# 摘 要

在移动通信迅猛发展的今天，移动终端已经成为各国通信技术发展的热点。移动终端软件主要包括底层系统软件、协议栈软件和 MMI 软件三部分。MMI 软件即人机接口(MMI)软件，主要负责人机的交互和功能的应用，是决定移动终端质量优劣的重要因素之一。本课题是某通信技术有限公司的研发内容之一，目标是开发一款具有较高效率的移动终端 MMI 软件。

在查阅大量相关资料的基础上，对移动终端软件进行了分析与研究，并以嵌入式软件设计方法学为指导，设计和实现了移动终端 MMI 软件。本文首先设计了 GSM 移动终端系统，包括硬件系统和软件系统。接着在此 GSM 移动终端系统之上，设计了一种通用性较强的移动终端 MMI 软件架构，按服务把它划分为框架层、用户接口层和应用层的三层次结构来设计与实现。最后在该架构上设计了电话本软件，包括姓名查找、电话添加、电话删除、电话复制和设置五个子功能。在电话本软件的设计与实现中，引入了三元搜索树(TST)技术，提高了电话本的运行效率。在程序实现上，以面向对象程序设计为思想，采用消息驱动机制，并选择标准 C 语言编写程序，进一步确保了软件的可移植性。

课题所设计的 MMI 软件在 GSM 移动终端系统上运行良好，并已经商业化。在设计中充分考虑到软件的独立性和可移植性，软件只需较小的修改，便可移植到 3G 通信网的各类移动终端中。

**关键词：移动终端　MMI 软件　三元搜索树　消息驱动机制**

# Abstract

With the rapid development of telecommunications, mobile terminals have become the hotspots of communication technology worldwide. The software of moblile terminals includes the underlying system, protocol stack and MMI software. MMI software (Man Machine Interface software), which is responsible for man-machine interaction and application, is one of the major factors influencing the quality of the mobile terminals. This subject is one of the researches of a communication technology corporation, with its target to develop MMI software with higher efficiency.

By referring to a great deal of the information available, a research on the software of mobile terminals has been done. In this paper, the GSM mobile terminal system is designed firstly, which includes the hardware and software system. Then the MMI Structure in the GSM mobile terminal system is designed and divided into three layers according to its services: the framework, the graphical user interface and the application layer. Finally, the phonebook software is designed, which includes the applications of find, add, delete, copy and set. During the design and implementation of the phonebook, the ternary search tree (TST) is used to improve the operating efficiency of the phonebook. In the program, the object-oriented thinking, message-driven mechanism and the standard C programming language are used to guarantee the portability of the software.

The MMI software designed in this paper goes well in GSM mobile terminals. Taking full account of its independence and portability, the software can be applied to 3G mobile terminals with a little variation.



**Keyword: Mobile terminals MMI software Ternary search tree Message-driven mechanism**

# 目 录

# 第一章 绪论

## 1.1 课题研究背景

二十世纪后期，随着大规模集成电路和计算机技术的迅猛发展，现代社会已经迈进了信息时代。在信息化的社会，人类对通信技术尤其是对移动通信的追求越来越高，移动通信得到飞速的发展。目前，移动通信已经成为各国通信技术发展的热点。

伴随着移动通信的发展，以移动通信技术和嵌入式技术相结合的数字化产品——移动终端也得以快速的发展，并成为数字化产品中的主流产品之一。世界各国在此领域开始了激烈的竞争，以争取获得主导地位。我国政府十分重视嵌入式系统产业的发展，并正在努力推进嵌入式系统技术在中国信息化建设中的应用，促进了嵌入式设备(包括软件和硬件)在中国实现产业化。

在过去的二十年里，通信产业成为了全球增长最为迅速的产业之一，有力地推动了全球经济的快速增长。其中，移动终端设备存在着一个很大的市场，且其发展十分迅猛。手机融合了移动通信技术和嵌入式技术，是移动终端设备的典型代表，也是目前世界上发展最为快速的通信产品之一，人们对它的需求量是日益倍增。

据有关数据统计显示，2007 年我国手机的销量突破 1.5 亿部，并预计 2008 年将继续呈现上涨趋势。随着我国 3G 牌照即将在 2008 年年底到 2009 年年初发放，未来几年，移动通信业将进入一个快速增长的时期，新的增长点将来自于无线数据业务。未来的移动终端将是一个移动的娱乐和事务处理工具，它把电话、PDA、数码相机、MP3/MP4 播放器等功能逐渐融合，这些都依赖于面向移动通信终端的嵌入式软件平台的研发。

1999 年智能手机产品开始在美国硅谷出现。智能手机是兼具手机和掌上电脑两种功能于一身的移动终端，可进行多方通话，并支持无线上网，还可与桌上电脑进行资料更新、交换。近年来，与智能手机相关的增值业务内容也开始丰富，产品的增多和应用环境的完善使智能手机市场呈现快速发展的态势。据瑞典市场研究公司 Berg Insight 最新发表的研究报告称，2007 年全球采用高级操作系统的智能手机销售量将达到 1.13 亿部。这家研究公司预测，智能手机销售量的年增长率将达到 25.6%，到 2012 年的销售量将达到 3.65 亿部，占全球手机市场份额的 22%。在我国，目前有高达 5 亿的手机用户，从 2007 年上半年手机整体市场情况来看，国内手机市场中的音乐手机和拍照手机依然是销售热点，这些具有多媒体功能的手机逐渐走向高端。同时，智能手机开始崭露头角，将在我国逐渐成为新

的增长热点。

随着 3G 牌照发放的日益临近，通信领域的竞争将更加激烈。各大通信设备提供商纷纷加大对移动终端产品的研发力度，不惜人力，物力和财力投入 3G 移动终端产品的研发工作，比如国内的通信设备商中兴、华为和大唐等，许多通信设备商更是把移动终端作为其主流产品之一。

## 1.2 课题研究依据及意义

当前，我国移动通信终端产业发展形势良好，然而，我国在研发技术上，特别在核心技术上，要远远落后于国外通信设备公司。移动终端技术和功能的发展日新月异，国内厂商在技术、实力和规模等多方面则显得相对薄弱，再加上移动终端开发的多样性、复杂性，导致了大量的重复劳动。其结果往往是，国内厂商的开发落后于市场的发展变化，以导致错失商机和市场的情况时有发生。技术、市场、信息等方面的交流匮乏，也局限了国内移动终端厂商的发展。

另一方面，移动终端平台、技术的多样性，也直接使得不同品牌终端之间数据、文件不能进行自由交换，这已逐渐成为厂商和用户共同关注的焦点，如不能有效解决，必将极大阻碍移动终端市场的发展。

智能手机产业的高速发展和我国 3G 通信技术[1]即将产业化为手机产业提供了广阔的市场，同时我国手机软件本土化优势强，结合国际化合作，将会给我国手机市场带来巨大商机。在庞大的国内手机市场中，国产手机发展迅猛，所占市场份额逐年上升。

以国产手机平台软件为基础，垂直纵向发展核心技术、横向联合应用软件的发展策略已经成为我国智能手机产业界的共识。国产软件开发商、手机制造商、移动运营商、配套资源提供商已经构成自主核心技术的国产手机支撑体系。强大的立体产业生态结构将确保我国在下一代移动通信产品中具备核心竞争力，使国产移动终端产业立于不败之地。而且，在国家 863 项目的引领下，我国各大高校和科研单位，如中科院、清华大学、浙江大学、西安电子科技大学、北京邮电大学等都在进行这方面的研究工作。现状要求我们在已有的基础上，不断学习创新，研发出高质量的核心技术。

因而，移动终端软件开发技术已成为当前一个重要的课题。研发出具有高性能、实用型的移动终端软件，必将推动我国通信产业的发展，提高我国在通信领域的竞争力。课题对于移动终端软件进行深入的分析和研究，并对移动终端 MMI 软件的设计与实现，将会对该领域多提供了一个参考的作用。对移动终端软件的研究，具有一定的实际应用价值。

## 1.3 课题研究内容

移动终端软件主要包括底层系统软件、协议栈软件和应用层软件三部分，论文将着重讨论应用层软件的开发。应用层软件即人机接口(Man Machine Interface，MMI)软件，主要负责人机的交互和功能的应用，是决定手机质量优劣的重要因素之一。本课题是某通信技术有限公司的研发内容之一，任务是开发一款具有较高效率的移动终端 MMI 软件。

在查阅大量相关资料的基础上，对移动终端软件进行了分析与研究，并以嵌入式软件设计方法学为指导，设计和实现了移动终端 MMI 软件。课题首先设计了 GSM 移动终端系统，包括硬件系统和软件系统。在此 GSM 移动终端系统之上，设计了一种通用性较强的移动终端 MMI 软件架构，按服务把它划分为框架层、用户接口层和应用层的三层次结构来设计与实现。最后在该架构上设计了电话本软件，包括姓名查找、电话添加、电话删除、电话复制和设置五个子功能。在电话本软件的设计与实现中，引入了三元搜索树(TST)技术，提高了电话本的运行效率。另外，在 MMI 软件与协议栈之间设计了中间层 L4 层，确保了 MMI 软件的独立性和可移植性。课题所设计的 MMI 软件，已经有效应用于 GSM 通信网的移动终端上，并已经商业化。

本论文的结构安排如下：

第一章，绪论。主要对课题的研究背景进行了介绍，阐明了课题的研究依据和研究意义，以及课题的研究内容。

第二章，移动终端系统及相关技术概论。分别介绍了移动通信系统和嵌入式系统，接着介绍由这两种系统相结合发展起来的移动终端系统。

第三章，GSM 移动终端系统设计。设计了 GSM 移动终端硬件系统和软件系统，并对系统中各功能模块进行了必要的介绍。

第四章，MMI 软件架构设计与实现。分析了嵌入式软件架构设计方法，按服务把 MMI 软件分为框架层、用户接口层和应用层三个层次进行设计与实现。

第五章，电话本软件设计与实现。对电话本软件进行设计与实现，并引入三元搜索树技术，提高了电话本的运行效率。

第六章，结束语。总结论文主要工作，提出课题接下来的工作要点。

# 第二章 移动终端系统及相关技术概述

在过去的20年中，通讯产业是全球增长最为迅速的产业，成为推动全球经济增长的领导力量。移动终端是目前世界上发展最为快速的通讯类产品之一，它的发展集中体现了嵌入式技术在通讯行业的应用状况。移动终端融合了嵌入式通讯、嵌入式操作系统、嵌入式数据管理系统与无线网络等技术，成为信息技术时代产品的典型代表。

## 2.1 移动通信系统

目前，世界的移动通信业蓬勃发展。我国正处于第二代移动通信系统向第三代移动通信网的过渡阶段，在2008年年底到2009年年初期间国家将颁发3G牌照，也即表明我国即将进入3G时代；而欧美和日韩等发达国家现已进入3G时代，享受着移动通信给生活带来的无限便利。

### 2.1.1 移动通信发展概论

最早的移动通信系统[2]是模拟移动通信系统，到上个世纪70年代，这种系统所能容纳的用户数目还很低，于上世纪80年代初投入商用的AMPS(Advanced Mobile Phone Service)系统的出现标志着第一代蜂窝移动通信系统(1G)的诞生。

针对第一代模拟移动通信系统具有的众多缺点，欧洲电信管理部门会议CEPT于1982年成立了移动特别小组(Group Special Mobile, GSM)，开始制定一种数字移动通信系统的技术规范，1987最终确定了系统标准，并将标准命名为GSM，标志着第二代移动通信系统(2G)的诞生。由于GSM标准的开放性，世界上许多著名的通信公司都在制作和提供GSM系统设备，因而系统获得了广泛的应用。对第二代移动通信系统GSM及IS-95CDMA来说，其主要业务是话音，并辅以低速数据，速率为14.4kbit/s。随着移动数据业务要求的增加，近年提出所谓2.5代的产品，提供速率为100kbit/s左右的数据业务。

近年来，第三代移动通信也得到了快速的发展，随着我国3G牌照即将发放，3G技术更成为人们所关注的热点。而我国提出的TD-SCDMA[3]无线传输技术现已作为中国提出的第三代移动通信标准，被国际电信联盟接纳为第三代移动通信标准之一，与欧洲支持的WCDMA[4]和美国的CDMA2000，成为全球第三代移动通信标准的三个最主要的竞争者。从GSM向3G的演进路径是通过GPRS，然后至EDGE，最终达到宽带CDMA(WCDMA)。从CDMA向3G的演进路径是通过称为"1X"的CDMA2000-1X的，然后至1xEV，而最终达到CDMA2000。CDMA2000

是美国向 ITU 提出的第三代移动通信空中接口标准的建议，是 IS-95 标准向第三代演进的技术体制方案，这是一种宽带 CDMA 技术。CDMA2000 室内最高数据速率为 2Mbit/s 以上，步行环境时为 384kbit/s，车载环境时为 144kbit/s 以上。CDMA2000 可采用多载波方式，能支持多种射频带宽，即射频带宽可为 NX1.25MHz，其中 N=1、3、5、9 或 12。目前技术仅支持前两种，即 1.25MHz (CDMA2000-1X)和 3.75MHz(CDMA2000-3X)。

2007 年底的世界无线电大会确定了 IMT-Advanced 的使用频谱，ITU 的 IMT-Advanced 技术征集函也已在 2008 年初发出，4G 也逐步进入行业，即将成为新的发展趋势。目前，较为热门的技术是 LTE(Long Term Evolution)，它是 3GPP(3rd Generation Partnership Project)近两年来启动的最大新技术研发项目，可以被看成是“准 4G”技术。它改进并增强了 3G 的空中接入技术，采用正交频分复用技术和多路进多路出(MIMO)技术作为其无线网络演进的基础性技术。当前，全球的移动通信产业对 LTE 寄予了厚望，期望这一技术能够增强通信产业持续发展的能力。

本项目是在 GSM 中测试实现的，由于软件的 MMI 层与协议层具有独立性，因此，只需要较小的接口改动，便可以方便的移植到 3G 的移动终端中。下面对 GSM 协议进行介绍。

### 2.1.2 GSM 协议体系结构

GSM 全名为 Global System for Mobile Communications，中文名称为全球移动通讯系统，俗称“全球通”。它是一种起源于欧洲的移动通信技术标准，是第二代移动通信技术，其开发目的是让全球各地可以共同使用一个移动电话网络标准，让用户使用一部手机就能行遍全球。

GSM 中采用了 OSI 的分层协议结构[5]。其中下一层协议为上一层协议提供服务，上一层协议利用下一层所提供的功能，上下层之间通过原语进行通信。在建立连接之后，对等层之间形成逻辑上的通路。在信令系统(MS)中，可简单分为物理层(L1)、链路层(L2)和网络层(L3)。GSM 协议体系的分层模型如图 2.1 所示。

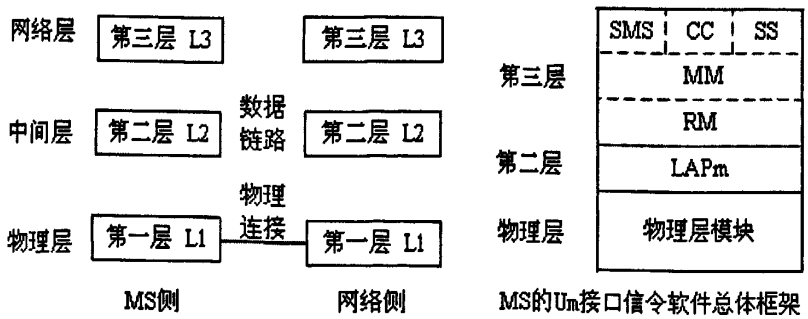


图 2.1 GSM 协议结构分层模型

由结构图可以看出，处于最底层，也即第一层为物理层，又称为L1层[6]。该层包含各类信道，为高层信道的传输提供基本的逻辑信道。每个信道都有自己的服务接入点(SAP)，移动台的接入方式采用多址接入方式，可以在空闲时间检测周围的无线电环境，把检测结果定时传给基站，确定是否进行小区切换。

第二层为链路层，也称 L2 层。链路层使用调制解调器链路存取协议 (Link Access Protocol for modem, LAPm)，它属于中间层，该层包括各类数据传输结构，对数据传输进行控制。LAPm 的基本功能是把单个 bit 构成一个集合，以便在移动台和基站之间提供可靠的无线数据链路，所有的链路功能都是建立在这个基本的结构单位上，该单位定义为帧(Frame)。帧的结构采用高层数据链路控制(HDLC)的定义方式[7][8]。

最顶层为网络层，又称为L3层[9]。包括无线资源管理子层、移动管理子层和呼叫控制子层。无线资源管理子层(RM)向公共控制信道(CCCH)和专用控制信道(DCCH)操作提供控制功能；移动管理子层(MM)提供移动台号码的保密、确认身份、位置登记、用户登记和取消、定时登记等功能；呼叫控制子层包括呼叫控制(CC)、短信息业务(SMS)和附加业务(SS)。呼叫控制包括呼叫建立、呼叫释放和呼叫相关的补充业务处理等；短信息业务利用信令信道为用户提供短信息的服务；附加业务是提供独立的附加业务，如短信的增值业务。

## 2.2 嵌入式系统

移动终端产品属于嵌入式产品系统应用开发的范畴。嵌入式概念早在20世纪60年代末就已经被提出，经历了三十多年的发展，已经到达成熟阶段。目前，随着IT向数字化、网络化和智能化的方向发展，嵌入式技术的应用得到全面的展开，已经在国防、国民经济及社会生活各个领域普及应用，用于企业、军队、办公室、实验室及个人家庭等各种场所。比如，现在流行的个人数字助理 PDA、机顶盒STB、IP电话等等，都属于嵌入式系统。所谓的嵌入式系统[10]，是以应用为中心，以计算机技术为基础，软、硬件可裁剪，适应应用系统对功能、可靠性、成本、体积及功耗严格要求的专用计算机系统。

### 2.2.1 嵌入式系统发展概述

嵌入式系统技术的发展[11]，大致经历了四个阶段：

第一阶段是以单芯片为核心的可编程控制器形式的系统，同时具有与监测、伺服及指示设备相配合的功能。这种系统大部分应用在一些专业性极强的工业控制系统中，一般没有操作系统的支持，通过汇编语言编程对系统进行直接控制，运行结束后清除内存。这一阶段系统的主要特点是：系统结构和功能都相对单一，

针对性强，处理效率较低，以前在国内工业领域应用较为普遍，但是已经不能适应高效的、需要大容量存储介质的现代工业控制和新兴的互联网信息设备等领域的需求。

第二阶段是以嵌入式 CPU 为基础、以简单监控式操作系统或 BIOS 为核心的嵌入式系统。这一阶段系统的主要特点是：CPU 种类繁多，通用性比较弱；系统开销小，效率高；系统具有一定的兼容性和扩展性；应用软件较专业，用户界面不友好；操作系统主要用来控制系统负载以及监控应用程序运行。

第三阶段是以通用型嵌入式实时操作系统(Real-Time Operating System，RTOS)为标志的嵌入式系统。这一阶段系统的主要特点是：嵌入式系统能运行在各种不同类型的微处理器上，兼容性好；操作系统内核精简、效率高，并且具有高度的模块化和扩展性；具备文件和目录管理、设备支持、多任务管理；具有大量的应用程序接口(API)，开发应用程序简单；嵌入式应用软件丰富。这个阶段的嵌入式系统有别于一般计算机处理系统，一般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)等作为存储介质；任何嵌入式系统都是针对具体应用、为了完成某个特定功能的专门系统；而通用计算机系统可以做各种不同的事情，用户既可以用 PC 写小说，玩游戏，也可以编程序、处理公务等。

