WAP), WAP 应用环境 (WAE), IrMc。

2.2 HCI 协议

蓝牙协议栈可以用集成的方式实现, 全部集中在同一台主机 (主板或处理器) 上, 该主机上运行了使用蓝牙协议栈的应用程序^[16]。另外, 它们也可以独立于主机实现。这时, 单独的蓝牙模块将作为附加的附件或插卡, 通过主机上的一些物理接口 (如: USB, RS232 等) 连接到主机上。在分开实现时, 蓝牙模块还包含一个主机控制器单元, 用来解释从主机接收到的信息并将其正确地发送到模块中合适的组件中去。类似的, 主机控制器收集来自蓝牙模块的数据和硬件/固件的状态信息, 并根据需要将其传送给主机。本文所实现的蓝牙通讯系统属于独立主机结构模式。

为了让不同厂商的非集成蓝牙模块能够互通, SIG 定义了一个标准接口与模块中的主机控制器进行通信。这个接口与主机和主机控制器之间使用的物理接口和传输机制无关。SIG 还定义了一个事务型通信协议, 用于在主机和主机控制器之间传送信息。这种主机和主机控制器间的标准接口和它们之间相应的通信协议一起被称为主机控制器接口 (Host Controller Interface, HCI)。

到目前为止, 规范中 HCI 是最长的一部分。从某种意义上说, HCI 的能力就代表了蓝牙技术可实现的功能。HCI 定义了蓝牙模块的一组功能, 它们可被主机及其应用访问。由此可见, HCI 相当于蓝牙模块能提供给用户的服务看守者。

HCI 部分包含一组到高层的接口^[4]; HCI 还有一个 HCI 驱动程序, 它通过执行通信协议与主机控制器交换信息; 最后还包括在物理接口间传送数据的 HCI 传输协议 (也叫 HCI 传输)。对于 HCI 传输, SIG 没有开发任何新的协议, 而是使用了三个已有的协议: 通用串行总线 (Universal Serial Bus, USB) 协议, RS-232 串口协议和通用异步接收/发送 (Universal Asynchronous Receive and Transmitter, UART) 协议, UART 协议是 RS-232 协议的一个固有子集。

蓝牙系统的协议模型如图 2-1 所示，而下图（图 2-2）是蓝牙软件层次的简单结构，从这两幅图可以看出，HCI 是位于蓝牙系统的 L2CAP（逻辑链路控制与适配协议）层和 LMP（链路管理协议）层之间的一层协议。但其实 HCI 层的地位比较特殊，它并不是严格意义上的通信协议^[3]。它一般位于 L2CAP 之下，但有时候可以位于其之上，在一个集成度很高的嵌入式系统中甚至可以不存在。但 HCI 提供对基带控制器和链路管理器的命令接口以及对硬件状态和控制注册成员的访问，该接口还提供了进入蓝牙基带的统一方式，所以是整个蓝牙协议体系中十分重要的部分。HCI 对多数的工控、智能仪器仪表应用具有非常特殊意义。通常，射频、基带和链路管理器封装一个固件，形成一个具有蓝牙无线通信功能的模块，例如常见的蓝牙适配器，而 HCI 就提供了主机与蓝牙模块之间的统一操作接口。在简单的缆线替代应用场合，蓝牙的核心协议之一——L2CAP 可以不需要实现，而通过 HCI 直接操控链路管理器，访问基带和寄存器等硬件。

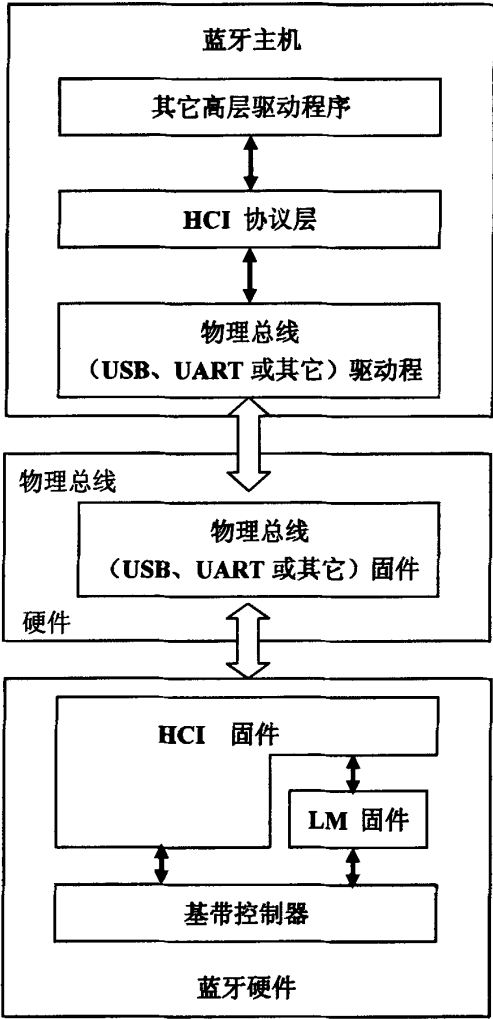


图 2-2 蓝牙协议层剖析

正如上文所说，HCI 形成了软件协议栈与其下的链路管理程序之间的接口，后者在蓝牙设备固件中实现。注意，这是面向 HCI 与链路管理程序之间而不是面向 HCI 与设备驱动之间的通信。因为 HCI 不直接访问蓝牙设备的寄存器和内存空间。HCI 是向设备发送指令和数据封包，并且从该设备接收数据封包和事件消息封包。

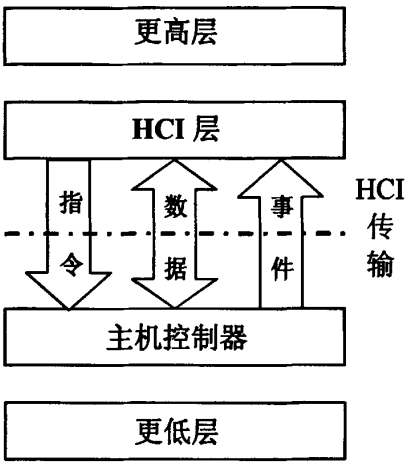


图 2-3 HCI 信息交换示意图

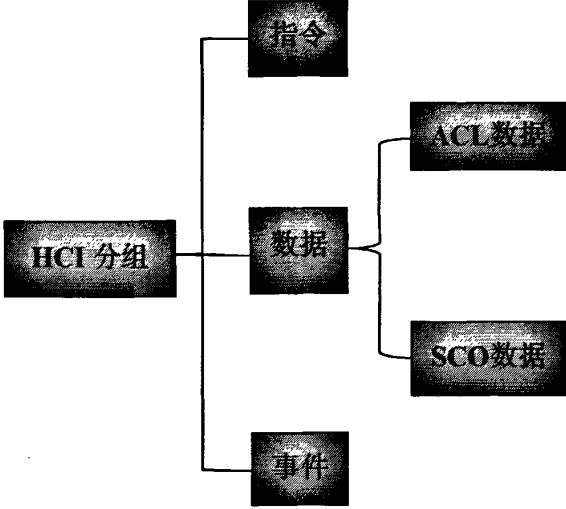


图 2-4 HCI 分组示意图

图 2-3 和图 2-4 介绍 HCI 信息交换和数据分组的情况。在 HCI 中，有三种类型的数据包用于主机的 HCI 层与蓝牙模块中主机控制器之间的信息交换。指令（command）类的数据包携带从 HCI 层发往主机控制器的控制和管理信息，事件（event）类的数据包携带从主机控制器发往 HCI 层的控制和管理信息，数据（data）类的数据包则是双向的，携带 ACL 和 SCO 数据的分片数据。有关 ACL 和 SCO，在后面有更详细的论述。

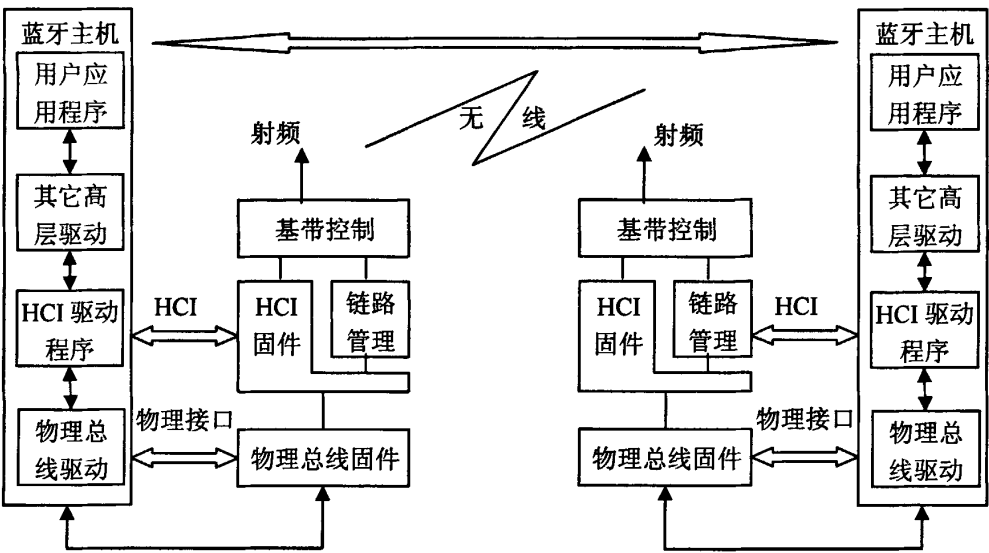


图 2-5 HCI 的数据传输模式

基于 HCI 的数据传输模式如图 2-5 所示。射频、基带控制器、HCI 固件及链路管理构成了蓝牙硬件模块。对于上层数据而言，蓝牙模块唯一可以识别的命令是 HCI 指令。在主机 1 和主机 2 中包含了高层驱动、HCI 驱动以及各种接口（USB 或 UART 等）的驱动。而在蓝牙模块中则包含有 HCI 固件层，用来处理主机传来的 HCI 指令和数据，然后再传向链路管理固件和基带硬件，最后通过射频发出。而接收端正好是一个逆过程。主机和蓝牙模块之间是通过物理接口（本课题为 USB）来连接和通信的。

2.3 设计方案

因为基于本实验室的往届成果，目前已经实现 USB 蓝牙设备的内核驱动，完成了传输层的驱动工作，在这个基础上，逐步根据实际需要完成用户空间的协议栈，以实现基于 USB 蓝牙设备的 Linux 平台间的无线通讯。

论文总体设计：

- Linux 用户空间和内核空间数据交换。
- Linux 网络驱动程序的结构和层次。
- 蓝牙协议栈的层间交互方式。
- 编写 HCI 协议，建立基于 USB 蓝牙设备的两台主机无线通讯。

所以，协议的设计必须建立在对这些关键技术的熟悉掌握的基础上：蓝牙官方规范；熟悉 Linux 内核和用户空间编程（网络部分）；在代码级理解协议运行流程，整体设计思路。

以蓝牙官方协议为基础，基于应用而进行简化，增强以及改进，熟练 Linux 系统关键技术 PC 机间基于 HCI 指令的蓝牙串口通信的实现，第一步，是必须理解 HCI 的各种分组格式，明白各指令对应的十六进制表示形式，既要与底层驱动程序交换数据，又要向高一层的协议提供接口。第二步，则需要对 Linux 的系统机制不断摸索，并实现蓝牙通讯协议。

所以，根据蓝牙设备的特点和官方协议的规范，它主要完成蓝牙芯片初始化、蓝牙链路管理命令的编辑、链路建立、链路监视、链路拆除以及数据的打包和拆包等功能，采用最适于对硬件操作的 C 语言来实现。

本章给出了基于 USB 设备的 Linux 蓝牙 HCI 层协议的构建目标，对其总体设计方案进行详细的阐述，并且分析了开发过程中的关键技术和技术难点。下文将论述详细的设计开发过程。

第3章 Linux 下的蓝牙驱动程序

3.1 Linux 下设备驱动程序概述

3.1.1 内核空间(kernel-space) 和 用户空间(user-space)

作为一个 Linux 开发者, 首先应该清楚内核空间 and 用户空间的区别^[17]。关于这个话题, 已经有很多相关资料, 这里作一个简单描述:

现代的计算机体系结构中存储管理通常都包含保护机制。提供保护的目的是, 要避免系统中的一个任务访问属于另外的或属于操作系统核心的存储区域。如在 IntelX86 体系中, 就提供了特权级这种保护机制, 通过特权级别的区别来限制对存储区域的访问。基于这种构架, Linux 操作系统对自身进行了划分: 一部分核心进程独立于普通应用程序, 运行在较高的特权级别上, 它们驻留在被保护的内存空间上, 拥有访问硬件设备的所有权限, Linux 将此称为内核空间。

相对地, 其它部分被作为应用程序在用户空间执行。它们只能看到允许它们使用的部分系统资源, 并且不能使用某些特定的系统功能, 不能直接访问硬件, 不能直接访问内核空间, 当然还有其他一些具体的使用限制。

从安全角度考虑, 将用户空间和内核空间置于这种非对称访问机制下是很有效的, 它既能抵御恶意用户的窥探, 也能防止质量低劣的用户程序的侵害, 从而使系统运行得更稳定可靠。但是, 这会产生另外一个极端: 像这样完全不允许用户程序访问和使用内核空间的资源, 那么系统也就无法提供任何有意义的功能了。为了方便用户程序使用那些必须在内核空间才能完全控制的资源, 而又不违反上述的特权规定, 从硬件体系结构本身到操作系统, 都定义了标准的访问界面和相应接口^[18]。

出于效率和代码大小的考虑, 内核程序不能使用标准库函数^[19]。因此内核开发不如用户程序开发那么方便。但是当前几个主要的开源蓝牙协议栈, 包括 BlueZ, 均工作于内核空间, 这意味着在协议栈上的任何改动都有可能导致内核的不稳定甚至崩溃。为了避免在研究过程中出现频繁的系统崩溃和重新启动, 把蓝牙协议栈迁移到用户空间来是必要的。而且, 在内核空间进行程序开发也比在用户空间进行程序开发的难度要大得多, 有效的调试工具也并不多, 很多情况下只能通过系统提供

的一些 oops⁽¹⁾信息来对错误进行诊断。

3.1.2 系统调用

在用户空间的程序对系统资源的访问必须通过操作系统所提供的标准接口来进行，并接受操作系统的调度。一般的硬件体系机构都提供一种“门”机制。“门”的含义是指在发生了特定事件的时候低特权的应用程序可以通过这些“门”进入高特权的内核空间。对于 IntelX86 体系来说，Linux 操作系统正是利用了“系统门”这个硬件界面，构造了各种各样的系统调用作为软件界面，为应用程序从用户态嵌入到内核态提供了通道。通过“系统调用”来使用“系统门”并不需要特别的权限，但嵌入到内核的具体位置却不是随意的，这个位置由“系统调用”来指定，有这样的限制才能保证内核安全无虞。所以，系统调用是用户程序跟操作系统交互的通道。他们之间的关系可以用图 3 - 1 表示。

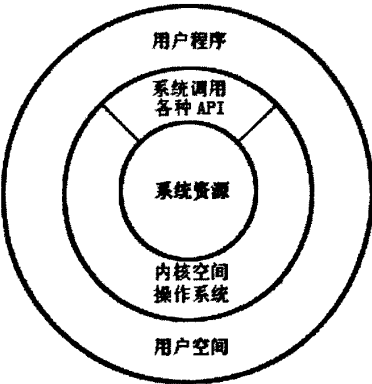


图 3 - 1 用户空间内核空间和系统资源的关系

系统调用是用户级程序访问内核最基本的方法。目前 Linux 大致提供了二百多个标准的系统调用，并且允许我们根据实际需要，添加新的系统调用来实现指定进程和内核间的信息交换。由图 3 - 2 可看出整个系统的软硬件资源关系，其中标有橙色箭头的表示编写本驱动程序需要重点把握的内容。

