北京邮电大学

硕士学位论文

嵌入式Linux蓝牙无线终端的研究与实现

姓名：葛亮

申请学位级别：硕士

专业：通信与信息系统

指导教师：白长清

20090215

嵌入式Linux蓝牙无线终端的研究与实现

摘 要

蓝牙技术是一种全球开放的无线通信技术标准，为用户提供低成本低功耗的短距离无线连接，已广泛应用于无线个域网中。经过多年的发展，如今蓝牙已经成为个人电脑和各种移动终端设备中必不可少的部件。

论文给出了基于蓝牙技术的智能终端的硬件结构和软件体系结构的总体设计方案。硬件平台是以 Freescale 公司的 ARM 内核的 32 位嵌入式微处理器 i.MX21 为控制核心，外部扩展了蓝牙模块、以太网模块、SD 卡存储模块等。文中对各个硬件模块的实现做了详尽的论述。在硬件平台的基础上，移植了嵌入式 Linux 操作系统，按照操作系统、驱动程序和应用程序的分层软件体系结构设计了系统软件，并针对蓝牙部分给出了详细的分析。然后，描述了蓝牙在网络接入中的应用场景，指出蓝牙在移动状态下接入局域网存在一定缺陷，例如，在网间移动情况下使用网络电话，由于网间漫游时延较大，严重影响通话质量。最后，使用一种中间件的方法，改进网间移动情况下实时业务的性能。

关键字： 蓝牙 Linux ARM 嵌入式系统 网络接入

RESEARCH AND IMPLEMENTATION OF EMBEDDED LINUX BULETOOTH WIRELESS TERMINAL

ABSTRACT

Bluetooth technology is a kind of global-open criteria of wireless communication, which provide with low-cost and low-consumption connection for user. It has been popular in the wireless personal area network, and been the necessary part of PC and terminal.

The paper presents the scheme of the intelligent terminal hardware and software system. The Freescale i.MX21 is the core of the intelligent terminal's hardware platform, which is the 32-bit ARM kernel embedded microprocessor, Periphery expands Bluetooth module, Ethernet communication module, SD card module and so on. The paper will depict the design of the hardware in detail. Based on the hardware platform, we port the embedded OS (operating system) Linux. And the whole software is divided into three parts of OS, driver and application software. And the software about Bluetooth will be in detail analysis. Afterwards, describe the application situation of the Bluetooth in the wireless LAN, and figure out there are some disadvantage in the mobile state. Such as, call network phone when moving between the micronetworks, the performance is bad because of the long hand-over time. At last, introduce the middleware to improve the performance of the quality of the real-time service when moving between micronetworks.

KEY WORDS: bluetooth Linux ARM embedded system network access

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：高亮　　日期：2009.3.1

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：高亮　　日期：2009.3.1

导师签名：[illegible]　　日期：2009.3.1

第一章 绪论

1.1 蓝牙技术的发展历史、现状、应用场所及研究的现实意义

蓝牙是从英文单词 Bluetooth 直译而来，是一种低功耗短距离的无线通信技术，其最初设计意图是取代现有的个人计算机、打印机、传真机等设备上接口的有线电缆。其主要优点是：可以方便地建立无线连接、代替传统的有线电缆连接；移植性较高，适用面广；而且蓝牙设备功耗低，成本也低，与其他无线通信设备相比，开发周期也较短。大大节省的生产的成本和时间，有利于快速的将产品推向市场。

1.1.1 蓝牙技术的发展历史简介

1994 年，爱立信移动通信公司为移动电话和电话附件之间寻找一种低功耗、低成本的无线接口，在爱立信公司的引领下，世界很多厂家的研发部门加入共同协作开发此技术的行列中。

1998 年 5 月，爱立信、IBM、英特尔、诺基亚和东芝等 5 家公司联合制定了短距离无线通信技术标准，其目的是为了实现最高数据传输率为 1Mb/s(有效传输率为 721Kb/s)、最大传输距离为 10m 的无线通信技术标准。并命名该技术标准为 Bluetooth。

1999 年 7 月，蓝牙 SIG（蓝牙国际组织）正式公布了蓝牙 1.0 版本规范。

2000 年 10 月，SIG 非正式的发布了 1.1 版本蓝牙规范，至到 2001 年 3 月，其 1.1 版规范才正式发布。

2003 年 11 月，SIG 公布了蓝牙 1.2 版本规范。其标准在实现设备识别高速化，减少与无线局域网电波干扰的同时，还能与现有的 1.1 版本完全兼容。

2004 年 11 月，蓝牙 2.0 标准（2.0 + EDR）正式推出，使蓝牙应用正式扩展到多媒体设备中，有望赋予蓝牙永久的生命力。新版的蓝牙标准具有更高的数据传输速率，提供的带宽是之前蓝牙带宽的 3 倍，而且在大量文件传输时，功耗只有旧版本蓝牙标准的一半。

1.1.2 蓝牙技术的现状与应用场所

如今，蓝牙技术已经成为许多智能终端设计中不可分割的特点。市场调查机构 Forrester 的数字也重点地指出：蓝牙技术的应用正从单一的移动电话领域向其他应用领域转移。

应用蓝牙技术的产品包括如下领域：

移动通信： 手机、无绳电话和传真机等。

计算机及其周边设备： 台式计算机、笔记本电脑、键盘、鼠标、游戏手柄等计算机配件，打印机扫描仪、摄像头等计算机外围设备。

蓝牙耳机等消费娱乐产品： 目前，市场最常见的蓝牙产品就是蓝牙耳机、蓝牙 MP3 等消费娱乐型产品，已经形成了一定的市场规模，相关的品牌也很多。

蓝牙汽车： 蓝牙在汽上的应用主要有两方面:一个方面是车载通讯系统，另一个方面是汽车的车轮定位仪。车载通讯系统可以免提接听拨打电话，来电时与汽车音响自动切换、自动静音，避免开车打手机的危险性。而具有蓝牙模块的车轮定位仪可以极大的提高操作的工作效率，用户可以自定义一些系统参数。目前市场上的高端汽车已经具备了以上功能。

蓝牙信息家电： 利用蓝牙无线通讯技术把传统的家电改造成蓝牙信息家电，在家庭中建立一个通讯网络，为家电信息提供必要的无线通路。然后在蓝牙 AP 的控制下，通过无线的方式实现对家庭网络上的家电和设备进行远程控制和监测。

商业： 蓝牙无线点菜机、零售终端。

另外，蓝牙技术在诸如智能楼宇监控系统、防盗报警系统、分布式测控系统、汽车工业和军事装备等相关领域也有广泛的应用价值。

1.1.3 蓝牙技术的研究意义

在近距离通信中，蓝牙无线接入技术是无线单元间的通信变得十分容易，将计算机技术与通信技术更紧密的结合在一起，人们可以随时随地进行信息的交换与传输。而且，蓝牙技术还可以为数字网络和外设提供通用接口，以组建远离固定网路的个人特别连接设备群。蓝牙作为一个全球公开的无线应用标准，通过把各种语音和数据设备用无线链路连接起来，使人们能随时随地地进行数据信息的交换与传输，无疑，它将在人们的日常生活和工作中扮演重要的角色，其市场潜力巨大。

1.2 嵌入式系统的组成和特点

由于嵌入式系统是计算机结构中的一个分支，所以它在硬件上的组成与标准的计算机类似，其中最重要的部分也是微处理器。与标准计算机结构相同，嵌入式系统中也包含了中央处理器、内存、输入/输出设备，只不过在嵌入式系统里，这些单元以比较特殊的形式存在。例如：计算机的标准输入设备为键盘，但是嵌入式的输入设备可能是触摸面板。

嵌入式系统一般有三个主要的组成部分：

1.硬件。嵌入式系统硬件一般由微处理器、存储器、输入输出设备以及其他专用系统电路组成。其中，微处理器是系统运算的核心，它性能的好坏直接影响到整个系统的性能。存储器用来保存可执行代码，以及中间结果；输入输出设备完成与系统外部的信息交换；其他系统专用电路是根据系统的特有的应用辅助系统完成相应的功能。

2.应用软件。应用软件是完成特定应用功能的程序。

3.嵌入式操作系统。该系统是用来管理应用软件，并提供一种机制，使得处理器分时地执行各个任务并满足一定的时限要求。对于小型的嵌入式系统可能只完成一个任务，因此不需要操作系统；而复杂的嵌入式系统一般会利用操作系统减小开发的工作量，并提高产品的可靠性。

嵌入式系统的关键在于结合系统硬件电路与特定的软件，以达到系统运行性能成本的最高比。其中硬件设计包括微处理器及存储器电路的设计、网络功能设计、无线通信设计及接口电路设计，等等。而嵌入式软件专门负责硬件电路的驱动、控制处理，以提高硬件产品的价值，是硬件产品不可或缺的部分。

嵌入式系统一般具有如下几个重要特征：

1.嵌入式系统通常是面向特定的应用。嵌入式处理器与通用型处理器的最大不同就是嵌入式 CPU 大多工作在为特定用户群设计的系统中，通常具有低功耗、体积小、集成度高等特点。

2.嵌入式系统的硬件和软件都必须高效率的设计，量体裁衣、去除冗余、这样才能满足功能、可靠性和功耗的苛刻要求。

3.操作系统支持。嵌入式系统的应用程序可以不需要操作系统的支持直接运行，但是为了合理的调度多任务，充分的利用资源，用户必须自此能够选配合适的操作系统开发平台，这样才能保证程序执行的实时性和可靠性，减少开发时间，保障软件质量。

4.专门的开发工具支持。嵌入式系统本身不具备自主开发能力，必须有一套开发工具和环境才能进行开发。开发工具和环境一般基于通用的计算机软硬件设

备。

1.3 论文结构安排及作者所做工作介绍

本论文一共分为五个章节，其主要内容如下：

第一章：介绍了蓝牙技术的发展历史、现状、及主要应用方式。并简单介绍嵌入式系统的主要组成和共同特点。

第二章：较为详细地介绍了嵌入式操作的概念，通过分析嵌入式 Linux 操作系统的优点及应用范围,说明了在本设计中选择嵌入式 Linux 操作系统的原因及其优点。

第三章：详细的介绍了系统的硬件及软件设计：包括所选的主控制器结构及其主要功能模块的硬件原理图设计，Linux 设备驱动开发的几个重要组成部分，蓝牙协议的主要功能模块。

第四章：描述了蓝牙在网络接入中的应用场景，指出蓝牙在移动状态下接入局域网存在一定缺陷，影响实时业务的质量，通过使用一种引用中间件的方法，改进网间移动情况下实时业务的性能。

第五章：论文工作总结与展望。

第二章 ARM-Linux 嵌入式技术介绍

2.1 ARM 微处理器介绍

ARM（Advanced RISC Machines），既可以认为是一个公司的名字，也可以认为是对一类微处理器的通称，还可以认为是一种技术的名字。1991 年 ARM 公司成立于英国剑桥，主要出售芯片设计技术的授权。目前，采用 ARM 技术知识产权（IP）核的微处理器，即我们通常所说的 ARM 微处理器。

ARM 微处理器结构特点：

1、RISC 体系结构

传统的 CISC（Complex Instruction Set Computer,复杂指令集计算机）结构有其固有的缺点,即随着计算机技术的发展而不断引入新的复杂的指令集,为支持这些新增的指令,计算机的体系结构会越来越复杂,然而,在 CISC 指令集的各种指令中,其使用频率却相差悬殊,大约有 20%的指令会被反复使用,占整个程序代码的 80%。而余下的 80%的指令却不经常使用,在程序设计中只占 20%,显然,这种结构是不太合理的。

基于以上的不合理性,1979 年美国加州大学伯克利分校提出了 RISC（Reduced Instruction Set Computer,精简指令集计算机）的概念,RISC 并非只是简单地去减少指令,而是把着眼点放在了如何使计算机的结构更加简单合理地提高运算速度上。RISC 结构优先选取使用频率高的简单指令,避免复杂指令;将指令长度固定,指令格式和寻址方式种类减少;以控制逻辑为主,不用或少用微码控制等措施来达到上述目的。

到目前为止,RISC 体系结构也还没有严格的定义。一般认为,RISC 体系结构应具有如下特点：

— 采用固定长度的指令格式,指令归整、简单、基本寻址方式有 2～3 种。

— 使用单周期指令,便于流水线操作执行。

— 大量使用寄存器,数据处理指令只对寄存器进行操作,只有加载/ 存储指令可以访问存储器,以提高指令的执行效率。

除此以外,ARM 体系结构还采用了一些特别的技术,在保证高性能的前提下尽量缩小芯片的面积,并降低功耗：

— 所有的指令都可根据前面的执行结果决定是否被执行,从而提高指令的

执行效率。

— 可用加载/存储指令批量传输数据,以提高数据的传输效率。

— 可在一条数据处理指令中同时完成逻辑处理和移位处理。

— 在循环处理中使用地址的自动增减来提高运行效率。

当然,和CISC架构相比较,尽管RISC架构有上述的优点,但决不能认为RISC架构就可以取代CISC架构,事实上,RISC和CISC各有优势,而且界限并不那么明显。现代的CPU往往采用CISC的外围,内部加入了RISC的特性,如超长指令集CPU就是融合了RISC和CISC的优势,成为未来的CPU发展方向之一。

2、ARM微处理器的寄存器结构

ARM处理器共有37个寄存器,被分为若干个组(BANK),这些寄存器包括:

— 31个通用寄存器,包括程序计数器(PC指针),均为32位的寄存器。

— 6个状态寄存器,用以标识CPU的工作状态及程序的运行状态,均为32位,目前只使用了其中的一部分。

同时,ARM处理器又有7种不同的处理器模式,在每一种处理器模式下均有一组相应的寄存器与之对应。即在任意一种处理器模式下,可访问的寄存器包括15个通用寄存器(R0~R14)、一至二个状态寄存器和程序计数器。在所有的寄存器中,有些是在7种处理器模式下共用的同一个物理寄存器,而有些寄存器则是在不同的处理器模式下有不同的物理寄存器。

3、ARM微处理器的指令结构

ARM微处理器的在较新的体系结构中支持两种指令集:ARM指令集和Thumb指令集。其中,ARM指令为32位的长度,Thumb指令为16位长度。Thumb指令集为ARM指令集的功能子集,但与等价的ARM代码相比较,可节省30%~40%以上的存储空间,同时具备32位代码的所有优点。

ARM微处理器的应用领域

基于ARM技术的微处理器应用约占据了32位RISC微处理器75%以上的市场份额,ARM技术正在逐步渗入到我们生活的各个方面:

1、工业控制领域:作为32的RISC架构,基于ARM核的微控制器芯片不但占据了高端微控制器市场的大部分市场份额,同时也逐渐向低端微控制器应用领域扩展,ARM微控制器的低功耗、高性价比,向传统的8位/16位微控制器提出了挑战。

2、无线通讯领域:目前已有超过85%的无线通讯设备采用了ARM技术,ARM以其高性能和低成本,在该领域的地位日益巩固。

3、网络应用:随着宽带技术的推广,采用ARM技术的ADSL芯片正逐步获得竞争优势。此外,ARM在语音及视频处理上行了优化,并获得广泛支持,

也对 DSP 的应用领域提出了挑战。

4、消费类电子产品：ARM 技术在目前流行的数字音频播放器、数字机顶盒和游戏机中得到广泛采用。

5、成像和安全产品：现在流行的数码相机和打印机中绝大部分采用 ARM 技术。手机中的 32 位 SIM 智能卡也采用了 ARM 技术。

除此以外，ARM 微处理器及技术还应用到许多不同的领域，并会在将来取得更加广泛的应用。

2.2 嵌入式操作系统简介

嵌入式操作系统 EOS（Embedded Operating System）是一种用途广泛的系统软件，过去它主要应用于工业控制和国防系统领域。EOS 负责嵌入式系统的全部软、硬件资源的分配、调度工作，控制协调并发活动；它必须体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能。目前，已推出一些应用比较成功的 EOS 产品系列。随着 Internet 技术的发展、信息家电的普及应用及 EOS 的微型化和专业化，EOS 开始从单一的弱功能向高专业化的强功能方向发展。嵌入式操作系统在系统实时高效性、硬件的相关依赖性、软件固态化以及应用的专用性等方面具有较为突出的特点。EOS 是相对于一般操作系统而言的，它除具备了一般操作系统最基本的功能，如任务调度、同步机制、中断处理、文件功能等外，还有以下特点：

1、体积小。嵌入式系统有别于一般的计算机处理系统，它不具备像硬盘那样大容量的存储介质，而大多使用闪存（Flash Memory）作为存储介质。这就要求嵌入式操作系统只能运行在有限的内存中，不能使用虚拟内存，中断的使用也受到限制。因此，嵌入式操作系统必须结构紧凑，体积微小。