第四阶段是以基于 Internet 访问为标志的嵌入式系统，这是一个正在迅速发展的阶段。目前大多数嵌入式系统还孤立于 Internet 之外，但随着 Internet 的发展以及 Internet 技术与无线通信、工业控制等结合日益密切，嵌入式设备与 Internet 的结合将代表着嵌入式技术的真正未来。

### 2.2.2 嵌入式系统组成

一般来说，嵌入式系统通常由两部分组成：嵌入式硬件和嵌入式软件。嵌入式硬件包括嵌入式处理器和外围设备，而嵌入式软件包括嵌入式操作系统和应用软件。下面对各个部分进行介绍。

(1) 嵌入式处理器

嵌入式处理器是嵌入式的核心部分。嵌入式处理器与通用处理器的最大不同在于其大多工作在为特定用户群设计的系统中。它通常把通用计算机中许多由板卡完成的任务集成在芯片内部，从而有利于嵌入式系统设计趋于小型化，并且具有高效率、高可靠性等特征。

现在，市场上的嵌入式处理器品牌较多，而嵌入式处理器芯片的类型数目可高达上千种。其中广泛使用的处理器有 ARM、MILPS、MC、PowerPC 等。项目中所使用的处理器为 ARM7。

(2) 外围设备

嵌入式外围设备是指在一个嵌入式系统中，除了嵌入式处理器以外用于完成存储、通信、显示等辅助功能的其他设备。根据其功能，可以分为三类：存储器，如静态易失性存储器 RAM/SRAM，动态存储器 DRAM 和非易失性存储器 Flash 等；接口，如 RS232 串口、USB 通用串行总线接口等；人机交互，如 LCD、键盘、触摸屏等。

(3) 嵌入式操作系统

嵌入式操作系统为嵌入式系统提供了处理器管理、存储器管理、设备管理和文件管理等功能，使嵌入式系统开发更方便、快捷。嵌入式操作系统大大提高了嵌入式系统的功能，方便了应用软件的设计，但同时也占用了宝贵的嵌入式系统资源。因此，嵌入式操作系统的引入，一般也是裁剪后的引入。

国外嵌入式操作系统已经从简单走向成熟，有代表性的操作系统有：美国 WindRiver 公司的 VxWorks，微软公司的 WindowsCE，还有μC/OS(μC/OS-Ⅱ)、Linux 和 Nucleus 等。项目中所使用的嵌入式操作系统为 Nucleus。

(4) 应用软件

嵌入式系统的应用软件是针对特定的实际专业领域，基于相应的嵌入式硬件平台，并能完成用户预期任务的计算机软件。应用软件是实现嵌入式系统功能的关键，其具有的特点有：软件要求固态化存储；软件代码要求高质量、高可靠性；所运行的嵌入式系统必须具备多任务实时性。项目中所设计的 MMI 软件便属于应用软件。

### 2.2.3 嵌入式系统的特点

嵌入式系统具有以下几个特点[12]：

(1) 微内核：由于嵌入式系统一般是应用于小型电子装置，系统资源相对有限，所以内核较之传统要小得多。比如 ENEA 公司的 OSE 分布式系统，内核只有 5KB，而 Windows 的内核则要大好几个数量级。

(2) 高实时性：高实时性的操作系统软件是嵌入式软件的基本功能要求，而且软件要求固化存储，以提高速度；软件代码要求高质量和高可靠性。

(3) 多任务管理：嵌入式软件开发要想走向标准化就必须使用多任务的操作系统。简单的嵌入式系统的应用程序可以在没有操作系统的支持下直接在芯片上运行，但是功能复杂、实时性强和高可靠的嵌入式系统就要使用多任务的操作系统。为了合理地调度多任务，利用系统资源、系统函数以及专用库函数接口，用户必须使用 RTOS 开发平台，这样才能保证程序逻辑执行的实时性、可靠性，并减少开发时间，保障软件质量。项目所采用的 Nucleus 操作系统属于 RTOS，将

在 2.2.3 节对 RTOS 作进一步的介绍。

(4) 专门的开发工具和环境：嵌入式开发不同于桌面应用程序的开发，嵌入式系统本身不具备自主开发能力，一般都是通过交叉调试模式来开发，因此对于嵌入式开发工具也有特殊的要求，特别对调试器部分。有很多硬件平台的目标处理器提供特殊的调试模式，如 JTAG, BDM 等调试。因此，调试器必须提供对这些调试模式的支持。开发时往往有宿主机和目标机的概念，宿主机用于程序逻辑的开发，目标机作为最后的执行机，开发时需要交替结合进行。

(5) 多样化、广泛化：嵌入式应用领域已应用到社会的各个领域如信息家电、工业控制、通信和智能终端等。嵌入式应用的多样化主要体现在嵌入式设备主控芯片和外围设备的多样化，目前嵌入式设备的主控芯片类型包括四类：微控制器、嵌入式处理器、DSP 处理器和片上系统(SoC)。就嵌入式处理器而言有很多系列如 386ex、PowerPC、MPC 系列、MIPS 和 ARM 等；嵌入式外围设备种类繁多，而且不同的嵌入式应用有不同的外围设备，为了支持这些不同的外围设备就必须有这些不同设备的板级支持包(Board Support Package，BSP)。

(6) 软件重用性：嵌入式系统应用最早是基于单片机的简单控制应用，因为单片机上的资源有限，因此在程序上采用前后台的控制设计。但这种程序设计方式已经不能满足特定和复杂的嵌入式应用。现在很多嵌入式应用都引入了嵌入式操作系统，利用嵌入式操作系统提供的特有的机制来满足嵌入式一些特定的应用，如多任务运行、实时控制和可靠冗余处理。因此需要对嵌入式操作系统之上提供一个抽象层以提供给应用软件一致的接口，保证应用软件的可移植性。同时随着嵌入式硬件平台的多样化，也需要提供硬件抽象层以保证高层组件的可重用性。

### 2.2.4 实时操作系统体系结构

移动终端的嵌入式操作系统是实时操作系统，主要强调任务执行和切换的确定性。“任务”(Task)是 RTOS 调度的基本单位，类似于 Windows 的进程或线程。一个 RTOS 的体系结构[13]中包含以下成分：

(1) 硬件抽象层及特定代码

提供了基本的硬件平台管理，包含所有硬件平台相关的代码，如上下文切换、各种硬件的设备驱动程序和 I/O 寄存器访问等。它位于 RTOS 的最低层，直接访问和控制硬件，对其上层的 RTOS 机器无关代码提供访问和控制服务。这样简化了 RTOS 内核的移植工作，在移植的时候只需要修改硬件抽象层的代码就可以了。该部分是 RTOS 能快速提供多平台支持的关键。

(2) RTOS 内核

嵌入式系统通常存在任务并发需求，RTOS 内核是支持并发任务调度，提供

任务同步和通讯机制的主要工具。内核的结构和调度算法基本确定 RTOS 的实时性能。内核需要实现任务管理、定时器管理、中断异常管理、存储器管理、资源管理以及消息管理等功能。这些管理功能通过内核服务函数形式提供给用户调用，也就是 RTOS 的系统调用(或称 API)，从而使用户程序及上层 OS 组件对系统设备透明。

(3) 对用户程序提供的函数接口

以便专门为用户定制网络、图形、视频等接口，并且提供驱动程序开发界面，方便开发者对不同需求的设备定制驱动程序。

### 2.2.5 嵌入式系统开发过程

通用计算机具有完善的人机交互界面，加上一些开发环境即可进行自身的应用程序的开发。而嵌入式系统本身不具备自主开发能力，其中的软件通常是固化的，必须有一系列开发工具才能进行开发、调试与固化。不同的开发阶段与开发不同的嵌入式软件需要不同的工具。开发工具包括交叉编译器与链接器、源程序模拟器、仿真调试工具、下载固化工具等。在开发过程中，把运行嵌入式系统的设备称为目标机，嵌入式软件运行在目标机的嵌入式处理器上。而把辅助开发软件的通用计算机称为宿主机，宿主机可以是 PC 或工作站，宿主机上要有一套开发工具来辅助我们进行嵌入式软件的开发和调试，开发工具通常运行在 Windows、Unix 或 Linux 下。

嵌入式软件的开发、调试及固化过程[14]如下：

首先，在宿主机上用交叉编译器与链接器将源程序进行编译和链接。交叉编译器可以在宿主机上编译生成目标机的目标文件，并针对特定的目标机处理器进行优化。连接器将多个目标文件及库链接生成目标机处理器的二进制文件。可采用 RTOS 提供的开发平台。

然后，用源程序模拟器在宿主机上进行模拟调试。源程序模拟器是运行在宿主机上的一个程序，用软件来模拟目标机处理器的功能和指令集。模拟调试正常后，可以用仿真调试工具来进行进一步的调试。仿真调试工具包括在线仿真器(In Circuit Emulator，ICE)、ROM 仿真器和 BDM/JTAG 调试器等。

ICE 用于仿真目标机的处理器。使用时 ICE 通过串口或 LAN 与宿主机连接，同时将仿真头插在目标机处理器的插座位置。通过 ICE 可以实时监视和控制目标机处理器的各种活动和状态，能在总线级上给出目标机系统当前行为的清晰描绘并支持硬件断点和实时跟踪调试。ROM 仿真器用来仿真目标机的存储器。ROM 仿真器与目标机的连接和 ICE 类似，不过它通过存储芯片插座与目标机连接。ROM 仿真器可将目标机的程序执行情况显示到宿主机的显示器上，以及设断点、

单步执行等。BDM/JTAG 调试器是通过将宿主机与目标机直接相连接。要求目标机处理器支持 BDM 或 JTAG 接口，并且嵌入式软件要支持串口。

最后，用下载固化工具将调试通过的程序生成映像文件写入到目标机的 Flash 等非挥发性存储器中，在目标机上电时，程序便能自动运行。不同的非挥发性存储器有不同的下载固化方法，Flash 通常是在相应驱动程序配合下，在线下载写入；EPROM 通常插在专用编程器上，编程写入(烧入)。下载固化工具可以是 IDE 的一部分，也可以是单独的程序，下载固化工具包括定址工具，用户可根据自己需要安排地址空间。

## 2.3 移动终端系统

移动终端系统是移动通信系统和嵌入式系统两种技术的融合，因此，可定义移动终端系统[15]为采用一定的通信协议、具有与许多网络供应商设备的互操作性、自身具有无线通信等功能的复杂嵌入式系统。移动终端的类型有车载台、固定台和手机等。

项目所开发的是 GSM 移动终端，即 GSM 移动通信网中用户所使用的移动通信设备。GSM 移动终端除了提供通过无线接口进入 GSM 系统的常规无线通信功能和相关处理功能外，还必须提供与使用者之间的接口。比如提供通话呼叫所需要的话筒、扬声器、显示屏和键盘等；进行数据通信时还需要提供与其它一些终端之间的接口，如与计算机之间的接口。

GSM 移动终端的使用还必须用到客户识别模块(Subscriber Identity Model，SIM)卡，GSM 移动通信网通过 SIM 卡来识别用户，SIM 卡的应用使移动终端的应用不是固定的束缚于某一固定用户，同时也为网络运营商的管理带来了便捷。

### 2.3.1 移动终端软件的特点

移动终端软件属于嵌入式实时软件，它首先应在功能和性能上满足用户需求，其次，个性化的人机界面也是必须的。具体来说，应具有如下特点：

(1) 功能上，应满足相应移动通信的标准，如呼叫、短信等服务，还须完成诸如文本字符的编辑、音量调节等非规定性功能。

(2) 性能上，最主要的要求是软件的实时性、健壮性。实时性是以普通用户感受不到的操作延迟为标准。健壮性则尽量将软件的故障概率控制在一个很小的范围内。

(3) 人性化设计上，要求软件界面丰富、个性化强、易操作，人机界面的设计应以此为目标。

### 2.3.2 开发原则

根据上 2.3.1 节的功能描述，移动终端软件开发应遵循如下原则：

(1) 功能完备和软件的可扩展性：移动终端需要具有必备的业务和附加业务功能，而且软件功能可根据需要在功能和配置上进行扩展。

(2) 及时响应：这是对软件实时处理的要求，以普通用户能容忍的程度为限。

(3) 健壮性：软件模块应具有较强的安全机制和容错机制。

(4) 复用性：一种软件解决方案的确立、设计和完成需要投入巨大的人力，软件解决方案一般要移植到同一档次的多款机型上，因此软件可复用很重要。

(5) 易调试性：由于移动终端软件都比较庞大，而且采用的是交叉调试方法，对这样的嵌入式实时软件，调试的任务将非常艰巨。因此，良好的编码风格有助于程序的调试。另外，通用软件上成熟的调试技巧也可用于软件的调试。

(6) 界面人性化：人性化的界面将体现以人为本的设计理念。从技术实现上，可包扩图形界面、用户提示音等，将给用户以亲切感，从而增加商业卖点。

### 2.3.3 移动终端的发展趋势

随着通信产业的飞速发展，移动终端的概念也在逐渐发生变化。它的功能早已超出了移动电话本身的定义，其应用日益多样化，对整个系统的软硬件资源要求不断提高，除了话音通信功能外，它还具备数据通信和数据计算功能，这就要求移动终端数据处理能力要不断增强。因此，以后的智能终端上一般都采用双 CPU 结构，围绕这两个 CPU 形成移动智能终端中的两个子系统[16]：通信子系统和应用子系统。其中通信子系统适应各种无线接口协议标准，选择适当的移动通信网络，建立和维持网络连接，实现话音和数据通信。应用子系统负责管理存储器、外围设备、外部接口等系统资源，运行应用程序，提供用户界面，此外还包括终端的电源管理。如此强大而复杂的硬件资源势必需要系统化管理，因此单独的智能移动终端操作系统也就成为大势所趋。它主要完成诸如进程、内存、外部设备等系统资源的调度和管理，并为上层应用软件平台提供服务，在操作系统之上可以执行各种各样的应用程序。无线移动终端发展到了智能终端后就产生了类似 PC 机的明确的逻辑结构。

## 2.4 本章小结

移动终端系统的发展，是移动通信系统与嵌入式系统相结合的产物。移动通信系统经历几十年的迅猛发展，现在已经进入第三代移动通信的时代；与此同时，嵌入式技术也得到前所未有的发展，嵌入式产品广泛应用于我们生活的每一部分。

两者的结合，给通信产业注入了新的血液，它们的结合产物——移动终端，得到了广泛的应用与重视。人们对于产品的无止境追求，推动了移动终端系统的发展。现在，移动终端具备了多种多样的功能，为我们的生活提供了极大的便利。本章对移动终端系统及相关技术的探讨分析，为接下来的开发工作提供了强有力的技术支持。

# 第三章 GSM 移动终端系统设计

GSM 移动终端系统设计，是基于移动终端通信模块进行的整机周边电路设计以及软件开发。通信模块，就是集成了射频和基带元件的主板，它嵌入了GSM/GPRS 通信协议；周边电路设计，就是要在这块主板上设计显卡、声卡、键盘等设备的周边电路。软件设计与开发，主要是进行人机界面交互、高层数据协议软件处理等。GSM 基带、射频、协议处理则由通信模块完成。项目中所开发的移动终端系统支持电路交换和分组交换，可提供语音和数据服务，并支持双频接入，即 GSM900 和 DCS1800。下面分别从硬件系统和软件系统两方面对该项目进行系统设计。

## 3.1 硬件系统

一般来讲，移动终端的硬件系统组成包括射频模块、基带模块和电源管理模块。射频模块主要完成 GSM 协议的物理层功能，实现突发脉冲的发射、接收、调制、解调等功能。基带部分主要完成 GSM 链路层和网络层协议，对从射频发送过来的信号实现均衡、解密、去交织、信道解码、语音解码和 A/D 转化，最后把模拟信号发送到听筒；反之，对从话筒接收到的模拟信号实现 D/A 转化、语言编码、信道编码、交织、加密和正交同相，最后把信号发送给射频模块。电源管理模块主要负责电源管理和给系统提供各种稳定的电压。这三个部分相互配合，完成移动终端的各项功能。其基本电路框图如图 3.1 所示。

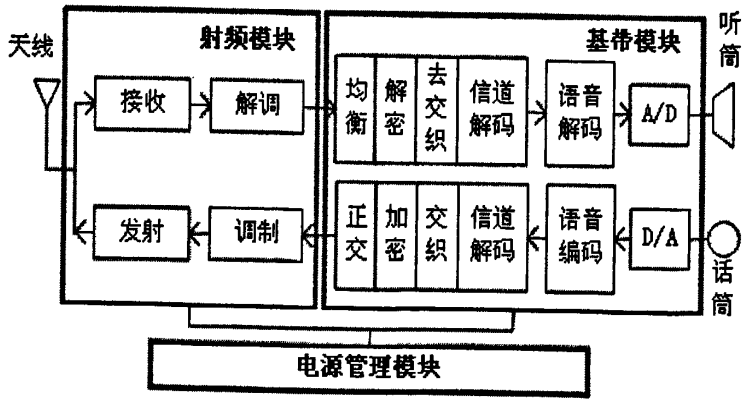


图 3.1 基本电路框图

在项目开发硬件架构设计中，射频(Radio Frequency，RF)模块由收发信机 MT6129、功率放大器和天线开关等组成；基带部分采用基带芯片模块 MT6219 GPRS Baseband，该模块主要由 ARM7、DSP 和一些 LCD、键盘、Flash 等外围设备组成；电源管理模块主要由电源管理 MT6305 芯片实现。整个系统的硬件架构

如图 3.2 所示。

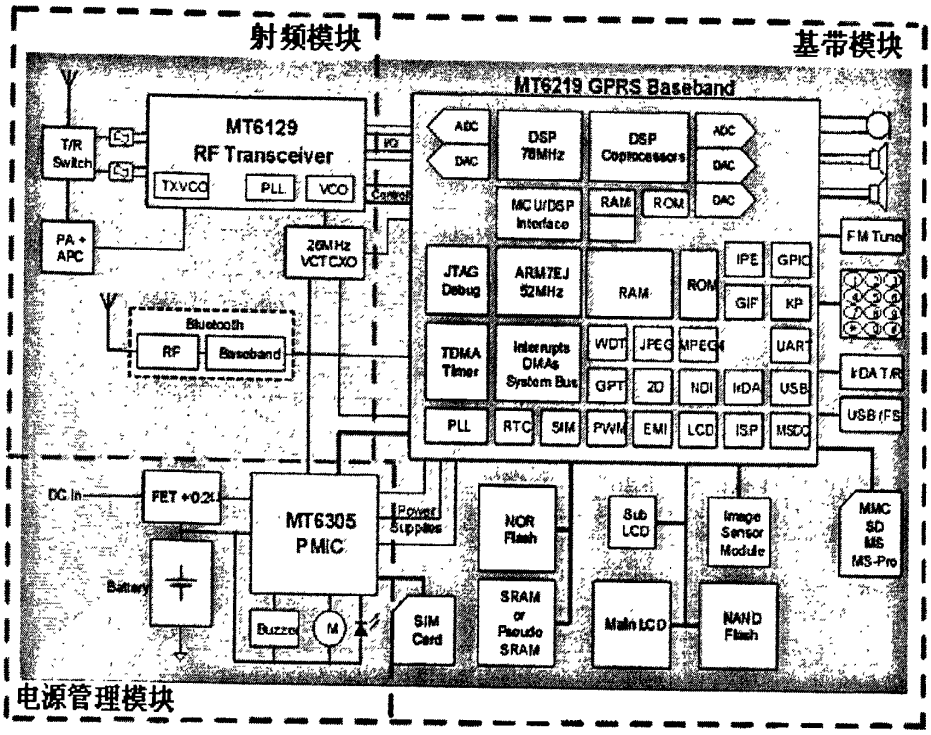


图 3.2 移动终端硬件构架

### 3.1.1 射频模块

项目中所使用的射频芯片是 MT6129 RF，属于 MT612x RF 系列。MT612x 是为 GSM 和 GPRS 的多频带全球系统而设计的具有很高集成度的收发集成芯片。MT612x 包括四个低噪声放大器、两个 RF 积分合成器、一个集成的信道滤波器，一个可编程的增益放大器、一个接收机的 IQ 解调器、两个内部发射信号压控振荡器(TX VCO)、一个片上可调整的压控晶体振荡器(VCXO)和一个完全可编程的片上射频压控振荡器(RF VCO)合成器。MT612x 也包括控制电路来支持不同的操作模块。