¹ 把计算机拟人化，表示出现异常情况时，计算机出现的反应。

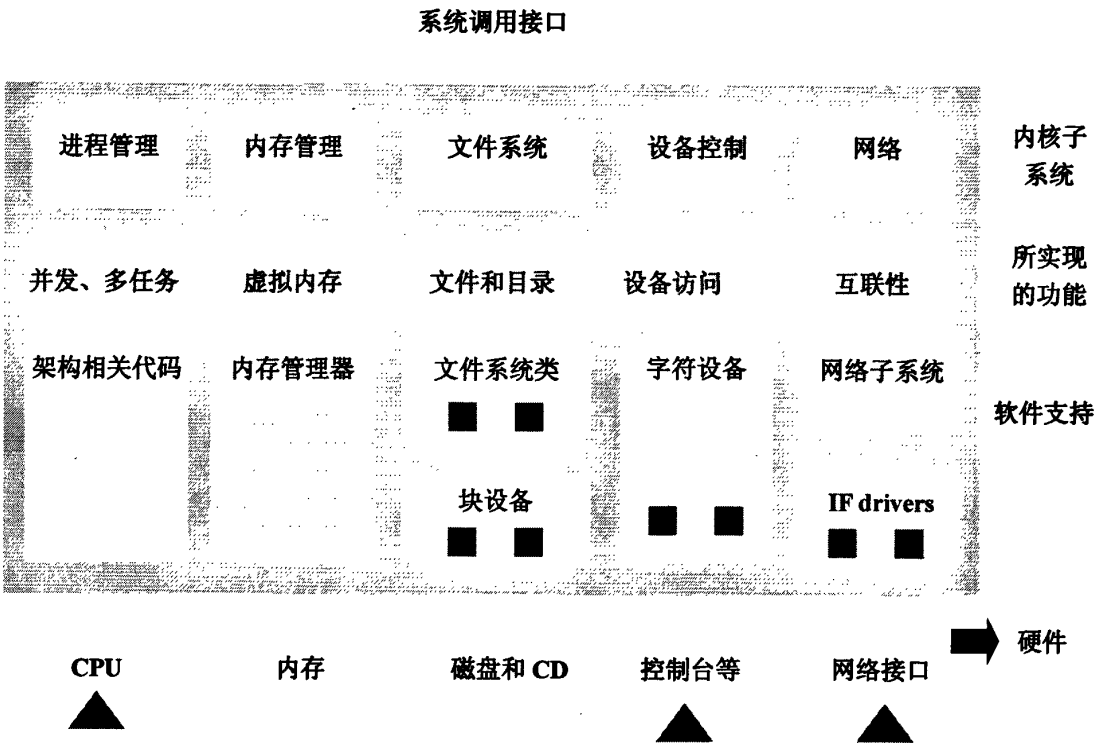


图 3-2 系统资源关系

3.1.3 驱动程序

Linux/Unix 的一个特点就是把所有的东西都看作是文件^[20]。而设备驱动程序在 Linux 中扮演一个非常重要的角色，几乎所有系统可用的设备都需要各自的驱动程序来协助访问。这些驱动程序隐藏了关于设备操作的实际细节和复杂性，对外提供一个定义良好的内部编程接口，使得用户可以通过一组标准化的调用来操作设备，而不需要关心设备实际的操作细节。设备驱动程序将这些标准化的调用映射成实际作用于设备的特有操作。系统定义了简洁完善的驱动程序界面，客户程序可以用统一的方法透过这个界面和内核驱动程序交互。而使用者和开发者已经非常熟悉这种开发流程了。

驱动程序运行于内核空间，用户空间的应用程序通过文件系统中/dev/目录下的一个相对应的文件来和它交互。这就是我们熟悉的那个文件操作流程^[21-22]：
`open()→read()→ write()→ioctl()→close()`。

这里我们集中讨论这一部分：在内核中，设备驱动程序与用户级程序的交互。操作系统为此定义了一种统一的交互界面，就是前面所说的 `open()`, `read()`, `write()`,

ioctl()和 close()等等。每个驱动程序按照自己的需要做独立实现，把自己提供的功能和服务隐藏在这个统一界面下。客户级程序选择需要的驱动程序或服务，按照上述界面和文件操作流程，就可以跟内核中的驱动交互了。在面向对象的角度来分析，系统实际上定义了一个抽象的界面（abstract interface），每个具体硬件的驱动程序都是这个界面的实现（implementation）。

驱动程序是用户空间和内核信息交互的重要方式之一。其实 ioctl, read, write 本质上讲也是通过系统调用去完成的，只是这些调用已被 Linux 内核进行了标准封装，统一定义。因此用户不必修改内核代码来添加新系统调用和重新编译新内核。需要使用虚拟设备，通过模块方法^[23-24]将新的虚拟设备安装到内核中（insmod 上），就能方便使用新设备。

在 Linux 中，设备大致可分为：字符设备，块设备，和网络接口（字符设备包括那些必须以顺序方式，像字节流一样被访问的设备；如字符终端，串口等。块设备是指那些可以用随机方式，以整块数据为单位来访问的设备，如硬盘等；网络接口，就指通常网卡和协议栈等复杂的网络输入输出服务）^[13]。前两种设备通常都会在/dev/目录下建立一个映射节点。网络设备通常与一个网络接口相关联，由内核的网络子系统驱动，但它不是一个面向流的设备，因此将网络接口映射到文件系统中的节点会比较困难，它通过另外一套完全不同于字符设备和块设备的接口进行访问。本文正是要将原来被挂接到 Linux 网络子系统得蓝牙网络设备映射到文件系统中，这个蓝牙设备驱动工作于 USB 子系统下。由于各个总线子系统隐藏了具体总线的实现逻辑并提供了相关的编程接口，驱动开发人员就可以专心实现面向设备的总操作逻辑，大大简化了驱动开发的过程并减少代码的重复。

Linux 还提供了一个非常重要而灵活的特性来方便开发人员进行内核空间的程序开发——运行时的内核特性扩展。就是说我们能动态对内核进行添加或者删减模块，向内核添加功能是通过在运行时向内核添加“模块”来实现的，每个模块实际上是没有连接成完整可执行程序的目标代码，我们可以通过 insmod 命令在运行时把模块连接如内核，或者用 rmmod 命令动态的移除一个模块。这使得驱动程序的开发能在不重新编译内核，甚至不重启系统的情况下测试程序，大大地降低了驱动开发的复杂性。

3.2 基于 USB 的蓝牙设备概述

本课题采用的是基于 USB 端口的蓝牙适配器，所以下面将介绍采用 USB 作为数据通信接口的原理和技术。

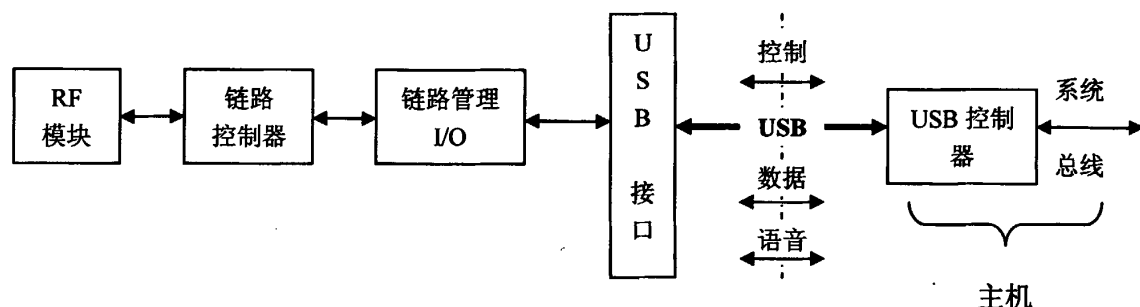


图 3-3 基于 USB 的蓝牙模块框图

由图 3-3 显示了蓝牙设备通过 USB 连接到主机 PC 的框图。USB 其中一个很重要的特性是它只担当设备和主控制器之间通信通道的角色，对它所发送的数据没有任何特殊的内容和结构上的要求⁽²⁾。USB 可以通过一个物理信道处理多个逻辑信道，因此，控制、数据和语音信道不需要其它的物理接口。但要注意的是，在通过 USB 实现的蓝牙模块中没有对寄存器和内存的直接访问，这些功能是通过使用适当的 HCI 命令和使用主机控制器传输层接口实现的。

所有的 USB 设备都可以通过 Linux 的 USB 子系统进行访问。我们可以把每一个 USB 设备的驱动分成两个部分：总线的驱动和设备的驱动^[17]。所谓总线的驱动可以看成是对某种类型总线逻辑的软件支持，我们可以把它看成是 Linux 的某个总线子系统。也就是说，Linux 的 USB 子系统实际上已经实现了对 USB 总线的访问操作。另外一部分则是设备的驱动。USB 总线可以连接到各种各样的 USB 设备，每一种的 USB 都有它独特的访问操作和工作方式。为了能够让设备正常工作，我们还需要为 USB 设备编写相应的驱动程序。

在 Linux 系统下 USB 设备可以抽象成下图的结构：

² 实际上，还是存在一些结构，但通常被降低标准：例如，键盘不需要分配带宽，而某些摄像头需要。

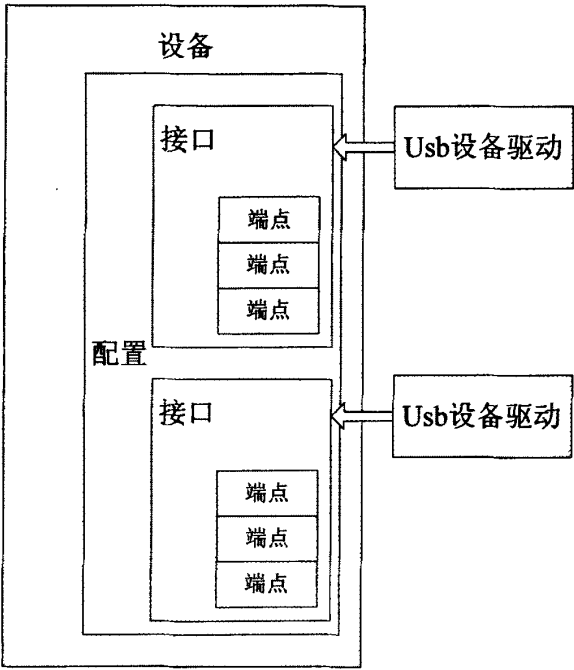


图 3-4 USB 设备概观

USB 通信^[25]最基本的形式是端点。USB 端点只能往一个方向传送数据，从主机到设备（输出端点）或者从设备到主机（输入端点）。端点可以看作是单向的管道。

USB 端点被捆绑为接口^[26]。USB 接口只处理一种逻辑连接，例如鼠标、键盘或者音频流。一些 USB 设备具有多个接口，例如 USB 扬声器可以包括两个接口：一个 USB 键盘用于按键和一个 USB 音频流。因为 USB 接口代表了一个基本功能，而每个 USB 驱动程序控制一个接口。不同的 USB 接口可以由不同的设置，用来以不同的方式控制同一个设备的端点。

USB 接口被捆绑为配置。一个 USB 设备可以有多个配置，而且可以在配置之间切换以改变设备的状态。

概言之，USB 设备是非常复杂的，由许多不同的逻辑单元组成，通过对这些逻辑单元之间的关系的把握，可以产生不同的方法去控制同一个设备。

3.3 实现及分析

前面两节讲述了在 Linux 系统下进行内核程序开发的一些基础以及蓝牙设备的 USB 驱动开发概要，本节将介绍对 USB 蓝牙设备驱动改写中的技术细节，包括关于该设备驱动对并发访问提供的支持、阻塞型 I/O 以及内核空间跟用户空间的通

信技巧等。

3.3.1 驱动程序的主要数据结构

对于一个 USB 的设备驱动来说，他必须实现若干个系统已经定义好的数据结构^[27]。这些数据通常用于向内核进行某种注册。

首先第一个是 `usb_driver` 结构体^[20]，他的定义如下：

```
static struct usb_driver hci_usb_driver = {
    .owner = THIS_MODULE,           //结构的所有者
    .name = "hci_usb",              //结构的名称
    .probe = hci_usb_probe,         //设备接入初始化函数
    .suspend = hci_usb_suspend,     //设备挂起函数
    .resume = hci_usb_resume,       //设备恢复函数
    .disconnect = hci_usb_disconnect, //设备拔出清理函数
    .id_table = bluetooth_ids,      //驱动所支持的设备 ID 列表
};
```

这个结构将用于向系统注册新的 USB 驱动程序和该驱动所支持的设备的 ID。当设备接入系统后，内核通过注册的 `id_table` 确定对应的驱动程序，并调用该驱动的 `probe` 函数。而当设备与主机断开连接时，相应的 `disconnect` 函数就会被调用。

因为本文要完成的驱动需要在文件系统中创建相应节点，所以还需要用到另外两个数据结构，`usb_class_driver` 和 `file_operations`，定义如下：

```
static struct file_operations hci_usb_fops = {
    .owner = THIS_MODULE,
    .read = hci_usb_read,
    .write = hci_usb_write,
    .open = hci_usb_open,
    .release = hci_usb_release,
};
```

```
static struct usb_class_driver hci_usb_class = {
```

```
.name = "hci_usb%d",  
.fops = &hci_usb_fops,  
.minor_base = HCI_USB_MINOR_BASE,  
};
```

`usb_class_driver` 结构用于向系统注册映射到 `/dev/` 目录下的文件节点, 所以它需要一个次设备号的起始数值。对于 USB 设备来说, `usb_class_driver` 结构体为我们提供了注册节点和文件操作函数。所以, 它还包括了一个文件操作结构指针和一个名字, 下文将要讨论的 HCI 协议就是直接对这个结构进行读写操作。

另外在驱动程序当中, 我们也需要把设备抽象成一个数据结构。对于本文所用的设备, 我们用下面的数据结构描述 `struct hci_usb`。具体的结构成员这里不一一列出, 该结构体记录了几几乎所有该蓝牙设备需要用到的资源和属性, 所有的操作都将围绕着这个数据结构进行。

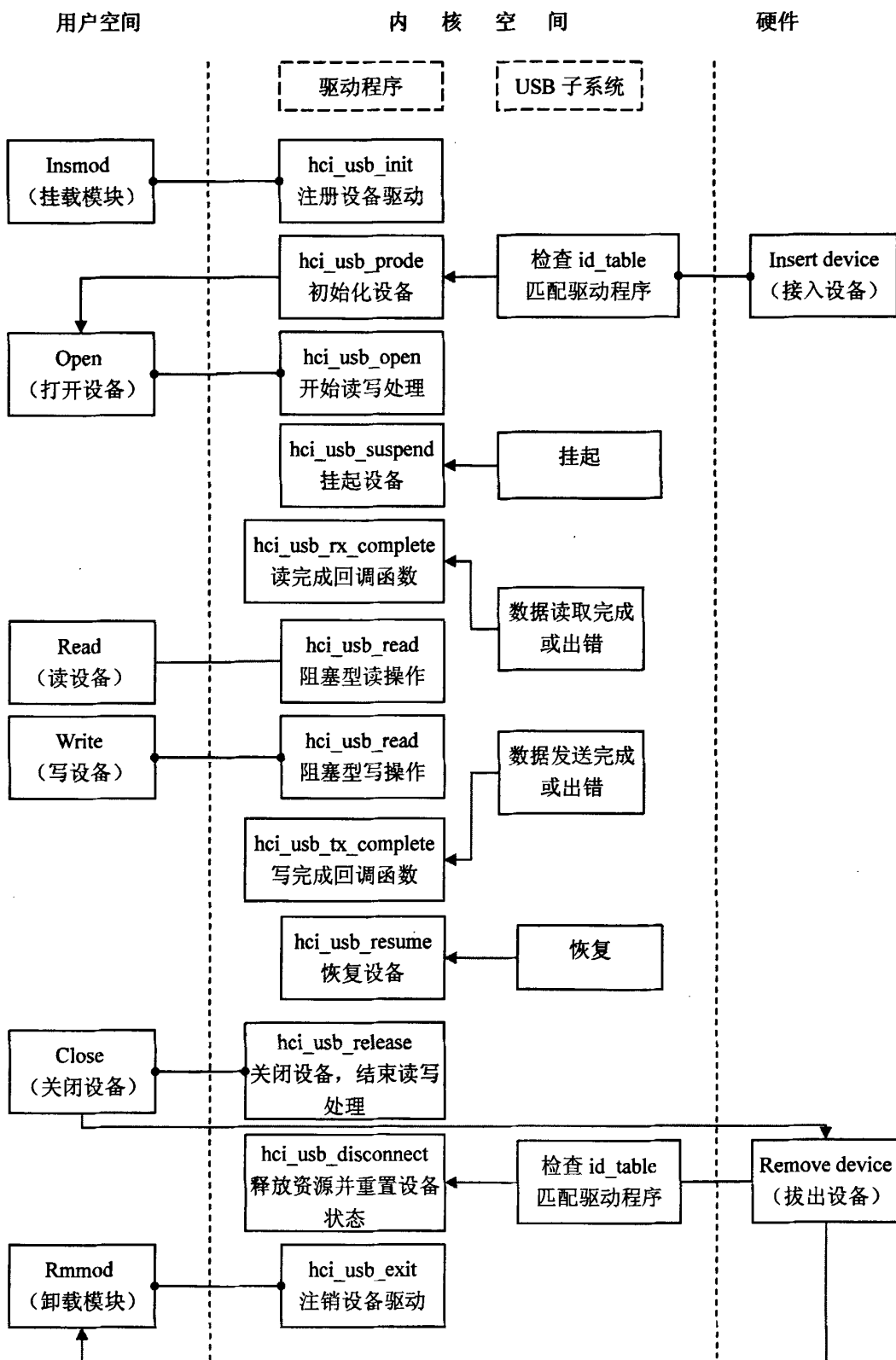


图 3-5 驱动程序的函数调用关系及流程

图 3 - 5 是驱动程序的大致结构，它阐述了程序中各个主要接口的触发事件，并表明了设备工作的大致流程。途中以 $\bullet \longrightarrow$ 代表用户空间的请求和内核空间的处理函数或者是某事件触发条件与处理例程的对应关系。而 \longrightarrow 则代表一般的操作流程。