2、实时性。大多数嵌入式系统都是实时系统，而且多是强实时多任务系统，要求相应的嵌入式操作系统也必须是实时操作系统(RTOS)。实时操作系统作为操作系统的一个重要分支已成为研究的一个热点，主要探讨实时多任务调度算法和可调度性、死锁解除等问题。

3、特殊的开发调试环境。提供完整的集成开发环境是每一个嵌入式系统开发人员所期待的。一个完整的嵌入式系统的集成开发环境一般需要提供的工具是编译/连接器、内核调试/跟踪器和集成图形界面开发平台。其中的集成图形界面开发平台包括编辑器、调试器、软件仿真器和监视器等。

2.3 嵌入式Linux的特点及应用前景

Linux 是个与生俱来的网络操作系统，成熟而且稳定。Linux 是源代码开放软件，不存在黑箱技术，任何人都可以修改它，或者用它开发自己的产品。Linux 系统是可以定制的，系统内核目前已经可以做得很小。一个带有中文系统及图形化界面的核心程序也可以做到不足 1MB，而且同样稳定。Linux 作为一种可裁减的软件平台系统，是发展未来嵌入设备产品的绝佳资源，遍布全球的众多 Linux 爱好者又能给予 Linux 开发者强大的技术支持。因此，Linux 作为嵌入式系统新的选择，是非常有发展前途的。

（1）与硬件芯片的紧密结合

后 PC 时代的智能设备已经逐渐地模糊了硬件与软件的界限，SOC 系统（System On Chip）的发展就是这种软硬件无缝结合趋势的证明。随着处理器片内微码的发展，在将来可能出现在处理器片内嵌进操作系统的代码模块。

嵌入式 Linux 的一大特点是：与硬件芯片(如 SOC 等)的紧密结合。它不是一个纯软件的 Linux 系统，而比一般操作系统更加接近于硬件。嵌入式 Linux 的进一步发展，逐步地具备了嵌入式 RTOS 的一切特征：实时性及与嵌入式处理器的紧密结合。

（2）开放的源代码

嵌入式 Linux 的另一大特点是：代码的开放性。代码的开放性是与后 PC 时代的智能设备的多样性相适应的。代码的开放性主要体现在源代码可获得上，Linux 代码开发就像是“集市式”开发，任意选择并按自己的意愿整合出新的产品。

对于嵌入式 Linux，事实上是把 BIOS 层的功能实现在 Linux 的 driver 层。目前，在 Linux 领域，已经出现了专门为 Linux 操作系统定制的自由软件的 BIOS 代码，并在多款主板上实现此类的 BIOS 层功能。

（3）嵌入式 Linux 与硬件芯片的紧密结合

对于许多信息家电的应用来说，嵌入的性能指标是最难满足的，只有靠提高芯片的集成度与装配密度来解决。嵌入式 Linux 与标准 Linux 的一个重要区别是嵌入式 Linux 与硬件芯片的紧密结合。这是一个不可逾越的难点，也是嵌入式 Linux 技术的关键之处。嵌入式 Linux 和商用专用 RTOS 一样，需要编写 BSP（Board Support Package），这相当于编写 PC 的 BIOS。这不仅仅是嵌入式 Linux 的难点，也是使用商用专用 RTOS 开发的难点。硬件芯片（SOC 芯片或者是嵌入式处理器）的多样性也决定了代码开放的嵌入式 Linux 的成功。嵌入式系统的发展，必然导致软硬件无缝结合的趋势，逐渐地模糊了硬件与软件的界限，在将

来可能出现SOC片内的操作系统代码模块。

随着处理器片内微码的发展，在将来应出现在处理器片内嵌进操作系统的代码模块,很显然模块将具有安全性好、健壮性强、代码执行效率高等特点。着眼于未来的嵌入式系统的发展，我们基于对嵌入式Linux技术的深入研究，对嵌入式处理器及SOC系统的深刻理解和研究；对EDA技术的深入研究；对模拟数字混合集成电路芯片的深入研究；对SOC片内进行嵌入式Linux操作系统代码的植入研究。此类的研究有可能减轻系统开发者对BSP开发的难度要求，并使得嵌入式Linux能够成为普及的嵌入式操作系统，而大大提高嵌入式Linux的易用性，提高其开发出的高智能设备的安全性、稳定性，同时也大大提高智能设备的计算能力、处理能力。

在嵌入式应用的领域里，从因特网设备到专用的控制系统，Linux操作系统的前景都很光明。由于Linux功能强大、可靠、灵活而且具有伸缩性，再加上它支持大量的微处理器体系结构、硬件设备、图形支持和通信协议，这些都使它作为许多方案和产品的软件平台越来越流行。

第三章 系统设计

3.1 嵌入式系统构架

一般嵌入式 Linux 系统构架包含硬件、Linux 内核、系统函数库以及应用程序组成。总的来说，硬件是整个系统需要基础，它给操作系统内核、函数库及应用程序的运行提供了必须的物理条件。硬件之上就是内核，使用内核的目的是希望以一致的方式管理硬件，以及为用户软件提供高层抽象层，其主要功能有驱动设备、管理 I/O 的存取、进程调度、共享存储空间等工作。应用程序通过操作系统提供的高层抽象层以及一些系统函数库，根据特定的应用场景来完成特定的应用任务。而硬件以上的部分又可以统一称为软件，一般固化在非易失性存储设备中，如 flash 等，由于在这种类型的存储器中，程序运行速度较慢，所以一般将程序搬移到运行速度快的动态存储器中在执行，然而这种存储器断电后所存储的内容会丢失，所以不能直接将软件固化在这种动态存储器中，为了解决这种矛盾，在固化程序时一般在操作系统前面固化一个引导程序（bootloader）,用于将操作系统等软件引导到动态存储器（DRAM）中执行。

本章下面将从硬件和软件两个方面详细介绍蓝牙终端的系统结构。

3.2 系统硬件总体设计介绍

本智能终端设计是以 Freescale 公司的 i.MX21 系列处理器 MC9328MX21VH 为硬件平台，MC9328MX21VH 是一款基于 ARM926EJ-S 内核的 32 位 RISC 嵌入式微处理器，主要面向可视电话、楼宇的对讲系统、VoIP、网络监控、多媒体终端产品、医疗电子设备、车载系统等高性价比、低功耗的智能终端的应用。如图 3.1 所示是智能终端的系统框图。

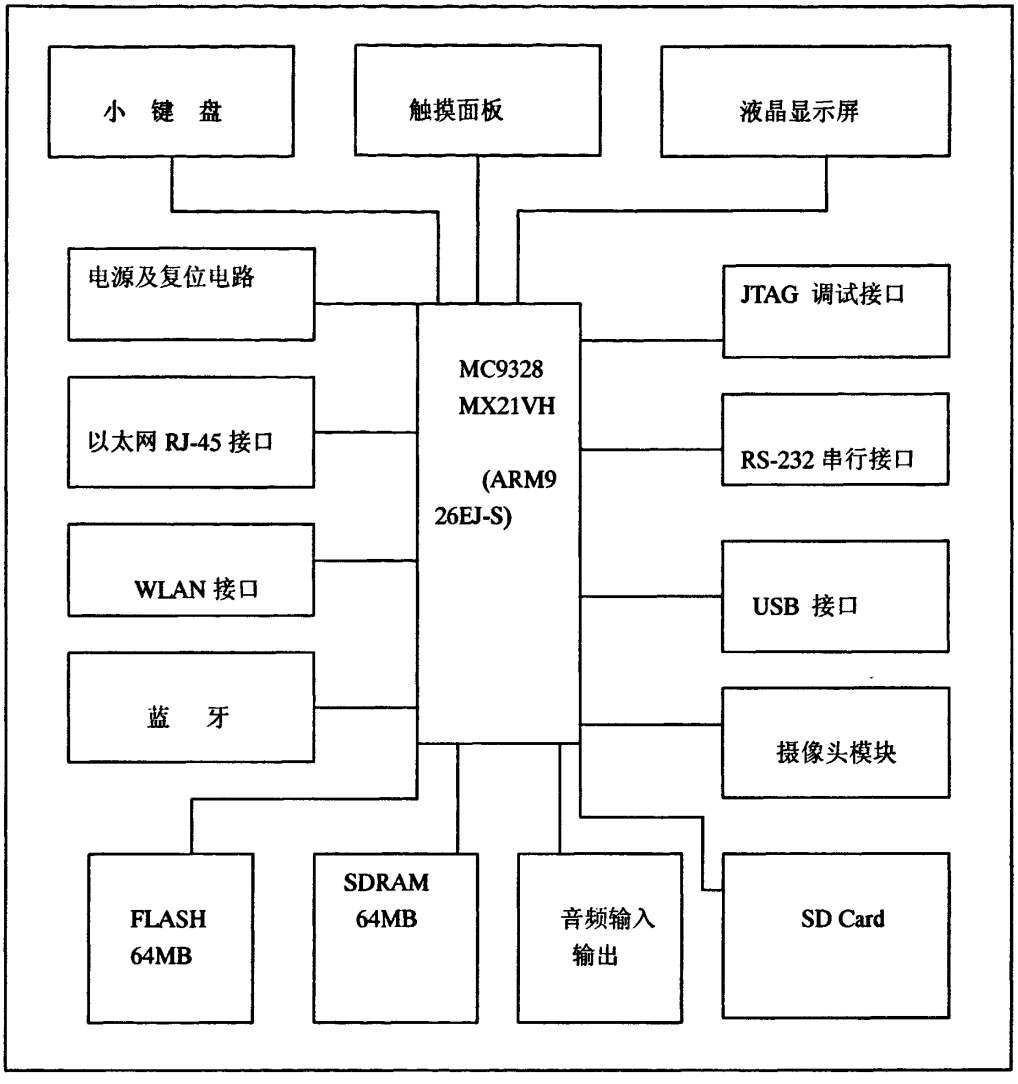

图 3.1　系统结构框图

该终端在尽可能小的电路板上集成了电源复位电路、64M SDRAM、64M Nand Flash、小键盘、触摸面板、液晶显示屏、JTAG 调试接口、RS-232、USB 接口、以太网接口、WLAN 接口、蓝牙接口、音频输入输出、SD 卡插座、摄像头模块等设备及接口。本章将在下面详细介绍部分重要模块的作用以及详细电路原理图的设计。

3.2.1 主控制器 MC9328MX21VH 概述

MC9328MX21VH 属于 Freescale 的 i.MX 系列产品，又称之为 i.MX21。i.MX21 采用先进的、低功耗的 ARM926EJ-S 内核，速度高达 266MHz。偏上集成了许多功能模块，例如：视频加速模块、LCD 控制器、USB On-The-Go、1-Wire 接口、

CMOS 感光器接口以及同步串行接口（SSI）。对于成本要求较高的应用，内置的 Nand Flash 控制器使得系统可以使用价格较低的 NAND Flash 做为主要的非易失性存储器。像 WLAN,Bluetooth 和其他的一些外部功能模块扩展可以利用 PCMCIA/CF,USB 以及 MMC/SD 主控制器实现，即简单又方便。功能框图如图 3.2 所示：

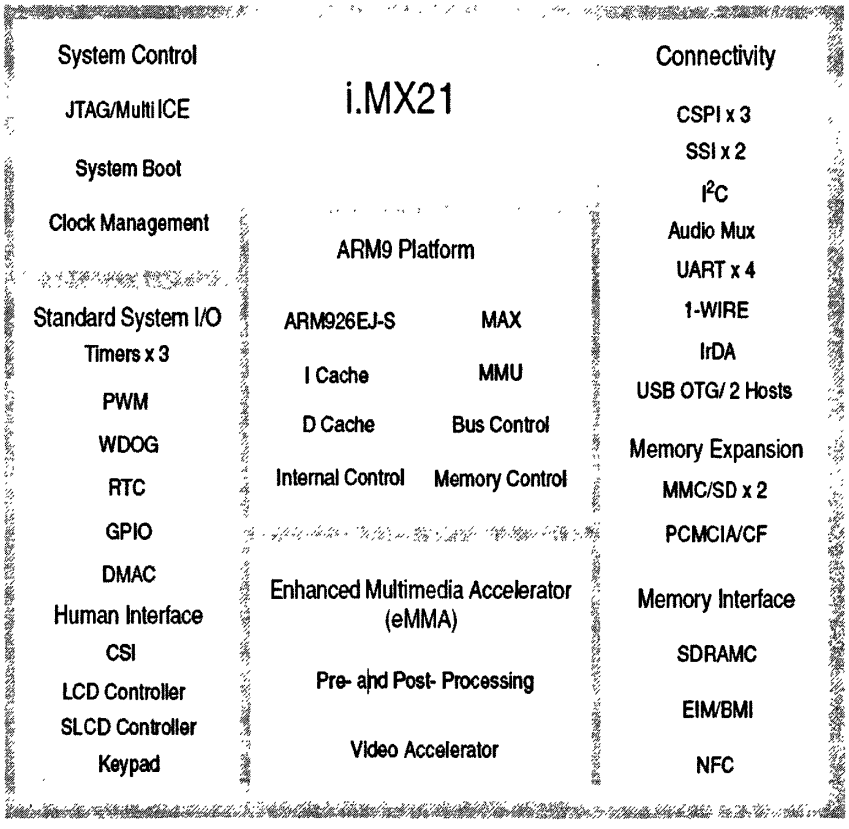

图 3.2 i.MX21 功能框图

i.MX21 主要功能特点详细如下：

◆ARM926EJ-S 内核

◆增强型多媒体加速器（eMMA）

◆显示即视频模块

——LCD 控制器（LCDC）

——轻便型 LCD 控制器（SLCDC）

——CMOS 感光器接口（CSI）

◆ 总线主控接口（BMI）

◆ 无线连接

——快速红外接口（FIRI）

◆ 有线连接

——USB On-The-Go(USB OTG)控制器

——四个 UART(UARTx)

——三个可配置的串行外部接口（CSPIx）,用于高速数据传输

——Inter-IC(I2C)总线模块

——两个带 I2S 的同步串行接口（SSI）

——数字音频混合器

——1-Wire 控制器

——键盘接口

◆ 存储卡扩展支持

——两个多媒体卡和安全数字（MMC/SD）主控制器模块

◆内存接口

——外部接口模块（EIM）

——SDRAM 控制器（SDRAMC）

——NAND Flash 控制器（NFC）

——PCMCIA/CF 接口

◆系统资源

——时钟产生模块（CGM）和电源控制模块

——三个通用 32 位计数/定时器

——看门狗定时器

——实时时钟/采样定时器（RTC）

——PWM 模块

——DMA 控制器

——GPIO 端口

——调试支持

3.3 系统主要功能模块介绍及电路原理图设计

该系统设计所包含主要功能模块可参考 3.2 节，本节将针对作者所参与设计的主要功能模块给予详细介绍，主要有存储器、串行接口、JTAG、以太网和蓝牙模块部分。

3.3.1 存储器系统设计

在该系统中，使用了两种类型存储器，Nor Flash 和 SDRAM。其中 Nor Flash

是用来存储引导程序，操作系统内核、文件系统，存储在Flash中的数据断电后不会丢失。SDRAM主要是用来存储执行中的程序和产生的数据，存储在SDRAM中的所有数据断电后会丢失。

3.3.1.1 Nor Flash接口电路设计

该系统使用的Flash存储器是28F640J3A,单片容量为64M位，16位的数据宽度，以16位（字模式）数据宽度的方式工作，工作电压仅需1.7V～1.95V，这可以有效的减少系统功耗。28F640J3A的引脚信号描述如表3.1所述：

表3.1 28F640J3A的引脚信号描述

引脚	类型	描　述
A24-A0	I	地址总线
DQ15-DQ0	I/O	数据总线
CE#	I	片选，低电平有效，与CLK异步相关
OE#	I	输出使能，低电平有效，与CLK异步相关
WE#	I	写使能
VCC	电源	设备供电
Vio	I	通用IO输入，应与VCC相连
VSS	—	接地
NC	—	悬空
RDY	O	有效数据已经准备好，可以执行读操作
CLK	I	时钟输入，突发模式时，在数据读出之后第一个CLK有效沿时，内部访问地址自动加一；异步模式时，CLK应处于VIL或VIH
AVD	I	地址有效。异步模式时，低电平表示地址有效；突发模式事，低电平使起始地址在下一个CLK有效沿被锁存。当AVD为高电平时，设备忽略地址输入。
RESET#	I	硬件复位。
WP	I	写保护。为VIL时，禁能编程，擦除操作。对于其他情况，WP应该为VIH。
ACC	I	加速输入。为VHH时，加速编程操作。为VIL时，禁能编程，擦除操作。其他情况，应为VIH。