(1) 接收机：MT612x 的接收机包括四频段的微分输入低噪声放大器、RF 积分合成器、片上信道滤波器、可编程的增益放大器、正交混频器、低通滤波器和四个差分低噪声放大器。MT612x 使用正交混频器和滤波器来消除干扰；使用 RF 积分合成器和匹配技术，MT612x 的镜像抑制在所有的频段都能达到 35dB。同直接合成的接收机相比，MT612x 的中频(IF)结构能够提供块抑制、调幅抑制和邻道干扰的抵消。此外，IF 结构消除了在直接合成的接收机中精确的直流缓冲标准。而且，和声表面滤波器中输入的普通模式平衡需求是相关的。MT612x 不需要任何其他的频率合成成分就能提供模拟的 IQ 基带输出。

接收机的四个差分低噪声放大器各用于 GSM850(869MHz-893MHz)、

EGSM(925MHz-960MHz)、DCS1800(1805MHz-1880MHz)和 PCS1900(1930MHz-1990MHz)四个频段的处理。差分的输入通过 LC 电路和 200Ω的声表面滤波器相匹配。MT612x 的接收机特性如下：

- 具有非常低频率的 IF 结构；
- 四频段的微分输入低噪声放大器；
- 积分射频合成器；
- 高度集成的信道滤波；
- 大于 100dB 的增益；
- 大于 110dB 的可控范围。

(2) 发射机：MT612x 的发射机由两个片上的 TX VCO、缓冲放大器、下行链路的合成器、四项的调制器、一个模拟相位检测器和一个数字相频检测器组成，每一个部分有一个电荷泵电路输出和一个片上环路滤波。分路器和环路滤波器用来从下行链路合成器和四项调制器中得到理想的 IF 频率。对于一个给定的传输信道来讲，发射机要从两个不同的 TX 号码中选择一个。这些成分同内部的压控振荡器和环路滤波实现转化的环路调制。TX VCO 适用于功率放大器。外部实现的一个控制环用来控制功率放大器的输出功率值。其特性如下：

- 精确的 IQ 调制；
- 平移环结构；
- 高度集成的宽度 TX VCO；
- 高度集成的 TX 环滤波。

(3) 频率合成器：MT612x 包括一个含有高集成度 RF 压控振荡器的频率合成器，来产生信号接收端和发送端的本地振荡频率。锁相环把 RF 压控振荡器的频率锁定在 26MHz。为了减少部分-N 合成器所引起的影响，带有抖动功能的 3 阶 δ-σ调制来产生分频数 N。为了减少捕获时间或者说为了多时隙的数据服务(比如 GPRS)快速设定时间，在合成器中使用了带有快速捕获系统的数字环(振荡环)。一旦合成器设定后，射频压控振荡器通过数字校准环，把数字环设在理想频率的附近。校准后，在一定时期内，利用快速捕获系统来使锁定更准确。一旦捕获完成，锁相环就会回到通常的状态。其特性如下：

- 单集成，完全可编程的部分-N 合成器；
- 高度集成的宽带 RF VCO；
- 为多时隙 GPRS 应用快速准确的设置时间。

### 3.1.2 基带模块

MT6219 是双内核结构，不仅支持 GPRS Class12 modem，同时提供全面、先

进的多媒体解决方案。MT6219 集成了 ARM7EJ-S 内核和 DSP 内核。ARM7EJ-S 是主处理器，主要运行 GSM/GPRS 上层协议软件、多媒体应用、外设驱动程序等。DSP 处理底层的调制解调以及复杂的语音功能。除了混合信号电路，其他 MT6219 的内置电路和微控制器或数字信号处理器相联。MT6219 包括以下子系统：

(1) 微控制器子系统(MCU)：包括 ARM7EJ-S 处理器、内存管理单元和终端控制器；

(2) DSP 子系统：包括 DSP、内存、内存管理单元和中断控制器；

(3) MCU/DPS 接口：MCU 和 DSP 交换软硬件信息；

(4) 微控制器外设：所有的用户接口模块和 RF 接口模块；

(5) 微控制器的协处理器：代替微控制器执行计算密集的操作；

(6) DSP 外设：GSM/GPRS 信道编解码加速硬件；

(7) 多媒体子系统：集成了几个先进的硬件加速器来支持多媒体应用；

(8) 语音前端：语音的模数、数模转换数据通道；

(9) 基带前端：RF 模块的数模、模数转换数据通路；

(10) 时序发生器：产生和 TDMA 帧时序相关的控制信号；

(11) 电源、复位和时钟子系统：管理电源，复位和时钟分配。

基带模块中还包括了 LCD、键盘、Camera、Flash 等外围设备，其中，人机接口设备有键盘和液晶 LCD，用于输入、输出显示；Camera 模组提供照相和动画录像等多媒体功能；NOR Flash 芯片用于存储整个手机软件和用户数据；NAND Flash 用做 USB 存储器。

### 3.1.3 电源管理模块

MT6305 是为 GSM 手机所优化的电源管理芯片。它包含 7 个线性稳压器，每一个用于管理 GSM 子模块的电源，MT6305 的优化设计使电池具有最长的生命期，在待机状态仅仅有 107μA 的低电流，在工作状态也只有 187μA。MT6305 支持锂电和镍锌电池。MT6305 包含了 3 个开漏输出分别连接到 LED、警报器和振荡器。SIM 接口为 SIM 卡和微处理器提供了电压的水平转换。MT6305 有 48 个管脚，操作温度范围是-25 摄氏度到+85 摄氏度。其特性如下：

(1) 管理 GSM 基带芯片的所有电源；

(2) 2.8V 到 5.5V 的输入电压；

(3) 最高 15V 充电电压；

(4) 为 GSM 子系统所优化的 7 个线性稳压器；

(5) 高效率操作和低待机电流；

(7) 锂电和镍锌电充电功能；

(8) SMI 卡接口；

(9) 三个开漏输出控制 LED、警报器和振荡器；

(10) 高温过载保护；

(11) 高电压锁保护；

(12) 低电压保护；

(13) 复位和开启计时器；

(14) 48 管脚封装。

## 3.2 软件系统

移动终端的软件模块主要由 Nucleus 操作系统、GSM 协议栈、设备驱动程序、MMI 和 WAP 模块组成。整个系统软件的系统架构如图 3.3 所示。

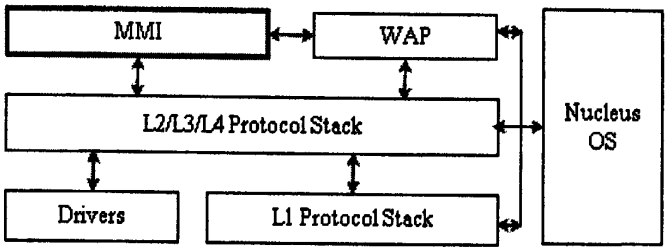


图 3.3 软件构架

图 3.3 中各个部分的具体含义如下：

(1) Nucleus OS：Nucleus 实时嵌入式操作系统；

(2) L1 Protocol Stack：GSM/GPRS 物理层协议；

(3) Drivers：系统外围设备的驱动程序，如 SIM、LCD、URAT、GPIO 等；

(4) L2/L3/L4 Protocol Stack：对应 GSM/GPRS 协议栈链路层/网络层；

(5) MMI：Man Machine Interface，即人机接口，主要负责人机交互；

(6) WAP：无线应用协议。

### 3.2.1 Nucleus 操作系统

Nucleus 嵌入式操作系统是目前最受欢迎的操作系统之一，Nucleus PLUS[17]是为实时嵌入式应用而设计的一个抢先式多任务操作系统内核，其 95%的代码是用 ANSI C 写成的，因此非常便于移植并能够支持大多数类型的处理器。从实现角度来看，Nucleus PLUS 是一组 C 函数库，应用程序代码与核心函数库连接在一起，生成一个目标代码，下载到目标板的 RAM 中或直接烧录到目标板的 ROM 中执行。在典型的目标环境中，Nucleus PLUS 核心代码区一般不超过 20K 字节大小。Nucleus PLUS 采用了软件组件的方法。每个组件具有单一而明确的目的，通常由几个 C 及汇编语言模块构成，提供

清晰的外部接口，对组件的引用就是通过这些接口完成的。除了少数一些特殊情况外，不允许从外部对组件内的全局进行访问。由于采用了软件组件的方法，Nucleus PLUS 各个组件非常易于替换和复用。Nucleus PLUS 的组件包括任务控制、内存管理、任务间通信、任务的同步与互斥、中断管理、定时器及 I/O 驱动等。

项目选用 Nucleus 操作系统作为开发系统。采用嵌入式操作系统目的是降低软件开发的难度，把整个软件划分为若干 task，通过 Nucleus 进行调度管理。Nucleus 的作用主要表现在任务调度和向上层软件提供系统服务，例如：消息传递、信号量、时钟管理、中断管理等。在系统软件设计中增加了一个操作系统适配层(KAL)，使得上层的软件独立于 Nucleus 操作系统。

### 3.2.2 协议栈

如 2.1.2 节所述，GSM 系统按照开放互联系统(OSI)模型采用分层的协议结构设计，并按通信过程分为三个层次，即物理层(L1)，数据链路层(L2)和网络层(L3)。将协议体系进行必要的转化，提出协议栈的概念，并按协议规范对协议栈的功能进行设计，把它划分为四层，分别是 L1 层、L2 层、L3 层和 L4 层。

(1) L1 协议栈

又称物理层或 L1 层，对应 GSM 协议的 L1 层。它依照 GSM05 系列技术规范，支持在无线媒质上比特流传输。它向上层提供服务、控制物理信道和逻辑信道之间的映射，并且控制时序的安排，同时执行 TDMA 取帧和无线控制等。其逻辑结构图 3.4 如下所示：

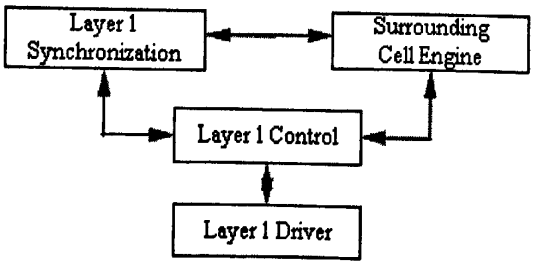


图 3.4 L1 协议栈逻辑

把 L1 协议栈设计成四个组成部分，分别是 Layer 1 Synchronization(L1 层异步逻辑单元)、Surrounding Cell Engine(SCE 单元)、Layer 1 Control(L1 层控制单元)和 Layer 1 Driver(L1 驱动单元)。其中 L1 层异步逻辑单元负责处理上层软件的消息请求和把 L1 处理后的结果发送给上层软件；SCE 单元负责处理相邻小区的功率测量以及同步信息获取；L1 层控制单元负责处理无线环境中的 TDMA 时序安排，包括定时提前以及来自基站的功率控制；L1 层驱动单元负责 DSP 以及无线控制。

(2) L2/L3/L4 协议栈

这里提到的协议栈是对应与 GSM 标准的 L2 和 L3 层，其中信令 L3 层又分为无线资源管理(RM)，移动管理(MM)和连接管理(CM)。而连接管理(CM)又分为三个协议实体，分别是呼叫控制(CC)，补充业务(SS)和短消息业务(SMS)。同时，此协议栈还实现 L3 层之上的短消息结构，分别为中继层(SM-RL)和传输层(SM-TL)。此外，此协议栈还需要支持 SIM 卡单元和小区广播的功能模块。

由协议栈总体结构的描述可知，L3 层是整个协议栈的核心，它包括了大部分的移动终端通信功能实现。它提供一个蜂窝移动网和与其相连接的其他公众移动网之间的建立、维护和释放电路交换的功能；提供必要的支持补充业务、短消息业务和呼叫控制的功能；L3 还包括移动管理和无限资源管理的功能。此外，最新开发的 L3 层应当能支持 GPRS 业务，并提供相应复杂的控制功能。在软件设计过程中，L3 层主要由大量的程序模块构成，这些程序块在第三层各主体之间、第三层与相邻层以及相关层之间传递携带各种信息的消息单元。

L4 是底层驱动和协议栈的封装层，也是和 MMI 通信的接口层，MMI 通过 L4 和底层、协议栈进行信息交互，完成上层应用的各个功能。L4 层又可细分为 L4C 层(L4 Control Layer)和 L4A 层(L4 adaptation Layer)。其中 L4C 处理所有的应用程序请求和响应，L4A 完成与 FMI (Feature rich MMI)的通信。L4C 的划分，确保了 MMI 软件的独立性和可移植性。协议栈结构图如图 3.5 所示。

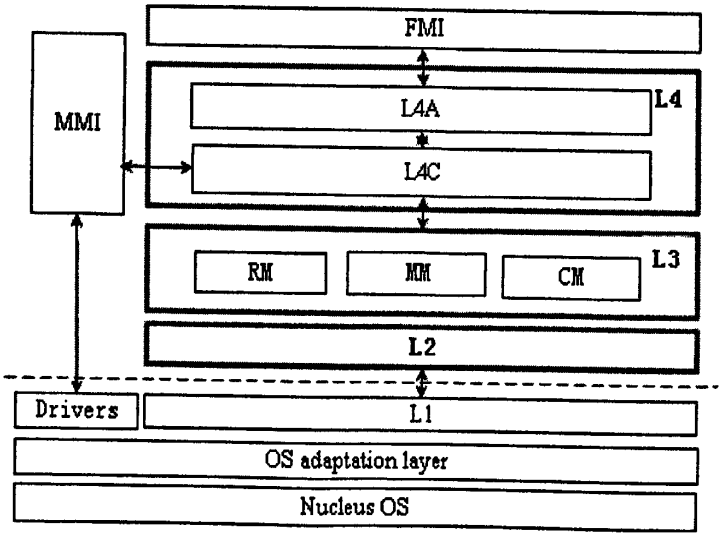


图 3.5 协议栈结构

### 3.2.3 设备驱动程序模块

设备驱动[18]主要用来解释来自微控制器子系统(Micro Controller Unit，MCU)的命令，实现对外设的控制。这个模块处理许多用户可观察的移动终端行为，例

如键盘处理、LCD 显示等。该模块通过函数调用或与 L4 实体进行消息交互的方式被引用，通过访问设备寄存器来提供必要的控制功能。目前，移动终端系统所包括的外围设备如图 3.6 所示。

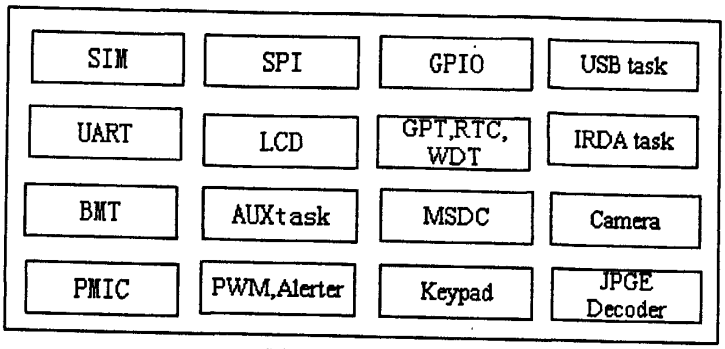


图 3.6 MCU 外围设备

表 3.1 描述了各个外设的基本功能：

表 3.1 MCU 外围设备模块功能描述

| 模块名称 | 模块描述 |
|---|---|
| SIM | 用户识别模块，写入用户信息 |
| UART | 通用异步收发设备，用于 MCU 和外设之间的通信 |
| SPI | 串行接口 |
| LCD | 液晶显示器 |
| GPIO | 通用输入输出接口，用于控制外设 |
| RTC | 实时时钟，用于提供时间 |
| WDT | 看门狗，检测 MCU 的运行情况 |
| GPT | 通用定时器 |
| PWM | 脉冲宽度调制 |
| Alerter | 报警设备 |
| Keypad | 该模块处理所有从键盘发送来的信息，处理后把按键信息发送给上层 |
| PMIC | 电源管理芯片 |
| BMT | 电池充/放电管理 |
| AUX task | 耳机 task |
| USB task | USB2. 0 协议和驱动程序 |
| IRDA task | IRDA 和驱动程序 |
| MSDC | 内存卡驱动程序，支持 SD，MMC 卡和内存片 |
| Camera | 照相机 |
| JPGE Decoder | 控制硬件 PJEG 解码器的软件 |

### 3. 2. 4 MMI 模块

人机接口(MMI)，主要负责人机的交互，MMI 软件[19]既包括基本的人机界面

和菜单部分，还包括各个应用功能模块，主要有电话本、短信息、通话记录、情景模式和显示设置等应用软件。MMI 软件是决定移动终端质量优劣的重要因素之一。

在该课题中，MMI 模块软件的设计是工作的核心，本论文将详细描述 MMI 软件架构的实现与设计，和 MMI 软件典型代表电话本软件的设计与实现，分别在第四章和第五章中阐述。

## 3.3 本章小结

本章介绍了 GSM 移动终端系统的设计，主要包括硬件系统和软件系统。首先对组成硬件系统的各个模块，即射频模块、基带模块和电源管理模块分别进行介绍，并对所选取的芯片作了详细的介绍，设计出移动终端的硬件系统；然后对系统软件进行了选取与设计，包括嵌入式操作系统的选择，协议栈的模块设计，设备驱动程序模块的整体设计。本章所设计的 GSM 移动终端系统，具有一定的代表性，可作为任何一款移动终端系统设计参考所用，并为接下来的 MMI 软件架构和 MMI 软件开发提供了系统平台支持。

# 第四章 MMI 软件架构设计与实现

人机接口(MMI)软件,既包括基本的人机界面和菜单部分，还包括各个应用功能模块，主要有电话本、短信息、通话记录、情景模式和显示设置等。一个良好 MMI 软件架构，可以简化 MMI 软件的开发复杂度、增强 MMI 软件的健壮性、可移植性和可裁剪性，是确保 MMI 软件质量的重要因素之一。