3.3.2 并发访问

对于本设备驱动程序来说，并发访问包括两个方面：共享的数据结构和代码的临界区^[20]。

在并发访问中，为了保证数据结构在还有用的时候不会被意外释放掉（比如设备被多个进程打开时），内核需要对该数据结构的引用计数。第一个属性 `kref` 是一个计数器结构体，内核通过它维护对包含该结构体的数据结构的引用计数。它还可以注册一个清理函数，在内核对该结构的计数降为 0 时，自动调用该函数作相关的清理工作。所以我们需要对该结构使用 `kref` 结构体进行引用计数。只有当该结构已经不在被引用时，释放的操作才会发生。通常地说，只有在设备被拔出时，该结构才会被释放。

而从图 3 - 5 可以看出，很多情况下，系统中多个任务函数会试图同时地对同一个数据结构进行操作，这时，我们需要引入锁机制，保证其在某个关键操作进行时不会有其它并发的操作调用。

Linux 提供了几种不同类型的锁^[14]，它们各有不同的特性。互斥锁是一种较常用的锁，它可以有两种实现方式：信号量和自旋锁。在性能上说，自旋锁有较好的表现，但它有一个缺点，就是使用自旋锁的代码不能进入休眠。另外，Linux 提供另一种类型的锁——读写锁。它同样也有信号量和自旋锁两种实现方式，同样遵从相关使用限制，但是通过它，可以针对性地解决这种情况：进入运行阶段，系统很少改变该数据结构，却会多次读取该数据结构。在使用锁前，先要进行初始化，当代码进入临界区前首先要获得相关的锁，则立即禁止其它的写入活动，使得对数据的访问不会被干扰。这里我们采用的是读写自旋锁，在取得了锁以后，代码不能显式地休眠，也不能调用可能休眠的函数，通过试图获取该锁的进程是出于读取意图，还是写入意图而加以区分。

3.3.3 阻塞型 I/O

一般来讲，文件描述符所指定的文件或设备有两种方式：阻塞与非阻塞^[13,18]。所谓阻塞方式是指，当试图对该文件描述符进行读写时，如果当时没有东西可读，

或者暂时不可写，程序就进入等待状态，直到有东西可读或者可写为止。而对于非阻塞状态，如果没有东西可读，或者不可写，读写函数马上返回，而不会一直等待。缺省情况下，文件描述符处于阻塞状态。

实现阻塞型的 I/O，我们通常需要用到休眠和异步事件。在 Linux 系统下实现休眠首先要初始化一个的等待队列，该操作可以通过宏

```
DECLARE_WAIT_QUEUE_HEAD (name)
```

静态地进行，或者 API 调用动态地进行：

```
wait_queue_head_t my_wait_queue
```

```
init_wait_queue_head(&my_wait_queue)
```

当我们需要等待某个事件发生时，我们可以调用下列其中一个函数来使例程休眠：

```
wait_event(queue, condition)
```

```
wait_event_interruptible(queue, condition)
```

休眠的另一部分是唤醒，主要通过下面两个函数来完成：

```
wake_up(wait_queue_head_t *queue)
```

```
wake_up_interruptible(wait_queue_head_t *queue) 。
```

3.4 本章小结

本章重点阐述了蓝牙 HCI 协议的基础——底层协议栈，即物理驱动程序。首先介绍 Linux 的网络驱动程序的结构和层次，以及如何实现，包括硬件架构，驱动程序的基本概念和基本方法；在这个基础上，对蓝牙设备及其协议进行分析，讨论了蓝牙设备的工作方式和底层协议。接着介绍 USB 蓝牙设备的驱动程序的设计过程，所实现的驱动程序对并发访问给与了初步的支持，并对用户空间提供可阻塞得 I/O 操作，使得用户空间的程序可以通过标准的文件操作系统调用对设备进行访问。以此作为基础，我们可以在用户空间编写蓝牙协议栈的代码，一方面降低代码开发的难度，另一方面加快代码开发的速度。

这一部分工作是实验室往届人员的成果，但是本课题所要实现的 HCI 协议是建立在这个基础上的，包括驱动程序里面的数据结构和关键机制。所以介绍必要的背景材料是这一章的目的所在。

但是，本驱动程序只能够正确响应单设备的命令，需要继续编写高一层的协议，

才对蓝牙设备进行控制和连接操作，所以，接下来将讲述本课题的重点——如何实现 HCI 协议。

第4章 HCI 协议的主机控制模块及实现

4.1 主机控制模块（hci_host_control）的概述

在已经实现蓝牙 USB 驱动程序的基础上,我们可以由下图看出 HCI 与 USB 的数据传输情况:

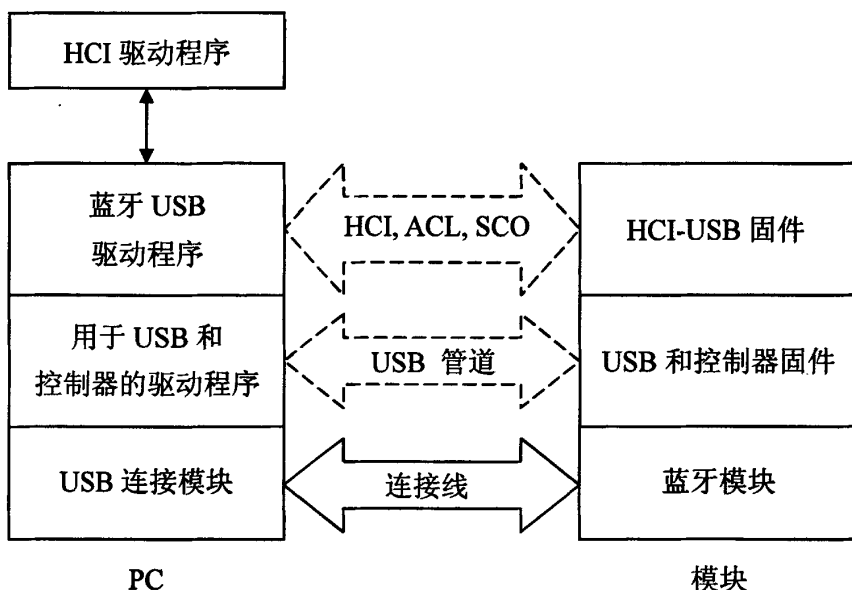


图 4-1 访问 HCI 和数据的 USB 接口

HCI 命令使用 USB 控制通道传输^[28-29], HCI 事件使用 USB 中断通信传输, 异步数据使用 USB 的 bulk 数据传输, 同步数据通过 USB 的等时通道传输, 这些都在 HCI USB 规范中定义。

在具体实现对蓝牙适配器模块控制前,必须先使蓝牙模块恢复到正常的工作状态,即初始化工作,而要做好初始化工作,首先要对 HCI 分组有进一步的理解。承接第二章的简单介绍,下面将详细分析 HCI 各个信息分组。

蓝牙规范定义了 HCI 命令格式,如图 4-2 所示。HCI 是通过包的方式来传送数据、命令和事件的,所有在主机和主机控制器之间的通信都是以包的形式进行。HCI 命令分组用于主机发送命令到主机控制器,而主机发出的大多数命令包都会出发主机控制器产生相应的事件包作为响应。

主机控制器在完成多数命令后发给主机一个命令完成事件，当然，其中一些指令在它们完成后并不接收指令完成事件，这种情况下主机控制器在收到命令开始执行时发给主机一个命令状态事件，当执行完后发给主机一个与这条命令相关的事件，此事件中的参数的有效性要依据状态参数中的错误原因来判断。如果命令的参数有错，或者当前状态不允许执行该命令，主机控制器就在返回的命令状态事件的状态参数中标明错误代码。如果状态参数后有连接句柄或蓝牙地址之类的参数也要返回，这样主机可以知道错误的命令对应于那一个实体。命令完成事件和命令状态事件都有一个 HCI 命令分组数，它指出了当前主机可以发往主机控制器的命令分组的数目，主机控制器可以将一个或几个命令缓冲，但是必须依照收到的顺序执行这些命令。

主机控制器可以缓存一个或多个 HCI 指令分组，主控制器应按照收到的顺序执行指令。但主机控制器在前一条命令完成之前可以开始执行新的命令，所以命令的完成的顺序并不完全等同于它们开始的顺序，相应的例子将会在后面的连接管理模块中大量出现。另外，主机控制器必须能够接收 HCI 指令分组和除 HCI 指令报头以为的 255 字节数据。

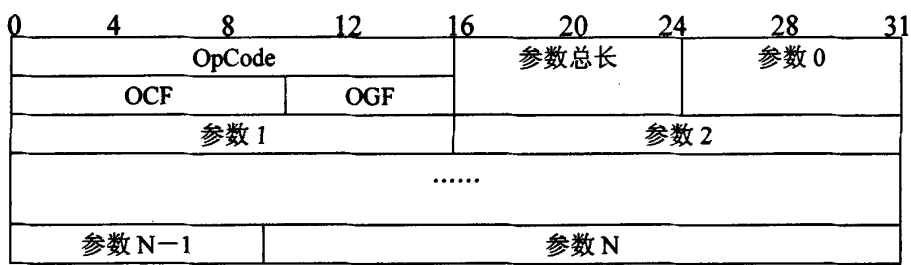


图 4-2 HCI 命令分组格式

每一条指令 (OpCode) 都指定了一个 2 字节的操作码，可以唯一标识指令类型。操作码本身又分为两部分，操作码段(OGF: OpCode Group Field)和操作码指令段(OCF: OpCode Command Field)。OGF 占用操作码的前 6 位，OCF 占用其余 10 位。这样的结构使得能够在不必对整个操作码完全解码的情况下即可获得附加信息。

表 4-1 HCI 指令分组各段定义

名称	值	参数描述
操作码	0xXXXX	OGF 占用 6 位: 0x00~0x3F. OCF 占用 10 位: 0x0000~0x03FF
参数总长	0xXX	所有数据分组中的参数总长以字节度量。
参数 0-N	0xXX	每一指令都有几个与其相关的参数, 这些参数及其长度由相应指令定义, 其长度一般为整数个字节

主机控制器利用 HCI 事件分组在事件发生时通知主机。HCI 事件分组格式如所示:

0	4	8	12	16	24	31
事件类型码			参数总长		事件参数 0	
事件参数 1					事件参数 2	
.....						
事件参数 N-1				事件参数 N		

图 4-3 HCI 命令分组格式

事件包的事件类型码用来区分不同的事件包, 参数总长与命令码中的意义相同, 表示所带参数的长度, 以字节数为单位, 随后就是所带的参数列表。

表 4-2 HCI 事件分组各段定义

名称	值	参数描述
操作码	0xXXXX	OGF 占用 6 位: 0x00~0x3F. OCF 占用 10 位: 0x0000~0x03FF
参数总长	0xXX	所有数据分组中的参数总长以字节度量。
参数 0-N	0xXX	每一指令都有几个与其相关的参数, 这些参数及其长度由相应指令定义, 其长度一般为整数个字节

4.2 初始化原语及 HCI 命令封装

4.2.1 指令和事件类型

总的来说, 命令包分为六种类型 [30]:

- 链路控制命令;

- 链路策略和模式命令；
- 主机控制和基带命令；
- 信息命令；
- 状态命令；
- 测试命令。

事件包也可分为三种类型：

- 通用事件，包括命令完成包(Command Complete)和命令状态包(Command Status)；
- 测试事件；
- 出错返回事件，如未知的 HCI 指令或者链接超时等。

蓝牙设备初始化是向主机控制器发送一系列指令分组，发送后，每一条指令分组都会返回至少一个事件分组，可以通过判断事件分组的状态位来确定指令的运行情况。下面一小节介绍关键的初始化指令，剖析其发送过程和相应返回事件。

4.2.2 HCI 协议中主要的初始化指令及其相应返回事件

- 主机控制器与基带指令提供识别和控制各种蓝牙硬件的能力。主机可利用这些指令修改本地设备的行为，下面先介绍 HCI 控制和基带指令，OGF 为 0x03。

(1) Reset

该指令复位蓝牙主机控制器、链路管理器和无线射频设备，而且放弃当前的操作选择，同时也放弃分组排队。在复位完成后，蓝牙设备进入待机模式。指令描述如表 4-3 所示。

表 4-3 HCI Reset 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI Reset	0x0003		Status (0x00 表示指令成功接收，将执行 ⁽³⁾)

(2) Set event filter

该指令用来指定不同的事件过滤器，指令描述如表 4-4 所示。对于同类事件过滤器或不同类事件过滤器，主机可多次发送各种链接申请。事件过滤器通过主机指定有关的对象，这些对象允许主机控制器只发送与主机有关的事件。仅由一部分事

³ Status 的参数值说明同样适用于下面的指令，接下来不再说明。另外，错误值说明将在下文补充说明。

件具有事件过滤器。默认（开机和复位后）方式无过滤器设置，而且自动识别标志关闭。每次从主机发送该命令时，都加入事件过滤器。

表 4-4 HCI Set event filter 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Set_Event_Filter	0x0005	Filter_Type, Filter_Condition_Type, condition	Status

(3) Write/Read scan enable

Scan_Enable 参数控制蓝牙设备是否周期性地扫描其他蓝牙设备的呼叫期望或查询申请。Write scan enable/Read scan enable 分别是写入和读出参数值。指令描述如表 4-5 所示，写入的命令参数和读取返回参数描述如表 4-6 所示。

表 4-5 HCI Write/Read scan enable 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_Scan_Enable	0x0019		Status; Scan_enable
HCI_Write_Scan_Enable	0x001A	Scan_Enable	Status

表 4-6 HCI Write/Read scan enable 事件分组参数描述

参数	值	参数说明
Status 1 字节	0x00	指令成功发送，主机控制器将执行
	0x01~0xFF	指令失败，具体错误请查看错误码表
Scan_Enable 1 字节	0x00	无扫描允许
	0x01	查询扫描允许；呼叫扫描禁止
	0x02	查询扫描禁止；呼叫扫描允许
	0x03	查询扫描允许；呼叫扫描允许

如果 Page_Scan 允许，则设备将基于相关时间间隔设置进入呼叫扫描模式；如果 Inquiry_Scan 允许，则设备将基于相关时间间隔设置进入查询扫描模式。

(4) Write/Read authentication enable

Authentication_Enable 参数控制是否由本地设备申请在链接设置（在创建链接命令或引入 ACL 链接的接收而且符合链接完成事件）下鉴权远程设备。在链接设置下，只有使用 Authentication_Enable 参数允许的设备可期望鉴权其他的设备。Write authentication enable/Read authentication enable 分别是写入和读取 Authentication_Enable 的参数值，指令描述如表 4-7 所示，参数描述如表 4-8 所示。

表 4-7 HCI Write/Read authentication enable 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_Authentication_Enable	0x001F		Status; Authentication_enable
HCI_Write_Authentication_Enable	0x0020	Authentication_Enable	Status

表 4-8 HCI Write/Read authentication enable 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送, 主机控制器将执行
	0x01~0xFF	指令失败, 具体错误请查看错误码表
Authentication_Enabled (1 字节)	0x00	鉴权禁止
	0x01	允许所有的链接鉴权
	0x02~0xFF	保留

(5) Write/Read connection accept timeout

Connection_Accept_Timeout 定义为从当主机控制器发出链接申请事件起到主机控制器自动拒绝引入链接止的时间间隔。在指定周期已出现且新的链接还没有识别的时候, 该参数允许蓝牙硬件自动地拒绝链接申请。Write connection accept timeout/Read connection accept timeout 分别是写入和读出这个参数值。指令描述如表 4-9 所示, 写入的命令参数和读取返回参数描述如表 4-10 所示。

表 4-9 HCI Write/Read connection accept timeout 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_Connection_Accept_Timeout	0x001F		Status; Connection_Accept_Timeout
HCI_Write_Connection_Accept_Timeout	0x0020	Connection_Accept_Timeout	Status

表 4-10 HCI Write/Read connection accept timeout 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送, 主机控制器将执行
	0x01~0xFF	指令失败, 具体错误请查看错误码表
Connection_Accept_Timeout (1 字节)	N=0xFFFF	在基带时隙的链接识别超时: 最大超时 = N * 0.625 ms N 的范围: 0x0001 ~ 0xb540 时间范围: 0.625 ms~29 s

(6) Write/Read page timeout

Page_Timeout 结构参数定义本地链路管理器等待基带呼叫响应的最大时间,

该基带呼叫响应来自本地初始化链接期望的远程设备。如果该时间终止，远程设备还没有在基带级上响应呼叫，本次链接则认为是失败。指令描述如表 4-11 所示，写入的命令参数和读取返回参数描述如表 4-12 所示。