RFU	保留	保留为将来使用

原理图如下

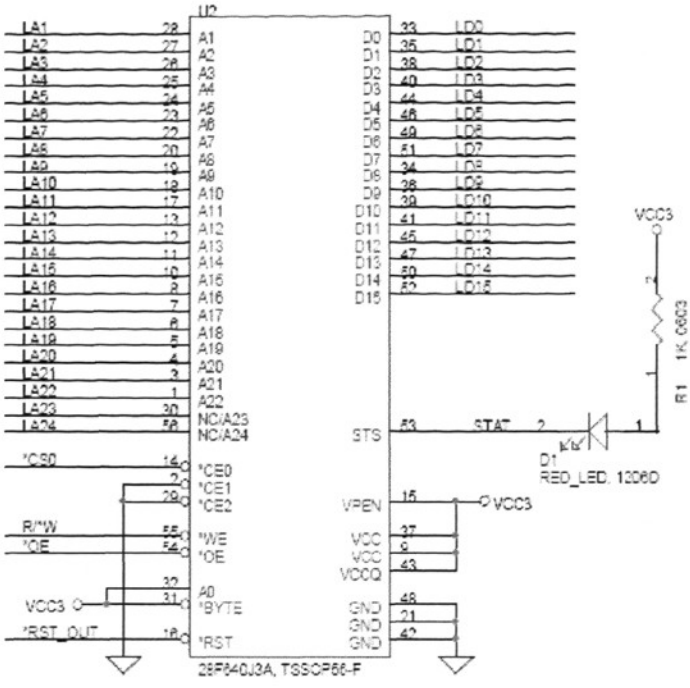

图 3.3 NorFlash 原理图

3.3.1.2 SDRAM 接口电路设计

与 Flash 存储器相比较，SDRAM 不具有掉电保持数据的特性，但是由于其存取速度大大高于 Flash 存储器，且具有读/写的属性，因此,SDRAM 在系统中主要用做程序的运行空间、数据区和堆栈区。当系统启动时，CPU 首先从复位地址处读取启动代码，在完成系统的初始化后，程序代码一般应调入 SDRAM 中运行，以提高系统的运行速度。同时，系统及用户运行堆栈、运行数据也都应放在 SDRAM 中。

SDRAM 具有单位空间存储容量大和价格便宜的优点，已广泛应用于嵌入式系统中。SDRAM 的存储单元可以理解为一个电容，总是倾向于放电，为避免数据丢失，必须定时刷新即充电。所以在系统中要使用 SDRAM 就要求微处理器要具有刷新控制逻辑，或者另外加入刷新控制逻辑电路。i.MX21 在片内集成了 SDRAM 刷新控制逻辑，可以方便的与 SDRAM 连接。

这里选用的 SDRAM 是 HY25L256160AF，它是一款专为移动手持设备设计的低功耗的 SDRAM 存储器。其引脚信号描述如下表所示：

表 3.2 HY25L256160AF 引脚信号描述

引脚	名称	描述
CLK	时钟	芯片时钟输入
CKE	时钟使能	片内时钟信号控制
CS#	片选	禁止或使能除 CLK、CKE 和 DQM 外的所有输入信号
BA0,BA1	组地址选择	用于片内 4 个组的选择
A12-A0	地址总线	行地址：A12-A10,列地址：A8-A0
RAS#	行地址锁存	时钟沿和 RAS#有效时锁存列地址
CAS#	列地址锁存	时钟沿和 CAS#有效时锁存行地址
WE#	写使能	使能写信号
LDQM,UDQM	数据 I/O 屏蔽	在读模式下控制输入输出缓冲；在写模式下屏蔽输入数据
DQ15-DQ0	数据总线	数据输入输出引脚
VDD/VSS	电源/地	内部电路及输入缓冲电源/地
VDDQ/VSSQ	电源/地	输出缓冲电源/地
NC	-	悬空

原理图如下

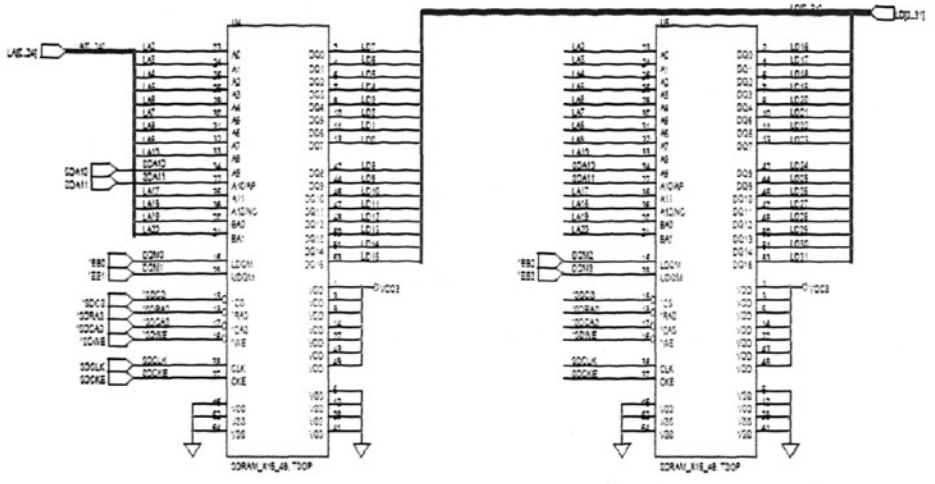

图 3.4 SDRAM 原理图

3.3.2 串行接口电路设计

串行通信是计算机系统中常用的通信机制之一，串行通信的数据是按位进行传输的，与并行通信相比，串行通信所使用的传输线少。目前，在国际上最通用的串行通信接口标准是由电子工业协会（EIA）制定的 RS-232C 标准。RS-232C 是一种常用的串行数据传输总线标准。RS-232C 采用的接口是 9 芯或 25 芯的 D 型插头。这里选用的是 9 芯 D 型插头，各引脚定义如下表：

表 3.3 9 芯 D 型引脚信号描述

引脚	名称	描述
1	DCD	数据载波检测
2	RXD	数据接收
3	TXD	数据发送
4	DTR	数据终端准备好
5	GND	地
6	DSR	数据设备准备好
7	RTS	请求发送
8	CTS	清除发送
9	RI	振铃指示

在这里串行接口主要用于与计算机主机建立起通信连接。通过在主机上运行一个串口程序，如超级终端、minicom 等，输入命令，并在终端上显示目标板相应的信息。

要完成基本的串行通信，实际上只需要 RXD、TXD 和 GND 三根线即可。但是由于 RS-232C 标准所定义的高低电平信号与目标板系统的 LVTTL 电路所定义的高低电平信号完全不同，LVTTL 的标准逻辑“1”对应 2～3.3V 电平，标准逻辑“0”对应 0～0.4V 电平，而 RS-232C 标准采用负逻辑方式，标注逻辑“1”对应-5～-15V 电平，标准逻辑“0”对应+5~+15V 电平。所以，两者之间要建立通信必须经过一个信号电平转换电路。这里选用的电平转换芯片为 MAX3232。

接口电路原理图，如下图所示：

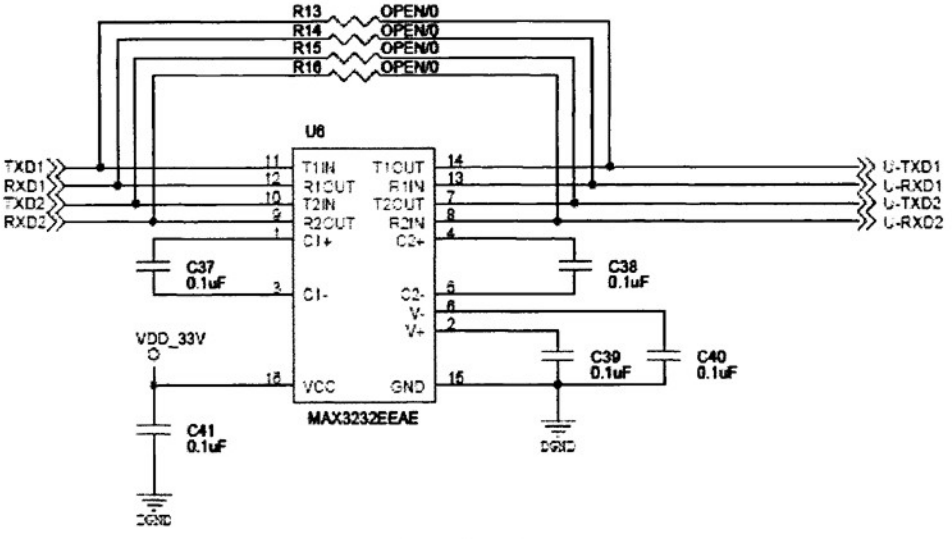

图 3.5 串口原理图

3.3.3 JTAG 调试接口电路

JTAG(Joint Test Action Group，联合测试行动小组)是 1985 年制定的检测 PCB 和 IC 芯片的一个标准，1990 年被修改后成为 IEEE 的一个标准，即 IEEE1149.1-1990。IEEE 1149.1 标准就是由 JTAG 这个组织最初提出的，最终由 IEEE 批准并且标准化的。所以，这个 IEEE 1149.1 这个标准一般也俗称 JTAG 调试标准。

JTAG 标准主要用于芯片内部测试及对系统进行仿真、调试。JTAG 技术是一种嵌入式调试技术，它在芯片内部封装了专门的测试电路 TAP(Test Access Port，测试访问口)，通过专门的 JTAG 测试工具对内部节点进行测试。标准的 JTAG 接

口是4线：TMS、TCK、TDI、TDO，分别为测试模式选择、测试时钟、测试数据输入和测试数据输出。JTAG 接口还常用于实现 ISP(In-System Programmable)功能,如对 FLASH 器件进行编程。

1.边界扫描

边界扫描技术的基本思想是在靠近芯片的输入输出管脚上增加一个移位寄存器单元。因为这些移位寄存器单元都分布在芯片的边界上，所以被称为边界扫描寄存器（Boundary-Scan Register Cell）。当芯片处于调试状态的时候，这些边界扫描寄存器可以将芯片和外围的输入输出隔离开来。通过这些边界扫描寄存器单元，可以实现对芯片输入输出信号的观察和控制。对于芯片的输入管脚，可以通过与之相连的边界扫描寄存器单元把信号（数据）加载倒该管脚中去；对于芯片的输出管脚，也可以通过与之相连的边界扫描寄存器“捕获”（CAPTURE）该管脚上的输出信号。在正常的运行状态下，这些边界扫描寄存器对芯片来说是透明的，所以正常的运行不会受到任何影响。这样，边界扫描寄存器提供了一个便捷的方式用以观测和控制所需要调试的芯片。另外，芯片输入输出管脚上的边界扫描（移位）寄存器单元可以相互连接起来，在芯片的周围形成一个边界扫描链（Boundary-Scan Chain）。一般的芯片都会提供几条独立的边界扫描链，用来实现完整的测试功能。边界扫描链可以串行的输入和输出，通过相应的时钟信号和控制信号，就可以方便的观察和控制处在调试状态下的芯片。

利用边界扫描链可以实现对芯片的输入输出进行观察和控制。那么，如何来管理和使用这些边界扫描链？对边界扫描链的控制主要是通过TAP（Test Access Port）Controller来完成的。

2.TAP 接口

在 IEEE 1149.1 标准里面，寄存器被分为两大类：数据寄存器(DR－Data Register)和指令寄存器(IR－Instruction Register)。边界扫描链属于数据寄存器中很重要的一种。边界扫描链用来实现对芯片的输入输出的观察和控制。而指令寄存器用来实现对数据寄存器的控制，例如：在芯片提供的所有边界扫描链中，选择一条指定的边界扫描链作为当前的目标扫描链，并作为访问对象。

TAP 是一个通用的端口，通过 TAP 可以访问芯片提供的所有数据寄存器（DR）和指令寄存器（IR）。对整个 TAP 的控制是通过 TAP Controller 来完成的。TAP 总共包括 5 个信号接口 TCK、TMS、TDI、TDO 和 TRST ：其中 4 个是输入信号接口和另外 1 个是输出信号接口。一般，我们见到的开发板上都有一个 JTAG 接口，该 JTAG 接口的主要信号接口就是这 5 个。下面，分别介绍这个 5 个接口信号及其作用。

◆TCK： 测试时钟输入（Test Clock Input），为 TAP 的操作提供了一个独立

的、基本的时钟信号，TAP 的所有操作都是通过这个时钟信号来驱动的。TCK 在 IEEE 1149.1 标准里是强制要求的。

◆TMS: 测试模式选择（Test Mode Selection Input），用来控制 TAP 状态机的转换。通过 TMS 信号，可以控制 TAP 在不同的状态间相互转换。TMS 信号在 TCK 的上升沿有效。TMS 在 IEEE 1149.1 标准里是强制要求的。

◆TDI: 数据输入（Test Data Input），所有要输入到特定寄存器的数据都是通过 TDI 接口一位一位串行输入的（由 TCK 驱动）。TDI 在 IEEE 1149.1 标准里是强制要求的。

◆TDO: 数据输出（Test Data Output），所有要从特定的寄存器中输出的数据都是通过 TDO 接口一位一位串行输出的（由 TCK 驱动）。TDO 在 IEEE 1149.1 标准里是强制要求的。

◆TRST: JTAG 复位信号（Test Reset Input），可以用来对 TAP Controller 进行复位（初始化）。不过这个信号接口在 IEEE 1149.1 标准里是可选的，并不是强制要求的。因为通过 TMS 也可以对 TAP Controller 进行复位（初始化）。

JTAG 是目前流行的一种调试嵌入式系统的简捷高效的手段。目前 JTAG 接口的连接有两种标准，14 针接口和 20 针接口。具体定义如下表所示。

表 3.4 14 针 JTAG 接口定义

引脚	名称	描述
1、13	VCC	接电源
2、4、6、8、10、14	GND	接地
3	nTRST	测试系统复位信号
5	TDI	测试数据串行输入
7	TMS	测试模式选择
9	TCK	测试时钟
11	TDO	测试数据串行输出
12	NC	未连接

表 3.5 20 针 JTAG 接口定义

引脚	名称	描述
1	VTref	目标板参考电压，接电源
2	VCC	接电源

3	nTRST	测试系统复位信号
4、6、8、10、12、14、16、18、20	GND	接地
5	TDI	测试数据串行输入
7	TMS	测试模式选择
9	TCK	测试时钟
11	RTCK	测试时钟返回信号
13	TDO	测试数据串行输出
15	nRESET	目标系统复位信号
17、19	NC	未连接

这里JTAG接口电路设计选用的是20针接口标准，其电路原理图设计如下：

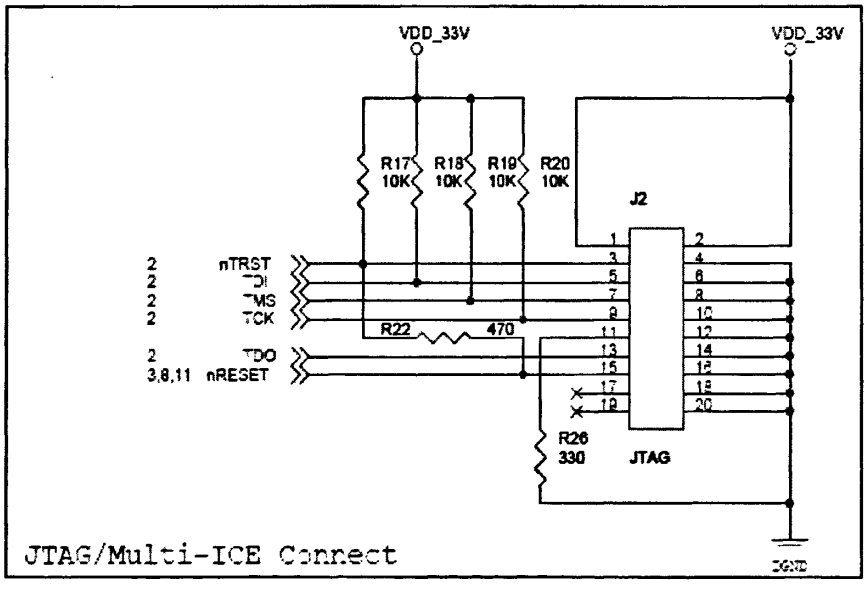

图3.6 JTAG原理图

3.3.4 SD卡模块设计

SD卡简介

SD卡（Secure Digital Memory Card）中文翻译为安全数码卡，是一种基于半导体快闪记忆器的新一代记忆设备，它被广泛地于便携式装置上使用，例如数码相机、个人数码助理(PDA)和多媒体播放器等。SD卡由日本松下、东芝及美