在 MMI 软件架构设计中，采用嵌入式软件架构设计方法，按服务划分层次，并采用消息序列图描述软件设计。

## 4.1 架构设计方法概述

一般来讲，可将嵌入式系统设计划分为三个阶段[20]：架构设计阶段、机制设计阶段和详细设计阶段，如图 4.1 所示。

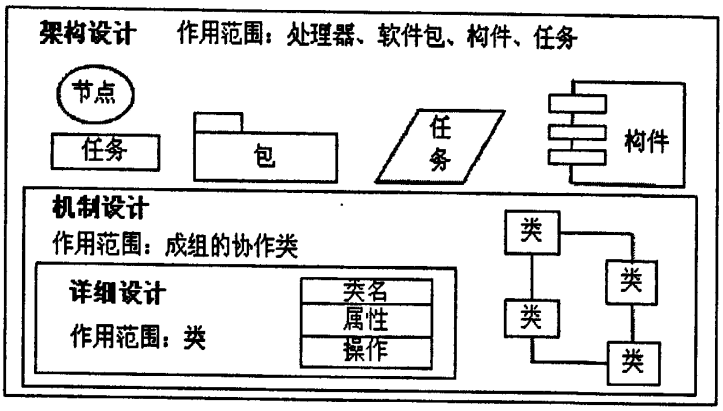


图 4.1 系统设计三个阶段

架构设计在这三个设计阶段的线条最为粗犷，它主要关注策略性的决定，这些决定会影响系统中的所有或大部分成份。比如，任务的集合及它们之间的交互(并发模型)，存在于运行时刻的制品集合及它们之间的接口(构件模型)，构件与物理硬件之间的映射(部署模型)，针对故障的存在而提供的大规模冗余结构(安全性模型)。架构设计相当重要，它为更低层的设计构造进行有效合作提供了基础设施。其作用范围包括处理器、软件包、构件和任务等。

机制设计是设计阶段的中间层次，主要处理小的类和对象集合，协作完成共同的目标。这个层次包含了通过协作实现共同目标的各个类组成的机制方面的设计，其作用范围是成组的协作类。

详细设计的作用范围最小、最具体，它规定了单个类的原始数据结构与算法，要给出设计细节，比如关联的实现策略、对象提供的操作集合、内部算法的选择、异常处理的规定等，其作用范围是类。

本节主要介绍嵌入式系统的软件架构设计。架构设计是从系统的角度，对整

个系统的高层次设计，重点是将系统分解成为更易于管理的子系统，并设计子系统之间的接口。在架构设计中，首先要确定架构设计的目标，整个设计方案都应该以设计目标为决策依据。具体来讲，可从以下几个方面进行架构设计活动：确定设计目标、确定软件体系结构、确定并发解决策略、选择开发平台与已有组件、映射子系统到软硬件、确定数据管理策略、选择完整型控制策略、选择全局控制流、确定边界条件处理、设计界面和应对预期变化设计。

### 4.1.1 确定系统设计目标

系统设计目标[21]是后续活动决策的主要依据，一般来说包括性能、可靠性、费用、维护和最终用户五类设计指标。性能指标主要是指对系统速度和空间的要求，主要参数有响应时间、吞吐量和存储器空间；可靠性指标确定需要作出多大努力来减少系统崩溃的可能性及其带来的后果，主要参数有健壮性、可靠性、可用性、容错、安全性和平稳性；费用指标是指系统开发、配置及管理的费用，主要参数有开发费用、配置费用、升级费用、维护费用和管理费用；维护指标确定系统部署完成后修改系统的难度，主要参数有扩展性、可修改性、适应性、可移植性、可读性和需求的可追溯性；最终用户指标是从最终用户的角度对系统提出的期望，包括可用性和实时性等，它体现了软件产品作为一种商品的品质。

### 4.1.2 系统分解

软件架构设计[22]的首要任务是将系统分解成为易于管理的子系统，并且设计子系统间的接口、确定系统的软件体系结构。其中，涉及到三个重要的基本概念：子系统、服务和子系统接口。子系统是一组为提供某类特定服务的相互关联的模块的集合；服务是子系统为实现某一共同目标而提供的一组相关操作；子系统接口也称为子系统 API(Application Programmer's Interface)，服务通过子系统接口来提供服务。

系统分解通常按照服务进行划分，即将为实现共同目标的服务放在一个子系统中。系统分解不仅要将系统分解为子系统，而且还要确定子系统之间的接口，即确定子系统间的交互及信息流，子系统接口应尽可能简单。系统分解的原则是子系统应具有最大的聚合度和最小的耦合度[23]。

系统分解一般有两种分解方式：分层和分区，大型系统常常同时采用分层与分区两种方式进行系统分解。分区是指将系统分解为对等的子系统，子系统间依赖较小，每个子系统可以独立运行。分层是将系统按照一定的层级结构划分，一个层是一个子系统，每个层至少包括一个子系统，各层也可以继续划分成为更小的子系统，图 4.2 将系统分解为三层。分层结构有以下三个特点：

(1) 每一层为更高级别的抽象层提供服务；

(2) 每一层仅依赖于比自己级别低的层次；

(3) 每一层都不知道比自己级别高的层次的任何信息。

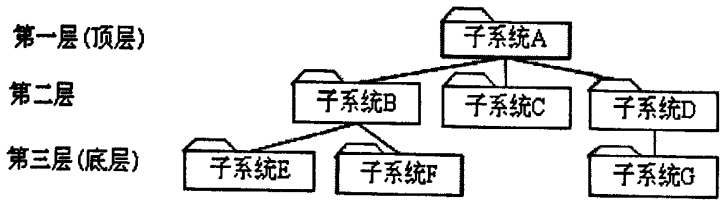


图 4.2 系统分解 (UML 对象图)

### 4.1.3 软件体系结构

系统分解的目的是确定系统的体系结构，体系结构[24]的设计过程主要是为系统建立一个基本架构，包括构建系统主要的子系统及这些子系统之间的通信。嵌入式系统的体系结构由硬件体系结构和软件体系结构构成，课题研究的是软件体系结构。系统的软件体系结构包括子系统的划分、全局控制流机制、错误处理策略和子系统间的通信协议。

常见的嵌入式软件可分为无操作系统的嵌入式软件体系结构和有操作系统的嵌入式软件体系结构，结合论文的需要，这里只简述有操作系统的嵌入式软件体系结构的设计，无操作系统的嵌入式软件体系结构可参考资料[25]。嵌入式操作系统可以分为分时操作系统和实时操作系统，其中，实时操作系统又可分为抢占式实时操作系统和非抢占式实时操作系统。下面分别介绍这些操作系统的嵌入式软件体系结构。

(1) 基于分时操作系统的软件结构。分时操作系统使用定时器和任务调度管理器来管理任务的执行，如 Linux 操作系统。其缺点是无法体现任务的重要性，即优先级。基于分时操作系统的软件结构如图 4.3 所示。这种系统的缺点是无法保证事务处理的优先级，适合不需要实时处理的应用，如 PDA。

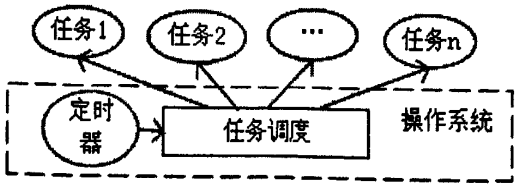


图 4.3 基于分时操作系统的软件架构

(2) 基于实时操作系统的软件结构。实时操作系统把系统处理的事件根据轻重缓急进行分类，并赋予不同的优先级。

非抢占式操作系统总是从最高优先级的任务开始执行，直至任务放弃处理

器。系统总是先运行最高优先级的任务；在低优先级的任务运行时，高优先级的任务不能中断低优先级的任务；系统简单，操作系统的开销小。

抢占式操作系统会定时检查任务表，如果有更高优先级的任务出现，处理器会把正在执行的任务挂起，先执行更高优先级的任务。抢占式调度特点是：系统总是先运行最高优先级的任务；一旦高优先级的任务就绪，低优先级的任务就会被中断；系统复杂，操作系统开销大。

在实时操作系统中，中断程序处理大多数紧急情况并发出“需要任务代码来完成工作”的请求。基于实时操作系统的软件结构中，中断程序和任务代码之间的必要信号的发送是由实时操作系统处理的，不需要使用共享变量来达到这个目标；代码中不用循环来决定下一步的操作内容，实时操作系统知道各种任务代码的子程序，并且可以在任何时刻运行它们中相对紧急的一个；实时操作系统可以在一个任务代码子程序运行期间将其挂起，以便运行另一个子程序。

如图 4.4 所示，基于实时操作系统的软件系统由以下部分组成：

- 任务：包括用户任务和操作系统；
- 操作系统的任务调度器：根据任务的优先级进行调度任务的执行；
- 引起任务调度的因素：硬件中断、定时器溢出、任务之间的通信和同步等；
- 其它程序：如主程序、子程序等。

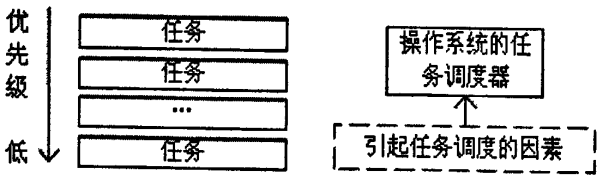


图 4.4 基于实时操作系统的软件系统组成

嵌入式系统的需求越来越复杂，比如项目的移动终端系统，不仅需要嵌入式操作系统嵌入式数据块，还需要网络通信协议、应用支撑平台等，在此基础上的应用软件的架构也变得复杂起来。基于嵌入式操作系统的嵌入式软件体系结构非常丰富，许多通用软件体系结构都可以用于设计此类嵌入式软件。常见的有：

- 仓库体系结构：每个子系统只依赖数据仓库中心数据结构，而仓库并不清楚其他的子系统；
- 模型/视图/控制器(MVC)：控制器收集来自用户的输入并发消息给模型，模型保存中心数据结构，视图显示模型，每当模型发生变化时通过通过签署/通知协议得到通知；
- 客户/服务器体系结构：一个或几个子系统作为服务器，为其他称作客户端的子系统提供服务；客户端负责与用户交互。基于数据库服务器的信息管理系统是典型的客户/服务器体系结构；
- 对等体系结构：对等体可以向其他对等体请求服务，也可以向它们提供服

务。

- 管道和过滤器体系结构：一个过滤器可以有多个输入和输出，一个管道将某个过滤器的输出和另外一个过滤器的输入连接起来。

(3) 基于嵌入式操作系统的分层体系结构。常见的基于嵌入式操作系统的软件分层体系结构如图 4.5 所示。

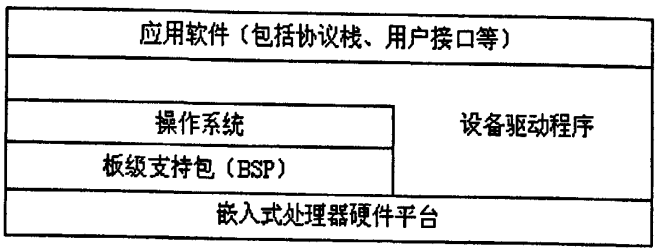


图 4.5 基于嵌入式操作系统的软件分层体系结构

由于各种嵌入式操作系统的 API 不同，为了提高应用软件的可移植性，可以增加操作系统抽象层，以屏蔽不同的嵌入式操作系统 API 的多样性，如图 4.6 所示。嵌入式操作系统抽象层的特点：

- 操作系统抽象层具有与实际的嵌入式操作系统的密切相关性；
- 操作系统的抽象层具有与应用软件的无关性；
- 接口定义的功能应包括嵌入式操作系统的所有功能；
- 操作系统的抽象层的实现应该可以裁剪；
- 接口定义简单明了，符合大多数流行操作系统接口；
- 具有可测性的接口设计有利于系统的测试和集成；
- 实现的时候，尽量保存原来嵌入式操作系统的性能。

基于抽象层的软件系统设计的优点是可移植性好，缺点是效率低。因此，在设计时，抽象层尽可能只是一些抽象接口，也可以采用开放的分层体系结构。

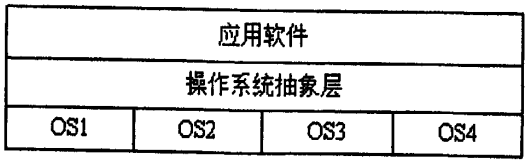


图 4.6 基于操作系统抽象层的软件分层体系结构

### 4.1.4 全局控制流机制

控制流是系统中动作的先后次序。一般有过程驱动、消息驱动和线程驱动三种控制流机制。

过程驱动：一旦需要来自操作者的数据时，操作就等待输入。这种控制流机制大多用于使用过程化编写的系统，许多嵌入式系统都是用过程化语言编写的。

消息驱动：主循环等待外部消息。一旦外部消息发送，就触发相应的事件操

作。消息机制可以很好的解决事件驱动的多应用设计问题[26]，它是课题所采用的控制流机制。

线程驱动：也称轻量级线程，以便与需要更多计算开销的进程分开，它是过程控制流的并发变异：系统可以创建任意数量个线程，每个线程对应不同的事件。

### 4.1.5 人机界面设计

架构设计阶段，考虑人机界面设计仍然是较高层次的设计决策。嵌入式系统的人机界面设计与整个系统的设计均有关系。选择何种人机界面，将会影响硬件结构设计、软件架构、编程语言的选择等。

系统人机交互界面设计需要从用户、任务、系统和环境等各个方面综合考虑，一般需要考虑的问题有系统使用者是谁、系统的目的、系统的使用时间以及系统的使用环境等。人机界面的设计一般要尽可能满足以下设计原则：

(1) 一致性：界面的布局、色彩、字体、操作方式尽可能一致；

(2) 效率：系统的效率不是仅靠界面设计策略决定的，但界面设计策略同样会影响系统的效率。尤其对使用者而言，许多时候，系统效率是通过人机界面的响应效率体现的；

(3) 易用：输入操作越简单，系统越易用，这对设计者的要求也越高；

(4) 格式化/规范化：尽可能格式化或规范化输入输出，这样可以提高使用效率，也可以减少输入输出的错误；

(5) 灵活性：系统应该适应不同人群的使用。

在架构设计阶段，主要确定界面体系结构。设计一个好的用户界面体系结构涉及到定义工具、材料以及用来摆放它们的环境，并且把用户界面的所有内容分布到彼此不同但却相互关联的若干交互空间去。

在课题的系统设计中，采用图形界面，以菜单的形式来提供交互操作。

### 4.1.6 描述工具

嵌入式软件开发过程复杂，软件内部各个层次之间、各个模块之间、实例之间，以及与外部环境之间，都存在着许多联系，怎么正确处理这些关系，在嵌入式软件的设计与实现中起着重要的作用。消息序列图(Message Sequence Charts，MSC)[27]在消息交互处理方面有着突出的应用，符合嵌入式软件设计与开发的需要，贯穿在项目的 MMI 软件架构设计以及 MMI 软件设计开发的全过程中。

消息序列图主要用于描述软件系统中通信实体之间异步信息交换，反映了实体之间的动态交互特征。消息序列图最初一般用于在系统建模初期描述系统动作行为，随着 MSC 的发展，MSC 的应用领域越来越广泛，不仅用于分析和设计阶

段，也可用于程序理解。

消息序列图 MSC 的定义为 MSC=(Γ，E,P,A), 其中，Γ为消息的集合，E 为事件的集合，P 为过程的集合，P={P1 ,P2 ,...,Pn}，其中 Pn (n=1,2,...)为 P 过程的各个子过程，A={S∪R∪ACT}，ACT 为原子动作集合，S= {send(i,j,m)} ,R= {receive(i,j,m)}，i,j ∈P，m ∈Γ。

MSC 由有限多个实例(Instance)组成，图 4.7 给出了 MSC 的示例，i1，i2 表示实例，m1, m2 表示信息，其中 m2 表示输出到环境的信息，a 表示本地动作。

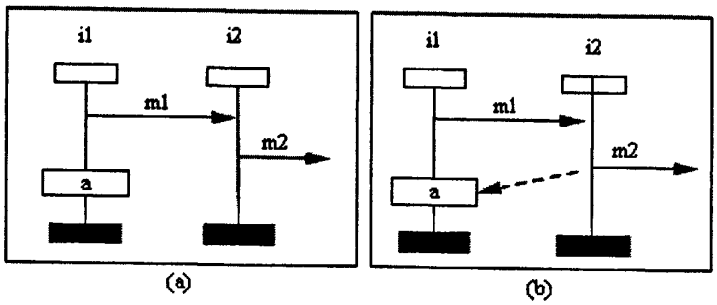


图 4.7 MSC 示例

实例是一个抽象的实体(entity)，用垂直轴表示。消息是实例间或者实例与环境间信息交互的基本单位，用箭头表示。消息既可以是一个简单的信号，也可以是承载数据的复杂数据包, 在实体上可以指定本地动作。

事件是可观察的基本单位，例如：两个实例之间交换一条消息可以看成两个事件：消息入(Input event)和消息出(Output event)，同一消息的输入与输出是异步事件。事件还包括本地动作、定时器等。同一实例轴上的事件按照从上到下的时间顺序依次发生，如图 4.7(a)中实例 i1 上的事件发送顺序是：消息 m1 输出，本地动作 a 发送。任一消息的输出事件发送在该消息的输入事件之前，即实例 i1 上消息 m1 输出在实例 i2 上消息 m1 输入之前发生。但是，在实例 i1 上本地动作 a 和实例 i2 上消息 m2 输出事件哪个先发生无法确定。图 4.7(b)中引入了全局顺序(general ordering)的概念。由此可以定义本地动作 a 和消息 m2 输出事件间的全局顺序，消息 m2 输出事件是高端事件，本地动作 a 是低端事件。全局顺序的高端事件发生在低端事件之前。不同实例上的事件通过消息的发生和全局顺序确定发生的先后顺序。除此之外事件之间没有其它顺序约束关系，因此，MSC 事件集上存在偏序关系。

嵌入式软件设计中，要描述系统内部过程与过程之间的交互作用，还有系统环境与外部环境之间的交互作用。MSC 是一个简洁、直观、图示意的标准化注释型系统，可以很好表示嵌入式软件内部各部件之间及其与环境的交互作用，正是适合和解决这种消息驱动系统的一种绝佳的手段。因此，将 MSC 应用于本软件开发中，不仅在系统设计时十分有用，而且在后期的功能实现、问题分析和测试中也发挥着很大的作用。

## 4.2 MMI 软件基本架构设计

MMI 软件的基本架构[28]是后续架构设计的总概括，为更低层的设计进行有效合作提供了基础依据。主要任务是把 MMI 软件按服务划分为几个有效层次，并确定各个层次之间的服务关系和与外面环境之间的交互接口。

### 4.2.1 系统分解

从 MMI 软件所提供的功能来看，既包括基本的人机界面和菜单部分，还包括各个应该功能模块，主要有电话本、短信息、通话记录、情景模式和显示设置等应用软件。根据这些功能，从服务的观点把 MMI 软件分解为三个层次：

(1) 提供满足用户基本功能需求的服务，比如电话本等应用软件；

(2) 提供用户操作各项功能的操作界面的服务，比如菜单等；

(3) 提供连接以上两种服务并与外部环境交互的服务。

### 4.2.2 基本架构设计

根据以上系统分解，把 MMI 软件基本架构划分成的三个层次：应用层(Application Layer)、图形用户接口层(Graphical User Interface Layer， GUI Layer)和框架层(Framework Layer)。如下图 4.8 所示，应用层对应 4.2.1 节的第一个层次的服务，它为用户定义的各种应用程序的集合，如电话本、短信息、通话记录、情景模式和显示设置等；GUI 层对应 4.2.1 节的第二个层次的服务，包含了主题元素、UI 组件、类别屏幕、字体、编辑器和各种用户工具等，为应用层提供绘屏的服务；框架层则对应 4.2.1 节的第三个层次的服务，该层负责对消息和事件的处理，优化应用层发送过来的信息流和对操作系统的抽象。