表 4-11 HCI Write/Read page timeout 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_Page_Timeout	0x0017		Status; Page_Timeout
HCI_Write_Page_Timeout	0x0018	Page_Timeout	Status

表 4-12 HCI Write/Read page timeout 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送，主机控制器将执行
	0x01~0xFF	指令失败，具体错误请查看错误码表
Page_Timeout (1 字节)	N=0xFFFF	在基带时隙的链接识别超时： 最大超时 = $N * 0.625\text{ ms}$ N 的范围： 0x0001 ~ 0xb540 时间范围： 0.625 ms~29 s

(7) Host buffer size

该指令用于主机通知主机控制器有关控制器到主机 HCI ACL 和 SCO 数据分组发送的数据部分的最大长度。根据长度设置，主机控制器将分段传输到这些数据，所以 HCI 数据分组包含最大使用长度。该指令也通知主机控制器能够存放在主机数据缓冲区的 HCI ACL 和 SCO 数据分组的总数。其指令及其参数描述见下表。

表 4-13 HCI Host buffer size 事件分组参数描述

指令	OCF	命令参数	返回参数
HCI_host_Buffer_Size	0x0033	host_ACL_Data_Packet_Length, host_SCO_Data_Packet_Length, host_Total_Num_ACL_Data_Packets, host_Total_Num_SCO_Data_Packets,	Status

如果从主机控制器到主机的控制被关闭，而且 Host_Buffer_Size 命令还没通过主机发布，这意味着主机控制器可随意使用任何长度发送 HCI 数据分组到主机，同时可假设数据缓冲区是无限的。如果从主机控制器到主机的流控制是打开的，则 Host_Buffer_Size 命令必须要在电源打开或复位后通过主机在第一次 Host_Number_Of_Completed_Packed⁽⁴⁾ 命令发送前发送。

⁴ 该指令用于由主机指出主机控制器完成每次链接句柄的 HCI 数据分组数，指示主机里的相应缓冲区已经释放。

Host_ACL_Data_Packet_Length 参数用来确定包含在 ACL 数据分组内的 L2CAP 段的长度,该分组从主机控制器传送到主机。Host_SCO_Data_Packet_Length 参数用来确定 HCI SCO 数据分组的最大容量。主机和主机控制器双方都必须支持该指令和事件分组,在分组里的数据部分(含数据包头)长度是 255 字节。Host_Total_Num_ACL_Data_Packets 参数包含有可存储在主机数据缓冲区里的 HCI ACL 数据分组总数。主机控制器可确定在不同链接句柄间缓冲区如何划分。host_Total_Num_HCO_Data_Packets 参数给出 HCI SCO 数据分组的同样信息。

- 接下来是读取蓝牙设备信息参数的指令,这些信息参数是由硬件制造商确定的,主机不能修改这些参数的任何东西。其中 OGF 为 0x04。

(1) Read buffer size

该指令用来读出从主机到主机控制器发送 HCI ACL 和 SCO 数据分组的数据部分最大值。主机根据这些分组大小,分段从主机传输到主控制器,以便 HCI 数据分组包含这类大小的数据。

表 4-14 HCI Read buffer size 事件分组参数描述

指令	OCF	命令参数	返回参数
HCI_host_Buffer_Size	0x0005		host_ACL_Data_Packet_Length (2 字节) host_SCO_Data_Packet_Length (1 字节) host_Total_Num_ACL_Data_Packets (2 字节) host_Total_Num_SCO_Data_Packets (2 字节)
<div><div>3</div><div>ogf: 4, ocf: 5 .</div><div><div>hci_info_param</div><div><div><div>1. READ_LOCAL_VERSION</div><div>2. READ_LOCAL_FEATURES</div><div>3. READ_BUFFER_SIZE</div><div>4. READ_BD_ADDR</div><div>5. back</div></div><div>hci_info_param</div></div></div><div>read 13 bytes. received a Event packet. evt: e, plen: b .</div><div>event is: 1 5 10 0 c0 0 40 8 0 8 0 .</div></div>			
<div>说明: 0x04 表示 HCI 事件分组; evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: c, 即 0x04, 表示返回事件分组的参数长度为 12; 下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个; 第二和第三个参数 0x0510 为指令的 OpCode, 对应 Read_Buffer_Size 的操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。0xC000 表示 ACL_Data_Packet_Length, 实际值为 0x00C0, 表示允许的 ACL 分组的最大长度为 192 (十六进制为 C0); 0x40 表示 SCO_Data_Packet_Length, 表示允许的 SCO 分组的最大长度为 64; 后面连续两个 0x0800 表示主机控制器缓冲区允许接受的 ACL 分组和 SCO 分组数量均为 8。</div>			

其中这四个返回参数跟上文第（7）Host buffer size 的描述一样，这里不再复述。

(2) Read_local_Version_Information

该指令读出本地蓝牙设备版本信息值，其指令及其描述如表 4-15 和 4-16 所示。版本信息由 2 个参数组成：版本和修正参数。版本参数定义了蓝牙硬件的主要硬件版本。当蓝牙硬件新版本为新的蓝牙 SIG 说明生产时，只有版本参数改变。版本参数由 SIG 控制，修订参数由制造上控制，当需要时，可以修改。

表 4-15 HCI Read_local_Version_Information 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_local_Version_information	0x0001		Status; HCI Version; LMP Version; Manufacturer_Name; LMP Subversion

表 4-16 HCI Read_local_Version_Information 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送，主机控制器将执行
	0x01~0xFF	指令失败，具体错误请查看错误码表
HCI_Version (1 字节)	0xXX	当前蓝牙硬件的 HCI 版本
HCI_Revision (2 字节)	0xFFFF	当前蓝牙硬件的 HCI 的修订版本
LMP_Version (1 字节)	0xXX	当前蓝牙硬件的 LMP 版本
Manufacturer_Name (2 字节)	0xFFFF	蓝牙硬件制造商名
LMP_Subversion (2 字节)	0xFFFF	当前蓝牙硬件的 LMP 子版本

```
1
ogf: 4, ocf: 1 .
-----hci_info_param-----
*      1. READ_LOCAL_VERSION      2. READ_LOCAL_FEATURES      *
*      3. READ_BUFFER_SIZE        4. READ_BD_ADDR          *
*      5. back                     *
-----hci_info_param-----
read 14 bytes.  received a Event packet.  evt: e, plen: c .
event is:  1 1 10 0 1 60 4 1 a 0 60 4 .
```

说明：0x04 表示 HCI 事件分组；evt: e，即 0x0E 事件码，表示该返回事件为指令完成事件；plen: c，即 0x04，表示返回事件分组的参数长度为 12；下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个；第二和第三个参数 0x0110 为指令的 OpCode，对应 Read_local_Version_information 的操作码；0x00 表示指令返回的 Status，取值 0 表示执行正确。0x01 表示当前 HCI 版本为 1.0；0x6004 表示当前蓝牙硬件的 HCI 的修订版本；0x01 表示当前 LMP 版本为 1.0；0x0A00 表示蓝牙硬件制造商名；0x6004 表示当前蓝牙硬件 LMP 子版本。

(3) Read_local_Supported_Features

该指令为本地设备支持特征表。该指令返回 LMP 特征表，其指令及其描述如表 4-17 和 4-18 所示。

表 4-17 HCI Read_local_Supported_Features 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_local_Supported_Features	0x0003	Page_Timeout	Status; LMP_feature

表 4-18 HCI Read_local_Supported_Features 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送，主机控制器将执行
	0x01~0xFF	指令失败，具体错误请查看错误码表
LMP_Features (8 字节)	0XXXXXXXXX XXXXXXXX	LMP 特征的位屏蔽表。

```
2
ogf: 4, ocf: 3 .
-----hci_info_param-----
*      1. READ_LOCAL_VERSION      2. READ_LOCAL_FEATURES      *
*      3. READ_BUFFER_SIZE        4. READ_BD_ADDR              *
*      5. back                     *
-----hci_info_param-----
read 14 bytes.  received a Event packet.  evt: e, plen: c .
event is:  1 3 10 0 ff ff f 0 0 0 0 0 .
```

说明：0x04 表示 HCI 事件分组；evt: e，即 0x0E 事件码，表示该返回事件为指令完成事件；plen: c，即 0x04，表示返回事件分组的参数长度为 12；下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个；第二和第三个参数 0x0310 为指令的 OpCode，对应 Read_local_Supported_Features 的操作码；0x00 表示指令返回的 Status，取值 0 表示执行正确。0xFF FF 0F 00 00 00 00 00 表示 LMP 特征的位屏蔽表。

(4) Read_BD_ADDR

该指令读取 BD_ADDR 参数值。BD_ADDR 是 48 位蓝牙设备的唯一标识符。当该命令完成时，命令完成事件产生。其指令描述如表 4-19 和 4-20 所示。

表 4-19 Read_BD_ADDR 事件分组各段设定

指令	OCF	命令参数	返回参数
HCI_Read_BD_ADDR	0x0009		Status; BD_ADDR

表 4-20 Read_BD_ADDR 事件分组参数描述

参数	值	参数说明
Status (1 字节)	0x00	指令成功发送，主机控制器将执行
	0x01~0xFF	指令失败，具体错误请查看错误码表
BD_ADDR (6 字节)	0x XX XX XX XX XX XX	设备的硬件地址

4

```
ogf: 4, ocf: 9 .
-----hci_info_param-----
*      1. READ_LOCAL_VERSION      2. READ_LOCAL_FEATURES      *
*      3. READ_BUFFER_SIZE        4. READ_BD_ADDR              *
*      5. back                     *
-----hci_info_param-----
read 12 bytes.  received a Event packet.  evt: e, plen: a .
event is:  1 9 10 0 de 17 1 18 d 0 .
```

说明：0x04 表示 HCI 事件分组；evt: e，即 0x0E 事件码，表示该返回事件为指令完成事件；plen: c，即 0x04，表示返回事件分组的参数长度为 12；下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个；第二和第三个参数 0x0910 为指令的 OpCode，对应 Read_BD_ADDR 的操作码；0x00 表示指令返回的 Status，取值 0 表示执行正确。此后 0xDE 17 01 18 0D 00 为本地蓝牙设备的六字节蓝牙地址 BD_ADDR。

由于本课题的主要对象并没有涉及加密机制，所以这里不介绍加密的 HCI 指令设置。

4.3 程序流程与结果分析

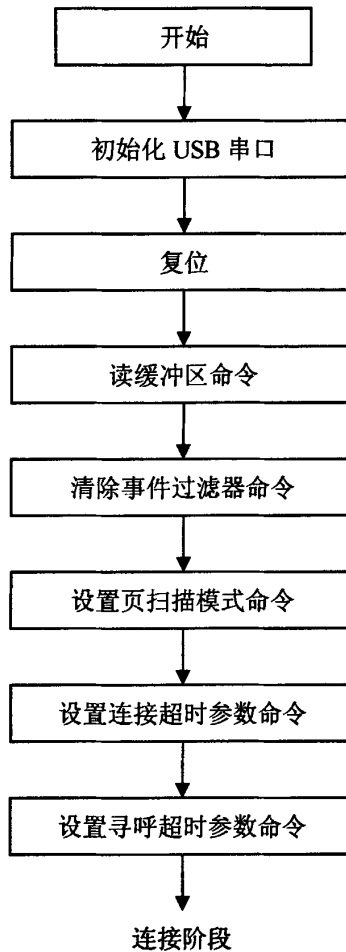


图 4-4 蓝牙设备初始化流程

设备初始化工作如上图所示，下面将介绍如何通过 HCI 协议代码实现这些功能。

如本章第二节所介绍的分组格式可以看出，每一条 HCI 指令都包含 OGF、OCF 和附带参数描述，即每一个 HCI 指令信息都由若干不同数据类型或者不同意义的数据所组成。在 C 语言里面，我们不能采用数组的形式把它们组织起来。为了保持 HCI 数据信息的逻辑关系，这里采用结构变量^[31-32]把这些数据信息有组织地存放在一起。所以负责传输数据信息的数据结构在这里定义如下：

```
struct _buf {  
    unsigned char buf[HCI_MAX_FRAME_SIZE];
```



```

int type;
__u8 from_dev;
size_t length;
};

```

说明如下：buf[] 数组是用来存放 OGF、OCF 和附带参数，type 存放 HCI 分组的类型（数据、指令和事件），from_dev 是指具体设备，length 是指本次信息所有有效数据的长度。

设置了传输载体，那如何进行传输呢？针对 HCI 而言，设置一个数据结构作为 HCI 设备的标准接口。其中记录 HCI 的属性和状态信息以及需要注册的标准服务函数指针，是一个数据与操作的集合体。由于它是 HCI 设备的标准接口，实现了 HCI 设备的抽象，所以它对所有的 HCI 传输层都是必须的。针对不同的 HCI 传输层（即针对不同的设备），把该数据结构嵌入到一个具体设备的对应数据结构中去。这样，在一个具体设备的数据结构中就不但包含 HCI，而且还包含了与具体设备相关的特殊属性与方法。在本课题中，这个具体设备就是位于 Linux 文件系统^[33-34]的/dev/hci_usb0（其中第一个设备为 hci_usb0，第二个设备为 hci_usb1，以此类推）。

所以，在用户空间对 hci_usb0 进行打开、读取、关闭等一系列文件操作，就等于对蓝牙设备进行相同操作。这里采用下面两个语句对设备进行写和读的操作：

```

write(file_num, _buf, sizeof(struct _buf));
read(file_num, _buf, 1028);

```

其中 file_num 为 hci_usb0，write() 函数里面是把 _buf 的所有数据写进 file_num，read() 函数是从 file_num 中读取 1028 字节个数，放进 _buf 结构里面。

根据不同指令的设定（具体指令功能见上一章），我们得到下面操作信息：

(1) Reset

指令分组：[0x01 03 0C 00]

事件分组：[0x04 0E 04 01 03 0C 00]

<pre> read 6 bytes. received a Event packet. evt: e, plen: 4 . event is: 1 3 c 0 . </pre>
--

图 4-5 Set_Event_Filter 的返回事件

指令分组说明：0x01 是 HCI 传输层的分组指示器（下同），表明为 HCI 指令分组；03 0C 00 为 HCI 指令分组的内容；因为 Reset 是主机控制指令，OGF 为 0x03

(6 位), OCF 为 0x003 (10 位), 根据十六进制的位运算⁵⁾, OGF 和 OCF 合起来的操作码是 030C。而按照蓝牙标准, 所有参数值的发送和接收都使用 Little Endian (小端格式), 即发送时先发送 0x03, 再发送 0x0C; 0x00 为指令参数长度, 由于该指令没有任何参数, 所以参数长度为 0。

事件分组说明: 0x04 是 HCI 传输层的分组指示器 (下同), 表明为 HCI 事件分组; 0x0E 为事件码, 表示该返回事件为指令完成事件; 0x04 是参数长度, 表示返回的参数长度为 4 字节; 0x01 为 Num_HCI_Command_Packtes, 表示当前允许发送到主机控制器的 HCI 命令的分组数为一个; 0x030C 为指令 OpCode, 指示了触发这一返回事件的指令为 030C (Reset); 0x00 表示指令的返回参数 Status, 表示指令执行正确。

(2) Set_Event_Filter

指令分组: [0x01 05 0C 01 00]

事件分组: [0x04 0E 04 01 05 0C 00]

```
read 6 bytes.  received a Event packet.  evt: e, plen: 4 .
event is:   1   5   c   0   .
```

图 4 - 6 Set_Event_Filter 的返回事件

指令分组说明: 0x01 表明为 HCI 指令分组; Set_Event_Filter 是主机控制指令, OGF 为 0x03 (6 位), OCF 为 0x005 (10 位), 合起来的操作码 0x0C05; 0x01 为指令所带参数的长度; 0x00 为指令参数 Filter_Type, 取 0 表示清除所有事件过滤器。

事件分组说明: 0x04 表示 HCI 事件分组; 见图 4 - 6, evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: 4, 即 0x04, 表示返回事件分组的参数长度为 4; 下一行 “event is” 的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个; 第二和第三个参数 0x050C 为指令的 OpCode, 对应 Set_Event_Filter 的操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。

(3) Write_Scan_Enable

指令分组: [0x01 03 0C 00]

事件分组: [0x04 0E 04 01 1A 0C 00]

⁵⁾ 合并运算具体的 C 语言实现代码: $((ocf \ \& \ 0x03ff) | (ogf \ \ll \ 10))$

```
read 6 bytes.  received a Event packet.  evt: e, plen: 4 .
event is:  1  1a  c  0  .
```

图 4-7 Write_Scan_Enable 的返回事件

指令分组说明：0x01 表明为 HCI 指令分组；Write_Scan_Enable 是主机控制指令，OGF 为 0x03（6 位），OCF 为 0x01A（10 位），合起来的操作码 0x0C1A；0x01 为指令所带参数的长度；0x03 为指令参数 Scan_Enable，取 3 表示寻呼扫描、查询扫描都允许。