国 SanDisk 公司于 1999 年 8 月共同开发研制。大小犹如一张邮票的 SD 记忆卡，重量只有 2 克，但却拥有高记忆容量、快速数据传输率、极大的移动灵活性以及很好的安全性。

SD 卡在 24mm×32mm×2.1mm 的体积内结合了 SanDisk 快闪记忆卡控制与 MLC（Multilevel Cell）技术和 Toshiba（东芝）0.16u 及 0.13u 的 NAND 技术，通过 9 针的接口界面与专门的驱动器相连接，不需要额外的电源来保持其上记忆的信息。而且它是一体化固体介质，没有任何移动部分，所以不用担心机械运动的损坏。

SD 卡的技术建是基于 MultiMedia 卡（MMC）格式上发展而来，大小和 MMC 差不多，尺寸为 32mm x 24mm x 2.1mm。长宽和 MMC 一样，只是比 MMC 厚了 0.7mm，以容纳更大容量的存贮单元。SD 卡与 MMC 卡保持着向上兼容，也就是说，MMC 可以被新的 SD 设备存取，兼容性则取决于应用软件，但 SD 卡却不可以被 MMC 设备存取。（SD 卡外型采用了与 MMC 厚度一样的导轨式设计，以使 SD 设备可以适合 MMC)。

SD 接口除了保留 MMC 的 7 针外，还在两边加多了 2 针，作为数据线。采用了 NAND 型 Flash Memory，基本上和 SmartMedia 的一样，平均数据传输率能达到 2MB/s。

SDIO 简介

SD 插口的用途不止是插存储卡。支持 SDIO 接口的 PDA，笔记本电脑等都可以连接象 GPS 接收器，Wi-Fi 或蓝牙适配器，调制解调器，局域网适配器，条型码读取器，FM 无线电，电视接收器，射频身份认证读取器，或者数码相机等等采用 SD 标准接口的设备。

另外一些设备也宣布将支持，包括 RS-232 序列口适配器，指纹扫描仪，SDIO 转 USB 主/从适配器(可支持 SDIO 接口的手持设备使用 USB 外设或连接至电脑)，消磁读取装置，蓝牙/Wi-Fi/GPS 无线电收发器， 手机调制解调器 (个人通讯服务(PCS), CDPD, GSM 等)，和 APRS/TNC 适配器。

DRM 特色

SD 卡内嵌的数字版权保护方案是按 4C 提出的 可纪录介质内容保护标准 (CPRM) 所制定。其核心是使用了 Cryptomeria 密码（也称为“C2“）。这一特性是保密的。DVD-Audio 光盘也采用了与 CPPM 非常相似的加密方案。 SD 卡（Secure Digital Memory Card）是一种基于半导体快闪记忆器的新一代记忆设备。SD 卡由日本松下、东芝及美国 SanDisk 公司于 1999 年 8 月共同开发研制。大小犹如一张邮票的 SD 记忆卡，重量只有 2 克，但却拥有高记忆容量、快速数据传输率、极大的移动灵活性以及很好的安全性。

SD 卡体积小巧，广泛应用在数码相机上，是由日本的松下公司、东芝公司和 SanDisk 公司共同开发的一种全新的存储卡产品，最大的特点就是通过加密功能，保证数据资料的安全保密。SD 卡在外形上同 MultiMedia Card 卡保持一致，并且兼容 MMC 卡接口规范。不过注意的是，在某些产品例如手机上，SD 卡和 MMS 卡是不能兼容的。SD 卡在售价方面要高于同容量的 MultiMedia Card 卡。

SD 卡多用于 MP3 随身听、数码摄象机、数码相机等，其投影面积与 MMC 卡相同，只是略微厚一点，为 2.1mm，但是 SD 卡的容量大得多，且读写速度也 MMC 卡快 4 倍。同时，SD 卡的接口与 MMC 卡是兼容的，支持 SD 卡的接口大多支持 MMC 卡。目前 SD 卡在数码相机中正在迅速普及，大有成为主流之势。SD 卡在今年的发展很快，已经开始威胁到 CF 卡的市场分额了。这是由于 SD 卡的体积要比 CF 卡小很多，并且 SD 卡在容量、性能和价格上和 CF 卡的差距越来越小，而这两年支持 SD 卡的手机迅速在市场走热，因此，SD 卡的迅速成长绝对不是偶然的。最重要的一点就是 MMC 卡也能和 SD 卡相兼容，这也正是 SD 卡迅速走红的原因之一。

随着 SD 卡存储技术的发展，逐渐出现了 Mini SD 和 Micro SD 卡。

MiniSD 由松下和 SanDisk 共同开发。为了方便更多使用者能在不同存储卡中转换使用 mini SD，SanDisk 还特意推出了 SD 转接卡，可与现在使用 SD 卡的数字相机、PDA 掌上电脑和 MP3 音乐播放器共同使用。Mini SD 只有 SD 卡 37% 的大小，但是却拥有与 SD 存储卡一样的读写效能与大容量，并与标准 SD 卡完全兼容，通过附赠的 SD 转接卡还可当作一般 SD 卡使用。

在超小型存储卡产品上，SD 协会率先将 T-flash 纳入其家族，并命名为 Micro SD，只有指甲般大小的 Micro SD 在推出后，令消费者惊艳不已，同时，MOTO 的几款手机上也使用了这种存储卡。MMC micro 与 Micro SD 并列为目前全球最小的迷你存储卡，超小体积却拥有着更大的优势，可以运用于各类的数码产品，不浪费产品内部设计的空间，令产品设计者所喜爱，对于精致化数码生活也起到了“推波助澜”的作用。

SD 卡的使用

SD 卡应用于以下的手提数码装置：

- 数码相机储存相片及短片
- 数码摄录机储存相片及短片
- 个人数码助理（PDA）储存各类资料
- 手提电话储存相片、铃声、音乐、短片等资料
- 多媒体播放器

SD/MMC 主控制器引脚分配如下表所示：

表 3.6 SD/MMC 主控制器引脚分配

SD				MMC	
引脚	名称	描述	其他功能	名称	描述
1	DAT3	数据线[bit3]	卡检测（SD 和 SDIO）	保留	保留给以后使用
2	CMD	命令/应答	-	CMD	命令/应答
3	Vss1	地	-	Vss1	地
4	Vdd	电源	-	Vdd	电源
5	CLK	时钟	-	CLK	时钟
6	Vss2	地	-	Vss2	地
7	DAT0	数据线[bit0]	-	DAT	-
8	DAT1	数据线[bit1]	中断（仅 SDIO）	N/A	-
9	DAT2	数据线[bit2]	读等待（仅 SDIO）	N/A	-

电路原理图如下

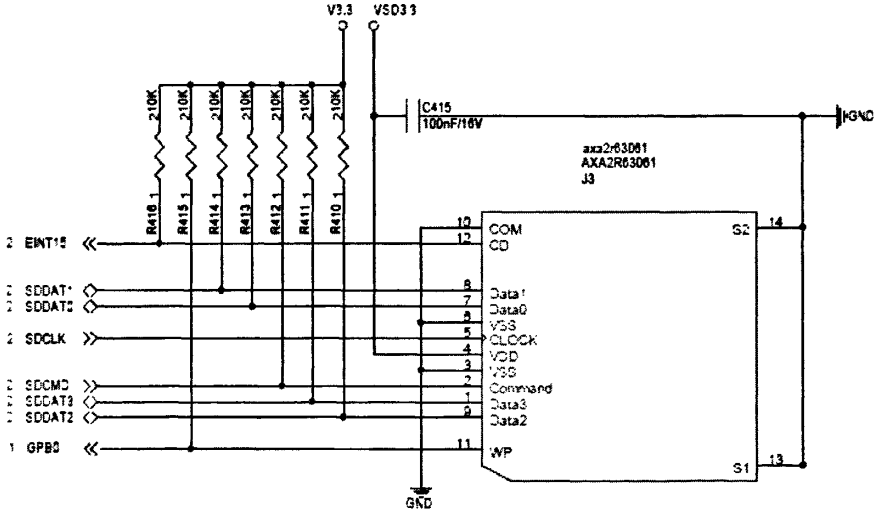

图 3.7 SD 卡接口原理图

3.3.5 以太网接口设计

在 ARM 系统中，以太网接口是与远程机进行通信及调试的基础，因此，就整个系统而言，以太网电路应是必不可少的。

从硬件的角度看，以太网接口电路主要由媒质接入控制 MAC 控制器和物理层接口两大部分构成。目前常见的以太网接口芯片，如 CS8900、RTL8019、DM9000 等，其内部结构也主要包含这两部分。这里所选用的芯片是 CS8900。

CS8900 芯片是 Cirrus Logic 公司生产的一种以太网处理芯片，提供 ISA 总线接口，内部集成了 EEPROM 控制器，802.3 MAC 控制器、片内 RAM 和 10BASE-T 收发滤波器，支持 I/O 和内存模式两种模式（这里使用 I/O 模式），可采用 DMA 方式与主机以 16 位或 8 位数据长度进行数据交换。如下图所示为 CS8900 的内部结构框架。

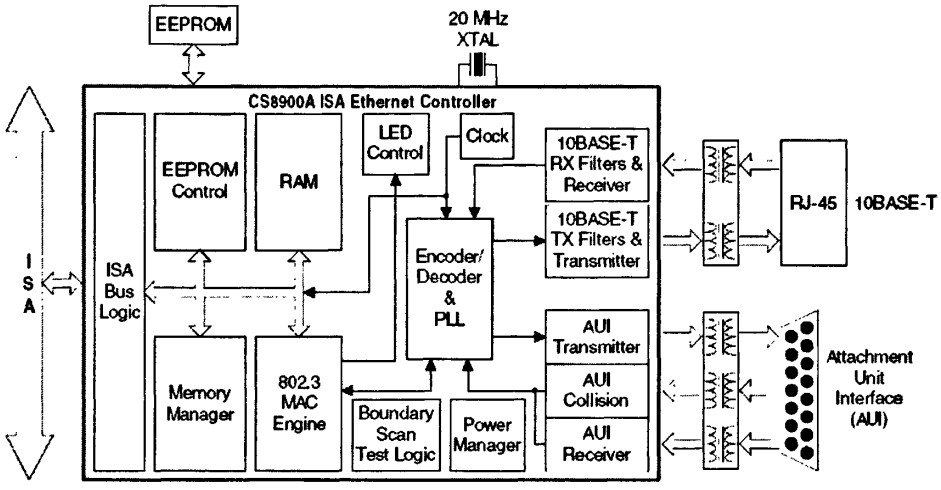

图 3.8 CS8900 结构图

在一般的嵌入式系统硬件原理设计时，CS8900 可以直接挂接在 CPU 的存储器总线上。将 CS8900A 网卡连接在处理器的数据总线上、控制总线和地址总线上的原理图如图所示。

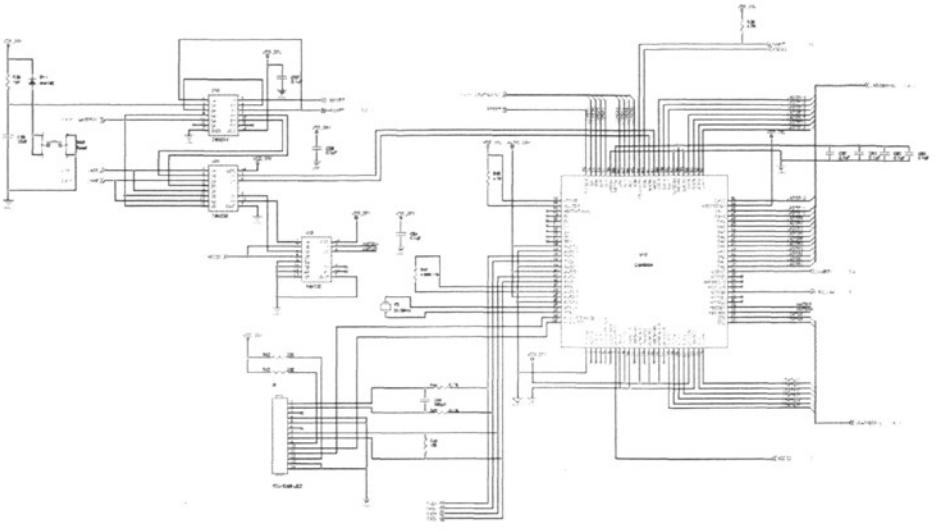

图 3.9 以太网部分原理图

CS8900 采用了独特的 PacketPage(封包页)机制来实现总线对其内部 4KB RAM 空间的访问。位于 CS8900 PacketPage RAM 空间的地址只需要通过直接映射到 CPU 内存或 I/O 空间的 16 字节连续的可直接访问的端口中的 PacketPage 指针端口和 PacketPage 数据端口即可间接访问，最大程度地减少了 CS8900 对硬件资源的占用。

当 CS8900 处于 I/O 模式下时，可以通过以下几个 PacketPage 空间内的寄存器（LINECTL、RXCTL、RCCFG、BUSCT、BUSST 等）来控制 CS8900 的行为，具体寄存器的说明可以参考 CS8900 的说明手册，这里就不再叙述。一般在初始化时对上述寄存器进行正确的配置，写入配置字之后就可以通过 PORT0、TXCMD、TXLENG 等寄存器进行收发数据包的操作。

在 I/O 模式下，发送数据包的步骤如下。

（1）向控制寄存器 TXCMD 寄存器写入发送命令。

（2）将发送数据长度写入 TXLENG 寄存器。

（3）读取 PacketPage 空间内的 BUSST 寄存器，确定其第 8 位被设置为 Rdy4TxNOW，即设备处于准备发送状态。

（4）将要发送的数据循环写入 PORT0 寄存器。

完成上述操作后，CS8900 会将数据组织为以太网帧并添加填充位和 CRC 校验信息，然后将数据转化为比特流传送到网络媒介。

在 I/O 模式下，接收数据包的方法如下。

（1）接收到网络适配器产生的终端，查询相对于 I/O 基地址 0008H 中断状态队列端口，判断中断类型为接收终端。

（2）读 PORT0 寄存器依次获得接收状态 rxStatus、接收数据长度 rxLength。

（3）循环继续对 PORT0 寄存器读取 rxLength 次，获得整个数据包。

3.3.6 蓝牙模块电路设计

目前蓝牙芯片供应商提供的芯片有两种：一种是集成了射频和基带部分的单芯片解决方案，另一种是射频芯片和基带芯片相互独立，采用射频芯片和基带芯片可构成与第一种功能相应的模块。为了缩短开发周期，减少 PCB 制作的复杂性，一般采用第一种方案开发，如 CSR 公司的 BlueCore 系列芯片。CSR 公司提供了全球首批比较完整的蓝牙兼容系统解决方案，其芯片得到了广泛的应用。这里使用了 BlueCore04b 实现了蓝牙部分的功能。

根据 HCI 的定义，分别可以用 UART 和 USB 两种接口方式实现芯片模块与嵌入式主控制器的连接，由于 USB 的速度快，配置方便的特点这里选用了 USB 的接口方式。

具体电路原理图如下图所示

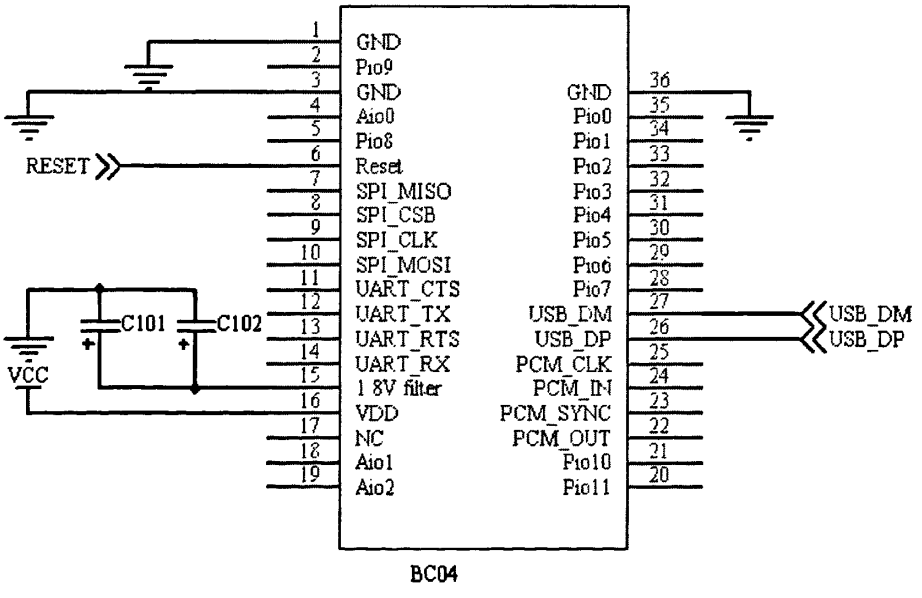

图 3.10 蓝牙模块部分原理图

3.3.7 PCB 板设计与制作

PCB 设计中，布线是完成产品设计的重要步骤，可以说前面的准备工作都是为它而做的， 在整个 PCB 中，以布线的设计过程限定最高，技巧最细、工作量最大。PCB 布线有单面布线、 双面布线及多层布线。布线的方式也有两种：