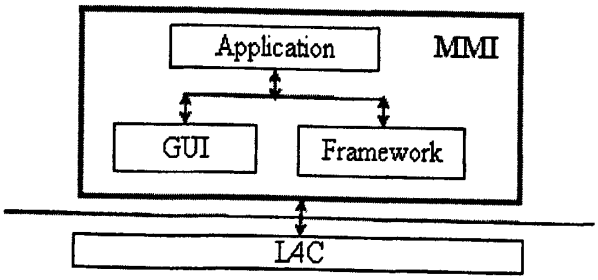


图 4.8 MMI 软件基本架构图

因此可以说框架层主要实现调度功能；而应用层实现各个服务功能，每个功能又自己组成一个模块，应用层可以根据具体的需要增加或删减子模块；GUI 负责显示用户友好界面。设计中把框架层与应用层划分开，增强了软件的可扩展性和灵活性，方便模块的封装和程序的二次开发。

MMI 通过 L4C 和底层、协议栈进行信息交互，因此设计一个 MMI 层与 L4C

通信的软件接口。根据三层各自的内容，可知框架层主要实现调度功能。因此，接口的设计采用由框架层负责，实现与 L4C 层的通信。接口的设计，保证了 MMI 软件与系统的有效连接和独立。

### 4.2.3 控制流设计

在 4.1.4 节中已经提到，项目采用消息驱动的控制流机制来设计软件。这里把在 Nucleus 嵌入式操作系统的任务(task)概念引入到设计中来，每一个 task 都是相互独立的死循环，由各自的任务管理模块处理，在系统运行期间一直执行某一具体动作。各个任务模块有不同的优先级，操作系统根据优先级的高低，分别处理各个任务。在任务模块之间，采用消息驱动机制来实现通信，每当任务管理模块接收到消息，就会对消息做出识别并进行响应处理。

利用 task，采用消息驱动机制，可以方便的实现 MMI 在系统中的运行机制。为了提高系统的实时效率，在 MMI 与系统通信中加入队列机制。这样，task 通过查询队列的消息来驱动事件的执行。运行机制设计如图 4.9 所示。

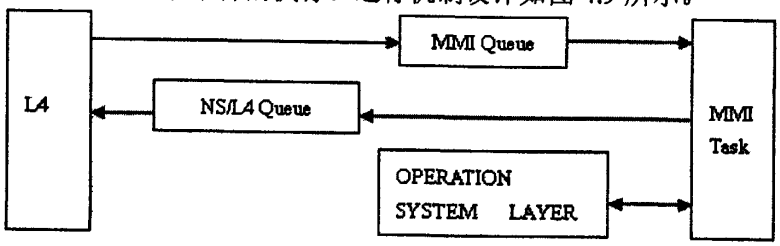


图 4.9 MMI 运行机制

L4 层与 MMI 通信，向 MMI 队列写入消息，MMI Task 不断询问队列，一旦发现有信息到达，MMI 任务框架层将负责从 MMI 队列中读取信息并进行处理，根据消息的类型，或向另一个 MMI 任务发出新消息，或触发应用层的某一回调函数[29]，该回调函数将调用 GUI 层的窗口函数和主题资源来绘制窗口。

MMI 任务向 L4 层通信时，系统由应用层通过接口函数调用框架层，框架层负责发送消息到 NS/L4 队列中，L4 层将从队列读取该消息，根据消息作进一步处理后，或向 MMI 任务做出响应，把响应消息写到 MMI 队列，或对 PS/L1 层发出底层操作请求消息。

## 4.3 框架层设计与实现

框架层(Framework Layer)对应 4.2.1 节的第三个层次的服务，该层负责对消息和事件的处理，优化应用层发送过来的信息流和对操作系统的抽象。在 MMI 软件三层架构中，框架层为 MMI 调度的核心，它是通过解释消息来实现的。该层主要实现的功能为：负责查询 MMI 队列，接收来自所有 task 的消息；处理并根

据消息回调Application显示屏幕的流程；为系统提供调用应用时所需要的数据包。按照这些功能划分模块，把 Framework 的内容设计为事件处理器、历史管理、操作系统适配层(OS adaptation layer，OSL)、存储器管理(NVRAM)和系统文件五个子功能，结构图如图 4.10 所示。

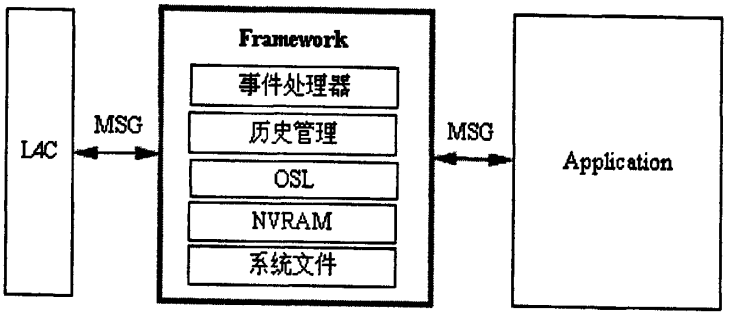


图 4.10 框架层结构

### 4.3.1 事件处理器

事件处理器的主要作用是注册各种事件和执行 Application 中的回调函数。在框架层中，框架层 task 处理 MMI 队列的消息并触发各类事件的发生，根据消息源的不同，事件可分为协议栈事件、高亮事件、按键事件和计时器事件。

(1) 协议栈事件

框架层 task 不断从 MMI 队列获取消息，其中，有一部分消息来自协议栈，所触发的事件定义为协议栈事件。协议栈事件通过 task 实现底层和 MMI 之间的通信。例如，当有一个来电，L4 层将产生一个 INCOMING_CALL_EVENT 事件，如果用户选择了接听，MMI 应用产生一个 CALL_ACCEPT_EVENT 事件。协议栈的实现过程主要分为设置协议事件处理器和协议栈事件回调两个步骤。

- 设置协议事件处理器：在 Framework 中定义一个全局数组 ProtocalEventHandle[]来存放事件和相对应的 Handler Function。无论何时 Application 调用 SetProtocalEventHandler(CallbackFunc，EventID)，EventID 对应的一个回调函数 CallbackFunc 被加入到该数组中。这个数组在开机初始化时被赋值。其消息序列图如图 4.11 所示。

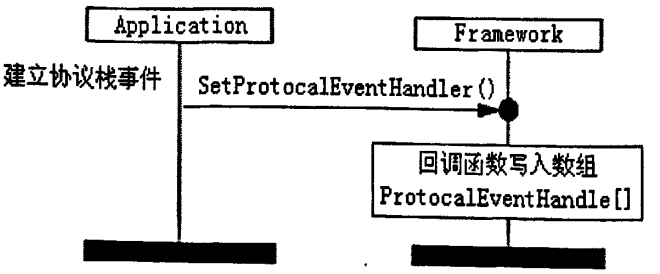


图 4.11 设置协议栈事件处理器

- 协议栈事件回调：L4C 先 MMI 发送协议栈事件，Framework 的 MMI_task 查询来自 MMI 队列的消息。一旦确认消息，调用 ProtocalEventHandler( ) 函数，这个函数将调用执行 ExecuteCurProtocalHandler( )函数。ExecuteCurProtocalHandler( )检查协议事件处理器数组来寻找响应的 EventID。如果一个 EventID 被找到，则调用其对用的回调函数，否则，事件将被忽略。其消息序列图如图 4.12 所示。

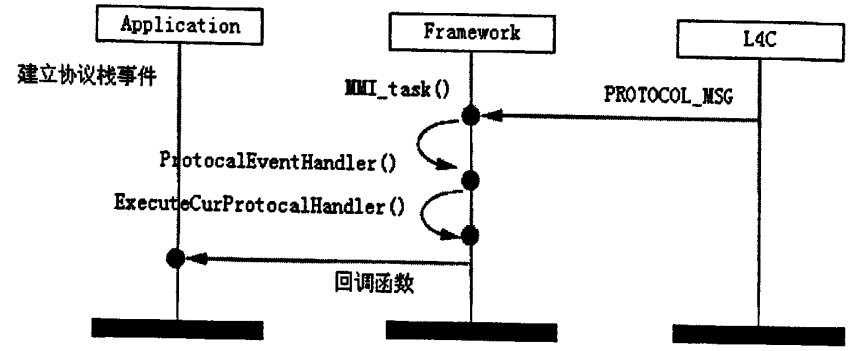


图 4.12 协议栈事件回调

(2) 高亮事件

又称为 Highlight 事件，属于内部事件。当一个特定的菜单被选中时，即处于 Highlight 状态，此时，Framework 将产生一个 Highlight 事件。Application 利用这些事件执行高亮度显示、键盘匹配和改变右软键的标签。Highlight 函数通常作为回调函数被菜单项的 Application 函数注册。

(3) 按键事件

用户使用键盘产生的事件，定义为按键事件。按键事件实现过程分为设置按键事件处理器和按键事件回调两个步骤。

- 设置按键事件处理器:Framework 建立一个全局数组 CurrKeyFuncPtr[][]来存储事件及其对应的响应处理函数。Application 建立按键事件并调用 SetKeyHandle(keycode，keytype，CallbackFunc)函数，Framework 的 MMI task 查询获得该事件，把 keycode、keytype 和 CallbackFunc 等项目加入全局数组 CurrKeyFuncPtr[][]，其消息序列图如图 4.13 所示。

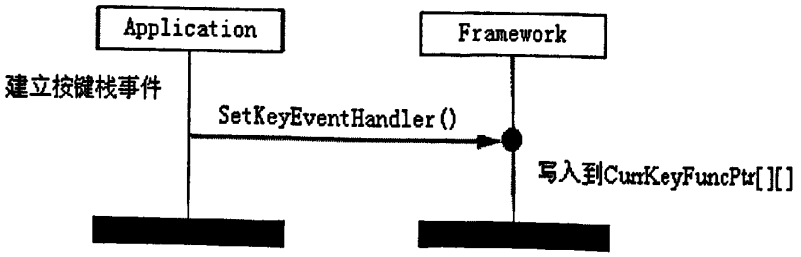


图 4.13 建立按键事件

- 按键事件回调：把 L4C 的按键处理函数注册为按键事件的协议处理函数。这个事件作为来自 L4 的协议事件，MMI task 查询收到该事件，L4 的

L4KeyHandle( )将会被作为回调函数被调用。这个函数检查信息中的 keytype 和 keycode 并调用 CurrKeyFuction[][]数组中的回调函数，并做进一步处理。如果 keycode 不存在或回调函数为空，则按键事件被忽略。按键事件回调的消息序列图如图 4.14 所示。

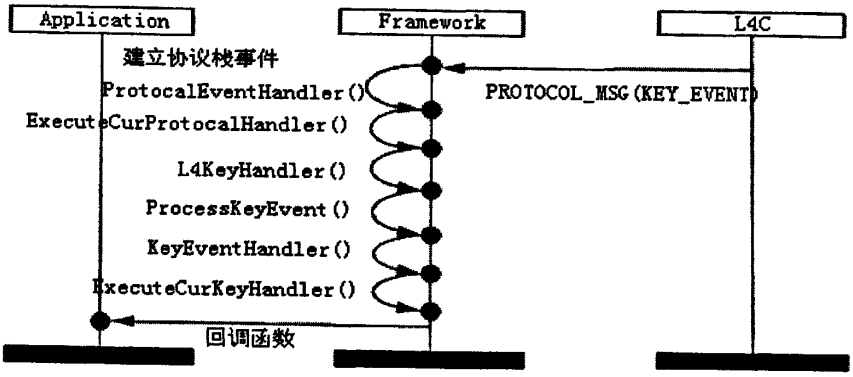


图 4.14 按键事件回调流程

(4) 计时器事件

计时器事件用于通知底层开始和结束计时。其实现函数为：void StartTimer(U16 timerid, U32 delay, FuncPtr funcPtr)。

### 4.3.2 历史管理

框架层要求处理消息，并根据消息回调 Application 显示屏幕的流程，该任务由历史管理实现。历史管理是对屏幕流程的一个记录，一个屏幕看作一个历史节点。历史是以堆栈的形式来实现的，启动后的第一个屏幕被存储，而且不会从堆栈中移除，新的节点被添加在堆栈的头位置。

历史节点的数据结构如下：

```
typedef struct _historyNode {
    U16         screnID;   //屏幕的 ID
    FuncPtr     entryFuncPtr;   //进入新屏的指针，即重新绘图函数
    U8          *inputBuffer;   //保存当前屏幕数据的 buffer 指针
    U8          *guiBuffer;   //保存当前屏幕 UI 的相关信息的 buffer 指针
} historyNode;
```

设当前屏为 CurrScreen，当进入一个新屏 NewScreen 时，我们调用 EntryNewScreen( )函数，把当前屏的信息，包括进入函数、退出函数、屏 ID、历史信息及 Buffer 中的其它信息，压入屏历史堆栈中。刚刚压入屏历史堆栈的当前屏位于栈顶，栈底为待机屏 IdleScreen。若继续进入新屏，则按照上述，继续保存当前屏信息到历史堆栈，再绘制新屏。当退出当前屏时，回调 GoBackHistory( )函数，屏历史堆栈中恢复最近一次显示的屏信息，并根据此信息恢复屏。实现机

制及函数如图 4.15 所示。

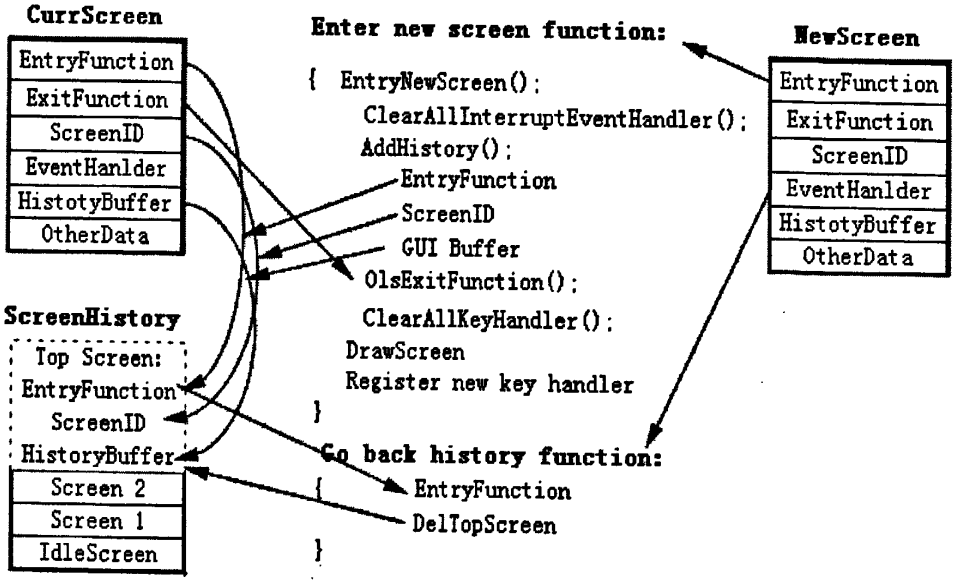


图 4.15 历史管理实现

### 4.3.3 操作系统适配层

框架层中的操作系统适配层(OS adaptation layer，OSL)组装队列的一些 API，这些 API 主要有两个功能，一个是创建消息、发送消息和读取消息；另一个是创建计时器、开始和停止计时器。调用该层的消息 API 函数可以实现 MMI task 和其它 task 之间的信息交换，调用该层的计时器 API 函数可以实现 MMI task 中所有计时器的功能。下面主要介绍队列和计时器的 API。

(1) 队列操作的 API 函数：

- 创建队列：

oslMsgqid OslIntCreateMsgQ(PS8 queue_name，U32 max_msg_size，U32 max_msgs);

其中：queue_name 为队列的名称，max_msg_size 为消息的最大长度，max_msgs 为一个队列中所容纳的最多消息数目。

- 向队列中写入消息：

OSLSTATUS

OslInitWriteWsgQ(oslMsgqid，void *msgPtr,U32 *msgsize,OSLWAITMODE wait_mode);

- 从队列中读取消息：

OSLSTATUS

OslInitReadWsgQ(oslMsgqid，void *msgPtr,U32 *msgsize,OSLWAITMODE

wait_mode);

(2) 计时器操作的 API 函数：

- 开始一个计时器

StartTimer(U16 timerid,U32 delay,FuncPtr funcPtr);

其中，timerid 为计时器 ID，delay 表示延迟的时间，funPtr 表示当前延迟时间结束后调用的处理函数指针。

- 结束一个计时器

StopTimer(U16 timeid)。

### 4.3.4 存储器管理

框架层的存储器管理主要是针对非易失性随机存储器(NVRAM)而言的，NVRAM 保存着手机中非易失性数据，终端中所有参数的设置值都保存在 NVARM 中，其大小为 2Mbytes，数据项的管理方式像文件系统管理文件一样。在 16M 的 Flash 中，NVRAM 的位置如图 4.16 所示。

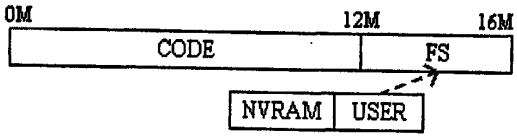


图 4.16 Flash 存储图

MMI 在 NVRAM 中保存的数据类型如下：

| | |
|---|---|
| Byte Data | 1byte |
| Short Data | 2bytes |
| Double Data | 8bytes |
| Application specific Record | 2K bytes |

对存储器的数据进行操作时，设计操作函数如下：

(1) 读出或写入单项记录(Value)函数：

ReadValue(nId,pBuffer,nData Type,pError);

WriteValue(nId,pBuffer,nData Type,pError);

(2) 读出或写入多项记录(Record)函数：

WriteRecord(nFiled,nRecordId,pBuffer,nBufferSize,pError);

ReadRecord(nFiled,nRecordId,pBuffer,nBufferSize,pError)。

### 4.3.5 文件系统

操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称文件系统。文件系统由三部分组成：与文件管理有关的软件、被管理的文

件以及实施文件管理所需的数据结构。从系统角度来看，文件系统是对文件存储器空间进行组织和分配，负责文件的存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。系统的文件系统为 Nucleus 所提供，所要到的 API 函数主要有：

int FS_Open(const WCHAR* FileName, UINT Flag);

int FS_Close(FS_HANDLE FileHandle);

int FS_Read(FS_HANDLE FileHandle, void* DataPtr, UINT Length, UINT* Read);

int FS_Write(FS_HANDLE FileHandle, void* DataPtr, UINT Length, UINT* Written);

int FS_Seek(FS_HANDLE FileHanle，int Offset, int Whence);

int FS_Delete(const WCHAR* FileName)。

## 4.4 用户接口层设计与实现

用户接口层(GUI)对应 4.2.1 节的第二个层次的服务，包含了主题元素、UI 组件、类别屏幕、字体、编辑器和用户工具等，为应用层提供绘屏服务，负责显示屏幕所需要的资源，包括图像、主题、字体和声音等资源。GUI 层的设计主要任务是有效的管理这些资源元素，实现窗口的有效运行。根据 GUI 层所提供的功能把 GUI 层划分为窗口管理、类屏幕和 UI 元素三个层次，框架图如图 4.17 所示。