事件分组说明：0x04 表示 HCI 事件分组；见图 4-7，evt: e，即 0x0E 事件码，表示该返回事件为指令完成事件；plen: 4，即 0x04，表示返回事件分组的参数长度为 4；下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个；第二和第三个参数 0x1A0C 为指令的 OpCode，对应 Write_Scan_Enable 的操作码；0x00 表示指令返回的 Status，取值 0 表示执行正确。

(4) Write_Connection_Accept_Timeout

指令分组：[0x01 16 0C 02 00 79]

事件分组：[0x04 0E 04 01 16 0C 00]

```
read 6 bytes.  received a Event packet.  evt: e, plen: 4 .
event is:  1  16  c  0  .
```

图 4-8 Write_Connection_Accept_Timeout 的返回事件

指令分组说明：0x01 表明为 HCI 指令分组；Write_Connection_Accept_Timeout 是主机控制指令，OGF 为 0x03（6 位），OCF 为 0x016（10 位），合起来的操作码 0x0C16；0x02 为指令所带参数的长度；0x0079 代表 Write_Connection_Accept_Timeout 指令中的参数 Connection_Accept_timeout 为 0x7900，该参数的十进制值乘以 0.625ms 得到的连接超时时间为 19 秒。

事件分组说明：0x04 表示 HCI 事件分组；见图 4-8，evt: e，即 0x0E 事件码，表示该返回事件为指令完成事件；plen: 4，即 0x04，表示返回事件分组的参数长度为 4；下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个；第二和第三个参数 0x160C 为指令的 OpCode，对应 Write_Connection_Accept_Timeout 的操作码；0x00 表示指令返回的 Status，取值 0 表示执行正确。

(5) Write_Page_Timeout

指令分组: [0x01 18 0C 02 00 80]

事件分组: [0x04 0E 04 01 18 0C 00]

```
read 6 bytes.  received a Event packet.  evt: e, plen: 4 .
event is:    1  18  c  0  .
```

图 4 - 9 Write_Page_Timeout 的返回事件

指令分组说明: 0x01 表明为 HCI 指令分组; Write_Page_Timeout 是主机控制指令, OGF 为 0x03 (6 位), OCF 为 0x018 (10 位), 合起来的操作码 0x180C; 0x02 为指令所带参数的长度; 0x0080 代表 Write_Page_Timeout 指令中的参数 Page_Timeout 为 0x8000, 该参数的十进制值乘以 0.625ms 得到连接超时时间为 20 秒。

事件分组说明: 0x04 表示 HCI 事件分组; 见图 4 - 9, evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: 4, 即 0x04, 表示返回事件分组的参数长度为 4; 下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个; 第二和第三个参数 0x180C 为指令的 OpCode, 对应 Write_Page_Timeout 的操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。

(6) 第 3-5 指令相对应的读取命令操作图示:

● Read_Scan_Enable

```
read 7 bytes.  received a Event packet.  evt: e, plen: 5 .
event is:    1  19  c  0  3  .
```

说明: evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: 5, 表示返回事件分组的参数长度为 5; 下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的分组数目为一个; 第二和第三个参数 0x190C 为指令的 OpCode, 对应 Read_Scan_Enable 操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。0x03 为指令参数 Scan_Enable, 取 3 表示寻呼扫描、查询扫描都允许。

● Read_Connection_Accept_Timeout

```
read 8 bytes.  received a Event packet.  evt: e, plen: 6 .
event is:    1  15  c  0  0  79  .
```

说明：evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: 5, 表示返回事件分组的参数长度为 5; 下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个; 第二和第三个参数 0x150C 为指令的 OpCode, 对应 Read_Connection_Accept_Timeout 的操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。0x7900 为 Connection_Accept_Timeout 参数, 与上文所设置的相符合。

● Read_Page_Timeout

```
read 8 bytes.  received a Event packet.  evt: e, plen: 6 .
event is:    1  17  c  0  0  80  .
```

说明：evt: e, 即 0x0E 事件码, 表示该返回事件为指令完成事件; plen: 5, 表示返回事件分组的参数长度为 5; 下一行“event is”的第一个参数 1 表示当前可以从主机发往主机控制器的指令分组数目为一个; 第二和第三个参数 0x170C 为指令的 OpCode, 对应 Read_Page_Timeout 的操作码; 0x00 表示指令返回的 Status, 取值 0 表示执行正确。0x8000 为 Page_Timeout 参数, 与上文所设置的相符合。

(8) Hci_info_param (读取蓝牙设备的预设信息, 见本章第二节的表格和图示) :

Read_Local_Version; Read_Locat_Features; Read_Buffer_Size; Read_BD_ADDR.

4.4 本章小结

本章重点阐述了蓝牙设备的初始化—主机控制模块, 针对蓝牙设备的物理信道属性、分组方式、配置流程等规范, 并给出解决方案。实现蓝牙设备的初始化。

而相关的截图中的数据都是蓝牙设备初始化流程里面的关键分组数据, 这些数据都是基于本人在进行蓝牙点对点实验所得出的, 只有这些指令能正确执行, 才能进行接下来的蓝牙点对点连接, 当然这个连接是基于 USB 传输层的, 基于其它传输层^[35]的会稍有不同。其中指令分组的一些参数也可以取其它的值, 返回事件分组相应有所不同。

对蓝牙设备进行参数配置后, 包括过滤器重置、呼叫与查询、最大相应时间、读取本地蓝牙设备地址 BD_ADDR 等, 接着就进入连接管理部分^[36], 在下一章进行详细讨论。

第5章 HCI 协议的连接控制模块及实现

5.1 主机链路模块 (hci_link_control) 概述

根据第二章总体设计方案, 连接控制模块主要完成蓝牙链路的管理和数据的打包和拆包等功能。连接控制模块^[37]允许主机控制器控制与其它设备的连接。其中包括以下指令: 链路管理器(LM)创建和修改与蓝牙远程设备的链路层的连接, 查询有效范围内的其它蓝牙设备, 以及执行其他的链路管理器协议(LMP)中的指令^[38]。

承接上一章的图 4 - 4 设备初始化流程图的连接阶段, 下图为连接的具体流程。

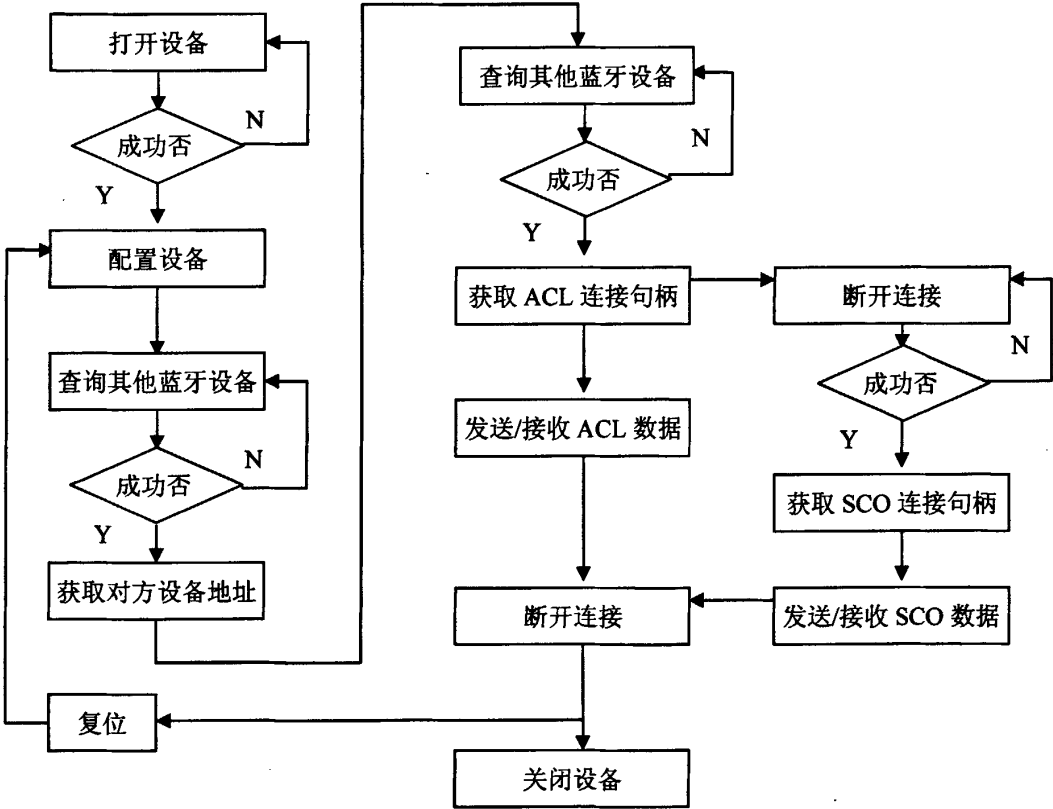


图 5-1 蓝牙设备连接流程

简单来讲, 在两个蓝牙设备传送数据之前, 先要建立逻辑链接。首先, 两个节点的主机各自初始化, 使各自模块进入正常工作状态; 然后, 一端模块被设置成呼叫查询模式, 即随时侦听其他蓝牙设备建立连接的请求; 同时, 另一端模块根据主

机的命令进行查询，进而请求建立连接。通过这一双方主机、模块间的协调工作，才能建立起最初的连接。

两个蓝牙设备之间最多只能有一条 ACL 链路^[39]。一个主控设备与其从属设备间最多只能有七条活动的 ACL 链路（每个活动的从属设备有一条）。ACL 是深入的基础，没有 ACL 物理链路的建立，就无法进一步建立高层的逻辑信道^[40]，发展高层协议就成无源之水。

所以下面两小节将介绍两个蓝牙设备进行最简单的点对点 ACL 数据通信的流程：查询、建立链路、进行数据通信和断开链路。

5.2 链路控制原语及 HCI 命令封装

对于链路控制指令，OGF 设为 0x01。

5.2.1 查询 (Inquiry)

主设备在与其它蓝牙设备创建连接之前使用查询指令来搜索邻近是否存在其它蓝牙设备，指令描述如表 5-1 所示，写入的命令参数和读取返回参数描述如表 5-2 所示。其中，而不同的接入码类型有不同的运行模式，LAP 表示相应的接入码类型的低地址部分；Inquiry_Length 参数指定查询模式的持续时间，超出该时间则终止查询；Num_Response 参数指定查询终止前能收到的应答数量。

如果查询到其他蓝牙设备，就获取对方的设备地址以及远端设备与主设备的时钟偏差，有了设备地址和时钟偏差，就可以为下一步快速创建 ACL 链接和时钟同步准备了必要的数

表 5-1 HCI Inquiry 指令分组各段设定

指令	OCF	命令参数	返回参数
HCI_Inquiry	0x0001	LAP, Inquiry_length, Num_Response	

表 5-2 HCI Inquiry 指令分组参数描述

参数	值	参数说明
LAP (3 字节)	0x9E8B00~0x9E8B3F	代表开始接入进程时的接入识别码。
Inquiry_Length (1 字节)	N=0xXX	查询最长持续时间 时间=N*1.28 s；范围：1.28 秒~61.44 秒
Num_Response (1 字节)	0x00	缺省值，不限制应答次数
	0xXX	从查询开始的最大应答次数， 范围：0x01~0xFF

注意：如果远端节点的主机没有运行 `Write_Scan_Enable` 指令，设置成查询扫描允许，则主节点的主机不会收到查询结果事件，也就是查询不到目标设备。

5.2.2 链路建立(Create_Connection)

本指令将使链路管理器创建与指令参数 `BD_ADDR`（程序里面用全局变量保存，见下文）指定的蓝牙设备的相互间链接，将使本地蓝牙设备开始呼叫进程以创建链路层链接，链路管理器将确定新的 ACL 链接如何建立。指令如表 5-3 所示。`Packete_Type` 指令参数指定链路管理器为 ACL 链接使用何种分组类型。`Page_scan_Repetition_Mode` 和 `Page_Scan_Mode` 参数指定 `BD_ADDR` 代表设备的呼叫扫描模式，`Clock_offset` 参数是本地时钟与 `BD_ADDR` 所代表的远端设备时钟之间的偏差。`Allow_Role_Switch` 参数说明当远端设备在链接初始化过程中请求主从角色切换^[41]时，本地设备是接受还是拒绝该请求。

表 5-3 HCI Create_Connection 分组各段设定

指令	OCF	命令参数	返回参数
Create_Connection	0x0005	BD_ADDR, Packete_Type, Page_scan_Repetition_Mode, Page_Scan_Mode, Clock_offset, Allow_Role_Switch	

注意：如果远端节点的主机没有运行 `Write_Scan_Enable` 指令，设置成查询扫描允许，则即时主节点发送创建连接指令，目标节点也不会收到连接请求分组，主节点和目标节点都会收到一个连接错误事件。

5.2.3 接受连接请求(Accept_Connection_Request)

如果收到连接请求，目标节点也同意建立连接，则可以使用该指令来接受请求。该指令使链路管理器确定如何建立新的连接。它通过 `Role` 参数来决定是否让链路管理器执行主从转换以及是否成为连接的主设备。

当链路管理器确认链路已建立时，连接的两蓝牙设备的主机控制器将向各自主机返回连接完成事件。如果指令执行成功，则连接完成事件将包括连接句柄。

表 5-4 HCI Accept_Connection_Request 分组各段设定

指令	OCF	命令参数	返回参数
Accept_Connection_Request	0x0009	BD_ADDR, Role	

5.2.4 数据传输

由于 HCI 协议主要的功能是把主机的信息封装成统一的规格, HCI 主要负责主机控制指令和连接管理指令,而编辑或者读取 ACL 或者 SCO 数据不是 HCI 协议的任务,换言之, HCI 只负责把更高层传输过来的数据传给蓝牙设备的主机控制器,而这些数据主要是 ACL 和 SCO 数据。这里简单演示 ACL 数据的传输有两个原因,第一是为了验证本课题所实现的 HCI 协议能否正常工作;第二是为了接下来的蓝牙协议栈的继续开发设计作一个启发。

ACL 数据分组格式为:

表 5-5 HCI Accept_Connection_Request 分组各段设定

0	4	8	12	16	24	31
Connect_Handle			PB Flag	BC Flag	Data total length	
Data						

实例分析见下一小节。

5.2.5 拆除链路 (Disconnect)

主设备和从设备都可以通过该指令终止现有链接,当主机控制器接收到 Disconnect 指令时,它将向主机返回指令状态事件分组和断开连接完成事件两个分组。

表 5-6 HCI Accept_Connection_Request 分组各段设定

指令	OCF	命令参数	返回参数
Accept_Connection_Request	0x0009	Connection_Handle, Reason	

该错误原因码 (Reason) 用于指示链接断开原因。当主单元发出拆除链接指令时,将把一个错误原因码作为参数使用。其中 0x05 表示验证或匹配失败; 0x13 表示其他终端的用户主动拆除链路; 0x14 表示其他终端因资源限制而拆除链路; 0x15 表示其他终端的设备关机而拆除链路。

5.3 程序实现与结果分析

程序流程见本章第一小节图 5 - 1，下面介绍程序代码的几个关键步骤。

1. 定义一个全局结构变量保存远端蓝牙的设备地址，因为如果当成功查询到设备，会返回对方的 48 位设备地址，所以这里采用包含一个数组成员的 bdaddr_t 结构变量：

```
struct bdaddr_t {
    __u8 b[6];
} __attribute__((packed)) ;
```

2. 定义一个全局变量保存连接句柄，链路连接完成的事件会返回一个连接句柄，以便主机能够判定本次指令完成事件属于哪个设备实例，以后发送的 HCI 数据包都用这个连接句柄标识远端设备。
3. 通过下面结构体来传递查询指令的参数（具体成员的含义与表 5-2 的介绍一一对应）：

```
struct hci_cp_inquiry {
    __u8    lap[3];
    __u8    length;
    __u8    num_rsp;
} __attribute__((packed));
```

4. 通过下面结构体来传递链路建立指令的参数（具体成员的含义与表 5-3 的介绍一一对应）：

```
struct hci_cp_create_conn {
    bdaddr_t bdaddr;
    __le16    pkt_type;
    __u8      pscan_rep_mode;
    __u8      pscan_mode;
    __le16    clock_offset;
    __u8      role_switch;
} __attribute__((packed));
```

5. 通过下面结构体来传输链路断开指令的参数（具体成员的含义与表 5-5 的介绍一一对应）：

```
struct hci_cp_disconnect {
    __le16    handle;
    __u8      reason;
} __attribute__((packed));
```

下面简单介绍这些链路控制指令的运行结果：

（1）查询 (Inquiry)

指令分组：[0x01 01 04 05 33 8B 9E 20 00]

指令状态事件分组：[0x04 0F 04 00 01 01 04]

查询结果事件分组：[0x04 02 0F 01 65 1C 01 18 0D 00 01 02 00 04 02 5A 1A 36]