自动布线及交互式布线，在自动布线之前， 可以用交互式预先对要求比较严格的线进行布线，输入端与输出端的边线应避免相邻平行， 以免产生反射干扰。必要时应加地线隔离，两相邻层的布线要互相垂直，平行容易产生寄生耦合。

自动布线的布通率，依赖于良好的布局，布线规则可以预先设定， 包括走线的弯曲次数、导通孔的数目、步进的数目等。一般先进行探索式布经线，快速地把短线连通， 然后进行迷宫式布线，先把要布的连线进行全局的布线路径优化，它可以根据需要断开已布的线。 并试着重新再布线，以改进总体效果。

1 电源、地线的处理

既使在整个 PCB 板中的布线完成得都很好，但由于电源、 地线的考虑不周到而引起的干扰，会使产品的性能下降，有时甚至影响到产品的成功率。所以对电、 地线的布线要认真对待，把电、地线所产生的噪音干扰降到最低限度，以保证产品的质量。 对每个从事电子产品设计的工程人员来说都明白地线与电源线之间噪音所产生的原因， 现只对降低式抑制噪音作以表述：

众所周知的是在电源、地线之间加上去耦电容。

尽量加宽电源、地线宽度，最好是地线比电源线宽，它们的关系是：地线＞电源线＞信号线，通常信号线宽为：0.2～0.3mm,最经细宽度可达 0.05～0.07mm, 电源线为 1.2～2.5 mm 对数字电路的 PCB 可用宽的地导线组成一个回路, 即构成一个地网来使用(模拟电路的地不能这样使用)

用大面积铜层作地线用,在印制板上把没被用上的地方都与地相连接作为地线用。或是做成多层板，电源，地线各占用一层。

2 数字电路与模拟电路的共地处理

现在有许多 PCB 不再是单一功能电路（数字或模拟电路），而是由数字电路和模拟电路混合构成的。因此在布线时就需要考虑它们之间互相干扰问题，特别是地线上的噪音干扰。

数字电路的频率高，模拟电路的敏感度强，对信号线来说，高频的信号线尽可能远离敏感的模拟电路器件，对地线来说，整人 PCB 对外界只有一个结点，所以必须在 PCB 内部进行处理数、模共地的问题，而在板内部数字地和模拟地实际上是分开的它们之间互不相连，只是在 PCB 与外界连接的接口处（如插头等）。数字地与模拟地有一点短接，请注意，只有一个连接点。也有在 PCB 上不共地的，这由系统设计来决定。

3 信号线布在电（地）层上

在多层印制板布线时，由于在信号线层没有布完的线剩下已经不多，再多加层数就会造成浪费也会给生产增加一定的工作量，成本也相应增加了，为解决这个矛盾，可以考虑在电（地）层上进行布线。首先应考虑用电源层，其次才是地

层。因为最好是保留地层的完整性。

4 大面积导体中连接腿的处理

在大面积的接地（电）中，常用元器件的腿与其连接，对连接腿的处理需要进行综合的考虑，就电气性能而言，元件腿的焊盘与铜面满接为好，但对元件的焊接装配就存在一些不良隐患如:①焊接需要大功率加热器。②容易造成虚焊点。所以兼顾电气性能与工艺需要，做成十字花焊盘，称之为热隔离（heat shield）俗称热焊盘（Thermal），这样，可使在焊接时因截面过分散热而产生虚焊点的可能性大大减少。多层板的接电（地）层腿的处理相同。

3.4 嵌入式软件结构设计

嵌入式系统软件结构如下图所示，一般包括 bootloader、Linux 内核、驱动程序和应用程序等部分。关于各个部分的功能和作用在后面会做详细说明。

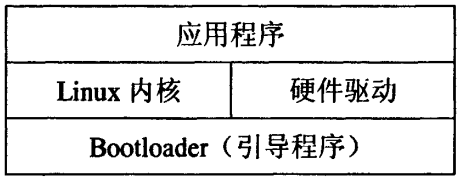

图 3.11 嵌入式软件结构图

3.4.1 嵌入式交叉编译环境的搭建

搭建交叉编译环境是嵌入式软件开发重要的一步，不管是 bootloader、Linux 内核、驱动程序还是应用程序，要最终运行在目标板上，都必须经过交叉编译，搭建交叉编译环境的方法很多，不同的体系结构，不同的操作系统甚至不同的内核版本，都会用到不同的交叉编译器，所以选择适合的交叉编译器对于嵌入式开发是非常重要的。

嵌入式软件开发过程的特点之一就是采用交叉编译。所谓交叉编译就是在一个平台上生成可以在另一个平台上执行的代码。编译最主要的工作就在将程序转化成运行该程序的 CPU 所能识别的机器代码，由于不同的体系结构有不同的指令系统。因此，不同的 CPU 需要有相应的编译器，而交叉编译就如同翻译一样，把相同的程序代码翻译成不同的 CPU 对应语言。要注意的是，编译器本身也是程序，也要在与之对应的某一个 CPU 平台上运行。嵌入式系统的交叉编译环境如下图所示。

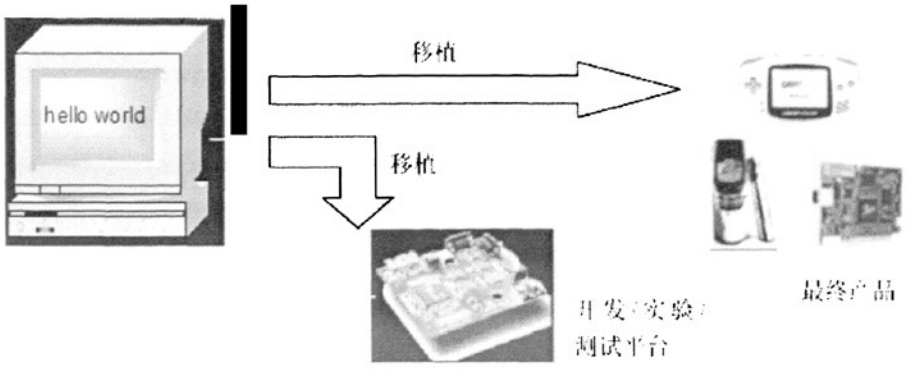

图 3.12 嵌入式交叉编译环境

这里一般把进行交叉编译的主机称为宿主机，也就是普通的通用计算机，而把程序实际运行环境称为目标机，也就是嵌入式系统。由于一般通用计算机拥有非常丰富的系统资源，可以使用方便的集成开发环境和调试工具，而嵌入式系统的系统资源非常紧缺，没有相关的编译工具，因此，嵌入式系统的开发需要借助宿主机来编译出目标机的可执行代码。由于一般的编译过程包括编译、链接等几个阶段，因此，嵌入式的交叉编译也包括交叉编译、交叉链接等过程。

为了安装交叉编译器，首先需要从网上下载编译器安装包，其中主要是:gcc,glibc,glibc-linuxthreads, gdb, binutils 2 具包的安装。一般都采用二 ke, make install 就可以直接装上。由于采用的是开发板供应商提供的开发工具包，他们已经把相关的工具打包到一起，我们只要把 arm-linux-toolchains.tgz 用命令 #tar xvzf arm-linux-toolchains.tgz -C/ 解压到根目录下，然后修改/etc/profile 文件，把交叉编译器的路径(/usr/local/arm/2.95 .3/ bin)加入到系统环境变量 PATH 中，这样交叉编译环境就建立起来了。

3.4.2 bootloader 介绍及启动流程分析

3.4.2.1 bootloader 基本概念介绍

Bootloader(引导加载程序)是嵌入式系统加电后运行的第一段代码。一般它只在系统启动时运行非常短的时间，但对于嵌入式系统来说，这是一个非常重要的系统组成部分。当我们使用单片机一般只需要在初始化 CPU 和其他硬件设备后，直接加载程序即可，不需要单独构建一个引导加载程序。但对于稍微大型的系统，构建或移植一个 Bootloader，能给后续的开发带来许多方便。在 PC 中，引导加载程序由 BIOS(其本质就是一段固件程序)和位于硬盘 MBR 中的操作系统引导加载程序(如 GRUB, LILO)一起组成。BIOS 在完成硬件检测和资源分配后，将硬盘 MBR 中的 Bootloader 读到系统 RAM 中，然后将控制权交给操作系统引导程序。引导加载程序的主要任务就是将内核映像从硬盘上读到 RAM 中，然后

跳转到内核的入口点去运行，即开始启动操作系统。

在嵌入式系统中，通常并没有像 BIOS 那样的固件程序(有的嵌入式 CPU 会在芯片内部嵌入式一段短小的程序，一般用来将 Bootloader 装载进 RAM 中，它有点类似 BIOS 的功能，但远比 BIOS 弱)，因此在一般典型的系统中，整个系统的加载启动任务就完全由 Bootloader 来完成。在一个基于 ARM 的嵌入式系统中，系统在上电或复位时通常都从地址 0x00000000 处开始执行，而在这个地址处安排的通常就是系统的 Bootloadere 通过这段小程序可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。大多数 Bootloader 都包含两种不同的操作模式:启动加载模式和下载模式。

1. 启动加载模式

在这种模式下，Bootloader 从目标机上的某个固态存储设备上将操作系统加载到 RAM 中运行，整个过程并没有用户的介入.这种模式是 Bootloader 的正常工作模式，因此在嵌入式产品发布时，Bootloader 必须工作在这种模式下。

2. 下载模式

在这种模式下，目标机上的 Bootloader 将通过串口、USB 接口或网络等通讯手段从开发主机上下载内核映像和根文件系统映像等到 RAM 中。然后可以再被 Bootloader 写到目标机上的固态存储介质上，或者直接进行系统的引导。前一种功能通常用于第一次烧写内核与根文件系统到固态存储介质时或者以后的系统更新时使用:后者多用于开发人员在前期开发的过程中。工作于这种模式下的 Bootloader 通常都会向它的终端用户提供简单的命令行接口。

Bootloader 与主机之间进行文件传输所用的通信设备及协议，最常见的情况通过串口与主机之间进行文件传输，传输协议通常是 Xmodem/Ymodem/Zmodem 协议中的一种。但是串口传输的速度是有限的，因此通过以太网连接并借助 TFTP 协议或是通过 USB 接口来下载文件是一个更好的选择。

Bootloader 是严重地依赖于硬件而实现的。每种不同体系结构的处理器都有不同的 Bootloader。不过 Bootloader 的发展也趋于支持多种体系结构，比如 U-Boot 从最初的只支持 PowerPC，到现在同时支持 PowerPC, ARM, MIPS, X86 等多种体系结构。除了依赖于处理器的体系结构外，Bootloader 实际上也依赖于具体的嵌入式板级设备的配置，也就是说，对于两块不同的嵌入式板而言，即使它们是由同一种处理器构建的，要想让运行在一块板子上的 Bootloader 程序也能运行在另一块板子上，通常也都需要对 Bootloader 进行移植。所以在嵌入式世界里建立一个通用的 Bootloader 几乎是不可能的.于是出现多种引导加载程序，如 armboot, blob, redboot, vivi 和 U-Boot 等。尽管如此，我们仍然可以对 Bootloader 归纳出一

个通用的启动流程来。

3.4.2.2 Bootloader 的启动流程

Bootloader 启动大多数都分为两个阶段。

第一个阶段(stagel)主要包含依赖于 CPU 的体系结构硬件初始化的代码，通常都用汇编语言来实现。这个阶段的任务有:

1. 基本的硬件设备初始化(屏蔽所有的中断、关闭处理器内部指令/数据 cache 等)。

2. 为第二阶段准备 RAM 空间。通常为了获得更快的执行速度，通常把 stage2 加载到 RAM 空间来执行,因此必须为加载 Bootloader 的 stage2 准备好一段可用的 RAM 空间。

3. 如果系统是从某种不支持代码执行的固态存储介质中比如 NandFlash 启动,则还需要复制 Bootloader 的第二阶段代码到 RAM。

4. 设置堆栈 sp,这是为执行 stage2 的 C 语言代码做好准备。

5. 跳 转 到第二阶段的 C 程序入口点。

第二阶段(stage2)通常用 C 语言完成，以便实现更复杂的功能，也使程序有更好的可读性和可移植性。但是与普通 C 语言应用程序不同的是，在编译和链接 Bootloader 这样的程序时,不能使用 glibc 库中的任何支持函数。在这个阶段 Bootloader 完成的任务有:

1. 初始化本阶段要使用到的硬件设备，包括初始化串口、初始化计时器等。在初始化这些设备之前、可以输出一些打印信息。

2. 检测系统内存映射。

3. 将内核映像和根文件系统映像从 Flash 读到 RAM。

4. 为内核设置启动参数。

5. 调用内核。

3.4.3 Linux 内核裁剪与移植

Linux 内核分为体系结构相关部分和体系结构无关部分。在 Linux 启动的第一阶段，内核与体系结构相关部分(在 arch 目录下)首先执行，它会完成硬件寄存器设置，内存映像等初始化工作。然后把控制权转给内核中与体系结构无关部分，系统的其余部分在这个阶段期间进行初始化。在移植工作中要改动的代码主要集中在与体系结构相关部分。内核目录树下的 arch 目录由许多子目录组成，每个子目录用于一个不同的体系结构(NIPS,AR M,13 86,SP ARC,PP C 等)。每一个这样的子目录都包含 kernel 和 Turn 子目录，它们包含特定于体系结构代码来完成诸如初始化内存、设置 IRQ、启用高速缓存、设置内核页面表等操作。一旦装入

内核并给予其控制，就首先调用这些函数，然后初始化系统的其余部分。Linux内核目录树中与ARM平台相关的部分主要为arch/arm”和include/asm-arm目录下的内容，详细内容如下:

1. 目录arch/arm下的子目录和重要文件如下:

kernel: 与ARM平台相关的内核代码，主要包括与ARM平台相关的汇编程序，如启动时的head.S;与异常中断处理相关的entry-armv.S;以及其它一些和体系结构中的中断异常处理，进程调度管理，系统调用等相关的C程序 。

mmu: 与ARM平台下内存管理部分相关的内核代码。比较重要的有:

(1)与内存管理相关的 mm-armvc,主要是针对ARM处理器中定义的页表，页目录表的操作。

(2)与内存异常处理相关的fault-armvc 。

(3)与各处理器的 MMU、cache 等内存管理部件操作相关的汇编代码，如pros-armXXX.S 等 。

lib: ARM 平台下经过了优化的内部库函数和低级I/O操作，例如内存拷贝函数memcpy ,I/O函数，位操作函数等，主要是用ARM汇编写成。Linux内核代码不能使用标准C库,需要自己实现一些C库函数。在内核树的主目录下也有一个目录lib对基本C库函数进行了实现，但如果特定的体系结构下有这些函数的实现，就优先使用这些实现。因为它们能针对硬件平台做最大程度的优化。

boot: 最终编译得到的内核就存放在这个目录下，并且还包含了一些为在启动时对压缩内核进行解压所需要的代码，如compressed目录下的文件head.S和misc.co。

tools: 用于自动生成象mach-types.h那样的文件的脚本文件。

configs: 包含每一种不同的机器类型的内核配置文件.内核配置文件是在对Linux内核进行编译的时候所需要的文件，它描述了系统从CPU类型到基本包含硬件方面的种种信息。可以说是对硬件系统的一个非常全面的概述。

mach-XXXX:对于每一种不同的机器类型，都有一些和该机器类型相关的一些程序。这是比kernel和mm目录下的代码更贴近底层硬件平台的代码。

vmlinux-armv.lds.in:链接描述文件，这是在生成内核的时候非常重要的一个文件.内核可执行文件由许多链接在一起的目标文件组成。目标文件有许多节,如文本、数据、init数据、bss等。这些目标文件都根据该文件链接并装入的。

2. 目录include/asm-arm/下的子目录内容如下:

arch-XXXX: 与特定机型相关的头文件。

hardware: 与ARM平台相关的一些芯片和设备的头文件。

mach: 包含机器用到的接口定义和对机器进行描述的宏。

3.4.3.1 ARM Linux 内核引导分析

Linux 内核引导部分的代码主要涉及到 CPU 及系统中各种硬件资源的初始化,是向特定硬件平台移植 Linux 内核时需要注意及修改的关键部分。ARM Linux 的启动流程图如下图示。