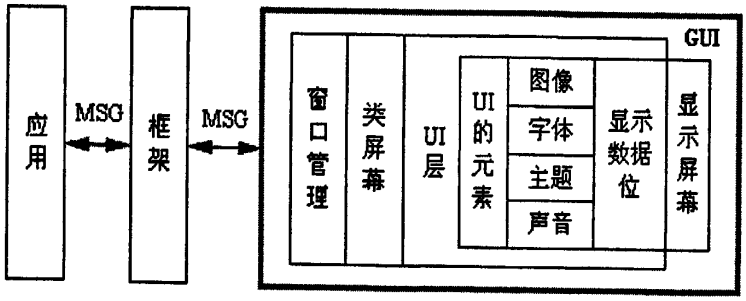


图 4.17 GUI 层框架图

### 4.4.1 窗口管理

窗口[30]管理调用窗口函数，向窗口函数传递参数，这些参数描述了窗口的许多特性，如窗口类型、大小、颜色、透明度等，再把消息送到目的窗口。窗口函数是其所属窗口在窗口管理中注册的系统函数，它根据接收到的消息，确定窗口的表现和行为特征，因此，在应用程序就只需要负责创建窗口、处理窗口产生的高级事件和窗口的异常情况。

窗口管理中，消息驱动机制贯穿始终。在底层，协议栈消息既可以由用户通过窗口进行处理，也可以由系统对它进行缺省处理；而把按键事件的消息设计成被窗口管理所接收，然后负责传送到具有当前焦点的窗口。在整个程序运行过程中，总存在一个具有焦点的窗口，该窗口负责处理接收到的输入消息，根据消息的内容，可能调用窗口函数重新绘制当前焦点窗口，也可能把消息沿着从子窗口到父窗口的方向传递链，送到一个可以接收并处理该消息的目的窗口。

### 4.4.2 类屏幕

一般来讲，移动终端的 LCD 显示大小有所限制，其中显示的屏具有很多相同的特性，据此，我们给出了这样的定义：类屏幕(Category Screen)是一组定义用户界面的函数，这些函数称为类函数，它们在结构和功能上都是相近的。这里以一个简单例子进行说明，如图 4.18 所示。

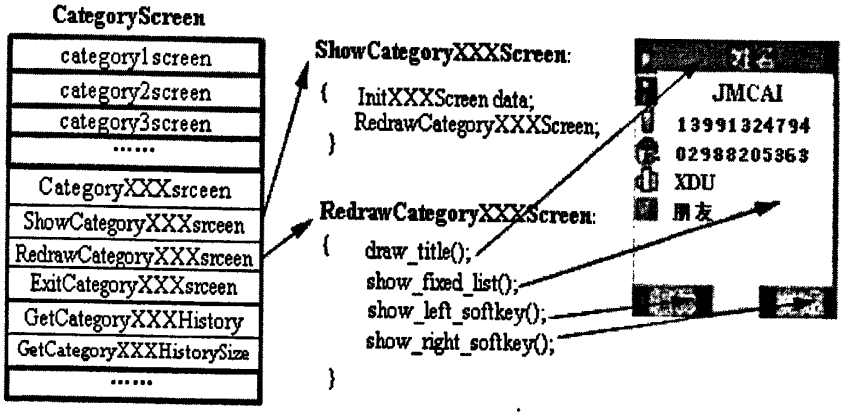


图 4.18 类屏幕

图中，左边类屏幕 CategoryScreen 中的函数均为类函数，它们实现了相近的画屏功能，函数 ShowCategoryXXXScreen( )函数画屏效果如右边图所示。类屏幕主要包括的功能函数有：

(1) 显示一个类屏幕的函数，又称为进入类屏幕函数。

显示一个类屏幕所用的函数为 ShowCategoryXScreen( )，其中的 X 是某一数字，代表显示不同的类屏，类函数实现注册事件、预处理 UI 元素和调用重新绘图函数。比如，ShowCategory1Screen( )函数声明如下：

```
void ShowCategory1Screen(
        STRING_ID    Title,//标题
        IMAGE_ID     TitleIcon,//标题前所显示的小图标的 ID
        STRING_ID    LSKLabel，//左键标签
        IMAGE_ID     LSKIcon，//左键小图标
        STRING_ID    RSKLabel,//右键标签
```

```
    IMAGE_ID      RSKIcon，//右键小图标
    INT           NumberOfItems，//菜单项数目
    STRING_ID*        ListOfItems，//菜单项列表
    BYTE*         HistoryBuffer//历史缓冲器
);
```

(2) 退出一个类屏幕的函数，当类屏幕不再需要显示时调用该函数。

当类屏幕不再需要显示，即要画新屏时，需退出当前的屏函数。在退出时调用的函数是 ExitCategoryXScreen( )，值得注意的是，这个屏幕不会从显示中清除，而是压栈到历史缓冲器中，当需要时再调出。该机制可参考 4.3.2 历史管理。同样以 1 类屏幕为例，其退出屏幕的函数声明为 void ExitCategory1Screen(void)。

(3) 得到一个类屏幕历史数据的函数，得到类屏幕的状态信息。

这个状态信息能被存储和恢复，可以通过调用显示函数来返回到它以前的状态，该过程同样可参考 4.3.2 历史管理。获取一个类屏幕历史数据的函数是 GetCategoryXHistory( )，这使得屏幕的状态信息能被存储和恢复，可以通过 ExitCategoryXScreen( )函数来返回到它以前的状态，主要用于更新屏幕。这里同样以 1 类屏幕为例，声明为：U8 *GetCategory1History(U8 *history_buffer)。

(4) 得到历史的大小函数，该函数得到存储屏幕的历史数据所需要的空间比特数。调用的函数为 GetCategoryXHistorySize( )。再次以 1 类屏幕的函数为例，该函数声明为：S32 GetCategory1HistorySize(void)。

(5) 改变按键标签的函数：

改变左键的标签：void ChangeLeftSoftKey(string_id,image_id);

改变右键的标签：void ChangeRightSoftKey(string_id,image_id)。

(6) 建立按键的函数，分为建立左键函数和建立右键函数：

void SetLeftSoftKeyFunction(void(*f)(void),key_event_type_k);

void SetRightSoftKeyFunction(void(*f)(void),key_event_type_k)。

(7) 在菜单屏幕中 highlight 的项目函数：int GetHighlightedItem(void)。

(8) 设置被 highlight 的函数：void SetHighlightedItem(int item_index)。

(9) 移除 highlight 处理函数：void ClearHighlightHandler(void)。

### 4.4.3 UI 元素

UI 元素利用图像、主题、字体和声音等资源来建立用户界面，是通过利用数据结构和函数来实现的。UI 元素的函数构成了 UI 层，用来连接类屏幕和 UI 元素的数据结构，支持类屏幕的有效绘图机制；UI 元素的数据结构：包括所有的 UI 需要显示的信息、包括与绘图相关的部分、绘图信息包括坐标，维数和与主题相

关的信息。

例如，可定义一个简单的按钮 UI 元素，如下所示：

```
button {
    int     x,y;  //平面坐标
    int     width, height;  //维数，即宽度和高度
    string  text;  //标志
    };
```

那么，可定义它的创建 UI 元素函数为 CreateButton(button* b,int x,int y,string text)；显示 UI 元素函数为 ShowButton(button *b)。

## 4.5 应用层设计与实现

应用层(Application Layer)对应 4.2.1 节的第一个层次的服务，它为用户定义的各种应用程序的集合，如电话本、短信息、通话记录、情景模式和显示设置等。可以看出，应用层包含着移动终端提供给用户的各种功能，它是呈现给用户最直接的交互方式；并且，移动终端功能的多样化和特殊化，都是在这一层得以体现的。

在设计中，把该层进一步进行分解成多个子功能模块，每个子功能模块为用户提供特定的服务，比如电话本为用户提供了管理电话信息的特定服务。这样可以根据具体的功能需要方便地增加或删除子模块。该层既建立于框架层和 GUI 层又独立于框架层和 GUI 层，极大的增强了软件的可扩展性。具体来讲，完整的移动终端应用层一般包括以下子功能模块：

(1) 电话本
(2) 信息(SMS，EMS，CB)
(3) 呼叫记录
(4) 设置(话机设置，通话设置，安全设置)
(5) 情景模式
(6) 娱乐与游戏
(7) 工具箱
(8) 快捷方式
(9) 呼叫管理
(10) 终端事件(如充电，闹钟)

本文将在第五章中，以电话本子模块为例，设计并实现电话本软件。其他子功能模块在实现方法上大同小异，可以采用同样的方式，按照功能需求实现，本文不作展开叙述。

## 4.6 软件接口设计与实现

由 4.2.2 基本架构设计可知，根据需要，系统设计一个 MMI 与 L4C 的软件接口，也可以看成是 Framework 与 L4C 层的接口，用来实现 MMI 与外部的通信。该接口的设计如图 4.19 所示，MMI 通过此接口和 L4C 层进行消息的发送和接收，完成信息的交互，实现底层或协议栈所需要的服务。其中 MOD_MMI、MOD_L4C 等表示为 MMI 层与 L4C 层通信的具体信息。

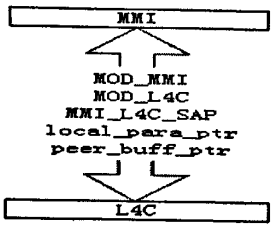


图 4.19 MMI 与 L4 的接口

按照 4.2.3 控制流的设计机制，在 MMI 和 L4C 分别设计添加一个外部消息队列，通过此消息队列来发送或接收消息。MMI_task 和 L4C_task 分别查询自己的队列，来实现实时交互。实现过程及函数设计如下。

从队列中读取或发送并接收信息，此时需要建 MMI_task 或 L4C_task 来操作，具体实现函数如下：

```
OslReadCircularQ(&Message);
OslWriteCircularQ(&ilm_ptr);
OslReceiveMsgExtQ(MMI_qid,&MMI_message);
OslMsgSendExtQueue(&Message);
```

其中 ilm_ptr 为消息的结构体 ilm_struct 的一个变量，该结构体的定义如下：

```
typedef struct ilm_struct {
    oslModuleType    oslSrcId; // 源模块 ID
    oslModuleType    oslDestId; // 目的模块 ID
    oslMsgType       oslSapId; // 服务接入点
    oslMsgType       olsMsgId; // 消息 ID
    oslParaType      *oslDataPtr; // 本地参数缓冲器
    oslPeerParaPtr   *oslPeerBuffer; // peer 缓冲器指针
  }ilm_struct;
```

接下来注册事件的函数，所设计的函数为：

```
SetProtocolEventHandler(FuncCB,msg_id);
```

MMI Framework 根据 msg_id 找到该注册函数，执行回调函数实现所要求的功能。

## 4.7 本章小结

一个良好的 MMI 软件架构，可以简化 MMI 软件的开发复杂度、增强 MMI 软件的健壮性、可移植性和可裁剪性，是确保 MMI 软件质量的重要因素之一。本章从嵌入式软件架构设计方法入手，按照服务把 MMI 进行系统分解为框架层、用户接口层和应用层三个层次，并对三个层次进行设计实现。同时，设计实现了 MMI 与系统交互的软件接口，确保 MMI 与 L4C 的有效通信。在实现过程中，充分利用了消息序列图来描述各个层次的消息驱动机制，有效的实现了软件设计中的控制流设计。

本章在 MMI 软件架构过程中，严格按照嵌入式软件架构的设计方法，并结合软件开发的概要设计理论来进行设计，对各类移动终端 MMI 软件架构的设计具有一定的参考意义。在实现时采用标准 C 语言并以面向对象的思想来编写软件，提高了软件的可移植性。

在 4.5 节已经交代过，应用层包含着移动终端提供给用户的各种功能，如电话本、短信息、通话记录、情景模式和显示设置等。由于各种子功能在设计与实现方法是一致的，故不对每个子功能进行详细叙述，在第五章中，将重点的介绍电话本软件模块的设计与实现。

# 第五章 电话本软件设计与实现

移动终端的 MMI 软件包括电话本、呼叫管理、短信息、基本人机界面和菜单等部分，某些移动终端还具备了闹钟、日历、字典和计算器等附加功能，这些应用软件也属于 MMI 软件的一部分。MMI 软件主要是实现与用户的交互，并完成用户的功能需求。电话本是典型的 MMI 软件，本章主要是对电话本软件(PHone Book software，PHB)进行设计与实现。在设计与实现过程中，重点突出了 MMI 三层架构之间的消息驱动运行机制。

## 5.1 电话本软件概要设计

电话本是移动终端不可缺少的一个基本功能模块之一，它提供了对电话号码的各项操作，包括对各种不同的电话号码的读出、添加、编辑和删除等功能；PHB 还可以支持对某一个具体电话号码的组织操作，比如固定拨号，紧急呼叫号码等。具体来讲，PHB 所支持的功能[31]如下：

(1) 从电话本中读取记录列表；

(2) 读取一个电话号码相关记录；

(3) 添加、更新或删除电话号码相关的一个记录；

(4) 通过用户名删除电话号码某一类型的一项记录，如最近拨出号码；

(5) 添加、更新或删除个人电话号码；

(6) 确定是否为紧急呼叫号码；

(7) 在当前拨打模式下，确定一个来电是否可转化为控制模式；

(8) 通过姓名快速检索电话本的记录；

(9) 通过电话号码快速检索电话本的记录。

在系统功能设计中，把以上功能进一步整理，归纳成姓名查找、电话添加、电话删除、电话复制和设置(来电群组、其它号码、紧急号码)五个子模块。每个模块完成各自特定的功能，共同组成功能齐全的电话本。

### 5.1.1 PHB 框架设计

根据上述对 PHB 的功能分析，PHB 模块设计成为两大模块：PHB 内核模块(PHB-Kernel)和 PHB 搜索引擎模块(PHB-Search-Engine)。PHB-Kernel 主要负责 PHB 各个子功能的上层功能，这些功能为上面所讲述的姓名查找、电话添加、电话删除、电话复制和设置子功能；PHB-Search-Engine 主要提供对 PHB-Kernel 的服务支持，比如对姓名查找的支持等。PHB 框架图设计如图 5.1 所示。

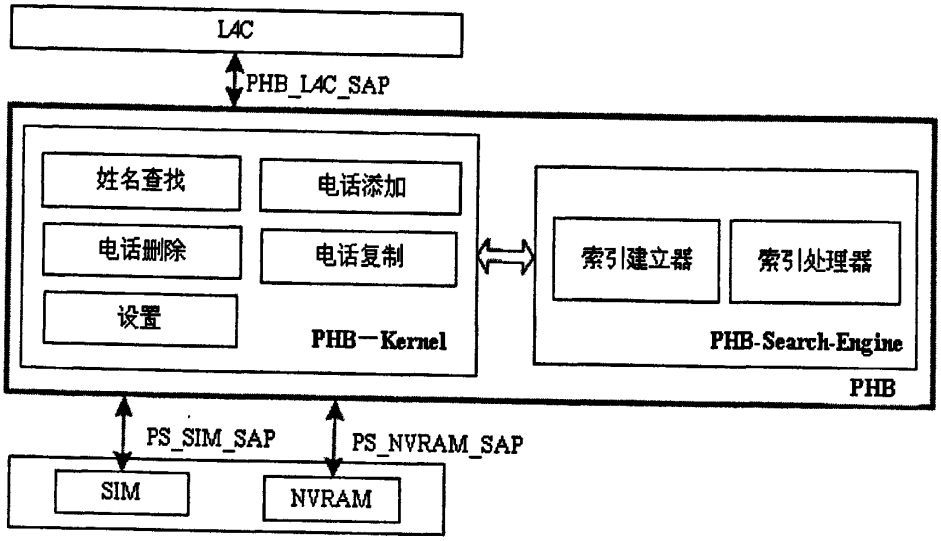


图 5.1 PHB 架构图

### 5.1.2 PHB 内核模块

由图 5.1 知 PHB-Kernel 包括姓名查找、电话添加、电话删除、电话复制和设置五个子功能模块。这样，PHB-Kernel 代替了 MMI 应用层负责管理电话号码的相关记录，MMI 应用层就可以把控制电话号码的相关功能省去，只负责运行显示等简单的功能。PHB-Kernel 还提供管理 PHB-Search-Engine 模块来建立电话号码或姓名的索引，支持快速查找功能。PHB-Kernel 提供与 L4C 的通信接口和与 SIM 和 NVRAM 通信的接口，通过消息驱动机制向 L4C 请求服务和管理存储在 SIM 和 NVRAM 中的电话号码数据。

PHB-Kernel 的执行流程是：在移动终端初始化阶段，L4C 根据 SIM 的初始化信息通知 PHB-Kernel 当前的操作模式(FDN/BDN)，接到该通知消息后，PHB-Kernel 开始扫描存储在 SIM 和 NVRAM 中的电话号码相关信息，然后操作 PHB-Search-Engine 建立电话号码或姓名的索引。当 MMI 需要操作某一个电话号码，PHB-Kernel 获取到数据并通过 L4C 发送给 MMI。为了加快检索速度，建立一个检索处理器，并在 MMI 中加入 cache。PHB-Kernel 五个子模块的功能如下：

(1) 姓名查找

进入 PHB 姓名查找界面，输入某一个姓名进行姓名查找。MMI 应用层向 PHB-Kernel 发送姓名查找请求信息，PHB-Kernel 接收信息并处理，把查找请求信息发送到 PHB-Search-Engine，PHB-Search-Engine 接收信息，根据输入姓名执行查找操作，若查找成功，返回查找姓名的电话号码等相关信息，若查找失败，弹出查找失败对话框。

(2) 电话添加

进入 PHB 电话添加界面，输入添加的电话号码，MMI 应用层把添加号码请

求信息发送到 PHB-Kernel，PHB-Kernel 接收该请求信息，调用 WriteValue( )函数，把电话号码等信息写入 SIM 或 NVRAM 中。操作成功，返回号码成功信息，若操作失败，弹出电话添加失败对话框。

(3) 电话删除

进入 PHB 电话删除界面，输入待删除的电话号码的用户姓名，执行姓名查找功能，查找到指定电话后，PHB-Kernel 确定信息是存放在 SIM 还是 NVRAM 中，执行 DeleteValue( )函数，删除在 SIM 或 NVRAM 中的电话信息。删除操作成功，返回电话删除成功信息，若删除失败，弹出电话删除失败对话框。

(4) 电话复制

进入 PHB 电话复制界面，输入待复制的电话号码的用户姓名，执行姓名查找功能，查找到指定电话号后，指定电话将从 SIM 复制到 NVRAM 或是从 NVRAM 复制到 SIM，PHB-Kernel 调用 CopyValue( )函数执行复制操作。复制成功，更新电话本信息并返回电话复制成功信息，若操作失败，弹出电话复制失败对话框。

(5) 设置

把除了以上四种功能之外的其它功能，包括来电群组、其它号码和紧急号码等，归入设置模块，实现电话本的管理功能。

### 5.1.3 PHB 搜索引擎模块

PHB-Search-Engine 提供了 5.2 节中功能列表中的最后两项功能，实现对 PHB-Kernel 的服务支持。它由两部分组成：索引建立器和检索处理器，结构图如图 5.1 中 PHB-Search-Engine 模块所示。索引建立器是建立和维持对内部检索结构的插入/删除操作。检索处理器主要用于对数据的检索操作，输出的结果是存储在 SIM 或 NVRAM 中的记录的索引。PHB 只把符合要求的数据的索引存入内存，而不是全部数据存入内存，这样可以使内存消耗最小化。为了对输入电话号码的完整信息的检索，PHB-Kernel 调用 PHB-Search-Engine 生成的索引，这样可以快速的获取所需信息的记录。