查询完成事件分组：[0x04 01 01 00]

```
read 6 bytes.  received a Event packet.  evt: f, plen: 4 .
event is:  0  1  1  4  .

read 17 bytes.  received a Event packet.  evt: 2, plen: f .
event is:  1 65 1c 1 18 d 0 1 0 0 0 0 0 d9 59 .
After INQ, bda: 65:1C:01:18:0D:00 .

read 3 bytes.  received a Event packet.  INQ end, device number: 1 .
event is:  0  .
```

图 5-2 查询指令的返回事件

指令分组说明：0x01 表明为 HCI 指令分组；Write_Scan_Enable 是主机控制指令，OGF 为 0x03（6 位），OCF 为 0x033（10 位），合起来的操作码 0x0104；0x05 为指令所带参数的长度；0x33 8B 9E 为查询接入码；接着的 0x20 为 Inquiry_Length 参数，标识查询所允许的时间，范围 0x01~0x30；0x00 为参数 Num_Response 参数的缺省值，表示不限制响应数。

指令状态事件分组说明：0x04 表示 HCI 事件分组；见图 5-2，evt: f，即 0x0F 事件码，表示该返回事件为指令状态事件；plen: 4，即 0x04，表示返回事件分组的参数长度为 4；下一行“event is”的第一个参数 0 表示当前指令接收成功；下一个参数为 1，表示当前可以从主机发往主机控制器的指令分组数目为一个；接下来两个参数 0x0104 为指令的 OpCode，对应指令 Inquiry 的操作码。

查询结果事件分组说明：0x04 表示 HCI 事件分组；0x02 为事件码，表示该返回事件为查询结果事件分组；0x0F 代表返回 HCI 事件分组的长度；0x01 为参数 Num_Response，表示有一个设备应答；此后 0x 65 1C 01 18 0D 00 为查询到的蓝牙设备的六字节蓝牙地址 BD_ADDR；此后分别是 Page_scan_Repetition[0x01]，Page_Scan_Priod_Mode[0x02]，Page_Scan_Mode[0x00]，Class_of_Device[0x04 02 5A] 和两字节的 Clock_offset 参数[0xD9 59]。

查询完成事件分组说明：device number 为 1 表示查询到一个设备。

(2) 链路建立

指令分组：[0x01 05 04 0D 65 1C 01 18 0D 00 08 00 00 00 00 01]

指令状态事件分组：[0x04 0F 04 00 01 05 04]

与此同时，远端设备也会收到连接请求，如果答应请求，则收到：

连接完成事件分组：[0x04 03 0B 00 29 00 65 1C 01 18 0D 00 01 00]

```
read 6 bytes.  received a Event packet.  evt: f, plen: 4 .
event is:  0  1  5  4  .

read 13 bytes.  received a Event packet.  evt: 3, plen: b .
event is:  0 29 0 65 1c 1 18 d 0 1 0  .
conn_handle 29 .
connected, bda:65:1C:01:18:0D:00 .

read 9 bytes.  received a Event packet.  evt: 20, plen: 7 .
event is:  65 1c 1 18 d 0 1  .

read 5 bytes.  received a Event packet.  evt: 1b, plen: 3 .
event is:  29 0 5  .
```

图 5-3 链路建立的返回事件

指令分组说明：0x01 表明为 HCI 指令分组；0x0504 为创建连接的操作码；0x0D 为指令所带参数的长度；后面六个参数 0x 65 1C 01 18 0D 00 为要与之建立连接的蓝牙设备的六位地址；0x08 00 表示传送的分组类型为 ACL DM1 包；0x00 表示寻呼查询重复模式；接着的 0x00 表示寻呼查询模式为强制性的；再接下来的 0x00 00 表示无时钟偏移；最后的 0x01 表示在连接建立时不接受由目标设备请求的主从呼唤。

指令状态事件分组说明：0x04 表示 HCI 事件分组；见图 5-3，evt: f，即

0x0F 事件码，表示该返回事件为指令状态事件；plen: 4，即 0x04，表示返回事件分组的参数长度为 4；下一行“event is”的第一个参数 0 表示当前指令接收成功；下一个参数为 1，表示当前可以从主机发往主机控制器的指令分组数目为一个；接下来两个参数 0x0105 为指令的 OpCode，对应指令 Create_Connection 的操作码。

连接完成事件分组说明：0x04 表示 HCI 事件分组；0x03 为事件码，表示该返回事件为查询结果事件分组；0x0B 代表返回 HCI 事件分组的长度；0x00 表示命令成功，即已成功建立连接；0x2900 为连接句柄，此后 0x 65 1C 01 18 0D 00 为目标节点的蓝牙设备的六字节蓝牙地址 BD_ADDR；0x01 表示连接类型为 ACL 连接；最后一个字节表示加密模式，取值 0x00 表示不加密。

图中后面两个事件 0x20 和 0x1B，表示两设备设置的交互过程，经过这若干步的协商设置之后，双方就可正常通讯。

（3）接受连接请求

首先接收到远端设备的连接请求，相应的事件为：

连接请求事件：[0x04 0A 65 1C 01 18 0D 00 00 00 01]

接着，本设备开始应答请求，发送的指令和接收到的事件如下：

指令分组：[0x01 09 04 07 65 1C 01 18 0D 00 01]

指令状态事件分组：[0x04 0F 04 00 01 09 04]

（与此同时，远端设备也会收到应答连接请求的信息。）

连接完成事件分组：[0x04 03 0B 00 29 00 65 1C 01 18 0D 00 01 00]

```

read 12 bytes.  received a Event packet.  evt: 4, plen: a .
event is:  65 1c 1 18 d 0 0 0 0 1 .
bda:65:1C:01:18:0D:00 sent request .

ACCEPT_CONN_REQ

accept request, bda:65:1C:01:18:0D:00
ogf: 1, ocf: 9 .
read 6 bytes.  received a Event packet.  evt: f, plen: 4 .
event is:  0 1 9 4 .

read 13 bytes.  received a Event packet.  evt: 3, plen: b .
event is:  0 29 0 65 1c 1 18 d 0 1 0 .
conn_handle 29 .
connected, bda:65:1C:01:18:0D:00 .

```

图 5-4 链路建立的返回事件

连接请求事件说明：见图 5-4 中的第一行，evt: 4 表示该事件为连接请求事件，因为该事件的 OCF 码为 0x04；长度为 a，十字节的参数长度。下一行“event is”分别表示六字节的 BD_ADDR，三字节的 Class_of_Device 和一字节的 Link_Type。下一行则显示了发送连接请求的设备地址。

指令分组说明：0x01 表明为 HCI 指令分组；0x0904 为创建连接的操作码；0x0a 为指令所带参数的长度；后面六个参数 0x65 1C 01 18 0D 00 为要与之建立连接的蓝牙设备的六位地址；0x01 表示在连接建立时不接受由目标设备请求的主从呼唤。

指令状态事件分组说明：0x04 表示 HCI 事件分组；见图 5-4，evt: f，即 0x0F 事件码，表示该返回事件为指令状态事件；plen: 4，即 0x04，表示返回事件分组的参数长度为 4；下一行“event is”的第一个参数 0 表示当前指令接收成功；下一个参数为 1，表示当前可以从主机发往主机控制器的指令分组数目为一个；接下来两个参数 0x0904 为指令的 OpCode，对应指令 Accept_Connection_Request 的操作码。

连接完成事件分组说明：0x04 表示 HCI 事件分组；0x03 为事件码，表示该返回事件为查询结果事件分组；0x0B 代表返回 HCI 事件分组的长度；0x00 表示命令成功，即已成功建立连接；0x2900 为连接句柄，此后 0x65 1C 01 18 0D 00 为目标节点的蓝牙设备的六字节蓝牙地址 BD_ADDR；0x01 表示连接类型为 ACL 连接；最后一个字节表示加密模式，取值 0x00 表示不加密。

(4) 读取对方设备信息

经过了上面几个步骤之后，两设备已经建立连接，可以进行数据的传输和互相查询对方的信息的操作，下面简单介绍读取对方设备信息的情况：

- 读取远端设备的支持特性：Read_Remote_Supported_Feature 指令，通过把连接句柄作为参数，即可得到对方设备的信息，相应的返回事件如下图：

```
read 13 bytes.  received a Event packet.  evt: b, plen: b .
event is:   0 29 0 ff ff f 0 0 0 0 0 .
```

event is: 0x00 表示指令成功，2900 表示连接句柄，后面 8 个字节表示 LMP_feature (LMP 屏蔽位列表)。可对照上一章表 4-18，跟初始化时的数据相吻合。

- 读取远端设备版本信息：Read_Remote_Version_Information 指令，同样只需把连接句柄作为参数，即可得到对方设备的信息，相应的返回事件如下图：

```
read 10 bytes.  received a Event packet.  evt: c, plen: 8 .
event is:   0 29 0 1 a 0 60 4 .
```

event is: 0x00 表示指令成功，2900 表示连接句柄，0x01 表示 LMP_Version，0x0A10 表示制造商名字，0x6004 表示 LMP_Subversion。可对照上一章表 4-16，跟初始化时的数据相吻合。

(5) 数据传输

ACL 数据分组格式：[0x02 29 01 02 00 12 34]

说明：0x02 表示 ACL 数据包；在上面的连接中生成的连接句柄为 0x2900（其中 12 位有意义），其中 PB 和 BC 标志位是用于对高一层的接口，这里设为 PB=01，BC=00，则按照 ACL 数据包格式，连接句柄和两个标志位合并起来表示成^[6]：0x0129。0x0002 表示数据长度，占用两个字节；0x1234 为数据内容，这里为了简单测试 HCI 协议的正确性，先预先设好要传输数据。因为根据小端格式的收发规定，复字节参数在发送时要进行字节反序，即按照先后顺序发送 0x29 0x01 和 0x02 0x00。

发送 ACL 事件分组说明：见下图，其中 evt:13，表示本次事件是 Number of Completed Packets event，指示自从前一个 ACL 数据发送后，本主机对相应连接句柄又发送了多少个 HCI 数据分组。其中返回参数长度为 5 个字节。下一句 event is，

⁶ 合并运算具体的 C 语言实现代码：hci_handle_pack(h, f)(__u16)((h & 0x0fff)|(f << 12))

则表示本次对多少个连接句柄发送 ACL 数据，为一个链路；0x2900 表示该连接句柄，0x0100，表示已完成一组 HCI 数据的发送。

```
7. SEND ACL
read 7 bytes.  received a Event packet.  evt: 13, plen: 5 .
event is:   1  29  0  1  0  .
```

接收 ACL 数据的分组说明：见下图，其中系统提示接收到 6 个字节，本次数据包是 ACL 数据包，除去相应的包头标志位，实际 ACL 数据长度为 2，分别是 0x3412，与上文所发送的数据向吻合，验证了 HCI 协议的有效性。

```
read 6 bytes.  received a ACL packet.   ACL data length: 2.
ACL data:  34  12
```

(6) 拆除链路

指令分组：[0x01 06 04 03 29 00 13]

指令状态事件分组说明：[0x04 0F 04 00 01 06 04]

拆除链路事件分组说明：[0x04 05 04 29 00 00 16]

指令分组说明：0x01 表示 HCI 指令分组；0x0604 为拆除链路的指令操作码；0x03 为参数长度，0x2900 为连接句柄；0x13 为断开连接原因码。

指令状态事件分组说明：0x04 表示 HCI 事件分组；0x0F 表示该事件为指令状态事件；0x04 表示返回事件分组长度；0x00 表示命令正在执行；0x01 表示当前可以发送指令数目为一个；0x0604 为拆除链路的指令操作码。

拆除链路完成事件分组：0x04 表示 HCI 事件分组；0x05 为事件码，表示本事件为拆除链路完成事件；0x04 为参数长度；0x00 为状态位，表示拆除链路成功；0x2900 表示连接句柄；0x16 为断开原因（本地主机终止连接参数，专门用于主机控制器发送拆除链路指令后，返回的指令完成事件中的参数，区别于其他终端终止连接的原因码，这里为 0x13）。

5.4 本章小结

本章剖析了蓝牙 HCI 协议中的连接控制模块，与上一章结构类似，本章主要包括蓝牙逻辑链路控制规范的概述、链路管理器协议、建立连接的操作以及相关的原语，并结合实例进行分析说明，最后探讨了模块运行结果，即对蓝牙链路管理器

进行控制的实现，使得两 PC 机可通过蓝牙适配器建立连接。

通过 ACL 链路的建立，就可以在次基础上建立 SCO 链路，向上开发 L2CAP 等高一层的协议。

第6章 实际操作过程与改进

6.1 设备连接

本课题在相关实现时分为硬件和软件两个部分。在硬件，也就是设备的选用方面，因为兼容性和扩展性是本课题的首要考虑因素，所以使用的是电脑市场上非常普遍的蓝牙适配器。其制造商是英国的 Cambridge Silicon Radio (CSR) 公司，该公司是全世界第一家推出真正意义的单片蓝牙解决方案的公司，目前在全球蓝牙晶片市场占有率最高，比如：诺基亚手机中使用的蓝牙耳机都为 CSR 公司生产，苹果公司推出的手机 iPhone 的蓝牙模块也是 CSR 公司的产品。可以说现在所有的嵌入式手持设备中使用最多的蓝牙芯片就是 CSR 公司的系列芯片。该蓝牙适配器的功能跟其他蓝牙适配器的功能大体相同，这里不作介绍。软件方面，采用 LINUX 系统 Ubuntu 的 6.10 版本，内核版本为 2.6.20.4。

因为本实验是点对点的连接，所以需要两台电脑和两个蓝牙设备，具体连接情况如下图：

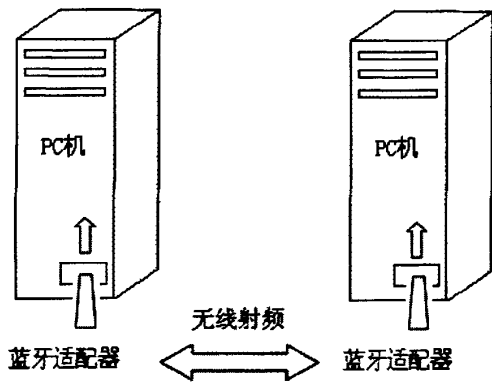


图 6-1 链路建立的返回事件

当设备准备好以后，就可以进行实验了：

- (1) 通过 Linux 系统的 `insmod` 命令加载蓝牙模块的 USB 驱动程序，以实现随后插进的蓝牙适配器的控制；把蓝牙适配器插进两台 PC 主机的 USB 断开，以实现系统里面的 `device` 文件夹出现相应的文件操作符，以代表蓝牙设备，随后对这个文件操作符运行 HCI 协议，以实现对该蓝牙设备的指令操作；

- (2) 两个 PC 端的蓝牙模块各自进行初始化, 进入正常工作状态;
- (3) 一端进行查询指令, 查询其他蓝牙设备, 这里把这一端定为主节点;
- (4) 另一端以其地址号响应主节点的查询指令, 这个响应动作由该模块的低层协议完成的, 不需要 PC 端的用户手工操作。这里把这一端定为从节点;
- (5) 主节点接收到查询请求的反馈信息, 根据返回的事件分组中的设备地址, 发出建立连接的请求;
- (6) 从节点接受主节点的连接请求, 两端成功建立链路;
- (7) 传输 ACL 数据, 查询对方设备信息。
- (8) 断开链路, 拨出蓝牙设备。

相应的图示说明可参见图 3-3, 图 4-1, 图 4-4, 图 5-1。

6.2 命令操作

6.2.1 命令界面

首先介绍本课题所开发的 HCI 协议的界面, 虽然 HCI 属于蓝牙协议栈中的底层协议, 但是它完成了指令和数据封装工作, 所以同时也属于面向用户的协议, 在操作界面上如何实现人和机器的高效互动是必须考虑的部分。在界面的设计上, 基本是基于以下四方面: 置界面于用户的控制之下; 导航功能, 以减少用户的记忆负担; 保持界面的一致性; 有清晰的针对性的信息提示。

所以从下面四幅图可看到的操作界面都是基于这些原则设计的:

```

*****
*      1. OPEN FILE          2. START READING          *
*      3. HCI_INFO_PARAM    4. HCI_HOST_CONTROL        *
*      5. HCI_LINK_POLICY   6. HCI_LINK_CONTROL        *
*      7. STOP READING      8. CLOSE FILE              *
*      9. EXIT              *
*****

```

上图是 HCI 协议运行后的出现第一个界面, 即主界面, 从 1-9 的选项说明依次是打开设备、新建一个线程以读数据、切换到查看设备信息的菜单、切换到主机控

制指令的菜单、切换到链路策略指令的菜单、切换到链路控制指令的菜单、释放读的线程、关闭文件、退出 HCI 协议。

hci_info_param		
1. READ_LOCAL_VERSION	2. READ_LOCAL_FEATURES	
3. READ_BUFFER_SIZE	4. READ_BD_ADDR	
5. BACK		

上图是查看设备信息的界面，在主界面选择 3 即可进入该界面。从 1-5 的选项说明依次是读取本地蓝牙设备版本信息值、本地设备支持特征表、本地设备缓冲区大小、本地设备的设备地址、返回主界面。