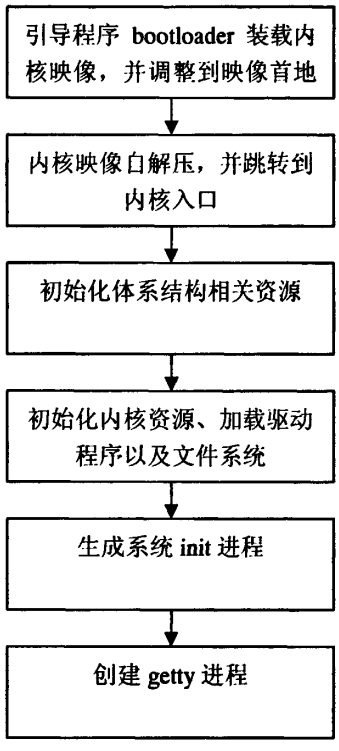

图 3.13 linux 内核启动基本流程

引导过程具体步骤如下:

1.压缩内核的启动:嵌入式系统中内核映像文件通常使用 zImage 或者 bzImage 的压缩格式，压缩映像文件存放于 flash:之中.在连接脚本文件 arch/am/boot/compressed/vmlinux.lds 为编译器指定了内核目标文件的连接顺序和地址。位于 arch/arm/boot/compressed/head.S 中的 start 段放在最前面，是整个压缩内核映像的入口。在 head.S 中通过调用 arch/arm/boot/compressed/misc.c 文件的 decompress_kernel()函数进行内核解压缩操作，在 decompress_kernel 函数中，打印“UncompressingLinux...”信息后调用 gunzip 函数进行实际内核解压缩工作，正确解压完成后打印出“Done, Booting the kernel” 信息并返回 head.s,接着跳转到 call_kernel 执行:

call_kernel: bl cache clean flush

bl cache off

mov r0, #0

mov rl, r7 @restore architecture number

mo vp c, r4 @ call kernel

其中，r7 机器类型号，r4=内核执行地址。

在跳转到真正的内核入口 arch/arm/kernel/head-armv.S 中的 ENTRY(stext)执行前，要求 CPU 的状态应该已满足如下条件:

r0=0;

r1:机器类型号;

MMU:必须关闭;

I-cache:不用关心，开/关皆可:

D-cache:必须关闭;

2. 非压缩内核的启动:非压缩内核即真正 Linux 内核入口在 arch/arm/kernel/head-armv.S 中的 stext 段。这些与体系结构相关的汇编代码执行一些底层的初始化设置工作，比如初始化内存,设置 CPU 寄存器，初始化内核堆栈，设置内核页目录、标识体系结构以及调用 start_kernel 函数等等,具体步骤如下:

进入内核入口的 ENTRY(stext)执行后，首先设置当前程序状态寄存器 CPSR 的中断禁止位与处理器模式位,确保处理器禁止 IRQ 和 FIQ,确保处理器处于 SVC 模式。

检查处理器，调用_lookup_processor_type 判断处理器类型，查找运行时处理器的 ID 值,是否与此 Linux 编译支持的 ID 值相等,不等则跳到错误处理;调用_lookup_architecture_type 判断体系结构类型，无效则跳到出错处理;

调用_create_page_tables 创建核心初始化页表，为内核映射 4M 的 RAM 空间,为 打开 MMU 做准备。

跳转到 arch/arm/mm/proc-arm929.sh 中的_arm920_setup 进行设置 CPU 的控制 寄存器、页目录地址等初始化工作;

设置控制寄存器，打开数据缓存、指令缓存和 MMU。ARM 的数据缓存必须和 MMU 一起打开，而指令缓存可以单独打开。

跳转至_mmap_switched 处执行，初始化 SP，使其指向 init 内核线程栈栈顶(init_ thread_union+8192)，清除 bss，保存处理器 ID 和机器架构类型到全局变量 processor_id 和_machine_arch_type。

跳转到 init/main.c 中的 start_kernel 函数,启动内核的进一步初始化工作。

3.从内核源码的 C 语言文件 init/main.c 中的 start_kernel 函数开始进一步的初

始化工作:初始化处理器平台、重新初始化内存管理、初始化异常和中断、设置系统定时器与控制台等，最后执行 reset_init 函数，以便调用其中的 kernel_thread 函数来创建一个内核线程，系统调用返回后，父进程执行 cpu_idl 函数，子进程执行 init 函数。

4. init 函数中调用 do_basic_setup 函数做最后的内核初始化工作，主要初始化网络模块等。然后 init 进程根据配置文件/etc/inittab 中每一行的要求生成新的进程以完成相应的系统启动初始化工作。在系统初始化结束后，最后由 in it 进程来创建终端注册进程 getty，屏幕显示"login:"开始用户登录，登录成功后 ，getty 进程将被 shell 进程所取代.自此整个系统就启动起来了。

3.4.4 Linux 驱动程序开发

3.4.4.1 设备驱动程序简介

设备驱动是操作系统的重要组成部分，对于一个特定的硬件设备来说，其所对应的设备驱动程序是不同的，比如网卡、声卡、键盘、鼠标、显卡等等。对于操作系统来说，挂接的设备越多，所需要的设备驱动程序也越多。操作系统本身并没有对种类繁多的硬件设备提供通用的设备驱动，操作系统在没有设备驱动程序支持下是无法正常支配硬件的行为的。这个时候就需要独立开发一套适合自己产品的设备驱动.对于嵌入式开发，更没有通用的驱动可以使用。因此，驱动程序开发是整个嵌入式系统设计过程中必不可少的一部分.设备驱动程序控制了应用程序和硬件设备之间的交互，下图简单描述了应用程序和设备驱动的关系。

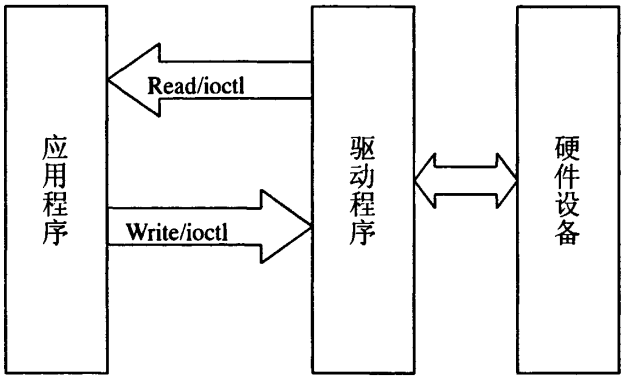

图 3.14 应用程序和设备驱动的关系

设备驱动程序的任务包括自动配置和初始化子程序，负责检测所要驱动的硬件设备是否存在和是否能正常工作。如果该设备正常，则对这个设备及其相关的

设备驱动程序需要的软件状态进行初始化。这部分驱动程序仅在初始化的时候被调用一次。应用程序对驱动程序的具体操作都要通过固定的驱动程序入口。

3.4.4.2 设备驱动程序的入口

在系统内部，I/O 设备的存取通过一组固定的入口点来进行，这组入口点是由每个设备的设备驱动程序提供的。设备驱动程序所提供的入口点，在设备驱动程序初始化的时候向系统进行登记，以便系统在适当的时候调用。

一般来说,设备驱动程序能够提供如下几个入口点:

1、open 入口点。打开设备准备操作。对字符设备文件进行打开操作，都会调用设备的 open 入口点。open 子程序必须对将要进行的 I/O 操作做好必要的准备工作，如清除缓冲区等。如果设备是独占的，即同一时刻只能有一个程序访问此设备，则 open 子程序必须设置标志以表示设备处于忙状态。

2、close 入口点。关闭一个设备.当最后一次使用设备结束后，调用 close 子程序。独占设备必须标记设备可再次使用。

3、read 入口点。从设备上读数据。对于有缓冲区的 I/O 操作，一般是从缓冲区里读数据。对字符设备文件进行读操作将调用 read 子程序。

4、write 入口点。向设备写数据。对于有缓冲区的写操作，一般是把数据写入缓冲区里。对字符设备文件进行写操作将调用 write 子程序。

5、ioctl 入口点。执行读、写之外的操作。

3.4.4.3 设备驱动的分类

Linux 系统的设备分为字符设备、块设备和网络设备三种。

字符设备是指存取时没有缓存的设备。块设备的读写都有缓存来支持，并且块设备必须能够随机存取。典型的字符设备包括鼠标，键盘，串行口等。字符设备和普通文件之间的主要区别是:普通文件可以来回读写，而大多数字符设备仅仅是数据通道，只能顺序读写。字符设备是 Linux 最简单的设备，可以像文件一样访问。应用程序使用标准系统调用打开、读取、写和关闭，好像这个设备是一个普通文件一样。初始化字符设备时，它的设备驱动程向 Linux 登记，并在字符设备向量表中增加一个 device struct 数据结构条目。

块设备是文件系统的物质基础，它也支持像文件一样被访问。这种为打开的块设备文件提供的操作组的机制和字符设备的十分相似。Linux 用 blkdevs 向量表维护已经登记的块设备文件，使用设备的主设备号作为索引。它的条目也是 device struct 数据结构.和字符设备不同，块设备进行分类:SCSI 是其中一类，而 IDE 是另一类，向 Linux 内核登记并向内核提供文件操作。块设备的设备驱动程序向这种类提供相关的接口，硬盘、CDR 服等是典型的块设备。

字符设备和块设备的主要区别是:在对字符设备发出读/写请求时，实际的硬

件 I/O 一般就紧接着发生了，块设备利用一块系统内存作缓冲区，当用户进程对设备请求能满足用户的要求，就返回请求的数据，如果不能，就调用请求函数来进行实际的 I/O 操作。块设备是主要针对磁盘等慢速设备设计的，以免耗费过多的 CPU 时间来等待。网络设备在 Linux 里做专门的处理，Linux 的网络系统主要是基于 BSD unix 的 Socket 机制，在系统和驱动程序之间定义有专门的数据结构(sk_buff)进行数据的传递，系统里支持对发送数据和接收数据的缓存。

3.4.4.4 蓝牙核心协议分析

蓝牙技术标准的开发性使设备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，也就是说只有遵守协议标准的蓝牙软硬件模块才能实现与其他蓝牙设备的互连互通，所以在分析蓝牙驱动之前有必要先介绍下蓝牙协议栈的基本结构。下图所示为蓝牙协议体系

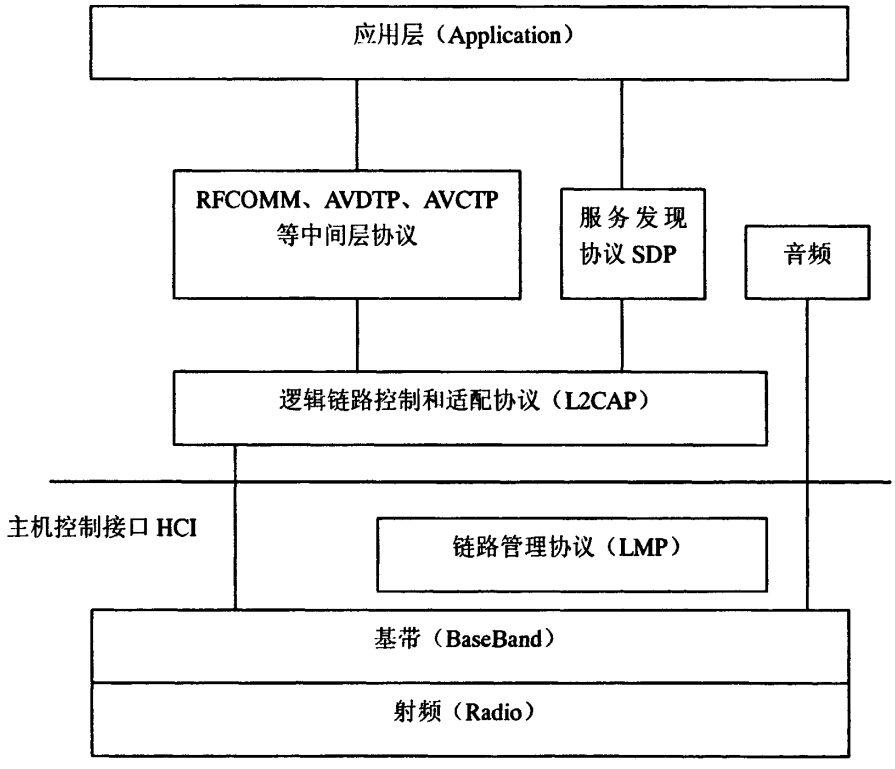

图 3.15 蓝牙体系结构

核心协议包括基带（BB）、链路管理协议(LMP)、逻辑链路控制和适配协议(L2CAP)、服务发现协议(SDP)，它构成一个蓝牙设备的核心系统，其他的协议和剖面都依赖于核心协议。中间层协议在逻辑链路上进行复用，为上层应用提供

服务，根据不同的应用剖面设备会选择相应的中间层协议，如 HFP 需要有 RFCOMM 协议支持，音视频的应用则需要 AVDTP 的支持。应用剖面规定了蓝牙技术在具体应用中应符合的特性，每种蓝牙设备都必须符合相应的应用剖面，这是实现互操作性的重要手段。

1、基带协议(BB)

基带协议确保各个蓝牙设备之间的物理射频连接，以形成微微网。蓝牙的射频系统使用跳频扩频技术，其任一分组在指定时隙通过指定频率进行发送，这层使用查询(Inquiry)和呼叫(Page)进程来同步不同设备间的传输跳频频率和时钟。

基带相对应的基带数据分组提供两种不同的物理链路：同步面向连接(SCO)和异步无连接(ACL)，其中 ACL 可以在同一个射频系统中采用多路技术的方式进行传输。ACL 只能传输数据分组；而 SCO 既能传输语音分组也能传输数据分组。所有的语音和数据分组都附有不同级别的前向纠错(FEC)或循环冗余校验(CRC)编码，并可以进行加密，以保证可靠传输。另外，对于链路管理信息和控制信息分别分配一个特殊的传输信道。

包含语音数据的分组可以使用不同的应用模型在一个或多个蓝牙设备上传输。SCO 分组中的话音数据与基带有直接通路，而不需要通过 L2CAP。话音模型在蓝牙规范中相对简单，任意两个蓝牙设备仅通过开通一条话音链路，就可以相互发送和接收话音数据。

2、链路管理协议(LMP)

链路管理协议不但负责蓝牙各设备间链路的建立和控制，还用于安全方面的鉴权和加密;另外，还可以控制无线部分的能量模式和工作周期、微微网内的各设备的连接状态。

每个设备上的链路管理器(LM)利用 LMP 协商彼此之间蓝牙空中接口的特性。其中包括带宽的分配，设备间协商确定基带数据分组的大小，通过支持适配协议数据业务所需的服务级以及保留的周期性带宽，来支持话音通信业务。通信设备上的蓝牙 LMP 利用“竞争一响应”的方式对设备进行鉴权，产生、交换、核实链路和加密连接密钥，以进行身份认证和加密等安全措施。在必要时，对 LM 监控设备的配对和对设备之间空中接口上的数据流加密，其中配对是通过产生和存储连接密钥来建立起设备之间的相互信任关系，为以后的设备鉴权做准备。如果鉴权失败，LM 将切断设备间的链路，以禁止设备间的任何通信。LM 还支持能量控制，通过交换彼此间的参数信息(例如低活动性基带模式的周期)来协商低活动性基带运行方式，从而控制功耗。

接收端的链路管理器对 LMP 消息进行过滤和解释，从而它们不会向上层传递。因为 LMP 消息的优先权大于用户数据，所以如果一个链路管理器需要发送

一条消息，不会被 L2CAP 会话延迟。另外，逻辑信道通过了一个可靠的链路，所以使得 LMP 消息不需要被普遍公认。

3、逻辑链路控制和适配协议(L2CAP)

逻辑链路控制和适配协议完成基带与高层协议间的适配，并通过协议复用、分割及重组操作为高层提供数据业务和分类提取。来自数据应用的通信信号首先通过 L2CAP，L2CAP 层屏蔽了高层协议和应用层与低层传输协议之间的关联。因此，高层协议不需要知道在无线电波和基带上的调频序列，也不需要知道在蓝牙接口中传输的特殊的分组格式。L2CAP 支持协议复用，允许多个协议和应用共享空中接口，它支持分组的分割和重组，将高层使用的大分组分割成适合于基带传输的小分组，在设备接收端又将这些小分组重组。最后，两个对等设备上的 L2CAP 层通过协商达成一个双方都能接受的业务等级，并能维护和保持此业务级别。基于要求的业务等级，一个 L2CAP 的使用既能行使允许新的通信信号进入的控制权，同时与低层协调，以保持所需的业务等级。