在 PHB-Search-Engine 设计中，索引技术主要应用于电话本的记录上(SIM/NVRAM)。在查找姓名和电话号码时，采用的检索技术有两种，分别是三元搜索树[32] (The Ternary Search Tree，TST)和哈希表(Hash Table)。三元搜索树是对字符串进行排序和查找的快速算法，特别适合用于搜索前缀字符串，速度比传统的排序与查找算法更快，效率也更高，因此选用 TST 作为姓名查找的算法。哈希表并不适合前缀字符串的查找，然而，通过观察电话号码的位数，因为地域的原因，致使电话号码有某些数字或数字组合是相同或相近的，根据此特性，选用了哈希函数作为电话号码的查找算法。

TST 结合了二元搜索树和数字搜索尝试算法的特性，在处理方面，它跟尝试算法一样，都是对字符进行比较，在空间效率上，具有二元搜索树的高效特性，尽管它每个节点具备了三个子节点。进行字符搜索时，从 TST 的根节点开始，如果搜索节点小于比较节点，则转向比较节点的左子树进行搜索；如果搜索节点大于比较节点，则转向比较节点的右子树进行搜索；如果搜索节点等于比较节点，则转向比较节点的中间子树搜索，然后处理搜索串里面的下一个搜索字符。

例如，图 5.2 为包含 12 个单词的平衡 TST[33]，左右子树用实线连接，中间子树用虚线来连接，每个输入字符在节点中显示。比如查找串“is”时，从根节点开始比较查找，因为第一个字符与根节点相等，故转向中间子树比较第二字符“s”，经过 2 次比较搜索成功；再比如对于“ax”的搜索，要经过 3 次与“a”的比较和 2 次与“x”的比较后判定该字符串不存在。

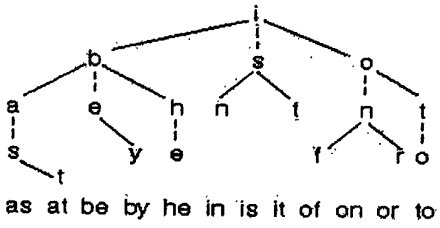


图 5.2 TST 实例

对 TST 的数据结构简单定义如下：

```
typedef struct tnode* Tptr;
typedef struct tnode {
    char splitchar;   //存储节点的值
    Tptr lokid,eqkid,hiked;   //三个子节点
}Tnode;
```

TST 搜索算法采用递归编写，核心代码如下：

```
int rsearch(Tptr p, char* s)
{
    if(!ip) return 0;
    if(*s < p->splitchar)
        return rsearch(p->lokid,s);
    else if (*s > p->splitchar)
        return rsearch(p->hiked,s);
    else {
            if (*s == 0) return 1;
            retrun rsearch(p->eqkid,++s);
}
```

## 5.2 电话本软件详细设计

有限状态机(FSM)[34]是一种具有离散输入输出系统的模型，它在任何时刻，都处于一个特定的状态，对于以消息驱动的程序设计，它是一种非常有用的设计模型。PHB 的有限状态机如图 5.3。

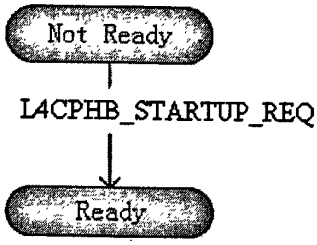


图 5.3 PHB 有限状态机

(1) Not Ready：在该状态下，PHB 处于未初始化状态，并等待来自 L4C 的消息 L4CPHB_STARTUP_REQ。一旦接收到 L4CPHB_STARTUP_REQ，重新设置 PHB 模块的状态，并再次执行启动程序。

(2) Ready：在该状态下，PHB 已完成了初始化工作，处于服务就绪状态。

(3) 状态变化描述：Not Ready→Ready。当 PHB 接收来自 L4C 的 L4CPHB_STARTUP_REQ 消息，PHB 完成自己的初始化并进入 Ready 状态。Ready 状态内部状态可分为 READ（读）、WRITE（写）、DELETE（删除）、SEARCH（查找）和 APPROVE（允许）五种状态，其中 L4_APV_RED_RSP 等为触发状态转化的信息，其 FSM 图如图 5.4 所示。

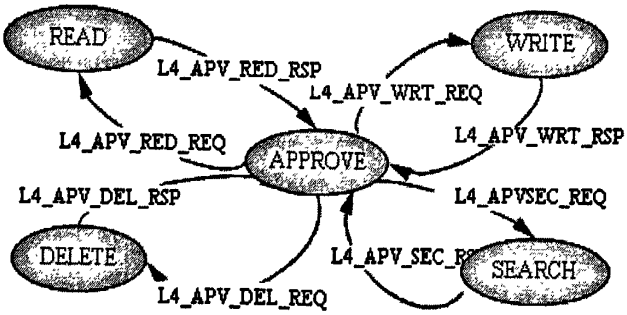


图 5.4 Ready 有限状态机

由 5.2 电话本软件的概要设计可知，PHB 包括姓名查找、电话添加、电话删除、电话复制和设置五个子功能模块。结合 Ready 状态的有限状态机，从技术实现角度可归纳出主要操作有以下四种：

(1) 初始化；

(2) 查找电话；

(3) 读取电话；

(4) 添加/删除/更新电话。

### 5.2.1 初始化

进入PHB时，进行初始化工作。L4C向PHB发送L4CPHB_STARTUP_REQ请求初始化信息，PHB向SIM/NVRAM发送NVRAM_EF_PHB_LID命令，进行各项参数设置，设置完成，由SIM/NVRAM返回NVRAM_PHB_LID信息到PHB，PHB收到信息，通过PhbGetSpeedDialPhbIndexReq( )函数调用PHB-Search-Engine模块进行号码索引操作，获取号码索引并调用PhbGetSpeedDialPhbIndexRsp( )将索引存入内存，操作完毕向L4C返回L4CPHB_STARTUP_RSP信息，告诉L4C初始化结束，至此，PHB完成初始化工作，进入就绪状态。

初始化的消息序列图如图5.5所示。

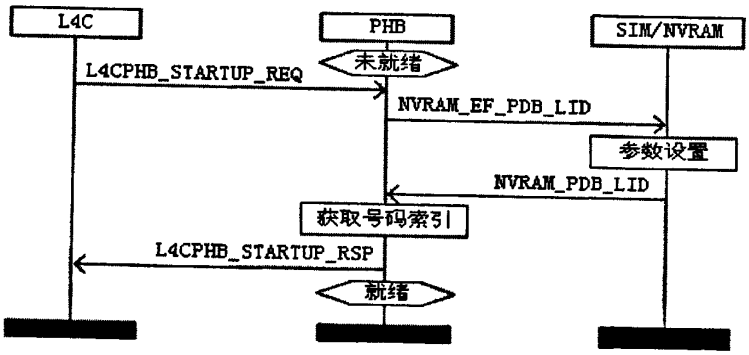


图 5.5 初始化

### 5.2.2 查找电话

PHB 的查找功能主要是由 PHB-Search-Engine 来执行实现的。L4C 向 PHB 发送 L4CPHB_SEARCH_REQ 信息，请求查找电话，PHB 处理消息，并向 SIM/NVRAM 发送 SIM_SER_REQ 或 NVRAM_SER_REQ 信息，调用 mmi_phb_highlight_search_nam( )进入 PHB-Search-Engine 模块，进行电话号码搜索，并返回查找的电话的记录，查找成功，SIM/NVRAM 向 PHB 返回 SIM_SER_RSP 或 NVRAM_SER_RSP 确认信息，PHB 获取信息，向 L4C 返回确认信息 L4CPHB_SEARCH_RSP。

查找电话的消息序列图如图 5.6 所示。

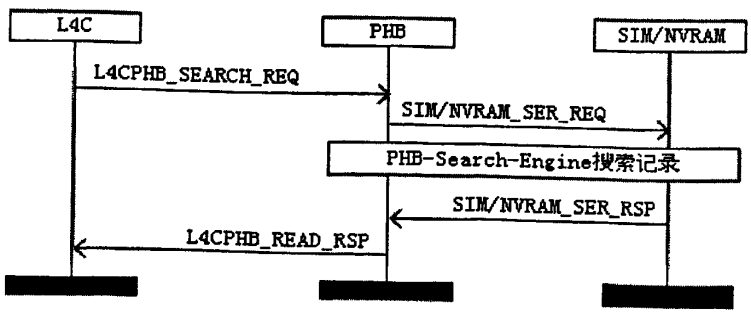


图 5.6 查找电话

### 5.2.3 读取电话记录

电话本的添加、查找和修改等多数功能都必须读取SIM/NVRAM的记录。读取记录时，L4C向PHB发送L4CPHB_READ_REQ信息请求读取电话记录，PHB接收信息并向SIM/NVRAM发送SIM_READ_REQ或NVRAM_READ_REQ信息，执行函数PhbGetDialPhbInfoRsp( )进入记录检索和读取操作，读取成功SIM或NVRM向PHB返回SIM_READ_REQ或NVRAM_READ_REQ信息，PHB收到SIM/NVRAM的确认信息后，获取号码索引并调用PhbGetSpeedDialPhbIndexRsp( )将索引存入内存，操作完毕，向L4C返回L4CPHB_READ_RSP确认信息，电话记录读取结束。

读取电话记录的消息序列图如图 5.7 所示。

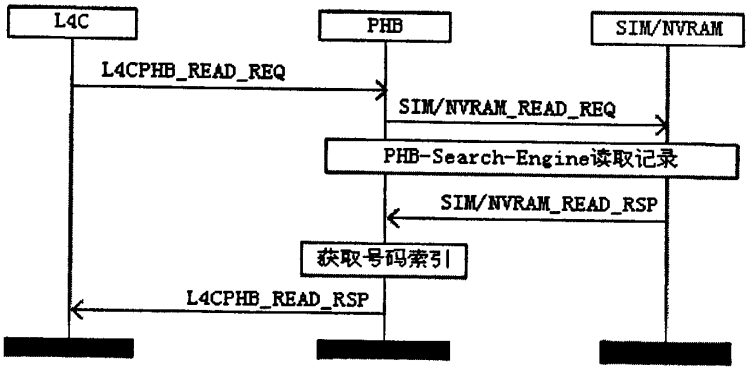


图 5.7 读取 SIM/NVRAM 的记录

### 5.2.4 添加/删除/更新电话

PHB 的各个子功能模块多处涉及到添加、删除和更新电话记录，该部分具有许多共同点，添加是把记录写入 SIM/NVRAM，删除是读出信息并把原来的 SIM/NVRAM 存储空间写入全 0 值，更新是读取记录并把新的记录写入 SIM/NVRAM。因此，从本质上来说，添加、删除和更新都是对 SIM/NVRAM 的写操作，故把这三个操作归纳为一个来设计实现。

执行操作时，L4C 向 PHB 发出请求信息 L4CPHB_WRITE_REQ 或 L4CPHB_DELETE_REQ，进行添加或删除操作请求，更新是对信息读取然后写入，所以更新也属于添加操作的内容。PHB 处理信息并向 SIM/NVRAM 发送写请求 SIM/NVRAM_WRITE_REQ，之后进入查找电话、读取记录并根据消息内容选择执行函数 MMI_phb_op_add_entry_req( )、 MMI_phb_op_delete_entry_req( )或 MMI_phb_op_updata_entry_req( )进行添加/删除/更新操作。操作成功，SIM/NVRAM 返回确认信息 SIM/NVMAM_WRITE_REQ，PHB 进行处理并向 L4C 返回确认信息 L4CPHB_WRITE_CNF 或 L4CPHB_DELETE_CNF，至此，操作完毕。其消息序列图如图 5.8 所示。

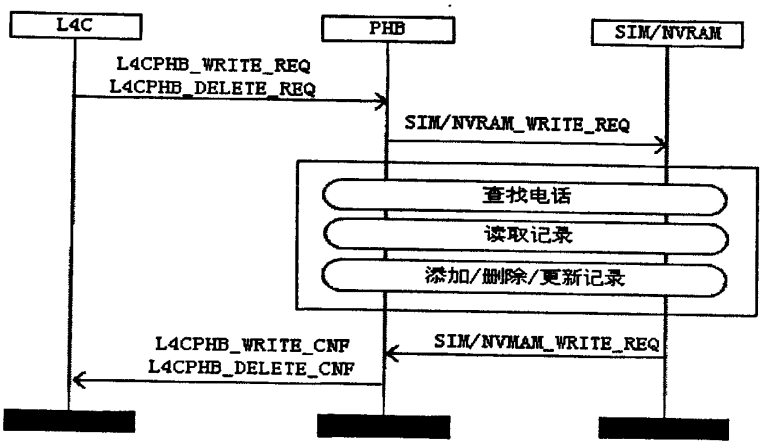


图 5.8 添加/删除/更新电话记录

前面已经分析了添加、删除和更新记录的相似性，这里仅给出删除某一项记录的详细设计流程，其它子功能的设计实现基本相同，在此不再深入详述。删除一项记录，其流程如下：

(1) 进入 PHB 选项菜单中选择删除功能；

(2) 发送待删除的记录的索引到 SIM/NVRAM；

(3) 从 MMI 队列中读取 SIM/NVRAM 返回的信息；

(4) 判断返回的结果是否为真，若 TURE，删除记录并更新电话本；若 FLASE，不作删除操作；

(5) 操作结束，弹出对话框显示操作成功或失败提示信息。

按此执行流程，设计的数据流程图如图 5.9 所示。

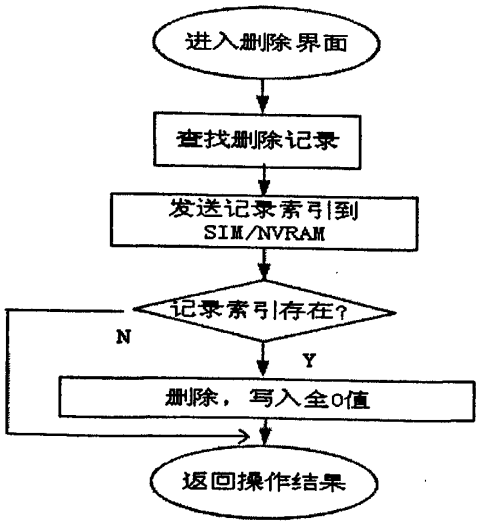


图 5.9 编辑流程图

接下来从 MMI 角度来设计该删除操作的消息序列，如下：

(1) 高亮删除电话记录；

(2) MMI 执行 MMI_phb_op_delete_entry_req( )，进入电话本删除记录函数，

该函数向 L4 发送 PRT_PHB_DEL_ENTRY_REQ，L4 接收到该消息并判断返回 PRT_PHB_DEL_ENTRY_RSP；

(3) MMI 从 MMI 队列中读取消息，启动 MMI_phb_op_delete_entry_rsp()函数进入删除操作；

(4) 利用 PhbUpdateSpeedKey( )执行删除操作；在对 SIM 或 NVRAM 操作时，使用第四章已经介绍过的函数 ReadValue( )/WriteValue( )或 WriteRecord( )/ ReadRecord( )；

(5) 当查找表出现无效的入口时，调用 MMI_phb_lookup_table_sort( )对查找表排序并更新电话本索引列表；

(6) 操作成功，返回成功提示，反之，则弹出错误信息。

删除记录函数的核心代码如下：

```
void MMI_phb_op_delete_entry_req(void)
{
    ……;  //首先是局部变量的初始化
    //判断待删除的记录的存储位置
    storage = (store_index >= MAX_PB_PHONE_ENTRIES) ? MMI_SIM : MMI_NVRAM;
    //各项属性设置
    myMsgPtr->storage = storage;
    myMsgPtr->del_all = MMI_FALSE;
    myMsgPtr->no_data = 1;
    myMsgPtr->type = MMI_PHB_PHONEBOOK;
    myMsgPtr->index = (storage == MMI_SIM) ? (store_index + 1) - MAX_PB_PHONE_ENTRIES : (store_index + 1); //这里的 index 表示的是 record_index，该处理只对 MMI_PHB_PHONEBOOK(ADN)类型有效，对于其它类型(BDN,FDN,MSISDN...)则仍然采用其 index 为参数
    //设置消息参数，这里充分体现了系统的消息驱动机制
    Message.oslSrcId = MOD_MMI;  //消息的发出地为 MMI
    Message.oslDestId = MOD_L4C;  //消息的目的地为 L4C
    Message.oslMsgId = PRT_PHB_DEL_ENTRY_REQ;  //消息类型是执行删除请求，该宏定义值为 1493
    Message.oslDataPtr = (oslParaType*) myMsgPtr;  //设置数据指针，用于存储记录信息
    //注册回调函数，用于处理来自协议栈的消息。当删除记录被执行并收到 L4C 发送来的删除消息 PRT_PHB_DEL_ENTRY_RSP，系统执行回调函数
```

```
    SetProtocolEventHandler(MMI_phb_op_delete_entry_rsp,
PRT_PHB_DEL_ENTRY_RSP);
    OslMsgSendExtQueue(&Message);  //把消息发送到 L4C 的外部队列
    DeleteNScrId(SCR_PBOOK_ENTRY_OPTIONS);  //删除该历史记录
    ……;  //最后其它扫尾工作
}
```

## 5.3 软件调试及实现

在程序设计完成后，需要进行编译，生成目标代码并烧写到移动终端的存储器上，然后进行调试。调试结束，把修改后的源代码重新烧写进移动终端的存储器中，即可实现 MMI 软件的各项操作功能，具体过程可参考 2.2.5 节。

### 5.3.1 调试环境介绍

在 Windows XP 环境下，采用 ADSv1.2 编译器编译程序，编译完成后生成二进制目标文件，用 FlashTool 工具把目标文件烧写进存储器中，然后用 Catcher 工具获取 log 进行调试。

(1) ADSv1.2

ADSv1.2(ARM Developer Suite v1.2)[35]是 ARM 公司的 ARM 核处理器集成开发工具，集成了汇编、C、C++编译器和调试器，编译效率高，提供了功能强大的系统库，支持软件调式、JTAG 仿真调试及硬件调试。运行环境如图 5.10 所示。