OGF_HOST_CTL		
1. RESET	2. SET_EVENT_FLT_clear	
3. WRITE_CONN_ACCEPT_TIMEOUT	4. WRITE_PAGE_TIMEOUT	
5. WRITE_PAGE_SCAN_ACTIVITY	6. WRITE_SCAN_ENABLE	
7. SET_EVENT_FLT	8. WRITE_ENCRYPT_MODE	
9. WRITE_CLASS_OF_DEV	10. WRITE_AUTOMATIC_FLUSH_TIMEOUT	
11. WRITE_VOICE_SETTING	12. WRITE_SCO_FLOW_CTRL_EN	
13. SET_H_CTRL_2_HOST_FLOW_CTRL	14. HOST_BUFFER_SIZE	
15. READ	16. BACK	

上图是主机控制指令的界面，在主界面选择 4 即可进入该界面。从 1-16 的选项说明依次是重置本地设备、清除事件过滤器、设置连接超时、设置呼叫相应超时、设置呼叫扫描间隔和呼叫扫描区间、设置扫描允许、设置事件过滤器、设置加密模式、设置本地设备类参数、设置刷新超时（针对指定的连接句柄）、设置话音链路、设置 SCO 流控制允许、设置流控制（用于主机控制器直接打开或关闭主机控制器到主机的流控制）、设置主机控制器的 ACL 和 SCO 数据缓冲区的大小、返回主界面。

hci_link_control		
1. CREATE_CONN	2. ACCEPT_CONN_REQ	
3. DISCONNECT	4. INQUIRY	
5. SHOW CONNECT HANDLE	6. AUTHENTICATION_REQUESTED	
7. SEND ACL	8. READ_REMOTE_SUPP_FEATURES	
9. READ_REMOTE_VERSION_INFO.	10. READ_CLOCK_OFFSET	
11. BACK		

上图是链路控制指令的界面，在主界面选择 6 即可进入该界面。从 1-11 的选项说明依次是新建链路、接受连接请求、断开现有连接、查询、显示当前的连接句柄、建立身份鉴权（与制定连接句柄）、发送预设的 ACL 数据（测试用途）、获取对方设备的支持特征表、获取对方设备的版本信息值、读取对方设备的时隙信息、返回主界面。

6.2.2 多线程机制

HCI 协议具备接收和发送数据的功能，但是，如何在面向用户的情况下，实现数据信息在本地设备上的进出情况，并清晰有效地显示呢？这时，跟第三章 USB 驱动程序的情况类似，同样要解决并发的问题。不过这里并不是采取锁机制，而是采用多线程的机制^{[42][43]}。

接下来通过对开发过程中遇到的并发事件的分析，讨论如何通过多线程来处理。

蓝牙每个指令的运行时间都可以设置成各不相同，而每发送一个指令，蓝牙设备都会返回一个指令完成事件（有时还有指令状态事件），这时，必须通过读取设备文件符来读取这些事件。而多线程机制正是处理这种情况的理想选择。

按照标准定义，进程(process)是资源管理器的最小单位，线程(thread)是程序执行的最小单位。在不同的操作系统，两者定义会不同。但无论按照怎样的分法，一个进程至少需要一个线程作为它的指令执行体。一个进程可以拥有多个线程，此时，如果进程运行在多处处理器的机器上，它就可以同时使用多个 cpu 来执行各个线程。即时是在单 cpu 的机器上，采用多线程模型来设计程序有很多好处。

- 有些问题可以通过将其分解从而改善整个程序进程的吞吐量。在只有一个控制线程的情况下，单个进程需要完成多个任务时，需要把这些任务串行化；有了多个控制线程，相互独立的任务的处理就可以交叉进行，只需要为每个任务分配一个单独的线程。
- 交互的程序可以通过多线程实现响应时间的改善，把程序中处理用户输入输出的部分与其他部分分开。
- 通过为每种事件类型的处理工作分配单独的线程，能够简化处理异步事件的代码。每个线程在进行事件处理时可以采用同步编程模式，而同步编程模式比异步编程模式简单得多。

每次运行 HCI 协议对蓝牙设备进行操作，cpu 都新建一个进程，作为这个程序的运行实例。一般来讲，在向文件读写时，这个进程有可能在读写处阻塞，直到一

定的条件满足。比如我们从蓝牙设备读取数据，可能缓冲区里面没有数据可读，这个时候，读调用就会一直等待，直到有数据可读，严重影响了主机端对设备的指令或者数据包发送。

所以我们通过创建一个读的线程来进行对蓝牙设备获取到的远端信息的显示。

因此，多线程机制使得本设计更简洁、功能更完备，程序的执行效率也更高。

6.3 实验结果

两个 PC 主机各自连接上蓝牙设备之后，首先新建一个终端通过“cat /proc/kmsg”查看内核的运行信息^[44]，过程中会显示所有在内核空间里运行的程序。

通过 insmod 命令装载蓝牙模块的 USB 驱动程序，使得主机可支持插进来的蓝牙设备。命令成功运行后，在刚刚新建的终端会显示“有新的 USB 设备的”内核信息。同时在/dev 下面会自动生成一个文件符，hci_usb0，代表本地蓝牙设备。

运行本论文所实现的 HCI 协议，对 hci_usb0 进行 HCI 指令操作。当看到主界面后，可以进行打开设备、新建读线程、查看本地设备信息等操作，而每一个指令操作都会生成相应的内核信息。

当两个设备都初始化，可以正常工作后，可以进入设备的连接阶段。通过主界面的菜单选择进入 hci_link_control 子菜单。将任意一个设备定为主设备，首先发出查询指令，当有远端蓝牙设备响应之后，会返回相应事件，随之可以进行“建立连接”指令。作为从设备响应了主设备查询后，会一直等待主设备来建立连接，一旦收到连接请求，即可答应。这时双方即可成功建立链路。

建立连接后，双方都可以进行查询对方信息、发送 ACL 数据、断开等操作。在查看内核信息的终端上可以观察到数据的读写、缓冲区的操作等信息。

当设备需要从主机拔出时，首先必须停止读的线程，然后关闭文件符(hci_usb0)，退出 HCI 协议，这时即可拔出设备，同时会出现相应的内核信息。

实验证明，蓝牙设备的初始化、查找、连接、传输的过程准确无误，这些现象表明了本课题所实现的 HCI 协议能够正常工作。

6.4 本章小结

因为系统测试是系统质量保证的关键，本章重点描述了本论文所实现的蓝牙协

议在 Linux 系统的运作情况，记录了实际操作流程、遇到的困难和解决方案，其中这个解决方案是通过采用 Linux 系统上的较新颖的技术机制而得到的。最后针对实验结果进行分析。

本课题在两个 PC 主机上完成。在各自完成初始化工作后，将其中一个主机设为主设备，另一个设为从设备时，主设备能够查找并且连接到从设备，能够读取对方设备信息，也能够将数据发送到从设备。初始化、查找、连接、传输的过程准确无误，这些现象表明了本课题所实现的 HCI 协议能够正常工作，这也与撰写论文初期对本论文定下的实验预期结果相吻合。

第7章 结束语

7.1 成果及总结

本文主要从蓝牙协议栈的分层结构入手,先介绍了基于 Linux 下的底层驱动程序,接着具体介绍了 HCI 协议,包括服务原语、操作流程、协议代码的设计与调试等,然后通过实验验证,实现了预期目标。本课题取得的主要研究成果包括:

1. 分析目前 Linux 用户空间和内核空间数据交换的方式,来建立用户空间和内核空间数据交换的通道以进一步提高通信效率。熟悉 Linux 网络驱动程序的结构和层次,以及如何实现,包括硬件架构,驱动程序的基本概念和基本方法。
2. 开发出对蓝牙规范兼容的基于嵌入式系统的蓝牙协议栈,在上一届研究人员实现 Linux 系统内核空间下的 USB 蓝牙设备驱动程序的基础上,在用户空间实现 HCI 协议,通过该协议栈各模块提供的功能使两台蓝牙设备间能够达到无线通讯的目的。
3. 在对 HCI 层的较为透彻研究和功能实现的基础上,进一步与实验室下一届研究人员进行更高一层协议的研究和讨论,以使得协议栈开发工作得以无缝延续。

◇ 本课题研究的应用价值

1. 本课题将要实现的蓝牙通讯程序具有良好的扩展性与工程应用前景。可在此基础上建立 SCO 连接发送语音信息;从设备配置数据采集板卡后,即可建立基于蓝牙技术的无线数据采集系统,在无线网络化测控领域具有十分广阔的发展前景。
2. 就目前的发展情况来看,蓝牙技术在手机、耳机、PDA、数码相机和数码摄像机等设备上都有应用。由于手机的普及率较高,因此蓝牙手机成为目前蓝牙技术的主要应用,所以应用者可以通过本文将要实现的协议,建立起个人手机和电脑的个性化服务。
3. 由于本课题采用 Linux 系统,而 Linux 凭着灵活的可裁剪特性、高度的可移植性,以及极低的成本,使得 Linux 系统在嵌入式应用领域有着优秀的表现。所以通过嵌入式产品,蓝牙将会是如虎添翼地发挥无线短距优势。

4. 这种蓝牙系统模式是了解蓝牙、入门蓝牙的基础模式，具有重大的实验和预演意义，其所需投入很少，一般实验室都有能力承担开发成本。普通工程人员可以对本课题所开发的协议略作修改，以根据不同功能要求实现产品的蓝牙无线通讯化，对于普及蓝牙技术进行产品应用开发具有重要的指导作用和现实意义。

7.2 不足与展望

不足之处：由于时间关系，实现了 HCI 协议中大概 65% 的服务原语，虽然这些原语可以实现设备的基本功能，但是如果考虑到安全性^[45-46]或者流量控制^[47-48]，还必需把剩下的原语补充完整。

当今蓝牙规范的版本已到 2.0，而本论文所使用的是与蓝牙规范 v1.2 兼容的芯片，其特点与蓝牙规范 v2.0 有显著的区别，最大的区别是传输的速率。在当今突飞猛进的电子世界，电子设备间的传输速率的提高是一个重点的方向，而且更高的速率意味着更高的性能^[49]。

所以下一步的工作有以下大部分：

1. 继续完善蓝牙协议栈：由于 HCI 协议的庞大与复杂，尽管实现了整个协议的流程框架，但并没有考虑到某些细节。所以一方面要完善好本课题的 HCI 协议，另一方面继续根据蓝牙协议栈进行更高层协议的实现工作，将更多的蓝牙协议应用起来，以达到最优化整个系统，充分使用资源的目的。
2. 密切关注蓝牙技术发展动态，尽可能使用最新技术产品^[50]，针对本课题，首先可以把原来的与蓝牙规范 v1.2 兼容的芯片换成与蓝牙规范 v2.0 兼容的芯片，则可以大大提高传输速率，并且提高稳定性和可靠性。

参考文献

- [1] 单鹏, 唐宏, 龙薇, 赵全军. 划时代蓝牙通信的研究与探索[J]. 通信技术, 2007, 7(40): 53-55.
- [2] <http://www.bluetooth.com/>.
- [3] <http://www.bluetooth.org.cn/>.
- [4] 张禄林. 蓝牙协议及其实现[M]. 北京: 人民邮电出版社, 2001.
- [5] 双鱼. 沟通从“牙”开始 渐入佳境的蓝牙技术[J]. 现代计算机, 2007, 3: 44-45.
- [6] 刘继州. 蓝牙技术分析及应用前景预测[J]. 科技咨询导报, 2007, 3: 20-22.
- [7] 郑孟. Linux—网络时代的操作系统[J]. 计算机与信息技术, 2007, 13: 69-71.
- [8] 张铁强. Linux 操作系统的发展优势[J]. 辽宁教育行政学院学报, 2006, 11: 167-168.
- [9] 毛宇斌, 邹乐. 基于 Windows2000 蓝牙局域网接入系统的实现[J]. 电子工程师, 2001, 27(12), 33-35.
- [10] 郭锴任娜. 基于嵌入式 Linux 的网络设备驱动程序的开发[J]. 电子科技, 2006, 8: 25.
- [11] 通信世界网. <http://www.cww.net.cn/>.
- [12] Bluetooth Special Interest Group. BLUETOOTH SPECIFICATION Version 2.0 + EDR[S].
- [13] Dabid Kammer, Gordon McNutt, Brian Senese, Jennifer Bray 著, 李静, 奉继辉, 王婷, 杨莉 译. 蓝牙应用开发指南—近程互联解决方案[M]. 北京: 科学出版社, 2006.
- [14] W.Richard Stevens Stephen A.Rago 著, 尤晋元, 张亚英, 戚正伟 译. UNIX 环境高级编程 (第二版) [M]. 北京: 人民邮电出版社, 2006: 397-434.
- [15] 杨宇音, 李志淮. Linux 中用户空间与内核空间的通信实现[J]. 微机发展, 2005, 15(5): 75-76.
- [16] 金纯, 林金朝, 万宝红 著. 蓝牙协议及其源代码分析[M]. 北京: 国防工业出版社, 2006.
- [17] 内核之旅. <http://kernel.lupaworld.com/>.
- [18] 周明德. 保护方式下的 80386 及其编程[M]. 北京: 清华大学出版社, 1993.
- [19] Robert Love, Linux Kernel Development[M], Sams Publishing, 2003.

- [20] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman 著, 魏永明, 耿岳, 钟书毅 译. Linux 设备驱动程序 (第三版) [M]. 北京: 中国电力出版社, 2006.
- [21] W.Richard Stevens. Advanced Programming in the UNIX Environment[M]. Addison Wesley, 1992.
- [22] W.Richard Stevens. UNIX Network Programming[M]. Prentic Hall, 1998.
- [23] Maurice J. Bach. The Design of the UNIX Operating System[M]. Prentic Hall, 1990.
- [24] Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. Linux Device Driver[M]. O' Reilly Media, 2005.
- [25] <http://www.usb.org/developer> . USB Specification.
- [26] 王云飞. USB 系统研究. 清华大学工学硕士学位论文[D], 2001: 14-17.
- [27] Sun Microststems. 编程接口指南[P]. 2005.
- [28] 梁正平, 毋国庆, 肖敬. Linux 中 USB 设备驱动程序研究[J]. 计算机应用研究, 2004, 6: 70-72.
- [29] 杨伟, 刘强, 顾新. Linux 下 USB 设备驱动研究与开发[J]. 计算机工程, 2006, 32(19): 283-285.
- [30] 马建仓, 罗亚军, 赵玉婷. 蓝牙核心技术及应用[M]. 北京: 科学出版社, 2003.
- [31] 谭浩强. C 程序设计[M]. 北京: 清华大学出版社, 2000.
- [32] 夏宽理. C 语言与程序设计[M]. 上海: 复旦大学出版社, 2000.
- [33] 卫耀辉, 郑之光. Linux 系统下蓝牙设备驱动程序研究和实现[J]. 计算机应用研究, 2002, 7: 147-149.
- [34] 刘峥嵘, 张智超. 嵌入式 Linux 应用开发详解[M]. 北京: 机械工业出版社, 2004, 7.
- [35] 陆佳炜. 蓝牙主控制器接口简析[J]. 半导体技术, 2003, 28(3): 54-57.
- [36] 张元, 黄小莹. 基于蓝牙技术的无线数据传输系统的设计[J]. 杭州电子工业学院学报, 2004, 24(4): 78-81.
- [37] Bluetooth Special Interest Group. Personal Area Networking Profile[J]. <http://www.bluetooth.org/> . 2003.
- [38] Bluetooth Special Interest Group. Bluetooth Network Encapsulation Protocol(BNEP) Specification[J], Version1.0, 2003, <http://www.bluez.org/> .
- [39] 马斌, 罗汉文, 郭环球. 蓝牙散射网网间通信问题的研究[J]. 中国电子商情:

通信市场, 2005, 8: 38-40.

- [40] 宋明中, 候思祖, 马昕霞. 基于蓝牙技术的 PC 机间数据通信接口设计与实现[J]. 电力系统通信, 2003, 24(8): 18-21.
- [41] 周丽雅, 任志考, 郭忠文. 基于设备实际性能的蓝牙散列网构建算法[J]. 计算机工程, 2007, 33(19): 119-121.
- [42] 杨志文. 深入 Linux 建构与管理[M]. 北京: 人民邮电出版社, 2000.
- [43] 毛德操, 胡希明. Linux 内核源代码情景分析[M]. 杭州: 浙江大学出版社, 2001.
- [44] K.Wall, M.Watson, M.Whitis 著, 王勇, 王一川, 林花军, 甘泉 译. GNU/Linux 编程指南[M]. 2002.
- [45] 赫芳, 郑志蓉. 蓝牙安全技术研究[J]. 计算机与网络, 2007, 5: 37-38.
- [46] 韩慧华, 肖扬. 加密算法在蓝牙安全机制中的应用[J]. 北方交通大学学报, 2003, 27(2): 24-28.
- [47] 张晶, 李丰, 牛丽萍. 基于蓝牙 HCI 传输层流控模型的构建[J]. 海军工程大学学报, 2007, 19: 67-72.
- [48] 张晶, 李铁盘. 蓝牙 HCI 传输层流量控制的研究[J]. 计算机工程与设计, 2006, 27(23): 4500-4503.
- [49] 蓝牙中国网. <http://www.bluetoothchina.com/>.
- [50] Sun Microsystems. <http://docs.sun.com/>.