逻辑链路控制和适配协议是基带的上层协议，可以认为它是与 LMP 并行工作的，它们的区别在于，当数据不经过 LMP 时，L2CAP 将采用多路技术、数据分组分割和重组技术、群提取技术以及服务质量等，为上层提供数据服务。虽然基带协议提供了 SCO 和 ACL 两种连接类型，但是 L2CAP 只支持 ACL，并允许高层协议以每秒 64K 字节的速率收发数据分组。话音和电话应用的语音质量信道通常在基带 SCO 链路上运行，然而话音数据可以打包并使用通信协议在 L2CAP 上进行传输。

4、服务发现协议(SDP)

服务发现协议(SDP)是蓝牙技术框架中非常重要的一个部分，它是所有应用模型的基础。任一蓝牙应用模型的实现都是利用某些服务的结果。在设备之间组网的基本动机就是使这些设备相互通信，并获得彼此的服务。多数情况下传统网络的客户机通过一些静态设置确定这些网络业务的位置，如以太网通过服务器在网络中提供文件传输、打印、命名、网桥和网关服务，供客户使用。客户机的设置是由系统管理员建立并维护的，而对于蓝牙无线通信来说，建立在蓝牙链路上的任何两个或多个设备随时都有可能开始通信，仅仅是静态设置是不够的。如果这些设备要能够相互利用彼此间的业务，就需要确定这些业务位置的动态方式。一旦建立起一条通信信道，就能够找到需要的业务。这就是蓝牙服务发现协议的功能。

使用 SDP，可以查询到设备信息、服务和服务类型。在对邻近的可获得的服务定位以后，蓝牙设备才能建立连接。SDP 是在动态网中发觉终端用户使用价值的重要环节。蓝牙 SDP 是专门为使用蓝牙无线通信的环境设计的，以有效和优

化的方式执行该项功能。SDP 支持三种查询方式:按服务类别搜寻、按服务属性搜寻和业务浏览。

5、蓝牙主控制器接口概述

主控制器接口（HCI）功能规范就是蓝牙与主机系统之间的接口规范，提供控制基带与链路控制器、链路管理器、状态寄存器等硬件功能的指令分组格式以及进行数据通信的数据分组格式。即提供了一种访问蓝牙硬件能力的通用接口。

3.4.4.5 蓝牙驱动模块分析

由图 3.15 蓝牙体系结构图可以看出在实现蓝牙的时候，一般是将蓝牙分成两部分来实现的，一部分是与具体蓝牙控制器硬件相关的软件实现部分，它位于HCI 的下面,亦即底层硬件模块（这里是由 CSR 公司的 BC04 模块自带的固件来实现的）；另一部分是与具体蓝牙控制器硬件无关的软件实现部分，位于 HCI 的上面,包括了蓝牙协议栈上层的 L2CAP、RFCOMM、SDP 和 TCS 以及蓝牙的一些应用。两部分软件之间的通信是通过 HCI 层来实现的。由于与 BC04 相关的软件是完全固化在模块中的 flash 中的，其具体实现是不可见的，因此本章以下部分将详细介绍 HCI 层及其位于其上的软件部分实现。

1、HCI USB 传输层

使用 USB 连接蓝牙模块有两种方法：一是作为 USB 专用模块，二是与主控制器直接连接在一个 PCB 板上。这里使用的是第二种方法。

USB 端点要求

描述符：USB 固件配置由两个接口组成：第一个接口为固定设置，第二个接口提供可扩展的等时带宽占用能力。

一个 HCI 帧应该包含在一个 USB 事务中，一个 USB 事务定义为一个或多个 USB 帧，一个输入输出端口的请求数据都包含在这些 USB 帧中。端点分布在两个接口，因此当调整占用的带宽时，所有挂起的 Bulk 和中断的交易不需要终止或重新发送。

控制端点要求：端点 0 用于配置和控制 USB 设备，也用于允许主机向主机控制器发送特定的 HCI 命令。当 USB 固件通过该端点收到一个具有蓝牙分类码的分组，它就按 HCI 命令分组对待。

Bulk 端点要求：对于 ACL 分组，数据的完整性是一个重要的方面。数据的完整性要求和带宽的要求是使用 Bulk 端点的原因。建议的最大 Bulk 分组是 64 字节。通过总线每毫秒可以发送多个 64 字节分组。

中断端点的要求：为确保事件可以预测并按时的发送，中断端点是必须的。通过 USB 发送的事件分组时延是有保证的。中断端点的间隔是 1ms。USB 软件和固件对送往主机控制器的事件不需要太多了解。

等时端点的要求：等时端点用于向（从）主机控制器发送（接收）SCO 数据。对于此类应用，时间是一个重要参数。USB 固件把数据的内容送到主机控制器的 SCO FIFO 中。如果 FIFO 满，将使用新数据进行覆盖。

2、 逻辑链路控制与适配协议 L2CAP 代码分析

L2CAP 位于基带协议层之上，属于数据链路层，是一个为高层传输和应用层协议屏蔽基带协议的适配协议。L2CAP 目标是提供一种类似 TCP/IP 函数调用的接口。L2CAP 采用协议复用、分段和重组操作以及组抽象等方式向高层协议提供了连接和无连接数据服务。L2CAP 只支持 ACL 链路。L2CAP 采用了信道的概念在蓝牙设备不同应用之间建立不同的路径。信道由信道标识符(CID)表示，代表设备上每个应用程序连接的逻辑端点。CID 是一个 16 位长的数字，其中 0x0001 到 0x003F 一段保留给特定 L2CAP 功能使用(0x0001 是信令信道，0x0002 是无连接接收信道，其他都被保留或禁用)。L2CAP 允许高层协议和应用收发 64KB 的 L2CAP 分组。

L2CAP 事件和动作

L2CAP 的操作采用上下层间收发事件和动作。这些事件包括来自上层的连接请求、写数据请求或者断开连接请求等。低层则可以通过事件通知 L2CAP 入站连接、断开连接或者其他请求等。L2CAP 信道建立主要有:请求/响应、配置、控制信道 DLCI0 建立等三个阶段。其中 DLCI0 控制信道的建立是为建立用户 DLCI 链路作准备。

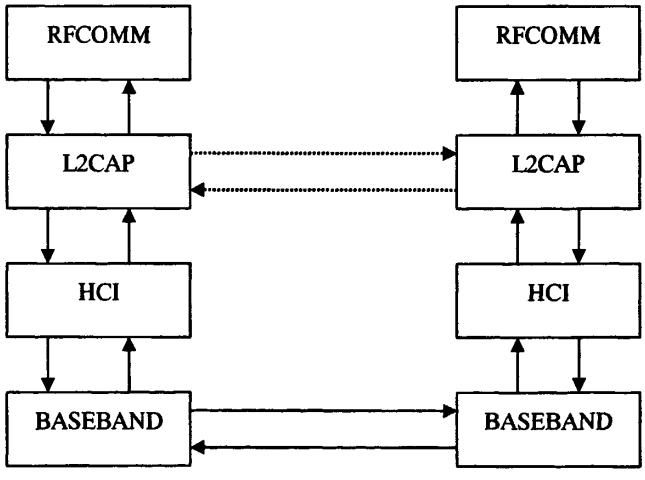

图 3.16　L2CAP 层间互操作

类似于 TCP 状态机，L2CAP 有 7 种运行状态:CLOSED、W4_L2CA_CONNECT_REQ、W4_L2CAP_CONNECT_RSP、CONFIG、OPEN、

W4_L2CAP_DISCONNECT_REQ 和 W4_L2CA_DISCONNECT_RSP 等。

L2CAP 信道的建立

L2CAP 采用信令动作 L2CAP_ConnectReq 和 L2CAP_ConnectRsP 建立 L2CAP 信道。上层使用 L2CAP 信道发送数据以及其他信息。用于建立信道的 HCI 指令和事件主要有命令：

HCI_Create_Connection(OCF:0x0005),HCI_Accept_Connection_request (OCF:Ox0009), HCI_Reject_Connection_Request(OCF:0x000a), 事件：Connecion Complete(事件码：0x03)；Connection Request Event(事件码：0x04)。

BlueZ l2cap.c 文件分析

12cap.h 中首先定义了信道配置选项，即默认的 MTU 为 672byte，刷新超时时间为 0xffff ms 和连接超时时间。接下来定义 L2CAP 协议地址族结构体 sockaddr_l2，信道配置选项结构体 12cap_options，连接信息结构体 12cap_coninfo，信道服务质量结构体 l2cap_qos。然后定义了 L2CAP 分组头结构体，信令指令头结构体以及各种信令指令相关结构体和宏。L2CAP 连接结构如下:

```
struct l2cap_conn{
bdaddr_t *dst; // 目的地址
bdaddr_t *src; // 源 地 址
…
};
```

这个结构体定义了 L2CAP 连接涉及到的各种信息。

在 l2cap.c 文件中实现了 L2CAP 层的套接字相关操作函数;L2CAP 逻辑信道相关的操作函数;连接相关操作，如建立连接，删除连接等;L2CAP 信令指令的函数实现:L2CAP 与 HCI 层的接口函数;另外还加入了对 proc 文件系统和定时器操作的支持。

```
struct proto_ops l2cap_sock_ops //l2caP 协议套接字操作结构体
static struct net_proto_family l2cap_sock_family_ops={
    family: PF_BLUETOOTH,
    create: l2cap_sock_create,
};
```

l2cap_sock_family_ops 结构体仅支持 create 操作。

```
static struct hci_proto l2cap_hci_proto={
    …
};
```

上面的结构体定义了 L2CAP 与 HCI 层的交互，以 L2CAP 为中心，L2CAP

层接收高层或底层的信息称为事件:L2CAP 层发起的信息称为行为。在这个结构体中，定义了 L2CAP 层与 HCI 层之间交互的来自 HCI 的连接指示和确认函数;断开链路指示;接收 HCI 层的 ACL 数据分组;以及来自 HCI 的鉴权加密确认信息。int __init l2cap_init(void)是 L2CAP 协议的初始化函数，当不采用可安装模块化方式时，就要给 init_module()函数换个名字，并且在前面加上__init 表示这个函数是初始化函数。同时还要在本文件中加上 module_init(12cap_init)表示在系统初始化时调用该初始化函数。这种处理方式在 HCI、RFCOMM 等协议层都有采用。从这种处理方式可见即使不把它编译成可安装模块，但是代码设计与实现仍是高度模块化的。在各协议层套接字初始化时都会调用 bluez_sock_register(),然后调用 hci_register_proto(&l2cap_hci_proto)注册 HCI 协议。

第四章 局域网移动接入应用

4.1 蓝牙局域网接入应用简介

蓝牙网路接入是在便携终端和移动电话或蓝牙网关之间仿真一条 EIA/TIA 232 线路之后，建立拨号上网连接。其基本操作为，在数据终端第一次用网关业务之前，两个设备要先进行初始化，具体包括交换 PIN 码、创建链路密钥、完成业务发现等。在发起或接收某一呼叫之前要先建立一条链路，这就需要寻呼另一台设备。由数据终端负责链路建立的发起。具体发起工作为：网关和数据终端利用串口应用规范提供串口仿真接口。串口仿真接口用于传输用户数据、调制解调控制信号、以及以数据终端到网关的 AT 指令集。网关解析 AT 指令集，将响应送回数据终端。SCO 链路用于数据传输，也可服务于呼叫建立时的音频反馈信号。

在局域网接入应用规范中描述了设备怎样使用蓝牙无线技术来访问因特网所提供的服务，接入规范是基于点对点协议（PPP）的，同时也定义了两个使用蓝牙无线技术的设备终端如何使用 PPP 机制互联。蓝牙局域网接入应用规范，如图 4.1 所示。

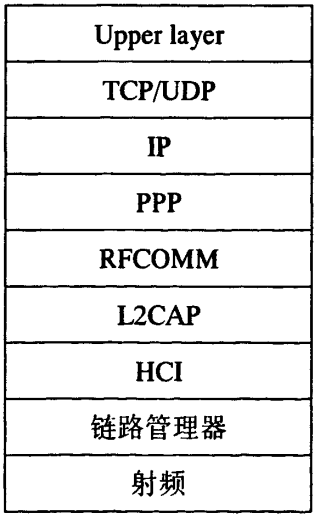

图 4.1 蓝牙局域网接入应用规范

这类应用中，用户可以在通信期间在某一个网关覆盖范围内四处走动，而且能够很好的享受各种网络业务。然而当用户在两个网关所覆盖的范围之间移动时，由于从进入了一个新的覆盖范围，在第一次使用新网关业务之前，就需要发起并建立一条新的链路。此时这种重新建立链路的过程对不同的业务有着不同程度的影响。对一般的数据业务，对时间要求不苛刻的情况下，对服务质量的影响可以忽略不计，但对于语言、视频等一些实时性较高的业务来说，影响是相当严重的。下面将对这种接入方式在网间移动状态下，对像语音这样的实时业务的影响做一些详细的分析，然后试图通过一种较新的接入方式来改进实时业务在这种状态下的服务质量。

4.1.1 网间移动接入对语音实时业务的影响

下面通过一个结构图，简单描述下蓝牙在无线局域网接入，结构如下图所示。

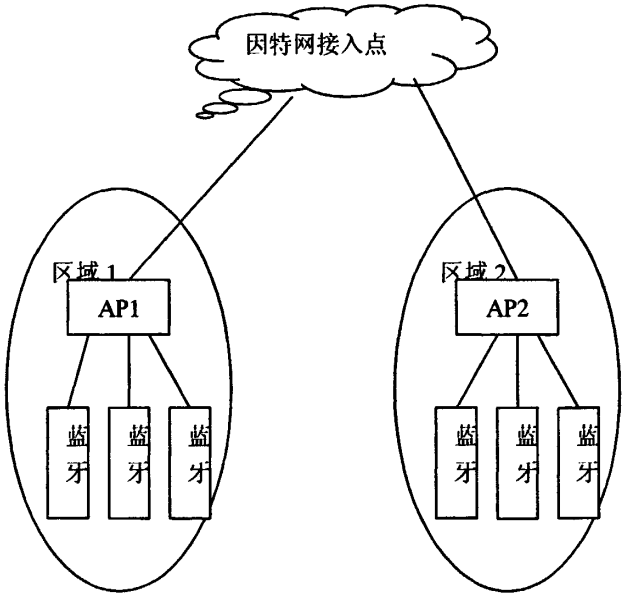

图 4.2 蓝牙无线局域网接入结构

在接入点（Access Point, AP）的覆盖范围内，不同的蓝牙在特定的时间片里通过特定的通道与 AP 建立起无线连接，从而取得因特网的服务。如果蓝牙终端在固定的位置接入网络，蓝牙终端与 AP 之间的连接是固定的，这种网络接入方法是没有任何问题的。但是在移动接入应用的情况下，由于蓝牙技术和 PPP 协议的一些固有特点使得这种接入存在了一些缺陷。

蓝牙是一种短距离无线通信技术，其通信距离一般是几十到一百米，但是受

设备功率及相邻区域的蓝牙 AP 的影响，有效距离只有几十米，这里假设 AP 覆盖范围的直径为 30m。设想这样一种应用场景，用户通过蓝牙终端使用 VOIP 业务与远程用户通话，而用户习惯于边走动边打电话，假设移动的速度为 2km/h，也就是说用户会在 54s 内就会在区域一和区域二之间漫游一次。那么漫游时间是由那些因素决定的呢？

蓝牙终端建立 PPP/RFCOMM/L2CAP 连接后，就会选择合适的 PPP 鉴权机制，例如质询握手鉴权协议（CHAP）。鉴权通过后，局域网接入点和数据终端之间会协商确定一个合适的 IP 地址，IP 数据从而能够流经 PPP 连接。当终端在不同的区域之间漫游时，PPP 连接都会断开，然后重新建立起 PPP/RFCOMM/L2CAP 连接，建立 RFCOMM 和 L2CAP 连接的时间相对于建立 PPP 连接的时间可以忽略，因此这里将漫游时间近似为建立 PPP 连接的时间。当我们使用 PPP 拨号上网时，拨号的时间是明显能够感觉到的，也就是对于这种移动应用场景来说每隔不到一分钟的时间就要建立起一次 PPP 连接，这对于大多数用户都是无法容忍的。