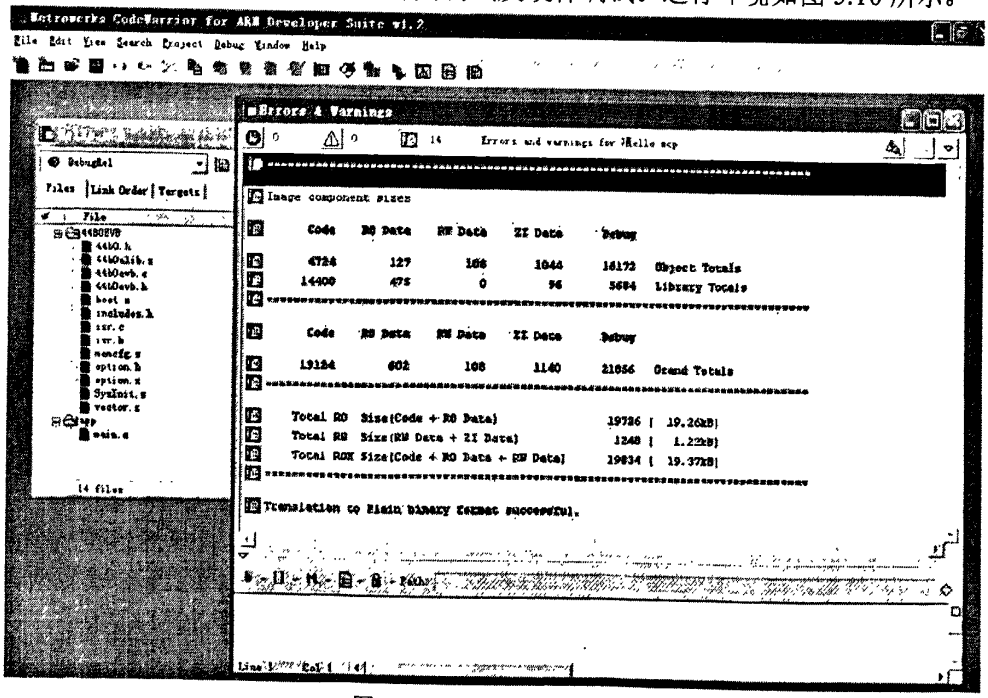


图 5.10 ADSv1.2 运行环境

(2) FlashTool

FlashTool[36]是一个 PC 端的烧写工具，可以把二进制的目标源文件烧写到目标移动终端的 ROM 存储器中。FlashTool 的使用方法见参考文献[36]，其运行环境如图 5.11 所示。

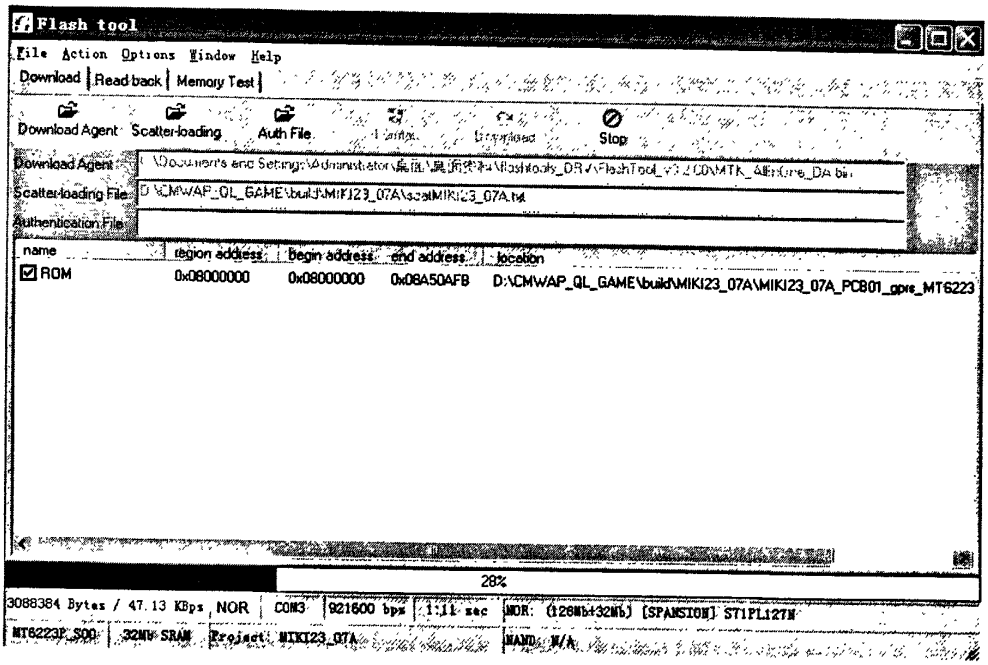


图 5. 11 FlashTool 运行环境

(3) Catcher

Catcher[37]是用于测试 GSM/GPRS 移动设备的 PC 端应用测试软件，可以提取 GSM/GPRS 移动设备代码中的 log 信息和调试信息，它的运行主界面如图 5.12 所示。它是我们项目的主要调试工具，具体用法这不展开叙述，请查阅文献[37]。

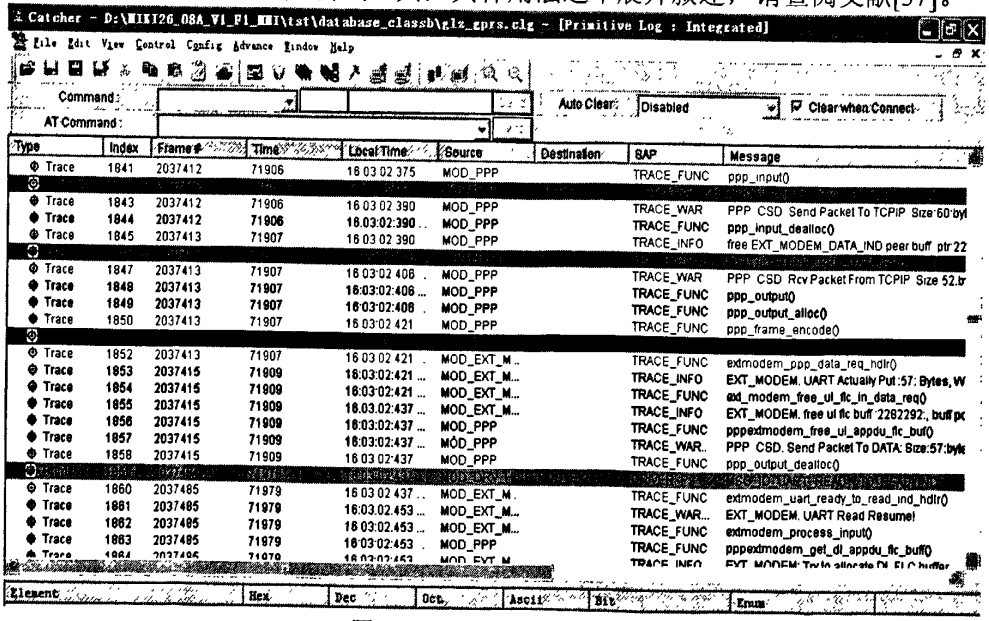


图 5.12 catcher 运行主界面

### 5.3.2 实现结果

经过调试以后，把正确源文件编译成目标文件，再下载到所设计的 GSM 移动终端中。开机运行效果良好。以添加/删除电话号码为例，运行结果如图 5.13 所示。

(1) 添加电话号码

进入电话本界面(a)，选择“添加号码”功能，进入“存储位置选择”界面(b)，进行存储位置的选择：SIM 和本机(即 NVRAM)；选择“至 SIM 卡”，进入“姓名”界面(c)，选择“姓名”选项，进入“姓名”的编辑界面(d)，在这里录入所要添加的姓名，录入后选择“选项”中的完成，进入“电话号码”录入界面(e)；录入电话号码后按左软键“确定”，即可完成电话号码的添加功能。

(2) 删除电话号码

同样进入电话本界面(a)，选择“删除”选项，进入删除界面(f)，有三个待选项，其中“从 SIM 卡”表示删除 SIM 卡上的所有电话号码记录，“从本机”表示删除 NVRAM 上的所有电话号码的记录，“逐条删除”表示选择某一待定的电话号码或姓名单独删去；我们选择“从 SIM 卡”，进入删除界面(g)，为了确保移动终端用户的安全，删除功能要求输入密码；输入正确密码后确定，弹出“完成”提示对话框(h)，表示删除操作成功。

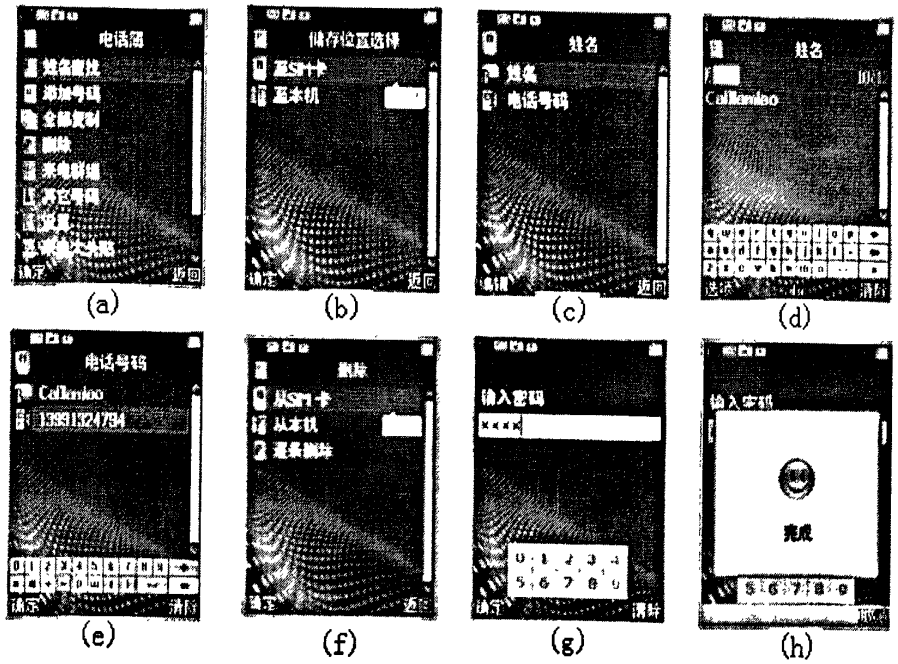


图 5.13 实现结果图

## 5.4 本章小结

电话本软件是移动终端 MMI 软件的典型代表，本章对电话本进行设计与实现。电话本软件的开发是建立在第四章所设计的 MMI 软件架构之上，其 MMI 三层结构在设计中得到了充分的体现。在设计与实现过程中，通过有限状态机和消息序列图等工具，把消息驱动机制运贯穿于整个开发过程中。在编程实现上，采用 ANSI C 语言，增强了软件模块的可移植性。

电话本软件实现了姓名查找、电话添加、电话删除、电话复制和设置五个子功能模块，满足了移动终端的基本要求。鉴于电话本多处涉及到查找操作，在实现技术上采用了查找效率较高的三元搜索树技术，提高了电话本的运行效率。所设计的电话本软件移植到前文所设计的 MMI 软件架构中，兼容效果良好。

电话本是移动终端的基本功能之一，本章不失一般性的讲述了电话本的开发，其实现方法可以应用到 MMI 软件的其它软件中，诸如短信息、通话记录、情景模式、显示设置等。对于增强型移动终端软件的开发，如闹钟、日历、字典和计算器等，同样具有一定的参考价值。

# 第六章 结束语

移动终端系统的开发是一项复杂的工程，它涉及到移动通信技术、嵌入式技术以及软件工程等多项技术。本文主要论述了移动终端 MMI 软件的开发，它是与用户交互最直接的一层。MMI 软件质量的好坏，不仅影响整个移动终端系统的运行效率和功能实现，也影响到它的实际应用和市场价值。

本课题是某通信技术有限公司的研发内容之一，任务是开发一款具有较高效率的移动终端 MMI 软件。在嵌入式软件设计方法学的指导下，对移动终端软件进行了分析与研究，重点设计了一种通用性较强的移动终端 MMI 软件架构，并在该架构上设计并实现了电话本软件。在程序实现上，以面向对象程序设计为思想，采用消息驱动机制，并选择标准 C 语言编写程序，以增添软件的可移植性。课题所设计的 MMI 软件在 GSM 移动终端系统上运行良好，并已经商业化。现将本论文的主要工作总结如下：

(1) 对移动终端系统及相关技术进行综述，重点对 GSM 协议和嵌入式技术进行了较为详尽的探讨；

(2) 分析了移动终端系统组成原理，并设计了一款具有代表性的 GSM 移动终端系统；

(3) 对嵌入式软件架构方法学进行了综述，并以此为指导，设计出一种具有较强通用性的移动终端 MMI 软件架构并编程实现；

(4) 在已设计的 MMI 软件架构的基础上，设计并实现了电话本软件，包括姓名查找、电话添加、电话删除、电话复制和设置五个子功能；

(5) 对三元搜索树(TST)进行了探讨，并在电话本软件开发中引入了 TST 技术，提高了电话本的运行效率。

课题所研究的 MMI 软件是在 GSM 通信网中实现的，然而在设计过程中，在 MMI 软件与协议栈之间添加了中间层 L4 层的概念，这样确保了 MMI 软件的独立性和可移植性，经过较小的修改，便可移植到 3G 通信网的移动终端中。在接下来的工作中，主要有两点：

(1) 把该软件移植到 3G 移动终端中进行调试，并加以完善，以满足移动通信行业的发展和市场的迫切需求；

(2) 继续深入对移动终端软件，特别是 MMI 软件的研究，增强我国在移动终端开发中的实力，争取研发出独立自主的品牌技术。

随着我国 3G 进程的推进，可以预测，移动终端将进入一个新的快速增长时期。研发出具有高性能、实用型的移动终端软件，必将推动我国通信产业的发展，提高我国在世界通信领域的竞争力。

# 致 谢

三年的研究生生活即将过去，在本论文完成之际，我由衷地感谢指导、教诲、关心以及帮助过我的老师、同学和亲人。

首先衷心感谢我的导师牛海军教授对我的悉心指导和严格要求。在整个研究生的学习生活阶段，牛老师一直给予我无私的指导与帮助，使得我能有机会在实践中学习提高。在做课题研究期间，牛老师也一直关心课题的进展，并提出了很多意见和建议。同时，牛老师严谨的治学态度、谦和的处事风格、精益求精的科研精神也时刻影响和激励着我。在此，我再次向牛老师表达深深的敬意和谢意！

同时，感谢计算机外部设备研究所实验室为我提供了良好的学习和工作环境，感谢实验室的所有兄弟姐妹，特别是要感谢李树文、吴娜、赵世强、林琳、吴善鹏、张昕等同学，他们在我的课题研究、论文撰写和生活中给予了很大的帮助。

感谢项目组的各位同事，特别是王西君、郭刘占、鲁新护、杨伟清等工程师，他们在我实习期间提供了很多的帮助，使我的课题研究得以顺利的进行。

感谢一如既往地支持我奋发向上的家人，父母的关心和支持，帮我度过了一个个难关，没有他们的支持和鼓励，我也不可能取得今天的成绩。

最后，感谢所有关心、帮助过我的老师、同学、亲人和朋友们，感谢为评阅本论文而付出辛勤劳动的各位老师。

# 参考文献

[1] Jing Zhang, Xiong-jian Liang.3G in China:Environment and Prospect.Management of Engineering and Technology, Portland International Center.2007:2988-2992

[2] Wada H, Fukushima H.Mobile computing on wireless telecommunication networks.Universal Personal Communications, 1996. Record, 1996 5th IEEE International Conference.1996:778-782

[3] Bo Li, Dongliang Xie, Shiduan Cheng, et al.Recent advances on TD-SCDMA in China Communications Magazine, IEEE.2005 43(1):30 -37

[4] Patachaianand R., Sandrasegaran K.System-Level Modeling and Simulation of Uplink WCDMA.Information Technology: New Generations, Fifth International Conference.2008:1071-1076

[5] ETSI.Digital cellular telecommunications system (Phase 2+): GSM Public Land Mobile Network (PLMN) (GSM01.02).1996

[6] ETSI.Digital cellular telecommunications system (Phase 2+): Physical layer on the radio path (GSM05.01).1996

[7] ETSI.Digital cellular telecommunications system (Phase 2+): DataLink (DL) layer: General aspects (GSM04.05).1997

[8] ETSI.Digital cellular telecommunications system (Phase 2+): Mobile Station-Base Station System interface: Data Link (DL) layer Specification (GSM04.06).1997

[9] ETSI. Digital cellular telecommunications system (Phase 2+): Mobile Radio Interface Layer3 (GSM02.01).1996

[10] Micheal J.Pont.Embedded C.北京:中国电力出版社.2003:33-35

[11] Damm Werner.Embedded system development for automotive applications:Trends and challenges.IEEE International Conference on Embedded Software.2006:1-13

[12] Zha Xuan F, Sriram Ram D.Feature technology and ontology for embedded system design and development.Proceedings of 2006 ASME International Design Engineering Technical Conferences.2006:14-23

[13] 赵刚荣.CDMA 手机外围驱动设计[硕士学位论文].西安:西安电子科技大学. 2007

[14] 马忠梅,李善平等.ARM&Linux 嵌入式系统教程.北京:北京航空航天大学出版社.2004:36-48

[15] Anisetti.Advanced localization of mobile terminal.2007 International Symposium on Communications and Information Technologies Proceedings. 2007:1071-1076

[16] Young Kar-Keung D, Ou Yong Quan, Cai Lun Huai, et al.Real time embedded control system development for wireless mobile platforms.IEEE International Symposium on Industrial Electronics.2008:2022-2027

[17] Accelerated Technology, Inc.Accelerated Technology Embedded Systems Division of Mentor Graphics Corporation. 2002

[18] 宋宝华.Linux 设备驱动程序开发详解.北京:人民邮电出版社.2008:74-274

[19] ETSI.Digital cellular telecommunications system (Phase 2+): Man-Machine Interface (MMI) of the Mobile Station (MS) (GSM 02.30).1996

[20] 舒风笛,毋国庆,李明树.面向嵌入式实时软件的需求规约语言及检测方法.软件学报.2004, 15(11):1595-1606

[21] 刘良松,刘海岩等.软件工程.西安:西安电子科技大学出版社.2004:21-32

[22] Shari Lawrence Pfleeger.Software Engineering Theory and Practice.北京:清华大学出版社.2003:170-221

[23] 张海.软件工程导论.北京:清华大学出版社.1999:54-107

[24] Raj Kamal.Embeded Systems Architechture,Programming and Design.北京:清华大学出版社.2005:13-301

[25] 康一梅.嵌入式软件设计.北京:机械工业出版社.2008:96-100

[26] 童蕾.事件驱动的消息发布/订阅研究和实现[硕士学位论文].北京:中国科学院研究生院(软件研究所).2005

[27] ITU-T.ITU-T Recommendation Z.120. Message Sequence Charts (MSC'96).1996

[28] 顾增辉,王淑仙.一种 GSM/GPRS 手机 MMI 的体系结构及其实现.上海:华东师范大学学报(自然科学版).2003, 12(4):42-48

[29] Axel-Tobias.Schreiner. Object-Oriented Programming with ANSI-C.Hanser Fachbuch.1994:111-125

[30] Charles Petzold. Windows 程序设计.北京:北京大学出版社.2004:37-63

[31] Media Tek Inc.High Level Design Specification of Phone Book.2005

[32] Jon Bentley, Robert Sedgewick.Fast Algorithms for Sorting and Searching Strings. Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.New Orleans.1997:360-369

[33] Jon Bentley, Bob Sedgewick.Ternary Search Trees.Dr. Dobb's Journal.1998

[34] Miro Samek Ph.D.嵌入式系统的微模块化程序设计——实用状态图的 C/C++ 实现.北京:北京航空航天大学出版社.2004:40-66

[35] 李驹光.ARM 应用系统开发详解.北京:清华大学出版社.2004:225-305

[36] Media Tek Inc., Application Notes of FlashTool V2.6.2005

[37] Media Tek Inc., User Manual of Cacher V3.3.2004

# 在读期间的研究成果

[1] 蔡佳苗,牛海军,吴娜. 基于GSM手机MMI软件的设计与实现.西安电子科技大学学术年会.2008.已录用

[2] 吴娜,牛海军,蔡佳苗. 基于自动标记特征点的快速人脸识别算法.西安电子科技大学学术年会.2008.已录用