致 谢

首先，我深深地感谢黄晓老师！黄老师渊博的学识，严谨的治学态度，以身作则的工作行动都是我学习的榜样，无时无刻不在感染着我，激励我不断地进步。黄老师的言传身教，使我在这一段求学期间，理论知识水平和科研实践能力有了本质的提高，对我以后的工作和学习都受益非浅。在论文撰写过程中，黄老师严格要求，在理论推导和方案设计等方面提出了许多宝贵的意见和建议，使得论文顺利完成。

实验室就像一个温暖的家，系里的同学和师兄弟在我的学业和生活方面给了我很大的帮助，我衷心感谢潘练锋、余杰文、徐博、蔡锦周同学，无论在设计或者调试过程中，他们都给予了宝贵的建议和指导，从他们身上我学到了很多東西。

最后再次衷心感谢所有一切关心、鼓励、帮助和支持我的人，祝你们一生平安。

在这篇论文即将完成的时候，我对所有向本文及作者直接或间接提供过帮助的老师 and 朋友们表示衷心的感谢！

作者：[汤杰](#)
学位授予单位：[中山大学](#)

相似文献(10条)

1. 学位论文 [臧雨霖](#) [无线通信在智能交通系统中的应用与研究](#) 2007

改革开放以来,我国城市经济水平迅速提高,城市人口数量激增,城市道路建设不断改善。但是,城市道路建设速度远远落后于城市车辆增加的速度,城市交通管理效率较低,城市公交车车辆拥挤现象表现的尤为突出,如何运用现代通信技术对公交车进行监控已成为城市交通发展的重要研究课题。

本文对现有公交车辆管理系统中的几种通信方式进行对比,提出了利用移动通信的GPRS网络, GPS网络和互联网络等多种现代通信技术手段相结合的系统解决方案,实现了公交车辆运行的实时、动态监控管理。该监控系统具有集成化、信息化管理的特点,可以有效提高公交车辆的运行效率。

本文首先提出了城市智能公交监控系统的总体设计方案,然后以车载端硬件平台的构建和软件设计为主要内容,对GPS模块、GPRS模块,嵌入式系统平台的构建进行了详细的设计和模块的调试。对于定位模块的选择,设计选用了GARMIN公司的GPS25模块,该模块能够实现车辆预定站点经纬度数据的采集。数据处理模块选用广泛应用的SANGSUNG公司的ARM处理器芯片S3C2410,通过外围FLASH, SDRAM, 串行口和语音电路的扩展,从而满足系统经纬度数据的处理和存储, GPRS模块的控制和车辆到站广播的要求。软件平台选用嵌入式Linux系统,在此嵌入式系统上构建Qt/Embedded的图形库,实现公交车到站信息的显示,并方便司机在故障时实现手动模式的转换。

设计实现了公交车辆经纬度信息的采集和自动报站功能,并对该系统的功能的进一步的完善和发展前景进行了展望。从而为实现我国城市公交车辆的智能化管理开辟了一条探索之路。

2. 学位论文 [王爱矛](#) [基于ARM-Linux的嵌入式数据采集与远传系统](#) 2008

随着通信技术的发展,无线通信技术在工业领域的应用日益增多。以前,工业中大多采用有线或人工的方式进行数据采集与传输,虽然简单实用,却耗费了大量人力、物力资源,且很大程度上限制了应用场所的拓展。因此,选取一种相对经济、稳定而又高效的无线传输方式就变得紧迫和必要。

随着GPRS网络技术的逐渐成熟, GPRS无线网络逐渐显露出其在远距离通信应用中的优势。于此同时,嵌入式软硬件技术的飞速发展也使得嵌入式产品进入千家万户。因此,采用基于嵌入式系统和GPRS网络进行无线通信渐渐成为当今应用的热点之一。

本系统采用高性能嵌入式微处理器S3C2410和GPRS无线通讯模块MC39i构建硬件平台,以嵌入式Linux操作系统和TCP/IP协议建立软件平台,完成基于ARM-Linux的嵌入式数据采集与远传系统设计。

本文首先对嵌入式系统的概况进行了综述,接着对嵌入式处理器、嵌入式操作系统和GPRS无线网络技术进行了概要介绍,然后提出了基于ARM-Linux的嵌入式数据采集与远传系统的设计方案,并从硬件设计和软件实现两方面具体阐述了该系统的开发实现过程,包括搭建以S3C2410和MC39i为核心的硬件平台以及在该硬件平台上建立基于嵌入式Linux操作系统的软件平台,并最终实现了数据采集与远传功能。

此系统由于采用了高性能的ARM处理器和嵌入式Linux系统,因此在多任务并行处理和进程实时处理等方面具有一定的优势。该系统可以广泛应用于燃气、油田和电力等部门,具有较好的发展前景。

3. 学位论文 [蔡锦州](#) [基于Linux的蓝牙L2CAP及RFCOMM层协议的实现](#) 2009

在当今的信息化社会中,短距离无线通信的应用越来越广泛。蓝牙技术作为一种新型的无线数据和语音通信的开放性标准,具有保密性高、使用方便、功能强大、价格低廉,功耗低等优点,在日常生产和生活中得到了广泛的应用。另一方面, Linux操作系统作为开放源代码的代表,不但拥有卓越的功能和性能,而且日趋成熟,受到巨大的嵌入式设备市场的重视,许多嵌入式应用产品都采用Linux作为系统平台。

本课题以蓝牙规范为基础, Linux系统为平台,常见的蓝牙适配器作为实现工具,研究设计一个基于嵌入式Linux的蓝牙协议栈,重点对协议栈中的L2CAP层和RFCOMM层进行了分析并讨论了这两层协议的设计和实现方法。L2CAP协议位于基带协议之上,为高层提供面向连接和面向无连接的数据服务,完成协议复用、分组分段和重装、服务质量管理等功能。RFCOMM协议提供了对RS-232串口的仿真,使得传统的基于串口的应用无需改变即可使用蓝牙技术。

论文首先分析L2CAP层和低层HCI协议层之间的数据交换方式。其次阐述了L2CAP层和RFCOMM层协议原理和实现方法。最后,通过实际操作解析了两台蓝牙设备间建立链接、配置链路、数据传输、断开链接的过程,并根据协议实现L2CAP层和RFCOMM层主要功能,同时给出了课题后续研发工作的思路 and 方向。

本文在论述过程中,穿插讨论了在开发过程中遇到的技术难点及其解决思路,并给出一些关键的程序代码。

实验证明,通过所设计的蓝牙协议,能够成功使得两台PC主机通过蓝牙设备建立无线连接,实现无线数据传输。

4. 期刊论文 [任秀丽](#). [于海斌](#). [Ren Xiuli. Yu Haibin](#) [Linux操作系统的蓝牙应用设计与实现](#) [—仪器仪表学报](#)

2006, 27 (z1)

蓝牙技术是一种短距离、低成本的无线通信技术,是一种能够实现语音和数据无线传输的开放性全球规范,为固定与移动设备通信环境建立一种连接方式. 本文在Linux操作系统下,设计并实现了一个典型应用—文件传输模型,同时给出了传输过程的流程. 测试结果表明,该系统具有良好的可靠性、可用性和可移植性,支持蓝牙产品的开发.

5. 学位论文 [曲晨](#) [基于QT的嵌入式综合媒体无线触摸屏系统](#) 2007

随着硬件成本的下降和嵌入式系统的普及,具有方便直观特点的触摸屏已经取代了传统的输入设备——键盘和鼠标,成为了目前公共场合最主要的人机交流输入设备。触摸屏易使用、坚固、反应速度快、节省空间的许多优点也使其有着非常广阔的应用前景。但是目前大部分基于触摸屏开发的系统都有着软硬件成本高、操作时不易进行移动的缺点,这也限制了触摸屏的应用。

本文根据会议室、家庭的特殊需要,设计了一套嵌入式综合媒体无线触摸屏系统。触摸屏上位机和执行功能的下位机采用分离设计的方式,用户通过触摸屏发送的命令经过无线通信传递给下位机,再由下位机执行相应的功能。即用户手持触摸屏终端点击按钮,命令就可以经由无线通信传递给下位机,下位机就可以执行诸如会议记录、打开照明、媒体播放等预先设定的功能,实现了一键完成对所有设备的操作。

本文所设计的上下位机系统均在嵌入式系统上进行开发,这样可以最大限度的使用硬件资源;触摸屏用户图形界面(GUI)软件在Linux操作系统上使用开源软件QT进行开发,极大的节省了软件的费用;下位机操作系统选择μC/OS-II嵌入式实时操作系统,满足了每个任务实时、独立运行的需求,也为今后的任务扩展留下了空间。该系统的设计给触摸屏的使用提供了一个新的思路,也使得嵌入式系统更广泛的应用在日常生活中,方便人们的生活。

6. 学位论文 [申研](#) [嵌入式WiMAX终端系统设计](#) 2007

随着人们对宽带无线通信需求的不断增加,无线宽带通信向更高速率、更大的覆盖范围、更好的移动性方向发展。WiMAX技术的出现正好满足了人们对于无线Internet的需求, WiMAX技术是“最后一公里”接入的最佳解决方案。

本文旨在提出一种嵌入式WiMAX终端系统的具体设计方案,即设计一个能够应用在WiMAX网络的嵌入式无线宽带终端。在此终端上可继续进行多种应用功能的开发,可以是无线 VoIP 手机、具有WiMAX 功能的 PDA、无线监控终端或者是在工业现场工作的通信模块等。

本文首先对WiMAX无线宽带接入技术以及嵌入式系统的开发做了较为全面、深入的阐述与分析。然后提出了基于本课题研究的可行的设计方案,并确定了最终选取的具体设计方案。

在系统设计中选用主处理器芯片+WiMAX 芯片的模块化结构。主处理器芯片选用Freescale公司的MC9328MX1处理器,在其上可开发多种终端应用功能; WiMAX 芯片选用富士通开发的MB87M3400 系统芯片,这是一款性能卓越、高度集成的WiMAX 系统芯片。由主处理器模块对WiMAX模块进行控制并进行数据传输,实现所设计终端系统的WiMAX接入。

在本课题研究的系统设计中,作者主要完成了以下几方面的工作:

1. 通过学习调研并进行可行性分析,确定了具体设计方案。
2. 提出系统的模块化结构,选取合适的硬件设备,完成系统硬件设计。
3. 进行系统软件设计工作,主要包括构建嵌入式 Linux 操作系统以及相应设备驱动程序的开发。

7. 期刊论文 [李熠, 胡修林, 陈勇全, 周奕, LI Yi, HU Xiu-lin, CHEN Yong-quan, ZHOU Yi 基于嵌入式Linux的点对多点](#)

[无线数据采集系统 - 电子工程师2006, 32 \(3\)](#)

介绍了一种在500m范围内进行多点数据采集的系统,采用多个远端采集器收集和发射远端传感器信号,采用ARM嵌入式系统和通信模块组成的主控制端接收信号并完成数据分析,主控制端采用ARM-Linux操作系统,通信协议采用自行设计的查询式通信协议,采用动态的从机注册机制管理远端采集器,经实际使用测试表明系统结构简单,性能稳定,可扩展性好,在不便于铺设有线电缆的场合有很高的实用价值。

8. 学位论文 [谢海波 嵌入式Linux的研究及其在一种无线通信数据处理机中的应用 2003](#)

该文主要研究了嵌入式Linux操作系统,并根据其特点将一个改造过的嵌入式Linux操作系统应用于一种以GPRS作为通信手段的无线通信数据处理机中。文章在介绍了嵌入式系统的一些特点及其开发模式之后,引出了嵌入式操作系统在嵌入式系统领域的重要地位。通过对几种实时操作系统的分析,将实时操作系统与通用操作系统进行了比较,从而得出Linux操作系统在嵌入式系统领域的主要一些优缺点。嵌入式Linux操作系统在应用过程中还存在着实时化和嵌入化不足的缺点,而这些也正是嵌入式Linux操作系统目前在国内外研究的热点。文章通过对国内外几种流行技术进行了对比介绍,并自己对Linux进行了几种针对性的改造,分析了改造结果,得出嵌入式Linux操作系统实时化和嵌入化改造后的结论。对于GPRS技术的研究及其在嵌入式Linux操作系统中的应用也是该文的一个重要内容。利用GPRS作为通信手段进行数据传输具有技术新、效果好、可平滑技术升级等特点,文章所描述的系统在GPRS业务一经推出之后便迅速推出产品抢得先机,这也是嵌入式系统领域在所谓后PC时代应用的一个典型例子。

9. 学位论文 [吴楠 基于远程医疗手术床的嵌入式控制器研究 2008](#)

在远程医疗手术床系统中,控制指令通过网络远端传送至手术床,按照指令要求,系统准确控制手术床的工作状态、各种姿态及位移等,从而进一步配合医疗机器人实施医疗手术。研究远程手术床控制的目的在于辅助远程医疗手术,为实现精确的远程手术提供帮助。

嵌入式控制器是远程手术床系统的核心部分,是实现系统控制的关键。它的上一级为手术床监控中心,与控制器相连的下一级是被控对象——手术床,其功能在整个远程手术床系统中类似于人的大脑,手术床系统的所有相关功能都要通过它才能得以实现。本文应用嵌入式技术,对控制器和上层手术床监控中心进行了研究。

本课题以嵌入式控制器为主要研究内容。首先,为控制器搭建了一个基本的嵌入式硬件平台,并选择了合适的Linux操作系统。在此基础上,开发以Linux为基础的触摸屏驱动和应用程序、网卡驱动和应用程序、GPRS扩展模块,通过这些软件和硬件外围扩展设备,为控制器控制手术床提供了现场及远程不同的方式。

第二,为实现嵌入式控制器对远程控制命令的接收、存储与处理,本课题采用异地与本地两种指令传输、动作控制方式。异地监视、控制方式包括以太网有线通信和GPRS无线通信两种,其中,以太网通信是主要的传输途径,GPRS无线通信作为以太网通信的辅助手段,保证了控制器网络通信的可靠性,构成了远程控制指令的双重传输模式,确保了指令安全可靠发送和接收。采用触摸屏控制,以方便、快捷的方式,实现了本地控制。

最后,利用VC++软件开发了上位监控中心平台控制界面,应用Winsock编写网络通信程序,构架了监控中心与嵌入式控制器的远程通信桥梁,同时初步尝试了远程视频监控功能。引入运动建模理论,并以RBT—4S01S机械臂模拟手术床。在以上研究的基础上,进行了控制指令传输的实验,实验结果表明可以达到控制手术床运动的预期要求。

本课题得到了国家高技术研究发展计划(863计划)所属预研子课题(2005AA421022-2)的支持,主要的工作是针对基于嵌入式的远程手术床系统的控制器,在其控制与通信方面进行了研究。所完成的控制器部分软硬件设计,为未来远程手术床系统的后续开发和实际应用奠定了基础。

10. 学位论文 [梅海军 基于“蓝牙”技术的嵌入式终端设备开发及硬件实现 2004](#)

随着无线通信网络的快速发展,蓝牙技术作为一种新型的无线通信技术逐渐发展和壮大起来,并迅速应用于手机、蓝牙耳机、标识牌等终端设备。目前,蓝牙技术是一种颇具优势的无线通信技术,逐渐成为嵌入式终端设备的研究热点问题。

本文就是基于“蓝牙”技术的基础上开发了嵌入式终端设备,综合运用了二维条码扫描技术、嵌入式操作系统技术和液晶显示技术等关键技术。文中着重研究了嵌入式终端设备的硬件设计思路并加以实现。从嵌入式产品开发的角度,对嵌入式终端设备进行了功能需求分析和功能规划设计,提出了嵌入式终端设备的核心板与接口板相互结合的模块化设计思想,基于此思想设计与制作了嵌入式终端设备的目标板。此外,还研究了蓝牙技术的实现原理,在嵌入式终端设备实现了蓝牙HCI层数据通信。

除此之外,提出了基于EPA的蓝牙接入装置的设计思路,开发了基于EPA的蓝牙接入装置,此装置很好解决了蓝牙无线通信与有线数据通信之间的切换。

本文链接: http://d.g.wanfangdata.com.cn/Thesis_Y1294566.aspx

下载时间: 2010年5月26日