这里针对移动蓝牙无线接入应用，将介绍一种方法来实现 IP over Bluetooth，来代替传统的无线局域网接入，来改善实时业务的质量。

4.2 改进型蓝牙无线接入

下面先介绍一下改进型蓝牙移动无线网络的结构，如图 4.3 所示。

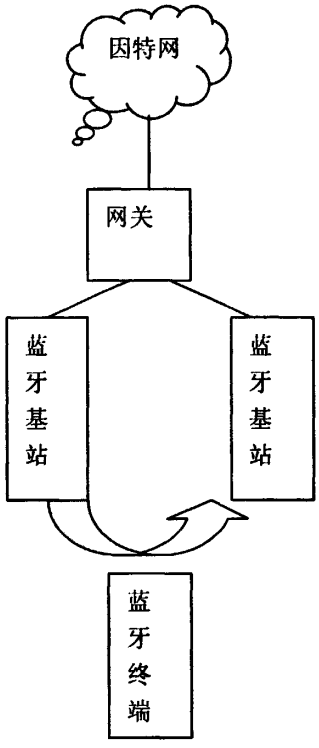

图 4.3 改进型蓝牙移动无线网络结构图

单纯从结构上来看，图 4.3 和图 4.1 似乎没有什么区别，但是网关和蓝牙基站在网络里的功能发生了变化，这里网关和蓝牙基站的功能相似，都起到网络地址转换（NAT）的作用，只不过网关是将内网地址转换为外网地址，而蓝牙基站则是将蓝牙地址转换为网路 IP 地址。这就需要在 IP 和 L2CAP 层之间引入一个中间件来实现蓝牙地址和 IP 地址转换的功能，其结构图如图 4.4 所示。

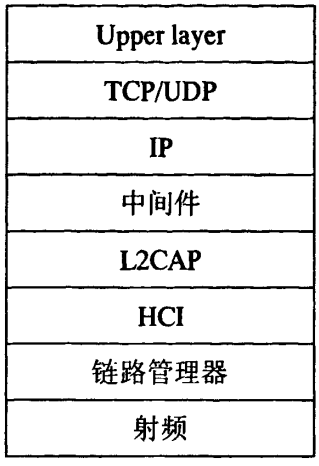

图 4.4 改进型蓝牙局域网接入协议栈

4.2.1 中间件的程序设计实现

由于中间件中涉及一些网络系统的数据结构，下面先对 Linux 网络系统做一些简单的介绍。Linux 网路系统遵守一个四层的模型概念，从上到下依次为应用层、传输层、互联层和网络接口层。其中网络接口层负责数据帧的发送和接收。互联协议将数据包封装成 Internet 数据报，并运行响应的路由算法。Linux 支持多种类型的套接字地址簇或域，其中包括 UNIX 域套接字、TCP/IP 协议的 Internet 地址簇等。所以在下面中间件程序中，IP 报头的封装及数据的转发就可以通过内核维护的一些数据结构和函数来完成。

中间件从功能主要分为三部分，包括向 IP 层提供的统一接口部分，L2CAP 层的接口以及如何通过 HCI 接口实现数据的传输与存储。

1、IP 层接口

在 Linux 内核中定义了一个 net_device 的数据结构，用来处理与网络设备相关的函数和参数。其主要定义如下：

```
struct net_device
{

    char        name[IFNAMSIZ];
    ......
    ......
    .......
```

```
        /* Pointers to interface service routines.    */
    int             (*open)(struct net_device *dev);
    int             (*stop)(struct net_device *dev);
#define HAVE_NETDEV_POLL
    int             (*hard_start_transmit)(struct sk_buff *skb,  struct net_device
*dev);
    ......
    ......
};
```

```
    char            name[IFNAMSIZ];
```

设备名称，这里定义为 bluetoothnet%d，当使用 ifconfig Bluetooth0 up/down 时就会对这个蓝牙设备进行操作。

```
    int             (*open)(struct net_device *dev);
```

这个函数会打开接口，当在终端中键入 ifconfig 时，它会向系统注册驱动所需的资源（例如：I/O 口，IRQ 等），打开硬件，使模块计数器加一。

```
    int             (*stop)(struct net_device *dev);
```

这个函数会关闭接口，所做的操作与 open 函数正好相反。

```
    int             (*hard_start_transmit)(struct sk_buff *skb,  struct net_device
*dev);
```

这个函数负责将存储在 sk_buff 中的数据传送给硬件。它是通过调用底层 L2CAP 的一些函数，最终通过 HCI 接口控制蓝牙控制器来实现的。

2、与 L2CAP 的接口

像上面所说，蓝牙中间件想要控制设备实现蓝牙两个终端之间的通信最终都是要调用 L2CAP 的一些函数，下面将介绍如何通过 L2CAP 建立蓝牙之间的连接及数据传输。

```
static ether_to_l2cap ether_operate={
ether_connect_ind,
ether_config_ind,
ether_disconnect_ind,
ether_connect_rsp,
ether_connect_cfm,
ether_config_cfm,
```

ether_disconnect_cfm,

ether_receive_data

}

这个结构体中定义了一些在蓝牙设备之间建立连接和传输数据所必须的函数。下图清楚的描述了两个设备之间进行连接、交换数据以及断开连接的过程。

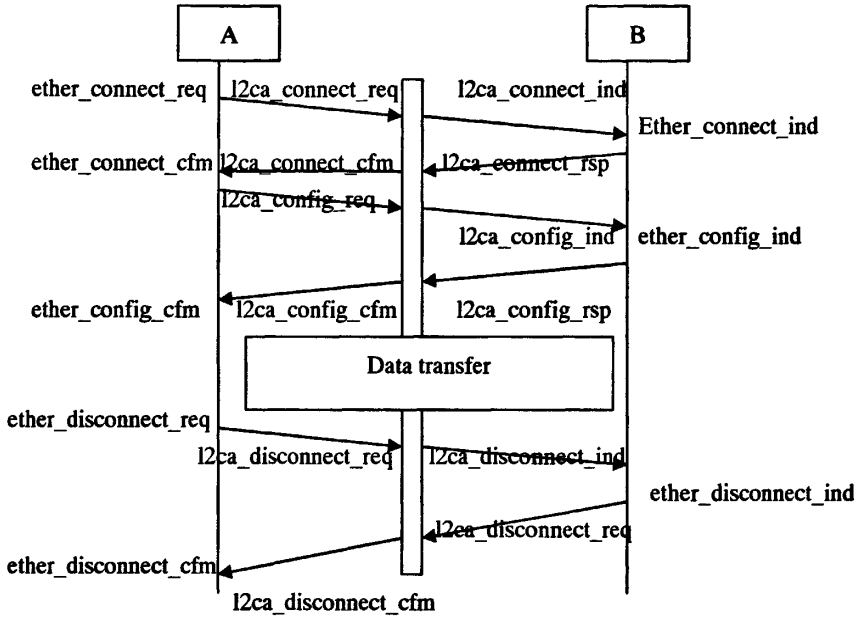

图 4.5 中间件功能示意图

为了更好的描述连接过程，这里假设发动连接者为 A，远端连接对象为 B，当 A 需要与 B 建立连接时，A 会调用 ether_connect_req(unsigned char *bt_addr) 函数，其中 bt_addr 为 B 的蓝牙硬件地址。这个函数会条用 L2CAP 协议层的 l2ca_connect_req 函数来实现具体的功能。在 B 端，l2ca_connect_ind 将会调用 ether_connect_ind（l2cap_con *l2cap）来通知 B 端中间件一个连接请求到来了。然后 l2ca_connect_rsp 将会被调用来回应这个连接请求。在连接建立之后，双方再通过 erther_connect_cfm、ether_config_ind、ether_config_cfm 来协商连接建立的一些参数，例如 MTU、连接超时等。当 A 端打算结束连接时，将会调用 ether_disconnect_req(l2cap_con *con)，它通过 l2ca_disconnect 通知 B 端 L2CAP 层，然后 B 通过 l2ca_disconnect_ind 调用 ether_disconnect_ind，最后 B 通过 l2ca_disconnect_rsp 回应 A 端的断开连接请求，然后 A 通过 ether_disconnect_cfm 断开连接。

3、数据传输函数与存储

Linux 网络层采用统一的缓冲区结构 sk_buff，每个单独的 sk_buff 被组织成双向链表的形式。网卡接收到数据帧后，系统内核把接收的数据帧放在 sk_buff 结构管理的缓冲区内。在网络协议处理的时候，数据均以 sk_buff 的形式在各层之间传递、处理。所以这里也用 sk_buff 来存储相关数据。

在中间件里面有两个主要函数来处理驱动和硬件之间的数据传输，输入函数和输出函数。当有数据从硬件输入时，将会产生一个中断，然后唤醒一个中断处理例程，这个例程通过 ether_receive_data(l2cap_con *l2cap,u8 *data,u32 len)来实现，它将所收到数据封装成一个 IP 包，送给 IP 层进行传输。当 IP 层有数据需要传送给蓝牙设备发送时，hard_start_transmit 将会被调用将数据送到一个输出队列。蓝牙硬件然后从队列中依次取出数据进行传送。

4.2.2 效果测试方法简介

在终端上安装上网络电话软件之后，这里在终端上移植并安装了 Kphone 网络电话软件，Kphone 是一个以 SIP 协议为基础，使用互联网连接提供即时消息、多方电话会议、拨打普通电话的 VoIP 客户端软件。允许免费内部相对打，还可以呼叫和接听有线和手机电话，具有友好的图像界面。

操作步骤简介如下：

1、首先需要一个服务帐户。

2、启动程序后、输入用户名和密码即可开始测试网络通话效果。

经过测试，在网间移动时，通话质量得到了很好的改进。

第五章 总结与展望

5.1 工作总结

课题研究的主要内容包括对嵌入式系统与蓝牙的核心技术进行了较为深入的研究和探索。主要包括 ARM 微处理器架构及 Bootloader 作用，嵌入式 Linux 内核移植、蓝牙协议栈等。课题完成基于 i.MX21 系列 MC9328MX21VH 微处理器的嵌入式硬件电路设计、一个通用的 Bootloader 的分析与移植、以及 Linux 内核在 MC9328MX21VH 目标板上的移植、并对蓝牙模块驱动及协议栈在 Linux 中的实现机理进行较为详细的分析等工作，然后，描述了蓝牙在网络接入中的应用场景，指出蓝牙在移动状态下接入局域网存在一定缺陷，例如，在网间移动情况下使用网络电话，由于网间漫游时延较大，严重影响通话质量。通过使用一种引用中间件的方法，改进网间移动情况下实时业务的性能。最后，由于效果测试工作由组内其他成员负责，文中仅对测试方法做简单介绍。

所做工作主要工作总结：

1、采用了一款基于 ARM926EJ-S 内核的 32 位 RISC 嵌入式微处理器 MC9328MX21VH 作为硬件核心，其主要面向高性价比、低功耗的应用，所设计的终端在节能方面具有一定的优势。

2、软件设计上采用了开源的高性能的嵌入式 linux 操作系统作为软件核心，充分利用了其开源优势，一定程度上减少了开发成本，而且其系统稳定性也增强嵌入式终端的可靠性。

3、通采用了基于蓝牙中间件的方法来减少蓝牙无线网络网间漫游时延对实时业务的影响。

但是，系统设计时也存在一定的缺点，例如，这种蓝牙接入模式需要部署专用的蓝牙网络，并采用专门的蓝牙网关，这给应用的推广也带来了一定的限制，但对于一些大型的集体单位，与通过使用网络电话节省的电话费用相比，部署网络的成本还是相对较小的，因此，也有一定的实际意义。

5.2 展望

在本文已经完成的基础上，更进一步的研究工作包括：

1、由于目前无线个域网中的无线技术应用增加，无线技术之间互相干扰也日益加重，需要进一步研究其解决方案。

2、对蓝牙应用进行深入研究，希望提出一些新的应用规范。

参考文献

[1]沈连丰等.嵌入式系统及其开发应用.北京:电子工业出版社，2005

[2]刘峥嵘等嵌入式 Linux 应用开发详解.北京:机械工业出版社，2004

[3]田泽.嵌入式系统开发与应用.北京:北京航空航天大学出版社，2005

[4]杜春雷.ARM 体系结构与编程.北京:清华大学出版社，2003

[5]陈莉君.深入分析 Linux 内核源代码.北京:人民邮电出版社，2002

[6]JONATHAN CORBET ALESSSANDRO RUBINI&JONATHAN CORBET.LINUX 设备驱动程序(第三版).北京:中国电力出版社，2006

[7]Nathan J.Muller 著 周正 等译.蓝牙揭秘.北京：人民邮电出版社，2001

[8]张禄林，雷春娟，郎晓虹，蓝牙协议及其实现，人民邮电出版社，2005

[9]金纯等，蓝牙技术，电子工业出版社，2001

[10] Bluetooth Specifications, http://www.bluetooth.com

[11]禹帆，蓝牙技术，清华大学出版社，2002

[12]i.MX21 Applications Processor Reference Manual Rev. 3, Freescale Semiconductor, 2007.4

[13]MC9328MX21 Rev. 3.2, Freescale Semiconductor, 2008.6

[14]吴明晖主编，徐睿等编著.基于 ARM 的嵌入式系统开发与应用[M].北京:人民邮电出版社，2004.6

[15]孙天泽,袁文菊、张海峰编著.嵌入式设计及 Linux 驱动开发指南——基于 ARM9 处理器[M]. 北京:电子工业出版社，2005.6

[16]张崴编著.32 位嵌入式系统硬件设计与调试[M].北京:机械工业出版社，2005.5。

[17] Jobn Catsoulis 著，徐君明，许铁军，黄年松等译.嵌入式硬件设计[M].中国电力出版社，2004．6。

[18] Xiao Ni, Weiren shi, Song Zheng, Design of Micro Mobility Support in Bluetooth Sensor Network, 2006 IEEE International Conference on Industrial Informatics

[19] [美]Andrew.N.sloss ,[美]Dominic Symes,[英]Chris Wright 著，沈建华译.ARM 嵌入式系统开发——软件设计与优化[M].北京:北京航空航天大学出版社，2005.5。

[20] Diego Melpignano, David Siorpaes, Bluetooth TCP Booster, Vehicular

Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd Volume 3, 6-9 May 2001 Page(s):2137 - 2141 vol.3

[21] Simon Baatz , Matthias Frank, Rolf Gopffarth, etc Handoff Support for Mobility with IP over Bluetooth, Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on 8-10 Nov. 2000 Page(s):143 – 154

[22] KO Sung-Yuan, The Embedded Bluetooth CCD Camera, Electrical and Electronic Technology, 2001. TENCON. Proceedings of IEEE Region 10 International Conference on Volume 1, 19-22 Aug. 2001 Page(s):81 - 84 vol.1

[23]马忠梅，李善平，康慨，叶楠。ARM & Linux 嵌入式系统教程[M] 北京航空航天大学出版社 2005.3

[24]卢军.Linux 0.01 内核分析与操作系统设计[M].清华大学出版社,2004.10

[25] 魏洪兴，胡亮，曲学楼 等。嵌入式系统设计与实例开发实验教材[M] 清华大学出版社 2005.12

[26]周立功.《ARM 嵌入式系统基础教程》[M].北京：北京航空航天大学出版社，2005.

[27] 金西、黄汪，Linux 操作系统是嵌入式系统新的选择，微计算机信息

[28] Andrew S. Tanebaum、Albert S. Woodhull，操作系统：设计与实现，清华大学出版社

[29] David A.Rusling 著、朱珂等译，Linux 编程白皮书，机械工业出版社，2000

[30] 邹思轶等，嵌入式 Linux 设计与应用，清华大学出版社，2002

[31] 王学龙，嵌入式 linux 系统设计与应用，清华大学出版社，2001.8

[32] 胡希明等，Linux 内核源代码分析，浙江大学出版社，2001

[33] 李善平等，linux 内核 2.4 版源代码分析大全，机械工业出版社，2002.1

致 谢

本学位论文是在我的导师白长清高级工程师的亲切关怀和悉心指导下完成的。从课题的选择到项目的最终完成，白老师都始终给予我细心的指导和不懈的支持。两年多来，白老师不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，白老师渊博的知识和严谨的治学态度、积极乐观的生活态度、艰苦朴素的作风、优秀的管理才能都给了我深远的影响，使我受益匪浅。在此谨向白老师致以诚挚的谢意和崇高的敬意。

此外，实验室的师兄师姐汪峰、叶爱萍、林萍、曲芸芸、罗士栋，还有同实验室的肖贺、刘芳、邓漫龄，以及寝室的同学，他们对于我的学习生活给予了很多的帮助，在这里也一并感谢！

特别要感谢我的家人，他们一直在我的身后默默的支持我，鼓励我，没有他们就没有我今天取得的成绩。

在此，对所有关心过我、帮助过我的人表示最衷心的谢意！

攻读硕士学位期间发表的学术论文

[1] 葛亮，基于 W78LE516B 的 PTR6000PA 驱动实现，中国科技论文在线，2008 年 6 月